Forward Kinematics of a 6-DOF Rot3u Robotic Arm

Code Link: Github Repository - Arduino Sketch
Code Link: Github - Python Scripts

The Cartesian Choreographers Team Members:

Name	ID
Jaysh Muhammad	450074
Syed Furqan Ali	451802
Zain Ul Abdeen	450682
Rana Talal Ahmad Khan	450436

1. Introduction

This report details the forward kinematics analysis performed on a 6-DOF Rot3u robot. Forward kinematics determines the end-effector position and orientation of a robot manipulator based on the joint angles. This information is crucial for robot control and path planning applications.

2. Denavit-Hartenberg (DH) Parameters

The analysis utilizes the DH convention to establish a systematic approach for calculating the robot's forward kinematics. The DH table includes the following parameters for each joint:

- α (alpha): Angle of rotation about the previous x-axis. This twist aligns the previous z-axis with the current z-axis.
- a (a): Offset distance along the previous x-axis between the previous and current joint axes.
- θ (theta): Joint angle variable (rotational).
- d (d):Distance along the previous z-axis from the previous origin to the current origin.

| i	a	α	d	θ
0	0	0	0	θ_0
1	0	90	0	θ_1
2	10.5cm	0	0	θ_2
3	10cm	0	0	θ_3
4	0	0	0	θ_4

From the DH table the Transformation matrices from base frame 0 to frame 4 were calculated using the python script, then the resultant matrix was multiplied by the last transformation of frame 4 to 5 shown below which gave us the final transformation from base to the end effecter 4x4 transformation Matrix from frame 4 to frame 5 is given below in tabular form:
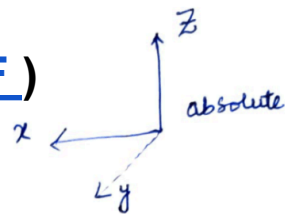
$^4T_5 =$

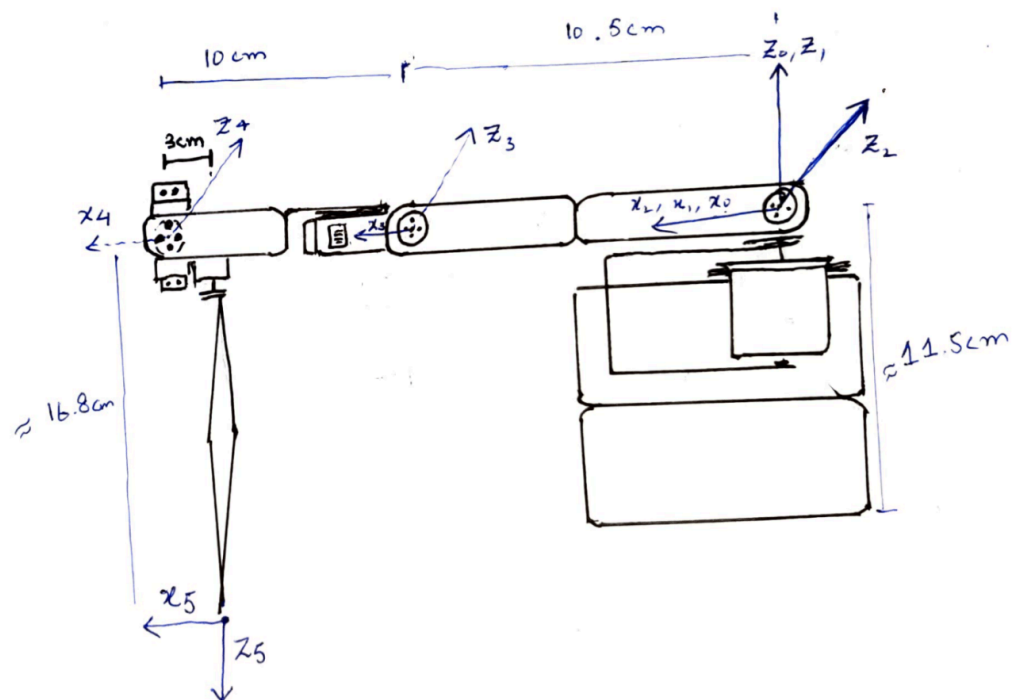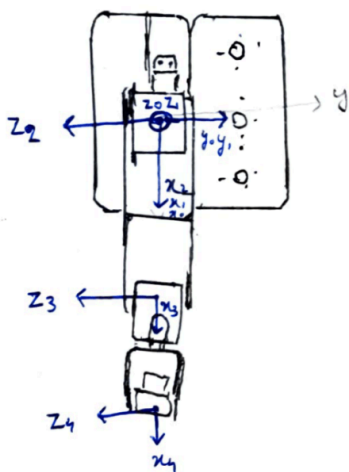| 1	0	0	-2.5cm
0	$\cos 90^0$	$-\sin 90^0$	-16.8cm
0	$\sin 90^0$	$\cos 90^0$	0
0	0	0	1

3. Initial Configuration

The initial configuration of the robot, including the joint angles (θ values = 0) for each joint, is specified. This serves as the starting point for the forward kinematics calculations.
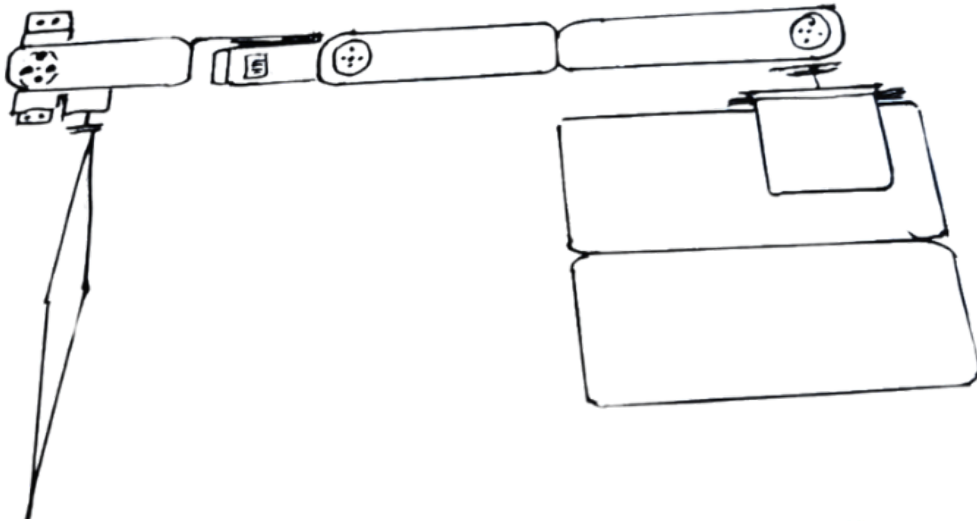
3. Frame assignment ([Frames PDF](#))

Top view.

4. Distributed Forward Kinematics Implementation for Rot3u Robot with Arduino Uno and Python (Code Implementation)

System Architecture

The system comprises two primary components:

1. **Arduino Uno: Responsible for low-level motor control based on received commands.**
2. **Python Script: Handles forward kinematics calculations, user interaction and serial communication with the Arduino.**

Communication Protocol

Serial communication via the `pyserial` library in Python establishes a communication channel between the Python script and the Arduino. The Python script transmits angle values as a series of bytes, which the Arduino receives and interprets to control the robot's joints.

Code Breakdown

1. Python Script:

- **Forward Kinematics Function: This function implements the forward kinematics algorithm using the DH parameters and received joint angles. It likely utilizes libraries like `numpy` for matrix operations. The function calculates the final homogeneous transformation matrix representing the end-effector pose.**
- **GUI Implementation: `Tkinter` and `customTkinter` is used to create a user interface for controlling the robot. The user can interact with input fields to specify desired joint angles, which are then sent to the Arduino.**
- **Serial Communication: The `pyserial` library is employed to establish a serial connection with the Arduino at the specified baud rate. The script transmits the calculated joint angles (converted to bytes) through the serial port.**

2. Arduino Uno Code:

- **Serial Communication Setup: The Arduino code initializes the serial communication library to receive data from the Python script.**
- **Motor Control: Based on the received angle values (interpreted from bytes), the Arduino code controls the movement of each joint using dedicated Servo Control Library `servo.h`**

Benefits of Distributed Approach

- **Offloading Computation: Python, on a PC, handles the computationally intensive forward kinematics calculations, freeing the Arduino for real-time motor control.**
- **Flexibility: The Python script offers flexibility for implementing a user interface**
- **Modular Design: The separation of concerns between control and computation simplifies code maintenance and future modifications.**
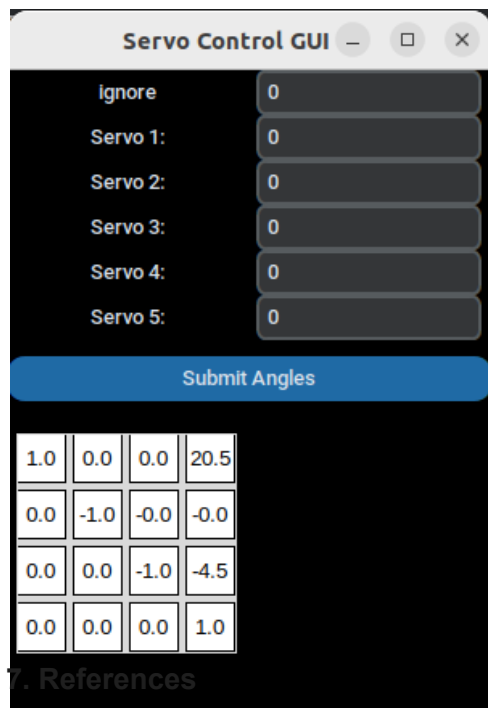
References and Special thanks

- [Gemini](#)
- [How to Build a DIY Aluminium 6-DOF Robotic Arm From Scratch – Automatic Addison](#)