Jurnal Minfo Polgan (JMP)
Volume 15, Nomor 2, Mei-Agustus 2026
DOI : https://doi.org/10.33395/jmp.v15i2.16308
e-ISSN : 2797-3298
p-ISSN : 2089-9424

Implementasi Validasi Transaksi ACID dan Protokol Hybrid pada Orkestrasi Multi-Robot di SmartEDU Station Warehouse

[1*]Yuliadi Erdani, [2]Ahshonat Khoerunnisa, [3]Aqsha Maulana Ilham, [4]Sarosa Castrena Abadi, [5]Abyannudin Salam, [6]Adhitya Sumardi Sunarya
[1*,2,3,4,6]Automation Engineering, Politeknik Manufaktur Bandung, Bandung, Indonesia
[5]School of Science Computing Emerging Technologies, Swinburne University of Technology, Australia
*Korespondensi: yul_erdani@yahoo.com

Submit : 03 Jun 2026 | **Diterima** : 24 Jun 2026 | **Terbit** : 29 Jun 2026

ABSTRACT

The implementation of Cyber-Physical Systems (CPS) in smart warehouses is often constrained by data integrity issues and communication protocol incompatibilities between the Information Technology (IT) and Operational Technology (OT) layers. This study focuses on the implementation of ACID (Atomicity, Consistency, Isolation, Durability)-based transaction validation and hybrid protocol integration to orchestrate Autonomous Mobile Robots (AMR) and Articulated Robotic Arms (ARM) within the SmartEDU Station Warehouse environment. The backend architecture was built using Express.js as the ACID validation gate, while Node-RED acts as the middleware bridging the hybrid communication, utilizing REST API for AMR telemetry and Modbus TCP for ARM control. Testing was conducted over 50 full operational cycles in real-time. The results indicate that the backend layer achieved a 100% success rate in preventing race conditions and data inconsistencies through an automatic rollback mechanism. The performance of the hybrid protocol was measured to be highly efficient, with an average Modbus TCP write latency of 14–25 ms and an optimal REST API polling interval of 2000 ms, which generated a CPU load of only 0.2–4.1%. The middleware response time proved to be stable within the 191–196 ms range across all scenarios. This confirms that the application of ACID validation in this decentralized architecture does not burden the system's latency performance while simultaneously ensuring the security of IT-OT data synchronization.

***Keywords**: ACID Transaction; Heterogeneous Robot Orchestration; Modbus TCP; Node-RED; REST API; Smart Warehouse*

ABSTRAK

Implementasi *Cyber-Physical System* (CPS) pada *smart warehouse* sering kali terkendala oleh masalah integritas data dan ketidaksesuaian protokol komunikasi antara lapisan Teknologi Informasi (IT) dan Teknologi Operasional (OT). Penelitian ini berfokus pada implementasi validasi transaksi berbasis ACID (*Atomicity, Consistency, Isolation, Durability*) dan integrasi protokol *hybrid* untuk mengorkestrasi *Autonomous Mobile Robot* (AMR) dan *Articulated Robotic Arm* (ARM) di lingkungan SmartEDU Station Warehouse. Arsitektur *backend* dibangun menggunakan Express.js sebagai gerbang validasi ACID, sementara Node-RED bertindak sebagai *middleware* yang menjembatani komunikasi *hybrid*, yaitu *REST API* untuk telemetri AMR dan Modbus TCP untuk kontrol ARM. Pengujian dilakukan sebanyak 50 siklus operasional penuh secara *real-time*. Hasil penelitian menunjukkan bahwa lapisan *backend* mencapai tingkat keberhasilan (*success rate*) 100% dalam mencegah *race condition* dan inkonsistensi data melalui mekanisme *rollback* otomatis. Performa protokol *hybrid* terukur sangat efisien, dengan latensi penulisan Modbus TCP rata-rata 14–25 ms dan interval optimal *REST API* pada 2000 ms yang hanya menghasilkan *CPU load* sebesar 0,2–4,1%. Waktu respons *middleware* terbukti stabil pada rentang 191–196 ms di setiap skenario pengujian. Hal ini mengonfirmasi bahwa penerapan validasi ACID pada arsitektur terdesentralisasi ini tidak membebani performa latensi sistem, sekaligus menjamin keamanan sinkronisasi data IT-OT.

Kata Kunci: ACID Transaction; Modbus TCP; Node-RED; Orkestrasi Robot Heterogen; REST API; Smart Warehouse.

PENDAHULUAN

Perkembangan teknologi pada era Industry 4.0 telah mendorong transformasi digital di berbagai sektor, termasuk manajemen logistik dan *Smart Warehouse* (Alam, Ugli, Tanha, & Jun, 2026). Meningkatnya kompleksitas rantai pasok global dan pertumbuhan *e-commerce* mendorong industri mengadopsi otomatisasi gudang yang lebih canggih, dengan digitalisasi proses internal sebagai faktor penentu efektivitas aliran barang (Tubis & Rohman, 2023). Konsep *Cyber-Physical System* (CPS) yang mengintegrasikan model virtual dengan sistem fisik nyata menjadi salah satu fondasi utama dalam mewujudkan otomatisasi manufaktur cerdas yang adaptif dan terhubung secara digital (Harja, Setiawan, Erdani, & Febriansyah, 2022). Dalam ekosistem gudang cerdas, penggunaan berbagai perangkat robotik secara bersamaan menjadi keharusan strategis, seperti *Autonomous Mobile Robot* (AMR) untuk navigasi spasial dan logistik internal (Lackner, Hermann, Kuhn, & Palm, 2024; Loganathan & Ahmad, 2023), serta lengan robot (*Articulated Robotic Arm* atau ARM) untuk manipulasi barang tingkat lanjut (Chen et al., 2026). Robot industri dirancang untuk pekerjaan berulang dalam lingkungan yang terorganisir dan dapat diprediksi (Zafar, Langås, & Sanfilippo, 2024), namun integrasi perangkat robot heterogen di lantai produksi memunculkan tantangan teknis yang kompleks.

Kajian terhadap kondisi implementasi sistem robotik gudang saat ini mengidentifikasi tiga kesenjangan utama. Pertama, *The Language Gap* antara lapisan IT dan OT: telemetri AMR berbasis REST API/JSON berbenturan dengan kontrol ARM berbasis Modbus TCP yang deterministik (Calderón, Folgado, González, & Calderón, 2024; Real Time Automation, n.d.). Konversi antara format data register dengan JSON memerlukan lapisan perantara dengan latensi rendah (Găitan, Zagan, & Găitan, 2024; Zagan & Găitan, 2022). Kedua, *The Coordination Gap*: AMR dan ARM beroperasi tanpa sinkronisasi status operasional real-time, sehingga sistem rentan terhadap *race condition*. Meskipun alokasi tugas multi-robot heterogen telah diteliti (Li, Chen, Wang, & Kan, 2025), pendekatan yang ada belum menjembatani kesenjangan protokol IT-OT. Ketiga, *The Integrity Gap*: belum ada lapisan logika yang menjamin prinsip asiditas (ACID: *Atomicity, Consistency, Isolation, Durability*) pada basis data sebelum instruksi dikirim ke lantai produksi (Oñate & Sanz, 2024), Tanpa validasi ini, setiap kegagalan komunikasi atau gangguan robot di lantai produksi akan membuat data transaksi IT menjadi tidak sinkron, sehingga informasi stok di basis data inventori gudang menjadi kacau dan tidak mencerminkan kondisi fisik yang sebenarnya.

Untuk menjawab ketiga kesenjangan tersebut, penelitian ini mengusulkan arsitektur prototipe orkestrasi robot heterogen berbasis Node-RED sebagai *middleware* sentral yang menjembatani komunikasi antara lantai produksi dengan infrastruktur *backend* (Node-RED Project, 2024; Oñate & Sanz, 2024). Implementasi dan pengujian arsitektur ini dilakukan pada platform SmartEDU Station Warehouse, sebuah sel kerja otomasi terpadu yang dikembangkan sebagai sarana pembelajaran dan pelatihan otomasi industri yang mampu mengintegrasikan komponen heterogen termasuk PLC, mikrokontroler, dan protokol IoT (Ridwan et al., 2025). Kemampuan Node-RED telah divalidasi dalam pemantauan PLC industri (A. H. Embong, Asbollah, & Abdul Hamid, 2024) serta dikonfirmasi sebagai strategi tepat untuk orkestrasi antar-subsistem manufaktur cerdas (Ochoa, Larrinaga, & Pérez, 2023). Solusi terdiri dari tiga komponen: pertama jalur komunikasi *hybrid* (REST API untuk telemetri AMR dan Modbus TCP untuk kontrol ARM); kedua logika *Task Queue* berbasis array dalam Node-RED sebagai orkestrator sekuensial; dan ketiga lapisan validasi transaksi *backend* berbasis Express.js yang menerapkan prinsip ACID.

Beberapa penelitian terkait telah mengkaji aspek parsial masalah ini, seperti integrasi IT-OT (Calderón et al., 2024), *interworking* Modbus IoT (Elamanov et al., 2024), konversi protokol 5G (Pan, Wu, Du, & Wang, 2023), serta optimasi akuisisi Modbus (Găitan et al., 2024; Zagan & Găitan, 2022). Pemanfaatan Node-RED dalam arsitektur RAMI 4.0 (Oñate & Sanz, 2024), implementasi *protocol manager* IIoT (Torres Ventura, Ruelas Puente, & Herrera García, 2023), dan pemantauan PLC (A. H. Embong et al., 2024) juga telah diteliti. Dari sisi koordinasi multi-robot, riset mencakup alokasi tugas (Li et al., 2025), kombinasi algoritma Hungarian-TSP (Msala, Oussama, Talea, & Aboulfatah, 2025), serta perencanaan tugas real-time (Zhang et al., 2025). Namun, belum ada penelitian yang mengintegrasikan konversi *payload* dual-protokol, mekanisme *feedback loop* berbasis *task queue* untuk meminimalkan *race condition* AMR-ARM (Li et al., 2025; Torres Ventura et al., 2023), dan jaminan ACID pada *backend* secara terpadu (Tubis & Rohman, 2023; Zagan & Găitan, 2022).

Penelitian ini bertujuan untuk merancang lapisan validasi transaksi *backend* berbasis Express.js yang menjamin prinsip ACID sebagai gerbang integritas data pra-eksekusi, mengembangkan logika orkestrasi sekuensial berbasis *Task Queue* dalam Node-RED, serta memvalidasi performa jalur komunikasi *hybrid* dalam hal latensi konversi *payload* dan keandalan *cycle time* pada prototipe sistem gudang cerdas (Alam et al., 2026; Ridwan et al., 2025; Zafar et al., 2024).

METODE PENELITIAN

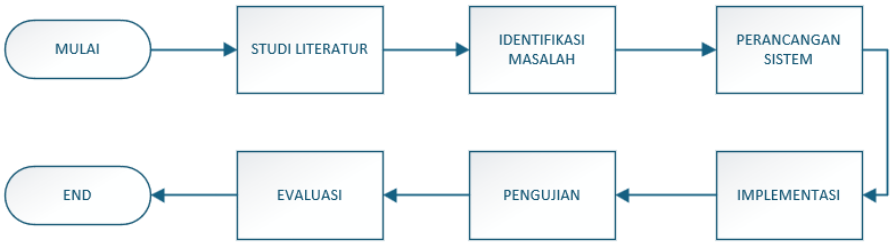
Penelitian ini menggunakan pendekatan eksperimental berbasis perancangan dan implementasi prototipe (*prototyping*). Pendekatan ini dipilih karena sebagian besar penelitian terdahulu di bidang arsitektur IIoT hanya bersifat konseptual tanpa validasi eksperimental pada kondisi operasional nyata, sehingga diperlukan implementasi dan pengujian langsung menggunakan perangkat keras dan perangkat lunak industri yang sesungguhnya (Calderón et al., 2024). Sistem dirancang, dibangun, diuji, dan dievaluasi dalam lingkungan terkontrol untuk menjawab tiga kesenjangan utama yang diidentifikasi pada pendahuluan.

Tahapan Penelitian

Penelitian dilaksanakan melalui enam tahapan sekuensial sebagaimana ditunjukkan pada Tabel 1 dan Gambar 1. Setiap tahapan menghasilkan luaran konkret yang menjadi masukan bagi tahapan berikutnya, mulai dari studi literatur hingga evaluasi akhir.

Tabel 1 Tahapan Penelitian dan Luaran

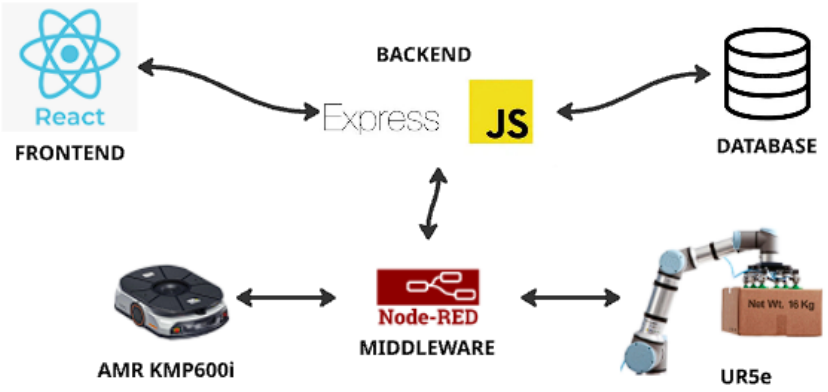
No	Tahapan	Luaran
1	Studi Literatur	Landasan Teori, Referensi Protokol & orkestrasi Robot
2	Identifikasi Masalah	Rumusan 3 Gap: Language, Coordination, Integrity
3	Perancangan Sistem	Diagram arsitektur, flowchart orkestrasi, skema ACID
4	Implementasi	Prototipe Node-RED, backend Express.js, koneksi robot
5	Pengujian	Validasi fungsional, latensi, cycle time, success rate
6	Evaluasi	Analisis hasil, perbandingan literatur, kesimpulan



Gambar 1. Flowchart Sistem

Arsitektur Sistem

Sistem terdiri dari lima lapisan yang saling terhubung, meliputi *frontend dashboard* ReactJS dan WebSocket untuk pemantauan *real-time*, *backend* Express.js sebagai gerbang validasi data dan logika bisnis, Node-RED sebagai *middleware* orkestrasi terpusat, lapisan perangkat robot berupa AMR via *REST API* dan ARM via Modbus TCP, serta basis data MySQL sebagai penyimpan inventori dan log transaksi. Arsitektur keseluruhan ditunjukkan pada Gambar 2, sedangkan spesifikasi komponen dirangkum pada Tabel 2.



Gambar 2. Arsitektur Sistem

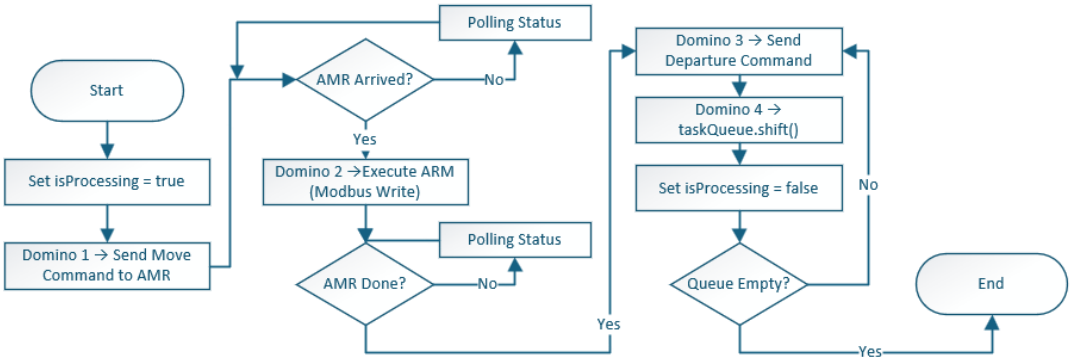
Tabel 2 Spesifikasi Perangkat Keras dan Perangkat Lunak Sistem

Kategori	Komponen	Spesifikasi
Hardware	AMR	Platform mobile, antarmuka REST API
Hardware	ARM	Lengan robot 6-DoF, kontroler Modbus TCP
Hardware	Server	Diagram arsitektur, flowchart orkestrasi, skema ACID
Software	Node-Red	node-red-contrib-modbus
Software	Backend	Express.js v4.18
Software	Database	MySQL 8.0
Protokol	AMR ↔ Node-RED	REST API HTTP/JSON
Protokol	ARM ↔ Node-RED	Modbus TCP port 502
Protokol	Frontend ↔ Backend	WebSocket real-time dashboard

Mekanisme Orkestrasi Robot

Orkestrasi diimplementasikan menggunakan struktur data *array* JavaScript dalam memori global Node-RED sebagai *task queue* berbasis FIFO (*First In, First Out*). Tugas baru ditambahkan ke akhir antrean via metode *.push()*, sementara tugas yang selesai dihapus dari posisi pertama via metode *.shift()*, secara otomatis memindahkan tugas berikutnya ke eksekusi.

Setiap siklus tugas berjalan melalui empat fase sekuensial (mekanisme domino): AMR bergerak ke stasiun → ARM mengeksekusi operasi → AMR bergerak ke tujuan → tugas selesai dan *.shift()* dipanggil. Untuk mencegah *race condition*, variabel globalContext.*isProcessing* = *true* mengunci siklus aktif dari instruksi konkuren hingga seluruh fase tuntas. Alur lengkap ditunjukkan pada Gambar 3.

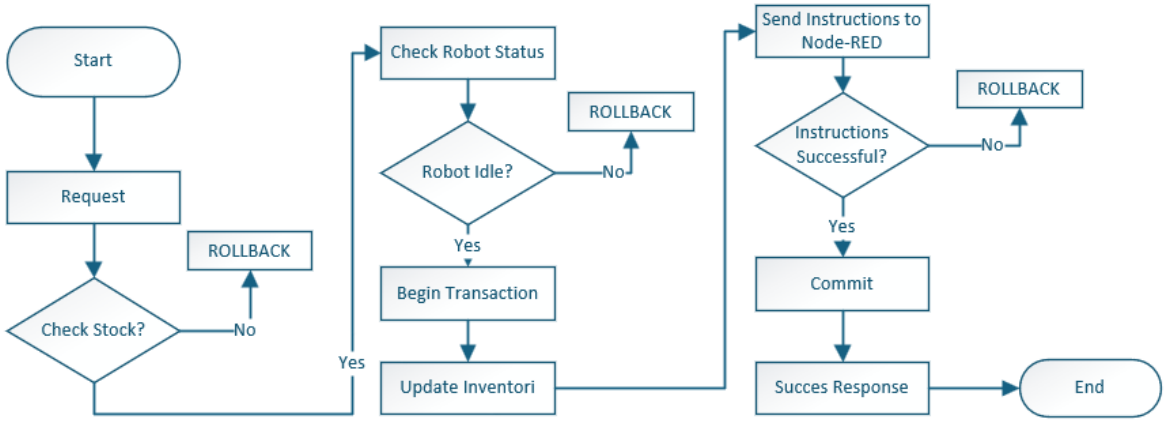


Gambar 3. Flowchart Mekanisme Orkestrasi

Implementasi Middleware Node-RED dan Validasi ACID Backend

Node-RED menggunakan kombinasi node *http request* untuk telemetri AMR, node *modbus-write/read* untuk kontrol ARM, node *function* untuk parsing JSON dan logika task queue, serta node *websocket out* untuk pembaruan status ke dashboard. Konversi payload dari objek JSON

ke nilai integer 16-bit Holding Register Modbus dilakukan pada node *function* menggunakan persamaan linier: RegisterValue = (ParameterValue / MaxValue) × 65535
Backend Express.js menerapkan pola transaksi BEGIN – COMMIT – ROLLBACK pada setiap permintaan tugas. Sebelum instruksi diteruskan ke Node-RED, backend memvalidasi ketersediaan stok dan status kesiapan robot. Jika validasi gagal—akibat stok habis, robot aktif, atau gangguan komunikasi jaringan seluruh perubahan data dibatalkan melalui ROLLBACK, menjamin konsistensi antara data inventori di lapisan IT dengan kondisi fisik robot di lapisan OT. Pemetaan prinsip ACID ke implementasi dirangkum sebagai berikut: *Atomicity* (semua atau tidak ada perubahan), *Consistency* (validasi stok & status sebelum COMMIT), *Isolation* (row-level lock mencegah akses konkuren), dan *Durability* (data tersimpan permanen pasca COMMIT).



Gambar 4. Flowchart Validasi Transaksi ACID

Skenario Pengujian

Pengujian sistem dilakukan dalam lingkungan prototipe terkontrol yang terdiri dari satu unit AMR, satu unit ARM, empat pos stasiun kerja, dan satu server host yang menjalankan seluruh komponen perangkat lunak secara bersamaan. Seluruh pengujian dilaksanakan sebanyak 50 siklus kerja penuh. Lima skenario pengujian beserta parameter dan metode validasi masing-masing dirangkum pada Tabel 3.

Tabel 3 Skenario Pengujian Sistem

Skenario Pengujian	Parameter Ukur	Metode Validasi
Konversi Payload JSON → Modbus	Latency konversi (ms)	Timestamp Node-RED
Orkestrasi Sekuensial 4 Fase	Cycle time, success rate	50× siklus penuh
Validasi Transaksi ACID	Rollback pada kondisi gagal	Injeksi error artifisial
Pencegahan Race Condition	Deteksi eksekusi bersamaan	Pengiriman instruksi konkuren
Feedback Loop Real-Time	Delay respons ARM → AMR (ms)	Interval polling Modbus

Metrik utama yang dikumpulkan meliputi latensi konversi payload (ms), cycle time end-to-end (s), success rate orkestrasi (%), perilaku ROLLBACK pada kondisi gagal, serta delay respons feedback loop antara ARM dan AMR (ms). Seluruh data hasil pengujian dianalisis dan dibahas pada Bab III.

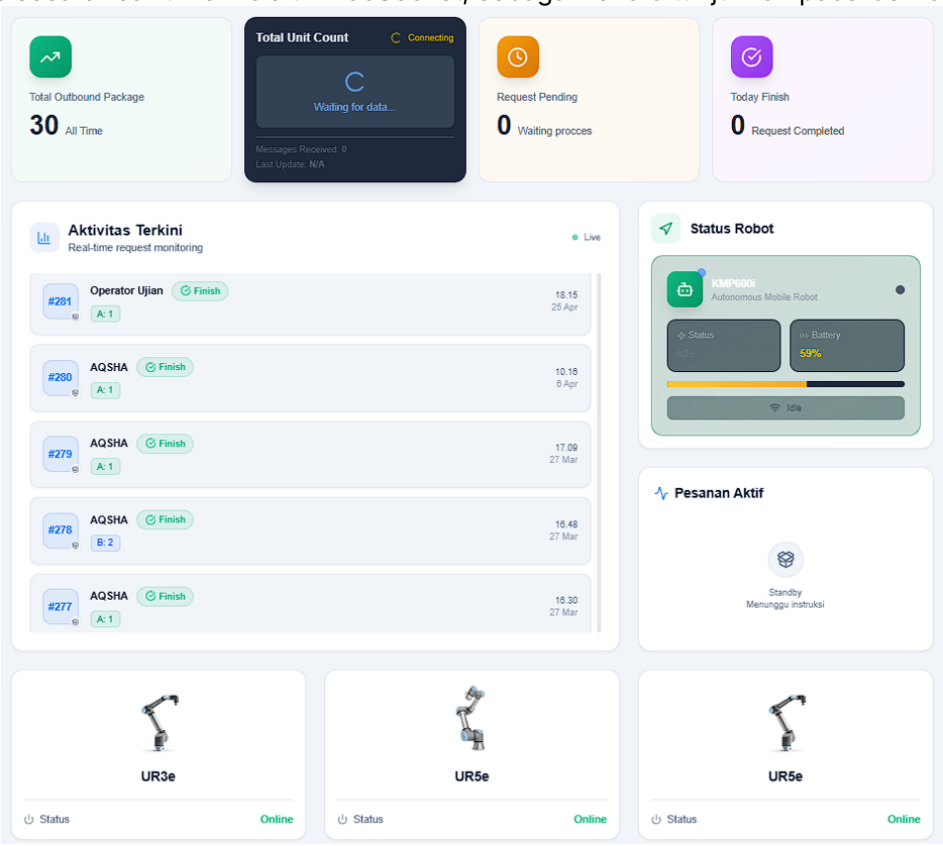
HASIL DAN PEMBAHASAN

Bagian ini memaparkan hasil pengujian sistem orkestrasi robot heterogen berdasarkan 50 siklus kerja dalam lingkungan prototipe terkontrol, mencakup analisis integritas data, performa komunikasi hybrid, keandalan orkestrasi, cycle time, dan perbandingan dengan penelitian sebelumnya.

Implementasi Prototipe Sistem

Prototipe sistem berhasil diimplementasikan dengan Node-RED sebagai *middleware* orkestrasi terpusat, Express.js sebagai *backend* validasi data, dan MySQL sebagai basis data

inventori. Sistem mendukung komunikasi *hybrid* melalui *REST API* untuk telemetri AMR dan Modbus TCP untuk kontrol ARM. *Frontend dashboard* berhasil menampilkan status robot dan antrean tugas secara *real-time* melalui WebSocket, sebagaimana ditunjukkan pada Gambar 5



Gambar 5. Antarmuka Frontend Dashboard Sistem Orkestrasi Robot

Gambar 5 menampilkan antarmuka *frontend dashboard* yang merepresentasikan integrasi data antara lapisan IT dan OT. Panel Aktivitas Terkini menunjukkan log eksekusi tugas secara kronologis (tugas #267 hingga #271) dengan status *Finish*, yang mengonfirmasi bahwa gerbang validasi *backend* berhasil mengamankan riwayat transaksi tanpa kehilangan data. Panel Status Robot dan Pesanan Aktif memperlihatkan telemetri AMR dalam kondisi *Idle* dan *task queue* dalam Node-RED yang siap menerima instruksi baru. Keberhasilan sinkronisasi ini membuktikan bahwa arsitektur komunikasi WebSocket mampu menjembatani pertukaran informasi dari perangkat fisik ke lapisan operator tanpa jeda komputasi yang berarti.

Pengujian Integritas Data (Validasi ACID)

Pengujian dilakukan sebanyak 10 kali pada setiap skenario untuk memastikan konsistensi perilaku backend dalam menangani kondisi data yang bervariasi. Hasil pengujian disajikan pada Tabel 4.

Tabel 4 Hasil Validasi ACID

Skenario	Kondisi Input	Ekspektasi	Hasil
Stok Tersedia	Item_A, Qty: 5	COMMIT & teruskan ke Node-RED	Transaction Success
Stok Habis	Item_B, Qty: 0	ROLLBACK & kembalikan error	Transaction Aborted
Robot Sedang Aktif	Item_C, Qty: 10	ROLLBACK & blokir antrean	Request Rejected
Gangguan Jaringan mid - Tx	Item_D, Qty: 3	ROLLBACK otomatis saat timeout	Auto Rollback
Dua permintaan Konkuren	Item_A & Item_E	Row-level lock, satu ditolak	Isolation Enforced

Penjelasan lebih mendalam terhadap Tabel 4 menunjukkan pentingnya isolasi (*Isolation*) dan konsistensi (*Consistency*) pada sistem basis data sebelum instruksi diteruskan ke lantai produksi. Pada skenario 'Stok Habis' dan 'Robot Sedang Aktif', jika sistem backend Express.js tidak melakukan interupsi melalui perintah ROLLBACK otomatis, maka status inventori di database MySQL akan tetap berkurang (mengalami desinkronisasi data) padahal robot secara fisik gagal mengeksekusi tugas tersebut. Kegagalan penanganan data pada fase ini dapat menyebabkan kekacauan pada sistem manajemen gudang (*Warehouse Management System*). Oleh karena itu, keberhasilan pengujian transaksi dengan success rate 100% ini membuktikan bahwa logika backend yang dirancang berhasil menjadi gerbang pengaman (gatekeeper) yang menjamin bahwa perubahan data di lapisan IT akan selalu sejalan dengan kondisi aktual perangkat OT di lapangan.

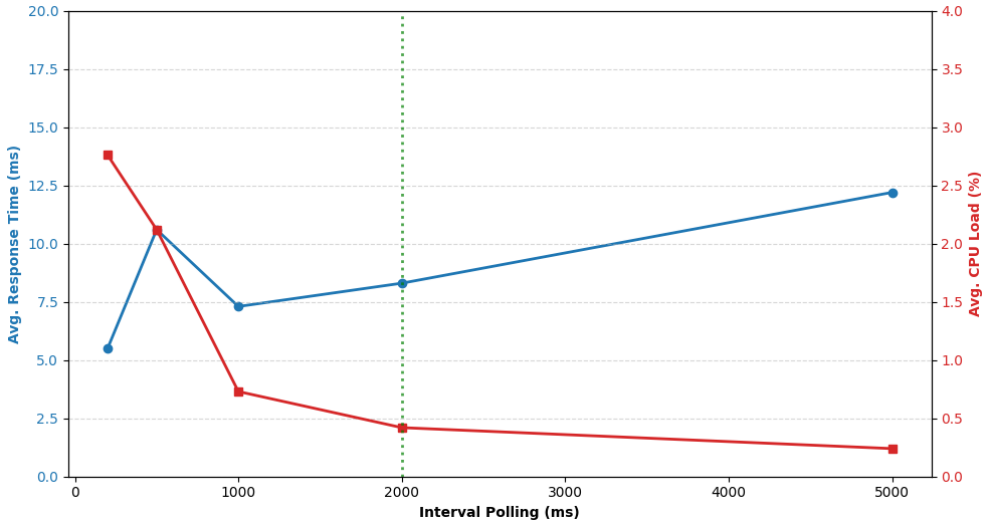
Analisis Performa Komunikasi Hybrid

Pengujian jalur kontrol ARM menggunakan Modbus TCP menghasilkan latensi penulisan rata-rata 14–25 ms dengan *CPU overhead* mendekati 0%, konsisten dengan karakteristik deterministik protokol ini (Găitan et al., 2024)(Zagan & Găitan, 2022). Untuk jalur telemetri AMR, hasil pengujian lima interval *polling* REST API disajikan pada Tabel 5.

Tabel 5 Hasil pengujian Interval Polling REST API terhadap performa sistem

Interval Polling (ms)	Response Time (ms)	CPU Load (%)	Idle Time (s/siklus)	Keterangan
200	3 – 11	1.5 – 5.5	0.2	Overload
500	4 – 71	0.8 – 14.2	0.4 – 0.5	CPU Tinggi
1000	4 – 41	0.4 – 4.1	1.0	Baik
2000	4 – 82	0.2 – 4.1	1.9 - 2.0	Optimal
5000	4 – 52	0.1 – 1.0	4.9 – 5.0	Idle Time Berlebih

Interval 200 ms dikategorikan *overload* karena CPU load mencapai 1,5–5,5% akibat frekuensi *request* yang terlalu tinggi, sementara interval 500 ms menunjukkan fluktuasi *response time* yang besar (4–71 ms) dengan CPU load hingga 14,2%, mengindikasikan ketidakstabilan sistem. Interval 2000 ms terpilih sebagai titik optimal karena CPU load turun signifikan ke rentang 0,2–4,1% dengan *idle time* 1,9–2,0 detik per siklus yang masih proporsional terhadap kebutuhan responsivitas sistem. Meskipun interval 5000 ms menghasilkan CPU load terendah (0,1–1,0%), *idle time*-nya yang mencapai 4,9–5,0 detik per siklus secara langsung mengurangi *throughput* sistem.



Gambar 6. Grafik Perbandingan Intervall Polling REST API

Untuk mengukur beban komputasi yang ditimbulkan oleh setiap interval *polling* secara kuantitatif, digunakan metrik *Overhead Ratio* (OR) yang merepresentasikan produk antara rata-rata beban CPU dengan rata-rata *response time*. Semakin kecil nilai OR, semakin efisien interval

tersebut dalam memanfaatkan sumber daya komputasi terhadap kecepatan respons yang dihasilkan (Calderón et al., 2024)(Găitan et al., 2024):

$$OR\_i = CPU\_avg,i \times (RT\_avg,i / 1000)$$

Keterangan Variabel :
OR_i = Overhead Ratio pada interval polling ke-i (% · s)
CPU_avg,i = rata-rata CPU load pada interval ke-i (%), dihitung dari (CPU_min + CPU_max) / 2
RT_avg,i = rata-rata response time pada interval ke-i (ms), dihitung dari (RT_min + RT_max) / 2
1000 = faktor konversi ms ke detik
Substitusi nilai rata-rata dari Tabel 5 ke dalam persamaan menghasilkan :

Tabel 6 Hasil Perhitungan Overhead Ratio (OR) per Interval Polling

Interval	RT_avg (ms)	CPU_avg (%)	OR = CPU x (RT/1000)	Idle Time (s)	Keterangan
200 ms	7.0	3.50	0.0245	0,2 s	Overload
500 ms	37.5	7.50	0.2813	0,4 – 0,5 s	CPU Tinggi
1000 ms	22.5	2.25	0.0506	1,0 s	Baik
2000 ms	43.0	2.15	0.0924	1,9 – 2,0 s	Optimal
5000 ms	28.0	0.55	0.0154	4,9 – 5,0 s	Idle Berlebih

Perhitungan untuk Interval 2000 ms :
RT_avg,2000 = (4 + 82) / 2 = 43,0 ms
CPU_avg,2000 = (0,2 + 4,1) / 2 = 2,15 %
OR_2000 = 2,15 × (43,0 / 1000) = 2,15 × 0,043 = 0,0924 % · s

Data pada Tabel 6 menunjukkan bahwa interval 5000 ms menghasilkan OR terkecil (0,0154 % · s) karena CPU load-nya yang sangat rendah. Namun pemilihan interval optimal tidak dapat didasarkan pada nilai OR semata, karena *idle time*-nya yang mencapai 4,9–5,0 detik per siklus berimplikasi langsung pada penurunan *throughput* sistem. Interval 200 ms memiliki OR sebesar 0,0245 % · s namun dikategorikan *overload* karena fluktuasi CPU yang tinggi (1,5–5,5%). Interval 2000 ms, meskipun memiliki OR sebesar 0,0924 % · s, dipilih sebagai titik optimal karena nilai OR-nya masih sangat rendah dalam skala absolut dan menghasilkan *idle time* 1,9–2,0 detik yang proporsional dengan kebutuhan responsivitas sistem gudang cerdas. Dengan demikian, pemilihan interval 2000 ms merupakan keputusan desain yang mempertimbangkan keseimbangan antara OR, *idle time*, dan stabilitas sistem secara menyeluruh (Calderón et al., 2024)(Oñate & Sanz, 2024). Hal ini membuktikan bahwa pemilihan interval 2000 ms bukan hanya mengoptimalkan kinerja fisik robot, tetapi secara arsitektural berhasil menjaga efisiensi konsumsi bandwidth jaringan dan mencegah terjadinya CPU Overload pada server backend Express.js

Sebagaimana divisualisasikan pada Gambar 6, kurva *response time* (biru, sumbu kiri) menunjukkan lonjakan tertinggi pada interval 500 ms sebelum kembali stabil, sementara kurva CPU load (merah, sumbu kanan) menurun konsisten seiring meningkatnya interval *polling*. Garis putus-putus vertikal hijau pada Gambar 6 menandai interval 2000 ms sebagai titik di mana kedua parameter mencapai kondisi stabil dan efisien secara bersamaan. Perbandingan karakteristik teknis kedua protokol dirangkum pada Tabel 7, yang mempertegas bahwa Modbus TCP unggul untuk kontrol deterministik sementara REST API lebih sesuai untuk pertukaran data telemetri yang dinamis (Calderón et al., 2024)(Elamanov et al., 2024).

Tabel 7 Perbandingan Karakteristik REST API dan Modbus dalam sistem

Parameter	REST API (AMR)	Modbus TCP (ARM)
Tipe komunikasi	Asinkron request-response	Sinkron
Format data	JSON	Register integer
Rata-rata Latency	191-196 ms.	14-25 ms per write
CPU overhead	<1% (pada interval 2s)	~0% (sangat ringan)
Fungsi utama	Telemetri & pelaporan status	Kontrol mesin

Evaluasi Orkestrasi Sekuensial Berbasis Task Queue

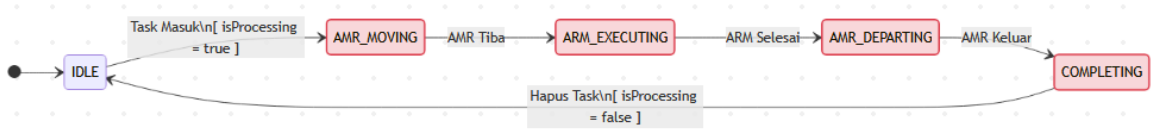
Sistem orkestrasi menggunakan *task queue* berbasis FIFO yang dikelola dalam memori Node-RED. Antrean diproses secara sekuensial melalui empat fase (mekanisme domino),

dengan mekanisme *state lock* (variabel globalContext.*isProcessing*) yang mengisolasi setiap siklus aktif dari instruksi konkuren. Log transisi antrean selama pengujian dengan lima tugas aktif ditunjukkan pada Tabel 8.

Tabel 8 Log Transisi State Task Queue Selama Pengujian (5 Tugas Aktif)

Parameter	Isi Task Queue	Aksi Robot	Event Pemicu
T0	[P1,P1,P2,P3,P3,P3]	Sistem standby	6 Tugas masuk dari backend
T1	[P1,P1,P2,P3,P3,P3]	AMR bergerak ke pos 1	isProcessing = true
T2	[P1,P1,P2,P3,P3,P3]	Amr Beroperasi di Pos 1	AMR: arrived (REST API)
T3	[P2,P3,P3,P3]	AMR bergerak ke Pos 2	ARM: done (Modbus Reg) -> .shift()
T4	[P3,P3,P3]	AMR bergerak ke Pos 3	ARM done -> .shift()
T5	[]	Sistem kembali IDLE	Tugas terakhir selesai

Data Tabel 7 mengkonfirmasi bahwa urutan eksekusi tugas selalu sesuai urutan masuk (FIFO) tanpa lompatan. Dari 50 siklus pengujian *end-to-end*, sistem mencapai success rate 100% pada lingkungan prototipe terkontrol tanpa satu pun kejadian *race condition*. Perlu dicatat bahwa pengujian ini dilakukan dalam skala prototipe satu unit AMR dan satu unit ARM; validasi pada lingkungan multi-robot berskala industri masih memerlukan kajian lanjutan.



Gambar 7. State Diagram Mekanisme Orkestrasi dan State Lock

Analisis Waktu Siklus

Pengukuran *cycle time* dilakukan dari saat Node-RED mengirimkan perintah gerak pertama ke AMR hingga sistem kembali ke kondisi siap tugas berikutnya, menggunakan fungsi timestamp bawaan Node-RED. Hasil pengukuran 50 siklus disajikan pada Tabel 9.

Tabel 9 Hasil pengukuran Waktu Siklus

Skenario	Siklus	Transport AMR (s)	Operasi ARM (s)	Resp. Middleware (ms)	Total
Normal	1	8.0	14.7	191	22.91
Baseline	25	8.1	14.8	196	23.12
	Rata-rata	8.0	14.9	195	23.12
Cepat	26	5.1	10.1	196	15.39
Optimasi	50	5.2	10.0	195	15.40
	Rata-rata	5.0	10.0	195	15.21

Untuk mengukur kapasitas layanan sistem secara kuantitatif, dihitung nilai *throughput* (λ) yang merepresentasikan jumlah tugas yang dapat diselesaikan sistem per satuan waktu, menggunakan persamaan berikut:

$$\lambda = N / T\_total$$

Dalam persamaan ini, λ merepresentasikan *throughput* sistem dalam satuan tugas/detik, N adalah jumlah tugas yang diselesaikan dalam satu skenario, dan T_{total} menunjukkan total waktu operasi seluruh siklus dalam skenario tersebut dalam satuan detik.

Berdasarkan data eksperimen, perhitungan pada skenario *baseline* menghasilkan nilai $\lambda = 25/(25x23\ 12) = 25/578\ 0 = 0\ 043$ tugas/detik. Sementara itu, perhitungan pada skenario optimasi menghasilkan nilai $\lambda = 25\ /\ (25\ x\ 15\ 21) = 25\ /\ 380\ 3 = 0\ 066$ tugas/detik. Melalui

perbandingan kedua kondisi tersebut, ditemukan peningkatan *throughput* sistem sebesar $((0\ 066 - 0\ 043)/0\ 043)x100\% = 53\ 5\%$,

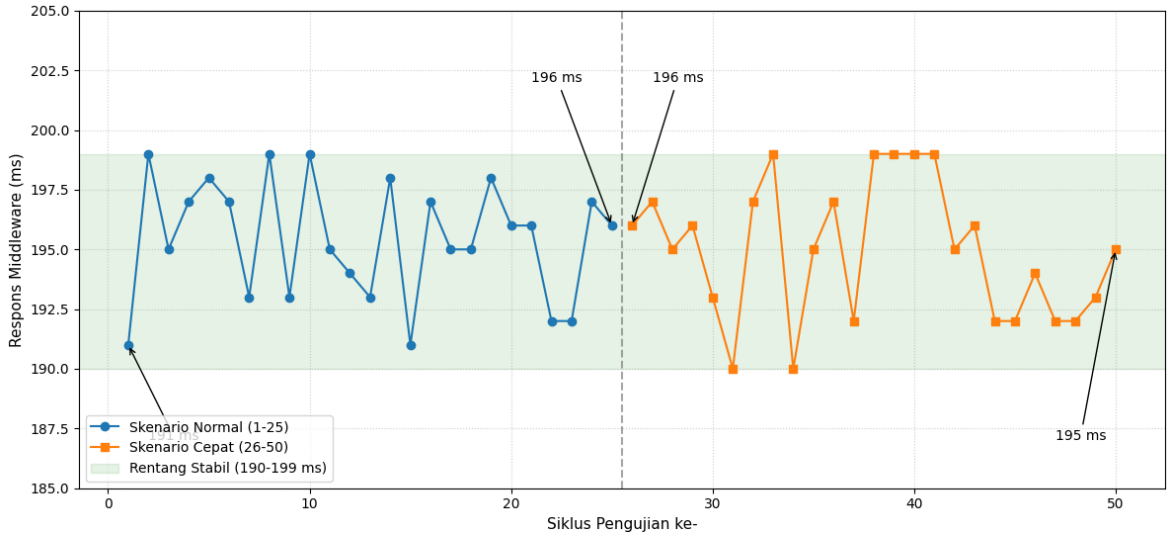
Selanjutnya, persentase peningkatan efisiensi total *cycle time* ΔCT dihitung melalui persamaan berikut:

$$\Delta CT = ((CT\_baseline - CT\_optimasi) / CT\_baseline) \times 100\%$$

Nilai ΔCT merupakan rata-rata *cycle time* pada skenario *baseline* sebesar 23,12 detik, sedangkan $CT_{optimasi}$ adalah rata-rata *cycle time* pada skenario optimasi sebesar 15,21 detik. Substitusi nilai ke dalam rumus menghasilkan $\Delta CT = ((23\ 12 - 15\ 21)/23\ 12)\ x100\% = 34\ 2\%$,

Nilai ΔCT sebesar 34,2% menunjukkan peningkatan efisiensi waktu siklus yang signifikan antara skenario *baseline* dan optimasi. Penting dicatat bahwa seluruh peningkatan ini berasal dari perbaikan kapasitas mekanis robot, sementara waktu respons *middleware* tetap stabil di 195 ms pada kedua kondisi. Hal ini secara matematis membuktikan bahwa *middleware* Node-RED tidak menjadi *bottleneck* dan arsitektur yang diusulkan bersifat *scalable* terhadap peningkatan performa fisik robot.

Data pada Tabel 9 menunjukkan bahwa sistem orkestrasi diuji pada dua kondisi operasional: skenario *baseline* (siklus 1–25) dan skenario optimasi kecepatan robot (siklus 26–50). Pada skenario *baseline*, rata-rata *cycle time* tercatat sebesar 23,12 detik, dengan komponen waktu transport AMR 8,0 detik (34,6%), waktu operasi ARM 14,9 detik (64,4%), dan waktu respons *middleware* 195 ms (0,8%). Pada skenario optimasi, rata-rata *cycle time* turun menjadi 15,21 detik, didorong oleh penurunan waktu transport AMR menjadi 5,0 detik dan operasi ARM menjadi 10,0 detik, sementara waktu respons *middleware* tetap stabil di 195 ms.



Gambar 8. Grafik Kestabilan Waktu Respon Middleware

Poin paling krusial ditunjukkan pada Gambar 8, di mana seluruh titik pengukuran respons *middleware* tetap konstan dan stabil pada rentang 191–196 ms, baik pada skenario *baseline* maupun optimasi cepat. Fenomena konstan ini secara matematis membuktikan bahwa lapisan validasi transaksi berbasis ACID pada *backend* Express.js serta pengolahan protokol *hybrid* pada Node-RED bersifat *decoupled* (terlepas) dari performa mekanis robot. Sistem perangkat lunak yang dirancang terbukti tidak menjadi *bottleneck* komputasi dan siap mendukung peningkatan skala utilitas robot heterogen di lantai produksi.Temuan ini mengindikasikan bahwa peningkatan produktivitas gudang lebih efektif dicapai melalui optimasi kecepatan mekanis robot dibandingkan peningkatan kapasitas komputasi *middleware*, konsisten dengan kajian Zafar et al(Zafar et al., 2024).

Perbandingan dengan penelitian sebelumnya

Tabel 10 menyajikan perbandingan sistematis antara penelitian ini dengan penelitian terkait berdasarkan empat dimensi: fokus, protokol, orkestrasi, dan validasi ACID.

Tabel 10 Perbandingan Penelitian Ini dengan Penelitian Sebelumnya

Penelitian	Fokus	Protokol	Orkestrasi	ACID	Catatan
Li et al.	Task Allocation robot heterogen	Simulasi	Tidak ada	Tidak ada	Tanpa integrasi IT-OT
Msala et. al	Multi Robot Warehous	Simulasi	Hungarian+TSP	Tidak	Fokus Path Planning
Elemanov et al.	Interworking Modbus-IoT	Modbus+oneM2M	Tidak ada	Tidak ada	Tanpa Task Orchestration
Onate & Sanz	Fog Computing + Node-RED	Node-RED	Parsial	Tidak ada	Tanpa Koordinasi AMR-ARM
Embong et al	Node-RED +PLC industri	MQTT+Modbus	Tidak ada	Tidak ada	Monitoring bukan orkestrasi
Penelitian ini	Orkestrasi AMR-ARM hybrid	REST API+Modbus	Task Queue + State Lock	Penuh	Integrasi Real-time

Berdasarkan Tabel 10, terlihat jelas bahwa penelitian-penelitian sebelumnya menangani masing-masing tantangan secara terpisah: Li et al. (Li et al., 2025) dan Msala et al. (Msala et al., 2025) berfokus pada alokasi dan sekuensiasi tugas multi-robot namun tidak menangani integrasi protokol IT-OT; Elamanov et al. (Elamanov et al., 2024) membahas interoperabilitas Modbus-IoT namun tanpa mekanisme orkestrasi tugas; Oñate dan Sanz (Oñate & Sanz, 2024) menggunakan Node-RED untuk integrasi manufaktur namun tidak spesifik pada koordinasi AMR-ARM; sementara Embong et al. (A. H. Embong et al., 2024) memvalidasi Node-RED untuk monitoring industri namun bukan untuk orkestrasi multi-robot.
Berbeda dari penelitian-penelitian tersebut, penelitian ini mengintegrasikan orkestrasi AMR-ARM, middleware Node-RED, konversi protokol hybrid REST API Modbus TCP, serta validasi transaksi backend berprinsip ACID dalam satu arsitektur yang terhubung secara *real-time*. Integrasi menyeluruh ini merupakan kontribusi utama penelitian yang menjawab ketiga kesenjangan yang telah diidentifikasi secara simultan, sesuatu yang belum dilakukan oleh penelitian sebelumnya secara individual.

KESIMPULAN

Penelitian ini telah berhasil mengimplementasikan dan mengevaluasi arsitektur validasi transaksi berbasis prinsip ACID serta jalur komunikasi protokol *hybrid* pada sistem orkestrasi multi-robot heterogen di lingkungan SmartEDU Station Warehouse. Lapisan *backend* berbasis Express.js dengan pola operasi transaksi basis data terbukti andal dalam menjamin integritas data, dengan tingkat keberhasilan penyelamatan data melalui mekanisme *rollback* otomatis yang mencapai angka 100% pada semua skenario kegagalan operasional yang disimulasikan. Sementara itu, integrasi jalur komunikasi *hybrid* yang dikelola oleh *middleware* Node-RED mampu menyelesaikan kendala perbedaan bahasa protokol secara struktural, sehingga menghasilkan nilai latensi penulisan kontrol Modbus TCP yang deterministik sebesar 14–25 ms serta pemrosesan telemetri *REST API* yang ringan dengan beban *CPU load* sebesar 0,2–4,1% pada interval pembaruan 2000 ms. Kestabilan waktu respons *middleware* yang berada dalam rentang aman 191–196 ms mengonfirmasi bahwa penambahan lapisan pengaman logika transaksi di sisi *backend* tidak menimbulkan efek *bottleneck* ataupun membebani kinerja operasional fisik armada robot otonom. Hasil akhir pengujian ini berhasil menyediakan sebuah cetak biru arsitektur perangkat lunak perantara yang aman, efisien, dan teruji untuk menjembatani sinkronisasi pertukaran informasi secara *real-time* antara lapisan teknologi informasi dengan perangkat keras di lantai produksi industrial

UCAPAN TERIMA KASIH

Penelitian ini didukung oleh Kementerian Pendidikan Tinggi, Sains, dan Teknologi Indonesia di bawah Skema Hibah Penelitian Fundamental 136/SPK/C.C4/PPK.DHK/IV/2026.

REFERENSI

A. H. Embong, Asbollah, L., & Abdul Hamid, S. B. (2024). Empowering industrial automation labs with IoT: A case study on real-time monitoring and control of induction motors using Siemens PLC and Node-RED. *Journal of Mechanical Engineering and Sciences*, 10004–10016. https://doi.org/10.15282/jmes.18.2.2024.3.0790

Alam, M. M., Ugli, R. A. N., Tanha, K. J., & Jun, T. (2026). AI-enhanced Digital Twin systems for warehouse logistics optimization: A review of challenges with solutions and future directions. *ICT Express*, *12*(2), 459–479. https://doi.org/10.1016/j.icte.2026.01.009

Calderón, D., Folgado, F. J., González, I., & Calderón, A. J. (2024). Implementation and Experimental Application of Industrial IoT Architecture Using Automation and IoT Hardware/Software. *Sensors*, *24*(24), 8074. https://doi.org/10.3390/s24248074

Chen, Y., Ren, T., Li, Y., Jiang, G., Liu, Q., Chen, Y., & Yang, S. X. (2026). AI-empowered intelligence in industrial robotics: Technologies, challenges, and emerging trends. *Intelligence & Robotics*, *6*(1), 1–18. https://doi.org/10.20517/ir.2026.01

Elamanov, S., Son, H., Flynn, B., Yoo, S. K., Dilshad, N., & Song, J. (2024). Interworking between Modbus and internet of things platform for industrial services. *Digital Communications and Networks*, *10*(2), 461–471. https://doi.org/10.1016/j.dcan.2022.09.013

Găitan, N. C., Zagan, I., & Găitan, V. G. (2024). Proposed Modbus Extension Protocol and Real-Time Communication Timing Requirements for Distributed Embedded Systems. *Technologies*, *12*(10), 187. https://doi.org/10.3390/technologies12100187

Harja, H. B., Setiawan, H., Erdani, Y., & Febriansyah, M. Z. (2022). Development of Virtual Model for Cyber-Physical Screw Turbine. *2022 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS)*, 38–42. https://doi.org/10.1109/IoTaIS56727.2022.9975960

Lackner, T., Hermann, J., Kuhn, C., & Palm, D. (2024). Review of autonomous mobile robots in intralogistics: State-of-the-art, limitations and research gaps. *Procedia CIRP*, *130*, 930–935. https://doi.org/10.1016/j.procir.2024.10.187

Li, L., Chen, Z., Wang, H., & Kan, Z. (2025). Task Allocation of Heterogeneous Robots Under Temporal Logic Specifications With Inter-Task Constraints and Variable Capabilities. *IEEE Transactions on Automation Science and Engineering*, *22*, 14030–14047. https://doi.org/10.1109/TASE.2025.3558977

Loganathan, A., & Ahmad, N. S. (2023). A systematic review on recent advances in autonomous mobile robot navigation. *Engineering Science and Technology, an International Journal*, *40*, 101343. https://doi.org/10.1016/j.jestch.2023.101343

Msala, Y., Oussama, H., Talea, M., & Aboulfatah, M. (2025). A Novel Method for Enhancing Warehouse Operations Using Heterogeneous Robotic Systems for Autonomous Pick-and-Deliver Tasks. *EAI Endorsed Transactions on AI and Robotics*, *4*. https://doi.org/10.4108/airo.9913

Node-RED Project. (2024). Node-RED: Low-code programming for event-driven applications. Retrieved April 23, 2026, from https://nodered.org/

Ochoa, W., Larrinaga, F., & Pérez, A. (2023). Context-aware workflow management for smart manufacturing: A literature review of semantic web-based approaches. *Future Generation Computer Systems*, *145*, 38–55. https://doi.org/10.1016/j.future.2023.03.017

Oñate, W., & Sanz, R. (2024). Integration of Fog Computing in a Distributed Manufacturing Execution System Under the RAMI 4.0 Framework. *Applied Sciences*, *14*(22), 10539. https://doi.org/10.3390/app142210539

Pan, Y., Wu, D., Du, D., & Wang, H. (2023). Design and Performance Analysis of Protocol Conversion between 5G and Modbus TCP. *2023 42nd Chinese Control Conference (CCC)*, 6262–6267. https://doi.org/10.23919/CCC58697.2023.10240445

Real Time Automation. (n.d.). Modbus TCP/IP (Modbus Over Ethernet). Retrieved April 21, 2026, from Real Time Automation website: https://www.rtautomation.com/technologies/modbus-tcpip/

Ridwan, Erdani, Y., Abadi, S. C., Hidayat, M. D. A., Harits, R., & Pratama, R. A. (2025). *Developing Sorting Algorithm for SmartEdu Conveyor using Computer Vision Technology*. https://doi.org/https://doi.org/10.52958/iftk.v21i3.12434

Torres Ventura, J., Ruelas Puente, A. H., & Herrera García, J. R. (2023). PERFORMANCE FOR INTEROPERABILITY BETWEEN RASPERRY PI AND ESP8266 WITH A PLC IN A NODE-RED SERVER FOR IIOT. *Ingenius*, (29), 90–97. https://doi.org/10.17163/ings.n29.2023.08

Tubis, A. A., & Rohman, J. (2023). Intelligent Warehouse in Industry 4.0—Systematic Literature Review. *Sensors*, *23*(8), 4105. https://doi.org/10.3390/s23084105

Zafar, M. H., Langås, E. F., & Sanfilippo, F. (2024). Exploring the synergies between collaborative robotics, digital twins, augmentation, and industry 5.0 for smart manufacturing: A state-of-the-art review. *Robotics and Computer-Integrated Manufacturing*, *89*, 102769. https://doi.org/10.1016/j.rcim.2024.102769

Zagan, I., & Găitan, V. G. (2022). Enhancing the Modbus Communication Protocol to Minimize Acquisition Times Based on an STM32-Embedded Device. *Mathematics*, *10*(24), 4686. https://doi.org/10.3390/math10244686

Zhang, S., Han, Q., Zhu, H., Wang, H., Li, H., & Wang, K. (2025). Real time task planning for order picking in intelligent logistics warehousing. *Scientific Reports*, *15*(1), 7331. https://doi.org/10.1038/s41598-025-88305-9