

Comparative Analysis of PID and Sliding Mode Control for a 3-DoF RRR Robot Arm

Dilara Galeli¹ , Kamil Çetin^{1,2*} 

¹ Izmir Katip Celebi University, Faculty of Engineering and Architecture, Department of Electrical and Electronics Engineering, 35620, Izmir, Türkiye

² Izmir Katip Celebi University, Akıllı Fabrika Sistemleri Uygulama ve Araştırma Merkezi, 35620, Izmir, Türkiye
* kamil.cetin@ikcu.edu.tr

* Orcid No: 0000-0003-1029-5626

Received: July 9, 2025

Accepted: September 21, 2025

DOI: 10.18466/cbayarfbe.1738853

Abstract

This study presents a comparative analysis of Proportional-Integral-Derivative (PID) and Sliding Mode Control (SMC) methods applied to a custom-designed and 3D-printed 3-Degree-of-Freedom (3-DoF) Revolute-Revolute-Revolute (RRR) robotic manipulator. A central contribution of this work is the development of a dual-environment validation framework that integrates ROS Noetic with the Gazebo simulation platform, enabling seamless testing of controllers in both virtual and physical settings. This framework provides a practical pathway for bridging the gap between simulation-based evaluations and real-world experimentation, an aspect that remains underexplored in existing studies. The performance of both controllers is assessed through joint position errors, trajectory tracking accuracy, and torque demands for a cubic trajectory application. Experimental results show that while both controllers achieve satisfactory performance, SMC demonstrates superior trajectory tracking, with consistently lower Root Mean Square (RMS) errors across all joints. This improvement, however, is accompanied by slightly higher torque requirements compared to PID, highlighting the trade-off between enhanced accuracy and increased actuator effort. By combining a low-cost robotic platform with a reproducible dual-environment methodology, this study not only offers insights into the practical strengths and limitations of model-free PID and SMC but also establishes a framework that can inform future research and industrial applications.

Keywords: Sliding Mode Control, PID Control, Robot Arm, Trajectory Tracking, Low-cost

1. Introduction

In recent years, the utilization of robotic manipulators has gained significant attention due to the growing need for autonomy and their increasing range of applications in industry such as manufacturing, welding, packaging, and high-precision tasks [1]. Hence, the effective control of robotic manipulators is vital for ensuring high accuracy, reliability, and stability, especially considering their possible nonlinear dynamics, parameter uncertainties, and external effects such as friction, and varying payloads [2]. In most industrial robots, motion controllers accept externally defined motion profiles, nevertheless the accuracy of trajectory tracking is significantly influenced by the internal dynamics of robot manipulators and the behavior of the joint servo controllers, which presents major challenges in achieving precise control [3].

Of the numerous robotic joint type configurations, the 3-Degree-of-Freedom (DoF) Revolute-Revolute-Revolute (RRR) robot arm represents a foundational design, which provides sufficient workspace while maintaining practicable complexity for control system development and assessment. The trajectory tracking performance problem becomes extremely challenging in higher DoF systems where the complex relation between joint dynamics, actuator characteristics, and coupling effects significantly affects the general system performance [4]. Conventional model-based approaches often require precise knowledge of manipulator dynamics parameters such as link masses, link lengths, mass-inertia, and friction coefficients. These parameters can be difficult to obtain accurately or may change during tasks because of wear, payload variations, or aging of mechanical parts. This limitation has motivated comprehensive research

into control approaches, which can achieve powerful performance without depending on attentive mathematical models of the system dynamics.

Between the control approaches, Proportional-Integral-Derivative (PID) control is one of the most preferred control methods in industrial applications because of its simplicity, easy implementation, uncomplicated tuning methodology, and acceptable performance characteristics in many robotic applications. The PID controller's popularity originates from its straightforward design and its ability to provide effective trajectory following performance without requiring detailed knowledge about the plant dynamics, which makes the use of PID control method especially appealing in practical implementations [5]. Nevertheless, conventional PID controllers may exhibit performance limitations when dealing with nonlinear systems, or applications where high trajectory following precision and fast response time are crucial. In contrast, Sliding Mode Control (SMC), a nonlinear control method developed by [6], provides superior performance characteristics against parametric uncertainties and external disturbances, thanks to its discontinuous nature of the control action [7] and robust control law [8]. The main principle of SMC is to force the system state trajectory to reach and then stay on a predefined sliding surface that is constructed to be insensitive to system uncertainties and disturbances, where the system exhibits desired dynamic behavior despite uncertainties and disturbances within specific bounds [9]. The sliding surface is constructed to be insensitive to system uncertainties and disturbances, which enables consistent and reliable performance. However, this discontinuous control action, while improving robustness, may lead to a chattering phenomenon that can cause mechanical wear. Nonetheless, a carefully designed sliding surface and appropriate switching function selection can minimize this effect and maintain effective performance.

Existing literature on the control of robotic manipulators reveals several critical gaps that motivate this research. Studies [8,10,11] have demonstrated that model-based sliding mode control (SMC) outperforms model-based PID control in two-link manipulators, resulting in faster and more reliable responses [8,10], and lower position tracking errors [11]. However, these model-based approaches depend on dynamic models, which may increase implementation complexity and reduce the models' practicality in real-world scenarios. In addition, the aforementioned studies are limited to 2-DoF systems and are confined to MATLAB simulations without validation on real robotic platforms. Furthermore, [12] has used evolutionary algorithms for model-based SMC and model-based PID optimization on a 2-DoF robot manipulator in the presence and absence of disturbances, however the study remains simulation-based and lacks real-world validation.

The development of the open-source Robot Operating System (ROS) framework has revolutionized robotics research and development by providing a standardized platform for robotic simulation, control, and hardware integration since its development in 2007 [13,14]. The popularity of ROS comes from its modular architecture, support for various programming languages, and open-source nature. The use of ROS with the powerful physics-based 3D simulation tool Gazebo helps robotic researchers to test their robotic systems effectively and safely before real-world implementation [15], which assists to identify and reduce the potential risks in the robotic operation process. Despite the acknowledged advantages of ROS-based Gazebo simulator for robotic manipulator simulation, most of the existing studies in the literature focus either on simulation-only applications or on real-world testing conducted independently of simulation, resulting in a lack of the dual-environment approach that allows simultaneous comparison and validation between the simulation and experimental setups [16,17,18]. Even though 3-DoF robotic manipulator studies exist in the literature focusing on uncertainty and disturbance effects, they generally lack the dual-environment validation framework. Specifically [4] compared PID, SMC, and Fuzzy SMC (FSMC), proposed a self-tuning fuzzy sliding mode controller (ST-FSMC) on a 3-DoF robot arm. However, the study is limited to simulation, therefore lacks experimental validation, and model-based, which may make the real-world implementation more difficult due to the complexity of accurately modeling robotic systems.

This paper conducts a comprehensive comparison of PID and SMC control strategies on a custom-designed, 3D-printed 3-DoF RRR manipulator. Unlike most previous studies that remain confined to simulations or rely heavily on precise dynamic modeling, our approach emphasizes a model-free implementation validated in both simulation and real-world environments. The originality of this work lies in its dual-environment experimental framework, which allows simultaneous observation of the robot's behavior in Gazebo simulation and in a physical setup under ROS Noetic integration. This framework enhances the reliability of the comparative analysis by bridging the persistent gap between virtual validation and hardware experimentation. By focusing on trajectory tracking performance, joint errors, and torque requirements, the study not only clarifies the strengths and limitations of PID and SMC but also contributes to a practical methodology that can inform the broader application of model-free control strategies in robotics.

2. Robot Model and Kinematic Analysis

2.1. Forward Kinematic Analysis

The forward kinematic model of the 3-DoF robot arm is constructed using the Denavit-Hartenberg (D-H) transformation method [19]. By defining link and joint

parameters corresponding to the robot's dimensions and obtaining transformation matrices, the position and orientation of the end effector relative to base frame can be determined from known joint positions. To accomplish this analysis, coordinate frames must first be assigned properly to each joint and link according to the D-H convention. Then, the D-H table must be constructed correctly, and the corresponding transformation matrices must be derived.

Figure 1 shows the frame assignments of the 3-DoF version of Robotis' OpenManipulator-X model robot arm, which serves as a reference for our experimental robot model. Based on these frame assignments, the D-H table is constructed, given in Table 1.

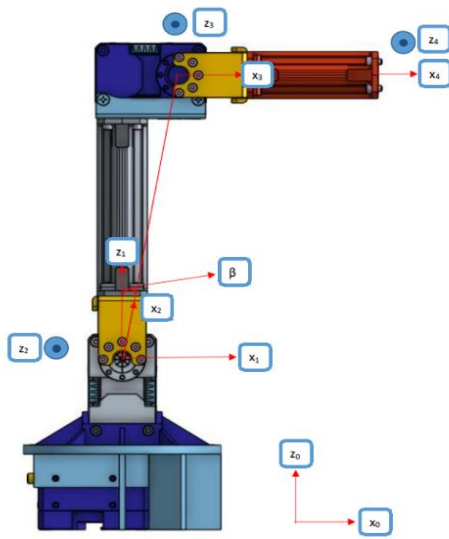


Figure 1. Frame attachments.

Table 1. Denavit–Hartenberg parameters for the 3-DoF robot arm.

Frame	α_{i-1} ($^\circ$)	a_{i-1} (m)	d_i (m)	θ_i ($^\circ$)
1	0	0	d_1	θ_1
2	90	0	0	$\theta_2 + 90 - \beta$
3	0	a_2	0	$\theta_3 - 90 + \beta$
4	0	a_3	0	0

The D-H parameters in Table 1 refer to the following [20]:

- a_i : the distance measured from frame $\hat{Z}_i$ to frame $\hat{Z}_{i+1}$ along frame $\hat{X}_i$
- α_i : the angle measured from frame $\hat{Z}_i$ to frame $\hat{Z}_{i+1}$ about frame $\hat{X}_i$
- d_i : the distance measured from frame $\hat{X}_{i-1}$ to frame $\hat{X}_i$ along frame $\hat{Z}_i$
- θ_i : the angle measured from frame $\hat{X}_{i-1}$ to frame $\hat{X}_i$ about frame $\hat{Z}_i$

According to forward kinematic analysis, the end-effector position equations in Cartesian space are

obtained as follows, where p_x , p_y , and p_z represent the position components along the x , y , and z axes, respectively:

$$p_x = a_3 \cos(\theta_1) \cos(\theta_2 + \theta_3) + a_2 \cos(\theta_1) \cos(\theta_2 + 90^\circ - \beta) \quad (1)$$

$$p_y = a_3 \sin(\theta_1) \cos(\theta_2 + \theta_3) + a_2 \sin(\theta_1) \cos(\theta_2 + 90^\circ - \beta) \quad (2)$$

$$p_z = d_1 + a_3 \sin(\theta_2 + \theta_3) + a_2 \sin(\theta_2 + 90^\circ - \beta) \quad (3)$$

In these equations, θ_1 , θ_2 , and θ_3 are the joint angles in radians, d_1 is the length between the base frame and the first frame in meters; a_2 is the length between the first frame and the second frame in meters; a_3 is the length between the second and the third frames in meters, and β is the offset of the joint angle in radians.

2.2. Trajectory Generation

To enable the robot arm to reach designated target points from the initial point, cubic polynomials are used to generate the trajectory in the joint space. The cubic trajectory is preferred so that the motion between the initial and final positions is smooth and continuous. The equations below represent the general form for cubic polynomials for position (θ_i), velocity ($\dot{\theta}_i$), and acceleration ($\ddot{\theta}_i$) with respect to time (t), where $i = 1, 2, 3$ denotes the joint index:

$$\theta_i(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 \quad (4)$$

$$\dot{\theta}_i(t) = b_1 + 2b_2 t + 3b_3 t^2 \quad (5)$$

$$\ddot{\theta}_i(t) = 2b_2 + 6b_3 t \quad (6)$$

The cubic trajectory is generated according to the following constraints on the initial and final positions and velocities:

$$\theta_i(t_0) = \theta_{i_{t_0}} \quad (7)$$

$$\dot{\theta}_i(t_0) = \dot{\theta}_{i_{t_0}} \quad (8)$$

$$\theta_i(t_f) = \theta_{i_{t_f}} \quad (9)$$

$$\dot{\theta}_i(t_f) = \dot{\theta}_{i_{t_f}} \quad (10)$$

The cubic polynomial coefficients b_0 , b_1 , b_2 , and b_3 can be obtained using the initial and final values of position and velocity. In the trajectory generation process, these coefficients are calculated using the initial position $\theta_{i_{t_0}}$, the final position $\theta_{i_{t_f}}$, the initial velocity $\dot{\theta}_{i_{t_0}}$, the final

velocity $\dot{\theta}_{it_f}$, and the final time t_f . The calculation of the coefficients is carried out as follows

$$b_0 = \theta_{it_0} \quad (11)$$

$$b_1 = \dot{\theta}_{it_0} \quad (12)$$

$$b_2 = \frac{3(\theta_{it_f} - \theta_{it_0})}{t_f^2} - \frac{2\dot{\theta}_{it_0} + \dot{\theta}_{it_f}}{t_f} \quad (13)$$

$$b_3 = -\frac{2(\theta_{it_f} - \theta_{it_0})}{t_f^3} + \frac{\dot{\theta}_{it_0} + \dot{\theta}_{it_f}}{t_f^2} \quad (14)$$

These equations are used to calculate the position, velocity, and acceleration of the robot arm throughout the motion. In the experimental scenario, the duration of motion t_f is selected to be 10 seconds. The initial and final joint positions used in trajectory generation are defined as follows.

The cubic trajectory is defined over the time interval as $t_0 = 0$ and $t_f = 10$. The initial and final joint positions are:

$$\theta_i(t_0) = \begin{bmatrix} \theta_1(t_0) \\ \theta_2(t_0) \\ \theta_3(t_0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (15)$$

$$\theta_i(t_f) = \begin{bmatrix} \theta_1(t_f) \\ \theta_2(t_f) \\ \theta_3(t_f) \end{bmatrix} = \begin{bmatrix} -\pi/2 \\ \pi/6 \\ \pi/12 \end{bmatrix} \quad (16)$$

The initial and final joint velocities are:

$$\dot{\theta}_i(t_0) = \begin{bmatrix} \dot{\theta}_1(t_0) \\ \dot{\theta}_2(t_0) \\ \dot{\theta}_3(t_0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (17)$$

$$\dot{\theta}_i(t_f) = \begin{bmatrix} \dot{\theta}_1(t_f) \\ \dot{\theta}_2(t_f) \\ \dot{\theta}_3(t_f) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (18)$$

Substituting the initial and final positions given in (15) and (16), and the initial and final velocities given in (17) and (18) into equations (11), (12), (13) and (14), the cubic polynomial coefficients b_0 , b_1 , b_2 , and b_3 are calculated, given in Table 2.

Table 2. Cubic polynomial coefficients.

Joint	b_0	b_1	b_2	b_3
1	0	0	-0.04712	0.00314
2	0	0	0.01571	-0.00105
3	0	0	0.00785	-0.00052

Using these coefficients and applying to (4), joint position functions are obtained as follows:

$$\theta_1(t) = -0.04712t^2 + 0.00314t^3 \quad (19)$$

$$\theta_2(t) = 0.01571t^2 - 0.00105t^3 \quad (20)$$

$$\theta_3(t) = 0.00785t^2 - 0.00052t^3 \quad (21)$$

2.3. Gazebo Simulation

Gazebo is a powerful 3D simulator for robots within the ROS framework, which helps robotics researchers simulate and test their robotic developments such as mechanical, software, controller, and trajectory generation before and during real-world implementations [15]. The experiments are conducted using the simulation environment of the Robot Operating System (ROS Noetic), with Gazebo 11.11.0, which operates under the Ubuntu 20.04 (64-bit) operating system, where the planned cubic trajectories are concurrently monitored and evaluated on both the physical 3-DoF robot arm and within the ROS-based Gazebo simulation environment. The cubic trajectories are implemented using PID and sliding mode controllers separately, with Python scripts developed for each controller. Figure 2 shows the initial position of the robot arm (as given in Equation 15), and Figure 3 shows the final position (as given in Equation 16) for both control strategies.

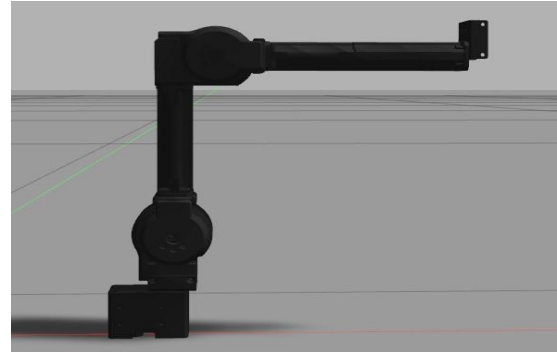


Figure 2. Starting position in Gazebo.



Figure 3. Final position in Gazebo.

3. Control Strategies

3.1. PID Control

In this study, a Proportional-Integral-Derivative (PID) control method is used where the proportional term provides an immediate response to current errors, the integral term eliminates steady-state errors by summing past error values, and the derivative term predicts future errors by responding to the rate of error [21]. In addition, a feedforward term is added using the desired accelerations for each joint, which anticipates the required control effort based on the reference trajectory's acceleration. As a result, tracking performance is improved by providing the nominal torque that is needed to follow the desired trajectory before any error occurs.

This control strategy does not require detailed mathematical models (since it is a model-free approach) of the manipulator dynamics, where complex system characteristics such as coupling and friction effects may be challenging to model accurately.

The error signal $e_i(t)$ for joints $i = 1, 2, 3$ is calculated as follows:

$$e_i(t) = \theta_{i_{desired}}(t) - \theta_{i_{actual}}(t) \quad (22)$$

where $\theta_{i_{desired}}(t)$ represents the desired joint position and $\theta_{i_{actual}}(t)$ is the actual joint position of the joint i . The complete control law implemented in the code combines feedforward and feedback control as follows:

$$u_i(t) = \ddot{\theta}_{i_{desired}}(t) + K_p e_i(t) + K_i \int e_i(t) dt + K_d \frac{d}{dt} e_i(t) \quad (23)$$

where $\ddot{\theta}_{i_{desired}}(t)$ is the feedforward term that represents the desired acceleration of the joint i , K_p , K_i and K_d represent the controller gains of proportional, integral, and derivative parameters, respectively and $u_i(t)$ is the torque control input of the joint i . In code implementation, PID gains are selected based on multiple trials as $K_p = 3$, $K_i = 0.001$, and $K_d = 0.5$ for avoiding instability and ensuring effective performance. Therefore, $u_i(t)$ becomes:

$$u_i(t) = \ddot{\theta}_{i_{desired}}(t) + 3e_i(t) + 0.001 \int e_i(t) dt + 0.5 \frac{d}{dt} e_i(t) \quad (24)$$

The computed torque control input is then converted to PWM signals for the Dynamixel servo motors using the scaling factor 885/4.1, where 885 represents the maximum PWM range (± 885) of the Dynamixel XM430-W350-T motors and 4.1 Nm corresponds to the maximum stall torque specification of the motors.

3.2. Sliding Mode Control

In this study, a Sliding Mode Control (SMC) method is implemented to handle system uncertainties and external disturbances, which is crucial for robot manipulators. The sliding mode control strategy uses a discontinuous control mechanism that forces the states of the system to reach and remain on a predefined sliding surface, where the system exhibits desired behavior and becomes insensitive to uncertainties [9,22]. The control approach for SMC does not require precise mathematical models of the manipulator dynamics, which is vital for systems with modeling uncertainties and external disturbances. Although SMC is widely recognized for its robustness and fast dynamic response, its main drawback is the chattering phenomenon, which may impose additional stress on the joints and lead to mechanical wear [22]. In this study, to minimize this chattering effect, a smooth switching function hyperbolic tangent function ($\tanh$) is utilized instead of the commonly used signum function (sgn), following a similar approach as in [23,24,25,26,27,28,29,30,31].

In the traditional SMC method [6], the control inputs $u_i(t)$ are obtained using only the system output, that is position $\theta_i(t)$ and velocity $\dot{\theta}_i(t)$. The design of the sliding surface $s_i(t)$ is based on tracking error $e_i(t)$:

$$e_i(t) = \theta_{i_{desired}}(t) - \theta_{i_{actual}}(t) \quad (25)$$

and velocity error $\dot{e}_i(t)$:

$$\dot{e}_i(t) = \dot{\theta}_{i_{desired}}(t) - \dot{\theta}_{i_{actual}}(t) \quad (26)$$

The sliding surface s is defined as [32]:

$$s_i(t) = \lambda_i e_i(t) + \dot{e}_i(t) \quad (27)$$

where λ_i is a positive design parameter that determines the convergence rate to the sliding surface.

The control law takes the form [33]:

$$u_i(t) = -u_{0_i} sgn(s_i(t)) \quad (28)$$

where u_{0_i} is the control gain for each joint i .

Smooth switching functions such as $\tanh(\cdot)$ can be used to fix the chattering problem, which is associated with the delay in control switching caused by the use of the signum function [23,29,33].

In the SMC approach, the controller design follows a similar methodology as presented in [24]. The position and velocity errors for each joint are calculated as follows:

$$e_i(t) = \theta_{i_{desired}}(t) - \theta_{i_{actual}}(t) \quad (29)$$

$$\dot{e}_i(t) = \dot{\theta}_{i_{desired}}(t) - \dot{\theta}_{i_{actual}}(t) \quad (30)$$

where $\theta_{i_{desired}}(t)$ and $\dot{\theta}_{i_{desired}}(t)$ represent the desired position and velocity, while $\theta_{i_{actual}}(t)$ and $\dot{\theta}_{i_{actual}}(t)$ are the actual position and velocity of the joint i . The sliding surface s for each joint is defined as:

$$s_i(t) = \lambda_i e_i(t) + \dot{e}_i(t) \quad (31)$$

where λ_i is a positive design parameter that determines the convergence rate to the sliding surface. The control law implemented in the code uses continuous approximation to avoid chattering with

$$u_i(t) = u_{0_i} \tanh(s_i(t)) \quad (32)$$

where u_{0_i} represents the control gain for the joint i , $\tanh(\cdot)$ is the hyperbolic tangent function used as a smooth approximation of the signum function, and $u_i(t)$ is the torque control input of the joint i . This smooth switching function eliminates the high frequency chattering typically associated with the use of the signum function in traditional sliding mode control while maintaining robustness properties [23,29,33]. The computed torque control input is then converted to PWM signals for the Dynamixel servo motors using the scaling factor $885/4.1$, where 885 represents the maximum PWM range (± 885) of the Dynamixel XM430-W350-T motors and 4.1 Nm corresponds to the maximum stall torque specification of the motors.

The implementation of sliding mode control uses the following design parameters: sliding surface parameters $\lambda_1 = 10.0$, $\lambda_2 = 9.0$, and $\lambda_3 = 5.0$, which determine the convergence rate to the sliding surface for joints 1, 2, and 3 respectively, with higher values providing faster convergence, and control gains $u_{0_1} = 1.0$ Nm, $u_{0_2} = 0.85$ Nm, and $u_{0_3} = 0.75$ Nm that define the maximum control torque output for each joint.

4. Experimental Setup

4.1. 3-DoF Robot Arm

For this study, a 3-DoF all-revolute joint robot arm is custom designed and fabricated using 3D printing technology to perform the required tasks. Figure 4 shows the appearance of the robot used in this study.



Figure 4. 3-DoF robot arm.

4.2. Dynamixel XM430-W350-T Servo Motor

In this study, the movements of the joints are enabled by using Dynamixel XM430-W350-T servo motors, with one motor assembled at each joint, allowing each joint to be independently driven. This motor model is selected because of its high torque output and PWM control option, both of which are crucial in our case.

Dynamixel XM430-W350-T supports multiple control modes, such as current control mode, velocity control mode, position control mode, extended position control mode, current-based position control mode and PWM control mode, also known as voltage control mode. In our case, the servo motors are operated in the PWM control mode, where the applied PWM signal directly controls the motor voltage. The PWM control ranges from -885 to +885, which corresponds to a full duty cycle. The motor delivers a maximum stall torque of 4.1 Nm at 12 V - 2.3 A, which is sufficient for the torque requirements of each link in the robotic arm. Communication is achieved using the serial TTL protocol. Figure 5 shows the appearance of the Dynamixel XM430-W350-T servo motor.



Figure 5. Dynamixel XM430-W350-T servo motor.

4.3. OpenCM 9.04-C Microcontroller

In this study, the OpenCM 9.04-C controller is preferred as the main control unit to control the robotic arm due to its compatibility with Dynamixel servo motors, easy ROS integration and real-time robotic control capabilities with the TTL communication protocol. The appearance of the OpenCM 9.04-C controller is given in Figure 6. The controller has an ARM Cortex-M3 32-bit processor (STM32F103CB) operating at 72MHz, which provides sufficient computational power for our case. The board accepts the voltage range from 5V to 16V, while operating at the 3.3V logic level internally. In our case, the controller is powered via USB (5V) for programming purposes, where Arduino IDE is used for programming the controller. However, an external power supply is needed to control multiple high-torque servo motors to obtain adequate current delivery for all motors. Therefore, an expansion board, namely OpenCM 485, is used for this purpose. Further details about the power configuration and expansion module are provided in a later section.



Figure 6. OpenCM 9.04-C controller.

4.4. OpenCM 485 Expansion Board

The OpenCM 485 expansion board is crucial for stable power distribution and extension of the communication interface for the OpenCM 9.04-C controller in the robotic arm implementation. This board fixes the power limitations of USB-powered operation by providing power input jack and terminals that can handle higher current demands required for simultaneous operation of multiple Dynamixel servo motors. In our setup, a 12V - 2A adapter is connected to the board's power jack to supply sufficient power for the entire system. The board features multiple RS-485 and TTL communication ports. In our system, all TTL connections to the three servo motors of the robotic arm are established through the dedicated communication ports on the expansion board. The appearance of the OpenCM 485 expansion board is shown in Figure 7.

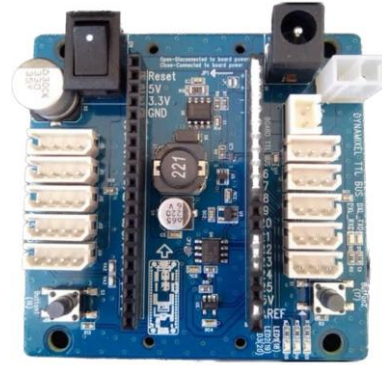


Figure 7. OpenCM 485 expansion board.

The complete configuration of the robotic system, including all electrical connections and component interfaces, is shown in Figure 8.



Figure 8. Robotic system configuration.

5. Results and Discussion

The performance results of PID and sliding mode control algorithms for identical cubic trajectory are obtained in terms of joint trajectories, joint position errors, and joint torque performance criteria. Performance graphs are generated by processing the recorded rosbag files during the experimental process in MATLAB R2024b for data analysis and visualization of the graphs.

Figure 9 shows the joint position tracking performance for the implementation of cubic trajectory using the PID control approach. The graph illustrates both actual and desired joint positions in radians as functions of time in seconds. In the graph, θ_1 , θ_2 and θ_3 represent the actual

joint positions, depicted in blue, of joint 1, joint 2 and joint 3, respectively. Similarly, θ_{d1} , θ_{d2} and θ_{d3} represent the desired joint positions, depicted in red, of joint 1, joint 2 and joint 3, respectively.

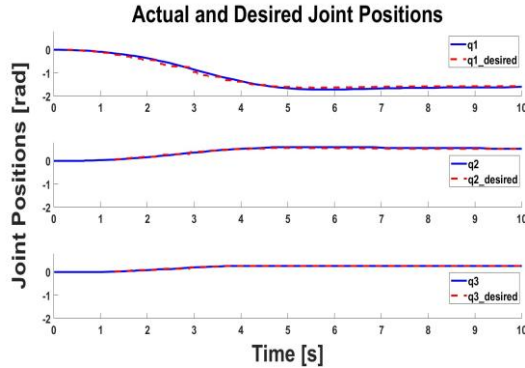


Figure 9. Actual and desired joint positions in PID control.

When Figure 9 is examined, it is seen that the actual and desired joint position trajectories are closely aligned with each other, which means that the desired cubic trajectory is approximately followed for all three joints. As a result of the PID control application for the 3-DoF RRR robotic arm system, it is seen that there may be slight deviations between the actual and desired joint positions.

Figure 10 shows the joint position tracking performance for the implementation of cubic trajectory using the sliding mode control approach. The graph illustrates both actual and desired joint positions in radians as functions of time in seconds. In the graph, θ_1 , θ_2 and θ_3 represent the actual joint positions, depicted in blue, of joint 1, joint 2 and joint 3, respectively. Similarly, θ_{d1} , θ_{d2} and θ_{d3} are the desired joint positions, depicted in red, of joint 1, joint 2 and joint 3, respectively.

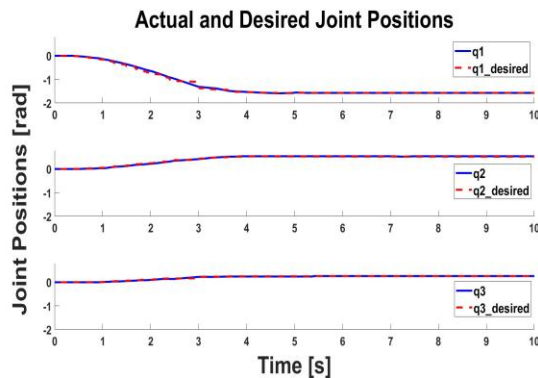


Figure 10. Actual and desired joint positions in sliding mode control.

From the observation of Figure 10, the actual and desired joint position trajectories align closely with each other, meaning that the desired cubic trajectory is followed accurately for all three joints. The minimal deviation between the actual and desired joint positions shows the effectiveness of the sliding mode control implementation

for the 3-DoF RRR robotic arm system. Although both the PID and sliding mode controllers demonstrate that the actual and desired trajectory profiles align closely, the sliding mode control displays slightly better trajectory following performance with tighter convergence between the actual and desired positions.

Figure 11 illustrates the joint position error profiles in radians as functions of time for the implementation of the cubic trajectory using the PID control method. The graph shows the position errors for three joints with e_1 (in red), e_2 (in black), and e_3 (in blue), respectively.

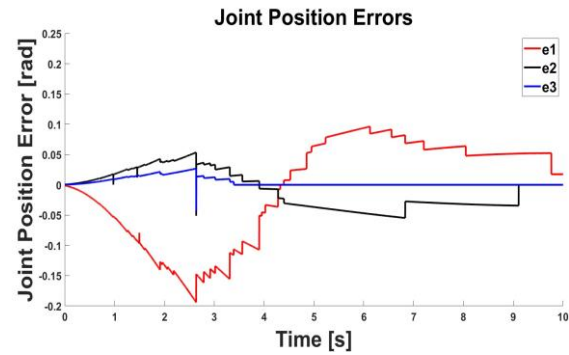


Figure 11. Joint position errors in PID control.

In Figure 11, the position errors of the three joints remain within a certain range throughout the trajectory and almost converge to zero at the end of the motion.

Figure 12 illustrates the joint position error in radians as functions of time for the cubic trajectory implementation using the sliding mode control method. The graph shows the position errors for three joints with e_1 (in red), e_2 (in black), and e_3 (in blue), respectively.

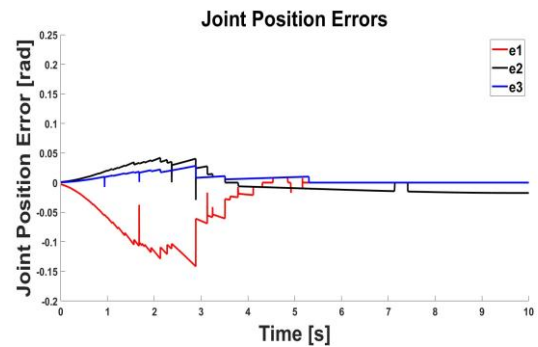


Figure 12. Joint position errors in sliding mode control.

In Figure 12, the position errors for three joints using the sliding mode control remain within a significantly smaller range throughout the execution of the trajectory compared to the PID control and converge to zero at the end of the motion. This demonstrates the superior effectiveness of the sliding mode control strategy in achieving more precise trajectory following in joint space with minimal steady-state error.

Figure 13 shows the joint torque profiles for the implementation of cubic trajectory using the PID control method. The graph shows the torques in Newton meters as functions of time for three joints with torque 1 (in red), torque 2 (in black), and torque 3 (in blue), respectively.

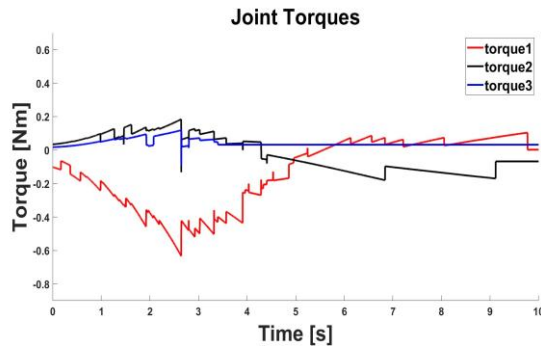


Figure 13. Joint torques in PID control.

In Figure 13, the joint torques for three joints show smooth variations without dramatic changes, indicating stable control performance. The PID controller generates an appropriate torque command, which enables successful trajectory following while maintaining manageable actuator demands across three joints of the robotic arm system.

Figure 14 shows the joint torque profiles for the cubic trajectory implementation using the sliding mode control method. The graph shows the torques in Newton meters as functions of time for three joints with torque 1 (in red), torque 2 (in black), and torque 3 (in blue), respectively.

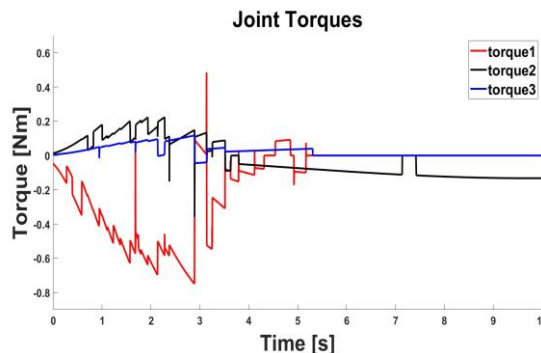


Figure 14. Joint torques in sliding mode control.

In Figure 14, the joint torques for three joints generally exhibit smooth variations throughout the following trajectory, except for joint 1 which shows an abrupt torque spike at one point during the motion. Although joints 2 and 3 maintain relatively consistent torque profiles that correspond to the dynamic requirements of the cubic trajectory, the torque increase in joint 1 reflects the characteristic control action of the sliding mode controllers when confronting system uncertainties or disturbances. This behavior shows that the sliding mode controller requires higher torques compared to PID control to achieve the superior trajectory tracking accuracy, illustrating the trade-off between improved

trajectory following precision and increased actuator effort in robust control strategies.

Table 3, 4 and 5 show the root mean square (RMS) joint position errors, average joint torque efforts, and maximum joint error deviations for PID and SMC, respectively.

Table 3. RMS values of joint position errors in PID and SMC.

	e_1 (rad)	e_2 (rad)	e_3 (rad)
PID	0.0671	0.0246	0.0063
SMC	0.0269	0.0179	0.0051

Table 4. Average values of joint torque in PID and SMC.

	τ_1 (Nm)	τ_2 (Nm)	τ_3 (Nm)
PID	0.061	0.315	0.263
SMC	0.045	0.277	0.241

Table 5. Maximum deviation of joint position errors in PID and SMC.

	Δ_1 (rad)	Δ_2 (rad)	Δ_3 (rad)
PID	0.173	0.250	0.193
SMC	0.120	0.247	0.192

First, the RMS values of the joint position errors (Table 3) indicate that the SMC controller consistently achieves lower error magnitudes across all joints. For instance, in Table 3, the RMS error of e_1 drops from 0.0671 rad under PID to 0.0269 rad under SMC, while similar improvements are observed for e_2 and e_3 . This demonstrates that SMC offers superior precision and robustness in trajectory tracking compared to the conventional PID approach. Second, regarding the average torque demand (Table 4), both controllers exhibit comparable levels, but SMC generally requires slightly lower mean torque values. For example, the average torque τ_1 reduces from 0.061 Nm in PID to 0.045 Nm in SMC, suggesting that SMC not only improves tracking accuracy but also does so with more efficient actuator effort. This characteristic is particularly valuable in robotic systems where minimizing energy consumption and mechanical stress is important. Finally, the maximum deviations of joint position errors (Table 5) further highlight the advantage of SMC. Across all joints, the maximum excursions are either reduced or maintained close to PID levels, with Δ_1 , Δ_2 and Δ_3 notably improved under SMC. This implies that SMC enhances system stability and mitigates large error

spikes, which are critical for safety and reliability in practical implementations.

6. Conclusion

In this paper, a performance analysis of two control methods, namely PID and SMC, is conducted on a custom-designed and 3D-printed 3-DoF RRR robot manipulator. By evaluating these controllers based on joint position errors, trajectory tracking, and joint torques, the paper provides remarkable results regarding their performance for cubic trajectory implementation. The analysis is performed using a real environment validation framework, which validates the controller performance in the ROS-based physical setup. This approach provides a methodological advancement in robotics that establishes real-world implementation. In conclusion, the results demonstrate that while both control strategies exhibit applicable performance for the cubic trajectory implementation, the SMC approach achieves superior trajectory tracking performance with lower RMS values of joint position errors in each joint. In addition, joint torque analysis shows that the SMC requires lower actuator torque when achieving better trajectory tracking performance compared to the PID control. It should be noted that the experiments were performed on a 3D-printed manipulator, which differs in structural rigidity and material properties from industrial systems. Therefore, while the comparative insights between PID and SMC are valuable, further validation on industrial manipulators is required to fully generalize the findings. Although SMC offers robustness, accuracy, and efficiency, it is challenging to adopt in industrial applications due to issues such as complexity, vibration, difficulty in tuning, and industrial conservatism. Addressing these challenges through simplified design methods, improved software tools, and industrial demonstrations, as demonstrated in this study, could accelerate its wider adoption. Future work may extend this framework to higher degree-of-freedom industrial manipulators, investigating hybrid control strategies that combine the benefits of both approaches, and validating the performances of the controllers under various complex trajectory types to improve the practical applicability in industrial robotic applications.

Acknowledgement

This study was supported by Scientific and Technological Research Council of Türkiye (TUBITAK) under the Grant Numbers TEYDEB-5240102 and BIDEB-1139B412302608. The authors thank TUBITAK for their support.

Author's Contributions

Dilara Galeli: Drafted and wrote the manuscript, performed the experiment and result analysis.

Kamil Çetin: Supervised the overall structure of the manuscript, the progress of the experiments, the analytical analysis of the results, and revised the written manuscript.

Ethics

There are no ethical issues after the publication of this manuscript.

References

- [1]. Adebayo, RA, Obiuto, NC, Festus-Ikhuoria, IC, Olajiga, OK. 2024. Robotics in manufacturing: A review of advances in automation and workforce implications. *International Journal of Advanced Multidisciplinary Research and Studies*; 4(2): 632-638.
- [2]. Wang, X, Hou, B. 2021. Continuous time-varying feedback control of a robotic manipulator with base vibration and load uncertainty. *Journal of Vibration and Control*; 27(3-4): 392-403.
- [3]. Chen, S, Wen, JT. 2021. Industrial robot trajectory tracking control using multi-layer neural networks trained by iterative learning control. *Robotics*; 10(1): 50.
- [4]. Ashagrie, A, Salau, AO, Weldcherkos, T. 2021. Modeling and control of a 3-DOF articulated robotic manipulator using self-tuning fuzzy sliding mode controller. *Cogent Engineering*; 8(1): 1950105.
- [5]. Yin, X, Pan, L. 2018. Enhancing trajectory tracking accuracy for industrial robot with robust adaptive control. *Robotics and Computer-Integrated Manufacturing*; 51: 97-102.
- [6]. Utkin, VI. 1977. Variable structure systems with sliding modes. *IEEE Transactions on Automatic Control*; 22(2): 212-222.
- [7]. Young, KD, Utkin, VI, Ozguner, U. 1999. A control engineer's guide to sliding mode control. *IEEE Transactions on Control Systems Technology*; 7(3): 328-342.
- [8]. Mohammed, AA, Eltayeb, A. Dynamics and control of a two-link manipulator using PID and sliding mode control. In 2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCEEE), Khartoum, Sudan, 2018, pp 1-5.
- [9]. Derbel, N, Ghommam, J, Zhu, Q (eds). Applications of Sliding Mode Control; Springer: Singapore, 2017; pp 335-337.
- [10]. Singh, P, Prasad, L. 2020. A comparative performance analysis of PID control and sliding mode control of two-link robot manipulator. *International Research Journal on Advanced Science Hub*; 2: 43-54.
- [11]. Ohri, J, Vyas, DR, Topno, PN. Comparison of robustness of PID control and sliding mode control of robotic manipulator. In Proceedings of the International Symposium on Devices MEMS Intelligent Systems and Communication (ISDMISC), Sikkim, India, 2011, pp 5-10.
- [12]. Loucif, F, Kechida, S. 2020. Optimization of sliding mode control with PID surface for robot manipulator by evolutionary algorithms. *Open Computer Science*; 10(1): 396-407.
- [13]. Megalingam, RK, Rajendraprasad, A, Manoharan, SK. 2020. Comparison of planned path and travelled path using ROS navigation stack. In International Conference for Emerging Technology, Belgaum, India, 2020, pp 1-6.
- [14]. Breiling, B, Dieber, B, Schartner, P. Secure communication for the robot operating system. In IEEE International Systems Conference, Montreal, QC, Canada, 2017, pp 1-6.

- [15]. Quigley, M, Conley, K, Gerkey, B, et al. ROS: an open-source Robot Operating System. In ICRA Workshop on Open-Source Software, Kobe, Japan, 2009, 3-3.2, p 5.
- [16]. Qian, W, Xia, Z, Xiong, J, Gan, Y, Guo, Y, Weng, S, Zhang, J. Manipulation task simulation using ROS and Gazebo. In 2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014), Bali, Indonesia, 2014, pp 2594-2598.
- [17]. Deng, H, Xiong, J, Xia, Z. Mobile manipulation task simulation using ROS with MoveIt. In Proceedings of the 2017 IEEE International Conference on Real-time Computing and Robotics (RCAR), Okinawa, Japan, 2017, pp 612-616.
- [18]. Chen, J, Deng, H, Chai, W, Xiong, J, Xia, Z. Manipulation task simulation of a soft pneumatic gripper using ROS and Gazebo. In Proceedings of the 2018 IEEE International Conference on Real-time Computing and Robotics (RCAR), Kandima, Maldives, 2018, pp 378-383.
- [19]. Denavit, J, Hartenberg, RS. 1955. A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*; 22: 215-221.
- [20]. Craig, J. Introduction to Robotics: Mechanics and Control; 4th edn. Pearson: Boston, USA, 2018.
- [21]. Ogata, K. Modern Control Engineering; 5th ed. Prentice Hall: Upper Saddle River, NJ, 2010.
- [22]. Tinoco, V, Silva, MF, Santos, FN, Morais, R, Magalhães, SA, Oliveira, PM. 2025. A review of advanced controller methodologies for robotic manipulators. *International Journal of Dynamics and Control*; 13(1): 1-17.
- [23]. Soon, CC, Ghazali, R, Ghani, MF, Shern, CM, Sam, YM, Has, Z. 2022. Chattering analysis of an optimized sliding mode controller for an electro-hydraulic actuator system. *Journal of Robotics and Control (JRC)*; 3(2): 160-165.
- [24]. Abukan, Y, Almalı, MN. 2023. Control of 2-DoF TRMS MIMO system using FOPID & FOSTSMC method. *Journal of the Faculty of Engineering and Architecture of Gazi University*; 38(1): 605-615.
- [25]. Wang, J, Zhu, H, Zhang, C, Chen, Z, Huang, Y, Chen, W, et al. 2018. Adaptive hyperbolic tangent sliding-mode control for building structural vibration systems for uncertain earthquakes. *IEEE Access*; 6: 74728-74736.
- [26]. Shi, Z, Deng, C, Zhang, S, Xie, Y, Cui, H, Hao, Y. 2020. Hyperbolic tangent function-based finite-time sliding mode control for spacecraft rendezvous maneuver without chattering. *IEEE Access*; 8: 60838-60849.
- [27]. Gu, J, Pan, S. 2024. Sliding mode control of dual active bridge converter based on hyperbolic tangent function. *Journal of Physics: Conference Series*; 2803(1): 012050.
- [28]. Zhang, M, Zhang, Y, Chen, H, Cheng, X. 2019. Model-independent PD-SMC method with payload swing suppression for 3D overhead crane systems. *Mechanical Systems and Signal Processing*; 129: 381-393.
- [29]. Fang, JS, Tsai, JSH, Yan, JJ, Guo, SM. 2019. Adaptive chattering-free sliding mode control of chaotic systems with unknown input nonlinearity via smooth hyperbolic tangent function. *Mathematical Problems in Engineering*; 2019(1): 4509674.
- [30]. Guo, L, Huangfu, Y, Ma, R. A novel high-order sliding mode observer based on tanh-function for a fuel cell UAV power system with uncertain disturbance. In 2019 IEEE Industry Applications Society Annual Meeting, Baltimore, MD, USA, 2019, pp 1-7.
- [31]. Guan, M, Qu, C, Lv, J, Yang, L, Li, X. 2024. A novel RBF neural network-based sliding mode controller for a master-slave motor coordinated drive system. *The International Journal of Advanced Manufacturing Technology*; 133(9): 4907-4921.
- [32]. Slotine, J-JE, Li, W. Applied Nonlinear Control; Prentice Hall: Englewood Cliffs, NJ, 1991; pp 276-278.
- [33]. Khalil, HK. Nonlinear Systems; Prentice Hall: Upper Saddle River, NJ, 2002; pp 552-575.