Balance Control of Hexapod Robot Against Uneven Terrain

Sahdad Jovito[1], Senanjung Prayoga[1], Anugerah Wibisana[1] and Aldi Wahyudi[1]

Department of Electrical Engineering, Politeknik Negeri Batam, Batam, Indonesia
sahdad.jovito@students.polibatam.ac.id

Abstract. This research focuses on addressing the challenges faced by robots when going through uneven terrain by maintaining stability in an inclined position. The main objective is to design a PID control algorithm that allows the robot to adjust its motion based on real-time sensory feedback from the IMU sensor. The research methods include Inverse Kinematics, Matrix Rotation, Body Offset, PID controller design, and controller parameter optimization using the Ziegler-Nichols method which can help reduce the tuning time. This system works well producing an average response time when balancing the body for 1.1 seconds with a success rate of 80% and can climb stairs with an average time of 44.9 seconds with a success rate of 84%.

Keywords: Balancing, PID control, Matrix Rotation.

1 Introduction

Robotics has evolved from simple devices to complex systems, helping humans in routine work and precision tasks to achieve high results [1]. One of the applications is in the Indonesian SAR Robot Contest. Where the legged robot is assigned to rescue victims after an earthquake disaster, and the robot can pass through obstacles that illustrate the effects of post-earthquake disasters. The robot is designed to be able to overcome various obstacles including broken roads, sloping surfaces, muddy terrain, stairs, and ravines. Such obstacles disrupt the stability of the robot, leading to the risk of the robot tipping over or skidding due to improper leg movements, hampering its ability to move effectively [2]. Lack of balance control on uneven terrain can lead to instability, causing the robot's legs to slip frequently due to poor mobility. The robot's ability to maintain traction is compromised by the suboptimal positioning of its legs [3]. This makes the servo vulnerable to damage due to snagging or hitting obstacles.

In research [4] and [5] conducted with the aim of improving the stability of the robot system through the use of Fuzzy logic, which requires a fairly long period of time in producing an appropriate response, it is related to the heavy computational load and memory capacity when using this method. In contrast, research [6] conducted using the PID (Proportional-Integral-Derivative) approach resulted in a faster response time. The

L. Lumombo et al. (eds.), *Proceedings of the 7th International Conference on Applied Engineering (ICAE 2024)*, Advances in Engineering Research 251,
https://doi.org/10.2991/978-94-6463-620-8_7

contrasting results of the two methods lead to the consideration of integrating PID control in the study framework as a method of efficiently balancing the robot.

A key aspect of successful robot balancing lies in the ability of the robot to position the body based on the inclination experienced. using a PID approach can improve the stability of the robot system resulting in fast response times. By implementing a balance control system on a hexapod robot, it can reduce the risk of damage to the robot's joints while making it easier for the robot to traverse uneven terrain, allowing the robot to increase stability and traverse obstacles effectively [7].

2 Method

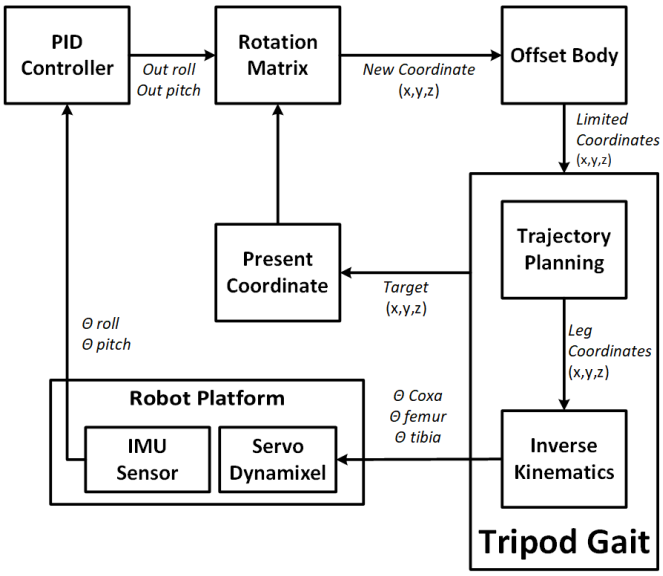


Fig. 1. System diagram.

The robot used in this research is a six-legged robot or commonly called a hexapod. For the movement of the robot, calculations, trajectory planning and inverse kinematics are applied to ensure proper control and positioning. The robot also uses a tripod gait pattern to determine the division of legs to step.

2.1 Inverse Kinematics

Inverse Kinematics works by converting the coordinate values of the End-Effector into degree values at each robot joint [8]. Inverse Kinematics on legged robot functions as a method that can automatically calculate the degree of the servo motor at each joint, so there is no need to enter it manually [6]. The robot leg is designed to follow the shape of an insect leg which consists of 3 joints namely coxa, femur, and tibia. the joints of this leg also describe the number of degrees of freedom of the robot leg.

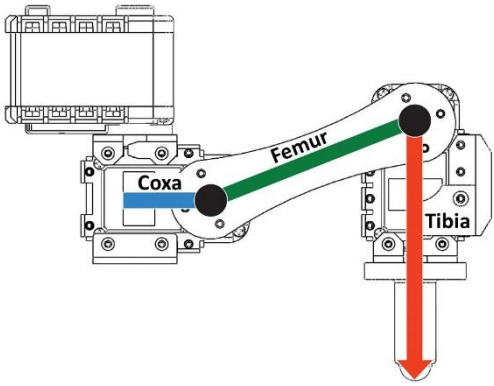


Fig. 2. Robot leg structure.

Given that the robot legs have three degrees of freedom, the Inverse Kinematics applied to the robot can use geometric approximation [9] as follows:

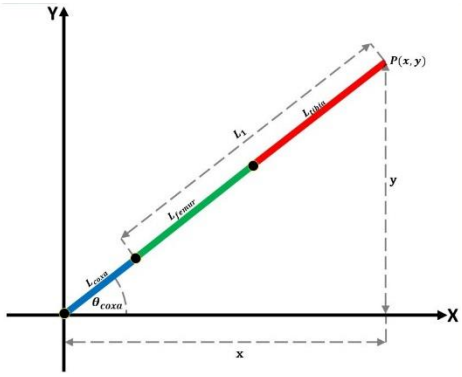


Fig. 3. Leg perspective on XY axis.

Based on Figure 3, to find the value of the degree of coxa and the value of the length of the stretch of the leg can be found with the equation:

$$\theta_{coxa} = \tan^{-1}\left(\frac{y}{x}\right) \tag{1}$$

$$a = \sqrt{z^2 + (x - L_{coxa})} \tag{2}$$

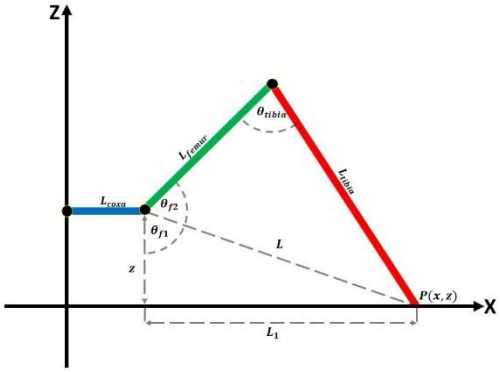


Fig. 4. Leg perspective on XZ axis.

The angle of the femur can be found using the following equation:

$$\theta_{f1} = \tan^{-1}\left(\frac{z}{(x-L_{coxa})}\right) \tag{3}$$

$$\theta_{f2} = \cos^{-1}\left(\frac{{L_{femur}}^2+a^2-t^2}{2\times a\times L_{femur}}\right) \tag{4}$$

$$\theta_{femur} = \theta_{f1} + \theta_{f2} \tag{5}$$

Based on Figure 4, the direction of the End-Effector is targeted downwards, but in the implementation in the robot leg the zero point of the tibia join is placed horizontally parallel to the femur join so that to find the tibia degree, we can use the equation:

$$\theta_{tibia} = \left(\cos^{-1}\left(\frac{{L_{femur}}^2+{L_{tibia}}^2-a^2}{2\times L_{femur}\times L_{tibia}}\right)\right) - 180° \tag{6}$$

2.2 Trajectory Planning

Trajectory planning on robots uses the 4th order trajectory polynomial method to create robot footstep patterns. This method is only used to create End-Effector motion trajectory patterns, and what produces the degree of robot foot motion is inverse kinematics, so a combination of the two algorithms is needed. Here is the equation.

$$P(t)_{x,y,z} = (1-t)^3 P1_{x,y,z} + 3t(1-t)^2 P2_{x,y,z} + 3t^2(1-t)P3_{x,y,z} + t^3 P4_{x,y,z} \tag{7}$$

Based on equation 7, the input parameters given are the destination coordinate points of the robot legs consisting of the starting, peak and end points. Then the equation will produce a curved trajectory graph like Figure 5.

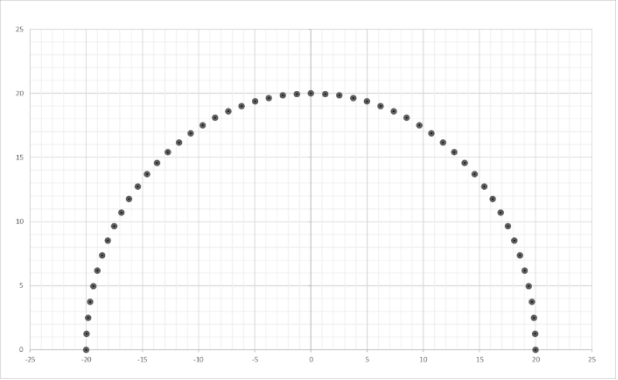


Fig. 5. Trajectory graph.

2.3 Walking Gait

Tripod gait is a periodic walking pattern that produces rhythmic movements that are synchronized with the environment [10]. The actuation of each robot limb can be categorized into two different phases, specifically, the swing phase where the leg is lifted off the ground and swings forward to start the next phase, and the stance phase where the leg provides thrust to the robot body to provide movement in the intended direction.

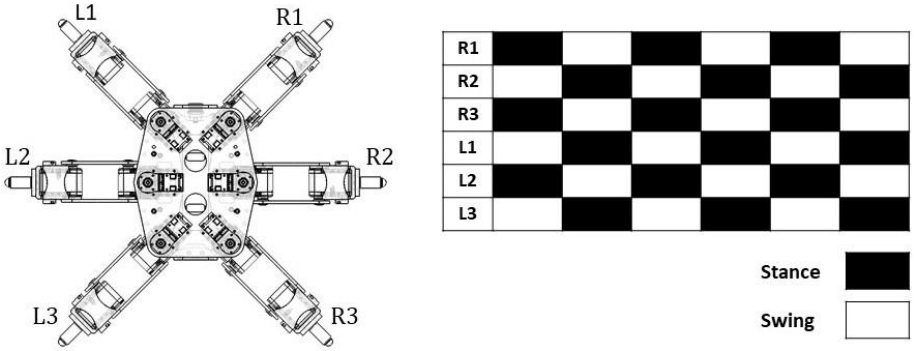


Fig. 6. Tripod Gait Timing.

The gait pattern, which consists of the sequence of motion of each robot leg so that the robot can move dynamically, is known as the tripod gait pattern, using three legs to tread and three legs to swing. When the hexapod robot moves forward, legs R1, R3, and L2 shift forward in the standing phase, while legs L1, L3, and R2 move toward the rear in the swing phase. As a result, both sets of legs take turns performing these movements to achieve locomotion of the hexapod robot [11].

2.4 Offset Body

The Body Offset calculation is used to limit the motion of the robot legs so that they do not pass through coordinates that cannot be reached by the End-Effector during the Inverse Kinematics calculation process. By using data on the relative position of each End-Effector to the body center, this calculation can adjust each End-Effector of each leg so that the robot body can move [12].

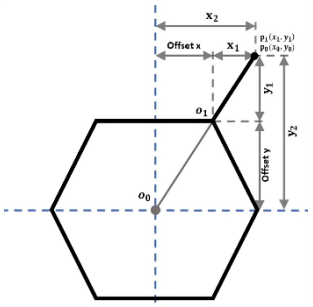


Fig. 7. Robot model in the XY plane.

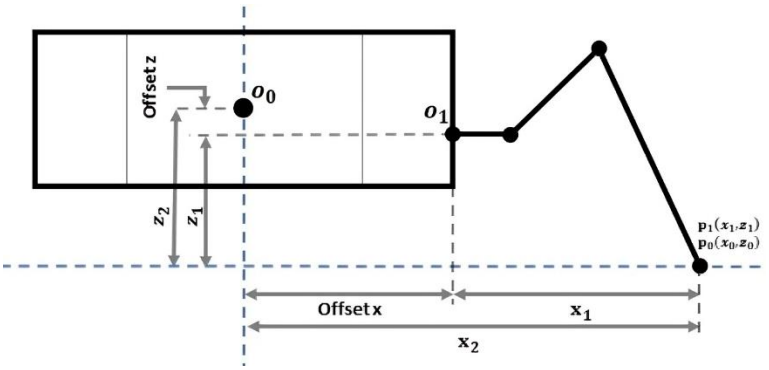


Fig. 8. Robot model in the XZ plane.

Point o_0 is the center point of the robot body while o_1 is the center point connecting the legs and the robot body as the center of inverse kinematics. Values for offsets x, y and z are obtained through manual measurement of the robot body. Point P_0 is the starting point of the robot's End-Effector while point P_1 is the point to which the End-Effector will go. The following equation explains how to calculate the position of the End-Effector with respect to the robot body.

$$x_1 = x_0 + Offset\ x \tag{8}$$

$$y_1 = y_0 + Offset\ y \tag{9}$$

$$z_1 = z_0 + Offset\ z \tag{10}$$

2.5 Rotation Matrix

The rotation matrix simplifies the movement of the robot by entering the desired rotation angle. The robot can directly adjust the rotational position of its body, so that the program does not need to add logic to provide a decision on the current state of the robot's rotation [13]. The geometry rotation matrix consists of the displacement or shift of a point on the geometry plane along the arc of a circle with the center of the circle as the rotation point [14]. The rotation matrix can work with respect to the X (roll), Y (pitch), and Z (yaw) axes. The rotation matrix is expressed as follows:

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \tag{11}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{12}$$

$$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{13}$$

$$P_{x,y,z} = R_x(\gamma) \cdot R_y(\beta) \cdot R_z(\alpha) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{14}$$

2.6 Balancing Algorithm

To adjust the balance of the robot using the PID control method. PID control consists of a combination of proportional (P), integral (I), and derivative (D) control. The robot will provide feedback from the inertial sensor (IMU), namely pitch and roll angle data.

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \tag{15}$$

By utilizing the above equations, and the combination of the calculation of rotation matrix, body offset and inverse kinematics can produce a robot balancing algorithm.

```
robot balancing algorithm
Input        : Pitch, Roll
Output       : Robot Movement

1:  rollData, pitchData = readSensorData()
2:  if (rollData > offsetRoll) then

3:        errorRoll  = rollData - spRoll
4:        errorPitch = pitchData - spPitch
```

```
5:        outRoll  = calculatePID(errorRoll)
6:        outPitch = calculatePID(errorPitch)

7:        x,y,z = RotationMatrix(outRoll, outPitch)
8:        x1,y1,z1 = offsetBody(x,y,z)

9:        inverseKinematics(x1,y1,z1)
```

This algorithm was created with the aim of identifying the rotational inclination of the robot, utilizing data from the IMU sensor. Starting with IMU sensor calibration and servo initiation, the system proceeds to acquire sensor data, extract Roll and Pitch angle information, then compared to a predefined offset. This offset serves as the determining factor in ensuring significant tilting of the robot. After detecting the tilt, the robot immediately performs PID calculations and searches for the right coordinates based on the rotation matrix calculation.

3 Result

This chapter covers system tuning, robot balance test, and the success rate of the robot balance algorithm when going through stairs.

3.1 Tuning PID Parameter

This process is carried out to get good parameters for each controller namely Kp, Ki and Kd using the Nichols Ziegler tuning method. Based on the Nichols Ziegler tuning rules to get each value on each controller can be started with zeroing the integral and differential gains and then raising the proportional gain until the system is unstable.

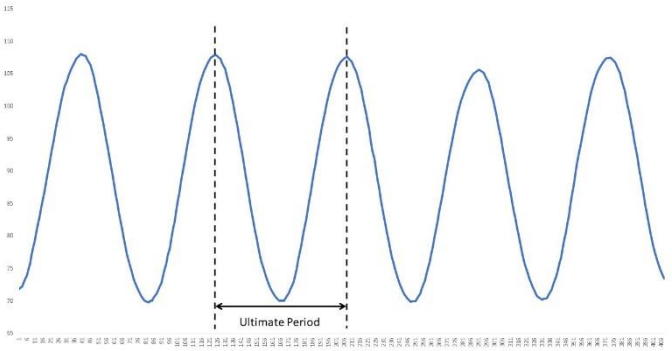


Fig. 9. Feedback IMU sensor oscillation graph

When the system has produced constant oscillations, record the gain value (Ku) and oscillation period (Pu). The value of Ku is obtained from the amount of proportional

value given so that the system becomes unstable. While the value of Pu is obtained from the signal period of the system. Put these values into the Ziegler-Nichols closed-loop rule and determine the required settings for the controller.

Table 1. Closed-Loop Calculations of Kp, Ti, Td.

Controller	Kp	Ti	Td
P	Ku/2	0	0
PI	Ku/2.2	Pu/1.2	0
PID	Ku/1.7	Pu/2	Pu/8

To get the value of the integral amplifier (Ki) is to divide the Kp value against Ti, while for the derivative amplifier (Kd) value by multiplying the Kp and Td values. Its application in this balance system, to provide constant oscillation results, the Kp amplifier is given a value of 2, so if observed in the graph in Figure 9, the value of Pu will be 0.85s while the value of Ku is equal to 2.

3.2 Robot Balance Test

The robot is tested on an inclined road to determine the effectiveness of the control system made. where the robot must be able to balance its body using a controller that has been designed previously. At this stage, the performance generated by the robot will be seen by analyzing the graph of the IMU sensor feedback value applied based on the value of the P, PI and PID controllers based on the Nichols-Ziegler table. The performance of the robot can be seen in the response table with several parameters displayed are the value of rise time, settling time, overshoot, and steady state error based on the resulting graph.

P-Controller. Based on the rules in the previous section, the parameter value for Kp is 1, and Ki and Kd are 0. When tested on the robot, it produces movements as shown in Figure 10 and the following response graph and performance table.

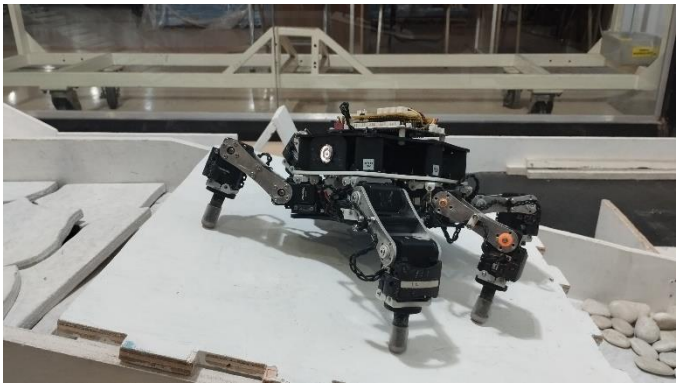
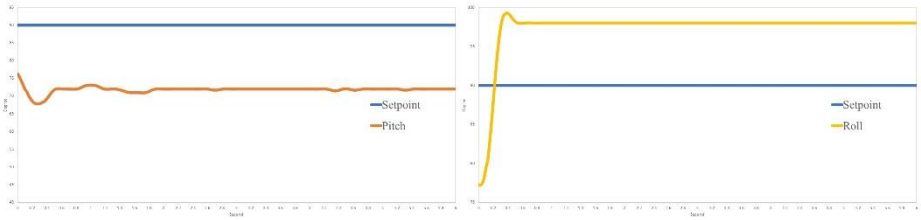


Fig. 10. Robot testing of the roll and pitch axes using the P-Controller



(a) (b)

Fig. 11. (a) Graph P-Controller Pitch, (b) Graph P-Controller Roll.

Table 2. P-Controller response.

Angle	Rise Time (s)	Settling Time (s)	Overshoot (%)	Steady State Error
Pitch	0.00	0.07	-15.33	18.00
Roll	0.13	0.28	10.30	8.00

PI-Controller. For the PI controller, the parameter value for Kp is 0.91, Ki is 1.6 and Kd is 0. When tested on the robot, it produces movements as shown in Figure 12 and the following response graph and performance table.

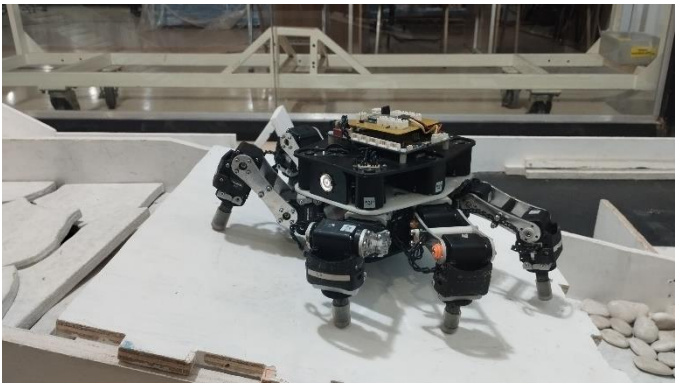
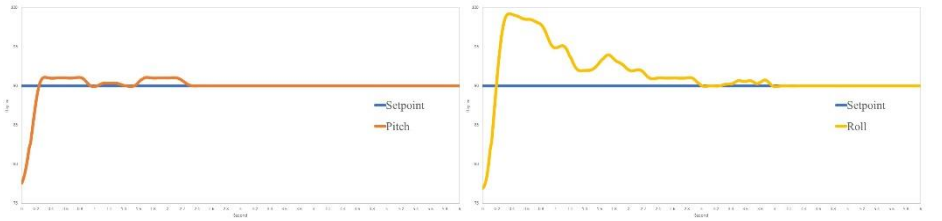


Fig. 12. Robot testing of the roll and pitch axes using the PI-Controller.



(b) (b)

Fig. 13. (a) Graph PI-Controller Pitch, (b) Graph PI-Controller Roll.

Table 3. PI-Controller response.

Angle	Rise Time (s)	Settling Time (s)	Overshoot (%)	Steady State Error
Pitch	0.09	0.20	1.20	0.00
Roll	0.09	2.20	10.22	0.00

PID-Controller. For the last controller, the parameter value for Kp is 1.18, Ki is 3.46 and Kd is 0.1. When tested on the robot, it produces movements as shown in Figure 14 and the following response graph and performance table.

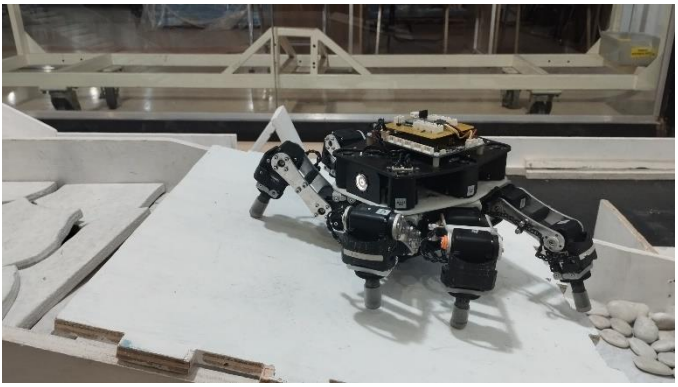
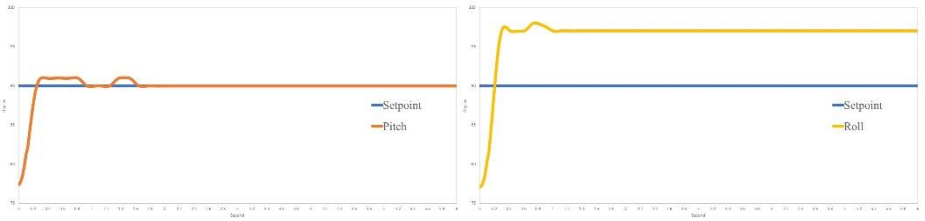


Fig. 14. Robot testing of the roll and pitch axes using the PID-Controller.



(a) (b)

Fig. 15. (a) Graph PID-Controller Pitch, (b) Graph PID-Controller Roll.

Table 4. PID-Controller Response.

Angle	Rise Time (s)	Settling Time (s)	Overshoot (%)	Steady State Error
Pitch	0.11	0.21	1.19	0.00
Roll	0.12	0.26	8.90	7.00

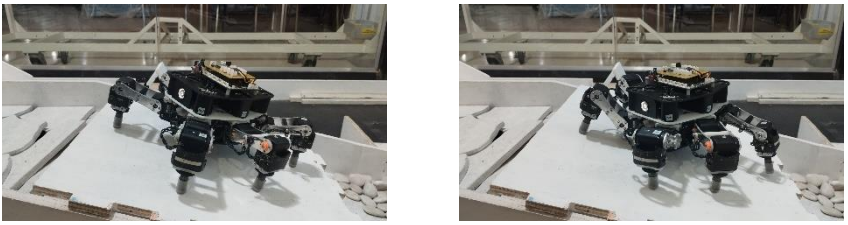
Based on the performance results of each controller, it can be seen that the PI controller produces better performance than the other controllers. in conjunction with the PID controller, the rotation matrix algorithm significantly contributes to producing fast motion, as it can directly generate coordinates to be implemented in the inverse kinematics algorithm. this also causes the P and PID controllers to achieve fast settling times, albeit at the expense of steady state errors at the specified angles the following figure shows the application of the PI controller selected for the robot's balance when viewed from a certain angle.

(a) (b)

Fig. 16. (a) Robot tilt to roll without PID, (b) Robot tilt to roll with PID.

(a) (b)

Fig. 17. (a) Robot tilt to pitch without PID, (b) Robot tilt to pitch with PID.



(a) (b)

Fig. 18. (a) Robot tilt to roll and pitch without PID, (b) Robot tilt to roll and pitch with PID.

3.3 Stair Climbing Testing

The robot is designed to conquer the stairs by applying balance control quickly and the robot does not topple over due to the slope of the stairs. So, it is necessary to test the robot directly in the arena. This test is carried out on the stairs used in the Indonesian SAR Robot contest which has a slope of about 29 degrees with a step height of 20 mm and a distance between steps of 36 mm. The following is a picture of the arena used in the test.

(a) (b)

Fig. 19. (a) Staircase illustration (b) Original testing staircase.

There are 25 tests carried out. The tests carried out are testing the balance of the robot and the success of the robot across the stairs and the time it takes. The test can be observed in Table 5.

Table 5. system testing.

Number of trial	The success of the system		time used (s)	
	balanced	climbed	balancing	climbing
1	Success	Success	1	50
2	Success	Success	1	55
3	Failed	Success	1	57
4	Success	Failed	1	94
5	Success	Success	1	44
6	Success	Success	1	47
7	Failed	Failed	1	54
8	Success	Success	1	48
9	Success	Success	2	38
10	Success	Success	1	52
11	Success	Success	1	56
12	Success	Success	1	48
13	Success	Success	1	37
14	Success	Success	1	39
15	Success	Success	1.5	37
16	Success	Success	1	39
17	Success	Success	1	36
18	Failed	Failed	1	46
19	Success	Success	1	36
20	Success	Success	1	53
21	Failed	Success	1	53
22	Success	Success	1	45
23	Failed	Failed	1	49
24	Success	Success	1	38
25	Success	Success	1.5	43

The failure criteria in the balance test is when the robot cannot reach the balance set-point determined based on the slope produced by the robot, this usually occurs because the position of the robot before balancing the body is not correct so that a slip occurs, while the criteria for failing to pass the stairs is when the robot's legs cannot pass the stairs because the stairs to be passed are not the same height or the robot cannot

respond to the slope while walking which is usually caused by drift problems from the IMU sensor. To get the percentage of success of this system can be calculated using the following equation.

$$succes\ percentage = \frac{Total\ Success}{Total\ number\ of\ tests} \times 100\% \quad (16)$$

$$Average\ Success\ Time = \frac{total\ time\ of\ success}{number\ of\ successes} \quad (17)$$

From the above equation, the percentage result for balance is 80% with an average robot entry in a successful balanced position of 1.1 seconds. For the percentage of passing the stairs is 84% with an average successful climbing speed of 44.9 seconds. While the 25 tests provide a good initial overview, but the authors recognize that more tests in more diverse conditions, such as more extreme terrain or testing outside the laboratory, can provide a deeper understanding of system performance. More tests can also help minimize variations in results and identify potential causes of failure in more detail.

4 Conclusion

Based on the results of this research, the robot can manage the balance well using a combination of PID algorithm, rotation matrix, body offset and inverse kinematics. By utilizing the Nichols Ziegler tuning method, it can get a fast response to the robot balance and save time in the parameter search process. This system also works well as evidenced by the 80% success rate for balancing with an average response time of 1.1 seconds and 84% success passing the stairs with an average successful climbing speed of 44.9 seconds. Nevertheless, there should still be further application of PID tuning or application of other methods so that the robot can work effectively.

References

1. Chawla, S.: ADVANCEMENT OF ROBOTICS IN HEALTHCARE. IJSSER. 07, 3936–3952 (2022). https://doi.org/10.46609/IJSSER.2022.v07i12.006.
2. Mitchell, A., Martin, A.E.: Quantifying the effect of sagittal plane joint angle variability on bipedal fall risk. PLOS ONE. 17, e0262749 (2022). https://doi.org/10.1371/journal.pone.0262749.
3. Yu, Z., Li, J., Huang, Q., Chen, X., Ma, G., Meng, L., Zhang, S., Liu, Y., Zhang, W., Zhang, W., Chen, X., Gao, J.: Slip prevention of a humanoid robot by coordinating acceleration vector. In: 2014 IEEE International Conference on Information and Automation (ICIA). pp. 683–688 (2014). https://doi.org/10.1109/ICInfA.2014.6932740.
4. Antok, A.T.B., Darmawan, A., Alasiry, A.H., Hermawan, H., Binugroho, E.H., Marta, B.S., Wibowo, I.K., Julian, A., Suparman, A.F.I.: Quadruped Robot Balance Control For Stair Climbing Based On Fuzzy Logic. In: 2021 International Electronics Symposium (IES). pp. 552–557 (2021). https://doi.org/10.1109/IES53407.2021.9594046.

5. Wibowo, I.K., Preistian, D., Ardilla, F.: Kontrol Keseimbangan Robot Hexapod EILERO menggunakan Fuzzy Logic. ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika. 9, 533–547 (2021). https://doi.org/10.26760/elkomika.v9i3.533.
6. Nasrudin, A.I., Anam, K., N, M.A.P.: Evaluasi Inverse Kinematics untuk Robot Quadruped Menggunakan Sensor Accelerometer. Jurnal Rekayasa Elektrika. 15, 186–194 (2019). https://doi.org/10.17529/jre.v15i3.14079.
7. Rangkuti, S., Liestyowati, D., Syaripudin, A.: Design and implementation the stability and direction of hexapod robot motion. International research journal of engineering, IT & scientific research. 8, 256–269 (2022). https://doi.org/10.21744/irjeis.v8n6.2196.
8. Horigome, N., Terui, A., Mikawa, M.: A Design and an Implementation of an Inverse Kinematics Computation in Robotics Using Gröbner Bases. In: Bigatti, A.M., Carette, J., Davenport, J.H., Joswig, M., and de Wolff, T. (eds.) Mathematical Software – ICMS 2020. pp. 3–13. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-52200-1_1.
9. Nugraha, I.D.: Pendekatan Geometri untuk Perhitungan Inverse Kinematics Gerakan Lengan Robot 4 Derajat Kebebasan. JTM-ITI (Jurnal Teknik Mesin ITI). 5, 1–8 (2021). https://doi.org/10.31543/jtm.v5i1.572.
10. Szadkowski, R., Prágr, M., Faigl, J.: Self-Learning Event Mistiming Detector Based on Central Pattern Generator. Front. Neurorobot. 15, (2021). https://doi.org/10.3389/fnbot.2021.629652.
11. Ma, J., Qiu, G., Guo, W., Li, P., Ma, G.: Design, Analysis and Experiments of Hexapod Robot with Six-Link Legs for High Dynamic Locomotion. Micromachines. 13, 1404 (2022). https://doi.org/10.3390/mi13091404.
12. Kurniawan, I.A., Feriyonika, F., Pramono, S.: Inverse dan Body Kinematics pada Robot Hexapod. Prosiding Industrial Research Workshop and National Seminar. 9, 115–123 (2018). https://doi.org/10.35313/irwns.v9i0.1050.
13. Sarabandi, S., Thomas, F.: A Survey on the Computation of Quaternions From Rotation Matrices. Journal of Mechanisms and Robotics. 11, (2019). https://doi.org/10.1115/1.4041889.
14. Apriandy, K., Dewantara, B.S.B., Dewanto, R.S., Pramadihanto, D.: Analisis Kinematika Maju dari Tangan Robotik Berjari 4 yang Digunakan pada Robot Humanoid T-FLoW. Indonesian Journal of Computer Science. 12, (2023). https://doi.org/10.33022/ijcs.v12i4.3291.