# ENPM 662: Project-2

## Medicines Transferring Robot

Group Number: 42

Vijay Chevireddi

UID: 119491485

Abraruddin Syed

UID: 120109997


Master of Engineering Robotics

A. James Clark School of Engineering
University of Maryland- College Park
USA

# INTRODUCTION

In the realm of robotics, the combination of manipulators with mobile platforms represents a significant leap towards versatility and functionality. Our project focuses on this integration by combining a UR10 manipulator with a mobile robot. This endeavour encompasses several critical steps, each contributing to the development of a sophisticated robotic system capable of complex tasks.

Our initial step involves constructing a detailed SolidWorks model of the UR10 manipulator mounted on a mobile robot. This detailed modelling is crucial for visualizing and understanding the physical constraints and capabilities of our robotic system. Subsequently, this model is converted into a URDF (Unified Robot Description Format) file. This conversion is essential for visualization and simulation purposes, particularly in the Gazebo simulation environment, which provides a realistic physics and rendering platform for testing robotic applications.

Further enhancing the system's functionality, we added position controllers to the joints of the UR10 manipulator. This addition allows for precise control of the manipulator's movements, a critical aspect for any task-oriented robotic system. To ensure the accuracy and reliability of our system, we conducted forward kinematics analysis. We validated this analysis using the Peter Corke toolbox in MATLAB, a renowned tool in the field of robotics for its robust analysis capabilities.

Building upon our progress, we also tackled the challenge of inverse kinematics. This complex process is vital for determining the necessary joint configurations to achieve a desired end-effector position. We validated our inverse kinematics solution by programming the manipulator to trace a semicircle, demonstrating the system's precision and agility.

The culmination of our project is showcased in the system's ability to perform a pick-and-place task. This task involves the manipulator picking up an object and precisely placing it in a different location. This demonstrates not only the integration of the various components and analyses but also the practical applicability of our system in a real-world scenario.

Overall, our project represents a comprehensive approach to integrating and controlling a robotic manipulator on a mobile platform. It demonstrates the intricate interplay between mechanical design, control theory, and practical implementation in robotics.

# APPLICATION

The main goal of your project is to develop a self-operating robot designed specifically to transport medications within healthcare environments, such as hospitals and pharmacies. This robot will be uniquely constructed with two major components:

Mobile Platform: This forms the base of the robot, enabling it to move around. The design of this platform is crucial for ensuring that the robot can navigate through different spaces within healthcare settings. This would typically involve integrating advanced wheel technology, such as omnidirectional wheels, which allow the robot to move in any direction without needing to turn its body. This feature is particularly useful in confined spaces common in healthcare facilities. Additionally, the platform should be built to maintain stability, ensuring that the robot doesn’t tip over while in motion, especially when carrying delicate medications.

Manipulative Arm: The second key component is a robotic arm designed for handling medications. This arm must be exceptionally precise and capable of delicate movements to avoid damaging the medications. The end-effector, or the part of the robot that interacts with the environment (in this case, the medications), needs to be versatile enough to handle containers of various sizes and shapes. The arm's design should focus on gentle yet firm handling to securely transport medicines from one location to another.

The integration of these two components aims to accomplish several objectives:

Reducing Manual Labor: By automating the process of transporting medications, the workload on healthcare staff can be significantly reduced.

Minimizing Errors: Automated systems can often carry out repetitive tasks with greater accuracy than humans, reducing the risk of errors in medicine handling.

Enhancing Speed and Safety: The robot is expected to operate more quickly than manual handling while also ensuring that medications are transported safely, without the risk of drops or mishandling.

Overall, this robot is envisioned to streamline the process of medicine handling in healthcare settings, making it more efficient, safe, and error-free.

# ROBOT TYPE

UR10 Manipulator on top of a Mobile Robot

# PARTS

## UR10 MANIPULATOR-

The UR10 is a versatile robotic arm developed by Universal Robots, known for its flexibility and user-friendly interface. It's part of the company's series of collaborative robots (cobots), designed to work alongside human workers in a variety of settings, including manufacturing, packaging, assembly, and research.

Here's a detailed look at the UR10 manipulator:

**Key Features**

**Payload and Reach:**

The UR10 has a payload capacity of 10 kg (22 lbs), which makes it suitable for handling heavier objects compared to its smaller counterparts, like the UR5.

It boasts a reach of 1300 mm, providing a wide working radius.

**Flexibility and Movement:**

The arm has six rotational joints, granting it a high degree of freedom in movement. This flexibility allows it to mimic the range of motion of a human arm, making it capable of performing complex tasks.

Its design allows for easy reprogramming and redeployment for different tasks, increasing its utility in various applications.

**Ease of Use:**

One of the standout features of the UR10 is its ease of programming. It can be programmed through a touchscreen interface or by physically moving the arm to desired positions (a process known as "teach programming").

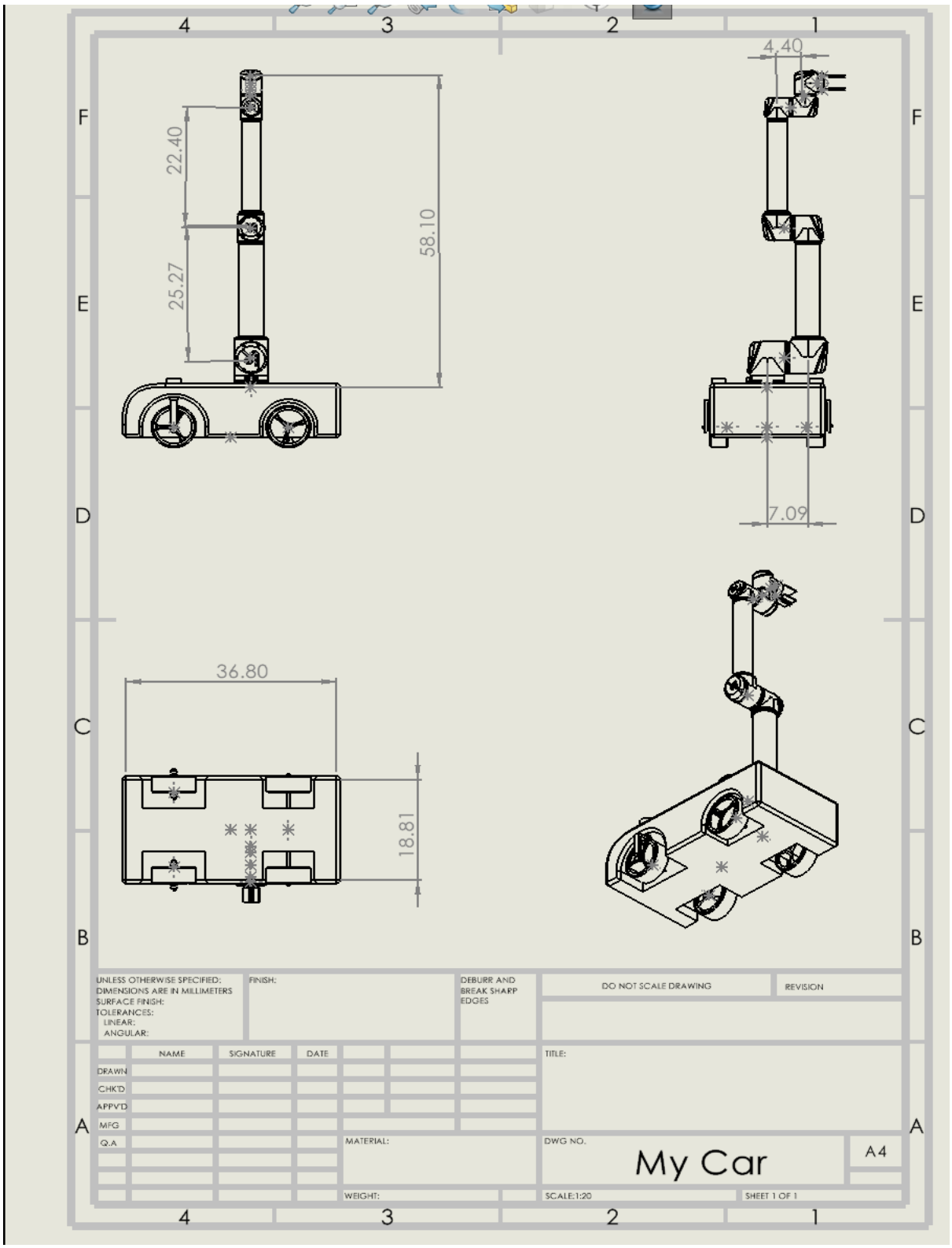
It has a user-friendly interface that doesn't require extensive programming knowledge, making it accessible to a broader range of users.

Safety: As a collaborative robot, the UR10 is designed to work safely alongside humans. It has built-in safety features such as force limiters, which allow it to stop automatically if it encounters an unexpected obstacle or a certain level of resistance.

These safety mechanisms make it suitable for environments where human-robot interaction is frequent.

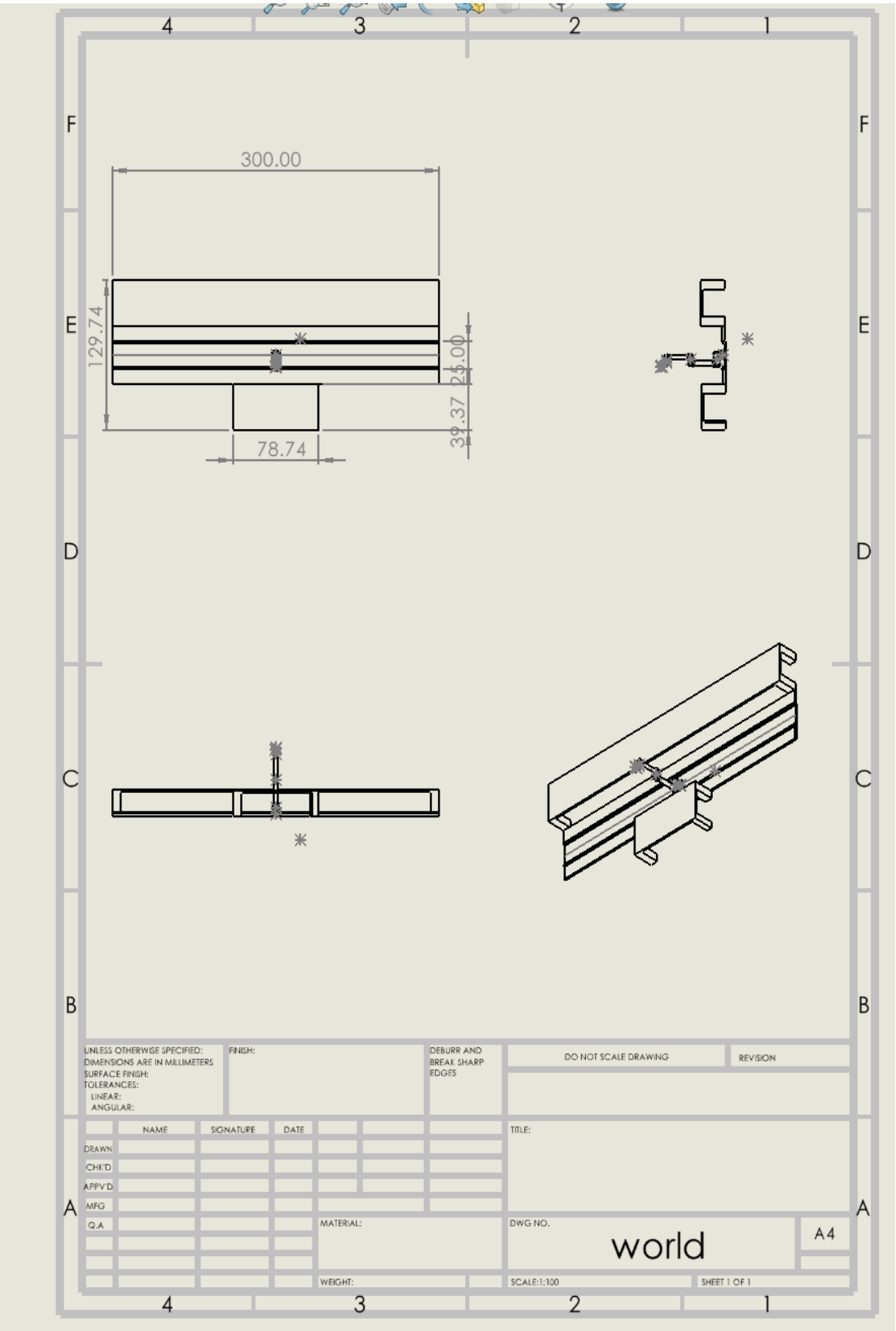
## MOBILE ROBOT-

The UR10 manipulator was placed on top of Mobile Robot

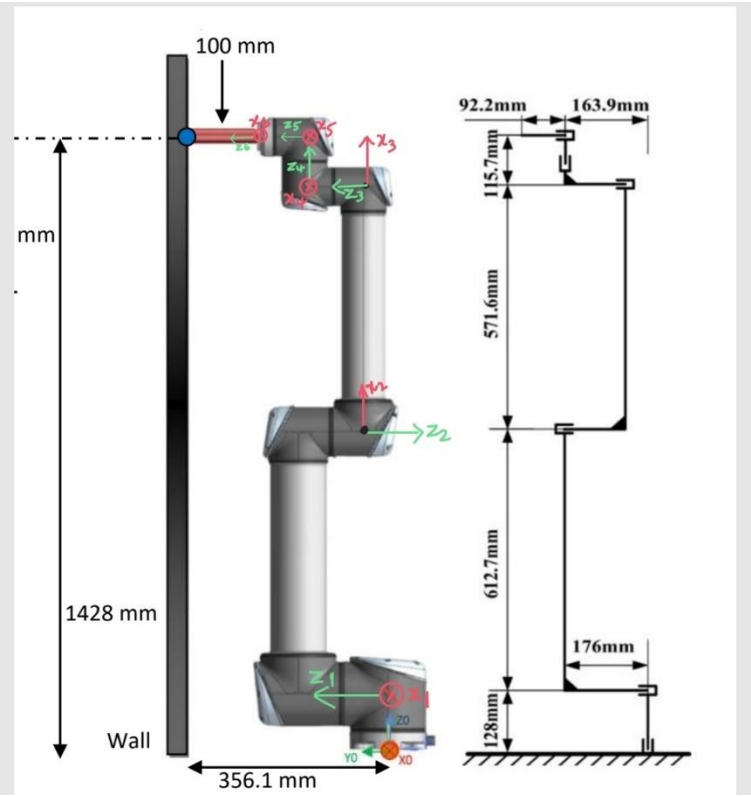


## WORLD –

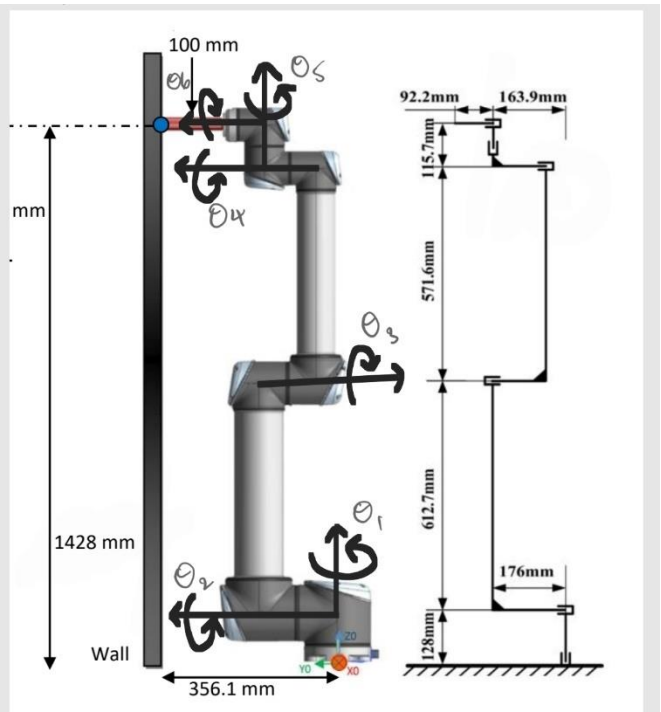
A table was created to place the medicine box. Moreover, a ramp was used as a guide to make the robot move in a straight path.



# FRAME ASSIGNMENT




# Relative Movement of Joints



# DH PARAMETERS

**Updated DH table:(all values are in mm and degrees)**

| Link | θi | di | ai | αi |
|---|---|---|---|---|
| 0-1 | θ1 | 128 | 0 | -90 |
| 1-2 | θ2 -90 | 0 | 612.7 | 180 |
| 2-3 | θ3 | 0 | 571.6 | 180 |
| 3-4 | θ4 +90 | 163.9 | 0 | 90 |
| 4-5 | θ5 | 115.7 | 0 | -90 |
| 5-6 | θ6 | 192.2 | 0 | 0 |

# FORWARD KINEMATICS

Forward kinematics (FK) is the process used to determine the position and orientation of a robot's end effector (such as a hand or tool) given the values of its joint parameters (like angles for rotational joints or displacements for prismatic joints). This is typically straightforward to compute. In FK, you start at the base of the robot and work your way through each joint, calculating the position and orientation of the end effector step by step. The process involves using the known joint parameters (like angles and lengths) and applying them to the robot's linkage geometry. By combining these parameters with the robot's kinematic equations, you can predict the final position and orientation of the end effector.

Forward kinematics involve transforming from the joint space, denoted as $q_i$ , to Cartesian coordinates (x, y, z, Φ, Θ, Ψ). This transformation is utilized to ascertain the position and orientation of a manipulator's end-effector, based on the provided joint angles using kinematic equations. However, within the Denavit-Hartenberg (D-H) convention, only the position coordinates (x, y, z) are calculated. The D-H convention assigns four specific parameters to each link $'i'$ in the manipulator:

- $\theta_i$: the angle between the $x_{i-1}$ and $x_i$ axes, measured around the $z_{i-1}$ axis.
- $d_i$: the distance measured along $z_{i-1}$ axis to the common normal.
- $a_i$: the length of the common normal, or the distance along the $x_i$ axis.
- $\alpha_i$: the angle between the $z_{i-1}$ and $z_i$ axes, measured around the $x_i$ axis.

The equations that describe the motion of a robot's series chain are derived through the application of a rigid transformation, denoted as [Z], which represents the movement possibilities at each joint. Additionally, another rigid transformation, labelled [X], is used to specify the dimensions of each link in the chain. This process involves creating a series of rigid transformations that alternate between joint movements and link dimensions, starting from the base of the chain and extending to the final link. This sequence is then matched with the predetermined position of the end link.

$$T_6^0 = T_1^0 . T_2^1 . T_3^2 . T_4^3 . T_5^4 . T_6^5$$

The use of the Denavit-Hartenberg convention yields the link transformation matrix, $[^{i-1}T_i]$ as

$$ {}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_{i,i+1} & \sin\theta_i\sin\alpha_{i,i+1} & a_{i,i+1}\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_{i,i+1} & -\cos\theta_i\sin\alpha_{i,i+1} & a_{i,i+1}\sin\theta_i \\ 0 & \sin\alpha_{i,i+1} & \cos\alpha_{i,i+1} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$

- We know that at t = 0 (initial condition) all the values of theta are 0(i.e, q= 0 0 0 0 0 0T). For this initial condition we get the manipulator at the erect home position as seen in figure 1.
- At this initial condition values for theta and the values shown in the DH table we can build the intermediate transformation matrices (T10, T21, T32, T43, T54, T65).
- The intermediate transformation matrices, we can build these matrices T10, T20, T30, T40, T50, T60 by respective matrix multiplication respectively as follows:

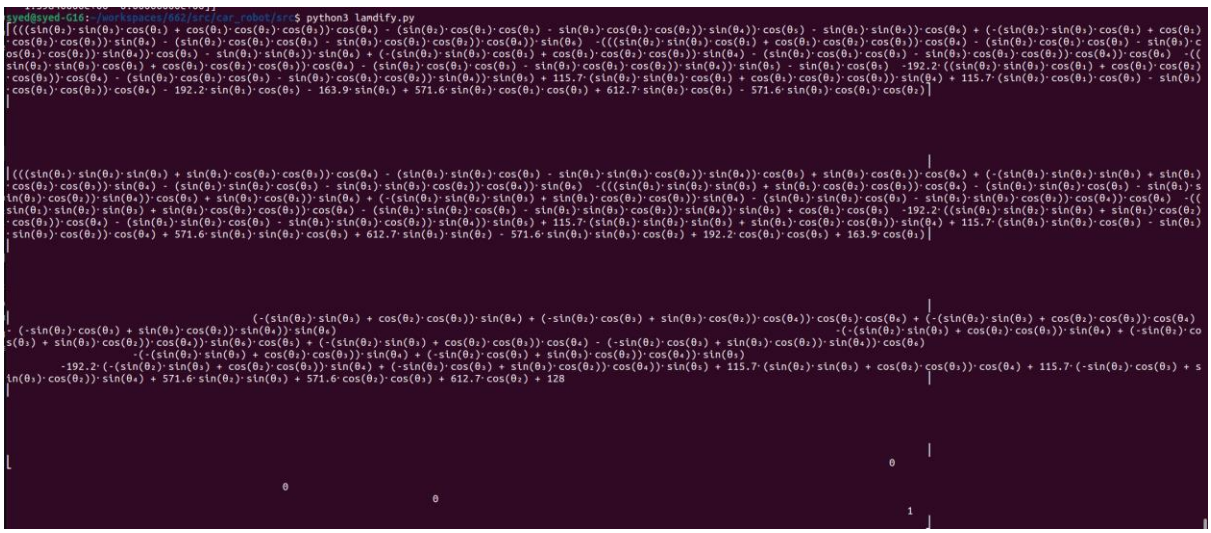
$$\left[\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 128 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 740.7 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1312.3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 163.9 \\ 0 & 0 & 1 & 1312.3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}\right.$$

$$\left.\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 163.9 \\ -1 & 0 & 1428.0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 356.1 \\ 0 & 1 & 0 & 1428.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\right]$$

- The FINAL TRANSFORMATION MATRIX (T60):

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 356.1 \\ 0 & 1 & 0 & 1428.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

GENERIC FINAL TRANSFORMATION MATRIX:



# INVERSE KINEMATICS

Inverse kinematics (IK) in the context of robotics and computer animation is a key concept, especially when dealing with manipulators (robotic arms) or animated characters. The main objective of IK is to compute the necessary joint angles or movements that will position the end of a kinematic chain (like a robot's arm or an animated character's limb) in a desired location and orientation, starting from the base or the first joint of the chain.

**The Jacobian Matrix computation:**

We used the second method to compute the Jacobian Matrix where I use the approach of Partial differentiation of the translation component w.r.t. the joint angles.

In the program, we used functions to create a block of codes to compute the transformation matrices using below generalized form:

Transformation matrix

$$ {}^{i-1}_{i}T = A_i = \begin{bmatrix} c\theta_i & -c\alpha_i s\theta_i & s\alpha_i s\theta_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -s\alpha_i c\theta_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

While computing the Homogeneous transformation matrix with respect to base frame, I appended the first three elements of third column of these Homogeneous transformation Matrices from corresponding frames to frame 0 into z matrix and the Xp from the first three elements of the fourth column of the Final Homogeneous transformation matrix into x_p_matrix.

I calculation Jacobian for each joint using the below formulae and appended each Jacobian matrix to form final Jacobian matrix.

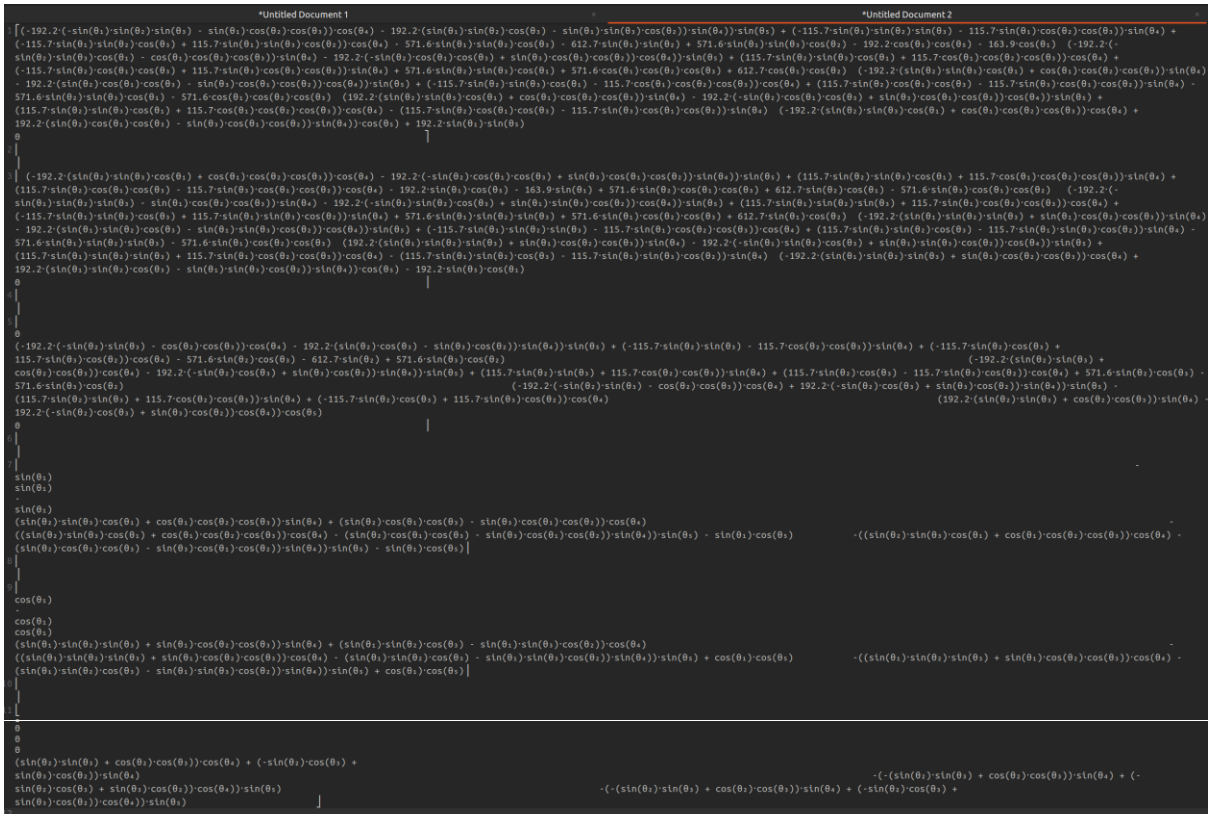
$$ {}^{0}_{n}J = \begin{bmatrix} \frac{\partial {}^{0}X_p}{\partial q_1} & \cdots & \frac{\partial {}^{0}X_p}{\partial q_i} & \cdots & \frac{\partial {}^{0}X_p}{\partial q_n} \\ {}^{0}Z_1 & \cdots & {}^{0}Z_i & \cdots & {}^{0}Z_n \end{bmatrix} $$

```
partial_xp_theta1 = sp.diff(x_p_matrix, theta1)
partial_xp_theta2 = sp.diff(x_p_matrix, theta2)
partial_xp_theta3 = sp.diff(x_p_matrix, theta3)
partial_xp_theta4 = sp.diff(x_p_matrix, theta4)
partial_xp_theta5 = sp.diff(x_p_matrix, theta5)
partial_xp_theta6 = sp.diff(x_p_matrix, theta6)


J1 = sp.Matrix([[partial_xp_theta1], [z_10]])
J2 = sp.Matrix([[partial_xp_theta2], [z_20]])
J3 = sp.Matrix([[partial_xp_theta3], [z_30]])
J4 = sp.Matrix([[partial_xp_theta4], [z_40]])
J5 = sp.Matrix([[partial_xp_theta5], [z_50]])
J6 = sp.Matrix([[partial_xp_theta6], [z_60]])

jacobian_matrix = sp.Matrix([[J1, J2, J3, J4, J5, J6]])
```

The above code showcases the calculation of the Jacobian matrix. The Jacobian Matrix is a function of joint angles i.e (theta1, theta2, theta4, theta5, theta6,theta7), and hence the Jacobian matrix at initial position when the angles are as [0.0, 0.0, 0.0, 0.0, 0.0, 0.0] is:

```
The Jacobian matrix for the initial position is :
⎡-356.1  1300.0  -687.3  115.7  -192.2  0⎤
⎢                                        ⎥
⎢  0       0       0       0      0     0⎥
⎢                                        ⎥
⎢  0       0       0       0      0     0⎥
⎢                                        ⎥
⎢  0       0       0       0      0     0⎥
⎢                                        ⎥
⎢  1       -1      1       0      1     1⎥
⎢                                        ⎥
⎣  0       0       0       1      0     0⎦
```

GENERIC JACOBIAN MATRIX:



# FORWARD KINEMATICS VALIDATION

Forward Kinematics Validation was done using Peter Corke Toolbox.

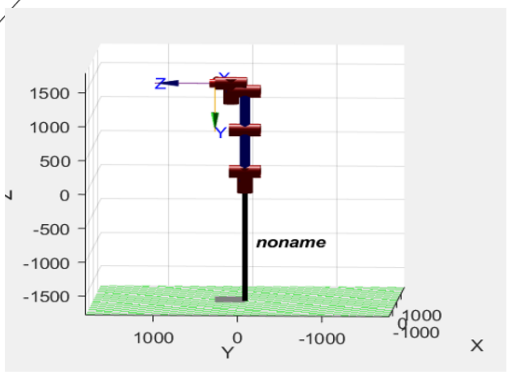
MATLAB implementation of Peter Corke Toolbox:

```
dh = [
    [0.0, 128, 0, -pi/2],
    [0, 0, -612.7, pi],
    [0.0, 0, -571.6, 0.0],
    [0, -163.9, 0.0, -pi/2],
    [0.0, 115.7, 0, -pi/2],
    [0.0, 192.2, 0, 0.0]
]
robot = SerialLink(dh);
%robot
robot.fkine([0 pi/2 0 0 -pi 0])
robot.plot([0 pi/2 0 0 -pi 0])
```

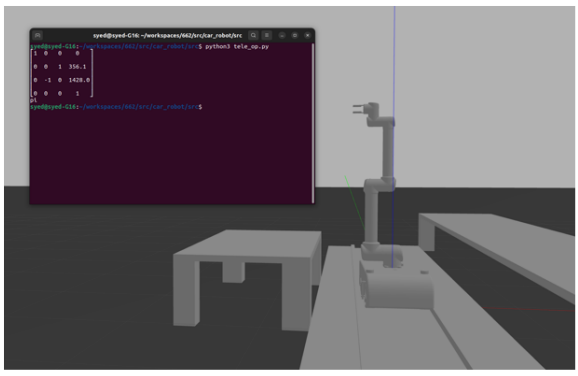
- Initial Theta Values q= [0 0 0 0 0 0]

Q = [ 0 0 0 0 0 0 ]

PETER CORKE




GAZEBO VISUALISATION



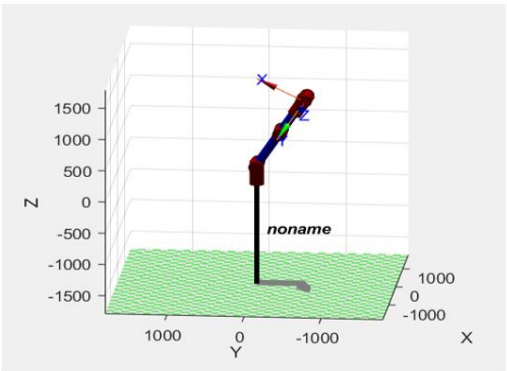
```
ans =
     1     0     0     0
     0     0     1   356.1
     0    -1     0  1428
     0     0     0     1
```



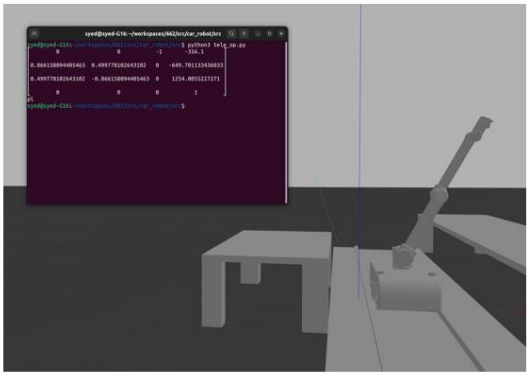
- Initial Theta Values q=[pi/2-pi/6 0 0 0 0]

Q = [ pi/2 –pi/6 0 0 0 0 ]

PETER CORKE




GAZEBO VISUALISATION

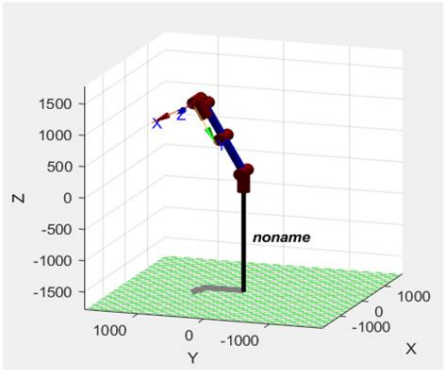


```
ans =
         0         0   -1.0000  -356.1000
    0.8660    0.5000         0  -650
    0.5000   -0.8660         0  1254
         0         0         0     1
```

- Initial Theta Values q= [pi/2 pi/6 0 0 0 0]

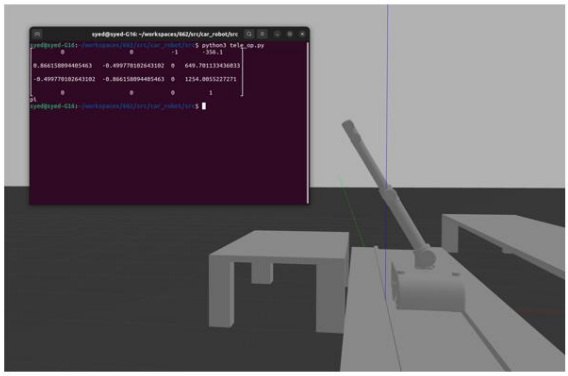
Q = [ pi/2 pi/6 0 0 0 0 ]

PETER CORKE




```
ans =
         0         0   -1.0000  -356.1000
    0.8660   -0.5000         0   650.0000
   -0.5000   -0.8660         0  1254.0000
         0         0         0     1.0000
```

GAZEBO VISUALISATION

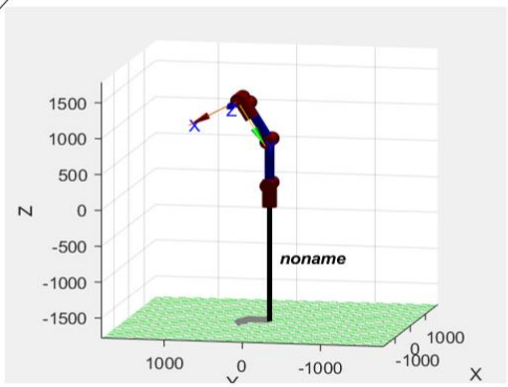




- Initial Theta Values q= [pi/2-pi/2 -pi/6 pi/2 0 0]

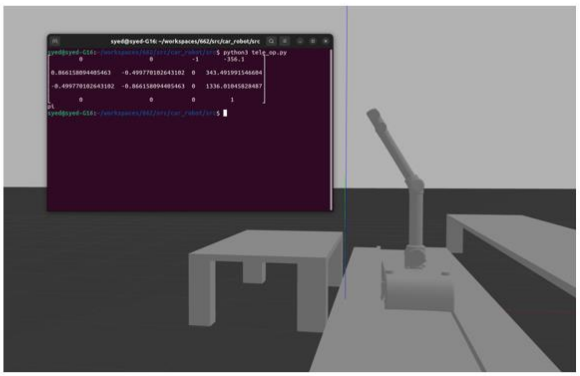
Q = [ pi/2 -pi/2 –pi/6 pi/2 0 0 ]

PETER CORKE




```
ans =
         0         0   -1.0000  -356.1000
    0.8660   -0.5000         0   343.7000
   -0.5000   -0.8660         0  1336.0000
         0         0         0     1.0000
```

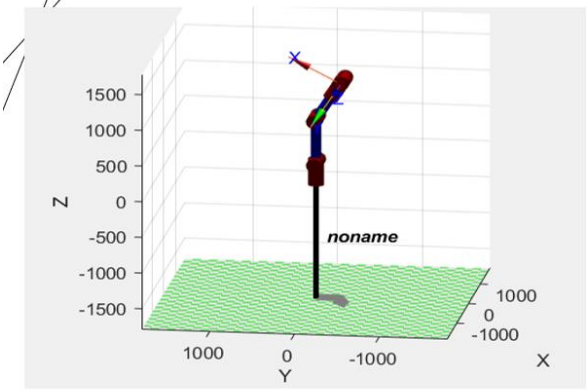
GAZEBO VISUALISATION



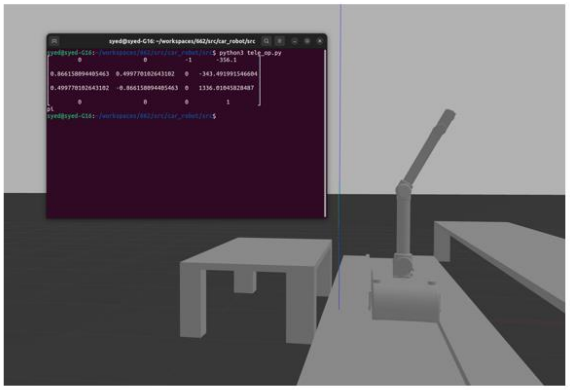
- Initial Theta Values q= [pi/2-pi/2 pi/6  pi/2 0 0]

```
Q = [ pi/2 -pi/2 pi/6 pi/2 0 0 ]
```

PETER CORKE




GAZEBO VISUALISATION



```
ans =
         0         0   -1.0000  -356.1000
    0.8660    0.5000         0  -343.6000
    0.5000   -0.8660         0   1336.0000
         0         0         0    1.0000
```



# INVERSE KINEMATICS VALIDATION

We validated the inverse kinematics of the manipulator by making it trace a semi-circle.

Circle Trajectory Velocities:

$$V = \frac{d}{t} => \frac{2\pi \times 250}{20}$$

$$V = 25\pi\ mm/sec$$

Also, $\omega = V/R$ => $\frac{10\pi}{250}$ => $\frac{\pi}{25} rad/sec$

The velocity of the pencil should be $\frac{\pi}{25}$ tracing the circle.

$$x = rcos\theta => rcos\omega t$$

$$y = 0$$

$$z = rsin\theta => rsin\omega t$$

The differentiation of the above equations with respect to time gives us linear velocity in x, y, and z directions respectively.

$$\dot{x} = V_x = -r\omega sin\theta$$

$$\dot{y} = V_y = r\omega cos\theta$$

$$\dot{z} = V_z = 0$$

Moreover, we need to consider $\frac{\pi}{2}$ as phase difference to start the circle from desired point.

$$\dot{x} = V_x = -r\omega \sin\left(\theta + \frac{\pi}{2}\right)$$

$$\dot{y} = V_y = r\omega cos\left(\theta + \frac{\pi}{2}\right)$$

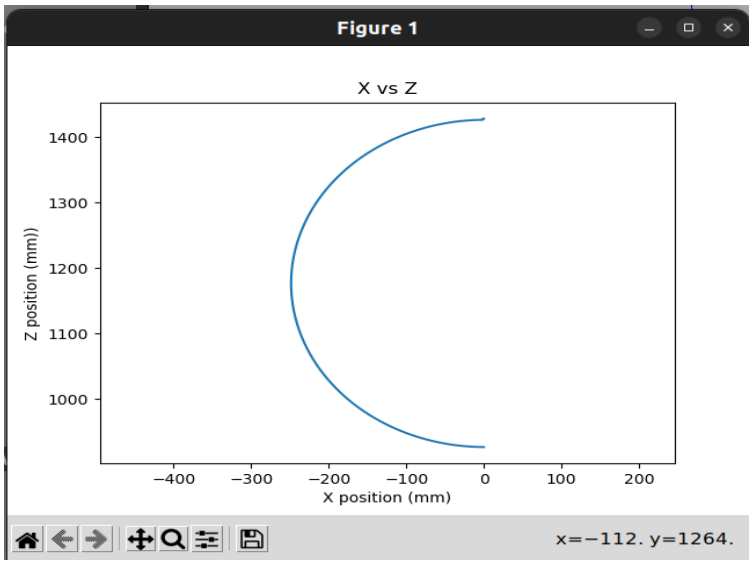
$$\dot{z} = V_z = 0$$

$$\omega_x = 0$$

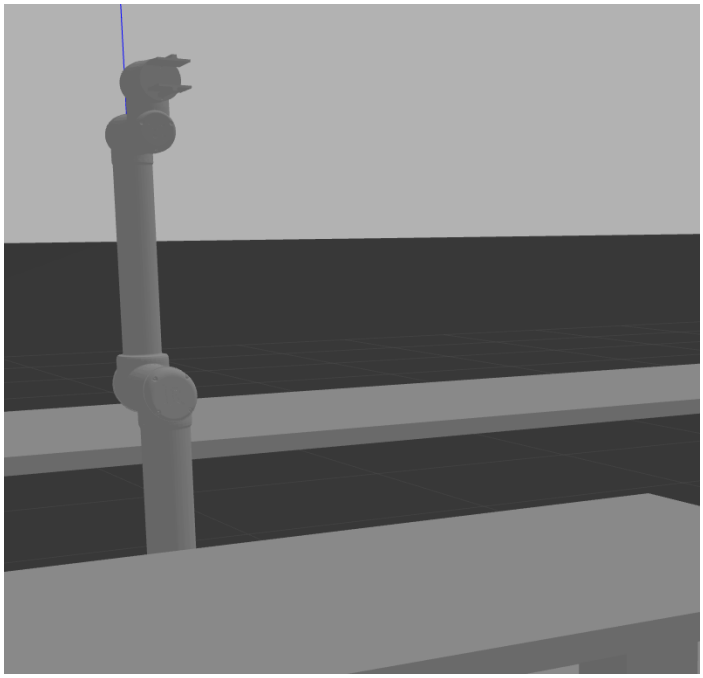
$$\omega_y = 0$$

$$\omega_z = 0$$

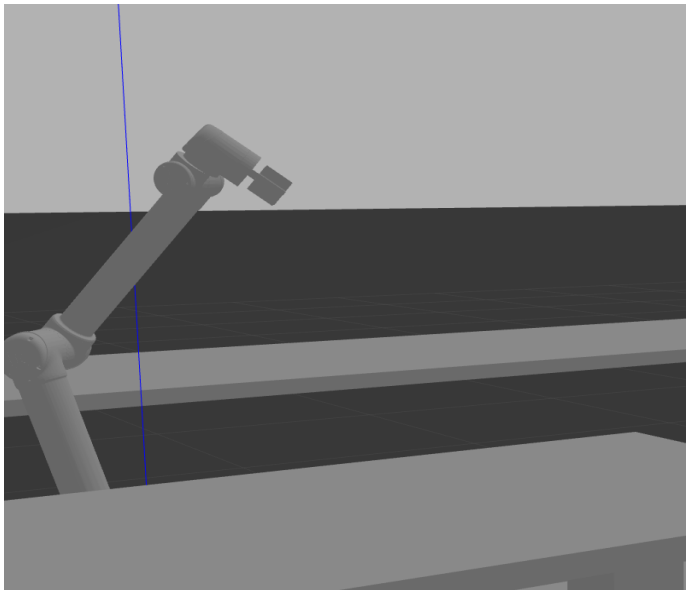
**Path Traced:**



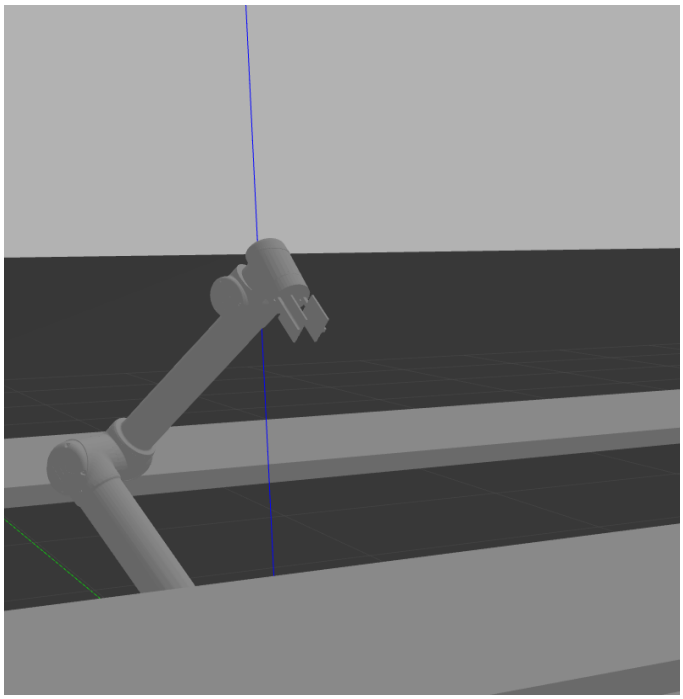

**Initial Pose:**



**Intermediary Pose:**



**Final Pose:**



Video Link that shows the gazebo visualization of this movement:

https://drive.google.com/file/d/1ysZdQxD78NK-CAfd_Dpuean11q0eC0Bw/view?usp=sharing

## ASSUMPTIONS

The affect of gravity of the links were neglected. In real world we have to compensate for the gravity force on the links.

## CONTROL METHOD

In our project, we implemented an open-loop control system for the UR10 manipulator mounted on a mobile robot. This method involves directly sending input commands to the manipulator without feedback for real-time adjustments. The accuracy of the manipulator's movements depends largely on the precision of these initial commands and the predictable response of the system.

Open-loop control was chosen for its simplicity and efficiency, especially suitable for executing predetermined tasks in controlled environments. While this approach lacks the ability to compensate for unexpected disturbances, it was considered appropriate for our specific applications, such as tracing pre-defined paths and performing specific manoeuvres in stable conditions.

## GAZEBO VISUALIZATION OF PICK AND PLACE

https://drive.google.com/file/d/1UZokWf7jdvRplK_B3MUOam8u5n8nlwS1/view?usp=sharing

## PROBLEMS FACED

- We faced errors in making the bot trace the desired path.
- It was tedious to find the optimal number of iterations to precisely trace the path.
- The bot used to move a lot while tracing the path. We removed the mobile robot and just used the UR10 manipulator for pick and place operation to overcome this.
- The object that we placed for the manipulator to pick was moving away as the flaps close on it. So, in the video that where the pick and place operation was shown we did not keep any object on the table.

## LESSONS LEARNED

**Significance of Accurate Designing:** Our work in creating a detailed SolidWorks model for the UR10 manipulator on the mobile robot underscored the criticality of precise and thorough design in robotics. This step was key to understanding the system's physical limits and abilities, laying a solid groundwork for subsequent simulations and control strategies.

**Utility of Simulation and Validation Tools:** Employing simulation tools like Gazebo and the Peter Corke toolbox in MATLAB was invaluable. They allowed us to test our designs and hypotheses in a risk-free, cost-effective virtual setting, proving essential for refining our models and strategies.

**Choosing the Right Control System:** Opting for an open-loop control system brought to light the importance of aligning the control strategy with the project's goals and the operational context. This decision was a practical lesson in balancing efficiency with the level of control precision required.

**Role of Kinematics in Robotics:** The project reinforced the pivotal role that kinematics plays in the control and movement of robotic systems. Successfully implementing and validating the kinematics models was a hands-on lesson in the intricacies of programming and controlling robotic movements.

**Limitations of Non-Feedback Systems:** Working within the constraints of an open-loop system highlighted the limitations posed by the absence of a feedback mechanism, especially in dynamic or unpredictable environments. This experience emphasized the critical role of feedback in enhancing adaptability and accuracy.

**Effective Collaboration and Project Management:** The project was a practical example of the necessity for good project management and teamwork in engineering. Coordinating various project facets demanded effective communication, collaborative problem-solving, and a unified approach.

**Adaptability and Creative Problem Solving:** Facing and overcoming unforeseen challenges during the project was a lesson in adaptability and creativity. These skills are indispensable in engineering, particularly when dealing with sophisticated systems such as robotic manipulators.

# CONCLUSION

Our project's journey in integrating a UR10 manipulator onto a mobile robot platform and controlling it through a series of sophisticated processes has been both challenging and enlightening. Through the detailed modelling in SolidWorks, conversion to a URDF file for Gazebo simulation, and the implementation of open-loop control, we have successfully demonstrated the capabilities of our robotic system in various tasks.

The use of open-loop control, while limiting in terms of adaptability to environmental changes, proved to be efficient and effective for the predetermined tasks we set out to accomplish. The precision with which the manipulator executed tasks, from tracing a semicircle to performing pick-and-place operations, showcased the robustness of our design and control strategy.

One of the key achievements of this project was the successful validation of both forward and inverse kinematics using the Peter Corke toolbox in MATLAB. This not only verified the accuracy of our theoretical models but also provided a strong foundation for the practical applications demonstrated.

However, it's important to recognize the limitations inherent in an open-loop system, particularly its lack of feedback mechanisms to dynamically adjust to environmental changes or unforeseen errors. Future enhancements could include the integration of closed-loop control systems for improved adaptability and precision.

In conclusion, this project has not only achieved its initial goals but has also provided valuable insights into the complexities of robotic manipulation and control. It serves as a promising foundation for further research and development in the field of robotics, opening doors to more advanced applications and control methodologies.

# FUTURE WORK

- We can connect the base link of the UR10 manipulator to a prismatic link to increase the reach of the manipulator.

- We can use computer vision and machine learning models to make the robot purely autonomous and make it learn from its mistakes just like us (humans)

# REFERENCES

https://www.universal-robots.com/

https://wiki.ros.org/Documentation

https://gazebosim.org/home

https://my.solidworks.com/training/catalog