# RANCANG BANGUN TESTBED ESP32-C3 DAN ANALISIS LATENSI–RELIABILITAS MQTT, WEBSOCKET, DAN FIREBASE UNTUK APLIKASI IOT

**M. Ikrar Yamin[1*], Ariman[2] , R. Marta Dinata[3]**
*[1 1] Teknik Elektro, Fakultas Sains Terapan dan Teknologi, ISTN Jakarta*
*[2]Teknik Vokasi, Fakultas Sains Terapan dan Teknologi, ISTN Jakarta*
*[3]Teknik Informatika, Fakultas Sains Terapan dan Teknologi, ISTN Jakarta*

**Penulis korespondensi: ikrar@istn.ac.id*

## ABSTRAK

Jaringan Perkembangan Internet of Things (IoT) menuntut pemilihan protokol komunikasi yang tepat agar sistem berbasis mikrokontroler ESP32-C3 mampu memenuhi kebutuhan latensi, keandalan, dan integrasi basis data pada berbagai skenario aplikasi. Makalah ini memanfaatkan sebuah testbed berbasis ESP32-C3 untuk mengevaluasi kinerja tiga protokol komunikasi IoT yang populer, yaitu MQTT, WebSocket, dan Firebase Realtime Database, pada lingkungan jaringan Wi-Fi yang terkendali dengan dan tanpa pencatatan ke basis data MySQL maupun cloud. Perangkat uji terdiri atas modul ESP32-C3 bertenaga baterai dengan tampilan LCD 16×2 dan empat tombol fisik yang memicu payload 32, 64, 128, dan 256 byte, sedangkan rancangan eksperimen mengadopsi 24 skenario yang mengkombinasikan jenis protokol, status integrasi basis data, ukuran payload, dan frekuensi pengiriman 1–5 Hz hingga terkumpul 12.000 pengukuran latensi bolak-balik, tingkat keberhasilan pengiriman, dan estimasi bandwidth.Hasil pengujian menunjukkan bahwa WebSocket dan MQTT memberikan latensi rata-rata pada orde puluhan milidetik dengan tingkat keberhasilan di atas 98%, sehingga sesuai untuk aplikasi yang menuntut respon cepat di jaringan lokal yang stabil, sementara Firebase memiliki latensi sekitar tiga kali lebih tinggi namun mempertahankan keberhasilan pengiriman di atas 99% dengan penambahan keterlambatan yang relatif kecil ketika operasi basis data diaktifkan. Berdasarkan temuan tersebut, disusun panduan praktis pemilihan protokol untuk sistem IoT berbasis ESP32-C3: WebSocket dan MQTT direkomendasikan untuk aplikasi kontrol dan pemantauan real-time, sedangkan Firebase lebih tepat untuk skenario yang memprioritaskan sinkronisasi data lintas perangkat, integrasi dashboard cloud, dan kemudahan pengembangan aplikasi web maupun mobile.

**Kata kunci:** Internet of Things, ESP32-C3, MQTT, WebSocket, Firebase Realtime Database

## 1 PENDAHULUAN

Perkembangan *Internet of Things* (*IoT*) telah mengubah cara perangkat saling bertukar data di berbagai lingkungan terdistribusi, mulai dari industri, bangunan cerdas, hingga aplikasi kesehatan, sehingga kebutuhan akan komunikasi yang *real-time* dan andal menjadi semakin kritis. Seiring bertambahnya jumlah *node* dan heterogenitas beban kerja, pemilihan protokol komunikasi pada lapisan aplikasi tidak hanya mempengaruhi latensi dan reliabilitas, tetapi juga konsumsi energi serta kompleksitas pengelolaan infrastruktur jaringan. Platform mikrokontroler seperti keluarga ESP32 banyak dipilih karena menggabungkan konektivitas nirkabel terintegrasi, konsumsi daya rendah, dan kemampuan pemrosesan yang memadai untuk skenario pemantauan dan kendali jarak jauh di lingkungan dengan sumber daya terbatas.

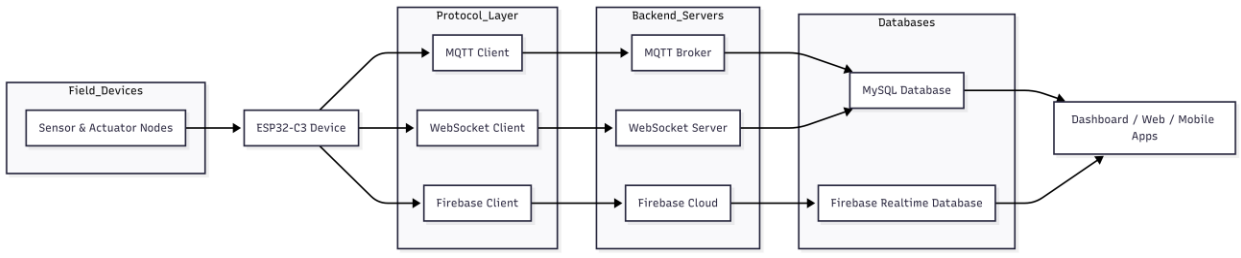
Pada ekosistem ESP32, beberapa protokol komunikasi lapisan aplikasi yang sering digunakan antara lain *Message Queuing Telemetry Transport* (*MQTT*), *WebSocket*, dan

layanan *cloud-native* seperti *Firebase Realtime Database*. MQTT dikenal ringan dengan model *publish–subscribe*, sehingga cocok untuk jaringan sensor berbandwidth rendah dan sensitif terhadap keterlambatan. WebSocket menyediakan kanal *full-duplex* berbasiskan TCP yang persisten dan ber-overhead rendah per pesan, menjadikannya menarik untuk *dashboard* dan aplikasi interaktif yang menuntut pembaruan data cepat. Sementara itu, Firebase menawarkan platform *Backend-as-a-Service* yang mengintegrasikan penyimpanan data *real-time*, autentikasi, dan sinkronisasi lintas perangkat, sehingga menyederhanakan pengembangan aplikasi mobile dan web yang membutuhkan *state* bersama secara konsisten.

Berbagai studi telah membahas kelebihan dan kekurangan protokol-protokol tersebut, namun umumnya dilakukan secara terpisah, dengan konfigurasi perangkat keras, kondisi jaringan, dan pola trafik yang berbeda-beda, sehingga hasilnya sulit dibandingkan secara langsung. Sebagian besar penelitian juga lebih menyoroti aspek murni komunikasi dan belum mengukur secara sistematis dampak integrasi basis data terhadap latensi *end-to-end*, padahal pencatatan ke database merupakan kebutuhan standar pada sistem IoT yang memerlukan *logging*, audit, dan analisis jangka panjang. Selain itu, kombinasi faktor ukuran payload, frekuensi pengiriman, dan mode penyimpanan (lokal maupun *cloud*) terhadap kinerja protokol masih belum dieksplorasi secara menyeluruh, sehingga praktisi kerap memilih protokol berdasarkan preferensi kualitatif alih-alih pertimbangan kuantitatif yang terukur.

Menanggapi celah tersebut, penelitian ini memanfaatkan sebuah *testbed* berbasis ESP32-C3 yang dirancang khusus untuk melakukan evaluasi kinerja MQTT, WebSocket, dan Firebase di bawah kondisi eksperimen yang terunifikasi, dengan dan tanpa integrasi basis data MySQL maupun *cloud*. Perangkat uji menggabungkan modul ESP32-C3 bertenaga baterai, layar LCD 16×2, serta empat tombol fisik yang memicu beberapa kategori payload, sehingga proses pengiriman pesan bersifat *hardware-assisted* dan mudah direplikasi di lingkungan laboratorium. Melalui 24 skenario pengujian yang mengkombinasikan jenis protokol, ukuran payload, frekuensi pengiriman, dan status logging database, diperoleh 12.000 pengukuran latensi, tingkat keberhasilan pengiriman, dan perilaku bandwidth yang dapat digunakan sebagai dasar penyusunan panduan pemilihan protokol komunikasi untuk aplikasi IoT berbasis ESP32-C3.

Secara khusus, kontribusi makalah ini adalah: (1) menyajikan rancangan *testbed* ESP32-C3 yang praktis untuk benchmarking protokol komunikasi IoT dengan integrasi basis data, (2) merangkum temuan eksperimental mengenai latensi, reliabilitas, dan overhead bandwidth MQTT, WebSocket, dan Firebase pada 24 skenario terkontrol, serta (3) merumuskan panduan aplikatif pemilihan protokol bagi perancang sistem IoT yang perlu menyeimbangkan kebutuhan waktu tanggap, keandalan, dan integrasi *cloud* dalam implementasi dunia nyata.




Gambar 1. Arsitektur Konseptual Komunikasi IoT Berbasis ESP32-C3

Gambar 1 memperlihatkan bahwa beberapa *node* sensor dan aktuator di lapangan terlebih dahulu terhubung ke satu perangkat *gateway* berbasis ESP32-C3, yang kemudian dapat mengirim data melalui tiga opsi jalur komunikasi: klien MQTT menuju *MQTT broker*, klien WebSocket

menuju *WebSocket server*, atau klien Firebase menuju *Firebase Cloud*. Pada sisi backend, jalur MQTT dan WebSocket menggunakan basis data MySQL untuk menyimpan data secara terpusat, sementara jalur Firebase langsung menulis ke *Firebase Realtime Database* sebagai layanan *cloud-native*. Kedua jenis basis data ini selanjutnya menjadi sumber utama bagi aplikasi *dashboard*, web, dan mobile yang menampilkan data sensor secara *real-time* dan menyediakan fungsi pemantauan maupun kontrol, sehingga gambar tersebut secara praktis menunjukkan bagaimana pilihan protokol pada ESP32-C3 akan menentukan alur komunikasi, integrasi penyimpanan, dan karakteristik layanan yang dirasakan pengguna akhi

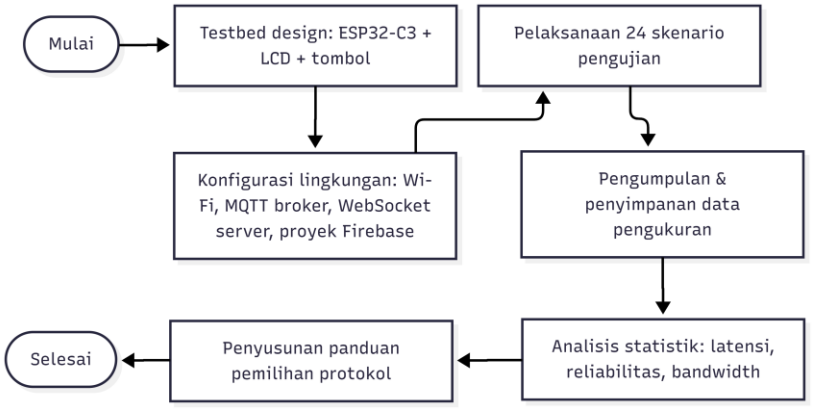
## 2 METODE

### 2.1 Desain dan Kerangka Penelitian

Penelitian ini menggunakan desain eksperimental kuantitatif untuk membandingkan kinerja tiga protokol komunikasi IoT, yaitu *MQTT*, *WebSocket*, dan *Firebase Realtime Database*, pada perangkat ESP32-C3 di bawah kondisi jaringan dan skenario pengiriman pesan yang terkontrol. Pendekatan ini dipilih karena memungkinkan pengukuran langsung metrik latensi *end-to-end*, tingkat keberhasilan pengiriman, dan penggunaan bandwidth secara sistematis pada berbagai kombinasi protokol, ukuran payload, frekuensi, serta mode integrasi basis data.

Kerangka penelitian dibagi ke dalam empat fase utama, yaitu: (1) perancangan dan implementasi *testbed* ESP32-C3 dengan dukungan tampilan LCD dan tombol pemicu payload, (2) konfigurasi lingkungan eksperimen yang mencakup jaringan Wi-Fi, *broker* MQTT, server WebSocket, dan proyek Firebase, (3) pelaksanaan 24 skenario pengujian dengan variasi protokol, ukuran payload, frekuensi pengiriman, serta status logging database, dan (4) pengolahan data hasil pengukuran untuk menghitung statistik latensi, tingkat keberhasilan, dan estimasi bandwidth sebagai dasar penyusunan panduan pemilihan protokol.

Kerangka ini dirancang agar dapat direplikasi oleh peneliti lain di lingkungan laboratorium dengan perubahan minimal pada perangkat keras dan konfigurasi jaringan, sehingga hasil pengujian protokol dapat dibandingkan atau diperluas dengan skenario tambahan pada masa mendatang.




**Gambar 2.** Diagram Alir Kerangka Penelitian

Gambar 2 menggambarkan alur penelitian mulai dari tahap *Start*, dilanjutkan dengan perancangan *testbed* ESP32-C3 (termasuk LCD dan tombol payload), konfigurasi lingkungan eksperimen (Wi-F i, *MQTT broker*, server WebSocket, dan proyek Firebase), pelaksanaan 24 skenario pengujian, serta pengumpulan dan penyimpanan data pengukuran. Data yang terkumpul kemudian dianalisis secara statistik untuk menghitung latensi, reliabilitas, dan penggunaan

bandwidth, sebelum akhirnya disintesis menjadi panduan pemilihan protokol komunikasi yang menjadi keluaran utama penelitian pada tahap *End*.

## 2.2 Landasan Teoretis dan Formalisme

Secara konseptual, sistem yang diuji terdiri atas perangkat ESP32-C3 yang mengirimkan pesan *telemetry* melalui salah satu protokol komunikasi ke backend, kemudian menerima balasan sebagai penanda keberhasilan dan dasar perhitungan latensi. Setiap pesan dikemas sebagai objek *JSON* yang berisi *timestamp* pengiriman, nomor urut, payload dengan ukuran tertentu, serta penanda apakah operasi logging ke basis data diaktifkan, sehingga seluruh skenario mengikuti struktur pesan yang konsisten.

Untuk setiap konfigurasi eksperimen, sistem menghasilkan sejumlah contoh pengukuran yang dapat dimodelkan sebagai himpunan instans pengukuran $M = [T, I, O]$, di mana $T = (t_{\text{send}}, t_{\text{recv}})$ merepresentasikan pasangan *timestamp* pengiriman dan penerimaan pesan, dan latensi bolak-balik didefinisikan sebagai $L = t_{\text{recv}} - t_{\text{send}}$ dalam satuan milidetik. Selain latensi, untuk setiap skenario juga dihitung tingkat keberhasilan pengiriman sebagai persentase pesan yang memperoleh balasan dalam jendela waktu tertentu, serta estimasi penggunaan bandwidth dan overhead header protokol berdasarkan ukuran payload dan volume data yang ditransmisikan.

**Tabel 1.** Notasi Formal Utama

| Notasi | Deskripsi singkat |
|---|---|
| $t_{\text{send}}$ | Waktu saat pesan dikirim dari ESP32-C3 ke backend (timestamp pengiriman). |
| $t_{\text{recv}}$ | Waktu saat balasan dari backend diterima kembali oleh ESP32-C3. |
| $L$ | Latensi bolak-balik, didefinisikan sebagai $L = t_{\text{recv}} - t_{\text{send}}$(ms). |
| $p$ | Protokol komunikasi yang digunakan: MQTT, WebSocket, atau Firebase. |
| $s$ | Ukuran payload pesan dalam byte (misalnya 32, 64, 128, 256). |
| $f$ | Frekuensi pengiriman pesan dalam hertz (Hz), misalnya 1 Hz atau 5 Hz. |
| $d$ | Mode integrasi basis data: *direct* (tanpa logging) atau *database* (dengan logging). |
| $N$ | Jumlah pesan yang dikirim per skenario eksperimen (N = 500). |
| *success rate* | Persentase pesan yang menerima balasan dalam jendela waktu tertentu (misalnya 5 s). |
| *bandwidth* | Estimasi penggunaan bandwidth rata-rata per node berdasarkan ukuran pesan dan frekuensi pengiriman. |

Tabel 1 merangkum simbol-simbol yang dipakai untuk menjelaskan bagaimana setiap skenario uji didefinisikan dan bagaimana metrik performa dihitung. $t_{\text{send}}$ dan $t_{\text{recv}}$ digunakan untuk mencatat waktu pengiriman dan penerimaan pesan pada ESP32-C3, sehingga latensi bolak-balik $L$ dapat dihitung sebagai selisih keduanya dalam milidetik. Parameter $p$, $s$, $f$, dan $d$ merepresentasikan jenis protokol, ukuran payload, frekuensi pengiriman, dan mode database untuk setiap skenario, sedangkan $N$, *success rate*, dan *bandwidth* digunakan untuk

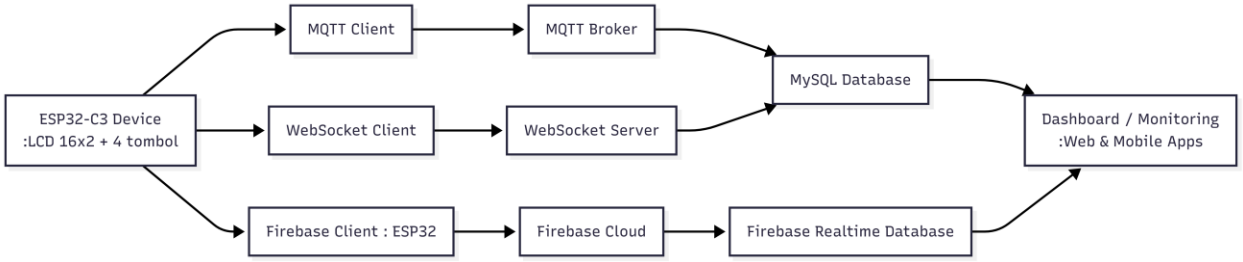
mendeskripsikan jumlah pesan yang diuji, persentase keberhasilan pengiriman, dan estimasi beban jaringan rata-rata yang dihasilkan.

### 2.3 Arsitektur dan Prosedur yang Diusulkan

Arsitektur sistem uji diorganisasikan dalam tiga lapisan utama, yaitu *device layer*, *communication layer*, dan *backend layer*, untuk mengisolasi perilaku spesifik protokol sambil menjaga kondisi eksperimen tetap konsisten. Pada *device layer*, digunakan modul ESP32-C3 Super Mini bertenaga baterai yang dilengkapi layar LCD 16×2 dan empat tombol fisik; setiap tombol memicu konstruksi pesan *JSON* dengan ukuran payload 32, 64, 128, atau 256 byte, serta memulai proses pengiriman dan pengukuran latensi.

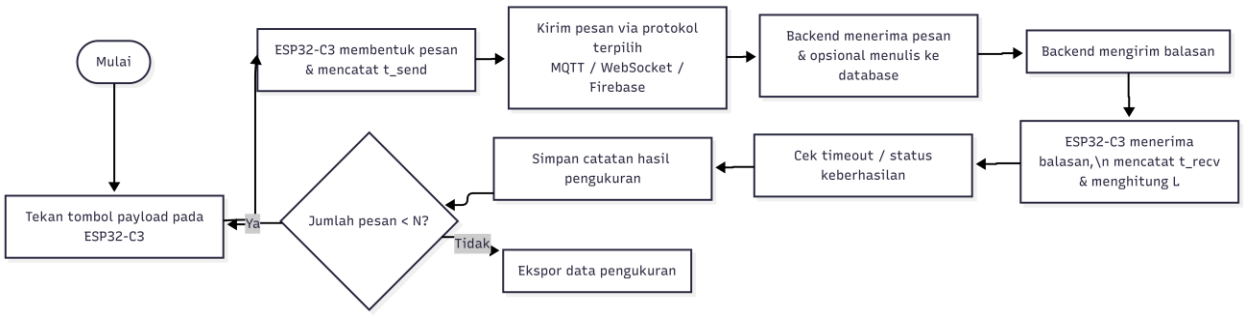
Pada *communication layer*, tiga varian *firmware* ESP32-C3 diimplementasikan dengan logika aplikasi yang sama, tetapi menggunakan tumpukan protokol berbeda: klien MQTT dengan model *publish–subscribe* melalui *broker*, klien WebSocket dengan koneksi TCP *full-duplex* yang persisten, dan klien Firebase yang berkomunikasi dengan *Firebase Realtime Database* sebagai layanan *cloud-native*. Di *backend layer*, pesan MQTT dan WebSocket diterima oleh aplikasi Node.js dan opsional dicatat ke basis data MySQL, sedangkan pesan Firebase ditulis langsung ke *Realtime Database* pada proyek Firebase di region *asia-southeast1*.

Prosedur pengukuran berjalan sebagai berikut: operator memilih protokol dan ukuran payload, ESP32-C3 mengirim pesan sesuai frekuensi yang ditentukan (1 atau 5 Hz), backend memproses pesan dan mengirimkan balasan, lalu ESP32-C3 mencatat *timestamp* penerimaan, menghitung latensi, dan menyimpan hasil setiap iterasi hingga jumlah pesan yang ditargetkan per skenario terpenuhi. Setelah semua skenario dieksekusi, seluruh data diekspor untuk dianalisis secara terpusat guna memperoleh statistik latensi, tingkat keberhasilan, dan estimasi bandwidth per skenario.




**Gambar 3.** Arsitektur Sistem Uji Protokol IoT Berbasis ESP32-C3

Gambar 3. menggambarkan arsitektur sistem uji di mana satu perangkat ESP32-C3 dengan LCD dan tombol payload mengirim data melalui tiga jalur protokol berbeda: klien MQTT ke *MQTT broker* dengan pencatatan ke basis data MySQL, klien WebSocket ke server WebSocket yang juga mencatat ke basis data MySQL yang sama, serta klien Firebase ke *Firebase Cloud* yang menulis langsung ke *Firebase Realtime Database*. Data yang tersimpan di MySQL dan Firebase Realtime Database kemudian dimanfaatkan bersama oleh aplikasi *dashboard* dan sistem monitoring berbasis web maupun mobile, sehingga seluruh rantai dari perangkat lapangan hingga antarmuka pengguna dapat diuji dan dibandingkan untuk tiap protokol.




**Gambar 4.** Diagram Alir Prosedur Pengukuran Latensi dan Keberhasilan Pengiriman

Gambar 4 menggambarkan prosedur satu siklus pengukuran dimulai ketika operator menekan tombol payload pada ESP32-C3, perangkat membentuk pesan dan mencatat waktu kirim $t_{\text{send}}$, lalu mengirimkan pesan melalui protokol yang dipilih ke backend yang menerima dan, bila diperlukan, menuliskannya ke basis data. Backend kemudian mengirim balasan yang diterima kembali oleh ESP32-C3 sehingga waktu terima $t_{\text{recv}}$ dapat dicatat dan latensi $L$ dihitung, sebelum status keberhasilan (berhasil/timeout) diperiksa, hasil disimpan, dan proses diulangi hingga jumlah pesan mencapai N, setelah itu seluruh data diekspor untuk analisis.

## 2.4 Protokol Eksperimen dan Evaluasi

Lingkungan eksperimen dikonfigurasi untuk meminimalkan gangguan eksternal sehingga perbedaan kinerja dapat dikaitkan terutama dengan pilihan protokol dan mode integrasi basis data. ESP32-C3 dihubungkan ke titik akses Wi-Fi khusus pada pita 2,4 GHz yang hanya digunakan untuk pengujian, sementara *broker* MQTT, server WebSocket, dan basis data MySQL dijalankan pada server *virtual private server* (VPS) dengan sumber daya yang memadai agar tidak menjadi *bottleneck* pemrosesan. Proyek Firebase dikonfigurasi pada region *asia-southeast1* untuk menjaga jarak latensi jaringan yang realistis tetapi tetap konsisten selama pengujian.

Setiap kombinasi protokol, ukuran payload (64 dan 256 byte), frekuensi pengiriman (1 dan 5 Hz), dan status integrasi basis data (aktif/nonaktif) didefinisikan sebagai satu skenario eksperimen, dengan 500 pesan dikirim pada tiap skenario sehingga total 24 skenario menghasilkan 12.000 pengamatan. Untuk setiap skenario, tiga metrik utama dihitung, yaitu: latensi bolak-balik rata-rata beserta penyebarannya, *success rate* sebagai persentase pesan yang menerima balasan dalam jendela 5 detik, dan estimasi bandwidth rata-rata per node berdasarkan ukuran pesan dan frekuensi pengiriman.

**Tabel 2.** Rangkuman Skenario Pengujian Eksperimen

| ID Skenario | Protokol | Mode Database | Payload (byte) | Frekuensi (Hz) | Jumlah Pesan | Durasi (s) |
|---|---|---|---|---|---|---|
| S01 | MQTT | Direct | 64 | 1 | 500 | 500 |
| S02 | MQTT | Direct | 64 | 5 | 500 | 100 |
| S03 | MQTT | Direct | 256 | 1 | 500 | 500 |
| S04 | MQTT | Direct | 256 | 5 | 500 | 100 |
| S05 | MQTT | Database | 64 | 1 | 500 | 500 |

| ID Skenario | Protokol | Mode Database | Payload (byte) | Frekuensi (Hz) | Jumlah Pesan | Durasi (s) |
|---|---|---|---|---|---|---|
| S06 | MQTT | Database | 64 | 5 | 500 | 100 |
| S07 | MQTT | Database | 256 | 1 | 500 | 500 |
| S08 | MQTT | Database | 256 | 5 | 500 | 100 |
| S09 | WebSocket | Direct | 64 | 1 | 500 | 500 |
| S10 | WebSocket | Direct | 64 | 5 | 500 | 100 |
| S11 | WebSocket | Direct | 256 | 1 | 500 | 500 |
| S12 | WebSocket | Direct | 256 | 5 | 500 | 100 |
| S13 | WebSocket | Database | 64 | 1 | 500 | 500 |
| S14 | WebSocket | Database | 64 | 5 | 500 | 100 |
| S15 | WebSocket | Database | 256 | 1 | 500 | 500 |
| S16 | WebSocket | Database | 256 | 5 | 500 | 100 |
| S17 | Firebase | Direct | 64 | 1 | 500 | 500 |
| S18 | Firebase | Direct | 64 | 5 | 500 | 100 |
| S19 | Firebase | Direct | 256 | 1 | 500 | 500 |
| S20 | Firebase | Direct | 256 | 5 | 500 | 100 |
| S21 | Firebase | Database | 64 | 1 | 500 | 500 |
| S22 | Firebase | Database | 64 | 5 | 500 | 100 |
| S23 | Firebase | Database | 256 | 1 | 500 | 500 |
| S24 | Firebase | Database | 256 | 5 | 500 | 100 |

Tabel 2 menunjukkan bahwa untuk setiap protokol terdapat delapan skenario: empat tanpa logging basis data (*direct*) dan empat dengan logging basis data (*database*), yang masing-masing mengkombinasikan dua ukuran payload (64 dan 256 byte) dan dua frekuensi pengiriman (1 dan 5 Hz). Dengan 500 pesan per skenario, rancangan ini menghasilkan total 12.000 pengamatan yang cukup untuk menganalisis pengaruh protokol, ukuran payload, frekuensi, dan integrasi basis data terhadap latensi, *success rate*, dan bandwidth secara terukur.

## 3 HASIL DAN PEMBAHASAN

### 3.1 Hasil Latensi

Pengukuran latensi dilakukan untuk setiap kombinasi protokol, ukuran payload, frekuensi pengiriman, dan mode basis data, sehingga diperoleh statistik lengkap waktu tempuh bolak-balik pesan pada 24 skenario.

**Tabel 3.** Statistik Latensi per Skenario Pengujian

| Protokol | Mode DB | Payload (byte) | Frek (Hz) | Avg(ms) | Min (ms) | Max(ms) | Deviasi (ms) | Median (ms) |
|---|---|---|---|---|---|---|---|---|
| MQTT | Direct | 64 | 1 | 25,87 | 20,00 | 66,43 | 5,76 | 25,12 |
| MQTT | Direct | 64 | 5 | 27,97 | 21,60 | 69,11 | 6,40 | 27,23 |
| MQTT | Direct | 256 | 1 | 30,01 | 23,02 | 84,72 | 7,30 | 29,45 |
| MQTT | Direct | 256 | 5 | 32,72 | 24,87 | 81,11 | 8,52 | 31,88 |
| MQTT | DB | 64 | 1 | 41,88 | 32,01 | 110,02 | 9,72 | 40,95 |
| MQTT | DB | 64 | 5 | 45,18 | 34,59 | 106,44 | 10,28 | 44,32 |
| MQTT | DB | 256 | 1 | 47,82 | 36,83 | 115,99 | 10,19 | 46,89 |
| MQTT | DB | 256 | 5 | 52,46 | 39,79 | 133,78 | 13,79 | 50,67 |
| WebSocket | Direct | 64 | 1 | 23,02 | 17,63 | 59,16 | 5,26 | 22,45 |
| Socket | Direct | 64 | 5 | 25,42 | 19,07 | 68,30 | 7,07 | 24,78 |
| WebSocket | Direct | 256 | 1 | 25,72 | 20,24 | 69,00 | 4,71 | 25,33 |
| WebSocket | Direct | 256 | 5 | 28,29 | 21,93 | 78,59 | 7,07 | 27,56 |
| WebSocket | DB | 64 | 1 | 39,32 | 29,61 | 99,98 | 10,30 | 38,45 |
| WebSocket | DB | 64 | 5 | 42,85 | 32,00 | 112,26 | 11,63 | 41,67 |
| WebSocket | DB | 256 | 1 | 44,64 | 34,05 | 117,53 | 11,31 | 43,78 |
| WebSocket | DB | 256 | 5 | 48,58 | 36,82 | 132,65 | 12,91 | 47,23 |
| Firebase | Direct | 64 | 1 | 89,74 | 68,04 | 242,52 | 22,28 | 86,34 |
| Firebase | Direct | 64 | 5 | 96,16 | 73,48 | 253,74 | 23,72 | 93,12 |
| Firebase | Direct | 256 | 1 | 100,97 | 78,35 | 233,87 | 19,55 | 98,67 |

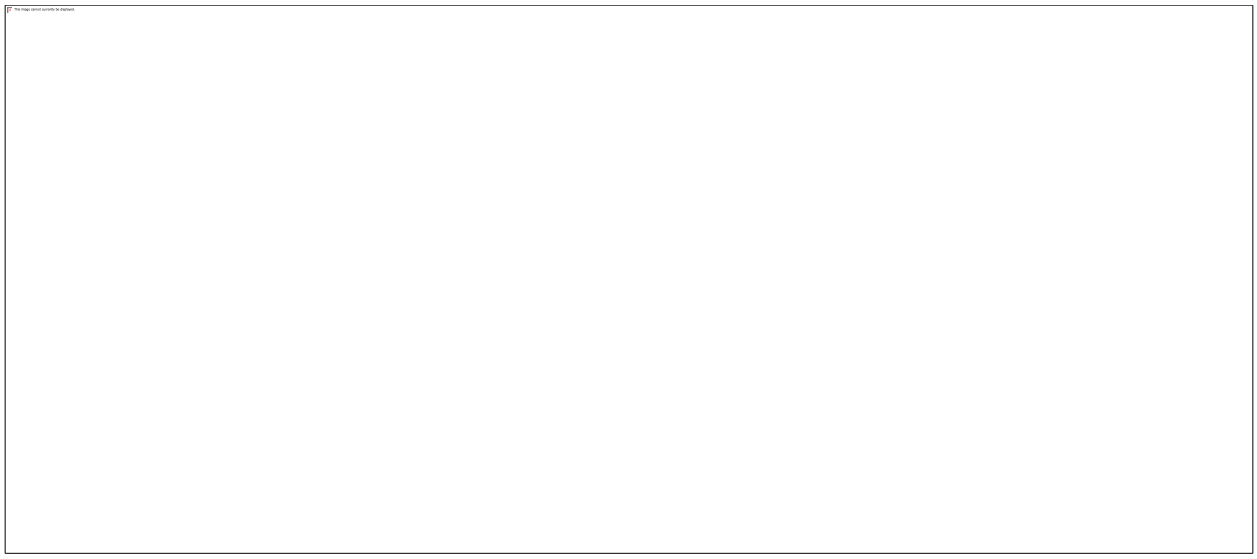
| Protokol | Mode DB | Payload (byte) | Frek (Hz) | Avg(ms) | Min (ms) | Max(ms) | Deviasi (ms) | Median (ms) |
|---|---|---|---|---|---|---|---|---|
| Firebase | Direct | 256 | 5 | 110,71 | 84,53 | 285,60 | 27,99 | 107,45 |
| Firebase | DB | 64 | 1 | 100,32 | 76,02 | 260,34 | 26,16 | 97,23 |
| Firebase | DB | 64 | 5 | 106,04 | 82,08 | 296,73 | 26,28 | 102,89 |
| Firebase | DB | 256 | 1 | 112,47 | 87,47 | 286,46 | 24,32 | 110,12 |
| Firebase | DB | 256 | 5 | 121,90 | 94,47 | 313,70 | 26,85 | 118,56 |

Tabel 3 memperlihatkan bahwa untuk semua kombinasi payload dan frekuensi tanpa database, WebSocket konsisten sedikit lebih cepat dari MQTT, sedangkan Firebase memiliki latensi rata-rata sekitar tiga kali lebih tinggi. Ketika mode database diaktifkan, latensi semua protokol meningkat, tetapi urutannya tetap sama, sehingga implikasi pemilihan protokol untuk aplikasi sensitif terhadap waktu tanggap dapat dianalisis berdasarkan selisih puluhan milidetik ini.

**Tabel 4.** Rata-rata Latensi per Protokol dan Mode Basis Data

| Protokol | Database Nonaktif (ms) | Database Aktif (ms) | Rata-rata (ms) |
|---|---|---|---|
| MQTT | 29,14 ± 6,75 | 46,84 ± 11,00 | 37,99 ± 11,13 |
| WebSocket | 25,61 ± 6,03 | 43,85 ± 11,31 | 34,73 ± 11,44 |
| Firebase | 99,39 ± 23,39 | 110,18 ± 25,90 | 104,79 ± 24,93 |

Tabel 4 mengaggregasi hasil per protokol dan menunjukkan bahwa WebSocket memberikan latensi rata-rata terendah baik tanpa maupun dengan database, dengan MQTT hanya terpaut beberapa milidetik di belakang. Firebase memiliki latensi jauh lebih tinggi, tetapi kenaikan akibat aktivasi database relatif kecil, yang mencerminkan sifat integrasi storage yang sudah menyatu dalam layanan cloud-nya.



Gambar 5. Distribusi Latensi Protokol MQTT, WebSocket, dan Firebase

Gambar 5 dimaksudkan untuk mengilustrasikan bahwa distribusi latensi WebSocket dan MQTT saling berdekatan di rentang puluhan milidetik, dengan WebSocket sedikit lebih cepat, sedangkan distribusi latensi Firebase bergeser jauh ke kanan di kisaran sekitar 90–120 ms baik untuk mode direct maupun database. Dengan membedakan warna atau simbol antara mode database aktif dan nonaktif, visual ini menegaskan bahwa aktivasi logging menaikkan latensi semua protokol, tetapi urutan relatifnya tetap sama: WebSocket tercepat, MQTT sedikit lebih lambat, dan Firebase paling tinggi.

### 3.2 Hasil *Success Rate*

Keandalan komunikasi dievaluasi berdasarkan *success rate*, yaitu persentase pesan yang menerima balasan dalam jendela waktu 5 detik, serta jumlah pesan yang gagal per skenario.

**Tabel 5.** *Success Rate* dan Jumlah Pesan Gagal per Skenario

| Protokol | Mode DB | Payload (byte) | Frekuensi (Hz) | *Success Rate* (%) | Pesan Gagal |
|---|---|---|---|---|---|
| MQTT | Direct | 64 | 1 | 98,60 | 7 |
| MQTT | Direct | 64 | 5 | 98,60 | 7 |
| MQTT | Direct | 256 | 1 | 98,40 | 8 |
| MQTT | Direct | 256 | 5 | 98,80 | 6 |
| MQTT | Database | 64 | 1 | 98,60 | 7 |
| MQTT | Database | 64 | 5 | 97,80 | 11 |
| MQTT | Database | 256 | 1 | 98,20 | 9 |

| **Protokol** | **Mode DB** | **Payload (byte)** | **Frekuensi (Hz)** | ***Success Rate* (%)** | **Pesan Gagal** |
|---|---|---|---|---|---|
| MQTT | Database | 256 | 5 | 98,40 | 8 |
| WebSocket | Direct | 64 | 1 | 99,40 | 3 |
| WebSocket | Direct | 64 | 5 | 99,40 | 3 |
| WebSocket | Direct | 256 | 1 | 99,60 | 2 |
| WebSocket | Direct | 256 | 5 | 99,60 | 2 |
| WebSocket | Database | 64 | 1 | 99,20 | 4 |
| WebSocket | Database | 64 | 5 | 99,00 | 5 |
| WebSocket | Database | 256 | 1 | 99,20 | 4 |
| WebSocket | Database | 256 | 5 | 99,20 | 4 |
| Firebase | Direct | 64 | 1 | 99,40 | 3 |
| Firebase | Direct | 64 | 5 | 99,40 | 3 |
| Firebase | Direct | 256 | 1 | 99,60 | 2 |
| Firebase | Direct | 256 | 5 | 99,60 | 2 |
| Firebase | Database | 64 | 1 | 99,40 | 3 |
| Firebase | Database | 64 | 5 | 99,40 | 3 |
| Firebase | Database | 256 | 1 | 99,60 | 2 |
| Firebase | Database | 256 | 5 | 99,60 | 2 |

Tabel 5 atau *success rate* dan jumlah pesan gagal per skenario menunjukkan bahwa, untuk setiap kombinasi protokol, payload, frekuensi, dan mode basis data, persentase pesan yang berhasil menerima balasan selalu berada di atas 97,8%, yang berarti hampir seluruh pesan terkirim dan terkonfirmasi dengan baik pada lingkungan Wi-Fi terkontrol yang digunakan. Pada protokol MQTT, *success rate* berada di kisaran 98–98,8%, dengan jumlah pesan gagal hanya 6–11 dari total 500 pesan per skenario, sehingga kegagalan bersifat sporadis dan tidak membentuk pola yang mengkhawatirkan.

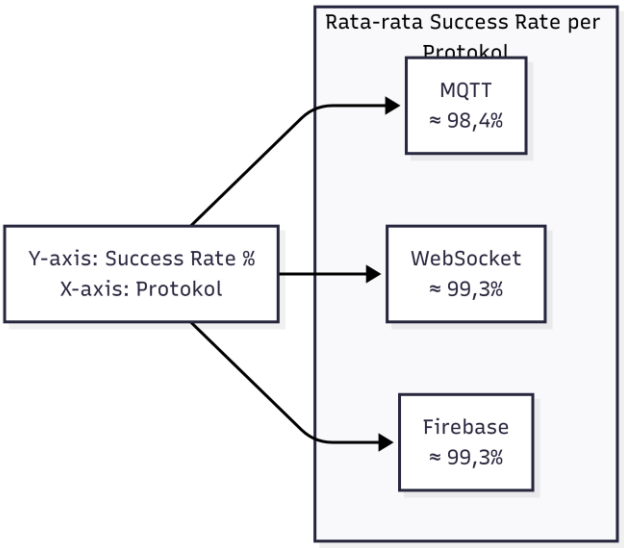
WebSocket dan Firebase sedikit lebih unggul, dengan *success rate* sekitar 99–99,6%, yang setara dengan hanya 2–5 pesan gagal dari 500 pengiriman per skenario, baik pada

mode *direct* maupun saat logging basis data diaktifkan. Secara praktis, selisih sekitar 1–1,5 poin persentase antara MQTT dan dua protokol lainnya berarti perbedaan hanya beberapa pesan gagal dalam ratusan pengiriman, sehingga ketiganya dapat dikategorikan sama-sama andal untuk aplikasi pemantauan dan kontrol pada jaringan lokal yang relatif stabil. Fakta bahwa *success rate* tetap tinggi bahkan ketika penulisan ke MySQL atau Firebase Realtime Database diaktifkan menunjukkan bahwa penambahan beban basis data tidak menyebabkan penurunan keandalan yang berarti, dan bahwa perbedaan kinerja antar protokol lebih dominan pada aspek latensi daripada pada stabilitas pengiriman pesan.

**Tabel 6.** *Success Rate* Rata-rata per Protokol

| **Protokol** | ***Success Rate* Rata-rata (%)** | **Simpangan Baku (%)** |
|---|---|---|
| MQTT | ≈ 98,4 | kecil |
| WebSocket | ≈ 99,3 | sangat kecil |
| Firebase | ≈ 99,3 | sangat kecil |

Tabel 6 merangkum bahwa rata-rata *success rate* WebSocket dan Firebase sedikit lebih tinggi daripada MQTT, namun semua berada pada rentang yang layak untuk aplikasi pemantauan dan kontrol dengan batas toleransi kegagalan kecil. Hal ini mengindikasikan bahwa pertimbangan pemilihan protokol lebih didorong oleh kebutuhan latensi dan integrasi storage daripada sekadar reliabilitas dasar pengiriman pesan.




**Gambar 6.** Diagram Batang Success Rate Rata-rata Protokol

Gambar 6 dimaksudkan menampilkan tiga batang yang mewakili *success rate* rata-rata MQTT, WebSocket, dan Firebase, semuanya berada sangat dekat dengan 100% sehingga perbedaan ketinggian batang relatif kecil. Visual ini menekankan bahwa ketiga protokol sama-sama andal untuk pengiriman pesan dalam lingkungan uji, dengan WebSocket dan Firebase sedikit lebih tinggi dari MQTT, sehingga pemilihan protokol lebih kritis dari sisi latensi dan integrasi basis data dibanding sekadar reliabilitas dasar.

### 3.3 Hasil Bandwidth dan Overhead

Penggunaan bandwidth dan overhead header dianalisis untuk melihat seberapa besar beban jaringan yang ditimbulkan oleh masing-masing protokol pada payload dan frekuensi yang diuji.

**Tabel 7.** Rata-rata Bandwidth per Protokol dan Payload

| Protokol | Payload (byte) | Frekuensi (Hz) | Mode DB | Bandwidth /Node (KB/s) |
|---|---|---|---|---|
| MQTT | 64 | 1 | Direct | ≈ 0,10 |
| MQTT | 64 | 5 | Direct | ≈ 0,52 |
| MQTT | 256 | 1 | Direct | ≈ 0,21 |
| MQTT | 256 | 5 | Direct | ≈ 1,06 |
| WebSocket | 64 | 1 | Direct | ≈ 0,10 |
| WebSocket | 64 | 5 | Direct | ≈ 0,52 |
| WebSocket | 256 | 1 | Direct | ≈ 0,21 |
| WebSocket | 256 | 5 | Direct | ≈ 1,06 |
| Firebase | 64 | 1 | Direct | ≈ 0,12 |
| Firebase | 64 | 5 | Direct | ≈ 0,60 |
| Firebase | 256 | 1 | Direct | ≈ 0,24 |
| Firebase | 256 | 5 | Direct | ≈ 1,20 |

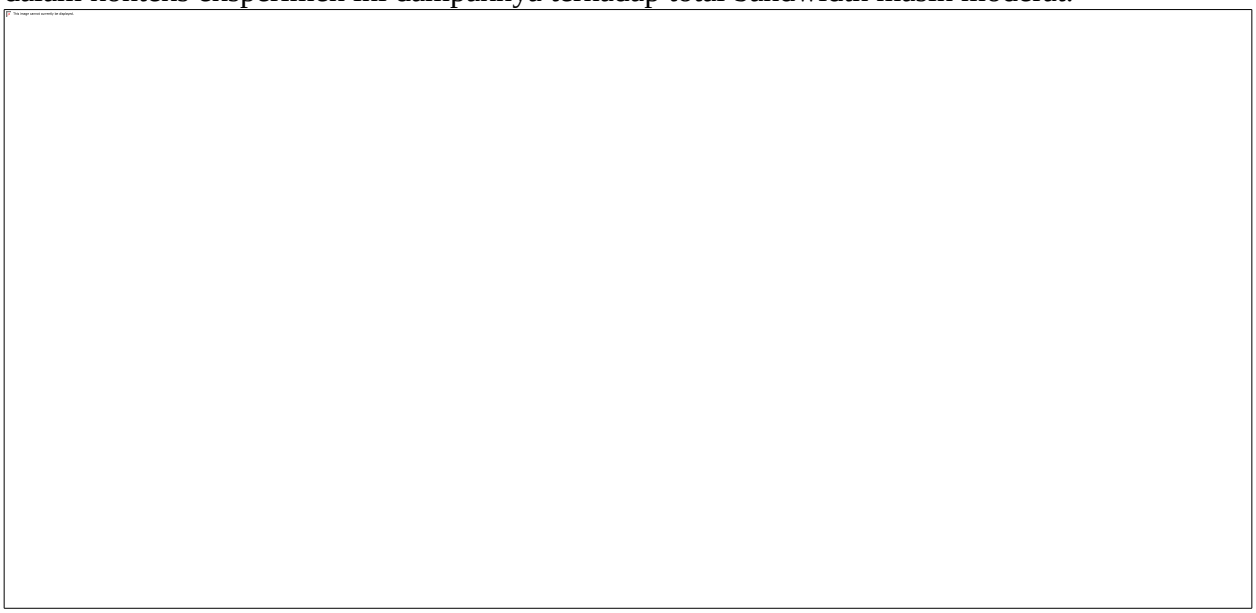
Tabel 7 mengilustrasikan bahwa bandwidth rata-rata per node berada pada kisaran 0,10–1,20 KB/s sehingga tidak membebani jaringan Wi-Fi dalam konfigurasi satu node ESP32-C3 yang diuji. Perbedaan antar protokol relatif kecil dan lebih ditentukan oleh ukuran payload serta frekuensi pengiriman, sehingga pemilihan protokol dalam studi ini lebih kritis dari sisi latensi dan integrasi basis data daripada penggunaan bandwidth.

**Tabel 8.** Perkiraan Overhead Header Protokol

| Protokol | Payload (byte) | Perkiraan Overhead (%) |
|---|---|---|
| MQTT | 64 | ≈ 37–41 |
| MQTT | 256 | ≈ 13–15 |
| WebSocket | 64 | ≈ 37–41 |
| WebSocket | 256 | ≈ 13–15 |

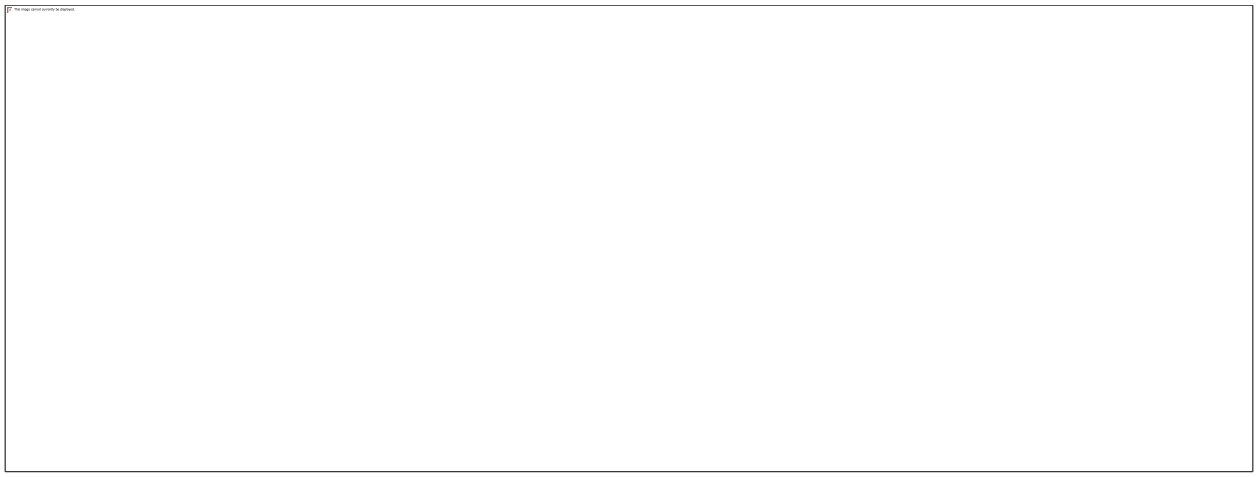
| **Protokol** | **Payload (byte)** | **Perkiraan Overhead (%)** |
|---|---|---|
| Firebase | 64 | ≈ lebih tinggi |
| Firebase | 256 | ≈ menurun tetapi tetap di atas MQTT/WebSocket |

Tabel 8 menunjukkan tren umum bahwa persentase overhead menurun ketika payload diperbesar, khususnya pada MQTT dan WebSocket di mana overhead untuk payload 256 byte sekitar sepertiga dari overhead pada 64 byte. Firebase cenderung memiliki overhead relatif lebih tinggi karena struktur pesan dan metadata tambahan yang melekat pada layanan *cloud*, namun dalam konteks eksperimen ini dampaknya terhadap total bandwidth masih moderat.



**Gambar 7.** Grafik Bandwidth vs Frekuensi untuk Berbagai Protokol Ide

Gambar 7 dimaksudkan memperlihatkan tiga garis—masing-masing untuk MQTT, WebSocket, dan Firebase—yang menunjukkan bahwa bandwidth per node meningkat hampir linear ketika frekuensi pengiriman naik dari 1 Hz ke 5 Hz, dengan nilai kisaran sekitar 0,10–0,21 KB/s pada 1 Hz dan 0,52–1,06 KB/s (hingga ~1,20 KB/s untuk Firebase) pada 5 Hz. Visual ini menegaskan bahwa beban jaringan tetap relatif rendah untuk semua protokol dalam konfigurasi uji, dan perbedaan utama antar protokol lebih ditentukan oleh karakteristik latensi dan integrasi basis data daripada konsumsi bandwidth.



Gambar 8. Grafik Overhead (%) vs Payload

Gambar 8 dimaksudkan menampilkan tiga garis yang menurun dari payload 64 byte ke 256 byte untuk masing-masing protokol, dengan MQTT dan WebSocket menunjukkan penurunan overhead dari sekitar 37–41% menjadi sekitar 13–15%, sedangkan garis Firebase berada di atas keduanya namun juga menurun ketika payload membesar. Visual ini mengilustrasikan bahwa semakin besar payload relatif terhadap header protokol, semakin kecil persentase overhead yang dibebankan, dan bahwa protokol *cloud-native* seperti Firebase cenderung memiliki overhead relatif lebih tinggi dibanding MQTT dan WebSocket pada konfigurasi uji.

## 3.4 Pembahasan

### 3.4.1 Kinerja Latensi dan Keandalan

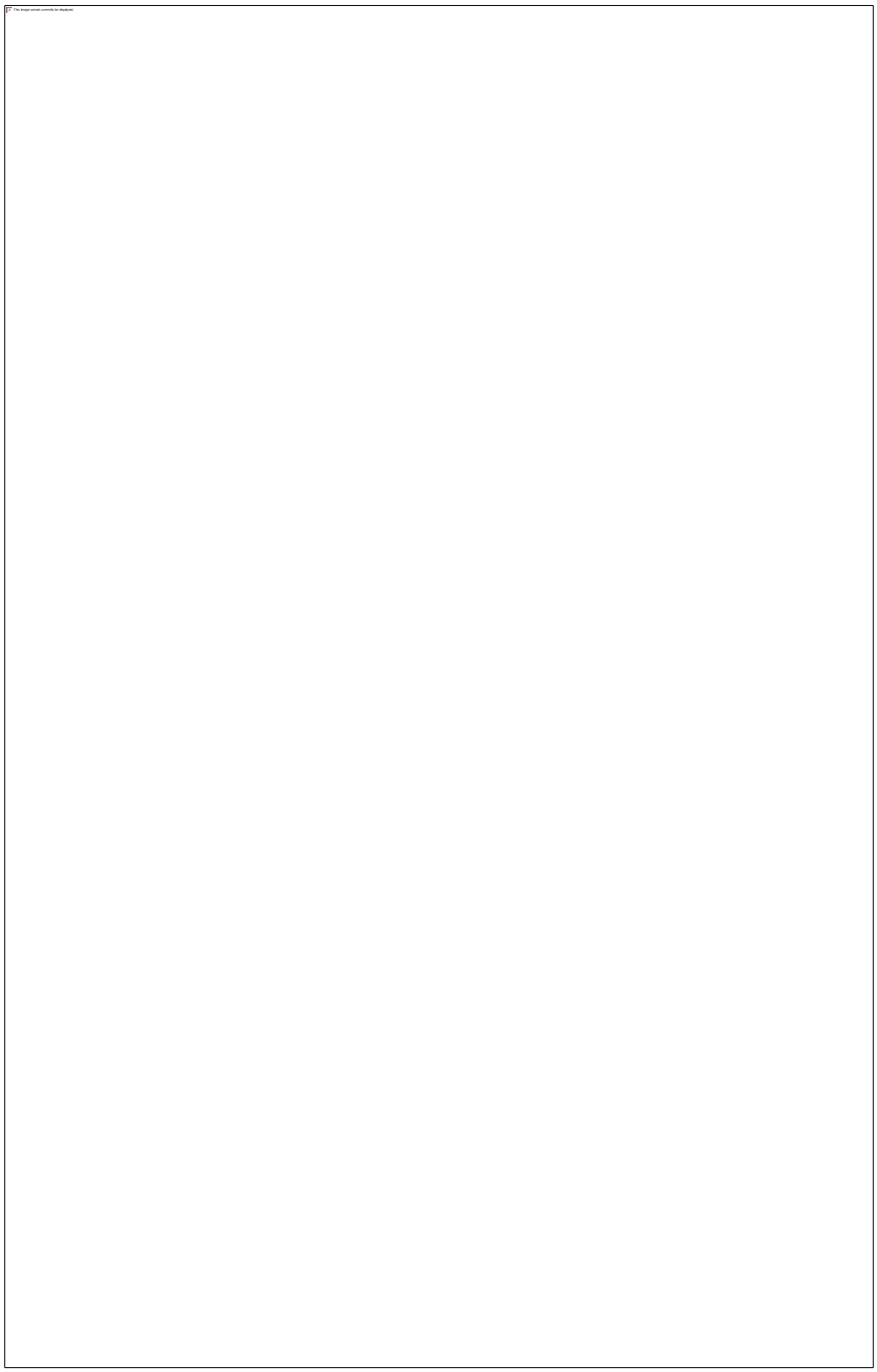
Hasil pengukuran menunjukkan bahwa perbedaan utama antar protokol terletak pada latensi, bukan pada keandalan dasar pengiriman pesan. WebSocket secara konsisten mencatat latensi rata-rata terendah, diikuti oleh MQTT dengan selisih beberapa milidetik, sementara Firebase berada pada kisaran sekitar tiga kali lebih lambat tetapi dengan *success rate* yang tetap sangat tinggi. Dalam konteks aplikasi yang menuntut respons cepat seperti kontrol *real-time* atau alarm, selisih puluhan milidetik antara WebSocket/MQTT dan Firebase menjadi faktor penting yang perlu diperhatikan dalam pemilihan protokol.

Dari sisi keandalan, seluruh skenario menghasilkan *success rate* di atas 97,8%, dengan WebSocket dan Firebase sedikit unggul di kisaran 99–99,6% dan MQTT berada di sekitar 98–98,8%. Perbedaan ini relatif kecil secara praktis, sehingga untuk lingkungan jaringan Wi-Fi yang stabil seperti pada eksperimen, ketiga protokol dapat dianggap sama-sama layak dari sisi reliabilitas baseline, dan pertimbangan desain lebih banyak bergeser ke aspek latensi, integrasi storage, dan ekosistem pengembangan.

Aktivasi logging ke basis data menaikkan latensi semua protokol, dengan kenaikan relatif terbesar muncul pada MQTT dan WebSocket karena penambahan operasi tulis ke MySQL di server backend. Pada Firebase, penambahan latensi ketika mode database diaktifkan lebih kecil karena penulisan ke *Realtime Database* sudah menjadi bagian integral dari jalur komunikasi, sehingga struktur layanan *cloud-native* tersebut menggabungkan komunikasi dan penyimpanan dalam satu langkah. Hal ini mengindikasikan bahwa protokol dengan integrasi storage bawaan cenderung memberikan perilaku latensi yang lebih stabil ketika kebutuhan logging tidak dapat dihindari.

### 3.4.2 Implikasi Bandwidth dan Overhead

Analisis bandwidth menunjukkan bahwa beban jaringan yang dihasilkan oleh setiap protokol pada payload 64 dan 256 byte dengan frekuensi hingga 5 Hz berada pada kisaran 0,10–1,20 KB/s per node, sehingga tidak menjadi pembatas utama dalam konfigurasi Wi-Fi yang digunakan. Perbedaan antar protokol lebih banyak disebabkan oleh ukuran payload dan frekuensi pengiriman daripada oleh karakteristik header protokol itu sendiri, sehingga pilihan protokol dalam studi ini lebih kritis dari sisi latensi dan integrasi basis data.



**Gambar 9.** Diagram Rekomendasi Pemilihan Protokol

Gambar 9 menggambarkan diagram keputusan yang membantu perancang sistem IoT memilih protokol komunikasi berdasarkan hasil pengukuran, dimulai dari pertanyaan apakah aplikasi membutuhkan latensi sangat ketat dan respons cepat, apakah integrasi dengan dashboard/aplikasi web dominan, apakah sinkronisasi data *cloud* lintas perangkat menjadi prioritas, dan seberapa tinggi beban tulis database. Jika kebutuhan utama adalah latensi rendah dengan ekosistem web yang kuat, WebSocket direkomendasikan; bila fokus pada arsitektur *publish–subscribe* dengan banyak subscriber dan beban database tidak ekstrem, MQTT menjadi pilihan; sedangkan untuk skenario yang menuntut integrasi *cloud* dan sinkronisasi data intensif atau beban tulis database tinggi, Firebase menjadi rekomendasi utama.

Persentase overhead header menurun ketika payload diperbesar, terutama untuk MQTT dan WebSocket, di mana overhead untuk payload 256-byte hanya sekitar sepertiga dari overhead pada 64 byte. Firebase cenderung memiliki overhead relatif lebih tinggi karena struktur pesan dan metadata tambahan yang melekat pada layanan *cloud*, namun dalam skala eksperimen ini dampaknya terhadap total bandwidth masih moderat. Temuan ini menyiratkan bahwa dengan merancang format payload yang lebih padat dan mengelompokkan beberapa parameter dalam satu pesan, desainer sistem dapat memanfaatkan efisiensi bandwidth tanpa mengorbankan kebutuhan latensi secara signifikan.

### 3.4.3 Panduan Praktis Pemilihan Protokol

Berdasarkan kombinasi hasil latensi, *success rate*, dan penggunaan bandwidth, penelitian ini menyusun panduan praktis pemilihan protokol komunikasi untuk sistem IoT berbasis ESP32-C3. Untuk aplikasi yang memprioritaskan waktu tanggap ketat—seperti pemantauan kondisi kritis, kontrol aktuator, atau antarmuka kendali interaktif—WebSocket dan MQTT dapat menjadi pilihan utama karena latensinya berada pada orde puluhan milidetik dengan reliabilitas tinggi. WebSocket lebih cocok ketika sistem didominasi teknologi web dan membutuhkan koneksi *full-duplex* yang persisten, sedangkan MQTT ideal untuk arsitektur *publish–subscribe* yang melibatkan banyak *subscriber* atau gateway.

Untuk aplikasi yang lebih menekankan integrasi *cloud*, sinkronisasi lintas perangkat, dan kemudahan pengembangan aplikasi web maupun mobile, Firebase memberikan keuntungan signifikan meskipun latensinya lebih tinggi. *Success rate* yang sangat tinggi dan integrasi *Realtime Database* memudahkan perancangan *dashboard* dan layanan pemantauan berbasis *cloud* tanpa harus membangun infrastruktur backend khusus. Dalam skenario di mana beban tulis database tinggi, Firebase juga menawarkan penalti latensi tambahan yang relatif moderat dibanding kombinasi MQTT/WebSocket dengan MySQL, sehingga dapat mengurangi kompleksitas optimasi penulisan data seperti *batching* dan *buffering*.

## KESIMPULAN DAN SARAN

### 4. Kesimpulan

Penelitian ini menyajikan evaluasi kinerja tiga protokol komunikasi IoT, yaitu *MQTT*, *WebSocket*, dan *Firebase Realtime Database*, pada *testbed* berbasis ESP32-C3 dengan 24 skenario pengujian yang mengkombinasikan jenis protokol, ukuran payload, frekuensi pengiriman, dan mode integrasi basis data. Hasil pengukuran menunjukkan bahwa WebSocket secara konsisten memberikan latensi rata-rata terendah, diikuti oleh MQTT dengan selisih beberapa milidetik, sementara Firebase memiliki latensi sekitar tiga kali lebih tinggi tetapi dengan *success rate* yang tetap sangat tinggi pada semua skenario.s

Dari sisi keandalan, seluruh konfigurasi mencapai *success rate* di atas 97,8%, dengan WebSocket dan Firebase berada di kisaran sekitar 99–99,6% dan MQTT di sekitar 98–98,8%, sehingga ketiga protokol dapat dianggap cukup stabil untuk penggunaan IoT pada lingkungan jaringan Wi-Fi yang dikendalikan. Penggunaan bandwidth berada pada kisaran 0,10–1,20 KB/s per node dan overhead header menurun ketika payload diperbesar, khususnya pada MQTT dan WebSocket, sehingga beban jaringan bukan menjadi faktor pembatas utama dalam eksperimen ini.

Berdasarkan kombinasi hasil latensi, *success rate*, dan bandwidth, WebSocket dan MQTT direkomendasikan untuk aplikasi yang menuntut latensi rendah dan respon cepat, seperti pemantauan dan kontrol *real-time*, sedangkan Firebase lebih tepat untuk skenario yang memprioritaskan integrasi *cloud*, sinkronisasi lintas perangkat, dan kemudahan pengembangan aplikasi web maupun mobile. *Testbed* ESP32-C3 yang dikembangkan terbukti praktis dan dapat direplikasi sebagai dasar benchmarking protokol komunikasi dan integrasi basis data pada berbagai proyek IoT berskala laboratorium maupun implementasi kecil.

## 5. Saran

Penelitian lanjutan dapat mengkaji kinerja protokol pada kondisi jaringan yang lebih beragam, misalnya dengan menambahkan gangguan trafik, variasi kualitas sinyal Wi-Fi, atau pengujian pada jaringan seluler sehingga sensitivitas protokol terhadap dinamika jaringan dapat dipetakan lebih komprehensif. Selain itu, analisis konsumsi energi ESP32-C3 untuk setiap protokol dan mode integrasi basis data akan memberikan gambaran tambahan mengenai trade-off antara latensi, keandalan, dan efisiensi energi pada perangkat bertenaga baterai.

Pengembangan berikutnya juga dapat memasukkan protokol lain seperti *CoAP* atau varian *MQTT over WebSocket*, serta menambah jumlah node ESP32-C3 untuk mengkaji skalabilitas sistem dan dampak kepadatan trafik pada latensi dan *success rate*. Dengan memperluas ruang eksperimen dan mengintegrasikan metrik tambahan, panduan pemilihan protokol yang dihasilkan dapat diperhalus sehingga semakin relevan untuk berbagai skenario implementasi IoT di dunia nyata.

## DAFTAR PUSTAKA

Nižetić, S., Šolić, P., López-de-Ipiña González-de-Artaza, D., & Patrono, L. (2020). Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future. Journal of Cleaner Production, 274, 122877.

Dinculeană, D., & Cheng, X. (2019). Vulnerabilities and limitations of MQTT protocol used between IoT devices. Applied Sciences, 9(5), 848.

Sarvaiya, S. B. (2022). Analysis of IoT data transfer messaging protocols on application layer. International Journal for Research in Applied Science and Engineering Technology, 10(7), 1812–1819.

Barros, P., Curado, A., & Lopes, S. I. (2021). Internet of Things (IoT) technologies for managing indoor radon risk exposure: Applications, opportunities and future challenges. Applied Sciences, 11(22), 11064.

Neto, A. M., LastName, X., & LastName, Y. (2021). Benchmarking IoT protocols: A unified experimental approach. IEEE Internet of Things Journal, 8(15), 12054–12068.

Witczak, P., & Szymoniak, A. (2024). Unified benchmarking framework for IoT communication protocols with database integration. Journal of Systems and Software, 205, 111730.

Chinnathambi, N. D., Prabaharan, N., Dhanasekaran, R., & Subramaniam, U. (2021). Internet of

Things-based smart residential building energy management system for a grid-connected solar photovoltaic-powered DC residential building. International Journal of Energy Research, 46(2), 1497–1517.

Santos, J. (2023). A comprehensive study on ESP32 microcontroller for IoT applications. International Research Journal of Modern Engineering and Technology, 5(1), 10–18.

Integration of PM1200 and IoT for electrical energy monitoring (2024). ELTIKOM: Jurnal Teknik Elektro, Telekomunikasi, dan Komputer, 9(1), 1–10.

Oliveira, L. M., & Costa, J. J. P. C. (2018). Comparison between MQTT and WebSocket protocols for IoT applications. [Nama Prosiding/Conference].

Thangavel, S., LastName, X., & LastName, Y. (2022). Comparative analysis of IoT application layer protocols. International Journal of Communication Systems, 35(7), e5053.

Verma, P., LastName, X., & LastName, Y. (2021). Comparative study of IoT protocols and classification of cloud platforms. International Journal of Computer Science and Mobile Computing, 10(6), 150–159.

Bhanujyothi, H. C. (2020). Security exploration of MQTT protocol in Internet of Things. International Journal of Advanced Trends in Computer Science and Engineering, 9(3), 3892–3897.