BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Sebelum melakukan penelitian, peneliti telah menelusuri beberapa penelitian mengenai protokol komunikasi MQTT dan HTTP pada sistem *Internet of Things*. Berikut beberapa penelitian yang terkait dengan tema yang akan diteliti :

Penelitian yang dilakukan oleh Nitin Naik pada tahun 2017 yang berjudul "*Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP*" [5] menjelaskan bahwa memilih protokol komunikasi yang standar dan efisien adalah hal yang penting, karena bergantung pada jenis sistem IoT dan persyaratan komunikasinya. Penelitian ini memberikan perbandingan kemampuan dari masing-masing protokol komunikasi. Kemudian, dilakukan analisis untuk mengetahui kelebihan dan kekurangan masing-masing protokol komunikasi. Hasilnya menunjukkan bahwa HTTP memiliki skor tertinggi dalam hal ukuran dan *overhead* pesan, sumber daya, *bandwidth*, dan *delay*, sedangkan CoAP memiliki skor terendah. AMQP menawarkan dukungan tertinggi untuk keamanan informasi.

Penelitian yang dilakukan oleh F Luthfi, E A Juanda, dan I Kustiawan pada tahun 2017 yang berjudul "*Optimization of Data Communication on Air Control Device Based on Internet of Things with Application of HTTP and MQTT Protocols*" [1] menjelaskan bahwa protokol HTTP memiliki *overhead* tinggi, yang menyebabkan konsumsi daya terbuang percuma. Oleh karena itu, perlu adanya pengoptimalan daya. Penelitian ini melakukan pengoptimalan komunikasi data pada perangkat untuk mencapai penghematan energi yang efektif. Hasilnya menunjukkan bahwa protokol MQTT memiliki *delay* rendah, *overhead* rendah, *throughput* tinggi, dan efisiensi transmisi rendah

dibandingkan dengan HTTP. Masa pakai baterai bisa lebih lama saat menggunakan protokol MQTT.

Penelitian yang dilakukan oleh Claudia Felicia Permatasari, Harry Dhika pada tahun 2018 yang berjudul "Optimasi Jalur *Transfer Data* dari HTTP menjadi MQTT pada *IoT* menggunakan *Cloud Services*" [9] menjelaskan bahwa protokol HTTP memiliki beberapa keterbatasan kinerja dan sumber daya. Selain itu, sering terjadi kehilangan data akibat koneksi internet yang tidak stabil. Penelitian ini membandingkan protokol HTTP dan MQTT dalam hal model protokol, kecepatan transfer data, ukuran pesan yang akan dikirim dan penggunaan optimal masing-masing protokol. Penelitian ini juga menggunakan RabbitMQ yang berfungsi sebagai tempat penyimpanan data sementara saat koneksi internet tidak stabil atau terputus sementara. Hasil menunjukkan bahwa protokol MQTT dapat memproses data lebih cepat (dalam 0,7 detik) dibandingkan protokol HTTP (dalam 7,8 detik).

Penelitian yang dilakukan oleh Anang Dwi Prakoso, Fikra Titan Syifa, dan Danny Kurnianto pada tahun 2020 yang berjudul "Analisis Perbandingan Kualitas Layanan Sistem Antara Protokol HTTP dan MQTT Pada Monitoring Kelembaban Tanah" [2] menjelaskan bahwa kualitas layanan dipengaruhi dengan adanya paket yang hilang atau kurang saat informasi dikirimkan. Penelitian melakukan kualitas layanan terhadap perangkat monitoring kelembaban tanah berbasis protokol HTTP dan MQTT. Hasil penelitian ini menunjukkan bahwa rata-rata nilai *delay* untuk protokol HTTP dan MQTT masing-masing kurang dari 150 dan 63 ms. HTTP memiliki nilai *jitter* rata-rata 0 hingga 62 ms sedangkan MQTT memiliki rentang *jitter* 0 hingga 16 *ms*. *Throughput* HTTP adalah 88 *ms* dan protokol MQTT adalah 68 *ms*. Kemudian parameter paket memiliki kualitas yang sama ketika memberikan nilai *packet loss* 0%.

Penelitian yang dilakukan oleh Ata Amrullah, M. Udin Harun Al Rasyid, dan Idris Winarno pada tahun 2022 yang berjudul "Implementasi dan Analisis Protokol Komunikasi IoT untuk *Crowdsensing* pada Bidang Kesehatan" [6] menjelaskan bahwa dalam bidang kesehatan, *crowdsensing*

dapat membantu mendapatkan banyak data penting mengenai kondisi kesehatan masyarakat. Namun, teknik *crowdsensing* yang ada hanya menggunakan satu protokol komunikasi, sehingga dapat menimbulkan masalah jika perangkat IoT menggunakan berbagai protokol komunikasi yang berbeda. Penelitian ini bertujuan untuk menggabungkan tiga jenis protokol komunikasi yang berjalan di *gateway*, yaitu MQTT, HTTP, dan CoAP. *Gateway* ini berfungsi mengubah ketiga protokol menjadi protokol yang sama dengan *server backend cloud*. Hasilnya menunjukkan bahwa Protokol MQTT memiliki tingkat *delay* terbaik, sedangkan protokol CoAP memiliki tingkat *throughput* terbaik. Selama pengujian, paket yang hilang dari ketiga protokol tersebut kurang dari 4%.

Penelitian yang dilakukan oleh Ahmad Rifa'i, M. Udin Harun Al Rasyid, dan Agus Indra Gunawan pada tahun 2021 yang berjudul "Sistem Pemantauan dan Kontrol Otomatis Kualitas Air Berbasis IoT Menggunakan Platform Node-Red untuk Budidaya Udang" [12] menjelaskan bahwa kualitas air yang baik dapat menyebabkan tingkat keberhasilan budidaya udang. Oleh karena itu, diperlukan suatu sistem yang dapat memantau kualitas air, mengontrol aktuator yang mengolah air di kolam budidaya, dan memberikan pesan peringatan dini secara *real-time*. Pada penelitian ini dilakukan tindakan pencegahan dengan menggunakan komunikasi protokol MQTT. Hasil menunjukkan bahwa keterlambatan pengiriman informasi dari *publisher* ke *subscriber* saat menggunakan *broker* HIVEMQ publik rata-rata 260 *ms*.

Penelitian yang dilakukan oleh Heriberto Javier Jara Ochoa pada tahun 2020 yang berjudul "*Comparative analysis of power consumption between MQTT and HTTP protocols for an IoT platform designed and implemented for remote real-time monitoring of long-term cold chain transport operations*" [13] menjelaskan bahwa solusi teknologi IoT telah ditemukan dan dikembangkan untuk industri transportasi seperti untuk memantau nilai suhu, kelembaban muatan, dan lokasi kendaraan. Sehingga pengembangan dan produksi perangkat IoT dalam sistem portabel meningkatkan pentingnya

konsumsi daya. Penelitian ini bertujuan untuk melakukan eksperimen terhadap protokol komunikasi HTTP dan MQTT untuk membuat perbandingan dan menunjukkan perbedaan konsumsi daya. Hasil penelitian menunjukan bahwa protokol MQTT dengan QoS 0 dan 1 mampu menghasilkan penghematan daya masing-masing 6,03% dan 8,33% dibandingkan dengan protokol HTTP yang menghasilkan tingkat daya yang lebih besar daripada protokol MQTT.

Penelitian yang dilakukan oleh Hairatunnisa, Hapsoro Agung Nugroho, dan Relly Margiono pada tahun 2021 yang berjudul "Analisis Kinerja Protokol MQTT dan HTTP pada Akuisisi Data Magnet berbasis *Internet of Things*" [14] menjelaskan bahwa Salah satu faktor yang mempengaruhi kualitas data magnetik yaitu data yang kontinyu, sehingga diperlukan suatu sistem transmisi data yang dapat mengirimkan data pengamatan secara kontinyu. Pada penelitian ini dirancang sistem komunikasi magnetometer dengan konsep *Internet of Things* menggunakan protokol MQTT dan HTTP, data pengukuran ditampilkan dalam bentuk sumbu x, sumbu y, sumbu z, komponen horizontal dan komponen medan magnet total yang ditampilkan pada *dashboard* secara *real time* dan terus menerus. Hasil penelitian menunjukan bahwa protokol MQTT memiliki kualitas pengiriman yang lebih baik dibandingkan dengan protokol HTTP dengan nilai *delay* 0,0120 detik, *packet length* 54 byte dan *packet loss* 0,11%, sedangkan protokol HTTP memiliki nilai *delay* 0,0257 detik, *packet length* 268,1 *byte* dan *packet loss* sebesar 0,5%.

Beberapa penelitian yang menggunakan beberapa protokol komunikasi untuk sistem *Internet of Things* dapat dilihat pada Tabel 2.1

Tabel 2.1 *Penelitian Terkait*

No	Judul	Penulis (Tahun)	Subjek	Protokol Komunikasi	Hasil	Perbedaan Penelitian
1.	*Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP*	N. Naik (2017) [5]	Sistem Internet of Things	MQTT, CoAP, AMQP, dan HTTP	HTTP memiliki nilai tertinggi dari segi ukuran dan *overhead* pesan, sumber daya, *bandwidth* dan *delay*, sedangkan untuk nilai terendah dimiliki oleh CoAP. MQTT menawarkan tingkat kualitas layanan tertinggi dengan interoperabilitas paling sedikit. AMQP	Penelitian hanya menyajikan perbandingan untuk memperkenalkan karakteristik masing-masing protokol komunikas secara komparatif. Kemudian melakukan analisis untuk mendapatkan kekurangan dan kelebihan masing-

No	Judul	Penulis (Tahun)	Subjek	Protokol Komunikasi	Hasil	Perbedaan Penelitian
					memiliki tingkat dukungan tertinggi untuk keamanan dan layanan tambahan.	masing protokol komunikasi
2.	Optimization of Data Communication on Air Control Device Based on Internet of Things with Application of HTTP and MQTT Protocols	F. Luthfi, E. A. Juanda, dan I. Kustiawan (2017) [1]	Air Control Device	HTTP, MQTT	Protokol MQTT memiliki *latency* rendah, *overhead* rendah, *throughput* tinggi, dan daya transmisi rendah dibandingkan dengan HTTP. Dengan menggunakan Protokol MQTT, penggunaan baterai dapat menjadi lebih lama.	Pengujian dilakukan pada alat pengatur suhu ruangan pada AC. Tujuan penelitian untuk mengoptimalkan komunikasi data menjadi lebih efisien dan hemat daya.

No	Judul	Penulis (Tahun)	Subjek	Protokol Komunikasi	Hasil	Perbedaan Penelitian
3.	Optimasi Jalur Transfer Data dari HTTP menjadi MQTT pada IoT menggunakan Cloud Services	C. F. Permatasari dan H. Dhika (2018) [9]	RabbitMQ (*broker-message*) pada *cloud-server*	HTTP, MQTT	Protokol MQTT dapat memproses data lebih cepat (berada di angka 0.7 detik) dibanding protokol HTTP (berada di angka 7.8 detik).	Penelitian menggunakan RabbitMQ pada *cloud server* sebagai *message broker* untuk meminimalisir terhadap permasalahan kehilangan data akibat koneksi *internet* yang tidak stabil.
4.	Analisis Perbandingan Kualitas Layanan Sistem Antara Protokol HTTP dan MQTT Pada	A. D. Prakoso, F. dan T. Syifa (2020) [2]	Kelembaban Tanah	HTTP, MQTT	Protokol HTTP dan MQTT memiliki rata-rata nilai *delay* dibawah 150 atau 63 *ms*. HTTP memiliki nilai *jitter* rata-rata antara 0 - 62 *ms*,	Subjek penelitian yaitu Kelembaban Tanah disekitar tanaman. Penelitian bertujuan untuk mengetahui perbandingan kualitas layanan pada protokol

No	Judul	Penulis (Tahun)	Subjek	Protokol Komunikasi	Hasil	Perbedaan Penelitian
	Monitoring Kelembaban Tanah				sedangkan MQTT memiliki nilai rentang *jitter* antara 0 - 16 *ms*. Sedangkan nilai *throughput* HTTP adalah 88 *ms* dan protokol MQTT adalah 68 *ms*. Kemudian untuk parameter *packet loss* keduanya memiliki kualitas yang sama karena menghasilkan nilai *packet loss* sebesar 0%.	HTTP dan MQTT terhadap perangkat sensor kelembaban tanah.
5.	Implementasi dan Analisis Protokol Komunikasi *IoT*	A. Amrullah, dan M. U. H.	*Crowdsensing*	MQTT, HTTP, CoAP	Protokol MQTT memiliki tingkat *delay* terbaik, sedangkan	Penelitian menggunakan arsitektur *gateway*

No	Judul	Penulis (Tahun)	Subjek	Protokol Komunikasi	Hasil	Perbedaan Penelitian
	untuk *Crowdsensing* pada Bidang Kesehatan	Al Rasyid (2020) [6]			protokol CoAP memiliki tingkat *throughput* terbaik. Selama pengujian, ketiga protokol tersebut memiliki *packet loss* kurang dari 4%	multi-protokol komunikasi sebagai penerima data sensor dengan beragam protokol komunikasi. Subjek penelitian yaitu *Crowdsensing* pada sektor kesehatan
6.	Sistem Pemantauan dan Kontrol Otomatis Kualitas Air Berbasis IoT Menggunakan Platform Node-Red untuk Budidaya Udang	A. Rifa'i, dan M. U. H. Al Rasyid (2021) [12]	Kualitas Air	MQTT	*Delay* yang terjadi pada pengiriman data dari *publisher* ke *subscriber* diperoleh rata-rata 260 *ms* dengan menggunakan publik *Broker* HIVEMQ. Sedangkan	Penelitian menggunakan *broker* HIVEMQ. Subjek penelitian yaitu pada sensor untuk memantau kondisi kualitas air. *Platfrom* yang digunakan pada

No	Judul	Penulis (Tahun)	Subjek	Protokol Komunikasi	Hasil	Perbedaan Penelitian
					pada pengujian kontrol otomatis,	penelitian yaitu Node-Red.
7.	*Comparative analysis of power consumption between MQTT and HTTP protocols for an IoT platform designed and implemented for remote real-time monitoring of long-term cold chain transport operations*	H. J. J. Ochoa (2020) [13]	Transportasi Kargo Darat	HTTP, MQTT	Protokol MQTT dengan QoS 0 dan 1 mampu menghasilkan penghematan daya masing-masing 6,03% dan 8,33% dibandingkan dengan protokol HTTP yang menghasilkan tingkat daya yang lebih besar daripada protokol MQTT.	Penelitian ini melakukan analisis konsumsi daya antara protokol komunikasi MQTT dan HTTP untuk Transportasi Kargo Darat sebagai subjek penelitian.
8.	Analisis Kinerja Protokol *MQTT*	H. Hairatunnisa,	*Magnetometer*	MQTT, HTTP	Protokol MQTT memiliki kualitas	Penelitian ini melakukan analisis

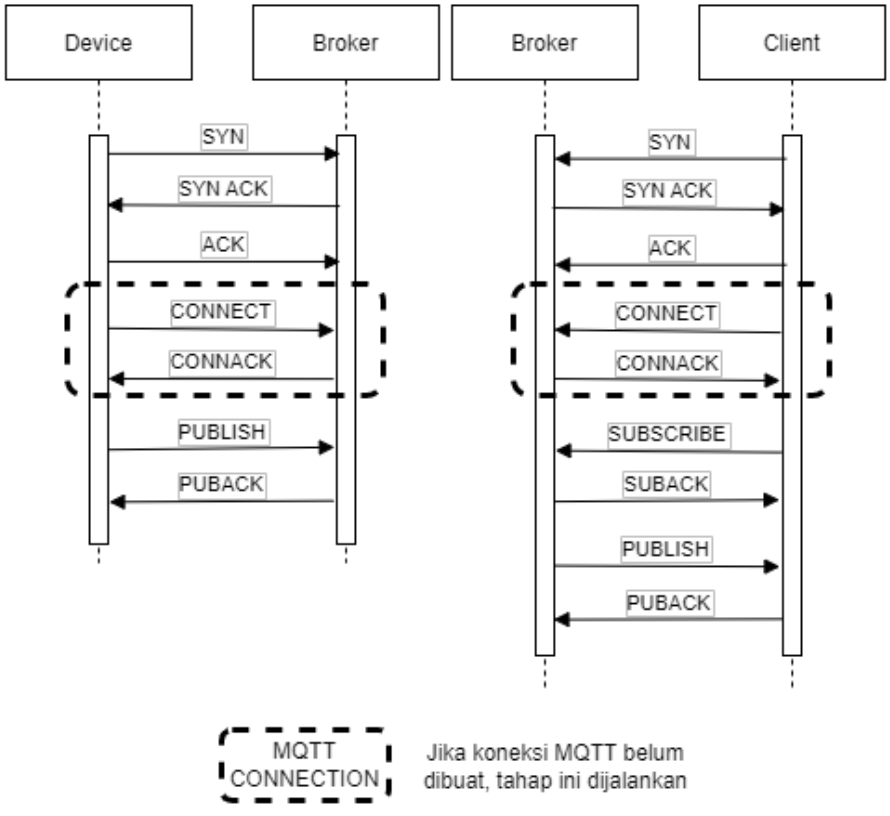
No	Judul	Penulis (Tahun)	Subjek	Protokol Komunikasi	Hasil	Perbedaan Penelitian
	DAN *HTTP* pada Akuisisi Data Magnet berbasis *Internet of Things*	dan H. A. Nugroho (2021) [14]			pengiriman yang lebih baik dibandingkan dengan protokol HTPP dengan nilai *delay* 0,0120 detik, *packet length* 54 *byte* dan *packet loss* 0,11%, sedangkan protokol HTTP memiliki nilai *delay* 0,0257 detik, *packet length* 268,1 *byte* dan *packet loss* sebesar 0,5%.	kinerja protokol komunikasi MQTT dan HTTP untuk Akuisisi Data Magnet sebagai subjek penelitian.

Terdapat beberapa perbedaan antara penelitian tersebut dengan yang peneliti tulis, diantaranya protokol komunikasi yang digunakan pada penelitian ini yaitu HTTP dan MQTT, sedangkan dalam penelitian terdahulu berupa perbandingan protokol komunikasi yang berbeda. kemudian penelitian terdahulu menggunakan subjek dan *platform* yang berbeda, sedangkan peneliti menggunakan tanaman cabai merah untuk sistem monitoring sebagai media implementasi dan *website* Node.js yang di *deploy* pada google cloud platform. Selanjutnya penelitian terdahulu hanya melakukan analisis performa dari protokol komunikasi yang digunakan berdasarkan parameter yang berbeda, sedangkan peneliti dalam hal ini melakukan perbandingan dari protokol komunikasi MQTT dan HTTP, sehingga dapat mengetahui protokol mana yang terbaik pada *internet of things* berbasis *website* Node.js untuk sistem monitoring tanaman cabai merah berdasarkan parameter *throughput*, *jitter*, *packet loss*, dan *delay*.

2.2 Landasan Teori

2.2.1 MQTT

Message Queuing Telemetry Transport adalah protokol komunikasi berbasis pesan *Publish/Subscribe* yang dirancang untuk komunikasi mesin-ke-mesin yang ringan. MQTT awalnya dikembangkan oleh International Business Machines Corporation (IBM). Protokol MQTT berorientasi pada pesan dan setiap pesan dipublikasikan pada suatu topik. Arsitektur MQTT memiliki model *Client/Server* dimana setiap sensor adalah *client* dan terhubung ke *server* [15].



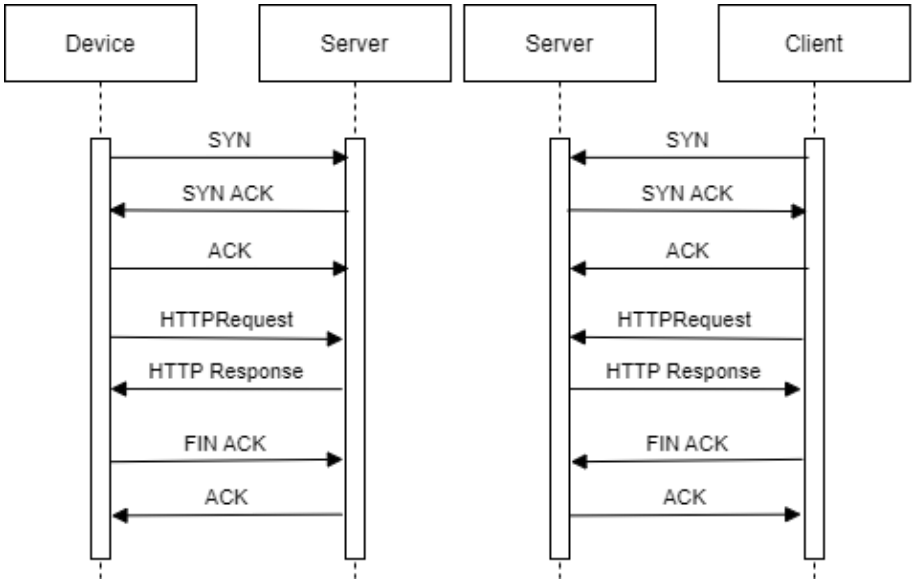
Gambar 2.1 *Workflow* MQTT

Pada gambar 2.1 merupakan alur kerja dari protokol kommunikasi MQTT. *Server* dikenal sebagai *broker* yang menggunakan komunikasi TCP. Setiap *client* yang *subscribe* suatu topik akan menerima setiap pesan yang di *publish* di topik tersebut. MQTT menyediakan tiga mode transfer berdasarkan

keandalan yang diperlukan: QoS0 (Transmisi tidak terjamin), QoS1 (transmisi Terjamin), dan QoS2 (Layanan terjamin pada aplikasi) [16].

2.2.2 HTTP

Hypertext Transfer Protocol adalah protokol *request/response* (permintaan dan tanggapan) antara klien dan *server* dimana klien meminta informasi dari *server* (*request*) dan *server* menanggapi permintaan (*response*) yang sesuai [7].



Gambar 2.2 *Workflow* HTTP

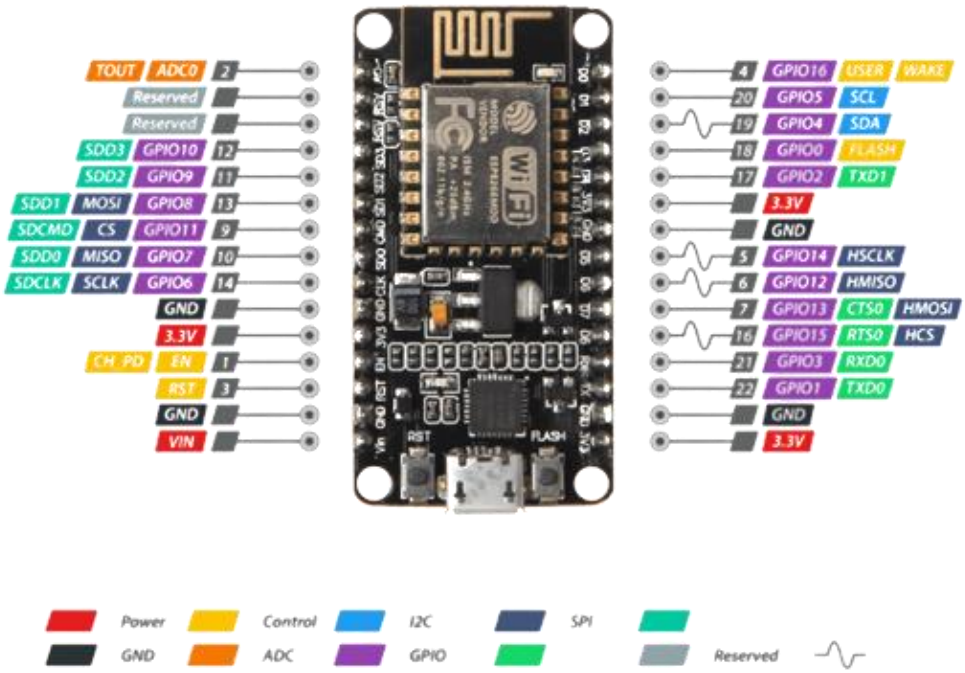
Pada gambar 2.2 merupakan alur kerja dari protokol komunikasi HTTP. Data HTTP berada di atas protokol TCP, yang menjamin keandalan pengiriman, memecah permintaan dan respons data besar menjadi potongan-potongan yang dapat dikelola jaringan. Namun, koneksi tersebut dibuat pada setiap akses, karena data yang diakses ditransfer berdasarkan alamat IP dan URL yang hubungannya berubah secara dinamis. Singkatnya, setelah berkali-kali melepaskan koneksi, komunikasi selesai. Oleh karena itu, Protokol HTTP biasanya digunakan untuk sistem *internet of things* yang memiliki banyak data untuk dipublikasikan [5], [16].

2.2.3 Cabai Merah

Cabai Merah banyak digunakan sebagai bumbu ajaib yang digunakan untuk persiapan masakan dan juga dikenal sebagai bahan dalam makanan untuk membuatnya pedas. Cabai Merah merupakan tanaman tropis dan subtropis yang membutuhkan kombinasi cuaca hangat, lembab namun kering. Selama tahap pertumbuhan membutuhkan cuaca yang hangat dan lembab. Namun, cuaca kering cocok untuk kematangan buah. Suhu 20º-25ºC dan kelembaban tanah 60%-80% merupakan kisaran yang ideal untuk pertumbuhan cabai. Pada suhu 37ºC atau lebih, perkembangan tanaman akan terpengaruh. Kurangnya cahaya akan mengakibatkan tanaman cabai merah menjadi lemah, pucat, dan pertumbuhannya cenderung memanjang. Demikian pula dalam kasus hujan lebat tanaman mulai membusuk. Dalam kasus kondisi kelembaban rendah selama periode waktu berbuah, tunas tidak berkembang dengan baik. Oleh karena itu, bunga dan buah dapat jatuh. Artinya, suhu yang tinggi dan tingkat kelembapan yang relatif rendah akan menyebabkan deflowering dan buah yang jika dikembangkan akan sangat kecil [17], [18].

2.2.4 Nodemcu

Nodemcu (Node Micro Controller Unit) adalah platform *open source* untuk menerapkan *Internet of Things*. Elemen penting komputer seperti CPU, jaringan (Wi-Fi), RAM, sistem operasi, dan SDK semuanya terdapat di Nodemcu, sehingga menjadikannya sangat berguna untuk semua jenis proyek IoT [19]. Nodemcu mirip dengan *board* arduino, tetapi kelebihannya sudah memiliki WIFI ESP8266, sehingga sangat cocok buat *project IoT* [20]. Nodemcu memiliki beberapa pin yang bisa digunakan. Sambungan dari masing-masing pin tersebut dapat dilihat pada Gambar 2.3



Gambar 2.3 Nodemcu

2.2.5 DHT22

DHT22 adalah sensor untuk mengukur suhu dan kelembaban yang memiliki keluaran tegangan analog dan dapat diproses lebih lanjut oleh mikrokontroler. Modul DHT22 merupakan modul pengukur suhu dan kelembaban yang memiliki keunggulan dalam hal kualitas pembacaan data yang lebih responsif dan [21].

Gambar 2.4 DHT22 *Pinout*

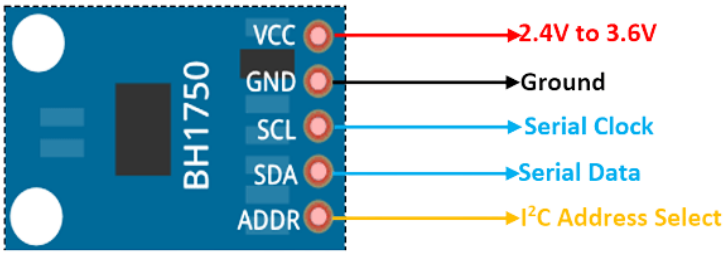
DHT22 memiliki 4 pin diantarnya yaitu VCC, Data, Tidak Ada Koneksi, dan pin *Ground*. Kemudian datang dengan versi terpasang *PCB* yang memiliki tiga pin yaitu VCC, Data, dan pin *Ground*. Papan *breakout* ini berisi resistor pull-up, yang membuatnya lebih mudah terhubung dengan Arduino. Spesifikasi sensor DHT22 dapat dilihat pada tabel 2.2

Tabel 2.2 Spesifikasi sensor DHT22

Pengukuran Suhu / Akurasi	- 40 ˚C – 80 ˚C
Pengukuran Kelembaban / Akurasi	0 % – 100 %
Keluaran Data	Digital
Ukuran Papan	15.1 mm x 25mm x 7.7mm
Input Tegangan	3 – 5 V

2.2.6 BH1750

BH1750 merupakan sensor Intensitas Cahaya yang memiliki jangkauan luas. Sensor ini dapat mengukur nilai *lux* tepatnya hingga 65535lx. *BH1750* umumnya digunakan pada *smartphone* yang mendeteksi pencahayaan lingkungan dan menyesuaikan kecerahan tampilan layar.



Gambar 2.5 *BH1750 Pinout*

Sensor memiliki lima pin, dan menggunakan protokol komunikasi I2C (pin SCL dan SDA), sehingga mudah digunakan dengan NodeMCU. Sensor cahaya BH1750 mengukur intensitas berdasarkan jumlah cahaya yang mengenainya dan langsung memberikan nilai *lux*. Kesalahan sensor ini rendah karena variasi pengukurannya hanya 20% [22]. Spesifikasi sensor BH1750 dapat dilihat pada tabel 2.3

Tabel 2.3 Spesifikasi BH1750

Pengukuran Intensitas Cahaya	1 – 65535 lux
Keluaran Data	Digital
Ukuran Papan	13.9mm x18.5mm
Input Tegangan	3.3 V

2.2.7 Capacitive Soil Moisture

Capacitive Soil Moisture adalah sensor untuk melakukan pembacaan kelembaban tanah yang dapat mengukur kandungan *volumetric* air di dalam tanah dan memberi tingkat kelembapan sebagai keluaran. Sensor ini terbuat dari bahan tahan korosi yang memberikan masa pakai yang sangat baik.

Gambar 2.6 *Sensor Capacitive Soil Moisture*

Sensor analog ini memiliki 3 *Pinout* diantaranya yaitu pin untuk dihubungkan ke GND, VCC, dan pin sinyal analog yang mengeluarkan tegangan dalam proporsi langsung dengan tingkat kelembaban tanah [23]. Spesifikasi sensor BH1750 dapat dilihat pada tabel 2.4

Tabel 2.4 Spesifikasi Capacitive Soil Moisture

Keluaran Data	Analog
Ukuran Papan	98mm x 23mm
Input Tegangan	3.3 – 5.5 V

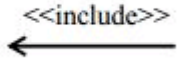
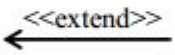
2.2.8 Blok Diagram Sistem

Diagram Blok adalah sebuah diagram berbentuk kotak (blok) yang digunakan untuk menjelaskan suatu proses kerja pada ilmu rekayasa atau engineering. Diagram blok ini paling sering digunakan untuk menjelaskan proses kerja dari suatu alat yang dibuat pada ilmu engineering atau teknik. Setiap bagian blok sistem memiliki fungsi masing-masing, dengan memahami gambar blok diagram maka sistem yang dirancang sudah dapat dibangun dengan baik [24].

2.2.9 *Use Case* Diagram

Use Case Diagram merupakan diagram yang mengammbarkan hubungan antara pengguna dengan sistem secara keseluruhan. *Use Case* mendeskripsikan interaksi tipikal antara para pengguna sistem dengan sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem tersebut digunakan. *Use Case* Diagram memiliki beberapa symbol yang dapat digunakan untuk menghubungan antara aktor dan *use case* [25]. Berikut merupakan symbol-simbol yang dapat digunakan untuk membuat *Use Case* Diagram.

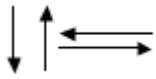
Tabel 2.5 Simbol *Use Case* Diagram

No	Simbol	Nama	Keterangan
1		Aktor	Mewakili peran orang, sistem yang lain, atau alat Ketika berkomunikasi dengan *use case*
2		*Use Case*	Abstraksi dan interaksi antara sistem dan *actor*
3		*Assocation*	Abstraksi dari penghubung antara *actor* dengan *use case*
4		*Generalisasi*	Menunjukkan spesialisasi *actor* untuk dapat berpartisipasi dengan *use case*
5		*Include*	Menunjukkan bahwa suatu *use case* seluruhnya merupakan fungsionalitas dari *use case* lainnya
6		*Extend*	Menunjukkan bahwa suatu *use case* merupakan tambahan fungsionalitas dari *use case* lainnya jika suatu kondisi terpenuhi

2.2.10 *Activity* Diagram

Activity diagram adalah teknik untuk menggambarkan logika prosedural, proses bisnis, dan jalur kerja. Activity diagram digunakan untuk menggambarkan rangkaian aliran aktivitas baik proses bisnis maupun use case. Activity diagram dapat juga digunakan untuk memodelkan action yang akan dilakukan saat sebuah operasi dieksekusi, dan memodelkan hasil dari action tersebut [25]. Berikut merupakan symbol-simbol yang dapat digunakan untuk membuat *Activity* Diagram.

Tabel 2.6 Simbol *Activity* Diagram

No	Simbol	Nama	Keterangan
1		Status awal	Sebuah diagram aktivitas memiliki sebuah status awal
2		Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
3		Percabangan	Percabangan dimana ada pilihan aktivitas yang lebih dari satu
4		*Line Conenector*	Digunakan untuk menghubungkan satu simbol dengan simbol lainnya
5		Status akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki status akhir

2.2.11 Node.js

Node.js merupakan *runtime environment* yang memungkinkan untuk mengembangkan aplikasi *web* menggunakan bahasa pemrograman *javascript* sisi *server* (*server-side*). Salah satu *framework* dari NodeJS yaitu Express js yang ditulis dengan bahasa pemrograman JavaScript. Express JS memiliki

fitur-fitur yang lengkap untuk membuat web application seperti *routing*, *rendering view*, *middleware*, dan lain sebagainya [26].

2.2.12 Google Cloud Platform (GCP)

GCP adalah layanan *cloud* yang menawarkan ruang *server* pada *virtual machines*, Koneksi VPN, *disk storage*, dan lain sebagainya. Sedangkan *Cloud Run* merupakan layanan komputasi terkelola pada *GCP* yang memungkinkan menjalankan *container*.

2.2.13 Wireshark

Wireshark merupakan *tools* untuk menganalisis lalu lintas jaringan dan paket. *Tools* ini juga sering disebut sebagai penganalisa jaringan, penganalisa protokol jaringan atau *sniffer*. Wireshark dapat menangkap paket secara *real-time* dan menampilkannya dalam format yang dapat dibaca manusia. Wireshark juga memiliki fitur *filter ip source*, *destination*, *protocol*, dan lainnya yang dapat membantu menganalisis lebih dalam lalu lintas jaringan [27], [28].

2.2.14 Quality of service (QoS)

QoS merupakan penggunaan mekanisme teknologi yang bekerja pada jaringan untuk mengontrol lalu lintas dan memastikan kinerja aplikasi dengan kapasitas jaringan terbatas. Maksud dan tujuan utama QoS adalah untuk memberikan jaminan bahwa jaringan akan menampilkan hasil yang diharapkan [29]. Terdapat beberapa parameter untuk pengukuran QoS, diantaranya yaitu:

a. *Throughput*

Throughput adalah jumlah data yang berhasil diterima pada waktu tertentu. Satuan yang digunakan sama dengan *bandwidth* yaitu *bit per second (bps). Throughput* didapatkan dari jumlah paket yang berhasil mencapai tujuan dan ukuran paket terhadap total waktu simulasi dalam detik [30].

Tabel 2.7 *QoS Throughput* TIPHON

Kategori *Throughput*	***Throughput* (*bps*)**	***Indeks***
Sangat Bagus	100	4
Bagus	75	3
Sedang	50	2
Jelek	<25	1

(Sumber: TIPHON)

Rumus untuk menghitung *Throughput* diberikan pada Persamaan

$$Throughput = \frac{Paket\ yang\ diterima}{Lama\ Pengamatan} \quad (2.1)$$

b. *Jitter*

Jitter adalah variasi dalam latensi atau waktu tunda saat sinyal ditransmisikan dan diterima. Perbedaan ini diukur dalam milidetik (ms), semakin tinggi skor *jitter*, semakin tidak konsisten waktu responsnya. *Jitter* sering disebabkan oleh kemacetan jaringan dan perubahan rute [28].

Tabel 2.8 *QoS Jitter* TIPHON

Kategori *Jitter*	***Jitter* (ms)**	***Indeks***
Sangat Bagus	0	4
Bagus	0-75	3
Sedang	76-125	2
Jelek	125-255	1

(Sumber: TIPHON)

Rumus untuk menghitung *Jitter* diberikan pada Persamaan

$$Jitter = \frac{Total\ Variasi\ Delay}{Total\ Paket\ yang\ diterima - 1} \quad (2.2)$$

c. *Packet Loss*

Packet loss merupakan total atau jumlah data yang hilang, biasanya ditampilkan sebagai persentase dari paket yang hilang yang dikirim. *Packet loss* dapat digambarkan sebagai paket data yang hilang tidak mencapai tujuannya setelah dikirim melalui jaringan. Kehilangan paket terjadi ketika kemacetan jaringan, masalah perangkat keras, *bug* perangkat lunak, dan sejumlah faktor lain menyebabkan paket turun selama *transmisi* data [31].

Tabel 2.9 *QoS Packet Loss* TIPHON

Kategori *Packet Loss*	***Packet Loss* (%)**	***Indeks***
Sangat Bagus	0%	4
Bagus	3%	3
Sedang	15%	2
Jelek	25%	1

(Sumber: TIPHON)

Rumus untuk menghitung *Packet Loss* diberikan pada Persamaan

$$Packet\ Loss = \frac{(Paket\ dikirim - Paket\ diterima) \times 100\%}{Paket\ yang\ dikirim} \quad (2.3)$$

d. *Delay*

Delay adalah waktu total dari sebuah paket yang dihitung pada waktu mulai mengirim paket sampai berhasil terkirimnya paket di tujuan. *Delay* dihitung menggunakan perbedaan waktu antara paket yang dikirim dari sumber dan diterima di tujuan [29]. Sehingga nilai

delay rata-rata didapatkan dengan membagi total *delay packet* dan total *packet* diterima.

Tabel 2.10 *QoS Delay* TIPHON

Kategori *Delay*	***Delay (ms)***	***Indeks***
Sangat Bagus	<150	4
Bagus	150-300	3
Sedang	300-450	2
Jelek	>450	1

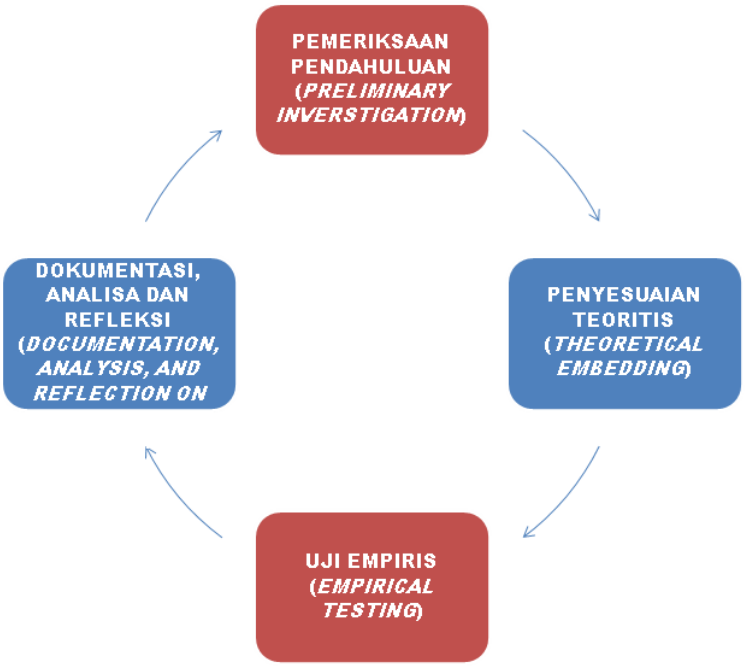
(Sumber: TIPHON)

Adapun rumus untuk menghitung *delay* diberikan dalam Persamaan:

$$Delay = \frac{Total\ Delay}{Total\ Paket\ yang\ diterima} \quad (2.4)$$

2.2.15 Metode R&D

Metode R&D merupakan metode penelitian yang digunakan untuk membuat produk tertentu dan menguji keefektifan produk tersebut [32]. Untuk mengetahui keefektifan produk yang dihasilkan maka dilakukan eksperimen untuk mengujinya. Eksperimen adalah prosedur (langkah-langkah) yang dilakukan sebelum melakukan eksperimen untuk memperoleh informasi yang diperlukan untuk analisis dan kesimpulan objektif [33]. Metode eksperimen mencakup beberapa tahapan, diantaranya yaitu melakukan perancangan, membuat dan melakukan uji coba pada sistem.



Gambar 2.7 Tahapan Metode R&D

2.2.16 Black Box Testing

Pengujian *Black Box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak. Metode ini digunakan untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pengujian *black box* merupakan metode perancangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dieksekusi pada perangkat lunak dan kemudian keluar dari perangkat lunak dicek apakah telah sesuai yang diharapkan [34].

Teknik pengujian *Black Box* berguna untuk menemukan kerentanan tertentu seperti masalah validasi *input/output*, kesalahan terkait konfigurasi *server*, dan masalah lain yang spesifik untuk aplikasi. Adapun ciri-ciri dari pengujian *blackbox* adalah sebagai beriukut :

1. *Black box* testing berfokus pada kebutuhan fungsional pada software,berdasarkan pada spesifikasi kebutuhan dari *software*
2. Merupakan pendekatan pelengkap dalam mencangkup *error* dengan kelas yang berbeda dari metode *white box testing*.

3. Melakukan pengujian tanpa pengetahuan detil struktur *internal* dari sistematau komonen yang dites. Juga disebut sebagai *behavioural testing*, *specification-based testing*, *input/ouput testing* atau *functional testing*.
4. Terdapat jenis test yang dapat dipilih berdasarkan pada tipe *testing* yang digunakan.
5. Kategori *error* yang akan diketahuai melalui *black box testing* seperti fungsi yang hilang atau tidak benar, *error* dari antar-muka, *error* dari struktur data atau akses eksternal database, *error* dari kinerja dan *error* dari inisialisasi.