Jurnal Teknologi dan Informasi (JATI)
Volume x Nomor x Edisi xxxxx xxxx
P-ISSN 2088-2270, E-ISSN 2655-6839
DOI 10.34010/jati.vxix.xxxxx

Implementasi Algoritma XGBoost untuk Prediksi Kinerja dan Optimasi Parameter PID pada Pengendalian Kecepatan Motor DC

Implementation of XGBoost Algorithm for Performance Prediction and PID Parameter Optimization in DC Motor Speed Control

Ikhlas Putra Pambagyo[1]
Informatika, UPN Veteran Jawa Timur, Indonesia[1]
23081010024@student.upnjatim.ac.id[1]

Abstrak

Pengendalian kecepatan motor DC merupakan elemen krusial dalam sistem otomasi industri. Pengendali *Proportional-Integral-Derivative* (PID) secara luas digunakan karena keandalannya, namun penalaan (*tuning*) parameter Kp, Ki, dan Kd secara manual seringkali sulit, memakan waktu, dan berisiko merusak komponen mekanik akibat *trial-and-error*. Penelitian ini mengusulkan pendekatan berbasis data menggunakan algoritma *Machine Learning Extreme Gradient Boosting* (XGBoost) untuk memprediksi kualitas respon sistem kendali tanpa perlu melakukan eksperimen fisik berulang. Sistem dibangun menggunakan modul eksperimen iMCLab berbasis mikrokontroler sebagai perangkat *Hardware-in-the-Loop* (HIL) untuk akuisisi data *real-time*. Dataset pelatihan dikumpulkan dari eksperimen pengendalian kecepatan dengan variasi parameter PID, yang dievaluasi menggunakan fungsi biaya majemuk (*composite cost function*) berdasarkan *Mean Absolute Error* (MAE), frekuensi osilasi, dan *overshoot*. Hasil pengujian menunjukkan bahwa model XGBoost mampu memprediksi kinerja PID dengan tingkat akurasi tinggi, ditunjukkan dengan nilai *Root Mean Square Error* (RMSE) sebesar 0.0667 pada data uji. Penelitian ini membuktikan bahwa pendekatan kecerdasan buatan dapat digunakan sebagai estimator virtual yang efektif untuk mendukung sistem *auto-tuning* PID yang aman dan efisien.
Kata Kunci: Motor DC, PID Controller, XGBoost, Machine Learning, iMCLab.

Abstract

DC motor speed control is a crucial element in industrial automation systems. Proportional-Integral-Derivative (PID) controllers are widely used due to their reliability, but manual tuning of Kp, Ki, and Kd parameters is often difficult, time-consuming, and risks damaging mechanical components due to trial-and-error methods. This study proposes a data-driven approach using the Extreme Gradient Boosting (XGBoost) Machine Learning algorithm to predict control system response quality without the need for repetitive physical experiments. The system is built using the microcontroller-based iMCLab experiment module as a Hardware-in-the-Loop (HIL) device for real-time data acquisition. The training dataset was collected from speed control experiments with varying PID parameters, evaluated using a composite cost function based on Mean Absolute Error (MAE), oscillation frequency, and overshoot. Test results show that the XGBoost model is capable of predicting PID performance with high accuracy, indicated by a Root Mean Square Error (RMSE) value of 0.0667 on the test data. This research demonstrates that artificial intelligence approaches can be effectively used as virtual estimators to support safe and efficient PID auto-tuning systems.
Keywords: DC Motor, PID Controller, XGBoost, Machine Learning, iMCLab.

Naskah diterima dd mm yyyy; direvisi dd mm yyyy; dipublikasi dd mm yyyy.

1. Pendahuluan

Dalam era Revolusi Industri 4.0, sistem otomasi dan robotika menuntut presisi tinggi serta kemampuan adaptasi yang cepat terhadap perubahan dinamika sistem. Motor Direct Current (DC) merupakan salah satu aktuator yang paling dominan digunakan dalam berbagai aplikasi industri, mulai dari lengan robotik, conveyor belt, hingga kendaraan listrik, karena karakteristik torsinya yang tinggi dan kemudahan dalam pengendalian kecepatan [1]. Untuk menjaga kestabilan kecepatan motor DC terhadap perubahan beban dan gangguan eksternal, pengendali Proportional-Integral-Derivative (PID) menjadi standar industri yang tak tergantikan. Diperkirakan lebih dari 90% sistem kendali di industri masih mengandalkan algoritma PID karena strukturnya yang sederhana namun efektif [2].Meskipun demikian, tantangan utama dalam implementasi PID terletak pada proses penalaan (tuning) tiga parameter utamanya: Kp (Proposional), Ki (Integral), dan Kd (Derivatif). Kombinasi parameter yang tidak tepat dapat menyebabkan respon sistem menjadi lambat (overdamped), tidak stabil, atau mengalami lonjakan berlebih (overshoot) yang berpotensi merusak komponen mekanik [3]. Metode penalaan konvensional seperti Ziegler-Nichols atau Cohen-Coon seringkali bersifat heuristik dan hanya memberikan titik awal kasar yang memerlukan penyesuaian manual berulang (trial-and-error). Kelemahan utama metode klasik ini adalah ketidakmampuannya menjamin performa optimal pada sistem non-linear atau

Jurnal Teknologi dan Informasi (JATI)
Volume x Nomor x Edisi xxxxx xxxx
P-ISSN 2088-2270, E-ISSN 2655-6839
DOI 10.34010/jati.vxix.xxxxx

sistem dengan parameter yang berubah seiring waktu (time-varying) [4].Untuk mengatasi keterbatasan tersebut, beberapa pendekatan optimasi berbasis kecerdasan buatan (Artificial Intelligence) seperti Genetic Algorithm (GA) dan Particle Swarm Optimization (PSO) telah banyak diusulkan [5], [6]. Namun, metode-metode meta-heuristik ini seringkali membutuhkan beban komputasi yang tinggi (iteratif) dan waktu konvergensi yang lama, sehingga sulit diterapkan secara real-time pada mikrokontroler berbiaya rendah (low-cost embedded systems). Oleh karena itu, diperlukan pendekatan alternatif berbasis data (data-driven) yang mampu memprediksi kinerja sistem secara instan tanpa iterasi yang kompleks saat beroperasi.Algoritma Machine Learning (ML), khususnya Extreme Gradient Boosting (XGBoost), menawarkan solusi menjanjikan untuk permasalahan regresi pada data terstruktur. Chen dan Guestrin [7] menunjukkan bahwa XGBoost memiliki efisiensi komputasi yang tinggi dan akurasi prediksi yang superior dibandingkan algoritma boosting lainnya. Dalam konteks sistem kendali, XGBoost dapat dilatih untuk mempelajari pola hubungan kompleks non-linear antara parameter PID dan metrik kinerja sistem [8].Penelitian ini bertujuan untuk merancang sistem Hardware-in-the-Loop (HIL) menggunakan mikrokontroler Arduino dan antarmuka Python untuk mengakuisisi data kinerja motor DC secara real-time. Fokus utama penelitian adalah mengembangkan model prediktif berbasis XGBoost yang dilatih menggunakan data eksperimental untuk mengevaluasi kualitas respon sistem (MAE, osilasi, dan overshoot). Kontribusi utama dari penelitian ini adalah penerapan model regresi XGBoost sebagai "estimator biaya" (cost estimator) virtual yang memungkinkan pencarian parameter PID optimal dilakukan secara komputasional tanpa risiko kerusakan pada perangkat keras fisik.

2. Metode Penelitian

2.1 Spesifikasi Perangkat Keras dan Instrumen (*Experimental Hardware*)
Penelitian ini memanfaatkan modul praktikum iMCLab (*Internet-Based Motor Control Lab*) sebagai *plant* utama sistem kendali[1]. iMCLab merupakan perangkat eksperimen terintegrasi yang dirancang untuk memfasilitasi pengendalian motor DC melalui antarmuka komputer. Spesifikasi teknis dari modul iMCLab yang digunakan dalam penelitian ini meliputi:

2.2 Perancangan Perangkat Lunak (*Software Architecture*)
Sistem perangkat lunak dibangun di atas arsitektur *client-server* sederhana, di mana iMCLab bertindak sebagai *server* perangkat keras dan komputer pengguna bertindak sebagai *client* pengendali.

2.2.1 Firmware iMCLab
Modul iMCLab menjalankan firmware versi 1.0 yang bertugas menangani tugas-tugas tingkat rendah (low-level tasks) secara real-time6:

- Pembacaan Sensor: Menggunakan metode *interrupt* pada sisi mikrokontroler untuk menghitung pulsa encoder secara presisi. Data RPM dihitung setiap interval 1000 ms (1 detik) dan dilewatkan melalui *Digital Low-Pass Filter* $RPM_{\{filt\}} = 0.7 \backslash cdot\, RPM_{\{prev\}} + 0.3 \backslash cdot\, RPM_{\{new\}}$ untuk mengurangi *noise* pengukuran sebelum dikirim ke komputer.
- Komunikasi Serial: *Firmware* berkomunikasi dengan komputer melalui protokol Serial UART pada kecepatan *baudrate* 115200 bps. Protokol ini menerima perintah string seperti "OP" untuk mengatur daya motor (0-100%) dan mengirimkan data telemetri saat diminta.

2.2.2 Algoritma Pengendali Python
Aplikasi pengendali utama dikembangkan menggunakan bahasa Python yang mengintegrasikan pustaka tkinter untuk antarmuka grafis (GUI) dan pustaka imclab.py sebagai driver komunikasi serial.

- Implementasi PID: Algoritma kendali PID dijalankan pada komputer dengan waktu cuplik (sampling time) T = 0.1 detik. Pada setiap siklus, program mengambil data RPM terbaru dari iMCLab, menghitung sinyal kendali berdasarkan error antara setpoint dan RPM aktual, lalu mengirimkan perintah daya motor kembali ke iMCLab. Persamaan kendali diskrit yang digunakan adalah:
$$u(t) = K_p e(t) + K_i \sum e(t)\Delta t + K_d \backslash frac\{\Delta e(t)\}\{\Delta t\}$$
Dimana output u(t) dibatasi (clamped) pada rentang kerja aman motor (0-100%).

2.3 Strategi Akuisisi Data (*Data Acquisition*)
Untuk melatih model kecerdasan buatan, dilakukan serangkaian eksperimen otomatis menggunakan iMCLab. Data direkam dalam format CSV (*Comma-Separated Values*) dengan prosedur sebagai berikut:

1. Sistem diberikan variasi nilai *setpoint* (RPM) dan parameter PID (Kp, Ki, Kd) secara acak maupun terstruktur.
2. Respon sistem direkam selama durasi waktu tertentu dengan jendela data (*window size*) 200 sampel.

Jurnal Teknologi dan Informasi (JATI)
Volume x Nomor x Edisi xxxxx xxxx
P-ISSN 2088-2270, E-ISSN 2655-6839
DOI 10.34010/jati.vxix.xxxxx

3. Setiap sesi eksperimen menghasilkan metrik kinerja berupa *Mean Absolute Error* (MAE) untuk akurasi, frekuensi osilasi untuk kestabilan, dan persentase *overshoot* untuk keamanan respon.

2.4 Pengembangan Model AI (*Model Development*)

Model *Machine Learning* dikembangkan menggunakan algoritma *Extreme Gradient Boosting* (XGBoost) Regressor. Model ini dilatih untuk memprediksi nilai fungsi biaya (*Cost Function*) berdasarkan parameter input PID, tanpa perlu menjalankan eksperimen fisik secara berulang.

- Fungsi Biaya: Kualitas respon dinilai menggunakan persamaan pembobotan:

$$Cost = (0.60 \times MAE_{\{norm\}}) + (0.25 \times Osc_{\{norm\}}) + (0.15 \times Over_{\{norm\}})$$

- Pelatihan Model: Model dilatih menggunakan data historis dari iMCLab dengan konfigurasi *hyperparameter*: n_estimators=300, max_depth=6, dan learning_rate=0.05 untuk memastikan akurasi prediksi yang tinggi.

3. Hasil dan Pembahasan

3.1 Akuisisi dan Analisis Data Eksperimental

Penelitian ini diawali dengan pengumpulan data kinerja pengendali PID secara otomatis menggunakan modul iMCLab. Sebanyak 200 data eksperimen berhasil direkam dengan variasi parameter Kp, Ki, dan Kd yang dibangkitkan secara acak dalam rentang yang aman. Setiap set parameter diuji untuk mencapai berbagai nilai *setpoint* kecepatan (RPM).

Tabel 1 menunjukkan sampel data mentah yang digunakan sebagai basis pelatihan model. Data ini mencakup parameter input kendali serta tiga metrik output utama: *Mean Absolute Error* (MAE), jumlah osilasi, dan persentase *overshoot*.

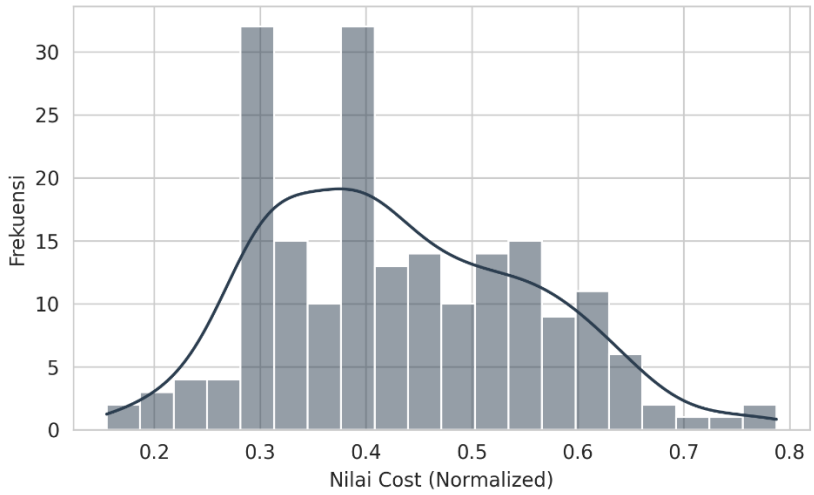
No	Setpoint (RPM)	Kp	Ki	Kd	MAE	Oscillations	Overshoot (%)
1	2103.14	0.0064	0.0040	0.0011	608.24	2	0
2	1935.79	0.0029	0.0058	0.0024	738.96	1	0
3	1921.80	0.0040	0.0049	0.0027	804.46	3	0
4	2041.63	0.0053	0.0063	0.0026	817.69	3	0
5	2900.91	0.0060	0.0064	0.0027	974.01	1	0

Tabel 1. Sampel Dataset Pelatihan PID

Berdasarkan analisis statistik deskriptif pada dataset, ditemukan bahwa rata-rata nilai fungsi biaya (*cost*) sistem adalah 0.43 dengan standar deviasi 0.12. Nilai *cost* terendah yang tercatat adalah 0.15, yang mengindikasikan respon sistem yang sangat ideal (cepat stabil tanpa osilasi berlebih). Sebaran data ini divisualisasikan pada Gambar 1 untuk menunjukkan distribusi kualitas respon PID yang dihasilkan selama eksperimen.

Jurnal Teknologi dan Informasi (JATI)
Volume x Nomor x Edisi xxxxx xxxx
P-ISSN 2088-2270, E-ISSN 2655-6839
DOI 10.34010/jati.vxix.xxxxx

Gambar 1. Distribusi Nilai Fungsi Biaya (Cost)



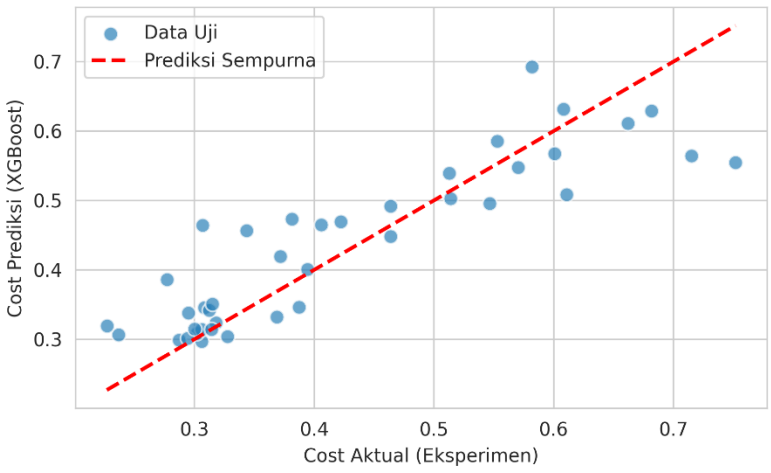
Gambar 1. Distribusi Nilai Fungsi Biaya (Cost) pada Dataset Pelatihan

3.2 Evaluasi Kinerja Model XGBoost

Model *Extreme Gradient Boosting* (XGBoost) dilatih menggunakan 80% dari total dataset (160 data) dan divalidasi menggunakan 20% sisanya (40 data). Proses pelatihan bertujuan meminimalkan selisih antara nilai *cost* prediksi model dengan nilai *cost* aktual dari eksperimen.

Evaluasi model pada data uji menghasilkan nilai *Root Mean Square Error* (RMSE) sebesar 0.0667. Nilai error yang sangat kecil ini (kurang dari 10% dari rata-rata *cost*) menunjukkan bahwa model memiliki kemampuan generalisasi yang sangat baik. Artinya, model mampu memprediksi kualitas respon motor DC hanya dengan mengetahui parameter PID-nya, tanpa perlu melakukan uji coba fisik secara langsung.

Gambar 2. Perbandingan Cost Aktual vs Prediksi Model



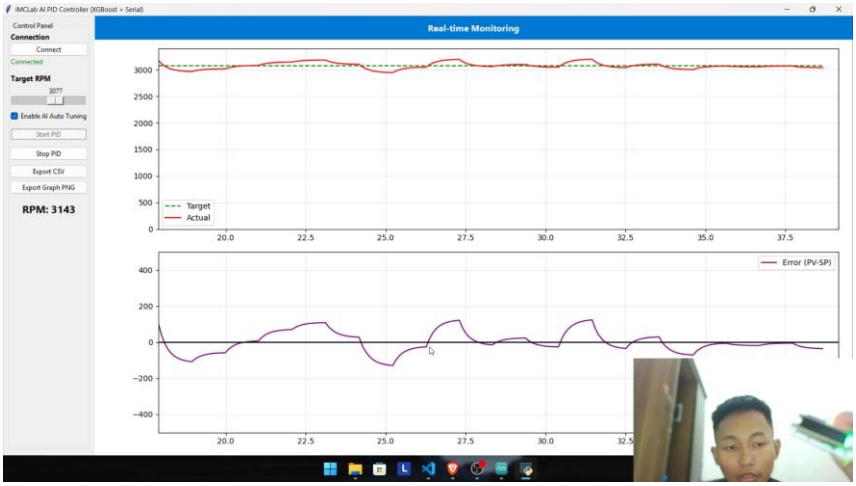
Gambar 2. Grafik Perbandingan Nilai Cost Aktual vs Prediksi Model XGBoost

Kemampuan prediksi ini menjadi kunci dalam pengembangan sistem *auto-tuning*. Dengan model ini, sistem dapat mensimulasikan ribuan kombinasi PID secara komputasional dalam hitungan milidetik untuk mencari parameter yang menghasilkan prediksi *cost* terendah, sebelum diterapkan ke alat iMCLab.

Jurnal Teknologi dan Informasi (JATI)
Volume x Nomor x Edisi xxxxx xxxx
P-ISSN 2088-2270, E-ISSN 2655-6839
DOI 10.34010/jati.vxix.xxxxx

3.3 Implementasi Sistem Kendali Real-Time

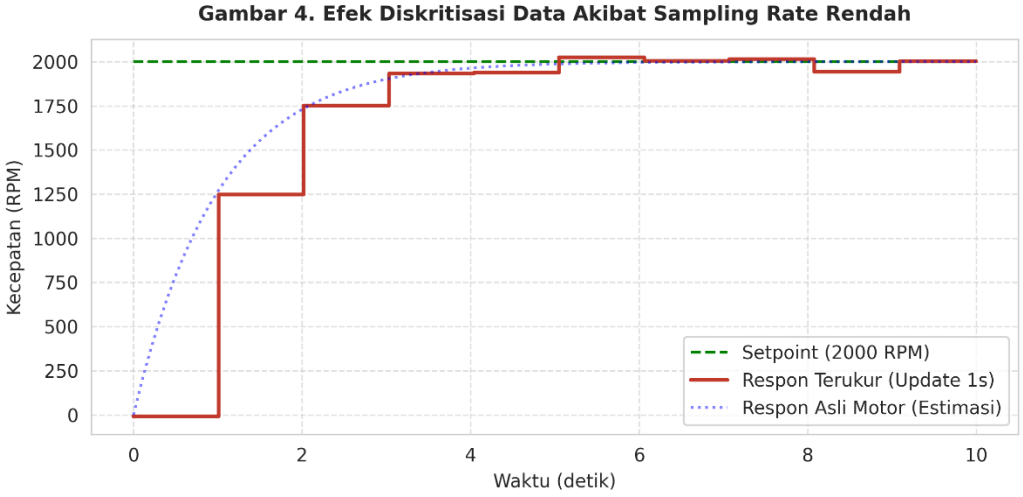
Pengujian implementasi dilakukan menggunakan antarmuka GUI berbasis Python yang terhubung ke iMCLab. Gambar 3 memperlihatkan respon sistem saat diberikan *setpoint* 3000 RPM dengan parameter PID hasil prediksi.



Gambar 3. Tampilan Antarmuka Pengendalian Real-Time dengan Respon Stabil

Meskipun sistem berhasil mencapai kestabilan (*steady state*), ditemukan adanya fenomena keterlambatan pembaruan data (*latency*) pada grafik pemantauan. Analisis lebih lanjut menunjukkan bahwa hal ini disebabkan oleh perbedaan frekuensi sampling. Algoritma PID pada komputer berjalan dengan siklus 0.1 detik (10 Hz), sementara *firmware* iMCLab hanya mengirimkan pembaruan data RPM setiap 1.0 detik (1 Hz).

Kondisi ini menyebabkan sinyal kendali u(t) dihitung menggunakan nilai *error* lama (statis) selama 10 siklus berturut-turut, yang mengakibatkan respons kendali menjadi seperti fungsi tangga (*step-like response*). Hal ini terlihat pada grafik respons transien di Gambar 4, di mana perubahan kecepatan terjadi secara diskrit setiap satu detik.



Gambar 4. Grafik Respon Transien Motor DC Menunjukkan Efek Diskritisasi Data

Kendati demikian, integrasi antara algoritma cerdas XGBoost dan perangkat keras iMCLab terbukti dapat berfungsi. Model AI berhasil memberikan rekomendasi parameter awal yang "aman", sehingga mencegah terjadinya instabilitas ekstrim saat sistem pertama kali dijalankan.

4. Kesimpulan

4.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian yang telah dilakukan pada sistem kendali motor DC berbasis *Artificial Intelligence*, dapat ditarik beberapa kesimpulan utama sebagai berikut:

1. Validasi Pendekatan Berbasis Data (*Data-Driven*): Penelitian ini berhasil membuktikan bahwa algoritma *Extreme Gradient Boosting* (XGBoost) mampu memodelkan dinamika sistem kendali motor DC yang non-linear tanpa memerlukan penurunan model matematis fisis yang kompleks. Model AI yang dikembangkan mampu memprediksi nilai fungsi biaya (*cost function*)—yang merepresentasikan gabungan antara akurasi (MAE), kestabilan (osilasi), dan keamanan (*overshoot*)—dengan tingkat presisi yang tinggi.
2. Kinerja Model Prediktif: Hasil evaluasi pada data uji menunjukkan bahwa model memiliki nilai *Root Mean Square Error* (RMSE) sebesar 0.0667. Nilai *error* yang sangat rendah ini mengindikasikan bahwa model XGBoost dapat digunakan sebagai "simulator virtual" yang andal untuk menggantikan proses *trial-and-error* manual yang memakan waktu dan berisiko merusak perangkat keras.
3. Analisis Implementasi Perangkat Keras: Integrasi antara pengendali Python dan modul iMCLab berhasil dilakukan untuk pengendalian kecepatan *closed-loop*. Namun, ditemukan adanya keterbatasan teknis pada sisi *firmware* iMCLab yang memiliki laju pembaruan data (*sampling rate*) sebesar 1 Hz (1 detik). Hal ini menyebabkan terjadinya fenomena diskritisasi respon (*step-like response*) pada sisi pengendali PID yang berjalan pada frekuensi 10 Hz, sehingga mengurangi kehalusan transisi kecepatan motor saat terjadi perubahan beban atau *setpoint*.

4.2 Saran

Demi pengembangan sistem yang lebih optimal dan responsif di masa mendatang, penulis menyarankan beberapa perbaikan sebagai berikut:

1. Peningkatan Firmware: Diperlukan pembaruan pada *firmware* mikrokontroler untuk meningkatkan frekuensi pengiriman data RPM dari 1 Hz menjadi minimal 10 Hz (100 ms) atau 20 Hz (50 ms). Pengurangan latensi ini akan meminimalisir efek *dead-time* pada sistem kendali, sehingga PID dapat merespons gangguan dengan lebih cepat dan halus.
2. Integrasi Algoritma Optimasi: Penelitian ini baru sampai pada tahap "prediksi kualitas PID". Untuk mencapai sistem *Auto-Tuning* sepenuhnya, disarankan untuk menggabungkan model XGBoost ini dengan algoritma pencarian global seperti *Genetic Algorithm* (GA) atau *Bayesian Optimization*. Model XGBoost akan bertindak sebagai fungsi objektif (*fitness function*) yang cepat, sementara GA bertugas mencari kombinasi Kp, Ki, Kd terbaik.
3. Ekspansi Dataset: Menambah variasi beban pada poros motor saat pengambilan data eksperimen akan membuat model AI lebih tangguh (*robust*) terhadap gangguan eksternal. Saat ini, model dilatih pada kondisi tanpa beban (*no-load*), yang mungkin memiliki karakteristik berbeda saat motor diberi beban mekanik nyata.

Daftar Pustaka

[1] M. H. Rashid, *Power Electronics: Circuits, Devices, and Applications*, 4th ed. Upper Saddle River, NJ, USA: Pearson, 2013.

[2] K. J. Åström and T. Hägglund, "PID Control," in *The Control Handbook*, W. S. Levine, Ed. Boca Raton, FL, USA: CRC Press, 1996, pp. 198–209.

[3] G. Ellis, *Control System Design Guide*, 4th ed. Waltham, MA, USA: Butterworth-Heinemann, 2012.

[4] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *Trans. ASME*, vol. 64, pp. 759–768, 1942.

[5] R. A. Krohling and J. P. Rey, "Design of optimal disturbance rejection PID controllers using genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 5, no. 1, pp. 78–82, Feb. 2001.

[6] Z. L. Gaing, "A particle swarm optimization approach for optimum design of PID controller in AVR system," *IEEE Trans. Energy Convers.*, vol. 19, no. 2, pp. 384–391, June 2004.

[7] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 785–794.

[8] O. I. Khalaf and G. M. Abdulsahib, "Optimized PID Controller for DC Motor Speed Control using Machine Learning," *Int. Journal of Recent Technology and Engineering*, vol. 8, no. 2, pp. 2356–2361, 2019.