Jurnal Teknologi dan Manajemen Industri Terapan (JTMIT) Vol. 5, No. 1, Maret 2026 pp. 407 - 414
P-ISSN: 2829-0232 E-ISSN: 2829-0038

Implementasi Komunikasi Real-Time Berbasis WebSocket pada Sistem Quickshifter Dual-Mikrokontroler

Erlangga Putra Ramadhan[1], Mohammad Idhom[2], Firza Prima Aditiawan[3]
[1,3)] Informatika, Fakultas Ilmu Komputer, UPN "Veteran" Jawa Timur
Jl. Raya Rungkut Madya, Gunung Anyar, Surabaya, Jawa Timur 60294
Email: 22081010075@student.upnjatim.ac.id, firzaprima.if@upnjatim.ac.id
[2)] Sains Data, Fakultas Ilmu Komputer, UPN "Veteran" Jawa Timur
Jl. Raya Rungkut Madya, Gunung Anyar, Surabaya, Jawa Timur 60294
Email: idhom@upnjatim.ac.id

ABSTRAK

Penelitian ini mengimplementasikan komunikasi real-time berbasis WebSocket pada sistem quickshifter sepeda motor dengan arsitektur dual-mikrokontroler. Arduino Nano digunakan untuk menangani proses inti quickshifter, meliputi pembacaan sensor CKP, perhitungan RPM, serta pengendalian pemutusan pengapian, sementara ESP32-C3 berfungsi sebagai gateway komunikasi nirkabel melalui Wi-Fi dan WebSocket. Kedua mikrokontroler berkomunikasi menggunakan UART dengan baudrate 115200. Evaluasi kinerja difokuskan pada parameter latensi, jitter, packet loss, serta kepraktisan penggunaan, dengan USB Serial sebagai pembanding. Hasil pengujian menunjukkan bahwa WebSocket memiliki latensi rata-rata 19,81 ms dan jitter 1,79 ms tanpa packet loss, sedikit lebih tinggi dibandingkan USB Serial namun masih dalam batas toleransi untuk aplikasi real-time. Selain itu, sistem mampu melakukan penyetelan dan pemantauan RPM secara responsif serta stabil. Implementasi ini membuktikan bahwa protokol WebSocket pada arsitektur dual-mikrokontroler mampu menyeimbangkan kebutuhan performa real-time dan fleksibilitas koneksi nirkabel pada sistem quickshifter.

Kata kunci: Quickshifter, WebSocket, ESP32-C3, Arduino Nano, Real-time.

ABSTRACT

This study implements a real-time WebSocket-based communication system for a motorcycle quickshifter using a dual-microcontroller architecture. Arduino Nano is responsible for core quickshifter operations, including CKP sensor reading, RPM calculation, and ignition cut control, while ESP32-C3 acts as a wireless communication gateway via Wi-Fi and WebSocket. The two microcontrollers communicate through UART at a baud rate of 115200. Performance evaluation focuses on latency, jitter, packet loss, and usability, with USB Serial communication used as a benchmark. Experimental results show that WebSocket achieves an average latency of 19.81 ms and a jitter of 1.79 ms with zero packet loss, slightly inferior to USB Serial but still acceptable for real-time applications. The system demonstrates responsive and stable RPM monitoring and configuration. These results indicate that WebSocket implementation on a dual-microcontroller architecture effectively balances real-time performance and wireless flexibility for quickshifter systems.

Keywords: *Quickshifter, WebSocket, ESP32-C3, Arduino Nano, Real-time*

Pendahuluan

Perkembangan teknologi Internet of Things (IoT) telah mendorong integrasi sistem embedded dengan konektivitas jaringan, termasuk pada sektor transportasi. Dalam praktiknya, komunikasi antar perangkat IoT dapat dilakukan melalui koneksi berkabel maupun nirkabel. Meskipun koneksi berkabel memiliki keunggulan dalam stabilitas dan determinisme waktu, keterbatasan dari sisi kepraktisan dan portabilitas menjadikan koneksi nirkabel lebih banyak digunakan, meskipun keandalannya masih dipengaruhi oleh berbagai faktor lingkungan [1], [2].

Keandalan komunikasi nirkabel sangat dipengaruhi oleh teknologi yang digunakan, seperti Bluetooth dan Wi-Fi. Bluetooth menawarkan kemudahan konfigurasi dan konsumsi daya yang rendah, namun memiliki keterbatasan pada bandwidth dan jangkauan. Sebaliknya, Wi-Fi unggul dalam kecepatan transmisi, jangkauan, dan stabilitas, sehingga lebih sesuai untuk aplikasi yang memerlukan pertukaran data cepat dan berkelanjutan [3].

Protokol komunikasi Wi-Fi yang digunakan juga memengaruhi performa koneksi nirkabel. Dalam kasus IoT, protokol yang paling sering digunakan antara lain yaitu HTTP (*Polling* dan *long polling*), MQTT, serta WebSocket. HTTP *Polling* memerlukan sebuah proses berurutan yang berupa *request* untuk membuka koneksi dan *response* untuk membalas permintaan lalu menutup koneksi ketika transmisi selesai. Sedangkan WebSocket tidak memerlukan adanya *request* dan *response*, koneksi juga tidak ditutup setelah transmisi data selesai sehingga latensi yang dihasilkan menjadi lebih rendah dan cocok untuk

penggunaan transmisi data "*real-time*" [4], [5]. MQTT juga tidak lebih baik dari WebSocket karena memiliki latensi yang cukup tinggi serta memerlukan koneksi internet [6].

IoT juga memengaruhi kenyamanan transportasi, contohnya pada teknologi *Quickshifter* yang memungkinkan pengendara sepeda motor kopling dapat menaikkan gigi persneling tanpa perlu menarik tuas kopling untuk meminimalisir kelelahan otot lengan [7]. *Quickshifter* bekerja dengan mendeteksi usaha perpindahan persneling menggunakan sensor yang peka terhadap perubahan gaya tekan, disebut dengan *force-sensing resistor* (FSR) [8]. Teknologi *Quickshifter* ini perlu dilakukan proses penyetelan untuk menyesuaikan waktu putus pengapian mesin tiap rentang RPM agar perpindahan gigi bisa terjadi lebih halus dan aman. Penyetelan waktu putus pengapian harus bersamaan dengan pemantauan RPM mesin terkini. Pada penelitian sebelumnya [9], penyetelan serta pemantauan RPM mesin pada *Quickshifter* masih memakai koneksi kabel yang dirasa kurang praktis sehingga memerlukan implementasi koneksi nirkabel untuk mengejar kepraktisan.

Maka dari itu, penelitian ini bertujuan untuk mengimplementasikan koneksi nirkabel pada perangkat *Quickshifter* melalui protokol WebSocket dengan tetap mempertimbangkan keandalan transmisi data *real-time*. Implementasi dicapai menggunakan kombinasi dua mikrokontroler yang memiliki tugas masing-masing agar mikrokontroler tidak terbebani dengan dua tugas sekaligus [10], [11] atau juga bisa digunakan sebagai mikrokontroler cadangan ketika salah satu mengalami kerusakan[12]. Pada penelitian ini, Arduino Nano digunakan untuk kalkulasi proses *Quickshifter* dan ESP32-C3 untuk melakukan komunikasi nirkabel melalui WebSocket mewakili Arduino Nano (ESP32-C3 menjadi *gateway* komunikasi).

Meskipun teknologi *quickshifter* telah banyak diteliti, sebagian besar penelitian sebelumnya masih berfokus pada aspek mekanik, pemilihan sensor, serta kontrol pemutusan pengapian secara lokal. Di sisi lain, studi yang membahas integrasi komunikasi *real-time* nirkabel pada sistem *quickshifter*, khususnya dengan arsitektur dual-mikrokontroler, masih sangat terbatas. Penelitian yang secara kuantitatif mengevaluasi performa komunikasi nirkabel yang meliputi latensi, *jitter*, dan *packet loss* dalam konteks *quickshifter* hampir belum ditemukan. Oleh karena itu, diperlukan penelitian yang secara khusus mengkaji keandalan dan karakteristik komunikasi *real-time* nirkabel pada sistem *quickshifter* sebagai dasar pengembangan sistem yang lebih fleksibel tanpa mengorbankan kinerja *real-time*.

Kebaruan penelitian ini terletak pada implementasi dan evaluasi kuantitatif komunikasi *real-time* berbasis WebSocket pada sistem *quickshifter* dengan arsitektur dual-mikrokontroler, yang dianalisis melalui pengukuran latensi, *jitter*, dan *packet loss* serta dibandingkan langsung dengan komunikasi USB Serial sebagai *baseline*.

Metode Penelitian

Penelitian ini menggunakan desain eksperimen komparatif yang berfokus pada pengujian performa komunikasi *real-time*, dengan membandingkan protokol WebSocket berbasis Wi-Fi dan komunikasi USB Serial berdasarkan parameter latensi, *jitter*, dan *packet loss*.

Perancangan Sistem

Sistem dirancang menggunakan dua unit pemroses utama menggunakan mikrokontroler untuk konfigurasi yang lebih mudah dan familier [13]. Arduino Nano (ATMega328p) dipilih untuk bertugas membaca sensor CKP (*Crankshaft Position*) guna menghitung RPM, membaca sensor tuas persneling, dan mengendalikan MOSFET karena Arduino Nano memiliki voltase kerja 5V yang cukup mudah untuk mengendalikan dan membaca sensor dibandingkan dengan voltase 3.3V, sesuai dengan spesifikasi Arduino Nano [14]. ESP32-C3 bertugas menyediakan Access Point Wi-Fi dan *Server* WebSocket untuk komunikasi dengan klien. ESP32-C3 dipilih karena memiliki fitur yang melimpah untuk koneksi nirkabel dengan ukuran papan yang sangat kecil dan hemat daya dibanding ESP32 standar [15]. Kedua mikrokontroler tersebut berkomunikasi melalui jalur UART dengan *baudrate* 115200 yang sesuai untuk kebutuhan komunikasi *real-time* [16]. ESP32-C3 akan menerima pesan dari klien dan langsung diteruskan menuju Arduino Nano. Begitu pula pesan yang berasal dari Arduino Nano akan langsung diteruskan ke klien.

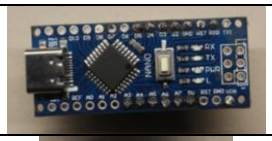
Konfigurasi pemutusan pengapian akan dikerjakan oleh MOSFET P-Channel dan bukan relay untuk mencapai kondisi *Normally Close* (NC) sembari menghindari pergerakan mekanis [17], sesuai perintah dari Arduino Nano. Lama pemutusan pengapian sesuai dengan nilai yang tersimpan pada EEPROM Arduino Nano. EEPROM dapat menyimpan data secara *persistent* [18]. Begitu juga EEPROM pada Arduino Nano, walaupun dilakukan perubahan pada kode, nilai yang tersimpan pada EEPROM tidak akan berubah jika tidak secara aktif diubah.

Perancangan Perangkat Keras

Perangkat keras yang digunakan disesuaikan dengan kebutuhan dari sistem. Dua mikrokontroler, Arduino Nano dan ESP32-C3 dipakai untuk melakukan tugas yang berbeda tetapi saling berkomunikasi. Untuk menjamin kendaraan tetap berfungsi normal ketika modul rusak, digunakan MOSFET P-Channel yang ditarik ke GND agar secara standar pengapian tetap terhubung. Selain itu, penggunaan MOSFET yang tidak memiliki gerak mekanik meminimalisir potensi terjadinya kerusakan pada "saklar".Kemudian, perbedaan tegangan kerja sepeda motor (12V) dengan mikrokontroler (5V dan 3.3V) memerlukan penyesuaian agar sistem dapat berjalan normal tanpa mengalami kerusakan. Penurunan tegangan dilakukan memakai DC-to-DC Stepdown untuk suplai tegangan

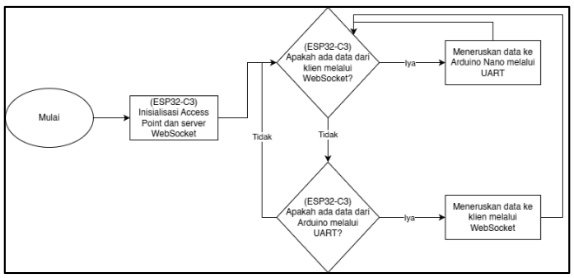
sistem dan Optocoupler untuk mengisolasi tegangan 12V masuk ke sistem 5V. Sensor gaya tekan (FSR) dipasang pada tuas persneling agar Arduino Nano dapat mengetahui ketika pengguna melakukan perpindahan gigi.

Tabel 1. Daftar perangkat keras utama

No	Nama Perangkat Keras	Gambar
1	Arduino Nano	
2	ESP32-C3	
3	MOSFET P-Channel (IRF9530N)	
4	Optocoupler (PC817)	
5	DC-to-DC Stepdown (MP1584EN)	
6	Sensor Tuas Persneling (FSR)	

Perancangan Perangkat Lunak

Perangkat lunak ditulis menggunakan bahasa pemrograman C++ dengan *library* milik Arduino IDE untuk mempermudah penulisan kode. ESP32-C3 diprogram untuk meneruskan segala jenis data dari kedua belah pihak. Ketika ESP32-C3 mendeteksi ada data dari klien melalui WebSocket, ESP32-C3 akan langsung meneruskan pesan tersebut menuju Arduino Nano melalui jalur UART. Sebaliknya, jika ESP32-C3 mendeteksi data datang dari UART Arduino Nano, data tersebut akan langsung diteruskan ke klien melalui WebSocket. UART dikonfigurasi pada kecepatan 115200 untuk mempercepat proses komunikasi data antar mikrokontroler. Melalui perangkat lunak, ESP32-C3 juga diprogram agar memancarkan SSID (bekerja sebagai *Access Point*) sekaligus WebSocket *server* sehingga tidak memerlukan *Access Point* tambahan. Diagram alur dari program ESP32-C3 dapat dilihat pada Gambar 1



Gambar 1. Diagram alur kerja sistem komunikasi ESP32-C3

Skenario Pengujian

Pengujian berfokus pada performa komunikasi nirkabel WebSocket. Untuk mempermudah analisis hasil pengujian, dilakukan pula uji menggunakan kabel USB melalui protokol serial sebagai data perbandingan performa komunikasi. Pengujian dilakukan sebanyak 30 kali sebagai *baseline* evaluasi kestabilan komunikasi *real-time*, jumlah yang umum digunakan untuk memperoleh gambaran awal performa sistem secara konsisten. Hasil pengujian ini bersifat indikatif dan eksploratif untuk menilai kelayakan implementasi WebSocket pada sistem quickshifter. Pengujian dilakukan dengan prosedur sebagai berikut:

1. Komputer klien dihubungkan dengan sistem *Quickshifter* secara nirkabel menggunakan Wi-Fi dengan protokol WebSocket. Secara bergantian, komputer klien dihubungkan ke sistem *Quickshifter* menggunakan kabel dengan protokol serial.
2. Menyalakan mesin sepeda motor untuk mendapatkan bacaan RPM sebagai data *stream*.
3. Mengirim permintaan data EEPROM ('h') berulang sebanyak 30 kali dan mencatat waktu respons
4. Melakukan batch test memakai Python untuk memuat, memperbarui, dan memverifikasi EEPROM
5. Melakukan streaming data RPM dan mencatat kestabilan koneksi
6. Melakukan pembacaan data EEPROM pada aplikasi dan mencatat hasilnya

Parameter Uji

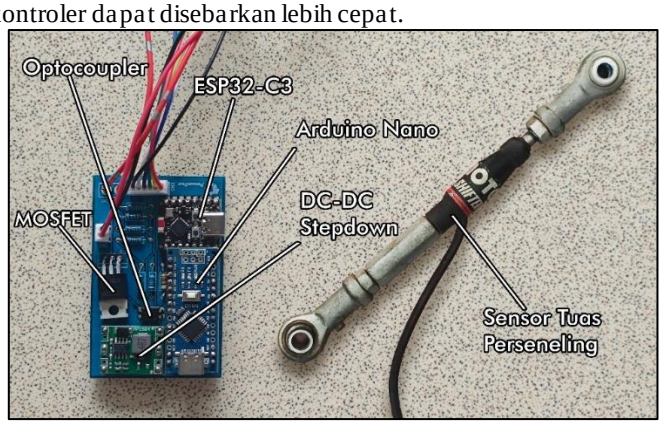
Untuk mendapatkan hasil yang valid, diperlukan penentuan alat dan parameter uji. Alat yang digunakan pada pengujian melibatkan modul *quickshifter*, komputer Windows, aplikasi Windows, kode Python dan Serial Monitor. Parameter yang akan dijadikan acuan hasil berupa:

1. *Round-trip Time* (RTT): Waktu dari saat pengirim mengirim paket sampai menerima respon.
2. *Jitter*: Variasi latensi antar paket.
3. *Packet Loss*: Persentase paket yang rusak atau hilang.
4. Kepraktisan Penggunaan: Perbandingan kemudahan pemakaian antara Wi-Fi dan USB.

Hasil Dan Pembahasan

Implementasi Prototipe

Sistem diimplementasikan pada PCB *double-layer* berukuran 46mm x 71mm dengan komponen *through-hole* untuk mempermudah perakitan. PCB didesain menggunakan *Ground Plane* untuk mengurangi dampak dari interferensi elektromagnetik (EMI) [19] hasil dari percikan busi [20]. Selain itu, dengan menggunakan *Ground Plane*, panas dari mikrokontroler dapat disebarkan lebih cepat.



Gambar 2. Implementasi sistem selesai rakit

Analisis Statistik Latensi dan Stabilitas

Pengujian pertama memiliki tujuan untuk mengukur stabilitas koneksi dasar menggunakan perintah permintaan data yang berupa char 'h' secara berulang sebanyak 30 kali (*continuous loop*) memakai kode Python. Setelah kode selesai dijalankan, muncul rangkuman mengenai pengujian yang telah dilakukan. Hasil pengujian tersebut dirangkum pada Tabel 2.

Tabel 2. Hasil pengujian berulang WebSocket

No.	Parameter Statistik	Hasil Pengukuran	Satuan	Keterangan
1	Jumlah Sampel Paket	30	Paket	*Continuous Loop*
2	Tingkat Keberhasilan	100	%	Sempurna
3	*Packet Loss*	0.00	%	Sangat Baik

4	Latensi Minimum	17.68	ms	Respons tercepat
5	Latensi Maksimum	30.19	ms	Respons terlambat
6	Rata-rata Latensi	19.81	ms	Sangat Baik
7	*Jitter*	1.79	ms	Sangat stabil

Mengacu pada Tabel 2, terlihat bahwa protokol WebSocket memberikan performa yang sangat stabil dengan rata-rata latensi 19.81ms. Variasi nilai latensi dirata-rata dan didapati nilai *jitter* sebesar 1.79ms, sangat minim variasi waktu tunda antar paket yang sangat krusial agar data RPM dapat tampil di layer klien sinkron dengan kondisi aktual mesin. WebSocket yang didasari oleh TCP berhasil menjaga integritas data pada media nirkabel dengan tingkat keberhasilan 100%.

Analisis statistik pada penelitian ini bersifat deskriptif non-inferensial, dengan fokus pada nilai rata-rata, latensi minimum–maksimum, dan *jitter* untuk menggambarkan karakteristik performa komunikasi *real-time* tanpa melakukan pengujian hipotesis statistik.

Analisis Respon Fungsional

Pada pengujian kedua, memakai metode *Batch Test*, yaitu pengujian menggunakan banyak perintah secara sekuensial untuk memahami karakteristik koneksi WebSocket dan perilaku sistem dalam menangani perintah yang berbeda sembari menguji fungsional konfigurasi *Quickshifter*.

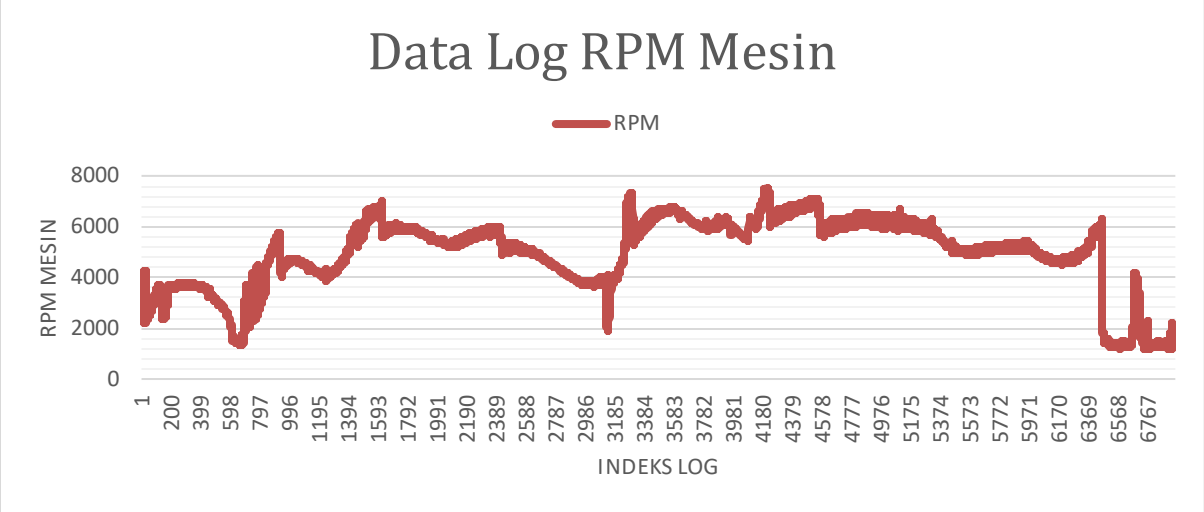
Tabel 3. Hasil pengujian fungsional WebSocket

No.	Jenis Perintah	Perintah	Respon Sistem	Parameter Ukur	Hasil Uji	Status
1	Muat EEPROM (h)	h	JSON Data	Latensi (RTT)	84.61 ms	Berhasil
2	Perbarui EEPROM *killtime* (u)	u 0 85	OK_MAP	Latensi (RTT)	10.75 ms	Berhasil
3	Perbarui EEPROM RPM Aktif(u)	u 16 4500	OK_ACT	Latensi (RTT)	10.74 ms	Berhasil
4	*Streaming* RPM	(auto)	Data RPM	*Jitter*	27.3 ms (avg)	Stabil
5	Verifikasi Data	h	JSON Terupdate	Latensi (RTT)	141.87 ms	Valid

Pada data yang terdapat pada Tabel 3, tercatat sebuah perintah memiliki waktu respons yang cukup tinggi dengan nilai mencapai 84.61ms. Hal ini disebabkan oleh kondisi *Cold Start*, kondisi saat sistem memerlukan waktu untuk melakukan inisiasi alokasi memori dan pemrosesan awal setelah beberapa saat tidak ada data yang lewat. Tetapi, saat sistem sudah dalam kondisi *Warm State* (koneksi sudah aktif dan *buffer* siap), latensi langsung turun drastis dibandingkan perintah sebelumnya. Tercatat perintah berikutnya dieksekusi dan dilaporkan balik dengan waktu 10.75ms. Membuktikan WebSocket memiliki model koneksi *persistent* yang tidak memerlukan *handshake* baru untuk setiap pengiriman data. Latensi kembali naik saat proses verifikasi data dieksekusi. Hal ini sepenuhnya disebabkan oleh Arduino Nano yang mengalami "*hiccup*" karena melakukan pemrosesan JSON dan *streaming* RPM secara bersamaan.

Analisis Kinerja Streaming Data RPM

Pengujian kemampuan transaksi data WebSocket secara berkelanjutan dilakukan lagi menggunakan *streaming* data RPM. Setiap tonjolan pada *flywheel* terdeteksi oleh Arduino akan dikirimkan data RPM menuju ESP32-C3 yang kemudian diteruskan melalui WebSocket. Data *streaming* RPM direkam menggunakan kode Python dengan luaran *log* berupa file .txt. Hasil data log RPM divisualisasikan pada Gambar 3.



Gambar 3. Visualisasi data log RPM

Data uji *streaming* RPM dapat direkam dengan jelas tanpa ada kerusakan. Kemudian data *log* RPM tersebut diproses dan didapatkan nilai rata-rata *jitter* sebesar 27.3ms. Perlu diketahui bahwa Arduino Nano mengirim data RPM setiap tonjolan terdeteksi pada *flywheel* sehingga secara langsung *jitter* juga dipengaruhi oleh rata-rata posisi RPM ketika *streaming* terjadi. Walaupun sudah menggunakan rata-rata *jitter* yang "terpengaruh" oleh RPM, nilai *jitter* masih sangat layak untuk visualisasi pada layer klien.

Perbandingan dengan USB Serial

Sebagai perbandingan, pengujian yang sama dilakukan pada komunikasi berkabel. Data yang didapat bisa digunakan untuk mengukur *trade-off* antara kenyamanan koneksi nirkabel dan stabilitas performa Didapatkan hasil seperti pada Tabel 4.

Tabel 4. Perbandingan performa WebSocket dan Serial USB

No.	Parameter Pengujian	WebSocket	USB Serial	Selisih Performa WebSocket dengan USB
1	Latensi Minimal	17.68ms	15.78ms	+1.90ms
2	Latensi Maksimal	30.19ms	18.70ms	+11.49ms
3	Latensi Rata-rata	19.81ms	16.83ms	+2.98ms
4	Jitter Rata-rata	1.79ms	0.65ms	+1.14ms

Berdasarkan pada Tabel 4, terlihat jelas bahwa koneksi berkabel mengungguli koneksi nirkabel WebSocket pada aspek kecepatan dan stabilitas. USB Serial mencatatkan latensi rata-rata 16.83ms dengan tingkat kestabilan sinyal (jitter) yang sangat presisi di angka 0.65ms. Sedangkan koneksi WebSocket mendapat rata-rata latensi 2.98ms lebih tinggi dari USB Serial, begitu pula untuk rata-rata *jitter* dengan perbedaan 1.14ms lebih tinggi. Perbedaan paling mencolok terlihat pada Latensi Maksimal, sebesar 30.19ms pada koneksi WebSocket. Lonjakan pada nilai latensi maksimal dapat diwajarkan dalam kasus komunikasi nirkabel karena dapat dipengaruhi oleh berbagai macam aspek seperti mekanisme enkripsi, modulasi sinyal udara, serta interferensi lingkungan.

Selisih latensi rata-rata sebesar 2–3 ms antara WebSocket dan USB Serial tidak memberikan dampak fungsional yang signifikan terhadap kinerja modul *quickshifter*. Sistem mekanik perpindahan gigi dan proses pembakaran mesin memiliki toleransi waktu yang jauh lebih besar dibandingkan selisih tersebut. Dengan demikian, perbedaan performa komunikasi ini tidak memengaruhi kehalusan perpindahan gigi maupun keamanan operasi, sehingga WebSocket tetap layak digunakan untuk keperluan penyetelan dan pemantauan *quickshifter* secara *real-time*.

Meskipun pada WebSocket terdapat "degradasi" performa jika dibandingkan dengan USB Serial, selisih nilai tersebut masih jauh di bawah nilai toleransi manusia merasakan perbedaan. Dalam konteks penyetelan mesin, respons pada sisi klien tetap terasa instan tanpa ada perubahan signifikan secara perseptual. Sedangkan pengguna mendapatkan keuntungan besar berupa fleksibilitas koneksi nirkabel dengan mengorbankan sangat sedikit performa yang dapat dikatakan marginal.

Dari sudut pandang desain sistem *embedded real-time*, hasil penelitian ini menunjukkan adanya *trade-off* antara fleksibilitas koneksi nirkabel dan determinisme waktu. Komunikasi berbasis WebSocket menawarkan kemudahan konfigurasi, mobilitas, dan skalabilitas sistem, namun memiliki variabilitas latensi yang lebih tinggi dibandingkan koneksi berkabel. Meskipun demikian, selama nilai latensi dan *jitter* masih berada dalam batas

toleransi sistem mekanik mesin, pendekatan ini dapat diterima dan memberikan keuntungan praktis yang signifikan dalam pengembangan sistem otomotif modern berbasis IoT.

Namun, penelitian ini memiliki beberapa keterbatasan. Pengujian hanya dilakukan pada satu konfigurasi perangkat keras dan satu lingkungan jaringan Wi-Fi, sehingga variasi kondisi jaringan belum sepenuhnya terwakili. Selain itu, pengujian belum mencakup kondisi interferensi berat, jarak komunikasi yang ekstrem, maupun pengujian berkelanjutan pada kecepatan kendaraan dan RPM yang sangat tinggi. Oleh karena itu, hasil penelitian ini masih bersifat awal dan perlu divalidasi lebih lanjut pada skenario yang lebih kompleks.

Penelitian selanjutnya dapat diarahkan pada perbandingan performa WebSocket dengan protokol komunikasi lain seperti MQTT atau ESP-NOW, serta penerapan mekanisme *Quality of Service* (QoS) dan penjadwalan *real-time* pada sistem *embedded*. Selain itu, integrasi sistem dengan *cloud logging* atau *dashboard* IoT juga dapat dikembangkan untuk mendukung analisis data jangka panjang dan pemantauan sistem *quickshifter* secara lebih komprehensif.

Simpulan

Penelitian ini berhasil merealisasikan sistem *quickshifter* dengan arsitektur dual-mikrokontroler yang memisahkan tugas pengolahan utama mesin dan komunikasi nirkabel secara efektif, dengan Arduino Nano menangani pemrosesan sensor dan pemutusan pengapian, sementara ESP32-C3 berperan sebagai *gateway* komunikasi berbasis WebSocket. Implementasi WebSocket terbukti mampu menyediakan komunikasi *real-time* yang stabil dengan latensi rata-rata 19,81ms dan *jitter* 1,79ms tanpa *packet loss*, meskipun secara performa masih sedikit di bawah koneksi USB Serial. Namun, selisih performa tersebut tidak berdampak signifikan terhadap persepsi respons sistem, sehingga penyetelan parameter dan pemantauan RPM tetap berjalan secara responsif. Dengan demikian, penggunaan WebSocket pada sistem *quickshifter* memberikan kompromi yang seimbang antara keandalan performa *real-time* dan fleksibilitas koneksi nirkabel, sekaligus meningkatkan kepraktisan operasional tanpa mengorbankan keselamatan dan stabilitas kerja sistem.

Daftar Pustaka

[1] Y. B. Zikria, R. Ali, M. K. Afzal, dan S. W. Kim, "Next-Generation Internet of Things (IoT): Opportunities, Challenges, and Solutions," *Sensors*, vol. 21, no. 4, hlm. 1174, Feb 2021, doi: 10.3390/s21041174.

[2] J. Y. Goh dan D. W. Chua, "A comparative analysis of wired vs. wireless data communication technologies".

[3] D. Eridani, A. F. Rochim, dan F. N. Cesara, "Comparative Performance Study of ESP-NOW, Wi-Fi, Bluetooth Protocols based on Range, Transmission Speed, Latency, Energy Usage and Barrier Resistance," dalam *2021 International Seminar on Application for Technology of Information and Communication (iSemantic)*, Semarangin, Indonesia: IEEE, Sep 2021, hlm. 322–328. doi: 10.1109/iSemantic52711.2021.9573246.

[4] N. Mitrovic, M. Eordevic, S. Veljkovic, dan D. Dankovic, "Implementation of WebSockets in ESP32 based IoT Systems," dalam *2021 15th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS)*, Nis, Serbia: IEEE, Okt 2021, hlm. 261–264. doi: 10.1109/TELSIKS52058.2021.9606244.

[5] V. Pimentel dan B. G. Nickerson, "Communicating and Displaying Real-Time Data with WebSocket," *IEEE Internet Comput.*, vol. 16, no. 4, hlm. 45–53, Jul 2012, doi: 10.1109/MIC.2012.64.

[6] K. A. Nugraha, "Efisiensi Pertukaran Data Client-Server menggunakan Web Socket pada Perangkat Berbasis Internet of Things," *J. Edukasi Dan Penelit. Inform. JEPIN*, vol. 10, no. 1, hlm. 33, Apr 2024, doi: 10.26418/jp.v10i1.73145.

[7] M. Stanglmayr dan G. Prokop, "Method for modeling of motorcycle shifting operations for test rig applications," *Proc. Inst. Mech. Eng. Part J. Automob. Eng.*, hlm. 09544070241306818, Jan 2025, doi: 10.1177/09544070241306818.

[8] D. Giovanelli dan E. Farella, "Force Sensing Resistor and Evaluation of Technology for Wearable Body Pressure Sensing," *J. Sens.*, vol. 2016, hlm. 1–13, 2016, doi: 10.1155/2016/9391850.

[9] R. A. Himawan dan N. I. Prasetya, "Pembuatan Modul Quickshifter Menggunakan Arduino Uno Sebagai Pengganti Kopling Motor".

[10] P. W. Rusimamto, E. Endryansyah, L. Anifah, R. Harimurti, dan Y. Anistyasari, "Implementation of Arduino Pro Mini and ESP32 cam for temperature monitoring on automatic thermogun IoT-based," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 23, no. 3, hlm. 1366, Sep 2021, doi: 10.11591/ijeecs.v23.i3.pp1366-1375.

[11] A. D. Hassebo, K. B. Montes, dan E. Cabrera, "Arduino-ESP32 based Smart Irrigation System".

[12] J. Li, D. Li, dan Y. Xu, "Design of Dual MCU Vehicle Control Unit Based on Electric Vehicles to Respond to Control Failure," *IOP Conf. Ser. Earth Environ. Sci.*, vol. 512, no. 1, hlm. 012132, Jun 2020, doi: 10.1088/1755-1315/512/1/012132.
[13] E. Montanez dan A. Mastronardi, "Microcontrollers In Education: Embedded Control – Everywhere And Everyday," dalam *2005 Annual Conference Proceedings*, Portland, Oregon: ASEE Conferences, Jun 2005, hlm. 10.938.1-10.938.10. doi: 10.18260/1-2--14812.
[14] "Nano | Arduino Documentation." Diakses: 21 September 2025. [Daring]. Tersedia pada: https://docs.arduino.cc/hardware/nano/#tech-specs
[15] RayMing, "ESP32-WROOM vs. ESP32-C3: Key Differences and Best Use Cases," RayPCB. Diakses: 21 September 2025. [Daring]. Tersedia pada: https://www.raypcb.com/esp32-wroom-vs-esp32-c3/
[16] M. Daraban, C. Corches, A. Taut, dan G. Chindris, "Protocol over UART for Real-Time Applications," dalam *2021 IEEE 27th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, Timisoara, Romania: IEEE, Okt 2021, hlm. 85–88. doi: 10.1109/SIITME53254.2021.9663605.
[17] "MOSFET Types, Working, Structure, and Applications." Diakses: 25 September 2025. [Daring]. Tersedia pada: https://www.electronicsforu.com/technology-trends/learn-electronics/mosfet-basics-working-applications
[18] P. Teuwen dan C. Herrmann, "EEPROM: It Will All End in Tears".
[19] Peng Zhang dan Shufang Li, "The relationship between Ground and EMI," dalam *2005 IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications*, Beijing, China: IEEE, 2005, hlm. 662–665. doi: 10.1109/MAPE.2005.1617997.
[20] C. Marczok, U. Maass, E. Hoene, I. Ndip, K.-D. Lang, dan D. Hasselberg, "Analysis and improvement of a spark plug for less radiated electromagnetic emissions," dalam *2014 International Symposium on Electromagnetic Compatibility*, Gothenburg: IEEE, Sep 2014, hlm. 385–390. doi: 10.1109/EMCEurope.2014.6930937.