International Journal of Artificial Intelligence & Robotics (IJAIR)
Vol.7, No.2, 2025, pp.72-82
E - ISSN : 2686-6269

Design and Implementation of a PID Controller for a Two-Axis Gimbal System Using ESP32

Dimas Cahyono[1], Anang Habibi[2*], Aina Afrina Fairuziya[3], Wahyu Caesarendra[4]
[1,2,3]Electrical Engineering Department, Universitas Islam Malang, Indonesia
[4] Faculty of Engineering Technology and Science, Higher Colleges of Technology, Ras Al Khaimah, United Arab Emirates
[1]22201053012@unisma.ac.id; [2]ananghabibi@unisma.ac.id*; [3]22201053032@unisma.ac.id; [4]w.caesarendra@curtin.edu.my
*corresponding author

ABSTRACT

This study presents the design and implementation of a two-axis smartphone camera stabilizer system that uses an ESP32 microcontroller with a Proportional–Integral–Derivative (PID) control algorithm. The system is designed to maintain camera stability by minimizing unwanted vibrations and angular deviations on the roll and pitch axes. The hardware configuration consists of an ESP32 as the main control unit, an MPU6050 sensor for real-time orientation measurement, and two servo motors functioning as actuators for both axes. The control process operates in a closed loop, with the PID algorithm continuously using feedback from the MPU6050 sensor to correct the camera's orientation relative to the 0° setpoint. Experimental testing was carried out over a 40-second interval under four load conditions—no load, and additional weights of 186 g, 204 g, and 228 g—while manual disturbances were applied to simulate dynamic movement. The PID parameters (Kp, Ki, Kd) were tuned using a trial-and-error approach to achieve optimal response speed, stability, and steady-state accuracy. The best performance was obtained with parameter values of Kp = 1.3, Ki = 0.05, and Kd = 0, which produced the lowest accumulated error and sTable motion across all load variations. The corresponding settling times were 2.2 s (no load), 3.2 s (186 g), 4.2 s (204 g), and 4.5 s (228 g), indicating that increased load inertia slightly extended the stabilization period. Overall, the results indicate that the proposed ESP32-based two-axis gimbal system with PID control provides an effective, low-cost, and reliable stabilization solution suiTable for porTable videography, robotics, and drone applications.

Keywords: PID Controller; Camera Stabilizer; ESP32; MPU6050; PID; Voltage Sensor.

Article History		
Received : July 23rd 2025	Accepted: November, 3rd 2025	Published: November, 20th 2025

I. INTRODUCTION

Technological advancements in embedded systems and automatic control have significantly impacted the development of camera stabilization devices. All aspects of human life today are also influenced by rapid technological progress, and cameras are among the technologies that have advanced considerably [1]. In practice, image stability is heavily influenced by the system's ability to suppress unwanted vibrations or movements. In videography, gimbals have become a crucial tool for every videographer to produce high-quality images and videos [2].

To improve the quality of aerial photos and videos, a system is needed to maintain the camera's position and reduce vibrations caused by the quadcopter's motion [3]. A camera stabilizer system must have at least two axes—roll and pitch—to achieve good stability [4]. One effective approach to address this challenge is to use a PID (Proportional-Integral-Derivative) control system, which can provide a smooth, controlled response to changes in orientation. PID remains the most widely used controller due to its simplicity and robustness in dealing with nonlinearities in drone and gimbal systems [5].

Two-axis camera stabilizers (2-axis gimbals) are increasingly used for their compact size and ability to stabilize horizontal and vertical movements [6]. To support such performance, a reliable microcontroller system and accurate motion sensors are required. The ESP32 has become one of the most popular platforms due to its high processing speed, wireless connectivity, and compatibility with various sensors such as the MPU6050. Two-axis gimbals are commonly used for their ability to stabilize horizontal (roll) and vertical (pitch) movements. PID control implementation in gimbals has been extensively studied. Final tuning results for the roll axis yielded PID constants of Kp = 0.37, Ki = 0.01, and Kd = 0.29, with an average response time of 0.8 seconds. For the pitch axis, the constants were Kp = 0.55, Ki = 0.01, and Kd = 0.29, with an average response time of 0.7 seconds [7].

These results demonstrate the potential of PID methods in maintaining mechanical structure stability. IMU sensors, such as the MPU6050, are widely used to detect pitch and roll angles in real time. Previous research has demonstrated that implementing PID control on a quadcopter using the MPU6050 sensor yields minimal overshoot and excellent aerodynamic stability. Through balance testing on a quadcopter using a PID controller with parameters obtained through trial and error (Kp = 1.3, Ki = 0.05, and Kd = 15),

DOI: http://dx.doi.org/10.25139/ijair.v7i1.10771

the resulting balance response from the IMU was very fast and accurate [8]. Using the ESP32 as the core of the control system provides flexibility in sensor signal processing and digital PID tuning.

Additionally, integration with an I2C LCD screen enables real-time monitoring of system parameters, which is crucial during the tuning and evaluation process. The ESP32, when operated in DualCore mode, performs more optimally than in SingleCore mode, with an average maximum loop time per cycle recorded at 75.3 ms, whereas SingleCore operation recorded 88.9 ms [9]. The ESP32 microcontroller serves as the data processing core due to its WiFi and Bluetooth connectivity, which support real-time data transmission, and offer higher processing performance than its predecessor, the ESP8266 [22]. The combination of PID control and the ESP32 platform promises a fast and affordable camera stabilization system.

This study aims to design and implement a two-axis smartphone camera stabilizer system based on the ESP32 microcontroller, controlled using a PID algorithm. The primary focus of the research is to evaluate the system's performance in maintaining pitch and roll stability and to analyze the effect of PID parameters on stabilization quality. The results of this study are expected to serve as a foundation for developing effective, efficient, and affordable gimbal systems for everyday videography applications.

II. METHOD

This chapter outlines the methodological stages involved in designing and implementing a two-axis camera stabilizer system based on the ESP32 microcontroller. The process includes mechanical design, electronic circuit development, software implementation, system operation principles, PID parameter testing, and error data acquisition.

A. Camera Gimbal Mechanism Design

The mechanical design of the two-axis camera gimbal is presented from two perspectives: a side view and a top view, as shown in Fig.1. The main structure consists of two servo motors (servo X and servo Y), arranged perpendicularly to each other, enabling independent control of the roll and pitch axes. The smartphone is mounted on a holder directly connected to servo Y, while servo X supports the entire upper frame structure.

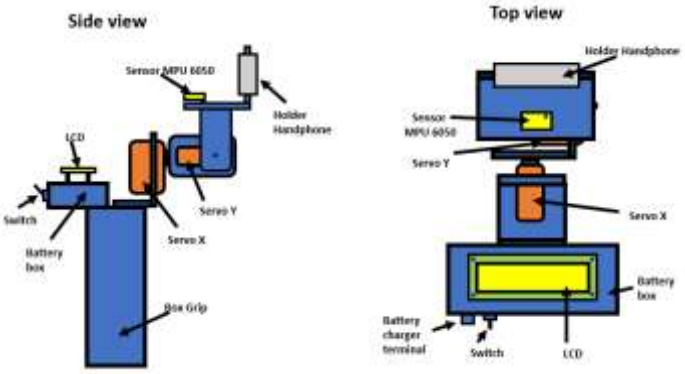

Fig. 1. Mechanical Scheme of the Gimbal Mechanism

The MPU6050 sensor is positioned close to the smartphone holder to provide more accurate angle readings that correspond to the camera's orientation. The bottom section features a handheld grip box that also serves as a battery compartment, housing the main power switch that activates the system. Additional components, such as an I2C LCD, are placed on top of the battery box to display angle readings or PID parameter values. A battery charging terminal is also provided on the exterior side of the box for convenient charging without disassembling the system.

Fig. 2 Gimbal Visualization

DOI: http://dx.doi.org/10.25139/ijair.v7i1.10771

Fig.2 for the prototype of a two-axis gimbal system based on the ESP32, designed to stabilize the camera along the roll and pitch axes. The system employs two servo motors as actuators and an MPU6050 sensor to detect real-time angular motion. The ESP32 microcontroller functions as the central controller, processing sensor data and adjusting servo movements using a PID control algorithm. All components are compactly assembled onto a lightweight, rigid frame to ensure stability and minimize vibration during operation.

B. Hardware Design

The hardware system of the two-axis gimbal stabilizer consists of several main components, including:

1) ESP32: The main advantages of this microcontroller are its relatively low cost, ease of programming, a sufficient number of I/O pins, and an integrated WiFi adapter for Internet connectivity [10]. Table I describes the specifications of the ESP32 microcontroller.

TABLE I
ESP32 SPECIFICATIONS

Category	Specification
Processor	Dual-core Tensilica LX6 (up to 240 MHz)
Architecture	32-bit RISC
RAM	520 KB SRAM
Flash Memory	Up to 16 MB (depending on the module/board)
WiFi	802.11 b/g/n (2.4 GHz)
Bluetooth	v4.2 BR/EDR and BLE (Bluetooth Low Energy)
GPIO	34 GPIO pins
ADC	12-bit, 18 channels
DAC	2 pins, 8-bit
PWM	16 channels

2) MPU6050: The MPU6050 is a versatile Inertial Measurement Unit (IMU) sensor that integrates a 3-axis accelerometer and a 3-axis gyroscope [11]. Table II describes the specifications of the MPU6050 module. The MPU6050 sensor communicates with the ESP32 via the I2C protocol, which uses two main lines, SCL and SDA, typically connected to GPIO 22 and GPIO 21 on the ESP32. Using the Wire.h and Adafruit_MPU6050.h libraries in the Arduino IDE, the ESP32 initializes the sensor and reads accelerometer and gyroscope data. These values are then processed to obtain pitch and roll angles, which the PID algorithm uses to control the servo motors and maintain gimbal stability.

TABLE II
MPU6050 MODULE SPECIFICATIONS

Category	Specification
Power Supply	3.3 – 5V
Chipset	MPU-6050
Interface	I2C
Pins	VCC, GND, SCL, SDA, AD0, and INT
Acceleration Range	±2g, ±4g, ±6g, ±8g, ±16g

3) MG995 Servo Motor: A servo is an actuator used to control position, speed, and acceleration with high precision [12]. However, in previous research, the servo angle resulted in an average error of 12.68%. The servo caused this error due to a significant difference between the input angle and the robot's angle [21]. Table III describes the specifications of the MG995 servo.

TABLE III
MG995 SERVO SPECIFICATIONS

Category	Specification
Power Supply	4.8 – 7.2V
Current	8.8 mA at 4.8V, 9.1 mA at 6V
Stall Current	350 mA at 4.8V, 450 mA at 6V
Speed	0.2 s / 60º (4.8V), 0.16 s / 60º (6V)
Torque Load	8.5 kgf·cm (4.8V), 10 kgf·cm (6V)

DOI: http://dx.doi.org/10.25139/ijair.v7i1.10771

The diagram in Fig. 3 illustrates the connections between the system's main components, where the ESP32 serves as the central controller, connected to the MPU6050 sensor and the I2C LCD via the I2C communication line. The two servo motors are driven by the ESP32's PWM pins, based on the PID algorithm's output [13]. The system is powered by a Li-ion battery, connected through a switch and a step-down voltage regulator module. A charging terminal is also provided for convenient battery recharging. This diagram illustrates a straightforward yet effective integration of components to support the two-axis gimbal's functionality. Additionally, the system features a voltage sensor that monitors the battery level in real-time, ensuring safe and efficient system operation [14].

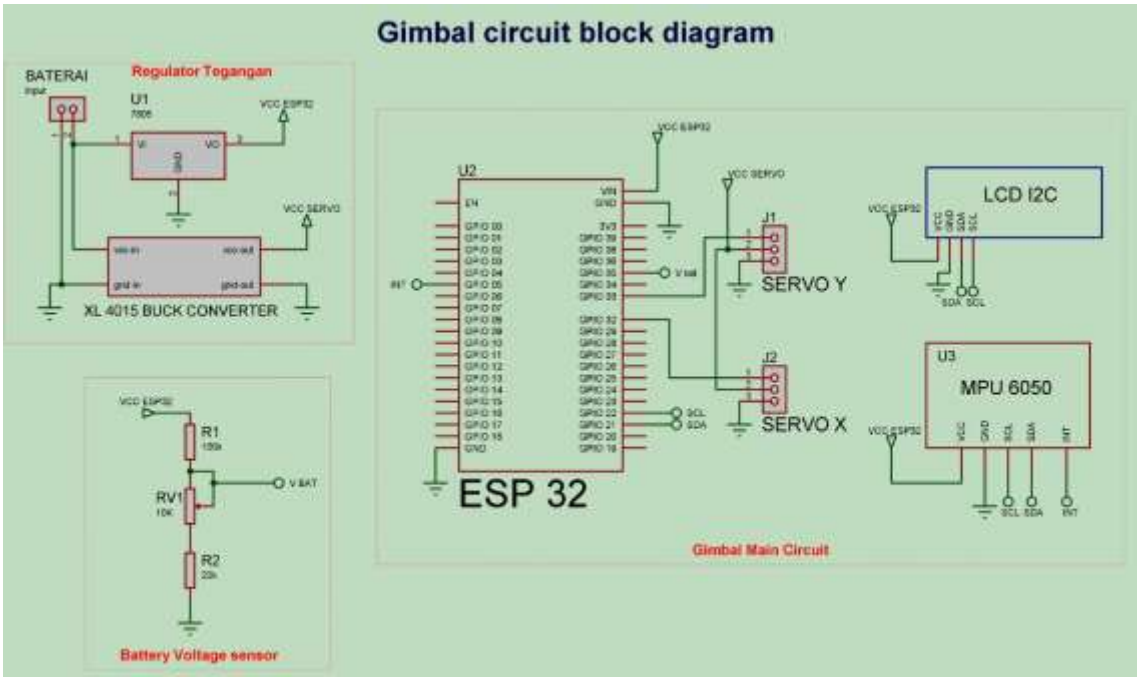

Fig. 3 Block Diagram of the System

C. *Software Design*

The software used to design this system is the Arduino IDE, which uses the C programming language. The software design of this system utilizes the Arduino IDE due to its ease of use and compatibility with the ESP32 microcontroller. The program is written in C/C++, enabling the control of the MPU6050 sensor, execution of the PID algorithm, and adjustment of servo motors through PWM signals. The Arduino IDE also supports various libraries such as Wire.h and Servo.h for I2C communication and actuator control. As an open-source platform with extensive community support, the Arduino IDE streamlines the development and real-time testing of systems.

D. *PID Calculation*

The control method implemented is the PID (Proportional–Integral–Derivative) controller, which consists of three main elements: proportional, integral, and derivative. PID control is a feedback-based method that uses a PID controller as its core component [15]. The PID controller generates a control signal by comparing the error, which is the difference between the process variable and the setpoint. The following is Equation (1) for PID.

$$u(t) = Kp.e(t) + Ki\int_0^t e(t) + Kd\frac{de(t)}{dt} \quad (1)$$

The PID equation describes that the controller output signal is determined by three primary parameters: the proportional constant Kp, which responds to the present error; the integral constant Ki, which accumulates error over time to eliminate steady-state error; and the derivative constant Kd, which reacts to the rate of change of the error to anticipate future variations. The combination of these parameters allows the PID controller to produce a fast, accurate, and sTable system response to changes in the setpoint or

E. *System Workflow*

The diagram in Fig. 4 illustrates the workflow of a two-axis camera stabilizer system using a PID algorithm. The system begins with a setpoint value representing the desired ideal angle of 0 degrees. This setpoint is then compared to the actual angle measured by the MPU6050 sensor on the X-axis (roll) and Y-axis (pitch). The difference between the setpoint and the actual angle on each axis produces an error value, which is then sent to the ESP32 microcontroller [16].

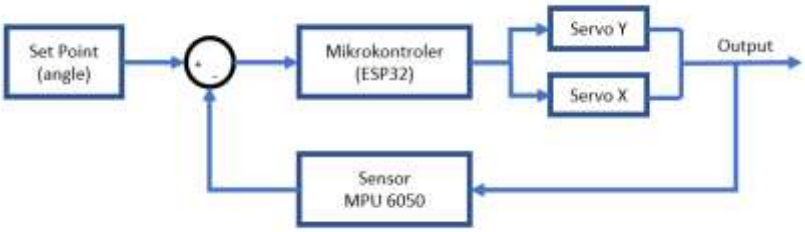

Fig. 4 System Workflow Diagram

In the microcontroller, control is carried out using two independent PID algorithms—one for the X-axis and one for the Y-axis. Consequently, the X and Y servos do not move to identical angles, as each servo is controlled by the PID computation results for its respective axis. The PID controller on the X-axis adjusts the roll angle, while the PID controller on the Y-axis adjusts the pitch angle. The angle changes caused by motor movement are read again by the MPU6050 sensor, forming a closed-loop system that enables continuous adjustment to maintain the camera's orientation stability [17]

F. PID Tunning

The tuning process of the PID (Proportional–Integral–Derivative) parameters was carried out gradually to obtain a combination of Kp, Ki, and Kd values that produced the most sTable and accurate system response described in Fig.5. In the initial stage, the parameters were manually adjusted using a trial-and-error approach based on the system's dynamic characteristics.

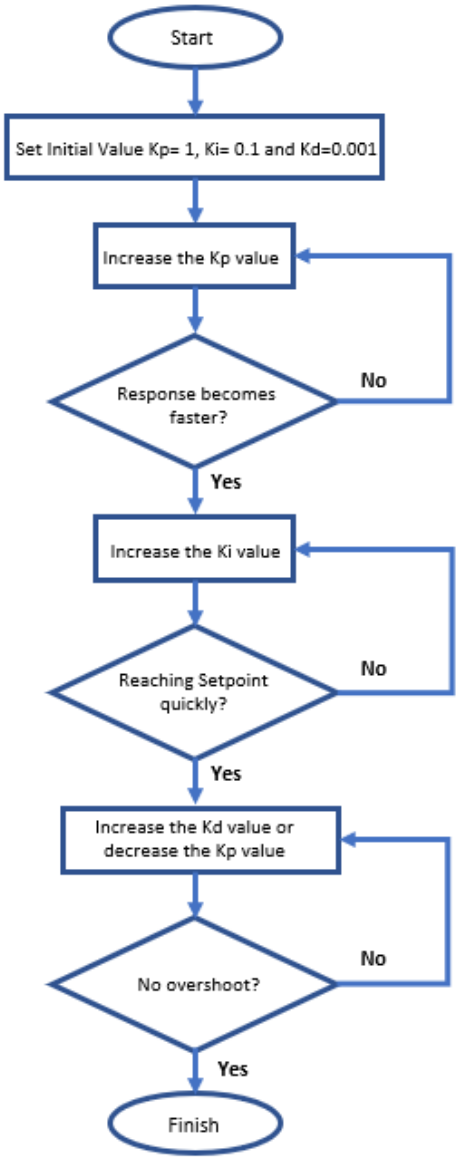

Fig. 5 PID Tuning Flowchart

The first step was to assign initial guess values: Kp was set to a small value, while Ki and Kd were set to zero. This setup aimed to observe how the system responded solely to the proportional action. If the system showed a slow response or failed to reach the setpoint, the Kp value was gradually increased. This adjustment continued until the system exhibited a fast response while maintaining stability without excessive oscillations. However, if the Kp value was too high, the system tended to oscillate or even become unsTable due to increased sensitivity to errors.

Once the optimal Kp value was determined, the next step was to introduce a small Ki value to correct the steady-state error. The Ki parameter serves to eliminate accumulated errors, ensuring the system output precisely reaches the setpoint. The Ki value was increased gradually until the system reached the setpoint more quickly. However, if Ki was increased excessively, the system could experience significant overshoot and instability.

The next stage involved adjusting the Kd parameter, which functions to dampen oscillations and smooth the system response. The Kd value was gradually increased to reduce overshoot caused by the addition of Ki and to accelerate the system's stabilization process. However, if an increase in Kd caused the system to oscillate again or respond sluggishly, the Kp or Kd value had to be reduced until a proper balance was achieved.

Overall, the tuning process was conducted by considering a balance between response speed, stability, and system accuracy. The final values of Kp, Ki, and Kd were determined from observations of the system's response to setpoint changes, including rise time, overshoot, settling time, and steady-state error. Through this approach, a combination of PID parameters was obtained that enabled the two-axis gimbal system based on the ESP32 microcontroller to achieve sTable, responsive, and highly precise performance. Increase the Kd value until the overshoot is reduced and the system approaches the setpoint. If increasing the Kd value makes the system more unsTable, then reduce the Kp value until the overshoot approaches the setpoint [18].

G. Error Data Collection Method

System accuracy testing is performed by running the gimbal continuously for 40 seconds, during which the actual angle from the MPU6050 sensor is read and compared in real-time with a setpoint of 0° as the sTable position. The angle difference is calculated as the error at each iteration and then averaged to obtain the accumulated error on the pitch and roll axes. The test is conducted under four conditions: without load, and with smartphone loads of 186 grams, 204 grams, and 228 grams, to assess the system's stability and consistency under varying weights. During testing, external disturbances are applied manually through movements to simulate real-world conditions, allowing evaluation of how quickly and accurately the system returns to a sTable position [19]. This approach provides a comprehensive overview of the PID algorithm's performance in handling dynamic disturbances.

III. RESULT AND DISCUSSION

A. Analysis of Battery Voltage Reading (Voltmeter)

Table IV presents the comparison results of voltage measurements between the ESP32-based voltage divider and a reference measuring device (an avometer). The test was conducted over a voltage range of 3.5 V to 9.3 V. The results indicate a discrepancy between the ESP32 readings and the avometer, with the smallest error of 0.06 V and the largest of 0.76 V. The average error value of 0.343 V remains within the accepTable tolerance range for simple battery monitoring applications.

TABLE IV
ACCUMULATED VOLTAGE SENSOR ERROR

No	Source Voltage (V)	Volt meter Esp32 (V)	AVO meter (V)	Error V
1	3,5	3,5	3,56	0,06
2	4,5	4,65	4,52	0,13
3	5,5	5,78	5,51	0,27
4	6,5	6,79	6,52	0,27
5	7,5	8,1	7,53	0,57
6	8,5	9,3	8,54	0,76
Average				0,343

B. Analysis of PID Parameter Testing Results

Table V shows the accumulated angle error on the pitch and roll axes of the two-axis camera stabilizer system. The testing was carried out by varying the PID values (Kp, Ki, Kd) and the smartphone load (none, 186 grams, 204 grams, and 228 grams) to observe their effect on the system's ability to reach the 0° setpoint.

TABLE V
ACCUMULATED ERROR IN PID PARAMETER TESTING

No	Parameter PID			Accumulated Errors							
				No load		Handphone (186 gram)		Handphone (204 gram)		Handphone (228 gram)	
	Kp	Ki	Kd	Roll	Pitch	Roll	Pitch	Roll	Pitch	Roll	Pitch
1.	0,5	0	0	10,8	12	12,3	12,3	11,2	11,1	10,6	9,4
2.	0,7	0	0	10,1	8,6	8,2	9,1	9,2	10,3	10	11,8
3.	0,9	0	0	6,5	7,1	11,3	9,4	8,8	9	8,9	9,1
4.	1	0	0	7,5	6	9,7	9,4	9,5	9,3	9,9	9,4
5.	1,2	0	0	7,9	6,4	10,8	9,6	8,6	9,4	8,9	9,4

DOI: http://dx.doi.org/10.25139/ijair.v7i1.10771

No	Parameter PID			Accumulated Errors							
				No load		Handphone (186 gram)		Handphone (204 gram)		Handphone (228 gram)	
	Kp	Ki	Kd	Roll	Pitch	Roll	Pitch	Roll	Pitch	Roll	Pitch
6.	1,2	0,02	0	5,4	6,4	7,7	7,5	6,5	7,3	6,6	7,4
7.	1,2	0,05	0	3,8	5,2	4,6	6,5	4,5	6,4	4,3	5,8
8.	1,1	0,01	0	8,6	7,6	7,1	9	7,1	9,4	9,3	8,2
9.	1,3	0,05	0	4	3,6	4,2	5,3	4,7	5,8	4	5,8
10.	1,3	0,04	0,001	4	4,6	4,9	6,1	4,9	7,5	4,8	5,6
11.	1	0,03	0,001	4,2	4,3	5,7	5,7	5,5	6,5	5,7	6
12.	1	0,05	0,001	4,5	6	4,5	4,4	4,6	4,5	3,4	4,8
13.	1,2	0,05	0,001	4,1	5,8	3,3	6,1	3,6	5	4	4,4
14.	1,3	0,06	0,002	5	5	3,3	4,7	3,9	5,5	3,3	4,3
15.	1,3	0,02	0,002	6,5	6,7	5,7	7,7	6,7	7,1	6,1	6,4

In the initial tests (rows 1 and 2), using a low Kp value caused the system to fail to reach the setpoint, as indicated by the high accumulated errors across all load variations. For example, in the first row with Kp = 0.5, the pitch value was 12 and the roll value was 10.8 under no load. This condition indicates that the corrective force provided by the controller was too weak, preventing the actuator from generating sufficient torque to move the gimbal toward a sTable position. As a result, the system responded slowly to changes and was unable to maintain the desired angular stability.

Conversely, in tests with excessively high Kp values (rows 14 and 15), the system exhibited oscillatory behavior, resulting in increased accumulated error. This was especially evident in tests using a smartphone load, where in row 15 (Kp = 1.3; Ki = 0.02; Kd = 0.002), the pitch value reached 7.7 and the roll value 5.7 for a 186-gram load, increasing further to a pitch of 7.1 and roll of 6.1 for a 228-gram load. This oscillation occurred because the system became overly sensitive to small errors, leading to an exaggerated response and dynamic instability.

The most optimal performance was obtained when Kp was set to a moderate level, within the range of rows 4 to 10, where the system demonstrated a good balance between response speed and stability. Within this range, the accumulated error was generally low, allowing the system to remain sTable even under varying loads. For instance, in row 6 (Kp = 1.2; Ki = 0.02; Kd = 0), the pitch value was 6.4 and the roll value was 5.4 under no load, and the system remained sTable with a pitch of 7.4 and roll of 6.6 at a 228-gram load. A similar condition was observed in row 9 (Kp = 1.3; Ki = 0.05; Kd = 0), where the system recorded the lowest accumulated error, with pitch and roll values of 3.6 and 4.0 under no load, and pitch and roll values of 4.7 and 5.8 at the maximum load of 228 grams.

These results indicate that selecting a moderate Kp value, ranging from 1.0 to 1.3, significantly improves the system's stability and accuracy. Additionally, the inclusion of small Ki and Kd values proved effective in smoothing the response and reducing residual errors. However, these parameters must be carefully adjusted to avoid oscillations caused by increased sensitivity to variations in the error signal. Therefore, the precise determination of PID parameters, particularly the proportional gain, is crucial for maintaining optimal performance of this ESP32-based two-axis gimbal system.

C. System Response Time

Table VI presents the settling time test results for various PID parameter combinations under four load conditions: no load, 128 grams, 204 grams, and 228 grams. The settling time indicates the time required for the gimbal system to reach a sTable state at the 0° setpoint, with a maximum test duration of 30 seconds. In the initial tests (rows 1 and 2) with Kp values of 0.5 and 0.7, the system failed to reach the setpoint within the allotted time due to insufficient proportional gain, resulting in a corrective force too weak to drive the actuator toward a sTable position. Consequently, the recorded settling time of 30 seconds represents the upper limit of the test rather than the actual stabilization time, indicating that the system did not achieve steady-state conditions within the test duration.

TABLE VI
SYSTEM RESPONSE TIME

Parameter PID			Settling time (Sec)			
Kp	Ki	Kd	Tanpa beban	186 Gram	204 Gram	228 Gram
0,5	0	0	30	30	30	30
0,7	0	0	28	29	30	30
0,9	0	0	25	27	26	29
1	0	0	12	15	13	15
1,2	0	0	5	7	6,2	7,5
1,2	0,02	0	3,3	4	6,3	7,6
1,2	0,05	0	2,7	3	4,7	8,3
1,1	0,01	0	2,2	3,2	4,2	4,5
1,3	0,05	0	1,3	1,6	4	5,3
1,3	0,04	0,001	0,9	1	2	4
1	0,03	0,001	2	2,2	4,3	7,2
1	0,05	0,001	2,1	2,4	7,6	6
1,2	0,05	0,001	5	4	5	8,7
1,3	0,06	0,002	5,3	7,3	7,3	8
1,3	0,02	0,002	7,3	5,8	6,9	9,2

As the Kp value increased and Ki and Kd parameters were introduced, the system's performance improved significantly, with shorter stabilization times observed across all load conditions. The parameter combination Kp = 1.3; Ki = 0.05; Kd = 0 (row 9) produced the most optimal result, achieving the shortest settling time of 1.3 seconds without load and 4 seconds at a 228-gram load, demonstrating an effective balance between response speed and system stability. However, in higher-gain configurations (rows 14 and 15), the settling time increased to 9.2 seconds due to oscillations caused by excessive system sensitivity. These findings confirm that moderate PID parameter values yield the most optimal performance, maintaining an appropriate balance between responsiveness, stability, and positional accuracy under varying load conditions.

D. Analysis of Setpoint vs. Actual Value Graph

The graphs presented the two-axis gimbal system's response to the setpoint (0°) on the X and Y axes, with variations in the proportional constant (Kp) of the PID algorithm and smartphone loads attached to the gimbal, namely, no load, 186 grams, 204 grams, and 228 grams. This testing aims to evaluate the system's ability to reach and maintain a sTable position in the presence of external disturbances under various loading conditions.

Fig.6 illustrates the explanation of the X and Y axis degrees in relation to the Setpoint value. In the no-load test, the system exhibited a relatively fast response but significant overshoot and initial oscillation, mainly due to the absence of additional mass that would otherwise provide mechanical damping. With fairly aggressive PID parameters (without Kd), the system stabilized quickly; however, this led to small, recurring vibrations, indicating insufficient damping of high-frequency noise. The overall settling time in this condition was approximately 2.2 seconds, the fastest of all tests due to the system's minimal inertia.

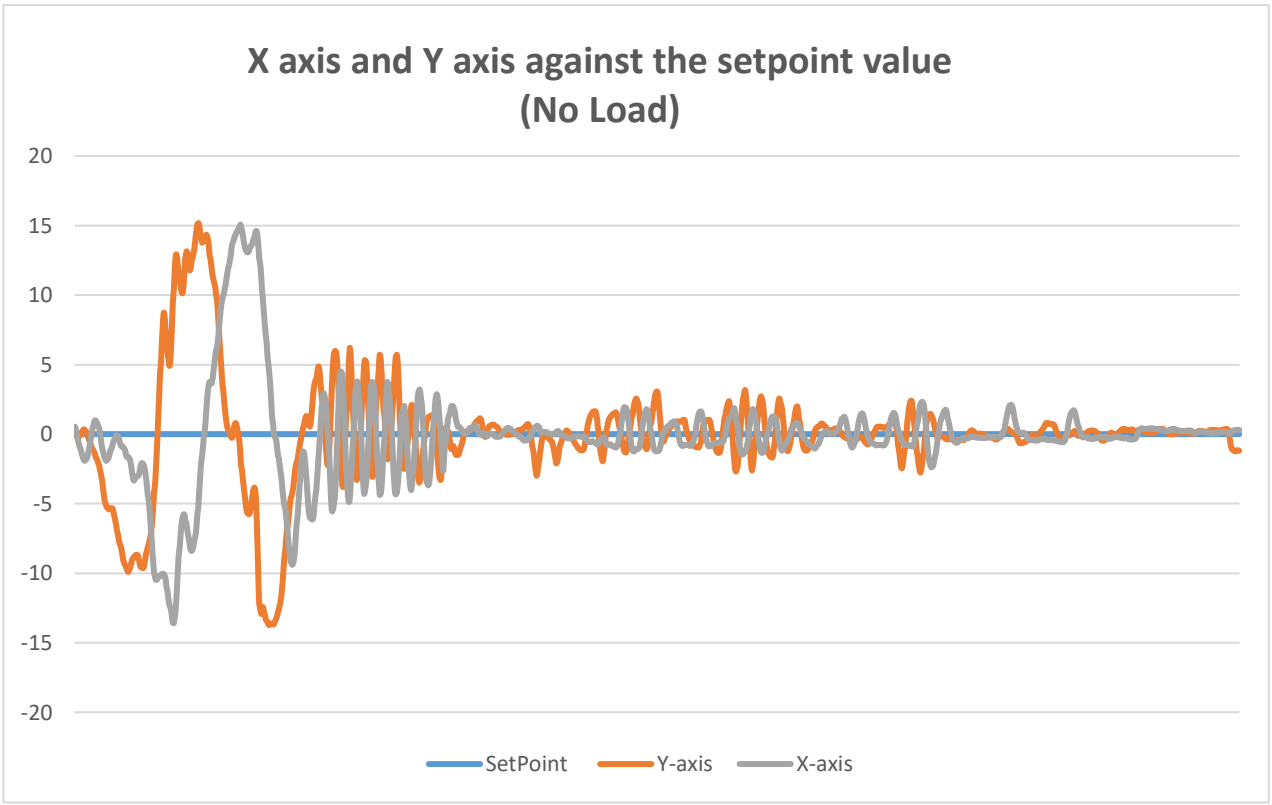

Fig. 6 X Axis and Y Axis Graphs Compared to Setpoint

Fig.7 shows the explanation of the X and Y axis angles in relation to the Setpoint value. The addition of a 186-gram load and a slight increase in Kd resulted in the system demonstrating improved damping of the initial oscillations. Overshoot still occurred but was more quickly controlled, and high-frequency vibrations began to diminish, especially on the X-axis. These parameters provided a better balance between responsiveness and stability compared to the no-load condition. The settling time increased slightly to 3.2 seconds, indicating that the added mass provided mechanical damping, reducing oscillations but requiring a longer time to reach steady-state stability.

Fig. 8 explains the X- and Y-axis angles relative to the Setpoint value. Despite using the same parameters as in the no-load condition, the system exhibited a greater initial overshoot under a 204-gram load, particularly on the Y-axis. However, the added weight provided a natural damping effect, allowing the system to reach stability more quickly, even without the Kd contribution. This indicates that system stability is influenced not only by PID tuning but also by the load's mass. The settling time for this test was recorded at 4.2 seconds, indicating that the increased inertia caused by the heavier load slowed the system's return to a sTable state, despite maintaining good steady-state accuracy.

DOI: http://dx.doi.org/10.25139/ijair.v7i1.10771

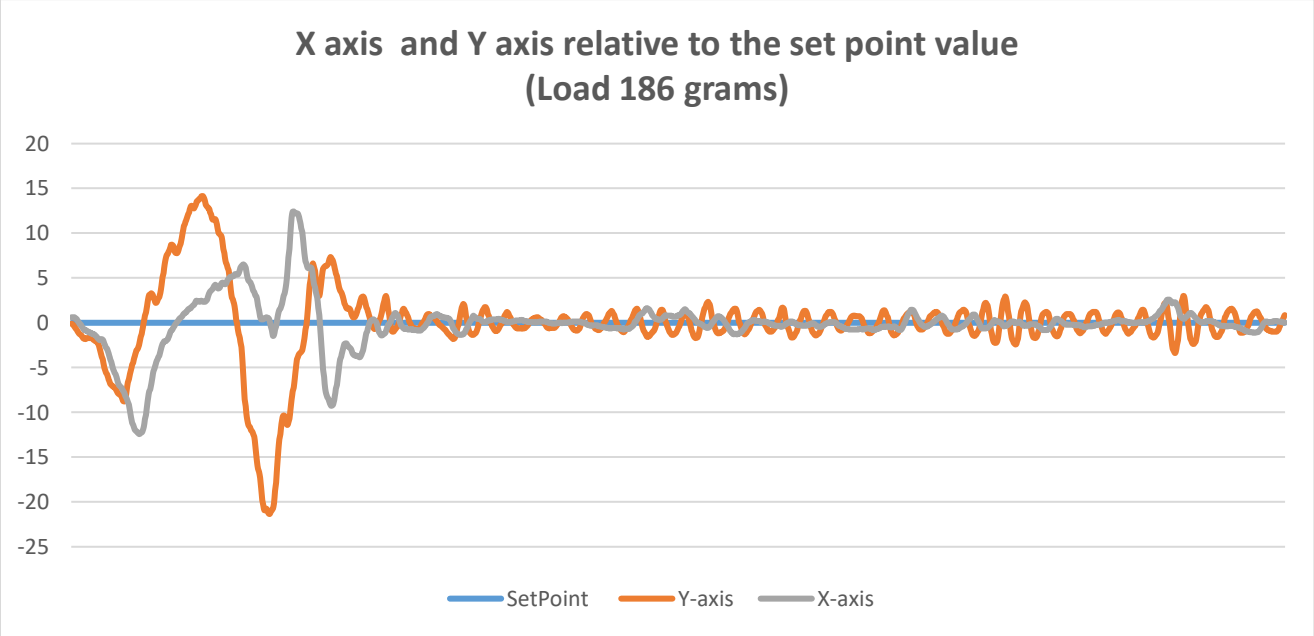

Fig. 7 X Axis and Y Axis Graphs Compared to Setpoint

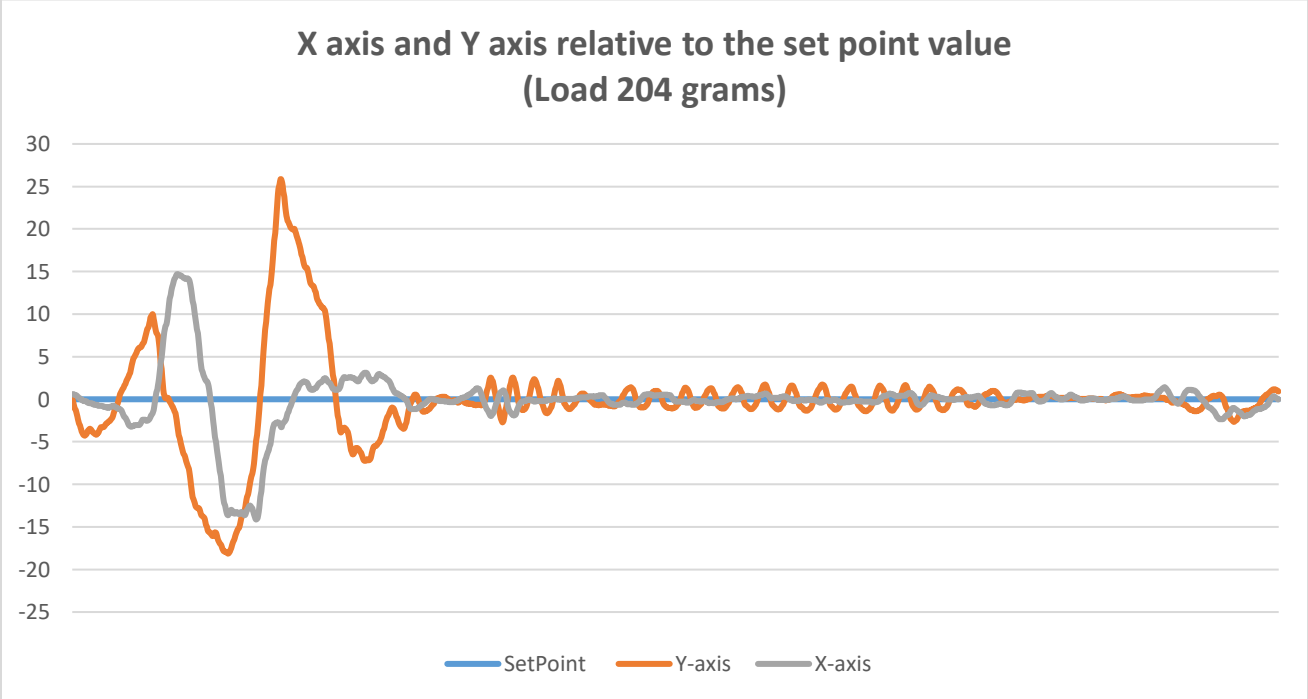

Fig. 8 X Axis and Y Axis Graphs Compared to Setpoint

Fig. 9 above explains the angles of the X and Y axes relative to the Setpoint value. At the highest load, the system remained stable despite greater inertia. With a slight reduction in the Ki value, the system exhibited less oscillation during the initial phase while remaining responsive. Although Kd was not used, the damping effect from the larger mass helped suppress noise, allowing the system to remain sTable with relatively low accumulated error and smooth motion. The settling time increased to 4.5 seconds, representing the slowest response among all tests due to the larger mass moment of inertia. Nevertheless, the system maintained a sTable steady-state condition with minimal residual error, confirming that the implemented PID controller can sustain reliable stability under maximum load conditions. The damping effect from the larger mass helped suppress noise, allowing the system to remain sTable with relatively low accumulated error and smooth motion.

DOI: http://dx.doi.org/10.25139/ijair.v7i1.10771

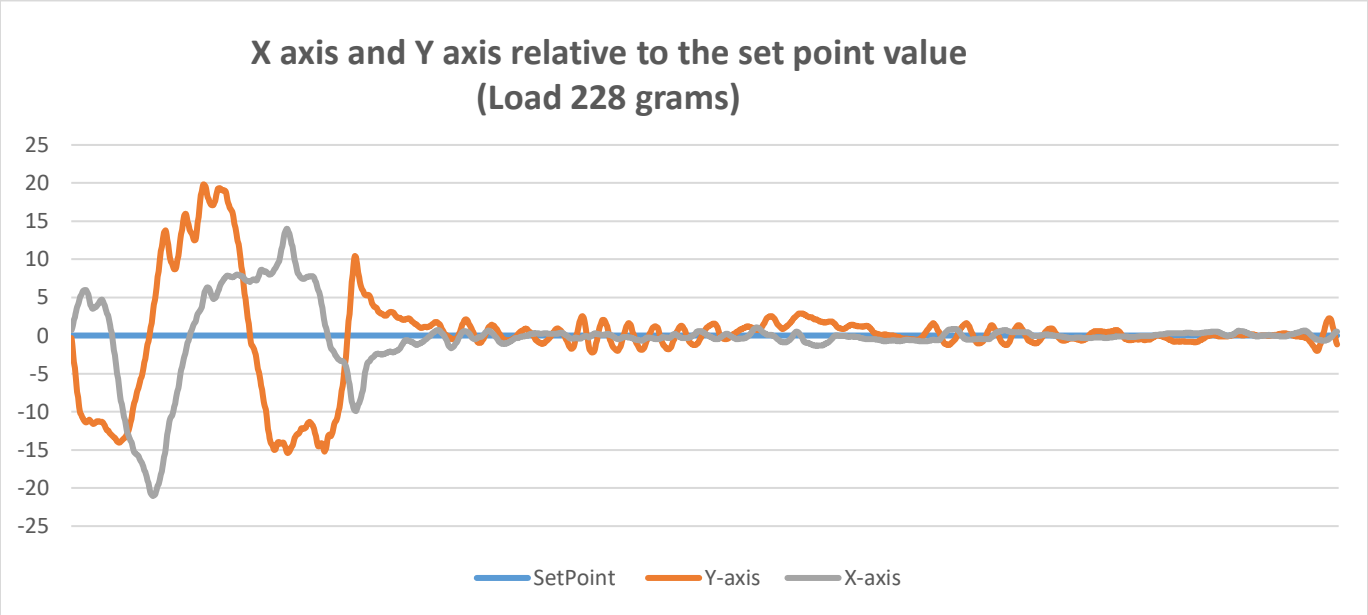

Fig. 9 X Axis and Y Axis Graphs Compared to Setpoint

IV. CONCLUSION

Based on the design and testing results, the ESP32-based two-axis camera stabilizer system, utilizing the PID algorithm, effectively maintained camera orientation under both unloaded and loaded conditions. The MPU6050 sensor accurately measured pitch and roll angles in real time, while the PID controller provided precise corrections to minimize deviations. The best PID parameters varied depending on the load: for no load, the optimal values were Kp = 1.3, Ki = 0.05, and Kd = 0; for 186 g and 204 g loads, the best performance was achieved with Kp = 1.2, Ki = 0.05, and Kd = 0; and for the 228 g load, the optimal parameters were Kp = 1.3, Ki = 0.05, and Kd = 0. Additionally, voltage readings obtained using a voltage divider circuit demonstrated sufficient accuracy, with an average error of 0.343 V. Overall, this system functions effectively as a basic gimbal stabilizer. It has potential for further development in terms of control optimization and feature integration. These results indicate that a moderate proportional gain with small integral action ensures system stability and accuracy across varying loads. In future work, this research will focus on developing an adaptive PID control system that automatically adjusts parameter values based on detected smartphone weight, thereby further improving response speed, stability, and overall performance.

REFERENCES

[1] A. Chiko Pratama, D. Syauqy, M. Hannats, and H. Ichsan, “Stabilizer Kamera 2-Axis Dengan Pid Control Berdasarkan Setpoint pada Atmega 328,” vol. 2, no. 9, pp. 2548–964, 2018.
[2] B. Irianti and N. Karlinah, “PEMANFAATAN SENSOR GYRO PADA SISTEM PENSTABIL KAMERA,” vol. 3, no. 2, pp. 195–200, 2021.
[3] R. T. Asnada, "Effect of Inertial Measurement Unit (IMU) MPU-6050 3-Axis Gyro and 3-Axis Accelerometer on Camera Stabilizer System (Gimbal) For Videography Applications," vol. 11, no. 1, pp. 48–55, 2020.
[4] R. T. Asnada and S. Sulistyono, “Pengaruh Inertial Measurement Unit (IMU) MPU- 6050 3-Axis Gyro dan 3-Axis Accelerometer pada Sistem Penstabil Kamera (Gimbal) Untuk Aplikasi Videografi,” J. Teknol. Elektro, vol. 11, no. 1, p. 48, 2020, doi: 10.22441/jte.2020.v11i1.007.
[5] S. Abdelhay and A. Zakriti, "Modeling of a Quadcopter Trajectory Tracking System Using PID Controller," Procedia Manuf., vol. 32, no. 2, pp. 564–571, 2019, doi: 10.1016/j.promfg.2019.02.253.
[6] M. S. Haris, A. Dharmawan, and C. Atmaji, “Sistem Kendali Gimbal 2-Sumbu Sebagai Tempat Kamera Pada Quadrotor Menggunakan PID Fuzzy,” IJEIS (Indonesian J. Electron. Instrum. Syst., vol. 7, no. 2, p. 185, 2017, doi: 10.22146/ijeis.24220.
[7] T. K. Priyambodo, “Implementasi Sistem Kendali PID pada Gimbal Kamera 2-sumbu dengan Aktuator Motor Brushless,” IJEIS (Indonesian J. Electron. Instrum. Syst., vol. 7, no. 2, p. 111, 2017, doi: 10.22146/ijeis.18238.
[8] F. Palaha and Yolnasdi, “Analisa Rancangan Keseimbangan Menggunakan Sensor Imu Type – Mpu6050 Pada Quadcopter,” Sainstek (e-Journal), vol. 8, no. 2, pp. 96–104, 2020, doi: 10.35583/js.v8i2.125.
[9] G. Al Azhar, S. Sungkono, M. N. Achmadiyah, and S. Izza, “Peningkatan Kestabilan Sistem Kontrol UGV melalui Optimalisasi Manajemen Core dan Free-RTOS pada ESP32,” J. Elektron. dan Otomasi Ind., vol. 10, no. 2, pp. 253–263, 2023, doi: 10.33795/elkolind.v10i2.3720.

[10] A. Wagyana, “Prototipe Modul Praktik untuk Pengembangan Aplikasi Internet of Things (IoT),” Setrum Sist. Kendai-Tenaga-elektronika-telekomunikasi-komputer, vol. 8, no. 2, p. 238, 2019, doi: 10.36055/setrum.v8i2.6561.
[11] J. Ilmiah, M. Disiplin, M. Esp, and D. A. N. Microsd, “PENGEMBANGAN DETEKTOR PORTABEL TINGKAT KERUSAKAN JALAN,” vol. 8, pp. 7–22, 2024.
[12] Dicky Juliansyah and M. Dr. Ir. Yohannes Dewanto, “Prototype Robot Pensortir Kemasan Obat Berdasarkan Warna RGB Pada Warna Kemasan Obat Menggunakan Sensor TCS-3200 Dan HC-SR04,” J. Tek. Elektro Fak. Tek. Dirgant. dan Ind., vol. 13 N0. 1, pp. 1–8, 2024.
[13] P. A. Guntoro, Muhamad Syariffuddien Zuhrie, Bambang Suprianto, and I. G. P. A. Buditjahjanto, “Sistem Kendali Dual Motor Propeller Pada Alat Self Balancing Menggunakan Kontroler Pid Dengan Tuning Chr,” J. Tek. Elektro, vol. 10, no. 1, pp. 19–27, 2021.
[14] R. I. Putra, S. Sunardi, and R. D. Puriyanto, “Monitoring Tegangan Baterai Lithium Polymer pada Robot Line Follower Secara Nirkabel,” Bul. Ilm. Sarj. Tek. Elektro, vol. 1, no. 2, p. 73, 2019, doi: 10.12928/biste.v1i2.907.
[15] R. Rizanty, “Rancang Bangun Pembangkit Listrik Tenaga Gerak Melalui Sepeda Statis dengan Kontrol Torsi Berbasis Arduino Menggunakan Metode PID,” 2024.
[16] A. A, "Camera Stabilizer 2 Axis by Proporsional Integral Derivative (PID) Based LabView," Telekontran J. Ilm. Telekomun. Kendali dan Elektron. Terap., vol. 3, no. 2, pp. 25–27, 2015, doi: 10.34010/telekontran.v3i2.1879.
[17] F. Beny, “Implementasi Sensor IMU MPU6050 Berbasis Serial I2C pada Self-Balancing Robot,” J. Teknol. Technoscientia, vol. 9, no. 1, 2016.
[18] Mutiar, “Analisa Perubahan Parameter Sistem Pengendalian PID Terhadap Tangapan Keluaran Dengan Menggunakan Aplikasi Matlab,” J. Tek. Elektro, vol. 7, no. 1, pp. 45–53, 2021.
[19] A. Nurmiranto et al., “Pengembangan Desain, Simulasi Dan Pengujian Robot Tangan Menggunakan Flex Sensor Terintegrasi Dengan 3D Animation Simmechanics,” J. Tek. Mesin, vol. 4, no. 1, pp. 105–116, 2016.
[20] A. Fahruzi, B. S. Agomo, and Y. A. Prabowo, "Design Of 4DOF 3D Robotic Arm to Separate the Objects Using a Camera," Int. J. Artif. Intell. Robot., vol. 3, no. 1, pp. 27–35, 2021, doi: 10.25139/ijair.v3i1.3787.
[21] R. R. Ramadhani, M. Yuliana, and A. Pratiarso, "Smart Room Lighting System for Energy Efficiency in Indoor Environment," Int. J. Artif. Intell. Robot., vol. 4, no. 2, pp. 48–58, 2022, doi: 10.25139/ijair.v4i2.5266.