İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

Automating Farming Operations Using Robotic Technologies

Submitted to the Graduate School of Natural and Applied Sciences in partial fulfillment of the requirements for the degree of

Master of Science / Doctor of Philosophy

in Robotics Engineering

by

Basheer Altawil

ORCID 0000-0002-5716-7587

January ,2023

This is to certify that we have read the thesis **Automating farming operations using robotic technologies** submitted by **Basheer Altawil**, and it has been judged to be successful, in scope and in quality, at the defense exam and accepted by our jury as a MASTER'S THESIS.

APPROVED BY:

Advisor:

Asst. Prof. Dr. Fatih Cemal CAN …….
İzmir Kâtip Çelebi University

Committee Members:

Prof. Dr. Levent ÇETİN ……
İzmir Kâtip Çelebi University

Asst. Prof. Dr. Murat AKDAĞ …….
Dokuz Eylül University

Date of Defense: January 23, 2023

Declaration of Authorship

I, **Basheer Altawil**, declare that this thesis titled **Automating Farming Operations Using Robotic Technologies** and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for the Master's / Doctoral degree at this university.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. This thesis is entirely my own work, with the exception of such quotations.
- I have acknowledged all major sources of assistance.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Date: 23.01.2023

Automating Farming Operations Using Robotic Technologies

Abstract

Recent years have seen a rise in interest in agricultural robots, both academically and commercially. This is since agricultural robotics tackles important concerns such as seasonal labor shortages, during harvest, as well as the growing concern over ecologically friendly techniques. On the one hand, several distinct agricultural robots have already been created for a variety of purposes, with varying degrees of success, including monitoring, spraying, harvesting, transport, etc.

It is more practical to have a simple environment that can include all of the characteristics and components required for controlling the entire system, including middleware, hardware, and software. This ecosystem can assist users, developers, and investors in growing their aspirations and businesses to meet both market demands and novel approaches to agricultural issues.

This study identifies the state-of-the-art and provides a new methodology for multitasking in performing agriculture operations. In addition, agriculture automation is unfeasible and unprofitable as a result of these difficulties, we used Robot Operating System (ROS) to provide potential solutions to these issues, and the potential is increased when paired with other open-source technologies, such as android or IoT.

For individuals working in the robotics industry, the open-source ROS framework has a lot of potential. It is simple to use and simple to install. Developers may be able to engage with it and create new products that offer insights into the requirements of people and the future.

In this paper, we design a four degrees of freedom (4DoF) robotic arm with a different mechanism that has a different configuration. It has two grippers to be easily integrated with ROS, and it can work synchronously with other hardware, sensors, cameras, and agricultural machines concerning farming operations.

The robotic arm can be easily integrated with mobile platforms where it can autonomously work in agricultural space without facing any challenges. In conclusion, we believe that this new configuration will open a door for agricultural tasks to be easily automated and achieved using robotic technologies.

Keywords: ROS, IoT, Mechanism, Robot Manipulator, Mobile Platforms

Robotik Teknolojileri Kullanarak Tarım Operasyonlarını Otomatikleştirme

Öz

Son yıllarda, hem akademik hem de ticari olarak tarım robotlarına olan ilgide bir artış görüldü. Bunun nedeni, tarımsal robot teknolojisinin, hasat sırasındaki mevsimsel işçi sıkıntısı gibi önemli endişeleri ve ayrıca çevre dostu tekniklere yönelik artan endişeyi ele almasıdır. Bir yandan, izleme, ilaçlama, hasat, nakliye vb. dahil olmak üzere çeşitli amaçlar için farklı derecelerde başarı gösteren farklı tarım robotu yaratılmıştır.

Ara yazılım, donanım ve yazılım dahil olmak üzere tüm sistemi kontrol etmek için gereken tüm özellikleri ve bileşenleri içerebilen basit bir ortama sahip olmak daha pratiktir. Bu ekosistem, kullanıcıların, geliştiricilerin ve yatırımcıların hem pazar taleplerini hem de tarım sorunlarına yeni yaklaşımları karşılama isteklerini ve işletmelerini büyütmelerine yardımcı olabilir.

Bu çalışma, en son teknolojiyi tanımlar ve tarım operasyonlarının gerçekleştirilmesinde çoklu görev için yeni bir metodoloji sağlar. Ayrıca, tarım otomasyonu bu zorlukların bir sonucu olarak uygulanamaz ve kârsızdır, bu sorunlara potansiyel çözümler sağlamak için Robot İşletim Sistemini (ROS) kullandık ve android, IoT veya benzeri diğer açık kaynak teknolojilerle eşleştirildiğinde potansiyel artar.

Robotik endüstrisinde çalışan bireyler için açık kaynaklı ROS çerçevesi çok fazla potansiyele sahiptir. Kullanımı ve kurulumu basittir. Geliştiriciler, onunla etkileşim kurabilir ve insanların ve geleceğin gereksinimlerine ilişkin içgörüler sunan yeni ürünler oluşturabilir.

Bu tezde, farklı bir konfigürasyona sahip farklı bir mekanizmaya sahip dört serbestlik dereceli (4DoF) bir robotik kol tasarlıyoruz. ROS'a kolayca entegre edilebilecek iki adet kıskaca sahiptir ve diğer donanımlar, sensörler, kameralar ve tarım makineleri ile senkron çalışabilmektedir. Robotik kol, tarama alanında herhangi bir zorlukla karşılaşmadan otonom olarak çalışabileceği mobil platformlara kolayca entegre

edilebilir. Sonuç olarak, bu yeni konfigürasyonun, tarım işlerinin robotik teknolojiler kullanılarak kolayca otomatikleştirilmesi ve gerçekleştirilmesi için bir kapı açılacağına inanıyoruz.

Anahtar Kelimeler: ROS, IoT, Mekanizma, Robot Manipülatörü, Mobil Platformlar.

To my mother, father and family,

To my dear friends,

Acknowledgment

I'd want to express my gratitude and appreciation to my supervisor, Asst. Prof. Dr. Fatih Cemal CAN, who helped me with all aspect of this study preparation, including my academic and social lives. I am deeply indebted for his generous support throughout the entire process of writing this thesis. I would especially like to extend my gratitude to all of my professors, mentors, and other faculty members who gave me with priceless advice that enabled me to successfully complete academic journey in the university.

I want to express my gratitude to my genuine, kind friends, who have become my second family while I've been living overseas. They have helped me overcome the difficulties that come with living apart from family and have held my hand when I've fallen in my life's path. I want to express my gratitude to my loving parents, my brother Mohammed, and my sisters who supported me with their unwavering tolerance and excellent Dua—the most potent tool for bolstering me when I was weak—until I reached the point where I am now.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

HTM	Homogeneous transformation matrices
DH	Denavit Hartenberg
IK	Inverse Kinematics
CFDH	Coordinate-Fixed Denavit–Hartenberg Method
(PoE)	Product of Exponentials representation
URDF	Universal Robot Description Format
XML	eXtensible Markup Language
ROS	Robotic Operating System
GHC	Green House chambers
DoF	Degrees of freedom
Rmn	The rotation matrix between axis m to n
${}^{n}_{m}T$	The transformation matrix between axis m to n

List of Symbols

$\boldsymbol{I_i}$	Robot Link Inertia [kg.m^2]
$\boldsymbol{V}$	Linear Velocity [m/s]
$\boldsymbol{\omega}$	Angular Velocity[rad/s]
$\boldsymbol{\tau}$	Robot Joints Torque [kg.cm]
F_e	Force Exerted on Robot Joints [N]
f_r	Friction force [N]
m_i	Manipulator Link Mass[kg]
a_i	Manipulator Link Length[m]

Chapter 1

1 Introduction

In recent years, farming activities have become highly expensive, demanding, and difficult to perform. One of the emerging technologies that will change, revolutionize, and address challenges and difficulties in the agriculture sector is robot farmers. To maximize the efficiency and production of any agriculture system, it is now necessary to combine agriculture and technology tools, making it simpler for farmers to produce a large amount of food that can fit the market needs. As a result, by 2050, the total cost of agriculture and technology is expected to be roughly $240 billion [1].

The future of robotic agriculture lies in agricultural robots' solutions, especially because soil and field crops are being severely damaged by climate change. Agricultural robotics makes precision agriculture possible and more productive by automating the slow and repetitive tasks that farmers perform during work, such as planting seeds and harvesting crops, weeding, and farm management [2]. Agri-robotics can handle variations in crop types, soil moisture content, growth follows, and farm scales including field and sub-field, as well as greenhouse chambers (GHC), glasshouse, vertical farms, and hydroponic close and open systems [2]–[4].

1.1. Motivation

Harvesting, planting, irrigation, and weeding are some of the most interesting tasks that everybody working in the agriculture field tries to implement robotic technology on them. Therefore, modern farms are supposed to provide higher-quality crops at reduced costs in a sustainable manner that is less reliant on manpower. Digital farming and site-specific precision management are two examples of prospective solutions for those tasks' implementation. Furthermore, Reactions to this expectation, which is influenced by factors other than the sensor technology, but it is the continual collection of field data that is only possible with technology. Agricultural mobile robot manipulators, data processing, and autonomous robots, when used properly, can make this possible. Scientists, farmers, and growers in the agricultural industry are all dealing with the same

issues (Figure 1.1) [4], [5]. Their main goal is Producing more food on less land in a sustainable manner is a challenge that is a suitable strategy to address the needs of the 9.8 billion people expected by 2050[2].

Moreover, it is becoming more difficult to manage all of these technologies, and it's becoming more complex to cover all of the details with limited resources of robotics software and hardware. As a result, using the Robotic Operating System (ROS) is preferred, as it can simplify all complex operations in one package.

Figure 1.1: Harvesting Robotic Arm, Source [3]

With this robotic platform, it is easy and simple to be able to spray targets, pluck foliage, and harvest specific fruits (detects the fruit, determines its ripeness, moves towards the fruit, grasps it, and softly detaches it). This research will focus on the development of the ***AIBOMECH Agrobot*** harvesting robot manipulator in Crops that we are planning to use in farming and agriculture for different [3], [6].

1.2. Literature Review

Robot arms have long been used in agriculture, particularly in broad-acre applications, horticulture, and livestock operations. Modern agriculture is highly industrialized, with hundreds of different equipment and machinery [3]. This opens the door for robotic applications, as computer science advances promise better productivity and safety. At a time when farming efficiency has plateaued as a result of the effective integration of bigger irrigation systems and crop diversification, such technology is desired. As a result, the purpose of this thesis project was to serve the agricultural industry using a robotic

arm that can be easily mounted on a mobile platform. It is also one of several examples of multi-DOF robotic arms utilized in the agriculture business.

A basic articulated SCARA spatial manipulator with full 4-DOF will be used to harvest crops and can be readily placed on a wheeled robot. Several agricultural arms in the literature, on the other hand, are attached to a base and are frequently neither robustly constructed nor propelled by workspace-dense manipulation. The ones that are installed are having their programming platform and it is difficult to be robustly integrated with other systems. However, our ***AIBOMECH Agrobot*** Crop arm was developed for high precision and for multi-purposes that will be implemented especially in the agriculture field[7].

In robotic development, simulation approaches have played a vital role in the rapid verification of new prototypes, algorithms, and/or applications of motion in robotic fields, as well as a system performance optimization. A robotic simulator may simulate the motion of robots and all items in a virtual work envelope without relying on an actual complete hardware system, saving both money and time.

This has sparked a surge in interest in robotic simulation in recent years and became a must case to build any robotic system (Figure 1.2)[8]. Modeling and rendering of a robot and its environment have been thoroughly researched in their most basic form to be implemented in the environment around us.

Robot manufacturers such as ABB, KUKA, Staubli, Universal robot, and others have developed simulation suits for their robots for KUKA Work Visual, which allow robotics programs to be safely written, debugged off-line, and tested on an actual robot, but third-party hardware and services, such as sensor-based actions or physical engines, are hard to integrate with more complex applications like these[8]**.**

There are many software simulators for robot manipulators and differential mobile robots, and the one we'll use is Robot Operation System (ROS). It's a robotic middleware that provides software frameworks for robotic software development. Many open-source implementations of common robotics functions and algorithms can be found in ROS. Packages provided in ROS deployments arrange these open-source implementations, while others are developed by people and released through code-sharing sites like GitHub. ROS has also been the most extensively utilized platform for robotic research in recent years [9].

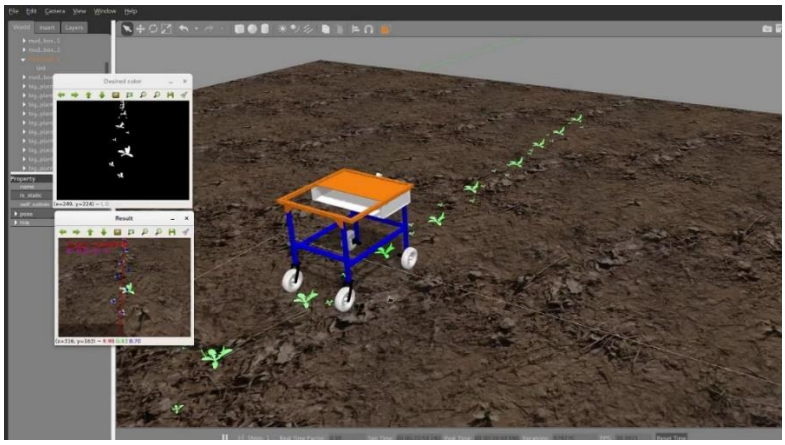


Figure 1.2: Gazebo Simulator ,Source [8]

In this thesis, we offer a rapid and modular architecture that incorporates the complete system design, modeling, kinematics, Jacobian, control, and visualization configuration process. A case study of an arm manipulation tasks simulation in an RVIZ, Gazebo in integration with MOVEIT.They can display and demonstrates the methods that we will use for robot tasks like path planning, picking, and place for harvesting purposes.

Using the ROS MOVEIT and Navigation Stack Technical is one of the most valuable and interesting topics for ROS developers. In ROS, MOVEIT offers primary manipulation capabilities. It has the potential to develop and incorporate both library capabilities and a vibrant community. As a result, it has manipulation, motion planning, control, and mobile manipulation capabilities. Furthermore, having a robust community of users and developers aids in the maintenance and extension of new applications, allowing for greater work flexibility and accuracy [10], [11].

The architecture of MOVEIT is seen in Figure 6. Its principal node is named MOVE GROUP. It's made to be light, with a wide range of capabilities and integrated kinematics, motion planning, and perception. When contrasted to Arm Navigation, it employs a plugin-based approach (inspired by ROS) that significantly enhances extensibility. Users may easily add and share features such as a new pick and place implementation or motion planning that has been organized using a plugin architecture.

MOVEIT's ability to employ plugins is a crucial feature that distinguishes it from Arm Navigation [12].We can say that the heart of MOVEIT is the MOVE_GROUP node (Figure 1.3) [8], which functions as an integrator of the robot's many components and offers actions/services based on the user's demands. Looking at the architecture, it's evident that the MOVE_GROUP node collects robot data in the form of topics and

services, such as the robot's point cloud, joint state, and transforms (TFs). It gathers robot kinematics data from the parameter server, including the Unified Robot Description Format (URDF), the Semantic Robot Description Format (SRDF), and configuration files. While creating a MoveIt! package for our robot, the SRDF file, and configuration files are created [13].

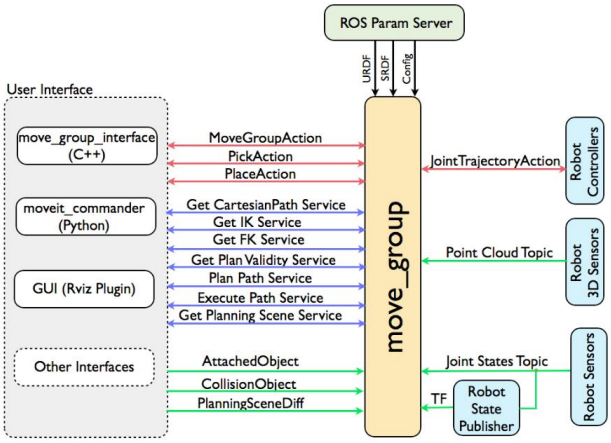


Figure 1.3: MOVEIT MOVE_GROUP node ,Source [13]

Introducing our own articulated SCARA manipulator mechanism, see Figure 1.4) will provide various capabilities and benefits to farm robots that will be readily integrated with the ROS community. The main benefit is that it eliminates the term "barrier to entry," which is frequently used in the context of robotics software engineering and refers to the amount of time, effort, and experience that a new user must have to integrate software components with any brand of a farming robotics system.

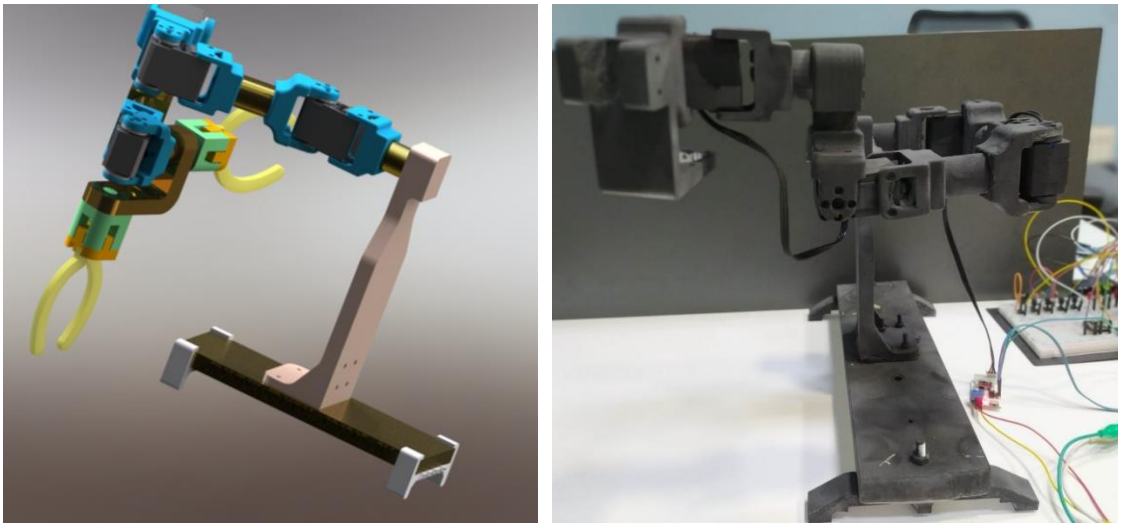


Figure 1. 4: 4DoF Serial manipulator CAD Design and assembly

For example, creating a virtual model of the robot's geometry and kinematics, customizing configuration and design files, choosing the fastest algorithmic approach for a specific application, and finding the best parameters for various algorithms that can be extremely simple with a very useful workspace and mechanism manipulation that will allow the farmer to easily employ them in their tasks are some of the advantages that will be added to robot farming technologies[14].

1.3. Agricultural Measurements and Robotization

A set of tools with comparable interfaces, sensors, actuators, and similar power and communication requirements were designed to illustrate the modularity and feasibility of any agricultural robotic system. This thesis' instrumentation aims to take advantage of approaches for collecting various farm data, such as environmental parameters like temperature and air quality at the same time using robot arm harvesting tasks for different types of crops. These techniques, which have been presented in previous research, are introduced here, and each is packed into the modular equipment shown in this project, see Figure 1.5[7].

In the past, spectroscopic approaches such as detecting the chlorophyll content via light reflectance were frequently employed to assess plant health. Several distinct forms of vegetation indices, such as the Normalized Difference Vegetation Index (NDVI) or the Modified Simple Ratio (MSR), have been used to quantify leaf cover or crop development within a confined area with differing degrees of performance.
While the majority of these experiments and tails focus on visible light, infrared and near-infrared light have also been utilized to investigate plant cell wall compositional changes[5]. This technique is completely non-destructive, making it simple to examine not only the impacts of stress on plants but also soil health indicators like water retention and organic carbon concentration. Measuring Ph, electrical conductivity, soil properties, and soil characteristics must be included to get good productivity for farmers. These all measurements are being run using automation technologies in farming. However, this toolkit is far from comprehensive, but it includes areas of a farm that may require regular monitoring and repair. The difference in tooling, on the other hand, poses a distinct problem in that the shape of the parts, and form factors of each device modify the overall

kinematics of the robotic arm that will be used to approach to the tasks needed to be performed during robot working.

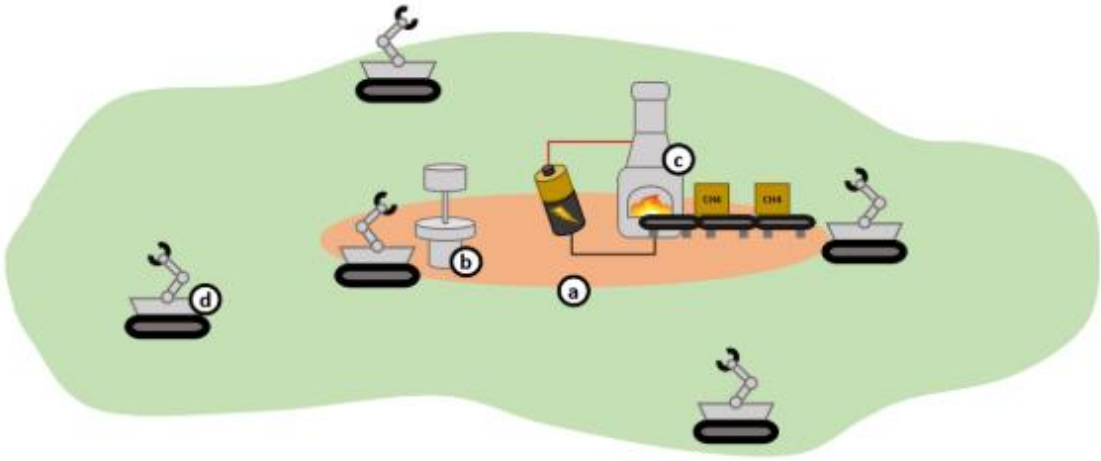


Figure 1.5: Farms Robotization ,Source [7]

The changing kinematics, together with the challenges of managing a serial linkage installed on a flexible base, create intriguing control concerns with many solutions [7]. Therefore, it became important to integrate and deploy robotic technology in farming to mitigate all of these issues and make the output of the targeted regions more dynamic and efficient.

Chapter 2

2 Kinematics and Kinetic Analysis

In this section, the kinematics analysis, dynamics analysis, and mathematical computations will be present to achieve the next parts which will depend on this chapter.

2.1. Denavit–Hardenberg Based Analysis

Analysis for robotics, particularly robotic manipulators, is extremely tough, and difficult. Therefore, many well-known scientists are striving to minimize the problems and make them more adaptable for those who are interested in robotics science. The four Denavit–Hartenberg parameters (DH parameters) are related to a specific standard for attaching reference frames to the links of a spatial kinematic chain, or robot manipulator. This convention was established in 1955 by Jacques Denavit and Richard Hardenberg to standardize the coordinate frames for spatial links. In 1981, Richard Paul demonstrated its utility in the kinematic analysis of robotic systems [15]. Even though several conventions for attaching reference frames have been created, the Denavit–Hardenberg convention is still a common way for engineers to simplify their analysis and get closer to their solutions. DH principles parameters can help engineers do and perform forward kinematics (FW), inverse kinematics (IK), Jacobian, and dynamics analysis easily and concerning the tasks that are needed to be implemented.

2.1.1. Denavit-Hartenberg Axis

In general, serial manipulators consist of several links, connected with some of the common joints. Those joints might be revolute, or prismatic joints. To relate the end effector of the manipulator with the fixed frame of the base of the manipulator we have to use some mathematical calculations which we call position analysis [16]. The scientists Denavit and Hardenberg come up with some principles to accurately solve this problem and it is lying on link coordination and orientation. When describing a link, the

relative position of the two axes in space can be described by two parameters, **link length,** and **link twist**. Two parameters may be used to define joints which are the **offset length** is the separation between each link along the axis of the joint, and the rotation of one link concerning the next about the joint axis is known as **the joint angle** [17].

The principle is based on geometric operations and dual vector algebra to process and determine the relative transformation matrices, from which it is computed the Standard Denavit- Hartenberg (DH) parameters ($\mathbf{a_n, \alpha_n, d_n, \theta_n}$)[14].The axis and parameters (Figure 2.1) [13] [14]. The link axis and parameters will be clarified like the following:

- The Z_n axis which is aligned with joint (**n**)
- The X_{n-1} axis aligned along the common perpendicular of two joint axes of the link n-1 points to joint n.
- The Y_n axis is determined by the right-hand role cartesian coordinate system.

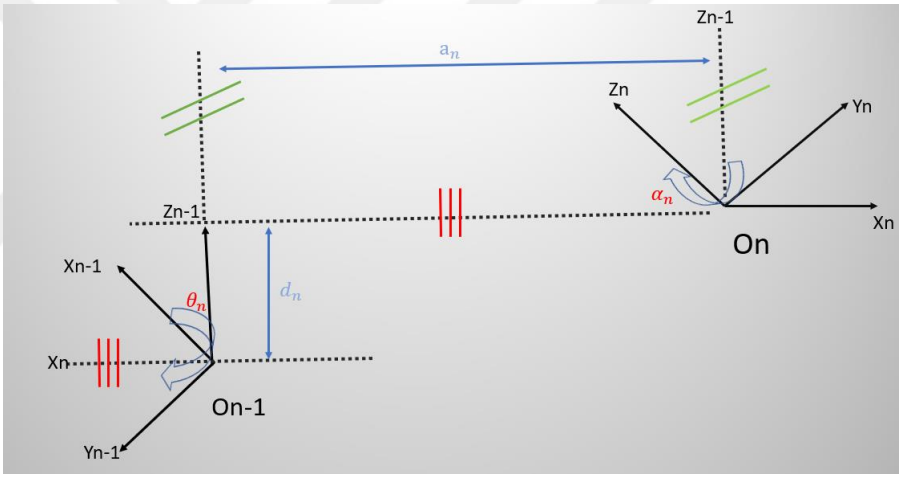


Figure 2.1: DH link parameters

However, the link parameters will be defined like the following and they will be filled into the DH table to be used in HTM analysis[16].

- Parameter a_i which is the offset distance between the origin of the n-1 frame to the origin of n along the X_n axis direction.
- Parameter d_n which is the translation distance between X_{n-1} to X_n along the Z_{n-1} axis direction.
- Parameter α_n which is the twist angle between the Z_{n-1} axis to Z_n around the positive X_n axis.

- Parameter θ_n which is the joint angle and can be measured from X_{n-1} to X_n around Z_{n-1}.

2.1.2. Homogeneous Transformation Matrices (HTM)

To apply DH principles to the frames of any kind of robotic arms, we have to follow some sequencing steps frequently to come up with the exact result [18], and the steps will be like the following.

Rotation around the Z_{n-1} axis, $\mathrm{Rot}_{z(n-1)}\ (\Theta_n)$,see equation 2.1.

$$Rot_{z(n-1)}\ (\Theta_n) = \begin{pmatrix} C(\Theta_n) & -S(\Theta_n) & 0 & 0 \\ S(\Theta_n) & C(\Theta_n) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.1)$$

Translation along Z_{n-1} axis, $\mathrm{Trans}_{z(n-1)}\ (d_n)$,see equation 2.2.

$$\mathrm{Trans}_{z(n-1)}\ (d_n) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.2)$$

Translation along the X_n axis, $\mathrm{Trans}_{x(n)}\ (a_n)$,see equation 2.3.

$$\mathrm{Trans}_{x(n)}\ (a_n) = \begin{pmatrix} 1 & 0 & 0 & a_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.3)$$

Rotation around the X_n axis, $\mathrm{Rot}_{x\,(n)}\ (\alpha_n)$,see equation 2.4.

$$\mathrm{Rot}_{x\,(n)}\ (\alpha_n) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & C(\alpha_n) & -S(\alpha_n) & 0 \\ 0 & S(\alpha_n) & C(\alpha_n) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.4)$$

Regarding the steps that we applied to the DH axis and the matrices shown in equations 2.1 to 2.4 respectively, we could obtain the arm kinematic labeling in Figure 2.2.

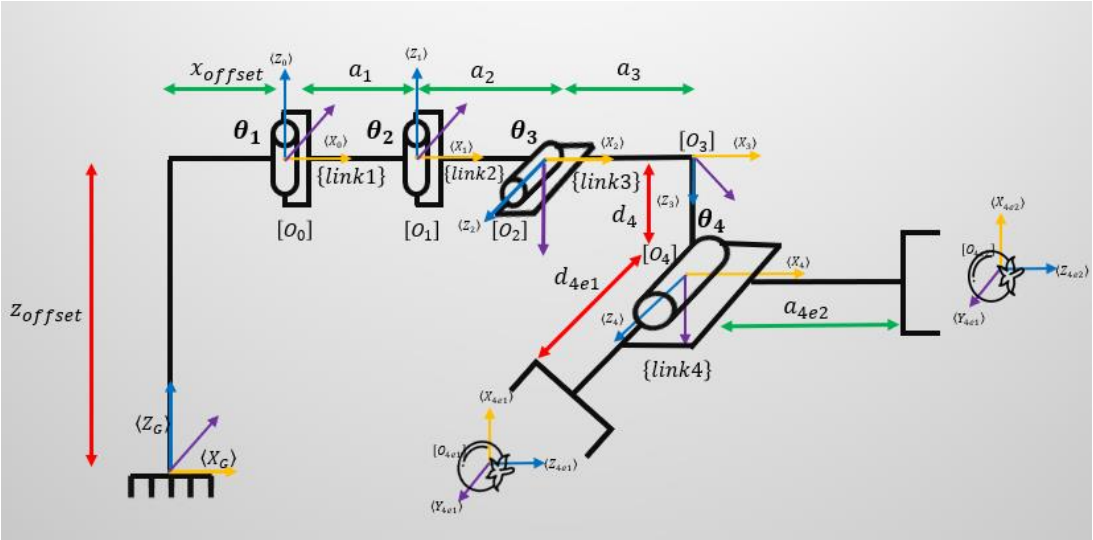


Figure 2. 2 AIBOMECH Agrobot Arm *Kinematic labeling*

In addition to that, the DH convention table is abstracted too and it will consist of four variables which we mentioned before, two variables are used for rotation, and two variables for translation or displacement. The table number of Rows is equal to the number of ***Frames – 1***, the number of Columns = ***4***, two columns for rotation, and two columns for translation [19].

The set of parameters twist angle $\boldsymbol{\alpha}_n$,offset distance $\mathbf{a}_{\boldsymbol{n}}$,translation distance $\mathbf{d}_{\boldsymbol{n}}$,and joint angle $\boldsymbol{\Theta}_{\boldsymbol{n}}$ respectively will be presented in Table 2.1.This table notations are taken from the representations explained in Figure 2.1 and every row is giving us four HTMs' that will be multiplied in sequence to give us general transformation matric to be used for the specified axis, the kinematic representation for DH parameters can be converted to another representation easily by following some constraints.

For example, there are other representations, such as the Screw theory, in which the axes are modeled as normalized twists with the origin determined by a directional vector and a linear velocity vector, they call it the Product of Exponentials (PoE) representation. Recently, the use of the Universal Robot Description Format (URDF), an XML (eXtensible Markup Language) file format to describe the kinematics and dynamics of robots in a tree structure, has expanded as a result of the Robot Operating System (ROS) where every robotic simulation platform in nowadays simulation might be implemented in MATLAB, Gazebo, and CoppeliaSim in addition to ROS[20]. Therefore, we can convert between these representations easily following the constraints specified by the scientists who discovered them.

Table 2.1: Denavit-Hardenberg classical Convention Table

Link, n	a_n	α_n	dn	Θ_n	Joint rotation limits (Degree)
1	a_1	0	0	Θ_1	0-170
2	a_2	$-\frac{\pi}{2}$	0	Θ_2	0-170
3	a_3	$-\frac{\pi}{2}$	0	Θ_3	0-170
4	0	$-\frac{\pi}{2}$	d_4	0	
4_{e1}	0	0	d_{4e1}	Θ_4	0-170
4_{e2}	a_{4e1}	0	0	Θ_4	0-170

As a result of that, doing kinematics analysis in a sufficient way, can guarantee the developers to implement those analysis in ROS to be performed with high productivity and clever solutions for the industry and the nature.

2.1.3. Robot arm matrices formation and forward kinematics

We can easily create the HTM by vectorially multiplying the four matrices described in equations 2.1 to 2.4 using the formulation Eq. 2.5 after creating the matrices for every singular frame[16]. The matric that we obtain is shown in Eq.2.6 and it will be used as general HTM for the axis to help us analyze and do computational calculations for the robot arm.

The transformation matrix that enables transforming the position and orientation of one joint to another is represented by the parameter ${}^{n-1}_{n}T$ in Eq.2.6, and it can provide a mathematical definition of it. HTM provides the general transformation relation for the sixth joint of robot arm. It can relate and match the end-effector frame's transformation matrix with respect to the inertial basis frame that fixed to the ground or mobile robot. The first stage, which is illustrated in Eq.2.6, is to determine the position of the end-effector as a function of joint angles following the creation of DH parameters[2], [21]. This is the result of the vector multiplication that we did in the previous stages and it will be used for all axis of the robot arm.

$$= \mathrm{Rot}_{z(n-1)}(\Theta_n).\mathrm{Trans}_{z(n-1)}(d_n).\ \mathrm{Trans}_{x(n)}(a_n).\ \mathrm{Rot}_{x(n)}(\alpha_n) \quad {}^{n-1}_{n}T \qquad (2.5)$$

$${}^{n-1}_{n}T = \begin{pmatrix} C(\Theta_n) & -C(\alpha_n)S(\Theta_n) & S(\alpha_n)S(\Theta_n) & a_n C(\Theta_n) \\ S(\Theta_n) & C(\alpha_n)C(\Theta_n) & -S(\alpha_n)C(\Theta_n) & a_n S(\Theta_n) \\ 0 & S(\alpha_n) & C(\alpha_n) & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (2.6)$$

After obtaining the general formulation for transformation matric HTM for the manipulator axis, we apply it to the frame starting from frame one, up to frame four [22] as shown in Figure 2.2. Furthermore, the manipulator has two end effectors so that the extension from frame 5 will have two orientations for two different grippers.

The manipulator base is fixed to the ground, and we suppose it to have an offset on the Z axis, which we call it $\boldsymbol{O_{ffset}}$, and the offset on the X axis, which we call the $\boldsymbol{O_{ffset}}$ as shown in figure 2.2. In kinematic labeling, in total, it has four revolute joints, seven transformation frames, and four links with one base link that is fixed to the ground with two end effectors. The origin of every singular frame is defined with $[O_1]$, $[O_2]$, $[O_3]$, $[O_4]$, $[O_{4e1}]$, and $[O_{4e2}]$ starting from revolute joint1, up to both grippers respectively.

The links of the manipulator are defined as $\{\boldsymbol{link_n}\}$,n=1,2,3,4. The axis Z, X, and Y are represented as $\langle Z_0\rangle, \langle X_0\rangle, \langle Y_0\rangle$ for frame one,$\langle Z_1\rangle, \langle X_1\rangle, \langle Y_1\rangle$ for frame two, $\langle Z_2\rangle, \langle X_2\rangle, \langle Y_2\rangle$ for frame three,$\langle Z_3\rangle, \langle X_3\rangle, \langle Y_3\rangle$ for frame four, $\langle Z_4\rangle, \langle X_4\rangle, \langle Y_4\rangle$ for frame five, $\langle Z_{4e1}\rangle, \langle X_{4e1}\rangle, \langle Y_{4e1}\rangle$ for the end effector one frame, $\langle Z_{4e2}\rangle, \langle X_{4e2}\rangle, \langle Y_{4e2}\rangle$ for end effector two frames. As shown in figure 2.2, table 2.1.

Frame 1 has a translation on the X axis, rotation around the Z axis, frame two has a translation on the X axis, rotation around the Z axis, and rotation around the X axis, frame three has a translation on the X axis, rotation around the Z axis, and rotation around the X axis, frame four has translation along X axis and rotation around X axis too, frame five has translation along the Z axis, rotation around an axis, frame six has translation along an axis, rotation around the Z axis.

Table 2.2: HTM for link-1(frame 0)

	Translation matrices	Rotation matrices
Z Axis	*No translation along Z axis*	$Rot_{z(0)}(\Theta_1) = \begin{pmatrix} C(\Theta_1) & -S(\Theta_1) & 0 & 0 \\ S(\Theta_1) & C(\Theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
X Axis	$\text{Trans}_{z(1)}(d_1) = \begin{pmatrix} 1 & 0 & 0 & a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	No rotation around X-Axis

As shown in Table 2.2 the process done in the previous steps, we apply them to frame 0, using Eq.2.5. By substituting n with 1, we obtain a general formulation like it shown in Eq.2.7.

$$ {}^{0}_{1}T = \text{Rot}_{z(0)}(\Theta_1).\text{Trans}_{z(0)}(d_1).\ \text{Trans}_{x(1)}(a_1).\ \text{Rot}_{x(1)}(\alpha_1) \qquad (2.7) $$

After multiplying the matrices in Table 2.2, we arrive at the general form for HTM, which is depicted in Eq. 2.6 for axis one and will later be utilized to produce the general matrices that will readily match and precisely relate the end effector of the robot arm with the base frame like it shown on Eq2.8.

$$ {}^{0}_{1}T = \begin{pmatrix} C(\Theta_1) & -S(\Theta_1) & 0 & a_1C(\Theta_1) \\ S(\Theta_1) & C(\Theta_1) & 0 & a_1S(\Theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (2.8) $$

As shown in Table 2.3 the process done in the previous steps, we apply them to frame 1, using Eq.2.5. By substituting n with 2, we obtain a general formulation like it shown in Eq.2.9.

Table 2.3: HTM for link-2(frame-1)

Translation matrices		Rotation matrices
Z Axis	***No translation along Z axis***	$Rot_{z(1)}(\Theta_2) = \begin{pmatrix} C(\Theta_2) & -S(\Theta_2) & 0 & 0 \\ S(\Theta_2) & C(\Theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
X Axis	$\text{Trans}_{x(2)}(a_2) = \begin{pmatrix} 1 & 0 & 0 & a_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	$\text{Rot}_{x\,(2)}\,(\alpha_2) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

$$ {}^{1}_{2}T = \text{Rot}_{z(1)}\,(\Theta_2).\text{Trans}_{z(1)}\,(d_2).\ \text{Trans}_{x(2)}\,(a_2).\ \text{Rot}_{x\,(2)}\,(\alpha_2) \qquad (2.9) $$

After multiplying the matrices in Table 2.3, we come up with the general form for HTM, which is depicted in Eq.2.9 for axis two and will later be utilized to produce the general matrices that will readily match and precisely relate the end effector of the robot arm with the base frame like it shown on Eq2.10.

$$ {}^{0}_{2}T = {}^{0}_{1}T.{}^{1}_{2}T = \begin{pmatrix} C(\Theta_{12}) & 0 & -S(\Theta_{12}) & a_1C(\Theta_1) + a_2C(\Theta_{12}) \\ S(\Theta_{12}) & 0 & C(\Theta_{12}) & a_1S(\Theta_1) + a_2S(\Theta_{12}) \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (2.10) $$

As shown in Table:2.4 the process done in the previous steps, we apply them to frame 2, using Eq.2.5. By substituting n with 3, we obtain a general formulation like it shown in Eq.2.11.

Table 2.4: HTM for link-3(frame2)

Translation matrices		Rotation matrices
Z Axis	***No translation along Z axis***	$Rot_{z(2)}(\Theta_3) = \begin{pmatrix} C(\Theta_3) & -S(\Theta_3) & 0 & 0 \\ S(\Theta_3) & C(\Theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
X Axis	$\text{Trans}_{x(3)}(a_3) = \begin{pmatrix} 1 & 0 & 0 & a_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	$\text{Rot}_{x\,(3)}\,(\alpha_3) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

$$ {}^{2}_{3}T = \mathrm{Rot}_{z(2)}(\Theta_3).\mathrm{Trans}_{z(2)}(d_3).\mathrm{Trans}_{x(3)}(a_3).\ \mathrm{Rot}_{x(3)}(\alpha_3) \qquad (2.11) $$

We generate new parameters which are A1, A2, A3, A4, A5, and A6 that take expressions into them to minimize the size of the matrix like it shown in equations group1.

After multiplying the matrices in Table 2.4, we come up with the general form for HTM, which is depicted in Eq.2.11 for axis three and will later be utilized to produce the general matrices that will readily match and precisely relate the end effector of the robot arm with the base frame like it shown on Eq2.12.

$$ \left.\begin{aligned} A1 &= C(\Theta_3)C(\Theta_{12}) \\ A2 &= C(\Theta_3)S(\Theta_{12}) \\ A3 &= -S(\Theta_3)C(\Theta_{12}) \\ A4 &= -S(\Theta_3)S(\Theta_{12}) \\ A5 &= a_1C(\Theta_1) + a_2C(\Theta_{12}) + a_3C(\Theta_3)C(\Theta_{12}) \\ A6 &= a_1S(\Theta_1) + a_2S(\Theta_{12}) + a_3C(\Theta_3)S(\Theta_{12}) \end{aligned}\right\} \text{Equations group 1} $$

$$ {}^{0}_{3}T = {}^{0}_{2}T.{}^{2}_{3}T = \begin{pmatrix} A1 & S(\Theta_{12}) & A3 & A5 \\ A2 & -C(\Theta_{12}) & A4 & A6 \\ -S(\Theta_3) & 0 & -C(\Theta_3) & -a_3S(\Theta_3) \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (2.12) $$

As it is shown in Table:2.5 the process done in the previous steps, we apply them to frame 3, using Eq.2.5. By substituting n with 4, we obtain a general formulation like it shown in Eq.2.13.

$$ {}^{3}_{4}T = \mathrm{Rot}_{z(3)}(\Theta_4).\mathrm{Trans}_{z(3)}(d_4).\ \mathrm{Trans}_{x(4)}(a_4).\ \mathrm{Rot}_{x(4)}(\alpha_4) \qquad (2.13) $$

Table 2.5: HTM for link-3(frame3)

	Translation matrices	**Rotation matrices**
Z Axis	$\mathrm{Trans}_{x(3)}(a_4) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	No rotation round z axis
X Axis	No translation along X axis	$\mathrm{Rot}_{x(4)}(\alpha_4) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

After multiplying the matrices in Table 2.5, we come up with the general form for HTM, which is depicted in Eq.2.13 for axis four and will later be utilized to produce the general matrices that will readily match and precisely relate the end effector of the robot arm with the base frame like it shown on Eq2.14. We generate new parameters which are B1, B2, B3, B4, B5, B6 and B7 that take expressions into them to minimize the size of the matrix like it shown in equations group2.

$$\left.\begin{aligned} B1 &= C(\Theta_3)C(\Theta_{12}) \\ B2 &= C(\Theta_3)S(\Theta_{12}) \\ B3 &= S(\Theta_3)C(\Theta_{12}) \\ B4 &= S(\Theta_3)S(\Theta_{12}) \\ B5 &= a_1C(\Theta_1) + a_2C(\Theta_{12}) + a_3C(\Theta_3)C(\Theta_{12}) - d_4S(\Theta_3)C(\Theta_{12}) \\ B6 &= a_1S(\Theta_1) + a_2S(\Theta_{12}) + a_3C(\Theta_3)S(\Theta_{12}) - d_4S(\Theta_3)S(\Theta_{12}) \\ B7 &= -a_3S(\Theta_3) - d4C(\Theta_3) \end{aligned}\right\} \text{Equations group 2}$$

Table 2.6:HTM for link-4 (frame4_{e1}) (End effector-1)

	Translation matrices	**Rotation matrices**
Z Axis	$\text{Trans}_{z(4)}(d_{4e1}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{4e1} \\ 0 & 0 & 0 & 1 \end{pmatrix}$	$Rot_{z(4)}(\Theta_4) = \begin{pmatrix} C(\Theta_4) & -S(\Theta_4) & 0 & 0 \\ S(\Theta_4) & C(\Theta_4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
X Axis	No translation along the X axis	No rotation around X axis

$$ {}^0_4T = {}^0_3T\,{}^3_4T = \begin{pmatrix} B1 & B3 & S(\Theta_{12}) & B5 \\ B2 & B4 & -C(\Theta_{12}) & B6 \\ -S(\Theta_3) & C(\Theta_3) & 0 & B7 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (2.\ 14)$$

As shown in Table:2.6 the process done in the previous steps, we apply them to frame 4, using Eq.2.5. By substituting n with 4_{e1}, we obtain a general formulation like it shown in Eq.2.15 which is the general formula for end effector 1.

$$ {}^4_{4\,e1}T = \text{Rot}_{z(4)}(\Theta_4).\text{Trans}_{z(4)}(d_{4e1}).\ \text{Trans}_{x(4e1)}(a_{4e1}).\ \text{Rot}_{x\,(4e1)}(\alpha_{4e1}) \qquad (2.\ 15)$$

After multiplying the matrices in Table 2.6, we come up with the general form for HTM, which is depicted in Eq.2.15 for the axis 4_{e1} and will later be utilized to produce the

general matrices that will readily match and precisely relate the end effector 1 of the robot arm with the base frame like it shown on Eq2.16. We generate new parameters which are Q1, Q2, Q3, Q4, Q5, Q6 and Q7 that take expressions into them to minimize the size of the matrix like it shown in equations group3.

$$\left.\begin{array}{c} Q1=C(\Theta_{12})C(\Theta_{3_4}) \\ Q2 = C(\Theta_{34})S(\Theta_{12}) \\ Q3=C(\Theta_{12})S(\Theta_{34}) \\ Q4=S(\Theta_{34})S(\Theta_{12}) \\ Q5=a_1C(\Theta_1) + a_2C(\Theta_{12}) + a_3C(\Theta_3)C(\Theta_{12}) - d_4S(\Theta_3)C(\Theta_{12}) + d_{4e1}S(\Theta_{12}) \\ Q6=a_1S(\Theta_1) + a_2S(\Theta_{12}) + a_3C(\Theta_3)S(\Theta_{12}) - d_4S(\Theta_3)S(\Theta_{12}) - d_{4e1}C(\Theta_{12}) \\ Q7=a_3S(\Theta_3) - d_4C(\Theta_3) \end{array}\right\} \text{Equations group 3}$$

$$ {}^{0}_{4e1}T = {}^{0}_{4}T\,{}^{4}_{4e1}T = \begin{pmatrix} \overbrace{\begin{matrix} Q1 & Q3 & S(\Theta_{12}) \\ Q2 & Q4 & -C(\Theta_{12}) \\ -S(\Theta_{34}) & C(\Theta_{34}) & 0 \\ 0 & 0 & 0 \end{matrix}}^{R} & \overbrace{\begin{matrix} Q5 \\ Q6 \\ Q7 \\ 1 \end{matrix}}^{T} \end{pmatrix} \qquad (2.\ 16)$$

As seen in Eq. 2.16, we have 4X4 metrics that describe the forward kinematics and final transformation metrics for the first end effector. The rotation part of this metric is represented by the red shape with the R sample, and the translation part is represented by the blue shape with the T sample. The fourth column in matrix 2.16 with first row in the translation section represents translation in the X direction, the second row represents translation in the Y direction, and the third row represents translation in the Z direction.

Table 2.7:HTM for for link-4 (frame4$_{e2}$) (End effector-2)

	Translation matrices	**Rotation matrices**
Z Axis	No translation along the Z axis	$Rot_{z(4)}(\Theta_5) = \begin{pmatrix} C(\Theta_4) & -S(\Theta_4) & 0 & 0 \\ S(\Theta_4) & C(\Theta_4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
X Axis	$\text{Trans}_{x(3)}(a_4)=\begin{pmatrix} 1 & 0 & 0 & a_{4e2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	No rotation around the X axis

As shown in Table:2.7 the process done in the previous steps, we apply them to frame 4, using Eq.2.5. By substituting n with 4$_{e2}$, we obtain a general formulation like it shown in Eq.2.17 which is the general formula for end effector 2.

$$ {}^{4}_{4e2}T = \mathrm{Rot}_{z(4)}(\Theta_4).\mathrm{Trans}_{z(4)}(d_{4e2}).\ \mathrm{Trans}_{x(4e2)}(a_{4e2}).\ \mathrm{Rot}_{x(4e2)}(\alpha_{4e2}) \qquad (2.17) $$

After multiplying the matrices in Table 2.7, we come up with the general form for HTM, which is depicted in Eq.2.17 for axis 4_{e2} and will later be utilized to produce the general matrices that will readily match and precisely relate the end effector 2 of the robot arm with the base frame like it shown on Eq2.18.

We generate new parameters which are D1, D2, D3, D4, D5, D6 and D7 that take expressions into them to minimize the size of the matrix like it shown in equations group4.

As seen in Eq. 2.18, we have 4X4 metrics that describe the forward kinematics and final transformation metrics for the second-end effector.

$$ \begin{gathered} D1 = C(\Theta_{12})C(\Theta_{34}) \\ D2 = S(\Theta_{12})C(\Theta_{34}) \\ D3 = C(\Theta_{12})S(\Theta_{34}) \\ D4 = S(\Theta_{12})S(\Theta_{34}) \\ D5 = a_1C(\Theta_1) + a_2C(\Theta_{12}) + a_3C(\Theta_3)C(\Theta_{12}) + a_{4e2}C(\Theta_3)C(\Theta_{12}) - d_4S(\Theta_3)C(\Theta_{12}) \\ D6 = a_1S(\Theta_1) + a_2S(\Theta_{12}) + a_3C(\Theta_3)S(\Theta_{12}) + a_{4e2}C(\Theta_3)S(\Theta_{12}) - d_4S(\Theta_3)S(\Theta_{12}) \\ D7 = -a_3S(\Theta_3) - a_{4e2}S(\Theta_3) - d_4C(\Theta_3) \end{gathered} $$

Equations group 4

The rotation part of this metric is represented by the red shape with the R sample, and the translation part is represented by the blue shape with the T sample.

$$ {}^{0}_{4e2}T = {}^{0}_{4}T\,{}^{4}_{4e2}T = \begin{pmatrix} D1 & D3 & S(\Theta_{12}) & D5 \\ D2 & D4 & -C(\Theta_{12}) & D6 \\ -S(\Theta_{34}) & C(\Theta_{34}) & 1 & D7 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (2.18) $$

R T

The fourth column in matrix 2.18, first row in the translation section represents translation in the X direction, the second row represents translation in the Y direction, and the third row represents translation in the Z direction. All forward Kinematics has been done using Python programming and it has been shown in **Appendix A, part1**,with name of forward kinematics analysis.

2.1.4. Robot arm inverse kinematics

Finding the joint values for the position (**p**) and orientation (**o**) that connect the robot linkages is the goal of inverse kinematics. To perform inverse kinematics, θ1, θ2, θ3, and θ4 of the robot arm for the specified inverse orientation R and inverse position P are required. The ***AIBOMECH Agrobot*** is a two-gripper articulated arm that will be utilized in the farming process. It is necessary to determine the joint values 1,2,3, and 4 for the inverted position of an anthropomorphic/articulated arm[23].

For the first two joints of the robot arm, if we look from the top view, we will have two revolute joints both of which are rotating on the same axis. By using the close form of analytical method for kinematic analysis, we will get two equations for X and Y directions respectively as shown in Eq.2.19, and Eq.2.20.

$$Xp = a_1\, C(\Theta_1) + a_2\, C(\Theta_{12}) \tag{2.19}$$

$$Yp = a_1\, S(\Theta_1) + a_2\, S(\Theta_{12}) \tag{2.20}$$

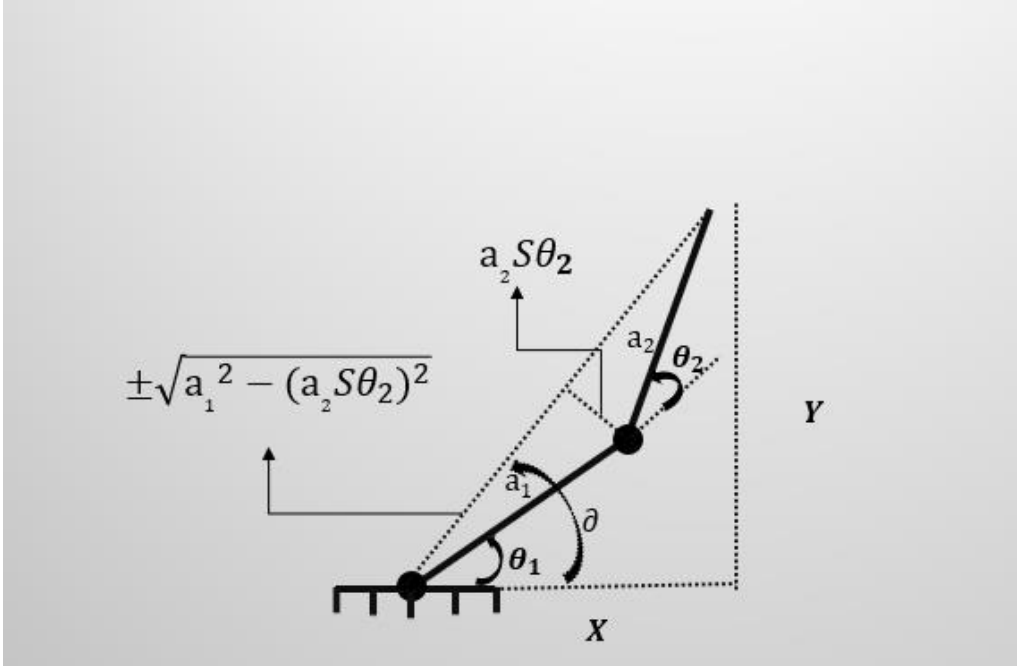


Figure 2. 3 AIBOMECH Agrobot Arm *first two link labelling*

$\Theta_{12} = \Theta_1 + \Theta_2$ which is the summation of joint angle one and joint angle two.

By squaring and sum of equations Eq.2.19 and Eq 2.20 for both sides, we end up with Eq.2.21, and Eq.2.22 respectively.

$$C2 = \frac{(Xp^2 + Yp^2 - {a_1}^2 - {a_2}^2)}{2a_1a_2} \quad (2.\ 21)$$

$$S2 = \pm\sqrt{1 - C2^2} \quad (2.\ 22)$$

$$\left.\begin{aligned} \beta &= a_2 S(\Theta_2) \\ \forall &= \pm\sqrt{{a_1}^2 - (a_2 S\theta_2)^2} \\ \partial &= \text{Atan2}\left(\frac{Y}{X}\right) \\ \Theta_1 &= \partial - \text{Atan2}\left(\frac{\beta}{\forall}\right) \end{aligned}\right\} \text{Equations group 5}$$

By using Eq2.21, Eq2.22, and ***Atan2*** trigonometric rule, we can find the value of the revolute joint two parametrically to allow us using it in obtaining the numeric values for that joint to be use in programming to actuate the motors later. See equation 2.23.

$$\Theta_2 = Atan2\left(\frac{S2}{C2}\right) \quad (2.\ 23)$$

For Θ_1 , like it is shown in Figure 2.3, by using geometric analysis of the first two joints we end up with group of equations shown in equation group2.

To find angle 3, and angle 4 of the manipulator, in our articulated mechanism, we suppose that to find the general rotation matrix of the end effector we assumed the rotation matrix of the end effector 2 concerning global frame. In this approach, there are three ways to use for rotation matrix to be represented which are **Euler angle** representation**, roll-pitch-yaw** representation and the **axis/angle** representation. In our case ,we supposed to used **roll-pitch-yaw** , see Figure 2.4 [24].

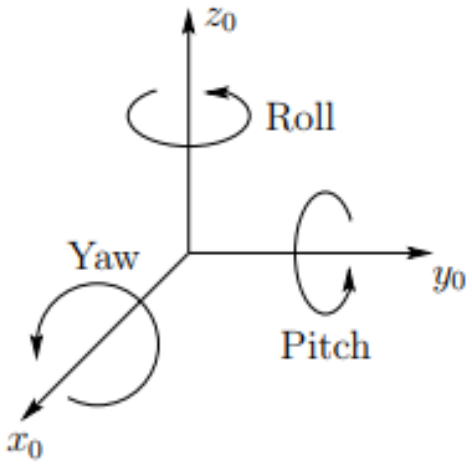


Figure 2. 4 Roll, pitch, and yaw angles.

Therefore, the order of rotation will follow the Z-Y-X ordering as an approach, which is **roll, pitch, yaw** which can represented as $R_{z,\emptyset}$, $R_{y,\theta}$,and $R_{x,\gamma}$ respectively, like in the following matrices.Equation 2.24 represents the rotation about Z-axis which is the roll angle.

$$R_{z,\emptyset} = \begin{pmatrix} C(\emptyset) & -S(\emptyset) & 0 \\ S(\emptyset) & C(\emptyset) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (2.\ 24)$$

Equation 2.25 represents the rotation about Y-axis which is the yaw angle.

$$R_{y,\theta} = \begin{pmatrix} C(\theta) & 0 & S(\theta) \\ 0 & 1 & 0 \\ -S(\theta) & 0 & C(\theta) \end{pmatrix} \quad (2.\ 25)$$

Equation 2.26 represents the rotation about X-axis which is the pitch angle.

$$R_{x,\gamma} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & C(\gamma) & -S(\gamma) \\ 0 & S(\gamma) & C(\gamma) \end{pmatrix} \quad (2.\ 26)$$

The **roll-pitch-yaw** resultant of multiplication will be shown in equation 2.27, and equation 2.28 respectively.

$$\boldsymbol{R_{ZYX}= R_{z,\emptyset}\, R_{y,\theta}\, R_{x,\gamma}} \quad (2.\ 27)$$

$$R_{ZYX} = \begin{pmatrix} C(\emptyset)C(\theta) & -S(\emptyset)C(\gamma) + C(\emptyset)S(\theta)S(\gamma) & S(\emptyset)S(\gamma) + C(\emptyset)S(\theta)C(\gamma) \\ S(\emptyset)C(\theta) & C(\emptyset)C(\gamma) + S(\emptyset)S(\theta)S(\gamma) & -C(\emptyset)S(\gamma) + S(\emptyset)S(\theta)C(\gamma) \\ -S(\theta) & C(\theta)S(\gamma) & C(\theta)C(\gamma) \end{pmatrix} \quad (2.\ 28)$$

As it shown in Figure 2.2, for base frame and the end effector the orthogonality of the axis, and by substituting the angles values in equation 2.28, we can end up with the rotation matrix, see equation 2.29, that we will use in the following analysis. The equation is representing the general rotation matrix between base frame and end effector 2.

$$Re_20= \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} \quad (2.\ 29)$$

As we know from transformation matrices that to map any axis to another one, we have to multiply the previous axis rotation matrix which is the rotation matrix between axis 2 and base frame of manipulator with the next one rotation matrix which is the matrix between end effector 2 frame with frame 2. Therefore, the rotation matrix between end effector 2 and base frame will be like it shown in equation 2.30.

$$Re_20= R20*R_{e2}2 \quad (2.\ 30)$$

$$R_{e2}2=\text{Inverse } (R20) * Re_20 \quad (2.\ 31)$$

To find the $R_{e2}2$, we multiplied both sides by the inverse matrix of (R20), see Eq31 so that we end up with the Eq2.32.

$$R_{e2}2= \begin{pmatrix} 0 & S(\theta_{12}) & C(\theta_{12}) \\ 1 & 0 & 0 \\ 0 & C(\theta_{12}) & -S(\theta_{12}) \end{pmatrix} \quad (2.\ 32)$$

On the other hand, we map axis 2 directly to the end effector 2 and obtain the result shown in Eq2.28.

$$R_{e2}2=\begin{pmatrix} C(\theta_3)*C(\theta_4) & -C(\theta_3)*S(\theta_4) & -S(\theta_3) \\ S(\theta_3)C(\theta_4) & -S(\theta_3)S(\theta_4) & C(\theta_3) \\ -S(\theta_4) & -C(\theta_4) & 0 \end{pmatrix} \quad (2.\ 33)$$

By equating Eq2.32, Eq2.33, we obtain θ4 in equation 2.34 which we got it from column 2, row 3 in equation 2.32, equation 2.33 respectively.

$$\theta_4 = -\cos^{-1} \mathrm{C}(\theta_1 + \theta_2) \quad (2.\ 34)$$

And we obtain θ3 in equation 2.35 which we got it from column 3, row 1 in equation 2.32, equation 2.33 respectively

$$\theta_3 = -\sin^{-1} \mathrm{C}(\theta_1 + \theta_2) \quad (2.\ 35)$$

By substituting values of $\theta 2, and\ \theta 1$ from Eq 2.23 and equations group5 respectively, we could get the values of $\theta 4, and\ \theta 3$ like it shown in equation 2.34, and equation 2.35 respectively. So that we can use them in our inverse kinematics analysis. All inverse Kinematics has been done using Python programming and it has been shown in **Appendix A**, **part2**, with name of inverse kinematics analysis.

2.2. Jacobian analysis

Robotics' Jacobian matrix defines the relationship between the velocities of a robot manipulator's joints ($\dot{\boldsymbol{\theta}}$)and end-effectors ($\dot{\boldsymbol{V}}$). If the robot's joints move at a certain speed, we might be interested in knowing how fast the end effector moves. Jacobian can be of use to us here. It is the connection between kinematics analysis and dynamic analysis that allows us to determine the inertia matrices, joint torque, and singularity of the robot arms from Jacobian analysis and take those values into account while designing the robotic arms' control algorithm.The calculations in this section are based on the kinematic analysis we completed in **Section 2.1**, so first, we must use rotation matrices and transformation matrices to characterize the differential kinematics of rigid bodies (***links of the robot arm***).

The vector space covered by the joint variables is referred to as the joint space, and the end-effector space is the vector space spanned by the end-effector location in the transformation expression. The Jacobian matrix is the matrix that translates the joint rates in the actuator space to the velocity state in the end-effector space for robot manipulators. We applied the Eq2.6 that we used in the previous section to obtain the rotation matrix for every singular joint of the robot arm.

The Jacobian matrix is an essential component in constructing trajectories with defined geometry in the end-effector space for Jacobian computations and related work of needed matrices. The bulk of industrial robot coordination algorithms avoids numerical inversion of the Jacobian matrix by producing analytical inverse solutions on the spot[16].In Eq2.36, we map the rotation of the joint of axis one to the joint of axis zero.

$$R10 = \begin{pmatrix} C(\theta_1) & -S(\theta_1) & 0 \\ S(\theta_1) & C(\theta_1) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (2.\ 36)$$

In Eq2.37, we map the rotation of the joint of axis two to the joint of axis zero.

$$R20 = \begin{pmatrix} C(\theta_{12}) & 0 & -S(\theta_{12}) \\ S(\theta_{12}) & 0 & C(\theta_{12}) \\ 0 & -1 & 0 \end{pmatrix} \quad (2.\ 37)$$

In Eq2.38, we map the rotation of the joint of axis three to the joint of axis zero.

$$\mathrm{R30} = \begin{pmatrix} \mathrm{C}(\theta_3)\mathrm{C}(\theta_{12}) & \mathrm{S}(\theta_{12}) & -\mathrm{S}(\theta_3)\mathrm{C}(\theta_{12}) \\ \mathrm{C}(\theta_3)\mathrm{S}(\theta_{12}) & -\mathrm{C}(\theta_{12}) & -\mathrm{S}(\theta_3)\mathrm{S}(\theta_{12}) \\ -\mathrm{S}(\theta_3) & 0 & -\mathrm{C}(\theta_3) \end{pmatrix} \quad (2.\ 38)$$

In Eq2.39, we map the rotation of the joint of axis four to the joint of axis zero.

$$\mathrm{R40} = \begin{pmatrix} \mathrm{C}(\theta_3)\mathrm{C}(\theta_{12}) & \mathrm{S}(\theta_3)\mathrm{C}(\theta_{12}) & \mathrm{S}(\theta_{12}) \\ \mathrm{C}(\theta_3)\mathrm{S}(\theta_{12}) & \mathrm{S}(\theta_3)\mathrm{S}(\theta_{12}) & -\mathrm{C}(\theta_{12}) \\ -\mathrm{S}(\theta_3) & \mathrm{C}(\theta_3) & 0 \end{pmatrix} \quad (2.\ 39)$$

In Eq2.40, we map the rotation of the joint of the end effector to the joint of axis zero.

$$R4e_{20} = \begin{pmatrix} C(\theta 34)C(\theta_{12}) & S(\theta_{34})C(\theta_{12}) & S(\theta_{12}) \\ C(\theta 34)S(\theta_{12}) & S(\theta_{34})S(\theta_{12}) & -C(\theta_{12}) \\ -S(\theta_{34}) & C(\theta_{34}) & 0 \end{pmatrix} \quad (2.\ 40)$$

2.2.1. Obtaining the direction and location of each joint axis

It is necessary to first know the position and orientation of each joint axis before computing the Jacobian matrix. The general formulation that we will use to compute the direction for every singular joint is shown in Eq2.41 so we will use the rotation matrix to the specified joint concerning the global frame, and multiply it with the joint axis direction.

$$Z_{i-1} = {}^{0}_{i-1}R * \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (2.\ 41)$$

After we use the Eq2.41we got $\mathbf{Z_0}$ like it shown in Eq2.42 and it will be used later to compute the Jacobian of that joint.

$$Z_0 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (2.\ 42)$$

After we use Eq2.41 and multiply the rotation matrix ${}^{0}_{1}R$ in Eq2.36 we got $\mathbf{Z_1}$ like it shown in Eq2.43 and it will be used later to compute the Jacobian of that joint.

$$Z_1 = {}^{0}_{1}R * \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (2.\ 43)$$

After we use Eq2.41 and multiply the rotation matrix ${}^{0}_{2}R$ in Eq2.37 we got $\mathbf{Z_2}$ like it shown in Eq2.44 and it will be used later to compute the Jacobian of that joint.

$$Z_2 = {}^{0}_{2}R * \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -S(\theta_{12}) \\ C(\theta_{12}) \\ 0 \end{pmatrix} \quad (2.\ 44)$$

After we use Eq2.41 and multiply the rotation matrix ${}^{0}_{3}R$ in Eq2.38 we got **Z₂** like it is shown in Eq2.45 and it will be used later to compute the Jacobian of that joint.

$$Z_3 = {}^{0}_{3}R * \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -S(\theta_{12}) \\ C(\theta_{12}) \\ 0 \end{pmatrix} \quad (2.45)$$

After we use Eq2.41 and multiply the rotation matrix ${}^{0}_{4}R$ in Eq2.39 we got **Z₃** like it shown in Eq2.46 and it will be used later to compute the Jacobian of that joint.

$$Z_4 = {}^{0}_{4}R * \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} S(\theta_{12}) \\ -C(\theta_{12}) \\ 0 \end{pmatrix} \quad (2.46)$$

After we use Eq2.41 and multiply the rotation matrix $R4e_{20}$ in Eq2.40 we got **Z₅** like it is shown in Eq2.47 and it will be used later to compute the Jacobian of that joint.

$$Z_5 = {}^{0}_{5}R * \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} S(\theta_{12}) \\ -C(\theta_{12}) \\ 0 \end{pmatrix} \quad (2.47)$$

After we obtained the position and orientation of all joints in previous calculations, we can go one step further to obtain the Jacobian by the following process.

2.2.2 Obtaining the position vector of the end effector concerning all frames

To obtain the Jacobian of every singular joint, the position vector of that joint is necessary to be used in equation 2.48 and it can be accomplished like in the following.

The general formulation that will be followed is described in the following equations which are representing the direction of the joints that we interested in.

$${}^{i-1}_{n}P *= ({}_{i-1}{}^{0}R)({}^{i-1}_{i}r) + {}^{i}_{n}P * \quad (2.48)$$

Like it shown in equation 2.49, the general formulation for axis position is represented to be multiplied with the rotation matrix of the axis with respect to base frame. The

process of obtaining the position vector of the joints of ***AIBOMECH Agrobot*** arm will be carried like in the following equations.

$$ {}^{i-1}_{i}r = \begin{pmatrix} a_i C\theta_i \\ a_i S\theta_i \\ d_i \end{pmatrix} \qquad (2.\ 49) $$

Like it shown in equation 2.50, to obtain the position vector of the fourth axis, we multiply the rotation matrix of the same axis with the position of that axis to get the final result where it shown in equation 2.51.

$$ {}^{3}_{4}\mathrm{P} *= \begin{pmatrix} \mathrm{d}_4 \mathrm{S}\theta_{12} \\ \mathrm{d}_4 \mathrm{C}\theta_{12} \\ 0 \end{pmatrix} \qquad (2.\ 51) $$

$$ {}^{3}_{4}P *= ({}^{0}_{3}R)({}^{3}_{4}r) \qquad (2.\ 50) $$

Like it shown in equation 2.52, to obtain the position vector of the third axis, we multiply the rotation matrix of the same axis with the position of that axis to get the final result where it shown in equation 2.53.

$$ {}^{2}_{4}P *= ({}^{0}_{2}R)({}^{2}_{3}r) + {}^{3}_{4}P * \qquad (2.\ 52) $$

$$ {}^{2}_{4}P *= \begin{pmatrix} a_3 C\theta_3 C\theta_{12} - d_4 S\theta_{12} \\ a_3 C\theta_3 S\theta_{12} + d_4 C\theta_{12} \\ -a_3 S\theta_3 \end{pmatrix} \qquad (2.\ 53) $$

Like it shown in equation 2.54, to obtain the position vector of the second axis, we multiply the rotation matrix of the same axis with the position of that axis to get the final result where it shown in equation 2.55.

$$ {}^{1}_{4}P *= ({}^{0}_{1}R)({}^{1}_{2}r) + {}^{2}_{4}P * \qquad (2.\ 54) $$

$$ {}^{1}_{4}P* = \begin{pmatrix} a_2 C\theta_{12} + a_3 C\theta_3 C\theta_{12} - d_4 S\theta_{12} \\ a_2 S\theta_{12} + a_3 S\theta_{12} C\theta_3 + d_4 C\theta_{12} \\ -a_3 S\theta_3 \end{pmatrix} \quad (2.55) $$

$$ {}^{0}_{4}P* = {}^{1}_{4}P* \quad (2.56) $$

$$ {}^{0}_{4}P* = \begin{pmatrix} a_2 C\theta_{12} + a_3 C\theta_3 C\theta_{12} - d_4 S\theta_{12} \\ a_2 S\theta_{12} + a_3 S\theta_{12} C\theta_3 + d_4 C\theta_{12} \\ -a_3 S\theta_3 \end{pmatrix} \quad (2.57) $$

Like it shown in equation 2.56, to obtain the position vector of the first axis, we multiply the rotation matrix of the same axis with the position of that axis to get the final result where it shown in equation 2.57.

2.2.3. Obtaining Jacobian matrices

As shown in equation 2.53, we can obtain the Jacobian matrices by cross product the direction of the joints with the position vector of that joint concerning the base frame.

For first joint, the Jacobian obtained using the equation shown in 2.58 to end up with the matrix shown in 2.59.

$$ J_1 = \begin{pmatrix} z_0 \times {}^{0}_{4}P* \\ z_0 \end{pmatrix} \quad (2.58) $$

$$ J_1 = \begin{pmatrix} -a_2 S\theta_{12} - a_3 S\theta_{12} C\theta_3 - d_4 C\theta_{12} \\ a_2 C\theta_{12} + a_3 C\theta_{12} C\theta_3 - d_4 S\theta_{12} \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad (2.59) $$

For second joint, the Jacobian obtained using the equation shown in 2.60 to end up with the matrix shown in 2.61.

$$J_2=\begin{pmatrix} z_1 \times {}^1_4P * \\ z_1 \end{pmatrix} \quad (2.\ 60)$$

$$J_2=\begin{pmatrix} -\mathrm{a}_2 S\theta_{12} - \mathrm{a}_3 S\theta_{12} C\theta_3 - d_4 C\theta_{12} \\ \mathrm{a}_2 C\theta_{12} + \mathrm{a}_3 C\theta_{12} C\theta_3 - d_4 S\theta_{12} \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad (2.\ 61)$$

For third joint, the Jacobian obtained using the equation shown in 2.62 to end up with the matrix shown in 2.63.

$$J_3=\begin{pmatrix} z_2 \times {}^2_4P * \\ z_2 \end{pmatrix} \quad (2.\ 62)$$

$$J_3=\begin{pmatrix} -\mathrm{a}_3 S\theta_3 C\theta_{12} \\ -\mathrm{a}_3 S\theta_{12} S\theta_3 \\ -\mathrm{a}_3 C\theta_3 \\ -S\theta_{12} \\ C\theta_{12} \\ 0 \end{pmatrix} \quad (2.\ 63)$$

$$J_4=\begin{pmatrix} z_3 \times {}^3_4P * \\ z_3 \end{pmatrix} \quad (2.\ 64)$$

$$J_4=\begin{pmatrix} 0 \\ 0 \\ 0 \\ -S\theta_{12} \\ C\theta_{12} \\ 0 \end{pmatrix} \quad (2.\ 65)$$

$$\left.\begin{aligned} \text{E1} &= -\text{a}_2 S\theta_{12} - \text{a}_3 S\theta_{12} C\theta_3 - d_4 C\theta_{12} \\ \text{E2} &= \text{a}_2 C\theta_{12} + \text{a}_3 C\theta_{12} C\theta_3 - d_4 S\theta_{12} \\ \text{E3} &= -\text{a}_2 S\theta_{12} - \text{a}_3 S\theta_{12} C\theta_3 - d_4 C\theta_{12} \\ \text{E4} &= \text{a}_2 C\theta_{12} + \text{a}_3 C\theta_{12} C\theta_3 - d_4 S\theta_{12} \end{aligned}\right\} \text{Equations group 6}$$

For fourth joint, the Jacobian obtained using the equation shown in 2.64 to end up with the matrix shown in 2.65.

For the general Jacobian of the ***AIBOMECH Agrobot*** manipulator and by applying the sequence shown in equations group 6, we obtain the matrix shown in equation 2.66 so that we can use it in theoretic calculations during programming and robot tasks implementation.

$$J = \begin{pmatrix} \text{E1} & \text{E3} & -\text{a}_3 S\theta_3 C\theta_{12} & 0 \\ \text{E2} & \text{E4} & -\text{a}_3 S\theta_{12} S\theta_3 & 0 \\ 0 & 0 & -\text{a}_3 C\theta_3 & 0 \\ 0 & 0 & -S\theta_{12} & -S\theta_{12} \\ 0 & 0 & C\theta_{12} & C\theta_{12} \\ 1 & 1 & 0 & 0 \end{pmatrix} \qquad (2.\ 66)$$

All Jacobian analysis has been done using Python programming and it has been shown in **Appendix A**, **part3**, with name of Jacobian analysis

2.3. Three DoF Parallel robot manipulators with gyroscope trajectory generation

Adding to serial kinematic chain, Parallel robots have usage in agricultural operation. A parallel robot is a form of manipulator that differs from conventional robotic manipulators in that it has three or more arm that operate concurrently rather than serially. Parallel robots are highly suited for usage in agriculture because to their many benefits, which include excellent precision, quick speed, and high payload capacities. Parallel robots can be work as modular machines so that it can be integrated with mobile platforms, with serial manipulators and perform several of complex tasks regarding to

the problem needs to be solved in real environment. Therefore, some of interesting potential applications of parallel robot manipulators are like the following.

Harvesting: Crops including fruit, vegetables, and nuts can be accurately and effectively harvested using parallel robots. For instance, a parallel robot could be employed to delicately grip and gather fruit from trees or to harvest vegetables from field plants.

Planting and sowing: Parallel robots can be used to precisely sow seeds in greenhouse environments or plant seedlings in fields. This may also serve to lessen the demand for human labor while increasing the efficiency and precision of planting and sowing processes.

Weeding: Weeds in fields can be located and eliminated utilizing parallel robots, either by hand pulling or applying herbicides. This can increase crop yields and lower the requirement for herbicides.

Fertilizing and irrigation: Parallel robots can be used to precisely and carefully apply water and fertilizer to crops. This may assist reduce the need for fertilizers and water while also improving the health and yields of crops.

In this section, we have used MPU6050 gyroscope to manipulate a 3 DoF freedom parallel robot manipulator to be used in some of pick and place purposes. Parallel robots are being used in a variety of applications as masterworks. These robots typically outperform serial robots in terms of speed, strength, and accuracy. However, because these robots have both parallel and serial singularities, their workspaces are both significantly smaller and more complicated. Many strategies have been put out to expand the workspace of parallel robots. One of these involves designing the robot's geometry so that there are no parallel singularities [25].

2.3.1. Kinematic analysis and design

In kinematic analysis for this section, we have three arms which have three active dependent joints with three servo actuators, and three passive joints that can move freely. Those arms intersect in the middle with triagonal platform that has three passive joints which allow the end effector moves freely in X-Y plane. We use SOLIDWORKS, see

figure 2.5 to design the robot links, holders and connections that will carry the actuators and fix them to the base of the arm which is fixed to the ground.

After designing the manipulator parts, we manufactured them using 3D printing, see figure 2.6. Then, we mount them together to with the bearings that help the platform to be moved horizontally without any deflection.

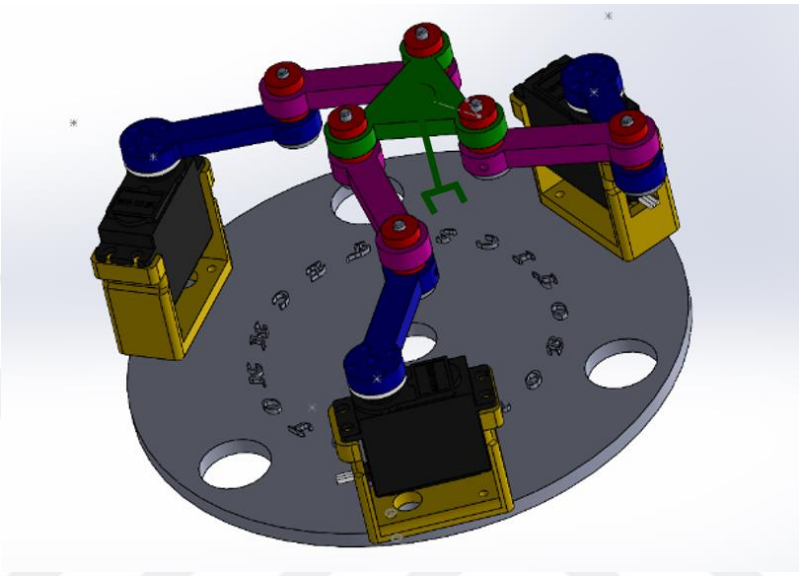


Figure 2. 5 3DoF parallel robot manipulator CAD Design

In addition, we used Arduino Mega microcontroller to control the motion of the robot with respect to the comments coming from Gyroscope and the calculation of inverse kinematic of the robot manipulator.

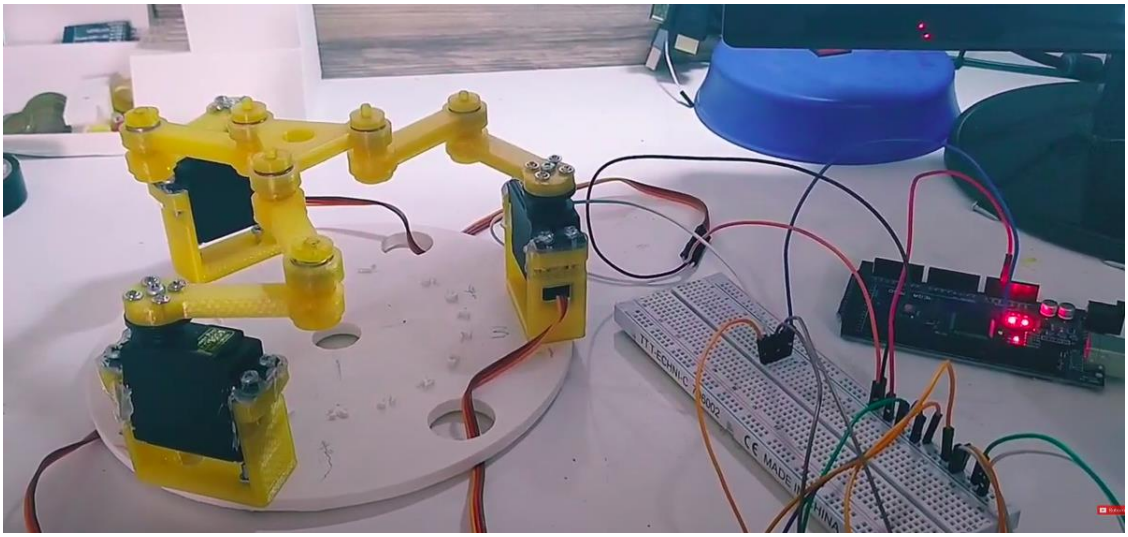


Figure 2. 6 3DoF parallel robot manipulator 3D Printed parts

- Geometric inverse kinematic

Like it shown in figure 2.7, the compact shape of kinematic labelling of parallel robot arm simplified and help us to carry out the kinematic calculation. However, it will have that different nine working mode regions with different enclosed areas.

In addition to that, like it shown in figure 2.8, and 2.9 respectively, the amount of joint rotation will have two different solutions which are elbow up and elbow down, and we can use them in our inverse kinematics calculations, these solutions can be computed combining both geometrical and analytical that guide us to come up with pure solution.

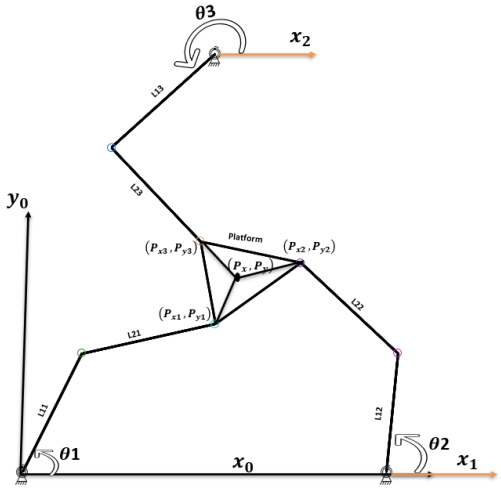
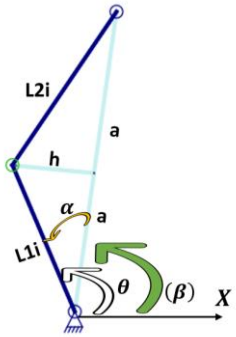
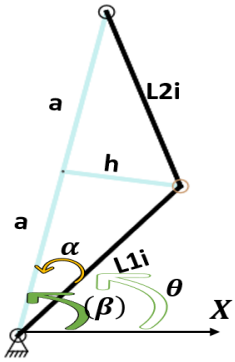


Figure 2. 7 3DoF parallel manipulator kinematic labelling



$$\boldsymbol{\theta_i = \beta_i + \alpha_i}$$

Figure 2. 8 Elbow down



$$\boldsymbol{\theta_i = \beta_i - \alpha_i}$$

Figure 2. 9 Elbow up

Using Pythagorean theorem, we can obtain Eq2.62, and it can be used in the following analysis.

$$a^2 + h^2 = L^2 \qquad (2.67)$$

By using symmetric shape of the rectangular shape shown in Figure 2.7, Figure 2,8, we make a perpendicular line to the middle of the line |***a***+***a***|. By square this length and equalize it with the end effector X and Y position, we can get the equation 2.63.

Also, we can get the equation 2.64 to find the value of ***a*** to use it in the coming analysis. In addition to that, after dividing the triangle shown on figure 2.7 and 2.8 to two symmetric rectangle, we can find the value of perpendicular hypothetical line ***h*** which will be used in arc tangent role to find the value of hypothetical $\boldsymbol{\alpha_i}$ which is between the link one and the line between end effector and origin, which will be used in inverse kinematics analysis.

Finally, for entire shape shown inf figure 2.6, we can use Arc tangent role to obtain the value of hypothetical $\boldsymbol{\beta_i}$ angle which is between the X axis and the hypothetical line between end effector and origin.

The amount of rotation of the actuator will be the algebraic summation of, and it will be dependent if the ram is elbow up or elbow down like it shown on figure 2.7,2.8 respectively.

$$(\mathbf{a}+\mathbf{a})^2 = \boldsymbol{P_{x_i}}^2 + \boldsymbol{P_{y_i}}^2 \qquad (2.\ 68)$$

$$\mathbf{a} = \frac{\sqrt{\boldsymbol{P_{x_i}}^2 + \boldsymbol{P_{y_i}}^2}}{\mathbf{2}} \qquad (2.\ 69)$$

$$\boldsymbol{h} = \sqrt{\boldsymbol{L^2} + \mathrm{a}^2} \qquad (2.\ 70)$$

$$\boldsymbol{\alpha_i} = \boldsymbol{ATAN2}(\boldsymbol{h}, \mathbf{a}) \qquad (2.\ 71)$$

$$\boldsymbol{\beta_i} = \boldsymbol{ATAN2}\left(\boldsymbol{P_{y_i}}, \boldsymbol{P_{x_i}}\right) \qquad (2.\ 72)$$

$$\boldsymbol{\theta_i} = \boldsymbol{\beta_i} \pm \boldsymbol{\alpha_i} \qquad (2.\ 73)$$

2.3.2. Working mode regions

Depending on the working dependencies of every singular manipulator, we end up with nine working mode regions as a compact shape for all three manipulators. In between those regions, we will have crossing singularities which can occur when the working mode region goes from one to another. Those singularities are mixed between serial and parallel singularities. Serial singularities are situations in which one or more degrees of freedom are lost for the end-effector. The limitations of each operating mode are set forth by these singularities. In parallel singularities, an end-effector is subjected to a force or moment for which the actuators are unable to rotate or act.

Table 2.8:Three DoF Parallel Robot Manipulator working mode regions

1st manipulator	2nd manipulator	3rd manipulator
+	+	+
+	+	-
+	-	+
+	-	-
-	+	+
-	+	-
-	-	+
-	-	-

Since they are inside the workspace, these singularities make it smaller. They can also be eliminated, but only from a specific working mode's workspace (a subspace). The resulting workspace will be less than all of the working mode subspaces put together, even if this subspace is devoid of singularities[25].

Using redundant actuators would be another approach to expand the workspace of a parallel robot and lessen or perhaps eliminate parallel singularities. The cost of the robot would drastically increase if motors and gearboxes were included. Table depicts the four Robot functioning mode areas. Calculating the subregions corresponding to the serial and parallel singularities reveals the boundaries of each working mode region, which each covers a different area of the entire workspace. The robot must bridge a serial singularity

to transition from one working mode zone to another one. All necessary Arduino codes for the parallel robot manipulator can be seen in **Appendix C, part 1**, with name of codes for 3DoF parallel manipulator.

2.4. Serial Manipulator Kinetic Analysis

Robotics kinetic calculations assess and explain a robot's motion and external forces using mathematical models and equations. This may involve computations in the fields of dynamics and kinematics, which both deal with the forces acting on the robot and how they affect its mobility. The robot's movement can be controlled, its behavior predicted, and its stability and safety can be guaranteed using these calculations. Inverse kinematics, forward kinematics, and motion planning are a few methods frequently utilized in dynamic computations for robotics.

These analysis uses two different kinds of dynamic calculations which are forward dynamics and inverse dynamics.

A technique called forward dynamics is used to calculate a robot's mobility and end effector motion based on the forces and torques operating on it. On the hand, inverse dynamics is utilized to ascertain the forces and torques necessary to cause a specific robot motion so that we determine the torques and forces based on end effector given motion. Both forward dynamics and inverse dynamics are crucial instruments for studying and regulating robot motion. They are used in numerous different applications, including trajectory planning, motion control, and collision avoidance [16].

In robotics, dynamic computations can be carried out in a variety of ways, including: Lagrangian mechanics, Newton-Euler methods, Kane's method, Featherstone's algorithm, Hybrid Dynamics. In my thesis calculations to do analysis for 4 DoF serial robot manipulator, I used Lagrangian mechanics. It formulates the equation of motion using a set of generalized coordinates, and it eliminates some of the forces of constraints at the outset [16].

2.4.1. General form of dynamical equation

Depending on Lagrangian principle that aims to derive the governing the general formulation of equation of motion, it could finalize this equation like it shown in 2.74.

$$\tau = M\ddot{\boldsymbol{\theta}} + \boldsymbol{V} + \boldsymbol{G}\ , \tau = \boldsymbol{Q} - J^T F_e + f_r \qquad (2.74)$$

In equation 2.74, the first term is called the inertia forces, the second term is called the Coriolis and centrifugal force, and the third term is representing the gravitational effects forces. The steps that we followed in order to compute the dynamical calculations of our ***AIBOMECH Agrobot*** manipulator will be summarized in the following session.

2.4.1.1. Inertia Matrix calculations ($\boldsymbol{M_{matrix}}$)

In this section, all materials and components that needed for calculating the first term of dynamic equation, see equation 2.74.

$$M(\theta) = J_{v1}^T m_1 J_{v1} + J_{v2}^T m_2 J_{v2} + J_{v3}^T m_3 J_{v3} + J_{v4}^T m_4 J_{v4} + J_{w1}^T I_1 J_{w1} + J_{w2}^T I_1 J_{w2} + J_{w3}^T I_3 J_{w3} + J_{w4}^T I_4 J_{w4} \qquad (2.75)$$

When everything is ready, it may be swapped out in equation 2.75 to get the generic 4 by 4 matrices required for the calculations. J_{vi}^T, J_{vi}, J_{wi}^T,and J_{wi} are representing links linear and angular velocity with their transpose.

- **Links inertia matrices calculations**

Inertia matrices are important to be used in the first term of 2.74 equation that we mentioned before. Firstly, I obtained $\boldsymbol{I_i^i}$ matrices which is representing the inertia matrix of link $\boldsymbol{i}$ about its center of mass and expressed in the link frame $\boldsymbol{i}$. To obtain this matrix, I used the equation 2.76, assuming that the robot links have symmetric rectangular and solid parts, m_i, a_i are representing the link mass and length respectively. Then, we compute I_i matrix which will be used in all coming calculations that is going to be the first term of general form of dynamic equation.

$$I_i^i = \frac{1}{12} m_i {a_i}^2 \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, I_i = R_i^0 I_i^i (R_i^0)^T \quad \text{for } i = 1, 2, \ldots 4 \qquad (2.76)$$

In our ***AIBOMECH Agrobot*** **manipulator** dynamics calculations, I used Mathematica software to do all computations needs. the results that obtained for I_1^1 , I_2^2 , I_3^3 , I_4^4 expressed in equations 2.77,2.778,2.79, and 2.80 respectively.

$$I_1^1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \frac{m_1 {a_1}^2}{12} & 0 \\ 0 & 0 & \frac{m_1 {a_1}^2}{12} \end{pmatrix} \quad (2.77)$$

$$I_2^2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \frac{m_2 {a_2}^2}{12} & 0 \\ 0 & 0 & \frac{m_2 {a_2}^2}{12} \end{pmatrix} \quad (2.78)$$

$$I_3^3 = \begin{pmatrix} \frac{m_3 {a_3}^2}{12} & 0 & 0 \\ 0 & \frac{m_3 {a_3}^2}{12} & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (2.79)$$

$$I_4^4 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \frac{m_4 {a_4}^2}{12} & 0 \\ 0 & 0 & \frac{m_4 {a_4}^2}{12} \end{pmatrix} \quad (2.80)$$

And the results that we obtained for I_1, I_2, I_3 ,and I_4 will be shown in **Appendix B**, **part1** with its required codes with name of Links Inertia Matrices Computation**.**

- **Jacobian submatrices calculations**

  In this part the Jacobian that will be used in equation 2.75 will be obtained, and some steps should be done before obtaining Jacobian. Firstly, the position vector of the center of mass of the links with respect to the base frame has been computed, see equation 2.81.

$$ {}^{i-1}_{c_i}r = \begin{pmatrix} \frac{a_i C\theta_i}{2} \\ \frac{a_i S\theta_i}{2} \\ \frac{d_i}{2} \end{pmatrix} , {}^{i-1}_{c_i}P = {}^{i-1}_{i}R \, {}^{i-1}_{c_i}r + {}^{i}_{i-1}P * \quad (2.81)$$

i representing the link number, ${}^{i-1}_{c_i}r$ the distance between the link's center and the end of the link that connects to the present link, ${}^{i}_{i-1}P *$ the position vector of the previous link

described on base frame of the manipulator, and ${}^{i-1}_{c_i}P$ the position vector of the position vector from the center of the link to previous one described on the base frame.

$$J_{vi}^{j} = \begin{Bmatrix} Z_{j-1} \times {}^{j-1}_{ci}P * \text{, for revolute joint} \\ Z_{j-1} \text{ , for prismatic joint} \end{Bmatrix}, \; J_{wi}^{j} = \begin{Bmatrix} Z_{j-1}\text{, for revolute joint} \\ 0 \text{ , for prismatic joint} \end{Bmatrix} \quad (2.82)$$

As it shown in equation 2.82 J_{vi}^{j} , J_{wi}^{j} are the partial rate of change of the velocity of the center of mass and the angular velocity if the link i and expressed in the base frame. z_{j-1}represents the direction of the link descried on base frame. Equation 2.41 is used to obtaining, z_{j-1} .

$$J_{vi} = \left[J_{vi}^{1}, J_{vi}^{2}, \dots, J_{vi}^{i} \;\; 0{,}0\right] \;, J_{vi} = \left[J_{wi}^{1}, J_{wi}^{2}, \dots, J_{wi}^{i} \;\; 0{,}0\right] \quad (2.83)$$

As expressed in equation 2.83, the J_{vi}, J_{wi} called the link Jacobian submatrices and, obtained them using Mathematica, see **Appendix B**, **part 2**, with name of links Jacobian computation.

$$M_{ij}(\theta) = \begin{pmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{pmatrix}, i, j \text{ for rows and columns} \quad (2.84)$$

After preparing all Jacobians and links Inertia matrices (M matrix) which we done above, and by using equation 2.75, we could end ep with 4 by 4 inertia matrix that shown in equation 2.84. There will be 4 matrices for robots 4 links, and the general matric that will be used in calculation will be the sum of them. The calculation of this part done in Mathematica and the code for it is shown in **Appendix B**, **part 3**, with name of General Mass inertia Matrix for Dynamic equation, first term.

2.4.1.2. Coriolis, Centrifugal force matrix calculations ($\boldsymbol{C_{matrix}}$)

In this section, I come up with new phenomenon known as the Coriolis force. This kind of force generated in a reference frame that is rotating relative to an inertial frame. Also, large-scale atmospheric and oceanic current rotation as well as the rotation of objects within a rotating reference frame are caused by it. In addition to that, a "fictitious" force

known as the centrifugal force affects things traveling in a circular motion in a frame of reference that is not inertial. These calculations can be challenging and often required for a solid foundation in differential equations and vector calculus. Therefore, I renormalize the equation 2.74 which was in matrix form to come up with the new equation, see equation 2.85.I done first term in the first section, and in this section, I will compute the second term in the following.

$$\sum_j m_{kj}(\theta)\ddot{\theta}_j + \sum_{i,j} c_{ikj}(\theta)\dot{\theta}_i\,\dot{\theta}_j + g_k(\theta) = \tau_k\,,\ k = 1\,,\ldots\ldots,n \qquad (2.85)$$

In equation 2.85 i , j , k represent matrices rows, column, and the manipulator joints, or degrees of freedom. In this study, $i = 4$, $j = 4$, and $k=4$.

If we look to the element c_{ikj}, it is similar to ***Christoffel Symbols of the First Kind***, see equation 2.86, which is frequently used in some of dynamical equations, and we are going to use it to compute the second term of dynamic equation of motion.

$$c_{ijk} = \frac{1}{2}\left\{\frac{\partial M_{kj}}{\partial q_i} + \frac{\partial M_{ki}}{\partial q_j} - \frac{\partial M_{ij}}{\partial q_k}\right\} \qquad (2.86)$$

$c'_{ijk}s$ calculations are totally depended on i , j , k factors . Therefore, the N elements of it calculated using the equation $\left(\boldsymbol{c_{ijk}}\right)_{\boldsymbol{N}} = \boldsymbol{4^3}\boldsymbol{=64}$ elements, because, we have three factor which are i , j , k ,and we have 4 cases of each of them at total.

Table 2.9: Christoffel Symbols c_{ijk} values

Column-1			Column-2		
ROW1	C1R1	C111+C121+C131+C141	ROW1	C2R1	C211+C221+C231+C241
ROW2	C1R2	C112+C122+C132+C142	ROW2	C2R2	C212+C222+C232+C242
ROW3	C1R3	C113+C123+C133+C143	ROW3	C2R3	C213+C223+C233+C243
ROW4	C1R4	C114+C124+C134+C144	ROW4	C2R4	C214+C224+C234+C244
Column-3			**Column-4**		
ROW1	C3R1	C311+C321+C331+C341	ROW1	C4R1	C411+C421+C431+C441
ROW2	C3R2	C312+C322+C332+C342	ROW2	C4R2	C412+C422+C432+C442
ROW3	C3R3	C313+C323+C333+C343	ROW3	C4R3	C413+C423+C433+C443
ROW4	C3R4	C314+C324+C234+C344	ROW4	C4R4	C414+C424+C434+C444

All preparations required for the ***Christoffel Symbols of the First Kind*** are shown at **Appendix B**, **part 4 table** ,with name of Christoffel Symbols of the First Kind.

We start with the $\boldsymbol{i}$ factor, started from 1, but repeated 16 times, the, $\boldsymbol{j}$ factor, started from 1, but repeated 4 times, and the factor k, started from 1, but it doesn't repeat in the same section as $\boldsymbol{i}$ and $\boldsymbol{j}$, as indicated in the appendix table. After simplifying every potential value for $\boldsymbol{c_{ijk}}$, we gather related elements to be in the desired column. 4 columns in total will be produced, as shown in Table 2.9.

The approach for c_{ijk} collection to be put in the suitable calculation and applied to their columns is carried like it shown in equation 2.87, and for our case we simplify it like it shown in table in **Appendix B**, **part 4 table. $\boldsymbol{i}$** is the number of columns, $\boldsymbol{j}$ is the value inside $\mathbf{c_{ijk}}$,the element that is changed with respect to its position.

$$C_i = \begin{bmatrix} \sum_{j=1}^{4} c_{ij1}\dot{q}_j \\ \sum_{j=1}^{4} c_{ij2}\dot{q}_j \\ \sum_{1}^{4} c_{ij3}\dot{q}_j \\ \sum_{1}^{4} c_{ijn}\dot{q}_j \end{bmatrix} \tag{2. 87}$$

All calculations for this section were performed using Mathematica software, and the necessary materials and codes for computing joint torque were computed using Mathematic software as well. These results are displayed in **Appendix B parts 5** through **7**. **Part 5** with name of Preparing M matrices that will be used to calculate $\boldsymbol{c_{ijk}}$ elements which is responsible for preparing Inertia matrix elements that will be used in the second term of dynamic equation. **Part 6**, with name of Computing $\boldsymbol{c_{ijk}}$ elements values which will carry out the calculations to prepare Christoffel Symbols of the First Kind elements. **Part 7**, with name of Computing Coriolis and Centrifugal 4 by 4 matrix which will carry out the calculations to Coriolis and Centrifugal forces matrix.

2.4.2. Joints Torque Calculations and their Graphics

After calculating the inertia matrix, the Coriolis force matrix, and the Centrifugal force matrix, we proceeded to calculate the final torques of the joints in order to get them ready for the final graphs that we will use to decide which actuators to use in our project that will be put into practice in real life. **Appendix B <u>Part 8</u>**, with name of Computing Torque values for robot Joints which will carry out the calculation of final robot joints torques.

$$T1 = Simplify\,[TORQUEMATRIX\ [[1,1]]]$$
$$T2 = Simplify\,[TORQUEMATRIX\ [[2,1]]]$$
$$T3 = Simplify\,[TORQUEMATRIX\ [[3,1]]]$$
$$T4 = Simplify\,[TORQUEMATRIX\,[[4,1]]]$$

Equations group 7

We derive the torques for the first, second, third, and fourth joints, as shown in Equations group 7, and we store them in new variables called T1, T2, T3, and T4, which stand for the torques for joints 1 through 4, respectively.

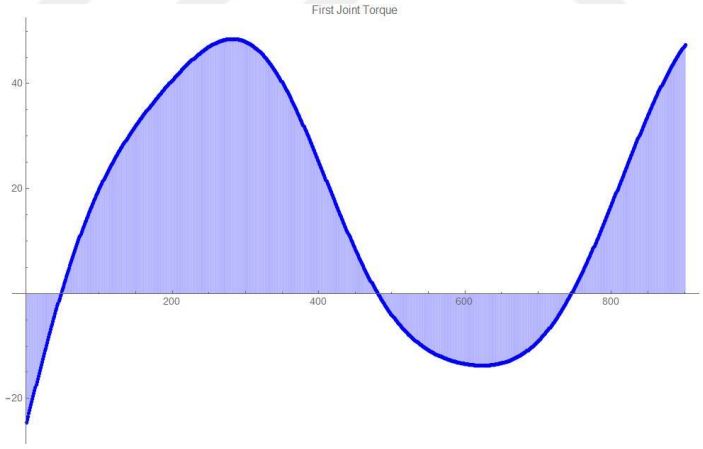


Figure 2. 10 First Joint Torque

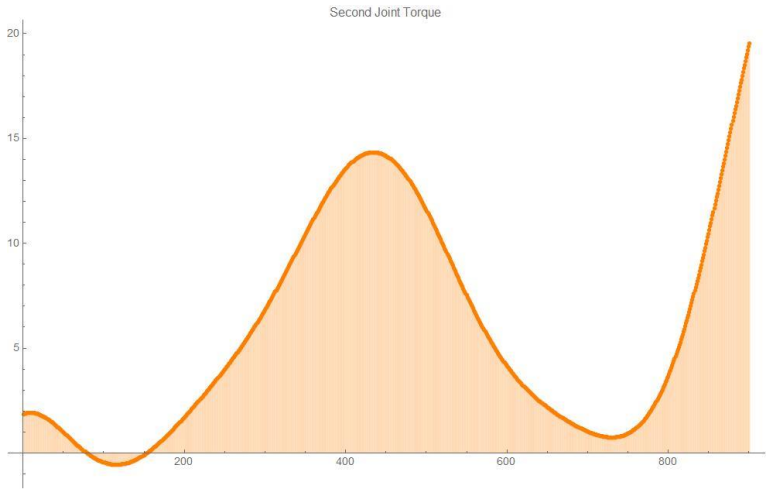


Figure 2. 11 Second Joint Torque

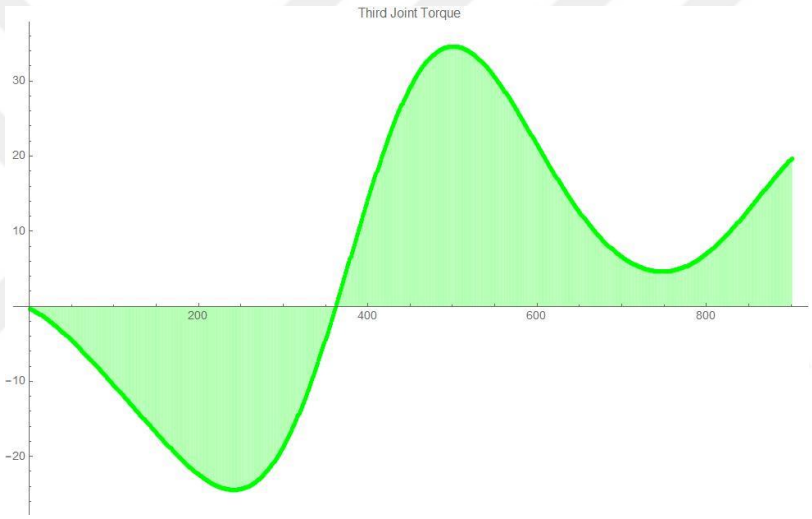


Figure 2. 12 Third Joint Torque

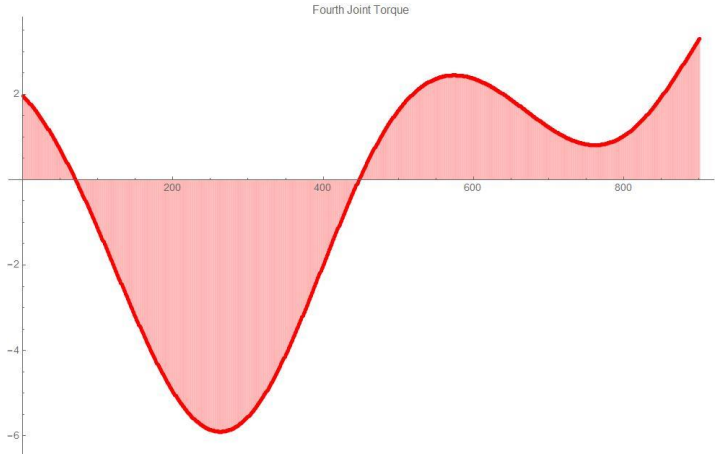


Figure 2. 13 Fourth Joint Torque

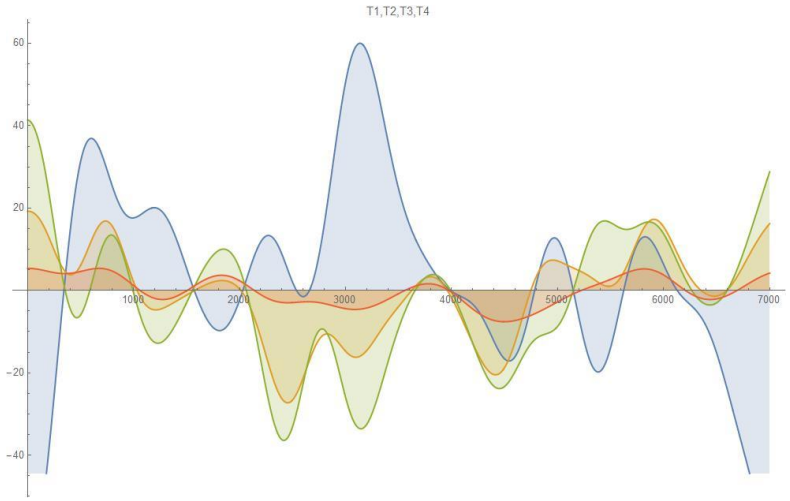


Figure 2. 14 T_1 , T_2 , T_3 ,and T_4 combined plot

Appendix B **Part 9**, with name of drawing torque values graphs. This part aids the robot designers in choosing actuators by illustrating the joint torques in a visual case to make it easy to grasp what is happening in manipulator joints during robot works. The figures 2.10 through 2.13 make it obvious that the largest torque is at joint 1, where it is going up to 40 kg.cm. This makes sense because the first joint is carrying all of the robot's parts, so the maximum torque must be there. Torque 2 in figure 2.11 varies between 0 and 20 kg.cm, which makes sense given that joint 2 is in the middle of the system and its torque shouldn't be that high or low. Due to the fact that joint 3 is curving the other half of the manipulator, which is traveling in an area that must have a high load of parts and grippers, the torque at joint 3 see figure 2.12 ranges from 0 to 30 kg.cm, which must be normal. The last joint, which is the fourth torque, see figure 13, has a value, which ranges from 0 to 9 kg.cm since it is free to move and does not have the weight necessary to support a high torque. Finally, to compare all joints torque in a visualized and clear manner, we combine all of them in one plot, see figure 2.14.

2.5. Serial Manipulator Workspace Analysis

Over the past 50 years, there has been a lot of discussion about the reachable workspaces of robotic manipulators. By sampling joint angles and assessing the set's boundary in the space of rigid-body motions, one method of creating workspaces is to sample joint angles, [19].Workspace analysis of a serial robot manipulator is securing the evaluation of the accessible area and range of motion of the end-effector . Also, the

purpose of workspace analysis is to establish the robot's operational boundaries and confirm that it is capable of carrying out the task for which it was designed.

The workspace of a serial manipulator can be obtained and depicted using a variety of techniques, including forward and inverse kinematics, simulation, and visualization. For the manipulator end-effector in our situation, there are two grippers, and each gripper has a unique workspace. In figure 2.15, which shows the ***brown shape*** represents gripper one and the ***purple shape*** represents gripper two, we used the forward kinematics approach to extract and plot the 3D shape to estimate the workspace of both grippers.

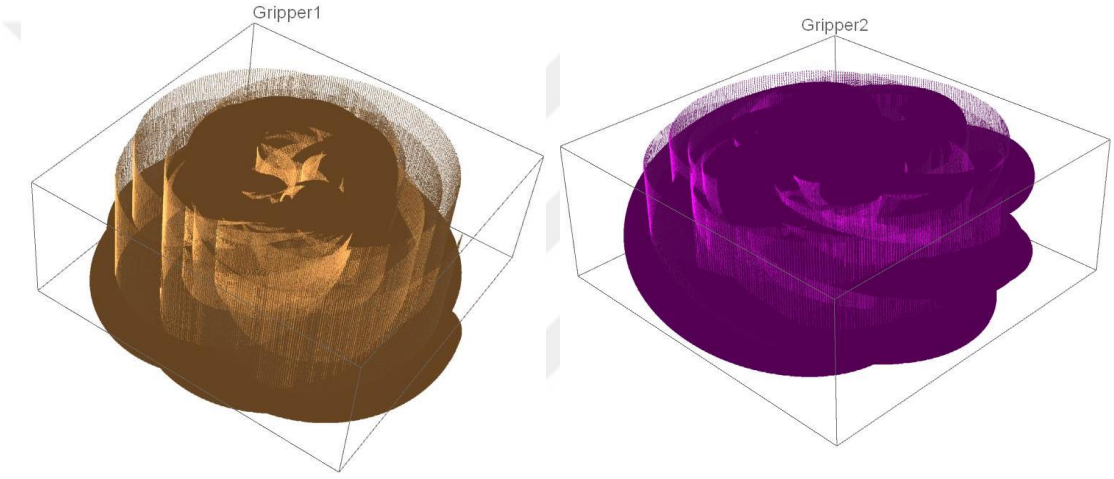


Figure 2. 15 Grippe-1, Gripper-2 Workspace Visualization

The forward analysis portion from section 2.1.3 is where the equations for this study were taken from. Additionally, equations 2.16, and 2.18 have been applied, respectively, for gripper 1 and gripper 2. We just used the fourth column of both matrices in this case because it determines where the end effector will be located.

As, we see from figure 2.15, the gripper one has semi ***prolate shape*** workspace which is a little bit longer than the second gripper. The reason for that is the wrist length which is in perpendicular to gripper one, it is longer than the link which is connected to the second gripper. Therefore, the workspace of the second gripper is closer to ***oblate shape*** which mean that the reachability of the gripper is just limited to the places where it is close to the bottom surfaces. We used Mathematic software to compute and plot the 3-dimensional workspace for manipulator. All required material and codes can be found in **Appendix A part 4**, under name of workspace analysis.

Chapter 3

3 Design and Implementing

In this section, the manipulator design, manufacturing using 3D printing, its parts assembly, the actuators used to actuate the arm joint and their benefits, the arm mechanism, and what are the purposes of its common shape will be presented in this chapter.

3.1. Parts CAD design using SOLIDWORKS

As we mentioned in previous chapters, this robot is an articulated manipulator, and its 4 DoF and all joints are revolute joints. The mechanical design of the mechanism was chosen following the conditions that the mechanism can be utilized on an experiment table because it was created for testing reasons. Additionally, it was discovered that all joints' limited rotation angles is relying on between 0 to 180[26].

Figure 3.1 depicts the CAD drawing of the created mechanism. There are some mechanical restrictions on the joints of the mechanical construction. [26]. These restrictions are provided to prevent robot manipulator damage while the motor rotates, and they are taken into account when performing kinematics and Jacobian computations. Due to the robot's intended application in agriculture, two grippers are made to be multifunctional and perform numerous jobs with ease and efficiency.

In our situation, we intended to use it during harvesting, so one gripper would hold the basket that would gather the product and the other would chop the branch to which the product was connected. It is simple to attach and detach the robot arm's base from the mobile robot. In another case, one gripper can hold a camera that might follow the end effector trajectory to be in contacting with the target and the other gripper could act on the target place depending on the feedback and information coming from gripper one.

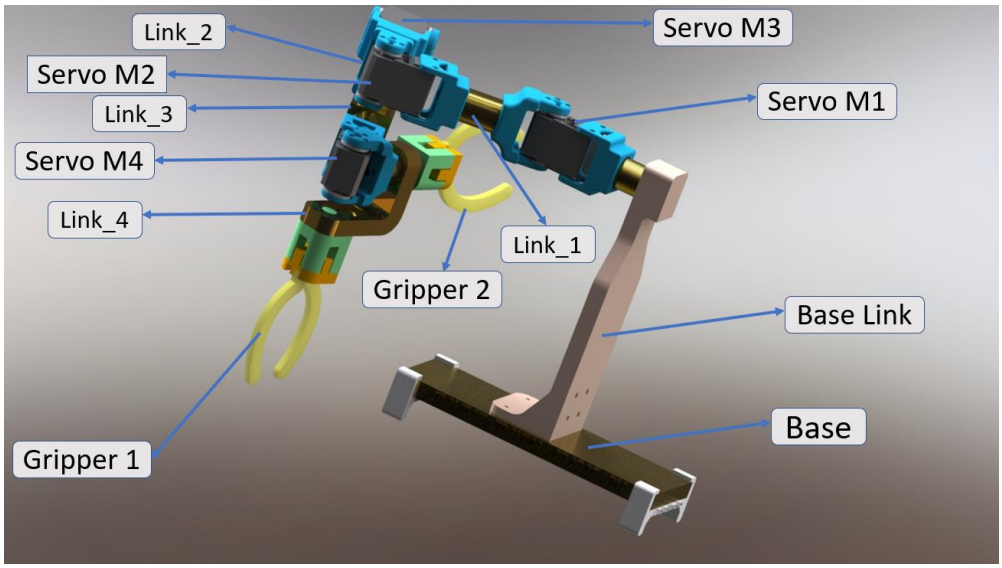


Figure 3. 1 manipulator mechanical design (CAD Design)

Adding to the previous features, instead of using prismatic joint to arrange the distance in Z direction, we replace it with common configuration so that the manipulator end effector could go through Z axis without facing any difficulties with wider workspace and more reachability to the target.

3.1.1. Links design

The manipulator base, base link, link 1, link 2, link 3, link 4, and the grippers holder on both sides are the linkages that we designed for the robot. When creating the connections, we take into account the need to include slots for the connectors that will join the motors so that the signal from the controllers can travel from motor 1 up to the end effectors.

Because we require the arm component integrated assembly to be symmetrical and as straightforward as we intend it to be, we chose a circular shape for the connections. Firstly, we design the base of the robot Figure 3.2, which will carry the entire robot body and it will be the connection between the robot body and other bodies. The dimensions of the base are 200x70x12 MM. The base can be fixed to the ground, to a mobile robot, or to any other platform easily without facing any difficulties

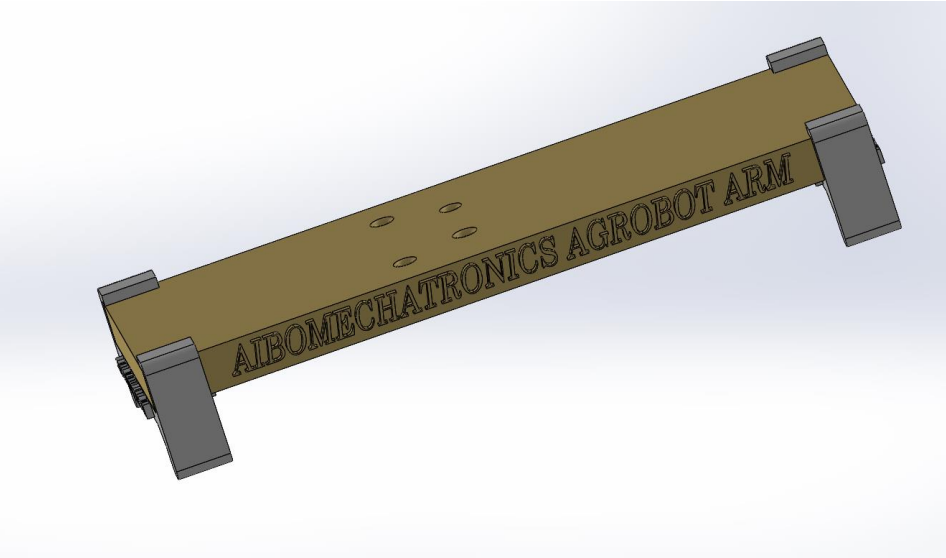


Figure 3. 2 Robot arm base

The link that will be fixed to the base is called the base link Figure 3.3, and it will displace the robot body on the z-axis and x-axis, and it is a compound of three different parts. The distance for this displacement is 200 mm on the z-axis and 58 mm on the x-axis.

These parts are designed for prototyping and to be suitable for printing them out using 3D printing. The parts are designed to be familiar to the user as a real robot arm so that we take beauty, industrial shape, and formality into consideration, like an already existing robotic brand.

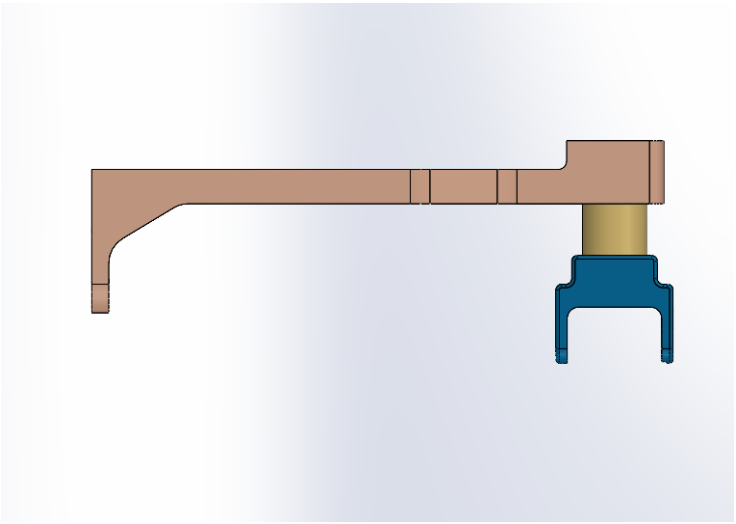


Figure 3. 3 robot arm base link

Link 1, link 2 Figure 3.4 of the robot arm has been designed to be identical in dimensions because the workspace that we need in the XY plane is not strictly important in variation.

The length of the links is 45 mm. We opened a tunnel for the servo motor cables from inside the links to keep the beauty of the robot high and allow the joints to move smoothly without colliding with the cables.

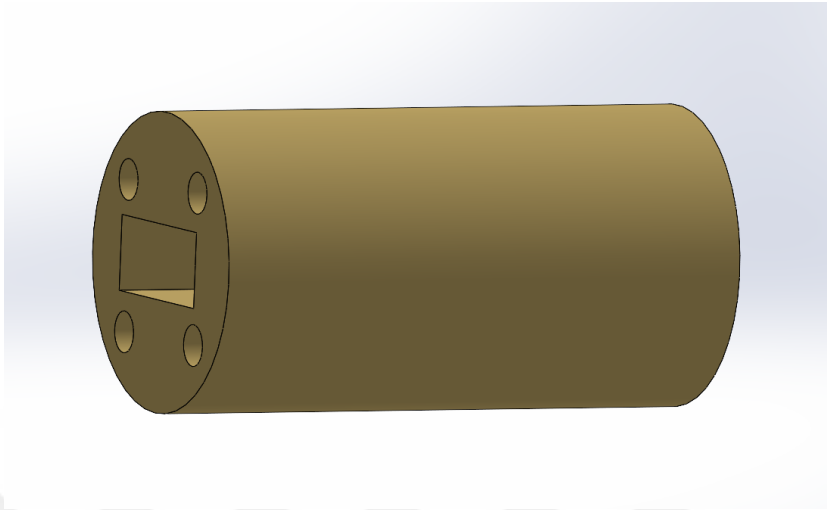


Figure 3. 4 link1,2 of the robot arm

The wrist of the robot has been designed to implement the movement in the z-axis and it is called link4 Figure 3.5 so it will replace the translation motion that the prismatic joint done in the SCARA robot mechanism with rotary motion.

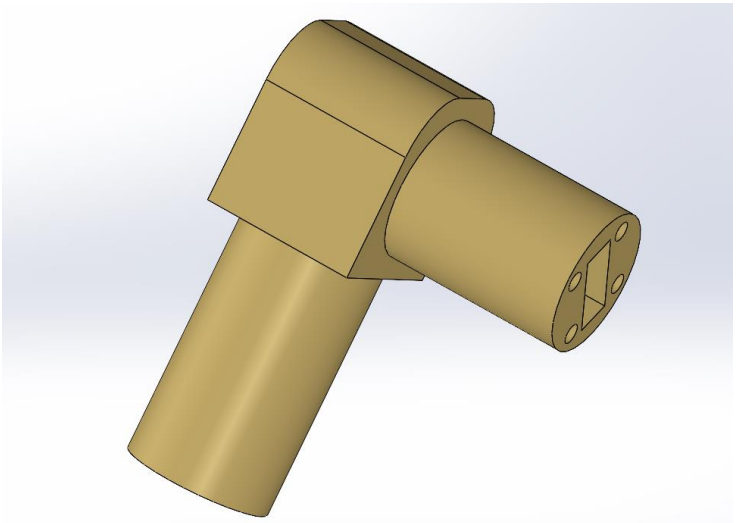


Figure 3. 5 robot arm wrist link

However, it will achieve the same goals. The dimensions of the link are 57x70 mm and it contains tunnels for cables too.

3.1.2. Joints design

LX-16A servo motor has designed its connections for the motor. However, it is a little bit expensive and limited for the purposes that the robot arm will be used so we designed our connections. These connections will carry the servo motor that will actuate the joints of the arm.

It has been designed to be suitable for the motor dimensions, holes, and edges so that it can be assembled and disassembled easily. It can be used for different tasks too. The backside connection Figure 3.4 is used to hold the servo motor from the backside and attach it to the previous link.

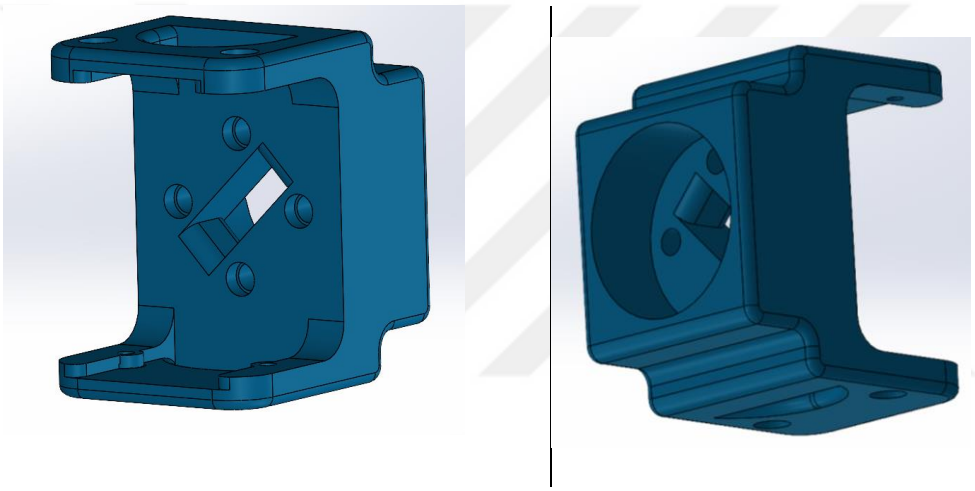


Figure 3. 6 servo backside connection

The component on the front side of the motor must be attached to the motor's revolute part and we called it the front connection Figure 3.5 so that it can be turned between 0 and 180 degrees, which is close enough for our desired workspace of the joints' work.

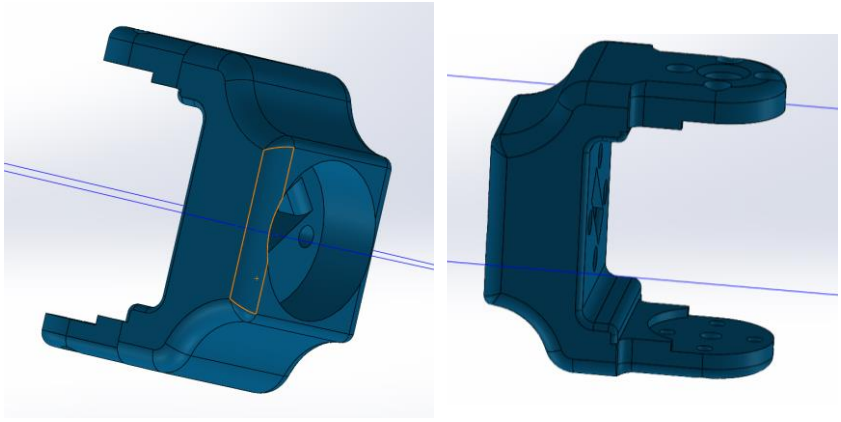


Figure 3. 7 servo frontside connection

When designing the joints, we consider the dimensions of the motor sides, cables, and screws that will hold these components together to simplify the assembly and disassembly of the robot when it is necessary.

3.1.3. Gripper’s design

Our robot has two grippers Figure 3.8 that will be used for farming and agricultural purposes, each of which has a single prismatic joint that allows it to be retained or released from the target while in operation. The two grippers will cooperate since they are perpendicular to one another. The first gripper is parallel to the fourth servo motor joint and the second gripper is coincident with it.

We were able to determine from this setup that the precision of the work can be ensured since even if the first gripper makes a tiny error, the second gripper may be able to minimize it and complete the operation as planned. On the first gripper, we will put the fertilizing machine that will be responsible for actuating the fertilizer to the plants that need it.

In this case, the pump will be triggered using a magnetic relay so that when the robot comes to the plant that needs to take fertilizer depending on the data that will be taken from the image using the camera and the required image processing and supply the required fertilizer to the plant that will help it to grow and kill the weeds and insects.

On the other hand, the other gripper will be responsible for harvesting the ready vegetables and fruits that are ready and put them in the basket installed on the mobile robot, and bringing them to the collection place. Any tube can be used for gripper one to transport liquid from the central unit to the robot arm's end effector so that it can respond to the actuator where it is positioned as soon as possible. The load on the robot arm will be reduced from this side, allowing it to operate with actuators that can handle a light load. On the other hand, gripper 2 can be easily replaced depending on the task for which it will be employed. For instance, we need to employ a little larger gripper if the robot is going to gather heavy, hard fruits like apples so that it can complete the task without any issues.

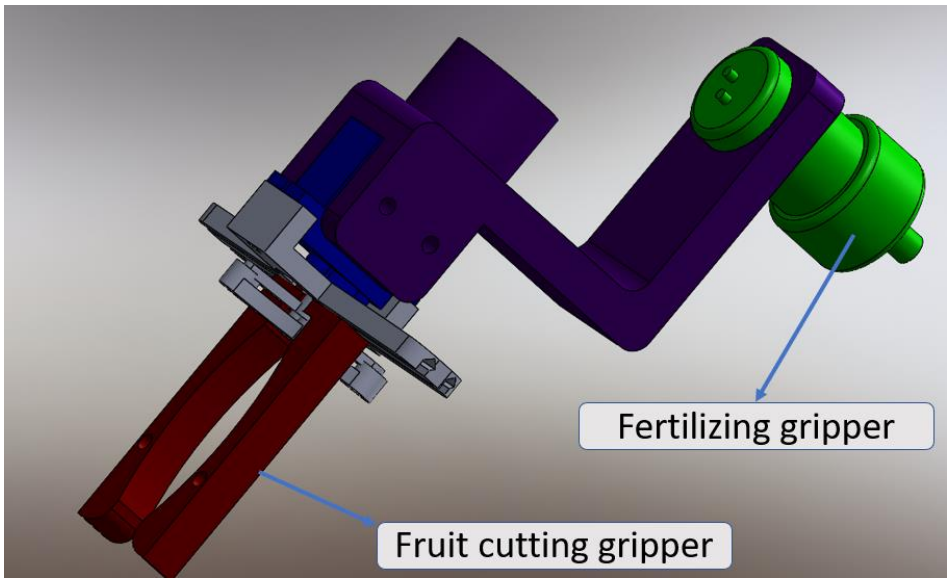


Figure 3. 8 end effector grippers

However, if it is going to be used for harvesting small fruits like strawberries, the gripper for this task should be designed as smoothly as possible so that it can carry out the work with no effect on the strawberry to be squeezed.

3.2. Actuator's selection (LX-16A Servo motor)

Servo motors are used in a very wide range of robotics applications, in control systems for quick operation, excessive axis movement, condition control, and other uses. Servo motors are utilized in conjunction with electronic or programmable circuits because of their high sensitivity.

There are two types of these motors which are AC and DC. The servo motors are brushed while the AC servo motors are brushless type motors. Most servo motors have three cables and everyone is for a different purpose. These are yellow for signal, black for ground, and red for voltage[27].In this project, we have selected an LX-16A servo motor as shown in Figure 3.1.

These servo motors were created and produced by the HIWONDER company. They were intended for robotic arm prototyping and were utilized in their unique pick-and-place robotic arm. It is a full metal gear serial bus servo with 17kg of strong torque. It has feedback on temperature and position too.

It uses LX-16A Bus Servo uses as a serial protocol, thus to control it, we must use our TTL/USB debug board is compatible with Arduino microcontroller[28].

Figure 3. 9 LX-16A Servo motor

Debugging board is designed separately by the company and it has to be used for calibration and control. No matter how many servos you are controlling, you only need one debug board Figure 3.3. The main function of the motor is simply depending on the setup that we have done before the system operation. Each servo can specify an ID number for identification.

The servo ID can be changed by the user. The controller and servo interact with each other via a single bus at a baud rate of 115200. Each servo can have an ID number assigned by the user, and the controller's command includes this information. Only the servos that match the ID number can receive this command and act by doing as instructed.

3.3. Servo controllers and drivers' selection

3.3.1. Bus servo terminal

While setting up the system and motors that will actuate the robot joints, we should use debugging board Figure 3.3 so that we can set the ID for every singular servo motor. We can instantly connect to the computer with the USB cable when the debug board is attached to the power source. The PC software can allow us to test the servo and configure its settings by connecting it to the TX and RX of a single chip, we can control a servo and read its angle using a debug board. Positive, negative, and signal wires are the three wires on the LX-16A servo, respectively.

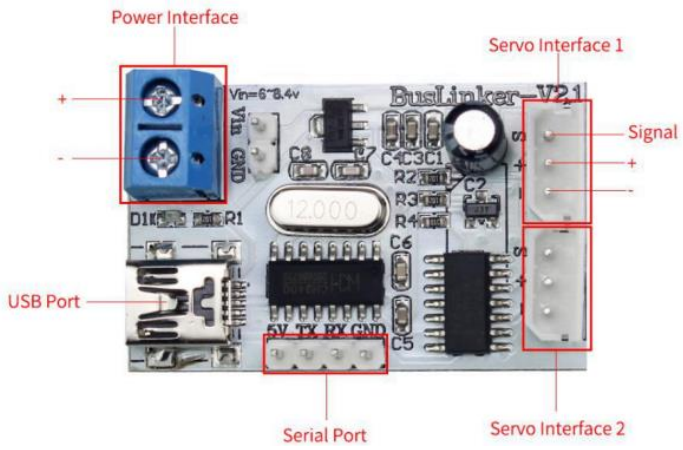


Figure 3. 10 LX-16A Servo debug board

Data can be sent and received simultaneously over this signal link. It's a little complex If we want to control the bus servo using a single chip. Therefore, the company of Hiwonder provides users with the BusLinker debug board, which can change the servo's serial port into a two-wire serial port. At the same time, we can set all the IDs of the servos that we will use in our system.

3.3.2. Bus servo controller

The bus servo controller Figure 3.4 is required to generate the point sets, either manually or by logging the angles of the joints of the robotic arm's initial and final position from the user interface designed by the Hiwonder company.

The motor can take values from 0 up to 1000 in servo modes, and it can be configured as DC motor mode as well. Hiwonder's intelligent serial bus servo controller can be combined with a PS2 wireless handle to realize servo remote control when utilized for serial servo control. It enables TTL serial port connectivity, manual programming, and online debugging. A comprehensive system may be fully and simultaneously set to function in complicated systems and in communication with ROS, which we are employing in our project, as well as with extremely popular programming languages like C++, Python, and MATLAB. These boards were chosen because they are simple to control, integrate, and/or communicate with other microcontrollers or computers.

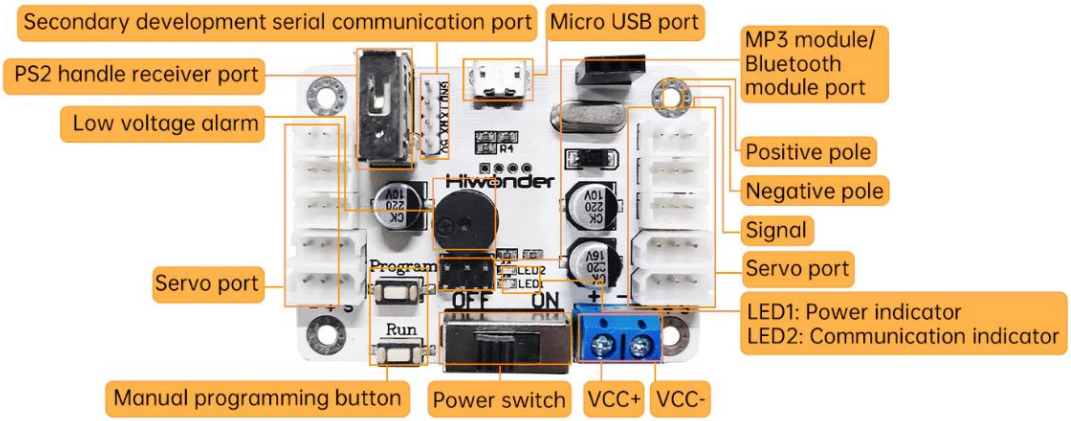


Figure 3. 11 Bus Servo Controller

Additionally, it is not necessary to wire every motor separately to the controllers; instead, setting their IDs is sufficient to send their designated signal to the precise positions where it is needed to command the motors to move. In addition, we use Rosserial protocol to enable the serial connection of the servo system of our robot arm for the LX-16A servo motor and the extra hardware elements such as the Arduino microcontroller to the ROS system.

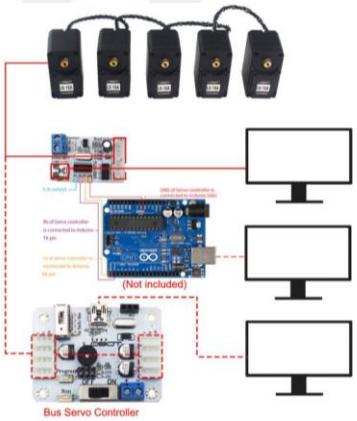


Figure 3. 12 .Connection Diagram of servo entire system

Through the use of serial libraries, the type of activity (Subscriber/ Publisher or Service) and a message format are directly established on the device. Additionally, a Python node on the host computer relays messages and makes ROS aware of any nodes generated on the device. Devices can have any functionality as long as they communicate utilizing the serial connection and the Rosserial format [29], see Figure 3.12.

Chapter 4

4 Controlling and Software

Every industrial robot has software installed, and they all depend heavily on it to function. In order to imitate robot work, virtual simulation is also necessary. Robotic arms have multiple degrees of freedom and can extend to increase accuracy and repeatability. The HMI interface, which must be given by the manufacturer or a third party in communication with the manufacturer, is also necessary to familiarize us with the robot when it is in operation. But the market needs a more dependable answer, one that gives the robot total autonomy and adds sensors for feedback.

To create a functional control system, the use of middleware packages and programs must be taken into consideration so that we can call the software environment between apps that interact with one another to carry out repetitive and accurate operations as a middleware. They never take any independent activity regarding to the tasks given to it. To select the suitable middleware for your system, you have to know all that system characteristics and specifications to decide which middleware you can use[30].

There has been research on middleware selection, and the pros and cons of each middleware package have been listed to help explain the decision. Typically, a certain operating system is required to execute and interact with robot middleware. It can be a virtual platform that controls the hardware and robot frameworks attached to the computer running the operating system. These operational systems might be Windows, macOS, and Linux or any other system that can run and manage the robot operations.

4.1. Robotic Operating System (ROS)

Being able to work with real robots is always a fantasy for robotics enthusiasts. These platforms were often quite expensive, and complicated, limited resources are available, and only large companies or research institutions could afford them. Additionally, it may be difficult to get a job in this field given its continued rapid growth and requirement-

intensive experience [31]. A meta platform, Robot Operating System (ROS) makes robotics programming easier and more reachable for developers. One of the goals of this thesis is to show how ROS may be used to create robot manipulator software by analyzing and computing the kinematics of robot issues, looking at existing packages, and creating a ROS-powered prototype arm[32].

ROS was created and developed as a specific software and framework unit for robotic platforms to be easily integrated with each other. In essence, many robotic tools offer message transport methods between various software nodes and hardware simulation. Nodes are separate modules that can function on their own and/or in communication with other system.

They communicate on the aforementioned topics utilizing the TCP/IP protocol and a one-to-many subscriber paradigm. When working with networked systems, this ROS capability is crucial and extremely important to robotic system.

For instance, the functionality needed for only one robot operation, specifically the parts for the sensor and actuator controls, can significantly rise and strictly important for the robot operations. A few of the applications that can be achieved with this distributed sensor network are motion planning, trajectory planning, collision detection, mapping, navigation, and obstacle avoidance.

Each system has a name, they call it as a server, also known as a ROS-Master, that notifies nodes of which node is publishing to which topic that we are interested in. As a result, every node needs to save the issues it sends to the ROS-Master. Another node that wants to subscribe to a topic asks ROS-Master for the contact information of pertinent publications and gets in touch with them. As in a peer-to-peer network, the nodes link directly after the initial phase of searching and start operate on it.

Robot arm path navigating and planning is one of the heavily researched areas in a robotics automation system, which deals with planning and moving toward the goal of avoiding obstacles surrounding the environment, Using the MOVEIT system in the ROS platform, path planning, and robot kinematics integration is the main focus of this study. For complete autonomy, ROS provides a variety of nodes and packages to build the system's functionality. Taking advantage of this, we intend to assess the ROS path

planning system for our objective, which is picking and placing crops for harvesting purposes in agriculture tasks[33].

Most of ROS's code is written in C++ and Python languages. Therefore, Client libraries for C++, Python, Java, and/or MATLAB are among the languages supported by ROS nodes. For communication between message delivery systems and protocols, any programming language is acceptable for ROS-Master. Additionally, other operating systems only partially support the ROS operating system, which is primarily supported by Ubuntu/Linux. However, ROS's modular design makes it simple to develop a variety of setups for varied scenarios.

The thesis study's codes underwent testing on both a real system and a simulation. The URDF file, image processing code, and MOVEIT module must be modified in accordance with the robot's mechanical, motion, and sensor model in order for these experiments to be performed on additional ROS-enabled robots.

To deal with any robotic manipulator using the ROS platform, there are a few procedures that must be taken to achieve the goals that robotics engineers and researchers are planning to achieve. Those steps are going to be discussed in this chapter and it will be like the following[34].

- Preparing the system environment and operating system
- Preparing the robot's physical appearance and configuration
- Preparing the script, packages, and codes that will run the system.
- Nodes
- Topics
- Actions
- Combine them

4.2. Preparing ROS Environment and its operating system

A robotic operating system (ROS) is a versatile and open-source software framework. Developers can create robotics apps using ROS' hardware abstraction layer without having to worry about the underlying hardware. In addition, ROS offers a variety of

software tools for analyzing and debugging robot data. The message-passing middleware at the heart of the ROS framework enables processes to talk to one another and exchange information even when they are running on different machines. Both synchronous and asynchronous message forwarding is possible with ROS [35].

4.2.1. Ubuntu 18.04 installation neither in virtual Box nor in dual bot

It is an open-source Linux distribution built on the Debian operating system. Ubuntu, which is supported by Canonical Ltd., is regarded as a decent distribution for novices. Although it may be used on servers, the operating system was designed primarily for personal computers (PCs). The word "ubuntu," which means "humanity to others," is derived from the African Zulu language[36]. Therefore, to download and install Ubuntu (Figure 3.1), we have to follow some steps as follows[37]:

Figure 4. 1: Ubuntu operating system

- Preparing the PC' DVD and USB, at least 25 GB must be free space
- Boot from DVD, or USB flash drive (*Figure 3.2*).
- Preparing for installation and allocating drive space (*Figure 3.3*).
- Following the steps that Ubuntu suggests user do the required settings (*Figure 3.4*)
- Start Ubuntu system testing

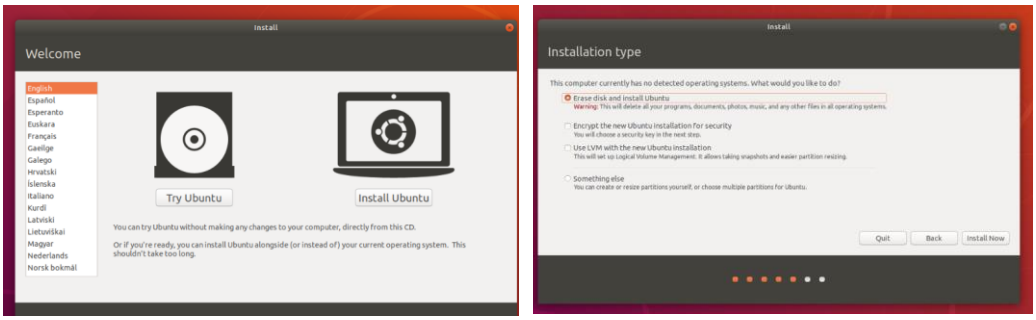


Figure 4. 2: Ubuntu Booting and Ubuntu system allocating

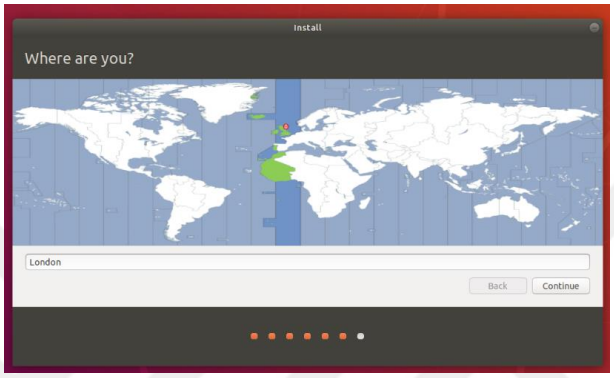


Figure 4. 3: Ubuntu system settings and preparation

These steps are must for installing Ubuntu and working with robotic operating systems (ROS).

4.2.2. ROS Melodic installation

Robot Operating System (ROS) is A collection of software libraries and development tools that are used to create robotic devices and programs. Melodic Morenia is the most recent long-term service version of ROS. It only works with Ubuntu 18.04 Bionic Beaver when using ROS Melodic[38].To install and work with this tool, we should follow the steps below[35] :

- Open a terminal and update the Ubuntu system using the command
  $ sudo apt-get update
- Install ROS full melodic package using command
  $ sudo apt-get install ros-melodic-desktop-full
- Initializing ROS dep using commands bellow
  $ sudo rosdep init
  $ rosdep update

- Setting up the ROS environment where the user will work, writing the following command into the terminal and copying the next one inside the file that will be opened .

  .bashrc

  source /opt/ros/melodic/setup.bash

4.2.3. Setting and preparing ROS workspace (Catkin_ws)

A catkin workspace is a directory (folder) where you can build new catkin packages or edit ones that already exist. The catkin structure streamlines the creation and setup of your ROS packages[39].To achieve this milestone, we should follow the next steps[35] .

- Open the terminal and create an empty workspace folder and open another file inside called src, using the following command.

  $ mkdir -p catkin_ws/src
- Switch to src and initialize the workspace using the command bellow

  catkin_init_workspace
- Switch to catkin_ws itself and build the packages inside it using the following command.

  $ catkin_make

4.2.4. Setting and preparing ROS packages

A ROS package could include any item that logically belongs in a useful module, including a library, a dataset, configuration files, etc. These packages' main objective is to make the functionality accessible to users so that software can be reused with ease[40].To create ROS packages, we have to follow the coming steps[41].

- Open the terminal and go to catkin_ws/src folder using the command bellow

  $ cd ~/catkin_ws/src
- Create the folder that will contain the project/package name and its required dependencies using the following command

  $ catkin_create_pkg progect_name std_msgs rospy roscpp
- Switch to the folder of catkin_ws and build the package using the next command

  $ cd ~/catkin_ws

  $ catkin_make

The project package will be ready once all of these processes have been completed, and we can then add any further scripts or information that will be required for the project's implementation.

4.3. Preparing the physical characteristics of the robot

In simulation space, it is necessary to have a physical characteristic for the robot and the URDF (universal robot description file) is a package used as a parser for the robot description file structure in XML file, it was created and used as part of that structure. There is a tree structure in the robot definition file. There can only be one primary link in the system named after the tree structure, to which a link is attached, forming a chain.

All of the infrastructures that make up the robot are defined in the file structure for the robot definition. The robot arm kinematic Model has to be equivalent to the real robot. Therefore, the robot model in ROS contains crucial packages and some important nodes that aid in creating the 3D robot models in the virtual frame. These packages employ the Unified Robot Description Format (URDF). So, the manipulator model is described in the URDF, which is an XML specification.

To perform robot work, we have to prepare the inertia matrix, collision detection matrix, robot joints and arms visualization, the transmission of joints actuators, reduction rate, actuators that will be used, gazebo plugin, sensor plugin, and required controllers that we may use during task implementation.

4.3.1. SOLIDWORK to URDF preparation

In this section, we have prepared the manipulator parts to be appropriate for prototyping and printing after being converted from CAD to extension STL. The components are made to look like genuine robot arms to users so that we may address aesthetics, industrial design, and formality, just like an established robotics company. Using the SOLIDWORKS CAD application, a 4-DOF manipulator with 2 distinct end effectors was developed to harvest and manipulate the objects.

Then, we installed the Unified Robotics Description Format (URDF) as a CAD plugin using SOLIDWORKS. Thanks to the SW2URDF plugin for making it simple to convert the design to ROS and begin controlling it. This plugin assists in converting CAD-

designed robots into a format that ROS can read and show on RVIZ and Gazebo in integration with MOVEIT.

To carry out all these processes, we can follow the following procedure.

firstly, top bar menu in SOLIDWORKS, we can click the tools manager and select the export as URDF plugin, see Figure 4.4.

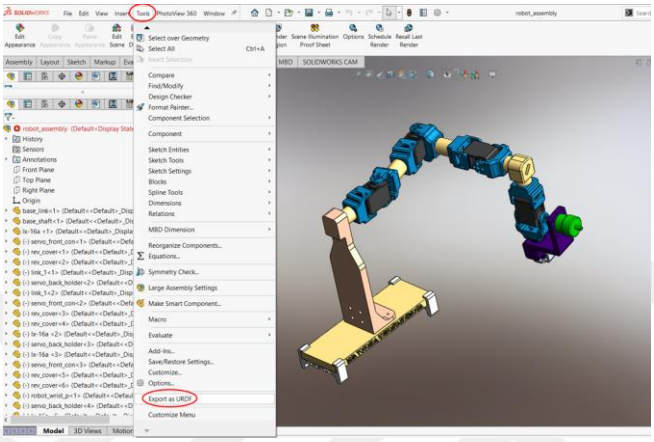


Figure 4. 4: URDF exporter selection

When we press export as URDF, the URDF setup exporter will pop up to show us the space where we will set the values and parameters of the manipulator properties and names, see Figure 4.5.

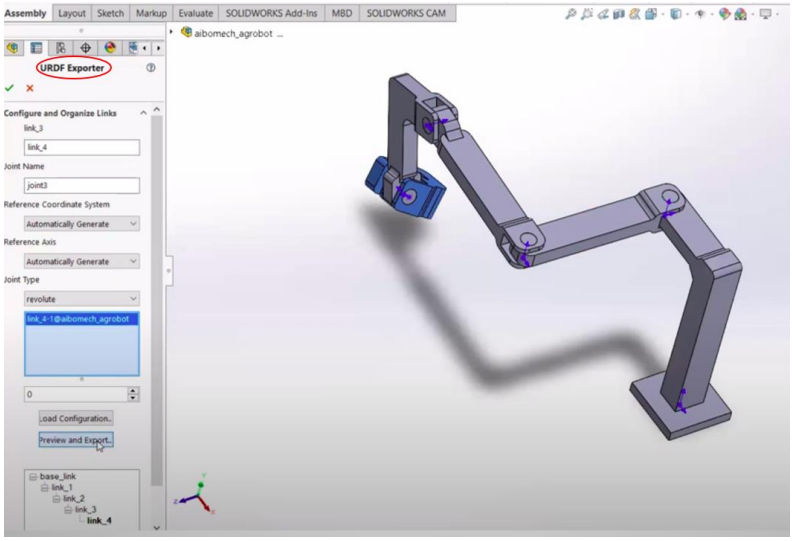


Figure 4. 5: URDF Axes and joints naming

However, the plane of the links and joints must be in top side of view of SOLIDWORKS during design, see Figure 4.6, otherwise, there will be a confliction after launching the file in ROS workspace. Therefore, each connection will need to be manually configured,

and the tree will need to be built. Each connection must have a distinct name, a distinct joint name, and it can be done by selection of that links and joints one by one, and the necessary number of child's and parents

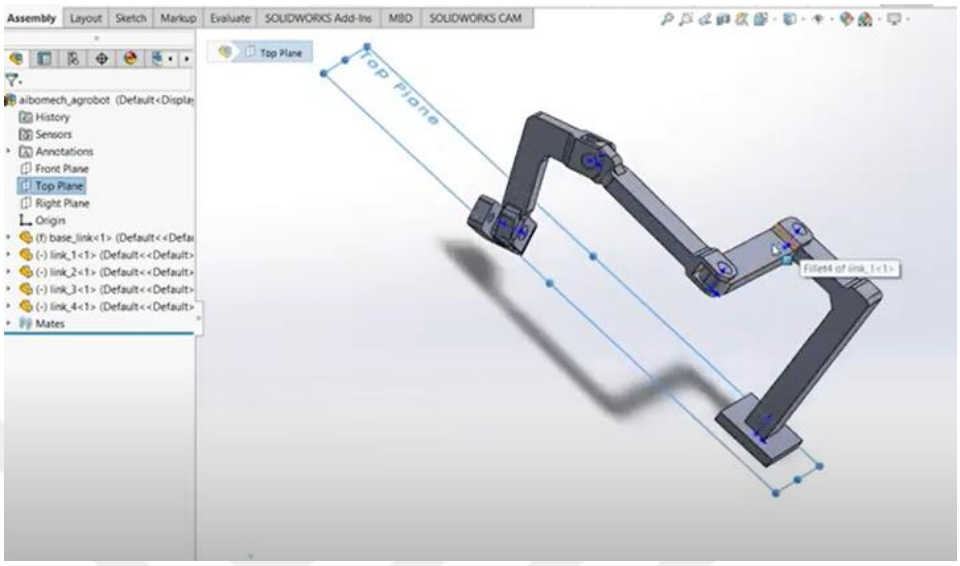


Figure 4. 6: SOLIDWORKS parts plane selection

In SOLIDWORKS, to avoid the errors and mistakes that may face us in ROS, we have to add the references coordinate systems to the links too, see Figure 4.7. Additionally, you can expressly state which joint type and for each joint belongs to.

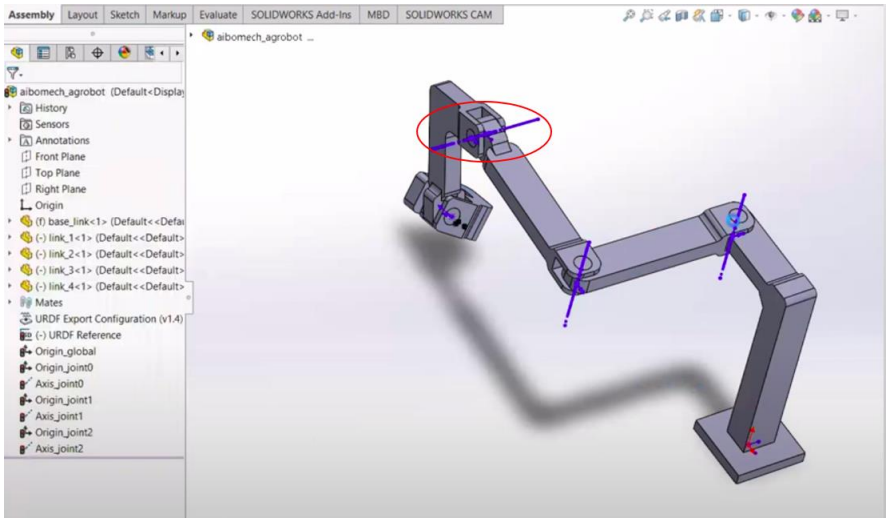


Figure 4. 7: Placing axes on SOLIDWORKS design

We can adjust the joints properties, such joint name, joint type to be prismatic or revolute or continuous, and link name, see Figure 4.8.

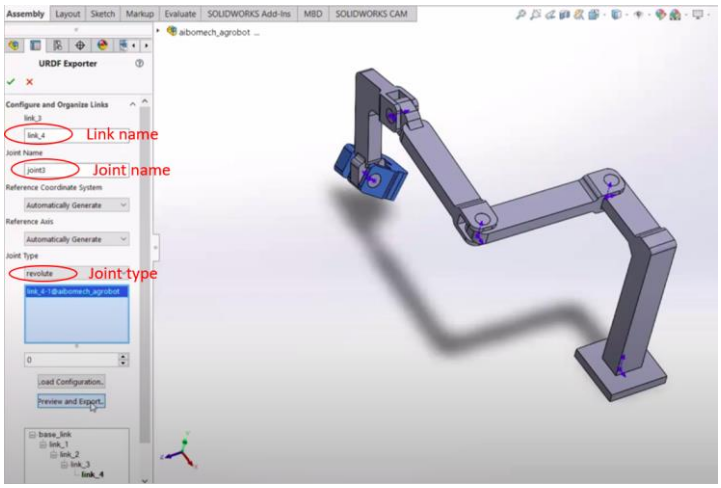


Figure 4. 8: defining the robot parameters

To decide the amount of rotation if it is revolute joint, and the limit of translation if it is prismatic joint. In addition to that, to decide the joint speed and acceleration. We can modify the link properties of any connection in our design. The mass, the moment of inertia, the origins of various sections, the texture, the color, and other characteristics can all be changed, see Figure 4.9.

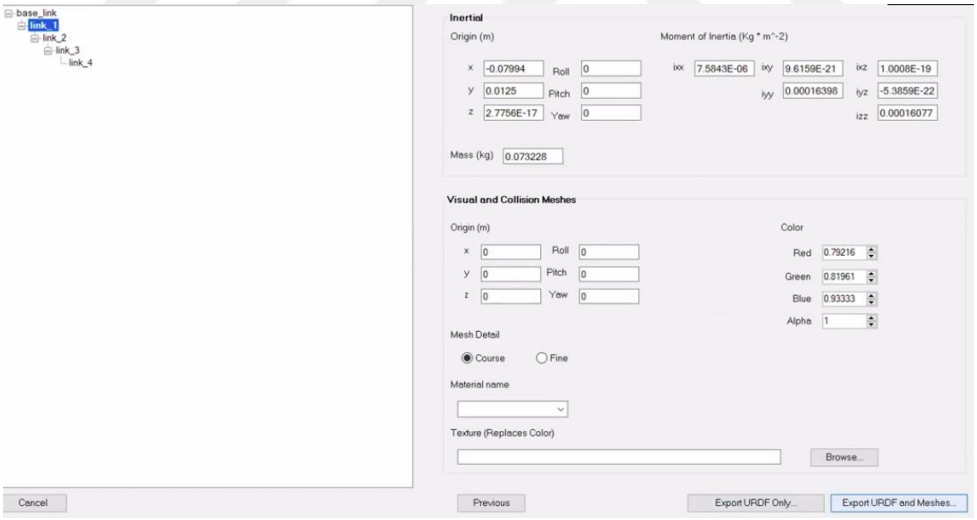


Figure 4. 9: Setting up robot links properties

We can change the reference coordinate systems and axes after exporting the files to URDF version to match our needs.

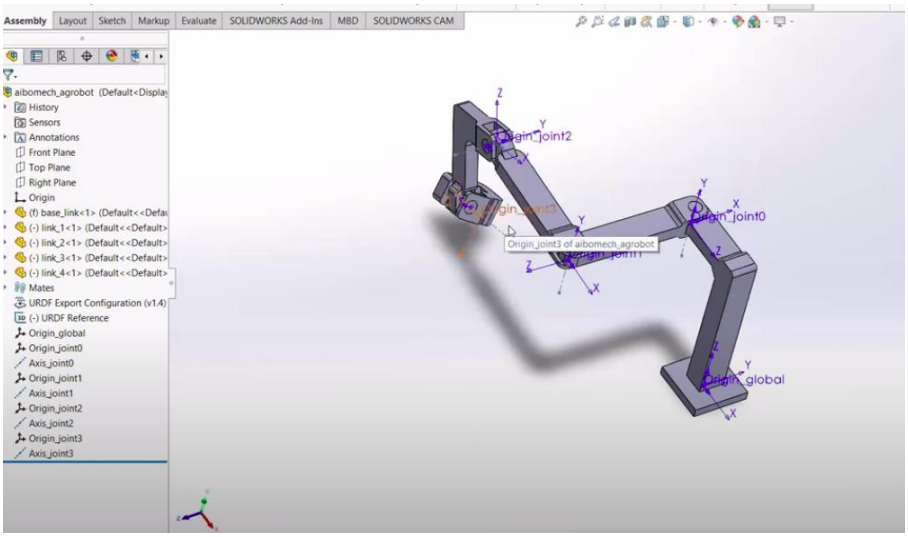


Figure 4. 10: Saved version of URDF file

The axes and coordinate systems can then be changed inside URDF files. After preparing all settings and configuration, we can select "File"> "Export to URDF" from the menu bar, see figure to start the export procedure and save the file with the name that we want, see Figure 4.10.

4.3.2. AIBOMECH Agrobot packages preparation

The bundle for the articulated ***AIBOMECH Agrobot*** includes every folder needed to run the robot application and carry out the tasks we have in mind. It has five folders which are configuration, launch, meshes, script, and URDF.

The physicality of the robot components is handled by the URDF package, while the system's operation and package management are handled by the launch file. The configuration file is in charge of directing and facilitating communication between nodes. Meshes are in charge of the robot's STL 3D components. The Python code we built to compute, perform, and carry out the actions we intend to do is contained in the script file.

In order to create these packages and make them executable, we have to follow the next procedure, with writing the required commands.

Firstly, we open the terminal in Linux operating system and go to the catkin_ws/src directory by writing the following command.

$ cd catkin_ws/src/

Then, we have to create the package with its required extensions, by writing the following command.

$ catkin_create_pkg aibomech_agrobot std_msgs rospy roscpp

Then ,by going back to catkin_ws by writing the following command .

$ cd ..

After that, we can execute and build the package that we name it by writing the following command.

$ catkin_make

After we create the package inside catkin_ws, there will be two files which will be responsible for files management and the communication and control between packages. Those extensions are CMakeLists.txt, and package.xml.

We have to go to the directory where we will create the packages files by writing the following command.

$ cd catkin_ws/src/

To create URDF folder writing command

$ cd mkdir urdf

To create launch folder writing command

$ cd mkdir launch

To create config folder writing command

$ cd mkdir config

by going to inside every folder and and creating the files that we will write the codes inside it by writing the following command.

$ touch TextName.txt

Inside all these files and folders, we will have Gazebo, RVIZ, MOVEIT, and other control files which will be necessary to run the system.

4.3.3. *AIBOMECH Agrobot* MOVEIT package preparation

In this package, we will prepare the package of MOVEIT that will be responsible to manage and execute the path of robot arm. It can provide the required files for the robot to do the motion that we are planning to do, to integrate with the inverse kinematics that we done before, and to do path planning[14]. The MOVEIT has been designed by developers to simplify the methodology of robot arm controlling. In this case we have used the MOVEIT setup helper tool. We used the robot definition file in the URDF format that has been generated from SOLIDWORKS. Before beginning the setup, ***roscore*** must be started by writing the following command.

$ roscore

After running the ***roscore*** from terminal, we can open another terminal to execute the MOVEIT setup assistance by writing the following command.

$ roslaunch moveit_setup_assistant setup_assistant.launch

A new window will be opened with name Moveit! setup assistance, see Figure4.11.

Then, we can start load the model of our robot by clicking on the icon called create new Moveit configuration package, another window will open to us.

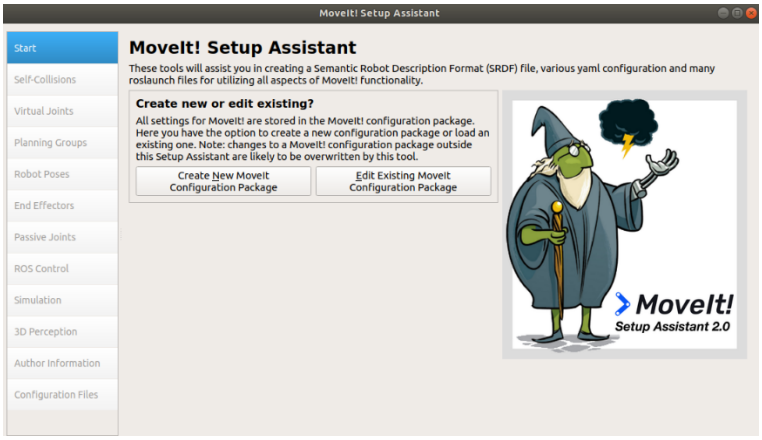


Figure 4. 11: Moveit! Setup Assistant

From the window, see Figure 4.12. We can select the files of URDF that we prepared before. Then, we can click the button of load file at the same windows, to parse the URDF, and the robot will be appeared at the right side of the window.

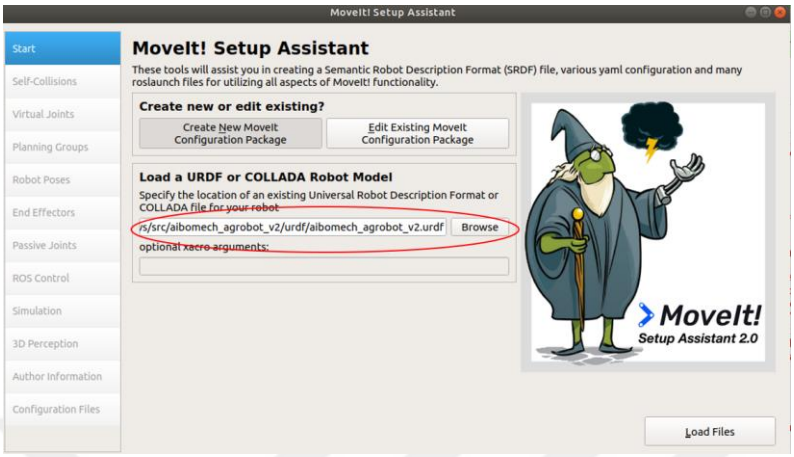


Figure 4. 12: Moveit! Setup files loading

After parsing the files that we load from the packages, we can start working on our robot arm preparation and installation to be ready for tasks that we need to achieve.

Firstly, we can start from self-collision matrix generation so that the joints and links can be linked together to be ready for manipulation, see Figure 4.13.

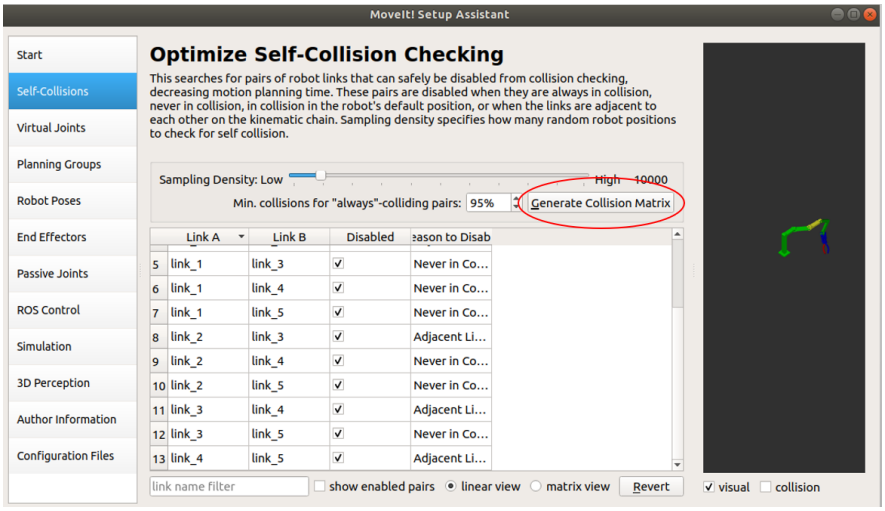


Figure 4. 13: Self-collision matrix generation

The second step is to generate and prepare planning groups that will help us to shape and compute the trajectories that we are going to achieve, by clicking on button called *Planning Groups*, see Figure 4.14.

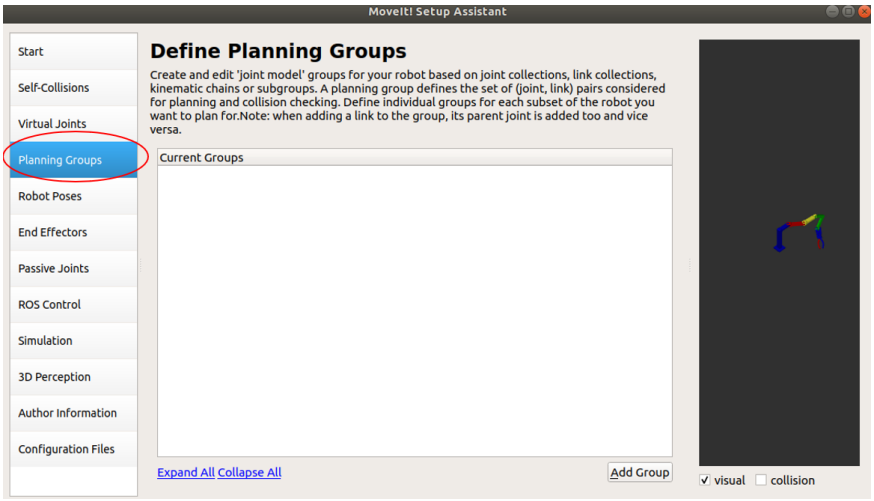


Figure 4. 14: Moveit planning groups

From define planning groups, we can create new planning groups using kinematic convention, we name the group with robotic arm, and choose **kdl_*kinematics_plugin/KDLKinematicsPlugin*** with ***RRTstar*** and click on ***Add Joints*** button to generate that group, *see Figure 4.15, Figure 4.16.*

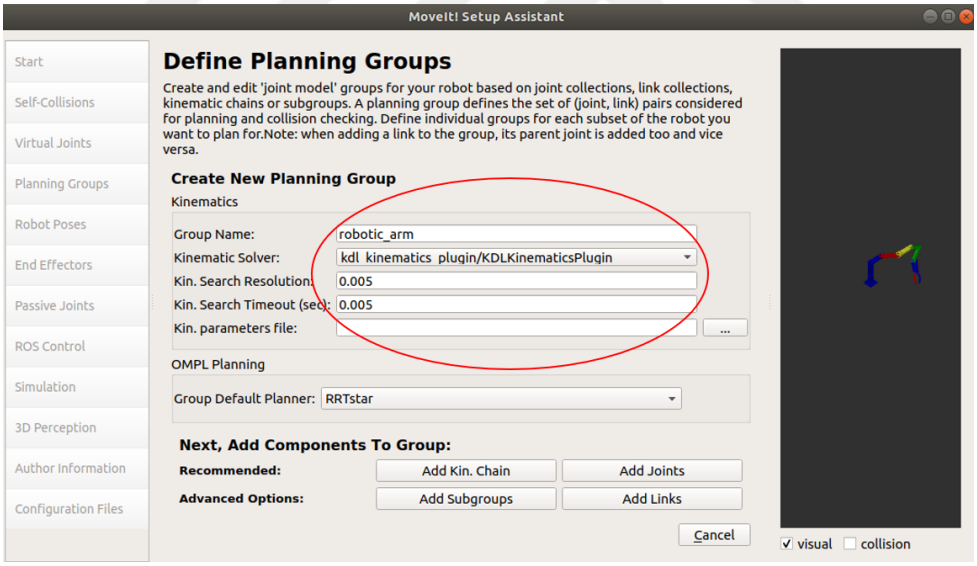


Figure 4. 15: Planning Groups setting

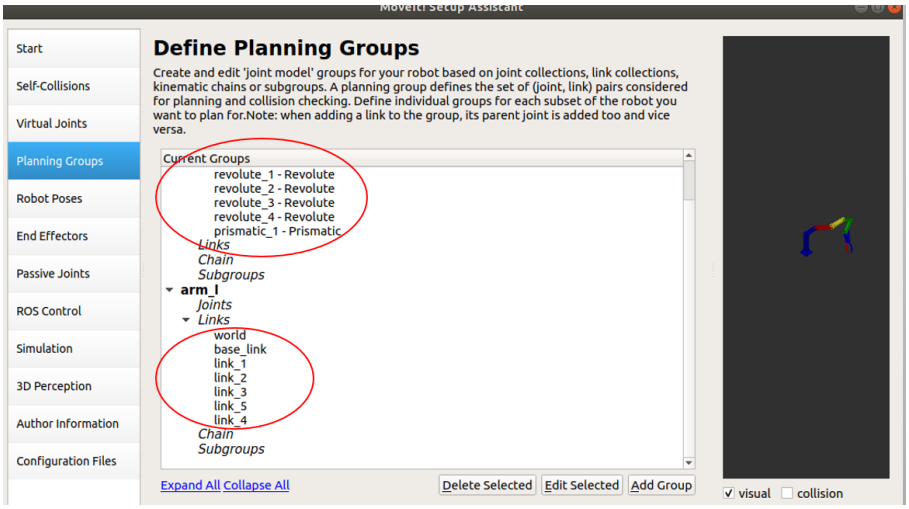


Figure 4. 16: Planning Groups setting

The third step is to prepare the poses and the tasks that the robot will do by clicking the robot poses button on the left side of the opened window, see Figure 4.17.

Figure 4. 17: Robot Poses preparing

From the poses window, we can select the pose name with *target1* and pick up the group that we prepared before from the planning group section. Then, we manipulate the joints of the arm regarding to the tasks that we need to do, see Figure 4.18.

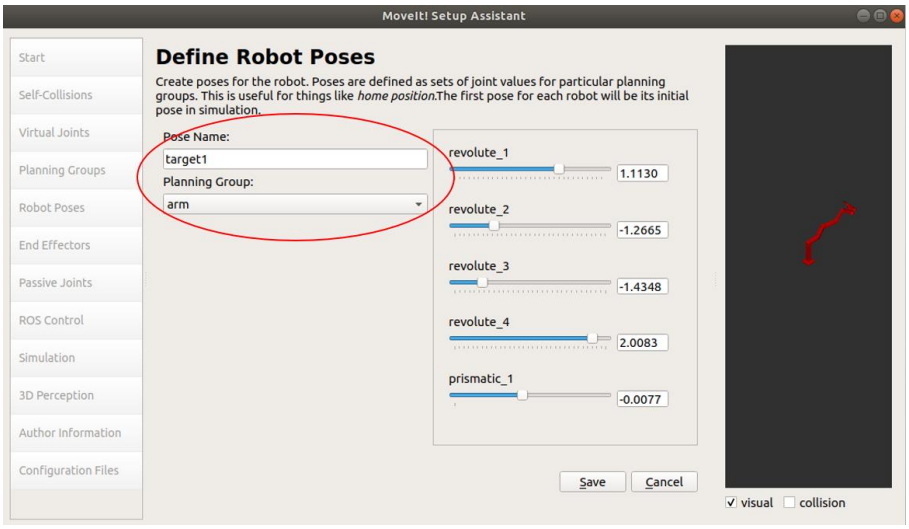


Figure 4. 18: target1 task preparation

For the target2(Task 2), we select arm1 group which we prepared it in groups section so that we can displace the joints position and orientation from the bar menu shown in the right side of robot poses, see Figure 4.19.

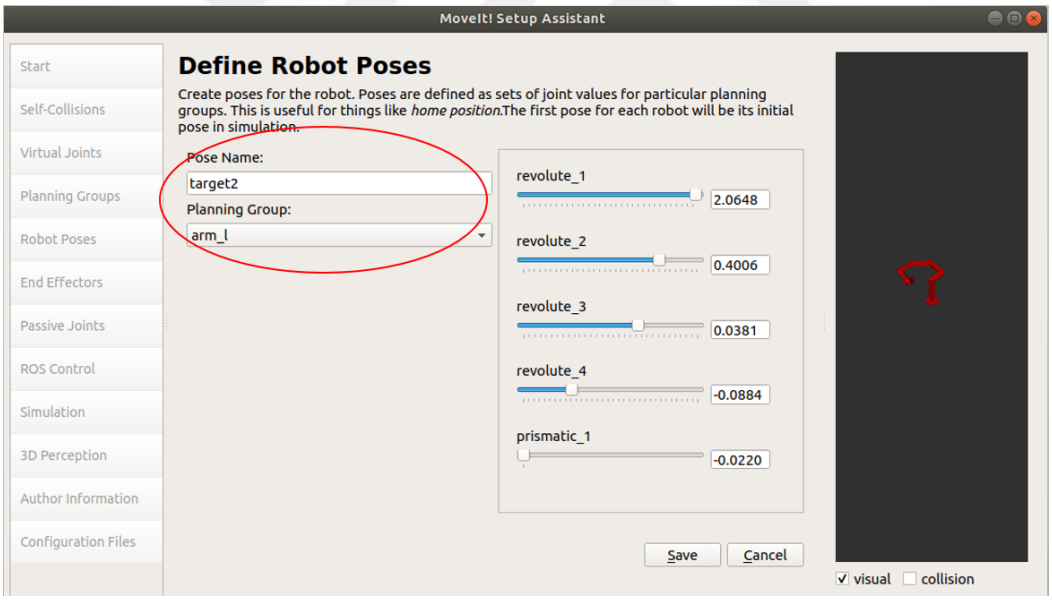


Figure 4. 19: target2 task preparation

In addition to the previous steps, ROS control must be added and prepared to the MOVEIT package by clicking on ROS Control button, and click on *Auto Add Follow Joints Trajectory Controllers for Each Planning Group,*to generate controller for the groups that we prepared before, see Figure 4.20.

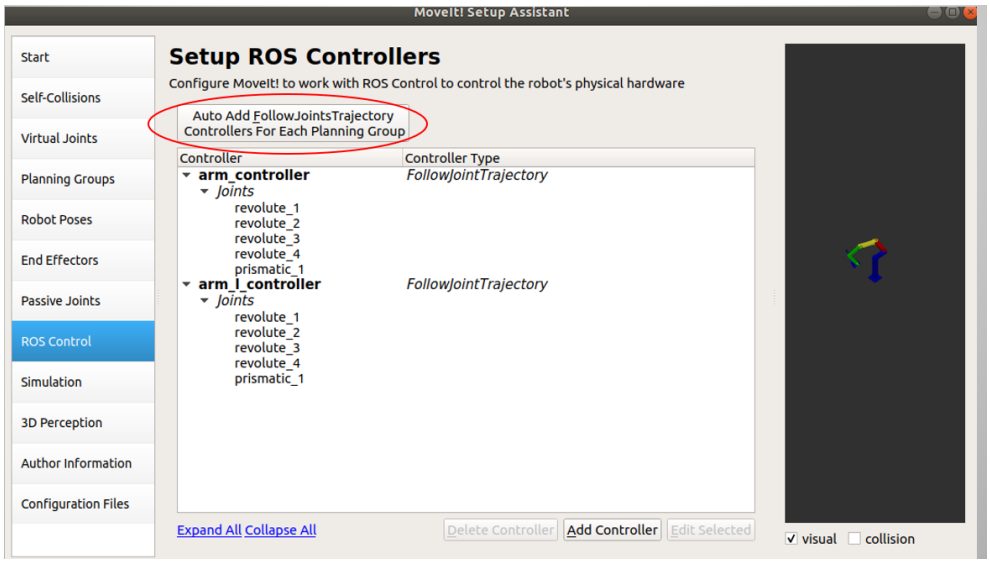


Figure 4. 20: Setup ROS Controllers

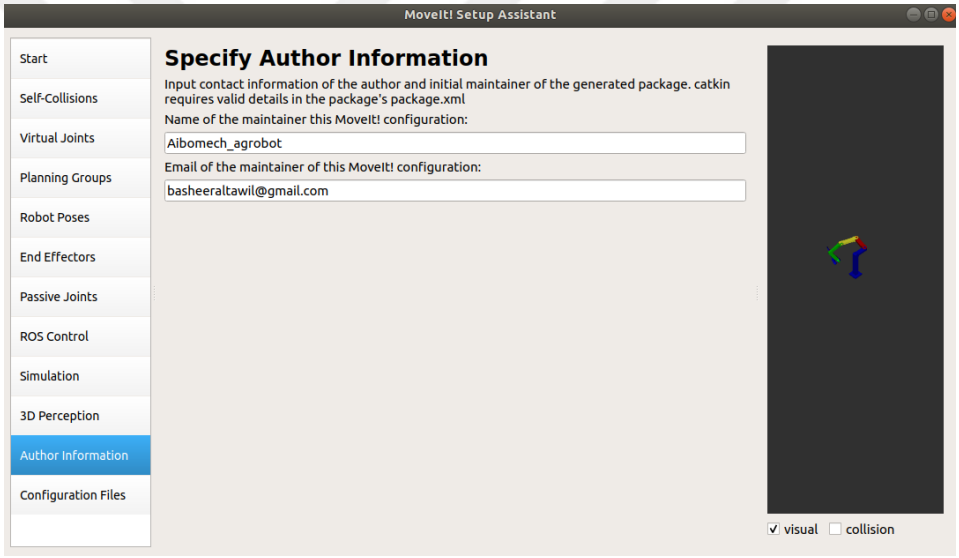


Figure 4. 21: File configuration Saving

Finally, we have to save our package that we prepared in a separate file to be ready for launching later, see Figure 4.21. ROS Packages and codes can be found in **Appendix D**.

4.4. Building the main structure of the control algorithm

Automation of labor is currently in great demand in contemporary greenhouses, orchards, plantations, and woods, and agricultural systems. The number of skilled workers willing to accept monotonous activities is steadily dwindling and the jobs are getting harder for

them. The working environment in greenhouses also has difficult climate conditions which may affect the human being health system.

The greenhouse industry's economic viability is under threat and getting few productions as a result of the rise in labor costs and decreased capacity[42],[3]. Developing scientific knowledge and experience for a highly configurable, modular, and clever carrier platform that includes a modular manipulator and intelligent tools (sensors, algorithms, sprayers, and grippers) that can be quickly installed onto the interested system and are capable of adapting to new tasks and conditions is one of the main goals of the ***Aibomech Agrobot*** project.

The robot arm will move following the requirements of the plant and the plant's readiness based on the feedback we will gather from the system we are using. Because of this, the entire governing algorithm will operate as indicated in Figure 4.22. The inputs for the system's master can come from a variety of sources. The expert who visits the farm daily, weekly, or even monthly can make use of these resources. Additionally, it can be obtained through the sensors that were mounted on the system during installation. These sensors may be vision sensors, such as cameras, or they may be digital, such nutrient solution sensors or soil analyzer sensors.

After receiving the inputs, we may examine them in the data base we set up on the backend of our software frame and provide the instructions for the robot to follow in order to complete the required work.

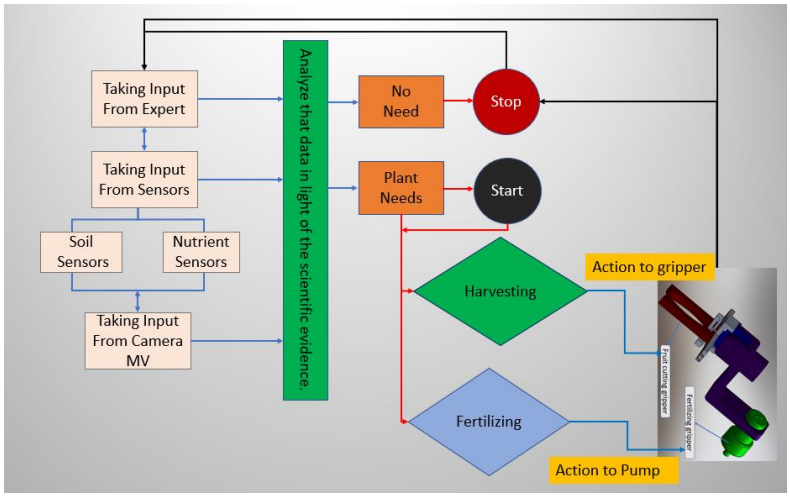


Figure 4. 22: Entire system controlling Algorithm

Modern robotics places, and come up with a strong emphasis on the close relationship between morphology, or outward appearance, and highly intelligent behavior. The behavior is seen and utilize as the outcome of sensors driving actuators in a sensory-motor loop controlling system that is inseparably mediated by the environment and controlled by the right selected middleware system.

While emphasizing the importance of hardware which is running the system. This viewpoint also suggests that the software component that links sensing components with acting components is principally responsible for flexibility and goal-directed agricultural intelligence system[3].

4.4.1. Building the architecture of the ROS controller

The robot arm kinematic Model has to be equivalent to the real robot. Therefore, the robot model in ROS contains crucial packages and some important nodes that aid in creating the 3D robot models in the virtual frame. These packages employ the Unified Robot Description Format (URDF). So the manipulator model is described in the URDF, which is an XML specification. To perform robot work, we have to prepare the inertia matrix, collision detection matrix, robot joints and arms visualization, the transmission of joints actuators, reduction rate, actuators that will be used, gazebo plugin, sensor plugin, and required controllers that we may use during task implementation.
Following the preparation of the packages we previously discussed, as shown in Figure 4.23, we could launch the files using a Linux terminal and visualize our robotic arm to interact with it using RVIZ ,GAZEBO, and MOVEIT interfaces. There are two packages to deal with to interact with the robot arm which are joint_state_publisher and robot_state_publisher. These packages are responsible for transforming the internal case of the joint of the robot arm so that we can move, stop, and visualize the status of every singular joint by helping these two packages.

MOVEIT, which enables us to operate robotic arms and create the desired trajectories, is one of the most intriguing packages. The procedure that MOVEIT work basically depends on robot kinematics and dynamics KDL, and inverse kinematics of the robot arm so that by bringing the end effector of the robot arm to the desired position in different places, we can implement and repeat that action as much as we can, see Figure 4.24.

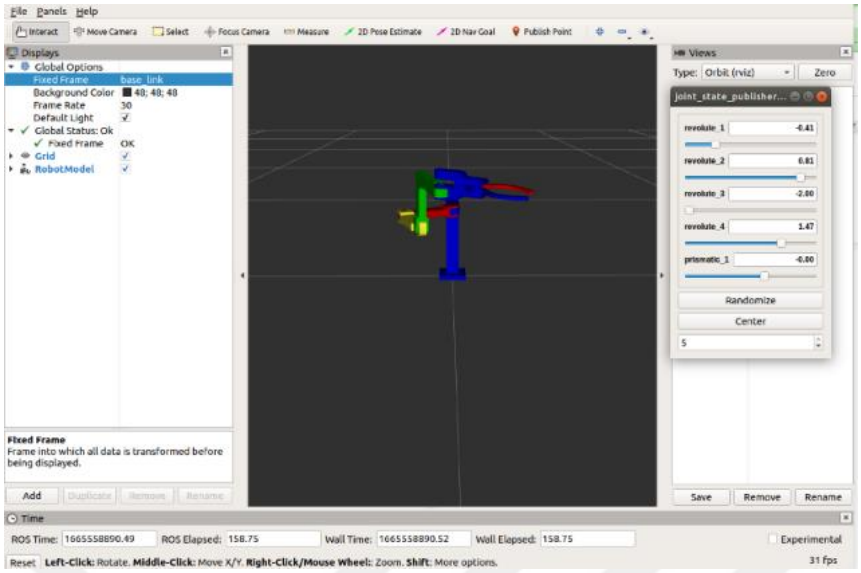


Figure 4. 23: RVIZ with its joint publisher GUI

As we have already indicated, there are already created packages available for editing, modifying, and improving.

4.4.2. Controlling the inputs coming from the agricultural system

As we mentioned before, we will get some data from some sensors, experts, and/or cameras. These data will help us to decide where the robot arm should go. Therefore, after collecting all these data, the robot should know which plant needs to be getting some operations such as harvesting and/or pesticide spraying as we mentioned before.

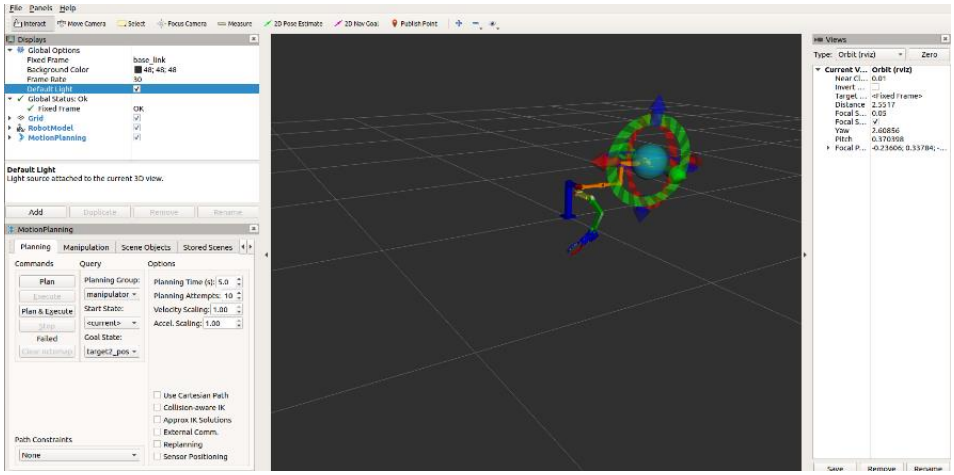


Figure 4. 24:Path planning with MOVEIT

However, firstly, we count the places where the robot should go and prioritize those places one by one depending on the readiness of the plant, the quantity of the plant that needs to be harvested, and the need for the plant that will be sprayed.

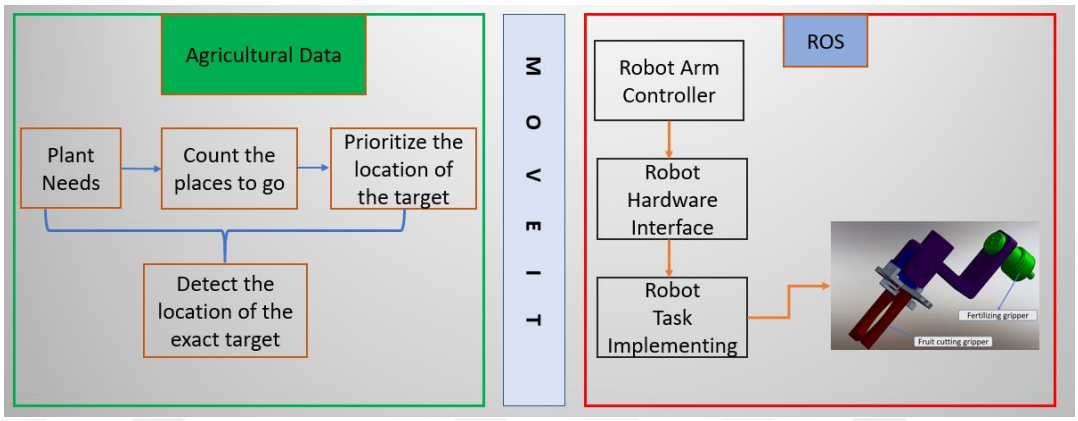


Figure 4. 25 The internal control algorithm of the robot arm

After considering all the previous parameters, we can decide the exact location of every singular plant, and the MOVEIT group will give the command to the robot controller that will help the robot arm hardware and physical interface to move to the target to implement its task, see Figure 4.25.

4.4.3. ROS and Arduino control and communicating structure

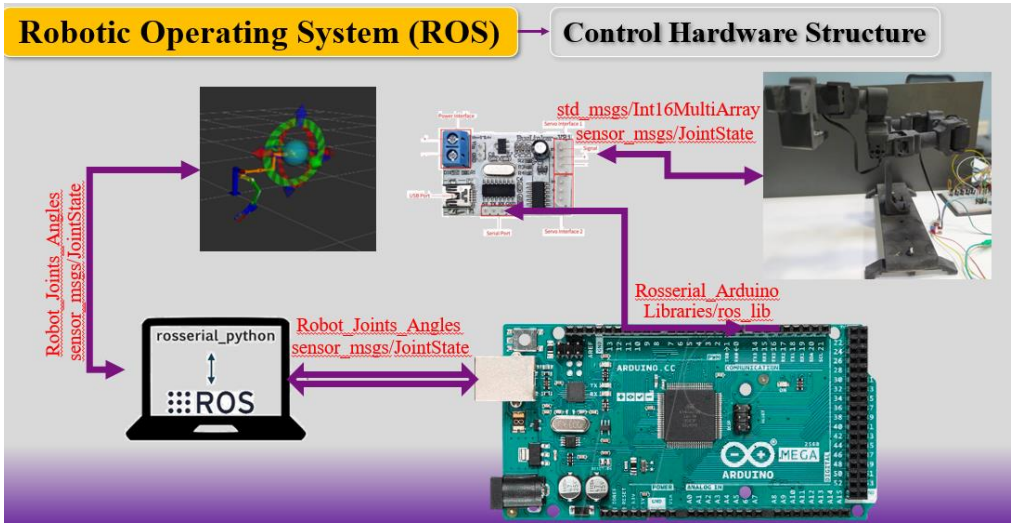


Figure 4. 26 ROS and Arduino Control Structure

As it shows in Figure 4.26 The control strategy is running by the help of Arduino Serial so that there is ROS with its requirements, and it is publishing sensor messages instantaneously that contains the joint states of the arm. Therefore, Arduino can receive that data, process them, and manipulate them, to make them suitable for servo motors. On the other hand, Arduino can publish the robot joints angles too. It can send them via rosserial, and we can display them in ROS terminal to understand the behavior of robot joints and follow the system functioning. All strategy has been achieved using ROS publisher and subscriber topics that we mentioned about them at the beginning of this chapter. Arduino script and required codes are in **Appendix C, part 2** under the name of Communication and control of Agrobot robotic Arm using ROS and Arduino Microcontroller.

Chapter 5

5 Conclusion

In this study, we created an articulated robot arm with new configurations that will be better suited for use in agriculture and the execution of its tasks. We perform the mathematical calculations required for the modeling and control of the robot arm. Additionally, we created, produced, and modified the robot's components so that they could be used with ROS APPs. Additionally, we developed a system control method that will aid researchers and developers who wish to carry out further research in this area. By integrating an arm into a mobile robot, the robot will be able to gather information from the agricultural field and determine where it should move. We could not build the complete agricultural system because of the enormous expense needed, so we built just the robot arm using 3D printing and simulated it using ROS interfaces.

We used mathematical modeling that primarily relies on ***Denavit-Hardenberg*** principles, which enabled us to create a new labeling table from which we were able to derive the homogenous transformation matrices for the model we created. The serial robotics arms supported by this new model configuration will have fresh images that will aid researchers and developers in finding solutions to global problems. Additionally, as we can see in the section on dynamics calculations, the manipulator joints torques were calculated using the ***Lagrangian method*** with the aid of the ***Christoffel Symbol of the First Kind***. Our goal was to use this method to systematically simplify the large number of dynamical equations so that any robotic developer could use it to perform some of the dynamical calculations necessary for a serial robot manipulator.

Dynamics calculations aid in the proper selection of the joints' actuators. As can be seen from joint torque graphs, our mechanism's joints' torque varies from joint to joint, necessitating the employment of various motors with various torques. However, we employed the ***LX-16A*** servo motor in our investigation, which has a 15 kg.cm maximum torque. This caused the joint 1 and joint 3 of the manipulators to have various deficiencies and vibration during experimental work, which can help us choose new actuators.

However, because to the restricted funds available for the project, it was challenging to choose new, more expensive actuators when ROS would have been sufficient to complete and run the simulation.

Over the past ten years, there has been a substantial increase in research efforts aimed at creating agricultural robots that can efficiently complete laborious field jobs. A commercial level of robotics has not been attained for agricultural applications and researchers, developers, and robotics companies are intensively trying to commercialize robotics systems to become more helpful to users.

Making an agriculture machine that can be integrated with ROS to be ready to be included in the agricultural system is difficult today, so the tasks where we must succeed as developers include developing its description model, ROS sensor drivers, and ROS controller interfaces with the required controllers.

That’s why we design a new model with a new controlling algorithm that can enable developers to create and put into practice useful agriculture automation, which is another step toward increasing agriculture productivity and efficiency and enhancing food security for lighting future generations' dreams.

References

[1] Mitra, Manu. Robotic farmers in agriculture. *Advances in Robotics & Mechanical Engineering*.2019; pp. 91-93,doi: 10.32474/ARME.2019.01.000125.

[2] Concepcion II, Ronnie S.,and et al. 'Denavit-Hartenberg-based Analytic Kinematics and Modeling of 6R Degrees of Freedom Robotic Arm for Smart Farming. *Journal of Computational Innovations and Engineering Applications* . 5.2 .2021 ; pp.1-7.

[3] Barth, R., Baur, J., Buschmann, T., Edan, Y., Hellström, T., Nguyen, T.,and Vitzrabin, R. Using ROS for agricultural robotics-design considerations and experiences. In *Proceedings of the Second International Conference on Robotics and associated High-technologies and equipment for agriculture and forestry* , 2014, pp. 509-518.

[4] R Shamshiri, Redmond, Cornelia Weltzien, Ibrahim A. Hameed, Ian J Yule, Tony E Grift, Siva K. Balasundram, Lenka Pitonakova, Desa Ahmad, and Girish Chowdhary. Research and development in agricultural robotics: *A perspective of digital farming*.2018. doi: 10.25165/j.ijabe.20181104.4278.

[5] Ulloa, C. C., Krus, A., Barrientos, A., Del Cerro, J., and Valero, C. Trend Technologies for Robotic Fertilization Process in Row Crops. *Frontiers in Robotics and AI*. 2022; pp. 9. https://doi.org/10.3389/frobt.2022.808484.

[6] Cruz Ulloa, C., Krus, A., Barrientos, A., Del Cerro, J., and Valero, C. Robotic fertilisation using localisation systems based on point clouds in strip-cropping fields. *Agronomy*. 11.1.2020 ; pp. 11. doi:10.3390/agronomy11010011.

[7] Poon, R. J. M. *Design and Control of a Mounted Robotic Arm Tool Changer and Measurement Tools for Agriculture* (Doctoral thesis), Massachusetts Institute of Technology). 2021; pp. 122. https://hdl.handle.net/1721.1/139075.

[8] Deng, H., Xiong, J., and Xia, Z. Mobile manipulation task simulation using ROS with MoveIt.*IEEE International Conference on Real-time Computing and Robotics (RCAR)* IEEE.2017, ;pp. 612-616. doi: 10.1109/rcar.2017.8311930.

[9] A. Mahtani, L. Sanchez, E. Fernandez, A. Martinez, and L. Joseph, *ROS Programming: Building Powerful Robots Design, build, and simulate complex robots using the Robot Operating System* ; 2018.

[10] Koubâa, A. ed. *Robot Operating System (ROS)*.Vol. 2. Cham: Springer. 2017; pp. 112-156.

[11] Koubâa, Anis, ed. *Robot Operating System (ROS).* Vol. 1. *Cham: Springer. 2016.*

[12] Lentin, Joseph. ROS Robotics Projects*: Build a Variety of Awesome Robots that Can See, Sense, Move, and Do a Lot More Using the Powerful Robot Operating System.* Packet publishing *; 2017.*

[13] L. Joseph, J. Cacace, and O'Reilly for Higher Education (Ed). *Mastering ROS for robotics programming.* Packt Publishing,3d ed *;*2021.

[14] Coleman, D., Sucan, I., Chitta, S., and Correll, N. *Reducing the barrier to entry of complex robotic software: a moveit! case study* . 2014 ; https://doi.org/10.48550/arXiv.1404.3785.

[15] DBpedia community . Denavit–Hartenberg_parameters [internet] ; 2022 [date off access 22.10.2022]. https://dbpedia.org/page/Denavit–Hartenberg_parameters.

[16] Tsai, Lung-Wen. *Robot analysis : the mechanics of serial and parallel manipulators* , John Wiley and Sons *;* 1999.

[17] Reddy, A. C. Difference between Denavit-Hartenberg (DH) classical and modified conventions for forward kinematics of robots with case study. In *International Conference on Advanced Materials and manufacturing Technologies (AMMT)* Chandigarh, India: JNTUH College of Engineering Hyderabad. 2014 ; pp. 267-286. doi:10.13140/2.1.2012.9607.

[18] Faria, Carlos, et al. *Automatic Denavit-Hartenberg parameter identification for serial manipulators*. IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society. Vol. 1. 2019 ; pp. 610-617. doi: 10.1109/iecon.2019.8927455.

[19] Dong, Hui, Zhijiang Du, and Gregory S. Chirikjian. *Workspace density and inverse kinematics for planar serial revolute manipulators*. Mechanism and Machine Theory 70

,2013 ; pp. 508-522. https://doi.org/10.1016/j.mechmachtheory.2013.08.008.

[20] Huczala, Daniel, Tomáš Kot, and Martin Pfurner. *An Automated Conversion Between Selected Robot Kinematic Representations*. 2022 ; https://doi.org/10.48550/arXiv.2204.02629.

[21] Jha, Aparna, Manoj Soni, and Mohd Suhaib. *Simulation and kinematic analysis of KUKA KR5 Arc robot*. Vol. 1149. No. 1 . IOP Conference Series: Materials Science and Engineering.. IOP Publishing ; 2021.

[22] Ding, Feng, and Cong Liu. Applying coordinate fixed Denavit–Hartenberg method to solve the workspace of drilling robot arm. *International journal of advanced robotic systems* 15, no. 4 ; 2018. https://doi.org/10.1177/1729881418793283.

[23] Asif, Seemal, and Philip Webb. *Kinematics analysis of 6-DoF articulated robot with spherical wrist*. Mathematical Problems in Engineering 2021 ; 2021. https://doi.org/10.1155/2021/6647035

[24] Spong, M. W., Hutchinson, S., and Vidyasagar, M. *Robot modeling and control* .Vol. 3. New York: Wiley . 2006 ; pp. 75-118.

[25] Bourbonnais, Francis, Pascal Bigras, and Ilian A. Bonev. *Minimum-time trajectory planning and control of a pick-and-place five-bar parallel robot*. IEEE/ASME Transactions on Mechatronics. 20.2.2014 ; pp. 740-749. doi: 10.1109/tmech.2014.2318999.

[26] Uzuner, Sabri, Nihat AKKUŞ, and T. O. Z. Metin. *5-DOF serial robot manipulator design, application and inverse kinematic solution through analytical method and simple search technique*. Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi .26.2.2020; pp. 392-401.

[27] Aimn M,Abdellatif Baba. Robot arm control with arduino , report submitted to department of mechanical and aeronautical engineering (graduation project).TÜRK HAVA KURUMU ÜNİVERSİTESİ REKTÖRLÜĞÜ. doi:10.13140/RG.2.2.10227.53286.

[28] Hiwonder Company. LX-16A-full-metal-gear-serial-bus-servo [internet] ;2022 [date off

access 10.10.2022].https://www.hiwonder.hk/products/hiwonder-lx-16a-full-metal-gear-serial-bus-servo.

[29] Zubrycki, Igor, and Grzegorz Granosik. Introducing modern robotics with ros and arduino, including case studies. Journal of Automation Mobile Robotics and Intelligent Systems 8.1 . 2014 ; pp. 69-75. doi:10.14313/jamris_1-2014/9.

[30] Park, Jaeho, Raimarius Delgado, and Byoung Wook Choi. *Real-time characteristics of ROS 2.0 in multiagent robot systems: an empirical study*. IEEE Access 8 .2020 ; pp. 154637-154651. doi: 10.1109/access.2020.3018122.

[31] PRASANNA, VEMULA, and G. ASHOK. *Design, Modeling And Control of A Biped Robot Platform Based On Poppy Project.* Vol. 10; 2021.

[32] Živković, Aleksandar. *Development of autonomous driving using Robot Operating System* (master's thessis) . Polytechnic University of Madrid ; 2018 . EMSE-2018-3.

[33] Gelan, ABRAHAM AYELE. AUTONOMOUS SEARCH AND RESCUE ROBOT USING ROS PLATFORM. (master's thessis) . Near East University ;2019.

[34] Instructable . ROS MoveIt Robotic Arm [internet]; 2022 [date off access 10.09.2022]. https://www.instructables.com/ROS-MoveIt-Robotic-Arm/.

[35] Gandhinathan, Ramkumar, and Lentin Joseph (Ed). *ROS Robotics projects: build and control robots powered by the Robot Operating System, machine learning, and virtual reality*. Packt Publishing Ltd *;* 2019.

[36] Techtarget website . Ubuntu system [internet]; 2022 [date off access 15.09.2022] . https://www.techtarget.com/searchdatacenter/definition/Ubuntu.

[37] Ubuntu website. Ubuntu operating system installation steps [internet]; 2022 [date off access 15.09.2022] . https://ubuntu.com/server/docs/installation.

[38] wiki website .ROS Melodic installation steps [internet] ; 2022 [date off access 10.09.2022]. http://wiki.ros.org/melodic.

[39] Joseph, Lentin (Ed). *Robot operating system (ros) for absolute beginners* , 2nd ed. *Springer, 2018.*

[40] ROS documentations. ROS Packages building [internet] ; 2022 [date off access 11.09.2022]. http://docs.ros.org/en/independent/api/rospkg/html/packages.html.

[41] Yoon S-P,Han C-C, Ryu W-J ,and Tae H-i (Ed). *ROS Robot Programming, from the basic concept to practical programming and robot application*, 1st ed. *ROBOTIS Co, 2017.*

[42] Lytridis, C., Kaburlasos, V. G., Pachidis, T., Manios, M., Vrochidou, E., Kalampokas, T., and Chatzistamatis, S. *An Overview of Cooperative Robotics in Agriculture*. Agronomy, *11*(9). 2021; pp. 1818. https://doi.org/10.3390/agronomy11091818

Appendices

Appendices

Appendix A Kinematic, and Workspace Calculations

1 Forward Kinematics

```
import numpy as np

import pandas as pd

import sympy as sym

from IPython.display import display,Math

import math

from sympy import *

print("-----symbolic-------")

a1,a2,a3,a4_e2,d4,d4_e1,th1,th2,th3,th4 = symbols("a1 a2 a3 a4_e2 d4 d4_e1 th1 th2 th3 th4") #symbolization of parameters

###############

        ####first link

##X-Axis

TX_1=np.array([[1 ,0 ,0, a1],[0 ,1, 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]])  #DISPLACEMENT IN X FOR LINK_1

##Z-Axis

RZ_1=np.array([[cos(th1) ,-sin(th1) ,0, 0],[sin(th1) ,cos(th1), 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]])  #ROTATION AROUND Z FOR LINK_1

T10=RZ_1@TX_1                    #GENERAL TRANSFORMATION LINK_1
```

```
print("Simplified General HTM for T10 =",simplify(T10))

        ####SECOND link

##X-Axis

TX_2=np.array([[1 ,0 ,0, a2],[0 ,1, 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]])  #DISPLACEMENT IN X FOR LINK_1

RX_2=np.array([[1 ,0 ,0, 0],[0 ,0, 1, 0],[0 ,-1 ,0 ,0],[0 ,0 ,0 ,1]]) #ROTATION AROUND X FOR LINK_1

##Z-Axis

RZ_2=np.array([[cos(th2) ,-sin(th2) ,0, 0],[sin(th2) ,cos(th2), 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]])  #ROTATION AROUND Z FOR LINK_2

T21=RZ_2@TX_2@RX_2                #GENERAL TRANSFORMATION LINK_2

print("Simplified General HTM for T20 =",simplify(T10@T21 ))

         ####THIRD link

##X-Axis

TX_3=np.array([[1 ,0 ,0, a3],[0 ,1, 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]])  #DISPLACEMENT IN X FOR LINK_3

RX_3=np.array([[1 ,0 ,0, 0],[0 ,0, 1, 0],[0 ,-1 ,0 ,0],[0 ,0 ,0 ,1]]) #ROTATION AROUND X FOR LINK_3

##Z-Axis

RZ_3=np.array([[cos(th3) ,-sin(th3) ,0, 0],[sin(th3) ,cos(th3), 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]])  #ROTATION AROUND Z FOR LINK_3

T32=RZ_3@TX_3@RX_3              #GENERAL TRANSFORMATION LINK_3

print("Simplified General HTM for T30 =",simplify(T10@T21@T32 ))

       ####FOURTH link
```

```
##X-Axis

RX_4=np.array([[1 ,0 ,0, 0],[0 ,0, 1, 0],[0 ,-1 ,0 ,0],[0 ,0 ,0 ,1]]) #ROTATION AROUND X FOR LINK_4

##Z-Axis

TZ_4=np.array([[1 ,0 ,0, 0],[0 ,1, 0,0],[0 ,0 ,1 ,d4],[0 ,0 ,0 ,1]]) #DISPLACEMENT IN Z FOR LINK_4

#RZ_4=np.array([[0 ,0 ,1, 0],[-1 ,0, 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]]) #ROTATION AROUND Z FOR LINK_4

T43=TZ_4@RX_4                    #GENERAL TRANSFORMATION LINK_4

print("Simplified General HTM for T40 =",simplify(T10@T21@T32@T43 ))

        ####end effector_1 link

##X-Axis

     #NO

##Z-Axis

TZ_E1=np.array([[1 ,0 ,0, 0],[0 ,1, 0, 0],[0 ,0 ,1 ,d4_e1],[0 ,0 ,0 ,1]]) #DISPLACEMENT IN Z FOR LINK_E1

RZ_E1=np.array([[cos(th4) ,-sin(th4) ,0, 0],[sin(th4) ,cos(th4), 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]]) #ROTATION AROUND Z FOR LINK_E1

TE14=TZ_E1@RZ_E1                    #GENERAL TRANSFORMATION LINK_E1

        ####end effector_2 link

##X-Axis

TX_E2=np.array([[1 ,0 ,0, a4_e2],[0 ,1, 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]]) #DISPLACEMENT IN X FOR LINK_E2

##Z-Axis
```

```
RZ_E2=np.array([[cos(th4) ,-sin(th4) ,0, 0],[sin(th4) ,cos(th4), 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]])  #ROTATION AROUND Z FOR LINK_E2

TE24=TX_E2@RZ_E2                #GENERAL TRANSFORMATION LINK_E2

General_HTM_E1=T10@T21@T32@T43@TE14

General_HTM_SIMP1=simplify(General_HTM_E1)

print("Simplified General HTM_E1 =",General_HTM_SIMP1 )

###########

General_HTM_E2=T10@T21@T32@T43@TE24

General_HTM_SIMP2=simplify(General_HTM_E2)

print("Simplified General HTM_E2 =",General_HTM_SIMP2 )

####################################3
```

2 Inverse kinematics

```
import numpy as np

import pandas as pd

import sympy as sym

from IPython.display import display,Math

import math

from sympy import *

print("-----symbolic-------")

a1,a2,a3,a4_e2,d4,d4_e1,th1,th2,th3,th4,X,Y= symbols("a1 a2 a3 a4_e2 d4 d4_e1 th1 th2 th3 th4 X Y") #symbolization of parameters

##############
```

```
    ####first link

##X-Axis

TX_1=np.array([[1 ,0 ,0, a1],[0 ,1, 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]])  #DISPLACEMENT IN
X FOR LINK_1

##Z-Axis

RZ_1=np.array([[cos(th1) ,-sin(th1) ,0, 0],[sin(th1) ,cos(th1), 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0
,1]])  #ROTATION AROUND Z FOR LINK_1

T10=RZ_1@TX_1                  #GENERAL TRANSFORMATION LINK_1

print("Simplified General HTM for T10 =",simplify(T10))

    ####SECOND link

##X-Axis

TX_2=np.array([[1 ,0 ,0, a2],[0 ,1, 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]])  #DISPLACEMENT IN
X FOR LINK_1

RX_2=np.array([[1 ,0 ,0, 0],[0 ,0, 1, 0],[0 ,-1 ,0 ,0],[0 ,0 ,0 ,1]]) #ROTATION AROUND
X FOR LINK_1

##Z-Axis

RZ_2=np.array([[cos(th2) ,-sin(th2) ,0, 0],[sin(th2) ,cos(th2), 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0
,1]])  #ROTATION AROUND Z FOR LINK_2

T21=RZ_2@TX_2@RX_2                #GENERAL TRANSFORMATION LINK_2

print("Simplified General HTM for T20 =",simplify(T10@T21 ))

    ####THIRD link

##X-Axis

TX_3=np.array([[1 ,0 ,0, a3],[0 ,1, 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]])  #DISPLACEMENT IN
X FOR LINK_3
```

```
RX_3=np.array([[1 ,0 ,0, 0],[0 ,0, 1, 0],[0 ,-1 ,0 ,0],[0 ,0 ,0 ,1]]) #ROTATION AROUND X FOR LINK_3

##Z-Axis

RZ_3=np.array([[cos(th3) ,-sin(th3) ,0, 0],[sin(th3) ,cos(th3), 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]]) #ROTATION AROUND Z FOR LINK_3

T32=RZ_3@TX_3@RX_3              #GENERAL TRANSFORMATION LINK_3

print("Simplified General HTM for T30 =",simplify(T10@T21@T32 ))

      ####FOURTH link

##X-Axis

RX_4=np.array([[1 ,0 ,0, 0],[0 ,0, 1, 0],[0 ,-1 ,0 ,0],[0 ,0 ,0 ,1]]) #ROTATION AROUND X FOR LINK_4

##Z-Axis

TZ_4=np.array([[1 ,0 ,0, 0],[0 ,1, 0,0],[0 ,0 ,1 ,d4],[0 ,0 ,0 ,1]]) #DISPLACEMENT IN Z FOR LINK_4

#RZ_4=np.array([[0 ,0 ,1, 0],[-1 ,0, 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]]) #ROTATION AROUND Z FOR LINK_4

T43=TZ_4@RX_4                   #GENERAL TRANSFORMATION LINK_4

print("Simplified General HTM for T40 =",simplify(T10@T21@T32@T43 ))

       ####end effector_1 link

##X-Axis

     #NO

##Z-Axis

TZ_E1=np.array([[1 ,0 ,0, 0],[0 ,1, 0, 0],[0 ,0 ,1 ,d4_e1],[0 ,0 ,0 ,1]]) #DISPLACEMENT IN Z FOR LINK_E1
```

```
RZ_E1=np.array([[cos(th4) ,-sin(th4) ,0, 0],[sin(th4) ,cos(th4), 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]])  #ROTATION AROUND Z FOR LINK_E1

TE14=TZ_E1@RZ_E1                    #GENERAL TRANSFORMATION LINK_E1

        ####end effector_2 link

##X-Axis

TX_E2=np.array([[1 ,0 ,0, a4_e2],[0 ,1, 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]]) #DISPLACEMENT IN X FOR LINK_E2

##Z-Axis

RZ_E2=np.array([[cos(th4) ,-sin(th4) ,0, 0],[sin(th4) ,cos(th4), 0, 0],[0 ,0 ,1 ,0],[0 ,0 ,0 ,1]])  #ROTATION AROUND Z FOR LINK_E2

TE24=TX_E2@RZ_E2                    #GENERAL TRANSFORMATION LINK_E2

General_HTM_E1=T10@T21@T32@T43@TE14

General_HTM_SIMP1=simplify(General_HTM_E1)

print("Simplified General HTM_E1 =",General_HTM_SIMP1 )

############

General_HTM_E2=T10@T21@T32@T43@TE24

General_HTM_SIMP2=simplify(General_HTM_E2)

print("Simplified General HTM_E2 =",General_HTM_SIMP2 )

####################################3

##INVERSE KINEMATICS

R0_4e1=np.array([[1 ,0 ,0, 0],[0 ,0, 1, 0],[0 ,-1 ,0 ,0],[0 ,0 ,0 ,1]])

R0_3=RZ_1@RX_2@RX_3@RZ_3
```

```
iNvR0_3=np.linalg.inv(R0_3)

print("Simplified General R0_2 =",R0_3 )

print("Simplified General invR0_2 =",iNvR0_3 )

R3_6=iNvR0_3@R0_4

print("Simplified General r3_6 =",R3_6 )
```

3 Jacobian Analysis

```
import numpy as np

import pandas as pd

import sympy as sym

from IPython.display import display,Math

from math import *

from sympy import *

init_printing()

print("-----symbolic for abriviate using R4E2 TO ZERO-------")

a1,a2,a3,a4_e2,d4,d4_e1,th1,th2,th3,th4,xp,yp = symbols("a1 a2 a3 a4_e2 d4 d4_e1 th1 th2 th3 th4 xp yp") #symbolization of parameters

###############theta2

R0_0=Matrix([[1 ,0 , 0],[0 ,1, 0],[0 ,0 ,1]])

##Z-Axis

R0_1=Matrix([[cos(th1) ,-sin(th1) , 0],[sin(th1) ,cos(th1), 0],[0 ,0 ,1]])

print("Simplified General HTM for R10 =",simplify(R0_1))
```

```
        ####SECOND link
##X-Axis
RX_2=Matrix([[1 ,0 ,0],[0 ,0, 1],[0 ,-1 ,0 ]])
##Z-Axis
RZ_2=Matrix([[cos(th2) ,-sin(th2) , 0],[sin(th2) ,cos(th2), 0],[0 ,0 ,1]])
R21=RZ_2@RX_2                #GENERAL TRANSFORMATION LINK_2
R20=R0_1@R21                 #GENERAL TRANSFORMATION LINK_2
print("Simplified General ROTATION  for R20 =",simplify(R20))
#print("Simplified General HTM for T20 =",simplify(T10@T21 ))
         ####THIRD link
RX_3=Matrix([[1 ,0 ,0],[0 ,0, 1],[0 ,-1 ,0 ]])
##Z-Axis
RZ_3=Matrix([[cos(th3) ,-sin(th3) , 0],[sin(th3) ,cos(th3), 0],[0 ,0 ,1]])
R32=RZ_3@RX_3                #GENERAL TRANSFORMATION LINK_2
R30=R20@R32
print("Simplified General ROTATION  for R30 =",simplify(R30))
#####
R43=np.array([[1 ,0 ,0],[0 ,0, 1],[0 ,-1 ,0 ]])  #ROTATION AROUND X FOR LINK_4
R40=R30@R43
print("Simplified General ROTATION  for R40 =",simplify(R40))
########
```

```
Re24=np.array([[cos(th4) ,-sin(th4) , 0],[sin(th4) ,cos(th4), 0],[0 ,0 ,1]])  #ROTATION AROUND Z FOR LINK_E2

R4E_20=R40@Re24

print("Simplified General ROTATION  for R4E_20 =",simplify(R4E_20))

print("*********************************")

R4e2_0=Matrix([[1 ,0 ,0],[0 ,1, 0],[0 ,0 ,-1 ]])

#GENERAL ROTATION FOR R3_0

R30=R0_1@R21

print("Simplified General ROTATION  for R30 =",simplify(R30))

invR30=R30.inv()

R4e2_3=invR30@R4e2_0

print("General ROTATION  for R4e2_3 =",simplify(R4e2_3))

######################

print("direction of joints axix ----------------------")

Z0=Matrix([[0],[0],[1 ]])

print("DIRECTION OF JINT Z0 =",simplify(Z0))

Z1=R0_1@Z0

print("DIRECTION OF JINT Z1 =",simplify(Z1))

Z2=R20@Z0

print("DIRECTION OF JINT Z2 =",simplify(Z2))

Z3=R30@Z0

print("DIRECTION OF JINT Z3 =",simplify(Z3))
```

```
Z4=R40@Z0

print("DIRECTION OF JINT Z4 =",simplify(Z4))

Z5=R4E_20@Z0

print("DIRECTION OF JINT Z5 =",simplify(Z5))

print("position vector of end effector ----------------------")

r4_3=Matrix([[0],[0],[d4]])

r3_2=Matrix([[a3*cos(th3)],[a3*sin(th3)],[0]])

r2_1=Matrix([[a2*cos(th2)],[a2*sin(th2)],[0]])

r1_0=Matrix([[a1*cos(th1)],[a1*sin(th1)],[0]])

P43=R30@r4_3

print("POSITION VECTOR OF P43* =",simplify(P43))

P42=R20@r3_2+P43

print("POSITION VECTOR OF P42* =",simplify(P42))

P41=R0_1@r2_1+P42

print("POSITION VECTOR OF P41* =",simplify(P41))

P40=R0_0@r1_0+P41

print("POSITION VECTOR OF P40* =",simplify(P40))

print("JACOBIAN ANALYSIS ----------------------")

Z0_num=np.array(Z0)

Z0T=Z0_num.transpose()

#print(Z0)
```

```
print("Z0T ",simplify(Z0T))

P40_num=np.array(P40)

P40T=P40_num.transpose()

J1R1=np.cross(Z0T,P40T)          #JACOBIAN 1 ROW 1

print("JACOBIAN 1 ROW 1",simplify(J1R1))

################

Z1_num=np.array(Z1)

Z1T=Z1_num.transpose()

#print(Z1)

print("Z1T ",simplify(Z1T))

P41_num=np.array(P41)

P41T=P41_num.transpose()

J2R1=np.cross(Z1T,P41T)          #JACOBIAN 2 ROW 1

print("JACOBIAN 2 ROW 1",simplify(J2R1))

################

Z2_num=np.array(Z2)

Z2T=Z2_num.transpose()

#print(Z2)

print("Z2T ",simplify(Z2T))

P42_num=np.array(P42)

P42T=P42_num.transpose()
```

```
#print("Z2T ",simplify(Z2T))

#print("P42T ",simplify(P42T))

J3R1=np.cross(Z2T,P42T)          #JACOBIAN 3 ROW 1

print("JACOBIAN 3 ROW 1",simplify(J3R1))

################

Z3_num=np.array(Z3)

Z3T=Z3_num.transpose()

#print(Z3)

print("Z3T ",simplify(Z3T))

P43_num=np.array(P43)

P43T=P43_num.transpose()

#print("Z3T ",simplify(Z3T))

#print("P43T ",simplify(P43T))

J4R1=np.cross(Z3T,P43T)          #JACOBIAN 4 ROW 1

print("JACOBIAN 4 ROW 1",simplify(J4R1))

print("GENERAL JACOBIAN MATRICES ----------------------")

print(Matrix(J1R1.transpose()))

JACOBIAN_1=print(Matrix(Z0T.transpose()))

//////////////////////
```

4 Workspace Analysis

```
a1 = 12;
```

```
a2 = 12;

a3 = 8;

a4 = 8.5;

d4 = 8;

d4e1 = 5;

a4e2 = 5;

tpp = 8;

tstep = 0.001;(*TIME STEP*)

tpath = t - 6;(*END EFFECTOR FUNCTION*)

tdpath = 1;

tddpath = 0;

q1value = 360;

q2value = 360;

q3value = 360;

q4value = 360;

m1 = 0.07;

m2 = 0.07;

m3 = 0.05;

m4 = 0.05;

q1 = Table[(47 + q1value)*tpath*\[Pi]/180, {t, 1, tpp, tstep}];

q2 = Table[(191 + q2value)*tpath*\[Pi]/180, {t, 1, tpp, tstep}];
```

```
q3 = Table[(164 + q3value)*tpath*\[Pi]/180, {t, 1, tpp, tstep}];

q4 = Table[(180 + q4value)*tpath*\[Pi]/180, {t, 1, tpp, tstep}];

                (*GRIPPER ONE FORWARD KINEMATIC*)

Forward1 = ({

   {Cos[q1 + q2]*Cos[q3 - q4], Cos[q1 + q2]*Sin[q3 - q4],

    Sin[q1 + q2],

    a1*Cos[q1] + a2*Cos[q1 + q2] + a3*Cos[q3]*Cos[q1 + q2] -

     d4*Sin[q3]*Cos[q1 + q2] + d4e1*Sin[q1 + q2]},

   {Sin[q1 + q2]*Cos[q3 - q4],

    Sin[q1 + q2]*Sin[q3 - q4], -Cos[q1 + q2],

    a1*Sin[q1] + a2*Sin[q1 + q2] + a3*Cos[q3]*Sin[q1 + q2] -

     d4*Sin[q3]*Sin[q1 + q2] - d4e1*Cos[q1 + q2]},

   {-Sin[q3 - q4], Cos[q3 - q4], 0, a3*Sin[q3] - d4*Cos[q3]},

   {0, 0, 0, 1}

  });

                (*GRIPPER TWO FORWARD KINEMATIC*)

Forward2 = ({

   {Cos[q1 + q2]*Cos[q3 - q4], Cos[q1 + q2]*Sin[q3 - q4],

    Sin[q1 + q2],

    a1*Cos[q1] + a2*Cos[q1 + q2] + a3*Cos[q3]*Cos[q1 + q2] +

     a4e2*Cos[q3]*Cos[q1 + q2] - d4*Sin[q3]*Cos[q1 + q2]},
```

```
    {Sin[q1 + q2]*Cos[q3 - q4],
     Sin[q1 + q2]*Sin[q3 - q4], -Cos[q1 + q2],
     a1*Sin[q1] + a2*Sin[q1 + q2] + a3*Cos[q3]*Sin[q1 + q2] +
      a4e2*Cos[q3]*Sin[q1 + q2] - d4*Sin[q3]*Sin[q1 + q2]},
    {-Sin[q3 - q4], Cos[q3 - q4],
     1, -a3*Sin[q3] - a4e2*Sin[q3] - d4*Cos[q3]},
    {0, 0, 0, 1}
   });


(*GRIPPER ONE POSITIONS*)
XPOSITIONEE1 = Forward1[[1, 4]];
YPOSITIONEE1 = Forward1[[2, 4]];
ZPOSITIONEE1 = Forward1[[3, 4]];
Points1 =
  Table[{XPOSITIONEE1[[i]], YPOSITIONEE1[[i]], ZPOSITIONEE1[[i]]}, {i,
    0, Dimensions[XPOSITIONEE1][[1]]}];
(*ListPointPlot3D[Points1,PlotRange\[Rule]All,ColorFunction\[Rule]\
Function[{x,y,z},Hue[z]],PlotLabel\[Rule]"Gripper1",Filling\[Rule]Top]\
*)
ListPointPlot3D[Points1, PlotRange -> All,
 PlotStyle -> {Darker[Brown], Specularity[Red, 5], PointSize[0.1]},
```

```
 Axes -> None, PlotLabel -> "Gripper1", Filling -> Top]

(*GRIPPER TOW POSITIONS*)

XPOSITIONEE2 = Forward2[[1, 4]];

YPOSITIONEE2 = Forward2[[2, 4]];

ZPOSITIONEE2 = Forward2[[3, 4]];

Points2 =

  Table[{XPOSITIONEE2[[j]], YPOSITIONEE2[[j]], ZPOSITIONEE2[[j]]}, {j,

    0, Dimensions[XPOSITIONEE2][[1]]}];

(*ListPointPlot3D[Points2,PlotRange\[Rule]All,PlotStyle\[Rule]\

PointSize[Tiny],Filling\[Rule]Top]*)

ListPointPlot3D[Points2, PlotRange -> All,

 PlotStyle -> {Darker[Purple], Specularity[Red, 5], PointSize[0.1]},

 Axes -> None, PlotLabel -> "Gripper2", Filling -> Top]
```

Appendix B Dynamic Calculations

Link Inertia matrix calculations codes

1 - Links Inertia Matrices Computation

```
(*Inertia Matrices computation*)
```

$$I11 = \frac{1}{12}\, m1\, a1\, a1 \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} // \text{MatrixForm}$$

$$I22 = \frac{1}{12}\, m2\, a2\, a2 \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} // \text{MatrixForm}$$

$$I33 = \frac{1}{12}\, m3\, a3\, a3 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} // \text{MatrixForm}$$

$$I44 = \frac{1}{12}\, m4\, a4\, a4 \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} // \text{MatrixForm}$$

$$I1 = \begin{pmatrix} \frac{1}{12}\, a1^2\, m1 \sin^2(q1) & -\frac{1}{12}\, a1^2\, m1 \cos(q1) \sin(q1) & 0 \\ -\frac{1}{12}\, a1^2\, m1 \cos(q1) \sin(q1) & \frac{1}{12}\, a1^2\, m1 \cos^2(q1) & 0 \\ 0 & 0 & \frac{a1^2\, m1}{12} \end{pmatrix}$$

$$I2 = \begin{pmatrix} \frac{1}{12}\, a2^2\, m2 \sin^2(q1+q2) & -\frac{1}{24}\, a2^2\, m2 \sin(2\,(q1+q2)) & 0 \\ -\frac{1}{24}\, a2^2\, m2 \sin(2\,(q1+q2)) & \frac{1}{12}\, a2^2\, m2 \cos^2(q1+q2) & 0 \\ 0 & 0 & \frac{a2^2\, m2}{12} \end{pmatrix}$$

$$I3 = \begin{pmatrix} \frac{1}{12}\, a3^2\, m3\, (\cos^2(q1+q2)\cos^2(q3) + \sin^2(q1+q2)) & -\frac{1}{24}\, a3^2\, m3 \sin(2\,(q1+q2)) \sin^2(q3) & -\frac{1}{24}\, a3^2\, m3 \cos(q1+q2) \sin(2\,q3) \\ -\frac{1}{24}\, a3^2\, m3 \sin(2\,(q1+q2)) \sin^2(q3) & \frac{1}{12}\, a3^2\, m3\, (\cos^2(q1+q2) + \cos^2(q3) \sin^2(q1+q2)) & -\frac{1}{24}\, a3^2\, m3 \sin(q1+q2) \sin(2\,q3) \\ -\frac{1}{24}\, a3^2\, m3 \cos(q1+q2) \sin(2\,q3) & -\frac{1}{24}\, a3^2\, m3 \sin(q1+q2) \sin(2\,q3) & \frac{1}{12}\, a3^2\, m3 \sin^2(q3) \end{pmatrix}$$

$$I4 = \begin{pmatrix} \frac{1}{12}\, a4^2\, m4\, (\sin^2(q1+q2) + \cos^2(q1+q2) \sin^2(q3-q4)) & -\frac{1}{48}\, a4^2\, m4\, (\cos(2\,(q3-q4)) - 3) \sin(2\,(q1+q2)) & \frac{1}{24}\, a4^2\, m4 \cos(q1+q2) \sin(2\,(q3-q4)) \\ -\frac{1}{48}\, a4^2\, m4\, (\cos(2\,(q3-q4)) - 3) \sin(2\,(q1+q2)) & \frac{1}{12}\, a4^2\, m4\, (\cos^2(q1+q2) + \sin^2(q1+q2) \sin^2(q3-q4)) & \frac{1}{24}\, a4^2\, m4 \sin(q1+q2) \sin(2\,(q3-q4)) \\ \frac{1}{24}\, a4^2\, m4 \cos(q1+q2) \sin(2\,(q3-q4)) & \frac{1}{24}\, a4^2\, m4 \sin(q1+q2) \sin(2\,(q3-q4)) & \frac{1}{12}\, a4^2\, m4 \cos^2(q3-q4) \end{pmatrix}$$

2 - **Links Jacobian Computation**

Links radius Calculations

$$rc10 = \begin{pmatrix} \frac{1}{2} a1 \cos[q1] \\ \frac{1}{2} a1 \sin[q1] \\ 0 \end{pmatrix} // \text{MatrixForm}$$

$$rc21 = \begin{pmatrix} \frac{1}{2} a2 \cos[q2] \\ \frac{1}{2} a2 \sin[q2] \\ 0 \end{pmatrix} // \text{MatrixForm}$$

$$rc32 = \begin{pmatrix} \frac{1}{2} a3 \cos[q3] \\ \frac{1}{2} a3 \sin[q3] \\ d3 \end{pmatrix} // \text{MatrixForm}$$

$$rc43g1 = \begin{pmatrix} \frac{1}{2} a4 \cos[q4] \\ \frac{1}{2} a4 \sin[q4] \\ 0 \end{pmatrix} // \text{MatrixForm}$$

Links Z-Directions

$$z0 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} // \text{MatrixForm}$$

$$z1 = \begin{pmatrix} \cos[q1] & -\sin[q1] & 0 \\ \sin[q1] & \cos[q1] & 0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} // \text{MatrixForm}$$

$$z2 = \begin{pmatrix} \cos[q1+q2] & 0 & -\sin[q1+q2] \\ \sin[q1+q2] & 0 & \cos[q1+q2] \\ 0 & -1 & 0 \end{pmatrix} . \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} // \text{MatrixForm}$$

$$z3 = \begin{pmatrix} \cos[q3]\cos[q1+q2] & \sin[q1+q2] & -\sin[q3]\cos[q1+q2] \\ \cos[q3]\sin[q1+q2] & -\cos[q1+q2] & -\sin[q3]\sin[q1+q2] \\ -\sin[q3] & 0 & -\cos[q3] \end{pmatrix} . \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} // \text{MatrixForm}$$

$$z4 = \begin{pmatrix} \cos[q3-q4]\cos[q1+q2] & \sin[q3-q4]\cos[q1+q2] & \sin[q1+q2] \\ \cos[q3-q4]\sin[q1+q2] & \sin[q3-q4]\sin[q1+q2] & \cos[q1+q2] \\ -\sin[q3-q4] & \cos[q3-q4] & 0 \end{pmatrix}$$

$$. \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} // \text{MatrixForm}$$

General position vector of the links described on base frame

```
PC10 = {{Cos[q1], 0, -Sin[q1]}, {Sin[q1], 0, Cos[q1]}, {0, -1, 0}} . {{a1/2}, {0}, {0}} // MatrixForm

PC20 = {{Cos[q1], 0, -Sin[q1]}, {Sin[q1], 0, Cos[q1]}, {0, -1, 0}} . {{a1}, {0}, {0}} + {{Cos[q1 + q2], 0, -Sin[q1 + q2]}, {Sin[q1 + q2], 0, Cos[q1 + q2]}, {0, -1, 0}} . {{a2/2}, {0}, {0}} // MatrixForm

PC30 = {{Cos[q1], 0, -Sin[q1]}, {Sin[q1], 0, Cos[q1]}, {0, -1, 0}} . {{a1}, {0}, {0}} + {{Cos[q1 + q2], 0, -Sin[q1 + q2]}, {Sin[q1 + q2], 0, Cos[q1 + q2]}, {0, -1, 0}} . {{a2}, {0}, {0}} +
 {{Cos[q3] Cos[q1 + q2], Sin[q1 + q2], -Sin[q3] Cos[q1 + q2]}, {Cos[q3] Sin[q1 + q2], -Cos[q1 + q2], -Sin[q3] Sin[q1 + q2]}, {-Sin[q3], 0, -Cos[q3]}} . {{a3/2}, {0}, {d3/2}} // MatrixForm

PC40 = {{Cos[q1], 0, -Sin[q1]}, {Sin[q1], 0, Cos[q1]}, {0, -1, 0}} . {{a1}, {0}, {0}} + {{Cos[q1 + q2], 0, -Sin[q1 + q2]}, {Sin[q1 + q2], 0, Cos[q1 + q2]}, {0, -1, 0}} . {{a2}, {0}, {0}} +
 {{Cos[q3] Cos[q1 + q2], Sin[q1 + q2], -Sin[q3] Cos[q1 + q2]}, {Cos[q3] Sin[q1 + q2], -Cos[q1 + q2], -Sin[q3] Sin[q1 + q2]}, {-Sin[q3], 0, -Cos[q3]}} . {{a3}, {0}, {d3}} +
 {{Cos[q3 - q4] Cos[q1 + q2], Sin[q3 - q4] Cos[q1 + q2], Sin[q1 + q2]}, {Cos[q3 - q4] Sin[q1 + q2], Sin[q3 - q4] Sin[q1 + q2], Cos[q1 + q2]}, {-Sin[q3 - q4], Cos[q3 - q4], 0}} . {{a4/2}, {0}, {0}} // MatrixForm
```

$$PC10 = \begin{pmatrix} \frac{1}{2}\, a1 \cos(q1) \\ \frac{1}{2}\, a1 \sin(q1) \\ 0 \end{pmatrix} PC20 = \begin{pmatrix} a1 \cos(q1) + \frac{1}{2}\, a2 \cos(q1 + q2) \\ a1 \sin(q1) + \frac{1}{2}\, a2 \sin(q1 + q2) \\ 0 \end{pmatrix}$$

$$PC30 = \begin{pmatrix} a1 \cos(q1) + a2 \cos(q1 + q2) + \frac{1}{2}\, a3 \cos(q1 + q2) \cos(q3) - \frac{1}{2}\, d3 \cos(q1 + q2) \sin(q3) \\ a1 \sin(q1) + a2 \sin(q1 + q2) + \frac{1}{2}\, a3 \cos(q3) \sin(q1 + q2) - \frac{1}{2}\, d3 \sin(q1 + q2) \sin(q3) \\ -\frac{1}{2}\, d3 \cos(q3) - \frac{1}{2}\, a3 \sin(q3) \end{pmatrix}$$

$$PC40 = \begin{pmatrix} a1 \cos(q1) + a2 \cos(q1 + q2) + a3 \cos(q1 + q2) \cos(q3) + \frac{1}{2}\, a4 \cos(q1 + q2) \cos(q3 - q4) - d3 \cos(q1 + q2) \sin(q3) \\ a1 \sin(q1) + a2 \sin(q1 + q2) + a3 \cos(q3) \sin(q1 + q2) + \frac{1}{2}\, a4 \cos(q3 - q4) \sin(q1 + q2) - d3 \sin(q1 + q2) \sin(q3) \\ -d3 \cos(q3) - a3 \sin(q3) - \frac{1}{2}\, a4 \sin(q3 - q4) \end{pmatrix}$$

General formulation for joints submatrices Jacobian

$$JV1 = \begin{pmatrix} -\frac{1}{2}\, a1 \sin(q1) & 0 & 0 & 0 \\ \frac{1}{2}\, a1 \cos(q1) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$JV2 = \begin{pmatrix} -a1 \sin(q1) - \frac{1}{2}\, a2 \sin(q1 + q2) & -\frac{1}{2}\, a2 \sin(q1 + q2) & 0 & 0 \\ a1 \cos(q1) + \frac{1}{2}\, a2 \cos(q1 + q2) & \frac{1}{2}\, a2 \cos(q1 + q2) & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$JV3 = \begin{pmatrix} -\cos(q1+q2)\,(d3\cos(q3)+a3\sin(q3)) & -a1\sin(q1)-a2\sin(q1+q2) & -\frac{1}{2}a2\sin(q1+q2) & 0 \\ -\sin(q1+q2)\,(d3\cos(q3)+a3\sin(q3)) & a1\cos(q1)+a2\cos(q1+q2) & \frac{1}{2}a2\cos(q1+q2) & 0 \\ -a2-a1\cos(q2)-a3\cos(q3)+d3\sin(q3) & 0 & 0 & 0 \end{pmatrix}$$

JV4 = ({

{1/2 (cos(q2) (2 a3 + 2 a2 cos(q3) + a4 cos(q4)) sin(q1) +

cos(q1) (2 a3 + a4 cos(q4)) sin(q2) +

2 cos(q3) (a1 sin(q1) + a2 cos(q1) sin(q2))), -(1/2) cos(

q1 + q2) (d3 cos(q3) + a3 sin(q3)), -a1 sin(q1) -

1/2 a2 sin(q1 + q2), -(1/2) a2 sin(q1 + q2)},

{1/2 ((2 a3 + 2 a2 cos(q3) + a4 cos(q4)) sin(q1) sin(q2) -

cos(q1) (2 a1 cos(q3) +

cos(q2) (2 a3 + 2 a2 cos(q3) + a4 cos(q4)))), -(1/2) sin(

q1 + q2) (d3 cos(q3) + a3 sin(q3)),

a1 cos(q1) + 1/2 a2 cos(q1 + q2), 1/2 a2 cos(q1 + q2)},

{a1 sin(q2) sin(q3),

1/2 (-2 a2 - 2 a1 cos(q2) - a3 cos(q3) + d3 sin(q3)), 0, 0}

})

$$Jw1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \text{// MatrixForm}$$

$$Jw2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \text{// MatrixForm}$$

$$Jw3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix} \text{// MatrixForm}$$

$$Jw4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \text{// MatrixForm}$$

3 - General Mass inertia Matrix for Dynamic equation, first term

MTT=Simplify[M1+M2+M3+M4]//TraditionalForm

({

{1/24 (8 m1 $a1^2$+24 m2 $a1^2$+12 m3 $a1^2$+18 m4 $a1^2$+6 (2 m3-m4) cos(2 q2) $a1^2$+3 m4 cos(2 (q2-q3)) $a1^2$+6 m4 cos(2 q3) $a1^2$+3 m4 cos(2 (q2+q3)) $a1^2$+24 a2 (m2+2 m3+m4) cos(q2) a1+12 a2 m4 cos(q2-2 q3) a1+24 a3 m3 cos(q2-q3) a1+24 a3 m4 cos(q2-q3) a1+24 a3 m3 cos(q2+q3) a1+24 a3 m4 cos(q2+q3) a1+12 a2 m4 cos(q2+2 q3) a1+6 a4 m4 cos(q2-q3-q4) a1+6 a4 m4 cos(q2+q3-q4) a1+6 a4 m4 cos(q2-q3+q4) a1+6 a4 m4 cos(q2+q3+q4) a1+24 d3 m3 sin(q2-q3) a1-24 d3 m3 sin(q2+q3) a1+8 $a2^2$ m2+24 $a2^2$ m3+25 $a3^2$ m3+24 $d3^2$ m3+12 $a2^2$ m4+24 $a3^2$ m4+4 $a4^2$ m4+48 a2 a3 m3 cos(q3)+48 a2 a3 m4 cos(q3)-$a3^2$ m3 cos(2 q3)+12 $a2^2$ m4 cos(2 q3)+12 a2 a4 m4 cos(q3-q4)+$a4^2$ m4 cos(2 (q3-q4))+24 a3 a4 m4 cos(q4)+3 $a4^2$ m4 cos(2 q4)+12 a2 a4 m4 cos(q3+q4)-48 a2 d3 m3 sin(q3)), 1/24 (-6 m4 cos(2 q2-q3) $a1^2$+6 m4 cos(2 q2+q3) $a1^2$+12 a2 m2 cos(q2) a1-12 (a3 m3+a2 m4) cos(q2-q3) a1+12 a3 m3 cos(q2+q3) a1+12 a2 m4 cos(q2+q3) a1+12 d3 m4 sin(q2) a1-12 d3 m3 sin(q2-q3) a1-12 d3 m3 sin(q2+q3) a1+8 $a2^2$ m2+$a3^2$ m3+$a4^2$ m4-$a3^2$ m3 cos(2 q3)+$a4^2$ m4 cos(2 (q3-q4))), 1/24 (-24 m4 cos(q3) $a1^2$-24 a3 m4 cos(q2) a1-18 a2 m4 cos(q2-q3) a1-18 a2 m4 cos(q2+q3) a1-6 a4 m4 cos(q2-q4) a1-6 a4 m4 cos(q2+q4) a1+$a3^2$ m3+$a4^2$ m4-12 a2 a3 m4-12 $a2^2$ m4 cos(q3)-$a3^2$ m3 cos(2 q3)+$a4^2$ m4 cos(2 (q3-q4))-6 a2 a4 m4 cos(q4)), 1/24 m4 (-12 cos(q3) $a2^2$-12 a3 a2-6 a1 cos(q2-q3) a2-6 a1 cos(q2+q3) a2-6 a4 cos(q4) a2+$a4^2$+$a4^2$ cos(2 (q3-q4)))},

{1/24 (-6 m4 cos(2 q2-q3) $a1^2$+6 m4 cos(2 q2+q3) $a1^2$+12 a2 m2 cos(q2) a1-12 (a3 m3+a2 m4) cos(q2-q3) a1+12 a3 m3 cos(q2+q3) a1+12 a2 m4 cos(q2+q3) a1+12 d3 m4 sin(q2) a1-12 d3 m3 sin(q2-q3) a1-12 d3 m3 sin(q2+q3) a1+8 $a2^2$ m2+$a3^2$ m3+$a4^2$ m4-$a3^2$ m3 cos(2 q3)+$a4^2$ m4 cos(2 (q3-q4))), 1/24 (24 m3 $a1^2$+12 m4 $a1^2$+12 m4 cos(2 q2) $a1^2$+48 a2 (m3+m4) cos(q2) a1+12 a3 m4 cos(q2-q3) a1+12 a3 m4 cos(q2+q3) a1+12 d3 m4 sin(q2-q3) a1-12 d3 m4 sin(q2+q3) a1+8 $a2^2$ m2+24 $a2^2$ m3+$a3^2$ m3+24 $a2^2$ m4+6 $a3^2$ m4+$a4^2$ m4+6 $d3^2$ m4+24 a2 a3 m4 cos(q3)-$a3^2$ m3 cos(2 q3)+$a4^2$ m4 cos(2 (q3-q4))-24 a2 d3 m4 sin(q3)), 1/24 (12 m3 $a2^2$+12 a1 m3 cos(q2) a2+$a3^2$ m3+$a4^2$ m4-6 a1 a3 m4 cos(q2-q3)-$a3^2$ m3 cos(2 q3)+6 a1 a3 m4

$\cos(q2+q3)+a4^2\ m4\ \cos(2\ (q3-q4))-6\ a1\ d3\ m4\ \sin(q2-q3)-6\ a1\ d3\ m4\ \sin(q2+q3))$, $1/12\ a4^2\ m4\ \cos^2(q3-q4)\}$,

$\{1/24\ (-24\ m4\ \cos(q3)\ a1^2-24\ a3\ m4\ \cos(q2)\ a1-18\ a2\ m4\ \cos(q2-q3)\ a1-18\ a2\ m4\ \cos(q2+q3)\ a1-6\ a4\ m4\ \cos(q2-q4)\ a1-6\ a4\ m4\ \cos(q2+q4)\ a1+a3^2\ m3+a4^2\ m4-12\ a2\ a3\ m4-12\ a2^2\ m4\ \cos(q3)-a3^2\ m3\ \cos(2\ q3)+a4^2\ m4\ \cos(2\ (q3-q4))-6\ a2\ a4\ m4\ \cos(q4))$, $1/24\ (12\ m3\ a2^2+12\ a1\ m3\ \cos(q2)\ a2+a3^2\ m3+a4^2\ m4-6\ a1\ a3\ m4\ \cos(q2-q3)-a3^2\ m3\ \cos(2\ q3)+6\ a1\ a3\ m4\ \cos(q2+q3)+a4^2\ m4\ \cos(2\ (q3-q4))-6\ a1\ d3\ m4\ \sin(q2-q3)-6\ a1\ d3\ m4\ \sin(q2+q3))$, $1/24\ (24\ m4\ a1^2+24\ a2\ m4\ \cos(q2)\ a1+6\ a2^2\ m3+a3^2\ m3+6\ a2^2\ m4+a4^2\ m4-a3^2\ m3\ \cos(2\ q3)+a4^2\ m4\ \cos(2\ (q3-q4)))$, $1/24\ m4\ (6\ a2^2+12\ a1\ \cos(q2)\ a2+a4^2+a4^2\ \cos(2\ (q3-q4)))\}$,

$\{1/24\ m4\ (-12\ \cos(q3)\ a2^2-12\ a3\ a2-6\ a1\ \cos(q2-q3)\ a2-6\ a1\ \cos(q2+q3)\ a2-6\ a4\ \cos(q4)\ a2+a4^2+a4^2\ \cos(2\ (q3-q4)))$, $1/12\ a4^2\ m4\ \cos^2(q3-q4)$, $1/24\ m4\ (6\ a2^2+12\ a1\ \cos(q2)\ a2+a4^2+a4^2\ \cos(2\ (q3-q4)))$, $1/24\ m4\ (6\ a2^2+a4^2+a4^2\ \cos(2\ (q3-q4)))\}$

$\}$

4 - Christoffel Symbols of the First Kind

Christoffel Symbols of the First Kind									
I	1	J	1	K	1	C111	m11/q1+m11/q1-m11/q1	C1R1	COLUMN1
				K	2	C112	m21/q1+m21/q1-m11/q2	C1R2	
				K	3	C113	m31/q1+m31/q1-m11/q3	C1R3	
				K	4	C114	m41/q1+m41/q1-m11/q4	C1R4	
		J	2	K	1	C121	m12/q1+m11/q2-m12/q1	C1R1	
				K	2	C122	m22/q1+m21/q2-m12/q2	C1R2	
				K	3	C123	m32/q1+m31/q2-m12/q3	C1R3	
				K	4	C124	m42/q1+m41/q2-m12/q4	C1R4	
		J	3	K	1	C131	m13/q1+m11/q3-m13/q1	C1R1	
				K	2	C132	m23/q1+m21/q3-m13/q2	C1R2	
				K	3	C133	m33/q1+m31/q3-m13/q3	C1R3	
				K	4	C134	m43/q1+m41/q3-m13/q4	C1R4	
		J	4	K	1	C141	m14/q1+m11/q4-m14/q1	C1R1	
				K	2	C142	m24/q1+m21/q4-m14/q2	C1R2	
				K	3	C143	m34/q1+m31/q4-m14/q3	C1R3	
				K	4	C144	m44/q1+m41/q4-m14/q4	C1R4	

I	2	J	1	K	1	C211	m11/q2+m12/q1-m21/q1	C2R1	COLUMN2
				K	2	C212	m21/q2+m22/q1-m21/q2	C2R2	
				K	3	C213	m31/q2+m32/q1-m21/q3	C2R3	
				K	4	C214	m41/q2+m42/q1-m21/q4	C2R4	
		J	2	K	1	C221	m12/q2+m12/q2-m22/q1	C2R1	
				K	2	C222	m22/q2+m22/q2-m22/q2	C2R2	
				K	3	C223	m32/q2+m32/q2-m22/q3	C2R3	
				K	4	C224	m42/q2+m42/q2-m22/q4	C2R4	
		J	3	K	1	C231	m13/q2+m12/q3-m23/q1	C2R1	
				K	2	C232	m23/q2+m22/q3-m23/q2	C2R2	
				K	3	C233	m33/q2+m32/q3-m23/q3	C2R3	
				K	4	C234	m43/q2+m42/q3-m23/q4	C2R4	
		J	4	K	1	C241	m14/q2+m12/q4-m23/q1	C2R1	
				K	2	C242	m24/q2+m22/q4-m23/q2	C2R2	
				K	3	C243	m34/q2+m32/q4-m23/q3	C2R3	
				K	4	C244	m44/q2+m42/q4-m23/q4	C2R4	
I	3	J	1	K	1	C311	m11/q3+m13/q1-m31/q1	C3R1	COLUMN3
				K	2	C312	m21/q3+m23/q1-m31/q2	C3R2	
				K	3	C313	m31/q3+m33/q1-m31/q3	C3R3	
				K	4	C314	m41/q3+m43/q1-m31/q4	C3R4	
		J	2	K	1	C321	m12/q3+m13/q2-m32/q1	C3R1	
				K	2	C322	m22/q3+m23/q2-m32/q2	C3R2	
				K	3	C323	m32/q3+m33/q2-m32/q3	C3R3	
				K	4	C324	m42/q3+m43/q2-m32/q4	C3R4	
		J	3	K	1	C331	m13/q3+m13/q3-m33/q1	C3R1	
				K	2	C332	m23/q3+m23/q3-m33/q2	C3R2	
				K	3	C333	m33/q3+m33/q3-m33/q3	C3R3	
				K	4	C334	m43/q3+m43/q3-m33/q4	C3R4	
		J	4	K	1	C341	m14/q3+m13/q4-m34/q1	C3R1	
				K	2	C342	m24/q3+m23/q4-m34/q2	C3R2	
				K	3	C343	m34/q3+m33/q4-m34/q3	C3R3	
				K	4	C344	m44/q3+m43/q4-m34/q4	C3R4	
I	4	J	1	K	1	C411	m11/q4+m14/q1-m41/q1	C4R1	COLUMN4
				K	2	C412	m21/q4+m24/q1-m41/q2	C4R2	
				K	3	C413	m31/q4+m34/q1-m41/q3	C4R3	
				K	4	C414	m41/q4+m44/q1-m41/q4	C4R4	
		J	2	K	1	C421	m12/q4+m14/q2-m42/q1	C4R1	

				K	2	C422	m22/q4+m24/q2-m42/q2	C4R2	
				K	3	C423	m32/q4+m34/q2-m42/q3	C4R3	
				K	4	C424	m42/q4+m44/q2-m42/q4	C4R4	
		J	3	K	1	C431	m13/q4+m14/q3-m43/q1	C4R1	
				K	2	C432	m23/q4+m24/q3-m43/q2	C4R2	
				K	3	C433	m33/q4+m34/q3-m43/q3	C4R3	
				K	4	C434	m43/q4+m44/q3-m43/q4	C4R4	
		J	4	K	1	C441	m14/q4+m14/q4-m44/q1	C4R1	
				K	2	C442	m24/q4+m24/q4-m44/q2	C4R2	
				K	3	C443	m34/q4+m34/q4-m44/q3	C4R3	
				K	4	C444	m44/q4+m44/q4-m44/q4	C4R4	

Christoffel Symbols of the First Kind Columns preparation

$C_1 = \begin{bmatrix} \sum_{j=1}^{4} c_{1j1}\dot{q}_j \\ \sum_{j=1}^{4} c_{1j2}\dot{q}_j \\ \sum_{1}^{4} c_{1j3}\dot{q}_j \\ \sum_{1}^{4} c_{1j4}\dot{q}_j \end{bmatrix}, Column1$	$C_2 = \begin{bmatrix} \sum_{j=1}^{4} c_{2j1}\dot{q}_j \\ \sum_{j=1}^{4} c_{2j2}\dot{q}_j \\ \sum_{j=1}^{4} c_{2j3}\dot{q}_j \\ \sum_{j=1}^{4} c_{2j4}\dot{q}_j \end{bmatrix}, Column2$
$C_3 = \begin{bmatrix} \sum_{j=1}^{4} c_{3j1}\dot{q}_j \\ \sum_{j=1}^{4} c_{3j2}\dot{q}_j \\ \sum_{j=1}^{4} c_{3j3}\dot{q}_j \\ \sum_{j=1}^{4} c_{3j4}\dot{q}_j \end{bmatrix}, Column3$	$C_4 = \begin{bmatrix} \sum_{j=1}^{4} c_{4j1}\dot{q}_j \\ \sum_{j=1}^{4} c_{4j2}\dot{q}_j \\ \sum_{j=1}^{4} c_{4j3}\dot{q}_j \\ \sum_{j=1}^{4} c_{4j4}\dot{q}_j \end{bmatrix}, Column4$

5 – Preparing M matrices that will be used to calculate c_{ijk} elements

```
M11 = 1/24 (8 a1^2 m1 + 24 a1^2 m2 + 8 a2^2 m2 + 12 a1^2 m3 +
    24 a2^2 m3 + 25 a3^2 m3 + 24 d3^2 m3 + 18 a1^2 m4 + 12 a2^2 m4 +
    24 a3^2 m4 + 4 a4^2 m4 + 24 a1 a2 (m2 + 2 m3 + m4) Cos[q2] +
    6 a1^2 (2 m3 - m4) Cos[2 q2] + 12 a1 a2 m4 Cos[q2 - 2 q3] +
    24 a1 a3 m3 Cos[q2 - q3] + 24 a1 a3 m4 Cos[q2 - q3] +
    3 a1^2 m4 Cos[2 (q2 - q3)] + 48 a2 a3 m3 Cos[q3] +
    48 a2 a3 m4 Cos[q3] - a3^2 m3 Cos[2 q3] + 6 a1^2 m4 Cos[2 q3] +
    12 a2^2 m4 Cos[2 q3] + 24 a1 a3 m3 Cos[q2 + q3] +
    24 a1 a3 m4 Cos[q2 + q3] + 3 a1^2 m4 Cos[2 (q2 + q3)] +
    12 a1 a2 m4 Cos[q2 + 2 q3] + 6 a1 a4 m4 Cos[q2 - q3 - q4] +
    12 a2 a4 m4 Cos[q3 - q4] + a4^2 m4 Cos[2 (q3 - q4)] +
    6 a1 a4 m4 Cos[q2 + q3 - q4] + 24 a3 a4 m4 Cos[q4] +
    3 a4^2 m4 Cos[2 q4] + 6 a1 a4 m4 Cos[q2 - q3 + q4] +
    12 a2 a4 m4 Cos[q3 + q4] + 6 a1 a4 m4 Cos[q2 + q3 + q4] +
    24 a1 d3 m3 Sin[q2 - q3] - 48 a2 d3 m3 Sin[q3] -
    24 a1 d3 m3 Sin[q2 + q3]);
M12 = 1/24 (8 a2^2 m2 + a3^2 m3 + a4^2 m4 + 12 a1 a2 m2 Cos[q2] -
    12 a1 (a3 m3 + a2 m4) Cos[q2 - q3] - 6 a1^2 m4 Cos[2 q2 - q3] -
    a3^2 m3 Cos[2 q3] + 12 a1 a3 m3 Cos[q2 + q3] +
    12 a1 a2 m4 Cos[q2 + q3] + 6 a1^2 m4 Cos[2 q2 + q3] +
```

```
    a4^2 m4 Cos[2 (q3 - q4)] + 12 a1 d3 m4 Sin[q2] -
    12 a1 d3 m3 Sin[q2 - q3] - 12 a1 d3 m3 Sin[q2 + q3]);
M13 = 1/24 (a3^2 m3 - 12 a2 a3 m4 + a4^2 m4 - 24 a1 a3 m4 Cos[q2] -
    18 a1 a2 m4 Cos[q2 - q3] - 24 a1^2 m4 Cos[q3] -
    12 a2^2 m4 Cos[q3] - a3^2 m3 Cos[2 q3] -
    18 a1 a2 m4 Cos[q2 + q3] - 6 a1 a4 m4 Cos[q2 - q4] +
    a4^2 m4 Cos[2 (q3 - q4)] - 6 a2 a4 m4 Cos[q4] -
    6 a1 a4 m4 Cos[q2 + q4]);
M14 = 1/24 m4 (-12 a2 a3 + a4^2 - 6 a1 a2 Cos[q2 - q3] -
    12 a2^2 Cos[q3] - 6 a1 a2 Cos[q2 + q3] + a4^2 Cos[2 (q3 - q4)] -
    6 a2 a4 Cos[q4]);
M21 = 1/24 (8 a2^2 m2 + a3^2 m3 + a4^2 m4 + 12 a1 a2 m2 Cos[q2] -
    12 a1 (a3 m3 + a2 m4) Cos[q2 - q3] - 6 a1^2 m4 Cos[2 q2 - q3] -
    a3^2 m3 Cos[2 q3] + 12 a1 a3 m3 Cos[q2 + q3] +
    12 a1 a2 m4 Cos[q2 + q3] + 6 a1^2 m4 Cos[2 q2 + q3] +
    a4^2 m4 Cos[2 (q3 - q4)] + 12 a1 d3 m4 Sin[q2] -
    12 a1 d3 m3 Sin[q2 - q3] - 12 a1 d3 m3 Sin[q2 + q3]);
M22 = 1/24 (8 a2^2 m2 + 24 a1^2 m3 + 24 a2^2 m3 + a3^2 m3 +
    12 a1^2 m4 + 24 a2^2 m4 + 6 a3^2 m4 + a4^2 m4 + 6 d3^2 m4 +
    48 a1 a2 (m3 + m4) Cos[q2] + 12 a1^2 m4 Cos[2 q2] +
    12 a1 a3 m4 Cos[q2 - q3] + 24 a2 a3 m4 Cos[q3] -
```

```
    a3^2 m3 Cos[2 q3] + 12 a1 a3 m4 Cos[q2 + q3] +
    a4^2 m4 Cos[2 (q3 - q4)] + 12 a1 d3 m4 Sin[q2 - q3] -
    24 a2 d3 m4 Sin[q3] - 12 a1 d3 m4 Sin[q2 + q3]);
M23 = 1/24 (12 a2^2 m3 + a3^2 m3 + a4^2 m4 + 12 a1 a2 m3 Cos[q2] -
    6 a1 a3 m4 Cos[q2 - q3] - a3^2 m3 Cos[2 q3] +
    6 a1 a3 m4 Cos[q2 + q3] + a4^2 m4 Cos[2 (q3 - q4)] -
    6 a1 d3 m4 Sin[q2 - q3] - 6 a1 d3 m4 Sin[q2 + q3]);
M24 = 1/12 a4^2 m4 Cos[q3 - q4]^2;
M31 = 1/24 (a3^2 m3 - 12 a2 a3 m4 + a4^2 m4 - 24 a1 a3 m4 Cos[q2] -
    18 a1 a2 m4 Cos[q2 - q3] - 24 a1^2 m4 Cos[q3] -
    12 a2^2 m4 Cos[q3] - a3^2 m3 Cos[2 q3] -
    18 a1 a2 m4 Cos[q2 + q3] - 6 a1 a4 m4 Cos[q2 - q4] +
    a4^2 m4 Cos[2 (q3 - q4)] - 6 a2 a4 m4 Cos[q4] -
    6 a1 a4 m4 Cos[q2 + q4]);
M32 = 1/24 (12 a2^2 m3 + a3^2 m3 + a4^2 m4 + 12 a1 a2 m3 Cos[q2] -
    6 a1 a3 m4 Cos[q2 - q3] - a3^2 m3 Cos[2 q3] +
    6 a1 a3 m4 Cos[q2 + q3] + a4^2 m4 Cos[2 (q3 - q4)] -
    6 a1 d3 m4 Sin[q2 - q3] - 6 a1 d3 m4 Sin[q2 + q3]);
M33 = 1/24 (6 a2^2 m3 + a3^2 m3 + 24 a1^2 m4 + 6 a2^2 m4 + a4^2 m4 +
    24 a1 a2 m4 Cos[q2] - a3^2 m3 Cos[2 q3] +
    a4^2 m4 Cos[2 (q3 - q4)]);
```

```
M34 = 1/24 m4 (6 a2^2 + a4^2 + 12 a1 a2 Cos[q2] +
    a4^2 Cos[2 (q3 - q4)]);
M41 = 1/24 m4 (-12 a2 a3 + a4^2 - 6 a1 a2 Cos[q2 - q3] -
    12 a2^2 Cos[q3] - 6 a1 a2 Cos[q2 + q3] + a4^2 Cos[2 (q3 - q4)] -
    6 a2 a4 Cos[q4]);
M42 = 1/12 a4^2 m4 Cos[q3 - q4]^2;
M43 = 1/24 m4 (6 a2^2 + a4^2 + 12 a1 a2 Cos[q2] +
    a4^2 Cos[2 (q3 - q4)]);
                M44 = 1/24 m4 (6 a2^2 + a4^2 + a4^2 Cos[2 (q3 - q4)]);
```

6– Computing c_{ijk} elements values

```
(*Christoffel Symbols Of the first Kind calculations*)
(*for I equal to 1*)
C111 = Simplify[D[M11, q1] + D[M11, q1] - D[M11, q1]];
C112 = Simplify[D[M21, q1] + D[M21, q1] - D[M11, q2]];
C113 = Simplify[D[M31, q1] + D[M31, q1] - D[M11, q3]];
C114 = Simplify[D[M41, q1] + D[M41, q1] - D[M11, q4]];
C121 = Simplify[D[M12, q1] + D[M11, q2] - D[M12, q1]];
C122 = Simplify[D[M22, q1] + D[M21, q2] - D[M12, q2]];
C123 = Simplify[D[M32, q1] + D[M31, q2] - D[M12, q3]];
C124 = Simplify[D[M42, q1] + D[M41, q2] - D[M12, q4]];
C131 = Simplify[D[M13, q1] + D[M11, q3] - D[M13, q1]];
C132 = Simplify[D[M23, q1] + D[M21, q3] - D[M13, q2]];
C133 = Simplify[D[M33, q1] + D[M31, q3] - D[M13, q3]];
C134 = Simplify[D[M43, q1] + D[M41, q3] - D[M13, q4]];
C141 = Simplify[D[M14, q1] + D[M11, q4] - D[M14, q1]];
C142 = Simplify[D[M24, q1] + D[M21, q4] - D[M14, q2]];
C143 = Simplify[D[M34, q1] + D[M31, q4] - D[M14, q3]];
C144 = Simplify[D[M44, q1] + D[M41, q4] - D[M14, q4]];
(*for I equal to 2*)
C211 = Simplify[D[M11, q2] + D[M12, q1] - D[M21, q1]];
C212 = Simplify[D[M21, q2] + D[M22, q1] - D[M21, q2]];
C213 = Simplify[D[M31, q2] + D[M32, q1] - D[M21, q3]];
C214 = Simplify[D[M41, q2] + D[M42, q1] - D[M21, q4]];
C221 = Simplify[D[M12, q2] + D[M12, q2] - D[M22, q1]];
C222 = Simplify[D[M22, q2] + D[M22, q2] - D[M22, q2]];
C223 = Simplify[D[M32, q2] + D[M32, q2] - D[M22, q3]];
C224 = Simplify[D[M42, q2] + D[M42, q2] - D[M22, q4]];
C231 = Simplify[D[M13, q2] + D[M12, q3] - D[M23, q1]];
C232 = Simplify[D[M23, q2] + D[M22, q3] - D[M23, q2]];
C233 = Simplify[D[M33, q2] + D[M32, q3] - D[M23, q3]];
C234 = Simplify[D[M43, q2] + D[M42, q3] - D[M23, q4]];
C241 = Simplify[D[M14, q2] + D[M12, q4] - D[M24, q1]];
C242 = Simplify[D[M24, q2] + D[M22, q4] - D[M24, q2]];
C243 = Simplify[D[M34, q2] + D[M32, q4] - D[M24, q3]];
C244 = Simplify[D[M44, q2] + D[M42, q4] - D[M24, q4]];
```

```
(*Christoffel Symbols Of the first Kind calculations*)
(*for I equal to 3*)
C311 = Simplify[D[M11, q3] + D[M13, q1] - D[M31, q1]];
C312 = Simplify[D[M21, q3] + D[M23, q1] - D[M31, q2]];
C313 = Simplify[D[M31, q3] + D[M33, q1] - D[M31, q3]];
C314 = Simplify[D[M41, q3] + D[M43, q1] - D[M31, q4]];
C321 = Simplify[D[M12, q3] + D[M13, q2] - D[M32, q1]];
C322 = Simplify[D[M22, q3] + D[M23, q2] - D[M32, q2]];
C323 = Simplify[D[M32, q3] + D[M33, q2] - D[M32, q3]];
C324 = Simplify[D[M42, q3] + D[M43, q2] - D[M32, q4]];
C331 = Simplify[D[M13, q3] + D[M13, q3] - D[M33, q1]];
C332 = Simplify[D[M23, q3] + D[M23, q3] - D[M33, q2]];
C333 = Simplify[D[M33, q3] + D[M33, q3] - D[M33, q3]];
C234 = Simplify[D[M43, q3] + D[M43, q3] - D[M33, q4]];
C341 = Simplify[D[M14, q3] + D[M13, q4] - D[M34, q1]];
C342 = Simplify[D[M24, q3] + D[M23, q4] - D[M34, q2]];
C343 = Simplify[D[M34, q3] + D[M33, q4] - D[M34, q3]];
C344 = Simplify[D[M44, q3] + D[M43, q4] - D[M34, q4]];
(*for I equal to 4*)
C411 = Simplify[D[M11, q4] + D[M14, q1] - D[M41, q1]];
C412 = Simplify[D[M21, q4] + D[M24, q1] - D[M41, q2]];
C413 = Simplify[D[M31, q4] + D[M34, q1] - D[M41, q3]];
C314 = Simplify[D[M41, q4] + D[M44, q1] - D[M31, q4]];
C421 = Simplify[D[M12, q4] + D[M14, q2] - D[M42, q1]];
C422 = Simplify[D[M22, q4] + D[M24, q2] - D[M42, q2]];
C423 = Simplify[D[M32, q4] + D[M34, q2] - D[M42, q3]];
C424 = Simplify[D[M42, q4] + D[M44, q2] - D[M42, q4]];
C431 = Simplify[D[M13, q4] + D[M14, q3] - D[M43, q1]];
C432 = Simplify[D[M23, q4] + D[M24, q3] - D[M43, q2]];
C433 = Simplify[D[M33, q4] + D[M34, q3] - D[M43, q3]];
C434 = Simplify[D[M43, q4] + D[M44, q3] - D[M43, q4]];
C441 = Simplify[D[M14, q4] + D[M14, q4] - D[M44, q1]];
C442 = Simplify[D[M24, q4] + D[M24, q4] - D[M44, q2]];
C443 = Simplify[D[M34, q4] + D[M34, q4] - D[M44, q3]];
C444 = Simplify[D[M44, q4] + D[M44, q4] - D[M44, q4]];
```

7– Computing Coriolis and Centrifugal 4 by 4 matrix

```
(*Christoffel Symbols with Coriolis and Centrifugal
 4 by 4 matrix calculations*)
(*COLUMN 1 ROW 1 CALCULATIONS*)
C1R1 = Simplify[C111 * q1d + C121 * q2d + C131 * q3d + C141 * q4d];
(*COLUMN 1 ROW 2 CALCULATIONS*)
C1R2 = Simplify[C112 * q1d + C122 * q2d + C132 * q3d + C142 * q4d];
(*COLUMN 1 ROW 3 CALCULATIONS*)
C1R3 = Simplify[C113 * q1d + C123 * q2d + C133 * q3d + C143 * q4d];
(*COLUMN 1 ROW 4 CALCULATIONS*)
C1R4 = Simplify[C114 * q1d + C124 * q2d + C134 * q3d + C144 * q4d];
(*////////////////////////////////*)
(*COLUMN 2 ROW 1 CALCULATIONS*)
C2R1 = Simplify[C211 * q1d + C221 * q2d + C231 * q3d + C241 * q4d];
(*COLUMN 2 ROW 2 CALCULATIONS*)
C2R2 = Simplify[C212 * q1d + C222 * q2d + C232 * q3d + C242 * q4d];
(*COLUMN 2 ROW 3 CALCULATIONS*)
C2R3 = Simplify[C213 * q1d + C223 * q2d + C233 * q3d + C243 * q4d];
(*COLUMN 2 ROW 4 CALCULATIONS*)
C2R4 = Simplify[C214 * q1d + C224 * q2d + C234 * q3d + C244 * q4d];
(*////////////////////////////////*)
(*COLUMN 3 ROW 1 CALCULATIONS*)
C3R1 = Simplify[C311 * q1d + C321 * q2d + C331 * q3d + C341 * q4d];
(*COLUMN 3 ROW 2 CALCULATIONS*)
C3R2 = Simplify[C312 * q1d + C322 * q2d + C332 * q3d + C342 * q4d];
(*COLUMN 3 ROW 3 CALCULATIONS*)
C3R3 = Simplify[C313 * q1d + C323 * q2d + C333 * q3d + C343 * q4d];
(*COLUMN 3 ROW 4 CALCULATIONS*)
C3R4 = Simplify[C314 * q1d + C324 * q2d + C234 * q3d + C344 * q4d];
(*////////////////////////////////*)
(*COLUMN 4 ROW 1 CALCULATIONS*)
C4R1 = Simplify[C411 * q1d + C421 * q2d + C431 * q3d + C441 * q4d];
(*COLUMN 4 ROW 2 CALCULATIONS*)
C4R2 = Simplify[C412 * q1d + C422 * q2d + C432 * q3d + C442 * q4d];
(*COLUMN 4 ROW 3 CALCULATIONS*)
C4R3 = Simplify[C413 * q1d + C423 * q2d + C433 * q3d + C443 * q4d];
(*COLUMN 4 ROW 4 CALCULATIONS*)
C4R4 = Simplify[C414 * q1d + C424 * q2d + C434 * q3d + C444 * q4d];
```

$$\text{CMATRIX} = \text{Simplify}\left[\begin{pmatrix} C1R1 & C2R1 & C3R1 & C4R1 \\ C1R2 & C2R2 & C3R2 & C4R2 \\ C1R3 & C2R3 & C3R3 & C4R3 \\ C1R4 & C2R4 & C3R4 & C4R1 \end{pmatrix}\right] \text{// MatrixForm}$$

8- Computing Torque values for robot Joints

$$\text{TORQUEMATRIX} = \text{Simplify}[\text{MMATRIX}] + \text{Simplify}[\text{CMATRIX}]$$

```
TORQUEMATRIX = ({
 {-2 a1^2 m4 Cos[q2]^2 (q2d^2 + q1d q3d Cos[q3]) Sin[q3] +
  1/24 (m4 q4dd (-12 a2 a3 + a4^2 - 6 a1 a2 Cos[q2 - q3] -
      12 a2^2 Cos[q3] - 6 a1 a2 Cos[q2 + q3] +
      a4^2 Cos[2 (q3 - q4)] - 6 a2 a4 Cos[q4]) -
    q3dd (-a3^2 m3 + 12 a2 a3 m4 - a4^2 m4 +
      24 a1 a3 m4 Cos[q2] + 18 a1 a2 m4 Cos[q2 - q3] +
      24 a1^2 m4 Cos[q3] + 12 a2^2 m4 Cos[q3] +
      a3^2 m3 Cos[2 q3] + 18 a1 a2 m4 Cos[q2 + q3] +
      6 a1 a4 m4 Cos[q2 - q4] - a4^2 m4 Cos[2 (q3 - q4)] +
      6 a2 a4 m4 Cos[q4] + 6 a1 a4 m4 Cos[q2 + q4]) +
    q1dd (8 a1^2 m1 + 24 a1^2 m2 + 8 a2^2 m2 + 12 a1^2 m3 +
      24 a2^2 m3 + 25 a3^2 m3 + 24 d3^2 m3 + 18 a1^2 m4 +
      12 a2^2 m4 + 24 a3^2 m4 + 4 a4^2 m4 +
      24 a1 a2 (m2 + 2 m3 + m4) Cos[q2] +
      6 a1^2 (2 m3 - m4) Cos[2 q2] +
      12 a1 a2 m4 Cos[q2 - 2 q3] + 24 a1 a3 m3 Cos[q2 - q3] +
      24 a1 a3 m4 Cos[q2 - q3] + 3 a1^2 m4 Cos[2 (q2 - q3)] +
      48 a2 a3 m3 Cos[q3] + 48 a2 a3 m4 Cos[q3] -
```

```
    a3^2 m3 Cos[2 q3] + 6 a1^2 m4 Cos[2 q3] +
    12 a2^2 m4 Cos[2 q3] + 24 a1 a3 m3 Cos[q2 + q3] +
    24 a1 a3 m4 Cos[q2 + q3] + 3 a1^2 m4 Cos[2 (q2 + q3)] +
    12 a1 a2 m4 Cos[q2 + 2 q3] +
    6 a1 a4 m4 Cos[q2 - q3 - q4] + 12 a2 a4 m4 Cos[q3 - q4] +
    a4^2 m4 Cos[2 (q3 - q4)] + 6 a1 a4 m4 Cos[q2 + q3 - q4] +
    24 a3 a4 m4 Cos[q4] + 3 a4^2 m4 Cos[2 q4] +
    6 a1 a4 m4 Cos[q2 - q3 + q4] + 12 a2 a4 m4 Cos[q3 + q4] +
    6 a1 a4 m4 Cos[q2 + q3 + q4] + 24 a1 d3 m3 Sin[q2 - q3] -
    48 a2 d3 m3 Sin[q3] - 24 a1 d3 m3 Sin[q2 + q3]) +
  q2dd (8 a2^2 m2 + a3^2 m3 + a4^2 m4 + 12 a1 a2 m2 Cos[q2] -
    12 a1 (a3 m3 + a2 m4) Cos[q2 - q3] -
    6 a1^2 m4 Cos[2 q2 - q3] - a3^2 m3 Cos[2 q3] +
    12 a1 a3 m3 Cos[q2 + q3] + 12 a1 a2 m4 Cos[q2 + q3] +
    6 a1^2 m4 Cos[2 q2 + q3] + a4^2 m4 Cos[2 (q3 - q4)] +
    12 a1 d3 m4 Sin[q2] - 12 a1 d3 m3 Sin[q2 - q3] -
    12 a1 d3 m3 Sin[q2 + q3])) -
 a1 Cos[q2] (-d3 m4 q2d^2 + 4 a1 m3 q1d q2d Sin[q2] +
   2 a3 m3 q2d^2 Sin[q3] + 2 a2 m4 q2d^2 Sin[q3] +
   4 a3 m3 q1d q3d Sin[q3] + 4 a3 m4 q1d q3d Sin[q3] -
   a2 m4 q3d q4d Sin[q3] + 2 a4 m4 q1d q3d Cos[q4] Sin[q3] -
```

```
  a4 m4 q3d q4d Sin[q4] +
  2 Cos[q3] (d3 m3 q2d^2 + 2 d3 m3 q1d q3d +
    a1 m4 q2d q3d Sin[q2] + 4 a2 m4 q1d q3d Sin[q3] +
    a4 m4 q1d q4d Sin[q4])) +
1/6 (6 a1^2 m4 q1d q2d Sin[2 q2] -
  6 a1^2 m4 q1d q2d Cos[2 q3] Sin[2 q2] -
  9 a1 a2 m4 q3d^2 Sin[q2 - q3] +
  3 a1 a2 m4 q2d q4d Sin[q2 - q3] -
  24 a2 a3 m3 q1d q3d Sin[q3] - 24 a2 a3 m4 q1d q3d Sin[q3] +
  12 a1^2 m4 q3d^2 Sin[q3] + 6 a2^2 m4 q3d^2 Sin[q3] +
  6 a2^2 m4 q3d q4d Sin[q3] -
  12 a2 a4 m4 q1d q3d Cos[q4] Sin[q3] +
  12 a1^2 m4 q2d^2 Sin[q2]^2 Sin[q3] -
  6 a1 q2d Sin[
    q2] (2 a2 m2 q1d + 4 a2 m3 q1d + a2 m4 q1d + a2 m2 q2d -
     2 a3 m4 q3d + a2 m4 q1d Cos[2 q3] - a4 m4 q3d Cos[q4] -
     4 d3 m3 q1d Sin[q3] - 2 d3 m3 q3d Sin[q3]) +
  a3^2 m3 q1d q3d Sin[2 q3] - 12 a2^2 m4 q1d q3d Sin[2 q3] +
  a3^2 m3 q2d q3d Sin[2 q3] + a3^2 m3 q3d^2 Sin[2 q3] -
  3 a1^2 m4 q1d q3d Cos[2 q2] Sin[2 q3] +
  a4^2 m4 q1d q4d Cos[q4]^2 Sin[2 q3] -
```

```
    a4^2 m4 q1d q3d Cos[2 q4] Sin[2 q3] +
    9 a1 a2 m4 q3d^2 Sin[q2 + q3] +
    3 a1 a2 m4 q2d q4d Sin[q2 + q3] -
    a4^2 m4 q2d q3d Sin[2 (q3 - q4)] -
    a4^2 m4 q3d^2 Sin[2 (q3 - q4)] +
    a4^2 m4 q2d q4d Sin[2 (q3 - q4)] +
    a4^2 m4 q4d^2 Sin[2 (q3 - q4)] -
    12 a3 a4 m4 q1d q4d Sin[q4] + 3 a2 a4 m4 q3d q4d Sin[q4] +
    3 a2 a4 m4 q4d^2 Sin[q4] -
    a4^2 m4 q1d q4d Cos[2 q3 - q4] Sin[q4] -
    5 a4^2 m4 q1d q4d Cos[q4] Sin[q4] -
    2 a4^2 m4 q1d q3d Cos[q4] Sin[q3]^2 Sin[q4] -
    6 Cos[q3] (a1 q2d (4 a3 m3 q1d + 4 a3 m4 q1d + 2 a3 m3 q3d -
        a2 m4 q3d + 2 a4 m4 q1d Cos[q4]) Sin[q2] +
      q1d (4 a2 d3 m3 q3d + a1^2 m4 q3d Sin[q3] +
        2 a2 a4 m4 q4d Sin[q4])) -
    m4 q1d Cos[
      q3]^2 (12 a1 a2 q2d Sin[q2] +
      a4^2 (-q3d + q4d) Sin[2 q4]))},
{1/24 (2 a4^2 m4 q4dd Cos[q3 - q4]^2 +
    q1dd (8 a2^2 m2 + a3^2 m3 + a4^2 m4 + 12 a1 a2 m2 Cos[q2] -
```

```
    12 a1 (a3 m3 + a2 m4) Cos[q2 - q3] -
    6 a1^2 m4 Cos[2 q2 - q3] - a3^2 m3 Cos[2 q3] +
    12 a1 a3 m3 Cos[q2 + q3] + 12 a1 a2 m4 Cos[q2 + q3] +
    6 a1^2 m4 Cos[2 q2 + q3] + a4^2 m4 Cos[2 (q3 - q4)] +
    12 a1 d3 m4 Sin[q2] - 12 a1 d3 m3 Sin[q2 - q3] -
    12 a1 d3 m3 Sin[q2 + q3]) +
  q2dd (8 a2^2 m2 + 24 a1^2 m3 + 24 a2^2 m3 + a3^2 m3 +
    12 a1^2 m4 + 24 a2^2 m4 + 6 a3^2 m4 + a4^2 m4 +
    6 d3^2 m4 + 48 a1 a2 (m3 + m4) Cos[q2] +
    12 a1^2 m4 Cos[2 q2] + 12 a1 a3 m4 Cos[q2 - q3] +
    24 a2 a3 m4 Cos[q3] - a3^2 m3 Cos[2 q3] +
    12 a1 a3 m4 Cos[q2 + q3] + a4^2 m4 Cos[2 (q3 - q4)] +
    12 a1 d3 m4 Sin[q2 - q3] - 24 a2 d3 m4 Sin[q3] -
    12 a1 d3 m4 Sin[q2 + q3]) +
  q3dd (12 a2^2 m3 + a3^2 m3 + a4^2 m4 + 12 a1 a2 m3 Cos[q2] -
    6 a1 a3 m4 Cos[q2 - q3] - a3^2 m3 Cos[2 q3] +
    6 a1 a3 m4 Cos[q2 + q3] + a4^2 m4 Cos[2 (q3 - q4)] -
    6 a1 d3 m4 Sin[q2 - q3] - 6 a1 d3 m4 Sin[q2 + q3])) +
1/6 (-3 a1 d3 m4 (2 q2d - q3d) q3d Cos[q2 - q3] -
  6 a1 d3 m4 q2d q3d Cos[q2 + q3] -
  3 a1 d3 m4 q3d^2 Cos[q2 + q3] + 6 a1 a2 m2 q1d^2 Sin[q2] +
```

```
12 a1 a2 m3 q1d^2 Sin[q2] + 6 a1 a2 m4 q1d^2 Sin[q2] -

12 a1 a2 m3 q2d^2 Sin[q2] - 12 a1 a2 m4 q2d^2 Sin[q2] -

12 a1 a3 m4 q1d q3d Sin[q2] + 6 a1 a2 m4 q3d^2 Sin[q2] +

6 a1 a2 m4 q3d q4d Sin[q2] -

6 a1^2 m4 q1d^2 Cos[q2] Sin[q2] +

6 a1 a2 m4 q1d^2 Cos[q3]^2 Sin[q2] -

6 a1 a4 m4 q1d q3d Cos[q4] Sin[q2] +

6 a1 Cos[
  q3] (a3 m4 (2 q1d^2 - q2d^2) + 2 a3 m3 q1d (q1d - q3d) -
   5 a2 m4 q1d q3d + a4 m4 q1d^2 Cos[q4]) Sin[q2] +

6 a1^2 m3 q1d^2 Sin[2 q2] - 6 a1^2 m4 q2d^2 Sin[2 q2] +

3 a1^2 m4 q1d^2 Cos[2 q3] Sin[2 q2] -

6 m4 q3d Cos[q3] (2 a2 d3 q2d + a1^2 q1d Sin[2 q2]) +

6 a1 a3 m4 q2d q3d Sin[q2 - q3] -

3 a1 a3 m4 q3d^2 Sin[q2 - q3] -

3 a1 a2 m4 q1d q4d Sin[q2 - q3] -

12 a2 a3 m4 q2d q3d Sin[q3] -

12 a1 d3 m3 q1d^2 Sin[q2] Sin[q3] +

6 a1 d3 m4 q2d^2 Sin[q2] Sin[q3] +

12 a1 d3 m3 q1d q3d Sin[q2] Sin[q3] -

6 a1 a2 m4 q1d^2 Sin[q2] Sin[q3]^2 +
```

```
    a3^2 m3 q1d q3d Sin[2 q3] + a3^2 m3 q2d q3d Sin[2 q3] +
    a3^2 m3 q3d^2 Sin[2 q3] - 6 a1 a3 m4 q2d q3d Sin[q2 + q3] -
    3 a1 a3 m4 q3d^2 Sin[q2 + q3] -
    3 a1 a2 m4 q1d q4d Sin[q2 + q3] -
    a4^2 m4 q1d q3d Sin[2 (q3 - q4)] -
    a4^2 m4 q2d q3d Sin[2 (q3 - q4)] -
    a4^2 m4 q3d^2 Sin[2 (q3 - q4)] +
    a4^2 m4 q1d q4d Sin[2 (q3 - q4)] +
    a4^2 m4 q2d q4d Sin[2 (q3 - q4)] +
    a4^2 m4 q4d^2 Sin[2 (q3 - q4)])},
{1/24 (m4 q4dd (6 a2^2 + a4^2 + 12 a1 a2 Cos[q2] +
     a4^2 Cos[2 (q3 - q4)]) +
    q3dd (6 a2^2 m3 + a3^2 m3 + 24 a1^2 m4 + 6 a2^2 m4 +
     a4^2 m4 + 24 a1 a2 m4 Cos[q2] - a3^2 m3 Cos[2 q3] +
     a4^2 m4 Cos[2 (q3 - q4)]) -
    q1dd (-a3^2 m3 + 12 a2 a3 m4 - a4^2 m4 +
     24 a1 a3 m4 Cos[q2] + 18 a1 a2 m4 Cos[q2 - q3] +
     24 a1^2 m4 Cos[q3] + 12 a2^2 m4 Cos[q3] +
     a3^2 m3 Cos[2 q3] + 18 a1 a2 m4 Cos[q2 + q3] +
     6 a1 a4 m4 Cos[q2 - q4] - a4^2 m4 Cos[2 (q3 - q4)] +
     6 a2 a4 m4 Cos[q4] + 6 a1 a4 m4 Cos[q2 + q4]) +
```

```
q2dd (12 a2^2 m3 + a3^2 m3 + a4^2 m4 + 12 a1 a2 m3 Cos[q2] -
    6 a1 a3 m4 Cos[q2 - q3] - a3^2 m3 Cos[2 q3] +
    6 a1 a3 m4 Cos[q2 + q3] + a4^2 m4 Cos[2 (q3 - q4)] -
    6 a1 d3 m4 Sin[q2 - q3] - 6 a1 d3 m4 Sin[q2 + q3])) +
1/12 (6 a1 a2 m4 q1d q4d Sin[q2 - q3] +
  24 a2 a3 m3 q1d^2 Sin[q3] + 24 a2 a3 m4 q1d^2 Sin[q3] +
  12 a2 a3 m4 q2d^2 Sin[q3] - 12 a2^2 m4 q1d q4d Sin[q3] +
  24 a1 a3 m3 q1d^2 Cos[q2] Sin[q3] +
  24 a1 a3 m4 q1d^2 Cos[q2] Sin[q3] +
  12 a2 a4 m4 q1d^2 Cos[q4] Sin[q3] +
  12 a1 a4 m4 q1d^2 Cos[q2] Cos[q4] Sin[q3] -
  12 a1 q2d Sin[
    q2] (-2 a3 m4 q1d + a2 m3 q2d + 2 a2 m4 q3d + a2 m4 q4d -
     a4 m4 q1d Cos[q4] + 2 d3 m3 q1d Sin[q3]) +
  2 Cos[q3] (12 a2 d3 m3 q1d^2 + 6 a2 d3 m4 q2d^2 +
     6 a1 (2 a3 m3 + 5 a2 m4) q1d q2d Sin[q2] +
     6 a1^2 m4 q1d q2d Sin[2 q2] - a3^2 m3 q1d^2 Sin[q3] +
     3 a1^2 m4 q1d^2 Sin[q3] - a3^2 m3 q2d^2 Sin[q3] +
     12 a1 q1d^2 Cos[q2] (d3 m3 + 2 a2 m4 Sin[q3])) +
  12 a2^2 m4 q1d^2 Sin[2 q3] - 2 a3^2 m3 q1d q2d Sin[2 q3] +
  a3^2 m3 q3d^2 Sin[2 q3] +
```

```
    6 a1^2 m4 q1d^2 Cos[q2]^2 Sin[2 q3] +
    3 a1^2 m4 q1d^2 Cos[2 q2] Sin[2 q3] +
    a4^2 m4 q1d^2 Cos[2 q4] Sin[2 q3] -
    6 a1 a2 m4 q1d q4d Sin[q2 + q3] -
    6 a1 a4 m4 q1d q4d Sin[q2 - q4] +
    2 a4^2 m4 q1d q2d Sin[2 (q3 - q4)] +
    a4^2 m4 q2d^2 Sin[2 (q3 - q4)] -
    a4^2 m4 q3d^2 Sin[2 (q3 - q4)] +
    4 a4^2 m4 q1d q4d Sin[2 (q3 - q4)] +
    4 a4^2 m4 q2d q4d Sin[2 (q3 - q4)] +
    2 a4^2 m4 q3d q4d Sin[2 (q3 - q4)] +
    3 a4^2 m4 q4d^2 Sin[2 (q3 - q4)] +
    6 a2 a4 m4 q1d q4d Sin[q4] -
    2 a4^2 m4 q1d^2 Cos[q3]^2 Cos[q4] Sin[q4] +
    a4^2 m4 q1d^2 Sin[q3]^2 Sin[2 q4] +
    6 a1 a4 m4 q1d q4d Sin[q2 + q4])},
{1/24 m4 (2 a4^2 q2dd Cos[q3 - q4]^2 +
    q4dd (6 a2^2 + a4^2 + a4^2 Cos[2 (q3 - q4)]) +
    q3dd (6 a2^2 + a4^2 + 12 a1 a2 Cos[q2] +
      a4^2 Cos[2 (q3 - q4)]) +
    q1dd (-12 a2 a3 + a4^2 - 6 a1 a2 Cos[q2 - q3] -
```

```
      12 a2^2 Cos[q3] - 6 a1 a2 Cos[q2 + q3] +
      a4^2 Cos[2 (q3 - q4)] - 6 a2 a4 Cos[q4])) +
  1/12 m4 (q2d (3 a1 a2 q1d Sin[q2 - q3] +
      3 a1 a2 q1d Sin[q2 + q3] -
      a4^2 (q1d + q2d + 3 q3d) Sin[2 (q3 - q4)]) -
    q3d (6 a1 a2 q2d Sin[q2] +
      a4 (a4 (2 q2d + 3 q3d + q4d) Sin[2 (q3 - q4)] +
        6 a1 q1d Cos[q2] Sin[q4])) +
    q4d (q2d (3 a1 a2 Sin[q2 - q3] + 3 a1 a2 Sin[q2 + q3] +
        a4^2 Sin[2 (q3 - q4)]) +
      2 a4 q4d (a4 Sin[2 (q3 - q4)] + 3 a2 Sin[q4]) +
      3 q3d (2 a2 (a2 + a1 Cos[q2]) Sin[q3] +
        a4 (a2 + 2 a1 Cos[q2]) Sin[q4]) -
      a4 q1d ((12 a3 + a4 Cos[2 q3 - q4] + 5 a4 Cos[q4]) Sin[
          q4] - 2 Cos[
          q3] (a4 Cos[q4]^2 Sin[q3] -
           6 (a2 + a1 Cos[q2]) Sin[q4]) +
        a4 Cos[q3]^2 Sin[2 q4])) +
    q1d (q2d (3 a1 a2 Sin[q2 - q3] + 3 a1 a2 Sin[q2 + q3] -
        a4^2 Sin[2 (q3 - q4)]) +
      a4 q1d ((12 a3 + a4 Cos[2 q3 - q4] + 5 a4 Cos[q4]) Sin[
```

```
q4] - 2 Cos[
q3] (a4 Cos[q4]^2 Sin[q3] -
6 (a2 + a1 Cos[q2]) Sin[q4]) +
a4 Cos[q3]^2 Sin[2 q4]) -
q3d (3 a1 a2 Sin[q2 - q3] - 6 a2^2 Sin[q3] -
3 a1 a2 Sin[q2 + q3] - 3 a1 a4 Sin[q2 - q4] +
2 a4^2 Sin[2 (q3 - q4)] + 3 a2 a4 Sin[q4] +
3 a1 a4 Sin[q2 + q4])))}
                                        });
                T1 = Simplify[TORQUEMATRIX [[1, 1]]]
                T2 = Simplify[TORQUEMATRIX [[2, 1]]]
                T3 = Simplify[TORQUEMATRIX [[3, 1]]]
               T4 = Simplify[TORQUEMATRIX[[4, 1]]]
```

9- Drawing Torque Values Graphs

```
a1=12;
a2=12;
a3=8;
a4=8.5;
d3=8;
tpp=1.9; (*time range*)
tstep=0.001;  (*time step*)
```

```
tpath=2t+6;(*time dependent function*)

tdpath=1;(*time dependent function derivative*)

tddpath=0;

q1value=35;

q2value=55;

q3value=-40;

q4value=50;

q1=Table[(47+q1value)*tpath*π/180,{t,1,tpp,tstep}];

q2=Table[(191+q2value)*tpath*π/180,{t,1,tpp,tstep}];

q3=Table[(164+q3value)*tpath*π/180,{t,1,tpp,tstep}];

q4=Table[(180+q4value)*tpath*π/180,{t,1,tpp,tstep}];

q1d=Table[q1value*tdpath*π/180,{t,1,tpp,tstep}];

q2d=Table[q2value*tdpath*π/180,{t,1,tpp,tstep}];

q3d=Table[q3value*tdpath*π/180,{t,1,tpp,tstep}];

q4d=Table[q4value*tdpath*π/180,{t,1,tpp,tstep}];

q1dd=Table[q1value*tddpath*π/180,{t,1,tpp,tstep}];

q2dd=Table[q2value*tddpath*π/180,{t,1,tpp,tstep}];

q3dd=Table[q3value*tddpath*π/180,{t,1,tpp,tstep}];

q4dd=Table[q4value*tddpath*π/180,{t,1,tpp,tstep}];

m1=0.07;

m2=0.07;
```

m3=0.05;

m4=0.05;

T1=1/24 (m4 q4dd (-12 a2 a3+a4^2-6 a1 a2 Cos[q2-q3]-12 a2^2 Cos[q3]-6 a1 a2 Cos[q2+q3]+a4^2 Cos[2 (q3-q4)]-6 a2 a4 Cos[q4])-q3dd (-a3^2 m3+12 a2 a3 m4-a4^2 m4+24 a1 a3 m4 Cos[q2]+18 a1 a2 m4 Cos[q2-q3]+24 a1^2 m4 Cos[q3]+12 a2^2 m4 Cos[q3]+a3^2 m3 Cos[2 q3]+18 a1 a2 m4 Cos[q2+q3]+6 a1 a4 m4 Cos[q2-q4]-a4^2 m4 Cos[2 (q3-q4)]+6 a2 a4 m4 Cos[q4]+6 a1 a4 m4 Cos[q2+q4])-48 a1^2 m4 Cos[q2]^2 (q2d^2+q1d q3d Cos[q3]) Sin[q3]+q1dd (8 a1^2 m1+24 a1^2 m2+8 a2^2 m2+12 a1^2 m3+24 a2^2 m3+25 a3^2 m3+24 d3^2 m3+18 a1^2 m4+12 a2^2 m4+24 a3^2 m4+4 a4^2 m4+24 a1 a2 (m2+2 m3+m4) Cos[q2]+6 a1^2 (2 m3-m4) Cos[2 q2]+12 a1 a2 m4 Cos[q2-2 q3]+24 a1 a3 m3 Cos[q2-q3]+24 a1 a3 m4 Cos[q2-q3]+3 a1^2 m4 Cos[2 (q2-q3)]+48 a2 a3 m3 Cos[q3]+48 a2 a3 m4 Cos[q3]-a3^2 m3 Cos[2 q3]+6 a1^2 m4 Cos[2 q3]+12 a2^2 m4 Cos[2 q3]+24 a1 a3 m3 Cos[q2+q3]+24 a1 a3 m4 Cos[q2+q3]+3 a1^2 m4 Cos[2 (q2+q3)]+12 a1 a2 m4 Cos[q2+2 q3]+6 a1 a4 m4 Cos[q2-q3-q4]+12 a2 a4 m4 Cos[q3-q4]+a4^2 m4 Cos[2 (q3-q4)]+6 a1 a4 m4 Cos[q2+q3-q4]+24 a3 a4 m4 Cos[q4]+3 a4^2 m4 Cos[2 q4]+6 a1 a4 m4 Cos[q2-q3+q4]+12 a2 a4 m4 Cos[q3+q4]+6 a1 a4 m4 Cos[q2+q3+q4]+24 a1 d3 m3 Sin[q2-q3]-48 a2 d3 m3 Sin[q3]-24 a1 d3 m3 Sin[q2+q3])+q2dd (8 a2^2 m2+a3^2 m3+a4^2 m4+12 a1 a2 m2 Cos[q2]-12 a1 (a3 m3+a2 m4) Cos[q2-q3]-6 a1^2 m4 Cos[2 q2-q3]-a3^2 m3 Cos[2 q3]+12 a1 a3 m3 Cos[q2+q3]+12 a1 a2 m4 Cos[q2+q3]+6 a1^2 m4 Cos[2 q2+q3]+a4^2 m4 Cos[2 (q3-q4)]+12 a1 d3 m4 Sin[q2]-12 a1 d3 m3 Sin[q2-q3]-12 a1 d3 m3 Sin[q2+q3])-24 a1 Cos[q2] (-d3 m4 q2d^2+4 a1 m3 q1d q2d Sin[q2]+2 a3 m3 q2d^2 Sin[q3]+2 a2 m4 q2d^2 Sin[q3]+4 a3 m3 q1d q3d Sin[q3]+4 a3 m4 q1d q3d Sin[q3]-a2 m4 q3d q4d Sin[q3]+2 a4 m4 q1d q3d Cos[q4] Sin[q3]-a4 m4 q3d q4d Sin[q4]+2 Cos[q3] (d3 m3 q2d^2+2 d3 m3 q1d q3d+a1 m4 q2d q3d Sin[q2]+4 a2 m4 q1d q3d Sin[q3]+a4 m4 q1d q4d Sin[q4]))+4 (6 a1^2 m4 q1d q2d Sin[2 q2]-6 a1^2 m4 q1d q2d Cos[2 q3] Sin[2 q2]-9 a1 a2 m4 q3d^2 Sin[q2-q3]+3 a1 a2 m4 q2d q4d Sin[q2-q3]-24 a2 a3 m3 q1d q3d Sin[q3]-24 a2 a3 m4 q1d q3d Sin[q3]+12 a1^2 m4 q3d^2 Sin[q3]+6 a2^2 m4 q3d^2 Sin[q3]+6 a2^2 m4 q3d q4d Sin[q3]-12 a2 a4 m4 q1d q3d Cos[q4] Sin[q3]+12 a1^2 m4 q2d^2 Sin[q2]^2 Sin[q3]-6 a1 q2d Sin[q2] (2 a2 m2 q1d+4 a2 m3 q1d+a2 m4 q1d+a2 m2 q2d-2 a3 m4 q3d+a2 m4 q1d Cos[2 q3]-a4 m4 q3d Cos[q4]-4 d3 m3 q1d Sin[q3]-2 d3 m3 q3d Sin[q3])+a3^2 m3 q1d q3d Sin[2 q3]-12 a2^2 m4 q1d q3d Sin[2 q3]+a3^2 m3 q2d q3d

Sin[2 q3]+a3^2 m3 q3d^2 Sin[2 q3]-3 a1^2 m4 q1d q3d Cos[2 q2] Sin[2 q3]+a4^2 m4 q1d q4d Cos[q4]^2 Sin[2 q3]-a4^2 m4 q1d q3d Cos[2 q4] Sin[2 q3]+9 a1 a2 m4 q3d^2 Sin[q2+q3]+3 a1 a2 m4 q2d q4d Sin[q2+q3]-a4^2 m4 q2d q3d Sin[2 (q3-q4)]-a4^2 m4 q3d^2 Sin[2 (q3-q4)]+a4^2 m4 q2d q4d Sin[2 (q3-q4)]+a4^2 m4 q4d^2 Sin[2 (q3-q4)]-12 a3 a4 m4 q1d q4d Sin[q4]+3 a2 a4 m4 q3d q4d Sin[q4]+3 a2 a4 m4 q4d^2 Sin[q4]-a4^2 m4 q1d q4d Cos[2 q3-q4] Sin[q4]-5 a4^2 m4 q1d q4d Cos[q4] Sin[q4]-2 a4^2 m4 q1d q3d Cos[q4] Sin[q3]^2 Sin[q4]-6 Cos[q3] (a1 q2d (4 a3 m3 q1d+4 a3 m4 q1d+2 a3 m3 q3d-a2 m4 q3d+2 a4 m4 q1d Cos[q4]) Sin[q2]+q1d (4 a2 d3 m3 q3d+a1^2 m4 q3d Sin[q3]+2 a2 a4 m4 q4d Sin[q4]))-m4 q1d Cos[q3]^2 (12 a1 a2 q2d Sin[q2]+a4^2 (-q3d+q4d) Sin[2 q4])));

T2=1/24 (2 a4^2 m4 q4dd Cos[q3-q4]^2+q1dd (8 a2^2 m2+a3^2 m3+a4^2 m4+12 a1 a2 m2 Cos[q2]-12 a1 (a3 m3+a2 m4) Cos[q2-q3]-6 a1^2 m4 Cos[2 q2-q3]-a3^2 m3 Cos[2 q3]+12 a1 a3 m3 Cos[q2+q3]+12 a1 a2 m4 Cos[q2+q3]+6 a1^2 m4 Cos[2 q2+q3]+a4^2 m4 Cos[2 (q3-q4)]+12 a1 d3 m4 Sin[q2]-12 a1 d3 m3 Sin[q2-q3]-12 a1 d3 m3 Sin[q2+q3])+q2dd (8 a2^2 m2+24 a1^2 m3+24 a2^2 m3+a3^2 m3+12 a1^2 m4+24 a2^2 m4+6 a3^2 m4+a4^2 m4+6 d3^2 m4+48 a1 a2 (m3+m4) Cos[q2]+12 a1^2 m4 Cos[2 q2]+12 a1 a3 m4 Cos[q2-q3]+24 a2 a3 m4 Cos[q3]-a3^2 m3 Cos[2 q3]+12 a1 a3 m4 Cos[q2+q3]+a4^2 m4 Cos[2 (q3-q4)]+12 a1 d3 m4 Sin[q2-q3]-24 a2 d3 m4 Sin[q3]-12 a1 d3 m4 Sin[q2+q3])+q3dd (12 a2^2 m3+a3^2 m3+a4^2 m4+12 a1 a2 m3 Cos[q2]-6 a1 a3 m4 Cos[q2-q3]-a3^2 m3 Cos[2 q3]+6 a1 a3 m4 Cos[q2+q3]+a4^2 m4 Cos[2 (q3-q4)]-6 a1 d3 m4 Sin[q2-q3]-6 a1 d3 m4 Sin[q2+q3])+4 (-3 a1 d3 m4 (2 q2d-q3d) q3d Cos[q2-q3]-6 a1 d3 m4 q2d q3d Cos[q2+q3]-3 a1 d3 m4 q3d^2 Cos[q2+q3]+6 a1 a2 m2 q1d^2 Sin[q2]+12 a1 a2 m3 q1d^2 Sin[q2]+6 a1 a2 m4 q1d^2 Sin[q2]-12 a1 a2 m3 q2d^2 Sin[q2]-12 a1 a2 m4 q2d^2 Sin[q2]-12 a1 a3 m4 q1d q3d Sin[q2]+6 a1 a2 m4 q3d^2 Sin[q2]+6 a1 a2 m4 q3d q4d Sin[q2]-6 a1^2 m4 q1d^2 Cos[q2] Sin[q2]+6 a1 a2 m4 q1d^2 Cos[q3]^2 Sin[q2]-6 a1 a4 m4 q1d q3d Cos[q4] Sin[q2]+6 a1 Cos[q3] (a3 m4 (2 q1d^2-q2d^2)+2 a3 m3 q1d (q1d-q3d)-5 a2 m4 q1d q3d+a4 m4 q1d^2 Cos[q4]) Sin[q2]+6 a1^2 m3 q1d^2 Sin[2 q2]-6 a1^2 m4 q2d^2 Sin[2 q2]+3 a1^2 m4 q1d^2 Cos[2 q3] Sin[2 q2]-6 m4 q3d Cos[q3] (2 a2 d3 q2d+a1^2 q1d Sin[2 q2])+6 a1 a3 m4 q2d q3d Sin[q2-q3]-3 a1 a3 m4 q3d^2 Sin[q2-q3]-3 a1 a2 m4 q1d q4d Sin[q2-q3]-12 a2 a3 m4 q2d q3d Sin[q3]-12 a1 d3 m3 q1d^2 Sin[q2] Sin[q3]+6 a1 d3 m4 q2d^2 Sin[q2] Sin[q3]+12 a1 d3 m3 q1d q3d Sin[q2] Sin[q3]-6 a1 a2 m4 q1d^2 Sin[q2] Sin[q3]^2+a3^2 m3 q1d q3d Sin[2 q3]+a3^2 m3 q2d q3d Sin[2 q3]+a3^2 m3 q3d^2 Sin[2 q3]-6 a1 a3 m4 q2d q3d Sin[q2+q3]-3 a1 a3 m4 q3d^2

Sin[q2+q3]-3 a1 a2 m4 q1d q4d Sin[q2+q3]-a4^2 m4 q1d q3d Sin[2 (q3-q4)]-a4^2 m4 q2d q3d Sin[2 (q3-q4)]-a4^2 m4 q3d^2 Sin[2 (q3-q4)]+a4^2 m4 q1d q4d Sin[2 (q3-q4)]+a4^2 m4 q2d q4d Sin[2 (q3-q4)]+a4^2 m4 q4d^2 Sin[2 (q3-q4)]));

T3=1/24 (m4 q4dd (6 a2^2+a4^2+12 a1 a2 Cos[q2]+a4^2 Cos[2 (q3-q4)])+q3dd (6 a2^2 m3+a3^2 m3+24 a1^2 m4+6 a2^2 m4+a4^2 m4+24 a1 a2 m4 Cos[q2]-a3^2 m3 Cos[2 q3]+a4^2 m4 Cos[2 (q3-q4)])-q1dd (-a3^2 m3+12 a2 a3 m4-a4^2 m4+24 a1 a3 m4 Cos[q2]+18 a1 a2 m4 Cos[q2-q3]+24 a1^2 m4 Cos[q3]+12 a2^2 m4 Cos[q3]+a3^2 m3 Cos[2 q3]+18 a1 a2 m4 Cos[q2+q3]+6 a1 a4 m4 Cos[q2-q4]-a4^2 m4 Cos[2 (q3-q4)]+6 a2 a4 m4 Cos[q4]+6 a1 a4 m4 Cos[q2+q4])+q2dd (12 a2^2 m3+a3^2 m3+a4^2 m4+12 a1 a2 m3 Cos[q2]-6 a1 a3 m4 Cos[q2-q3]-a3^2 m3 Cos[2 q3]+6 a1 a3 m4 Cos[q2+q3]+a4^2 m4 Cos[2 (q3-q4)]-6 a1 d3 m4 Sin[q2-q3]-6 a1 d3 m4 Sin[q2+q3])+2 (6 a1 a2 m4 q1d q4d Sin[q2-q3]+24 a2 a3 m3 q1d^2 Sin[q3]+24 a2 a3 m4 q1d^2 Sin[q3]+12 a2 a3 m4 q2d^2 Sin[q3]-12 a2^2 m4 q1d q4d Sin[q3]+24 a1 a3 m3 q1d^2 Cos[q2] Sin[q3]+24 a1 a3 m4 q1d^2 Cos[q2] Sin[q3]+12 a2 a4 m4 q1d^2 Cos[q4] Sin[q3]+12 a1 a4 m4 q1d^2 Cos[q2] Cos[q4] Sin[q3]-12 a1 q2d Sin[q2] (-2 a3 m4 q1d+a2 m3 q2d+2 a2 m4 q3d+a2 m4 q4d-a4 m4 q1d Cos[q4]+2 d3 m3 q1d Sin[q3])+2 Cos[q3] (12 a2 d3 m3 q1d^2+6 a2 d3 m4 q2d^2+6 a1 (2 a3 m3+5 a2 m4) q1d q2d Sin[q2]+6 a1^2 m4 q1d q2d Sin[2 q2]-a3^2 m3 q1d^2 Sin[q3]+3 a1^2 m4 q1d^2 Sin[q3]-a3^2 m3 q2d^2 Sin[q3]+12 a1 q1d^2 Cos[q2] (d3 m3+2 a2 m4 Sin[q3]))+12 a2^2 m4 q1d^2 Sin[2 q3]-2 a3^2 m3 q1d q2d Sin[2 q3]+a3^2 m3 q3d^2 Sin[2 q3]+6 a1^2 m4 q1d^2 Cos[q2]^2 Sin[2 q3]+3 a1^2 m4 q1d^2 Cos[2 q2] Sin[2 q3]+a4^2 m4 q1d^2 Cos[2 q4] Sin[2 q3]-6 a1 a2 m4 q1d q4d Sin[q2+q3]-6 a1 a4 m4 q1d q4d Sin[q2-q4]+2 a4^2 m4 q1d q2d Sin[2 (q3-q4)]+a4^2 m4 q2d^2 Sin[2 (q3-q4)]-a4^2 m4 q3d^2 Sin[2 (q3-q4)]+4 a4^2 m4 q1d q4d Sin[2 (q3-q4)]+4 a4^2 m4 q2d q4d Sin[2 (q3-q4)]+2 a4^2 m4 q3d q4d Sin[2 (q3-q4)]+3 a4^2 m4 q4d^2 Sin[2 (q3-q4)]+6 a2 a4 m4 q1d q4d Sin[q4]-2 a4^2 m4 q1d^2 Cos[q3]^2 Cos[q4] Sin[q4]+a4^2 m4 q1d^2 Sin[q3]^2 Sin[2 q4]+6 a1 a4 m4 q1d q4d Sin[q2+q4]));

T4=1/24 m4 (2 a4^2 q2dd Cos[q3-q4]^2+q4dd (6 a2^2+a4^2+a4^2 Cos[2 (q3-q4)])+q3dd (6 a2^2+a4^2+12 a1 a2 Cos[q2]+a4^2 Cos[2 (q3-q4)])+q1dd (-12 a2 a3+a4^2-6 a1 a2 Cos[q2-q3]-12 a2^2 Cos[q3]-6 a1 a2 Cos[q2+q3]+a4^2 Cos[2 (q3-q4)]-6 a2 a4 Cos[q4])+2 (q2d (3 a1 a2 q1d Sin[q2-q3]+3 a1 a2 q1d Sin[q2+q3]-a4^2 (q1d+q2d+3 q3d) Sin[2 (q3-q4)])-q3d (6 a1 a2 q2d Sin[q2]+a4 (a4 (2 q2d+3 q3d+q4d) Sin[2 (q3-q4)]+6 a1 q1d Cos[q2] Sin[q4]))+q4d (q2d (3 a1 a2 Sin[q2-q3]+3 a1 a2 Sin[q2+q3]+a4^2 Sin[2 (q3-q4)])+2 a4

q4d (a4 Sin[2 (q3-q4)]+3 a2 Sin[q4])+3 q3d (2 a2 (a2+a1 Cos[q2]) Sin[q3]+a4 (a2+2 a1 Cos[q2]) Sin[q4])-a4 q1d ((12 a3+a4 Cos[2 q3-q4]+5 a4 Cos[q4]) Sin[q4]-2 Cos[q3] (a4 $Cos[q4]^2$ Sin[q3]-6 (a2+a1 Cos[q2]) Sin[q4])+a4 $Cos[q3]^2$ Sin[2 q4]))+q1d (q2d (3 a1 a2 Sin[q2-q3]+3 a1 a2 Sin[q2+q3]-$a4^2$ Sin[2 (q3-q4)])+a4 q1d ((12 a3+a4 Cos[2 q3-q4]+5 a4 Cos[q4]) Sin[q4]-2 Cos[q3] (a4 $Cos[q4]^2$ Sin[q3]-6 (a2+a1 Cos[q2]) Sin[q4])+a4 $Cos[q3]^2$ Sin[2 q4])-q3d (3 a1 a2 Sin[q2-q3]-6 $a2^2$ Sin[q3]-3 a1 a2 Sin[q2+q3]-3 a1 a4 Sin[q2-q4]+2 $a4^2$ Sin[2 (q3-q4)]+3 a2 a4 Sin[q4]+3 a1 a4 Sin[q2+q4])));

```
(*Plotting Torque 1*)
ListPlot[T1, PlotLabel → "First Joint Torque", Filling → Axis, PlotStyle → Directive[Blue]]
(*Plotting Torque 2*)
ListPlot[T2, PlotLabel → "Second Joint Torque", Filling → Axis, PlotStyle → Directive[Orange]]
(*Plotting Torque 3*)
ListPlot[T3, PlotLabel → "Third Joint Torque", Filling → Axis, PlotStyle → Directive[Green]]
(*Plotting Torque 4*)
ListPlot[T4, PlotLabel → "Fourth Joint Torque", Filling → Axis, PlotStyle → Directive[Red]]
(*Plotting Torque1,Torque2,Torque3,Torque4 *)
ListLinePlot[{T1, T2, T3, T4}, Filling → Axis, PlotLegends → {"TORQUE1", "TORQUE2", "TORQUE3"
   , "TORQUE4"}, PlotLabel → "T1,T2,T3,T4"]
```

Appendix C Arduino Microcontroller Codes

1- Codes for 3DoF Parallel Robot Manipulator

```
#include <math.h>
#include <Wire.h>
#include <Servo.h>
int del=10;
float xi = -10, xf = 12, yi =-8, yf = 15;
int L = 50; //links length
float xp = 76; //footprint x extention
float yp = 44; //footprint y extention
float yR = 87.5; //radius of second manipulator
float px1, py1,px2,py2,px3,py3; //x,y end effector cooordination
float r = 26, fi = 30; //platform radius and orientation
float dx;   //platform x-center cordination gyro
float dy; //platform y-center cordination gyro
float rx = r * cos(30 * PI / 180), ry = r * sin(30 * PI / 180);
//platform center
//gyroscope and servo
Servo tests,tests2,tests3;        //initialize a servo object for
the connected servo
```

```
const int MPU_ADDR = 0x68; // I2C address of the MPU-6050. If AD0
pin is set to HIGH, the I2C address will be 0x69.
int16_t accelerometer_x, accelerometer_y, accelerometer_z; //
variables for accelerometer raw data
int16_t gyro_x, gyro_y, gyro_z; // variables for gyro raw data
int16_t temperature; // variables for temperature data
char tmp_str[7]; // temporary variable used in convert function
int16_t xs = 0, ys = 0, zs = 0;
int16_t xgs = 0, ygs = 0, zgs = 0;
char* convert_int16_to_str(int16_t i)
{ // converts int16 to string. Moreover, resulting strings will
have the same length in the debug monitor.
  sprintf(tmp_str, "%6d", i);
  return tmp_str;
}
//gyro and servo finish

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Wire.begin();
  Wire.beginTransmission(MPU_ADDR); // Begins a transmission to
the I2C slave (GY-521 board)
  Wire.write(0x6B); // PWR_MGMT_1 register
  Wire.write(0); // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
  tests.attach(9);
  tests2.attach(10);
  tests3.attach(11);
}

void loop() {
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
[MPU-6000 and MPU-6050 Register Map and Descriptions Revision
4.2, p.40]
  Wire.endTransmission(false); // the parameter indicates that
the Arduino will send a restart. As a result, the connection is
kept active.
  Wire.requestFrom(MPU_ADDR, 7 * 2, true); // request a total of
7*2=14 registers
  // "Wire.read()<<8 | Wire.read();" means two registers are read
and stored in the same variable
  accelerometer_x = Wire.read() << 8 | Wire.read(); // reading
registers: 0x3B (ACCEL_XOUT_H) and 0x3C (ACCEL_XOUT_L)
```

```
  accelerometer_y = Wire.read() << 8 | Wire.read(); // reading
registers: 0x3D (ACCEL_YOUT_H) and 0x3E (ACCEL_YOUT_L)
  accelerometer_z = Wire.read() << 8 | Wire.read(); // reading
registers: 0x3F (ACCEL_ZOUT_H) and 0x40 (ACCEL_ZOUT_L)
  temperature = Wire.read() << 8 | Wire.read(); // reading
registers: 0x41 (TEMP_OUT_H) and 0x42 (TEMP_OUT_L)
  gyro_x = Wire.read() << 8 | Wire.read(); // reading registers:
0x43 (GYRO_XOUT_H) and 0x44 (GYRO_XOUT_L)
  gyro_y = Wire.read() << 8 | Wire.read(); // reading registers:
0x45 (GYRO_YOUT_H) and 0x46 (GYRO_YOUT_L)
  gyro_z = Wire.read() << 8 | Wire.read(); // reading registers:
0x47 (GYRO_ZOUT_H) and 0x48 (GYRO_ZOUT_L)
  //accelerometer mapping values
  /*dx = map(accelerometer_x, -32768, 32767, xi, xf);
  dy = map(accelerometer_y, -32768, 32767, yi, yf);
  zs = map(accelerometer_y, -32768, 32767, 0, 180);*/
  //delay(1000);
  //gyroscope mapping values
  dx = map(gyro_x, -32768, 32768, xi, xf);
  dy= map(gyro_y, -32768, 32768, yi, yf);
  zgs = map(gyro_z, -32768, 32767, 0, 180);
  //tests.write(ys);                  //command to rotate the
servo to the specified angle
  //robot geometrics and inverse kinematics
  //for first manipulator
  px1 = -(xp - 13) + dx; //cordination of x position of end
effictor of manipulator one
  py1 = (yp + 7.5) + dy;  //cordination of y position of end
effictor of manipulator one

  float a = sqrt((pow(px1, 2) + pow(py1, 2)) / 4); //geometrics
  float h = sqrt(pow(L, 2) - pow(a, 2));
  float alfa = atan2(h, a) * 180 / PI;
  float beta = atan2(py1, px1) * 180 / PI;

  float theta, theta1, thet1;
  if (beta < 0)
  {
    theta1 = (alfa + abs(beta));
    theta = 360 - theta1;
    tests.write(theta);
    Serial.print("servo1\t  ");
    Serial.println(theta);
    delay(del);
  }
  else
```

```
  {
    thet1 = -alfa + abs(beta);
    tests.write(theta);
    Serial.print("servo1\t  ");
    Serial.println(theta);
    delay(del);
    if (thet1 < 0)
    {
      theta = thet1 + 360;
      tests.write(theta);
      Serial.print("servo1\t  ");
    Serial.println(theta);
      delay(del);
    }
    else
    {
      theta = thet1;
      tests.write(theta);
      Serial.print("servo1\t  ");
    Serial.println(theta);
      delay(del);
    }
  }

  //for second manipulator
  px2 = dx-13; //cordination of x position of end effictor of
manipulator one
  py2 = -(yR-7.5)+dy;  //cordination of y position of end
effictor of manipulator one

  float a2 = sqrt((pow(px2, 2) + pow(py2, 2)) / 4); //geometrics
  float h2 = sqrt(pow(L, 2) - pow(a2, 2));
  float alfa2 = atan2(h2, a2) * 180 / PI;
  float beta2 = atan2(py2, px2) * 180 / PI;

  float theta22, theta2, thet2;
  if (beta2 < 0)
  {
    theta22 = (alfa2 + abs(beta2));
    theta2 = 360 - theta22;
    tests2.write(theta2);
     Serial.print("servo2\t  ");
    Serial.println(theta2);
    delay(del);
  }
  else
```

```
  {
    thet2 = -alfa2 + abs(beta2);
    tests2.write(theta2);
     Serial.print("servo2\t  ");
    Serial.println(theta2);
    delay(del);
    if (thet2 < 0)
    {
      theta2 = thet2 + 360;
      tests2.write(theta2);
      Serial.print("servo2\t  ");
    Serial.println(theta2);
      delay(del);
    }
    else
    {
      theta2 = thet2;
      tests2.write(theta2);
       Serial.print("servo2\t  ");
    Serial.println(theta2);
      delay(del);
    }  }
 //for third manipulator
  px3 =xp+dx; //cordination of x position of end effictor of
manipulator one
  py3 =(yp-15)+dy;  //cordination of y position of end effictor
of manipulator one
  float a3 = sqrt((pow(px3, 2) + pow(py3, 2)) / 4); //geometrics
  float h3 = sqrt(pow(L, 2) - pow(a3, 2));
  float alfa3 = atan2(h3, a3) * 180 / PI;
  float beta3 = atan2(py3, px3) * 180 / PI;
  float theta33, theta3, thet3;
  if (beta3 < 0)
  {
    theta33 = (alfa3 + abs(beta3));
    theta3 = 360 - theta33;
    tests3.write(theta3);
     Serial.print("servo3\t  ");
    Serial.println(theta3);
    delay(del);  }
  else
 {
    thet3 = -alfa3 + abs(beta3);
    tests3.write(theta3);
    Serial.print("servo3\t  ");
    Serial.println(theta3);
```

```
    delay(del);
    if (thet3 < 0)
    {
      theta3 = thet3 + 360;
      tests3.write(theta3);
      Serial.print("servo3\t  ");
    Serial.println(theta3);
      delay(del);
    }
    else
    {
      theta3 = thet3;
      tests3.write(theta3);
      Serial.print("servo3 \t");
    Serial.println(theta3);
      delay(del);
    }
  }
}
```

2- Communication and control of Agrobot robotic Arm using ROS and Arduino Microcontroller.

```
#if defined(ARDUINO) && ARDUINO >= 100
  #include "Arduino.h"
#else
  #include <WProgram.h>
#endif
//////////////////ROS
#include "Wire.h"
#define MIN_PULSE_WIDTH       650
#define MAX_PULSE_WIDTH       2350
#define FREQUENCY             50
#include "Arduino.h"
#include "ArduinoHardware.h"
#include <ros.h>
#include <std_msgs/UInt16.h>
#include <std_msgs/Int16MultiArray.h>
#include <sensor_msgs/JointState.h>

// Set ROS - handler, subscribe message, publish message
(debugging)
ros::NodeHandle  nh;
std_msgs::Int16MultiArray str_msg2;
```

```
ros::Publisher chatter("Robot_Joints_Angles", &str_msg2);
int servoDegree[5];
int degreetolx15a[5];
// Define Motor position variables
double mtrDegreeBase;
double mtrDegreeSholder;
double mtrDegreeElbow;
double mtrDegreeWrist;
double mtrDegreePivot;
double mtrDegreeJaws;

//////////////////
#include <Stepper.h>
//BYTES LIMITS DEFINTIONS
#define GET_LOW_BYTE(A) (uint8_t)((A))
//Macro function  get lower 8 bits of A
#define GET_HIGH_BYTE(A) (uint8_t)((A) >> 8)
//Macro function  get higher 8 bits of A
#define BYTE_TO_HW(A, B) ((((uint16_t)(A)) << 8) | (uint8_t)(B))
//put A as higher 8 bits   B as lower 8 bits   which amalgamated
into 16 bits integer
///SERVO CONTROLLING PARAMETERS DEFINITIONS
#define
LOBOT_SERVO_FRAME_HEADER         0x55   //0x55   received  indica
ting  the  arrival  of  data packets.
#define LOBOT_SERVO_MOVE_TIME_WRITE      1
//SERVO IDs DEFINITIONS
#define ID1   1
#define ID2   2
#define ID3   3
#define ID4   4
#define ID5   5
int motor_delays=250;
// Define LX16A position variables
double mtrDegreeBaseLX=195;
double mtrDegreeSholderLX=795;
double mtrDegreeElbowLX=687;
double mtrDegreeWristLX=748;
double mtrDegreePivotLX;
double mtrDegreeJawsLX;
////////////////////ROS callback function
// Function move motor to ROS angle
void servo_cb(const sensor_msgs::JointState& cmd_msg)
{
  /*mtrDegreeBase =
trimLimits(radiansToDegrees(cmd_msg.position[0]));
```

```
  mtrDegreeSholder =
trimLimits(radiansToDegrees(cmd_msg.position[1]));
  mtrDegreeElbow =
trimLimits(radiansToDegrees(cmd_msg.position[2]));
  mtrDegreeWrist =
trimLimits(radiansToDegrees(cmd_msg.position[3]));
  mtrDegreePivot =
trimLimits(radiansToDegrees(cmd_msg.position[4]));*/
  //
  mtrDegreeBase = (radiansToDegrees(cmd_msg.position[0]));
  mtrDegreeSholder = (radiansToDegrees(-cmd_msg.position[1]));
  mtrDegreeElbow =(radiansToDegrees(cmd_msg.position[2]));
  mtrDegreeWrist = (radiansToDegrees(cmd_msg.position[3]));
  mtrDegreePivot =(radiansToDegrees(cmd_msg.position[4]));
  //

  // Store motor movements for publishing back to ROS (debugging)
  servoDegree[0] = mtrDegreeBase;
  servoDegree[1] = mtrDegreeSholder;
  servoDegree[2] = mtrDegreeElbow;
  servoDegree[3] = mtrDegreeWrist;
  servoDegree[4] = mtrDegreePivot;
//motorBase.write(mtrDegreeBase);
//motorSholder.write(mtrDegreeSholder);
//motorElbow.write(mtrDegreeElbow);
//motorWrist.write(mtrDegreeWrist);
/*motorPivot.write(mtrDegreePivot);*/

//LX16A
 mtrDegreeBaseLX=DegreestoLX16A(mtrDegreeBase);
 mtrDegreeSholderLX=DegreestoLX16A(mtrDegreeSholder-(30));
 mtrDegreeElbowLX=DegreestoLX16A(mtrDegreeElbow);
 mtrDegreeWristLX=DegreestoLX16A(mtrDegreeWrist);
 mtrDegreePivotLX=DegreestoLX16A(mtrDegreePivot);
//store LX16A VALUES
// Store motor movements for publishing back to ROS (debugging)
  degreetolx15a[0] = mtrDegreeBaseLX;
  degreetolx15a[1] = mtrDegreeSholderLX;
  degreetolx15a[2] = mtrDegreeElbowLX;
  degreetolx15a[3] = mtrDegreeWristLX;
  degreetolx15a[4] = mtrDegreePivotLX;
///
LobotSerialServoMove(Serial1, ID1,mtrDegreeBaseLX, motor_delays);
LobotSerialServoMove(Serial1, ID2,mtrDegreeSholderLX,
motor_delays);
```

```
LobotSerialServoMove(Serial1, ID3,mtrDegreeElbowLX,
motor_delays);
/*LobotSerialServoMove(Serial1, ID4,mtrDegreeWristLX, 1000);
//LobotSerialServoMove(Serial1, ID4,mtrDegreeWristLX, 1000);*/
}
////////////////////ROS callback function finish
/////////ROS subscribe handling
ros::Subscriber<sensor_msgs::JointState> sub("joint_states",
servo_cb);
void setup() {
  // put your setup code here, to run once:
 Serial1.begin(115200);
 Serial.begin(115200);
  // Setup ROS fir subscribe and publish
    nh.getHardware()->setBaud(115200);
    nh.initNode();
    nh.subscribe(sub);
    nh.advertise(chatter);

}
void loop() {
   //str_msg2.data = servoDegree;
   //lx16a
  str_msg2.data = degreetolx15a;
    str_msg2.data_length = 5;
    chatter.publish( &str_msg2 );
    nh.spinOnce();
     //put your main code here, to run repeatedly:
  /*LobotSerialServoMove(Serial1, ID5,900, 1000);
  Serial.println("turn1");

  delay(4000);
  LobotSerialServoMove(Serial1, ID5, 100, 1000);
  Serial.println("turn2");
  delay(4000);*/
}
                    ///*****FUNCTION DEFINITIONS FOR SERVO MOTOR
PARAMETERS START /*****
//#define LOBOT_DEBUG 1  /*Debug  : print debug
value        (function defintion)*/
byte LobotCheckSum(byte buf[])
{
  byte i;
  uint16_t temp = 0;
  for (i = 2; i < buf[3] + 2; i++) {
    temp += buf[i];
```

```
  }
  temp = ~temp;
  i = (byte)temp;
  return i;
}
//function to move servo postion with time setting
void LobotSerialServoMove(HardwareSerial &SerialX, uint8_t id,
int16_t position1, uint16_t time1)
{
  byte buf[10];
  if(position1 < 0)
  {
    position1 = 0;
  }
  if(position1 > 1000)
  {
     position1 = 1000;
  }
  buf[0] = buf[1] = LOBOT_SERVO_FRAME_HEADER;//header defintion
with hexadeicemel 0
  buf[2] = id;// servo id byte position
  buf[3] = 7;
  buf[4] = LOBOT_SERVO_MOVE_TIME_WRITE;
  buf[5] = GET_LOW_BYTE(position1);
  buf[6] = GET_HIGH_BYTE(position1);
  buf[7] = GET_LOW_BYTE(time1);
  buf[8] = GET_HIGH_BYTE(time1);
  buf[9] = LobotCheckSum(buf);
  SerialX.write(buf, 10);//write byffer length
}
// Convert radians to degreees
  double radiansToDegrees(float position_radians)
  {/*
    position_radians = position_radians * 57.2958;
    return position_radians;*/

  position_radians = position_radians + 1.6;

  return position_radians * 57.2958;

  }

  // Convert degrees to LX-16A values
double DegreestoLX16A(float position_LX16A)
  {
```

```
  position_LX16A = position_LX16A*(100/18);
  //
  if(position_LX16A < 0)
  {
    position_LX16A = 0;
  }
  if(position_LX16A > 1000)
  {
     position_LX16A = 1000;
  }
  //
  return  position_LX16A;
  }
```

Appendix D ROS packages and Codes

URDF Codes

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

<!-- This URDF was automatically created by SolidWorks to URDF Exporter! Originally created by Stephen Brawner (brawner@gmail.com)

Commit Version: 1.6.0-4-g7f85cfe Build Version: 1.6.7995.38578

For more information, please see http://wiki.ros.org/sw_urdf_exporter -->

<robot

name="aibomech_agrobot_v2">

<!-- * * * Link color for Rviz * * * -->

<material name="blue">

<color rgba="0 0 0.8 1"/>

</material>

<material name="red">

```
    <color rgba="0.8 0 0 1"/>
  </material>
  <material name="green">
    <color rgba="0 0.8 0 1"/>
  </material>
  <material name="yellow">
    <color rgba="1 1 0.2 1"/>
  </material>
  <!-- * * * Gazebo controller  * * * -->
  <gazebo>
    <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
      <robotNamespace>/</robotNamespace>
    </plugin>
  </gazebo>
  <!--
  <gazebo>
   <plugin                                    name="joint_state_publisher"
filename="libgazebo_ros_joint_state_publisher.so">

<jointName>revolute_1,revolute_2,revolute_3,revolute_4,prismatic_1</jointName>
   </plugin>
  </gazebo>-->
  <!-- * * * Link Definitions * * * -->
```

```
<link name="world"/>
<link name="base_link">
  <inertial>
    <origin
      xyz="-0.010581 0.10145 0.0"
      rpy="0 0 0" />
    <mass
      value="0.2" />
    <inertia
      ixx="0.000025417"
      ixy="0.0"
      ixz="0.0"
      iyy="0.0006121"
      iyz="0.0"
      izz="0.0006167" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
```

```
        filename="package://aibomech_agrobot_v2/meshes/base_link.STL" />
    </geometry>
    <material name="blue"/>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://aibomech_agrobot_v2/meshes/base_link.STL" />
    </geometry>
  </collision>
</link>
<gazebo reference="base_link">
    <kp>1000.0</kp>
    <kd>10.0</kd>
    <mu1>10.0</mu1>
    <mu2>10.0</mu2>
    <material>Gazebo/Blue</material>
</gazebo>
```

```
<joint name="fixed" type="fixed">
    <parent link="world"/>
    <child link="base_link"/>
    <origin rpy="0 0 0" xyz="0.0 0.0 0.05"/>
</joint>
<link
 name="link_1">
 <inertial>
  <origin
   xyz="-0.0659 0.0125 0"
   rpy="0 0 0" />
  <mass
   value="0.15" />
  <inertia
   ixx="0.000014063"
   ixy="0.0"
   ixz="0.0"
   iyy="0.00025486"
   iyz="0.0"
   izz="0.0002633" />
 </inertial>
 <visual>
```

```
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://aibomech_agrobot_v2/meshes/link_1.STL" />
    </geometry>
    <material name="red"/>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://aibomech_agrobot_v2/meshes/link_1.STL" />
    </geometry>
  </collision>
</link>
<gazebo reference="link_1">
    <kp>1000.0</kp>
    <kd>10.0</kd>
```

```
    <mu1>10.0</mu1>
    <mu2>10.0</mu2>
    <material>Gazebo/Red</material>
</gazebo>
<joint
  name="revolute_1"
  type="revolute">
  <origin
    xyz="-0.0905 0 0.1735"
    rpy="1.5708 0 -0.65897" />
  <parent
    link="base_link" />
  <child
    link="link_1" />
  <axis
    xyz="0 1 0" />
  <limit
    lower="-1.09"
    upper="2.1"
    effort="150"
    velocity="0" />
</joint>
```

```
<link
 name="link_2">
 <inertial>
  <origin
   xyz="-0.067695 0.012623 -4.1633E-17"
   rpy="0 0 0" />
  <mass
   value="0.15" />
  <inertia
   ixx="0.000014063"
   ixy="0.0"
   ixz="0.0"
   iyy="0.00026926"
   iyz="0.0"
   izz="0.0002777" />
 </inertial>
 <visual>
  <origin
   xyz="0 0 0"
   rpy="0 0 0" />
  <geometry>
   <mesh
```

```
        filename="package://aibomech_agrobot_v2/meshes/link_2.STL" />
    </geometry>
    <material name="yellow"/>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://aibomech_agrobot_v2/meshes/link_2.STL" />
    </geometry>
  </collision>
</link>
<gazebo reference="link_2">
    <kp>1000.0</kp>
    <kd>10.0</kd>
    <mu1>10.0</mu1>
    <mu2>10.0</mu2>
    <material>Gazebo/Yellow</material>
</gazebo>
```

```
<joint
  name="revolute_2"
  type="revolute">
  <origin
    xyz="-0.12 0 0"
    rpy="0 0.60223 0" />
  <parent
    link="link_1" />
  <child
    link="link_2" />
  <axis
    xyz="0 1 0" />
  <limit
    lower="-2.1"
    upper="1.11"
    effort="150"
    velocity="0" />
</joint>
<link
  name="link_3">
  <inertial>
    <origin
```

```
    xyz="-0.082394 -5.5511E-17 0.042771"
    rpy="0 0 0" />
  <mass
    value="0.081686" />
  <inertia
    ixx="0.00016721"
    ixy="0.0"
    ixz="0.0"
    iyy="0.0002399"
    iyz="0.0"
    izz="7.7701E-05" />
</inertial>
<visual>
  <origin
    xyz="0 0 0"
    rpy="0 0 0" />
  <geometry>
    <mesh
      filename="package://aibomech_agrobot_v2/meshes/link_3.STL" />
  </geometry>
  <material name="green"/>
</visual>
```

```
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://aibomech_agrobot_v2/meshes/link_3.STL" />
    </geometry>
  </collision>
</link>
<gazebo reference="link_3">
    <kp>1000.0</kp>
    <kd>10.0</kd>
    <mu1>10.0</mu1>
    <mu2>10.0</mu2>
    <material>Gazebo/Green</material>
</gazebo>
<joint
  name="revolute_3"
  type="revolute">
  <origin
    xyz="-0.12 0.013028 0"
```

```
    rpy="1.5708 0 0.49429" />
  <parent
    link="link_2" />
  <child
    link="link_3" />
  <axis
    xyz="0 1 0" />
  <limit
    lower="-2"
    upper="1.1"
    effort="5"
    velocity="0" />
</joint>
<link
  name="link_4">
  <inertial>
    <origin
      xyz="-0.044405 0.073633 -0.0018404"
      rpy="0 0 0" />
    <mass
      value="0.10" />
    <inertia
```

```
    ixx="0.000024375"
    ixy="0.0"
    ixz="0.0"
    iyy="0.00004208"
    iyz="0.0"
    izz="0.00003271" />
</inertial>
<visual>
  <origin
    xyz="0 0 0"
    rpy="0 0 0" />
  <geometry>
    <mesh
      filename="package://aibomech_agrobot_v2/meshes/link_4.STL" />
  </geometry>
  <material name="blue"/>
</visual>
<collision>
  <origin
    xyz="0 0 0"
    rpy="0 0 0" />
  <geometry>
```

```
      <mesh
        filename="package://aibomech_agrobot_v2/meshes/link_4.STL" />
    </geometry>
  </collision>
</link>
<gazebo reference="link_4">
    <kp>1000.0</kp>
    <kd>10.0</kd>
    <mu1>10.0</mu1>
    <mu2>10.0</mu2>
    <material>Gazebo/Blue</material>
</gazebo>
<joint
  name="revolute_4"
  type="revolute">
  <origin
    xyz="-0.041826 0 0.115"
    rpy="0 0.88733 1.5708" />
  <parent
    link="link_3" />
  <child
    link="link_4" />
```

```
  <axis
    xyz="0 1 0" />
  <limit
    lower="-1"
    upper="2.3"
    effort="10"
    velocity="0" />
</joint>
<link
  name="link_5">
  <inertial>
    <origin
      xyz="0.016259 0.041204 0.0025"
      rpy="0 0 0" />
    <mass
      value="0.05" />
    <inertia
      ixx="0.0000020833"
      ixy="0.0"
      ixz="0.0"
      iyy="0.00005208"
      iyz="0.0"
```

```
      izz="0.00005083" />
    </inertial>
    <visual>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://aibomech_agrobot_v2/meshes/link_5.STL" />
      </geometry>
      <material name="red"/>
    </visual>
    <collision>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://aibomech_agrobot_v2/meshes/link_5.STL" />
      </geometry>
    </collision>
  </link>
```

```
<gazebo reference="link_5">
    <kp>1000.0</kp>
    <kd>10.0</kd>
    <mu1>10.0</mu1>
    <mu2>10.0</mu2>
    <material>Gazebo/Red</material>
</gazebo>
<joint
  name="prismatic_1"
  type="prismatic">
  <origin
    xyz="-0.075 0.080174 0"
    rpy="-0.84443 -1.5708 2.4684" />
  <parent
    link="link_4" />
  <child
    link="link_5" />
  <axis
    xyz="1 0 0" />
  <limit
    lower="-0.022"
    upper="0.01"
```

```
    effort="10"
    velocity="0" />
  </joint>
  <!-- * * * Motor and Hardware Definitions * * * -->
  <transmission name="tran1">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="revolute_1">
<hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
    </joint>
    <actuator name="motor1">
  <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>
  <transmission name="tran2">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="revolute_2">
<hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
    </joint>
    <actuator name="motor2">
 <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
```

```
    </actuator>
  </transmission>
  <transmission name="tran3">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="revolute_3">
   <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
    </joint>
    <actuator name="motor3">
<hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>
  <transmission name="tran4">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="revolute_4">
<hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
    </joint>
    <actuator name="motor4">
   <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
    </actuator>
```

```
  </transmission>
  <transmission name="tran5">
     <type>transmission_interface/SimpleTransmission</type>
     <joint name="prismatic_1">
   <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
     </joint>
     <actuator name="motor5">
<hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
     </actuator>
  </transmission>
</robot>
```

ROS Configuration Codes

```
joint_state_controller:
  type: joint_state_controller/JointStateController
  publish_rate: 50
arm_controller:
  type: position_controllers/JointTrajectoryController
  joints:
    - revolute_1
```

```
  - revolute_2
  - revolute_3
  - revolute_4
  - prismatic_1
gains:
  revolute_1:
    p: 100
    d: 1
    i: 1
    i_clamp: 1
  revolute_2:
    p: 100
    d: 1
    i: 1
    i_clamp: 1
  revolute_3:
    p: 100
    d: 1
```

```
  i: 1
  i_clamp: 1
revolute_4:
  p: 100
  d: 1
  i: 1
  i_clamp: 1
prismatic_1:
  p: 100
  d: 1
  i: 1
  i_clamp: 1
```

Republic of Turkey
İzmir Kâtip Çelebi University
Graduate School of Natural and Applied Sciences

Automating Farming Operations Using Robotic Technologies

Department of Robotics Engineering

Master's / Doctoral Thesis

Basheer Altawil
ORCID 0000-0002-5716-7587

Thesis Advisor: Asst. Prof. Dr. Fatih Cemal CAN

January 2023

Altawil	Automating Farming Operations Using Robotics Technologies	MASTER’ S THESIS	2023

Curriculum Vitae

Name Surname : Basheer Altawil

Education:

2014–2019 İzmir Kâtip Çelebi University, Dept. of Mechatronics Eng.

2020–2023 İzmir Kâtip Çelebi University, Dept. of Robotics Eng.

Work Experience:

2017	Summer Internship	Botaş – İzmir
2018	Summer Internship	Delta process Automation - İzmir
2019	Robotics Engineer	FabLab- İzmir municipality
2020	R&D Engineer	Lazermarket – İzmir
2021-Continue…	Co-Founder	iFarm Startup

Publications (if any):

1. Automating Farming Operations Using Robotic Technologies, 10th International Conference on Advanced Technologies (ICAT'22) ,from page 353 to page 359).