Implementation of 5-DOF Robot Arm Control with Inverse Kinematics Through Interactive GUI

Ryan Satria Wijaya[1*], Muhamad Ilham[1], Eko Rudiawan Jamzuri[1], Anugerah Wibisana[1], Rifky Afriza[1], Emelia Rosari Siregar[1], and Senanjung Prayoga[1]

[1]Robotics Engineering Technology Study Program, Electrical Engineering Department, Politeknik Negeri Batam, Batam, Indonesia

[*]Email: ryan@polibatam.ac.id

Received on 10-03-2026 | Revised on 24-06-2026 | Accepted on 26-06-2026

***Abstract*—This work presents a control system for a five-joint (5-DOF) robotic arm designed for educational use, combining geometric inverse kinematics with a custom Python graphical interface. Instead of relying on iterative approaches, a closed-form inverse kinematics formulation is implemented to directly derive joint configurations from Cartesian coordinates. System functionality is supported through communication between a host computer and an ESP32 microcontroller via UART serial interface, enabling real-time control, workspace verification, and kinematic validation using forward kinematics. For simplicity in operation, the wrist orientation is kept fixed so that the control system focuses primarily on positional accuracy. experimental results show an average IK computation time of 1,7 ms, a mean positioning error of 2,46 mm, and a peak deviation of 4,91 mm across representative workspace targets. The obtained results confirm that the system achieves stable and reliable performance suitable for low-cost laboratory experimentation and educational robotics applications.**

Keywords: 5-DOF Robot Arm, Educational Robotics, ESP32, Geometric Inverse Kinematics, Interactive GUI, Position Control.

I. INTRODUCTION

Robotic system Have become increasingly important in both industrial and educational environments, leading to a growing need for control systems that are accurate, efficient, and easy to use [1]. Five-degree-of-freedom (5-DOF) manipulators are popular because they strike a compromise between a straightforward mechanical design and adequate flexibility, making them appropriate for activities like pick-and-place operations, motion demonstrations, and lab investigations [2]. Despite these advantages, developing an intuitive and computationally efficient an effective control solution for a 5-DOF manipulator is still non-trivial to achieve, particularly in academic environments constrained by limited budgets and low-performance processing hardware [3].

Numerical techniques, such as Jacobian-based or gradient-based approaches, are flexible for complex systems but often involve iterative computation and may result in unstable convergence or inconsistent execution time [4]. In contrast, analytical geometric methods provide direct solutions with faster and more predictable performance, making them well suited for real-time applications, particularly on embedded platforms such as ESP32 microcontrollers with limited computational resources [5].

Educational robotic systems impose specific requirements that differ from those of industrial platforms. Computational efficiency is essential to ensure real-time performance on resource-constrained embedded systems commonly found in academic laboratories. Algorithm transparency is also important, as it allows students to examine, modify, and experiment with control algorithms, encouraging active learning of kinematic principles [6]. Affordability plays a key role in enabling wider adoption in educational institutions with limited budgets. In addition, visualization capabilities help users relate abstract mathematical concepts to actual robot motion, improving understanding of coordinate transformations and kinematic relationships. Although many inverse kinematics implementations exist, integrating these requirements into a single accessible platform remains a challenge [7].

Recent advances in open-source software ecosystem, particularly Python-based frameworks, have enabled the development of interactive and affordable robotic control systems. Tools such as Tkinter and Matplotlib allow the creation of graphical user interfaces and real-time visualization without requiring specialized proprietary software [8]. When combined with embedded without requiring platform such as the ESP32, these tools support real-time communication, data logging, and visualization even on low-cost system [9]. By bridging the gap between theoretical kinematic modeling and practical robotic implementation, this integration allows students to see firsthand how mathematical calculation and actual robot behavior are related [10].

In this context, the present study introduces a modular Python-based graphical user interface control system for a 5-DOF robotic arm that employs geometric inverse kinematics for real-time operation [11]. The system is designed to meet

educational requirements through open-source architecture, hardware accessibility, and interactive visualization. The implemented closed-form geometric inverse kinematics solution achieves an average computation time of 1,7 ms, enabling real-time control on an ESP32 microcontroller without the need for high-performance processing hardware [12]. The system integrates real-time 3D visualization, interactive GUI control, and automated testing features within a unified platform, providing immediate visual feedback that reinforces the connection between kinematic computation and physical motion. The experimental results indicate that all tested positions (25 points) were successfully reached. The system achieved an average positioning error of 2,46 mm, with a standard deviation of 0,88 mm, reflecting consistent performance across the workspace. [13].

II. METHOD

A. System Overview

The proposed 5-DOF robotic arm control system adopts a modular architecture that integrates software processing and hardware execution for real-time operation. As illustrated in Figure 1, the overall system is organized into three primary subsystems: the user interaction and computation unit, the communication module, and the hardware execution unit. This modular approach allows each subsystem to be developed and tested separately while maintaining reliable integration for real-time robotic control [14].

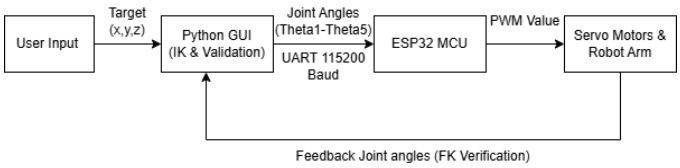


Figure 1. Block diagram of the proposed 5-DOF Robot arm control System

The control process starts with the operator specifying the desired end-effector target in Cartesian coordinates (X, Y, Z) through a Python-based graphical interface. Prior to computing inverse kinematics, the input values are validated to make sure the intended position is within the joint restrictions and the predetermined robot workspace [15]. This validation step prevents unreachable target commands and reduces the risk of singular configuration or mechanical overload. If the target position violates these constraints, the system rejects the command and displays an error notification through the GUI.

Following the validation phase, the necessary joint variables are obtained by processing the target coordinates through a geometric inverse kinematics (IK) model [16]. This IK module determines the joint angles (θ_1–θ_5) using a closed-form analytical method. The arm's arrangement to reach the desired goal position is determined by these calculated angles. A UART communication port running at 115200 bps is then used to transmit these calculated angle values to the integrated controller [17].

Joint angle information generated by the IK module is transmitted to an ESP32 microcontroller, which functions as the hardware execution unit of the system. The ESP32 processes these inputs and converts these signals into PWM duty cycles used to drive each servo motor accordingly. Through this process, the servo motors are driven accordingly, enabling the robotic arm to reach the specified target position [18].

Since the system does not employ physical joint angle sensors, feedback is implemented at the software level through forward kinematics (FK) validation. After command transmission, the same joint angle values are reprocessed by the forward kinematics model within the Python GUI to compute the expected end-effector position [19]. This computed position is compared with the original target coordinates to verify execution consistency and detect potential kinematic discrepancies. Although this approach does not constitute closed-loop hardware control, it provides an effective and lightweight validation mechanism suitable for educational applications [20].

Overall, the proposed system architecture enables deterministic real-time control, clear data flow, and interactive visualization while remaining computationally efficient and accessible. The combination of geometric inverse kinematics, serial communication, and GUI-based validation makes the system well suited for educational robotics, allowing students to directly observe the relationship between kinematic computation and physical robot motion [21].

B. Robot Kinematics Model

The proposed 5-DOF manipulator's kinematic model defines how joint movements map to end-effector location in three-dimensional space [22]. Built around an anthropomorphic configuration with five revolute joints, the system supports spatial positioning while maintaining a fixed wrist orientation. This configuration provides a simple yet functional structure that is well-suited for educational use, including learning, visualization, and laboratory-scale experimentation.

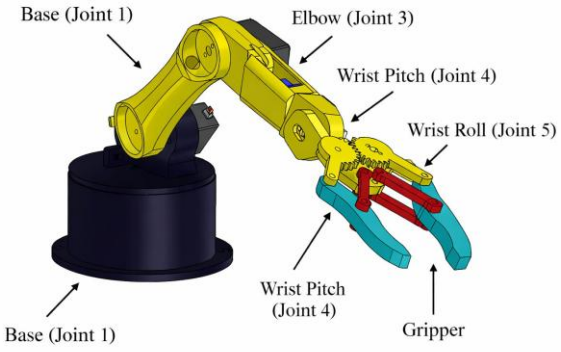


Figure 2. Mechanical structure and joint configuration of the five-joint robotic manipulator

Figure 2 illustrates the physical layout and joint arrangement of the 5-DOF manipulator used in this study. The system consists of five revolute joints that provide multi-axis motion for three-dimensional positioning tasks. The first two joints are positioned at the base and shoulder, followed by the elbow joint responsible for primary arm extension. In the wrist section, two additional joints enable pitch and roll movements, allowing

adjustment of the end-effector orientation. This arrangement offers a balance between workspace coverage and mechanical simplicity, making the system suitable for educational and experimental applications [23].

Each joint is given a coordinate frame according to the Denavit–Hartenberg (DH) convention in order to quantitatively depict the kinematic relationships. In robotic modeling, this standardized approach is frequently used to methodically characterize the spatial relationship between neighboring links. With these frame definitions, joint-to-joint transformations can be derived in a structured manner, providing the foundation for forward and inverse kinematics analysis [24].

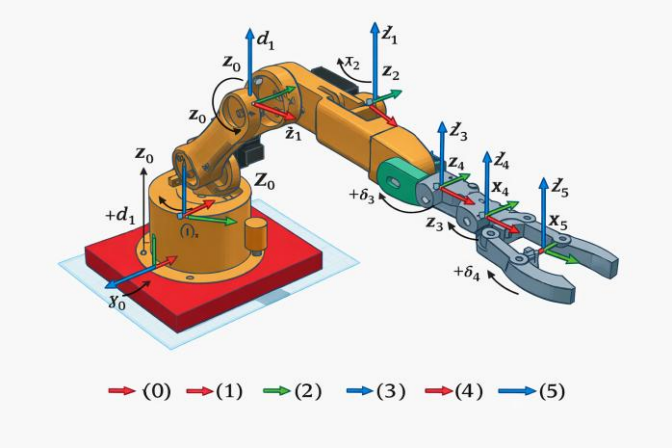


Figure 3. Coordinate Frame robot Arm 5-DOF

As illustrated in Figure 3, the coordinate frames are assigned sequentially from the base to each joint, where frame (0) represents the base and frames (1) to (5) correspond to the joints along the manipulator. The z-axis of each frame is aligned with the joint rotation axis, while the x-axis is defined based on the relative positioning between adjacent joints. This configuration provides a consistent geometric reference for describing joint motion and link relationships.

Based on these coordinate definitions, the kinematic parameters of the manipulator are derived following the Denavit–Hartenberg (DH) convention [25]. Each link is characterized by four quantities: the joint angle (θ), link offset (d), link length (a), and twist angle (α). These values encode the spatial relationship between consecutive links and serve as the building blocks for constructing the transformation matrices.

TABLE I
DENAVIT–HARTENBERG PARAMETERS FOR THE PROPOSED 5-DOF ROBOTIC ARM

Joint	θ_i	d_i	a_i	α_i
Base	θ_1	d_1	0	90
Shoulder	θ_2	0	a_2	0
Elbow	θ_3	0	a_3	0
Wrist Roll	θ_4	0	0	90
Wrist Pitch	$\theta 5$	d_5	0	0

Table I provides a summary of the robotic arm's kinematic parameters according to the DH convention. Each joint is represented by a corresponding angle variable (θ_1 to θ_5), while the geometric characteristics of each link are defined through the associated parameters. In this configuration, θ_1 corresponds to the base rotation, whereas θ_2 and θ_3 describe the motion of the arm segments. The parameter $d$represents the displacement along the joint axis, and the link lengths are defined accordingly. The twist angle α describes the angular relationship between consecutive joint axes based on the assigned coordinate frames.

C. Forward Kinematics Formulation

Forward kinematics computes the end-effector's Cartesian position from a specified set of joint angle values. It is mostly used in this system to validate inverse kinematics results and guarantee consistency during operation. The Denavit–Hartenberg parameters given in Table I and the coordinate frame definitions displayed in Figure 3 serve as the foundation for the formulation.

For a serial-chain robot, the relationship between adjacent coordinate frames is described using a homogeneous transformation matrix:

$$T_0^5 = T_0^1 . T_1^2 . T_2^3 . T_3^4 . T_4^5 \quad (1)$$

From the resulting transformation, the end-effector's Cartesian coordinates are extracted as:

$$X = (a_2 cos\theta_2 + a_3 cos\theta_{23} + d_5 sin\theta_{234}) cos\theta_1 \quad (2)$$
$$Y = (a_2 cos\theta_2 + a_3 cos\theta_{23} + d_5 sin\theta_{234}) sin\theta_1 \quad (3)$$
$$Z = d_1 + a_2 sin\theta_2 + a_3 sin\theta_{23} - d_5 cos\theta_{234} \quad (4)$$

The joint variables and the final end-effector position are related by these equations. Assuming a fixed wrist orientation, θ_4 regulates vertical alignment and θ_5 modifies the end-effector orientation without changing its Cartesian position.

A Python-based forward kinematics module computes the end-effector position after each inverse kinematics step, serving as a validation tool and supporting the geometric inverse kinematics approach described in the next section [26].

D. Geometric Inverse Kinematics Algorithm

The geometric inverse kinematics approach converts a desired Cartesian target (X, Y, Z) into a set of five joint angles θ_1–θ_5 to position the end-effector at the specified target point [27]. The position and orientation can be solved independently by using the manipulator's anthropomorphic geometry, enabling a closed-form solution without iterative calculation. Consequently, the approach offers a steady and predictable execution duration throughout the workspace [28].

1) Step 1: Base Rotation (θ_1)

The base joint rotates the entire arm about the vertical Z-axis to align the robot with the target direction in the horizontal plane. From the forward kinematics formulation, the X and Y components of the end-effector position contain the base rotation terms $\cos\theta_1$and $\sin\theta_1$. Taking the ratio of these components eliminates the common radial term:

$$\frac{Y_{target}}{X_{target}} = tan\,\theta^1 \quad (5)$$

Thus, the base angle is computed as:

$$\theta_1 + atan2(Y_{target}, X_{target}) \quad (6)$$

The atan2 function ensures correct angle determination across all four quadrants of the horizontal plane. This step aligns the arm with the target azimuth and reduces the spatial inverse kinematics problem to a planar problem in the vertical plane containing the target.

2) Step 2: 2D Planar Projection

After determining the base rotation, the inverse kinematics problem reduces to a two-dimensional configuration in the vertical plane. The remaining joints operate within this plane, forming a planar mechanism analogous to a two-revolute (2R) manipulator.

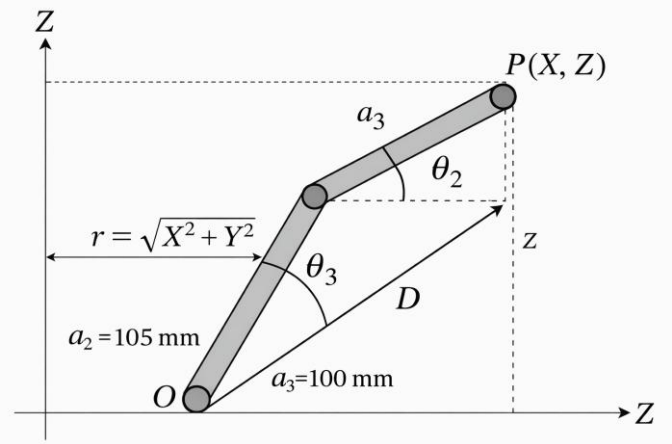


Figure 4. Planar kinematic model of the robot arm after base rotation (2R mechanism projection)

As shown in Figure 3, the horizontal distance from the base axis to the target position is determined as follows:

$$r = \sqrt{(X^2target + Y^2target)} \quad (7)$$

The height difference between the shoulder joint and the target point is given by:

$$z = Z\_target - d_1 \quad (8)$$

where $d_1 = 95$ mm represents the base height offset. The straight-line distance from the shoulder joint to the wrist center point is then:

$$D = \sqrt{(r^2 + z^2)} \quad (9)$$

This planar projection transforms the three-dimensional positioning problem into a classical geometric configuration suitable for analytical solution.

3) Reachability Validation

Before computing joint angles, the algorithm verifies whether the target position lies within the robot's reachable workspace. For a planar two-link mechanism, the reachable region is bounded by the minimum and maximum distances determined by the link lengths.

The minimum reachable distance occurs when the links are folded toward each other:

$$Dmin = | a_2 - a_3 | = | 105 - 100 | = 5\ mm \quad (10)$$

The maximum reachable distance occurs when both links are fully extended:

$$Dmax = a_2 + a_3 = 205\ mm \quad (11)$$

If $D < D_{min}$ or $D > D_{max}$, the target position is considered unreachable, and the algorithm terminates without further computation. This validation step prevents invalid inverse trigonometric operations and avoids commanding physically infeasible robot configurations.

4) Elbow Angle (θ_3)

The shoulder, elbow, and wrist center collectively form a triangle within the vertical plane. This triangle is constructed from the upper arm segment a_2, the forearm segment a_3, and the reach distance D, which together capture the geometric configuration of the manipulator.

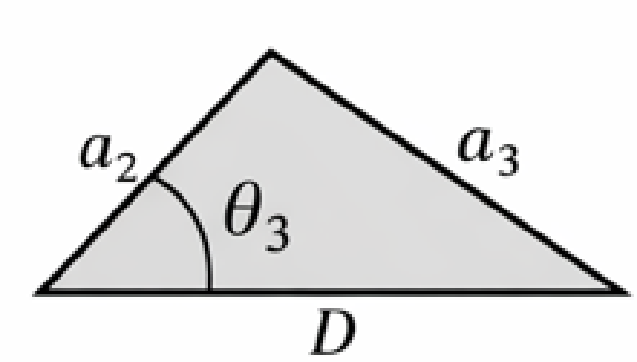


Figure 5. Geometric approach for determining the elbow joint angle

Applying the law of cosines:

$$\cos\theta_3 = \frac{D^2 - \alpha_2^2 - \alpha_3^2}{2 \cdot \alpha_2 \cdot \alpha_3} \quad (12)$$

The elbow angle is obtained as:

$$\theta_3 = arccos\ \frac{D^2 - \alpha_2^2 - \alpha_3^2}{2 \cdot \alpha_2 \cdot \alpha_3} \quad (13)$$

The positive solution corresponds to the elbow-up configuration, which is selected in this implementation to ensure consistent motion behavior and to avoid workspace limitations near singular configurations.

5) Shoulder Angle (θ_2)

The shoulder angle is determined by combining two geometric components. The first component represents the elevation angle from the shoulder to the target point:

$$\phi = atan2(z, r) \quad (14)$$

The second component accounts for the elbow geometry:

$$\psi = \arccos \frac{\alpha_2^2 - D^2 - \alpha_3^2}{2 \cdot \alpha_2 \cdot D} \quad (15)$$

The total shoulder angle is computed as:

$$\theta_2 = \phi + \psi \quad (16)$$

where both angles are measured in the same rotational direction within the vertical plane.

6) Wrist Orientation (θ_4, θ_5)

To maintain a consistent end-effector orientation suitable for educational and pick-and-place tasks, the wrist joints are configured to keep the gripper pointing vertically downward. The wrist roll angle is fixed as:

$$\theta_4 = 90° \quad (17)$$

The wrist pitch angle compensates for the cumulative pitch introduced by the shoulder and elbow joints:

$$\theta_5 = 90° - (\theta_2 + \theta_3) \quad (18)$$

This formulation ensures that the end-effector maintains a consistent vertical orientation across different arm configurations. Alternative orientation strategies can be implemented by modifying the constant offset term in equation (23).

The complete geometric inverse kinematics algorithm computes all five joint angles using closed-form analytical expressions without iterative refinement. This deterministic approach results in predictable computation time within a few milliseconds across the entire workspace, as demonstrated in the experimental results section. The algorithm is implemented within the Python-based GUI and supports real-time operation on low-cost embedded hardware [29].

E. GUI Implementation

The graphical user interface (GUI) is developed to provide an intuitive interaction layer for controlling the 5-DOF robotic arm control system [30]. Implemented in Python, the GUI serves as the primary platform for defining target end-effector positions in Cartesian coordinates, executing inverse kinematics computation, visualizing robot motion, and managing communication with the embedded controller [31]. Before inverse kinematics computation is performed, the input values are validated against predefined workspace limits and joint constraints to prevent unreachable or unsafe commands [32]. This design enables safe and reliable real-time operation while allowing users to interact with the robotic system without requiring low-level programming knowledge, making it suitable for both experimental use and educational applications [33].

Once a valid target position is entered, the GUI invokes the geometric inverse kinematics algorithm to compute the corresponding joint angles. The computed joint angles are displayed within the interface and transmitted to the ESP32 microcontroller through serial communication using the UART protocol at a baud rate of 115200 bps. This explicit presentation allows users to observe the relationship between Cartesian input commands and joint-level motion [34].

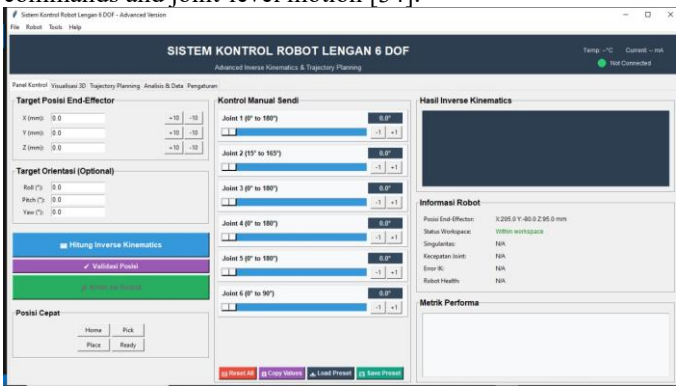


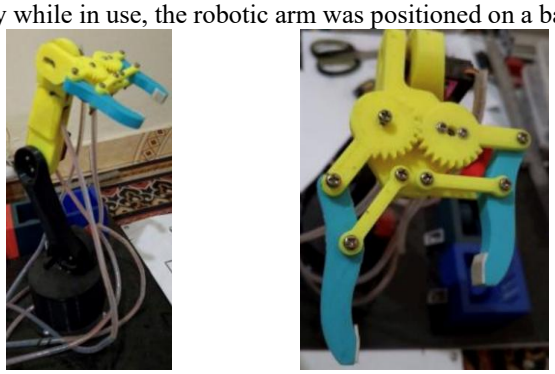
Figure 6. Graphical user interface of the 5-DOF robot arm control system

Upon receiving a valid target coordinate, the system automatically triggers the geometric IK module to calculate the required joint angles. The resulting angle values are then displayed on the interface and forwarded to the ESP32 microcontroller via UART serial communication at a baud rate of 115200 bps, allowing users to directly monitor the mapping between Cartesian input coordinates and the corresponding joint-level motion commands [35].

III. RESULT AND DISCUSSION

A. Experiment Setup

The experimental setup was created to assess the real-world operational performance of the designed 5-DOF arm control platform and validate the geometric inverse kinematics algorithm under actual operating conditions. Every experiment was carried out in a lab setting, and to guarantee mechanical stability while in use, the robotic arm was positioned on a base.



(a) Isometric View (a) Top View

Figure 7. Real Manipulator Robot Installed (a) Isometric View (b) Top View.

The robotic platform consists of a 5-DOF actuated by servo motors and controlled by an ESP32 microcontroller as shown in figure 7. Joints 1–3 are driven by TD8120MG servo motors to provide sufficient torque for base rotation and main arm motion, while joints 4–5 utilize MG90S servo motors for wrist pitch and roll. All servo motors are driven using standard PWM signals generated by the microcontroller, enabling rotation within an approximate range of 180°. The system does not employ physical joint position sensors, aligning with the low-cost and educational objectives of the platform.

Control software is executed on a host computer and implemented using Python, integrating the graphical user interface, inverse kinematics computation, forward kinematics validation, and serial communication within a single application. Communication between the host computer and the ESP32 microcontroller is established via UART at a baud rate of 115200 bps. Joint angle commands are transmitted only after the target position is validated to be within the reachable workspace.

To evaluate system behavior systematically, representative test positions were selected to cover different regions of the workspace rather than using random sampling. These positions include a central home position, horizontally extended and contracted positions, and positions near the upper and lower vertical workspace limits. Experimental execution and data collection were automated using custom Python scripts integrated into the GUI, ensuring consistent testing conditions across all trials.

B. *Inverse Kinematics Performance*

Computational performance of the geometric IK algorithm is evaluated by measuring execution time during actual operation. The goal is to confirm that IK calculations are completed fast enough for real-time interactive control on low-cost embedded hardware. All timing measurements are taken on the host computer immediately before joint commands are transmitted to the microcontroller.

TABLE II
INVERSE KINEMATICS COMPUTATION PERFORMANCE

Test Position	Target Position (mm)	Computation Time (ms)
Home	(0, 0, 300)	1,2
Extended	(200, 0, 250)	1,5
Contracted	(-150, 100, 200)	1,8
Up	(100, 100, 400)	2,1
Down	(150, -50, 150)	1,9
Overall Mean		1,7

Table II summarizes the computation time for representative target positions within the workspace. The results show that the computation time remains within a narrow range between 1,2 ms and 2,1 ms, with an overall mean computation time of 1,7 ms. This indicates deterministic behavior of the geometric inverse kinematics algorithm, as the computation is based on closed-form analytical expressions rather than iterative numerical convergence.

Slight variations in computation time are observed across different target positions. The up position exhibits the longest computation time due to additional trigonometric evaluations and reachability validation checks required when operating near the vertical workspace limit. However, the observed variation remains small and does not affect the real-time responsiveness of the system.

Unlike iterative inverse kinematics methods that require repeated updates until convergence, the geometric approach employed in this study computes joint angles in a single pass. As a result, the computation time remains predictable and suitable for real-time educational robotic applications.

C. *Position Accuracy Analysis*

Position accuracy is evaluated to assess how closely the robotic arm reaches the desired target positions within the workspace. Five representative target positions are selected to cover central, extended, contracted, upper, and lower regions of the workspace. Each target position is tested five times under identical conditions to evaluate repeatability. Position error is computed as the straight-line spatial deviation between the specified target coordinates and the end-effector position obtained via forward kinematics verification.

TABLE III
POSITION ACCURACY TEST – HOME POSITION

Test No	Target Position (mm)	Actual Position (mm)	Total Error (mm)
1	(0, 0, 300)	(-0.8, 1.2, 298.5)	2,09
2	(0, 0, 300)	(0.5, -0.9, 299.2)	1,27
3	(0, 0, 300)	(-1.1, 0.7, 297.8)	2,54
4	(0, 0, 300)	(0.3, 1.5, 298.9)	1,89
5	(0, 0, 300)	(-0.6, -0.8, 299.4)	1,16
Mean			1,79

Table III presents the position accuracy results for the home position located near the center of the workspace. The mean positioning error of 1.79 mm indicates stable performance under nominal joint configurations. Although the target position is (0, 0, 300) mm, slight deviations are observed in the X and Y directions. These offsets are attributed to mechanical backlash, servo deadband, and structural compliance inherent to low-cost servo-driven robotic arms. The relatively low standard deviation of 0,58 mm demonstrates good repeatability across trials.

TABLE IV
POSITION ACCURACY TEST – EXTENDED POSITION

Test No	Target Position (mm)	Actual Position (mm)	Total Error (mm)
1	(200, 0, 250)	(198.2, 1.2, 248.7)	2,52
2	(200, 0, 250)	(199.1, -0.8, 249.5)	1,30
3	(200, 0, 250)	(197.5, 0.6, 248.2)	3,13
4	(200, 0, 250)	(198.7, -1.1, 249.1)	1,88
5	(200, 0, 250)	(199.3, 0.9, 248.8)	1,67
Mean			2,10

Table IV shows the accuracy results for the extended position near the maximum horizontal reach. The mean error increases to 2,10 mm compared to the home position, reflecting higher geometric sensitivity when the arm operates at extended configurations. In this region, small joint angle deviations result in amplified Cartesian errors. Nevertheless, the error values remain consistent across trials, indicating stable inverse kinematics execution near workspace boundaries.

TABLE V
POSITION ACCURACY TEST – CONTRACTED POSITION

Test No	Target Position (mm)	Actual Position (mm)	Total Error (mm)
1	(-150, 100, 200)	(-148.3, 98.5, 198.2)	2,89
2	(-150, 100, 200)	(-149.2, 99.3, 199.1)	1,39
3	(-150, 100, 200)	(-149.2, 99.3, 199.1)	3,81
4	(-150, 100, 200)	(-147.8, 98.1, 197.5)	1,86
5	(-150, 100, 200)	(-148.9, 99.2, 198.7)	1,48
Mean			2,29

The contracted position results in Table V correspond to

configurations closer to the base. A mean error of 2,29 mm is observed, slightly higher than the extended position. This behavior is associated with increased joint coupling effects and sensitivity of planar joint interactions in compact configurations. Despite these effects, the observed errors remain within an acceptable range for educational applications.

TABLE VI
POSITION ACCURACY TEST – UP POSITION

Test No	Target Position (mm)	Actual Position (mm)	Total Error (mm)
1	(100, 100, 400)	(98.1, 98.7, 397.2)	3,61
2	(100, 100, 400)	(99.2, 99.1, 398.5)	1,90
3	(100, 100, 400)	(97.5, 98.3, 396.8)	4,36
4	(100, 100, 400)	(98.8, 98.9, 397.9)	2,63
5	(100, 100, 400)	(99.3, 99.5, 398.1)	2,08
Mean			2,92

Table VI presents the accuracy results for the up position near the upper vertical limit of the workspace. The mean error increases to 2,92 mm, with a maximum inaccuracy of 4,36 mm. This rise is expected when the robotic arm operates near its upper vertical reach limit, where mechanical constraints and cumulative joint angle effects become more pronounced. The results indicate consistent behavior across repeated trials, with no indication of instability in the inverse kinematics.

TABLE VII
POSITION ACCURACY TEST – DOWN POSITION

Test No	Target Position (mm)	Actual Position (mm)	Total Error (mm)
1	(150, -50, 150)	(148.6, -49.2, 148.4)	2,09
2	(150, -50, 150)	(148.5, -49.1, 148.2)	1,90
3	(150, -50, 150)	(146.9, -47.8, 146.8)	4,36
4	(150, -50, 150)	(148.2, -48.7, 147.9)	2,63
5	(150, -50, 150)	(149.1, -49.5, 148.6)	2,08
Mean			2,92

The down position accuracy results are shown in Table VII. Among all evaluated configurations, this position produced the largest positioning error, registering an average offset of 2,92 mm and a worst-case error of 4,91 mm. This degraded accuracy is attributed to greater servo torque requirements, the influence of gravity, and heightened geometric sensitivity when operating near the lower workspace boundary. This configuration represents the worst-case positioning system performance.

TABLE VIII
POSITION ACCURACY TEST PERFORMANCE SUMMARY

Position	Mean Error(mm)	Max Error (mm)	Std. Dev (mm)
Home	1,79	2,54	0,58
Extended	2,10	3,13	0,69
Contracted	2,29	3,81	0,93
Up	2,92	4,36	0,96
Down	2,92	4,91	1,25
Overall Mean	2,46	4,91	0,88

Table VIII summarizes the positioning accuracy results across all tested positions. The overall positioning accuracy results are summarized in Table VIII, producing an overall average error of 2,46 mm with a positional standard deviation of 0,88 mm throughout the evaluated workspace points. The increase in error observed near the workspace boundaries is consistent with the inherent kinematic behavior of serial manipulators and does not indicate instability in the inverse kinematics formulation.

D. Discussion

The experimental results demonstrate that the proposed 5-DOF robotic arm control system achieves consistent positioning behavior across different regions of the workspace. The observed positioning accuracy varies depending on the robot configuration, with smaller errors near the center of the workspace and larger errors near workspace boundaries. This trend is consistent with the kinematic characteristics of serial manipulators, where geometric sensitivity increases as the arm operates closer to its reach limits.

The inverse kinematics performance results indicate that the geometric inverse kinematics approach provides deterministic and predictable computation time. The computation time remains within a narrow range for all tested positions, confirming that the closed-form analytical solution is suitable for real-time operation. Slight increases in computation time at vertical workspace extremes are attributed to additional validation and trigonometric evaluations, rather than instability or inefficiency in the algorithm itself.

Position accuracy analysis reveals that the maximum positioning error occurs at the lower workspace boundary, where the combined effects of gravitational loading, servo resolution, mechanical backlash, and geometric sensitivity are most pronounced. The presence of small lateral offsets at the home position further highlights the influence of mechanical compliance and servo deadband inherent to low-cost servo-driven robotic arms. These effects are expected in educational robotic platforms and do not indicate deficiencies in the inverse kinematics formulation.

Overall, the results confirm that the proposed system is well suited for educational robotics applications, where algorithm transparency, real-time responsiveness, and affordability are prioritized over high-precision industrial performance. While the system does not employ sensor-based closed-loop control, the achieved accuracy and stability are sufficient for kinematic demonstrations, laboratory experiments, and light manipulation tasks. Future improvements may include the integration of joint angle feedback sensors and enhanced calibration procedures to further improve positioning accuracy.

IV. CONCLUSION

This study presents the design and evaluation of a 5-DOF robotic arm control system based on geometric inverse kinematics and a Python-based graphical user interface. The system integrates computation, serial communication, and real-time visualization within a modular and user-friendly framework. Experimental findings show stable positioning performance across different workspace regions, achieving an average IK computation time of 1,7 ms alongside a mean end-effector positioning error of 2,46 mm. Although sensor-based closed-loop control is not implemented, the achieved performance is adequate for educational robotics and laboratory-scale manipulation using low-cost hardware. Future improvements might include the addition of joint feedback

sensors and enhanced calibration methods to increase positioning accuracy and system robustness.

REFERENCES

[1] H. Ye, D. Wang, J. Wu, Y. Yue, and Y. Zhou, "Forward and inverse kinematics of a 5-DOF hybrid robot for composite material machining," *Robot. Comput. Integr. Manuf.*, vol. 65, no. February, p. 101961, 2020, doi: 10.1016/j.rcim.2020.101961.

[2] K. Nasir, R. L. A. Shauri, N. M. Salleh, and N. H. Remeli, "Implementation of two-axis position-based impedance control with inverse kinematics solution for A 2-DOF robotic finger," *Int. J. Eng. Technol.*, vol. 7, no. 3, pp. 10–14, 2018, doi: 10.14419/ijet.v7i3.11.15920.

[3] A. Kurniawan Saputro, D. Rahmawati, and A. Fiqhi Ibadillah Teknik Elektro, "Implementasi Sistem Inverse Kinematics Pada Robot Arm Untuk Pemindahan Dan Penempatan Gelas Implementation of Inverse Kinematics System in Robotic Arm for Glass Pick and Place Operations," *Jambura J. Electr. Electron. Eng.*, vol. 63, pp. 63–69, 2025.

[4] V. Kumar, S. Sen, S. S. Roy, S. . Das, and S. N. Shome, "Inverse Kinematics of Redundant Manipulator using Interval Newton Method," *Int. J. Eng. Manuf.*, vol. 5, no. 2, pp. 19–29, 2015, doi: 10.5815/ijem.2015.02.03.

[5] T. F. Abaas, A. A. Khleif, and M. Q. Abbood, "Inverse Kinematics Analysis and Simulation of a 5 DOF Robotic Arm using MATLAB," *Al-Khwarizmi Eng. J.*, vol. 16, no. 1, pp. 1–10, 2020, doi: 10.22153/kej.2020.12.001.

[6] J. Wang, Z. Shao, H. Kang, H. Zhao, G. Song, and Y. Guan, "Inverse Kinematics of 6-DOF Robot Manipulator via Analytic Solution with Conformal Geometric Algebra," *2018 IEEE Int. Conf. Robot. Biomimetics, ROBIO 2018*, pp. 2508–2513, 2018, doi: 10.1109/ROBIO.2018.8665317.

[7] S. Yu and G. Tan, "Inverse Kinematics of a 7-Degree-of-Freedom Robotic Arm Based on Deep Reinforcement Learning and Damped Least Squares," *IEEE Access*, vol. 13, pp. 4857–4868, 2025, doi: 10.1109/ACCESS.2024.3521539.

[8] K. A. Wibisono *et al.*, "Implementation 4 – DoF Arm Robot Object Sorting Controlled Based On Color Using Inverse Kinemathics Algorithm," vol. 1, no. Icst, pp. 539–544, 2018, doi: 10.2991/icst-18.2018.112.

[9] M. G. S. Adiguna, M. Saleh, and E. D. Marindani, "Color and Size Sorting System with an ESP32-Based Robot Arm," *J. Electr. Eng. Energy Inf. Technol.*, vol. 11, no. 2, p. 73, 2023, doi: 10.26418/j3eit.v11i2.68483.

[10] R. Grassmann, V. Modes, and J. Burgner-Kahrs, "Learning the Forward and Inverse Kinematics of a 6-DOF Concentric Tube Continuum Robot in SE(3)," *IEEE Int. Conf. Intell. Robot. Syst.*, no. 3, pp. 5125–5132, 2018, doi: 10.1109/IROS.2018.8594451.

[11] A. A. Al-Hamadani and M. Z. Al-Faiz, "Design and Implementation of Inverse Kinematics Algorithm to Manipulate 5-DOF Humanoid Robotic Arm," *2021 Int. Conf. Innov. Intell. Informatics, Comput. Technol. (3ICT)*, pp. 693–697, 2021, doi: 10.1109/3ICT53449.2021.9581570.

[12] C. A. Pena, M. A. Guzman, and P. F. Cardenas, "Inverse kinematics of a 6 DOF industrial robot manipulator based on bio-inspired multi-objective optimization techniques," *2016 IEEE Colomb. Conf. Robot. Autom. CCRA 2016 - Conf. Proc.*, pp. 2–7, 2017, doi: 10.1109/CCRA.2016.7811428.

[13] X. Shi, Z. Guo, J. Huang, Y. Shen, and L. Xia, "A Distributed Reward Algorithm for Inverse Kinematics of Arm Robot," *Proc. - 5th Int. Conf. Autom. Control Robot. Eng. CACRE 2020*, pp. 92–96, 2020, doi: 10.1109/CACRE50138.2020.9230347.

[14] Z. Zou, J. Han, and M. Zhou, "Research on the inverse kinematics solution of robot arm for watermelon picking," *Proc. 2017 IEEE 2nd Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2017*, vol. 2018-January, pp. 1399–1402, 2017, doi: 10.1109/ITNEC.2017.8285026.

[15] A. Gregg-Smith and W. W. Mayol-Cuevas, "Inverse kinematics and design of a novel 6-DoF handheld robot arm," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2016-June, pp. 2102–2109, 2016, doi: 10.1109/ICRA.2016.7487359.

[16] I. D. Nugraha and P. M. Santika, "Pendekatan Geometri untuk Perhitungan Inverse Kinematics Gerakan Lengan Robot 4 Derajat Kebebasan," *J. Tek. Mesin ITI*, vol. 5, no. 1, p. 1, 2021, doi: 10.31543/jtm.v5i1.572.

[17] R. C. Siagian, "Inverse Kinematic Algorithm with Newton-Raphson Method iteration to Control Robot Position and Orientation based on R programming language," *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 17, no. 2, pp. 161–170, 2023, doi: 10.22146/ijccs.82781.

[18] S. Oh Park, J. Gon Yoon, M. Gyu Jung, and M. Cheol Lee, "Robot Manipulator Arm Inverse Kinematics Analysis by Jacobian *," *Proc. Int. Conf. Artif. Life Robot.*, vol. 23, pp. 282–285, 2018, doi: 10.5954/icarob.2018.os2-1.

[19] W. Xiang, J. Chen, H. Li, Z. Chai, and Y. Lou, "Research on End-Effector Position Error Compensation of Industrial Robotic Arm Based on ECOA-BP," *Sensors*, vol. 25, no. 2, p. 378, 2025, doi: 10.3390/s25020378.

[20] T. Morishita and O. Tojo, "Integer inverse kinematics for arm control of a compact autonomous robot," *Artif. Life Robot.*, vol. 22, no. 4, pp. 435–442, 2017, doi: 10.1007/s10015-017-0379-9.

[21] M. Ulum *et al.*, "Designing and Implementing Trajectory Planning and Inverse Kinematics Algorithms using Hexapod Robot Platform," vol. 1, no. Icst, pp. 838–842, 2018, doi: 10.2991/icst-18.2018.164.

[22] X. Zhang, M. Liu, L. Hu, S. Gou, and S. Wang, "Kinematics Analysis of a Five-Degree-of-Freedom Lightweight Explosive Ordnance Disposal Robotic Arm," *2022 China Autom. Congr. (CAC)*, pp. 2567–2571, 2022, doi: 10.1109/CAC57257.2022.10056037.

[23] S. S. Tørdal, G. Hovland, and I. Tyapin, "Efficient Implementation of Inverse Kinematics on a 6-DOF Industrial Robot using Conformal Geometric Algebra," *Adv. Appl. Clifford Algebr.*, vol. 27, no. 3, pp. 2067–2082, 2017, doi: 10.1007/s00006-016-0698-2.

[24] N. Koban, N. Çavli, H. Doğan, and E. Benli, "7 DOF Robotic Arm Inverse Kinematic Analysis with Cuckoo and Whale Algorithms," *2023 Innov. Intell. Syst. Appl. Conf. (ASYU)*, pp. 1–5, 2023, doi: 10.1109/ASYU58738.2023.10296820.

[25] F. Zhong, G. Liu, Z. Lu, Y. Han, and T. Ye, "Inverse Kinematics Analysis of Humanoid Robot Arm by Fusing Denavit–Hartenberg and Screw Theory to Imitate Human Motion With Kinect," *IEEE Access*, vol. 11, pp. 67126–67139, 2023, doi: 10.1109/ACCESS.2023.3291589.

[26] C. Urrea and D. Saa, "Design, Simulation, Implementation, and Comparison of Advanced Control Strategies Applied to a 6-DoF Planar Robot," *Symmetry (Basel).*, vol. 15, no. 5, 2023, doi: 10.3390/sym15051070.

[27] I. Sulaeman, A. W. Dani, and T. Pangaribowo, "Analisa Inverse Kinematics Pada Prototype 3-DoF Arm Robot Dengan Metode Anfis," *J. Teknol. Elektro*, vol. 13, no. 1, p. 14, 2022, doi: 10.22441/jte.2022.v13i1.003.

[28] A. Patil, M. Kulkarni, and A. Aswale, "Analysis of the inverse kinematics for 5 DOF robot arm using D-H parameters," *2017 IEEE Int. Conf. Real-Time Comput. Robot. RCAR 2017*, vol. 2017-July, pp. 688–693, 2017, doi: 10.1109/RCAR.2017.8311944.

[29] A. K. Mishra and O. Meruvia-Pastor, "Robot arm manipulation using depth-sensing cameras and inverse kinematics," *2014 Ocean. - St. John's, Ocean*, 2015, doi: 10.1109/OCEANS.2014.7003029.

[30] A. Sriram, A. Robilin, R. Krishnan, S. Jagadeesh, and K. Gnanasekaran, "IoT-Enabled 6DOF Robotic Arm with Inverse Kinematic Control: Design and Implementation," *2023 IEEE World Conf. Appl. Intell. Comput. (AIC)*, pp. 795–800, 2023, doi: 10.1109/AIC57670.2023.10263943.

[31] A. P. P. Prasetyo, I. Rahmatullah, K. Exaudi, and Rendyansyah, "Control System for U-Arm Robot Arm Movement with Linear Gripper Based on Inverse Kinematic Method," *JITCE (J. Inf. Technol. Comput. Eng.)*, vol. 8, no. 2, pp. 97–103, 2024, doi: 10.25077/jitce.8.2.97-103.2024.

[32] S. S. Tordal and G. Hovland, "Inverse kinematic control of an industrial robot used in Vessel-to-Vessel Motion Compensation," *2017 25th Mediterr. Conf. Control Autom. MED 2017*, pp. 1392–1397, 2017, doi: 10.1109/MED.2017.7984313.

[33] B. Gao, Z. Zhu, J. Zhao, and L. Jiang, "Inverse Kinematics and Workspace Analysis of a 3 DOF Flexible Parallel Humanoid Neck Robot," *J. Intell. Robot. Syst. Theory Appl.*, vol. 87, no. 2, pp. 211–229, 2017, doi: 10.1007/s10846-017-0502-0.

[34] S. Li, Z. Wang, Q. Zhang, and F. Han, "Solving Inerse Kinematics Model for 7-DoF Robot Arms Based on Space Vector," *2018 Int. Conf. Control Robot. ICCR 2018*, pp. 1–5, 2018, doi: 10.1109/ICCR.2018.8534498.

[35] F. Limanuel, C. Susanto, and F. R. G. Manalu, "Design and Implementation of 6 DOF ROTARIC Robot Using Inverse Kinematics Method," *J. Elektro*, vol. 13, no. 2, pp. 125–134, 2021, doi: 10.25170/jurnalelektro.v13i2.1930.