KENDALI KECEPATAN MOTOR DC MENGGUNAKAN PENGENDALI PID DENGAN ENCODER SEBAGAI FEEDBACK

Muhammad Reza Aditya Nurkholis Putera[1], Rahmat Hidayat[2]
Program Studi Teknik Elektro, Universitas Singaperbangsa Karawang [1,2]
priv.rezaadityamuha@gmail.com[1], rahmat.hidayat@staff.unsika.ac.id[2]

Submitted May 25, 2022; Revised July 29, 2022; Accepted August 1, 2022

Abstrak

Pengendali PID (*Proportional Integral Derivative*) banyak digunakan untuk memperbaiki respon sistem, karena menawarkan fungsionalitas yang jelas, sederhana, dan kemudahan penerapan serta penggunaan. Salah satu peranan penting pengendali PID dapat dilihat pengaplikasiannya untuk mengontrol dan menstabilkan kecepatan sudut dalam nilai referensi pada motor arus searah (DC). Tujuan penelitian ini adalah mengimplementasikan pengendali PID sebagai pengendali kecepatan motor DC sehingga dapat memahami karakteristik masing-masing parameter pengendali PID terhadap kurva respon sistem. Hasil pengujian menunjukkan bahwa masing-masing parameter nilai PID (Proporsional, Integral, Derivatif) memiliki karakteristik yang berbeda pula terhadap respon sistem. Misalnya menambahkan parameter Kp dapat mempengaruhi berkurangnya *Steady-state error,* mengurangi waktu *rise time*, namun dapat meningkatkan nilai *overshot*. Sedangkan menambahkan parameter Ki dapat mempengaruhi meningkatnya *overshot*, mengurangi waktu *rise time*, dan mengurangi bahkan hampir menghilangkan nilai *steady state error*. Lalu, dengan menambahkan parameter Kd dapat mempengaruhi mengurangnya *overshot*, mengurangi *rise time*, dan meningkatkan *undershot*.
Kata Kunci : Pengendali PID, Motor DC, Arduino Uno

Abstract

PID (Proportional Integral Derivative) controllers are widely used to improve system response , as they offer clear, simple functionality and easy application and use. One of the important roles of the PID controller can be seen in its application to control and stabilize the angular velocity in the reference value of a direct current (DC) motor. The purpose of this research is to implement a PID controller as a DC motor speed controller so that it can understand the characteristics of each PID controller parameter on the system response curve. The test results show that each PID value parameter (Proportional, Integral, Derivative) has different characteristics to the system response. For example, adding the Kp parameter can reduce the steady-state error, reduce the rise time, but can increase the overshot value. Meanwhile, adding the Ki parameter can affect the increase in overshot, reduce the rise time, and reduce or even almost eliminate the steady state error value. Then, adding the Kd parameter can reduce the overshot, reduce the rise time, and increase the undershot.
Key Words : *PID Controller, DC Motor, Arduino Uno*

1. PENDAHULUAN

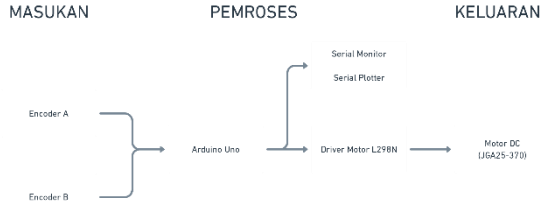
Pengendali PID telah menjadi pengendali yang populer untuk memperbaiki respon sistem kontrol loop tertutup, karena menawarkan fungsionalitas yang jelas, sederhana, dan kemudahan penerapan serta penggunaan [1]. Salah satu peranan penting pengendali PID dapat dilihat pengaplikasiannya pada motor arus searah (DC), masalah utama motor DC adalah bagaimana mengontrol dan menstabilkan kecepatan sudut dalam nilai referensi, karena kecepatan motor DC terkadang mengalami pelemahan akibat pembebanan yang dapat mempengaruhi melambatnya putaran sehingga kecepatan tidak konstan. Selain itu, melambat atau tidaknya kecepatan motor DC dipengaruhi oleh besaran nilai masukan tegangannya [2]. Maka, jika ingin meningkatkan kecepatan

motor DC dapat dilakukan pula menaikkan nilai masukkan tegangannya pada motor DC tersebut agar kecepatan motor DC stabil. Masalah pengendalian kecepatan motor DC ini dapat diatasi dengan menerapkan pengendali dengan beberapa metode seperti pengendali PID agar dapat mencapai kestabilan dan memperbaiki respon sistem yang diinginkan [3]. Pengendali PID ini memiliki 3 parameter, yaitu parameter kontrol *Proportional* (P), parameter kontrol *Integral* (I), dan parameter kontrol *Derivative* (D)[4]. Masing-masing parameter tersebut memiliki keunggulan tertentu[5]. Pada penelitian ini dilakukan perancangan perangkat keras sistem tertanam Arduino Uno, motor DC dengan rotary encoder sebagai umpan balik sistem dan mengimplementasikan pengendali PID sebagai pengendali kecepatan motor DC sehingga dapat memahami karakteristik masing-masing parameter pengendali PID terhadap kurva respon sistem.

Pada penelitian yang dilakukan oleh I. Qosim dan M. Mujirudin, respon sistem ketika diterapkan pengendali PID menunjukkan kinerja yang cukup baik [6]. Namun penelitian ini dilakukan dalam ruang lingkup simulasi yang dimana mempunyai kondisi yang ideal sehingga kinerja menghasilkan perbedaan jika di-implementasikan nyata pada perangkat keras dikarenakan terdapat berbagai faktor yang dapat mempengaruhi kinerja.

2. METODE PENELITIAN

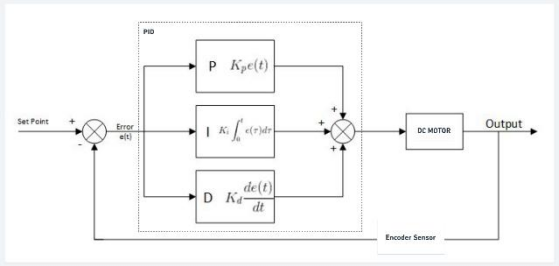
Diagram Blok Sistem Tertanam



Gambar 1. Diagram Blok Sistem Tertanam

Terlihat pada gambar 1 yang merupakan blok diagram sistem tertanam dari pengendali kecepatan motor DC, sensor *encoder* sebagai masukkan mengirim data pulsa menuju Arduino Uno sehingga dapat dilakukan perhitungan nilai kecepatan putaran per menit. Lalu fitur *timer* dan *counter* yang dipakai pada Arduino Uno memproses data kecepatan putaran sehingga data tersebut dapat ditampilkan pada serial monitor dan serial plotter untuk melihat kurva respon sistem. Untuk mengatur tegangan masukkan motor DC, digunakan Pulse Width Modulation (PWM) dan *driver motor* L298N untuk mengubah nilai tegangan digital (PWM) menjadi tegangan analog sehingga kecepatan motor DC dapat bervariasi sesuai yang diinginkan. Perangkat keluarannya adalah motor DC JGA25-370 dengan spesifikasi 12V 1000RPM dengan sensor encoder yang menyatu dengan motor DC tersebut.

Diagram Blok Sistem Kendali



Gambar 2. Diagram Blok Sistem Kendali

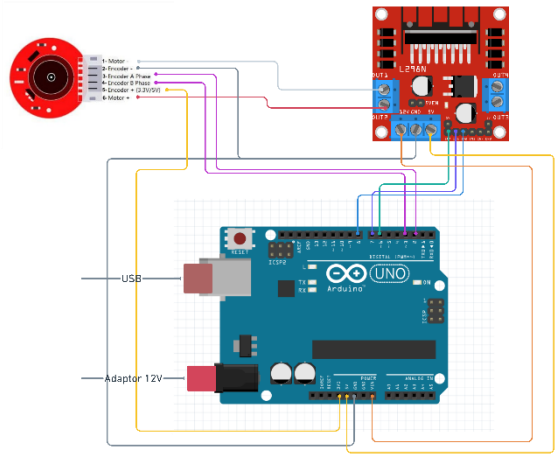
Sedangkan blok diagram untuk sistem kendali ditunjukkan pada gambar 2. terlihat bahwa sistem ini menunjukkan sistem kontrol *loop* tertutup dengan *encoder* sebagai umpan baliknya. *Set Point* merupakan nilai referensi yang akan digunakan sebagai batas yang wajib dicapai oleh sistem motor DC.

Sensor *encoder* ini akan membaca nilai kecepatan motor, lalu dari pembacaan tersebut dibandingkan dengan nilai referensi (*set point*) sehingga menghasilkan nilai error[7]. Nilai *error* ini digunakan

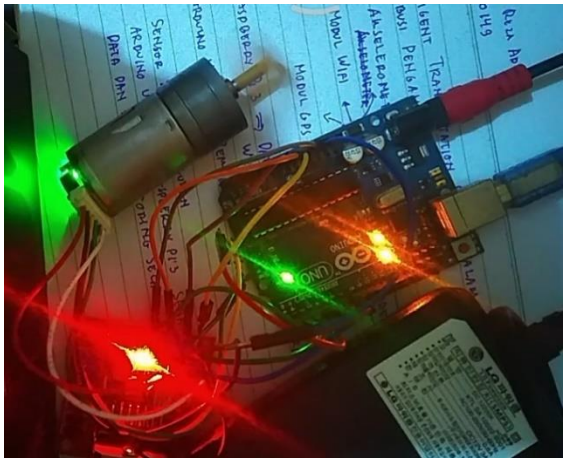
sebagai masukkan pengendali PID agar dapat mengontrol kecepatan motor DC sehingga mencapai nilai referensi. Nilai yang keluar setelah proses pengendali PID adalah Pulse Width Modulation (PWM), melalui *mikrokontroller* Arduino UNO nilai PWM 8bit dengan nilai antara 0-255 dikirimkan ke *driver* motor L298N sehingga nilai PWM berubah dan juga nilai tegangan digital (PWM) dikodifikasi menjadi tegangan analog sehingga kecepatan motor DC berubah mengikuti hasil keluaran kendali PID. Proses ini akan terus-menerus (kontinu) hingga kecepatan motor DC mencapai nilai referensi (*Set point*) yang ditentukan.

Skema Pengkabelan Rangkaian

Gambar 3 merupakan skema pengkabelan rangkaian, terlihat bahwa sumber tergangan didapat dari adaptor 12V. Konfigurasi PIN untuk masing-masing perangkat terlihat pada tabel 1, yang dimana motor driver L298N membutuhkan tegangan 5V sebagai sumber tegangan rangkaian, dan juga membutuhkan tegangan 12V sebagai sumber tegangan untuk motor DC. PIN 6, 7, dan 8 yang terhubung dengan driver motor memiliki fungsi untuk mengatur kecepatan motor dengan memanfaatkan PWM (PIN 6), dan juga memiliki fungsi untuk mengatur arah putaran motor DC (PIN 7 dan 8). Driver motor L298N terhubung dengan motor DC yang dilengkapi sensor encoder. Sensor encoder membutuhkan tegangan 3.3V, data encoder terhubung dengan PIN 2 dan 3.



Gambar 3. Skema Pengkabelan Rangkaian



Gambar 4. *Hardware Setup*

Tabel 1. Arduino Uno Pin Number

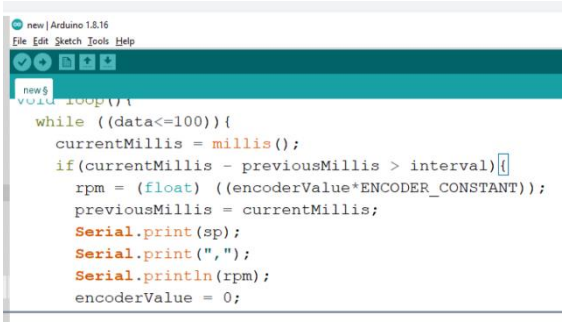
Pin Number	I/O Function
2	Encoder A
3	Encoder B
6	PWM Pin
7	Motor Driver Direction
8	Motor Driver Direction

Rumus Pengukur Kecepatan

Kecepatan putaran diperoleh dengan menghitung jumlah putaran dalam satu menit. Namun, pada percobaan pada penelitian ini menggunakan waktu sampel sebesar 100ms. Sehingga persamaan dapat dituliskan pada (1), dimana jumlah putaran (r) dan waktu sampel (t).

$$\omega = \frac{r}{t} \quad (1)$$

Encoder yang terpasang pada ujung motor DC berguna untuk menghitung jumlah pulsa yang dikeluarkan oleh motor DC pada saat berputar. Motor DC yang digunakan mengeluarkan sebanyak 100 pulsa dalam kurun satu putaran, sehingga untuk menghitung jumlah putaran (r) diperoleh dari (2) dimana p adalah jumlah pulsa, dan P_R adalah jumlah pulsa saat motor diputar 360°.



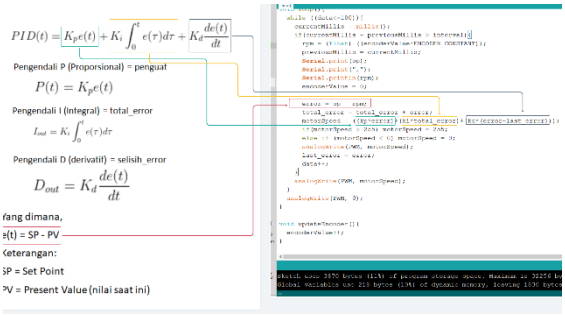
Gambar 5. Implementasi Rumus Pengukur Kecepatan pada Arduino IDE

$$r = \frac{p}{P_R} = \frac{p}{100} \quad (2)$$

Karena kecepatan dihitung dalam menit, maka perlu dilakukan konversi pada waktu sampel. Maka konversi didapatkan dari (3) dimana t_s adalah waktu sampel dalam ms, yaitu 100ms.

$$t = \frac{t_s}{1000 * 60} = \frac{100}{1000 * 60} = \frac{1}{600} \quad (3)$$

Sehingga didapatkan rumus untuk menghitung kecepatan motor DC pada (4).



Gambar 6. Implementasi Pengendali PID pada Arduino IDE

$$\omega = \frac{r}{t} = \frac{\frac{p}{100}}{\frac{1}{600}} = \frac{p}{100} * \frac{600}{1} = \frac{600p}{100} = 6p \quad (4)$$

Persamaan (4) menunjukkan bahwa untuk mendapatkan kecepatan dalam rotasi per menit (RPM) perlu mengalikan jumlah pulsa dengan nilai konstanta 6 (ENCODER_CONSTANT = 6).

Pengendali PID

Pengendali PID dapat ditulis dalam bentuk persamaan sebagai berikut [8], [9].

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \quad (5)$$

dimana Kp, Ki, dan Kd merupakan nilai parameter *proportional*, *integral*, *derivative*. e(t) merupakan nilai kesalahan yang diperoleh dari perhitungan selisih antara nilai yang diinginkan (set point) dengan nilai keluaran sistem, dalam hal ini adalah kecepatan motor DC yang dibaca oleh *sensor encoder*.

Pengendali PID memiliki karakteristik memanfaatkan umpan balik sistem untuk dibandingkan dengan nilai setpoint dan berjalan secara kontinu serta pengendali tersebut mencoba untuk meminimalkan nilai kesalahan sehingga dapat mempengaruhi respon sistem[10]. Pengendali *proportional* memiliki karakteristik penguatan, misalnya jika nilai kesalahan besar maka nilai tersebut akan dikalikan dengan parameter Kp sehingga nilai kesalahan menguat dan pengendali akan merespon dengan keluaran lebih tinggi. Sedangkan pengendali integral mengakumulasikan nilai kesalahan keseluruhan yang tersisa sebelumnya, lalu dikalikan dengan parameter Ki dan pengendali berusaha menghilangkan nilai kesalahan sisa tersebut. Lalu pengendali *derivative* mencoba mengambil selisih dari nilai kesalahan saat ini dengan nilai kesalahan sebelumnya untuk mengetahui rentang perubahan kesalahan saat ini, semakin cepat tingkat perubahan dari nilai

kesalahan, maka semakin besar efek pengendalian.

3. HASIL DAN PEMBAHASAN

Pengujian Sistem tanpa Pengendali PID dengan nilai PWM Bervariasi

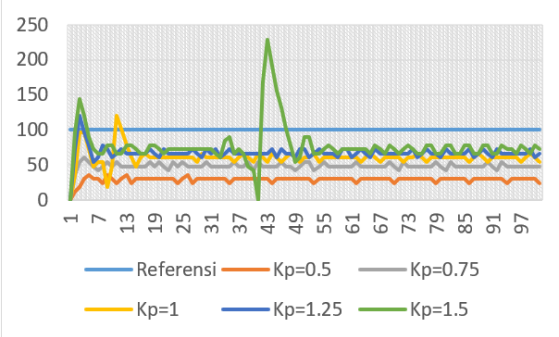
Gambar 7 merupakan grafik dari respon sistem tanpa pengendali PID dengan sumbu X yaitu domain waktu dan sumbu y adalah kecepatan sudut dengan nilai PWM bervariasi. Nilai yang bervariasi ini ditentukan pada rentang 0-255 dikarenakan Arduino Uno menggunakan PWM 8 bit.

Gambar 7. Grafik Hasil Pengujian Sistem tanpa Pengendali PID dengan nilai PWM Bervariasi

Berdasarkan hasil pengujian dengan nilai PWM konstan bervariasi, minimum PWM yang dapat diberikan pada sistem ini adalah 50, apabila di bawah itu, motor DC hanya berdengung dan tidak berputar. Lalu, semakin tinggi nilai PWM maka kecepatan motor DC semakin meningkat sehingga jika nilai referensi (set point) yang diinginkan sebagai acuan dalam bentuk RPM, maka cara tersebut kurang efisien karena pengkalibrasian perlu dilakukan ulang apabila nilai RPM yang dituju berubah. Karena itu dibutuhkan pengendali untuk mencapai nilai set point yang diinginkan.

Pengujian Parameter Pengendali PID yang diterapkan pada motor DC

Pengendali *proportional*, *integral*, dan *derivative* memiliki masing-masing karakteristik berbeda yang dapat mempengaruhi respon sistem, karena itu dilakukan pengujian masing-masing parameter untuk mengetahui pengaruhnya dengan nilai referensi (*set point*) yang sama, yaitu 100RPM. Pengujian pertama dilakukan penerapan pengendali *Proportional* (P) dengan hasil yang ditunjukkan pada gambar 8.



Gambar 8. Grafik Hasil Pengujian Sistem Pengendali P dengan nilai Kp Bervariasi

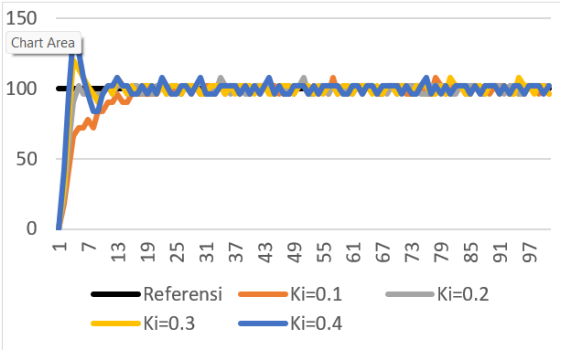
Tabel 2. Respon Sistem Pengendali Proportional (P)

Kp	Ki	Kd	Rise Time	Settling Time	Overshot (%)	Steady-state Error
0.5	0	0	-	-	0	76
0.75	0	0	-	-	0	52
1	0	0	1.62	-	20	46
1.25	0	0	1.23	-	20	34
1.5	0	0	0.83	-	128	28

Grafik respon sistem setelah diterapkannya pengendali *Proportional* yang ditunjukkan pada gambar 8 dan tabel 2, dapat dikenali bahwa pengendali *proportional* dapat mempengaruhi penurunan nilai *steady state error*, nilai *rise time*, tetapi dapat meningkatkan nilai *overshot*. Namun, nilai *settling time* tidak ada dikarenakan respon sistem tidak mencapai nilai *set point* yang diinginkan (100RPM).

Pengujian kedua dilakukan untuk mengenali karakteristik dari pengendali integral (I) terhadap respon sistem. Hasilnya dapat dilihat pada gambar 9 dan tabel 3 yang menunjukkan bahwa pengaruh pengendali integral berefek pada minimalisasi nilai *steady state error* yang sangat signifikan dan penurunan nilai *rise*

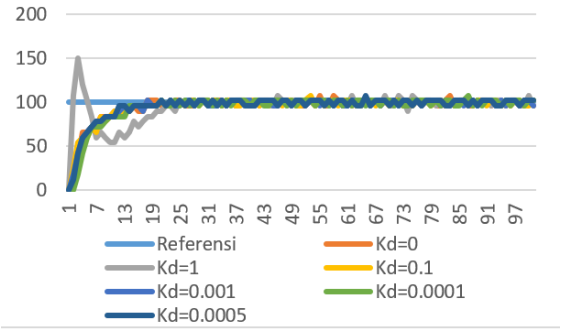
time. Namun pemakaian nilai parameter Ki yang terlalu besar menyebabkan semakin besar pula *overshot* dan *undershot*.



Gambar 9. Grafik Hasil Pengujian Sistem Pengendali I dengan nilai Kp=0.5 dan nilai Ki Bervariasi

Tabel 3. Respon Sistem Pengendali Integral (I)

Kp	Ki	Kd	Rise Time	Settling Time	Overshot (%)	Steady-state Error
0.5	0.1	0	10.44	-	8	4
0.5	0.2	0	2.58	96.3	8	2
0.5	0.3	0	1.95	-	20	46
0.5	0.4	0	1.65	100	38	34



Gambar 10. Grafik Hasil Pengujian Sistem Pengendali D dengan nilai Kp=0.5; Ki=0.1; dan nilai Kd Bervariasi

Tabel 4. Respon Sistem Pengendali Derivative (D)

Kp	Ki	Kd	Rise Time	Settling Time	Overshot (%)	Steady-state Error
0.5	0.1	0	12.44	-	8	4
0.5	0.1	0.0001	10.94	96.3	8	4
0.5	0.1	0.0005	9.66	98	8	2
0.5	0.1	0.001	10.44	-	2	4
0.5	0.1	0.1	12.58	100.3	8	4
0.5	0.1	1	0.74	-	50	4

Pengujian terakhir yaitu pengujian parameter pengendali *Derivative* (D) dengan hasil yang ditunjukkan oleh grafik respon sistem pada gambar 10 dan tabel 4. Berdasarkan hasil pengujian, pengendali *derivative* ini dapat digunakan untuk mengurangi nilai *overshot* yang berlebihan, mengurangi nilai *rise time*, namun *undershot* dapat meningkat ketika nilai *Derivative* membesar.

4. SIMPULAN

Respon sistem yang baik memiliki karakteristik nilai *steady-state error* sekecil mungkin, nilai *rise time* sekecil mungkin, nilai *overshot* yang tidak begitu besar, dan nilai *settling time* yang kecil. Maka dari itu kendali PID dapat memperbaiki respon sistem tersebut, dengan catatan, mengetahui pengaruh masing-masing parameter Kp, Ki, Kd terhadap respon sistem jika diganti-ganti. Tabel 5 merupakan rangkuman dari karakteristik masing-masing pengendali.

Tabel 5. Rangkuman Respon Sistem Pengendali PID

	Rise Time	Settling Time	Steady State Error	Overshot	Undershot
Pengendali P	-		-	+	
Pengendali I	-		-	+	
Pengendali D			-	-	+

Namun, jika sistem ini dibandingkan dengan sistem motor DC tanpa pengendali PID, maka dapat disimpulkan bahwa

- Jika sistem yang ingin dibuat acuannya adalah kecepatan putaran sudut (RPM), butuh banyak waktu/percobaan untuk mengatur nilai PWM agar sesuai dengan RPM yang kita inginkan. Sedangkan sistem yang menggunakan kendali PID, hanya perlu mengatur set point RPM yang diinginkan.
- Tanpa kendali PID, ada banyak steady-state error. Sedangkan jika menggunakan kendali PID, kita hanya

perlu mengatur parameter KP, Ki, Kd untuk mengurangi steady-state error tersebut.

- Jika sistem yang ditentukan ingin mengganti RPM, motor DC tanpa kendali PID memerlukan penyesuaian ulang secara manual. Sedangkan kendali PID yang diterapkan pada motor DC hanya perlu mengganti nilai set point nya.

DAFTAR PUSTAKA

[1] T. J. D. Barnes, L. Wang, and W. R. Cluett, "Frequency domain design method for PID controllers," *Am. Control Conf.*, no. 2, pp. 890–894, 1993, doi: 10.23919/acc.1993.4792991.

[2] W. Purbowaskito and C.-H. Hsu, "Sistem Kendali PID untuk Pengendalian Kecepatan Motor Penggerak Unmanned Ground Vehicle untuk Aplikasi Industri Pertanian," *J. Infotel*, vol. 9, no. 4, p. 376, 2017, [Online]. Available: http://ejournal.st3telkom.ac.id/index.php/infotel/article/view/253

[3] R. Muhardian and K. Krismadinata, "Kendali Kecepatan Motor DC Dengan Kontroller PID dan Antarmuka Visual Basic," *JTEV (Jurnal Tek. Elektro dan Vokasional)*, vol. 6, no. 1, pp. 328–339, 2020, [Online]. Available: http://ejournal.unp.ac.id/index.php/jtev/index

[4] N. N. ROKHMAH, "KENDALI KECEPATAN MOTOR DC DENGAN METODE PID BERBASIS ARDUINO UNO," 2018. [Online]. Available: http://repository.unjani.ac.id/index.php?p=show_detail&id=1031&keywords=

[5] R. Arindya, "Penalaan Kendali PID untuk pengendali proses," *J. Teknol. Elektro*, vol. 8, no. 2, p. 109, 2017.

[6] I. Q. Rosalina Rosalina and M. Mujirudin, "Analisis Pengaturan Kecepatan Motor DC Menggunakan Kontrol PID (Proportional Integral Derivative)," *Pros. Semin. Nas. Teknoka*, vol. 2, pp. 89–94, 2017, [Online]. Available: https://journal.uhamka.ac.id/index.php/teknoka/article/view/782

[7] R. A. Ardiansyah, "Perancangan dan Pengujian Sistem Pengendali Sudut untuk Motor DC Brushless Menggunakan Kendali Algoritma P-D," *J. Rekayasa Elektr.*, vol. 13, no. 2, p. 82, 2017, doi: 10.17529/jre.v13i2.7149.

[8] E. Apriaskar, F. Fahmizal, N. A. Salim, and D. Prastiyanto, "Performance Evaluation of Balancing Bicopter using P, PI, and PID Controller," *J. Tek. Elektro*, vol. 11, no. 2, pp. 44–49, 2019, doi: 10.15294/jte.v11i2.23032.

[9] D. Irawan and P. Perdana SS, "Kontrol Motor Brushless DC (BLDC) Berbasis Algoritma AI - PID," *J. Tek. Elektro dan Komputasi*, vol. 2, no. 1, pp. 41–48, 2020, doi: 10.32528/elkom.v2i1.3146.

[10] A. Novandri, Roslidar, and A. Rahman, "Rancang Bangun Robot Self Balancing Berbasis Mikrokontroler Atmega328P Dengan Kendali Pid," *Kitektro, J. Online Tek. Elektro*, vol. 2, no. 2, pp. 15–23, 2017.