Implementasi Sistem Otomasi Gedung Berbasis IoT Menggunakan ESP8266 dan Node-RED

Keanrich Cordana[1]* **Laurentius Santiago[1]** **Matthew Lay[1]**

[1]*IT and Big Data Analytics, School of Science and Engineering, Calvin Institute of Technology, Indonesia*
* Corresponding author's Email: cordana2006@gmail.com

Abstract: Perkembangan teknologi Internet of Things (IoT) mendorong penerapan sistem otomasi gedung yang mampu meningkatkan efisiensi energi dan kenyamanan pengguna. Penelitian ini membahas perancangan dan implementasi sistem otomasi gedung berbasis IoT menggunakan NodeMCU ESP8266 sebagai pengendali utama dan Node-RED sebagai platform pemantauan serta kendali. Sistem mengintegrasikan beberapa sensor lingkungan, yaitu sensor PIR untuk deteksi gerakan, sensor DHT22 untuk pemantauan suhu dan kelembaban, serta sensor LDR untuk pengukuran intensitas cahaya. Komunikasi data antar perangkat dilakukan menggunakan protokol Message Queuing Telemetry Transport (MQTT) dengan broker Eclipse Mosquitto untuk mendukung pertukaran data yang ringan dan real-time. Sistem menerapkan logika kendali hybrid yang menggabungkan mode otomatis berbasis sensor dan kendali manual oleh pengguna melalui dashboard Node-RED. Data sensor dan status aktuator dipublikasikan ke topik MQTT untuk divisualisasikan secara real-time, serta disimpan ke dalam Google Firebase Realtime Database sebagai data historis. Pengembangan sistem dilakukan menggunakan framework Arduino pada ESP8266 sehingga memungkinkan implementasi firmware yang modular dan mudah dikembangkan. Hasil pengujian fungsional menggunakan metode black box menunjukkan bahwa sistem mampu membaca kondisi lingkungan dengan baik, merespons perintah kendali secara tepat, serta menjaga kestabilan komunikasi dua arah antara perangkat keras dan perangkat lunak. Berdasarkan hasil tersebut, dapat disimpulkan bahwa sistem yang dikembangkan layak digunakan sebagai prototipe otomasi gedung berbasis IoT yang fleksibel, berbiaya rendah, dan mudah dikembangkan lebih lanjut.

Keywords: IoT, Mosquitto, MQTT, PlatformIO, ESP8266, Arduino, Node-RED, Python, Firebase, Smart Building

1. PENDAHULUAN

1.1. Latar Belakang

Pada era modern saat ini, kemajuan teknologi telah menjadi faktor penting dalam meningkatkan efektivitas operasional dan efisiensi penggunaan energi. Salah satu teknologi yang berkembang pesat adalah Internet of Things (IoT), yang memungkinkan berbagai perangkat fisik saling terhubung dan bertukar data secara real-time. Pemanfaatan IoT kini dapat ditemukan pada berbagai sektor, seperti pertanian, perkebunan, manajemen gedung, otomasi rumah pintar, hingga industri manufaktur. [1]

Meskipun IoT terlihat bekerja secara otomatis, sistem ini memiliki rangkaian proses kompleks di belakangnya, mulai dari pengumpulan data menggunakan sensor, pemrosesan informasi melalui perangkat mikrokontroler, hingga pengiriman dan pengendalian perangkat melalui protokol komunikasi tertentu. Dengan demikian, sebuah sistem IoT yang berfungsi dengan baik membutuhkan integrasi yang selaras antara hardware (sensor dan aktuator), software (mikrokontroler, firmware, dan protokol komunikasi), serta platform pendukung seperti dashboard pemantauan dan server komunikasi.[2]

Penelitian ini secara khusus difokuskan pada pengembangan sistem otomasi gedung yang memanfaatkan sensor gerak, sensor suhu, dan sensor cahaya untuk mendeteksi kondisi lingkungan dalam ruangan. Sistem ini dirancang agar mampu mengontrol perangkat listrik seperti lampu dan pendingin ruangan berdasarkan data yang diterima. Dari sisi komunikasi, penelitian ini menggunakan protokol MQTT (Mosquitto) sebagai penghubung antara perangkat IoT dan server, serta platform Node-RED untuk visualisasi data dan pengelolaan alur otomasi. Sementara itu, proses pemrograman perangkat dilakukan menggunakan PlatformIO yang menyediakan lingkungan pengembangan modern dan efisien.

Meskipun teknologi otomasi gedung berbasis IoT telah banyak dikembangkan, implementasinya masih menghadapi sejumlah tantangan, terutama dalam hal integrasi sensor, keandalan komunikasi data, dan efisiensi respons sistem terhadap kondisi lingkungan secara real-time.[4] Banyak sistem otomasi yang ada saat ini masih bergantung pada perangkat komersial yang bersifat tertutup (*closed-source*), sehingga kurang fleksibel untuk dikembangkan atau disesuaikan dengan kebutuhan spesifik suatu gedung. Selain itu, beberapa penelitian sebelumnya belum memberikan perhatian lebih pada penggabungan multi-sensor sederhana seperti PIR, LDR, dan DHT22 dengan protokol MQTT yang ringan namun efisien untuk kebutuhan otomasi skala kecil hingga menengah. Oleh karena itu, penelitian ini diperlukan untuk mengevaluasi bagaimana kombinasi platform open-source seperti ESP8266, Mosquitto MQTT, dan Node-RED dapat menghasilkan sistem otomasi gedung yang lebih terjangkau, mudah dikembangkan, dan tetap responsif terhadap perubahan kondisi lingkungan.

Dengan pemanfaatan ESP8266 sebagai mikrokontroler utama, sistem otomasi gedung yang dibangun diharapkan mampu menghadirkan solusi yang terjangkau, mudah dikembangkan, dan dapat meningkatkan efisiensi energi serta kenyamanan dalam pengelolaan gedung.

1.2. Rumusan Masalah

Beberapa rumusan masalah yang dapat dikemukakan dalam penelitian ini adalah:

1. Bagaimana sistem data mengatur perangkat seperti lampu dan AC secara otomatis berdasarkan sensor dan perintah manual?
2. Bagaimana menampilkan dan mengolah data yang didapat dari sensor untuk diaplikasikan sebagai sistem otomasi gedung?
3. Bagaimana mengimplementasikan pertukaran data dari hardware dan software menggunakan protokol Mosquitto?
4. Bagaimana mengimplementasikan keamanan pertukaran data dengan menggunakan protokol Mosquitto?

1.3. Tujuan Penelitian

Adapun beberapa tujuan dari penelitian ini dilakukan adalah sebagai berikut:

1. Merancang dan membangun sistem otomasi gedung berbasis Internet of Things (IoT) yang mampu mengontrol perangkat seperti lampu dan AC berdasarkan kondisi lingkungan secara otomatis melalui sensor PIR, LDR, DHT-22, serta mikrokontroler ESP8266.
2. Mengimplementasikan mekanisme pertukaran data antara hardware dan software menggunakan protokol MQTT serta Node-RED sebagai platform monitoring dan kontrol jarak jauh.
3. Mengimplementasikan pengolahan dan visualisasi berdasarkan data sensor untuk meningkatkan efisiensi energi pada gedung.
4. Merancang sistem keamanan pertukaran data dalam komunikasi software dan hardware menggunakan protokol Mosquitto.

2. LANDASAN TEORI

Dalam mengaplikasikan sistem *Internet of things* otomasi gedung, diperlukan pemahaman lebih lanjut mengenai setiap aspek yang ada di dalam sistem tersebut, baik dari software, hardware, *machine learning* dan *cybersecurity*.

2.1. Perangkat Lunak (Software)

Software memiliki peranan yang penting dalam menyusun sistem Internet of Things (IoT). Jika perangkat keras (*hardware*) berfungsi sebagai indra dan otot yang menggerakkan suatu sistem, maka perangkat lunak berperan sebagai sistem dan saraf otak yang mengatur cara berpikir, komunikasi, serta User Interface (UI). Untuk membangun sistem otomasi gedung yang responsif dan dapat diandalkan, sistem bukan hanya membutuhkan sensor dan mikrokontroler, tapi juga *platform-platform* yang digunakan untuk memproses data diterima [1]. Penelitian ini menggunakan tiga lapisan perangkat lunak (software layers), mulai dari *firmware* hingga *application layer*. Pemilihan perangkat lunak pada penelitian ini didasarkan pada prinsip efisiensi sumber daya, kemudahan integrasi, dan ketersediaan informasi secara open source.

1. *Message Queuing Telemetry Transport* (MQTT)

Komunikasi data yang efisien adalah poin penting dalam hal keberhasilan implementasi IoT, terutama ketika melibatkan *Low-Power Microcontrollers* dan komponen-komponen yang hemat energi. Oleh sebab itu, penelitian ini tidak mengadopsi protokol HTTP yang biasa digunakan pada web, melainkan MQTT.[18]

MQTT adalah protokol komunikasi ringan yang dirancang untuk berkomunikasi mesin ke mesin. Keunggulan utama MQTT terletak pada mekanisme *Publish-Subscribe*. Berbeda halnya dengan model *Request-Response* yang klien harus terus-menerus meminta data ke server. *Publish-Subscribe* memungkinkan adanya pemisahan antara pengirim data (publisher) dan penerima (subscriber).[18]

Dalam skema ini, terdapat komponen yang berperan sebagai distributor, yaitu Broker. Ketika sensor mendeteksi adanya perubahan suhu, mikrokontroler atau publisher cukup mengirimkan data ke Broker satu kali saja. Broker kemudian bertanggung jawab untuk mendistribusikan data tersebut ke semua perangkat lainnya yang membutuhkan, seperti dashboard untuk memantau kondisi lingkungan. Dengan cara ini, maka secara signifikan mengurangi beban *bandwidth* dan konsumsi daya pada mikrokontroler, menjadikannya solusi yang ideal untuk sistem otomasi gedung yang beroperasi terus menerus tanpa henti.[4]

2. Node-RED

Setelah data berhasil dikirim ke jaringan, data harus diolah dan divisualisasikan agar bermanfaat bagi pengguna [2]. Dalam mengolah dan memvisualisasikan data dari hardware penelitian ini menggunakan Node-RED, sebuah alat pengembangan berbasis aliran atau *flow-based programming* [3].

Node-RED memberikan pendekatan visual dalam pemrograman, yang memungkinkan logika sistem dibangun dengan menghubungkan *node* fungsi. Keunggulan utama Node-RED terletak pada arsitekturnya yang dibangun berdasarkan Node.js, yang memungkinkannya berjalan secara *event-driven* dan *non-blocking* [3]. Artinya, sistem dapat menangani banyak proses data sensor secara bersamaan tanpa mengalami latensi.

Dalam konteks penelitian ini, Node-RED berfungsi sebagai pengelola logika otomasi dan penyedia antarmuka (*dashboard*) yang memungkinkan pemantauan kondisi ruangan secara visual dan *real-time* [4]. Fleksibilitas ini memungkinkan pengembangan prototipe dilakukan dengan cepat serta memudahkan modifikasi alur kerja sistem di kemudian hari nanti tanpa perlu merombak kode program.

3. PlatformIO

Pengembangan *firmware* untuk sistem IoT yang kompleks membutuhkan pemrograman yang handal, terstruktur, dan memiliki *library management* yang efisien. Untuk memenuhi kebutuhan tersebut, penelitian ini menggunakan PlatformIO sebagai ekosistem pengembangannya.

PlatformIO adalah sebuah ekosistem pengembangan *cross-platform* untuk perangkat IoT yang terintegrasi dengan *code editor* seperti Visual Studio Code [5]. PlatformIO memberikan pendekatan manajemen proyek yang terisolasi. Sehingga, setiap proyek memiliki berkas konfigurasi mandiri yang mendefinisikan secara spesifik jenis *board* mikrokontroler, *framework*, serta versi yang dibutuhkan [6].

Dalam konteks penelitian ini, PlatformIO digunakan sebagai basis pemrograman C++ untuk bisa membaca maupun memberikan perintah kepada hardware. PlatformIO memberikan keuntungan yang signifikan dalam hal reprodusibilitas kode dan kecepatan pengembangan. Melalui PlatformIO, kode program ditulis menggunakan *framework* Arduino untuk kemudian dikompilasi dan diunggah ke dalam mikrokontroler ESP8266 [7].

4. Google Firebase

Sistem IoT yang dapat diandalkan membutuhkan tempat penyimpanan yang mampu menangani sinkronisasi secara *real-time*. Untuk menjawab hal ini, penelitian ini menggunakan layanan Google Firebase, khususnya fitur Realtime Database.

Firebase adalah platform pengembangan aplikasi seluler web yang menyediakan layanan Backend-as-a-Service (BaaS) [8]. Berbeda dengan database SQL pada umumnya yang menggunakan MySQL dan berbasis tabel, Firebase menggunakan basis data NoSQL yang menyimpan data dalam format JSON. Keunggulannya, terletak pada sistem sinkronisasi data secara otomatis [8]. Ketika terjadi perubahan suhu pada suatu klien, Firebase akan memperbaharui data tersebut di semua perangkat lain yang saling terhubung.

Firebase berperan sebagai tempat penyimpanan data historis (cloud storage) dan pengelola status perangkat. Penggunaan Firebase memenuhi kekosongan dalam kebutuhan untuk membangun dan mengelola server basis data, sehingga dapat berfokus pada pengembangan logika dan efisiensi sistem [9].

5. Wireshark

Wireshark adalah sebuah alat network protocol analyzer open-source yang digunakan untuk menangkap dan menganalisa lalu lintas data pada jaringan komputer secara real-time. Dengan Wireshark, pengguna dapat melihat paket data yang melintas pada interface jaringan, memeriksa header dan isi paket, serta mendiagnosis masalah jaringan atau aktivitas tidak biasa seperti serangan atau trafik abnormal. Alat ini banyak dipakai oleh administrator jaringan, profesional keamanan, dan peneliti untuk memahami komunikasi protokol yang terjadi dalam jaringan dan memecahkan berbagai permasalahan terkait performa atau keamanan jaringan. [19]

2.2. Perangkat Keras (Hardware)

Dalam perancangan sistem Internet of Things (IoT) ini, berbagai komponen perangkat keras digunakan untuk melakukan proses pengumpulan data, aktuasi, serta interkoneksi antar perangkat. Setiap komponen memiliki karakteristik dan peran spesifik yang mendukung mekanisme deteksi lingkungan dan respon otomatis. Adapun perangkat keras yang digunakan meliputi sensor lingkungan, aktuator, media prototipe rangkaian, serta komponen pendukung.

1. Sensor DHT 22

Sensor DHT22 merupakan sensor digital yang berfungsi mengukur suhu dan kelembaban dengan tingkat akurasi yang tinggi. DHT22 bekerja dengan mengubah perubahan kelembaban pada material resistif menjadi sinyal digital yang kemudian dikirim melalui satu pin data menuju mikrokontroler. Sensor ini memiliki akurasi suhu ±0,5°C serta rentang pengukuran kelembaban hingga 100% RH, sehingga cocok digunakan untuk mengukur suhu pada setiap sudut ruangan agar sistem dapat mengatur suhu pada ruangan sesuai kondisi nyata pada ruangan [4].

2. Sensor PIR (Passive Infrared Sensor)

Sensor PIR digunakan untuk mendeteksi keberadaan manusia berdasarkan perubahan radiasi inframerah yang dipancarkan tubuh. PIR tidak memancarkan sinyal, melainkan hanya menerima dan membedakan perubahan energi panas di area cakupannya. Ketika terdeteksi adanya gerakan tubuh manusia, sensor memberikan sinyal logika HIGH pada mikrokontroler sebagai tanda adanya aktivitas di ruangan [1].

3. Sensor LDR (Light Dependent Resistor)

Sensor LDR bekerja dengan cara mengubah energi dari foton menjadi elektron, umumnya satu foton dapat

membangkitkan satu electron. Ketika cahaya terang, resistansi LDR menurun; ketika ruangan gelap, resistansi meningkat. Nilai resistansi yang berubah ini diterjemahkan sebagai variasi tegangan oleh mikrokontroler untuk mengetahui kondisi pencahayaan lingkungan [10].

4. Breadboard

Breadboard adalah board yang digunakan untuk membuat rangkaian elektronik sementara dengan tujuan uji coba atau prototipe tanpa harus menyolder. Beberapa orang terkadang menyebutnya papan proyek atau bahkan papan prototipe [11].

5. Kabel Jumper

Kabel jumper berfungsi sebagai penghubung antar-komponen, seperti sensor ke mikrokontroler, LED ke resistor, ataupun servo ke pin digital. Kabel jenis male-to-male, male-to-female, dan female-to-female digunakan sesuai dengan kebutuhan pin pada breadboard dan modul sensor.

6. LED (Light Emitting Diode)

LED digunakan sebagai indikator status sistem. Misalnya, LED dapat menyala ketika sistem mendeteksi LDR bernilai rendah, atau mati ketika intensitas cahaya di dalam ruangan tinggi.

7. Servo Motor

Servo motor merupakan aktuator yang digunakan untuk menghasilkan gerakan mekanis presisi berdasarkan sinyal PWM dari mikrokontroler. Servo digunakan ketika sistem membutuhkan gerakan terkontrol, seperti membuka–menutup penutup otomatis,. Pada sistem yang dirancang, servo digunakan sebagai aktuator yang akan menyalakan AC.

8. Resistor

Resistor merupakan komponen pasif yang berfungsi mengatur dan membatasi aliran arus listrik dalam rangkaian. Resistor banyak dijumpai pada berbagai rangkaian elektronik sederhana maupun kompleks. Pada sistem IoT, resistor digunakan untuk berbagai keperluan, seperti pembatas arus LED, penstabil sinyal sensor, serta bagian dari rangkaian *voltage divider* untuk membaca nilai analog dari sensor seperti LDR [10].

9. ESP8266

ESP8266 adalah sebuah modul mikrokontroler yang dikembangkan oleh Espressif Systems dan dirancang khusus untuk kebutuhan Internet of Things (IoT). Modul ini mengintegrasikan prosesor, memori, serta kemampuan komunikasi Wi-Fi dalam satu chip, sehingga memungkinkan perangkat elektronik untuk terhubung langsung ke jaringan internet tanpa memerlukan modul tambahan. [11]

ESP8266 dapat diprogram untuk mengendalikan input dan output seperti sensor dan aktuator, mirip dengan fungsi mikrokontroler pada platform Arduino. Namun, keunggulan utama ESP8266 terletak pada konektivitas nirkabelnya yang telah terintegrasi, sehingga sangat efisien untuk aplikasi berbasis jaringan, seperti monitoring jarak jauh, otomasi rumah, dan sistem kendali berbasis web. Dengan konsumsi daya yang relatif rendah, ukuran yang kecil, serta biaya yang terjangkau, ESP8266 menjadi salah satu platform yang populer dan banyak digunakan dalam pengembangan sistem IoT. [11]

10. Arduino

Arduino Uno merupakan papan mikrokontroler satu papan (single-board microcontroller) yang dirancang agar

proses pembuatan proyek elektronika menjadi jauh lebih mudah, terutama bagi pemula maupun pengembang dari berbagai disiplin ilmu. Dengan menyediakan antarmuka yang sederhana, port input–output digital dan analog, serta lingkungan pemrograman (IDE) yang mudah digunakan, Arduino Uno memungkinkan siapa saja, baik dari bidang teknik, desain, seni, hingga pendidikan, untuk mewujudkan ide interaktif tanpa harus memahami detail elektronika tingkat lanjut [12].

2.3. Keamanan dan Pengolahan Data

Data sensor IoT yang diterima dalam format JSON merupakan data mentah yang bersifat heterogen dan deret waktu, sehingga memerlukan proses pengolahan sebelum dianalisis. Dalam kajian statistika, pengolahan data mencakup serangkaian upaya untuk meningkatkan kualitas, keterbacaan, dan keterwakilan data agar layak dianalisis secara kuantitatif. Rahm dan Do menegaskan bahwa pembersihan data merupakan tahap fundamental dalam pengolahan data karena kesalahan, duplikasi, dan nilai tidak valid dapat menurunkan akurasi hasil analisis secara signifikan [13]. Setelah data berada dalam kondisi konsisten, pengolahan dilanjutkan dengan transformasi dan ekstraksi fitur untuk merepresentasikan informasi penting, seperti pola aktivitas dan kondisi lingkungan, sehingga data mentah dapat diubah menjadi bentuk yang bermakna secara statistik. Pendekatan ini sejalan dengan pandangan Han, Kamber, dan Pei yang menyatakan bahwa transformasi dan reduksi data berperan penting dalam meningkatkan efisiensi serta interpretabilitas analisis data [14]. Data yang telah diolah kemudian dianalisis menggunakan statistika deskriptif, yaitu cabang statistika yang bertujuan untuk merangkum dan menggambarkan karakteristik utama data melalui ukuran frekuensi, kecenderungan, dan distribusi tanpa melakukan penarikan kesimpulan inferensial, sebagaimana dijelaskan oleh Montgomery [15]. Melalui statistika deskriptif, perilaku sistem dapat dipahami secara sistematis dan menjadi dasar bagi visualisasi serta pengambilan keputusan pada sistem IoT.

Setelah karakteristik data dipahami melalui statistika deskriptif, analisis dilanjutkan dengan pendekatan peramalan deret waktu untuk memprediksi perilaku sistem IoT di masa mendatang. Mengingat data sensor IoT bersifat temporal dan menunjukkan kecenderungan tren serta pola musiman, model Holt-Winters dipilih sebagai metode peramalan karena kemampuannya dalam memodelkan komponen level, tren, dan musiman secara simultan. Holt-Winters merupakan bagian dari metode *exponential smoothing* yang memberikan bobot lebih besar pada data terbaru sehingga responsif terhadap perubahan kondisi lingkungan, sebagaimana dijelaskan oleh Hyndman dan Athanasopoulos [16]. Penerapan model ini memungkinkan sistem untuk memprediksi nilai sensor secara jangka pendek berdasarkan pola historis yang telah diolah, sehingga mendukung pengambilan keputusan yang lebih proaktif pada sistem IoT, seperti deteksi anomali atau peringatan dini terhadap kondisi lingkungan yang tidak normal.[20]

Keamanan pada protokol Message Queuing Telemetry Transport (MQTT) merupakan aspek fundamental dalam sistem Internet of Things (IoT) mengingat karakteristiknya yang ringan dan terbuka terhadap jaringan publik. Standar internasional ISO/IEC menegaskan bahwa pengamanan komunikasi data harus mencakup mekanisme pengendalian akses dan pencatatan aktivitas sebagai bagian dari sistem manajemen keamanan informasi. Dalam ISO/IEC 27001 dan ISO/IEC 27002, kontrol autentikasi dan pengelolaan identitas diklasifikasikan sebagai *access control*, yang bertujuan memastikan bahwa hanya entitas yang berwenang yang dapat mengakses sistem dan layanan jaringan, termasuk layanan komunikasi data seperti protokol MQTT [16]. Autentikasi username dan password pada broker MQTT dengan demikian berperan sebagai mekanisme dasar untuk mencegah akses tidak sah dan

penyalahgunaan layanan.Selain autentikasi, standar keamanan juga menekankan pentingnya pemantauan dan pencatatan aktivitas sistem. NIST melalui *Special Publication* 800-92 menegaskan bahwa logging dan monitoring merupakan komponen krusial dalam keamanan sistem informasi karena memungkinkan pendeteksian aktivitas mencurigakan, analisis insiden, serta mendukung proses *forensic analysis* pasca kejadian [17].

3. METODOLOGI PENELITIAN

3.1 Metode Penelitian

Metodologi penelitian ini disusun untuk memastikan pengembangan desain sistem otomasi gedung berbasis IoT dapat berjalan dengan terstruktur dan sistematis. Pendekatan yang digunakan pada penelitian ini adalah metode *Prototyping*, yang memungkinkan pengembangan perangkat lunak (*software*) serta perangkat keras (*hardware*) dilakukan secara bertahap dan berulang (*iterative*). Tahapan penelitian disusun secara terstruktur dimulai dari identifikasi masalah, studi literatur, analisis kebutuhan, perancangan, hingga pengujian.

3.2 Analisis Kebutuhan

Tahap ini bertujuan untuk menentukan spesifikasi kebutuhan agar sistem otomasi gedung dapat berjalan dengan efisien dan optimal. Kebutuhan dibagi menjadi dua kategori, yaitu:

3.2.1 Kebutuhan Perangkat Keras (Hardware)

Hardware yang digunakan dipilih berdasarkan tingkat efisiensi dan kompatibilitas dengan sistem:

1. NodeMCU ESP 8266: Sebagai unit pemrosesan utama dan modul komunikasi.
2. Sensor PIR (HC-SR501): Sebagai pemicu utama untuk mendeteksi ada tidaknya manusia dalam ruangan.
3. Sensor DHT22: Untuk mengukur suhu dan kelembaban ruangan secara presisi.
4. Sensor LDR (5mm): Untuk mengukur intensitas cahaya suatu ruangan.
5. Servo Motor (SG90): Sebagai aktuator untuk menekan saklar AC dan lampu secara fisik.
6. LED: Sebagai demonstrasi lampu
7. Arduino: Sebagai power supply tambahan
8. kabel jumper: Sebagai penghubung antar komponen seperti sensor, aktuator, dan mikrokontroler.
9. breadboard: Sebagai media perakitan rangkaian sementara tanpa perlu penyolderan, sehingga memudahkan proses pengujian dan pengembangan.
10. Resistor: Sebagai pembatas arus, khususnya pada LED dan sensor LDR agar komponen tidak rusak dan pembacaan data lebih stabil.

3.2.2 Kebutuhan Perangkat Lunak (Software)

Software dibutuhkan untuk mengembangkan logika, komunikasi, dan *user interface*:

1. PlatformIO: Sebuah *Integrated Development Environment* (IDE) yang terintegrasi dengan Visual Studio Code untuk penulisan *firmware* menggunakan *framework* Arduino.
2. Eclipse Mosquitto: Sebagai broker MQTT untuk mengelola pertukaran pesan secara *real-time*.
3. Node-RED: Platform pengembangan berbasis pengembangan (*flow-based*) yang digunakan untuk merancang logika otomasi serta visualisasi data.
4. Google Firebase: Cloud database menyimpan data sensor yang dapat digunakan untuk mengambil data keperluan analisis.
5. Python: Bahasa pemrograman yang dapat membantu melaksanakan pengolahan data dan memberikan statistika deskriptif mengenai data yang sudah diambil.
6. Bot Telegram: Sarana yang dapat digunakan untuk bisa memberikan perintah mematikan atau menyalakan lampu.

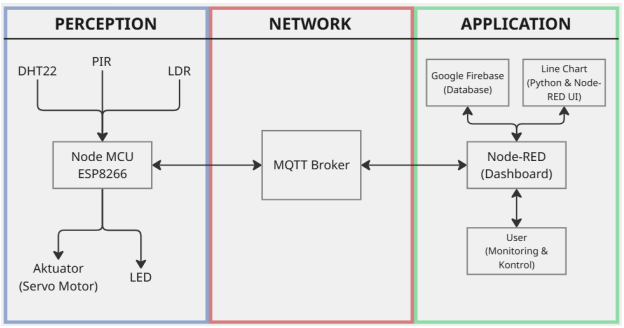
3.3 Perancangan Sistem

Perancangan sistem merupakan tahap yang menghubungkan seluruh komponen hasil analisis kebutuhan. Tahapan ini dibagi menjadi perancangan arsitektur sistem dan perancangan logika sistem.

3.3.1 Perancangan Arsitektur Sistem

Untuk mengintegrasikan berbagai komponen perangkat keras dan perangkat lunak yang digunakan, sistem ini dirancang dengan menggunakan model Three Tier Architecture. Arsitektur ini membagi sistem menjadi tiga lapisan utama, yaitu Perception Layer, Network Layer, dan Application Layer. Pembagian ini bertujuan untuk meningkatkan efisiensi aliran data secara real-time dan memudahkan proses debugging jika terjadi

kesalahan dalam salah satu bagian. Detail pembagian lapisan arsitektur sistem dapat dilihat pada **Gambar 1**.

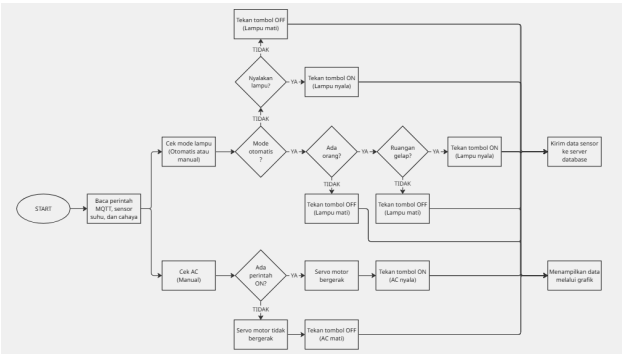


Gambar 1. Diagram Blok Arsitektur Sistem

1. *Perception Layer*: Lapisan ini merupakan bagian yang berinteraksi langsung dengan lingkungan. Mikrokontroler NodeMCU ESP8266 berfungsi sebagai unit pemrosesan utama yang memproses logika dari sensor PIR, DHT22, dan LDR. Selain sebagai pembaca sensor, ESP8266 juga berperan sebagai eksekutor yang mengirimkan sinyal listrik ke aktuator (servo motor) dan LED.
2. *Network Layer*: Lapisan ini berfungsi sebagai jembatan untuk komunikasi data. Protokol MQTT digunakan karena sifatnya yang ringan dan efisien untuk komunikasi data *real-time*. Eclipse Mosquitto berperan sebagai broker yang mengatur lalu lintas pesan. ESP8266 men-*publish* data sensor ke broker melalui Wi-fi, sekaligus men-*subscribe* topik perintah kendali ke server.
3. *Application Layer*: Lapisan ini merupakan antarmuka interaksi antara sistem dan pengguna. Node-RED digunakan untuk mengelola data dari MQTT lalu divisualisasikan Node-RED UI dengan bantuan Python. Node-RED juga terintegrasi dengan Google Firebase untuk menyimpan data (logging) secara teratur.

3.3.2 Perancangan Logika Sistem

Perancangan logika sistem bertujuan untuk mengubah kebutuhan fungsional menjadi algoritma yang akan dipakai oleh mikrokontroler NodeMCU ESP8266. Logika kendali sistem ini bersifat hybrid, yaitu menggabungkan otomasi berdasarkan sensor lingkungan dan kendali manual oleh pengguna. Alur logika program digambarkan melalui flowchart pada **Gambar 2**.



Gambar 2. *Flowchart* logika sistem

Berdasarkan gambar di atas, algoritma sistem terdiri dari beberapa tahapan proses:

1. Inisialisasi: Saat sistem dinyalakan, mikrokontroler menyiapkan sensor dan aktuator, lalu terhubung ke Wi-Fi dan broker MQTT. Setelah koneksi berhasil dilakukan, sistem akan *subscribe* ke topik perintah perintah kendali untuk menerima instruksi dari pengguna.

2. Pengumpulan data sensor: Sistem secara *real-time* membaca kondisi lingkungan di sekitarnya. Data yang diambil meliputi deteksi gerakan dari sensor PIR, suhu dan kelembaban dari sensor DHT22, serta mengukur intensitas cahaya dari sensor LDR.

3. Logika kendali lampu (hybrid): Sistem mengecek mode yang dipilih oleh pengguna melalui dashboard Node-RED:
    a. Mode Manual: Sistem menghiraukan pembacaan sensor LDR dan hanya mengubah status lampu LED (ON/OFF) berdasarkan perintah tombol yang diterima dari MQTT.
    b. Mode Otomatis: Sistem menerapkan logika kondisional AND. Lampu LED akan dinyalakan hanya jika sensor PIR mendeteksi gerakan dan sensor LDR mendeteksi intensitas cahaya yang rendah.

4. Logika kendali AC (remote): AC dikendalikan sepenuhnya secara manual oleh pengguna. Sistem memeriksa apakah terdapat perintah masuk pada topik MQTT. Jika pengguna mengirimkan perintah “ON”, mikrokontroler akan mengirimkan sinyal *Pulse Width Modulation* (PWM) yang

menggerakkan servo motor untuk menekan saklar AC.

5. Publikasi data: Pada akhir *loop*, seluruh data sensor dan status aktuator (dalam format JSON) dipublikasikan ke topik MQTT untuk divisualisasikan pada dashboard Node-RED dan disimpan di Google Firebase database.

3.4 Skenario Pengujian

Pengujian sistem dilakukan untuk memvalidasi apakah hardware dan software telah berjalan sesuai dengan ketentuan yang dirancang. Metode pengujian yang digunakan adalah Black Box Testing, yang berfokus pada pengujian fungsionalitas *input* dan *output* tanpa melihat struktur kode internal program.

Pengujian ini mencakup berbagai kondisi operasional sistem. Rincian mengenai skenario pengujian yang akan dilakukan pada masing-masing modul sistem ditunjukkan pada **Tabel 1.**

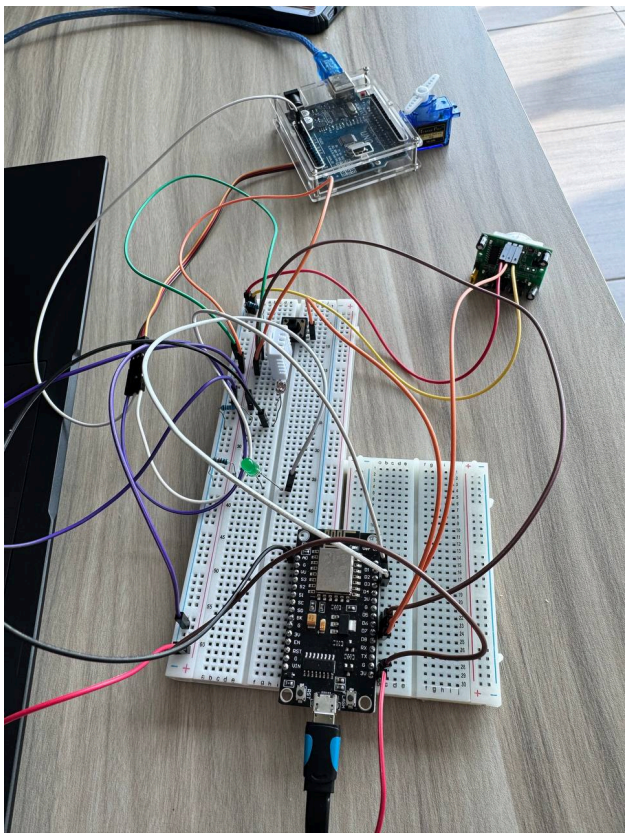
Tabel 1. Rencana Skenario Pengujian Sistem

Skenario Pengujian	Hasil yang Diharapkan
Uji konektivitas: Menghubungkan ESP8266 ke sumber daya listrik	Status “Online” muncul pada MQTT broker dan dashboard.
Uji sensor DHT22: Membiarkan sistem berjalan di ruangan.	Nilai suhu dan kelembapan muncul pada Dashboard Node-RED dan Firebase
Uji sensor PIR: Melakukan gerakan fisik di depan sensor.	Status gerakan pada dashboard berubah menjadi ‘HIGH”.
Uji sensor LDR: Menutup sensor dan menyinarinya secara bergantian.	Nilai intensitas cahaya berubah drastis sesuai dengan kondisi cahaya.
Uji lampu mode otomatis: mengaktifkan mode otomatis, lalu memberi kondisi gelap dan ada gerakan.	LED menyala otomatis dan akan mati ketika tidak ada gerakan.
Uji lampu mode manual: mengaktifkan mode manual, lalu menekan tombol “ON” pada dashboard.	LED menyala tanpa terpengaruh kondisi sensor.
Uji AC: Menekan tombol “ON” pada dashboard.	Motor servo bergerak untuk menyalakan saklar AC.
Uji integrasi data: mengamati pembaharuan data Firebase pada dashboard.	Grafik Node-RED dan log Firebase terupdate secara *real-time*.
Uji Sistem Keamanan: mengamati sistem keamanan dari mqtt dalam melakukan komunikasi.	Terdapat Log monitor dari mqtt yang memberikan semua informasi dari publish, subscribe, transfer data dan lainnya.

4. IMPLEMENTASI

4.1. Perancangan Hardware

Perancangan hardware dilakukan untuk memastikan seluruh komponen sistem Internet of Things (IoT) dapat bekerja secara terintegrasi dan sesuai fungsi. Setiap komponen dipilih berdasarkan kebutuhan proses sensing, aktuasi, serta komunikasi data. Implementasi perancangan mencakup penyusunan rangkaian sensor, aktuator, mikrokontroler, dan komponen pendukung menggunakan breadboard sebagai media prototipe. Perlu diketahui setiap komponen yang terhubung dengan pin mikrokontroler akan juga terhubung pada sumber tegangan (positive) dan *Ground* (Negatif). Keseluruhan rangkaian dapat dilihat pada



Gambar 3. Rangkaian Hardware

1. Sensor DHT22 (Suhu dan Kelembaban)

Sensor DHT22 dipasang sebagai perangkat pemantau kondisi temperatur dan kelembaban ruangan. Sensor ini dihubungkan ke salah satu pin digital mikrokontroler ESP8266 untuk membaca data secara berkala. Penggunaan DHT22 relevan karena sensor ini telah terbukti mampu memberikan pembacaan suhu yang akurat dalam implementasi otomasi ruangan berbasis IoT. Data dari sensor ini digunakan sebagai parameter tambahan untuk pengambilan keputusan otomatis oleh mikrokontroler.

2. Sensor PIR (Passive Infrared Sensor)

Sensor PIR dipasang sebagai perangkat pendeteksi pergerakan. Sensor ini ditempatkan pada posisi terbuka dan diarahkan ke area yang ingin dipantau. Output PIR dihubungkan ke pin digital mikrokontroler untuk mendeteksi perubahan logika akibat pergerakan. Hasil deteksi PIR menjadi salah satu pemicu aktivasi LED maupun servo dalam sistem.

3. Sensor LDR

Sensor LDR digunakan untuk mengukur intensitas cahaya sekitar. Pada perancangan ini, LDR dikombinasikan dengan resistor 1kΩ untuk membentuk rangkaian *voltage divider*, sehingga menghasilkan perubahan tegangan yang dapat dibaca mikrokontroler melalui pin analog. Dengan demikian, kondisi terang atau gelap dapat diolah sistem untuk mengendalikan aktuator.

4. Breadboard dan Kabel Jumper

Seluruh komponen dirangkai pada breadboard untuk memudahkan proses modifikasi dan troubleshooting. Kabel jumper digunakan sebagai koneksi antar-komponen, baik untuk jalur daya maupun sinyal. Penggunaan breadboard mendukung fleksibilitas sistem selama fase pengembangan dan pengujian.

5. LED

LED dipasang sebagai output visual untuk menunjukkan status sistem, seperti indikasi deteksi gerakan oleh sensor PIR atau perubahan kondisi cahaya dari sensor LDR. LED dihubungkan dengan resistor untuk menghindari kerusakan akibat arus yang terlalu besar.

6. Servo

Servo digunakan untuk menghasilkan gerakan presisi, misalnya membuka atau menutup mekanisme AC. Servo dihubungkan ke pin 5V Arduino untuk mendapatkan daya dikarenakan ESP8266 hanya menyediakan 3.3V.

7. Resistor

Resistor yang digunakan dalam rangkaian ini adalah 1kΩ dan 330Ω . Resistor 1kΩ dipasang secara seri dengan LDR untuk membentuk pembagi tegangan yang stabil, Kombinasi ini memastikan sinyal analog berubah proporsional terhadap intensitas cahaya. Resistor 330Ω juga digunakan sebagai pembatas arus untuk LED agar komponen tidak mengalami kerusakan akibat arus berlebih.

8. Arduino

Penggunaan Arduino pada rangkaian ini adalah sebagai sumber daya sebesar 5V untuk dapat menggerakan servo.

4.2. Perancangan Software

Perancangan software dilakukan untuk memastikan sistem Internet of Things (IoT) dapat berjalan sesuai dengan logika kendali yang telah dirancang serta mampu melakukan komunikasi data secara andal antara perangkat keras, server, dan antarmuka pengguna. Perangkat lunak pada sistem ini terdiri dari program mikrokontroler NodeMCU ESP8266, middleware komunikasi berbasis MQTT, dashboard monitoring dan kontrol menggunakan Node-RED, serta penyimpanan data pada Google Firebase.

1. Program Mikrokontroler NodeMCU ESP8266

NodeMCU ESP8266 diprogram menggunakan Arduino IDE dengan bahasa pemrograman C++ melalui PlatformIO. Program ini bertanggung jawab untuk mengelola pembacaan data sensor, menjalankan logika kendali hybrid, serta melakukan komunikasi data melalui protokol MQTT. Pada tahap awal, program melakukan inisialisasi library-library yang digunakan, pin, sensor, servo motor, topik komunikasi MQTT, alamat ip, MQTT user, koneksi jaringan Wi-Fi dan variabel-variabel lainnya yang digunakan di keseluruhan program. Adapun library yang digunakan pada esp8266 adalah sebagai berikut:

A. Adafruit_Sensor.h

Library ini menyediakan antarmuka standar untuk berbagai jenis sensor. Penggunaan library ini memudahkan integrasi sensor DHT22 dengan sistem serta meningkatkan kompatibilitas pembacaan data sensor secara terstruktur.

B. DHT.h dan DHT_U.h

Library DHT digunakan untuk membaca data suhu dan kelembaban dari sensor DHT22. Library ini menangani proses komunikasi satu kawat (single-wire protocol) antara ESP8266 dan sensor, sehingga pembacaan suhu dan kelembaban dapat dilakukan secara akurat dan periodik.

C. ESP8266WiFi.h

Library ini digunakan untuk mengaktifkan kemampuan Wi-Fi pada NodeMCU ESP8266. Melalui library ini, ESP8266 dapat terhubung ke jaringan Wi-Fi lokal menggunakan SSID dan password yang telah ditentukan, sehingga memungkinkan komunikasi data berbasis jaringan.

D. PubSubClient.h

Library PubSubClient digunakan untuk mengimplementasikan protokol MQTT pada ESP8266. Library ini memungkinkan perangkat berperan sebagai *publisher* dan *subscriber*, sehingga dapat mengirim data sensor serta menerima perintah kendali dari broker MQTT secara real-time.

E. Servo.h

Library Servo digunakan untuk mengendalikan motor servo melalui sinyal Pulse Width Modulation (PWM). Pada sistem ini, servo berfungsi sebagai aktuator untuk mensimulasikan penekanan tombol AC berdasarkan perintah yang diterima melalui MQTT.

```
// PEMANGGILAN LIBRARY

#include <Library>

#define DHTPIN “”

#define DHTTYPE “”

DHT dht(DHTPIN, DHTTYPE);

Servo servo;



// WIFI, MQTT USER, IP ADDRESS

const char* ssid = ""; //

const char* password = "";

const char* mqtt_user   = ""; //

const char* mqtt_pass   = ""; //

const char* mqtt_server = "";
```

```
const int mqtt_port = 1883;


// TOPIC MQTT

const char* topic_temp = "";


// MENDEFINISKAN PIN

int ldrPin   = A0;

int servoPin = D0;

int ledPin   = D2;

int pirPin   = D6;

int button   = D7;


// VARIABEL LAINNYA

bool lampState = false;

float humidity = 0.0;

float temperatureC = 0.0;

int servoPos = 0;

bool autoModeLamp = true;
// default = AUTO

int pressCount = 0;

unsigned long
lastDebounceLamp = 0;
```

Setelah semua elemen sudah didefinisikan, maka langkah selanjutnya adalah menghubungkan MQTT dengan jaringan Wi-Fi pada bagian `void setup ()`. Setelah koneksi jaringan berhasil, koneksi jangan lupa dipanggil pada bagian `void loop ()` agar komunikasi antara mikrokontroler terhubung ke broker MQTT dan melakukan subscribe pada topik kendali untuk menerima perintah dari pengguna. pada fungsi ini juga akan menjadi bagian saat menerima data dari hardware. Selain itu, pada konteks penelitian ini, MQTT juga akan memberikan pesan atau perintah pada hardware maka perlu dipersiapkan fungsi `void callback ()`

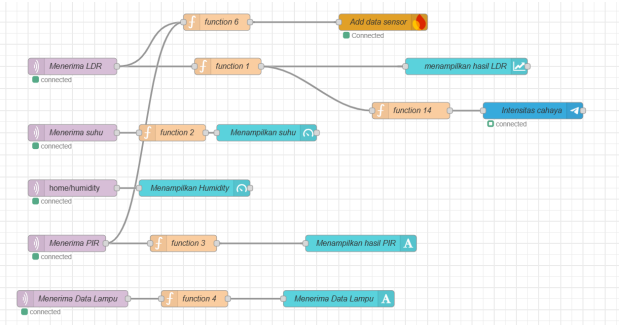
2. Komunikasi Data Menggunakan MQTT

Protokol Message Queuing Telemetry Transport (MQTT) digunakan sebagai media komunikasi utama antara NodeMCU, Node-RED dashboard, dan sistem penyimpanan data. MQTT dipilih karena memiliki overhead rendah dan sesuai untuk aplikasi IoT yang membutuhkan komunikasi real-time.

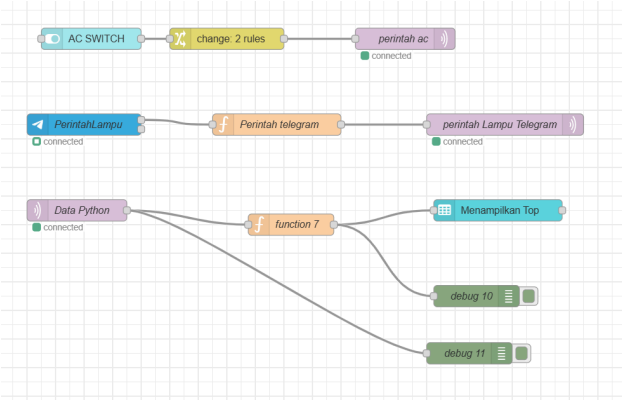
Dalam sistem ini, NodeMCU berperan sebagai publisher dan subscriber. NodeMCU mempublikasikan data sensor dan status aktuator ke topik tertentu, sementara perintah kendali dari pengguna dikirim melalui topik MQTT yang disubscribe oleh NodeMCU. Pendekatan ini memungkinkan sistem melakukan kontrol dua arah secara efisien dan responsif.

3. Perancangan Logika dan Dashboard Node-RED

Node-RED digunakan sebagai platform visual untuk monitoring dan pengendalian sistem. Dashboard Node-RED menampilkan data sensor secara real-time, seperti suhu, kelembaban, intensitas cahaya, serta status deteksi gerakan. Selain itu, dashboard menyediakan kontrol tombol untuk mengatur menyalakan atau mematikan AC melalui servo motor. Perlu diperhatikan pada setiap node MQTT (node berwarna ungu pada **Gambar 4** dan **Gambar 5.**) diperlukan pengaturan alamat ip, topic, MQTT user dan password. Hal ini bertujuan agar, Node-RED dapat membaca data yang sudah dikirimkan oleh MQTT.



Gambar 4. Rangkaian Node-RED



Gambar 5. Rangkaian Node-RED

4. Perancangan Database (Google Firebase)

Google Firebase digunakan sebagai media penyimpanan data sensor. Data yang dikirim oleh NodeMCU melalui MQTT diteruskan oleh Node-RED ke Firebase dalam bentuk database real-time. Penyimpanan ini memungkinkan data historis untuk dianalisis lebih lanjut serta mendukung pengembangan fitur monitoring jangka panjang. Data yang disimpan berupa data sensor LDR, PIR, dan Timestamp.

4.3. Keamanan dan Pengolahan data

Pengumpulan data sensor mentah perlu dilakukan analisis lebih lanjut agar dapat diolah untuk perkembangan sistem lebih lanjut. Dalam implementasi penelitian ini, data sensor yang sudah diambil, akan disimpan ke dalam *cloud database* (Google Firebase). Pengolahan data dimulai dengan mengambil data dari Firebase dalam bentuk Json melalui URL dengan menggunakan library `request` dari python.

```
import requests

url = "https://iotexperiment-9b165-default-rtdb.asia-southeast1.firebasedatabase.app/sensor/temperature.json"

response = requests.get(url)

raw = response.json()
```

Setelah melakukan pengambilan data, langkah selanjutnya adalah melakukan pelabelan data kemudian menyimpannya ke dalam satu kolom baru untuk dilanjutkan proses analisis.

```
if pir == 0 and ldr > 500:

    return "ruangan kosong"

if pir == 0 and ldr < 200:

    return "orang baru keluar"

if pir == 1 and ldr < 300:

    return "aktivitas belajar/kerja"

if pir == 1 and ldr > 500:

    return "istirahat / minim aktivitas"

if pir == 1 and 300 <= ldr <= 500:

    return "aktivitas normal"

return "lain"
```

Dengan label-label yang sudah ditetapkan pengolahan data dilanjutkan dengan mengambil data aktivitas, istirahat, ruangan kosong. Hal ini bertujuan agar dapat mengetahui kapan saja waktu yang paling sibuk, waktu istirahat dan waktu dimana ruangan kosong.

```
busy_hour_series = (

df[df["activity"].str.contains("aktivitas", case=False)]

    ["hour"]

    .value_counts()

)

jam_paling_sibuk = busy_hour_series.idxmax()
```

```
    rest_hour_series = (
df[df["activity"].str.contain
s("istirahat", case=False)]
        ["hour"]
        .value_counts()
    )
    jam_istirahat =
rest_hour_series.index.tolist
()
    empty_hour_series = (
        df[df["activity"] ==
"ruangan kosong"]
        ["hour"]
        .value_counts()
    )
    jam_paling_sepi =
empty_hour_series.idxmax()
```

Kemudian dalam mengimplementasikan analisis data lebih lanjut, penelitian ini menggunakan pendekatan *Machine Learning* dalam melakukan *forecasting* intensitas cahaya untuk 30 menit (*forcast_steps = 6)* kedepan. Dalam implementasi ini, model yang digunakan adalah *Holt-Winters* dengan parameter sebagai berikut:

```
hw_model =
ExponentialSmoothing(
    df_clean['ldr_smooth'],
    trend='add',
    seasonal='add',
    seasonal_periods=6
).fit(optimized=True)
```

Sebelum mengimplementasikan model ini, perlu diperhatikan sensor cahaya bisa saja memberikan data yang cenderung fluktuatif dan memberikan *outlier*. Untuk mencegah hal ini diperlukan pembersihan dan *smoothing* data dengan menggunakan metode interkuartil dan rolling window seperti pada kode yang diberikan berikut:

```
def remove_outlier_iqr(df,
column):
    Q1 =
df[column].quantile(0.25)
    Q3 =
df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df_clean = df[(df[column] >=
lower_bound) & (df[column] <=
upper_bound)]
    return df_clean
df_clean=remove_outlier_iqr(d
f, 'ldr')
# metode rolling window:
df_clean['ldr_smooth'] =
df['ldr'].rolling(30).mean()
df_clean['ldr_smooth'].fillna
(method='bfill', inplace=True)
print(df_clean)
```

Dari sisi keamanan, protokol mqtt diimplementasikan dengan menggunakan sistem keamanan autentikasi dan *monitor log*. Pada autentikasi, protokol mqtt akan menggunakan username dan password untuk bisa melakukan komunikasi dengan menjalankan CLI `mosquitto_passwd -c passwd.txt uname` pada powershell administrator dan kemudian melakukan pengaturan password yang ingin dipakai. Setelah melakukan proses di atas, protokol mqtt perlu dilakukan konfigurasi ulang pada file C:\Program Files\mosquitto\mosquitto.conf dengan menambahkan:

```
allow_anonymous false

password_file C:/Program Files/mosquitto/passwd.txt
```

Dalam menjalankan mosquitto, user perlu menjalankan CLI sebagai berikut

```
mosquitto -c "C:\Program Files\mosquitto\mosquitto.conf" -v
```

Kemudian proses dapat dilanjutkan dengan melakukan upload kode software ke hardware. Dengan melakukan pengaturan-pengaturan seperti di atas maka secara otomatis protokol mqtt akan memberikan monitor log data apa saja yang dikirim maupun diterima dalam berkomunikasi.

Selain itu menggunakan sistem keamanan moqquitto itu sendiri, dalam implementasi penelitian ini menggunakan sistem monitoring lebih lanjut dengan menggunakan Wireshark yang dijalankan pada Windows. Perlu diperhatikan bahwa implementasi ini dapat berhasil dikarenakan wireshark berjalan di host yang sama dengan tempat dijalankannya komunikasi mqtt.

5. HASIL PENELITIAN

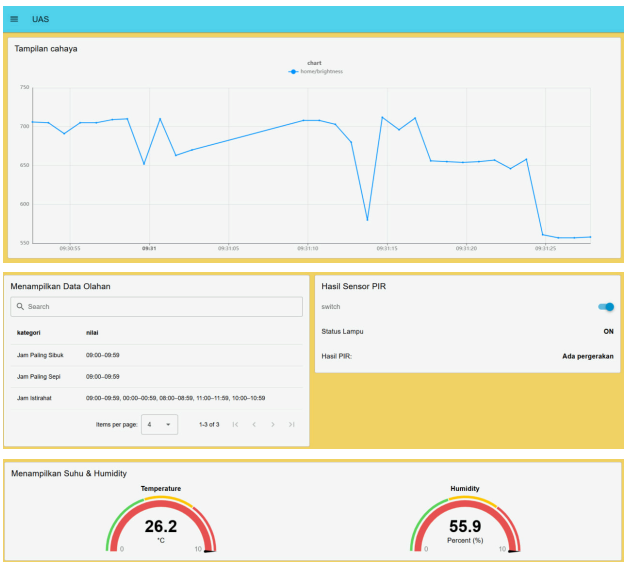
Bab ini menguraikan hasil implementasi dari perancangan sistem yang telah dilakukan. Pembahasan mencakup visualisasi hasil perancangan perangkat keras (hardware) dan perangkat lunak (software), data hasil pengujian, serta analisis terhadap kinerja sistem.

5.1 Hasil Perancangan

Tahap ini merupakan realisasi dari bab perancangan sebelumnya. Hasil rancangan terdiri dari implementasi hardware berupa node sensor dan kendali, serta implementasi software berupa antarmuka dashboard dan sistem basis data.

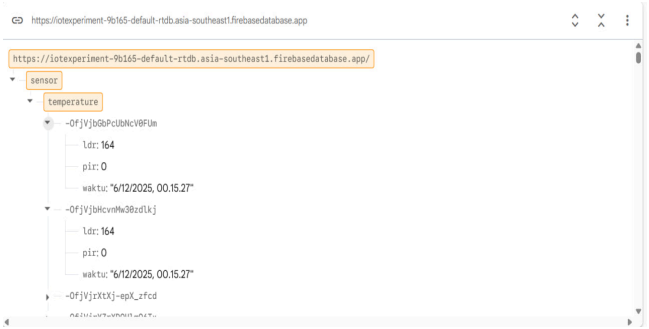
Realisasi hardware melibatkan perakitan mikrokontroler NodeMCU ESP8266 dengan aktuator dan sensor-sensor. Tampilan fisik perangkat keras yang telah dirakit dan beroperasi ditunjukkan pada **Gambar 3.**

Setelah perangkat keras terhubung ke internet, interaksi pengguna dilakukan melalui Dashboard Node-RED. Dashboard ini menampilkan visualisasi data sensor dalam bentuk grafik line chart, serta tombol switch untuk mengaktifkan aktuator. Tampilan antarmuka dashboard yang telah berjalan dapat dilihat pada **Gambar 6.**



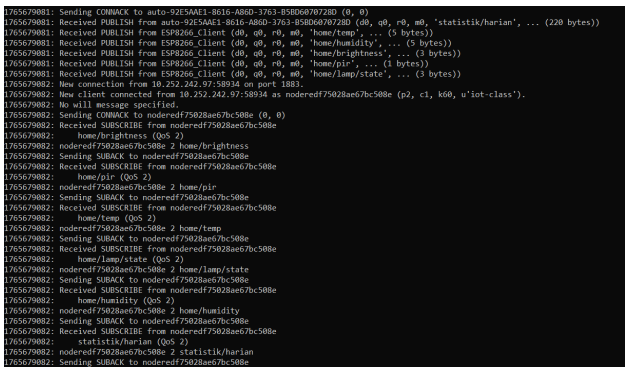
Gambar 6. Dashboard Node-RED

Seluruh data yang dikirimkan oleh NodeMCU melalui protokol MQTT diteruskan oleh Node-RED ke Google Firebase. Data disimpan dalam format JSON dan diperbaharui secara *real-time*. **Gambar 7.** memperlihatkan struktur data yang berhasil terekam pada Firebase saat beroperasi.



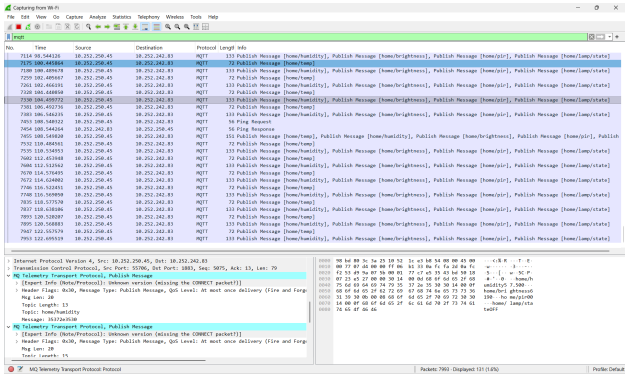
Gambar 7. Struktur data Firebase

Dari sisi keamanan data, dapat terlihat bahwa mosquitto memberikan log dari setiap pengiriman data yang ada baik yang diberikan sensor ataupun yang diberikan python. Dokumentasi mengenai hal ini dapat dilihat pada bagian **Gambar 8.**



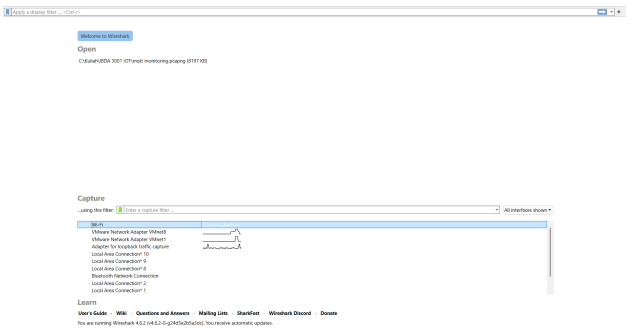
Gambar 8. Monitoring menggunakan Mosquitto

Kemudian dalam melakukan monitoring dengan menggunakan Wireshark dapat dilihat pada **Gambar 9.** bahwa protokol mosquitto yang digunakan masih tidak aman karena bisa dilakukan *packet-sniffing* dengan data yang tidak terenkripsi. Sebagai contoh pada bagian bawah terdapat deretan angka hexadecimal dari pesan yang dikirimkan dan tepat di sebelahnya terdapat pesan asli yang dikirim maupun diterima.



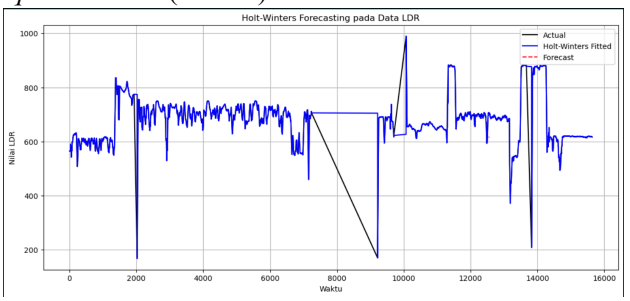
Gambar 9. Monitoring menggunakan Wireshark

Untuk bisa melihat setiap komunikasi yang dilakukan instalasi wireshark pada website resmi wireshark [Wireshark • Go Deep | Download](). Setelah melakukan instalasi, maka penelitian dapat dilanjutkan dengan menjalankan wireshark dengan interface monitoring menggunakan Wi-fi. Kemudian setelah masuk kedalam, monitoring jaringan seperti pada **Gambar 9** pengaturan filter dapat diubah dengan hanya melihat protokol mqtt saja atau tcp.port == 1883. Setelah melakukan semua langkah di atas maka monitoring dari jaringan mosquitto akan bisa dilihat.



Gambar 10. Homepage Wireshark

Selain itu pada sisi statistika deskriptif, data hasil pengolahan dapat dilihat pada **Gambar 6.** terdapat tabel yang menunjukan jam-jam yang paling sibuk, sepi dan istirahat. Terlebih lagi dalam *Modeling Machine Learning* menggunakan *Holt-Winters* dapat dilihat pada **Gambar 10.** bahwa hasil prediksi dari model ini dengan *Root mean squared error* (RMSE) sebesar 10.17.



Gambar 11. Hasil Prediksi *Holt-Winters*

Jika dibandingkan dengan rata-rata data sensor cahaya maka dapat dilihat pada **Gambar 11.** bahwa model cukup baik dalam memprediksi hasil sensor cahaya selama 30 menit kedepan.

$$\text{Relative RMSE} = \frac{10.70}{674} \approx 0.0159 \approx \mathbf{1.6\%}$$

Gambar 12. Relative RMSE

5.2 Pengujian Sistem

Pengujian dilakukan untuk memverifikasi apakah sistem telah berjalan dengan semestinya. Pengujian ini dibagi menjadi dua, yaitu: pengujian fungsional dan pengujian validasi sensor.

Pengujian fungsional bertujuan untuk memastikan bahwa setiap fitur input dan output berjalan sesuai logika program tanpa melihat struktur kode internalnya. Berdasarkan pengujian yang dilakukan terhadap fitur konektivitas, pembacaan data, dan kendali, diperoleh hasil seperti yang tercantum pada **Tabel 2.**

Tabel 2. Hasil Pengujian Fungsional

Skenario Pengujian	Hasil Pengujian
Koneksi Wi-Fi: Menyalakan NodeMCU untuk terhubung ke internet.	**Berhasil.** Lampu indikator menyala dan perangkat mendapatkan *IP address* secara otomatis
Pembacaan sensor: Sistem membaca suhu dari ruangan dan menampilkan pada dashboard Node-RED.	**Berhasil.** Data suhu ditampilkan secara *real-time* pada *dashboard* Node-RED sesuai dengan kondisi ruangan.
Kendali aktuator: Menekan tombol saklar pada dashboard Node-RED.	**Berhasil.** AC dapat dimatikan dan dinyalakan menggunakan tombol saklar pada dashboard Node-RED dengan
Penyimpanan data: Sistem mengirim data log ke cloud server.	**Berhasil.** Data baru tercatat dan muncul pada Google Firebase.
Penanganan putus koneksi: Mematikan sumber internet secara paksa.	**Berhasil.** Dashboard menampilkan status OFF ketika hilangnya internet.
Penggunaan Sistem Keamanan: autentikasi dan monitoring log	**Berhasil.** MQTT dapat memberikan monitor log dari setiap komunikasi yang terjalin selama terhubung.

5.3 Pembahasan

Berdasarkan hasil pengujian fungsional pada Tabel 5.1., dapat dianalisa bahwa sistem otomasi gedung berbasis Internet of Things (IoT) ini telah berhasil diimplementasikan sesuai dengan tujuan perancangan. Analisis kerja sistem dapat dilihat pada poin-poin berikut:

1. Keandalan dalam komunikasi data

Berdasarkan hasil pengujian, komunikasi data antara NodeMCU ESP8266 dan Node-RED melalui MQTT berjalan dengan baik. Ketika NodeMCU dinyalakan, perangkat dapat langsung terhubung ke jaringan Wi-Fi dan MQTT Broker secara otomatis. Hal ini menunjukkan bahwa sistem yang dibangun sudah mampu menghubungkan pengguna dengan perangkat fisik secara lancar.

2. Responsivitas kendali aktuator

Pada kendali AC, sistem merespons perintah sesuai dengan logika yang telah diprogram. Saat tombol saklar pada dashboard ditekan, perintah tersebut langsung dikirim dan diubah menjadi sinyal listrik yang mengaktifkan aktuator untuk menyalakan atau mematikan AC. Hasil ini menunjukkan bahwa Node-RED berfungsi dengan baik sebagai publisher dalam mengirimkan payload perintah ke topik yang di-*subscribe* oleh mikrokontroler.

3. Pemantauan dan integritas data

Sistem pemantauan suhu ruangan menunjukkan hasil yang konsisten. Data yang dibaca dari sensor berhasil ditampilkan secara real-time pada antarmuka pengguna. Selain itu, fitur penyimpanan data ke Google Firebase juga terbukti berjalan dengan baik. Sinkronisasi data yang ditampilkan dengan data yang ada pada database menunjukkan bahwa integritas data terjaga.

4. Ketergantungan terhadap jaringan

Meskipun seluruh fitur telah berjalan dengan baik, hasil pengujian pemutusan koneksi menunjukkan bahwa kinerja sistem ini sangat bergantung pada jaringan internet. Saat koneksi terputus, sistem secara otomatis mengganti status ke OFF pada dashboard. Hal ini merupakan fitur keamanan yang penting agar pengguna mengetahui bahwa data yang tampil bukanlah data yang *real-time* saat internet mati.

5. Sistem Keamanan MQTT

MQTT memberikan fitur monitoring log dan autentikasi username password pada saat ingin melakukan komunikasi baik dari publisher maupun juga dari subscriber.

Secara keseluruhan, sistem ini telah memenuhi spesifikasi kebutuhan fungsional dan layak digunakan sebagai prototipe pemantau ruangan.

6. SARAN DAN KESIMPULAN

Saran

Berdasarkan hasil perancangan dan pengujian sistem, terdapat beberapa saran yang dapat dijadikan acuan untuk pengembangan penelitian selanjutnya.

Pertama, sistem dapat dikembangkan dengan menambahkan mekanisme keamanan komunikasi yang lebih kuat, seperti penggunaan MQTT over TLS/SSL serta penerapan autentikasi berbasis token atau sertifikat, sehingga kerahasiaan dan integritas data lebih terjamin terutama pada implementasi di lingkungan nyata.

Kedua, pengendalian aktuator AC dapat ditingkatkan dengan mengganti metode aktuasi mekanik menjadi pemancar inframerah (IR) atau modul relay pintar, sehingga sistem mampu mengendalikan lebih banyak parameter AC seperti suhu, mode operasi, dan kecepatan kipas secara lebih presisi dan aman. Penggunaan aktuator yang lebih representatif juga akan meningkatkan relevansi sistem terhadap penerapan smart building modern.

Ketiga, sistem monitoring dapat diperluas dengan menambahkan jenis sensor lain, seperti sensor konsumsi daya listrik atau sensor kualitas udara (misalnya MQ atau CO_2 sensor), sehingga informasi yang diperoleh lebih komprehensif dan dapat digunakan untuk analisis efisiensi energi maupun kenyamanan ruangan.

Keempat, pada python dapat dilakukan analisis data yang lebih real time seperti melakukan analisis pada jangka waktu tertentu. Hal ini disebabkan karena adanya sistem analisis yang dilakukan masih bersifat manual atau berdasarkan input user kapan ingin menjalankan kode tersebut.

Terakhir, sistem dapat dikembangkan menjadi arsitektur multi-node dengan menambahkan beberapa perangkat ESP8266 pada ruangan yang berbeda. Pengembangan ini memungkinkan penerapan sistem otomasi gedung dalam skala yang lebih besar serta mendukung konsep smart building yang terdistribusi dan terintegrasi.

Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian yang telah dilakukan, dapat disimpulkan bahwa sistem otomasi gedung berbasis Internet of Things (IoT) menggunakan NodeMCU ESP8266 dan Node-RED berhasil dibangun dan berfungsi sesuai dengan tujuan penelitian. Sistem mampu melakukan pemantauan kondisi lingkungan ruangan secara real-time melalui integrasi sensor PIR, DHT22, dan LDR, serta menampilkan data tersebut secara visual pada dashboard Node-RED.

Implementasi protokol MQTT dengan broker Eclipse Mosquitto terbukti efektif dalam mendukung komunikasi dua arah antara perangkat keras dan perangkat lunak. ESP8266 dapat berperan sebagai *publisher* dan *subscriber* secara simultan, sehingga sistem mampu mengirim data sensor serta menerima perintah kendali aktuator dengan latensi yang rendah dan respons yang stabil. Hal ini menunjukkan bahwa MQTT merupakan solusi komunikasi yang efisien untuk sistem otomasi gedung skala kecil hingga menengah.

Penerapan logika kendali hybrid pada sistem memungkinkan penggabungan antara mode otomatis berbasis sensor dan mode manual berbasis perintah pengguna. Hasil pengujian menunjukkan bahwa sistem dapat menyesuaikan pengoperasian lampu dan AC sesuai dengan kondisi lingkungan maupun instruksi pengguna, sehingga mendukung peningkatan efisiensi energi dan kenyamanan penggunaan ruangan.

Selain itu, integrasi Google Firebase sebagai media penyimpanan data berhasil menyediakan mekanisme pencatatan data sensor secara real-time dan terstruktur dalam format JSON. Data historis yang tersimpan dapat dimanfaatkan untuk analisis lebih lanjut, termasuk analisis aktivitas dan pola penggunaan ruangan. Dari sisi keamanan, penerapan autentikasi username dan password pada broker MQTT serta mekanisme logging telah memberikan perlindungan dasar terhadap akses tidak sah dan memungkinkan pemantauan aktivitas komunikasi data.

Secara keseluruhan, sistem yang dikembangkan telah memenuhi kebutuhan fungsional yang dirancang dan layak digunakan sebagai prototipe otomasi gedung berbasis IoT. Sistem ini menunjukkan bahwa pemanfaatan platform open-source seperti ESP8266, Node-RED, Mosquitto, dan Firebase dapat menghasilkan solusi otomasi gedung yang terjangkau, fleksibel, dan mudah dikembangkan untuk kebutuhan penelitian maupun implementasi awal di lingkungan nyata.

References

[1] A. S. Pramuda, A. W. W. Nugraha, and A. Fadli, "Perancangan Sistem Deteksi Manusia Menggunakan Sensor PIR, RCWL, dan Infrared Pada Sistem Manajemen Lampu Gedung Berbasis Internet of Things," *Jurnal Pendidikan dan*

Teknologi Indonesia (JPTI), vol. 3, no. 1, pp. 1-11, 2023.

[2] R. U. M. Raharja, A. Pudoli, and D. Kusumaningsih, "Prototype Smart Home Berbasis IoT dengan NodeMCU ESP8266, Motor Servo dan Sensor Suhu DHT11 Berbasis Web," SKANIKA: Sistem Komputer dan Teknik Informatika, vol. 5, no. 2, pp. 265–274, 2022.

[3] OpenJS Foundation, "Node-RED: A visual tool for wiring the Internet of Things," nodered.org, 2024. [Online]. Available: https://nodered.org/

[4] A. A. Prasetyo, R. Arifin, and R. I. Vidyastri, "Sistem otomatisasi AC berbasis IoT untuk pengendalian suhu ruangan secara optimal," in *SinarFe7: Seminar Nasional FORTEI Regional 7*, vol. 6, no. 1, pp. 270–280, 2025.

[5] PlatformIO, "What is PlatformIO?", PlatformIO Documentation. [Online]. Available: https://docs.platformio.org/en/latest/what-is-platformio.html

[6] H. A. Al-agele et al., "Low-cost composite autosampler for wastewater sampling," *Sensors (Basel)*, 2025. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11925591/

[7] PlatformIO, "Espressif 8266 — PlatformIO," 2024. [Online]. Available: https://docs.platformio.org/en/latest/platforms/espressif8266.html

[8] Google Developers, "Firebase Realtime Database," firebase.google.com, 2023. [Online]. Available: https://firebase.google.com/docs/database.

[9] W.-J. Li, C.-M. Yen, Y.-S. Lin, S.-C. Tung, and S.-M. Huang, "JustIoT: Internet of Things based on the Firebase Real-time Database," in Proc. 2018 Int. Symp. on Semiconductor Manufacturing Intelligence (ISMI), 2018, pp. (lihat prosiding), doi:10.1109/SMILE.2018.8353979.

[10] R. Irsyada, M. A. Haq, N. A. Rohmah, P. A. H. Saputra, and R. Jannah, "Implementasi NodeMCU ESP8266 dan Sensor Cahaya Pada Lampu Berbasis Internet of Things," *Jurnal Ilmiah Sistem Informasi dan Ilmu Komputer*, vol. 2, no. 1, pp. 22–32, 2022.

[11] M. A. Raharjo and F. Sabur, "Perancangan System Smart Office Berbasis Internet of Things Politeknik Penerbangan Makassar," *Airman: Jurnal Teknik dan Keselamatan Transportasi*, vol. 3, no. 2, pp. 1–?, 2020.

[12] E. D. Kusuma, O. I. Tarunay, L. Lunardi, and Andika. "Prototipe Sistem Otomasi Smart Office dengan Menggunakan Lock Door, Motion Sensor, dan LCD Berbasis Arduino UNO," Rubinstein, 2023

[13] F. Rahm and H. H. Do, "Data Cleaning: Problems and Current Approaches," *IEEE Data Engineering Bulletin*, vol. 23, no. 4, pp. 3–13, 2000.

[14] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed., San Francisco, CA, USA: Morgan Kaufmann, 2012.

[15] D. C. Montgomery, *Applied Statistics and Probability for Engineers*, 6th ed., Hoboken, NJ, USA: John Wiley & Sons, 2014.

[16] ISO/IEC, *ISO/IEC 27002:2013 — Information technology — Security techniques — Code of practice for information security controls*, Geneva, Switzerland: ISO, 2013.

[17] K. Kent and M. Souppaya, *Guide to Computer Security Log Management*, NIST Special Publication 800-92, Gaithersburg, MD, USA: National Institute of Standards and Technology, 2006.

[18] Banks and R. Gupta, MQTT Version 3.1.1, OASIS Standard, Oct. 2014. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[19] G. Combs *et al.*, *Wireshark User's Guide*, Wireshark Foundation, 2024. [Online]. Available: https://www.wireshark.org/docs/wsug_html/

[20] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 3rd ed., OTexts, 2021. [Online]. Available: https://otexts.com/fpp3/holt-winters.html

Lampiran kode: python-statistika deskriptif, machine-learning, node-red, esp8266, video demo, ppt presentasi dan lainnya

- https://www.canva.com/design/DAG7khez9qI/PEiEIMZCgyJdA5JnrZGifA/edit?utm_content=DAG7khez9qI&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton
- https://console.firebase.google.com/u/0/project/iotexperiment-9b165/database/iotexperiment-9b165-default-rtdb/data/~2F
- https://drive.google.com/drive/folders/1N1NmnaVjUe8H3eWKSFP7Idv59UeIjWKf?usp=sharing

Komitmen Integritas

"Di hadapan TUHAN yang hidup, saya menegaskan bahwa saya tidak memberikan maupun menerima bantuan apapun—baik lisan, tulisan, maupun elektronik—di dalam ujian ini selain daripada apa yang telah diizinkan oleh pengajar, dan tidak akan menyebarkan baik soal maupun jawaban ujian kepada pihak lain."

Keanrich Cordana - 232301226

Matther Layadi - 232203068

Laurentius Santiago Pratama - 232200410