EMS511U - Robot Design and Mechatronics

Individual Technical Report

Author: Sara Hamandi
Student ID: 230346871
Team: B6
Team Members: Asia Roberts, Aneeq Rehman

Abstract

This report presents the design and implementation of a two-degree-of-freedom planar robotic manipulator developed as part of the EMS511U group project. The selected mechanism was a five-bar, pantograph-type linkage actuated by two base-mounted DC motors. The work combined conceptual design, CAD modelling, kinematic analysis, mechanical fabrication, electronics integration, and preliminary controller development in a single prototype system. The final concept used a compact base structure, equal-length moving links in the CAD arrangement, revolute joints supported by bearings, and a gripper-style end effector.

A geometric method was used to derive the forward and inverse kinematics of the mechanism. Inverse kinematics was formulated by relating a desired Cartesian point to the two grounded actuator joints, while forward kinematics was obtained from the circle-intersection geometry of the distal links. The kinematic model was then verified numerically in Simulink using a round-trip consistency test. Sinusoidal joint-angle inputs were applied to the forward kinematics block, the resulting Cartesian coordinates were passed into the inverse kinematics block, and the reconstructed joint variables were compared with the original signals. The maximum absolute reconstruction errors were 4.4409×10^{-16} rad for Joint 1 and 5.5511×10^{-16} rad for Joint 2, confirming numerical consistency of the implemented model over the tested trajectory.

The prototype incorporated an Arduino Mega 2560, EMG30 geared DC motors with encoders, an L298N motor driver, an external DC power supply, and a mechanical linkage assembly developed through CAD and physical fabrication. The PID controller results were developed in MATLAB/Simulink using simulated motor-plant models rather than physical PWM actuation and measured encoder feedback. This simulation-based approach allowed the two-motor control architecture, controller gains, transient response, overshoot, settling behaviour, and steady-state error to be evaluated before full hardware implementation. The project highlighted the practical importance of calibration, structural stiffness, backlash reduction, wiring quality, and consistency between the mathematical model and the embedded implementation. The principal limitations observed in the prototype were backlash, screw interference, wiring complexity, voltage drop sensitivity, and the requirement for correct matching of encoder polarity, motor direction, and angular scaling. These observations inform the recommendations for future improvement.

Individual teamwork statement: My main contributions were the engineering drawings, the inverse and forward kinematic equations, and the Simulink verification. Asia Roberts contributed to the CAD and robot design, as well as the Simulink implementation, while Aneeq Rehman supported the robot wiring, troubleshooting, and assembly. As a team, we all contributed to the PowerPoint presentation, robot assembly, and wiring.

1. Introduction

The objective of this project was to design, build, and evaluate a two-degree-of-freedom robotic system integrating mechanical design, mechatronic hardware, kinematic modelling, and feedback control. The selected robot was a planar five-bar manipulator driven by two base-mounted actuated joints. This configuration was chosen because it provides constrained planar motion while keeping the actuators fixed to the base, reducing moving mass and producing a compact layout suitable for student-level fabrication.

A five-bar mechanism was selected because it provides an effective balance between mechanical simplicity, planar motion capability, and control relevance. Unlike a serial two-link arm, the actuators can remain fixed at the base, which reduces the inertia of the moving structure and improves the practicality of the design for a small prototype. At the same time, the closed-chain arrangement introduces useful engineering challenges, including branch selection in the kinematic model, sensitivity to joint alignment, and the need for coordinated control of both actuated joints.

The project developed from initial sketches into a CAD assembly, physical prototype, electronics layout, kinematic model, and MATLAB/Simulink control simulation. The final system used a lightweight planar linkage, EMG30 encoder-capable motors, an Arduino Mega 2560, an L298N motor driver, and a two-loop PID joint-control architecture. This report describes the kinematics, CAD design, manufacturing, electronic integration, and simulation-based controller evaluation. Section 3 presents the two-motor PID results, and the appendices provide supporting CAD, PID, wiring, and bill-of-materials information.

2. Materials and Methods

2.1 Forward and Inverse Kinematics

2.1.1 Mechanism Definition

The robot was modelled as a planar five-bar closed-chain mechanism with two grounded revolute joints and one common end-effector point. The left base joint was defined as

$$A = (0, 0)$$

and the right base joint as

$$B = (d, 0)$$

where d is the fixed horizontal spacing between the two actuator axes. The left proximal and distal link lengths were denoted l_1 and l_2, while the right proximal and distal link lengths were denoted l_3 and l_4. The actuated joint angles were denoted θ_1 and θ_2, and the end-effector point was denoted

$$P = (x, y)$$

The kinematic derivation used a geometric modelling approach, in which the joint variables are related to the Cartesian end-effector position through link-length constraints and trigonometric relationships [1, 2]. This method is appropriate for the five-bar mechanism because the end-effector position is obtained from the intersection of two distal-link circles.

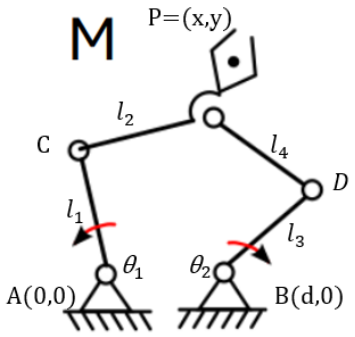


Figure 1: Kinematic model of the five-bar manipulator showing A, B, C, D, P, d, l_1–l_4, θ_1, and θ_2

2.1.2 Intermediate Joint Coordinates

Let the intermediate joints on the left and right sides of the mechanism be C and D, respectively. Their coordinates are

$$x_C = l_1 \cos\theta_1, \qquad y_C = l_1 \sin\theta_1$$

and

$$x_D = d + l_3 \cos\theta_2, \qquad y_D = l_3 \sin\theta_2$$

Since P lies on a circle of radius l_2 centred at C, and also on a circle of radius l_4 centred at D, the geometric constraints are

$$(x - x_C)^2 + (y - y_C)^2 = l_2^2$$

and

$$(x - x_D)^2 + (y - y_D)^2 = l_4^2$$

2.1.3 Forward Kinematics

Forward kinematics was used to determine $P = (x, y)$ from the two actuated joint angles. The distance between C and D is

$$r = \sqrt{(x_D - x_C)^2 + (y_D - y_C)^2}$$

The projection term and perpendicular offset are

$$a = \frac{l_2^2 - l_4^2 + r^2}{2r},$$

and

$$h = \sqrt{l_2^2 - a^2}$$

The end-effector coordinates were therefore written as

$$x = x_C + \frac{a(x_D - x_C)}{r} \pm \frac{h(-(y_D - y_C))}{r}$$

and

$$y = y_C + \frac{a(y_D - y_C)}{r} \pm \frac{h(x_D - x_C)}{r}$$

The $\pm$ term represents the two possible geometric branches of the circle-intersection problem. The implementation used the branch that matched the physical assembly. The coordinate system and main geometric variables are shown in Figure 1.

2.1.4 Inverse Kinematics

Inverse kinematics was used to calculate the required actuator angles for a desired end-effector point $P = (x, y)$. The distances from the target point to the two grounded joints were

$$r_A = \sqrt{x^2 + y^2}$$

and

$$r_B = \sqrt{(x - d)^2 + y^2}$$

Using the cosine rule, the actuator angles were obtained as

$$\theta_1 = \operatorname{atan2}(y, x) \pm \cos^{-1}\left(\frac{l_1^2 + r_A^2 - l_2^2}{2 l_1 r_A}\right)$$

and

$$\theta_2 = \operatorname{atan2}(y, x - d) \pm \cos^{-1}\left(\frac{l_3^2 + r_B^2 - l_4^2}{2 l_3 r_B}\right)$$

The same branch convention was used in both the analytical model and the Simulink implementation.

2.1.5 Workspace and Singularities

The workspace is constrained by the reach of both subchains. A point is reachable only if

$$|l_1 - l_2| \leq r_A \leq l_1 + l_2$$

and

$$|l_3 - l_4| \leq r_B \leq l_3 + l_4$$

The reachable Cartesian workspace is therefore the overlap of the two annular regions defined by the left and right link pairs. Singular configurations occur when the two circles become tangent and h approaches zero. Near these configurations, the mechanism loses local dexterity and small joint errors may create large end-effector errors, making singularity avoidance important during operation.

2.1.6 Simulink Numerical Verification

The implemented kinematic model was verified numerically in Simulink using the round-trip consistency model shown in Figure 2. Sinusoidal joint-angle inputs $\theta_1(t)$ and $\theta_2(t)$ were passed through the forward kinematics block to compute (x, y), then through the inverse kinematics block to reconstruct $(\hat{\theta}_1, \hat{\theta}_2)$. The resulting trajectories are compared in Figure 3, and the maximum reconstruction errors are summarised in Table 1. The verification structure was

$$(\theta_1, \theta_2) \rightarrow \text{Forward Kinematics} \rightarrow (x, y) \rightarrow \text{Inverse Kinematics} \rightarrow (\hat{\theta}_1, \hat{\theta}_2)$$

This round-trip verification was important because it checked the internal consistency of the analytical model before using it for control interpretation. If the forward and inverse kinematics were implemented inconsistently, the reconstructed joint angles would deviate from the original input signals even in simulation. The near-perfect overlap in Figure 3 therefore confirms that the selected branch convention, link-length definitions, coordinate frame, and trigonometric relationships were implemented consistently in Simulink. This provides confidence that any later difference between simulated and physical robot behaviour is more likely to arise from practical effects such as backlash, friction, link compliance, encoder calibration, or assembly tolerance rather than from an error in the kinematic equations themselves.

The reconstruction errors were defined as

$$e_{\theta_1}(t) = \theta_1(t) - \hat{\theta}_1(t)$$

and

$$e_{\theta_2}(t) = \theta_2(t) - \hat{\theta}_2(t)$$

The maximum absolute errors obtained from MATLAB were

$$\max |e_{\theta_1}| = 4.4409 \times 10^{-16} \text{ rad}$$

and

$$\max |e_{\theta_2}| = 5.5511 \times 10^{-16} \text{ rad}$$

These values correspond to floating-point round-off error, confirming numerical consistency over the tested sinusoidal motion range. This verification is numerical rather than experimental and does not include backlash, friction, encoder quantisation, structural compliance, or assembly tolerances.

Table 1: Numerical kinematic verification results

Signal comparison	**Maximum absolute error (rad)**
θ_1 vs. $\hat{\theta}_1$	4.4409×10^{-16}
θ_2 vs. $\hat{\theta}_2$	5.5511×10^{-16}

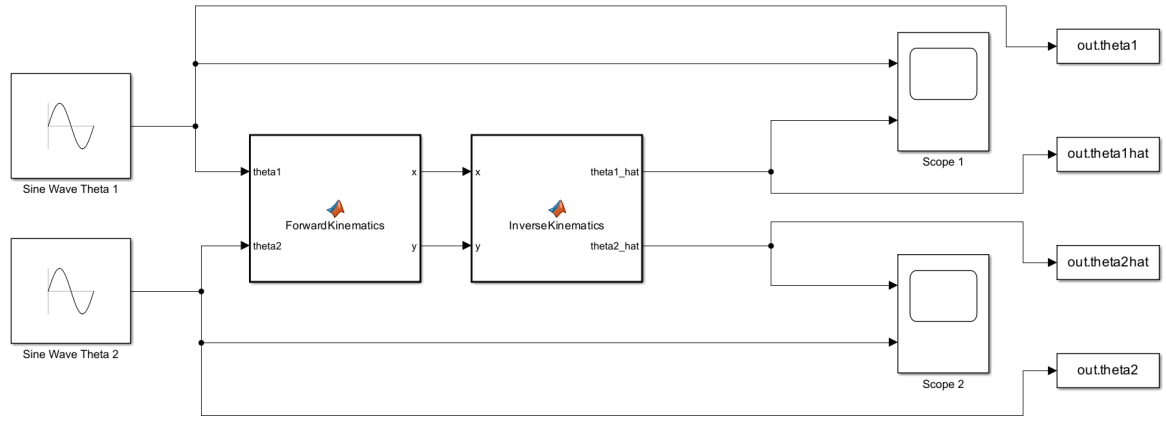
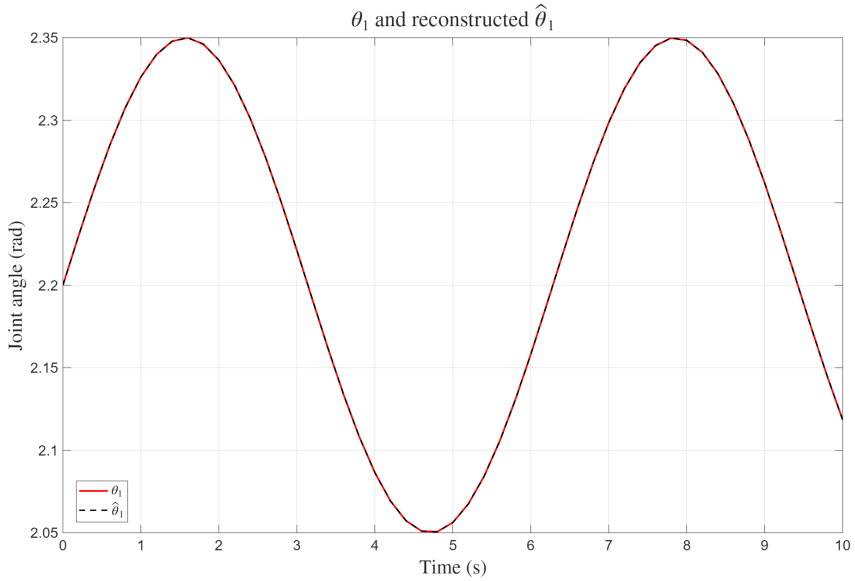
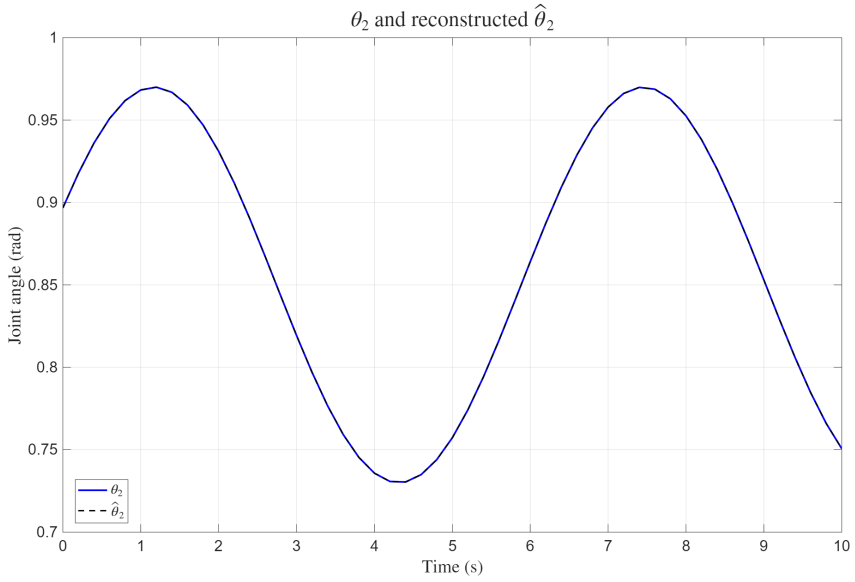


Figure 2: Simulink round-trip verification model using sinusoidal joint inputs, forward kinematics, and inverse kinematics



(a) Scope 1



(b) Scope 2

Figure 3: Comparison of input joint trajectory θ_1 and reconstructed trajectory $\hat{\theta}_1$, and comparison of input joint trajectory θ_2 and reconstructed trajectory $\hat{\theta}_2$

2.2 Design Sketches and Concept Development

The concept-development stage defined the overall linkage arrangement, grounded-joint positions, end-effector location, and approximate space available for actuation and support components. The initial sketches in Figure 4 show the early development of the base, motor positions, and linkage dimensions, while Figure 5 shows the final concept sketch for the assembled robot.

The concept was guided by four main objectives: to achieve planar 2-DOF motion, keep the motors mounted on the base to reduce moving inertia, maintain an approximately symmetric linkage for balanced motion, and provide an end-effector region capable of supporting a gripper. Refining the geometry at the sketch stage reduced ambiguity during CAD modelling and helped ensure consistency between the mechanical concept and the intended control architecture.

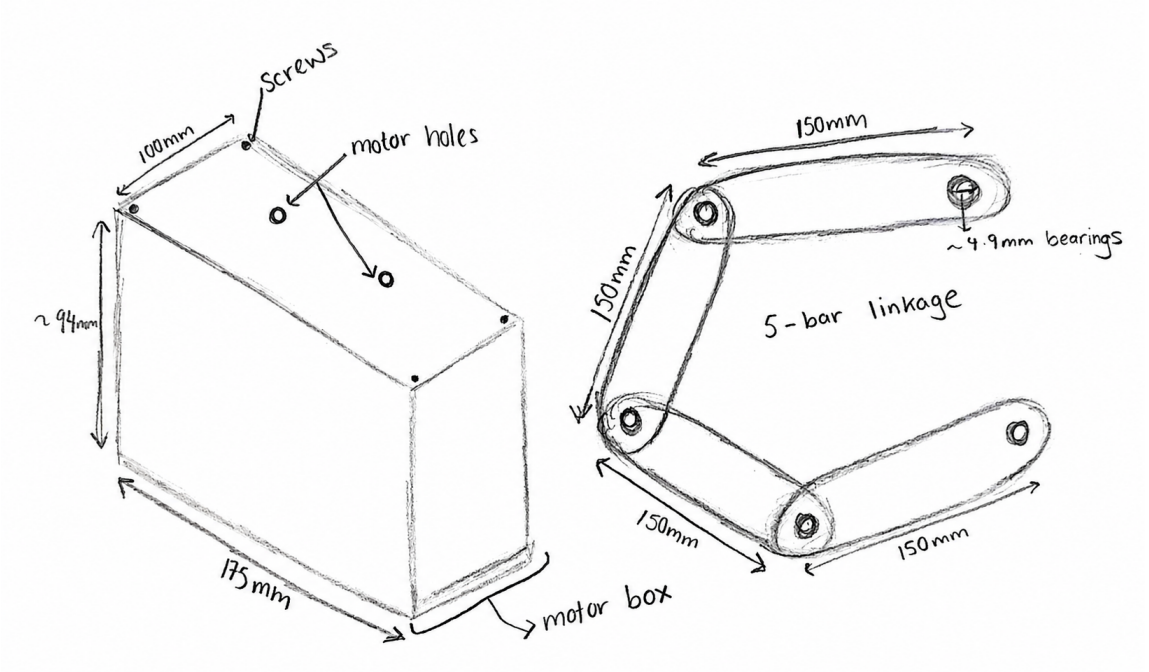


Figure 4: Initial concept sketches

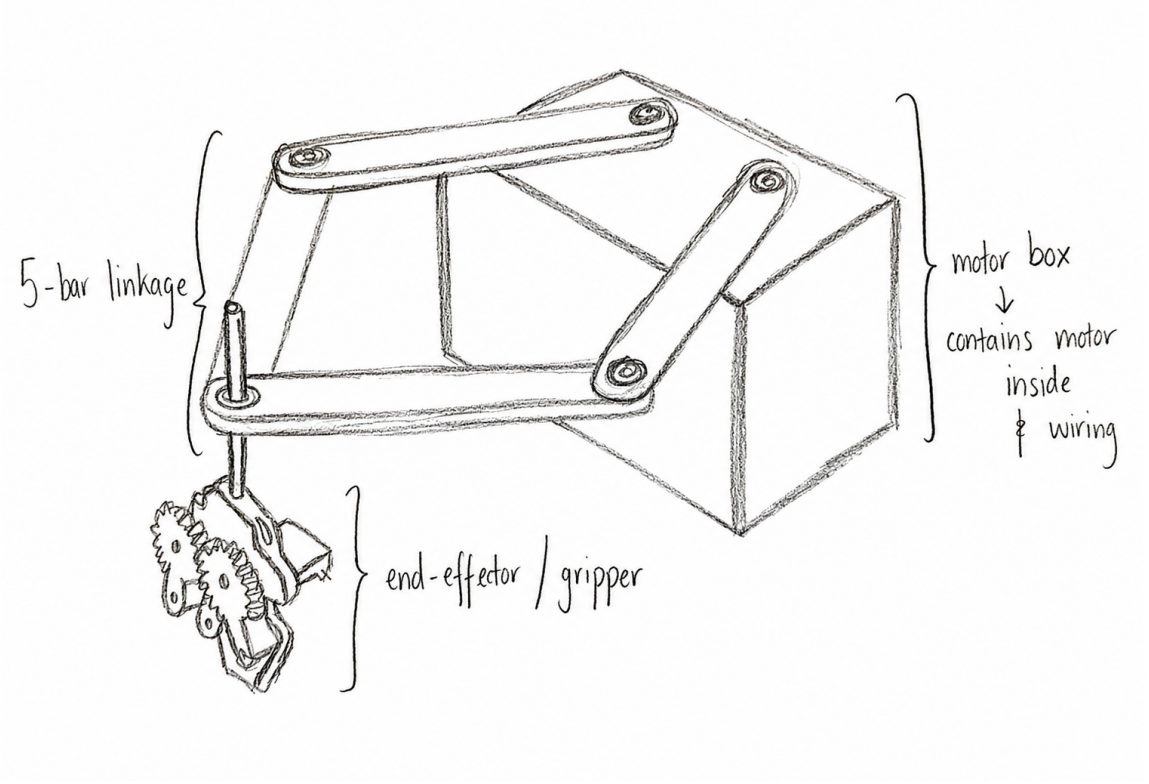


Figure 5: Final concept sketch for assembled robot

2.3 CAD Design and Description

The CAD model translated the sketch-stage concept into a full assembly containing the base, motor mounts, bearings, revolute joints, moving links, connecting rods, and gripper/end-effector region. It was used to confirm geometric compatibility between the linkage, actuators, and support

structure before manufacture.
The final CAD arrangement used equal-length moving links and base-mounted actuation to support approximately symmetric planar motion while reducing moving mass. Bearings were included at the revolute joints to improve rotational smoothness and reduce friction. The base structure supported the motors, fixed the lower joint spacing, improved stiffness, and provided the main interface between the robot and the work surface.

Two representative parts are discussed below:

Motor Mount - This part positioned the motor relative to the base and transmitted motor reaction torque into the support structure. Its function was to preserve shaft alignment, provide local stiffness, and maintain a reliable interface between the actuator and the linkage input joint. The drawing in Figure 6 documents the motor-mount geometry and dimensions.

Motor Box / **Base Plate -** This component formed the main structural housing of the robot and maintained the fixed spacing between the grounded joints. It also provided space for the motors and wiring, improved structural stiffness, and protected the internal electromechanical components. Its compact geometry reduced the overall footprint, but increased constraints on wiring access, fastener placement, and motor clearance. Figure 7 shows the base-plate and motor-box geometry, and Table 2 summarises the individual CAD responsibilities.

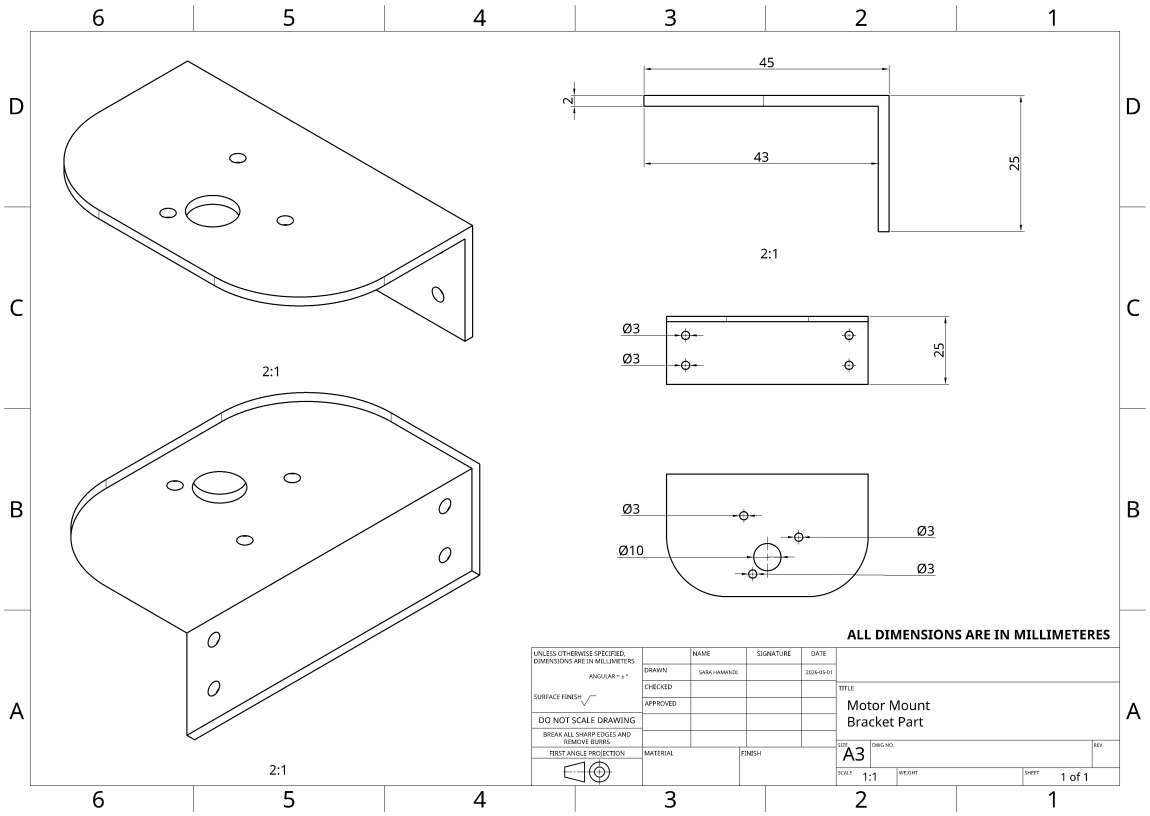


Figure 6: Motor Mount component drawing

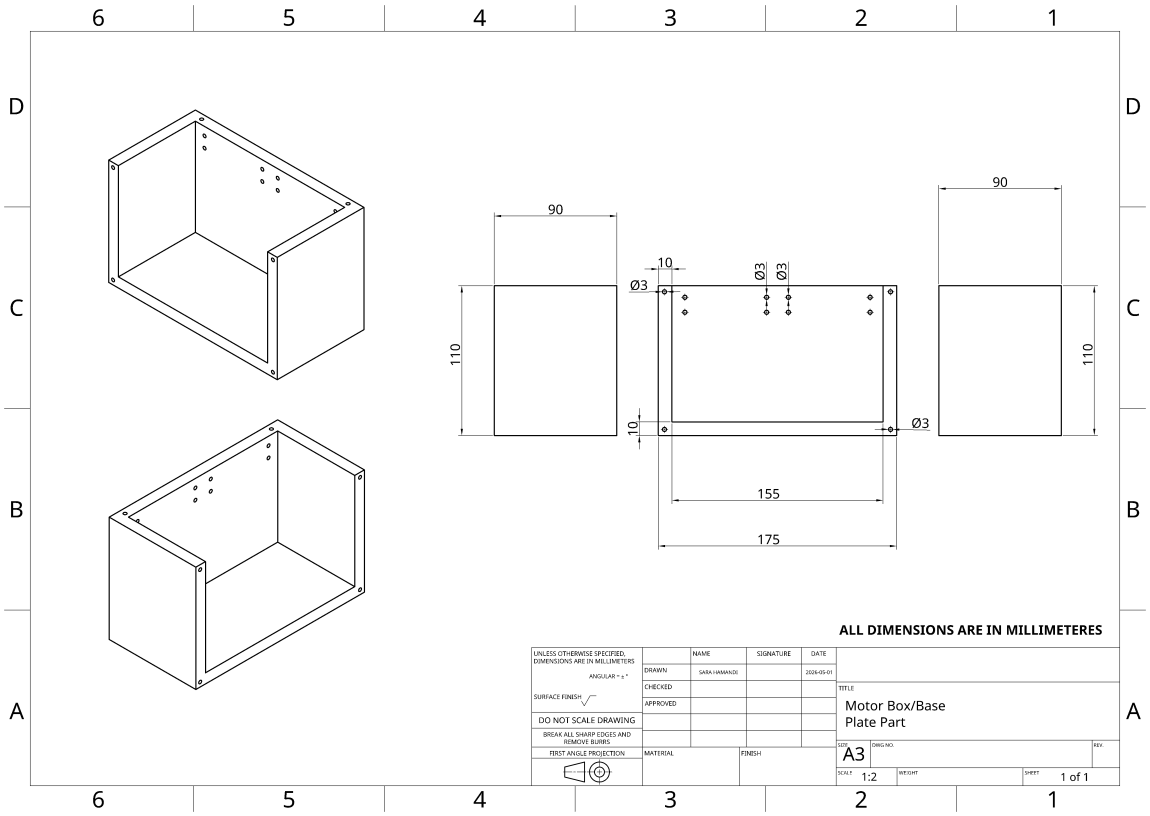


Figure 7: Motor Box/Base Plate component drawing

Table 2: Team contribution to CAD development

Team member	CAD responsibility
Sara Hamandi	Engineering Drawings
Asia Roberts	Assembly of parts, linkage between parts etc
Aneeq Rehman	Base plate/Motor Box

2.3.1 Engineering Reasoning and Design Trade-Offs

Several engineering trade-offs influenced the final CAD design. A symmetric linkage improves motion regularity, but the base becomes crowded by motors, shafts, fasteners, and wiring. Lightweight links reduce motor load, but insufficient stiffness increases compliance and makes backlash more visible. Compact packaging improves the footprint and appearance, but reduces maintainability and wiring access. The use of PLA 3D-printed parts made manufacturing accessible and fast, although it provided lower dimensional precision and stiffness than machined metal parts. These trade-offs shaped the final choice of a compact base-mounted layout with bearing-supported joints and printable structural components.

2.4 Manufacturing and Assembly

The prototype was assembled from purchased components, 3D-printed PLA parts, and manually cut wooden linkage members. The motors, brackets, bearings, power electronics, microcontroller, and fasteners were standard off-the-shelf items. The base and several support components were produced using PLA 3D printing, while the final linkage arms were manufactured from wooden

planks cut to approximately 150 mm. Wooden links were used because the originally intended printed linkage parts presented dimensional and fit issues during fabrication.
Assembly showed that manufacturing quality had a strong effect on robot performance. Joint clearance, shaft alignment, screw length, and manually drilled link holes all influenced backlash, symmetry, and resistance within the mechanism. As a result, the physical prototype did not fully match the ideal assumptions of the kinematic model. Future improvements should focus on tighter joint fits, more accurate linkage manufacture, improved bearing seating, shorter fasteners, and stiffer link materials. The assembled prototype is shown in Figure 8.

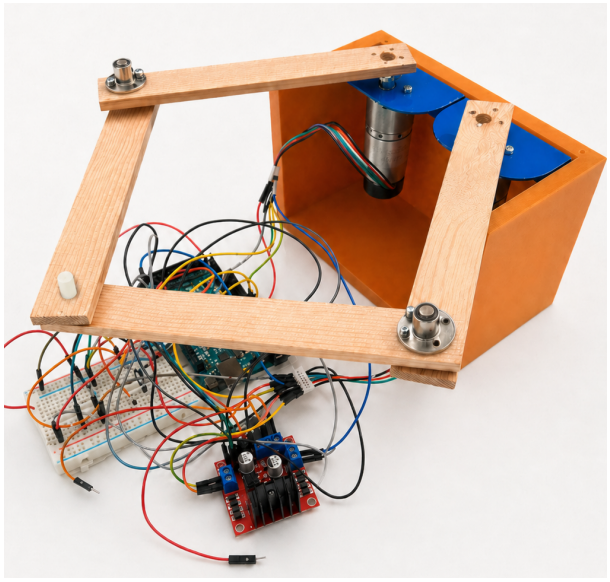


Figure 8: Photograph of the assembled prototype

2.5 Electronic and Electromechanical Components

The robot prototype incorporated an Arduino Mega 2560, EMG30 geared DC motors with integrated encoders, an L298N motor driver, an external DC power supply, and associated wiring. The Arduino Mega 2560 was selected because it provides sufficient input/output capability for motor-driver interfacing, encoder reading, and future closed-loop control implementation [3]. The EMG30 motors were selected because they combine geared DC actuation with integrated encoder feedback, making them suitable for joint-level angular-position control [5]. The L298N module acted as the interface between the Arduino and motors, allowing bidirectional actuation through direction and PWM control signals [4].
Although the hardware included motors, encoders, and motor-driver circuitry, the PID response plots in this report were not obtained from physical PWM actuation or measured encoder feedback. Due to time constraints, the closed-loop PID investigation was completed in MATLAB/Simulink using simulated motor-plant models and simulated angular-position feedback. The simulation evaluated controller structure, gain selection, transient response, and steady-state behaviour, but did not fully include practical effects such as encoder quantisation, electrical noise, friction variation, backlash, voltage drop, wiring resistance, and imperfect calibration.
For future hardware implementation, reliable encoder feedback will be essential for comparing desired and actual joint positions. Correct operation will also require consistent wiring, correct signal polarity, accurate conversion between encoder counts and joint angle, and calibration of motor direction relative to the mathematical model.

2.6 PID Control Strategy

Joint position control was investigated using proportional-integral-derivative (PID) control. For each simulated motor, the controller compared the desired joint angle with the angular-position output of the simulated motor plant and generated a control effort to reduce the tracking error. The continuous-time PID law is

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}$$

where $e(t)$ is the angular-position error and $u(t)$ is the controller output [6]. A discrete form and PWM explanation are provided in Appendix B.

Because the robot has two actuated joints, the simulation used two independent PID loops. Independent tuning was appropriate because the two joints may not have identical transient behaviour due to differences in loading, friction, effective inertia, and assembly tolerance. In Simulink, each loop included error calculation, PID control, output limiting, simulated motor dynamics, and angular-position feedback. The selected gains and resulting performance are presented in Section 3 and Table 4.

Although the PID loops were simulated independently, their performance is still directly linked to the five-bar robot behaviour because θ_1 and θ_2 are the actuated variables used in the kinematic model. Any difference in rise time, overshoot, or settling time between the two joints can produce temporary Cartesian tracking error at the end effector. For this reason, the controller cannot be judged only by whether each motor reaches its own reference angle; it must also be considered in terms of how synchronised the two joint responses are during coordinated robot motion.

2.7 Control System Architecture and Integration

The control architecture was implemented in Simulink, as shown in Figure 9. Each joint was represented by an independent closed-loop PID controller acting on a simulated motor plant. The same structure was duplicated for both motors, producing a dual-loop joint-control architecture suitable for a two-motor planar robot.

This simulation allowed the transient behaviour of each joint to be evaluated before full hardware implementation. The controlled variables correspond to the actuated joint angles θ_1 and θ_2 used in the kinematic model, so accurate joint control is necessary for the end-effector to reach the Cartesian positions predicted by the forward and inverse kinematics. Full trajectory-level validation from inverse kinematics to PID joint control and forward-kinematic end-effector tracking was not completed experimentally, and remains a key future improvement. The architecture therefore represents a staged development route rather than a final hardware controller. The first stage was to confirm that each joint could be stabilised using a PID loop in simulation. The next stage would be to replace the simulated angular-position feedback with encoder measurements from the EMG30 motors and replace the simulated actuator limiting with Arduino PWM commands through the L298N driver. This would allow the same control structure to be tested under real electrical and mechanical conditions while keeping the signal flow consistent with the Simulink model. The team contribution to control and electronics integration is summarised in Table 3.

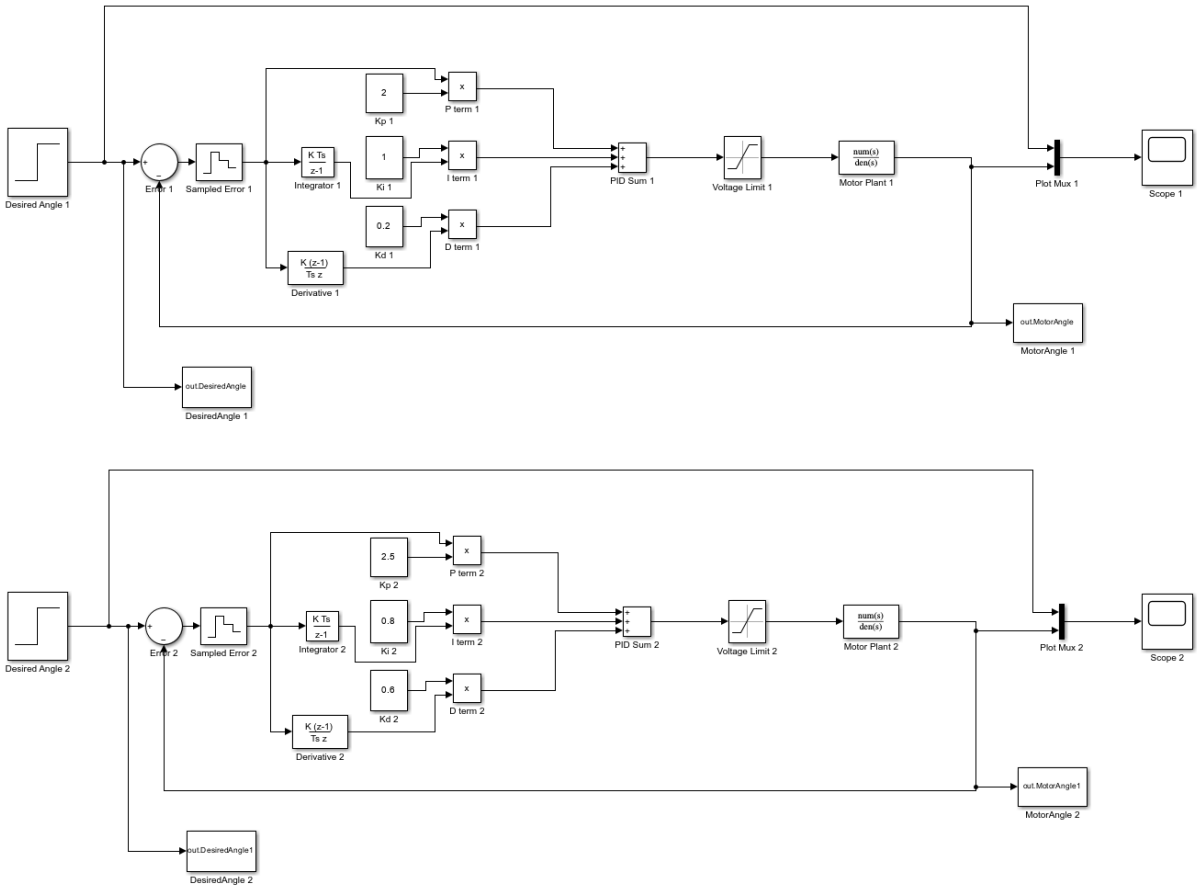


Figure 9: Dual-joint PID control architecture using simulated motor plant models

Table 3: Team contribution to control and electronics integration

Team member	**Main Tasks**
Sara Hamandi	Forward and Inverse Kinematics and Integration
Asia Roberts	Simulink Implementation
Aneeq Rehman	Supported with wiring and troubleshooting

3. Results

3.1 Two-Motor PID Control Performance

The two-joint control system was evaluated in MATLAB/Simulink using independent PID feedback loops. A step change in desired angle was applied to each joint, and the simulated angular response was recorded. These results are simulation results rather than physical encoder measurements, since the physical PWM and encoder-based closed-loop PID test was not completed within the available time.
The purpose of the test was to assess rise time, settling time, overshoot, and steady-state error for both actuated joints. The gain values and performance metrics are summarised in Table 4.

3.2 Joint 1 Response

Figure 10 shows the step response of Joint 1 for a reference angle of approximately 1.0 rad, comparing the tuned PID controller with an untuned baseline. The tuned response rose more quickly toward the reference and reduced the final deviation from the desired angle, although some overshoot remained during the transient. The tuned controller reached a peak slightly above the reference before gradually converging, while the untuned response rose more slowly and remained further from the desired value over the same simulation interval.
This comparison shows that the tuning process improved the dynamic response of Joint 1 by increasing responsiveness and reducing long-term error. However, the presence of overshoot indicates that the tuned gains still represent a compromise between speed and damping. In the context of the five-bar robot, this is important because temporary overshoot at the joint level can contribute to transient Cartesian deviation at the end effector, even when the final angular position error is small.

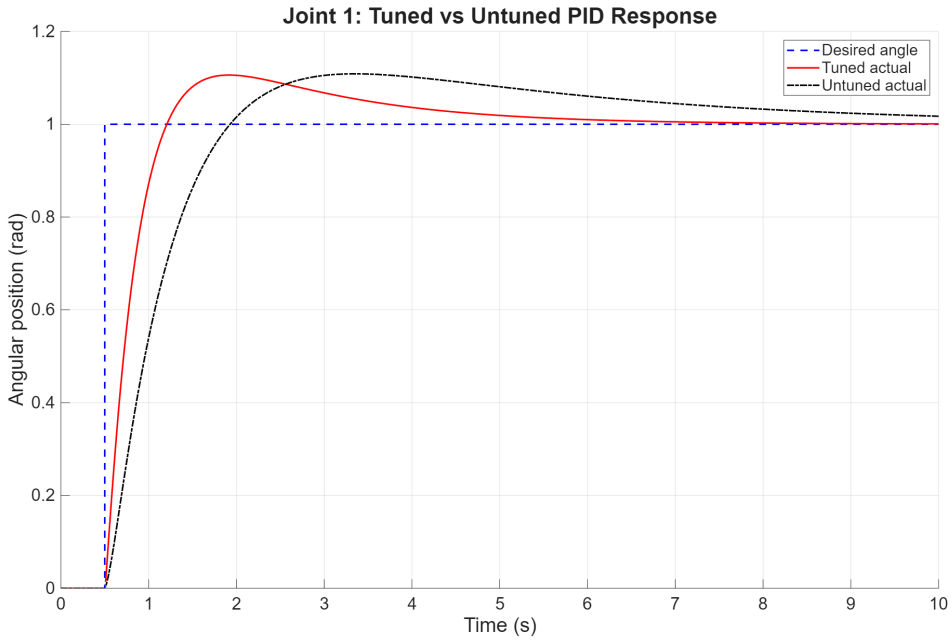


Figure 10: Joint 1 step response comparing the tuned PID controller with an untuned baseline. The dashed blue line shows the desired angle, the red line shows the tuned response, and the black dashed line shows the untuned response

3.3 Joint 2 Response

Figure 11 compares the tuned and untuned step responses of Joint 2. The tuned controller improved the transient behaviour relative to the untuned case, although the response remained slower to settle than Joint 1. For the tuned case, the rise time was 0.69 s, the settling time was 6.71 s, the overshoot was 8.1567%, and the steady-state error was 3.3479×10^{-3} rad. This comparison shows that gain tuning improved the overall dynamic response, but also confirms that the second joint remained slower and less tightly regulated than Joint 1.
Compared with Joint 1, the tuned response of Joint 2 remained slightly more gradual, which is consistent with the different gain selection used for the second motor. This difference in joint dynamics is significant for coordinated robot motion, since unequal rise and settling behaviour between the two joints can distort the end-effector path even if both joints ultimately converge to their target positions.

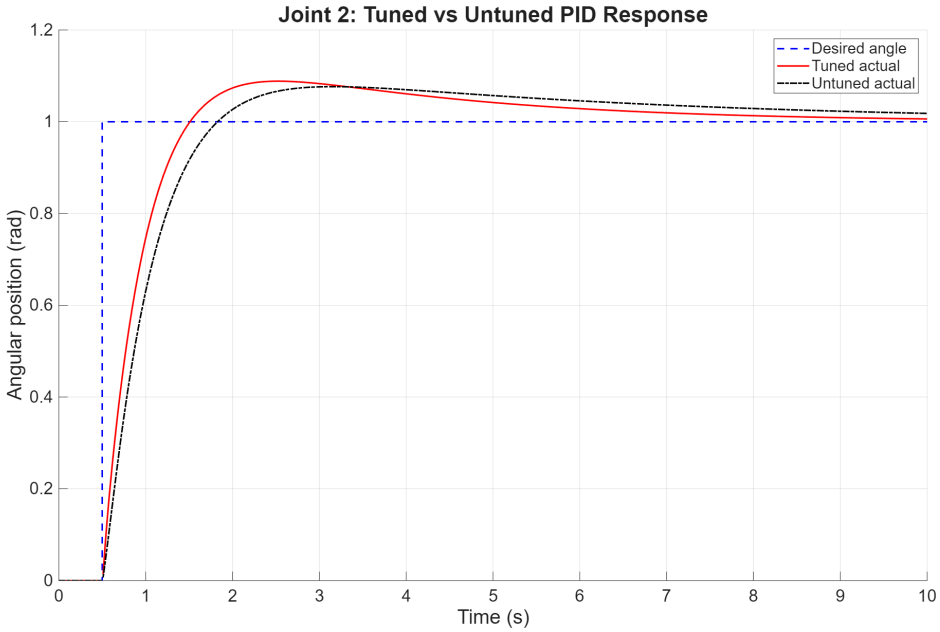


Figure 11: Joint 2 step response comparing the tuned PID controller with an untuned baseline. The dashed blue line shows the desired angle, the red line shows the tuned response, and the black dashed line shows the untuned response

3.4 Comparative Performance of the Two-Joint System

The comparative tuned-response results in Table 4 show that both controllers achieved stable simulated tracking with small steady-state error. Joint 1 responded faster and settled sooner, while Joint 2 produced lower overshoot but required more time to settle. The tuned-versus-untuned comparisons presented in Figures 10 and 11 also show that gain adjustment improved the transient behaviour of both joints relative to the untuned baseline.
This difference between the two tuned joint responses is important for a five-bar robot because unequal rise time, overshoot, and settling behaviour can cause the end effector to deviate temporarily from the ideal Cartesian trajectory predicted by the kinematic model. Further tuning should therefore focus on reducing the settling-time mismatch while maintaining low overshoot and small final error.

Table 4: Two-motor PID performance metrics

Joint	$\mathbf{K_p}$	$\mathbf{K_i}$	$\mathbf{K_d}$	Rise Time (s)	Settling Time (s)	Overshoot (%)	Steady-State Error (rad)
Joint 1	2.0	1.0	0.2	0.49	4.85	10.099	5.6047×10^{-4}
Joint 2	2.5	0.8	0.6	0.69	6.71	8.1567	3.3479×10^{-3}

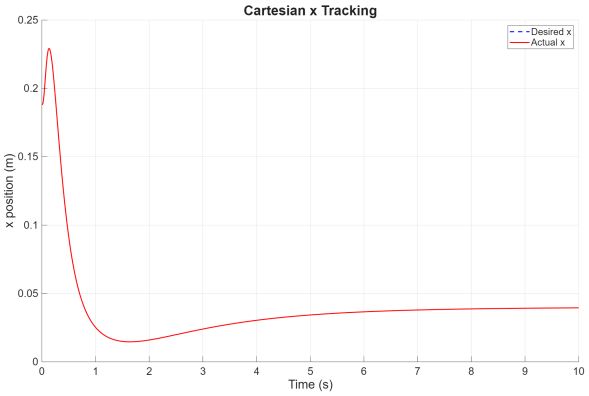
3.5 Coordinated Cartesian Point Tracking

To evaluate the robot at system level rather than at individual joint level, a coordinated Cartesian point-tracking test was performed in Simulink. A desired end-effector position of approximately $x_d = 0.04\,\text{m}$ and $y_d = 0.20\,\text{m}$ was specified in Cartesian space. The inverse-kinematics model converted this desired point into joint-angle references for the two actuated joints. These reference angles were then tracked by the two independent PID-controlled motor models. The resulting joint

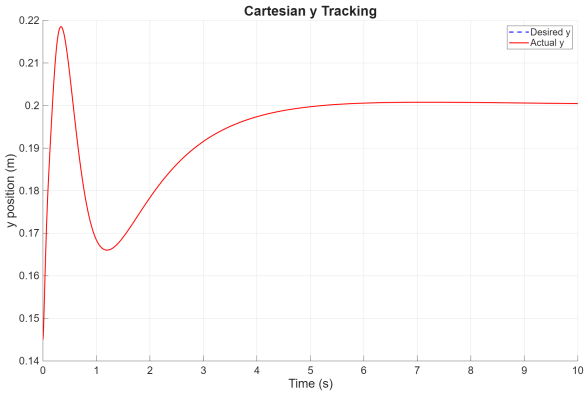
responses were passed through the forward-kinematics model to reconstruct the actual end-effector trajectory.

Figures 12a, 12b, 12c, and 12d show the corresponding Cartesian tracking results. The end effector converged to the desired point with a final position error approaching zero by the end of the simulation. This confirms that the inverse kinematics, joint-level PID controllers, and forward kinematics operated consistently as an integrated closed-loop system.

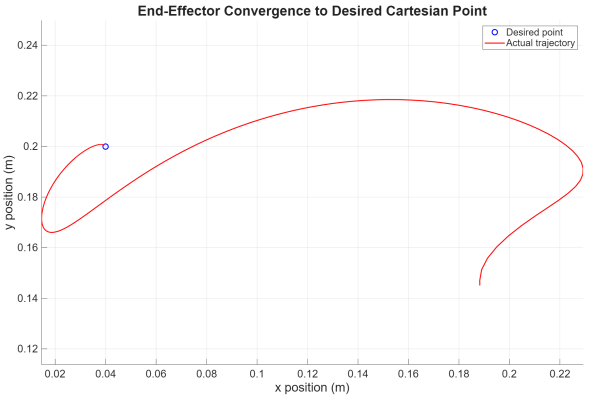
However, the transient response was not direct. The end effector initially followed a curved path with noticeable deviation from the target before converging. This behaviour is consistent with the overshoot and differing settling times previously observed in the individual joint responses. In particular, the mismatch between the transient responses of Joint 1 and Joint 2 caused temporary Cartesian deviation, even though both joints ultimately reached their reference values. The result therefore demonstrates successful coordinated Cartesian regulation, while also indicating that further controller tuning would be beneficial to reduce transient path distortion and improve Cartesian accuracy during motion.



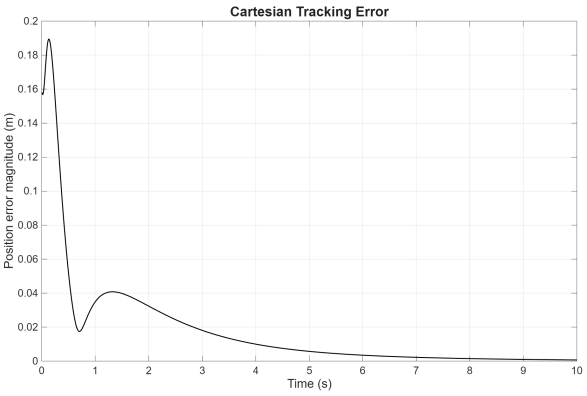
(a) Desired and actual Cartesian x-position during coordinated end-effector point tracking



(b) Desired and actual Cartesian y-position during coordinated end-effector point tracking



(c) Actual end-effector trajectory in Cartesian space during convergence to the desired point



(d) Magnitude of Cartesian position error during coordinated end-effector point tracking

Figure 12: Coordinated Cartesian point-tracking results obtained by linking inverse kinematics, dual joint-level PID control, and forward kinematics

Table 5: Summary of coordinated Cartesian point-tracking performance

Tracking Type	Max Cartesian Error (m)	RMS Cartesian Error (m)
Fixed Cartesian point	0.189415	0.037619

The coordinated Cartesian point-tracking results are shown in Figure 12 and summarised in Table 5. For the fixed desired Cartesian target, the maximum position error was 0.189415 m, while the RMS Cartesian error over the simulation was 0.037619 m. The relatively large maximum error occurred during the initial transient, when the two joints responded at different rates and the end effector followed a curved path before converging to the target. The lower RMS error indicates that the position error reduced substantially as the response progressed. This confirms that the robot was able to regulate to the desired Cartesian point, while also showing that transient mismatch between the two joint responses produced temporary end-effector deviation. The result therefore demonstrates successful coordinated Cartesian regulation, although further tuning would be required to improve transient path accuracy.

3.6 Interpretation

Overall, the two-motor simulation confirmed that the proposed control architecture could regulate both actuated joints using independent PID loops and could also achieve coordinated Cartesian point regulation through the combined inverse kinematics, PID, and forward kinematics chain. However, the results remain simulation-based and do not yet include hardware effects such as backlash, friction, encoder quantisation, voltage drop, or motor-driver non-idealities. The simulation should therefore be treated as a controller-design baseline rather than final experimental validation.

4. Discussion and Conclusion

This project developed a 2-DOF planar five-bar robot from concept to physical prototype, supported by CAD design, kinematic modelling, mechanical assembly, electromechanical integration, and simulation-based PID control. The main strengths of the work were the successful derivation of the forward and inverse kinematics, the numerical Simulink verification of the kinematic model, and the implementation of a two-joint PID structure.

The robot-level control results showed that both joints could be regulated with stable simulated responses and low steady-state error. Joint 1 responded faster, while Joint 2 had lower overshoot but settled more slowly. This difference was also reflected in the coordinated Cartesian point-tracking test, where the end effector converged to the target point but followed a curved transient path with noticeable temporary error. This shows that joint-level control alone is not enough to assess overall robot performance, since differences between the two joint responses directly affect end-effector motion.

The main limitations were backlash, joint clearance, alignment error, wiring complexity, and the fact that the PID validation was completed in simulation rather than through full hardware PWM and encoder feedback. As a result, practical effects such as friction variation, voltage drop, encoder quantisation, and motor-driver non-idealities were not fully captured.

Future work should focus on improving mechanical stiffness, reducing backlash, refining the wiring and schematic quality, and implementing the PID controller on the physical robot using measured encoder feedback. A further improvement would be full Cartesian trajectory tracking to evaluate end-effector accuracy directly. Overall, the project provides a strong foundation for future hardware-based closed-loop control of the robot.

Video summary of the technical report link: Robot Video Summary

References

[1] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, 2nd ed. Hoboken, NJ: Wiley, 2020.

[2] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. London, UK: Springer, 2009.

[3] Arduino, “Arduino Mega 2560 Rev3,” Accessed: May 6, 2026. Available: https://store.arduino.cc/products/arduino-mega-2560-rev3

[4] STMicroelectronics, “L298 dual full-bridge driver datasheet,” Accessed: May 6, 2026. Available: https://www.st.com/resource/en/datasheet/l298.pdf

[5] Robot Electronics, “EMG30 gearmotor with encoder,” Accessed: May 6, 2026. Available: https://www.robot-electronics.co.uk/emg30-gearmotor-with-encoder.html

[6] MathWorks, “Discrete PID Controller,” Accessed: May 6, 2026. Available: https://www.mathworks.com/help/simulink/slref/discretepidcontroller.html

A. Professional CAD Drawings

A.1 Robot Assembly Engineering Drawing

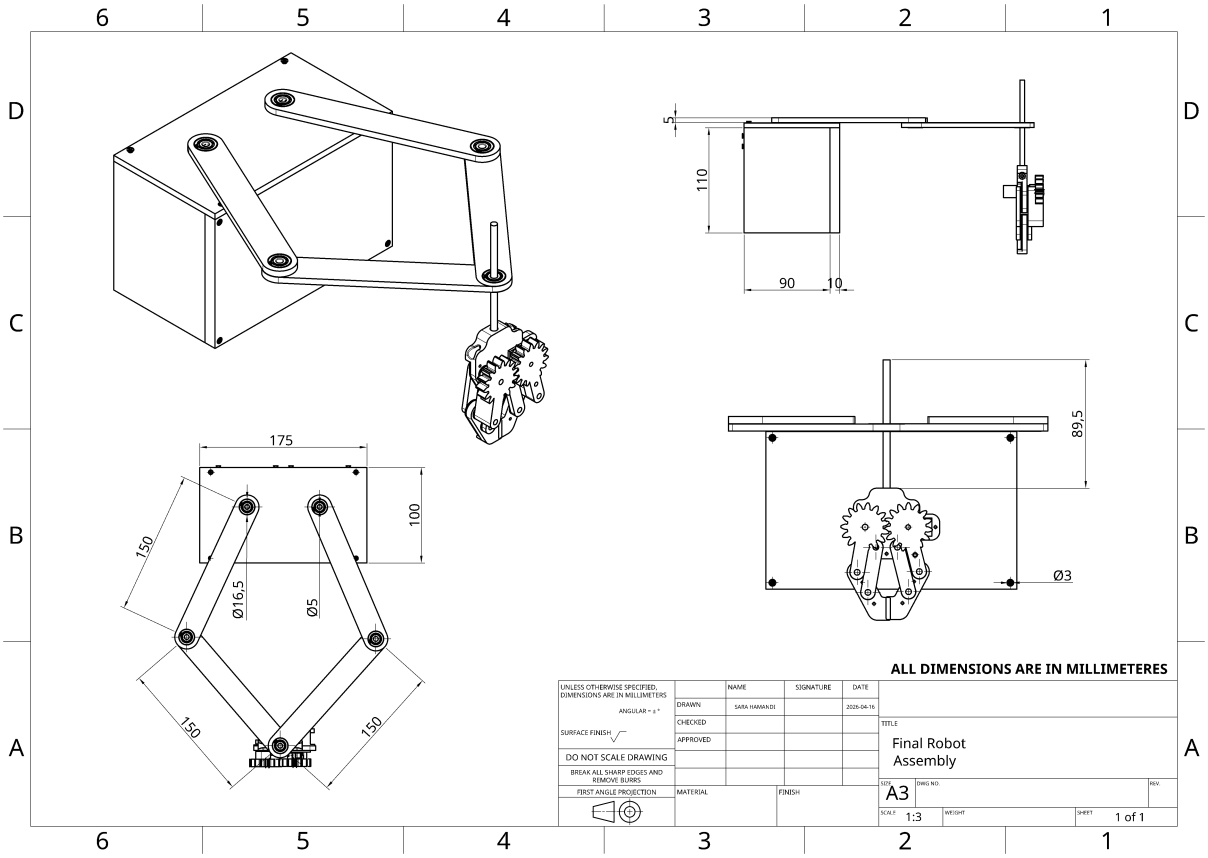


Figure 13: Final rendered assembly of the complete robot

A.2 Rendered Assembly

The rendered assembly views in Figures 14 and 15 provide additional visual evidence of the complete mechanical integration.

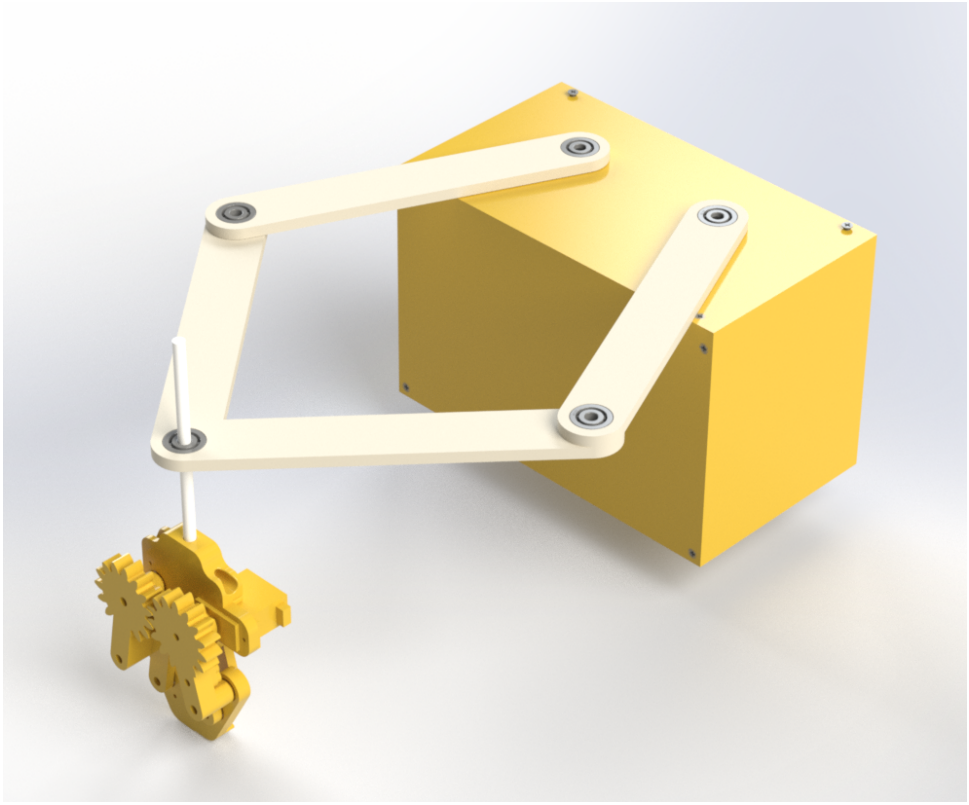


Figure 14: Final rendered assembly of the complete robot

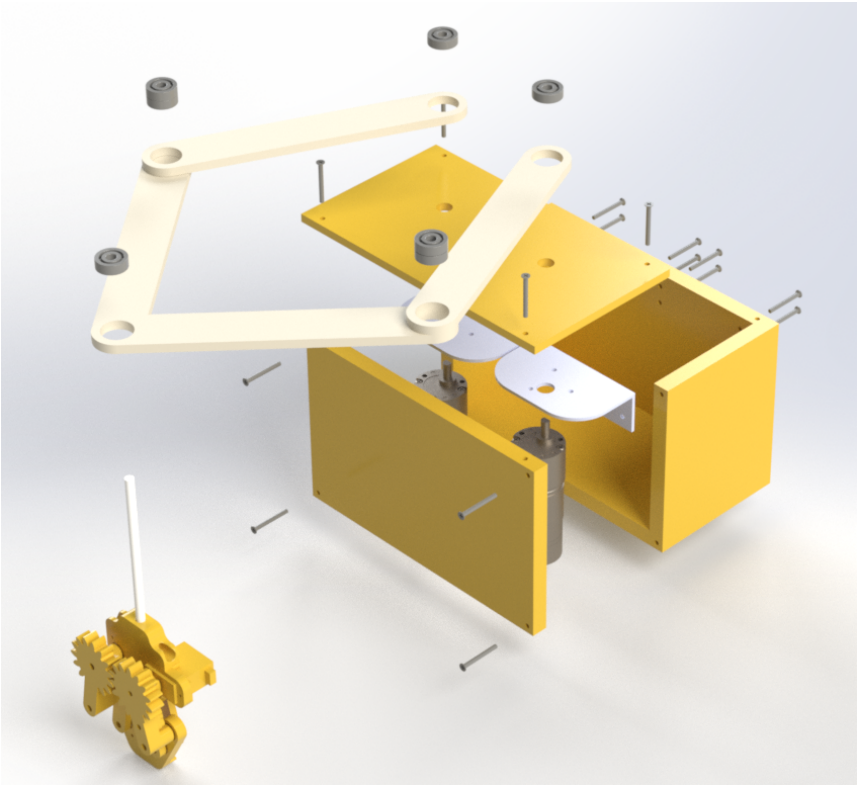


Figure 15: Exploded rendering of the robot assembly

B. PID Control of a DC Motor

B.1 System Description

The PID controller appendix provides additional detail on the single-motor control model used to support the two-joint simulation in the main report. The intended physical system consisted of an Arduino Mega 2560, an L298N motor driver, an EMG30 geared DC motor with integrated encoder feedback, and an external DC power supply. In a complete hardware implementation, the Arduino would read the encoder signal, calculate the angular-position error, compute the PID control output, and apply the command to the motor through PWM and direction signals sent to the L298N driver.

Due to time constraints, the physical PWM and encoder-based PID experiment was not completed. Therefore, the results in this appendix are simulation results generated in MATLAB/Simulink. The Simulink model in Figure 16 represents the same control structure that would be used in the physical system: reference angle, error calculation, PID control, actuator limiting, motor dynamics, and angular-position feedback. This allowed the controller behaviour to be evaluated before final hardware implementation.

The simulation made several simplifying assumptions. The motor plant was represented using a transfer-function model rather than a fully identified experimental model. The angular-position feedback was simulated rather than measured from the EMG30 encoder. The voltage/PWM limit was represented by a limiting block, and non-ideal physical effects such as backlash, friction variation, encoder quantisation, wiring losses, and mechanical compliance were not fully included. These assumptions mean that the simulated response is useful for controller design, but physical testing would still be required to confirm final robot performance.

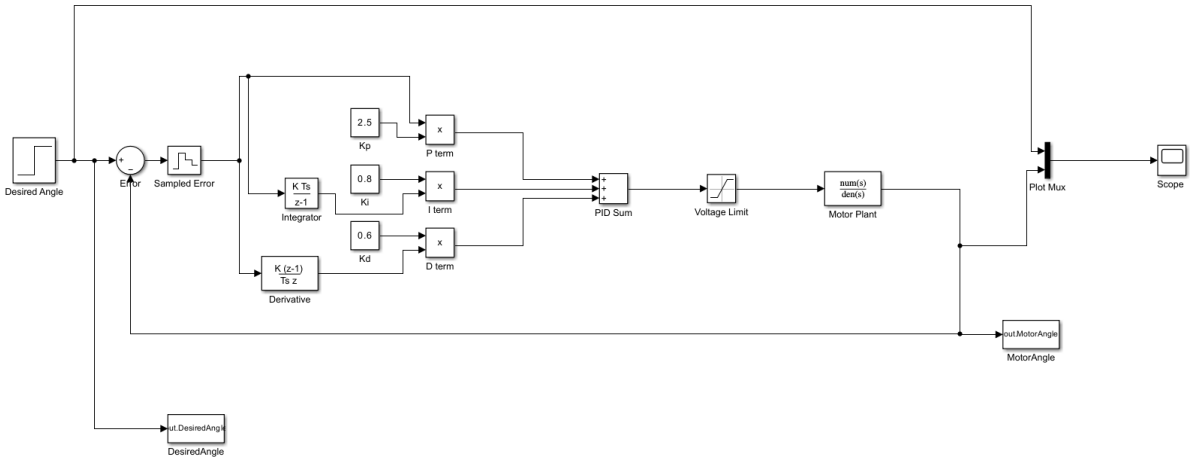


Figure 16: Single-motor PID control block diagram

B.2 Motor Plant Model and Assumptions

The single-motor PID study was carried out in MATLAB/Simulink using a simplified transfer-function representation of the motor plant. The transfer-function block used in the simulation had numerator coefficient 1.885 and denominator coefficients $[0.08 \;\; 1 \;\; 0]$, giving

$$G(s) = \frac{1.885}{0.08s^2 + s}$$

This model was used to represent the motor dynamics for preliminary controller tuning before full hardware implementation. It allowed the effect of changing the PID gains on rise time, overshoot, settling behaviour, and residual error to be evaluated in a controlled simulation environment.

The plant model was not obtained through full experimental system identification. As a result, the simulation should be interpreted as a preliminary controller-design study rather than a fully validated motor model. Practical effects such as backlash, encoder quantisation, friction variation, supply-voltage drop, and driver non-idealities were not fully captured.

B.3 PID and PWM Theory

A PID controller reduces the angular-position error between a desired motor angle and the actual or simulated motor angle. The continuous-time controller is written as

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}$$

where K_p, K_i, and K_d are the proportional, integral, and derivative gains respectively. The proportional term increases the immediate corrective action, the integral term reduces long-term steady-state error, and the derivative term adds damping by opposing rapid changes in error [6]. For microcontroller implementation, the controller must operate in discrete time. A suitable discrete approximation is

$$u[k] = K_p e[k] + K_i T_s \sum_{i=0}^{k} e[i] + K_d \frac{e[k] - e[k-1]}{T_s}$$

where T_s is the sampling time and k is the sample index. This equation reflects how the Arduino would update the control output at regular intervals using the latest encoder-based angular-position error.

PWM is the practical method used to apply the controller output to a DC motor through a motor driver. Instead of applying a continuously variable voltage, the driver rapidly switches the motor supply on and off. The duty cycle determines the effective average voltage applied to the motor. A higher duty cycle produces a larger average voltage and therefore a stronger motor response. In this project, PWM was not physically used to generate the plotted PID results; instead, the Simulink model represented the actuator command and its practical limits using a simulated control signal and saturation stage.

B.4 Simulation Results

The single-motor simulation was used as a preliminary PID tuning study to examine how changes in proportional, integral, and derivative gains affected the angular-position response. Two gain conditions were tested in order to compare transient behaviour, overshoot, settling characteristics, and the remaining error at the end of the simulation window. The corresponding angular-position responses are shown in Figures 17 and 18, and the comparative metrics are summarised in Table 6.

These appendix results are intended to show the effect of gain adjustment at single-motor level and to support the later two-joint tuning presented in the main report. They should not be interpreted as final hardware validation, since the simulations used the transfer-function model in Section B.1 and simulated angular-position feedback rather than physical PWM actuation and measured encoder data.

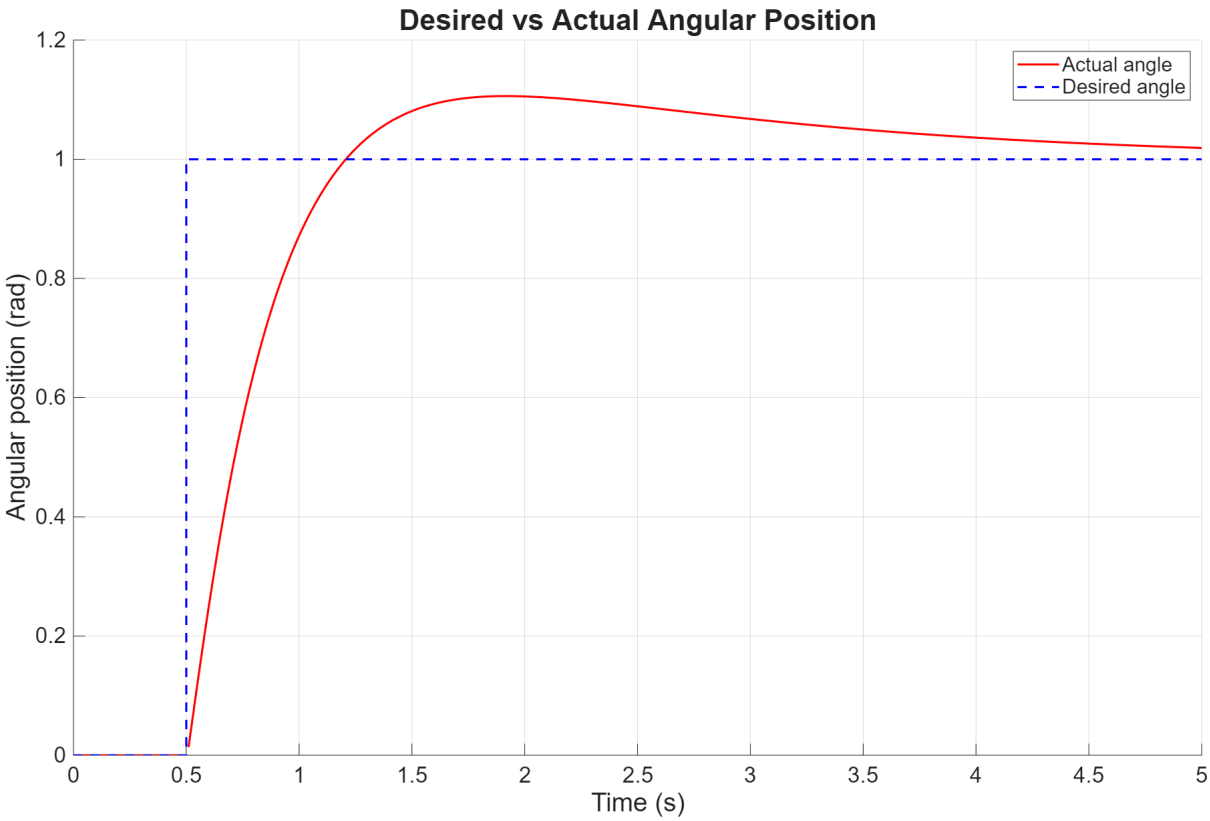


Figure 17: Condition 1

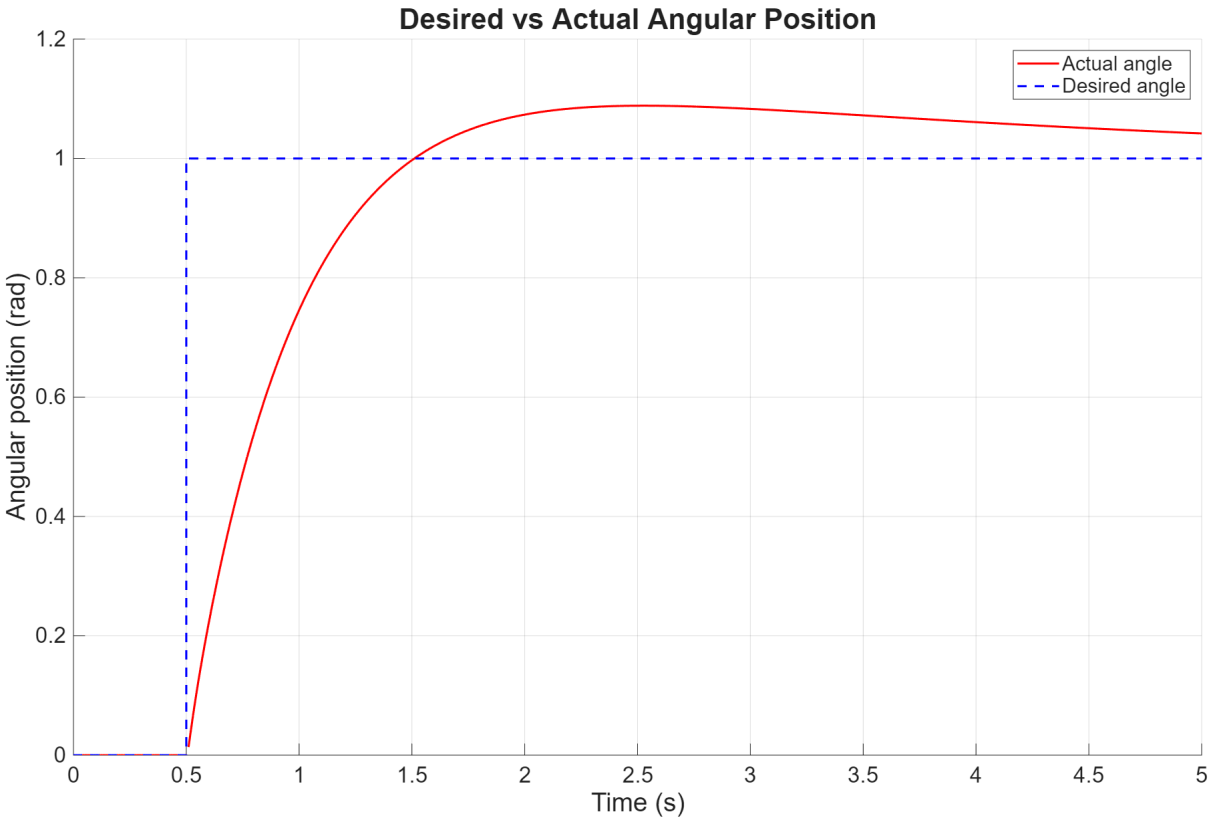


Figure 18: Condition 2

Table 6: Single-motor preliminary PID tuning comparison

Condition	$\mathbf{K_p}$	$\mathbf{K_i}$	$\mathbf{K_d}$	Rise Time (s)	Settling Time (s)	Overshoot (%)	Final Error at 5 s (rad)
1	2.0	1.0	0.2	0.4900	4.9300	10.60	0.018994
2	2.5	0.8	0.6	0.6900	*Not reached within 5 s*	8.84	0.041901

The final-error values reported in Table 6 are not directly comparable with the steady-state errors in Table 4 of the main report. Table 6 summarises preliminary single-motor tuning behaviour using the error remaining at the end of the 5 s simulation window, whereas Table 4 reports the final tuned two-joint results used in the main controller evaluation.

B.5 PID Gain Adjustment Summary

The two gain conditions were used as a preliminary tuning comparison rather than as final controller candidates. Condition 1 $(K_p = 2.0, K_i = 1.0, K_d = 0.2)$ produced a faster response and smaller final error within the 5 s simulation window, but with higher overshoot. Condition 2 $(K_p = 2.5, K_i = 0.8, K_d = 0.6)$ reduced overshoot slightly, but responded more slowly and did not settle within the available simulation duration.
This comparison demonstrates the expected trade-off between responsiveness and damping. Increasing proportional action tends to improve response speed, integral action reduces long-term residual error, and derivative action increases damping. However, no single gain combination is universally optimal, and the final tuning must be selected according to the required balance between overshoot, settling time, and tracking accuracy. For this reason, the appendix results were used as a preliminary tuning study, while the final two-joint tuned responses are reported in the main body of the report.

C. Electrical Schematic / System Diagram

The intended electrical system consisted of an Arduino Mega 2560, an L298N dual H-bridge motor driver, two EMG30 geared DC motors with encoders, an external DC supply, and common grounding between the logic and motor-power circuits. The Arduino would provide direction and PWM control signals to the motor driver, while the encoder channels would return angular-position feedback for each motor. The physical wiring photographs in Figure 20 show the practical implementation layout, including the Arduino, breadboard wiring, motor driver, encoder wiring, and motor connections.

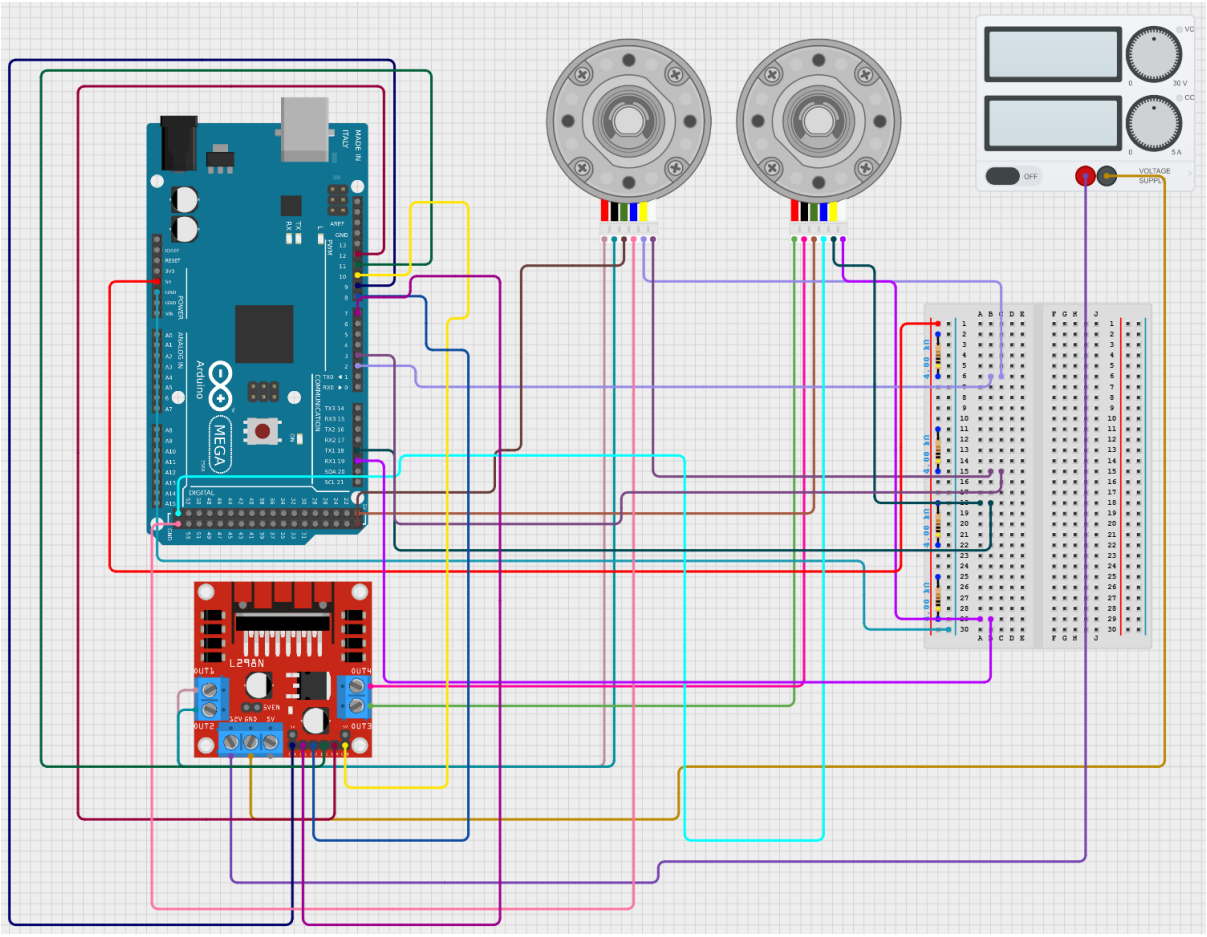
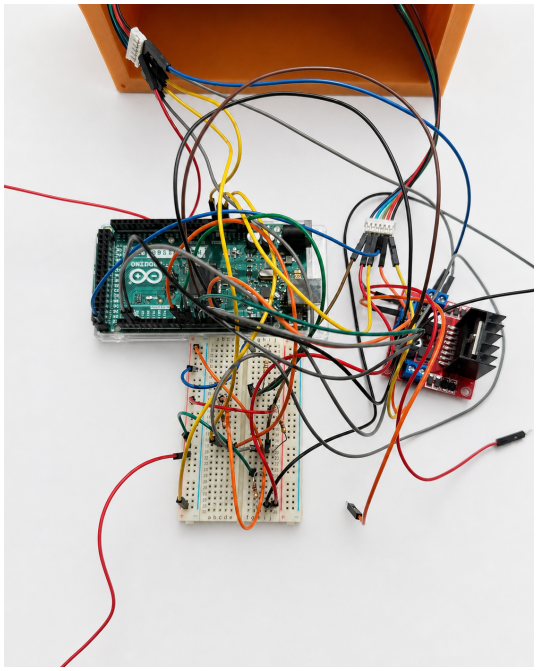
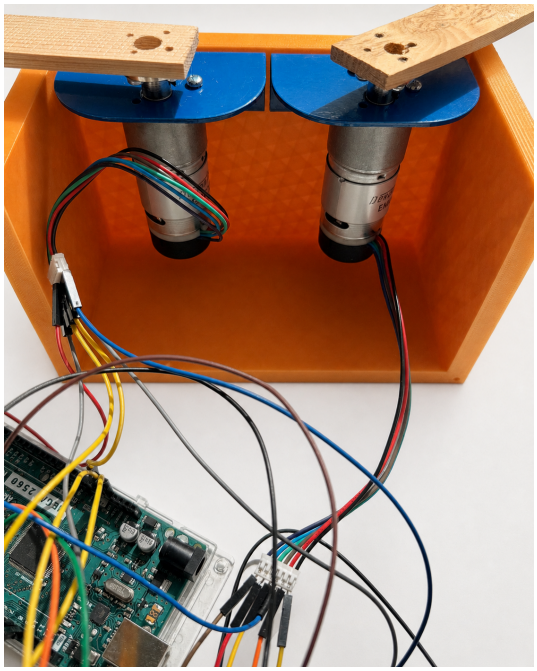


Figure 19: Conceptual wiring diagram showing the main electrical components and signal flow



(a) Physical wiring including Arduino Mega, Encoder and Breadboard



(b) Physical wiring including L298N DC Motor Driver

Figure 20: Physical wiring layout of the robot

D. Bill of Materials

The bill of materials summarises the main mechanical, electrical, electronic, and custom-manufactured components used in the prototype.

Table 7: Bill of materials

Item	Quantity	Unit Cost (£)	Total Cost (£)	Link
EMG30 geared motor with encoder	2	32.84	65.68	Product link
EMG30 mounting bracket	2	4.38	8.76	Product link
Power Supply cables	1 set	6.19	6.19	Product link
L298 motor driver	1	2.85	2.85	Product link
Arduino Mega 2560	1	45.57	45.57	Product link
Breadboard	1	1.99	1.99	Product link
5 mm bearings	7	0.25	1.75	Product link
Hook-up wire	2 m	1.60	3.20	Product link
3D printed parts set	1 set	12.65	12.65	
		Known subtotal	**148.64**	