İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010

FACULTY OF ENGINEERING AND ARCHITECTURE

MECHATRONICS ENGINEERING DEPARTMENT

DEVELOPMENT OF A MULTI DEGREES OF FREEDOM REDUNDANT RECONFIGURABLE PLANAR PARALLEL MANIPULATOR WITH 2 DOF PLANAR DETACHABLE SERIAL DYADS

GRADUATION PROJECT

BASHEER ALTAWIL 140412046

SUPERVISOR: Assist.Prof.Dr. Osman AKIN

CO-SUPERVISOR: Assist.Prof.Dr. Osman AKIN

MAY 2019

DEVELOPMENT OF A MULTI DEGREES OF FREEDOM REDUNDANT RECONFIGURABLE PLANAR PARALLEL MANIPULATOR WITH 2 DOF PLANAR DETACHABLE SERIAL DYADS

GRADUATION PROJECT

BASHEER ALTAWIL   140412046

Submitted to the MECHATRONICS ENGINEERING DEPARTMENT of

İZMİR KATİP ÇELEBİ UNIVERSITY

GROUP MEMBERS

SARAN SAPMAZ	ESRA UÇAR
ADEM CANDEMİR	

ABSTRACT

The main purpose of this work is to have a mechanism with three serial manipulators which can be used separately or synchronized like parallel manipulators. One of the constraints is to have a maximum footprint workspace when it's the most compact position within a circle radius 800mm. The other aim is to have maximum dexterous workspace. This paper contains both serial and parallel manipulators' working principle, and direct and inverse task calculations with the methodology. The mechanism of this project's paper is a redundant system, because the system has 6 motors, but DOF is only 3. As another constraint, actuators can be only DC motors. So, this paper contains information about servo-mechanism systems. To control the motors' positions, velocities and torques, Ziegler-Nichols method is mentioned. Also, these manipulators should reach the platform and if there is green obstacle, they should escape and arrive the leg of the platform. Also, this study contains information about image processing, path generation and lock systems. Image processing is for to define the green obstacle and distinguish the other colors, it lets the system find its way with respect to the obstacle. Path generation is used to produce new tracking functions by defining the last position and the first position. On the other hand, while the gripper holds the platform, it should stay still within gripper. If there is no lock system, the gripper behaves as passive joint, and the position of platform cannot be determined. As conclusion, to have such robot mechanism, there should be a task, calculations of workspace, proper link lengths, shape, connections and materials, proper motor that handles the system, calculations for redundant system, simulations and mathematic programs that can solve matrices, electrical circuit, and proper control algorithm.

Keywords: redundant, multi degree of freedom, reconfigurable

ÖZET

Bu çalışmanın temel amacı, ayrı olarak kullanılabilen veya paralel manipülatörler gibi senkronize edilebilen üç seri manipülatöre sahip bir mekanizmaya sahip olmaktır. Kısıtlamalardan biri, 800 mm'lik bir daire yarıçapı içindeki en kompakt konum olduğunda, maksimum ayak izi çalışma alanına sahip olmaktır. Diğer amaç, azami çalışma alanına sahip olmaktır. Bu makale hem seri hem de paralel manipülatörlerin çalışma prensibini ve metodoloji ile ileri ve geri görev hesaplamalarını içerir. Bu projenin mekanizması 6 motora sahiptir, ancak DOF sadece 3'tür. Başka bir kısıtlama olarak, aktüatörler sadece DC motorlar olmak zorunda. Bu nedenle, bu makale servo mekanizma sistemleri hakkında bilgi içermektedir. Motorların pozisyonlarını, hızlarını ve torklarını kontrol etmek için Ziegler-Nichols metodundan bahsedilmiştir. Ayrıca, bu manipülatörler platforma ulaşmalı ve yeşil bir engel varsa engelden kaçmalı ve platformun kolonuna ulaşmalıdır. Ayrıca, bu çalışma görüntü işleme, yol oluşturma ve kilit sistemleri hakkında bilgi içerir. Görüntü işleme, yeşil engeli tanımlamak ve diğer renkleri ayırt etmek içindir, sistemin engele göre yolunu bulmasını sağlar. Yol oluşturma, son konumu ve ilk konumu tanımlayarak yeni yollar üretmek için kullanılır. Öte yandan, kıskaç platformu tutarken, kıskacın içinde sabit kalmalıdır. Kilit sistemi yoksa, tutucu pasif bağlantı görevi görür ve platformun konumu belirlenemez. Sonuç olarak, böyle bir robot mekanizmasına sahip olmak için bir görev olmalı, çalışma alanı hesaplamaları, uygun uzunluklar, şekil, bağlantılar ve malzemeler, sistemi idare eden uygun motor, yedek sistem için hesaplamalar, matrisleri çözebilecek simülasyonlar ve matematik programları, elektrik devresi ve uygun kontrol algoritması belirlenmelidir
.

Anahtar Kelimeler: manipülator, Ziegler-Nichols, görüntü işleme

TABLE OF CONTENTS

ABSTRACT………………………………………………………………………………………………………3
ÖZET………4
I.INTRODUCTION………………………………………………………………………………………………7
1.1. METHODS………………………………………………………………………………………………10
1.2.Purpose Of The Project……………………………………………………………………………………11
II.ANALYSIS……………………………………………………………………………………………………12
2.1. Workspace Analysis………………………………………………………………………………………12
2.2. Kinematic Analysis…………………………………………………………………………………………13
  2.2.1 D-H Analaysis……………………………………………………………………………………13
  2.2.2 Forward Kinamatics………………………………………………………………………………14
  2.2.3 Invert The Jacobian………………………………………………………………………………15
  2.2.4 Inverse Kinematics………………………………………………………………………………16
2.3 Dynamic Analysis……………………………………………………………………………………………18
  2.3.1 For Serial Case……………………………………………………………………………………18
  2.3.2 For Parallel Case …………………………………………………………………………………20
2.4.TRAJECTORY ANALYSIS……………………………………………………………………………………21
  2.4.1.Verification Of The Mathematica Code …………………………………………………………21
III.DESIGN………………………………………………………………………………………………………25
3.1. Manipulator Design and Simulation………………………………………………………………………29
3.2.Hexagonal Design……………………………………………………………………………………………29
3.3.Gripper Design……………………………………………………………………………………………… 30
3.4. Camera Handle Arm Design…………………………………………………………………………………30
3.5.Assembly………………………………………………………………………………………………………31
  3.5.1.For Graduation Project-1……………………………………………………………………………31
  3.5.2.For Graduation Project-2…………………………………………………………………………… 31
4.ELECTRONICS…………………………………………………………………………………………… 32
4.1.Actuator Selection With Respect to Dynamic Analysis……………………………………………… 32
4.2.Potatiometer Selection and Linearization………………………………………………………… 32
4.3.Design and Manufacture of Motor Driver Cards………………………………………………… 34
  4.3.1.For Graduation Project 1……………………………………………………………………………34
  4.3.2.For Graduation Project 2……………………………………………………………………………43

5.CONTROL……46

5.1.Path Generation and Verification by using Solidworks…………………………………………………………….46

5.2.Pure Rotation Of Platform………………………………………………………………………………………………50

5.3.Drawing Square(No Obstacle)…………………………………………………………………………………………..51

5.4 Drawing Square(With Obstacle)………………………………………………………………………………………..53

5.5.Image Processing………..55

CONCLUSION………..58

REFERENCES……59

A. APPENDIX1 (TRAJECTORY GENERATION TASK CODE)

B.APPENDIX2 (COMMUNİCATE BETWEEN RAPBERRY Pİ AND ATMEGA MICROCONTROLLER-CODE)

C. APPENDIX3 (MASTER CODE PYTHON-CODE)

D. APPENDIX4 (SLAVE CODE-ARDUINO ATMEGA)

E. APPENDIX5 (PURE ROTATION TASK-CODE)

F. APPENDIX6 (DRAWING SQUARE TASK-CODE)

1. INTRODUCTION

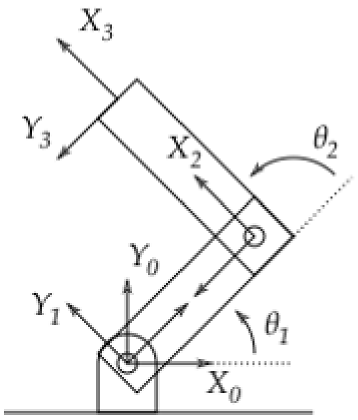


Figure 1.1 serial manipulator

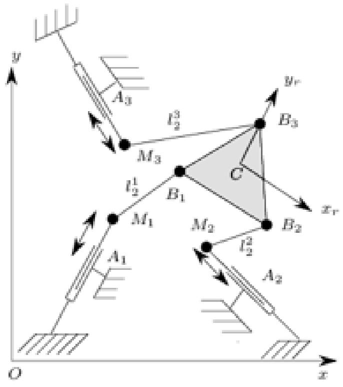


Figure 1.2 parallel manipulator

Serial and parallel manipulators are commonly used in industrial area. A serial manipulator (Fig1) consists of several links connected in series by various types of joints, typically revolute and prismatic joints. One end of the manipulator is attached to the ground and the other end is free to move in space. 1 A parallel manipulator (Fig2) is closed loop kinematic chain mechanism. Both have advantages and disadvantages. Serial manipulator has more workspace than parallel manipulator, in the other side, it is less accurate than parallel manipulator. Direct force transformation calculations are well defined for parallel manipulators. Direct kinematics are much harder in parallel mechanism due to their closed chain mechanism.

Figure-1.3SCARA

After that information, it can be talked about SCARA (Fig 3) (Selective Compliant Assembly Robot Arm) mechanism. It is a type of serial manipulator, in this study both joints are revolute. SCARA mechanism is getting popular day by day, because it covers less area, is faster, and easier to manufacture than Cartesians robot. Usage area are commonly packaging, pick-and-up. The reason choosing this mechanism in this study is the shape. The shape of mechanism is like human-arm and for reaching large areas there is no need too long arm-lengths.2 In this study, this is also a need to reach the largest area and with some constraints of link lengths.

The main purpose of those all study to design three equal serial manipulators and not only use as serial manipulators but also as parallel manipulators. There is a plate, and one of the gripper endings is always touching the plate. In this case, gripper design also is important, it should be detectable and attachable in case needed. Plus, the mechanism should have 3 DoF in the end, but with 6 motors which means it is a redundant system. A redundant mechanism is one that contains more degrees of freedom than are needed to perform a given task. Redundant mechanisms can solve a given primary task in an infinite number of ways.

It is just like human arm, in real life if a human arm (without wrist) had only 6 DoF, it would be enough to survive.But having 7 DoF helps the human being to move the arm to somewhere even when there are obstacles. A human arm is considered to have 7 DOF. A shoulder gives pitch, yaw and roll, an elbow allows for pitch, and a wrist allows for pitch, yaw and roll. Only 3 of those movements would be necessary to move the hand to any point in space, but people would lack the ability to grasp things from different angles or directions.2 In the mechanism, it is the same thing, this is why redundant systems are needed for. This feature lets the system avoiding the obstacles and arrive where it should go.

These manipulators have some workspaces. There are two defined workspaces: dexterous and reachable. This information defines the volume of the mechanism, that's why it is a critical criterion. The dexterous workspace can be defined as the area where a manipulator is able to reach with any orientation.5 It is like a human hand, it can still hold in the same position with different orientation of the wrist. The reachable workspace means that the maximum volume that the manipulator can reach. As an example, again the maximum volume of the human arm is when it is all straight, and the arm cannot move forward. There is the last point the human can reach. In SolidWorks, these workspaces can be defined as circles that is used as formulas in Mathematica. While choosing the link length there are some important things to take care, such as increasing the link length gives the more workspace but in the other hand it increases the mass of the system which cause also torque increases and so does deflection possibility. Also, in this study, there is a constraint that in most compact, the links must stay within a circle 800mm radius. Another constraint is that up to 2.5mm deflection is allowed. As a conclusion, there is a limitation to increase the link length. So, to find the optimized link lengths, formulas of circles are helping. Easily, it can be seen the possible link lengths and their areas. The other important thing after choosing the lengths, the shape of links. Because when shape is changing, the inertia changes so does torque calculations and force calculations. For motor selection, those calculations are done in methods part, but briefly definition of torque is for mechanical systems, (…)a measure of how much a force acting on an object causes that object to rotate. Links itself is a mass, and it causes torque because of gravity, but also torque is calculated due to some payload.

Torque calculations are done to find proper DC motor for the system. In this study, DC motor is used and plus a potentiometer is added to have a servo-mechanism. A servo motor itself gives the position and velocity feedback, a servo-mechanism is made of DC motor and potentiometer or encoders which gives the position feedback to a microcontroller. One of the main thing to do is get position feedback by getting analog values of potentiometer with respect to angles, and have a characteristic. To convert 0-1023 to 0-3600, the only thing should be done to multiply the analog value with 3600/1023.

To have sync motion the only thing is done to have a coupling between potentiometer and DC motor shaft. To drive this system bidirectionally, it is needed to use H-bridge, and to make this powerful it is made of MOS-FETs.

Connections of links are done with proper bearings and belts. To have these connections, motor is selected, and its all dimensions are drawn in SolidWorks to fix the motor properly and to have proper housings. Also, they are assembled and observed the motion by using SolidWorks. The other good side of SolidWorks is that the materials can be defined, so that the simulation gets more realistic. Before manufacturing it is the most important thing to see things so real, because it is manufactured once, also all torque and force calculations are done by usingthis usage of SolidWorks. The program can provide the mass of the material with respect to chosen material and volume, and it makes possible to choose motor before manufacturing the parts.

To control algorithm is made by help Ziegler-Nichols 2nd method of tuning. It suggests that first overshoot and second overshoot should have a relationship 1:4. It is to be sure that system is not in unstable region. After being sure, some Kp, Ki and Kd parameters should be given to the system and found a system response due to the type of controller. In this article, P and PI controller are mentioned. Basically, P controller is used to find Ku, and after that multiplying this value with 0.5, Kp value is to be found. In this point, one is sure that the system is stable but, with some error value. That's why extra parameters are needed. PI controller has a response that with overshoot but without steady state error.

On the other hand, legs should avoid the green obstacle, and reach the hexagonal platform. Image processing is widely used in many areas such as security area, agriculture, face-identifications and etc. In this paper it is used to escape from a green cubic obstacle, and reach the gripper at all costs. For obstacle avoiding there are lots of sensors like ultrasonic distance sensor, IR sensor and etc., but they are not enough to define a color. That's why, image processing is held in this paper by using Phyton. As it will be mentioned more detailed, image processing procedure is done thanks to OpenCV library. This library allows users to do several things with the camera logo adding, distinguish a color, add pictures to other pictures can be given as applications. In this project, one of the crucial things is that with only one camera, any point of inside workspace should be seen, and define the grippers' new position with respect to the obstacle. So, the most proper position of the camera is the middle of the robot.

Path generation allows the gripper to track its way. When it across such an obstacle, thanks to this generation, it can change its way to reach the point. Basically, the last point is known and so the first point also, then the gripper defines the via points and track its way.

After the gripper reaches the leg of platform, another important term is being mentioned, "lock mechanism". Gripper is holding the leg, but if there is no lock mechanism, the coordinates of the object cannot be defined, and gripper starts to swing freely. To avoid this bad situation, a lock mechanism on gripper is designed. There are two possible ways to lock the system, first one is to lock the wrist of gripper, and the second one is to lock platform's leg itself. In this paper, the second one is valid.

Lastly, in this paper PIC and Atmega circuits are mentioned. So are the connection between them. Briefly, both circuits (for only one cart) can drive two motors at the same time. Main thing that PIC has only two pwm pins, and Atmega has more than two. Their methods to drive are mentioned. Also the problems, advantages and disadvantages between them are held.

As conclusion, to have such project many things should be mentioned and taken care of. Choosing the proper motors, link lengths, connection types, lock mechanism are the mechanic part of the project. Circuits and communication is the electronic part of the project. Lastly, image processing, path generation, forward-inverse kinematic analysis are the control part of the project. All topics above are separately important, and detailed in this study.

METHODS

At the beginning, to start analyzing, it is needed to calculate the links' positions parametrically. There are two things to do in this case; direct analysis and inverse analysis. Direct analysis of a serial manipulator means, joints' angles are known and the position of the end effector should be found. DH matrices are used to define end effector's position by using four elements. The algorithm of DH matrices is to fill up a table which consists four elements respectively to see all possibilities, those elements are translation, rotation of frame, offset and rotation of the links. The reason there is four elements is, some of axis have constraints. Xi intersects with Zi-1, and xi is perpendicular to zi-1. First of all, z axis is labelled. Then, x axis should be found by right-hand rule. After that, if any offsets exist (distances between x axis), lastly angles between z axis and x axis should be found. The table is ready to be filled up. After this, the translation and rotation matrices should be constructed, and then, the direct position analysis is found, if it is derived once, the velocity formulas are found. Taking the derivative of velocity formulas, the acceleration formulas can be obtained. In the velocity part, there is another important thing, Jacobian matrices. Jacobian matrices are constructed by using these velocity formulas. They are crucial to find out at which points the system will go under singularity. In engineering, a mechanical singularity is a position or configuration of a mechanism or a machine where the subsequent behavior cannot be predicted, or the forces or other physical quantities involved become infinite or nondeterministic.4 For serial manipulators, it loses 1 or more degree of freedom, and for parallel manipulators, it gains 1 degree of freedom which can not be controlled at all. In this paragraph, the serial singularities will be examined. One way is to get in singularity, the arm should be all extended form because in this manner, the arm can not move forward, or backward. So, it loses one degree of freedom. The other way to find the singularity, if the determinant of Jacobian is equal to zero, this means the Jacobian matrix can not be inversed, and as a result the velocities of joints can not be determined.

$$V=J(\theta)\,\dot{\theta}$$

$$\dot{\theta} = J(\theta)^{-1}v$$

$$V=J(\theta)^{-1}\dot{\theta}$$

Then, the inverse kinematics of serial should be done. This is more difficult than direct kinematics because this time the end effector positions are known, but joint angles are unknown (Atc 5). After that, dynamic analysis is done. This time, forces and torques are in role. One of these constraints is, the system should carry at least 0.200 kg, and it should not deflect more than 2.5 mm. In this analysis, mass and link lengths should be found. Mass is related to with the type of material, link length and thickness. To find the link length, another constraint should be obeyed which is footprint of workspace can not pass 800mm in most compact way. At this point, SolidWorks is used (Fig 4). Also, in SolidWorks the manipulators are placed like equilateral triangle.

The challenge is to avoid a green obstacle, and reach the hexagonal platform. One of the methodology is image processing. By using OpenCV library, green color is identified very well. Also the place of the camera is important, there is other design to fix the cam. While escaping, the method is called path generation is used. In this generation, with respect to initial and final points some equations are produced. By solving these equations, some via points are defined. This method lets robot to decide its way in constraint conditions. Forward and inverse kinematics are effectively used to drive the robot. Even Rasperry Pi and Atmega can communicate each other in both sides, the main programs are written in Rasperry Pi such as inverse-forward kinematics, path generation.

Also, a lock system should be used to clarify the position of hexagonal platform. First of all, it is designed in SolidWorks, one of them is chosen. Another thing, the legs of hexagonal are stripped by sandpaper to increase its friction while gripper is holding the leg. To create this system, worm gear is used. As soon as gripper is attached to hexagonal, the servo inside the platform works and make the leg locked.

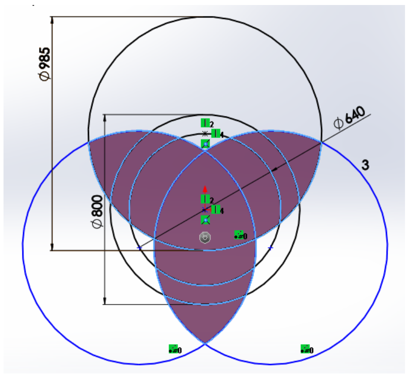
1.2.PURPOSE OF THE PROJECT

With this project, it is aimed to be able to use three 2-degree series of free-range manipulators with automatic disconnection and integration with various configurations as both serial and parallel manipulators. The system, which will be designed with this feature, will find its place in many markets from artificial agriculture to industrial applications. In order to perform the tasks in the mass production lines in the industrial area, the changing time of the apparatus connected to the end point can be reduced to seconds and a wide variety of configurations can be made according to the need of a single mechanism. For example; When two or three jobs need to be done at the same time in a factory's mass production line, the robot will be separated from the arms and the process will be started once the required platform is automatically mounted and the configurations are done. In another example, parallel robotic requirements will be required in processes that require more precise positioning or simulation, in this case two arms or three arms will be automatically connected to the appropriate platform and a completely different operation will be made with the same robots. The biggest feature that makes the project frigid and important is that the robotic arms can do multivariate tasks alone or together.

2.ANALYSIS

2.1 WORKSPACE ANALYSIS

In the dexterous area analysis, the algorithm was developed in the mathematica program in order to find the maximum dexterous area . But in mathematica code,workspace is found as shown in Figure-1.1.But dexterous workspace is inside of the 800 mm diameter circle so dexterous workspace was not found by using mathematica. Instead, link lengths and robot foot positions were determined by changing the link lengths and the position of the robot feet with a certain iteration to maximize the dexterous area where sufficient PC1 value and three robot arms would work together.



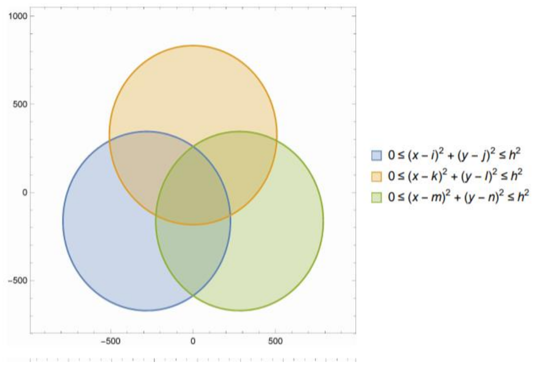


Figure 2.1. SolidWorks demonstration

Figure.2.1.2 Mathematica Workspace of Mathematica

Link lengths and robot foot positions were obtained as a result of the iterations we made in order to obtain PC1 value higher than 0.8 and to maximize the area where the three robot arms would work together.

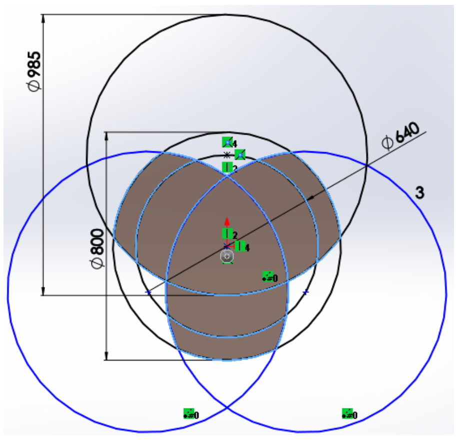


Figure-2.1.3 Dexterous workspace of robot manipulator

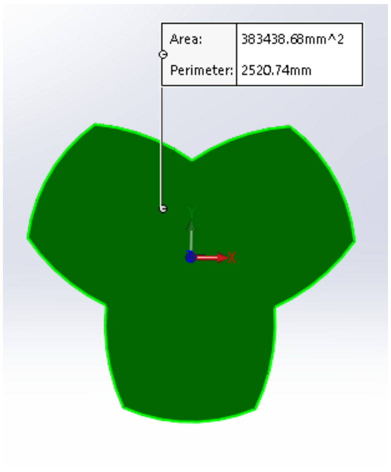


Figure-2.1.4

Dexterous Area:383438 mm^2

Link lenths; l_1=220 mm

l_2=220 mm

Hexagonal shape dimension: 60 mm

PC1=Dexterous workspace / (0.8*Footprint Area)=383438/(0.8*pi*400^2)=0,95

2.2 KINAMATIC ANALYSIS

2.2.1 DENAVIT-HARTENBERG ANALYSIS

D-H analysis is used to define end effector position. To create those matrices, a table including 4 parameters are filled. Those parameters are, translation, rotation of frame(?), offset and rotation of the links(?). First matrices for all links stand for translation among x1 and y1 axis. Second and forth matrices stand for the rotation motion of the links around z axis. The last matrices to define its last position, which is a matrix including the length of b.

Table 1. DH parameters for 1st manipulator

i	ai	α	s	θ
1	x1,y1	0	0	Θ1
2	a	0	0	Θ2

$$ {}_3^0T = \begin{bmatrix} 1 & 0 & 0 & x1 \\ 0 & 1 & 0 & y1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\Theta 1 & -s\Theta 1 & 0 & 0 \\ s\Theta 1 & \mathrm{c}\Theta 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\Theta 2 & -s\Theta 2 & 0 & 0 \\ s\Theta 2 & \mathrm{c}\Theta 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b \\ 0 \\ 0 \\ 1 \end{bmatrix} $$

Table 2. DH parameters for 2nd manipulator

i	ai	α	s	ψ
1	x2,y2	0	0	ψ1
2	a	0	0	ψ2

$$ {}_3^0T = \begin{bmatrix} 1 & 0 & 0 & x1 \\ 0 & 1 & 0 & y1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\psi 1 & -s\psi 1 & 0 & 0 \\ s\psi 1 & \mathrm{c}\psi 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\psi 2 & -s\psi 2 & 0 & 0 \\ s\psi 2 & \mathrm{c}\psi 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b \\ 0 \\ 0 \\ 1 \end{bmatrix} $$

Table 3. DH parameters for 3rd manipulator

i	ai	α	s	Q
1	x3,y3	0	0	Q1
2	a	0	0	Q2

$$ {}^{0}_{3}T = \begin{bmatrix} 1 & 0 & 0 & x1 \\ 0 & 1 & 0 & y1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cQ1 & -sQ1 & 0 & 0 \\ sQ1 & cQ1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cQ2 & -sQ2 & 0 & 0 \\ sQ2 & cQ2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b \\ 0 \\ 0 \\ 1 \end{bmatrix} $$

For all links, forward kinematics should be done. Here it is supposed that angles are known and link lengths should be found. Respectively position, velocity and Accelaration analysis are done. By derivating the formulas of position, the velocity formulas are found. And the same, by derivating the velocity formulas, the Accelaration formulas are found.

2.2.2 FORWARD KINEMATICS

-POSITION ANALYSIS

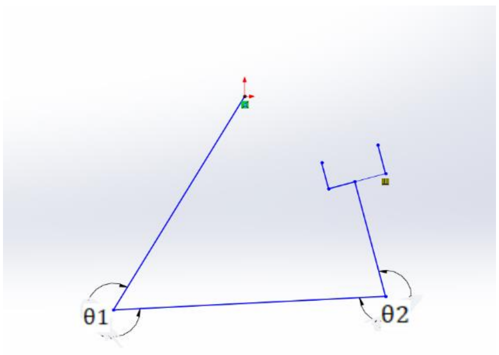


Figure 2.2.2

1ST manipulator's formulations:

$$ x = x1 + b * ((c\theta1 * c\theta2) - (s\theta1 * s\theta2)) + a * c\theta1 $$

$$ y = y1 + b * ((c\theta1 * s\theta2) + (c\theta2 * s\theta1)) + a * s\theta1 $$

2ND manipulator's formulations:

$$ x = x2 + b * ((c\Psi1 * c\Psi2) - (s\Psi1 * s\Psi2)) + a * c\Psi1 $$

$$ y = y2 + b * ((c\Psi1 * s\Psi2) + (c\Psi2 * s\Psi1)) + a * s\Psi1 $$

3RD manipulator's formulations:

$$ x = x3 + b * ((cQ1 * cQ2) - (sQ1 * sQ2)) + a * cQ1 $$

$$ y = y3 + b * ((cQ1 * sQ2) + (cQ2 * sQ1)) + a * sQ1 $$

-VELOCITY ANALYSIS

$$\dot{x} = b * [-s(\theta1 + \theta2) * (\dot{\theta 1} + \dot{\theta 2})] - a * s\theta1 * \dot{\theta 1}$$

$$\dot{x} = -b * \dot{\theta 1} * s(\theta1 + \theta2) - b * \dot{\theta 2} * s(\theta1 + \theta2) - a * s\theta1 * \dot{\theta 1}$$

$$\dot{y} = b * [c(\theta1 + \theta2) * (\dot{\theta 1} + \dot{\theta 2})] + a * c\theta1 * \dot{\theta 1}$$

$$\dot{y} = b * \dot{\theta 1} * c(\theta1 + \theta2) + b * \dot{\theta 2} * c(\theta1 + \theta2) + a * c\theta1 * \dot{\theta 1}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -a * s\theta1 - b * s(\theta1 + \theta2) & -b * s(\theta1 + \theta2) \\ a * c\theta1 + b * c(\theta1 + \theta2) & b * c(\theta1 + \theta2) \end{bmatrix} \begin{bmatrix} \dot{\theta 1} \\ \dot{\theta 2} \end{bmatrix}$$

$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = tip\ velocity\ matrix$

$\begin{bmatrix} -a * s\theta1 - b * s(\theta1 + \theta2) & -b * s(\theta1 + \theta2) \\ a * c\theta1 + b * c(\theta1 + \theta2) & b * c(\theta1 + \theta2) \end{bmatrix} = Jacobian\ matrix$

$\begin{bmatrix} \dot{\theta 1} \\ \dot{\theta 2} \end{bmatrix} = joints\ velocity$

For a robotic arm, there are two possible ways to describe its current position: end-effector and set of joint angles. The Jacobian system relates how movement of the joints causes movement of the elements of joints.

2.2.3 INVERT THE JACOBIAN

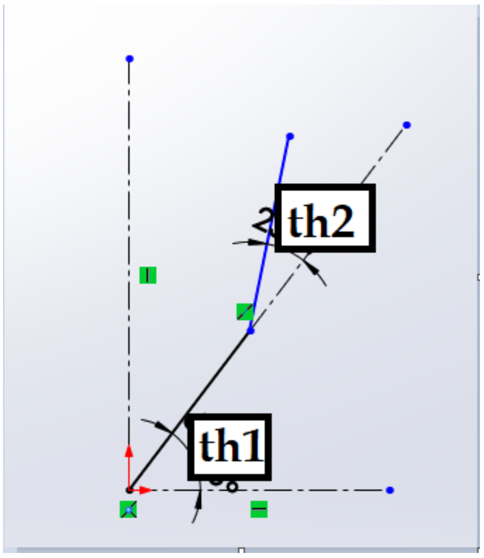


Figure.2.2.3

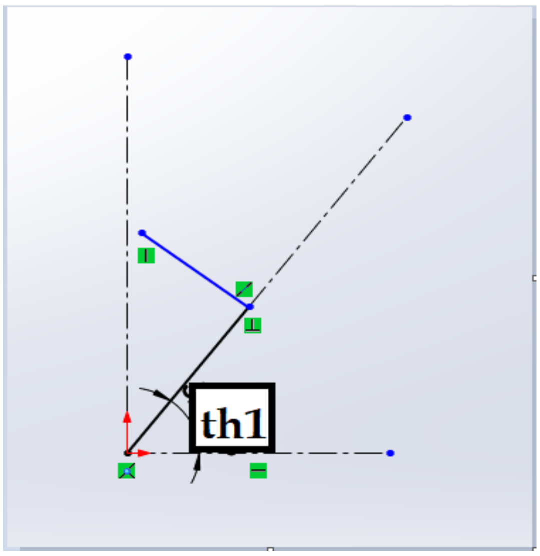


Figure.2.2.3

When $\theta2(\text{th2}) = 0$ it will be infinite solution which means singularity. In the second figure it can be observed that position. It is impossible to move first link forward.

Also if jacobian determinant is small, this occurs when motion is difficult and akward, and jacobian constant is too large which can be determined by using MatLab, the mechanism will go under singularity.

$$V=J(\theta)\,\dot{\theta}$$

$$\dot{\theta} = J(\theta)^{-1}v$$

$$V=J(\theta)^{-1}\dot{\theta}$$

From previous matrix:

$$J(\theta)^{-1} = \frac{1}{ab * s\theta 2}\begin{bmatrix} b * c(\theta 1+\theta 2) & b * s(\theta 1+\theta 2) \\ -a * c(\theta 1) - b * c(\theta 1+\theta 2) & -a * s(\theta 1) - b * s(\theta 1+\theta 2) \end{bmatrix}$$

2.2.4 INVERSE KINEMATICS

$\mathrm{x} - \mathrm{xi} = f$
y-yi=L
$\theta 1 + \theta 2 = \emptyset$

$$x - xi = b * \left(c(\theta 1 + \theta 2)\right) + \mathrm{a} * \mathrm{c}(\theta 1)$$

$$y - yi = b * \left(s(\theta 1 + \theta 2)\right) + \mathrm{a} * \mathrm{s}(\theta 1)$$

f= b*cØ + a*cθ1

l = b*sØ + a*sθ1

Take square and sum them.

$$f^2 + l^2 = b^2c^2\emptyset + 2\mathrm{ba} * \mathrm{c}\emptyset\mathrm{c}\,\theta 1 + a^2c^2\theta 1 + b^2s^2\emptyset + 2\mathrm{ba} * \mathrm{s}\emptyset\mathrm{s}\theta 1 + a^2s\theta 1$$

$$f^2 + l^2 = b^2 + a^2 + 2\mathrm{ba}(\mathrm{c}\emptyset\mathrm{c}\,\theta 1 + \mathrm{s}\emptyset\mathrm{s}\theta 1)$$

$$f^2 + l^2 = b^2 + a^2 + 2\mathrm{ba} * \mathrm{c}\,\theta 2$$

$$\theta 2 = \mathrm{ArcCos}[\frac{-a^2 - b^2 + f^2 + l^2}{2ab}]$$

$$\theta 1 = \mathrm{ArcTan}[\frac{l}{\sqrt{a^2 + b^2 - l^2 + 2ab\mathrm{Cos}[\theta 2]}}] - \mathrm{ArcTan}[\frac{b\mathrm{Sin}[\theta 2]}{a + b\mathrm{Cos}[\theta 2]}]$$

after simplification of derivative with respect to time by using both mathematic and MATLAB programs we could get the final results as follow:

Angular Velocities:

$$\boldsymbol{\dot{\theta}2} = -\frac{0.5(2.f * f' + 2.l * l')}{ab\sqrt{1. - \frac{0.25(-1.a^2 - 1.b^2 + f^2 + l^2)^2}{a^2b^2}}}$$

$$\dot{\boldsymbol{\theta}}\mathbf{1} = -\frac{\frac{b\mathrm{Cos}[\theta 2] * \theta 2'}{a + b\mathrm{Cos}[\theta 2]} + \frac{b^2\mathrm{Sin}[\theta 2]^2\theta 2'}{(a + b\mathrm{Cos}[\theta 2])^2}}{1 + \frac{b^2\mathrm{Sin}[\theta 2]^2}{(a + b\mathrm{Cos}[\theta 2])^2}} + \frac{\frac{l'}{\sqrt{a^2 + b^2 + 2ab\mathrm{Cos}[\theta 2] - l^2}} - \frac{l * (-2l * l' - 2ab\mathrm{Sin}[\theta 2] * \theta 2')}{2(a^2 + b^2 + 2ab\mathrm{Cos}[\theta 2] - l^2)^{3/2}}}{1 + \frac{l^2}{a^2 + b^2 + 2ab\mathrm{Cos}[\theta 2] - l^2}}$$

Angular accelerations:

$$\mathbf{\theta 2dd} = -\frac{0.125(-1.a^2 - 1.b^2 + f^2 + l^2)(2.f * \dot{f}' + 2.l * l')^2}{a^3b^3(1. - \frac{0.25(-1.a^2 - 1.b^2 + f^2 + l^2)^2}{a^2b^2})^{3/2}} - \frac{0.5(2.f'^2 + 2.l'^2 + 2.f * f'' + 2.l * l'')}{ab\sqrt{1. - \frac{0.25(-1.a^2 - 1.b^2 + f^2 + l^2)^2}{a^2b^2}}}$$

$$\mathbf{\theta 1dd} = \frac{(\frac{b\mathrm{Cos}[\theta 2] * \theta 2'}{a + b\mathrm{Cos}[\theta 2]} + \frac{b^2\mathrm{Sin}[\theta 2]^2 * \theta 2'}{(a + b\mathrm{Cos}[\theta 2])^2})(\frac{2b^2\mathrm{Cos}[\theta 2] * \mathrm{Sin}[\theta 2] * \theta 2'}{(a + b\mathrm{Cos}[\theta 2])^2} + \frac{2b^3\mathrm{Sin}[\theta 2]^3 * \theta 2'}{(a + b\mathrm{Cos}[\theta 2])^3})}{(1 + \frac{b^2\mathrm{Sin}[\theta 2]^2}{(a + b\mathrm{Cos}[\theta 2])^2})^2}$$

$$- ((\frac{2l * l'}{a^2 + b^2 + 2ab\mathrm{Cos}[\theta 2] - l^2} - \frac{l^2(-2l * l' - 2ab\mathrm{Sin}[\theta 2] * \theta 2')}{(a^2 + b^2 + 2ab\mathrm{Cos}[\theta 2] - l^2)^2})(\frac{l'}{\sqrt{a^2 + b^2 + 2ab\mathrm{Cos}[\theta 2] - l^2}} - \frac{l * (-2l * l}{2(a^2 + b^2 +}$$

$$- \frac{1}{1 + \frac{b^2\mathrm{Sin}[\theta 2]^2}{(a + b\mathrm{Cos}[\theta 2])^2}}(\frac{3b^2\mathrm{Cos}[\theta 2] * \mathrm{Sin}[\theta 2] * \mathrm{q}2'^2}{(a + b\mathrm{Cos}[\theta 2])^2} - \frac{b\mathrm{Sin}[\theta 2] * \theta 2'^2}{a + b\mathrm{Cos}[\theta 2]} + \frac{2b^3\mathrm{Sin}[\theta 2]^3 * \mathrm{q}2'^2}{(a + b\mathrm{Cos}[\theta 2])^3}$$

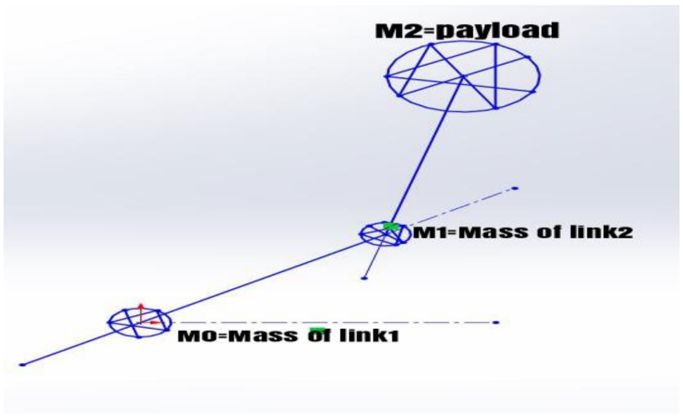
$$+ \frac{b\mathrm{Cos}[\theta 2]\theta 2''}{a + b\mathrm{Cos}[\theta 2]} + \frac{b^2\mathrm{Sin}[\theta 2]^2\theta 2''}{(a + b\mathrm{Cos}[\theta 2])^2})$$

$$+ (-\frac{l' * (-2l * l' - 2ab\mathrm{Sin}[\theta 2] * \mathrm{q}2')}{(a^2 + b^2 + 2ab\mathrm{Cos}[\theta 2] - l^2)^{3/2}} + \frac{3l * (-2l * l' - 2ab\mathrm{Sin}[\theta 2] * \theta 2')^2}{4(a^2 + b^2 + 2ab\mathrm{Cos}[\theta 2] - l^2)^{5/2}} + \frac{l''}{\sqrt{a^2 + b^2 + 2ab\mathrm{Cos}[\theta 2] - l^2}} -$$

After simplification of forward and inverse kinamatik we give some trajectory points for forward one and get some values for the joint values . Then, we put that values in inverse kinamatic formulations and we got the same results which we object them in forward kinamatic.In concolusion we could be sure that our calculation is totally correct in kinamatic part.

2.3 DYNAMIC ANALYSIS

2.3.1 FOR SERIAL CASE



m_0, m_1 ,and m_2 = mass of links and payload respectively

I_i = moment of inertia wrt center of mass

a,and b = mass center of links,and payload coordinates to the intial points

Figure-2.3.1

POSITION ANALYSIS

$x1 = a * c\theta 1$

$y1 = a * s\theta 1$

$x2 = a * c\theta 1 + b * c(\theta 1 + \theta 2)$

$y2 = a * s\theta 1 + b * s(\theta 1 + \theta 2)$

VELOCITY ANALYSIS

$\dot{x}1 = -a * \dot{\theta}1 * s\theta 1$ $\quad$ $\dot{x}2 = -\dot{\theta}1[as\theta 1 + bs(\theta 1 + \theta 2)] - b\dot{\theta}2s(\theta 1 + \theta 2)$

$\dot{y}1 = -a * \dot{\theta}1 * c\theta 1$ $\quad$ $\dot{y}2 = \dot{\theta}1[ac\theta 1 + bc(\theta 1 + \theta 2)] - b\dot{\theta}2c(\theta 1 + \theta 2)$

NOTATION: $\theta 1 + \theta 2 = \psi$

According to Robotic Analysis book kinetic energy formula is:

$$Ki = \frac{1}{2} * vi^2 * mi + \frac{1}{2} * I * wi^2$$

For our system the center of the mass is at the joint, so that the rotation will be about the end of the rod.

For link(1) CM:

$K0 = \frac{1}{2} * I * \dot{\theta}1^2$:Only inertia play rule in this case

$K0 = \frac{1}{2} * \dot{\theta}1^2 * \left(\frac{1}{3}m0 * a^2\right) = \left(\frac{1}{6}m0 * a^2\dot{\theta}1^2\right)$

For link(2) CM :

$$K1 = \frac{1}{2}m1 * a^2 * \dot{x1}^2 + \frac{1}{2} * I * \dot{\theta 1}^2$$

$$K1 = \frac{1}{2}m1 * a^2\dot{\theta 1}^2 + \frac{1}{2} * \dot{\theta 1}^2 * \left(\frac{1}{3}m1 * a^2\right) = \left(\frac{2}{3}m1 * a^2\dot{\theta 1}^2\right)$$

For payload CM :

For $\dot{x2}^2$

$$-\dot{\theta}1(as(\theta 1) + bs\,\psi) - \dot{\theta}2 * bs\,\psi$$

$$-\dot{\theta}1(as(\theta 1) + bs\,\psi) - \dot{\theta}2 * bs\,\psi$$

$$-\dot{\theta}1^2(a^2s^2\theta 1 + 2abs\psi s\theta 1 + b^2s^2\,\psi) + b^2\dot{\theta}2s^2\psi + 2ab\,\dot{\theta}1\dot{\theta}2s\psi(as\theta 1 + bs\psi)$$

After simplification of $\dot{x2}^2$ and $\dot{y2}^2$ we got

$\dot{x2}^2 = \dot{\theta}1^2a^2s^2\theta 1 + \dot{\theta}1^2(2ab\,s\psi\,s\theta 1) + \dot{\theta}1^2b^2s^2\,\psi + \dot{\theta}1\dot{\theta}2(2ab\,s\psi\,s\theta 1) + \dot{\theta}1\dot{\theta}2(2b^2s^2\psi)$

$\dot{y2}^2 = \dot{\theta}1^2a^2c^2\theta 1 + \dot{\theta}1^2(2ab\,c\psi\,c\theta 1) + \dot{\theta}1^2b^2c^2\,\psi + \dot{\theta}1\dot{\theta}2(2ab\,c\psi\,c\theta 1) + \dot{\theta}1\dot{\theta}2(2b^2c^2\psi)$

$$Total - Kinatic\ energy = \frac{1}{2}v2^2m2 + \frac{1}{2}\,I\,w2^2 = \frac{1}{2}m2\left(\dot{x2}^2 + \dot{y2}^2\right) + \frac{1}{6}\,\dot{\theta}2^2m2$$

$$= \frac{1}{2}\left(\frac{1}{6}m0 + \frac{2}{3}m1 + m2\right)\dot{\theta}1^2a^2 + \frac{1}{2}m2\,b^2\dot{\theta}1^2 + \frac{2}{3}m2\,b^2\dot{\theta}2^2 + m2\,b^2\dot{\theta}1\dot{\theta}2 + m2\,a\,b\,c\theta 2(\dot{\theta}1\dot{\theta}2 + \dot{\theta}1^2)$$

By using lagrang principles for link(1);

H = KE – PE and PE = 0 T1,T2 = $\frac{d}{dt}\left(\frac{\lambda H}{\lambda\dot{\theta}1}\right) - \frac{\lambda H}{\lambda\theta 1}$

$$\frac{\lambda H}{\lambda\dot{\theta}1} = \left(\frac{1}{6}m0 + \frac{2}{3}m1 + m2\right)a^2\dot{\theta}1 + m2\,b^2(\dot{\theta}1 + \dot{\theta}2) + m2\,a\,b\,c\theta 2(\dot{\theta}2 + 2\dot{\theta}1)$$

$$\frac{d}{dt}\left(\frac{\lambda H}{\lambda\dot{\theta}1}\right) = \left[\left(\frac{1}{6}m0 + \frac{2}{3}m1 + m2\right)a^2 + m2\,b^2 + 2m2\,ab\,c\theta 2\right]\ddot{\theta}1 + [m2\,b^2 + m2\,a\,b\,c\theta 2]\ddot{\theta}2 - (m2\,a\,b\,s\theta 2)\dot{\theta}2 - (2m2\,a\,b\,s\theta 2)\dot{\theta}1\dot{\theta}2$$

$$\frac{\lambda H}{\lambda\theta 1} = 0$$

For link(2):

$$\frac{\lambda H}{\lambda\dot{\theta}2} = m2\,b^2\left(\frac{2}{3}\dot{\theta}2 + \dot{\theta}1\right) + m2\,a\,b\,c\theta 2(\dot{\theta}1)$$

$$\frac{d}{dt}\left(\frac{\lambda H}{\lambda\dot{\theta}2}\right) = m2\,b^2\left(\frac{2}{3}\ddot{\theta}2 + \ddot{\theta}1\right) + m2\,a\,b\,c\theta 2\,\ddot{\theta}1 - m2\,a\,b\,s\theta 2\,\dot{\theta}1\,\dot{\theta}2$$

$$\frac{\lambda H}{\lambda\theta 2} = -\left(-m2\,a\,b\,s\theta 2\left(\dot{\theta}1\,\dot{\theta}2 + \dot{\theta 1}^2\right)\right)$$

$$\begin{bmatrix} T1 \\ T2 \end{bmatrix} = \begin{bmatrix} \left(\frac{1}{6}m0 + \frac{2}{3}m1 + m2\right)a^2 + m2\,b^2 + 2m2\,ab\,c\theta 2 & m2\,b^2 + m2\,a\,b\,c\theta 2 \\ m2\,a\,b\,c\theta 2 + m2\,b^2 & \frac{2}{3}m2\,b^2 \end{bmatrix} \begin{bmatrix} \ddot{\theta 1} \\ \ddot{\theta 2} \end{bmatrix} + \begin{bmatrix} 0 & -m2\,a\,b\,s\theta 2 \\ m2\,a\,b\,s\theta 2 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta 1}^2 \\ \dot{\theta 2}^2 \end{bmatrix} + \begin{bmatrix} -m2\,a\,b\,s\theta 2 & -m2\,a\,b\,s\theta 2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta 1}\dot{\theta 2} \\ \dot{\theta 1}\dot{\theta 2} \end{bmatrix}$$

2.3.2 FOR PARALLEL CASE

Dynamic model of the parallel manipulator with constant coefficients can be defined for each serial kinematic chain.

$$\alpha_i = J_{ai} + m_{ai}r_{ai} + m_{bi}d^2$$

$$\beta_i = J_{bi} + m_{bi}{r_{bi}}^2$$

$$(g)i = m_{bi}d\,r_{bi}$$

Combine the dynamics of 3 serial chains and consider the constraint forces due to the closed-loop constraints, the dynamic model of the parallel manipulator can be calculated as shown:

$$m\ddot{\theta} + c\dot{\theta} = To + A^T(Ln)$$

$$where\ \theta = [\theta_{11}, \theta_{12}, \theta_{13}, \theta_{21}, \theta_{22,}\theta_{23}]^{\,T} = vector\ joint\ position$$

$$\mathrm{To} = [\mathrm{To}_{11}, \mathrm{To}_{12}, \mathrm{To}_{13}, \mathrm{To}_{21,}\mathrm{To}_{22,}\mathrm{To}_{23}]^{\,T} = vector\ of\ input\ torques$$

$A^T(Ln)$ refers to the vector of constraint forces. The inertia Matrix M and Corilis matrix C as defined for parallel case:

$$M = \begin{bmatrix} \alpha_1 & 0 & 0 & (g)1c_{12-1} & 0 & 0 \\ 0 & \alpha_2 & 0 & 0 & (g)2c_{12-2} & 0 \\ 0 & 0 & \alpha_3 & 0 & 0 & (g)3c_{12-3} \\ (g)1c_{12-1} & 0 & 0 & \beta_1 & 0 & 0 \\ 0 & (g)2c_{12-2} & 0 & 0 & \beta_2 & 0 \\ 0 & 0 & (g)3c_{12-3} & 0 & 0 & \beta_3 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & (g)1s_{12-1}\dot{\theta}_{2-1} & 0 & 0 \\ 0 & 0 & 0 & 0 & (g)2s_{12-2}\dot{\theta}_{2-2} & 0 \\ 0 & 0 & \alpha_3 & 0 & 0 & (g)3s_{12-3}\dot{\theta}_{2-3} \\ -(g)1s_{12-1}\dot{\theta}_{1-1} & 0 & 0 & \beta_1 & 0 & 0 \\ 0 & -(g)2s_{12-2}\dot{\theta}_{1-2} & 0 & 0 & \beta_2 & 0 \\ 0 & 0 & -(g)3s_{12-3}\dot{\theta}_{1-3} & 0 & 0 & \beta_3 \end{bmatrix}$$

$$\boxed{\begin{gathered} c_{12-i} = \cos(\theta 1(\mathrm{i}) - \theta 2(\mathrm{i})) \\ s_{12-i} = \sin(\theta 1(\mathrm{i}) - \theta 2(\mathrm{i})) \end{gathered}}$$

So the constraint force $A^T(Ln)$ is included explicitly to guarantee the closed-loop constraints of the parallel manipulator are satisfied at every instant.

Matrix A is differential of the closed-loop constrained equation and multiplier (Ln) denotes the magnitude of constraint forces.

2.4 TRAJECTORY ANALYSIS

In trajectory analysis.a trajection path is created in solidworks as shown in Figure-1.4. X and y coordinates of all points on the generated path were taken and written to excel (Figure-1.4.1).

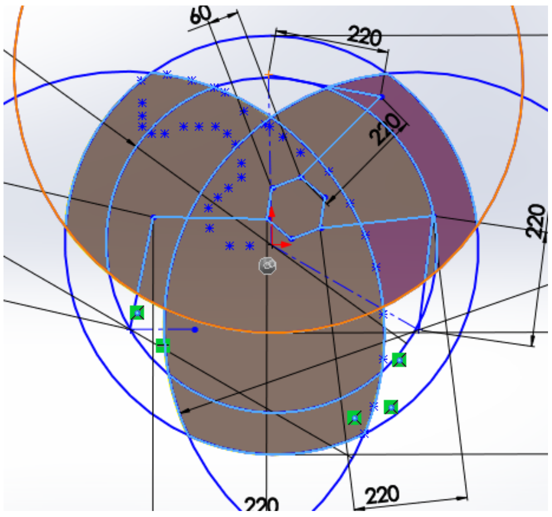
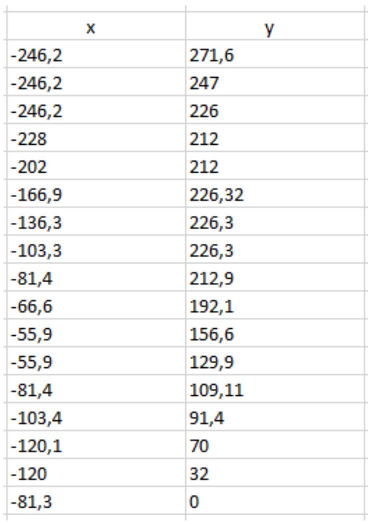


Figure-2.4



x	y
-246,2	271,6
-246,2	247
-246,2	226
-228	212
-202	212
-166,9	226,32
-136,3	226,3
-103,3	226,3
-81,4	212,9
-66,6	192,1
-55,9	156,6
-55,9	129,9
-81,4	109,11
-103,4	91,4
-120,1	70
-120	32
-81,3	0

Figure-2.4.1

X and Y equations are obtained with respect to time from Figure-1.4.2. The reason for obtaining the fifth order polynomial equation is to minimize the error by increasing the precision points.so;

X(t) = 0.0051*t^5-0.1765*t^4+1.6861*t^3-1.9407*t^2-6.1531*t-239.36

Y(t) = -0.0037*t^5+0.1941*t^4-3.7616*t^3+31.482*t^2-113.41*t+363.72

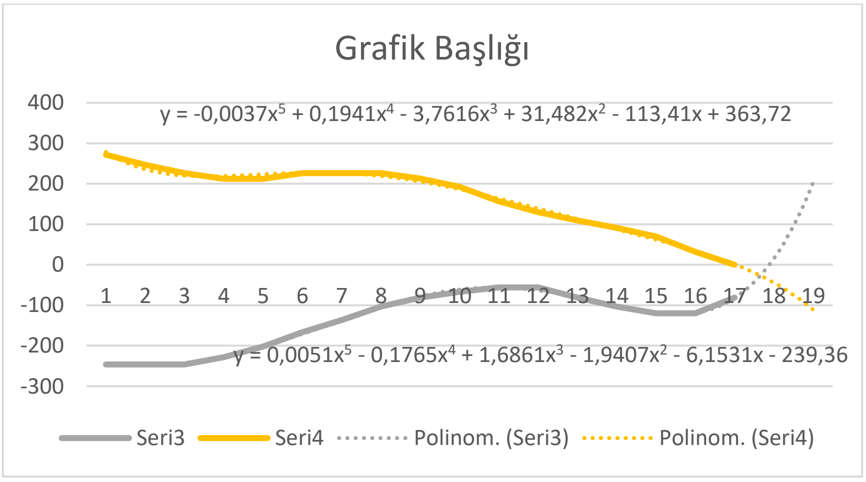


Figure-2.4.2

In the torque calculations found in the dynamic analysis, the time-dependent x and y equations are put in the appropriate places in the torque equation to obtain the following torque values.

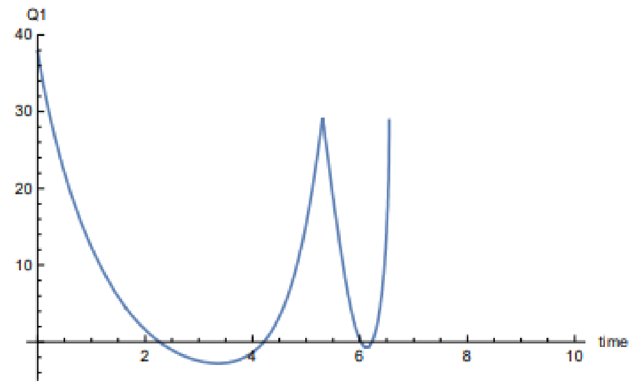


Figure-2.4.3

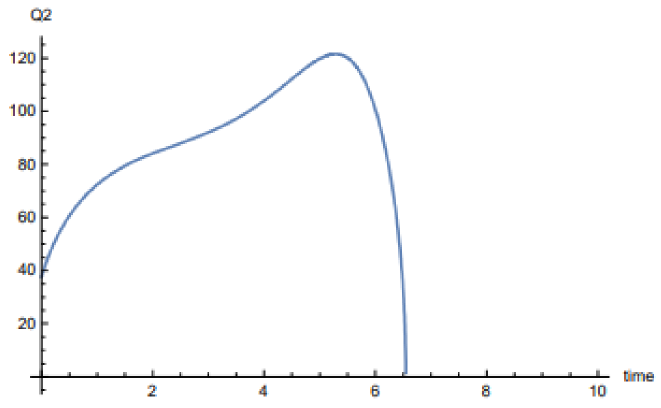


Figure-2.4.4

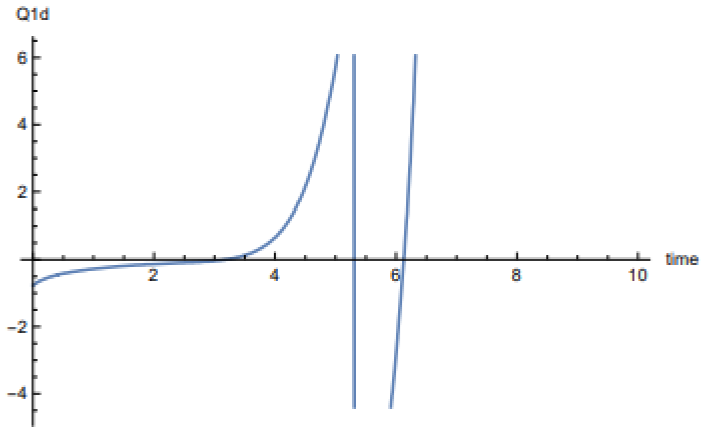


Figure-2.4.5

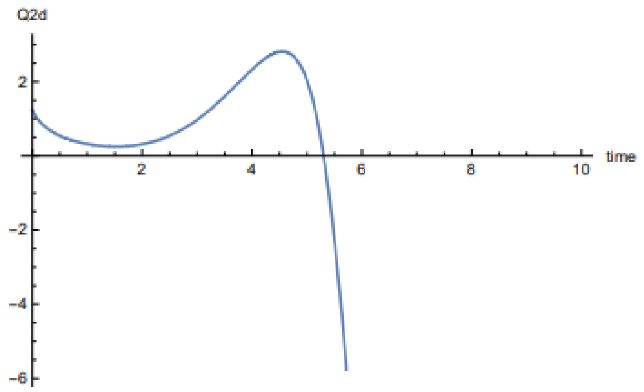


Figure-2.4.6

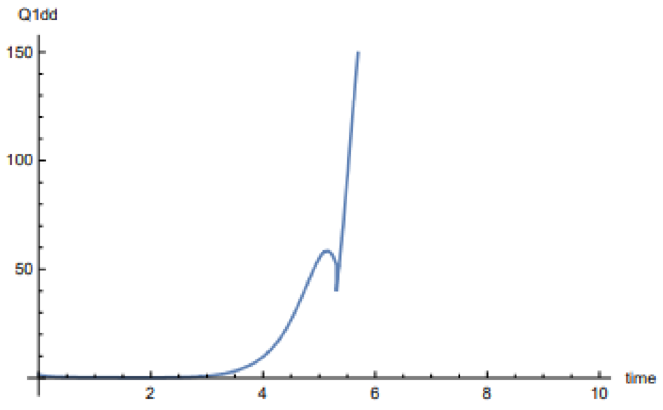


Figure-2.4.7

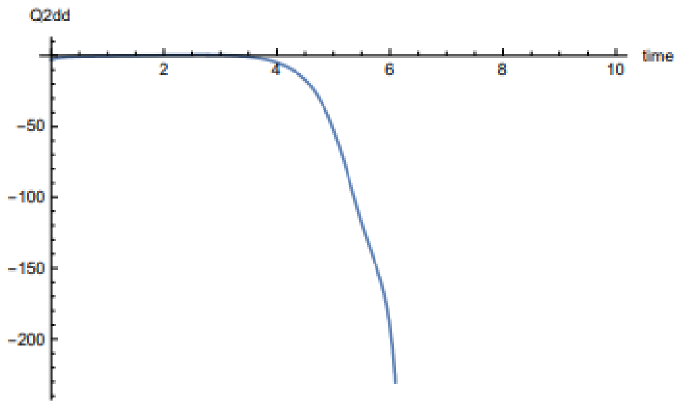


Figure-2.4.8

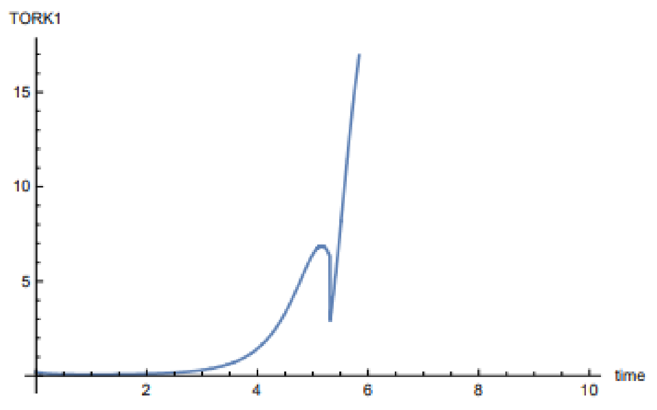


Figure-2.4.9

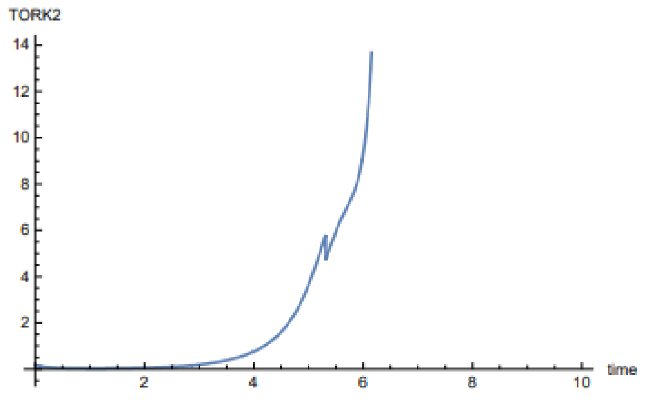


Figure-2.4.10

The accuracy of the code written in the mathematic is compared and verified with the values obtained from the solidworks motion analyze for another path, but for this path , verification could not be performed because this path could not be created in solidworks.From these results, the required torque value is determined as 16kg.cm.

2.4.1.Verification Of The Mathematica Code Which ıs Used For Finding Of The Motor Torque

The motors were fitted to the rotating centers on the assembled series arm in the solidworks program and the rotation speed was set at 5 rpm. The path in Figure-2.4.1 was created and the points on the path were obtained.

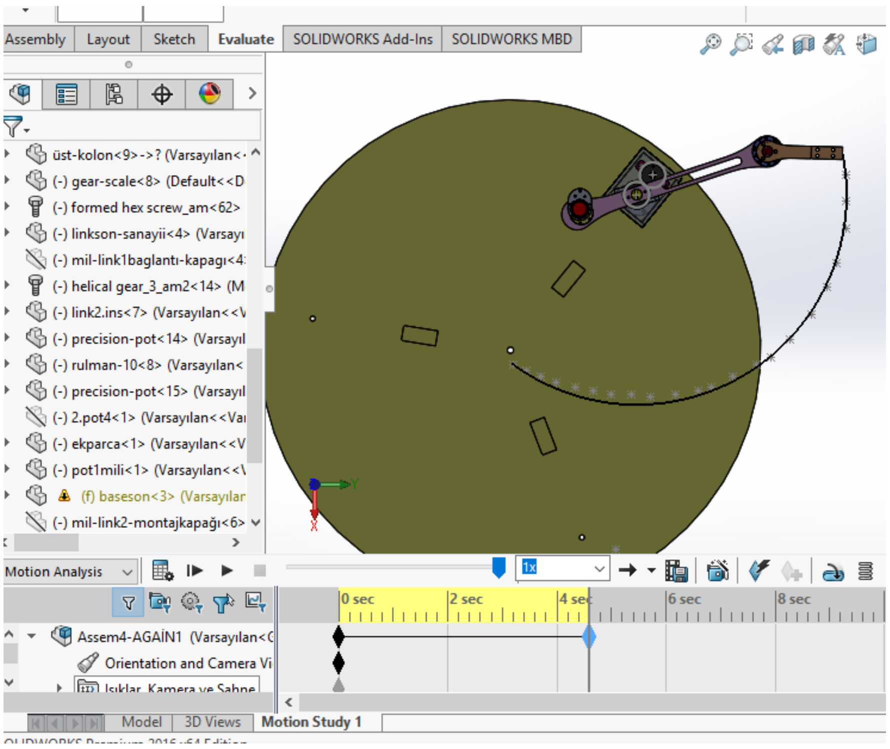
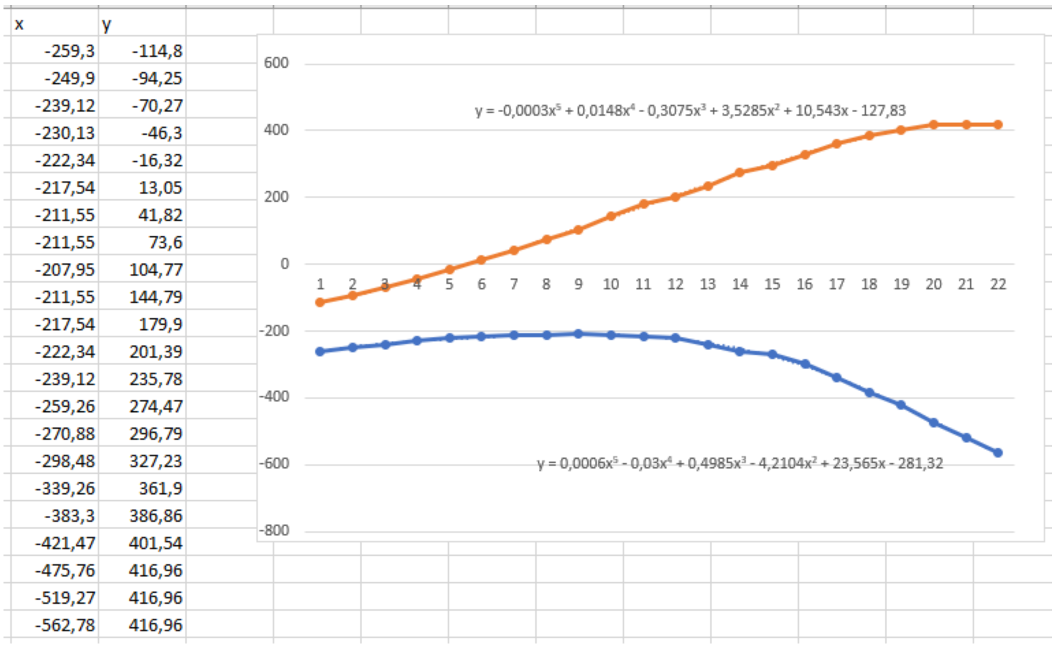

Assembly
Layout
Sketch
Evaluate
SOLIDWORKS Add-Ins
SOLIDWORKS MBD
üst-kolon<9>->? (Varsayılan<
(-) gear-scale<8> (Default<<D
(-) formed hex screw_am<62>
(-) linkson-sanayii<4> (Varsayı
(-) mil-link1bagIantı-kapagı<4
(-) helical gear_3_am2<14> (M
(-) link2.ins<7> (Varsayılan<<V
(-) precision-pot<14> (Varsayıl
(-) rulman-10<8> (Varsayılan<
(-) precision-pot<15> (Varsayıl
(-) 2.pot4<1> (Varsayılan<<Va
(-) ekparca<1> (Varsayılan<<V
(-) pot1mili<1> (Varsayılan<<\
(f) baseson<3> (Varsayılar
(-) mil-link2-montajkapağı<6>
Motion Analysis
1x
0 sec
2 sec
4 sec
6 sec
8 sec
Assem4-AGAİN1 (Varsayılan<
Orientation and Camera Vi
Isıklar, Kamera ve Sahne
Model
3D Views
Motion Study 1

Figure-2.4.1

Determined. X-Y coordinates of each point were taken to the excel. X-Y coordinates of each point were obtained and entered to the Excel. X and Y equations with respect to time of the path was found from Excel (Figure-2.4.1.2).



x	y
-259,3	-114,8
-249,9	-94,25
-239,12	-70,27
-230,13	-46,3
-222,34	-16,32
-217,54	13,05
-211,55	41,82
-211,55	73,6
-207,95	104,77
-211,55	144,79
-217,54	179,9
-222,34	201,39
-239,12	235,78
-259,26	274,47
-270,88	296,79
-298,48	327,23
-339,26	361,9
-383,3	386,86
-421,47	401,54
-475,76	416,96
-519,27	416,96
-562,78	416,96

y = -0,0003x⁵ + 0,0148x⁴ - 0,3075x³ + 3,5285x² + 10,543x - 127,83
y = 0,0006x⁵ - 0,03x⁴ + 0,4985x³ - 4,2104x² + 23,565x - 281,32
600
400
200
0
-200
-400
-600
-800
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

Figure-2.4.1.1

X(t)= 0.0006*t^5-0.03*t^4+0.4985*t^3-4.2104*t^2+23.565*t-281.32

Y(t)= -0.0003*t^5+0.0148*t^4-0.3075*t^3+3.5285*t^2+10.543*t-127.83

From results option of the SolidWorks program, Torque graphics of the each motor were obtained(Figure 2.4.1.3-2.1.4)

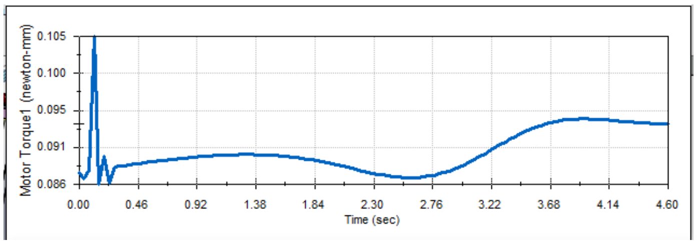


Figure-2.4.1.3

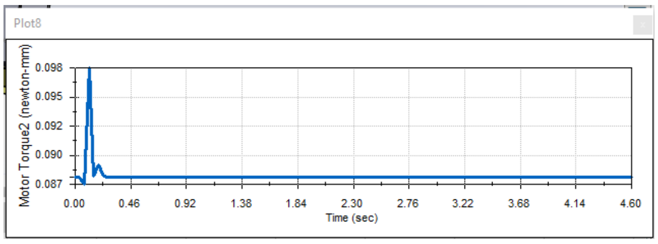


Figure-2.4.1.4

3.DESIGN

Before the beginning of the design, a literature search was conducted to examine the robotic arm samples that have been made so far and ideas for the design to be made were obtained. As a result of investigations, some constraints are determined. It was decided to use a single link for each series of arm to not aggravate the system. In addition, in order to minimize the torque applied to the first motor, the center of mass of the first link was attempted to be brought to the first motor-driven shaft and therefore the second motor was placed in the back of the first link. The movement of the second link will be provided by the belt pulley mechanism which is connected to the motor placed on the first link. Since the position and speed information will be read with the help of a precision pot, it is considered to use the gear mechanism and it is decided to use the helical gear to prevent the loss of sensitivity due to the high amount of gaps in the spur gear mechanism. End effector height is determined as maximum 400 mm and column heights will be designed accordingly. The column width to be placed on the first link is determined as 100 mm×80 mm in order to not to affect the dexterous area.

In the design part, it was first started from the section where the first motor was to be mounted and it was decided to mount it on the inside of the engraved column(Figure-3). The interior of the column is designed to be empty to accommodate electrical components and thus to save space.

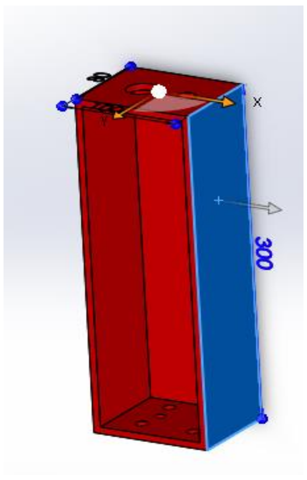


Figure-3.Bottom column

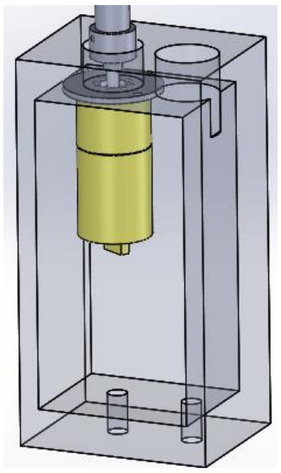
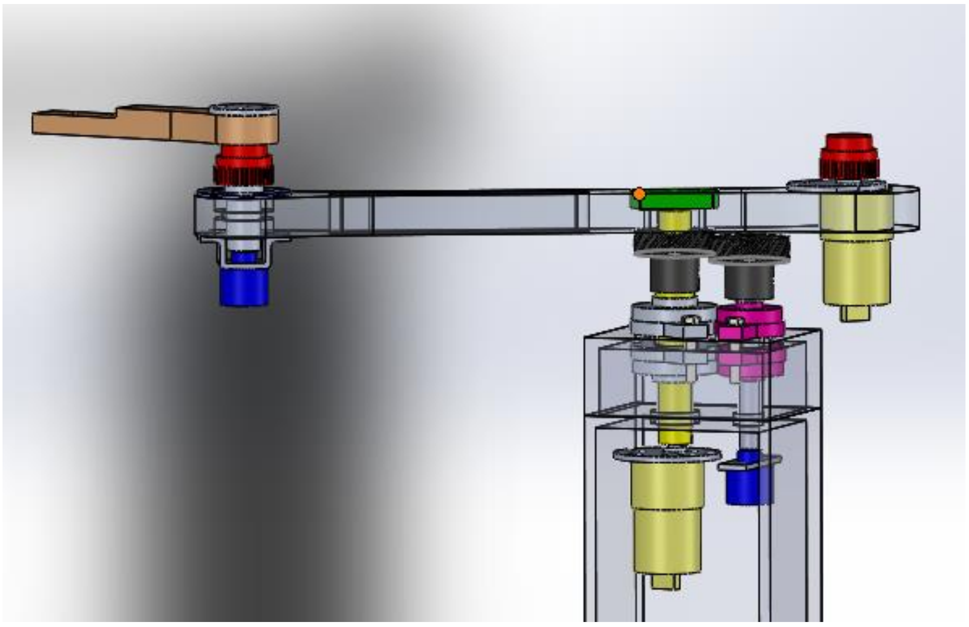


Figure-3.1. column-motor assembly

A ball bearing with an internal diameter of 17 mm was selected for bearing the shaft driven by the first motor, and the shaft diameter was determined as 16.95 mm. Twice the bearing was made to prevent the axial load of the shaft and bending. In order to be easy to assemble, two deep groove ball bearing with housing was preferred. The helical gear dimensions were determined as 1.5 M-25 teeth to fit the column width after mounting of the helical gears.



(Figure-3.2)

In the design of link1 (Figure-3.3), the distance between the turning point of the motor shaft and the second link was determined as 220 mm in the workspace analysis. The link width was determined to be 25 mm and the location of the second motor was removed 100 mm from the turning point so that the second motor would not hit the helical gears while the link was rotating.Since the motor diameter is 37 mm, the part on which the motor is mounted on the link will be pierced with a diameter of 38 mm. As a result of solidworks static simulation for link 1, link thickness is determined as 20 mm.

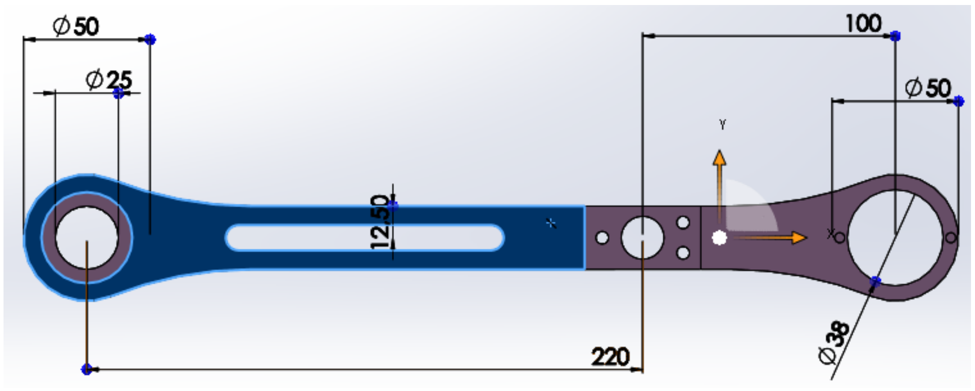


Figure-3.3.Link1

For the connection to the shaft and link, the rectangular groove will be opened on the link and the rectangular shape on the shaft will be fixed with 3 bolts. In order to prevent the link1 from hitting the second helical gear, a piece (Figure-3.5) of 8 mm thickness will be produced and assembled between link1 and gear.

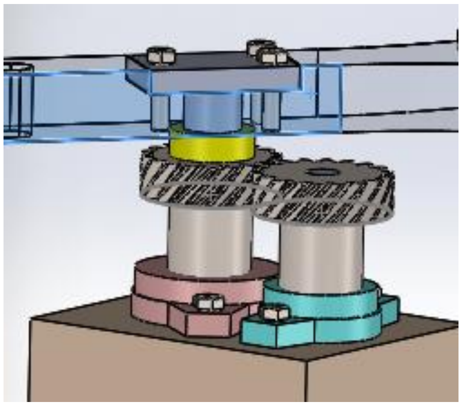


Figure-3.4

It was decided to make two bearings for the shaft between link1 and link2. As shown in Figure 3.5, two deep groove ball bearings with an internal diameter of 17 mm are used. In first link, the ring grooves are opened and the bearings are fixed by internal rings. The thickness of the shaft is 16.90 mm. Bearings will be fixed with the help of setscrew. shaft and link 2 will be assembled together with three bolts as shown Figure-3.6. The precision potentiometer used for reading the speed and position information of the second motor will be installed under the link1 and the shaft of the precision potentiometer will be mounted on the shaft.

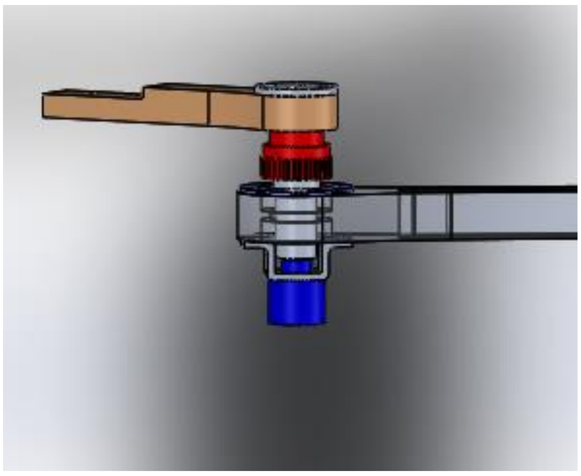


Figure-3.5.Assembling of two links

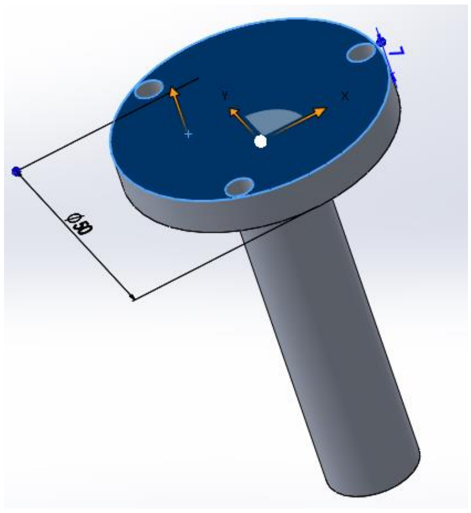


Figure-3.6.connecting mil of link2

The link2 was defined as 220 mm from the dexterous workspace analysis. but considering the installation of the gripper, the instantaneous part to be produced was determined to be 120 mm. After finishing the production of the gripper, the arm and the gripper will be assembled with four bolts.

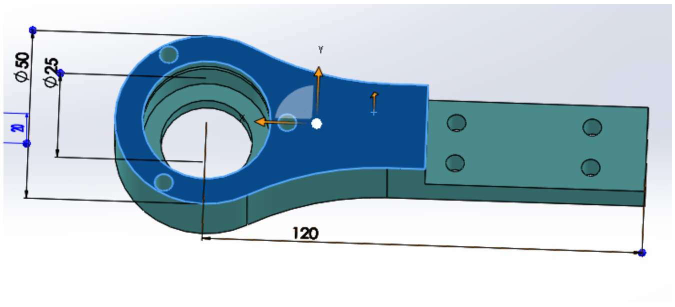


Figure-3.7.Link2

The timing belt and chain is used in order to increase the sensitivity between the movement of the second motor and the movement of the link 2. When the belt and pulley is assembled, belt tensioning will be ensured with the rotating of the motor .

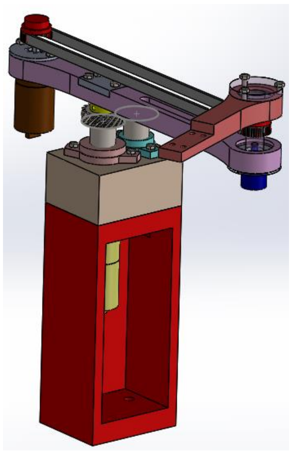


Figure-3.8

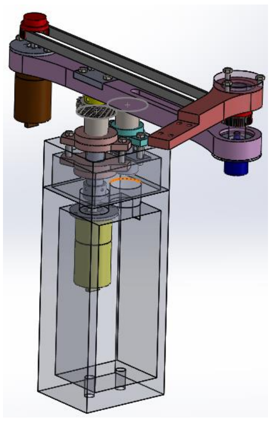


Figure-3.9

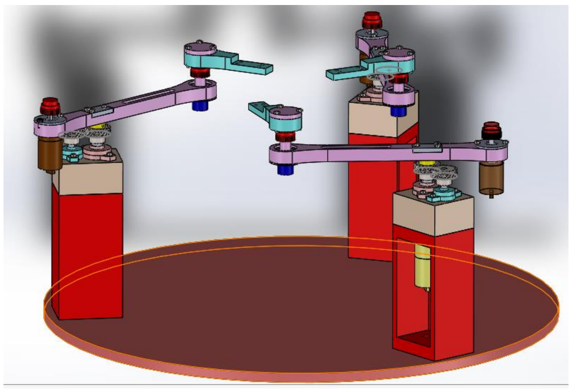


Figure-3.10.Assembling of all serial manipulators

SIMULATION

In this simulation, the static plugin in solidworks simulation is used. The simulation was started by assuming that a force of 70 N was created from link2 and a force of 10 N was created in the position of the actuator. Although the system is overloaded, there is no deflection on the aluminum link.

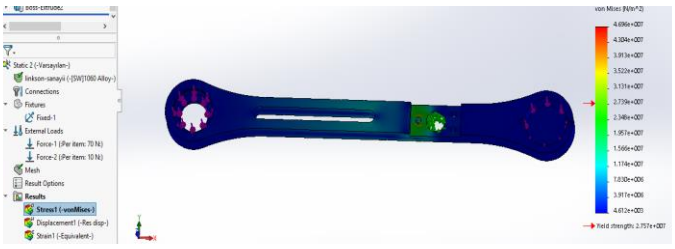


Figure-3.11

Figure-3.12

3.2.HEXAGONAL DESIGN

When the robotic arms are working in series and parallel, a hexagonal platform is designed to perform the desired task and to work in both serial and parallel condition. By using this platform, the laser cutting tip will be on the platform and the robot arms will be able to laser cut by working in series and in parallel. Each robot arm can be connected to the joints located at the six corner of the platform with the grippers.

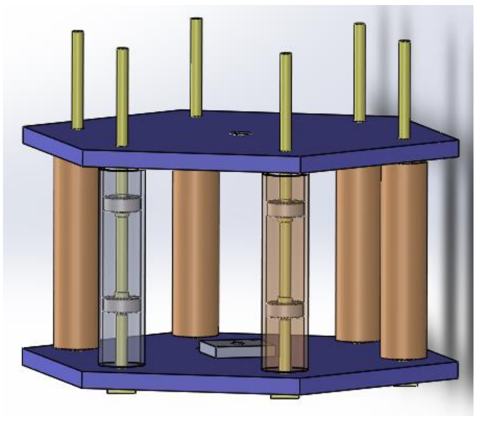


Figure-3.2

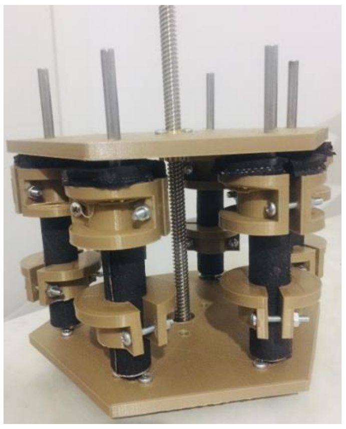


Figure-3.2.1

The entire design of the platform was made using the SolidWorks program. Material list and how to produce the material from which the materials were determined.The platform consists of ;

- 6 pieces of iron shaft
- 1 piece of worm screw and nut
- Servo motor
- Coupling for worm and motor connection
- Rubber to increase friction between grippers and joints
- Filament material
- Bolts and nuts

Figure-3.2.2

The platform is made with filament material for the top and bottom sides of the platform by using 3D printer to make the platform lighter. Iron discs were welded to the bottom of the six mils at the corners, which were fixed to the lower platform with bolts and nuts. In this way, the six shafts are fixed to the lower platform. There are two deep groove ball bearings on each shaft.These bearings were fixed with the help of flanges and segments and the movement in the z-axis was destroyed. In the case of a single operation of the robot, the upper platform moves downward on the z axis and presses the joints to make the joints fixed. This movement is provided by an endless shaft-screw mechanism in the center of the platform. The effect of the brake system is increased by using rubber to increase the pressure on the joints. During the double or triple operation of the robot, the brake system is eliminated by providing upward movement of the upper plate in the z axis to make the joints to be revolute joint.

3.3.GRIPPER DESIGN AND ASSEMBLY

In the Gripper design, the opening and closing mechanism is provided by using the rack and pinion mechanism. A rack and pinion is a type of linear actuator that comprises a circular gear (the pinion) engaging a linear gear (the rack), which operate to translate rotational motion into linear motion. Driving the pinion into rotation causes the rack to be driven linearly. Driving the rack linearly will cause the pinion to be driven into a rotation. Filament material is produced with 3d printer. Servo motors are installed and the production is finished. The second links were reproduced and bolt holes were opened on the links and the gripper was mounted on the second arm of the robot.The each gripper is driven by a servo motor.

The reason for this design is that the movement in the y-axis is eliminated and the gripper movement is only on the x-axis. In this way, the gripper directly reaches the position of the joints in the platform corners.

If the four-bar mechanism was used instead of this design; Since the gripper would also move on the y-axis if it closed, the distance on the y-axis should be added to the control algorithm.

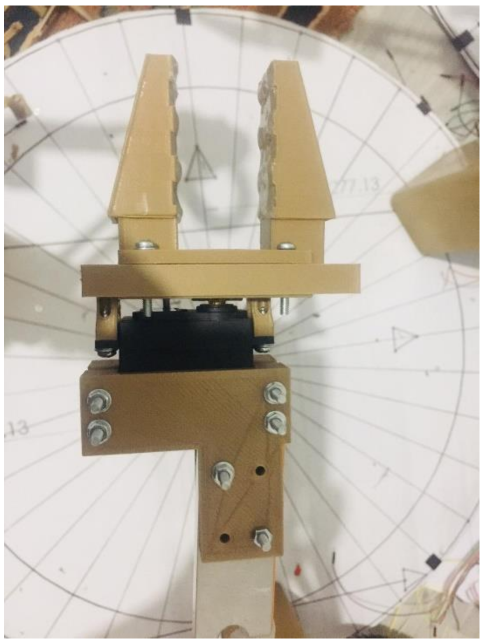


Figure-3.3

3.4.CAMERA HANDLE ARM DESIGN

For image detection to detect the obstacle, the camera needs to see the entire working area. Since the camera view angle is directly proportional to the price, it was determined that the affordable models could see the table in approximately 1 meter distance. In our design, since the distance of our robot in the z-axis is 30 cm, we did not want to lift the robot and put the camera down. Therefore it was decided that the camera should be on the robot by a holding arm. In the Z and X axis, the motion of the camera is not fixed, and the design for adjusting the position of the camera is done in this direction.

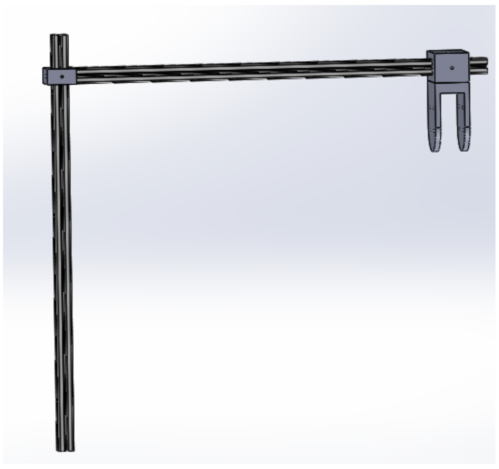
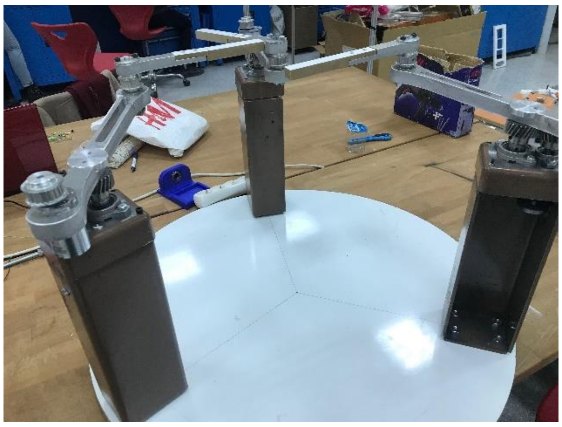


Figure-3.4

3.5. ASSEMBLY

3.5.1.For Graduation Project-1


Figıre-3.5.1

Figıre-3.5.2

3.5.2.For Graduation Project-2

The second link of the robot was rebuilt and assembled with the gripper.

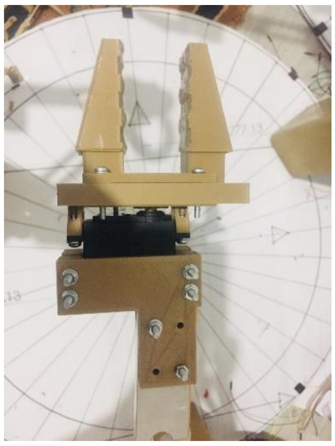


Figure-3.5.2

The platform was assembled and tested.

Figure-3.5.2.1

Grippers are connected to whole second links.As shown in Figure…,all of the grippers are attached with platform to operate in triple configuration.

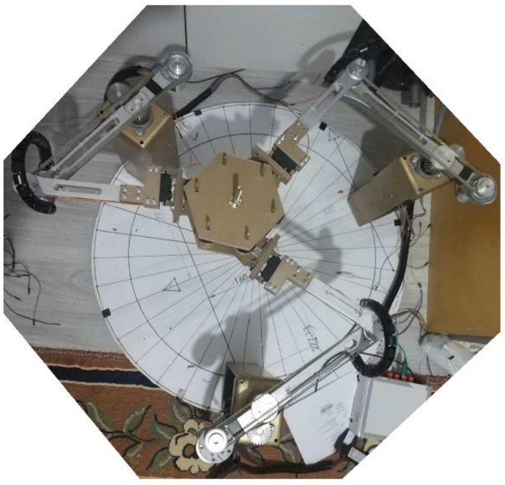


Figure-3.5.2.1

4.ELECTRONICS

4.1.ACTUATOR SELECTION WITH RESPECT TO DYNAMIC ANALYSIS

As a result of the torque calculation, the minimum torque requirement was set to 16kg.cm. When selecting the motor, it was necessary to select a motor to meet the 16 kg.cm torque requirement at 30 rpm. The stop torque of the motor shown in the figure below is 42 kg.cm. Since it is inversely proportional between torque and rpm, it is decided to take this motor because it provides about 16 kg.cm at 30 rpm.

Figure-4.1

Actuator Properties ;

- Operating Voltage: 12V
- Gear Ratio: 50:1
- Speed: 100Rpm
- Strain current: 680mA
- Motor diameter: 34mm
- Reducer diameter: 37mm
- Shaft: 3mm- D shaft
- Shaft length: 14mm
  Mass: 210 g

4.2 POTANTIOMETER SELECTION AND LINEARIZATION

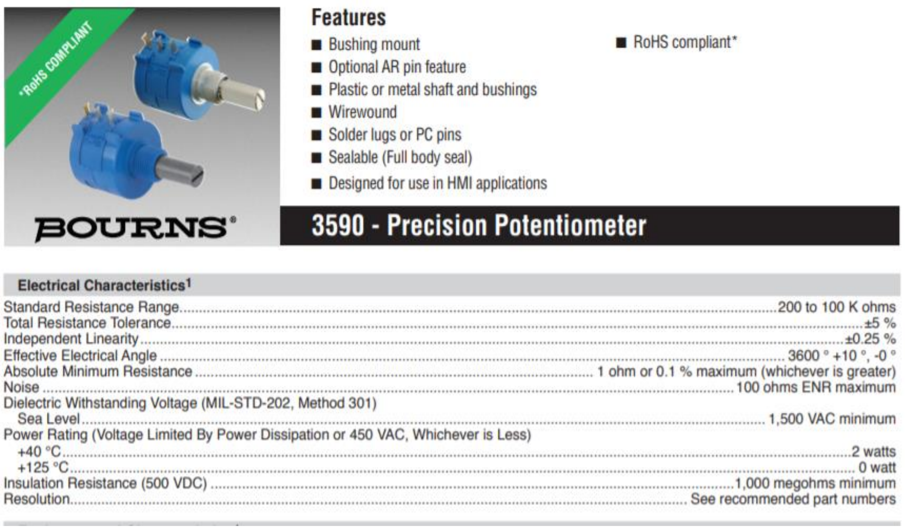
In order to get feedback control of the position we decided to use potentiometer as a feedback device. We select 10K 3590 preccisson potentiometer because of these features listed below;

-Longevity of operational life

-Virtually %5 resolution

-Improved linearity

Figure 4.2.**10K** 3590 Potentiometer



*RoHS COMPLIANT

BOURNS®

Features

- Bushing mount
- Optional AR pin feature
- Plastic or metal shaft and bushings
- Wirewound
- Solder lugs or PC pins
- Sealable (Full body seal)
- Designed for use in HMI applications
- RoHS compliant*

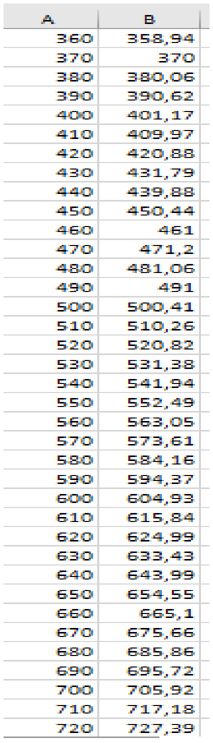
3590 - Precision Potentiometer

Electrical Characteristics[1]

Standard Resistance Range	200 to 100 K ohms
Total Resistance Tolerance	±5 %
Independent Linearity	±0.25 %
Effective Electrical Angle	3600 ° +10 °, -0 °
Absolute Minimum Resistance	1 ohm or 0.1 % maximum (whichever is greater)
Noise	100 ohms ENR maximum
Dielectric Withstanding Voltage (MIL-STD-202, Method 301)	
Sea Level	1,500 VAC minimum
Power Rating (Voltage Limited By Power Dissipation or 450 VAC, Whichever is Less)	
+40 °C	2 watts
+125 °C	0 watt
Insulation Resistance (500 VDC)	1,000 megohms minimum
Resolution	See recommended part numbers

Figure-4.2.1

“A” column represents the desired values and “B” column represents the measured angles which comes from the potatiometer. Starting from 360 to 720 degrees, all values are tested in Excel and the best result is between 360-540 angles, because it is the closest one to each other.Finaly the equation of the characteristics of the potantiometer is obtained as shown below.



A	B
360	358,94
370	370
380	380,06
390	390,62
400	401,17
410	409,97
420	420,88
430	431,79
440	439,88
450	450,44
460	461
470	471,2
480	481,06
490	491
500	500,41
510	510,26
520	520,82
530	531,38
540	541,94
550	552,49
560	563,05
570	573,61
580	584,16
590	594,37
600	604,93
610	615,84
620	624,99
630	633,43
640	643,99
650	654,55
660	665,1
670	675,66
680	685,86
690	695,72
700	705,92
710	717,18
720	727,39

Figure 4.2.2

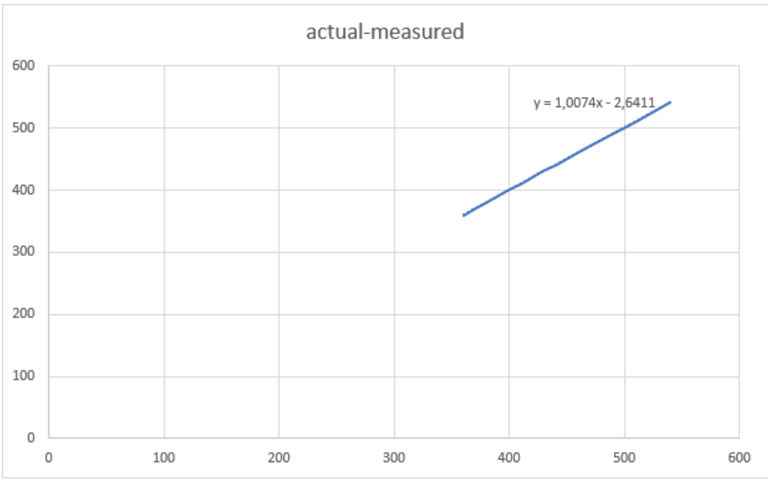


Figure 4.2.3

4.3.DESIGN AND MANUFACTURE OF MOTOR DRIVER CARDS

4.3.1.For Graduation Project-1

A servo motor is a rotary actuator or motor that allows for a precise control in terms of angular position, acceleration and velocity, capabilities that a regular motor does not have. It makes use of a regular motor and pairs it with a sensor for position feedback.[1] As a constraint, in the Project this type of motor cannot be used to drive the system, instead of that a servo-mechanism is used. It can be created by utilizing a DC motor and a precision potentiometer (or without potentiometer, the DC motor with encoder), and a proper driver circuit. Precision potentiometer gives the position feedback, and it is done by coupling the shaft of precision potentiometer and the shaft of DC motor. In Figure 1,2 and 3 it shows an example of coupling. This kind of systems allow motor shafts rotates with precision potentiometer, so that by using a microcontroller, the position of potentiometer can be known so can the position of the motor.

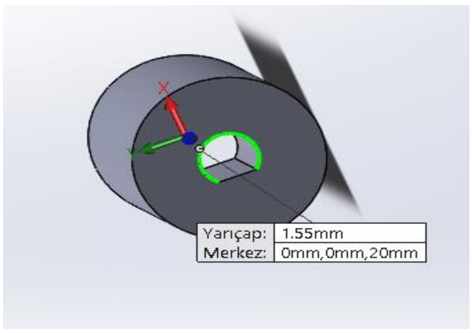


Figure-4.3.1

As another constraint, a ready driver cannot be used. Ready drivers have position, velocity circuits and also circuits which let the motor driven. There are lots of way to drive a motor, such as unidirectionally, bidirectionally. The most common way to drive a motor bidirectionally, H bridge is used.

Briefly, H bridge’s working principle to forward the input current and define the motor polarization. It is useful to drive the motor in both directions.

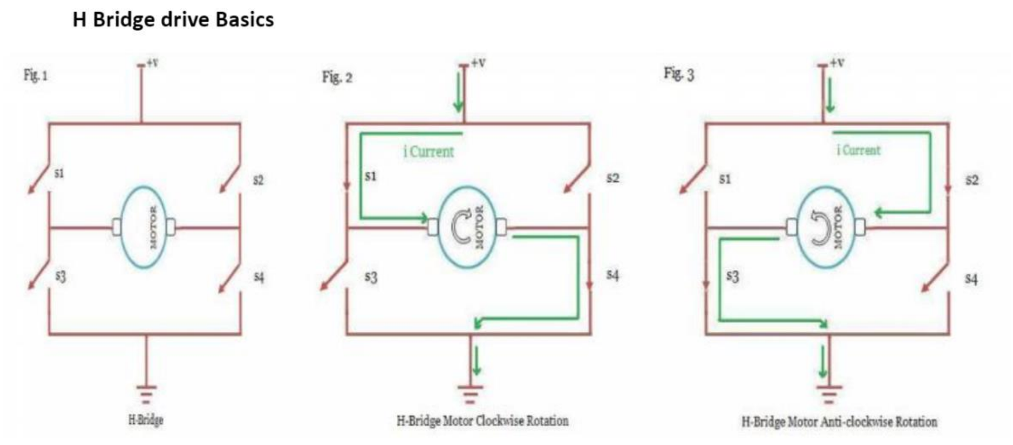


Figure 4.3.1.2[1f]

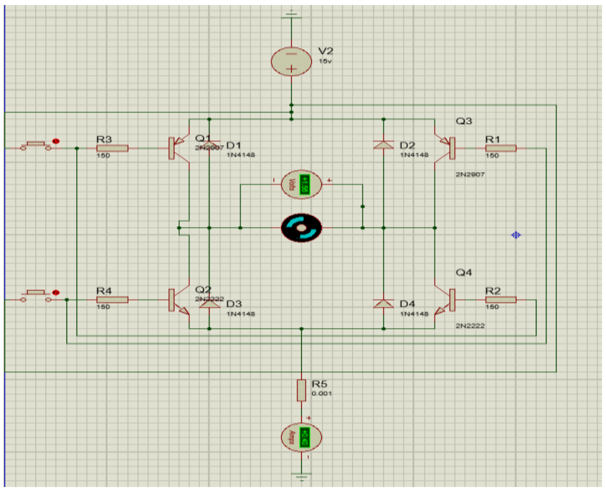
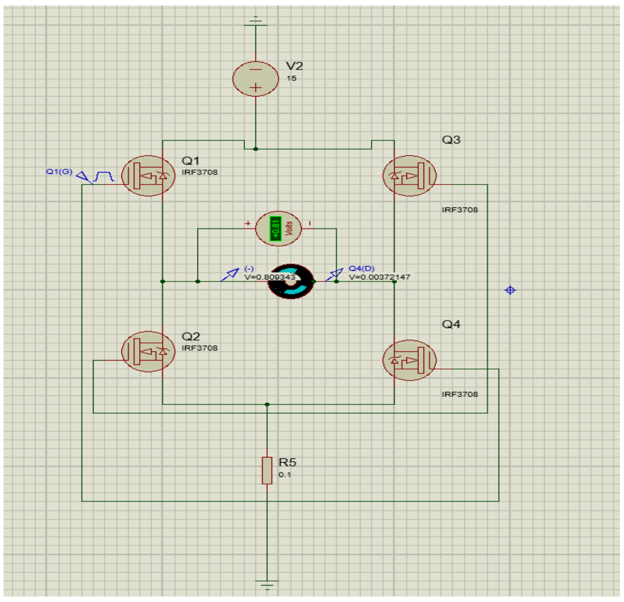


Figure 4.3.1.3

First try was to build a H-bridge full of transistors and diodes as Figure 5. In Proteus it didn't work well. It is reasonable, because the voltage drops the motor is too low, if 15 V is given, the motor will turn only with 1.5V.



Second try, it was with full of NPN MOS-FETs. It was the same problem with first try. Also, voltage drops were observed in this circuit by a multimeter, but since it was a fail, there is no real circuit photo of it. The reason of failure is that the H-bridges by same types of polarized MOS-FETs, needs more voltage to drive the motor properly. In this case, each MOS-FET work 6V, and the rest goes to the motor. A research sentence for that: (…) they can be made with either p-type or n-type semiconductors, complementary pairs of MOS transistors can be used to make switching circuits with **very low power consumption(…)**.The main advantage is that it (MOS-FET) requires almost no input current to control the load current, when compared with bipolar transistors.[2]

So, the most proper circuit consists of NPN, PNP MOS-FETs and transistors. This circuit works well in real life too.

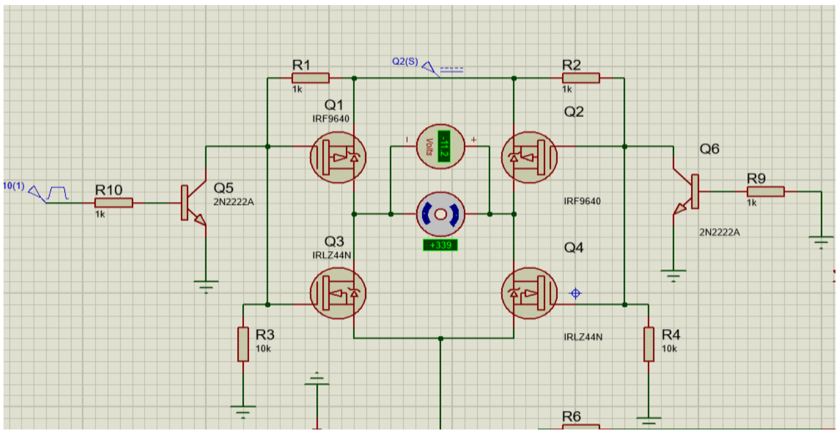


Figure 4.3.1.4

Equipment list for H-bridge:

-4 x (1k ohm)

-2 x (10k ohm)

-2 x (IRF9640)

-2 x (IRFZ44N)

-2 x 2N2222A

-1 x (1 ohm)

The algorithm of this circuit:

- To make easier the circuit, 2 NPNs are placed upside, and 2 PNPs are placed downward.
- It is easier because (for one NPN and PNP connection), PNP's drain leg is connected to NPN's drain leg.
- PNP's gate leg is connected to NPN's gate leg.
- To active H bridge, PNPs' source legs are connected to each other, and fed by 12V.
- NPNs' source legs are connected to each other, and have a resistor and end with ground.
- Sides of motor are connected to each drain connections.
- 1k ohm in front of transistor is placed to prevent the transistor burn. Because in reality without a resistor, we burned at least 5 transistors.
- Each NPN's gate legs have 10k ohm pull-down resistors.
- Each PNPs have 1k ohm resistor between its gate leg and source leg.
- Transistors' collector legs are connected to gate legs of PNPs.
- One of the gate leg of the transistor is fed by 5V pwm, the other one is to ground.
- Both transistors' bases are connected to ground.

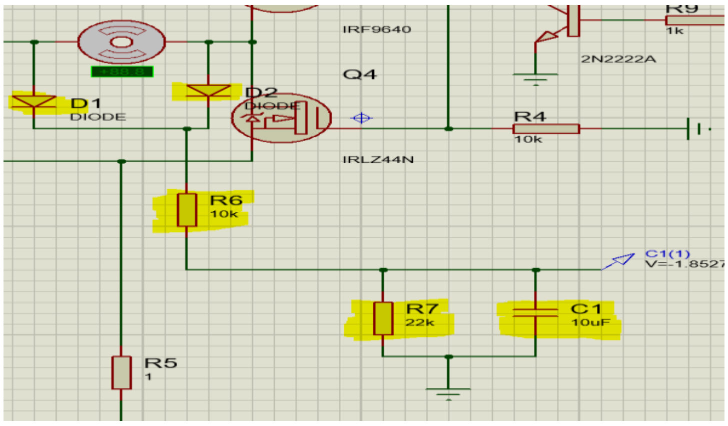


Figure 4.3.1.5

Velocity feedback circuit:

Equipment list:

-2x diodes

-1x 10k ohm

-1x 22k ohm

-1x 10uF

The algorithm of this circuit;

This H-bridge can be driven at both sides, so that to get a velocity feedback for both sides diodes should be used.

- 10k and 22k ohms are used as voltage divider. When 12V comes through 10k,22k, this voltage divider regulates the voltage between 0-5V.
- Because, the feedback is read by a microcontroller, and a microcontroller cannot handle more than 5V.
- Also 22k and 10uF are used as low-pass filter, it cleans the output from noise.

Position control algorithm:

Ziegler Nichols second method of tuning is used to tune the system response. There are three parameters to be tuned: K, Ti, Td.

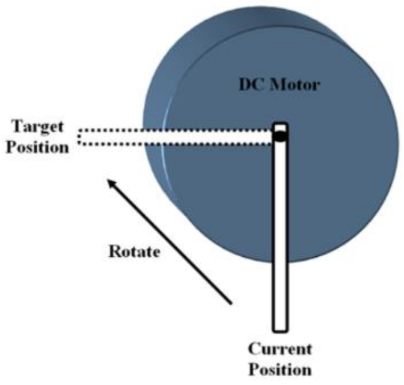
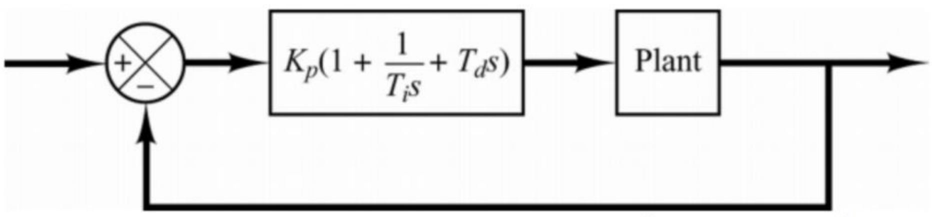


Figure 4.3.1.6



$$\frac{U(s)}{E(s)} = G_{PID}(s) = K_P + K_I \frac{1}{s} + K_D s = K_P \left(1 + \frac{1}{T_I s} + T_D s \right)$$

Figure 4.3.1.7[2f]

<u>Proper Kp (Algorithm):</u>

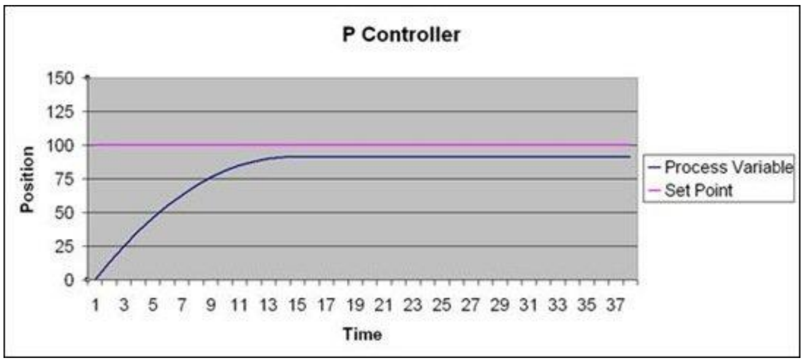
- Set up the H bridge, and coupling between shafts. (Starting from second step is about coding)
- Have a while(condition==1) loop just before while(TRUE) and after void main().
- Define an array to collect the values. It will let the Kp value is stored inside an array.
- To finish collecting, make a definition. (For example 100*, the last char will make the condition zero, because it shows that 100 will be used.)
- Here, array is in string form. So, 100 should be converted to int value. Because this value will be multiplied with error, and if it is not converted to int, its ansii values will be accounted.
- In this point, Kp is ready, and condition is already zero because of * character.
- With this Kp value, the code gets in a while(TRUE) loop, which means if one wants to change Kp value, microcontroller should be reset, so that the code will start again, condition will become 1 and so on. With Arduino, it is easy to do it. Also with PIC, it only requires a basic button-resistor circuit from its first MCLR leg, or directly shut down the 5V going through MCLR leg.
- There is a desired value that the motor's shaft should arrive, and in the other hand there is actual position of the shaft. The difference between them gives us the "error". As an example, desired value is calculated as 200, but in that instant the shaft's position value is 100. So, error becomes 200-100 = 100.

- Then, Kp value is multiplied with this error. (Kp*100)
- This value defines the pwm, and it is between 0-1023.
- As physical system, an oscillation should be observed to get true Kp value which is called Kc(K critical). If no oscillation, it is not the value that is looked forward to. The microcontroller should be reset and have another Kp value to find.

<u>Ku values for the project:</u>

Theoretically, after the proper Kc is found, Pc(period critical: oscillation period in Kc value) should be measured by using a oscillator or counting the oscillation within a time. Then a table exists that let us know Kp, Ti and Td value.

Type of Controller	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$



P controller is used to get a steady-state response with some error. As an example, if ref. point is 100, and actual value is 50, this value will increase up to somewhere around 90 or 110, but never it is fixed at 100. As can be seen from the graph of P controller, the actual position of the motor shaft, when settles down will not reach the target position. This is because when the current position is near to the target position, the error becomes very small and the computed PWM duty cycle is too small for the motor to overcome the friction and gravity[3].

If it is needed to get rid of this steady-state error, then the integral controller should be added to the system. Here it comes Ki value, it is the constant value of integral controller. This time the procedure happens like below:

- Just like Kp value, there is an error value. This error is defined as difference between reference point and actual point.
- User should define Kp and Ki, separately.
- So here, there will be two while(condition=1) loop. One is for Kp and the other is for Ki.
- Then, like its name suggest, this error values should be accumulated into a variable, like “integral”. integral = integral+error.
- Lastly, the control parameter should become control = kp*error + ki*integral.

In this part of the paper, final version of the motor driver circuit and PCB is mentioned. The only difference between older version of the circuit is that there is no resistor connected to ground after the source legs of the NPN MOS-FETs. Sources are directly connected to ground.

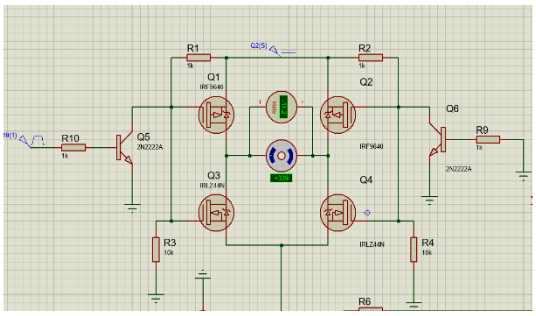


Figure 4.3.8

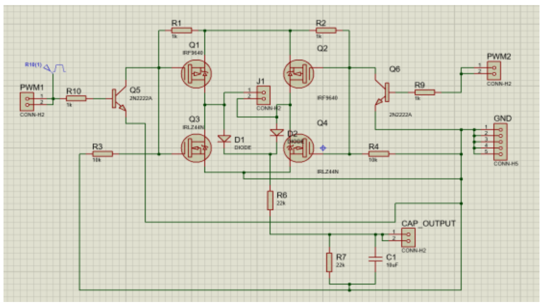


Figure 4.3.9

Figure 4.8 is the earlier version of the driver. This circuit is designed in Proteus, as a step of PCB circuit. To convert the circuit into ARES, all grounds are held in one terminal block, output of the capacitor, pwm, 12V and motor components are shown as terminal blocks to design in ARES. ISIS is a tool that can be used to simulate circuits in real time. Another usage way of ISIS is to convert the scheme to ARES to have PCB circuit ways. The version of 8.0 in Proteus, there is no need to do extra thing to convert, because all files have the same extent which is .pdsprj. In earlier versions, in order to make this converting, the file should have been saved in another extent, and do more steps such as activating the Netflix. In this version, Netflix is already in usage in real time. The second step for 8.0 version is to click little red box writing ARES (Figure 4.3.10).

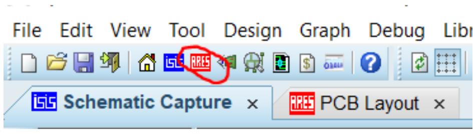


Figure 4.3.10

After clicking, the page is opening. First thing in this page to do is to define board edge, otherwise Proteus sends an error message that PCB layout needs to know the dimensions of the board to place components. To define board edge, the left side of the screen there is a square block (Figure 4.3.11). If it is needed to use metric system, “M” is pressed, and inches are converted to mm’s automatically which can be seen at the right bottom side of the screen (Figure 4.3.12, Figure 4.3.13)

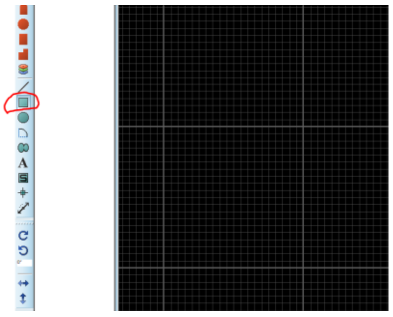


Figure 4.3.11

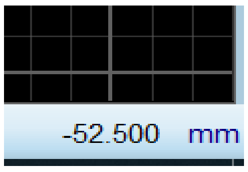


Figure 4.3.12

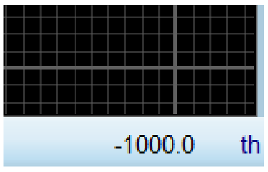


Figure 4.3.13

Then, at the right bottom of the screen, there is a combobox that can be pressed and selected different things, so there, "board edge" should be pressed (Figure 4.14). When the selection is done, the size od board edge is decided by clicking "Techonology->Set Board Properties" (Figure 4.3.15).

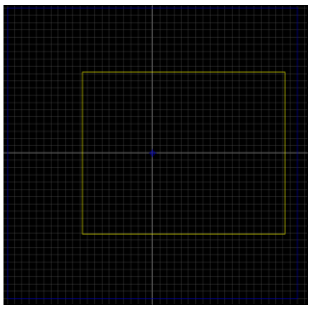


Figure 4.3.14

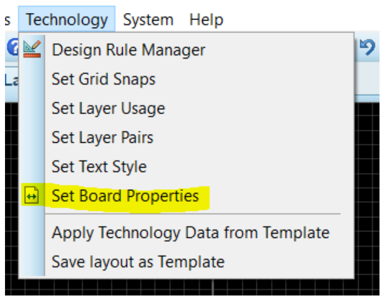


Figure 4.3.15

After that, there is two ways to go. One of them is to let Proteus do the job and place all components, and second placing all components by user. In this study, components are placed by user for the benefit of designing the ways (Figure 4.16).

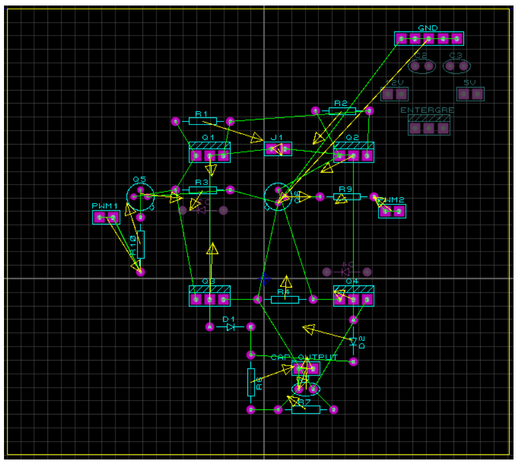


Figure 4.3.16

Forward step is to draw the path. Here there are again two ways to do, one is to let Proteus do the job, and second is to draw the path. In this study, Proteus did the job, but since Proteus draws the path by considering both bottom and top copper, in real life it is difficult to realize.

So, after Proteus finishes its job, the only thing to do is to cancel top coppers and find a way such that all ways are in the bottom (Figure 4.3.17, Figure 4.3.18).

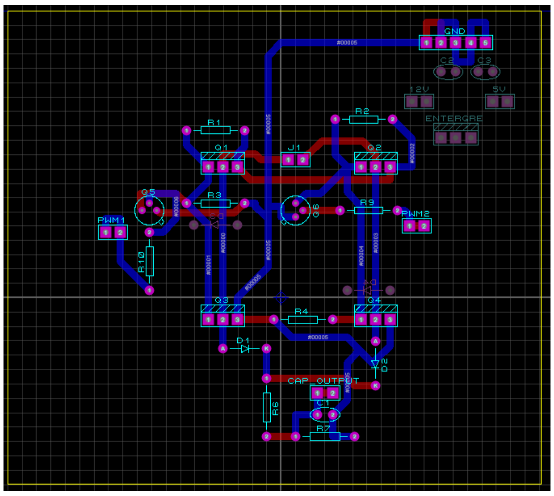


Figure 4.3.17

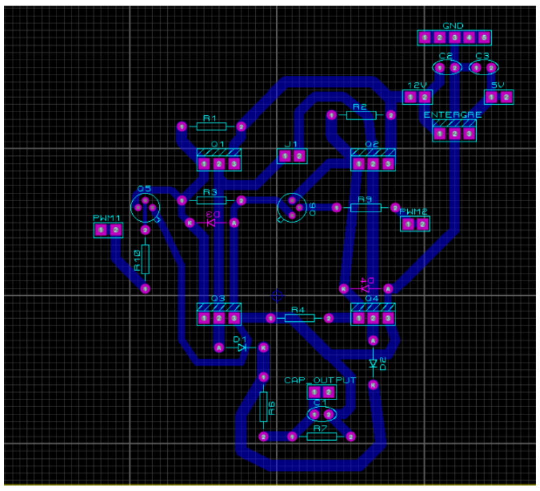


Figure 4.3.18

In this project, ways are good to be thick rather than thin. As a last step in Proteus, this circuit is converted to PDF format, and saved (Figure 4.1.1). First step of the realizing the circuit is to print the circuit into a glossy paper. Second step is to iron the paper on to copper plate during 10 minutes (the side of copper). Third step is to let the plate cold down, and get rid of paper slowly. If in the ways, there are mistakes, by using an acetate paper they are fixed. Forth step is to have a acid solution which contains 1.5 spoon perhydrol, and half of a cup HCI (Figure 4.319,Figure 4.3.20, Figure 4.3.21). Then, let the plate's ways are getting seen.With same solution up to 4 circuits can be made. After 15 minutes, plate is taken out, and cleaned (Figure 4.3.19 -23).

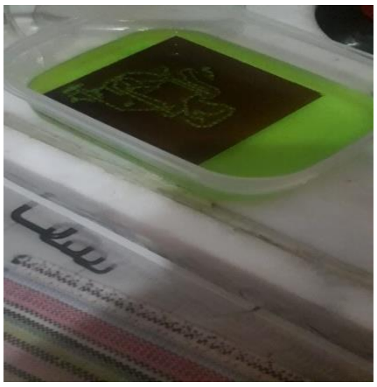


Figure 4.3.19

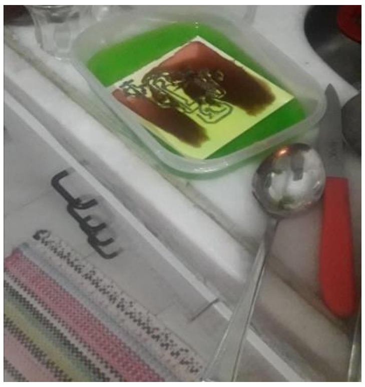
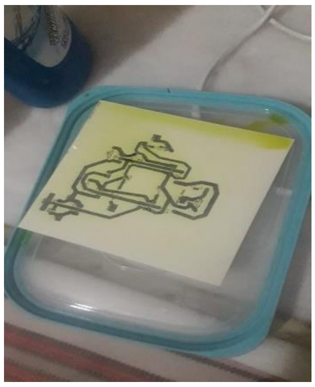


Figure 4.3.20


4.3.21

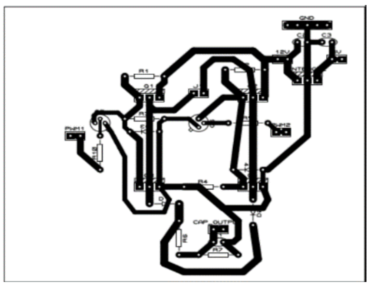


Figure 4.3.22

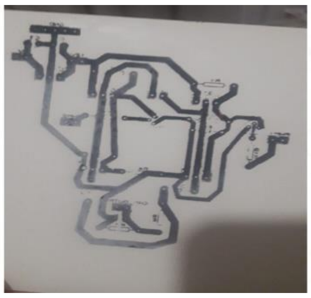


Figure 4.3.23

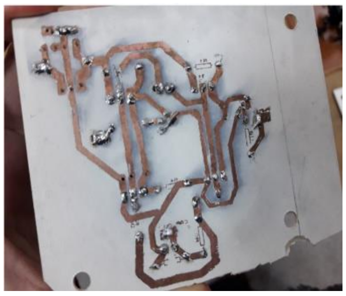


Figure 4.3.24

The logic of PCB to have components' connection via coppers. The only thing to do after that point is to drill the points of components' places, and place the component by using solder tool (Figure 4.3.23).

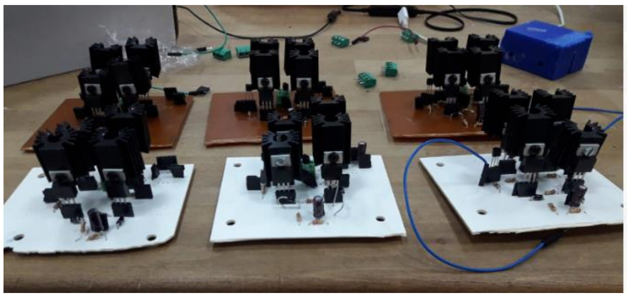


Figure 4.3.24

MOSFETs and transistors are soldered with headers for the benefit of changing the part easily. Motor outputs are soldered with terminal blocks.

4.3.3 For Graduation II

In this paper, two circuits are mentioned. First one is PIC circuit, and the second one is Atmega circuit. Both circuits work in the same manner, the principle is the same. To clarify, they are mentioned seperately.

-PIC circuit

This circuit is consisted of one pic, (malzeme listesi). It can drive two motors at the same time even though PIC 16F877A has only two pwm. This circuit is smaller, because instead of MOS-FETs, the ready H-bridge is used. Another thing that, thanks to regulator, 12V can be converted to 5V.

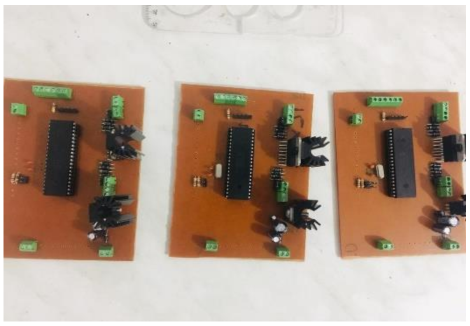


Figure 4.3.3.1

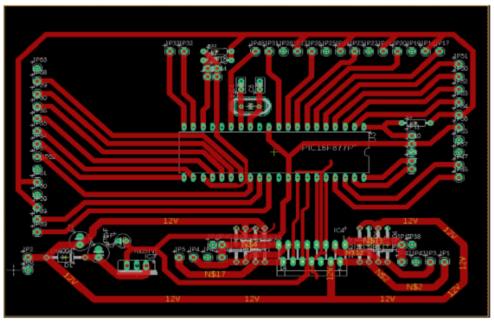


Figure 4.3.3.2

The below's diagram shows the connected devices. PIC is connected to computer, Arduino without Atmega, and Rasperry Pi3. PIC is programmed by using PicKit this is why it is connected to PC. Since PIC has no serial print like Arduino, RS 232 should be used. PIC's RX and TX are connected to respectively Arduino's RX TX. The crucial thing is that this Arduino has to be with Atmega, so that chip can be removed easily and used as RS 232. This lets the user see values by using Arduino's serial screen. PIC is also connected to Rasperry Pi3, to have communication. The main program is inside Rasperry Pi3, but both PIC and Rasperry Pi3 can send and receive data. This communication is done with pins RX and TX. It is mentioned in Fig.4.3.3.3 Since Rasperry Pi3 is a computer itself, it should have its own monitor, keyboard and mouse to be programmed. That's why it is not connected to computer but monitor.

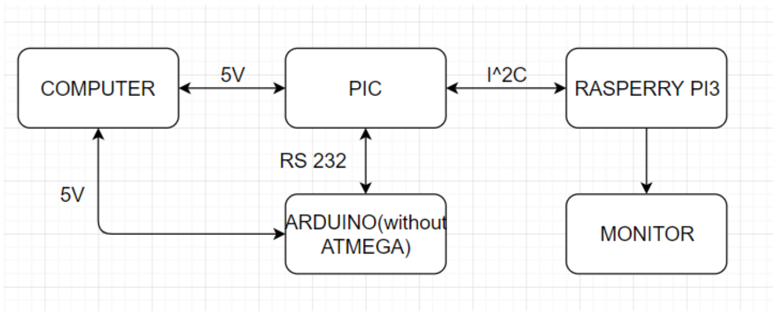


Figure 4.3.3.3

First of all, the communication between PIC and Raspberry Pi3 is made by using the protocol of I^2C. It was successful within both sides, they could communicate each other either way. If it is sent an integer number from PIC to Raspberry Pi3, and the either, that's successful. From PIC, potentiometer values are converted into angles and then sent to Rasperry PI3 which includes main program for forward and inverse kinematics. After Rasperry PI3 receives the data, it calculates the actual coordinates of this manipulator, then with respect to task, it calculates for other angle that can go to desired coordinates. These angles are sent back to PIC, and PIC drives the motors respectively. The main function of PIC is driving motor which means finding the proper pwm, and also measuring the potentiometer values. All of the other things are done by using Rasperry PI3.

The problem is while sending a set of data by using potentiometer, the program sent random values with correct values. As example, if the set of data is 10,20,30,40, the actual coming set of data is 10, 11, 12, 13, 15, 20, 21, 25, 28, 30.   The same problem occurs either directions. These values can be only seen by using Arduino's RS 232. Since this problem occurred, Atmega circuit is made.

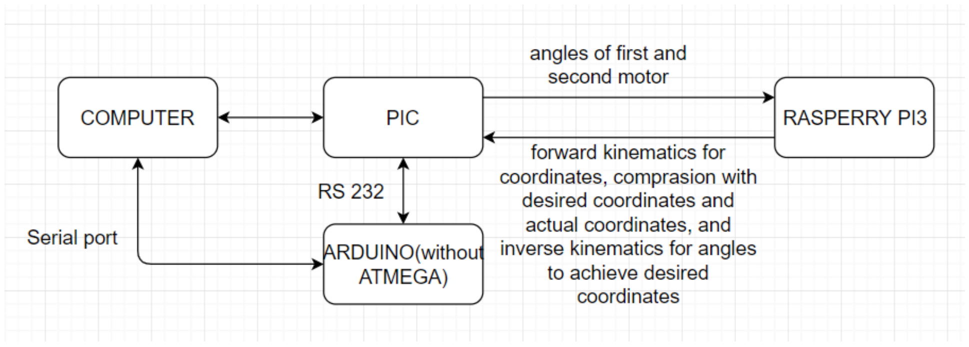

angles of first and second motor
COMPUTER
PIC
RASPERRY PI3
forward kinematics for coordinates, comprasion with desired coordinates and actual coordinates, and inverse kinematics for angles to achieve desired coordinates
RS 232
Serial port
ARDUINO(without ATMEGA)

Figure 4.3.3.4

-ATMEGA circuit

This circuit is consisted of one Atmega, diodes, resistors, regulator, capacitors, cyristal, clemens, cable, and pertinaks. The difference between PIC and Atmega, number of pwm. Since, Atmega has more pwm pins, it makes the things simplier. Also, it has its own serial window, there is no other thing to be connected like RS 232. This can also drive two motors at the same time.

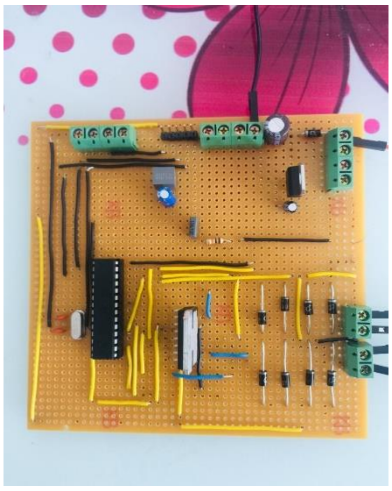


Figure 4.3.3.5

The below's diagram shows the connection. It is simpler than PIC diagram because there are less devices connected. There is no PICKIT, and RS 232. Because, Atmega can be programmed directly and it has its own RS 232. Again between Atmega and Rasperry PI3, there is I^2C communication and they can send and receive data either way. Rasperry PI3 should be used with other monitor to be programmed.

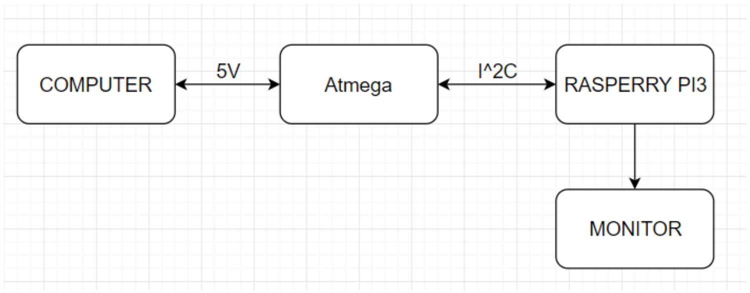


Figure 4.3.3.6

The communication between Atmega and Raspberry Pi3 is made by using the protocol of I^2C. It is successful within both sides; they could communicate each other either way. From Atmega, potentiometer values are converted into angles and then sent to Rasperry PI3 which includes main program for forward and inverse kinematics. But this time the problem is I^2C cannot send or receive float data. So, all values are multiplied with 100 inside Atmega, and after that the new values are sent to Rasperry PI3. After Rasperry PI3 receives the data, it is divided by 100 and then it calculates the actual coordinates of this manipulator, then with respect to task, it calculates for other angle that can go to desired coordinates. These angles are sent back to Atmega by multiplying with 100 again, and Atmega drives the motors respectively by dividing 100. The main function of Atmega is driving motor which means finding the proper pwm, and also measuring the potentiometer values. All of the other things are done by using Rasperry PI3.

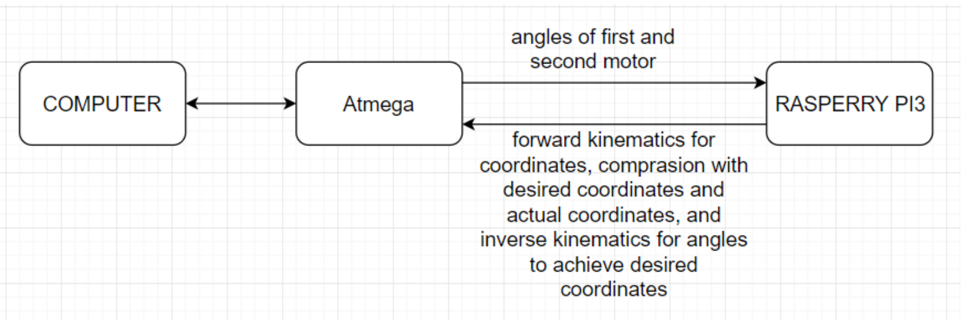

angles of first and second motor
COMPUTER
Atmega
RASPERRY PI3
forward kinematics for coordinates, comprasion with desired coordinates and actual coordinates, and inverse kinematics for angles to achieve desired coordinates

Figure 4.3.3.7

5.CONTROL

5.1.PATH GENERATION

Consider the problem of moving the tool from its initial position to a goal position in a certain amount of time. Inverse kinematics allow the set of joint angles that correspond to the goal position and orientation to be calculated. The initial position of the manipulator is also known in the form of a set of joint angles. What is required is a function for each joint whose value at t_0 is the initial position of the joint and whose value at t_f is the desired goal position of that joint. As shown in Figure-5.1, there are many smooth functions, $\Theta(t)$, that might be used to interpolate the joint value. In making a single smooth motion, at least four constraints on $\Theta(t)$ are evident. Two constraints on the function's value come from the selection of initial and final values:

$$\Theta(0)=\Theta_0$$

$$\Theta(t_f) = \Theta_f \qquad (5.1)$$

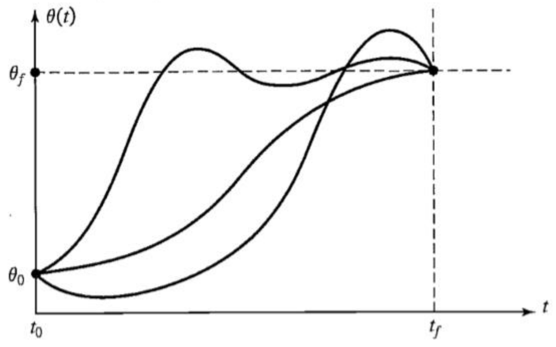


Figure-5.1

An additional two constraints are that the function be continuous in velocity, which in this case means that the initial and final velocity are zero:

$$\Theta(0) = 0$$

$$\Theta(t_f) = 0 \qquad (5.2)$$

These four constraints can be satisfied by a polynomial of at least third degree.These constraints uniquely specify a particular cubic. A cubic has the form

$$\Theta(t)=a_0 + a_1t + a_2t^2 + a_3t^3 \qquad (5.3)$$

so the joint velocity and acceleration along this path are

$$\dot{\theta}(t) = a1 + 2a_2t + 3a_3t^2$$

$$\ddot{\theta}(t) = 2a_2 + 6a_3t \qquad (5.4)$$

Combining (5.3) and (5.4) with the four desired constraints yields four equations in four unknowns:

$$\Theta_0 = a_0$$

$$\Theta_f = a_0 + a_1t_f + a_2t_f^2 + a_3t_f^3,$$

$$a_1=0,$$

$$a_1 + 2a_2t_f + 3a_3t_f^2 = 0.$$

Solving these equations for the a_i, we obtain

$$\Theta_0 = a_0$$

$$a_1=0$$

$$a_2=3(\Theta_f - \Theta_0) / t_f^2$$

$$a_3= -2(\Theta_f - \Theta_0) / t_f^3 \qquad (5.6)$$

Using (5.6), we can calculate the cubic polynomial that connects any initial joint angle position with any desired final position. This solution is for the case when the joint starts and finishes at zero velocity.If the manipulator is to come to rest at each via point, then we can use the cubic solution of Section 5.3.Usually, we wish to be able to pass through a via point without stopping, and so we need to generalize the way in which we fit cubics to the path

constraints.

As in the case of a single goal point, each via point is usually specified in terms of a desired position and orientation of the tool frame relative to the station frame. Each of these via points is "converted" into a set of desired joint angles by application of the inverse kinematics.Then, we consider the problem of computing cubics that connect the via-point values for each joint together in a smooth way.

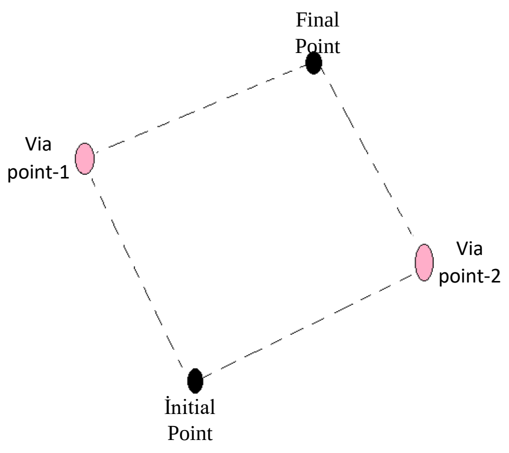


Figure-5.1.1

If desired velocities of the joints at the via points are known, then we can construct cubic polynomials as before; now, however, the velocity constraints at each end are not zero, but rather, some known velocity. The constraints of (5.3) become

$$\dot{\theta}(0)=\dot{\theta}_0$$

$$\dot{\theta}(t_f)=\dot{\theta}_f$$

The four equations describing this general cubic are

$$\Theta_0 = a_0$$

$$\dot{\theta}_0=a_1$$

$$\Theta_f = a_0 + a_1t_f + a_2t_f^{\ 2} + a_3t_f^{\ 3}$$

$$\dot{\theta}_f = a_1 + 2a_2t_f + 3a_3t_f^2 \qquad (5.10)$$

Solving these equations for the we obtain

$$\Theta_0 = a_0$$

$$\dot{\theta}_0=a_1 \qquad (5.11)$$

$$a_2=3(\Theta_f - \Theta_0) / t_f^2-2*\dot{\theta}_f / t_f - \dot{\theta}_f / t_f$$

$$a_3=-2(\Theta_f - \Theta_0) / t_f^3+ (\dot{\theta}_f + \dot{\theta}_0) / t_f$$

When we draw in SolidWorks we get the same results as shown below. The coefficients are found and they will be used for creating two equations in the contol code.

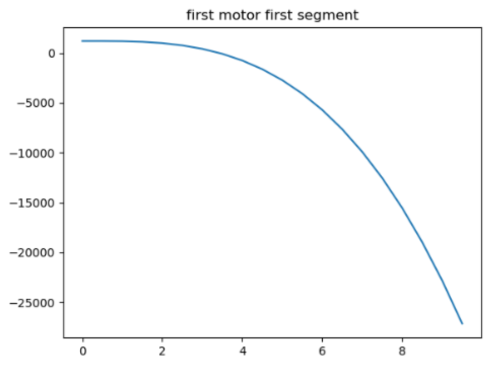


Figure-5.1.2

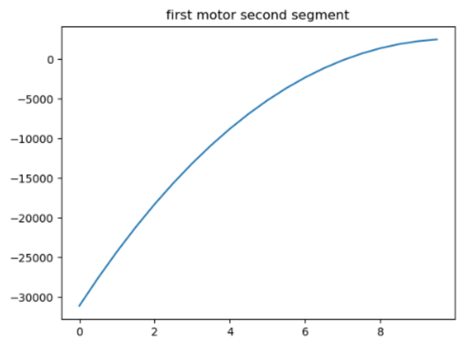


Figure-5.1.3

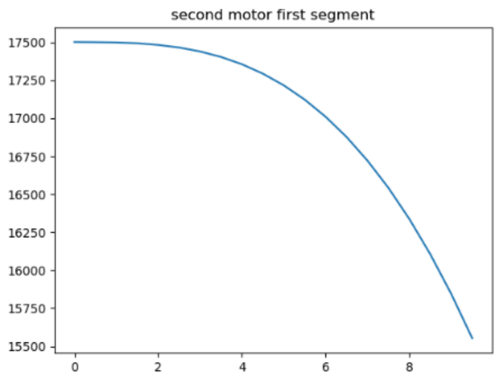


Figure-5.1.4

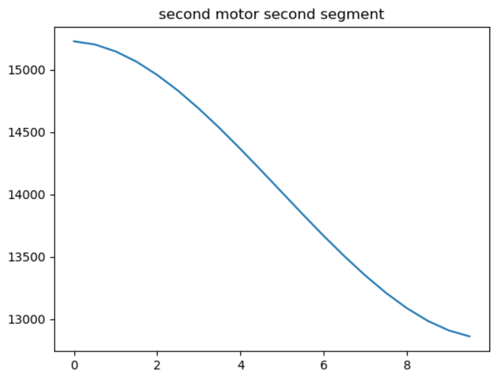


Figure-5.1.5

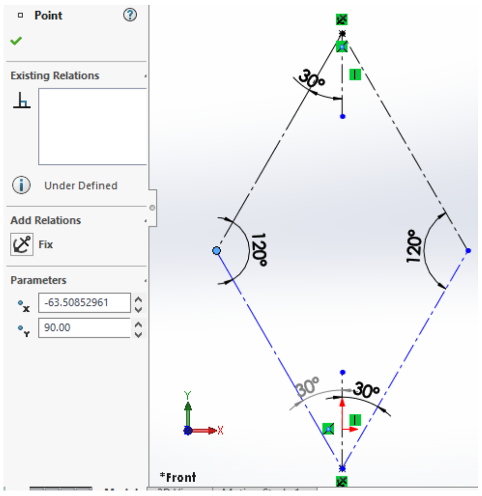


Figure-5.1.6

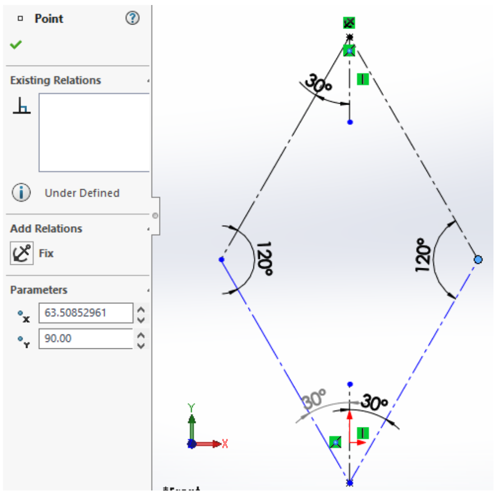


Figure-5.1.7

Control Algorithm

I'2C Communication

The most important thing with integrated squared communication is that, it can be had multiple slaves and multiple masters. This is important because in this study, the data should be sent and recieved either way. Also, this communication is held with only two cables which are RX (receiver) and TX (transmitter). I2C is a serial communication protocol, so data is transferred bit by bit along a single wire. The first data sent is angle values from Atmega to Rasperry Pi so, the first received is the same with the first data sent. The second data sent is inverse angle values from Rasperry Pi to Atmega. If this is not occuring, the project fails.

SDA(Serial Data), this is the line receving and sending occur. SCL(Serial Clock), and this is the one which carries the clock data. I2C is sync with clock. The clock signal is always controlled by the master**.**

For the coefficients in the C1 and C2 matrices, two equations of the via points are created and defined as tls1,tls2 for the first actuator; and defined as t2s1 and t2s2 for the second actuator.Since the maximum 32 bits of data are sent in the I2C communication, we can implement up to 8 precision points for each quadratic equation that we find. So we can send 4×8=32 bit data for two actuator. In order to find the position of the end effector of the robot, we read the angle values of the potentiometers and find the position of the end effector by forward kinematics. These values are expressed in the code with X1 and Y1 variables. The coordinates of the target point are expressed as x3 and y3. Each angle value found as a result of path generation equations could not be sent to arduino with I2C communication. This was because float numbers were included in the array in which the coefficients were placed. However, since I2C communication can send only integer numbers, a scaling process was performed between raspberry and arduino. Each angle value found as a result of path generation equations could not be sent to arduino with I2C communication. This was because float numbers were included in the array in which the coefficients were found. But since the I2C communication could only send integer numbers, a scaling was needed between the raspberry and the arduino, so the float numbers would need to be converted to integer numbers. All the coefficients in the array were multiplied by 100 and all values are coverted to integer numbers and sent to arduino. All the values that came to the Arduino were divided into 100 in arduino and used as floats.

5.2.PURE ROTATION OF PLATFORM

The pure rotation of the platform around the z-axis is provided by two robot manipulators. Initially, two robot manipulators are in the open position, the platform will be manually placed in the gripper's mouth and compressed by the grippers.In figure1 two grippers attached to the platform

The code in Raspberry will start to work and the positions of the end efectors required for the platform to rotate 60 degrees will be calculated(Figure-2). In the inverse task, theta1, theta2, theta3, theta4 angles will be calculated and the command will be sent to the Arduino (slave) by using I2C communication. The code and descriptions are at appendix..111. The movement of the platform continues with the sort shown below.

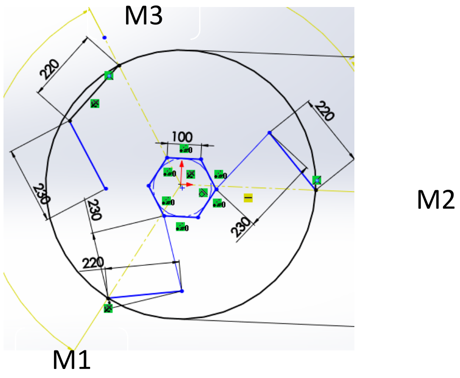


Figure-5.2

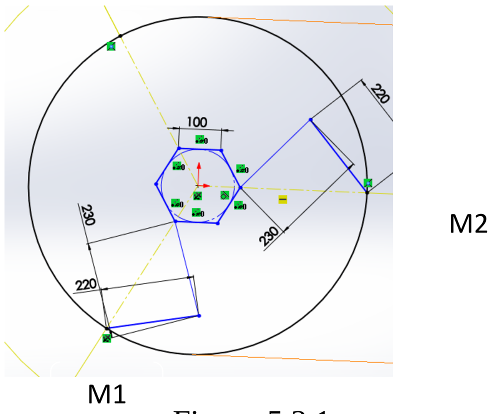


Figure-5.2.1

1-With the M1 and M2 manipulators the platform is rotated 60 degrees in the z-axis.

2-The gripper of third manipulator(M3) opens.It goes to the position according to calculated code in Raspberry and attaches to the platform. As shown in FIG. 3, three manipulators(M1,M2,M3) connected to the platform at that time.

3-The gripper of the M1 opens and detached to the platform(Fig.3).It goes back to the its first position and as in figure 1, it is reconnected to the platform.

4--The gripper of the M2 opens and detached to the platform(Fig.3).It goes back to the its first position and as in figure 1, it is reconnected to the platform.

5-The gripper of M3 opens and detached the platform.It goes back to its first position and waits for the platform to rotate.

This cycle is repeated 6 times and a 360-degree rotation is completed on the z-axis.

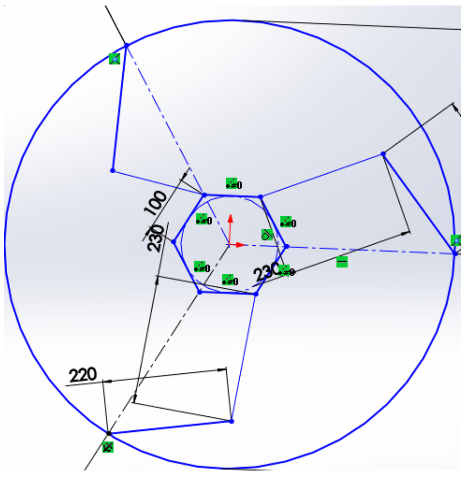


Figure-5.2.3

5.3.DRAWING SQUARE (WITHOUT OBSTACLE)

At the beginning of task, the grippers are attacted to legs manually. Then, the grippers come to defined zero point to start. To do this, Atmegas reads the potentiometer values and converted to angle, then send it to Rasperry Pi3. Inside Rasperry Pi3, there are forward-inverse kinematic analysis and path generation codes. This lets the robot go to desired point, but first with sent angle values forward kinematics are done to find its actual position, then comparing with the zero point coordinates, calculate distance and proper angle by using inverse kinematics, finally Rasperry Pi3 sends back to these last angle values. In the code, there is a for loop. This for loop is the main thing which draw the square.

When i=0, the system goes to first edge of the square. Since it is a hexagonal, and coordinate edges of hexagonal are known, below's code is made by using geometry. This lines are to draw a straight line until the first edge of the square.

……………………………………………………………………

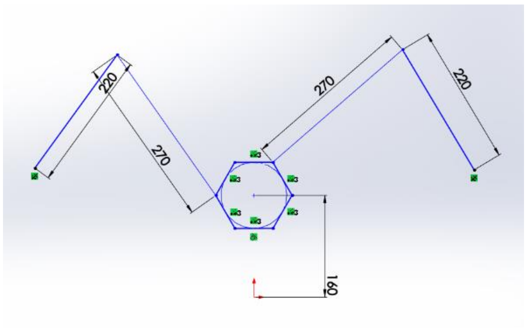


Figure 5.3.1 (first position)

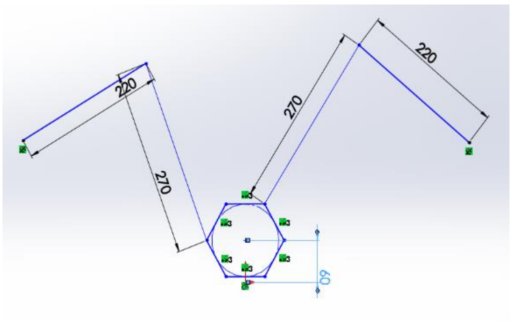


Figure 5.3.2 (last position)

```
for i in [0,1,2,3,4]:

if i==0:

xg21=60*m.cos(60*m.pi/180);

xg23=60*m.cos(60*m.pi/180);

yg21=160+60*m.sin(60*m.pi/180); yg23=100+60*m.sin(60*m.pi/180);

xg31=-60; xg33=-60; yg31=160; yg33=100;

f(xg21,xg23,yg21,yg23,xg31,xg33,yg31,yg33) ##defined function for path generation

print("stationary line ")

time.sleep(1)
```

When i=1, the system start to go second edge of the square in the direction CW.

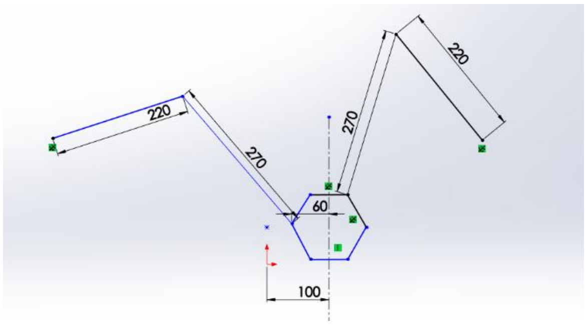


Figure 5.3.3

```
if i==1:

xg21=60*m.cos(60*m.pi/180); xg23=130;

yg21=100+60*m.sin(60*m.pi/180); yg23=100+60*m.sin(60*m.pi/180); ##position w.r.t global frame

xg31=-60; xg33=40; yg31=100; yg33=100;

f(xg21,xg23,yg21,yg23,xg31,xg33,yg31,yg33)

print("square first line CW ")

time.sleep(1)
```

When i=2, the system start to go third edge of the square in the direction CW.

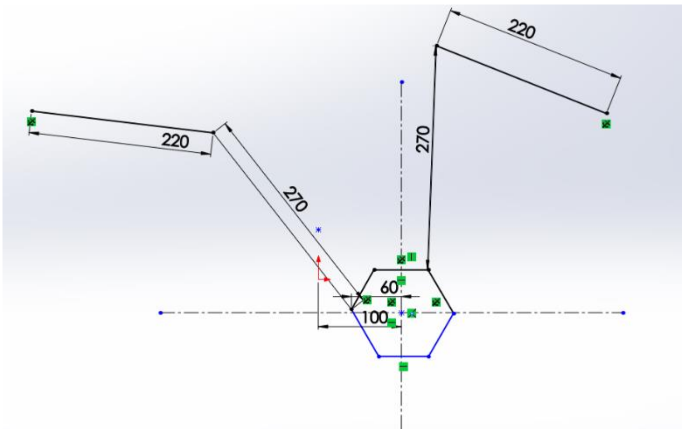


Figure 5.3.4

```
if i==2:

xg21=130; xg23=130; yg21=151.96; yg23=51.96

xg31=40; xg33=40; yg31=100; yg33=0;

f(xg21,xg23,yg21,yg23,xg31,xg33,yg31,yg33)

print("square second line CW ")

time.sleep(1)
```

When i=3, the system start to go last edge of the square in the direction CW.

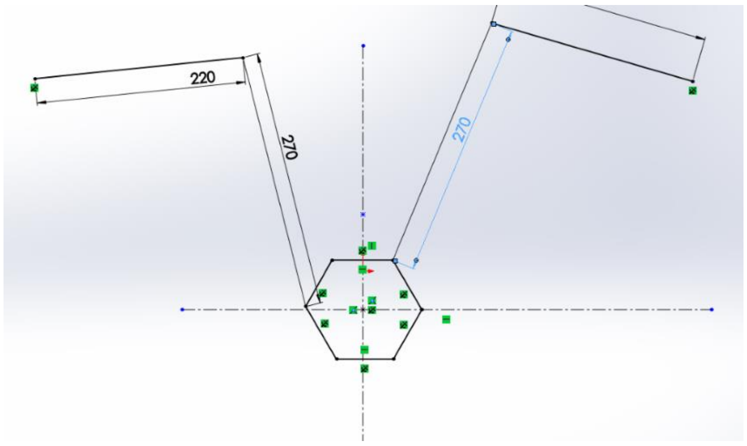


Figure 5.3.5

if i==3:

xg21=100+60*m.cos(60*m.pi/180); xg23=30;

yg21=60*m.sin(60*m.pi/180); yg23=60*m.sin(60*m.pi/180); ##position w.r.t global frame

xg31=40; xg33=-60; yg31=0; yg33=0;

f(xg21,xg23,yg21,yg23,xg31,xg33,yg31,yg33)

print("square third line CW ")

time.sleep(1)

Lastly, it completes the square.

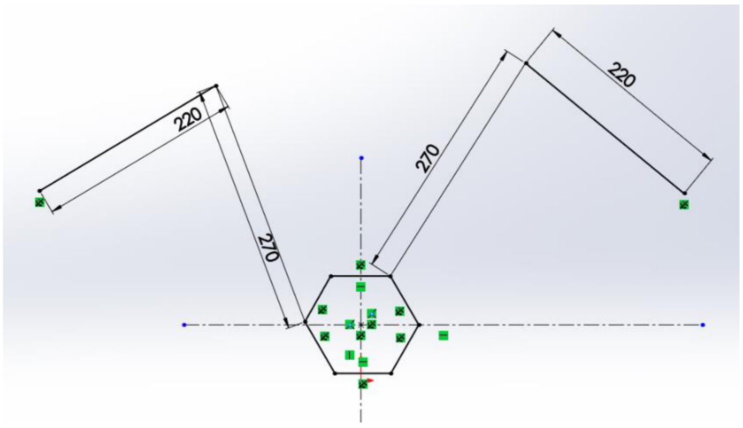


Figure 5.3.6

```
if i==4:

xg21=60*m.cos(60*m.pi/180); xg23=60*m.cos(60*m.pi/180);

yg21=60*m.sin(60*m.pi/180); yg23=100+60*m.sin(60*m.pi/180); ##position w.r.t global frame

xg31=-60; xg33=-60; yg31=0; yg33=100;

f(xg21,xg23,yg21,yg23,xg31,xg33,yg31,yg33)

print("square fourth line CCW ")

time.sleep(1)
```

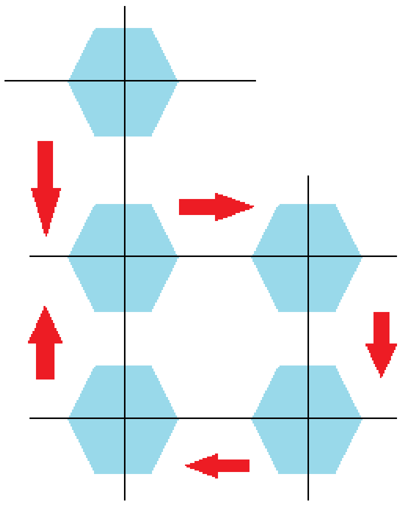


Figure 5.3.7

5.4. DRAWING SQUARE (WITH OBSTACLE)

The difference between with and without obstacle, path generetor should decide one of the solutions. There are two possible solutions, and if one of solutions intercept with the coordinates of obstacle, then the system defines its actual position again as initial positon, final position is always the same, so the rest is to define path function. Normally, it should go straight, but since there is a obstacle, it defines new paths to go like Fig..., and finds its way like this:

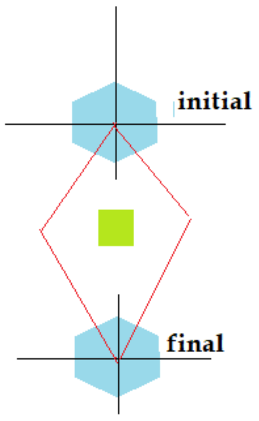


Figure-5.4.8

5.5.IMAGE PROCESSING

Image Processing done by using "opencv" library in Pyhton. This library allows users to do many things with the camera, such as recognizing the shape, color and etc. In this study, light green should be recognized because the obstacle is in this color. First step is to import those libraries. cv2 for opencv, and numpy for arrays. import cv2 import numpy as np

Then, the color's "h, s, v" values are defined in array. Respectively, "hue, saturation and value". Basically hue means the color type such as red, blue, or yellow, ranges 0 to 360 degree and 0 is for red, 180 is for blue and 55 is shade of yellow. Saturation defines the intensity of the color, and ranges 0 to 100. 100 is most intensed color. Lastly, value refers to brightess of the color. Ranges from 0 to 100 which means black to saturated. Figure 1 indicates the HSV system. The HSB model is also known as HSV (Hue, Saturation, Value) model. The HSV model was created in 1978 by Alvy Ray Smith. It is a nonlinear transformation of the RGB color space. In other words, color is not defined as a simple combination (addition/substraction) of primary colors but as a mathematical transformation.*

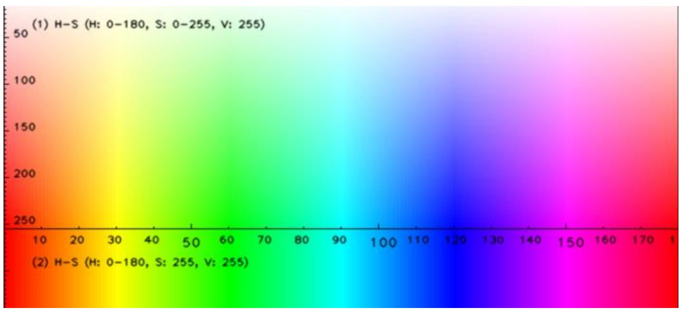


Figure 5.5.1

So, for this study light green should be defined. Figure 2 and 2.1 show for the least case and Figure 3 and 3.1 show for the highest case.

dusuk=np.array([60,50,50])

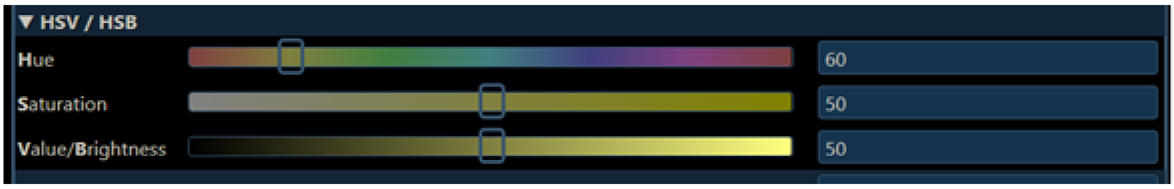

▼ HSV / HSB
Hue 60
Saturation 50
Value/Brightness 50

Figure 5.5.2

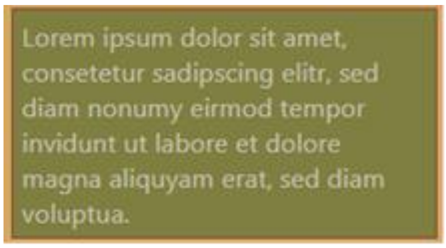
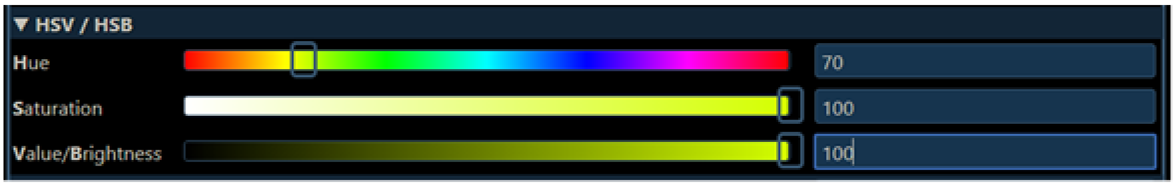

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Figure 5.5.3

yuksek=np.array([70,100,100])



▼ HSV / HSB

Hue 70

Saturation 100

Value/Brightness 100

Figure 5.5.4

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Figure 5.5.5

Then, the camera type is defined such as computer's camera or an external camera. In this study, external camera is used. This definition is made by simply 0 and 1.

kamera=cv2.VideoCapture(1)

After this point, the code goes under a while(1) loop. First step inside the loop is obtaining the frame from the camera, and check if this is true or not. The methods grab the next frame from camera and return.

ret,kare=kamera.read()

The extention of cv2 which is "cvtColor" converts an image from one color space to another. It take two inputs respectively frame and the converted color space. In this line, this is converted RGB to HSV, and new value saved in hsv variable.

hsv=cv2.cvtColor(kare,cv2.COLOR_BGR2HSV)

Other step is to mask all other colors other than green in range is defined as yuksek and dusuk.

mask=cv2.inRange(hsv,dusuk,yuksek)

First line takes only region of green material. bitwise_and() take three object, the frame, and again the frame, and lastly mask value. Second line is to smooth the last image. medianBlur() takes median of all the pixels under kernel area and central element is replaced with this median value. This is highly effective against salt-and-pepper noise in the images. It reduces the noise effectively.

son_resim=cv2.bitwise_and(kare,kare,mask=mask) median=cv2.medianBlur(son_resim,15)

First line converts between RGB/BGR and grayscale. Second line loads an image from a file. imread(filename,[,flags]).

gri=cv2.cvtColor(median,cv2.COLOR_BGR2GRAY) c=cv2.imread("Adsız2.jpg",0)

First line converts between RGB/BGR and grayscale again, and second line converts gri variable into float32 type.

gri=cv2.cvtColor(median,cv2.COLOR_BGR2GRAY) gri=np.float32(gri)

It is good to define strong corners on an image, for this one, foodFeauturesToTrack(image, maxCorners, quality level, min distance). And to use this for further lines, this value is kept in koseler variable, and converted to int type.

```
koseler=cv2.goodFeaturesToTrack(gri,1000000000,0.001,0.05) koseler=np.int0(koseler)
```

After this point, the code contunies under for loop which is defined as kose in koseler. numpy.ravel(array, order = ‘C’) : returns contiguous flattened array(1D array with all the input-array elements and with the same type as it). This for loop is to define the shape of track.

```
for kose in koseler:
```

```
x,y=kose.ravel() cv2.circle(median,(x,y),10,(0,255,0),-1) cv2.line(median,(x,y),(x+3,y+3),(0,255,0),10)
```

End of the for loop, here comes another function which is called "Canny()". Basically, it detects the edges of the image. Second and third arguments are our minVal and maxVal respectively. Third argument is aperture_size.

```
kenarlar=cv2.Canny(kare,50,100)
```

In this line, the inverse of green object will be taken, and the second one is to black-out the area of object.

```
mask_inver=cv2.bitwise_not(kenarlar)andleme=cv2.bitwise_and(median,median,mask=mask_inver)
```

Those lines are for to display the cams.

```
cv2.imshow("gri",gri) cv2.imshow("median",median) cv2.imshow("sonuc",andleme)
```

This is delay.

```
if cv2.waitKey(25)& 0xFF==ord('q'): break
```

This is the end of the lines.

```
kamera.release() cv2.destroyAllWindows()
```

CONCLUSION

As conclusion, this manipulators are designed in a way that reconfigurable, attachable and deteachable. By any changes in coding algorithm, these manipulators can be used in vary areas such as laser cutting, soldering, pick-and-placing. If there is a obstacle, the grippers can escape, for this study this obstacle is defined as green object but it can be easily changed to another color by playing with openCV variables. Also the coordinates are defined as variables that make easier to change such parameters. So, it can be adapted for any obstacles and designing shapes thanks to path generation. But, the handicape is the usage of potentiometer instead of servo motor. Because potentiometers may give false results in certain ranges. To have such system, many calculations and simulations are done. As calculations, inverse and forward kinematic analysis, torque and velocity calculations, path generation. For simulations, Mathematica, SolidWorks are used to verify the result with the coding. In the end, these manipulators are designed with lots of calculations, mechanism knowledge, and coding knowledge.

REFERANCES

1-https://www.techopedia.com/definition/13274/servo-motor

2- https://electronicsforu.com/resources/learn-electronics/mosfet-basics-working-applications

3-https://tutorial.cytron.io/2012/06/22/pid-for-embedded-design/

1f-https://github.com/leventcetin/MEE210/blob/master/LN11.pdf

2f- https://github.com/mee427/mee427.github.io/blob/master/PID%20ZN%20Tuning.pdf

3f- https://tutorial.cytron.io/2012/06/22/pid-for-embedded-design/

Fig2-https://www.researchgate.net/profile/Jean-Pierre_Merlet2/publication/263076422/figure/fig1/AS:341315150729216@1458387281079/The-3-P-RR-planar-parallel-manipulator.png

Fig3. http://yapbenzet.kocaeli.edu.tr/wp-content/uploads/SerialRobot.png

3) https://link.springer.com/chapter/10.1007/978-94-007-4522-3_6

4) https://en.wikipedia.org/wiki/Mechanical_singularity

5-Lung - Wen Tsai, Robot Analysis: The mechanics of Serial and Parallel Manipulators, John Wiley and Sons,1999