SEMINAR NASIONAL 2026 Universitas Ma Chung
Prosiding Seminar Nasional Universitas Ma Chung 2026
(Sistem Informasi dan Informatika)
pp. 126-137
E-ISSN: 2808-7771

# PENGEMBANGAN DAN EVALUASI KINERJA *SERVER IOT* BERBASIS *RUST* MENGGUNAKAN *FRAMEWORK NTEX*

**Irfan Alamsyah[1], Ahmad Ridha[2]**

[1,2]Program Studi Ilmu Komputer, Sekolah Sains Data, Matematika dan Informatika, Institut Pertanian Bogor, Bogor, Indonesia

***Email korespondensi:** ridha@apps.ipb.ac.id

*Received: 27 05 2026 – Revised: 29 07 2026 - Accepted: 04 08 2026 - Published: 05 08 2026*

**Abstrak.** Pertumbuhan *Internet of Things* (IoT) mendorong peningkatan kebutuhan terhadap infrastruktur server yang mampu mengelola data dari berbagai perangkat secara efisien. Perangkat IoT umumnya memiliki keterbatasan sumber daya komputasi dan penyimpanan sehingga memerlukan dukungan server yang memiliki kinerja tinggi dan mampu menangani banyak permintaan secara bersamaan. Penelitian sebelumnya telah mengembangkan server IoT menggunakan *framework* Python Falcon dan Go Fiber dengan berbagai penyempurnaan pada desain sistem. Penelitian ini bertujuan mengembangkan server IoT berbasis bahasa pemrograman Rust menggunakan *framework* Ntex serta mengevaluasi kinerjanya dibandingkan dengan implementasi sebelumnya yang menggunakan Go Fiber. Pengembangan sistem dilakukan melalui tahapan analisis kebutuhan, desain, implementasi, dan pengujian. Sistem yang dikembangkan menggunakan arsitektur REST API dengan PostgreSQL sebagai basis data dan JSON sebagai format pertukaran data. Pengujian fungsional dilakukan menggunakan Postman untuk memastikan seluruh fitur berjalan sesuai kebutuhan, sedangkan pengujian kinerja dilakukan menggunakan Grafana k6 dengan metrik transaksi per detik pada tingkat konkurensi 200 hingga 1000 pengguna simultan. Hasil pengujian menunjukkan bahwa seluruh kebutuhan fungsional sistem berhasil dipenuhi. Selain itu, implementasi menggunakan *framework* Ntex menghasilkan kinerja yang lebih tinggi pada seluruh *endpoint* yang diuji dibandingkan dengan implementasi Go Fiber. Ntex menunjukkan peningkatan kinerja rata-rata sekitar 20% berdasarkan transaksi per detik yang berhasil diproses. Hasil penelitian menunjukkan bahwa Ntex merupakan alternatif yang layak untuk pengembangan *backend* server IoT yang membutuhkan performa tinggi.

**Kata kunci:** IoT, Ntex, REST API, Rust, server IoT

## PENDAHULUAN

*Internet of Things* (IoT) memungkinkan berbagai perangkat fisik saling terhubung dan bertukar data melalui jaringan internet. Peningkatan jumlah perangkat yang terhubung berkonsekuensi kebutuhan terhadap sistem yang mampu mengelola data secara efektif dan andal juga semakin besar. Pasar IoT di Indonesia diproyeksikan terus bertumbuh sebesar 14,76% per tahun di periode 2026-2031 (Mordor Intelligence Research & Advisory, 2026a). Tren serupa juga terlihat pada tingkat global dengan semakin luasnya adopsi solusi IoT untuk mendukung aneka proses bisnis dengan prediksi pertumbuhan pasar perangkat

IoT sebesar 14,34% untuk lima tahun ke depan (Mordor Intelligence Research & Advisory, 2026b).

Sebagian besar perangkat IoT memiliki keterbatasan sumber daya komputasi, kapasitas penyimpanan, dan konsumsi daya. Oleh karena itu, pemrosesan dan penyimpanan data umumnya terpusat pada server yang berperan sebagai penghubung antara perangkat, basis data, dan pengguna. Server menerima dan menyimpan data dari perangkat serta menyediakan layanan bagi aplikasi dan pengguna untuk mengakses data tersebut. Kemampuan server menjadi salah satu faktor penting yang menentukan skalabilitas dan keandalan sistem.

Hanin (2021) mengembangkan server IoT yang memenuhi kebutuhan dasar sistem IoT menggunakan *framework* Falcon pada bahasa pemrograman Python. Selanjutnya, Ferdiansyah (2023) menyempurnakan rancangan sistem melalui perbaikan struktur data dan pengelolaan komponen sehingga sistem menjadi lebih terorganisasi dan mudah dikembangkan. Athaullah (2023) menggunakan *framework* Go Fiber dengan rancangan sistem yang sama dan menghasilkan peningkatan kinerja. Hasil-hasil penelitian sebelumnya menunjukkan bahwa pilihan teknologi implementasi berpengaruh terhadap kinerja server IoT.

Rust berfokus pada keamanan memori dan efisiensi eksekusi (Matsakis & Klock, 2014; Bugden & Alahmar, 2022). Ntex merupakan *framework* dalam ekosistem Rust yang menyediakan dukungan untuk pengembangan layanan berbasis HTTP dan REST API (Ntex Contributors, 2025). Kombinasi Rust dan Ntex menjadi implementasi yang menarik untuk dievaluasi pada pengembangan server IoT.

Berdasarkan kesenjangan penelitian tersebut, penelitian ini mengimplementasikan server IoT berbasis Rust dan *framework* Ntex dengan menggunakan rancangan sistem pada penelitian sebelumnya agar perbandingan dapat difokuskan pada pengaruh teknologi implementasi yang digunakan. Evaluasi mencakup pengujian fungsional untuk kesesuaian sistem terhadap kebutuhan yang telah ditetapkan dan pengujian kinerja menggunakan Grafana k6 dengan metrik *transactions per second* (TPS).

Penelitian ini bertujuan mengimplementasikan server IoT berbasis Rust dan framework Ntex serta mengevaluasi kinerjanya melalui pengujian fungsional dan pengujian beban. Hasil pengujian kemudian dibandingkan dengan implementasi server IoT berbasis Go Fiber yang dikembangkan pada penelitian sebelumnya. Dengan demikian,

penelitian ini diharapkan dapat memberikan gambaran mengenai kelayakan Rust dan Ntex sebagai alternatif teknologi *backend* untuk pengembangan aplikasi IoT.

## MASALAH

Rancangan server IoT yang dikembangkan pada penelitian-penelitian sebelumnya telah menyediakan kebutuhan fungsional yang diperlukan untuk pengelolaan pengguna, perangkat keras, *node*, dan data hasil pengukuran. Selain itu, implementasi berbasis Go Fiber menunjukkan peningkatan kinerja dibandingkan implementasi sebelumnya yang menggunakan Python Falcon. Namun, hasil tersebut belum menunjukkan apakah kinerja sistem telah mencapai tingkat yang optimal atau masih dapat ditingkatkan melalui pemilihan teknologi implementasi yang berbeda.

Dalam pengembangan sistem *backend*, pemilihan bahasa pemrograman dan *framework* memengaruhi berbagai aspek, seperti efisiensi pemrosesan permintaan, penggunaan sumber daya, dan kemampuan menangani beban yang tinggi. Rust dan Ntex dikenal sebagai teknologi yang banyak digunakan untuk membangun layanan jaringan dan aplikasi berkinerja tinggi sehingga menarik untuk diuji pada rancangan sistem penelitian sebelumnya.

Dengan demikian, penelitian ini berfokus pada dua permasalahan utama. Pertama, bagaimana mengimplementasikan server IoT berbasis Rust dan *framework* Ntex dengan rancangan sistem sebelumnya. Kedua, bagaimana kinerja implementasi tersebut dibandingkan dengan implementasi berbasis Go Fiber ketika diuji menggunakan skenario dan lingkungan pengujian yang setara. Hasil penelitian ini diharapkan dapat menunjukkan pengaruh penggunaan Rust dan Ntex terhadap kinerja server IoT pada rancangan sistem yang sama.

## METODE PELAKSANAAN

Penelitian ini mengimplementasikan server IoT berbasis Rust dan *framework* Ntex berdasarkan rancangan sistem pada penelitian sebelumnya. Evaluasi difokuskan pada pengaruh teknologi implementasi terhadap kinerja server. Tahapan penelitian mencakup analisis kebutuhan, desain sistem, implementasi, dan pengujian.

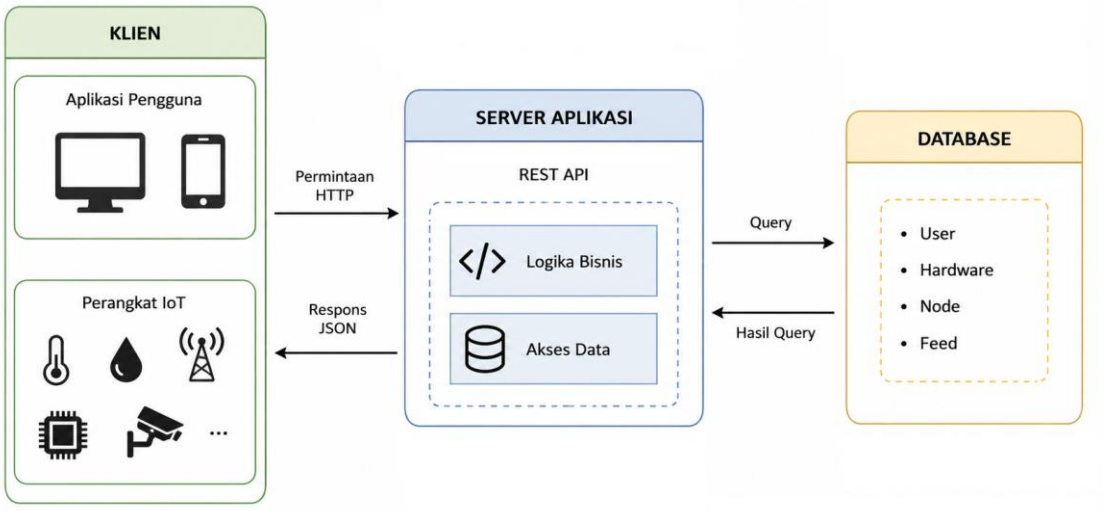
### Analisis Kebutuhan

Analisis kebutuhan mengacu pada rancangan server IoT yang telah dikembangkan sebelumnya. Fungsi utama sistem mencakup autentikasi pengguna serta pengelolaan data perangkat keras, *node* IoT, dan hasil pengukuran. Sistem juga diharapkan mampu menangani permintaan dari banyak pengguna dan perangkat secara bersamaan tanpa penurunan layanan yang signifikan.

**Desain Sistem**

Dalam sistem ini, klien dan server berkomunikasi dengan REST API. Klien mengakses layanan melalui *endpoint* HTTP yang disediakan server. Setiap permintaan diproses, kemudian hasilnya dikembalikan dalam format JavaScript *Object Notation* (JSON) (Fielding, 2000; Severance, 2012).

Sistem terdiri atas tiga komponen utama, yaitu klien, server aplikasi, dan basis data (*lihat* Gambar 1). Klien mencakup aplikasi pengguna maupun perangkat IoT yang mengirimkan data ke server. Server memroses permintaan dan mengirimkan respons, sedangkan basis data menyimpan data yang diperlukan sistem.




Gambar 1. Arsitektur sistem

Kebutuhan fungsional sistem dirancang menggunakan *Use Case Diagram* untuk menggambarkan interaksi antara aktor dan layanan yang tersedia. Kemudian, skema basis data dirancang untuk memuat data pengguna, perangkat keras, *node*, dan hasil pengukuran.

**Implementasi Sistem**

Server dikembangkan dengan bahasa pemrograman Rust dan *framework* Ntex. Implementasi mencakup seluruh *endpoint* yang telah ditetapkan pada tahap desain,

meliputi layanan autentikasi, pengelolaan perangkat keras, pengelolaan node, dan pengiriman data hasil pengukuran.

Akses ke layanan yang memerlukan otorisasi diamankan menggunakan JSON *Web Token* (JWT). Setelah proses autentikasi berhasil, server menghasilkan token yang digunakan pada setiap permintaan berikutnya. Pendekatan ini memungkinkan verifikasi identitas pengguna tanpa penyimpanan sesi pada sisi server.

Penyimpanan data memanfaatkan PostgreSQL sebagai basis data relasional. Seluruh operasi penambahan, pembaruan, penghapusan, dan pengambilan data dilakukan melalui lapisan akses data yang terintegrasi dengan aplikasi.

### Pengujian Sistem

Evaluasi sistem mencakup pengujian fungsional dan pengujian kinerja. Pengujian fungsional bertujuan memverifikasi kesesuaian fitur terhadap kebutuhan yang telah ditetapkan. Seluruh *endpoint* dievaluasi untuk memastikan setiap fungsi menghasilkan respons yang sesuai pada berbagai skenario masukan.

Pengujian kinerja menggunakan Grafana k6 dengan pendekatan *load testing*. Lingkungan pengujian terdiri atas dua *virtual machine*, yaitu satu mesin untuk menjalankan aplikasi dan satu mesin untuk menghasilkan beban uji. Konfigurasi tersebut bertujuan mengurangi pengaruh proses pembangkitan beban terhadap aplikasi yang dievaluasi.

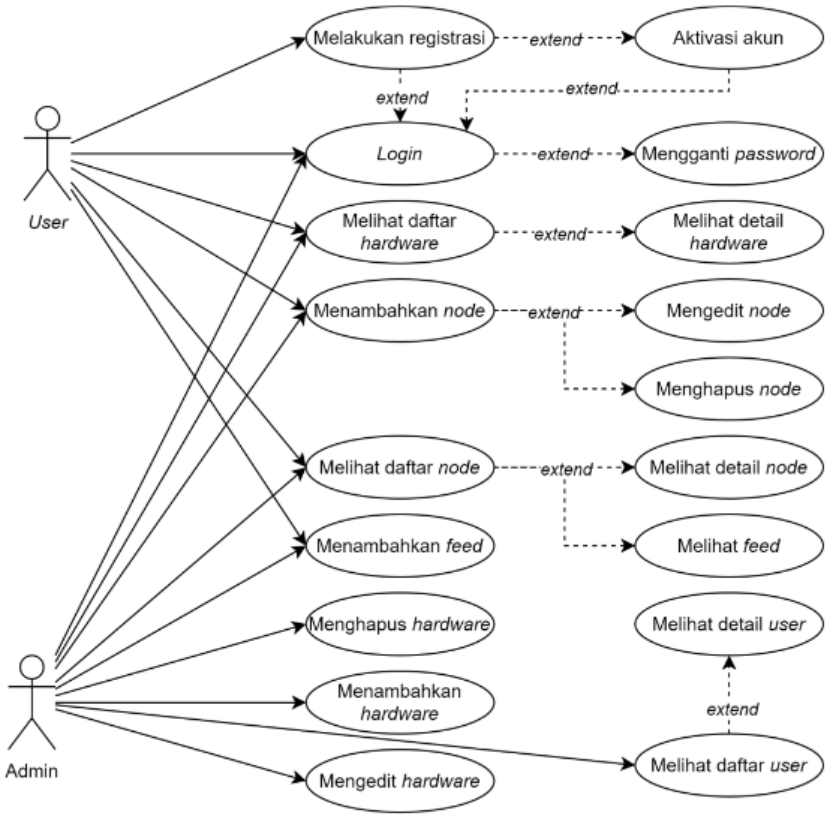
Pengujian mencakup tingkat konkurensi 200, 400, 600, 800, dan 1000 pengguna simultan. Setiap skenario berlangsung selama satu menit dan diulang sebanyak lima kali dengan interval dua menit antarpengujian. Metrik yang digunakan adalah *transactions per second* (TPS), yaitu jumlah permintaan yang berhasil diproses server setiap detik. Nilai rata-rata TPS dari seluruh pengulangan menjadi dasar perbandingan kinerja antara implementasi berbasis Ntex dan implementasi berbasis Go Fiber.

## HASIL DAN PEMBAHASAN

### Hasil Analisis dan Desain Sistem

Tahap analisis menghasilkan kebutuhan fungsional yang menjadi dasar pengembangan server IoT. Kebutuhan tersebut mencakup autentikasi pengguna serta pengelolaan data perangkat keras, *node*, dan hasil pengukuran. Berdasarkan kebutuhan tersebut, sistem menyediakan *endpoint* untuk masing-masing fungsi tersebut.

Gambar 2 menunjukkan *Use Case Diagram* sistem yang menggambarkan interaksi antara aktor dan layanan yang tersedia. Sebagian besar fungsi sistem berpusat pada pengelolaan perangkat dan data hasil pengukuran, yang merupakan komponen utama dalam arsitektur IoT.

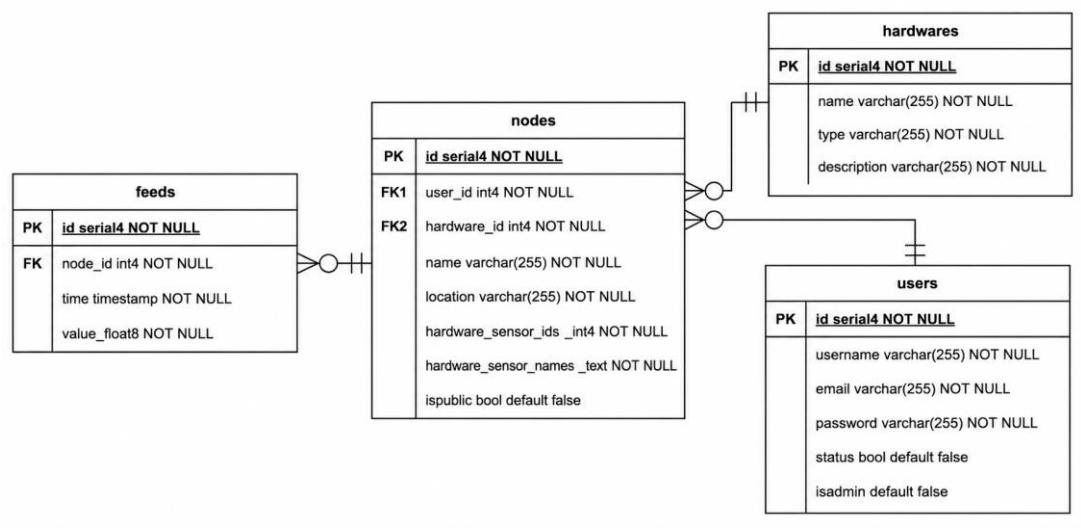



Gambar 2. *Use case diagram*

Sementara itu, struktur penyimpanan data dirancang untuk mendukung keterkaitan antara pengguna, perangkat keras, node, dan data hasil pengukuran. Skema basis data yang digunakan ditunjukkan pada Gambar 3. Rancangan tersebut memungkinkan setiap data yang dikirimkan perangkat dapat ditelusuri berdasarkan *node* dan pengguna yang terkait.

Untuk merealisasikan kebutuhan tersebut, sistem menyediakan sejumlah endpoint untuk akses ke basis data. Ringkasan *endpoint* utama yang digunakan dalam penelitian ditunjukkan pada Tabel 1. Rancangan sistem tersebut menjadi dasar implementasi server yang selanjutnya dievaluasi melalui pengujian fungsional dan pengujian kinerja.

SEMINAR NASIONAL 2026 Universitas Ma Chung
Prosiding Seminar Nasional Universitas Ma Chung 2026
(Sistem Informasi dan Informatika)
pp. 126-137
E-ISSN: 2808-7771




Gambar 3. Skema basis data yang digunakan

Tabel 1. Ringkasan Endpoint Utama

| **Endpoint** | **Operasi** |
|---|---|
| `GET /nodes` | Mengambil daftar *node* |
| `GET /nodes/:id` | Mengambil detail *node* |
| `POST /nodes` | Menambahkan *node* |
| `PUT /nodes/:id` | Memperbarui *node* |
| `DELETE /nodes/:id` | Menghapus *node* |
| `POST /channel` | Mengirim data pengukuran |

**Hasil Implementasi Sistem**

Implementasi menghasilkan server IoT berbasis bahasa pemrograman Rust dan framework Ntex yang mampu menyediakan seluruh layanan sesuai kebutuhan sistem. Server mengadopsi arsitektur REST API dan menyediakan *endpoint* untuk autentikasi pengguna, pengelolaan perangkat keras, pengelolaan *node*, serta penerimaan data hasil pengukuran dari perangkat IoT. Struktur data, fitur sistem, dan *endpoint* dipertahankan sesuai penelitian sebelumnya sehingga perbedaan hasil pengujian lebih mencerminkan pengaruh teknologi implementasi daripada perubahan rancangan sistem.

Untuk mendukung keamanan akses, sistem menerapkan mekanisme autentikasi berbasis JSON *Web Token* (JWT). Setelah proses autentikasi berhasil, server menghasilkan token yang digunakan pada setiap permintaan yang memerlukan otorisasi. Pendekatan ini memungkinkan autentikasi tanpa penyimpanan sesi pada sisi server sekaligus membatasi akses ke layanan sesuai hak pengguna.

PostgreSQL digunakan untuk menyimpan data pengguna, perangkat keras, *node*, dan data hasil pengukuran. Seluruh operasi pengambilan, penambahan, pembaruan, dan penghapusan data terintegrasi dengan basis data melalui lapisan akses data yang terpisah dari logika bisnis aplikasi. Pemisahan tersebut mempermudah pemeliharaan dan pengembangan aplikasi tanpa mengubah struktur dasar sistem.

**Hasil Pengujian Fungsional**

Pengujian fungsional bertujuan memeriksa kesesuaian implementasi terhadap kebutuhan sistem yang telah ditetapkan. Evaluasi mencakup seluruh layanan utama sistem, yaitu registrasi dan autentikasi pengguna, pengelolaan perangkat keras, pengelolaan *node*, serta pengiriman data hasil pengukuran.

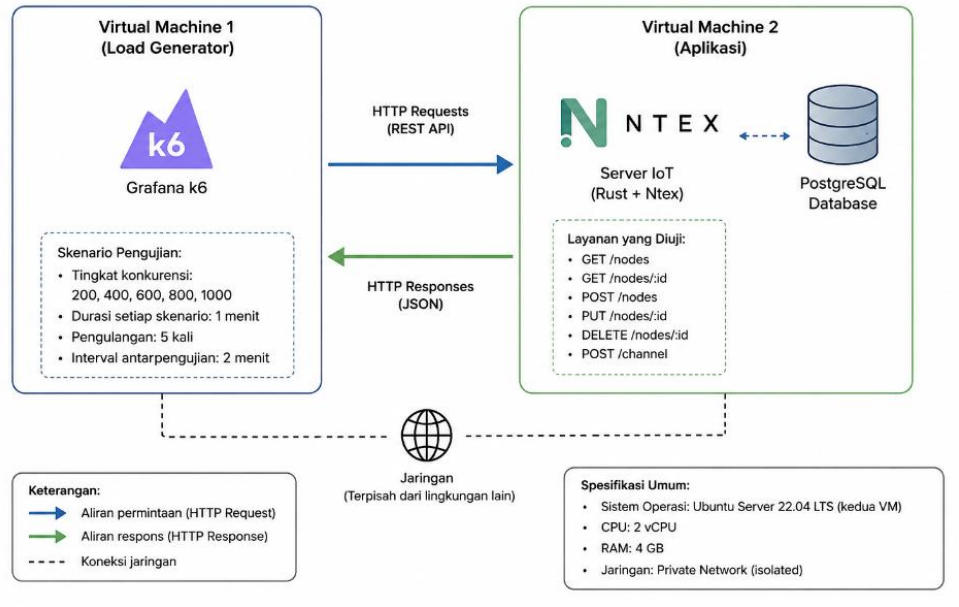
Hasil pengujian menunjukkan bahwa seluruh layanan beroperasi sesuai spesifikasi. Setiap *endpoint* menghasilkan respons yang sesuai untuk skenario masukan yang valid maupun kondisi kesalahan yang diuji. Selain itu, mekanisme autentikasi dan otorisasi berfungsi sebagaimana dirancang sehingga akses terhadap layanan yang memerlukan hak tertentu dapat dikendalikan dengan benar.

Hasil tersebut menunjukkan bahwa implementasi berbasis Rust dan Ntex mampu memenuhi seluruh kebutuhan fungsional sistem. Dengan demikian, evaluasi kinerja dapat dilakukan tanpa dipengaruhi oleh perbedaan fitur atau ruang lingkup layanan dibandingkan sistem pembanding.

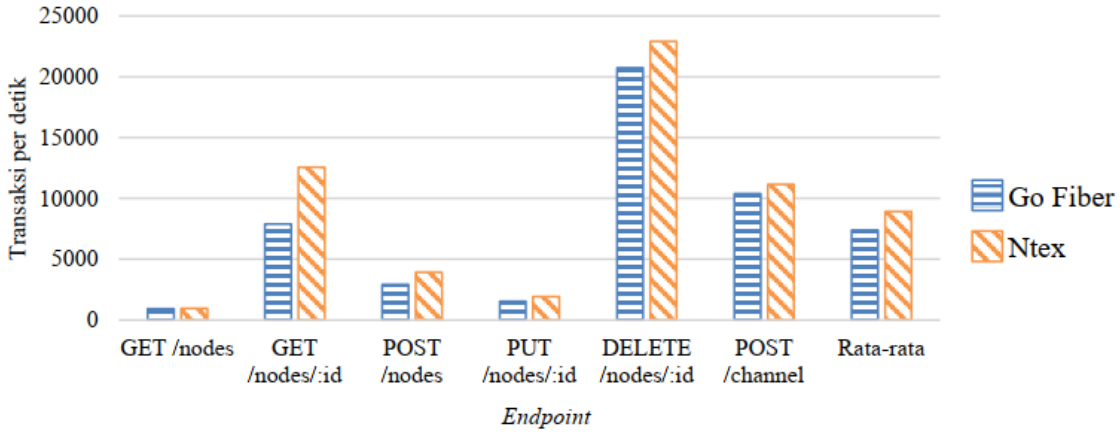
**Hasil Pengujian Kinerja**

Lingkungan pengujian yang digunakan dalam penelitian disajikan pada Gambar 4. Evaluasi dilakukan terhadap enam *endpoint* utama yang merepresentasikan operasi pengelolaan data pada sistem.

Hasil pengujian menunjukkan bahwa implementasi berbasis Ntex menghasilkan nilai TPS yang lebih tinggi dibandingkan implementasi berbasis Go Fiber pada seluruh *endpoint* yang diuji. Perbandingan rata-rata TPS antara implementasi Ntex dan Go Fiber disajikan pada Gambar 5. Secara keseluruhan, implementasi berbasis Ntex menghasilkan peningkatan kinerja rata-rata sekitar 20% dibandingkan implementasi berbasis Go Fiber berdasarkan rata-rata nilai TPS pada enam *endpoint* yang diuji.




Gambar 4. Lingkungan pengujian




Gambar 5. Kinerja Go Fiber dan Ntex pada berbagai *endpoint*

Pada *endpoint* GET /nodes, perbedaan kinerja antara kedua implementasi relatif kecil. Peningkatan terbesar dihasilkan pada *endpoint* GET /nodes/:id. Hasil ini mengindikasikan bahwa keunggulan Ntex lebih terlihat pada operasi yang melibatkan pemrosesan data spesifik dibandingkan operasi pengambilan seluruh data.

Peningkatan kinerja juga terlihat pada *endpoint* POST /nodes dan PUT /nodes/:id, tetapi tidak *endpoint* GET. *Endpoint* DELETE /nodes/:id menghasilkan nilai TPS tertinggi di antara seluruh *endpoint* yang diuji. Pada *endpoint* ini kedua implementasi menunjukkan kinerja yang sangat baik, namun Ntex tetap unggul dibandingkan Go Fiber. Pola serupa juga terlihat pada *endpoint* POST /channel, yang merepresentasikan proses pengiriman data dari perangkat IoT ke server.

Secara umum, peningkatan jumlah pengguna simultan tidak menyebabkan penurunan *throughput* yang signifikan. Hasil ini menunjukkan bahwa sistem mampu mempertahankan kualitas layanan pada berbagai tingkat beban yang diuji.

Hasil pengujian menunjukkan bahwa Ntex unggul pada seluruh *endpoint* yang dievaluasi walau besarnya bervariasi. Pada beberapa *endpoint*, faktor seperti akses basis data atau ukuran respons kemungkinan memberikan kontribusi yang lebih besar terhadap waktu pemrosesan dibandingkan *framework* yang digunakan. Hasil ini menunjukkan implementasi berbasis Rust dan Ntex berpotensi untuk meningkatkan kapasitas layanan tanpa perubahan pada rancangan sistem maupun kebutuhan fungsional aplikasi.

Keunggulan kinerja yang ditunjukkan oleh implementasi berbasis Ntex kemungkinan berkaitan dengan karakteristik bahasa pemrograman Rust dan desain *framework* yang digunakan. Rust memungkinkan pengelolaan memori secara aman tanpa memerlukan *garbage collector* pada saat eksekusi (Matsakis & Klock, 2014). Tidak adanya *garbage collector* pada saat eksekusi menghilangkan kebutuhan proses pembersihan memori secara periodik yang pada beberapa kondisi dapat menambah *overhead* pemrosesan. Karakteristik ini berpotensi membantu server menjaga *throughput* ketika jumlah permintaan meningkat. Selain itu, Ntex dirancang untuk pengembangan layanan jaringan berkinerja tinggi dengan memanfaatkan kemampuan Rust dalam menghasilkan kode yang efisien dan mendekati performa bahasa pemrograman sistem tradisional. Rust mampu menawarkan kombinasi keamanan dan performa yang kompetitif dibandingkan berbagai bahasa pemrograman populer (Bugden dan Alahmar, 2022).

Perlu diperhatikan bahwa penelitian ini tidak dirancang untuk memisahkan pengaruh bahasa pemrograman dan *framework* secara independen. Hasil penelitian hanya menunjukkan bahwa kombinasi keduanya menghasilkan *throughput* yang lebih tinggi dibandingkan implementasi berbasis Go Fiber pada lingkungan dan skenario pengujian yang digunakan.

Selain itu, evaluasi hanya menggunakan metrik TPS sebagai indikator utama. Pengujian juga menggunakan satu konfigurasi perangkat keras dan satu jenis basis data sehingga hasilnya belum tentu dapat digeneralisasi pada lingkungan yang berbeda.

## KESIMPULAN

Penelitian ini menghasilkan implementasi server IoT berbasis bahasa pemrograman Rust dan *framework* Ntex dengan mengadopsi rancangan sistem yang telah digunakan pada penelitian sebelumnya. Hasil pengujian fungsional menunjukkan bahwa seluruh layanan yang dirancang, meliputi autentikasi pengguna, pengelolaan perangkat keras, pengelolaan *node*, dan pengelolaan data hasil pengukuran, beroperasi sesuai kebutuhan yang ditetapkan.

Evaluasi kinerja menunjukkan bahwa implementasi berbasis Ntex menghasilkan nilai TPS yang lebih tinggi dibandingkan implementasi berbasis Go Fiber pada seluruh *endpoint* yang diuji. Peningkatan kinerja terbesar terjadi pada *endpoint* GET /nodes/:id, sedangkan peningkatan terkecil terjadi pada *endpoint* GET /nodes. Implementasi berbasis Ntex menghasilkan peningkatan *throughput* rata-rata sekitar 20% dibandingkan implementasi berbasis Go Fiber.

Hasil penelitian menunjukkan bahwa kombinasi Rust dan Ntex mampu menjadi alternatif implementasi yang efektif untuk pengembangan server IoT tanpa perubahan terhadap kebutuhan fungsional maupun arsitektur sistem yang telah ada. Temuan ini menunjukkan bahwa penggantian teknologi implementasi tanpa perubahan rancangan sistem dapat meningkatkan kinerja layanan IoT secara signifikan.

Penelitian selanjutnya dapat memperluas evaluasi dengan menambahkan metrik kinerja lain, seperti latensi, penggunaan memori, dan utilisasi CPU. Selain itu, pengujian pada konfigurasi infrastruktur dan beban kerja yang lebih beragam dapat memberikan gambaran yang lebih komprehensif mengenai karakteristik implementasi berbasis Rust dan Ntex.

## UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Institut Pertanian Bogor, khususnya Program Studi Ilmu Komputer, Sekolah Sains Data, Matematika dan Informatika, serta seluruh pihak yang telah memberikan dukungan selama pelaksanaan penelitian ini.

## DAFTAR PUSTAKA

Athaullah, M. D. (2023). Pengembangan dan Peningkatan Kinerja Server IoT Berbasis REST API Menggunakan Golang Fiber. Skripsi. Institut Pertanian Bogor.

Bugden, W., & Alahmar, H. (2022). The Safety and Performance of Prominent Programming Languages. *Int. J. Soft. Eng. Knowl.* Eng. 32(05):713–744.doi:10.1142/S0218194022500231.

Ferdiansyah A. (2023). Pengembangan dan Peningkatan Kinerja Server IoT Berbasis Python dengan Framework Sanic dan Database PostgreSQL. Skripsi. Institut Pertanian Bogor.

Fielding, R. T. (2000). Architectural Styles and the Design of Network-Based Software Architectures. Doctoral Dissertation. University of California, Irvine.

Hanin, F. (2021). Pengembangan Back-end Aplikasi Server Data IoT Berbasis REST API Menggunakan Framework Falcon. Skripsi. Institut Pertanian Bogor.

Matsakis, N. D., & Klock, F. S. (2014). The Rust Language. Proceedings of the 2014 ACM SIGAda Annual Conference on High Integrity Language Technology. pp. 103–104.

Mordor Intelligence. (2026a). Indonesia IoT Market Size & Share Analysis - Growth Trends & Forecasts (2026-2031). https://www.mordorintelligence.com/industry-reports/indonesia-iot-market

Mordor Intelligence. (2026b). IoT Devices Market Size & Share Analysis - Growth Trends & Forecasts (2026 - 2031). https://www.mordorintelligence.com/industry-reports/iot-devices-market

Ntex Contributors. (2025). Ntex Documentation. https://ntex.rs/docs

Severance, C. (2012). Discovering JavaScript Object Notation. *Computer*. 45(4):6–8.doi:10.1109/MC.2012.132.