Modeling and Control of an upper Humanoid Robot as a Basketball Player

Dina Mohamed*, Marwa Lotfy*,
Nada Tamer*, Ahmed Mohamed Sleem* and Karim Emad Adam*
*German University in Cairo (GUC), Egypt
Emails: dina.mohamed@student.guc.edu.eg, marwa.hassan@student.guc.edu.eg, nada.abdelhay@student.guc.edu.eg,
ahmed.sleem@student.guc.edu.eg, karim.adam@student.guc.edu.eg

***Abstract*—This project aims to design a system that can pick up and shoot a basketball with two robotic arms. It also focuses on creating interactive experiences for elderly individuals and children by engaging in playful activities with a softball. The project combines robotics, control systems, and entertainment to enhance well-being, social interaction, and physical activity for different age groups. Success will be measured by the system's ability to accurately pick up and shoot the basketball.**

***Keywords*-Robotics, Basketball, Control Systems.**

I. Introduction

The integration of robotics and sports has marked a groundbreaking advancement in the field of technology, offering us the opportunity to explore new dimensions of human-machine interaction. In this context, our research delves into the fascinating realm of humanoid robotics, aiming to develop a Poppy humanoid arm system capable of playing basketball. This innovative project merges the intricacies of mechanical design, control algorithms, and real-time sensing, presenting a significant step forward in the pursuit of enhancing both robotic capabilities and human-robot cooperation in dynamic and demanding environments.

The primary objective of this paper is to detail the design, modeling, and control strategies employed in the creation of a Poppy upper humanoid arm specifically optimized for playing basketball. The project seeks to bridge the gap between theoretical robotic design and real-world application in sports, with the ultimate goal of demonstrating how advanced robotics can be integrated into recreational activities, education, or even therapy for individuals with physical disabilities.

You should also state the contribution of this report in a paragraph stating what you are going to present.

In section II, the robot's hardware design and implementation is going to be presented. section V presents the simulation results.

II. Hardware Design and Implementation

In order to build the hardware of the system, certain components are required as well as the circuit diagram of the system is built.

A. Hardware Components

The poppy upper humanoid open source is used and 3D printed for hardware testing. For the motors, three double axes motors holding 10 kg.cm torque max and one motor holding 15 kg.cm for the shoulder to hold the whole system. Arduino Uno is used to control the four motors.

Table I: Hardware Components Table

Name of component	Location of purchasing	No. of items	Price
3D printed arm	Make It Real	1	500
Servo motor 10kg.cm	Future electronics	3	550
Servo motor 15kg.cm	Future electronics	1	750
Arduino Uno	Future electronics	1	410
Power Supply	Ram electronics	1	650
Wires, Jumpers	Future electronics	Some	50
3d printed arm	Make It Real	1	320
Aluminum Extrusion	CNC Egypt	2 m	160
L shape corner	CNC Egypt	10	200
3d connecting part	Abdelhameed	1	50
Total			3640

B. Circuit Diagram

As shown in Figure 1, the circuit consists of 4 motors, each having three connections for the voltage, the ground, and the signal which is the output of the Arduino. A breadboard to connect the wires at common nodes.

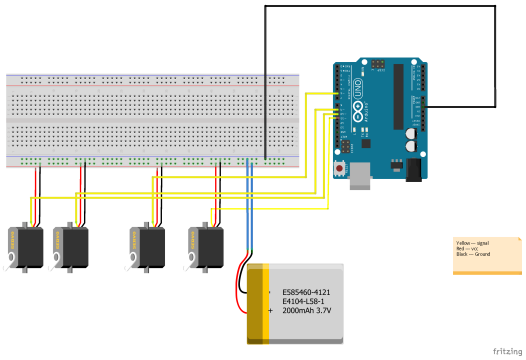

Figure 1: Circuit Diagram

III. Forward Kinematics

A. Denavit-Hartenberg (DH) convention

Forward kinematics using the Denavit-Hartenberg (DH) convention can be broken down into four key steps. First,

assign DH axes to each joint of the robotic arm, ensuring they follow the right-hand rule as seen in Fig. 8. Second, fill out the DH parameters table, which includes values for the link lengths, joint angles, link twists, and offset distances, seen in II. This table is vital for the DH transformation process. Third, using these DH parameters, calculate the DH transformation matrices for each joint. These matrices represent the transformation between consecutive joints and are the building blocks for the final step. Lastly, obtain the end effector's transformation matrix by multiplying the individual DH transformation matrices in a cascading manner. This final matrix describes the position and orientation of the robot's end effector in the reference frame of the base, providing valuable information for controlling and planning the robot's motion.

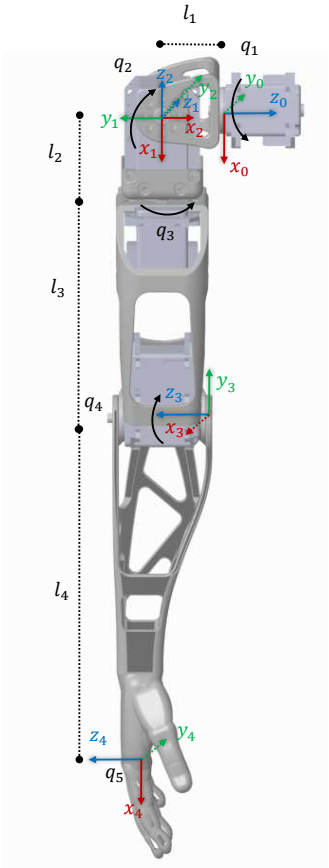

Figure 2: DH Frames.

joint(i)	θ_i	d_i	a_i	α_i
1	q_1	$-L_1$	0	$-\frac{\pi}{2}$
2	$q_2 - \frac{\pi}{2}$	0	0	$\frac{\pi}{2}$
3	$q_3 - \frac{\pi}{2}$	$-(L_2 + L_3)$	0	$\frac{\pi}{2}$
4	$q_4 - \frac{\pi}{2}$	0	L_4	0

Table II: DH - Parameters Table

The distances between each joint is evaluated using a transform sensor and have the following values:

L_1	0.02888114877491
L_2	0.026599975
L_3	0.122400025
L_4	0.15

Table III: Lengths table

IV. Inverse Kinematics

Inverse kinematics refers to the process of determining the joint configurations of a robot's manipulator that will result in a desired end-effector position and orientation. It involves finding the joint angles or joint displacements needed to achieve a specific pose (position and orientation) of the robot's end-effector.

Let $\mathbf{q}$ represent the vector of joint variables, and $\mathbf{p}$ represent the desired end-effector position and orientation. The relationship between joint variables and end-effector pose is typically given by a kinematic function $\mathbf{F}(\mathbf{q})$:

$$\mathbf{F}(\mathbf{q}) = \mathbf{p}$$

The goal of inverse kinematics is to solve for $\mathbf{q}$ given $\mathbf{p}$. This is often a nonlinear problem due to the complex nature of robot kinematics.

A. Jacobian Matrix

The Jacobian matrix, denoted as $\mathbf{J}$, relates the joint velocities $\dot{\mathbf{q}}$ to the end-effector linear and angular velocities $\mathbf{v}$:

$$\mathbf{v} = \mathbf{J}(\mathbf{q}) \cdot \dot{\mathbf{q}}$$

Here, $\mathbf{J}(\mathbf{q})$ is the Jacobian matrix.

B. Newton-Raphson Method for Inverse Kinematics

The Newton-Raphson method is an iterative numerical technique used to find successively better approximations of the root of a real-valued function. In the context of inverse kinematics, it is applied to iteratively refine the joint variable estimates until the desired end-effector pose is achieved. The Newton-Raphson iteration for overactuated systems using the pseudo Jacobian can be expressed as:

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \mathbf{J}^{+}(\mathbf{p}_{\text{desired}} - \mathbf{F}(\mathbf{q}_n))$$

Here,

- $\mathbf{q}_{n+1}$ is the updated estimate of joint variables.
- $\mathbf{q}_n$ is the current estimate of joint variables.
- $\mathbf{J}^{+}$ is the pseudo Jacobian matrix.
- $\mathbf{p}_{\text{desired}}$ is the desired end-effector pose.
- $\mathbf{F}(\mathbf{q}_n)$ is the current value of the kinematic function.

Pseudo Jacobian for Overactuated System

In overactuated systems where we control more joints than the number of degrees of freedom of the end-effector (e.g., controlling x, y, z with 4 joints), the pseudo Jacobian $\mathbf{J}^+$ can be used. For a 3-by-4 Jacobian $\mathbf{J}$, the pseudo Jacobian is given by:

$$\mathbf{J}^+ = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}$$

This pseudo Jacobian allows the computation of joint velocities even in cases where the system is overactuated.

The Newton-Raphson iteration for solving the inverse kinematics problem can then be modified with the pseudo Jacobian.

V. Simulations

A. Block diagram

The robot arm is exported from Solidworks, and then imported to Simscape and tested with different inputs, constants, sine waves, signal editor, and lastly, a slider. the following figures show the block diagram.

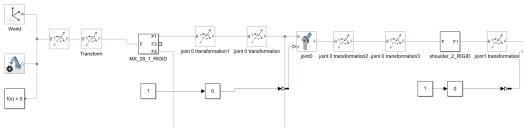

Figure 3: Block diagram part 1.

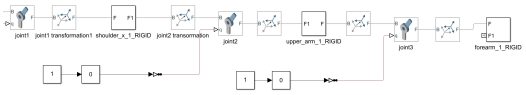

Figure 4: Block diagram part 2.

B. Forward position Simulations

As shown below in the figure, Here is a comparison between the output for the simulation when q1=90, q2=60 in using the forward kinematics simulink model and simscape.

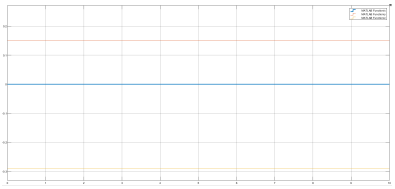

Figure 5: Simulink Output

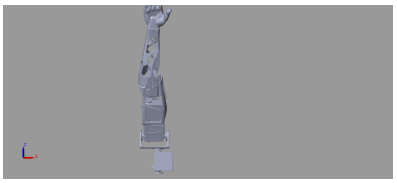

Figure 6: SimScape Output

Here are the results obtained for q1=90 q2=60 q3=20 q4=50

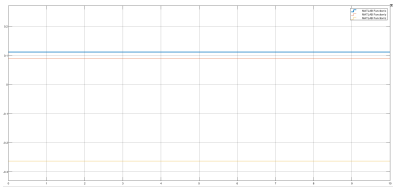

Figure 7: Simulink Output

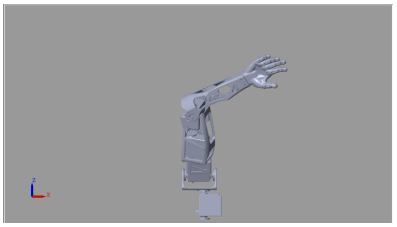

Figure 8: SimScape Output

VI. Tajectory planning

The Robot End effector will be first tested to follow a straight line according to the following data:
$\mathbf{x}_i = [0.231, 0.07834, -0.05057]$
$\mathbf{x}_f = [0, 0.2, -0.05057]$
$T_f = 3$
$T_s = 0.1$
$q_{in} = [40; 30; 20; 50]$

where $\mathbf{x}_i$ is the initial position, $\mathbf{x}_f$ is the final position, T_f is the duration (final time), T_s is the sampling time, and the q_{in} is the initial angel The trajectory is shown in the following Figure:

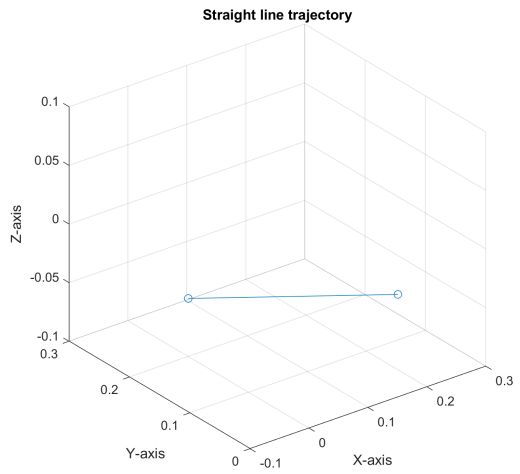

Figure 9: Straight line equation.

The produced angles will follow the trajectory in Fig.10, and the values for the Cartesian coordinates of the end effector will follow the values in Fig.11.

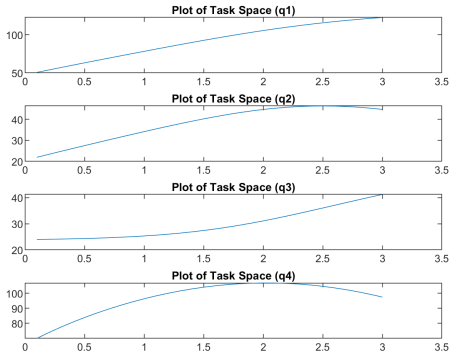

Figure 10: joint angles values at each time sample.

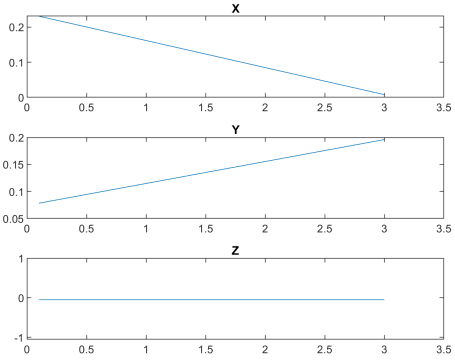

Figure 11: End effector path in space.

These values are tested in the Simulink model as well to verify that the angles will ensure that the trajectory of the end effector is a straight line. After getting the values of the angles of each joint it was saved in an array and loaded in Arduino IDE to run on the fabricated hardware, the resulting trajectory is similar to the simulation with some tolerance, and shift that might be due to the shift in the zero position of the motor which can be solved by adding this shift to the given angle.

VII. Final Implementation

The arm will be mounted on an aluminum extrusion fixation that is fixed together using L-shaped corners, in addition to another 3d printed part to fix the top motor with the aluminum extrusion, the arm is seen in the Fig. 12.

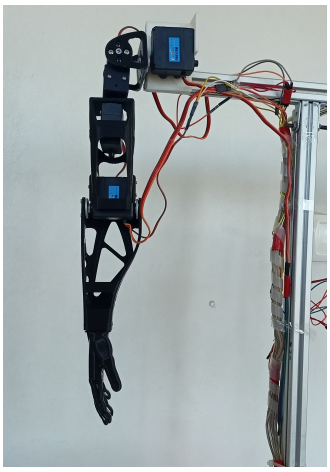

Figure 12: Fixed arm.

Finally, the other arm and the circuitry will be mounted as well, in order to achieve the objective of the robotic arm, which is playing basketball, as seen in Fig. 13 and Fig. 14.

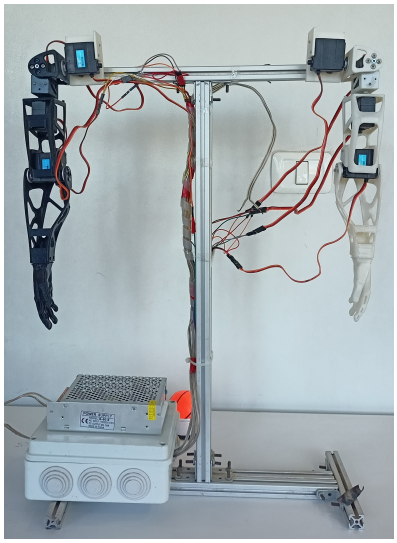

Figure 13: Wires, circuitry and the other arm.

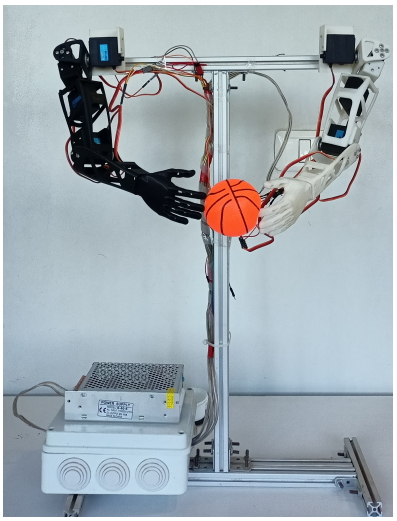

Figure 14: Arm with basketball.

VIII. Appendix

The Jacobian matrix **J** is given by:

$$\begin{bmatrix} j1 & j2 & j3 & j4 \\ j5 & j6 & J7 & j8 \\ j9 & j10 & j11 & j12 \end{bmatrix}$$

Where

$$j1 = \frac{\pi \sin(q4)(\cos(q1)\cos(q3) - \sin(q1)\sin(q2)\sin(q3))}{1200} - \frac{149\pi \cos(q2)\sin(q1)}{180000} - \frac{\pi \cos(q2)\cos(q4)\sin(q1)}{1200}$$

$$j2 = -\frac{\pi \cos(q1)(149\sin(q2) + 150\cos(q4)\sin(q2) - 150\cos(q2)\sin(q3)\sin(q4))}{180000}$$

$$j3 = -\frac{\pi \sin(q4)(\sin(q1)\sin(q3) - \cos(q1)\cos(q3)\sin(q2))}{1200}$$

$$j4 = \frac{\pi \cos(q4)(\cos(q3)\sin(q1) + \cos(q1)\sin(q2)\sin(q3))}{1200} - \frac{\pi \cos(q1)\cos(q2)\sin(q4)}{1200}$$

$$j5 = \frac{\pi \sin(q4)(\cos(q3)\sin(q1) + \cos(q1)\sin(q2)\sin(q3))}{1200} + \frac{149\pi \cos(q1)\cos(q2)}{180000} + \frac{\pi \cos(q1)\cos(q2)\cos(q4)}{1200}$$

$$j6 = -\frac{\pi \sin(q1)(149\sin(q2) + 150\cos(q4)\sin(q2) - 150\cos(q2)\sin(q3)\sin(q4))}{180000}$$

$$j7 = \frac{\pi \sin(q4)(\cos(q1)\sin(q3) + \cos(q3)\sin(q1)\sin(q2))}{1200}$$

$$j8 = -\frac{\pi \cos(q4)(\cos(q1)\cos(q3) - \sin(q1)\sin(q2)\sin(q3))}{1200} - \frac{\pi \cos(q2)\sin(q1)\sin(q4)}{1200}$$

$$j9 = 0$$

$$j10 = -\frac{\pi \cos(q2)(149\pi)}{180000} - \frac{\pi \cos(q2)\cos(q4)}{1200} - \frac{\pi \sin(q2)\sin(q3)\sin(q4)}{1200}$$

$$j11 = \frac{\pi \cos(q2)\cos(q3)\sin(q4)}{1200}$$

$$j12 = \frac{\pi \sin(q2)\sin(q4)}{1200} + \frac{\pi \cos(q2)\cos(q4)\sin(q3)}{1200}.$$

References