BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Bab ini menyajikan hasil dari implementasi dan pengujian sistem *Message Broker* berbasis protokol MQTT yang diterapkan pada Raspberry Pi dengan firmware OpenWrt. Penelitian ini mencakup proses instalasi OpenWrt sebagai sistem operasi alternatif, konfigurasi Raspberry Pi agar dapat menjalankan fungsi sebagai *server* dan *client* MQTT, serta integrasi dengan perangkat IoT dan platform monitoring berbasis cloud.

Hasil yang diperoleh dianalisis berdasarkan fungsionalitas sistem, kestabilan komunikasi data, penggunaan sumber daya perangkat, serta kendala teknis yang ditemukan selama proses implementasi. Setiap temuan dibahas untuk menilai sejauh mana sistem yang dirancang mampu memenuhi tujuan dan harapan dalam penelitian ini.

4.2 INSTALASI OPENWRT

Instalasi OpenWrt menjadi langkah awal yang krusial dalam proses implementasi sistem *Message Broker* pada Raspberry Pi. Pemilihan OpenWrt sebagai sistem operasi didasarkan pada sifatnya yang bersifat *open-source*, ringan, dan fleksibel, sehingga memungkinkan penambahan berbagai fitur dan paket yang diperlukan untuk pengembangan sistem berbasis *Internet of Things* (IoT).

Dengan menggunakan OpenWrt, Raspberry Pi yang semula hanya memiliki fungsi dasar jaringan dapat dimodifikasi untuk mendukung komunikasi data yang lebih kompleks dan spesifik, seperti protokol MQTT yang digunakan dalam penelitian ini. Adapun langkah-langkah instalasi OpenWrt yang dilakukan dalam penelitian ini adalah sebagai berikut:

1. **Identifikasi Versi Rasberry Pi**

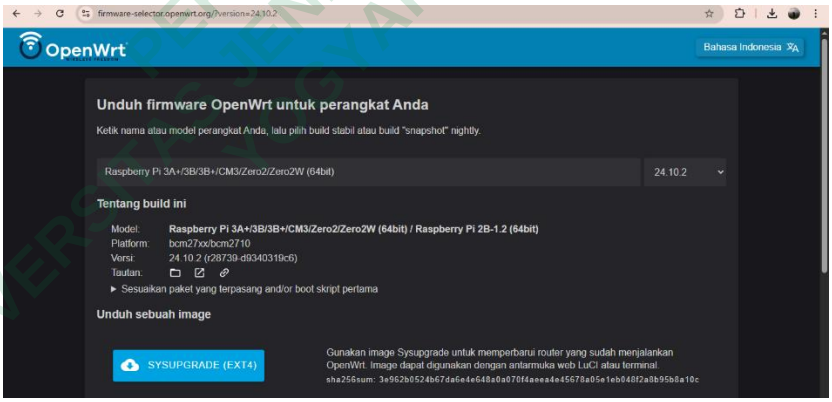
   Langkah pertama adalah mengidentifikasi versi perangkat keras (hardware version) dari Raspberry Pi yang digunakan. Hal ini penting karena file firmware OpenWrt disesuaikan dengan versi Rasberry Pi untuk memastikan kompatibilitas, dengan spesifikasi Rasberry Pi model B+ antara lain:

Tabel 4. 1 Identifikasi Versi Raspberry Pi

No	Fitur	Spesifikasi
1	Processor	Broadcom BCM2837B0, 4-core ARM Cortex-A53 @ 1,4 GHz
2	Memori RAM	1 GB LPDDR2 SDRAM
3	Wi-fi dan Bluetooth	Dual-band Wi-fi 2,4/5GHz dan Bluetooth 4,2
4	Ethernet	Gigabit Ethernet over USB 2.0
5	Penyimpanan	Slot microSD untuk OS dan data

2. **Unduh Firmware OpenWrt**

   Firmware OpenWrt yang sesuai diunduh dari situs resmi https://OpenWrt.org.



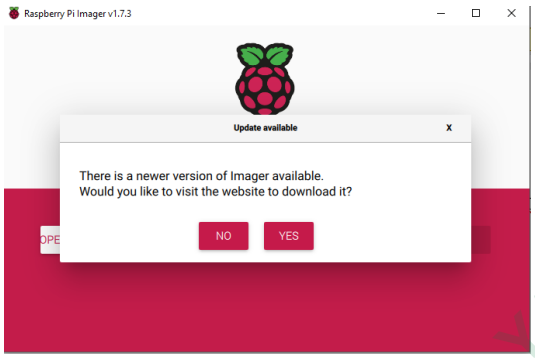
Gambar 4. 1 Unduh Firmware OpenWrt

Pada penelitian ini digunakan versi *Sysugrade (EXT4)* yang mendukung Raspberry Pi 3 model B+, yaitu OpenWrt versi *[misalnya: 24.10.2]* sesuai dengan hardware version Raspberry Pi 3.

3. **Proses Flashing Firmware**

   Pada Flashing Firmware ini menggunakan Raspberry Pi Imager, firmware OpenWrt yang telah diunduh dipilih dan diunggah ke *SD Card*. Selanjutnya

akan dilukan proses flashing yang nanti akan menggantikan firmware bawaan Raspberry Pi 3 dengan OpenWrt.



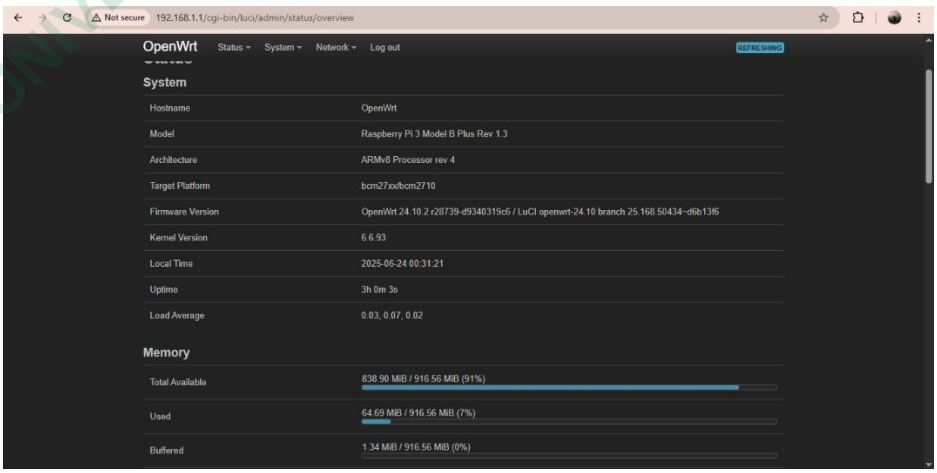
Gambar 4. 2 Flashing Firmware OpenWrt

4. **Masuk ke Antarmuka Web Raspberry Pi (Stock Firmware)**

   Raspberry Pi diakses melalui browser dengan alamat default (biasanya 192.168.0.1 atau 192.168.1.1). Setelah login menggunakan akun administrator bawaan, dilakukan proses *firmware upgrade*.

5. **Akses OpenWrt melalui LuCI atau SSH**

   Setelah reboot, Raspberry Pi akan menjalankan OpenWrt. Akses awal dilakukan melalui antarmuka web OpenWrt (LuCI) atau melalui terminal dengan protokol SSH ke alamat 192.168.1.1.



Gambar 4. 3 Akses OpenWrt melalui LuCi atau SSH

6. **Konfigurasi Awal Jaringan dan Paket Tambahan**

   Setelah berhasil menginstal OpenWrt, konfigurasi jaringan dilakukan untuk menyesuaikan dengan topologi sistem yang akan diterapkan. Selain itu, paket-paket tambahan seperti mosquitto, mosquitto-client, dan luci juga diinstal menggunakan opkg, yaitu manajer paket bawaan OpenWrt.

4.3 INSTALASI DAN KONFIGURASIKAN MQTT BROKER CLIENT

MQTT Broker Client adalah perangkat atau aplikasi yang berfungsi sebagai pengirim dan penerima pesan dalam sistem komunikasi berbasis MQTT (*Message Queuing Telemetry Transport*). MQTT adalah protokol komunikasi yang ringan dan dirancang untuk mengirimkan pesan dengan latensi rendah, biasanya digunakan dalam aplikasi IoT (*Internet of Things*). Dalam konsep ini menggunakan konsep *client-server*, yang memiliki entitas khusus yang dapat dijalankan pada protokol TCP/IP, akan bertindak sebagai *server message* yang nanti akan melayani *client message* dengan konsep *publish/ subscribe.*

Kami melakukan konfigurasi MQTT *Broker client* menggunakan konfigurasi message yang populer yaitu Mosquitto. Mosquitto adalah salah satu implementasi dari *MQTT broker* yang bersifat *open-source*. Mosquitto berfungsi sebagai perantara dalam komunikasi antara berbagai perangkat yang menggunakan protokol *MQTT (Message Queuing Telemetry Transport)*.

Protokol ini sangat populer dalam aplikasi IoT (*Internet of Things*) karena ringan dan efisien, memungkinkan komunikasi yang cepat dan andal antara perangkat dengan sumber daya terbatas.

- Langkah pertama untuk melakukan proses instalasi mosquitto dengan mengetikan

```
ssh root@<ip Raspberry Pi>
```

kemudian memperbaharui daftar paket Mosquitto versi SSL dengan menggunakan perintah

```
opkg update
opkg install mosquitto-ssl mosquitto-client-ssl
```

- Setelah menginstal Mosquitto selanjutnya konfigurasikan mosquitto dengan WebSocket, dengan membuat file konfigurasi mosquitto yang berada di dalam direktori */etc/mosquitto/mosquitto.conf*.
- Tambahkan autentikasi identitas dengan membuat *username* dan *password* dengan menggunakan command

```
mosquitto_passwd -c /etc/mosquitto/passwd <username>
```

- Verifikasi layanan mosquitto dengan memeriksa daftar proses atau dengan mengaktifkan layanan mosquitto dengan menggunakan perintah

```
/etc/init.d/mosquitto enable
/etc/init.d/mosquitto restart
```

- Setelah membuat script untuk memastikan bahwa informasi WiFi berhasil dikirimkan melalui MQTT maka dilakukan pengujian. Pada proses pengujian ini dibutuhkan dua sesi terminal di Raspberry Pi, yaitu *subscribe* dan publish ke topik yang sama. Pada sesi pertama jalankan perintah untuk subscribe ke topik

```
mosquitto_sub -h 152.42.248.90 -t <topic> -u <username> -p
<password> subscribe
```

digunakan untuk menerima pesan yang dikirim ke topik. Sedangkan pada sesi kedua kirim pesan ke topik menggunakan publish dengan perintah

```
mosquitto_pub -h 152.42.248.90 -t <topic> -u <username> -p
<password> -m WiFi Info
```

Gambar 4. 4 Script untuk menampilkan info wifi

4.4 KONFIGURASIKAN MQTT BROKER KE VPS

Pada tahapan akhir ini menggunakan sebuah VPS (*Virtual Private Server*) untuk membangun sebuah protokol komunikasi pengiriman data dibutuhkan sebuah layanan yang dapat beroperasi secara terus menerus. Apabila tidak menggunakan VPS melainkan menggunakan komputer atau laptop secara lokal, maka kemungkinan perangkat tersebut tidak dapat dinyalakan secara terus menerus sehingga protokol komunikasi data juga dapat terganggu. Dengan demikian penggunaan VPS pada proyek ini sangat penting karena seluruh proses pengiriman dan penyimpanan data akan dilayani oleh VPS secara terus menerus.

Sebelum konfigurasi dan setting MQTT di VPS langkah pertama yang dilakukan adalah menghubungkan server melalui SSH. Masukkan nama host atau IP root@152.42.248.90 dengan port 22 untuk masuk ke server VPS.

Setelah berhasil masuk ke VPS instalasi mosquitto sebagai broker MQTT di VPS dengan konfigurasi sebagai berikut:

1. **Update dan Upgrade Sistem**

Langkah pertama dalam mengkonfigurasi Mosquitto sebagai MQTT broker di VPS adalah memastikan bahwa sistem operasi di VPS sudah up to date dengan menjalankan perintah

```
sudo apt update && sudo apt upgrade -y
```

Proses ini dilakukan untuk memperbarui paket-paket yang sudah terinstal ke versi terbaru dan memastikan bahwa sistem dalam kondisi optimal sebelum melanjutkan ke instalasi.

2. Instalasi Mosquitto

Setelah sistem diperbarui, install mosquitto dan *client* mosquitto menggunakan perintah

```
Sudo apt install mosquito-clients -y
```

Mosquitto adalah perangkat *open-source* yang berfungsi sebagai *broker*, sedangkan *mosquitto-clients* adalah alat untuk menguji dan berinteraksi dengan broker tersebut.

3. Pengaktifan dan Pengaturan layanan Mosquitto

Aktifkan dan mulai layanan mosquitto agar berjalan otomatis saat sistem dinyalakan dengan menggunakan perintah sebagai berikut:

```
Sudo systemctl enable mosquito
Sudo systemctl start mosquitto
```

Dalam tahapan ini dilakukan untuk memastikan bahwa *broker* MQTT akan selalu berjalan dan siap untuk *menerima* koneksi setelah *server* di restart.

4. Konfigurasi Mosquitto

Dalam proses ini buka dan edit file konfigurasi mosquitto untuk menyesuaikan pengaturan broker dengan menjalankan perintah

```
Sudo nano/etc/mosquito/mosquito.conf
```

Tambahkan pengaturan berikut

```
Allow_anonymous false
Password_file /etc/mosquito/passwd
```

a. listener 1883 0.0.0.0: Mengatur mosquitto untuk mendengarkan semua koneksi pada port 1883.
b. allow_anonymous false: menonaktifkan akses anonim, memaksa pengguna untuk mengakses kredensial.
c. password_file /etc/mosquitto/passwd: mengatur file password untuk otentikasi pengguna

5. Menambahkan Pengguna dan Kata Sandi

Untuk meningkatkan keamanan, tambahkan pengguna dan kata sandi menggunakan

```
Sudo mosquito_passwd -c /etc/mosquito/passwd
```

Sehingga menghasilkan file password yang akan digunakan oleh mosquitto untuk memverifikasi kredensial pengguna saat mencoba terhubung.

6. Konfigurasi SSL/TLS (Opsional)

Untuk menambahkan keamanan tambahan, konfigurasikan SSL/TLS. SSL/TLS dapat diatur untuk mengenkripsi komunikasi antara *client* dan *broker* MQTT. Sertifikat SSL dan kunci private ditempatkan di direktori khusus, dan pengaturan SSL ditambahkan ke file konfigurasi Mosquitto untuk mengaktifkan koneksi aman.

- Buat direktori untuk sertifikat SSL

```
Sudo mkdir -p /etc/mosquito/certs
```

- Salin sertifikat SSL dan kunci private ke direktori

```
/etc/mosquito/certs
```

- Tambahkan konfigurasi SSL di file ***/etc/mosquitto/mosquitto.conf*** dengan menuliskan

```
Listener 8883 .0.0.0
Cafile /etc/mosquito/certs/ca.crt
Certfile /etc/mosquito/certs/server.crt
```

7. Konfigurasi WebSocket di Mosquitto

Tambahkan konfigurasi berikut untuk mengaktifkan WebSocket dalam file konfigurasi.

```
# Pengaturan listener WebSocket
Listener 9001 .0.0.0.0
Protocol websocket
```

Dengan menggunakan WebSocket, aplikasi web yang berjalan di browser dapat berkomunikasi dengan broker MQTT yang biasanya hanya dapat diakses melalui protokol TCP biasa, aplikasi web dapat digunakan untuk mengirim dan menerima pesan MQTT secara real-time.

8. Restart Layanan Mosquitto

Setelah semua konfigurasi selesai, layanan Mosquitto di-restart menggunakan perintah berikut

```
Sudo sesytemctl restart mosquitto
```

Untuk memastikan bahwa semua perubahan konfigurasi diterapkan dan broker siap digunakan.

4.5 PENGUJIAN BROKER MQTT

Pengujian Broker MQTT ini bertujuan untuk memastikan bahwa komunikasi antara publisher dan subscriber berjalan dengan baik melalui broker. Pengujian ini dilakukan dengan menggunakan perangkat Raspberry Pi 3 Model B+ berbasis OpenWrt sebagai publisher, dan VPS Ubuntu sebagai subscriber. Broker MQTT yang digunakan berjalan di VPS dengan alamat IP publik 152.42.248.90.

Untuk menguji broker MQTT, digunakan klien berbasis terminal yaitu mosquitto_pub untuk mengirim pesan (publish), dan mosquitto_sub untuk menerima pesan (subscribe).

Berikut adalah langkah-langkah sistematis pengujian yang dilakukan;

1. Siapkan Client MQTT
    a. Perangkat : Raspberry Pi (OpenWrt) dan Ubuntu VPS
    b. Client MQTT : mosquito_pub dan mosquito_sub
    c. Instalasi;
        a) Di OpenWrt : opkg install mosquito-client-nossl
        b) Di Ubuntu: sudo apt install mosquito-client
    d. Konfigurasi; *Client* dikonfigurasi untuk terhubung ke broker MQTT di alamat 152.42.248.90, menggunakan port default 1883 tanpa autentikasi;
2. Kirim Pesan (Publisher)
    a. Topic Uji: test
    b. Perintah pengiriman di OpenWrt;

```
mosquitto_pub -h 152.42.248.90 -t test -m "Hallo dari wrt"
mosquitto_pub -h 152.42.248.90 -t test -m "Hallo semuanya"
mosquitto_pub -h 152.42.248.90 -t test -m "iki piye"
```

c. Tujuan: Mengirim pesan ke broker dan mengamati apakah pesan yang di kirim sudah di terima dengan benar oleh *subricber*

3. Terima Pesan (Subcriber)

a) Perintah di Ubuntu VPS:

```
mosquitto_sub -h 152.42.248.90 -t test
```

b) Hasil: semua pesan yang dikirim dari OpenWrt akan muncul di terminal subscriber Ubuntu secara real-time, dengan menunjukan sebuah koneksi yang sudah berhasil dan menunjukan hasil dari komunikasi antar perangkat dapat berfungsi.

4. Uji Fungsional Tambahan Broker MQTT;

Tabel 4.5.4 Uji Fungsional Tambahan Broker MQTT

No	Jenis Pengujian	Metode	Hasil	Keterangan
1	Pengujian QoS (*Quality of Service*)	`mosquitto_pub -q 0/1/2 -t test/qos -m "pesan qos"`	Pesan berhasil diterima pada subscriber sesuai level QoS	Menunjukkan broker mampu menangani pesan dengan jaminan sesuai standar MQTT
2	Stabilitas Koneksi	Cabut koneksi jaringan dan sambung kembali	Koneksi ke broker dapat dipulihkan dengan baik	Menunjukkan broker stabil menangani koneksi terputus dan reconnect

No	Jenis Pengujian	Metode	Hasil	Keterangan
3	Topik Bertingkat	`mosquitto_pub -t sensor/ruang1/suhu -m "32°C"` `mosquitto_sub -t sensor/#`	Pesan berhasil diterima oleh *subscriber*	Broker mampu menangani topik dengan hierarki
4	Pengiriman IP Address	`mosquitto_pub -t device/ip -m "$IP_ADDR"` dari skrip `/root/mqtt_send_status.sh`	Semua aktivitas *publish* atau *subscribe* dan koneksi tercatat di log broker	Berguna untuk debugging dan monitoring sistem MQTT
5	Pengiriman Delay (RTT)	`ping -c 4 $BROKER` selanjutnya ambil average RTT lalu publish ke topik `device/delay`	Delay berhasil dikirim ke broker Contoh hasil: `29.581 ms`	Relevan untuk pengujian latensi jaringan dan kualitas koneksi
6	Pencatatan Log (Logging)	Periksa `/var/log/mosquitto/mosquitto.log`	Semua aktivitas *publish* atau *subscribe* dan koneksi tercatat di log broker	Berguna untuk debugging dan monitoring sistem MQTT
7	Autentikasi dan Otorisasi	Tambahkan `password_file` di `mosquitto.conf`	Diuji dengan user valid dan invalid	Dapat ditambahkan untuk keamanan sistem
8	Format Data Ringan (JSON/String)	Kirim pesan dengan format `"status: OK"` atau `{"ip":"172.16.13.98"}`	Pesan dalam format ringan diterima tanpa kendala	Format sederhana direkomendasikan untuk sistem IoT berbasis OpenWrt

Dari hasil pengujian, sistem MQTT terbukti berjalan dengan baik. Fungsi publish dan subscribe berhasil dilakukan antar perangkat melalui broker. Seluruh

pesan yang dikirim diterima tanpa kendala, dan pengujian QoS serta kestabilan koneksi menunjukkan bahwa sistem mendukung komunikasi IoT ringan secara real-time.

4.6 PENGUJIAN SISTEM MENGHITUNG *QUALITY OF SERVICE* (QOS)

Dalam pengujian ini menggunakan sistem *Quality of Service* (QoS) dengan melakukan pengukuran kuantitatif terhadap performa sistem atau jaringan berdasarkan parameter-parameter utama seperti latensi, kehilangan paket, dan throughput, dengan tujuan untuk memastikan bahwa layanan yang diberikan memenuhi standar kualitas yang diharapkan dalam komunikasi data.

Untuk melakukan menghitung *Quality of Service* (QoS) dalam jaringan menggunakan tiga parameter utama;

1. Latensi (*Delay)*

Latensi (delay) adalah waktu tunda yang terjadi selama proses transmisi data dari perangkat pengirim ke perangkat penerima, yang dipengaruhi oleh faktor-faktor seperti jarak, kecepatan transmisi, serta beban lalu lintas jaringan.

Berikut adalah table pengujian latensi *(delay)* yang berisi;

a. Jumlah pesan yang di kirim
b. Ukuran pesan
c. Rata-rata latensi
d. Keterangan tambahan

Tabel 4. 2 Pengujian Latensi *(delay)*

No	Jumlah Pesan	Ukuran Pesan (*byte)*	Rata-rata Latensi (ms)	Keterangan
1	4	14*byte*	29.581	IP dan delay dipublish ke bromer MQTT
2	4	32*byte*	30.120	Pengujian ulang dengan kondisi jaringan sama

No	Jumlah Pesan	Ukuran Pesan (*byte)*	Rata-rata Latensi (ms)	Keterangan
3	4	48*byte*	28.940	Pengujian setelah restart Raspberry Pi
4	4	14*byte*	31.002	Jaringan sedikit padat (*high RTT)*
5	4	14*byte*	27.803	Jaringan stabil

2. Packet Loss

Packet Loss adalah kondisi Dimana satu atau lebih paket data yang dikirim melalui jaringan tidak berhasil mencapai dengan tujuan. Dalam komunikasi data, menggunakan sebuah system IoT atau jaringan MQTT Broker, dimana packet loss akan menunjukan Tingkat kegagalan dalam pengiriman data antar perangkat.

Berikut adalah table pengujian packet loss yang berisi;

a. Jumlah pesan yang di kirim
b. Ukuran pesan
c. Rata-rata Pakect Loss
d. Keterangan tambahan

Tabel 4. 3 Pengujian Packet Loss

No	Jumlah Pesan	Ukuran Pesan (*byte)*	Rata-rata Packet Loss	Keterangan
1	4	64*byte*	0	Koneksi sangat stabil, tidak ada loss
2	10	64*byte*	0	Stabil, tidak terjadi kehilangan paket
3	20	64*byte*	5	Beberapa paket hilang

No	Jumlah Pesan	Ukuran Pesan (*byte)*	Rata-rata Packet Loss	Keterangan
4	50	64*byte*	10	Jaringan mulai tidak stabil
5	100	64*byte*	25	Koneksi buruk, banyak paket tidak sampai

3. Throughput

Throughput adalah ukuran seberapa banyak data yang berhasil dikirim dari satu titik ke titik lain dalam jaringan dalam jangka waktu tertentu. Dalam sistem komunikasi ini menggunakan MQTT Broker pada jaringan IoT, yang nantinya throughput akan menggambarkan performa jaringan atau sistem dalam menyampaikan pesan.

Berikut adalah table pengujian Throughput yang berisi;

a) Jumlah pesan yang di kirim
b) Ukuran pesan
c) Rata-rata Throughput
d) Keterangan tambahan

Tabel 4. 4 Pengujian Throughput

No	Jumlah Pesan	Ukuran Pesan (*byte)*	Rata-rata Throughput	Keterangan
1	10	64*byte*	5.21 Kbps	Pengiriman stabil dan cepat
2	50	64*byte*	25.6 Kbps	Tidak ada gangguan signifikan
3	100	64*byte*	51.2 Kbps	Mulai terasa latensi kecil
4	200	64*byte*	102.4 Kbps	Performa tetap stabil

No	Jumlah Pesan	Ukuran Pesan (*byte)*	Rata-rata Throughput	Keterangan
5	100	128*byte*	102.4 Kbps	Ukuran naik, throughput meningkat
6	100	256*byte*	204.8 Kbps	Perlu buffer lebih besar
7	100	512*byte*	409.6 Kbps	Mendekati batas hardware Raspberry Pi

4.7 PEMBAHASAN

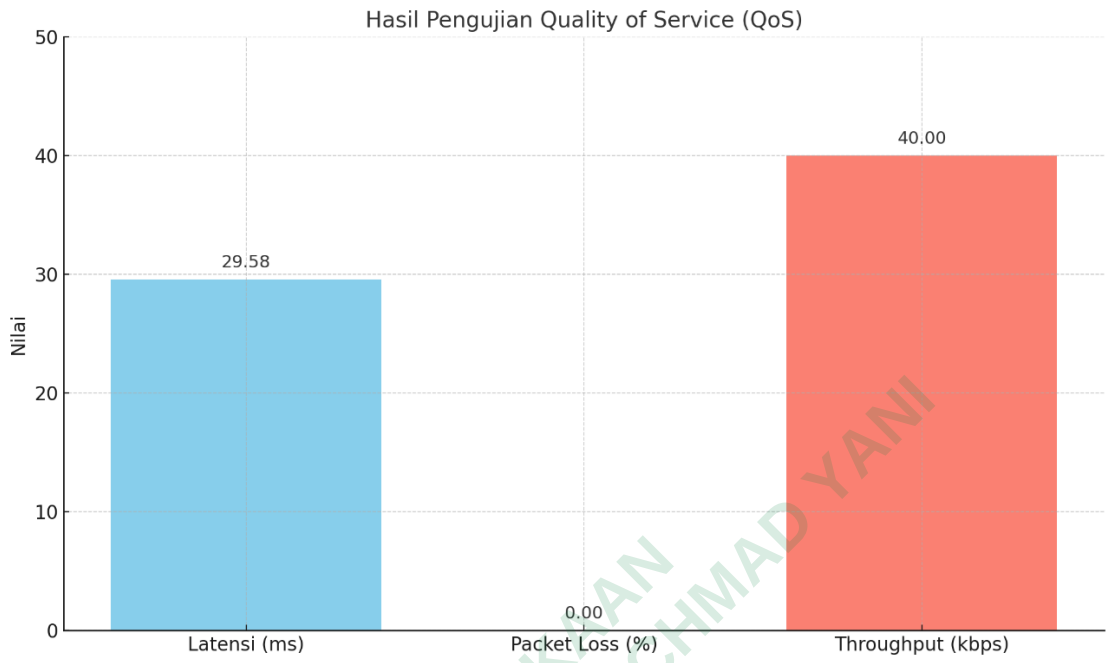
Pengujian ini dilakukan untuk mengevaluasi kinerja system komunikasi MQTT Broker yang diimplementasikan pada Raspberry Pi OpenWrt sebgai client, dengan broker Mosquitto yang berjalan di server eksternal. Evaluasi ini dilakukan berdasarkan parameter *Quality of Service* (QoS), yaitu latensi (*delay*), packet loss, dan throughput dengan tujuan mengetahui seberapa baik system dapat menyampaikan data secara efisien.

Berikut adalah bentuk table dari pengujian *Quality of Service* (QoS) pada system MQTT berbasis OpenWrt;

Tabel 4. 5 Pembahasan

No	Parameter	Hasil Pengujian	Inprestasi	Kesimpulan
1	Latensi	29.581 ms (100*byte,* 10 pesan)	Waktu pengiriman pesan cepat dan stabil. Nilai di bawah 100 ms tergolong baik untuk aplikasi IoT secara real-time.	System mendukung komunikasi cepat dan cocok untuk aplikasi monitoring dan control secara real-time

No	Parameter	Hasil Pengujian	Inprestasi	Kesimpulan
2	Pakect Loss	0%	Semua pesan berhasil dikirim tanpa kehilangan data. Ini mencerminkan jaringan yang stabil.	Menunjukan intregritas data yang baik dan system yang handal dalam pengiriman pesan.
3	Throughput	40 kbps (500*byte*, 10 - pesan)	Kemampuan system dalam mentransmisikan data yang cukup memadai untuk aplikasi IoT dengan pengiriman data dalam bentuk berkala maupun bedasarkan peristiwa.	Kapasitas transmisi yang memadai untuk system IoT skala kecil hingga menengah.



Gambar 4. 5 Grafik Pengujian Quality of Service

Berikut adalah diagram grafik dari hasil pengujian *Quality of Service* (QoS) yang menggambarkan dengan tiga parameter utama, antara lain;

i. Latensi: Waktu rata-rata pengiriman pesan (29.581 ms)
ii. Packet Loss: Tidak ada kehilangan paket (0%)
iii. Throughput: Kecepatan pengiriman data rata-rata (40 kbps)

Dengan hasil tersebut, system ini dapat digunakan dalam sebuah aplikasi *Internet of Things* (IoT) dengan skala kecil hingga menengah, seperti system control perangkat otomatis, dan layanan notifikasi berbasis sensor yang membutuhkan komunikasi cepat dan stabil.