Modeling and Control of Poppy Humanoid Robot

Ahmed Yasser Sallam*, Mohamed Elsaeed Bargout*
, Mostafa Moataz Abdelmagid*,Youssef Tamer Ismail*, and Ziyad Mohamed Wahba*
*German University in Cairo (GUC), Egypt
Emails: {ahmed.aminsallam, mohamed.bargout, mostafa.abdelmagid, youssef.tamerelakkad, ziyad.wahba}@student.guc.edu.eg

***Abstract*—We present an application of robotic manipulation in the context of service robotics, focusing on the task of pouring water into a cup using the Poppy Humanoid Robot's left arm. The project involves creating a kinematic model, deriving inverse kinematics in MATLAB, and validating the equations through Simscape simulations. A joint space trajectory is developed, enabling the robot to transfer water between robots and pour it into a cup. Simulation results in Simscape are validated through implementation on the actual robot, providing insights into real-world challenges. This work contributes to the field by demonstrating a practical application of kinematics, trajectory planning, and hardware implementation on a humanoid robotic platform.**

***Keywords*-Service Robotics; Robotic Arm; Humanoid Robot; Simscape Multibody.**

I. Introduction

Service robotics has emerged as a critical domain, contributing to various applications that enhance human life. In this context, our work focuses on the application of robotic manipulation in service robotics, with a specific emphasis on the task of pouring water into a cup. Efficient and precise manipulation is a fundamental aspect of service robots, enabling them to perform tasks that are intricate and context-dependent. The chosen robotic platform for this project is the left arm of the Poppy Humanoid Robot shown in Figure 1, a versatile and humanoid system.

Figure 1: Poppy Humanoid Robot.

The primary objective of our study is to develop a comprehensive understanding of the kinematics involved in pouring water, ranging from the creation of a detailed kinematic model to the derivation of inverse kinematics using MATLAB. To ensure the reliability of our theoretical models, we employ Simscape simulations for both forward and inverse kinematics, providing a robust foundation for the subsequent trajectory planning.

The trajectory planning aspect of our work involves the generation of a joint space trajectory, allowing the robot to seamlessly transfer water from one robotic entity to another and execute the precise task of pouring it into a cup. Through rigorous simulation in Simscape and practical implementation on the Poppy Humanoid Robot, we validate the effectiveness of our proposed methodology.

This paper contributes to the field by presenting a practical application of kinematics and trajectory planning in the context of service robotics, addressing the nuanced challenge of pouring water into a cup. The insights gained from our hardware implementation on the humanoid robot provide valuable contributions to the broader landscape of robotic manipulation.

The remainder of this paper is organized as follows: Section II details the methodology, including the kinematic model, inverse kinematics, and trajectory planning. Section III presents the simulation results. Finally, Section IV concludes the paper, summarizing the key findings and suggesting avenues for future research.

II. Methodology

In this section, we provide a comprehensive overview of the methodology employed in our project, encompassing forward and inverse kinematics operations, trajectory planning, hardware implementation, and Simscape Multibody modeling.

A. Kinematic Model

The foundation of our project lies in the development of a detailed kinematic model for the Poppy Humanoid

Robot's left arm. This model forms the basis for subsequent kinematic operations, including the calculation of forward and inverse kinematics. The robotic arm is composed of four servo motors, resulting in a 4-degree-of-freedom (4-DOF) system with four revolute joints. To describe the relationship between the joint angles (q_1, q_2, q_3, q_4) and the task space coordinates (x, y, z), we employed the Denavit-Hartenberg (DH) convention.

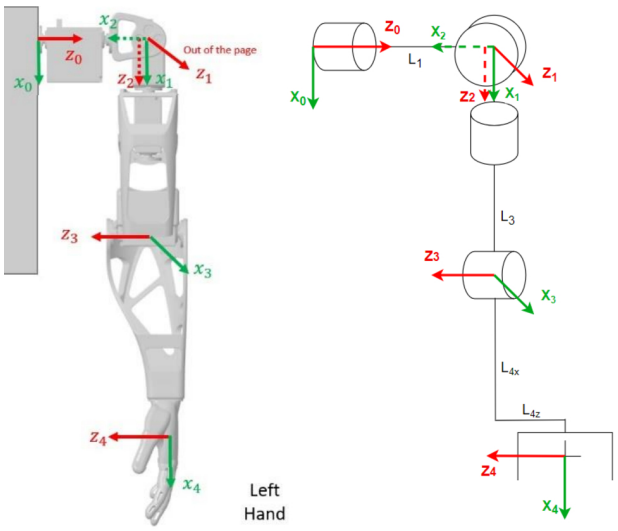

Figure 2: DH Convention Frame Assignment

Using the frame assignment shown in Figure 2, the DH convention table can be derived which encapsulates the kinematic parameters of each joint, allowing us to derive the transformation matrix that relates consecutive joints in the robotic arm. The resulting kinematic equations enable us to predict the end-effector's position and orientation based on the servo motor angles. The DH parameters and corresponding transformation matrices are outlined in Table I.

Table I: DH Convention Parameters Table

$Joint_i$	θ_i	d_i	a_i	α_i
1	q_1	L_1	0	$90°$
2	$-(90° - q_2)$	0	0	$-90°$
3	$-(90° - q_3)$	L_3	0	$-90°$
4	$-(90° - q_3)$	L_{4z}	L_{4x}	0

Where,

L_1: 65.73mm L_{4z}: -13.5mm
L_3: 145.66mm L_{4x}: 143.55mm

Here, θ_i is the joint angle, d_i is the link offset, a_i represents the link length, and α_i is the twist angle. These parameters, combined with the DH convention, facilitate the derivation of the transformation matrix for each joint, providing a systematic approach to modeling the kinematics of the robotic arm.

B. Forward Kinematics

With the established kinematic model, forward kinematics are vital for deducing the robotic arm's position, orientation, velocity, and acceleration. Derived from DH parameters, these equations seamlessly bridge joint and task space, facilitating a comprehensive understanding of the robot's spatial configuration and motion. Figure 3 provides a visual representation of the transition from joint space to task space.

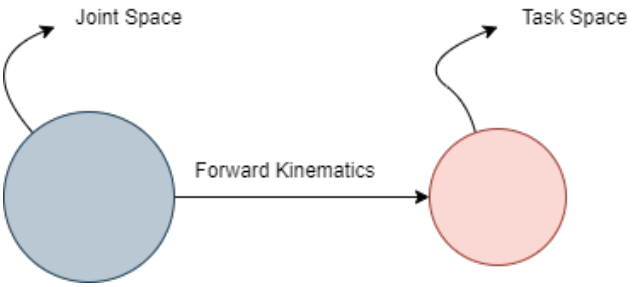

Figure 3: Forward Kinematics.

1) Forward Position Kinematics: The forward kinematics for position involves deriving transformation matrices $T_{(i-1)}$ for each joint, which collectively transform the robot's coordinate system from the base to the end-effector. The transformation matrix for each joint i is defined as:

$$ {}^{i-1}T_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The product of these transformation matrices, starting from the base $(i = 1)$ up to the end-effector, yields the overall transformation matrix for the robot's forward kinematics:

$$ {}^0T_4 = {}^0T_1 \cdot {}^1T_2 \cdot {}^2T_3 \cdot {}^3T_4 $$

2) Forward Velocity Kinematics: The total transformation matrix for forward position kinematics is represented as:

$$ {}^0T_4 = \begin{bmatrix} {}^0R_4 & \vec{X} \\ \vec{0} & 1 \end{bmatrix}, \vec{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} $$

Here, ${}^0R_4$ represents the 3x3 rotation matrix, and $\vec{X}$ represents the translation vector specifying the end-effector position. This format provides a clear separation between the rotation matrix and translation components in the final transformation matrix.

By differentiating the translation vector with respect to time and applying the chain rule, we obtain the velocity vector $\dot{\vec{X}}$:

$$\dot{\vec{X}} = \frac{d\vec{X}}{dt} = \frac{\partial \vec{X}}{\partial \vec{q}} \cdot \frac{d\vec{q}}{dt} = J \cdot \dot{\vec{q}}$$

The Jacobian matrix J, defining the relationship between each motor's velocity and the change in velocity in the X, Y, and Z directions, is given by:

$$J = \frac{\partial \vec{X}}{\partial q} = J = \begin{bmatrix} \frac{\partial X}{\partial q_1} & \frac{\partial X}{\partial q_2} & \frac{\partial X}{\partial q_3} & \frac{\partial X}{\partial q_4} \\ \frac{\partial Y}{\partial q_1} & \frac{\partial Y}{\partial q_2} & \frac{\partial Y}{\partial q_3} & \frac{\partial Y}{\partial q_4} \\ \frac{\partial Z}{\partial q_1} & \frac{\partial Z}{\partial q_2} & \frac{\partial Z}{\partial q_3} & \frac{\partial Z}{\partial q_4} \end{bmatrix}$$

3) Forward Acceleration Kinematics: By differnetiating the velocity Vector with respect to time, we obtain the acceleration vector $\ddot{\vec{X}}$:

$$\ddot{\vec{X}} = \frac{d\dot{\vec{X}}}{dt} = \frac{d(J \cdot \dot{\vec{q}})}{dt} = \dot{J} \cdot \dot{\vec{q}} + J \cdot \ddot{\vec{q}}$$

C. Inverse Position Kinematics

Inverse kinematics operations are essential for determining joint configurations that achieve a desired end-effector position and orientation. These inverse kinematics equations are derived using MATLAB, ensuring precise control over the robot's movements.

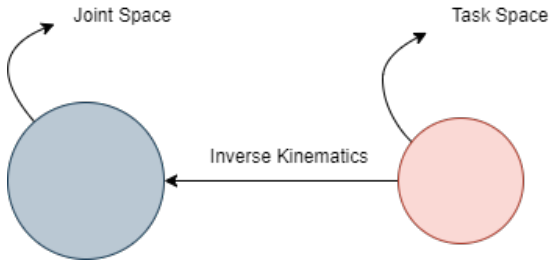

Figure 4: Inverse Kinematics.

Two methods are employed for calculating inverse kinematics. The first is the trigonometric method, but it becomes impractical for complex robots. The second method utilizes the numerical approach through the Newton-Raphson method.

Newton-Raphson Approach: In this approach, the roots of the function $\vec{F} = \vec{X} - \vec{X}_{desired}$ are sought iteratively to determine the $\vec{q}$ values necessary to move the end effector to the desired location:

$$\begin{bmatrix} q_{n+1,1} \\ q_{n+1,2} \\ q_{n+1,3} \\ q_{n+1,4} \end{bmatrix} = \begin{bmatrix} q_{n,1} \\ q_{n,2} \\ q_{n,3} \\ q_{n,4} \end{bmatrix} - \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \\ J_{41} & J_{42} & J_{43} \end{bmatrix} \cdot \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix}$$

Where $\vec{F}$ and J are evaluated at $\vec{q}_n$.

D. Trajectory Planning

The trajectory planning phase focuses on creating a joint space trajectory for the robot. This trajectory encompasses the entire task, including the robot's motion to acquire water from another robot and the subsequent pouring into a cup. A flowchart representing the trajectory planning process is provided in Figure 5.

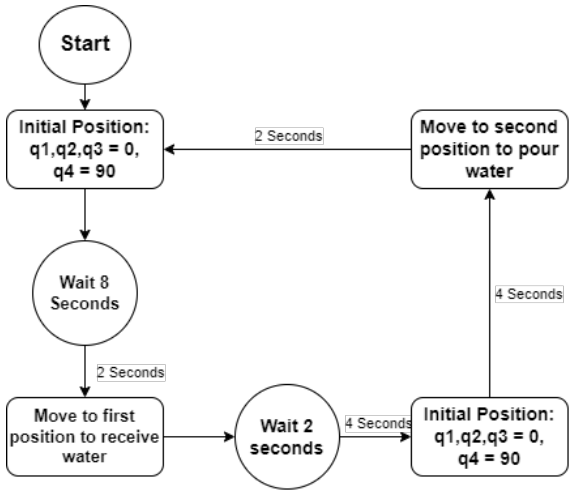

Figure 5: Trajectory Flowchart.

Joint Space Trajectory: In pursuit of our task to pour water from a cup, we've crafted a meticulous trajectory.

Employing a cubic 3rd order polynomial for joint angles ensures a seamlessly smooth motion:

$$q = C_0 + C_1 \cdot t + C_2 \cdot t^2 + C_3 \cdot t^3$$

For the first trajectory, we set $T_f = 2$ seconds with a sampling time $T_s = 0.05$ seconds. The second trajectory, with $T_f = 4$ seconds, is accompanied by a sampling time $T_s = 0.075$ seconds.

For the first trajectory, determining the coefficients (C_0, C_1, C_2, C_3) involves specifying initial and final positions and velocities:

$$\vec{q}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 90 \end{bmatrix}, \quad \vec{q}_f = \begin{bmatrix} 30 \\ 40 \\ 65 \\ 45 \end{bmatrix}, \quad \dot{\vec{q}}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \dot{\vec{q}}_f = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Solving for the coefficients yields the essential polynomial equations for trajectory generation.

Figure 6 visually depicts the robot's joint angles over time for the first trajectory, where each curve corresponds to a specific joint angle trajectory.

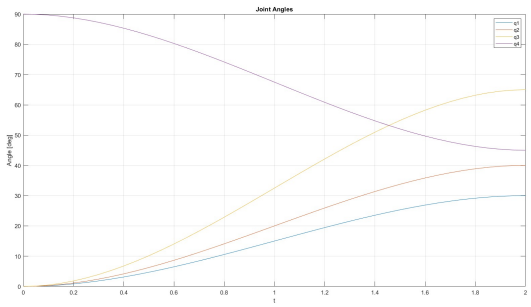

Figure 6: First Trajectory Joint Angles.

Figure 7 and Figure 8 show the robot in its initial and final positions respectively.

In its initial state, the robot holds an empty cup, moving inward to receive water from another robotic arm.

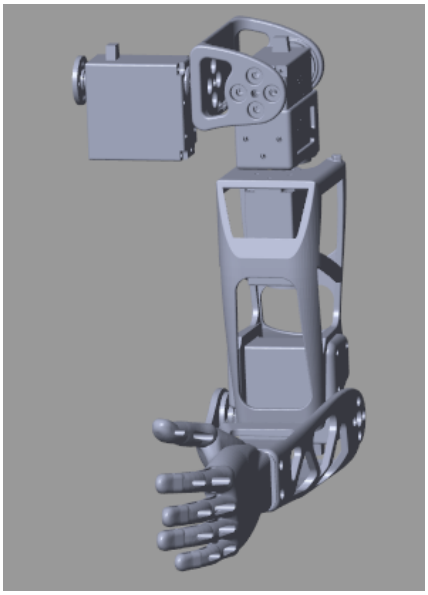

Figure 7: Initial Position.

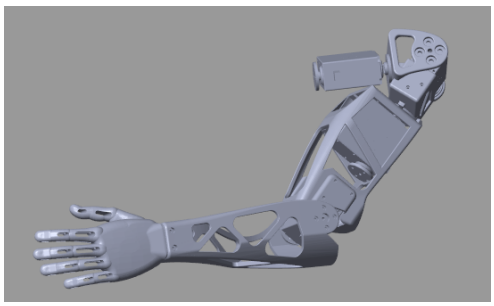

Figure 8: Position 1.

Similarly, the trajectory for moving to position 2 to pour water can be derived using a comparable methodology. Figure 9 visually presents the joint angles over time for the second trajectory.

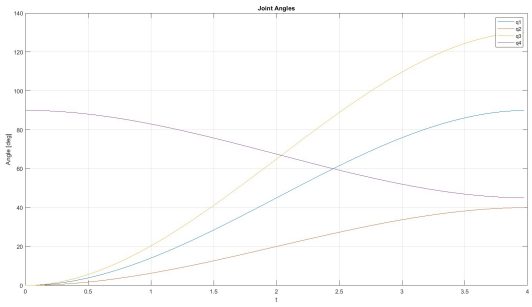

Figure 9: Second Trajectory Joint Angles.

The complete robot trajectory, excluding pauses between motions, is depicted in Figure 10.

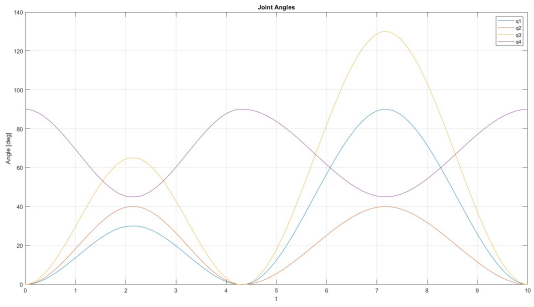

Figure 10: Total Trajectory Joint Angles.

E. Simscape Multibody Modeling

To validate our kinematic and trajectory models, we utilize Simscape Multibody simulations. These simulations allow us to assess the performance of the robot in a virtual environment before hardware implementation. Figure 11 illustrates the Simscape Multibody model of our robotic system.

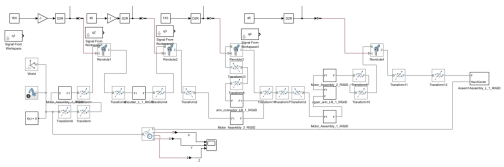

Figure 11: Simscape Multibody Model.

F. Hardware Implementation

The hardware in our system consists of the Poppy humanoid robot's left arm, a task-specific end effector, and four servo motors (two with 10 Kg.cm torque and two with 15 Kg.cm torque). Additionally, we use a 5V/5A power supply unit, a microcontroller (Arduino UNO), jumper wires, and securely fastened screws. These components, detailed in Table II, collectively form the physical framework of our robotic system.

Table II: Hardware Components Table

Component Name	Purchase Location	Quantity	Price (per item)
Servo Motor 10kg	Future Electronics	2	550
Servo Motor 15kg	Future Electronics	2	650
Left Arm	Home 3D Printer	1	500
Power Supply 5V	Ampere Electronics	1	175
Arduino UNO	Ampere Electronics	1	360

Hardware: The robot is mounted using a bracket on an aluminum 2020 extrusion profile stand, as shown in Figure 12.

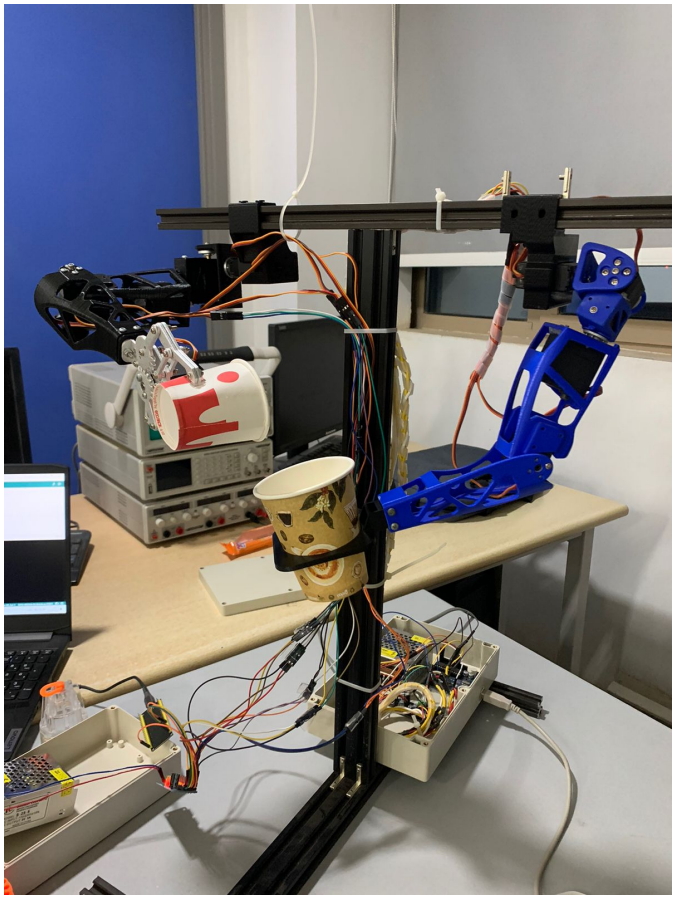

Figure 12: Robot Mounting and Circuitry.

End-Effector: To handle the cup for our water-pouring task, we designed a purpose-built end effector using SOLIDWORKS, as depicted in Figure 13. This specialized attachment plays a crucial role in ensuring precise execution.

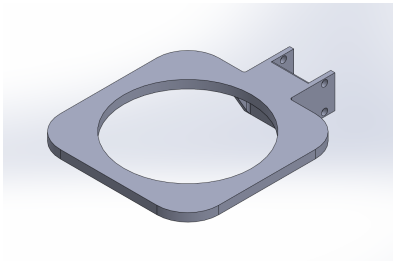

Figure 13: End-Effector Design.

Circuit Diagram: The circuit connections for our system are exemplified in Figure 14. An Arduino microcontroller UNO serves as the central control unit, interfacing with the four servo motors through PWM (Pulse Width Modulation) pins. These signals provide precise control over the servo motors, dictating their position and movement. The microcontroller and servo motors are powered by a 5V/5A power supply. The organized circuitry is shown in **??**

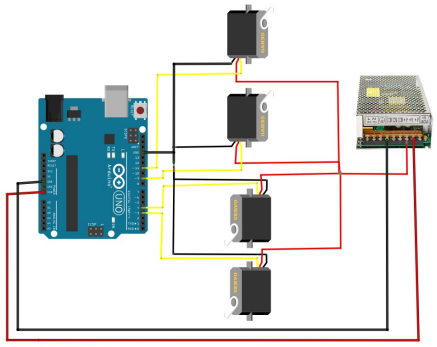

Figure 14: Circuit Diagram.

Figure 15: Organized Circuitry.

III. Results

Simulation was conducted using MATLAB/Simulink, comprising four tests to analyze the motion of the robotic arm. In the initial test, the robot was observed in its default state, with all joints set to zero positions. The second test applied constant values as inputs to various joints. The third test utilized a signal builder to apply a customized signal to the joints. Finally, the fourth test involved applying a sine wave signal as input to the upper shoulder joint of the robot arm.

Figures 16 and 17 showcase snapshots from the simulation, providing visual insights into the robotic arm's behavior during the initial and fourth tests, respectively.

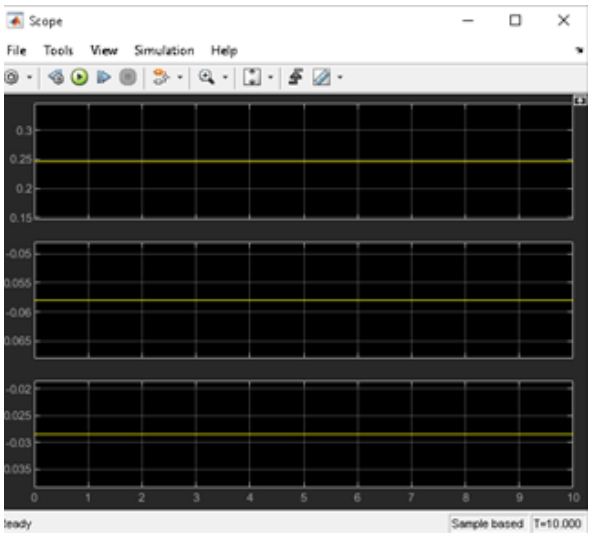

Figure 16: Motion Results in Initial Position.

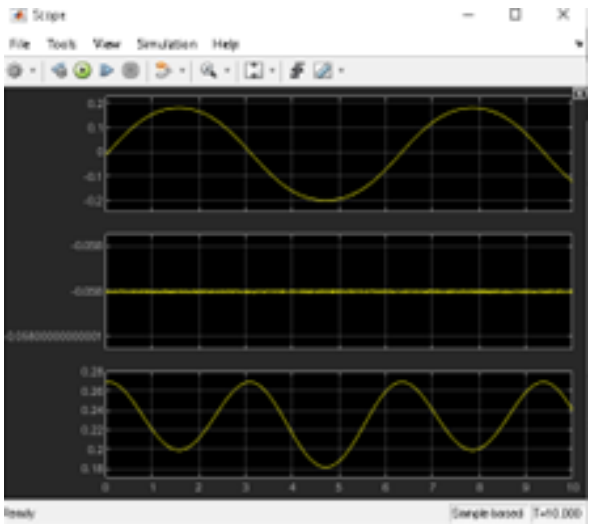

Figure 17: Motion Results due to Sine Wave Test.

IV. Conclusion and Future Work

A. Conclusion

In conclusion, the robot arm successfully executed its designated trajectory, effectively pouring liquids into cups and demonstrating functionality suitable for assisting elderly individuals. The design, modeling, hardware, and control mechanisms proved adequate for the intended task. However, as the arm's desired features and functionalities increase, the complexity rises, necessitating more sophisticated trajectories and control strategies.

To witness the robot arm in action and view a simulation of the trajectory, a video is accessible by scanning the QR code displayed in Figure 18.

Figure 18: QR Code for Video and Simulation.

B. Future Work

Upon reviewing the presented video, a minor shaking in the robot arm was observed, potentially leading to liquid spillage. This occurrence is attributed to the relatively large sampling time chosen, a compromise made due to the limited storage capacity of the Arduino, allowing only for a shorter trajectory. As part of future work, the implementation could benefit from a microcontroller with an expanded memory capacity to accommodate longer and smoother trajectories, minimizing any unintended oscillations.

References