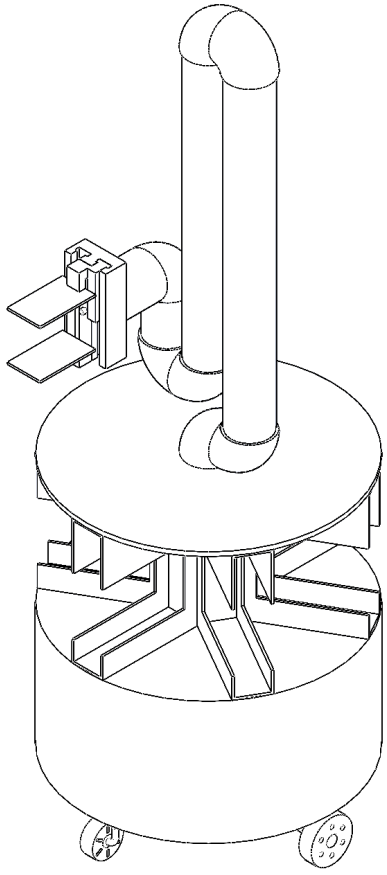
The Library Robot

ENPM662 - Introduction to Robot Modeling

December 8, 2024

Group 1



Contents

Team Members

Name	Areas of Focus
Anne-Michelle Lieberson (aliebers@umd.edu)	• ROS Development • Demo Orchestration
Christopher Collins (ccollin5@umd.edu)	• ROS Development • World Building
Daniel Zinobile (zinobile@umd.edu)	• Robot Design • SolidWorks Modeling
James Fehrmann (jsf@umd.edu)	• DevOps • ROS Development

Introduction

The objective of this project was to design, model, and demonstrate a mobile robot that can retrieve and return books to and from library shelves. The simulation will demonstrate the robot's ability to move to a specified location, retrieve a book from the robot's storage system, and place the book on a library shelf. There are two motivations for this robot:

- The robot can aid with book retrieval tasks, such as if an individual is physically unable to reach a book off a shelf.
- The robot can automate tasks normally performed by library staff, such as reshelving returned books.

Application

The design of the robot is inspired by the UJI librarian robot described here:

The UJI robot was able to autonomously locate and retrieve a book in an ordinary library, provided only with the book code and a library map. The UJI robot employed stereo vision, visual tracking, and other techniques to locate and retrieve the book.

Group 1's robot simulation will focus on kinematics and will not include path planning and environmental response capabilities such as those found on the UJI librarian robot.

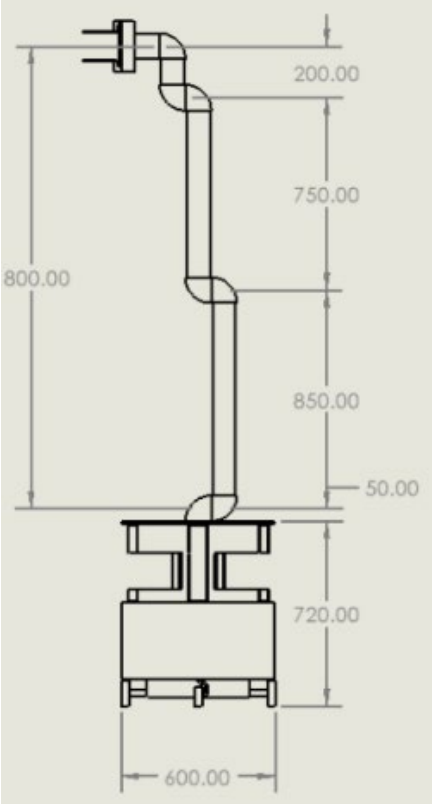
Robot Type

The Library Robot is effectively two robots in a single package. The base is a mobile robot consisting of two drive wheels, two wheels for directional control, and six slots for storing books of assorted sizes. The lower section will have a cavity capable of holding large batteries for extended runtime and additional ballast.

Mounted to the top of the base robot is a manipulator arm with five links. The links and joints have a compact design with a low center of gravity to facilitate locomotion without risk of tipping.

The dimensions of the arm are based on those of the UR3, but the CAD model was designed from scratch. The manipulator arm has a custom gripper capable of rotating 360 degrees and independently controlled pads. The base and storage slots are also custom.

DOFs and Dimensions

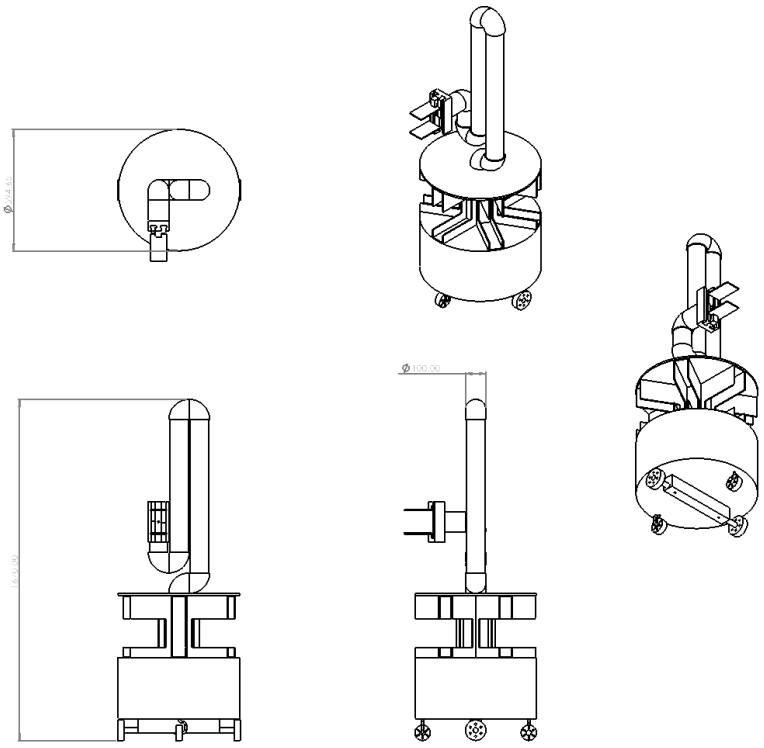


The mobile base has three degrees of freedom that allow it to move across a large, flat workspace to a desired x-y coordinate and 2D orientation. The arm and gripper assembly combine for six degrees of freedom, allowing the gripper to be placed at any x-y-z coordinate and 3D orientation inside the robot's effective workspace.

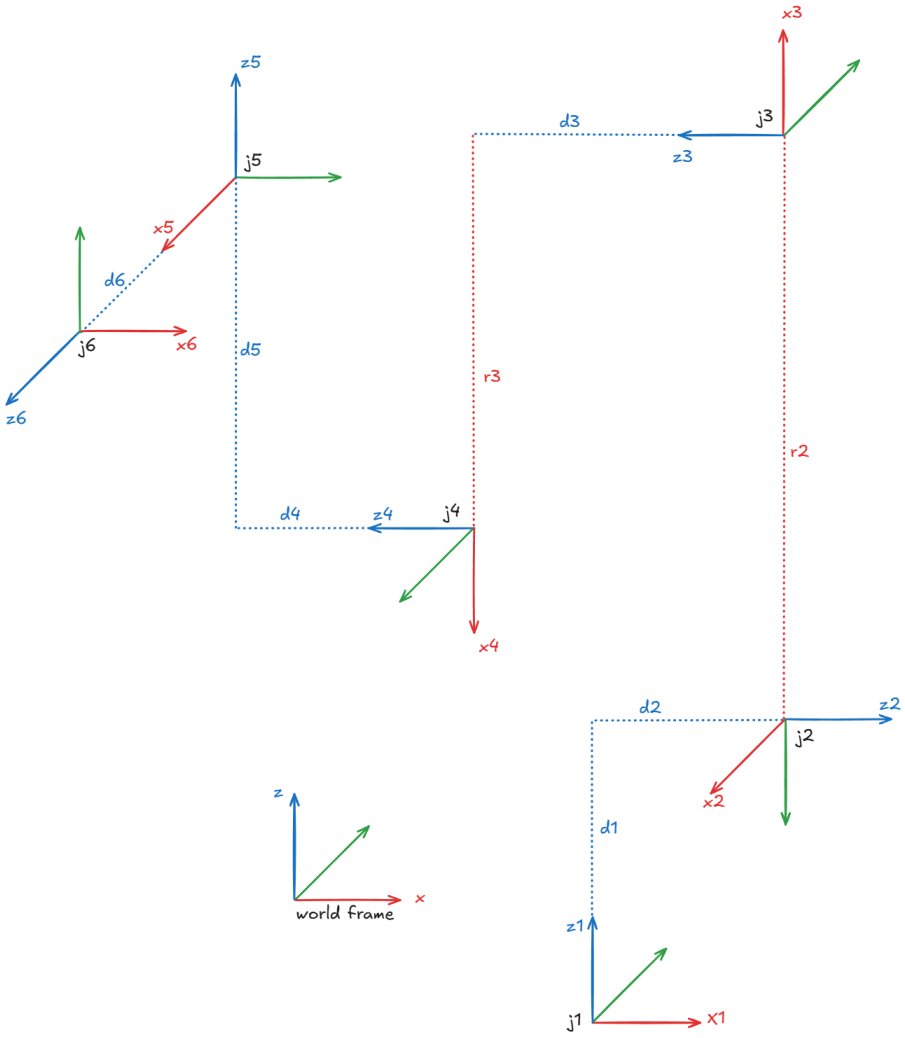
The robot has a circular footprint with a 600mm diameter, and the base/storage system sits approximately 720mm above the floor. The arm, when collapsed has a height of 900mm (above the base), and a maximum extension of 1850mm (above the base). The total overall height of the robot, at full extension is 2750mm, and has a sweeping radius (with arm extended fully perpendicular) of approximately 1950mm.

CAD Model

The team modeled the robot and other assets in SolidWorks and exported in URDF format. Model files are included with this report.



Frame Assignments



DH Parameters

(Deg.)	θ,z	α,x	d	r
J1	-90.000	-90.000	50.000	0.000
J2	-90.000	180.000	50.000	850.000
J3	180.000	0.000	100.000	750.000
J4	90.000	-90.000	50.000	0.000
J5	90.000	90.000	200.000	0.000
J6	0.000	0.000	50.000	0.000

Forward Kinematics

Note 1: The code in this section was based on a team member's work product from an individual HW2/3 submission. To protect the integrity of the homework assignments, code was not shared with other team members until after the due date of HW3.

Note 2: The code in this section focuses on the robot arm/manipulator. A similar approach is used for the movement of the robot base.

The following code creates a single parametric transformation matrix for a given joint identifier (e.g., "θ1"). This code is used multiple times to create parametric matrices for each joint.

```
def create_parametric_dh_matrix(joint_id):  1 usage
    j = str(joint_id)
    theta = smp.symbols('θ' + j)
    alpha = smp.symbols('α' + j)
    d = smp.symbols('d' + j)
    r = smp.symbols('r' + j)

    r11 = smp.cos(theta)
    r12 = -smp.sin(theta) * smp.cos(alpha)
    r13 = smp.sin(theta) * smp.sin(alpha)
    r14 = r * smp.cos(theta)
    r21 = smp.sin(theta)
    r22 = smp.cos(theta) * smp.cos(alpha)
    r23 = -smp.cos(theta) * smp.sin(alpha)
    r24 = r * smp.sin(theta)
    r31 = 0
    r32 = smp.sin(alpha)
    r33 = smp.cos(alpha)
    r34 = d
    transform_matrix = [
        [r11, r12, r13, r14],
        [r21, r22, r23, r24],
        [r31, r32, r33, r34],
        [0, 0, 0, 1]
    ]
    return smp.Matrix(transform_matrix)
```

The following code performs matrix multiplication on the parametric transform matrices and returns a list of each joint's complete transform. For example, the third element (second if considering base 0 indexing) in this list will have the complete transform of joint 3.

```
def prebuild_transforms(link_count, dh_table):  2 usages
    transforms = []
    transform = identity_matrix(4)
    transforms.append(transform.copy())
    for i in range(link_count):
        transform_i = create_parametric_dh_matrix(i + 1)
        transform = transform * transform_i
        transforms.append(pre_eval_dh_transforms(transform.copy(), dh_table))
    return transforms
```

Lastly, the following code uses the transform for joint 6 (e.g., the 6th element of the transform list from the previous step) to return the forward kinematic equation for the arm's end effector. Please note, this position is relative to the robot's mobile base since we consider the base and arm to function as two independent robots. It also does not account for additional transformations performed by the gripper assembly.

```
def forward_position_kinematics(transform):  2 usages
    x = transform[0, 3]  # [1 - 1, 4 - 1]
    y = transform[1, 3]  # [2 - 1, 4 - 1]
    z = transform[2, 3]  # [3 - 1, 4 - 1]
    Rx = smp.atan2( *args: transform[2, 1], transform[2, 2])
    e1 = (transform[0, 0]) ** 2  # [1 - 1, 1 - 1]
    e2 = (transform[1, 0]) ** 2  # [2 - 1, 1 - 1]
    Ry = smp.atan2( *args: -1 * transform[2, 0], smp.sqrt(e1 + e2))
    Rz = smp.atan2( *args: transform[1, 0], transform[0, 0])
    q_function_vector = smp.Matrix([x, y, z, Rx, Ry, Rz])
    return q_function_vector
```

Below is the output from the above `forward_position_kinematics()` solution. It is a parameterized version of `{x,y,z,Rx,Ry,Rz}`. The values for α, d, and r have been resolved since those are constant values in the kinematic chain.

```
Forward Position Kinematics:

Matrix([

[50.0*((-(-6.12323399573677e-17*sin(01)*sin(02) + cos(01)*cos(02))*sin(03) + (6.12323399573677e-17*sin(01)*cos(02) - 1.22464679914735e-16*sin(01) + 1.0*sin(02)*cos(01))*cos(03))*sin(04) + ((-6.12323399573677e-17
*sin(01)*sin(02) + cos(01)*cos(02))*cos(03) + (6.12323399573677e-17*sin(01)*cos(02) - 1.22464679914735e-16*sin(01) + 1.0*sin(02)*cos(01))*sin(03))*cos(04))*sin(05) + 200.0*(-(-6.12323399573677e-17*sin(01)*sin(02)
+ cos(01)*cos(02))*sin(03) + (6.12323399573677e-17*sin(01)*cos(02) - 1.22464679914735e-16*sin(01) + 1.0*sin(02)*cos(01))*cos(03))*cos(04) - 200.0*((-6.12323399573677e-17*sin(01)*sin(02) + cos(01)*cos(02))*cos(03)
+ (6.12323399573677e-17*sin(01)*cos(02) - 1.22464679914735e-16*sin(01) + 1.0*sin(02)*cos(01))*sin(03))*sin(04) + 750.0*(-6.12323399573677e-17*sin(01)*sin(02) + cos(01)*cos(02))*cos(03) + 750.0
*(6.12323399573677e-17*sin(01)*cos(02) - 1.22464679914735e-16*sin(01) + 1.0*sin(02)*cos(01))*sin(03) - 50.0*(6.12323399573677e-17*(-(-6.12323399573677e-17*sin(01)*sin(02) + cos(01)*cos(02))*sin(03) +
(6.12323399573677e-17*sin(01)*cos(02) - 1.22464679914735e-16*sin(01) + 1.0*sin(02)*cos(01))*cos(03))*cos(04) - 6.12323399573677e-17*((-6.12323399573677e-17*sin(01)*sin(02) + cos(01)*cos(02))*cos(03) +
(6.12323399573677e-17*sin(01)*cos(02) - 1.22464679914735e-16*sin(01) + 1.0*sin(02)*cos(01))*sin(03))*sin(04) - 7.49879891330929e-33*(sin(01)*cos(02) - 1.0*sin(01) - 1.22464679914735e-16*sin(02)*cos(01))*cos(05) -
5.20474889637625e-14*sin(01)*sin(02) + 1.12481983699639e-30*sin(01)*cos(02) + 100.0*sin(01) + 1.83697019872103e-14*sin(02)*cos(01) + 850.0*cos(01)*cos(02)],

[50.0*(((sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*sin(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*cos(03))*sin(04) + ((sin(01)*cos(02) +
6.12323399573677e-17*sin(02)*cos(01))*cos(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*sin(03))*cos(04))*sin(05) + 200.0*(-(sin(01)*cos(02) +
6.12323399573677e-17*sin(02)*cos(01))*sin(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*cos(03))*cos(04) - 200.0*((sin(01)*cos(02) + 6.12323399573677e-17
*sin(02)*cos(01))*cos(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*sin(03))*sin(04) + 750.0*(sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*cos(03) +
750.0*(1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*sin(03) - 50.0*(6.12323399573677e-17*(-(sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*sin(03) + (1.0
*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*cos(03))*cos(04) - 6.12323399573677e-17*((sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*cos(03) + (1.0
*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*sin(03))*sin(04) - 1.22464679914735e-16*sin(01)*sin(02) + 7.49879891330929e-33*cos(01)*cos(02) + 1.0*cos(01))*cos(05) +
1.83697019872103e-14*sin(01)*sin(02) + 850.0*sin(01)*cos(02) + 5.20474889637625e-14*sin(02)*cos(01) - 1.12481983699639e-30*cos(01)*cos(02) - 100.0*cos(01)],

[50.0*(((1.0*cos(02) + 7.49879891330929e-33)*sin(03) - 1.0*sin(02)*cos(03))*cos(04) + ((1.0*cos(02) + 7.49879891330929e-33)*cos(03) + 1.0*sin(02)*sin(03))*sin(04))*sin(05) - 200.0*(1.0*cos(02) +
7.49879891330929e-33)*sin(03) - 1.0*sin(02)*cos(03))*sin(04) + 200.0*((1.0*cos(02) + 7.49879891330929e-33)*cos(03) + 1.0*sin(02)*sin(03))*cos(04) + 750.0*(1.0*cos(02) + 7.49879891330929e-33)*sin(03) - 50.0
*(-6.12323399573677e-17*((1.0*cos(02) + 7.49879891330929e-33)*sin(03) - 1.0*sin(02)*cos(03))*sin(04) + 6.12323399573677e-17*((1.0*cos(02) + 7.49879891330929e-33)*cos(03) + 1.0*sin(02)*sin(03))*cos(04) -
1.22464679914735e-16*cos(02) + 6.12323399573677e-17)*cos(05) - 750.0*sin(02)*cos(03) - 850.0*sin(02) + 1.83697019872103e-14*cos(02) + 50.0],

[atan2(-((((1.0*cos(02) + 7.49879891330929e-33)*sin(03) - 1.0*sin(02)*cos(03))*cos(04) + ((1.0*cos(02) + 7.49879891330929e-33)*cos(03) + 1.0*sin(02)*sin(03))*sin(04))*cos(05) + (-6.12323399573677e-17*((1.0*cos(02)
+ 7.49879891330929e-33)*sin(03) - 1.0*sin(02)*cos(03))*sin(04) + 6.12323399573677e-17*((1.0*cos(02) + 7.49879891330929e-33)*cos(03) + 1.0*sin(02)*sin(03))*cos(04) - 1.22464679914735e-16*cos(02) +
6.12323399573677e-17)*sin(05))*sin(06) - (-6.12323399573677e-17*(((1.0*cos(02) + 7.49879891330929e-33)*sin(03) - 1.0*sin(02)*cos(03))*cos(04) + ((1.0*cos(02) + 7.49879891330929e-33)*cos(03) + 1.0
*sin(02)*sin(03))*sin(04))*cos(05) - (-6.12323399573677e-17*((1.0*cos(02) + 7.49879891330929e-33)*sin(03) - 1.0*sin(02)*cos(03))*sin(04) + 6.12323399573677e-17*((1.0*cos(02) + 7.49879891330929e-33)*cos(03) + 1.0
*sin(02)*sin(03))*cos(04) - 1.22464679914735e-16*cos(02) + 6.12323399573677e-17)*cos(05) + 7.49879891330929e-33*cos(02) - 3.74939945665464e-33)*cos(06), 1.0*(((1.0*cos(02) + 7.49879891330929e-33)*sin(03) - 1.0
*sin(02)*cos(03))*cos(04) + ((1.0*cos(02) + 7.49879891330929e-33)*cos(03) + 1.0*sin(02)*sin(03))*sin(04))*sin(05) - 6.12323399573677e-17*((1.0*cos(02) + 7.49879891330929e-33)*sin(03) - 1.0*sin(02)*cos(03))*sin(04)
+ 6.12323399573677e-17*((1.0*cos(02) + 7.49879891330929e-33)*cos(03) + 1.0*sin(02)*sin(03))*cos(04) - 1.0*(-6.12323399573677e-17*((1.0*cos(02) + 7.49879891330929e-33)*sin(03) - 1.0*sin(02)*cos(03))*sin(04) +
6.12323399573677e-17*((1.0*cos(02) + 7.49879891330929e-33)*cos(03) + 1.0*sin(02)*sin(03))*cos(04) - 1.22464679914735e-16*cos(02) + 6.12323399573677e-17)*cos(05) + 4.59169004331693e-49*sin(02)*cos(02) -
2.29584502165847e-49)],

[atan2(-((((1.0*cos(02) + 7.49879891330929e-33)*sin(03) - 1.0*sin(02)*cos(03))*cos(04) + ((1.0*cos(02) + 7.49879891330929e-33)*cos(03) + 1.0*sin(02)*sin(03))*sin(04))*cos(05) + (-6.12323399573677e-17*((1.0*cos(02)
+ 7.49879891330929e-33)*sin(03) - 1.0*sin(02)*cos(03))*sin(04) + 6.12323399573677e-17*((1.0*cos(02) + 7.49879891330929e-33)*cos(03) + 1.0*sin(02)*sin(03))*cos(04) - 1.22464679914735e-16*cos(02) +
6.12323399573677e-17)*sin(05))*cos(06) - (-6.12323399573677e-17*(((1.0*cos(02) + 7.49879891330929e-33)*sin(03) - 1.0*sin(02)*cos(03))*cos(04) + ((1.0*cos(02) + 7.49879891330929e-33)*cos(03) + 1.0
*sin(02)*sin(03))*sin(04))*cos(05) - (-6.12323399573677e-17*((1.0*cos(02) + 7.49879891330929e-33)*sin(03) - 1.0*sin(02)*cos(03))*sin(04) + 6.12323399573677e-17*((1.0*cos(02) + 7.49879891330929e-33)*cos(03)
+ 1.0*sin(02)*sin(03))*cos(04) - 1.22464679914735e-16*cos(02) + 6.12323399573677e-17)*cos(05) + 7.49879891330929e-33*cos(02) - 3.74939945665464e-33)*sin(06), sqrt((((-(-6.12323399573677e-17*sin(01)*sin(02) +
cos(01)*cos(02))*sin(03) + (6.12323399573677e-17*sin(01)*cos(02) - 1.22464679914735e-16*sin(01) + 1.0*sin(02)*cos(01))*cos(03))*sin(04) + ((-6.12323399573677e-17*sin(01)*sin(02) + cos(01)*cos(02))*cos(03) +
(6.12323399573677e-17*sin(01)*cos(02) - 1.22464679914735e-16*sin(01) + 1.0*sin(02)*cos(01))*sin(03))*cos(04))*sin(05) + (6.12323399573677e-17*(-(-6.12323399573677e-17*sin(01)*sin(02) + cos(01)*cos(02))*sin(03) +
(6.12323399573677e-17*sin(01)*cos(02) - 1.22464679914735e-16*sin(01) + 1.0*sin(02)*cos(01))*cos(03))*cos(04) - 6.12323399573677e-17*((-6.12323399573677e-17*sin(01)*sin(02) + cos(01)*cos(02))*cos(03) +
(6.12323399573677e-17*sin(01)*cos(02) - 1.22464679914735e-16*sin(01) + 1.0*sin(02)*cos(01))*sin(03))*sin(04) - 7.49879891330929e-33*sin(01)*cos(02) + 1.0*sin(01) - 1.22464679914735e-16*sin(02)*cos(01))*cos(05) +
4.59169004331693e-49*sin(01)*sin(02) + 1.22464679914735e-16*cos(01))**2 + ((((sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*sin(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16
*cos(01))*cos(03))*sin(04) + ((sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*cos(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16
*cos(01))*sin(03))*cos(04))*cos(05) + (6.12323399573677e-17*(-(sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*sin(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16
*cos(01))*cos(03))*cos(04) - 6.12323399573677e-17*((sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*cos(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16
*cos(01))*sin(03))*sin(04) - 1.22464679914735e-16*sin(01)*sin(02) + 7.49879891330929e-33*cos(01)*cos(02) + 1.0*cos(01))*sin(05) + 4.59169004331693e-49*cos(01)*cos(02) -
6.12323399573677e-17*sin(01)*sin(02) + 1.22464679914735e-16*cos(01))*sin(06))**2))],

[atan2(((((sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*sin(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*cos(03))*sin(04) + ((sin(01)*cos(02) +
6.12323399573677e-17*sin(02)*cos(01))*cos(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*sin(03))*cos(04))*cos(05) + (6.12323399573677e-17*(-(sin(01)*cos(02) +
6.12323399573677e-17*sin(02)*cos(01))*sin(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*cos(03))*cos(04) - 6.12323399573677e-17*((sin(01)*cos(02) +
6.12323399573677e-17*sin(02)*cos(01))*cos(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*sin(03))*sin(04) - 1.22464679914735e-16*sin(01)*sin(02) +
7.49879891330929e-33*cos(01)*cos(02) + 1.0*cos(01))*sin(05) + 4.59169004331693e-49*sin(01)*cos(02) + 6.12323399573677e-17*sin(01)*sin(02) - 1.22464679914735e-16*cos(01))*cos(06) + (-6.12323399573677e-17*((-(sin(01)*cos(02) +
6.12323399573677e-17*sin(02)*cos(01))*sin(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*cos(03))*cos(04) - 1.0*((sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*sin(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*cos(03))*sin(04) + ((-(sin(01)*cos(02) +
6.12323399573677e-17*sin(02)*cos(01))*cos(03) + (6.12323399573677e-17*sin(02)*cos(01) + cos(01))*sin(03))*cos(04) - 6.12323399573677e-17*(-(sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*sin(03) +
(1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*cos(03))*sin(04) - 7.49879891330929e-33*sin(01)*cos(02) + 1.0*sin(01) - 1.22464679914735e-16*sin(02)*cos(01))*sin(05) +
4.59169004331693e-49*sin(01)*sin(02) + 1.22464679914735e-16*cos(01))*sin(06))**2))],

[atan2(((((-(sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*sin(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*cos(03))*sin(04) + ((sin(01)*cos(02) +
6.12323399573677e-17*sin(02)*cos(01))*cos(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*sin(03))*cos(04))*cos(05) + (6.12323399573677e-17*(-(sin(01)*cos(02) +
6.12323399573677e-17*sin(02)*cos(01))*sin(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*cos(03))*sin(04) + ((sin(01)*cos(02) +
6.12323399573677e-17*sin(02)*cos(01))*cos(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*sin(03))*cos(04))*sin(05) + 1.0*(-(sin(01)*cos(02) +
6.12323399573677e-17*sin(02)*cos(01))*cos(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*sin(03))*cos(04) + 1.0*(-(sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*sin(03) + (1.0*sin(01)*sin(02) -
6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*cos(03))*sin(04) + 6.12323399573677e-17*(((-(sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*sin(03) + (1.0*sin(01)*sin(02) -
6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*cos(03))*sin(04) + ((-(sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*cos(03) + (6.12323399573677e-17*sin(02)*cos(01) + cos(01))*sin(03))*cos(04))*cos(05) +
(6.12323399573677e-17*(-(sin(01)*cos(02) + 6.12323399573677e-17*sin(02)*cos(01))*sin(03) + (1.0*sin(01)*sin(02) - 6.12323399573677e-17*cos(01)*cos(02) + 1.22464679914735e-16*cos(01))*cos(03))*sin(04) + ((-(sin(01)*cos(02) +
6.12323399573677e-17*sin(02)*cos(01))*cos(03) + (6.12323399573677e-17*sin(02)*cos(01) + cos(01))*sin(03))*cos(04))*sin(05) + 7.49879891330929e-33*sin(01)*cos(02) + 1.0*sin(01) - 1.22464679914735e-16*sin(02)*cos(01))*cos(06), (((-(-6.12323399573677e-17*sin(01)*sin(02) + cos(01)*cos(02))*sin(03) + (6.12323399573677e-17*sin(01)*cos(02) - 1.22464679914735e-16*sin(01) + 1.0*sin(02)*cos(01))*cos(03))*sin(04) + ((-6.12323399573677e-17*sin(01)*sin(02) + cos(01)*cos(02))*cos(03) +
(6.12323399573677e-17*sin(01)*cos(02) - 1.22464679914735e-16*sin(01) + 1.0*sin(02)*cos(01))*sin(03))*cos(04))*cos(05) + (6.12323399573677e-17*(-(-6.12323399573677e-17*sin(01)*sin(02) + cos(01)*cos(02))*sin(03) +
(6.12323399573677e-17*sin(01)*cos(02) - 1.22464679914735e-16*sin(01) + 1.0*sin(02)*cos(01))*cos(03))*cos(04) - 6.12323399573677e-17*((-6.12323399573677e-17*sin(01)*sin(02) + cos(01)*cos(02))*cos(03) +
(6.12323399573677e-17*sin(01)*cos(02) - 1.22464679914735e-16*sin(01) + 1.0*sin(02)*cos(01))*sin(03))*sin(04) - 1.22464679914735e-16*sin(01)*sin(02) + 7.49879891330929e-33*cos(01)*cos(02) + 1.0*cos(01))*sin(05) +
4.59169004331693e-49*sin(01)*cos(02) + 6.12323399573677e-17*sin(01) + 7.49879891330929e-33*sin(02)*cos(01))*sin(06))**2))],

[0]

])
```

Inverse Kinematics

Note 1: The code in this section was based on a team member's work product from an individual HW2/3 submission. To protect the integrity of the homework assignments, code was not shared with other team members until after the due date of HW3.

Note 2: The code in this section focuses on the robot arm/manipulator. A similar approach is used for the movement of the robot base.

The following code derives the linear elements of the Jacobian matrix using the partial derivatives method. The output of this function is a 3x6 matrix, three rows representing the x, y, and z functions, and six columns representing partial derivatives with respect to joints θ1 through θ6.

```
def jacobian_linear_functions(q_function_vector, link_count):  1 usage
    x_row = []
    y_row = []
    z_row = []
    for i in range(link_count):
        theta = 'θ' + str(i + 1)
        x_row.append(smp.diff(q_function_vector[0],  *symbols: theta))
        y_row.append(smp.diff(q_function_vector[1],  *symbols: theta))
        z_row.append(smp.diff(q_function_vector[2],  *symbols: theta))
    return smp.Matrix([x_row, y_row, z_row])
```

Using the transforms list (produced by `prebuild_transforms()`) in the Forward Kinematics section, we can substitute theta values and extract the angular rows of the Jacobian. The output from this function is a 3x6 matrix, with each column representing the incremental z values from the transformation matrices.

```
def jacobian_angular_values(transforms, dh_table, joint_values):  1 usage
    z_matrix = []
    for i in range(len(joint_values)):
        transform_evaluated = theta_subber(transforms[i + 1], dh_table, joint_values)
        z1 = transform_evaluated[0, 2]  # [1 - 1, 3 - 1]
        z2 = transform_evaluated[1, 2]  # [2 - 1, 3 - 1]
        z3 = transform_evaluated[2, 2]  # [3 - 1, 3 - 1]
        z_matrix.append([z1, z2, z3])
    jw = smp.Matrix(z_matrix).transpose()
    return jw
```

The complete and evaluated Jacobian can be resolved by stacking the linear and angular submatrices together and substituting the current joint values.

```
def jacobian_evaluated(jl_function, dh_table, joint_values, transforms):  1 usage
    jl_val = theta_subber2(jl_function, dh_table, joint_values)
    jw_val = jacobian_angular_values(transforms, dh_table, joint_values)
    j_combined = smp.Matrix.vstack( *args: jl_val, jw_val)
    return j_combined
```

The following is the output of the 3x6 linear matrix representing the top half of the Jacobian Matrix produced by the `jacobian_linear_functions()` code above.

```
Jacobian Linear Functions:

Matrix([

[50.0*((-(-sin(θ1)*cos(θ2) - 6.123233995736677e-17*sin(θ2)*cos(θ1))*sin(θ3) + (-1.0*sin(θ1)*sin(θ2) + 6.123233995736677e-17*cos(θ1)*cos(θ2) - 1.22464679914735e-16*cos(θ1))*cos(θ3))*sin(θ4) + ((-sin(θ1)*cos(θ2) -
6.123233995736677e-17*sin(θ2)*cos(θ1))*cos(θ3) + (-1.0*sin(θ1)*sin(θ2) + 6.123233995736677e-17*cos(θ1)*cos(θ2) - 1.22464679914735e-16*cos(θ1))*sin(θ3))*cos(θ4))*sin(θ5) + 200.0*(-(-sin(θ1)*cos(θ2) -
6.123233995736677e-17*sin(θ2)*cos(θ1))*sin(θ3) + (-1.0*sin(θ1)*sin(θ2) + 6.123233995736677e-17*cos(θ1)*cos(θ2) - 1.22464679914735e-16*cos(θ1))*cos(θ3))*cos(θ4) - 200.0*((-sin(θ1)*cos(θ2) - 6.123233995736677e-17
*sin(θ2)*cos(θ1))*cos(θ3) + (-1.0*sin(θ1)*sin(θ2) + 6.123233995736677e-17*cos(θ1)*cos(θ2) - 1.22464679914735e-16*cos(θ1))*sin(θ3))*sin(θ4) + 750.0*(-sin(θ1)*cos(θ2) - 6.123233995736677e-17*sin(θ2)*cos(θ1))*cos(θ3) +
750.0*(-1.0*sin(θ1)*sin(θ2) + 6.123233995736677e-17*cos(θ1)*cos(θ2) - 1.22464679914735e-16*cos(θ1))*sin(θ3) - 50.0*(6.123233995736677e-17*(-(-sin(θ1)*cos(θ2) - 6.123233995736677e-17*sin(θ2)*cos(θ1))*sin(θ3) + (-1.0
*sin(θ1)*sin(θ2) + 6.123233995736677e-17*cos(θ1)*cos(θ2) - 1.22464679914735e-16*cos(θ1))*cos(θ3))*cos(θ4) - 6.123233995736677e-17*((-sin(θ1)*cos(θ2) - 6.123233995736677e-17*sin(θ2)*cos(θ1))*cos(θ3) + (-1.0
*sin(θ1)*sin(θ2) + 6.123233995736677e-17*cos(θ1)*cos(θ2) - 1.22464679914735e-16*cos(θ1))*sin(θ3))*sin(θ4) + 1.22464679914735e-16*sin(θ1)*sin(θ2) - 7.498798913309e-33*cos(θ1)*cos(θ2) - 1.0*cos(θ1))*cos(θ5) -
1.836970198721003e-14*sin(θ1)*sin(θ2) - 850.0*sin(θ1)*cos(θ2) - 5.204748896376625e-14*sin(θ2)*cos(θ1) + 1.124819836996639e-30*cos(θ1)*cos(θ2) + 100.0*cos(θ1),

50.0*(((-6.123233995736677e-17*sin(θ1)*sin(θ2) + 1.0*cos(θ1)*cos(θ2))*sin(θ3) + (-6.123233995736677e-17*sin(θ1)*cos(θ2) - sin(θ2)*cos(θ1))*cos(θ3))*cos(θ4) + ((-6.123233995736677e-17*sin(θ1)*sin(θ2) + 1.0
*cos(θ1)*cos(θ2))*cos(θ3) - (-6.123233995736677e-17*sin(θ1)*cos(θ2) - sin(θ2)*cos(θ1))*sin(θ3))*sin(θ4))*sin(θ5) - 200.0*((-6.123233995736677e-17*sin(θ1)*sin(θ2) + 1.0*cos(θ1)*cos(θ2))*sin(θ3) +
(-6.123233995736677e-17*sin(θ1)*cos(θ2) - sin(θ2)*cos(θ1))*cos(θ3))*sin(θ4) + 200.0*((-6.123233995736677e-17*sin(θ1)*sin(θ2) + 1.0*cos(θ1)*cos(θ2))*cos(θ3) - (-6.123233995736677e-17*sin(θ1)*cos(θ2) -
sin(θ2)*cos(θ1))*sin(θ3))*cos(θ4) + 750.0*(-6.123233995736677e-17*sin(θ1)*sin(θ2) + 1.0*cos(θ1)*cos(θ2))*sin(θ3) + 750.0*(-6.123233995736677e-17*sin(θ1)*cos(θ2) - sin(θ2)*cos(θ1))*cos(θ3) - 50.0
*(-6.123233995736677e-17*((-6.123233995736677e-17*sin(θ1)*sin(θ2) + 1.0*cos(θ1)*cos(θ2))*sin(θ3) + (-6.123233995736677e-17*sin(θ1)*cos(θ2) - sin(θ2)*cos(θ1))*cos(θ3))*sin(θ4) + 6.123233995736677e-17
*((-6.123233995736677e-17*sin(θ1)*sin(θ2) + 1.0*cos(θ1)*cos(θ2))*cos(θ3) - (-6.123233995736677e-17*sin(θ1)*cos(θ2) - sin(θ2)*cos(θ1))*sin(θ3))*cos(θ4) + 7.498798913309e-33*sin(θ1)*sin(θ2) - 1.22464679914735e-16
*cos(θ1)*cos(θ2))*cos(θ5) - 1.124819836996639e-30*sin(θ1)*sin(θ2) - 5.204748896376625e-14*sin(θ1)*cos(θ2) - 850.0*sin(θ2)*cos(θ1) + 1.836970198721003e-14*cos(θ1)*cos(θ2),

-50.0*(-6.123233995736677e-17*(-(-6.123233995736677e-17*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*sin(θ3) + (6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*cos(θ3))*sin(θ4) +
6.123233995736677e-17*(-(-6.123233995736677e-17*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*cos(θ3) - (6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*sin(θ3))*cos(θ4))*cos(θ5) +
50.0*((-(-6.123233995736677e-17*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*sin(θ3) + (6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*cos(θ3))*cos(θ4) + (-(-6.123233995736677e-17
*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*cos(θ3) + (6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*sin(θ3))*sin(θ4))*sin(θ5) + 200.0*(-(-6.123233995736677e-17*sin(θ1)*sin(θ2)
+ cos(θ1)*cos(θ2))*sin(θ3) + (6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*cos(θ3))*sin(θ4) + 200.0*(-(-6.123233995736677e-17*sin(θ1)*sin(θ2) +
cos(θ1)*cos(θ2))*cos(θ3) - (6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*sin(θ3))*cos(θ4) - 750.0*(-6.123233995736677e-17*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*sin(θ3) +
750.0*(6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*cos(θ3),

-50.0*(-6.123233995736677e-17*(-(-6.123233995736677e-17*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*sin(θ3) + (6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*cos(θ3))*sin(θ4) -
6.123233995736677e-17*((-6.123233995736677e-17*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*cos(θ3) + (6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*sin(θ3))*cos(θ4))*cos(θ5) +
50.0*((-(-6.123233995736677e-17*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*sin(θ3) + (6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*cos(θ3))*cos(θ4) - ((-6.123233995736677e-17
*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*cos(θ3) + (6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*sin(θ3))*sin(θ4))*cos(θ5) + 50.0*(6.123233995736677e-17*(-
*((-6.123233995736677e-17*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*cos(θ3) + (6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*sin(θ3))*sin(θ4) - 7.498798913309e-33
*sin(θ1)*cos(θ2) - 1.0*sin(θ1) - 1.224646799147356-16*sin(θ2)*cos(θ1))*sin(θ5),

0],

[50.0*((-(-6.123233995736677e-17*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*sin(θ3) + (6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*cos(θ3))*sin(θ4) + ((-6.123233995736677e-17
*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*cos(θ3) + (6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*sin(θ3))*cos(θ4))*cos(θ5) + 50.0*(-(-6.123233995736677e-17*sin(θ1)*sin(θ2)
+ cos(θ1)*cos(θ2))*sin(θ3) + (6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*sin(θ3))*cos(θ4) - 200.0*((-6.123233995736677e-17*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*cos(θ3)
+ (6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*sin(θ3))*cos(θ4) - 200.0*(-(-6.123233995736677e-17*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*cos(θ3) + 750.0
*(6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*sin(θ3) - 50.0*(6.123233995736677e-17*(-(-6.123233995736677e-17*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*sin(θ3) +
(6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*cos(θ3))*cos(θ4) - 6.123233995736677e-17*((-6.123233995736677e-17*sin(θ1)*sin(θ2) + cos(θ1)*cos(θ2))*cos(θ3) +
(6.123233995736677e-17*sin(θ1)*cos(θ2) - 1.22464679914735e-16*sin(θ1) + 1.0*sin(θ2)*cos(θ1))*sin(θ3))*sin(θ4) - 7.498798913309e-33*sin(θ1)*cos(θ2) - 1.0*sin(θ1) - 1.22464679914735e-16*sin(θ2)*cos(θ1))*cos(θ5) -
5.204748896376625e-14*sin(θ1)*sin(θ2) + 1.124819836996639e-30*sin(θ1)*cos(θ2) + 100.0*sin(θ1) + 1.836970198721003e-14*sin(θ2)*cos(θ1) + 850.0*cos(θ1)*cos(θ2),

50.0*((-(-sin(θ1)*sin(θ2) + 6.123233995736677e-17*cos(θ1)*cos(θ2))*sin(θ3) + (1.0*sin(θ1)*cos(θ2) + 6.123233995736677e-17*sin(θ2)*cos(θ1))*cos(θ3))*sin(θ4) + ((-sin(θ1)*sin(θ2) + 6.123233995736677e-17
*cos(θ1)*cos(θ2))*cos(θ3) + (1.0*sin(θ1)*cos(θ2) + 6.123233995736677e-17*sin(θ2)*cos(θ1))*sin(θ3))*cos(θ4))*sin(θ5) + 200.0*(-(-sin(θ1)*sin(θ2) + 6.123233995736677e-17*cos(θ1)*cos(θ2))*sin(θ3) + (1.0*sin(θ1)*cos(θ2)
+ 6.123233995736677e-17*sin(θ2)*cos(θ1))*cos(θ3))*cos(θ4) - 200.0*((-sin(θ1)*sin(θ2) + 6.123233995736677e-17*cos(θ1)*cos(θ2))*cos(θ3) + (1.0*sin(θ1)*cos(θ2) + 6.123233995736677e-17*sin(θ2)*cos(θ1))*sin(θ3))*sin(θ4) +
750.0*(-sin(θ1)*sin(θ2) + 6.123233995736677e-17*cos(θ1)*cos(θ2))*cos(θ3) + 750.0*(1.0*sin(θ1)*cos(θ2) + 6.123233995736677e-17*sin(θ2)*cos(θ1))*sin(θ3) - 50.0*(6.123233995736677e-17*(-(-sin(θ1)*sin(θ2) +
6.123233995736677e-17*cos(θ1)*cos(θ2))*sin(θ3) + (1.0*sin(θ1)*cos(θ2) + 6.123233995736677e-17*sin(θ2)*cos(θ1))*cos(θ3))*cos(θ4) - 6.123233995736677e-17*((-sin(θ1)*sin(θ2) + 6.123233995736677e-17
*cos(θ1)*cos(θ2))*cos(θ3) + (1.0*sin(θ1)*cos(θ2) + 6.123233995736677e-17*sin(θ2)*cos(θ1))*sin(θ3))*cos(θ4) - 7.498798913309e-33*sin(θ2)*cos(θ1))*cos(θ5) - 850.0
*sin(θ1)*cos(θ2) + 1.836970198721003e-14*sin(θ1)*cos(θ2) + 1.124819836996639e-30*sin(θ2)*cos(θ1) + 5.204748896376625e-14*cos(θ1)*cos(θ2),

-50.0*(-6.123233995736677e-17*(-(sin(θ1)*cos(θ2) + 6.123233995736677e-17*sin(θ2)*cos(θ1))*sin(θ3) + (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*cos(θ3))*sin(θ4) +
6.123233995736677e-17*(-(sin(θ1)*cos(θ2) + 6.123233995736677e-17*sin(θ2)*cos(θ1))*cos(θ3) - (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*sin(θ3))*cos(θ4))*cos(θ5) +
50.0*((-(sin(θ1)*cos(θ2) + 6.123233995736677e-17*sin(θ2)*cos(θ1))*sin(θ3) + (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*cos(θ3))*cos(θ4) + (-(sin(θ1)*cos(θ2) +
6.123233995736677e-17*sin(θ2)*cos(θ1))*cos(θ3) + (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*sin(θ3))*sin(θ4))*sin(θ5) - 200.0*(-(sin(θ1)*cos(θ2) +
6.123233995736677e-17*sin(θ2)*cos(θ1))*cos(θ3) + (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*sin(θ3))*cos(θ4) + 200.0*((sin(θ1)*cos(θ2) + 6.123233995736677e-17
*sin(θ2)*cos(θ1))*cos(θ3) + (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*sin(θ3))*sin(θ4) - 750.0*(sin(θ1)*cos(θ2) + 6.123233995736677e-17*sin(θ2)*cos(θ1))*sin(θ3) +
750.0*(1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*sin(θ3),

-50.0*(-6.123233995736677e-17*(-(sin(θ1)*cos(θ2) + 6.123233995736677e-17*sin(θ2)*cos(θ1))*sin(θ3) + (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*cos(θ3))*sin(θ4) -
6.123233995736677e-17*((sin(θ1)*cos(θ2) + 6.123233995736677e-17*sin(θ2)*cos(θ1))*cos(θ3) + (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*sin(θ3))*cos(θ4))*cos(θ5) + 50.0
*((-(sin(θ1)*cos(θ2) + 6.123233995736677e-17*sin(θ2)*cos(θ1))*sin(θ3) + (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*cos(θ3))*cos(θ4) - ((sin(θ1)*cos(θ2) +
6.123233995736677e-17*sin(θ2)*cos(θ1))*cos(θ3) + (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*sin(θ3))*sin(θ4))*sin(θ5) - 200.0*(-(sin(θ1)*cos(θ2) +
6.123233995736677e-17*sin(θ2)*cos(θ1))*cos(θ3) + (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*sin(θ3))*cos(θ4) + 200.0*((sin(θ1)*cos(θ2) + 6.123233995736677e-17
*sin(θ2)*cos(θ1))*cos(θ3) + (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*sin(θ3))*sin(θ4),

50.0*((-(sin(θ1)*cos(θ2) + 6.123233995736677e-17*sin(θ2)*cos(θ1))*sin(θ3) + (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*cos(θ3))*sin(θ4) + ((sin(θ1)*cos(θ2) +
6.123233995736677e-17*sin(θ2)*cos(θ1))*cos(θ3) + (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*sin(θ3))*cos(θ4))*cos(θ5) + 50.0*(6.123233995736677e-17*(-(sin(θ1)*cos(θ2)
+ 6.123233995736677e-17*sin(θ2)*cos(θ1))*cos(θ3) + (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*sin(θ3))*cos(θ4) - 6.123233995736677e-17*((sin(θ1)*cos(θ2) +
6.123233995736677e-17*sin(θ2)*cos(θ1))*cos(θ3) + (1.0*sin(θ1)*sin(θ2) - 6.123233995736677e-17*cos(θ1)*cos(θ2) + 1.22464679914735e-16*cos(θ1))*sin(θ3))*sin(θ4) - 1.22464679914735e-16*sin(θ1)*sin(θ2) +
7.498798913309e-33*cos(θ1)*cos(θ2) + 1.0*cos(θ1))*sin(θ5),

0],

[0,

50.0*((-1.0*sin(θ2)*sin(θ3) - 1.0*cos(θ2)*cos(θ3))*cos(θ4) + (-1.0*sin(θ2)*cos(θ3) + 1.0*sin(θ3)*cos(θ2))*sin(θ4))*sin(θ5) - 200.0*(-1.0*sin(θ2)*sin(θ3) - 1.0*cos(θ2)*cos(θ3))*sin(θ4) + 200.0*(-1.0*sin(θ2)*cos(θ3)
+ 1.0*sin(θ3)*cos(θ2))*cos(θ4) - 50.0*(-6.123233995736677e-17*(-1.0*sin(θ2)*sin(θ3) - 1.0*cos(θ2)*cos(θ3))*sin(θ4) + 6.123233995736677e-17*(-1.0*sin(θ2)*cos(θ3) + 1.0*sin(θ3)*cos(θ2))*cos(θ4) + 1.22464679914735e-16
*sin(θ2))*cos(θ5) - 750.0*sin(θ2)*sin(θ3) - 1.836970198721003e-14*sin(θ2) - 750.0*cos(θ2)*cos(θ3) - 850.0*cos(θ2),

50.0*((-(1.0*cos(θ2) + 7.498798913309e-33)*sin(θ3) + 1.0*sin(θ2)*cos(θ3))*sin(θ4) + ((1.0*cos(θ2) + 7.498798913309e-33)*cos(θ3) + 1.0*sin(θ2)*sin(θ3))*cos(θ4))*sin(θ5) - 50.0*(6.123233995736677e-17*(-(1.0
*cos(θ2) + 7.498798913309e-33)*sin(θ3) + 1.0*sin(θ2)*cos(θ3))*cos(θ4) - 6.123233995736677e-17*((1.0*cos(θ2) + 7.498798913309e-33)*cos(θ3) + 1.0*sin(θ2)*sin(θ3))*sin(θ4))*cos(θ5) + 200.0*(-(1.0*cos(θ2) +
7.498798913309e-33)*sin(θ3) + 1.0*sin(θ2)*cos(θ3))*cos(θ4) - 200.0*((1.0*cos(θ2) + 7.498798913309e-33)*cos(θ3) + 1.0*sin(θ2)*sin(θ3))*sin(θ4) + 750.0*(1.0*cos(θ2) + 7.498798913309e-33)*cos(θ3) + 750.0
*sin(θ2)*sin(θ3),

50.0*(-((1.0*cos(θ2) + 7.498798913309e-33)*sin(θ3) - 1.0*sin(θ2)*cos(θ3))*sin(θ4) + ((1.0*cos(θ2) + 7.498798913309e-33)*cos(θ3) + 1.0*sin(θ2)*sin(θ3))*cos(θ4))*sin(θ5) - 50.0*(-6.123233995736677e-17*((1.0
*cos(θ2) + 7.498798913309e-33)*sin(θ3) - 1.0*sin(θ2)*cos(θ3))*cos(θ4) - 6.123233995736677e-17*((1.0*cos(θ2) + 7.498798913309e-33)*cos(θ3) + 1.0*sin(θ2)*sin(θ3))*sin(θ4))*cos(θ5) - 200.0*((1.0*cos(θ2) +
7.498798913309e-33)*sin(θ3) - 1.0*sin(θ2)*cos(θ3))*cos(θ4) - 200.0*((1.0*cos(θ2) + 7.498798913309e-33)*cos(θ3) + 1.0*sin(θ2)*sin(θ3))*sin(θ4),

50.0*(((1.0*cos(θ2) + 7.498798913309e-33)*sin(θ3) - 1.0*sin(θ2)*cos(θ3))*cos(θ4) + ((1.0*cos(θ2) + 7.498798913309e-33)*cos(θ3) + 1.0*sin(θ2)*sin(θ3))*sin(θ4))*cos(θ5) + 50.0*(-6.123233995736677e-17*((1.0
*cos(θ2) + 7.498798913309e-33)*sin(θ3) - 1.0*sin(θ2)*cos(θ3))*sin(θ4) + 6.123233995736677e-17*((1.0*cos(θ2) + 7.498798913309e-33)*cos(θ3) + 1.0*sin(θ2)*sin(θ3))*cos(θ4) - 1.22464679914735e-16*cos(θ2) +
6.123233995736677e-17)*sin(θ5),

0]])
```

[THE REMAINDER OF THIS PAGE IS INTENTIONALLY BLANK]

Forward Kinematics Validation

Geometric validation of the forward kinematics chain was performed using transformation matrices in an Excel spread sheet. This methodology was adapted from a YouTube video by Chris Annin of Annin Robotics (see references). A copy of the excel file is included with this report.

	IN (Deg)
J1	0.0000
J2	0.0000
J3	0.0000
J4	0.0000
J5	0.0000
J6	0.0000

(Deg.)	θ,z	α,x	d	r
J1	-90.000	-90.000	50.000	0.000
J2	-90.000	180.000	50.000	850.000
J3	180.000	0.000	100.000	750.000
J4	90.000	-90.000	50.000	0.000
J5	90.000	90.000	200.000	0.000
J6	0.000	0.000	50.000	0.000

(RAD.)	TOOL
X	0.000
Y	0.000
Z	0.000
α	0.000
β	0.000
γ	0.000

OUT (Deg)	
X	-100.000
Y	-50.000
Z	350.000
R_z	0.000
R_y	0.000
R_x	90.000

Using this "spreadsheet model" we can change joint values in isolation and confirm the output is as expected.

Rotating J1 by 90 degrees should rotate the robot's arm around the Z axis. As we can see in the table below, the z value of the output does not change, the Rz value rotates by 90 degrees, and the absolute x/y values swap values. Ry and Rx do not change. These are strong indicators that the DH table is accurate.

	IN (Deg)
J1	90.0000
J2	0.0000
J3	0.0000
J4	0.0000
J5	0.0000
J6	0.0000

(Deg.)	θ,z	α,x	d	r
J1	-90.000	-90.000	50.000	0.000
J2	-90.000	180.000	50.000	850.000
J3	180.000	0.000	100.000	750.000
J4	90.000	-90.000	50.000	0.000
J5	90.000	90.000	200.000	0.000
J6	0.000	0.000	50.000	0.000

(RAD.)	TOOL
X	0.000
Y	0.000
Z	0.000
α	0.000
β	0.000
γ	0.000

OUT (Deg)	
X	50.000
Y	-100.000
Z	350.000
R_z	90.000
R_y	0.000
R_x	90.000

For the next validation, we can rotate J2 and J3 by the same angle. With the design of this manipulator, if J2 and J3 are rotated by the same value, we should see the end effector move away and down from the arm's origin, without any rotational changes. As shown in the table below, compared with the at-rest position, there is an absolute value increase in y (outward motion), a decrease in z (downward motion), and no change in x (lateral motion). Also, there was no change in orientation of the end effector (Rz, Ry, Rx).

	IN (Deg)
J1	0.0000
J2	33.0000
J3	33.0000
J4	0.0000
J5	0.0000
J6	0.0000

(Deg.)	θ,z	α,x	d	r
J1	-90.000	-90.000	50.000	0.000
J2	-90.000	180.000	50.000	850.000
J3	180.000	0.000	100.000	750.000
J4	90.000	-90.000	50.000	0.000
J5	90.000	90.000	200.000	0.000
J6	0.000	0.000	50.000	0.000

(RAD.)	TOOL
X	0.000
Y	0.000
Z	0.000
α	0.000
β	0.000
γ	0.000

OUT (Deg)	
X	-100.000
Y	-512.943
Z	212.870
R_z	0.000
R_y	0.000
R_x	90.000

Joints J3 and J4 can be rotated together for a similar validation as J2 and J3. However, this will cause the end effector to rise above the origin instead of lower. Below, we can see the output is as expected: No change in x or in orientation. The z value has increased (upward motion), and the absolute y value has also increased (outward motion).

	IN (Deg)
J1	0.0000
J2	0.0000
J3	66.0000
J4	-66.0000
J5	0.0000
J6	0.0000

(Deg.)	θ,z	α,x	d	r
J1	-90.000	-90.000	50.000	0.000
J2	-90.000	180.000	50.000	850.000
J3	180.000	0.000	100.000	750.000
J4	90.000	-90.000	50.000	0.000
J5	90.000	90.000	200.000	0.000
J6	0.000	0.000	50.000	0.000

(RAD.)	TOOL
X	0.000
Y	0.000
Z	0.000
α	0.000
β	0.000
γ	0.000

OUT (Deg)	
X	-100.000
Y	-735.159
Z	794.948
R_z	0.000
R_y	0.000
R_x	90.000

Below, J5 is rotated 45 degrees, which is reflected in the Rz rotation, and in changes in the x/y position of the end effector. No change to z or to other rotational axes is as expected.

	IN (Deg)
J1	0.0000
J2	0.0000
J3	0.0000
J4	0.0000
J5	45.0000
J6	0.0000

(Deg.)	θ,z	α,x	d	r
J1	-90.000	-90.000	50.000	0.000
J2	-90.000	180.000	50.000	850.000
J3	180.000	0.000	100.000	750.000
J4	90.000	-90.000	50.000	0.000
J5	90.000	90.000	200.000	0.000
J6	0.000	0.000	50.000	0.000

(RAD.)	TOOL
X	0.000
Y	0.000
Z	0.000
α	0.000
β	0.000
γ	0.000

OUT (Deg)	
X	-64.645
Y	-35.355
Z	350.000
R_z	45.000
R_y	0.000
R_x	90.000

Finally, J6 is rotated which, as expected, only affects the Ry value when the other joints are in the home position.

	IN (Deg)
J1	0.0000
J2	0.0000
J3	0.0000
J4	0.0000
J5	0.0000
J6	45.0000

(Deg.)	θ,z	α,x	d	r
J1	-90.000	-90.000	50.000	0.000
J2	-90.000	180.000	50.000	850.000
J3	180.000	0.000	100.000	750.000
J4	90.000	-90.000	50.000	0.000
J5	90.000	90.000	200.000	0.000
J6	0.000	0.000	50.000	0.000

(RAD.)	TOOL
X	0.000
Y	0.000
Z	0.000
α	0.000
β	0.000
γ	0.000

OUT (Deg)	
X	-100.000
Y	-50.000
Z	350.000
R_z	0.000
R_y	-45.000
R_x	90.000

[THE REMAINDER OF THIS PAGE IS INTENTIONALLY BLANK]

Inverse Kinematics Validation

Note 1: The code in this section was based on a team member's work product from an individual HW2/3 submission. To protect the integrity of the homework assignments, code was not shared with other team members until after the due date of HW3.

Note 2: The code in this section focuses on the robot arm/manipulator. A similar approach is used for the movement of the robot base.

For inverse kinematics validation, the team leveraged HW2/3's requirement to draw the rounded rectangle or "keystone" shape. Using this code was as simple as replacing the DH table values with values that represented the manipulator used in this project.

To account for singularities and other oddities when certain joints were near orthogonal positions, the team employed a Dampened Least Squares model. The code for this was derived from exercises and examples provided by Kreyszig (Advanced Engineering Mathematics), Lay (Linear Algebra and Its Applications), and Press (Numerical Recipes). It is important to note that this algorithm may not be implemented perfectly. However, it worked as expected, correcting for singularities, and was thus viable for this project.

```
def jacobian_inverse_dls(jacobian, lambda_factor):  1 usage
    identity = smp.eye(jacobian.shape[1])
    regularization = lambda_factor ** 2 * identity
    damped_inverse = (jacobian.T * jacobian + regularization).inv() * jacobian.T
    return damped_inverse
```

Using the Dampened Least Squares algorithm and a scaling lambda factor, the fully computed (i.e., numerical) inverse Jacobian can be used to validate inverse kinematics. Since this is a numerical inverse (due to computational limitations on deriving a parametric inverse), it must be computed iteratively for specific joint values.

```
# Jacobian and Jacobian Inverse
jj = jacobian_evaluated(jl_function, dh_table, joint_values, transforms)

jj_condition = np.linalg.cond(np.array(jj).astype(float))
verbose and print(f'Jacobian Condition Number: {jj_condition}')
effective_lambda = lambda_factor
if use_scaled_lambda:
    effective_lambda *= jj_condition
verbose and print(f'Lambda Factor: {lambda_factor}')

jj_inv = jacobian_inverse_dls(jj, effective_lambda)
```

New joint values can be calculated by numerically integrating the theta velocities with current joint values.

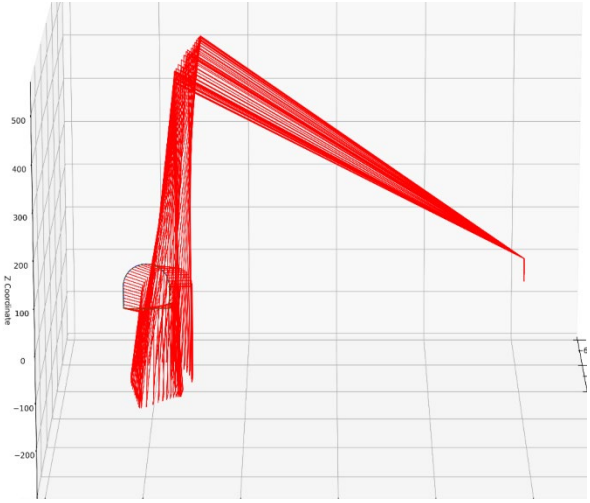
```
# thetas (joint values) and theta_dots
theta_dots = compute_theta_dots(jj_inv, q_dot_goal)
verbose and print(f'theta_dots: {round_matrix(theta_dots)}')

# add to plot data
path_data.append(q)
if time > path_times[0]:
    goal_data.append(q_goal)

# increment joint values for next loop
joint_values_history.append(joint_values.copy())
joint_values = increment_thetas(joint_values, theta_dots, dt)
verbose and print(f'New Joint Values: {round_matrix(smp.Matrix([joint_values]))}')
```

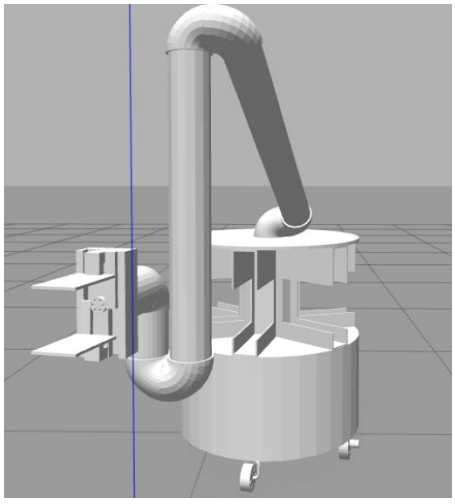
Use the methods described above, the path of the end effector and the projection of the arm can be plotted in 3D space. These can be visually compared to the model in Gazebo by nudging the joints until the model's joint values map to those in the 3D plots. Below are samples showing the full 3D plot versus the Gazebo model in its starting position.

Please note, the 3D plot and the Gazebo model will not perfectly align. This is due to two limitations. First, the two visualizations do not share synchronized camera positions. Therefore, parallax error and perspective distortion will cause the images to look different. Second, the 3D plot is limited to drawing lines from the origin of one link to the origin of the next link. Since in DH, the modeled origin does not always align with the actual origin, there are deviations for the intermediate links. This is notably visible for the offset from link1 to link2. In the model, there is clearly an offset in the join. However, in the plot, this offset is proportionately spread over the entire link.

The 3D plot of the end effector and the projection of the arm.

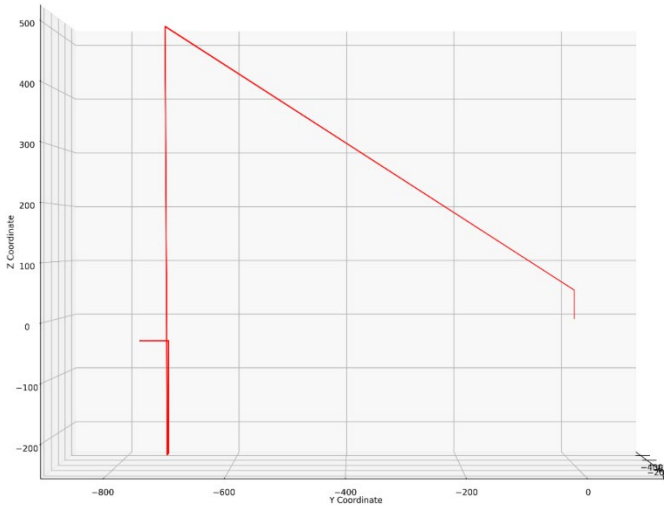
The Gazebo model is in the same starting position as used for the plot above. Starting joint values are approximately (0, 1, 1, 0, 0, 0) radians.

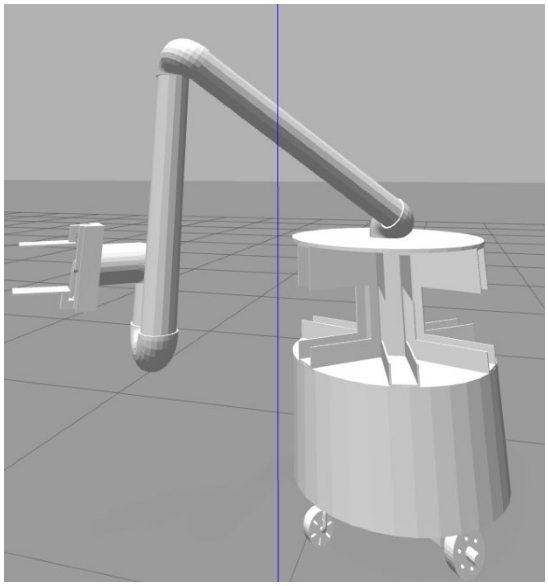




Side view of the plot of the starting joint values, (0, 1, 1, 0, 0, 0)radians. (This is just the starting position, not the entire keystone plot shown previously)

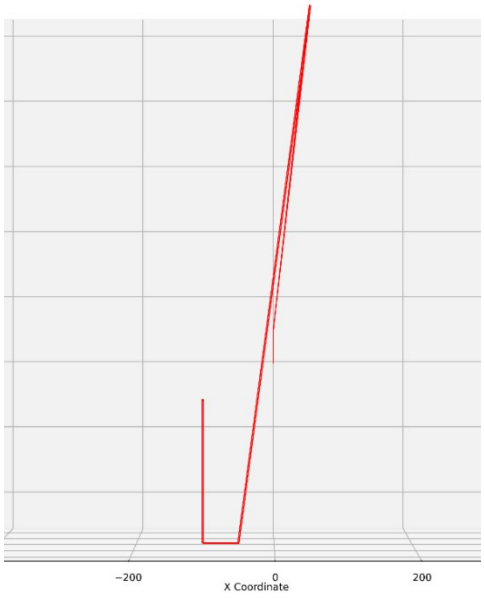
The Gazebo model is in the same starting position as used for the plot above.

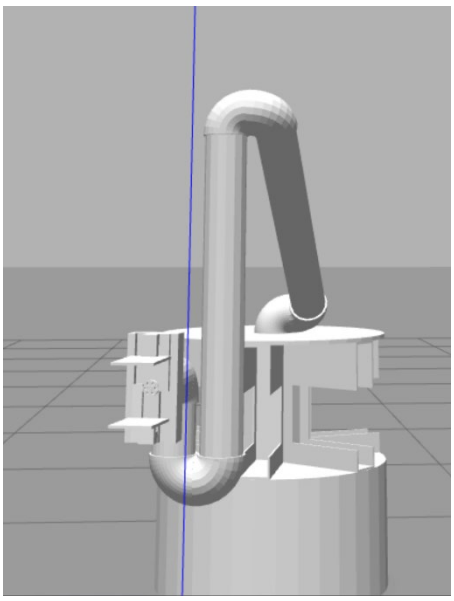




Forward view of the plot of the starting joint values, (0, 1, 1, 0, 0, 0)radians. (This is just the starting position, not the entire keystone plot shown previously)

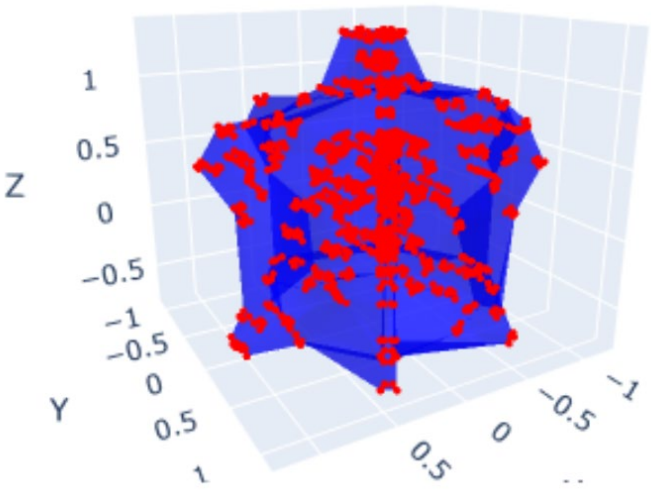
The Gazebo model is in the same starting position as used for the plot above.



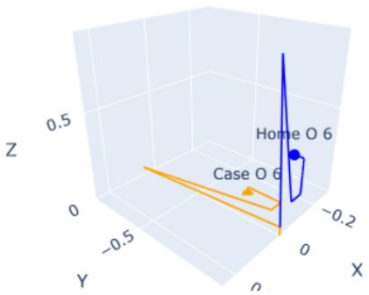


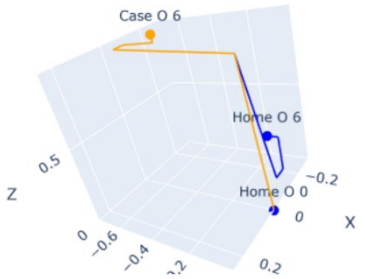
Workspace Analysis

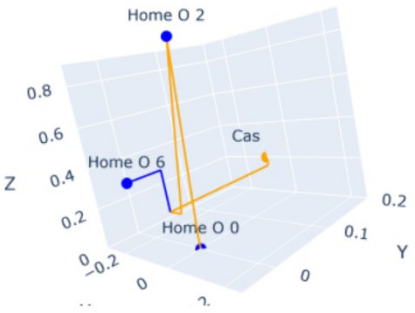
The robot workspace was assessed by limiting joint movement to +/- 90 degrees and running every combination of 4 linearly spaced joint angles from -90 to +90 degrees. The constraint is due to the possibility of collisions with the floor and could later be relaxed leveraging condition-based logic to avoid floor collisions. The plot (right) shows the end effector position at each point run and the shaded blue indicates possible positions for the end effector.

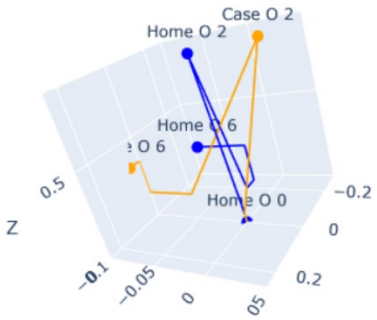


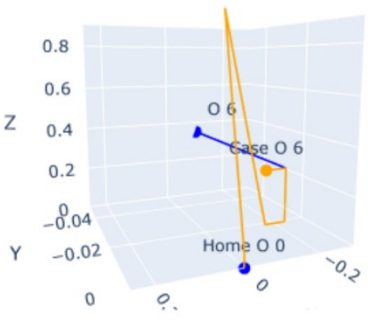
Below are additional plots of the robot's manipulator arm. These plots were generated for additional validation of the workspace and forward kinematic chain. The blue lines represent the manipulator arm at its rest position. The yellow lines represent the manipulator arm at θ=90degrees for a single joint.











Assumptions

The following assumptions were made for this project:

- Robot navigation will be simulated by following a pre-determined path. The robot will not determine a path based on identifying the home location of an arbitrary book.
- The robot will not use environmental feedback as part of its path planning. Course changes will be limited to adjustments made by the PID controller to compensate for trajectory errors. The robot will not adjust for path obstructions.
- The act of picking up a book will be simulated. Handing the dynamics associated with picking up an object in Gazebo are beyond the scope of this project.

Control Method

The robot has two control methods, a Teleop controller and an autonomous controller. The Teleop Controller provides forward, reverse and turning controls for the robot base, as well as individual controls for each joint and gripper. The Teleop controller reports joint positions that can be recorded for use in the autonomous controller.

The autonomous controller works by replaying recorded joint angles for the arm and using a PID controller to assist in steering the robot to a destination position. The autonomous controller receives its input in the form of ROS parameters.

- `goal {x, y}` parameter to send robot to specific location.
- `arm_goal {a, b, c, d, e}` parameter to recall position of each arm joint.
- `gripper_goal {a, b, c, d}` parameter to orient and manipulate the gripper.

Gazebo / RViz Visualization

A Gazebo demo using the autonomous controller can be found on YouTube, at the following link. The video includes demonstrating how to spawn the robot and bootstrap the controllers.

https://www.youtube.com/watch?v=lB9bhnnsFkw

- 0:00 - Spawn Gazebo World
- 0:30 - Boot Autonomous Controller
- 0:40 - Run Autonomous Demo Script
- 0:48 - DEMO

RViz was not used for this project.

Problems Faced

This project had an ambitious scope. Many topics were new to every team member. The team was required to perform a material amount of “hacking and slashing" to get things working in Gazebo, without fully understanding each software component. This resulted in no time for refinement or major changes.

The simulation had systemic fragility. There were too many sources for issues: tedious math, ROS and its ecosystem of plugins, multiple programming languages. This made it hard to narrow down problems and find root causes.

Time management was also challenging for this project. It was hard to balance time between learning mathematics modeling topics and spending time debugging ROS issues. The team was forced to accept best efforts over best outcomes.

Lastly, time pressures forced team members into fixed roles. There was no time or capacity for team members to switch roles. This unfortunately forced team members to work within their comfort zone.

Future Work

The following concepts could lead to future work or extensions of this project:

- Use LiDAR to adjust the robot’s position relative to the surrounding bookshelves when putting a book into its home location.
- Use cameras or environmental sensors to identify bookshelf capacity for the book in the gripper.
- Add waypoints to the environment and let the robot determine the best path from its current location to its intended destination.
- Develop a "reload" scenario for simulation. In this scenario the robot would load or reload books from its starting area and then proceed to put the additional book away in the library.

Conclusion

Overall, the Library Robot project was successful; many of the goals were met and the team was able to present a viable design for the intended application. The team was exposed to the challenges associated with using Gazebo's physics engine and gained experience in developing nodes for ROS. There were also auxiliary benefits for the team such as experience working with Docker and GitHub.

If there was an opportunity to rework the project, the team would like to have better synchronization between the DH-based model and the SolidWorks model. The team would also like to have better integration between the kinematics modeling and ROS modeling.

The team's most notable successes included:

- **DevOps:** Codified a portable runtime and dev environment which will be tremendously useful for future projects.
- **ROS:** Learned basic ROS concepts such as spawning a robot, creating nodes, topics, and parameters.
- **PID Controller:** Implemented a PID Controller.
- **Quaternion Math:** Worked with quaternion representations for robot orientation, and converting to roll-pitch-yaw, Euler angles, and Transform matrices.
- **Gazebo:** Gained experience working in Gazebo and parsing Gazebo state data.

Given the team's lack of experience with this subject matter, and the compressed schedule of the project, having a working Gazebo simulation can be considered a successful outcome.

GitHub Link

The code for this project can be found at: https://github.com/jamesfehrmann/enpm662p2

Please contact James Fehrmann at jsf@umd.edu or 202-285-8025 if the repository is not accessible.

The following steps can be used to run the demo:

- Clone the repository to `~/enpm662p2`, `~/cpp/enpm66p2`, or `~/projects/enpm662p2`. Cloning to another location will result in the repo not being mounted to the container.
- Run `init.sh` to build and launch the container. If the container is running but you are not connected, you can run `attach.sh` to connect. Reviewing the code of these bash scripts will reveal clues on how the project is configured to run in Docker.
- Once in the Docker container (and assuming the project files mounted correctly, as noted in the first step), run the following scripts:

    - `project2/terp2_build.sh` = build the project files
    - `project2/terp2_empty.sh` = run the gazebo simulation
    - `project2/terp2_teleop.sh` = run the teleop controller
    - `project2/terp2_controller.sh` = run the autonomous controller
    - `project2/terp2_library_nav.sh` = run the library demo (requires controller online)
- The `project2` directory includes other interesting scripts (for example, `set_goal.sh` and `set_arm.sh`) that interact with the autonomous controller.

Package Submission

See instructions under "GitHub Link" for how to run the demo or see the video in the Gazebo Simulation section.

[THE REMAINDER OF THIS PAGE IS INTENTIONALLY BLANK]

References

Annin, Chris. 6 Axis Robot...with the AR4-MK2. https://www.youtube.com/watch?v=FNuiNmoqaZM . Accessed Nov. 2024

cppreference.com. C++ Reference. https://en.cppreference.com/w/cpp. Accessed Oct. 2024.

Docker. Dockerfile Reference. https://docs.docker.com/reference/dockerfile/. Accessed Oct. 2024.

Eater. Quaternions. https://eater.net/quaternions. Accessed Oct. 2024.

Kreyszig, Erwin. Advanced Engineering Mathematics. 9th ed., Wiley, 2006.

Lay, Steven R., et al. Linear Algebra and Its Applications. 6th ed., Pearson, 2020.

Lynch, Kevin M., and Frank C. Park. Modern Robotics: Mechanics, Planning, and Control. Cambridge University Press, 2017.

Open Robotics. ROS 2 Documentation: Humble. https://docs.ros.org/en/humble/. Accessed Oct. 2024.

Parab, Shantanu. ENPM662 Introduction to Robot Modeling. https://enpm-662introduction-to-robotmodelling.readthedocs.io/en/latest/. Accessed Oct. 2024.

Press, William H., et al. Numerical Recipes, The Art of Scientific Computing. 3rd ed., Cambridge University Press, 2007

Prats, Mario & Martínez, Ester & Sanz, Pedro & del Pobil, Angel P. (2008). The UJI librarian robot. Intelligent Service Robotics. 1. 321-335. 10.1007/s11370-008-0028-1.

Python Software Foundation. The Python Language Reference. https://docs.python.org. Accessed Nov. 2024.

ros2_control Development Team. ros2_control documentation. https://control.ros.org/rolling/index.html. Accessed Oct. 2024.

Spong, Mark W., et al. Robot Modeling and Control. 2nd ed., Wiley, 2020.

Stewart, James. Calculus: Early Transcendentals. 9th ed., Cengage Learning, 2021.

SymPy Development Team. SymPy: Python Library for Symbolic Mathematics. https://www.sympy.org. Accessed Nov. 2024.

■

GROUP 1 – Project 2 Contribution Report

Class: ENPM662 - Introduction to Robot Modeling
Due: December 8, 2024

Prepared By: James Fehrmann (Team Lead)

Contribution Statement

Project 2 was a challenging project that required active and regular contributions by all team members. Every team member contributed to the success of the project and was engaged daily in discussion and in completing objectives.

Each team member was assigned an area of focus, which is outlined in both the final report and the in-class presentation.

Please contact James Fehrmann with any questions.

	IN (Deg)
J1	0.0000
J2	0.0000
J3	0.0000
J4	0.0000
J5	0.0000
J6	0.0000

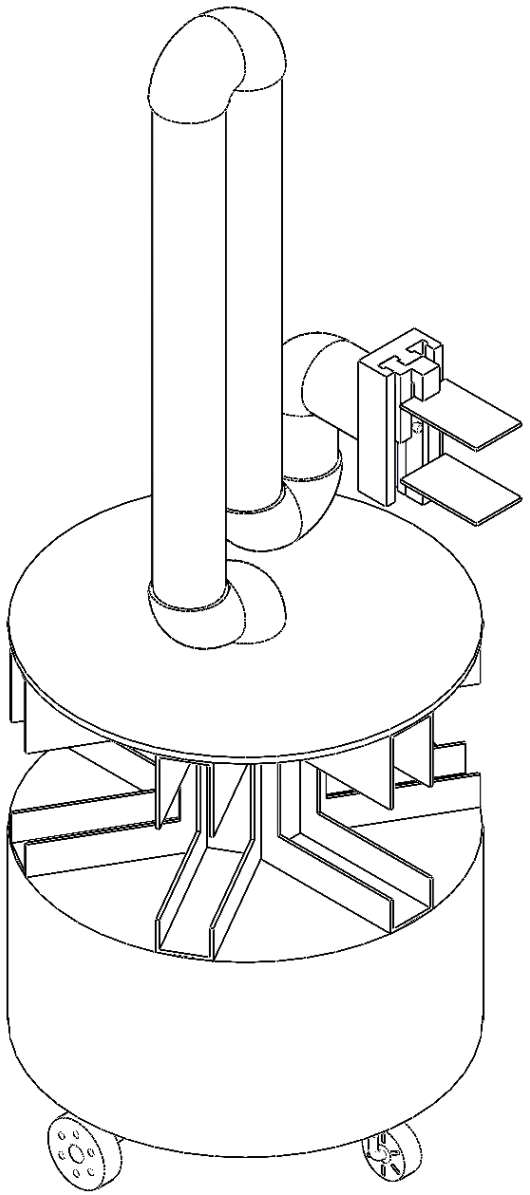
(Deg.)	θ,z	α,x	d	r
J1	-90.000	-90.000	50.000	0.000
J2	-90.000	180.000	50.000	850.000
J3	180.000	0.000	100.000	750.000
J4	90.000	-90.000	50.000	0.000
J5	90.000	90.000	200.000	0.000
J6	0.000	0.000	50.000	0.000

(RAD.)	TOOL
X	0.000
Y	0.000
Z	0.000
α	0.000
β	0.000
γ	0.000

OUT (Deg)	
X	-100.000
Y	-50.000
Z	350.000
R_z	0.000
R_y	0.000
R_x	90.000

θ	*-1.571*	J1	0.000	0.000	1.000	0.000	⇒	R0-1	0.000	0.000	1.000	0.000
a	*-1.571*		-1.000	0.000	0.000	0.000			-1.000	0.000	0.000	0.000
d	*50.000*		0.000	-1.000	0.000	50.000			0.000	-1.000	0.000	50.000
r	*0.000*		0.000	0.000	0.000	1.000			0.000	0.000	0.000	1.000
θ	*-1.571*	J2	0.000	-1.000	0.000	0.000	⇒	R0-2	0.000	0.000	-1.000	50.000
a	*3.142*		-1.000	0.000	0.000	-850.000			0.000	1.000	0.000	0.000
d	*50.000*		0.000	0.000	-1.000	50.000			1.000	0.000	0.000	900.000
r	*850.000*		0.000	0.000	0.000	1.000			0.000	0.000	0.000	1.000
θ	*3.142*	J3	-1.000	0.000	0.000	-750.000	⇒	R0-3	0.000	0.000	-1.000	-50.000
a	*0.000*		0.000	-1.000	0.000	0.000			0.000	-1.000	0.000	0.000
d	*100.000*		0.000	0.000	1.000	100.000			-1.000	0.000	0.000	150.000
r	*750.000*		0.000	0.000	0.000	1.000			0.000	0.000	0.000	1.000
θ	*1.571*	J4	0.000	0.000	-1.000	0.000	⇒	R0-4	0.000	1.000	0.000	-100.000
a	*-1.571*		1.000	0.000	0.000	0.000			-1.000	0.000	0.000	0.000
d	*50.000*		0.000	-1.000	0.000	50.000			0.000	0.000	1.000	150.000
r	*0.000*		0.000	0.000	0.000	1.000			0.000	0.000	0.000	1.000
θ	*1.571*	J5	0.000	0.000	1.000	0.000	⇒	R0-5	1.000	0.000	0.000	-100.000
a	*1.571*		1.000	0.000	0.000	0.000			0.000	0.000	-1.000	0.000
d	*200.000*		0.000	1.000	0.000	200.000			0.000	1.000	0.000	350.000
r	*0.000*		0.000	0.000	0.000	1.000			0.000	0.000	0.000	1.000
θ	*0.000*	J6	1.000	0.000	0.000	0.000	⇒	R0-6	1.000	0.000	0.000	-100.000
a	*0.000*		0.000	1.000	0.000	0.000			0.000	0.000	-1.000	-50.000
d	*50.000*		0.000	0.000	1.000	50.000			0.000	1.000	0.000	350.000
r	*0.000*		0.000	0.000	0.000	1.000			0.000	0.000	0.000	1.000
		TOOL	1.000	0.000	0.000	0.000	⇒	R0-T	1.000	0.000	0.000	-100.000
			0.000	1.000	0.000	0.000			0.000	0.000	-1.000	-50.000
			0.000	0.000	1.000	0.000			0.000	1.000	0.000	350.000
			0.000	0.000	0.000	1.000			0.000	0.000	0.000	1.000

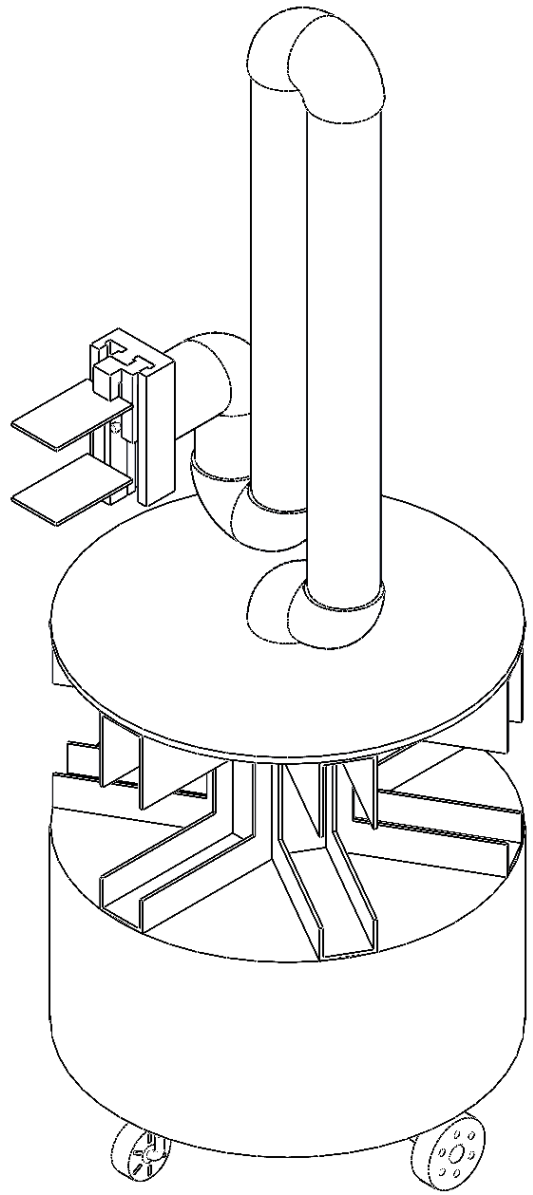
For reference, the in-class presentation file is included after this page



Project 2 – Group 1

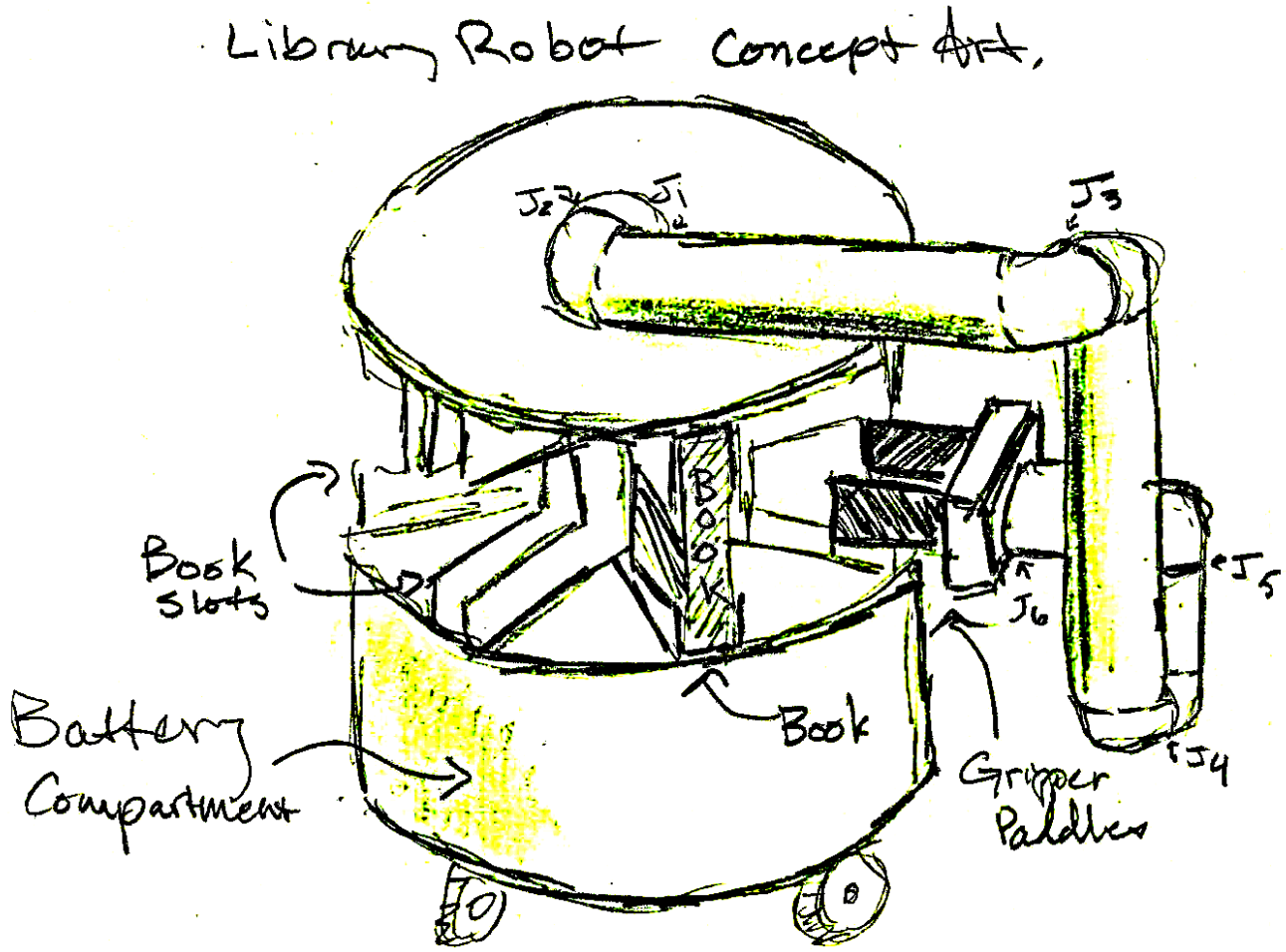
ENPM662 – Introduction to Robot Modeling

The Library Robot



Topics

- Team Members
- Robot Design
- ROS Nodes and Controllers
- Autonomous Mode
- Teleop Control
- Demo
- DH Table and Validation
- Deployment Model
- Challenges and Success

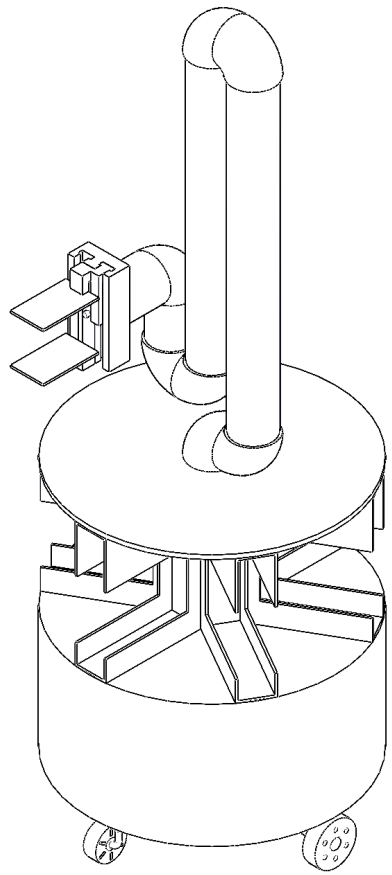


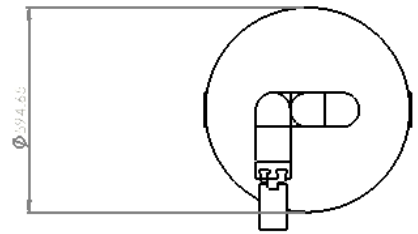
Team Members

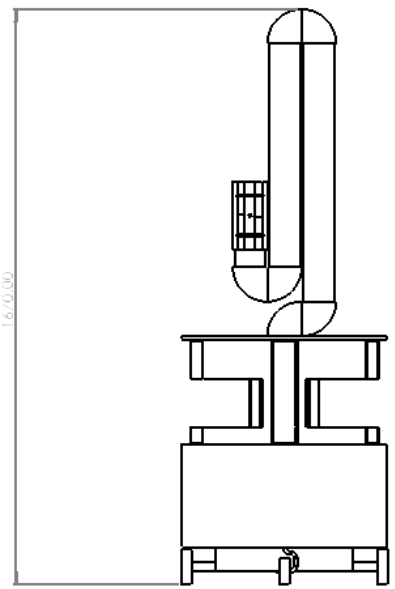
- **Anne-Michelle**, EE
  - ROS Development
  - Demo Orchestration
- **Chris Collins**, AE
  - ROS Development
  - World Building
- **Daniel Zinobile**, ME
  - Robot Design
  - SolidWorks Modeling
- **James Fehrmann**, CS
  - DevOps
  - ROS Development

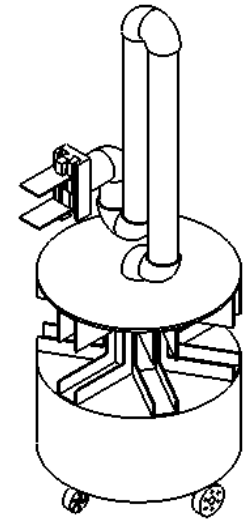
Collective ROS Knowledge as of September 1, 2024: **0 Hours**

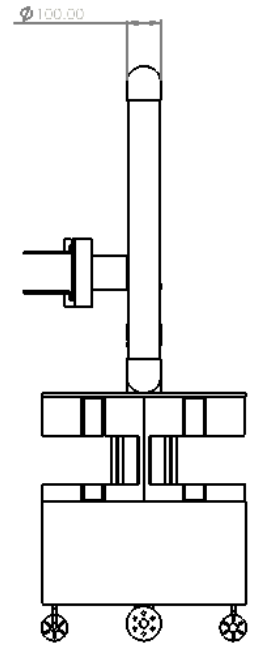
Robot Design

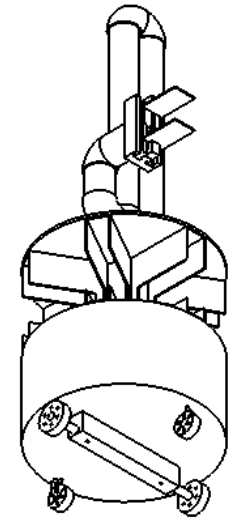












ROS Nodes and Controllers

- ***terp2*** launch file with nodes for robot state and control management. (python)
- ***terp2_controller*** node for autonomous mode control. (cpp)
- ***terp2_teleop*** node for manual control. (python)
- ***terp2_monitor*** node for continuous monitoring of pose data. (cpp)

Autonomous Mode

- terp2_controller designed to respond to various parameters
  - ***goal {x, y}*** parameter to send robot to specific location.

```
ros2 param set /terp2_controller goal "[${1}.0,${2}.0]"
```

  - ***arm_goal {a, b, c, d, e}*** parameter to recall position of each arm joint.

```
ros2 param set /terp2_controller arm_goal "[${1},${2},${3},${4},${5}]"
```

  - ***gripper_goal {a, b, c, d}*** parameter to orient and manipulate the gripper.

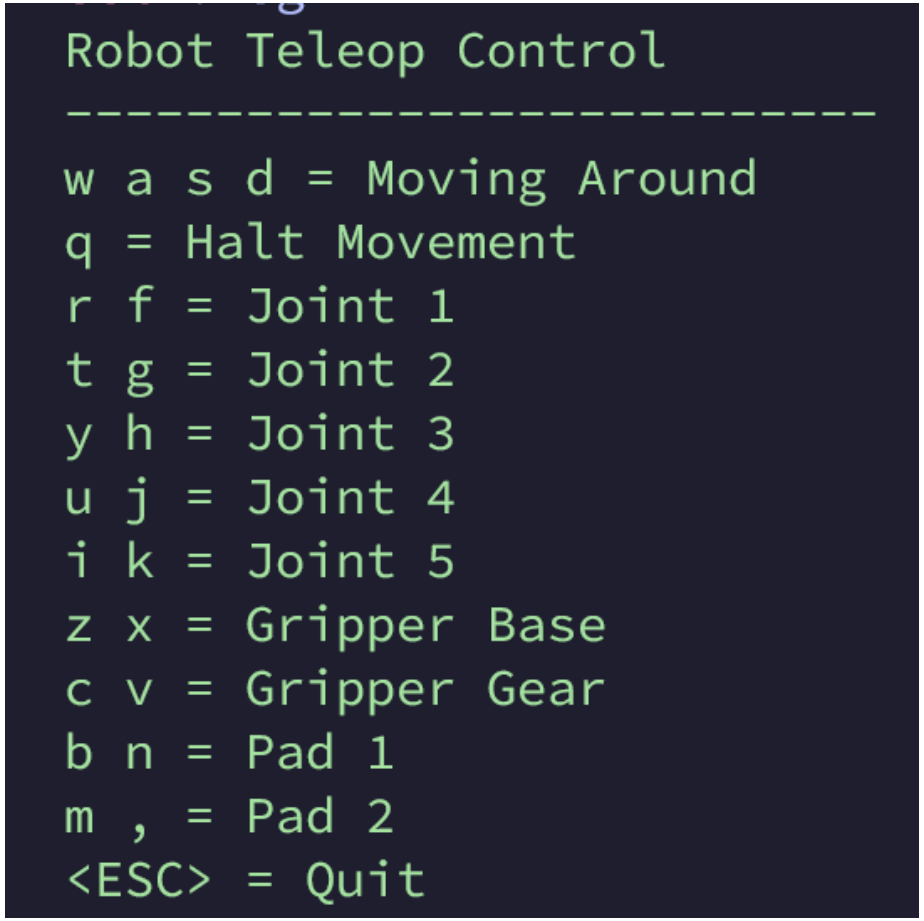
```
ros2 param set /terp2_controller gripper_goal "[${1},${2},${3},${4}]"
```

- parameter calls can be chained together in bash script files to orchestrate complex sequences.

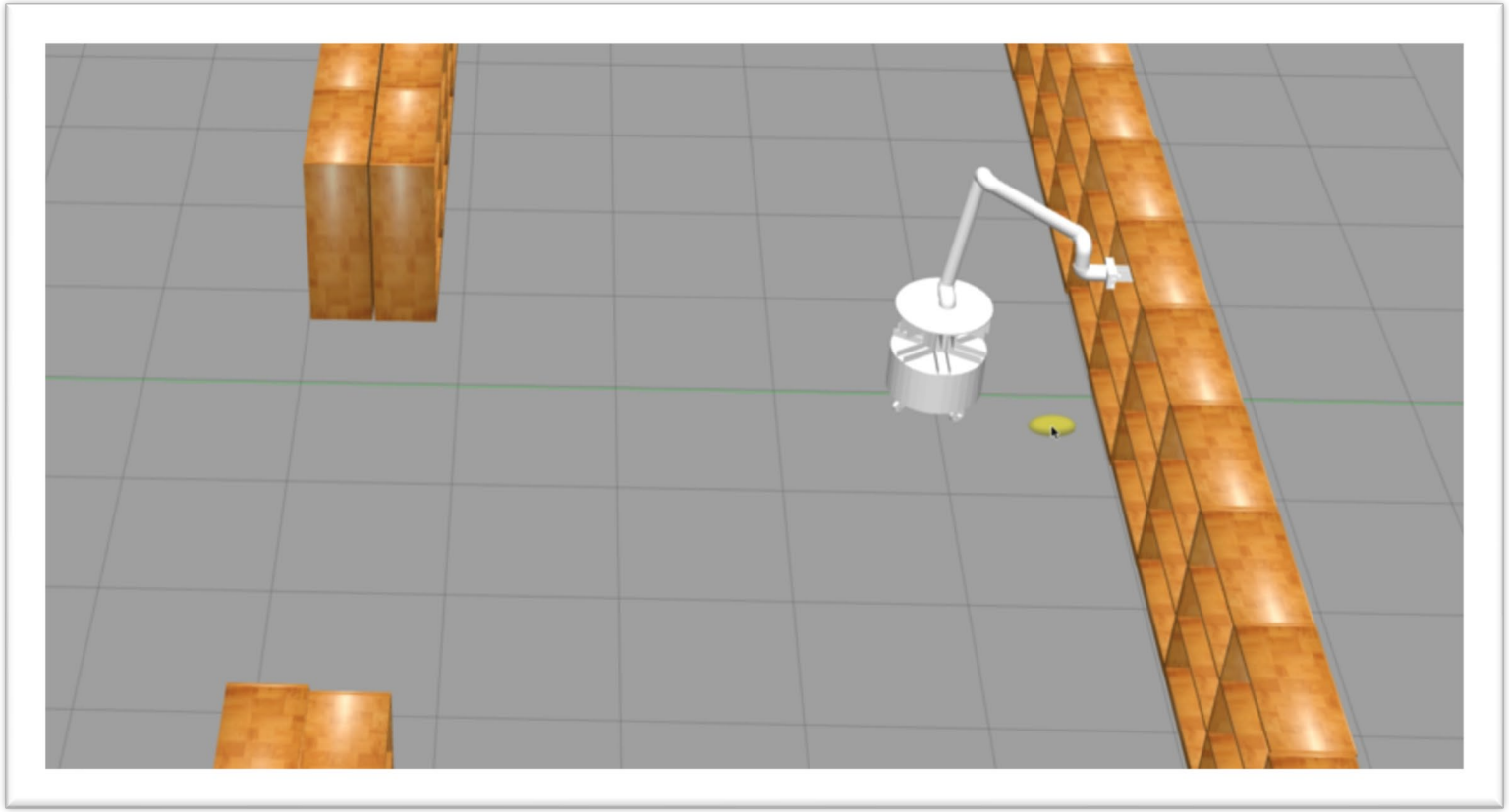
Teleop Control

```
[INFO] [1733022288.222921743] [keyboard_control_node]: Steer Angle: 0.0
[INFO] [1733022288.223137171] [keyboard_control_node]: Linear Velocity: 0.0
[INFO] [1733022288.223412576] [keyboard_control_node]: Joint Values: [0.0, 0.0, 0.0, 0.0, 0.0]
[INFO] [1733022288.223577523] [keyboard_control_node]: Joint Values (degrees): [0.0, 0.0, 0.0, 0.0, 0.0]
[INFO] [1733022288.223843124] [keyboard_control_node]: Gripper Values: [-1.5, 0.0, 0.0, 0.0]
```

- Pressing movement keys increments velocity or changes yaw orientation.
- Pressing joint and gripper controls step the joints forwards and backwards.
- Screen displays values that can be transcribed into autonomous mode script files.

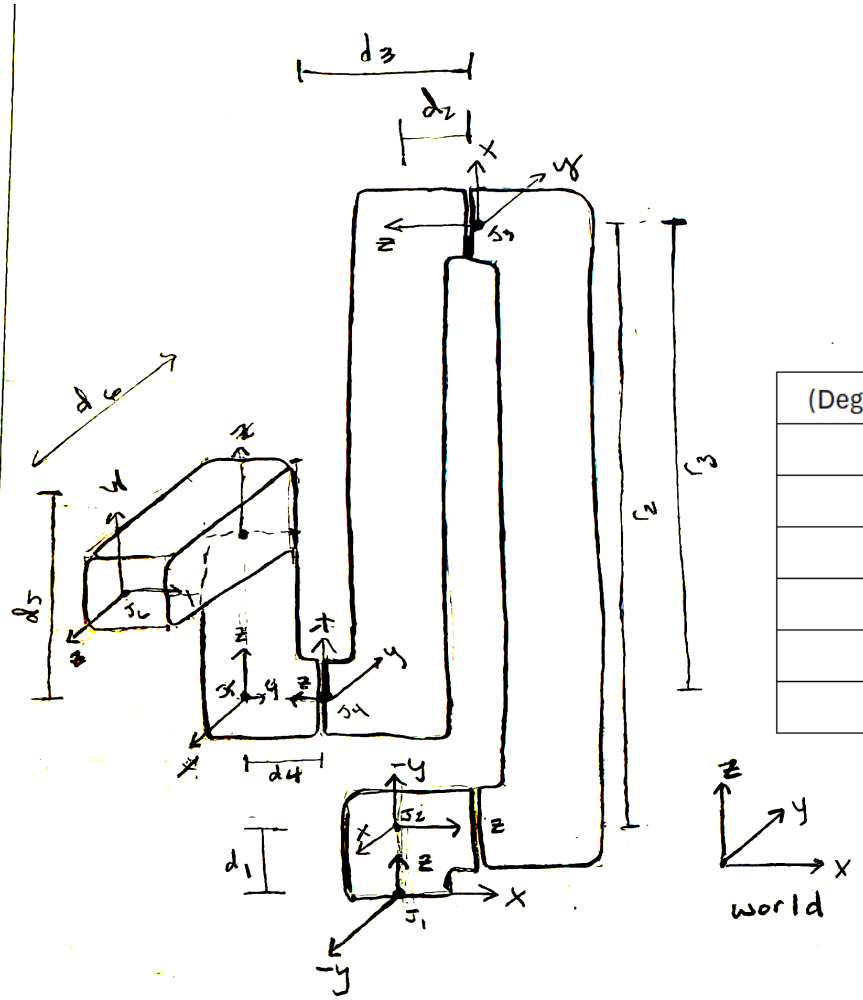


Demo

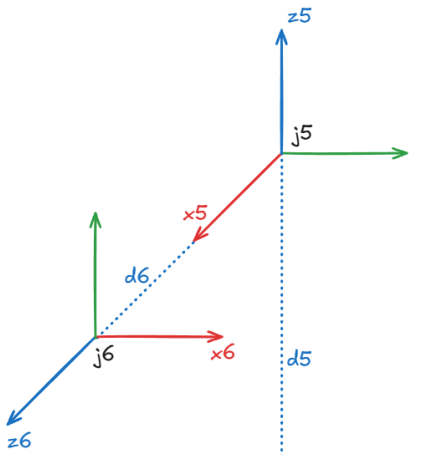


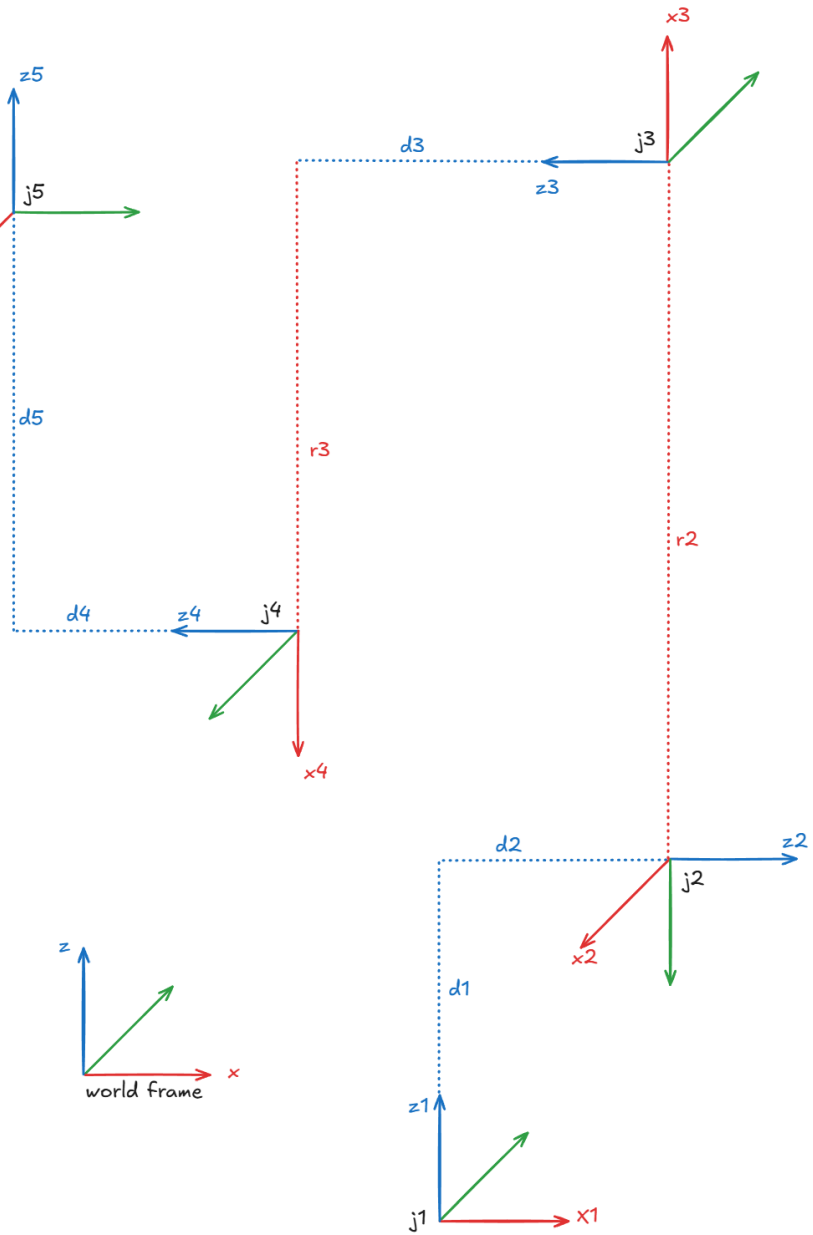
https://www.youtube.com/watch?v=lB9bhnnsFkw

DH Table



(Deg.)	θ,z	α,x	d	r
J1	-90.000	-90.000	50.000	0.000
J2	-90.000	180.000	50.000	850.000
J3	180.000	0.000	100.000	750.000
J4	90.000	-90.000	50.000	0.000
J5	90.000	90.000	200.000	0.000
J6	0.000	0.000	50.000	0.000





DH Table Validation

Transform Modeling in Excel

	IN (Deg)
J1	0.0000
J2	0.0000
J3	0.0000
J4	0.0000
J5	0.0000
J6	0.0000

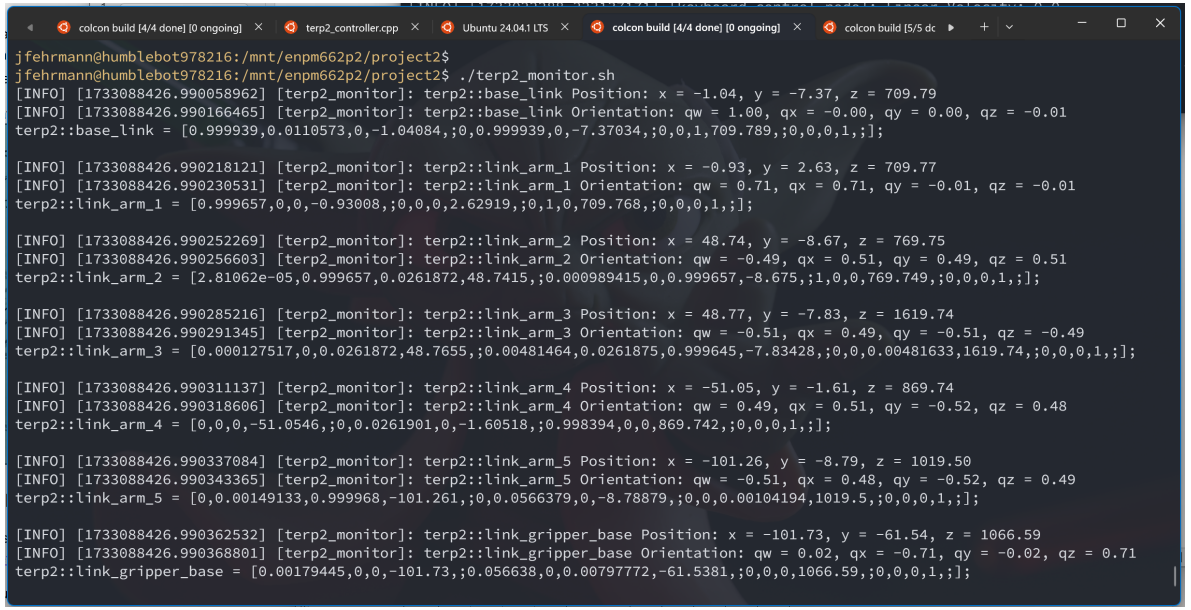
(Deg.)	θ,z	α,x	d	r
J1	-90.000	-90.000	760.000	0.000
J2	-90.000	180.000	50.000	850.000
J3	180.000	0.000	100.000	750.000
J4	90.000	-90.000	50.000	0.000
J5	90.000	90.000	200.000	0.000
J6	0.000	0.000	50.000	0.000

(RAD.)	TOOL
X	0.000
Y	0.000
Z	0.000
α	0.000
β	0.000
γ	0.000

OUT (Deg)	
X	-100.000
Y	-50.000
Z	1060.000
R_z	0.000
R_y	0.000
R_x	90.000

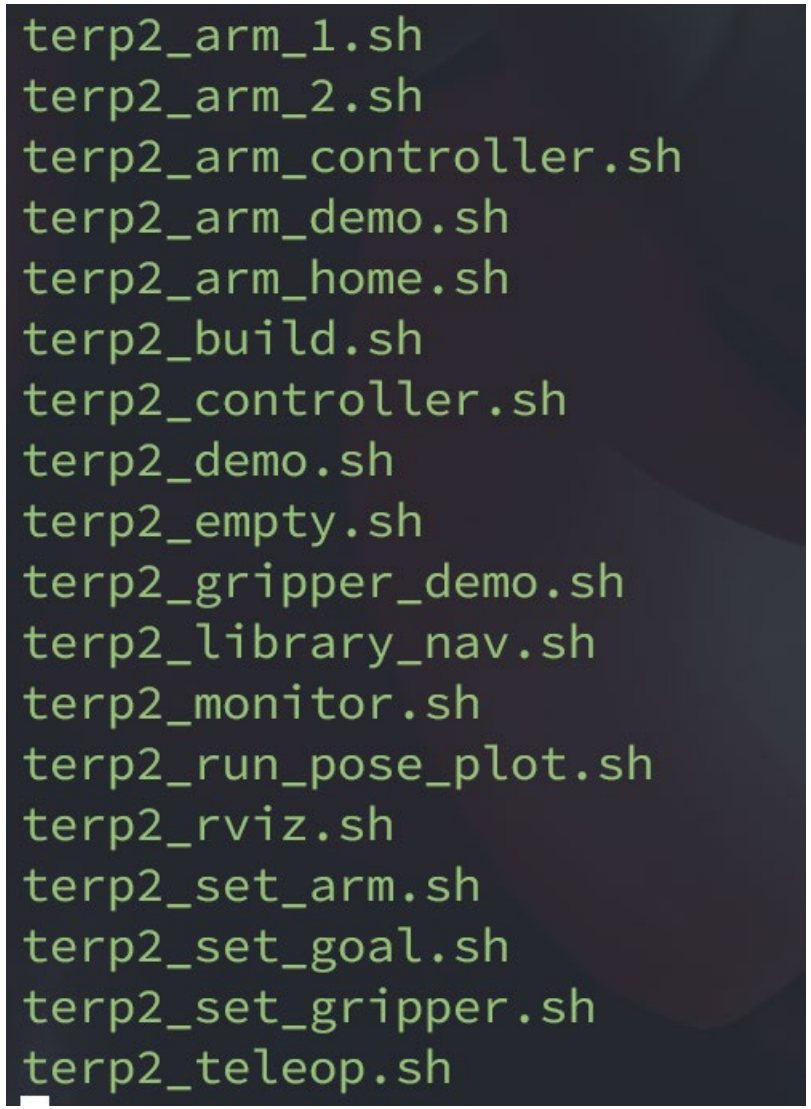
θ	-1.571	J1	0.000	0.000	1.000	0.000	⇒	R0-1	0.000	0.000	1.000	0.000
α	-1.571		-1.000	0.000	0.000	0.000			-1.000	0.000	0.000	0.000
d	760.000		0.000	-1.000	0.000	760.000			0.000	-1.000	0.000	760.000
r	0.000		0.000	0.000	0.000	1.000			0.000	0.000	0.000	1.000
θ	-1.571	J2	0.000	-1.000	0.000	0.000	⇒	R0-2	0.000	0.000	-1.000	50.000
α	3.142		-1.000	0.000	0.000	-850.000			0.000	1.000	0.000	0.000
d	50.000		0.000	0.000	-1.000	50.000			1.000	0.000	0.000	1610.000
r	850.000		0.000	0.000	0.000	1.000			0.000	0.000	0.000	1.000
θ	3.142	J3	-1.000	0.000	0.000	-750.000	⇒	R0-3	0.000	0.000	-1.000	-50.000
α	0.000		0.000	-1.000	0.000	0.000			0.000	-1.000	0.000	0.000
d	100.000		0.000	0.000	1.000	100.000			-1.000	0.000	0.000	860.000
r	750.000		0.000	0.000	0.000	1.000			0.000	0.000	0.000	1.000
θ	1.571	J4	0.000	0.000	-1.000	0.000	⇒	R0-4	0.000	1.000	0.000	-100.000
α	-1.571		1.000	0.000	0.000	0.000			-1.000	0.000	0.000	0.000
d	50.000		0.000	-1.000	0.000	50.000			0.000	0.000	1.000	860.000
r	0.000		0.000	0.000	0.000	1.000			0.000	0.000	0.000	1.000
θ	1.571	J5	0.000	0.000	1.000	0.000	⇒	R0-5	1.000	0.000	0.000	-100.000
α	1.571		1.000	0.000	0.000	0.000			0.000	0.000	-1.000	0.000
d	200.000		0.000	1.000	0.000	200.000			0.000	1.000	0.000	1060.000
r	0.000		0.000	0.000	0.000	1.000			0.000	0.000	0.000	1.000
θ	0.000	J6	1.000	0.000	0.000	0.000	⇒	R0-6	1.000	0.000	0.000	-100.000
α	0.000		0.000	1.000	0.000	0.000			0.000	0.000	-1.000	-50.000
d	50.000		0.000	0.000	1.000	50.000			0.000	1.000	0.000	1060.000
r	0.000		0.000	0.000	0.000	1.000			0.000	0.000	0.000	1.000
		TOOL	1.000	0.000	0.000	0.000	⇒	R0-T	1.000	0.000	0.000	-100.000
			0.000	1.000	0.000	0.000			0.000	0.000	-1.000	-50.000
			0.000	0.000	1.000	0.000			0.000	1.000	0.000	1060.000
			0.000	0.000	0.000	1.000			0.000	0.000	0.000	1.000

Output from terp2_monitor



Deployment Model

- Worked from OSRF docker image, ***osrf/ros:humble-desktop-full***
- Runtime and Development environment fully codified in Docker.
  - Developers working on Ubuntu, Arch, and Windows WSL environments.
  - No local dependencies except **git** and **docker**
- Codify anything useful as a bash script.
- Clean workflow facilitates fast iterative testing:
  - checkout code
  - update image (if required)
  - launch container
  - make edits
  - build and run scripts
  - push code updates



Challenges

- **Ambitious Project Scope:** Many topics were new to everyone. Lots of "hack/slash" to get things working without fully understanding each software component.
    - *No time for refinement or major changes.*
- **Systemic Fragility:** Too many sources for issues: tedious math, ROS and its ecosystem of plugins, multiple programming languages.
    - *Very hard to narrow down and find root causes.*
- **Time Management:** Hard to balance time between learning mathematics modeling topics and spending time debugging ROS issues.
    - *Accept best efforts over best outcomes.*
- **Role Management:** Time pressures forced team members into fixed roles. No time or capacity to switch roles.
    - *No opportunity to step outside comfort zone.*

Successes

- **DevOps:** Codified and portable runtime and dev environment will be tremendously useful for future project.
- **ROS:** Learned basic ROS concepts such as spawning a robot, creating nodes, topics, parameters, etc.
- **PID Controller:** Implementing a PID Controller and seeing how changes to values affect the robot's motion.
- **Quaternion Math:** Worked with quaternion representations for robot orientation, and converting to roll-pitch-yaw, Euler angles, and Transform matrices.
- **Gazebo:** Gained experience working in Gazebo and parsing Gazebo state data.