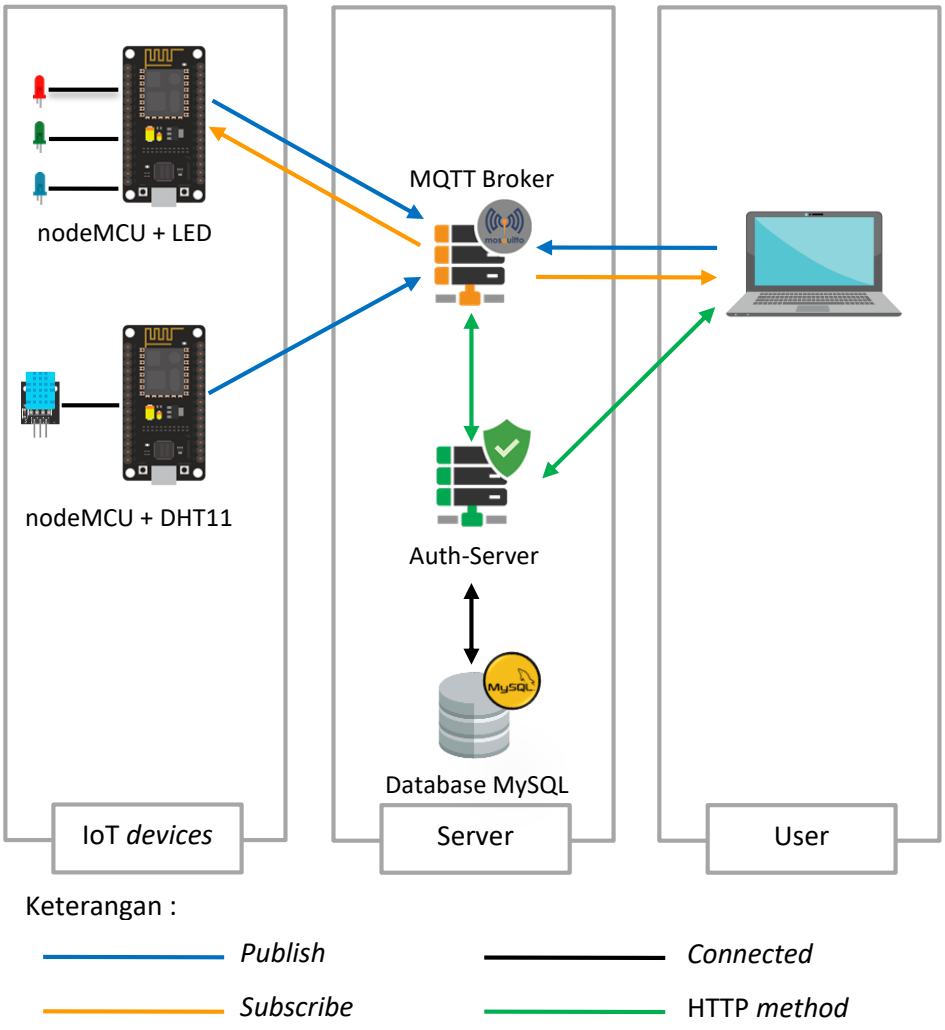
BAB 4 REKAYASA KEBUTUHAN DAN PERANCANGAN

4.1 Rekayasa Kebutuhan Sistem

Kebutuhan sistem merupakan proses untuk mengetahui hal-hal yang akan dibutuhkan untuk merancang sebuah sistem serta implementasinya. Kebutuhan sistem didefinisikan berdasarkan deskripsi umum dari sistem dan rumusan masalah. Daftar kebutuhan ini bertujuan untuk menyelesaikan permasalahan yang dirumuskan dengan menggunakan sistem yang akan dibuat. Kebutuhan sistem akan dinyatakan melalui daftar kebutuhan fungsional dan kebutuhan non-fungsional.

4.1.1 Deskripsi Umum Sistem



Gambar 4.1 Gambaran Arsitektur Umum Sistem

Sistem akan dirancang dengan menerapkan mekanisme pengamanan melalui autentikasi dan otorisasi, dan akan dibagi menjadi dua komponen server. Komponen pertama akan bertindak sebagai broker yang akan mengatur semua kegiatan pada protokol MQTT, dari koneksi yang akan dilakukan klien, transaksi

publish-subscribe pesan, diskoneksi klien. Komponen kedua akan bertidak sebagai server yang akan melakukan kegiatan pengecekan identitas klien dan hak akses klien tersebut dengan melihat database sistem terkait informasi klien tersebut yang akan bertransaksi menggunakan HTTP *method*. Klien yang terdapat dalam sistem dibagi menjadi beberapa peran dan peran tersebut akan bertindak sebagai penentu hak akses yang dimiliki oleh klien tersebut.

Tujuan utama penerapan mekanisme autentikasi dan otorisasi adalah untuk memberikan mekanisme pengamanan dari pihak-pihak yang tidak memiliki hak untuk mengakses sistem serta informasi yang ada dalam sistem MQTT. Autentikasi pada MQTT akan mengetahui identitas dari klien yang ada dalam sistem dan otorisasi akan mengatur kegiatan klien tersebut di dalam sistem berdasarkan pada ACL. Token JWT akan digunakan sebagai media untuk melakukan autentikasi yang menggunakan token berisi *username* dan *password* yang kemudian akan dicek dalam database, sedangkan ACL akan digunakan untuk melihat hak yang dimiliki klien tersebut terhadap topik dan jenis akses (*publish*/*subscribe*) berdasarkan peran yang dimiliki oleh klien tersebut dalam database. Kedua mekanisme ini akan dilakukan setiap ada kegiatan yang dilakukan oleh klien dalam sistem untuk memenuhi mekanisme pengamanan sistem.

Arsitektur sistem secara umum akan digambarkan pada gambar 4.1, dimana arsitektur sistem akan memiliki 3 batasan yaitu pada IoT *devices*, server dan *user*. IoT *devices* merupakan kumpulan perangkat nodeMCU yang bertindak sebagai klien pada MQTT yang akan berperan sebagai mikrokontroler untuk sensor dan aktuator, sensor yang akan digunakan adalah sensor suhu dan kelembaban DHT11 dan aktuator akan menggunakan 3 buah LED yang berbeda warna. NodeMCU yang terhubung dengan DHT11 akan *publish* data suhu dan kelembaban sedangkan nodeMCU yang terhubung ke LED akan *subscribe* pada topik yang akan menyalakan/mematikan LED tersebut dan akan *publish* status keadaan LED yang berubah keadaannya. Server akan terdiri dari broker mosquitto yang menangani komunikasi data melalui protokol MQTT dan Auth-server yang digunakan untuk memeriksa database setiap terjadi mekanisme autentikasi dan otorisasi. User merupakan klien MQTT yang dapat melakukan *publish*/*subscribe* ke broker mosquitto melalui protokol MQTT dan melakukan fungsi CRUD (Create, Read, Update dan Delete) database menggunakan HTTP method pada Auth-server. User sebagai klien akan memiliki peran peran tertentu dan peran tersebut akan bertindak sebagai penentu hak akses yang dimiliki oleh klien tersebut.

4.1.2 Kebutuhan Fungsional

Kebutuhan fungsional mendeskripsikan kemampuan dan layanan yang dapat dilakukan oleh sistem yang akan dibangun, serta bagaimana sistem akan bereaksi terhadap aksi yang dilakukan oleh pengguna. Kebutuhan fungsional dibuat berdasarkan pada deskripsi dan arsitektur umum dari sistem yang akan dirancang dan diimplementasi. Kebutuhan fungsional dalam penelitian ini didaftarkan kedalam tabel 4.1 sebagai berikut:

Tabel 4.1 Kebutuhan Fungsional

No.	Kebutuhan Fungsional
1.	Broker & auth-server dapat saling berkomunikasi melalui HTTP method.
2.	Broker dapat menerima koneksi dari klien MQTT dan melakukan mekanisme autentikasi ke auth-server.
3.	Broker dapat menerima dan meneruskan pesan *publish* dan menerima *subscribe* klien MQTT & melakukan mekanisme otorisasi ke auth-server.
4.	Auth-server dapat melakukan mekanisme autentikasi dengan memeriksa identitas klien MQTT pada database.
5.	Auth-server dapat melakukan mekanisme otorisasi dengan memeriksa ACL terkait hak akses klien MQTT pada database.
6.	Klien MQTT dapat membangun koneksi dengan broker menggunakan token JWT untuk menggantikan username dan password.
7.	Klien MQTT dapat *publish* atau *subscribe* ke topik tertentu ke broker.
8.	NodeMCU dapat membaca data sensor DHT dan menyalakan LED.
9.	User dengan peran tertentu dapat melakukan CRUD (*Create*, *Read*, *Update* dan *Delete*) peran, user dan *permissions* pada database melalui auth-server.

4.1.3 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional mendeskripsikan tingkatan dari kualitas, batasan layanan atau fungsi yang ditawarkan oleh sistem seperti batasan waktu, batasan pengembangan proses, standarisasi dll. Beberapa batasan dalam melakukan perancangan dan implementasi dibuat agar sistem dapat difokuskan untuk memenuhi permasalahan yang telah dirumuskan. Batasan ini dibuat berdasarkan batasan masalah yang telah dirumuskan dan dibuat berdasarkan kebutuhan sistem yang akan dirancang, beberapa batasan tersebut meliputi:

Tabel 4.2 Kebutuhan Non-Fungsional

No.	Kebutuhan Non-Fungsional
1.	Broker yang digunakan adalah Mosquitto Broker.
2.	Mekanisme autentikasi dan otorisasi menggunakan mosquitto auth-plug.
3.	Sertifikat dan *Key* untuk TLS/SSL menggunakan OpenSSL.
4.	Auth-server backend akan diimplementasikan menggunakan komunikasi HTTP-*method* yang dijalankan melalui python dengan *framework* bottle.
5.	Autentikasi dilakukan dengan menggunakan token JWT yang di-*generate* secara sederhana pada auth-server.

6.	Payload JWT akan berisikan *username* dan *password hash*, token didesain secara sederhana tanpa *expiration*.
7.	Data yang dibutuhkan oleh ACL, seperti *username*, *role* dan *password*, daftar topik dan perizinan, akan disimpan kedalam database menggunakan MySQL.
8.	Pemrograman nodeMCU akan dilakukan melalui arduino IDE dan akan menggunakan beberapa *library* pendukung bagi MQTT, WiFi dan sensor.

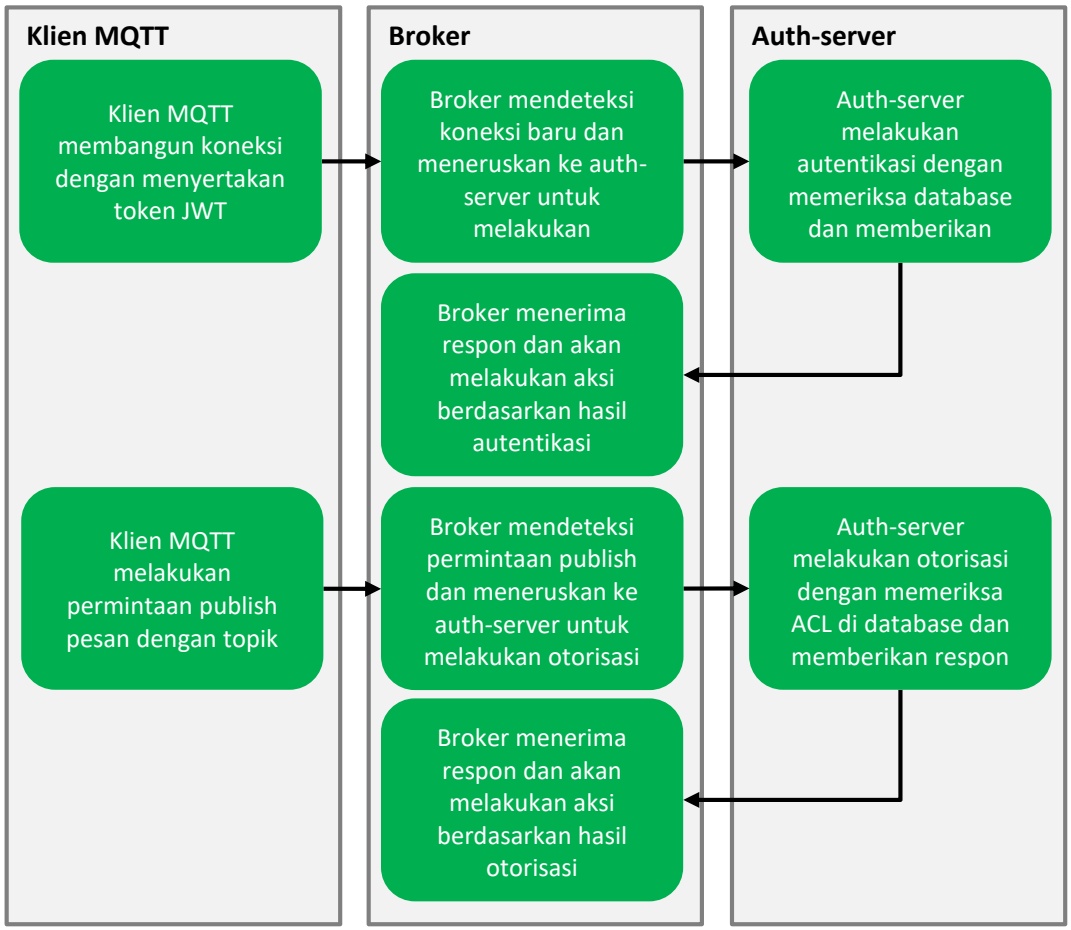
4.2 Perancangan Sistem

Proses perancangan sistem dibagi menjadi beberapa tahap terkait tiap komponen yang akan diimplementasikan, perancangan sistem dilakukan untuk mempermudah proses implementasi karena setiap komponen akan dirancang secara sistematis dari arsitektur, mekanisme komunikasi dan alur kerja dari komponen tersebut. Perancangan sistem akan dibuat berdasarkan pada rekayasa kebutuhan yang telah didefinisikan, baik dari deskripsi umum sistem, arsitektur umum sistem, kebutuhan fungsional hingga kebutuhan non-fungsional. Perancangan sistem akan dimulai dengan membuat alur kerja penanganan *publisher* dan *subscriber*, perancangan broker, perancangan auth-server, perancangan skema database, perancangan topik dan *access control list* (ACL), perancangan klien MQTT dan perancangan nodeMCU.

4.2.1 Perancangan *Publisher*

Pada gambar 4.2, dijelaskan mengenai alur kerja dari sistem yang akan dibangun dalam kasus penanganan *publisher*. Penanganan *publisher* akan dimulai oleh klien MQTT (dapat berupa pengguna / nodeMCU) sebelum dapat melakukan *publish* pesan diharuskan untuk membangun koneksi dengan broker MQTT terlebih dahulu, untuk membangun koneksi klien akan menyertakan sebuah token JWT yang berisi *username* dan *password*. Kemudian broker akan mendeteksi bahwa terdapat permintaan koneksi dari klien MQTT untuk kemudian diteruskan menuju auth-server untuk dilakukan mekanisme autentikasi. Auth-server kemudian menerima permintaan dari broker, kemudian auth-server akan men-*decode* token JWT dan menggunakan *username* dan *password* yang terdapat didalamnya untuk melakukan mekanisme autentikasi dengan memeriksa identitas tersebut dalam database. Hasil dari pemeriksaan identitas kemudian dikirimkan kembali ke broker, kemudian broker akan menentukan aksi selanjutnya berdasarkan pada hasil autentikasi. Jika hasil autentikasi berhasil dan identitas klien MQTT terdapat dalam database maka broker akan memelihara koneksi hingga koneksi terputus, namun jika identitas klien MQTT tidak terdapat dalam database maka broker akan menolak permintaan koneksi. Selanjutnya klien MQTT akan mengirimkan permintaan *publish* pesan dengan topik ke broker setelah koneksi berhasil, broker yang mendeteksi bahwa terdapat permintaan *publish* akan meneruskan ke auth-server untuk dilakukan mekanisme otorisasi. Auth-server akan menerima permintaan dari broker dan memeriksa ACL pada database

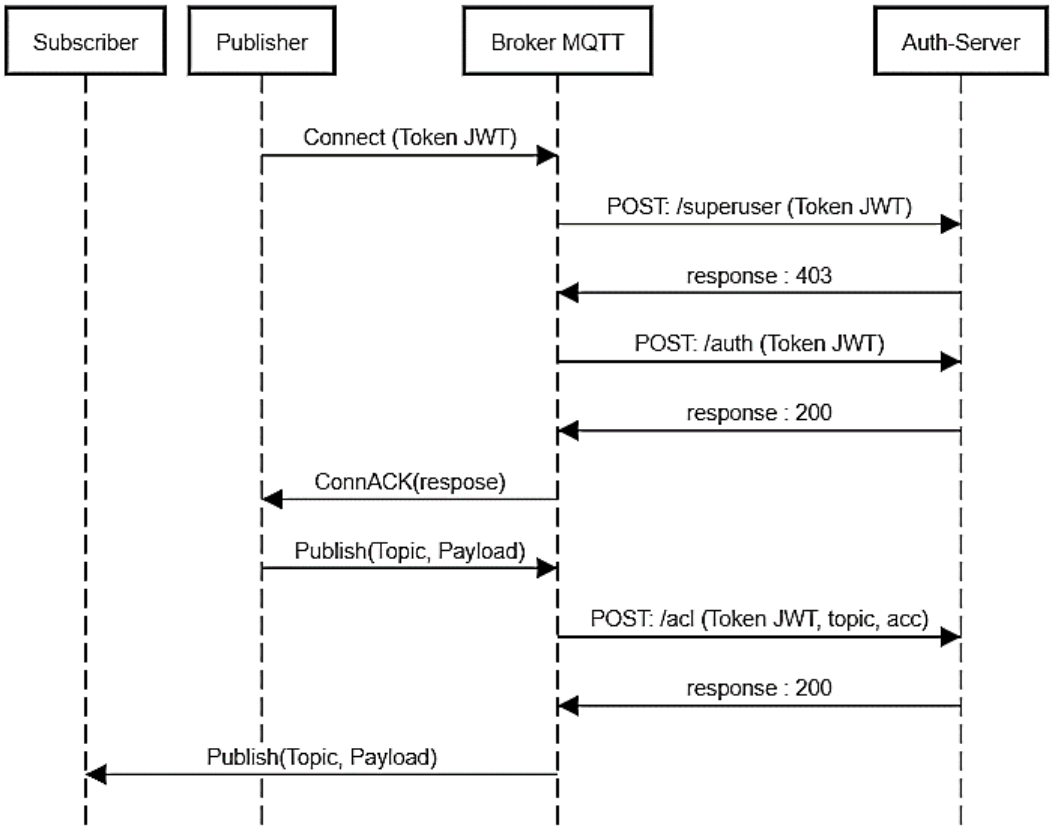
perihal apakah klien MQTT tersebut memiliki hak akses untuk *publish* dengan topik tersebut, kemudian hasil dari otorisasi dikirimkan kembali ke broker. Setelah broker menerima hasil otorisasi maka akan meneruskan pesan yang dipublish ke setiap klien yang subscribe ke topik tersebut jika hasil otorisasi berhasil, namun jika gagal maka permintaan *publish* akan ditolak.



Gambar 4.2 Alur Kerja Sistem Penanganan Publisher

Pada gambar 4.3, dijelaskan mengenai format pesan yang akan dikirimkan terkait dengan alur kerja sistem penanganan *publisher*. Format pesan pertama merupakan klien MQTT melakukan pesan *Connect* dengan token JWT ke broker untuk membuat koneksi, lalu broker akan mengirimkan pesan POST pada url '/superuser' dengan *header* token JWT dari *publisher* ke auth-server untuk melihat apakah klien MQTT tersebut adalah superuser atau tidak, jika ya maka akan auth-server akan mengirimkan *response* dengan nilai 200, namun jika tidak maka akan auth-server akan mengirimkan *response* 403. Selanjutnya broker akan mengirimkan pesan POST pada url '/auth' dengan header token JWT dari *publisher* ke auth-server untuk melakukan mekanisme autentikasi, lalu auth-server akan melakukan *decode* token JWT dan memeriksa nilai *username* dan *password* dari hasil *decode* ke database, jika sesuai maka akan auth-server akan mengirimkan *response* dengan nilai 200, namun jika tidak maka akan auth-server akan mengirimkan *response* 403. Jika autentikasi sesuai maka broker akan menyimpan

data klien MQTT tersebut dan akan mengirimkan pesan ConnACK yang menandakan bahwa klien tersebut berhasil terhubung. Saat klien MQTT mengirimkan pesan publish berisi topik dan *payload* ke broker, broker akan meneruskan pesan ke auth-server melalui pesan POST dengan url '/acl' disertai dengan *header* token JWT yang disimpan, topik, dan tipe pesan (*publish*/*subscribe*) ke auth-server. Auth-server akan melakukan mekanisme otorisasi dengan mencocokan data dari broker dan data pada ACL database, jika cocok maka akan dikirimkan *response* 200, namun jika tidak cocok maka akan dikirimkan *response* 403. Saat broker menerima *response* 200 maka pesan *publish* dari klien akan diteruskan ke *subscriber* pada topik tersebut, tetapi jika *response* 403 maka pesan *publish* dari klien tersebut akan di-*drop* oleh broker.

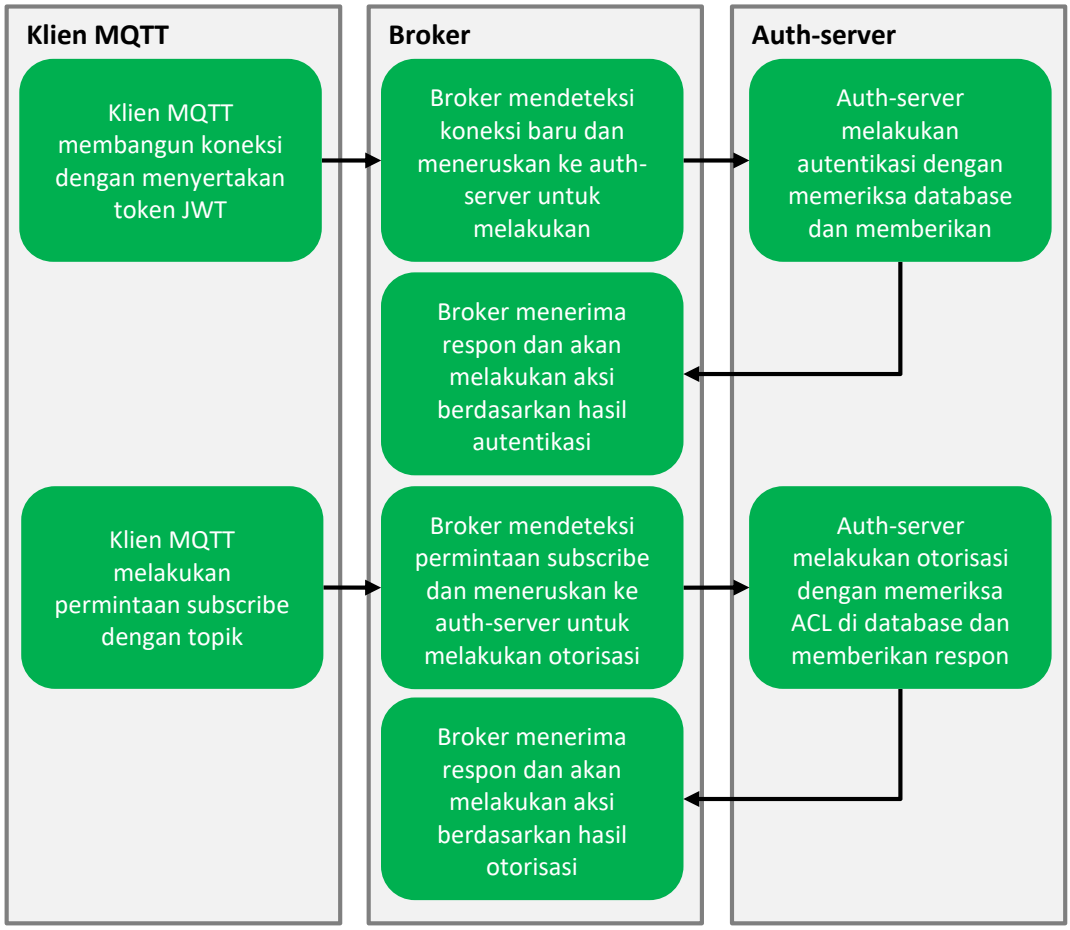


Gambar 4.3 Format Pesan Saat Publish

4.2.2 Perancangan *Subscriber*

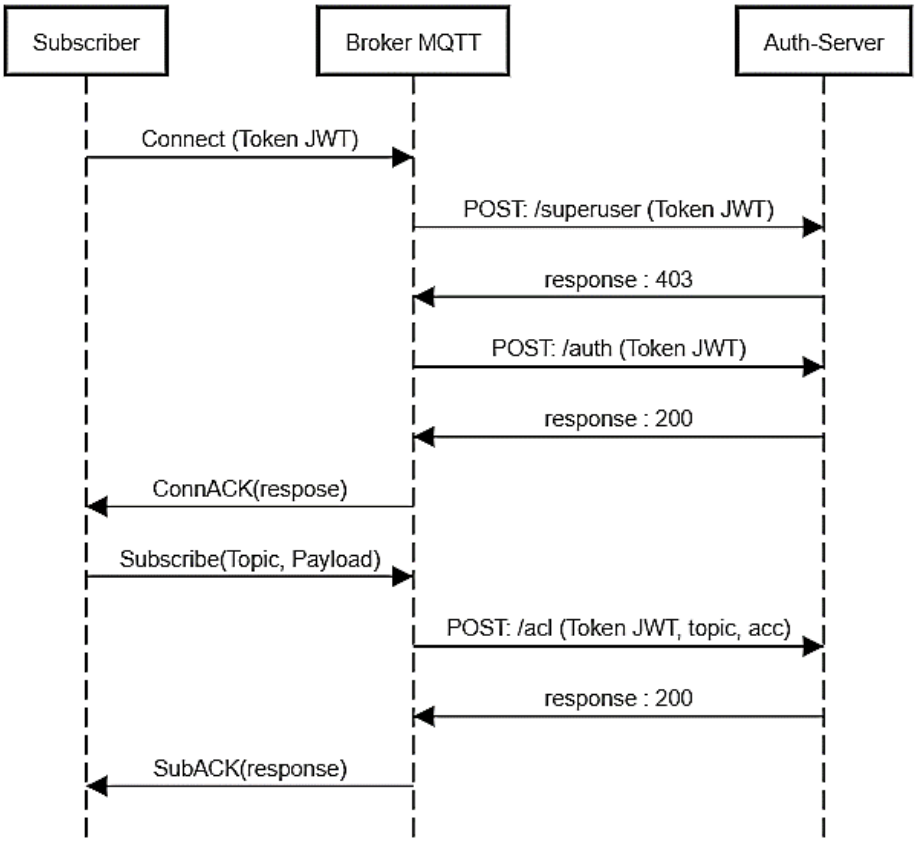
Pada gambar 4.4, dijelaskan mengenai alur kerja dari sistem yang akan dibangun dalam kasus penanganan *subscriber*. Penanganan *subscriber* akan dimulai oleh klien MQTT (dapat berupa pengguna / nodeMCU) sebelum dapat melakukan *subscribe* pada topik tertentu, diharuskan untuk membangun koneksi dengan broker MQTT terlebih dahulu, untuk membangun koneksi klien MQTT akan menyertakan sebuah token JWT yang berisi *username* dan *password*. Kemudian broker akan mendeteksi bahwa terdapat permintaan koneksi dari klien MQTT untuk kemudian diteruskan menuju auth-server untuk dilakukan mekanisme autentikasi. Auth-server kemudian menerima permintaan dari broker,

kemudian auth-server akan men-*decode* token JWT dan menggunakan *username* dan *password* yang terdapat didalamnya untuk melakukan mekanisme autentikasi dengan memeriksa identitas tersebut dalam database. Hasil dari pemeriksaan identitas kemudian dikirimkan kembali ke broker, kemudian broker akan menentukan aksi selanjutnya berdasarkan pada hasil autentikasi. Jika hasil autentikasi berhasil dan identitas klien MQTT terdapat dalam database maka broker akan memelihara koneksi hingga koneksi terputus, namun jika identitas klien MQTT tidak terdapat dalam database maka broker akan menolak permintaan koneksi. Selanjutnya klien MQTT akan mengirimkan permintaan *subscribe* pada topik ke broker setelah koneksi berhasil, broker yang mendeteksi bahwa terdapat permintaan *subscribe* akan meneruskan ke auth-server untuk dilakukan mekanisme otorisasi. Auth-server akan menerima permintaan dari broker dan memeriksa ACL pada database perihal apakah klien MQTT tersebut memiliki hak akses untuk *subscribe* pada topik tersebut, kemudian hasil dari otorisasi dikirimkan kembali ke broker. Setelah broker menerima hasil otorisasi maka akan maka klien MQTT tersebut akan disimpan sebagai *subscriber* pada topik yang diminati dan akan dikirimkan pesan saat terjadi *publish* pesan pada topik tersebut oleh *publisher* jika hasil otorisasi berhasil, namun jika gagal maka permintaan *subscribe* oleh klien akan ditolak.



Gambar 4.4 Alur Kerja Sistem Penanganan Subscriber

Pada gambar 4.5, dijelaskan mengenai format pesan yang akan dikirimkan terkait dengan alur kerja sistem penanganan *subscriber*. Format pesan awal untuk melakukan koneksi dari klien ke broker akan dilakukan dengan cara yang sama dengan penjelasan terkait format pesan *publisher* hingga broker mengirimkan psan ConnACK ke klien MQTT yang menandakan bahwa permintaan koneksi berhasil. Saat klien MQTT mengirimkan pesan *subscribe* berisi topik ke broker, broker akan meneruskan pesan ke auth-server melalui pesan POST dengan url 'acl' disertai dengan *header* token JWT yang disimpan, topik, dan tipe pesan (*publish*/*subscribe*) ke auth-server. Auth-server akan melakukan mekanisme otorisasi dengan mencocokan data dari broker dan data pada ACL database, jika cocok maka akan dikirimkan *response* 200, namun jika tidak cocok maka akan dikirimkan *response* 403. Saat broker menerima *response* 200 maka broker akan menyimpan data *subscriber* dan mengirimkan pesan SubACK yang menandakan bahwa klien tersebut berhasil melakukan *subscribe* pada topik tersebut, tetapi jika *response* 403 maka pesan *subscribe* dari klien tersebut akan di-*drop* oleh broker.

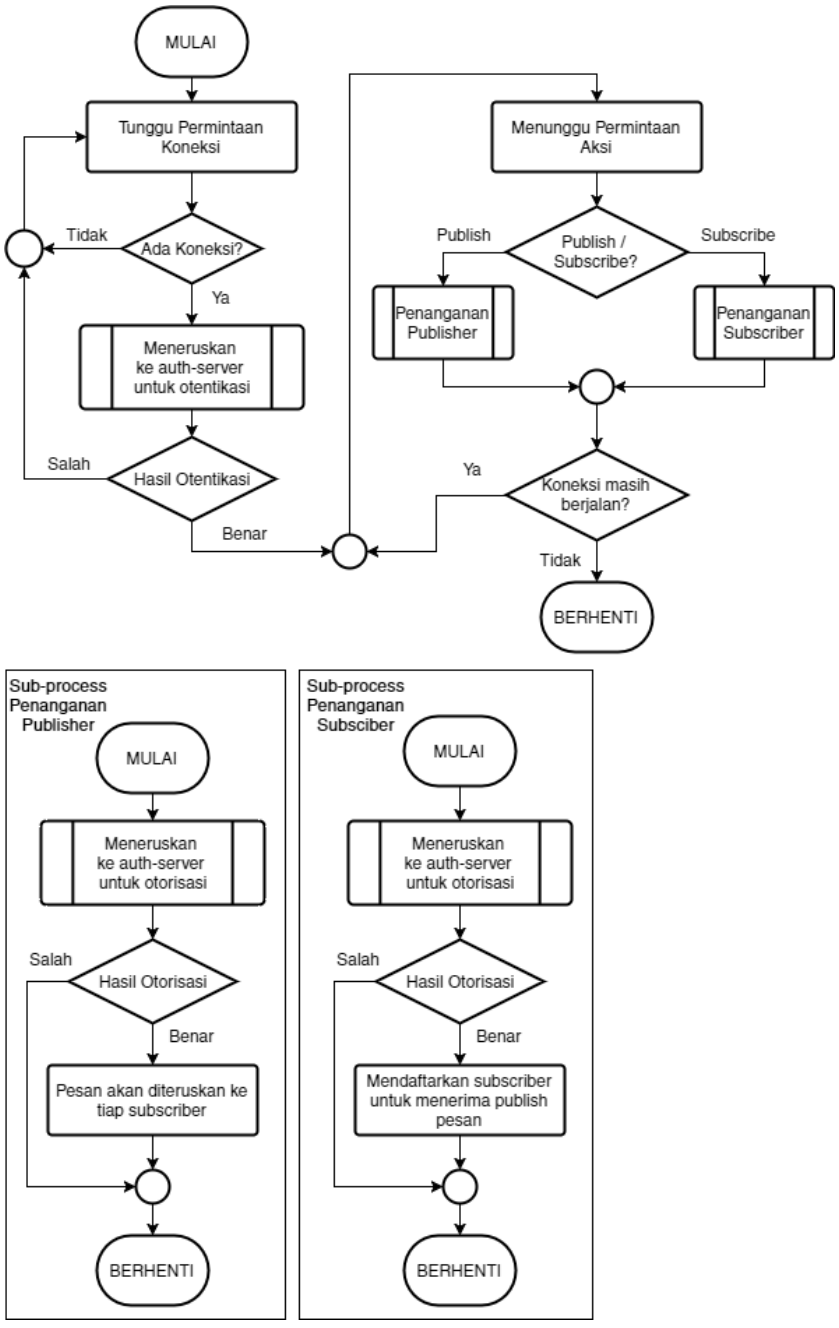


Gambar 4.5 Format Pesan Saat Subscribe

4.2.3 Perancangan Broker MQTT

Broker MQTT yang akan digunakan dalam sistem yang dirancang adalah broker dari *eclipse foundation*, yaitu mosquitto broker. Mosquitto broker berjalan pada protokol MQTT v3.1.1, yang akan digunakan sebagai server dan perantara antara klien MQTT yang melakukan *publish* dan *subscribe*. Mosquitto broker sendiri sudah menerapkan mekanisme autentikasi menggunakan username dan

password, hanya saja mekanisme autentikasi ini masih sangat sederhana dengan menyimpan data *raw username* dan *password* pada sebuah file. Sedangkan untuk mekanisme otorisasi, dapat dilakukan dengan menyimpan data *raw* dari *username*, topik dan hak akses yang juga disimpan pada sebuah file. Namun, mosquitto juga menyediakan opsi untuk menggunakan plugin eksternal untuk menerapkan mekanisme autentikasi dan otorisasi. Pada penelitian ini akan digunakan sebuah plugin mosquitto auth-plug, yang dapat dilihat pada *https://github.com/jpmens/mosquitto-auth-plug*. Plugin ini menyediakan mekanisme autentikasi dan otorisasi melalui beberapa *backend*. Konfirgurasi plugin ini akan dilakukan pada sebuah file "mosquitto.conf".



Gambar 4.6 Flowchart Broker Mosquitto

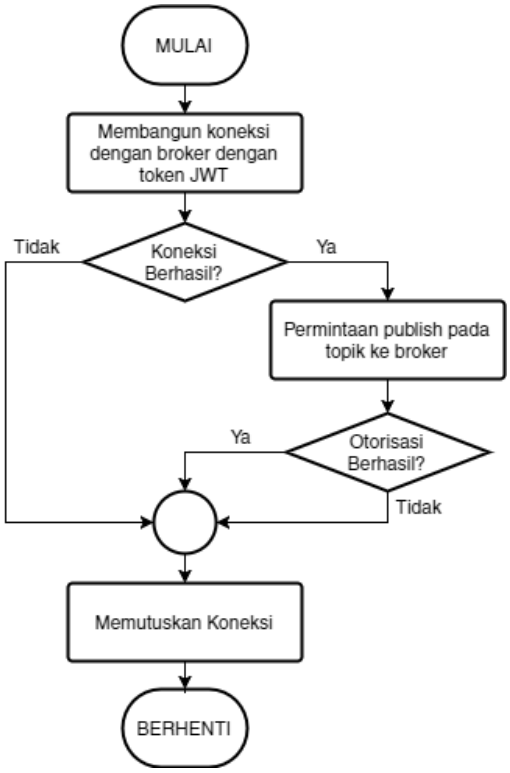
Flowchart algoritma dari mosquitto sebagai broker MQTT digambarkan pada gambar 4.6, diawali pada saat broker pertama kali berjalan maka akan menunggu tiap permintaan koneksi dari klien MQTT dan melakukan mekanisme autentikasi pada auth-server pada tiap permintaan koneksi. Jika respon dari auth-server adalah autentikasi berhasil maka koneksi dengan klien akan dipelihara oleh broker dan menunggu permintaan aksi (*publish*/*subscribe*) selanjutnya oleh klien MQTT tersebut. Lalu permintaan aksi tersebut akan dilakukan mekanisme otorisasi ke auth-server pada tiap permintaan, jika respon dari auth-server adalah otorisasi berhasil maka permintaan tersebut akan dijalankan sesuai dengan tipe aksi. Jika *publish*, maka pesan akan diteruskan ke tiap *subscriber* yang *subscribe* ke topik tersebut. Jika *subscribe*, maka klien MQTT tersebut akan didaftarkan sebagai *subscriber* pada topik tersebut.

4.2.4 Perancangan Klien MQTT

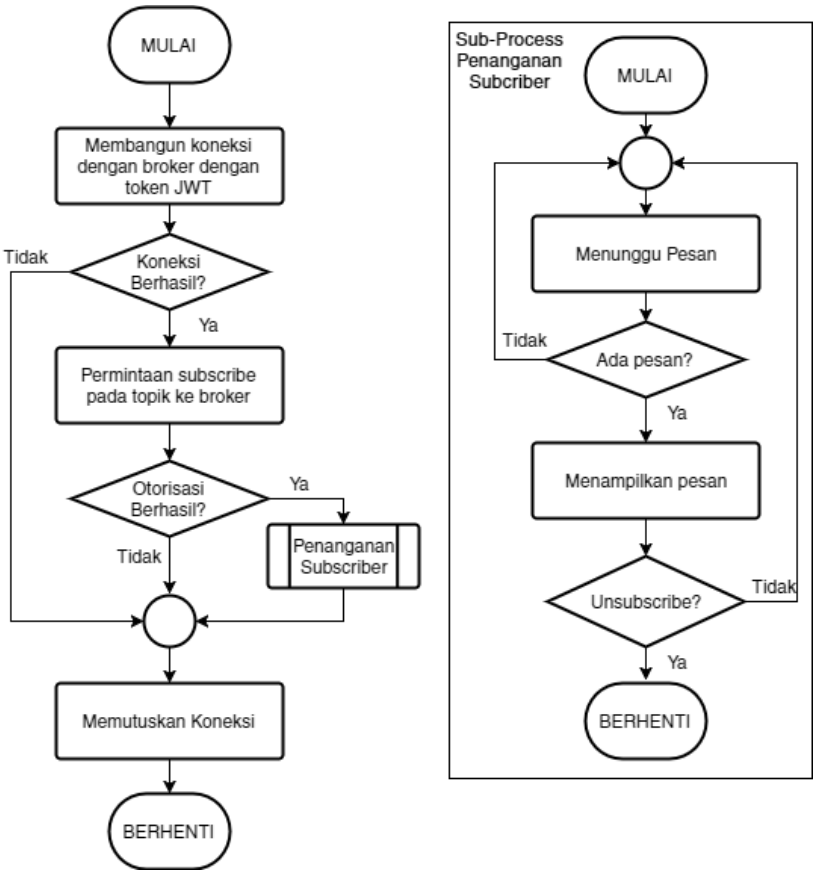
Klien MQTT merupakan semua klien yang dapat melakukan *publish* atau *subscribe* dalam sistem, baik itu berbentuk perangkat IoT ataupun berbentuk pengguna yang mengakses sitem. Oleh karena itu, setiap klien dapat melakukan *publish* atau *subscribe* dalam sistem, namun sistem yang dirancang disini kemudian akan menentukan apakah klien MQTT tersebut memiliki hak untuk mengakses aksi tersebut. Aksi *publish* pada mosquitto broker dapat dilakukan dengan menggunakan *command* mosquitto_pub pada terminal, yang akan melakukan *publish* satu pesan pada sebuah topik, perintah ini memiliki beberapa opsi yang dapat digunakan seperti topik, pesan, qos, *hostname*, *username*, *password,* dll. Aksi *subscribe* pada mosquitto broker dapat dilakukan dengan menggunakan *command* mosquitto_sub pada terminal, yang akan melakukan *subscribe* pada sebuah topik dan menampilkan tiap pesan yang diterima, perintah ini memiliki bebapa opsi yang dapat digunakan seperti topik, qos, *hostname*, *username*, *password,* dll.

Pada gambar 4.5, digambarkan *flowchart* algoritma dari klien MQTT sebagai *publisher* pada sistem yang menggunakan mosquitto broker yang menerapkan *plugin* auth-plug sebagai mekanisme autentikasi dan otorisasi. *Flowchart* tersebut menggambarkan mekanisme *publish* pesan menggunakan perintah mosquitto_pub dengan menggunakan token JWT yang dimasukkan sebagai *username*. Token JWT tersebut memiliki *payload username* dari klien MQTT dan *password*-nya, token ini akan digunakan sebagai identitas klien saat melakukan mekanisme autentikasi dan otorisasi yang akan dilakukan oleh auth-server.

Pada gambar 4.6, digambarkan *flowchart* algoritma dari klien MQTT sebagai *subscriber* pada sistem yang menggunakan mosquitto broker yang menerapkan *plugin* auth-plug sebagai mekanisme autentikasi dan otorisasi. *Flowchart* tersebut menggambarkan mekanisme *subscribe* pada topik menggunakan perintah mosquitto_sub dengan menggunakan token JWT yang dimasukkan sebagai *username*. Token JWT tersebut memiliki *payload username* dari klien MQTT dan *password*-nya, token ini akan digunakan sebagai identitas klien saat melakukan mekanisme autentikasi dan otorisasi yang akan dilakukan oleh auth-server.



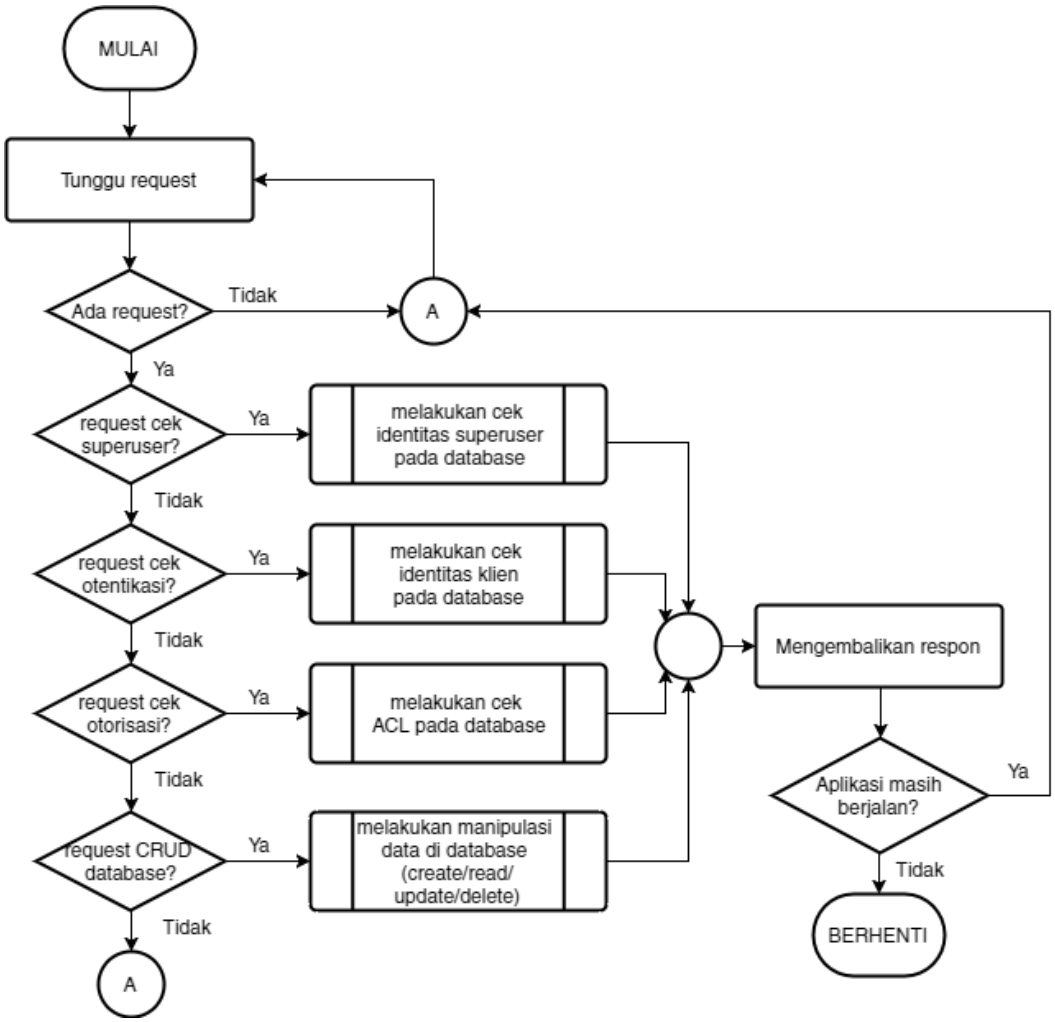
Gambar 4.7 Flowchart Klien MQTT sebagai Publisher



Gambar 4.8 Flowchart Klien MQTT sebagai Subscriber

4.2.5 Perancangan Auth-Server

Auth-server merupakan server yang bertindak sebagai server yang akan melakukan mekanisme cek autentikasi, cek otorisasi, cek superuser dan manipulasi database. Auth-server dirancang sebagai *backend* untuk pengimplementasian mosquitto auth-plug menggunakan python dan *framework* HTTP Bottle. Program python ini akan berjalan layaknya sebuah *micro web server* yang berkomunikasi dengan *request-response* melalui HTTP *method*. Auth-server dirancang sebagai penguhubung broker dan database MySQL, dimana didalam database tersebut tersimpan data klien MQTT dan data ACL sistem. Informasi yang tersimpan dalam database tersebut akan digunakan untuk melakukan mekanisme otorisasi, autentikasi dan superuser.



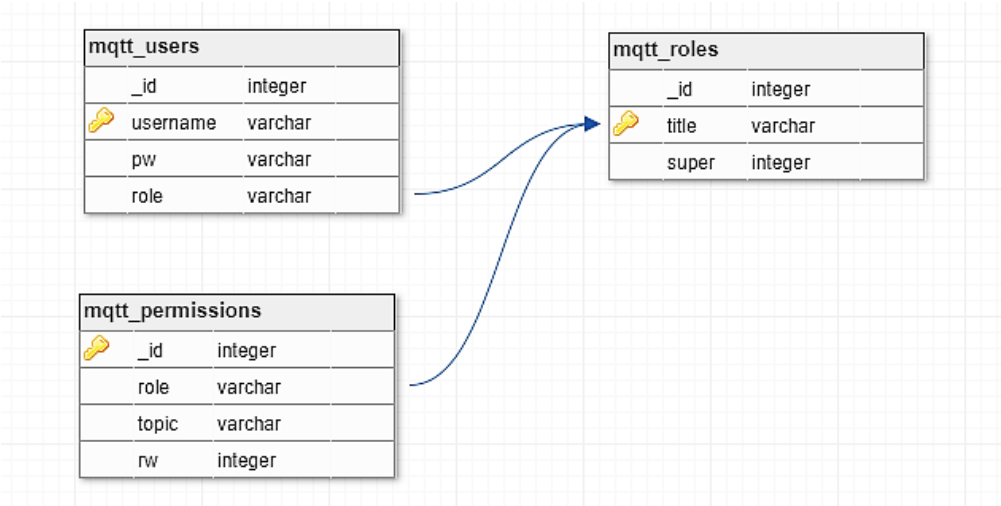
Gambar 4.9 Flowchart Auth-Server

Pada gambar 4.5, digambarkan *flowchart* algoritma dari auth-server. Saat pertama kali dijalankan dan selama masih berjalan auth-server akan menunggu *request* yang masuk. Ketika ada sebuah *request* yang masuk maka *request* tersebut akan diseleksi berdasarkan pada HTTP *method* dan url yang digunakan. Auth-server akan memiliki 4 fungsi utama, yaitu cek superuser, cek autentikasi, cek otorisasi dan CRUD (*Create, Read, Update* dan *Delete*) database, fungsi-fungsi ini

tujuan kemana *request* akan dilanjutkan. Tiap fungsi memiliki proses masing-masing didalamnya tetapi akan berakhir dengan sebuah *response* yang akan dikembalikan ke pihak yang melakukan *request*. Cek superuser akan memeriksa identitas dari klien MQTT apakah identitas klien merupakan superuser dalam database dan akan mengembalikan response yaitu hasil dari pemeriksaan tersebut. Cek autentikasi akan melakukan mekanisme autentikasi, dimana akan memeriksa identitas klien MQTT apakah identitas klien terdapat dalam database atau tidak dan akan mengembalikan hasil pemeriksaan tersebut. Cek otorisasi akan melakukan mekanisme otorisasi, dimana akan memeriksa apakah klien MQTT memiliki hak untuk mengakses topik dan jenis akses yang akan dilakukan berdasarkan data ACL yang ada dalam database. Dan CRUD database akan melakukan manipulasi data dalam database, tetapi sebelum melakukan CRUD akan dilakukan pengecekan identitas pihak yang mengakses dan akan mengembalikan sesuai dengan hasil dari manipulasi database.

4.2.6 Perancangan Skema Database

Perancangan skema database merupakan proses merancang database yang akan digunaan oleh sistem untuk menyimpan informasi-informasi terkait klien MQTT dan ACL. Database yang akan digunakan oleh sistem ini merupakan MySQL yang diimplementasikan dengan phpmyadmin. Skema database akan dirancang berdasarkan ACL yang akan digunakan oleh sistem, ACL yang akan digunakan ini dibuat berdasarkan peran yang akan dimiliki oleh tiap klien MQTT. Peran ini yang kemudian akan diberikan perizinan untuk mengakses (*publish*/*subscribe*) pada topik tertentu. Skema ACL semacam ini diharapkan dapat mempermudah tidak hanya proses mekanisme autentikasi dan otorisasi tetapi juga mempermudah dalam melakukan manajeman informasi pada database ketika sewaktu-waktu terjadi perubahan terkait perizinan dan hak akses klien MQTT.



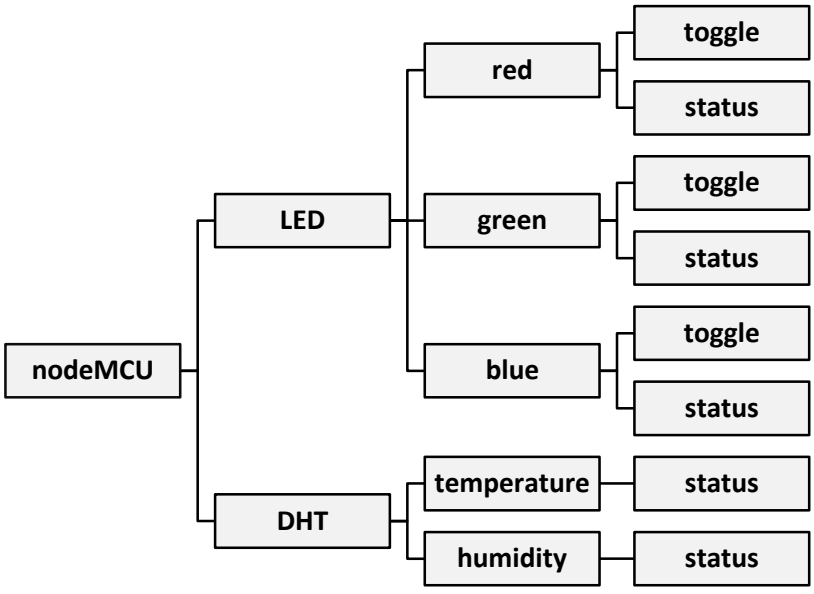
Gambar 4.10 Skema Database

Pada gambar 4.8, digambarkan hasil perancangan skema database yang akan digunakan oleh sistem. Database sistem akan memiliki 3 buah tabel yaitu mqtt_users, mqtt_roles dan mqtt_permissions. Tabel mqtt_roles akan diisi

dengan informasi terkait peran yang akan ada dalam sistem dari nama peran yang digunakan sebagai *primary key* dan apakah peran tersebut merupakan *superuser*. Tabel mqtt_users akan diisi dengan informasi milik klien dari *username* yang digunakan sebagai *primary key*, *hash password* dengan PBKDF2 dan peran berdasarkan pada informasi pada mqtt_roles. Tabel mqtt_permissions akan diisi dengan perizinan ACL yang akan diterapkan dalam sistem dari peran berdasarkan pada informasi di mqtt_roles, topik yang dapat diakses oleh peran tersebut, dan jenis akses yang dapat dilakukan oleh peran tersebut pada topik tersebut. Topik dan peran yang terdapat didalam database akan dirancang sedemikian rupa sehingga pengguna hanya dapat mengakses topik dan peran yang telah ditentukan.

4.2.7 Perancangan Topik & *Access Control List*

Access Control List (ACL) pada sistem yang akan dibuat akan dirancang berdasarkan topik apa yang dapat diakses oleh klien MQTT dan jenis akses seperti apa yang dapat dilakukan. Untuk itu, dibutuhkan sebuah struktur topik pada sistem yang dapat diakses oleh klien MQTT. Struktur topik akan dibuat kedalam *topic tree* dimana tiap cabang pada topic tree akan merepresentasikan hirarki topik yang dipisahkan dengan *forward slash* "/". Tiap topik dalam *topic tree* dirancang berdasarkan arsitektur sistem yang akan dibuat, yaitu berdasarkan sensor dan aktuator yang diterapkan pada mikrokontroler nodeMCU. Selain membutuhkan sebuah struktur topik pada sistem, dibutuhkan juga mekanisme terkait bagaimana ACL agar dapat mengatur mekanisme otorisasi. ACL akan dirancang menggunakan mekanisme *role-based access control*, dimana akan ada sebuah peran yang digunakan oleh sistem untuk menentukan peran tersebut dapat melakukan akses pada topik yang mana saja serta jenis akses. *Topic tree* ini akan dijadikan informasi dan dimasukkan kedalam database bersama dengan peran untuk menentukan hak akses klien MQTT.



Gambar 4.11 MQTT Topic Tree

Topic tree yang dirancang digambarkan pada gambar 4.9 diatas, perancangan *topic tree* ini berdasarkan pada tiap perangkat nodeMCU yang akan diimplementasi pada sistem dan sumberdaya yang dapat diakses oleh pengguna nantinya. Tiap cabang pada *topic tree* direpresentasikan oleh level hirarki topik pada protokol MQTT, dikarenakan topik MQTT bersifat *case-sensitive* maka tiap karakter huruf harus dinyatakan secara benar apakah menggunakan karakter besar atau karakter kecil. Pada topik level pertama terdapat nodeMCU yang merepresentasikan perangkat IoT yang digunakan dalam sistem, pada topik level kedua terdapat LED dan DHT yang merepresentasikan kedua tipe sensor atau aktuator yang digunakan oleh perangkat IoT untuk diakses sumberdayanya. Pada level ketiga setelah level LED terdapat *red*, *green* dan *blue* yang merepresentasikan warna dari tiap LED yang akan digunakan dalam sistem, dan pada level keempat di setiap warna tersebut terdapat status yang akan digunakan untuk melihat apakah LED tersebut dalam kondisi menyala atau mati dan *toggle* yang digunakan untuk menyalakan/mematikan LED. Pada level ketiga setelah level DHT terdapat *temperature* dan *humidity* yang merepresentasikan data mana yang akan dibaca oleh sensor apakah kelembaban atau suhu, dan pada level keempat dari tiap data tersebut terdapat status yang digunakan oleh mikrokontroler nodeMCU untuk mengambil data sensor dan mengirimkan ke pengguna.

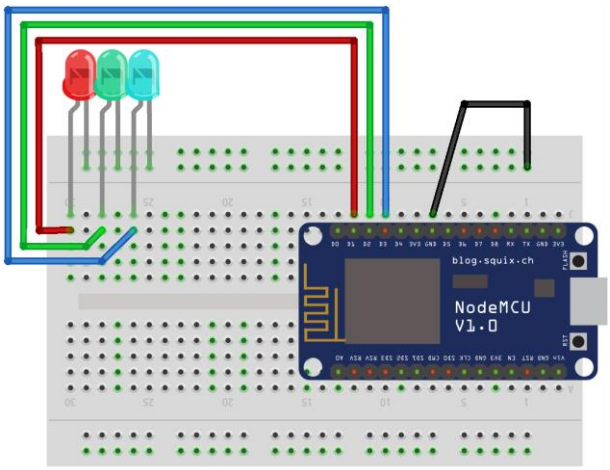
Hasil perancangan *topic tree* digunakan sebagai dasar untuk melakukan perancangan daftar peran dan perizinan pada sistem yang dapat dilihat pada tabel 4.3 dibawah, daftar peran dan perizinan akan dirancang berdasarkan pada tiap level hirarki topik MQTT serta bagaimana setiap peran tersebut memiliki fungsi yang dapat diakses masing-masing di dalam sistem berdasarkan pada perizinan yang dirancang tersebut. Setiap peran dapat melakukan *publish* atau *subscribe* pada topik tertentu, penentuan peran mana yang dapat mengakses topik yang mana dan jenis akses seperti apa dirancang berdasarkan pada *behavior* tiap peran yang akan dimiliki oleh pengguna terhadap sumberdaya apa yang dapat diakses di dalam sistem. Pada tabel 4.3 dibawah, terdapat karakter *wildcard* yang dinyatakan dengan menggunakan tanda plus “+” dan tanda pagar “#”. *Wildcard* dengan tanda plus mengartikan bahwa karakter tersebut dapat digantikkan nilainya dengan topik apapun yang ada pada level hirarki tersebut, dan *wildcard* dengan tanda pagar mengartikan bahwa karakter tersebut dapat digantikkan nilainya dengan topik apapun yang ada pada level hirarki tersebut dan juga semua topik yang ada dibawahnya. Setiap klien MQTT nantinya akan memiliki masing-masing hanya satu peran dan tidak dapat memiliki peran ganda. Data pada tabel juga digunakan sebagai daftar perizinan yang kemudian akan dimasukkan kedalam database MySQL sebagai *Access Control List* (ACL). Desain ACL yang seperti ini diharapkan dapat memudahkan dalam melakukan mekanisme otorisasi dan mekanisme autentikasi, dikarenakan sistem akan melihat peran yang dimiliki klien MQTT terkait hak akses mereka pada topik tersebut dan juga dapat memudahkan dalam proses manajemen hak akses karena setiap akan dilakukan perubahan hak akses pada sistem kita tidak perlu menetapkan atau mengganti tiap klien MQTT secara spesifik melainkan hanya perlu menetapkan atau mengganti peran tertentu dan semua klien dalam peran tersebut akan terpengaruhi secara langsung.

Tabel 4.3 Desain Role & Permissions

Role	Permissions		Deskripsi
	PUB / SUB	Topik	
Admin	PUB	#	Pengguna dapat Publish pada topic manapun; bertindak sebagai superuser
	SUB	#	Pengguna dapat Subscribe pada topic manapun; bertindak sebagai superuser
toggle_user	PUB	nodeMCU/LED/+/ toggle	Pengguna dapat menyalakan led red, green dan blue
status_user	SUB	nodeMCU/+/+/ status	Pengguna dapat menerima status sensor, baik tiap led (red, green dan blue) atau dht (temperature dan humidity)
red_user	PUB	nodeMCU/LED/red/ toggle	Pengguna dapat menyalakan led red saja
	SUB	nodeMCU/LED/red/ status	Pengguna dapat menerima status led red saja
green_user	PUB	nodeMCU/LED/ green/toggle	Pengguna dapat menyalakan led green saja
	SUB	nodeMCU/LED/ green/status	Pengguna dapat menerima status led green saja
blue_user	PUB	nodeMCU/LED/ blue/toggle	Pengguna dapat menyalakan led blue saja
	SUB	nodeMCU/LED/ blue/status	Pengguna dapat menerima status led blue saja
temperature _user	SUB	nodeMCU/DHT/ temperature/ request	Pengguna dapat request status temperatur dht
humidity _user	SUB	nodeMCU/DHT/ humidity/request	Pengguna dapat request status humidity dht
led_sensor	PUB	nodeMCU/LED/+/ status	Pengguna mengirim status tiap led (red, green, blue); untuk nodeMCU
	SUB	nodeMCU/LED/+/ toggle	Pengguna menerima perintah untuk menyalakan led (red, green, blue) ; untuk nodeMCU
dht_sensor	PUB	nodeMCU/DHT/+/ request	Pengguna mengirim status tiap dht (temperature, humidity); untuk nodeMCU

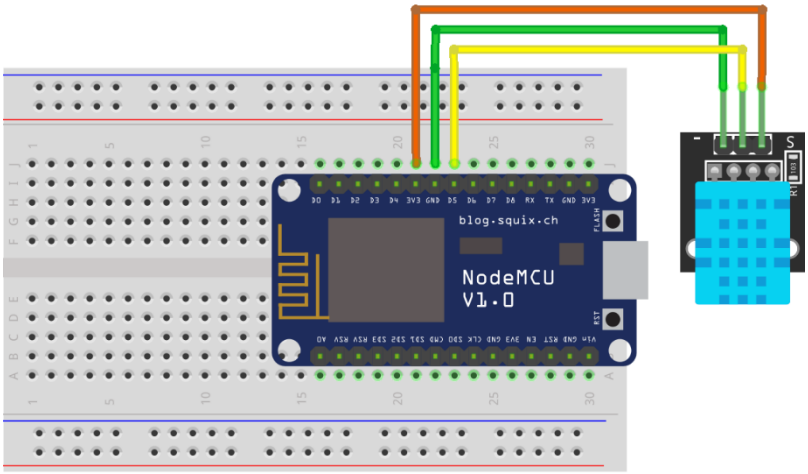
4.2.8 Perancangan Perangkat NodeMCU

Perangkat nodeMCU berperan sebagai klien MQTT dalam sistem, yang dapat melakukan publish dan subscribe ke broker. Hanya peran nodeMCU akan berbeda dengan klien lainnya, yaitu sebagai mikrokontroler untuk sensor dan aktuator. Klien MQTT yang lain akan mengakses informasi miliki sensor dan aktuator pada perangkat nodeMCU. Sensor yang akan digunakan adalah DHT11 sebagai sensor suhu dan kelembaban yang akan dihubungkan dengan perangkat nodeMCU sebagai mikrokontrolernya, sementara aktuator yang digunakan adalh 3 buah LED yang berbeda warna yang terhubung dengan nodeMCU yang berbeda sebagai mikrokontrolernya. Perancangan nodeMCU akan dilakukan pada bagian perangkat keras dan perangkat lunaknya.



Gambar 4.12 Skema nodeMCU + LED

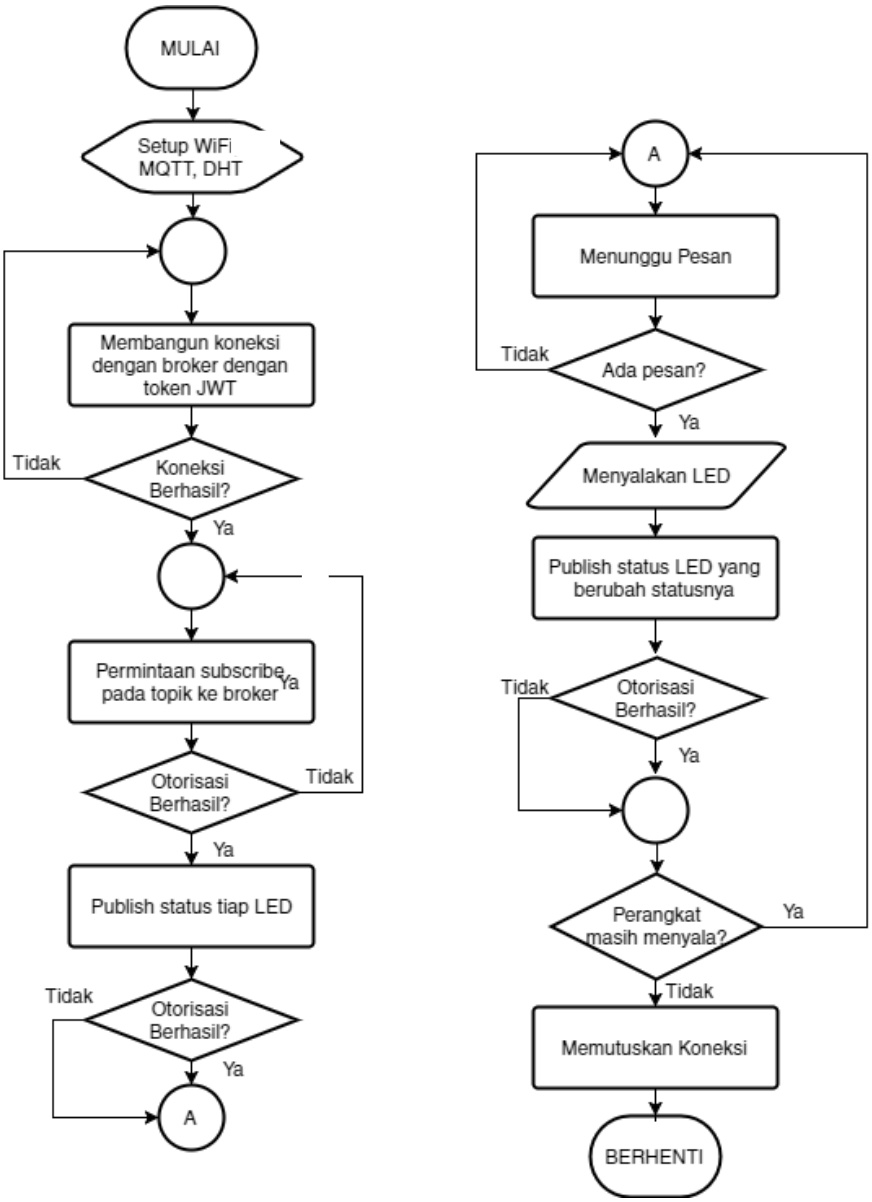
Perancangan perangkat keras nodeMCU dengan LED dapat dilihat pada gambar 4.10, dimana perangkat nodeMCU yang pertama akan dihubungkan dengan 3 buah LED yang berbeda warna yaitu merah, hijau dan biru. Tiap LED akan dihubungkan melalui kabel *jumper* ke salah satu pin GPIO yang tersedia pada nodeMCU dan juga akan terhubung ke *ground* dari nodeMCU. NodeMCU dengan LED ini akan berperan sebagai mikrokontroler dan aktuator dimana klien MQTT lainnya akan dapat menyalakan/mematikan LED serta dapat membaca status dari tiap LED yang ada. Karena itu, nodeMCU ini akan berperan sebagai *subscriber* yang menunggu perintah menyalakan LED dan *publisher* yang akan mengirim status LED pada *subscriber*.



Gambar 4.13 Skema nodeMCU + DHT11

Perancangan perangkat keras nodeMCU dengan DHT11 dapat dilihat pada gambar 4.12, dimana perangkat nodeMCU yang kedua akan dihubungkan dengan sebuah sensor DHT11. DHT11 akan dihubungkan melalui kabel *jumper* ke salah satu pin GPIO yang tersedia pada nodeMCU, juga akan terhubung ke *ground* dari nodeMCU, serta ke salah satu pin 3v milik nodeMCU sebagai daya untuk sensor. NodeMCU dengan DHT11 ini akan berperan sebagai mikrokontroler dan sensor dimana klien MQTT lainnya akan dapat menerima status dari suhu dan

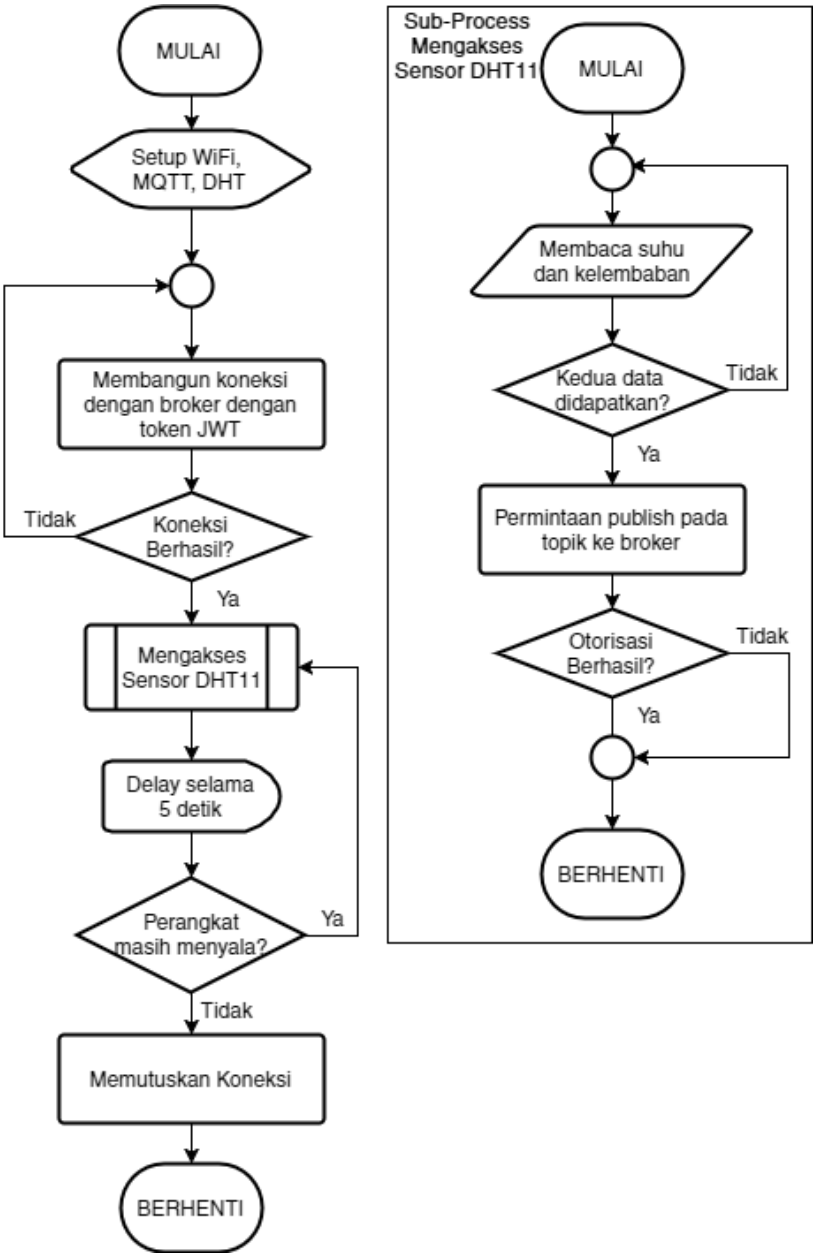
kelembaban yang diperoleh oleh DHT11. Karena itu, nodeMCU ini akan berperan *publisher* yang akan mengirim status suhu dan kelembaban pada *subscriber*.



Gambar 4.14 Flowchart nodeMCU + LED

Pada gambar 4.11, digambarkan *flowchart* algoritma dari nodeMCU + LED sebagai klien MQTT pada sistem yang menggunakan mosquitto broker yang menerapkan *plugin* auth-plug sebagai mekanisme autentikasi dan otorisasi. Pada dasarnya algoritma yang digunakan sama dengan milik klien MQTT pada perancangan sebelumnya hanya saja karena nodeMCU + LED ini dapat melakukan publish dan subscribe maka prosesnya menjadi sedikit lebih kompleks. Pertama akan membuat setup untuk WiFi dan MQTT, kemudian akan dilakukan koneksi ke broker dengan token JWT yang dimasukkan sebagai *username*. Token JWT tersebut memiliki *payload username* dari klien MQTT dan *password*-nya, token ini akan digunakan sebagai identitas klien saat melakukan mekanisme autentikasi dan otorisasi yang akan dilakukan oleh auth-server. Kemudian akan melakukan

subscribe ke topik "*toggle*" sampai berhasil, setelah itu akan melakukan *publish* status tiap LED untuk pertama kali. Lalu akan menunggu *publish* pesan untuk menyalakan/mematikan salah satu LED, jika ada pesan maka LED tersebut akan dinyalakan dan di-*publish* status LED yang diubah statusnya tersebut.



Gambar 4.15 Flowchart nodeMCU + DHT11

Pada gambar 4.13, digambarkan *flowchart* algoritma dari nodeMCU + DHT11 sebagai klien MQTT pada sistem yang menggunakan mosquitto broker yang menerapkan *plugin* auth-plug sebagai mekanisme autentikasi dan otorisasi. Pada dasarnya algoritma yang digunakan sama dengan milik klien MQTT pada perancangan sebelumnya hanya saja karena nodeMCU + DHT11 ini berperan sebagai sensor maka prosesnya menjadi sedikit lebih kompleks. Pertama akan membuat setup untuk WiFi, DHT dan MQTT, kemudian akan dilakukan koneksi ke

broker dengan token JWT yang dimasukkan sebagai *username*. Token JWT tersebut memiliki *payload username* dari klien MQTT dan *password*-nya, token ini akan digunakan sebagai identitas klien saat melakukan mekanisme autentikasi dan otorisasi yang akan dilakukan oleh auth-server. Kemudian akan melakukan pembacaan data suhu dan kelembaban hingga data didapatkan, karena terkadang sensor DHT11 gagal membaca dan memberikan nilai *null*. Setelah data berhasil dibaca maka data tersebut akan dipublish menuju ke broker, lalu akan melakukan delay selama 5 detik untuk memberikan jeda pembacaan sensor agar tidak membanjiri broker dan *subscriber*.

4.3 Perancangan Pengujian

Perancangan pengujian dilakukan untuk mengetahui hasil dari sistem yang telah dibuat. Dimana sistem yang telah dibuat akan dilakukan pengujian dengan beberapa macam pengujian dengan menentukan batasan-batasan minimal yang harus dipenuhi oleh sistem berdasarkan pada skenario yang dirancang. Dalam penelitian ini akan dilakukan tiga macam pengujian, yang dibagi menjadi pengujian fungsionalitas, pengujian keamanan dan pengujian performa. Dengan dilakukan pengujian ini diharapkan dapat mengetahui apakah sistem dapat menyelesaikan permasalahan yang telah dirumuskan sebelumnya.

4.3.1 Pengujian Fungsional Sistem

Pengujian fungsional adalah pengujian yang dilakukan untuk mengetahui apakah kebutuhan fungsional yang telah dirancang untuk sistem yang akan dibuat sudah terpenuhi dan juga dapat berjalan dengan baik. Pengujian ini akan dilakukan dengan menguji setiap kebutuhan fungsional yang telah didefinisikan. Hasil pengujian fungsional merupakan tiap proses yang harus dijalankan dan dipenuhi oleh sistem yang telah dibuat. Pengujian fungsional sistem akan dilakukan dengan membuat sebuah skenario berdasarkan berdasarkan pada kebutuhan fungsional yang telah didefinisikan, dan akan dibuat skenario dari fungsionalitas tersebut secara lebih terperinci dan lebih detail. Pengujian akan dilakukan dengan mengikuti beberapa prosedur yang berbeda untuk tiap skenario, yang digambarkan dalam tabel 4.4 berikut.

Tabel 4.4 Skenario Pengujian Fungsional Sistem

Kode	Fungsi	Prosedur Skenario
PFS_001	Klien MQTT dapat membangun koneksi dengan broker MQTT menggunakan token JWT	1. Klien mengirimkan pesan *Connect* dengan menyertakan token JWT yang *valid* 2. Melihat hasil di *interface* broker 3. Melihat hasil di *interface* auth-server
PFS_002	*Publisher* dapat melakukan *Publish* pada suatu topik ke broker MQTT	1. Klien mengirimkan pesan *Publish* pada topik tertentu dengan menyertakan token JWT 2. Melihat hasil di *interface* broker 3. Melihat hasil di *interface* auth-server

PFS_003	*Subscriber* dapat melakukan *Subscribe* pada suatu topik ke broker MQTT	1. Klien mengirimkan pesan *Subscribe* pada topik tertentu dengan menyertakan token JWT 2. Melihat hasil di *interface* broker 3. Melihat hasil di *interface* auth-server
PFS_004	Auth-server dapat melakukan mekanisme autentikasi dengan memeriksa identitas klien MQTT pada database.	1. Klien mengirimkan token JWT yang *valid* pada *header request* melalui HTTP-*method POST* 2. Klien mengirimkan token JWT yang tidak *valid* pada *header request* melalui HTTP-*method POST* 3. Melihat hasil di *interface* auth-server
PFS_005	Auth-server dapat melakukan mekanisme otorisasi dengan memeriksa ACL terkait hak akses klien MQTT pada database.	1. Klien mengirimkan token JWT yang *valid* di *header request*, topik dan acc di *body request* melalui HTTP-*method POST* 2. Klien mengirimkan token JWT yang tidak *valid* di *header request*, topik dan acc di *body request* melalui HTTP-*method POST* 3. Melihat hasil di *interface* auth-server
PFS_006	Auth-server dapat melakukan *generate* token JWT berdasarkan *username* dan *password* pengguna yang ada di database	1. Klien mengirimkan *request* dengan *body request username* dan *password* yang *valid* melalui HTTP-*method POST* 2. Klien mengirimkan *request* dengan *body request username* dan *password* yang tidak *valid* melalui HTTP-*method POST* 3. Melihat hasil di *interface* auth-server
PFS_007	Admin dapat melakukan *Create User* untuk membuat sebuah pengguna baru lewat auth-server	1. Klien mengirimkan *request* dengan *header request* token JWT admin melalui HTTP-*method POST*, dengan *body request username*, *pasword* dan peran pengguna baru 2. Melihat hasil di *interface* auth-server 3. Melihat hasil di database
PFS_008	Admin dapat melakukan *Update User* untuk mengubah pengguna lewat auth-server	1. Klien mengirimkan *request* dengan *header request* token JWT admin melalui HTTP-*method PUT*, dengan *body request username*, *pasword* dan peran 2. Melihat hasil di *interface* auth-server 3. Melihat hasil di database
PFS_009	Admin dapat melakukan *Delete User* untuk menghapus pengguna lewat auth-server	1. Klien mengirimkan *request* dengan *header request* token JWT admin melalui HTTP-*method DELETE*, dengan *body request username*, *pasword* dan peran 2. Melihat hasil di *interface* auth-server 3. Melihat hasil di database

PFS_010	Admin dapat melakukan *Create Role* untuk membuat peran baru lewat auth-server	1. Klien mengirimkan *request* dengan *header request* token JWT admin melalui HTTP-method POST, dengan *body request title* peran baru 2. Melihat hasil di *interface* auth-server 3. Melihat hasil di database
PFS_011	Admin dapat melakukan *Delete Role* untuk menghapus peran lewat auth-server	1. Klien mengirimkan *request* dengan *header request* token JWT admin melalui HTTP-*method DELETE*, dengan *body request title* peran 2. Melihat hasil di *interface* auth-server 3. Melihat hasil di database
PFS_012	Admin dapat melakukan *Create Permission* untuk membuat perizinan baru lewat auth-server	1. Klien mengirimkan *request* dengan *header request* token JWT admin melalui HTTP-*method POST*, dengan *body request* peran, topik dan acc baru 2. Melihat hasil di *interface* auth-server 3. Melihat hasil di database
PFS_013	Admin dapat melakukan *Delete Permission* untuk menghapus perizinan lewat auth-server	1. Klien mengirimkan *request* dengan *header request* token JWT admin melalui HTTP-*method DELETE*, dengan *body request* peran, topik dan acc baru 2. Melihat hasil di *interface* auth-server 3. Melihat hasil di database
PFS_014	NodeMCU + LED berhasil terhubung dengan broker serta melakukan *subscribe* dan melakukan *publish*	1. Menghubungkan perangkat nodeMCU + LED dengan daya 2. Melakukan *publish* untuk menyalakan salah satu LED 3. Melakukan *subscribe* untuk menerima status dari nodeMCU 4. Melihat hasil *publish* di serial monitor 5. Melihat hasil *subscribe* dari nodeMCU
PFS_015	NodeMCU + DHT11 berhasil terhubung dengan broker serta membaca data dan melakukan *publish*	1. Menghubungkan perangkat nodeMCU + DHT11 dengan daya 2. Melakukan *subscribe* untuk menerima status dari nodeMCU 3. Melihat hasil *subscribe* dari nodeMCU

4.3.2 Pengujian Keamanan Sistem

Pengujian keamanan akan dilakukan untuk mengetahui aspek keamanan dari sistem berbasis protokol MQTT yang menerapkan mekanisme autentikasi dan otorisasi. Pengujian ini dilakukan dengan melihat bagaimana hasil penerapan auth-server sebagai pihak yang melakukan cek autentikasi dan otorisasi pada sistem, bagaimana klien MQTT dapat mengakses sistem dan bagaimana hasil perancangan ACL dapat mengatur otorisasi klien MQTT. Pengujian keamanan akan dilakukan dengan membuat tiga skenario pengujian menggunakan konfigurasi mosquitto broker yang berbeda, ketiga konfigurasi ini digunakan untuk

membandingkan aspek keamanan sistem terkait broker MQTT yang digunakan. Skenario konfigurasi mosquitto broker untuk pengujian keamanan akan dijelaskan sebagai berikut:

1. Konfigurasi mosquitto broker yang tidak menerapkan mekanisme otorisasi dan autentikasi, serta tidak menggunakan keamanan pada *transport layer* menggunakan TLS/SSL.
2. Konfigurasi mosquitto broker yang menerapkan mekanisme otorisasi dan autentikasi menggunakan ACL pada auth-server, tetapi tidak menggunakan keamanan pada *transport layer* menggunakan TLS/SSL.
3. Konfigurasi mosquitto broker yang menerapkan mekanisme otorisasi dan autentikasi menggunakan ACL pada auth-server, serta menggunakan keamanan pada *transport layer* menggunakan TLS/SSL.

Pengujian pada ketiga konfigurasi ini dilakukan untuk membandingkan pengaruh dari auth-server dan TLS terhadap keamanan sistem broker MQTT. Dengan menggunakan konfigurasi auth-server diharapkan dapat mengamankan sistem dari klien yang tidak memiliki hak akses atau tidak terdaftar dalam sistem dengan menggunakan pemeriksaan mekanisme autentikasi dan otorisasi melalui ACL. Sedangkan dengan menggunakan konfigurasi TLS diharapkan dapat mengamankan *user-credential* milik klien agar tidak dapat dicuri melalui pengamanan enkripsi. Pengujian keamanan sendiri akan dilakukan dengan melakukan kedua mekanisme pengamanan dan melihat hasil yang didapatkan, kedua mekanisme itu merupakan autentikasi dan otorisasi.

4.3.2.1 Pengujian Mekanisme Autentikasi

Pengujian ini dilakukan untuk menguji bagaimana mekanisme autentikasi dari ketiga konfigurasi broker dalam aspek keamanan. Untuk melakukan pengujian ini akan dilakukan beberapa skenario pengujian dimana masing-masing skenario akan memiliki prosedur yang akan dilakukan untuk pengujian seperti yang dijelaskan pada tabel 4.5 berikut ini.

Tabel 4.5 Skenario Pengujian Mekanisme Autentikasi Sistem

Konfigurasi	Kode	Skenario
Tanpa Auth-Server & Tanpa TLS	PMO_101	Klien membangun koneksi dengan broker tanpa menggunakan *username* dan *password*.
	PMO_102	Klien membangun koneksi dengan broker menggunakan *username* dan *password*.
Dengan Auth-server & Tanpa TLS	PMO_201	Klien membangun koneksi dengan broker tanpa menggunakan token JWT
	PMO_202	Klien membangun koneksi dengan broker menggunakan token JWT yang *invalid*
	PMO_203	Klien membangun koneksi dengan broker menggunakan token JWT yang *valid*

Dengan Auth-server & TLS	PMO_301	Klien membangun koneksi dengan broker tanpa menggunakan token JWT
	PMO_302	Klien membangun koneksi dengan broker menggunakan token JWT yang *invalid*
	PMO_303	Klien membangun koneksi dengan broker menggunakan token JWT yang *valid*

4.3.2.2 Pengujian Mekanisme Otorisasi dengan Publish

Pengujian ini dilakukan untuk menguji bagaimana mekanisme otorisasi dari ketiga konfigurasi broker dalam aspek keamanan dengan cara melakukan *publish*. Untuk melakukan pengujian ini akan dilakukan beberapa skenario pengujian dimana masing-masing skenario akan memiliki prosedur yang akan dilakukan untuk pengujian seperti yang dijelaskan pada tabel 4.6 berikut ini.

Tabel 4.6 Skenario Pengujian Mekanisme Otorisasi (Publish)

Konfigurasi	Kode	Skenario
Tanpa Auth-Server & Tanpa TLS	PMOP_101	Klien melakukan *publish* pada topik yang tidak ada dalam *topic tree* yang dibuat.
	PMOP_102	Klien melakukan *publish* pada topik yang ada dalam *topic tree* yang dibuat.
Dengan Auth-server & Tanpa TLS	PMOP_201	Klien melakukan *publish* pada topik diluar *topic tree* yang dibuat.
	PMOP_202	Klien melakukan *publish* pada topik yang tidak sesuai dengan hak akses di *topic tree* yang dibuat.
	PMOP_203	Klien melakukan *publish* pada topik yang sesuai dengan hak akses di *topic tree* yang dibuat.
Dengan Auth-server & TLS	PMOP_301	Klien melakukan *publish* pada topik diluar *topic tree* yang dibuat.
	PMOP_302	Klien melakukan *publish* pada topik yang tidak sesuai dengan hak akses di *topic tree* yang dibuat.
	PMOP_303	Klien melakukan *publish* pada topik yang sesuai dengan hak akses di *topic tree* yang dibuat.

4.3.2.3 Pengujian Mekanisme Otorisasi dengan Subscribe

Pengujian ini dilakukan untuk menguji bagaimana mekanisme autentikasi dari ketiga konfigurasi broker dalam aspek keamanan dengan cara melakukan *subscribe*. Untuk melakukan pengujian ini akan dilakukan beberapa skenario pengujian dimana masing-masing skenario akan memiliki prosedur yang akan dilakukan untuk pengujian seperti yang dijelaskan pada tabel 4.7 berikut ini.

Tabel 4.7 Skenario Pengujian Mekanisme Otorisasi (Subscribe)

Konfigurasi	Kode	Skenario
Tanpa Auth-Server & Tanpa TLS	PMOS_101	Klien melakukan *subscribe* pada topik yang tidak ada dalam *topic tree* yang dibuat.
	PMOS_102	Klien melakukan *subscribe* pada topik yang ada dalam *topic tree* yang dibuat.
Dengan Auth-server & Tanpa TLS	PMOS_201	Klien melakukan *subscribe* pada topik diluar *topic tree* yang dibuat.
	PMOS_202	Klien melakukan *subscribe* pada topik yang tidak sesuai dengan hak akses di *topic tree* yang dibuat.
	PMOS_203	Klien melakukan *subscribe* pada topik yang sesuai dengan hak akses di *topic tree* yang dibuat.
Dengan Auth-server & TLS	PMOS_301	Klien melakukan *subscribe* pada topik diluar *topic tree* yang dibuat.
	PMOS_302	Klien melakukan *subscribe* pada topik yang tidak sesuai dengan hak akses di *topic tree* yang dibuat.
	PMOS_303	Klien melakukan *subscribe* pada topik yang sesuai dengan hak akses di *topic tree* yang dibuat.

4.3.3 Pengujian Performa Sistem

Pengujian performa akan dilakukan untuk mengetahui bagaimana performa sistem berdasarkan pada tiga konfigurasi mosquitto broker yang dilakukan pada pengujian kemanan yang akan dibandingkan berdasarkan pengaruh dari tiap konfigarasi tersebut terkait pengiriman paket dan waktu yang dibutuhkan untuk mengirimkan paket tersebut. Pengujian performa sistem diharapkan akan mengetahui pengaruh dari penerapan auth-server sebagai mekanisme autentikasi dan otorisasi, serta pengaruh dari penerapan TLS yang digunakan untuk mengamankan sistem apakah dari penerapan tersebut mempengaruhi performa dari broker untuk menangani klien apakah terjadi penurunan performa dari broker secara drastis atau masih dalam kondisi wajar.

Untuk mengetahui performa dari broker, pada pengujian ini akan digunakan beberapa parameter untuk menentukan hasil dari performa sistem dengan membandingkan ketiga konfigurasi broker yang digunakan pada pengujian keamanan, beberapa parameter tersebut akan meliputi :

1. Waktu yang dibutuhkan untuk membangun koneksi (*Average Connection Time*), dalam satuan detik.
2. Waktu yang dibutuhkan untuk pengiriman / publish satu pesan (*Average Publish Time*), dalam satuan detik.
3. Jumlah pesan yang mampu dikirimkan dalam satu detik (*Throughput*), dalam satuan pesan per detik.