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

Design and Development of a Compound Mobile Serial Robot with Remote Control Application

Submitted to the Institute of Science and Technology in partial fulfillment of the requirements for the degree of

Master of Science of Engineering

in Robotics Engineering

by

Alpay Toprak

ORCID 0000-0003-3417-8926

August, 2023

This is to certify that we have read the thesis **Design and Development of a Compound Mobile Serial Robot with Remote Control Application** submitted by **Alpay Toprak**, 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: **Assist. Prof. Dr. Duygu ATCI**
İzmir Katip Celebi University

Committee Members:

Assist. Prof. Fatih Cemal CAN
İzmir Katip Celebi University

Prof. Dr. H. Seçil ARTEM
İzmir Institute of Technology

Date of Defense: August 7, 2023

Declaration of Authorship

I, **Alpay Toprak**, declare that this thesis titled **Design and Development of a Compound Mobile Serial Robot with Remote Control Application** 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 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: 07.08.2023

Design and Development of a Compound Mobile Serial Robot with Remote Control Application

Abstract

The design of a compound mobile serial robot controlled remotely by a C# program utilizing the MQTT (Message Queuing Telemetry Transport) protocol is the main topic of this thesis. The robot is powered by a Raspberry Pi. The advantages of mobility and serial manipulators are combined in the compound mobile serial robot, allowing it to carry out difficult tasks in a variety of settings. As the main computer, the Raspberry Pi gives the robot and its control application connectivity and processing capability. Data interchange and command execution are made possible by the integration of the MQTT protocol, which guarantees effective and dependable connection between the robot and the control system. The integration of the MQTT protocol, hardware components, mechanical design, kinematic and dynamic analysis of the robot, and software implementation are all investigated in this study. This thesis intends to stimulate additional ideas in the field of intelligent and remotely operated robotic systems by utilizing the capabilities of Raspberry Pi and MQTT. For the protocol implementation, a 4 DoF serial arm and the mobile station connected to it were designed with mechanical design and prototype production was made, additionally direct and inverse kinematics of the robot arm has automatized with the C# WinForm application, thanks to this developed application, robot arm control was fully automatized and controlled. The thesis has demonstrated a novel applicacation of the protocol to a compound mobile serial robot, only there is no video for both mobile and serial robot arm working at the same time.

Keywords: Compound mobile serial robot, Remote Control, C# Application, MQTT Protocol, Raspberry Pi, Robotics, Kinematic and Dynamic Analysis.

Uzaktan Kontrol Uygulamalı Bileşik Mobil Seri Robot Tasarımı ve Geliştirilmesi

Öz

Bu tez, Raspberry Pi tarafından güçlendirilen MQTT (Message Queuing Telemetry Transport) protokolünü kullanarak uzaktan kontrol edilen bir bileşik hareketli seri robotun tasarımı ve geliştirilmesine odaklanmaktadır. Bileşik hareketli seri robot, hareket kabiliyeti ve seri manipülatörlerin avantajlarını bir araya getirerek çeşitli ortamlarda karmaşık görevleri gerçekleştirebilme yeteneğine sahiptir. Raspberry Pi, robotun ve kontrol uygulamasının bağlantı ve işleme yeteneklerini sağlayan merkezi bir bilgisayar olarak hizmet vermektedir. MQTT protokolünün entegrasyonu, robot ile kontrol sistemi arasında etkili ve güvenilir iletişimi sağlayarak gerçek zamanlı veri alışverişi ve komut yürütme imkanı sunmaktadır. Bu araştırma kapsamında, donanım bileşenleri, mekanik tasarım, robotun kinematik ve dinamik analizi ile yazılım uygulaması incelenmektedir. Bu tez, Raspberry Pi ve MQTT'nin yeteneklerinden faydalanarak zeki ve uzaktan kontrol edilebilen robot sistemleri alanında ilave fikirlerin ortaya çıkmasını amaçlamaktadır. Protokol implementasyonu için mekanik tasarım ile 4 serbestlik dereceli bir seri kol ve buna bağlı mobil istasyon tasarlandı ve prototip üretim yapıldı, aynı zamanda tüm ileri ve ters kinematik analizler C# uygulamasına entegre edeldi ve gerçek zamanlı veri değişimi ile robot kontrolü sağlandı. Tez, yukarıdakilerin uygulanabilirliğini göstermiştir. Mobil ve robot kolunun birlikte aynı anda çalışırken bir videosu bulunmamaktadır.

Anahtar Kelimeler: Bileşik Mobil Seri Robot, Uzaktan Kontrol, C# Uygulaması, MQTT Protokolü, Raspberry Pi, Robotik, Akıllı Sistemler, Kinematik ve Dinamik Analiz.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

IKC	Izmir Katip Celebi University
DoF	Degree of Freedom

List of Symbols

θ	Joint Angle
α	Torsion Angle
S	Joint Offset
a	Length of the Link

1 Introduction

In recent years, the field of robotics has made incredible strides, changing numerous sectors and pushing the limits of human capability. Mobile serial robots have become a dynamic and adaptable solution in this field, enabling detailed movements and complex operations in a range of settings. Parallel to this, a paradigm change has been brought about by the Internet of Things (IoT), which connects systems and gadgets in novel ways. Intelligent and remote-controlled robotic systems have been made possible by the fusion of robotics and the internet of things (IoT).

This thesis aims to explore the complexities of designing and developing a compound mobile serial robot, powered by a remote control C# WinForm Desktop application. The MQTT (Message Queuing Telemetry Transport) protocol's integration is crucial since it will allow for flawless coordination and communication between the robot and its control system. The robot may be remotely controlled while enabling real-time data interchange and decision-making processes by taking advantage of this protocol's possibilities. Importantly, the Raspberry Pi will act as the system's main computer as client, controlling the robot and its control application with the connectivity and processing capacity as needed.

The compound mobile serial robot, which combines the benefits of mobility and serial manipulators, provides a revolutionary approach to robotics. The robot can carry out a variety of duties thanks to its special combination, including negotiating challenging terrain, handling objects and inspection of them and performing accurate moves in various environments. The C# program also offers a simple and easy-to-use interface for controlling the robot's functions and motions remotely without being same network in safe, additionally automatized calculation and control of the servo motors by the tool. Users can enter desired angle to the servo motor, can calculate servo motor angles by end-effector position of the gripper and run the robot arm joints with those angles.

The MQTT protocol is used to greatly increase the efficiency and dependability of communication between the robot and the control system. The publish-subscribe architecture of the protocol allows for seamless data transmission and command execution between the robot and the control application. One of the most crucial points is that it offers a secure workplace for challenging tasks.

In this thesis, all the different aspects of designing and creating the compound mobile serial robot will be explored. The hardware components, mechanical design, workspace, kinematics, and dynamic analysis of the robot will be examined. Additionally, Raspberry Pi's computational capability and its interoperability with communication, actuator, and communication modules will also be analyzed. The software implementation, based on the creation of the C# remote control application and the integration of the MQTT protocol for fluid communication using the Python language for servo motor control on Raspberry Pi, will be studied.

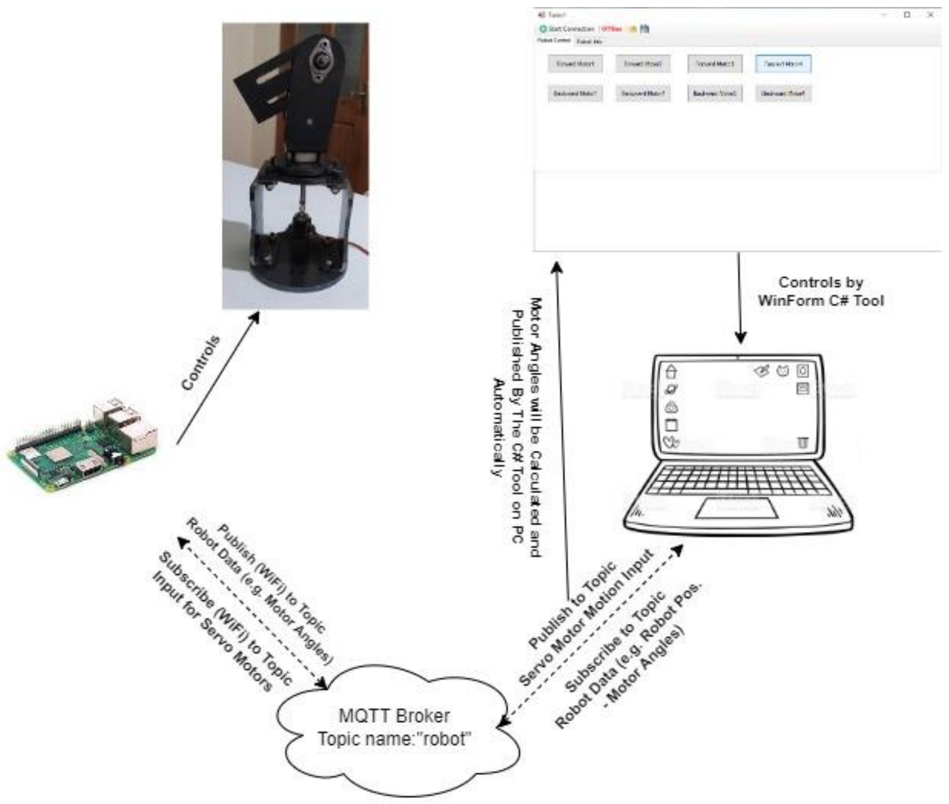

Controls
Controls by WinForm C# Tool
Motor Angles will be Calculated and Published By The C# Tool on PC Automatically
Publish (WiFi) to Topic Robot Data (e.g. Motor Angles)
Subscribe (WiFi) to Topic Input for Servo Motors
Publish to Topic Servo Motor Motion Input
Subscribe to Topic Robot Data (e.g. Robot Pos. - Motor Angles)
MQTT Broker
Topic name:"robot"

Figure 1.1 MQTT protocol overview

The findings of this study will develop robotic devices that can be controlled remotely and their intelligence and advantages. The MQTT protocol and Raspberry Pi, along with the compound mobile serial robot, have the potential to transform sectors like manufacturing, logistics, space sciences, bomb disposal areas. This thesis intends to inspire additional ideas and direct future research in the area of intelligent and remotely operated robotic systems by pushing the boundaries of robotics and utilizing the capabilities of the Raspberry Pi. The publication of the written thesis was presented at the 4. Baskent International Conference on Multidisciplinary Studies, and the abstract of the thesis is included in the published book[1].

1.1 Literature Review & Research and Analysis

1.1.1 Remote Control of Robot Arm with five DoF

Explains whole process of making a system for remote control of a robot arm with five DOF[2]. Serial RS-232 protocol is between PC and Microcontroller, and this communication is used to operate the arm. Uses TCP/IP protocol for remote control provides communication between server and client computers and sends information of position of robot arm. GUI is implemented in MATLAB for user interaction.

For every degree of freedom two pins of microcontroller and two relays are assigned (pins RD0 and RD1 for base, RD2 and RD3 for shoulder, RD4 and RD5 for elbow, RD6 and RD7 for wrist and RC0 and RC1 for fist).

Depending on the state of two pins, there are four situations:

- If both pins are low, two relays controlled by them are open and motor of the appropriate DOF is not running;
- if one pin is high and another is low, current flows in one direction and motor is running in appropriate direction;
- for opposite state of pins, motor is running in opposite direction;
- 'forbidden combination' is when both pins are high, because then both relays are active and source is short circuited.

A microcontroller were used to establish a communication with the server PC. The used communication is the serial communication RS-232. Serial communication is the most common low-level. To establish communication via ETHERNET, Real VNC program and MATLAB Server are used. Real VNC program is used to obtain visual feedback by camera, and MATLAB Server is used For transmission of control messages from client PC to server PC. The communication was established through MATLAB using two m-files. First m-file creates serial port and configures its properties. The communication between the server PC and the microcontroller is realized using second m-file. In this m-file, function was created to collect data set by user inside GUI. The user monitor movement of robot arm by camera. GUI is implemented in MATLAB as show below. TCP/UDP/IP toolbox is used for establish connection between server and client, VNC server also was used to transmit visual feedback.

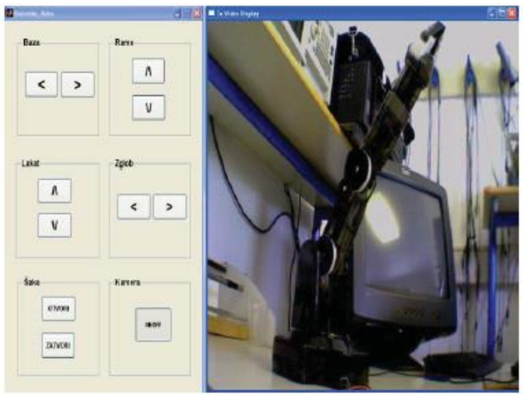


Figure 1.2 GUI for the researched project

1.1.2 Multi-sensor based glove control of industrial mobile robot arm

Performance and efficiency are more safe than an actual human performing the task especially in dangerous environments. The aim of this task is to ease an operation's complexity and hazardousness by only using a single hand to control a mobile robot with a 6-axis robotic arm[3]. Both mobile robot and robotic arm can be controlled wirelessly using a wearable data glove that is equipped with multiple sensors and a microcontroller.

The payload (robot arm) must be carefully placed at the top of the mobile robot in between its right and left wheels. The pressure sensitive sensor, flex bending sensor, inertial sensor and Arduino mini is used in the glove.

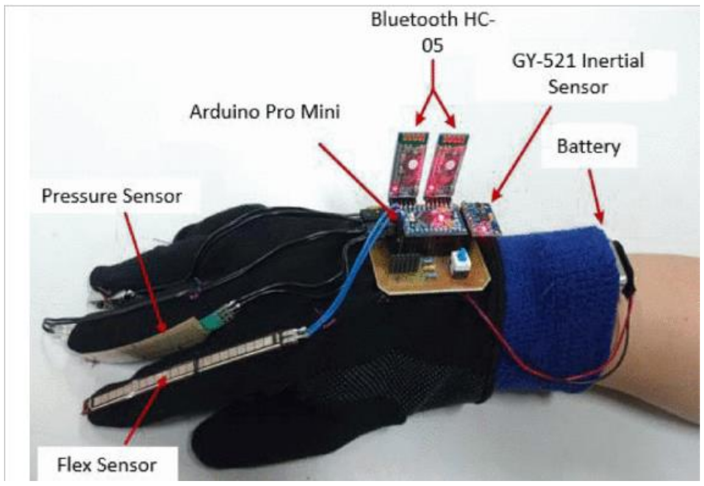


Figure 1.3 Hand controller of the robot

The mode selection is switched according to the hand action measured by the inertial sensor IMU, and the signals of the pressure and bending sensors on the data gloves respectively control various instructions action of the vehicle mode and the robot arm mode.

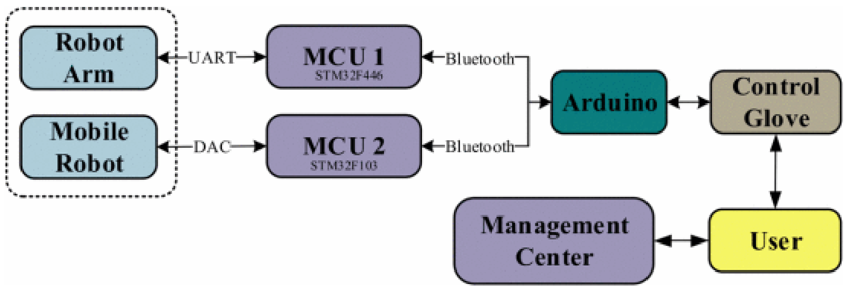


Figure 1.4 Overview of the article

1.1.3 Wireless Network for Mobile Robot Applications

The idea of a wireless network for information exchange between mobile robot nodes, which can be utilized for monitoring and control applications, is discussed in the article[4]. The primary goal is to decrease the amount of energy and computational power used by robot nodes. A central host computer that can be linked to a cloud network is outfitted with sensors and communication gear to collect data from each node and transfer it to it. A Wireless Sensor Network (WSN) is the term used to describe this system. In order to attain the desired efficiency, the study underlines the significance of employing appropriate communication protocols. The suggested method transfers data between networked nodes using the MQTT (MQ Telemetry Transport) protocol.

The article describes how communication is organized amongst the nodes and outlines how the system is verified through message exchange between the nodes and the central system. The main reason for deploying networked mobile robots is to handle difficult-for-people jobs that are complex and potentially dangerous, like air monitoring, radiation from nuclear power plant failures, and land pollution assessment.

In conclusion, the article concludes with a proposal for a wireless network for applications involving mobile robots, highlighting the usage of the MQTT protocol for effective data transfer between networked nodes. The objective is to use a grid of networked mobile robots to provide cost- and energy-efficient monitoring and control applications.

1.1.4 Finger Robotic control use M5Stack board and MQTT Protocol based

The study on using the MQTT protocol and the M5Stack board to remotely control a robotic hand's finger is presented in the paper. The goal of the project is to create remote control technology that will enable robotic fingers to perform various activities, like pressing buttons and adjusting volume. Servo angles are represented by values x and y or 1 and 2, and the MQTT protocol simplifies communication. For operating the robotic finger and using Python and MQTT Brokers to broadcast and subscribe to data, the study uses blockly programming. The WiFi-enabled M5Stack board acts as the platform for controlling the finger robot. Additionally, the study examines aspects like power usage, security, and data transmission, illustrating finger movement instances and the effects of interference on data transmission.

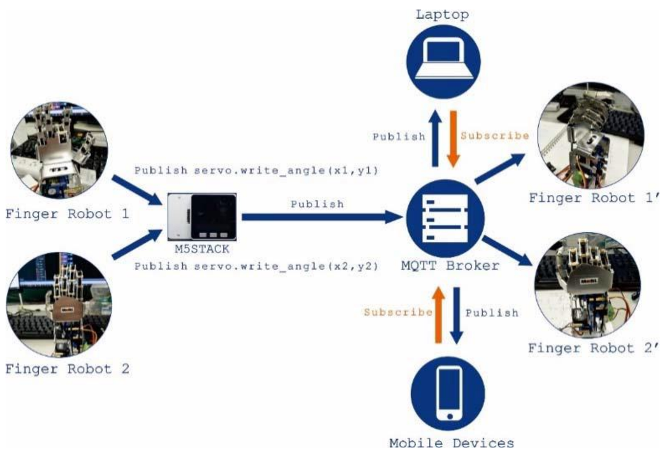


Figure 1.5 Architecture Network Design of this research

The use of inverse kinematics for robotic finger control and the MQTT protocol for communication are the two primary topics which are taken into account for this thesis to take as an reference of the paper. Here is a more thorough breakdown of these elements:

a. Inverse Kinematics for Robotic Finger Control:

To demonstrate finger motion, it proposes the planar two-link manipulator model. The foundation for calculating the joint angles (1 and 2) of the robotic finger is provided by the forward kinematics equations, which are represented by x and y coordinates. The study illustrates the relationship between these equations and finger movement in the planar space.

b. MQTT Protocol for Communication:

The research uses the MQTT (Message Queuing Telemetry Transport) protocol to make it easier for the robotic finger and distant gadgets to communicate. A well-liked IoT (Internet of Things) protocol called MQTT is well-known for its effectiveness and simplicity. Using MQTT Brokers, which serve as a middleman for data exchange between publishers (who transmit data) and subscribers (who receive data), is a part of it.

The M5Stack board, which has an ESP8266 WiFi module and is MQTT compatible, will be used in the study's configuration. The M5Stack board serves as the robotic finger's controller and talks with MQTT Brokers to transfer data.

1.2 Project Design Processes

a. Job Definition and Task of the Project
b. Conceptual Design
c. Kinematic Analysis
    i. Direct Analysis
    ii. Inverse Analysis
    iii. Jacobian Analysis
d. Dynamic Analysis
    i. Forward Analysis
    ii. Backward Analysis
    iii. Torque Analysis
e. Material, Hardware Selection and Integration
f. Software Development
g. Testing and Evaluation

1.3 Job Definition and Task of the Project

The robot has the benefit of being serial and mobile compound, in addition to allowing safe and effective use in numerous areas with remote control. This remote control can be assigned as scheduled tasks and perform certain tasks automatically thanks to the WinForm C# Tool features.

The robot will basically be based on mechanical or any type of part examination and analysis it remote or difficult terrains, with the integration of artificial intelligence and part recognition by the Raspberry Pi. The parts can be picked an place, explosive, dangerous or can be worked at any point that is closed to human access.

Main goal will be to pick a part from a certain point, define it, place and drop it to a different point automatically and handle all of these tasks remotely.

2 Design and Analysis of the Serial Robot

2.1 Conceptual Design

Robot design have been done on SolidWorks. The robot has the structure with 4 DoF. The kinematic structure of the serial arm is the same as one of the traditional industrial 4-DoF robots. All joints are revolute joint in the robot design. Table-1 shows DH table of the serial arm and Z axis means the rotation axis of joint.

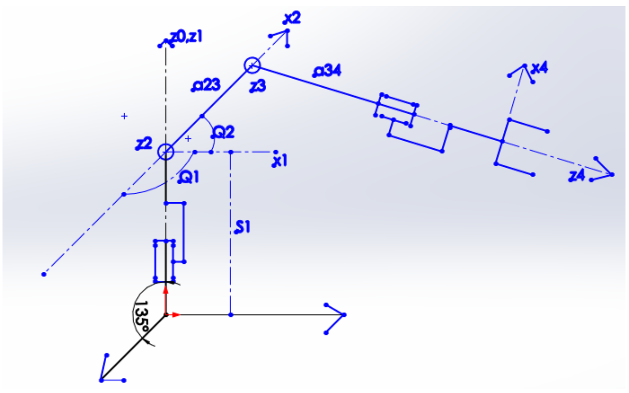


Figure 2.1 Conceptual Design DH Parameters

Table 1.1 DH Table

i	**$a_{i-1,i}$**	**$\alpha_{i-1,1}$**	**S_i**	**Q_i**
1	$a_{0,1}$	$\alpha_{0,1}$	S_1	Q_1
2	$a_{1,2}$	$\alpha_{1,2}$	S_2	Q_2
3	$a_{2,3}$	$\alpha_{2,3}$	S_3	Q_3
4	$a_{3,4}$	$\alpha_{3,4}$	S_4	Q_4

Length of link (a): It is determined as the distance measured between the mutual perpendiculars axis.

Torsion angle (α): It is the angle formed between the orthogonal. Projections of along the pivot axes in a plane perpendicular to the usual normal.

Joint. Offset (S): Length of connections of the normal perpendicular to the joint axis.

Joint. Angle (θ): The angle among the orthogonal. Projections which is normal perpendicular to the. Plane perpendicular to the pivot axes.

The parameters required for the table are defined by making measurements of the drawing.

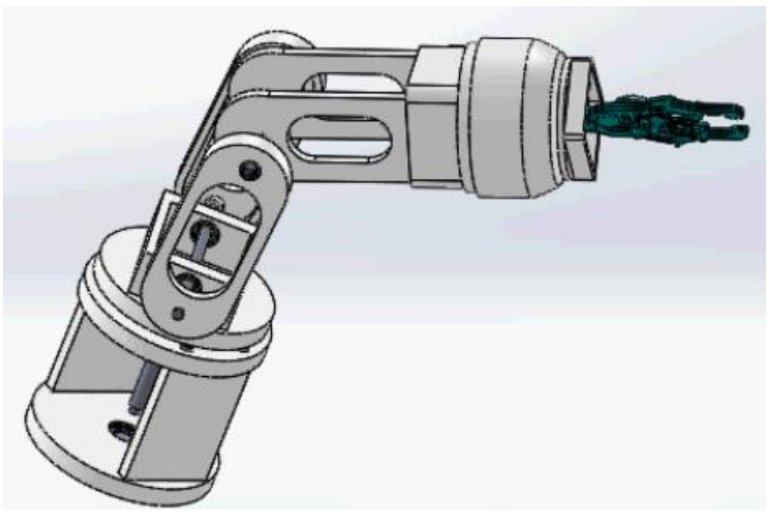


Figure 2.2 Detailed Mechanical Design of Robot in SolidWorks

In this final design of the serial arm robot there were some issues with the production. Linkages are restructured according to laser cutting necessities.

For laser cutting process, robot arm design is restructed in AutoCAD according to laser cutting production methods. This is shown below.

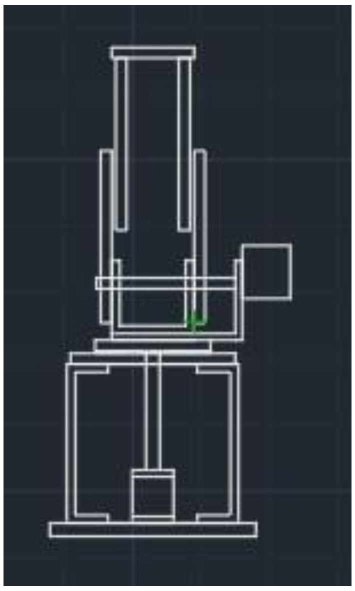


Figure 2.3 AutoCAD Overall View

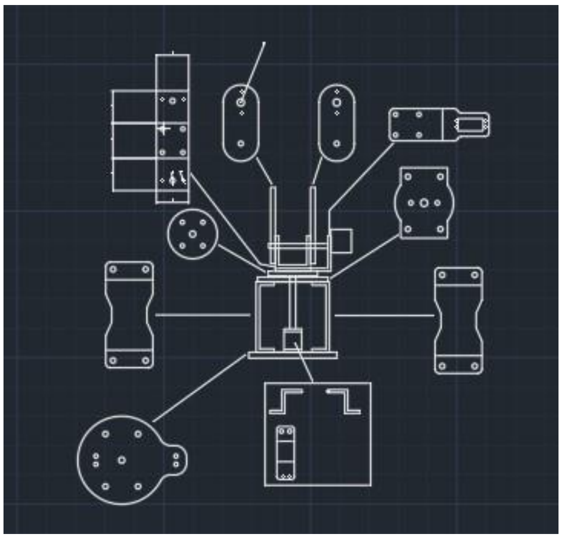


Figure 2.4 AutoCAD drawing by part

2.2 Kinematic Analysis

Robot kinematics is the study of the motion of robots. In a kinematic analysis, the position, velocity and acceleration of all connections are calculated regardless of the forces that cause this motion. Robot kinematics is about redundancy, collision avoidance and singularity avoidance. When dealing with the kinematics used in robots, a reference frame assigns each part of the robot, and so a serial arm robot can have many individual frames assigned to each moving part.

There are two separate problems to be solved in the kinematic analysis of the manipulator position: direct kinematics and inverse kinematics, which are presented in the sections in below.

2.2.1 Workspace Analysis with Matlab Simulation in Simulink

Matrices were defined in the Matlab. Matrix ${}^{0}T_{4}$ has been found according to the defined matrices. Then parametrical values of our matrices were entered. For finding workspace of the system Q angle values are assigned randomly with command of :

Q_i=(-6.28*rand(i))

Random angle values given in matlab as :

```
q1=(-6.28*rand(1))      q2=(6.28*rand(1))       q3=(-6.28*rand(1))      q4=0
```

Example of finding ${}^{i-1}T_{i}$ matrices in matlab is shown as:

```
% T01
t01transx = [1  0   0   a01; 0   1   0   0; 0 0   1   0; 0 0   0   1]
t01rotx = [1    0   0   0; 0 cos(a1) -sin(a1) 0; 0 sin(a1) cos(a1) 0; 0    0   0   1]
t01transz = [1  0   0   0; 0 1    0   0; 0 0   1   s1; 0    0   0   1]
t01rotz = [cos(q1)  -sin(q1) 0  0; sin(q1) cos(q1) 0 0 ;0   0   1   0;0 0 0 1]
T01=t01transx*t01rotx*t01transz*t01rotz
```

After that ${}^{0}T_{4}$ were found according to defined matrices as:

$$T_{04} = T_{01} * T_{12} * T_{23} * T_{34} \tag{2.1}$$

Finally plot command of position points were entered which are x, y and z. ‘’hold on’’ command was used since there is more than one value in the chart.

```
x=T04(1,4)         y=T04(2,4)         z=T04(3,4)         scatter3(x,y,z)      hold on
```

This command were executed many times and workspace is calculated.

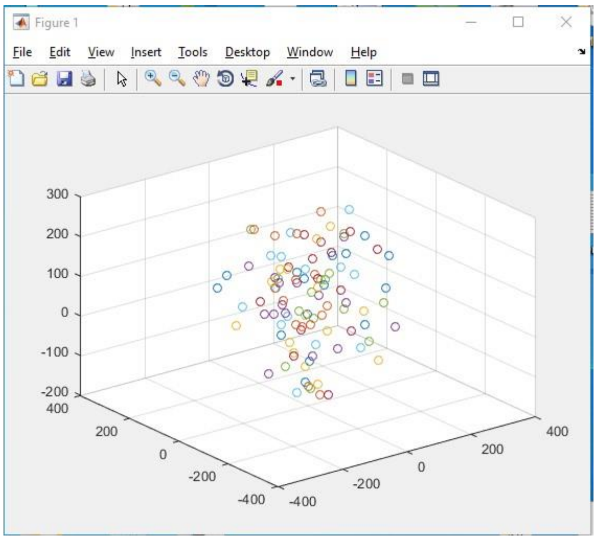


Figure 2.5 Workspace Analysis Result

The SolidWorks design was transferred to the matlab simulation for checking positions of the robot arm with respect to the changement of angle values of linkages. Simple sections of the simulation model of our robot arm are shown.

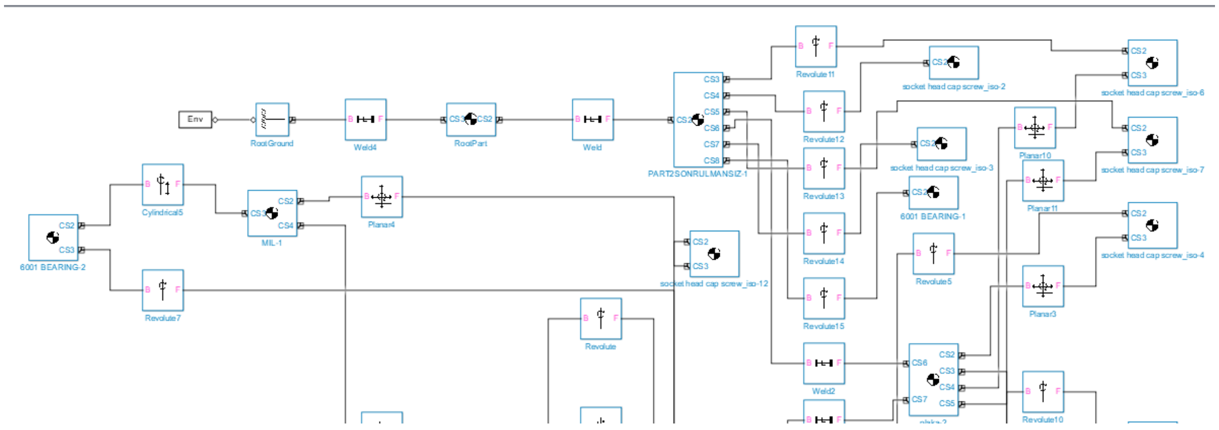


Figure 2.6 Section from the Matlab Simulation

After the simulation of the model has started robotic design can be seen in the Matlab. Then the control model were handled for linkage joint angels.

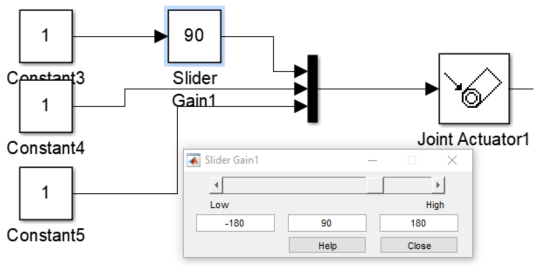


Figure 2.7 Slider Gain for Preparing Angle Value of the Link

Angle value of the linkage can be controlled by changing the slider gain.

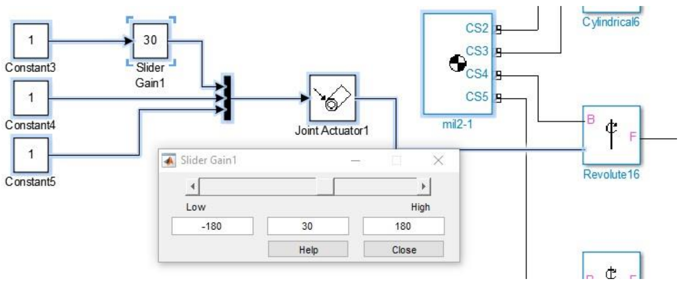


Figure 2.8 30 Degree Slider Gain For Preparing Angle Value of Link

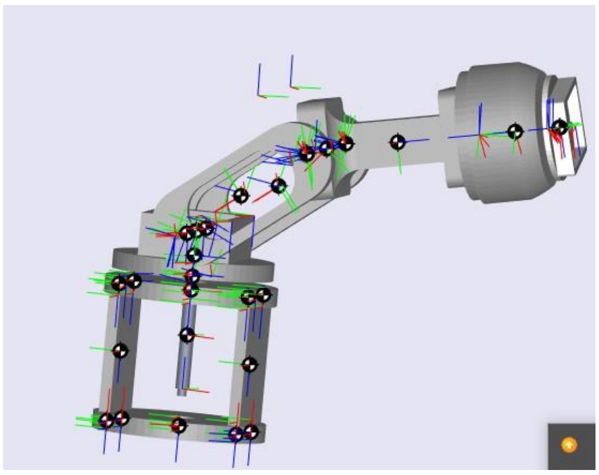


Figure 2.9 30 Degree Slider Gain of Robot Link

2.2.2 Forward Kinematics (Direct Task)

In forward kinematics, the length of each link and the angle of each joint is given and I have to calculate the position of any point in the work volume of the robot.

Direct kinematics involves solving the forward transformation equation to find the location of the hand in terms of the angles and displacements between the links.

Denavit-Hartenberg (DH) method uses the four parameters including $a_{i-1,i}$,$\alpha_{i-1,i}$, S_i and θ_i, which are the link length, link twist, link offset and joint angle, respectively. Transformation matrices will be used as a method for making our direct task. Transformation matrices are initially created as T_{tx} , T_{rx} , T_{tz} , T_{rz} . These transformation matrices should be created with their individual models. These models are shown below.

Transformation matrices of x axes:

$$T_{tx} = \begin{pmatrix} 1 & 0 & 0 & a_{i-1,i} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad T_{rx} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & Cos(\alpha_{i-1,i}) & -Sin(\alpha_{i-1,i}) & 0 \\ 0 & Sin(\alpha_{i-1,i}) & Cos(\alpha_{i-1,i}) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transformation matrices of z axes:

$$T_{tz} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & S_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad T_{rz} = \begin{pmatrix} Cos(Q_i) & -Sin(Q_i) & 0 & 0 \\ Sin(Q_i) & Cos(Q_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

For finding $^{i-1}T_i$ matrix:

$$^{i-1}T_i = {}^{i-1}T_{i\text{tx}}\, {}^{i-1}T_{i\text{rx}}\, {}^{i-1}T_{i\text{tz}}\, {}^{i-1}T_{i\text{rz}} \qquad (2.1)$$

First transformation matrices need to be defined, $^0T_1$ and then $^0T_1$matrix will be evaluated.

$$ {}^{0}T_{1\text{tx}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^{0}T_{1\text{rx}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{0}T_{1\text{tz}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 76 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad {}^{0}T_{1\text{rz}} = \begin{pmatrix} 0.76402128 & 0.64519103 & 0 & 0 \\ -0.6451910 & 0.76402128 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ \boldsymbol{{}^{0}T_{1}} \;=\; {}^{0}T_{1\text{tx}}\ {}^{0}T_{1\text{rx}}\ {}^{0}T_{1\text{tz}}\ {}^{0}T_{1\text{rz}} \text{ this matrix will be:} \qquad (2.2) $$

$$ {}^{0}T_{1} = \begin{pmatrix} 0.764021 & 0.645191 & 0 & 0 \\ -0.64519 & 0.764021 & 0 & 0 \\ 0 & 0 & 1 & 76 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Then $\boldsymbol{{}^{1}T_{2}}$ matrix will be evaluated by using Eq 2.1 with same procedures**.**

$$ {}^{1}T_{2\text{tx}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^{1}T_{2\text{rx}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{1}T_{2\text{tz}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad {}^{1}T_{2\text{rz}} = \begin{pmatrix} 0.47991728 & -0.8773137 & 0 & 0 \\ 0.8773137 & 0.47991728 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{1}T_{2} \;=\; {}^{1}T_{2\text{tx}} * {}^{1}T_{2\text{rx}} * {}^{1}T_{2\text{tz}} * {}^{1}T_{2\text{rz}} \text{ this matrix will be:} \qquad (2.3) $$

$$ {}^{1}T_{2} = \begin{pmatrix} 0.479917 & -0.87731 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0.877314 & 0.479917 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

After ${}^{2}\boldsymbol{T_3}$ matrix will be evaluated by using Eq 2.1 with same procedures**.**

$$ {}^{2}\boldsymbol{T_3}_{\text{tx}} = \begin{pmatrix} 1 & 0 & 0 & 137 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^{2}\boldsymbol{T_3}_{\text{rx}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{2}\boldsymbol{T_3}_{\text{tz}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^{2}\boldsymbol{T_3}_{\text{rz}}\text{m}= \begin{pmatrix} 0.0287939 & 0.9995853 & 0 & 0 \\ -0.9995853 & 0.0287939 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

${}^{2}\boldsymbol{T_3}$ **=** ${}^{2}\boldsymbol{T_3}_{\text{tx}} * {}^{2}\boldsymbol{T_3}_{\text{rx}} * {}^{2}\boldsymbol{T_3}_{\text{tz}} * {}^{2}\boldsymbol{T_3}_{\text{rz}}$ this matrix will be: (2.4)

$$ {}^{2}\boldsymbol{T_3} = \begin{pmatrix} 0.028794 & 0.999585 & 0 & 137 \\ -0.99959 & 0.028794 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

After that ${}^{3}\boldsymbol{T_4}$ matrix will be evaluated by using Eq 2.1 with same procedures**.**

$$ {}^{3}\boldsymbol{T_4}_{\text{tx}} = \begin{pmatrix} 1 & 0 & 0 & 100 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^{3}\boldsymbol{T_4}_{\text{rx}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{3}\boldsymbol{T_4}_{\text{tz}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^{3}\boldsymbol{T_4}_{\text{rz}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

${}^{3}\boldsymbol{T_4}$ = ${}^{3}\boldsymbol{T_4}_{\text{tx}} * {}^{3}\boldsymbol{T_4}_{\text{rx}} * {}^{3}\boldsymbol{T_4}_{\text{tz}} * {}^{3}\boldsymbol{T_4}_{\text{rz}}$ this matrix will be: (2.5)

$$ {}^{3}\boldsymbol{T_4} = \begin{pmatrix} 1 & 0 & 0 & 100 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Finally, the position values will be evaluated by using same equation with others.

$\boldsymbol{{}^{0}T_{4}}$ = ${}^{0}T_1 * {}^{1}T_2 * \boldsymbol{{}^{2}T_3 * {}^{3}T_4}$ this matrix will be:

(2.6)

$${}^{0}T_4 = \begin{pmatrix} 0.680566 & 0.645191 & 0.347215 & 118,29 \\ -0.57472 & 0.764021 & -0.29321 & -99.892 \\ -0.45446 & 0 & 0.890769 & 150.7463 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Evaluation of matrix is done that will give the position vectors of the end effector. This matrix called as final transformation matrix it is shown in Fig. 3

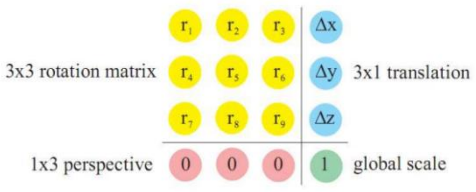


Figure 2.10 Model of transformation matrix

So the robot arm positions will be:

x = 118.29 mm   y = -99.89 mm   z =150.74mm

It was checked in the SolidWorks with initial design of the robot.

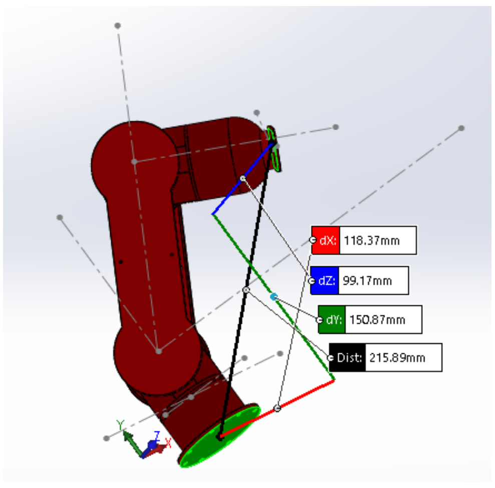
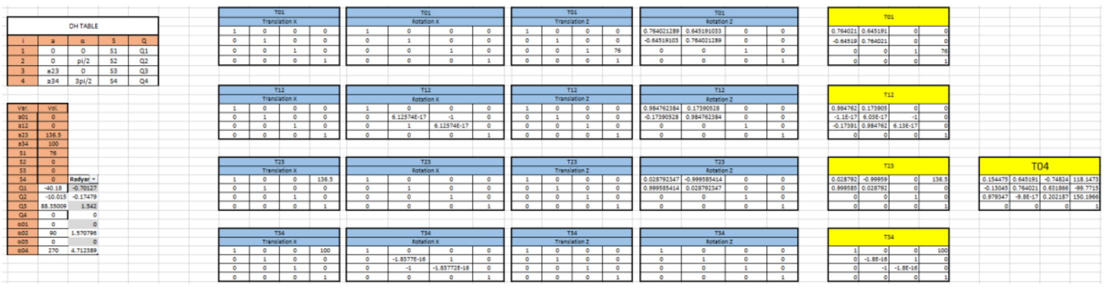


Figure 2.11 Position analysis in SolidWorks with conceptual design

When the desired position values, defined using SolidWorks measurements, were checked and the position values obtained through direct task evaluation using the transformation matrices method were evaluated, it was observed that the position values were identical. Thus, the direct task was successfully performed.

In summarize, the desired angles of the robotic arm must be entered into SolidWorks in order to determine position values. The SolidWorks position vectors should then be checked. The next step is to carry out the direct task using the transformation matrices approach, which calls for an understanding of the robotic arm's Denavit-Hartenberg (DH) characteristics. Comparison between the two sets of position values is necessary.

Microsoft Excel macro has created to perform those procedures.



DH TABLE				
i	a	α	S	Q
1	0	0	S1	Q1
2	0	pi/2	S2	Q2
3	a23	0	S3	Q3
4	a34	3pi/2	S4	Q4

Var.	Vol.	
a01	0	
a12	0	
a23	136.5	
a34	100	
S1	76	
S2	0	
S3	0	
S4	0	Radyan
Q1	-40.18	-0.70127
Q2	-10.015	-0.17479
Q3	88.35009	1.542
Q4	0	0
α01	0	0
α02	90	1.570796
α03	0	0
α04	270	4.712389

T01 Translation X			
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

T01 Rotation X			
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

T01 Translation Z			
1	0	0	0
0	1	0	0
0	0	1	76
0	0	0	1

T01 Rotation Z			
0.764021289	0.645191033	0	0
-0.64519103	0.764021289	0	0
0	0	1	0
0	0	0	1

T01			
0.764021	0.645191	0	0
-0.64519	0.764021	0	0
0	0	1	76
0	0	0	1

T12 Translation X			
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

T12 Rotation X			
1	0	0	0
0	6.12574E-17	-1	0
0	1	6.12574E-17	0
0	0	0	1

T12 Translation Z			
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

T12 Rotation Z			
0.984762384	0.17390528	0	0
-0.17390528	0.984762384	0	0
0	0	1	0
0	0	0	1

T12			
0.984762	0.173905	0	0
-1.1E-17	6.03E-17	-1	0
-0.17391	0.984762	6.13E-17	0
0	0	0	1

T23 Translation X			
1	0	0	136.5
0	1	0	0
0	0	1	0
0	0	0	1

T23 Rotation X			
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

T23 Translation Z			
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

T23 Rotation Z			
0.028792347	-0.999585414	0	0
0.999585414	0.028792347	0	0
0	0	1	0
0	0	0	1

T23			
0.028792	-0.99959	0	136.5
0.999585	0.028792	0	0
0	0	1	0
0	0	0	1

T34 Translation X			
1	0	0	100
0	1	0	0
0	0	1	0
0	0	0	1

T34 Rotation X			
1	0	0	0
0	-1.8377E-16	1	0
0	-1	-1.83772E-16	0
0	0	0	1

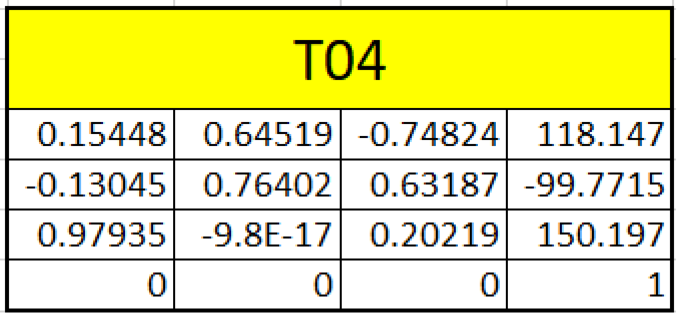
T34 Translation Z			
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

T34 Rotation Z			
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

T34			
1	0	0	100
0	-1.8E-16	1	0
0	-1	-1.8E-16	0
0	0	0	1

T04			
0.154475	0.645191	-0.74824	118.1473
-0.13045	0.764021	0.631866	-99.7715
0.979347	-9.8E-17	0.202187	150.1966
0	0	0	1

Figure 2.12 Transformation matrices method in Microsoft Excel



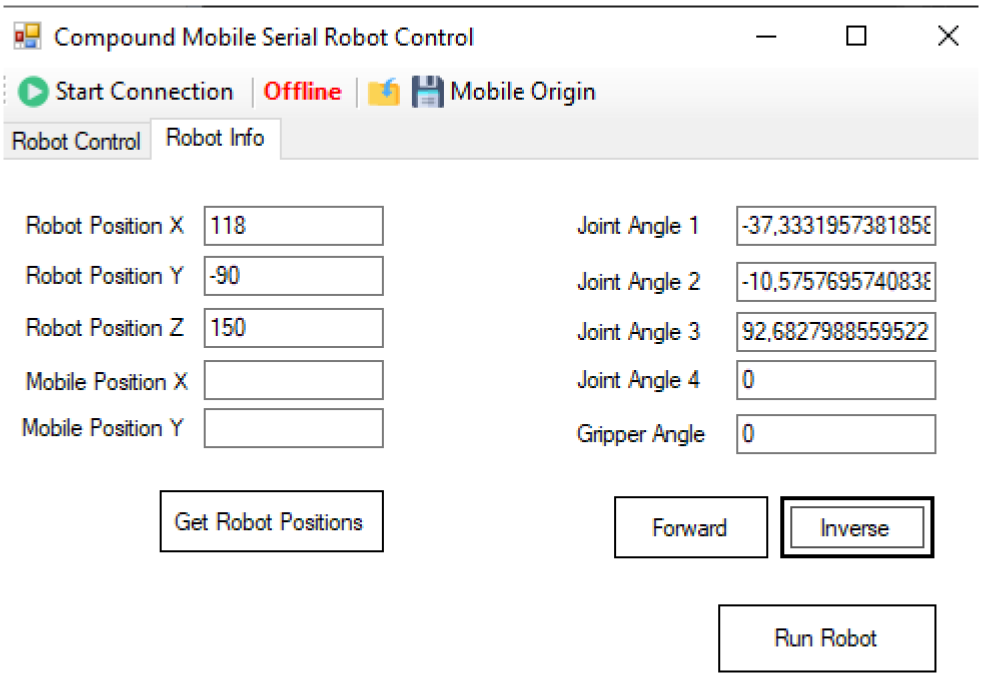
T04			
0.15448	0.64519	-0.74824	118.147
-0.13045	0.76402	0.63187	-99.7715
0.97935	-9.8E-17	0.20219	150.197
0	0	0	1

Figure 2.13 Transformation matrix of final position of the robotic arm

After all of those calculations are completed in the mathematic tool and Microsoft Excel, C# tool has been developed and forward kinematics has implemented into it. Thanks to the tool there is no need to define any other equation or use any external item to calculate robot position and control it with the inverse kinematics.

All matrices are defined in the tool, and each formulations for forward kinematics was followed with dynamic parameters. ForwardKinematics function were developed for this purpose, a part of the code snippet for matrix multiplication and matrix declaration and UI of the tool with forward kinematics can be seen below.

```
try
{
     int rowA = firstMatrix.GetLength(1);
     int columnA = firstMatrix.GetLength(0);
     int rowB = secondMatrix.GetLength(1);
     int columnB = secondMatrix.GetLength(0);
     double temp = 0;
     double[,] finalMatrix = new double[rowA, columnB];
     for (int i = 0; i < rowA; i++)
     {
          for (int j = 0; j < columnB; j++)
          {
               temp = 0;
               for (int k = 0; k < columnA; k++)
               {
                    temp += A[i, k] * B[k, j];
               }
               finalMatrix[i, j] = temp;
          }
      }

      return finalMatrix;
 }
 catch(Exception ex)
 {
     MessageBox.Show(ex.Message.ToString());
     double[,] nullMatrice = new double[0,1];
     return nullMatrice;

 }


double[,] translationXZeroOne = new double[4, 4];
translationXZeroOne[0, 0] = 1;
…
translationXZeroOne[3, 3] = 1;

double[,] rotationXZeroOne = translationXZeroOne;
double[,] translationZZeroOne = new double[4, 4];
translationZZeroOne[0, 0] = 1;
…
translationZZeroOne[3, 3] = 1;

double[,] rotationZZeroOne = new double[4, 4];
rotationZZeroOne[0, 0] = Math.Cos(Math.PI * firstAngle / 180.0);
…
rotationZZeroOne[3, 3] = 1;

double[,] firstResult = MultiplyMatrix(translationXZeroOne,
rotationXZeroOne);
double[,] secondResult = MultiplyMatrix(firstResult,
translationZZeroOne);
double[,] finalResultTZeroOne = MultiplyMatrix(secondResult,
rotationZZeroOne);
```


Compound Mobile Serial Robot Control

Start Connection | Offline | Mobile Origin

Robot Control | Robot Info

Robot Position X	118	Joint Angle 1	-37,333195738185ᵋ
Robot Position Y	-90	Joint Angle 2	-10,575769574083ᵋ
Robot Position Z	150	Joint Angle 3	92,6827988559522
Mobile Position X		Joint Angle 4	0
Mobile Position Y		Gripper Angle	0

Figure 2.14 Forward & Inverse Kinematics Calculation

2.2.3 Inverse Kinematics

Inverse kinematics is the opposite of forward kinematics. In inverse kinematics, the length of each link and position of the point in work volume is given and the angle of each joint has to be calculated.

Inverse kinematics involves solving the inverse transformation equation to find the relationships between the links of the manipulator from the location of the hand in space. This is when you have a desired end effector position, but need to know the joint angles required to achieve it the inverse position kinematics solves the following problem: end effector pose, what are the corresponding joint positions?" In contrast to the forward problem, the solution of the inverse problem is not always unique: the same end effector pose can be reached in several configurations, correspond position vectors.

Inverse kinematics is done in modern technical computing program. This program is used since it gives possibility to make matrix computing with parametric values.

Equations are started to use by defining the matrices of ${}^{0}T_1$ as :

$$ {}^{0}T_{1\ tx} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad {}^{0}T_{1\ rx} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{0}T_{1\ tz} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 76 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{0}T_{1\ rz} = \begin{pmatrix} Cos[Q1] & -Sin[Q1] & 0 & 0 \\ Sin[Q1] & Cos[Q1] & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{\mathbf{0}}\boldsymbol{T}_{\mathbf{1}} = {}^{0}T_{1\mathrm{tx}} * {}^{0}T_{1\mathrm{rx}} * {}^{0}T_{1\mathrm{tz}} * {}^{0}T_{1\mathrm{rz}} $$
(2.7)

$$ {}^{0}T_1 = \begin{pmatrix} Cos[Q1] & -Sin[Q1] & 0 & 0 \\ Sin[Q1] & Cos[Q1] & 0 & 0 \\ 0 & 0 & 1 & 76 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

As shown in above all of the rest matrices (${}^{0}T_1$, ${}^{1}T_2$, ${}^{\mathbf{2}}\boldsymbol{T}_{\mathbf{3}}$, ${}^{\mathbf{3}}\boldsymbol{T}_{\mathbf{4}}$) can be found with same formula. ${}^{\mathbf{0}}\boldsymbol{T}_{\mathbf{4}}$ should be found with below formula**.**

$$ {}^{\mathbf{0}}\boldsymbol{T}_{\mathbf{4}} = {}^{0}T_1\ {}^{1}T_2\ {}^{\mathbf{2}}\boldsymbol{T}_{\mathbf{3}}\ {}^{\mathbf{3}}\boldsymbol{T}_{\mathbf{4}} $$
(2.8)

Whenever all matrices are found, left side of the equation should be found by multiplying inverse of the ${}^{0}T_1$.

$T_{1\mathrm{left}}$ = Inverse[${}^{0}T_1$] ${}^{\mathbf{0}}\boldsymbol{T}_{\mathbf{4}}$ (2.9)

$$ \mathrm{Eq}_{left} = \begin{pmatrix} XXCos[Q1] + XYSin[Q1] & YXCos[Q1] + YYSin[Q1] & ZXCos[Q1] + ZYSin[Q1] & PXCos[Q1] + PYSin[Q1] \\ XYCos[Q1] - XXSin[Q1] & YYCos[Q1] - YXSin[Q1] & ZYCos[Q1] - ZXSin[Q1] & PYCos[Q1] - PXSin[Q1] \\ XZ & YZ & ZZ & -76 + PZ \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Then the right side of the equation should be calculated without multiplying matrix because it was multiplied by left side of the equation with inverse of this matrix. So, it will be simplified in right side if it is multiplied with inverse of it because when it was multiplied inverse matrix of and matrix, unit matrix will be calculated.

$$T_{1\text{right}} = {}^{1}T_{2}\ {}^{2}\boldsymbol{T_3}\ {}^{3}\boldsymbol{T_4}$$

(2.10)

$$\text{Eq}_{right} = \begin{pmatrix} Cos[Q2+Q3]\,Cos[Q4] & -Cos[Q2+Q3]\,Sin[Q4] & -Sin[Q2+Q3] & 137Cos[Q2]+100Cos[Q2+Q3] \\ Sin[Q4] & Cos[Q4] & 0 & 0 \\ Cos[Q4]\,Sin[Q2+Q3] & -Sin[Q2+Q3]\,Sin[Q4] & Cos[Q2+Q3] & 137Sin[Q2]+100Sin[Q2+Q3] \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

After both sides of the equation is calculated, parameters should be found and then Q1 can be easily calculated.

3rd column and 2nd row from Fig.10 (Simplified version of left side of our equation) and 3rd column and 2nd row again was taken as easiest equalities to start with.

This equation can be shown below.

$$PYCos[Q1] - PXSin[Q1] = 0 \qquad (2.11)$$

PY and PX values can be found from Microsoft Excel table because that table represents parametrical values of T_{04} matrix.

PY = -99   PX=118

Then Q1 can be calculated from this equation easily for solving this equation ‘Solve’ command in mathematica will be used as shown below.

$$Solve[-99Cos[Q1] - 118Sin[Q1] == 0, Q1] \qquad (2.12)$$

Q1 will be resulted in the mathematica as shown below.

$$Q1 = -ArcTan\left(\frac{99}{118}\right) = -0.70127 \qquad (2.13)$$

Q2 and Q3 values were calculated with 2 equations by 2 unknowns so below equations were selected.

$$-76 + 150.31 = (136.5 + 100\ Cos[Q3])Sin[Q2] + 100\ Cos[Q2]Sin[Q3] \quad (2.14)$$

$$-0.37227 = Cos[Q3]Sin[Q2] + Cos[Q2]Sin[Q3] \quad (2.15)$$

With the “Solve” command of the tool by using already founded Q1, Q2 and Q3 are calculated.

$$Q2 = 1.07024 \quad Q3 = -1.542$$

For calculating Q4, ${}^{1}T_{2}$ should be found with left and right equations, it was handled as same with above and Q4 was calculated as 0.

In addition to the ForwardKinematics function, InverseKinematics function has been also developed to handle inverse kinematics automatically and run servo motors accordingly. Desired robot positions can be entered to the textboxes of the tool where the end-effector should go, and joint angles will be calculated automatically, with the “Run Robot” button it can be controlled. A part of the code snippet can be seen below and UI can be checked from the Figure 2.14.

```
double firstAngleCalculation =
(double)Double.Parse(txtPosY.Text)/Double.Parse(txtPosX.Text);
double firstAngleCalculated = Math.Atan(firstAngleCalculation);

firstJoint.Text = ((180 / Math.PI) * firstAngleCalculated).ToString();

MathKernel mathKernel = new MathKernel();
var firstAngleToVar = firstAngleCalculated.ToString().Replace(',', '.');
var solveEquation = "Solve["+txtPosX.Text+"Cos["+ firstAngleToVar + "] +
"+ txtPosY.Text + "Sin["+ firstAngleToVar + "] == 137 Cos[Q2] + 100
Cos[Q2] Cos[Q3] - 100 Sin[Q2] Sin[Q3] && -76 + " + txtPosZ.Text+" ==
(136.5 + 100 Cos[Q3]) Sin[Q2] + 100 Cos[Q2] Sin[Q3] , {Q2 , Q3}]";
mathKernel.Compute(solveEquation);

string angleTwoPattern = @"Q2\s?\-\>\s?(.*?),";
string angleThreePattern = @"Q3\s?\-\>\s?(.*?)}";
string input = mathKernel.Result.ToString();
RegexOptions options = RegexOptions.Multiline;
var qTwoRadian = "";
var qTwoRadianCheck = "";
var counterCheck = 0;
var qThreeRadian = "";

foreach (Match m in Regex.Matches(input, angleTwoPattern, options))
{
qTwoRadianCheck = m.Groups[1].Value;
if (!qTwoRadianCheck.Contains("I"))
{
qTwoRadian = m.Groups[1].Value;
qThreeRadian = Regex.Matches(input, angleThreePattern,
options)[counterCheck].Groups[1].Value;
break;
}
counterCheck++;
```

```
}

double secondAnleCalculated = double.Parse(qTwoRadian,
CultureInfo.InvariantCulture);
double thirdAnleCalculated = double.Parse(qThreeRadian,
CultureInfo.InvariantCulture);

secondJoint.Text = ((180 / Math.PI) * secondAnleCalculated).ToString();
thirdJoint.Text = ((180 / Math.PI) * thirdAnleCalculated).ToString();

mathKernel.Dispose();
```

In summarize, both forward and inverse kinematics have been automatized and calculated in the C# WinForm tool which was developed for this thesis. It was investigated that those processes can be automatized and controlled, and combine with the MQTT protocol opportunities for controlling remotely within even different network and task automatization.

3 Jacobian and Dynamic Analysis of the Robot

3.1 Jacobian Analysis

It is used when linkage is complicated. The joint angles change to approach the goal position and orientation. Jacobian matrices are a super useful tool, and heavily used throughout robotics and control theory. Basically, a Jacobian defines the dynamic relationship between two different representations of a system. For example, if I have a 2-link robotic arm, there are two obvious ways to describe its current position: 1- the end-effector position and orientation which I will denote x, and 2- as the set of joint angles which I will denote q. The Jacobian for this system relates how movement of the elements of q causes movement of the elements of x. Jacobian can be thought as a transform matrix for velocity. Formally, a Jacobian is a set of partial differential equations:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}.\dot{\boldsymbol{q}} \tag{3.1}$$

where $\dot{x}$ and $\dot{q}$represent the time derivatives of x and q. This tells that the end-effector velocity is equal to the Jacobian, J, multiplied by the joint angle velocity.

3.1.1.1 Building the Jacobian

First, the relationship between the position of the end-effector and the robot’s joint angles should be defined. Distances are known from the shoulder to the elbow, and elbow to the wrist, as well as the joint angles, where the end-effector is relative to a base coordinate frame should be figured out. Those forward transformation matrices should be used.

That transformation matrices allow a given point to be transformed between different reference frames. In this case, the position of the end-effector relative to the second joint of the robot arm is known, but where it is relative to the base reference frame (the first joint reference frame in this case) is of interest. So, the rotation part of this matrix is straight-forward to define can be shown as.

$$^{0}\boldsymbol{R}_{i-1} = \begin{pmatrix} \mathrm{a} & \mathrm{b} & \mathrm{c} \\ \mathrm{d} & \mathrm{e} & \mathrm{f} \\ \mathrm{g} & \mathrm{h} & \mathrm{i} \end{pmatrix}$$

The translation part of the transformation matrices is a little different than before because reference frame 1 changes as a function of the angle of the previous joint's angles. From trigonometry, given a vector of length r and an angle q the x position of the end point is defined **r**.cos(q), and the y position is **r**.sin(q). And the z position of the end point is defined with offset of our robotic arm. It can be shown below.

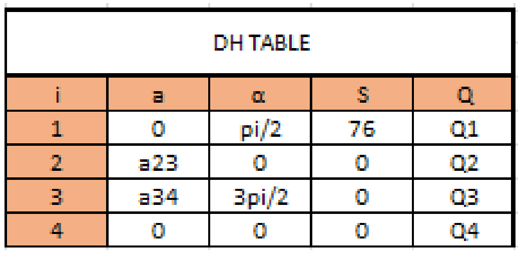
$$^{i-1}\boldsymbol{r}_i = \begin{pmatrix} a_{i-1,i}Cos(Qi) \\ a_{i-1,i}Sin(Qi) \\ S_i \end{pmatrix}$$

Then z_{i-1} is a unit vector along 'i'th joint axis, and $^{i-1}p_n{}^*$ is a vector defined from the origin of the (i-1)th link frame.

$$z_{i-1} = {}^{0}R_{i-1}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{3.2}$$

$$^{i-1}p_n{}^* = {}^{0}R_{i-1}\,{}^{i-1}r_i + {}^{i}p_n{}^* \tag{3.3}$$

Before these formulas are applied, a new Denavit-Hartenberg (DH) table should be created for finding same positions where it was found in direct task before.



DH TABLE				
i	a	α	S	Q
1	0	pi/2	76	Q1
2	a23	0	0	Q2
3	a34	3pi/2	0	Q3
4	0	0	0	Q4

Figure 3.1 DH table for Jacobian Analysis

$^{i-1}T_i = {}^{i-1}T_{i\text{tx}} * {}^{i-1}T_{i\text{rx}} * {}^{i-1}T_{i\text{tz}} * {}^{i-1}T_{i\text{rz}}$ (Eq. 1) formulation was used at this procedure but now with respect to the new Denavit-Hartenberg table.

${}^{i-1}T_i = {}^{i-1}T_{i\text{tz}} * {}^{i-1}T_{i\text{rz}} * {}^{i-1}T_{i\text{tx}} * {}^{i-1}T_{i\text{rx}}$ formulation should be used with respect to this formulation.

$\boldsymbol{{}^{0}T_{1}} = {}^{0}T_{1\text{tz}}\ {}^{0}T_{1\text{rz}}\ {}^{0}T_{1\text{tx}}\ {}^{0}T_{1\text{rx}}$ this matrix will be: (3.4)

$${}^{0}T_1 = \begin{pmatrix} \text{Cos(Q1)} & 0 & \text{Sin(Q1)} & 0 \\ \text{Sin(Q1)} & 0 & \text{-Cos(Q1)} & 0 \\ 0 & 1 & 0 & 76 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

${}^{1}T_{2} = {}^{1}T_{2\text{tz}} * {}^{1}T_{2\text{rz}} * {}^{1}T_{2\text{tx}} * {}^{1}T_{2\text{rx}}$ this matrix will be: (3.5)

$${}^{1}T_2 = \begin{pmatrix} \text{Cos(Q2)} & \text{-Sin(Q2)} & 0 & 137\ \text{Cos(Q2)} \\ \text{Sin(Q2)} & \text{Cos(Q2)} & 0 & 137\ \text{Sin(Q2)} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$\boldsymbol{{}^{2}T_{3}} = \boldsymbol{{}^{2}T_{3}}_{\text{tz}} * \boldsymbol{{}^{2}T_{3}}_{\text{rz}} * \boldsymbol{{}^{2}T_{3}}_{\text{tx}} * \boldsymbol{{}^{2}T_{3}}_{\text{rx}}$ this matrix will be: (3.6)

$$\boldsymbol{{}^{2}T_3} = \begin{pmatrix} \text{Cos(Q3)} & \text{-Sin(Q3)} & 0 & 100\ \text{Cos(Q3)} \\ \text{Sin(Q3)} & \text{Cos(Q3)} & 0 & 100\ \text{Sin(Q3)} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$\boldsymbol{{}^{3}T_{4}} = \boldsymbol{{}^{3}T_{4}}_{\text{tz}} * \boldsymbol{{}^{3}T_{4}}_{\text{rz}} * \boldsymbol{{}^{3}T_{4}}_{\text{tx}} * \boldsymbol{{}^{3}T_{4}}_{\text{rx}}$ this matrix will be: (3.7)

$$\boldsymbol{{}^{3}T_4} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$\boldsymbol{{}^0T_4}$ = ${}^0T_1\,{}^1T_2\,\boldsymbol{{}^2T_3}\,\boldsymbol{{}^3T_4}$ this matrix will be: (3.8)

$${}^0T_4=\begin{pmatrix} 0.680566 & 0.645191 & 0.347215 & 118.29 \\ -0.57472 & 0.764021 & -0.29321 & -99.892 \\ -0.45446 & 0 & 0.890769 & 150.7463 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

So, it can be easily seen that our final transformation matrix is the same which was found in direct task analysis. This shows that the new Denavit-Hartenberg table was created correctly, and proper equations was applied.

After finishing procedures of the new method of Denavit-Hartenberg solution ${}^{i-1}R_i$matrices can be found. That matrices are the rotation matrix part of ${}^0T_1\,{}^1T_2$ $\boldsymbol{{}^2T_3}$ and $\boldsymbol{{}^3T_4}$

$${}^0R_1=\begin{pmatrix} \text{Cos[Q1]} & 0 & \text{Sin[Q1]} \\ \text{Sin[Q1]} & 0 & \text{-Cos[Q1]} \\ 0 & 1 & 0 \end{pmatrix}$$

For ${}^0R_2$ and ${}^0R_3$ matrices ${}^0T_2$ and ${}^0T_3$ matrices should be found. Because ${}^0R_2$ and ${}^0R_3$ will be rotation matrices of ${}^0T_2$ and ${}^0T_3$ matrices. Those matrices can be found as same what was done for ${}^0T_4$. One example can be seen on below.

Rotation matrices will be found as shown on below.

$${}^0R_2=\begin{pmatrix} \text{Cos(Q1) Cos(Q2)} & \text{-Cos(Q1) Sin(Q2)} & \text{Sin(Q1)} \\ \text{Cos(Q2) Sin(Q1)} & \text{-Sin(Q1) Sin(Q2)} & \text{-Cos(Q1)} \\ \text{Sin(Q2)} & \text{Cos(Q2)} & 0 \end{pmatrix}$$

$${}^0R_3=\begin{pmatrix} \text{Cos(Q1) Cos(Q2 + Q3)} & \text{-Sin(Q1)} & \text{-Cos(Q1) Sin(Q2 + Q3)} \\ \text{Cos(Q2 + Q3) Sin(Q1)} & \text{Cos(Q1)} & \text{-Sin(Q1)Sin(Q2 + Q3)} \\ \text{Sin(Q2 + Q3)} & 0 & \text{Cos(Q2 + Q3)} \end{pmatrix}$$

${}^{i-1}r_i$ variables should be also defined.

$$\boldsymbol{r}_{01} = \begin{pmatrix} 0 \\ 0 \\ 76 \end{pmatrix} \boldsymbol{r}_{12} = \begin{pmatrix} 137Cos[Q2] \\ 137Sin[Q2] \\ 0 \end{pmatrix} \boldsymbol{r}_{12} = \begin{pmatrix} 100Cos[Q3] \\ 100Sin[Q3] \\ 0 \end{pmatrix} \boldsymbol{r}_{34} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

After finding those matrices below equation should be used.

$${}^{i-1}p_n^{\ *} = {}^{0}R_{i-1}\, {}^{\boldsymbol{i-1}}\boldsymbol{r_i} + {}^{i}p_n^{\ *} \tag{3.9}$$

There should be started with giving i=4 and n always equal to 4 so the first formulation.

$${}^{3}p_4^{\ *} = {}^{0}R_3\, {}^{3}r_4 + {}^{4}p_4^{\ *} \tag{3.10}$$

Here ${}^{4}p_4^{\ *} = 0$ ${}^{3}\boldsymbol{r}_4 = 0$ so it can be measured that ${}^{3}p_4^{\ *} = 0$.

After finding ${}^{3}p_4^{\ *}$ matrix it should be given as i = 3 and ${}^{2}p_3^{\ *}$ will be found.

$${}^{2}p_4^{\ *} = {}^{0}R_2\, {}^{2}\boldsymbol{r}_3 + {}^{3}p_4^{\ *} \tag{3.11}$$

Here ${}^{3}p_4^{\ *} = 0$ and ${}^{2}r_3$ and ${}^{0}R_2$ are also known as:

$${}^{0}R_2 = \begin{pmatrix} \text{Cos(Q1) Cos(Q2)} & \text{-Cos(Q1) Sin(Q2)} & \text{Sin(Q1)} \\ \text{Cos(Q2) Sin(Q1)} & \text{-Sin(Q1) Sin(Q2)} & \text{-Cos(Q1)} \\ \text{Sin(Q2)} & \text{Cos(Q2)} & 0 \end{pmatrix}$$

After multiplying ${}^{2}r_3$ and ${}^{0}R_3$, ${}^{2}p_4^{\ *}$ can be found as:

$${}^{2}p_4^{\ *} = \begin{pmatrix} 100\ Cos[Q1]\ Cos[Q2+Q3] \\ 100\ Cos[Q2+Q3]\ Sin[Q1] \\ 100\ Sin[Q2+Q3] \end{pmatrix}$$

${}^{i-1}p_n{}^* = {}^0R_{i-1}\,{}^{i-1}r_i + {}^ip_n{}^*$ formulation will be applied for finding ${}^1p_4{}^*$ by giving i=2:

$${}^1p_4{}^* = {}^0R_1\,{}^1r_2 + {}^2p_4{}^* \tag{3.12}$$

$${}^1p_4{}^* = \begin{pmatrix} Cos[Q1](137\ Cos[Q2] + 100\ Cos[Q2+Q3]) \\ (137\ Cos[Q2] + 100\ Cos[Q2+Q3])\ Sin[Q1] \\ 137\ Sin\ [Q2] + 100\ Sin[Q2+Q3] \end{pmatrix}$$

With respect to the formulation if i=1 is given ${}^1p_4{}^* = {}^0p_4{}^*$ So ${}^0p_4{}^*$ will be as same.

After finding all ${}^{i-1}p_n{}^*$ values as a vector defined from the origin of the (i-1)th link frame z_{i-1} should be found which is a unit vector along 'i'th joint axis.

$$Z_0 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$Z_1 = R01.\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} Sin[Q1] \\ -Cos[Q1] \\ 0 \end{pmatrix} \tag{3.13}$$

$$Z_2 = R02.\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} Sin[Q1] \\ -Cos[Q1] \\ 0 \end{pmatrix} \tag{3.14}$$

$$Z_3 = R03.\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -Cos[Q1]\ Sin[Q2+Q3] \\ -Sin[Q1]\ Sin[Q2+Q3] \\ Cos[Q2+Q3] \end{pmatrix} \tag{3.15}$$

All these desired matrices for building the final version of Jacobian matrix is defined and calculated. Those needs to be built up as shown below:

$J = [J_1,J_2,J_3,J_4]$

Those J_1,J_2,J_3,J_4 matrices can be written individually by using the below model.

$$J_i = \begin{pmatrix} Z_{i-1} x^{i-1} P_n^* \\ Z_{i-1} \end{pmatrix} \text{ for revolute joint}$$

Cross products will be handled and Jacobian matrices will be found as shown below.

$$J_1 = \begin{pmatrix} -(137\,\mathrm{Cos}(Q2) + 100\,\mathrm{Cos}(Q2+Q3))\,\mathrm{Sin}(Q1) \\ \mathrm{Cos}(Q1)\,(137\,\mathrm{Cos}(Q2) + 100\,\mathrm{Cos}(Q2+Q3)) \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$J_2 = \begin{pmatrix} -\mathrm{Cos}(Q1)\,(137\,\mathrm{Sin}(Q2) + 100\,\mathrm{Sin}(Q2+Q3)) \\ -\mathrm{Sin}(Q1)\,(137\,\mathrm{Sin}(Q2) + 100\,\mathrm{Sin}(Q2+Q3)) \\ 137\,\mathrm{Cos}(Q2) + 100\,\mathrm{Cos}(Q2+Q3) \\ \mathrm{Sin}(Q1) \\ -\mathrm{Cos}(Q1) \\ 0 \end{pmatrix}$$

$$J_3 = \begin{pmatrix} -100\,\mathrm{Cos}(Q1)\,\mathrm{Sin}(Q2+Q3) \\ -100\,\mathrm{Sin}(Q1)\,\mathrm{Sin}(Q2+Q3) \\ 100\,\mathrm{Cos}(Q2+Q3) \\ \mathrm{Sin}(Q1) \\ -\mathrm{Cos}(Q1) \\ 0 \end{pmatrix}$$

$$J_4 = \begin{pmatrix} -(137\,\mathrm{Cos}(Q2) + 100\,\mathrm{Cos}(Q2+Q3))\,\mathrm{Sin}(Q1) \\ \mathrm{Cos}(Q1)\,(137\,\mathrm{Cos}(Q2) + 100\,\mathrm{Cos}(Q2+Q3)) \\ 0 \\ -\mathrm{Cos}\,(Q1)\,\mathrm{Sin}(Q2+Q3) \\ -\mathrm{Sin}(Q1)\,\mathrm{Sin}(Q2+Q3) \\ \mathrm{Cos}(Q2+Q3) \end{pmatrix}$$

3.2 Dynamic Analysis

In a dynamic model of a system there are two main aspects with which one is concerned: motion and forces. The motion of a system is called its trajectory and consists of a sequence of desired positions, velocities, and accelerations of some point or points in the system. Forces are usually characterized as internal (or constraint) forces and external (or applied) forces. The external forces are the ones which cause motion. In robotics, a dynamic robot model usually describes relationships between robot motion and forces causing that motion, so that given one of these quantities, other one can be determined.

There are, therefore, the following problems to be considered forward Dynamics and inverse Dynamics. In principle, solving forward or inverse dynamics for rigid-link robot manipulators presents no difficulty.

A robot manipulator is just a system of rigid bodies, and the equations of motion of such systems have been known for a long time. The real problem in robot dynamics is a practical one, namely, that of finding formulations for the equations of motion that lead to efficient computational algorithms. To derive these equations, I can use well established procedures from classical mechanics such as those based on the equations of Newton and Euler, Euler and Lagrange, Kane, etc.

The Newton and Euler method will be used to solve dynamic analysis [5].

3.2.1 Forward Dynamic Analysis

The Forward or direct dynamics problem is one where the forces which act on a robot are given and the resulting motion will be solved. The importance of forward dynamics in robotics stems mainly from its use in simulation. Simulation of robot motion is a way of testing control strategies or manipulator designs prior to the expensive task of working with the actual manipulator.

First, I started to compute angular velocity, angular acceleration, linear velocity, and linear acceleration of each link in terms of its preceding link. These velocities can be computed as starting at the first moving link and ending at end-effector link.

a) Angular Velocity Propagation

Due to serial construction of the manipulator, the angular velocity of link i relative to link i-1 is equal to $z_{i-1}\acute{Q_i}$ for revolute joint, where z_{i-1} denotes a unit vector pointing along the ith joint axes. Angular velocity of i link can be written as

$$\omega_i = \omega_{i-1} + z_{i-1}\acute{Q_i} \tag{3.16}$$

$$z_0 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$R_{01} = \begin{pmatrix} Cos[Q1] & 0 & Sin[Q1] \\ Sin[Q1] & 0 & -Cos[Q1] \\ 0 & 1 & 0 \end{pmatrix}$$

$$z_1 = R_{01}\begin{pmatrix}0\\0\\1\end{pmatrix} \quad (3.17)$$

After finding all of the Z_{i-1} matrices, below formula should be applied.

$$\omega_{\mathrm{i}} = Z_{\mathrm{i}-1}\, Qid \quad (3.18)$$

This will be applied from i=0 to i=4, after all of those are found it should be expressed in the i_{th} link frame with the below formula.

$${}^{i}w_i = {}^{i}R_{i-1}\,({}^{i-1}w_{i-1} + {}^{i-1}z_{i-1}\,\acute{Q}_i) \quad (3.19)$$

First ${}^{i}R_{i-1}$ should be defined Then ${}^{i-1}z_{i-1}$ will be called as Z_{nn}. Angular velocities of i_{th} link frame will found as:

$$R34n = \begin{pmatrix} Cos[Q4] & Sin[Q4] & 0\\ -Sin[Q4] & Cos[Q4] & 0\\ 0 & 0 & 1\end{pmatrix}$$

$$Znn = \begin{pmatrix}0\\0\\1\end{pmatrix}$$

$$\omega 11 = R01n.(Znn\ Qd1)$$

$$\omega 22 = R12n.(\omega 11 + Znn\ Qd2)$$

b) Angular Acceleration Propagation

That will be link i is obtained by using the below equation.

$$\dot{\omega}_i = \dot{\omega}_{i-1} + z_{i-1}\ddot{Q}_i + \omega_{i-1} \times z_{i-1}\acute{Q}_i \quad (3.20)$$

$\dot{\omega}_i$ will be defined as Wd1 in mathematic tool. Below will be calculated.

$$\omega d1 = Z0Qdd1$$

$$Z1Qd2 = Z1Qd2$$

$$\omega d2 = \omega d1 + Z1.Qdd2 + Cross\ [\omega d1, Z1Qd2]$$

Then this should be expressed in the i_{th} link frame.

$$\dot{\omega}_i = {}^{i}R_{i-1}\,({}^{i-1}\dot{\omega}_{i-1} + z_{i-1}\ddot{Q}_i + {}^{i-1}\omega_{i-1}\ \mathrm{X}\ {}^{i-1}z_{i-1}\acute{Q}_i) \quad (3.21)$$

$\omega d11 = R01n.(ZnnQdd1)$

$\omega d22 \; = \; R12n.(\omega d11 + Znn\ Qdd2 + Cross\ [\omega 11, Znn\ Qd2])$

ωd11, ωd22, ωd33 and ωd44 angular acceleration were found as same as above.

c) Linear Velocity Propagation

It needs to be considered as if the i$_{th}$ joint is a revolute joint, link i does not translate along the i$_{th}$ joint axis. Then the velocity can be written as:

$$V_i = V_{i-1} + \omega_i\, x\, r_i \tag{3.22}$$

First, r_i should be defined to for the Eq 3.22:

$$r2 = \begin{pmatrix} a12\ Cos[Q2] \\ a12\ Sin[Q2] \\ 0 \end{pmatrix} \qquad r3 = \begin{pmatrix} a23\ Cos[Q3] \\ a23\ Sin[Q3] \\ 0 \end{pmatrix}$$

Then linear velocity can be found by using Eq. 3.22 which can be shown as below:

$V1 \; = \; Cross[\omega 1, r1]$

V2 = V1 + Cross [ω2, r2]   same formula for the V3 and V4.

Then those again need to be expressed in the i$_{th}$ link frame as it was done before for angular velocity and angular acceleration. This expression will be shown as:

$${}^{i}V_{i} = {}^{i}R_{i-1}\,({}^{i-1}V_{i-1} + {}^{i}\omega_i\ x\ {}^{i}r_i) \tag{3.23}$$

${}^{i}r_i$ is defined as r$_{ii}$ in mathematic tool for finding r$_{ii}$ constant vector for a revolute joint. These will be found as:

$${}^{i}r_i = \begin{pmatrix} a_i \\ S_i Sin\alpha_i \\ S_i Cos\alpha_i \end{pmatrix} \tag{3.24}$$

Each of them was found parametrically and Eq 3.23 was applied for linear velocity i$_{th}$ link.

$$r11 = \begin{pmatrix} 0 \\ S1 \\ 0 \end{pmatrix} \quad r11 = \begin{pmatrix} a12 \\ 0 \\ 0 \end{pmatrix} \quad r33 = \begin{pmatrix} a23 \\ 0 \\ 0 \end{pmatrix} \quad r44 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$V11 = Cross\,[\omega 11, r11]$

$V22 = R12n.V11 + Cross[\omega 22, r22]$ same formula for the V3 and V4.

d) Linear Acceleration Propagation

Linear acceleration of the frame i can be obtained by differentiating Eq3.24 with respect to time. It will be shown as:

$\dot{V}_i = \dot{V}_{i-1} + \dot{\omega}_i \, x \quad r_i + \quad \omega_i \, x \,(\quad \omega_i \, x \quad r_i)$

(3.25)

$\omega 2r2 = Cross[\omega 2, r2]$

$Vd2 = Cross[\omega d2, r2] + Cross[\omega 2, \omega 2r2]$

$\omega 3r3 = Cross\,[\omega 3, r3]$

$Vd3 = Vd2 + Cross[\omega d3, r3] + Cross[\omega 3, \omega 3r3]$

$\omega 3r3 = Cross[\omega 4, r4]$

Vd4 will be equal to Vd3 since ω3r3 was found as 0.

These need to expressed ith link frame as same before for the angular velocity and angular acceleration. This expression will be shown as:

$${}^{i}\dot{V}_i = {}^{i}R_{i-1}\, {}^{i-1}\dot{V}_{i-1} + {}^{i}\dot{\omega}_i \, x \;\; {}^{i}r_i + {}^{i}\omega_i \, x \,({}^{i}\omega_i \, x \;\; {}^{i}r_i) \qquad (3.26)$$

$Vd22 = Cross[\omega d22, r22] + Cross[\omega 22, \omega 22r22]$

$\omega 3r33 = Cross[\omega 33, r33]$

$Vd33 = R23n.Vd22 + Cross[\omega d33, r33] + Cross[\omega 33, \omega 33r33]$

e) Linear Acceleration of the Center of Mass

$${}^{i}\dot{V}_{ci} = {}^{i}\dot{V}_i + {}^{i}\dot{\omega}_i x \;\; {}^{i}r_{ci} + {}^{i}\omega_i \, x \,({}^{i}\omega_i \, x \;\; {}^{i}r_{ci}) \qquad (3.27)$$

First, ${}^{i}r_{ci}$, position vector of the center of mass of the link with i link frame is shown:

$${}^{i}r_{ci} = -\boldsymbol{a_i/2}\begin{pmatrix}\boldsymbol{CosQ_i}\\ \boldsymbol{SinQ_i}\\ \boldsymbol{0}\end{pmatrix}$$

Then, Eq 3.27 will be applied as shown below.

$$\omega 22c22 = Cross[\omega 22, rc22]$$

$$Vdc22 = Cross[\omega d22, rc22] + Cross[\omega 22, \omega 22rc22]$$

$$W33r33 = Cross[\omega 33, rc33]$$

$$Vdc33 = Vdc22 + Cross[\omega d33, rc33] + Cross[\omega 33, \omega 33rc33]$$

Since the ω44rc44 equals to zero, Vdc44 will be as same with Vdc33.

f) Acceleration of the Gravity

As a final, the acceleration of gravity is transformed from the (i-1) link frame to the ith in frame as:

$${}^{i}g = {}^{i}R_{i-1}\,{}^{i-1}g \tag{3.28}$$

$$g1 = R01n.\begin{pmatrix}0\\0\\g\end{pmatrix}$$

$$g2 = R12n.g1$$

g3 and g4 will be found with the same formula as above.

3.2.2 Backward Dynamic Analysis

When the velocities and accelerations of the links are found, the joint forces can be computed at a time starting from the end-effector link and ending at the base link.

First inertia force exerted at the center of mass link i should be computed as:

$${}^{i}f_i^{*} = m_i + {}^{i}\dot{V}_{ci} \tag{3.29}$$

$fs11 = -m1Vdc11$

$fs22 = -m2dc22$

Inertia forces of the fs11, fs22, fs33 and fs44 will be found as same shown above. After finding inertia forces of each i link. System should be solved recursively, starting from the end-effector link. For the end-effector link, represent the end-effector output force. This output force is considered and defined as below:

$$f045 = \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix}$$

After defining the output force, recursive function equation should be defined.

$${}^{i}f_{i,i-1} = {}^{i}f_{i+1,i} - m_i \; {}^{i}g - {}^{i}f_i^* \tag{3.30}$$

When the reaction forces are computed in the ith link frame, these are converted into the (i-1)th link by following transformations:

$${}^{i-1}f_{i,i-1} = {}^{i-1}R_i \; {}^{i}f_{i,i-1} \tag{3.31}$$

As a result of the definition of the all these computing steps it should be started by finding an external output force of end-effector as:

$f445 = R04.f045$

$f454 = -f445$

Finally, all of those should be calculated recursively, finding joint forces can be shown as:

$m4g4 = m4\,g4$

$f443 = f454 - m4g4 - fs44$

$f343 = R34.f443$

Same formulas should be applied until the f110 joint force, and then, as same processes should be implemented for the inertia moment exerted at the center of the mass of the link i:

$${}^{i}n_i^* = -{}^{i}I_i\ {}^{i}\dot{\omega}_i - {}^{i}\omega_i x\ ({}^{i}I_i\ {}^{i}\omega_i) \tag{3.32}$$

For finding inertia moment of the system inertia matrix of link i about its center of mass coordinate frame should be defined as:

$${}^{i}I_i = m_i a_i^2/12 \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.33}$$

By using the Eq. 3.22 inertia moments of center of mass of link i can be calculated.

$ns11 = -Cross[\omega 11, I11\omega 11]$

$I22\omega 22 = I22.\omega 22$

$I22\omega d22 = I22.\omega d22$

$ns22 = -I22\,.\omega d22 - Cross[\omega 22, I22\omega 22]$

$I33\omega 33 = I33.\omega 33$

The rest of the inertia moments will be found parametrically as same on above.

After finding the inertia moment of each i link. System should be resolved recursively, starting from the end-effector link. For the end-effector link, ${}^{i}n_{i+1,i}$ represent the end-effector output force. This output moment is considered as 0.

$${}^{i}n_{i,i-1} = {}^{i}n_{i+1,i} + ({}^{i}r_i + {}^{i}r_{ci})\ x\ {}^{i}f_{i,i-1} - {}^{i}r_{ci}\ x\ {}^{i}f_{i+1,i} - {}^{i}n_i^* \tag{3.34}$$

When the reaction moments are computed in the ith link frame, these are converted into the (i-1)th link by following transformations:

$${}^{i-1}n_{i,i-1} = {}^{i-1}R_i\ {}^{i}n_{i,i-1} \tag{3.35}$$

Finally, it can easily computed by keeping going these procedures.

$n443 = Cross[r44rc44, f443] - Cross[rc44, f454] - ns44$

$n343 = R34.n443$

$n332 = n343 + Cross[r33rc33, f332] - Cross[rc33, f343] - ns33$

Joint moments have been calculated as same with the formulas and followings shown in above until the n010.

3.2.3 Determination of the Torques of the Motors

Actuator torques or forces T_i, are obtained by projecting the forces of constraint onto their corresponding joint axes, that can be shown as:

$$T_i = {}^{i-1}n_{i,i-1}^{T}\ {}^{i-1}z_{i-1} \tag{3.36}$$

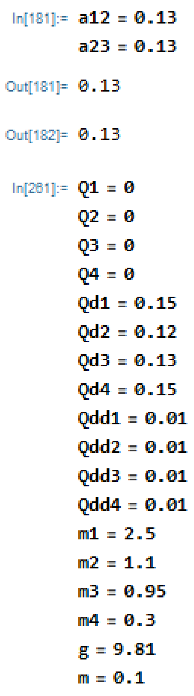
First torque values should be calculated parametrically. Then unknown values can be entered by using the design values.

$\mathrm{T1} = \{-\mathrm{Sin}[\mathrm{Q2}]\mathrm{Sin}[\mathrm{Q3}](-\frac{1}{4}\mathrm{a12a23m3Qdd2} - \frac{1}{2}\mathrm{a12a23m4Qdd2} -$
$\frac{1}{4}\mathrm{a23}^2\mathrm{m3Qdd1Cos}[\mathrm{Q2}]\mathrm{Cos}[\mathrm{Q3}] - \frac{1}{2}\mathrm{a23}^2\mathrm{m4Qdd1Cos}[\mathrm{Q2}]\mathrm{Cos}[\mathrm{Q3}] -$
$\frac{1}{4}\mathrm{a12a23m3Qd1}^2\mathrm{Cos}[\mathrm{Q2}]\mathrm{Sin}[\mathrm{Q2}] - \frac{1}{2}\mathrm{a12a23m4Qd1}^2\mathrm{Cos}[\mathrm{Q2}]\mathrm{Sin}[\mathrm{Q2}] +$
$\frac{5}{12}\mathrm{a23}^2\mathrm{m3Qd1Qd2Cos}[\mathrm{Q3}]\mathrm{Sin}[\mathrm{Q2}] + \mathrm{a23}^2\mathrm{m4Qd1Qd2Cos}[\mathrm{Q3}]\mathrm{Sin}[\mathrm{Q2}] +$
$\frac{5}{12}\mathrm{a23}^2\mathrm{m3Qd1Qd3Cos}[\mathrm{Q3}]\mathrm{Sin}[\mathrm{Q2}] + \mathrm{a23}^2\mathrm{m4Qd1Qd3Cos}[\mathrm{Q3}]\mathrm{Sin}[\mathrm{Q2}] +$
$\frac{5}{12}\mathrm{a23}^2\mathrm{m3Qd1Qd2Cos}[\mathrm{Q2}]\mathrm{Sin}[\mathrm{Q3}] + \mathrm{a23}^2\mathrm{m4Qd1Qd2Cos}[\mathrm{Q2}]\mathrm{Sin}[\mathrm{Q3}] +$
$\frac{5}{12}\mathrm{a23}^2\mathrm{m3Qd1Qd3Cos}[\mathrm{Q2}]\mathrm{Sin}[\mathrm{Q3}] + \mathrm{a23}^2\mathrm{m4Qd1Qd3Cos}[\mathrm{Q2}]\mathrm{Sin}[\mathrm{Q3}] +$
$\frac{1}{4}\mathrm{a23}^2\mathrm{m3Qdd1Sin}[\mathrm{Q2}]\mathrm{Sin}[\mathrm{Q3}] + \frac{1}{2}\mathrm{a23}^2\mathrm{m4Qdd1Sin}[\mathrm{Q2}]\mathrm{Sin}[\mathrm{Q3}] +$
$\frac{1}{12}\mathrm{a23}^2\mathrm{m3}(\mathrm{Cos}[\mathrm{Q3}](\mathrm{Qdd1Cos}[\mathrm{Q2}] - \mathrm{Qd1Qd2Sin}[\mathrm{Q2}] - \mathrm{Qd1Qd3Sin}[\mathrm{Q2}]) -$
$(\mathrm{Qd1Qd2Cos}[\mathrm{Q2}] + \mathrm{Qd1Qd3Cos}[\mathrm{Q2}] + \mathrm{Qdd1Sin}[\mathrm{Q2}])\mathrm{Sin}[\mathrm{Q3}]) -$
$\mathrm{a23}gm\mathrm{Sin}[\mathrm{Q1}]\mathrm{Sin}[\mathrm{Q2} + \mathrm{Q3}] + \mathrm{a23}g\mathrm{m4Cos}[\mathrm{Q3}]\mathrm{Sin}[\mathrm{Q2}]\mathrm{Sin}[\mathrm{Q4}] +$

$$\mathrm{a23}g\mathrm{m4Cos[Q2]Sin[Q3]Sin[Q4]}) + \mathrm{Cos[Q2]}(-\frac{1}{2}\mathrm{a12}^2\mathrm{m3Qdd2} - \frac{1}{2}\mathrm{a12}^2\mathrm{m4Qdd2} - \frac{1}{4}\mathrm{a12}^2\mathrm{m2Qdd1Cos[Q2]} - \frac{1}{2}\mathrm{a12a23m3Qdd1Cos[Q2]Cos[Q3]} - \frac{1}{2}\mathrm{a12a23m4Qdd1Cos[Q2]Cos[Q3]} + \frac{5}{12}\mathrm{a12}^2\mathrm{m2Qd1Qd2Sin[Q2]} - \frac{1}{2}\mathrm{a12}^2\mathrm{m3Qd1}^2\mathrm{Cos[Q2]Sin[Q2]} - \frac{1}{2}\mathrm{a12}^2\mathrm{m4Qd1}^2\mathrm{Cos[Q2]Sin[Q2]} + \mathrm{a12a23m3Qd1Qd2Cos[Q3]Sin[Q2]} + \mathrm{a12a23m4Qd1Qd2Cos[Q3]Sin[Q2]} + \mathrm{a12a23m3Qd1Qd3Cos[Q3]Sin[Q2]} + \mathrm{a12a23m4Qd1Qd3Cos[Q3]Sin[Q2]} + \frac{1}{12}\mathrm{a12}^2\mathrm{m2(Qdd1Cos[Q2]} - \mathrm{Qd1Qd2Sin[Q2])} + \mathrm{a12a23m3Qd1Qd2Cos[Q2]Sin[Q3]} + \mathrm{a12a23m4Qd1Qd2Cos[Q2]Sin[Q3]} + \mathrm{a12a23m3Qd1Qd3Cos[Q2]Sin[Q3]} + \mathrm{a12a23m4Qd1Qd3Cos[Q2]Sin[Q3]} + \frac{1}{2}\mathrm{a12a23m3Qdd1Sin[Q2]Sin[Q3]} + \frac{1}{2}\mathrm{a12a23m4Qdd1Sin[Q2]Sin[Q3]} - \mathrm{a12}gm\mathrm{Sin[Q1]Sin[Q2 + Q3]} + \mathrm{a12}g\mathrm{m4Cos[Q3]Sin[Q2]Sin[Q4]} + \mathrm{a12}g\mathrm{m4Cos[Q2]Sin[Q3]Sin[Q4]} + \mathrm{Cos[Q3]}(-\frac{1}{4}\mathrm{a12a23m3Qdd2} - \frac{1}{2}\mathrm{a12a23m4Qdd2} - \frac{1}{4}\mathrm{a23}^2\mathrm{m3Qdd1Cos[Q2]Cos[Q3]} - \frac{1}{2}\mathrm{a23}^2\mathrm{m4Qdd1Cos[Q2]Cos[Q3]} - \frac{1}{4}\mathrm{a12a23m3Qd1}^2\mathrm{Cos[Q2]Sin[Q2]} - \frac{1}{2}\mathrm{a12a23m4Qd1}^2\mathrm{Cos[Q2]Sin[Q2]} + \frac{5}{12}\mathrm{a23}^2\mathrm{m3Qd1Qd2Cos[Q3]Sin[Q2]} + \mathrm{a23}^2\mathrm{m4Qd1Qd2Cos[Q3]Sin[Q2]} + \frac{5}{12}\mathrm{a23}^2\mathrm{m3Qd1Qd3Cos[Q3]Sin[Q2]} + \mathrm{a23}^2\mathrm{m4Qd1Qd3Cos[Q3]Sin[Q2]} + \frac{5}{12}\mathrm{a23}^2\mathrm{m3Qd1Qd2Cos[Q2]Sin[Q3]} + \mathrm{a23}^2\mathrm{m4Qd1Qd2Cos[Q2]Sin[Q3]} + \frac{5}{12}\mathrm{a23}^2\mathrm{m3Qd1Qd3Cos[Q2]Sin[Q3]} + \mathrm{a23}^2\mathrm{m4Qd1Qd3Cos[Q2]Sin[Q3]} + \frac{1}{4}\mathrm{a23}^2\mathrm{m3Qdd1Sin[Q2]Sin[Q3]} + \frac{1}{2}\mathrm{a23}^2\mathrm{m4Qdd1Sin[Q2]Sin[Q3]} + \frac{1}{12}\mathrm{a23}^2\mathrm{m3(Cos[Q3](Qdd1Cos[Q2]} - \mathrm{Qd1Qd2Sin[Q2]} - \mathrm{Qd1Qd3Sin[Q2])} - \mathrm{(Qd1Qd2Cos[Q2]} + \mathrm{Qd1Qd3Cos[Q2]} + \mathrm{Qdd1Sin[Q2])Sin[Q3])} - \mathrm{a23}gm\mathrm{Sin[Q1]Sin[Q2 + Q3]} + \mathrm{a23}g\mathrm{m4Cos[Q3]Sin[Q2]Sin[Q4]} + \mathrm{a23}g\mathrm{m4Cos[Q2]Sin[Q3]Sin[Q4]}))\} \tag{3.37}$$

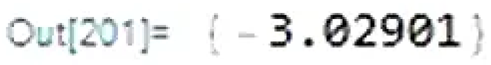
3.2.4 Selection of The Motors

For selection of exact motor numerical values will be found. Torque1 value will be only shown and selection of motor for this link. By using the inverse kinematic analysis joint angles are found. Weight of the links were found using the SolidWorks.



```
In[181]:= a12 = 0.13
          a23 = 0.13

Out[181]= 0.13

Out[182]= 0.13

In[261]:= Q1 = 0
          Q2 = 0
          Q3 = 0
          Q4 = 0
          Qd1 = 0.15
          Qd2 = 0.12
          Qd3 = 0.13
          Qd4 = 0.15
          Qdd1 = 0.01
          Qdd2 = 0.01
          Qdd3 = 0.01
          Qdd4 = 0.01
          m1 = 2.5
          m2 = 1.1
          m3 = 0.95
          m4 = 0.3
          g = 9.81
          m = 0.1
```

Figure 3.2 Torque values of the robotic arm



```
Out[201]= {-3.02901}
```

Figure 3.3 One of the torque value

$T_0 = -0.000359829\ Nm$

$T_1 = -3.02901 Nm$

$T_2 = -0.860866\ Nm$

The servo motor usage is decided in the robot. These are listed in below.

For the T_0value most proper servo motor is MG996R. Datasheet of this servo motor shown as[6]:

- Operating voltage: 4.8 ~ 6.6V
- Holding Torque: 9.4kg/cm(4.8v)-11kg/cm(6.0v)
- It holds 10mA current at idle. No-load current: 170mA
- Holding current: 1400mA
- Weight: 55g
- Size: 40.9×20×42.7mm

For the T_1value most proper servo motor is DS3230MG. Datasheet of this servo motor shown as[7]:

- Holding Torque (5V): 27 kg / cm
- Holding Torque (6.8 V): 32 kg / cm
- Speed: 0.16 sec / 60 ° (5V) / 0.12 sec / 60 ° (6.8 V)
- Operating voltage: 4.8 ~ 7.2 DC
- Weight: 65 g
- Size: 40 x 20 x 40.5 mm

For the T_2 value most proper servo motor is DS3225. Datasheet of this servo motor shown as[7]:

- Holding Torque (5V): 21 kg / cm
- Holding Torque (6,8 V): 24,5 kg / cm
- Speed: 0.15 sec / 60 ° (5V) / 0,13 sec / 60 ° (6,8 V)
- Operating voltage: 4.8 ~ 6.8 dc volt
- Weight: 60 g
- Type of Motor: DC Motor
- Gear Type: Copper and Aluminum
- Operating frequency: 50-333Hz
- Size: 40 x 20 x 40,5 mm

4 Remote Control based on MQTT Protocol

4.1 Software Programming and MQTT Protocol

A lightweight messaging protocol called MQTT (Message Queuing Telemetry Transport) was created for effective and dependable device-to-device communication, especially in constrained settings with little bandwidth or high latency. Messages can be published to topics by devices or applications using the publish-subscribe messaging pattern used by MQTT, and other devices or applications can subscribe to those topics to receive the messages.

Mosquitto should be installed into the Rasperry Pi to be able to use MQTT in it. Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1 and 3.1. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers.

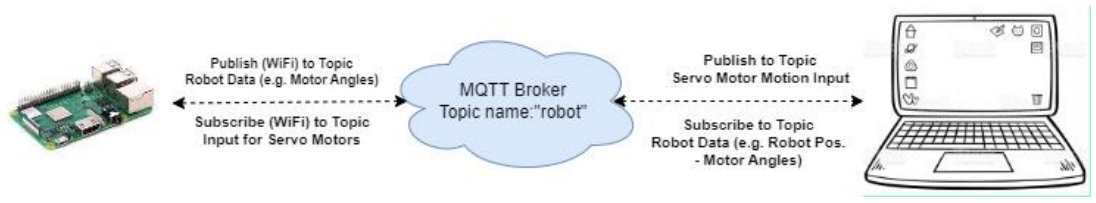


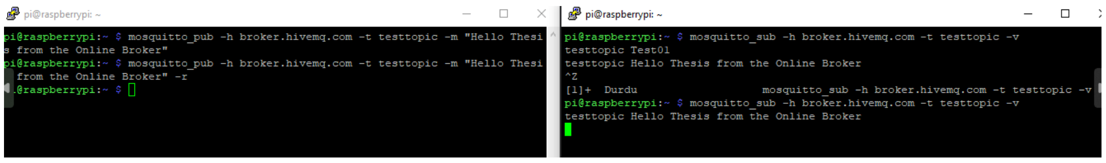
Figure 4.1 Simple overview of the MQTT protocol for robot and PC

4.1.1.1 Publish-Subscribe Messaging Pattern

By placing a broker in the middle, the publish-subscribe pattern in MQTT decouples message senders (publishers) from message receivers (subscribers). Publishers are in charge of sending communications to the broker without knowing whether or not they will be read by anyone. By subscribing to particular topics on the broker, subscribers indicate their interest in receiving messages. The broker serves as a middleman, transferring published messages from publishers to the appropriate subscribers.

In this thesis, messages should be published and subscribed in both Raspberry Pi python and PC with the C# to be able to exchange the data and control the mobile and serial robot arm. Simple code snippet can be seen below.

```
private void gripperForward_Click(object sender, EventArgs e)
{
      Task.Run(() =>
      {
            if (mqttClient != null && mqttClient.IsConnected)
            {
                  mqttClient.Publish("testtopic",
            Encoding.UTF8.GetBytes("Gripper Forward"));
            }
      });

}
```



```
pi@raspberrypi:~ $ mosquitto_pub -h broker.hivemq.com -t testtopic -m "Hello Thesi
s from the Online Broker"
pi@raspberrypi:~ $ mosquitto_pub -h broker.hivemq.com -t testtopic -m "Hello Thesi
 from the Online Broker" -r
.@raspberrypi:~ $ 
```

```
pi@raspberrypi:~ $ mosquitto_sub -h broker.hivemq.com -t testtopic -v
testtopic Test01
testtopic Hello Thesis from the Online Broker
^Z
[1]+  Durdu                   mosquitto_sub -h broker.hivemq.com -t testtopic -v
pi@raspberrypi:~ $ mosquitto_sub -h broker.hivemq.com -t testtopic -v
testtopic Hello Thesis from the Online Broker
```

Figure 4.2 Communication with the Broker

Additionally, connection needs to be created by the C# tool on the PC.

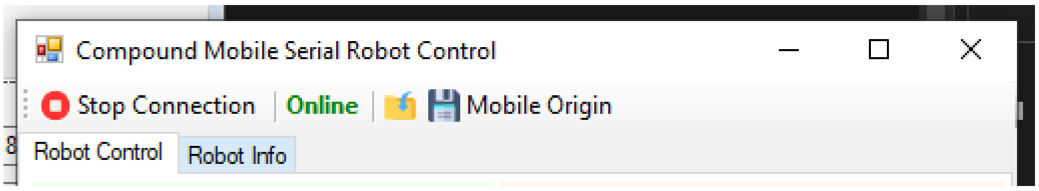


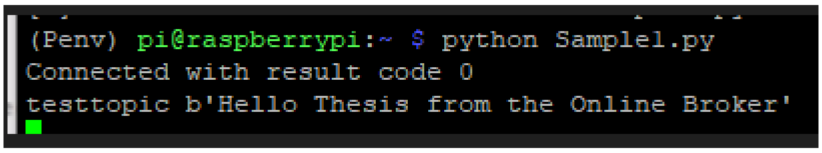
Figure 4.3 Establishing of the connection in the PC

4.1.1.2 MQTT Broker

A central server known as the MQTT broker serves as a go-between for publishers and subscribers. Based on the topic hierarchy and subscription patterns, it receives published messages from publishers and distributes them to the appropriate subscribers. The broker is in charge of overseeing client connections, dealing with subscriptions and unsubscriptions, and making sure messages are delivered consistently. Popular MQTT broker implementations are readily accessible, including Mosquitto, HiveMQ, and EMQ.In this thesis, HiveMQ broker have been used.

4.1.1.3 MQTT Topics

In MQTT, topics act as a hierarchical structure and constitute the foundation for message filtering and routing. A topic is a string that designates a message's subject or category. Similar to a file system path, topics are arranged hierarchically using forward slashes (/) as separators. Examples of acceptable MQTT subjects include "sensors/temperature" and "devices/+/status". Multiple levels are possible for topics, allowing for adaptable subscription structures. In subscriptions, wildcards can be used to match various topics: The topic hierarchy is just one level deep when using the "+" wildcard. The wildcard "#" matches levels of any number, including 0 or more levels. For instance, "devices/+/status" will match subjects such as "devices/device1/status" and "devices/device2/status" if you subscribe to it.



```
(Penv) pi@raspberrypi:~ $ python Sample1.py
Connected with result code 0
testtopic b'Hello Thesis from the Online Broker'
```

Figure 4.4 Topic of the communication

In the publish-subscribe approach, message decoupling and effective message delivery are made possible by publishers and subscribers interacting with the MQTT broker. Subscribers receive communications by subscribing to pertinent topics, and publishers publish messages to specified topics. Based on the subscribers' subscriptions and the subject hierarchy, the broker makes sure that published messages are delivered to the correct subscribers.

MQTT is a flexible and scalable messaging architecture that works well for Internet of Things (IoT) applications where a large number of devices need to exchange data in a quick and effective way. Across distributed systems, it enables simple integration, real-time communication, and efficient information dissemination.

4.1.1.4 C# WinForm Application & Python Script for Remote Control

First of everything, connection to the MQTT broker should be established, this can be done with the “Start Connection” button with the simple code snippet of the function.

```
Task.Run(() =>
{
	mqttClient = new MqttClient("broker.hivemq.com");
	mqttClient.MqttMsgPublishReceived +=
MqttClient_MqttMsgPublishReceived;
	mqttClient.Subscribe(new string[] { "testtopic" }, new byte[] {
MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE });
	mqttClient.Connect("testtopic");

});
```

Thanks to the MqttMsgPublishReceived function, tool will be able to read the data from the broker continuously, since the message also can be published by the robot via Raspberry Pi, it's code snippet can be seen.

```
var message = Encoding.UTF8.GetString(e.Message);
if (message.StartsWith("Servo1"))
{
	string pattern =
@"Servo1\=(.*?),Servo2\=(.*?),Servo3\=(.*?),Servo4\=(.*)";
	RegexOptions options = RegexOptions.Multiline;
	foreach (Match m in Regex.Matches(message, pattern, options))
	{
		firstJoint.Invoke((MethodInvoker)(() => firstJoint.Text =
	m.Groups[1].Value));
		secondJoint.Invoke((MethodInvoker)(() => secondJoint.Text =
	m.Groups[2].Value));
		thirdJoint.Invoke((MethodInvoker)(() => thirdJoint.Text =
	m.Groups[3].Value));
		fourthJoint.Invoke((MethodInvoker)(() => fourthJoint.Text =
	m.Groups[4].Value));
	}
}
```

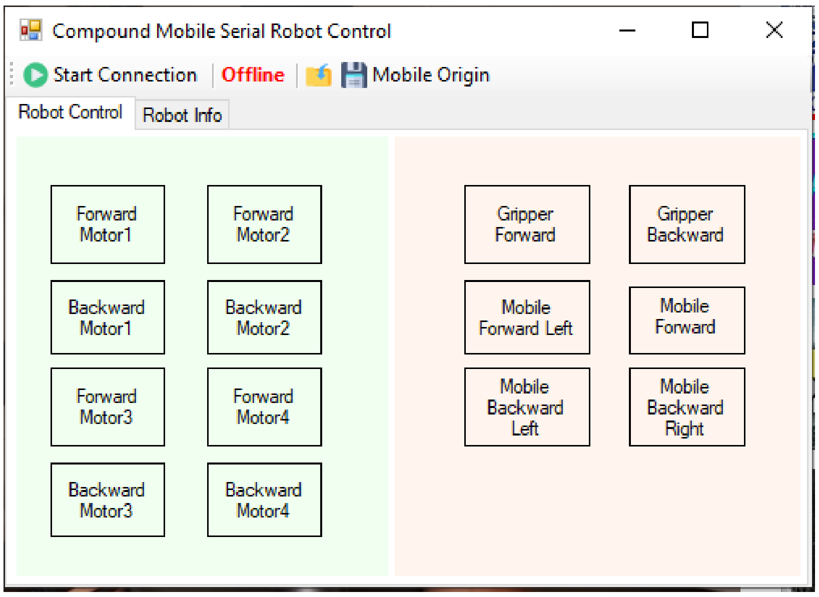


Figure 4.5 Main view and control of the motors on WinForm tool

4.1.1.5 Robot Control Window

In this window, each motor included in the mobile and robot arm can be controlled to forward and backward, forward will increase robot angle 20 degrees and backward will decrease robot angle 20 degrees, with this window mobile and serial robot can be controlled manually to the desired point.

4.1.1.6 Robot Info Window

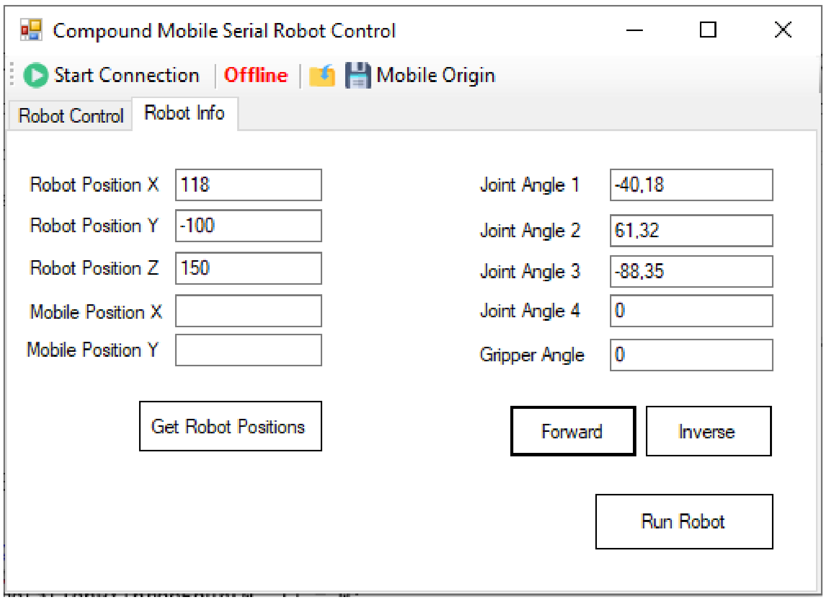
With this window mobile and serial robot arm control can be controlled automatically with Forward and Inverse kinematics automatization.

Robot Position X, Y, Z represents position of the end-effector for the gripper, Mobile Position X and Y represents position of the mobile robot about the save mobile origin from the Mobile Origin save button of the tool. Joint Angle 1, 2, 3 ,4 and gripper angle represents angles of the servo motors. Whenever Robot Position X, Y, and Z is entered Joint Angles' can be calculated by "Inverse" button which calls InverseKinematics function, this calculation and end-effector position can be checked by the "Forward" button that calls ForwardKinematics function.

By the "Get Robot Positions" button, C# tool will publish data to the MQTT broker via topic, raspberry pi python script will be reading the on messages from the broker again via same topic, whenever this button is clicked desired data will be published and python script will identify it thanks to its subscription and then raspberry pi will behave as publisher to send real-time data from the robot to the C# Tool, PC.

Automatized tasks can be handled with the "Run Robot", whenever end-effector position is decided and inverse kinematics is calculated automatically by the tool, "Run Robot" button will change values of the servo motors to the calculated joint angle values by publishing textbox values filled by the calculation the the MQTT broker via topic, and then raspberry pi will identify and resolve it with the Regular Expression (Regex) usage to check each joint angle value of the servo motors from the input text sent by the C# Tool. Simple part of the "Run Robot" button functionality can be seen.

```
var robotRunningTest =
"RunServo1="+firstJoint.Text+",RunServo2="+secondJoint.Text+",RunServo3="
+thirdJoint.Text+",RunServo4="+fourthJoint.Text+",";
Task.Run(() =>
{
     if (mqttClient != null && mqttClient.IsConnected)
     {

mqttClient.Publish("testtopic",Encoding.UTF8.GetBytes(robotRunningTest));
}});
```



Compound Mobile Serial Robot Control

Start Connection | Offline | Mobile Origin

Robot Control | Robot Info

Robot Position X	118	Joint Angle 1	-40,18
Robot Position Y	-100	Joint Angle 2	61,32
Robot Position Z	150	Joint Angle 3	-88,35
Mobile Position X		Joint Angle 4	0
Mobile Position Y		Gripper Angle	0

Figure 4.6 Control of the robot automatically by direct and inverse analysis

Direct and inverse kinematics will be automatically handled by the WinForm C# tool to find joint angles (servo motor angles) from the end-effector X, Y, Z position of the robotic arm. Information of the end effector position will be received by the python script running on the raspberry pi which behaves like a publisher in this case.

Necessary transformation matrices were defined to be used in the equations, in addition to this MathKernel could be used by using NuGet Package of the external tool. MultiplyMatrix function were written to multiply two matrices, Regex were used to extract equation result as radian and then it was converted to angle.

There was before jitter when controlling the servo motors, the issue was figured out and resolved by using PiGPIOFactory as an input for the servo motor variable decleration, in this case pigpiod service needs to be run by “sudo pigpiod” command.

Some parts of the initial version of the Raspberry Pi python code can be seen below.

```
from gpiozero import AngularServo
import pigpio
from time import sleep
import paho.mqtt.client as mqtt
import re
from gpiozero.pins.pigpio import PiGPIOFactory

factory = PiGPIOFactory()

servo = AngularServo(18, min_angle=-90, max_angle=90,
pin_factory=factory)
servo2 = AngularServo(23, min_angle=-90, max_angle=90,
pin_factory=factory)
servo3 = AngularServo(24, min_angle=-90, max_angle=90,
pin_factory=factory)
servo4 = AngularServo(25, min_angle=-90, max_angle=90,
pin_factory=factory)
servo5 = AngularServo(12, min_angle=-90, max_angle=90,
pin_factory=factory)

def on_connect(client, userdata, flags, rc):
    print("Connected to the broker succesfully!"+str(rc))
    client.subscribe("testtopic")

def on_message(client, userdata, msg):
    print(str(msg.payload))
    if (str(msg.payload) == "Forward1"):
        servo.angle = 90
    elif (str(msg.payload) == "Backward1"):
        servo.angle = 0
    elif (str(msg.payload) == "Forward2"):
        servo2.angle = servo.angle + 20
    elif (str(msg.payload) == "Backward2"):
        servo2.angle = servo.angle - 20
    elif (str(msg.payload) == "Forward3"):
        servo3.angle = servo.angle + 20
    elif (str(msg.payload) == "Backward3"):
        servo3.angle = servo.angle - 20
    elif (str(msg.payload) == "Forward4"):
        servo4.angle = 90
        print("test item")
    elif (str(msg.payload) == "Backward4"):
        servo4.angle = 0
    elif (str(msg.payload) == "Get Data"):
        servoAngles = "Servo1="+ str(servo.angle) + ",Servo2=" +
str(servo2.angle) + ",Servo3=" + str(servo3.angle) + ",Servo4=" +
str(servo4.angle)
        client.publish("testtopic", servoAngles)
    elif (str(msg.payload).startswith("RunServo1")):
        regex =
r"RunServo1\=(.*?),RunServo2\=(.*?),RunServo3\=(.*?),RunServo4\=(.*)"
        pattern = re.compile(regex)
        for match in pattern.finditer(str(msg.payload)):
            servo.angle = int(match.group(1))
            sleep(100)
            servo2.angle = int(match.group(2))
            sleep(100)
            servo3.angle = int(match.group(3))
            sleep(100)
            servo4.angle = int(match.group(4))
    elif (str(msg.payload) == "Gripper Forward"):
        servo5.angle = servo5.angle + 20
    elif (str(msg.payload) == "Gripper Backward"):
        servo5.angle = servo5.angle - 20
    print(msg.topic + " " + str(msg.payload))

    // Mobile Conditions
    if (str(msg.payload) == "MobileForward1"):
        GPIO.output(in1,GPIO.HIGH)
        GPIO.output(in2,GPIO.LOW)
    elif (str(msg.payload) == "MobileBacward1"):
        print("backward")
        GPIO.output(in1,GPIO.LOW)
        GPIO.output(in2,GPIO.HIGH)

     GPIO.cleanup()

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect("broker.hivemq.com", 1883, 60)
client.loop_forever()
```

5 Prototyping of the Robotic Arm

5.1 Assembly and Specification of the Mechanism

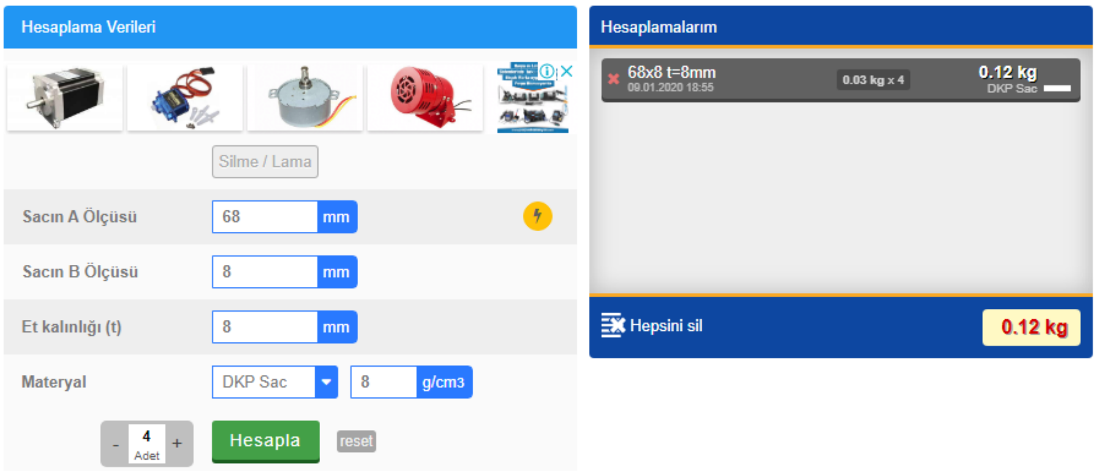
5.1.1 Materials

The type of material for the robot arm needed to be chosen before employing laser cutting. To make the base and linkages stronger and more affordable, sheet metal were selected. Transmission steel used for the shaft.

After producing everything is needed, there was some issues during the assembling phase. The alignment of the shaft and motor shaft is the issue at hand. Spacer should be employed in order to resolve this issue. Delrin fiber were used for this spacer. Additionally, in order to reduce friction and achieve balanced movement, Delrin fiber were also employed in the rotating portion of the base.

5.1.2 Weight Analysis for the Links

Weight analyses were handled to sheet metal of the Link-1. There are two Link-1 in the robot arm.



Hesaplama Verileri

Sacın A Ölçüsü	68 mm
Sacın B Ölçüsü	8 mm
Et kalınlığı (t)	8 mm
Materyal	DKP Sac, 8 g/cm3
Adet	4

Hesaplamalarım

68x8 t=8mm 09.01.2020 18:55 0.03 kg x 4 0.12 kg DKP Sac

Hepsini sil 0.12 kg

Figure 5.1 Weight of the Link-1

5.1.3 Assembly of the Robot Arm

During assembly, the robot arm's base began. A top tray, bottom tray, side holders, and fiber delrin were used in the first phase of assembly.

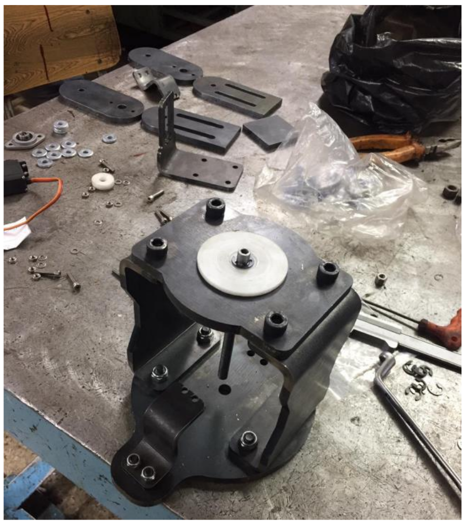


Figure 5.2 Base of the robot arm

The assembly was then completed by adding the spinning portion of the base. There were alignment problems with the motor shaft and shaft within this spinning component. The use of fiber delrin was a solution to these alignment issues. As an example, consider the following:

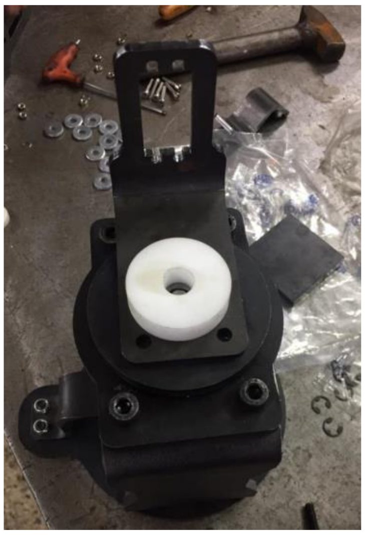


Figure 5.3 Fiber Derlin

After that, the first link to the seating component of our system was fully assembled. A sheet bending issue that occurred during the laser cutting process was present in this seating component. The heating procedure that was used to successfully bend the seating portion was implemented to remedy this issue.

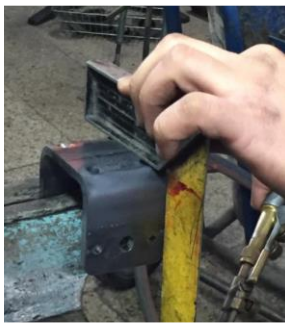


Figure 5.4 Bending test

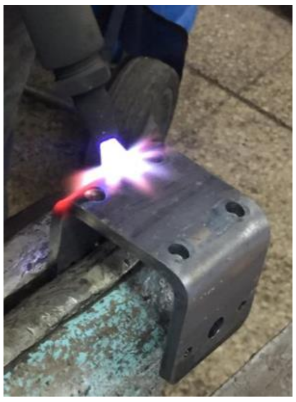


Figure 5.5 Heating process

The set screw hole was opened once the issue was fixed, and tapping was then done.

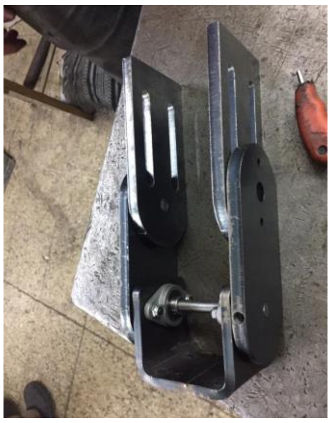


Figure 5.6 Assembly of the upper body

The assembly was proceeded by connecting the seating part of the upper body to the base.

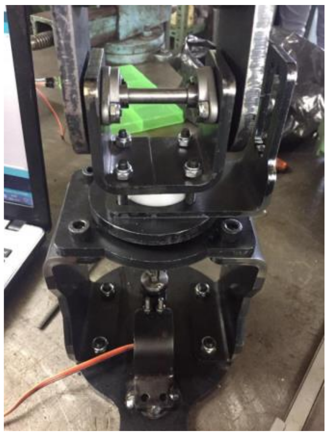


Figure 5.7 Assembly of base to the body

After the two links of the robot is assembled to the base of the robot arm, the servo motor holding parts made of sheet metal are welded to the links so that the servo motor can be fixed to be able to control the shafts. Additionally, another sheet metal is welded to the third link to be able to assemble and control the fourth joint, holding part of the last servo motor were also welded to that sheet metal.

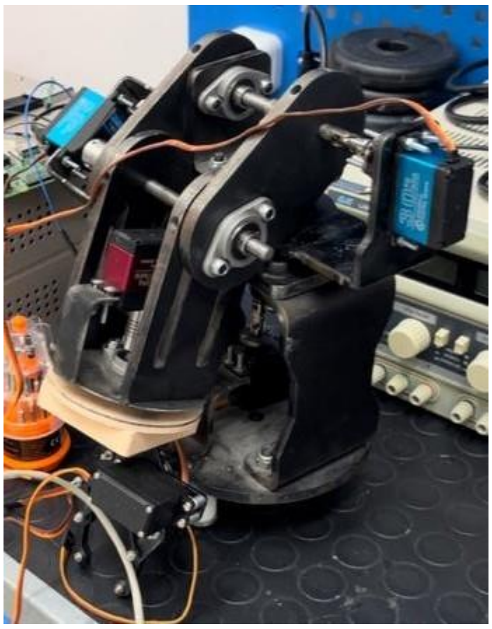


Figure 5.8 Fourth Joint and Gripper

After that, assembly of the mobile robot has been handled, DC motors and gear system of the mobile robot is checked, configured and fixed. Sheet metal was produced for the mobile robot and robot arm assembly, Delrin was used for this assemply in addition to sheet metal. It can be seen below.

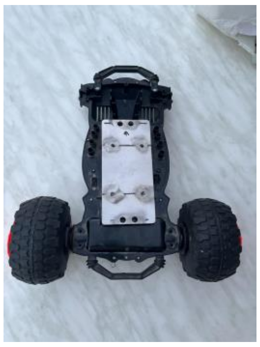


Figure 5.9 Front side of the mobile robot

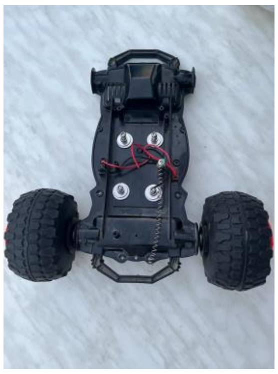


Figure 5.10 Back side of the mobile robot

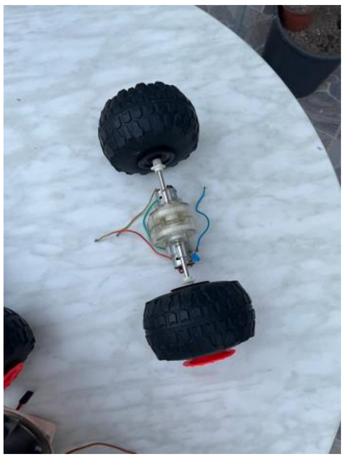


Figure 5.11 Wheels and gear system with DC motor

Assembly of the mobile and serial robot arm has been established.

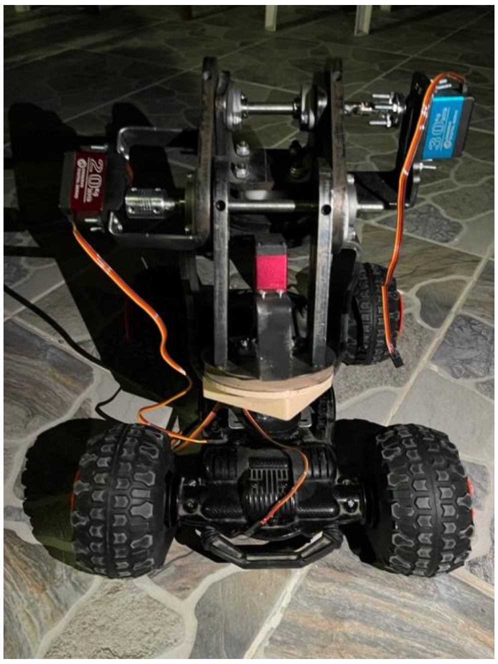


Figure 5.12 Compound mobile and serial robot arm

In this thesis, additionally to the robot arm, mobile robot can be also controlled remotely with MQTT protocol by publish and subscriber pattern from the desired topic from the broker. In that case, mobile robot DC motor can be controlled from the C# WinForm tool.

At the circuit level, electrical electronic parts such as voltage reducer, cable, LiPo battery, DC power supply, L298N motor driver, servo motors, Raspberry Pi 3B, breadboard and etc. were used.

5.2 Trajectory Planning

Trajectory planning is planning of the desired movements of the manipulator. Manipulators with multi degree of freedom for accomplishing various complex manipulation in the work space. Path is only for geometric description but trajectory also include timing change of the manipulator.

Trajectory planning include 2 terms that are joint space and operational space. Joint space is motion to be made by the robot by its joint values. The motion between the two points is unpredictable. In operational space two points is known at all times and it is controllable.

5.2.1 Joint-Space Trajectories

Trajectories are specified by defining some characteristic points that are directly assigned by some specifications and assigned by defining desired configurations x in the work-space, which are then converted in the joint space using the inverse kinematic model.

In that given points trajectories must be computationally efficient, the position and velocity profiles must be continuos functions of time, undesired effects must be minimized or completely avoided.

5.2.2 Polynomial Trajectories

In these cases a trajectory is specified by assigning initial and final conditions on: time , position, velocity, acceleration. Then, the problem is to determine a function q = q(t) so that condition is satisfied.

Polynomial functions should solved as;

$$q(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_n t^n \tag{5.1}$$

The degree n (3, 5, ...) of the polynomial depends on the number of boundary conditions that must be verified and on the desired “smoothness” of the trajectory.
Given an initial and a final instant t_i,t_f , a (segment of a) trajectory may be specified by assigning initial and final conditions:

- initial position and velocity $q_i, \dot{q}_i$
- final position and velocity $q_f, \dot{q}_f$

There are four boundary conditions in this situation, so a polynomial of degree at least 3 must be considered from the Eq. 5.1.

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

Where the four parameters a_0, a_1, a_2, a_3 must be defined so that the boundary conditions are satisfied.

From the boundary conditions, it follows that

$$q(ti) = a_0 + a_1 ti + a_2 ti^2 + a_3 ti^3 = q_i \tag{5.2}$$

Equations should be followed as shown below, by taking derivative of the equation.

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

$$q(tf) = a_0 + a_1 tf + a_2 tf^2 + a_3 tf^3 = q_f$$

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

In order to solve these equations, the first moment of the motion is assumed that

$t_i = 0$.

Therefore:

$a_0 = q_i$

$a_1 = \dot{q}_i$

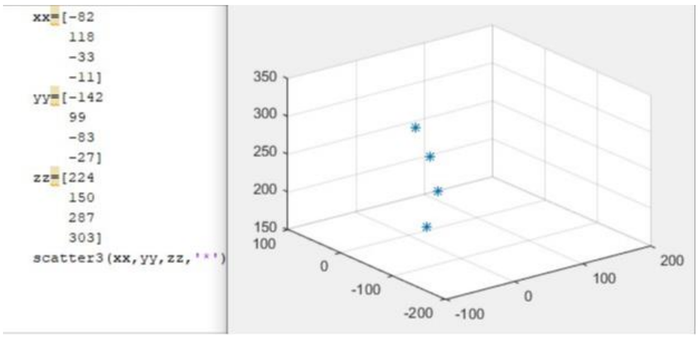
$a_2 = (-3(q_i - q_f) - (2\,\dot{q}_i + \dot{q}_f)\, t_f) \ / \ t_f^{\,2}$

$a_3 = (2(q_i - q_f) + (\dot{q}_i + \dot{q}_f)\, t_f \ ) \ / \ t_f^{\,3}$

5.2.3 Task Planning for the End Effector

For the robot arm, necessary steps of the task are determined. First of all, start position is setted. This point represent as a pole position and the movement starts from here. After that, the movement path of the robot arm are needed to be defined. For this movement, the way points of the end effector are determined. Thanks to these way points, joint positions and velocities can be found.

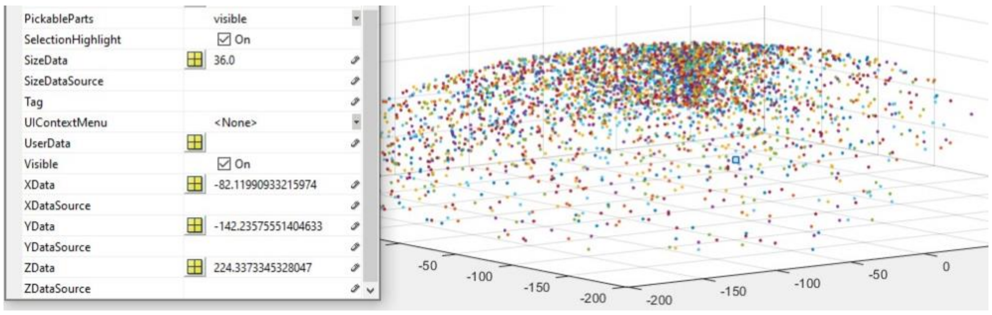
Firstly, waypoints of the robot arm are identified according to workspace of the robot arm. These waypoints also include pole position and final position of the end effector.



```
xx=[-82
    118
    -33
    -11]
yy=[-142
    99
    -83
    -27]
zz=[224
    150
    287
    303]
scatter3(xx,yy,zz,'*')
```

Figure 5.13 Waypoints of our Robot Arm

Pole position of the robot arm is selected by using the workspace analysis which is done previously. This pole position coordinates are shown below:



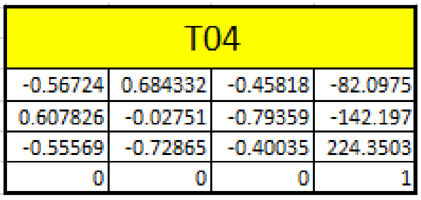
```
PickableParts        visible
SelectionHighlight   ☑ On
SizeData             36.0
SizeDataSource
Tag
UIContextMenu        <None>
UserData
Visible              ☑ On
XData                -82.11990933215974
XDataSource
YData                -142.23575551404633
YDataSource
ZData                224.3373345328047
ZDataSource
```

Figure 5.14 Workspace analysis for the pole position

After the selection of coordinates of the pole position, inverse kinematics is applied to find corresponding joint positions. After inverse analysis of robot arm, selection of the joint position done according to robot arm.

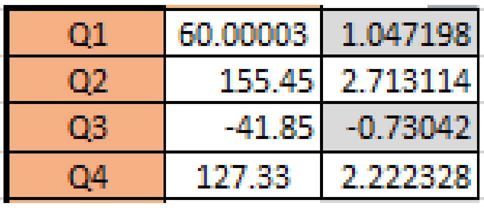
Q1 = 1.0472   Q2 = 2.71313   Q3 = -0.729867   Q4 = 2.22228

Finally, after finding the joint positions by inverse analysis, accuracy of these positions is checked according to direct analysis.



T04			
-0.56724	0.684332	-0.45818	-82.0975
0.607826	-0.02751	-0.79359	-142.197
-0.55569	-0.72865	-0.40035	224.3503
0	0	0	1

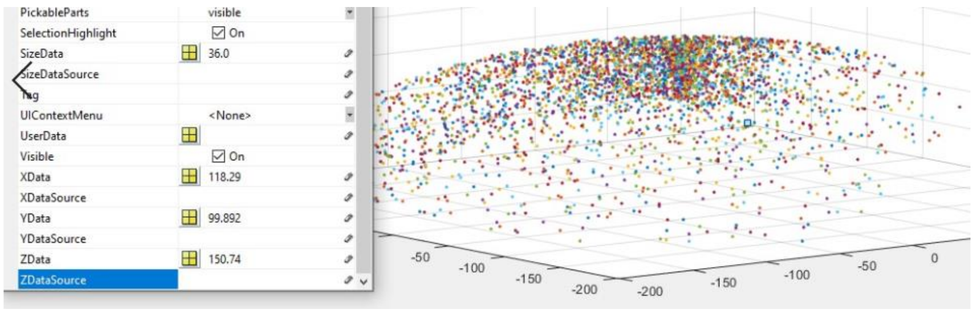
Figure 5.15 End-Effector Position



Q1	60.00003	1.047198
Q2	155.45	2.713114
Q3	-41.85	-0.73042
Q4	127.33	2.222328

Figure 5.16 Joint Angles

The second position of the robot arm is selected by using the workspace analysis which is done previously. This second position coordinates are shown below:



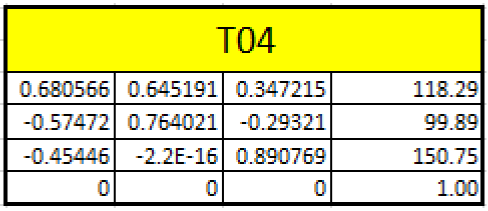
PickableParts	visible
SelectionHighlight	☑ On
SizeData	36.0
SizeDataSource	
Tag	
UIContextMenu	<None>
UserData	
Visible	☑ On
XData	118.29
XDataSource	
YData	99.892
YDataSource	
ZData	150.74
ZDataSource	

Figure 5.17 Workspace analysis for the second position

After the selection of coordinates of the second position, inverse kinematics is applied for finding joint positions.

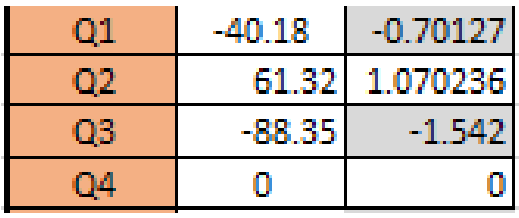
Q1 = -0.70127 Q2 = 1.07024 Q3 = -1.542 Q4 = 0

Finally, after finding the joint positions by inverse analysis, accuracy of these positions checked according to direct analysis.



T04			
0.680566	0.645191	0.347215	118.29
-0.57472	0.764021	-0.29321	99.89
-0.45446	-2.2E-16	0.890769	150.75
0	0	0	1.00

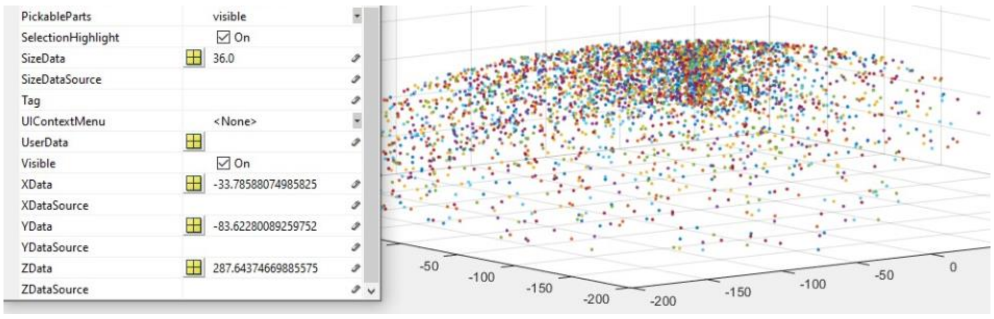
Figure 5.18 End-effector positions



Q1	-40.18	-0.70127
Q2	61.32	1.070236
Q3	-88.35	-1.542
Q4	0	0

Figure 5.19 Joint angles

Third position of the robot arm is selected by using the workspace analysis which is done previously. This third position coordinates are shown below:



PickableParts		visible
SelectionHighlight		☑ On
SizeData	▦	36.0
SizeDataSource		
Tag		
UIContextMenu		<None>
UserData	▦	
Visible		☑ On
XData	▦	-33.78588074985825
XDataSource		
YData	▦	-83.62280089259752
YDataSource		
ZData	▦	287.64374669885575
ZDataSource		

Figure 5.20 Workspace analysis for the third postion

After selection of coordinates of third position, inverse kinematics is applied for finding joint positions.

Q1 = 1.18682   Q2 = 2.17322   Q3 = -0.473627   Q4 1.66013

Finally, after finding the joint positions by inverse analysis, the accuracy of these positions are checked according to the direct analysis.

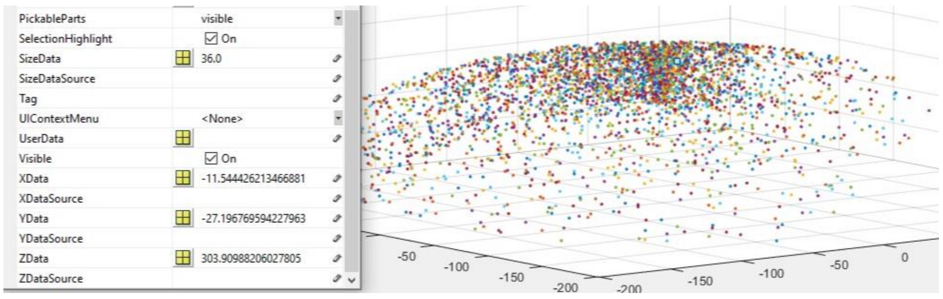
T04			
-0.91919	0.13063	-0.37152	-33.7854
0.38375	0.08519	-0.9195	-83.6176
-0.08847	-0.98776	-0.12843	287.646
0	0	0	1

Figure 5.21 End-effector positions

Q1	67.999	1.18681
Q2	124.514	2.17318
Q3	-27.135	-0.4736
Q4	95	1.66012

Figure 5.22 Joint Angles

Fourth position of the robot arm is selected by using the workspace analysis which is done previously. This fourth position coordinates are shown below:

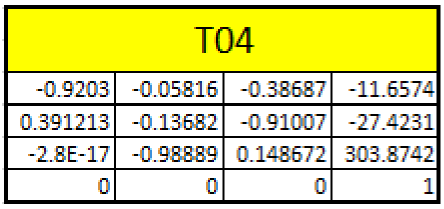


PickableParts		visible
SelectionHighlight		☑ On
SizeData	▦	36.0
SizeDataSource		
Tag		
UIContextMenu		<None>
UserData	▦	
Visible		☑ On
XData	▦	-11.544426213466881
XDataSource		
YData	▦	-27.196769594227963
YDataSource		
ZData	▦	303.90988206027805
ZDataSource		

Figure 5.23 Workspace analysis for the fourth position

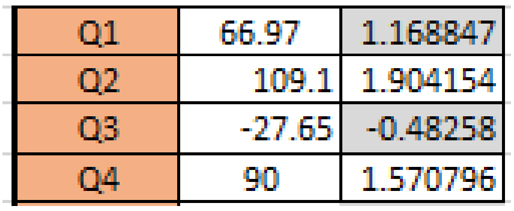
After the selection of coordinates of the last position, inverse kinematics is applied for finding joint positions.

Q1 = 1.16937 Q2 = 1.90315 Q3 = -0.482841 Q4 = 1.57087



T04			
-0.9203	-0.05816	-0.38687	-11.6574
0.391213	-0.13682	-0.91007	-27.4231
-2.8E-17	-0.98889	0.148672	303.8742
0	0	0	1

Figure 5.24 End-effector position



Q1	66.97	1.168847
Q2	109.1	1.904154
Q3	-27.65	-0.48258
Q4	90	1.570796

Figure 5.25 Joint Angles

5.2.4 Assumed Polynomial Functions for Each Joint Positions

First Joint

In order to find polynomial equations in the first step, first time of the joint is assumed as $t_i = 0$.

Therefore:

$a_0 = q_i$

$a_1 = \dot{q}_i$

$a_2 = (-3(q_i - q_f) - (2\,\dot{q}_i + \dot{q}_f)\,t_f\,) \; / \; t_f^{\,2}$

$a_3 = (2(q_i - q_f) + (\dot{q}_i + \dot{q}_f)\,t_f\,) \, / \; t_f^{\,3}$

For pole position of our robot arm, velocity is assumed as 0 so;

$$\dot{q}_{11} = 0$$

Then;

$$a_1 = 0$$

According to position of first joint for pole position of our robot arm, previously position is found as;

$$q_{11} = 1.0472$$

Then;

$$a_0 = 1.0472$$

For finding a_2 , q_f value should be known, then this value equal to joint value in the second position so;

$$q_{f1} = -0.70127$$

For finding a_2, also $\dot{q}_f$ value should be known, this value is assumed as ‘0.07’. This assumption is made according to motor datasheet.

Then;

$$a_2 = (-3(1.0472+0.70127) - (2 * 0 +0.07)\, t_f) \;/\; t_f^{\,2}$$

$$a_3 = (2(1.0472+0.70127) + (0 +0.07)\, t_f) \;/\; t_f^{\,3}$$

Second Joint

In order to find polynomial equations in first step, the moment that t_i assumed as 0.

For pole position of our robot arm, velocity is assumed as 0 so;

$$\dot{q}_{21} = 0$$

Then;

$$a_1 = 0$$

According to position of second joint for pole position of our robot arm, previously this position is found as;

$$q_{21} = 2.71313$$

Then;

$$a_0 = 2.71313$$

For finding a_2, q_f value should be known, then this value equal to joint value in the second position so ;

$$q_{f1} = 1.07024$$

For finding a_2, also $\dot{q}_f$ value should be known, this value is assumed as '0.07'. This assumption is made according to motor datasheet.

Then;

$$a_2 = (-3(2.71313-1.07024) - (2 * 0 + 0.07) t_f) \ / \ t_f^{\,2}$$

$$a_3 = (2(2.71313-1.07024) + (0 + 0.07) t_f) \ / \ t_f^{\,3}$$

Third Joint

For pole position of our robot arm, velocity is assumed as 0 so ;

$$\dot{q}_{31} = 0$$

Then;

$$a_1 = 0$$

According to position of third joint for pole position of our robot arm, previously this position is found as;

$$q_{31} = -\,0.729867$$

Then;

$$a_0 = -0.729867$$

For finding a_2, q_f value should be known, then this value equal to joint value in the second position so ;

$$q_{f3} = -1.542$$

For finding a_2, also $\dot{q}_f$ value should be known, this value is assumed as '0.07'. This assumption is made according to motor datasheet.

Then;

$$a_2 = (-3(-0.729867+1.542) - (2 * 0 +0.07)\, t_f) \; / \; t_f^{\,2}$$

$$a_3 = (2(-0.729867+1.542) + (0 +0.07)\, t_f) \; / \; t_f^{\,3}$$

Fourth Joint

For pole position of our robot arm, velocity is assumed as 0 so;

$$\dot{q}_{41} = 0$$

Then;

$$a_1 = 0$$

According to position of fourth joint for pole position of our robot arm, previously this position is found as;

$$q_{41} = 2.22228$$

Then;

$$a_0 = 2.22228$$

For finding a_2, q_f value should be known, then this value equal to joint value in the second position so;

$$q_{f4} = 0$$

For finding a_2, also $\dot{q}_f$ value should be known, this value is assumed as '0.07'. This assumption is made according to motor datasheet.

Then;

$$a_2 = (-3(2.22228 - 0) - (2 * 0 + 0.07) t_f) \ / \ t_f^{\,2}$$

$$a_3 = (2(2.22228 - 0) + (0 + 0.07) t_f) \ / \ t_f^{\,3}$$

Finding Polynomial Equations For Second Position

First Joint

After finding this equation for pole position t_i can't considered as 0. So equations are became as;

$$a_0 = q_i$$

$$a_1 = \dot{q}_i$$

$$a_2 = (-3(q_i - q_f) - (2\,\dot{q}_i + \dot{q}_f) (t_f - t_i)) \ / \ (t_f - t_i)^2$$

$$a_3 = (2(q_i - q_f) + (\dot{q}_i + \dot{q}_f) (t_f - t_i)) \ / \ (t_f - t_i)^3$$

According to position of first joint for second position of our robot arm, previously this position is found as;

$$q_{12} = -0.70127$$

Then;

$$a_0 = -0.70127$$

For finding a_2, also $\dot{q}_f$ value should be known, this value is assumed as '0.07'. This assumption is made according to motor datasheet. Velocity will be equal throughout the entire movement.

Therefore;

$$\dot{q}_{12} = 0.07$$

Then;

$$a_1 = \dot{q}_{12}$$

For finding a_2, q_f value should be known, then this value equal to joint value in the third position so;

$$q_{f1} = 1.16937$$

$$a_2 = (-3(-0.70127-1.18682) - (2 * 0.07 + 0.07) (t_f - t_i)) \; / \; (t_f - t_i)^2$$

$$a_3 = (2(-0.70127 -1.18682) + (0.07 +0.07) (t_f - t_i)) \; / \; (t_f - t_i)^3$$

Second Joint

According to position of second joint for second position of our robot arm, previously this position is found as;

$$q_{22} = 1.07024$$

Then;

$$a_0 = 1.07024$$

For finding a_2, also$\dot{q}_f$ value should be known, this value is assumed as ‘0.07’. This assumption is made according to motor datasheet. Velocity will be equal throughout the entire movement.

Therefore;

$$\dot{q}_{22} = 0.07$$

Then;

$$a_1 = 0.07$$

For finding a_2 , $q_f$value should be known, then this value equal to joint value in the third position so;

$$q_{f2} = 2.17322$$

$$a_2 = (-3(1.07024 - 2.17322) - (2 * 0.15 + 0.15) (t_f - t_i)) \;/\; (t_f - t_i)^2$$

$$a_3 = (2(1.07024 - 2.17322) + (2 * 0.15 + 0.15) (t_f - t_i)) \;/\; (t_f - t_i)^3$$

Third Joint

According to position of third joint for second position of our robot arm, previously this position is found as;

$$q_{32} = -1.542$$

Then;

$$a_0 = -1.542$$

For finding a_2, also$\dot{q}_f$ value should be known, this value is assumed as ‘0.07’. This assumption is made according to motor datasheet. Velocity will be equal throughout the entire movement.

Therefore;

$$\dot{q}_{32} = 0.07$$

Then;

$$a_1 = 0.07$$

For finding a_2 , q_f value should be known, then this value equal to joint value in the third position so;

$$q_{f3} = -0.473627$$

$$a_2 = (-3(-1.542 + 0.473627) - (2 * 0.07 + 0.07) (t_f - t_i)) \ / \ (t_f - t_i)^2$$

$$a_3 = (2(-1.542 + 0.473627) + (0.07 + 0.07) (t_f - t_i)) \ / \ (t_f - t_i)^3$$

Fourth Joint

According to position of fourth joint for second position of our robot arm, previously this position is found as;

$$q_{42} = 0$$

Then;

$$a_0 = 0$$

For finding a_2, also$\dot{q}_f$ value should be known, this value is assumed as ‘0.07’. This assumption is made according to motor datasheet. Velocity will be equal throughout the entire movement.

Therefore;

$$\dot{q}_{42} = 0.07$$

Then;

$$a_1 = 0.07$$

For finding a_2 , $q_f$value should be known, then this value equal to joint value in the third position so ;

$$q_{f4}= 1.66013$$

$$a_2= (-3(0 - 1.66013) - (2 * 0.15 +0.15) (t_f - t_i)) \ / \ (t_f - t_i)^2$$

$$a_3 = (2(0 - 0.473627) + (2 * 0.15 +0.15) (t_f - t_i)) \ / \ (t_f - t_i)^3$$

Finding Polynomial Equations For Third Position

First Joint

According to position of first joint for third position of our robot arm, previously this position is found as;

$$q_{13}= 1.18682$$

Then;

$$a_0 = 1.18682$$

For finding a_2, also$\dot{q}_f$ value should be known, this value is assumed as ‘0.07’. This assumption is made according to motor datasheet. Velocity will be equal throughout the entire movement.

Therefore;

$$\dot{q}_{13}= 0.07$$

Then;

$$a_1= 0.07$$

For finding a_2 , $q_f$value should be known, then this value equal to joint value in the fourth position so;

$$q_{f1}= 1.16937$$

$$a_2= (-3(1.18682-1.16937) - (2 * 0.07+0.07) (t_f - t_i)) \ / \ (t_f - t_i)^2$$

$$a_3 = (2(1.18682-1.16937) + (0.07+0.07) (t_f - t_i)) \ / \ (t_f - t_i)^3$$

Second Joint

According to position of second joint for third position of our robot arm, previously this position is found as;

$$q_{23} = 2.17322$$

Then;

$$a_0 = 2.17322$$

For finding a$_2$, also$\dot{q}_f$ value should be known, this value is assumed as '0.07'. This assumption is made according to motor datasheet. Velocity will be equal throughout the entire movement.

Therefore;

$$\dot{q}_{23} = 0.07$$

Then;

$$a_1 = 0.07$$

For finding a$_2$, $q_f$value should be known, then this value equal to joint value in the fourth position so;

$$q_{f2} = 1.90315$$

$$a_2 = (-3(2.17322-1.90315) - (2 * 0.07 +0.07) (t_f - t_i)) \ / \ (t_f - t_i)^2$$

$$a_3 = (2(2.17322-1.90315) + (0.07 +0.07) (t_f - t_i)) \ / \ (t_f - t_i)^3$$

Third Joint

According to position of third joint for third position of our robot arm, previously this position is found as;

$$q_{33} = -0.473627$$

Then;

$$a_0 = -0.473627$$

For finding a$_2$, also $\dot{q}_f$ value should be known, this value is assumed as '0.07'. This assumption is made according to motor datasheet. Velocity will be equal throughout the entire movement.

Therefore;

$$\dot{q}_{33} = 0.07$$

Then;

$$a_1 = 0.07$$

For finding a$_2$, q_f value should be known, then this value equal to joint value in the fourth position so;

$$q_{f3} = -0.482841$$

$$a_2 = (-3(-0.473627+0.482841) - (2 * 0.07 + 0.07)(t_f - t_i)) \;/\; (t_f - t_i)^2$$

$$a_3 = (2(-0.473627+0.482841) + (0.07 + 0.07)(t_f - t_i)) \;/\; (t_f - t_i)^3$$

Fourth Joint

According to position of fourth joint for third position of our robot arm, previously this position is found as;

$$q_{43} = 1.66013$$

Then;

$$a_0 = 1.66013$$

For finding a$_2$, also$\dot{q}_f$ value should be known, this value is assumed as ‘0.07’. This assumption is made according to motor datasheet. Velocity will be equal throughout the entire movement.

Therefore;

$$\dot{q}_{43} = 0.07$$

Then;

$$a_1 = 0.07$$

For finding a$_2$, $q_f$value should be known, then this value equal to joint value in the fourth position so;

$$q_{f4} = 1.57087$$

$$a_2 = (-3(1.66013-1.57087) - (2 * 0.07 + 0.07) (t_f - t_i)) \ / \ (t_f - t_i)^2$$

$$a_3 = (2(1.66013-1.57087) + (0.07 + 0.07) (t_f - t_i)) \ / \ (t_f - t_i)^3$$

Finding Polynomial Equations for the Last Position

First Joint

According to position of first joint for fourth position of our robot arm, previously this position is found as;

$$q_{14} = 1.16937$$

Then;

$$a_0 = 1.16937$$

For finding a$_2$, also$\dot{q}_f$ value should be known, this value is assumed as ‘0.07’. This assumption is made according to motor datasheet. Velocity will be equal throughout the entire movement.

Therefore;

$$\dot{q}_{14} = 0.07$$

Then;

$$a_1 = 0.07$$

For finding a_2 , $q_f$value should be known, then this value equal to joint value in the fourth position so;

$$q_{f1} = 1.16937$$

$$a_2 = (-3(1.16937-1.16937) - (2 * 0.07 + 0.07) (t_f - t_i)) \;/\; (t_f - t_i)^2$$

$$a_3 = (2(1.16937-1.16937) + (0.07 + 0.07) (t_f - t_i)) \;/\; (t_f - t_i)^3$$

Second Joint

According to position of second joint for fourth position of our robot arm, previously this position is found as ;

$$q_{24} = 1.90315$$

Then;

$$a_0 = 1.90315$$

For finding a_2, also$\dot{q}_f$ value should be known, this value is assumed as '0.07'. This assumption is made according to motor datasheet. Velocity will be equal throughout the entire movement.

Therefore;

$$\dot{q}_{14} = 0.07$$

Then;

$$a_1 = 0.07$$

For finding a_2 , q_f value should be known, then this value equal to joint value in the fourth position so ;

$$q_{f2} = 1.90315$$

$$a_2 = (-3(1.90315-1.90315) - (2 * 0.07 + 0.07)\ (t_f - t_i))\ /\ (t_f - t_i)^2$$

$$a_3 = (2(1.90315-1.90315) + (0.07 + 0.07)\ (t_f - t_i))\ /\ (t_f - t_i)^3$$

Third Joint

According to position of third joint for fourth position of our robot arm, previously this position is found as;

$$q_{34} = -0.482841$$

Then;

$$a_0 = -0.482841$$

For finding a_2, also $\dot{q}_f$ value should be known, this value is assumed as '0.07'. This assumption is made according to motor datasheet. Velocity will be equal throughout the entire movement.

Therefore;

$$\dot{q}_{34} = 0.07$$

Then;

$$a_1 = 0.07$$

For finding a_2, q_f value should be known, then this value equal to joint value in the fourth position so;

$$q_{f3} = -0.482841$$

$$a_2 = (-3(-0.482841+0.482841) - (2 * 0.07 + 0.07)\ (t_f - t_i))\ /\ (t_f - t_i)^2$$

$$a_3 = (2(-0.482841+0.482841) + (0.07 + 0.07)\ (t_f - t_i))\ /\ (t_f - t_i)^3$$

Fourth Joint

According to position of fourth joint for fourth position of our robot arm, previously this position is found as;

$$q_{44} = 1.57087$$

Then;

$$a_0 = 1.57087$$

For finding a_2, also $\dot{q}_f$ value should be known, this value is assumed as '0.07'. This assumption is made according to motor datasheet. Velocity will be equal throughout the entire movement.

Therefore;

$$\dot{q}_{44} = 0.07$$

Then;

$$a_1 = 0.07$$

For finding a_2, q_f value should be known, then this value equal to joint value in the fourth position so;

$$q_{f4} = 1.57087$$

$$a_2 = (-3(1.57087-1.57087) - (2 * 0.07 +0.07) (t_f - t_i)) \;/\; (t_f - t_i)^2$$

$$a_3 = (2(1.57087-1.57087) + (0.07 +0.07) (t_f - t_i)) \;/\; (t_f - t_i)^3$$

5.3 NUMERICAL POLYNOMIAL FUNCTIONS

After found all parametrical polynomial functions, all numeric values should be found according to parametrical equations.

First Joint with Numerical Values

Numerical values were found by entering all values in numerically according to parametrical polynomial functions.

```
a01 = 1.0472
a02 = -0.70127
ad01 = 0
a21 = (-3 * (a01 - a02) - (2 * 0 + 0.07) 2) / 2^2
a31 = (2 * (a01 - a02) + (0 + 0.07) 2) / 2^3

Out[•]= 1.0472

Out[•]= -0.70127

Out[•]= 0

Out[•]= -1.34635

Out[•]= 0.454617
```

Figure 5.26 First Joint Motion from First Position to Second Position

Numerical values are entered in the parametrical polynomial function, then position graph is drawn according to numerical polynomial function.

```
= Plot[0.45461749999999995` t^3 - 1.3463524999999998` t^2 + 1.0472, {t, 0, 2}]
```

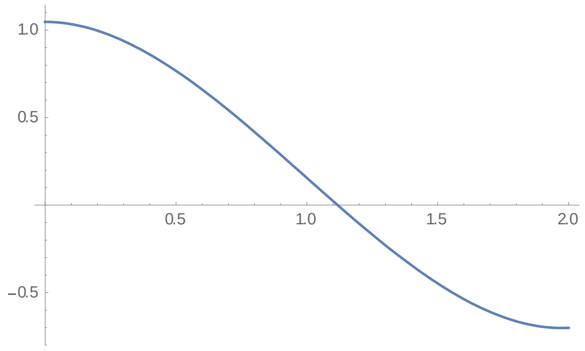


Figure 5.27 Position Change Graph

The position graph of the first position at each position was found with the same method and checked to see if it was following each other.

In the first graph the movement of the first joint from the first position to the second position starts at 1.0472 and ends at -0.70127, and these values can be seen on the graph.

In the second graph the movement of the first joint from the second position to the third position should start at -0.70127 and end at 1.18682, and these values can also be seen on the graph.

Therefore, it was concluded that the first two movements followed each other. This compare shown below;

```
In[ ]:= Plot[-0.4370225` t^3 + 1.3110675` t^2 + 0.07 t - 0.70127, {t, 0, 2}]
```

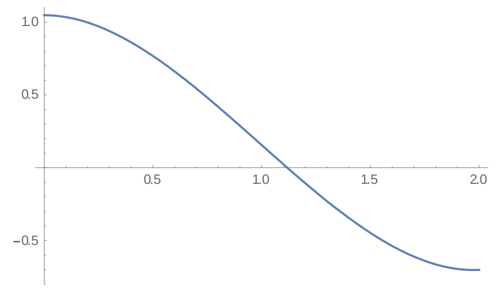


Figure 5.28 First to Second Position

```
In[ ]:= Plot[0.039362499999999995` t^3 - 0.11808749999999998` t^2 + 0.07 t + 1.18682`, {t, 0, 2}]
```

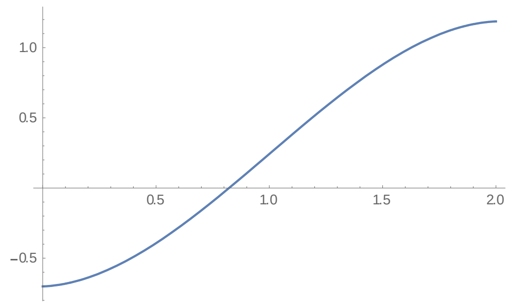


Figure 5.29 Second to Third Position

In the third graph the movement of the first joint from the third position to the fourth position starts at 1.18682 and ends at 1.16937, and these values can be seen on the graph.

In the fourth graph the movement of the first joint from the fourth position to the fourth position which is made for to get zero to joint velocities should start at 1.16937 and end at 1.16937, and these values can also be seen on the graph.

Therefore, it was concluded that the third movement followed the second and the fourth movement followed the third thank to graphs which are drawn according to numerical polynomial functions. This compare shown below;

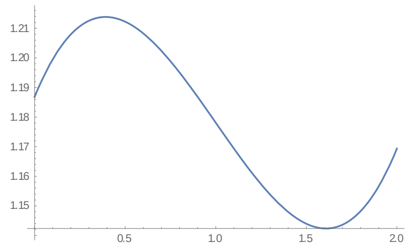


Figure 5.30 Third to Fourth Position

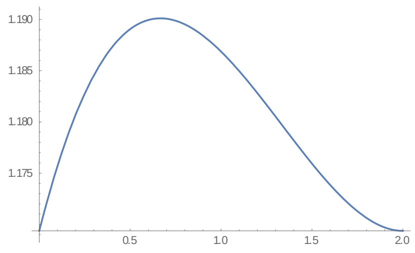
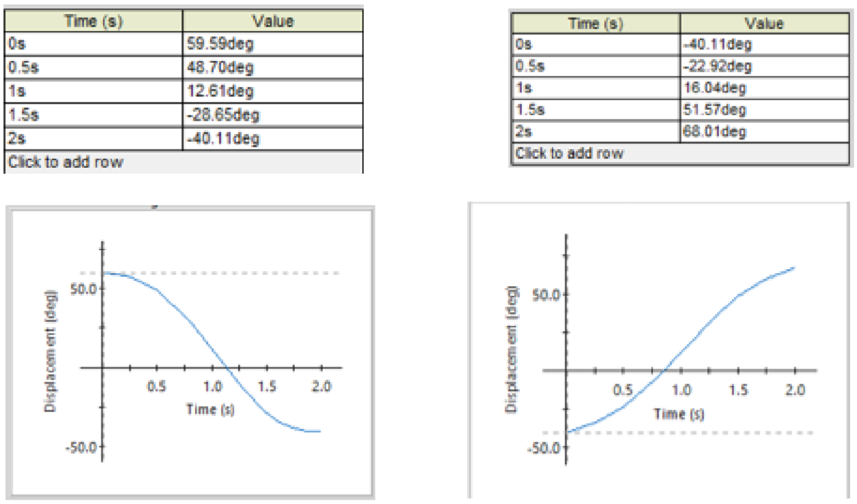


Figure 5.31 Stop Motion of the Robot Arm

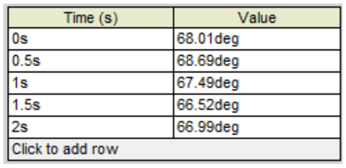
These data were entered as a SolidWorks data points for using motion simulation of our robot arm. All values are converted from radians to degrees.



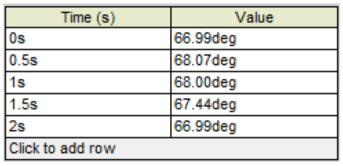
Time (s)	Value
0s	59.59deg
0.5s	48.70deg
1s	12.61deg
1.5s	-28.65deg
2s	-40.11deg
Click to add row	

Time (s)	Value
0s	-40.11deg
0.5s	-22.92deg
1s	16.04deg
1.5s	51.57deg
2s	68.01deg
Click to add row	

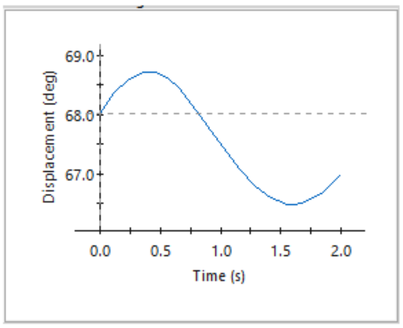
Figure 5.32 SolidWorks Data Entries



Time (s)	Value
0s	68.01deg
0.5s	68.69deg
1s	67.49deg
1.5s	66.52deg
2s	66.99deg
Click to add row	



Time (s)	Value
0s	66.99deg
0.5s	68.07deg
1s	68.00deg
1.5s	67.44deg
2s	66.99deg
Click to add row	



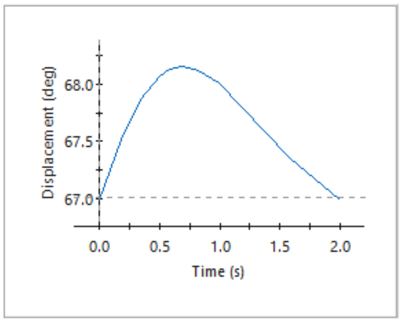


Figure 5.33 SolidWorks Data Entries for other joints

After, finding first joint position graphs for each position change, velocity change graphs is found by taking derivative of the third order polynomial function. Initially the velocity will be 0, then the velocity rises to constant speed of 0.07.

```
        D[0.45461749999999995` t^3 - 1.3463524999999998` t^2 + 1.0472, t]
Out[-]= -2.6927 t + 1.36385 t^2
 In[-]= Plot[-2.6927049999999997` t + 1.3638525` t^2, {t, 0, 2}]
```

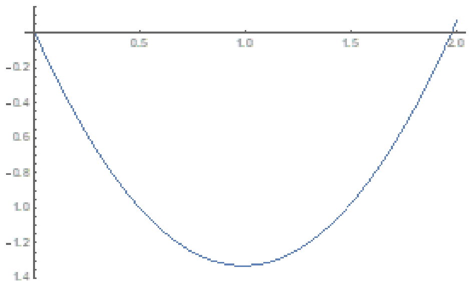
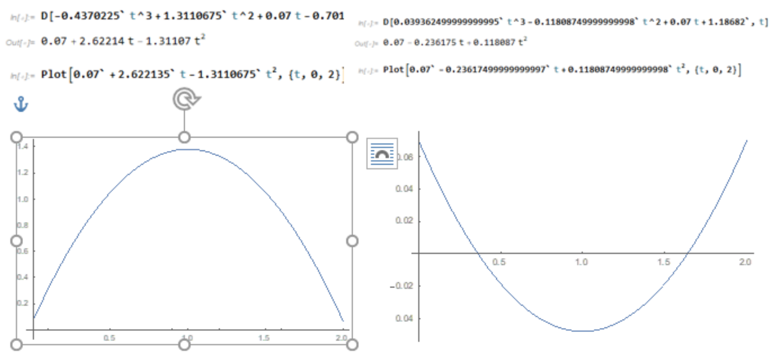


Figure 5.34 First Joint Velocity From First Position to Second Position

After the velocity rised to constant speed of 0.07, it will be equal throughout the entire movement as 0.07.



```
In[-]:= D[-0.4370225` t^3 + 1.3110675` t^2 + 0.07 t - 0.701
Out[-]= 0.07 + 2.62214 t - 1.31107 t²

In[-]:= Plot[0.07` + 2.622135` t - 1.3110675` t², {t, 0, 2}]
```

```
In[-]:= D[0.039362499999999995` t^3 - 0.11808749999999998` t^2 + 0.07 t + 1.18682`, t]
Out[-]= 0.07 - 0.236175 t + 0.118087 t²

In[-]:= Plot[0.07` - 0.23617499999999997` t + 0.11808749999999998` t², {t, 0, 2}]
```

Figure 5.35 Constant Velocity of the Second and Third Position Change

Finally, the velocity will decrease to 0 again and the motion will end.

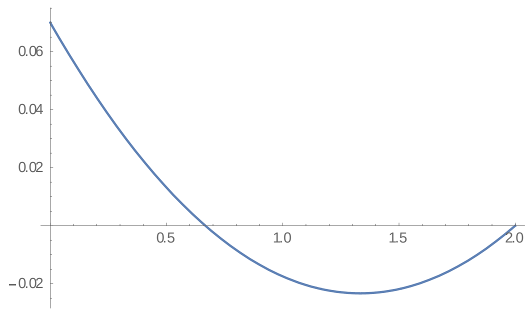
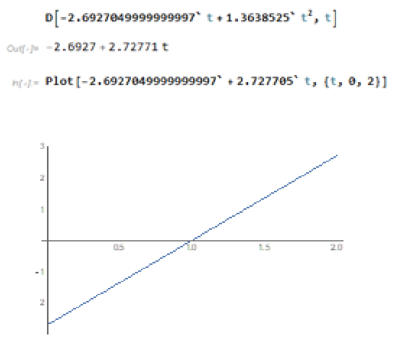


Figure 5.36 Decrease to the ‘0’ Velocity

Finally, after finding first joint velocity graphs for each position change, acceleration change graphs is found by taking derivative of the velocity polynomial functions.



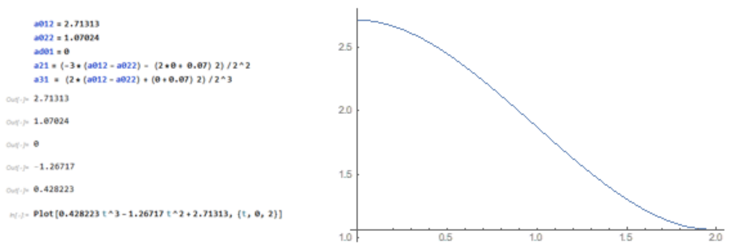
```
D[-2.6927049999999997` t + 1.3638525` t², t]
Out[-]= -2.6927 + 2.72771 t

In[-]:= Plot[-2.6927049999999997` + 2.727705` t, {t, 0, 2}]
```

Figure 5.37 Acceleration Change of the Figure 5.28

Acceleration calculation was made for each position change.

Joint Motions for First Position to Second Position

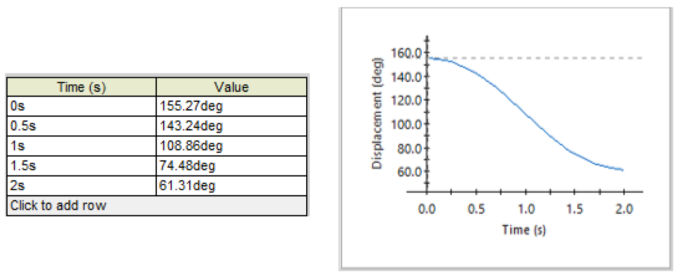
For second joint in first position to second position;



```
         a012 = 2.71313
         a022 = 1.07024
         ad01 = 0
         a21 = (-3 * (a012 - a022) - (2 * 0 + 0.07) 2) / 2^2
         a31 = (2 * (a012 - a022) + (0 + 0.07) 2) / 2^3
Out[-]= 2.71313
Out[-]= 1.07024
Out[-]= 0
Out[-]= -1.26717
Out[-]= 0.428223
 In[-]:= Plot[0.428223 t^3 - 1.26717 t^2 + 2.71313, {t, 0, 2}]
```

Figure 5.38 Second Joint From First Position to Second Position

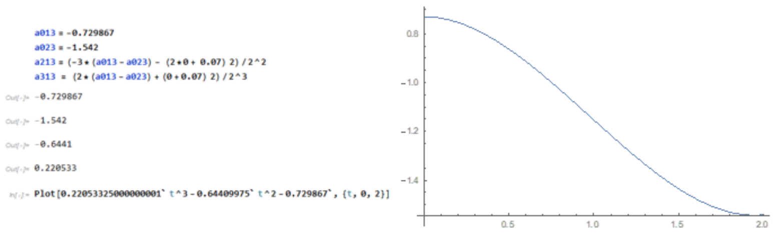
All numerical data were converted from radians to degrees, then entered as simulation data in SolidWorks and the same position graphs were obtained.



Time (s)	Value
0s	155.27deg
0.5s	143.24deg
1s	108.86deg
1.5s	74.48deg
2s	61.31deg
Click to add row	

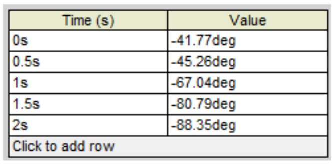
Figure 5.39 SolidWorks Motion of the Figure 5.38

For third joint in first position to second position;



```
         a013 = -0.729867
         a023 = -1.542
         a213 = (-3 * (a013 - a023) - (2 * 0 + 0.07) 2) / 2^2
         a313 = (2 * (a013 - a023) + (0 + 0.07) 2) / 2^3
Out[-]= -0.729867
Out[-]= -1.542
Out[-]= -0.6441
Out[-]= 0.220533
 In[-]:= Plot[0.22053325000000001` t^3 - 0.64409975` t^2 - 0.729867`, {t, 0, 2}]
```

Figure 5.40 Third Joint From First Position to Second Position



Time (s)	Value
0s	-41.77deg
0.5s	-45.26deg
1s	-67.04deg
1.5s	-80.79deg
2s	-88.35deg
Click to add row	

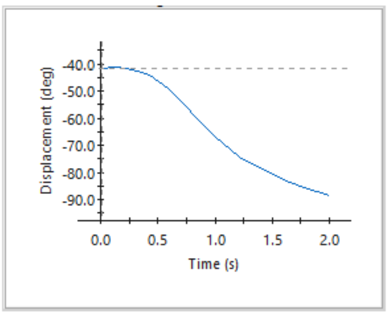
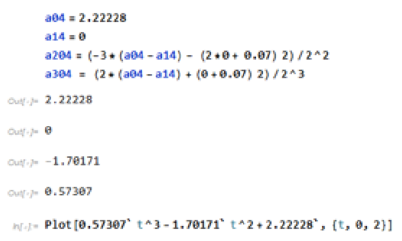


Figure 5.41 SolidWorks Motion Data of the Figure 5.40

For the fourth joint in first position to second position;



```
a04 = 2.22228
a14 = 0
a204 = (-3*(a04 - a14) - (2*0+ 0.07) 2) / 2^2
a304 = (2*(a04 - a14) + (0+0.07) 2) / 2^3
Out[-]= 2.22228
Out[-]= 0
Out[-]= -1.70171
Out[-]= 0.57307
In[-]:= Plot[0.57307` t^3 - 1.70171` t^2 + 2.22228`, {t, 0, 2}]
```

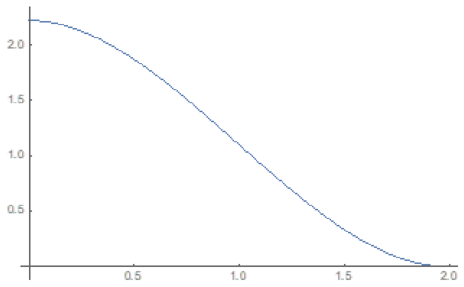
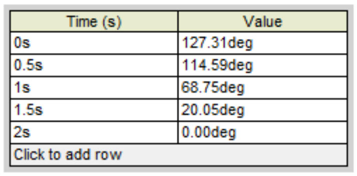


Figure 5.42 Fourth Joint From First Position to the Second Position



Time (s)	Value
0s	127.31deg
0.5s	114.59deg
1s	68.75deg
1.5s	20.05deg
2s	0.00deg
Click to add row	

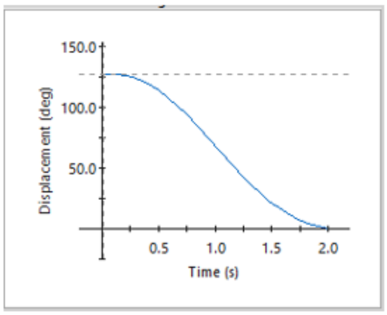


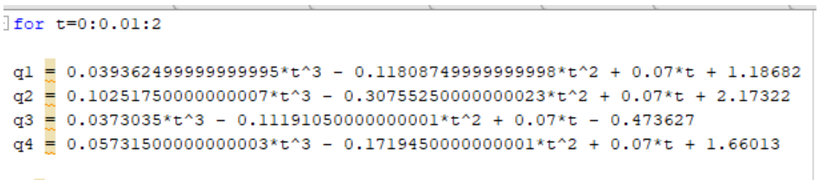
Figure 5.43 SolidWorks Motion Data of Figure 5.42

All these procedures were repeated for each position change of each joint. Then each of them was entered into the SolidWorks as simulation data.

5.4 Path Drawing

The waypoints obtained from the workspace analysis were checked on the graph which is drawn by using polynomial equations that was found with cubic polynomial function method.

The polynomial equations obtained after the correctness of our position, velocity and acceleration graphs were checked in the workspace analysis by entering these polynomial equations into the matlab code in a for loop as ;



```
for t=0:0.01:2

 q1 = 0.039362499999999995*t^3 - 0.11808749999999998*t^2 + 0.07*t + 1.18682
 q2 = 0.10251750000000007*t^3 - 0.30755250000000023*t^2 + 0.07*t + 2.17322
 q3 = 0.0373035*t^3 - 0.11191050000000001*t^2 + 0.07*t - 0.473627
 q4 = 0.05731500000000003*t^3 - 0.17194500000000001*t^2 + 0.07*t + 1.66013
```

Figure 5.44 Data Entry in Matlab as Polynomial Function

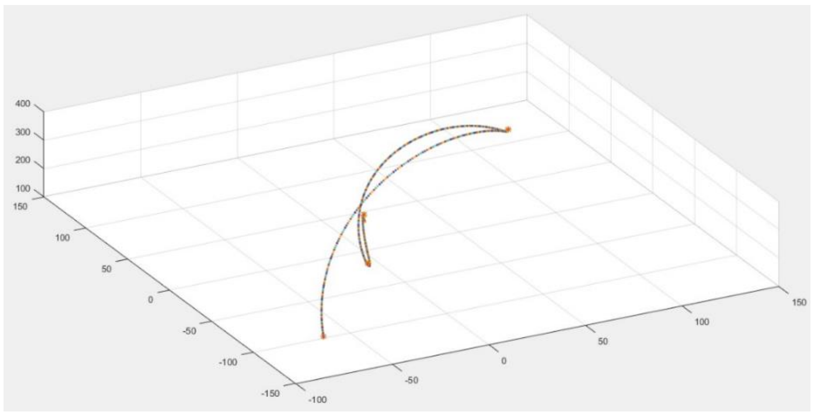


Figure 5.45 Path Drawing of our Robot Design for Four Position

5.5 Position Change of the Robot Arm

5.5.1 Position Simulation in SolidWorks

Pole position of our robot arm shown below;

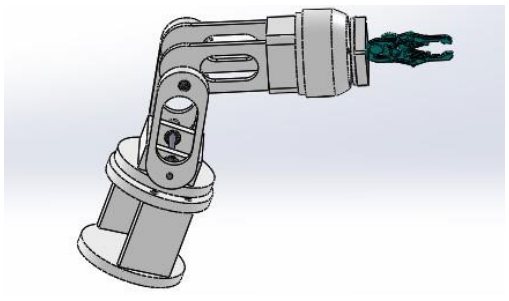


Figure 5.16 The Pole Position to Second Position SolidWorks Simulation

After pole position to second position, each joint make movement. Below, the robot takes the piece.

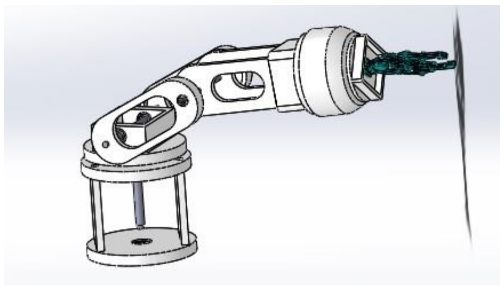


Figure 5.17 The Second Position to Third Position SolidWorks Simulation

After second position to third position, Below is the middle waypoint of the robot.

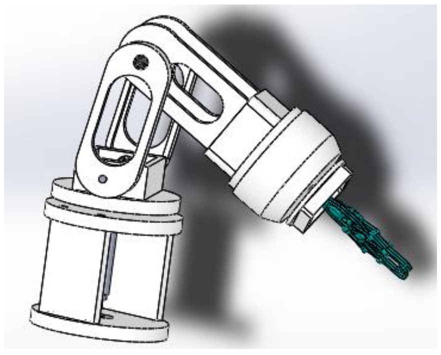


Figure 5.18 The Third Position to Fourth Position SolidWorks Simulation

After third position to fourth position, Below, the robot leaves the piece on the table.

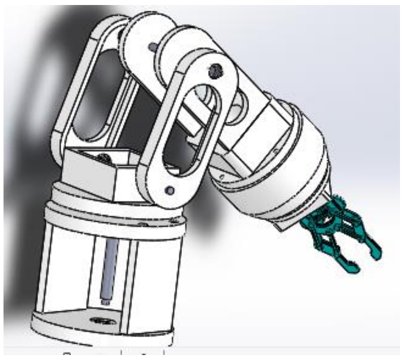


Figure 5.19 The Last Position SolidWorks Simulation

5.5.2 Position Simulation in the Produced Robot

After the robot parts are produced and assembled, its electrical circuit has been done, by using python script in the Raspberry Pi which is connected to the compound mobile and serial robot and C# WinForm tool executing in the PC, compound mobile and serial robot and its joints are controlled.

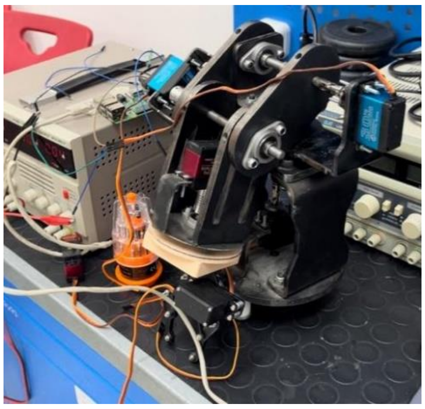


Figure 5.20 Initial position of the robot arm

After robotic arm setup is done, joints were controlled and position is changed as shown below to the second position.

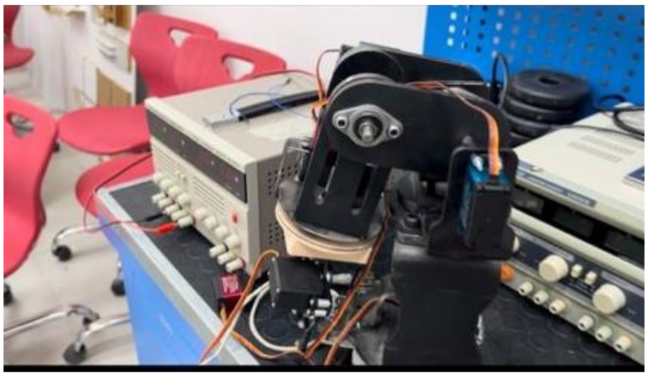


Figure 5.21 Second Position of the robot arm

After robot is reached to second position, it can be stand in there, by running the scripts and controlling robot from the C# WinForm tool manually or automatically by the inverse kinematics, robot were moved to the final position which was planned to go.

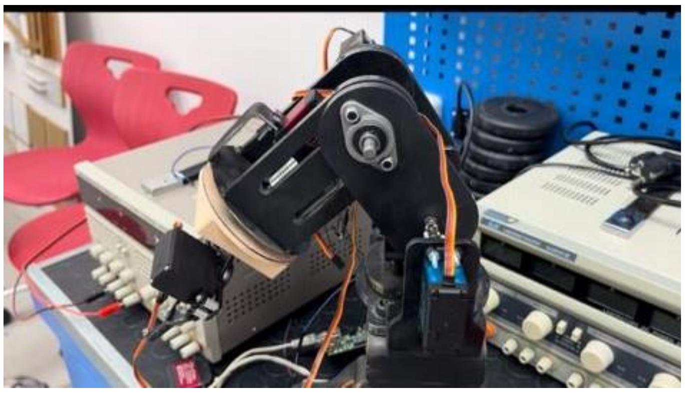


Figure 5.22 Final Position of the robot arm

6 References

[1] Duygu Atcı, Alpay Toprak, 4. Baskent International Conference on Multidisciplinary Studies, Abstract Book, Design and Remote Control of a Compound Mobile Serial Robot, August 4-6, 2023, Ankara, Türkiye, Page Number 109

[2] Vedran Vajnberger, Tarik Terzimehić, Semir Silajdžić and Nedim Osmić Faculty of Electrical Engineering, Department of Automatic Control and Electronics, Sarajevo, Bosnia and Herzegovina, Remote Control of Robot Arm with five DOF, 2011

[3] Li-Hu Jhang, Carlo Santiago and Chian-Song Chiu Department of Electrical Engineering, Chung Yuan Christian University, Taiwan, ROC. Multi-Sensor Based Glove Control of An Industrial Mobile Robot Arm

[4] R. Kazala, A. Taneva, M. Petrov, St. Penkov, Department of Industrial Electrical Engineering and Automatics, Kielce University of Technology, Kielce, Poland, Wireless Network for Mobile Robot Applications

[5] Lung-Wen Tsai, Robot Analysis: The Mechanics of Serial and Parallel Manipulators, John Wiley and Sons, 1999.

[6] Datasheet of the Motor, MG996R High Torque Metal Gear Dual Ball Bearing Servo This High-Torque MG996R

[7] Datasheet of the Motor, DS3230 - digital servo, https://kamami.pl/en/servos-standard/583003-ds3230-digital-servo.html

[7] Datasheet of the Motor, DS3225, Dongguan City Dsservo Technology Co.Ltd https://www.dsservo.com/en/d_file/DS3225%20datasheet.pdf

7 Curriculum Vitae

Name Surname: Alpay Toprak

Education:

2015–2020	İzmir Kâtip Çelebi University, Bachelor’s Degree, Dept. of Mechatronics Engineering
2018-2019	West Pomeranian University of Technology, Dept of Mechatronics Engineering with Erasmus+ Program
2020–2023	İzmir Kâtip Çelebi University, Master’s Degree, Dept. of Robotics Engineering

Work Experience:

March 2020 – April 2020	Almanyalı Diesel-Test Pump, Test Engineer
October 2020 – April 2021	Tolkar Makina ve Sanayi Ticaret A.Ş, Software Tool Development Engineer
April 2021 – April 2023	Borgwarner Inc., Systems/Software Tool Development Engineer
April 2023 – Ongoing	Vestel A.Ş, Senior Automotive Software Specialist

Publication:

Duygu Atcı, Alpay Toprak, 4. Baskent International Conference on Multidisciplinary Studies, Abstract Book, Design and Remote Control of a Compound Mobile Serial Robot, August 4-6, 2023, Ankara, Türkiye, Page Number 109