TIN: Terapan Informatika Nusantara
Vol 6, No 10, March 2026, page 1975-1988
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i10.9406

Peningkatan Efisiensi Pembaruan Aplikasi Android Tap On Bus Menggunakan Sistem OTA Update Berbasis REST API

Rohmat Julianto*, Mutaqin Akbar

Fakultas Teknologi Informasi, Program Studi Informatika, Universitas Mercu Buana Yogyakarta, Yogyakarta, Indonesia
Email: [1,*]juliantojuly@gmail.com, [2]mutaqin@mercubuana-yogya.ac.id
Email Penulis Korespondensi: juliantojuly@gmail.com

Abstrak–Pengelolaan pembaruan aplikasi pada lebih dari 400 perangkat validator Android Tap On Bus yang tersebar di berbagai rute transportasi publik di Indonesia menghadapi tantangan efisiensi dan skalabilitas yang signifikan. Proses pembaruan yang masih mengandalkan intervensi teknisi secara manual melalui media penyimpanan fisik (flashdisk) terbukti tidak efisien karena membutuhkan perjalanan ke setiap lokasi perangkat, berisiko terhadap integritas data, serta menyebabkan inkonsistensi versi antar perangkat. Penggunaan platform Mobile Device Management (MDM) komersial sebagai alternatif juga menimbulkan ketergantungan vendor dan biaya lisensi berlangganan yang tinggi. Penelitian ini bertujuan mengembangkan dan mengimplementasikan sistem Over-The-Air (OTA) Update berbasis REST API yang bersifat mandiri (in-house) untuk mengotomatisasi distribusi dan instalasi pembaruan aplikasi Android secara terpusat dan jarak jauh. Sistem dikembangkan menggunakan metode Research and Development (R&D) dengan model Waterfall, diimplementasikan menggunakan Go (Golang) untuk backend REST API dengan basis data MySQL yang dijalankan pada web server XAMPP, Kotlin dengan PackageInstaller API untuk klien Android pada perangkat Telpo T10, serta framework Vuetify untuk dashboard monitoring. Pengujian fungsional menggunakan metode Black Box Testing terhadap 19 skenario uji mencakup registrasi perangkat, pengecekan versi saat aplikasi dijalankan, validasi versi, pengunduhan paket APK, penanganan kegagalan unduhan dan kegagalan instalasi beserta pemulihannya, dan instalasi otomatis menunjukkan tingkat keberhasilan 100%. Analisis komparatif menunjukkan bahwa sistem OTA berhasil menurunkan rata-rata waktu pembaruan per perangkat dari 16 menit (manual) menjadi sekitar 1,2 menit (otomatis), setara peningkatan efisiensi sebesar 92,5%, sekaligus mengeliminasi kebutuhan biaya perjalanan teknisi. Sistem juga terbukti menjaga integritas data transaksi selama proses pembaruan dan mencatat seluruh aktivitas pembaruan pada basis data secara terpusat, sehingga dinyatakan layak diterapkan pada lingkungan operasional transportasi publik berskala besar.

Kata Kunci: *Distribusi APK Jarak Jauh; Manajemen Armada; Over-The-Air (OTA) Update; REST API; Validator Android*

Abstract−Managing application updates for more than 400 Android validator devices of the Tap On Bus system, deployed across various public transportation routes in Indonesia, presents significant challenges in efficiency and scalability. The existing manual update process, relying on physical storage media (flash drives) and direct technician intervention at each device location, has proven inefficient, poses risks to data integrity, and causes version inconsistencies across devices. Using commercial Mobile Device Management (MDM) platforms as an alternative also introduces vendor dependency and high recurring license costs. This research aims to develop and implement an independent (in-house) Over-The-Air (OTA) Update system based on REST API to automate the centralized and remote distribution and installation of Android application updates. The system was developed using the Research and Development (R&D) method with the Waterfall model, implemented using Go (Golang) for the REST API backend with a MySQL database running on the XAMPP web server, Kotlin with the PackageInstaller API for the Android client on Telpo T10 devices, and the Vuetify framework for the monitoring dashboard. Functional testing using the Black Box Testing method across 19 test scenarios covering device registration, startup-triggered update checking, version validation, APK package download, download-failure and installation-failure handling with recovery, and automatic installation yielded a 100% success rate. Comparative analysis demonstrated that the OTA system successfully reduced the average update time per device from 16 minutes (manual) to approximately 1.2 minutes (automatic), equivalent to a 92.5% efficiency improvement, while eliminating technician travel costs. The system also demonstrated the ability to maintain transaction data integrity during updates and centrally log all update activities in the database, confirming its readiness for deployment in large-scale public transportation environments.

Keywords: Remote APK Distribution; Fleet Management; Over-The-Air (OTA) Update; REST API; Android Validator

1. PENDAHULUAN

Transformasi digital pada sektor transportasi publik telah mendorong adopsi luas teknologi berbasis mobile untuk meningkatkan efisiensi operasional dan kenyamanan pengguna. Dalam konteks ini, sistem pembayaran non-tunai berbasis aplikasi Android menjadi salah satu solusi utama yang diimplementasikan pada armada bus di berbagai kota di Indonesia. Perangkat Android, seperti Telpo T10, digunakan sebagai terminal validasi penumpang karena menawarkan fleksibilitas tinggi, kemudahan integrasi dengan berbagai perangkat keras, serta dukungan ekosistem pengembangan yang matang (Falderika dkk., 2021). Namun, adopsi teknologi ini membawa tantangan tersendiri dalam aspek pemeliharaan perangkat lunak (software maintenance), khususnya manajemen pembaruan aplikasi secara berkala untuk memperbaiki bug, menambah fitur baru, serta menyesuaikan regulasi keuangan digital yang terus berkembang (Syahputra dkk., 2025).

Penelitian ini difokuskan pada aplikasi Tap On Bus, yaitu sistem validasi pembayaran yang dioperasikan pada lebih dari 400 perangkat Android yang tersebar di berbagai rute dan pool kendaraan. Skala operasional sebesar ini menciptakan kompleksitas tersendiri dalam manajemen aset perangkat lunak. Kondisi eksisting di lokasi penelitian menunjukkan bahwa proses pembaruan aplikasi masih dilakukan secara manual oleh teknisi yang harus melakukan perjalanan ke lokasi perangkat, menyalin berkas instalasi (APK) menggunakan flashdisk, dan melakukan instalasi satu per satu pada setiap perangkat validator. Meskipun metode ini efektif untuk jumlah perangkat yang kecil,

TIN: Terapan Informatika Nusantara
Vol 6, No 10, March 2026, page 1975-1988
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i10.9406

implementasinya pada ratusan perangkat yang tersebar (distributed devices) terbukti tidak efisien, memakan waktu operasional teknisi yang signifikan, dan berisiko tinggi terhadap integritas data akibat inkonsistensi versi aplikasi yang terpasang (Han dkk., 2022). Lebih jauh, metode manual ini rentan terhadap human error dan kerusakan fisik media penyimpanan yang berulang, yang dapat mengganggu kontinuitas layanan transportasi (Rastogi, 2024).

Untuk mengatasi permasalahan distribusi pembaruan pada perangkat terdistribusi skala besar, terdapat beberapa pendekatan solusi yang telah tersedia di industri. Pertama, penggunaan toko aplikasi resmi seperti Google Play Store. Namun, untuk aplikasi internal enterprise atau sistem khusus seperti validator bus, Play Store publik tidak ideal karena kendala privasi data, proses review yang memakan waktu, serta kebutuhan kontrol versi yang ketat terhadap perangkat khusus. Kedua, platform Mobile Device Management (MDM) komersial seperti VMware Workspace ONE atau Microsoft Intune yang memungkinkan pengelolaan dan distribusi aplikasi secara terpusat. Akan tetapi, implementasi MDM komersial seringkali menghadapi kendala biaya lisensi berlangganan (recurring cost) yang tinggi untuk skala ratusan perangkat, serta potensi vendor lock-in yang membatasi fleksibilitas kustomisasi alur kerja pembaruan sesuai kebutuhan spesifik bisnis (Gunawan dkk., 2025). Keterbatasan inilah yang mendasari kebutuhan akan solusi alternatif yang bersifat mandiri (in-house), biaya operasional lebih rendah, namun tetap mampu menjawab tantangan distribusi pembaruan secara massal.

Mekanisme Over-The-Air (OTA) Update merupakan solusi teknis yang memungkinkan perangkat mengunduh dan menginstal pembaruan perangkat lunak secara nirkabel melalui jaringan internet tanpa memerlukan intervensi fisik. Penelitian terkait implementasi OTA telah banyak dilakukan, khususnya pada ranah Internet of Things (IoT). Ikhsani dan Budi (2024) menunjukkan bahwa OTA mampu memperbarui firmware pada mikrokontroler ESP8266 secara massal dengan waktu rata-rata 28,845 detik, dan menilai metode ini efisien untuk sistem IoT berskala besar. Wijaya dkk. (2025) berhasil mengimplementasikan sistem OTA pada perangkat Raspberry Pi berbasis GitHub Repository dengan rata-rata waktu pembaruan 18,069 detik dan downtime 5,013 detik. Rahmawan dkk. (2025) mendemonstrasikan efektivitas OTA simultan pada banyak perangkat ESP32 menggunakan pustaka AsyncElegantOTA, dengan waktu unggah rata-rata 44-45 detik. Cahyono dkk. (2023) mengembangkan gateway Raspberry Pi untuk mendukung OTA pada perangkat Wireless Sensor Network (WSN) melalui Bluetooth Low Energy (BLE). Ramadhan dkk. (2022) merancang sistem auto-config OTA pada perangkat IoT menggunakan protokol HTTP dan konsep REST API.

Meskipun literatur tersebut memberikan landasan konseptual yang kuat mengenai efektivitas mekanisme OTA, terdapat kesenjangan (research gap) yang fundamental antara domain IoT dan domain aplikasi Android. Penelitian-penelitian tersebut berfokus pada firmware flashing atau penulisan ulang memori tingkat rendah (low-level memory rewriting) pada mikrokontroler. Sebaliknya, pembaruan aplikasi Android (APK) beroperasi pada level sistem operasi yang lebih tinggi dan menghadirkan tantangan teknis yang berbeda secara substansial. Pertama, penanganan runtime permissions pada Android versi modern (Android 8.0+) yang menerapkan kebijakan keamanan semakin ketat terhadap instalasi aplikasi dari sumber tidak dikenal (unknown sources), memerlukan penanganan teknis spesifik melalui PackageInstaller API. Kedua, manajemen proses instalasi di latar belakang tanpa interaksi pengguna (silent installation) pada perangkat khusus (rugged device) yang beroperasi terus-menerus. Ketiga, perlunya menjaga integritas database lokal aplikasi selama proses migrasi versi agar data transaksi tidak hilang atau korup (Ramadhan dkk., 2022). Keempat, penanganan kegagalan unduhan atau instalasi yang andal sehingga proses pembaruan dapat diulang dengan aman tanpa merusak aplikasi yang sedang terpasang. Kesenjangan ini menunjukkan bahwa hasil penelitian OTA berbasis IoT tidak dapat langsung diterapkan pada konteks pembaruan aplikasi Android, sehingga diperlukan studi implementasi khusus.

Li dkk. (2024) dalam kajian komprehensif mereka mengenai OTA untuk kendaraan cerdas (intelligent connected vehicles) menyoroti bahwa keamanan transmisi data, validasi integritas paket, dan mekanisme rollback merupakan aspek kritis dalam sistem OTA skala produksi. Pattinama dan Susanti (2023) menegaskan bahwa REST API menawarkan protokol komunikasi yang stateless, ringan, dan platform-agnostic, sehingga memudahkan integrasi antara server backend dengan klien Android tanpa keterikatan pada vendor tertentu. Muharam dan Hidayat (2024) menunjukkan bahwa Golang memiliki keunggulan dalam menangani concurrency tinggi dari banyak klien secara bersamaan, yang relevan untuk skenario ratusan perangkat yang melakukan pengecekan versi secara simultan.

Penegasan perbedaan ini penting karena mayoritas penelitian OTA terdahulu beroperasi pada perangkat IoT berbasis mikrokontroler (ESP8266, ESP32, Raspberry Pi) yang memperbarui firmware melalui penulisan langsung ke memori flash. Pada konteks tersebut, perangkat memiliki kendali penuh atas proses booting dan tidak menghadapi lapisan abstraksi sistem operasi. Sebaliknya, pembaruan aplikasi Android (APK) yang menjadi fokus penelitian ini harus melewati sandbox keamanan Android, tunduk pada kebijakan signature verification yang mengharuskan APK baru ditandatangani dengan kunci yang sama dengan versi terpasang, serta memerlukan izin sistem khusus untuk menginstal tanpa interaksi pengguna. Tantangan inilah yang membuat penerapan OTA pada APK Android tidak dapat sekadar mengadopsi metode flashing firmware IoT, melainkan memerlukan pemanfaatan API tingkat sistem operasi seperti PackageInstaller serta penanganan proses pengunduhan dan instalasi yang sesuai dengan batasan Android modern.

Berdasarkan analisis kesenjangan penelitian tersebut, penelitian ini bertujuan untuk mengimplementasikan sistem OTA Update berbasis REST API yang dirancang khusus untuk konteks aplikasi Android pada armada transportasi publik berskala besar, dengan tujuan utama meningkatkan efisiensi waktu dan biaya operasional pembaruan dibandingkan metode manual. Sistem ini dirancang untuk mengatasi keterbatasan solusi MDM komersial (biaya, vendor lock-in) sekaligus menjawab tantangan teknis spesifik pembaruan APK Android yang tidak dicakup oleh penelitian OTA berbasis IoT sebelumnya.

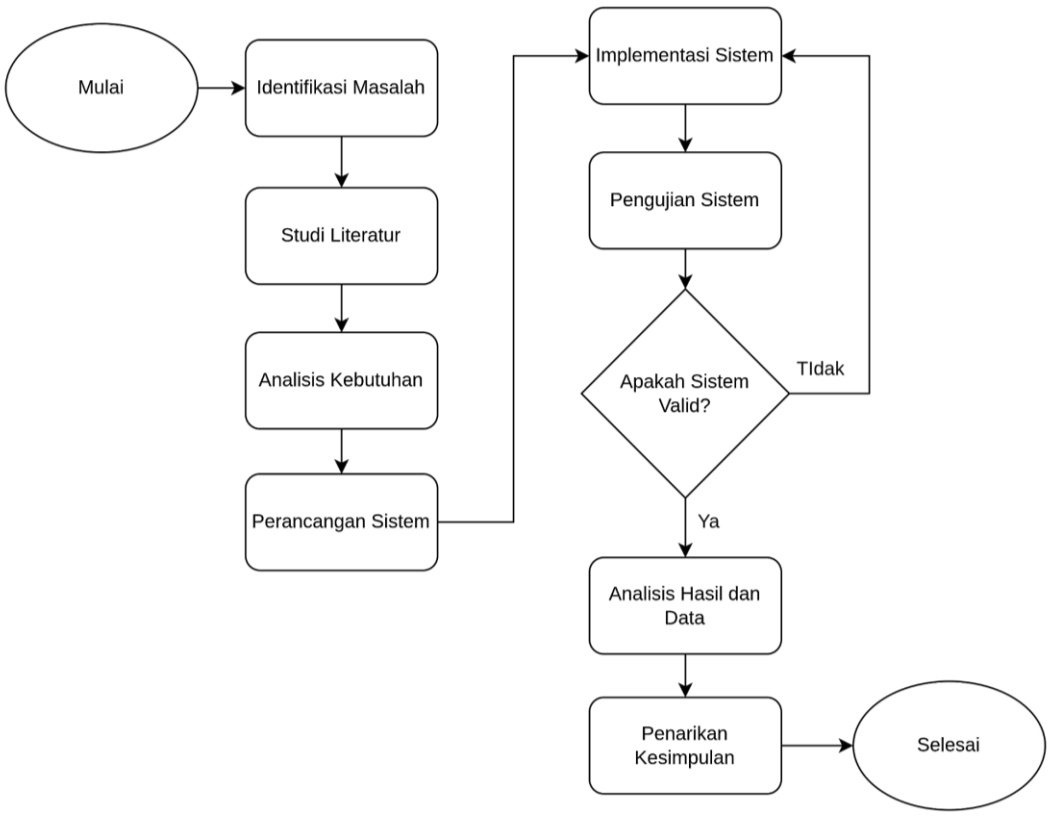
TIN: Terapan Informatika Nusantara
Vol 6, No 10, March 2026, page 1975-1988
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i10.9406

Kontribusi penelitian ini mencakup tiga aspek utama: (1) rancangan arsitektur sistem OTA mandiri berbasis REST API yang mampu melayani ratusan perangkat Android secara bersamaan dengan backend Golang berkinerja tinggi; (2) mekanisme pengecekan versi otomatis saat aplikasi dijalankan (startup-triggered) dan instalasi otomatis (silent installation) yang aman tanpa interaksi pengguna menggunakan PackageInstaller API; serta (3) bukti empiris kuantitatif mengenai peningkatan efisiensi waktu operasional dibandingkan metode manual tradisional, yang dapat menjadi acuan bagi pengembangan sistem serupa di sektor transportasi, logistik, atau industri lain yang mengelola armada perangkat Android dalam jumlah besar.

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Penelitian ini menggunakan metode Research and Development (R&D) dengan pendekatan rekayasa perangkat lunak (software engineering). Metode R&D merupakan metode penelitian yang digunakan untuk menghasilkan produk tertentu dan menguji keefektifan produk tersebut, di mana produk yang dihasilkan dapat berupa perangkat lunak (Sugiyono, 2022). Metode ini dipilih karena penelitian ini tidak hanya bertujuan mengidentifikasi masalah, tetapi juga menghasilkan produk konkret berupa sistem OTA Update yang dapat diuji dan divalidasi kinerjanya. Proses pengembangan sistem mengadopsi model Waterfall (Ningsih & Nurfauziah, 2023). Pemilihan model Waterfall pada konteks pengembangan sistem OTA untuk industri transportasi ini didasari pertimbangan praktis: kebutuhan sistem telah terdefinisi secara pasti dan stabil sejak awal—yaitu mendistribusikan APK ke ratusan perangkat validator yang seragam (Telpo T10)—sehingga risiko perubahan kebutuhan di tengah pengembangan sangat kecil. Selain itu, sistem ini bersifat mission-critical bagi operasional pembayaran transportasi sehingga setiap tahap (analisis, desain, implementasi, pengujian) perlu diselesaikan dan diverifikasi secara tuntas sebelum tahap berikutnya, guna meminimalkan risiko kegagalan pada lingkungan produksi yang melayani penumpang secara langsung (Anis dkk., 2023).



Gambar 1. Diagram Alir Penelitian

Adapun penjelasan detail dari setiap tahapan dalam diagram tersebut adalah sebagai berikut:

2.1.1 Identifikasi Masalah dan Studi Literatur

Tahap identifikasi masalah dilakukan melalui observasi langsung ke lokasi operasional armada bus dan wawancara mendalam dengan teknisi serta manajemen IT perusahaan. Observasi difokuskan pada pengukuran waktu aktual proses pembaruan manual, identifikasi titik-titik rawan kegagalan (failure points), serta estimasi biaya operasional teknisi per siklus pembaruan. Hasil observasi menunjukkan bahwa satu siklus pembaruan pada satu perangkat membutuhkan rata-rata 16 menit, mencakup waktu perjalanan, penyalinan APK, instalasi, dan verifikasi. Secara rinci, prosedur pembaruan

TIN: Terapan Informatika Nusantara
Vol 6, No 10, March 2026, page 1975-1988
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i10.9406

manual pada setiap perangkat terdiri atas beberapa langkah berurutan: (1) teknisi membuka panel dashboard pada unit bus untuk memperoleh akses ke port USB perangkat validator; (2) memasang (mencolokkan) flashdisk yang telah berisi berkas APK terbaru ke port tersebut; (3) membuka aplikasi file manager pada perangkat untuk menavigasi dan mengakses berkas APK dari dalam flashdisk; serta (4) menjalankan instalasi paket pembaruan secara manual dan menunggu hingga proses selesai. Rangkaian langkah fisik inilah—mulai dari membuka panel hingga menunggu instalasi—yang menyebabkan satu perangkat membutuhkan waktu relatif lama, di samping ketergantungan penuh pada kehadiran teknisi di lokasi. Pembaruan manual selama ini dilakukan secara bertahap per pool bus oleh 2–3 orang teknisi, dengan setiap pool menampung sekitar 10–15 perangkat. Sebagai gambaran, satu pool berisi 12 perangkat yang dikerjakan oleh 3 teknisi secara paralel tetap membutuhkan sekitar 64 menit (4 perangkat per teknisi × 16 menit) untuk satu siklus pembaruan, belum termasuk waktu dan biaya perjalanan menuju setiap pool. Dengan lebih dari 400 perangkat yang tersebar di banyak pool, beban kerja kumulatif teknisi menjadi sangat besar dan tidak efisien.

Berdasarkan hasil identifikasi tersebut, ditetapkan kriteria teknis minimal sebagai batasan (scope) sistem agar pengembangan terarah, yaitu: (1) sistem hanya menangani perangkat Android dengan sistem operasi minimal versi 8.0 (API level 26) sesuai spesifikasi Telpo T10; (2) ukuran berkas APK yang didistribusikan sekitar 16 MB; (3) proses pembaruan harus berlangsung tanpa intervensi pengguna (silent) dan tanpa menghapus data transaksi yang tersimpan; (4) sistem mengasumsikan ketersediaan koneksi internet melalui jaringan seluler 4G pada perangkat; serta (5) keputusan ketersediaan pembaruan ditentukan murni berdasarkan perbandingan nomor versi antara perangkat dan server, bukan berdasarkan validasi konten berkas. Studi literatur dilakukan secara sistematis dengan menelaah jurnal ilmiah dari basis data Scopus, IEEE Xplore, dan Google Scholar menggunakan kata kunci 'Over-The-Air update Android', 'REST API mobile update', 'remote APK deployment', dan 'fleet device management'. Fokus kajian meliputi arsitektur REST API (Pattinama & Susanti, 2023), mekanisme silent installation pada Android menggunakan PackageInstaller API, analisis keamanan sistem OTA (Li dkk., 2024), serta studi komparatif terhadap penelitian OTA terdahulu pada ranah IoT (Ikhsani & Budi, 2024; Wijaya dkk., 2025; Rahmawan dkk., 2025). Studi ini menjadi dasar pemilihan teknologi dan strategi implementasi.

2.1.2 Analisa Kebutuhan Sistem

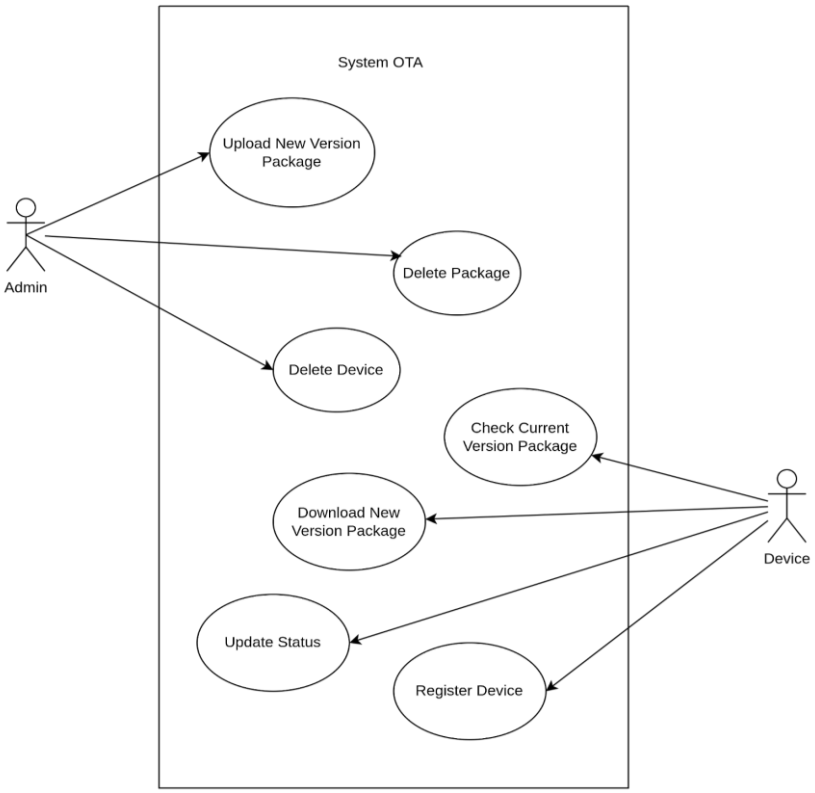
Tahap analisis kebutuhan menerjemahkan hasil observasi dan studi literatur menjadi spesifikasi teknis yang terstruktur. Analisis dibagi menjadi dua aspek utama. Kebutuhan fungsional sistem mencakup: (1) registrasi perangkat otomatis berdasarkan serial number unik Telpo T10; (2) pengecekan versi aplikasi yang dipicu (triggered) secara otomatis pada saat aplikasi Tap On Bus dijalankan/dimulai (startup), dengan membandingkan versi yang terpasang pada perangkat dan versi terbaru pada server melalui REST API; (3) pengunduhan paket APK apabila tersedia versi yang lebih baru; (4) instalasi otomatis (silent installation) menggunakan PackageInstaller API tanpa interaksi pengguna; (5) pelaporan status pembaruan (sukses/gagal) ke server; serta (6) dashboard monitoring terpusat untuk admin.

Kebutuhan non-fungsional mencakup: (1) komunikasi antara perangkat dan server melalui protokol HTTP berbasis REST API; (2) keandalan sistem dengan mekanisme retry otomatis untuk mengulang pembaruan apabila terjadi kegagalan unduhan; (3) kompatibilitas dengan sistem operasi Android versi 8.0 (API level 26) ke atas yang digunakan oleh perangkat Telpo T10; serta (4) kemampuan backend menangani concurrency dari ratusan perangkat secara bersamaan tanpa degradasi performa.

2.1.3 Perancangan Sistem

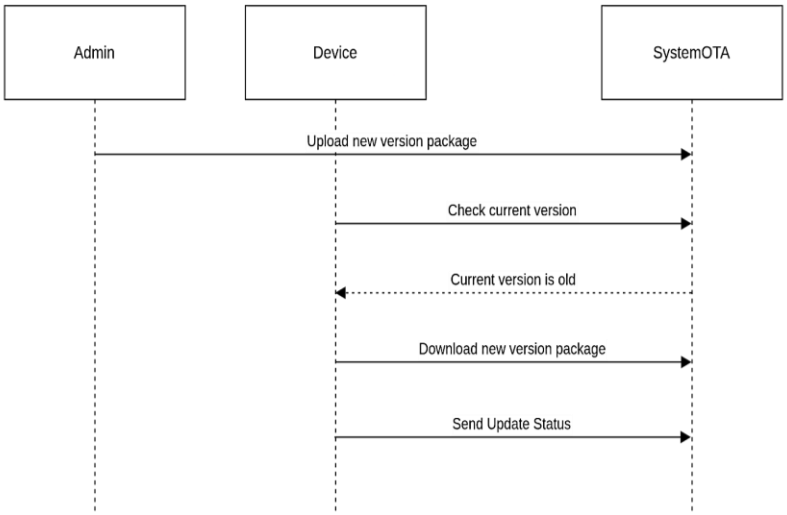
Tahap perancangan sistem menghasilkan rancangan teknis sebagai solusi dari permasalahan yang telah diidentifikasi (Saputra dkk., 2023). Perancangan meliputi desain arsitektur sistem OTA Update berbasis client-server, perancangan basis data relasional untuk pengelolaan versi aplikasi dan perangkat, serta pemodelan sistem menggunakan Unified Modeling Language (UML). Arsitektur sistem terdiri dari tiga komponen utama: (1) OTA Server berbasis Golang yang mengekspos endpoint REST API untuk registrasi perangkat, pengecekan versi, dan pengunduhan paket APK; (2) Android Client berbasis Kotlin yang melakukan pengecekan versi secara otomatis saat aplikasi dijalankan di perangkat Telpo T10; dan (3) Dashboard Admin berbasis framework Vuetify untuk pengelolaan paket dan monitoring status perangkat. Gambar 2 menampilkan use case diagram sistem OTA Update yang memodelkan logika bisnis interaksi dua aktor (Aurellia, 2025). Aktor Admin bertanggung jawab mengunggah paket APK versi terbaru, menghapus perangkat tidak aktif, dan menghapus paket lama melalui dashboard. Sementara itu, aktor Perangkat Tap On Bus bertindak sebagai aktor non-manusia (automated actor) yang secara mandiri melakukan registrasi otomatis, pengecekan versi saat aplikasi dijalankan, pengunduhan paket, dan pelaporan status pembaruan tanpa keterlibatan operator. Pemisahan peran ini menjadi inti logika bisnis sistem: beban orkestrasi pembaruan dipindahkan dari teknisi (manual) menjadi inisiatif perangkat itu sendiri (otomatis).

TIN: Terapan Informatika Nusantara
Vol 6, No 10, March 2026, page 1975-1988
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i10.9406



Gambar 2. Use Case Diagram

Gambar 3 menyajikan sequence diagram yang menggambarkan urutan interaksi antara Admin, Perangkat Tap On Bus, dan Sistem OTA selama proses pembaruan aplikasi berlangsung. Diagram ini menyoroti alur kritis sistem berdasarkan urutan waktu (Nabila dkk., 2021): proses dimulai dari Admin mengunggah APK ke server, lalu perangkat memicu pengecekan versi pada saat aplikasi dijalankan, server merespons dengan informasi pembaruan hanya bila nomor versi perangkat lebih lama dari versi server, perangkat kemudian mengunduh dan menginstal APK secara otomatis, dan akhirnya melaporkan status keberhasilan kembali ke server. Titik keputusan (decision point) perbandingan versi di sisi server inilah yang memastikan perangkat tidak mengunduh berkas secara berulang ketika sudah berada pada versi terkini, sehingga menghemat bandwidth pada skala ratusan perangkat.

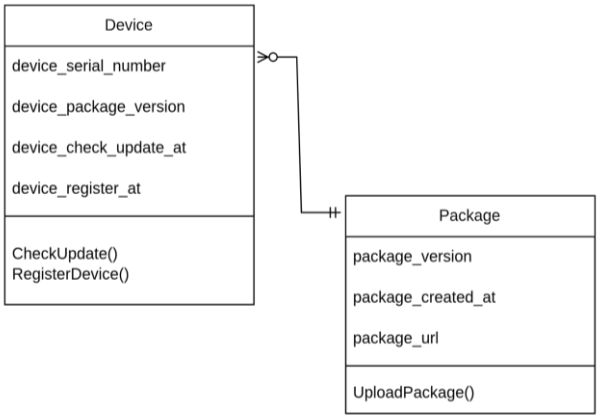


Gambar 3. Sequence Diagram

Gambar 4 memperlihatkan class diagram yang terdiri dari dua entitas utama: Device dan Package. Class Device merepresentasikan perangkat Tap On Bus terdaftar dengan atribut device_serial_number, device_package_version, device_check_update_at, dan device_register_at, dilengkapi method CheckUpdate() dan RegisterDevice(). Class Package merepresentasikan paket APK yang tersedia di server dengan atribut package_version, package_created_at, dan package_url, dilengkapi method UploadPackage(). Relasi antara Device dan Package bersifat asosiasi many-to-one untuk versi aktif, di mana satu Package dapat menjadi target pembaruan bagi banyak Device, sementara tabel histori terpisah menyimpan riwayat pembaruan setiap perangkat (Susanto, 2024). Struktur ini dirancang agar penambahan satu

TIN: Terapan Informatika Nusantara
Vol 6, No 10, March 2026, page 1975-1988
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i10.9406

paket APK baru oleh Admin secara otomatis menjadi acuan versi terbaru bagi seluruh 400+ perangkat tanpa perlu konfigurasi per perangkat, yang merupakan kunci skalabilitas sistem.



Gambar 4. Class Diagram

2.2 Implementasi Sistem

Implementasi sistem dilakukan berdasarkan rancangan yang telah dibuat dengan stack teknologi yang dipilih secara selektif sesuai kebutuhan teknis. Backend server dibangun menggunakan Go (Golang) karena kemampuannya menangani concurrency tinggi dari ratusan perangkat secara bersamaan melalui goroutine, dengan overhead memori yang sangat rendah dibandingkan bahasa lain (Muharam & Hidayat, 2024). Server mengekspos endpoint REST API dan menyimpan metadata paket serta perangkat dalam basis data MySQL yang dikelola melalui web server XAMPP, sementara berkas APK disimpan pada penyimpanan server. Penggunaan XAMPP sebagai lingkungan basis data dipilih karena bersifat gratis, mudah dikonfigurasi, dan menyediakan paket lengkap Apache dan MySQL yang memudahkan proses pengembangan serta pengelolaan data secara terpusat. Logika inti penentuan pembaruan ditangani di sisi backend melalui mekanisme perbandingan versi: ketika perangkat mengirimkan nomor versi aplikasi yang sedang terpasang, server membandingkannya dengan nomor versi paket terbaru yang tersedia pada basis data. Apabila versi pada perangkat lebih lama dari versi terbaru di server, backend mengembalikan respons berisi URL unduhan paket; sebaliknya, apabila versi sudah sama, server mengembalikan status bahwa perangkat telah terbaru sehingga tidak ada proses unduhan yang dilakukan.

Pada sisi klien, aplikasi Tap On Bus berbasis Android dikembangkan menggunakan Kotlin. Mekanisme pemicuan pembaruan diimplementasikan pada proses startup aplikasi: setiap kali aplikasi Tap On Bus dijalankan, modul pengecekan versi langsung dieksekusi untuk mengambil nomor versi aplikasi yang sedang terpasang pada perangkat, lalu membandingkannya dengan nomor versi terbaru yang tersedia di server melalui pemanggilan REST API. Pendekatan startup-triggered ini dipilih karena sesuai dengan pola operasional armada bus yang tidak berlangsung 24 jam—aplikasi validator dijalankan kembali setiap kali armada mulai beroperasi—sehingga momen startup menjadi titik pengecekan yang alami, efisien, dan andal tanpa memerlukan layanan penjadwalan berkala (periodic scheduling) maupun pemantauan latar belakang yang terus-menerus membebani perangkat. Apabila hasil perbandingan menunjukkan versi server lebih baru, aplikasi mengunduh paket APK dan melanjutkan ke proses instalasi; apabila versi sudah sama, proses dihentikan. Proses instalasi otomatis tanpa interaksi pengguna (silent installation) diimplementasikan menggunakan PackageInstaller API, yang memerlukan izin INSTALL_PACKAGES pada perangkat dengan mode device owner atau melalui konfigurasi khusus perangkat Telpo T10 sebagai dedicated device. Apabila pengunduhan gagal di tengah proses, berkas parsial dibuang dan pembaruan diulang dari awal pada pemicuan startup berikutnya. Dashboard monitoring dibangun menggunakan framework Vuetify yang berbasis Vue.js, yang berkomunikasi dengan backend melalui REST API yang sama untuk menampilkan data paket dan status perangkat secara real-time.

2.3 Pengujian Sistem

Pengujian sistem dilakukan dalam dua tahap. Pertama, pengujian fungsional menggunakan metode Black Box Testing untuk memverifikasi bahwa setiap fungsi sistem berjalan sesuai spesifikasi yang telah ditetapkan. Pengujian black box adalah metode pengujian perangkat lunak yang memeriksa spesifikasi tanpa memperhatikan kode internal (Raihan & Voutama, 2023). Pengujian mencakup 19 skenario yang dibagi dalam dua kategori: pengujian registrasi dan manajemen perangkat (6 skenario) serta pengujian alur OTA Update end-to-end (13 skenario), termasuk skenario kegagalan unduhan, kegagalan instalasi, dan pemulihannya untuk menguji ketangguhan sistem. Kedua, pengujian komparatif performa dilakukan untuk mengukur efisiensi sistem OTA dibandingkan metode manual. Pengukuran dilakukan terhadap waktu rata-rata per perangkat pada kedua metode, mencakup seluruh tahapan dari persiapan hingga verifikasi selesainya pembaruan. Seluruh pengujian dilakukan pada kondisi operasional sebenarnya, yaitu ketika perangkat berada di pool bus dalam keadaan diam dengan koneksi jaringan seluler 4G (SIM operator komersial) yang digunakan di lapangan. Data dikumpulkan dari 10 sampel pengujian untuk masing-masing metode. Hasil pengujian dianalisis

TIN: Terapan Informatika Nusantara
Vol 6, No 10, March 2026, page 1975-1988
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i10.9406

menggunakan analisis deskriptif untuk menghitung persentase peningkatan efisiensi waktu antara metode manual dan metode OTA.

Beberapa rumus berikut digunakan sebagai dasar perhitungan nilai-nilai pada tabel pengujian. Pertama, waktu rata-rata suatu proses dihitung sebagai rata-rata aritmatika dari seluruh sampel pengukuran, sebagaimana Persamaan (1):

$$T = (1/n) \times \Sigma\, T_i \quad (1)$$

Dengan T adalah waktu rata-rata (detik), T_i adalah waktu pada pengukuran ke-i, dan n adalah jumlah sampel pengujian (n = 10). Kedua, throughput efektif pengunduhan (Tabel 4) dihitung dari ukuran berkas APK dibagi waktu pengunduhan, dengan konversi satuan dari megabyte (MB) ke megabit (Mb), sebagaimana Persamaan (2):

$$\text{Throughput (Mbps)} = (S \times 8) / Td \quad (2)$$

Dengan S adalah ukuran berkas APK (MB), faktor 8 mengkonversi byte ke bit, dan Td adalah waktu pengunduhan (detik). Sebagai contoh, untuk APK 16 MB dengan waktu unduh 30 detik pada jaringan 4G diperoleh (16 × 8) / 30 = 128/30 ≈ 4,3 Mbps. Ketiga, total waktu proses OTA per perangkat (Tabel 5) merupakan penjumlahan seluruh komponen waktu, yaitu waktu pemicuan dan pengecekan versi, waktu pengunduhan, waktu instalasi, serta waktu verifikasi dan pencatatan log, sebagaimana Persamaan (3):

$$TOTA = Tp + Td + Ti + Tv \quad (3)$$

Dengan $TOTA$ adalah total waktu OTA (detik), Tp adalah waktu pemicuan dan pengecekan versi, Td adalah waktu pengunduhan, Ti adalah waktu instalasi, dan Tv adalah waktu verifikasi dan pencatatan log. Contohnya, pada jaringan 4G diperoleh $TOTA$ = 12 + 30 + 24 + 6 = 72 detik. Keempat, persentase peningkatan efisiensi waktu (Tabel 3) dihitung dari selisih waktu metode manual dan metode OTA terhadap waktu metode manual, sebagaimana Persamaan (4):

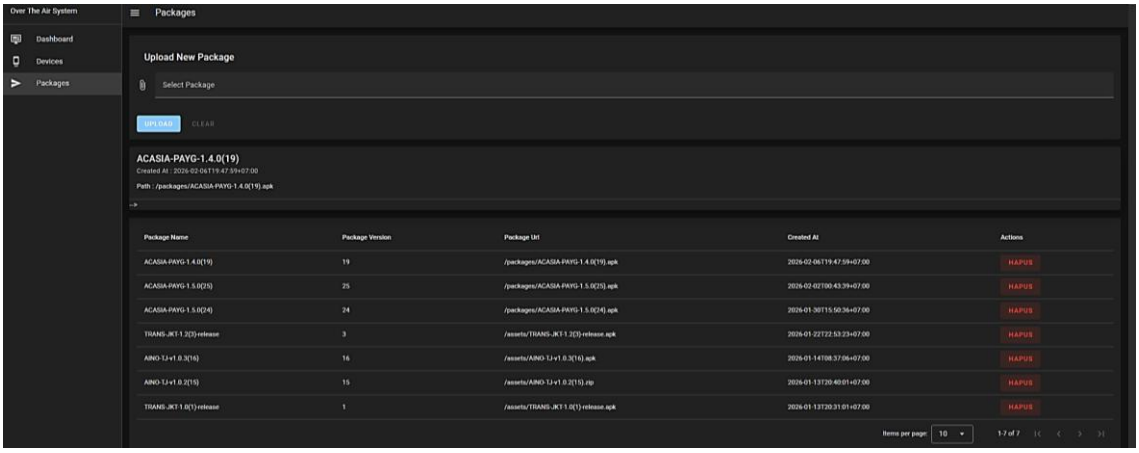
$$\text{Efisiensi (\%)} = ((Tmanual - TOTA) / Tmanual) \times 100\% \quad (4)$$

Dengan $Tmanual$ adalah waktu rata-rata metode manual dan $TOTA$ adalah waktu rata-rata metode OTA. Berdasarkan data Tabel 3 diperoleh ((16,0 − 1,2) / 16,0) × 100% = (14,8/16,0) × 100% = 92,5%.

3. HASIL DAN PEMBAHASAN

3.1 Hasil Implementasi pada Sistem

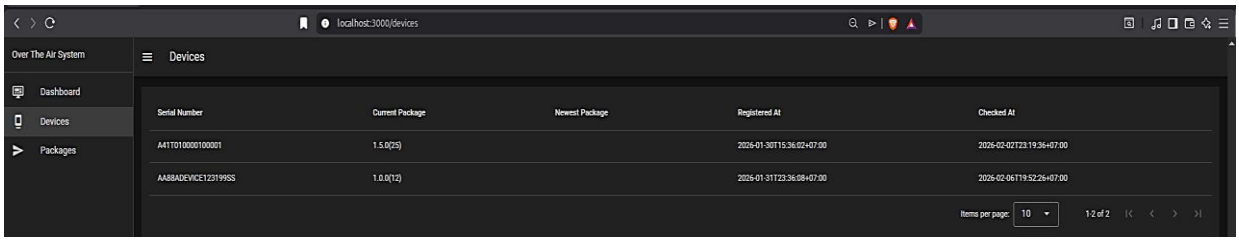
Sistem OTA Update berhasil diimplementasikan dan dioperasikan dengan backend REST API berbasis Golang serta basis data MySQL yang dijalankan pada web server XAMPP. Backend Golang dikompilasi menjadi binary yang ringan untuk melayani permintaan dari perangkat, sementara XAMPP menyediakan layanan Apache dan MySQL sebagai lingkungan penyimpanan dan pengelolaan data. REST API yang dikembangkan mengekspos tiga kelompok endpoint utama, yaitu registrasi perangkat, pengecekan versi, dan distribusi paket aplikasi.



Gambar 5. Tampilan Halaman Packages pada Dashboard OTA Update

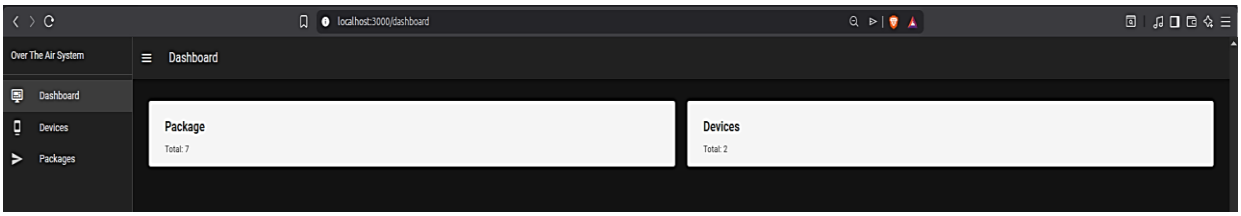
Gambar 6 menampilkan halaman Devices yang menyajikan daftar seluruh perangkat yang telah terdaftar dalam sistem beserta status pembaruannya. Informasi yang ditampilkan meliputi nomor seri perangkat (serial number), versi aplikasi yang sedang terpasang (current version), versi paket terbaru yang tersedia di server (latest version), status pembaruan (up-to-date atau needs update), waktu registrasi, dan waktu terakhir perangkat melakukan pengecekan. Halaman ini memungkinkan admin memantau konsistensi versi di seluruh armada perangkat secara real-time.

TIN: Terapan Informatika Nusantara
Vol 6, No 10, March 2026, page 1975-1988
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i10.9406



Gambar 6. Tampilan Halaman Devices pada Dashboard OTA Update

Gambar 7 menampilkan halaman Dashboard yang berfungsi sebagai ringkasan informasi utama untuk pemantauan kondisi sistem secara keseluruhan. Halaman ini menampilkan jumlah total paket aplikasi yang tersedia, jumlah total perangkat terdaftar, jumlah perangkat yang sudah menggunakan versi terbaru, dan jumlah perangkat yang masih perlu diperbarui.

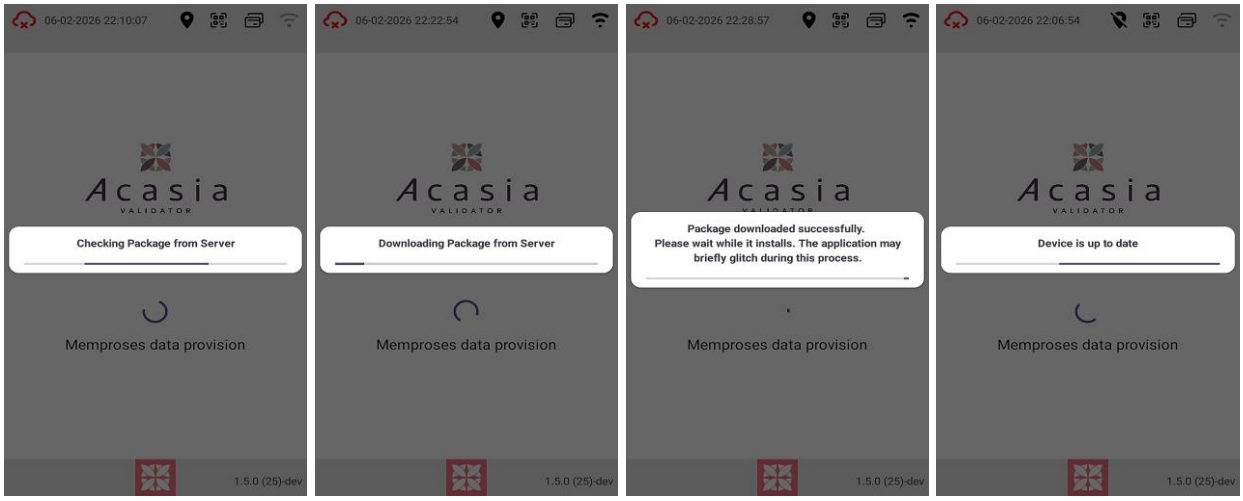


Gambar 7. Tampilan Halaman Dashboard Monitoring Sistem OTA Update

Dari sisi efektivitas pengelolaan armada, ketiga halaman pada Gambar 5, Gambar 6, dan Gambar 7 secara kolektif menjawab persoalan inti yang sebelumnya tidak terselesaikan oleh metode manual, yaitu visibilitas dan kontrol versi terpusat atas 400+ perangkat. Pada metode manual menggunakan flashdisk, tidak terdapat satu pun titik pemantauan yang memungkinkan teknisi mengetahui versi aplikasi yang terpasang pada setiap perangkat secara real-time, sehingga inkonsistensi versi sulit dideteksi hingga terjadi gangguan. Dengan dashboard ini, status konsistensi versi seluruh armada dapat dipantau dalam satu tampilan: kolom current version dan latest version pada halaman Devices (Gambar 6) memungkinkan identifikasi instan perangkat yang tertinggal versi, sementara ringkasan agregat pada halaman Dashboard (Gambar 7) memberikan indikator kesehatan armada secara keseluruhan. Mekanisme unggah satu paket pada halaman Packages (Gambar 5) yang otomatis menjadi acuan versi terbaru bagi seluruh perangkat membuktikan bahwa upaya pengelolaan tidak meningkat secara linear seiring bertambahnya jumlah perangkat—satu aksi unggah berlaku untuk 400 perangkat maupun 4.000 perangkat—yang merupakan indikator skalabilitas penting bagi pertumbuhan armada di masa depan.

3.2 Hasil Implementasi pada Aplikasi Android Tap On Bus

Gambar 8 menampilkan tampilan antarmuka aplikasi Android Tap On Bus selama proses OTA Update berlangsung. Implementasi pada sisi klien berhasil menyelesaikan seluruh tantangan teknis yang diidentifikasi dalam analisis kebutuhan.



(a) (b) (c) (d)

Gambar 8. Tampilan Aplikasi Android Tap On Bus Terdiri Dari (a) Cek Pembaruan dari Server, (b) Pengunduhan *Package*, (c) Pengunduhan *Package* Selesai, (d) Device sudah terupdate

TIN: Terapan Informatika Nusantara
Vol 6, No 10, March 2026, page 1975-1988
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i10.9406

Gambar 8a menampilkan kondisi saat aplikasi melakukan pengecekan versi ke server OTA melalui endpoint REST API GET /api/v1/check-update. Sistem menampilkan indikator loading dengan pesan 'Checking Package from Server' untuk memberikan umpan balik visual kepada pengguna. Gambar 8b menampilkan proses pengunduhan paket pembaruan dari server dengan indikator progress bar persentase secara bertahap. Pengunduhan dijalankan langsung oleh aplikasi setelah pengecekan versi pada startup mendeteksi adanya versi yang lebih baru; pendekatan ini sesuai karena aplikasi validator memang dirancang berjalan di latar depan selama proses pembaruan di pool, sehingga pengunduhan APK berukuran 16 MB dapat diselesaikan dengan cepat dalam satu sesi.

Gambar 8c menampilkan kondisi setelah pengunduhan selesai dengan notifikasi 'Package downloaded successfully', dilanjutkan proses instalasi otomatis menggunakan PackageInstaller API tanpa memerlukan konfirmasi pengguna. Secara teknis, kemampuan instalasi tanpa intervensi pengguna (silent installation) ini dimungkinkan karena perangkat Telpo T10 dikonfigurasi dalam mode device owner, sehingga aplikasi Tap On Bus memperoleh privilese untuk membuat PackageInstaller.Session dan menyelesaikan instalasi secara terprogram melalui pemanggilan commit() tanpa memunculkan dialog konfirmasi sistem yang biasanya wajib pada perangkat Android standar. Pendekatan ini krusial untuk konteks validator bus yang beroperasi tanpa operator teknis di lokasi: bila instalasi memerlukan satu kali sentuhan layar oleh pengguna, otomatisasi penuh tidak akan tercapai dan tujuan efisiensi penelitian ini gugur. Gambar 8d menampilkan kondisi perangkat telah menggunakan versi terbaru dengan pesan 'Device is up to date'. Kondisi pada Gambar 8d ini membuktikan bahwa mekanisme perbandingan versi di backend bekerja sebagai gerbang efisiensi: perangkat yang sudah terbaru tidak melakukan pengunduhan ulang, sehingga beban jaringan dan server hanya terjadi pada perangkat yang benar-benar memerlukan pembaruan.

3.3 Hasil Pengujian Fungsional (Black Box Testing)

Tabel 1 menyajikan hasil pengujian fungsional terhadap enam skenario registrasi dan manajemen perangkat. Seluruh skenario dieksekusi pada perangkat Telpo T10 yang terhubung ke server OTA melalui jaringan seluler 4G.

Tabel 1. Hasil Pengujian Registrasi dan Manajemen Perangkat

Skenario Pengujian	Input / Data Uji	Proses yang diuji	Hasil yang Diharapkan	Hasil Aktual	Status
Registrasi Perangkat	Serial number Telpo T10	Penyimpanan data perangkat ke DB	Perangkat berhasil terdaftar dengan ID unik	Sesuai	Berhasil
Pengecekan status perangkat	Serial number terdaftar	Sistem memverifikasi identitas perangkat	Perangkat dikenali; status ditampilkan	Sesuai	Berhasil
Pencatatan waktu registrasi	Timestamp saat registrasi	Sistem menyimpan waktu pendaftaran	Data timestamp tersimpan akurat di DB	Sesuai	Berhasil
Pencatatan waktu check update	Timestamp pengecekan saat aplikasi dijalankan	Sistem memperbarui field last_check_time	Waktu check update tersimpan di DB	Sesuai	Berhasil
Penampilan data perangkat di dashboard	Data 10 perangkat terdaftar	Dashboard merender tabel perangkat	Semua informasi perangkat tampil benar	Sesuai	Berhasil
Pencegahan duplikasi perangkat	Serial number yang sama dikirim ulang	Validasi uniqueness serial number	Sistem menolak registrasi duplikat; error 409 dikembalikan	Sesuai	Berhasil

Tabel 2 menyajikan hasil pengujian fungsional terhadap tiga belas skenario alur OTA Update secara end-to-end, mulai dari trigger pengecekan saat aplikasi dijalankan, skenario kegagalan unduhan dan instalasi serta pemulihannya, hingga pencatatan log pembaruan.

Tabel 2. Hasil Pengujian Sistem OTA Update End-to-End

Skenario Pengujian	Input / Data Uji	Proses yang Diuji	Hasil yang Diharapkan	Hasil Aktual	Status
Trigger pengecekan saat aplikasi dijalankan	Aplikasi Tap On Bus dimulai (startup)	Aplikasi memicu pengecekan versi ke server	Pengecekan terpicu otomatis saat aplikasi dijalankan	Sesuai	Berhasil
Pengecekan update tersedia	Versi perangkat lebih lama dari server	Perbandingan versi semantik	Respons berisi URL unduhan paket terbaru	Sesuai	Berhasil
Deteksi perangkat sudah terbaru	Versi perangkat sama dengan server	Validasi versi	Respons 'Device up to date'; tidak ada unduhan	Sesuai	Berhasil
Proses download package	URL paket APK valid	Pengunduhan file APK	File APK berhasil diunduh lengkap	Sesuai	Berhasil

TIN: Terapan Informatika Nusantara
Vol 6, No 10, March 2026, page 1975-1988
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i10.9406

Skenario Pengujian	Input / Data Uji	Proses yang Diuji	Hasil yang Diharapkan	Hasil Aktual	Status
Kegagalan download di tengah proses	Koneksi terputus saat unduhan 40%	Penanganan unduhan tidak lengkap	Berkas parsial dibuang; instalasi dibatalkan; versi lama tetap berjalan	Sesuai	Berhasil
Pemulihan unduhan via restart aplikasi	Aplikasi dijalankan ulang setelah download gagal	Pengulangan unduhan dari awal saat startup	Unduhan dimulai ulang dari 0% dan selesai lengkap; instalasi berhasil	Sesuai	Berhasil
Tampilan progres download	Byte data yang diterima secara bertahap	Progress bar diperbarui setiap 1%	Progress bar menampilkan persentase bertahap	Sesuai	Berhasil
Silent installation	File APK valid pasca unduhan	Instalasi via PackageInstaller API	APK terinstal tanpa konfirmasi pengguna	Sesuai	Berhasil
Kegagalan instalasi (device mati saat instalasi)	Perangkat mati mendadak saat instalasi berlangsung	Penanganan instalasi tidak tuntas oleh PackageInstaller	Instalasi dibatalkan sistem; versi lama tetap utuh dan dapat berjalan	Sesuai	Berhasil
Pemulihan instalasi via restart aplikasi	Aplikasi dijalankan kembali setelah device menyala	Pengecekan versi ulang dan pengulangan instalasi	Sistem mendeteksi versi belum terbaru; mengunduh dan menginstal ulang hingga berhasil	Sesuai	Berhasil
Integritas data transaksi	Data transaksi sebelum update	Proses update aplikasi	Semua data transaksi tetap ada pasca update	Sesuai	Berhasil
Fungsi aplikasi pasca update	Aplikasi versi terbaru	Menjalankan fungsi inti aplikasi	Semua fungsi berjalan normal	Sesuai	Berhasil
Pencatatan log pembaruan	Timestamp instalasi berhasil	Penyimpanan log update ke DB	Log tersimpan; status di dashboard diperbarui	Sesuai	Berhasil

Berdasarkan hasil black box testing terhadap seluruh 19 skenario pengujian, sistem OTA Update menunjukkan tingkat keberhasilan 100% dengan seluruh hasil aktual sesuai hasil yang diharapkan. Pengujian juga memverifikasi bahwa sistem berhasil mengatasi tantangan teknis kritis: mekanisme perbandingan versi pada backend mampu membedakan secara akurat antara perangkat yang sudah terbaru dan yang memerlukan pembaruan, silent installation berjalan tanpa memerlukan konfirmasi pengguna pada perangkat Telpo T10 yang dikonfigurasi sebagai dedicated device, dan data transaksi terbukti tetap utuh setelah proses pembaruan berhasil dieksekusi (Raihan & Voutama, 2023).

Pengujian skenario kegagalan menjadi bukti penting bahwa sistem tidak hanya berfungsi pada kondisi ideal, tetapi juga tangguh (resilient) terhadap gangguan. Pada skenario kegagalan download di tengah proses, koneksi sengaja diputus ketika unduhan mencapai sekitar 40%. Sistem menangani kondisi ini dengan membuang berkas APK parsial yang belum lengkap dan membatalkan proses instalasi, sehingga perangkat tetap menjalankan versi aplikasi lama tanpa mengalami kerusakan (corrupt install). Pendekatan all-or-nothing ini dirancang secara sengaja: instalasi hanya dilakukan apabila berkas APK terunduh secara utuh, sebab menginstal berkas yang tidak lengkap berisiko merusak aplikasi dan mengganggu operasional validator. Selanjutnya, pada skenario pemulihan, aplikasi yang sebelumnya gagal mengunduh dijalankan ulang. Saat aplikasi dimulai kembali, mekanisme pengecekan versi pada startup kembali aktif, mendeteksi bahwa versi terpasang masih lebih lama dari versi server, lalu memulai ulang proses pengunduhan dari awal (0%) hingga selesai sempurna dan instalasi berhasil. Hasil ini menunjukkan bahwa mekanisme pengecekan berbasis startup tidak hanya berfungsi sebagai pemicu pembaruan, tetapi juga berperan sebagai mekanisme pemulihan otomatis (self-healing) terhadap pembaruan yang gagal, tanpa memerlukan intervensi teknisi sama sekali.

Ketangguhan sistem juga diuji pada fase yang lebih kritis, yaitu kegagalan instalasi ketika berkas APK sudah terunduh penuh namun perangkat tiba-tiba mati di tengah proses instalasi. Skenario ini penting karena fase instalasi merupakan titik paling rawan: apabila penulisan paket aplikasi terputus, terdapat risiko aplikasi menjadi rusak (corrupt) dan tidak dapat dijalankan. Hasil pengujian menunjukkan bahwa PackageInstaller API pada Android menangani kondisi ini secara transaksional (atomic), yaitu instalasi yang tidak tuntas akan dibatalkan oleh sistem operasi sehingga versi aplikasi lama tetap utuh dan dapat berjalan normal setelah perangkat dinyalakan kembali. Dengan demikian, kegagalan di tengah instalasi tidak meninggalkan aplikasi dalam keadaan setengah terpasang. Pada skenario pemulihannya, ketika perangkat menyala dan aplikasi Tap On Bus dijalankan kembali, mekanisme pengecekan versi pada startup kembali mendeteksi bahwa versi terpasang masih lebih lama dari versi server, lalu mengulang seluruh proses pembaruan—mengunduh dan menginstal ulang—hingga berhasil. Kedua skenario kegagalan ini (gagal saat unduhan maupun gagal saat instalasi) membuktikan bahwa desain sistem mengikuti prinsip idempotensi: berapa kali pun proses pembaruan terganggu dan diulang, hasil akhirnya konsisten menuju versi terbaru tanpa merusak data maupun aplikasi yang sedang berjalan.

3.4 Hasil Pengujian Komparatif Performa

Perbandingan langsung antara metode manual dan metode OTA merupakan inti dari evaluasi efisiensi pada penelitian ini. Untuk menjamin keadilan perbandingan, kedua metode diukur pada konteks yang sama, yaitu proses pembaruan perangkat di pool bus saat kendaraan dalam keadaan diam, sehingga perbedaan waktu yang diperoleh murni mencerminkan perbedaan mekanisme—bukan perbedaan kondisi lingkungan. Tabel 3 menyajikan perbandingan waktu

TIN: Terapan Informatika Nusantara
Vol 6, No 10, March 2026, page 1975-1988
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i10.9406

rata-rata proses pembaruan aplikasi antara metode manual (menggunakan flashdisk) dan metode OTA berdasarkan 10 sampel pengujian untuk masing-masing metode, mencakup seluruh tahapan dari persiapan hingga verifikasi selesainya pembaruan pada satu perangkat.

Tabel 3. Perbandingan Waktu Pembaruan: Metode Manual vs. Sistem OTA

Komponen Waktu	Metode Manual (Menit)	Metode OTA (Menit)	Selisih (Menit)
Persiapan (akses panel & port USB / startup aplikasi)	6	0,1	5,9
Akses flashdisk via file manager / pengecekan versi	4	0,1	3,9
Proses unduhan paket APK	–	0,5	–
Instalasi paket pembaruan	4,5	0,4	4,1
Verifikasi selesai	1,5	0,1	1,4
Total rata-rata per perangkat	16	1,2	14,8
Peningkatan efisiensi	–	92,50%	–

Data pada Tabel 3 menunjukkan bahwa sistem OTA berhasil menurunkan rata-rata waktu pembaruan per perangkat dari 16,0 menit menjadi sekitar 1,2 menit. Dengan menerapkan Persamaan (4), peningkatan efisiensi dihitung sebesar ((16,0 − 1,2) / 16,0) × 100% = 92,5%. Komponen waktu yang mengalami reduksi paling signifikan adalah komponen persiapan dan akses berkas, yang pada metode manual menuntut teknisi membuka panel dashboard bus, mengakses port USB, mencolokkan flashdisk, lalu menavigasi berkas APK melalui file manager. Seluruh rangkaian langkah fisik tersebut nyaris hilang pada metode OTA karena aplikasi memicu pembaruannya sendiri saat dijalankan dan mengunduh paket secara otomatis dari server. Komponen instalasi juga menurun dari 4,5 menit menjadi 0,4 menit berkat mekanisme silent installation pada APK berukuran kecil (16 MB) yang tidak memerlukan navigasi maupun konfirmasi manual oleh teknisi.

Keunggulan sistem OTA paling jelas terlihat pada konteks pembaruan per pool bus. Pada metode manual, pembaruan dilakukan secara bertahap oleh 2–3 teknisi yang menangani 10–15 perangkat per pool; sebuah pool berisi 12 perangkat yang dikerjakan 3 teknisi secara paralel membutuhkan sekitar 64 menit. Sebaliknya, sistem OTA memungkinkan seluruh perangkat dalam satu pool diperbarui secara serentak (simultan), karena setiap perangkat memicu pengecekan dan pembaruannya sendiri saat aplikasi dijalankan, sementara backend Golang mampu melayani banyak permintaan secara bersamaan melalui mekanisme concurrency berbasis goroutine. Dengan demikian, waktu penyelesaian pembaruan satu pool tidak lagi merupakan akumulasi linear dari jumlah perangkat dikali waktu per perangkat, melainkan mendekati waktu pembaruan satu perangkat (sekitar 1,2 menit), ditambah sedikit overhead jaringan. Perbedaan fundamental antara model eksekusi serentak (OTA) dan bertahap (manual) inilah dari sekitar 64 menit menjadi hanya beberapa menit per pool yang menjadikan penghematan operasional sistem OTA sangat substansial, dan keunggulan tersebut berlipat seiring bertambahnya jumlah pool yang harus dilayani.

3.5 Hasil Pengujian Performa Pengunduhan dan Instalasi

Untuk memberikan bukti kuantitatif yang lebih rinci di luar status fungsional, dilakukan pengujian performa terhadap waktu pengunduhan dan instalasi paket APK. Seluruh pengujian performa dilakukan pada kondisi operasional sebenarnya, yaitu ketika perangkat berada di pool bus dalam keadaan diam (idle) menggunakan koneksi jaringan seluler 4G yang dipakai di lapangan, sesuai prosedur pembaharuan yang dirancang berlangsung di pool dan bukan saat kendaraan beroperasi. Pengujian pertama mengukur pengaruh ukuran berkas APK terhadap waktu pengunduhan menggunakan koneksi jaringan seluler 4G yang dipakai di lapangan, dengan tiga sampel ukuran APK yang merepresentasikan rentang ukuran aplikasi Tap On Bus. Hasilnya disajikan pada Tabel 4.

Tabel 4. Hasil Pengujian Waktu Pengunduhan Berdasarkan Ukuran APK (Jaringan 4G)

Ukuran APK (MB)	Rata-rata Waktu Unduh (detik)	Throughput Efektif (Mbps)
10	19	≈ 4,3
16	30	≈ 4,3
22	41	≈ 4,3

Nilai throughput efektif pada Tabel 4 diperoleh menggunakan Persamaan (2); sebagai contoh, untuk APK 16 MB dengan waktu unduh 30 detik dihasilkan (16 × 8) / 30 ≈ 4,3 Mbps. Tabel 4 menunjukkan bahwa waktu pengunduhan meningkat secara proporsional terhadap ukuran berkas APK, dengan throughput efektif yang relatif konstan di kisaran 4,3 Mbps pada jaringan 4G. Pola linear ini mengindikasikan bahwa proses pengunduhan berjalan stabil tanpa bottleneck pada sisi aplikasi klien, dan waktu unduh dapat diprediksi berdasarkan ukuran paket. Untuk ukuran APK aktual aplikasi Tap On Bus, yaitu 16MB, waktu pengunduhan rata-rata hanya 30 detik, sehingga proses pembaruan tergolong ringan dan cepat.

Pengujian kedua merinci komponen waktu pada proses OTA untuk APK berukuran 16MB menggunakan koneksi jaringan seluler 4G yang dipakai di lapangan (SIM 4G operator komersial), guna mengetahui tahapan mana yang paling mempengaruhi total waktu pembaruan. Penyebutan operator hanya untuk mendokumentasikan kondisi pengujian dan bukan bentuk endorsement. Hasilnya disajikan pada Tabel 5.

TIN: Terapan Informatika Nusantara
Vol 6, No 10, March 2026, page 1975-1988
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i10.9406

Tabel 5. Rincian Komponen Waktu Proses OTA pada Jaringan 4G (APK 16 MB)

Komponen Proses OTA	Waktu (detik)	Persentase (%)
Pemicuan & pengecekan versi (startup)	12	16,7
Pengunduhan paket APK	30	41,7
Instalasi (silent install)	24	33,3
Verifikasi & pencatatan log	6	8,3
Total proses OTA	72	≈ 1,2 menit

Nilai total OTA pada Tabel 5 dihitung dengan Persamaan (3) sebagai akumulasi seluruh komponen, yaitu sekitar 72 detik atau 1,2 menit per perangkat. Rincian ini menunjukkan bahwa komponen pengunduhan paket merupakan penyumbang waktu terbesar (30 detik atau 41,7%), diikuti komponen instalasi (24 detik atau 33,3%), sementara komponen pemicuan, verifikasi, dan pencatatan log relatif kecil. Temuan ini mengindikasikan bahwa upaya optimasi lanjutan paling efektif difokuskan pada efisiensi pengunduhan, misalnya melalui kompresi paket atau penggunaan mekanisme pembaruan diferensial (delta update). Meskipun demikian, total waktu 1,2 menit pada jaringan 4G ini sudah jauh lebih efisien dibandingkan 16 menit pada metode manual, sehingga keunggulan sistem OTA terbukti secara konsisten.

3.6 Pembahasan

Jika dibandingkan dengan penelitian OTA pada domain IoT, terdapat beberapa perbedaan mendasar yang perlu dicermati. Ikhsani dan Budi (2024) melaporkan waktu pembaruan rata-rata 28,845 detik untuk firmware ESP8266; Wijaya dkk. (2025) melaporkan 18,069 detik untuk Raspberry Pi. Angka-angka ini tidak dapat dibandingkan langsung dengan hasil penelitian ini (sekitar 1,2 menit untuk APK Android berukuran 16 MB) karena perbedaan ukuran payload dan kompleksitas proses: firmware IoT umumnya berukuran puluhan hingga ratusan kilobyte, sementara paket APK Android berukuran belasan megabyte. Kompleksitas proses instalasi juga berbeda secara fundamental, di mana APK Android memerlukan validasi oleh sistem operasi, penanganan permission, dan migrasi data antar versi.

Dari aspek arsitektur, penelitian ini berkontribusi dengan mengimplementasikan solusi OTA yang secara khusus menangani tantangan unik pembaruan APK Android: penggunaan PackageInstaller API untuk silent installation pada dedicated device, serta mekanisme pengecekan versi berbasis startup aplikasi yang efisien tanpa memerlukan layanan latar belakang permanen. Aspek-aspek ini tidak ditemukan dalam penelitian OTA berbasis IoT yang menggunakan pendekatan flashing firmware langsung (Cahyono dkk., 2023; Ramadhan dkk., 2022).

Li dkk. (2024) dalam survei OTA untuk kendaraan cerdas menekankan bahwa keamanan merupakan aspek paling kritis dalam sistem OTA skala produksi. Pada tahap penelitian ini, komunikasi antara perangkat dan server masih menggunakan protokol HTTP tanpa enkripsi, mengingat fokus penelitian diarahkan pada validasi fungsionalitas dan efisiensi mekanisme pembaruan. Aspek keamanan yang lebih kuat seperti penerapan protokol HTTPS/SSL, validasi integritas berkas (misalnya melalui checksum), autentikasi berbasis token (JWT), dan enkripsi payload masih perlu ditambahkan untuk kesiapan produksi penuh, yang diidentifikasi sebagai keterbatasan penelitian ini.

4. KESIMPULAN

Penelitian ini berhasil mengimplementasikan sistem Over-The-Air (OTA) Update berbasis REST API yang bersifat mandiri (in-house) untuk menggantikan proses pembaruan manual aplikasi Android Tap On Bus pada lebih dari 400 perangkat validator Telpo T10 yang tersebar di berbagai lokasi operasional. Hasil pengujian fungsional menggunakan Black Box Testing terhadap 19 skenario menunjukkan tingkat keberhasilan 100%, dengan seluruh fungsi kritis sistem—mulai dari registrasi perangkat otomatis, pengecekan versi yang terpicu saat aplikasi dijalankan (startup), perbandingan versi pada backend, pengunduhan paket APK, penanganan kegagalan unduhan maupun kegagalan instalasi (saat perangkat mati di tengah proses) dengan pemulihan otomatis (self-healing) melalui pengulangan saat startup berikutnya, silent installation menggunakan PackageInstaller API, hingga pencatatan log pembaruan secara terpusat—berjalan sesuai spesifikasi yang ditetapkan tanpa ditemukan ketidaksesuaian. Pengujian komparatif performa menunjukkan bahwa sistem OTA berhasil menurunkan rata-rata waktu pembaruan per perangkat dari 16 menit (metode manual) menjadi sekitar 1,2 menit (metode OTA), setara peningkatan efisiensi sebesar 92,5%, sekaligus mengeliminasi kebutuhan perjalanan teknisi ke setiap lokasi perangkat dan risiko kerusakan data akibat penggunaan media penyimpanan fisik berulang. Pengujian performa di pool bus pada kondisi diam menunjukkan bahwa untuk APK berukuran 16 MB melalui jaringan seluler 4G, total waktu proses OTA hanya sekitar 72 detik (1,2 menit) per perangkat—jauh lebih cepat dibandingkan 16 menit (960 detik) metode manual. Sistem juga terbukti mampu menjaga integritas data transaksi selama proses pembaruan, sehingga operasional layanan transportasi tidak terganggu. Meskipun demikian, penelitian ini memiliki keterbatasan yang perlu diakui: komunikasi data masih menggunakan protokol HTTP tanpa enkripsi, sementara aspek keamanan mendalam seperti penerapan HTTPS/SSL, validasi integritas berkas, autentikasi

TIN: Terapan Informatika Nusantara
Vol 6, No 10, March 2026, page 1975-1988
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i10.9406

berbasis token (JWT), dan enkripsi payload belum diimplementasikan, pengujian ketangguhan telah mencakup skenario kegagalan unduhan dan kegagalan instalasi beserta pemulihannya, namun skenario rollback otomatis ke versi sebelumnya bila aplikasi baru (yang berhasil terpasang) ternyata bermasalah secara fungsional belum diuji secara komprehensif, serta pengujian load testing untuk mensimulasikan ratusan perangkat yang melakukan pembaruan secara bersamaan di pool belum tercakup. Oleh karena itu, penelitian lanjutan disarankan untuk menerapkan protokol HTTPS/SSL, mekanisme autentikasi JWT, enkripsi end-to-end, pengujian beban (load testing), serta fitur rollback otomatis untuk meningkatkan keandalan dan keamanan sistem pada skala produksi yang lebih besar.

REFERECES

Anis, Y., Mukti, A. B., & Rosyid, A. N. (2023). Penerapan model waterfall dalam pengembangan sistem informasi aset destinasi wisata berbasis website. *KLIK: Kajian Ilmiah Informatika Dan Komputer, 4(2)*, 1134–1142. https://doi.org/10.30865/klik.v4i2.1287

Aurellia, A. (2025). Pemanfaatan UML dalam perancangan sistem informasi produk kreatif daur ulang sampah berbasis web. *Jurnal Informatika dan Teknik Elektro Terapan, 13(3S1)*. https://doi.org/10.23960/jitet.v13i3S1.8073

Cahyono, E. B., Budi, A. S., & Akbar, S. R. (2023). Pengembangan gateway untuk mendukung proses over the air firmware update pada perangkat WSN berbasis bluetooth low energy. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 7(2), 953–959.

Falderika, F., Sakti, N. O., Ramadhan, I., Alfaridzi, M. S., & Albar, C. N. (2021). Rancang bangun sistem informasi transportasi umum perkotaan berbasis android. *IJIS - Indonesian Journal On Information System*, 6(2). https://doi.org/10.36549/ijis.v6i2.141

Gunawan, A., Handayani, E. T. E., Andrianingsih, A., & Sari, R. T. K. (2025). Keamanan siber (1st ed.). Literasi Nusantara Abadi.

Han, C., Liu, X., Liu, B., & Zhang, Q. (2022). Design of embedded remote software update system based on FPGA+ARM. *Journal of Interconnection Networks*, 22(Supp02), 2143040. https://doi.org/10.1142/S0219265921430404

Ikhsani, M. F., & Budi, A. S. (2024). Implementasi over the air update firmware secara massal pada ESP8266 untuk perangkat internet of things. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 8(6). https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/13810

Li, B., Hu, W., Da, L., Wu, Y., Wang, X., Li, Y., & Yuan, C. (2024). Over-the-air upgrading for enhancing security of intelligent connected vehicles: A survey. Artificial Intelligence Review, 57(11), 314. https://doi.org/10.1007/s10462-024-10968-z

Muharam, Y., & Hidayat, T. (2024). Pengembangan aplikasi back-end e-commerce menggunakan rest api golang untuk optimalisasi kinerja server. *COMPUTING: Jurnal Informatika*, 11(01), 7–13. https://doi.org/10.55222/computing.v11i01.1479

Nabila, S., Putri, A. R., Hafizhah, A., Rahmah, F. H., & Muslikhah, R. (2021). Pemodelan diagram UML pada perancangan sistem aplikasi konsultasi hewan peliharaan berbasis android. *Jurnal Ilmu Komputer dan Bisnis*, 12(2), 130–139. https://doi.org/10.47927/jikb.v12i2.150

Ningsih, W., & Nurfauziah, H. (2023). Perbandingan model waterfall dan metode prototype untuk pengembangan aplikasi pada sistem informasi. *Jurnal Ilmiah METADATA*, 5(1), 83–95. https://doi.org/10.47652/metadata.v5i1.311

Pattinama, Y. L., & Susanti, I. (2023). Implementasi rest api web service dengan otentifikasi JSON web token untuk aplikasi properti. *Informatik: Jurnal Ilmu Komputer*, 19(1), 81–89. https://doi.org/10.52958/iftk.v19i1.5724

Rahmawan, H., Rahmad Akbar, A., Sarjoko, F. M., & Wahjuni, S. (2025). Pembaruan firmware secara over the air pada banyak perangkat IoT berbasiskan AsyncElegantOTA dan AutoIt. *Proceedings National Conference Sinesia*, 1(1), 162–175. https://doi.org/10.69836/ncrcs-sinesia.v1i1.35

Raihan, H., & Voutama, A. (2023). Pengujian black box pada aplikasi database perguruan tinggi dengan teknik equivalence partition. *Antivirus: Jurnal Ilmiah Teknik Informatika*, 17(1), 1–18. https://doi.org/10.35457/antivirus.v17i1.2501

Ramadhan, S., Budi, A. S., & Ichsan, M. H. H. (2022). Rancang bangun sistem auto-config sensor baru pada perangkat IoT secara over-the-air menggunakan protokol HTTP berbasis Raspberry-Pi. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 6(1), 216–224.

Rastogi, D. (2024). Over-the-air (OTA) software upgrades for IoT devices. Journal of Quantum Science and Technology, 1(3). https://doi.org/10.63345/jqst.v1i3.206

Saputra, D., Dharmawan, W. S., Syarif, M., & Risdiansyah, D. (2023). Analisis dan perancangan sistem informasi (1st ed.). Insan Cendekia Mandiri.

Sugiyono. (2022). Metode penelitian kuantitatif, kualitatif, dan R&D. Alfabeta.

Susanto, D. D. (2024). Implementasi UML pada perancangan sistem informasi pelatihan kerja di balai latihan kerja kota Mojokerto. *Jurnal Informatika Teknologi dan Sains (Jinteks)*, 6(4), 862–871. https://doi.org/10.51401/jinteks.v6i4.4851

TIN: Terapan Informatika Nusantara
Vol 6, No 10, March 2026, page 1975-1988
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v6i10.9406

Syahputra, P. S., Alamsyah, A., & Mirza, M. (2025). Sosialisasi sistem pembayaran nontunai penumpang pada pelayanan bus rapid transit (studi kasus bus Tayo kota Tangerang). *Social Science Academic*, 3(1), 84–93. https://doi.org/10.37680/ssa.v3i1.8134

Wijaya, D. Y., Shaffan, N. H., & Primananda, R. (2025). Implementasi sistem pembaruan aplikasi 'over the air (OTA)' pada perangkat IoT berbasis Raspberry Pi menggunakan platform Github. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 9(7). https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/15161