MEKATRONIKA
JOURNAL OF MECHATRONICS AND INTELLIGENT MANUFACTURING
E-ISSN: 2637-0883
VOLUME 7, ISSUE 1, JANUARY, 2025, 30 – 40
DOI: https://doi.org/10.15282/mekatronika.v7i1.12151

RESEARCH ARTICLE

A Two-Wheel Mobile Robot Platform for Simulating and Understanding Spacecraft Attitude Dynamics and Control

Dwi Pebrianti[1,*], Amir Ariff Abd. Halim[1], Amirullah Jais[1], Mohamad Mukri Najmi Muhammad Nadzri[1], Rosdiyana Samad[2] and Luhur Bayuaji[3]

[1]Department of Mechanical & Aerospace Engineering, Kulliyyah of Engineering, International Islamic University Malaysia, Jalan Gombak, 53100, Kuala Lumpur, Malaysia
[2]Faculty of Electrical & Electronics Engineering Technology, Universiti Malaysia Pahang Al-Sultan Abdullah, 26600 Pekan, Pahang, Malaysia
[3]Faculty of Data Science & Information Technology, INTI International University, Persiaran Perdana BBN Putra Nilai, 71800 Nilai, Negeri Sembilan, Malaysia

ABSTRACT - This study investigates the design and simulation of spacecraft attitude control using a two-wheeled robot as a model. Spacecraft attitude control is a complex system requiring precision in maintaining the orientation of a spacecraft relative to a desired reference frame. However, due to its complexity, it is challenging to understand the system dynamics directly. Thus, this research simulates spacecraft attitude control by mimicking the system behavior using a two-wheeled, Arduino-based mobile robot. The primary objective is to develop a control system that reads gyroscope data and adjusts the motor speeds to replicate spacecraft stabilization through reaction wheels. Additionally, a PID controller is implemented to control the robot wheel's speed to simulate the operation of reaction wheels. The platform and model were designed using Arduino and tested with system identification techniques. The motor's RPM and transfer function were modeled to analyze and fine-tune the PID controller, which significantly enhanced the system's performance. A Particle Swarm Optimization (PSO) method was used to determine the optimal PID parameters, improving system stability and precision in the movement. The results indicated that the system achieved stability with less than 10% steady-state error, minimal overshoot, and fast settling times, thus demonstrating effective attitude control through PID regulation. The system exhibited the ability to adjust the motor speed based on the yaw angle, thereby simulating the torque-free motion typical of spacecraft. This study demonstrates how combining theoretical modeling, simulation, and hardware implementation can simplify the understanding of spacecraft attitude control.

ARTICLE HISTORY
Received : 23rd Oct 2024
Revised : 19th Dec 2024
Accepted : 27th Dec 2024
Published : 10th Jan 2025

KEYWORDS
Spacecraft Attitude Control
PID Controller
Reaction Wheels
Particle Swarm Optimization (PSO)
Two-Wheeled Mobile Robot

1.0 INTRODUCTION

Spacecraft attitude control is a critical process that involves maintaining and adjusting the orientation of a spacecraft relative to an inertial frame of reference or other entities such as the celestial sphere, magnetic fields, or nearby objects [1]. This control is essential for various mission objectives including ensuring that high-gain antennas are accurately pointed towards Earth for communication, aligning onboard experiments for precise data collection, managing thermal effects from sunlight, and executing propulsive maneuvers in the correct direction [2]. The broader field that encompasses attitude control, along with navigation and guidance, is known as Guidance, Navigation, and Control (GNC), which plays a pivotal role in spacecraft operations.

Torque-free motion refers to the rotational dynamics of a spacecraft or rigid body in the absence of external torques, which is a common scenario in space missions where external forces are minimal. Reaction wheels are a type of actuator commonly used in spacecraft attitude control systems to adjust the spacecraft's orientation by transferring angular momentum between the wheels and the spacecraft [1]. When a reaction wheel spins, it creates a torque that can be used to change the spacecraft's attitude. Understanding the torque-free motion and dynamics of reaction wheels is crucial for designing effective attitude control systems, as they can significantly affect the spacecraft's rotational behavior, including its stability and maneuverability [3-4].

Recent advancements have introduced a generalized framework that integrates optimization techniques with adaptive nonlinear complementary filtering to improve attitude estimation in multi-sensor systems. This method leverages the Levenberg–Marquardt algorithm to generate quaternion-based attitude measurements, which are used to adaptively correct estimation errors. Compared to conventional complementary filters, this approach enhances adaptability and fault tolerance through parameter tuning [5].

***CORRESPONDING AUTHOR | Dwi Pebrianti** | ✉ dwipebrianti@iium.edu.my

Another development in satellite attitude control have focused on improving stability, robustness, and convergence time under system uncertainties and external disturbances. A fast fixed-time stability framework has been proposed to guarantee convergence within a smaller upper bound compared to conventional fixed-time methods. Traditional sliding mode controllers often require switching surfaces near the origin to avoid singularities, which may compromise performance. To overcome this, a novel nonsingular sliding surface is introduced, enabling smooth convergence to the equilibrium point without switching logic or restrictive parameter constraints [6].

Controlled reentry of small spacecraft requires precise integration of deorbit and atmospheric guidance strategies. A hybrid thruster system capable of delivering 100 N-class thrust, combined with a spin-stabilized attitude control using cold-gas thrusters, enables robust deorbit maneuvers from low Earth orbit. Numerical simulations demonstrate that this approach maintains effectiveness even under uncertainties such as center-of-gravity shifts and thrust misalignments [7].

Study done by Shafiq, et. al., explores fixed-time synchronization in chaotic spacecraft formation flying, addressing two key research gaps. First, it introduces the concept of fixed-time leader-follower spacecraft formation (LFSF) synchronization in chaotic systems—a topic previously unexplored. Second, it proposes a novel fixed-time feedback control (FTBC) law that enhances synchronization performance by ensuring faster, smoother, and more robust convergence. Unlike traditional fixed-time methods, the proposed FTBC guarantees state error convergence within a predefined time regardless of initial conditions, using Lyapunov-based stability analysis. Numerical simulations with multi-spacecraft formations validate the method's superiority in synchronizing chaotic systems, confirming its potential for precise and rapid coordination in spacecraft formation control applications [8].

To better understand and demonstrate the principles of spacecraft attitude control, particularly torque-free motion and reaction wheel dynamics, it is beneficial to develop a terrestrial replica using a two-wheeled mobile robot. This approach enables students and researchers to simulate and visualize key control algorithms—such as PID, adaptive filtering, and fixed-time stability techniques—in an accessible, hands-on environment. By mimicking satellite attitude behaviors through differential drive mechanisms, the robot serves as a practical platform for testing estimation and control strategies, ultimately enhancing comprehension of Guidance, Navigation, and Control (GNC) concepts in spacecraft operations within resource-constrained educational or research settings.

In this project, a two-wheeled Arduino-based mobile robot as shown in Figure 1 will be used to model spacecraft attitude control. By using a gyro sensor to measure the wheel's angular speed and motion, the torque-free motion and dynamics of reaction wheels can be mimicked and analyzed. Furthermore, this study will incorporate PID control, which is widely used, in order to enhance its stability and ensure the output accuracy [9].

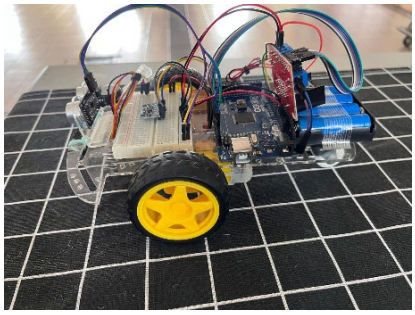

Figure 1. Two-wheeled Arduino-based mobile robot

PID control is a widely used control strategy in spacecraft attitude control systems due to its simplicity and effectiveness in managing complex dynamic systems. The PID controller consists of three primary components which are as follows:

a) Proportional gain (K_P)

b) Integral gain (K_I)

c) Derivative gain (K_D)

The Proportional term (K_P) adjusts the control output based on the current error between the desired and actual states, providing immediate response to deviations. The Integral term (K_I) accumulates past errors over time, helping to eliminate steady-state errors and ensure that the system reaches its desired state. The Derivative term (K_D) predicts future errors by considering the rate of change of the error, allowing for anticipatory adjustments to prevent overshooting [10]. Using MATLAB, this project will employ Particle Swarm Optimization (PSO) to determine the ideal K_P, K_I, and K_D for the robot.

Particle Swarm Optimization (PSO) is a powerful evolutionary algorithm inspired by the social behavior of bird flocking and fish schooling, first introduced in 1995 [11]. For Particle swarm operation (PSO), the search space is traversed by a population of candidate solutions known as particles. Each particle stands for a possible answer to the

optimization issue. The algorithm keeps track of both the best solution so far, known as the global best, and the best solution each individual particle finds, known as the personal best. Particles move according to their individual experience (personal best) and the swarm's collective behavior (global best). Based on these two factors, the particles modify their locations and velocities at each cycle. This cycle continues until a termination condition is met, yielding a set of optimized PID parameters that enhance the system's stability, response speed, and accuracy.

1.1 Problem Statement

Spacecraft rely on reaction wheels to control their orientation in space using the principle of angular momentum conservation. Understanding these systems can be challenging due to their complexity and lack of hands-on experience.

This project aims to simplify and simulate spacecraft attitude control using a two-wheel Arduino-based mobile robot. The robot will mimic torque-free motion by adjusting the speed and direction of its wheels, similar to how reaction wheels work in space. An MPU6050 sensor will measure angular velocity, providing feedback to the control system and change the motor speed.

1.2 Objectives

The primary objective of this study is to design and simulate a spacecraft attitude control mechanism using a two-wheeled mobile robot as a terrestrial analog. The specific objectives are as follows:

1. To develop an embedded control system based on Arduino that utilizes gyroscope sensor data to adjust the differential wheel speeds for simulating spacecraft stabilization dynamics.

2. To design and implement a PID control scheme for precise velocity regulation of the robot's wheels, effectively mimicking the behavior of spacecraft reaction wheels under torque-free conditions.

3. To simulate and evaluate spacecraft attitude dynamics by replicating angular momentum exchange through differential motor actuation, providing a cost-effective experimental platform for understanding spacecraft control strategies.

2.0 METHODS AND MATERIAL

2.1 Platform Design

In this experiment, a two wheeled Arduino based car robot configuration is created to simulate the spacecraft stabilization of the reaction wheels. The configuration of hardware used in this experiment is shown in Figure 2.

Figure 3 illustrates the placement and orientation of the MPU 6050 gyroscopic sensor module on the robotic model. The MPU 6050, a widely used 6-axis inertial measurement unit (IMU), integrates a 3-axis accelerometer and a 3-axis gyroscope, enabling precise measurement of angular velocity and acceleration.

The key aspects of the sensor placement include the positioning, directional definition, and functional significance. In case of positioning, the module is mounted at the front of the robot, ensuring minimal interference from mechanical vibrations and optimal detection of directional changes. Additionally, this placement allows the sensor to accurately capture pitch, roll, and yaw movements, which are critical for motion analysis and stabilization.

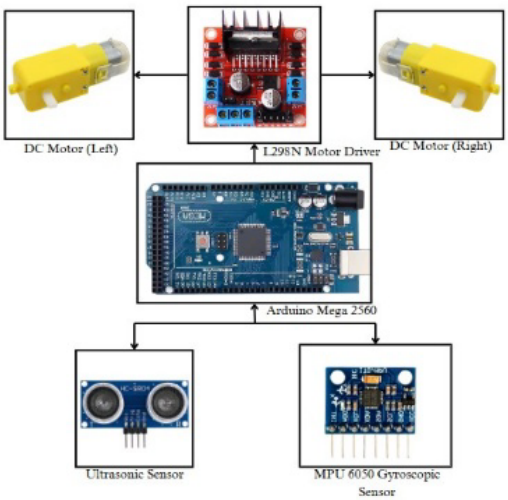

Figure 2. Two-wheel Arduino-based mobile robot configuration

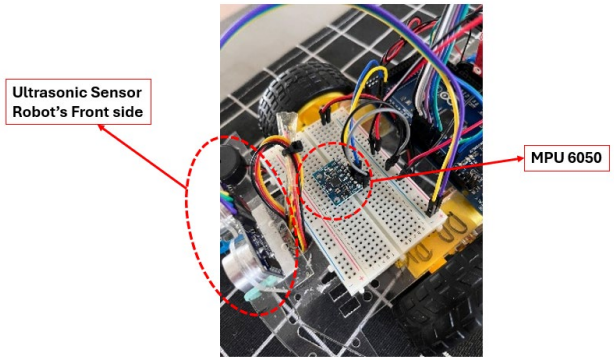

Figure 3. MPU 6050 position and definition of direction

For the directional definition, the axes (*X, Y, Z*) of the MPU 6050 are aligned with the robot's forward, lateral, and vertical directions, respectively. Proper alignment ensures that angular displacement readings correspond correctly to the robot's actual motion.

The MPU 6050 provides real-time angle change data, essential for applications such as balance control, navigation, and autonomous movement correction. Its high sensitivity and low noise characteristics make it suitable for dynamic robotic systems. These features ensure the functionality of the sensor implemented in spacecraft attitude control by using a two-wheeled mobile robot.

2.2 System Identification

To ensure success in this experiment, the experiment will be conducted with the help of MATLAB system identification tools. To use this function of MATLAB there is a need for preparation of data from the actual platform.

The reading of the testing is recorded in an excel sheet. The reading of rotation per minute (*RPM*) is taken for the input of pulse width modulation (PWM) 30, 60, 90, 120, 150, 180, 210, 230 and 255. The rotation speed reading was done by using tachometer as shown in Figure 4.

Then, the excel sheet is in the same folder as a MATLAB file that will read the excel data and tabulate into the MATLAB. The *system identification toolbox* is called in the command box using the prompt '*ident*'. Then both left and right motor data will be processed separately. The data for the left motor is inserted into the toolbox and will be illustrated as a graph. Then the transfer function is generated by changing the number of *poles* and *zeros*. The percentage of similarity and behavior is observed. Finally, the process is repeated for the right motor. The stability of the system is then checked by using step response analysis.

2.3 PID Controller Design

For this project, PID controller gains will be found using PSO. The PSO is set up with 10 particles, where each particle represents a candidate solution which is a set of PID parameters. The search space for these parameters is bound between 0 and 200, and the optimization process is set to run for 20 iterations. This particle and iteration parameter is chosen as follows to enhance its computational efficiency and the boundary parameter is in the range of 0 to 200 provides a broad enough search space to cover most practical PID values while avoiding extremely high gains that might lead to instability.

The PSO algorithm begins with the initialization of particle positions and velocities within the specified bounds. Each particle's fitness is then evaluated using a performance index, which assesses how well each set of PID parameters performs in controlling the system. The velocity of each particle is updated based on its current velocity, its personal best position (*pbest*), and the global best position (*gbest*) found by the swarm so far. Following the velocity update, the position of each particle is adjusted using the new velocity: The formula in which being implemented in the MATLAB is as follows:

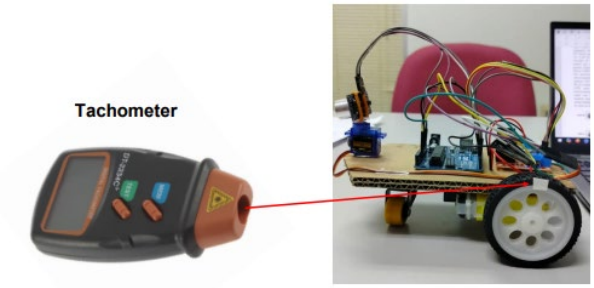

Figure 4. Digital Optical Tachometer with two-wheel robot system

$$v_i^{t+1} = v_i^t + c_1 \cdot r_1 \cdot (pbest - x_i^t) + c_2 \cdot r_2 \cdot (gbest - x_i^t)$$
$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (1)$$

where:

v_i^{t+1} is the new velocity of particle i

v_i^t is the velocity of particle i

$\boldsymbol{c_i^{t+1}}$ is the new position of particle i

$\boldsymbol{c_i^t}$ is the position of particle i

c_1 and $\boldsymbol{c_2}$ are acceleration coefficient

$\boldsymbol{r_1}$ and $\boldsymbol{r_2}$are random parameters between 0 and 1

$pbest$ is the s the personal best position of particle i

$gbest$ is the global best position found by the swarm so far.

By using PSO to optimize PID parameters, systems can achieve improved stability, response speed, and accuracy, making it a valuable tool in controlling engineering applications.

3.0 RESULTS AND DISCUSSION

3.1 Modelling

As shown in Figure 1, the final platform of test has been completed before the project is done. The platform only uses an ultrasonic sensor which helps determine the distance of the robot to an obstacle. It is also installed with a motor module L298N which allows the Arduino to split the signal into two motors. Lastly, it was provided with an MPU 6050 which is a gyroscopic sensor which will be used to calculate the angle that the robot is currently moving.

3.2 Gyroscopic Application

The MPU 6050 gyroscope requires precise calibration to eliminate inherent biases in its angular velocity measurements. During calibration, the robot is held stationary while the Arduino IDE software records raw sensor outputs. Offsets for each axis (*roll, pitch, yaw*) are iteratively adjusted until the serial monitor displays near-zero values at rest. This process ensures the gyroscope's readings reflect true orientation changes, free from drift or initial bias. Accurate calibration is foundational for the subsequent system identification (Section 3.3) and PID tuning (Sections 3.4–3.5), as uncalibrated data would introduce errors in motor control, compromising the robot's ability to simulate spacecraft attitude dynamics (Section 3.6). The calibrated gyroscope thus serves as the primary feedback source for the closed-loop control system, enabling real-time adjustments to wheel speeds based on yaw angle deviations.

3.3 System Identification

Table 1. Tabulated data for RPM of left and right motors for given PWM signal

PWM Signal	Left Motor (rpm)	Right Motor (rpm)
30	115.9	81.4
60	301.9	259.6
90	394.2	355.2
120	434.4	402.5
150	463.3	427
180	467.5	425.7
210	469.6	433.9
230	489.2	441.2
255	480.6	463.5

To model the dynamic behavior of the motors used in the two-wheeled mobile robot, system identification was performed using MATLAB's System Identification Toolbox. Experimental data were collected by applying various PWM input values (ranging from 30 to 255) and recording the corresponding RPM outputs for both the left and right motors. The data is tabulated in Table 1. The input-output data were stored in an Excel file and imported into MATLAB. Using

the *ident* command, the toolbox was initialized to generate a transfer function model for each motor. By adjusting the number of poles and zeros, a transfer function was identified for each motor with a fit accuracy exceeding **90%**. These models were validated using step response analysis to assess system stability, and they served as the basis for subsequent PID controller tuning.

3.4 Left Motor PID Controller Design

The transfer function is obtained using MATLAB system identification toolbox. The transfer function to stabilize the system is taken around 93% of fit estimation. The equation is as follows:

$$\frac{0.04449}{s^2 + 0.7181s + 0.006071} \tag{2}$$

To check the stability of the initial system without the PID controller, a step input analysis is done. The result is shown in Figure 5. The detailed system performance is tabulated in Table 2.

The identified transfer function model for the left motor exhibits stable but suboptimal performance in response to a unit step input. The system achieves a steady-state gain of approximately 0.87, indicating a 13% steady-state error. It shows no overshoot and a smooth, overdamped response, with a rise time of about 40–50 seconds and a settling time close to 70 seconds. While the response confirms system stability, the slow dynamics and residual error highlight the need for closed-loop control—such as a well-tuned PID controller—to improve tracking accuracy, reduce steady-state error, and enhance overall responsiveness for attitude control applications.

Table 2. Step response information of the graph before PID implementation of the left motor

Time Response	Values
Rise Time	28.1302
Transient Time	51.1096
Settling Time	51.1096
Settling Min	0.7925
Settling Max	0.8786
Overshoot	0
Undershoot	0
Peak	0.8786
Peak Time	83.4590

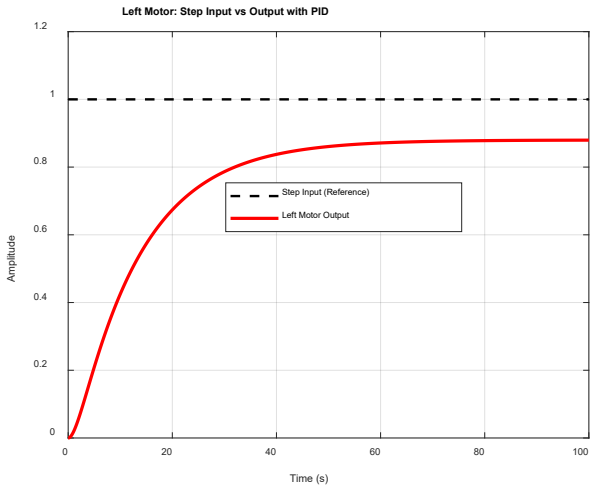

Figure 5. Step response analysis before PID implementation for left motor

To improve the response accuracy and reduce the steady-state error observed in the open-loop system, a PID controller was implemented. The PID control gains were optimized using the Particle Swarm Optimization (PSO) algorithm, which efficiently searched for the best combination of proportional, integral, and derivative gains to minimize the tracking error. The objective function for the PSO algorithm was based on minimizing the integral of the time-weighted absolute error (ITAE), ensuring faster settling time and smoother response. The optimized PID gains for both the left and right motors are summarized in Table 3.

Table 3. Left motor PID information

Controller Parameters	Values
Proportional (K_P)	175.9733
Integral (K_I)	48.3574
Differential (K_D)	188.6340

The step response of the left motor after PID implementation in Figure 6 shows significant improvement in system performance. As tabulated in Table 4, the rise time was reduced to 0.24 seconds, and the system reached steady-state conditions with a settling time of only 2.88 seconds, demonstrating a much faster response compared to the open-loop configuration. The peak value of 1.03 indicates a small overshoot of 2.77%, which is acceptable in practical control systems. No undershoot was observed, and the system quickly stabilized within the acceptable range defined by the settling limits. These results confirm that the PID controller, optimized using PSO, effectively enhanced the response speed, accuracy, and stability of the left motor's dynamics.

Table 4. Step response information of the graph after PID implementation of the left motor

Time Response	Values
Rise Time	0.2433
Transient Time	2.8829
Settling Time	2.8829
Settling Min	0.9030
Settling Max	1.0277
Overshoot	2.7685
Undershoot	0
Peak	1.0277
Peak Time	1.0746

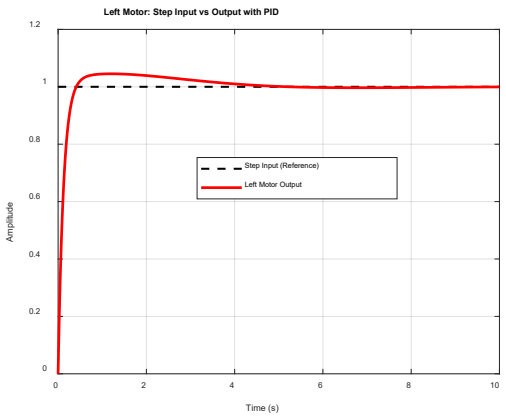

Figure 6. Step response analysis of the left motor after PID implementation

3.5 Right Motor PID Controller Design

The transfer function to stabilize the system is taken around 97% of fit estimation. The equation is as follows:

$$\frac{0.3144}{s^2 + 0.4543s + 0.1197} \tag{2}$$

To check the stability of the initial system without the PID controller, a step input analysis is conducted. The step response of the right motor prior to PID implementation shown in Figure 7, reveals a sluggish and oscillatory system behavior. Table 5 shows the time response analysis of the right motor before the PID implementation. The rise time is relatively long at 2.10 seconds, and the system requires approximately 16.73 seconds to settle, indicating a slow transient response. A significant overshoot of 31.54% is observed, peaking at 0.95, which reflects poor damping and potential instability. Furthermore, the presence of visible oscillations before stabilization suggests the system is underdamped. With a steady-state value lower than the reference input, the system also exhibits a noticeable steady-state error. These characteristics highlight the necessity for controller tuning to improve accuracy, speed, and stability in the motor's performance.

Table 5. Step response information of the graph before PID implementation of the right motor

Time Responses	Values
Rise Time	2.1029
Transient Time	16.7333
Settling Time	16.7333
Settling Min	0.6522
Settling Max	0.9527
Overshoot	31.5415
Undershoot	0
Peak	0.9527
Peak Time	5.0684

The PID controller gains for the right motor, as listed in Table 6, were determined using the Particle Swarm Optimization (PSO) algorithm. PSO was employed to optimize the proportional (K_p), integral (K_i), and derivative (K_D)

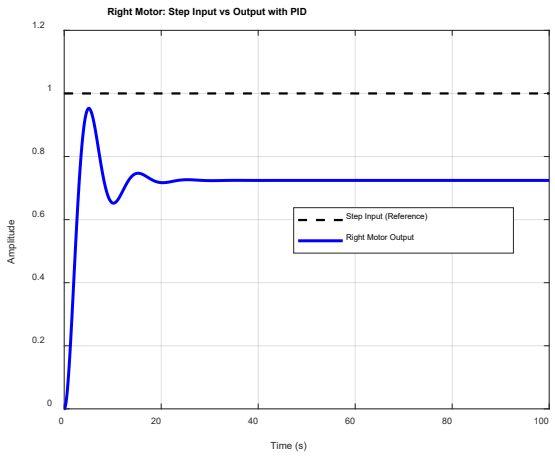

Figure 7. Step response analysis before PID implementation for right motor

gains by minimizing an objective function based on the system's time-domain response characteristics, such as rise time, overshoot, and settling time. This method allowed for an efficient and robust search of the optimal gain values that significantly improved the dynamic performance of the system. The resulting controller enhances stability, reduces overshoot, and accelerates convergence to the desired reference input.

Table 6. Right motor PID information

Controllers Gain	Values
Proportional (K_P)	183
Integral (K_I)	97.1
Differential (K_D)	187

The PID controller significantly enhanced the dynamic performance of the right motor system. As shown in Figure 8, the motor output closely tracks the reference input with minimal delay and negligible oscillations. The rise time was reduced drastically to 0.06 seconds, while the settling time was improved to just 0.12 seconds, compared to over 16 seconds in the open-loop case. The overshoot was minimized to 2.50%, indicating a well-damped response. No undershoot was observed, and the peak amplitude remained within acceptable limits. These results listed in Table 7 confirm that the PID controller—optimized via PSO—successfully improved response speed, accuracy, and overall system stability.

Table 7. Step response information of the graph after PID implementation of the right motor

Time Response	Values
Rise Time	0.0580
Transient Time	0.1203
Settling Time	0.1203
Settling Min	0.9050
Settling Max	1.0250
Overshoot	2.4984
Undershoot	0
Peak	1.0250
Peak Time	0.4030

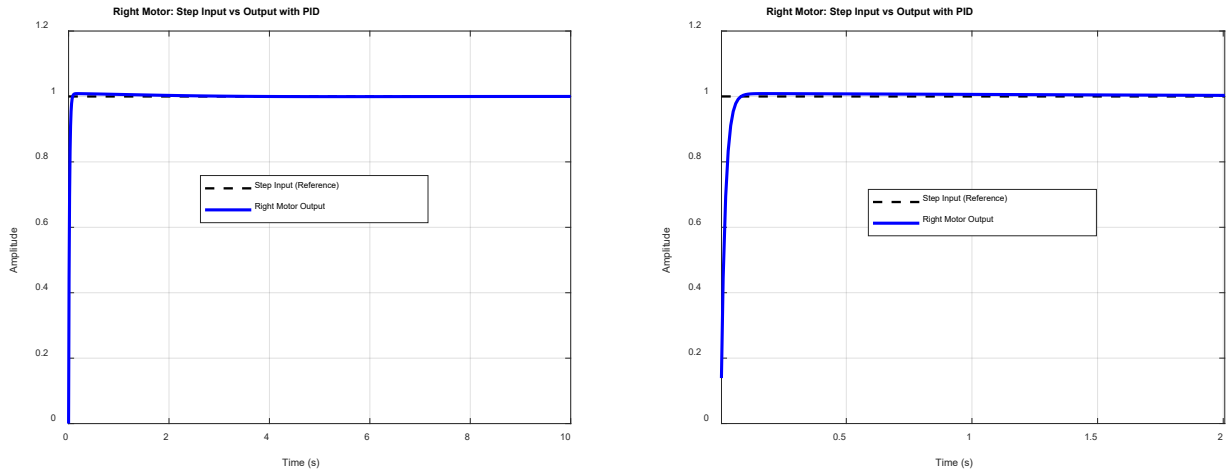

Figure 8. Step response analysis of the right motor after PID implementation up to (a) 10 seconds (b) 2 seconds of simulation

3.6 Reaction Wheel Simulation by Model

To test the application of both the ultrasonic sensor and gyroscope, a test has been done which the angle that needs to be maintained is the yaw is always at zero. Theoretically, when each motor rotates about the same speed the robot will moves in a straight line in which the yaw is zero. However, due to the uneven surface of track the robot will tend to have different values of yawing angle.

When the robot is facing a negative value of yaw angle, which is shown in Figure 9. The Arduino will send the signal to slow down the left motor and speed up the right motor, Figure 9 (a).

The same concept is applied when the robot is facing a positive yaw. However, the reaction of the motor will be reversed in which the left motor will increase in rpm and the right motor will slow down, Figure 9(b).

(a) (b)

Figure 9. (a) Robot tilted to the right **(b)** Robot tilted to the left

4.0 CONCLUSION

To conclude this study, we successfully mimicked attitude control of the spacecraft using the Two-Wheeled robot.

Modeling and control of a two-wheel robot using both open-loop and closed-loop systems. System identification techniques were employed to derive accurate transfer functions for the motors, achieving high fit percentages, which formed the foundation for controller design. The PID controller, tuned through the Particle Swarm Optimization (PSO) method, provided an effective solution for enhancing system stability and reducing steady-state error to less than 10%. While initial instability was observed in the closed-loop system due to the PID controller, the system ultimately stabilized, achieving the desired performance metrics, including minimal overshoot and precise response.

This study highlights the importance of integrating theoretical modeling, simulation, and real-world hardware implementation to address challenges in control systems. The findings not only demonstrate the capability of PID controllers in robotics but also underline the broader relevance of advanced motor control in applications such as automation, industrial machinery, and aerospace systems. These results contribute to the ongoing development of robust control strategies for improving system stability and performance in dynamic environments.

6.0 REFERENCES

[1] Wertz, J. R. (Ed.). (2012). Spacecraft attitude determination and control (Vol. 73). Springer Science & Business Media.

[2] Marcel J. Sidi, “Cambridge Aerospace Series 7 General editors Spacecraft Dynamics and Control,” 1997. doi: https://doi.org/10.1017/CBO9780511815652.

[3] T. S. Abdel Aziz, G. I. Salama, M. S. Mohamed, and S. Hussein, “Efficient machine learning based techniques for fault detection and identification in spacecraft reaction wheel,” *Aerospace Systems*, Dec. 2024, doi: 10.1007/s42401-024-00322-0.

[4] M. Bassetto, G. Mengali, and A. A. Quarta, “Drag sail attitude tracking via nonlinear control,” *Acta Astronaut*, vol. 225, pp. 845–856, Dec. 2024, doi: 10.1016/j.actaastro.2024.09.046.

[5] W. Wu, Z. Jin, and A. Zeya, “Attitude estimation using an adaptive generalized complementary filter,” *Measurement*, vol. 251, p. 117265, 2025, doi: https://doi.org/10.1016/j.measurement.2025.117265.

[6] S. Barman and M. Sinha, “A Novel Nonsingular Fast Fixed-Time Sliding Mode Control and Its Application to Satellite Attitude Control,” *International Journal of Robust and Nonlinear Control*, vol. 35, pp. 2184–2198, 2025, doi: https://doi.org/10.1016/j.ast.2024.109746.

[7] T. Saito, T. Kuwahara, Y. Saito, and Y. Sato, “Guidance strategies for controlled Earth reentry of small spacecraft in low Earth orbit,” *Acta Astronaut*, vol. 229, pp. 684–697, 2025, doi: https://doi.org/10.1016/j.actaastro.2024.12.054.
[8] M. Shafiq and I. Ahmad, “A novel fixed-time smooth synchronization controller for stabilizing chaotic spacecraft formation,” *Alexandria Engineering Journal*, vol. 117, pp. 577–592, 2025, doi: https://doi.org/10.1016/j.aej.2024.12.076.
[9] M. Seddighi and M. Jafari-Nadoushan, “Designing a concurrent detumbling and redirection mission for asteroid mining purposes via optimization,” *Astrodynamics*, Dec. 2024, doi: 10.1007/s42064-024-0213-9.
[10] B. C. . Kuo and Farid. Golnaraghi, *Automatic control systems*. John Wiley & Sons, 2003.
[11] Kennedy, J. and Eberhart, R. (1995) Particle Swarm Optimization. Proceedings of the IEEE International Conference on Neural Networks, 4, 1942-1948.