BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian Daharmi et al. (2022) dalam judul Pengembangan IoT *Cloud* Platform berbasis pada Layanan *Server*less Computing dengan menggunakan HTTP dan CoAP untuk komunikasi menunjukkan bahwa *server*less computing lebih hemat biaya dan dapat meningkatkan fleksibilitas aplikasi IoT *cloud*, menjadikannya pilihan yang menarik untuk pengelolaan perangkat IoT secara efisien. Penelitian yang membandingkan performa protokol Hypertext Transfer Protocol (HTTP) dan Constrained Application Protocol (CoAP) dalam komunikasi IoT telah dilakukan oleh beberapa peneliti. Salah satu studi oleh Shahrokhi & Ahmadi (2023) mengevaluasi konsumsi daya dari protokol: protokol lapisan aplikasi IoT, termasuk HTTP dan CoAP. Hasil penelitian menunjukkan bahwa CoAP lebih efisien dalam konsumsi daya dibandingkan dengan HTTP, menjadikannya lebih sesuai untuk perangkat IoT dengan sumber daya terbatas.

Firmansyah et al. (2021) berhasil mengembangkan prototipe aplikasi kendali lampu rumah jarak jauh menggunakan protokol CoAP. Sistem ini memanfaatkan mikrokontroler Arduino Mega 2560 dengan modul ESP8266 dan Ethernet Shield. Pengelolaan komunikasi dilakukan dengan library microcoap, dan data dikirim ke *server* melalui koneksi jaringan lokal menggunakan kabel UTP dengan konektor RJ-45. Pengguna dapat mengontrol sistem melalui *browser* Chromium menggunakan ekstensi COPPER CU. Penelitian ini menunjukkan penerapan sukses protokol CoAP dalam kendali perangkat IoT secara remote. Yangui (2020) meninjau berbagai Platform as-a-Service (PaaS) untuk aplikasi IoT di industri, dengan fokus pada Industrial IoT (IIoT). Penelitian ini menyatakan bahwa penerapan IIoT dalam lingkungan *cloud* memerlukan desain yang tepat untuk PaaS guna mendukung pengembangan, penyebaran, dan manajemen aplikasi IIoT. Hal ini penting untuk membuat pengelolaan perangkat IoT secara efisien, terutama dengan menggunakan pendekatan berbasis PaaS seperti yang digunakan dalam sistem *cloud* untuk aplikasi IoT.

Dalam peneltian Widana dan Nasution (2025) meneliti *cloud* computing dalam mengurangi biaya dan meningkatkan inovasi di sektor bisnis. Mereka

menemukan bahwa *cloud* computing bisa membuat pengurangan biaya operasional dan mendukung inovasi melalui teknologi seperti AI, ML, dan IoT. Protokol yang digunakan dalam *cloud* computing, seperti HTTP, MQTT, dan CoAP, menjadikan pengelolaan IoT secara lebih efisien. Implementasi Constrained Application Protocol (CoAP) pada Semantic IoT Web Service. Penelitian ini berhasil mengimplementasikan protokol CoAP pada Semantic IoT Web Service, dimana penerimaan berbagai data seperti JSON, gambar, dan video. CoAP dan HTTP dapat bekerja baik secara terpisah maupun bersama, mengatasi masalah interoperabilitas, terutama syntactical interoperability, antara node sensor dan *server*. Implementasi ini relevan dalam sistem *Cloud* IoT berbasis PaaS yang memanfaatkan protokol CoAP, memberikan solusi untuk efisiensi dan keamanan data IoT.

Amrullah et al. (2022) menjelaskan bahwa dalam penelitian yang berjudul Implementasi dan Analisis Protokol Komunikasi IoT untuk Crowdsensing pada Bidang Kesehatan, diusulkan arsitektur gateway multi: protokol yang mengintegrasikan MQTT, HTTP, dan CoAP untuk mendukung crowdsensing di sektor kesehatan. Hasil pengujian menunjukkan bahwa CoAP menawarkan kinerja terbaik dalam throughput, sementara MQTT unggul dalam mengurangi delay. Temuan ini sangat relevan dengan pengembangan sistem *Cloud* IoT berbasis PaaS, yang dapat meningkatkan efisiensi pengelolaan data IoT dan memastikan interoperabilitas antar protokol komunikasi. Setiawan, dkk (2022) menjelaskan bahwa menjelaskan bahwa *cloud* computing dengan model layanan PaaS, dapat mengelola aplikasi IoT secara efisien. Karakteristik *cloud* seperti elastisitas dan layanan terukur mendukung kebutuhan sistem IoT yang skalabel (hal. 35–37). Oleh karena itu, penggunaan protokol CoAP dalam sistem ini dapat menyediakan komunikasi yang lebih hemat bandwidth dan sesuai dengan prinsip efisiensi *cloud* computing.

Tabel 2. 1 Penelitian Terdahulu

Penelitian	Judul	Variabel yang Diteliti	Metode yang Diteliti / Hasil
Daharmi, Bhawiyuga, & Basuki (2022)	Pengembangan IoT *Cloud* Platform berbasis pada Layanan *Server*less Computing	Studi kasus dan analisis teknologi CoAP dibandingkan HTTP untuk perangkat dengan keterbatasan sumber daya.	Hasil penelitian menunjukkan bahwa protokol CoAP lebih efisien daripada HTTP untuk perangkat dengan

Penelitian	Judul	Variabel yang Diteliti	Metode yang Diteliti / Hasil
			keterbatasan sumber daya, terutama pada platform IoT *Cloud* yang menggunakan layanan *server*less.
Shahrokhi & Ahmadi (2023))	Power Evaluation of IOT Application Layer Protocols (Evaluasi Konsumsi Daya Protokol Lapisan Aplikasi IoT)	Konsumsi daya pada protokol lapisan aplikasi (HTTP dan CoAP) di perangkat IoT.	Pengujian konsumsi daya pada perangkat IoT menunjukkan perbandingan kinerja antara protokol HTTP dan CoAP.
Firmansyah et al. (2021)	Internet of Things Prototype House Light	Penggunaan protokol CoAP untuk kendali perangkat IoT secara remote.	Penelitian ini mengimplementasikan mikrokontroler Arduino Mega 2560, ESP8266, dan Ethernet Shield dengan library microcoap untuk komunikasi berbasis CoAP.
Yangui (2020)	A Panorama of *Cloud* Platforms for IoT Applications Across Industries	Desain *Platform as a Service*(PaaS) yang optimal untuk mendukung pengembangan, penyebaran, dan pengelolaan aplikasi perangkat Industrial IoT.	Analisis desain dilakukan dengan pendekatan berbasis *cloud* untuk mendukung berbagai aplikasi IoT di industri.
Widana dan Nasution (2025)	Sistem Informasi Berbasis *Cloud*: Solusi Untuk Bisnis Modern	Efisiensi biaya operasional, inovasi teknologi *cloud* computing (AI, ML), model bayar sesuai pemakaian, skalabilitas dinamis, serta mitigasi tantangan keamanan dan pengendalian biaya.	Studi literatur dilakukan untuk menganalisis potensi *cloud* computing dalam mendukung sistem informasi bisnis modern.
Putra et al. (2019)	Implementasi Constrained Application Protocol (CoAP) Pada Semantic IoT Web Service	Penggunaan protokol CoAP dan HTTP dalam komunikasi IoT, interoperabilitas, jenis data (JSON, gambar, video), serta fokus pada enkripsi data dan komunikasi.	Penelitian ini mengimplementasikan CoAP pada Semantic IoT untuk menangani penerimaan data dan komunikasi yang lebih aman.
Amrullah et al. (2022)	Implementasi dan Analisis Protokol Komunikasi IoT untuk Crowdsensing pada Bidang	Arsitektur gateway multi protokol yang mencakup CoAP, MQTT, dan HTTP untuk crowdsensing di sektor kesehatan.	Pengujian kinerja protokol dilakukan dengan menganalisis through put dan delay dalam konteks aplikasi

Penelitian	Judul	Variabel yang Diteliti	Metode yang Diteliti / Hasil
	Kesehatan		kesehatan.
Setiawan et al. (2022)	Analisa Layanan *Cloud* Computing Di Era Digital	Efisiensi pengelolaan aplikasi IoT menggunakan layanan *cloud* yang mendukung elastisitas, layanan terukur, dan skalabilitas aplikasi IoT.	Analisis dilakukan terhadap karakteristik *cloud* computing, termasuk elastisitas dan kemampuan layanan terukur untuk mendukung pengelolaan aplikasi IoT.
Shelby et a.. (2014)	The Constrained Application Protocol (CoAP) — RFC 7252	Spesifikasi teknis protokol CoAP, format pesan, metode request/response, dan mekanisme keandalan	Dokumentasi resmi IETF yang menjadi dasar teori dan implementasi protokol CoAP dalam penelitian ini. Menjelaskan arsitektur berlapis CoAP, jenis pesan CON/NON/ACK/RST, serta kode respons standar seperti 2.04 Changed.
Erlangga Prasetya et al.2022	Penerapan Iot (Internet Of Things) Untuk Sistem Monitoring Air Dan Controlling Pada Kolam Ikan Gurami Berbasis Website	Model komunikasi client-*server* pada sistem IoT, peran perangkat IoT sebagai client dan *cloud* sebagai *server*	Menjelaskan bahwa perangkat IoT bertindak sebagai client dan sistem *cloud* sebagai *server* dalam model komunikasi client-*server*.
Erwin & Pratama (2023)	Pengembangan IoT *Cloud* Platform berbasis pada Layanan *Server*less Computing	Studi kasus dan analisis teknologi CoAP dibandingkan HTTP untuk perangkat dengan keterbatasan sumber daya	Hasil penelitian menunjukkan bahwa protokol CoAP lebih efisien daripada HTTP untuk perangkat dengan keterbatasan sumber daya, terutama pada platform IoT *Cloud* yang menggunakan layanan *server*less.

2.2 Landasan Teori

2.2.1 Internet of Things

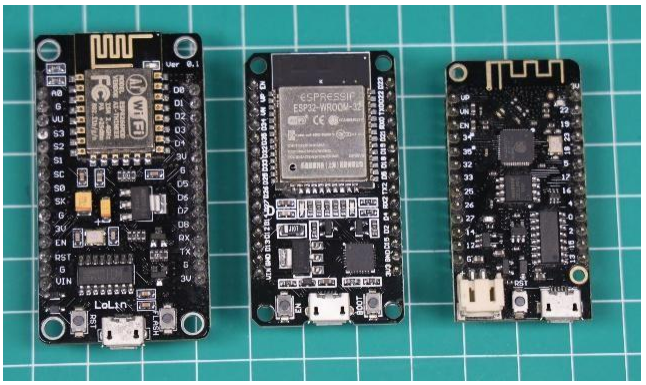
Menurut Rouse (2019, sebagaimana dikutip dalam Rusnawati & Hariyati, 2022) mendefinisikan *Internet of Things* (IoT) sebagai sistem perangkat komputasi yang saling terkait, mencakup mesin mekanik dan digital, objek, serta individu yang diberi pengidentifikasi unik dan memiliki kemampuan untuk mentransfer data melalui jaringan tanpa memerlukan interaksi manusia ke manusia atau manusia ke

komputer.

Sementara itu, Banafa (2016, sebagaimana dikutip dalam Rusnawati & Hariyati, 2022) mendefinisikan IoT sebagai "platform yang terdiri dari perangkat keras dan perangkat lunak yang berinteraksi secara mulus untuk menghubungkan hal-hal sehari-hari ke internet, sehingga memungkinkan pengumpulan dan pertukaran informasi" (hlm. 570).

2.2.2 Mikrokontroler

Mikrokontroler adalah sistem komputer sederhana yang terintegrasi dalam satu chip. Di dalamnya terdapat berbagai komponen pendukung, seperti unit pemrosesan, ROM, RAM, input/output (I/O), dan clock, yang memiliki fungsi serupa dengan komponen pada komputer pribadi (Widiyanto & Fitriyani, 2021).

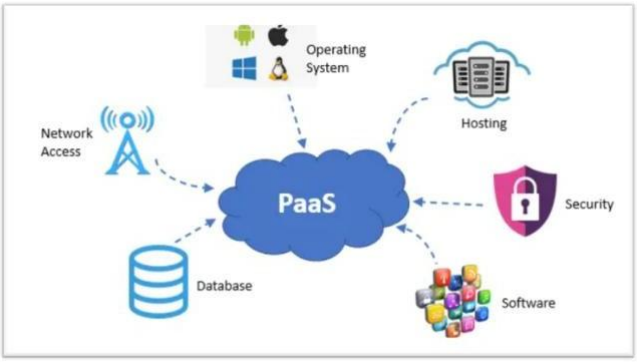


Gambar 2. 1 Mikrokontroller
(Sumber: Elektrologi, 2021, https://elektrologi.iptek.web.id/perbedaan-esp32-dan-esp8266/)

2.2.3 Platform as a Service(PaaS)

Platform as a Service(PaaS) adalah jenis model layanan *cloud* computing yang menawarkan platform *cloud* yang fleksibel dan skalabel untuk mengembangkan, men-deploy, menjalankan, serta mengelola aplikasi. PaaS menyediakan semua yang diperlukan *developer* untuk pengembangan aplikasi tanpa harus repot mengupdate sistem operasi dan alat pengembangan atau memelihara hardware. Sebagai gantinya, seluruh lingkungan PaaS atau platform dikirimkan oleh penyedia layanan pihak ketiga melalui *cloud* (Google *Cloud*, n.d.). Dalam perancangan sistem *cloud* IoT ini, platform PaaS dibangun secara mandiri menggunakan Node.js sebagai *runtime server* dan SQLite sebagai basis data Platform ini berjalan secara lokal (localhost) pada laptop peneliti sebagai lingkungan pengembangan, sehingga pengelolaan data dan keamanan informasi dapat dikendalikan sepenuhnya tanpa ketergantungan pada penyedia layanan

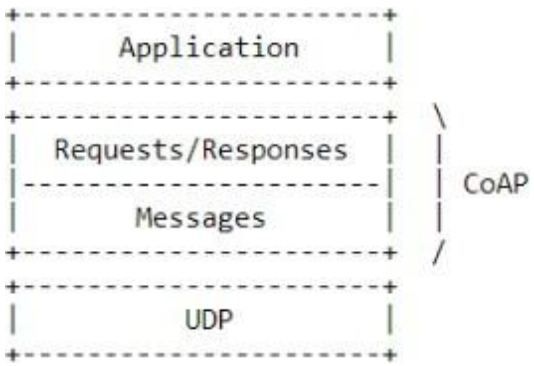
eksternal.



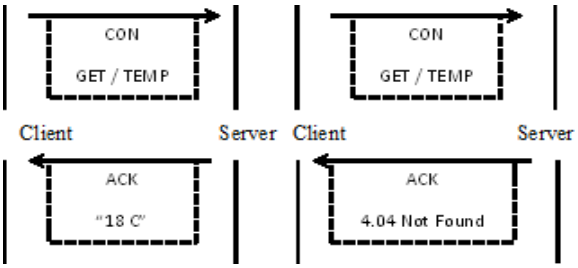
Gambar 2. 2 Platform as a Service
(Sumber: https://we*cloud*data.com/blog/platform-as-a-service-paas/)

2.2.4 Constrained Application Protocol (CoAP)

Constrained Application Protocol (CoAP) adalah protokol komunikasi yang dirancang untuk perangkat IoT dengan sumber daya terbatas *(constrained networks)*. CoAP dikembangkan oleh IETF CoRE Working Group dan beroperasi di atas protokol UDP *(User Datagram Protocol)* (Alhaj Ali, 2018). Protokol CoAP ini adalah protokol yang alternatif ringan untuk HTTP yang mempertahankan konsep-konsep utama arsitektur REST (*Representational State Transfer*), seperti identifikasi sumber daya melalui URI, metode-metode HTTP (GET, POST, PUT, DELETE), dan tipe media, namun dioptimalkan untuk lingkungan dengan sumber daya terbatas.



Gambar 2. 3 lapisan Abstrak Protokol CoAP
(Sumber: Shelby et al., 2014, p. 10)



Gambar 2. 4 Elemen Protokol CoAP
(Sumber: Shelby et al., 2014, pp. 12–13)

Dalam melakukan penelitian ini, maka peneliti menguraikan tentang judul PA ini yang menggunakan Protokol CoAP sebagai aspek utama mencakup:

1) Arsitektur Berlapis CoAP

CoAP disusun dalam dua lapisan abstraksi guna menangani komunikasi asinkron di atas UDP dan menyediakan mekanisme keandalan opsional.

a. *Message Layer* (Lapisan Pesan)

Adalah Lapisan ini menangani pertukaran pesan asinkron melalui UDP dan bertanggung jawab atas keandalan transmisi, deteksi duplikasi pesan, dan mekanisme retransmission. CoAP mendefinisikan empat jenis pesan pada lapisan ini, yaitu:

1. *Confirmable* (CON)

Pesan yang memerlukan konfirmasi. Jika pesan CON tidak diakui, maka akan diulang secara otomatis dengan penundaan pengiriman hingga batas maksimum.

2. *Non-confirmable* (NON)

Pesan yang tidak memerlukan konfirmasi (fire-and-forget). Cocok untuk data telemetri periodik di mana kehilangan sesekali masih dapat diterima.

3. *Acknowledgement* (ACK)

Pesan konfirmasi untuk pesan CON. ACK dapat membawa respons (piggybacked response) atau hanya konfirmasi kosong (empty ACK) jika *server* memerlukan waktu lebih lama untuk memproses (separate response).

4. *Reset* (RST)

Pesan yang menandakan bahwa pesan yang diterima tidak dapat diproses, misalnya karena konteks yang hilang pada simpul penerima.

b. *Request/Response Layer* (Lapisan Meminta/Respon)

Lapisan ini bertanggung jawab atas transmisi permintaan dan respons dengan menggunakan kode metode dan kode respons, serta untuk manipulasi sumber daya. Lapisan ini mengikuti model interaksi client/*server* dan protokol CoAP ini mendukung metode-metode REST yang setara dengan HTTP.

1. GET: Untuk mengambil representasi dari suatu sumber daya.
2. POST: Berperan membuat sumber daya baru atau memperbarui sumber daya yang ada.
3. PUT: Sebagai memperbarui sumber daya dengan representasi yang disertakan.
4. DELETE: Untuk menghapus suatu sumber daya.

Dari lapisan yang ada, CoAP merupakan protokol tunggal dimana fitur messaging dan request/response merupakan bagian dari header CoAP itu sendiri (Shelby et al., 2014, pp. 10–12).

2.2.5 Perbandingan CoAP dengan HTTP dan MQTT

Pemilihan protokol komunikasi dalam sistem IoT sangat bergantung pada kebutuhan aplikasi, karakteristik perangkat, dan kondisi jaringan. CoAP, HTTP, dan MQTT adalah tiga protokol yang paling umum digunakan dalam komunikasi IoT. Protokol CoAP dirancang khusus untuk perangkat *constrained* (terbatas) dengan *overhead* yang sangat rendah dan dioptimalkan untuk perangkat berdaya rendah. Sementara itu, HTTP memiliki *overhead* yang tinggi dan kurang cocok untuk perangkat dengan sumber daya terbatas, sedangkan MQTT menggunakan model *publish/subscribe* dengan *broker* pusat yang ideal untuk sistem berbasis *event* dengan banyak node (Dauda et al., 2025). Perbandingan ketiga protokol ini dilihat pada Tabel 2.2.

Tabel 2. 2 Perbandingan Protokol CoAP, HTTP dan MQTT

Parameter	CoAP	HTTP	MQTT
Model Komunikasi	Request/Response	Request/Response	Publish/Subscribe
Protokol Transport	UDP	TCP	TCP
Overhead	Sangat rendah	Tinggi	Rendah
Efisiensi Daya	Sangat tinggi	Rendah	Tinggi
Reliabilitas	CON/NON (opsional)	Bawaan TCP	3 level QoS
Keamanan	DTLS	TLS	TLS
Arsitektur	Point-to-point	Client-*Server*	Broker-based
Kesesuaian	Perangkat sangat terbatas	Perangkat berdaya besar	Jaringan tidak stabil

Mekanisme komunikasi CoAP mengikuti model request-response antara client dan *server*. Client mengirimkan permintaan ke *server*, kemudian *server* memproses permintaan tersebut dan mengembalikan tanggapan.

2.2.6 Node.js

Node.js adalah runtime environment untuk menjalankan kode JavaScript di sisi *server*. Node.js menggunakan arsitektur event-driven dan non-blocking I/O, sehingga cocok untuk aplikasi real-time seperti IoT (Official Node.js Documentation, 2024).

Gambar 2. 5 Node.js

(Sumber: https://www.svgrepo.com/svg/303360/nodejs-logo)

Dalam proyek ini, Node.js digunakan untuk membangun tiga *server* sekaligus:

1) *Server* HTTP (port 3000) untuk menyajikan *dashboard* dan API.
2) *Server* CoAP (port 5683) untuk menerima data dari perangkat IoT.
3) *Server WebSocket* (port 3000) untuk broadcast data real-time.

Node.js dipilih karena ekosistem library yang kaya, seperti express untuk HTTP, ws untuk *WebSocket*, dan coap-packet untuk parsing CoAP.

2.2.7 SQLite

SQLite adalah sebuah sistem manajemen basis data relasional yang bersifat *server*less dan embedded, yang berarti *database* ini tidak memerlukan *server* terpisah untuk beroperasi dan tertanam langsung ke dalam aplikasi yang menggunakannya Nugraha & A. Susetyo (2023).

Gambar 2. 6 SQlite *Database*

(Sumber. Wikipedia Bahasa Indonesia.

(https://id.wikipedia.org/wiki/Berkas:SQLite370.svg)

SQLite *database* bersifat ringan yang tidak memerlukan *server* terpisah. SQLite menyimpan data dalam satu file dan mendukung perintah SQLite. SQLite cocok untuk proyek IoT skala kecil hingga menengah karena ukurannya kecil, mudah diintegrasikan dengan Node.js, dan tidak memerlukan konfigurasi rumit. Dalam proyek ini, SQLite menyimpan data pengguna, perangkat, sensor, aturan automasi, jadwal, dan notifikasi dalam file iot.db.

Karakteristik ini menjadikan SQLite sangat ringan, dengan seluruh data disimpan dalam satu file tunggal .db, sehingga cocok untuk perangkat dengan sumber daya terbatas, termasuk perangkat IoT dan sistem embedded seperti mikrokontroler ESP8266.

2.2.8 Autentikasi Berbasis Token

Autentikasi berbasis token adalah metode verifikasi identitas pengguna atau perangkat tanpa menyimpan status sesi di *server*. Token merupakan potongan kode yang berisi informasi sensitif klien yang diperlukan untuk autentikasi dan otorisasi sebelum mengambil data dari *server*, yang bertujuan untuk mencegah akses ilegal terhadap data sensitif (Mahindrakar & Pujeri, 2020). Setelah login berhasil, *server* menerbitkan token yang harus disertakan dalam setiap permintaan berikutnya.

Gambar 2. 7 JWT Web Token Sumber:
(https://medium.com/@majdsoubh/understanding-jwt-json-web-token-177ce1e7d38a)

JWT (JSON Web Token) adalah format token yang terdiri dari tiga bagian, yaitu header, *payload*, dan signature. *Payload* berisi informasi pengguna seperti id, *username*, dan role. Dalam proyek ini, JWT digunakan untuk autentikasi pengguna *dashboard*, sedangkan token unik digunakan untuk autentikasi perangkat IoT.

2.2.9 Keamanan Data dalam IoT

Keamanan data menjadi perhatian utama dalam sistem Internet of Things (IoT) karena data dikirim melalui jaringan yang rentan terhadap berbagai ancaman. Beberapa ancaman umum dalam komunikasi IoT antara lain serangan *DDoS (Distributed Denial of Service)* yang membanjiri *server* hingga tidak dapat diakses, serangan *MITM (Man-in-the-Middle)* yang menyadap komunikasi antara perangkat dan *server*, serta peretasan perangkat *(device hijacking)* yang mengambil alih kendali perangkat IoT (Susanti et al., 2022).

Dalam proyek ini, keamanan diterapkan melalui beberapa tahapan mekanisme yang dipasangkan. Berikut tabel 2.1.

Tabel 2. 3 Mekanisme Keamanan

Mekanisme Keamanan	Fungsi
Autentikasi JWT	Memverifikasi pengguna *dashboard* sebelum mengakses API
Token unik per perangkat	Memastikan hanya perangkat terdaftar yang dapat mengirim data
Hash password (*bcrypt*)	Menyimpan *password* dalam bentuk terenkripsi, bukan teks jelas
Validasi token pada setiap request	Mencegah akses dari pengguna atau perangkat tidak terdaftar

2.2.10 Pengujian Komunikasi CoAP

Pengujian sistem dilakukan untuk memastikan komunikasi antara perangkat IoT dan *server* berjalan dengan baik. Pengujian difokuskan pada tiga aspek utama: *fungsionalitas, throughput, dan latensi*. Pengujian performa merupakan aspek penting dalam pengembangan sistem IoT untuk memastikan bahwa infrastruktur komunikasi yang dibangun dapat beroperasi secara efisien dan andal. Dalam konteks protokol CoAP, pengujian performa umumnya difokuskan pada beberapa metrik kunci yang mencerminkan kualitas layanan, seperti throughput, latensi, dan keandalan transmisi.

Hal ini sejalan dengan penelitian (Sifawa et al., 2026) yang mengevaluasi empat protokol IoT (CoAP, MQTT, AMQP, dan HTTP) dalam jaringan rumah pintar dan menemukan bahwa CoAP mencapai latensi terendah dan konsumsi energi terendah dibandingkan protokol lainnya, menjadikannya sangat cocok

untuk perangkat berdaya rendah dan aplikasi yang sensitif terhadap penundaan. Berdasarkan hal tersebut, pengujian dalam penelitian ini difokuskan pada tiga aspek utama: fungsionalitas, throughput, dan latensi.