UNIVERSITY OF MARYLAND

ENPM 662 Introduction to robot modelling

Project – 2

Alpha - Pick and Place Mobile Manipulator

by

Varun Lakshmanan

UID: 120169595

Nitish Ravisankar Raveendran

UID: 120385506

Contents

1. Introduction:

Robot Manipulators are devices used to accomplish tasks related to grasping, moving, and manipulating entities in a 3D space. These are used all around the world in numerous industries, such as research, logistics, manufacturing and healthcare. An example of applications of these kind of robot in today's industry is Handling and picking, where Robot manipulators are assigned tasks to transport goods and pick and place objects out of a tote and placing them into a shipping container.

This Project deals with the process of designing a mobile manipulator with a pick and place operation which involves the following tasks:

- Design and assembly of the Mobile Manipulator with a four-wheel differential base.
- Converting the CAD model into a URDF file and exporting it.
- Implementing the URDF file into ROS 2 and spawning the robot in the gazebo.
- Designing the required controllers to operate the robot in the gazebo.
- Calculating the forward and inverse kinematics for the manipulator and validating them.
- Analyzing the robot workspace.
- Analyzing the robot behavior and taking the results.

2. Application:

The aim of the manipulator is to pick and place the object from one place to another as per the user's requirements.

3. Robot Type:

The Type of the robot used here is a mobile serial manipulator consisting of a mobile base with four-wheels driven by differential, which hosts a chain of two links connected with revolute joints. There is an end-effector at the end of the third link, which consists of four pick and place links attached to it.

4. Degrees of Freedom of the Robot:

- The Mobile base consists of 3 degrees of freedom.
- The Manipulator consists of 4 degrees of freedom.

5. CAD Model of the robot:

The Robot is designed from scratch using SolidWorks Modelling Software using Custom Dimensions.

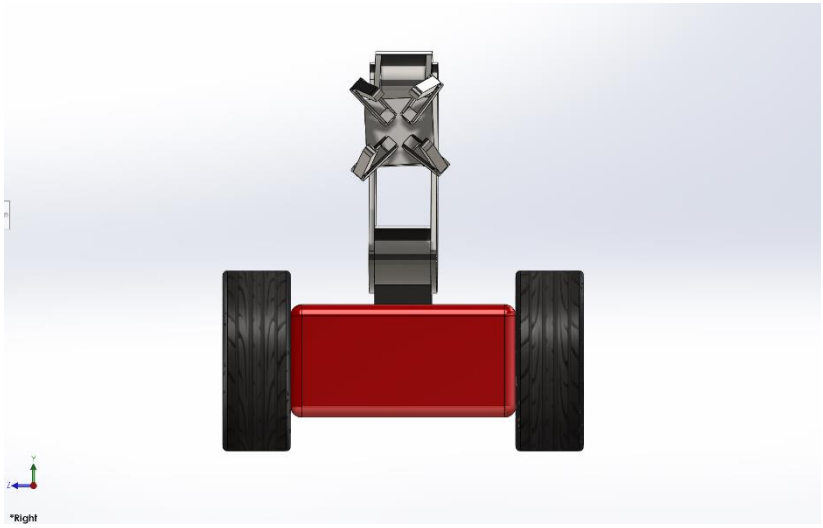

Figure 1: Right View of the Manipulator

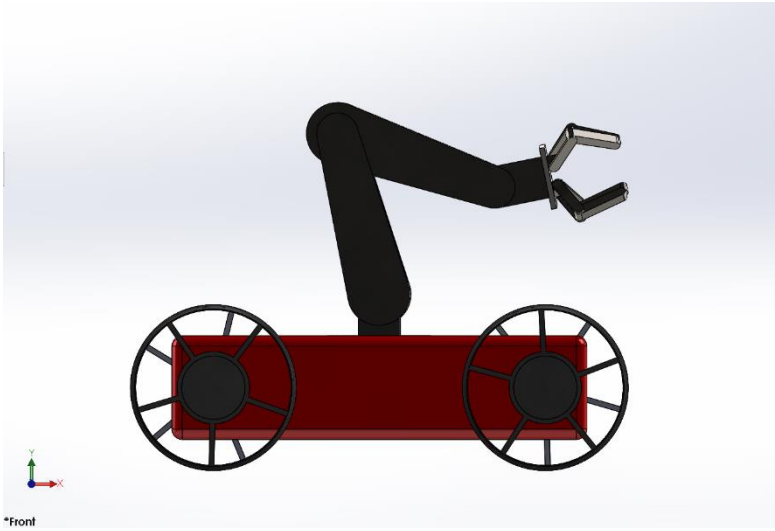

Figure 2: Front View of the Manipulator

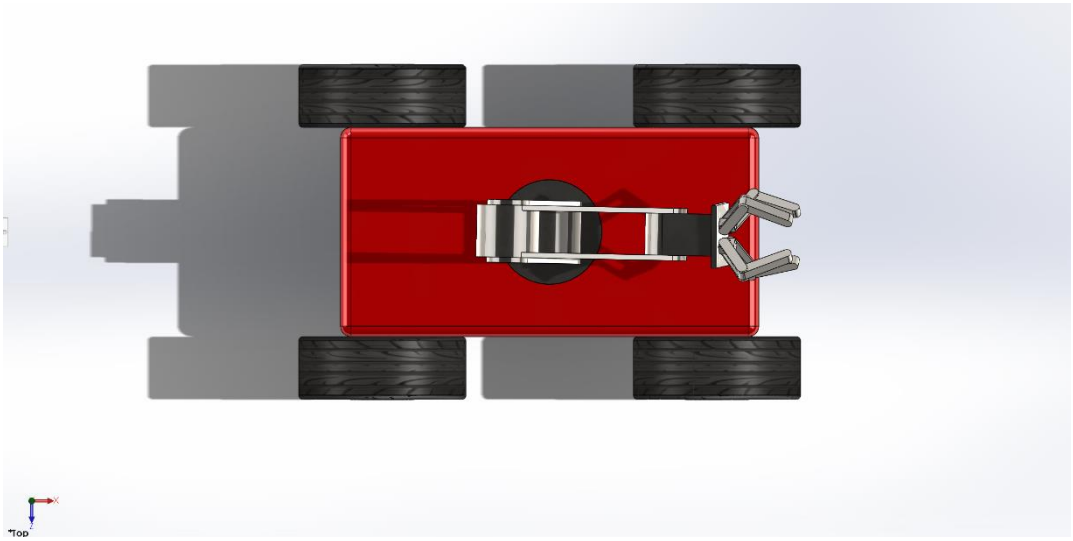

Figure 3: Top View of the Manipulator

The above images show the front, right and top view of the CAD model mobile pick and place manipulator designed by using SolidWorks Software.

6. Dimensions of the robot:

Specifications	Measurement
Base length	**508 mm**
Base Width	**254 mm**
Base Height	**127 mm**
Link 1 length	**76.20 mm**
Link 2 length	**203.20 mm**
Link 3 Length	**203.20 mm**
End-Effector Length	**158.50 mm**
Radius of the Wheel	**203.20 mm**
Wheel thickness	**76.20 mm**

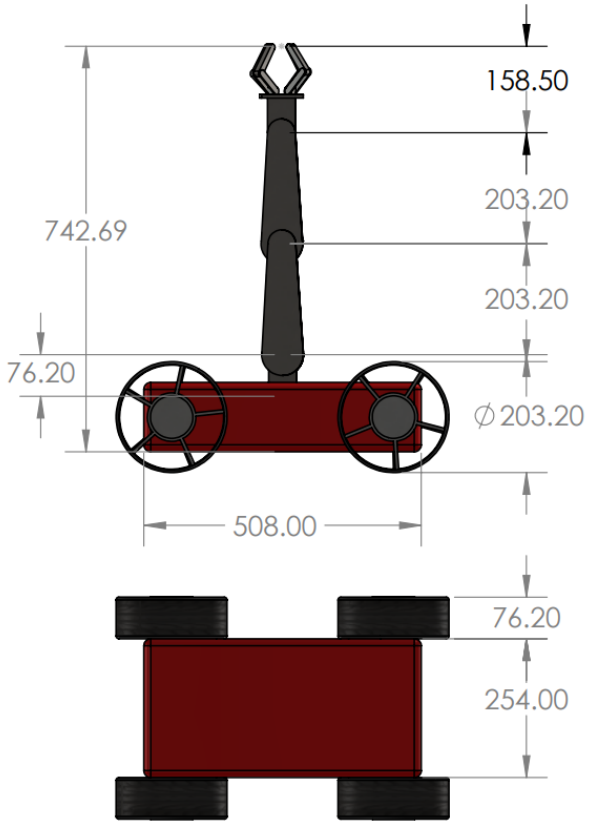

Figure 4: Dimensions of the Manipulator

7. Frame Diagram of the Robot:

The Frame Diagram used to calculate the Denevit-Hartenberg Parameters is given below:

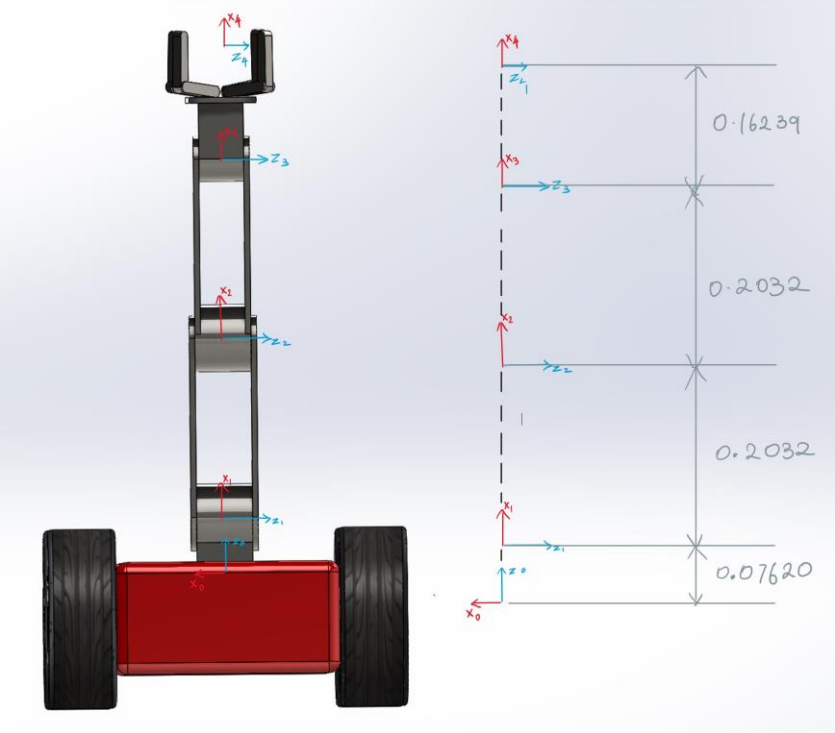

Figure 5: Frame Diagram in (meters)

8. Denevit-Hartenberg Parameters for the robot:

Transforms	Link Length (a) in mm	Angle of Twist (α)	Link offset (d) in mm	Joint Angles (θ)
0 - 1	0	$\frac{\pi}{2}$	76.20	θ_1
1 - 2	203.20	0	0	θ_2
2 - 3	203.20	0	0	θ_3
3 - 4	158.50	0	0	θ_4

9. Forward Kinematics of the robot:

The Forward Kinematics is done to determine the position and orientation of the end effector of the robot using the joint angles provided. For solving the forward kinematics, the coordinate frames of each joint are attached and the relationship among these frames is expressed in terms of homogenous transformation matrices. Here, we use a systematic process called **Denavit-Hartenberg** convention for attaching these frames to the robot. The Position and orientation of the end-effector is reduced to homogenous transformation matrices.

The forward kinematics of the robot is determined using the DH table by the formula:

$$A_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where,

c_{θ_i} - *Cos* θ_i

S_{θ_i} - *Sin* θ_i

θ_i – *Joint angles*

α_i – *Angle of Twist*

a_i – *Link Length*

d_i – *Link Offset*

The transformation matrix for each frame is calculated individually and multiplied to get the final transformation matrix. These calculations are performed on the python using sympy libraries.

The Final Transformation matrix determines the position and orientation of the end effector, which is given below:

```
The symbolic reprentation of the transformation matrix is:
⎡(-sin(θ₂)·sin(θ₃)·cos(θ₁) + cos(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) + (-sin(θ₂)·cos(
⎢
⎢(-sin(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·cos(θ₂)·cos(θ₃))·cos(θ₄) + (-sin(θ₁)·sin(
⎢
⎢                (-sin(θ₂)·sin(θ₃) + cos(θ₂)·cos(θ₃))·sin(θ₄) + (sin(θ₂)·cos(θ
⎢
⎣                                                              0

θ₁)·cos(θ₃) - sin(θ₃)·cos(θ₁)·cos(θ₂))·sin(θ₄)  -(-sin(θ₂)·sin(θ₃)·cos(θ₁) + c

θ₂)·cos(θ₃) - sin(θ₁)·sin(θ₃)·cos(θ₂))·sin(θ₄)  -(-sin(θ₁)·sin(θ₂)·sin(θ₃) + s

₃) + sin(θ₃)·cos(θ₂))·cos(θ₄)                                    (-sin(θ₂)·sin


os(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + (-sin(θ₂)·cos(θ₁)·cos(θ₃) - sin(θ₃)·cos(θ₁)·

in(θ₁)·cos(θ₂)·cos(θ₃))·sin(θ₄) + (-sin(θ₁)·sin(θ₂)·cos(θ₃) - sin(θ₁)·sin(θ₃)·

(θ₃) + cos(θ₂)·cos(θ₃))·cos(θ₄) - (sin(θ₂)·cos(θ₃) + sin(θ₃)·cos(θ₂))·sin(θ₄)

                                0

cos(θ₂))·cos(θ₄)  sin(θ₁)   158.5·(-sin(θ₂)·sin(θ₃)·cos(θ₁) + cos(θ₁)·cos(θ₂)·

cos(θ₂))·cos(θ₄)  -cos(θ₁)  158.5·(-sin(θ₁)·sin(θ₂)·sin(θ₃) + sin(θ₁)·cos(θ₂)·

                   0                                158.5·(-sin(θ₂)·sin(θ₃) +

                   0

cos(θ₃))·cos(θ₄) + 158.5·(-sin(θ₂)·cos(θ₁)·cos(θ₃) - sin(θ₃)·cos(θ₁)·cos(θ₂))·

cos(θ₃))·cos(θ₄) + 158.5·(-sin(θ₁)·sin(θ₂)·cos(θ₃) - sin(θ₁)·sin(θ₃)·cos(θ₂))·

 cos(θ₂)·cos(θ₃))·sin(θ₄) + 158.5·(sin(θ₂)·cos(θ₃) + sin(θ₃)·cos(θ₂))·cos(θ₄)

                                                                1

sin(θ₄) - 203.2·sin(θ₂)·sin(θ₃)·cos(θ₁) + 203.2·cos(θ₁)·cos(θ₂)·cos(θ₃) + 203.

sin(θ₄) - 203.2·sin(θ₁)·sin(θ₂)·sin(θ₃) + 203.2·sin(θ₁)·cos(θ₂)·cos(θ₃) + 203.

+ 203.2·sin(θ₂)·cos(θ₃) + 203.2·sin(θ₂) + 203.2·sin(θ₃)·cos(θ₂) + 76.2


⎡2·cos(θ₁)·cos(θ₂)⎤
⎢                 ⎥
⎣2·sin(θ₁)·cos(θ₂)⎦
```

10.Validation of Forward Kinematics:

A validation of forward kinematics is necessary to make sure that the derived kinematics is reliable. Here, the geometrical validation is done considering four configurations.

The Validation is done using **Peter Corke Toolbox in MATLAB.**

10.1. For Home Configuration:

Here,

$$\theta_1 = \theta_2 = \theta_3 = \theta_4 = 0$$

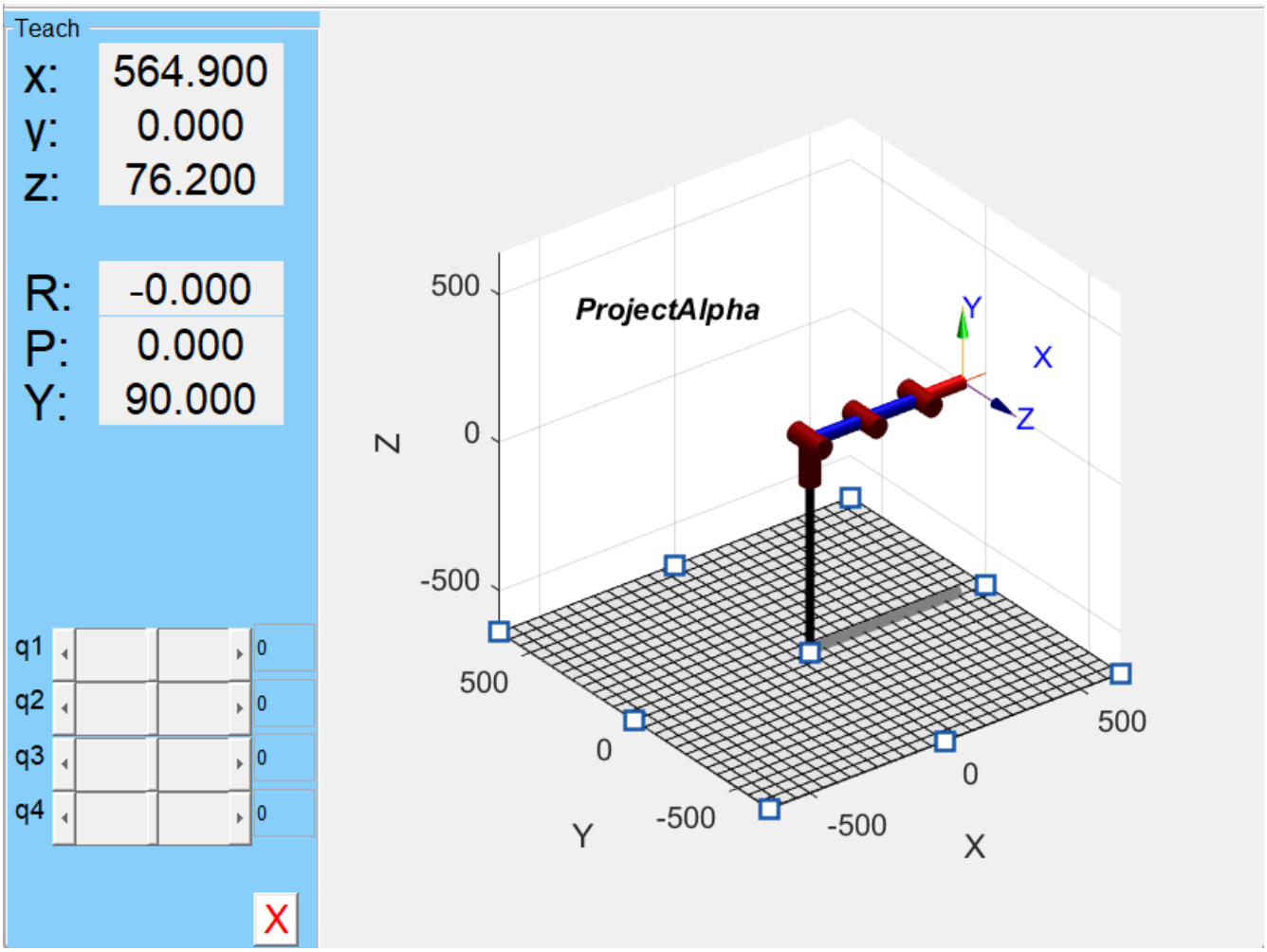

Figure 6: Plot for Home Configuration

10.2. For Configuration 1:

Here,

$$[\theta_1 = \theta_3 = 0],[\theta_2 = \theta_4 = \frac{\pi}{2}]$$

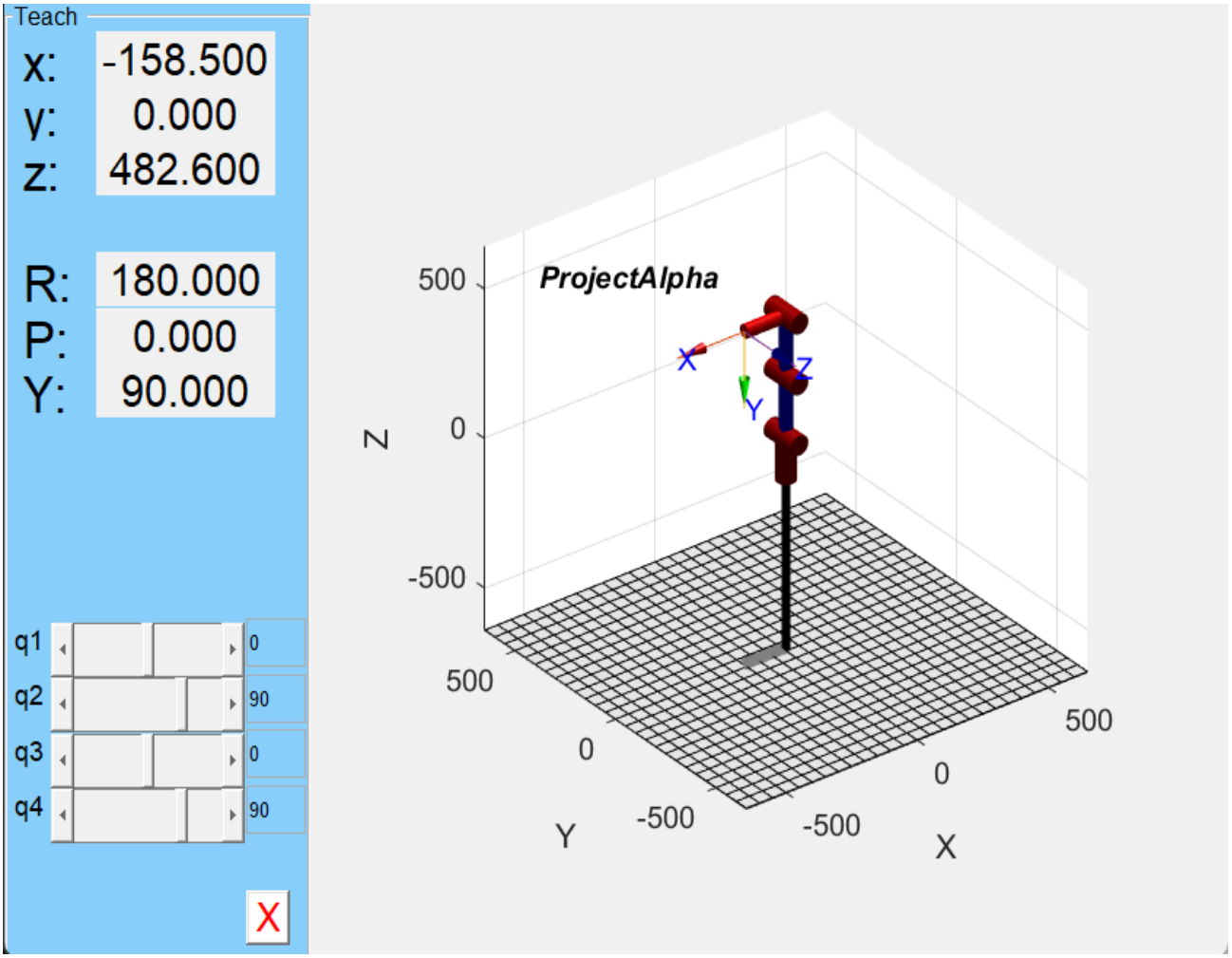

Figure 7: Plot for Configuration 1

Final Transformation matrix for configuration 1:

-1 0 0 -158.5

0 0 -1 0

0 -1 0 482.6

0 0 0 1

10.3. For Configuration 2:

Here,

$$[\boldsymbol{\theta_1 = 0},\ \boldsymbol{\theta_2 = \pi},\ \boldsymbol{\theta_3 = \frac{\pi}{2}},\ \boldsymbol{\theta_4 = -\frac{\pi}{2}}]$$

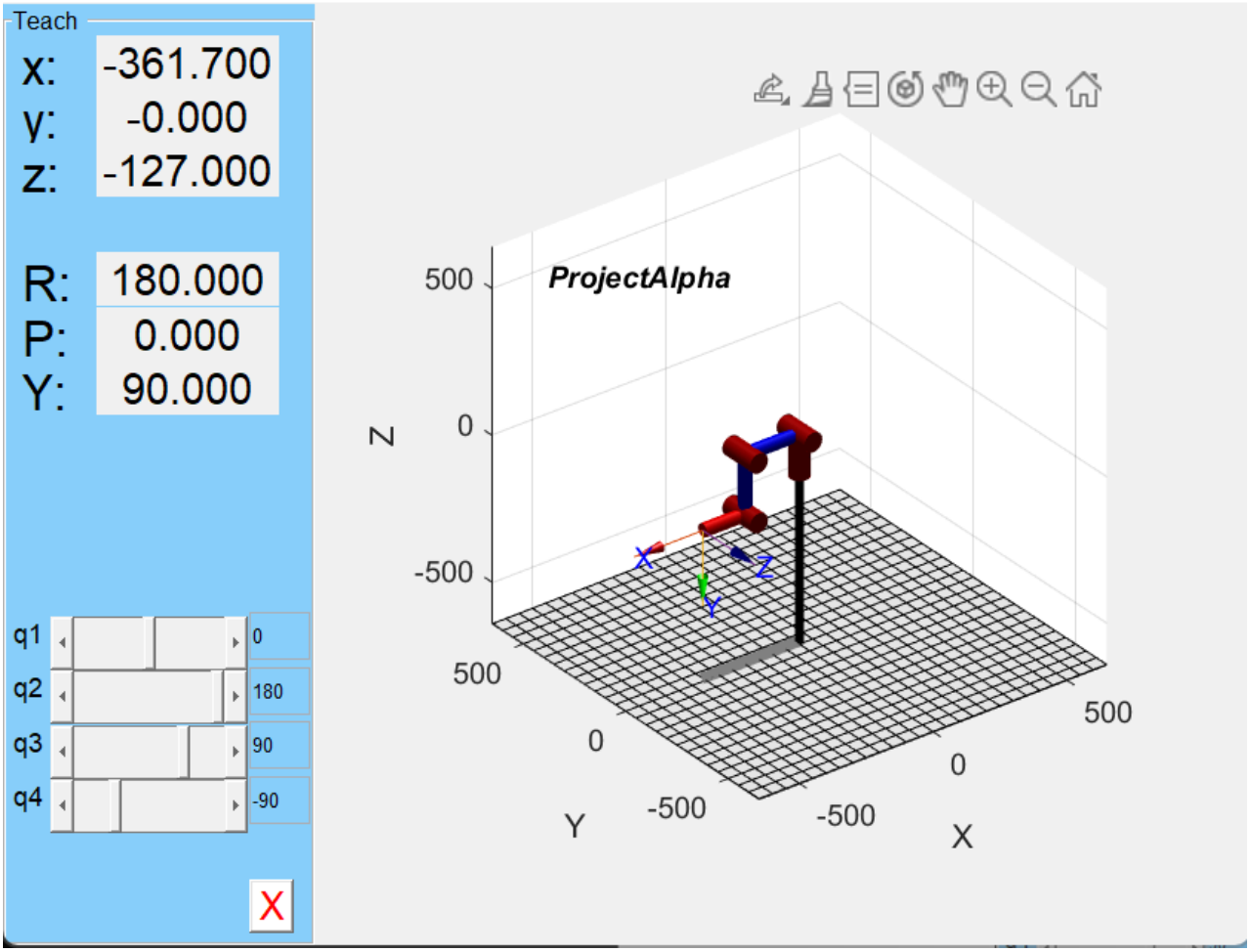

Figure 8: Plot for Configuration 2

Final Configuration Matrix for Configuration 2:

-1	0	0	-361.7
0	0	-1	0
0	-1	0	-127.0
0	0	0	1

10.4. For Configuration 3:

Here,

$$[\theta_1 = \theta_3 = -\frac{\pi}{2}]$$

$$[\theta_2 = \theta_4 = \frac{\pi}{2}]$$

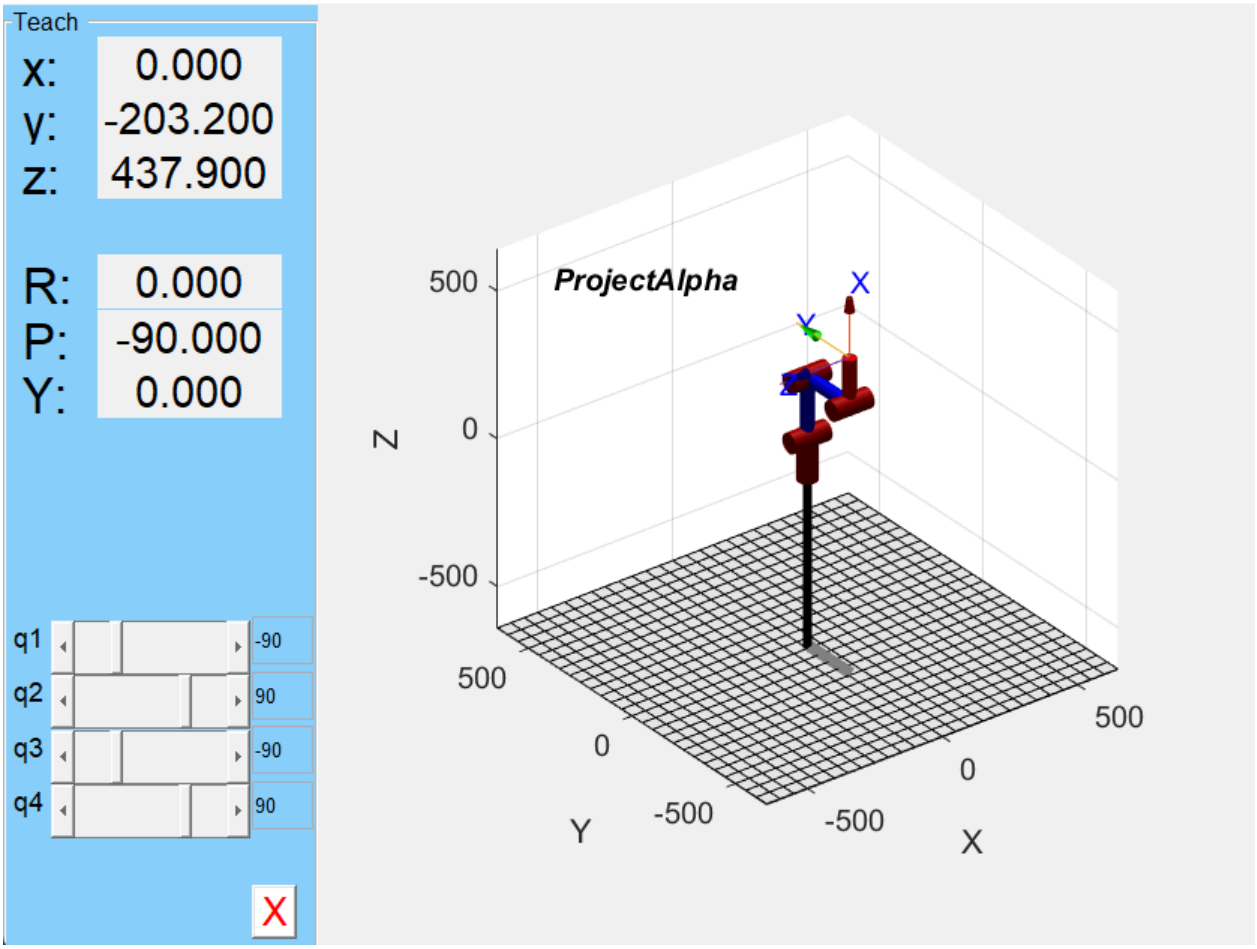

Figure 9: Plot for Configuration 3

Final Transformation Matrix for Configuration 3:

0	0	-1	0
0	1	0	-203.2
1	0	0	437.9
0	0	0	1

11. Inverse Kinematics of the Robot:

Inverse kinematics is done to find the joint variables from the position and orientation of end-effectors. Here, we use the Jacobian matrix obtained from the final transformation matrix of forward kinematics. Which is obtained using the following process:

- The Z-axis is extracted from each transformation matrix.
- Cumulative transformation matrices are calculated from the base to each joint.
- The End-effector position of the final cumulative transformation matrix is extracted.
- Partial derivatives of the end-effector position with respect to each joint angle is obtained.
- The obtained columns are assembled to form the complete Jacobian matrix.

The Final Jacobian Matrix for the forward kinematics is obtained below by using the python sympy libraries:

Now Using the Jacobian Matrix, the inverse kinematics of the robot is found by finding the inverse of the Jacobian matrix using the formula:

$$\boldsymbol{q' = J^{-1}(q)\ \times\ \varepsilon}$$

Where,

$$q' - Joint\ Velocity$$

$$q - Joint\ angles$$

12. Inverse Kinematics Validation:

The validation of inverse kinematics is done to check whether the inverse kinematics is reliable. Here we use a python script to make the end effector move in a particular trajectory. We are focusing on a circular trajectory on the XZ- plane. We are performing this to verify whether the end effector can move from one end of the diameter of the circle to another end.

The Circle equations are used in the python script to calculate the trajectory, which are given below:

$$\boldsymbol{X = C_x + Rcos(\theta)}$$

$$\boldsymbol{Z = C_Z + Rsin(\theta)}$$

Differentiating the above X and Z equations we get:

$$\boldsymbol{X' = -R\theta'\ sin(\theta)}$$

$$\boldsymbol{Z' = R\theta' cos\ (\theta)}$$

Here,

$$\boldsymbol{\theta = \theta' dt}$$

$$\boldsymbol{\theta' = \frac{2\pi}{T}}$$

X and Z are positions of the points on the XZ plane.

(C_x, C_z) is the center of the circle.

R is the radius of the circle.

θ' is the angular velocity of the trajectory.

T is the Time taken for the end-effector to complete one revolution.

The End-Effector Velocity is calculated by the formula:

$$\boldsymbol{\varepsilon = \left[v_x, v_y, v_z, \omega_x, \omega_y, \omega_z\right]^T}$$

Here, $\boldsymbol{\left[v_x = X', v_y = 0, v_z = Z'\right],\ [\omega_x = \omega_y = \omega_z = 0]}$

The Final Trajectory is given by the plot below:

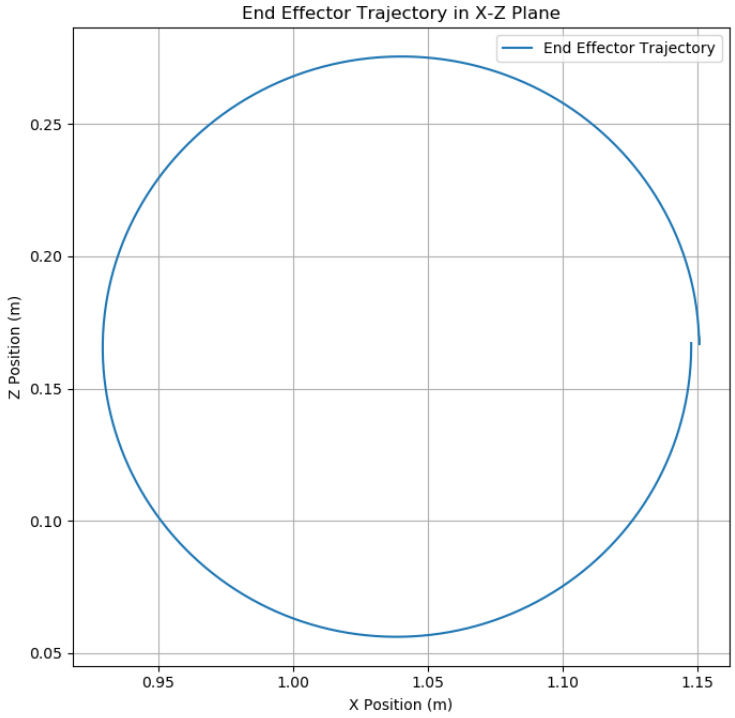

Figure 10: Plot Showing Circular Trajectory

The slight deviation at the circular trajectory is due to the number of iterations implemented. Further increase of iterations may lead to longer runtime.

13. Workspace Study:

The Workspace study is done to specify the various configurations that a robot's end-effector can reach. Although it does not have any impact on the task performed, we are doing this to analyze the effectiveness of the robot by understanding its limitations and capabilities.

The End-effector position of the robot is calculated by joint angle combinations. We used Denevit-Hartenberg Parameters and the final transformation matrices to get the joint angles.

A python script is used to calculate the end-effector positions. These positions are plotted using Matplot Libraries in a 3D space through which the workspace is represented.

The Results are analyzed from the graph:

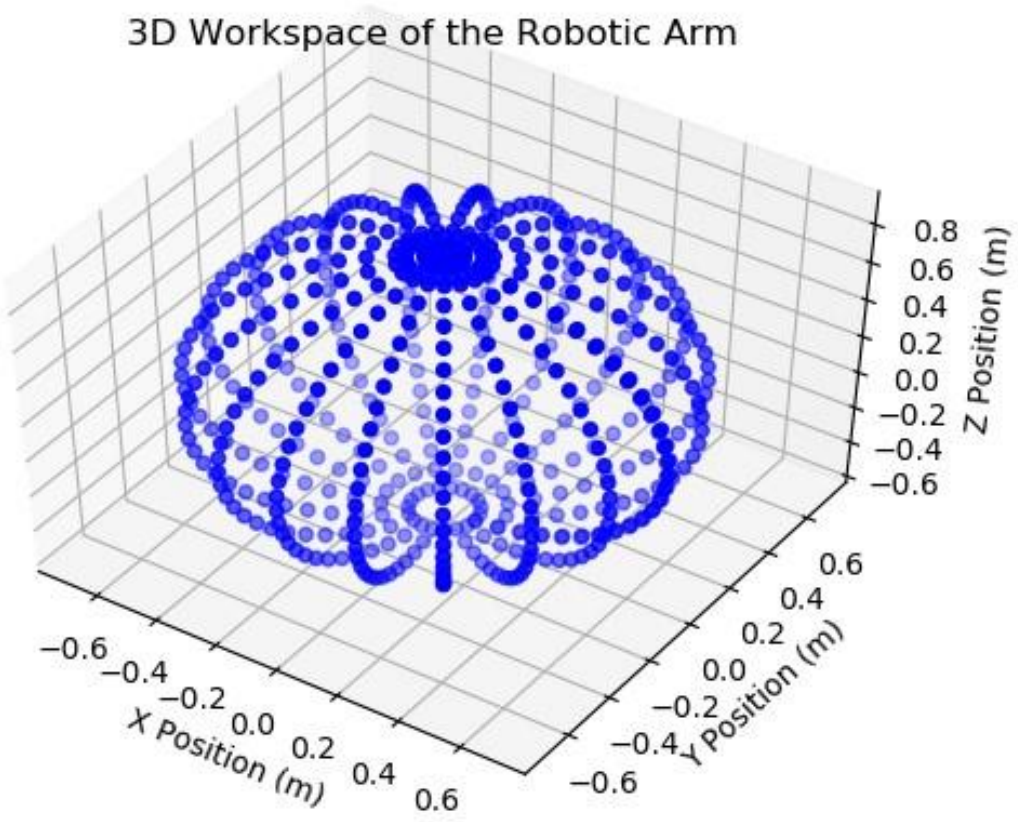

Figure 11: 3D Plot Showing Workspace Analysis

The analysis shows that the end-effector can reach the following extremes as shown below:

In X axis it can reach from 564.9mm to -564.9mm.

In Y axis it can reach from 564.9mm to -564.9mm.

In Z axis it can reach from 0 to 742.69mm.

14. Assumptions:

We are assuming that the manipulator moves only while the mobile base is idle. This is why we found the forward kinematics for the manipulator taking the origin of link 1 as reference.

15. Control Methods Used in the robot:

In order to control the robot, we have implemented two types of controllers:

- Tele-op controller
- Open-loop controller

15.1. Implementation of Tele-op controller:

Teleoperation is a method of controlling the robot using a keyboard, joystick or other manual control devices. Here we are controlling the robot using a keyboard.

The Controls are given below:

W – Move forward

A – Turn left

S – Move backward

D – Turn right

Q – Force Stop

I – For Increasing the current joint angle.

K – For Decreasing the current joint angle.

U – Selecting Previous joint

O – Selecting Next joint

ESC – Quit control loop.

The Working of the Robot using Tele-op control is shown in the link attached below:

https://youtu.be/TxN2vZlRw6Q- Tele-op PR2

Each and every joint can be controlled as per the user's interest. This is the major positive ability of the Tele-op controller.

15.2 Implementation of open-loop controller:

The open-loop controller is a non-feedback controller, which implies there is no need for feedback mechanisms. The robot moves in a straight path for the first 10 seconds, then it turns to the left side and continues to move in the straight path for the next 10 seconds.

A circular movement of the tip of the end-effector is achieved by using inverse kinematics. The Working video of the Mobile manipulator using the open-loop controller is attached as a link below:

https://youtu.be/CXajCPhOpYY - OpenLoop PR2

16. Gazebo and RViz Visualization:

There are several steps involved in making the robot work well in Gazebo environment and visualizing it in RViz. Those steps are mentioned below:

- Converting the SolidWorks model of the mobile manipulator to a URDF file and exporting it.
- The URDF file is exported from windows to Ubuntu operating system.
- Setting up of a workspace in Ubuntu to create a ROS2 package.
- URDF and Mesh folders are added to ROS2 package and CMakeLists are Updated.
- Modifying the URDF file, for it to be flexible to add colors, a LiDar Sensor and IMU.
- Providing mobility to the robot joints by adding controllers
- Sourcing Commands to Initialize the controllers and data is published to the joints.
- The LiDar properties of the robot are evaluated in RviZ and Obstacles are introduced to robot in order to test it.

The Working of the Robot in the Gazebo workspace is shown in the video link attached below:

https://youtu.be/ApNRZknIP0c

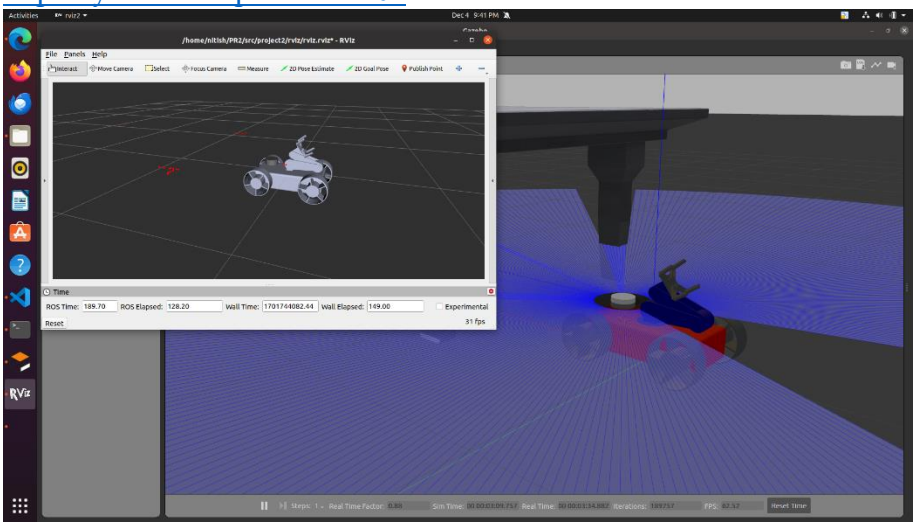

Figure 12: Robot subjected to RViz and in Gazebo World

17. Problems Faced:

In this Project, we faced a couple of problems in the following:

- Initially, there was an issue with turning robot as intended, which was rectified later by increasing the mass properties of the mobile base to increase stability.
- There was an issue after implementation of position controller where the robot's joints showed a tendency of shifting. We solved this problem by adding a velocity controller on the existing position controller to stabilize the joints.
- Larger command inputs were required for the four-finger claw. This was overcome by standardizing the final four publishing commands.
- The inverse kinematics caused issues by producing angles that are beyond the robot's operational limits. This was resolved by angle clamping.

18. Lessons Learned:

From this project, we got a lot of insight about the following:

- The Significance of being creative and adaptive while working on difficult challenges.
- The Difficulties in kinematics calculations and the importance of joint angle limits.
- The Necessity of effective computation methods that increase the ability of robots to respond.
- The Significance of Feedback mechanisms which was emphasized using open-loop controller.

19. Conclusion:

In this project, we have successfully Designed a Mobile Base Manipulator with a pick and place operation, computed the forward and inverse kinematics and validated them. We also spawned the robot manipulator in the gazebo world and analyzed it by controlling it using open-loop and tele-op controllers. As a result, we were able to get the desired output with slight deviations, which are negligible.

20. Future scope with this project:

There are few plans which we are planning to implement to this project in future like developing a method to efficiently calculate the joint angles, Making the manipulator work in real time conditions such as pick and placing of an actual object from one container to another container and implementing a closed-loop controller for a more efficient control. We are also planning to add a sensor to the robot to sense the object lying before it and pick it up without user input.

21. References:

https://enpm-662introduction-to-robot-modelling.readthedocs.io/

Robot Modeling and Control – Second Edition | Mark W. Spong | Seth Hutchinson | M. Vidyasagar

ROS 2 Documentation — ROS 2 Documentation: Rolling documentation