

# **Introduction to Mechatronics and Robotics**

**A Summer Intern Project**

*Submitted by*

**Prateek Yadav**

**SR Number: 11-01-00-10-91-14-1-10991**

*To*

**Prof. Subir Kumar Saha**

*(Professor, IIT Delhi, Mechatronics Lab)*

*In partial fulfilment of Summer Internship for the award of Degree*

**Bachelor of Science (Research)**



**Indian Institute of Science**

**C V Raman Ave, Bengaluru, Karnataka 560012**

*Summer 2015*

# **CERTIFICATE**

Certified that the summer internship project report “**Introduction to Mechatronics and Robotics**” is a bonafied work of Prateek Yadav , SR Number: 11-01-00-10-91-14-1-10991 , 1<sup>st</sup> year BSc (Research) in Indian Institute of Science, Bangalore, Karnataka carried out under my supervision for a period of 6 weeks from 12<sup>th</sup> May – 24<sup>th</sup> June 2015.

Place: IIT Delhi

Date: 24<sup>th</sup> June 2015

**Professor Subir Kumar Saha**

Mechatronics Lab

Department of Mechanical Engineering

Signature

Name

## **Acknowledgements**

I would like to thank Dr. Subir Kumar Saha, Prof, Department of Mechanical Engineering, IIT-Delhi for giving me opportunity to undertake the summer internship. I wish to express my profound gratitude and indebtedness to him for introducing the present topic and for his inspiring guidance, constructive criticism and valuable suggestions throughout the course of internship.

I also thank Mr. Sasanka Sekhar Sinha, MTech student, Department of Mechanical Engineering, IIT Delhi for his invaluable suggestions and guidance during this period. My sincere thanks to all the members of Mechatronics Lab of IIT Delhi for all the encouragement and support they have extended at the beginning of this undertaking.

Prateek Yadav

Indian Institute of Science, Bangalore

# **ABSTRACT**

The time we currently live in is experiencing exponential growth in the field of Science and Engineering, the world is progressing at a very fast rate and is switching to high level of automations in each and every field. Machines are replacing human being at each and every possible place and for that to happen Autonomous Robotic system are required.

So some theoretical and practical knowledge related to this field can be of great use. My project was basically related introduction to Robotics, different types of mechanisms and machines, programming aspect of Robotics and modelling of Robotic mechanical structures, printed circuit boards, including some work on Stewart Platform used in flight simulators.

**Keywords:** Stewart Platform, Parallel Manipulators, Matlab, SolidWorks, CAD, Four Bar Mechanism, Robo-Analyser, Animation.

# Table of Contents

CHAPTER 1: THEORY OF MACHINES .....	7
CHAPTER 2: ANIMATING STEWART PLATFORM (MATLAB) .....	15
CHAPTER 3: INTERFACING AND CODING .....	17
CHAPTER 4: AUTOMATING STEWART PLATFORM (BATTERY & PCB).....	18
CHAPTER 5: FOUR BAR MECHANISM.....	21
CHAPTER 6: DESIGNING BASE.....	25
REFERENCES.....	27
APPENDIX A: ANIMATION CODES OF STEWART (MATLAB).....	28
APPENDIX B: ARDUINO CODE FOR TRAJECTORIES.....	32
APPENDIX C: FOUR BAR ANIMATION CODE.....	43
APPENDIX D: ALL MATERIAL RELATED TO PROJECT.....	48

## *List of Figures*

Figure 1: Types of Links.....	7
Figure 2: Four Bar Mechanism.....	8
Figure 3: Different Types of Mechanisms.....	9
Figure 4: connected, unconnected links.....	10
Figure 5: Open closed Mechanisms.....	10
Figure 6: Cylindrical joint.....	11
Figure 7: Spherical Joint.....	11
Figure 8: Example Degree of Freedom.....	11
Figure 9: Mechanism and Structure.....	12
Figure 10: Transmission Angle.....	12
Figure 11: Velocity Analysis (Graphically).....	13
Figure 12: Instantaneous Center of Velocity.....	13
Figure 13: Acceleration Analysis.....	13
Figure 14(a): Model of Stewart.....	15
Figure 14: Stewart design.....	15
Figure 14(C): Designed Model of Stewart.....	16
Figure 15: Defining Trajectory.....	16
Figure 16: Lipo Battery.....	18
Figure 17: Voltage Regulator (LM7805).....	18
Figure 18: PCB.....	18
Figure 19(a): PCB Assembly.....	19
Figure 19(b): Designed Printed Circuit Board.....	19
Figure 20: Four Bar Mechanism.....	21
Figure 21: Inversions of Four Bar Mechanism.....	21
Figure 22: Double Crank Mechanism.....	22
Figure 23: Crank-Rocker Mechanism.....	22
Figure 24: Double Rocker Mechanism.....	23
Figure 25: Parallel Crank Mechanism.....	23
Figure 26: FBM Solution.....	24
Figure 27(a): Modified Base Design.....	25
Figure 27(b): Modified Base Design.....	26

## **CHAPTER 1: THEORY OF MACHINES**

**Theory of Machines** may be defined as a branch of Engineering-Science, which deals with the study of relative motion between different parts of a machine, and forces which act on them. The knowledge of this subject is very essential for an engineer in designing various parts of a machine.

**Kinematics:** The study of motion without regards of force. More particularly, it is the study of position, displacement, rotation, speed, velocity and acceleration.

**Kinetics:** The study of forces on a system in motion.

**Mechanism:** A mechanism is a device that transform the motion to some desirable pattern and typically develops very low force and transmits little power.

**Machine:** it typically contains mechanism that are designed to provide significant force and transmit significant power.

### **Link, Joints, and Kinematic Chains**

**Node:** is an attachment point

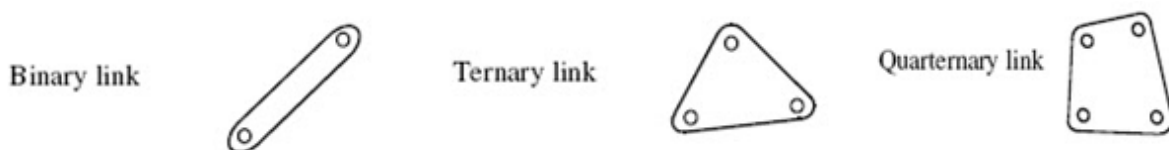
**Links:** rigid member having nodes.

**Joints (Kinematic Pairs):** connection between two or more links (at their nodes) which allows motion.

### **Linkage Design:**

- Linkages are basic building block of all mechanisms.
- Linkages are a combination of links and joints.

### **Types of Links:**



*Figure 1: Types of Links*

Binary Link: 2 Nodes

Ternary Link: 3 Nodes

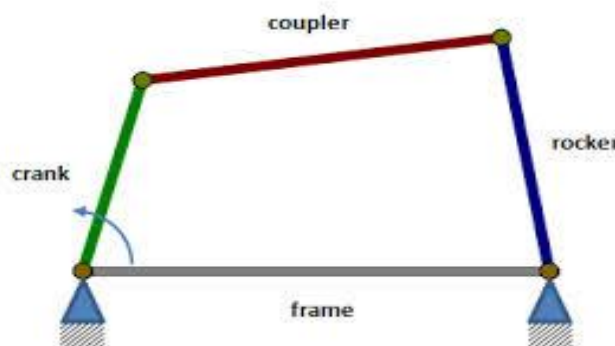
Quarternary Link: 4 Node

## Terminologies of Joints:

- A **Joint** (also called kinematic pair) is a connection between two or more links at their nodes, which may allow motion between the links.
- A **Lower Pair** is a Joint with surface contact, a **Higher pair** is a joint with point or line contact.
- A full joint has one degree of freedom; a half joint has two degrees of freedom. Full joints are lower pairs; half-joints are higher pairs and allow both rotation and translation (roll-slide).
- A form-closed joint is one in which the links are kept together form by its geometry; a force-closed joint requires some external force to keep the links together.
- Joint order = number of links - 1 (e.g. 1<sup>st</sup> order means two links).

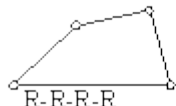
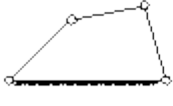
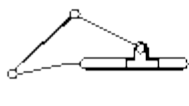
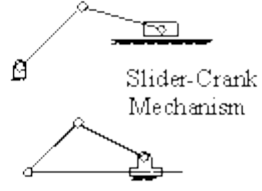
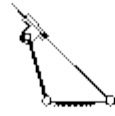
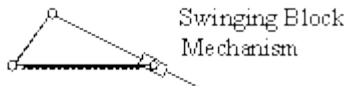
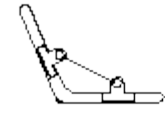
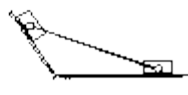


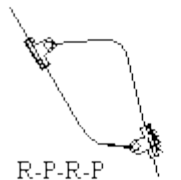
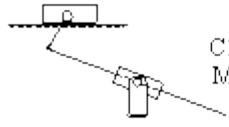
## Kinematics Chain, Mechanisms, Machines, Link Classification:

- **Kinematics Chain:** Links joined together for motion.
- **Mechanism:** Grounded Kinematics Chain.
- **Machines:** Mechanism designed to work.
- **Link Classification:**
  - **Ground:** Any link or links that are fixed, non-moving with respect to their reference frame.
  - **Crank:** Pivoted to ground, makes complete revolutions.
  - **Rocker:** Pivoted to ground, has oscillatory motion.
  - **Coupler:** Link has complex motion, not attached to ground.



*Figure 2: Four Bar Mechanism*



 R-R-R-R	 Four-Bar Mechanism		
 R-R-R-P	 Slider-Crank Mechanism	 Inverted Slider or Quick-Return Mechanism	 Swinging Block Mechanism
 R-R-P-P	 Double Slider	 Scotch-Yoke Mechanism	 Oldham Coupling
 R-P-P-P	 Chebyshev Motion Mechanism		

*Figure 3: Different Types of Mechanisms*

### Degree of Freedom:

- It is the number of independent parameters required to uniquely define the position of a system in space at any instant of time.
- DOF is defined with respect to a selected frame of reference (ground).
- Rigid body in a plane has 3 DOF:  $x, y, z$
- Rigid body in 3D-space has 6 DOF, 3 translations & 3 rotations □ three lengths( $x, y, z$ ), plus three angles ( $\theta, \phi, \rho$ ).

### Types of Motion:

- **Pure rotation:** The body possesses one point (center of rotation) that has no motion with respect to the “stationary” frame of reference. All other points move in circular arcs.
- **Pure translation:** All points on the body describe parallel (curvilinear or rectilinear) paths.
- **Complex motion:** A simultaneous combination of rotation and translation.

## Determining Degree of Freedom:

- Gruebler's equation for planar mechanisms

$$\text{DOF (M)} = 3L - 2J - 3G = 3(L-1) - 2J$$

Where

M = degree of freedom or mobility

L = number of links

J = number of full joints (half joints count as 0.5)

G = number of grounded links = 1

- Two unconnected links: 6 DOF (each link has 3 DOF)
- When connected by a full joint: 4 DOF (each full joint eliminates 2 DOF)

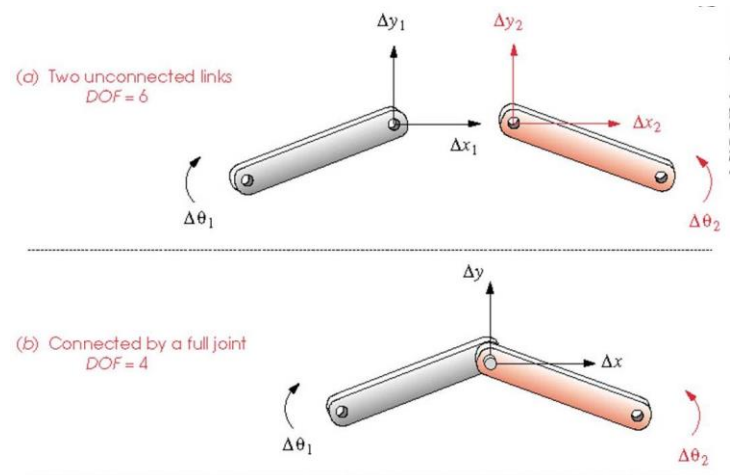


Figure 4: connected, unconnected links

- An open kinematics chain will have more than one degree of freedom.
- A closed kinematics chain will have no open nodes and may have one or more degree of freedom.

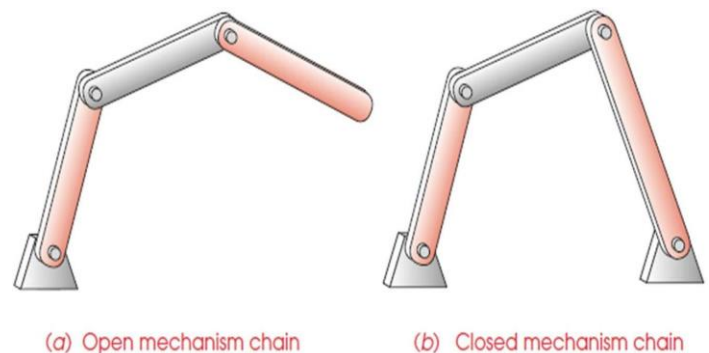


Figure 5: Open closed Mechanisms

**The Cylindrical (cylindric) joint** -two degrees of freedom. It permits both angular rotation and an independent sliding motion (C joint).

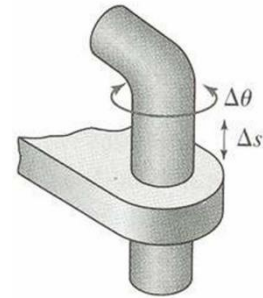
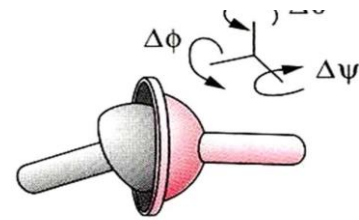


Figure 6: Cylindrical joint

**The Spherical (spheric) -Three degree of freedom.** It permits rotational motion about all three axes, a ball-and-socket joint (S joint).



Spherical (S) joint—3 DOF

Figure 7: Spherical Joint

Note:  
There are no  
roll-slide  
(half) joints  
in this  
linkage

$$L = 8, \quad J = 10$$

$$DOF = 1$$

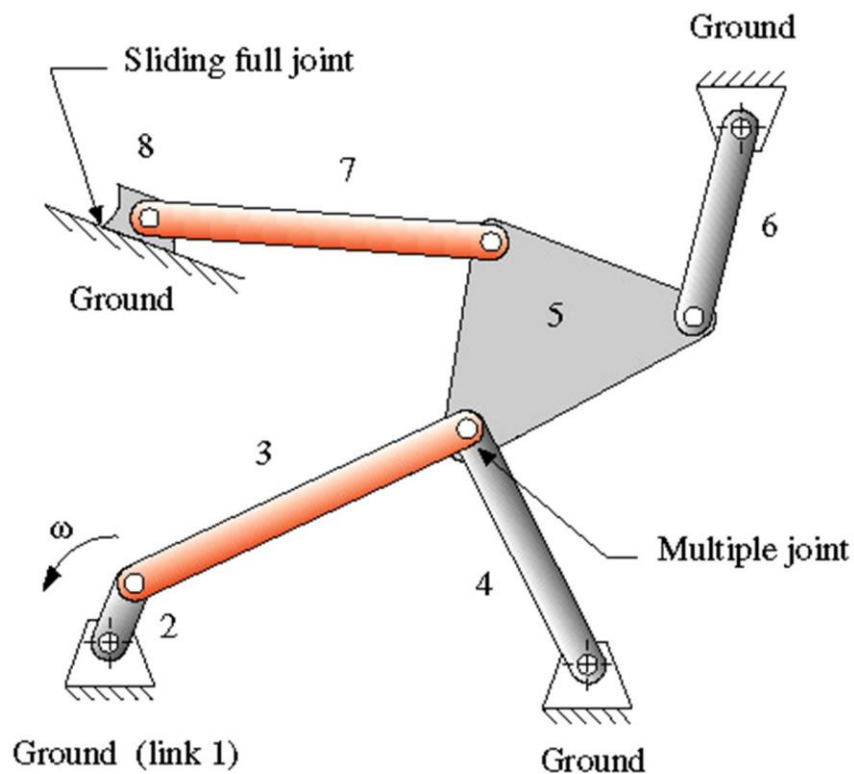


Figure 8: Example Degree of Freedom

## Mechanisms and Structures:

- If  $\text{DOF} > 0$ , the assembly of links is a mechanism and will exhibit relative motion
- If  $\text{DOF} = 0$ , the assembly of links is a structure and no motion is possible.
- If  $\text{DOF} < 0$ , then the assembly is a preloaded structure, no motion is possible, and in general stresses are present.

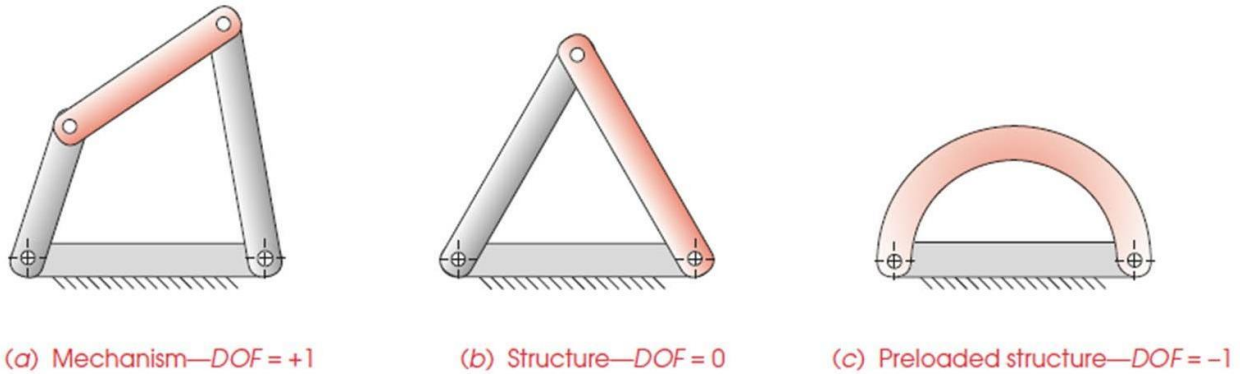


Figure 9: Mechanism and Structure

**Transmission angle:** For a 4R linkage, the transmission angle ( $\mu$ ) is defined as the acute angle between the coupler (AB) and the follower (BO4).

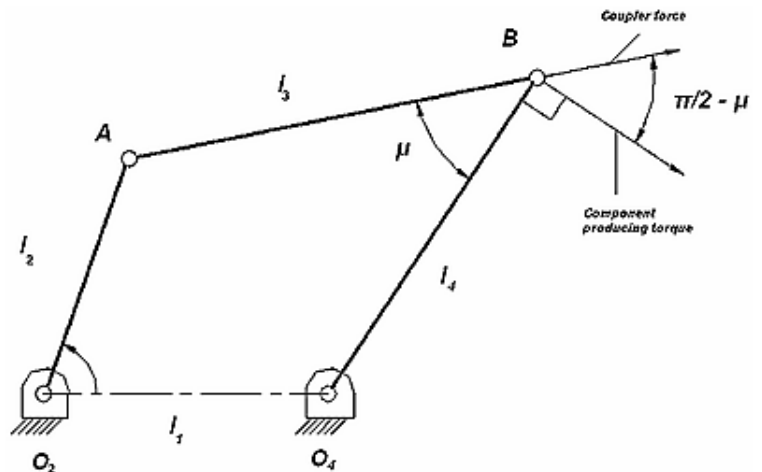


Figure 10: Transmission Angle

**Velocity Analysis:** A graphical technique for the determination of the velocities of the parts of a mechanical device, especially those of a plane mechanism with rigid component links.

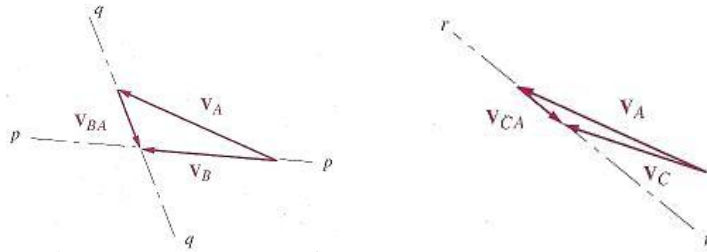


Figure 11: Velocity Analysis (Graphically)

### Instantaneous Center of Velocity:

- An instant center of velocity is a point, common to two bodies in plane motion, which point has the same instantaneous velocity in each body.
- The numbers of IC is calculated with;  

$$C = \frac{n(n-1)}{2}$$
- **Kennedy's Rule** – Any three bodies in plane motion will have exactly three instant centers, and they will lie on the same straight line.

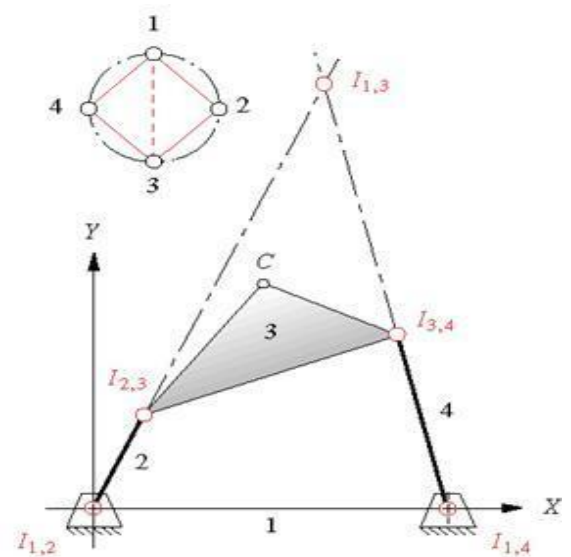
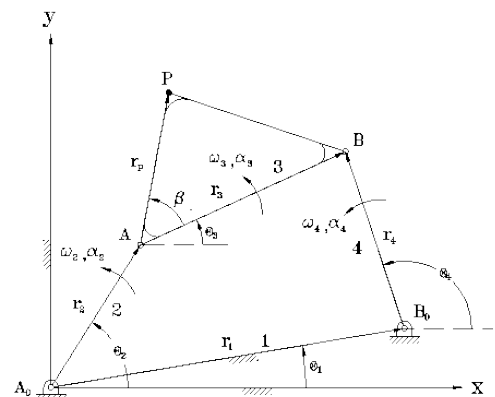


Figure 12: Instantaneous Center of Velocity

### Acceleration Analysis:

- Angular ( $\alpha$ ) = rate of change in angular velocity
- Linear ( $a$ ) = rate of change in linear velocity

Figure 13: Acceleration Analysis



**RoboAnalyser:**

Robo-Analyser is a software designed to understand basics of Robotics with the help of visual animation and high level of customization while animating changing type of mechanisms, increasing number of links, DOF, DH parameters. It helps to learn the forward/inverse kinematics of robotic structures.

This software is being developed at Mechatronics lab, Mechanical Engineering Department, IIT Delhi under the guidance of Prof S K Saha.

## **CHAPTER 2: ANIMATING STEWART PLATFORM (MATLAB)**

For this part of the project there was an additional requirement i.e proficiency in using MATLAB. Due to no prior experience of working with Matlab, the first task was to get familiar with matlab and gain proficiency so that all the required tasks can be performed. So I started learning matlab by the help of online tutorials and mathwork forums, getting familiar with syntax like plot, surf, making required shapes, creating data structures, matrix operations etc.

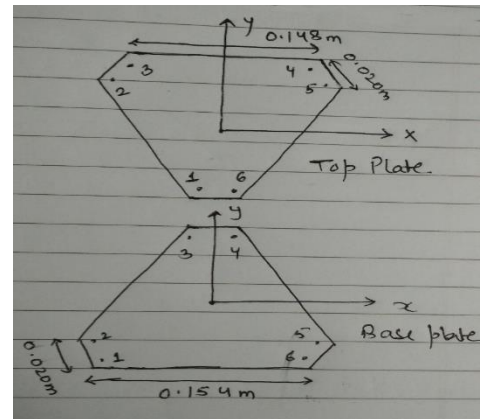


*Figure 14(a): Model of Stewart*

### **Process of Creating a Model of Stewart Platform:**

Finding Out the co-ordinate of the base plate assuming the Geometrical center of the base plate origin (co-ordinates are calculated in base plates frame) with these parameters base longer width= 0.154m, base shorter width= 0.020m. Then calculating the co-ordinates (co-ordinates are calculated in top plates frame) of the top plate assuming its geometrical center to be origin for its co-ordinate with these parameters top longer width= 0.148m and top shorter width= 0.020m

(x, y, z) are the co-ordinates of top plate's center in frame of bottom plate. So now we add (x,y,z) to the co-ordinates we calculated for the top plate in its own frame. What we just did is that we created a common origin i.e base plate's geometrical center. So we will use 3D fill to create the plates using the co-ordinates in base frame.



*Figure 14(b): Stewart design*

Now we have to join all the points with same number on them by using 3D lines.

Last thing is to just improve upon the graphics aspect of the model and that depends on personal expertise in Matlab.



## Animating the Created Model on different Trajectories:

In matlab there is only one way to create animation and that is by using a loop inside which we keep on making small changes to the previous model and then running it all with some delay so that we can see those sets of models changing at a rate more than 25fps and thus looking like an animation to human eyes.

For animation Stewart platform we change the co-ordinates of the Geometrical center of the top plate in its own frame and run that in a loop.

Now the only task left is defining trajectories for the center of top plate which can be done as shown in figure.

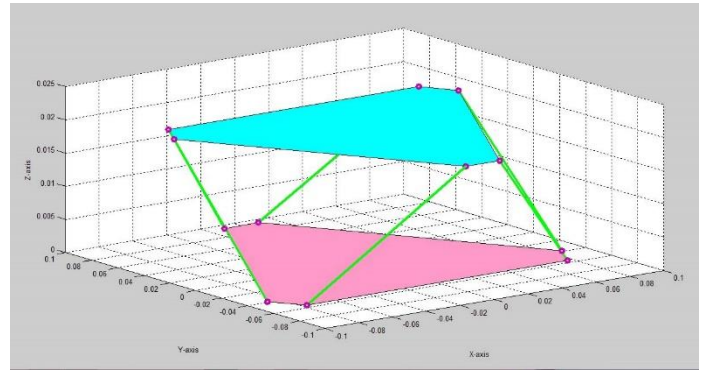


Figure 14(C): Designed Model of Stewart

Now we run this in a loop for some time with theta depending on time for making an animation of circular trajectory. The trajectories defined by me during the period of project were Horizontal Circle, Vertical Circle, and Horizontal eight.

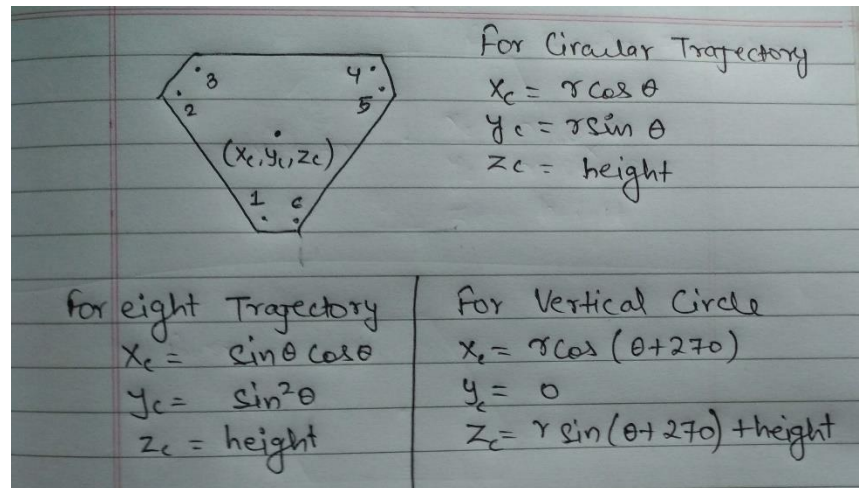


Figure 15: defining Trajectories

### Horizontal circle Animation:

<https://drive.google.com/open?id=0B9XUd9XNloWtVlhUckx2cmU4Z0k&authuser=0>

### Vertical Circle Animation:

<https://drive.google.com/open?id=0B9XUd9XNloWtYU9rYmh5ZzREdWM&authuser=0>

### Eight shaped Animation:

<https://drive.google.com/open?id=0B9XUd9XNloWtdV92YXhkQU5sa0U&authuser=0>



## **CHAPTER 3: INTERFACING AND PROGRAMMING**

This task was one of the major work of the whole project. The main thing included in it were:

- Creating a circuit connecting Arduino Board, Servo motors, power supply and Computer.
- Writing an Arduino code which include the following things:
  1. Connecting Arduino to the Servo motors.
  2. Defining the co-ordinates of the top and base plate.
  3. Defining trajectory to be run on Platform
  4. Inverse kinematics of Stewart platform to return leg lengths of servo motors.
- Debugging the Code for errors

### **Connecting Arduino and Servo motors:**

We first have to define a servo motor in Arduino then set a position that is considered the zero/no extension state of servo motor and then calculate distance between the two points connecting a servo motor, then calculate its range and the return a value in terms of its extension.

To define and attach a servo the code is:

```
#include <Servo.h>

Servo ref_servo;

const int servo_zero=1000;

ref_servo.attach(servo_pin_num);
```

Full code is in **APPENDIX B**

### ***Circular Trajectory Video:***

<https://drive.google.com/open?id=0B9XUd9XNloWtX2dCUkRDaU9yelk&authuser=0>

<https://drive.google.com/open?id=0B9XUd9XNloWta3lLcIJrN3M2V2M&authuser=0>

### ***Eight Shaped Trajectory:***

<https://drive.google.com/open?id=0B9XUd9XNloWtLUVHqXdzZDVLMVE&authuser=0>

<https://drive.google.com/open?id=0B9XUd9XNloWtMDRjXzBYbjR3TWc&authuser=0>

## **CHAPTER 4: AUTOMATING STEWART PLATFORM** **(BATTERY & PCB)**

This task involves the making the platform autonomous i.e completely independent of external circuits, wires, breadboard, and power supply with the help of a Lipo battery, a Voltage regulator (LM7805) and a printed circuit board (PCB).

**Lipo Battery:** It is a lithium polymer rechargeable battery of lithium ion technology in a pouch form, it lacks rigidity and is in a form of pouch. It can supply a maximum voltage of 7.2 ~7.5 V. As the time goes on the output voltage goes on decreasing. So here we need a constant 5V supply for which we use a LM7805 chip which supplies a constant output voltage of 5V.

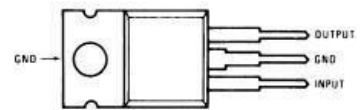


*Figure 16: Lipo Battery*

**LM78M05:** This chip is a voltage regulator which can take input voltage up to 25V and supply a constant output voltage of 5V at 25 degree centigrade until the input voltage is decreased to 7V with a maximum output current of 0.5A. So our purpose can be full filled by the use of this chip in our circuit.

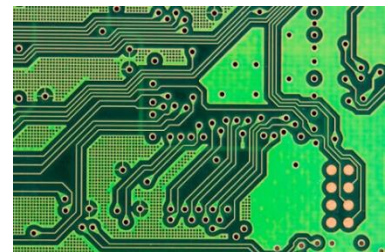


LM7805  
Voltage Regulator  
TO-220



*Figure 17: Voltage Regulator (LM78M05)*

**Printed Circuit Boards:** PCB mechanically supports and electrically connects electronic components using conductive tracks, pads and other features like copper sheets laminated onto a non-conductive substrate.



*Figure 18: PCB*

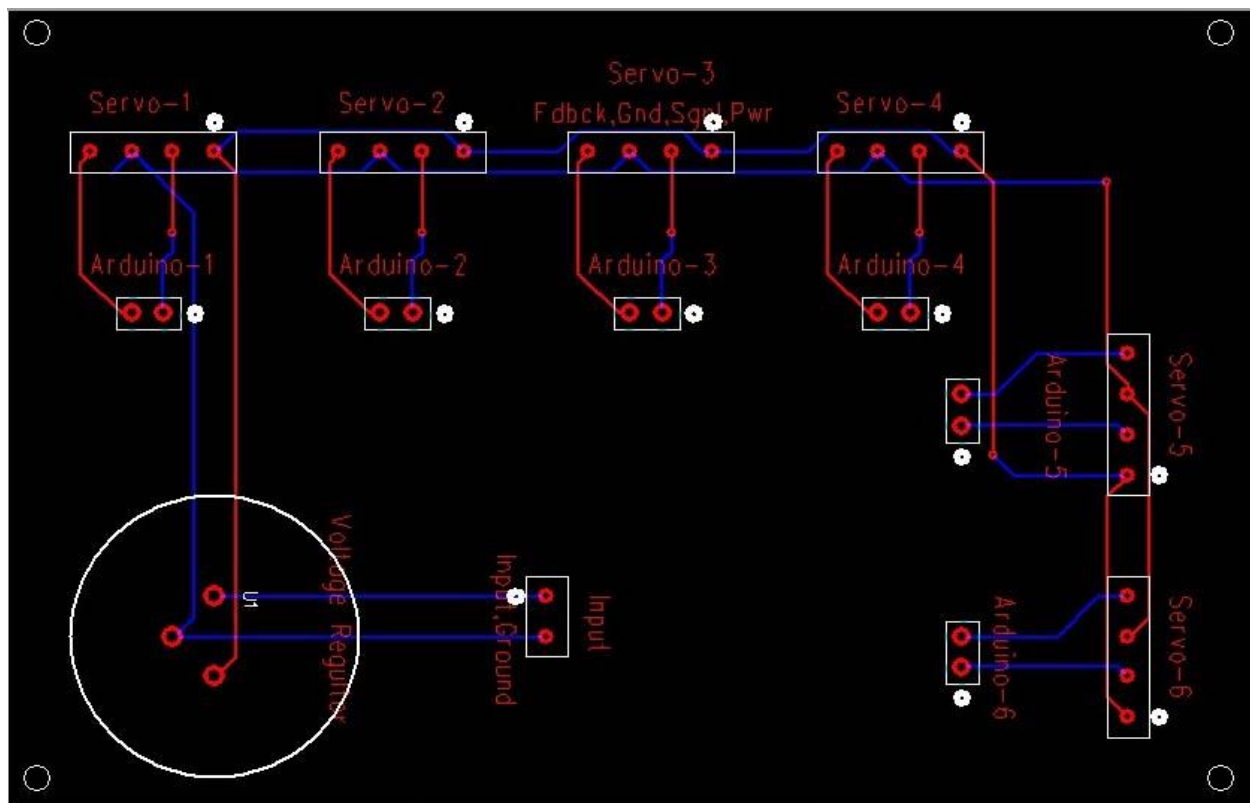
**Printed circuit board assembly:** It is an advanced PCB containing additional components like capacitors, resistors and other chips embedded in the substrate.

PCB are useful as they have no wire and all things are internally connected, human made errors are not applicable here, compact and easy to handle.



*Figure 19(a): PCB Assembly*

**Designing PCB:** Now the last thing left is to design a PCB according to our circuit in which we have 6 servo motors each containing 4 wire ( Power, Ground, Signal, Feedback), so on our PCB we need 6 slots of 4 wires and in which all Powers slots are connected to each other and to the slot for main power supply, in which all Grounds slots are connected to each other and to the slot for Ground of supply, all signal slots and all feedback slot are connected to their corresponding slots in Arduino. If we want it to be battery powered the we need to have a LM78M05 on our



**Figure 19(b): Designed Printed Circuit Board**

board too whose output voltage will be input for all servo's and input supply will be from the battery. The ground for battery, LM 7805 and the servo's is same.

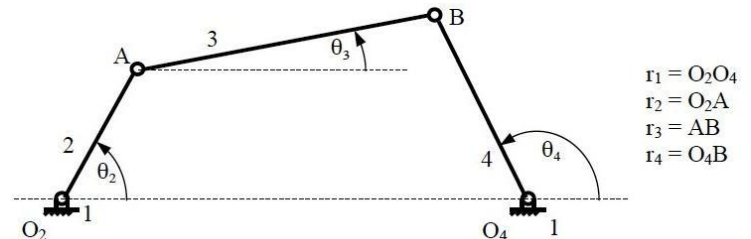
For this task I used PCB123 a professional tool for this purpose, we require all the model number of all the components that are going to be used in our assembly like wire to board connectors, chips etc because designing a PCB need all the physical dimensions of such components. so we create a digital footprint of all the components that we are going to use according to their length, width, pin width, pin length, pin hole diameter, and after that we have to use these digital footprint to make a Schematic Diagram that looks more like a diagram drawn on a paper but is digital and then the software converts the schematic diagram to a messed up layout on the board whose dimensions, number of layers, and type. Then we just have to arrange it properly and make sure that connections are not present at places where chips are present. This Part Could Not Be Completed Due To Lack of Information Related to Model of Parts Going To be Used.

***Battery Powered test (Circular Trajectory):***

<https://drive.google.com/open?id=0B9XUd9XNloWtU1pRdHdkV3ZPc0k&authuser=0>

## CHAPTER 5: FOUR BAR MECHANISM

**Grashoff's Law:** For a planar four bar linkage, the sum of the shortest and longest links cannot be greater than the sum of the remaining links if there is to be continuous relative rotation between two members.

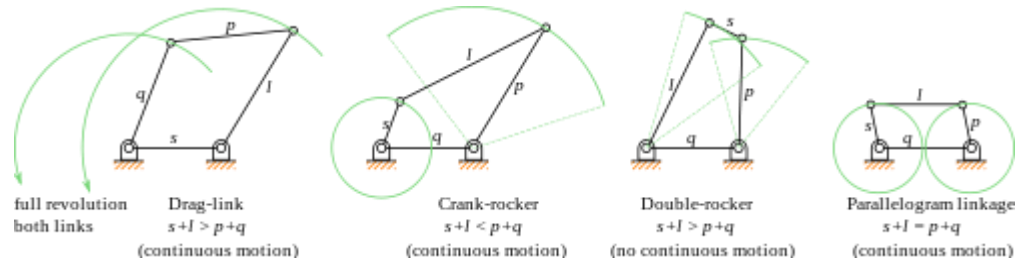


*Figure 20: Four Bar Mechanism*

### **Inversions:**

A mechanism has been defined above as a kinematic chain in which one of the links is fixed. From the four bar mechanism, different versions of each of them can be obtained by fixing any one of the links p, q, l or s. Such different versions, which can be obtained by fixing any of the different links, are called its “Inversions”. Many a time, a particular inversion of a mechanism may give rise to different mechanisms of practical utility, when the proportions of the link lengths are changed.

By this principle of inversion of a four bar chain, several useful mechanisms can be obtained.



*Figure 21: Inversions of Four Bar Mechanism*

There are three inversions of four bar mechanisms, which are obtained by fixing different links of the kinematic chain. They are:

1. Double Crank Mechanism
2. Crank Rocker Mechanism
3. Double Rocker Mechanism

### Double Crank Mechanism:

A double crank converts rotary motion from a crank to a second crank or link in a different plane or axis. It is also known as crank-crank, drag-crank or rotary-rotary converter. The links p, q and l shown above rotate through one complete revolution. This is one of the first inversions of four-bar mechanisms.

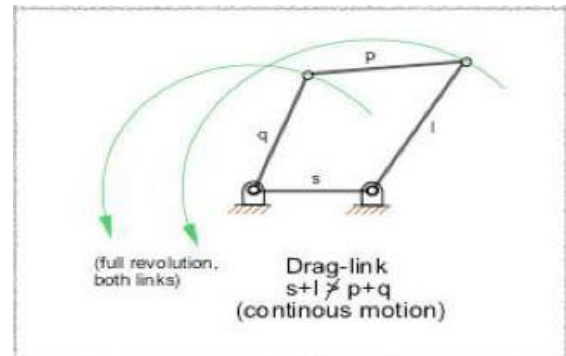


Figure 22: Double Crank Mechanism

### Crank-Crank Animation:

<https://drive.google.com/open?id=0B9XUd9XNloWtdG9GRk02a3piT2M&authuser=0>

### Crank-Rocker Mechanism:

In a four bar linkage, if the shorter side link revolves and the other rocks (i.e., oscillates), it is called a crank-rocker mechanism.

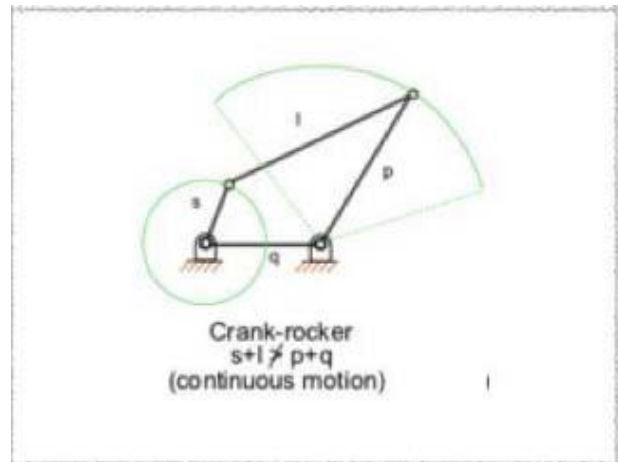


Figure 23: Crank-Rocker Mechanism

### Crank-Rocker Animation:

<https://drive.google.com/open?id=0B9XUd9XNloWtZXVJd2VBSE5VS00&authuser=0>

### Double-Rocker Mechanism:

A linkage in which no link undergoes entire 360-degree revolution but only oscillations is known as a double-lever mechanism.

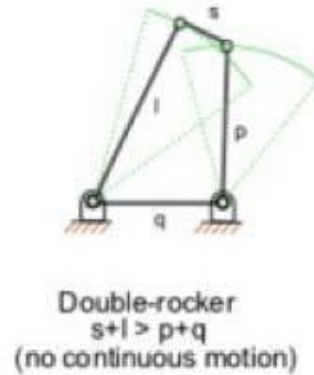


Figure 24: Double Rocker Mechanism

### Parallel Crank Mechanism:

If in a 4 bar linkage, two opposite links are parallel and equal in length, then any of the links can be made fixed, regardless, the two adjacent links will always act as a pair of cranks, i.e. , both will have complete revolution about their joints on the frame.

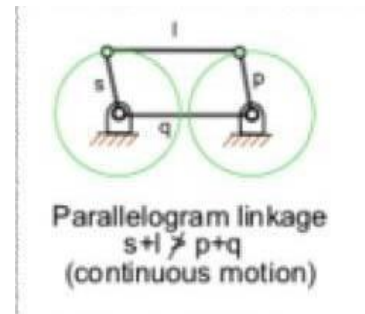


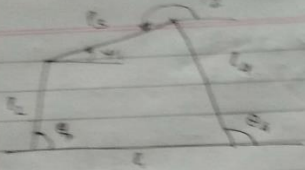
Figure 25: Parallel Crank Mechanism

### Parallel Crank Animation:

<https://drive.google.com/open?id=0B9XUd9XNloWtdFQwenZBaGIUNGS&authuser=0>



## Solution of Four bar Mechanism:



$$\begin{aligned}
 l_2 \cos \theta_2 + l_3 \cos \theta_3 &= l_1 + l_4 \cos \theta_4 \\
 l_2 \sin \theta_2 + l_3 \sin \theta_3 &= l_4 \sin \theta_4
 \end{aligned}$$

$$\begin{aligned}
 &\left( l_1 + l_4 \cos \theta_4 - l_2 \cos \theta_2 \right)^2 + \left( l_4 \sin \theta_4 - l_2 \sin \theta_2 \right)^2 \\
 &= l_3^2
 \end{aligned}$$

$$\begin{aligned}
 l_1^2 + l_4^2 + l_2^2 + 2l_1l_4 \cos \theta_4 - 2l_1l_2 \cos \theta_2 \\
 - 2l_2l_4 \cos \theta_2 \cos \theta_4 \\
 - 2l_2l_4 \sin \theta_2 \sin \theta_4 &= l_3^2
 \end{aligned}$$

$$\begin{aligned}
 2l_1l_4 \cos \theta_4 - 2l_2l_4 \cos \theta_2 \cos \theta_4 - 2l_2l_4 \sin \theta_2 \sin \theta_4 \\
 = l_3^2 - l_1^2 - l_4^2 - l_2^2 + 2l_1l_2 \cos \theta_2
 \end{aligned}$$

$$\cos \theta_4 \left( 2l_1l_4 - 2l_2l_4 \cos \theta_2 \right) - 2l_2l_4 \sin \theta_2 \sin \theta_4$$

$$\frac{1-x^2}{1+x^2} \qquad \frac{2x}{1+x^2}$$

$$\begin{aligned}
 A &= l_3^2 - l_1^2 - l_4^2 - l_2^2 + 2l_1l_2 \cos \theta_2 + 2l_1l_4 - 2l_2l_4 \cos \theta_2 \\
 B &= +4l_2l_4 \sin \theta_2 \\
 C &= l_3^2 - l_1^2 - l_4^2 - l_2^2 + 2l_1l_2 \cos \theta_2 - 2l_1l_4 + 2l_2l_4 \cos \theta_2
 \end{aligned}$$

Figure 26: FBM Solution



## **CHAPTER 6: DESIGNING BASE**

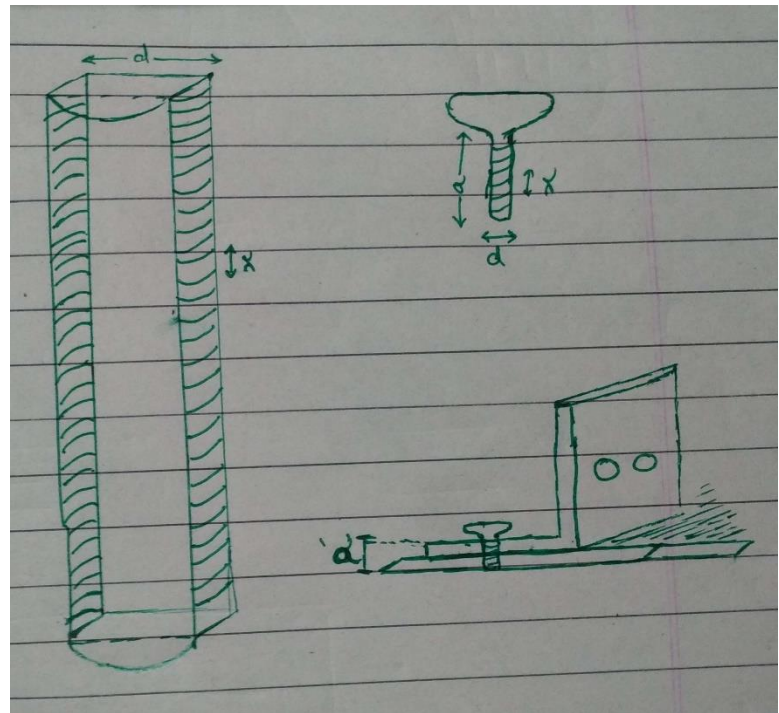
This part of the project was to replicate an already designed base for Stewart platform in which the distance between attachment points can be increased according to the requirement, enabling different type of orientation. This task was to learn the basic of computer aided design and modelling (CAD) using SolidWorks. Major task was to suggest modification to attach the servos to this newly designed base and then changing base lengths with ease.

**Kinematic Singularity:** A kinematic singularity is a point within the robot's workspace where the robot's Jacobian matrix loses rank. The Jacobian is the matrix relating joint velocities to end effector velocities.

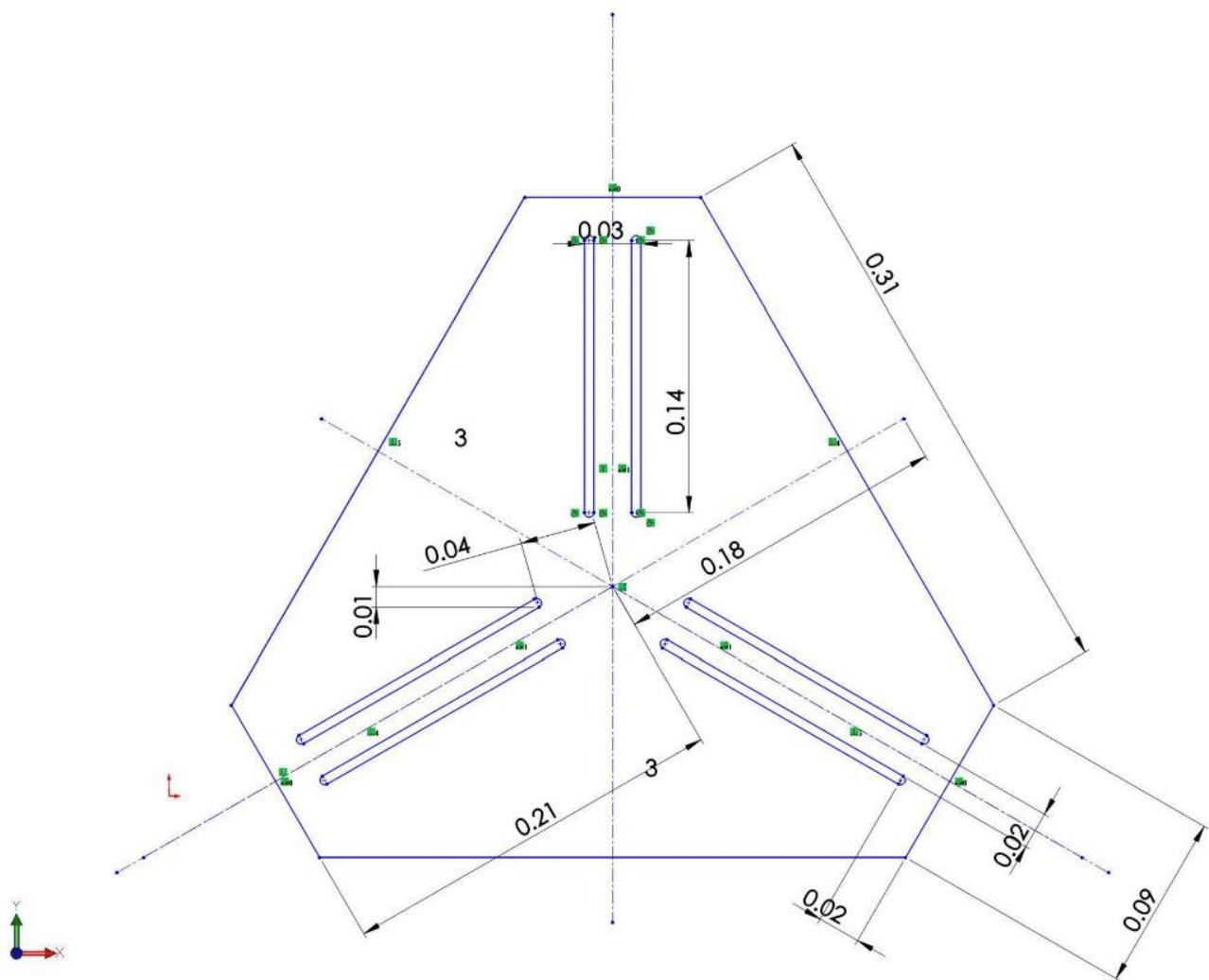
Kinematic singularity is a point in the workspace where the robot loses its ability to move the end effector in some direction no matter how it moves its joints.

### **Design:**

The requirement was that we need to change the distance between joints. So we created two slots of equal length which are equidistant from the centerline. We have a connector over it which has a right angle in it to connect Servo's, so the screw just fits in through the hole in the connector and then goes through the slots in the base and screws length is just sufficient to fit in through the connector and the base.



*Figure 27(a): Modified Base Design*



*Figure 27(b): Modified Base Design*

## **References:**

1. <https://en.wikipedia.org/wiki/Portal:Technology>
2. <http://in.mathworks.com/matlabcentral/answers/contributors>
3. <https://forum.solidworks.com/welcome>
4. [http://ocw.metu.edu.tr/pluginfile.php/3961/mod\\_resource/content/2/ch3/3-7.htm](http://ocw.metu.edu.tr/pluginfile.php/3961/mod_resource/content/2/ch3/3-7.htm)
5. [https://en.wikipedia.org/wiki/Printed\\_circuit\\_board](https://en.wikipedia.org/wiki/Printed_circuit_board)
6. <https://www.fairchildsemi.com/products/power-management/voltage-regulators/positive-voltage-linear-regulators/LM7805.html>
7. [https://en.wikipedia.org/wiki/Lithium\\_polymer\\_battery](https://en.wikipedia.org/wiki/Lithium_polymer_battery)
8. <http://mechdesigner.support/md-kinematics-grashoff-criterion.htm>
9. <http://www.roboanalyzer.com/tutorials.html>
10. <http://www.arduino.cc/en/Reference/Servo>
11. <http://www.roboanalyzer.com/about.html>
12. [https://en.wikipedia.org/wiki/Degrees\\_of\\_freedom\\_%28mechanics%29](https://en.wikipedia.org/wiki/Degrees_of_freedom_%28mechanics%29)

# APPENDIX A: ANIMATION CODES OF STEWART (MATLAB)

Matlab code for animation of Stewart Platform on a horizontal/ vertical circular trajectory (center of top plate).

```
clear all; close all; clc;
d=0.020;c=0.148;d1=0.020;c1=0.154;h=d1;
A=[];
B=[];
C=[];

for j=0:4:360

    % for horizontal circle
    r=0.04;
    xCircle= [r*cos(j*3.14159/180)];
    yCircle= [r*sin(j*3.14159/180)];

    % for vertical circle
    r=0.02;
    xCircle=r*cos((270+j)*3.14159/180);
    yCircle=0;
    zCircle=r*sin((270+j)*3.14159/180)+0.04;

    %top center co-ordinate in the frame of base center
    xCTop=xCircle;
    yCTop=yCircle;
    zCTop=[h]; %height

    %co-ordinates of the base plate in its own frame
    xBase=[-c1/2, -c1/2-d/2, -d/2, d/2, c1/2+d/2, c1/2];
    yBase=[-sqrt(3)*(c1+2*d)/6, -sqrt(3)*(c1-d)/6, sqrt(3)*(2*c1+d)/6,
    sqrt(3)*(2*c1+d)/6, -sqrt(3)*(c1-d)/6, -sqrt(3)*(c1+2*d)/6];
    zBase=[0,0,0,0,0,0];
    plate1=[xBase;yBase;zBase];

    %co-ordinates of the top plate in its own frame
    xTopFrame=[-d/2, -c/2-d/2, -c/2, +c/2, (c+d)/2, d/2];
    yTopFrame=[-sqrt(3)*(2*c+d)/6, sqrt(3)*(c-d)/6, sqrt(3)*(c+2*d)/6,
    sqrt(3)*(c+2*d)/6, sqrt(3)*(c-d)/6, -sqrt(3)*(2*c+d)/6];
    zTopFrame=[0,0,0,0,0,0];

    %co-ordinates of the top plate in frame of the bottom plates center
    xTop=xTopFrame+xCTop;
    yTop=yTopFrame+yCTop;
    zTop=zTopFrame+zCTop;

    plate2=[xTop;yTop;zTop];

    fill3(xBase,yBase,zBase,[1 0.6 0.78]); hold on;
    grid on; axis([-0.15 0.15 -0.2 0.2 0 0.025]);
```

```

fill3(xTop,yTop,zTop,'c'); hold on;

for i=1:6
    plot3([xBase(i),xTop(i)], [yBase(i),yTop(i)], [zBase(i),zTop(i)], ...
        'MarkerFaceColor',[1 0.694117665290833 0.39215686917305], ...
        'MarkerEdgeColor',[0.749019622802734 0 0.749019622802734], ...
        'Marker','o', ...
        'LineWidth',3, ...
        'Color',[0 1 0]);
    hold on;

xlabel('X-axis');
ylabel('Y-axis');
zlabel('Z-axis');
end
A=[A;xCTop];
B=[B;yCTop];
C=[C;zCTop];
plot3(A,B,C,'r');
hold off
pause(0.025);
end

```

Matlab code for animation of Stewart Platform on a Eight shaped trajectory (center of top plate).

```

clear all; close all; clc;
d=0.020;c=0.148;d1=0.020;c1=0.154;h=d1;
A=[];
B=[];
C=[];
% this loop makes the upper circle
for j=0:4:180

    r=0.04;
    xCircle=r*cos(j*3.14159/180)*sin(j*3.14159/180);
    yCircle=r*sin(j*3.14159/180)*sin(j*3.14159/180);
    %top center co-ordinate in the frame of base center
    xCTop=xCircle;
    yCTop=yCircle;
    zCTop=[h]; %height

    %co-ordinates of the base plate in its own frame
    xBase=[-c1/2, -c1/2-d/2, -d/2, d/2, c1/2+d/2, c1/2];
    yBase=[-sqrt(3)*(c1+2*d)/6, -sqrt(3)*(c1-d)/6, sqrt(3)*(2*c1+d)/6,
    sqrt(3)*(2*c1+d)/6, -sqrt(3)*(c1-d)/6, -sqrt(3)*(c1+2*d)/6];
    zBase=[0,0,0,0,0,0];

    %co-ordinates of the top plate in its own frame
    xTopFrame=[-d/2, -c/2-d/2, -c/2, +c/2, (c+d)/2, d/2];
    yTopFrame=[-sqrt(3)*(2*c+d)/6, sqrt(3)*(c-d)/6, sqrt(3)*(c+2*d)/6,
    sqrt(3)*(c+2*d)/6, sqrt(3)*(c-d)/6, -sqrt(3)*(2*c+d)/6];
    zTopFrame=[0,0,0,0,0,0];

```

```

%co-ordinates of the top plate in frame of the bottom plates center
xTop=xTopFrame+xCTop;
yTop=yTopFrame+yCTop;
zTop=zTopFrame+zCTop;

fill3(xBase,yBase,zBase,[1 0.6 0.78]); hold on;
grid on; axis([-0.15 0.15 -0.2 0.2 0 0.025]);
fill3(xTop,yTop,zTop,'c'); hold on;

for i=1:6
    plot3([xBase(i),xTop(i)], [yBase(i),yTop(i)], [zBase(i),zTop(i)],...
        'MarkerFaceColor',[1 0.694117665290833 0.39215686917305],...
        'MarkerEdgeColor',[0.749019622802734 0 0.749019622802734],...
        'Marker','o',...
        'LineWidth',3,...
        'Color',[0 1 0]);
    hold on;

    xlabel('X-axis');
    ylabel('Y-axis');
    zlabel('Z-axis');
end

A=[A;xCTop];
B=[B;yCTop];
C=[C;zCTop];
plot3(A,B,C,'y');
hold off;
pause(0.025);
end
% this loop makes the lower circle
for j=0:4:180

    r=0.04;
    xCircle=-r*cos(j*3.14159/180)*sin(j*3.14159/180);
    yCircle=-r*sin(j*3.14159/180)*sin(j*3.14159/180);
    %top center co-ordinate in the frame of base center
    xCTop=xCircle;
    yCTop=yCircle;
    zCTop=[h]; %height

    %co-ordinates of the base plate in its own frame
    xBase=[-c1/2, -c1/2-d/2, -d/2, d/2, c1/2+d/2, c1/2];
    yBase=[-sqrt(3)*(c1+2*d)/6, -sqrt(3)*(c1-d)/6, sqrt(3)*(2*c1+d)/6,
sqrt(3)*(2*c1+d)/6, -sqrt(3)*(c1-d)/6, -sqrt(3)*(c1+2*d)/6];
    zBase=[0,0,0,0,0,0];

    %co-ordinates of the top plate in its own frame
    xTopFrame=[-d/2, -c/2-d/2, -c/2, +c/2, (c+d)/2, d/2];
    yTopFrame=[-sqrt(3)*(2*c+d)/6, sqrt(3)*(c-d)/6, sqrt(3)*(c+2*d)/6,
sqrt(3)*(c+2*d)/6, sqrt(3)*(c-d)/6, -sqrt(3)*(2*c+d)/6];
    zTopFrame=[0,0,0,0,0,0];

    %co-ordinates of the top plate in frame of the bottom plates center
    xTop=xTopFrame+xCTop;

```

```

yTop=yTopFrame+yCTop;
zTop=zTopFrame+zCTop;

fill3(xBase,yBase,zBase,[1 0.6 0.78]); hold on;
grid on; axis([-0.15 0.15 -0.2 0.2 0 0.025]);
fill3(xTop,yTop,zTop,'c'); hold on;

for i=1:6
    plot3([xBase(i),xTop(i)], [yBase(i),yTop(i)], [zBase(i),zTop(i)],...
        'MarkerFaceColor',[1 0.694117665290833 0.39215686917305],...
        'MarkerEdgeColor',[0.749019622802734 0 0.749019622802734],...
        'Marker','o',...
        'LineWidth',3,...
        'Color',[0 1 0]);
    hold on;

    xlabel('X-axis');
    ylabel('Y-axis');
    zlabel('Z-axis');
end

A=[A;xCTop];
B=[B;yCTop];
C=[C;zCTop];
plot3(A,B,C,'y');
hold off;
pause(0.025);
end

```

```
float low_mpoints[3][6], top_mpoints[3][6], diff_low_up[3][6], dist_low_up[6];
```



```

/*
    s_width_top: width of the shorter side on the upper base in meters
    l_width_top: width of the longer side on the upper base in meters
    s_width_bottom: width of the shorter side on the lower base in meters
    l_width_bottom: width of the longer side on the lower base in meters
    i_sep: initial separation between the centers of the lower base and the upper base

*/

float s_width_top=0.020 , l_width_top=0.148 , i_sep=0.020, s_width_bottom=0.020, l_width_bottom=0.154;

//array storing the co-ordinates of the center points of the top base [x,y,z]
float top_center[3]={0.0,0.0,0.13};

// I don't know a shit about what this is :P
float a1[3]={radians(0),radians(0),radians(0)};

// contains the length of the legs returned by in inverseKinematics function
float servo_pos[6];

void setup(){
    Serial.begin(9600);

    /*
        Assigning co-ordinates of the lower/upper mounting points of servos
        origin at the Geometrical center of the base plate
    */

    low_mpoints[0][0] = -l_width_bottom/2;
    low_mpoints[1][0] = -sqrt(3)*(l_width_bottom+2*s_width_bottom)/6;
    low_mpoints[2][0] = 0;

    low_mpoints[0][1] = -l_width_bottom/2-s_width_bottom/2;

```

$\text{low\_mpoints}[1][1] = -\sqrt{3} * (\text{l\_width\_bottom} - \text{s\_width\_bottom}) / 6;$

$\text{low\_mpoints}[2][1] = 0;$

$\text{low\_mpoints}[0][2] = -\text{s\_width\_bottom} / 2;$

$\text{low\_mpoints}[1][2] = \sqrt{3} * (2 * \text{l\_width\_bottom} + \text{s\_width\_bottom}) / 6;$

$\text{low\_mpoints}[2][2] = 0;$

$\text{low\_mpoints}[0][3] = \text{s\_width\_bottom} / 2;$

$\text{low\_mpoints}[1][3] = \sqrt{3} * (2 * \text{l\_width\_bottom} + \text{s\_width\_bottom}) / 6;$

$\text{low\_mpoints}[2][3] = 0;$

$\text{low\_mpoints}[0][4] = \text{l\_width\_bottom} / 2 + \text{s\_width\_bottom} / 2;$

$\text{low\_mpoints}[1][4] = -\sqrt{3} * (\text{l\_width\_bottom} - \text{s\_width\_bottom}) / 6;$

$\text{low\_mpoints}[2][4] = 0;$

$\text{low\_mpoints}[0][5] = \text{l\_width\_bottom} / 2;$

$\text{low\_mpoints}[1][5] = -\sqrt{3} * (\text{l\_width\_bottom} + 2 * \text{s\_width\_bottom}) / 6;$

$\text{low\_mpoints}[2][5] = 0;$

$\text{top\_mpoints}[0][0] = -\text{s\_width\_top} / 2;$

$\text{top\_mpoints}[1][0] = -\sqrt{3} * (2 * \text{l\_width\_top} + \text{s\_width\_top}) / 6;$

$\text{top\_mpoints}[2][0] = 0;$

$\text{top\_mpoints}[0][1] = -\text{l\_width\_top} / 2 - \text{s\_width\_top} / 2;$

$\text{top\_mpoints}[1][1] = \sqrt{3} * (\text{l\_width\_top} - \text{s\_width\_top}) / 6;$

$\text{top\_mpoints}[2][1] = 0;$

$\text{top\_mpoints}[0][2] = -\text{l\_width\_top} / 2;$

```
top_mpoints[1][2] = sqrt(3)*(l_width_top+2*s_width_top)/6;  
top_mpoints[2][2] = 0;
```

```
top_mpoints[0][3] = l_width_top/2;  
top_mpoints[1][3] = sqrt(3)*(l_width_top+2*s_width_top)/6;  
top_mpoints[2][3] = 0;
```

```
top_mpoints[0][4] = l_width_top/2+s_width_top/2;  
top_mpoints[1][4] = sqrt(3)*(l_width_top-s_width_top)/6;  
top_mpoints[2][4] = 0;
```

```
top_mpoints[0][5] = s_width_top/2;  
top_mpoints[1][5] = -sqrt(3)*(2*l_width_top+s_width_top)/6;  
top_mpoints[2][5] = 0;
```

```
/* this loop attaches our physical servos with the specific digital pins in arduino uno board
```

```
   attach ref_servo[i] to servo_pin_num[i]  servo 1 -> pin 6
```

```
   servo 1 -> pin 6
```

```
   servo 2 -> pin 7
```

```
   servo 3 -> pin 8
```

```
   servo 4 -> pin 9
```

```
   servo 5 -> pin 10
```

```
   servo 6 -> pin 11
```

```
*/
```

```
for(int i=0;i<6;i++){
```

```
   ref_servo[i].attach(servo_pin_num[i]);
```

```
}
```

```
setBack();                                     //calling function setBack to get it to the no extended state
```

```
}
```

```
void loop(){
```

```
   /*
```

this code is for circular trajectory of center

```
*/  
/*  
float r= 0.04 ;                //radius of the circle  
  
for(int i=0;i<20;i++){          //takes the center at (r,0,0.12)  
    top_center[0]=r*i/20;  
  
    inverseKinematics();  
  
    for(int i = 0; i < 6; i++){  
        ref_servo[i].writeMicroseconds(servo_pos[i]);  
    }  
    delay(200);  
}  
*/  
  
for(int i=0;i<360;i+=4){  
  
    top_center[0]=0;//r*cos(i*3.14159/180);        //x-coordinate of the center  
    top_center[1]=0;//r*sin(i*3.14159/180);        //y-coordinate of the center  
    top_center[2]=0.13;                //z-coordinate of the center  
    a1[0]=0;  
    a1[1]=0;  
    a1[2]=0;  
  
    inverseKinematics();  
    for(int i = 0; i < 6; i++){  
        ref_servo[i].writeMicroseconds(servo_pos[i]);  
    }  
    delay(200);  
}
```

```

        setBack();
    }
/*
This function takes all the servo's back to the initial stage
i.e a state in which none of the arms is extended
*/
void setBack(){
    for(int i=0;i<6;i++){
        // This loop set's servo's back to not extended position
        ref_servo[i].writeMicroseconds(servo_zero[i]);
    }
    delay(4000);
    //Gives servos time to get back to initial stage
}

// Not written by me, just modified it to work with my code
void inverseKinematics()
{
    float rx1[3][3], ry1[3][3], rz1[3][3];
    //Rotation Matrices
    rx1[0][0]=1;
    rx1[0][1]=0;
    rx1[0][2]=0;
    rx1[1][0]=0;
    rx1[1][1]= cos(a1[0]);
    rx1[1][2]= -sin(a1[0]);
    rx1[2][0]=0;
    rx1[2][1]= sin(a1[0]);
    rx1[2][2]= cos(a1[0]);

    ry1[0][0]=cos(a1[1]);
    ry1[0][1]= 0;
    ry1[0][2]=sin(a1[1]);
    ry1[1][0]=0;

```

```
ry1[1][1]=1;
ry1[1][2]=0;
ry1[2][0]= -sin(a1[1]);
ry1[2][1]=0;
ry1[2][2]= cos(a1[1]);
```

```
rz1[0][0]=cos(a1[2]);
rz1[0][1]=-sin(a1[2]);
rz1[0][2]=0;
rz1[1][0]=sin(a1[2]);
rz1[1][1]= cos(a1[2]);
rz1[1][2]=0;
rz1[2][0]=0;
rz1[2][1]=0;
rz1[2][2]=1;
```

```
float r1[3][3],ri1[3][3];
int i=0;
int j=0;
int k=0;
for(i=0;i<3;i++)
{for (j=0;j<3;j++)
ri1[i][j]=0;
}
for(i=0;i<3;i++)
{for (j=0;j<3;j++)
{for(k=0;k<3;k++)
{ ri1[i][j]+=ry1[i][k]*rx1[k][j];
}}}
i=0;
j=0;
k=0;
```

```

for(i=0;i<3;i++)
{
    for (j=0;j<3;j++)
    r1[i][j]=0;
}

for(i=0;i<3;i++)
{
    for (j=0;j<3;j++)
    {
        for(k=0;k<3;k++)
        {
            r1[i][j]+=rz1[i][k]*ri1[k][j];
        }
    }
}

/*

Serial.print(r1[0][0]);

Serial.print(r1[0][1]);

Serial.print(r1[0][2]);

Serial.print(r1[1][0]);

Serial.print(r1[1][1]);

Serial.print(r1[1][2]);

Serial.print(r1[2][0]);

Serial.print(r1[2][1]);

Serial.print(r1[2][2]);

*/

//reference frame

float ppf[3][6];

i=0;

j=0;

k=0;

for(i=0;i<3;i++)
{
    for (j=0;j<6;j++)
    ppf[i][j]=0;
}

for(i=0;i<3;i++)
{
    for (j=0;j<6;j++)
    {
        for(k=0;k<3;k++)

```

```

{ppf[i][j]+=r1[i][k]*top_mpoints[k][j];
}}
}
for(i=0;i<3;i++)
{for (j=0;j<6;j++)
ppf[i][j]=ppf[i][j]+top_center[i];
// Serial.print(top_center[i]);
// Serial.print(" * ");
}
for(int i = 0; i < 6; i++)
{
diff_low_up[0][i] = low_mpoints[0][i] - ppf[0][i];
diff_low_up[1][i] = low_mpoints[1][i] - ppf[1][i];
diff_low_up[2][i] = low_mpoints[2][i] - ppf[2][i];

dist_low_up[i] = sqrt(diff_low_up[0][i]*diff_low_up[0][i] + diff_low_up[1][i]*diff_low_up[1][i] +
diff_low_up[2][i]*diff_low_up[2][i]) ;

servo_pos[i] = 1000 + ((dist_low_up[i]-0.130)/(0.184-0.130)) * ( 2000 - 1000 ) ;
Serial.print(servo_pos[i]);
Serial.print(" * ");
}
Serial.print("\n");
}

```



## Void loop's code for eight shaped trajectory:

```
void loop(){  
  /*  
  this code is to make a "8" type of trajectory of center  
  */  
  
  float r1= 0.025 ;           //radius of the upper circle  
  float r2= 0.025;           //radius of the lower circle  
  
  for(int th=0;th<360;th+=4){           //makes the upper circle in 90 steps  
    top_center[0]=r1/2*sin(th*3.14159*2/180);  
    top_center[1]=r1*sin(th*3.14159/180)*sin(th*3.14159/180);  
    inverseKinematics();  
    for(int i = 0; i < 6; i++){  
      ref_servo[i].writeMicroseconds(servo_pos[i]);  
    }  
    delay(200);  
  }  
  
  for(int th=0;th<360;th+=4){           //makes the upper circle in 90 steps  
    top_center[0]=-r1/2*sin(th*3.14159*2/180);  
    top_center[1]=-r1*sin(th*3.14159/180)*sin(th*3.14159/180);  
    inverseKinematics();  
  }  
  for(int i = 0; i < 6; i++){  
    ref_servo[i].writeMicroseconds(servo_pos[i]);  
  }  
  delay(200);  
}  
  
  setBack(); }
```

## Void loop's code for vertical circular trajectory:

```
void loop(){  
    /*  
    this code is for circular trajectory of center  
    */  
    float r= 0.02 ;           //radius of the circle  
    for(int i=0;i<360;i+=4){  
        top_center[0]= r*cos((270+i)*3.14159/180) ;           //x-coordinate of the center  
        top_center[1]= 0 ;           //y-coordinate of the center  
        top_center[2]= r*sin((270+i)*3.14159/180)+0.04 ;           //z-coordinate of the center  
        a1[0]=0;  
        a1[1]=0;  
        a1[2]=0;  
        inverseKinematics();  
        for(int i = 0; i < 6; i++){  
            ref_servo[i].writeMicroseconds(servo_pos[i]);  
        }  
        delay(200);  
    }  
    setBack();  
}
```

## APPENDIX C: FOUR BAR ANIMATION CODE

For this part we have some function files that you need to create and save them all in one directory and then run (FB\_run\_file). It will ask you for inputs and omega provide that and animation will begin.

getMin.m

```
function [ minIndex , minNum ] = getMin( A )
    minNum=A(1); minIndex=0;
    for i=1:4
        if le(A(i),minNum)
            minNum=A(i);
            minIndex=i;
        end
    end
end
```

getMax.m

```
function [ maxIndex , maxNum ] = getMax( A )
    maxNum=A(1); maxIndex=0;
    for i=1:4
        if ge( A(i),maxNum)
            maxNum=A(i);
            maxIndex=i;
        end
    end
end
```

checkType.m

```
function [Type] = checkType( minIndex , maxIndex , A )
    Atotal=A(1)+A(2)+A(3)+A(4);
    if 2*(A(minIndex) + A(maxIndex)) < Atotal
        disp('shortest+longest < sum of other two')
        if minIndex==1
            disp('Crank-Crank')
            Type=1;
            %disp('Type-1 Enter "1" in FB_Animation Function to get
desired animation')
        end
        if minIndex==3
            disp('Rocker-Rocker')
            Type=2;
            %disp('Type-2 Enter "2" in FB_Animation Function to get
desired animation')
        end
        if minIndex==2 || minIndex==4
            disp('Crank-Rocker')
            Type=3;
        end
    end
end
```

```

        %disp('Type-3 Enter "3" in FB_Animation Function to get
desired animation')
    end
end
if 2*(A(minIndex) + A(maxIndex)) == Atotal
    if A(1)==A(3) && A(2)==A(4)
        disp('Parallel-Crank Four-Bar Linkage');
        Type=4;
        %disp('Type-4 Enter "4" in FB_Animation Function to get
desired animation')
    end
end
if 2*(A(minIndex) + A(maxIndex)) > Atotal
    disp('Shortest + Longest > Sum of other two')
    Type=5;
    %disp('Type-5 Enter "5" in FB_Animation Function to get desired
animation')
end
end
end

```

## FB\_Animation.m

```

%this function work well for double crank and crank rocker
function FB_Animation( A, Omega, Type)
% A is a vector of 4th dimension
if Type==1 || Type==3

    figure1 = figure('Color',[1 1 1]);
    set(gcf,'color',[1 1 1])
    axis([-50 50 -30 30]);
    axis equal

    for t=1:0.05:10
        th2=t*Omega;

        a=A(3)*A(3)-A(1)*A(1)-A(4)*A(4)-
A(2)*A(2)+2*A(1)*A(2)*cos(th2)+2*A(1)*A(4)-2*A(4)*A(2)*cos(th2);
        b=4*A(2)*A(4)*sin(th2);
        c=A(3)*A(3)-A(1)*A(1)-A(4)*A(4)-A(2)*A(2)+2*A(1)*A(2)*cos(th2)-
2*A(1)*A(4)+2*A(4)*A(2)*cos(th2);
        th4=2*atan((-b+sqrt(b*b-4*a*c))/(2*a));

        x0=10 ; y0=0;
        x1=10+A(2)*cos(th2); y1=A(2)*sin(th2);
        x2=10+A(1)+A(4)*cos(th4); y2=A(4)*sin(th4);
        x3=10+A(1); y3=0;
        cla
        hold on
        plot([x0 x1],[y0 y1],'LineWidth',2,'Color',[0 0.749019622802734
0.749019622802734]);
        plot([x1 x2],[y1 y2], 'MarkerFaceColor',[0 0.749019622802734
0.749019622802734],...
'MarkerEdgeColor',[0.0784313753247261 0.168627455830574
0.549019634723663],...

```

```

        'MarkerSize',5,...
        'Marker','o',...
        'LineWidth',2,...
        'Color',[0.0588235296308994 0.874509811401367
0.470588237047195]);
        plot([x2 x3],[y2 y3],'LineWidth',2,...
        'Color',[0.0431372560560703 0.517647087574005
0.780392169952393]);
        plot([x0 x3],[y0 y3],...
        'MarkerFaceColor',[0.24705882370472 0.24705882370472
0.24705882370472],...
        'MarkerEdgeColor',[0.24705882370472 0.24705882370472
0.24705882370472],...
        'Marker','v',...
        'LineWidth',2,...
        'Color',[0.87058824300766 0.490196079015732 0]);
        pause(0.05);
        hold off
    end
end

if Type==4
    figure1 = figure('Color',[1 1 1]);
    set(gcf,'color',[1 1 1])
    axis([-5 15 -10 10]);
    axis equal

    for t=1:0.05:10
        th2=t*Omega;
        th4=th2;

        x0=10 ; y0=0;
        x1=10+A(2)*cos(th2); y1=A(2)*sin(th2);
        x2=10+A(1)+A(4)*cos(th4); y2=A(4)*sin(th4);
        x3=10+A(1); y3=0;
        cla
        hold on
        plot([x0 x1],[y0 y1],'LineWidth',2,'Color',[0 0.749019622802734
0.749019622802734]);
        plot([x1 x2],[y1 y2], 'MarkerFaceColor',[0 0.749019622802734
0.749019622802734],...
        'MarkerEdgeColor',[0.0784313753247261 0.168627455830574
0.549019634723663],...
        'MarkerSize',5,...
        'Marker','o',...
        'LineWidth',2,...
        'Color',[0.0588235296308994 0.874509811401367
0.470588237047195]);
        plot([x2 x3],[y2 y3],'LineWidth',2,...
        'Color',[0.0431372560560703 0.517647087574005
0.780392169952393]);
        plot([x0 x3],[y0 y3],...
        'MarkerFaceColor',[0.24705882370472 0.24705882370472
0.24705882370472],...

```

```

        'MarkerEdgeColor',[0.24705882370472 0.24705882370472
0.24705882370472],...
        'Marker','v',...
        'LineWidth',2,...
        'Color',[0.87058824300766 0.490196079015732 0]);
    pause(0.05);
    hold off
end
end

if Type==5
    disp('I have not studied this case')
end

if Type==2
    figure1 = figure('Color',[1 1 1]);
    set(gcf,'color',[1 1 1])
    axis([-5 15 -10 10]);
    axis equal

    H=[];I=[];

    for t=0:0.05:10
        th2=t*Omega;

        a=A(3)*A(3)-A(1)*A(1)-A(4)*A(4)-
A(2)*A(2)+2*A(1)*A(2)*cos(th2)+2*A(1)*A(4)-2*A(4)*A(2)*cos(th2);
        b=4*A(2)*A(4)*sin(th2);
        c=A(3)*A(3)-A(1)*A(1)-A(4)*A(4)-A(2)*A(2)+2*A(1)*A(2)*cos(th2)-
2*A(1)*A(4)+2*A(4)*A(2)*cos(th2);
        th4=2*atan((-b+sqrt(b*b-4*a*c))/(2*a));

        x1=10+A(2)*cos(th2); y1=A(2)*sin(th2);
        x2=10+A(1)+A(4)*cos(th4); y2=A(4)*sin(th4);

        if A(3)== sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))
            H=[H;th2];
            I=[I;th4];
        end
    end

    for th2=H(1,1):H(end,1)

        a=A(3)*A(3)-A(1)*A(1)-A(4)*A(4)-
A(2)*A(2)+2*A(1)*A(2)*cos(th2)+2*A(1)*A(4)-2*A(4)*A(2)*cos(th2);
        b=4*A(2)*A(4)*sin(th2);
        c=A(3)*A(3)-A(1)*A(1)-A(4)*A(4)-A(2)*A(2)+2*A(1)*A(2)*cos(th2)-
2*A(1)*A(4)+2*A(4)*A(2)*cos(th2);
        th4=2*atan((-b+sqrt(b*b-4*a*c))/(2*a));

        x0=10 ; y0=0;

```

```

x1=10+A(2)*cos(th2); y1=A(2)*sin(th2);
x2=10+A(1)+A(4)*cos(th4); y2=A(4)*sin(th4);
x3=10+A(1); y3=0;
cla
hold on
plot([x0 x1],[y0 y1], 'LineWidth',2, 'Color',[0 0.749019622802734
0.749019622802734]);
plot([x1 x2],[y1 y2], 'MarkerFaceColor',[0 0.749019622802734
0.749019622802734],...
'MarkerEdgeColor',[0.0784313753247261 0.168627455830574
0.549019634723663],...
'MarkerSize',5,...
'Marker','o',...
'LineWidth',2,...
'Color',[0.0588235296308994 0.874509811401367
0.470588237047195]);
plot([x2 x3],[y2 y3], 'LineWidth',2,...
'Color',[0.0431372560560703 0.517647087574005
0.780392169952393]);
plot([x0 x3],[y0 y3],...
'MarkerFaceColor',[0.24705882370472 0.24705882370472
0.24705882370472],...
'MarkerEdgeColor',[0.24705882370472 0.24705882370472
0.24705882370472],...
'Marker','v',...
'LineWidth',2,...
'Color',[0.87058824300766 0.490196079015732 0]);
pause(0.005);
hold off
end
disp('Work Going On , not completed')
end

```

## FB\_run\_file.m

```

clear all;close all;clc;
prompt='Input the four lengths of Four bar Mechanism as a vector
[fixedLink , InputLink , Coupler , EndEffector]= ' ;
A=input(prompt);
prompt='Input Omega= ' ;
Omega=input(prompt);

[ minIndex , minNum ] = getMin( A );
[ maxIndex , maxNum ] = getMax( A );
[Type] = checkType( minIndex , maxIndex , A );
FB_Animation(A,Omega,Type);

```

## **APPENDIX D: ALL MATERIAL RELATED TO PROJECT**

*This link contains all the videos, animation videos and codes*

<https://drive.google.com/open?id=0B9XUd9XNloWtfkVTWlhCSFFFOHB2azdJMUU4dlc0bFZVR0VqOXVwb0ljaFdvc1hha0h1bms&authuser=0>



