PENGEMBANGAN DAN IMPLEMENTASI *BACKEND* & REST-API *DASHBOARD IOT* PADA *INTELLIGENCE CONTROL SYSTEM* (*ICS*) *SMART GREENHOUSE* MENGGUNAKAN PROTOKOL MQTT

(Proposal)

Oleh

M. Affan Siddiqie Asmara

2015031048

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS LAMPUNG

2023

PENGEMBANGAN DAN IMPLEMENTASI *BACKEND* & REST-API *DASHBOARD IOT* PADA *INTELLIGENCE CONTROL SYSTEM* (*ICS*) *SMART GREENHOUSE* MENGGUNAKAN PROTOKOL MQTT

Oleh

M. Affan Siddiqie Asmara

Proposal Penelitian

Sebagai Salah Satu Syarat untuk Mencapai Gelar

SARJANA TEKNIK

Pada

Jurusan Teknik Elektro

Fakultas Teknik Universitas Lampung

FAKULTAS TEKNIK

UNIVERSITAS LAMPUNG

BANDAR LAMPUNG

2023

DAFTAR ISI

Halaman

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1. Latar belakang

Indonesia, sebagai negara agraris, mengandalkan pertanian sebagai tulang punggung kehidupan, terutama di kalangan masyarakat kecil. Saat ini, kegiatan pertanian merajalela di berbagai komunitas di Indonesia, terutama di lingkungan masyarakat kecil. Namun, masyarakat kecil yang berada di daerah terpencil masih mengalami kendala akibat minimnya pemanfaatan dan pengembangan teknologi yang dapat mendukung pengelolaan lahan pertanian dan hasilnya. Sebagian besar petani masih sangat tergantung pada kondisi cuaca alam, sehingga hasil pertanian tidak selalu memuaskan ketika cuaca tidak sesuai harapan [1]. Aktivitas manusia untuk mempertahankan hidupnya juga berdampak signifikan pada peningkatan kebutuhan lahan. Akibatnya, lahan pertanian menjadi unsur krusial dalam mendukung kehidupan manusia. Pemanfaatan lahan ini semakin meningkat karena banyak orang menggunakan lahan untuk tempat tinggal, usaha, pembangunan akses umum, dan fasilitas lainnya, yang pada akhirnya mengakibatkan keterbatasan luas lahan. Penggunaan lahan yang tidak terkendali ini dapat mengakibatkan gangguan pada keseimbangan ekosistem [2].

Agar pemantauan data pada *smart greenhouse* dapat dilakukan dengan baik maka diperlukan sebuah *dashboard* untuk memonitoring data-data yang telah diterima dari sensor yang ada pada *smart greenhouse*. *Dashboard* adalah alat yang menyediakan antarmuka visual, yang menggabungkan dan menyajikan informasi penting untuk mencapai tujuan tertentu secara sekilas. Tampilan visual *dashboard* yang mampu mengkomunikasikan informasi dengan jelas, cepat, dan memberikan persepsi benar-benar menjadi kunci keberhasilan *dashboard*. Konsep visualisasi data dan informasi akan digunakan saat merancang antarmuka *dashboard*. Visualisasi data dan informasi terkait hal-hal mengenai persepsi visual dan media penyajian data, penyampaian komponen *dashboard* harus mengutamakan efektifitas penyampaian informasi untuk memudahkan pengguna melihat,

memantau dan membantu dalam mengambil keputusan yang tepat. dalam waktu nyata [3].

Dalam pengembangan *dashboard*, keberadaan *backend* sangat penting karena berperan sebagai fondasi yang mendukung pengolahan dan penyimpanan data secara efisien. *Backend* merupakan bagian dari sistem yang bertanggung jawab untuk mengelola basis data, menyimpan informasi dari sensor, dan mengolah data mentah menjadi informasi yang dapat ditampilkan dengan jelas pada *dashboard*.

Implementasi *backend* pada *dashboard* ini mengadopsi protokol MQTT (*Message Queuing Telemetry Transport*) sebagai salah satu komponen utama dalam sistem *backend*. Protokol MQTT dipilih karena kemampuannya untuk menyampaikan data secara ringan dan efisien melalui jaringan, yang sangat relevan untuk aplikasi *Internet of Things* (IoT) seperti *Smart Greenhouse*. MQTT memungkinkan perangkat di dalam *greenhouse* untuk mengirimkan data ke *backend* secara real-time, memastikan bahwa informasi yang disajikan pada *dashboard* selalu terkini. *Virtual Private Server* (VPS) digunakan sebagai lingkungan *hosting* untuk *backend*, dan *Node.js*, sebagai *platform* pengembangan *server-side* yang berbasis *JavaScript*, menjadi pilihan dalam mengimplementasikan *backend* pada penelitian ini.

1.2. Rumusan Masalah

Rumusan masalah dari penelitian ini adalah :

1. Bagaimana data yang dikirimkan oleh sensor-sensor yang digunakan pada ICS dapat dikirimkan menggunakan protokol MQTT (*Message Queuing Telemetry Transport*)
2. Bagaimana menjalankan program *backend* pada *Virtual Private Server* (VPS)
3. Bagaimana data yang telah diterima dari sensor dapat disimpan dan diolah kedalam *database postgreeSQL*
4. Bagaimana cara data yang telah disimpan dapat diambil dan diolah sebagai bahan penelitian lain.

1.3. Batasan Masalah

Adapun batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Penelitian berfokus pada mengimplementasikan protokol komunikasi *Message Queuing Telemetry Transport (MQTT)* pada *Virtual Private Server (VPS)* yang terhubung dengan ICS *Smart Greenhouse* dan mengirimkan data ke *database PostgreeSQL*, dan perancangan API yang akan digunakan oleh *frontend* dan *mobile apps.*
2. Penelitian ini tidak mencakup proses pembuatan *dashboard* dan alat sensor *Intelligence Controlling System* pada Smart Greenhouse.

1.4. Tujuan Penelitian

Adapun tujuan penelitian dalam penelitian ini adalah sebagai berikut:

1. Mengimplementasikan sebuah protokol komunikasi *Message Queue Telemetry Transport (MQTT)* dalam proses pengiriman dan menerima data sensor.
2. Menyimpan data yang diterima oleh sensor pada ICS *Smart Greenhouse* kedalam *database PostgreeSQL*.
3. Membangun API yang dapat digunakan untuk mengambil data yang tersimpan di dalam *database*.

1.5. Manfaat Penelitian

Adapun manfaat dari penelitian adalah :

1. Optimalisasi kegiatan pertanian. Petani dan peneliti dapat memantau kondisi tanaman secara *real-time*, melalui data sensor yang dikirimkan oleh *smart* ICS
2. Memudahkan para peneliti terkait dan petani untuk mengolah data yang diperlukan.
3. Menyimpan data yang dikirimkan oleh *Intelligence Controlling System* kedalam sebuah *database* yang dapat diambil dan digunakan pada penelitian lebih lanjut.

1.7. Sistematika Penulisan

Sistematika penulisan dari penelitian ini adalah:

BAB I : PENDAHULUAN

Bab ini menjelaskan tentang latar belakang, tujuan penelitian, rumusan masalah, batasan masalah, manfaat penelitian, dan sistematika penulisan pada penelitian ini.

BAB II : TINJAUAN PUSTAKA

Membahas tentang penelitian-penelitian sebelumnya pada tinjauan pustaka, dan dasar-dasar teori dari penelitian pengembangan dan implementasi *backend dashboard* IoT pada *intelligence controlling system* (ICS) *smart greenhouse* menggunakan protokol MQTT

BAB III : METODOLOGI PENELITIAN

Menjelaskan waktu, tempat, alat dan bahan, dan metode penelitian beserta tahapannya mengenai pengembangan dan implementasi *backend dashboard* IoT pada *intelligence control system* (ICS) *smart greenhouse* menggunakan protokol MQTT.

BAB IV : HASIL DAN PEMBAHASAN

Menjelaskan bagaimana proses pelaksanaan penelitian dan juga hasil yang didapatkan pada penelitian.

BAB V : KESIMPULAN DAN SARAN

Memaparkan kesimpulan apa saja yang didapat dari proses pelaksanaan penelitian dan memberikan saran untuk dilakukan pada penelitian selanjutnya.

DAFTAR PUSTAKA

Bab ini berisikan referensi dari penulisan dan pelaksanaan riset.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Peneliti mengambil beberapa penelitian terkait sebagai referensi ataupun acuan untuk penelitian yang dilakukan.

Penelitian dilakukan oleh Muhammad Farid Ali Safii, Suwanto Raharjo, dan Uning Lestari pada tahun 2019, berjudul "Analisis *Quality of Service* Protokol MQTT dan HTTP pada Penerapan Sistem *Monitoring* Suhu Berbasis *NodeMCU* (Studi Kasus Ruang *Server* Kampus 3 IST AKPRIND Yogyakarta), peneliti merancang sistem monitoring pada sebuah ruang *server* dengan menggunakan *NodeMCU* dengan mengguankan sensor suhu dan kelembapan DHT11, pada percobaan ini peneliti menggunakan protokol MQTT dalam proses pengiriman data dari sensor, dan membandingkan nya dengan protocol HTTP [4].

Penelitian terkait berikutnya dilakukan oleh Narges A-hussein dan Ayman D Salman pada tahun 2020, mengenai IoT *Monitoring System Based on* MQTT *Publisher/Subscriber Protocol*. Peneliti melakukan percobaan pengiriman data dari sensor DHT11 menggunakan NodeMCU ESP32 dengan metode publish dan subscribe protocol MQTT, data yang dipublish pada NodeMCU menggunakan *topic* kemudian *disubscribe* pada *Node-red* yang telah di install pada Raspi 3 [5].

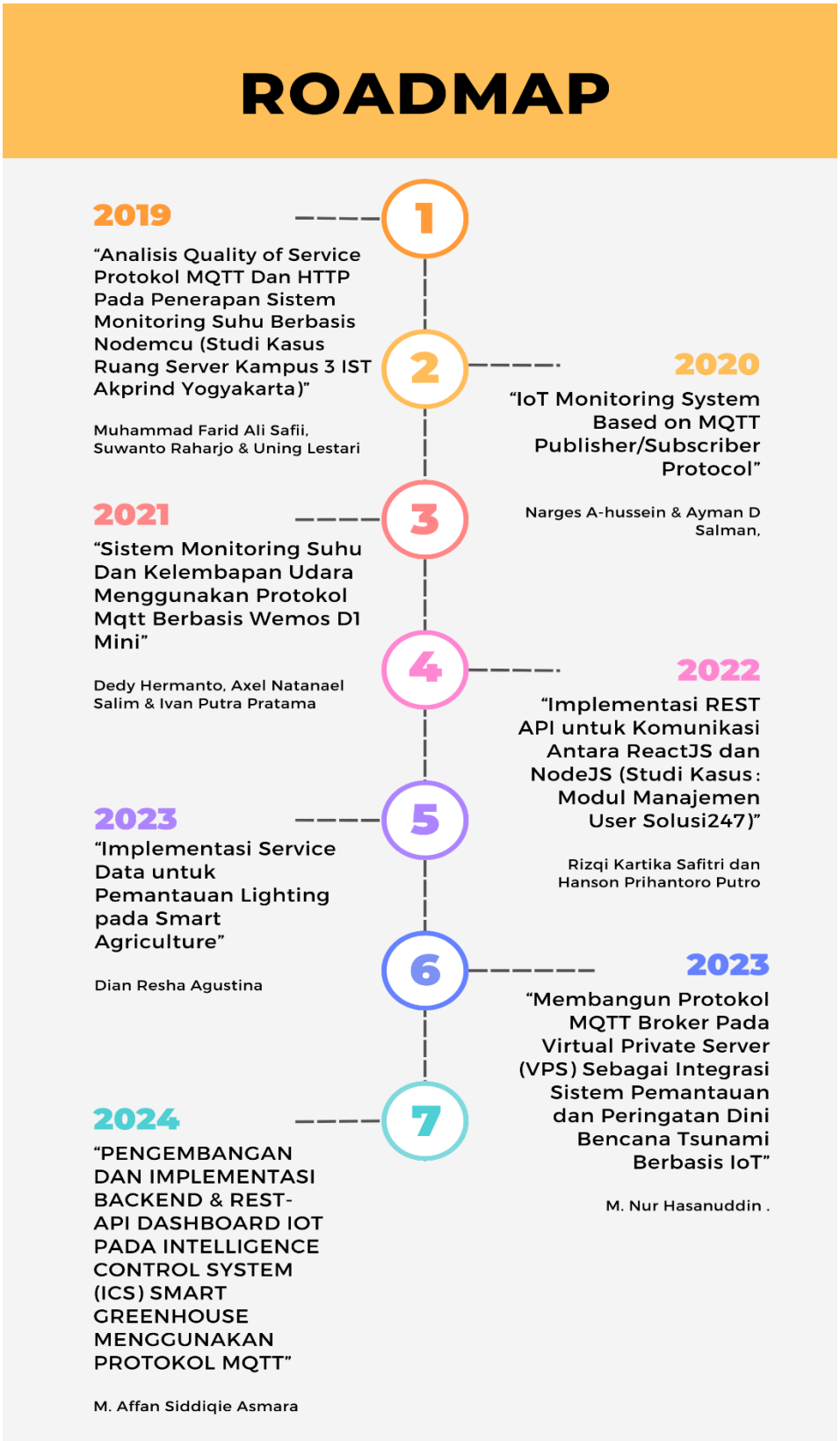
Penelitian dilakukan oleh Dedy Hermanto, Axel Natanael Salim dan Ivan Putra Pratama pada tahun 2021, berjudul "Sistem Monitoring Suhu dan Kelembapan Udara Menggunakan Protokol MQTT Berbasis Wemos D1 Mini". Penelitian ini berfokus pada visualisasi data yang diterima dari dht11 melalui ESP8266 Menggunakan MQTT. Penelitian ini menggunakan *firebase* sebagai *database* untuk menyimpan dan mengolah data yang diterima [6].

Penelitian dilakukan oleh Rizqi Kartika Safitri dan Hanson Prihantoro Putro pada tahun 2022 mengenai implementasi *Rest-API* untuk komunikasi antara *ReactJS* dan *NodeJS*, penelitian ini menjelaskan bagaimana *Rest-API* digunakan sebagai

jembatan komunikasi dalam pengembangan modul manajemen user yang lebih baik saat menggunakan *ReactJS* dan *NodeJS*. Perancangan *Rest-Api* menghasilkan endpoint yang akan menjembatani komunikasi dengan *rest client* pada bagian *Front End* [7].

Dian Resha Agustina. dalam jurnal yang diterbitkan pada September 2023, "Implementasi *Service Data* untuk Pemantauan *Lighting* pada *Smart Agriculture*", mengungkapkan inovasi dalam penggunaan data terstruktur dan teknologi Internet of Things (IoT) untuk pemantauan pencahayaan dalam pertanian cerdas. Penelitian ini menggabungkan teknologi *Service Worker* untuk memfasilitasi aplikasi web menggunakan *Node.JS*, *MongoDb* dan *broker RabbitMQ* [8].

Pada penelitian sebelumnya yang dilakukan oleh M. Nur Hasanuddin pada tahun 2023 menjelaskan mengenai pengembangan *protocol MQTT broker* pada *Virtual Private Server (VPS)* sebagai integrasi sistem pemantauan dan peringatan dini bencana tsunami berbasis *IoT*, pada pengembangan sistem pemantuan ini menggunakan protocol *MQTT* dengan mengirimkan data dari sensor serta *forecasting* pada data yang akan datang, pada penelitian ini dilakukan perbandingan antara *protocol MQTT* dan juga *HTTP* [9].



Gambar 2.1 *Roadmap*

2.2. *Smart Greenhouse*

Smart Greenhouse merupakan evolusi dari rumah kaca tradisional, di mana teknologi sensor dan otomatisasi digunakan untuk meningkatkan efisiensi dan pengendalian lingkungan pertanian. Dengan memanfaatkan sensor suhu, kelembaban, intensitas cahaya, dan CO2, serta sistem otomatisasi seperti pengairan dan pengaturan suhu [10]. *Smart Greenhouse* menawarkan pendekatan terintegrasi untuk meningkatkan produktivitas dan keberlanjutan dalam pertanian. Sistem ini memungkinkan pengguna untuk memantau dan mengontrol kondisi pertumbuhan tanaman secara *real-time*, membuat penyesuaian berdasarkan data, dan bahkan melakukan tindakan otomatis untuk menjaga kondisi yang optimal bagi tanaman.

2.3. *Intelligence Controlling System (ICS)*

Intelligence Controlling System (ICS) adalah sistem terpadu yang dirancang untuk mengoptimalkan dan mengontrol berbagai aspek dalam pertanian, termasuk pemantauan lingkungan, irigasi, pemupukan, dan manajemen sumber daya secara menyeluruh. ICS menggunakan teknologi sensor, dan konektivitas berbasis *Internet of Things* (IoT) untuk memberikan solusi terpadu yang memungkinkan petani untuk mengambil keputusan berdasarkan data secara akurat dan efisien. Salah satu elemen kunci dari ICS adalah integrasi data dari berbagai sensor yang terdistribusi di seluruh lahan pertanian. Sensor ini dapat mencakup sensor suhu, kelembaban tanah, tingkat nutrisi, dan lainnya. Data yang dikumpulkan oleh sensor-sensor ini kemudian diolah oleh sistem untuk memberikan pemahaman yang lebih mendalam tentang kondisi pertanian.

2.4. Internet Of Things (IoT)

Internet of Things (IoT) merupakan sebuah konsep yang sangat populer di masa ini dimana sebuah perangkat ataupun objek dapat ditanamkan sebuah teknologi seperti sensor yang dapat berkomunikasi, mengendalikan, menghubungkan dan bertukar data dengan perangkat lain dengan memanfaatkan internet.

Pada dasarnya IoT adalah sebuah ekosistem yang dilakukan secara berkala dalam pengambilan data dari sebuah perangkat ataupun sensor yang kemudian data tersebut dikirimkan menggunakan jaringan internet sebagai jalur komunikasi antara perangkat dengan *server* IoT, dimana *server* dapat berperan sebagai penyimpanan data. Manfaat yang didapat dalam penggunaan *Internet of Things* (IoT) ialah membuat pekerjaan yang dilakukan dapat menjadi lebih cepat, mudah dan efisien sehingga hal ini dapat meningkatkan kinerja penggunanya [11].

Pada penelitian ini, pengemembangan sistem yang dilakukan dalam penelitian ini menerapkan konsep IoT, dimana terdapat sebuah sensor pemantauan yang terhubung dengan internet, dan sensor tersebut dapat diakses datanya secara jarak jauh menggunakan jaringan internet.

2.5. Message Queue Telemetry Transport (MQTT)

Message Queue Telemetry Transport (MQTT) adalah sebuah protokol standar baru yang banyak digunakan dalam pengembangan sebuah sistem yang berbasis *Internet of Things (IoT).* Sistem ini sangat cocok dalam penggunaannya pada perangkat IoT dikarenakan protokol ini sangat ringan, mudah digunakan, dan hemat energi. *MQTT* dalam pengembangannya saat ini sudah banyak digunakan dalam berbagai industry seperti otomotif, manufaktur, telekomunikasi, minyak dan gas.

MQTT saat ini menjadi pilihan sebagai jalur komunikasi IoT dikarenakan beberapa kelebihan seperti :

1. Ringan dan Efisien.
   MQTT-clients yang sangat ringan, hanya membutuhkan sedikit sumberdaya dalam penggunaan nya sehingga dapat digunakan dalam mikrokontroller dengan memori berukuran kecil. Dalam pengiriman datanya, *MQTT Message Header* yang berfungsi sebagai identitas alamat komunikasi juga sangat ringan sehingga penggunaan *network bandwidth* juga sangat optimal.
2. Pengiriman pesan yang reliabel.
   Reliabilitas dalam pengiriman pesan sangat penting dalam berbagai perangkat IoT. *MQTT* memiliki 3 level *Quality of Service (QoS)* yang dapat disesuaikan.

3. Komunikasi dua arah.

   MQTT dapat melakukan pengiriman pesan antara *device* ke *cloud* dan juga *cloud* ke *device*. Hal ini memudahkan membagikan pesan dalam group dengan *broadcasting*.

4. Dapat berjalan dalam jaringan yang kurang terkendali.

   Banyak perangkat IoT terhubung melalui jaringan seluler yang kurang terkendali. *MQTT* dapat dijalankan secara persisten sehingga mengurangi waktu untuk menghubungkan kembali antara klien dengan *broker*.

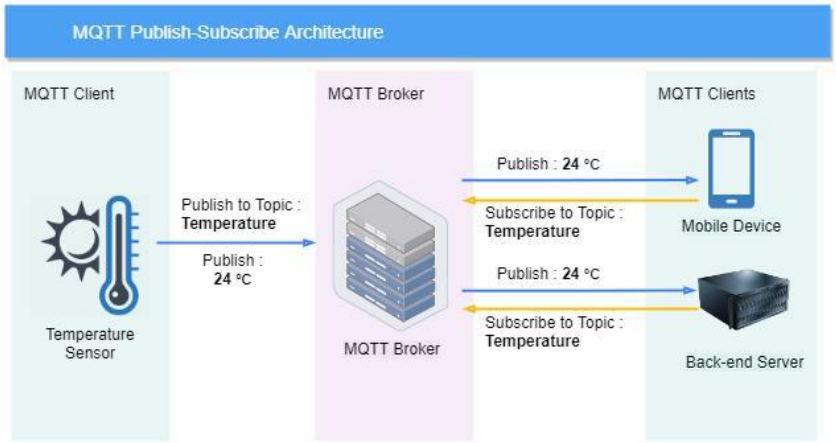
5. Skala besar.

   MQTT dapat terhubung dengan jutaan perangkat IoT.

6. Keamanan.

   MQTT memudahkan untuk mengenkripsi pesan menggunakan TLS dan mengautentikasi klien menggunakan protocol autentikasi modern.

Cara kerja dari *MQTT* adalah dengan menggunakan metode *publish / subsribe.* Metode ini bekerja dengan cara satu perangkat mengirimkan sebuah pesan (*publish*), kemudian perangkat lain yang bertindak sebagai penerima (*subscribe*) dapat menerima pesan yang telah dikirimkan tadi sesuai dengan topic yang sesuai. Seluruh pengiriman dan penerimaan pesan ini diatur oleh *MQTT* Broker [12]. Cara kerja dari MQTT dapat dilihat pada Gambar 2.1.



Gambar 2.1. Cara Kerja MQTT

(sumber : https://www.tatvasoft.com/blog/wp-content/uploads/2021/10/MQTT-Publish-Subscribe-Architecture.jpg)

Pada sistem yang akan dibangun dalam penelitian ini, *MQTT* yang digunakan sebagai broker adalah Mosquitto. Program *MQTT* tersebut akan diinstall ke sebuah *Virtual Private Server (VPS)* dan dapat diakses dari luar jaringan *VPS* sebagai protokol komunikasi dan pengiriman data. Pemilihan menggunakan Broker *MQTT* Mosquitto adalah karena kelebihannya yang ringan, mudah untuk digunakan, gratis, *Open-source*, dan tersedia di berbagai *platform*.

2.6. Mosquitto MQTT

Mosquitto adalah sebuah perangkat lunak *open-source* yang berfungsi sebagai *message broker* untuk protokol *MQTT (Message Queuing Telemetry Transport)*. *Mosquitto* dikembangkan oleh *Eclipse Foundation* dan tersedia untuk digunakan secara bebas dengan lisensi EPL/EDL [13].

Mosquitto dirancang untuk menjadi sebuah *message broker* yang ringan dan sederhana, sehingga cocok digunakan pada berbagai jenis perangkat, dari komputer papan tunggal hingga *server*. *Mosquitto* juga menyediakan perpustakaan C untuk mengimplementasikan klien *MQTT*, serta klien *MQTT* pada baris perintah seperti *mosquitto_pub* dan *mosquitto_sub* yang sangat populer.

Gambar 2.2. MQTT Broker Mosquitto

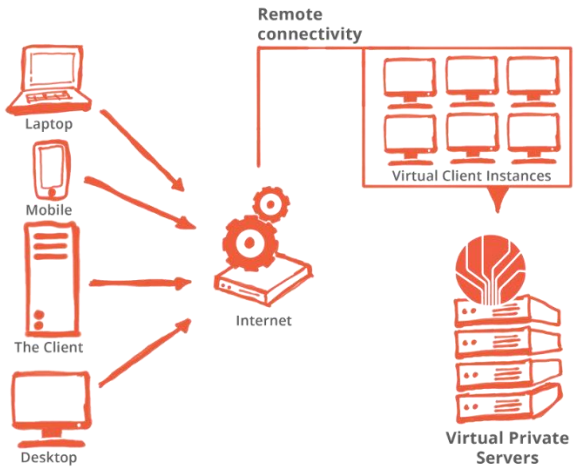
(sumber : https://images.app.goo.gl/FBdFPxsm7se2prMC6)

Mosquitto mendukung beberapa versi protokol *MQTT*, yaitu versi 5.0, 3.1.1, dan 3.1, sehingga dapat digunakan pada berbagai jenis aplikasi. Protokol *MQTT* yang digunakan oleh *Mosquitto* menggunakan model *publish*/*subscribe* yang sangat efisien untuk mengirim dan menerima pesan, dan dapat digunakan untuk berbagai jenis aplikasi, termasuk pada *Internet of Things (IoT)* dan perangkat berdaya rendah.

Beberapa fitur penting dari *Mosquitto* antara lain adanya dukungan untuk autentikasi dan enkripsi, adanya dukungan untuk *multiple broker*, serta kemampuan untuk mengatur *quality of service* (QoS) pada pengiriman pesan. *Mosquitto* juga dapat diintegrasikan dengan berbagai jenis perangkat dan *platform*, sehingga menjadi solusi yang fleksibel dan dapat disesuaikan dengan kebutuhan pengguna.

2.7. Virtual Private Server (VPS)

Virtual Private Server (VPS) merupakan sebuah teknologi virtualisasi dari sebuah *physical server*. Beberapa *Virtual Private Server* dapat dibuat dalam sebuah *physical server*. *VPS* dapat berjalan seperti sebuah *server* mandiri yang memiliki *full root access*, sistem operasi, konfigurasi, user, pemrosesan, *CPU*, dan *RAM* nya masing-masing sehingga sebuah *VPS* tidak akan mempengaruhi *VPS* lain dalam satu *physical server* [14].



Gambar 2.3. *Virtual Private Server*

(sumber : https://www.irontree.co.za/wp-content/uploads/2019/06/VPS-Graphic-1024x768.png)

Pada penelitian ini digunakan sebuah *VPS* dikarenakan dalam membangun sebuah protokol komunikasi pengiriman data dibutuhkan sebuah layanan yang dapat beroperasi terus menerus. Apabila tidak menggunakan *VPS* melainkan menggunakan komputer/laptop secara lokal, terdapat kemungkinan bahwa

perangkat tersebut tidak dapat dinyalakan secara terus menerus sehingga protocol komunikasi data juga dapat terganggu. Penggunaan *VPS* pada penelitian ini juga sangat penting dikarenakan seluruh proses pengiriman dan penyimpanan data akan dilayani oleh *VPS* secara terus menerus.

2.8. Backend Development

Pengembangan *Backend* pada *Dashboard* IoT *Smart Greenhouse* mencakup pembangunan infrastruktur yang mendukung fungsi dan integrasi semua komponen dalam sistem. *Dashboard* ini menjadi jantung operasional yang mengelola data dari sensor, memproses informasi, dan menyajikan hasilnya dalam bentuk yang dapat dimengerti oleh pengguna. Fungsi utama dari *Backend* pada *Dashboard* IoT *Smart Greenhouse* melibatkan:

1. Manajemen Data Sensor: *Backend* mengatur penerimaan dan penyimpanan data dari sensor-sensor yang terpasang di dalam *greenhouse*.
2. Integrasi dan Pemrosesan Data: Data yang dikumpulkan dari sensor diolah dan diintegrasikan oleh *Backend* untuk memberikan gambaran keseluruhan tentang kondisi lingkungan dalam *greenhouse*.
3. Interaksi dengan Database: *Backend* berkomunikasi dengan *database* untuk menyimpan dan mengelola data secara efisien. Ini memastikan keberlanjutan dan ketersediaan data untuk dianalisis atau ditampilkan pada *dashboard*.
4. API untuk Koneksi *Frontend*: *Backend* menyediakan API untuk menghubungkan *Dashboard* dengan *Frontend* (antarmuka pengguna).

2.9. ***Node.js***

Node.js adalah sebuah *platform runtime JavaScript* yang dibangun di atas mesin *JavaScript* V8 milik Google yang berguna untuk proses development aplikasi secara cepat dan efisien serta mempermudah pembangunan aplikasi berbasis jaringan yang memiliki *scalability* (daya pengembangan) yang tinggi, Node Js

menggunakan *event-driven non-blocking* I/O model yang membuat *Node.Js* ringan dan efisen, cocok untuk aplikasi data-intensive realtime yang berjalan pada cross *platform* [15].

Gambar 2.4. Node.js

(sumber:https://upload.wikimedia.org/wikipedia/commons/d/d9/Node.js_logo.svg)

Dengan menggunakan Node.js, pengembang dapat menulis kode *JavaScript* yang dapat dijalankan di sisi *server*, mengakses sistem file, dan melakukan tugas lainnya yang memerlukan akses ke sumber daya sistem.

Beberapa manfaat yang ditawarkan oleh Node.js adalah kecepatan eksekusi, skalabilitas, kemampuan untuk mengatasi banyak permintaan (*concurrency*), dan kemudahan penggunaan.

2.10. *Express.js*

Express.js adalah kerangka kerja web yang dirancang untuk mempermudah pengembangan aplikasi web dengan menggunakan bahasa pemrograman *JavaScript* di sisi server. Dibangun di atas *Node.js, Express.js* menyediakan cara yang cepat dan minimalis untuk membuat server HTTP dan mengelola rute aplikasi. Salah satu kekuatan Express.js adalah pendekatannya yang ringkas, memberikan fleksibilitas kepada pengembang untuk membangun aplikasi dengan struktur yang mudah dimengerti [16].

Gambar 2.5. Express.js

(sumber:https://miro.medium.com/v2/resize:fit:1400/1*i2fRBk3GsYLeUk_Rh7AzHw.png)

Express.js memudahkan penanganan permintaan HTTP dengan memungkinkan definisi rute-rute yang menentukan cara server menanggapi permintaan tertentu. Ini memungkinkan pengembang untuk membuat aplikasi web dengan mudah, termasuk pembuatan *endpoint API, pengaturan middleware*, dan manajemen sesi. Selain itu, *Express.js* memiliki sistem templat yang dapat diintegrasikan dengan berbagai mesin templat, memfasilitasi pembuatan tampilan dinamis.

Kerangka kerja ini juga mendukung pengembangan aplikasi *RESTful* dengan menyediakan alat untuk mengelola metode HTTP seperti *GET, POST, PUT*, dan *DELETE*. *Express.js* juga mendukung penggunaan *middleware*, yang dapat digunakan untuk menangani berbagai aspek seperti otentikasi, log, dan manipulasi permintaan sebelum mencapai penanganan utama.

2.11. *PostgreSQL*

PostgreSQL merupakan salah satu sistem basis data yang tergolong kedalam DBMS atau *Database Management System* yaitu suatu sistem perangkat lunak yang kegunaanya didesain untuk membantu pekerjaan penggunanya seperti mengelola suatu basis data dan menjalankan suatu perintah operasi terhadap data yang diminta oleh banyak penggunanya. *PostgreSQL* digunakan sebagai manajemen basis data yang menampung dan mengolah data aplikasi dan dapat dijalankan pada sistem operasi seperti *Windows, Linux, Mac,* dan lain sebagainya [17].

Gambar 2.6. PostgreeSQL

(sumber:https://itbox.id/wpcontent/uploads/2023/02/1_7AOhGDnRL2eyJMUidCHZEA.jpeg)

PostgreSQL dapat digunakan sebagai aplikasi manajemen *database* dengan aplikasi bawaannya *pgAdmin*. *PostgreSQL* bekerja dengan menangani proses pengolahan data dengan *sintaks query* SQL.

PostgreSQL dikenal sebagai sistem pengolahan *database* open source termaju di dunia yang umum digunakan dalam pengembangan perangkat lunak dengan kemampuannya yang mampu memproses banyak *sintaks query* dalam satu waktu .

PostgreSQL dipilih dalam penelitian ini sebagai *database* penyimpanan data dikarenakan beberapa kelebihan dari PostgreSQL yaitu *Open-source*, skalabilitas nya yang baik, mudah dikonfigurasi, dapat menangani volume data yang besar, dan lebih cepat dibandingkan beberapa *Database Management System* (DBMS) yang lain seperti *MySQL* atau *MongoDB* .

2.12. *Application Programming Interface (API)*

Application Programming Interface (API) adalah sebuah antarmuka yang memungkinkan aplikasi untuk berinteraksi dengan aplikasi atau layanan lainnya secara programatik. API menyediakan kumpulan instruksi dan protokol yang dapat digunakan oleh pengembang untuk membangun aplikasi yang dapat berkomunikasi dengan sistem atau layanan yang telah ada [18] .

API memungkinkan pengembang untuk mengakses fungsi dan layanan yang disediakan oleh sistem atau layanan lainnya tanpa harus mengetahui seluruh detail

implementasi di belakangnya. Dengan menggunakan API, pengembang dapat mempercepat proses pengembangan, mengurangi biaya, dan meningkatkan efisiensi dalam pengembangan aplikasi karena tidak perlu membangun seluruh fitur dari awal.

Contoh penggunaan API adalah ketika sebuah aplikasi mengakses data dari layanan seperti *Facebook*, *Twitter*, atau *Google Maps*. API yang disediakan oleh layanan tersebut memungkinkan aplikasi untuk mengambil data dari layanan tersebut dan menggunakannya dalam aplikasi mereka sendiri .

2.13. REST-API

REST API adalah salah satu jenis dari API yang didasarkan pada arsitektur *REST (Representational State Transfer)*. *REST* sendiri adalah sebuah arsitektur perangkat lunak yang terdiri dari aturan dan konvensi yang digunakan untuk membuat web service.

Dalam *REST API*, setiap sumber daya (*resource*) diidentifikasi dengan URI (*Uniform Resource Identifier*) dan diakses melalui metode HTTP seperti *GET*, *POST, PUT, DELETE*, dan sebagainya.

Selain itu, REST API juga menggunakan format data yang ringan dan terstruktur, seperti JSON (*JavaScript Object Notation*) atau XML (*eXtensible Markup Language*) untuk pertukaran data antara aplikasi dan *server*. Format data ini memudahkan pengolahan data oleh aplikasi klien (*client*) yang mengakses API.
Dalam penggunaannya, REST API sering digunakan oleh pengembang aplikasi untuk mengintegrasikan berbagai aplikasi atau layanan secara terdistribusi dan *scalable*. Selain itu, REST API juga sering digunakan dalam pembuatan aplikasi *mobile*, *Internet of Things* (IoT), dan integrasi antar sistem yang berbeda *platform*.

2.14. GitHub

GitHub adalah *platform hosting* kode sumber berbasis *web* yang sangat populer di kalangan pengembang perangkat lunak. *Platform* ini menyediakan repositori *git*

publik dan pribadi yang memungkinkan pengguna untuk menyimpan, mengelola, dan berkolaborasi dalam pengembangan perangkat lunak secara terdistribusi. *GitHub* memiliki fitur-fitur kolaborasi yang kuat, yang memungkinkan tim pengembang bekerja bersama dalam proyek perangkat lunak.

2.15. Visual Studio Code

Visual Studio Code adalah editor teks gratis dan *open-source* yang dikembangkan oleh *Microsoft*. *Visual Studio Code* memungkinkan pengguna untuk menulis kode pada berbagai bahasa pemrograman, termasuk HTML, CSS, *JavaScript*, PHP, *Python*, dan lain-lain. *Visual Studio Code* berjalan pada sistem operasi *Windows, Mac, dan Linux*. Editor teks ini dilengkapi dengan fitur-fitur seperti fitur *debugging*, Git *integration*, *autocomplete*, *snippet*, *syntax highlighting*, dan lain-lain .

Visual Studio Code memiliki beberapa kelebihan yang membuatnya populer di kalangan pengembang perangkat lunak. Kelebihan-kelebihan tersebut antara lain adalah :

- Gratis dan *open-source*
- Ringan dan cepat
- Mudah digunakan
- Mendukung banyak bahasa pemrograman
- Memiliki banyak ekstensi dan plugin

2.16. Kanban

Kanban merupakan sebuah teknik manajemen proyek sederhana yang efektif dan efisien dalam memecahkan sebuah masalah. Metode ini pada awalnya lebih fokus digunakan dalam industri manufaktur, namun seiring berjalannya waktu metode ini semkain banyak digunakan di berbagai jenis industri dalam mengembangkan bisnisnya.

kanban memiliki konsep utama yaitu *visualize workflow* atau memvisualisasikan alur kerja dengan visual sehingga tim dapat lebih mudah dalam melakukan komunikasi dalam pekerjaan nya.

Visualisasi ini dilakukan dengan menggunakan sebuah kanban board yang berisi informasi proses-proses dalam pengembangan seperti perencanaan, perancangan, persetujuan, pengembangan, pengujian, pengintegrasian, dan penerapan [19].

Kanban memiliki beberapa kelebihan dibandingkan metode pengembangan proyek yang lain yaitu *Fleksibilitas*. Kanban dapat disesuaikan dengan kebutuhan dan dinamika proyek pengembangan perangkat lunak yang sering berubah-ubah. Tim pengembang dapat dengan mudah menyesuaikan kapasitas dan prioritas pekerjaan sesuai dengan perubahan yang terjadi.

Kanban memiliki tiga prinsip dasar dalam penggunaan nya yaitu :

1. Visualisasikaan pekerjaan.

   Kanban Board adalah sebuah model visual yang digunakan dalam Kanban untuk merepresentasikan alur kerja secara grafis, sehingga memudahkan dalam memantau dan mengevaluasi jalannya proses dari awal hingga akhir. Model *Kanban Board* ini dibuat sesuai dengan tahapan yang harus dilalui dalam pengembangan perangkat lunak [20].
2. Batasi pekerjaan yang sedang berlangsung.

   Pada tahap awal, tim akan menetapkan batasan pekerjaan untuk setiap alur pada *Kanban Board* yang disebut "*Work In Progress (WIP)*". Tujuan dari WIP ini adalah untuk mengurangi pemborosan dan membantu tim untuk fokus menyelesaikan pekerjaan yang sedang dalam proses, sebelum memulai pekerjaan baru setelahnya [20].
3. Fokus pada alur kerja

   Manfaat yang efektif dari penggunaan *Kanban* adalah tim dapat berfokus pada aliran kerja proyek dari awal hingga selesai. Dalam mencapai fokus ini, tim harus mengikuti dua prinsip yang telah dijelaskan sebelumnya. Fokus pada aliran kerja proyek ini membantu tim untuk mengidentifikasi hambatan yang mungkin terjadi dan mengambil tindakan yang diperlukan untuk memastikan aliran kerja proyek tetap lancar dan terus berjalan [20].

2.17. Postman

Postman adalah sebuah alat yang digunakan oleh para pengembang perangkat lunak untuk berinteraksi dengan API (*Application Programming Interface*). API seperti pelayan di restoran, yang mengambil pesanan Anda (permintaan data atau fungsi) dan memberikan balasan dari dapur (sistem atau aplikasi). Dengan *Postman*, pengembang bisa menguji API dengan mengirimkan permintaan-nya dan melihat apa yang dikembalikan. Misalnya, jika Anda ingin melihat data pengguna dari sebuah layanan, Anda bisa "meminta" data tersebut melalui *Postman*, dan layanan akan "memberikan" data yang diminta .

Postman sendiri kemudian digunakan untuk mengirim permintaan ke API. “Permintaan dapat mengambil (*GET*), menambah (*POST*), menghapus (*DELETE*), atau memperbarui data (*PUT*)”. Permintaan dapat digunakan untuk mengirim parameter, detail login, informasi otorisasi, atau data tubuh lain yang diperlukan. Perangkat lunak Postman dapat diunduh ke komputer pengguna (sistem operasi Windows, Mac, dan Linux didukung) atau digunakan dari peramban web dengan versi Web *Postman* .

BAB III
METODE PENELITIAN

3.1. Waktu dan Tempat Penelitian

Penelitian ini dilakukan pada :

Waktu : Agustus 2023 – Januari 2023

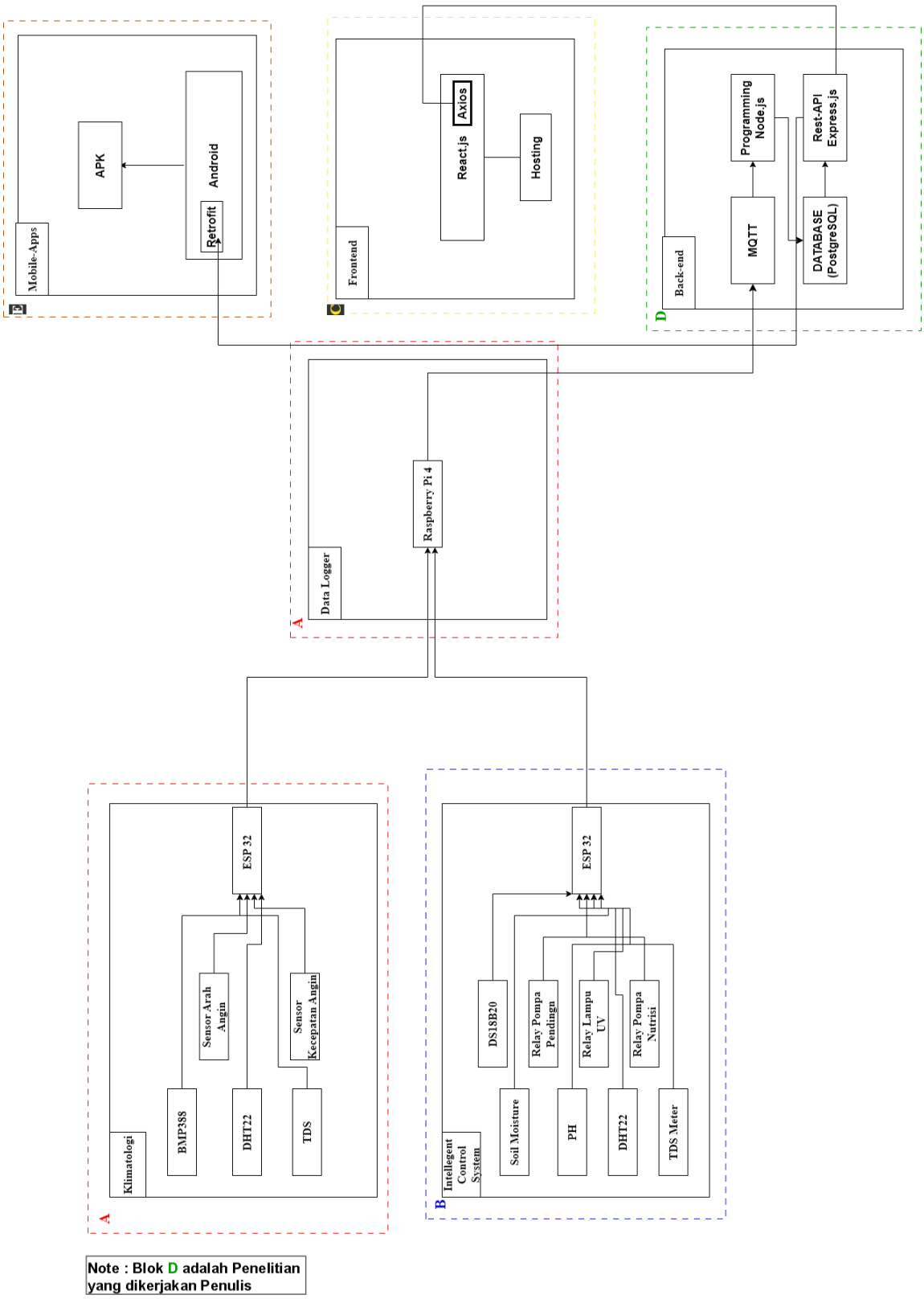
Tempat :

- Laboratorium Teknik Komputer, Teknik Elektro, Universitas Lampung
- Laboratorium terpadu fakultas pertanian universitas lampung

3.2. *Capstone Project*

Pengerjaan penelitian ini berkaitan dengan beberapa penelitian lain yang berkolaborasi untuk membangun sistem *Intelligence Controlling System* (ICS) pada *smart greenhouse*. Pada project yang akan dilakukan terdapat beberapa bagian yang akan dikerjakan secara terpisah. Pertama terdapat pengerjaan pada bagian *hardware* ICS *smart greenhouse* yang dikerjakan oleh dua orang dalam team ini, kemudian terdapat tiga orang yang menghandle *software* diantaranya adalah *frontend, mobile apps*, dan *backend* yang akan saya kerjakan. Diagram keseluruhan project yang dikerjakan dapat dilihat pada Gambar 3.1.

Peneliti hanya akan berfokus pada bagian *Backend* yang ditandai dengan garis putus berwarna hijau dan bagan D, sementara bagian lain seperti perangkat ICS yang merupakan rangkaian hardware sensor pemantauan dan juga bagian *front-end* sebagai dashboard monitoring akan dikerjakan dalam penelitian lain. *Backend* dalam proyek ini berfungsi sebagai penghubung antara perangkat sensor dengan *front-end*, dimana didalam *server* itu sendiri akan menjalankan berbagai service yang dibutuhkan dalam proyek ini seperti *database*, *MQTT Broker*, dan *API*.



Gambar 3.1. Diagram keseluruhan pengembangan sistem

3.3. Alat dan Bahan

3.3.1. Alat

Alat yang digunakan dalam pengerjaan penelitian dan pengembangan sistem yang dibuat adalah sebagai berikut :

Tabel 3.3.1.1. Alat berupa *Hardware* dan *Software* yang digunakan.

No	Perangkat	Spesifikasi	Kegunaan	Keterangan
1	Laptop	Processor Intel Core i5 10300H, RAM 16GB, SSD 512GB, Sistem Operasi Hackintosh	Perangkat pembuatan dan pengujian sistem.	
2	Virtual Private *Server* (VPS)	Penyedia : Niagahoster.co.id, Lokasi: Singapore, 3 CPU Core, RAM 3GB, Storage 60GB.	*Virtual private server* yang digunakan sebagai pusat layanan sistem yang akan dibuat.	Diakses melalui *Secure Shell (SSH)*
3	MQTTX	MQTTX Versi 1.9.6	Software yang digunakan untuk pengujian broker mqtt dalam mengirim dan menerima data	Terinstall di Laptop
4	Visual Studio Code	VS Code versi 1.84.2	Software yang digunakan dalam penulisan kode program untuk pengambilan dan penyimpanan data dari sensor	Terinstall di Laptop
3	PostgreSQL	PostgreSQL *database* versi 7.4	Program *database* yang digunakan dalam penyimpanan data dari sensor	Terinstall di VPS dan laptop

No	Perangkat	Spesifikasi	Kegunaan	Keterangan
4	Mosquitto MQTT Broker	Mosquitto versi 1.6.10	Program yang di install di dalam VPS sebagai broker MQTT	Terinstall di VPS dan laptop
5	Node.js & NPM	Node.js versi v16.19.0. NPM versi 8.19.3	Aplikasi yang digunakan untuk menjalankan kode program untuk penyimpanan data	Terinstall di VPS dan laptop
6	Postman	Postman V.10.19	Program yang digunakan untuk checking Endpoint API	Terinstall di Laptop

3.3.2. Bahan

Bahan yang digunakan dalam pengerjaan penelitian dan pengembangan sistem ini berupa data yang didapatkan dari sensor ICS adalah sebagai berikut :

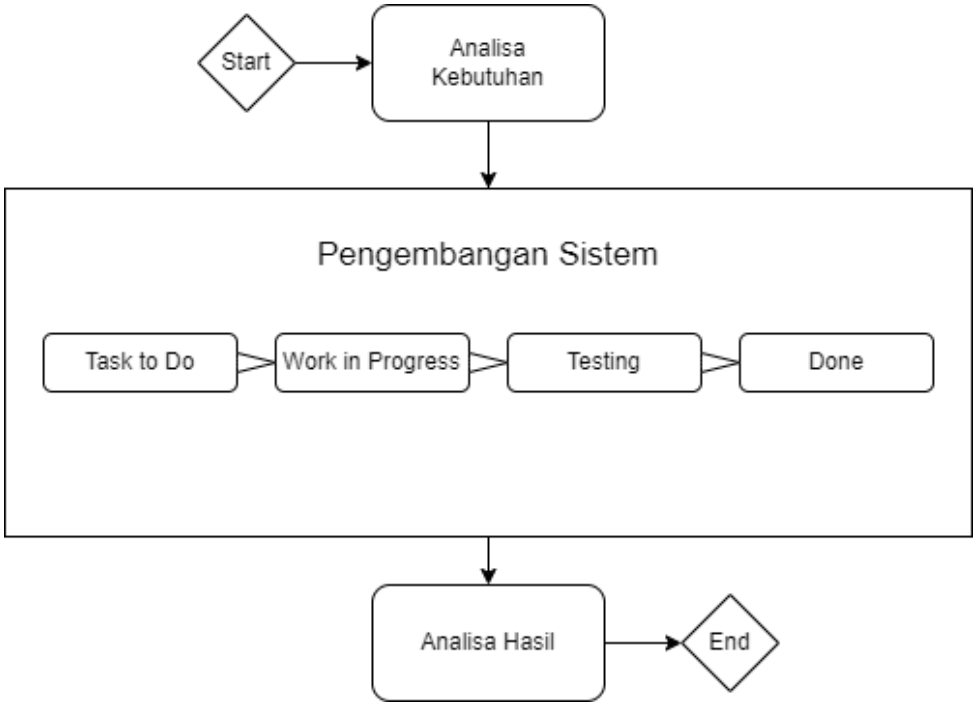
Tabel 3.3.2.1 Bahan berupa data sensor dari *esp*.

NO	Data yang diterima	Keterangan
1	Timestamp	Data yang menunjukan waktu
2	Ph	Data yang menunjukan tingkat keasaman atau kebasaan air
3	Tds	Data kualitas air
4	Suhu air	Data yang menunjukan suhu air
5	Arah mata angin	Data yang menunjukan arah mata angin pada greenhouse
6	Kecepatan angin	Data yang menunjukan kecepatan angin dalam greenhouse

NO	Data yang diterima	Keterangan
7	Soilmoisture	Data yang menunjukan kelembapan tanah pada tumbuhan
8	Suhu ruangan	Data yang menunjukan suhu greenhouse
9	Tekanan udara	Data yang menunjukan tekanan udara di dalam greenhouse
10	Pompa nutrisi	Data yang menunjukan apakah sistem pompa nutrisi menyala tau tidak
11	Pompa air	Data yang menunjukan apakah sistem pompa air menyala atau tidak
12	Kelembapan	Data yang menunjukan kelembapan didalam greenhouse

3.4 Tahapan Penelitian

Pengerjaan dalam penelitian ini dilakukan secara bertahap mulai dari studi literatur lalu dilanjutkan dengan tahapan perancangan sistem menggunakan metode *Kanban* lalu di akhiri dengan tahapan Analisa hasil. Berikut ini adalah diagram alir dari tahapan yang dilakukan.



Gambar 3.2 Diagram alir tahapan penelitian

3.5 Analisa Kebutuhan

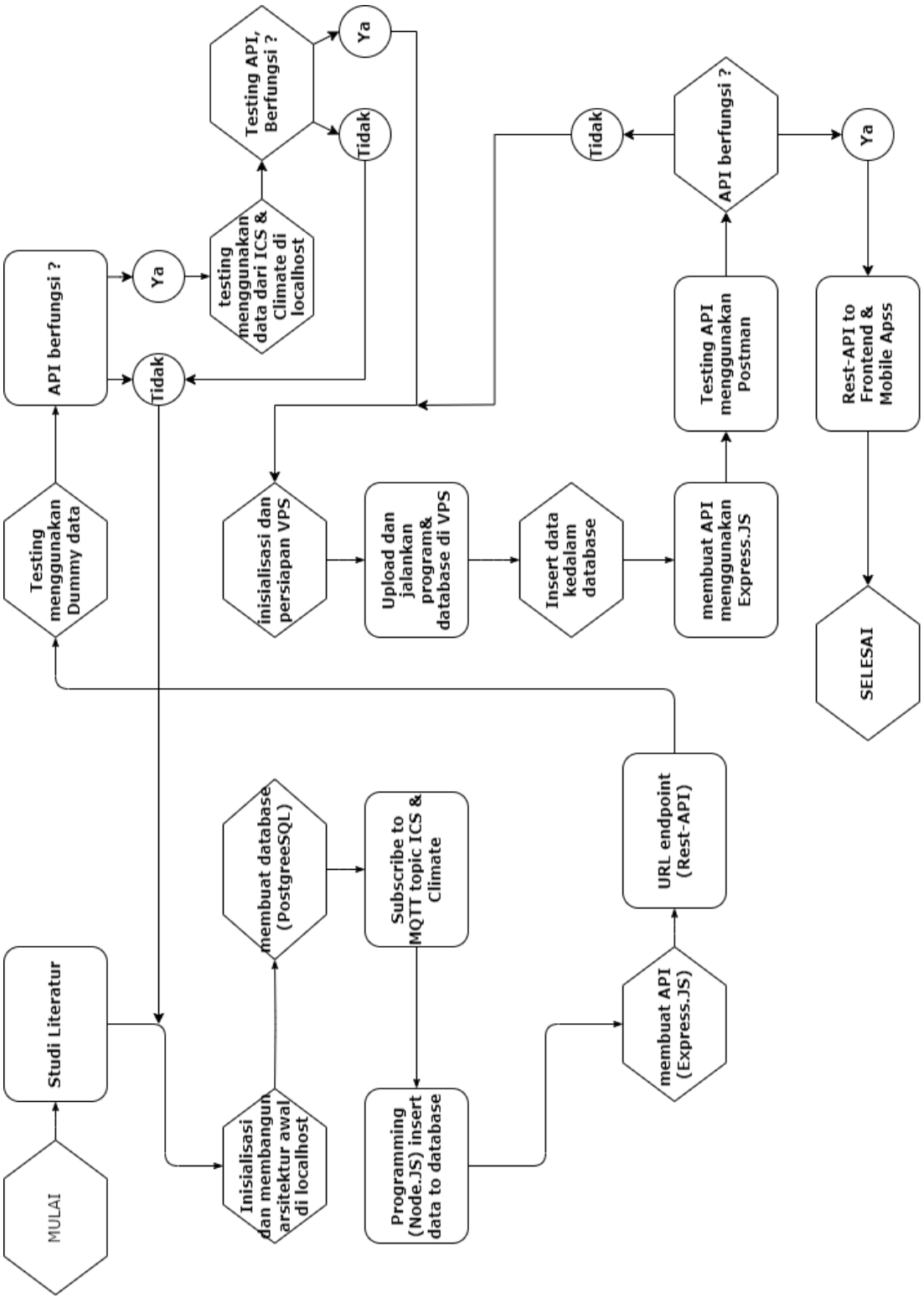
Penentuan kebutuhan didapatkan setelah mendapatkan beberapa data berupa literasi yang berkaitan dengan pengembangan dan implementasi *backend dashboard iot* pada *inteliggence control system (ics) smart greenhouse* menggunakan protokol *mqtt* yang selanjutnya akan dicari informasi yang berhubungan dengan aktivitas yang akan dilakukan dalam pengembangan sistem.

Berdasarkan studi pustaka dan melihat kondisi ICS pada *smart greenhouse* ini telah berjalan, maka dapat ditentukan apa saja yang dibutuhkan dalam penelitian ini untuk mengatasi beberapa kelemahan yang ada pada sistem yang saat ini sedang berjalan.

Adapun setelah dilakukannya identifikasi terhadap tujuan pengembangan aplikasi dan kebutuhan, didapat beberapa kebutuhan yang dapat diselesaikan dengan penenilian ini antara lain :

1. *Virtual Private Server* yang dapat berfungsi sebagai *MQTT Broker* dan penyimpanan data.
2. *MQTT Broker* yang dapat digunakan sebagai protokol pengiriman data antara sensor pemantauan dengan dashboard dan *database*.
3. Data yang didapat dari sensor harus dapat disimpan dalam *database* VPS dan tidak boleh hilang dikarenakan data tersebut dapat digunakan sebagai data yang akan digunakan dalam dashboard dan sebagai bahan penelitian lebih lanjut.
4. Data yang tersimpan di *database* bisa dikirimkan ke *dashboard* atau *front-end* menggunakan *API.*

Berdasarkan beberapa kebutuhan tersebut, maka didapatkan beberapa *task* yang harus dilakukan. *Task* tersebut digambarkan pada Gambar 3.3.



Gambar 3.3 *Task to do* pada penelitian yang dilakukan

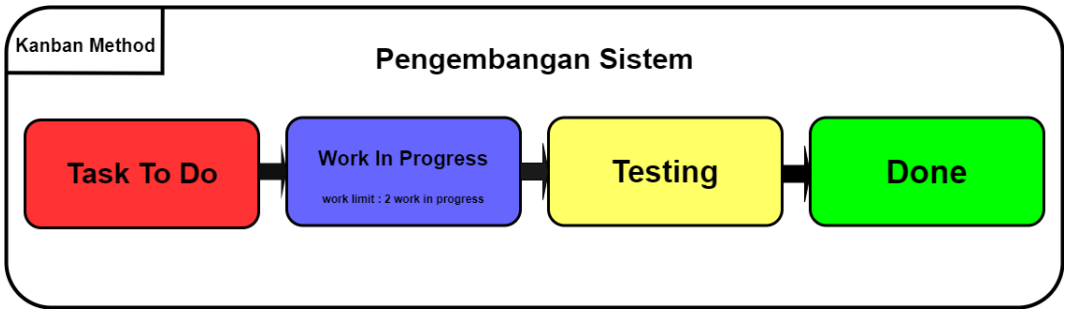
Proses pengembangan sistem dimulai dengan Studi Literatur untuk memahami dasar teori dan praktek terbaik yang relevan. Setelah memperoleh pengetahuan yang cukup, dilanjutkan dengan Inisialisasi dan Membangun Arsitektur Awal di *Localhost* yang meliputi pembuatan basis data menggunakan *PostgreSQL* dan pengaturan lingkungan pengembangan. Selanjutnya, dilakukan *Programming* dengan *Node.js* untuk *menginsert* data ke dalam *database*. Paralel dengan ini, sistem juga disiapkan untuk *Subscribe* to MQTT *topic* ICS & Climate, yang memungkinkan sistem untuk menerima data dari topik MQTT yang telah ditentukan.

Setelah arsitektur lokal siap, proses berlanjut ke Inisialisasi dan Persiapan (*Virtual Private Server*) VPS di mana program dan *database* di-*upload* dan dijalankan di VPS. Data kemudian diinsert ke dalam *database* di VPS. Sistem kemudian memasuki fase *Testing* menggunakan *Dummy* data untuk memastikan bahwa semua komponen berfungsi dengan benar. Jika API tidak berfungsi, maka perlu dilakukan pengecekan lagi terhadap tahapan yang dilakukan sebelumnya, jika berhasil selanjutnya dilakukan *testing* API menggunakan data dari ICS & *Climate* di *Localhost*. Setelah API terbukti berfungsi, dapat dilakukan pengujian lebih lanjut menggunakan *Postman* untuk memastikan bahwa API berfungsi sebagaimana mestinya.

Dengan API yang telah diverifikasi, tahap selanjutnya adalah mengembangkan URL *endpoint* menggunakan Rest-API yang akan digunakan oleh frontend dan aplikasi mobile. Setelah semua pengujian berhasil dan sistem stabil, proses pengembangan dianggap selesai, dan sistem siap digunakan dalam produksi. Seluruh proses ini dirancang untuk memastikan bahwa sistem yang dikembangkan tidak hanya memenuhi kebutuhan pengguna tetapi juga kuat, aman, dan skalabel.

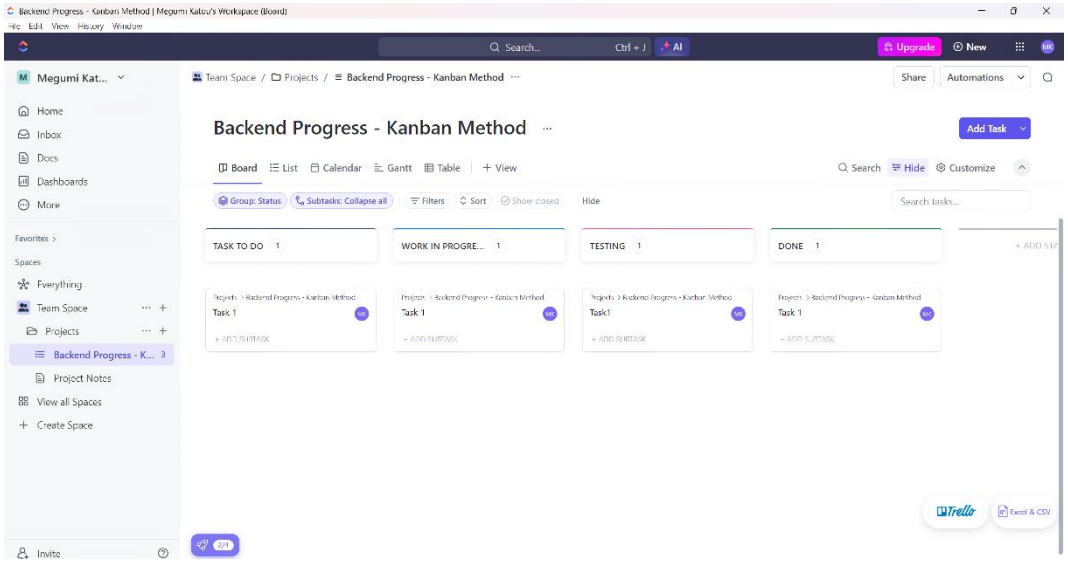
3.6 Pengembangan Sistem

Dalam penelitian ini, metode yang digunakan dalam pengembangan sistem akan mengikuti model *Kanban Board*. Berikut ini adalah tahapan yang akan dilaksanakan dalam pengembangan menggunakan metode *Kanban*.



Gambar 3.4 Tahapan pengembangan sistem dengan Metode *Kanban*

Berdasarkan Gambar 3.4 dapat dilihat bahwa pada penelitian yang dilakukan memiliki beberapa tahapan sesuai dengan metode *Kanban* yang digunakan. Tahapan pada penelitian ini juga divisualisasikan menggunakan software ClickUp seperti pada Gambar 3.5.



Gambar 3.5 visualisasi tahap pengembangan menggunakan software ClickUp

3.6.1 Tahap *Task To Do*

Pada tahap *task to do, task* tugas yang akan diklasifikasikan dengan skala tingkat prioritas. Pada tahap ini terdapat 3 skala prioritas:

1. Prioritas Tinggi

   Pada tugas yang prioritas tinggi, menunjukan tugas tersebut berprioritas tinggi untuk dikerjakan.Prioritas tinggi menandakan bahwa tugas tersebut adalah

tugas yang penting dan krusial untuk membuat sistem dapat berjalan dengan baik. Apabila tugas tersebut tidak dilakukan, maka akan mengganggu jalan nya sistem atau sistem bisa tidak berjalan sama sekali.

2. Prioritas Sedang

   Tugas dengan prioritas menengah menandakan bahwa tugas tersebut adalah tugas yang penting untuk dikerjakan namun tugas tersebut tidak akan mengganggu kinerja sistem apabila belum dikerjakan.

3. Prioritas Rendah

   Prioritas rendah dalam pengembangan sistem ini menandakan bahwa tugas tersebut merupakan tugas opsional yang apabila belum dikerjakan tidak akan mengganggu kinerja sistem

Penentuan prioritas dari setiap tugas yang dilakukan adalah sebagai berikut :

Tabel 3.6.1.1. Penentuan prioritas dari tugas yang dilakukan.

No	Tugas *(Task)* yang dilakukan	Prioritas	Alasan
1.	Desain arsitektur sistem	Sedang	Desain arsitektur adalah tugas yang penting untuk membantu mengidentifikasi fungsi – fungsi pada sistem yang akan dibangun, namun tidak berpengaruh secara langsung terhadap kinerja sistem
2.	Inisialisai dan membangun arsitektur awal di local	Sedang	Membangun arsitektur diawal dengan melakukanya pada server local dapat menghindari hal yang tidak diinginkan apabila terjadi error, dimana untuk penyelesaianya akan lebih mudah di local, juga sebagai tahapan perencanaan

No	Tugas *(Task)* yang dilakukan	Prioritas	Alasan
3.	Persiapan dan konfigurasi MQTT *Broker* di VPS	Tinggi	Konfigurasi MQTT pada *server* harus dilakukan dikarenakan apabila MQTT tidak dikonfigurasi telebih dahulu, MQTT tidak akan dapat digunakan.
4.	Tahap testing menggunakan dummy data	Rendah	Proses ini dilakukan ketika belum bisa memperoleh data dari sensor, sehingga diperlukan testing menggunakan data dummy untuk checking apakah program sudah dapat berjalan dengan baik
5.	Konfigurasi *Esp* untuk mengirimkan data sensor	Tinggi	Konfigurasi *Esp* dilakukan untuk mengambil data sensor secara *realtime*. Dalam hal ini tugas ini sangat penting agar backend dapat menerima data dari sensor
6.	Inisialisasi dan persiapan Virtual Private *Server* (VPS)	Tinggi	Tugas ini berfungsi agar program yang sudah dirancang pada local, dapat di deploy pada server, tugas ini sangat penting karena backend membutuhkan data yang realtime dan selalu dalam keadaan ON
7.	Membuat *database* untuk menyimpan data dari *Esp*	Tinggi	Data yang telah diterima dari raspberry harus disimpan kedalam *database server*
8.	Menerima data dari *Esp* dengan protokol MQTT	Tinggi	Data sensor yang dikrimkan akan diterima dengan metode subscribe ke dalam topic MQTT

No	Tugas *(Task)* yang dilakukan	Prioritas	Alasan
9.	Pengkodean untuk penyimpanan data sensor dari *Esp* ke *Database* menggunakan node.js	Tinggi	Data yang telah diterima akan disimpan kedalam database PostgreSQL agar dapat diakses dan datanya tidak hilang
10.	Mengambil data pada database dan membuat API MQTT untuk mengirim data ke *front-end* menggunakan node.js	Tinggi	Tugas ini berfungsi agar *front-end* dapat mengambil *record* data dari *database*

3.6.2 Tahap *Work in Progress*

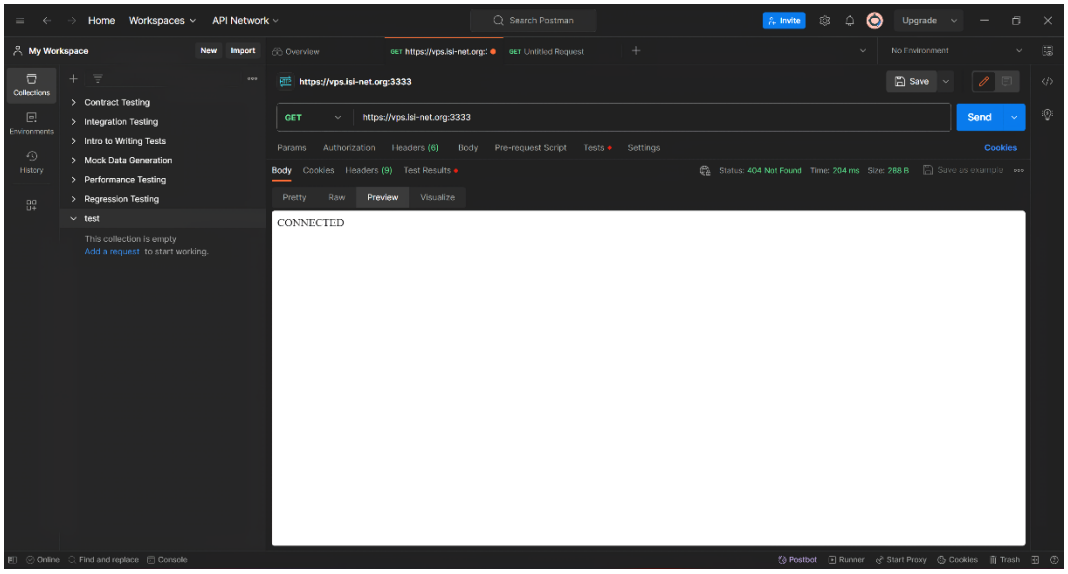
Bagian ini merupakan bagian dari *Task to Do yang sedang dikerjakan* akan dipindahkan ke bagian ini. Sesuai dengan prinsip dari *Kanban* yaitu membatasi tugas yang sedang berlangsung, maka tugas yang diletakan pada bagian ini akan dibatasi jumlah tugasnya pada setiap *flow*nya. Pembatasan tugas pada tahap ini berfungsi untuk menghindari terjadinya *overproduction* atau pemborosan dari pengerjaan banyak tugas sekaligus, sehingga fokus dapat lebih terarahkan pada menyelesaikan pekerjaan yang sedang dalam proses sebelum memulai pekerjaan baru.

Dalam penelitian ini ditentukan batasan dari tahapan ini adalah 2 buah *task* yang dapat dikerjakan secara bersamaan. Hal ini dikarenakan dalam pengembangan sistem ini dilakukan secara individu, sehingga membatasi dua buah pekerjaan sudah cukup efektif dalam penyelesaiannya.

3.6.3 Tahap *Testing*

Tugas yang telah selesai dari bagian sebelumnya akan dipindah ke bagian *testing*. Dalam tahap ini tugas yang selesai akan diuji coba dan evaluasi untuk menguji apakah tugas tersebut dapat terselesaikan dengan baik atau tidak. Apabila dalam evaluasi terdapat hal yang perlu dilakukan perubahan, maka tugas tersebut akan

dikembalikan ke tahap sebelumnya sesuai dengan tingkat prioritasnya.



Gambar 3.6 testing API

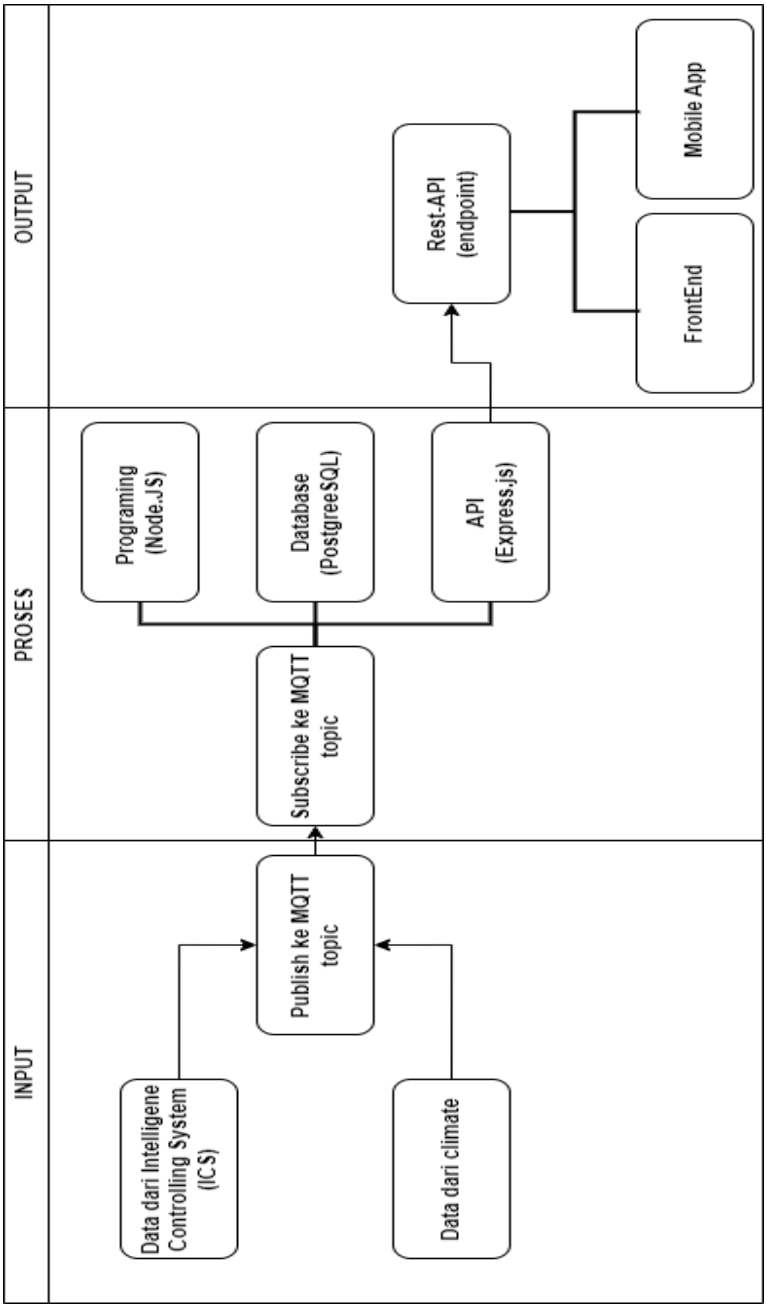
Pada tahap ini peneliti melakukan testing terhadap program yang telah dibuat dengan mengecek apakah ada *error* saat program dijalankan, dan mengecek API yang akan diberikan pada *front-end* apakah sudah dapat menerima data dan mengirimkanya dengan baik. Pada tahap ini peneliti menggunakan *software* postman. *Postman* adalah platform pengembangan API yang populer yang memungkinkan pengembang untuk merancang, menguji, mendokumentasikan, memantau, dan mempublikasikan API mereka dengan mudah. Aplikasi ini menyediakan antarmuka pengguna yang ramah untuk mempermudah interaksi dengan API tanpa perlu menulis kode. *Postman* terdapat fungsi *GET, PUT, POST* dan *DELETE* untuk menguji sebuah *endpoint* pada *backend* tersebut dapat berjalan sebagaimana yang diharapkan .

3.6.4 Tahap *Done*

Tugas yang telah selesai dilakukan pengujian dan dapat berfungsi dengan baik maka akan dipindahkan ke bagian *done*. Tugas yang dipindahkan ke bagian ini telah dianggap selesai. Ketika tugas sudah berada pada tahap ini maka tugas bisa dikatakan selesai dan siap untuk digunakan.

3.7 Analisa Hasil

Tahap analisis dilakukan untuk pengujian mengenai hasil pengembangan dan implementasi *backend dashboard iot* pada *inteliggence control system (ics) smart greenhouse* menggunakan protokol *MQTT* yang dilakukan setelah proses pengembangan telah selesai dan data hasil pengujian telah terkumpul.



Gambar 3.7 Input dan Output pada penelitian yang dilakukan

Seperti yang diilustrasikan pada Gambar 3.7, analisa hasil didasarkan pada tahapan input yang diterima dan proses yang dikerjakan sehingga menghasilkan sebuah output yang diharapkan pada pengembangan sistem ini. Pada tahap input, sistem menerima data dari *Intelligent Controlling System* (ICS) dan *climate* yang secara *real-time mempublish* data tersebut ke topik MQTT. Ini memungkinkan data untuk dikomunikasikan melalui protokol MQTT yang efisien dan cepat. Setelah data dipublish, *backend* kemudian *subscribe* ke topik MQTT yang sama untuk mengambil data yang dikirim.

Dengan *Node.js*, data yang diterima diproses dan kemudian disimpan dalam *database PostgreSQL*, yang merupakan basis data yang dapat menangani transaksi data besar dengan keamanan yang tinggi. Setelah data tersimpan, sistem menggunakan *Express.js*, sebuah *framework* populer dalam ekosistem *Node.js*, untuk membangun API yang melayani sebagai jembatan antara *database* dan aplikasi pengguna.

Output dari proses ini adalah sebuah *Rest-API* yang menyediakan *endpoint* untuk *frontend* dan aplikasi *mobile*. *Endpoint* ini memungkinkan sistem *frontend* untuk mengambil data yang dibutuhkan dan aplikasi *mobile* untuk mengintegrasikan data tersebut ke dalam pengalaman pengguna *mobile* yang interaktif. Dengan cara ini, data yang dikumpulkan dan diproses oleh sistem *backend* dengan aman dan efisien tersedia untuk aplikasi klien, memastikan bahwa pengguna akhir dapat mengakses informasi terbaru dengan cepat dan mudah.

DAFTAR PUSTAKA

[1] D. Kurniawati and S. Rahayu, "Tantangan dan Strategi Petani Kecil di Indonesia Dalam Menghadapi Ketidakpastian Cuaca," *Jurnal Agribisnis Indonesia*, vol. 10, no. 1, pp. 73-85, 2022.

[2] S. Aminah and J. Sujono, "Dinamika Pemanfaatan Lahan dan Dampaknya pada Lahan Pertanian di Indonesia," *Jurnal Agroekonomi*, vol. 37, no. 2, pp. 175-188, 2019.

[3] S. Few, *Information Dashboard Design: Displaying Data for At-a-Glance Monitoring*. Analytics Press, 2013.

[4] M. Farid Ali Safii, S. Raharjo, and U. Lestari, "Analisis Quality of Service Protokol MQTT Dan HTTP Pada Penerapan Sistem Monitoring Suhu Berbasis Nodemcu (Studi Kasus Ruang Server Kampus 3 IST Akprind Yogyakarta)," *J. JARKOM*, vol. 7, no. 1, pp. 11–19, 2019.

[5] N. A-hussein and A. D. Salman, "IoT Monitoring System Based on MQTT Publisher/Subscriber Protocol," *Iraqi J. Comput. Commun. Control Syst. Eng.*, no. April, pp. 75–83, 2020, doi: 10.33103/uot.ijccce.20.3.7.

[6] D. Hermanto, A. N. Salim, and I. P. Putra, "Sistem Monitoring Suhu Dan Kelembapan Udara Menggunakan Protokol Mqtt Berbasis Wemos D1 Mini," *AVoER (Applicable Innov. Eng. Sci. Res.*, pp. 27–28, 2021

[7] H. P. Safitri, R.K., dan Putro, "Implementasi REST API untuk Komunikasi Antara ReactJS dan NodeJS (Studi Kasus : Modul Manajemen User Solusi247)," *Automata*, vol. 2, no. 1, pp. 0–4, 2021,

[8] D. R. Agustina, A. Y. Vandika, W. Susanty, T. Tanjung, and R. N. Afiani, "Implementasi Service Data untuk Pemantauan Lighting pada Smart Agriculture," *Digit. Transform. Technol.*, vol. 3, no. 2, pp. 380–388, 2023.

[9] M Nur, Hasanuddin. "Membangun Protokol MQTT Broker Pada Virtual Private Server (VPS) Sebagai Integrasi Sistem Pemantauan dan Peringatan Dini Bencana Tsunami Berbasis IoT." 2023

[10] M. S. Tahir and A. Rahim, "The Role of Smart Greenhouses in Precision Agriculture: A Review," in *Proc. Int. Conf. on Advanced Technologies for Agricultural and Environmental Engineering (ATAGEE)*, 2021, pp. 134-139.

[11] S. Mulyono, M. Taufik, dan M. Taufiqurrohman, "Sistem IoT Terintegrasi Menggunakan Flow Based Programming dengan Protokol MQTT dan Time Series DB," *Jurnal Transistor Elektro dan Informatika* , vol. 3, no. 1, hlm. 9–20, 2018.

[12] MQTT.org, "MQTT (The Standard for IoT Messaging)." https://mqtt.org/ (diakses 20 Desember 2023).

[13] Eclipse, “Eclipse Mosquitto.” https://mosquitto.org/ (diakses 19 Februari 2023).

[14] R. P. Eka, A. Rachman, dan T. H. Wahyu, “Virtual Private Server (VPS) Sebagai Alternatif Pengganti Dedicated Server,” *11th Seminar on Intelligent Technology and Its Applications, SITIA*, 2010, [Daring]. Tersedia pada: http://www.apnicservices.com/

[15] C. Y. Andika dan S. Rudiarto, “Rancang Bangun Aplikasi Sosial Media Crawler Menggunakan Nodejs Menerapkan Konsep Non-Blocking I/O,” *Jurnal Ilmiah FIFO*, vol. IX, no. 2, 2017.

[16] A. Hidayatullah and B. Purwanto, "Pembangunan Web Service menggunakan Framework Express.js pada Node.js," *Jurnal Teknik Informatika*, vol. 12, no. 1, pp. 34-39, 2020.

[17] PostgreSQL, “PostgreSQL.” https://www.postgresql.org/about/ (diakses 20 Desember 2023).

[18] M. F. A. Muri, H. S. Utomo, dan R. Sayyidati, “Search Engine Get Application Programming Interface,” *Jurnal Sains dan Informatika*, vol. 5, no. 2, hlm. 88–97, Des 2019, doi: 10.34128/jsi.v5i2.175.

[19] E. Brechner dan J. Waletzky, *Agile Project Management with Kanban*. Redmont, Washington: Microsoft Press, 2015.

[20] N. Faizah, N. Santoso, dan A. A. Soebroto, “Pengembangan Sistem Aplikasi Manajemen Proyek menggunakan Kanban Framework,” *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer* , vol. 3, no. 10, hlm. 9747–9754, 2019, [Daring]. Tersedia pada: http://j-ptiik.ub.ac.id

[21] S. Mulyono, M. Qomaruddin, and M. Syaiful Anwar, “Penggunaan Node-RED pada Sistem Monitoring dan Kontrol Green House berbasis Protokol MQTT,” *J. Transistor Elektro dan Inform. (TRANSISTOR EI*, vol. 3, no. 1, pp. 31–44, 2018.

[22] S. Mulyono dan S. F. C. Haviana, “Implementasi MQTT untuk Pemantauan Suhu dan Kelembaban pada Laboratorium,” *Jurnal Transistor Elektro dan Informatika (TRANSISTOR EI*, vol. 3, no. 3, hlm. 140–144, 2018.

[23] A. Hidayat dan D. Prabowo, “Implementation of Virtual Private Server (VPS) Using Digital Ocean Cloud Server on BMT. Mentari East Lampung,” *Jurnal JTKSI*, vol. 03, no. 03, hlm. 116–121, 2020.

[24] A. Nurrahman dan N. A. Z, “Sistem Kontrol Suhu dengan PID dan Monitoring Daya Output Pada Panel Photovoltaic Portable dengan Virtual Private Server,” *Jurnal Maestro*, vol. 3, no. 2, 2020.

[25] I. Kurniawan, Humairah, dan F. Rozi, “REST API Menggunakan NodeJS pada Aplikasi Transaksi Jasa Elektronik Berbasis Android,” *Jurnal Ilmiah Teknologi*

Sistem Informasi, vol. 1, no. 4, hlm. 127–132, 2020, [Daring]. Tersedia pada: http://jurnal-itsi.org

[26] A. Hidayat dan D. Prabowo, "Implementation of Virtual Private Server (VPS) Using Digital Ocean Cloud Server on BMT. Mentari East Lampung," *Jurnal JTKSI*, vol. 03, no. 03, hlm. 116–121, 2020.

[27] I. Wahyudi and A. Syazili, "Dashboard Monitoring Website Dosen Studi Kasus Universitas Bina Darma," *J. Pengemb. Sist. Inf. dan Inform.*, vol. 2, no. 3, pp. 188–197, 2021, doi: 10.47747/jpsii.v2i3.555.

[28] C. Seviro Bima Sakti and I. Hermawan, "Implementasi Arsitektur Microservice pada Back End Sistem Informasi Atlantas berbasis Website," *J. Teknol. Terpadu*, vol. 6, no. 2, pp. 96–104, 2020, doi: 10.54914/jtt.v6i2.281.

[29] I. Albanna and H. Nugroho, "Fungsi Ganda Perangkat IoT-Client sebagai Kendali Aktuator dan Basis Layanan Melalui Komunikasi Paket Data JSON," pp. 119–129, 2023, doi: 10.31284/j.jtm.2023.v4i2.4699.

[30] A. A. Sahifa, R. Setiawan, and M. Yazid, "Pengiriman Data Berbasis Internet of Things untuk Monitoring Sistem Hemodialisis Secara Jarak Jauh," *J. Tek. ITS*, vol. 9, no. 2, pp. 4–9, 2021, doi: 10.12962/j23373539.v9i2.55650.

[31] A. Ambarwari, Dewi Kania Widyawati, and Anung Wahyudi, "Sistem Pemantau Kondisi Lingkungan Pertanian Tanaman Pangan dengan NodeMCU ESP8266 dan Raspberry Pi Berbasis IoT," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 5, no. 3, pp. 496–503, 2021, doi: 10.29207/resti.v5i3.3037.

[32] Y. Loui Pattinama, FERDIANSYAH, I. Susanti, and Painem, "Implementasi Rest API Web Service Dengan Otentifikasi JSON Web Token Untuk Aplikasi Properti," *Inform. J. Ilmu Komput.*, vol. 19, no. 1, pp. 81–89, 2023, doi: 10.52958/iftk.v19i1.5724.

[33] M. Saiqul Umam, S. Adi Wibowo, and Y. Agus Pranoto, "Implementasi Protokol Mqtt Pada Aplikasi Smart Garden Berbasis Iot (Internet of Things)," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 7, no. 1, pp. 899–906, 2023, doi: 10.36040/jati.v7i1.6131.

[35] K. Hwang, J. M. Lee, and I. H. Jung, "Performance Monitoring of MQTT-based Messaging Server and System," *J. Logist. Informatics Serv. Sci.*, vol. 9, no. 1, pp. 85–96, 2022, doi: 10.33168/LISS.2022.0107.

Plagiarism Checker X Originality Report

Similarity Found: 14%

Date: Wednesday, January 31, 2024
Statistics: 689 words Plagiarized / 4813 Total words
Remarks: Low Plagiarism Detected - Your Document needs Optional Improvement.

PENGEANGAN D IMPLENTASI BCE & REST - API DASOAR IOT PADA INTELLIGENCE CONTROL STEM (IC) SR GREE ME POL MQT (Prop) Ol M . Af Sdie Asmara 20150310 48 JURUSAN TKNIK EERO FAKULTAS TKNIK UNIVERS LUNG 2023 PENGEANGAN D IMPLENTASI BCE & REST - API DASOAR IOT PADA INTELLIGENCE CONTROL STEM (IC) SR GREE ME POL MQT Oleh M.

Affan Siddiqie Asmara Proposal Penelitian Sebagai Salah Satu Syarat untuk Mencapai Gelar SARJANA TEKNIK Pada Jurusan Teknik Elektro Fakultas Teknik Universitas Lampung FAKULTAS TEKNIK UNIVERSITAS LAMPUNG BANDAR LAMPUNG 2023 iii DAFTAR IS Hal DAFTAR ISI ..... iii DAFTAR TABEL ..... v DAFTAR GAMBAR .....

vi BAB I PENDAHULUAN ..... 1 1.1. Latar belakang ..... 1 1.2. Rumusan Masalah ..... 2 1.3. Batasan Masalah ..... 3 1.4. Tujuan Penelitian ..... 3 1.5. Manfaat Penelitian .....