DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

CONTROL OF A TRIPTERON ROBOT BY USING ARTIFICIAL NEURAL NETWORKS

by
Serdar SAYIN

August, 2019
İZMİR

CONTROL OF A TRIPTERON ROBOT BY USING ARTIFICIAL NEURAL NETWORKS

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
In Mechatronics Engineering

by

Serdar SAYIN

August, 2019

İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"CONTROL OF A TRIPTERON ROBOT BY USING ARTIFICIAL NEURAL NETWORKS"** completed by **SERDAR SAYIN** under supervision of **PROF.DR. ZEKİ KIRAL** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Zeki KIRAL

Supervisor

Dr. Öğr. Üyesi Şahin YAVUZ

(Jury Member)

Dr. Öğr. Üyesi Özgün BAŞER

(Jury Member)

Prof. Dr. Kadriye ERTEKİN

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

I would like to thank Prof. Dr. Zeki KIRAL for their kindness and support from the beginning of the thesis as mentor. Since the study required a lot of work time and cooperation either in office or laboratory, I would like to thank for their time and support at every steps of the study.

Finally, I would like to thank to my family for their patience and support during the time that had to be passed in thesis study.

Serdar SAYIN

CONTROL OF A TRIPTERON ROBOT BY USING ARTIFICIAL NEURAL NETWORKS

ABSTRACT

Developing technologies of today is aimed to be designed in new ways of control algorithms such as artificial neural networks. In this study, a tripteron robot was designed and manufactured to be controlled by an artificial neural network algorithm.

When the study considered in two main different sections, the first part of it would be the CAD design part in SolidWorks. After all the researches about the robot, the last design was built which was easy to assemble, cheap enough to be manufactured and sufficient to be used to test the developed algorithms. The second section of the study includes all the software of the robot. Two main programs being used were Matlab and Arduino IDE. The Matlab was firstly responsible to apply the determined neural network algorithm to convert the end effector position data to the linear guide position data. The determined artificial neural network algorithm was simple adaptive linear combiner. Secondly, the Matlab was responsible to calculate the required simultaneous stepper motor driving data by the other developed algorithms. Arduino microcontroller was used to drive the stepper motors according to the data which calculated on Matlab and stored in a micro SD card. While the Arduino can drive digital signals through only serial ports, the stepper motors were successfully driven simultaneously to achieve the most realistic robotic motion.

Keywords: Tripteron robot, LMS algorithm, open loop control

BİR TRİPTERON ROBOTUN YAPAY SİNİR AĞLARI İLE KONTROLÜ

ÖZ

Günümüzün gelişen teknolojilerinin yapay sinir ağları gibi yeni kontrol algoritmalarıyla tasarlanması amaçlanmaktadır. Bu çalışmada da bir tripteron robot yapay sinir ağlarıyla kontrol edilmek üzere tasarlanmış ve üretilmiştir.

Çalışma iki farklı ana kısma ayrıldığı zaman ilk bölüm SolidWorks'te yapılmış olan CAD tasarım bölümüdür. Robot hakkında yapılan bütün araştırmalardan sonra montajı kolay, üretimi ucuz ve geliştirilen algoritmaların test edilebileceği kadar yeterli olan son tasarım oluşturulmuştur. Çalışmanın ikinci kısmı robotun bütün yazılımını içermektedir. Kullanılmış olan iki program Matlab ve Arduino IDE'dir. Matlab ilk olarak manipülatör uç nokta pozisyon verisini lineer kızak pozisyon verisine dönüştürmek için belirlenen sinir ağı algoritmasını uygulamaktan sorumludur. Belirlenen yapay sinir ağı algoritması basit uyarlanabilir doğrusal birleştiricidir. İkinci olarak Matlab, diğer geliştirilen algoritmalarla gerekli olan eş zamanlı adım motor sürme verilerini hesaplamaktan sorumludur. Arduino mikrodenetleyicisi, Matlab'de hesaplanan ve bir SD kartta depolanan verilere göre adım motorları sürmek için kullanılmıştır. Arduino dijital sinyalleri sadece seri kanaldan gönderebiliyorken, adım motorlar en gerçekçi robotik hareketi elde etmek için başarılı bir şekilde eş zamanlı olarak sürülmüştür.

Anahtar Kelimeler: Tripteron robot, LMS algoritması, açık çevrim kontrol

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

The first mention to the robot name was made by Karel Capek at the play RUR (Rossum's Universal Robots) in 1921. The description made for the robot was a computerized machine devised to respond surroundings. Isaac Asimov was the creator of the first three laws of robotics that should be considered while studying. The first three laws of robotics have been defined as follows (Asimov, 1950):

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

George Devol whose name was encountered on the patent in 1954 invented the first robotic arm and this patent was granted in 1961. Today, industrial robots are in use at almost every factory. Robotic technologies make production easier, safer and a lot faster than before so the human health and life can be protected while manufacturing. The effects of the mechanization can be seen at the industrial age's textile factories. Those days machines were powered by steam rather than electricity, however the results of the mechanization was obvious on production levels. The mechanization followed by the programmable and smart robotic systems are shaping the future of the industry.

The programmable robotic manipulators can achieve any given task continuously and without any failures rather than human workforce. There are a lot of different tasks that today's robots are responsible to do such as painting, welding, assembling, carrying high loads. Therefore, the robotic technology has wide range of use at different business sectors such as automotive, logistic, consumer goods and food industry.

Robotic Manipulators can be categorized in two different groups such as serial and parallel manipulators. Even though serial manipulators have larger workspace and more flexible movement capacity, today parallel manipulators became popular for common use. The development of the parallel manipulator can be dated back to 1960s. The first designed test system was Gough and Whitehall's six-linear jack system as a universal tire testing machine. After that, a flight simulator platform was developed by Stewart. There are many different applications that parallel manipulators can be used such as mining machines, space applications, material handling and machine tools. Parallel manipulators offer several different advantages and disadvantages such as:

- Higher load carrying capacity by sharing the total load to several links
- Higher structural stiffness
- Lower inertia forces
- Lower error sensitivity for certain errors
- Built-in redundancy
- Smaller effectively used workspace
- Physical constrains of universal and spherical joints
- Platform Singularities

The kinematic equations of parallel manipulators can be considerably more complex than the serial manipulators because of their closed loop structure. These properties shows that the parallel manipulators can be preferred in precise positioning applications in contrast to the serial manipulators (Patel & George, 2012). The first designed and patented industrial parallel manipulator is shown in Figure 1.1.

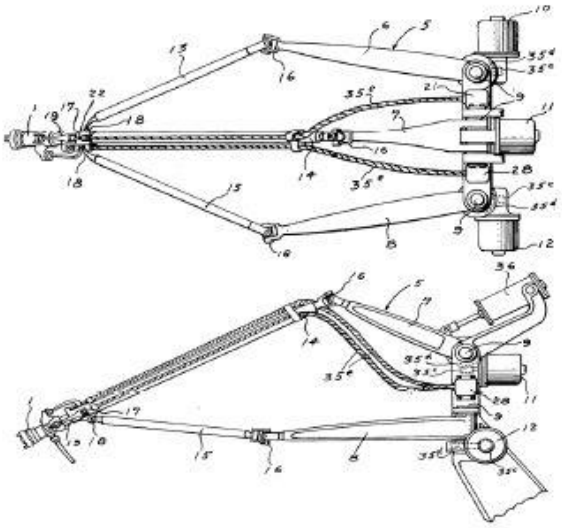


Figure 1.1 First patented industrial parallel robot (Patel & George, 2012)

There are different programming platforms to develop a control algorithm. Matlab is a specially designed programming platform for engineers and scientists with its matrix based language. The program helps to analyze data, develop algorithms and create models. By using Matlab, it is possible to do programming either by casual text editor or GUI based simulator.

The board that is used to control any robotic system can be called microcontroller board. The microcontroller board can be defined as a single printed board that includes all the required circuitry for any control task. Arduino is an open source platform based on hardware and software. There are various types of Arduino microcontrollers on the market to be used in different robotic applications. Today, Arduino Mega is one of the most popular and highly capable microcontroller board to handle complex tasks. The programming platform that Arduino offers is Arduino IDE. The Arduino IDE is an open source text editor based platform that helps developers to code. By using Arduino it's possible to control any digital or analog signal through its input and output ports.

The developing technologies of today lead to new computation techniques. The Artificial Neural Networks (ANN) has been inspired by the human brain. The human brain is a highly complex, nonlinear and parallel computer that can perform any given

task faster than any conventional digital computer of today. Neural Networks offers several properties and capabilities such as:

- Nonlinearity: An artificial neuron can be linear or nonlinear. Nonlinearity is an important property especially to process nonlinear inputs such as speech signal.
- Input-Output Mapping: Thus, ANN can perform any desired classification task of the given input signal according to the data already processed while learning.
- Adaptivity: Neural networks have ability to adapt the changes at the surrounding environment by updating its synaptic weights.
- Fault Tolerance: Because of the distributed structure of the artificial neural network, when a neuron or link is damaged, the damage has to be great to change to overall response of the network.

Overall, ANN can organize components called neurons to perform tasks such as pattern recognition, visual processing, perception and motor control (Haykin, 2009).

1.1 Tripteron Robot Manipulator

The complexity of parallel manipulators divert the industry to search of new low cost manipulators that can handle several simple manipulation or manufacturing tasks. Most of the machines introduced over the last decade were based on translational 3 degrees of freedom parallel mechanism called Delta robot. The large proportion of the robots at the industry in use have 3 or 4 degrees of freedom, particularly Cartesian and SCARA architecture are widely used. Since the Delta robot which proposed by Clavel, many different parallel mechanisms have been invented. The mechanism group called Multipteron has different variations such as 3, 4, 5 or 6 degrees of freedom. Tripteron Robot, is a completely decoupled translational closed loop mechanism which has 3 degrees of freedom. The mechanism has three legs joined orthogonally to a shared platform. The joints at each legs are PRRR (prismatic-revolute-revolute-revolute) type. Prismatic joint is parallel to the axes of the revolute joints. Each linear actuator can control one of the three translational movement, therefore the mechanism is fully decoupled (Gosselin, Masouleh, Duchaine, Richard, Foucault & Kong, 2007).

Since all the revolute joints are passive at each leg while prismatic joints are being driven by actuators, the kinematic equations of the tripteron robot are trivial, therefore the displacement of the end effector and the input joint is same in each axes (Elkady, Elkobrosy, Hanna & Sobh, 2008). Noticeable features of the Tripteron robot can be listed as:

- No singularity in workspace, due to its translational movement
- Easy to control, due to the simplicity of the kinematics
- Lower inertia forces and high acceleration capacity, due to the location of the actuator on the robot
- Rigid body structure, due to the closed loop system
- Small workspace, due to the link interference
- No equally distribution of the load to several arms

According to the given informations, it can be said that the tripteron robot is highly suitable for fast pick and place applications (Patel & George, 2012).

1.2 Literature Review of Control on Parallel Manipulators

Since the parallel manipulators are highly demanded by the industry according to their advantages at specific applications, there are many researches about the control of the parallel manipulators. The studies given as literature review includes various control methods such as PID, ANN, sliding mode control and fuzzy logic.

Stewart platform (Hexapod) is a highly popular mechanism to be used as flight simulator for aerospace applications due to the characteristics of the mechanism, however the complex kinematics of the mechanism requires high computation power. Since the artificial neural networks methods are suitable to be used on nonlinear highly complex structures, a closed loop control system based on the ANN can be designed to control a Stewart Platform at high precision values and faster response times (Zhukov, Korotkov, Moroz, Zhukova & Abramov, 2018). Hexapod mechanism studied in that paper is given in Figure 1.2.

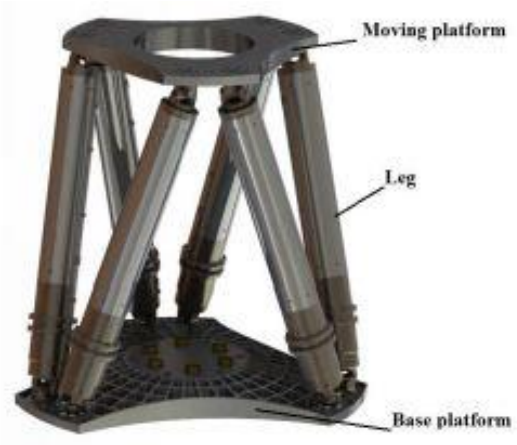


Figure 1.2 Hexapod Mechanism (Zhukov et al., 2018)

Since the Stewart Platform is highly preferred mechanism in such specific application, many researches have been made on the control methods. Hexapod is a parallel mechanism with 6 degrees of freedom. The forward kinematics problem (FKP) of the mechanism requires nonlinear equations to be solved. The traditional methods are commonly used to find the roots of the nonlinear equations, however the FKP is not fully solved just by finding all the possible solutions. Dehghani et al. recommended ANN algorithms in their study to solve the FKP of the Hexapod mechanism (Dehghani, Ahmadi, Khayatian, Eghtesad & Farid, 2008).

Since the PID control is a simple and reliable control method, it has extensive usage on motion control. The sliding mode control (SMC) was a modern nonlinear robust control method with its simplicity, convenience of realization, decoupling of the control object and the system cost reduction. In dynamic process of robot with parallel mechanism however, SMC produces high frequency chattering and this restricts the practical application of the SMC. Neural network control and fuzzy logic which don't depend on mathematical models directly can solve the uncertainty problems of the system (Xu, Wang & Lin, 2018).

Intelligent control methods can be applied to closed loop control system of highly nonlinear parallel manipulator to track any planned Cartesian trajectory in the presence of disturbances. Noshadi et al. used intelligent two level fuzzy tuning resolved

acceleration control method with active force control method (FLRAC-AFC) to acquire the most precise position control (Noshadi, Mailah & Zolfagharian, 2012).

Singh et al. introduced a control algorithm based on computer torque control integrated with a disturbance observer (Singh, Vinoth & Santhakumar, 2014). The studied mechanism was a 3-degree of freedom planar parallel manipulator with RPR configuration. The disturbance vector includes parameter uncertainties, variations on payload and effects of the friction. Other disturbances were determined by nonlinear disturbance observer with extended Kalman filter. The success of the made control loop was presented by comparing the results with traditional controller such as PID controller and only computed torque controller.

Depending on the manufacturing errors some kinematic uncertainties may occur between the theoretical design and the reality. Heydarzadeh et al. studied on control system of a fully decoupled 3 degrees of freedom parallel robot (Heydarzadeh, Karbasizadeh, Masouleh & Kalhor, 2018). The closed loop control system was achieved by using an infrared vision sensor called Kinect sensor on the determination of the end effector's spatial position. The calibration of the Kinect sensor was made by LoLiMoT algorithm which was a neuro-fuzzy based algorithm.

Since the Cartesian Parallel Manipulator (CPM) doesn't suffer from the complex kinematic model of parallel manipulators and high manufacture costs of precise joints, many studies are focused on this mechanism. Elkady et al. studied modeling, control and simulation of control parameters such as actuator force, position error and velocity error along the axes (Elkady, Elkobrosy, Hanna & Sobh, 2008). Since the model based control was more complex and difficult to be implemented because of the good model requirement and nicely estimated model parameters, they implemented a PID control.

Time varying load inertia is one of the main dynamic characteristic of parallel manipulators in joint space particularly while in acceleration or deceleration phases. L. Wang et al. presented a total dynamic performance analysis method by considering

the load inertia time varying characteristics, which heavily affects the performance of the system, and by using two inertia system (Wang, Wang & Wu, 2019).

The parallel manipulators may suffer from the geometric uncertainties, which affects the performance of the manipulator during motion planning. Reliable motion planning algorithms were necessity for parallel manipulators and very few algorithms were developed. Vieira et al. proposed an algorithm that includes Monte Carlo simulation method and an artificial neural network model to overcome the computational inefficiency of the Monte Carlo method on the possible failure estimation (Vieira, Wajnberg, Beck & Silva, 2019).

Parallel manipulators can be used in precise positioning applications due to their robust and stationary body structure. In mobile robot applications, some sensors need to be aimed directly on a specific target to achieve to most efficient measurement. Hüllmann et al. studied on a parallel manipulator with 3 degrees of freedom to perform that critical task (Hüllmann, Kohlhoff, Erdmann & Neumann, 2019). PID control method was used to generate control signals.

In robotic systems, increment in joint tracking accuracy can be effectively achieved by feedforward control. Liu et al. presented an approach for feedforward control parameter tuning of parallel mechanisms that consider joint crosstalk (Liu, Xiao, Yang, Liu, Huang & Chetwynd, 2019).

Due to their characteristics, parallel manipulators are in the scope of many studies especially the robots with outputs as a translation and two rotations (1T2R). 3PRS parallel mechanism is a typical example of this group, however 3PRS mechanism has unusual kinematic characteristic and low orientation capability. Herrero et al. have studied on 2PRU-1PRS mechanism which have several advantages such as higher capability of orientation and better kinematics (Herrero, Pinto, Altuzarra & Diez, 2018).

The first step of selecting an appropriate error model is the accuracy analysis of parallel manipulators. Ghazi et al. studied on the calculation of the kinematic accuracy of a delta parallel mechanism by jacobian and geometric error models (Ghazi, Nazir, Butt & Baqai, 2018). Jacobian error model was used for analysis and optimum design of the manipulator while the geometric error model was performed to calculate the exact positioning errors. The obtained result of the study was the geometric error model, which was capable of calculating exact positioning error value for known error in active joint input while the Jacobian error model was not capable of giving the exact error value.

In the case of high accuracy, speed and stiffness parallel manipulators show great advantages. While the design stage of a parallel manipulator it can be difficult to determine the appropriate dimensions of the structure and the workspace of the robot. Kinematic optimal design is crucial while designing a parallel manipulator and it would better to be aware of how good the mechanism works while in the design stage (Hamdoun, Bakkali & Baghli, 2017).

Nag et al. presented a comprehensive study on the forward kinematic problem of the 3-RPRS parallel mechanism which has three legs with two actuators in each leg. The three legs connect a triangular platform to a fixed base (Nag, Mohan & Bandyopadhyay, 2017).

Yang et al. introduced family of extendable kinematic chains, called dual scissor-like mechanisms. The chains have two translational degrees of freedom and they were constructed by linking two identical scissor-like element. The mobility and singularities of the chains were discussed in the paper (Yang, Peng, Pu, Chen, Ding, Chirikjian & Lyu, 2019).

Stigger et al. proposed a comprehensive examination on input and output singularities of a 3-RUU parallel mechanism during translational move. The study was made by using algebraic constraint equations (Stigger, Siegele, Scharler & Pfurner, 2019).

End effector position of a robot manipulator can be solved by artificial neural network algorithms rather than known computation method as forward kinematic analysis to decrease the required calculation time and effort. Yavuz and Hocaoğlu presented a neural network solution for forward kinematics of a serial robot manipulator with two degrees of freedom (Yavuz & Hocaoğlu, 2015).

In this study, a tripteron robot was designed by using SolidWorks modeling and analysis software and manufactured by using the standard automation parts. The control of the robot was achieved by using a simple artificial neural network algorithm. A stand-alone Matlab program was developed to create the end effector positions, which are fetched from an SD card via Arduino IDE. The thesis is organized as follows:

Chapter two explains the details of design, analysis and manufacturing stages.

Chapter three provides the details of the control software and the developed computer programs.

Chapter four comprises general results presented in this thesis and presents some future recommendations.

CHAPTER TWO
DESIGN, ANALYSIS AND PRODUCTION

2.1 Design and Analysis

In the design process, two different designs have been made. Common point of the designs were to have a robot which was easy to assemble, cheap enough to be manufactured and sufficient to be used to test the developed algorithms. Tripteron robot has three legs attached to a platform and three actuators based on a stable body to drive each of the legs separately in order to do any given task in the workspace. Since the robot has translational movement along three different axes with PRRR joint configuration and the correlation between the end effector and input joint is one to one, prismatic joints are the only joints that must be considered on trajectory planning and the other joints are almost passive. The two different designs are given in Figure 2.1 and Figure 2.2.

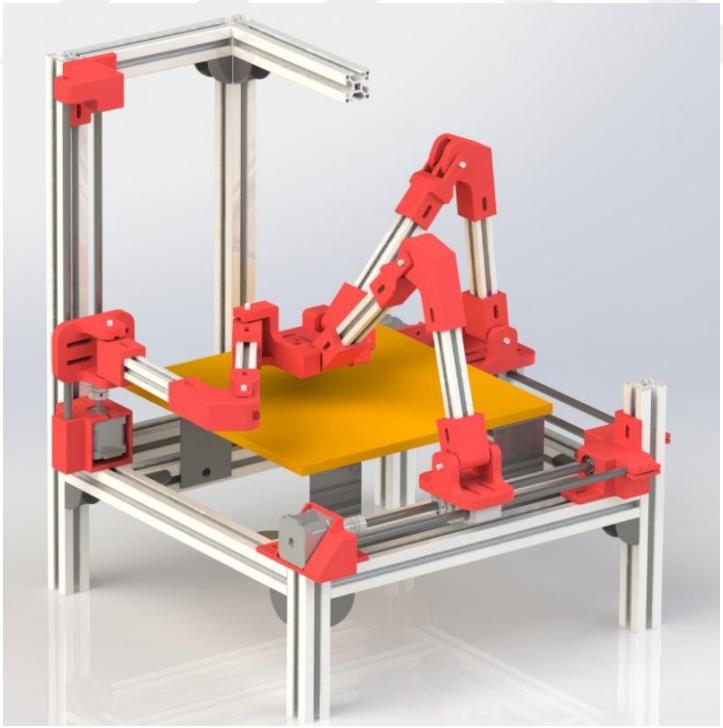


Figure 2.1 First tripteron design

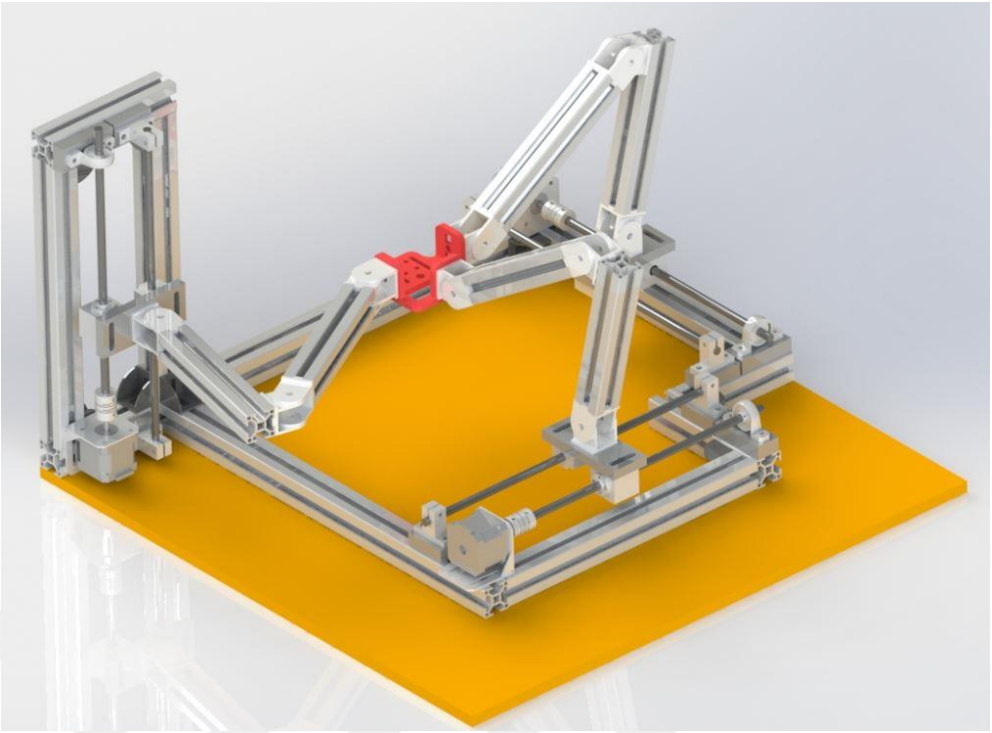


Figure 2.2 Final tripteron design

The design shown in Figure 2.1 was the first Tripteron Robot design. Mainly the design was made by the idea of using 3D printed models. ABS plastic was planned to be used on revolute joints, stepper grippers, linear cars and bearing grippers. Since the cost of the high quality printed models were high to be paid and the rigidity of the plastic material was not enough to be used on Tripteron Robot, the design in the Figure 2.2 was considered as final design to be manufactured. Only the common platform of the tripteron robot is planned to be 3D printed in the final design and the rest of the robot was planned to be made by standard parts which were cheaper and easy to find on the market. Since the robot has been made by standard parts, detailed parts and costs list will be given in production section of this chapter.

Workspace of the robot is planned to be 150x150x150 millimeters. The tripteron robot is the output of this experimental thesis study, therefore the workspace has been limited in the given dimensions. Also the possible effects of the payload on the robot affects the size of the workspace. Since the load was carried by one arm, more the length creates more bending force and the less precision at the end effector. The linear guides can drive the prismatic joints for 175 millimeters, however the limiters to have

reference at each axes are at 25.th millimeter of the linear guide so the usable workspace was 150 millimeters for each axes.

To decrease the total weight of the dynamic parts, the arms of the robot was planned to be made by 20x20 aluminum sigma profiles. Since the joints were changed to standard aluminum parts from 3D printed parts and the aluminum parts can only be used with 30x30 sigma profiles, the arms have been changed to 30x30 sigma profiles.

SolidWorks simulation is utilized in the finite element analysis of the robot. Performed analyzes consist of stress analysis, displacement analysis, modal analysis and pin quality check analysis. Since the required analysis time in SolidWorks is directly proportional to the details of the design, a simplified design is being used in order to decrease the calculation time. The simplified model is given in Figure 2.3.

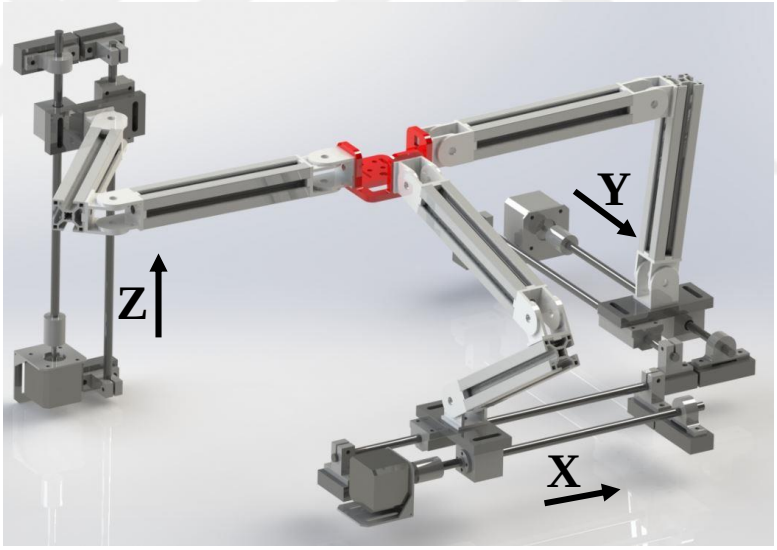


Figure 2.3 Simplified tripteron design

The simplified robot model is being made basically by ignoring the stationary body which will have no necessary effect on analysis results. Since the simplified model is being used, the analysis conditions of the robot will be given according to the simplified model. The first SolidWorks analysis were made at the end effector coordinates of x=25mm, y=175mm and z=175mm. In Figure 2.4, the applied analysis conditions can be seen. Since the workspace of the robot is designed to be 150mm and

the limit switches were positioned at 25.th mm, the control algorithms takes 25 mm as zero reference point.

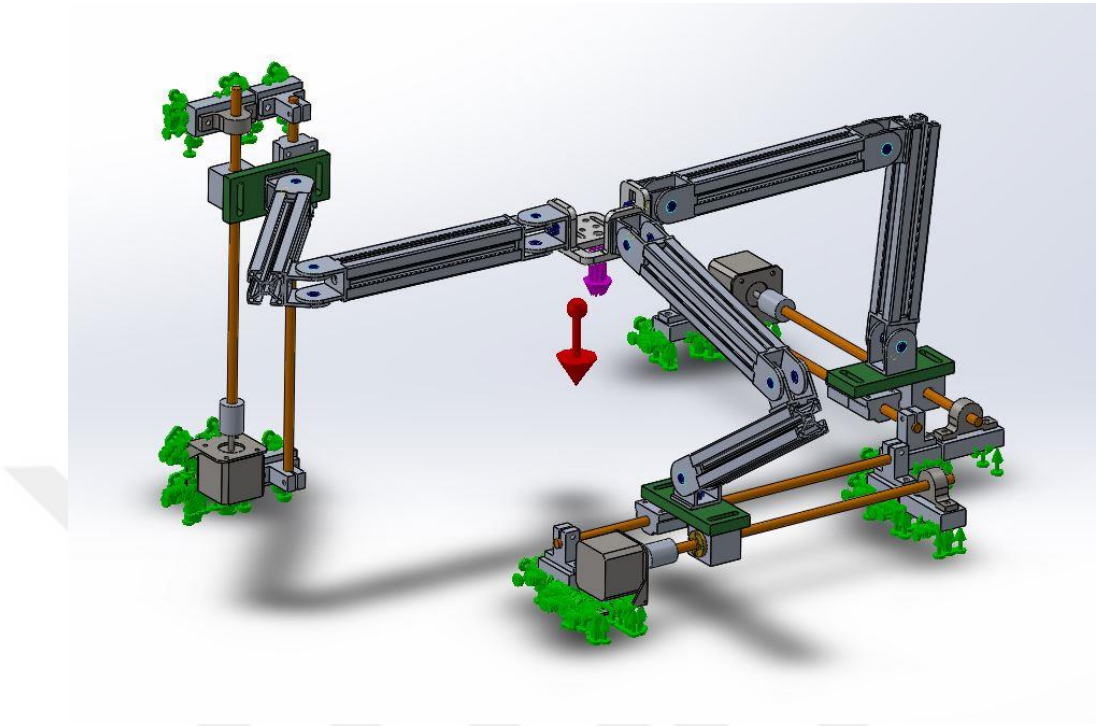


Figure 2.4 Tripteron Analysis conditions

Because of the linear guides are fixed on the rigid body structure and the body structure is ignored for simplified model, the connection between the dynamic system and the stable body is represented with green arrows so the model is fixed from the green arrows. Since the total weight of the robot is highly enough to be considered, the gravity force was applied to the system to include the mass of the robot itself to the calculations and the gravity force is represented in red arrow. The tripteron robot is mainly used for pick and place applications, therefore a simulated force is applied to the system to represent any load at the end effector. The decided load was 5 kilogram so the force was nearly 49 Newtons and represented in purple arrow.

The finite element mesh details are given in Figure 2.5. The minimum length in the model is almost 1 mm, however minimum and maximum mesh sizes are respectively 9 mm and 45 mm. Since the mesh type is curvature based, these mesh sizes were applicable on the model to do the fastest analysis.

Mesh Detaylar

Etüt adı	Static 1 (-Varsayılan-)
Mesh tipi	Katı Mesh
Kullanılan Meshleyici	Eğrilik tabanlı mesh
Jakoben noktalar	4 nokta
Maks. Eleman Boyutu	45 mm
Min. Eleman Boyutu	9 mm
Mesh kalitesi	Yüksek
Toplam düğüm	196030
Toplam eleman	108757
Maksimum En Boy Oranı	194.34
En Boy Oranı < 3 olan elemanların yüzdesi	19.8
En Boy Oranı < 10 olan elemanların yüzdesi	26.6
Şekli bozulmuş elemanların (Jakoben) %	0
Uyumsuz meshli başarısız parçaları yeniden mesh edin	Açık
Mesh tamamlama süresi (sa:dk:sn)	00:00:17
Bilgisayar adı	

Figure 2.5 FEM analysis mesh details

Stress analysis is made to be sure whether the design is safe for the determined analysis conditions. The results of the stress analysis is given in Figure 2.6.

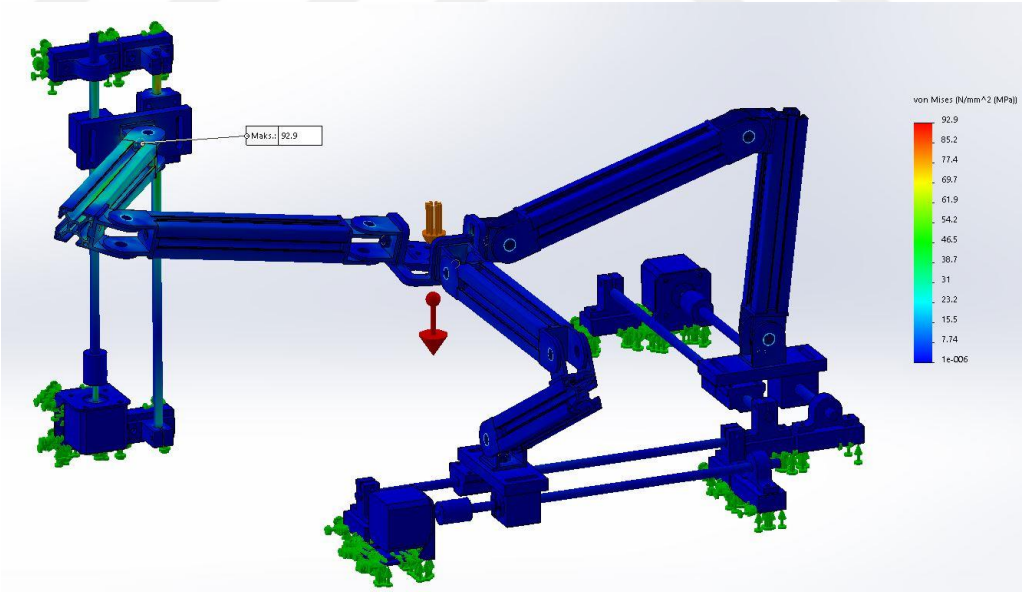


Figure 2.6 Stress analysis results

The stress analysis given in Figure 2.6 shows that only the vertical arm carries the determined test load while the other arms are free of load as expected. Parallel manipulators are commonly known with their ability to distribute the workload to

several arms equally in order to achieve the most precise motion, however the load distribution is not exist in tripteron robot so the mechanism is not capable for highly precise tasks. Aluminum sigma profile arms are tend to bend under high loads as seen from Figure 2.6. Since there is no load distribution on the arms, the vertical displacement of the end effector will increase directly proportional to the level of the load. Even the applied test load was extreme to be used in analysis, the robot's factor of safety (FOS) was more than 2 as 2.26 and the system was safe for considered test coordinates.

In parallel manipulators, the load at the end effector is carried by several arms. However in tripteron robot, mainly the workload is not carried by sharing by several arms because of the geometrical features of the mechanism and this may lead the tripteron robot to diverge from its planned trajectory. The acceptability of the deflection of the tripteron robot from its trajectory depends on the required precision of the task for which the robot is being used. The results of the total displacement is given in Figure 2.7.

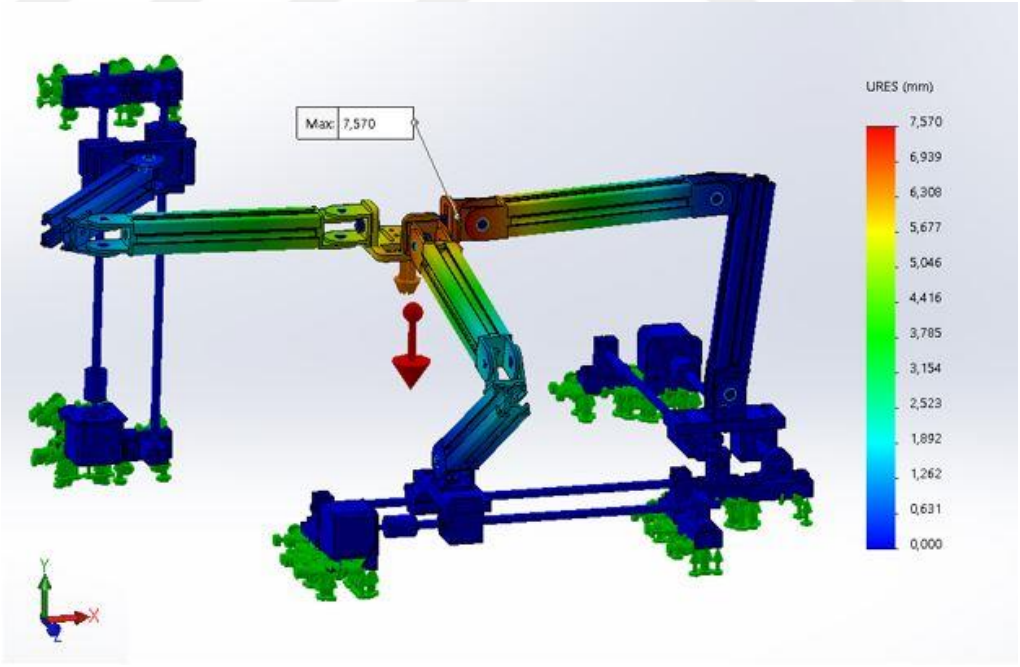


Figure 2.7 Total displacement results

The maximum total displacement of the tripteron robot occurs at the end effector part of the robot as expected. To examine the results of the total displacement in details, all the displacement results of the three axes can be analyzed. The detailed displacement analysis of the tripteron robot is given in Figure 2.8-2.10.

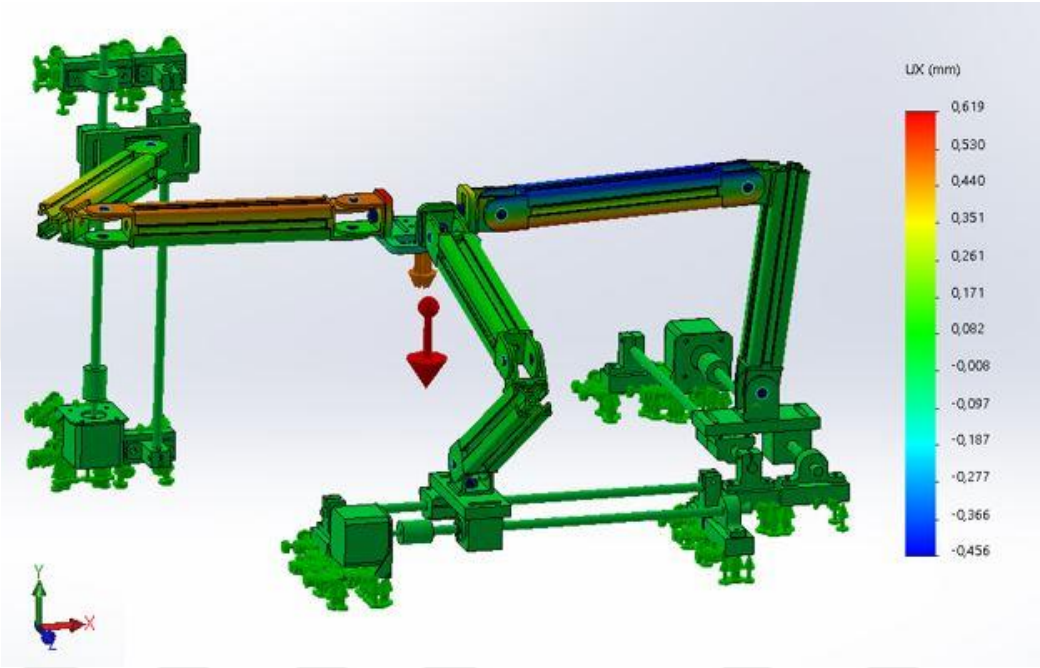


Figure 2.8 Displacement in x axis

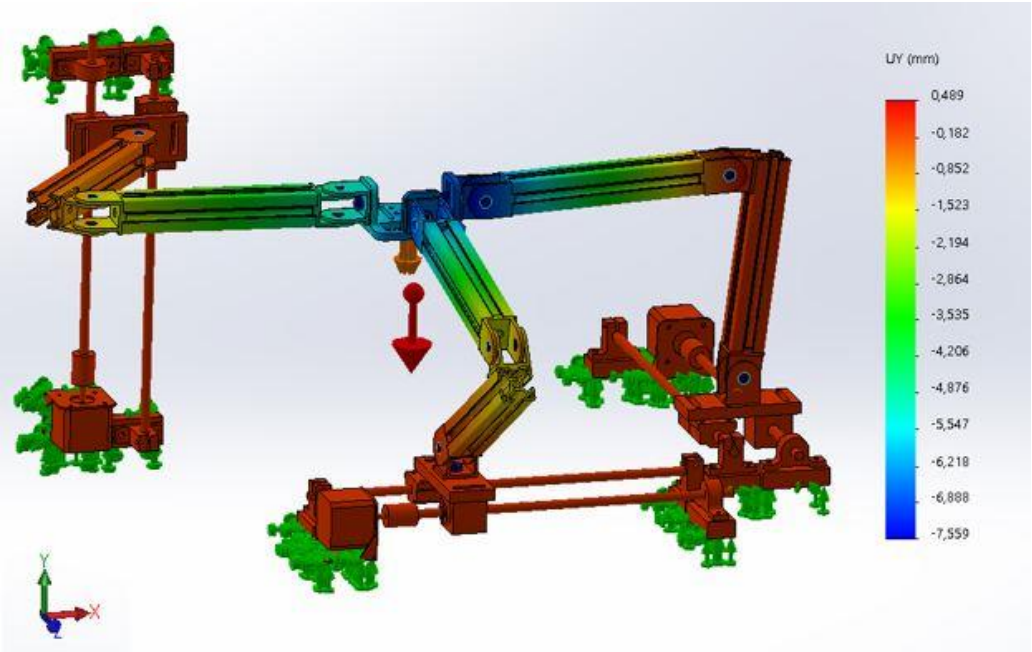


Figure 2.9 Displacement in y axis

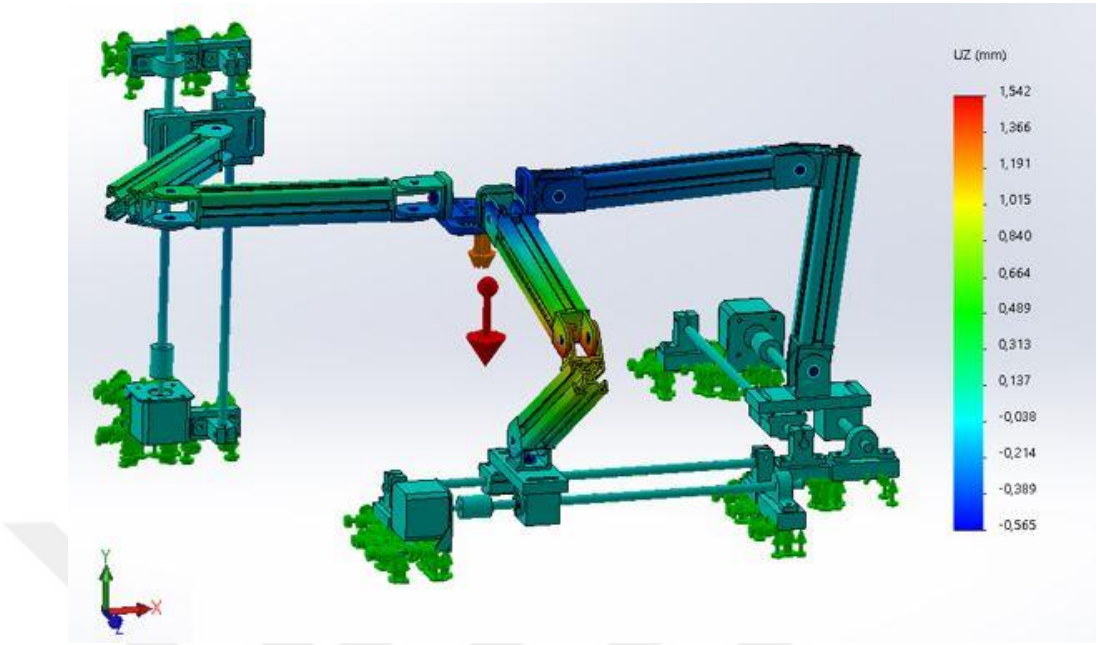


Figure 2.10 Displacement in z axis

The given x, y and z axes in Figure 2.3 was for the control system, however the axis orientation of the CAD model is different, therefore the axis orientation for the displacement analyzes is shown in the figures to prevent any confusion. The absolute displacement of the end effector at the x, y and z axes are respectively 0.08 mm, 7.5 mm and 0.5 mm. The detailed analyzes show that the vertical displacement is the greatest among all the three axes.

Overall, the stress and displacement studies show that the nonuniform distribution of the load affects the precision of the robot. Depending on the weight of the workload and the quality of the robot's material, the results of the analysis can change.

In the simplified CAD model, which was made for the analysis, the joint connections are defined as pins in SolidWorks, therefore the safety of the whole pins at the system under the determined workload needs to be checked. The results of the pin check safety analysis is given in Figure 2.11.

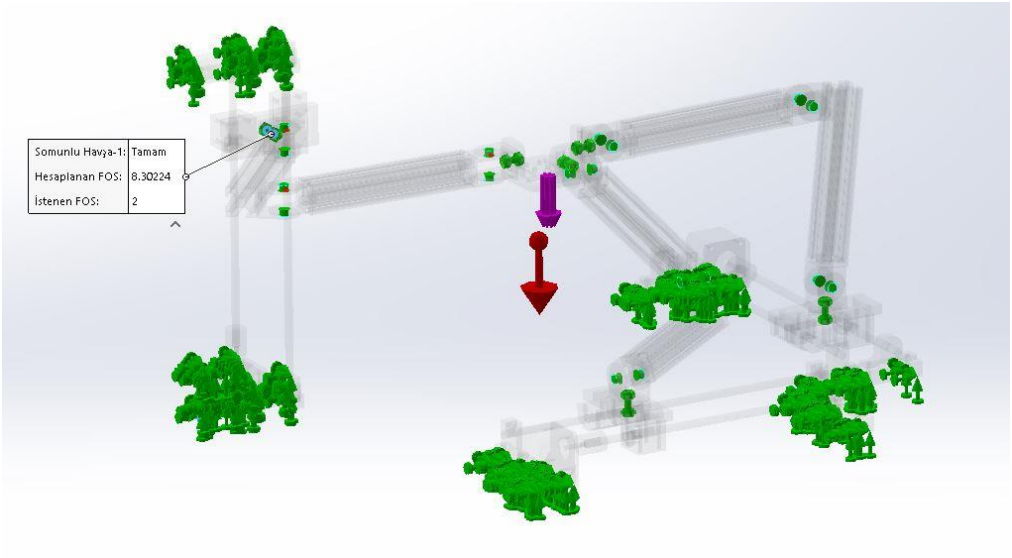


Figure 2.11 Pin check safety analysis results

Calculated minimum factor of safety (FOS) for pins was at vertical arm as 8.3. The pins' FOS results also show that the true effort by means of carrying the workload is performed by the vertical axis. Since the joints are made of steel and the material of the arms are aluminum, bending force affects mainly to the arms instead of pins.

By looking at the results of the three different analyzes (stress, displacement and pin check safety) it can be said that in order to make safer and more precise tripteron robot, the arm at the vertical axis should be more engineered. The load bearing arm needs to be rigid and light and also the joints must be strong enough to prevent bending.

Because of the workspace of the machine is 150x150x150 mm, the FEM analyzes should also be done at different positions of the robot. Multipoint analysis at SolidWorks requires high computation power in order to make the calculations in acceptable time, therefore predetermined 48 different positions of the end effector were inspected. The stress, displacement, FOS and first natural frequency analyzes results at different 48 locations are given respectively in Figure 2.12-2.15. Due to the analysis was made by using SolidWorks, the results were processed in Matlab in order to see the variation of the analysis results depending on the change of the position.

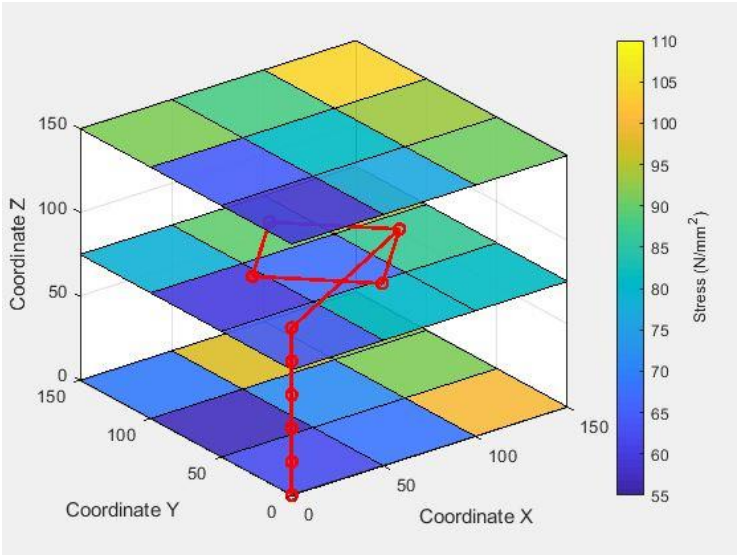


Figure 2.12 Multipoint stress analysis results

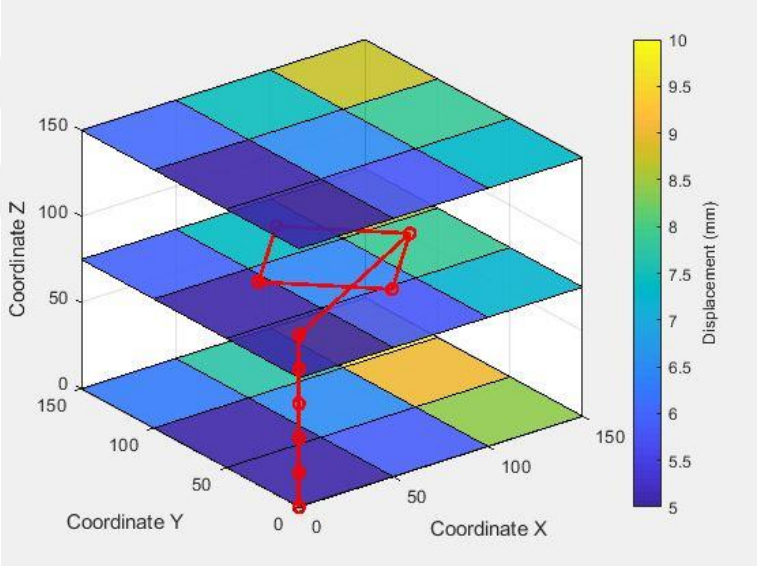


Figure 2.13 Multipoint displacement analysis results

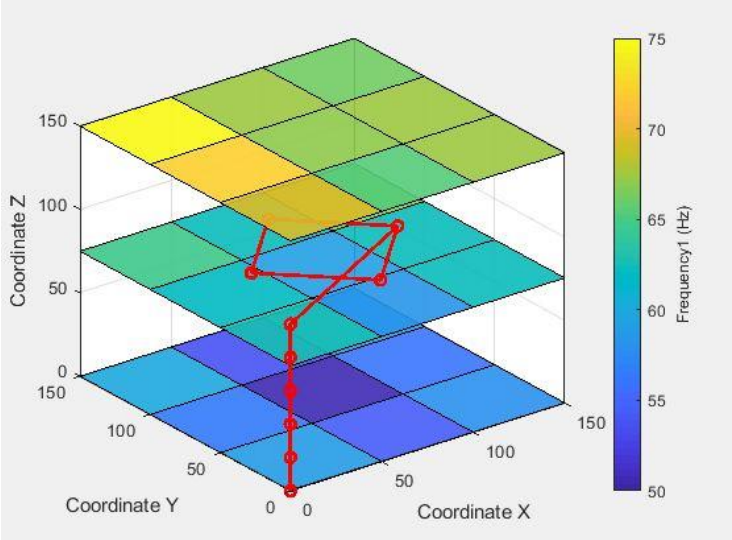


Figure 2.14 Multipoint first natural frequency analysis results

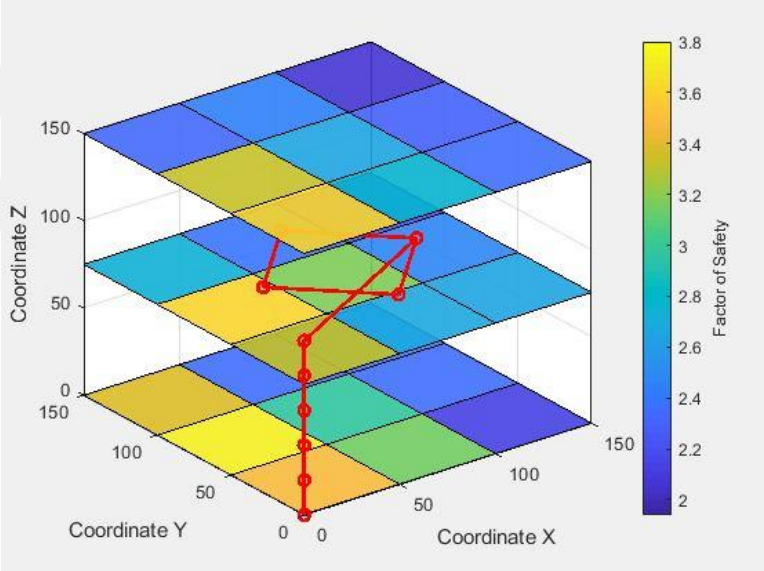


Figure 2.15 Multipoint factor of safety analysis results

To understand the variation of the parameters at different positions, an example end effector path is given within the 3D graphs. Figure 2.12 shows the change of the maximum stress result of the robot at different end effector positions and it can be seen that the stress tends to increase depending on the forward motion at any axis. The maximum stress value can be reached at the end point of the every axis, therefore 150x150x150 mm of the end effector is the most critical position for the robot. Since the FOS graph given in Figure 2.15 is the reverse of the stress graph, it can be seen that the least FOS point of the machine is at the end point of every axis which

represented in dark blue color. Figure 2.13 shows the change of the total displacement of the robot at different positions. Since the vertical displacement is almost equal to the total displacement, the results given in Figure 2.13 was accepted equal to the vertical displacement variation at different positions. The maximum displacement of the end effector occurs at the end of the x and y axis, however the forward motion at the z axis decreases the displacement. The maximum vertical displacement of the robot is calculated at 150x100x0 mm. Due to the change of the other arms' angles, the maximum displacement of the robot does not occur at the maximum stress point of the robot. Figure 2.14 shows the change of the first natural frequency of the robot at different positions. The Figure 2.14 shows that the change of the first natural frequency is directly proportional with the change at the z axis.

The results of the multipoint FEM analyzes show that the static and dynamic responses that can affect the control system of the tripteron robot change with the position of the end effector while tracking any predetermined trajectory. Since the relation between the input joint and the end effector is one to one, the position of the end effector should be equal to the position of the input joint, however in practice the position of the end effector diverges due to the structural deflections. The precision of the manipulator varies at different locations of the manipulator.

The control algorithm applied to the tripteron robot at this study does not include the data that consists of FEM results, due to the computation power that is required for this process. The open loop control algorithm created for the study is only based on the kinematics of the tripteron robot. The data obtained by the FEM analyzes can be used to train the artificial neural network for the future work related to this study.

2.2 Production

Due to the study is being made without any third party investment or scholarship, the design does not consist of highly professional mechanical parts. The used parts and cost list is given in Table 2.1.

Table 2.1 Cost table of the tripteron robot

Number	Part	Quantity	Cost [TL]
1	Sigma Profile 30x30	-	175.644
2	Sigma Profile 30x30 Connection Link	12	77.04
3	Nema 17 Stepper Gripper	3	19.11
4	KP08 Bearing	3	31.86
5	Nema 17 Stepper	3	208.35
6	Coupling 5x8mm	3	26.61
7	Trapezoidal Lead Screw T8 300mm	3	118.26
8	Aluminum Block Shaft	3	26.97
9	SK8 Aluminum Shaft Gripper	6	39.18
10	Aluminum Shaft 8x350mm	3	61.95
11	SCE 8mm Linear Bearing	6	103.97
12	Sigma Profile 30x30 Aluminum Joint	9	320.17
13	3D Printed End Effector	1	101.26
14	Nuts, Screws and Rings	228	132.84
15	Stepper Driver TB6600	3	152.13
16	Power Supply 12V 30A	1	120.94
17	Aluminum Machined Parts	12	220
18	Arduino Mega2560 R3	1	57.07
19	Jumper Cables	-	6.646

The manufactured tripteron robot in the scope of this study is given in Figure 2.16. The robot is capable of doing some tasks, however several mechanical parts needs to be upgraded to increase the capacity of the robot. Technical drawing of the tripteron robot's assembly model was given in appendix, detailed information about the parts' configuration in the assembly can be found in the given drawing. Since the 3D printed end effector part was designed from the scratch, technical drawing of the end effector part was given in appendix.

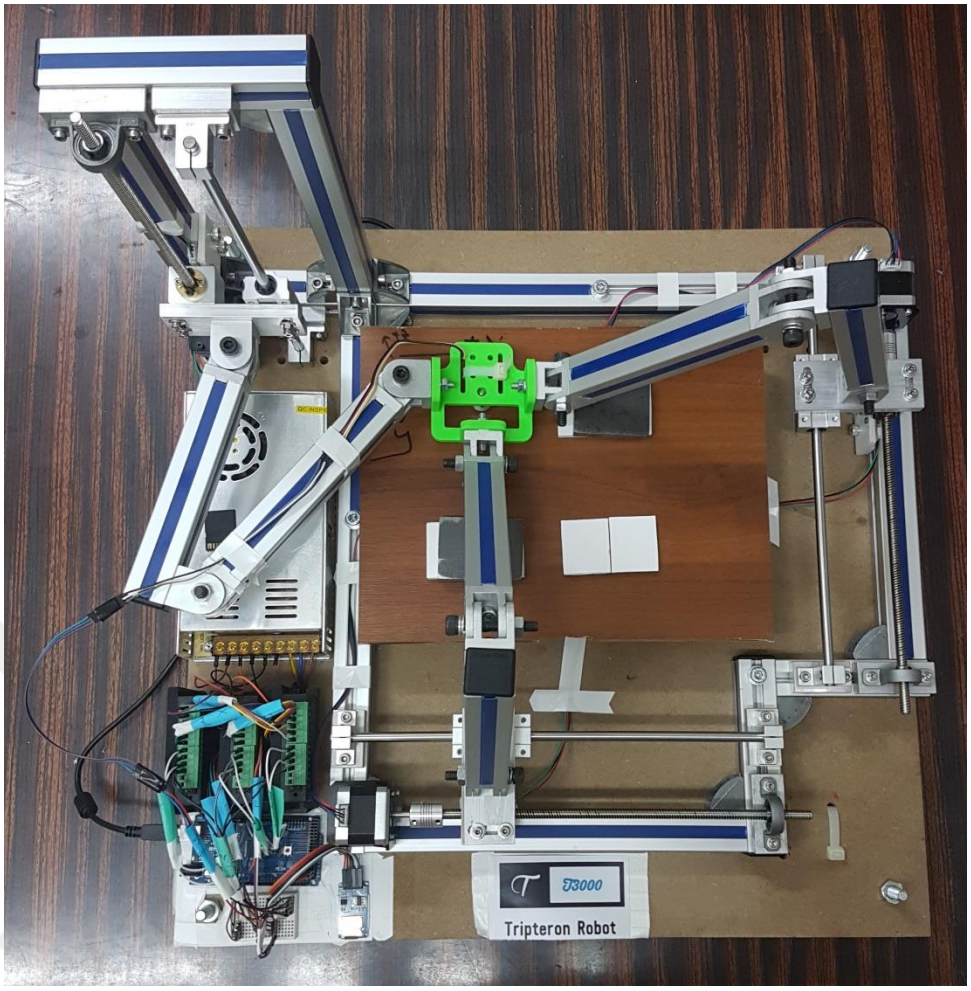


Figure 2.16 Manufactured tripteron robot (Personal archive, 2019)

The design of the robot was made generally by standard parts, which can easily be found on the market. To test the tripteron robot on several pick and place applications, an electromagnet was used on the end effector. Since the production cost limit was not high enough to afford professional mechanical parts, the robot is not capable of working on its full performance. The upgrades that can be done as future work are listed below:

- Lighter arms to decrease the dynamic weight
- Lead screws can be changed to ball screws
- Aluminum shaft can be increased in diameters to 12 mm
- Specially designed and manufactured joints with appropriate bearing
- SCE 8 mm linear bearings can be changed to SCE 12 mm long linear bearing

CHAPTER THREE
CONTROL SOFTWARE

Two different software products used to control the tripteron robot were Matlab and Arduino IDE. Communication between these two programs was made by transferring the data with SD card. Matlab was responsible to calculate the required step motor driving data according to the inputs of the user. The required step motor data to operate the system consists of number of steps at each step motor for each track, delay times and input signal distribution. The program written in Matlab was compacted to a graphical user interface (GUI) based package program to facilitate the usage of the program for any kind of user. Once the calculation completed by Matlab, created data is transferred to an SD card to be processed by the Arduino microcontroller board. Arduino microcontroller board is only responsible to read and process the data written in the SD card. The main calculations made in Matlab are based on two algorithms. The first algorithm is least mean square (LMS) algorithm and the second algorithm was simultaneous stepper drive (SSD) algorithm. LMS algorithm is a basic artificial neural network algorithm that is appropriate to be used on kinematics of the tripteron robot. SSD algorithm is a basic input pulse signal distributer to synchronize the different actuators to do one common task. Figure 3.1 is given to summarize how the control mechanism works.

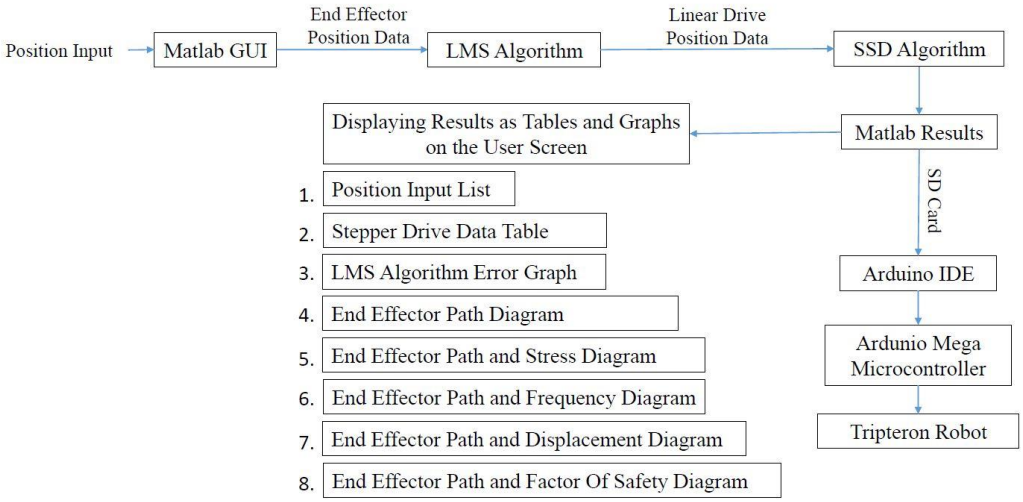


Figure 3.1 Control schema of the robot

3.1 Artificial Neural Networks

The control algorithm of the tripteron robot is designed as an open loop position controller. The tripteron mechanism is a fully decoupled parallel mechanism with trivial kinematic solutions at every arm. Since all the axes needs to be controlled separately and the relation between the input joint position change and the end effector position change is linear, the appropriate and sufficient artificial neural network algorithm is LMS algorithm.

LMS algorithm is a basic data processor inspired from perceptron. The algorithm basically capable of processing linear data with multiple inputs. Since the mechanism is fully decoupled and the position datasets are linear, LMS algorithm with one input and one output can be used. How the LMS algorithm works can be seen in Figure 3.2.

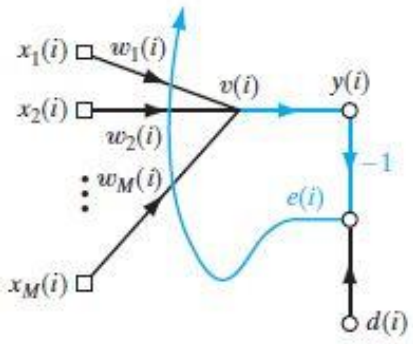


Figure 3.2 LMS algorithm schema (Haykin, 2009)

The network works as follows:

- Multiplication of the input (x) variables with weights (w)
- Addition of the multiplied variables, the result is called output (v or y), v=y
- Comparison of the output (y) and the desired output (d), error (e) determination
- Changing the weights according to the calculated error
- Stopping the training when acceptable error reached
- Testing the network with additional inputs

According to Haykin (Haykin, 2009), the system can simply be expressed with the given mathematical equations:

$Training\ Sample$: $Input\ signal\ vector = x(n)$

$Desired\ response = d(n)$

$User\ selected\ parameter: \eta$

$Initializition.\ Set\ w(0) = 0$

$For\ n = 1,2, \ldots.$

$$e(n) = d(n) - w^T(n)x(n) \quad (3.1)$$

$$w(n+1) = w(n) + \eta x(n)e(n) \quad (3.2)$$

Where η stands for a user selected parameter called learning rate and it represents the step size of the network between iterations. Step size may be a constant or a variable value. The comparison of the LMS algorithm results and kinematic equations results is given in Figure 3.3 to show why the algorithm is appropriate to be used.

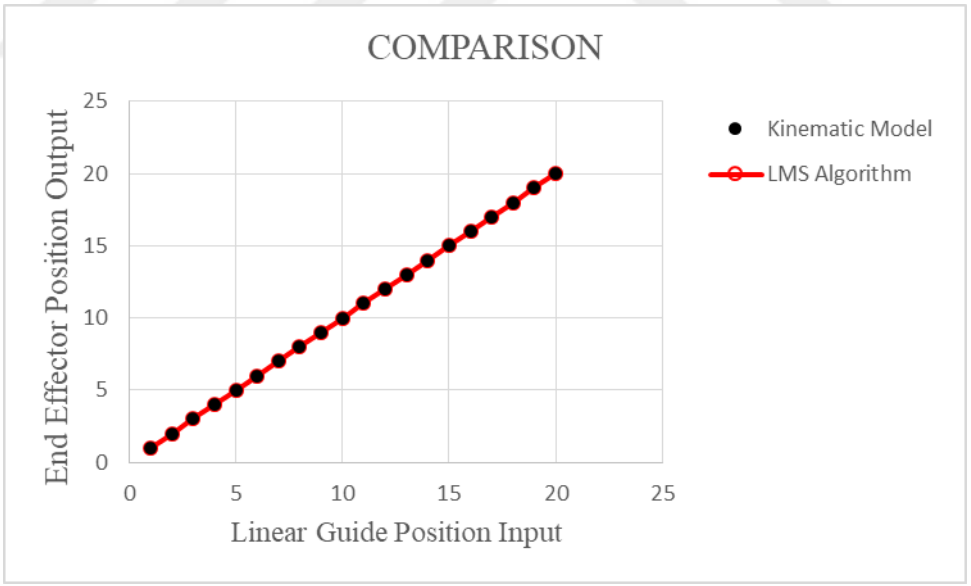


Figure 3.3 Overlapping position data

3.2 Simultaneous Stepper Drive

After the position data of the linear guide is calculated by LMS algorithm, number of steps, which are calculated by using the position data, needs to be distributed to three different step motors to synchronize the motion. The program follows the points one by one that entered by user. The distance between any points need to be changed from millimeter to number of steps. Number of steps is a dependent parameter to the resolution of the step motor driver and the lead distance of the screw. The step motor driver used were TB6600. The motor drivers were used in the maximum resolution (1/32) while doing any given task to decrease the vibration effects. The simultaneous stepper drive (SSD) algorithm works by proportionally distributing the pulse signal to each step motor for each track. Table 3.1 is given to express how the algorithm works.

Table 3.1 Pulse distribution table

Track	Stepper X Axis	Stepper Y Axis	Stepper Z Axis	Main Loop	Loop X	Loop Y	Loop Z
1	800	1600	400	400	2	4	1
2	3200	1800	600	200	16	9	3
3	500	3800	2600	100	5	38	26
4	400	400	0	400	1	1	0
5	0	0	1200	1200	0	0	1
6	500	600	950	50	10	12	19

In Table 3.1, stepper x axis, stepper y axis and stepper z axis shows the number of steps that needs to be taken to move next location. These three numbers needs to be processed simultaneously in order to achieve the most robotic and synchronized motion. Because of common step motor drive algorithms work by pulse loops, a common loop was thought to distribute the number of steps proportionally to each motor. For example in track 1, main loop is processed for 400 times whilst sub loops are processed proportionally. The most important point is that step motors with greater loop numbers should be processed faster than lower step motors to prevent the time delay between the pulse signals. For instance in track 1, step motor that drives the y axis should be two times faster than the step motor that drives x axis and four times faster than the step motor that drives z axis. Step motor speed limits depend on the mechanical conditions of the robot. In this study step motor speed is limited by the

software, however the limits can be changed when the required upgrades of the robot is done.

After all the coding work, the program converted to a package program by using Matlab Application Compiler. Since the software has a graphical user interface, workflow of the program was created in the form of questions and answers. All the created graphical user interface windows can be seen in Figure 3.4-3.11.

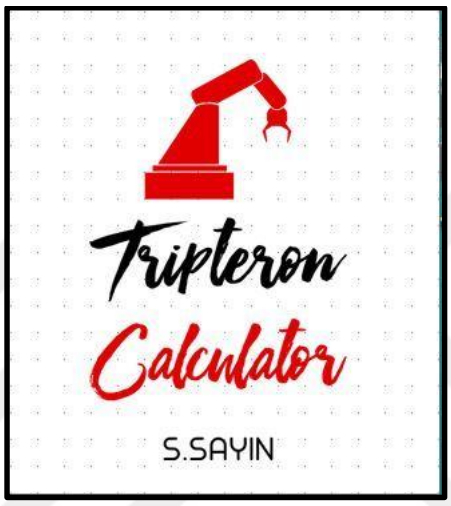


Figure 3.4 Software opening screen

Figure 3.5 Software opening message

Figure 3.6 Software mode selection

Figure 3.7 Transaction continuity warning

Figure 3.8 Collision prevention warning

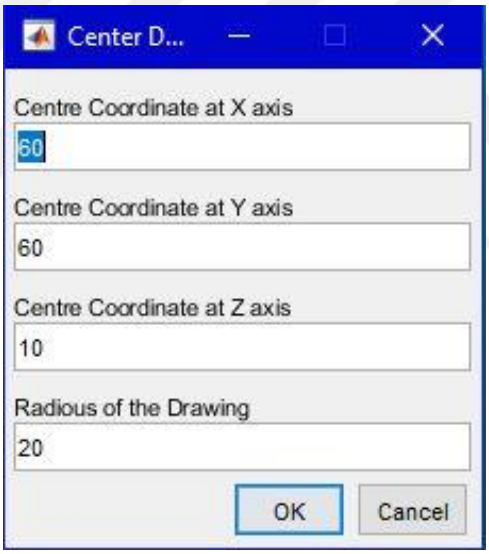


Figure 3.9 Data entry window

Figure 3.10 Drawing geometry selection window

Figure 3.11 End of the transaction warning

Matlab program also gives the results of all calculations made while the execution of the program. The Matlab outputs are given in Figure 3.12-3.15.

xmani	ymani	zmani
0	0	0
0	0	20
0	0	40
0	0	60
0	0	80
0	0	100
120	80	80

Figure 3.12 Position entry list

stpcrc			dlycrc			mlooptc	sbloop		
0	0	16000	188	188	75	16000	0	0	1
0	0	16000	188	188	75	16000	0	0	1
0	0	16000	188	188	75	16000	0	0	1
0	0	16000	188	188	75	16000	0	0	1
0	0	16000	188	188	75	16000	0	0	1
96000	64000	-16000	47	47	75	16000	6	4	1
-32000	32000	0	47	47	188	32000	1	1	0
-32000	-32000	0	47	47	188	32000	1	1	0
32000	-32000	0	47	47	188	32000	1	1	0
32000	32000	0	47	47	188	32000	1	1	0
-96000	-64000	16000	47	47	75	16000	6	4	1
0	0	-16000	188	188	75	16000	0	0	1
0	0	-16000	188	188	75	16000	0	0	1
0	0	-16000	188	188	75	16000	0	0	1
0	0	-16000	188	188	75	16000	0	0	1
0	0	-16000	188	188	75	16000	0	0	1

Figure 3.13 Pulse distribution list

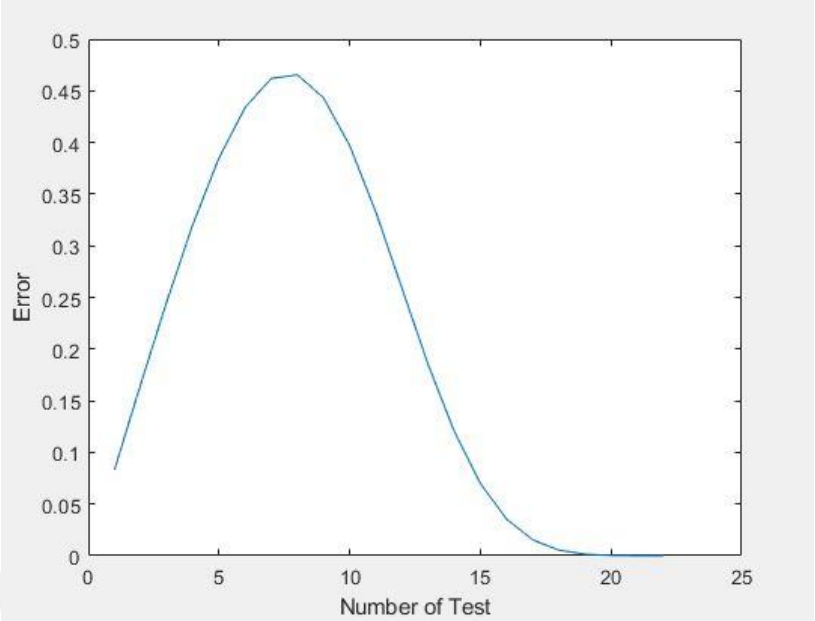


Figure 3.14 LMS algorithm error tracking

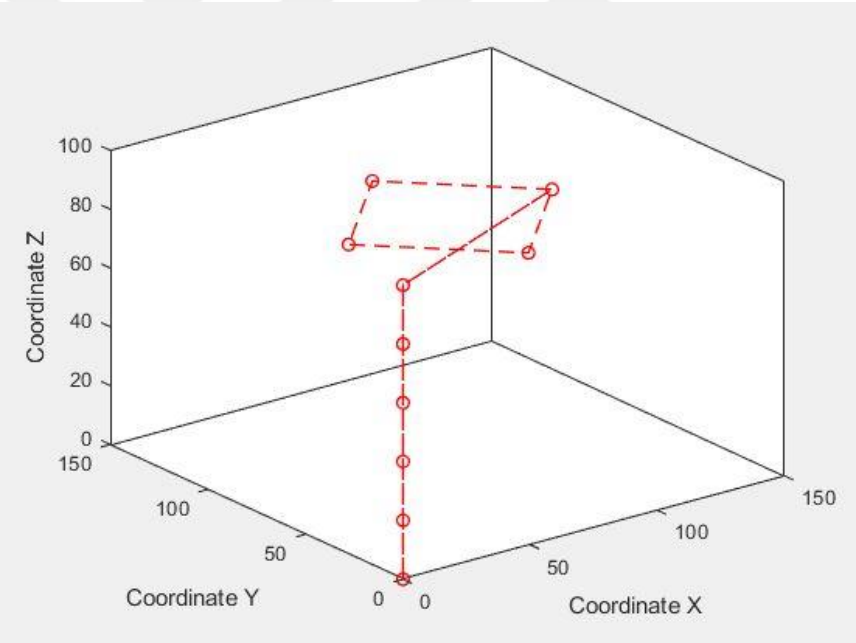


Figure 3.15 End effector trajectory planning

Figure 3.4 and Figure 3.5 give the opening screen and opening message of the software. Figure 3.6 asks to user about the task. Full auto G Code mode is planned to read any written G code, however the sub program is not ready to be used yet. Manuel mode is designed to drive the end effector from one point to another freely. Demonstration mode contains three different shape to be drawn such as square,

triangle and circle. Figure 3.7 asks to user whether the process should continue or not. Figure 3.8 is a warning message about the safety. The software prevents the end effector from moving to the edges of the linear guide. Since all the demonstration models are based on the circle model, Figure 3.9 asks for the required parameters of the drawing and the Figure 3.10 asks the geometry desired to be drawn. Once the program determined the points to be tracked according to the circle model, the end effector can draw many different shapes. At the end of the all process, calculated data is stored to be used in another software environment. Figure 3.11 warns the user when the process ends. Matlab presents the results to make sure the user about the calculations. Figure 3.12 shows the input positions entered by the user. Figure 3.13 shows the number of steps, delay times and pulse distribution details for each track. Figure 3.14 shows the result of the LMS algorithm error change to show the reliability of the network. Figure 3.15 presents the trajectory that planned to be tracked by the end effector.

Due to the unbalanced load distribution on the mechanism, the end effector displacement changes at different points of the workspace as mentioned in the analysis part of the study. The position control of the manipulator is made by only considering the kinematics of the tripteron robot, however the lack of precision especially on vertical axis may affect the accuracy of the robot while doing any task. The artificial neural networks can be used to estimate the displacement at any point in the workspace. Since the Matlab has artificial neural networks tool to solve nonlinear systems, the data calculated by FEM displacement analyses can be used in training process. In this study, number of samples was 48 so the input vector size was 3x48 and the output vector size was 1x48. Due to the lack of sufficient numbers of samples, the error was not low enough to have a reliable network. To obtain the sufficient numbers of data, a computer with higher computational power and faster processing time should be used. Figure 3.16 gives the error change that obtained for the developed network.

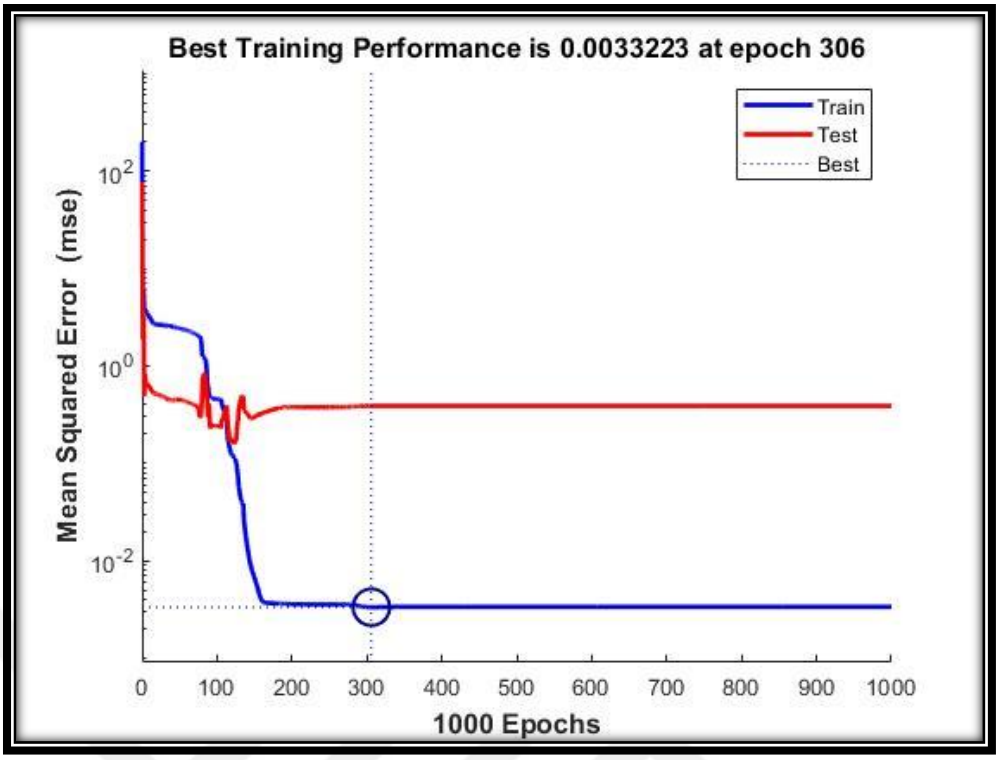


Figure 3.16 MSE for the planned neural network study

By considering the mechanics and the kinematics of the robot together, success rate of the precise control can be increased. As a result, to develop the control mechanism of the robot, the ANN that already in use for the kinematics of the robot can be improved by combining with another ANN that is responsible for the mechanics of the robot.

The data stored in an SD card is transferred to the Arduino Mega2560 R3 by using SD card shield. Arduino has its own programming environment called Arduino IDE. The data created in Matlab is processed by the code written in Arduino. Since the Matlab program was GUI based, the Arduino code was also developed GUI based to increase the effectiveness of the program. The developed Arduino code has some several interaction points with user as Matlab program does, therefore the program workflow is in the form of questions and answers. All program codes are given in the appendices.

CHAPTER FOUR
CONCLUSIONS

The design, analysis, production and control of the tripteron robot was studied in the scope of this thesis study. The design and analysis of the robot was performed by using SolidWorks computer aided design and analysis program. In the design stage affordability and the functionality of the components were considered. The production was made just as planned in the design stage, therefore the production plan was very effective when the production time and necessary effort considered. Control of the robot was achieved by using Matlab and Arduino IDE software. Artificial neural network was applied on the kinematics of the tripteron robot to convert the end effector position data to the linear guide position data. Due to the insufficient numbers of the samples, artificial neural network could not be applied to the robot's control software in order to compensate the static deflections; however the available data is trained to make a sample presentation.

As a result of the conducted studies, a tripteron robot was obtained in order to perform simple pick and place tasks by means of the developed control software. The robot can be improved in the future from the points that already described at related chapters. Required developments include mechanical improvements at design and production stage, and control algorithm improvements at analysis and software development stage.

REFERENCES

Asimov, I. (1950). *I, robot*. New York: Gnome Press.

Dehghani, M., Ahmadi, M., Khayatian, A., Enghtesad, M., & Farid, M. (2008). Neural network solution for forward kinematics problem of hexa parallel robot. *American Control Conference*, 4214-4219.

Elkady, A., Elkobrosy, G., Hanna, S., & Sobh, T. (2008). Cartesian parallel manipulator modeling, control and simulation. *Parallel Manipulator towards New Applications*, 269-296.

Ghazi, M., Nazir, Q., Butt, S.U., & Baqai, A.A. (2018). Accuracy analysis of 3-RSS delta parallel manipulator. *Procedia Manufacturing*, *17*, 174-182.

Gosselin, C.M., Masouleh, M.T., Duchaine, V., Richard, P.L., Foucault, S., & Kong, X. (2007). Parallel mechanisms of the multipteron family kinematic architectures and benchmarking. *International Conference on Robotics and Automation*, 555-560.

Hamdoun, O., Bakkali, L.E., & Baghli, F.Z. (2017). Analysis and optimum kinematic design of a parallel robot. *Procedia Engineering*, *181*, 214-220.

Haykin, S. (2009). *Neural networks and learning machines* (3rd ed.). Ontario: Pearson.

Herrero, S., Pinto, C., Altuzarra, O., & Diez, M. (2018). Analysis of the 2PRU-1PRS 3DOF parallel manipulator: kinematics, singularities and dynamics. *Robotics and Computer-Integrated Manufacturing*, *51*, 63-72.

Heydarzadeh, M., Karbasizadeh N., Masouleh, M.T., & Kalhor, A. (2018). Experimental kinematic identification and position control of a 3-DOF decoupled parallel robot. *Journal of Mechanical Engineering Science*, *233*, 1841-1855.

Hüllmann, D., Kohlhoff, H., Erdmann, J., & Neumann, P.P. (2019). Control of a spherical parallel manipulator. *Materials Today, 12,* 423-430.

Yavuz, Ş., & Hocaoğlu, M. (2015). İki serbestlik dereceli robot kollarında uç nokta konumunun yapay sinir ağları ile bulunması. *Proceedings of Trc-IFToMM Symposium on Theory of Machines and Mechanism*, 363-366.

Liu, Q., Xiao, J., Yang, X., Liu, H., Huang, T., & Chetwynd, D.G. (2019). An iterative tuning approach for feedforward control of parallel manipulators by considering joint couplings. *Mechanism and Machine Theory, 140*, 159-169.

Nag, A., Mohan, S., & Bandyopadhyay, S. (2017). Forward kinematic analysis of the 3-RPRS parallel manipulator. *Mechanism and Machine Theory, 116*, 262-272.

Noshadi, A., Mailah, M., & Zolfagharian, A. (2012). Intelligent active force control of a 3-RRR parallel manipulator incorporation fuzzy resolved acceleration control. *Applied Mathematical Modelling, 36,* 2370-2383.

Patel, Y.D., & George, P.M. (2012). Parallel manipulators applications-a survey. *Modern Mechanical Engineering, 2,* 57-64.

Singh, Y., Vinoth, .V, & Santhakumar, M. (2014). Dynamic modelling and control of a 3-DOF planar parallel robotic ($XY\theta_Z$ motion) platform. *Procedia Materials Science, 5,* 1528-1539.

Stigger, T., Siegele, J., Scharler, D.F., Pfurner, M., & Husty, M.L. (2019). Analysis of a 3-RUU parallel manipulator using algebraic constraints. *Mechanism and Machine Theory, 136,* 256-268.

Vieira, H.L., Wajnberg, E., Beck, A.T., & Silva, M.M. (2019). Reliable motion planning for parallel manipulators. *Mechanism and Machine Theory, 140,* 553-566.

Wang, L., Wang D., & Wu, J. (2019). Dynamic performance analysis of parallel manipulators based on two-inertia-system. *Mechanism and Machine Theory*, *137*, 237-253.

Xu, J., Wang, Q., & Lin, Q. (2018). Parallel robot with fuzzy neural network sliding mode control. *Advances in Mechanical Engineering*, *10* (10), 1-8.

Yang, Y., Peng, Y., Pu, H., Chen, H., Ding, X., Chirikjian, G.S., & Lyu, S. (2019). Deployable parallel lower-mobility manipulators with scissor-like elements. *Mechanism and Machine Theory*, *135*, 226-250.

Zhukov, Y.A., Korotkov, E.B., Moroz, A.V., Zhukova, V.V., & Abramov, A.M. (2018). Adaptive neural network control of hexapod for aerospace application. *IOP Conference Series: Materials Science and Engineering, 441* (1), 1-9.

APPENDICES

APPENDIX-1: Main Matlab Code “TripteronCalculator”

```
clc;clear;close all

myicon = imread('TripteronPro.jpg');
waitfor(msgbox('Welcome to Tripteron
Calculator','TripteronCalculator','custom',myicon));

while (1)
    answer = questdlg('How would you like to control the
Tripteron Robot?','Mode Selection','Full Auto G
Code','Manuel','Demonstration','Demonstration');
    switch answer
        case 'Full Auto G Code'
        waitfor(warndlg('In Progress,
Sorry!','Warning'));

        case 'Manuel'
        ProjectCode;
        break;

        case 'Demonstration'
        demonstrationcode;
        break;

        otherwise
        waitfor(msgbox('Program will shut
down','TripteronCalculator','custom',myicon));
        break;
    end
end
```

APPENDIX-2: Sub Matlab Code "ProjectCode"

```
a=1;
PR=6400; %Pulse/rev oranı sürücüden ayarladığımız değer
ttl=8;  %8mm for one full revolution of the screw
Trapezoidal Turn Lead T8
stepangle=360/PR;
maxdist = 150; %The maximum distance that can be reached
mindist = 0; %The minimum distance that can be reached

while(a==1)
testsystem; %Eğitim her sistem yeniden başladığında 1
defa gerçekleşiyor.
a=a+1;
end

linearcalculator; %Points to be tracked
truepoint; %Goes back to home at the end of the process
runtripteron; %Delay in Microseconds
loopvaluecalculator;
leadtheway; %Shows the results
documentation; %Creates text file from the required data

waitfor(msgbox('All data calculated and exported to
"Ardufiles" file in
documentary','TripteronCalculator','custom',myicon));
%grManipulator shows the points that we have sent to the
system X Y Z
%grStepact shows the steps of arduino to do X Y Z
%grdelayMicsex gives the delay time to be used on Arduino
%mainloop gives the Arduino main step motor control loop
values
%grloopvalues gives values for all subloops X Y Z
```

APPENDIX-3: Sub Matlab Code "testsystem"

```
s=50;      %Number of choosed  data to use
k=0.1;     %First data to be created
res=0.1;   %Input data resolution

maxres=(stepangle*ttl)/360; %Max distance resolution in
"mm" for determined pulse/rev

for  a=1:s

    input(a,1)=k;
    k=k+res;

end

desired=input; %Desired values according to the input
values
w=sqrt(0.1)*randn(1,1); %Randomly created weight matrix

n0=0.25; %LMS algorithm value
tao=200; %LMS algorithm value

for b=1:s %Test Cycle
    output(b,1)=input(b,1)*w;
    error(b,1)=desired(b,1)-output(b,1);
    nu=n0/(1+(b/tao));
    w=w+nu*error(b,1)*input(b,1);
    if abs(error(b,1))<1e-7
        break;
    end
end
```

APPENDIX-4: Sub Matlab Code "linearcalculator"

```
bg=2;
cg=1;
kg=1;

backhome;

while  (kg)
    prompt = {'Enter the Points to be Tracked'};
    title = 'Number of Inputs';
    dims = [1 35];
    definput = {'5'};
    szc = inputdlg(prompt,title,dims,definput);
    sz=str2num(szc{1});

    if sz<5 || sz>50
        waitfor(errordlg('Number of Inputs must be more
than 5 and less than 50','WARNING'));
    else
        break;
    end

end

sz=sz+2;
waitfor(msgbox('Lets do
it','TripteronCalculator','custom',myicon));


while(bg<sz)
    while(cg)
```

```
        prompt={'Coordinate X','Coordinate Y','Coordinate
Z'};
        title='Control Window';
        dims=[1 35];
        definput={'0','0','0'};
        answer = inputdlg(prompt,title,dims,definput);

        mcx(bg,1)=str2num(answer{1}); %mcx => Manipulator
Coordinate X
        mcy(bg,1)=str2num(answer{2}); %mcy => Manipulator
Coordinate Y
        mcz(bg,1)=str2num(answer{3}); %mcz => Manipulator
Coordinate Z

        if mcx(bg,1)>maxdist || mcy(bg,1)>maxdist ||
mcz(bg,1)>maxdist
           waitfor(errordlg('Inputs must be less than
150mm','WARNING'));
        elseif mcx(bg,1)<mindist || mcy(bg,1)<mindist ||
mcz(bg,1)<mindist
           waitfor(errordlg('Inputs must be more than
0mm','WARNING'));
        else
             break;
        end
    end


lscx(bg,1)=mcx(bg,1)*w;
lscy(bg,1)=mcy(bg,1)*w;
lscz(bg,1)=mcz(bg,1)*w;
```

```
SpaceX(bg-1,1)=lscx(bg,1)-lscx(bg-1,1); %Distance that
Linear System Should Move X
SpaceY(bg-1,1)=lscy(bg,1)-lscy(bg-1,1); %Distance that
Linear System Should Move Y
SpaceZ(bg-1,1)=lscz(bg,1)-lscz(bg-1,1); %Distance that
Linear System Should Move Z

turnx(bg-1,1)=SpaceX(bg-1,1)/ttl;  %Number of stepper
must fully revolute X
turny(bg-1,1)=SpaceY(bg-1,1)/ttl;  %Number of stepper
must fully revolute Y
turnz(bg-1,1)=SpaceZ(bg-1,1)/ttl;  %Number of stepper
must fully revolute Z

realturnx(bg-1,1)=turnx(bg-1,1)*PR;  %Number of steps
that stepper must do X
realturny(bg-1,1)=turny(bg-1,1)*PR;  %Number of steps
that stepper must do Y
realturnz(bg-1,1)=turnz(bg-1,1)*PR;  %Number of steps
that stepper must do Z


bg=bg+1;

end
```

APPENDIX-5: Sub Matlab Code "backhome"

```
homex=0;
homey=0;
homez=0;

mcx(1,1)=homex; %Manipulator Home X
mcy(1,1)=homey; %Manipulator Home Y
mcz(1,1)=homez; %Manipulator Home Z

lscx(1,1)=mcx(1,1)*w; %Linear System Home X
lscy(1,1)=mcy(1,1)*w; %Linear System Home Y
lscz(1,1)=mcz(1,1)*w; %Linear System Home Z
```

APPENDIX-6: Sub Matlab Code "truepoint"

```
truehomex=0;
truehomey=0;
truehomez=0;

mcx(sz,1)=truehomex;
mcy(sz,1)=truehomey;
mcz(sz,1)=truehomez;

lscx(sz,1)=mcx(sz,1)*w; %Linear System Home X
lscy(sz,1)=mcy(sz,1)*w; %Linear System Home Y
lscz(sz,1)=mcz(sz,1)*w; %Linear System Home Z

SpaceX(sz-1,1)=lscx(sz,1)-lscx(sz-1,1); %Distance that
Linear System Should Move X
SpaceY(sz-1,1)=lscy(sz,1)-lscy(sz-1,1); %Distance that
Linear System Should Move Y
SpaceZ(sz-1,1)=lscz(sz,1)-lscz(sz-1,1); %Distance that
Linear System Should Move Z
turnx(sz-1,1)=SpaceX(sz-1,1)/ttl;  %Number of stepper
must fully revolute X
turny(sz-1,1)=SpaceY(sz-1,1)/ttl;  %Number of stepper
must fully revolute Y
turnz(sz-1,1)=SpaceZ(sz-1,1)/ttl;  %Number of stepper
must fully revolute Z
realturnx(sz-1,1)=turnx(sz-1,1)*PR;  %Number of steps
that stepper must do X
realturny(sz-1,1)=turny(sz-1,1)*PR;  %Number of steps
that stepper must do Y
realturnz(sz-1,1)=turnz(sz-1,1)*PR;  %Number of steps
that stepper must do Z
```

APPENDIX-7: Sub Matlab Code "runtripteron"

```
maxstep=(150/ttl)*PR;
minrpm=50;
maxrpm=200;
py=1;

while py<sz
RPMX(py,1)=(abs(realturnx(py,1))*minrpm)/PR;
RPMY(py,1)=(abs(realturny(py,1))*minrpm)/PR;
RPMZ(py,1)=(abs(realturnz(py,1))*minrpm)/PR;
if RPMX(py,1)>=maxrpm
   RPMX(py,1)=maxrpm;
elseif RPMX(py,1)<=minrpm
   RPMX(py,1)=minrpm;
end

if RPMY(py,1)>=maxrpm
   RPMY(py,1)=maxrpm;
elseif RPMY(py,1)<=minrpm
    RPMY(py,1)=minrpm;
end
if RPMZ(py,1)>=maxrpm
   RPMZ(py,1)=maxrpm;
elseif RPMZ(py,1)<=minrpm
   RPMZ(py,1)=minrpm;
end
delayMicsecX(py,1)=(1/(RPMX(py,1)*6))*(stepangle)*1e6;
delayMicsecY(py,1)=(1/(RPMY(py,1)*6))*(stepangle)*1e6;
delayMicsecZ(py,1)=(1/(RPMZ(py,1)*6))*(stepangle)*1e6;
 py=py+1;
end
```

APPENDIX-8: Sub Matlab Code "loopvaluecalculator"

```
kh=1;

while kh<sz

mainloop(kh,1)=gcd(round(realturnx(kh,1)),gcd(round(realt
urny(kh,1)),round(realturnz(kh,1))));
    if mainloop(kh,1)==0
        loopx(kh,1)=0;
        loopy(kh,1)=0;
        loopz(kh,1)=0;
        else

loopx(kh,1)=abs(round(realturnx(kh,1))/mainloop(kh,1));

loopy(kh,1)=abs(round(realturny(kh,1))/mainloop(kh,1));

loopz(kh,1)=abs(round(realturnz(kh,1))/mainloop(kh,1));
    end

    kh=kh+1;
end
```

APPENDIX-9: Sub Matlab Code "leadtheway"

```
format longG
grManipulator=horzcat(mcx,mcy,mcz);
grStepact=round(horzcat(realturnx,realturny,realturnz));
grdelayMicsec=round(horzcat(delayMicsecX,delayMicsecY,del
ayMicsecZ));
mainloop;
grloopvalues=horzcat(loopx,loopy,loopz);

figure('Name','Error Data','NumberTitle','off');
plot(error);
xlabel('Number of Test')
ylabel('Error')

figure('Name','End Effector Path','NumberTitle','off');
plot3(mcx,mcy,mcz,'--or','LineWidth',1);
xlabel('Coordinate X')
ylabel('Coordinate Y')
zlabel('Coordinate Z')
box on

[X,Y]=meshgrid(0:50:150);
Z1=ones(4);Z1=Z1*0;
Z2=ones(4);Z2=Z2*75;
Z3=ones(4);Z3=Z3*150;

FOS1=[3.455518,3.155238,2.139948,2.344843;3.741647,2.9798
22,2.35572,2.347438;3.366769,2.354058,2.453176,2.487157;2
.725365,2.7629,2.337479,2.145712];
FOS2=[3.315726,2.673260,2.650327,2.424730;3.617711,3.1895
1,2.48558,2.428335;2.745854,2.385689,2.25,2.289611;2.7607
9,2.512680,2.191632,2.372771];
```

```
FOS3=[3.60171,2.782167,2.365208,2.399620;3.328458,2.64221
3,2.305737,2.084962;2.341992,2.450585,2.065295,1.940051;2
.281433,2.238832,2.215502,1.998946];
Sigma1=[62.219,68.141,100.47,101.31;57.461,72.152,91.3,91
.589;69.577,97.331,91.978,86.444;78.889,82.7005,99.678,10
0.2];
Sigma2=[64.843,80.426,81.122,88.67;59.43,67.2,86.5,88.538
;78.3,90.121,95.2,93.902;77.8762,85.566,98.1,105.75];
Sigma3=[59.7,77.278,90.901,89.598;64.594,81.371,93.246,10
3.12;91.802,87.734,104.1,110.82;85.585,91.097,97.043,107.
56];
Disp1=[4.39017,5.87795,8.42882,12.62111;4.98186,6.50389,9
.04,13.24291;6.28526,7.73299,9.7797,13.69557;10.19986,11.
9421,12.89035,13.67456];
Disp2=[4.33546,5.80844,7.25405,8.35215;4.95285,6.44,7.85,
8.81391;6.00916,7.40477,8.61,9.2215;7.45282,8.60174,9.417
18,9.65671];
Disp3=[4.37,5.85007,7.27108,8.37005;5.04891,6.49246,7.881
42,8.88097;6.0421,7.48911,8.74097,9.48378;7.61083,8.80482
,9.65232,9.8764];
Freq1=[58.82802,54.49824,58.12335,68.35083;56.27657,50.56
244,55.862,62.22261;60.03101,53.70969,58.61048,68.99489;6
9.32399,59.147,67.30082,85.82965];
Freq2=[62.53929,58.5826,62.48389,73.2844;61.79589,59.021,
62.344,71.29269;64.70587,62.29793,66.45,72.82217;72.134,6
7.27838,70.31713,86.83302];
Freq3=[69.38,65.60764,67.30749,76.42714;72.07026,66.85043
,67.45364,76.50078;74.76435,67.52454,66.35106,75.13898;79
.89684,71.34026,69.52203,78.65272];
figure('Name','Factor of Safety
Check','NumberTitle','off');
surf(X,Y,Z1,FOS1,'FaceAlpha',0.9,'FaceLighting','gouraud'
)
```

```
hold on
surf(X,Y,Z2,FOS2,'FaceAlpha',0.9,'FaceLighting','gouraud'
)
surf(X,Y,Z3,FOS3,'FaceAlpha',0.9,'FaceLighting','gouraud'
)
plot3(mcx,mcy,mcz,'-or','LineWidth',2);
xlabel('Coordinate X')
ylabel('Coordinate Y')
zlabel('Coordinate Z')
box on
hold off
ac1=colorbar;
caxis([1.94 3.8])
ac1.Label.String = 'Factor of Safety';
figure('Name','Stress Check','NumberTitle','off');
surf(X,Y,Z1,Sigma1,'FaceAlpha',0.9,'FaceLighting','gourau
d')
hold on
surf(X,Y,Z2,Sigma2,'FaceAlpha',0.9,'FaceLighting','gourau
d')
surf(X,Y,Z3,Sigma3,'FaceAlpha',0.9,'FaceLighting','gourau
d')
plot3(mcx,mcy,mcz,'-or','LineWidth',2);
xlabel('Coordinate X')
ylabel('Coordinate Y')
zlabel('Coordinate Z')
box on
hold off
ac2=colorbar;
caxis([55 110])
ac2.Label.String = 'Stress (N/mm^2)';
figure('Name','Displacement Check','NumberTitle','off');
```

```
surf(X,Y,Z1,Disp1,'FaceAlpha',0.9,'FaceLighting','gouraud
')
hold on
surf(X,Y,Z2,Disp2,'FaceAlpha',0.9,'FaceLighting','gouraud
')
surf(X,Y,Z3,Disp3,'FaceAlpha',0.9,'FaceLighting','gouraud
')
plot3(mcx,mcy,mcz,'-or','LineWidth',2);
xlabel('Coordinate X')
ylabel('Coordinate Y')
zlabel('Coordinate Z')
box on
hold off
ac3=colorbar;
caxis([5 10])
ac3.Label.String = 'Displacement (mm)';
figure('Name','Frequency1 Check','NumberTitle','off');
surf(X,Y,Z1,Freq1,'FaceAlpha',0.9,'FaceLighting','gouraud
')
hold on
surf(X,Y,Z2,Freq2,'FaceAlpha',0.9,'FaceLighting','gouraud
')
surf(X,Y,Z3,Freq3,'FaceAlpha',0.9,'FaceLighting','gouraud
')
plot3(mcx,mcy,mcz,'-or','LineWidth',2);
xlabel('Coordinate X')
ylabel('Coordinate Y')
zlabel('Coordinate Z')
box on
hold off
ac3=colorbar;
caxis([50 75])
ac3.Label.String = 'Frequency1 (Hz)';
```

APPENDIX-10: Sub Matlab Code “documentation”

```
mkdir Ardufiles;

tr=sz-1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
filespec0 = fullfile( 'Ardufiles', 'track.txt' );
fileID0 = fopen(filespec0,'w');
fprintf(fileID0,'%d;',tr);
fclose(fileID0);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
filespec1 = fullfile( 'Ardufiles','StepX.txt' );
fileID1 = fopen(filespec1,'w');
for i=1:tr
    fprintf(fileID1,'%d;\n',grStepact(i,1));
end
fclose(fileID1);

filespec2 = fullfile( 'Ardufiles','StepY.txt' );
fileID2 = fopen(filespec2,'w');
for i=1:tr
    fprintf(fileID2,'%d;\n',grStepact(i,2));
end
fclose(fileID2);

filespec3 = fullfile( 'Ardufiles','StepZ.txt' );
fileID3 = fopen(filespec3,'w');
for i=1:tr
    fprintf(fileID3,'%d;\n',grStepact(i,3));
end
fclose(fileID3);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
filespec4 = fullfile( 'Ardufiles','dlyX.txt' );
fileID4 = fopen(filespec4,'w');
for i=1:tr
    fprintf(fileID4,'%d;\n',grdelayMicsec(i,1));
end
fclose(fileID4);

filespec5 = fullfile( 'Ardufiles','dlyY.txt' );
fileID5 = fopen(filespec5,'w');
for i=1:tr
    fprintf(fileID5,'%d;\n',grdelayMicsec(i,2));
end
fclose(fileID5);

filespec6 = fullfile( 'Ardufiles','dlyZ.txt' );
fileID6 = fopen(filespec6,'w');
for i=1:tr
    fprintf(fileID6,'%d;\n',grdelayMicsec(i,3));
end
fclose(fileID6);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
filespec7 = fullfile( 'Ardufiles','mloop.txt' );
fileID7 = fopen(filespec7,'w');
for i=1:tr
   fprintf(fileID7,'%d;\n',mainloop(i,1));
end
fclose(fileID7);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
filespec8 = fullfile( 'Ardufiles','loopX.txt' );
```

```
fileID8 = fopen(filespec8,'w');
for i=1:tr
   fprintf(fileID8,'%d;\n',grloopvalues(i,1));
end
fclose(fileID8);

filespec9 = fullfile( 'Ardufiles','loopY.txt' );
fileID9 = fopen(filespec9,'w');
for i=1:tr
   fprintf(fileID9,'%d;\n',grloopvalues(i,2));
end
fclose(fileID9);

filespec10 = fullfile( 'Ardufiles','loopZ.txt' );
fileID10 = fopen(filespec10,'w');
for i=1:tr
   fprintf(fileID10,'%d;\n',grloopvalues(i,3));
end
fclose(fileID10);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%

T1=table(mcx,mcy,mcz);
T2=table(grStepact,grdelayMicsec,mainloop,grloopvalues);
T3=uitable(uifigure,'Data',T1,'Position',[150 150 250
150]);
T4=uitable(uifigure,'Data',T2,'Position',[0 0 550 400]);
```

APPENDIX-11: Sub Matlab Code "demonstrationcode"

```
a=1;
PR=6400; %Pulse/rev oranı sürücüden ayarladığımız değer
ttl=8;  %8mm for one full revolution of the screw
Trapezoidal Turn Lead T8
stepangle=360/PR;
maxdist = 150; %The maximum distance that can be reached
mindist = 0; %The minimum distance that can be reached
radimin = 20;
radimax = 150;
sens = 20;
pl=1;
kl=1;
maxstep=(maxdist/ttl)*PR; %not active
minrpm=50; %min hız
maxrpm=200; % max hız
clearence=20;
kh=2;
df=1;
ata=1;

answer = questdlg('Would you like to
draw?','Demonstration','YES','NO','YES');

   while(kl)
            switch answer
                  case 'YES'
                        waitfor(errordlg('Collision
Sensitivity is 20mm','WARNING'));
                        while(pl)
```

```
                        prompt = {'Centre Coordinate at X
axis','Centre Coordinate at Y axis','Centre Coordinate at
Z axis','Radious of the Drawing'};
                        title = 'Center Determination';
                        dims = [1 35];
                        definput = {'60','60','10','20'};
                        abc =
inputdlg(prompt,title,dims,definput);
                        xccent = str2num(abc{1});
                        yccent = str2num(abc{2});
                        zccent = str2num(abc{3});
                        radicircle = str2num(abc{4});
                            if(xccent<mindist ||
xccent>maxdist || yccent<mindist || yccent>maxdist ||
zccent<mindist || zccent>maxdist || radicircle<radimin ||
radicircle>radimax)
                            waitfor(errordlg('X Y Z
Coordinates must be between 0-150mm','WARNING'));
                            elseif((xccent-
radicircle)<mindist+sens || (yccent-
radicircle)<mindist+sens || (xccent+radicircle)>maxdist-
sens || (yccent+radicircle)>maxdist-sens)
                            waitfor(errordlg('Collision
detected','WARNING'));
                            else
                            break;
                            end
                        end

                        drch = questdlg('What would you
like to
draw?','Demonstration','Rectangle','Circle','Triangle','C
ircle');
```

```
switch drch
    case 'Rectangle'
        drawresolution=2;
    case 'Circle'
        drawresolution=20;
    case 'Triangle'
        drawresolution=3/2;
end
dtr = 0:pi/drawresolution:2*pi;
for kr=1:length(dtr)
    btr(1,kr)=zccent;
end
xunit = radicircle * cos(dtr) +
xccent;
yunit = radicircle * sin(dtr) +
yccent;
xcircle = round (xunit);
ycircle = round (yunit);
zcircle = round (btr);

if zccent>=100
    peakbegin=zccent;
else
    peakbegin=zccent+20;
end

fw=round(peakbegin/clearence);
cc=fw;
cdc=fw;

xmani(1,1)=0;
ymani(1,1)=0;
zmani(1,1)=0;
```

```
                        while fw>0
                        xmani(1,kh)=0;
                        ymani(1,kh)=0;
                        zmani(1,kh)=0+(clearence*df);
                        fw=fw-1;
                        kh=kh+1;
                        df=df+1;
                        end

                         for ppr=1:length(dtr)

xmani(1,ppr+cc+1)=xcircle(1,ppr);

ymani(1,ppr+cc+1)=ycircle(1,ppr);

zmani(1,ppr+cc+1)=zcircle(1,ppr);
                         end

                        df=df-1;
                        xmani(1,length(dtr)+cc+2)=0;
                        ymani(1,length(dtr)+cc+2)=0;

zmani(1,length(dtr)+cc+2)=clearence*(df);

                        while (cdc>0)
                        xmani(1,length(dtr)+cc+2+ata)=0;
                        ymani(1,length(dtr)+cc+2+ata)=0;

zmani(1,length(dtr)+cc+2+ata)=(clearence*df)-
(clearence*ata);
                        cdc=cdc-1;
                        ata=ata+1;
```

```
                    end


                    testsystem; % ANN

                    xlin=xmani*w;
                    ylin=ymani*w;
                    zlin=zmani*w;

                    for trd=1:length(xlin)-1

SpaceXcircle(1,trd)=xlin(1,trd+1)-xlin(1,trd); %Distance
that Linear System Should Move X

SpaceYcircle(1,trd)=ylin(1,trd+1)-ylin(1,trd); %Distance
that Linear System Should Move Y

SpaceZcircle(1,trd)=zlin(1,trd+1)-zlin(1,trd); %Distance
that Linear System Should Move Z


turnxcircle(1,trd)=SpaceXcircle(1,trd)/ttl;  %Number of
stepper must fully revolute X

turnycircle(1,trd)=SpaceYcircle(1,trd)/ttl;  %Number of
stepper must fully revolute Y

turnzcircle(1,trd)=SpaceZcircle(1,trd)/ttl;  %Number of
stepper must fully revolute Z


realturnxcircle(1,trd)=turnxcircle(1,trd)*PR;  %Number of
steps that stepper must do X
```

```
realturnycircle(1,trd)=turnycircle(1,trd)*PR;  %Number of
steps that stepper must do Y

realturnzcircle(1,trd)=turnzcircle(1,trd)*PR;  %Number of
steps that stepper must do Z
                        end

                        xtc = round(realturnxcircle);
                        ytc = round(realturnycircle);
                        ztc = round(realturnzcircle);

                        for ppyt=1:length(xtc)

RPMXcircle(1,ppyt)=(abs(xtc(1,ppyt))*minrpm)/PR;

RPMYcircle(1,ppyt)=(abs(ytc(1,ppyt))*minrpm)/PR;

RPMZcircle(1,ppyt)=(abs(ztc(1,ppyt))*minrpm)/PR;

                        if RPMXcircle(1,ppyt)>=maxrpm
                           RPMXcircle(1,ppyt)=maxrpm;
                        elseif RPMXcircle(1,ppyt)<=minrpm
                           RPMXcircle(1,ppyt)=minrpm;
                        end

                        if RPMYcircle(1,ppyt)>=maxrpm
                           RPMYcircle(1,ppyt)=maxrpm;
                        elseif RPMYcircle(1,ppyt)<=minrpm
                            RPMYcircle(1,ppyt)=minrpm;
                        end

                        if RPMZcircle(1,ppyt)>=maxrpm
```

```
                        RPMZcircle(1,ppyt)=maxrpm;
                    elseif RPMZcircle(1,ppyt)<=minrpm
                        RPMZcircle(1,ppyt)=minrpm;
                    end


xdelaycircle(1,ppyt)=(1/(RPMXcircle(1,ppyt)*6))*(stepangl
e)*1e6;

ydelaycircle(1,ppyt)=(1/(RPMYcircle(1,ppyt)*6))*(stepangl
e)*1e6;

zdelaycircle(1,ppyt)=(1/(RPMZcircle(1,ppyt)*6))*(stepangl
e)*1e6;
                    end

                    xdeltc=round(xdelaycircle);
                    ydeltc=round(ydelaycircle);
                    zdeltc=round(zdelaycircle);

                    for rtkh=1:length(xtc)

mlooptc(1,rtkh)=gcd(xtc(1,rtkh),gcd(ytc(1,rtkh),ztc(1,rtk
h)));
                        if mlooptc(1,rtkh)==0
                            lpxtc(1,rtkh)=0;
                            lpytc(1,rtkh)=0;
                            lpztc(1,rtkh)=0;
                            else

lpxtc(1,rtkh)=abs(xtc(1,rtkh)/mlooptc(1,rtkh));

lpytc(1,rtkh)=abs(ytc(1,rtkh)/mlooptc(1,rtkh));
```

```
lpztc(1,rtkh)=abs(ztc(1,rtkh)/mlooptc(1,rtkh));
                            end

                        end

                        stpcrc = vertcat(xtc,ytc,ztc);
                        dlycrc =
vertcat(xdeltc,ydeltc,zdeltc);
                        mlooptc;
                        sbloop=
vertcat(lpxtc,lpytc,lpztc);

                        figure('Name','Error
Data','NumberTitle','off');
                        plot(error);
                        xlabel('Number of Test')
                        ylabel('Error')

                        figure('Name','End Effector
Path','NumberTitle','off');
                        plot3(xmani, ymani, zmani,'--
or','LineWidth',1);
                        xlabel('Coordinate X')
                        ylabel('Coordinate Y')
                        zlabel('Coordinate Z')
                        box on

                        [X,Y]=meshgrid(0:50:150);
                        Z1=ones(4);Z1=Z1*0;
                        Z2=ones(4);Z2=Z2*75;
                        Z3=ones(4);Z3=Z3*150;
```

```
FOS1=[3.455518,3.155238,2.139948,2.344843;3.741647,2.9798
22,2.35572,2.347438;3.366769,2.354058,2.453176,2.487157;2
.725365,2.7629,2.337479,2.145712];

FOS2=[3.315726,2.673260,2.650327,2.424730;3.617711,3.1895
1,2.48558,2.428335;2.745854,2.385689,2.25,2.289611;2.7607
9,2.512680,2.191632,2.372771];

FOS3=[3.60171,2.782167,2.365208,2.399620;3.328458,2.64221
3,2.305737,2.084962;2.341992,2.450585,2.065295,1.940051;2
.281433,2.238832,2.215502,1.998946];

Sigma1=[62.219,68.141,100.47,101.31;57.461,72.152,91.3,91
.589;69.577,97.331,91.978,86.444;78.889,82.7005,99.678,10
0.2];

Sigma2=[64.843,80.426,81.122,88.67;59.43,67.2,86.5,88.538
;78.3,90.121,95.2,93.902;77.8762,85.566,98.1,105.75];

Sigma3=[59.7,77.278,90.901,89.598;64.594,81.371,93.246,10
3.12;91.802,87.734,104.1,110.82;85.585,91.097,97.043,107.
56];

Disp1=[4.39017,5.87795,8.42882,12.62111;4.98186,6.50389,9
.04,13.24291;6.28526,7.73299,9.7797,13.69557;10.19986,11.
9421,12.89035,13.67456];

Disp2=[4.33546,5.80844,7.25405,8.35215;4.95285,6.44,7.85,
8.81391;6.00916,7.40477,8.61,9.2215;7.45282,8.60174,9.417
18,9.65671];

Disp3=[4.37,5.85007,7.27108,8.37005;5.04891,6.49246,7.881
```

```
42,8.88097;6.0421,7.48911,8.74097,9.48378;7.61083,8.80482
,9.65232,9.8764];

Freq1=[58.82802,54.49824,58.12335,68.35083;56.27657,50.56
244,55.862,62.22261;60.03101,53.70969,58.61048,68.99489;6
9.32399,59.147,67.30082,85.82965];

Freq2=[62.53929,58.5826,62.48389,73.2844;61.79589,59.021,
62.344,71.29269;64.70587,62.29793,66.45,72.82217;72.134,6
7.27838,70.31713,86.83302];

Freq3=[69.38,65.60764,67.30749,76.42714;72.07026,66.85043
,67.45364,76.50078;74.76435,67.52454,66.35106,75.13898;79
.89684,71.34026,69.52203,78.65272];

                        figure('Name','Factor of Safety
Check','NumberTitle','off');

surf(X,Y,Z1,FOS1,'FaceAlpha',0.9,'FaceLighting','gouraud'
)
                        hold on

surf(X,Y,Z2,FOS2,'FaceAlpha',0.9,'FaceLighting','gouraud'
)

surf(X,Y,Z3,FOS3,'FaceAlpha',0.9,'FaceLighting','gouraud'
)
                        plot3(xmani, ymani, zmani,'-
or','LineWidth',2);
                        xlabel('Coordinate X')
                        ylabel('Coordinate Y')
                        zlabel('Coordinate Z')
                        box on
```

```
                        hold off
                        ac1=colorbar;
                        caxis([1.94 3.8])
                        ac1.Label.String = 'Factor of
Safety';

                        figure('Name','Stress
Check','NumberTitle','off');

surf(X,Y,Z1,Sigma1,'FaceAlpha',0.9,'FaceLighting','gourau
d')
                        hold on

surf(X,Y,Z2,Sigma2,'FaceAlpha',0.9,'FaceLighting','gourau
d')

surf(X,Y,Z3,Sigma3,'FaceAlpha',0.9,'FaceLighting','gourau
d')
                        plot3(xmani, ymani, zmani,'-
or','LineWidth',2);
                        xlabel('Coordinate X')
                        ylabel('Coordinate Y')
                        zlabel('Coordinate Z')
                        box on
                        hold off
                        ac2=colorbar;
                        caxis([55 110])
                        ac2.Label.String = 'Stress
(N/mm^2)';

                        figure('Name','Displacement
Check','NumberTitle','off');
```

```
surf(X,Y,Z1,Disp1,'FaceAlpha',0.9,'FaceLighting','gouraud
')
                        hold on

surf(X,Y,Z2,Disp2,'FaceAlpha',0.9,'FaceLighting','gouraud
')

surf(X,Y,Z3,Disp3,'FaceAlpha',0.9,'FaceLighting','gouraud
')
                        plot3(xmani, ymani, zmani,'-
or','LineWidth',2);
                        xlabel('Coordinate X')
                        ylabel('Coordinate Y')
                        zlabel('Coordinate Z')
                        box on
                        hold off
                        ac3=colorbar;
                        caxis([5 10])
                        ac3.Label.String = 'Displacement
(mm)';

                        figure('Name','Frequency1
Check','NumberTitle','off');

surf(X,Y,Z1,Freq1,'FaceAlpha',0.9,'FaceLighting','gouraud
')
                        hold on

surf(X,Y,Z2,Freq2,'FaceAlpha',0.9,'FaceLighting','gouraud
')
```

```
surf(X,Y,Z3,Freq3,'FaceAlpha',0.9,'FaceLighting','gouraud
')
                        plot3(xmani, ymani, zmani,'-
or','LineWidth',2);
                        xlabel('Coordinate X')
                        ylabel('Coordinate Y')
                        zlabel('Coordinate Z')
                        box on
                        hold off
                        ac3=colorbar;
                        caxis([50 75])
                        ac3.Label.String = 'Frequency1
(Hz)';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                        xmani=transpose(xmani);
                        ymani=transpose(ymani);
                        zmani=transpose(zmani);

                        stpcrc=transpose(stpcrc);
                        dlycrc=transpose(dlycrc);
                        mlooptc=transpose(mlooptc);
                        sbloop=transpose(sbloop);

                        mkdir Ardufiles;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
```

```
                        flespec0 = fullfile( 'Ardufiles',
'track.txt' );
                        fleID0 = fopen(flespec0,'w');

fprintf(fleID0,'%d;',length(xtc));
                        fclose(fleID0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
                        flespec1 = fullfile(
'Ardufiles','StepX.txt' );
                        fleID1 = fopen(flespec1,'w');
                        for i=1:length(xtc)

fprintf(fleID1,'%d;\n',stpcrc(i,1));
                        end
                        fclose(fleID1);

                        flespec2 = fullfile(
'Ardufiles','StepY.txt' );
                        fleID2 = fopen(flespec2,'w');
                        for i=1:length(xtc)

fprintf(fleID2,'%d;\n',stpcrc(i,2));
                        end
                        fclose(fleID2);

                        flespec3 = fullfile(
'Ardufiles','StepZ.txt' );
                        fleID3 = fopen(flespec3,'w');
                        for i=1:length(xtc)

fprintf(fleID3,'%d;\n',stpcrc(i,3));
```

```
                        end
                        fclose(fleID3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
                        flespec4 = fullfile(
'Ardufiles','dlyX.txt' );
                        fleID4 = fopen(flespec4,'w');
                        for i=1:length(xtc)

fprintf(fleID4,'%d;\n',dlycrc(i,1));
                        end
                        fclose(fleID4);

                        flespec5 = fullfile(
'Ardufiles','dlyY.txt' );
                        fleID5 = fopen(flespec5,'w');
                        for i=1:length(xtc)

fprintf(fleID5,'%d;\n',dlycrc(i,2));
                        end
                        fclose(fleID5);

                        flespec6 = fullfile(
'Ardufiles','dlyZ.txt' );
                        fleID6 = fopen(flespec6,'w');
                        for i=1:length(xtc)

fprintf(fleID6,'%d;\n',dlycrc(i,3));
                        end
                        fclose(fleID6);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
                        flespec7 = fullfile(
'Ardufiles','mloop.txt' );
                        fleID7 = fopen(flespec7,'w');
                        for i=1:length(xtc)

fprintf(fleID7,'%d;\n',mlooptc(i,1));
                        end
                        fclose(fleID7);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
                        flespec8 = fullfile(
'Ardufiles','loopX.txt' );
                        fleID8 = fopen(flespec8,'w');
                        for i=1:length(xtc)

fprintf(fleID8,'%d;\n',sbloop(i,1));
                        end
                        fclose(fleID8);

                        flespec9 = fullfile(
'Ardufiles','loopY.txt' );
                        fleID9 = fopen(flespec9,'w');
                        for i=1:length(xtc)

fprintf(fleID9,'%d;\n',sbloop(i,2));
                        end
                        fclose(fleID9);
```

```
                        flespec10 = fullfile(
'Ardufiles','loopZ.txt' );
                        fleID10 = fopen(flespec10,'w');
                        for i=1:length(xtc)

fprintf(fleID10,'%d;\n',sbloop(i,3));
                        end
                        fclose(fleID10);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
                        T1=table(xmani,ymani,zmani);

T2=table(stpcrc,dlycrc,mlooptc,sbloop);

T3=uitable(uifigure,'Data',T1,'Position',[150 150 250
150]);
T4=uitable(uifigure,'Data',T2,'Position',[0 0 550 400]);

                        waitfor(msgbox('All data
calculated and exported to "Ardufiles" file in
documentary','TripteronCalculator','custom',myicon));
                        break;
                case 'NO'
                        waitfor(msgbox('Program will shut
down','TripteronCalculator','custom',myicon));
                        break;
                otherwise
                        waitfor(msgbox('Program will shut
down','TripteronCalculator','custom',myicon));
                        break;
            end
   end
```

APPENDIX-12: Arduino Code

```
#include <SPI.h>
#include <SD.h>

File myFile1;
File myFile2;
File myFile3;
File myFile4;
File myFile5;
File myFile6;
File myFile7;
File myFile8;
File myFile9;
File myFile10;
File myFile11;

const int PULPinX = 8;
//Pulse pin X
const int DIRPinX = 9;
//Direction pin X
const int PULPinY = 6;
//Pulse pin Y
const int DIRPinY = 7;
//Direction pin Y
const int PULPinZ = 4;
//Pulse pin Z
const int DIRPinZ = 5;
//Direction pin Z
const int Mag = 3;
//Electromagnet pin

int endstopX=10;
```

```
int endstopY=11;
int endstopZ=12;

int homeXval;
int homeYval;
int homeZval;
int homedelay=40;  // 6400 pulse/rev için 40 ama 800
pulse/rev için 700

int track[1];
String inString = "";    // string to hold input
byte i=0;
int a=0;
int b=0;
int x;
int y;
int z;

void setup()
{
  pinMode(PULPinX,OUTPUT);
  pinMode(DIRPinX,OUTPUT);
  pinMode(PULPinY,OUTPUT);
  pinMode(DIRPinY,OUTPUT);
  pinMode(PULPinZ,OUTPUT);
  pinMode(DIRPinZ,OUTPUT);
  pinMode(Mag,OUTPUT);
  pinMode(endstopX,INPUT);
  pinMode(endstopY,INPUT);
  pinMode(endstopZ,INPUT);

  Serial.begin(9600);
  delay(100);
```

```
  while (!Serial) {}
  Serial.println("Initializing SD card...");
  if (!SD.begin(53)) {                          /* UNO-4
Mega-53 */
    Serial.println("initialization failed!");
    while (1);
  }
  Serial.println("initialization done!");
  Serial.println("");

  Serial.println("Electro Magnet LOW");
  digitalWrite(Mag,LOW);
  Serial.println("");
}

void loop()
{
      Serial.println("Hit S to START the tripteron");
       while(a==0)
      {
         if(Serial.available()>0)
         {
            char ans = Serial.read();
             if (ans==83)  // S
             {
              a=1;
              Serial.println("Starting the Process in 3
sec...");
              delay(3000);
             }
         }
      }
```

```
    homeXval = digitalRead(endstopX); //Anahtar Basınca 0
değerini döndürüyor
    homeYval = digitalRead(endstopY);
    homeZval = digitalRead(endstopZ);

    if(homeXval != 0)
    {
      Serial.println("Processing X Axis to homeX in 3
sec...");
      delay(3000);
      digitalWrite(DIRPinX,LOW);
      while(homeXval != 0)
      {
      digitalWrite(PULPinX,HIGH);
      delayMicroseconds(homedelay);
      digitalWrite(PULPinX,LOW);
      delayMicroseconds(homedelay);
      homeXval = digitalRead(endstopX);
      }
    Serial.println("X Axis at homeX");
    }
    else if(homeXval == 0)
    {
      Serial.println("X Axis at homeX");
    }

    if(homeYval != 0)
    {
      Serial.println("Processing Y Axis to homeY in 3
sec...");
      delay(3000);
      digitalWrite(DIRPinY,HIGH);
      while(homeYval != 0)
```

```
      {
      digitalWrite(PULPinY,HIGH);
      delayMicroseconds(homedelay);
      digitalWrite(PULPinY,LOW);
      delayMicroseconds(homedelay);
      homeYval = digitalRead(endstopY);
      }
    Serial.println("Y Axis at homeY");
    }
    else if(homeYval == 0)
    {
      Serial.println("Y Axis at homeY");
    }

    if(homeZval != 0)
    {
      Serial.println("Processing Z Axis to homeZ in 3
sec...");
      delay(3000);
      digitalWrite(DIRPinZ,HIGH);
      while(homeZval != 0)
      {
      digitalWrite(PULPinZ,HIGH);
      delayMicroseconds(homedelay);
      digitalWrite(PULPinZ,LOW);
      delayMicroseconds(homedelay);
      homeZval = digitalRead(endstopZ);
      }
    Serial.println("Z Axis at homeZ");
    }
    else if(homeZval == 0)
    {
      Serial.println("Z Axis at homeZ");
```

```
    }
    Serial.println();
  /*-----------------------------------------------------
-----------*/

    Serial.println("Hit A to START the SD Card Reading");
       while(a==1)
      {
         if(Serial.available()>0)
         {
            char ans = Serial.read();
             if (ans==65)
             {
              a=2;
              Serial.println("Starting the Process in 3
sec...");
              delay(3000);
             }
         }
      }

  /*-----------------------------------------------------
----------*/
   myFile1 = SD.open("track.txt");
    if (myFile1)
       {
         while (myFile1.available())
         {
           int inChar = myFile1.read();
            if (inChar != ';')
            {
              inString += (char)inChar;
            }
```

```
            else
            {
              track[i]=inString.toInt();
              i++;
              // clear the string for new input:
              inString = "";
            }
           }
          i=0;
          // close the file:
          myFile1.close();
        }
       else
        {
          // if the file didn't open, print an error:
          Serial.println("error opening track.txt");
        }
/*-------------------------------------------------------
---------*/
    double StepX[track[0]];
    double StepY[track[0]];
    double StepZ[track[0]];
    int dlyX[track[0]];
    int dlyY[track[0]];
    int dlyZ[track[0]];
    double mloop[track[0]];
    int loopX[track[0]];
    int loopY[track[0]];
    int loopZ[track[0]];
/*-------------------------------------------------------
-----------*/
         myFile2 = SD.open("StepX.txt");
    if (myFile2)
```

```
        {
          while (myFile2.available())
          {
            int inChar = myFile2.read();
             if (inChar != ';')
             {
               inString += (char)inChar;
             }
             else
             {
               StepX[i]=inString.toInt();
               i++;
               // clear the string for new input:
               inString = "";
             }
          }
           i=0;
          // close the file:
          myFile2.close();
        }
       else
        {
          // if the file didn't open, print an error:
          Serial.println("error opening StepX.txt");
        }
        delay(100);
/*-------------------------------------------------------
-----------*/
     myFile3 = SD.open("StepY.txt");
    if (myFile3)
       {
         while (myFile3.available())
         {
```

```
            int inChar = myFile3.read();
             if (inChar != ';')
             {
               inString += (char)inChar;
             }
             else
             {
               StepY[i]=inString.toInt();
               i++;
               // clear the string for new input:
               inString = "";
             }
          }
          i=0;
          // close the file:
          myFile3.close();
        }
      else
        {
          // if the file didn't open, print an error:
          Serial.println("error opening StepY.txt");
        }
        delay(100);
/*-------------------------------------------------------
------------------*/
     myFile4 = SD.open("StepZ.txt");
    if (myFile4)
       {
         while (myFile4.available())
         {
           int inChar = myFile4.read();
            if (inChar != ';')
            {
```

```
              inString += (char)inChar;
            }
            else
            {
              StepZ[i]=inString.toInt();
              i++;
              // clear the string for new input:
              inString = "";
            }
          }
          i=0;
          // close the file:
          myFile4.close();
        }
       else
        {
          // if the file didn't open, print an error:
          Serial.println("error opening StepZ.txt");
        }
        delay(100);
/*-----------------------------------------------------
------------------*/
     myFile5 = SD.open("dlyX.txt");
    if (myFile5)
       {
         while (myFile5.available())
         {
            int inChar = myFile5.read();
            if (inChar != ';')
            {
              inString += (char)inChar;
            }
            else
```

```
            {
              dlyX[i]=inString.toInt();
              i++;
              // clear the string for new input:
              inString = "";
            }
          }
          i=0;
          // close the file:
          myFile5.close();
        }
       else
        {
          // if the file didn't open, print an error:
          Serial.println("error opening dlyX.txt");
        }
/*-------------------------------------------------------
------------------*/
     myFile6 = SD.open("dlyY.txt");
    if (myFile6)
       {
         while (myFile6.available())
         {
           int inChar = myFile6.read();
            if (inChar != ';')
            {
              inString += (char)inChar;
            }
            else
            {
              dlyY[i]=inString.toInt();
              i++;
              // clear the string for new input:
```

```
            inString = "";
          }
        }
        i=0;
        // close the file:
        myFile6.close();
      }
     else
      {
        // if the file didn't open, print an error:
        Serial.println("error opening dlyY.txt");
      }
/*-------------------------------------------------------
------------------*/
     myFile7 = SD.open("dlyZ.txt");
    if (myFile7)
      {
        while (myFile7.available())
        {
          int inChar = myFile7.read();
           if (inChar != ';')
           {
             inString += (char)inChar;
           }
           else
           {
             dlyZ[i]=inString.toInt();
             i++;
             // clear the string for new input:
             inString = "";
           }
        }
        i=0;
```

```
          // close the file:
          myFile7.close();
        }
       else
        {
          // if the file didn't open, print an error:
          Serial.println("error opening dlyZ.txt");
        }
/*-------------------------------------------------------
------------------*/
     myFile8 = SD.open("mloop.txt");
    if (myFile8)
      {
        while (myFile8.available())
        {
          int inChar = myFile8.read();
           if (inChar != ';')
           {
             inString += (char)inChar;
           }
           else
           {
             mloop[i]=inString.toInt();
             i++;
             // clear the string for new input:
             inString = "";
           }
        }
        i=0;
        // close the file:
        myFile8.close();
      }
     else
```

```
        {
          // if the file didn't open, print an error:
          Serial.println("error opening mloop.txt");
        }
/*-----------------------------------------------------
------------------*/
     myFile9 = SD.open("loopX.txt");
    if (myFile9)
        {
          while (myFile9.available())
          {
            int inChar = myFile9.read();
             if (inChar != ';')
             {
               inString += (char)inChar;
             }
             else
             {
               loopX[i]=inString.toInt();
               i++;
               // clear the string for new input:
               inString = "";
             }
          }
          i=0;
          // close the file:
          myFile9.close();
        }
       else
        {
          // if the file didn't open, print an error:
          Serial.println("error opening loopX.txt");
        }
```

```
/*-------------------------------------------------------
------------------*/
     myFile10 = SD.open("loopY.txt");
    if (myFile10)
       {
         while (myFile10.available())
         {
           int inChar = myFile10.read();
            if (inChar != ';')
            {
              inString += (char)inChar;
            }
            else
            {
              loopY[i]=inString.toInt();
              i++;
              // clear the string for new input:
              inString = "";
            }
         }
         i=0;
         // close the file:
         myFile10.close();
       }
      else
       {
         // if the file didn't open, print an error:
         Serial.println("error opening loopY.txt");
       }
/*-------------------------------------------------------
------------------*/
     myFile11 = SD.open("loopZ.txt");
    if (myFile11)
```

```
        {
          while (myFile11.available())
          {
           int inChar = myFile11.read();
             if (inChar != ';')
             {
               inString += (char)inChar;
             }
             else
             {
               loopZ[i]=inString.toInt();
               i++;
               // clear the string for new input:
               inString = "";
             }
          }
          i=0;
          // close the file:
          myFile11.close();
        }
       else
        {
          // if the file didn't open, print an error:
          Serial.println("error opening loopZ.txt");
        }
/*-------------------------------------------------------
------------------*/
  Serial.println("/*-----------------------------------
-------------------------------------*/");
  Serial.print("track=");
  for(i=0;i<1;i++)
  {
    Serial.println(track[i]);
```

```
}
Serial.println();
Serial.println("Step");
Serial.print("X");
Serial.print("\t\t\t");
Serial.print("Y");
Serial.print("\t\t\t");
Serial.println("Z");
for(i=0;i<track[0];i++)
{
  Serial.print(StepX[i]);
  Serial.print("\t\t");
  Serial.print(StepY[i]);
  Serial.print("\t\t");
  Serial.println(StepZ[i]);
}
  Serial.println();
  Serial.println("dly");
  Serial.print("X");
  Serial.print("\t");
  Serial.print("Y");
  Serial.print("\t");
  Serial.println("Z");
  for(i=0;i<track[0];i++)
{
  Serial.print(dlyX[i]);
  Serial.print("\t");
  Serial.print(dlyY[i]);
  Serial.print("\t");
  Serial.println(dlyZ[i]);
}
  Serial.println();
  Serial.println("mloop");
```

```
  for(i=0;i<track[0];i++)
{
  Serial.println(mloop[i]);
}

  Serial.println();
  Serial.println("loop");
  Serial.print("X");
  Serial.print("\t");
  Serial.print("Y");
  Serial.print("\t");
  Serial.println("Z");
  for(i=0;i<track[0];i++)
{
  Serial.print(loopX[i]);
  Serial.print("\t");
  Serial.print(loopY[i]);
  Serial.print("\t");
  Serial.println(loopZ[i]);
}
Serial.println("/*-------------------------------------
-------------------------------------*/");
Serial.println();
Serial.println("Hit A to START Action");

     while(a==2)
    {
       if(Serial.available()>0)
       {
           char ans = Serial.read();
           if (ans==65)
           {
            a=0;
```

```
              Serial.println("Starting the Process  in 3
sec...");
              delay(3000);
             }
          }
       }
/*-------------------------------------------------------
----------*/
          for(i=0;i<track[0];i++)
             {
                if(i==2)
                {
                  Serial.println("Magnet ON");
                  digitalWrite(Mag,HIGH);
                }
                else if(i==(track[0]-2))
                {
                  Serial.println("Magnet OFF");
                  digitalWrite(Mag,LOW);
                  delay(10);
                }

                if (StepX[i]<0)
                {
                  digitalWrite(DIRPinX,LOW);   //Pin 9
                }
                else
                {
                  digitalWrite(DIRPinX,HIGH);  //Pin 9
                }

                if (StepY[i]<0)
                {
```

```
  digitalWrite(DIRPinY,HIGH);   //Pin 7
 }
 else
 {
  digitalWrite(DIRPinY,LOW);  //Pin 7
 }

 if (StepZ[i]<0)
 {
  digitalWrite(DIRPinZ,HIGH);   //Pin 5
 }
 else
 {
  digitalWrite(DIRPinZ,LOW);  //Pin 5
 }

  for(b=0;b<mloop[i];b++)
    {
      if(loopX[i]!=0)
      {
       for(x=0;x<loopX[i];x++)
       {
        digitalWrite(PULPinX,HIGH);
//Pin 8
        delayMicroseconds(dlyX[i]);
        digitalWrite(PULPinX,LOW);
//Pin 8
        delayMicroseconds(dlyX[i]);
       }
      }
      if(loopY[i]!=0)
      {
       for(y=0;y<loopY[i];y++)
```

```
                        {
                         digitalWrite(PULPinY,HIGH);
//Pin 6
                         delayMicroseconds(dlyY[i]);
                         digitalWrite(PULPinY,LOW);
//Pin 6
                         delayMicroseconds(dlyY[i]);
                        }
                       }
                       if(loopZ[i]!=0)
                       {
                        for(z=0;z<loopZ[i];z++)
                        {
                         digitalWrite(PULPinZ,HIGH);
//Pin 4
                         delayMicroseconds(dlyZ[i]);
                         digitalWrite(PULPinZ,LOW);
//Pin 4
                         delayMicroseconds(dlyZ[i]);
                        }
                       }
                     }
                Serial.print("Track(");
                Serial.print(i+1);
                Serial.print(")");
                Serial.print(" ");
                Serial.println("Processed!!!");
                delay(10);
              }

    Serial.println("Process Completed!!!!!!");
    Serial.println();
}
```

APPENDIX-13: Displacement estimation by ANN Matlab code

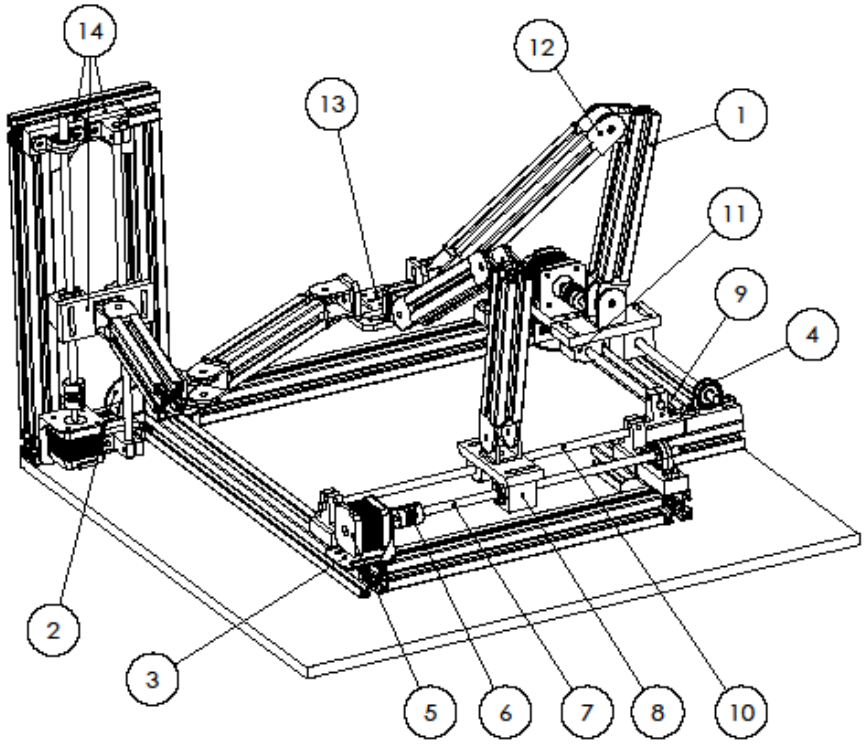
```
clc;clear;close all

% Solve an Input-Output Fitting problem with a Neural
Network
% Script generated by Neural Fitting app
% Created 03-Jul-2019 17:11:31
%
% This script assumes these variables are defined:
%
%   input - input data.
%   output - target data.
input =
[25,75,125,175,25,75,125,175,25,75,125,175,25,75,125,175,
25,75,125,175,25,75,125,175,25,75,125,175,25,75,125,175,2
5,75,125,175,25,75,125,175,25,75,125,175,25,75,125,175;25
,25,25,25,75,75,75,75,125,125,125,125,175,175,175,175,25,
25,25,25,75,75,75,75,125,125,125,125,175,175,175,175,25,2
5,25,25,75,75,75,75,125,125,125,125,175,175,175,175;25,25
,25,25,25,25,25,25,25,25,25,25,25,25,25,100,100,100,10
0,100,100,100,100,100,100,100,100,100,100,100,100,175,175
,175,175,175,175,175,175,175,175,175,175,175,175]
;
output =
[4.39017,5.87795,8.42882,12.62111,4.98186,6.50389,9.04,13
.24291,6.28526,7.73299,9.7797,13.69557,10.19986,11.9421,1
2.89035,13.67456,4.33546,5.80844,7.25405,8.35215,4.95285,
6.44,7.85,8.81391,6.00916,7.40477,8.61,9.2215,7.45282,8.6
0174,9.41418,9.65671,4.37,5.85007,7.27108,8.37005,5.04891
,6.49246,7.88142,8.88097,6.0421,7.48911,8.74097,9.48378,7
.61083,8.80482,9.65232,9.8764];
```

```
x = input;
t = output;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for
challenging problems.
% 'trainscg' uses less memory. Suitable in low memory
situations.
trainFcn = 'trainbr';  % Bayesian Regularization
backpropagation.

% Create a Fitting Network
hiddenLayerSize = 50;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation,
Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)

% View the Network
view(net)
```

```
% Plots
% Uncomment these lines to enable various plots.
figure, plotperform(tr)
figure, plottrainstate(tr)
figure, ploterrhist(e)
figure, plotregression(t,y)
figure, plotfit(net,x,t)
```

APPENDIX-14: Technical Drawing of the Tripteron Robot

ITEM NO	PART	QUANTITY
1	Sigma Profile 30x30	-
2	Sigma Profile 30x30 Connection Link	12
3	Nema 17 Stepper Gripper	3
4	KP08 Bearing	3
5	Nema 17 Stepper	3
6	Coupling 5x8mm	3
7	Trapezoidal Lead Screw T8 300mm	3
8	Aluminum Block Shaft	3
9	SK8 Aluminum Shaft Gripper	6
10	Aluminum Shaft 8x350mm	3
11	SCE 8mm Linear Bearing	6
12	Sigma Profile 30x30 Aluminum Joint	9
13	3D Printed End Effector	1
14	Aluminum Machined Parts	12



	NAME	SIGN	DATE	MATERIAL:	Tripteron Robot	A4
DESIGN	Serdar SAYIN					
CHECK	Zeki KIRAL				SCALE: 1:5	PAGE 1 / 1

Figure A.1 Technical drawing of the tripteron robot

APPENDIX-15: Technical Drawing of the 3D Printed End Effector

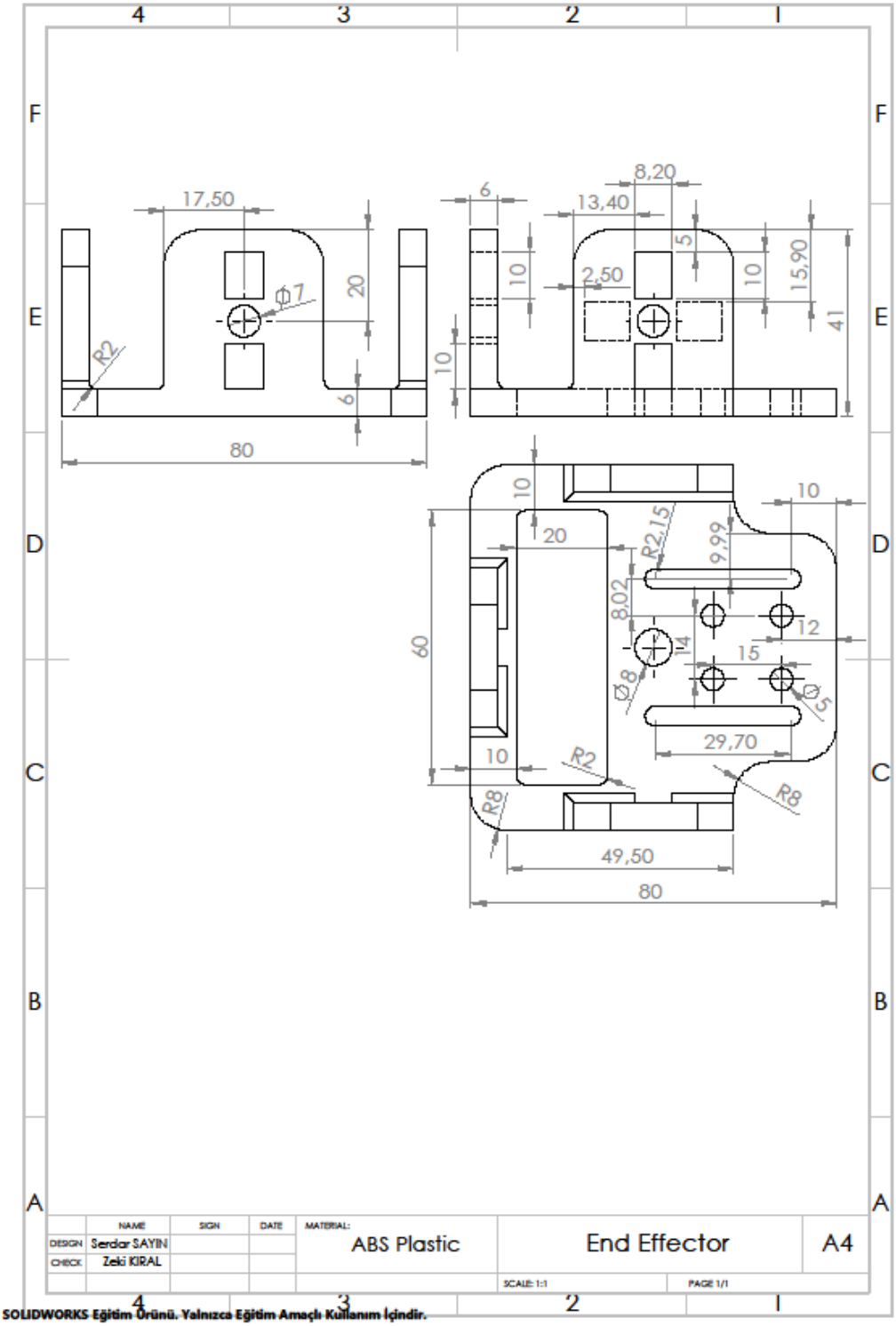


Figure A.2 Technical drawing of the end effector