BAB 2
TINJAUAN PUSTAKA

2.1 Landasan Teori

2.1.1 *Internet of Things* (IoT) dan Keamanannya

Internet of Things (IoT) adalah konsep yang menghubungkan berbagai objek fisik seperti sensor, RFID, dan aktuator melalui jaringan internet untuk memungkinkan pengumpulan dan pertukaran data secara otomatis. IoT terdiri dari tiga lapisan utama perangkat fisik untuk penginderaan, gateway untuk transmisi data, dan cloud untuk pemrosesan data. Teknologi ini memberikan manfaat besar, seperti rumah pintar, pemantauan kesehatan jarak jauh, dan efisiensi dalam industri. Namun, IoT juga menghadapi tantangan keamanan, termasuk ancaman seperti serangan *Denial of Service* (DoS), penyadapan data, dan manipulasi perangkat. Untuk mengatasi ancaman tersebut, diperlukan mekanisme keamanan seperti enkripsi data, autentikasi, otorisasi, dan manajemen identitas. Keamanan yang kuat menjadi kunci bagi keberhasilan implementasi IoT dalam mendukung otomatisasi dan efisiensi di berbagai sektor kehidupan (Ernita Dewi Meutia, 2015).

2.1.2 ESP32

ESP32 adalah mikrokontroler 32-bit yang dikembangkan oleh Espressif Systems, dirancang untuk aplikasi *Internet of Things* (IoT). Mikrokontroler ini dilengkapi dengan konektivitas nirkabel bawaan, termasuk Wi-Fi dan Bluetooth, yang memungkinkannya terhubung dengan berbagai perangkat dan jaringan tanpa memerlukan modul tambahan. Salah satu keunggulan utama ESP32 adalah efisiensi dayanya. Mikrokontroler ini menawarkan berbagai mode tidur dan fitur manajemen daya yang membantu mengurangi konsumsi energi, sehingga cocok untuk proyek yang ditenagai baterai atau memiliki keterbatasan energi.

Gambar 2. 1 ESP 32 (Sumber: Hercog et al., 2023, hal. 6-20)

Gambar 2. 1 menampilkan perangkat mikrokontroler ESP32 yang menyediakan pin *General Purpose Input/Output* (GPIO) yang mendukung berbagai antarmuka seperti SPI, I2C, UART, dan PWM. Fitur ini memudahkan koneksi dan kontrol perangkat eksternal serta sensor, menjadikannya pilihan ideal untuk aplikasi IoT yang memerlukan interaksi dengan berbagai komponen (Hercog et al., 2023).

2.1.3 DHT11

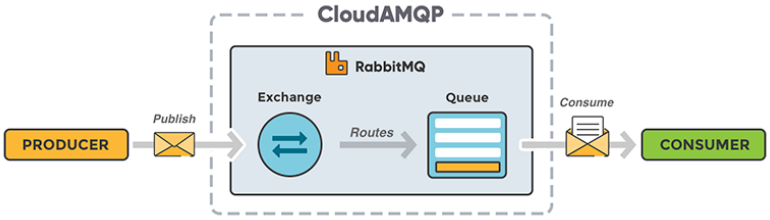
DHT11 adalah sensor yang sering digunakan untuk mengukur suhu dan kelembapan udara dalam berbagai aplikasi, terutama yang berbasis IoT. Sensor ini memiliki kemampuan untuk mengukur suhu dalam kisaran 0°C hingga 50°C dengan akurasi ±2°C, serta kelembapan dalam kisaran 20% hingga 90% RH dengan akurasi ±5% RH.

Gambar 2. 2 DHT 11 (Sumber: Hercog et al., 2023, hal. 6-20)

Gambar 2. 2 menampilkan bentuk fisik sensor DHT 11 yang memiliki ukuran kecil, konsumsi daya rendah, dan output digital yang sederhana, DHT11 mudah diintegrasikan dengan mikrokontroler seperti Arduino dan ESP32. Sensor ini bekerja dengan memanfaatkan elemen kapasitif untuk mendeteksi kelembapan dan termistor untuk suhu, lalu mengonversi data analog menjadi sinyal digital (Hercog et al., 2023).

2.1.4 Protokol AMQP

Advanced Message Queuing Protocol (AMQP) adalah protokol lapisan aplikasi yang dirancang untuk mendukung komunikasi antar proses melalui jaringan IP secara andal dan efisien. Protokol ini banyak digunakan dalam sistem antrian pesan karena kemampuannya untuk mengintegrasikan berbagai sistem dengan platform dan bahasa pemrograman yang berbeda. Komponen utama dalam AMQP meliputi *producer* sebagai pengirim pesan, *exchange* yang mengarahkan pesan berdasarkan aturan *routing*, *queue* sebagai tempat penyimpanan pesan sementara sebelum diproses, dan *consumer* yang mengambil serta memproses pesan dari antrian (Amqp et al., n.d.).



Gambar 2. 3 Topologi Protokol AMQP (Sumber: Prasetyo dan Nurwarsito, 2022. hal. 1630)

Gambar 2. 3 menunjukkan alur kerja protokol AMQP yang dimulai dari *producer* sebagai pihak yang mengirimkan pesan ke dalam sistem. Pesan tersebut kemudian diterima oleh *exchange* untuk diarahkan keberbagai *queue* berdasarkan tipe *routing*, seperti *direct exchange* yang mengarahkan pesan berdasarkan kecocokan *routing key*, *fanout exchange* yang mengirimkan pesan ke semua antrian yang terhubung tanpa mempertimbangkan *routing key*, *topic exchange* yang mendukung pola *routing key* dengan *wildcard*, serta *headers exchange* yang melakukan *routing* berdasarkan atribut *header* dalam pesan. Setelah pesan tersimpan di dalam *queue*, *consumer* akan mengambil dan memproses pesan tersebut.

Protokol ini memiliki keunggulan utama, yaitu interoperabilitas yang memungkinkan integrasi antar sistem yang berbeda, keandalan dalam pengiriman pesan, dan fleksibilitas untuk mendukung berbagai pola komunikasi, termasuk *point-to-point*, *publish/subscribe*, dan *request/response* (Wallarm, 2023). AMQP juga mendukung pengiriman pesan terenkripsi menggunakan TLS/SSL pada port 5671 untuk meningkatkan keamanan komunikasi. Dengan fitur-fitur ini, AMQP menjadi pilihan utama untuk implementasi sistem antrian pesan yang membutuhkan komunikasi yang andal, fleksibel, dan interoperabilitas tinggi (Amqp et al., n.d.).

2.1.5 RabbitMQ

RabbitMQ adalah perangkat lunak *open-source* yang berfungsi sebagai *message* broker, dirancang untuk mendukung komunikasi antar aplikasi secara asinkron menggunakan protokol *Advanced Message Queuing Protocol* (AMQP). Dikembangkan dengan bahasa pemrograman Erlang, RabbitMQ mengelola pengiriman pesan antara komponen-komponen dalam sistem terdistribusi. Fungsi utamanya meliputi penerimaan pesan dari *producer*, penyimpanan pesan dalam

antrian (*queue*), dan pengiriman pesan ke *Consumer* sesuai dengan aturan yang ditentukan. RabbitMQ mendukung berbagai pola komunikasi, seperti *point-to-point, publish/subscribe,* dan *request/response*, sehingga dapat memenuhi kebutuhan sistem yang kompleks. Keunggulan RabbitMQ meliputi interoperabilitas, memungkinkan integrasi dengan berbagai bahasa pemrograman dan *platform*, melalui mekanisme pengiriman pesan yang memastikan pesan sampai dengan tepat serta fleksibilitas dalam mendukung berbagai jenis komunikasi (Biznet Gio., 2024.).

2.1.6 Erlang

Erlang adalah bahasa pemrograman yang dikembangkan oleh Ericsson pada tahun 1980-an, dirancang khusus untuk sistem telekomunikasi yang membutuhkan keandalan tinggi, konkurensi, dan toleransi kesalahan (*fault tolerance*). Bahasa ini memungkinkan pengelolaan ribuan proses ringan yang berjalan secara bersamaan dan independen, dengan komunikasi antar proses dilakukan melalui pertukaran pesan. Salah satu fitur unggulan Erlang adalah *hot code swapping*, yang memungkinkan pembaruan kode tanpa menghentikan sistem, sehingga meminimalkan *downtime*. Erlang juga dilengkapi dengan kerangka kerja *Open Telecom Platform* (OTP), yang menyediakan alat bantu untuk membangun aplikasi terdistribusi, tahan kesalahan, dan mendukung konkurensi tinggi (Wangs.id., n.d.).

Dengan kemampuan Erlang dalam menangani konkurensi RabbitMQ dapat mengelola ribuan koneksi klien secara bersamaan, mendukung komunikasi antara *producer* dan *consumer* dengan efisien. Fitur toleransi kesalahan dari Erlang, seperti *supervisor trees*, memungkinkan RabbitMQ untuk terus berjalan meskipun terjadi kegagalan pada salah satu komponennya. Selain itu, *hot code swapping* dari Erlang membantu RabbitMQ melakukan pembaruan atau perbaikan tanpa mengganggu operasional sistem. Dukungan Erlang untuk arsitektur terdistribusi juga memungkinkan RabbitMQ untuk menerapkan *cluster*, sehingga meningkatkan skalabilitas dan redundansi. Dengan memanfaatkan fondasi yang kuat dari Erlang, RabbitMQ mampu memenuhi kebutuhan sistem pesan berskala besar yang andal dan fleksibel di berbagai industry (Wangs.id., n.d.).

2.1.7 *Transport Layer Security* (TLS)

Transport Layer Security (TLS) merupakan protokol keamanan yang digunakan untuk melindungi komunikasi data pada jaringan komputer. TLS dirancang untuk menjamin kerahasiaan (*confidentiality*), integritas (*integrity*), dan autentikasi (*authentication*) data selama proses transmisi. Protokol ini banyak digunakan pada berbagai layanan internet seperti HTTPS, email, VPN, maupun sistem *Internet of Things* (IoT). TLS bekerja dengan mengenkripsi data sehingga informasi yang dikirimkan tidak dapat dibaca oleh pihak yang tidak berwenang. Selain itu, TLS menggunakan sertifikat digital untuk memverifikasi identitas *Server* dan memastikan bahwa komunikasi dilakukan dengan pihak yang benar. (Vácha & Dostál, 2020).

TLS 1.3 merupakan versi terbaru dari protokol TLS yang menawarkan peningkatan keamanan dan efisiensi dibandingkan versi sebelumnya. Pada TLS 1.3, proses pembentukan koneksi (*handshake*) dilakukan dengan lebih sederhana sehingga waktu komunikasi menjadi lebih cepat. Selain itu, TLS 1.3 hanya mendukung algoritma kriptografi modern yang lebih aman dan telah menghapus beberapa algoritma lama yang dianggap memiliki kelemahan keamanan. Dalam proses komunikasi, TLS 1.3 menggunakan algoritma enkripsi seperti AES-GCM untuk melindungi data yang ditransmisikan sehingga kerahasiaan dan integritas data tetap terjaga selama proses pertukaran informasi.

Gambar 2. 4 *Transport Layer Security* 1.3 (Sumber: Indusface, 2023)

Berdasarkan Gambar 2. 4, proses komunikasi TLS 1.3 diawali dengan pengiriman pesan *Client Hello* yang berisi informasi parameter keamanan dan *key share* dari *client* kepada *server*. Selanjutnya *server* mengirimkan *Server Hello*, sertifikat digital, proses verifikasi, serta pesan *Finished* sebagai tanda bahwa

proses autentikasi dan pembentukan sesi komunikasi telah berhasil dilakukan. Setelah proses *handshake* selesai, *client* dan *Server* dapat melakukan pertukaran data aplikasi dalam bentuk HTTP *Request* dan HTTP *Response* yang telah dienkripsi menggunakan kunci sesi yang telah disepakati sebelumnya. Dengan mekanisme tersebut, TLS 1.3 mampu memberikan komunikasi yang lebih aman dan efisien karena data yang dikirimkan terlindungi dari penyadapan maupun perubahan oleh pihak yang tidak berwenang selama proses transmisi (Indusface, 2023).

2.1.8 *Flask*

Flask merupakan *framework* web berbasis bahasa pemrograman Python yang digunakan untuk membangun aplikasi web dan *Application Programming Interface* (API). *Flask* termasuk ke dalam *microframework* karena memiliki struktur yang ringan dan sederhana serta tidak memerlukan banyak dependensi tambahan dalam proses pengembangannya. *Framework* ini menyediakan berbagai fitur dasar untuk pengembangan aplikasi web seperti *routing*, *request handling*, *template engine*, dan integrasi dengan berbagai library Python.

Flask banyak digunakan dalam pengembangan sistem berbasis web karena fleksibel dan mudah diintegrasikan dengan *database*, layanan jaringan, maupun protokol komunikasi lainnya. Selain itu, *Flask* mendukung komunikasi berbasis HTTP sehingga sering dimanfaatkan dalam pengembangan sistem *Internet of Things* (IoT), *web service*, dan aplikasi *monitoring* berbasis web. Keunggulan Flask terletak pada kemudahan penggunaan, fleksibilitas pengembangan, serta dokumentasi yang lengkap sehingga memudahkan pengembang dalam membangun aplikasi berbasis Python (Projects, 2025).

2.1.9 *Web Dashboard*

Web dashboard merupakan antarmuka berbasis web yang digunakan untuk menampilkan informasi, visualisasi data, dan *monitoring* sistem secara realtime melalui *browser*. *Dashboard* berfungsi sebagai media penyajian data sehingga pengguna dapat memahami kondisi sistem dengan lebih mudah dan cepat. Informasi pada *dashboard* dapat ditampilkan dalam bentuk tabel, grafik, indikator, maupun visualisasi lainnya sesuai kebutuhan pengguna.

Dalam implementasinya, *web dashboard* banyak digunakan pada sistem *monitoring*, *Internet of Things* (IoT), dan sistem informasi karena mampu memberikan tampilan data secara realtime dan interaktif. Selain itu, *dashboard* berbasis web memiliki kelebihan berupa kemudahan akses karena dapat dibuka melalui berbagai perangkat yang memiliki browser dan koneksi jaringan. *Dashboard* juga mendukung integrasi dengan berbagai teknologi backend sehingga memudahkan proses pengolahan dan visualisasi data (Darmayanti & Samsugi, 2026).

2.1.10 OpenSSL

OpenSSL adalah pustaka perangkat lunak sumber terbuka yang mengimplementasikan protokol *Secure Sockets Layer* (SSL) dan *Transport Layer Security* (TLS). Pustaka ini menyediakan fungsi kriptografi dasar, termasuk enkripsi, dekripsi, tanda tangan digital, dan pembuatan sertifikat digital. OpenSSL mendukung berbagai algoritma kriptografi seperti AES, SHA, RSA, dan elliptic curve. Sebagai alat utama untuk komunikasi yang aman, OpenSSL sering digunakan dalam *server web*, aplikasi *clien-server*, dan sistem IoT untuk melindungi data dari ancaman seperti penyadapan dan pemalsuan identitas (DewaBiz, 2023).

2.1.11 Arduino IDE

Arduino IDE (*Integrated Development Environment*) adalah lingkungan pengembangan resmi yang dirancang khusus untuk platform Arduino. Peran Arduino IDE sebagai *framework* yang menyediakan tools pengembangan yang sederhana namun *powerful* untuk aplikasi *embedded systems*. IDE ini mengintegrasikan berbagai fitur esensial termasuk editor kode dengan syntax *highlighting*, compiler terintegrasi, serial monitor untuk debugging, serta *library* dan *board manager* untuk mengelola komponen tambahan. Arduino IDE mengimplementasikan *workflow development* yang *streamlined*, memungkinkan pengembang untuk fokus pada logika aplikasi tanpa terbebani kompleksitas *tools* pengembangan. Pendekatan ini telah membuat Arduino IDE menjadi pilihan utama bagi pengembang *embedded systems*, baik pemula maupun professional (KMTek, 2021).

2.1.12 *Wireshark*

Wireshark adalah *network protocol analyzer* yang *powerful* dan *widely-used* dalam analisis jaringan komputer. *Wireshark* juga didefinisikan sebagai tool analisis jaringan yang memungkinkan pengguna untuk menangkap dan menganalisis paket data secara *real-time*, memberikan insight mendalam tentang aktivitas jaringan. Kapabilitas utama *Wireshark* mencakup *deep packet inspection, protocol dissection, packet filtering,* dan *statistical analysis*, yang membuatnya menjadi *tool* yang tidak tergantikan dalam *network troubleshooting* dan analisis keamanan jaringan. *Wireshark* telah menjadi standar industri untuk analisis jaringan dan banyak digunakan dalam berbagai konteks, termasuk *troubleshooting* jaringan, analisis keamanan, pengembangan protokol, dan pendidikan jaringan computer (Universitas Ahmad Dahlan, 2023).

2.1.13 *Ancaman terhada Protokol AMQP*

Protokol AMQP juga memiliki beberapa ancaman keamanan yang dapat memengaruhi kerahasiaan, integritas, dan ketersediaan data apabila tidak disertai mekanisme keamanan yang memadai (Hafsa et al., 2025). Berikut beberapa ancaman yang dapat terjadi pada protokol AMQP:

1) Serangan *Man-in-the-Middle* (MitM)

   Serangan *Man-in-the-Middle* (MitM) terjadi ketika penyerang berada di antara komunikasi klien dan broker AMQP tanpa diketahui oleh kedua pihak. Pada kondisi ini, penyerang dapat menyadap, membaca, bahkan memodifikasi pesan yang dikirimkan. Ancaman ini umumnya terjadi ketika komunikasi tidak menggunakan enkripsi *Transport Layer Security* (TLS) atau ketika verifikasi sertifikat *Server* diabaikan. Oleh karena itu, penggunaan TLS sangat penting untuk menjaga keamanan komunikasi pada protokol AMQP (Aiash et al., 2025).

2) Eksploitasi Autentikasi

   AMQP umumnya menggunakan mekanisme autentikasi berbasis *username* dan *password* untuk mengamankan akses ke broker. Namun, apabila kredensial yang digunakan lemah atau proses autentikasi dilakukan tanpa enkripsi, maka informasi login dapat disadap oleh penyerang. Selain itu, konfigurasi hak akses yang tidak tepat juga dapat menyebabkan pengguna

memperoleh akses yang tidak sah terhadap *queue* maupun *exchange* pada broker AMQP (Hafsa et al., 2025).

3) *Injection Attack*

   Serangan ini terjadi ketika penyerang mengirimkan pesan atau *payload* berbahaya ke dalam broker AMQP. Pesan tersebut dapat dimanfaatkan untuk memanipulasi data, mengganggu proses sistem, atau memicu eksekusi perintah tertentu pada aplikasi penerima pesan. Risiko serangan ini meningkat apabila sistem tidak melakukan validasi data atau *filtering* terhadap pesan yang diterima dari broker (Aiash et al., 2025).

4) *Overloading* (DoS/DDoS)

   Penyerangan dapat terjadi dengan cara mengirimkan pesan dalam jumlah besar secara terus-menerus ke broker AMQP. Kondisi ini dapat menyebabkan *Server* mengalami *overload* sehingga layanan menjadi lambat atau bahkan tidak dapat diakses oleh pengguna yang sah. Ancaman ini menjadi salah satu tantangan utama pada implementasi AMQP di lingkungan IoT yang memiliki banyak perangkat terhubung (Aiash et al., 2025).

5) Eksfiltrasi Data

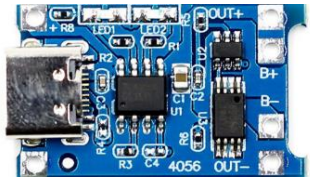
   Pesan yang dikirim melalui protokol AMQP sering kali mengandung informasi penting atau data sensitif. Jika komunikasi dilakukan tanpa enkripsi, maka data dapat disadap oleh pihak yang tidak berwenang melalui jaringan komunikasi. Selain melalui penyadapan, eksfiltrasi data juga dapat terjadi akibat kelemahan konfigurasi keamanan pada broker AMQP (Hafsa et al., 2025).

6) *Privilege Escalation*

   Privilege escalation terjadi ketika pengguna dengan hak akses terbatas berhasil memperoleh hak akses yang lebih tinggi akibat kelemahan sistem keamanan atau kesalahan konfigurasi kontrol akses. Serangan ini memungkinkan penyerang untuk mengakses queue, exchange, konfigurasi broker, maupun sumber daya lain yang seharusnya tidak dapat diakses oleh pengguna biasa. Oleh karena itu, penerapan access control yang baik sangat diperlukan pada sistem berbasis AMQP (Hafsa et al., 2025).

2.1.14 TP4056

TP4056 *Lithium Battery Charger Module* merupakan IC charger linear yang digunakan untuk mengisi daya baterai *Lithium-Ion* atau *Lithium-Polymer* (Li-Po) satu sel. Modul ini menggunakan metode pengisian CC/CV (*Constant Current/Constant Voltage*) sehingga proses pengisian daya dapat berlangsung dengan aman hingga mencapai tegangan maksimum 4,2V tanpa memerlukan rangkaian kontrol tambahan. TP4056 juga dilengkapi fitur *auto cut-off* yang dapat menghentikan *proses charging* secara otomatis ketika baterai telah penuh sehingga membantu menjaga keamanan dan umur baterai (Ratings, n.d.).



Gambar 2. 5 TP4056 (Sumber: https://share.google/B685XvVUgxV8yqElK)

Gambar 2. 5 menampilkan perangkat modul TP4056 yang dilengkapi dengan port *input Micro* USB atau Type-C serta terminal *output* baterai. Pada modul ini juga terdapat IC proteksi tambahan yang berfungsi untuk melindungi baterai dari kondisi *overcharge*, *overdischarge*, dan *short circuit*. (Ratings, n.d.).

2.1.15 MT3608

MT3608 DC-DC *Step Up Converter* merupakan modul DC-to-DC *boost converter* yang digunakan untuk menaikkan tegangan DC dari tegangan rendah menjadi tegangan yang lebih tinggi. Modul ini bekerja menggunakan metode *switching regulator* sehingga memiliki efisiensi daya yang cukup baik dibandingkan *regulator linear*. MT3608 memiliki frekuensi *switching* sebesar 1,2 MHz dan mampu menghasilkan tegangan *output* hingga 28V. Selain itu, modul ini juga dilengkapi trimpot yang digunakan untuk mengatur tegangan output sesuai kebutuhan sistem. Dengan ukuran yang kecil dan efisiensi daya yang tinggi, MT3608 banyak digunakan pada perangkat berbasis baterai dan sistem IoT (Mosfet, n.d.).

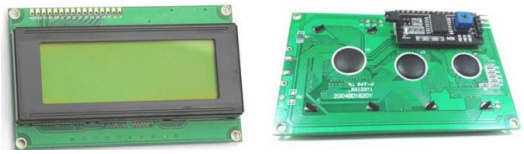
Gambar 2. 6 MT3608

(Sumber: https://share.google/YTGD1FcQr7xGrK6Jx)

Gambar 2. 6 menampilkan perangkat modul *step-up* MT3608 yang memiliki pin Input dan Output serta trimpot pengatur tegangan. Modul ini digunakan untuk menaikkan tegangan baterai Li-ion menjadi 5V agar dapat digunakan untuk menyuplai daya pada perangkat *monitoring* (Mosfet, n.d.).

2.1.16 LCD 20x4 I2C

LCD 20x4 I2C merupakan perangkat output yang digunakan untuk menampilkan data *monitoring* dalam bentuk karakter. LCD ini mampu menampilkan 20 karakter dan 4 baris tampilan sehingga cukup untuk menampilkan informasi suhu, kelembapan, maupun status sistem secara realtime. Penggunaan modul I2C pada LCD bertujuan untuk mengurangi penggunaan pin pada mikrokontroler karena komunikasi data hanya menggunakan jalur Serial Data (SDA) dan *Serial Clock* (SCL). Dengan demikian, proses integrasi LCD dengan mikrokontroler menjadi lebih sederhana dan efisien. (Guide, n.d.).



Gambar 2. 7 Layar LCD 20x4 Modul I2C (Sumber: Guide, n.d.)

Gambar 2. 7 menampilkan perangkat layar LCD 20x4 I2C yang Telah dilengkapi dengan modul I2C sehingga komunikasi data dapat dilakukan secara serial. LCD ini digunakan sebagai media tampilan untuk menampilkan data *monitoring* secara langsung pada perangkat (Guide, n.d.).

2.2 Penelitian Terdahulu

Pengembangan keamanan jaringan IoT berbasis ESP32 menggunakan protokol *Advanced Message Queuing Protocol* (AMQP) dan enkripsi *Transport Layer Security* (TLS) menunjukkan pentingnya kombinasi protokol komunikasi

yang aman, efisien, dan handal. Penelitian oleh Prasetyo dan Nurwarsito (2022) mengimplementasikan AMQP dalam sistem *monitoring* debit air depot air minum isi ulang menggunakan Raspberry Pi sebagai *producer*, *Cloud* AMQP sebagai broker, dan laptop pengguna sebagai *consumer*. Hasil pengujian menunjukkan kinerja sistem yang sangat baik, dengan *packet loss* 0%, delay rata-rata 63,06 ms, dan jitter rata-rata 0,32 ms. Hal ini mengonfirmasi keandalan AMQP untuk pengiriman data *real-time* dalam sistem *monitoring* berbasis IoT (Prasetyo & Nurwarsito, 2022).

Sementara itu penelitian oleh Gumelar et al. (2020) menunjukkan bahwa protokol AMQP memiliki keunggulan dibandingkan MQTT dalam pengiriman data besar dan kontinu, seperti data *Electrocardiogram* (ECG). Dalam pengujian, AMQP menunjukkan delay kurang dari 1 detik per pengiriman dengan throughput rata-rata sebesar 15,2 Mbps, sementara MQTT mengalami delay hingga 46 detik. Hal ini menegaskan bahwa AMQP lebih cocok untuk pengiriman data *real-time* yang berkapasitas besar (Gumelar et al., 2020). Penelitian oleh Nusantara et al. (Adiono et al., 2020) juga mendukung keunggulan AMQP dalam model *Publish/Subscribe* pada lingkungan *Smart Home*, di mana AMQP mencatat waktu pengiriman data yang stabil meskipun volume data meningkat, sedangkan MQTT mengalami peningkatan waktu pengiriman seiring bertambahnya jumlah data (Nusantara & Rizkiqa Akbar, Sabriansyah Rachmadi, 2016).

Selanjutnya, penelitian oleh Adiono et al. (2020) mengembangkan sistem cloud berbasis AMQP menggunakan RabbitMQ, dengan fitur *store-and-forward* yang mampu menangani koneksi internet tidak stabil. Penelitian ini menyoroti interoperabilitas, fleksibilitas, dan keamanan AMQP, menjadikannya solusi ideal untuk komunikasi perangkat IoT dalam berbagai kondisi jaringan (Adiono et al., 2020). Selain itu, penelitian oleh Wahyudi et al. (2024) mengimplementasikan protokol SSL/TLS pada web *Server* Apache untuk meningkatkan keamanan data dalam jaringan internal. Penelitian ini menunjukkan bahwa data yang dienkripsi menggunakan SSL/TLS terlindungi dengan baik dari akses tidak sah. Hal ini relevan untuk integrasi mekanisme enkripsi TLS dalam pengembangan sistem IoT berbasis AMQP, guna memastikan keamanan data selama transmisi (Wahyudi, 2024).

Penelitian lain oleh Baranauskas et al. (2019) mengevaluasi dampak penggunaan protokol MQTT dengan berbagai tingkat QoS yang dilengkapi TLS. Hasilnya menunjukkan bahwa TLS memberikan lapisan keamanan tambahan meskipun memengaruhi konsumsi energi, terutama pada protokol dengan tingkat QoS yang lebih tinggi. Penggunaan TLS sebagai lapisan enkripsi diidentifikasi sebagai langkah penting untuk menjaga kerahasiaan dan integritas data dalam kedua protokol tersebut. Tantangan utama dalam protokol komunikasi IoT, seperti MQTT, adalah kurangnya fitur keamanan bawaan, seperti autentikasi dan enkripsi. Oleh karena itu, pendekatan seperti penggunaan TLS atau Datagram *Transport Layer Security* (DTLS) dianggap penting untuk melindungi data, khususnya di lingkungan IoT (Baranauskas et al., 2019).

Implementasi keamanan berbasis enkripsi juga didukung oleh penelitian Mohammad Al-Mashhadani dan Mohamed Shujaa (2022), yang menggunakan algoritma AES-128 pada *platform* ESP32. Sistem mereka menunjukkan efisiensi tinggi dalam melindungi data IoT dengan menghubungkan sensor ke modul ESP32, di mana data dienkripsi sebelum dikirim ke *server* atau situs web yang aman (Al-Mashhadani & Shujaa, 2022). Selain itu, penelitian oleh Dharmawan et al. (2022) mengevaluasi kerentanan SSL/TLS pada jaringan Universitas Kristen Duta Wacana, dengan rekomendasi seperti penerapan HSTS Preload dan konfigurasi keamanan yang lebih baik untuk mengurangi risiko serangan seperti SSL Strip dan SWEET32 (Dharmawan et al., 2022). Penelitian-penelitian ini memberikan landasan kuat untuk pengembangan keamanan jaringan IoT berbasis ESP32 menggunakan protokol AMQP dan enkripsi TLS. Kombinasi ini memungkinkan terciptanya solusi keamanan IoT yang lebih komprehensif dan andal, yang dapat diadaptasi untuk berbagai kebutuhan aplikasi IoT.

Tabel 2.1 Penelitian Terdahulu

Peneliti	**Judul Penelitian**	**Metode yang diteliti**	**Hasil Penelitian**
Prasetyo & Nurwarsito	Implementasi Protokol AMQP pada Sistem *Monitoring* Debit Air Depot Air Minum Isi Ulang	Implementasi sistem *monitoring* dengan Raspberry Pi sebagai *producer*, *Cloud* AMQP sebagai broker, dan laptop pengguna sebagai *consumer*	Packet loss 0%, delay rata-rata 63,06 ms, jitter rata-rata 0,32 ms. AMQP terbukti andal untuk pengiriman data *real-time* dalam sistem IoT.

Peneliti	Judul Penelitian	Metode yang diteliti	Hasil Penelitian
Gumelar et al.	*Performance Analysis of AMQP Protocol for Patient Health Data in IoT Case*	Perbandingan *delay* dan *throughput* pada pengiriman data ECG menggunakan AMQP dan MQTT	AMQP memiliki *delay* < 1 detik dengan *throughput* 15,2 Mbps, sementara MQTT mengalami *delay* hingga 46 detik.
Nusantara et al.	Analisa Metode *Publish/Subscribe* Untuk Komunikasi Data Antar Perangkat Dalam Lingkungan *Smart Home*	Pengiriman data dalam lingkungan *Smart Home* menggunakan model *Publish/Subscribe* dengan AMQP dan MQTT.	AMQP menunjukkan waktu pengiriman data stabil meskipun volume data meningkat, sementara MQTT mengalami peningkatan waktu pengiriman.
Adiono et al.	*Cloud System Design using AMQP Protocol for Smart Devices System Applications*	Pengembangan sistem cloud dengan fitur *store-and-forward* untuk koneksi tidak stabil menggunakan RabbitMQ.	AMQP terbukti *interoperabel*, *fleksibel*, dan aman untuk komunikasi perangkat IoT.
Wahyudi	Mengimplementasikan SSL/TLS pada Web *Server* Apache di dalam Jaringan Internal Praktikum untuk Pengembangan Web *Server*	Penerapan SSL/TLS pada web *Server*.	SSL/TLS meningkatkan keamanan data dengan melindungi transmisi dari akses tidak sah.
Baranauskas et al.	*Evaluation of the impact on energy consumption of MQTT protocol over TLS*	Evaluasi dampak TLS pada protokol MQTT dengan berbagai tingkat QoS	TLS meningkatkan keamanan data tetapi memengaruhi konsumsi energi. Penting untuk menjaga kerahasiaan dan integritas data.
Al-Mashhadani & Shujaa	*IoT Security Using AES Encryption Technology based ESP32 Platform*	Implementasi algoritma AES-128	AES-128 pada ESP32 efektif melindungi data IoT.
Dharmawan et al.	Analisis Keamanan Jaringan Universitas Kristen Duta Wacana Dengan Serangan Ssl/Tls	Kerentanan jaringan	Mengurangi risiko serangan seperti SSL Strip dan SWEET32 dengan konfigurasi keamanan lebih baik.