Integration of a 4-DOF Arduino Robot for Placing Coloured Cubes in a Matlab Simulation

Hayadi Hamuda[1], Anjar Setiawan[2]*

[1]*Department Systems Computer, Universitas Pamulang, Indonesia*
[2]*Department Information Systems, Universitas Pamulang, Indonesia*
Email Correspondence : dosen02886@unpam.ac.id

ARTICLE INFO

Article history:
Received 2 August 202 4
Received in revised form
September 10, 2024
Accepted 20 October 2024
Available online 31 November 2024

Keywords:
Inverse Kinematics, Matlab, Robotic Arm, Sensor TCS 230 and Coloured Cube

ABSTRACT

Robotic arms are increasingly used in industry, particularly in Indonesia, as part of the Industry 4.0 transition from manual to automated processes to improve efficiency and reduce operational costs. This study investigates a 4-DOF Arduino-based robotic arm for color-based cube sorting using MATLAB simulation. The results show that the calculated maximum torque is approximately 20% lower than the servo motor's rated torque of 10 kg·cm. Testing of the HC-SR04 and TCS230 sensors demonstrated reliable object detection at distances of 6.5–7.0 cm, enabling accurate color-based sorting. The MATLAB GUI successfully generated inverse kinematics solutions, with an end-effector position error of only 0.1% compared to the programmed coordinates (X = 318 mm, Y = 0 mm, Z = 65 mm). Arduino implementation showed a servo deviation of approximately 10°, while maintaining an overall positioning error of around 0.1%. These results indicate that the proposed robotic arm performs effectively for automated color-based object sorting applications.

1 Introduction

Advances in robotics and automation technology have experienced significant growth over the past decade, driven by the increasing demand for efficiency, precision, productivity, and operational flexibility across industrial, educational, and research sectors. The emergence of the Fourth Industrial Revolution (Industry 4.0) has accelerated the adoption of intelligent automation systems by integrating robotics, artificial intelligence, the Internet of Things (IoT), and digital manufacturing technologies. This transformation has reshaped industrial processes from conventional manual operations to automated and data-driven systems capable of improving production quality, reducing human intervention, minimizing operational costs, and enhancing overall manufacturing performance [1]. As a result, robotic systems have become an essential component in modern industries, particularly in applications requiring repetitive, accurate, and continuous operations [2].

Among various robotic systems, robotic arms have attracted considerable attention due to their versatility and ability to mimic the motion and functionality of the human arm. A robotic arm is an electromechanical manipulator consisting of interconnected links and joints that can be programmed to perform specific tasks such as object handling, pick-and-place operations, assembly, inspection, sorting, and material transportation. The effectiveness of a robotic arm largely depends on the accuracy of its mechanical design and control system, which are closely related to its kinematic and dynamic characteristics. Therefore, one of the fundamental challenges in robotic arm development is the formulation of accurate mathematical models that describe the relationship between joint movements and end-effector positions. These models serve as the foundation for motion planning, trajectory generation, and control system implementation [3].

Kinematic analysis plays a crucial role in robotic arm operation because it determines the position and orientation of the end-effector based on the movement of individual joints. Forward kinematics is used to calculate the end-effector position from known joint angles, whereas inverse kinematics determines the required joint angles to achieve a desired end-effector position. Accurate kinematic modeling enables robotic arms to perform tasks with high precision and repeatability, which are essential requirements in industrial automation applications [3].

Four-degree-of-freedom (4-DOF) robotic arms represent a widely adopted manipulator configuration due to their balance between mechanical simplicity and operational flexibility. A 4-DOF robotic arm provides sufficient freedom of movement to perform fundamental tasks such as object grasping, sorting, stacking, and positioning while maintaining relatively low computational and mechanical complexity. Consequently, this configuration is frequently employed in educational laboratories, academic research, and prototype industrial automation systems, where cost-effectiveness and ease of implementation are important considerations [4]. Furthermore, the integration of sensor technologies, such as ultrasonic sensors for object detection and color sensors for object classification, enhances the capability of robotic arms to perform autonomous object-sorting tasks based on environmental information [5].

Considering the growing demand for intelligent automation systems, the development of a 4-DOF robotic arm integrated with object detection and color recognition capabilities offers significant potential for improving automated sorting applications. Such systems provide a practical platform for studying robotic kinematics, sensor integration, and control strategies while contributing to the advancement of low-cost automation solutions suitable for educational and industrial environments.

Arduino is an open-source microcontroller platform that is popular amongst academics and robotics enthusiasts due to its ease of use, relatively low cost, and a very large user community. Arduino enables users to control actuators, sensors, and other electronic devices through simple yet powerful programming[6]. On the other hand, matlab is widely used software for modelling, simulation, and analysis of dynamic systems, including robotic systems. matlab provides various toolboxes, one of which is the Robotics Toolbox, which facilitates the simulation of robot movements, trajectory planning, and image processing[7].

The integration of physical robots (Arduino 4-DOF) with matlab simulation is crucial for bridging the real and virtual worlds. Through this integration, workflows involving planning, simulation and physical implementation can be carried out more efficiently[8]. The simulation process in matlab allows users to visualise and test algorithms before applying them to physical robots, thereby minimising errors and damage to hardware[9]. The placement of coloured cubes as objects in robotics experiments is a frequently used case study to test a robot's ability to detect, identify, and move objects based on specific characteristics (colour-based object sorting). With the aid of image processing in matlab, the robot can be directed to recognise the colour of the cubes, determine their position, and plan a trajectory to pick up and place the cubes in the desired location[10].

Therefore, research and development of an Arduino 4-DOF robot integration system for placing coloured cubes in a Matlab simulation is highly relevant both as a learning tool, for algorithm testing, and for the development of more complex robotics applications in the future[11]. Furthermore, such integration can also serve as a foundation for the development of automation systems based on artificial intelligence, the Internet of Things (IoT), and intelligent control systems, whilst facilitating the development of systems capable of recognising the colour of a cube and placing it accurately and autonomously[12]. This research aims to advance Arduino-based robotics systems, particularly in terms of the recognition and manipulation of

coloured objects, whilst enhancing their use as a tool for studying and testing robotics algorithms[13].

2 Literature Study

Several studies have investigated the application of robotic arms from different perspectives. A study entitled "Modelling the Kinematics of a Manipulator Robot Using MATLAB" focused on the analysis of forward and inverse kinematics through a MATLAB-based graphical simulation environment. Although the study successfully demonstrated kinematic modeling and visualization, the research was limited to simulation, and no physical prototype was developed or experimentally validated [14].

Another study, "Design of a 5-Degree-of-Freedom Robot Arm Using a Kinematic Approach," examined the forward kinematic performance of a 5-DOF robotic arm. The research provided valuable insights into the mathematical modeling of robot motion; however, inverse kinematic implementation and experimental validation on a physical robotic arm platform were not conducted, limiting the evaluation of real-world system performance [15].

Research entitled "Development and Implementation of a 4-DOF Robot Arm Control System Equipped with an Arduino Uno-Based Graphical User Interface" presented the design and implementation of a 4-DOF robotic arm integrated with a GUI-based control system. While the study demonstrated effective user interaction and robot control, it did not address inverse kinematic modeling, which is essential for achieving accurate and autonomous end-effector positioning [16].

A comprehensive review of pick-and-place robotic arm systems summarized 25 robotic arm prototypes developed for object handling applications. The review highlighted various control architectures and hardware configurations, including systems utilizing Arduino microcontrollers. However, the study primarily focused on summarizing existing developments and did not propose or experimentally evaluate an integrated robotic arm system combining kinematic analysis, sensor-based object recognition, and autonomous object sorting [17].

Furthermore, a literature review on Arduino-based robotic arms identified significant opportunities for expanding the application of low-cost robotic platforms in industrial automation. The review emphasized the potential of Arduino-based systems for tasks such as pick-and-place operations, welding, painting, assembly, and manufacturing processes. Nevertheless, the study remained conceptual in nature and did not present a complete implementation integrating robotic kinematics, sensor technologies, and intelligent object manipulation capabilities [18].

Based on the literature review, several research gaps can be identified. Previous studies have generally focused on either simulation-based kinematic analysis, robotic arm design, graphical user interfaces, or literature surveys, with limited integration of these components into a single system. Moreover, few studies have combined forward and inverse kinematic modeling with real-time object detection and color-based sorting using a physical 4-DOF robotic arm platform. Therefore, this study proposes the development and implementation of a 4-DOF Arduino-based robotic arm equipped with ultrasonic and color sensors, integrated with MATLAB for kinematic analysis and visualization. The proposed system is expected to provide a practical and cost-effective solution for automated pick-and-place and color-sorting applications while contributing to the advancement of educational and industrial robotic systems.

3 Methodology

This study employs a hardware–software integration approach to develop and evaluate a 4-DOF robotic arm system for color-based object sorting. The hardware design phase involves the fabrication and assembly of mechanical components, including brackets for mounting servo motors that function as the robot's actuators. The system is equipped with a TCS230 color sensor for object color identification and an HC-SR04 ultrasonic sensor for object detection and distance measurement. Subsequently, all electronic components are integrated with an Arduino microcontroller and connected to an appropriate power supply to ensure stable system operation.

The software design phase consists of developing the control algorithm using the Arduino Integrated Development Environment (IDE) and implementing kinematic analysis using MATLAB. The Arduino program is responsible for sensor data acquisition, color classification, servo motor control, and execution of the pick-and-place sequence. MATLAB is utilized to model and analyze the inverse kinematics of the robotic arm, enabling the calculation of the required joint angles to achieve a desired end-effector position.

The operational workflow of the proposed system begins with object detection using the HC-SR04 ultrasonic sensor. Once an object is detected within the predefined range, the TCS230 sensor identifies its color. Based on the sensor readings, the robotic arm moves from its initial position toward the detected object, grasps the object using the end-effector, and subsequently transports it to a designated location according to its color classification. Furthermore, MATLAB simulation is employed to validate the inverse kinematic calculations and evaluate the positional accuracy of the robotic arm by comparing the calculated coordinates with the actual movement results. This analysis is conducted to determine system performance and identify potential errors in the kinematic model and control implementation.

3.1 Research Process

Figure 1 shows a colour detection system using a TCS230 sensor; the distance to an object to be moved to specified coordinates is calculated using inverse kinematics applied to a robotic arm with 4 degrees of freedom, a gripper, utilising an HC-SR04 sensor for measurement, which is determined by comparing the sensor's response range based on programming results with physical measurements taken using a ruler and an arduino Uno R3. The process begins with component initialisation, specifically activating the sensor on a 12 V power supply, which is stepped down to 6 V to drive the five servos; the 12 V power supply is also converted to 5 V to power the arduino Uno; the sensor requires a voltage of 3.3 V, which is supplied via the arduino pin. Sebuah layar LCD 20×4 dengan antarmuka I2C telah dipasang untuk menampilkan hasil warna dan jarak. Once all components are ready, the system begins collecting colour data in real time from the TCS230 sensor. The system then calculates the colour values for each RGB component (Red, Green, Blue) obtained from the sensor. This data is subsequently processed during the colour data processing phase, which involves normalising and calibrating the RGB values to align with the set thresholds. Once the calibration process is complete, the system identifies the dominant colour by analysing the highest observed RGB value. If the processing results indicate successful colour recognition, that colour is classified as the detected colour. If the sensor is not detected accurately, the system will restart the sensor reading process to obtain more accurate data.

Once a colour has been successfully identified, the system displays the name of that colour on a 20×4 LCD screen (for example, 'RED', 'GREEN' or 'BLUE'); in addition to the RGB colours, there are also yellow, black, pink and white. At the same time, the system sends this colour data to matlab for simulation purposes, which is carried out by altering each θ value (angle), and the inverse kinematics values for each axis specifically the positions of the X-axis, Y-axis, and Z-axis will be displayed. On the initial screen, all five values are initialised to 0; thereafter, the inverse kinematics values for the positions of these three axes will be displayed. The Arduino Uno R3 sends commands to five servos, each with different torque values influenced by various factors, including arm length; furthermore, the torque value of each servo is also influenced by whether the arm is moving without carrying an object or whilst carrying an object, thus requiring torque calculations for each servo used. This 20×4 LCD display with an I2C interface is used for text monitoring to quickly and accurately identify colours and distances. In this 4-DOF robotic arm method, after detecting an object using a distance sensor and recognising its colour with a colour sensor, the robotic arm moves from its initial position to the object's location, then transfers it to a predetermined destination. Matlab is used to evaluate any errors or shortcomings in the calculations.

Corresponding Author: dosen02886@unpam.ac.id
Doi:

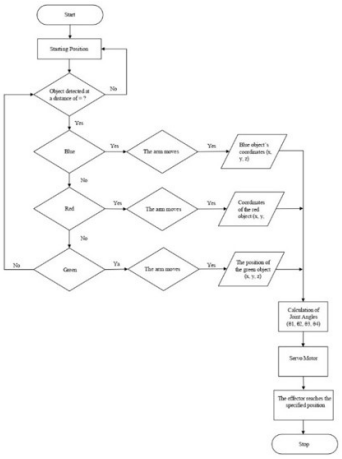

Figure 1. Research Process Flowchart

3.2 *System Architecture*

At this stage, the design concept and system architecture have been established to address the research challenges, utilising a circuit diagram for the colour detection device. The system is designed to identify colours via a TCS230 colour sensor connected to an Arduino Uno R3 microcontroller. The colour data is processed by the Arduino Uno R3 for categorisation, then displayed on a 20×4 LCD screen, identifying the colour and distance; the robot arm moves from its initial position to the object's location, then moves to the predetermined destination and sends a notification to matlab by evaluating errors or shortcomings in the calculations[19].

In Figure 2, each pin on the component is connected to the others using jumper wires. This design integrates hardware and software to ensure the device operates autonomously and in real-time (as shown in Table 1). The system is designed to be portable, simple and user-friendly for individuals using a 4-DOF robot to arrange coloured cubes, and is equipped with three output modes: text, matlab and digital messages[20].

Table 1. Kinematic configuration specifications for the 4-DOF IoT robotic arm

Software	Hardware
Arduino IDE (C language) Matlab (C, C++, Java)	Power supply 5 servos HC-SR04 ultrasonic sensor TCS230 colour sensor LM2595 buck converter Sismin PCB Arduino Uno R3 20x40 LCD Jumper cables Wooden board

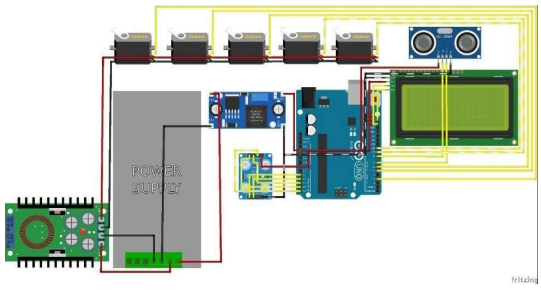

Figure 2: Wiring diagram 4-DOF robot matlab analysis

3.3 *Programme Planning*

See Figure 3 This programme facilitates the processing of input data from sensors, which then drives the servo motor via inverse kinematics calculations embedded within the Arduino programme. The servo motor movements generated by the programme are then analysed to ensure they correspond to the actual movements, thereby enabling the identification of errors that can form the basis for subsequent programme corrections[21].

Figure 3. Arduino 4-DOF Robot Programme

After each table or figure is inserted, provide a follow-up paragraph to explain it further. Don't forget to mention which figure or table it refers to.

Figure titles do not end with a period. For figures larger than one column wide, they should be placed at the beginning or end of the page, while for figures smaller than one column wide, placement is arbitrary. However, these small figures should not be placed before the sentence that references them, let alone before the title of the paper.

3.4 *Graphical User Interface Design in Matlab*

As shown in Figures 4 and 5, this analysis was carried out by developing a simulatable Graphical User Interface (GUI), thereby enabling a comparison of the results with the output from the Arduino application, with a focus on inverse kinematics. This simulation utilises four servos by adjusting θ1, θ2, θ3 and θ4; it also includes buttons for X Position, Y Position and Z Position, as well as buttons for the Forward and Reverse functions. This simulation is fairly straightforward, given that the focus of this project is on the Arduino IDE.

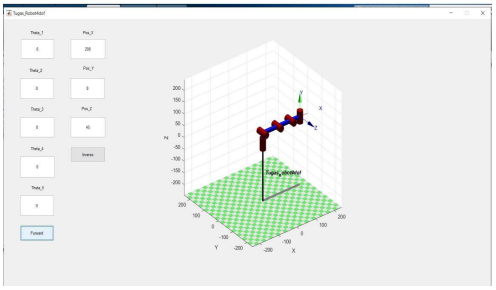

Figure 4. Visualisation in a Matlab Simulation

Below is a code library snippet developed using matlab software, presented in the form of an m-file.

```
MATLAB Programme Code for a 4DOF Robot
function varargout = Tugas_Robot4dof (varargin)
%---Executes on button press in btn Forward.
function btn_Forward Callback (hObject, eventdata, handles)
%hObject handle to btn_Forward (see GCBO)
% eventdata reserved - to be defined in a future version of
MATLAB
% handles structure with handles and user data (see GUIDATA)
Th_1 = str2double (handles.Theta_1.String) *pi/180;
Th_2 = str2double (handles.Theta_2.String) *pi/180;
Th_3 = str2double (handles.Theta_3.String) *pi/180;
Irh_4 = str2double (handles.Theta_4.String) *pi/180;
```

The matlab programme runs in real time, producing efficient simulation results to effectively control a 4-DOF robotic arm with a gripper, utilising input signals from the TCS230 colour sensor and the HCSR-04 distance sensor, as well as simulation results from the graphical user interface (GUI) developed in matlab.

4 Discussion

As illustrated in Figure 5, the developed prototype was evaluated by comparing the output generated by the Arduino Uno-based control system with the simulation results obtained

from the MATLAB Graphical User Interface (GUI). The proposed 4-DOF robotic arm, equipped with a gripper, utilizes input data from the TCS230 color sensor and the HC-SR04 ultrasonic sensor to perform object detection and color-based sorting tasks. To ensure reliable system performance, both sensors were tested and validated to identify potential measurement errors and inconsistencies.

The evaluation process was conducted in two stages: hardware assessment and software assessment. The hardware assessment focused on the performance of the robotic arm components, including sensor accuracy, servo motor operation, and object manipulation capabilities. Meanwhile, the software assessment evaluated the functionality of the Arduino control program and the MATLAB GUI, particularly the accuracy of the inverse kinematics calculations and the consistency between the simulated and actual robotic arm movements. The comparison of these results was used to determine the overall accuracy and effectiveness of the proposed robotic arm system.

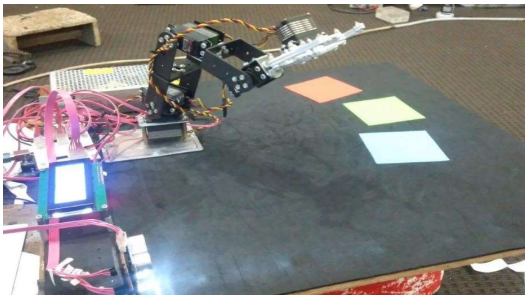

Figure 5. Standby Position

When the system is in standby mode, the HC-SR04 ultrasonic sensor continuously monitors the surrounding area for the presence of an object. If an object is detected within a distance of 7 cm and the TCS230 color sensor identifies the object as red, the robotic arm automatically exits standby mode and initiates the pick-and-place operation. The robotic arm then moves toward the detected object, grasps it using the gripper mechanism, and transports it to the designated location according to the predefined sorting criteria. This process demonstrates the integration of distance sensing, color recognition, and robotic arm control for autonomous object-handling applications.

4.1. Evaluation 4-DOF TCS 230 dan HC-SR04 Sensor Robot

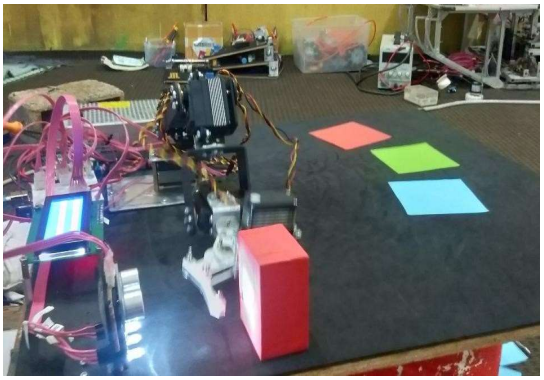

Figure 6. Two Views of the Red Object

See Figure 6 an object was placed at a specific location, with an HC-SR04 sensor attached. Initially, the object was positioned 7 cm from the sensor, then gradually moved to distances of 4 cm, 5 cm, 6 cm, 6.5 cm, 7.5 cm, 8 cm, 9 cm, 10 cm, 11 cm, and 12 cm. This test evaluates whether the robot arm will continue to move as the object is moved further away from the HC-SR04 distance sensor, which is positioned 7 cm from the sensor, and monitors for any errors during the HC-SR04 sensor's assessment. See Figure 7 testing of the TCS 230 colour sensor was carried out by placing a coloured object 7 cm away, so that the sensor could identify the object's colour based on its classification. The object is then placed in a designated position corresponding to its colour: the red position, the green position, or the blue position. Data collection for the TCS 230 colour sensor test utilised seven colours: the RGB colours (red, green, and blue), as well as yellow, black, pink, and white.

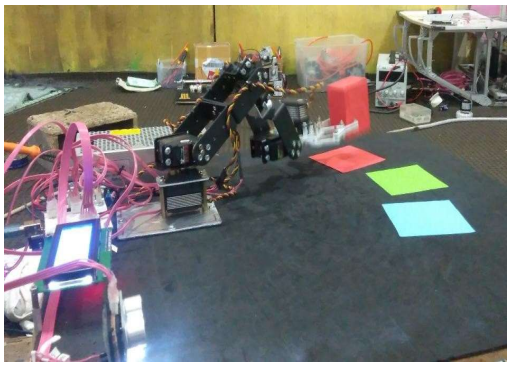

Figure 7. Three Positions for Placing the Red Object as Shown

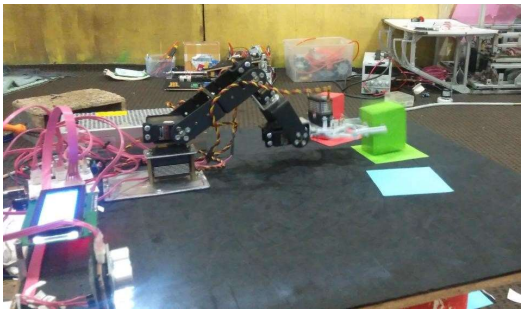

Figure 8. Locating Green Objects within a Specified Spatial Area

Figure 9. Positioning the Blue Objects so that They Align

Once the object has been successfully grasped by the gripper, the robotic arm transports and places it into a predefined location based on its detected color classification. Objects identified as red, green, and blue are sorted into their respective designated positions. The sorting process continues for each detected object until the final color test, namely white, is completed. Conversely, if the detected color does not belong to the predefined RGB categories (red, green, or blue), the robotic arm does not execute any pick-and-place action and remains in standby mode. This behavior ensures that the system only responds to authorized color classifications and prevents unintended object manipulation.

4.2. Matlab Software Testing

A procedure to verify that a robotic system can identify and respond to red, green and blue objects using the TCS230 and HCSR-04 colour sensors and MATLAB code. This test aims to evaluate the accuracy of colour identification and the robot's ability to move in accordance with instructions generated from the colour detection results.

Corresponding Author: dosen02886@unpam.ac.id
Doi:

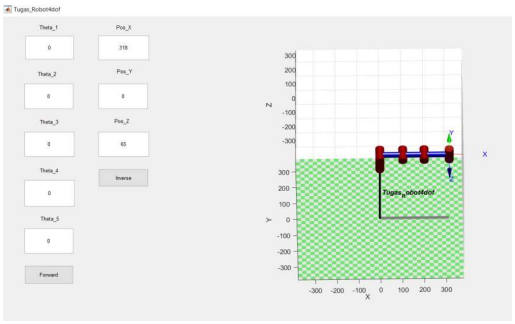

Figure 10. Initial Display with All five θ Values Set to Zero

As shown in Table 2 Initially, all angles θ are set to 0; when rotated, the position of the X-axis changes to 318, the position of the Y-axis remains at 0, and the position of the Z-axis changes to 65. These values are obtained from the formula defined in the M-file in matlab.

Table 2. Initial positions for the simulation

Order	Fonem	Number	Reverse	Number
1	Θ_1	0	Position X	318
2	Θ_2	0	Position Y	0
3	θ_3	0	Position Z	65
4	θ_4	0		
5	θ_5	0		
Forward				

In Table 3, when the arm is positioned to grasp an object, it rotates 180° at angle θ1, 60° at angle θ2, -120° at angle θ3, 60° at angle θ4, and 160° at the gripping section. The calculated inverse values are 216° on the X-axis, 0° on the Y-axis, and 71° on the Z-axis.

Table 3. Duration of Object Collection

Order	Fonem	Number	Reverse	Number
1	θ_1	180	Position X	216
2	θ_2	60	Position Y	0
3	θ_3	-120	Position Z	71
4	θ_4	60		
5	θ_5	160		
Forward				

In Table 4, The arm rotates 120° to position an object at angle θ1, 40° at angle θ2, -80° at angle θ3, -40° at angle θ4, and 90° at the clamp. The calculated inverse values are -136° on the X-axis, 234° on the Y-axis, and 69° on the Z-axis.

Table 4. Position of the Red Object

Order	Fonem	Number	Reverse	Number
1	θ_1	90	Position X	0
2	θ_2	40	Position Y	270
3	θ_3	-80	Position Z	69
4	θ_4	40		
5	θ_5	90		
Forward				

Table 5. The arm moves 120° to position an object at angle θ1, 40° at angle θ2, -80° at angle θ3, -40° at angle θ4, and 90° at the gripper. The calculated inverse values are -136° on the X-axis, 234° on the Y-axis, and 69° on the Z-axis.

Table 5. Placement of green objects

Order	Fonem	Number	Reverse	Number
1	θ_1	90	Position X	0
2	θ_2	40	Position Y	270
3	θ_3	-80	Position Z	69
4	θ_4	40		
5	θ_5	90		
Forward				

Table 6. The arm moves 60° to position an object at angle θ1, 40° at angle θ2, -80° at angle θ3, -40° at angle θ4, and 90° at the gripper. The resulting inverse values are 0° on the X-axis, 270° on the Y-axis, and 69° on the Z-axis.

Table 6. Position of the blue object

Order	Fonem	Number	Reverse	Number
1	θ_1	90	Position X	0
2	θ_2	40	Position Y	270
3	θ_3	-80	Position Z	69
4	θ_4	40		
5	θ_5	90		
Forward				

As shown in Table 6 above, when the arm is oriented to place an object at angle θ1, the arm rotates by 90°, at angle θ2 by 40°, at angle θ3 by -80°, at angle θ4 by -40°, and at the gripper by 90°. The resulting inverse values are 135° on the X-axis, 234° on the Y-axis, and 69° on the Z-axis.

4.3. Torque Calculation

This study examines the use of a 4-DOF Arduino robot to place coloured cubes in a matlab simulation, utilising five servos, each with different torque values influenced by various factors, including arm length. Furthermore, the movement of the arm both when not holding an object and when holding an object also affects the torque values of each servo; this is relevant to the calculation of torque values for each servo used.

The result of the calculation is servo base torque value As the servo base only moves horizontally, the speed must first be determined in order to calculate the servo torque. Referring to the formula for calculating the speed.

$T = 0,5$ s

$\omega = \frac{2\pi}{T}$

$\omega = \frac{2.\ 3,14}{0,5}$

$\omega = 12,56$ rad/s

Once the speed value has been determined, the torque value at the servo base can be calculated using the formula.

$\Sigma\tau = I\ \alpha$

$\Sigma\tau = m.r.\ \frac{\Delta\omega}{\Delta\tau}$

$\Sigma\tau = 0,55 \times (0,34)^2 \times \frac{12.56\text{-}0}{1\text{-}0}$

$\Sigma\tau = (0,064)\ \chi\ \frac{12,56}{1}$

$\Sigma\tau = 0,80$ N.m (1 N.m = 10,19 Kg.em)

$\Sigma\tau = 8$ kg.cm

Torque specifications for the lower arm servo Servo 2, located on the lower arm, moves vertically upwards whilst

supporting the weight of the upper arm servo and the gripping arm servo. Before determining the torque values, it is important to know the mass of each arm servo.

$m1 = servo\ lengan\ atas = 55\ gr$
$m2 = m1 +\ servo\ lengang\ gripper + servo\ gripper = 165\ gr$
$m3 = m2 +$ mass Obek $= 230\ gr$
So, the torque value for the lower arm servo is:
$\tau(servo\ 2) = (F1.r1.sin90) + (F2.r2.sin90) + (F3.r3.sin90)$
$\tau(servo\ 2) = (m1.g.r1.sin90) + (m2.g.r2.sin90) + (m3.g.r3.sin90)$
$\tau(servo\ 2) = (0{,}05x9{,}8x0{,}11x1) + (0{,}16x9{,}8x0{,}21x1) + (0{,}23x9{,}8x0{,}27x1)$
$\tau(servo\ 2) = (0{,}053) + (0{,}032) + (0{,}608)$
$\tau(servo\ 2) = 0{,}693\ N.m$
$\tau(servo\ 2) = 6{,}93\ Kg.cm$

The torque value of the upper arm servo is the same as that of the lower arm servo; the upper arm servo also moves vertically upwards and supports both the gripping arm servo and the gripping servo.

$m1 = servo\ lengan\ gripper = 55\ gr$
$m2 = m1 + servo\ gripper +$ obek benda $= 230\ gr$
So, the torque value for the upper arm servo is:
$\tau(servo\ 3) = (F1.r1.sin90) + (F2.r2.sin90)$
$\tau(servo\ 3) = (0{,}05x9{,}8x0{,}10x1) + (0{,}23x9{,}8x0{,}17x1)$
$\tau(servo\ 3) = (0{,}049) + (0{,}383)$
$\tau(servo\ 3) = 0{,}432\ N.m$
$\tau(servo\ 3) = 4{,}33\ Kg.cm$

The torque rating of the gripper arm servo is the same as that of the upper arm servo; the gripper arm servo also moves vertically upwards and supports both the gripper and the object being held

$m1 = servo\ \ gripper +$ obek benda $= 120\ gr$
So, the torque value of the gripper arm is:
$\tau(servo\ 4) = (F1.r1.sin90)$
$\tau(servo\ 4) = (0{,}05x9{,}8x0{,}6x1)$
$\tau(servo\ 4) = (0{,}294)\ N.m$
$\tau(servo\ 4) = 2{,}94\ Kg.cm$

The gripper torque is equal to the servo base torque, as the gripper rotates horizontally and supports the object; therefore, determining the gripper torque is equivalent to determining the servo base torque.
$\Sigma\tau = I\ \alpha$
$\Sigma\tau = m.r.\ \frac{\Delta\omega}{\Delta\tau}$
$\Sigma\tau = 0{,}06 \times (0{,}08)^2 \times \frac{12.56-0}{1-0}$
$\Sigma\tau = (0{,}000384)\ \chi\ \frac{12{,}56}{1}$
$\Sigma\tau = 0{,}0048$ N.m
$\Sigma\tau = 0{,}049$ kg.cm

The torque for each servo motor was analysed by first calculating its speed, which was determined to be 12.56 rad/s for both the base servo motor and the gripper servo motor. The base servo motor exhibited the highest torque, namely 0.80 N·m or 8 kg·cm, whilst the load carried was 65 grams. The minimum torque on the gripper servo motor is 0.0048 N·m or 0.049 kg·cm, corresponding to a load of 65 grams. Servo 2, located on the lower arm, moves vertically upwards, supporting the load from the upper arm servo and the gripper arm servo, thus requiring a torque of 6.93 kg·cm. Similar to the lower arm servo, the upper arm servo also moves vertically upwards, providing support to the gripper arm servo and the gripper servo, thus requiring a torque of 4.33 kg·cm. Meanwhile, the gripper arm servo requires a torque of 2.94 kg·cm. The specifications for the MG996R servo motor indicate a maximum torque of 10 kg·cm; exceeding this limit will result in servo failure.

5 Conclusion

The experimental results demonstrate the successful integration of a 4-DOF Arduino-based robotic arm with MATLAB simulation for color-based cube sorting. The calculated maximum torque was approximately 20% lower than the servo motor's rated maximum torque of 10 kg·cm, indicating that the actuator operated within a safe working range. Evaluation of the HC-SR04 ultrasonic sensor and TCS230 color sensor showed reliable object detection within a distance range of 6.5–7.0 cm, while the highest color recognition accuracy was achieved under an illumination level of 35.7 lux. The MATLAB simulation successfully generated inverse kinematics solutions by calculating the joint angles required to position the end-effector at the desired coordinates. Comparison between the simulation results and the physical implementation revealed a positional deviation of approximately 10° in servo movement; however, the overall positioning error remained low at approximately 0.1%. These findings indicate that the proposed system is capable of performing accurate color-based object sorting and demonstrates the effectiveness of integrating kinematic modeling, sensor-based object recognition, and robotic arm control.

1. Author Contribution

Nila Pratiwi contributed to the conceptualization of the study, methodology development, formal analysis, supervision, and manuscript preparation. Hayadi Hamuda was responsible for software development, system implementation, experimental testing, and data curation. Validation of the experimental results was conducted collaboratively by Nila Pratiwi and Hayadi Hamuda to ensure the accuracy and reliability of the collected data. The original draft of the manuscript was prepared by Nila Pratiwi, while manuscript review and editing were carried out jointly by all authors. In addition, Nila Pratiwi and Andre Nurdiansah contributed to the design of the system visualization and the integration of the 4-DOF Arduino-based robotic arm with the MATLAB simulation for color-based cube placement. All authors have read and approved the final version of the manuscript.

2. Thank-you note

The author would like to thank the Computer Systems Programme at Pamulang University for its support, which has facilitated the successful completion of this research.

3. Reference

[1] M. Cheng, H. Liao, C. Lan, and S. Member, "A 4-DoF Pick-and-Place Robot With High Structural Rigidity and Pose Repeatability," no. May, 2025.
[2] A. G. M. A. Aziz, S. Al Dawsari, A. E. Rafaat, A. G. A. El-magd, and A. A. Z. Diab, "A Smart Four-DOF SCARA Robot : Design , Kinematic Modeling , and Machine Learning-Based Performance Evaluation," pp. 1–30, 2026.
[3] A. J. Moshayedi, A. S. Roy, L. Liao, A. S. Khan, A. Kolahdooz, and A. Eftekhari, "Design and Development of FOODIEBOT Robot : From Simulation to Design," *IEEE Access*, vol. 12, no. March, pp. 36148–36172, 2024, doi: 10.1109/ACCESS.2024.3355278.
[4] M. Makulavi˙, "Industrial Robotic Setups : Tools and Technologies for Tracking and Analysis in Industrial Processes," 2025.
[5] K. M. Salem, M. S. Mohamed, M. H. Elmessmary, A. Ehsan, A. O. Elgharib, and H. Elshimy, "Design and Development of Cost-Effective Humanoid Robots for Enhanced Human – Robot Interaction," pp. 1–31, 2025.
[6] M. Leveziel *et al.*, "A 4-DoF Parallel Robot With a Built-in Gripper for Waste Sorting To cite this version : A 4-DoF parallel robot with a built-in gripper for waste sorting," 2022.
[7] T. N. Nguyen, Q. T. Truong, N. Duy, P. Tran, J. Yoon, and

Corresponding Author: dosen02886@unpam.ac.id
Doi:

D. T. Tran, "Practical Control of an Upper-Limb Rehabilitation Robotic Arm Using a PD Controller With Time-Delay Estimation," *IEEE Access*, vol. 14, no. January, pp. 2451–2462, 2026, doi: 10.1109/ACCESS.2025.3649756.

[8] D. T. Tran, X. T. Nguyen, T. N. Nguyen, and Q. T. Truong, "Practical Synchronous Sliding Mode Control With Time Delay Estimation for a 4-DOF Parallel Manipulator With Unknown Dynamics and Variable Payload," *IEEE Access*, vol. 13, no. April, pp. 102758–102770, 2025, doi: 10.1109/ACCESS.2025.3577689.

[9] C. J. Thomson *et al.*, "Simultaneous and Proportional Myoelectric Control of Multiple Degrees of Freedom in Individuals With Chronic Hemiparesis," vol. 33, pp. 3246–3258, 2025.

[10] Z. Ben, H. Nivine, G. Ali, and H. Altaif, *A study of advanced mathematical modeling and adaptive control strategies for trajectory tracking in the Mitsubishi RV - 2AJ 5 - DOF Robotic Arm*. Springer Nature Singapore, 2025. doi: 10.1007/s44430-025-00001-5.

[11] D. Lee and G. Kwon, "Terrain-Adaptive Mobile-Base Platform for Enhanced Robot Adaptability and Stability," *IEEE Access*, vol. 13, no. December, pp. 206572–206605, 2025, doi: 10.1109/ACCESS.2025.3637564.

[12] H. Hamuda and A. Setiawan, "Integrasi Sistem Keamanan Rumah Pintar Berbasis ESP32-CAM dan Sensor PIR dengan Notifikasi Real-Time melalui WhatsApp Bot," *SKANIKA Sist. Komput. dan Tek. Inform.*, vol. 8, no. 2, pp. 204–218, 2025.

[13] M. B. Kadri, "Trajectory Tracking Control of a Planar Robotic Arm Using Inverse Dynamics and Fuzzy Gain Scheduling : Simulation and Experimental Validation," *IEEE Access*, vol. 13, no. October, pp. 186736–186759, 2025, doi: 10.1109/ACCESS.2025.3626418.

[14] H. Mazdarani, G. S. Member, and B. E. N. Sainsbury, "Ultrasound-Based Visual Servoing for Out-of-Plane Longitudinal Needle Tracking in Robot-Aided Percutaneous Nephrolithotomy," *IEEE Access*, vol. 13, no. April, pp. 69090–69102, 2025, doi: 10.1109/ACCESS.2025.3562078.

[15] A. Zegeye, S. K. I. Jeong, H. A. E. Y. Park, and S. Jung, "Vision-Based Control of Robotic Arm Mounted on Unmanned Surface Vehicle for Rocket Recovery," *IEEE Access*, vol. 13, no. May, pp. 114038–114055, 2025, doi: 10.1109/ACCESS.2025.3583854.

[16] O. D. Laksono and C. I. Aditya, "MITOR : Jurnal Teknik Elektro," pp. 294–302, 2024, doi: 10.23917/emitor.v25i3.13552.

[17] D. T. Tran, T. N. Nguyen, T. Kim, T. Huynh, and K. K. Ahn, "Deep Deterministic Policy Gradient-Based Parameter Adaptation for Synchronous Sliding-Mode Control with Time-Delay Estimation in Dual-Arm Robot Manipulators Under System Uncertainties," pp. 1–26, 2026.

[18] K. A. W, E. Nuralim, R. Alfita, R. V. Nahari, M. Pramudia, and A. Fiqhi, "Implementation 4 – DoF Arm Robot Object Sorting Controlled Based On Color Using Inverse Kinematics Algorithm," vol. 1, no. Icst, pp. 539–544, 2018.

[19] H. Hamuda, "Komputika : Jurnal Sistem Komputer Rancang Bangun Sistem Monitoring Stok Miniatur Berbasis IoT Menggunakan Sensor Ultrasonik dan Load Cell Design of an IoT-based Miniature Stock Monitoring System Using Ultrasonic Sensors and Load Cells," vol. 14, no. April, pp. 11–19, 2025, doi: 10.34010/n2yvsh27.

[20] W. Yuanyang, M. N. Mahyuddin, and S. Member, "Grasping Deformable Objects in Industry Application : A Comprehensive Review of Robotic Manipulation," no. November 2024, pp. 33403–33423, 2025.

[21] J. Prakash, S. A. M. Altnji, K. Thiyagarajan, A. R. O. Y. Chowdhury, and S. Member, "Human-Aware Robot Collaborative Task Planning Using Artificial Potential Field and DQN Reinforcement Learning," vol. 13, no. August, pp. 140889–140899, 2025.

[22] M. Cheng, H. Liao, C. Lan, and S. Member, "A 4-DoF Pick-and-Place Robot With High Structural Rigidity and Pose Repeatability," 2025.

[23] A. G. M. A. Aziz, S. Al Dawsari, A. E. Rafaat, A. G. A. El-Magd, and A. A. Z. Diab, "A Smart Four-DOF SCARA Robot: Design, Kinematic Modeling, and Machine Learning-Based Performance Evaluation," pp. 1–30, 2026.

[24] K. A. W., E. Nuralim, R. Alfita, R. V. Nahari, M. Pramudia, and A. Fiqhi, "Implementation 4-DoF Arm Robot Object Sorting Controlled Based on Color Using Inverse Kinematics Algorithm," in Proc. ICST, vol. 1, pp. 539–544, 2018.

p-ISSN: 2527-6336 | e-ISSN: 2656-7075

Halaman ini sengaja dikosongkan

Corresponding Author: dosen02886@unpam.ac.id
Doi: