# IMPLEMENTASI DAN KONFIGURASI *SERVER* IOT DENGAN MOSQUITTO UNTUK SISTEM KOMUNIKASI MQTT DI LABORATORIUM FTI UNIVERSITAS DINAMIKA

**LAPORAN KERJA PRAKTIK**

**Program Studi**

**S1 Teknik Komputer**

**Oleh:**

**Kayla Nisrina Azzahra Rifananda**

**22410200011**

**FAKULTAS TEKNOLOGI DAN INFORMATIKA**

**UNIVERSITAS DINAMIKA**

**2025**

**IMPLEMENTASI DAN KONFIGURASI *SERVER* IOT DENGAN MOSQUITTO UNTUK SISTEM KOMUNIKASI MQTT DI LABORATORIUM FTI UNIVERSITAS DINAMIKA**

Diajukan sebagai salah satu syarat untuk menyelesaikan

Mata Kuliah Kerja Praktik


**Disusun Oleh:**

**Nama : Kayla Nisrina Azzahra Rifananda**

**NIM : 22410200011**

**Program : S1 (Strata Satu)**

**Jurusan : Teknik Komputer**


**FAKULTAS TEKNOLOGI DAN INFORMATIKA**

**UNIVERSITAS DINAMIKA**

**2025**

**LEMBAR PENGESAHAN**

**IMPLEMENTASI DAN KONFIGURASI *SERVER* IOT DENGAN MOSQUITTO UNTUK SISTEM KOMUNIKASI MQTT DI LABORATORIUM FTI UNIVERSITAS DINAMIKA**

Laporan Kerja Praktik oleh

**Kayla Nisrina Azzahra Rifananda**

**NIM. 22410200011**

Telah diperiksa, diuji, dan disetujui

Surabaya, 8 Juli 2025

Disetujui:

| Pembimbing | Penyelia |
|---|---|
| **Heri Pratikno, M.T.** | **Teguh Sutanto, M.Kom.** |
| NIDN. 0716117302 | NIDN. 0713027801 |

Mengetahui,

Ketua Program Studi Teknik Komputer

**Pauladie Susanto, S.Kom., M.T.**

NIDN. 0729047501

**PERNYATAAN**

**PERSETUJUAN PUBLIKASI DAN KEASLIAN KARYA ILMIAH**

Sebagai mahasiswa **Universitas Dinamika**, Saya :

| | |
|---|---|
| Nama | : **Kayla Nisrina Azzahra Rifananda** |
| NIM | : **22410200011** |
| Program Studi | : **S1 Teknik Komputer** |
| Fakultas | : **Fakultas Teknologi dan Informatika** |
| Jenis Karya | : **Laporan Kerja Praktik** |
| Judul Karya | : **IMPLEMENTASI DAN KONFIGURASI *SERVER* IOT DENGAN MOSQUITTO UNTUK SISTEM KOMUNIKASI MQTT DI LABORATORIUM FTI UNIVERSITAS DINAMIKA** |

Menyatakan dengan sesungguhnya bahwa :

1. Demi pengembangan Ilmu Pengetahuan, Teknologi dan Seni, Saya menyetujui memberikan kepada **Universitas Dinamika** Hak Bebas Royalti Non-Eksklusif (***Non-Exclusive Royalty Free Right***) atas seluruh isi/sebagian karya ilmiah Saya tersebut diatas untuk disimpan, dialihmediakan, dan dikelola dalam bentuk pangkalan data (***database***) untuk selanjutnya didistribusikan atau dipublikasikan demi kepentingan akademis dengan tetap mencantumkan nama Saya sebagai penulis atau pencipta dan sebagai pemilik Hak Cipta.
2. Karya tersebut diatas adalah hasil karya asli Saya, bukan plagiat baik sebagian maupun keseluruhan. Kutipan, karya, atau pendapat orang lain yang ada dalam karya ilmiah ini semata-mata hanya sebagai rujukan yang dicantumkan dalam Daftar Pustaka Saya.
3. Apabila dikemudian hari ditemukan dan terbukti terdapat tindakan plagiasi pada karya ilmiah ini, maka Saya bersedia untuk menerima pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada Saya.

Demikan surat pernyataan ini Saya buat dengan sebenar-benarnya.

Surabaya, 8 Juli 2025


Kayla Nisrina Azzahra Rifananda
NIM : 22410200011

# ABSTRAK

*Server* komunikasi berbasis MQTT dibutuhkan untuk mendukung pertukaran data antar perangkat *Internet of Things* (IoT) secara efisien dan aman. Eclipse Mosquitto merupakan broker MQTT *open-source* yang ringan dan mudah dikonfigurasi. Dalam kegiatan ini, *server* Mosquitto dikonfigurasi pada sistem Linux dengan pengaturan autentikasi menggunakan *username* dan *password*, sistem enkripsi SSL/TLS, serta pengaturan hak akses melalui *Access Control List* (ACL). Selain itu, *dashboard* monitoring *real-time* dibuat untuk memantau koneksi dan aktivitas broker, serta buku panduan disusun untuk memudahkan pengguna mengakses *server* melalui mikrokontroler, *smartphone*, dan komputer. Hasil pengujian menunjukkan bahwa komunikasi antar perangkat berhasil dilakukan baik melalui koneksi biasa maupun terenkripsi. Semua perangkat dapat mengirim dan menerima pesan sesuai topik yang ditentukan, dan *dashboard* mampu menampilkan data operasional broker secara *real-time*. Dapat disimpulkan bahwa *server* MQTT yang dibangun telah berfungsi dengan baik, aman, dan mudah digunakan untuk mendukung komunikasi IoT di lingkungan akademik.

**Kata Kunci**: MQTT, Mosquitto, IoT, autentikasi, *dashboard*.

# KATA PENGANTAR

Puji dan syukur ke hadirat Tuhan Yang Maha Esa karena atas rahmat dan karunia-Nya, penulis dapat menyelesaikan kerja praktik beserta laporannya yang berjudul "Implementasi dan Konfigurasi Server IoT dengan Mosquitto untuk Sistem Komunikasi MQTT di Laboratorium FTI Universitas Dinamika" dengan baik dan tepat waktu.

Laporan ini disusun sebagai salah satu syarat akademik dalam menyelesaikan program Sarjana Teknik Komputer di Universitas Dinamika. Pelaksanaan kerja praktik ini memberikan pengalaman dan juga wawasan secara langsung kepada penulis dalam mengelola dan mengimplementasikan sistem komunikasi IoT berbasis MQTT, khususnya dengan menggunakan broker Mosquitto. Selain itu, penulis juga mendapatkan kesempatan untuk membuat buku panduan penggunaan *server* MQTT yang dapat diakses oleh berbagai perangkat, serta memantau kinerja *server* melalui *dashboard* yang telah dikonfigurasi.

Penulis menyadari bahwa laporan ini tidak akan terselesaikan tanpa adanya bantuan, bimbingan, dan dukungan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaika ucapan terima kasih yang sebesar-besarnya kepada:

1. Seluruh keluarga atas dukungan, doa, dan semangat yang diberikan selama proses kerja praktik dan penyelesaian laporannya.
2. Bapak Heri Pratikno, M.T. selaku dosen pembimbing dalam kegiatan kerja praktik. Serta memberi bimbingan, dukungan, serta motivasi kepada penulis sepanjang proses kerja praktik.
3. Bapak Pauladie Susanto, S.Kom., M.T., selaku Ketua Program Studi S1 Teknik Komputer Universitas Dinamika yang telah memberikan izin kepada penulis untuk melakukan kerja praktik.
4. Bapak Teguh Sutanto, M.Kom., selaku koordinator laboratorium FTI Universitas Dinamika yang telah menerima dan memberikan pengalaman baru dalam lingkungan kerja.
5. Bapak Charisma Dimas Affandi, S.T., selaku mentor yang telah memberikan bimbingan dalam melakukan kerja praktik di laboratorium FTI Universitas Dinamika kepada penulis.

6. Ibu Elisabeth Ria Anggreani A.Md.Keb., selaku koordinator kerja praktik di Universitas Dinamika.
7. Teman-teman yang telah memberikan bantuan, semangat dan dukungan selama kerja praktik dan penyusunan laporan kerja praktik.
8. Serta, pihak lainnya yang tidak dapat disebutkan satu persatu yang telah memberikan bantuan dan dukungan kepada penulis.

Penulis menyadari bahwa laporan ini masih memiliki kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan untuk perbaikan di masa mendatang. Semoga laporan ini dapat memberikan manfaat dan menjadi referensi bagi pihak-pihak yang membutuhkan

Surabaya, 5 Juni 2025

Penulis

# DAFTAR ISI

Halaman

# DAFTAR GAMBAR

Halaman

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Perkembangan teknologi *Internet of Things* (IoT) yang sangat pesat telah mengubah cara komunikasi data dan pengelolaan perangkat, terutama di lingkungan yang membutuhkan pemantauan dan pengendalian secara *real-time*. IoT memungkinkan pengiriman data secara otomatis antar perangkat tanpa perlu adanya campur tangan dari manusia, sehingga teknologi ini mendorong terciptanya sistem pintar di berbagai bidang, seperti pada bidang pendidikan dan kesehatan (Abilovani et al., 2018). Pada lingkungan pendidikan, kebutuhan akan infrastruktur komunikasi data yang andal semakin meningkat seiring bertambahnya perangkat-perangkat IoT yang digunakan untuk keperluan pembelajaran, riset, dan sistem pemantauan. Oleh karena itu, dibutuhkan sebuah sistem komunikasi yang tidak hanya efisien dan ringan, tetapi juga mampu menjangkau berbagai jenis perangkat dan jaringan yang berbeda.

Salah satu solusi yang banyak digunakan dalam komunikasi IoT adalah *Message Queuing Telemetry Transport* (MQTT). MQTT adalah sebuah protokol jaringan ringan berbasis pada *publish-subscribe* yang dirancang khusus untuk kondisi jaringan terbatas, ia dikenal karena memiliki kemampuan dalam menghemat *bandwidth*, memiliki latensi rendah, dan efisiensi yang tinggi dalam pengiriman pesan (Eka Sari et al., 2025). Maka dari itu, ia sangat ideal untuk digunakan dalam lingkungan pendidikan.

MQTT bekerja dengan perantara yang disebut dengan broker MQTT, yang tugasnya adalah mengatur distribusi pesan antar perangkat. Salah satu broker MQTT yang banyak digunakan adalah Eclipse Mosquitto, yaitu sebuah perangkat lunak *open-source* yang dikenal karena kemudahannya dalam konfigurasi, keamanannya yang memadai, serta fleksibilitasnya untuk berbagai skala implementasi (Eka Sari et al., 2025). Ini membuatnya sangat cocok untuk digunakan di lingkungan akademik maupun proyek IoT praktis. Penerapan Mosquitto memungkinkan pengelolaan pesan antar perangkat seperti ESP32,

Raspberry Pi, hingga aplikasi *mobile* dan desktop dengan sistem autentikasi dan enkripsi untuk menjaga keamanan data (Eka Sari et al., 2025).

Dalam penelitian sebelumnya menunjukkan bahwa MQTT memiliki performa sangat baik dalam sistem IoT, dengan latensi pengiriman data yang sangat rendah (sekitar 0,0086 detik dan *throughput* tinggi (hingga 9,2 Mbit/s) (Abilovani et al., 2018). Hal ini sangat penting untuk memastikan sistem IoT dapat dipantau dan dikendalikan secara *real-time*, khususnya di lingkungan pendidikan. Aspek skalabilitas juga menjadi pusat perhatian utama dalam pembangunan IoT. Seiring meningkatnya jumlah perangkat, *server* MQTT harus mampu menangani banyak koneksi tanpa mengalami gangguan. Beberapa studi menunjukkan bahwa penerapan broker MQTT dengan pendekatan *auto-scaling* menggunakan *container* (wadah virtual) dan *orchestration tools* seperti Kubernetes dapat memperkuat sistem, mengurangi koneksi yang gagal, dan memastikan sistem tetap stabil meskipun terjadi lonjakan jumlah perangkat (Harnanta et al., 2020).

Selain itu, integrasi perangkat IoT yang menggunakan protokol berbeda seringkali menjadi tantangan. Dengan menggunakan *multi-protocol gateway*, seperti yang mendukung MQTT, HTTP, dan CoAP, komunikasi antar berbagai jenis perangkat dapat disatukan dalam satu sistem *backend* yang terintegrasi. Pendekatan ini terbukti efektif dalam aplikasi *crowdsensing* dan sangat relevan untuk digunakan dalam lingkungan pendidikan yang memiliki perangkat IoT beragam (Amrullah et al., 2022)

Berdasarkan hal-hal yang telah disebutkan di atas, kerja praktik ini difokuskan pada implementasi dan konfigurasi *server* IoT menggunakan Mosquitto MQTT di lingkungan pendidikan, yaitu Universitas Dinamika. Kegiatan ini mencakup pengaturan sistem komunikasi antar perangkat, pengamanan koneksi melalui *username* dan *password*, serta penyusunan panduan penggunaan MQTT broker untuk berbagai perangkat. Dengan demikian, diharapkan sistem IoT Universitas Dinamika dapat berkembang menjadi lebih terintegrasi, efisien, dan mudah digunakan oleh civitas akademika.

## 1.2 Rumusan Masalah

Berdasarkan yang telah disampaikan pada latar belakang, maka dapat disimpulkan bahwa rumusan masalah pada kerja praktik adalah sebagai berikut:

1. Bagaimana cara mengimplementasikan dan mengkonfigurasi *server* MQTT menggunakan Mosquitto untuk mendukung sistem komunikasi IoT di Universitas Dinamika?
2. Bagaimana cara mengamankan komunikasi MQTT dengan autentikasi *user* dan penggunaan SSL?
3. Bagaimana peran *dashboard server* dalam memantau dan mengelola aktivitas komunikasi MQTT?
4. Bagaimana menyusun panduan penggunaan *server* MQTT bagi pengguna dengan berbagai perangkat?

## 1.3 Batasan Masalah

Berdasarkan pada latar belakang dan rumusan masalah, maka dalam pelaksanaan kerja praktik terdapat batasan masalah, antara lain:

1. *Server* MQTT yang digunakan adalah Mosquitto yang berjalan pada sistem operasi Linux.
2. Konfigurasi hanya mencakup pengaturan *username* dan *password*, ACL, *port* SSL, dan file konfigurasi Mosquitto.
3. *Dashboard server* yang dibahas hanya mencakup pemantauan aktivitas koneksi dan trafik MQTT.
4. Panduan penggunaan dibuat hanya untuk tiga jenis perangkat: *smartphone*, mikrokontroler, dan PC.
5. Tidak membahas secara mendalam mengenai pengolahan data sensor pada mikrokontroler.

## 1.4 Tujuan Penelitian

Terdapat beberapa tujuan dari kerja praktik ini, yaitu sebagai berikut:

1. Mengimplementasikan dan mengkonfigurasi *server* MQTT dengan Mosquitto sebagai broker komunikasi untuk sistem IoT.

2. Melakukan konfigurasi keamanan dengan *username*, *password*, ACL, dan SSL/TLS untuk meningkatkan keamanan komunikasi data.
3. Memonitor aktivitas *server* MQTT melalui *dashboard* guna meningkatkan pengelolaan sistem.
4. Menyusun buku panduan (*user guide*) penggunaan *server* MQTT yang mencakup koneksi dari berbagai perangkat klien untuk pengguna di lingkungan Universitas Dinamika.

## 1.5 Manfaat Penelitian

Adapun manfaat dari pelaksanaan kerja praktik ini adalah sebagai berikut:

1. Menyediakan *server* MQTT mandiri yang dapat digunakan sebagai infrastruktur pengembangan proyek IoT.
2. Mempermudah proses pengujian dan implementasi sistem IoT melalui dokumentasi penggunaan yang lengkap.
3. Menambah pengalaman dalam mengelola *server* MQTT, meningkatkan pemahaman tentang protokol komunikasi IoT, dan menghasilkan dokumentasi teknis yang aplikatif.

# BAB II
# GAMBARAN UMUM PERUSAHAAN

## 2.1 Latar Belakang

Laboratorium FTI Universitas Dinamika bagian IoT merupakan fasilitas khusus yang disediakan untuk mendukung kegiatan pendidikan, praktikum, dan penelitian di bidang sistem tertanam (*embedded systems*) serta perancangan sistem Internet of Things (IoT). Laboratorium ini menjadi salah satu sumber daya penting bagi mahasiswa, khususnya Program Studi S1 Teknik Komputer, untuk melakukan pembelajaran dan penelitian secara langsung terkait pengembangan sistem tertanam dan IoT. Laboratorium IoT berfungsi sebagai pusat studi untuk mata kuliah yang berfokus pada sistem tertanam dan perancangan sistem IoT. Peran utamanya adalah membantu mahasiswa dalam kegiatan perkuliahan, praktikum, hingga proyek yang meningkatkan pemahaman teori sekaligus pengaplikasian praktis pada sistem tertanam dan IoT.



Gambar 2.1 Logo Lab FTI Universitas Dinamika
(Sumber: https://labfti.dinamika.ac.id/)

Laboratorium ini dilengkapi dengan 20 set komputer, 20 modul mikrokontroler AT-Mega32, 20 unit Arduino UNO R3 & R4, serta 20 unit modul ESP8266. Fasilitas ini memungkinkan hingga 20 mahasiswa menggunakan perangkat secara bersamaan. Selain itu, tersedia pula berbagai modul sensor dan aktuator yang mendukung kegiatan praktikum dan riset, sehingga mahasiswa dapat mengenal komponen-komponen perangkat keras IoT secara keseluruhan.

Laboratorium ini juga mendukung penelitian dalam bidang pengembangan sistem tertanam dan IoT, serta berperan sebagai pusat kajian teknologi *embedded* dan rekayasa sistem IoT. Laboratorium ini kerap digunakan dalam kegiatan pembelajaran dan penelitian dosen maupun mahasiswa, sehingga berkontribusi besar dalam pengembangan akademik di bidang tersebut.

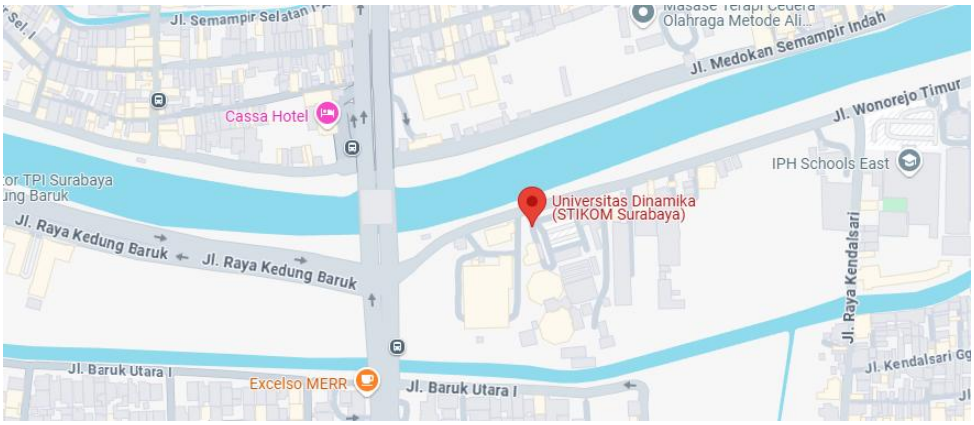
Gambar 2.2 Ruangan Lab IoT Universitas Dinamika
(Sumber: https://labfti.dinamika.ac.id/laboratorium-internet-of-things/)

Laboratorium ini dikoordinasikan oleh Charisma Dimas Affandi, S.T., yang juga berperan sebagai instruktur praktikum dan koordinator beberapa mata kuliah praktikum yang berkaitan dengan jaringan komputer dan infrastruktur IoT. Selain implementasi dan konfigurasi *server* IoT menggunakan Mosquitto untuk sistem komunikasi MQTT, di bawah arahannya, laboratorium ini juga mengerjakan berbagai proyek seperti pengembangan modul pembelajaran interaktif untuk pembelajaran gerbang logika menggunakan rangkaian IC digital dan pengembangan robot *arm* berbasis *image processing* untuk pemrosesan *gesture* tangan secara *real-time*. Laboratorium IoT juga terlibat dalam kegiatan pelatihan dan pengabdian masyarakat. Salah satu contohnya adalah pelatihan kepada para guru TIK di wilayah Sidoarjo mengenai metode pengajaran berbasis teknologi Arduino. Melalui pelatihan ini, para guru mendapatkan pengalaman langsung dalam menggunakan perangkat keras IoT yang dapat menunjang kemampuan mengajarnya.

## 2.2 Profil Perusahaan

Nama Instansi : Laboratorium FTI Universitas Dinamika

Alamat : Jl. Raya Kedung Baruk No. 98, Kedung Baruk, Kec. Rungkut, Surabaya, Jawa Timur 60298




Gambar 2.3 Lokasi Lab FTI Universitas Dinamika

(Sumber: https://maps.google.com/)

No. Telp : (031) 8721731

Website : https://www.labfti.dinamika.ac.id/

Email : lab.fti@dinamika.ac.id

## 2.3 Visi dan Misi Perusahaan

Laboratorium FTI Universitas Dinamika berada di bawah naungan Fakultas Teknologi dan Informatika Universitas Dinamika, sehingga visi dan misi mereka sama, yaitu:

**Visi** Fakultas Teknologi dan Informatika

"Menjadi Fakultas Teknologi dan Informatika berskala global yang produktif dalam berinovasi dan berjiwa *entrepreneur*"

**Misi** Fakultas Teknologi dan Informatika

A. Menyelenggarakan pendidikan yang berkualitas unggul, berkarakter, dan berjiwa *technopreneur*.

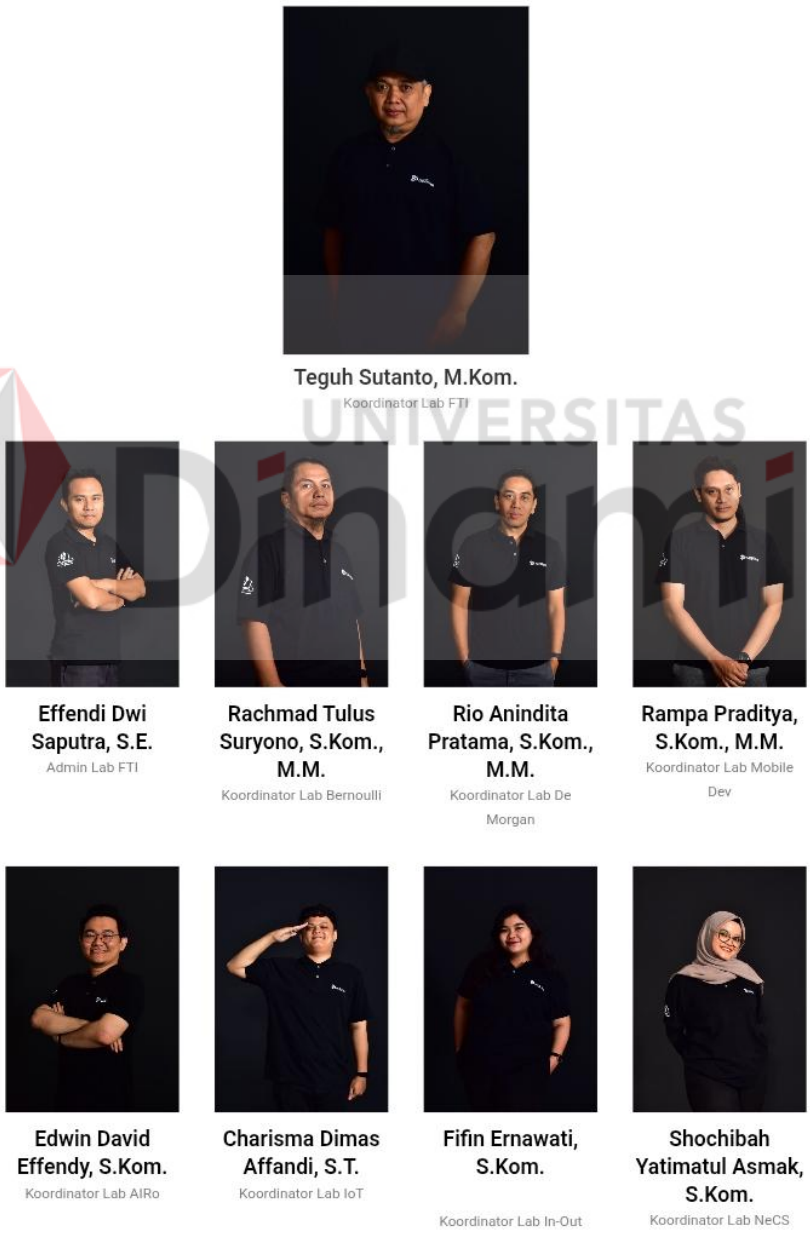
B. Melakukan penelitian dan pengembangan teknologi informatika yang inovatif dan solutif.

C. Melakukan pengabdian untuk menyebarluaskan hasil inovasi teknologi dan informatika bagi kesejahteraan masyarakat.

D. Melaksanakan kemitraan berskala global.

## 2.4 Struktur Organisasi

Struktur organisasi Laboratorium FTI Universitas Dinamika terlihat pada Gambar 2.4.




Gambar 2.4 Struktur Organisasi Perusahaan

(Sumber: https://labfti.dinamika.ac.id/)

# BAB III
# LANDASAN TEORI

## 3.1 *Internet of Things*

Internet of Things (IoT) adalah jaringan yang menghubungkan berbagai perangkat fisik, mulai dari sensor kecil hingga mesin industri besar, agar dapat bertukar data dan berkomunikasi melalui internet. Konsep dasarnya didukung oleh penggunaan sensor yang tersebar luas, konektivitas yang stabil, dan pemrosesan data yang cerdas.

Penelitian awal tentang IoT telah mengulas berbagai aspek penting, seperti struktur sistem, cara perangkat saling berkomunikasi, serta berbagai aplikasi yang dapat digunakan. Dari penelitian ini, para ahli berhasil merancang kerangka yang menjelaskan peluang dan tantangan dalam pengembangannya. Misalnya, ada model yang menggambarkan lapisan-lapisan utama dalam IoT, mulai dari bagian yang menangkap informasi dari lingkungan menggunakan sensor, hingga jaringan yang menghubungkan perangkat dan mengirimkan data melalui internet, serta aplikasi yang mengolah data agar dapat digunakan untuk berbagai keperluan (Choudhary, 2024; Nord et al., 2019).

Karena IoT melibatkan berbagai disiplin ilmu, para peneliti menggabungkan konsep dari bidang komunikasi, ilmu komputer, dan rekayasa sistem. Banyak studi menekankan pentingnya standar terbuka dan teknologi perantara agar perangkat dari berbagai merek dapat saling berkomunikasi tanpa kendala. Tren terbaru juga menunjukkan bahwa sistem jaringan yang bisa beradaptasi serta otomatis dalam mengambil keputusan sangat diperlukan agar IoT dapat berjalan secara efisien dan tahan terhadap berbagai tantangan.

## 3.2 *Message Queuing Telemetry Transport*

MQTT adalah protokol komunikasi yang ringan dengan metode *publish/subscribe*, awalnya dibuat untuk perangkat dengan koneksi internet yang terbatas dan sumber daya minimal. Konsep dasarnya adalah pemisahan antara

pengirim dan penerima data, sehingga memungkinkan komunikasi yang fleksibel, *scalable*, dan tidak bergantung pada koneksi langsung.

Protokol ini memiliki tiga tingkat *Quality of Service* (QoS) yang memastikan keandalan pengiriman pesan. Tingkatan ini mencakup opsi dari yang paling sederhana, di mana pesan dikirim tanpa konfirmasi (*fire-and-forget*), hingga tingkat tertinggi yang menjamin bahwa pesan hanya dikirim dan diterima satu kali secara akurat (*exactly-once delivery*), sesuai dengan kebutuhan aplikasi IoT.

Penelitian telah menunjukkan bahwa MQTT, dengan konsumsi sumber daya yang rendah, dapat meningkatkan komunikasi dalam sistem yang terbatas. Selain itu, fitur seperti penyimpanan pesan (*retained messages*) dan sesi yang bertahan memungkinkan aliran data secara real-time menjadi lebih stabil dan tangguh (Kaganurmath et al., 2024; Tuyishime et al., 2025).

### 3.3 Eclipse Mosquitto

Eclipse Mosquitto adalah broker MQTT *open-source* yang digunakan untuk menghubungkan klien dan perangkat IoT dalam komunikasi data. Mosquitto dirancang dengan fokus pada efisiensi dan kompatibilitas yang luas, sehingga bisa berjalan di berbagai platform tanpa membebani sumber daya perangkat.

Gambar 3.1 Logo Eclipse Mosquitto
(Sumber: https://mosquitto.org/)

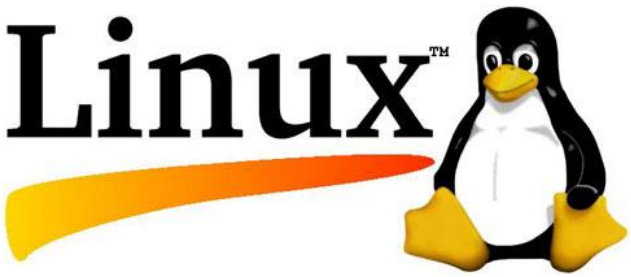
Mosquitto mendukung beberapa versi protokol MQTT, termasuk 5.0 dan 3.1.1, yang memungkinkan penggunaan fitur canggih sambil tetap menjaga kompatibilitas dengan perangkat lama yang memiliki keterbatasan sumber daya. Penelitian tentang performa Mosquitto telah menunjukkan bahwa, meskipun menggunakan sedikit daya komputasi, broker ini mampu menangani ribuan koneksi

klien secara bersamaan, menjadikannya pilihan yang tepat untuk implementasi IoT berskala besar.

Dalam pengembangannya, Mosquitto didesain dengan prinsip kesederhanaan, fleksibilitas, dan keamanan. Salah satu fitur penting yang dimilikinya adalah dukungan untuk protokol TLS, yang memastikan komunikasi data tetap terenkripsi dan aman. Selain itu, karena bersifat *open-source*, Mosquitto memungkinkan komunitas pengembang untuk terus menyempurnakan fungsinya dan berkontribusi dalam proyek-proyek IoT yang membutuhkan sistem komunikasi terdesentralisasi. Keberadaannya yang semakin luas dalam berbagai sektor membuktikan bahwa Mosquitto adalah salah satu broker MQTT yang berperan penting dalam perkembangan ekosistem IoT modern.

### 3.4 Linux

Linux menjadi sistem operasi utama yang mendukung berbagai perangkat IoT, *server*, dan infrastruktur *cloud* karena sifatnya yang *open-source*, tangguh, dan dapat berkembang sesuai kebutuhan. Konsep dasarnya didasari pada model pengembangan yang bersifat kolaboratif dan desain modular, memungkinkan Linux untuk beradaptasi dengan berbagai jenis perangkat, mulai dari sistem tertanam hingga aplikasi skala besar di perusahaan.




Gambar 3.2 Logo Linux

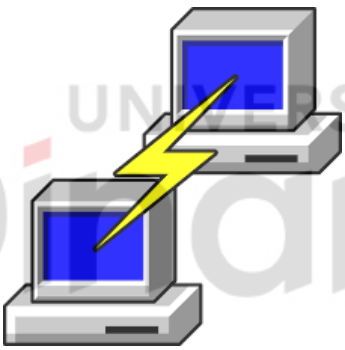
(Sumber: https://www.linux.org/threads/introduction-to-linux.4105/)

Linux terus berkembang berkat dukungan dari komunitas dan organisasi seperti Linux Foundation, yang berkontribusi dalam berbagai proyek *open-source* dan pendidikan teknologi. Upaya ini bertujuan untuk meningkatkan keamanan siber

dan memastikan berbagai perangkat serta aplikasi dapat bekerja sama tanpa hambatan. Seiring dengan pesatnya pertumbuhan IoT, fleksibilitas Linux dalam mendukung berbagai arsitektur tetap menjadi faktor penting dalam menciptakan sistem yang cerdas dan saling terhubung secara efisien. Perannya dalam dunia teknologi terus menjadi fondasi bagi inovasi di berbagai sektor.

### 3.5 PuTTY

PuTTY adalah aplikasi terminal emulator dan klien SSH/Telnet yang banyak digunakan untuk mengakses sistem secara jarak jauh dengan aman. Perannya sangat penting dalam administrasi sistem berbasis Unix dan dalam konfigurasi perangkat jaringan, karena memungkinkan pengguna untuk berinteraksi dengan *server* melalui komunikasi berbasis teks.



Gambar 3.3 Logo PuTTY

(Sumber: https://en.wikipedia.org/wiki/PuTTY)

Meskipun telah ada sejak beberapa dekade lalu, PuTTY tetap relevan hingga saat ini, terutama dalam ekosistem IoT dan komputasi awan. Keunggulannya terletak pada antarmuka yang sederhana namun kuat, sehingga ideal digunakan pada perangkat dengan sumber daya terbatas yang membutuhkan koneksi aman tanpa beban berlebih. Dalam lingkungan perusahaan, PuTTY tetap menjadi pilihan utama untuk manajemen sistem yang membutuhkan kepatuhan terhadap standar keamanan serta pengelolaan siklus hidup infrastruktur TI yang andal.

# BAB IV
# DESKRIPSI PEKERJAAN

## 4.1 Uraian Umum Pekerjaan

Kerja praktik ini merupakan bagian dari kewajiban akademik bagi mahasiswa Program Studi S1 Teknik Komputer di Universitas Dinamika. Kegiatan ini dilaksanakan secara individu, dengan bimbingan dari dosen pembimbing akademik serta pembimbing lapangan dari Laboratorium FTI Universitas Dinamika. Topik yang diangkat dalam kerja praktik ini adalah "Implementasi dan Konfigurasi Server IoT dengan Mosquitto untuk Sistem Komunikasi MQTT di Laboratorium FTI Universitas Dinamika."

Tujuan dari kerja praktik ini adalah untuk mengonfigurasi server komunikasi IoT yang memanfaatkan protokol MQTT dengan broker Mosquitto. Sistem tersebut dirancang agar dapat digunakan sebagai media komunikasi antarperangkat berbasis IoT, seperti ESP8266, ESP32, PC/laptop, dan *smartphone*. Selain itu, panduan teknis juga disusun untuk memudahkan pengguna dalam mengakses *server*, serta sebuah *dashboard* monitoring sederhana dibuat guna menampilkan statistik topik, jumlah koneksi, dan aktivitas yang terjadi pada broker.

## 4.2 Rancangan Pekerjaan

Untuk mendukung kebutuhan pengujian dan pengembangan sistem IoT, Laboratorium FTI Universitas Dinamika telah menyiapkan *server* MQTT dan merencanakan pembuatan infrastruktur *server* MQTT yang dapat diakses melalui berbagai jenis perangkat, seperti *smartphone*, mikrokontroler, serta komputer atau laptop.

Agar *server* dapat digunakan pada perangkat *smartphone*, aplikasi IoT MQTT Panel harus diunduh dan diinstal terlebih dahulu. Setelah aplikasi dijalankan, koneksi baru perlu dibuat dengan memasukkan konfigurasi sebagai berikut:

A. Alamat host yang digunakan adalah mqtt.dinamika.ac.id,

B. *Port* yang digunakan dapat disesuaikan, yaitu port 1883 untuk koneksi tanpa enkripsi atau port 8883 untuk koneksi dengan enkripsi SSL. Apabila koneksi dilakukan melalui *port* 8883, maka sertifikat harus disertakan. Sertifikat ini akan tersedia melalui tautan yang tertera dalam buku panduan pengguna.

Setelah itu, kredensial *login* perlu dimasukkan dengan memilih salah satu kombinasi *username* dan *password* berikut: mhs dengan *password* mahasiswa, dinamika dengan *password* semangat, atau riset dengan *password* maju. *Username* dan *password* yang telah disebutkan merupakan kombinasi yang telah direncanakan untuk publik di lingkup laboratorium FTI Universitas Dinamika. Jika ingin menambahkan *username* dan *password* lain, maka dapat menghubungi koordinator laboratorium bagian IoT secara langsung atau melalui pengisian Google Form yang ada di buku panduan pengguna. Setelah semua pengaturan disimpan, koneksi dapat diuji. Jika koneksi berhasil, panel dapat ditambahkan untuk melakukan proses *subscribe* atau *publish* pesan ke suatu topik. Format penulisan topik yang digunakan mengikuti pola username/topik.

Untuk penggunaan pada perangkat mikrokontroler, aplikasi Arduino IDE harus telah tersedia pada komputer pengguna. Setelah itu, pustaka (*library*) bernama PubSubClient perlu diunduh melalui fitur Library Manager yang tersedia di dalam Arduino IDE. Kode program akan disediakan melalui tautan yang tercantum pada buku panduan pengguna, dan kode tersebut harus dimasukkan ke dalam Arduino IDE serta disesuaikan dengan beberapa informasi penting. Informasi yang perlu disesuaikan meliputi SSID dan kata sandi jaringan Wi-Fi, alamat *server* MQTT dan *port* koneksi, serta *username* dan *password* sesuai dengan yang telah ditentukan sebelumnya. Setelah semua penyesuaian dilakukan, kode dapat diunggah ke mikrokontroler. Status koneksi dapat dipantau melalui fitur Serial Monitor untuk memastikan bahwa koneksi telah berhasil dilakukan.

Untuk penggunaan pada perangkat komputer atau laptop, langkah yang diambil bergantung pada sistem operasi yang digunakan. Bagi pengguna sistem operasi Windows, broker MQTT perlu diunduh terlebih dahulu melalui situs resmi Mosquitto. Setelah proses instalasi selesai, koneksi ke broker dapat dilakukan melalui Command Prompt. Sementara itu, bagi pengguna sistem operasi Linux, klien MQTT dapat diinstal melalui terminal dengan menggunakan perintah sudo

apt install mosquitto-clients. Setelah proses instalasi selesai, komunikasi dengan broker dapat dilakukan melalui terminal.

Untuk melakukan *subscribe* ke suatu topik pada *port* 1883, perintah yang digunakan adalah **mosquitto_sub -h mqtt.dinamika.ac.id -u namauser -P passworduser -t "namauser/topik"**. Jika menggunakan *port* 8883 dengan koneksi SSL, maka perintahnya adalah **mosquitto_sub -h mqtt.dinamika.ac.id -u namauser -P passworduser -t "namauser/topik" -p 8883 --cafile "lokasifile"**.

Adapun untuk mengirimkan (*publish*) pesan ke suatu topik melalui *port* 1883, perintah yang digunakan adalah **mosquitto_pub -h mqtt.dinamika.ac.id -u namauser -P passworduser -t "namauser/topik" -m "pesan"**. Jika menggunakan *port* 8883 dengan SSL, maka perintahnya menjadi **mosquitto_pub -h mqtt.dinamika.ac.id -u namauser -P passworduser -t "namauser/topik" -m "pesan" -p 8883 --cafile "lokasifile"**.

Sebagai penutup, semua parameter yang digunakan dalam konfigurasi harus disesuaikan dengan informasi yang telah dijelaskan sebelumnya. Keberhasilan dalam proses pengiriman dan penerimaan pesan sangat bergantung pada ketepatan konfigurasi *server*, kredensial yang digunakan, serta format penulisan topik yang sesuai dengan aturan Access Control List (ACL). Oleh karena itu, penting untuk memastikan bahwa setiap tahapan telah dilakukan dengan benar agar komunikasi data melalui broker MQTT dapat berjalan secara lancar dan aman.

### 4.3 Alur Penggunaan

Proses komunikasi melalui broker MQTT diawali dengan pengecekan apakah pengguna telah memiliki akun. Jika belum memiliki akun, pengguna diarahkan untuk menghubungi koordinator laboratorium bagian IoT atau mengunduh panduan pengguna melalui situs web labfti.dinamika.ac.id.

Jika pengguna telah memiliki akun, maka proses koneksi dapat dimulai dengan memilih metode koneksi melalui *port* 1883 atau 8883. *Port* 1883 digunakan untuk koneksi tanpa enkripsi, sedangkan *port* 8883 digunakan untuk koneksi yang menggunakan SSL. Apabila koneksi menggunakan *port* 8883, maka sertifikat harus dimasukkan. Sertifikat tersebut dapat diperoleh melalui tautan yang tertera pada buku panduan pengguna.

Setelah itu, pengguna diminta untuk memasukkan *username* dan *password* sebagai bagian dari proses autentikasi ke broker. Kredensial tersebut juga dapat diperoleh melalui tautan yang tertera pada buku panduan pengguna.

Selanjutnya, akan dilakukan pengecekan apakah koneksi berhasil dilakukan. Jika koneksi tidak berhasil, maka pengguna perlu memeriksa kembali metode koneksi serta kredensial yang digunakan. Jika koneksi berhasil, pengguna dapat melanjutkan dengan memilih peran (*role*) sebagai *publisher* atau *subscriber* terhadap suatu topik tertentu.

Setelah peran dipilih, sistem akan memeriksa apakah pengiriman pesan telah sesuai dengan format ACL (Access Control List). Format ACL yang digunakan adalah username/topik. Jika format sudah sesuai, proses pengiriman atau penerimaan pesan akan dilakukan.

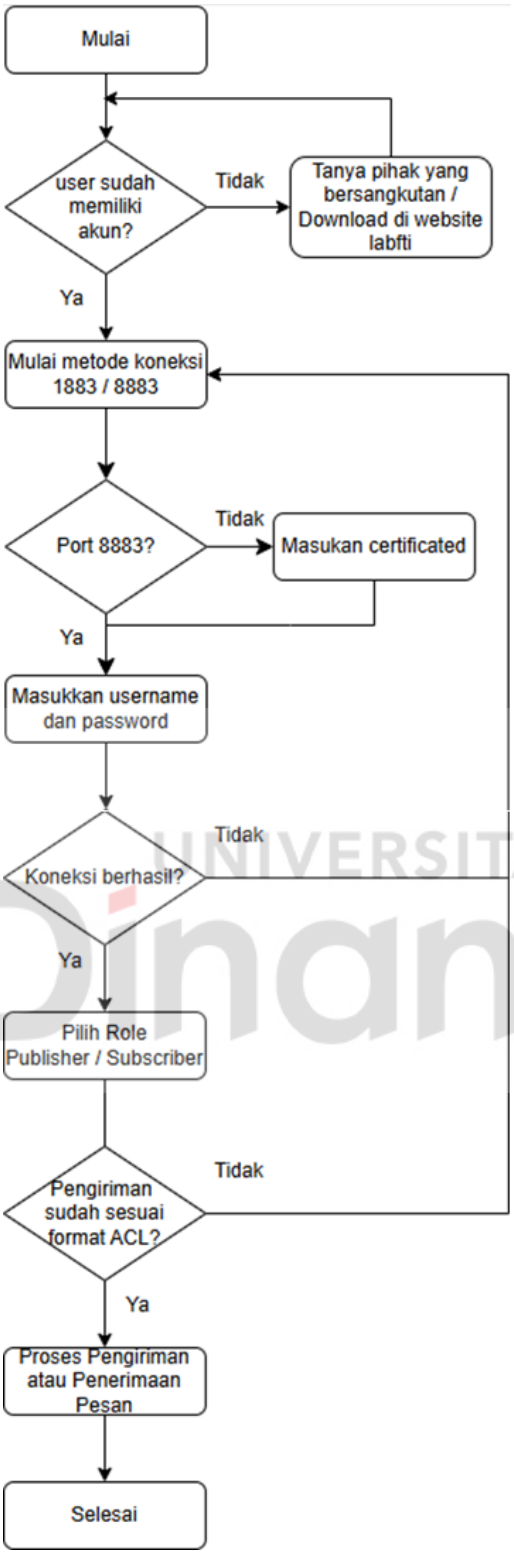
Keberhasilan pengiriman dan penerimaan pesan juga bergantung pada pengaturan nilai QoS (Quality of Service) yang digunakan, yaitu sebagai berikut:

A. QoS 0: Pesan dikirim tanpa memerlukan konfirmasi dari penerima. Pesan dapat saja tidak diterima tanpa adanya upaya pengiriman ulang.

B. QoS 1: Pesan yang dikirim akan diterima oleh *subscriber*, namun dimungkinkan terjadi penerimaan pesan lebih dari satu kali (duplikasi).

C. QoS 2: Pesan akan diterima oleh *subscriber* tanpa adanya kemungkinan duplikasi pesan.

Setelah pesan berhasil dikirim atau diterima sesuai ketentuan tersebut, maka proses komunikasi dinyatakan selesai. *Flowchart* untuk alur penggunaan dapat dilihat melalui Gambar 4.1.




Gambar 4.1 *Flowchart* Alur Penggunaan

## 4.4 Jadwal Pengerjaan

Jadwal pengerjaan kerja praktik disusun sebagai gambaran urutan dan rentang waktu setiap tugas yang dikerjakan selama masa kerja praktik berlangsung. Jadwal ini bertujuan untuk memberikan gambaran mengenai pelaksanaan kegiatan yang difokuskan pada proses implementasi, konfigurasi, serta pengujian sistem sesuai dengan tujuan kerja praktik. Perlu disampaikan bahwa jadwal yang disajikan hanya mencakup aktivitas teknis, tanpa memasukkan proses penulisan laporan maupun dokumen pendukung lainnya. Tabel dari *timeline* pengerjaan kerja praktik dapat dilihat pada Tabel 4.1 sebagai berikut:

Tabel 4.1 Jadwal Pengerjaan Kerja Praktik

| Aktivitas | Waktu Pengerjaan | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Februari | | | | Maret | | | |
| | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| Konfigurasi *User* | | | | | | | | |
| Konfigurasi ACL | | | | | | | | |
| Konfigurasi *Port* SSL | | | | | | | | |
| Pembuatan *User Guide* | | | | | | | | |

## 4.5 Pengerjaan Kerja Praktik

### 4.5.1 Instalasi dan Akses *Server*

Aplikasi PuTTY digunakan untuk melakukan akses jarak jauh (*remote*) ke *server* MQTT milik Laboratorium FTI Universitas Dinamika yang berbasis sistem operasi Linux. *Server* tersebut telah dilengkapi dengan Mosquitto sebagai MQTT Broker. Akses dilakukan melalui protokol SSH dengan menggunakan alamat IP dan *port* yang telah disediakan oleh pihak Laboratorium FTI Universitas Dinamika.

### 4.5.2 Membuat *Username* dan *Password*

Untuk membuat awal *username* dan *password*, maka menggunakan format perintah sebagai berikut:

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd username
```

Jika ingin menambahkan *username* dan *password* lagi, maka jalankan perintah tadi tanpa tanda -c seperti ini:

```
sudo mosquitto_passwd /etc/mosquitto/passwd usernamelain
```

Jika opsi -c digunakan dua kali maka akan membuat *user* di Linux dapat menyebabkan reset.

Setelah itu *restart* Mosquitto dan pastikan *server* pada mosquitto berjalan tanpa adanya *error* dengan menggunakan perintah di bawah ini:

```
sudo systemctl restart mosquitto
sudo systemctl status mosquitto
```

Apabila terjadi *error* hak akses pada file *password* Mosquitto (passwd), maka jalankan perintah berikut untuk memberikan hak akses yang sesuai:

```
sudo chmod 600 /etc/mosquitto/passwd
sudo chown mosquitto:mosquitto /etc/mosquitto/passwd
```

Setelah itu coba *restart* dan cek status kembali.

### 4.5.3 Mengatur File ACL

File ACL digunakan untuk menentukan hak akses tiap pengguna terhadap topik tertentu. Konfigurasi ini dilakukan pada file /etc/mosquitto/aclfile. Di dalam file tersebut ditambahkan baris:

```
pattern readwrite %u/#
```

Artinya, setiap pengguna hanya diberikan izin *read* (membaca) dan *write* (menulis) pada topik yang diawali dengan nama pengguna tersebut. Contoh: pengguna dengan *username* mhs hanya dapat mengakses topik mhs/#. Tanda "#" merujuk pada nama dari topik yang diinginkan, dapat diisi dengan bebas.

### 4.5.4 Mengatur File Konfigurasi

Untuk mengatur konfigurasi Mosquitto dilakukan pada file /etc/mosquitto/conf.d/default.conf yang akan dipanggil dengan include_dir /etc/mosquitto/conf.d pada file utama, yaitu /etc/mosquitto/mosquitto.conf. Hal ini dilakukan karena dapat memberikan struktur yang lebih rapi, fleksibel, dan mudah untuk dikelola dibandingkan menuliskan semuanya langsung di dalam file utama. Dengan cara ini, konfigurasi Mosquitto dapat dipisah-pisahkan sesuai fungsinya sehingga memudahkan untuk pemeliharaan, pembaruan, dan pemahaman konfigurasi.

Di dalam file ini terdapat pengaturan untuk menerima koneksi dari dua jalur, yaitu jalur tanpa enkripsi (*port* 1883) dan jalur dengan enkripsi SSL/TLS (*port* 8883). Berikut baris-baris yang dimasukkan ke dalam file ini:

```
per_listener_settings true
```

Baris ini mengaktifkan pengaturan agar setiap *listener* (*port*) dapat memiliki konfigurasi yang berbeda secara independen.

Bagian ini merupakan pengaturan untuk *listener* pertama, yaitu *port* 1883 tanpa menggunakan enkripsi. *listener* 1883 0.0.0.0 berfungsi ketika *server* akan menerima koneksi dari perangkat melalui *port* 1883 dan angka 0.0.0.0 berarti *server* menerima dari semua alamat IP.

```
listener 1883 0.0.0.0
```

Bagian ini merupakan pengaturan untuk *listener* kedua, yaitu *port* 8883 dengan menggunakan enkripsi. *Listener* ini lebih aman karena adanya enkripsi sehingga data yang dikirim dari dan ke perangkat tidak bisa dibaca oleh pihak lain. Sama seperti *listener* pertama yang mana listener 8883 0.0.0.0 berfungsi ketika *server* akan menerima koneksi terenkripsi melalui *port* 8883 dan *server* dapat menerima dari semua alamat IP. cafile, certfile, dan keyfile masing-masing menunjuk ke file sertifikat yang digunakan untuk mengamankan komunikasi. Sertifikat ini berfungsi seperti identitas resmi agar perangkat yakin bahwa mereka benar-benar terhubung ke *server* yang sah. Untuk dhparamfile menunjuk ke file yang berisi parameter Diffie-Hellman (DH), yaitu bagian dari sistem keamanan yang digunakan saat membuat koneksi terenkripsi. Dengan file ini, proses pertukaran kunci rahasia antar *server* dan klien tidak mudah disadap, walaupun ada yang mencoba untuk mengintip komunikasi.

```
listener 8883 0.0.0.0
cafile /etc/ssl/certs/ISRG_Root_X1.pem
certfile /etc/mosquitto/certs/server.pem
keyfile /etc/mosquitto/certs/server.key
dhparamfile /etc/ssl/certs/dhparam.pem
```

Pada masing-masing *listener* juga ditambahkan pengaturan ini, di mana protocol mqtt fungsinya adalah untuk menentukan bahwa protokol yang digunakan adalah MQTT. allow_anonymous false untuk melarang koneksi tanpa autentikasi

(anonymous), sehingga semua klien diharuskan untuk *login* terlebih dahulu. password_file menunjukkan file yang berisi daftar *username* dan *password* yang diizinkan untuk *login*. Dan, acl_file untuk menentukan file Access Control List (ACL) yang mengatur hak akses klien terhadap topik-topik tertentu.

```
protocol mqtt
allow_anonymous false
password_file /etc/mosquitto/passwd
acl_file /etc/mosquitto/aclfile
```

### 4.5.5 Mengatur File Utama Mosquitto

Agar kinerja *server* bekerja secara maksimal maka diperlukan parameter penting dalam pengoperasian *server* MQTT pada file konfigurasi utama broker MQTT Mosquitto.

Konfigurasi ini mengaktifkan fitur penyimpanan data (persistence true) sehingga Mosquitto dapat menyimpan data *session* dan pesan yang belum terkirim secara permanen. Data tersebut disimpan di direktori /var/lib/mosquitto/ dan akan otomatis disimpan ulang setiap 300 detik (5 menit) untuk mencegah kehilangan data saat broker dimatikan secara tiba-tiba.

```
persistence true
persistence_location /var/lib/mosquitto/
autosave_interval 300
```

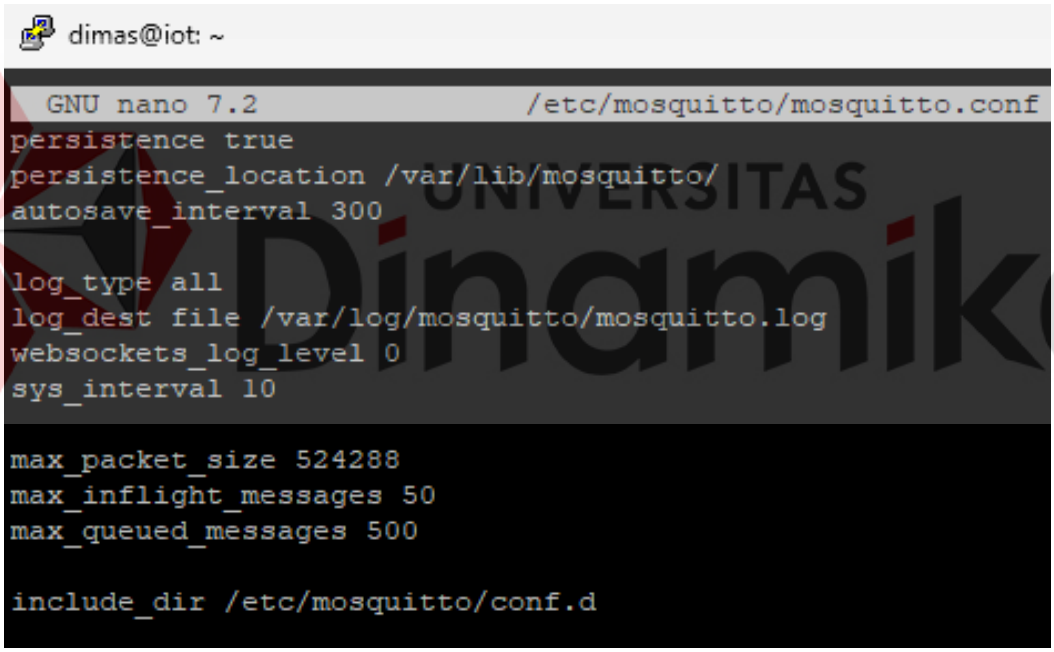
Untuk bagian ini bertujuan untuk mengatur log aktivitas Mosquitto. Semua jenis log akan dicatat (**log_type all**) dan disimpan ke dalam file /var/log/mosquitto/mosquitto.log. Nilai **websockets_log_level 0** berarti log untuk koneksi WebSocket tidak terlalu detail (tingkat minimum). **sys_interval 10** menunjukkan bahwa informasi sistem seperti jumlah koneksi aktif akan diperbarui setiap 10 detik.

```
log_type all
log_dest file /var/log/mosquitto/mosquitto.log
websockets_log_level 0
sys_interval 10
```

Pengaturan ini akan mengelola batas dan kinerja komunikasi, di mana **max_packet_size 52488** adalah untuk membatasi ukuran maksimum pesan MQTT menjadi 512 KB. **max_inflight_messages 50** akan membatasi jumlah pesan QoS 1 dan QoS 2 yang dapat dikirim tetapi belum dikonfirmasi dalam satu waktu. Dan **max_queued_messages 500** untuk menetapkan batas jumlah pesan yang dapat diantrekan untuk klien yang sedang offline.

```
max_packet_size 524288
max_inflight_messages 50
max_queued_messages 500
```

Berikut ini merupakan tampilan dari file /etc/mosquitto/mosquitto.conf dari aplikasi PuTTY:




Gambar 4.2 Tampilan File Utama Mosquitto

Untuk pengaturan ini berfungsi untuk memberitahu Mosquitto untuk membuat konfigurasi tambahan dari direktori /etc/mosquitto/conf.d. Ini berguna untuk menyusun konfigurasi menjadi beberapa file yang lebih terorganisir.

```
include_dir /etc/mosquitto/conf.d
```

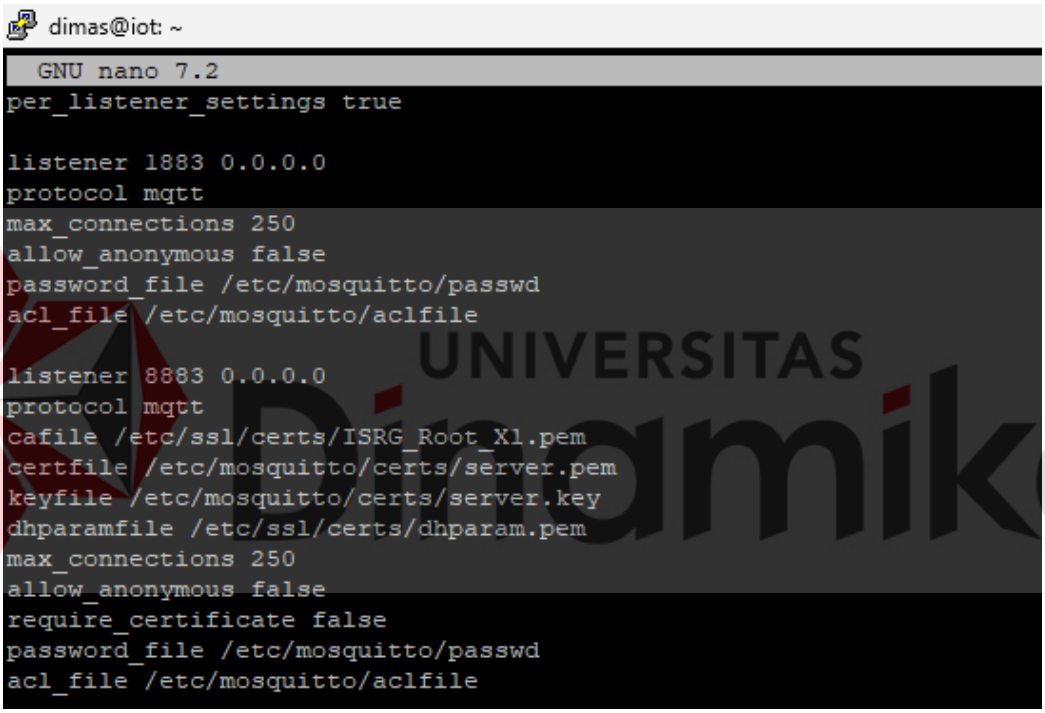
### 4.5.6 Tambahan: Konfigurasi Batas Koneksi

Pada masing-masing pengaturan *listener* juga ditambahkan konfigurasi tambahan untuk memberikan batas konfigurasi, yaitu:

```
max_connections 250
```

Digunakan untuk membatasi jumlah maksimal koneksi klien ke *server*, sebanyak 250 koneksi pada setiap *port*. Hal ini bertujuan untuk mencegah overload pada *server*.

Berikut ini merupakan tampilan dari file /etc/mosquitto/conf.d/default.conf pada aplikasi PuTTY:




Gambar 4.3 Tampilan File Konfigurasi Mosquitto

### 4.5.7 Tambahan: Memblokir Kata Terlarang

Pada bagian file ACL, yaitu /etc/mosquitto/aclfile ditambahkan baris yang digunakan untuk melarang akses (baik membaca maupun mengirim pesan) pada topik tertentu yang menggunakan kata-kata sensitif. Kata terlarang disini berarti adalah nama topik yang sengaja untuk dibatasi. Bagian ini dapat ditambahkan secara pribadi pada file ACL dengan mengganti kata terlarang tersebut menjadi kata-kata lainnya yang ingin dibatasi. Berikut baris yang dimaksud:

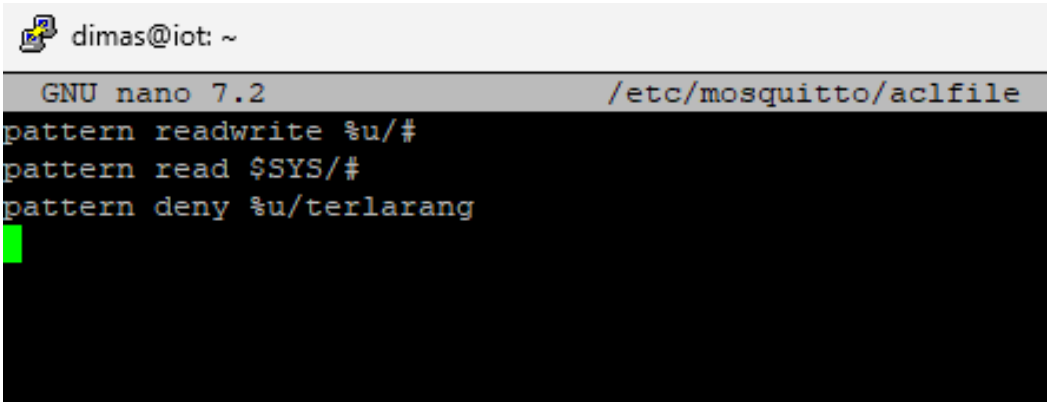
```
pattern deny %u/terlarang
```

### 4.5.8 Tambahan: Pembuatan *Dashboard Server*.

Untuk memunculkan data *server* ke dalam *dashboard* diperlukan izin akses baca (*read-only*) kepada seluruh pengguna terhadap topik sistem **$SYS/** pada broker Mosquitto. Topik ini adalah topik khusus yang secara otomatis disediakan oleh broker untuk menampilkan informasi internal seperti jumlah klien yang terhubung, total pesan yang dikirim dan diterima, serta lama waktu broker aktif (uptime). Sistem ini bisa ditambahkan pada file ACL di /etc/mosquitto/aclfile dengan baris:

```
pattern read $SYS/#
```

Berikut ini merupakan tampilan dari file /etc/mosquitto/aclfile dari aplikasi PuTTY:




Gambar 4.4 Tampilan File ACL Mosquitto

*Dashboard* monitoring disajikan dalam bentuk halaman web dan dibuat menggunakan kombinasi HTML, CSS, JavaScript, serta pustaka eksternal seperti Bootstrap, jQuery, D3.js, dan Paho MQTT untuk membentuk tampilan interface. Tujuannya adalah untuk menampilkan statistik operasional broker, termasuk jumlah pesan yang dikirim dan diterima, status koneksi klien, jumlah *subscriber*, dan lainnya.

Kode ini diawali dengan struktur dasar HTML yang bertindak sebagai fondasi tampilan *dashboard*. Di dalam bagian <head>, terdapat meta tag yang mengatur kompatibilitas dan responsivitas halaman, serta pemanggilan file CSS untuk styling. Berikut adalah bagian kode yang memuat tampilan *dashboard*:

```
<div id="mainPage">
    <div class="container-fluid">
        <div class="row pt-0 pb-0">
            <div class="col-xl-3 col-sm-6 mb-0 pr-1">
                <img src="./mosquitto.png" height="30">
            </div>
            <div class="col-xl-3 col-sm-6 mb-0 pr-1">
                Realtime Mosquitto Broker Dashboard
            </div>
        </div>
    </div>
</div>
```

Bagian ini menggunakan Bootstrap Grid System untuk menyusun *layout* yang responsif. Elemen gambar (img) berfungsi untuk menampilkan logo Mosquitto, sementara teks "Realtime Mosquitto Broker Dashboard" bertindak sebagai judul halaman.

Selanjutnya, bagian penting dari kode adalah widget yang digunakan untuk menampilkan data dari broker MQTT. Setiap widget dikaitkan dengan datastream MQTT menggunakan JavaScript. Contohnya, widget berikut digunakan untuk mengambil jumlah pesan yang diterima oleh broker:

```
CreateWidget({
    bindto: "widget1",
    datastream: "$SYS/broker/messages/received",
```

```
    type: "labeltext",
    height: 20
});
```

Kode ini menginstruksikan sistem untuk menghubungkan widget1 dengan topik sistem **$SYS/broker/messages/received**, yang akan menampilkan jumlah pesan yang diterima broker Mosquitto secara *real-time*.

Selain menampilkan statistik pesan, *dashboard* ini juga memiliki form *login* untuk mengautentikasi pengguna sebelum mereka dapat mengakses informasi lebih lanjut. Form *login* ini dibuat menggunakan HTML dan jQuery UI. Berikut adalah bagian kodenya:

```
<div id="dialog-form" title="User Authentication">
    <form>
        <fieldset>
            <label for="username">Username</label>
            <input type="text" name="username"
id="username">
            <label for="password">Password</label>
            <input type="password" name="password"
id="password">
            <input type="button" value="Login"
id="credentailsSubmit">
        </fieldset>
    </form>
</div>
```

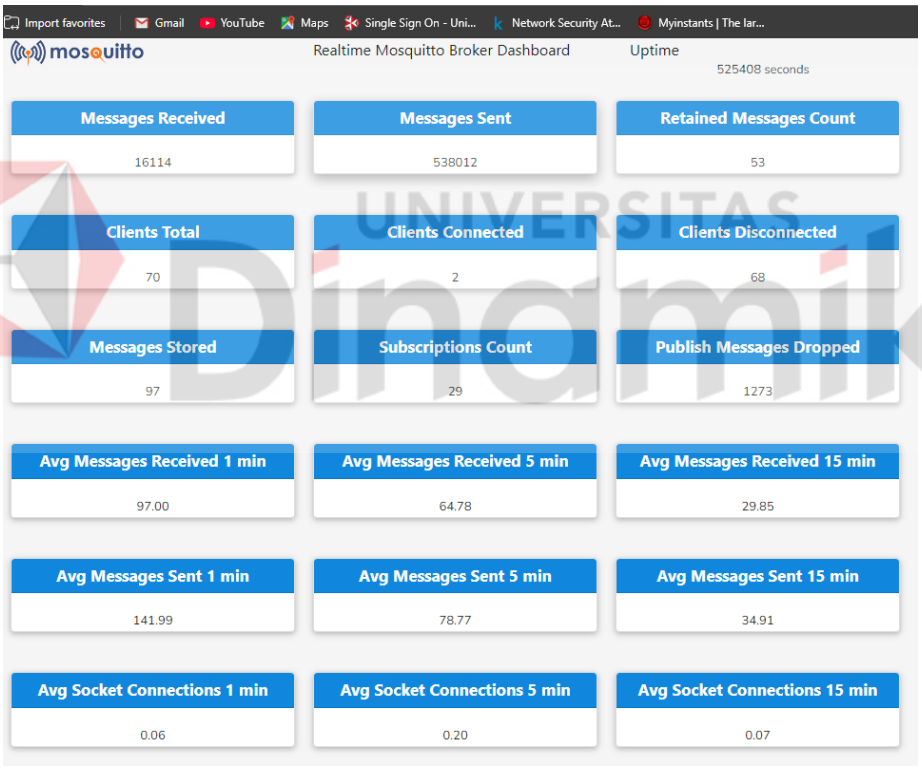
Kode ini membuat form input yang meminta pengguna untuk memasukkan *username* dan *password*. Saat pengguna menekan tombol "Login", fungsi JavaScript akan dipanggil untuk memverifikasi kredensial mereka sebelum mengizinkan akses ke *dashboard*.

Terakhir, kode ini juga memuat berbagai pustaka eksternal untuk mendukung fungsionalitasnya. Misalnya, pustaka jQuery digunakan untuk menangani efek tampilan, sedangkan Paho MQTT digunakan untuk mengambil data dari broker. Berikut bagian kode yang memuat pustaka-pustaka tersebut:

```
<script src="./vendor/jquery/js/jquery-
3.2.1.min.js"></script>
<script
src="./vendor/bootstrap/js/bootstrap.bundle.min.js"></scrip
t>
<script src="./vendor/d3/js/d3.v5.min.js"></script>
<script src="./vendor/paho/js/paho-mqtt-min.js"></script>
```

Kode ini memastikan bahwa semua pustaka yang diperlukan tersedia agar fitur interaktif seperti grafik, tampilan modal *login*, dan komunikasi MQTT dapat berfungsi dengan baik.

Berikut ini merupakan tampilan dari *Dashboard Server* yang dijalankan melalui website:




Gambar 4.5 Tampilan *Dashboard Server*

(Sumber: https://iot.dinamika.ac.id/)

Gambar tersebut memperlihatkan *interface* dari *dashboard server* MQTT yang dibangun menggunakan Mosquitto sebagai broker. *Dashboard* ini menyajikan data operasional *server secara real-time* dan dirancang untuk memberikan

informasi statistik terkait aktivitas komunikasi antara klien dan *server* MQTT. Berikut adalah penjabaran masing-masing komponen informasi yang ditampilkan:

A. **Messages Received**: menunjukkan jumlah total pesan yang telah diterima oleh broker dari seluruh klien.

B. **Messages Sent**: jumlah total pesan yang telah dikirim oleh broker kepada seluruh klien yang berlangganan topik terkait.

C. **Retained Messages Count**: menampilkan jumlah pesan yang ditandai sebagai *retained*, yaitu pesan yang disimpan oleh broker untuk disampaikan kepada klien baru yang berlangganan topik tertentu.

D. **Clients Total**: menunjukkan total jumlah klien yang pernah terhubung ke broker sejak *server* aktif.

E. **Clients Connected**: menampilkan jumlah klien yang sedang aktif atau terhubung saat ini.

F. **Clients Disconnected**: menunjukkan jumlah klien yang telah terputus dari broker.

G. **Messages Stored**: menginformasikan total pesan yang saat ini disimpan oleh broker, baik untuk keperluan *retained* maupun lainnya.

H. **Subscriptions Count**: jumlah langganan (*subscriptions*) aktif yang telah dilakukan oleh klien terhadap topik-topik yang tersedia di broker.

I. **Publish Messages Dropped**: menampilkan jumlah pesan publish yang gagal dikirim atau diabaikan oleh broker karena alasan tertentu, seperti kesalahan konfigurasi atau batasan tertentu.

J. **Average Messages Received** (1 *min*, 5 *min*, 15 *min*): menyajikan rata-rata jumlah pesan yang diterima oleh broker dalam jangka waktu 1 menit, 5 menit, dan 15 menit terakhir.

K. **Average Messages Sent** (1 *min*, 5 *min*, 15 *min*): menunjukkan rata-rata jumlah pesan yang dikirim oleh broker dalam periode waktu yang sama.

L. **Average Socket** Connections (1 min, 5 min, 15 min): menyajikan rata-rata jumlah koneksi socket aktif dalam waktu tertentu.

M. **Uptime**: menampilkan durasi waktu broker telah berjalan secara kontinu sejak terakhir kali diaktifkan.

### 4.5.9 Penyusunan Buku Panduan Pengguna

Buku panduan pengguna disusun secara bertahap dan sistematis guna memastikan seluruh konten yang disajikan dapat dipahami dengan mudah oleh pengguna lain. Proses penyusunan dimulai dengan melihat referensi serta penentuan cakupan materi yang dibahas, meliputi perangkat keras dan perangkat lunak yang akan digunakan dalam implementasi komunikasi MQTT di lingkungan Universitas Dinamika. Setelah ruang lingkup telah ditentukan, langkah selanjutnya ialah melakukan eksperimen dan dokumentasi langsung terhadap proses instalasi dan konfigurasi baik pada perangkat mikrokontroler, komputer pribadi, maupun smartphone, di mana setiap langkah pengujian yang dilakukan dicatat secara rinci.

Selanjutnya, seluruh hasil dokumentasi dikompilasi dan disusun menjadi sebuah naskah buku panduan dengan mengikuti struktur sistematis, dimulai dari pedoman penggunaan alat, tahapan instalasi perangkat lunak, hingga implementasi komunikasi antar perangkat. Penulisan dilakukan sebisa mungkin menggunakan gaya bahasa yang mudah dipahami pembaca dari berbagai latar belakang. Setiap bagian dilengkapi dengan ilustrasi serta tautan unduhan yang relevan untuk mendukung kenyamanan pengguna. Dengan demikian, buku ini disusun tidak hanya sebagai dokumentasi, tetapi juga sebagai media pembelajaran praktis yang dapat diandalkan oleh pengguna dalam mengakses dan mengimplementasikan *server* MQTT secara mandiri.

Berikut ini adalah tampilan dari *cover* buku panduan dan daftar isi dari buku panduan agar pembaca dapat bayangan isi dari buku panduan tersebut:


Gambar 4.6 *Cover* Buku Panduan Pengguna

**DAFTAR ISI**





Gambar 4.7 Daftar Isi Buku Panduan

# BAB V
# PENUTUP

## 5.1 Kesimpulan

Berdasarkan hasil kerja praktik yang telah dilaksanakan, maka dapat disimpulkan beberapa hal sebagai berikut:

1. Implementasi dan konfigurasi *server* MQTT menggunakan Mosquitto berhasil dilakukan dengan baik pada *server* Linux milik Laboratorium FTI Universitas Dinamika. *Server* mampu menerima koneksi dari berbagai perangkat melalui protokol MQTT, baik melalui koneksi standar (*port* 1883) maupun koneksi aman SSL/TLS (*port* 8883).
2. Pengaturan keamanan komunikasi MQTT telah diterapkan secara menyeluruh, meliputi konfigurasi *username* dan *password*, file Access Control List (ACL), serta sertifikat SSL. Hal ini mampu mencegah akses tidak sah dan memastikan komunikasi antarperangkat berlangsung aman dan terenkripsi.
3. *Dashboard* monitoring berbasis web berhasil dikembangkan untuk memantau status *server* secara *real-time*, seperti jumlah pesan yang dikirim/diterima dan jumlah klien yang terhubung, dengan memanfaatkan topik sistem $SYS pada Mosquitto dan pustaka visualisasi seperti D3.js dan Paho MQTT.
4. Panduan pengguna (*user guide*) telah disusun dengan lengkap, mencakup langkah-langkah penggunaan *server* MQTT pada tiga platform utama: smartphone (IoT MQTT Panel), mikrokontroler (ESP8266/ESP32 via Arduino IDE), dan PC (Command Prompt/Terminal).

## 5.2 Saran

Sebagai saran untuk pengembangan ke depan, sebaiknya *dashboard* monitoring diperluas fungsionalitasnya menjadi lebih interaktif, misalnya dengan menambahkan grafik historis, pencatatan log, atau notifikasi otomatis ketika terjadi anomali. Untuk buku panduan pengguna juga perlu dievaluasi dan diperbarui secara berkala agar tetap relevan dengan perkembangan teknologi dan kebutuhan pengguna.

# DAFTAR PUSTAKA


Abilovani, Z. B., Yahya, W., & Bakhtiar, F. A. (2018). Implementasi Protokol MQTT Untuk Sistem Monitoring Perangkat IoT. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *2*(12), 7521–7527. http://j-ptiik.ub.ac.id

Amrullah, A., Udin, M., al Rasyid, H., Winarno, I., & Udin, : M. (2022). Implementasi dan Analisis Protokol Komunikasi IoT untuk Crowdsensing pada Bidang Kesehatan. *JURNAL INOVTEK POLBENG - SERI INFORMATIKA*, *7*(1), 122–135. https://doi.org/https://doi.org/10.35314/isi.v7i1.2365

Choudhary, A. (2024). Internet of Things: a comprehensive overview, architectures, applications, simulation tools, challenges and future directions. In *Discover Internet of Things* (Vol. 4, Issue 1). Springer Nature. https://doi.org/10.1007/s43926-024-00084-3

Eka Sari, W., Triyono, A., Widiya Pratama, Y., Bagus, M., Tmur, B., & Negeri Samarinda, P. (2025). Rancang bangun Server Message Queuing Telemetry Transport ( MQTT) Internet of Things ( IoT) Berbasis Eclipse. *JSAI: Journal Scientific and Applied Informatics*, *8*(1), 9–22. https://doi.org/https://doi.org/10.36085/jsai.v8i1.7403

Harnanta, K. J., Bhawiyuga, A., & Basuki, A. (2020). Implementasi MQTT Broker dengan Kemampuan Auto Scaling pada Internet of Things. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *4*(6), 1783–1792. https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/7418

Kaganurmath, S., Cholli, N., & R, A. M. (2024). Performance Analysis of Dynamic Light Weight Cryptographic Algorithm for Message Queuing Telemetry Transport based IoT Communications. In *Journal of Information Systems Engineering and Management* (Vol. 2025, Issue 42s). https://www.jisem-journal.com/

Nord, J. H., Koohang, A., & Paliszkiewicz, J. (2019). The Internet of Things: Review and theoretical framework. In *Expert Systems with Applications*

(Vol. 133, pp. 97–108). Elsevier Ltd. https://doi.org/10.1016/j.eswa.2019.05.014

Tuyishime, E., Martalò, M., Cotfas, P. A., Popescu, V., Cotfas, D. T., & Rekeraho, A. (2025). Resource-Efficient Traffic Classification Using Feature Selection for Message Queuing Telemetry Transport-Internet of Things Network-Based Security Attacks. *Applied Sciences (Switzerland)*, *15*(8). https://doi.org/10.3390/app15084252