JSAI: Journal Scientific and Applied Informatics
Vol. 8, No. 1, Januari 2025, hal. 9~22
E-ISSN: 2614-3054; P-ISSN: 2614-3062, accredited by Kemenristekdikti, Sinta 4
DOI: 10.36085

Rancang bangun Server Message Queuing Telemetry Transport (MQTT) Internet of Things (IoT) Berbasis Eclipse

[1]Irwansyah, [2]Wahyuni Eka Sari, [3]Agus Triyono, [4]Yhunike Widiya Pratama, [5]Muhammad Bagus [6]Bintang Tmur,[7]Susanna
[1,2,3,4,5,6,7]Politeknik Negeri Samarinda, Indonesia
[1]irwansyah@polnes.co.id, [2]wahyuniekasari@gmail.com, [3]agustriyono@gmail.com, [4]yhunikewp@gmail.com, [5]mbagus@gmail.com, [5]bintangt@gmail.com, [6]sus4n77@gmail.com

Article Info

Article history:

Received, 2024-11-26
Revised, 2024-12-24
Accepted, 2024-12-26

Kata Kunci:

Internet of Things (IoT),
Message Queuing Telemetry Transport (MQTT),
Eclips,
QoS,
Eclipse Hono,
Server

ABSTRAK

Perancangan server *Message Queuing Telemetry Transpor*t (MQTT) untuk *Internet of Things* (IoT) memanfaatkan *Eclipse Mosquitto* sebagai perangkat lunak server inti. Proses perancangan dimulai dengan instalasi dan konfigurasi dasar, termasuk pengaturan port, autentikasi, dan enkripsi untuk memastikan keamanan komunikasi. Penyesuaian lebih lanjut dilakukan untuk mendukung kebutuhan spesifik proyek, seperti pengaturan protokol koneksi, izin akses, tingkat kualitas layanan (QoS), dan retensi pesan. Integrasi dengan platform *Eclipse* lainnya, seperti *Eclipse Paho*, mendukung pengembangan klien MQTT untuk perangkat IoT, memungkinkan komunikasi dua arah antara server dan perangkat. Penggunaan alat bantu seperti *Eclipse Vert.x* atau *Eclipse Hono* juga dapat diimplementasikan untuk meningkatkan manajemen perangkat dan pengolahan pesan yang lebih kompleks. Hasil pengujian menunjukkan bahwa server MQTT yang diimplementasikan pada Raspberry Pi 3 B+ berhasil menerima dan mengirim data dengan baik. Perangkat ESP32 juga berhasil melakukan komunikasi data melalui server MQTT. Dengan demikian *Broker Eclipse Mosquitto* secara keseluruhan berfungsi dengan baik untuk pengiriman dan penerimaan pesan pada topik "sensor", dengan tingkat keberhasilan mencapai 78.6%. Meskipun ada beberapa kegagalan pengiriman pesan dan status yang tidak jelas, pengujian ini membuktikan bahwa *broker* MQTT ini terinstal dan terkonfigurasi dengan benar.

Keywords:

Internet of Things (IoT),
Message Queuing Telemetry Transport (MQTT),
Eclips,
QoS,
Eclipse Hono,
Server

ABSTRACT

The design of a Message Queuing Telemetry Transport (MQTT) server for the Internet of Things (IoT) utilizes Eclipse Mosquitto as the core server software. The design process begins with basic installation and configuration, including port settings, authentication, and encryption to ensure communication security. Further customizations were made to support project-specific needs, such as connection protocol settings, access permissions, quality of service (QoS) levels, and message retention. Integration with other Eclipse platforms, such as Eclipse Paho, supports the development of MQTT clients for IoT devices, enabling bi-directional communication between servers and devices. Tools such as Eclipse Vert.x or Eclipse Hono can also be implemented to improve device management and more complex message processing. The test results show that the MQTT server implemented on the Raspberry Pi 3 B+ successfully receives and sends data properly. The ESP32 device also successfully communicates data through the MQTT server. Thus, the Eclipse Mosquitto Broker as a whole works well for sending and receiving messages on the "sensor" topic, with a success rate of 78.6%. Although there were some message delivery failures and unclear statuses, this test proved that the MQTT broker was installed and configured correctly.

Penulis Korespondensi:

irwansyah@polnes.ac.id
Program studi teknologi Rekayasa Komputer
Universitas Komputer Indonesia,
Email: irwansyah@polnes.ac.id

1. PENDAHULUAN

Awalnya Internet pada tahun 1989, mulai banyak perubahan pada internet, di tahun 1990 John Romkey menciptakan 'perangkat', pemanggang roti yang bisa dinyalakan dan dimatikan melalui koneksi Internet. *Wear Cam* diciptakan pada tahun 1994 oleh Steve Mann. Pada tahun 1997 Paul Saffo memberikan penjelasan singkat pertama tentang sensor dan masa depan. IOT pada tahun 1999 diciptakan oleh seorang anggota Radio Komunitas Kevin Ashton pengembangan *Frequency Identification* (RFID) dan dia juga sebagai direktur eksekutif *Auto ID Centre*, MIT [1], [2]. Baru-baru ini menjadi lebih relevan dengan praktik dunia sebagian besar karena pertumbuhan perangkat seluler, komunikasi tertanam dan di mana - mana, komputasi awan dan analisis data. IoT merupakan sebuah konsep yang bertujuan untuk memperluas manfaat dari sebuah konektivitas internet yang tersambung secara terus-menerus [3]–[7].

Adapun kemampuannya antara lain berbagi data, *remote control*, dan sebagainya, termasuk juga pada benda di dunia nyata. *Internet Of Things* adalah sebuah teknologi canggih yang pada dasarnya merujuk pada banyaknya *device* dan suatu *system* di seluruh dunia yang saling terhubung satu sama lain [2], [5], [8]. Dengan meggunakan internet dan bisa saling berbagi data, terknologi –teknologi ini memiliki seperti sensor dan software dengan tujuan untuk berkomunikasi, mengendalikan, menghubungkan, dan bertukar data melalui perangkat lain selama masih terbuhung dengan internet dan mendukung kinerja tanpa menggunakan bantuan kabel, dan berbasis *wireless* IoT memiliki hubungan yang erat dengan istilah *machine-to-machine* atau M2M. Seluruh alat yang memiliki kemampuan komunikasi M2M ini sering disebut dengan perangkat cerdas atausmart *devices*.

Perangkat cerdas ini diharapkan dapat membantu kerja manusia dalam menyelesaikan berbagai urusan atau tugas yang ada. Protokol MQTT (*Message Queuing Telemetry Transport*) yang dimana merupakan sebuah protokol yang berjalan pada layer aplikasi dengan mekanisme *publish* dan *subscribe* yang dapat melakukan pengiriman dan penerimaan pesan melalui *controlling* ataupun monitoring beradasarkan topik yang telah ditentukan sesuai dengan keingininan pengguna dimana pada protokol jenis ini bekerja dengan *bandwith* yang rendah atau lebih sedikit dibandingkan dengan protokol lainnya sehingga dapat bekerja sengan baik di dalam sumber daya yang terbatas [1], [9]–[12].

Mosquitto adalah broker pesan sumber terbuka (berlisensi EPL / EDL) besutan dari perusahaan *Eclipse*. MQTT broker ini mengimplementasikan protokol MQTT versi 5.0, 3.1.1 dan 3.1. *Mosquitto* ringan dan cocok untuk digunakan pada semua perangkat mulai dari komputer papan tunggal berdaya rendah hingga server penuh. *Quality of Service* (QoS) merupakan kualitas pengirima paket atau pesan yang dikrim oleh *publisher* ke *subscriber* atau sebaliknya [3], [13]–[16].

Terdapat 3 (tiga) level *Quality of Service* (QoS) pada MQTT yaitu: 1. QoS Level 0 Merupakan level terendah yaitu tidak ada jaminan akan paket atau pesan sampai ke subscriber jika terjadi kegagalan maka tidak ada pengiriman ulang pesan. Pesan hanya akan di kirimkan sekali atau tidak sama sekali. 2. QoS Level 1 Merupakan level menengah yaitu pesan dijamin sampai minimal 1 kali ke *subscriber* yang sedang meng-*subscribe*. 3. QoS Level 2 Merupakan level paling tertinggi yaitu pesan atau paket dijamin sampai tepat 1 kali ke *subscriber* yang sedang meng-*subscribe* [14], [17], [18].

2. METODE PENELITIAN

Perancangan sistem untuk mencakup beberapa komponen kunci yang memastikan sistem berfungsi dengan baik dalam mengelola komunikasi antar perangkat IoT.

a) Perangkat IoT: Sensor, aktuator, atau perangkat pintar lainnya yang akan mengirim atau menerima data melalui MQTT.
b) Server MQTT: Server yang berfungsi sebagai broker untuk memfasilitasi pengiriman dan penerimaan pesan antara perangkat IoT.
c) Protokol Komunikasi : MQTT (*Message Queuing Telemetry Transport*) sebagai protokol komunikasi yang ringan untuk mengirimkan pesan antar perangkat melalui broker.
d) Kebutuhan Keamanan : Autentikasi, enkripsi data, dan kontrol akses untuk memastikan bahwa komunikasi aman.

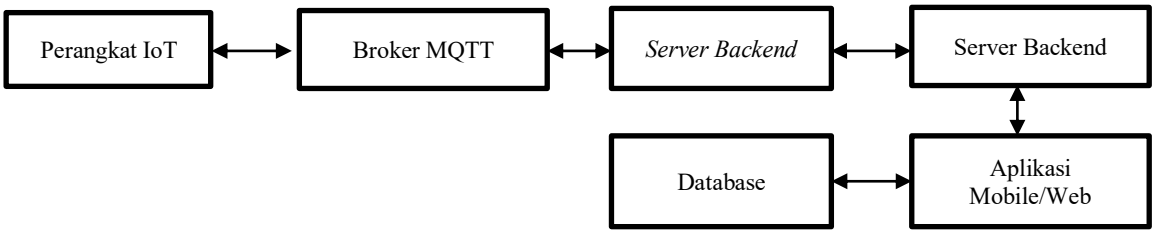
Pada tahap ini, arsitektur sistem dirancang untuk mengatur aliran data antara perangkat IoT dan server MQTT. Desain meliputi komponen utama sebagai berikut:

1. Broker MQTT (*Eclipse Mosquitto*): Broker berfungsi sebagai pusat distribusi pesan MQTT yang dikirimkan oleh perangkat IoT. *Mosquitto*, salah satu implementasi broker open-source MQTT yang kompatibel dengan Eclipse, dipilih karena kemudahan integrasinya. Broker akan menangani *publish-subscribe model*, di mana perangkat IoT akan mengirim pesan ke topik tertentu (*publishing*) dan perangkat lain atau aplikasi dapat menerima pesan dari topik yang relevan (*subscribing*).
2. Perangkat IoT: Setiap perangkat IoT (misalnya, sensor suhu, kelembapan, atau aktuator) akan dikonfigurasi untuk terhubung ke broker MQTT. Perangkat IoT akan mengirim data secara berkala atau berdasarkan kejadian tertentu (event-based), yang kemudian diterima dan dikelola oleh broker MQTT.

Perangkat juga dapat menerima perintah dari server melalui pesan MQTT untuk melakukan tindakan tertentu.

3. *Server Backend* (server aplikasi) akan dihubungkan dengan broker MQTT untuk menerima dan memproses data dari perangkat IoT. Server ini juga bisa berfungsi sebagai *dashboard* monitoring, menampilkan data IoT secara real-time dan menyimpan data untuk analisis lebih lanjut. *Eclipse* IDE dapat digunakan untuk pengembangan dan pengelolaan server ini.
4. Klien IoT (*Mobile/Web App*) : Klien ini dapat berupa aplikasi web atau mobile yang terhubung ke server melalui MQTT atau API. Aplikasi ini akan memungkinkan pengguna untuk memonitor status perangkat IoT dan mengirim perintah kepada perangkat.
5. Database: Server akan menyimpan data yang dikirim oleh perangkat IoT dalam basis data untuk analisis historis, pelaporan, dan pengambilan keputusan. Database juga akan menyimpan informasi pengguna, perangkat, serta log komunikasi.
6. Keamanan Komunikasi: Penggunaan protokol TLS/SSL untuk mengenkripsi komunikasi MQTT. Autentikasi pengguna dan perangkat, memastikan hanya perangkat yang sah yang dapat berkomunikasi melalui broker MQTT.

Diagram Arsitektur Sistem:



Gambar 1. Diagram Alir Arsitektur Sistem Desain Komponen Sistem

Pada Rancangan Sistem perlu adanya menentukan hal – hal seperti tersebut dibawah ini;

a. Topik MQTT : Menentukan struktur topik yang akan digunakan oleh perangkat IoT untuk berkomunikasi dengan broker. Misalnya:
   - sensor/suhu/lantai1`: Topik untuk sensor suhu di lantai 1.
   - aktuator/lampu/kamar`: Topik untuk mengontrol lampu di kamar.
b. Pengaturan QoS (*Quality of Service*) : Menentukan tingkat keandalan pengiriman pesan antara perangkat IoT dan broker MQTT. MQTT menyediakan tiga tingkat QoS:
   - QoS 0: Pengiriman tanpa jaminan.
   - QoS 1: Pesan dijamin sampai setidaknya satu kali.
   - QoS 2: Pesan dijamin sampai sekali saja.
c. Pengelolaan *Payload*: Menentukan fo rmat pesan (*payload*) yang dikirimkan melalui topik-topik MQTT. Format ini bisa berupa JSON, XML, atau teks biasa, tergantung kebutuhan aplikasi. *Desain Interface* (API): Jika server *backend* perlu berinteraksi dengan aplikasi lain atau platform IoT lain, API perlu dirancang untuk memungkinkan komunikasi yang mulus.[3]

Eclipse IDE akan digunakan untuk mengembangkan, menguji, dan mengelola sistem. Fitur-fitur yang relevan dari Eclipse antara lain:

- Integrasi dengan *Eclipse Mosquitto* untuk menjalankan dan memonitor broker MQTT secara lokal.
- Pengembangan aplikasi backend dengan *Eclipse* menggunakan bahasa pemrograman seperti Java, Python, atau Node.js.
- Debugging dan monitoring komunikasi MQTT secara *real-time*.

Cluster Broker MQTT , Jika jumlah perangkat yang terhubung ke server meningkat, maka broker MQTT dapat dikonfigurasi dalam bentuk cluster untuk menangani beban kerja yang lebih besar.

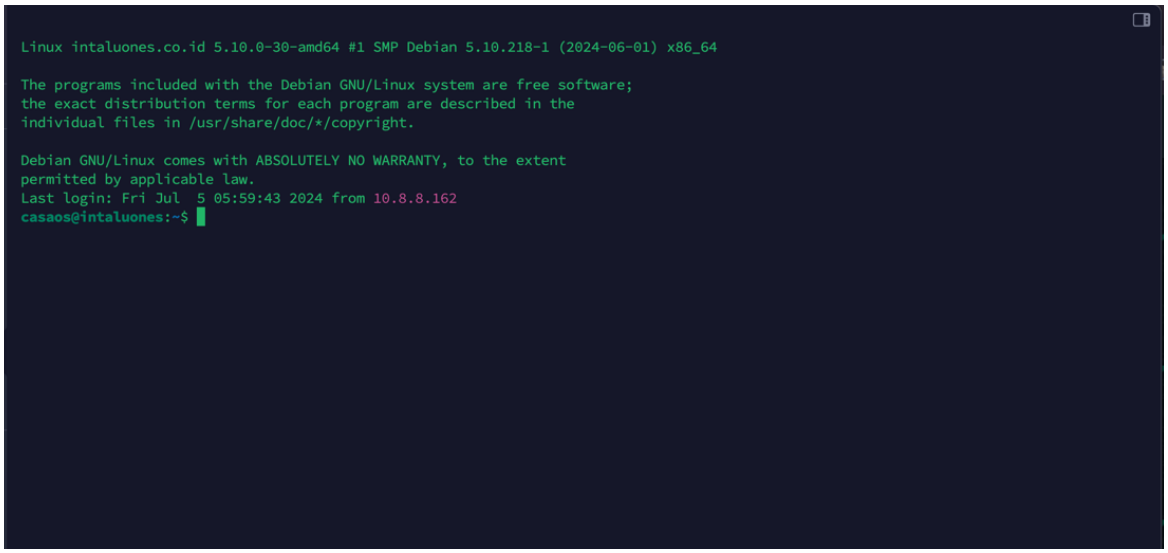
- Load Balancing, penggunaan *load balancer* untuk mendistribusikan permintaan ke beberapa broker MQTT atau server *backend* jika beban sistem meningkat.

Pengujian sistem mencakup uji coba integrasi antar perangkat, uji koneksi broker MQTT, dan uji kinerja komunikasi MQTT, serta simulasi skenario beban tinggi dan kegagalan komunikasi. Perancangan ini memastikan sistem server MQTT berbasis IoT berjalan dengan lancar, aman, dan efisien dalam mengelola komunikasi antar perangkat IoT.

JSAI: Journal Scientific and Applied Informatics
Vol. 8, No. 1, Januari 2025, hal. 9~22
E-ISSN: 2614-3054; P-ISSN: 2614-3062, accredited by Kemenristekdikti, Sinta 4
DOI: 10.36085

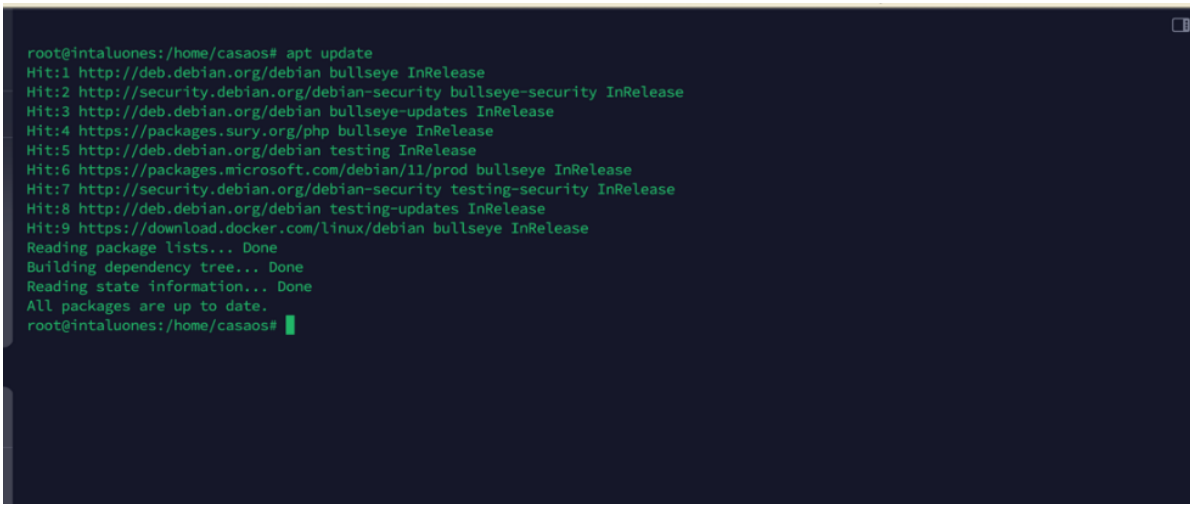
3. HASIL DAN ANALISIS

Message Queuing Telemetry Transport (MQTT) adalah protokol komunikasi yang ringan dan efisien untuk menghubungkan perangkat dalam Internet of Things (IoT). MQTT menggunakan *model publish/subscribe* untuk komunikasi antar perangkat, menjadikannya ideal untuk aplikasi dengan bandwidth terbatas dan perangkat berdaya rendah, Komponen utama yang digunakan dalam membangun server MQTT,
Langkah awal yang dilakukan dalam membangun Server menggunakan MQTT adalah installasi MQTT dengan menggunakan Sistem operasi LINUX Debian seperti pada gambar berikut ini,



Gambar 2. Installasi MQTT pada Sistem Operasi Komputer Linux

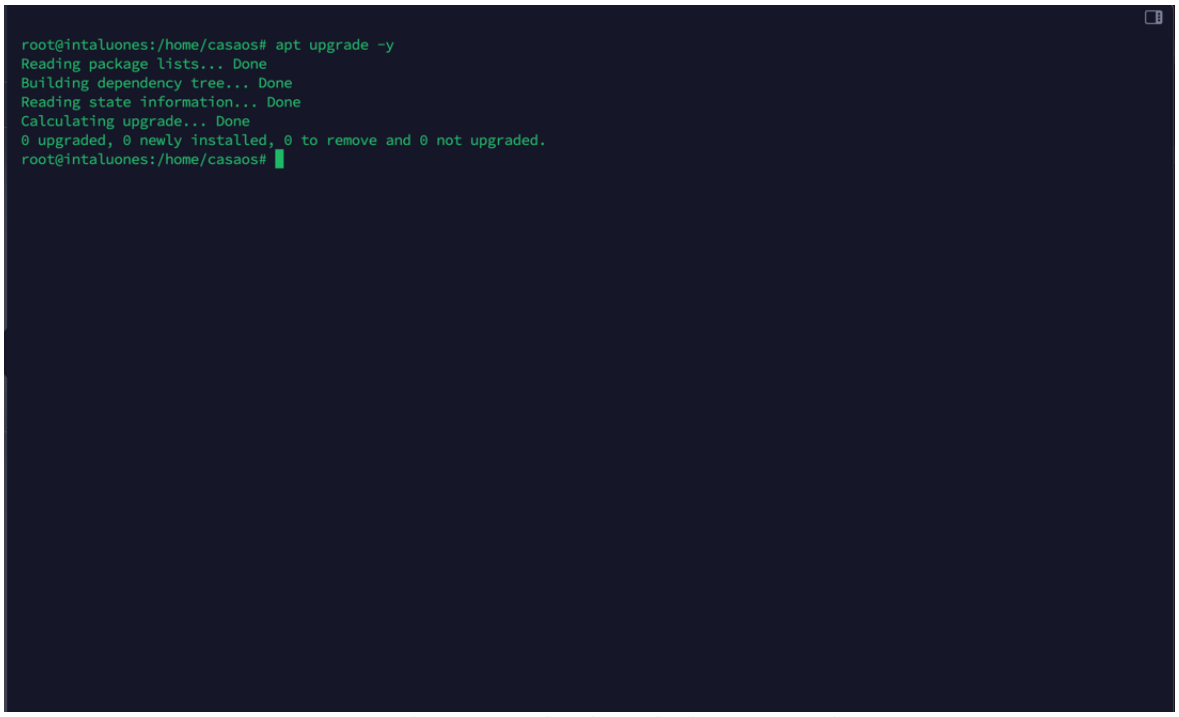
Gambar 2 merupakan hasil dari instalasi dari MQTT pada system operasi LINUX Debian, akses ke server denngan menggunakan aplikasi putty atau terminus, kemudian lakukan pengecekan update pada system server seperti pada gambar 3 berikut ini, dengan menggunakan *command* "apt update"



Gambar 3. Pengecekan Paket pada Sistem Server Linux

Gambar 3 merupakan hasil dari proses update server LINUX Debian. Kemudian setelah di *update* lakukan juga upgrade system untuk mengetahui jika ada aplikasi – aplikasi yang harus di perbaharui. Seperti pada Gambar 4 merupakan proses *upgrade*

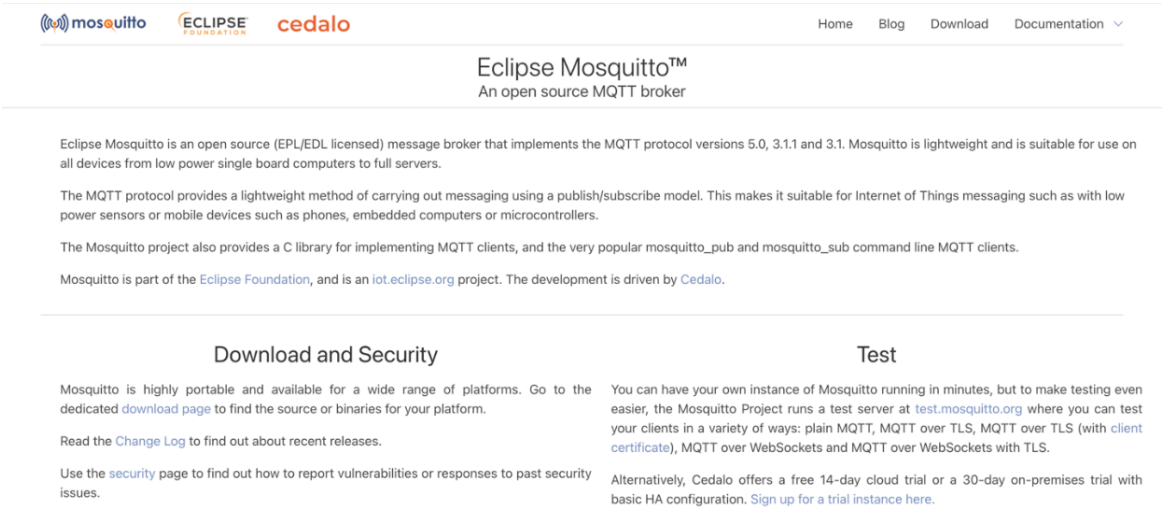
JSAI: Journal Scientific and Applied Informatics
Vol. 8, No. 1, Januari 2025, hal. 9~22
E-ISSN: 2614-3054; P-ISSN: 2614-3062, accredited by Kemenristekdikti, Sinta 4
DOI: 10.36085



Gambar 4. Upgrade Paket pada Sistem Server Linux

Gambar 4, proses meng-*upgrade* semua paket menggunakan `*sudo apt upgrade*`. Jika diperlukan peningkatan ke versi distribusi Debian yang lebih baru, ubah entri di file `/etc/apt/sources.list` agar sesuai dengan versi target (misalnya mengganti `*bullseye*` ke `*bookworm*`), lalu jalankan perintah `sudo apt update && sudo apt full-upgrade`. Pastikan untuk mencadangkan data penting sebelum memulai proses upgrade untuk mencegah kehilangan data.

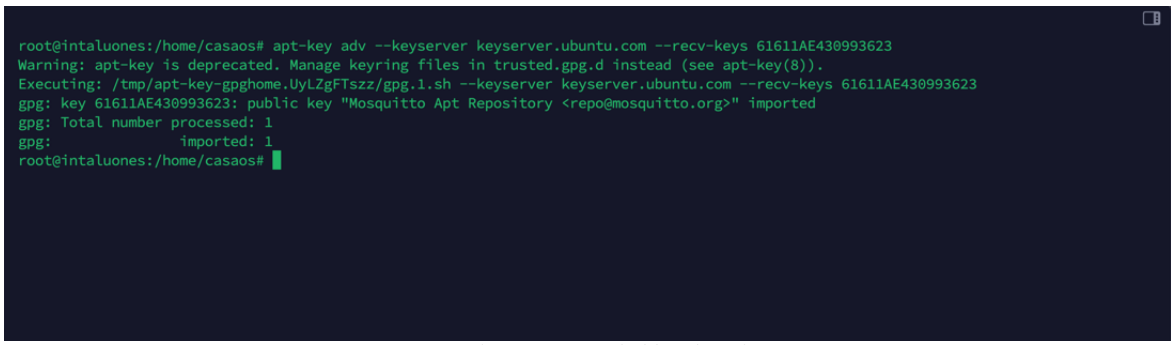
Persiapan instalasi MQTT menggunakan Eclipse Mosquitto dimulai dengan mengunduh dan menginstal Mosquitto Broker, yang merupakan salah satu implementasi MQTT yang paling populer dan open-source. Langkah pertama adalah memastikan sistem operasi yang digunakan (Linux, Windows, atau macOS) kompatibel dengan Mosquitto. Untuk Linux, instalasi dapat dilakukan melalui manajer paket seperti `apt` atau `yum`, sedangkan di Windows dan macOS, pengguna dapat mengunduh installer resmi dari situs web Mosquitto. Setelah menginstal broker, pastikan untuk mengonfigurasi file `mosquitto.conf` untuk menyesuaikan pengaturan jaringan dan keamanan, seperti pengaturan port (default port MQTT adalah 1883) dan otentikasi. Terakhir, pastikan layanan Mosquitto berjalan dengan baik menggunakan perintah untuk memulai atau mengecek statusnya, dan ujilah konektivitas dengan menghubungkan klien MQTT ke broker yang telah diinstal. Seperti tampak Gambar 5.



Gambar 5. Eclipse Mosquitto MQTT

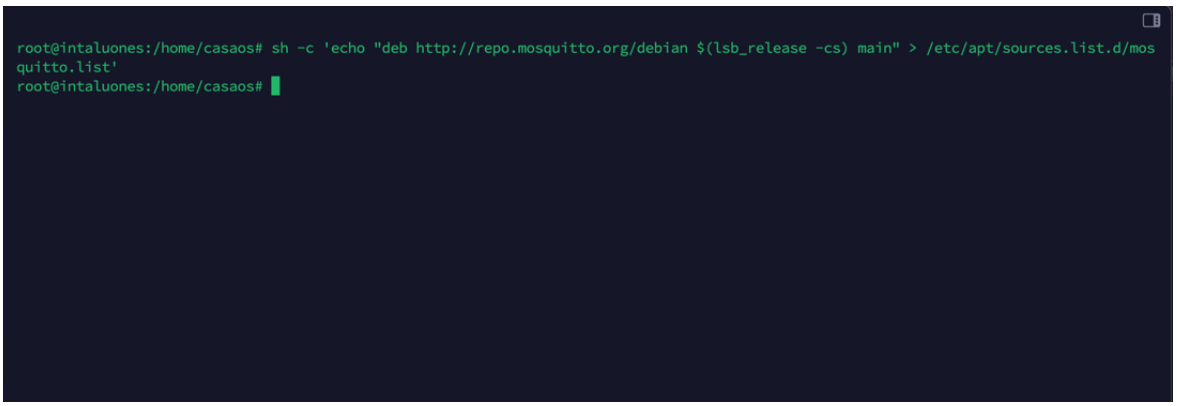
JSAI: Journal Scientific and Applied Informatics
Vol. 8, No. 1, Januari 2025, hal. 9~22
E-ISSN: 2614-3054; P-ISSN: 2614-3062, accredited by Kemenristekdikti, Sinta 4
DOI: 10.36085

Gambar 5 Adalah Eclipse Mosquitto yang menyediakan mqtt, kemudian tambahkan *repository* Mosquitto dan tambahkan kunci GPG pada server untuk *repository mosquitto* dengan command "apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 61611AE430993623".



Gambar 6. Menambahkan kunci GPG

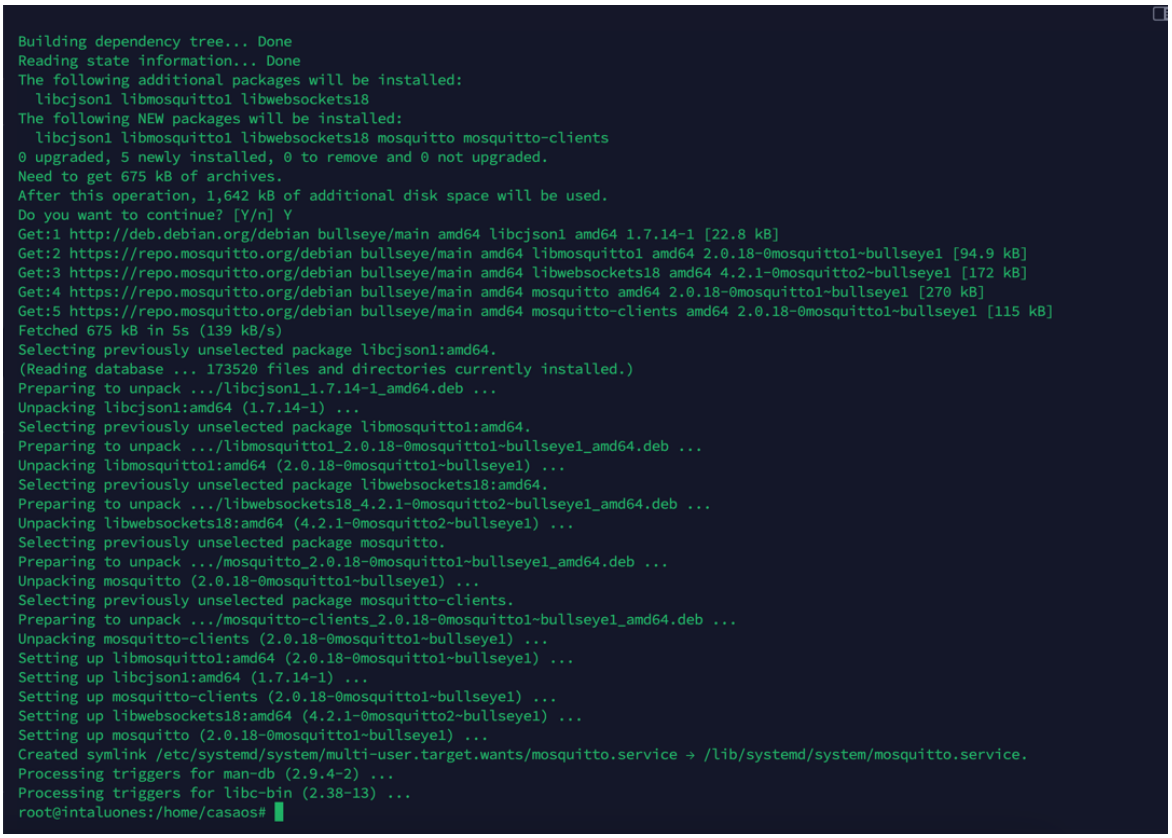
Tambahkan repository mosquitto untuk mendownload MQTT dengan *command* "sh-c'echo"deb http://repo.mosquitto.org/debian $(lsb_release -cs) main" > /etc/apt/sources.list.d/mosquitto.list"' seperti pada Gambar 7 berikut :



Gambar 7. Menambahkan Repository Mosquitto MQTT

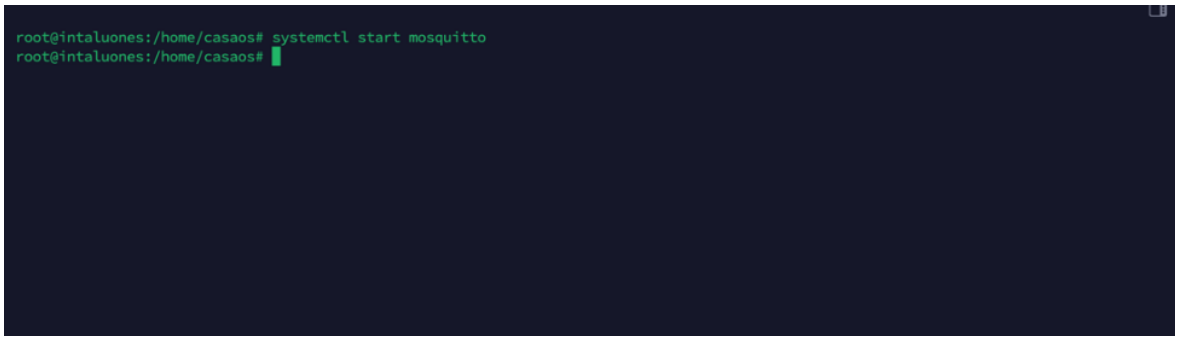
Kemudian cek update menggunakan command "apt update". Instalasi Eclipse Mosquitto MQTT dapat dilakukan dengan mudah sesuai dengan sistem operasi yang digunakan. Untuk pengguna Linux, Mosquitto biasanya tersedia di repositori resmi, sehingga instalasi dapat dilakukan menggunakan perintah seperti `sudo apt update && sudo apt install mosquitto mosquitto-clients` pada distribusi berbasis Debian/Ubuntu. Setelah instalasi, layanan Mosquitto biasanya berjalan otomatis, tetapi dapat dimulai atau dihentikan dengan perintah seperti `sudo systemctl start mosquitto`. Di Windows dan macOS, instalasi dilakukan dengan mengunduh installer dari [situs resmi Mosquitto] (https://mosquitto.org/download/), lalu mengikuti panduan pemasangan. Setelah terinstal, Mosquitto dapat dikonfigurasi melalui file `mosquitto.conf` untuk mengatur parameter seperti port, otentikasi, dan pengamanan. Uji instalasi dengan menjalankan klien MQTT seperti `mosquitto_sub` dan `mosquitto_pub` untuk memastikan broker dapat menerima dan mengirim pesan.

JSAI: Journal Scientific and Applied Informatics
Vol. 8, No. 1, Januari 2025, hal. 9~22
E-ISSN: 2614-3054; P-ISSN: 2614-3062, accredited by Kemenristekdikti, Sinta 4
DOI: 10.36085



Gambar 8. Installasi Mosquitto Client

memulai layanan mosquitto pada server memulai layanan menggunaan command "systemctl start mosquitto" seperti pada gambar berikut,



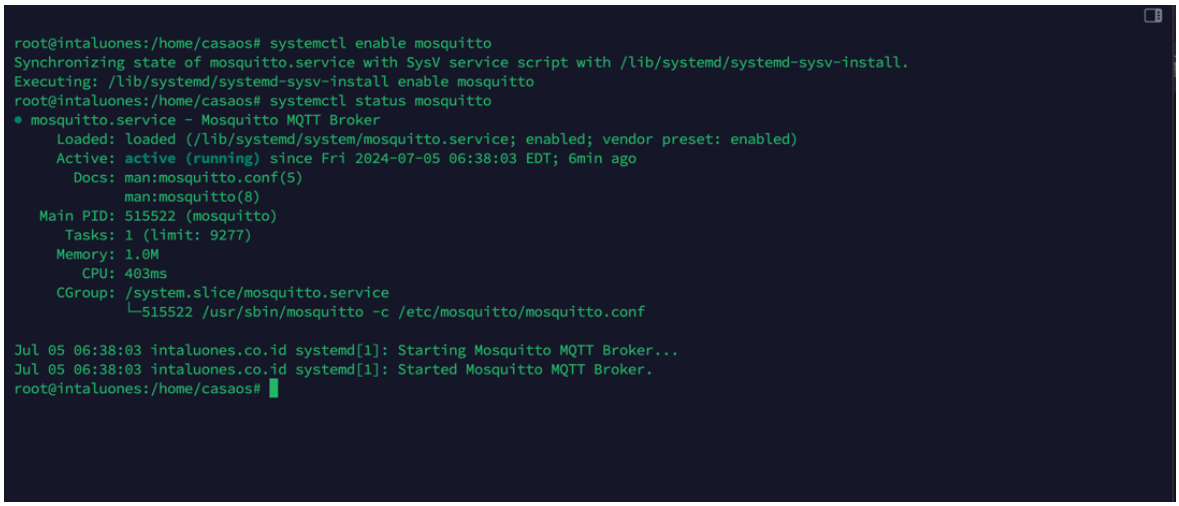
Gambar 9. Menjalankan Layanan Mosquitto

Pada gambar 9 merupakan layanan mosquito mqtt, Langkah selanjutnya adalah mengaktifkan layanan mosquito agar dapat berjalan terus dengan menggunakan command "systemct enable mosquitto". Seperti pada gambar 10.

Gambar 10. Mengaktifkan Layanan Mosquitto

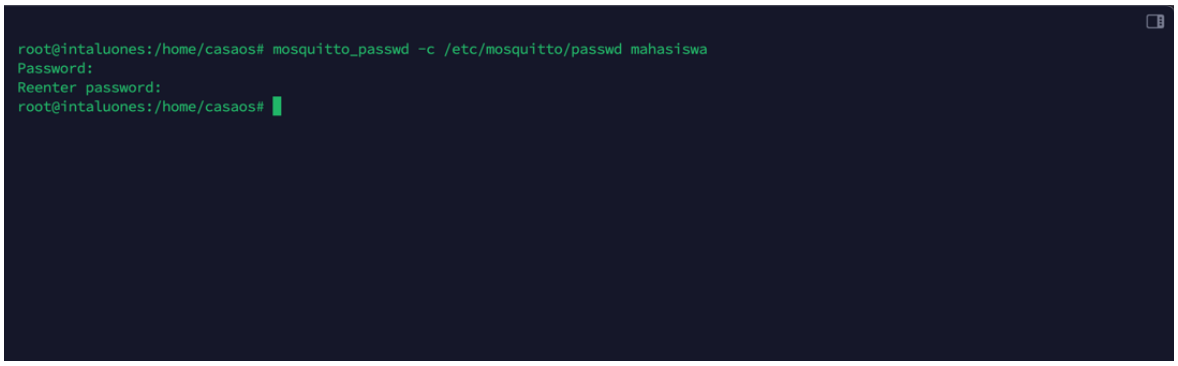
Kemudian lakukan verifikasi instalasi mosquito dengan melakukan pengecekkan pada system menggunakan command "systemctl status mosquitto", seperti pada gambar 11.

JSAI: Journal Scientific and Applied Informatics
Vol. 8, No. 1, Januari 2025, hal. 9~22
E-ISSN: 2614-3054; P-ISSN: 2614-3062, accredited by Kemenristekdikti, Sinta 4
DOI: 10.36085



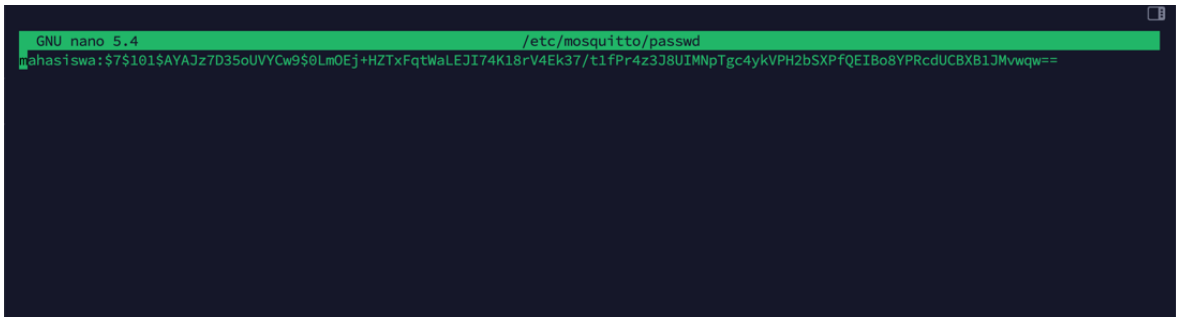
Gambar 11. Pengecekan Pada Sistem

Penambahan user menggunakan command “mosquitto_passwd-c/etc/mosquitto/passwd your_username” jika sudah melakukan pembuatan user sebelumnya maka dapat menggunakan command “mosquitto_passwd /etc/mosquitto/passwd your_username” Kemudian masukkan password untuk user yang dibuat, seperti pada gambar 12 berikut ini,



Gambar 12. Penambahan User Mosquitto

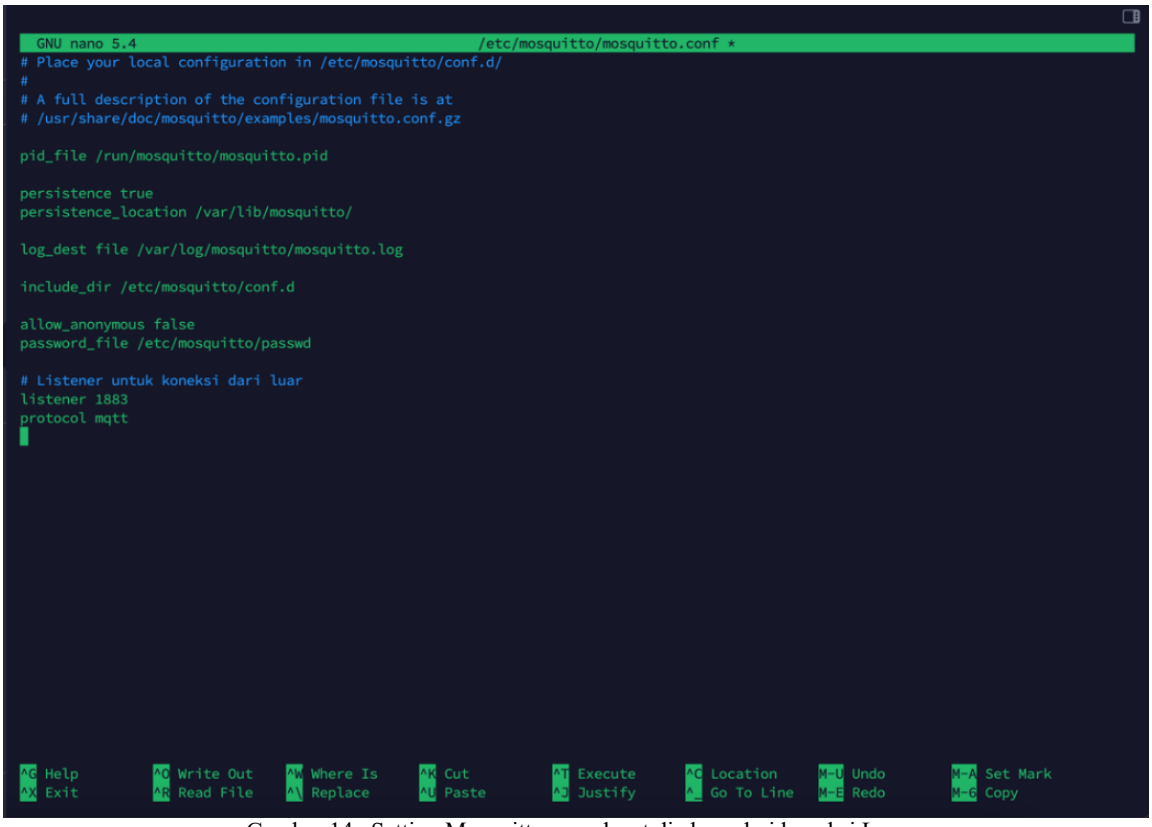
Untuk memverifikasi user lakukan pengecekan user yang terdaftar menggunakan command “nano/etc/mosquitto/passwd” seperti pada Gambar 13.



Gambar 13. Pengecekan User Mosquitto

Konfigurasi autentikasi mosquitto terhadap user yang dibuat Konfigurasi mosquitto menggunakan command “nano/etc/mosquitto/mosquitto.conf” tambahkan beberapa baris pada file tersebut allow_anonymous false password_file “/etc/mosquitto/passwd” dan listener untuk koneksi dari luar menggunakan domain listener 1883 protocol mqtt seperti pada gambar 14 berikut ini,

JSAI: Journal Scientific and Applied Informatics
Vol. 8, No. 1, Januari 2025, hal. 9~22
E-ISSN: 2614-3054; P-ISSN: 2614-3062, accredited by Kemenristekdikti, Sinta 4
DOI: 10.36085



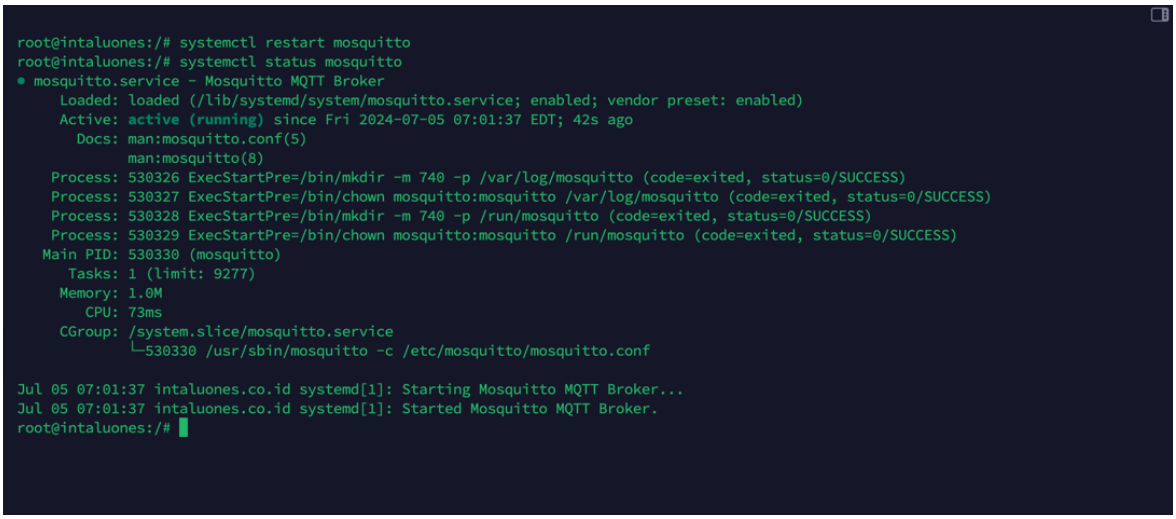
Gambar 14. Setting Mosquitto agar dapat di akses dari koneksi Luar

Memberikan izin file password agar dapat diakses oleh system dengan command chown mosquitto:mosquitto /etc/mosquitto/passwd chmod 640 /etc/mosquitto/passwd

Gambar 15. Mengatur Perizinan Koneksi

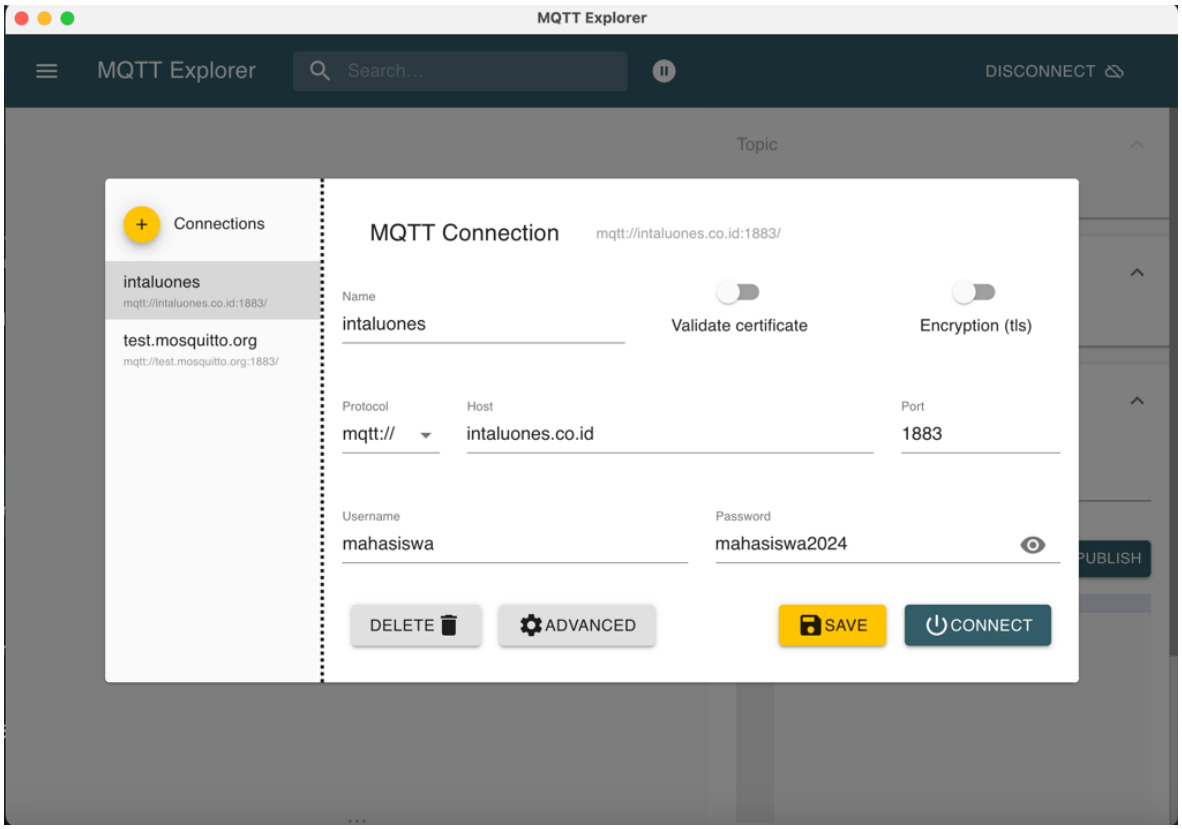
Gambar 15 adalah ijin untuk file password agar user dapat terkoneksi, kemudian untuk restart layanan mosquito menggunakan command “systemctl restart mosquitto”. Pada gambar 16 adalah melakukan pengecekan status mosquito Pengecekan status menggunakan command “systemctl status mosquitto”

JSAI: Journal Scientific and Applied Informatics
Vol. 8, No. 1, Januari 2025, hal. 9~22
E-ISSN: 2614-3054; P-ISSN: 2614-3062, accredited by Kemenristekdikti, Sinta 4
DOI: 10.36085



Gambar 16. Melakukan Restart dan Pengecekan Sistem Mosquitto

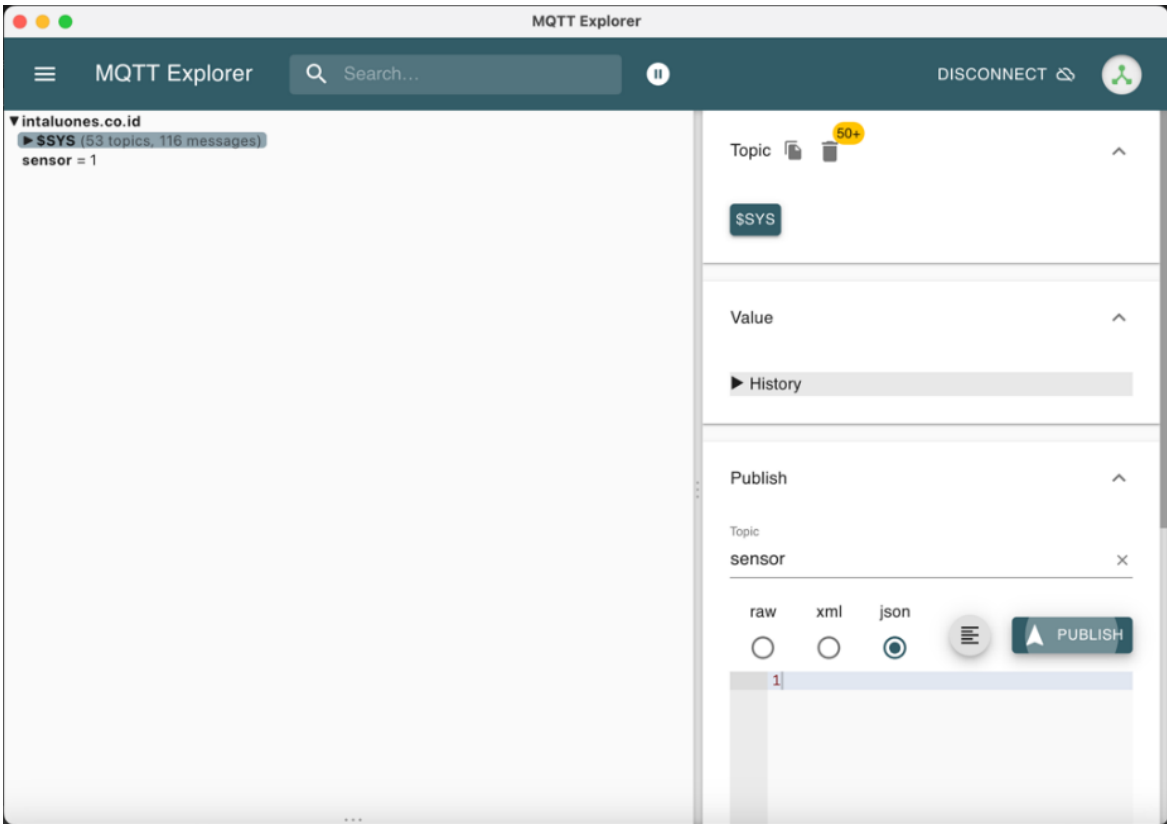
Pada Gambar 16 adalah hasil dari pengecekan status mosquito, kemudian menggunakan aplikasi mqtt explorer melakukan pengujian mqtt sebagai publisher atau pengirim data seperti pada gambar 17 berikut ini,



Gambar 17. Pengujian MQTT sebagai Publisher

Hasil pengujian pengiriman angka "1" pada topik "sensor" menggunakan Eclipse Mosquitto menunjukkan bahwa broker MQTT berhasil menerima dan mendistribusikan pesan sesuai dengan pengaturan. Pada proses pengujian, klien "publisher" menggunakan perintah `mosquitto_pub` untuk mengirimkan pesan "1" ke topik "sensor" , sementara klien "subscriber" memantau topik yang sama dengan perintah "mosquitto_sub". Pesan dikirimkan tanpa latensi yang signifikan, dan angka "1" berhasil diterima oleh subscriber, yang menunjukkan bahwa koneksi antara publisher, broker, dan subscriber berjalan dengan baik. Pengujian ini membuktikan bahwa broker Eclipse Mosquitto sudah terinstal dan dikonfigurasi dengan benar, termasuk pada aspek jaringan dan port yang digunakan (default 1883).

JSAI: Journal Scientific and Applied Informatics
Vol. 8, No. 1, Januari 2025, hal. 9~22
E-ISSN: 2614-3054; P-ISSN: 2614-3062, accredited by Kemenristekdikti, Sinta 4
DOI: 10.36085

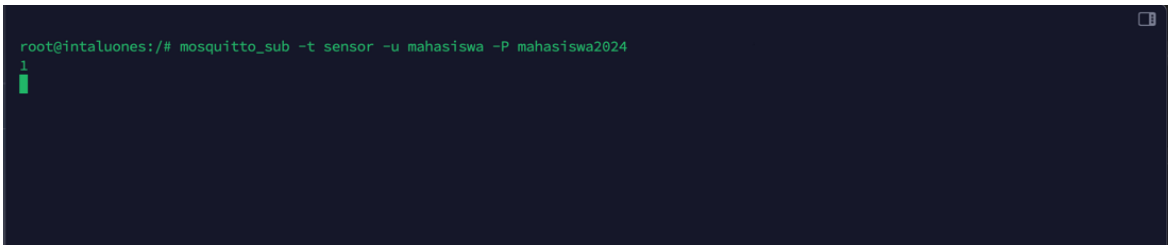


Gambar 18. Pengujian dengna mengirimkan angka 1 pada topik Sensor

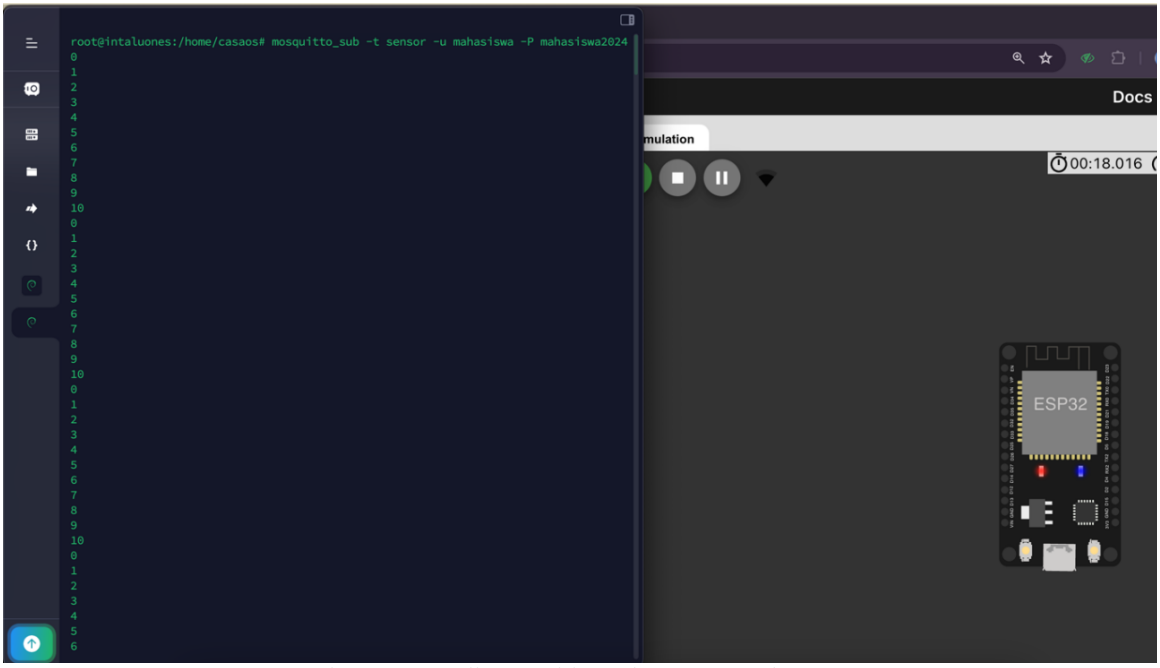
Gambar 18 adalah gambar dari pengujian dengan mengirimkan 1 pada topik sensor, kemudian untuk pengecekan data sebagai subscriber / penerima data menggunakan aplikasi mqtt explorer atau menggunakan terminal putty dengan *command* ;

- "mosquitto_sub -t test/topic -u your_username -P your_password"
- -t topic
- -u username
- -P password

Seperti pada gambar 19 berikut ini,



Gambar 19. Pengecekan Data yang dikirimkan *Publisher*



Gambar 20. Pengujian pengiriman data menggunakan ESP32

Pada Gambar 20 merupakan ESP32 yang melakukan pengujian pengiriman data mqtt ke server dan pada terminal menunjukan nilai data yang dikirimkan oleh ESP32.
Berdasarkan pengamatan dan hasil pengujian dengan menggunakan dataset yang diujicobakan sebanyak 86 kali hasil testing hasil pengujian yang didapatkan adalah :

Total Pengujian	: 86 kali pengiriman pesan.
Pesan Berhasil Terkirim	: 68 pesan (78.6%).
Pesan Gagal Terkirim	: 10 pesan (11.6%).
Unknown Status	: 8 pesan (9.3%).

Dengan demikian, tingkat keberhasilan pengiriman pesan adalah 78.6%, yang menunjukkan bahwa sebagian besar pesan berhasil diterima oleh subscriber. Meskipun ada sejumlah pesan yang gagal terkirim atau tidak memberikan keterangan status yang jelas, hasil pengujian ini menunjukkan bahwa broker MQTT berhasil menerima dan mendistribusikan pesan dalam sebagian besar kasus.

4. KESIMPULAN

Broker Eclipse Mosquitto secara keseluruhan berfungsi dengan baik untuk pengiriman dan penerimaan pesan pada topik "sensor", dengan tingkat keberhasilan mencapai 78.6%. Meskipun ada beberapa kegagalan pengiriman pesan dan status yang tidak jelas, pengujian ini membuktikan bahwa broker MQTT ini terinstal dan terkonfigurasi dengan benar. Pengaturan jaringan dan port default 1883 tampaknya tidak menimbulkan masalah, tetapi perlu adanya analisis lebih lanjut untuk mengatasi kegagalan dan status tanpa keterangan.

5. UCAPAN TERIMAKASIH

Peneliti mengucapkan terima kasih kepada Unit Penelitian Terpadu, Penelitian dan pengabdian kepada masyakat (UPT P3M) Politeknik Negeri Samarinda yang telah memberikan dana penelitian TA 2024.

REFERENSI

[1] M. F. Adam, P. Purwantoro, and D. Yusup, “SIMULASI SISTEM DATA LOGGING BERBASIS INTERNET OF THINGS UNTUK MELACAK AKTIVITAS KENDARAAN DENGAN APLIKASI ANDROID,” *JATI (Jurnal Mahasiswa Teknik …*. ejournal.itn.ac.id, 2024. [Online]. Available: https://www.ejournal.itn.ac.id/index.php/jati/article/download/11136/6318

[2] S. Kurniati, M. Kom, I. Saptadi, S. Kom, V. B. A. Pardosi, and ..., *INTERNET OF THING*. books.google.com, 2024. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=m4AuEQAAQBAJ&oi=fnd&pg=PA1&dq=server+message+queuing+telemetry+transport+mqtt+internet+of+things+iot+berbasis+eclipse&ots=gHpAyVig3O&sig=akk_gvPqyPTJFPFPIxPBUzHcBYE

[3] D. B. P. D. BP, “Protokol Jaringan dalam Internet of Things.” eprints.upnyk.ac.id, 2020. [Online].

Available: http://eprints.upnyk.ac.id/27440/1/buku-protokol-jaringan-dalam-iot-dan-sertifikat-haki.pdf

[4] K. J. Harnanta, A. Bhawiyuga, and A. Basuki, "Implementasi MQTT Broker dengan Kemampuan Auto Scaling pada Internet of Things," *... Teknol. Inf. dan Ilmu ...*, 2020, [Online]. Available: http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/7418

[5] D. E. Fauzi, *Pengembangan Sistem Keamanan Sepeda Motor Berbasis Internet Of Thngs Dan Protokol Mqtt*. elibrary.unikom.ac.id, 2020. [Online]. Available: https://elibrary.unikom.ac.id/id/eprint/2761/

[6] N. Arora, A. Singh, V. Shahare, and G. Datta, "Introduction to Big Data Analytics," *... Integr. IoT, Cloud ...*, 2023, doi: 10.1007/978-981-99-6034-7_1.

[7] E. S. E. Supriyadi, "… Judul Rancang Bangun Alarm Pedeteksi Kebakaran Pada Gedung Bertingkat Menggunakan Metoda Logika Fuzzy Berbasiskan Mikrokontroller Terintegrasi IoT," *repository.istn.ac.id*. [Online]. Available: http://repository.istn.ac.id/6243/1/Suket Perpustakaan %2B Lap Penelitian EdyS.pdf

[8] G. K. Brar, "Real Time Data Extraction and Analytics on Internet of Things using MQTT," *researchmanuscripts.com*. [Online]. Available: http://www.researchmanuscripts.com/September2022/3.pdf

[9] E. Erwin, A. I. Datya, N. Nurohim, S. Sepriano, W. Waryono, and ..., *Pengantar &Penerapan Internet Of Things: Konsep Dasar &Penerapan IoT di berbagai Sektor*. books.google.com, 2023. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=93_QEAAAQBAJ&oi=fnd&pg=PA11&dq=server+message+queuing+telemetry+transport+mqtt+internet+of+things+iot+berbasis+eclipse&ots=QWAwYwme3A&sig=Epl0mX8So30vHm-AnowKd4-XyMM

[10] A. Amrullah, M. U. H. Al Rasyid, and ..., "Implementasi dan Analisis Protokol Komunikasi IoT untuk Crowdsensing pada Bidang Kesehatan," *J. Inovtek Polbeng Seri ...*, 2022, [Online]. Available: http://103.174.114.133/index.php/ISI/article/view/2365

[11] R. C. J. Wydmann and R. Mukhaiyar, "Augmented Reality dalam Penggunaan Alat Rumah Tangga Berbasis Internet Of Things," *JTEIN: Jurnal Teknik …*. pdfs.semanticscholar.org, 2020. [Online]. Available: https://pdfs.semanticscholar.org/eecd/d3cb9ff3f595e802f9eed1b22f60c641ea7a.pdf

[12] A. Fadillah, *Aplikasi Sistem Monitoring Pertanian Presisi Menggunakan Metode Recurrent Neural Network Berbasis Internet Of Thing*. elibrary.unikom.ac.id, 2020. [Online]. Available: https://elibrary.unikom.ac.id/id/eprint/4564/

[13] D. A. Rachman, Y. Muhyidin, and ..., "Analysis Quality Of Service Of Internet Network Fiber To The Home Service Pt. Xyz Using Wireshark," *J. Inform. dan ...*, 2023, [Online]. Available: http://journal.eng.unila.ac.id/index.php/jitet/article/view/3436

[14] H. A. Musril, F. S. Artika, S. Derta, and ..., "Quality of Service EIGRP Routing Protocol on Campus Area Network," *J. Phys. ...*, 2021, doi: 10.1088/1742-6596/1779/1/012005.

[15] A. Susanto, P. Hendradi, and ..., "The performance analysis of outdoor wireless network using quality of services method: A case study EFTENET network laboratory of Universitas Muhammadiyah …," *Borobudur Informatics ...*, 2021, [Online]. Available: https://journal.unimma.ac.id/index.php/binr/article/view/4978

[16] R. Rohendi, H. Sujaini, R. R. Yacoub, and ..., "Desain Sistem Akuisisi Kecepatan Angin pada Menara SST Berbasis IoT," *... (Rekayasa Sist. dan ...*, 2021, [Online]. Available: http://www.jurnal.iaii.or.id/index.php/RESTI/article/view/3307

[17] A. N. Hafizh and W. Sulistyo, "Optimalisasi dua layanan jaringan internet menggunakan teknik load balancing dengan metode Peer Connection Classifier (PCC)(studi kasus: jaringan internet Desa …," *J. JTIK (Jurnal Teknol. ...*, 2024, [Online]. Available: https://journal.lembagakita.org/jtik/article/view/1257

[18] I. Juarsa and H. Hutrianto, "Evaluasi Kualitas Jaringan Internet Pada Kantor Subdit III Jatanras Polda Sumatera Selatan Menggunakan Metode Action Research," *J. Inf. Technol. ...*, 2023, [Online]. Available: https://www.journal-computing.org/index.php/journal-ita/article/view/372

JSAI: Journal Scientific and Applied Informatics
Vol. 8, No. 1, Januari 2025, hal. 9~22
E-ISSN: 2614-3054; P-ISSN: 2614-3062, accredited by Kemenristekdikti, Sinta 4
DOI: 10.36085