

ANALISIS PERBANDINGAN KINERJA PROTOKOL HTTP, WEBSOCKET, DAN MQTT PADA PLATFORM DETEKSI HAMA BERBASIS ARSITEKTUR MICROSERVICES

Muhammad Khairu Mufid

Program Studi Teknik Multimedia dan Jaringan, Jurusan Teknik Informatika dan Komputer, Politeknik Negeri Jakarta

Depok, Jawa Barat

muhammad.khairu.mufid.tik22@mhswn.pnj.ac.id

Abstract - This research aims to evaluate the performance of HTTP, WebSocket, and MQTT protocols within a microservices architecture to overcome computational bottlenecks on edge computing devices when continuously transmitting hydroponic image data. An experimental quantitative method was used by measuring TIPHON-standardized Quality of Service (QoS) and operational reliability. The results show all protocols have a Packet Loss below 1%. MQTT proved to be the most optimal with the highest Throughput (153.92 Kbps) and a competitive Latency (26.10 ms). Conversely, HTTP was the slowest (27.71 ms) due to application-layer bloat and synchronous mechanisms, while WebSocket produced the highest Jitter (78.22 ms) due to XOR Masking encryption burdens on the microcontroller. Furthermore, the microservices infrastructure proved immune to cascading failures with a pure recovery time ranging from 5.019 to 8.044 seconds. In conclusion, the integration of MQTT and microservices architecture produces a reliable, scalable, and highly fault-tolerant smart farming platform.

Keywords: Edge Computing, Hydroponics, Internet of Things, Microservices Architecture, MQTT, Quality of Service

Abstrak-- Penelitian ini bertujuan mengevaluasi kinerja protokol HTTP, WebSocket, dan MQTT pada arsitektur microservices untuk mengatasi hambatan komputasi pada perangkat edge computing saat mengirimkan data gambar hidroponik secara kontinu. Metode yang digunakan adalah kuantitatif eksperimental melalui pengukuran Quality of Service (QoS) standar TIPHON dan uji keandalan operasional. Hasil pengujian menunjukkan seluruh protokol memiliki Packet Loss di bawah 1%. MQTT terbukti paling optimal dengan Throughput tertinggi (153.92 Kbps) dan Latency bersaing (26.10 ms). Sebaliknya, HTTP mencetak waktu paling lambat (27.71 ms) akibat pembengkakan lapisan aplikasi dan mekanisme sinkron, sementara WebSocket menghasilkan fluktuasi Jitter tertinggi (78.22 ms) akibat beban komputasi enkripsi XOR Masking pada mikrokontroler. Infrastruktur microservices terbukti kebal terhadap kegagalan berantai dengan waktu pemulihan murni berkisar 5.019 hingga 8.044 detik. Kesimpulannya, integrasi MQTT dan arsitektur microservices menghasilkan platform smart farming yang andal, skalabel, dan bertoleransi kesalahan tinggi.

Kata kunci: Arsitektur Microservices, Edge Computing, Hidroponik, Internet of Things, MQTT, Quality of Service.

I. PENDAHULUAN

Keberhasilan sistem pertanian modern, khususnya pada metode hidroponik, sangat bergantung pada pengawasan berkelanjutan terhadap stabilitas lingkungan tumbuhnya. Sedikit saja fluktuasi lingkungan atau masuknya anomali biologis dapat berdampak fatal pada kualitas dan kuantitas hasil panen [1]. Salah satu ancaman operasional paling merugikan adalah serangan

hama, di mana intensitas kerusakannya dapat menyebar secara cepat hingga memicu tingkat kerusakan sebesar 60,7% dari total populasi tanaman [2]. Pemantauan hama secara manual oleh pekerja di fasilitas *greenhouse* terbukti sangat tidak efisien dan rentan terhadap keterlambatan penanganan akibat keterbatasan penglihatan manusia. Oleh karena itu, otomatisasi sistem pengawasan hama menjadi sebuah keharusan untuk menjaga keberlangsungan produksi.

Penerapan teknologi *Internet of Things* (IoT) yang dipadukan dengan kecerdasan buatan telah menjadi standar baru dalam menciptakan sistem peringatan dini yang beroperasi secara *real-time* dan meminimalkan ketergantungan pada tenaga manusia [3]. Implementasi pengolahan citra pada tingkat perangkat keras telah diteliti sebelumnya, namun eksekusi model kecerdasan buatan secara lokal pada *edge device* terbukti memicu *hardware bottleneck* dengan penurunan *frame rate* yang parah [4]. Ketika arsitektur sistem ini diekspansi untuk mendistribusikan data gambar secara kontinu menuju *server cloud microservices*, perangkat *edge* dituntut memikul beban ganda (*dual workload*), yakni mempertahankan inferensi visual dan mengeksekusi transmisi jaringan secara serentak. Kondisi ini memunculkan urgensi pemilihan protokol komunikasi yang memiliki efisiensi komputasi dan *Quality of Service* (QoS) tertinggi. Penelitian terdahulu membuktikan bahwa protokol *persistent* seperti WebSocket mampu menekan waktu respons komunikasi IoT [5]. Di sisi lain, protokol MQTT secara fundamental dikonfirmasi lebih unggul dalam meminimalkan latensi dan memangkas waktu tempuh transmisi dibandingkan WebSocket maupun HTTP [6], [7].

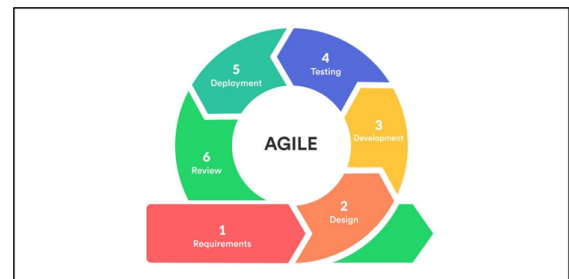
Meskipun karakteristik keunggulan protokol-protokol tersebut telah banyak diteliti, terdapat satu kesenjangan ilmiah yang krusial pada riset-riset sebelumnya. Mayoritas kajian komparatif terdahulu hanya menguji protokol menggunakan muatan data telemetry sensor lingkungan berskala sangat ringan. Belum ada pembuktian dan validasi empiris mengenai kinerja jaringan serta degradasi protokol saat HTTP, WebSocket, dan MQTT dipaksa memikul transmisi data gambar berukuran besar secara terus-menerus. Ketiadaan analisis pada skenario transmisi citra ini menjadi permasalahan (*delta*) yang harus diselesaikan, mengingat inefisiensi seperti pembengkakan teks (*application-layer bloat*) dan mekanisme sinkron (*half-duplex*) pada HTTP, serta penalti komputasi kriptografi (*XOR Masking*) pada WebSocket, sering kali terabaikan pada pengujian data berskala kecil.

Berdasarkan kesenjangan tersebut, permasalahan yang dikaji dalam penelitian ini adalah bagaimana perbandingan performa QoS antara protokol HTTP, WebSocket, dan MQTT pada transmisi data gambar secara kontinu, serta bagaimana tingkat keandalan operasional (*reliability*) dari arsitektur *microservices* saat dihadapkan pada kegagalan dependensi. Oleh

karena itu, penelitian ini bertujuan untuk mengevaluasi dan membandingkan kinerja ketiga protokol tersebut secara empiris berdasarkan parameter berstandar TIPHON, meliputi *Throughput*, *Packet Loss*, *End-to-End Latency*, dan *Jitter*. Selain itu, penelitian ini juga bertujuan untuk menguji keandalan arsitektur *microservices* dalam memulihkan aliran data secara otonom (*self-healing*). Analisis dari penelitian ini diharapkan dapat merumuskan standar arsitektur telemetry visual yang paling optimal, berskalabilitas tinggi, dan memiliki *fault-tolerance* untuk operasional *greenhouse* hidroponik.

II. II. METODOLOGI PENELITIAN

Penelitian ini menggunakan pendekatan kuantitatif dengan jenis riset eksperimental untuk mengevaluasi kinerja transmisi data protokol HTTP, WebSocket, dan MQTT. Pengujian dieksekusi menggunakan infrastruktur jaringan nirkabel Wi-Fi 5G berkapasitas tinggi untuk memastikan saluran fisik tidak menjadi faktor penghambat, sehingga perbandingan performa murni didasarkan pada karakteristik bawaan arsitektur masing-masing protokol pada lapisan aplikasi. Pada tahap perancangan dan pengembangan perangkat lunak sistem pemantauan, penelitian ini mengadopsi metode *Agile* [9]. Siklus *Agile* yang diterapkan divisualisasikan pada Gambar 1



Gambar 1. Siklus Pengembangan Perangkat Lunak Metode Agile

Berdasarkan Gambar 1, pengembangan sistem dieksekusi secara iteratif melalui tahapan *Requirements* untuk mengumpulkan spesifikasi teknis jaringan, *Design* untuk merancang topologi komunikasi data di dalam *docker bridge*, dan *Development* untuk penulisan kode sumber. Modul yang selesai kemudian dievaluasi pada tahap *Testing* menggunakan pengujian *Blackbox*. Setelah tervalidasi, sistem masuk ke tahap *Deployment* ke infrastruktur *cloud* dan diakhiri dengan tahap *Review* melalui eksekusi *User Acceptance Test* sebelum pengumpulan data.

Infrastruktur pengujian dibangun menggunakan perangkat keras Raspberry Pi 4 Model B berspesifikasi RAM 4GB sebagai *edge device* yang bertugas melakukan inferensi deteksi hama (YOLO) dan mengirimkan *payload* gambar berformat Base64 setiap interval tiga detik. Di sisi penerima, *Virtual Machine* Google Cloud Platform dengan sistem operasi Ubuntu 22.04 LTS digunakan sebagai *server* terpusat. Arsitektur perangkat lunak pada *server* dibangun menggunakan paradigma *microservices* terisolasi di dalam *Docker Bridge Network*, yang diilustrasikan pada Gambar 2.

Gambar 2. Diagram Arsitektur Microservices dan Jaringan

Arsitektur pada Gambar 2 memisahkan setiap fungsionalitas ke dalam layanan yang independen. Nginx bertindak sebagai *web server* dan *reverse proxy*, Node.js sebagai *backend* komputasi latensi, Eclipse Mosquitto sebagai *message broker* untuk lalu lintas asinkron, dan PostgreSQL sebagai pangkalan data relasional.

Prosedur pengumpulan data dieksekusi melalui tiga skenario utama. Pertama, pengujian fungsionalitas (*Blackbox*) dan penerimaan pengguna akhir (*User Acceptance Test*) untuk memvalidasi kelayakan operasi di lapangan. Kedua, pengujian kinerja jaringan (*Quality of Service*) yang diekstraksi secara pasif menggunakan utilitas *tcpdump* dan Wireshark selama 15 menit untuk setiap protokol. Ketiga, pengujian keandalan operasional (*reliability*) yang memvalidasi stabilitas memori melalui operasi sistem tanpa henti selama 6 jam (*soak testing*), serta uji pemulihan otomatis (*self-healing*) dengan mematikan kontainer dependensi secara paksa.

Data mentah hasil perekaman jaringan kemudian diolah menggunakan teknik analisis statistik deskriptif komparatif. Ekstraksi dan kalkulasi parameter jaringan disandarkan pada standar indeks *Telecommunications and Internet Protocol Harmonization Over Network* (TIPHON) [8]. Parameter pertama yang diukur adalah *Throughput* atau laju kecepatan transfer data, yang dikalkulasikan menggunakan persamaan (1)

$$Throughput = \frac{Jumlah\ Data\ yang\ Dikirim}{Waktu\ pengiriman\ data} \quad (1)$$

Parameter kedua adalah *Packet Loss* yang mengukur persentase paket data yang hilang atau gagal mencapai tujuan akibat kemacetan lalu lintas

tingkat *transport*, dihitung menggunakan persamaan (2).

$$Packet\ Loss = \frac{(Paket\ Dikirim - Paket\ Diterima)}{Paket\ Dikirim} \times 100\% \quad (2)$$

Parameter ketiga adalah *End-to-End Latency* yang mewakili total waktu tempuh propagasi dan komputasi aplikasi dari perangkat pengirim hingga selesai diproses oleh *server*. Rumus perhitungannya ditunjukkan pada persamaan (3).

$$Latency = Waktu\ Terima - Waktu\ Kirim \quad (3)$$

Parameter keempat adalah *Jitter* yang menghitung simpangan atau fluktuasi waktu kedatangan antar-paket data. Persamaan yang digunakan untuk mengukur nilai *Jitter* ditunjukkan pada persamaan (4).

$$Jitter = \frac{Total\ Variasi\ Delay}{Total\ Paket\ Diterima} \quad (4)$$

III. HASIL DAN PEMBAHASAN

A. Fungsionalitas dan Penerimaan Sistem

Pengujian fungsionalitas dilakukan melalui dua metode, yakni pengujian kotak hitam (*blackbox*) untuk memvalidasi rute data internal, dan *User Acceptance Test* yang dieksekusi langsung oleh pengelola kebun hidroponik. Hasil pengujian kotak hitam mencatatkan tingkat keberhasilan 100% tanpa adanya cacat logika pada orkestrasi kontainer maupun sistem otentikasi.

Secara praktis, evaluasi penerimaan pengguna menunjukkan keselarasan dengan hasil teknis tersebut. Pengelola kebun memberikan skor maksimal untuk indikator stabilitas sistem dan kinerja jaringan. Tingginya skor ini membuktikan bahwa arsitektur antarmuka berhasil merender aliran gambar Base64 secara asinkron tanpa membebani memori peramban pengguna. Namun, fitur sistem pendukung keputusan mendapatkan skor netral. Hal ini terkonfirmasi sebagai dampak dari ketiadaan data agronomi aktual dari pakar, yang secara langsung membatasi nilai fungsi praktis modul tersebut di lapangan. Secara keseluruhan, ketiadaan laporan galat dari pengguna menegaskan bahwa perangkat lunak telah matang secara operasional

B. Kinerja Jaringan dan Degradasi Protokol

Evaluasi kinerja jaringan bertujuan untuk mengungkap ketidakefisienan arsitektur lapisan aplikasi dari protokol HTTP, WebSocket, dan MQTT. Mengingat penggunaan Wi-Fi 5G telah mengamankan kapasitas saluran fisik, perbedaan performa yang terekam secara murni bersumber dari struktur bawaan masing-masing protokol. Rekapitulasi hasil pengujian kualitas layanan disajikan pada Tabel I.

TABEL I. REKAPITULASI KINERJA QUALITY OF SERVICE

Parameter	HTTP	WebSocket	MQTT
Throughput (Kbps)	150.88	152.95	153.92
Throughput (Kbps)	0.26	0.57	0.30
Latency (ms)	27.71	25.11	26.10
Jitter (ms)	16.00	78.22	39.70

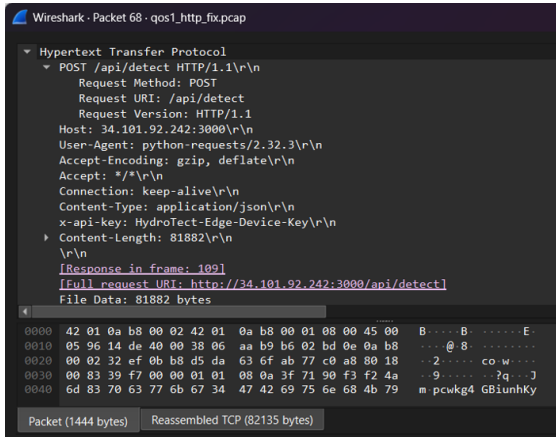
Berdasarkan Tabel I, parameter *Packet Loss* seluruh protokol berada di bawah angka 1%, yang mengklasifikasikannya ke dalam indeks sangat bagus menurut standar TIPHON. Hal ini mengonfirmasi bahwa saluran nirkabel terbebas dari kemacetan fisik.

Pada analisis kecepatan transmisi, HTTP menjadi protokol paling lambat dalam memanfaatkan saluran jaringan (*Throughput* terendah dan *Latency* tertinggi). Meskipun infrastruktur *server* telah menerapkan mekanisme penahanan soket persisten, keterlambatan ini murni dipicu oleh inefisiensi pembengkakan lapisan aplikasi. Mesin komputasi harus secara konstan membongkar struktur teks *header* berukuran besar yang berulang pada setiap siklus pengiriman. Hambatan ini diperparah oleh mekanisme komunikasi sinkron HTTP yang menahan antrian pengiriman selanjutnya hingga kode status penerimaan diterbitkan oleh *server*.

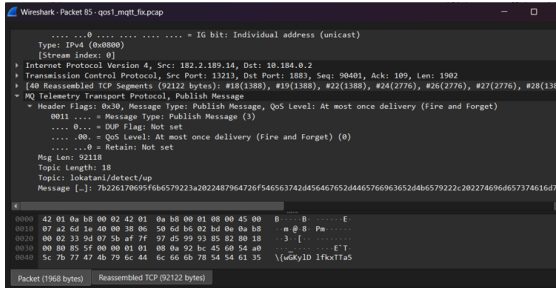
Di sisi lain, anomali performa terjadi pada protokol WebSocket. Meskipun mencatatkan *Latency* paling cepat (25.11 ms), WebSocket memicu lonjakan fluktuasi kedatangan paket (*Jitter*) terburuk hingga mencapai 78.22 ms. Tingginya nilai *Jitter* ini merupakan bukti empiris terjadinya hambatan perangkat keras pada *edge device*. Sesuai regulasi standar komunikasinya, WebSocket memaksa prosesor mikrokontroler untuk mengeksekusi operasi kriptografi XOR

Masking terhadap puluhan kilobita gambar sebelum masuk ke saluran jaringan. Beban penyandian ini memicu penundaan dorongan paket sesaat, yang bermanifestasi sebagai fluktuasi ketidakstabilan di *server* penerima.

Sebagai perbandingan, MQTT muncul sebagai standar arsitektur yang paling seimbang dan optimal. Dengan format data biner yang sangat ringan, MQTT berhasil memompa *Throughput* tertinggi (153.92 Kbps) dan memberikan *Latency* yang bersaing tanpa membebani prosesor perangkat dengan penalti komputasi kriptografi seperti WebSocket. Perbedaan efisiensi struktur muatan (teks berulang pada HTTP vs *framing* biner ringan pada MQTT) dapat diamati pada Gambar 1.



Gambar 3. Overhead Lapisan Aplikasi pada Protokol HTTP



Gambar 4. Overhead Lapisan Aplikasi pada Protokol HTTP

C. Keandalan Operasional Infrastruktur Microservices

Pengujian keandalan operasional membuktikan resiliensi arsitektur dalam menghadapi tekanan komputasi dan interupsi layanan. Pada uji beban selama 6 jam transmisi beruntun menggunakan MQTT, perangkat keras pengirim berhasil mencapai kesetimbangan suhu pada rentang 50.6°C hingga 55.0°C tanpa menyentuh batas kritis penurunan kinerja (*thermal*

throttling). Pada sisi *server* awan, persentase pemakaian RAM Node.js tertahan secara stabil di angka 2-3%. Kurva stabil ini membuktikan bahwa algoritma pembersihan memori (*garbage collection*) sukses menghancurkan penumpukan objek gambar Base64 secara berkala, sehingga sistem terbebas dari ancaman kebocoran memori

Ketangguhan isolasi *microservices* tervalidasi saat layanan dependensi dimatikan secara paksa. Data kecepatan pemulihan otonom saat interupsi terjadi disajikan pada Tabel II

TABEL II. KINERJA PEMULIHAN SISTEM (RECOVERY TIME)

Skenario Kegagalan	Status Utas Klien	Waktu Pemulihan Murni
Layanan <i>Broker</i> (MQTT)	Bertahan / <i>Reconnect</i>	5.019 Detik
Layanan Pangkalan Data	Mengabaikan Galat	8.044 Detik

Merujuk pada Tabel II, arsitektur berhasil mengisolasi kegagalan sehingga *server backend* utama terhindar dari kelumpuhan berantai. Perbedaan waktu pemulihan secara fundamental dipengaruhi oleh perilaku protokol. MQTT mencatatkan waktu pemulihan murni yang sangat efisien (5.019 detik) karena klien secara aktif melakukan upaya penyambungan ulang secara berkala sesaat setelah *broker* terputus. Sebaliknya, pemulihan layanan pangkalan data membutuhkan waktu sedikit lebih lama (8.044 detik). Hal ini disebabkan oleh sifat pemulihan pasif di mana *backend* baru melakukan pengaitan ulang soket saat ada permintaan masuk dari klien, ditambah beban waktu inisialisasi awal kontainer pangkalan data sebelum siap menerima tumpukan koneksi baru.

IV. KESIMPULAN DAN SARAN

Penelitian ini berhasil merancang bangun dan memvalidasi platform pemantauan deteksi hama berbasis *microservices* dengan tingkat keberhasilan uji fungsional mutlak dan tingkat penerimaan pengguna akhir yang tinggi. Arsitektur yang diimplementasikan secara terisolasi di dalam *Docker Bridge Network* terbukti mampu memfasilitasi transmisi dinamis matriks gambar Base64 secara stabil dari perangkat *edge* ke *server*. Infrastruktur ini menunjukkan keandalan

operasional yang ketat, ditandai dengan tercapainya kesetimbangan suhu perangkat keras (50.6°C–55.0°C) dan stabilitas pemakaian RAM (2-3%) selama 6 jam operasi kontinu tanpa kebocoran memori. Arsitektur sistem juga terbukti kebal terhadap kegagalan berantai (*cascading failure*), memiliki kapabilitas penyambungan ulang otomatis dalam 2-3 detik saat jaringan terputus, serta mencatatkan waktu pemulihan layanan murni yang sangat efisien, yakni 5.019 detik untuk gangguan pada layanan *broker* dan 8.044 detik untuk layanan pangkalan data

Evaluasi kinerja jaringan secara empiris mengonfirmasi MQTT sebagai standar protokol yang paling optimal untuk eksekusi transmisi telemetri visual. Meskipun seluruh protokol mencatatkan tingkat *Packet Loss* ideal di bawah 1%, MQTT mendominasi dengan rata-rata *Throughput* tertinggi (153.92 Kbps) dan *Latency* yang kompetitif (26.10 ms). Degradasi kualitas jaringan terbukti memiliki korelasi kausalitas dengan batasan arsitektural dari setiap protokol. Keterlambatan waktu tempuh pada HTTP (27.71 ms) dipicu oleh inefisiensi pembengkakan lapisan aplikasi (*application-layer bloat*) dan kewajiban mekanisme pengiriman sinkron. Di sisi lain, lonjakan *Jitter* pada WebSocket (78.22 ms) merupakan bentuk hambatan fisik (*bottleneck*) akibat kewajiban komputasi kriptografi *XOR Masking* di tingkat mikrokontroler. MQTT unggul karena berhasil mengeliminasi kedua kelemahan tersebut melalui penggunaan *framing* biner ringan dan mekanisme transmisi asinkron

Sebagai implikasi dan rekomendasi untuk pengembangan penelitian selanjutnya, integrasi data agronomi aktual yang divalidasi langsung oleh pakar pertanian mutlak diperlukan untuk meningkatkan validitas dan nilai guna praktis dari Sistem Pendukung Keputusan (DSS) di lapangan. Mengingat evaluasi jaringan pada riset ini dieksekusi menggunakan infrastruktur Wi-Fi 5G berkapasitas tinggi, pengujian masa depan perlu melakukan uji beban (*stress test*) menggunakan simulasi pembatasan kecepatan jaringan seluler, serta mengukur batas maksimal konkurensi koneksi *broker* MQTT dengan mengekspansinya ke ratusan simpul perangkat *edge* secara serentak. Terakhir, untuk mereduksi beban komputasi *payload* pada perangkat keras berspesifikasi terbatas, format penyandian teks Base64 disarankan untuk diganti dengan mekanisme serialisasi biner murni seperti *Protocol Buffers* atau *MessagePack*, diiringi dengan perluasan analisis komparatif menggunakan

protokol komunikasi berkinerja tinggi seperti gRPC atau *Constrained Application Protocol* (CoAP) .

V. REFERENSI

- [1] V. S. Kartika, A. F. Aji, S. Kusumastuti, R. A. Rochmanto, S. N. Hidayat, and S. A. Putra, "Penerapan Smart Greenhouse Hidroponik Berbasis IoT Menggunakan EBT di Nurus Sunnah Farm," *ABSYARA: Jurnal Pengabdian Pada Masyarakat*, vol. 6, no. 1, pp. 52-60, 2025.
- [2] Zulgani, D. Hastuti, Junaidi, Parmadi, Rafiqi, and Hardiani, "Penggunaan Sistem Hidroponik sebagai Alternatif Optimalisasi Budidaya Sayuran Organik: Studi Kasus Desa Tanjung Hutan," *Studium: Jurnal Pengabdian kepada Masyarakat*, vol. 3, no. 2, pp. 97-106, 2023.
- [3] H. Mital, K. Sharma, M. Sharma, P. Kumari, and R. Soni, "IoT-Based Smart Agriculture System," *International Journal of Engineering Trends and Applications (IJETA)*, vol. 12, no. 4, pp. 382-397, 2025.
- [4] D. C. Dinata, "Rancang Bangun Sistem Penyiraman Pestisida dan Deteksi Hama Menggunakan Metode YOLO," Laporan Penelitian, Politeknik Negeri Jakarta, Depok, 2023.
- [5] C. W. Kojansow, P. D. K. Manembu, and A. M. Rumagit, "Internet Of Things (IoT) Platform Development using WebSocket communication," *Jurnal Teknik Informatika*, vol. 19, no. 3, pp. 259-269, 2024.
- [6] M. F. Tsaqief and J. Sutopo, "Comparative performance analysis between the MQTT and WebSocket protocols," *Bit-Tech*, vol. 8, no. 2, pp. 2227-2237, 2025.
- [7] B. Amirkhanov, G. Amirkhanova, M. Kunelbayev, S. Adilzhanova, and M. Tokhtassyn, "Evaluating HTTP, MQTT over TCP and MQTT over WEBSOCKET for digital twin applications: A comparative analysis on Latency, stability, and integration," *International Journal of Innovative Research and Scientific Studies*, vol. 8, no. 1, pp. 679-694, 2025.
- [8] ETSI, "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); General aspects of Quality of Service (QoS)," *ETSI TR 101 329 V2.1.1*, 1999.
- [9] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed. McGraw-Hill Education, 2020.