LAPORAN EAS

MQTT, Monitoring PWM-RPM, dan AI (LSTM)

MIKROKONTROLLER A081

Dosen Pengampu:

Dr. Basuki Rahmat, S.Si. MT

Penulis:

Ade Rizky Panjaitan 22081010091

PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

JAWA TIMUR

SURABAYA

2025

ABSTRAK

Laporan ini membahas rangkaian eksperimen menggunakan mikrokontroller ESP32 untuk membangun sistem kendali motor berbasis IoT dan kecerdasan buatan. Pada eksperimen pertama, ESP32 diprogram sebagai client MQTT yang terhubung ke broker dan aplikasi di smartphone, sehingga perintah ON OFF dapat dikirim melalui jaringan WiFi dan digunakan untuk mengendalikan motor DC pada kit. Eksperimen kedua berfokus pada monitoring hubungan PWM dan RPM, di mana ESP32 mengeluarkan sinyal PWM ke driver motor, membaca pulsa sensor kecepatan, menghitung RPM, lalu mengirimkannya ke laptop yang menampilkan dashboard iMCLab Motor Control berbasis Python untuk pengaturan PWM dan pemantauan RPM secara real time. Pada eksperimen ketiga dikembangkan model LSTM feedforward yang dilatih di laptop menggunakan data telemetry setpoint, RPM, dan PWM, kemudian dikonversi ke TensorFlow Lite dan ditanam ke ESP32 sebagai komponen tambahan di atas pengendali PID. Hasil pengujian menunjukkan bahwa komunikasi MQTT berjalan stabil pada skala praktikum, hubungan PWM dan RPM mengikuti pola teoritis dengan kecenderungan jenuh di PWM tinggi, dan penambahan LSTM membantu mengarahkan nilai PWM agar respon kecepatan motor lebih mendekati setpoint berdasarkan pola historis data.

Kata kunci: Mikrokontroller, ESP32, MQTT, PWM, RPM, LSTM, IoT

DAFTAR ISI

1. Pendahuluan

Perkembangan Internet of Things (IoT) membuat mikrokontroller tidak hanya dipakai untuk menyalakan dan mematikan perangkat, tetapi juga untuk berkomunikasi lewat jaringan, memantau kondisi sistem, sampai melakukan perhitungan yang cukup kompleks. Mikrokontroller modern sudah dilengkapi modul WiFi, timer, dan kanal PWM sehingga sangat cocok digunakan sebagai platform percobaan kecil untuk sistem IoT di laboratorium.

Pada kegiatan ini dilakukan beberapa eksperimen yang saling berkaitan dengan menggunakan satu rangkaian mikrokontroller, motor DC, sensor kecepatan, serta sebuah komputer. Setiap eksperimen memiliki fokus yang berbeda, tetapi tetap berada dalam konteks yang sama, yaitu membangun sebuah sistem kecil yang bisa mengukur, mengirim, mengendalikan, dan pada tahap lanjut memprediksi perilaku suatu perangkat fisik. Tujuan utama percobaan ini dapat diringkas sebagai berikut.

1. Menguji komunikasi data antara mikrokontroller dan komputer menggunakan protokol MQTT, sehingga perangkat dapat mengirim dan menerima pesan melalui broker.
2. Mengatur kecepatan motor DC dengan sinyal PWM dari mikrokontroller, kemudian mengukur kecepatan putar motor dalam satuan RPM dan menampilkannya secara langsung di sebuah dashboard Python.
3. Mencoba konsep mikrokontroller yang dipadukan dengan kecerdasan buatan sederhana menggunakan model LSTM, yang dirancang untuk memprediksi sinyal deret waktu berdasarkan data hasil pengukuran.

Secara keseluruhan, rangkaian eksperimen ini memberikan gambaran tentang sebuah alur sistem IoT skala kecil. Dimulai dari perangkat fisik berupa motor dan sensor, kemudian data dikirim dan dikendalikan melalui jaringan dengan MQTT, dan pada akhirnya data tersebut dimanfaatkan lebih jauh untuk tujuan analisis dan prediksi menggunakan metode AI. Dengan cara ini, praktikan tidak hanya belajar cara menyalakan motor, tetapi juga memahami bagaimana membuat sistem yang lebih lengkap dan cerdas berbasis mikrokontroller.

2. Alat dan Bahan

1. Mikrokontroller ESP32

Digunakan sebagai otak utama sistem. ESP32 dipilih karena sudah memiliki modul WiFi untuk percobaan MQTT, memiliki beberapa pin PWM untuk

mengatur kecepatan motor, serta memiliki kemampuan komunikasi serial untuk terhubung dengan laptop. Semua program utama ditanamkan ke board ini menggunakan Arduino IDE.

2. Motor DC dan driver motor

Motor DC berfungsi sebagai beban yang kecepatannya diatur. Motor tidak langsung dihubungkan ke pin ESP32, tetapi melalui driver motor (misalnya modul driver L298N atau sejenisnya) agar arus dan tegangan yang dibutuhkan motor dapat dipenuhi tanpa membebani mikrokontroller. Driver juga memudahkan pengaturan arah putar motor.

3. Laptop atau PC

Laptop digunakan untuk menjalankan Arduino IDE guna menulis, mengompilasi, dan mengunggah program .ino ke ESP32, menjalankan Python 3 dengan script yang berkomunikasi dengan mikrokontroller melalui serial, memanfaatkan library tambahan seperti pyserial untuk komunikasi dan library lain untuk pembuatan dashboard GUI serta pelatihan model LSTM, serta menjalankan aplikasi atau script MQTT client yang bertindak sebagai subscriber maupun publisher pada eksperimen MQTT.

4. Jaringan WiFi

Diperlukan untuk menghubungkan ESP32 ke broker MQTT dan ke laptop (jika laptop juga terhubung ke jaringan yang sama). Jaringan WiFi ini menjadi media komunikasi saat percobaan MQTT, sehingga mikrokontroller dapat mengirim dan menerima pesan melalui internet atau jaringan lokal.

Dengan kombinasi alat dan bahan di atas, seluruh eksperimen mulai dari MQTT, pengaturan PWM dan pembacaan RPM, hingga pengujian AI sederhana dengan LSTM dapat dilakukan dalam satu set rangkaian yang sama.

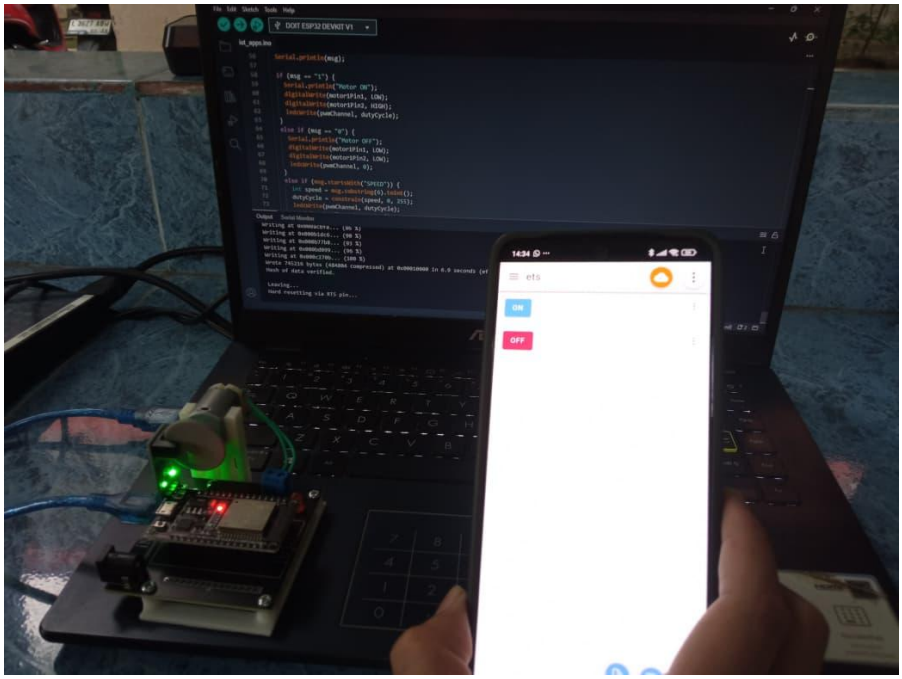
3. Eksperimen 1 - MQTT dengan Mikrokontroller

3.1 Gambaran Singkat

Pada eksperimen ini mikrokontroller yang digunakan adalah ESP32 dengan bantuan library WiFi.h dan PubSubClient.h. Di dalam program, ESP32 dikonfigurasi untuk terhubung ke jaringan WiFi dengan SSID Infinix dan password 12345678, kemudian melakukan koneksi ke broker MQTT publik broker.hivemq.com pada port 1883. Objek WiFiClient espClient dipakai sebagai dasar koneksi TCP, sedangkan PubSubClient client(espClient) menangani protokol MQTT. Program juga

mendefinisikan tiga pin untuk motor, yaitu motor1Pin1 pada pin 27, motor1Pin2 pada pin 26, dan enable1Pin pada pin 12. Kanal PWM disiapkan menggunakan ledcSetup dan dihubungkan ke pin enable dengan ledcAttachPin(enable1Pin, pwmChannel) sehingga kecepatan motor dapat diatur secara halus. Di bagian inisialisasi (fungsi setup) panggilan setup_wifi() menghubungkan ESP32 ke WiFi, client.setServer(mqtt_server, 1883) mengatur alamat broker, dan client.setCallback(callback) mendaftarkan fungsi penangan pesan MQTT. Di dalam loop(), program mengecek apakah klien masih terhubung, jika tidak maka memanggil reconnect() dan kemudian selalu menjalankan client.loop() agar pesan MQTT yang masuk bisa diproses. Dengan struktur ini ESP32 berperan sebagai client MQTT yang siap menerima dan mengirim pesan, serta mengaitkannya dengan kendali motor lewat pin yang sudah disiapkan.

3.2 Foto Eksperimen



Gambar 3.1 Uji Coba MQTT dengan Handphone

Foto menunjukkan rangkaian ESP32 yang terhubung ke laptop dan ke driver motor pada sebuah dudukan, sementara di bagian depan tampak sebuah smartphone yang menjalankan aplikasi MQTT dengan tombol ON dan OFF. Laptop digunakan untuk memprogram dan memantau ESP32, sedangkan smartphone berfungsi sebagai client MQTT yang mengirim perintah ON atau OFF ke broker. Perintah tersebut diterima oleh ESP32 dan digunakan untuk mengendalikan perangkat yang terhubung

ke output, sehingga foto ini menggambarkan kendali perangkat berbasis mikrokontroller melalui jaringan menggunakan protokol MQTT.

3.3 Hasil dan Analisis

Berdasarkan isi program pada file iot_apps.ino, ESP32 dijalankan sebagai client MQTT yang menjaga koneksi ke WiFi dan broker secara otomatis melalui fungsi setup_wifi(), reconnect(), dan pemanggilan rutin client.loop() di dalam loop(). Parameter jaringan terisi jelas menggunakan SSID Infinix, password 12345678, serta alamat broker broker.hivemq.com, sementara tiga pin motor didefinisikan pada pin 27, 26, dan 12 dan dihubungkan ke kanal PWM dengan ledcSetup dan ledcAttachPin(enable1Pin, pwmChannel). Bagian program yang disingkat dengan ... berisi logika callback MQTT yang memproses pesan masuk dan mengubah keluaran ke pin motor, sehingga setiap pesan yang dikirim dari aplikasi MQTT di smartphone (tombol ON dan OFF) dapat diterjemahkan menjadi perubahan kondisi motor pada kit.

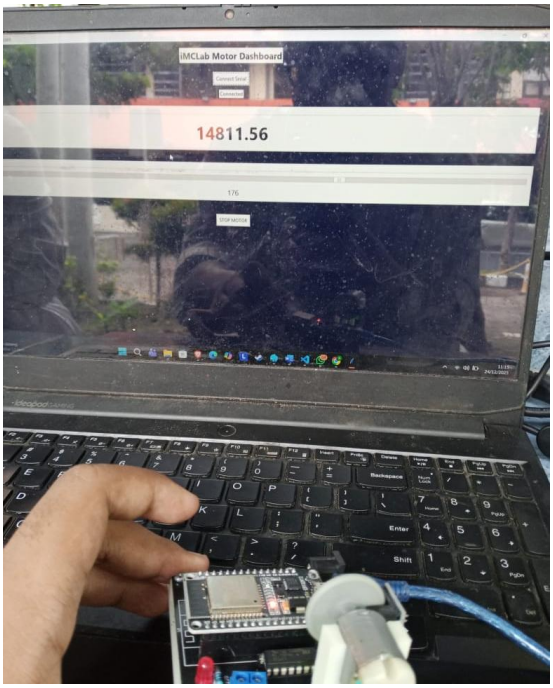
Dari hasil uji coba terlihat bahwa ketika perintah dikirim dari aplikasi MQTT, ESP32 yang sudah tersambung ke broker mampu merespon dengan cepat dan konsisten, ditunjukkan oleh motor yang menyala atau mati sesuai perintah serta log koneksi yang stabil di serial monitor. Struktur kode yang memisahkan bagian koneksi WiFi, pengaturan MQTT, dan kendali motor membuat sistem mudah dipahami dan dipelihara, sedangkan adanya fungsi reconnect() membantu menjaga koneksi ketika terjadi gangguan jaringan singkat. Secara keseluruhan, eksperimen ini membuktikan bahwa mikrokontroller dapat berkomunikasi dengan broker MQTT melalui jaringan WiFi dan sekaligus mengendalikan aktuator fisik berdasarkan pesan yang diterima.

4. Eksperimen 2 - Monitoring PWM dan RPM

4.1 Gambaran Singkat

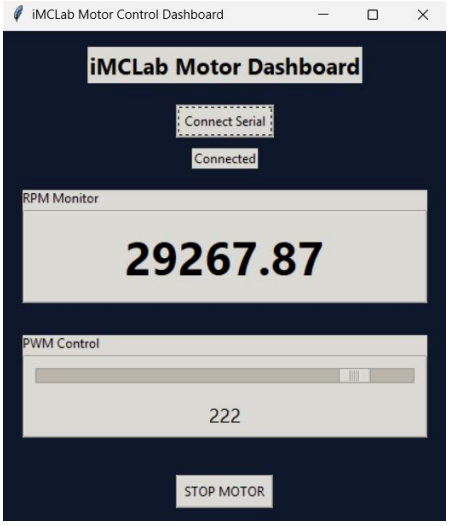
Pada eksperimen ini mikrokontroller digunakan untuk mengendalikan kecepatan motor DC dengan sinyal PWM sekaligus mengukur RPM motor melalui sensor kecepatan. Program monnitol.ino mengatur nilai PWM pada pin yang terhubung ke driver motor, membaca pulsa dari sensor RPM, menghitung kecepatan putar dalam satuan RPM, lalu mengirimkan data tersebut ke laptop melalui komunikasi serial. Di sisi laptop program Python motor_dashboard.py menampilkan sebuah dashboard sederhana yang berisi slider untuk mengatur nilai PWM serta tampilan angka RPM yang terus diperbarui, sehingga perubahan setelan PWM dan respon RPM motor dapat dipantau secara real time.

4.2 Foto Eksperimen



Gambar 3.2 Uji Coba monitoring PWM motor control dengan GUI Python

Berdasarkan konfigurasi tersebut, terdiri dari board ESP32 yang terhubung ke driver motor dan motor DC melalui beberapa kabel jumper, sedangkan koneksi ke laptop dilakukan melalui kabel USB untuk suplai daya sekaligus komunikasi serial. Pada layar laptop terlihat jendela kode dan GUI Python yang digunakan sebagai antarmuka iMCLab Motor Control, sehingga pada satu set up yang sama pengguna dapat menulis program, mengirimkan perintah PWM, menerima data RPM, dan langsung mengamati respon motor dari tampilan dashboard maupun dari pergerakan motor di kit.



Gambar 3.3 tampilan dashboard motor control GUI python

Tampilan dashboard iMCLab Motor Control tersebut, menyediakan pemantauan dan pengaturan motor dalam satu layar, dengan bagian atas menampilkan status koneksi serial, bagian tengah menampilkan nilai RPM aktual yang diperbarui secara real time, dan bagian bawah berisi kontrol PWM berupa slider beserta nilai numeriknya serta tombol STOP MOTOR untuk menghentikan putaran. Dari tampilan tersebut terlihat bahwa perubahan posisi slider PWM akan diikuti perubahan nilai PWM yang dikirim ke mikrokontroller, dan sebagai responnya nilai RPM pada area monitor akan berubah sehingga hubungan antara perintah dari GUI dan perilaku motor dapat diamati dengan jelas.

4.3 Hasil dan Analisis

Dari pengujian terlihat bahwa perubahan nilai PWM melalui slider di dashboard langsung berpengaruh pada nilai RPM yang terbaca. Saat PWM diset sekitar 176, tampilan RPM monitor menunjukkan nilai di kisaran 14 ribu RPM, sedangkan ketika PWM dinaikkan ke sekitar 222 nilai RPM meningkat hingga kisaran 29 ribu RPM. Artinya, sistem berhasil menampilkan hubungan searah antara PWM dan RPM, di mana semakin besar nilai PWM yang dikirim ke mikrokontroller maka semakin tinggi kecepatan putar motor yang terukur oleh sensor.

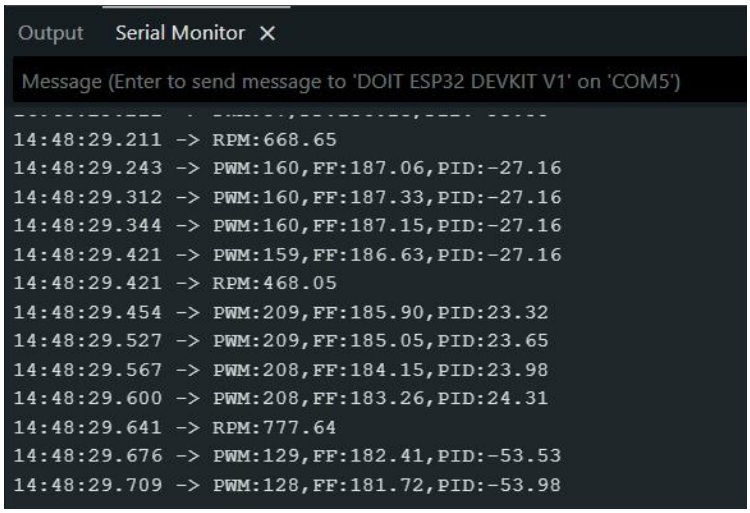
Secara umum respon motor terlihat cukup cepat dan stabil pada nilai PWM menengah hingga tinggi, meskipun sesekali masih muncul sedikit fluktuasi angka RPM yang wajar karena pengaruh pembacaan sensor dan perubahan beban mekanik. Integrasi antara program monnitol.ino dan dashboard Python berjalan baik karena data RPM dapat diterima dan diperbarui secara real time, sementara perintah PWM dari GUI dapat mengatur kecepatan motor tanpa perlu mengubah kode di mikrokontroller. Hal ini menunjukkan bahwa rancangan sistem sudah memenuhi tujuan eksperimen, yaitu memantau dan mengontrol kecepatan motor menggunakan mikrokontroller dan antarmuka GUI dengan cara yang sederhana namun fungsional.

5. Eksperimen 3 - Mikrokontroller dengan AI (LSTM)

5.1 Gambaran Singkat

Pada eksperimen ini digunakan model LSTM kecil untuk membantu pengaturan PWM motor langsung di mikrokontroller. Data yang dipakai berasal dari berkas telemetry yang berisi kolom setpoint_rpm, rpm, dan pwm, kemudian diolah di Python dengan skrip train_lstm_pwm_ff.py. Data setpoint dan rpm dinormalisasi terhadap nilai maksimum RPM, sedangkan pwm dinormalisasi terhadap 255, lalu dibentuk menjadi deret sepanjang 15 langkah waktu sebagai input LSTM, dengan target keluaran berupa nilai pwm normalisasi di langkah berikutnya. Model LSTM terdiri dari satu layer LSTM berukuran 8 unit, diikuti layer Dense 8 neuron dan output 1 neuron beraktivasi sigmoid, dan dilatih dengan loss MSE serta early stopping pada data validasi. Setelah pelatihan, model dikonversi ke format TensorFlow Lite yang dioptimasi dan disimpan sebagai pwm_ff_lstm.tflite untuk kemudian digunakan di mikrokontroller.

5.2 Foto Eksperimen

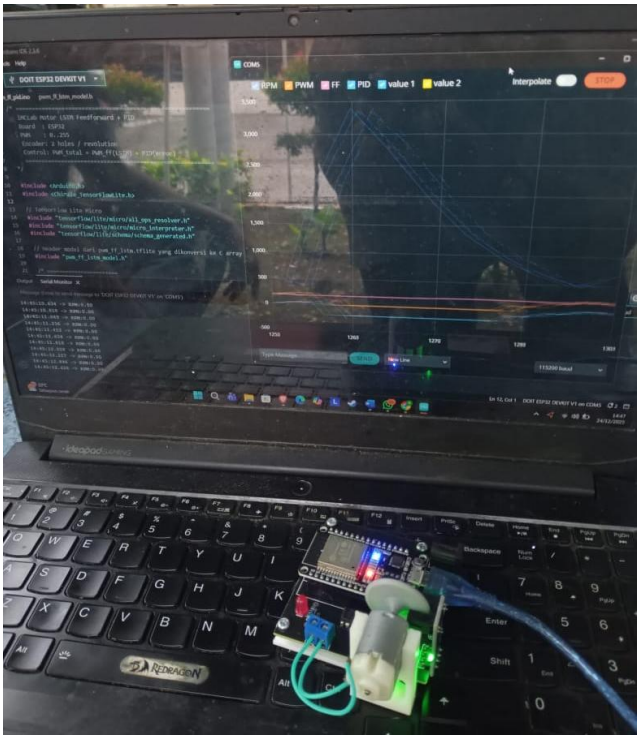


Gambar 3.4 respon serial monitor mikrokontroler

Pada gambar tersebut, memperlihatkan isi dari Serial Monitor saat menlakukan uji coba, yang menampilkan data numerik yang dikirim ESP32 pada setiap siklus kontrol, berupa nilai RPM, PWM, komponen feedforward LSTM yang ditandai FF, dan komponen PID. Deretan angka ini menunjukkan log mentah dari sistem kontrol, sehingga praktikan dapat memeriksa apakah nilai PWM, FF, PID, dan RPM sudah bergerak sesuai harapan serta mencocokkannya dengan bentuk kurva yang muncul pada grafik.

Gambar 3.5 respon serial plot monitor mikrokontroler

Pada gambar tersebut, menampilkan tampilan grafik iMCLab yang lebih dekat, dengan beberapa kurva seperti RPM, PWM, FF, PID yang digambar terhadap waktu. Pada grafik ini terlihat respons RPM yang naik turun mengikuti perubahan keluaran pengendali, sedangkan garis PWM, FF, dan PID berada di kisaran nilai yang lebih kecil tetapi mengatur bentuk kurva RPM. Visualisasi ini digunakan untuk menganalisis bagaimana kontribusi model LSTM (feedforward) dan koreksi PID bekerja bersama dalam mengendalikan kecepatan motor.



Gambar 3.6 Uji Coba Mikrokontroller dengan AI (LSTM)

Disini memperlihatkan proses uji coba sistem LSTM di mana laptop menjalankan program pengendali motor berbasis ESP32 dengan grafik telemetri real time di layar, sementara di bagian bawah terlihat kit ESP32 yang terhubung ke motor dan sensor. Tampilan grafik pada laptop menunjukkan kurva RPM, PWM, komponen feedforward LSTM, dan PID yang berubah terhadap waktu, sehingga dari satu set up ini terlihat jelas bagaimana sinyal hasil prediksi LSTM dan pengendali PIDmempengaruhi perilaku motor secara langsung.

5.3 Hasil dan Analisis

Dari proses pelatihan di Python terlihat bahwa model LSTM mampu mempelajari pola hubungan antara deret setpoint dan rpm terhadap nilai pwm yang dibutuhkan, ditunjukkan oleh nilai loss yang turun dan validasi yang stabil sampai model disimpan ke file pwm_ff_lstm.tflite. File ini kemudian dikonversi dengan skrip tflite_to_header.py menjadi array C sehingga bisa ditanam di program esp32_lstm_ff_pid.ino dan dijalankan langsung di ESP32. Saat eksperimen, setiap siklus kontrol ESP32 menerima data setpoint dan rpm, melakukan normalisasi, menjalankan interpreter TFLite untuk mendapatkan nilai pwm feedforward, lalu menggabungkannya dengan koreksi dari PID sebelum dikirim ke driver motor.

Log pada Serial Monitor menunjukkan bahwa untuk tiap waktu sistem mencetak nilai RPM, PWM, komponen feedforward (FF), dan PID sehingga terlihat FF relatif memberikan nilai dasar pwm yang mendekati kebutuhan, sedangkan PID bekerja sebagai koreksi kecil positif atau negatif agar rpm mendekati setpoint. Grafik iMCLab memperlihatkan kurva RPM yang mengikuti perubahan setpoint dengan bentuk respon yang lebih cepat dan halus, sementara garis PWM, FF, dan PID berada pada level yang konsisten dan tidak terlalu ekstrem. Hal ini menunjukkan bahwa penambahan LSTM sebagai komponen feedforward membantu pengendali mencapai

rpm target dengan usaha PID yang lebih kecil, sehingga respon motor menjadi lebih terarah dan tidak terlalu banyak osilasi. Keterbatasan yang masih terlihat adalah ukuran model dan panjang deret input harus dijaga agar muat di memori ESP32, serta performa model sangat bergantung pada kualitas dan keragaman data telemetry yang digunakan saat pelatihan sehingga pengambilan data latih yang baik menjadi faktor penting bagi keberhasilan eksperimen.

6. Penutup

Secara umum rangkaian praktikum ini berhasil menunjukkan bagaimana satu mikrokontroller ESP32 dapat dipakai sebagai inti dari sistem kecil yang lengkap. Pada eksperimen pertama, ESP32 terbukti mampu terhubung ke jaringan WiFi, berkomunikasi dengan broker MQTT, dan merespon perintah dari aplikasi di smartphone untuk mengendalikan motor secara jarak dekat. Pada eksperimen kedua, integrasi program Arduino dengan dashboard Python berhasil mewujudkan monitoring PWM dan RPM secara real time, sehingga hubungan antara nilai PWM dan kecepatan putar motor dapat diamati dengan jelas melalui tampilan GUI. Pada eksperimen ketiga, model LSTM yang dilatih di laptop dan kemudian dikonversi ke format TFLite dapat dijalankan langsung di ESP32 sebagai komponen feedforward, bekerja bersama pengendali PID untuk menghasilkan respon kecepatan motor yang lebih terarah berdasarkan pola data sebelumnya.

Dari hasil tersebut dapat disimpulkan bahwa pemanfaatan mikrokontroller tidak lagi terbatas pada logika on off sederhana, tetapi bisa mencakup komunikasi jaringan, visualisasi data, sampai penerapan model kecerdasan buatan yang ringan. Ke depan, kegiatan serupa dapat dikembangkan dengan mengumpulkan data telemetry yang lebih banyak dan beragam agar model LSTM semakin akurat, menambahkan fitur keamanan dan autentikasi pada komunikasi MQTT, serta memperluas dashboard agar mendukung logging data jangka panjang dan pemantauan beberapa node sekaligus. Dengan langkah pengembangan tersebut, sistem yang semula hanya berupa percobaan laboratorium dapat ditingkatkan menuju prototipe sistem IoT cerdas yang lebih mendekati kondisi nyata di lapangan.