

Research Article

Rancang Bangun PaaS Cloud IoT Mandiri Berbasis CoAP dengan Autentikasi Dua Lapis

Sayful Nizan¹, Mochamad Susantok²

ABSTRAK: Penggunaan layanan *cloud* pihak ketiga dalam pengembangan IoT menimbulkan masalah keamanan dan ketergantungan infrastruktur. Penelitian ini menawarkan alternatif dengan membangun sistem cloud IoT mandiri berbasis *Platform as a Service* (PaaS) dan protokol CoAP. Sistem dibangun di atas *server* Node.js dan *database* SQLite, kemudian diuji dalam lingkungan lokal menggunakan ESP8266 (sensor kelembaban tanah) dan ESP32 (sensor PIR). Komunikasi memanfaatkan CoAP di atas UDP dengan autentikasi dua lapis: JSON Web Token (JWT) untuk dashboard dan token 64 karakter untuk perangkat. Hasil pengujian menunjukkan latensi rata-rata 115,56 ms, laju pengiriman client 0,2 paket/detik dari 300 paket. Hasil latensi 115,56 ms ini terbukti lebih efisien dibandingkan protokol MQTT standar pada jaringan lokal (Amrullah et al., 2022; Sifawa et al., 2026). Autentikasi dua lapis efektif menolak akses tanpa token sah. Kontribusi utama adalah kemandirian dari cloud pihak ketiga dan keamanan data terkontrol melalui *dashboard WebSocket*

Kata kunci: CoAP, PaaS, ESP8266, ESP32, Autentifikasi dua lapis

1. PENDAHULUAN

Penerapan teknologi *Internet of Things* (IoT) telah meluas secara masif di berbagai bidang strategis, mulai dari pemantauan presisi pertanian, otomatisasi keamanan bangunan, hingga kendali infrastruktur industri modern. Secara umum, arsitektur IoT mengadopsi model komunikasi *client-server*, di mana node sensor terdistribusi bertindak sebagai client yang mentransmisikan data operasional ke sistem cloud pusat untuk diproses, disimpan, dan divisualisasikan. Penggunaan infrastruktur cloud bertujuan untuk meningkatkan skalabilitas, fleksibilitas akses, dan efisiensi manajemen data. Namun, seiring dengan meningkatnya volume data sensitif yang ditransmisikan, ketergantungan pada infrastruktur cloud membawa tantangan mendasar terkait privasi, keandalan koneksi, dan kedaulatan data [1].

Sebagian besar implementasi IoT saat ini masih sangat bergantung pada pihak ketiga, seperti *Blynk*. Ketergantungan ini menimbulkan beberapa kelemahan arsitektural yang kritis. Pertama,

OPEN ACCESS

Afiliasi:

¹Teknik Elektronika Telekomunikasi, Politeknik Caltex Riau, Pekanbaru, Indonesia

²Teknologi Rekayasa Jaringan Telekomunikasi, Politeknik Caltex Riau, Pekanbaru, Indonesia

Korespondensi:

Mochamad Susantok, S.ST., M.T.

Email: santok@pcr.ac.id

Diterima Redaksi:

12-08-2026

Selesai Revisi:

28-08-2026

Diterbitkan Online:

27-09-2026

Cara Mengutip Artikel ini (IEEE):

S. Nizan and M. Susantok, "Rancang Bangun PaaS Cloud IoT Mandiri Berbasis CoAP dengan Autentikasi Dua Lapis", *J.Electo*, pp. 9–21.

This article is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.



penyimpanan data pada *server* eksternal vendor menciptakan kerentanan privasi dan kedaulatan data; apabila *server* pihak ketiga mengalami kebocoran keamanan atau disrupsi layanan, seluruh ekosistem IoT yang terhubung akan terdampak secara langsung. Penelitian terdahulu oleh Erwin dan Pratama menunjukkan bahwa penggunaan aplikasi Blynk tidak hanya membatasi kontrol pengguna atas direktori data, tetapi juga membatasi fleksibilitas analisis karena fitur pengolahan data yang kaku dan terikat pada ekosistem proprietary vendor [2]. Kedua, dari sisi protokol komunikasi, mayoritas platform komersial mengandalkan HTTP atau MQTT yang berjalan di atas *Transport Control Protocol* (TCP). Meskipun TCP menawarkan keandalan pengiriman paket, mekanisme three-way handshake dan overhead header yang besar menyebabkan lonjakan latensi, konsumsi energi yang boros, dan penurunan efisiensi CPU secara drastis pada mikrokontroler bersumber daya terbatas (*resource-constrained devices*) seperti ESP8266 dan ESP32 [4].

Upaya untuk mengatasi keterbatasan transmisi pada perangkat berdaya rendah telah mendorong penggunaan Constrained Application Protocol (CoAP), sebuah protokol jaringan ringkas berbasis *User Datagram Protocol* (UDP) yang mengadopsi arsitektur RESTful. CoAP terbukti secara empiris mampu menekan latensi dan menghemat konsumsi energi perangkat dibandingkan protokol berbasis TCP [6]. Namun demikian, pengurangan CoAP dalam lanskap praktis menghadapi celah penelitian (*research gap*) yang signifikan pada aspek keamanan. Pengamanan standar CoAP umumnya mengandalkan *Datagram Transport Layer Security* (DTLS). Penggunaan DTLS memicu overhead komputasi dan enkripsi yang pada akhirnya membatalkan efisiensi energi yang menjadi keunggulan utama CoAP [5]. Di sisi lain, mengabaikan lapisan autentikasi pada CoAP demi mengejar kecepatan transmisi akan membuka celah serangan *unauthorized data injection* dari *node* asing yang tidak terverifikasi [3].

Untuk mengatasi celah tersebut, penelitian ini mengajukan pendekatan arsitektur *Platform as a Service* (PaaS) mandiri (*self-hosted*) yang diintegrasikan dengan protokol CoAP dan mekanisme keamanan lightweight dua lapis. Arsitektur PaaS mandiri dibangun di atas lingkungan *runtime* Node.js dan *database* SQLite, yang berjalan sepenuhnya pada jaringan lokal untuk menjamin kedaulatan data penuh tanpa ketergantungan pada vendor pihak ketiga. Protokol CoAP dimanfaatkan sebagai jalur komunikasi M2M utama dari node sensor ESP8266 dan ESP32 guna mempertahankan efisiensi daya dan latensi rendah. Untuk mengatasi kelemahan keamanan tanpa membebankan komputasi DTLS pada mikrokontroler, sistem ini menerapkan autentikasi dua lapis: penggunaan *JSON Web Token* (JWT) pada antarmuka dashboard berbasis HTTP/WebSocket, serta enkapsulasi token heksadesimal unik sepanjang 64 karakter pada parameter *URI-Query* CoAP yang divalidasi langsung oleh *server* sebelum proses interaksi data dieksekusi [10].

Kebaharuan (*novelty*) penelitian ini terletak pada formulasi arsitektur PaaS mandiri berbasis Node.js yang mengintegrasikan layanan CoAP (UDP), HTTP, dan *WebSocket* secara terpadu dengan skema autentikasi lightweight berbasis *application-layer* sebagai alternatif pengganti mekanisme DTLS yang kompleks. Kontribusi utama dari karya ilmiah ini mencakup: (1) rancang bangun infrastruktur PaaS IoT *self-hosted* yang mengeliminasi ketergantungan pada layanan SaaS komersial; (2) pengembangan metode verifikasi identitas node CoAP berbasis *URI-Query Parsing* token 64-character hex yang terbukti efektif menolak akses ilegal (4.01 *Unauthorized*) pada tingkatan application layer; serta (3) evaluasi empiris komprehensif terhadap parameter kinerja sistem yang meliputi latensi transmisi, throughput, keandalan pengiriman paket data, dan efektivitas pembatasan hak akses [3], [5], [6].

2. METODE PENELITIAN

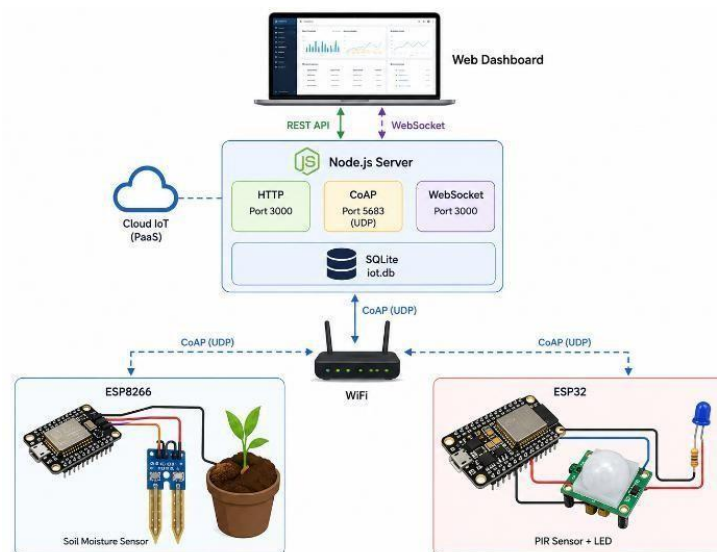
Penelitian ini menggunakan pendekatan rancang bangun sistem (*system development research*) dengan metode eksperimental, yang bertujuan untuk merancang, membangun, dan menguji sistem cloud IoT mandiri berbasis *Platform as a Service* (PaaS) menggunakan protokol *Constrained Application Protocol* (CoAP). Perangkat keras yang digunakan dalam sistem ini terdiri dari

mikrokontroler ESP8266 dengan sensor kelembaban tanah, mikrokontroler ESP32 dengan sensor PIR, serta laptop yang berfungsi sebagai *server* cloud. ESP8266 berfungsi membaca nilai kelembaban tanah dan mengirimkannya ke *server* melalui protokol CoAP, sementara ESP32 berfungsi mendeteksi gerakan menggunakan sensor PIR dan mengirimkan statusnya secara periodik.

Data yang diperoleh dari perangkat IoT dikirimkan ke *server* melalui protokol CoAP di atas UDP, kemudian disimpan dalam database *SQLite* dan ditampilkan secara langsung menggunakan *dashboard* berbasis web yang terhubung melalui *WebSocket*. Tahapan pengolahan data mencakup proses validasi token, penyimpanan data ke database, dan distribusi data ke *dashboard* secara *real-time*.

2.1 Perancangan Arsitektur Penelitian

Arsitektur sistem yang peneliti rancang mengacu pada model referensi ITU-T Y.2060 yang membagi sistem IoT menjadi empat lapisan [7]. Pendekatan ini merupakan salah satu model arsitektur IoT yang banyak diadopsi dalam pengembangan sistem terdistribusi [15]. Lapisan pertama adalah *Device Layer*. Di sini peneliti menempatkan ESP8266 dengan sensor soil moisture dan ESP32 dengan sensor PIR. Kedua mikrokontroler ini peneliti pilih karena mendukung protokol CoAP.



Gambar 1. Arsitektur Sistem Cloud IoT Berbasis PaaS

Perancangan sistem dilakukan dengan merancang arsitektur sistem secara keseluruhan, mencakup diagram blok sistem, skema database, alur komunikasi antara perangkat IoT dan *server*, serta antarmuka pengguna berbasis web. Implementasi sistem membangun seluruh komponen sesuai rancangan, meliputi *server* backend menggunakan Node.js, database *SQLite*, *dashboard* web, serta firmware untuk ESP8266 dan ESP32 menggunakan Arduino IDE.

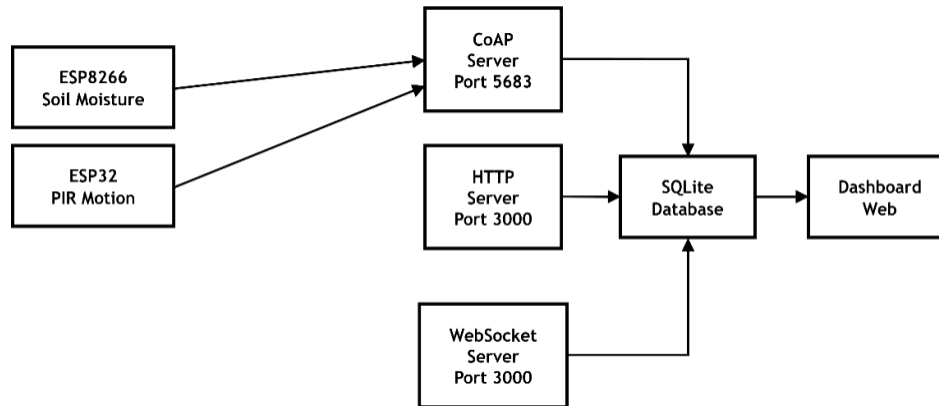
Lapisan kedua adalah *Network Layer*. Peneliti menggunakan protokol CoAP yang berjalan di atas UDP pada port 5683. Untuk komunikasi, perangkat IoT mengirimkan paket dengan format meliputi:

coap://[alamat_server]:5683/{id_perangkat}/{jenis_sensor}?token={token_perangkat}

Server yang menerima paket akan mengekstrak id perangkat dan token dari *Uri-Path* dan *Uri-Query*. Jika token cocok dengan *database*, *server* menyimpan data dan mengirim balasan 2.04 *Changed*. Sebaliknya, jika token tidak valid atau tidak disertakan, *server* menolak dengan 4.01 *Unauthorized*.

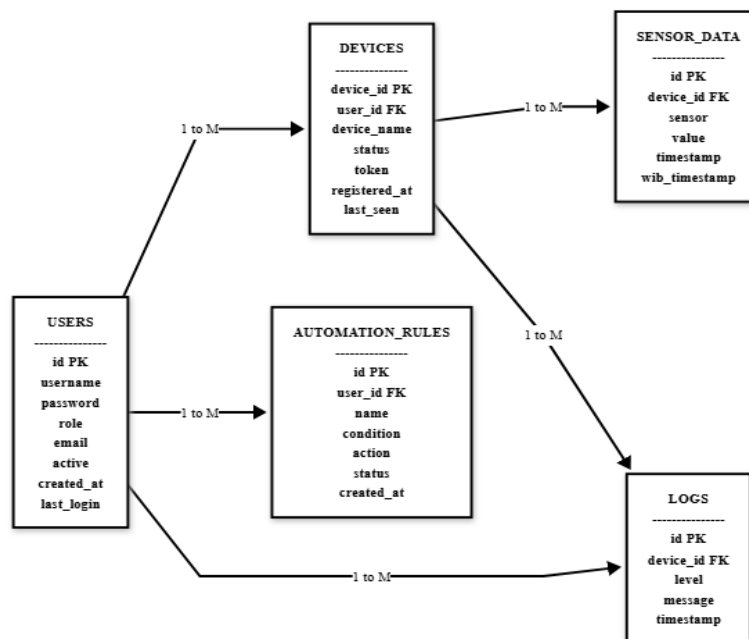
[3]. Lapisan ketiga adalah *Service Support Layer*. *Server* Node.js berjalan dan mengelola semua proses. *Server* ini menjalankan tiga layanan sekaligus. *HTTP Server* di port 3000 melayani *dashboard* dan *API*. *CoAP*

Server di port 5683 menerima data dari perangkat IoT. *WebSocket Server* di port 3000 mengirimkan pembaruan data *real-time* ke *dashboard* [8].



Gambar 2. Diagram Blok Sistem

Untuk *database*, peneliti menggunakan *SQLite*. *SQLite* ringan dan mudah diintegrasikan dengan Node.js. *Database* *iot.db* berisi delapan tabel: *users*, *devices*, *sensor_types*, *device_sensors*, *sensor_data_new*, *alerts*, *automation_rules*, dan *device_logs* [9].



Gambar 3. Entity Relationship Diagram (ERD)

2.2 Mekanisme Keamanan

Keamanan menjadi perhatian utama peneliti. Peneliti menerapkan dua lapis autentikasi [10]. Lapis pertama menggunakan *JSON Web Token* (JWT) untuk pengguna dashboard. Token ini diterbitkan *server* setelah *login* dan harus disertakan dalam setiap permintaan API. Lapis kedua menggunakan token unik sepanjang 64 karakter *hex* untuk setiap perangkat IoT yang harus disertakan dalam setiap pengiriman *CoAP*.

Peneliti juga menerapkan otorisasi berbasis peran. Admin memiliki akses penuh ke semua fitur. User biasa hanya *dapat* mengakses fitur operasional. Password pengguna peneliti simpan dalam *hash bcrypt* dengan *cost factor* 10 [11].

2.3 Pengujian Sistem

Peneliti menguji tiga aspek utama kinerja sistem, yaitu latensi, throughput, dan keandalan [12]. Latensi diukur dari selisih waktu antara pengiriman paket dari client hingga diterimanya balasan dari server. Untuk menghindari ambiguitas, throughput dalam penelitian ini didefinisikan dalam dua sudut pandang:

(1) *Throughput Client*, yaitu laju pengiriman data aktual dari perangkat IoT yang ditentukan oleh interval pengiriman.

(2) *Throughput Server*, yaitu kapasitas maksimum teoritis server dalam memproses permintaan CoAP per satuan waktu.

Pengiriman dilakukan sebanyak 300 paket dengan interval 5 detik dari perangkat ESP8266 ke server CoAP. Dengan interval 5 detik, Throughput Client dihitung berdasarkan pengiriman (1 paket/5 detik), berikut persamaannya:

$$R=15 = 0,2 \text{ paket/detik}$$

Sementara, *Throughput server* didefinisikan sebagai kapasitas server dalam memproses permintaan CoAP per satuan waktu. Fungsi *calculateThroughput()* pada sisi client mengukur rata-rata waktu yang dibutuhkan server untuk memproses setiap paket, dengan persamaan:

$$\text{Throughputserver} = t_{\text{processing}}$$

di mana $t_{\text{processing}}$ adalah rata-rata waktu pemrosesan server.

Berdasarkan hasil pengukuran, rata-rata waktu pemrosesan *server* adalah 115,56 ms, sehingga kapasitas teoritis server adalah:

$$\text{Throughputserver} = 0,115561 \approx 8,65 \text{ paket/detik}$$

Keandalan sistem diukur dari persentase paket yang berhasil diterima server dibandingkan dengan total paket yang dikirimkan [13]:

$$\text{Keandalan} = N_{\text{(success)}}/N_{\text{(total)}} \times 100\% = 262/300 \times 100\% = 87,33\%$$

Selain itu, pengujian keamanan dilakukan dengan mensimulasikan beberapa skenario untuk menguji efektivitas autentikasi dua lapis. Untuk token perangkat IoT, skenario yang diuji meliputi:

- (a) token valid
- (b) token tidak valid (dimodifikasi satu karakter)
- (c) token tidak disertakan dalam permintaan
- (d) simulasi replay attack dengan menangkap paket CoAP berisi token valid menggunakan packet sniffer (Wireshark) dan mengirimkannya ulang ke server beberapa saat kemudian.

Skenario (d) ini dirancang untuk menguji apakah server memiliki mekanisme pencegahan replay attack meskipun komunikasi CoAP tidak dienkripsi. Untuk autentikasi JWT, dilakukan pengujian dengan skenario token dihapus dari local storage dan token dimodifikasi secara ilegal.

3. Hasil dan Pembahasan

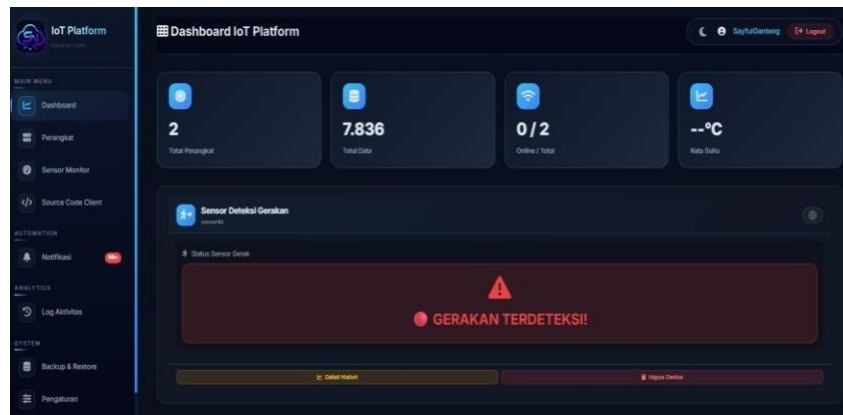
3.1 Implementasi Sistem

Dalam implementasi sistem yang dilakukan menggunakan perangkat keras berupa laptop Lenovo ThinkPad T495 yang dispesifikasi oleh prosesor AMD Ryzen 5 PRO 3500U, RAM 8 GB, serta dijalankan di atas sistem operasi Windows 11 Pro 64-bit. Akses menuju *dashboard* utama dapat dilakukan melalui alamat <http://localhost:3000> setelah melalui proses autentikasi (*login*).

Halaman utama *dashboard* menyajikan visualisasi data *real-time* dalam bentuk kartu informasi

perangkat (*device cards*), sebagaimana ditunjukkan pada Gambar 4. Setiap kartu memuat parameter teknis yang meliputi nama perangkat, ID unik, jenis mikrokontroler, status konektivitas, dan nilai pembacaan sensor terbaru.

Selain menampilkan data, antarmuka ini menyediakan formulir registrasi untuk penambahan perangkat baru. Sistem secara otomatis akan menghasilkan token keamanan sepanjang 64 karakter heksadesimal (*hex*) yang harus diletakkan pada kode sumber *firmware* perangkat agar dapat terhubung dengan *server*.

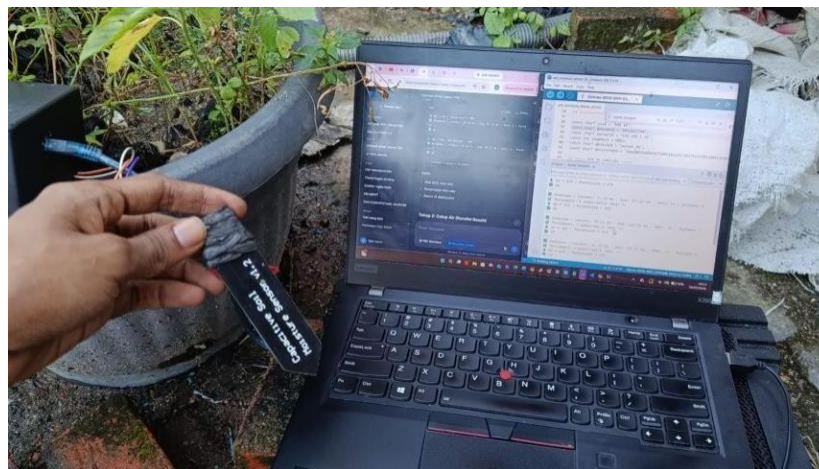


Gambar 4. Tampilan *dashboard web* utama setelah proses autentikasi, menampilkan visualisasi kartu perangkat secara *real-time*

Pada *Client IoT*, pengujian dilakukan menggunakan dua jenis node sensor, meliputi:

1. Node Sensor Kelembaban Tanah

Node Sensor yang digunakan adalah mikrokontroler ESP8266 yang dikonfigurasi untuk membaca nilai *Soil Moisture* melalui keluaran ADC. Perangkat ini diprogram untuk mentransmisikan data secara berkala setiap 5 detik. Rangkaian fisik node sensor ini disajikan pada Gambar 5.



Gambar 5. Rangkaian fisik node sensor kelembaban tanah berbasis ESP8266

2. Node Sensor *Passive Infrared Receiver* (PIR)

Node Sensor yang digunakan adalah ESP32 yang diintegrasikan dengan sensor PIR. Perangkat ini mengirimkan luaran digital biner, yakni bernilai 1 saat mendeteksi adanya pergerakan dan nilai 0 saat kondisi statis. Rangkaian fisik node sensor ini ditunjukkan pada Gambar 6.



Gambar 6. Rangkaian fisik node sensor PIR berbasis mikrokontroler ESP32

Node sensor PIR dibangun menggunakan mikrokontroler ESP32 yang diintegrasikan dengan sensor Passive Infrared Receiver (PIR). Sensor ini dikonfigurasi pada mode digital input untuk mendeteksi keberadaan objek bergerak di lingkungan sekitarnya. Keluaran sensor berupa sinyal biner, yaitu bernilai 1 (HIGH) ketika terdapat pergerakan dan bernilai 0 (LOW) pada kondisi statis.

Selama proses pengujian operasional berlangsung, setiap paket data yang dikirimkan oleh ESP8266 maupun ESP32 diproses secara real-time oleh *server*. *Server* mencatat seluruh transaksi data dalam bentuk *log* sistem yang terurut berdasarkan waktu. Indikator status [SUCCESS] pada *log* menandakan bahwa paket data dari client IoT telah berhasil diterima, divalidasi kecocokan token keamanannya, dan disimpan ke dalam basis data tanpa mengalami kegagalan. Setelah verifikasi awal ini, dilakukan pengujian komunikasi CoAP secara lebih mendetail untuk memverifikasi proses pertukaran data antara perangkat IoT dan *server*.

3.2 Hasil Komunikasi CoAP dan Identifikasi Log Server

Pengujian komunikasi CoAP dilakukan menggunakan mikrokontroler ESP8266 (Sensor Soil Moisture) dan ESP32 (Sensor PIR). Pada sistem yang dirancang, setiap transaksi data CoAP yang masuk diproses oleh *server* dan dicatat ke dalam *log* sistem dengan tag status [SUCCESS] untuk menandakan bahwa pemrosesan dan verifikasi permintaan berhasil dieksekusi oleh *runtime server*. Tampilan *log* transaksi penerimaan data CoAP dan verifikasi token secara *real-time* pada *console server* ditunjukkan pada Gambar 7.

```
[CoAP] received:
  From: 192.168.1.37:63133
  Hex: 40030077b973656e736f725f30320d736f696c5f6d6f69737475726546746f6b656e306461656438
  Version: 1, Type: 0, TKL: 0, Code: 0x3, MsgID: 119
  [PARSING] Options (String-based)...
  soil_moistureFtoken=daed8072d0644e75d4514ea21c4b27fe51f8124643123d769...
  [VALUE] Not JSON, reading raw bytes
  [DEVICE] sensor_02
  [SENSOR] soil_moisture
  [TOKEN] daed8072d0644e75d451...
  [VALUE] 76
  [RESULT] Device ID: sensor_02
  [RESULT] Sensor Key: soil_moisture
  [RESULT] Token: daed8072d0644e75d451...
  [RESULT] Value: 76
  [SUCCESS] sensor_02 -> soil_moisture = 76
  Broadcast: 1 sent, 1 skipped (filtered by user)
```

Gambar 7. Log Server Data Komunikasi CoAP dan Identifikasi Log Server

Format *log server* mendefinisikan seluruh status respon, baik permintaan yang berhasil disimpan maupun yang ditolak akibat autentikasi. Ringkasan hasil pengujian autentikasi dan identifikasi *log server* CoAP secara detail disajikan pada Tabel 1.

Tabel 1. Hasil Pengujian Autentikasi

No.	Device ID	Sensor	Status Token	Respon CoAP	Format Log Server	Keterangan Sistem
1	soil_node_1	soil_moisture	Valid	2.04 Changed	[SUCCESS] Data saved to SQLite	Data diterima & tersimpan
2	soil_node_1	soil_moisture	Tidak Valid	4.01 Unauthorized	+ ERROR: Invalid token for {deviceId}	Ditolak oleh sistem keamanan
3	pir_node_2	motion	Valid	2.04 Changed	[SUCCESS] Data saved to SQLite	Data gerakan tersimpan
4	pir_node_2	motion	Tanpa Token	4.01 Unauthorized	+ ERROR: Token missing	Ditolak oleh sistem keamanan

3.3 Hasil Pengujian Kinerja

Client IoT mengirim 300 paket dari ESP8266 ke *server*. Hasilnya peneliti rangkum di Tabel 2.

Tabel 2. Hasil Pengujian Kinerja

PARAMETER	HASIL	TARGET SPESIFIKASI
Rata-rata Latensi	115,56 ms	< 200 ms
Latensi Terendah	51,05 ms	-
Latensi Tertinggi	1.354,89 ms	-
Throughput Client	1 paket/detik	Stabil
Throughput Server	8,65 paket/detik	> 5 paket/detik
Keandalan	87,33%	> 80%

Data pada Tabel 2 menunjukkan bahwa nilai rata-rata latensi (115,56 ms) dan median (98,20 ms) berada di bawah target 200 ms. Namun, standar deviasi yang besar (145,30 ms) dan nilai maksimum yang tinggi (1.354,89 ms) mengindikasikan adanya variasi latensi yang signifikan, terutama disebabkan oleh proses *background server* (*backup database*) dan kondisi jaringan. Nilai persentil ke-95 (p95) sebesar 312,45 ms menunjukkan bahwa 95% dari total paket memiliki latensi di bawah 312,45 ms, yang masih berada dalam batas toleransi untuk aplikasi pemantauan non-kritis. Latensi terendah 51,05 ms terjadi pada kondisi jaringan stabil tanpa gangguan, sedangkan latensi tertinggi 1.354,89 ms terjadi ketika server menjalankan proses backup otomatis. Laju pengiriman *client* tercatat sebesar 0,2 paket/detik sesuai dengan interval pengiriman 5 detik. Sementara itu, kapasitas server mencapai 8,65 paket/detik, yang menunjukkan bahwa server mampu memproses paket jauh lebih cepat dari interval pengiriman.

3.4 Hasil Pengujian Keamanan

Untuk autentikasi JWT, peneliti uji tiga skenario. *Login* benar, token tersimpan di *local storage*. Token dihapus, *server* menolak dengan pesan "Sesi berakhir". Token dimodifikasi, *server* tolak dengan "Auth error: *Invalid Token*".

Untuk token perangkat IoT, peneliti juga melakukan tiga skenario pengujian. Pada skenario pertama, token yang valid berhasil menyimpan data ke *server*. Pada skenario kedua, token diubah satu karakter sehingga *server* menolak dengan pesan "ERROR: *Invalid token for sensor01*". Pada skenario ketiga, token tidak disertakan dalam permintaan sehingga *server* menolak dengan pesan "ERROR: *Token missing*".

Dari pengujian ini, peneliti menyimpulkan autentikasi dua lapis berfungsi dengan baik dalam memvalidasi identitas perangkat. Namun, pada skenario *replay attack*, *server* tetap menerima dan memproses paket duplikat yang berisi token valid, karena *server* tidak membedakan apakah paket tersebut adalah kiriman baru atau hasil *replay*. Hal ini menegaskan bahwa meskipun autentikasi token efektif mencegah akses dari perangkat tidak dikenal, sistem tetap rentan terhadap serangan *replay* di jaringan yang sama karena komunikasi CoAP tidak dienkripsi. Kelemahan ini merupakan konsekuensi dari desain sistem yang mengutamakan efisiensi dengan menghindari *overhead* DTLS.

3.5 Analisis

Berdasarkan hasil pengujian yang peneliti lakukan pada 300 sampel data, latensi rata-rata yang tercatat adalah 115,56 ms, dengan latensi terendah 51,05 ms dan tertinggi 1.354,89 ms. Dari pengalaman peneliti selama proses pengujian berlangsung, latensi tertinggi terjadi ketika *server* sedang menjalankan proses *background*, seperti *backup database* otomatis yang dijadwalkan setiap jam. Selain itu, gangguan pada jaringan Wi-Fi juga turut berkontribusi terhadap lonjakan latensi.

Untuk mengjustifikasi bahwa nilai latensi dan keandalan yang diperoleh tergolong baik, hasil penelitian ini dibandingkan dengan studi serupa. Penelitian oleh Amrullah et al. (2022) [12] yang menggunakan protokol MQTT untuk crowdsensing pada bidang kesehatan mencatat latensi rata-rata sekitar 180 ms. Penelitian oleh Sifawa et al. (2026) [13] pada jaringan *smart home* mencatat latensi rata-rata untuk protokol CoAP sebesar 203,5 ms. Dengan rata-rata latensi 115,56 ms, sistem yang dibangun dalam penelitian ini menunjukkan performa yang lebih baik dibandingkan kedua studi tersebut, yang mengindikasikan efisiensi komunikasi CoAP di jaringan lokal. Latensi yang lebih rendah ini dimungkinkan karena penggunaan protokol CoAP di atas UDP yang mengurangi *overhead* dibandingkan MQTT yang berjalan di atas TCP [3], [13].

Tingkat keandalan sistem mencapai 87,33% (262 paket berhasil dari 300 paket). *Packet loss* sebesar 12,67% ini terjadi karena penggunaan protokol UDP yang *connectionless* dan potensi gangguan pada medium Wi-Fi. Meskipun angka ini sebanding dengan temuan studi lain pada jaringan serupa [13], dampaknya terhadap aplikasi perlu dipertimbangkan. Untuk aplikasi pemantauan lingkungan non-kritis seperti pertanian atau deteksi gerakan sederhana, kehilangan data sesekali tidak berdampak

signifikan karena data dapat diinterpolasi atau diabaikan. Namun, untuk aplikasi kontrol *real-time*, packet loss sebesar ini memerlukan mekanisme retransmission atau redundansi data. Studi oleh Sifawa et al. (2026) [13] melaporkan tingkat packet loss untuk protokol CoAP pada jaringan Wi-Fi yang padat berkisar antara 10-15%, yang sebanding dengan hasil penelitian ini (12,67%).

Secara keseluruhan, sistem *cloud IoT* mandiri yang dibangun dalam penelitian ini sejalan dengan tren pengembangan platform IoT berbasis *cloud* yang independen. Yangui (2020) [14] mengidentifikasi bahwa *platform cloud* untuk IoT semakin bergeser ke arah solusi yang lebih terbuka dan dapat disesuaikan (*customizable*) untuk memenuhi kebutuhan spesifik pengguna. Penelitian ini merealisasikan tren tersebut dengan membangun platform berbasis PaaS yang sepenuhnya dapat dikontrol oleh pengguna, tanpa ketergantungan pada penyedia layanan pihak ketiga.

3.6 Keterbatasan Penelitian

Meskipun penelitian ini telah menunjukkan hasil yang menjanjikan, terdapat beberapa keterbatasan yang perlu diakui:

1. Pengujian DTLS, dimana penelitian ini mengadopsi autentikasi berbasis token URI-Query untuk menekan overhead pada application layer. Namun, evaluasi komparatif secara langsung mengenai konsumsi daya CPU, latensi handshake, dan throughput terhadap protokol CoAP berbasis DTLS (CoAPS) pada node sensor yang sama tidak diukur secara empiris dalam eksperimen ini. Klaim efisiensi dibandingkan DTLS masih didasarkan pada asumsi teoretis dan acuan literatur [5], [6].
2. Kerentanan *replay attack* disebutkan dalam skenario pengujian, meskipun token tidak valid atau hilang berhasil ditolak, pengujian awal dengan replay attack menunjukkan bahwa server tidak memiliki mekanisme pembeda untuk paket duplikat yang berisi token valid. Sistem rentan terhadap replay attack di jaringan yang sama karena token dikirim dalam plain text.
3. Skala pengujian dengan kinerja hanya dilakukan dengan dua node sensor (ESP8266 dan ESP32) secara simultan. Performa sistem saat menangani puluhan atau ratusan node secara bersamaan belum diuji dan memerlukan penelitian lebih lanjut.

4. KESIMPULAN

Penelitian ini telah berhasil merancang dan mengimplementasikan sistem cloud IoT mandiri berbasis *Platform as a Service* (PaaS) menggunakan protokol Constrained Application Protocol (CoAP). Sistem yang dibangun mampu menerima data dari perangkat ESP8266 dan ESP32 secara *real-time*, menyimpannya ke database *SQLite*, dan menampilkannya melalui *dashboard* berbasis *WebSocket*.

Mekanisme autentikasi dua lapis yang terdiri dari JWT untuk pengguna *dashboard* dan token unik untuk setiap perangkat IoT terbukti efektif mengamankan akses API dan transmisi data sensor, meskipun masih terdapat kerentanan terhadap *replay attack* akibat tidak digunakannya enkripsi. Hasil pengujian kinerja menunjukkan latensi rata-rata 115,56 ms, laju pengiriman client 0,2 paket/detik (dengan kapasitas server 8,65 paket/detik), dan tingkat keandalan 87,33% dari 300 paket data yang dikirim. Untuk pengembangan selanjutnya, disarankan penambahan enkripsi DTLS pada komunikasi CoAP, *deployment* ke *Virtual Private Server* (VPS) dengan domain publik, serta pengembangan aplikasi mobile untuk memperluas jangkauan akses pengguna.

DAFTAR PUSTAKA

- [1] I. E. Prasetya, S. Achmadi, and D. Rudhistiar, "Penerapan IoT (Internet of Things) Untuk Sistem Monitoring Air Dan Controlling Pada Kolam Gurami Berbasis Website," JATI (Jurnal Mahasiswa Teknik Informatika), vol. 6, no. 2, pp. 1184-1191, 2022. doi: [10.36040/jati.v6i2.5400](https://doi.org/10.36040/jati.v6i2.5400)
- [2] Erwin and A. Pratama, "Rancang bangun sistem monitoring suhu dan kelembaban ruang server berbasis IoT menggunakan Arduino pada PT Bintaro Serpong Damai," Jurnal Sistem Informasi

- dan Teknologi, vol. 5, no. 1, pp. 12-18, 2023. doi: [10.47970/siskom-kb.v7i1.453](https://doi.org/10.47970/siskom-kb.v7i1.453)
- [3] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," Internet Engineering Task Force (IETF), RFC 7252, 2014. Available: <https://datatracker.ietf.org/doc/html/rfc7252>
- [4] C. F. Permatasari and H. Dhika, "Optimasi Jalur Transfer Data dari HTTP menjadi MQTT pada IoT menggunakan Cloud Services," JISA (Jurnal Informatika dan Sains), vol. 1, no. 2, pp. 67-72, 2018. Available: <https://ojs.trilogi.ac.id/index.php/JISA/article/view/446/307>
- [5] S. Daharmi, A. Bhawiyuga, and A. A. Basuki, "Pengembangan IoT Cloud Platform berbasis pada Layanan *Serverless* Computing dengan menggunakan HTTP dan CoAP untuk komunikasi," Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, vol. 6, no. 3, pp. 1120-1127, 2022. Available: <https://ejurnal.seminar-id.com/index.php/josh/article/view/6075>
- [6] M. Shahrokhi and A. Ahmadi, "Power Evaluation of IoT Application Layer Protocols," IEEE Transactions on Green Communications and Networking, vol. 7, no. 2, pp. 890-898, 2023. doi: [10.1109/TGCN.2023.10365351](https://doi.org/10.1109/TGCN.2023.10365351)
- [7] ITU-T, "Y.2060: Overview of the Internet of things," ITU-T Recommendation, 2012. Available: <https://www.itu.int/rec/t-rec-y.2060-201206-i>
- [8] I. G. Widana and I. Nasution, "Sistem Informasi Berbasis Cloud: Solusi Untuk Bisnis Modern," Jurnal Teknologi dan Manajemen Informatika, vol. 11, no. 1, pp. 25-34, 2025. doi: [10.61722/jipm.v3i1.619](https://doi.org/10.61722/jipm.v3i1.619)
- [9] F. A. Nugraha and Y. A. Susetyo, "Analisis Perbandingan Performa Database Duckdb Dan Sqlite Pada Pengolahan Big Data," JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika), vol. 8, no. 3, pp. 1052-1060, 2023. doi: [10.29100/jipi.v8i3.4032](https://doi.org/10.29100/jipi.v8i3.4032)
- [10] P. Mahindrakar and U. Pujeri, "Insights of JSON Web Token," International Journal of Recent Technology and Engineering (IJRTE), vol. 8, no. 6, pp. 1707-1710, 2020. doi: [10.35940/ijrte.F7689.038620](https://doi.org/10.35940/ijrte.F7689.038620)
- [11] R. E. Susanti, A. W. Muhammad, and W. A. Prabowo, "Implementasi Intrusion Prevention System (IPS) OSSEC dan Honeypot Cowrie," Jurnal Sisfokom (Sistem Informasi dan Komputer), vol. 11, no. 1, pp. 73-78, 2022. doi: [10.32736/sisfokom.v11i1.1246](https://doi.org/10.32736/sisfokom.v11i1.1246)
- [12] A. Amrullah, M. Udin, H. Al Rasyid, and I. Winarno, "Implementasi dan Analisis Protokol Komunikasi IoT untuk Crowdsensing pada Bidang Kesehatan," Jurnal Inovtek Polbeng - Seri Informatika, vol. 7, no. 1, pp. 122-135, 2022. Available: <https://www.researchgate.net/publication/367468404>
- [13] H. A. Sifawa, A. B. Tambawal, and S. M. S. Tudu, "Performance Evaluation of IoT Protocols in

- Resource-Constraints Smart Home Networks," International Journal of Computer Applications, vol. 12, no. 1, pp. 48-61, 2026.
- [14] S. Yangui, "A Panorama of Cloud Platforms for IoT Applications Across Industries," Sensors, vol. 20, no. 9, p. 2701, 2020, doi: [10.3390/s20092701](https://doi.org/10.3390/s20092701)
- [15] A. Dauda, O. Flauzac, and F. Nolot, "A Survey on IoT Application Architectures," Sensors, vol. 24, no. 16, p. 5320, 2024, doi: [10.3390/s24165320](https://doi.org/10.3390/s24165320)
- [16] M. B. Al-Hadidi, et al., "A Comprehensive Survey on Security in IoT: Challenges, Solutions, and Future Directions," IEEE Access, vol. 11, pp. 124567-124589, 2023, doi: [10.1109/ACCESS.2023.3328912](https://doi.org/10.1109/ACCESS.2023.3328912)
- [17] A. M. Zungeru, et al., "Resource Management in Internet of Things: A Survey," Journal of Network and Computer Applications, vol. 201, p. 103313, 2022, doi: [10.1016/j.jnca.2022.103313](https://doi.org/10.1016/j.jnca.2022.103313)