Design and Implementation of a 6-DOF Robot Manipulator for *Inverse Kinematics* Education

Ryan Satria Wijaya[1], Al Badrul Munir[2]
Robotics Engineering Department, Politeknik Negeri Batam, Indonesia[1,2]

[*]Email: ryan@polibatam.ac.id[1], albadrulm21@gmail.com[2]

***Abstract*—This research designs and implements a 6-DOF robot manipulator based on an Arduino Uno and SpringRC SR-311 servos as a learning medium for inverse kinematics. A geometric-trigonometric analytical method is used to calculate the angle of each joint from the end-effector target coordinates. A Python interface was developed to serve as both the controller and a comparative 3D visualization tool. Testing was carried out using different target position scenarios. The results show that the IK algorithm works correctly mathematically, but physical deviations occur due to mechanical limitations. The average error in Experiment A was 7.66% (x), 11.38% (y), 9.38% (z), and in Experiment B: 6.34% (x), 5.55% (y), 6.67% (z). The dominant error is caused by servo backlash, link flexibility, and zero-position deviation, rather than algorithmic error. The platform proves effective as a practical learning tool for understanding the relationship between kinematic theory and the limitations of robotics hardware.**

Keywords*—*6-DOF, Arduino, Arm Robot, Inverse Kinematics, Robot Manipulator

I. INTRODUCTION

Robot manipulators play a very important role in various industrial sectors, ranging from manufacturing and logistics to healthcare. As one of the main components in modern automation systems, arm robots are used to perform various tasks that require precision, flexibility, and high efficiency. Along with the increasing use of arm robots in industry, the interest of students and researchers in studying robotics technology, particularly robot manipulators, has also been growing [1]. One fundamental aspect of controlling a robot manipulator is kinematics, which studies the relationship between the robot's joint motion and the position and orientation of the end-effector [2]. Inverse kinematics is a kinematic problem aimed at determining the joint angle configuration of a robot so that the end-effector can reach the desired position and orientation in the workspace [3]. Understanding inverse kinematics is very important because this concept is widely used in motion planning and control of robot manipulators.

Although various studies have discussed the implementation of inverse kinematics on robot manipulators, most of these studies focus on simulations or high-precision industrial robots [4][5]. Research on the application of inverse kinematics to low-cost educational arm robots is still relatively limited, particularly studies discussing the effect of mechanical limitations on robot movement accuracy.

Therefore, this research focuses on the design and implementation of a six-degree-of-freedom (6-DOF) robot manipulator as a learning medium for inverse kinematics. The arm robot system was developed using an Arduino Uno controller and SpringRC SR-311 servo actuators. The inverse kinematics method used is a trigonometry-based analytical approach to determine the angle of each joint based on the end-effector target coordinates.

This research aims to provide a hands-on learning experience for students in understanding the concept of inverse kinematics as well as the practical aspects of designing and controlling a robot manipulator. In addition, this research also analyzes the level of end-effector position error that occurs due to mechanical limitations, such as backlash in the servo gears. Thus, the results of this research are expected to serve as a reference and basis for further development in designing more accurate and reliable educational arm robots.

Unlike previous research, which generally focuses on simulations or high-cost industrial robots [4][5], this research contributes in three main areas: (1) the design of a reproducible, low-cost-component-based educational 6-DOF arm robot platform; (2) the implementation and validation of a geometric-trigonometric analytical inverse kinematics algorithm directly on hardware; and (3) a quantitative analysis of the effect of mechanical limitations on end-effector position accuracy as learning material. Thus, this research fills the gap between kinematic theory learning and practical understanding of the limitations of real robotics systems.

II. METHOD

Specifically, the method used in this research is the Geometric-Trigonometric Closed-Form Analytical Method, an analytical approach to solving inverse kinematics by utilizing the geometric relationships between arm segments in the vertical (x–z) and horizontal (x–y) planes. This method was chosen because it produces a direct (closed-form) solution without iteration, resulting in faster and more stable computation for serially configured robot structures.

In a robot manipulator system, predetermined motion planning is translated into simultaneous movement of each joint's actuator [7]. The main objective of inverse kinematics is to determine the joint angle vector (degree of

freedom/DoF) that allows the end-effector to reach the desired position and orientation [8].

Kinematics itself is a branch of robotics that studies the relationship between the motion and position of a robot, particularly the movement of joints and actuators, without considering force or torque [9]. In a six-axis (6-DOF) robot arm, inverse kinematics refers to the process of calculating each joint angle based on the position and orientation of the end-effector defined in three-dimensional space [10]. Mathematically, inverse kinematics for a 6-DOF robot can be solved using various approaches, such as geometric methods, vector analysis, and numerical methods. This research uses an analytical method based on geometry and trigonometry because this method offers the advantage of high computational speed and solution stability, particularly for robot structures with a closed-form solution [11] [12]. Once the joint angles are obtained, the robot can control each actuator to perform various tasks such as picking up and moving objects [13].

A. Design of the 6-DOF Robot Arm

This research focuses on the design and implementation of the inverse kinematics algorithm and trajectory planning for a 6-DOF robot arm. The arm robot developed is used as a learning medium for understanding the concept of inverse kinematics in practice.

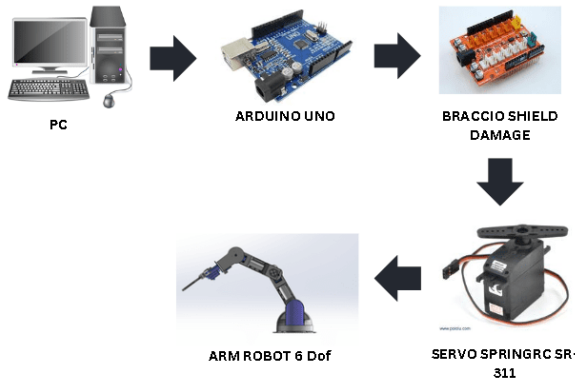


Fig. 1. Block diagram of the 6-DOF arm robot hardware system.

Fig. 1 shows the main components used in this research. The Arduino Uno is used as the system's main controller, while the Braccio Shield serves as an interface between the microcontroller and the actuators. The robot arm is driven by six SpringRC SR-311 servo motors, each acting as the actuator for one joint. This approach enables the development of a simple, effective, and easily understood inverse kinematics system as a robotics learning platform.

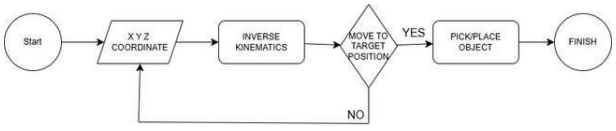


Fig. 2. Robot Arm System Process Flow

The workflow of the arm robot system is shown through a process diagram illustrating the stages of robot motion control. The system receives input in the form of target position coordinates (x, y, z) and the end-effector orientation angle. These parameters are then processed using the inverse kinematics algorithm to produce the angle for each joint, which is subsequently used as a servo motor control command so that the robot arm moves toward the desired position, as shown in Fig. 2.

In addition to the microcontroller-based control system, this research also utilizes Python software as a high-level controller and system evaluation tool. The Python software is used to input the end-effector target position coordinates (x, y, z) and orientation parameters, which are then sent to the robot system via serial communication.

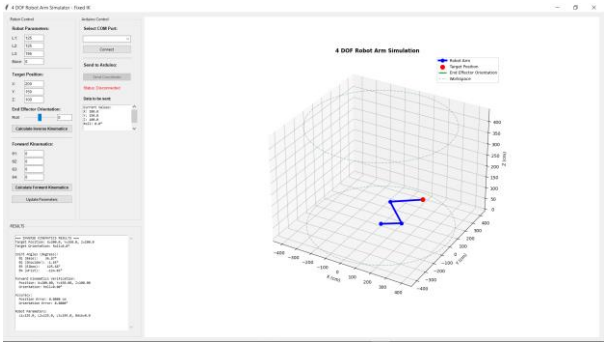


Fig. 3. Python Software for Controlling the 6-DOF Arm Robot

On the software side, the same inverse kinematics algorithm is also implemented to produce a three-dimensional visualization of the end-effector position, as shown in Fig. 3. This visualization is not used to directly correct the robot's movement, but rather serves as a reference for comparison against the actual end-effector position achieved by the robot.

Thus, the main role of the Python software in this research is as a medium for testing, validation, and performance analysis of inverse kinematics. System accuracy evaluation is carried out by comparing the theoretically calculated target position with the physically measured actual end-effector position, without applying a feedback-based correction mechanism (closed-loop control).

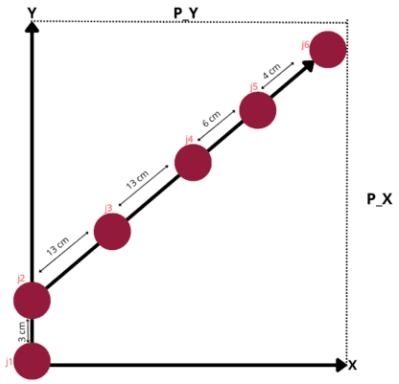


Fig. 4. Geometric representation of the 6-DOF Arm robot configuration on the X–Y plane.

The robot's geometric configuration is determined based on the distance between joints and the length of each arm segment. This information is used to simplify the calculation of the robot's center coordinates and the application of the inverse kinematics algorithm. By knowing these parameters, the robot's workspace can be systematically analyzed and determined, as shown in Fig. 4 and summarized in TABLE I.

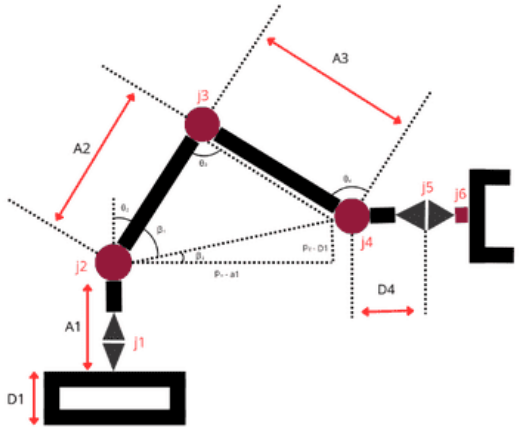


Fig. 5. Kinematic diagram of the 6-DOF Arm robot with angle and offset parameters

The relationship between the angles and lengths of the robot's segments is shown through the kinematic diagram. Angles A1, A2, and A3 respectively represent the angles between joints J1–J2, J2–J3, and J3–J4, while D4 indicates the offset value between joints J4 and J5. The red arrows in the diagram indicate segment length and offset, while the dashed lines illustrate the range of motion of each joint. This visualization provides a comprehensive understanding of the movement flexibility and workspace of the 6-DOF Arm robot, as shown in Fig. 5.

TABLE I
GEOMETRIC PARAMETERS OF THE 6-DOF ARM ROBOT

Joint	Maximum Degrees	*Arm* Length	Offset	Theta	Link		
					A1	A2	A3
1	180°	3 cm	D1 = 3 cm	Ø1	3 cm		
2	165°	13 cm	0	Ø2		13 cm	
3	180°	13 cm	0	Ø3			13 cm
4	180°	6 cm	0	Ø4			
5	180°	4 cm	D4 = 6 cm	Ø5			
6	73°	0 cm	0	Ø6			

Table 1 describes the main parameters of the 6-DOF robot arm, which consists of six joints. Each joint has characteristics including a maximum rotation limit, arm length, offset value, and joint angle (θ). These parameters play an important role in determining the robot's overall range of motion and kinematic configuration.

The first joint has a maximum rotation limit of 180° with an arm length of 3 cm and an offset of 3 cm, indicating a positional shift from the robot's base reference point. The second joint has a maximum rotation limit of 165° with an arm length of 13 cm and no offset, so there is no additional shift from the previous reference point. The third joint has similar characteristics, with a maximum rotation limit of 180°, an arm length of 13 cm, and no offset. Furthermore, the fourth joint has a maximum rotation limit of 180° with an arm length of 6 cm and no offset, functioning as the connector between the main arm and the robot's wrist.

The fifth joint has a maximum rotation limit of 180° with an arm length of 4 cm and an offset of 6 cm, indicating an additional shift relative to the position of the previous joint and playing a role in setting the end-effector orientation. The last joint, the sixth, has a maximum rotation limit of 73° with no arm length or offset, and functions as the direct controller of the gripper or end-effector.

Understanding the geometric parameters and range-of-motion limits of each joint is very important in the design process, kinematic analysis, and implementation of the inverse kinematics algorithm. By accurately knowing these parameters, the robot arm system can be designed and implemented more precisely so that it can achieve the desired movement and position according to the established specifications.

Before the inverse kinematics calculation is performed, the robot's kinematic parameters are defined using the Denavit-Hartenberg (DH) convention. Table II summarizes the DH parameters for each joint of the 6-DOF robot arm used in this research.

TABLE II
DENAVIT-HARTENBERG PARAMETERS OF THE 6-DOF ROBOT ARM

Joint	θ_i	d_i (mm)	a_i (mm)	α_i (°)
1	θ_1	30	0	90°
2	θ_2	0	130	0°
3	θ_3	0	130	0°
4	θ_4	0	60	90°
5	θ_5	60	0	-90°
6	θ_6	0	0	0°

B. *Inverse Kinematics Calculation*

The overall inverse kinematics calculation flow is shown in Fig. 6. The system receives input in the form of the Cartesian target coordinates of the end-effector (P_x, P_y, Pz), then calculates the angle of each joint sequentially using a geometric-trigonometric approach. The calculation begins with Joint 1, which determines the horizontal rotation direction, followed by the calculation of auxiliary angles (β_1 and β_2) as the basis for Joint 2, then Joints 3 and 4, which set the vertical position and orientation of the arm. Joints 5 and 6 are controlled independently to set the gripper's orientation and opening. The final output is the joint angle vector [θ_1, θ_2, θ_3, θ_4, θ_5, θ_6], which is sent to the Arduino Uno to simultaneously drive the six servo actuators.

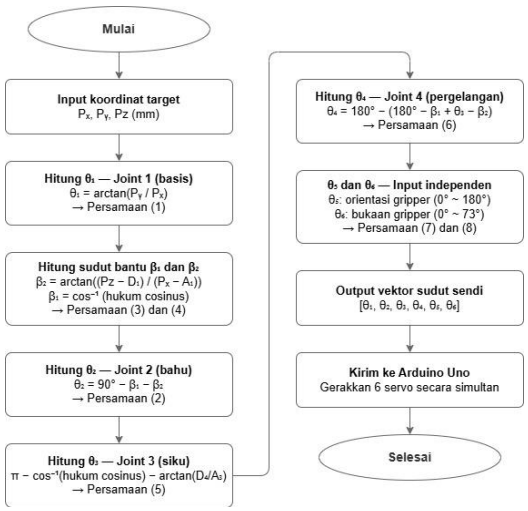


Fig. 6. Inverse Kinematics Calculation Flow for the 6-DOF Robot Arm

Joint 1

The first joint functions as the base joint, allowing the robot to rotate about the vertical axis (z-axis). Angle θ_1 is determined based on the projection of the target position onto the horizontal plane (x, y plane). This angle is calculated using a two-variable arctangent function to ensure the correct angle quadrant, as follows:

$$\emptyset_1 = \tan^{-1}\left(\frac{P_Y}{P_X}\right) \quad (1)$$

Angle θ_1 represents the direction of the robot base's rotation so that the arm can face directly toward the target position on the horizontal plane.

Joint 2

The second joint functions as the shoulder joint, supporting the entire upper arm, forearm, and end-effector. Angle θ_2 is determined based on the geometric relationship between the target position and the robot's link length on the vertical plane (X–Z plane).

Angle θ_2 is calculated using a combination of two auxiliary angles, β_1 and β_2, with the equation:

$$\emptyset_2 = 90 - \beta_1 - \beta_2 \quad (2)$$

Angle β_2 represents the elevation angle between the horizontal line and the line connecting the shoulder joint to the target position on the vertical plane, formulated as:

$$\beta_2 = \tan^{-1}\left(\frac{P_Z - D_1}{P_X - A_1}\right) \quad (3)$$

Meanwhile, angle β_1 is calculated using the law of cosines to determine the relationship between the link length and the target distance relative to the shoulder joint, namely:

$$\beta_1 = \cos^{-1}\left(\frac{A_2{}^2 + ((P_X - A_1)^2 + (P_Z - D_1)^2) - (D_4{}^2 - A_3{}^2)}{2A_2\sqrt{(P_X - A_1)^2 + (P_Z - D_1)^2)}}\right) \quad (4)$$

Angle θ_2 at Joint 2 is calculated based on the end-effector target position on the vertical plane (x, z plane) [14]. This calculation involves two auxiliary angles, β_1 and β_2, which respectively represent the geometric contribution of the vertical and horizontal distances to the target position. Angle β_2 is obtained using an arctangent function that describes the slope of the line connecting the shoulder joint and the target relative to the horizontal axis, taking into account the robot's base height (D_1) and the length of the arm's initial segment (A_1).

Meanwhile, angle β_1 is calculated using the law of cosines to determine the relationship between the arm's link length and the target distance from the shoulder joint [15]. By combining angles β_1 and β_2, angle θ_2 can be determined geometrically so that the robot's upper arm can move consistently toward the target position in the vertical direction, in accordance with the robot's mechanical configuration.

Joint 3

The third joint functions as the elbow joint, which controls the fold between the robot's upper arm and forearm [16]. Angle θ_3 plays an important role in determining the arm's range of motion and the end-effector's final position relative to the target.

The calculation of angle θ_3 is performed using a geometric approach based on the law of cosines, taking into account the arm's link length and mechanical offset in the robot's structure. In general, angle θ_3 is formulated as:

$$\emptyset_3 = \pi - \left(\frac{(A_2 + (D_4{}^2 - A_3{}^2)) - ((P_z - D_1) + (P_z - A_1)^2)}{2\,A_2\,(D_4{}^2 - A_3{}^2)}\right) - \tan^{-1}\left(\frac{D_4}{A_3}\right) \quad (5)$$

The formula for $\emptyset_3$ is used to calculate the angle at the third joint (Joint 3) of the robot arm, taking into account several important parameters related to the robot's geometric configuration. A constant is used in the angle calculation, expressed in radians. Parameter A_2 represents the arm length connecting the second and third joints. The difference between $D_4{}^2$ and $A_3{}^2$ represents the mechanical offset contribution between the fourth and fifth joints, while also accounting for the arm segment length between the third and fourth joints.

The vertical and horizontal position components are used to represent the vertical position of the target point relative to the robot's base height, as well as the effect of the arm's initial segment length on that position. The combination of all these parameters in Equation (5) is used to determine the contribution of each factor in calculating angle $\emptyset_3$. In addition, the arctangent function $\tan^{-1}(D_4/A_3)$ is used to calculate the additional angle arising from the offset D_4 relative to arm length A_3.

Overall, this formulation of angle $\emptyset_3$ enables the robot arm to reach the desired target position by comprehensively accounting for the arm segment lengths, mechanical offsets, and target coordinates in three-dimensional space.

Joint 4

The angle at the fourth joint (Joint 4) functions to adjust the robot's wrist orientation so that it aligns with the arm configuration formed by the previous joints. The calculation of angle $\emptyset_4$ is based on the geometric relationship between the angles at the second and third joints, so that the end-effector's orientation remains consistent with the target position [17]. Angle $\emptyset_4$ is calculated using the following equation:

$$\emptyset_4 = 180 - ((180 - (\beta_1 + \emptyset_3)) - \beta_2) \quad (6)$$

This equation is derived by taking into account the contributions of angles β_1 and θ_3, which represent the cumulative direction of the robot's main arm relative to the vertical plane. The sum of β_1 and θ_3 indicates the total angle formed by the arm segments up to the third joint [18]. This value is then corrected against a reference angle of 180° to maintain consistency with the mechanical angle convention used in the robot.

Next, angle β_2 is subtracted to compensate for the elevation angle produced by the second joint [16]. This correction process ensures that the fourth joint's orientation does not overlap with the angular contribution from the previous joint. The final subtraction from 180° aims to keep angle $\emptyset_4$ within the servo's mechanical working range and consistent with the

robot's kinematic configuration.

With this approach, angle $\emptyset_4$ allows the wrist to harmoniously adjust its orientation relative to the arm's position, so that the end-effector can maintain the desired orientation when reaching a target in the workspace.

Joint 5

The fifth joint (Joint 5) functions as the wrist joint responsible for controlling the end-effector's rotational orientation about the arm's longitudinal axis. Unlike the previous joints, which mainly determine the end-effector's position, this joint focuses more on orientation control, particularly the gripper's rotation angle to suit the object manipulation requirements. The working angle range of the fifth joint is expressed as follows:

$$\emptyset_5 = (0° \sim 180°) \quad (7)$$

This angle range is determined by the mechanical limitations of the servo motor used, as well as the physical design of the robot arm. A change in the value of angle $\emptyset_5$ does not affect the end-effector's translational position, but instead changes the gripper's rotational orientation relative to the object to be manipulated. Therefore, the angle at the fifth joint is generally controlled separately from the position calculation in the main inverse kinematics algorithm.

In this research, angle $\emptyset_5$ is used to adjust the gripper's direction so that it is aligned with the target object before the pick-up or transfer process is carried out. This angle setting allows the robot to manipulate objects more flexibly without changing the previously determined end-effector position.

Joint 6

The sixth joint (Joint 6) functions as the gripper actuator that controls the opening and closing mechanism of the end-effector. Unlike the previous joints, which play a role in determining the position and orientation of the robot arm, this joint is specifically responsible for directly performing object manipulation actions.

The working angle range of the sixth joint is expressed as follows:

$$\emptyset_6 = \text{Angle } (0° \sim 73°) \quad (8)$$

This angle limit is determined by the mechanical limitations of the servo motor and the physical design of the gripper used. A change in angle $\emptyset_6$ does not affect the end-effector's position or orientation, but instead controls the degree of gripper opening. At angles close to 0°, the gripper is in an open condition, allowing the robot to reach and pick up objects. Conversely, at angles close to 73°, the gripper is in a closed condition, allowing the robot to grip the object during the transfer process.

Setting the angle of this sixth joint is an important element in the success of manipulation tasks, such as picking up, moving, and releasing objects. Therefore, control of the sixth joint is performed separately from the position inverse kinematics calculation, while remaining integrated within the overall robot control system.

III. Results and Discussion

A. Test System Description

In this research, the testing system consists of two main components: Python-based control software and the robot control system. The Python software functions as a user interface used to input target parameters in the form of Cartesian position coordinates (x, y, z) and end-effector orientation angles. This data is then sent to the robot via serial communication.

The robot receives the end-effector position and orientation data, then independently performs the inverse kinematics calculation to determine the angle of each joint. The results of this inverse kinematics calculation are used by the robot to drive the servo actuators. In parallel, the Python software displays a three-dimensional (3D) visualization of the same inverse kinematics calculation results, which is then used as a visual comparison against the robot's actual end-effector position.

B. Testing Methodology and Actual Position Measurement

Testing was carried out by determining several end-effector target position points within the robot's workspace. For each target point, the robot was moved from the same initial position (home position) to minimize the effect of initial conditions on the test results.

The actual end-effector position was measured manually using a linear measuring instrument (precision ruler/tape measure), using the same coordinate reference system as the input coordinate system. Each test was performed once for each target point, so the error value obtained represents the result of a single execution per position. This approach was chosen to evaluate the basic performance of the inverse kinematics system without statistical averaging.

Although this method is sufficient for initial observation, the limited number of measurement repetitions is a factor that may affect the reproducibility of the experimental results.

C. Definition and Calculation of Position Error

End-effector position error is defined as the relative difference between the target position and the actual position on each Cartesian axis. The error percentage is calculated using the following equation:

$$Error(\%) = \left|\frac{P_{aktual} - P_{target}}{P_{target}}\right| x\ 100\% \quad (9)$$

where:

- P_{target} is the position value entered via the Python software
- P_{actual} is the *end-effector* position as physically measured

Errors are calculated separately for the x, y, and z axes, allowing for an analysis of error distribution in each direction of motion. The error value on the z-axis, reaching up to 9.38%, indicates a significant vertical deviation that cannot be ignored and requires further discussion regarding its cause.

TABLE III
POINT A MOVEMENT TEST

Input Value (mm)			Actual Position End Effector (mm)		
x	y	z	x	y	z
150	200	200	130	200	230
160	200	200	155	209	210
170	200	200	170	220	200
180	200	200	175	225	200
190	200	200	190	221	230
200	200	200	180	233	210
210	200	200	200	234	270
220	200	200	280	240	200

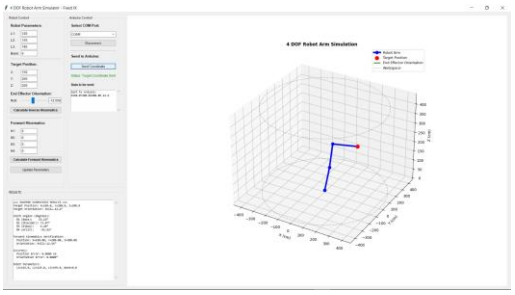
TABLE IV
POINT A ERROR VALUE

NO	Error Value (mm)			Error Value (%)		
	x	y	z	x	y	z
1	20	0	30	13,33	0	15
2	5	9	10	3,13	4,5	5
3	0	20	0	0	10	0
4	5	25	0	2,78	12,5	0
5	0	21	30	0	10,5	15
6	20	33	10	10	16,5	5
7	10	34	70	4,76	17	35
8	60	40	0	27,27	20	0
Average Error	15	22,75	18,75	7,66	11,38	9,38

(a) X-axis Measurement (127)

(b) Y-axis Measurement (205)

(c) Z-axis Measurement (230)



(d) Python GUI Simulation Result

Fig. 7. Experiment A Position, x: 150, y: 200, z: 200

TABLE V
POINT B MOVEMENT TEST

Input Value (mm)			Actual Position End Effector (mm)		
x	y	z	x	y	z
100	250	300	115	270	320
110	250	300	120	270	320
120	250	300	120	275	320
130	250	300	130	273	320
140	250	300	150	260	320
150	250	300	160	255	320
160	250	300	171	246	320
270	250	300	286	246	320

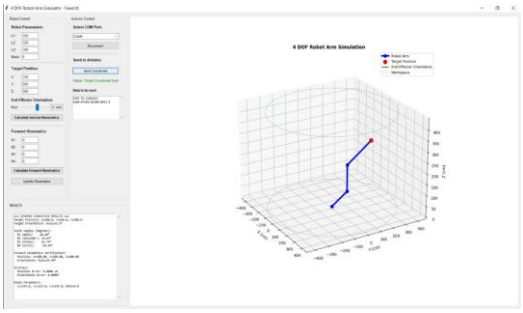
TABLE VI
POINT B ERROR VALUE

NO	Error Value (mm)			Error Value (%)		
	x	y	z	x	y	z
1	15	20	20	15	8	6,67
2	10	20	20	9,09	8	6,67
3	0	25	20	0	10	6,67
4	0	23	20	0	9,2	6,67
5	10	10	20	7,14	4	6,67
6	10	5	20	6,67	2	6,67
7	11	3	20	6,88	1,6	6,67
8	16	3	20	5,93	1,6	6,67
Average Error	9	13,62	20	6,34	5,55	6,67

(a) X-axis Measurement (115)

(b) Y-axis Measurement (270)

(c) Z-axis Measurement (320)



(d) Python GUI Simulation Result

Fig. 8. Experiment B Position, x: 100, y: 250, z: 200

D. *Analysis of Position A and Position B Test Results*

Based on the test results in Experiments A and B, different end-effector position error characteristics were obtained for each coordinate axis. In Experiment A, the average error was 7.66% on the x-axis, 11.38% on the y-axis, and 9.38% on the z-axis, with the largest error occurring on the y-axis. Meanwhile, in Experiment B, the average error obtained was relatively lower and more evenly distributed, at 6.34% on the x-axis, 5.55% on the y-axis, and 6.67% on the z-axis.

When compared with the inverse kinematics calculation results displayed in the Python software simulation, the theoretically calculated end-effector position shows agreement with the given target position. This indicates that the inverse kinematics algorithm used works correctly, mathematically. The differences observed in the actual end-effector position measurements are mainly influenced by the mechanical condition of the robot system, such as actuator precision limitations and mechanical tolerances at each joint, rather than kinematic calculation errors. Thus, the observed inaccuracy better represents the system's mechanical performance rather than the performance of the inverse kinematics algorithm itself.

E. *Causes of Error and System Limitations*

The observed position error cannot be explained by a single factor alone. Although backlash in the servo gears is suspected to contribute to the error, particularly during changes in rotation direction, this cannot yet be considered the sole primary cause.

Several other factors that potentially affect the test results include:

1. Servo angle calibration errors, where the servo's zero angle does not fully match the theoretical angle.
2. Servo motor resolution and nonlinearity, which limit the precision of the actual angle.
3. Mechanical flexibility in the links and joints, especially when the robot reaches positions farther from the base.

The high error on the z-axis indicates that the end-effector's vertical movement is highly sensitive to joint angle errors, particularly for joints that contribute to the arm's elevation.

Fig. 9. Zero Position of the 6-DOF Arm Robot

Fig. 9 shows the robot's actual condition when set to the zero position. Theoretically, this configuration should produce an arm position perpendicular to the base plane. However, observations show a fairly significant angular deviation. This deviation indicates limitations in the robot's mechanical system, particularly related to component-mounting tolerances and backlash in the servo gears.

This condition causes the initial angle of each joint to not fully match the angle value assumed in the inverse kinematics calculation. As a result, position error accumulates along the kinematic chain, especially in the end-effector's vertical movement. This phenomenon provides visual justification for the magnitude of the error observed on the z-axis in the test results, and reinforces the analysis that mechanical factors contribute significantly to the system's accuracy limitations.

Based on the test results and analysis performed, it can be concluded that the implemented inverse kinematics system on the robot is able to produce end-effector movement that approaches the target position, but still shows relatively large errors, especially on the vertical axis. The difference in error between Experiments A and B confirms that the direction of movement has an influence on system accuracy.

These results show that although the implemented inverse kinematics approach already functions conceptually, the robot's actual performance is still affected by various mechanical and actuation limitations that have not been explicitly compensated for.

IV. CONCLUSION

Based on the research results, it can be concluded that this 6-DOF robot manipulator system was successfully implemented as a functional learning platform. The analytical algorithm proved effective at computationally translating target coordinates into joint angles. However, physical testing showed an accumulated average error ranging from 5.55% to 11.38%, with the highest deviation on the vertical (z) axis. This is caused by non-ideal mechanical factors, including servo motor backlash, limited actuator resolution, and initial calibration misalignment. For future research, it is

recommended to implement a closed-loop control mechanism based on position feedback sensors to compensate for mechanical error in real time. In addition, exploration of numerical inverse kinematics methods, such as Jacobian Pseudoinverse or machine learning-based approaches, could be compared with the analytical approach used in this research to improve the accuracy and flexibility of the system in more complex workspaces.

REFERENCES

[1] S. Kucuk and Z. Bingul, *Robot Kinematics: Forward and Inverse Kinematics*, no. December. 2006.

[2] A. T. Hasan, A. M. S. Hamouda, and N. Ismail, "An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 D . O . F serial robot manipulator," vol. 37, pp. 432–438, 2006, doi: 10.1016/j.advengsoft.2005.09.010.

[3] A. Khatamian, "Solving Kinematics Problems of a 6-DOF Robot Manipulator," pp. 228–233.

[4] I. Eka and T. Agustinah, "pada Robot Denso Manipulator," vol. 4, no. 1, 2015.

[5] Z. Hidayat, M. Sahal, Y. Bilfaqih, R. Eak, and D. C. Sirait, "Perancangan Jaringan Saraf Tiruan untuk Menyelesaikan Kinematika Balik Manipulator Robot Denso 6-DoF," vol. 17, no. 2, pp. 2–9, 2023.

[6] I. Sulaeman, A. W. Dani, T. Pangaribowo, and F. Sirait, "Analisa Inverse Kinematics Pada Prototype 3-DOF *Arm* Robot Dengan Metode ANFIS," no. February 2022, pp. 13–18, 2023, doi: 10.22441/jte.2022.v13i1.003.

[7] R. Biped and T. Koordinat, "PENERAPAN INVERS KINEMATIKA UNTUK PERGERAKAN KAKI ROBOT BIPED," no. November, pp. 1–9, 2015.

[8] J. Iqbal, R. Islam, and H. Khan, "Modeling and analysis of a 6 DOF robotic *arm* manipulator Modeling and Analysis of a 6 DOF Robotic *Arm* Manipulator," vol. 3, no. October, 2015.

[9] M. Ali *et al.*, "INVERS KINEMATIK ROBOT *ARM* 4 DOF MENGUNAKAN SENSOR LEAP MOTION," vol. 6, no. 1, pp. 363–371, 2020.

[10] M. A. N. Huda, S. H. Susilo, and P. M. Adhi, "Implementation of Inverse Kinematic and Trajectory Planning on 6-DOF Robotic *Arm* for Straight-Flat Welding Movement," *Log. J. Ranc. Bangun dan Teknol.*, vol. 22, no. 1, pp. 51–61, 2022, doi: 10.31940/logic.v22i1.51-61.

[11] I. Agustian, N. Daratha, R. Faurina, A. Suandi, and S. Sulistyaningsih, "Robot Manipulator Control with Inverse Kinematics PD-Pseudoinverse Jacobian and Forward Kinematics Denavit Hartenberg," *J. Elektron. dan Telekomun.*, vol. 21, no. 1, p. 8, 2021, doi: 10.14203/jet.v21.8-18.

[12] R. Jhala, "Inverse Kinematics and Trajectory Planning of 5 DoF Scorbot-ER Vplus 1Rajsinh Jhala 1Research Scholar 1Birla Vishvak*arm*a Mahavidhyalaya," vol. 8, no. 3, pp. 442–449, 2012.

[13] S. Z. Saeed Al-khayyt, "Creating Through Points in Linear Function with Parabolic Blends Path by Optimization Method," *Al-Khwarizmi Eng. J.*, vol. 14, no. 1, pp. 77–89, 2018, doi: 10.22153/kej.2018.10.005.

[14] J. M. Mukherjee *et al.*, "Design and Analysis of six DOF Robotic Manipulator", doi: 10.1088/1757-899X/1055/1/012005.

[15] S. I. Afi, P. Studi, T. Elektro, F. Teknik, and U. M. Surakarta, "ROBOT LENGAN 5 DOF DENGAN INVERS KINEMATIK BERBASIS ROBOT LENGAN 5 DOF DENGAN INVERS KINEMATIK BERBASIS," 2022.

[16] B. Utomo *et al.*, "Analisa Forward dan Inverse Kinematics pada Simulator *Arm* Robot 5 Derajat Kebebasan," vol. 1, no. 3, pp. 11–20, 2013.

[17] A. D. Setiyadi and I. Setiawan, "PERANCANGAN DAN PENGENDALIAN MANIPULATOR ROBOT 4-DOF DENGAN GRIPPER BERBASIS INVERSE KINEMATICS DAN TRAJECTORY PLANNING DENGAN ROS," vol. 10, no. 4, 2021.

[18] E. Prasetya, "IMPLEMENTASI INVERSE KINEMATIC PADA PERGERAKAN MOBILE ROBOT KRPAI DIVISI BERKAKI," *Publ. J. Skripsi Univ. Brawijaya*, 2024.