SMALL ARM, BIG IMPACT: AL THE DRAWING ROBOTIC ARM

Andrea Walker Perez
Kings College London

ABSTRACT

This paper presents the design and development of a two-degree-of-freedom robotic arm tailored for precise applications within a 156 mm x 156 mm workspace. The arm demonstrates potential for use in medical scenarios such as fine motor surgical tasks and precision prototyping. The aim is to create an affordable, compact system integrating efficient control strategies. The robot integrates micro servo motors and a custom CAD design, controlled by an Arduino Uno and L298N motor controller. Forward and inverse kinematics were derived for accurate motion planning, and PID tuning was employed to optimise the stability and accuracy of the system. Experiments were conducted to evaluate the arm's performance under varied loads, PID settings, and predefined trajectories. The robotic arm achieved an average deviation of less than 2 mm in accuracy tests and exhibited reliable repeatability under varying conditions. Experimental results demonstrate a clear trade-off between speed and precision, with optimal PID gains minimising oscillations and ensuring stability.

This robotic system provides a foundation for low-cost, high-precision applications in robotics. Future enhancements will address system limitations such as oscillations under high loads and explore dynamic control strategies to expand its operational capabilities.

1. INTRODUCTION

Robotics in precision applications, particularly in medical fields, is a fast-evolving domain. With increasing demand for accuracy, affordability, and adaptability, two critical questions emerge:

1. Can robotic systems replicate the fine motor skills required in precision tasks such as surgeries?
2. How can low-cost robotic arms maintain high precision and stability under varying conditions?

Advancements in robotics have proven their ability to perform intricate tasks traditionally reserved for humans. The Da Vinci Surgical System has revolutionised surgery by enabling minimally invasive procedures with millimetre-scale accuracy [1]. Low-cost solutions incorporating modern control techniques, such as PID and kinematic modelling, make robotic precision more accessible.

While promising, affordability often comes at the cost of robustness and precision. Unlike high-end systems, low-cost robotic arms frequently struggle with stability under dynamic conditions and precision in repeatability. Additionally, integrating compact designs with high-performance electronics and motors remains challenging.

The integration of efficient control strategies with modular hardware designs offers a pathway to making robotic precision more accessible. By addressing stability and precision trade-offs, low-cost robots can extend their applications beyond prototyping into fields like medical assistance and microfabrication.

1.1. Aim and Objectives

The aim was to design, build, and evaluate a two-degree-of-freedom robotic arm capable of precise drawing operation within a defined workspace.

Specific objectives include:

- Develop a modular CAD design integrating micro servos, thrust bearings, and a secure pen holder.
- Implement forward and inverse kinematics for accurate motion planning.
- Optimise PID control to balance speed, precision, and stability.
- Evaluate performance through experiments focusing on accuracy, speed, and load-bearing capacity.

2. METHODOLOGY

This section details the design, development, and testing methodologies for the robotic system designed for medical applications. The focus was on achieving high precision, stability, and a wide range of motion. The system combines CAD design, hardware integration, kinematic analysis, control strategies, and experimental validation.

2.1 CAD Design and 3D Printing

The robotic arm was designed as a parallel SCARA configuration with an extension holding the end effector, tailored for precision and compactness. The CAD design comprises three main components: the base, the arm, and the protective housing. The arm features four links: two vertically stacked links directly connected to motors, maximising the range of motion and compactness, and two additional links extending the reach of the end effector. The parallel arrangement optimises workspace usage while maintaining high accuracy and mechanical robustness.

The base provides a robust foundation, securely housing the motors and control electronics to ensure consistent performance during operation. To enhance the grounding of the arms, the motor-connected links were equipped with ratcheting gears, allowing manual adjustments without affecting the motor's position.

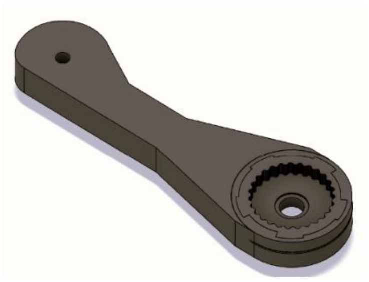


Figure 1 Ratcheting Gear Design

One motor-driven link is connected to an intermediate one, supporting the pen-holding assembly. In contrast, the final link bridges the end-effector and the second motor-driven link, completing the parallel SCARA-like mechanism. Figure 1 illustrates the arm's configuration.

The protective housing was designed to enclose the Arduino Uno and motor drivers, offering environmental protection while ensuring accessibility for maintenance. Features like a sliding door allow for easy access to electronics without compromising the overall design's integrity.

The design underwent iterative refinement using Fusion 360, where simulations validated key aspects such as mechanical strength, range of motion, and component interactions. After finalising each design, the CAD models were exported in the .3mf file format, which preserves intricate geometries and ensures compatibility with the 3D printing process.

3D printing was performed using Bamboo Studio, where the .3mf files were imported for slicing and G-code generation. The printing parameters were selected to balance dimensional accuracy and structural integrity. A layer height of 0.2 mm was used to achieve a smooth surface finish, while an infill density of 40% provided the necessary strength and reduced material usage. PLA was chosen as the primary material due to its favourable mechanical properties, lightweight nature, and ease of use. Supports were strategically applied to maintain precision during printing. Preliminary test prints were conducted to evaluate the dimensional accuracy of the 3D printed components compared to the CAD designs on Fusion 360. These tests ensure that the printed dimensions closely matched the intended specifications and allowed for fine-tuning of the design to account for such discrepancies, ensuring that critical dimensions met the necessary tolerances for proper assembly and functionality.

2.2 Hardware Integration

The robotic system's hardware was designed with precision and compactness in mind. At its core, the system comprises two micro servo motors controlled by an Arduino Uno, which interfaces with an L298N motor driver. This integration was essential for ensuring precise motion control of the robotic arm. The Arduino Uno serves as the central control unit, executing commands and managing the motor signals. It connects directly to the L298N motor driver through its digital pins. These connections include ENA, ENB, IN1, IN2, IN3, and IN4, which transmit the necessary Pulse Width Modulation (PWM) and directional signals for the motors. The motors themselves are powered via the OUT1/OUT2 and OUT3/OUT4 outputs of the L298N, ensuring smooth operation. The L298N receives power from an external 12V supply, with its GND connected to both the Arduino and the supply for common ground.

To enhance stability and efficiency, thrust bearings were used in the joints directly connected to the motors. These bearings handle the axial loads effectively, reducing wear and tear while transmitting torque reliably to the robotic arm. For the other joints, ball bearings were chosen for their ability to minimise friction and ensure smooth rotational motion, crucial for maintaining precision during operation.

Wiring was carefully organised and routed through protective housing to minimise clutter and facilitate troubleshooting. Cable ties were used to secure connections, and wires were routed away from moving parts to prevent damage during operation. The servos were powered by an external 5V power supply, ensuring sufficient torque and consistent operation without overloading the Arduino's onboard regulator.

A simplified schematic wiring diagram is provided in Figure 2, illustrating the connections between the components. The diagram highlights the integration of the Arduino, L298N motor driver, motors, and the external power supply, with clear labelling of all pins and connections. This setup ensures reliability and ease of replication.

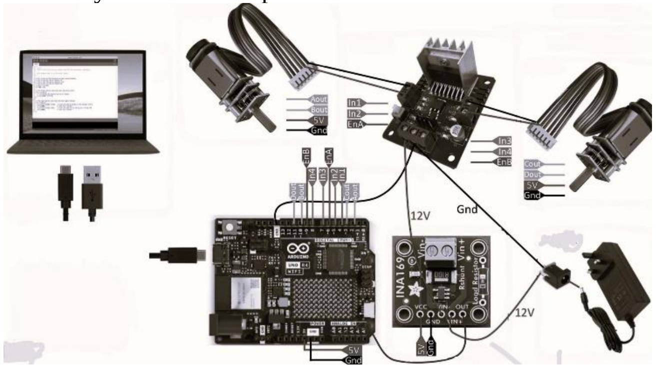


Figure 2 Wiring Diagram

2.3 Kinematics

2.3.1. Forward Kinematics

Forward kinematics was used to calculate the end-effector's position in the 2D workspace based on the joint angles (θ_1 and θ_2) and the lengths of the robot arm's links directly connected to the motors (l_1 and l_4).

The end-effector's coordinates (x, y) were computed using the following equations:

$$x = l_1 cos\theta_1 + l_4 cos\theta_4 + l_{ex} cos\theta_{ex}$$
$$= l_2 cos\theta_2 + l_3 cos\theta_3 + l_{ex} cos\theta_{ex}$$
$$y = l_1 sin\theta_1 + l_4 sin\theta_4 + l_{ex} cos\theta_{ex}$$
$$= l_2 sin\theta_2 + l_3 sin\theta_3 + l_{ex} sin\theta_{ex}$$

Here, l_1 and l_4 represent the two links extending from the motors, while l_2, l_3, and l_{ex} are the links connecting the dependent joints to the end-effector. The angles θ_1 and θ_4 are measured relative to the horizontal baseline and serve as the primary independent joint angles. The dependent angles, θ_2,

θ_3, and θ_{ex}, are proportional to the main angles, with $\theta_2 = \theta_{ex} = \theta_4$ and $\theta_3 = \theta_1$.

The workspace of the robotic arm was analysed using forward kinematics, considering the lengths of the links and the joint angle limits. As shown in Figure 3, the theoretical workspace, represented by the dotted curve, highlights all reachable positions. Within this theoretical workspace lies the operational workspace, a 156 mm x 156 mm square area selected for precision tasks. The operational workspace has an offset of approximately 110 mm from the centre of the robot, corresponding to the length of l_1, which creates a gap between the centre and the reachable area.

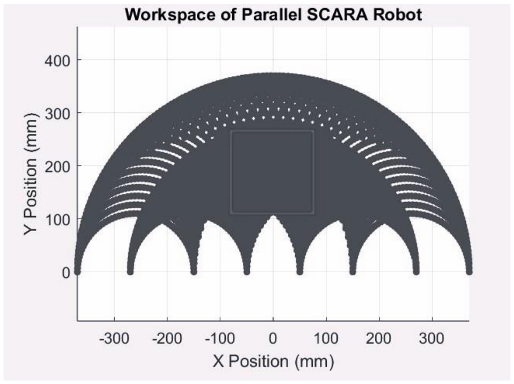


Figure 3 Theoretical and Operational Workspaces of the Robotic Arm

The derivation of the forward kinematics equations and methodology for determining the workspace were guided by principles discussed in the work of Le et al. [2], which provide a foundational approach to analysing planar parallel manipulators.

2.3.2. Analytical Inverse Kinematics

The analytical inverse kinematics for the robotic arm were derived to calculate the required joint angles (θ_1 and θ_4) based on the derived end-effector position (x,y) within the 2D workspace. This method relies on trigonometric relationships and the geometry of the parallel SCARA configuration, ensuring high precision in computing the joint parameters.

The inverse kinematics equations were derived using the lengths of the robot's links and the constraints imposed by the arm's geometry. These equations ensure that each desired position is mapped to a unique set of joint angles. The process involves calculating the distance from the origin to the intersection between both paths:

$$r = \sqrt{x^2 + y^2} \tag{1}$$

Then, using the cosine rule, the intermediary angles are derived:

$$A = \cos^{-1}\left(\frac{l_1^2 + r^2 - l_4^2}{2l_1 r}\right) \tag{2}$$

$$B = \cos^{-1}\left(\frac{l_1^2 + l_4^2 + r^2}{2l_1 l_4}\right) \tag{3}$$

$$E = 180 - A - D \tag{4}$$

$$F = B - E \tag{5}$$

To account for the offset:

$$w = (l_5 - l_4)\cos(F) \tag{6}$$

$$z = (l_5 - l_4)\sin(F) \tag{7}$$

The corrected position of the end-effector point is:

$$x_2 = x - w \tag{8}$$

$$y_2 = y - z \tag{9}$$

The angle D represents the orientation of the target position relative to the origin. To ensure the robot operates within its physical constraints and avoids unsafe configurations, the following conditions are applied:

- For

$$(x \geq 0, y \geq 0): D = \tan^{-1}(y/x) \tag{10}$$

- For

$$(x < 0, y \geq 0): D = 180 - \tan^{-1}(y/-x) \tag{11}$$

- For

$$(x \geq 0, y < 0): D = \tan^{-1}(y/x) \tag{12}$$

- For $(x < 0, y < 0)$: This configuration is unreachable and should return and error.

Limiting D ensured that the computed angles remain within the robot's workspace and prevents it from attempting to move to positions that could damage the hardware.

Finally, the angles G and H, required to resolve the kinematics of the arm, are calculated using the cosine rule as follows:

$$G = \cos^{-1}\left(\frac{r^2 + r_2^2 - (l_5 - l_4)^2}{2rr_2}\right) \tag{13}$$

$$H = \cos^{-1}\left(\frac{l_2^2 + r_2^2 - l_1^2}{2l_2 r_2}\right) \tag{14}$$

Finally, the joint angles are defined as follows:

$\theta_1 = D + A$ (15) $\theta_4 = (D + G) - H$ (16)

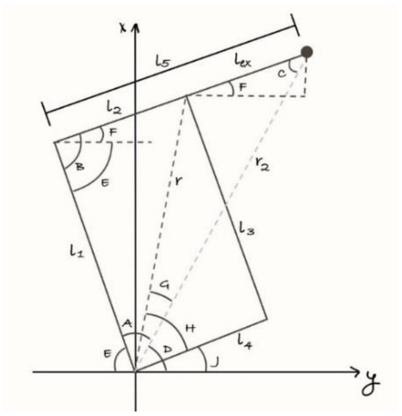


Figure 4 Analytical Inverse Kinematic Diagram

The derived equations account for the dependency of the secondary joint angles on the primary joint angles as mentioned in section 2.3.1. This dependency ensures that the arm operates cohesively, maintaining its parallel configuration. The final equations were validated through MATLAB simulations, where test cases confirmed that the computed angles accurately positioned the end-effector within the desired workspace. In section 3 you can see the GitHub repository link with all the code implementing this.

2.3.3. Differential Inverse Kinematics and Manipulability

The robotic arm's motion and manipulability were analysed using differential inverse kinematics methods, specifically the Jacobian transpose method and the damped least squares (DLS) method. These approaches were implemented to ensure accurate tracking of the desired end-effector trajectories while maintaining stability and avoiding singularities within the workspace.

The Jacobian matrix, derived as a function of the joint angles and link lengths, provides a relationship between the joint velocities and the end-effector velocity in Cartesian space [3]. Its form is given as:

$$J = \begin{bmatrix} -r_1 \sin(\theta_1) & -r_2 \sin(\theta_2) \\ r_1 \cos(\theta_1) & -r_2 \sin(\theta_2) \end{bmatrix} \tag{17}$$

, where r_1 and r_2 represent the respective link lengths. This matrix not only facilitates the computation of joint velocities but also provides insight into the manipulability of the robotic arm at various configurations. The manipulability at a specific configuration is defined as the square root of the determinant of JJ^T, which quantifies the arm's ability to perform precise motions in multiple directions. Higher manipulability values correspond to greater ability and flexibility in motion.

The DLS method computed joint velocity corrections, enhancing numerical stability near singularities [3]. This method modifies the pseudo-inverse of the Jacobian by introducing a damping term, ensuring that the control law remains stable even in configurations where the Jacobian matrix approaches singularity [3]. The control law is defined as:

$$\Delta q = (J^T J + \lambda^2 I)^{-1} J^T e \tag{18}$$

, where Δq represents the joint velocity corrections, e is the error between the desired and current end-effector positions, λ is the damping factor, and I is the identity matrix. The damping factor was tuned experimentally to balance responsiveness and stability, ensuring smooth trajectory tracking.

As an alternative, the Jacobian transpose method was used. This approach simplifies computation by avoiding matrix inversion, making it computationally efficient but less robust near singularities [3]. The control law for the Jacobian transpose method is expressed as:

$$\Delta q = kJ^T e \tag{19}$$

, where k is the gain factor. While Jacobian transpose is advantageous for its simplicity, it was primarily used for configurations away from singularities, where robustness was less critical.

2.4 PID Control

PID control was implemented to ensure smooth, precise, and responsive positioning of the servo motors in the robotic arm. The control law is expressed as

$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt} + K_i \int e \, dt + \tag{20}$$

, where e(t) is the error between the desired and actual position of the motor [4]. K_p, K_i, and K_d are the proportional, integral, and derivative gains. Graph 1 provides theoretical damping scenarios that may arise when tuning the PID gains and serves as a guide to understanding the influence of each. In an underdamped response, the system is sensitive to errors and tends to overshoot the setpoint with oscillations before eventually settling. This scenario typically arises when the proportional gain is too high, resulting in a system that reacts aggressively to deviations. In contrast, an overdamped response is characterised by a slow and conservative approach to the setpoint, with no overshoot. This behaviour often results from overly cautious tuning, such as a low K_p or an excessively high K_d. A critically damped response lies at the boundary between these two extremes, where the system reaches the setpoint without oscillating. However, it is often too slow for dynamic systems that require both speed and precision.

The dashed curve on the graph represents an optimally tuned PID system, which achieves a slightly underdamped response. This provides a balanced trade-off between responsiveness and stability, allowing the system to reach the setpoint quickly with minimal overshoot and without significant oscillations. This is typically the desired behaviour for robotic systems performing precision tasks such as drawing.

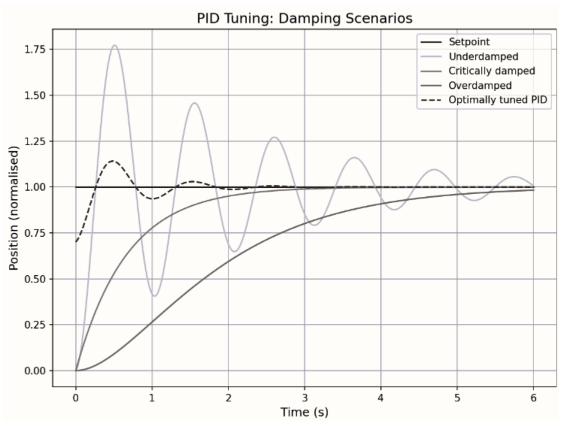
Each component of the PID controller contributes uniquely to the system's behaviour. The proportional term reacts proportionally to the current error, providing immediate corrective action. While increasing Kp improves responsiveness, it also risks overshoot and instability if set too high. The integral term addresses steady-state errors by considering the accumulation of past errors, correcting persistent offsets. However, excessive Ki can introduce oscillations due to overcompensation. Finally, the derivative term predicts future errors by responding to the rate of change of the error, effectively dampening oscillations and smoothing the system's approach to the setpoint.

Using this theoretical framework as a guide, the PID tuning process was conducted systematically. The process began with tuning Kp, while Ki and Kd were initially set to zero. Kp was incrementally varies within range of 1 t. Once a satisfactory value was determined, Ki was introduced between 0 and 20. Finally, Kd was incorporated to refine the transient response within a range of 0 to 20.

This structured approach ensured that each gain was tuned in isolation before combining them to achieve a balanced and effective PID control.

The PID tuning process aimed to optimise the PID parameters to balance speed and accuracy in the robotic arm's movements. The process began with a focus on the K_p, which varied incrementally from 1 to 50 whilst keeping $K_i = 0$ and $K_d = 0$.

The integral gain (Ki) was introduced to mitigate overshoot and address steady-state errors, ranging from 0 to 20. Then kd ranging from 0 to 20.



Graph 1 Damping Scenarios during PID Tuning (adapted from [5])

2.5 High-Level Control and GUI Implementation

The high-level control of the robotic system involves a structured workflow that bridges the gap between user commands and precise robotic movements. Figure 5 shows this workflow, which includes command input via a graphical user interface (GUI), real-time data processing, and the computation of joint angles for robot motion using inverse kinematics.

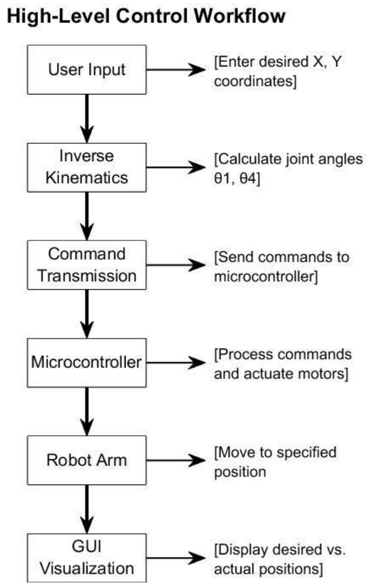

High-Level Control Workflow
User Input
[Enter desired X, Y coordinates]
Inverse Kinematics
[Calculate joint angles θ1, θ4]
Command Transmission
[Send commands to microcontroller]
Microcontroller
[Process commands and actuate motors]
Robot Arm
[Move to specified position
GUI Visualization
[Display desired vs. actual positions]

Figure 5 High-Level Control Workflow of the Robotic System

The GUI, developed in MATLAB, allows users to input desired Cartesian coordinates (X, Y) for the robot's end-effector position. These inputs are processed to compute the required joint angles (θ_1, θ_4) using the analytical inverse kinematics method. Computed angles are then formatted into commands and transmitted to the microcontroller via serial communication. On receiving commands, the microcontroller allows the robot to move to the specified position. Simultaneously, the robot's actual position, derived using forward kinematics, is sent back to the GUI for real-time plotting and comparison with the desired position. The GUI interface features input fields for X and Y coordinates, a send button for command transmission, and a dynamically updating plot area for visualising the robot's motion, as shown in Figure 6. Additionally, a real-time data stream from the microcontroller updates the end-effector position at a fixed rate, enabling continuous feedback during operation.

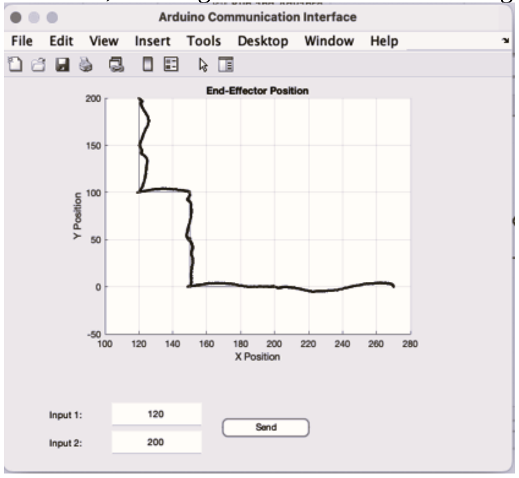

Arduino Communication Interface
File Edit View Insert Tools Desktop Window Help
End-Effector Position
Y Position
X Position
Input 1:
120
Input 2:
200
Send

Figure 6 GUI Interface

The GUI was critical in supporting the experiments described in section 2.6. By enabling precise position control and real-time monitoring, it facilitated the execution of the experiments. The MATLAB code for the GUI implementation is included in the GitHub repository linked at the end of section 3.

2.6 Experimental Design

Two experiments were designed to evaluate the robot's performance.

2.6.1. Accuracy Test

The first experiment was conducted to evaluate the robotic arm's precision in reaching predefined target positions within its workspace. Eight target positions were selected strategically to cover various regions of the workspace, including central locations, edges, and corners. The target positions were defined by their coordinates (X desired, Y desired) and were chosen to ensure a comprehensive assessment of the arm's ability to accurately traverse its full range of motion.

The robotic arm was programmed to move to each target position using the inverse kinematics model developed in section 2.3.2. Upon reaching a target position, the arm's actual coordinates (X actual, Y actual) were measured manually using a digital calliper. These measurements were taken at the end-effector, ensuring high precision in capturing the arm's true location. The deviation between the desired and actual coordinates was recorded for each position.

To assess repeatability and consistency, the experiment was repeated four times for each target position. Between trials, the arm was reset to a neutral home position to ensure the results reflected the arm's ability to accurately reach the target rather than any systematic drift or accumulated error. The data from these tests were used to calculate the positional error at each target, as well as the mean and standard deviation of the errors across all trials.

2.6.2. Speed vs. Accuracy Test

The second experiment was designed to investigate the trade-off between movement speed and positional accuracy. This test aimed to evaluate how varying the PID controller's gains affects the arm's ability to move quickly while maintaining precision. By tuning the proportional, integral, and derivative gains, the speed of the arm's movements could be increased or decreased, influencing its ability to accurately reach a target. For this test, a subset of the target positions used in the accuracy test was selected, focusing on positions that represented both central and edge locations within the workspace. The PID gains were incrementally adjusted to achieve three different speed settings: low speed (high damping, conservative motion), medium speed (balanced damping and responsiveness), and high speed (low damping, aggressive motion). Each speed setting represented a unique combination of Kp, Ki, and Kd, as determined during the tuning process described in section 2.4. At each speed setting, the arm was programmed to move to the predefined target positions, and its actual coordinates (X actual, Y actual) were recorded as in the accuracy test. The positional error was calculated for each target and analysed to identify any trends between movement speed and accuracy. The experiment was repeated three times for each speed setting to ensure the results were consistent and reliable. In addition to recording positional errors, the time taken to move from the home position to each target was measured using a stopwatch. This allowed for a direct comparison of movement speed and accuracy. This experiment aimed to quantify the relationship between speed and accuracy, highlighting the influence of PID tuning on the robotic arm's performance.

3. RESULTS

3.1 Final CAD Design, 3D Printing, and Wiring

The robotic arm was successfully designed, fabricated, and assembled, meeting the specified requirements for precision, functionality, and ease of integration. The CAD model, shown in Figure 8, was developed using Fusion 360 and features a compact and modular design that optimises workspace use while ensuring structural stability. The arm incorporates a 2DOF mechanism, enabling precise movements within the 156 mm x 156 mm workspace. Integrated cable routing channels and secure mounts for the micro servo motors were included to minimise wire entanglements and improve reliability. Additionally, the design emphasised modularity, allowing individual components to be replaced or upgraded with minimal effort.

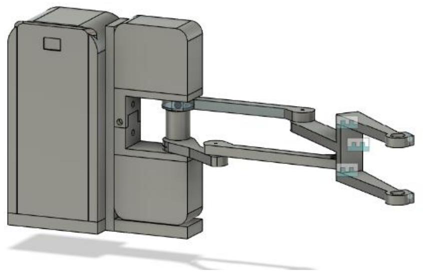


Figure 7 Final CAD Design in Fusion 360

Once the CAD design was finalised, the robotic arm components were fabricated using 3D printing. The printed model is displayed in Figure 9. The components were printed using PLA, selected for their balance between lightweight properties and sufficient structural strength. The 3D-printed parts demonstrated excellent dimensional accuracy, ensuring proper alignment and fit during assembly. No post-processing was required, as the tolerances were adhered to during the design phase.

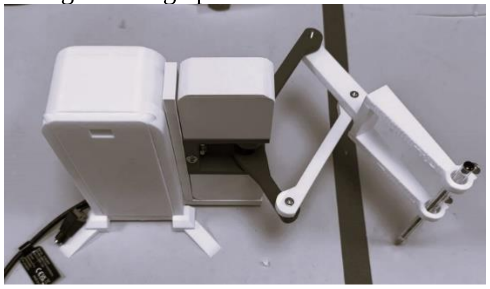


Figure 8 Final 3D Printed Robotic Arm

The hardware and wiring were organised within the base of the robotic arm, as shown in Figure 10. The design of the base provided sufficient space for all control electronics, including the Arduino Uno, L298N motor controller, and additional components. Wiring was routed through the channels incorporated into the CAD model, ensuring a tidy and secure layout. This approach reduced clutter and minimised the risk of disconnections or signal interference during operation. Cooling considerations were also considered, with the arrangement allowing sufficient airflow around the components.

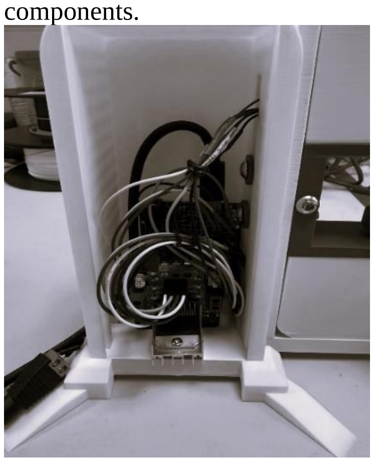
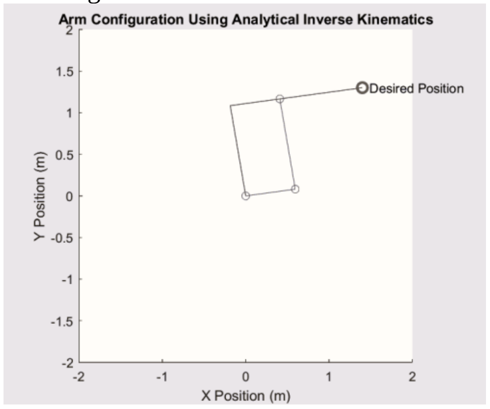


Figure 9 Final Wiring

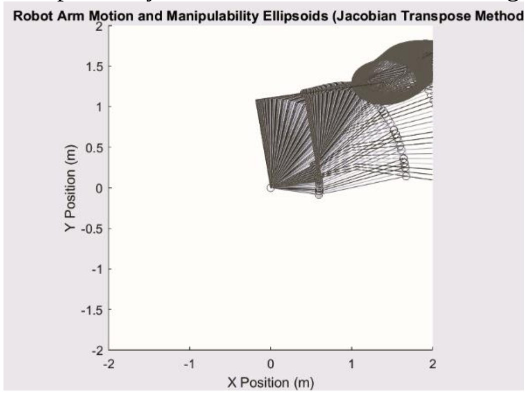
3.2 Analytical vs Differential Inverse Kinematics

The performance of the robotic arm was evaluated using three methods for computing joint angles and trajectory tracking: analytical inverse kinematics, the Jacobian Transpose method, and the Damped Least Squares (DLS) method. Each method was assessed based on its ability to compute joint angles, track desired trajectories, and maintain manipulability. The analytical inverse kinematics method provided direct computation of joint angles using geometric and trigonometric relationships. The results, illustrated in Graph 2, demonstrate the arm configuration aligning precisely with the desired position within the workspace. This method generated exact solutions for all tested target positions, provided they were within the workspace and not near singularities.

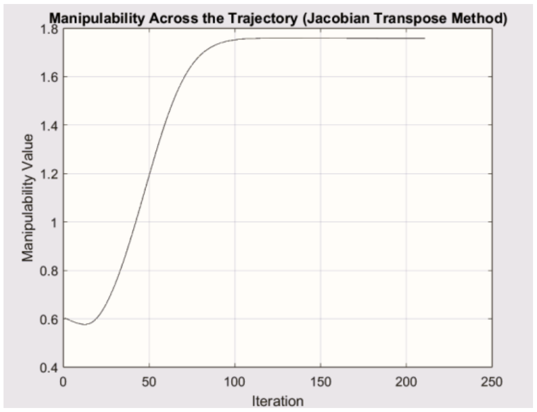


Graph 2 Analytical Inverse Kinematics Output

The Jacobian Transpose method uses an iterative approach to compute joint angle corrections by multiplying the Jacobian matrix transpose with the positional error. Graph 3 depicts the robot arm’s motion, and the manipulability of ellipsoids generated during trajectory tracking using this method. The manipulability values computed along the trajectory are shown in Graph 4, which highlights changes in manipulability as the arm crosses various configurations.

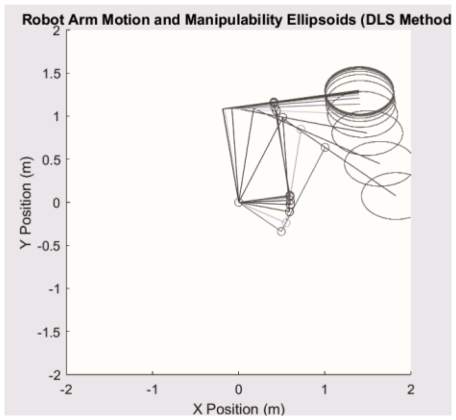


Graph 3 Jacobian Transpose Output

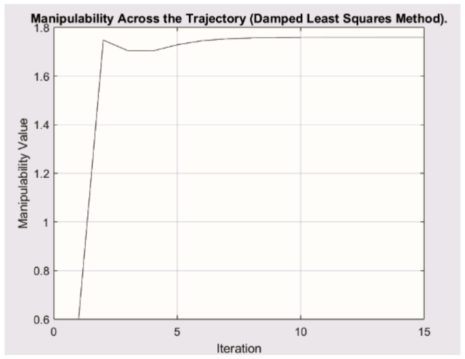


Graph 4 Manipulability when using Jacobian Transpose

The Damped Least Squared (DLS) method incorporated a damping factor to stabilise the pseudo-inverse of the Jacobian matrix, allowing it to handle singularities more effectively. The motion and manipulability of ellipsoids produced using this method are presented in Graph 5, showing smoother and more consistent ellipsoids compared to the Jacobian Transpose method. Graph 6 illustrates the manipulability values along the trajectory when using the DLS method, which remained stable across all configurations tested.



Graph 5 DLS method Output



Graph 6 Manipulability when using DLS method

The MATLAB code for implementing these methods is included in the GitHub repository link at the end of this section.

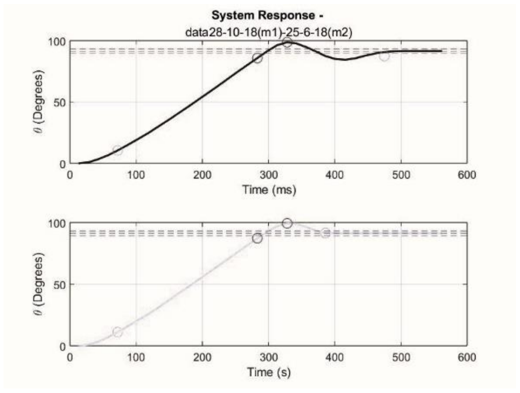
3.3. PID Tuning

The results of the PID tuning tests for the robotic arm are summarised in the table below. The table includes the system response metrics for different combinations of Kp, Ki, and Kd gains, tested on each motor. Key performance metrics measured include the steady-state value, steady-state error, rise time, peak time, peak overshoot, and settling time.

Table 1 PID Tuning Results

P	I	D	Steady-State Value (°)	Steady-State Error (°)	Rise Time(s)	Peak Time(s)	Peak Overshoot (%)	Settling Time (s)
5	0	0	89.82	0.18	221	366	0	337
15	0	0	91.84	-1.84	196	329	9.65	389
28	8	18	89.22	0.78	210	328	10.60	552
28	10	18	91.43	-1.43	211	328	7.93	475
30	0	0	91.84	-1.84	211	327	6.13	460
30	15	18	88.01	1.99	196	329	10.07	374
40	10	10	89.22	0.78	212	330	10.38	508
40	20	10	89.22	0.78	209	327	8.80	504
50	0	0	90.63	-0.63	208	311	6.44	808

The corresponding system response graphs for each tuning configuration are presented in Appendix A in Graphs 8-14, showing the angular position versus time for both motors under each set of gains. Optimal PID gains are 28-10-18 for motor 1 and 25-6-18 for motor 2 shown visually in Graph 8.



Graph 7 PID Response for Motor 1 and Motor 2 with PID Gains 28-10-18 and 25-6-18 respectively

3.2 Accuracy Test

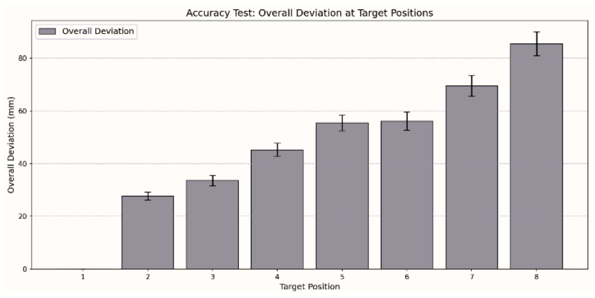
Table 2 contains the desired and actual positions when sent to multiple points for one of the iterations. It also contains the deviation in each axis and the overall deviation. Graph 9 shows visually all desired and actual positions in that same iteration.

Graph 8 is a bar graph summarising the average deviation for each position.

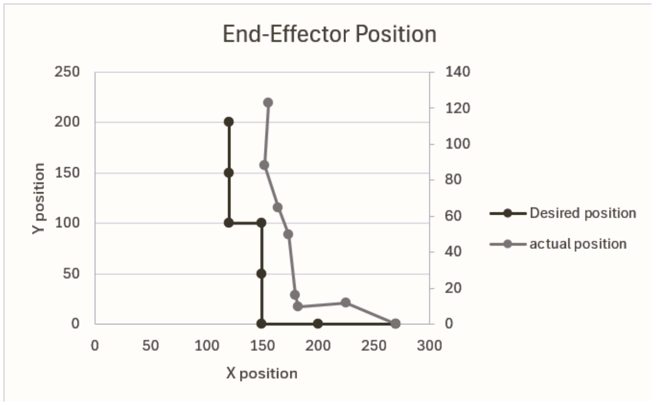
The average deviation across all trials was calculated as 4.5 mm in the X-axis and 6.7 mm in the Y-axis.

Table 2 Actual and Desired Positions from the robot when going to different points for the accuracy test

Desired X (mm)	Desired Y (mm)	Actual X (mm)	Actual Y (mm)	Deviation in X (mm)	Deviation in Y (mm)	Overall Deviation (mm)
270	0	270	0	0.00	0.00	0.00
200	0	225	12	25.00	12.00	27.73
150	0	182	10	32.00	10.00	33.54
150	50	180	16	30.00	34.00	45.26
150	100	174	50	24.00	50.00	55.43
120	100	164	65	44.00	35.00	56.04
120	150	152	88	32.00	62.00	69.46
120	200	156	123	36.00	77.00	85.34



Graph 8 Overall Deviation of Target Positions from the Accuracy Test



Graph 9 Desired and Actual Position of the Robot in one iteration when going to different points

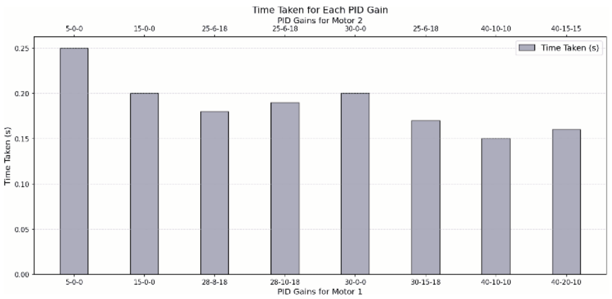
3.3 Speed vs. Accuracy Test

The speed vs. accuracy test was conducted to evaluate the robotic arm's performance under different PID gain configurations. The results were analysed in terms of the time taken to reach each target position and the overall deviation from the desired coordinates. The data for all tests of PID gain configurations is summarised in Table 4, which presents the actual end-effector coordinates, the time taken to reach the target, and the overall deviation.

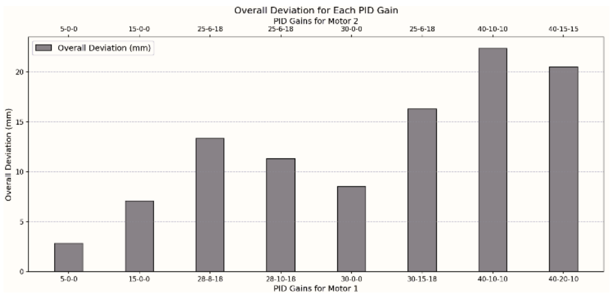
Table 3 PID Gain Results for the Speed vs Accuracy Test

P	I	D	Actual X (mm)	Actual Y (mm)	Time Taken (s)	Overall Deviation (mm)
5	0	0	198	148	0.25	2.83
15	0	0	195	145	0.20	7.07
28 28	8 6	18 18	188	142	0.18	13.34
28 28	10 6	18 18	193	146	0.19	11.31
30	0	0	192	145	0.20	8.49
30 25	15 6	18 18	185	140	0.17	16.28
40	10	10	180	138	0.15	22.36
40 40	20 15	10 15	182	139	0.16	20.52
50	0	0	175	128	0.2	25

Lower proportional gain (Kp = 5) resulted in a minimal overall deviation of 2.83 mm but required the longest time of 0.25 seconds to reach the target. As Kp increased, the system responded more quickly, with a minimum time of 0.15 seconds recorded for Kp = 40 (for both motors). However, higher gains also resulted in increased deviations, with the largest overall deviation of 25 mm occurring for Kp = 50. The time taken for each PID gain is illustrated in Graph 10, showing a general trend of decreasing time as the proportional gain increases. The overall deviation for each PID gain is displayed in Graph 11, which highlights a trade-off between speed and precision, as higher gains led to larger deviations.



Graph 10 Time Taken to settle for each PID gain



Graph 11 Overall Deviation for each PID gain

GitHub Link: https://github.com/leastetie/Team-5

4. DISCUSSION

4.1 Kinematics

The kinematic methods explored included analytical inverse kinematics, the Jacobian Transpose, and the DLS method, each offering distinct advantages and limitations.

The analytical inverse kinematics method provided precise joint angle computations for all target positions within the robot's workspace, using geometric relationships and trigonometric functions. Its direct, non-iterative approach ensures fast and accurate solutions, making it well-suited for fixed-geometry tasks like drawing. However, it is limited by workspace constraints and becomes unreliable near singularities.

The Jacobian Transpose method, while computationally efficient and adaptable for real-time applications, exhibited reduced performance near singular configurations due to diminished manipulability. This issue aligns with findings in the literature, which highlight the Jacobian Transpose method's sensitivity to configurations with low manipulability [2].

The DLS method addressed these limitations by introducing a damping factor, stabilising solutions near singularities and ensuring smoother and more consistent manipulability across trajectories [2]. However, both iterative methods required greater computational effort than the analytical approach. Given the precision and efficiency required for a drawing arm robot, the analytical inverse kinematics method was selected. Its ability to compute joint angles directly without iterative adjustments ensures rapid and accurate path generations, which is ideal for well-defined workspace and static constraints of this application

4.2 PID Tuning

The PID tuning process focused on optimising Kp, Ki, and Kd to achieve a balance between response speed, stability, and overshoot. Our findings indicate that lower proportional gains (Kp) resulted in minimal steady-state errors but led to prolonged rise and settling times. Conversely, higher Kp values enhanced response speed but introduced significant overshoot. Adjusting the integral gain (Ki) influenced the system's ability to eliminate steady-state errors. Lower Ki values resulted in slower correction of persistent errors, whereas higher Ki values increased the likelihood of oscillations and instability. Similarly, the derivative gain (Kd) played a critical role in damping oscillations. Lower Kd values allowed overshoot to persist, while excessively high Kd values could cause the system to overreact to changes, leading to excessive damping and slower responses. The optimal balance was achieved with Kp = 28, Ki = 10, Kd = 18 for motor 1 and Kp = 25, Ki = 6, Kd = 18 for motor 2, with a rise time of 211 s, minimal peak overshoot of 7.93%, and a steady-state error of -1.43°. Similar observations are reported by [4], who emphasised the sensitivity of PID parameters in achieving desired performance in robotic manipulators. These were further validated in subsequent speed vs. accuracy tests.

4.3 Accuracy Test

The accuracy test assessed the robotic arm's precision in reaching designated positions within its operational workspace. Results demonstrated high accuracy near the workspace centre, with deviations as low as 2.83 mm. However, as the arm extended towards the periphery, deviations increased, reaching up to 85.34 mm. This trend is consistent with findings in robotic systems, where positional accuracy often diminishes at the extremities of the workspace due to factors like mechanical deflection and joint compliance.

4.4 Speed vs. Accuracy Test

The trade-off between speed and accuracy was evident in our experiments. Lower Kp values resulted in minimal positional deviations but required longer times to reach targets. In contrast, higher Kp values reduced response times but compromised accuracy, with deviations increasing significantly. This inverse relationship underscores the necessity of balancing speed and precision in robotic applications. The phenomenon aligns with the speed-accuracy trade-off observed in various robotic tasks, where increasing speed often leads to decreased accuracy.

4.1. Risk Assessment

The risk assessment of the robotic system was conducted using a Failure Mode and Effects Analysis (FMEA) approach, which systematically identified potential failure modes, hazardous situations, and critical mitigation strategies. This process ensured that risks across the mechanical, electrical, and software subsystems were evaluated and addressed to enhance the safety and reliability of the system. The risk assessment methodology included scoring each failure mode based on its likelihood, severity, and probability of leading to harm. High-risk items such as large unintended motion and H-bridge failures, received the highest criticality scores and were prioritised for mitigation. For instance, the risk of large uncontrolled motion at the tooltip, with a likelihood and catastrophic severity, was mitigated by implementing safety interlocks, PID tuning, and extensive testing. The full risk assessment table, including all identified risks and their mitigation plans, is available in Appendix B.

4.2. Limitations

One notable limitation was the decrease in accuracy near the boundaries of the workspace, where deviations were significantly higher due to mechanical instability and reduced manipulability. The rigid design of the robotic arm, while effective for centre positions, limited flexibility and adaptability in dynamic or complex trajectories.

The analytical kinematics approach, though computationally efficient, is restricted to well-defined workspaces and struggles near singularities. Alternative numerical methods such as DLS could provide robustness but were not implemented due to computational overhead. Additionally, the PID controller required manual tuning, making it less adaptable to changing tasks or environments.

4.3. Future Work

Adaptive control strategies, such as model predictive control or machine-learning-based PID tuning, could be integrated to dynamically adjust parameters in real time, improving performance across varying tasks. Incorporating more advanced kinematic methods, such as hybrid approaches that combine analytical and numerical solutions, could provide flexibility while maintaining computational efficiency.

5. CONCLUSION

This project successfully designed, developed, and tested a robotic arm capable of precise movements for drawing to different points within a 156 mm x 156 mm workspace. Through consideration of kinematic methods and PID control strategies, the robotic arm demonstrated accurate positioning and responsive performance. The use of analytical inverse kinematics provided efficient and direct solutions for joint angles, ensuring precision in the well-defined workspace. PID tuning allowed for an effective balance between speed and accuracy, with the optimal gains minimising overshoot while maintaining stability and fast response times. The accuracy tests highlighted the arm's ability to achieve high precision near the workspace centre, though deviations increased at the boundaries due to mechanical and kinematic constraints. Similarly, the speed versus accuracy tests underscored the inherent trade-offs in PID-controlled systems, emphasising the importance of parameter optimisation for task-specific requirements. While the robotic arm performed effectively for its intended purpose, the project revealed several limitations, including accuracy degradation at workspace extremities and the lack of adaptability in control strategies. Addressing these challenges through adaptive control systems, improved mechanical design, and advanced manufacturing techniques would significantly enhance the arm’s functionality and versatility. In conclusion, the project demonstrated the feasibility of developing a low-cost, precise robotic arm for specialised tasks such as drawing. It provides a foundation for further research and development, paving the way for more robust and adaptable robotic systems for various applications.

6. REFERENCES

[1] J. J. C. S. Krovi, A. M. Okamura, and J. Rosen, "Guest editorial: Special issue on surgical robotics," *IEEE/ASME Transactions on Mechatronics*, vol. 20, 2015. [Online]. Available: https://web.eecs.umich.edu/~jjcorso/pubs/Krovi_SurgRob_SpIssue_DSC_Magazine_September2015.pdf.

[1] J. J. C. S. Krovi, A. M. Okamura, and J. Rosen, "Guest editorial: Special issue on surgical robotics," *IEEE/ASME Transactions on Mechatronics*, vol. 20, 2015. [Online]. Available: https://web.eecs.umich.edu/~jjcorso/pubs/Krovi_SurgRob_SpIssue_DSC_Magazine_September2015.pdf.

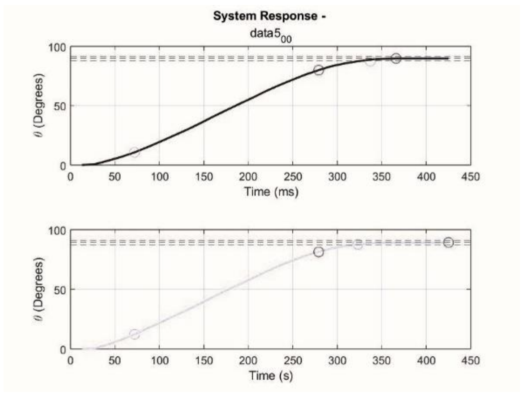
[2] H.-J. K. T. D. Le, and Q. V. Doan, "A method for optimal kinematic design of five-bar planar parallel manipulators," *Proc. Int. Conf. on Control, Automation and Information Sciences* 2013.

[3] A. Umamaheswara Rao and H. Arora, *A Review of Methods Used for Kinematic Modeling of A Manipulator*. 2017.

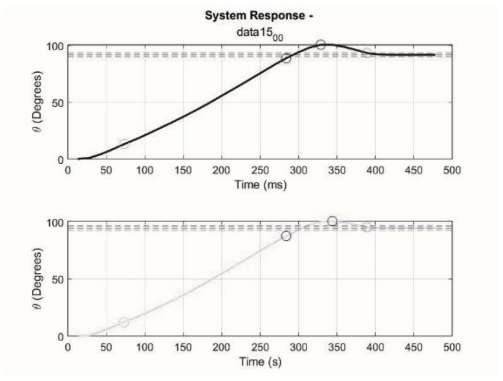
[4] J. Lin and Z. Z. Huang, "A novel PID control parameters tuning approach for robot manipulators mounted on oscillatory bases," *Robotica*, vol. 25, no. 4, pp. 467-477, 2007, doi: 10.1017/S0263574706003298.

[5] W. R. Library, "Introduction to PID Control," *WPILib Documentation*. [Online]. Available: https://docs.wpilib.org/en/stable/docs/software/advanced-controls/introduction/introduction-to-pid.html.

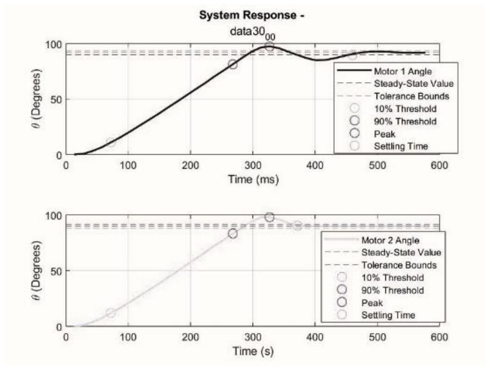
APPENDIX A. PID TUNING GRAPHIC RESULTS



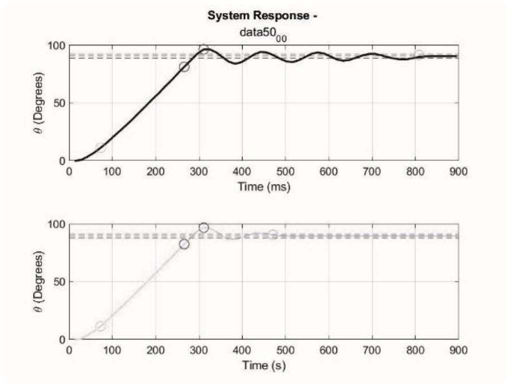
Graph 12 PID Response for Motor 1 and Motor 2 with PID Gains 5-0-0



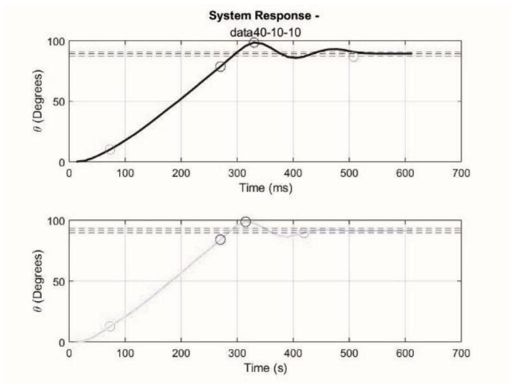
Graph 13 PID Response for Motor 1 and Motor 2 with PID Gains 15-0-0



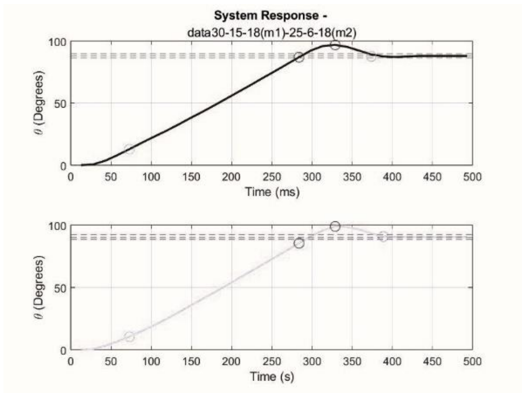
Graph 14 PID Response for Motor 1 and Motor 2 with PID Gains 30-0-0



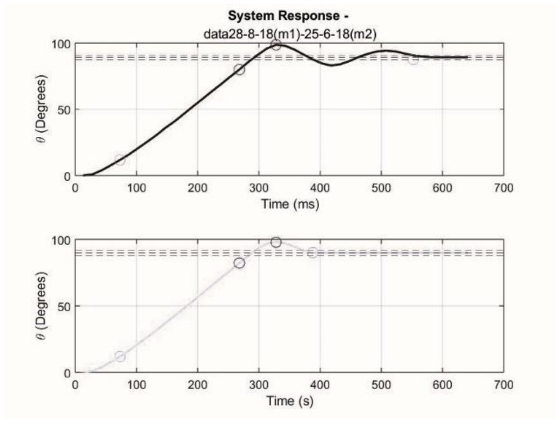
Graph 15 PID Response for Motor 1 and Motor 2 with PID Gains 50-0-0



Graph 16 PID Response for Motor 1 and Motor 2 with PID Gains 40-10-10

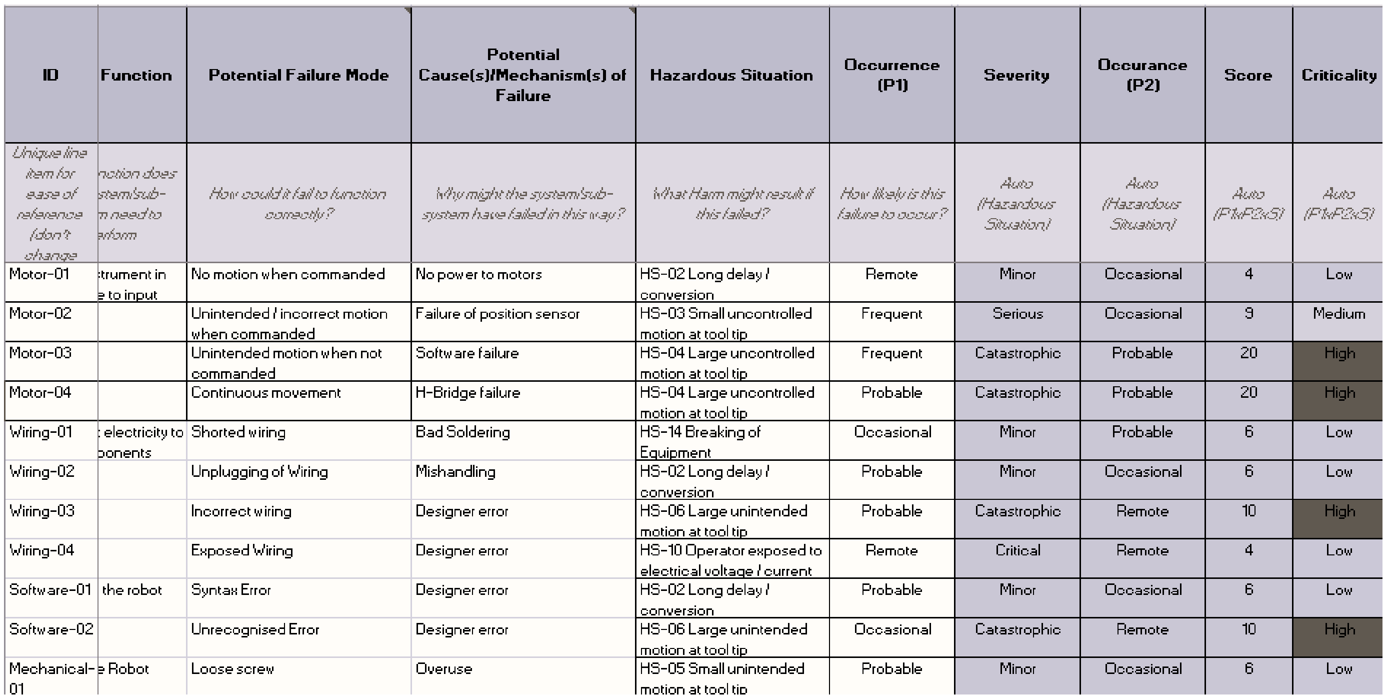


Graph 17 PID Response for Motor 1 and Motor 2 with PID Gains 30-15-18 and 25-6-18 respectively



Graph 18 PID Response for Motor 1 and Motor 2 with PID Gains 28-8-18 and 25-6-18 respectively

APPENDIX B. RISK ASSESSMENT



ID	Function	Potential Failure Mode	Potential Cause(s)/Mechanism(s) of Failure	Hazardous Situation	Occurrence (P1)	Severity	Occurance (P2)	Score	Criticality
Unique line item for ease of reference (don't change	*nction does stem/sub- m need to erform*	*How could it fail to function correctly?*	*Why might the system/sub-system have failed in this way?*	*What Harm might result if this failed?*	*How likely is this failure to occur?*	*Auto (Hazardous Situation)*	*Auto (Hazardous Situation)*	*Auto (P1xP2xS)*	*Auto (P1xP2xS)*
Motor-01	trument in e to input	No motion when commanded	No power to motors	HS-02 Long delay / conversion	Remote	Minor	Occasional	4	Low
Motor-02		Unintended / incorrect motion when commanded	Failure of position sensor	HS-03 Small uncontrolled motion at tool tip	Frequent	Serious	Occasional	9	Medium
Motor-03		Unintended motion when not commanded	Software failure	HS-04 Large uncontrolled motion at tool tip	Frequent	Catastrophic	Probable	20	High
Motor-04		Continuous movement	H-Bridge failure	HS-04 Large uncontrolled motion at tool tip	Probable	Catastrophic	Probable	20	High
Wiring-01	electricity to ponents	Shorted wiring	Bad Soldering	HS-14 Breaking of Equipment	Occasional	Minor	Probable	6	Low
Wiring-02		Unplugging of Wiring	Mishandling	HS-02 Long delay / conversion	Probable	Minor	Occasional	6	Low
Wiring-03		Incorrect wiring	Designer error	HS-06 Large unintended motion at tool tip	Probable	Catastrophic	Remote	10	High
Wiring-04		Exposed Wiring	Designer error	HS-10 Operator exposed to electrical voltage / current	Remote	Critical	Remote	4	Low
Software-01	the robot	Syntax Error	Designer error	HS-02 Long delay / conversion	Probable	Minor	Occasional	6	Low
Software-02		Unrecognised Error	Designer error	HS-06 Large unintended motion at tool tip	Occasional	Catastrophic	Remote	10	High
Mechanical-01	e Robot	Loose screw	Overuse	HS-05 Small unintended motion at tool tip	Probable	Minor	Occasional	6	Low