ISSN (Online) 2747-0563
Volume 4 Tahun 2024

Implementasi Kontrol Kecepatan Motor DC Berbasis ESP32 dengan Kombinasi LSTM Feedforward dan PID

Ade Rizky Panjaitan[1], Penulis Kedua[2]

[1,2] Afiliasi1 (Informatika, Universitas Pembangunan Nasional "Veteran" Jawa Timur)
[1]22081010091@student.upnjatim.ac.id
[2]22081010000@student.upnjatim.ac.id

*Corresponding author email: 22081010091@student.upnjatim.ac.id

***Abstrak*— Motor listrik berperan penting dalam berbagai sistem otomasi sehingga dibutuhkan kendali kecepatan yang akurat pada platform tertanam dengan sumber daya terbatas. Pengendali proporsional integral derivatif (PID) masih banyak digunakan, tetapi memiliki keterbatasan ketika sistem bersifat nonlinier dan mengalami perubahan beban. Penelitian ini mengusulkan rancangan pengendali hibrida yang mengombinasikan model jaringan saraf Long Short Term Memory (LSTM) sebagai feedforward dengan pengendali PID konvensional pada mikrokontroler ESP32. Model LSTM dilatih secara offline menggunakan data telemetri yang berisi setpoint kecepatan, kecepatan aktual motor, dan sinyal Pulse Width Modulation (PWM) yang dikumpulkan dari percobaan sebelumnya. Data diubah menjadi deret waktu berdimensi dua dengan panjang jendela tertentu, kemudian digunakan untuk melatih model LSTM berukuran kecil yang dipetakan ke keluaran PWM terukur dalam bentuk ternormalisasi. Model terlatih dikonversi ke format TensorFlow Lite dan diintegrasikan ke dalam program ESP32 menggunakan TensorFlow Lite Micro sehingga inferensi dapat dijalankan secara langsung pada mikrokontroler. Pada sisi perangkat keras, ESP32 mengukur kecepatan motor dengan sensor encoder dan menghitung sinyal PWM total sebagai penjumlahan antara keluaran feedforward LSTM dan koreksi PID. Hasil pengujian menunjukkan bahwa pendekatan hibrida ini mampu mempertahankan kecepatan motor lebih dekat terhadap setpoint dibandingkan pengendali PID murni, terutama saat terjadi perubahan setpoint dan gangguan beban.**

Kata Kunci— **ESP32, mikrokontroler, motor DC, PID, LSTM, TensorFlow Lite Micro**

I. PENDAHULUAN

Motor DC dan mikrokontroler banyak digunakan dalam sistem otomasi, robotika, serta peralatan laboratorium karena kemudahan pengaturan kecepatan dan torsi. Kinerja sistem sangat bergantung pada kemampuan pengendali kecepatan dalam mengikuti setpoint dan menolak gangguan, misalnya perubahan beban mekanik atau variasi tegangan suplai. Pada praktiknya, pengendali PID masih menjadi pilihan utama karena struktur yang sederhana dan mudah diimplementasikan pada mikrokontroler[2], [11].

Meskipun demikian, pengendali PID murni sering mengalami penurunan kinerja ketika karakteristik beban berubah atau ketika sistem memiliki nonlinieritas yang signifikan. Proses penalaan parameter juga memengaruhi performa, dan salah satu pendekatan klasik yang banyak digunakan adalah metode Ziegler Nichols [1], [2].

Sebagian besar penerapan jaringan saraf untuk kendali motor dilakukan pada komputer atau sistem tertanam berkapasitas besar. Tantangan yang muncul adalah bagaimana menjalankan model jaringan saraf pada mikrokontroler dengan memori dan kemampuan komputasi terbatas, sekaligus mempertahankan periode sampling yang cukup cepat untuk kendali kecepatan. TensorFlow Lite Micro menyediakan kerangka kerja inferensi yang dirancang untuk perangkat semacam ESP32 sehingga memungkinkan integrasi model LSTM berukuran kecil ke dalam perangkat kendali kecepatan motor.

Penelitian ini bertujuan merancang dan mengimplementasikan pengendali kecepatan motor DC berbasis ESP32 yang mengombinasikan feedforward LSTM dengan PID. Model LSTM dilatih untuk memetakan sejarah setpoint dan kecepatan aktual menjadi sinyal PWM feedforward, sedangkan PID digunakan sebagai pengendali umpan balik untuk mengoreksi error residu. Model terlatih kemudian dikonversi ke format TensorFlow Lite dan diintegrasikan dalam kode ESP32 menggunakan TensorFlow Lite Micro. Dengan demikian, kontribusi utama penelitian ini adalah: 1) merancang alur pelatihan dan konversi model LSTM yang sesuai untuk dijalankan pada mikrokontroler, 2) mengimplementasikan arsitektur pengendali hibrida LSTM feedforward dan PID pada ESP32, serta 3) mengevaluasi kinerja pengendali terhadap variasi setpoint dan gangguan beban.

II. TINJAUAN PUSTAKA

Pengendali PID telah digunakan secara luas untuk pengaturan kecepatan motor karena bentuk persamaan yang sederhana dan dapat diturunkan dari model linier sistem. Berbagai metode penalaan PID, termasuk Ziegler Nichols dan pendekatan berbasis optimasi, telah dikembangkan untuk memperbaiki performa pengendali [1], [2]. Namun, metode tersebut umumnya mengasumsikan model linier atau kondisi operasi yang relatif tetap sehingga masih menghadapi kesulitan ketika sistem bersifat nonlinier dan parameter berubah seiring waktu.

Perkembangan pembelajaran mesin memberikan alternatif pendekatan yang memanfaatkan data untuk memodelkan dinamika sistem secara langsung. Jaringan saraf multilayer dan jaringan saraf rekuren telah digunakan untuk membangun model plant atau langsung menghasilkan sinyal kendali.

ISSN (Online) 2747-0563
Volume 4 Tahun 2024

LSTM merupakan varian jaringan saraf rekuren yang dikembangkan untuk mengatasi masalah gradien menghilang dan efektif pada data deret waktu dengan ketergantungan jangka panjang [3]. Dalam konteks kendali motor, LSTM dapat digunakan untuk mempelajari hubungan antara urutan setpoint dan respon kecepatan terhadap sinyal PWM tanpa eksplisit menurunkan model matematis plant.

Pada sisi implementasi tertanam, TensorFlow Lite Micro memungkinkan model jaringan saraf kecil dijalankan pada mikrokontroler dengan memori terbatas. Berbagai studi telah menunjukkan bahwa Implementasi jaringan saraf pada mikrokontroler memerlukan perhatian terhadap ukuran model, kebutuhan memori tensor, dan waktu eksekusi per siklus, sehingga pemilihan arsitektur dan konfigurasi inferensi perlu disesuaikan dengan keterbatasan perangkat. TensorFlow Lite Micro mengadopsi desain interpreter yang ditujukan untuk lingkungan tanpa alokasi memori dinamis dan dapat dijalankan pada perangkat dengan memori kecil [4], [7]. Oleh karena itu, diperlukan kompromi antara kompleksitas model, periode sampling kendali, dan kapasitas perangkat keras.

Berdasarkan kajian tersebut, pendekatan hibrida yang mengombinasikan feedforward berbasis model data driven dengan pengendali PID konvensional menjadi menarik. Feedforward berbasis jaringan saraf diharapkan mampu menangkap nonlinieritas plant dan memprediksi sinyal kendali yang mendekati nilai optimum, sedangkan PID menjaga kestabilan dan robustnes sistem dengan mengoreksi error residu. Penelitian ini mengadopsi paradigma tersebut dan memfokuskan implementasinya pada platform ESP32 dengan model LSTM yang dioptimalkan untuk dijalankan melalui TensorFlow Lite Micro.

III. METODE PENELITIAN

A. *Arsitektur Sistem Kendali*

Sistem yang dikembangkan terdiri atas sebuah mikrokontroler ESP32, rangkaian driver motor H bridge, motor DC dengan encoder optik, serta catu daya yang sesuai. Sistem terdiri atas mikrokontroler ESP32, driver motor, motor DC, dan sensor encoder. ESP32 menghasilkan sinyal PWM melalui periferal LEDC, sedangkan pulsa encoder dapat dibaca menggunakan periferal penghitung pulsa untuk memperoleh estimasi kecepatan per menit [8]-[10]. Periode perhitungan kecepatan diatur menggunakan timer perangkat lunak sehingga nilai rpm diperbarui secara berkala.

Gbr. 1 Rangkaian pengujian pengendali kecepatan motor DC berbasis ESP32

Konfigurasi perangkat keras yang digunakan pada penelitian ini ditunjukkan pada Gambar 1. Pada level perangkat lunak, program ESP32 menghitung sinyal kendali total sebagai penjumlahan antara sinyal feedforward dari model LSTM dan sinyal koreksi dari pengendali PID. Nilai kecepatan referensi atau setpoint dapat diberikan melalui antarmuka serial. Program juga menyediakan perintah untuk mengaktifkan atau menonaktifkan modul kecerdasan buatan serta mode manual dengan memberi nilai PWM secara langsung, yang berguna untuk proses debugging dan pengujian awal.

B. *Pelatihan Model LSTM Feedforward*

Model LSTM dilatih secara offline menggunakan berkas data telemetri yang berisi kolom setpoint_rpm, rpm, dan pwm. Data dibaca menggunakan pustaka pandas dan kemudian dikonversi menjadi array numerik bertipe floating point. Sebelum pelatihan, data dinormalisasi ke rentang tertentu, misalnya 0 sampai 1, dengan cara membagi nilai setpoint dan rpm dengan nilai maksimum rpm, sedangkan nilai pwm dibagi dengan nilai maksimum PWM.

Data deret waktu kemudian diubah menjadi himpunan urutan dengan panjang jendela SEQ_LEN. Setiap sampel masukan berbentuk matriks dengan ukuran SEQ_LEN kali dua yang merepresentasikan sejarah setpoint dan rpm pada beberapa langkah waktu sebelumnya. Target keluaran untuk setiap urutan adalah nilai pwm ternormalisasi pada langkah waktu berikutnya. Pembagian data dilakukan ke dalam himpunan pelatihan dan pengujian dengan rasio tertentu, kemudian indeks data diacak untuk menghindari bias urutan.

Arsitektur model terdiri atas satu lapisan LSTM berukuran kecil yang diikuti oleh lapisan dense dengan fungsi aktivasi relu dan diakhiri satu neuron keluaran dengan aktivasi sigmoid untuk menghasilkan nilai pwm ternormalisasi antara 0 dan 1. Model dikompilasi menggunakan fungsi rugi mean squared error dan dioptimasi menggunakan algoritma Adam dengan laju pembelajaran kecil. Proses pelatihan menggunakan fitur early stopping untuk menghentikan pelatihan ketika nilai rugi pada data validasi tidak membaik selama beberapa epoch, sehingga membantu mencegah overfitting dan mengurangi waktu pelatihan.

Setelah pelatihan selesai, model dievaluasi secara singkat pada sebagian data uji dengan membandingkan keluaran prediksi dan nilai pwm target. Model kemudian dikonversi ke format TensorFlow Lite menggunakan konverter bawaan, dengan mengaktifkan opsi optimasi agar ukuran model dan penggunaan memori lebih efisien. Berkas model hasil konversi disimpan dengan ekstensi tflite dan informasi normalisasi seperti nilai maksimum rpm dan PWM disimpan dalam berkas terpisah agar dapat digunakan kembali pada sisi mikrokontroler.

C. *Konversi Model ke Header C dan Integrasi TensorFlow Lite Micro*

Untuk dapat diintegrasikan ke dalam program ESP32, berkas model tflite dikonversi menjadi berkas header C yang berisi array konstanta bertipe unsigned char. Konversi ini dilakukan dengan skrip Python yang membaca berkas biner tflite dan menuliskannya sebagai array yang disejajarkan di memori, sekaligus mencatat panjang array.

ISSN (Online) 2747-0563
Volume 4 Tahun 2024

Berkas header tersebut kemudian diikutkan dalam proyek Arduino atau PlatformIO dan dipanggil dari program utama ESP32. Untuk menjalankan inferensi, program menggunakan pustaka TensorFlow Lite Micro yang menyediakan resolver operasi, arena memori untuk tensor, serta interpreter. Pada saat inisialisasi, interpreter diinstansiasi dengan model yang dimuat dari array, dan dialokasikan arena memori dengan ukuran yang mencukupi sesuai kebutuhan model.

D. *Algoritma Kendali LSTM Feedforward dan PID di ESP32*

Pada setiap periode kendali, program ESP32 membaca nilai setpoint kecepatan r(k) dan kecepatan aktual motor y(k) dalam satuan rpm. Kedua nilai ini dinormalisasi menggunakan parameter maksimum yang sama seperti pada tahap pelatihan agar rentang masukan ke model LSTM sesuai dengan saat pelatihan. Error kecepatan pada saat ke k didefinisikan sebagai

$$e(k) = r(k) - y(k) \quad (1)$$

Dengan e(k) adalah error kecepatan, r(k) adalah setpoint, dan y(k) adalah kecepatan aktual motor. Algoritma PID diskrit digunakan sebagai pengendali umpan balik. Bentuk posisi dari PID yang digunakan dalam penelitian ini dapat dituliskan sebagai

$$u_{\text{PID}}(k) = K_p e(k) + K_i \sum_{i=0}^{k} e\,(i)\,T_s + K_d \frac{e(k) - e(k-1)}{T_s} \quad (2)$$

di mana masing masing menyatakan penguatan proporsional, integral, dan derivatif, sedangkan Ts adalah periode sampling. Bagian integral dibatasi pada rentang tertentu untuk mencegah saturasi integrator, sedangkan bagian derivatif dapat diberi pembatas agar respon tidak terlalu agresif terhadap derau pengukuran. Keluaran PID selanjutnya dibatasi ke suatu rentang maksimum yang mewakili besarnya koreksi yang diperbolehkan. Untuk pembangkitan sinyal feedforward, nilai setpoint dan rpm hasil pengukuran dimasukkan ke dalam buffer deret waktu dengan panjang SEQ_LEN yang selalu diperbarui pada setiap siklus. Ketika buffer telah terisi penuh, deret tersebut digunakan sebagai masukan ke model LSTM melalui TensorFlow Lite Micro. Secara fungsional, keluaran model LSTM dapat dinyatakan sebagai

$$u_{\text{FF}}(k) = f_{\text{LSTM}}\,(x(k)) \quad (3)$$

Dengan $x(k)$ adalah vektor masukan yang berisi sejarah setpoint dan kecepatan pada beberapa langkah waktu sebelumnya, sedangkan $f\text{LSTM}(\cdot)$ menyatakan pemetaan nonlinier yang direpresentasikan oleh jaringan LSTM yang telah dilatih dan dikonversi ke format TensorFlow Lite. Keluaran $u\text{FF}(k)$ mula mula berada dalam bentuk ternormalisasi, kemudian diskala kembali ke rentang nilai PWM aktual. Sinyal PWM total yang dikirimkan ke driver motor dihitung sebagai penjumlahan antara sinyal feedforward dari LSTM dan koreksi PID berikut.

$$u(k) = u_{\text{FF}}(k) + u_{\text{PID}}(k) \quad (4)$$

Nilai $u(k)$ kemudian dijepit ke dalam rentang $[0, u\text{max}]$ agar sesuai dengan batas fisik aktuator, lalu diubah menjadi nilai PWM aktual yang dikirim ke modul PWM internal ESP32. Dengan strategi ini, LSTM memetakan dinamika nonlinier plant dan memberikan sinyal kendali awal yang mendekati nilai optimal, sementara PID memastikan error terhadap setpoint tetap kecil dan sistem kendali berada pada kondisi stabil.

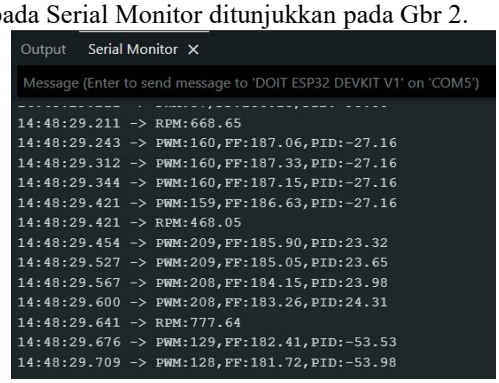
IV. HASIL DAN PEMBAHASAN

Bagian ini perlu Anda isi berdasarkan hasil eksperimen aktual. Struktur narasi yang disarankan sebagai berikut.

A. *Pengujian Respons Langkah*

Pengujian dilakukan pada sebuah motor DC yang digerakkan oleh driver berbasis H bridge dan dikendalikan oleh mikrokontroler ESP32. Frekuensi PWM diatur sebesar f_pwm Hz dengan resolusi N_bit bit sehingga nilai PWM berada pada rentang 0 sampai PWM_max. Kecepatan motor diukur menggunakan sensor encoder dengan resolusi R pulsa per putaran. Nilai kecepatan dinyatakan dalam satuan putaran per menit dengan periode pengukuran T_s milidetik.

Model LSTM yang digunakan memiliki panjang jendela deret waktu sebesar SEQ_LEN sampel, satu lapisan LSTM dengan n_unit unit, serta satu lapisan padat dengan n_dense neuron sebelum neuron keluaran. Inferensi model dilakukan pada ESP32 menggunakan TensorFlow Lite Micro sehingga keluaran model berupa sinyal PWM feedforward dapat dihitung pada setiap siklus kendali. Contoh log nilai PWM total, komponen feedforward, komponen PID, dan kecepatan motor pada Serial Monitor ditunjukkan pada Gbr 2.



Gbr. 2 Tampilan Serial Monitor ESP32 yang menampilkan nilai PWM, komponen feedforward, komponen PID, dan kecepatan motor

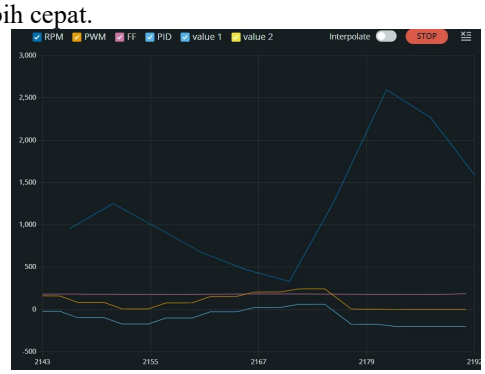
Model dijalankan menggunakan TensorFlow Lite Micro dengan ukuran arena memori sekitar M kilobyte. Parameter pengendali PID ditetapkan Kp = Kp_x, Ki = Ki_x, dan Kd = Kd_x yang sebelumnya ditentukan melalui uji coba awal sehingga konfigurasi PID murni memberikan respons yang stabil. Pengujian dilakukan untuk dua konfigurasi pengendali, yaitu PID murni dan pengendali hibrida LSTM feedforward plus PID. Pada setiap skenario, setpoint kecepatan dan pola perubahan yang diterapkan dibuat identik agar hasil kedua konfigurasi dapat dibandingkan secara langsung.

Respon kecepatan motor terhadap perubahan setpoint ditunjukkan pada Gbr 3. Motor diatur beroperasi pada kecepatan awal S1 rpm, kemudian setpoint dinaikkan secara tiba tiba menjadi S2 rpm pada detik ke t_step. Kurva RPM memperlihatkan bahwa pada konfigurasi PID murni kecepatan mengalami overshoot hingga sekitar O_pid persen dan memerlukan waktu penetapan sekitar Ts_pid detik hingga kembali berada dalam pita ±5 persen terhadap setpoint. Pada

ISSN (Online) 2747-0563
Volume 4 Tahun 2024

konfigurasi hibrida, overshoot berkurang menjadi sekitar O_hyb persen dengan waktu penetapan Ts_hyb detik. Error tunak rata rata juga berkurang dari Ess_pid rpm pada PID murni menjadi Ess_hyb rpm pada konfigurasi hibrida.

Perbedaan ini dapat dijelaskan dari perilaku sinyal kendali yang ditunjukkan pada bagian bawah Gbr 3. Sinyal feedforward dari LSTM cenderung menghasilkan nilai PWM dasar yang mendekati kebutuhan plant pada kondisi setpoint baru, sedangkan komponen PID hanya memberikan koreksi tambahan dalam bentuk sinyal positif atau negatif yang relatif kecil. Dengan demikian usaha kendali yang dilakukan PID berkurang, dan transisi ke kecepatan baru menjadi lebih halus serta lebih cepat.



Gbr. 3 Respon kecepatan motor dan sinyal PWM total, feedforward, serta PID terhadap perubahan setpoint

B. *Pengujian Gangguan Beban*

Pengujian gangguan beban dilakukan untuk mengevaluasi kemampuan sistem dalam mempertahankan kecepatan ketika terjadi perubahan torsi beban pada poros motor. Pada pengujian ini setpoint kecepatan dijaga tetap sebesar S_load rpm. Pada detik ke t_g1 beban mekanik tambahan dipasang pada poros motor, sedangkan pada detik ke t_g2 beban tersebut dilepaskan kembali.

Pada konfigurasi PID murni, penambahan beban menyebabkan kecepatan turun dari setpoint hingga sekitar drop_pid persen sebelum perlahan kembali naik. Waktu pemulihan hingga kecepatan kembali mendekati setpoint berada pada kisaran Tr_pid detik. Ketika beban dilepas, kecepatan melonjak di atas setpoint kemudian berangsur turun ke nilai semula. Hasil ini menunjukkan bahwa PID mampu mengoreksi perubahan kecepatan akibat gangguan, tetapi dengan deviasi yang relatif besar dan waktu pemulihan yang tidak terlalu cepat.

Pada konfigurasi hibrida LSTM feedforward plus PID, penurunan kecepatan akibat penambahan beban tercatat hanya sekitar drop_hyb persen dengan waktu pemulihan Tr_hyb detik yang lebih singkat. Nilai root mean square error terhadap setpoint juga lebih kecil dibandingkan konfigurasi PID murni. Hal ini menunjukkan bahwa kombinasi feedforward dan PID lebih mampu mempertahankan kecepatan pada sekitar nilai yang diinginkan ketika plant mengalami perubahan beban. Model LSTM yang dilatih dengan data operasi motor membantu memperkirakan nilai PWM yang sesuai untuk kondisi beban baru, sementara PID mengoreksi sisa error yang masih muncul. Pada naskah ini hasil pengujian gangguan beban disajikan dalam bentuk narasi tanpa grafik terpisah.

C. *Beban Komputasi dan Pemanfaatan Sumber Daya*

Selain performa dari sisi kecepatan dan error, penting untuk memastikan bahwa algoritma kendali dapat dijalankan dalam batas sumber daya yang tersedia di mikrokontroler ESP32. Pengukuran dilakukan terhadap waktu eksekusi satu siklus kendali yang meliputi pembacaan encoder, normalisasi data, pembaruan buffer deret waktu, inferensi LSTM, perhitungan PID, dan pembaruan sinyal PWM.

Hasil pengukuran menunjukkan bahwa rata rata waktu eksekusi satu siklus kendali pada konfigurasi hibrida adalah sekitar T_ctrl milidetik. Nilai ini masih lebih kecil dari periode sampling T_s sehingga ESP32 memiliki margin waktu yang cukup sebelum siklus berikutnya dimulai. Penggunaan memori untuk model LSTM, parameter, dan arena TensorFlow Lite Micro secara keseluruhan berada di kisaran M_tot kilobyte, yang masih berada di bawah kapasitas RAM yang tersedia sehingga tidak mengganggu proses lain seperti komunikasi serial dan tugas pemantauan.

Dibandingkan dengan konfigurasi PID murni, waktu eksekusi pada konfigurasi hibrida meningkat sekitar delta_T milidetik akibat adanya proses inferensi LSTM. Namun peningkatan ini masih dapat diterima mengingat peningkatan performa sistem yang diperoleh dari sisi pengurangan overshoot, penurunan error tunak, dan kemampuan penolakan gangguan beban. Dengan demikian dapat disimpulkan bahwa ESP32 mampu menjalankan kombinasi pengendali LSTM feedforward dan PID secara waktu nyata tanpa melebihi batas kemampuan komputasi dan memori yang tersedia.

V. KESIMPULAN

Penelitian ini telah merealisasikan sistem pengendali kecepatan motor DC berbasis mikrokontroler ESP32 yang menggabungkan sinyal feedforward hasil prediksi jaringan LSTM dengan pengendali umpan balik PID. Model LSTM dibangun dari data telemetri yang berisi setpoint, kecepatan aktual, dan PWM, kemudian dikonversi ke format TensorFlow Lite dan dijalankan pada ESP32 menggunakan TensorFlow Lite Micro. Hasil pengujian memperlihatkan bahwa model dengan ukuran relatif kecil masih dapat diinferensikan secara periodik tanpa melampaui batas waktu sampling maupun kapasitas memori yang tersedia pada mikrokontroler.

Dari sisi unjuk kerja kendali, konfigurasi hibrida LSTM feedforward dan PID menunjukkan kemampuan pelacakan setpoint yang lebih baik dibandingkan PID murni pada uji perubahan setpoint maupun pada saat sistem dikenai gangguan beban. LSTM berperan menyajikan perkiraan awal sinyal PWM yang sesuai dengan dinamika nonlinier motor, sedangkan PID melengkapi sebagai pengendali umpan balik untuk menjaga kestabilan dan mengecilkan error yang tersisa. Hal ini tercermin dari berkurangnya overshoot, mengecilnya error tunak, serta membaiknya waktu pemulihan menuju kondisi tunak.

Ke depan, beberapa pengembangan yang menarik untuk dilakukan antara lain pengujian berbagai arsitektur jaringan yang lebih kompak atau lebih dalam, penerapan teknik kuantisasi dan optimasi lain guna menurunkan ukuran model dan kebutuhan memori, serta evaluasi pada tipe motor dan

ISSN (Online) 2747-0563
Volume 4 Tahun 2024

variasi beban yang lebih luas. Selain untuk kendali kecepatan motor DC, pendekatan hibrida berbasis LSTM dan PID ini juga berpotensi diadaptasi pada sistem kendali nonlinier lain yang diimplementasikan pada platform mikrokontroler dengan sumber daya terbatas.

UCAPAN TERIMA KASIH

Judul untuk ucapan terima kasih dan referensi tidak diberi nomor. Terima kasih disampaikan kepada Tim SANTIKA yang telah meluangkan waktu untuk membuat *template* ini.

REFERENSI

[1] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," Transactions of the ASME, vol. 64, pp. 759–768, 1942
[2] K. J. Åström and T. Hägglund, PID Controllers: Theory, Design, and Tuning. ISA, 1995.
[3] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
[4] R. David, J. Duke, A. Jain, V. J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev, R. Rhodes, T. Wang, and P. Warden, "TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems," arXiv:2010.08678, 2020.
[5] Google AI for Developers, "Get started with microcontrollers (LiteRT for Microcontrollers)," dokumentasi resmi.
[6] Google AI for Developers, "Convert TensorFlow models (LiteRT converter)," dokumentasi resmi.
[7] TensorFlow, "TensorFlow Lite for Microcontrollers (tflite-micro) repository," dokumentasi dan kode sumber resmi.
[8] Espressif Systems, "ESP32 Datasheet," dokumentasi resmi (PDF).
[9] Espressif Systems, "LED Control (LEDC) API, Arduino-ESP32 documentation," dokumentasi resmi.
[10] Espressif Systems, "Pulse Counter (PCNT) API, ESP-IDF Programming Guide," dokumentasi resmi.
[11] J. Rantung and H. Luntungan, "DC Motor PID Controller with PWM Feedback," artikel open access (PDF).
[12] E. Schesch, J. E. Normey-Rico, and J. C. Adamy, "Analysis of Anti-windup Techniques in PID Control of Processes with Measurement Noise," prosiding (PDF).