

ANALISIS KINERJA PROTOKOL COAP DAN MQTT PADA SISTEM RUMAH CERDAS



Disusun Oleh:

N a m a : Ayudhia Surya Taufiq Rahma
NIM : 20523220

**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2025**

HALAMAN PENGESAHAN DOSEN PEMBIMBING

**ANALISIS KINERJA PROTOKOL COAP DAN MQTT PADA
SISTEM RUMAH CERDAS**

TUGAS AKHIR



N a m a : Ayudhia Surya Taufiq Rahma
NIM : 20523220

Yogyakarta, 27 Mei 2025

Pembimbing,



(Kurniawan Dwi Irianto, S.T., M.SC.)

HALAMAN PENGESAHAN DOSEN PENGUJI

ANALISIS KINERJA PROTOKOL COAP DAN MQTT PADA SISTEM RUMAH CERDAS

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 27 Mei 2025

Tim Penguji

Kurniawan Dwi Irianto, S.T., M.SC.

Anggota 1

Sheila Nurul Huda, S.Kom., M.Cs.

Anggota 2

Kholid Haryono, S.T., M.Kom.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia



(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Ayudhia Surya Taufiq Rahma

NIM : 20523220

Tugas akhir dengan judul:

ANALISIS KINERJA PROTOKOL COAP DAN MQTT PADA SISTEM RUMAH CERDAS

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 27 Mei 2025



(Ayudhia Surya Taufiq Rahma)

HALAMAN PERSEMBAHAN

Dengan penuh rasa syukur, laporan tugas akhir ini penulis persembahkan pertama kepada diri penulis sendiri karena akhirnya mampu berjuang menyelesaikan tugas akhir dengan penuh kesabaran hingga percaya pada kemampuan diri sendiri. Terima kasih kepada keluarga terutama kedua orang tua yang selalu menjadi sumber inspirasi dan kekuatan bagi penulis dan tentunya masku tercinta yang senantiasa memberikan dukungan untuk penulis agar segera menyelesaikan tugas akhirnya. Begitu juga kepada dosen pembimbing yang dengan penuh kesabaran dalam memberikan arahan serta ilmu yang sangat berharga kepada penulis. Selain itu, sebagai bentuk ungkapan terima kasih laporan tugas akhir ini dibuat dan dipersembahkan kepada Jurusan Informatika yang telah menjadi wadah bagi penulis untuk belajar dan berkembang.

HALAMAN MOTO

“Allah tidak membebani seseorang melainkan sesuai kesanggupannya”

(QS Al-Baqarah: 286)

“Sesungguhnya Allah tidak akan mengubah keadaan suatu kaum, sebelum mereka mengubah keadaan diri mereka sendiri.”

(QS Ar Rad 11)

"Cobaan hidupmu bukanlah untuk menguji kekuatan dirimu. Tapi menakar seberapa besar kesungguhan dalam memohon pertolongan kepada Allah." - Ibnu Qoyyim

"Kesuksesan tidak diukur dari seberapa sering Anda jatuh, tetapi seberapa sering Anda bangkit kembali." - Vince Lombardi

“Kamu tidak harus menjadi hebat untuk memulai, tetapi kamu harus mulai untuk menjadi hebat.” – Zig Ziglar

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Assalamu'alaikum warahmatullahi wabarakatuh

Alhamdulillah Rabbil 'alamin. Puji syukur penulis ucapkan kepada Allah SWT yang telah mengaruniakan segala rahmat dan karunia-Nya kepada penulis sehingga penulis dapat menyusun serta menyelesaikan skripsi ini dengan judul “Analisis Kinerja Protokol COAP dan MQTT Pada Sistem Rumah Cerdas” ini dengan baik. Shalawat serta salam semoga senantiasa terlimpah kepada Baginda Nabi Muhammad SAW, guru serta teladan bagi seluruh umat, juga kepada keluarga, para sahabat, dan InsyaAllah kepada kita semua selaku umatnya. Skripsi ini disusun sebagai salah satu syarat sidang skripsi untuk menyelesaikan studi pada Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.

Dalam penyusunan skripsi ini, penulis menyadari bahwa dengan bantuan, bimbingan dan dukungan pada penulis dapat terselesaikan dengan baik. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Allah SWT, atas segala nikmat, kesehatan, serta petunjuk-Nya yang telah diberikan sehingga penulis dapat melalui setiap proses penyusunan tugas akhir ini.
2. Kedua orang tua tercinta, Bapak Suryanto dan Ibu Tatik Amaroh, yang selalu memberikan dukungan, kasih sayang, serta doa tanpa henti kepada penulis.
3. Kakak penulis Adib Surya Mu'afian yang senantiasa memberikan semangat, dukungan dan pengingat sehingga penulis mampu menyelesaikan tugas akhir.
4. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku Ketua Program Studi Informatika Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
5. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku dosen pembimbing akademik yang telah mendampingi serta memberikan arahan dan nasehat selama masa studi di Program Studi Informatika.
6. Bapak Kurniawan Dwi Irianto, S.T., M.SC. selaku dosen pembimbing tugas akhir yang telah memberikan bimbingan, masukan, serta dukungan dengan sabar selama proses penyusunan tugas akhir ini.
7. Bapak – Ibu dosen di Program Studi Informatika, yang telah membekali penulis dengan ilmu dan pengalaman luar biasa selama masa perkuliahan.
8. Sahabat penulis, Annisa Yumna Azzahro, yang senantiasa menemani dan memberikan dukungan kepada penulis.

9. Para sahabat dari SMP, Umi Qultsum Malikha, Safira Nurulhaq, Maghfiroh Nurul Inayatulloh, Mutia Alifah, dan Zahrah Mufidah, yang selalu memberikan dukungan, semangat, serta doa dalam penyelesaian penulisan tugas akhir penulis.
10. Teman-teman penulis dalam grup whatsapp "*Calonmu mas*" yang selalu mengingatkan, memberikan semangat dan keceriaan di setiap langkah perjalanan penulis.
11. Teman dekat SMA penulis di grup WhatsApp "*L for Lesti (Lesgo besti)*" yang selalu berbagi cerita, tawa, dan dukungan sehingga membangun semangat penulis untuk menyelesaikan tugas akhir.
12. Teman-teman penulis yang ada di grup whatsapp "*An_ten squad*" yang telah menjadi tempat berbagi keluh kesah, keceriaan, dan semangat bagi penulis.
13. Teman-teman seperjuangan dalam grup whatsapp "*Dudidudidam*", Fionna Mawaddah Indrasari dan Amanda Githa Oktaviani yang selalu memberikan dorongan, semangat, dan kebersamaan yang sangat berarti.
14. Rekan-rekan KKN yang sampai saat ini masih berkomunikasi dan senantiasa mendukung penulis agar mampu menyelesaikan tugas akhirnya.
15. Teruntuk penulis sendiri, yang telah berjuang keras, melewati setiap tantangan, dan tetap bertahan hingga tugas akhir ini selesai. Semoga ini menjadi langkah awal untuk pencapaian-pencapaian yang lebih besar di masa depan.
16. Serta semua pihak yang penulis tidak dapat sebutkan satu per satu, yang telah memberikan bantuan dan dukungan dalam berbagai bentuk selama proses penyusunan tugas akhir.

Semoga segala dukungan, semangat, dan doa yang diberikan kepada penulis mendapatkan balasan yang terbaik oleh Allah SWT. Penulis menyadari bahwasannya penulisan tugas akhir ini masih belum sempurna. Maka dari itu, penulis mengharapkan kritik dan saran yang bersifat membangun untuk penulisan tugas akhir yang lebih baik. Penulis berharap, semoga hasil laporan tugas akhir ini dapat bermanfaat bagi semua pihak, baik akademis maupun pihak yang membacanya. Aamiin

Yogyakarta, 27 Mei 2025



(Ayudhia Surya Taufiq Rahma)

SARI

Bagian laporan tugas akhir ini berjudul “Analisis Kinerja Protokol CoAP dan MQTT pada Sistem Rumah Cerdas”. Penelitian ini bertujuan untuk menganalisis dan membandingkan kinerja dua protokol komunikasi, yaitu CoAP dan MQTT, dalam penerapan sistem rumah cerdas. Pengujian dilakukan dengan mengukur beberapa parameter kinerja seperti *packet loss*, *delay*, dan *throughput* pada perangkat-perangkat yang terhubung dalam sistem IoT. Sistem rumah cerdas yang digunakan dalam penelitian ini terdiri dari perangkat-perangkat seperti sensor DHT11 yang terhubung dengan mikrokontroler ESP32, serta Raspberry Pi yang berfungsi sebagai server untuk mengolah dan mendistribusikan data. Hasil penelitian menunjukkan bahwa meskipun MQTT lebih efisien dalam hal throughput, protokol CoAP lebih unggul dalam penghematan energi dan memiliki latensi yang lebih rendah. Penelitian ini memberikan rekomendasi mengenai pemilihan protokol yang lebih sesuai untuk sistem rumah cerdas berdasarkan analisis kinerja yang dilakukan.

Kata kunci: Protokol CoAP, Protokol MQTT, Sistem Rumah Cerdas, Packet Loss, Delay, Throughput.

GLOSARIUM

Glosarium memuat daftar kata tertentu yang digunakan dalam laporan dan membutuhkan penjelasan, misalnya kata serapan yang belum lazim digunakan. Urutkan sesuai abjad. Contoh penulisannya seperti di bawah ini:

API	<i>Application Programming Interface</i> , sekumpulan aturan yang memungkinkan perangkat lunak untuk berkomunikasi dengan perangkat lunak lain.
Internet of Things	Teknologi yang menghubungkan objek fisik seperti sensor, perangkat rumah tangga, atau perangkat lainnya ke internet untuk memungkinkan pengendalian dan pemantauan jarak jauh.
ESP32	Modul mikrokontroler yang digunakan dalam sistem IoT untuk menghubungkan perangkat dengan jaringan dan mengolah data dari sensor.
DHT11	Sensor yang digunakan untuk mengukur suhu dan kelembapan.
CoAP	<i>Constrained Application Protocol</i> , protokol komunikasi yang dirancang menggunakan model request-response untuk pertukaran data.
MQTT	<i>Message Queuing Telemetry Transport</i> , protokol komunikasi berbasis publish/subscribe yang digunakan dalam aplikasi IoT untuk mentransfer pesan antar perangkat dengan efisiensi tinggi.
QoS	<i>Quality of Service</i> , parameter yang digunakan untuk mengukur kualitas layanan dalam komunikasi jaringan.
Gateway	Perangkat yang berfungsi sebagai penghubung antara dua jaringan, seperti menghubungkan perangkat IoT di rumah pintar dengan server untuk pemantauan jarak jauh.
Packet Loss	Hilangnya paket data selama transmisi dalam jaringan yang dapat memengaruhi kinerja komunikasi antar perangkat dalam sistem IoT.
Delay	Waktu yang dibutuhkan untuk pengiriman data dari pengirim ke penerima.
Troughput	Jumlah data yang berhasil dikirim atau diterima dalam satuan waktu tertentu, biasanya diukur dalam bit per second (bps), yang mencerminkan kinerja suatu jaringan atau protokol komunikasi.

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN DOSEN PEMBIMBING	ii
HALAMAN PENGESAHAN DOSEN PENGUJI	iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR.....	iv
HALAMAN PERSEMBAHAN	v
HALAMAN MOTO.....	vi
KATA PENGANTAR.....	vii
SARI.....	ix
GLOSARIUM	x
DAFTAR ISI	xi
DAFTAR TABEL	xiii
DAFTAR GAMBAR.....	xiv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah.....	3
1.3 Batasan Masalah	3
1.4 Tujuan Penelitian	3
1.5 Manfaat Penelitian	4
1.6 Metode Penelitian	4
1.7 Sistematika Penulisan	5
BAB II LANDASAN TEORI	6
2.1 Dasar Teori.....	6
2.1.1 Internet of Things	6
2.1.2 Smart Home	7
2.1.3 Protokol CoAP	7
2.1.4 Protokol MQTT.....	9
2.1.5 ESP32/ESP8266.....	12
2.1.6 Raspberry PI.....	14
2.1.7 Sensor DHT11.....	15
2.1.8 Arduino IDE.....	16
2.2 Penelitian Terdahulu	18
BAB III METODOLOGI PENELITIAN	27
3.1 Studi Literatur	27
3.2 Analisis Kebutuhan.....	28
3.2.1 Analisis Kebutuhan Fungsional	28
3.2.2 Analisis Kebutuhan Non-Fungsional	28
3.3 Rancangan Sistem.....	30
3.3.1 Alur Kerja Sistem.....	30
3.3.2 Desain Perakitan Alat.....	33
3.3.3 Perakitan.....	35
3.4 Implementasi Protokol	35
3.5 Analisis Hasil Pengukuran Kinerja dan Evaluasi Kinerja	36
BAB IV HASIL DAN PEMBAHASAN.....	37
4.1 Hasil Pengujian Sistem	37
4.2 Pembahasan Hasil Pengujian	42
BAB V KESIMPULAN DAN SARAN	61
5.1 Kesimpulan	61

5.2 Saran	62
DAFTAR PUSTAKA.....	63

DAFTAR TABEL

Tabel 2.1 Jenis-jenis pesan MQTT	11
Tabel 2.2 Ringkasan dari protokol-protokol Iot, CoAP dan MQTT.....	12
Tabel 2.3 Contoh tabel yang dibuat menggunakan MS Word.....	18
Tabel 3.1 Kebutuhan Non-Fungsional Hardware	28
Tabel 3.2 Kebutuhan Non-Fungsional Software	29
Tabel 4.1 Ringkasan Perbandingan Hasil Pengujian	41

DAFTAR GAMBAR

Gambar 2.1 Model komunikasi menggunakan CoAP	8
Gambar 2.2 Model komunikasi menggunakan MQTT	10
Gambar 2.3 ESP32	14
Gambar 2.4 Raspberry Pi	15
Gambar 2.5 Sensor DHT11	16
Gambar 2.6 Arduino IDE	18
Gambar 3.1 Diagram alir penelitian	27
Gambar 3.2 Skema alur kerja sistem	31
Gambar 3.3 Diagram alir alur kerja sistem	32
Gambar 3.4 Diagram alir alur kerja sistem	33
Gambar 3.5 Rancangan desain rakitan alat	34
Gambar 3.6 Rakitan alat	35
Gambar 3.7 Implementasi protokol	36
Gambar 4.1 Hasil pengujian DHT 11	38
Gambar 4.2 Hasil perbandingan packet loss	39
Gambar 4.3 Hasil perbandingan delay	40
Gambar 4.5 Hasil perbandingan throughput	41
Gambar 4.6 Hasil kode program MQTT client	49
Gambar 4.7 Hasil kode program MQTT server	51
Gambar 4.8 Hasil kode program CoAP client	58
Gambar 4.9 Hasil kode program CoAP server	60

BAB I

PENDAHULUAN

1.1 Latar Belakang

Munculnya IoT (Internet of Things) merupakan salah satu pendorong utama transformasi teknologi modern saat ini. Contoh yang dapat diambil yaitu seperti perangkat pintar yang memungkinkan berkomunikasi dan berkoordinasi satu sama lain. Komunikasi antar perangkat pintar melalui jaringan merepresentasikan konsep IoT. Konsep tersebut bertujuan untuk memperluas manfaat koneksi internet yang selalu aktif. IoT dapat digambarkan sebagai koneksi objek seperti ponsel pintar, TV internet, sensor, dan aktuator ke internet, di mana perangkat-perangkat ini digabungkan dalam bentuk baru yang memungkinkan komunikasi antar manusia dan objek (Wicaksono, 2018). Transformasi ini juga tercermin dalam pengembangan rumah pintar yang menjadi fokus utama dalam memanfaatkan potensi IoT.

Definisi rumah pintar adalah tempat tinggal berbasis teknologi yang terpusat pada satu tempat di mana peralatan rumah tangga tertentu dapat diatur menggunakan sistem sehingga memungkinkan mereka untuk dikendalikan dari jarak jauh atau dekat (Wijayanti, 2022). Sistem ini, telah menjadi fokus pengembangan teknologi modern, memberikan solusi terintegrasi untuk meningkatkan kenyamanan, penghematan energi, dan keamanan pada rumah (YALÇINKAYA et al., 2020). Sistem rumah pintar (*smart home*) merupakan suatu lingkungan rumah yang terhubung secara digital, di mana perangkat-perangkat elektronik dan teknologi informasi diintegrasikan, menciptakan ekosistem yang cerdas dan terhubung.

Membahas mengenai rumah pintar dan semakin berkembangnya teknologi, semakin banyak orang yang memilih menggunakan sistem rumah pintar karena berbagai alasan. Pertama, rumah pintar menawarkan tingkat kenyamanan yang tinggi dengan memungkinkan pengguna dapat memantau serta mengelola perangkat rumah mereka secara jarak jauh melalui perangkat pintar, seperti tablet atau smartphone. Selain itu, fitur otomatisasi membuat kegiatan sehari-hari menjadi lebih efisien, seperti menyesuaikan suhu atau pencahayaan berdasarkan dengan jadwal dan preferensi pemilik rumah.

Aspek penghematan energi juga menjadi dorongan besar, karena rumah pintar memungkinkan pemantauan dan pengendalian yang lebih baik terhadap penggunaan energi sehingga menghasilkan potensi penghematan biaya dan dampak positif terhadap lingkungan. Tidak hanya itu saja, keamanan menjadi faktor krusial karena dengan adanya sistem keamanan

pintar yang melibatkan kamera, sensor gerak, dan pengunci pintu pintar. Dengan tujuan, memberikan tingkat perlindungan yang lebih baik untuk rumah dan barang-barang berharga di dalamnya.

Fleksibilitas adalah keuntungan lainnya karena rumah pintar memungkinkan integrasi berbagai perangkat untuk bekerja sama. Alhasil dapat menciptakan ekosistem terhubung yang bekerja secara serempak. Dengan penerapan teknologi ini, pengguna tidak hanya mendapatkan kendali yang lebih besar atas rumah mereka tetapi juga mengalami perubahan dalam kehidupan sehari-hari mereka menjadi lebih efisien, aman, dan terhubung.

Dari keuntungan penggunaan sistem rumah pintar di atas, terdapat beberapa aspek penting yang perlu dipertimbangkan, salah satunya adalah pemilihan protokol komunikasi yang sesuai. Seiring berkembangnya teknologi rumah pintar, berbagai protokol komunikasi seperti CoAP (Constrained Application Protocol) dan MQTT (Message Queuing Telemetry Transport) telah digunakan secara luas. Setiap protokol memiliki karakteristik dan performa yang berbeda-beda, sehingga perlu dilakukan analisis kinerja untuk memastikan efisiensi dan keandalan dalam pertukaran data antar perangkat. Pemilihan protokol yang tepat sangat penting agar sistem dapat berfungsi secara optimal terutama dalam hal kecepatan respons, keandalan transmisi, dan efisiensi jaringan.

Tantangan lain yang juga menjadi pertimbangan adalah efisiensi energi dan keterbatasan sumber daya perangkat. Protokol yang berbeda dapat memberikan beban jaringan dan konsumsi daya yang berbeda pula, yang mempengaruhi performa sistem secara keseluruhan. Oleh karena itu, dalam konteks rumah pintar yang umumnya menggunakan perangkat dengan sumber daya terbatas, perlu dilakukan evaluasi terhadap kinerja protokol komunikasi yang digunakan. Penelitian ini berfokus pada analisis parameter kinerja seperti delay, packet loss, dan throughput untuk memberikan gambaran menyeluruh mengenai efisiensi protokol CoAP dan MQTT dalam mendukung sistem rumah cerdas.

Terdapat beberapa jurnal yang telah menganalisis kinerja protokol MQTT serta melakukan perbandingan dengan protokol lain salah satunya seperti XMPP (Kusuma, Munadi, & Irawan, 2019). Selain itu, pada salah satu penelitian internasional yang telah dilakukan oleh Fikret Yalçınkaya dan temannya mengenai “Testbed Rumah Pintar berbasis IoT menggunakan Protokol Komunikasi MQTT” hasil kesimpulan menunjukkan bahwa dalam penelitian rumah pintar berbasis IoT menggunakan protokol MQTT berhasil diimplementasikan dengan baik. Mekanisme yang dikembangkan berhasil beroperasi tanpa masalah dalam berbagai skenario

pengujian, kecuali pada kasus tertentu di mana terdapat penundaan fungsi kontrol saat aplikasi seluler terhubung ke jaringan.

Meskipun terdapat masalah penundaan, aplikasi berhasil lulus semua tes dalam skenario lain, menunjukkan kecocokan protokol MQTT untuk sistem rumah pintar IoT. Dengan perbaikan pada aplikasi mobile, masalah yang diidentifikasi dapat diatasi. Sebagai saran untuk penelitian selanjutnya, membandingkan protokol MQTT dengan CoAP dan melakukan penskalaan sistem untuk memahami kinerjanya dalam skala yang lebih besar. (Yalçinkaya, Aydılek, Erten, & İnanç, 2020)

Dengan memahami latar belakang ini, penelitian dan pengembangan protokol rumah pintar saat ini perlu dikaji lebih lanjut. Analisis kinerja akan mencakup parameter penting seperti delay dan packet loss. Pemahaman menyeluruh tentang bagaimana kedua protokol ini beroperasi di lingkungan rumah pintar dapat memberikan panduan berharga bagi pengembang dan perancang sistem. Dapat memberikan informasi dasar untuk pengambilan keputusan mengenai pemilihan protokol komunikasi yang sesuai dengan kebutuhan spesifik pengembangan sistem rumah pintar menjadi tujuan utama dari penelitian ini. Dengan demikian, penggunaan protokol pada sistem rumah pintar dapat lebih diandalkan, aman, dan efisien dalam penggunaannya.

1.2 Rumusan Masalah

- a. Bagaimana perbandingan kinerja protokol CoAP dan MQTT pada sistem rumah pintar?

1.3 Batasan Masalah

Batasan masalah dalam melakukan penelitian ini guna menghindari perluasan pembahasan diantaranya, node sensor dan aktuator menggunakan perangkat IoT sederhana berbasis mikrokontroler yaitu ESP32. Selama proses pengerjaan penelitian protokol komunikasi dibatasi hanya protokol CoAP dan MQTT saja. Selanjutnya, parameter kinerja yang diukur hanya packet loss, delay, dan throughput. Tidak dilakukan implementasi atau pengembangan sistem maupun produk rumah cerdas.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini pertama adalah mengetahui perbandingan kinerja protokol CoAP dan MQTT pada sistem rumah pintar ditinjau dari parameter kinerjanya seperti delay dan packet loss. Selain itu, juga dapat menentukan protokol mana yang lebih optimal untuk

diterapkan pada sistem rumah pintar berdasarkan analisis kinerjanya. Dengan memahami perbandingan kedua protokol ini, penelitian ini bertujuan untuk memandu pengembang sistem rumah pintar dalam memilih protokol komunikasi yang paling sesuai dengan kebutuhan spesifik mereka.

1.5 Manfaat Penelitian

Dari tujuan tersebut harapannya dengan melakukan penelitian ini dapat memberikan manfaat diantaranya:

1. Mengetahui perbandingan antara kedua protokol yaitu MQTT dan CoAP pada penerapan sistem rumah pintar.
2. Memberikan informasi serta rekomendasi protokol komunikasi IoT yang sesuai dan lebih efisien untuk diterapkan pada sistem rumah pintar.
3. Memperkaya kajian akademis dan menjadi referensi untuk penelitian lebih lanjut mengenai penerapan protokol-protokol IoT pada sistem rumah pintar.

1.6 Metode Penelitian

Untuk mencapai tujuan dalam penelitian ini, terdapat beberapa metode yang diperlukan diantaranya:

1. Studi literatur yang bertujuan untuk mengumpulkan data-data serta referensi yang dibutuhkan dari buku, jurnal ataupun makalah. Dengan begitu, dalam melakukan penelitian memiliki dasar atau acuan yang dapat digunakan untuk mengambil keputusan serta langkah selanjutnya.
2. Konsultasi bersama Dosen Pembimbing terkait hasil yang didapat serta langkah-langkah yang akan dilakukan selama proses penelitian.
3. Mengumpulkan serta menyiapkan alat-alat yang dibutuhkan selama proses penelitian.
4. Mempelajari penggunaan dan cara merangkai alat yang digunakan serta hal-hal lain yang dibutuhkan dalam melakukan penelitian.
5. Melakukan perancangan sistem dan diimplementasikan berdasarkan teori yang telah didapatkan.
6. Melakukan pengujian serta analisis perbandingan dua protokol sistem melalui percobaan berdasarkan kelebihan dan kekurangan protokol-protokol tersebut.

1.7 Sistematika Penulisan

Untuk mempermudah dalam membaca dan memahami serta dapat memberikan gambaran yang jelas mengenai permasalahan yang akan dibahas, maka dibuatlah sistematika penulisan yang dibagi ke dalam beberapa bab. Adapun sistematika penulisan pada laporan tugas akhir ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini merupakan awalan dari laporan tugas akhir yang berisi mengenai latar belakang permasalahan yang mendorong dilakukannya penelitian, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian yang digunakan, dan sistematika penulisan.

BAB II LANDASAN TEORI

Bab ini berisi tentang landasan teori yang berhubungan dengan penelitian ini. Diantaranya berupa konsep serta sistem operasi yang digunakan, protokol MQTT dan CoAP, dan lain sebagainya.

BAB III METODOLOGI

Bab ini menjelaskan langkah-langkah yang diambil dalam melakukan proses penelitian. Diawali dengan merancang sistem yang telah dianalisis berupa instalasi dan konfigurasi, dilanjutkan dengan pengujian sistem yang telah dirancang.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini berisi mengenai hasil pengujian berdasarkan penelitian yang telah dilakukan. Bab ini juga memberikan hasil analisis perbandingan kinerja protokol sistem rumah cerdas dari data yang diperoleh pada saat proses pengujian.

BAB V PENUTUP

Bab ini merupakan bagian akhir dari laporan tugas akhir yang berupa kesimpulan yang diperoleh dari hasil analisis kinerja sebelumnya. Selain itu, terdapat saran yang sebaiknya dipertimbangkan berdasarkan keterbatasan dan asumsi yang diidentifikasi selama proses penelitian.

BAB II

LANDASAN TEORI

2.1 Dasar Teori

2.1.1 Internet of Things

Internet of Things (IoT) adalah paradigma komputasi dan komunikasi di mana benda-benda dalam kehidupan sehari-hari terhubung ke Internet. Sistem komunikasi antar mesin, yang sering disebut sebagai komunikasi M2M (Machine-to-Machine), menjadi elemen kunci dalam menghubungkan perangkat pintar di dalam rumah dan menciptakan apa yang dikenal sebagai Smart Home. Dalam konteks IoT, protokol pesan yang dirancang khusus untuk komunikasi M2M dalam aplikasi IoT memegang peran penting dalam menentukan seberapa baik komunikasi M2M dapat berfungsi (Naik & Bapat, 2020). Maka dari itu, protokol pesan dalam konteks Smart Home merupakan tulang punggung dari IoT yang memungkinkan perangkat untuk saling berkomunikasi dan berinteraksi.

IoT tidak hanya menghubungkan objek sehari-hari ke internet, tetapi juga menciptakan ekosistem yang menjadi dasar bagi pengembangan sistem rumah pintar. Konsep ini muncul melalui integrasi seamless antara perangkat elektronik dan internet. Di mana dalam konsep ini dapat memberikan kemampuan kepada penghuni rumah untuk mengendalikan segala sesuatu, mulai dari pencahayaan hingga keamanan melalui sentuhan ujung jari (Gomez et al., 2019). Pentingnya protokol pesan dalam konteks ini menjadi semakin nyata, memastikan bahwa komunikasi antar perangkat pintar dalam Smart Home berjalan efisien dan andal. Sehingga dalam implementasiannya, IoT sangat erat kaitannya dengan protokol pesan seperti MQTT dan CoAP yang telah dikembangkan untuk memenuhi kebutuhan komunikasi M2M dengan efisien.

Konsep IoT ini dikenalkan oleh Kevin Ashton pada tahun 1999, yang sejak saat itu telah berkembang dan diterapkan dalam berbagai aplikasi seperti transportasi cerdas, pertanian, dan kedokteran (Hosny et al., 2023). Pada masa kini, IoT tidak hanya menjadi solusi inovatif untuk meningkatkan kenyamanan di rumah, tetapi juga menjadi fondasi untuk perubahan lebih lanjut di berbagai sektor kehidupan sehari-hari. Perkembangan teknologi IoT memberikan harapan untuk masa depan yang semakin terhubung dan cerdas sehingga membawa manfaat besar dalam berbagai aspek kehidupan.

2.1.2 Smart Home

Rumah merupakan tempat yang istimewa bagi setiap individu, menjadi tempat untuk beristirahat dan melepas penat setelah menjalani berbagai aktivitas di luar. Kenyamanan dan keamanan rumah menjadi aspek utama yang diinginkan oleh setiap penghuni. Maka dari itu, rumah menjadi alasan salah satu tempat yang cocok untuk mengaplikasikan sistem tersebut. Selain itu, rumah juga merupakan lingkungan yang dapat dikendalikan oleh pemiliknya, sehingga mereka bisa mengeluarkan biaya untuk memasang dan merawat teknologi yang ada di dalamnya (Gomez et al., 2019).

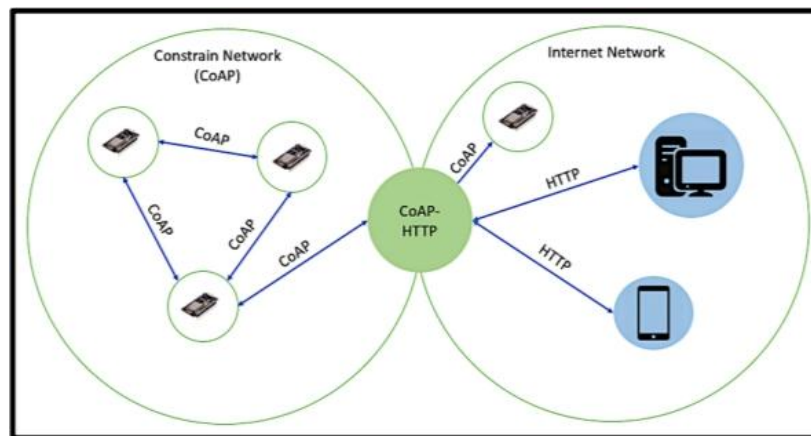
Konsep smart home, atau rumah cerdas, hadir sebagai solusi untuk memenuhi kebutuhan tersebut dengan menawarkan sistem yang memungkinkan pengaturan dan pemantauan berbagai perangkat elektronik di dalam rumah melalui konektivitas internet (Windesi et al., 2022; YALÇINKAYA et al., 2020). Sistem ini dirancang untuk meningkatkan kenyamanan, keamanan, dan efisiensi energi penghuni rumah dengan memungkinkan pengendalian perangkat secara otomatis dan jarak jauh menggunakan smartphone, tablet, atau perangkat lain yang terhubung ke internet (Pratama et al., 2018). Konsep rumah cerdas ini telah mengalami perkembangan pesat dalam beberapa tahun terakhir, terutama dengan adanya teknologi Internet of Things (IoT). Melalui teknologi ini, perangkat-perangkat dalam rumah dapat saling berkomunikasi dan bekerja sama untuk menciptakan lingkungan yang lebih efisien dan user-friendly.

Dengan kemajuan teknologi smart home, muncul tantangan baru terkait integrasi dan kesesuaian antara berbagai perangkat. Untuk itu, dibutuhkan protokol komunikasi yang dapat menyatukan berbagai standar, memastikan perangkat-perangkat tersebut dapat berfungsi dengan baik bersama. Protokol ini juga harus mampu mengelola berbagai jenis data dan menjamin komunikasi yang efisien serta dapat diandalkan untuk mengoptimalkan penerapan sistem rumah cerdas. Dalam konteks ini, protokol komunikasi seperti Constrained Application Protocol (CoAP) dan Message Queuing Telemetry Transport (MQTT) memainkan peran penting dalam mendukung komunikasi antar perangkat dalam sistem rumah cerdas, sehingga pertimbangan diperlukan guna mengatasi tantangan-tantangan dalam implementasi sistemnya.

2.1.3 Protokol CoAP

CoAP (Constrained Application Protocol) adalah protokol transfer web khusus yang dirancang untuk digunakan dengan perangkat dan jaringan yang terbatas dalam konteks Internet of Things (Almheiri & Maamar, 2021). CoAP juga merupakan protokol ringan yang

dibuat untuk aplikasi M2M dan didirikan oleh IETF (Internet Engineering Task Force) CoRE Working Group seperti yang ditunjukkan pada gambar 2.1. Tujuan desain utama CoAP yaitu untuk mengurangi overhead pesan dan membatasi fragmentasi. Selain itu, keunggulan utama yang dimiliki CoAP terletak pada efisiensi penggunaan daya yang tinggi, mengurangi penundaan transmisi data dan meminimalkan konsumsi daya perangkat (Seoane et al., 2021).



Gambar 2.1 Model komunikasi menggunakan CoAP

Sumber: Prayogo et al (2019)

Protokol CoAP merupakan protokol aplikasi *request-response* yang berdasarkan pada pertukaran pesan asinkron dengan keandalan opsional, karena beroperasi di atas *User Datagram Protocol* (UDP). Jaringan CoAP sendiri terdiri dari dua jenis node. Yang pertama adalah *server* CoAP, yang biasanya merupakan perangkat terbatas seperti sensor atau aktuator, yang dapat diakses atau dikendalikan melalui REST API; yang kedua adalah *client* CoAP, yang merupakan perangkat yang ingin mendapatkan informasi atau meminta tindakan tertentu dari server. CoAP juga memungkinkan komunikasi antara client HTTP dan server CoAP menggunakan protokol CoAP (Safitri & Priambodo, 2023). Seperti permintaan HTTP, CoAP memiliki metode seperti GET, POST, PUT, dan DELETE yang diidentifikasi dalam header pesan. Metode ini semua ditunjukkan dalam pesan header dan digunakan melalui komunikasi CoAP (Almheiri & Maamar, 2021).

CoAP dapat diandalkan dalam mendukung empat jenis pesan yang berbeda, yaitu *Confirmable* (CON), *Non-Confirmable* (NON), *Acknowledgment* (ACK), dan *Reset* (RST). CON ini pesan yang memerlukan keandalan dengan arti klien mengirim pesan, dan server akan merespons dengan ACK jika ID pesannya sama. Namun, jika server tidak dapat memproses pesan CON, server akan merespons dengan RST. Kemudian untuk pesan yang tidak memerlukan pengiriman yang dapat diandalkan, klien mengirimkan permintaan NON ke server

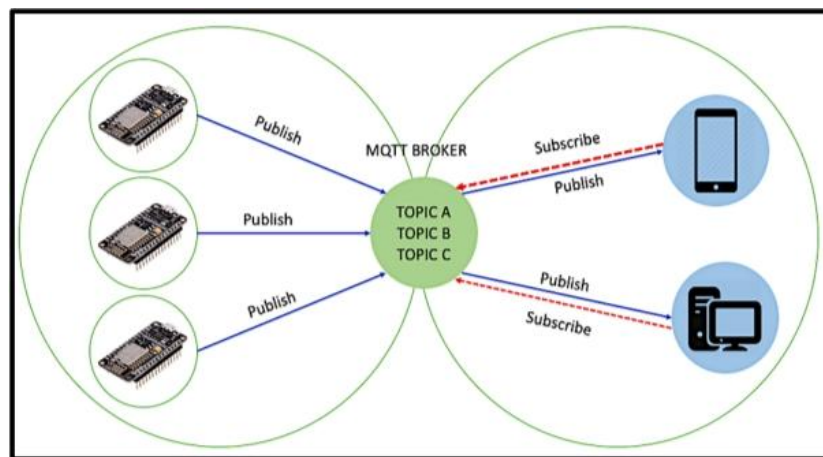
dan tidak mengirim pengakuan, namun menyimpan ID pesan untuk deteksi duplikat (Seoane, Rubio, Almenares, & Campo, 2021). Untuk menjamin keamanan komunikasi, CoAP menggunakan Datagram Transport Layer Security (DTLS) dengan memiliki empat mode keamanan yang berbeda yaitu diantaranya, NoSec, Pre-Shared Key (PSK), Raw PublicKey (RPK), dan Certificate. DTLS adalah pengganti TLS yang bekerja melalui UDP dan memberikan perlindungan integritas, otentikasi, dan enkripsi data (Fahd A & Ebtesam J, 2020).

2.1.4 Protokol MQTT

MQTT (Message Queuing Telemetry Transport) adalah protokol komunikasi yang dirancang untuk komunikasi M2M dan IoT (Internet of Things). MQTT diperkenalkan oleh Andy Stanford-Clark dari IBM dan Arlen Nipper dari Arcom Control Systems Ltd pada tahun 1999 (Kusuma, Munady, & Irawan, 2019). MQTT memiliki standarisasi yang didefinisikan oleh OASIS (Organization for the Advancement of Structured Information Standards) sebagai sebuah protokol yang sederhana, terbuka, ringan, dan efisien dalam penggunaan bandwidth, sehingga cocok untuk digunakan dalam lingkungan yang memiliki keterbatasan sumber daya (Esposito, Belli, Palma, & Pierleoni, 2023).

MQTT menggunakan arsitektur publish/subscribe, di mana data yang dipertukarkan diklasifikasikan berdasarkan topik yang diorganisir secara hierarkis. Setiap pesan yang dipublikasikan oleh klien diasosiasikan dengan topik tertentu, dan pesan tersebut diterima oleh klien lain yang berlangganan pada topik tersebut. Secara sederhana, protokol ini biasanya terdiri dari tiga jenis node: publisher (pengirim pesan), subscriber (penerima pesan), dan broker

(pengelola pesan) seperti yang divisualkan pada gambar 2.2 (Seoane, Rubio, Almenares, & Campo, 2021).



Gambar 2.2 Model komunikasi menggunakan MQTT

Sumber: Prayogo et al (2019)

Publisher adalah klien yang mengirimkan pesan yang terkait dengan topik tertentu, sering kali berupa sensor yang memproduksi data. Subscriber adalah klien yang meminta informasi dengan berlangganan pada topik tertentu, menjadi konsumen data tersebut. Broker berfungsi sebagai server pusat yang menerima dan mengelola permintaan topik dari subscriber serta menerima dan mengirimkan pesan yang dipublikasikan ke klien yang berlangganan topik tersebut (Seoane, Rubio, Almenares, & Campo, 2021).

Pesan MQTT terdiri dari beberapa jenis paket kontrol, yang meliputi header tetap, header variabel, dan payload. Header tetap hadir di semua paket kontrol MQTT, sedangkan header variabel dan payload hanya ada pada beberapa jenis paket kontrol tertentu. Header variabel dapat berisi informasi tambahan tergantung pada jenis pesan, sementara payload berisi informasi aplikasi, yaitu data yang akan dipertukarkan (Seoane, Rubio, Almenares, & Campo, 2021).

Tidak hanya itu saja, protokol MQTT mendukung tiga tingkat Quality of Service (QoS) untuk pengiriman pesan yang dapat diandalkan:

1. QoS 0 (At most once delivery): Pesan ditransmisikan tanpa *acknowledgment* dan dikirimkan sesuai dengan kemampuan jaringan. Pesan PUBLISH dikirimkan paling banyak sekali (Patti, Leonardi, Testa, & Bello, 2024).
2. QoS 1 (At least once delivery): Pesan PUBLISH harus diakui dengan pesan PUBACK, jika tidak, pesan tersebut akan dikirim ulang. Tingkat QoS ini menjamin

bahwa pesan mencapai tujuan, meskipun bisa terjadi lebih dari sekali karena retransmisi (Seoane, Rubio, Almenares, & Campo, 2021).

3. QoS 2 (Exactly once delivery): Pesan PUBLISH adalah pesan pertama dari empat kali jabat tangan yang memastikan pesan tiba tepat sekali. Pesan yang membentuk jabat tangan adalah PUBLISH, PUBREC, PUBREL, dan PUBCOMP (Seoane, Rubio, Almenares, & Campo, 2021).

Berikut adalah deskripsi singkat dari jenis-jenis pesan MQTT yang dirangkum dalam Tabel 2.1.

Tabel 2.1 Jenis-jenis pesan MQTT

MQTT Message	Penjelasan
CONNECT	Permintaan klien untuk terhubung ke server
CONNACK	Pengakuan koneksi
PUBLISH	Menerbitkan pesan
PUBACK	Pengakuan penerbitan
PUBREC	Penerbitan diterima
PUBREL	Pelepasan penerbitan
PUBCOMP	Penerbitan selesai
SUBSCRIBE	Permintaan berlangganan klien
SUBACK	Pengakuan berlangganan
UNSUBSCRIBE	Permintaan berhenti berlangganan
UNSUBACK	Pengakuan berhenti berlangganan
PINGREQ	Permintaan ping
PINGRESP	Respons ping
DISCONNECT	Klien sedang memutuskan koneksi

Keuntungan utama dari penggunaan MQTT dalam protokol IoT adalah overhead yang rendah, keandalan yang dijamin oleh TCP (Transmission Control Protocol), fleksibilitas QoS, kontrol aliran dan kemacetan yang disediakan oleh TCP, pesan selektif melalui arsitektur berbasis topik, serta kesederhanaan klien yang mengurangi konsumsi energi (Naik & Bapat, 2020). Namun, MQTT juga memiliki beberapa keterbatasan seperti kebutuhan akan broker, orientasi koneksi yang mempengaruhi efisiensi energi, isu keamanan yang memerlukan lapisan tambahan seperti TLS (Transport Layer Security) untuk mengamankan komunikasi, dan titik kegagalan tunggal pada broker (Naik & Bapat, 2020). Secara keseluruhan, penerapan MQTT dalam sistem rumah pintar dapat mengoptimalkan pengalaman pengguna dalam pengendalian rumah pintar, mempercepat respons sistem, dan meminimalkan latensi,

membuatnya menjadi pilihan yang sangat baik untuk komunikasi yang efisien dan andal antara perangkat kecil dan seluler (TechTarget, 2022).

Tabel 2.2 Ringkasan dari protokol-protokol Iot, CoAP dan MQTT

	CoAP	MQTT
Badan standarisasi	IETF	OASIS
Arsitektur	Request-Response, Resource “observer“	Publish-Subscribe
Transportasi	UDP	TCP
Keamanan	DTLS (Transportasi)	TLS (Transportasi), <i>client authentication</i>
Metode untuk RESTful	GET, POST, PUT, DELETE	Tidak mendukung
Tipe pesan	CON, NON, RST, ACK	CONNECT, CONNACK, PUBLISH, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK, PINGREQ, PINGRESP, DISCONNECT, AUTH
Sumber id	URI	Topics
Kekuatan	Ringan, prinsip <i>end-to-end</i> , asinkron dan sinkron, fleksibilitas dan keandalan (melalui pesan CON), interoperabilitas	Asinkron, keandalan, kliennya sederhana, pesan terselektif.

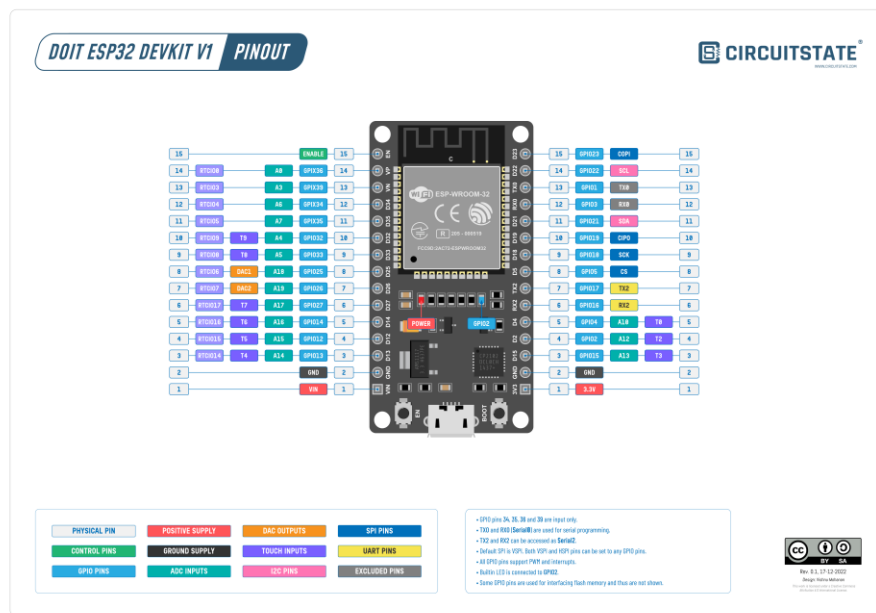
2.1.5 ESP32/ESP8266

ESP32 adalah salah satu komponen utama dalam pengembangan aplikasi Internet of Things (IoT) yang menawarkan solusi lebih canggih dibandingkan pendahulunya, ESP8266. Modul ini dilengkapi dengan dual-core mikroprosesor 32-bit, Wi-Fi (802.11 b/g/n), dan Bluetooth 4.2/LE sehingga menjadikannya platform yang sangat serbaguna untuk berbagai

aplikasi IoT seperti, sistem rumah pintar (Babiuch et al., 2019; Macheso & Thotho, 2022). Konektivitas ganda (Wi-Fi dan Bluetooth) pada ESP32 memungkinkan komunikasi yang lebih fleksibel dan mendukung lebih banyak perangkat dalam ekosistem IoT. Berbeda dengan ESP32, ESP8266 tidak dilengkapi dengan Bluetooth dan total GPIO masih 17, namun memiliki harga yang lebih terjangkau sehingga cocok untuk sistem yang sederhana (Maier et al., 2017).

Salah satu keunggulan utama ESP32 adalah kinerjanya yang lebih cepat dan kemampuannya yang lebih luas, termasuk konsumsi daya yang lebih rendah dalam mode deep sleep, membuatnya sangat ideal untuk perangkat IoT yang membutuhkan daya rendah dalam operasi jangka panjang (Maier et al., 2017). Modul ini juga menyediakan memori flash dan RAM yang lebih besar dibandingkan ESP8266, sehingga mendukung program yang lebih kompleks dan data sensor yang lebih banyak untuk pengoperasian rumah pintar (Babiuch et al., 2019; Ramli et al., 2018). Fleksibilitas ESP32 dalam mendukung protokol IoT seperti HTTP, MQTT, dan CoAP memungkinkan pengembang untuk memilih protokol yang paling sesuai dengan kebutuhan aplikasi mereka, memberikan solusi yang efisien dan mudah diimplementasikan.

Secara penggunaan ESP32/ESP8266 sama-sama memiliki peran yang sangat penting dalam berbagai aspek sistem rumah pintar. Dalam otomasi rumah, modul ini dapat digunakan untuk mengontrol perangkat rumah tangga seperti keamanan rumah, pencahayaan pintar, dan peralatan lainnya dengan memanfaatkan protokol seperti MQTT atau CoAP untuk komunikasi jarak jauh. Dalam pengelolaan lingkungan, mikrokontroler ini dapat terhubung dengan berbagai sensor, seperti sensor suhu, kelembapan, dan kualitas udara, untuk memantau kondisi lingkungan secara real-time dan mengirimkan data ke cloud atau aplikasi ponsel pintar. Begitupun di bidang keamanan, mikrokontroler ini juga dapat digunakan untuk mengelola sistem keamanan rumah yang cerdas, termasuk integrasi dengan sensor gerak, kamera, dan alarm, dengan dukungan komunikasi real-time yang andal melalui Wi-Fi dan Bluetooth.



Gambar 2.3 ESP32

Sumber: <https://www.circuitstate.com/tutorials/getting-started-with-esp32-wifi-bluetooth-soc-using-doit-esp32-devkit-v1-development-board/>

2.1.6 Raspberry PI

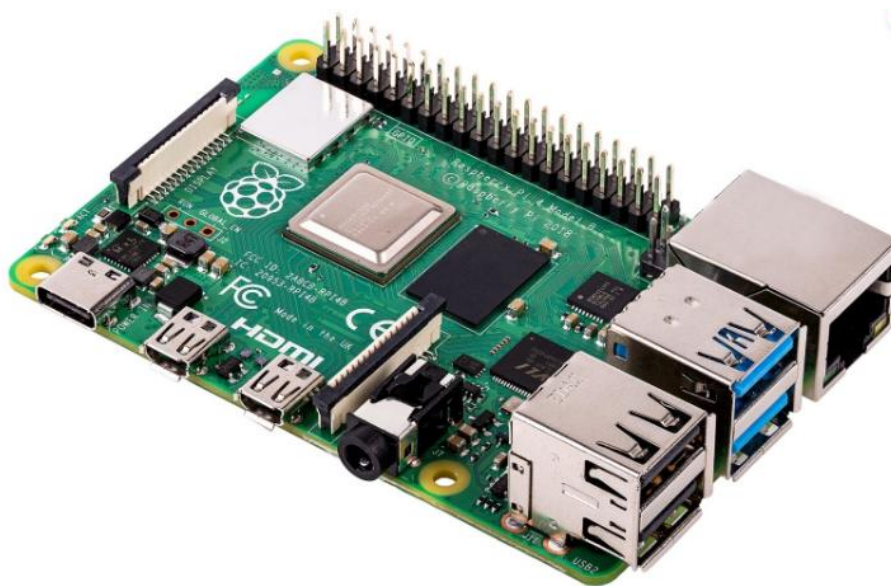
Raspberry Pi merupakan komputer mini yang memiliki peran signifikan dalam pengembangan sistem rumah cerdas dan Internet of Things (IoT). Perangkat ini menawarkan fleksibilitas dan kemampuan komputasi yang memadai untuk mengelola berbagai sensor dan aktuator dalam ekosistem rumah pintar. Raspberry Pi dapat berfungsi sebagai gateway atau pusat kontrol yang mengintegrasikan berbagai perangkat IoT, memproses data sensor, dan mengirimkan perintah ke aktuator (Sathya et al., 2022). Keunggulan Raspberry Pi terletak pada kemampuannya untuk menjalankan sistem operasi Linux yang open source, memberikan fleksibilitas dalam pengembangan aplikasi dan integrasi dengan berbagai protokol komunikasi IoT seperti CoAP dan MQTT (Rahman et al., 2017).

Dalam konteks sistem rumah cerdas, Raspberry Pi dapat digunakan untuk mengimplementasikan berbagai fungsi otomasi, termasuk kontrol pencahayaan, pemantauan suhu dan kelembapan, serta manajemen keamanan (Robbani et al., 2024). Perangkat ini juga mendukung integrasi dengan asisten suara seperti Alexa, meningkatkan interaktivitas dan kemudahan penggunaan sistem rumah cerdas (Calvopinna et al., 2019). Kemampuan Raspberry Pi untuk mengumpulkan, memproses, dan mengirimkan data sensor ke layanan

cloud memungkinkan pemantauan real-time dan analisis lanjutan untuk optimalisasi kinerja sistem rumah cerdas (Sunehra, 2019).

Dalam penelitian terbaru, Raspberry Pi telah digunakan sebagai platform untuk mengembangkan solusi rumah cerdas yang inovatif dan hemat biaya. Sebuah studi pada tahun 2024 mendemonstrasikan penggunaan Raspberry Pi 4B sebagai otak dari sistem rumah cerdas yang terintegrasi dengan platform Apple HomeKit dan Google Home, menunjukkan kemampuannya dalam mengelola sensor suhu, cahaya, dan relay untuk kontrol perangkat (Robbani et al., 2024). Penelitian lain menunjukkan efektivitas Raspberry Pi dalam implementasi sistem monitoring polusi dan iklim berbasis IoT, menekankan fleksibilitasnya dalam mengintegrasikan berbagai jenis sensor dan mengunggah data ke web (Sunehra, 2019).

Dengan kemampuannya yang luas dan dukungan komunitas yang kuat, Raspberry Pi menjadi pilihan yang ideal untuk pengembangan dan penelitian sistem rumah cerdas, termasuk dalam analisis kinerja protokol komunikasi seperti CoAP dan MQTT. Perangkat ini menyediakan platform yang stabil dan dapat diandalkan untuk menguji dan membandingkan efisiensi protokol-protokol tersebut dalam konteks aplikasi rumah cerdas yang nyata (Yaldaie et al., 2024).



Gambar 2.4 Raspberry Pi

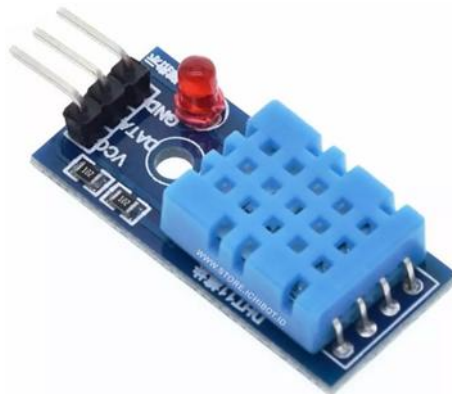
Sumber: [Getting Started with Raspberry Pi | Random Nerd Tutorials](#)

2.1.7 Sensor DHT11

DHT11 merupakan sensor yang umum digunakan dalam sistem rumah cerdas untuk memantau suhu dan kelembaban. Sensor ini memiliki keunggulan dalam hal kemudahan

integrasi dengan mikrokontroler serta konsumsi daya yang rendah, menjadikannya pilihan ideal untuk sistem yang membutuhkan efisiensi energi. DHT11 mampu mengukur suhu dalam rentang 0°C hingga 50°C dengan akurasi $\pm 2^\circ\text{C}$ dan kelembaban relatif dari 20% hingga 90% dengan akurasi $\pm 5\%$ (Wicaksono & Hidayat, 2017). Selain itu, maksimal dalam penggunaan arus selama konversi sebesar 2.5mA serta daya yang dimiliki mencapai sekitar 3-5V dan I/O (Wicaksono, 2018).

Keandalan dan stabilitas sensor ini dalam lingkungan rumah tangga membuatnya sering digunakan dalam proyek-proyek berbasis IoT (Internet of Things), termasuk yang melibatkan protokol komunikasi seperti CoAP dan MQTT untuk mengirimkan data pengukuran ke server atau aplikasi kontrol. Penggunaan sensor DHT11 dalam sistem rumah cerdas memungkinkan pemantauan kondisi lingkungan secara real-time dan otomatisasi yang lebih responsif, sehingga meningkatkan efisiensi pengelolaan energi dan kenyamanan penghuni rumah (Al-Kuwari et al., 2020; Sharma & Jindal, 2020). Selain itu, karena sensor ini mendukung komunikasi satu arah, penggunaannya dengan protokol seperti MQTT, yang mendukung pengiriman data berbasis publish-subscribe, menjadi lebih relevan dalam menciptakan sistem pemantauan rumah yang efisien dan terukur (Kumar et al., 2019; Gupta & Rani, 2021). Berikut bentuk fisik sensor DHT11 terdapat pada gambar 2.5.



Gambar 2.5 Sensor DHT11

Sumber: <https://store.ichibot.id/product/dht11-sensor-suhu-dan-kelembapan-temperature/>

2.1.8 Arduino IDE

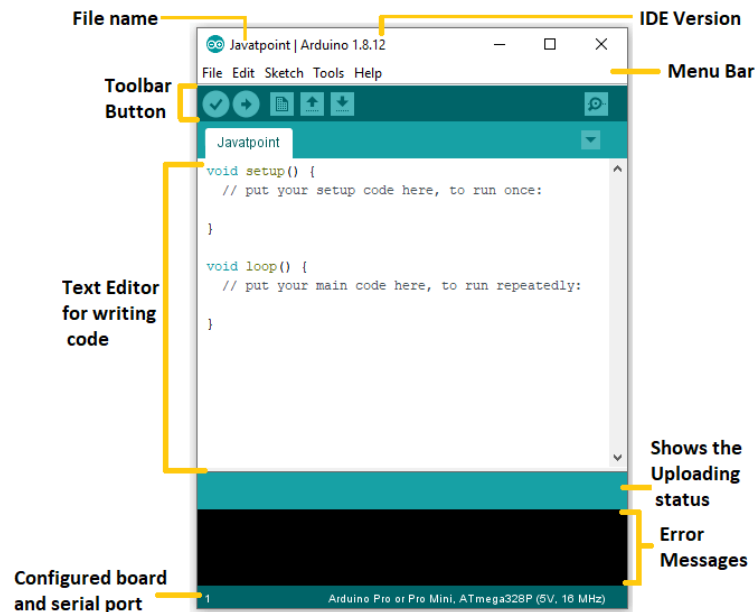
Arduino IDE (Integrated Development Environment) adalah perangkat lunak open-source yang memainkan peran penting dan digunakan untuk pengembangan aplikasi Internet of Things (IoT), termasuk sistem rumah cerdas. Lingkungan pengembangan ini menyediakan platform yang user-friendly untuk menulis, mengedit, dan mengunggah kode ke berbagai jenis

board Arduino dan mikrokontroler kompatibel lainnya, seperti ESP32 dan ESP8266. Arduino IDE mendukung bahasa pemrograman C++ yang telah disederhanakan, memungkinkan pengembang untuk dengan mudah mengimplementasikan protokol komunikasi IoT seperti CoAP (Constrained Application Protocol) dan MQTT (Message Queuing Telemetry Transport) dalam proyek-proyek mereka (Adi et al., 2023; Mr. Saurabh R. Chaudhari & Prof. Gajanan D. Nagoshe, 2024).

Dalam konteks sistem rumah cerdas, Arduino IDE memfasilitasi integrasi sensor, aktuator, dan modul komunikasi dengan menyediakan berbagai library yang dapat diunduh dan digunakan secara langsung. Hal ini sangat membantu dalam pengembangan prototipe cepat dan implementasi berbagai fitur rumah cerdas, seperti monitoring lingkungan, kontrol perangkat, dan manajemen energi. Kemampuan Arduino IDE untuk mendukung berbagai board dan shield memungkinkan fleksibilitas dalam pemilihan hardware yang sesuai dengan kebutuhan spesifik proyek rumah cerdas (Pasha Mohd Daud et al., 2020).

Salah satu keunggulan Arduino IDE adalah kemampuannya untuk memfasilitasi pengembangan aplikasi IoT yang hemat energi. Ini sangat relevan dalam implementasi protokol CoAP dan MQTT, yang dirancang untuk efisiensi dalam lingkungan dengan sumber daya terbatas. Arduino IDE memungkinkan pengembang untuk mengoptimalkan kode mereka untuk konsumsi daya yang lebih rendah, sebuah aspek kritis dalam perangkat IoT yang sering bergantung pada baterai atau sumber daya terbatas (Senbagavalli & Singh, 2022).

Dalam analisis kinerja protokol CoAP dan MQTT pada sistem rumah cerdas, Arduino IDE menawarkan tools debugging dan serial monitor yang membantu dalam pengujian dan optimalisasi implementasi protokol. Kemampuan ini memungkinkan pengembang untuk memantau lalu lintas data, mengukur latensi, dan menganalisis efisiensi protokol dalam kondisi nyata. Dengan dukungan komunitas yang luas dan update berkala, Arduino IDE terus berkembang untuk memenuhi kebutuhan yang semakin kompleks dalam pengembangan IoT, menjadikannya pilihan yang ideal untuk penelitian dan implementasi sistem rumah cerdas modern.



Gambar 2.6 Arduino IDE

Sumber: <https://www.javatpoint.com/arduino-ide>

2.2 Penelitian Terdahulu

Terdapat beberapa penelitian sebelumnya mengenai analisis penggunaan protokol CoAP dan MQTT yang digunakan sebagai salah satu acuan dalam melakukan penelitian. Berikut adalah hasil rangkuman penulis berdasarkan penelitian sebelumnya:

Tabel 2.3 Contoh tabel yang dibuat menggunakan MS Word

JUDUL	PENULIS	PENELITIAN	HASIL	
			MQTT	COAP
MQTT and CoAP Communication Protocol Analysis in Internet of Things System for Strawberry	(Safitri & Seto Priambodo, 2023)	Metodologi penelitian ini mengharuskan semua perangkat terhubung ke jaringan WiFi yang sama selama 3 jam, dengan data yang dikumpulkan dalam interval 5 menit. Setelah	Protokol MQTT terbukti lebih cocok karena konfigurasi pesannya yang efisien dan dapat diandalkan. Tetapi dalam kinerjanya, terdapat	Sebaliknya protokol CoAP, menunjukkan overhead dan ukuran paket yang lebih kecil, UDP sebagai lapisan transport, dan kinerja yang unggul dalam hal

Hydroponic Plants		diaktifkan, perangkat mengirimkan pembacaan sensor yang berkaitan dengan nutrisi tanaman dan kondisi stroberi hidroponik ke server pusat. Server ini memproses data, mengubahnya menjadi format JSON sebelum menyebarkannya melalui protokol MQTT, CoAP, dan Cloud (Firebase). Pendekatan ini memfasilitasi analisis dan tampilan data di dasbor untuk evaluasi komparatif kinerja protokol.	kekurangan yakni kehilangan paket dan penggunaan baterai yang tinggi karena overhead yang lebih besar, ukuran paket, dan ketergantungan pada TCP.	kehilangan paket, menunjukkan penggunaan baterai rata-rata yang lebih rendah yaitu 3% dan penundaan rata-rata yang jauh lebih rendah yaitu 1,80 detik.
The Use and Performance of MQTT and CoAP as Internet of Things Application Protocol using	(Prayogo et al., 2019)	Penelitian ini membandingkan protokol MQTT dan CoAP menggunakan NodeMCU ESP8266 untuk transmisi data dalam aplikasi IoT. Rancangan eksperimental meliputi satu	MQTT, metode transmisi data IoT berbasis TCP, yang menawarkan akurasi dan keandalan yang lebih tinggi tetapi membutuhkan konektivitas internet untuk	CoAP merupakan protokol IoT berbasis UDP yang menawarkan waktu transmisi lebih cepat, hanya membutuhkan

NodeMCU ESP8266		<i>publisher</i> , satu broker, tiga <i>subscriber</i> , dan penggunaan dua ISP yang berbeda untuk menganalisis secara menyeluruh efisiensi dan waktu transmisi data di seluruh protokol ini.	kinerja yang optimal, sehingga cocok untuk aplikasi dengan keandalan tinggi. Dalam percobaan tersebut, MQTT membutuhkan waktu 25 detik untuk mengirimkan 100 titik data melalui penyedia internet, dari data itu mencerminkan kinerjanya yang kuat namun lebih lambat dibandingkan dengan CoAP.	waktu 20 detik untuk mengirim 100 titik data dalam jaringan lokal. Tetapi tidak menjamin keandalan atau keakuratan data. Oleh karena itu, CoAP lebih cocok untuk lingkungan di mana transmisi data yang cepat lebih penting daripada keandalan dan keakuratan absolut dari data yang ditransmisikan.
A Brief Comparative Analysis on Application Layer Protocols of Internet of Things: MQTT, CoAP, AMQP and HTTP	(Naik & Bapat , 2020)	Penelitian ini menganalisis protokol pesan MQTT, CoAP, AMQP, dan HTTP untuk sistem IoT, membandingkan fitur dan fungsinya, untuk memberikan wawasan tentang skenario ideal dan penggunaannya	MQTT menawarkan pengiriman pesan yang dapat diandalkan dengan tingkat QoS dan komunikasi asinkron untuk aplikasi real-time. MQTT beroperasi melalui TCP/IP,	CoAP merupakan sebuah metode komunikasi yang cepat serta menggunakan paket UDP untuk keamanan dan integritas, tetapi pengirimannya yang tidak dapat

		dalam berbagai sistem.	tetapi memiliki ketergantungan pada broker pusatnya yang dapat menyebabkan kegagalan komunikasi jika gagal.	diandalkan dan membutuhkan paket ACK untuk konfirmasi sehingga menyebabkan ketidakpastian.
Analisa dan Implementasi Smart Home Menggunakan Protokol MQTT dan XMPP Berbasis Internet of Thing	(Kusuma et al., 2019)	Penelitian ini menggunakan dua protokol yang berbeda (MQTT dan XMPP) untuk menghubungkan kamera Raspberry Pi ke smartphone melalui internet. Koneksi dilakukan selama 60 menit menggunakan Wireshark, meningkatkan jumlah sensor nirkabel dari satu menjadi tiga. Ini juga termasuk transmisi data ke server protokol.	Studi ini mengevaluasi efektivitas protokol MQTT dalam alokasi paket, throughput, dan kebutuhan paket. MQTT lebih efisien dalam jaringan dengan bandwidth terbatas dibandingkan dengan XMPP, menawarkan bandwidth tinggi dan kinerja tinggi untuk komunikasi M2M dan platform cloud.	—
Performance evaluation of CoAP and	(Seoane et al., 2021)	Studi ini mengevaluasi kinerja protokol CoAP dan	MQTT membutuhkan lebih banyak	CoAP lebih efisien dalam penggunaan

MQTT with security support for IoT environments		MQTT dalam hal efisiensi bandwidth, penggunaan CPU, dan latensi komunikasi. Menganalisis berdasarkan 90 eksperimen serta menyesuaikan kinerja protokol untuk manajemen data IoT.	bandwidth karena overhead TCP, hingga 1.560 byte dalam skenario layanan berkualitas tinggi, dan meningkatkan konsumsi CPU. Hal ini dapat menyebabkan latensi yang lebih tinggi, yang berpotensi memengaruhi komunikasi real-timenya.	bandwidth dan penggunaan CPU, rata-rata hanya 226 byte dalam jaringan lossless yang tidak aman, menawarkan latensi yang lebih rendah dibandingkan dengan MQTT, sehingga cocok untuk komunikasi yang cepat dan real-time.
A systematic literature review: Messaging protocols and electronic platforms used in the internet of things for the purpose of building smart homes	(Yudidharma et al., 2023)	Penelitian ini menggunakan <i>Systematic Literature Review</i> (SLR), 32 makalah dari IEEE Xplore dianalisis untuk menjawab pertanyaan penelitian tentang Internet of Things (IoT) untuk rumah pintar, dengan fokus pada protokol pengiriman, kinerja, efektivitas, dan platform elektronik yang digunakan.	Mayoritas peneliti menggunakan MQTT sebagai protokol pengiriman pesan utama karena kinerjanya yang lebih baik dibandingkan protokol lain. MQTT memiliki bandwidth jaringan yang luas dan ukuran paket yang ditransfer lebih rendah.	CoAP sangat mirip dengan MQTT, tetapi kurang terstandarisasi. CoAP bagus untuk mencegah kegagalan jaringan, dan membutuhkan sumber daya jaringan lebih sedikit.

Berikut penjelasan lebih lengkapnya dari masing-masing poin terkait rangkuman jurnal-jurnal yang ada di tabel 2.3:

- a. Meninjau berdasarkan penelitian yang telah dilakukan oleh Nilam Andri Safitri dengan Andry Seto Priambodo mengenai “MQTT and CoAP Communication Protocol Analysis in Internet of Things System for Strawberry Hydroponic Plants”. Dalam penelitian tersebut penulis berfokus pada *Back End* untuk menganalisis kinerja kedua protokol MQTT dan CoAP. Dalam prosesnya, Karena protokol MQTT dapat mengkonfigurasi pesan atau paket yang terkirim, MQTT cocok untuk sistem otomasi tanaman hidroponik berbasis IoT. MQTT juga dapat menyediakan komunikasi yang handal dan efisien dalam lingkungan IoT. Tujuan lain dari pembuatan prototype budidaya stroberi dalam wadah ini adalah agar petani stroberi dapat mengawasi tanaman mereka tanpa harus dekat dengannya. Hasil pengujian IoT menggunakan metode MQTT dan CoAP menunjukkan bahwa CoAP lebih cocok untuk sistem *one-to-one*, di mana pesan dikirim antara satu client dan server. Sebaliknya, MQTT lebih cocok untuk sistem *many-to-many*, di mana pesan dikirim melalui broker antara beberapa client.
- b. Berdasarkan penelitian yang telah dilakukan oleh Sandy Suryo Prayogo, Yulisdin Mukhlis, dan Bayu Kumoro Yakti tentang “The Use and Performance of MQTT and CoAP as Internet of Things Application Protocol using NodeMCU ESP8266”, penulis mengusulkan skema untuk mengalirkan banyak data dari *publisher to subscriber* menggunakan perangkat IoT yaitu mikrokontroler NodeMCU 8266 menggunakan dua protokol yang berbeda, ada MQTT dan CoAP. Untuk mengetahui penggunaan dan efektifitas dari dua protokol aplikasi yang berbeda, penulis mencoba mentransfer data dari satu perangkat IoT ke perangkat IoT lainnya. Hasil eksperimen menunjukkan bahwa kinerja protokol yang berbeda tergantung pada kondisi jaringan yang berbeda. Perbedaan dalam penggunaan dan kinerja dapat dimanfaatkan untuk meningkatkan kebutuhan transmisi data dalam IoT. Sebagai contoh, CoAP diimplementasikan untuk banyak node IoT di area kecil dan MQTT untuk node tunggal. Ini juga dapat dikombinasikan seperti banyak node yang terhubung menggunakan CoAP kemudian server mempublikasikannya ke internet menggunakan MQTT.
- c. Pada jurnal penelitian yang berjudul “A Brief Comparative Analysis on Application Layer Protocols of Internet of Things: MQTT, CoAP, AMQP and HTTP” oleh Girija P. Naik dan Prof. A. Umesh Bapat berisi penjelasan mengenai keuntungan dan kerugian

dari penggunaan tiap protokol. Dapat kita lihat terutama perbandingan antara dua protokol, CoAP dan MQTT. Penulis menjelaskan bahwa CoAP mendukung komunikasi *many-to-many*, yang berarti memungkinkan interaksi antara banyak perangkat sekaligus, meningkatkan fleksibilitas dan konektivitas dalam jaringan. Sementara, mekanisme *publish/subscribe* pada MQTT memiliki kemampuan seperti *one-to-one*, *many-to-many*, atau *one-to-none*, serta menyediakan komunikasi dua arah. Kesimpulan yang didapatkan dari hasil analisis tersebut adalah bahwa perlu mempertimbangkan kelebihan dan kekurangan masing-masing protokol saat memilih protokol Internet of Things (IoT). Oleh karena itu, pengguna harus tahu apa yang dibutuhkan sistem IoT mereka dan melihat fitur protokol untuk memilih yang paling sesuai dengan kebutuhan sistem mereka.

- d. Berdasarkan penelitian yang telah dilakukan oleh Feri Teja Kusuma, Rendy Munady, dan Arif Indra Irawan tentang “Analisa dan Implementasi Smart Home Menggunakan Protokol MQTT dan XMPP Berbasis Internet of Things”, penulis membandingkan dua protokol TCP/IP di mana MQTT berada di bagian application layer karena memiliki kelebihan untuk mengirimkan pesan dengan ukuran yang kecil. Sedangkan XMPP terdapat kelebihan fitur yang tidak dimiliki oleh MQTT seperti identitas, federasi, serta berbagai extension IoT yang dapat digunakan pada rumah pintar. Pada hasil penelitian didapatkan bahwa, protokol MQTT unggul dalam performa pada koneksi sensor hingga end point device, terutama pada packet received, throughput, dan packet loss. Rekomendasi penggunaan MQTT muncul ketika jaringan memiliki bandwidth terbatas untuk menghindari bottleneck, dengan throughput 607 Byte/second, lebih baik daripada XMPP (3477 Byte/second). XMPP, di sisi lain, cocok untuk komunikasi dua arah antara Raspberry Pi dan client, menunjukkan average delay yang memadai, dan memberikan lapisan keamanan saat melewati cloud platform. Sistem yang diuji terbukti memiliki tingkat ketersediaan sebesar 97,647% dan tingkat kehandalan sebesar 99,4%, menandakan keandalan sistem yang digunakan.
- e. Penelitian yang berjudul “Performance evaluation of CoAP and MQTT with security support for IoT environments” membahas tentang protokol MQTT dan CoAP dalam konteks jaringan IoT, dengan fokus pada peran *publisher*, *subscriber*, dan broker. Penelitian yang dilakukan oleh Victor Seoane, Carlos Garcia-Rubio, Florina Almenares, dan Celeste Campo ini berfokus pada evaluasi kinerja protokol CoAP dan MQTT dalam hal efisiensi bandwidth, penggunaan CPU, dan latensi komunikasi.

Sebanyak 90 eksperimen dilakukan untuk menganalisis latensi protokol MQTT dengan berbagai tingkat QoS. Hasil dari penelitian tersebut menunjukkan bahwa CoAP lebih efisien daripada MQTT dalam penggunaan bandwidth dan CPU. CoAP menggunakan lebih sedikit data (rata-rata 226 byte) dibandingkan MQTT yang bisa mencapai 1560 byte. Selain itu, CoAP juga lebih ringan dalam penggunaan CPU. Dengan demikian, CoAP mungkin lebih baik untuk aplikasi IoT yang memerlukan efisiensi bandwidth dan pemrosesan, sementara MQTT mungkin menimbulkan latensi yang lebih tinggi. Akan tetapi, studi ini tidak hanya membandingkan kinerja protokol CoAP dan MQTT saja, tetapi juga dibandingkan dengan MQTT-SN dan QUIC untuk memahami *overhead* dalam pengambilan data dari perangkat IoT dan pendistribusiannya ke perangkat IoT.

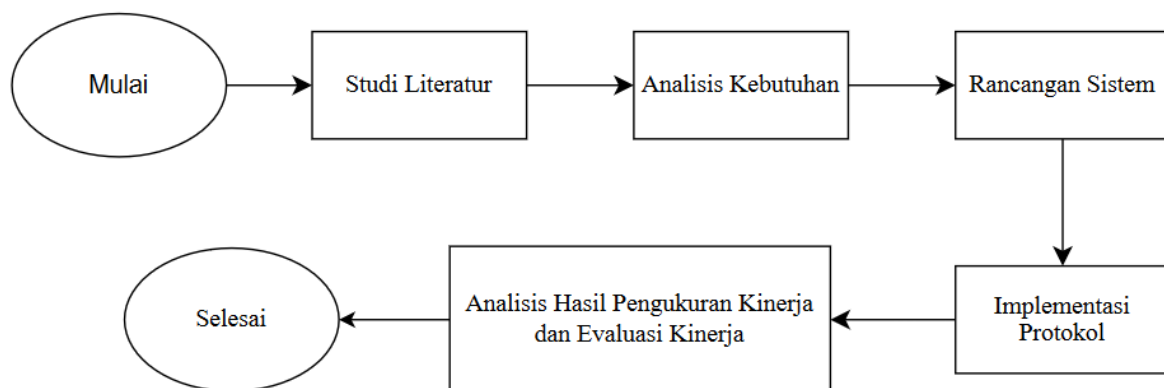
- f. Dalam penelitiannya Arya Yudidharma, Nicholas Nathaniel, Tang Nyquel Gimli, Said Achmad, dan Aditya Kurniawan yang berjudul “A systematic literature review: Messaging protocols and electronic platforms used in the internet of things for the purpose of building smart homes” membahas tentang implementasi IoT dalam pengembangan rumah pintar, dengan fokus pada protokol pesan dan platform elektronik yang digunakan. Penelitian ini dilakukan dengan menggunakan metode *Systematic Literature Review* (SLR) untuk mengeksplorasi teknologi yang digunakan dalam pembangunan smart home. Identifikasi dilakukan dengan memasukkan kata kunci "Internet of Things untuk Rumah Pintar" melalui IEEE Xplore dengan kriteria publikasi dalam enam tahun terakhir. Hasilnya, ditemukan 32 makalah relevan dengan topik penelitian. Penelitian ini menganalisis berbagai protokol messaging seperti MQTT, CoAP, AMQP, dan DDS dalam hal penggunaannya dan kinerjanya pada implementasi smart home. Hasil penelitian menunjukkan bahwa MQTT memiliki kinerja yang lebih baik karena bandwidth jaringan yang luas dan ukuran paket yang kecil, meskipun memiliki keterbatasan dalam hal kontrol aliran dan interoperabilitas. Selain itu, penelitian ini juga menyoroti Raspberry Pi, Arduino, dan DDS sebagai platform elektronik yang umum digunakan dalam implementasi sistem smart home. Khusus untuk perbandingan antara MQTT dan CoAP, penelitian ini menemukan bahwa MQTT adalah protokol messaging yang ringan dengan dukungan untuk beberapa tingkat QoS dan komunikasi dua arah melalui jaringan yang tidak stabil, namun memiliki keterbatasan pada fungsionalitas yang lebih canggih seperti kontrol aliran dan interoperabilitas. Di sisi lain, CoAP mendukung komunikasi one-to-one dan many-to-

many, menawarkan karakteristik keamanan seperti integritas, kerahasiaan, dan autentikasi, serta mengkonsumsi sumber daya jaringan lebih sedikit dibandingkan protokol lainnya.

BAB III

METODOLOGI PENELITIAN

Metodologi penelitian adalah pendekatan sistematis dalam mengumpulkan dan mengevaluasi data selama penelitian untuk memastikan keandalan, validitas, dan reliabilitas metode yang digunakan (Arhami, Husaini, Salahuddin, & Rudi F, 2023). Dalam penelitian analisis kinerja protokol CoAP dan MQTT pada sistem rumah cerdas, peneliti akan melakukan pengujian kinerja kedua protokol tersebut dengan mengimplementasikannya pada perangkat keras ESP8266 dan Raspberry Pi. Metode penelitian yang akan digunakan adalah pendekatan eksperimental, yakni meliputi studi pustaka, pengujian, serta analisis hasil untuk mendapatkan pemahaman yang mendalam tentang kinerja protokol tersebut. Gambaran metodologi ini direpresentasikan dalam diagram yang menunjukkan tahapan pengumpulan data, analisis kebutuhan, rancangan sistem, implementasi protokol, serta analisis hasil pengukuran kinerja dan evaluasi kinerja.



Gambar 3.1 Diagram alir penelitian

3.1 Studi Literatur

Dalam penelitian ini, pengumpulan data dilakukan melalui studi literatur yang menyeluruh melalui berbagai sumber, termasuk jurnal ilmiah dan buku. Studi literatur ini dilakukan untuk mengumpulkan data terkait kinerja sistem rumah cerdas yang menggunakan protokol CoAP dan MQTT. Data yang dikumpulkan merupakan data kuantitatif yang meliputi informasi tentang kinerja sistem rumah cerdas, termasuk metrik kinerja yang diukur dan hasil analisis yang diperoleh. Data ini kemudian digunakan sebagai bahan analisis dalam penelitian ini untuk mengevaluasi kinerja kedua protokol sistem rumah cerdas tersebut sehingga dapat memastikan keakuratan dan keandalannya dalam menyediakan data yang diperlukan.

Pendekatan ini memungkinkan peneliti untuk mengumpulkan informasi yang relevan dan terkini. Dengan begitu, peneliti dapat mendalami kerangka konseptual yang telah ada dan menyesuaikan penelitian mereka agar sejalan dengan temuan-temuan sebelumnya.

3.2 Analisis Kebutuhan

Analisis kebutuhan merupakan salah satu langkah penting yang harus dilakukan agar penelitian dapat berjalan dengan baik serta dapat mengetahui target yang akan dicapai. Analisis kebutuhan untuk penelitian ini melibatkan identifikasi kebutuhan fungsional dan non-fungsional seperti, hardware maupun software yang diperlukan.

3.2.1 Analisis Kebutuhan Fungsional

Mengidentifikasi kebutuhan fungsional menjadi patokan dalam menentukan proses-proses yang diperlukan untuk sistem yang akan dibangun. Dalam hal ini, kebutuhan fungsional sistem mencakup pemetaan detail proses penelitian yang harus dilakukan, termasuk langkah-langkah spesifik dan parameter pengukuran yang akan menjadi penilaian hasil akhir. Berikut adalah kebutuhan fungsional yang harus dianalisis dalam penelitian yang akan dikembangkan:

- a. Sistem dapat mendeteksi suhu kelembaban ruangan
- b. Keberhasilan client (MQTT/CoAP) mengirim pesan
- c. Keberhasilan server menerima pesan yang dikirim oleh client
- d. Keberhasilan membandingkan kinerja kedua protokol berdasarkan parameter yang telah ditentukan, diantaranya:
 1. Packet loss
 2. Delay
 3. Throughput

3.2.2 Analisis Kebutuhan Non-Fungsional

Mengidentifikasi kebutuhan non-fungsional bertujuan untuk menentukan spesifikasi yang diperlukan agar sistem yang akan dibangun dapat beroperasi dengan baik dan memberikan dukungan. Berikut adalah kebutuhan non-fungsional yang harus dipenuhi oleh sistem yang akan dikembangkan:

Tabel 3.1 Kebutuhan Non-Fungsional Hardware

No	Nama <i>Hardware</i>	Jumlah	Fungsi
1.	Sensor DHT 11	1	Membaca suhu dan kelembaban ruangan

2.	NodeMCU ESP 32 Devkit 1	1	Sebagai board mikrokontroler untuk menghubungkan sistem ke jaringan Wi-Fi dan client, mengolah data dari sensor untuk dikirimkan ke server
3.	Kabel data micro USB	1	Menghubungkan NodeMCU ke sumber daya dan perangkat komputer
4.	Raspberry Pi 3	1	Sebagai server untuk mengumpulkan dan menerima pesan dari client
5.	Breadboard	1	Tempat perakitan sementara untuk menghubungkan komponen hardware
6.	Kabel Jumper M to M	2	Penghubung antar pin di breadboard
7.	Kabel Jumper M to F	3	Menghubungkan pin male ke female pada breadboard atau komponen lain
8.	Resistor 10k ohm	1	Sebagai pull-up resistor untuk menjaga stabilitas sinyal data
9.	Router	1	Menghubungkan perangkat ke jaringan internet
10.	Monitor	1	Menampilkan visualisasi server dan hasil pengujian
11.	Mouse	1	Sebagai alat interaksi dengan Raspberry Pi
12.	Keyboard	1	Sebagai alat interaksi dengan Raspberry Pi
13.	Kabel HDMI	1	Menghubungkan Raspberry Pi ke monitor

Tabel 3.2 Kebutuhan Non-Fungsional Software

No	Nama <i>Software</i>	Jumlah	Fungsi
1.	Fritzing	1	Software untuk merancang dan mensimulasikan rangkaian elektronik secara visual
2.	Arduino IDE	1	Platform pengembangan perangkat lunak untuk mengolah program pada mikrokontroler

3.	Mosquitto	1	Protokol komunikasi MQTT untuk mengirim dan menerima pesan antar perangkat IoT
4.	CoAP	1	Protokol komunikasi ringan untuk perangkat terhubung dalam jaringan IoT
5.	Wireshark	1	Software untuk menganalisis lalu lintas jaringan dan memantau paket data secara real time.

Dengan menggunakan alat-alat tersebut, penelitian ini dapat melaksanakan pengujian dan analisis terhadap kinerja sistem rumah cerdas dalam mengumpulkan dan mengirimkan data, serta dalam penggunaan sumber daya seperti bandwidth dan energi.

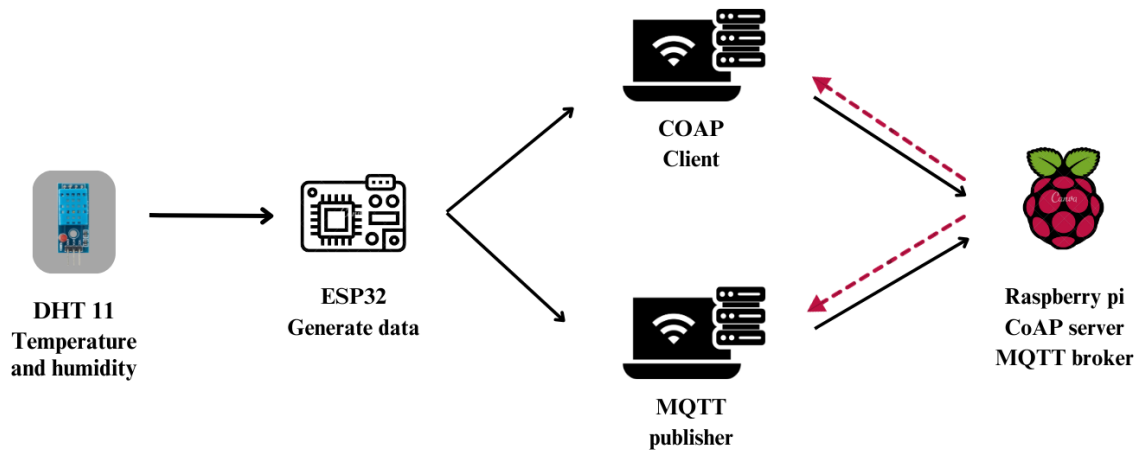
3.3 Rancangan Sistem

Perancangan sistem memberikan gambaran proyek yang akan diimplementasikan ke dalam proyek penelitian akhir. Rancangan sistem ini dibagi menjadi tiga sub bagian diantaranya, alur kerja sistem, desain alat, dan perakitan. Setiap tahapan perancangan bertujuan untuk memastikan bahwa setiap perangkat dalam sistem dapat berfungsi dengan baik, sehingga data dari sensor dapat dikumpulkan dengan akurat melalui mikrokontroler. Pada tahap perakitan, komponen perangkat keras diintegrasikan dengan perangkat lunak untuk membangun sistem secara fisik. Selain itu, rancangan ini juga memperhatikan kebutuhan komunikasi antarperangkat, di mana protokol CoAP dan MQTT digunakan untuk mendukung pengiriman data secara efisien. Dengan mengikuti rancangan ini, sistem yang dirancang diharapkan mampu bekerja secara optimal dalam berbagai kondisi pengujian.

3.3.1 Alur Kerja Sistem

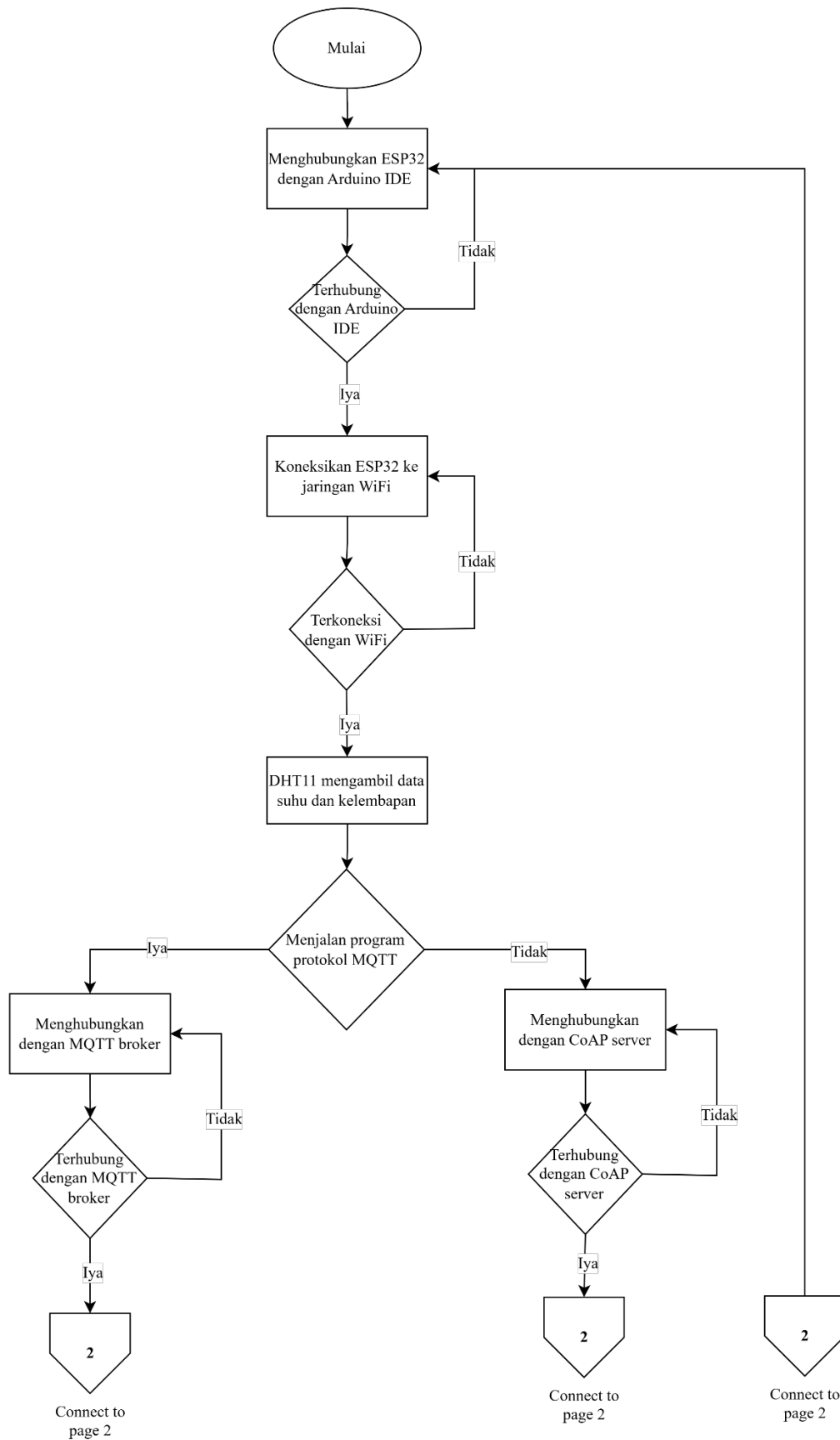
Berikut gambaran besar alur kerja sistem rumah cerdas yang akan dianalisis berdasarkan kinerja protokol CoAP dan MQTT dirancang dengan menggunakan arsitektur client-server. Raspberry Pi berperan sebagai pusat komunikasi data dalam sistem rumah cerdas dengan fungsi sebagai CoAP server dan MQTT broker, sedangkan ESP32 berfungsi sebagai client. Saat protokol MQTT yang dijalankan, Raspberry Pi digunakan untuk mengelola distribusi pesan antara publisher (ESP32) dan subscriber. Sedangkan saat protokol CoAP yang dijalankan, Raspberry Pi digunakan untuk memproses permintaan (request) dari client (ESP32) dan memberikan respons (response). ESP32 digunakan untuk Mengambil data dari DHT11 dan

mengirimkannya ke Raspberry Pi menggunakan protokol MQTT (sebagai publisher) atau CoAP (sebagai client). Seperti yang sudah divisualkan pada gambar 3.2 tersebut.

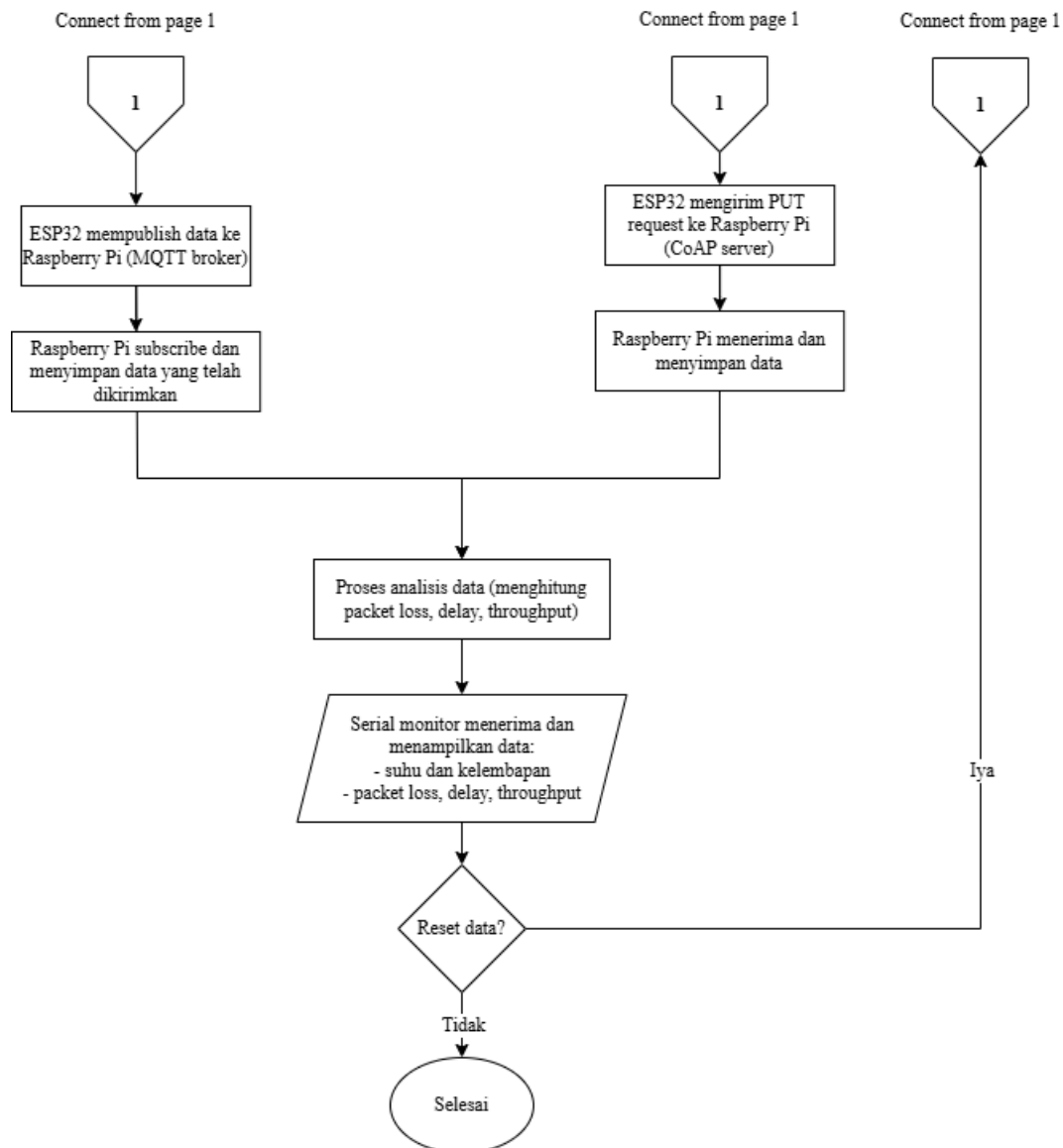


Gambar 3.2 Skema alur kerja sistem

Dilihat dari skema alur kerja sistem tersebut dimana sensor DHT11 bertindak sebagai pendeteksi suhu dan kelembaban ruangan. Data suhu dan kelembaban tersebut kemudian ditransfer ke ESP32 untuk dibaca dan ditampilkan melalui Arduino IDE. Kemudian dari client akan mengirim ke raspberry pi sebagai server dan disubscribe kembali untuk ditampilkan hasilnya melalui serial monitor pada Arduino IDE. Sebagai tambahan, agar memperjelas alur kerja sistem tersebut, berikut diagram alir sistem atau *flowchart* menggunakan CoAP dan MQTT yang akan ditunjukkan pada gambar 3.3 dan gambar 3.4 di bawah ini.



Gambar 3.3 Diagram alir alur kerja sistem



Gambar 3.4 Diagram alir alur kerja sistem

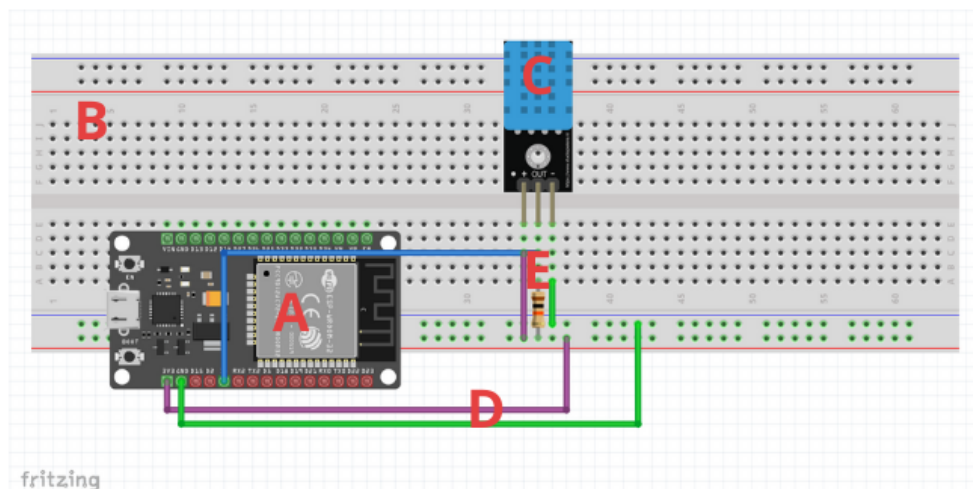
3.3.2 Desain Perakitan Alat

Pada tahap ini akan dibahas cara perangkat keras akan dipasang dan diintegrasikan dalam sistem, seperti sensor DHT11 dan ESP32. Hal ini mencakup penataan fisik dan hubungan antar komponen. Peneliti memastikan implementasi fisik sistem sesuai dengan desain yang diusulkan dengan menjelaskan proses perakitan secara detail. Sebelum perakitan dilakukan, pastikan bahan yang diperlukan tersedia terlebih dahulu, untuk detailnya sudah dijelaskan pada bagian analisis kebutuhan.

Setelah itu, proses perakitan ini diawali dengan memasang ESP32 dan DHT11 pada breadboard sejajar atau bersebelahan namun berjarak. Kemudian, hubungkan pin VCC

DHT11 dengan pin 3,3V modul ESP32 namun tidak secara langsung. Kabel pertama, letakan satu kaki kabelnya pada breadboard sejajar dibawah pin VCC kemudian satu kaki lainnya pada breadboard (+) sejajar di bawahnya. Kabel kedua, yang sama warnanya agar mudah memahaminya, satu kaki kabelnya diletakkan pada breadboard (+) sebelah kanannya diluar lurusan kaki DHT11 dan satu kaki lainnya pada pin 3,3V modul ESP32.

Selain itu, hubungkan juga pin GND DHT 11 dengan pin GND modul ESP32 namun tidak secara langsung. Kabel pertama, letakan satu kaki kabelnya pada breadboard sejajar dibawah pin GND DHT 11 kemudian satu kaki lainnya pada breadboard (-) sejajar di bawahnya. Kabel kedua, yang sama warnanya, satu kaki kabelnya diletakkan pada breadboard (-) sebelah kanannya diluar lurusan kaki DHT11 dan satu kaki lainnya pada pin GND modul ESP32. Kaki terakhir, hubungkan pin data DHT11 dengan pin D4 modul ESP32. Untuk resistornya, hubungkan satu kaki resistor 10K dengan pin data DHT11 kemudian satu kaki lainnya dari resistor tersebut dihubungkan dengan breadboard (+) dibagian sejajar bawahnya. Agar lebih jelas penjelasannya, peneliti memberikan gambaran detail rancangan rakitan secara visualnya di bawah ini.



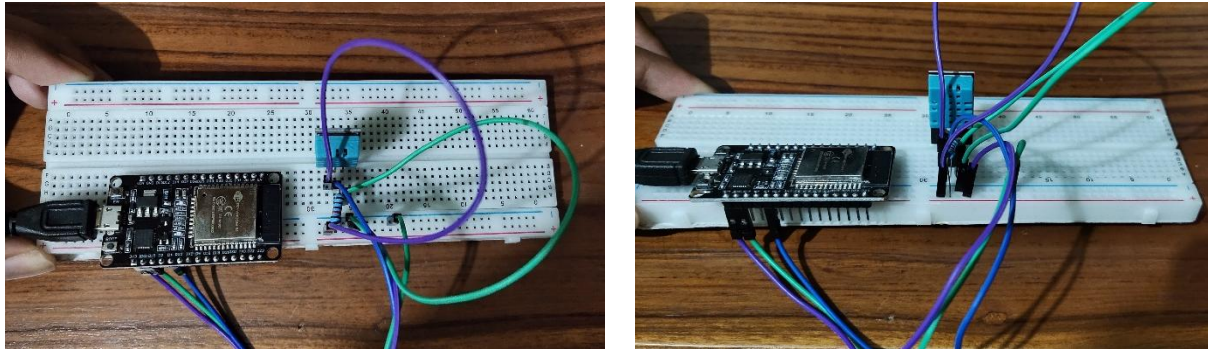
Keterangan:

- A. ESP32 Devkit 1**
- B. Breadboard**
- C. DHT 11**
- D. Kabel Jumper**
- E. Resistor 10k Ω**

Gambar 3.5 Rancangan desain rakitan alat

3.3.3 Perakitan

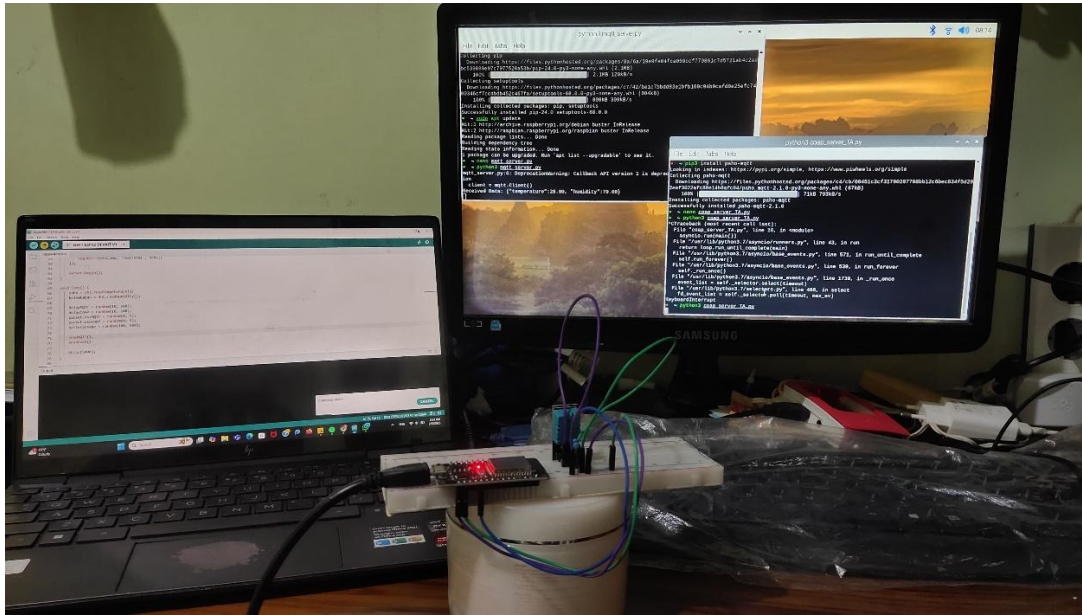
Setelah merancang sistem, peneliti melanjutkan ke tahap perakitan. Keseluruhan sistem dirangkai secara fisik sesuai dengan desain yang telah dirancang sebelumnya. Tahap ini mencakup pemasangan perangkat keras, pengecekan sensor dan aktuator berfungsi, serta integrasi perangkat lunak. Proses ini menjadi dasar untuk langkah pengujian berikutnya. Berikut peneliti sertakan gambar hasil rakitan sistem yang telah rangkai di bawah ini.



Gambar 3.6 Rakitan alat

3.4 Implementasi Protokol

Pada pengujian sistem rumah cerdas berdasarkan kinerja protokol CoAP dan MQTT ini, semua perangkat akan terhubung dalam satu jaringan WiFi yang sama. Proses pengujian dilakukan berkali-kali sampai memastikan tujuan pengujian sudah dapat disimpulkan sesuai dengan hasil perbandingan kinerja kedua protokol tersebut. Proses pengambilan data dilakukan dengan mengirimkan satu packet data setiap satu detik sebanyak 1000 packet data dari client ke server melalui masing-masing protokol. Hasil dari pengukuran tersebut akan disimpan terlebih dahulu, kemudian akan ditampilkan melalui serial monitor. Pastikan yang menjadi parameter dalam pengujian ini terukur dan terdata. Variabel-variabel tersebut akan diamati dan diukur selama pengujian dengan tujuan untuk mengevaluasi kinerja sistem rumah cerdas secara menyeluruh.



Gambar 3.7 Implementasi protokol

3.5 Analisis Hasil Pengukuran Kinerja dan Evaluasi Kinerja

Setelah menjalani tahap pengujian, langkah berikutnya adalah menganalisis hasil pengukuran kinerja sistem rumah cerdas yang menggunakan protokol CoAP dan MQTT. Hasil pengukuran tersebut diharapkan memberikan pemahaman yang lebih dalam tentang efektivitas dan efisiensi sistem dalam mengumpulkan, mengirimkan, dan mengolah data, termasuk evaluasi terhadap waktu respon sistem (delay), jumlah data yang berhasil dikirimkan (packet loss), serta kapasitas transfer data dalam satuan waktu (throughput). Evaluasi ini bertujuan untuk menilai sejauh mana sistem telah memenuhi tujuan penelitian dan ekspektasi yang telah ditetapkan, dengan membandingkan hasil pengukuran dengan standar atau kriteria yang telah ditetapkan sebelumnya. Hasil analisis dan evaluasi ini menjadi dasar untuk menarik kesimpulan mengenai kinerja protokol CoAP dan MQTT dalam konteks aplikasi sistem rumah cerdas, serta memberikan rekomendasi untuk perbaikan atau pengembangan lebih lanjut pada sistem yang telah diuji.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil Pengujian Sistem

Bagian ini menyajikan hasil pengujian sistem yang berisi laporan *output* (keluaran) dari implementasi sistem yang telah dikembangkan sebelumnya. Pada tahap ini memiliki peran penting dalam proses pengembangan sistem atau proyek akhir, karena bertujuan untuk memastikan bahwa pengembangan program yang dibuat dapat berfungsi sesuai dengan ketentuan yang telah direncanakan sebelumnya. Pengujian ini dilakukan pada dua kondisi jaringan, yaitu kondisi jaringan baik dan kondisi jaringan buruk. Pengambilan sampel jaringan buruk dilakukan di daerah pegunungan, tepatnya di Lamuk, Katekan, Temanggung, yang berada di kaki Gunung Sindoro. Selain itu, melalui pengujian ini, dilakukan analisis perbandingan kinerja antara kedua protokol yang digunakan yaitu MQTT dan CoAP berdasarkan parameter yang telah ditentukan, seperti delay, packet loss, dan throughput. Dengan begitu, melalui hasil pengujian sistem ini dapat memberikan wawasan yang lebih mendalam mengenai efisiensi dan efektivitas masing-masing protokol dalam konteks sistem rumah cerdas yang dikembangkan.

4.1.1 Hasil Pengujian pada Sensor DHT11

Pengujian sensor DHT11 dalam penelitian ini bertujuan untuk mengukur tingkat akurasi dan stabilitas sensor dalam membaca parameter suhu dan kelembaban. Pengukuran dilakukan dengan membaca nilai suhu dan kelembaban dari sensor secara periodik, kemudian mengirimkannya ke server menggunakan protokol komunikasi yang diuji, yaitu MQTT dan CoAP. Data hasil pembacaan sensor menjadi contoh data dalam pengujian ini yang akan dikirimkan ke masing-masing protokol untuk diuji kualitas kinerjanya. Berdasarkan pengujian yang telah dilakukan, sensor DHT11 menunjukkan hasil pembacaan yang konsisten dengan rata-rata kenaikan dan penurunan yang tidak signifikan.

Data Sensor DHT11

Temperature: 30.80 °C

Humidity: 60.00 %

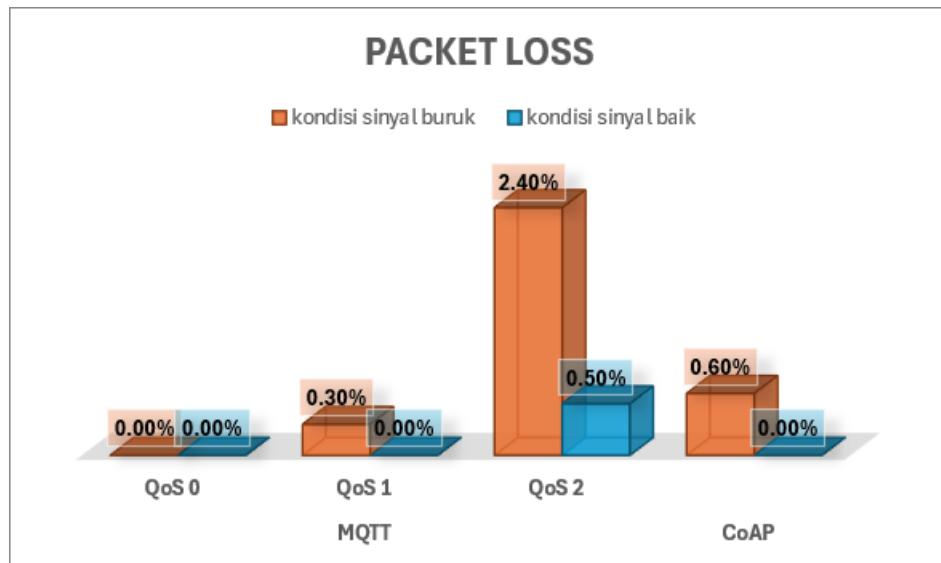
Gambar 4.1 Hasil pengujian DHT 11

4.1.2 Hasil Pengujian pada Protokol berdasarkan Packet Loss

Packet loss dalam penelitian ini diukur dengan membandingkan jumlah paket yang dikirimkan oleh perangkat ESP32 dengan jumlah paket yang diterima oleh server. Berdasarkan hasil pengujian, pada kondisi sinyal baik, seluruh protokol menunjukkan hasil packet loss yang sangat rendah. Untuk MQTT QoS 0, QoS 1, dan CoAP, tidak terdapat packet loss sama sekali yaitu sebesar 0.00%, sedangkan pada QoS 2 mengalami packet loss sebesar 0.50%. Hal ini menunjukkan bahwa dalam kondisi jaringan stabil, protokol mampu mengirimkan data dengan baik tanpa banyak kehilangan paket.

Sedangkan pada kondisi sinyal buruk, mulai terlihat adanya peningkatan packet loss, khususnya pada MQTT QoS 2 yang mencapai 2.40%. Sementara itu, MQTT QoS 1 mengalami packet loss sebesar 0.30%, dan CoAP sebesar 0.60%. Adapun MQTT QoS 0 tetap menunjukkan hasil 0.00% meskipun pada jaringan buruk, yang menunjukkan bahwa pengiriman data terus dilakukan tanpa mekanisme pengulangan ketika paket hilang.

Secara keseluruhan, peningkatan packet loss pada QoS 2 disebabkan oleh mekanisme retransmisi yang membutuhkan koneksi stabil, sehingga saat kondisi jaringan buruk justru meningkatkan peluang kegagalan pengiriman. Sedangkan pada CoAP, meskipun berbasis UDP tanpa retransmisi default, namun penggunaan mode confirmable tetap mampu menjaga reliabilitas pengiriman meski pada jaringan kurang stabil. Dengan demikian, baik MQTT maupun CoAP masih menunjukkan performa yang cukup andal dalam mengatasi packet loss, meskipun performa QoS 2 pada MQTT perlu menjadi perhatian saat jaringan kurang stabil. Hasil tersebut dapat dilihat pada gambar berikut.

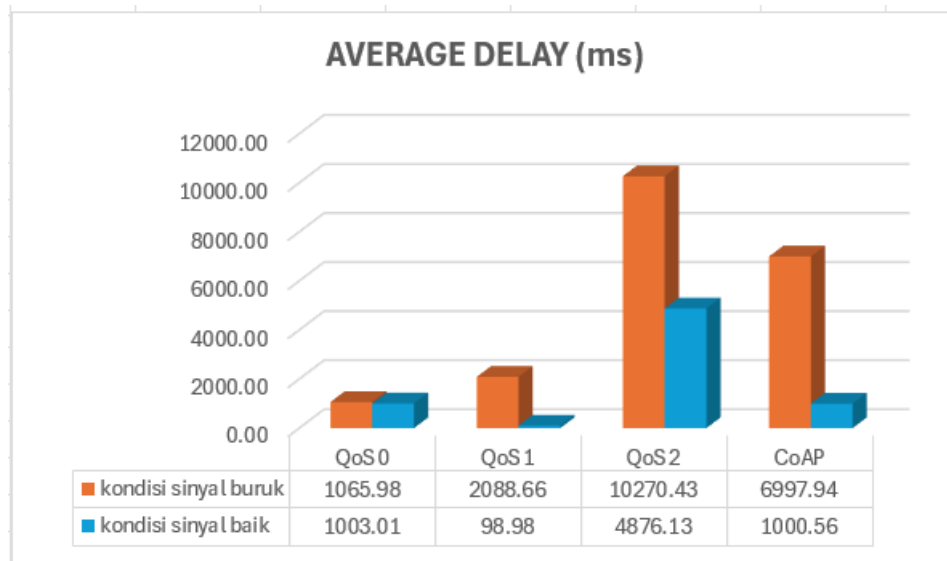


Gambar 4.2 Hasil perbandingan packet loss

4.1.3 Hasil Pengujian pada Protokol berdasarkan Delay

Pengujian delay dalam penelitian ini dilakukan dengan mengukur waktu yang dibutuhkan untuk mengirimkan suatu paket dari perangkat ESP32 hingga diterima oleh server. Delay dihitung dengan mencatat timestamp pada saat pengiriman dan penerimaan data, kemudian dihitung selisih waktu antara keduanya. Berdasarkan hasil pengukuran, pada kondisi sinyal baik, protokol MQTT dengan QoS 1 menghasilkan delay terendah yaitu 98.98 ms, menunjukkan performa pengiriman data yang cepat dan efisien berkat mekanisme pengiriman tepat satu kali. Sementara itu, QoS 0 mencatat delay sebesar 1003.01 ms, QoS 2 sebesar 4876.13 ms, dan CoAP sebesar 1000.56 ms. Hal ini menunjukkan bahwa semakin tinggi tingkat QoS, maka kompleksitas pengiriman meningkat dan berdampak pada kenaikan delay.

Pada kondisi sinyal buruk, terjadi peningkatan delay secara keseluruhan. Delay tertinggi dicapai oleh MQTT QoS 2 sebesar 10270.43 ms akibat proses retransmisi berulang yang memperlambat pengiriman ketika kualitas jaringan menurun. Sedangkan QoS 0 dan QoS 1 masing-masing mengalami delay sebesar 1065.98 ms dan 2088.66 ms. Untuk protokol CoAP, delay yang terjadi sebesar 6997.94 ms, yang menunjukkan bahwa meskipun CoAP berbasis UDP dan tidak memiliki proses konfirmasi seperti MQTT, kondisi jaringan yang buruk tetap mempengaruhi kecepatannya. Secara umum, dapat disimpulkan bahwa QoS 1 menjadi pilihan yang paling stabil pada kondisi jaringan baik maupun buruk, sedangkan QoS 2 cenderung rentan mengalami lonjakan delay pada saat kualitas jaringan menurun. Hasil perbandingan dapat dilihat pada gambar 4.3 di bawah ini.



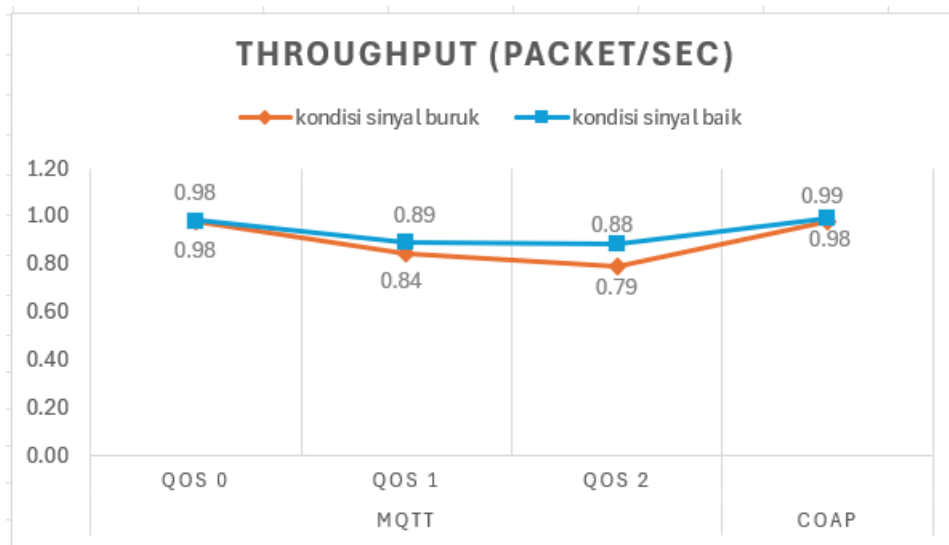
Gambar 4.3 Hasil perbandingan delay

4.1.4 Hasil Pengujian pada Protokol berdasarkan Throughput

Throughput dalam penelitian ini dihitung berdasarkan jumlah data yang berhasil dikirimkan dalam satuan waktu tertentu. Pengukuran dilakukan dengan mencatat ukuran paket yang dikirimkan dan menghitung jumlah paket yang diterima dalam interval waktu tertentu. Berdasarkan hasil pengujian, pada kondisi sinyal baik, nilai throughput tertinggi dicapai oleh protokol CoAP sebesar 0.99 packet/sec, disusul oleh MQTT QoS 0 sebesar 0.98 packet/sec. Sedangkan pada QoS 1 dan QoS 2 masing-masing menghasilkan throughput sebesar 0.89 packet/sec dan 0.88 packet/sec. Hal ini menunjukkan bahwa semakin tinggi tingkat QoS pada MQTT, maka throughput cenderung menurun akibat adanya tambahan proses pengiriman ulang dan konfirmasi.

Pada kondisi sinyal buruk, terjadi penurunan throughput pada seluruh pengujian. CoAP tetap menunjukkan performa yang baik dengan throughput sebesar 0.98 packet/sec. Sedangkan pada MQTT QoS 0 throughput tetap stabil di angka 0.98 packet/sec, QoS 1 turun menjadi 0.84 packet/sec, dan QoS 2 mengalami penurunan lebih besar hingga 0.79 packet/sec. Penurunan throughput ini menggambarkan pengaruh dari kestabilan jaringan dan beban proses retransmisi yang semakin berat pada QoS yang lebih tinggi. Secara keseluruhan, protokol CoAP menunjukkan performa throughput yang paling stabil pada kedua kondisi jaringan, sementara pada protokol MQTT, penggunaan QoS yang lebih tinggi cenderung menurunkan nilai

throughput, terutama saat kondisi jaringan memburuk. Hasil tersebut dapat dilihat pada gambar berikut.



Gambar 4.4 Hasil perbandingan throughput

Berikut tabel ringkasan perbandingan hasil pengujian dari protokol MQTT dan CoAP.

Tabel 4.1 Ringkasan Perbandingan Hasil Pengujian

Parameter	MQTT	CoAP
Packet Loss	Memiliki hasil packet loss yang bervariasi tergantung QoS. Pada QoS 0 tidak terjadi packet loss, sedangkan pada QoS 1 dan QoS 2 terjadi peningkatan packet loss saat kondisi sinyal buruk.	Memiliki hasil packet loss yang rendah dan stabil pada kedua kondisi jaringan, baik sinyal baik maupun buruk.
Delay	Delay bervariasi sesuai QoS. QoS 0 dan QoS 1 memiliki delay yang lebih rendah, namun pada QoS 2 terjadi peningkatan delay yang cukup signifikan, khususnya saat kondisi sinyal buruk.	Memiliki delay yang cenderung lebih tinggi dibandingkan MQTT, terutama dalam kondisi jaringan buruk. Namun, relatif stabil di kedua kondisi jaringan.
Throughput	Throughput menurun seiring peningkatan QoS, dengan penurunan	Memiliki throughput paling stabil dan tinggi pada kedua kondisi

	paling besar terjadi pada QoS 2 saat sinyal buruk.	jaringan karena proses pengiriman data yang sederhana tanpa overhead koneksi.
--	--	---

4.2 Pembahasan Hasil Pengujian

Pembahasan hasil pengujian disini akan membahas mengenai kode program yang dibutuhkan dan digunakan dari seluruh proses penelitian.

4.2.1 Pembahasan Kode Program ESP32 MQTT Client

```
#include <WiFi.h>
#include <MQTT.h>
#include <DHT.h>
#include <SPIFFS.h>
#include "time.h"

#define DHTPIN 4
#define DHTTYPE DHT11

const char* ssid = "(nama wifi)"; //diubah sesuai nama wifi yang digunakan
const char* password = "(pass wifi)"; // diubah sesuai pass wifi yang digunakan
const char* mqtt_server = "192.168.100.131"; // IP Raspberry Pi
const int qosLevel = 1; // Ubah ke 0, 1, atau 2 sesuai pengujian

WiFiClient espClient;
PubSubClient client(espClient);
DHT dht(DHTPIN, DHTTYPE);

// Variabel pengukuran
const int total_packets = 1000;
int packets_received = 0;
unsigned long start_time, end_time;
unsigned long send_times[total_packets];
```

```

unsigned long receive_times[total_packets];
int packet_index = 0;
int packetCount = 0;

```

Program ini diawali dengan menambahkan library yang dibutuhkan. Library `#include <WiFi.h>` yang digunakan untuk ESP32 agar dapat terhubung ke jaringan WiFi. Library `#include <MQTT.h>` berfungsi untuk mengimpor library MQTT ke dalam program, sehingga ESP32 dapat menggunakan fungsi-fungsi yang berkaitan dengan komunikasi MQTT seperti koneksi ke broker, publish, subscribe, dan pengolahan pesan. Library yang digunakan untuk membaca data suhu dan kelembapan dari sensor DHT11 diantaranya, `#include <DHT.h>`, `#define DHTPIN 4`, `#define DHTTYPE DHT11` yang terhubung ke pin GPIO 4. Selain itu, `#include <SPIFFS.h>` digunakan untuk menyimpan hasil pengukuran ke dalam file sistem internal ESP32, dan `#include "time.h"` digunakan untuk menyinkronkan waktu dari internet (NTP). Kemudian, dilanjutkan ke bagian konfigurasi WiFi dan MQTT, menggunakan SSID dan password agar ESP32 bisa terhubung. Alamat IP dari MQTT Broker (pada Raspberry Pi) disimpan dalam variabel `mqtt_server`. Pada bagian variabel pengukuran diperlukan guna melakukan pengukuran kinerja dari protokol MQTT. Total data yang dikirim sebanyak 1000 paket data. Waktu pengiriman dan penerimaan masing-masing paket disimpan dalam array, serta jumlah paket yang berhasil diterima kemudian akan dihitung untuk keperluan penghitungan matriks pengujian diantaranya, packet loss, delay, dan throughput.

```

void messageReceived(String &topic, String &payload) {
    if (packet_index < total_packets) {
        receive_times[packet_index] = millis();
        packets_received++;
        packet_index++;
    }
}

void saveTestResults(float packet_loss, float throughput, float avg_delay) {
    File file = SPIFFS.open("/test_results.txt", FILE_WRITE);
    if (file) {
        file.println("");
    }
}

```

```

    file.print(" \"packetReceived\": "); file.println(packets_received);
    file.print(" \"packetLoss\": "); file.println(packet_loss, 2);
    file.print(" \"throughput\": "); file.println(throughput, 2);
    file.print(" \"delay\": "); file.println(avg_delay, 2);
    file.println("{}");
    file.close();
} else {
    Serial.println("Failed to open file for writing");
}
}

String getCurrentTimeWithMillis() {
    struct timeval tv;
    gettimeofday(&tv, NULL);

    struct tm *timeinfo = localtime(&tv.tv_sec);
    char buffer[30];
    strftime(buffer, sizeof(buffer), "%Y-%m-%d %H:%M:%S", timeinfo);

    char fullTime[40];
    snprintf(fullTime, sizeof(fullTime), "%s.%03ld", buffer, tv.tv_usec / 1000);
    return String(fullTime);
}

void connectMQTT() {
    while (!client.connect("ESP32Client")) {
        Serial.print(".");
        delay(1000);
    }
    Serial.println("\nMQTT Connected!");
    client.subscribe("sensor/dht11", qosLevel);
}

```

Fungsi `messageReceived()` dipanggil saat ESP32 menerima pesan dari MQTT broker. Di dalamnya terdapat pencatatan waktu penerimaan data serta penambahan jumlah paket yang diterima. Kemudian dilanjutkan pada fungsi `saveTestResults()`, guna menyimpan hasil pengukuran performa dalam format JSON ke dalam file di SPIFFS. Ini berguna untuk menyimpan data hasil pengujian agar dapat diakses dan ditampilkan kemudian, misalnya melalui Web Server ESP32. Fungsi `getCurrentTimeWithMillis()` ini digunakan untuk mengambil waktu saat ini dengan resolusi hingga milidetik, berdasarkan waktu yang telah disinkronkan dari NTP. Ini berguna untuk mencatat waktu pengiriman data secara akurat.

```
void setup() {
  Serial.begin(115200);

  // Koneksi WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected!");

  configTime(25200, 0, "pool.ntp.org", "time.nist.gov"); // 25200 = UTC+7 (WIB)
  Serial.println("Menyinkronkan waktu...");
  struct tm timeinfo;
  while (!getLocalTime(&timeinfo) || timeinfo.tm_year + 1900 < 2020) {
    Serial.println("Menunggu sinkronisasi waktu...");
    delay(1000);
  }
  Serial.println("Waktu berhasil tersinkronisasi.");

  // MQTT Setup
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}
```

```

while (!client.connected()) {
  if (client.connect("ESP32Client")) {
    Serial.println("MQTT Connected");
    client.subscribe("sensor/dht11");
  } else {
    delay(1000);
  }
}

// DHT11 Setup
dht.begin();
start_time = millis();

// Setup SPIFFS (file system)
if (!SPIFFS.begin(true)) {
  Serial.println("Failed to mount file system");
  return;
}
}

```

Bagian kode program di atas merupakan bagian `setup()` yang berfungsi menjalankan inisialisasi awal seperti koneksi WiFi, sinkronisasi waktu, koneksi ke broker MQTT, inisialisasi sensor DHT11, dan inisialisasi SPIFFS. Kemudian pada sinkronisasi waktu dilakukan dengan `configTime()` dan pengecekan waktu lokal untuk memastikan sinkronisasi berhasil sebelum data mulai dikirim. Terakhir, kode program di bawah ini merupakan bagian `loop()` berfungsi untuk mengirimkan data suhu dan kelembapan setiap 1000 ms sebanyak 1000 kali. Setiap data yang dikirim dicatat waktunya, dan saat data diterima kembali, waktu penerimaan juga dicatat. Kode `long rssi = WiFi.RSSI()` ini berfungsi untuk mengetahui kekuatan jaringan selama proses pengujian. Setelah semua data dikirim, dilakukan perhitungan packet loss, throughput, dan rata-rata delay, lalu disimpan ke file SPIFFS dan ditampilkan di Serial Monitor. Setelah selesai, ESP32 dihentikan dengan `while(1)`.

```

void loop() {

```

```

if (!client.connected()) {
    client.connect("ESP32Client");
}
client.loop();

// Pengujian MQTT
if (packetCount < total_packets) {
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();

    long rssi = WiFi.RSSI();
    Serial.println("Signal strength (RSSI): " + String(rssi) + " dBm");

    if (!isnan(temperature) && !isnan(humidity)) {
        String payload = "{\"temp\": " + String(temperature) + ", \"hum\": " + String(humidity)
+ "}";
        String timestamp = getCurrentTimeWithMillis();

        Serial.print("Sending data at ");
        Serial.print(timestamp);
        Serial.print(": ");
        Serial.println(payload);

        send_times[packetCount] = millis();
        client.publish("sensor/dht11", payload.c_str());
        packetCount++;
    }
    delay(1000);
} else {
    end_time = millis();

    // Hitung hasil pengujian
    float packet_loss = ((total_packets - packets_received) / (float)total_packets) * 100;

```

```

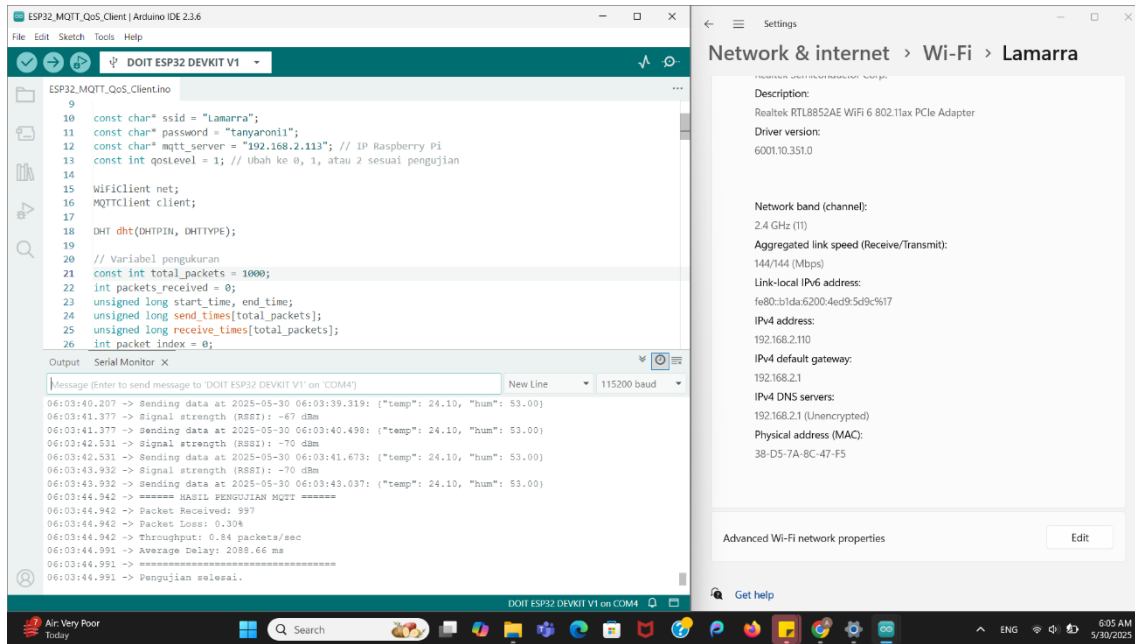
float throughput = packets_received / ((end_time - start_time) / 1000.0);
float total_delay = 0;
int valid_packets = 0;
for (int i = 0; i < packet_index; i++) {
    if (receive_times[i] > send_times[i]) {
        total_delay += (receive_times[i] - send_times[i]);
        valid_packets++;
    }
}
float avg_delay = (valid_packets > 0) ? total_delay / (float)valid_packets : 0;

// Simpan hasil pengujian ke file SPIFFS
saveTestResults(packet_loss, throughput, avg_delay);

// Cetak hasil pengukuran
Serial.println("===== HASIL PENGUJIAN MQTT =====");
Serial.print("Packet Received: "); Serial.println(packets_received);
Serial.print("Packet Loss: "); Serial.print(packet_loss); Serial.println("%");
Serial.print("Throughput: "); Serial.print(throughput); Serial.println(" packets/sec");
Serial.print("Average Delay: "); Serial.print(avg_delay); Serial.println(" ms");
Serial.println("=====");

// Selesai, hentikan pengujian
Serial.println("Pengujian selesai.");
while (1); // Menghentikan ESP32 setelah pengiriman selesai
}
}

```



Gambar 4.5 Hasil kode program MQTT client

4.2.2 Pembahasan Kode Program Raspberry Pi MQTT Server

```
import paho.mqtt.client as mqtt
import json
from datetime import datetime

MQTT_BROKER = "192.168.100.131"
MQTT_PORT = 1883
MQTT_TOPIC = "sensor/dht11"
data_log = []
```

Kode program di atas dimulai dengan mengimpor beberapa library, seperti library `paho.mqtt.client` digunakan untuk mengatur koneksi dan komunikasi dengan MQTT broker. Library `json` digunakan untuk memproses data dalam format JSON yang diterima dari ESP32. Terakhir ada library `datetime`, yang digunakan untuk mencatat waktu penerimaan data secara presisi hingga milidetik. Kemudian dilanjutkan ke bagian konfigurasi koneksi ke MQTT broker. Terdiri dari IP address dari broker (Raspberry Pi), port default MQTT yaitu 1883, dan topik dari MQTT yang akan disubscribe oleh client untuk menerima data dari ESP32.

```

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT Broker!")
        client.subscribe(MQTT_TOPIC)
    else:
        print("Failed to connect, return code ", rc)

def on_message(client, userdata, msg):
    global data_log
    payload = msg.payload.decode("utf-8")
    try:
        data = json.loads(payload)
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S.%f")[:-3]
        print(f'Received at {timestamp}: {msg.payload.decode()}')
        data_log.append(data)
    except Exception as e:
        print("Error parsing JSON", e)

```

Kode program di atas menunjukkan terdapat dua fungsi yang digunakan. Pertama, fungsi `on_connect()` yang akan dijalankan secara otomatis saat client berhasil atau gagal terhubung ke broker. Jika kode `rc` bernilai 0, maka koneksi sukses dan client langsung subscribe ke topik yang ditentukan. Jika tidak, akan dicetak pesan kegagalan sesuai dengan kode return. Kedua, fungsi `on_message()` yang dijalankan setiap kali ada pesan masuk dari topik yang disubscribe. Kemudian payload pesan didekode menjadi string, lalu di-parse menggunakan `json.loads()`. Waktu saat pesan diterima dicatat menggunakan `datetime.now()` dengan format hingga milidetik. Dari hasil tersebut, data yang diterima ditampilkan di terminal dan juga disimpan dalam list `data_log`. Jika terjadi kesalahan parsing, maka akan ditampilkan pesan error.

```

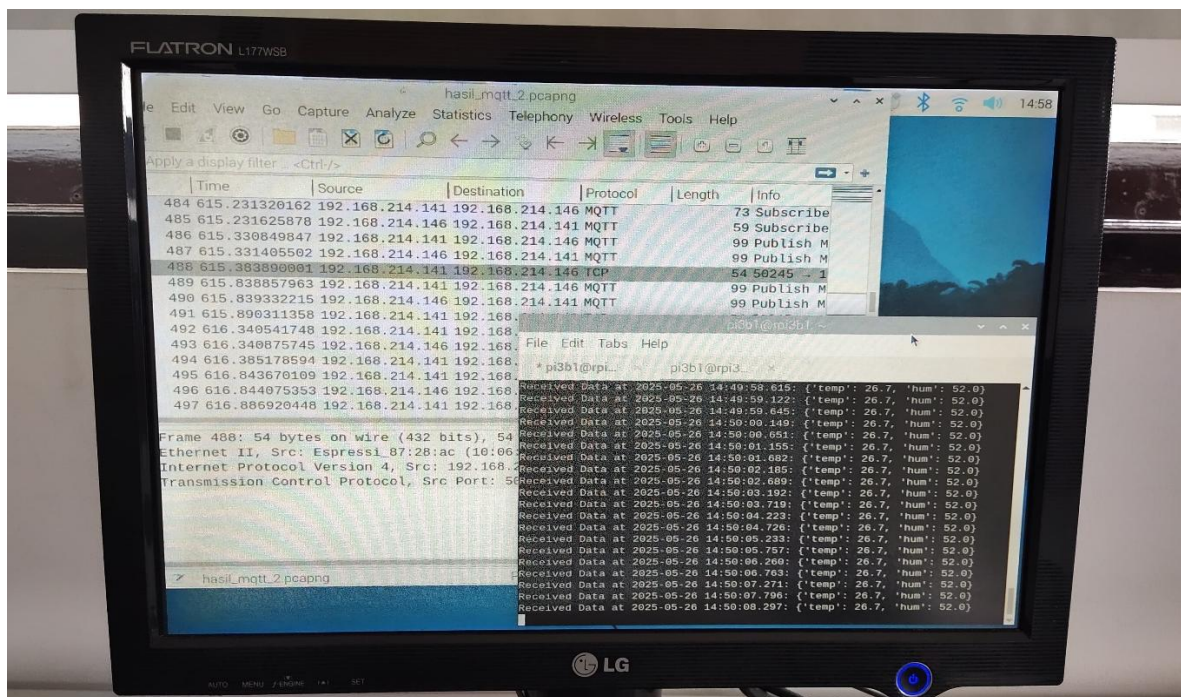
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect(MQTT_BROKER, MQTT_PORT, 60)

```

```
client.loop_forever()
```

Berdasarkan kode program di atas objek MQTT client dibuat dengan `mqtt.Client()`. Kemudian fungsi `on_connect` dan `on_message` dihubungkan ke objek tersebut agar bisa menangani koneksi dan pesan secara otomatis. Pada bagian dua baris terakhir berfungsi untuk menghubungkan client ke broker menggunakan IP dan port yang telah ditentukan sebelumnya. Fungsi `loop_forever()` membuat client terus berjalan dan mendengarkan pesan secara terus menerus tanpa henti.



Gambar 4.6 Hasil kode program MQTT server

4.2.3 Pembahasan Kode Program ESP32 CoAP Client

```
#include <WiFi.h>
#include <coap-simple.h>
#include <DHT.h>
#include <SPIFFS.h>

#define WIFI_SSID "(nama wifi)" // diubah sesuai nama wifi yang digunakan
#define WIFI_PASSWORD "(pass wifi)" // diubah sesuai pass wifi yang digunakan
```

```

#define SERVER_IP "192.168.100.131"
#define SERVER_PORT 5683

#define DHTPIN 4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

WiFiUDP udp;
Coap coap(udp);

// Variabel pengukuran
const int total_packets = 1000;
int packets_received = 0;
unsigned long send_times[total_packets];
unsigned long receive_times[total_packets];
int packet_index = 0;
unsigned long start_time, end_time;
int packetCount = 0;
bool testCompleted = false; // Tambahkan penanda pengujian selesai

```

Seperti pada mqtt, bagian awal program ini dimulai dengan import library yang diperlukan. Kode `#include <WiFi.h>` berfungsi untuk koneksi WiFi, `#include <coap-simple.h>` untuk komunikasi menggunakan protokol CoAP, `#include <DHT.h>` untuk pembacaan sensor suhu dan kelembaban DHT11, serta `#include <SPIFFS.h>` diperlukan agar hasil pengujian dapat disimpan ke memori internal ESP32. Beberapa variabel konstan didefinisikan di awal seperti nama dan password WiFi, alamat IP dan port server CoAP, serta pin sensor DHT11. Setelah itu, melakukan inisialisasi pada objek sensor DHT11, objek CoAP dan komunikasinya yaitu UDP agar siap digunakan. Dalam proses pengukuran kinerja protokol CoAP, digunakan variabel-variabel pengukuran untuk mencatat data. Sebanyak 1000 paket data dikirimkan selama pengujian berlangsung. Waktu saat pengiriman dan penerimaan setiap paket disimpan ke dalam array, sementara jumlah paket yang berhasil diterima dicatat untuk digunakan dalam perhitungan parameter kinerja seperti packet loss, delay, dan throughput.

```

void saveTestResults(float packet_loss, float throughput, float avg_delay) {
    File file = SPIFFS.open("/test_results.txt", FILE_WRITE);
    if (!file) {
        Serial.println("Failed to open file for writing");
        return;
    }
    file.println("{}");
    file.print("  \"packetReceived\": "); file.println(packets_received);
    file.print("  \"packetLoss\": "); file.println(packet_loss, 2);
    file.print("  \"throughput\": "); file.println(throughput, 2);
    file.print("  \"delay\": "); file.println(avg_delay, 2);
    file.println("{}");
    file.close();
    Serial.println("Hasil pengujian disimpan ke SPIFFS");
}

```

```

void callback(CoapPacket &packet, IPAddress ip, int port) {
    //Serial.println("Received response from CoAP Server!");
    if (packets_received < total_packets) {
        receive_times[packet_index] = millis();
        packets_received++;
        packet_index++;
    }
}

```

```

String getCurrentTimeWithMillis() {
    struct timeval tv;
    gettimeofday(&tv, NULL);

    struct tm *timeinfo = localtime(&tv.tv_sec);
    char buffer[30];
    strftime(buffer, sizeof(buffer), "%Y-%m-%d %H:%M:%S", timeinfo);
}

```

```

char fullTime[40];
sprintf(fullTime, sizeof(fullTime), "%s.%03ld", buffer, tv.tv_usec / 1000);

return String(fullTime);
}

```

Dari kode tersebut, terdapat array dan variabel untuk mencatat waktu pengiriman dan penerimaan data sebanyak 1000 kali. Ini digunakan untuk menghitung nilai matriks pengujian diantaranya, packet loss, delay, dan throughput. Sebuah fungsi `saveTestResults()` dibuat untuk menyimpan hasil pengukuran dalam format JSON ke dalam file `test_results.txt` di SPIFFS. Fungsi ini akan dipanggil saat pengujian selesai. Kemudian, terdapat fungsi `callback()` yang akan dijalankan secara otomatis ketika ESP32 menerima respon dari server CoAP. Di dalamnya, waktu penerimaan akan dicatat dan jumlah paket yang berhasil diterima akan dihitung. Ada juga fungsi tambahan `getCurrentTimeWithMillis()` yang digunakan untuk mencetak timestamp waktu lengkap hingga milidetik.

```

void setup() {
  Serial.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnected to WiFi");

  dht.begin();
  coap.response(callback);

  configTime(25200, 0, "pool.ntp.org", "time.nist.gov");
  Serial.println("Menyinkronkan waktu...");
  struct tm timeinfo;
  while (!getLocalTime(&timeinfo) || timeinfo.tm_year + 1900 < 2020) {
    Serial.println("Menunggu sinkronisasi waktu...");
  }
}

```

```

    delay(1000);
}
Serial.println("Waktu berhasil tersinkronisasi.");

// Setup SPIFFS (file system)
if (!SPIFFS.begin(true)) {
    Serial.println("Failed to mount file system");
    return;
}

start_time = millis(); // Baru mulai pencatatan
}

```

Pada bagian fungsi `setup()`, program melakukan koneksi ke WiFi dan mencetak status koneksi. Setelah itu, sensor DHT11 dimulai dan ESP32 menunggu sinkronisasi waktu dari server agar dapat mencatat waktu secara presisi. SPIFFS juga diinisialisasi agar ESP32 siap menyimpan data hasil uji. Selain itu, waktu mulai pengujiannya dicatat pada bagian `start_time`. Setelah fungsi `setup` berhasil dijalankan, dilanjutkan dengan fungsi `loop()` di bawah, dimana proses pengujian berlangsung. Kode `long rssi = WiFi.RSSI()` ini berfungsi untuk mengetahui kekuatan jaringan selama proses pengujian. Looping akan terus dilakukan jika jumlah paket yang dikirim belum mencapai 1000. Sensor akan membaca suhu dan kelembapan, dan jika data valid, maka akan dikirim ke server CoAP menggunakan metode `put()` dan waktu pengiriman dicatat. Ini dilakukan setiap 1000 milidetik. Setelah semua paket dikirim, program menghitung hasil pengujian seperti packet loss, average delay, dan throughput. Hasilnya akan disimpan ke SPIFFS dan ditampilkan pada serial monitor. Program akan berhenti menghitung setelah pengujian selesai agar tidak mengulang proses pengiriman. Dengan demikian, kode ini memungkinkan ESP32 untuk secara otomatis menguji performa CoAP dengan mengukur parameter penting dan menyimpan hasilnya secara lokal.

```

void loop() {
    coap.loop(); // Pastikan bisa menerima response dari server

    // Pengujian CoAP

```

```

if (packetCount < total_packets) {
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();

    long rssi = WiFi.RSSI();
    Serial.println("Signal strength (RSSI): " + String(rssi) + " dBm");

    if (!isnan(temperature) && !isnan(humidity)) {
        String payload = "{\"temp\": " + String(temperature) + ", \"hum\": " + String(humidity)
+ "}";
        String timestamp = getCurrentTimeWithMillis();

        Serial.print("Sending data at ");
        Serial.print(timestamp);
        Serial.print(": ");
        Serial.println(payload);

        send_times[packetCount] = millis();
        int result = coap.put(SERVER_IP, SERVER_PORT, "sensor/dht11", payload.c_str());

        packetCount++;
    }

    delay(1000);
}
else if (!testCompleted) {
    end_time = millis(); // Selesai kirim semua packet, baru catat end time

    // Hitung hasil pengujian
    float packet_loss = ((total_packets - packets_received) / (float)total_packets) * 100;
    float throughput = packets_received / ((end_time - start_time) / 1000.0);
    float total_delay = 0;
    int valid_packets = 0;

```

```

for (int i = 0; i < packet_index; i++) {
    if (receive_times[i] > send_times[i]) {
        total_delay += (receive_times[i] - send_times[i]);
        valid_packets++;
    }
}

float avg_delay = (valid_packets > 0) ? total_delay / valid_packets : 0;

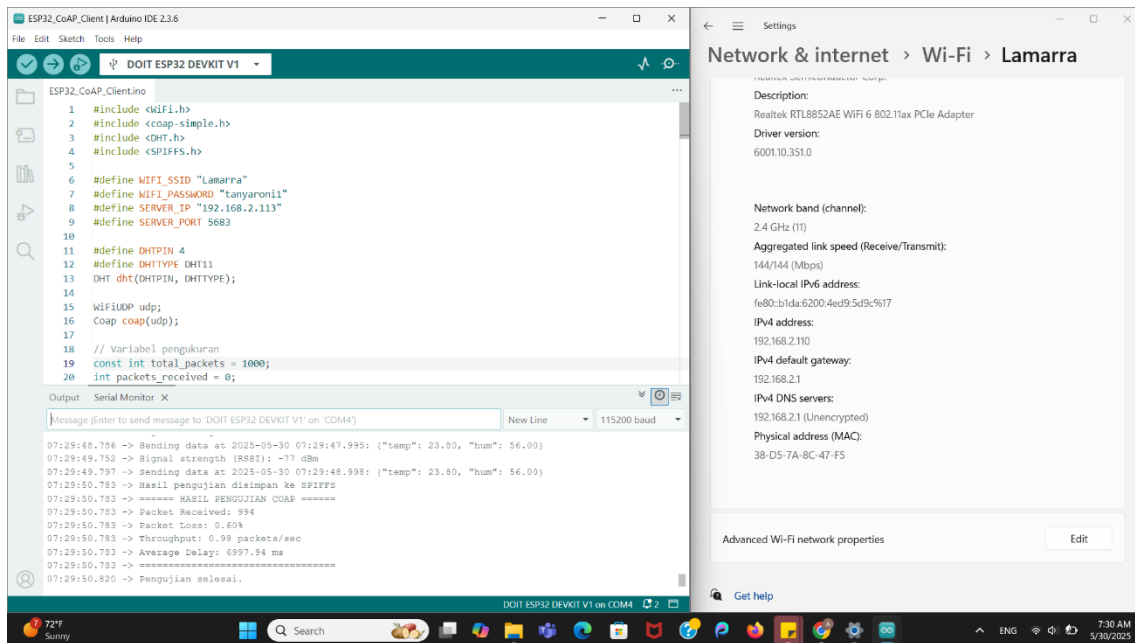
saveTestResults(packet_loss, throughput, avg_delay);

Serial.println("===== HASIL PENGUJIAN COAP =====");
Serial.print("Packet Received: "); Serial.println(packets_received);
Serial.print("Packet Loss: "); Serial.print(packet_loss); Serial.println("%");
Serial.print("Throughput: "); Serial.print(throughput); Serial.println(" packets/sec");
Serial.print("Average Delay: "); Serial.print(avg_delay); Serial.println(" ms");
Serial.println("=====");

Serial.println("Pengujian selesai.");

testCompleted = true; // Supaya tidak mengulang hitungan
}
}

```



Gambar 4.7 Hasil kode program CoAP client

4.2.4 Pembahasan Kode Program CoAP Server

```
from aiocoap import *
from aiocoap import resource, Context, Message
import asyncio
import json
import datetime
data_log = []
```

Kode program di atas digunakan untuk membuat CoAP server dengan python, yang bisa menerima data dari perangkat seperti ESP32. Kode pertama, dimulai dengan import beberapa library penting, seperti “from aiocoap import *” untuk menjalankan protokol CoAP, “import asyncio” untuk menangani proses asinkron, “import json” dan “import datetime” untuk mengolah data serta mencatat waktu. Pada baris terakhir terdapat sebuah list “data_log = []” yang juga disiapkan untuk menyimpan semua data yang diterima.

```
class CoAPServer(resource.Resource):
    async def render_put(self, request):
        global data_log
```

```

payload = request.payload.decode("utf-8")

now = datetime.datetime.now()
timestamp = now.strftime("%Y-%m-%d %H:%M:%S.%f")[:-3]
print (f'Received request at {timestamp}: {payload}')
try:
    data = json.loads(payload)
    data_with_time = {
        "timestamp": timestamp,
        "data": data
    }
    data_log.append(data_with_time)
    return Message(payload=b"Data received", content_format=0)
except Exception as e:
    print("Error parsing JSON", e)
    return Message(payload=b"Error parsing JSON", code=BAD_REQUEST)

```

Berdasarkan kode program di atas, ada kelas CoAPServer yang bertugas menangani data masuk. Fungsi `render_put()` di dalamnya akan otomatis dijalankan saat ada data yang dikirim ke server melalui metode PUT. Kemudian data diterima, dikonversi dari format JSON, saat waktu data diterima ditambahkan timestamp, dan disimpan dalam `data_log`. Jika berhasil, server akan membalas dengan pesan "Data received". Tapi kalau ada kesalahan dalam membaca data, maka server akan membalas dengan pesan error.

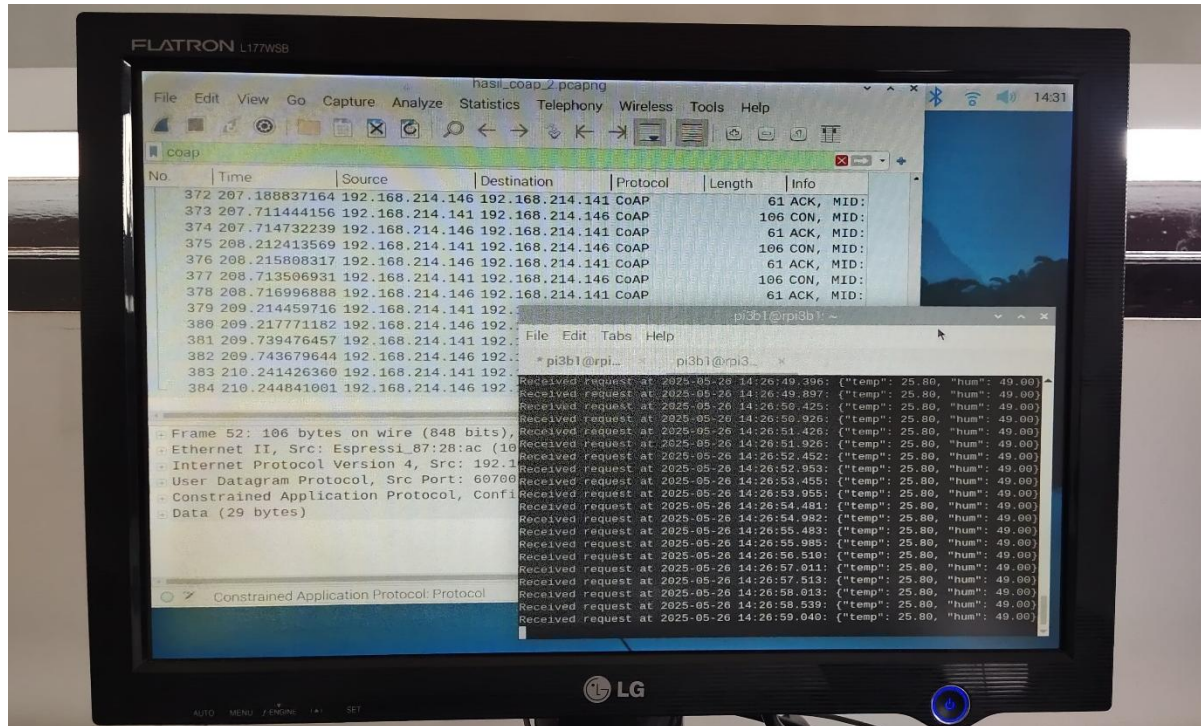
```

async def main():
    root = resource.Site()
    root.add_resource(["sensor", "dht11"], CoAPServer())
    await Context.create_server_context(root, bind="::", 5683)
    await asyncio.get_running_loop().create_future()

if __name__ == "__main__":
    asyncio.run(main())

```

Terakhir, terdapat fungsi main() yang berguna untuk menyiapkan struktur resource CoAP, yaitu endpoint sensor DHT11, lalu server dijalankan pada port 5683 dengan alamat bind “::” yang artinya menerima dari semua IP address. Program terus berjalan dengan asyncio agar server tetap aktif dan bisa terus menerima data dari client.



Gambar 4.8 Hasil kode program CoAP server

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini menghasilkan sistem pemantauan suhu ruangan pada rumah cerdas berbasis IoT dengan menggunakan protokol komunikasi MQTT dan CoAP. Sistem ini dirancang untuk menganalisis dan membandingkan kinerja kedua protokol berdasarkan packet loss, delay, dan throughput dalam proses pengiriman data sensor DHT11 dari ESP32 (client) ke Raspberry Pi (server). Seluruh proses pengujian dilakukan di jaringan lokal dengan dua kondisi jaringan, yaitu baik dan buruk dan hasilnya ditampilkan pada serial monitor dan juga dianalisis menggunakan bantuan Wireshark.

Dari seluruh tahapan penelitian, mulai dari perancangan, implementasi, hingga pengujian sistem, dapat disimpulkan bahwa sistem yang telah dirancang mampu mengukur suhu ruangan menggunakan sensor DHT11 dan mengirimkan data ke server menggunakan protokol MQTT dan CoAP. Berdasarkan hasil pengujian, delay protokol MQTT secara umum lebih rendah dibandingkan CoAP, khususnya pada QoS 0 dan QoS 1, sedangkan pada QoS 2 delay meningkat tajam, terutama saat kondisi sinyal buruk. Hal ini menunjukkan pengaruh mekanisme retransmisi dan jaminan pengiriman pada QoS yang lebih tinggi.

Untuk packet loss, secara umum kedua protokol menunjukkan hasil yang cukup baik, namun pada MQTT QoS 2 terjadi peningkatan packet loss sebesar 2.4% saat kondisi sinyal buruk, sementara pada CoAP packet loss tetap rendah dan stabil. Dari sisi throughput, CoAP mampu menghasilkan throughput yang lebih tinggi dan stabil di berbagai kondisi jaringan. Hal ini disebabkan karena CoAP menggunakan protokol UDP yang ringan, tanpa proses koneksi tambahan sebagaimana pada MQTT yang berbasis TCP.

Berdasarkan hasil tersebut, dapat disimpulkan bahwa pemilihan protokol komunikasi sangat bergantung pada kebutuhan dan kondisi jaringan. Dalam skenario sistem rumah cerdas skala kecil dengan keterbatasan sumber daya dan kondisi jaringan yang tidak selalu stabil, CoAP menjadi pilihan yang lebih unggul karena tetap stabil dan hemat sumber daya. Sedangkan, MQTT lebih cocok digunakan ketika dibutuhkan jaminan pengiriman data yang ketat dan stabilitas reliabilitas lebih diutamakan. Dengan demikian, hasil yang diperoleh mendukung bahwa pemilihan protokol komunikasi sangat bergantung pada konteks sistem yang digunakan.

5.2 Saran

- a. Sistem dapat dikembangkan lebih lanjut dengan menambahkan protokol komunikasi lain, seperti HTTP atau WebSocket, untuk memperluas analisis dan mengetahui perbedaan performa dibandingkan dengan MQTT dan CoAP.
- b. Penelitian selanjutnya dapat mengoptimalkan sistem untuk lingkungan dengan jaringan tidak stabil agar lebih adaptif terhadap kondisi jaringan.
- c. Melakukan pengujian dengan perangkat keras yang lebih bervariasi, seperti menggunakan sensor suhu yang lebih akurat atau board mikrokontroler dengan spesifikasi lebih tinggi untuk meningkatkan performa sistem.
- d. Mengembangkan sistem pemantauan dengan tampilan yang lebih interaktif, misalnya dengan menggunakan platform IoT berbasis cloud atau aplikasi mobile agar pengguna dapat mengakses data pengujian secara lebih fleksibel.
- e. Melakukan pengujian dalam skenario jaringan yang lebih kompleks, seperti pengujian pada jaringan dengan latensi tinggi.

DAFTAR PUSTAKA

- Almheiri, A., & Maamar, Z. (2021). IoT Protocols – MQTT versus CoAP. *Proceedings of the 4th International Conference on Networking, Information Systems & Security*, 1-5.
- Arhami, M., Husaini, Salahuddin, H., & Rudi F, F. Y. (2023). *Metodologi Penelitian untuk Teknologi Informasi dan Komputer* (1 ed.). (L. Mayasari, Ed.) Penerbit ANDI.
- Esposito, M., Belli, A., Palma, L., & Pierleoni, P. (2023). Design and Implementation of a Framework for Smart HomeAutomation Based on Cellular IoT, MQTT, andServerless Functions. *Sensors*, 23(9).
- Fahd A, A., & Ebtesam J, A. (2020, January). Securing Communication between Fog Computing and IoT. *Journal of Communications*, 15(1), 14-30. doi:10.12720/jcm
- Hendrik, Anjomshooa, A., & Tjoa, A. M. (2014). Towards Semantic Mashup Tools For Big Data Analysis. *Proceding of the Information & Communication Technology-EurAsia Conference 2014*, (pp. 100-145). Bali.
- Kusuma, F. T., Munadi, R., & Irawan, A. I. (2019). Analisa Dan Implementasi Smart Home Menggunakan Protokol Mqtt Dan Xmpp Berbasis Internet Of Things. *eProceedings of Engineering*, 6(2).
- Kusuma, F. T., Munady, R., & Irawan, A. I. (2019). Analisa dan Implementasi Smart Home Menggunakan Protokol MQTT dan XMPP Berbasis Internet of Things. *e-Proceeding of Engineering*, 6(2), 3157 - 3166.
- Naik, G. P., & Bapat, P. U. (2020). A Brief Comparative Analysis on Application Layer Protocols of Internet of Things: MQTT, CoAP, AMQP and HTTP. *International Journal of Computer Science and Mobile Computing*, 9(9), 135 - 141. Retrieved from <https://ijcsmc.com/docs/papers/September2020/V9I9202019.pdf>
- Patti, G., Leonardi, L., Testa, G., & Bello, L. L. (2024, April 15). PrioMQTT: A prioritized version of the MQTT protocol. *Computer Communications*, 220, 43-51. Retrieved from [https://pdf.sciencedirectassets.com/271515/1-s2.0-S0140366424X00067/1-s2.0-S0140366424001105/main.pdf?X-Amz-Security-Token=IQoJb3JpZ2luX2VjEKT%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJHMEUCIQDE6HppPjljULpOm9t79SS5Ri6A2rhy5NJO5T%2FVYeTFvAIgMaEN3Dup8N](https://pdf.sciencedirectassets.com/271515/1-s2.0-S0140366424X00067/1-s2.0-S0140366424001105/main.pdf?X-Amz-Security-Token=IQoJb3JpZ2luX2VjEKT%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJHMEUCIQDE6HppPjljULpOm9t79SS5Ri6A2rhy5NJO5T%2FVYeTFvAIgMaEN3Dup8N)

- Safitri, N. A., & Priambodo, A. S. (2023). MQTT and CoAP Communication Protocol Analysis in Internet of Things System for Strawberry Hydroponic Plants. *Journal of Robotics, Automation, and Electronics Engineering*, 1(1), 45-56.
- Seoane, V., Rubio, C. G., Almenares, F., & Campo, C. (2021). Performance evaluation of CoAP and MQTT with Security Support for IoT Environments. *Computer Networks*, 41-48. Retrieved from <https://doi.org/10.1016/j.comnet.2021.108338>
- Setiawan, A. M. (2013). *Integrated Framework For Business Process Complexity Analysis*. Retrieved from ECIS 2013 Completed Research: http://aisel.aisnet.org/ecis2013_cr/49
- Taufiq, H. (2015). *Argumentasi dan Validitas*. Yogyakarta: Darqin.
- Wahid, F. (2014). The Antecedents And Impacts of a Green Eprocurement Infrastructure: Evidence From The Indonesian Public Sector. *International Journal of internet Protocol Technology*, 7(4), 210-218.
- Yalçinkaya, F., Aydılek, H., Erten, M. Y., & İnanc, N. (2020). IoT based Smart Home Testbed using MQTT Communication Protocol. *International Journal of Engineering Research and Development*, 317-324.
- Zukhri, Z. (2014). *Algoritma Genetika: Metode Komputasi Evolusioner untuk Menyelesaikan Masalah Optimasi*. Yogyakarta: Andi Publisher.

