IT Journal Research and Development (ITJRD)
Vol.6, No.1, August 2021, E-ISSN : 2528-4053 | P-ISSN : 2528-4061
DOI : 10.25299/itjrd.2021.vol6(1).6125

Embedded Control System of DC Motor Using Microcontroller Arduino and PID Algorithm

Alfian Ma'arif[1], Naufal Rahmat Setiawan[2], Eka Suci Rahayu[3]
Department of Electrical Engineering, Faculty of Engineering Technology, Universitas Ahmad Dahlan
alfianmaarif@ee.uad.ac.id[1], Naufalrachmat98@gmail.com[2], eka1600022027@webmail.uad.ac.id[3]

Article Info

History:

Received Dec 25, 2020
Revised Jan 5, 2021
Accepted Mar 8, 2021

Keywords:

DC Motor
Angular Speed
Arduino
PID Controller
Encoder Sensor
Angular Velocity

ABSTRACT

The Direct Current (DC) motors have many applications, especially in robotics and industry. The most popular control method for controlling a DC motor is Proportional-Integral-Derivative Control (PID). DC motor control simulation has good performance. However, simulation is an ideal situation and is likely to differ from hardware implementations. This study proposes hardware design and implementation of DC motor angular speed control on Arduino Uno as an embedded control system using a PID controller. Other frames are the L298 Motor Driver and the JGA 25-370 DC Motor (including the encoder sensor). Based on testing, the PID controller has been successfully implemented into the Arduino UNO and can control the angular speed of a DC motor. System performance (in the form of system response) differs based on the choice of PID value parameters (Proportional, Integral, Derivative parameters) and sampling time. A proportional controller speeds up the rise time and increases overshoot. The integral controller eliminates steady-state errors and increases overshoot. Derivative control reduces overshoot a little. The best PID parameters were KP = 0.7, KI = 0.3, and KD = 0.2 in a sampling time of 50ms, with the characteristics of the system response has no overshoot, no undershoot fast rise time, and fast, stable time. Compared to systems without PID control, systems with PID control have the advantage of being able to reach the reference value even if the reference value changes.

Corresponding Author:

Alfian Ma'arif,
Department of Electrical Engineering, Faculty of Industrial Technology
Universitas Ahmad Dahlan,
Bantul, Daerah Istimewa Yogyakarta, Indonesia
Email: alfianmaarif@ee.uad.ac.id

1. INTRODUCTION

Direct Current (DC) motor is a device that converts DC electrical energy into mechanical energy. DC motors have many applications in the field of robotics [1][2][3] and industrial applications [4][5][6]. Some examples of implementation are tracking systems for the direction of the sun [7], balance robot [8][9], line follower robot [10][11], line maze robot [12][13], Quadrotor [14][15], temperature control [16], Omni Robot [17] etc. DC motors are very popular because they are easy to learn, easy to control, and have good performance.

The DC motor system must follow the given reference value and be stable at the reference value. This problem can be overcome by several methods such as using Proportional Derivative (PD) control [18], Proportional Integral Derivative (PID) [19][20][21], Fuzzy Logic Control (FLC) [22], Fuzzy Model Reference Learning Control (FMRLC) [23], State Feedback Controller [24], Linear Quadratic Regulator (LQR) [25] or Neural Network Controller [26][27][28]. In simulation, problems can be solved with good results and performance [29][30][31][32][33][34]. However, simulation has ideal conditions so that real implementation on hardware can produce different performance because of many factors that influence the hardware implementation design.

This study proposes a DC motor hardware design with a low-cost embedded system device, namely the Arduino Uno [35][36][37]. The PID controller is implemented in the system as an angular velocity controller (omega). PID control was chosen because it has advantages in its characteristics, namely easy to understand, simple but has good system response performance, and easy to implement in both software simulation and hardware implementation. [38][39][40][41][42].

2. METHODS

2.1. System Design

The block diagram of the embedded system for controlling a DC motor's speed is shown in Figure 1. At the same time, the system wiring diagram is shown in Figure 2. The system consists of input devices, processors (processors), output devices, and interface devices. The input device is an encoder sensor that functions to measure the angular speed of a DC motor. The output devices are the L298 motor driver and the JGA25-370 DC Motor. The processing device is the Arduino Uno. The interface device is a serial monitor or serial plotter from the Arduino IDE.

The encoder sensor sends pulse data to the Arduino Uno to calculate the Radian Per Minute angular velocity value (RPM). The features used by the Arduino Uno to process angular velocity are a timer and counter. Angular velocity data is sent to Serial Monitor and Serial Plotter via USB serial. Pulse Width Modulation (PWM) is used to adjust the input voltage to the DC motor so that the DC motor's speed can be varied. The motor driver functions to convert the digital voltage (PWM) into an analog voltage with a higher voltage level (5 volts to 10 volts).

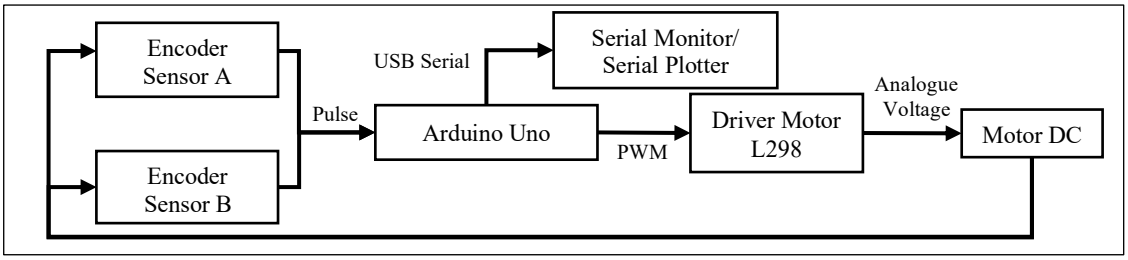


Figure 1. System Block Diagram

The configuration of the input PIN and output PIN can be seen in Figure 2. The Encoder sensor is connected to PIN 2 and 3. Both PINs have a counter feature to count pulses. The motor driver is connected to PIN 6, 7, and 8. PIN 6 functions to adjust the motor's angular speed with the Pulse Width Modulation (PWM) feature. At the same time, PIN 7 and 8 function to regulate the direction of rotation of the motor (clockwise or counterclockwise).

The voltage source is obtained from the Power Supply Unit (PSU) with a voltage of 12V. The encoder sensor requires a voltage of 3.3V, which is obtained from the Arduino Uno minimum system. The L298 Motor Driver requires a 5V and 12V voltage source. In the motor driver, the 5V voltage functions as the electronic circuit voltage source, while the 12V voltage is the DC motor voltage source.

The control system block diagram is shown in Figure 3. This system is categorized into a closed-loop control system. Setpoint block is a reference value that the system must follow. The PID controller is a Proportional-Integral-Derivative Control. The system to be controlled is a DC motor. The system output is angular velocity. The feedback uses an encoder sensor to calculate the angular velocity of the DC motor system.

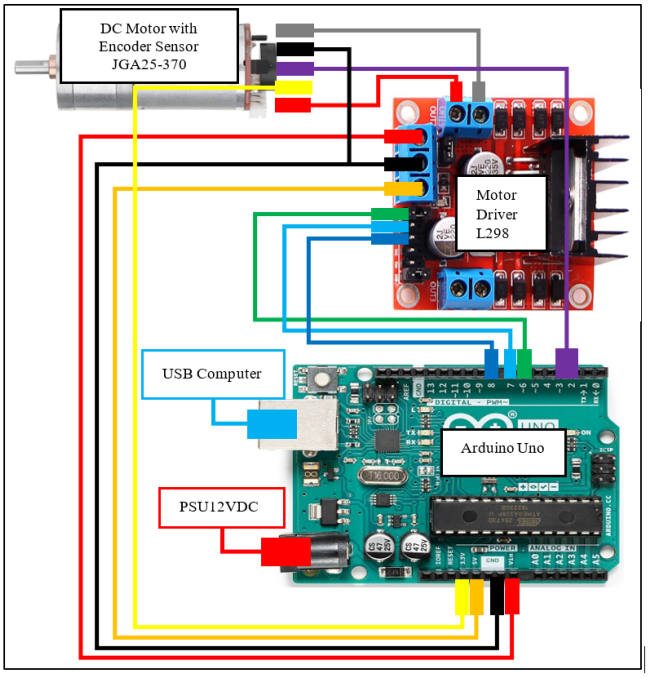


Figure 2. System Wiring Diagram

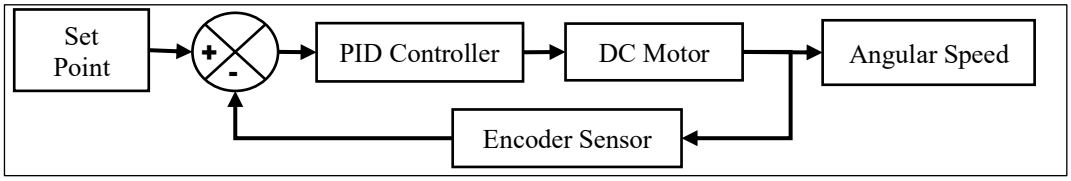


Figure 3. Control System Block Diagram

2.2. Angular Velocity Counter

The angular velocity can be obtained by counting the pulses from the encoder sensor in one minute. The angular speed is calculated using an encoder mounted on the end of the DC motor. The angular velocity calculation can be written as

$$\omega = \frac{r}{t} \tag{1}$$

Where r is the number of rotations on t (sample time). The number of rotations can be achieved from,

$$r = \frac{p}{p_R} \tag{2}$$

where p is the number of pulses in the sample time, the variable p_R is the number of pulses in one turn. According to the motor datasheet, there are 600 pulses in one loop.

The sample time used in the program t_s is 50ms need to be changed with (3) to get time, t, in one minute.

$$t = \frac{t_s}{1000 * 60} \tag{3}$$

The constant 1000 is the conversion from milliseconds to seconds, and 60 is the second to minute conversion constant. Thus, the RPM can be calculated as,

$$\omega = \frac{p}{600} \frac{1000 * 60}{50} = 2p \tag{4}$$

Therefore, it can be obtained that the angular velocity in Rotation per Minute (RPM) is the number of pulses multiplied by 2.

2.3. Derivative Integral Proportional Control (PID)

PID controls consist of proportional, integral, and derivative controls [43][44][45]. The PID Control equation in the time domain is,

$$u(t) = K_P e(t) + \frac{K_p}{T_i}\int_0^t e(t)\,dt + K_p T_d \frac{de(t)}{dt} \quad (5)$$

where $u(t)$ is a control signal, K_p is the proportional control parameter value, T_i is integral time dan T_d is derived time, $e(t)$ is the error or difference between the reference value and the feedback value. The PID controller can be written in another form as [46][47]

$$u(t) = K_P e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt} \quad (6)$$

where

$$K_i = \frac{K_p}{T_i} \quad K_d = K_p T_d \quad (7)$$

The gain constant K_i is the value of the integral control parameter and K_d is the value of the derived control parameter. PID control has characteristics that influence system response. It is because of the different controller structures. Proportional control deals with the error between the reference value and the feedback value. The integral control deals with the sum of all errors. The child control corresponds to the current error with the previous error.

2.4. Algorithm

The software used to create an embedded control program is the Arduino IDE. The Arduino IDE software and the embedded control main program with PID control are shown in Figure 4 (a). Simultaneously, the embedded control program flow chart is shown in Figure 4 (b). The main program has two parts, namely the angular velocity calculation program according to Equation (4) and the PID control program based on Equation (6). The parameters that must be determined before running the system are the reference value (SP) and the PID parameter (KP, KI, KD). The control system will continue to run until the supply is turned off or reaches some data. The PID control will calculate the PWM value sent to the motor. The angular velocity value is sent to computer to be displayed on the computer as a value or graph.

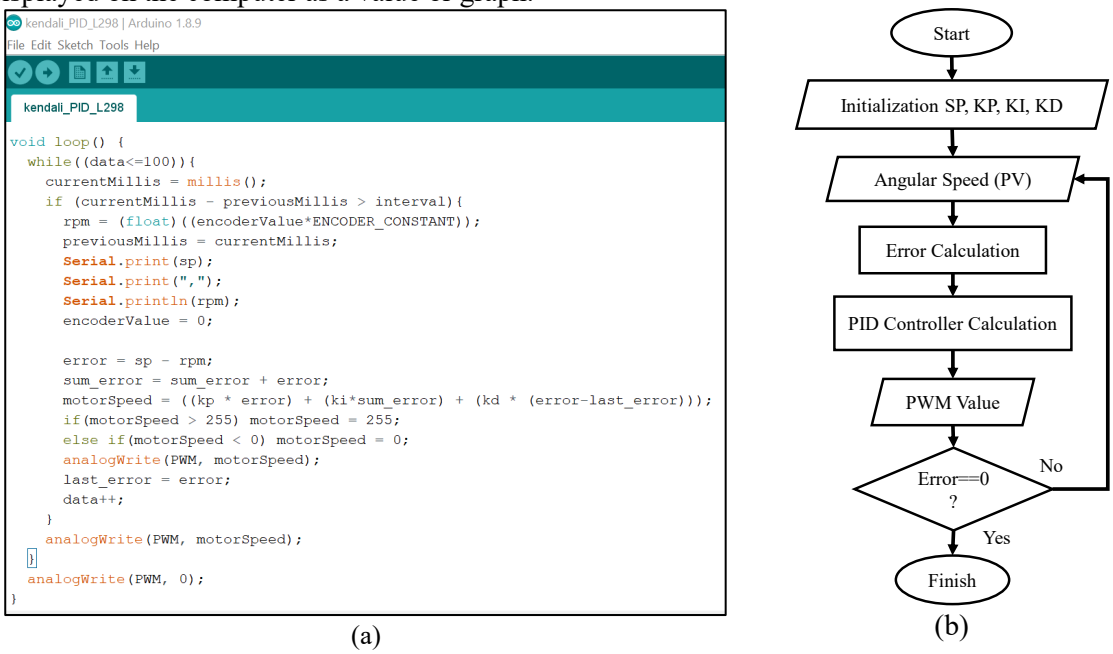


Figure 4. Arduino IDE (a) and PID Control Embedded Control System Flowchart (b)

3. RESULT AND DISCUSSION

In this section, there are several tests as follows. The first part is about the open-loop system response [48]. The second part is about the closed-loop system response of the PID Controller. The third part is about the response to variations in the reference value and the effect of different sample times. The last part compares the systems without and with PID control hardware used in this study is shown in Figure 5.

The price of the Arduino Uno component is IDR 70,000. The price of the L298 motor driver is IDR 25,000. The price of a 25GA370 DC Motor with an encoder sensor is IDR 125,000. The PSU price is IDR 30,000. The connecting cable, holder and plastic box is IDR50,000, so the total component price is IDR300,000. These components' price is lower than using other devices such as the NI-DAQ [49] or PLC [50].

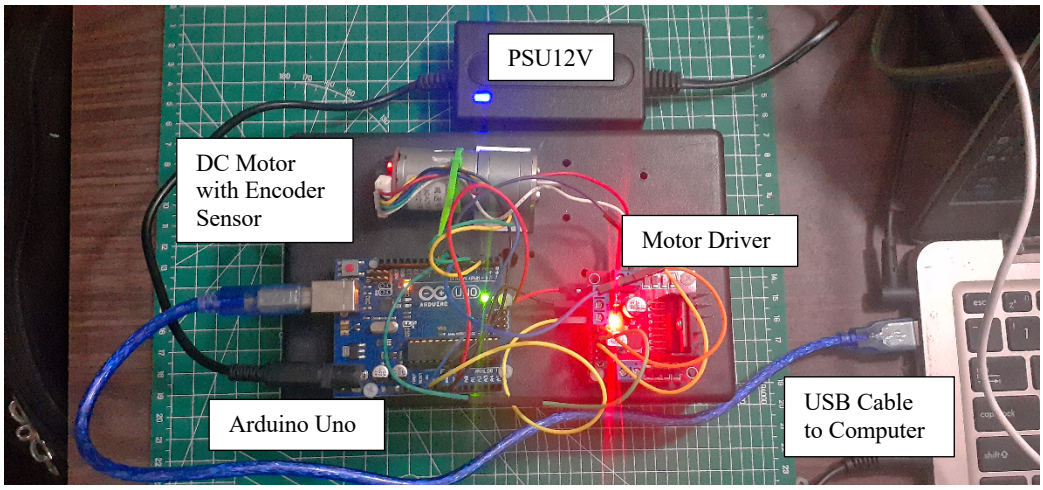


Figure 5. Hardware Setup

System performance is measured using system response. Some of the observed system response parameters are rise time, settling time, overshoot, and steady-state error. The expected system response is that it has a small rise time, has a short stabilization time, has a small overshoot, and the steady-state error is zero.

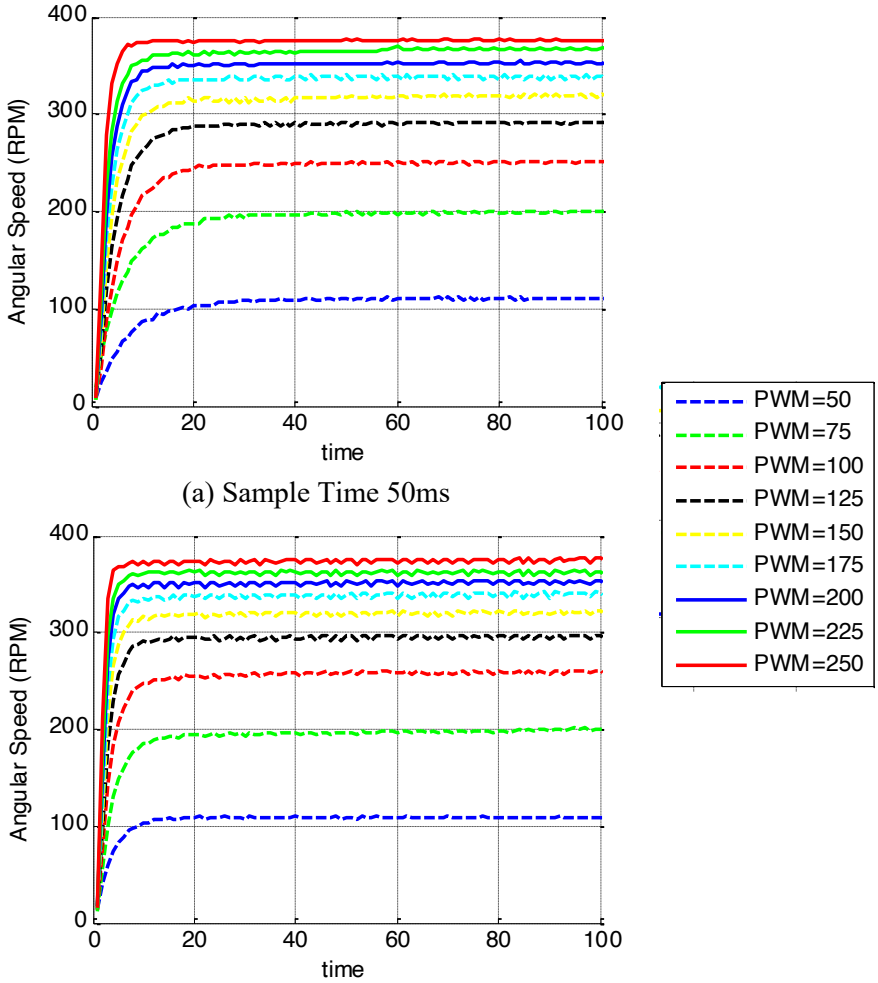
3.1. Open Loop Testing

The results of the implementation of the open-loop test hardware are shown in Table 1 and Figure 6. Arduino Uno uses 8-bit PWM, with a data range between 0-255. The motor driver uses a 12-volt power supply. The maximum voltage to the DC motor is 10 volts measured at the motor driver output. The minimum PWM is 50, and if the PMW is below 50, the DC motor cannot rotate just buzzing. There are two sample times used in this study, namely 50ms and 100ms.

Table 1. The relationship between stress and angular velocity

PWM (8-bit)	Voltage (volt)	RPM Specification Motor	RPM Sample Time 50ms	RPM Sample Time 100ms
50	2.8	98	105	108
75	5.0	175	190	197
100	6.5	227	250	257
125	7.6	266	290	294
150	8.1	283	316	319
175	8.6	301	337	338
200	9.0	315	352	351
225	9.4	329	365	361
250	9.6	336	376	374

Based on Figure 3, the 50ms sample time provides a more stable angular velocity than the 100ms sample time. It can be seen clearly in the system response with PWM = 250 that a sample time of 100ms produces a response with some oscillations (some ripples) after reaching a steady state. Sample time is critical for system accuracy and response speed. Using a smaller sample time, for example, 50ms, the delay for the system to respond to errors is also smaller. Therefore, the system can prevent the output from having errors. The system will then respond more quickly and produce a more stable output.



(a) Sample Time 50ms

(b) Sample Time 100ms

Figure 6. Open-loop system response using different sample times

3.2. DC Motor Step Reference Response with PID Control

The proportional control system response is shown in Table 2 and Figure 7. The reference value is 100RPM. In Table 2, increasing proportional control (KP) can reduce steady-state errors. In Figure 4, it can be seen that the steady-state error is reduced. Proportional control increases the overshoot value and reduces the rise time. Thus, in hardware implementations, it can be seen that proportional control affects the reduction in rising time, increased overshoot, and reduction in steady-state errors.

Table 2. Proportional Control (KP) system response

KP	KI	KD	Rise time	Settling time	Overshoot (%)	Steady State Error
0.5	0	0	-	-	-	64
0.75	0	0	-	-	-	48
1	0	0	-	-	-	38
1.25	0	0	1.7727	-	-	32
1.5	0	0	1.4286	-	12	26

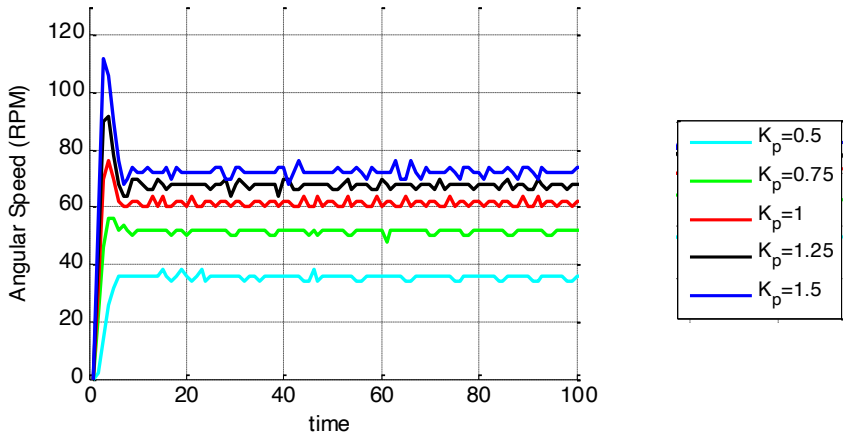


Figure 7. Proportional Control Closed Loop System Response

The response of the integral control system is shown in Table 3 and Figure 8. It can be seen that the increase in integral control (KI) can eliminate steady-state errors and result in faster system response. Larger integral controls have faster ride times but have greater overshoot and undershoot. The change in the value of the integral control that is not too large can make the system experience overshoot and undershoot. Thus, integral control affects increased overshoot, increases undershoot, reduces ride time, and eliminates steady-state errors.

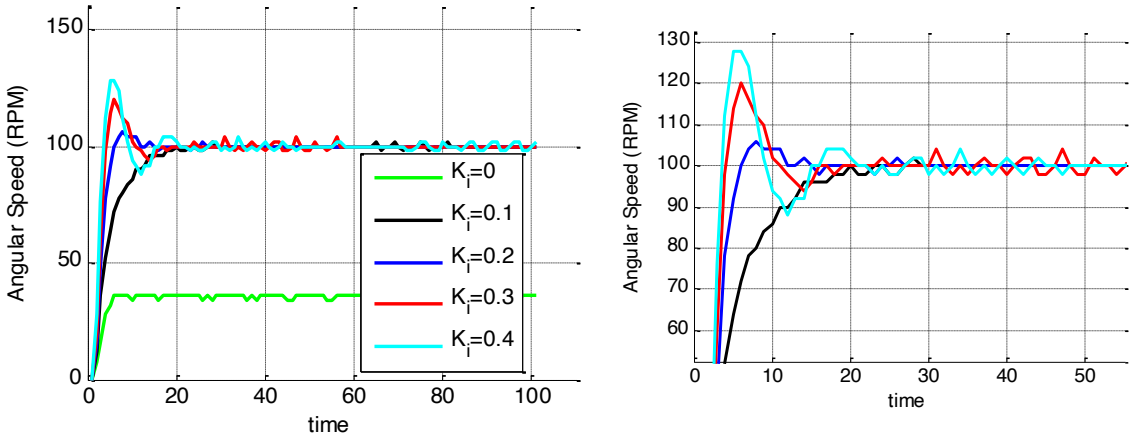


Figure 8. Integral Control Closed Loop System Response

Table 3. Integral Control (KI) system response

KP	KI	KD	Rise time	Settling Time	Overshoot (%)	Steady State Error
0.5	0	0	-	-	-	64
0.5	0.1	0	9.1667	18	2	0
0.5	0.2	0	3.2321	11.5	6	0
0.5	0.3	0	2.2895	56.5	20	0
0.5	0.4	0	2.0317	34.5	28	2

The response of the derivative controller is shown in Table 4 and Figure 9. It can be seen that increasing derivative control (KD) can reduce overshoot but make the system experience undershoot. The larger the derivative control, the greater the undershoot value. Descent control can increase the ride time but can also reduce the ride time after the undershoot appears. Thus, derivative control effects reducing overshoot, reducing rise time, and increasing undershoot.

Table 4. Derivative Control (KD) system response

KP	KI	KD	Rise time	Settling time	Overshoot (%)	Steady State Error
0.6	0.3	0	2.2857	60.5	16	2
0.6	0.3	0.1	2.3875	10.5	8	0
0.6	0.3	0.2	2.2778	11.5	6	2
0.6	0.3	0.3	2	13.3333	6	0
0.6	0.3	0.4	1.7311	97.3333	4	0

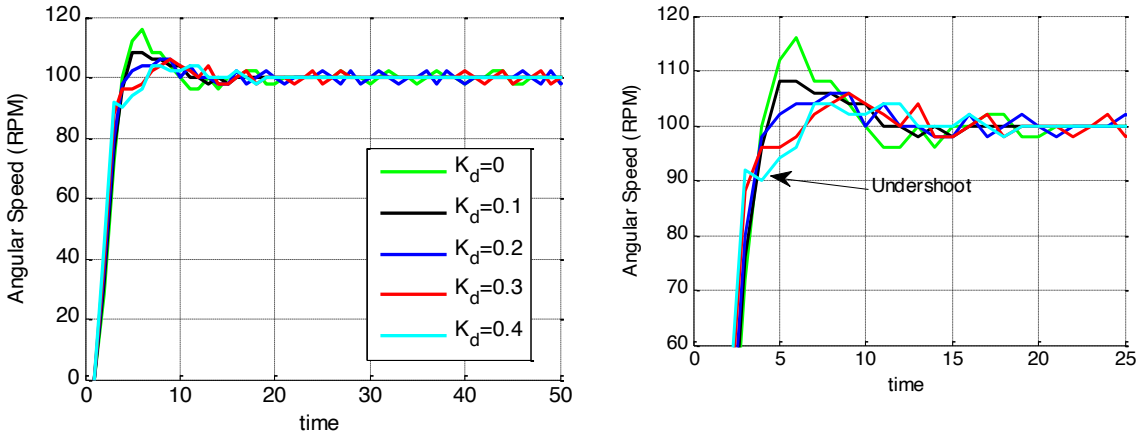


Figure 9. Closed-Loop System Response Derivative Control

Table 5 summarizes the characteristics of the PID controller based on the implementation of the hardware system response. Proportional control and integral control are suitable for reducing ride time but have the risk of increasing overshoot. The best function of proportional control is to reduce ride time, and the best function of integral control eliminates steady-state errors. Derivative controls are suitable for reducing overshoot but have the risk of increasing the undershoot. Accordingly, derivative control should not be overestimated.

Table 5. PID Control system response

	Rise Time	Settling Time	Overshoot	Undershoot	Steady State Error
Proportional Control	Reduce	-	Increase	-	Reduce
Integral Control	Reduce	-	Increase	-	Missing
Derivative Control	-	-	Increase	Increase	-

3.3 Testing Reference Value and Sampling Time

The best PID parameters are shown in Table 6 and Figure 10. The setpoint value is 100RPM. The best PID control parameter (KP, KI, KD) is number 4. System response numbers 2 and 3 have undershot. That's because of the great descent controls. Increased derivative control must be careful because it will give a decreased response. The overshoot response is given by number 1 because of its large proportional value.

Table 6. System response from PID Parameter controllers (KP, KI, KD)

No.	KP	KI	KD	Rise Time	Settling Time	Overshoot	Steady State Error
1	0.8	0.3	0.1	1.7115	88.3333	8	0
2	0.75	0.3	0.3	1.6317	86.3333	4	0
3	0.75	0.3	0.25	1.6573	66.3333	4	2
4	0.7	0.3	0.2	1.9167	14.5	2	2

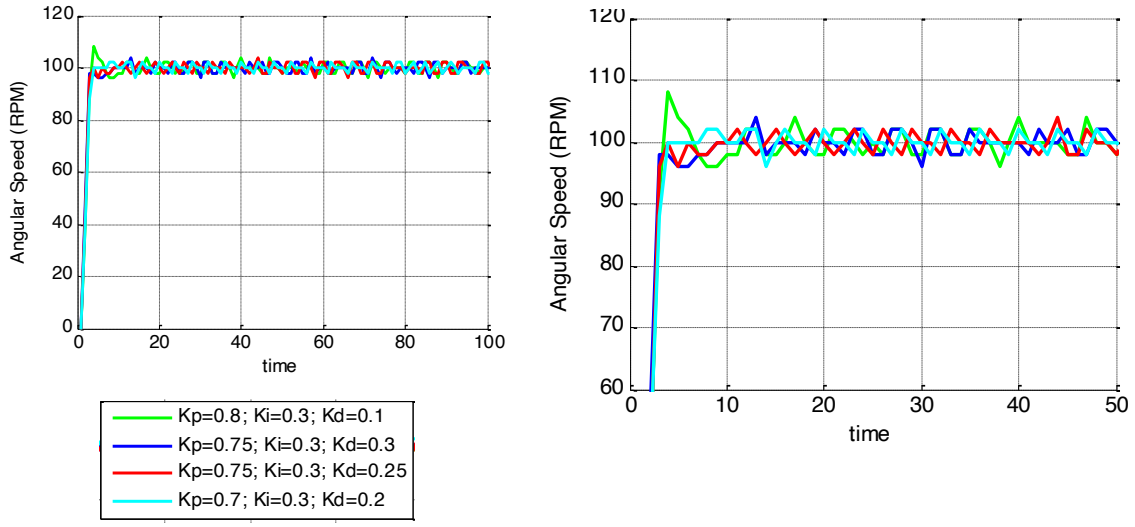


Figure 10. PID Controller Closed Loop System Response

The next test is the variation of the reference value and the response time of the sample. There are two sample times used in this study, 50ms and 100ms. The results are shown in Table 7 and Figure 11. The PID controller can control and stabilize the system at multiple set points and achieve a reference signal. Sample time affects system response but can still follow set points. 50ms sample time provides a faster response than 100ms response. Thus, a smaller sample time is good for faster system response. However, it should not be too small as it can eliminate the original characteristics of the angular velocity data.

Table 7. System response to various reference values and sampling times

SP	KP	KI	KD	Sampling time = 50ms				Sampling time = 100ms			
				Rise time	Settling time	Over-shoot	Steady-state error	Rise time	Settling time	Over-shoot	Steady-state error
50	0.7	0.3	0.2	3.25	-	4	2	5.27	73.5	4	0
100	0.7	0.3	0.2	1.74	67.33	4	0	1.25	10	20	2
150	0.7	0.3	0.2	1.64	6.75	6.67	0	1.12	11.33	26.67	0
200	0.7	0.3	0.2	1.45	5.5	9	0	0.86	11.57	32	1
250	0.7	0.3	0.2	1.61	7.25	2.4	0	1.04	10	20	2
300	0.7	0.3	0.2	2.09	12	3.33	2	1.42	7.5	5.67	2

3.4. Comparison of Systems without and with PID Control

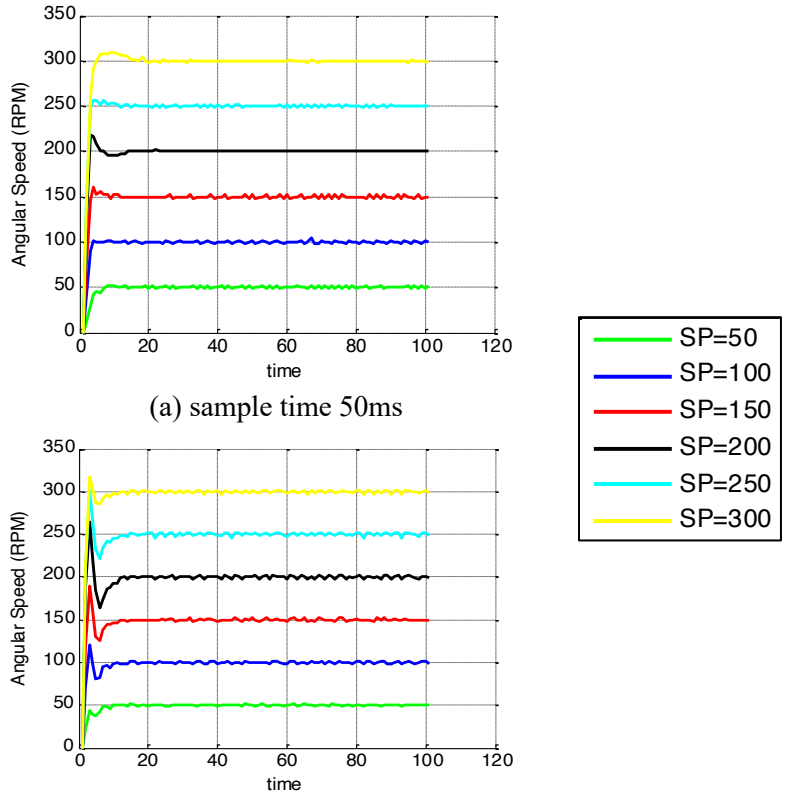
This section describes the comparison of systems without and with PID control which is summarized in Table 8. A system without PID control is shown in Figure 12 (a), and a system with PID control is shown in Figure 12 (b). The most visible thing from comparing the two systems is that the performance reaches the reference value. Systems without PID control need time to set the correct PWM value for the motor's angular speed to reach the reference value. It takes a lot of experimentation to reach the reference value, especially if the reference value has to be changed. Meanwhile, systems with PID control can easily reach the reference value, even with reference value changes. Systems with PID control can automatically adjust to the reference value, while systems without PID control have to adjust manually.

A system without a PID control cannot withstand disturbances which cause the angular velocity to not match the reference value. Meanwhile, systems with PID control are resistant to interference to maintain the DC motor's angular speed according to the reference value. Systems without a PID control will have a steady-state error value if there is a change in the reference value or interruption. Meanwhile, PID-controlled systems do not have steady-state error values. Therefore,

a system with PID control is better than a system without PID control in system performance in achieving the reference value.

Table 8. Comparison of systems without and with PID control

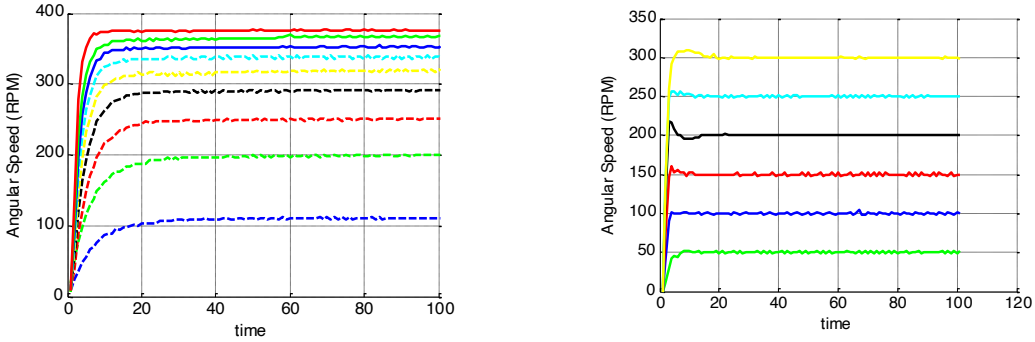
	Reference Value	Steady State Error	Change in Reference Values and Crashes
Without PID Control	It takes a lot of experimentation to set the PWM value manually	There is a steady-state error value	Re-adjustment by manual is required
With PID Control	It is easy to reach the reference value in an automatic way	There is no steady-state error value	Easily reach reference values that change automatically



(a) sample time 50ms

(a) 100ms sample time

Figure 11. Sampling Time Variation Response and Reference Value Using PID Control



(a) Without PID Control (b) With PID Control

Figure 12. Comparison of Systems without and with PID Control

4. CONCLUSIONS

This study proposes controlling a DC motor system using Proportional Integral Derivative Control (PID) using the embedded Arduino Uno system. The PID controller can control and stabilize DC motors in the Embedded System using the Arduino Uno. The system can achieve different reference values with a steady time of under one second. Proportional control has the characteristic of reducing rise time but increasing overshoot. Integral control has the characteristics of eliminating steady-state errors and increasing overshoot. Derivative control has the characteristic of reducing overshoot but increasing undershoot. Shorter sample times provide a faster and more stable system response. However, the sample time should not be too short of giving the original characteristics of the output. The best PID controller for 100RPM reference is KP = 0.7; KI = 0.3; KD = 0.2 with a sample time of 50ms. Comparison with systems without PID control, systems with PID control have the advantage of easily reaching the reference value even when there is a change in the reference value.

REFERENCES

[1] Kuncoro, W. Mulyo Utomo, S. Winardi, and K. Eko Susilo, "Perancangan Kontroller Proportional Integral Derivative Robot Segway Berbasis Mikrokontroler Arduino Nano," Jun. 2019.
[2] D. Dairoh, M. Khambali, and T. Mustofa, "Implementasi Fuzzy Logic dalam Pembuatan Kontrol Navigasi Mobile Robot," *Jurnal Fisika Flux*, vol. 16, no. 1, p. 9, May 2019.
[3] P. W. A. Sucipto and A. Firasanti, "Pengendali PID untuk Pengaturan Kecepatan Gerak Robot Omnidireksional Tiga Roda," *TELKA - Telekomunikasi, Elektronika, Komputasi dan Kontrol*, vol. 6, no. 1, pp. 66–74, May 2020.
[4] F. A. Aziz and R. D. Puriyanto, "Rancang Bangun Mesin Pengecat Dinding Otomatis Berbasis PLC CP1E-NA20DR-A," *Buletin Ilmiah Sarjana Teknik Elektro*, vol. 1, no. 3, pp. 118–130, 2019.
[5] W. Purbowaskito and C.-H. Hsu, "Sistem Kendali PID untuk Pengendalian Kecepatan Motor Penggerak Unmanned Ground Vehicle untuk Aplikasi Industri Pertanian," *Jurnal Infotel*, vol. 9, no. 4, pp. 376–381, 2017.
[6] A. S. Arifin and R. D. Puriyanto, "Rancang Bangun Pemberian Pakan Ayam Petelur Otomatis Menggunakan PLC," *Buletin Ilmiah Sarjana Teknik Elektro*, vol. 1, no. 1, p. 19, 2019.
[7] S. Azmi, Y. Away, and I. Devi Sara, "Kajian Aspek Kecepatan dan Ketepatan pada Sun Tracker Dua Sumbu Berbasis Sensor Berbentuk Tetrahedron," *Jurnal Rekayasa Elektrika*, vol. 15, no. 2, pp. 75–156, Sep. 2019.
[8] N. Tamami, I. M. Diin, B. Sumantri, and E. Pitowarno, "Robot Keseimbangan Beroda Dua dengan Sistem Kontrol Keseimbangan dan Posisi Menggunakan Metode PID Bertingkat," *Jurnal Rekayasa Elektrika*, vol. 14, no. 3, pp. 181–188, Dec. 2018.
[9] A. Maarif, R. D. Puriyanto, and F. R. T. Hasan, "Robot Keseimbangan dengan Kendali PID dan Kalman Filter," *IT Journal Research and Development (ITJRD)*, vol. 4, no. 2, pp. 117–127, 2020.
[10] N. E. Budiyanta, H. Tanudjaja, and M. Mulyadi, "Rancang Bangun Robot Line Follower Portable Sebagai Upaya Minimalisasi Sampah Elektronik di Ranah Robotika," *TESLA: Jurnal Teknik Elektro*, vol. 20, no. 2, p. 148, Feb. 2019.
[11] R. Ridarmin, F. Fauzansyah, E. Elisawati, and E. Prasetyo, "Prototype Robot Line Follower Arduino Uno Menggunakan 4 Sensor TCRT5000," *INFORMATIKA Jurnal Informatika, Manajemen dan Komputer*, vol. 11, no. 2, p. 17, Dec. 2019.
[12] R. Pranata, R. Tri Wahyuni, and Tianur, "Robot Line Maze Pencari Jalur Tercepat," *Jurnal Sistem Komputer dan Kecerdasan Buatan*, vol. 1, no. 2, pp. 1–10, Mar. 2018.
[13] A. S. Utomo, "Algoritma Floodfill untuk Menentukan Titik Koordinat Maze Mapping Robot Linefollower," *Simetris : Jurnal Teknik Mesin, Elektro dan Ilmu Komputer*, vol. 7, no. 1, p. 227, Apr. 2016.
[14] A. Mulyadi, W. Wijono, and B. Siswojo, "Desain dan Simulasi Sistem Kontrol PID pada Pengendalian Kecapatan Motor Penggerak Quadrotor," *TRANSMISI*, vol. 22, no. 4, pp. 107–116, Nov. 2020.
[15] I. Iswanto, A. Ma'arif, O. Wahyunggoro, and A. Imam, "Artificial Potential Field Algorithm Implementation for Quadrotor Path Planning," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 8, 2019.
[16] R. Sakti Ruzianto, B. Setiyono, and Sumardi, "Perancangan Plant Pencampur Air Menggunakan Kontrol PID untuk Pengaturan Suhu Cairan Berbasis ATMEGA16," *TRANSMISI*, vol. 19, no. 2, pp. 65–71, Jul. 2017.
[17] F. Fahmizal, D. U. Rijalussalam, M. Budiyanto, and A. Mayub, "Trajectory Tracking pada Robot Omni

dengan Metode Odometry," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI)*, vol. 8, no. 1, p. 35, Mar. 2019.
[18] A. Hadi, P. J. Bathinalam, S. Alam, and R. Bengkalis, "Perbandingan Tuning Parameter Kontroller PD Menggunakan Metode Trial and Error dengan Analisa Gain pada Motor Servo AC," Apr. 2016.
[19] S. F. Anggraini, A. Ma'arif, and R. D. Puriyanto, "Pengendali PID pada Motor DC dan Tuning Menggunakan Metode Differential Evolution (DE)," *TELKA - Telekomunikasi Elektronika Komputasi dan Kontrol*, vol. 6, no. 2, pp. 147–159, Nov. 2020.
[20] R. Muhardian and K. Krismadinata, "Kendali Kecepatan Motor DC Dengan Kontroller PID dan Antarmuka Visual Basic," *JTEV (Jurnal Teknik Elektro dan Vokasional)*, vol. 6, no. 1, pp. 328–339, Feb. 2020.
[21] A. Ma'arif, H. Nabila, Iswanto, and O. Wahyunggoro, "Application of Intelligent Search Algorithms in Proportional-Integral-Derivative Control of Direct-Current Motor System," in *The 2019 Conference on Fundamental and Applied Science for Advanced Technology*, 2019, vol. 1373, no. 1, pp. 1–10.
[22] I. W. R. Ardana and I. P. Sutawinaya, "Pemodelan Sistem Kontroller Logika Fuzzy pada Pengaturan Kecepatan Motor Induksi Menggunakan Perangkat Lunak Matlab / Simulink," *Matrix : Jurnal Manajemen Teknologi dan Informatika*, vol. 7, no. 1, p. 1, Mar. 2017.
[23] M. R. Fajrianto and S. Wahyudi, "Perancangan Kontroller Fuzzy Model Reference Learning Control (FMRLC) Berbasis Mikrokontroler ATMega16 sebagai Kendali Motor Brushless DC (BLDC)," Universitas Diponegoro, Nov. 2017.
[24] A. Ma'arif and N. R. Setiawan, "Control of DC Motor Using Integral State Feedback and Comparison with PID: Simulation and Arduino Implementation," *Journal of Robotics and Control (JRC*, vol. 2, no. 5, pp. 456–461, Sep. 2021.
[25] F. Fahmizal, F. Fathuddin, and R. Susanto, "Identifikasi Sistem Motor DC dan Kendali Linear Quadratic Regulator Berbasis Arduino-Simulink Matlab," *Majalah Ilmiah Teknologi Elektro*, vol. 17, no. 2, pp. 299–306, Nov. 2018.
[26] M. R. DJALAL, K. HUTORO, and A. IMRAN, "Kontrol Kecepatan Motor Induksi menggunakan Algoritma Backpropagation Neural Network," *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, vol. 5, no. 2, p. 138, Feb. 2018.
[27] L. Widaningrum, B. Setiyono, and M. A. Riyadi, "Perancangan Kontroller Jaringan Syarat Tiruan B-Spline Berbasis Mikrokontroler ATMega16 sebagai Kendali Kecepatan Motor Brushless DC (BLDC)," *TRANSIENT*, vol. 6, no. 3, p. 373, Nov. 2017.
[28] G. Dewantoro and J. N. Sukamto, "Implementasi Kendali PID Menggunakan Jaringan Syaraf Tiruan Backpropagation," *ELKHA*, vol. 11, no. 1, p. 12, Apr. 2019.
[29] I. Qosim and M. Mujirudin, "Analisis Pengaturan Kecepatan Motor DC Menggunakan Kontrol PID (Proportional Integral Derivative)," *Prosiding Seminar Nasional Teknoka*, vol. 2, pp. E89–E94, Nov. 2017.
[30] A. Asri, M. R. Djalal, and D. Rahmat, "Desain Optimal Kontroller Proporsional Integral Motor DC Menggunakan Algoritme Particle Swarm Optimization," *Jetri : Jurnal Ilmiah Teknik Elektro*, vol. 15, no. 2, pp. 155–170, Feb. 2018.
[31] K. Untuk, K. Motor, M. Ali, I. Umami, and H. Sopian, "Particle Swarm Optimization (PSO) Sebagai Tuning PID," *Jurnal Intake : Jurnal Penelitian Ilmu Teknik dan Terapan*, vol. 7, no. 1, pp. 10–20, Apr. 2016.
[32] M. R. Djalal and Rahmat, "Penalaan Optimal Kendali Motor DC Berbasis Ant Colony Optimization," *Jurnal Teknologi*, vol. 12, no. 1, pp. 49–56, Feb. 2020.
[33] A. Kusmantoro and M. Margono, "Peningkatan Kinerja MPPT Menggunakan Kontrol PWM Fuzzy dengan Tuning PID," *Jurnal Rekayasa Elektrika*, vol. 16, no. 2, pp. 57–134, Aug. 2020.
[34] R. Hartayu, S. Santoso, A. O. U. Kaleka, and M. K. Musakhol, "Desain Simulasi Robot Kesetimbangan Dua Roda Dengan Kecerdasan Buatan," *Jurnal Sains dan Informatika*, vol. 6, no. 2, pp. 175–182, Dec. 2020.
[35] M. I. Febryansah, A. Yudhana, and A. Ma'arif, "Urinoir Analyzer Pintar Pendeteksi Kelainan Pada Fungsi Ginjal Dengan Analisis Kadar Ph Dan Warna Pada Urin," *Mobile and Forensics*, vol. 2, no. 1, pp. 36–44, May 2020.
[36] A. Ma'arif, I. Iswanto, A. A. Nuryono, and R. I. Alfian, "Kalman Filter for Noise Reducer on Sensor Readings," *Signal and Image Processing Letters*, vol. 1, no. 2, pp. 11–22, Jul. 2019.
[37] R. I. Alfian, A. Ma'arif, and S. Sunardi, "Noise Reduction in the Accelerometer and Gyroscope Sensor with the Kalman Filter Algorithm," *Journal of Robotics and Control (JRC)*, vol. 2, no. 3, pp. 180–189, 2021.
[38] I. C. Utama, "PID Control For Ping-Pong Balance With Ultrasonic Sensor And Servo Motor Based On Labview," *Telekontran : Jurnal Ilmiah Telekomunikasi, Kendali dan Elektronika Terapan*, vol. 4, no. 2,

pp. 68–77, Jul. 2016.
[39] F. M. W. Fatih Mutamimul Wildan, "Sistem Pengaturan Kecepatan Motor Induksi Tiga Fasa Menggunakan Kontroler PID Berbasis Genetic Algorithm," *KINETIK*, vol. 1, no. 1, pp. 2503–2259, Oct. 2016.
[40] M. Irhas, I. Iftitah, and S. A. Azizah Ilham, "Penggunaan Kontrol PID dengan Berbagai Metode untuk Analisis Pengaturan Kecepatan Motor DC," *JFT : Jurnal Fisika dan Terapannya*, vol. 7, no. 1, p. 78, Jun. 2020.
[41] J. A. Prakosa, E. Kurniawan, H. Adinanta, S. Suryadi, and M. I. Afandi, "Kajian Eksperimen Teknik Kontrol Penerbangan Posisi Tinggal Landas Drone Bikopter dengan Metode PID," *Jurnal Otomasi Kontrol dan Instrumentasi*, vol. 12, no. 2, pp. 1–8, Oct. 2020.
[42] R. Sirait and C. Botiwicaksono, "Sistem Kontrol Kelembaban Tanah Pada Tanaman Tomat Menggunakan PID," *Techno.Com*, vol. 19, no. 3, pp. 262–273, Aug. 2020.
[43] A. Ma'arif, Iswanto, N. M. Raharja, P. Aditya Rosyady, A. R. Cahya Baswara, and A. Anggari Nuryono, "Control of DC Motor Using Proportional Integral Derivative (PID): Arduino Hardware Implementation," in *2020 2nd International Conference on Industrial Electrical and Electronics (ICIEE)*, 2020, pp. 74–78.
[44] E. Apriaskar, N. Azis Salim, and D. Prastiyanto, "Performance Evaluation of Balancing Bicopter using P, PI, and PID Controller," *Jurnal Teknik Elektro*, vol. 11, no. 2, pp. 44–49, Dec. 2019.
[45] D. Irawan and P. Perdana SS, "Kontrol Motor Brushless DC (BLDC) Berbasis Algoritma AI - PID," *Jurnal Teknik Elektro dan Komputasi (ELKOM)*, vol. 2, no. 1, pp. 41–48, Mar. 2020.
[46] K. Ogata, *Modern Control Engineering*. Boston: Prentice Hall, 2010.
[47] A. Najmurrokhman, K. Kusnandar, I. Irfansyah, and A. Daelami, "Rancang Bangun Auto Balancing Robot Menggunakan Metode Kendali PID," *TELKA - Telekomunikasi, Elektronika, Komputasi dan Kontrol*, vol. 5, no. 1, pp. 15–23, May 2019.
[48] M. R. Djalal and H. HR, "Speed Control Series Dc Motor Using Ant Colony Optimization," *Techno (Jurnal Fakultas Teknik, Universitas Muhammadiyah Purwokerto)*, vol. 20, no. 2, p. 105, Nov. 2019.
[49] F. Fauzy and S. Kasmungin, "Perancangan Sistem Kontrol Kecepatan Motor DC Dengan PID Labview 2010," *Manutech : Jurnal Teknologi Manufaktur*, vol. 9, no. 02, pp. 23–32, May 2019.
[50] M. F. Al Andzar and R. D. Puriyanto, "PID Control for Temperature and Motor Speed Based on PLC," *Signal and Image Processing Letters*, vol. 1, no. 1, pp. 7–13, Mar. 2019.

AUTHOR BIOGRAPHY

Alfian Ma'arif obtained a Bachelor of Engineering (ST) degree from the Islamic University of Indonesia's Electrical Engineering department in 2014. The Master of Engineering (M. Eng.) degree was obtained from the Electrical Engineering Department of Gadjah Mada University in 2017. The author is a teaching staff at the University's Electrical Engineering Study Program. Ahmad Dahlan since 2018. His research includes control systems and computer programming.

Naufal Rahmat Setiawan is a student of the Ahmad Dahlan University Electrical Engineering study program

Eka Suci Rahayu is a student of the Ahmad Dahlan University Electrical Engineering study program who has obtained a Bachelor of Engineering (ST) degree in the study program in 2020.