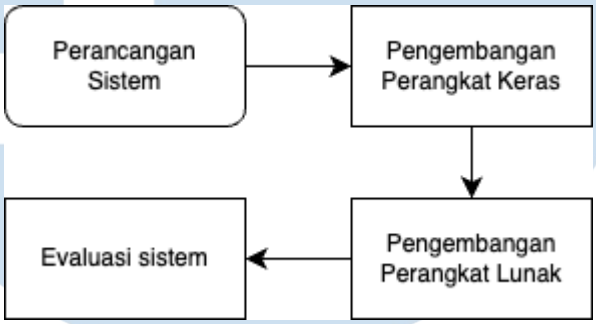
BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Metode Penelitian

Pada penelitian yang dilakukan oleh penulis, ada beberapa tahapan yang akan dilakukan untuk meneliti dan merancang sistem yang bertujuan untuk menjawab identifikasi masalah yang telah dijelaskan pada Bab I. Metode yang dilakukan oleh penulis terdiri dari studi literatur, perancangan sistem, pengembangan perangkat keras, pengembangan perangkat lunak dan evaluasi sistem seperti yang dapat dilihat pada gambar 3.1.



Gambar 3.1 Flowchart metode penelitian

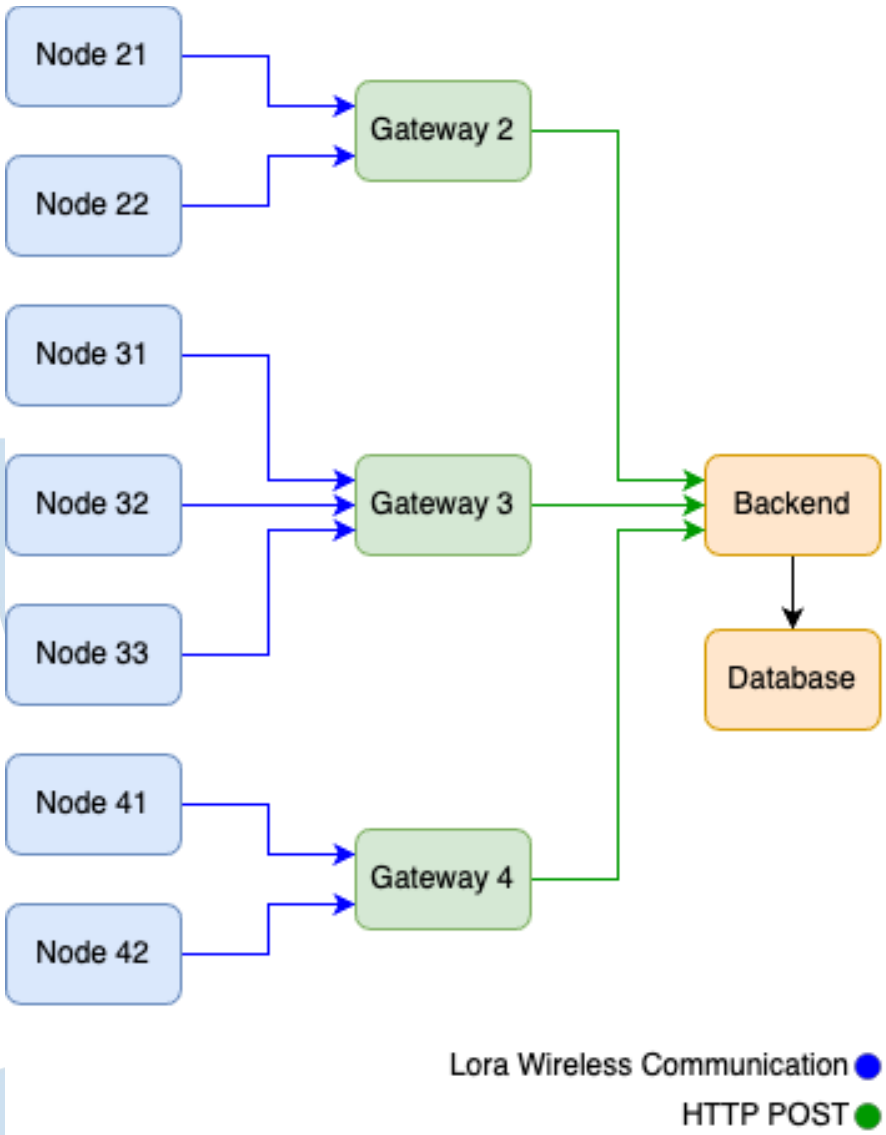
3.2 Studi Literature

Pada tahapan studi literatur, penulis mempelajari dan melakukan *brainstorming* tentang bagaimana sistem monitoring yang cocok untuk diimplementasikan pada perangkat IoT yang telah dirancang sebelumnya. Setelah itu penulis melakukan research dan melihat banyak dokumentasi tentang materi yang akan diperlukan seperti pemanfaatan konduktivitas pada besi dan menggunakan *Firebase Cloud Messaging*.

3.3 Perancangan Sistem

Pada pengimplementasian perangkat IoT, tim MySalak membuat sebuah sistem untuk mendapatkan data dari perangkat IoT tersebut. Sistem ini terdiri dari 3 bagian yaitu *Node*, *Gateway* dan *Backend* (Server). *Node* merupakan perangkat IoT yang terdiri dari 3 sensor yaitu DFRobot SHT20 untuk mendapatkan data suhu dan kelembapan, DFRobot Ambient Light Sensor untuk mendapatkan data intensitas cahaya dan *Water Tipping Bucket* untuk mendapatkan data curah hujan. Selain itu, *Node* juga menggunakan *LoRa Module* untuk komunikasi (*Bandwidth*: 125 kHz, *Coding Rate*: 4/8, *Spreading Factor*: 12), Baterai 18650 sebagai sumber power, *Solar Panel* sebagai pengisi daya baterai dan *Solar Panel Manager* sebagai pengatur daya pada *Node*. *Gateway* merupakan alat yang digunakan sebagai *receiver* agar data dari *Node* dapat dikirimkan ke *database* di *backend*. *Backend* digunakan untuk menyimpan dan mengolah data sensor yang diperlukan.

Sistem ini bekerja dengan Node yang secara otomatis mengambil data dari tiga sensor (DFRobot SHT20, DFRobot Ambient Light Sensor, dan Water Tipping Bucket) setiap satu jam sekali. Kemudian, data yang dihasilkan oleh sensor dikirimkan ke Gateway melalui komunikasi nirkabel menggunakan modul LoRa. Setelah Gateway menerima paket data dari Node, data tersebut diteruskan ke Backend melalui HTTP POST yang mengirimkan payload berupa JSON. Di Backend, data yang diterima disimpan ke dalam database agar kemudian dapat diolah dan digunakan sesuai kebutuhan analisis atau aplikasi yang dirancang. Sistem ini terdiri dari 7 Node, yaitu Node 21, 22, 31, 32, 33, 41 dan 42, yang mengirimkan data ke 3 Gateway, yaitu Gateway 2, 3 dan 4. Berikut arsitektur dari sistem yang telah diimplementasikan:

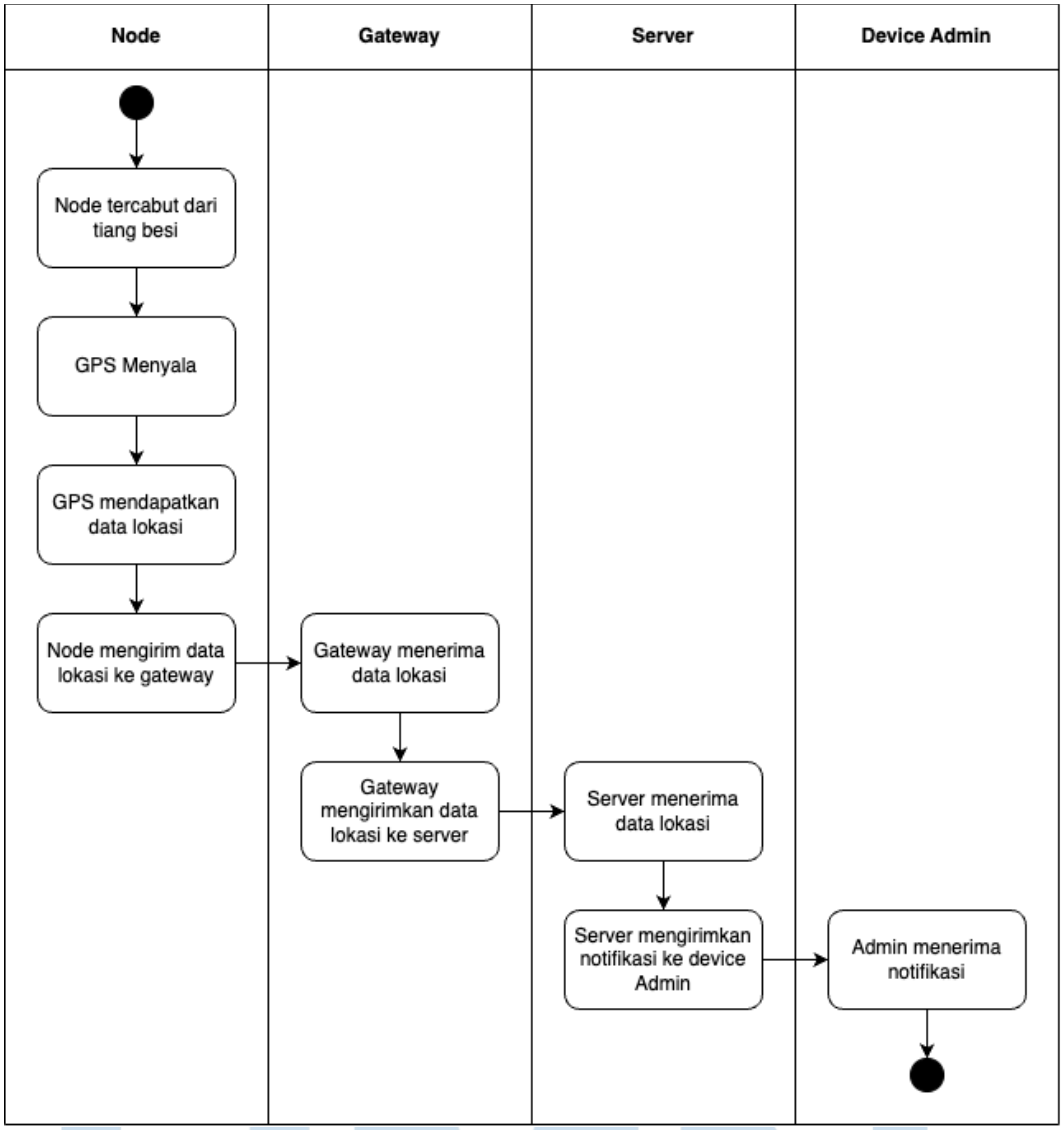


Gambar 3.2 Arsitektur Sistem Perangkat IoT MySalak

Pada tahapan ini, penulis akan merancang sebuah sistem yang berfungsi untuk memonitor perangkat IoT yang telah dipasang di perkebunan salak di Sleman, Yogyakarta. Perangkat IoT dipasang pada sebuah tiang besi seperti pada gambar 3.3.

Gambar 3.3 Perangkat IoT yang telah dipasang di Sleman, Yogyakarta

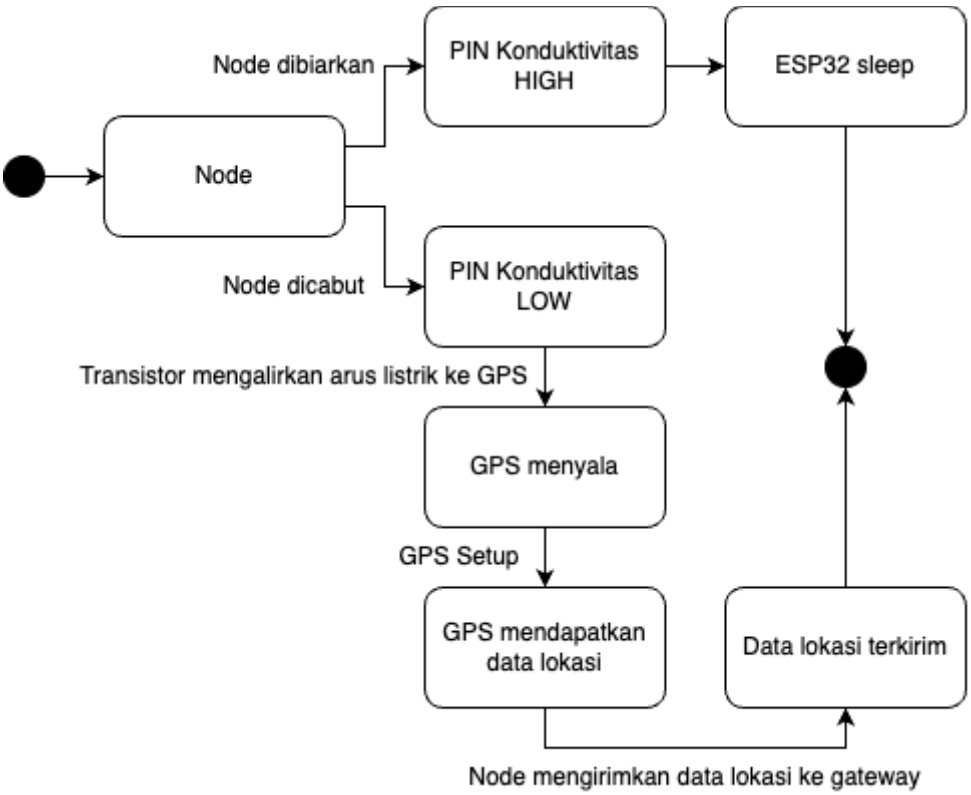
Sistem ini memanfaatkan konduktivitas pada tiang besi sehingga dapat mengecek apakah perangkat IoT berada di tempat awal pemasangannya. Jika perangkat IoT tercabut dari tiang besi, maka sistem akan mengirimkan notifikasi sebagai early warning kepada admin (Tim EPICS IEEE UMN, Tim EPICS IEEE UGM dan ketua kelompok tani) melalui aplikasi MySalak. Berikut rancangan sistem yang akan diimplementasikan terlihat pada gambar 3.3.



Gambar 3.4 Perancangan Sistem Monitoring

3.4 Perancangan Perangkat Keras

Pada tahapan pengembangan perangkat keras, penulis akan memodifikasi perangkat IoT yang telah dibuat sebelumnya karena perangkat IoT sebelumnya belum ada mekanisme otomatis untuk memastikan perangkat IoT tetap berada pada posisi yang telah ditentukan dan tidak dilengkapi dengan modul GPS sehingga lokasi perangkat IoT tidak dapat dilacak. Modul GPS yang akan digunakan adalah Ublox NEO 8M GPS Module. Proses perancangan sistem ini dirangkum dalam flowchart seperti yang terlihat pada gambar 3.4.

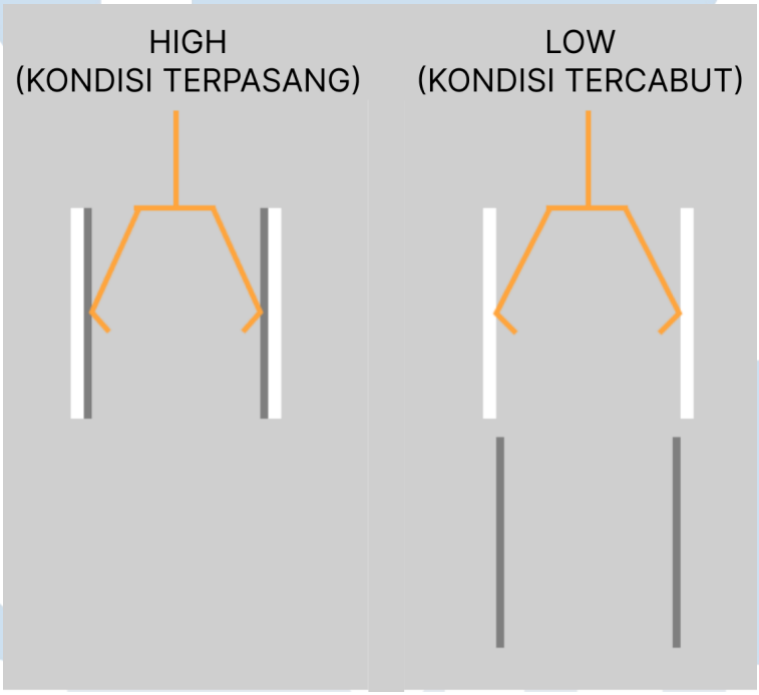


Gambar 3.5 Perancangan Perangkat Keras

Perancangan mekanisme pengecekan perangkat IoT otomatis ini dilakukan dengan memanfaatkan sifat konduktivitas material besi dan penerapan metode *falling edge*. Selain itu, pemilihan material besi yang digunakan juga perlu diperhatikan untuk memastikan keandalan sistem dalam jangka panjang. Material yang dipilih harus memiliki beberapa properti penting, yaitu memiliki ketahanan yang kuat terhadap karat, memiliki kekuatan mekanis yang cukup agar tidak mudah rusak, serta memiliki daya hantar listrik yang tinggi untuk memastikan deteksi perubahan sinyal dapat berjalan dengan optimal. Penulis memilih material berupa kuningan karena material yang digunakan akan konstan dialirkan arus listrik sehingga harus memiliki daya hantar listrik yang kuat dan tidak mudah terkena karat karena material kuningan sering digunakan pada stop kontak.

Mekanisme ini dirancang dengan cara membuat 2 pin besi yang bentuk sedemikian rupa (seperti pin pada stop kontak) agar kedua pin besi dapat bersentuhan dengan tiang besi sehingga memungkinan adanya aliran konduktivitas melalui tiang besi tersebut.

Kemudian metode falling edge akan digunakan untuk mendeteksi perubahan kondisi konduktivitas yang terjadi pada kedua pin besi dan tiang besi tersebut. Dengan menggunakan metode falling edge, mikrokontroler dapat mendeteksi apakah perangkat IoT masih terpasang pada tiang besi. Jadi jika perangkat IoT terpasang dan kedua pin besi bersentuhan pada tiang besi maka mikrokontroler akan mendeteksi sinyal *HIGH*, namun ketika perangkat IoT dicabut dari tiang besi maka mikrokontroler akan mendeteksi sinyal LOW seperti yang terlihat pada gambar 3.5.



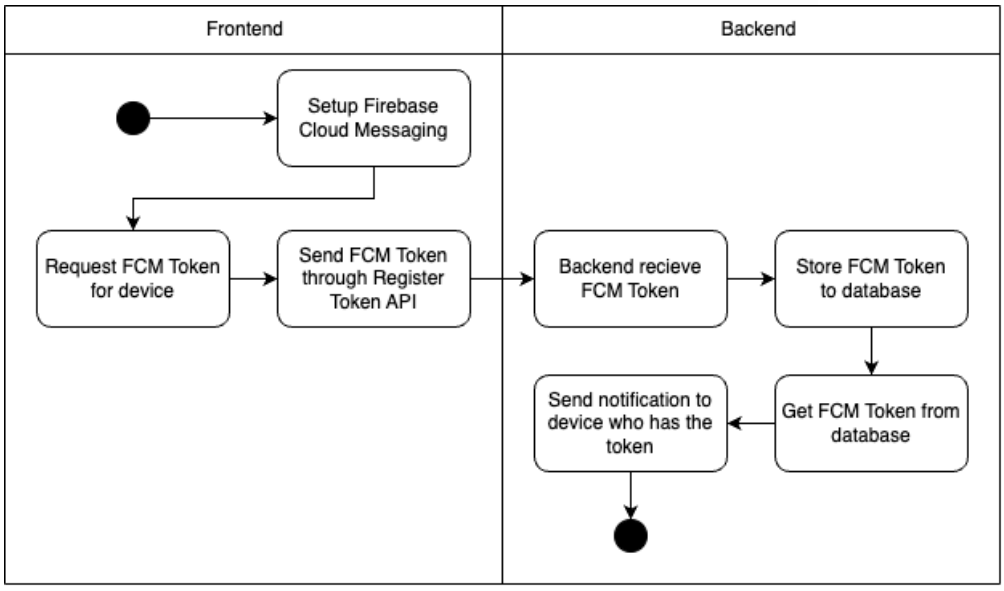
Gambar 3.6 Simulasi Perancangan Metode Falling Edge

Untuk mengoptimalkan efisiensi energi, modul GPS hanya akan dialiri arus listrik ketika mikrokontroler mendeteksi adanya perubahan sinyal dari pin konduktivitas. Hal ini dapat dicapai dengan menggunakan transistor sebagai saklar yang mengatur arus listrik yang masuk ke modul GPS. Ketika mikrokontroler mendeteksi sinyal LOW, transistor akan aktif dan mengalirkan

arus listrik mengalir ke modul GPS. Sebaliknya, jika perangkat IoT masih terpasang dengan benar dan sinyal konduktivitas tetap HIGH, transistor akan berada dalam kondisi cut-off sehingga modul GPS akan dalam keadaan mati total untuk menghemat daya.

3.5 Perancangan Perangkat Lunak

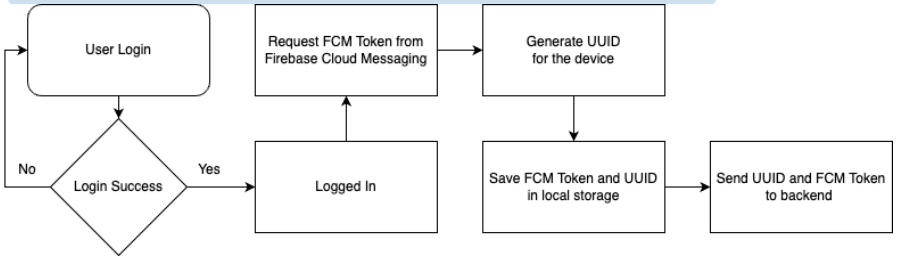
Pada tahapan pemgembangan perangkat lunak terdiri dari beberapa proses yaitu pengembangan *frontend* dan pengembangan *backend*. Untuk mengirimkan notifikasi ke sebuah *device* harus dilakukan setup pada *frontend* dan *backend*. Pertama-tama, device harus melakukan request token dari Server Firebase. Setelah mendapatkan response berupa FCM Token, token tersebut perlu dikirimkan dan disimpan di *backend*. Lalu jika FCM Token sudah disimpan di *backend*, notifikasi dapat dikirimkan dengan memasukkan payload berupa *title*, *message*, dan FCM Token pada function *send notification* dari Firebase. FCM Token diperlukan agar server Firebase mengetahui device tujuan yang akan dikirimkan notifikasi tersebut. Berikut diagram proses perancangan perangkat lunak yang akan diimplementasikan seperti yang terlihat pada gambar 3.6.



Gambar 3.7 Perancangan Perangkat Lunak

3.5.1 Perancangan *Frontend*

Perancangan *Frontend* pada sistem ini menggunakan framework React JS dengan menggunakan *Progressive Web Apps* (PWA) sehingga dapat diinstall sebagai aplikasi *native* pada mobile. Aplikasi ini akan digunakan oleh petani salak dan admin yang mana tampilannya akan terbagi menjadi 2 yaitu tampilan aplikasi MySalak dan tampilan MySalak For Admin. Pada tampilan *dashboard* admin akan dikembangkan agar admin dapat memantau atau memonitor setiap perangkat IoT yang telah dipasang di lahan perkebunan salak di Kab. Sleman, Yogyakarta. Berikut diagram proses perancangan *frontend* yang akan diimplementasikan seperti yang terlihat pada gambar 3.7.



Gambar 3.8 Perancangan Frontend

Pada *frontend*, user pertama-tama harus melakukan login terlebih dahulu untuk mengakses aplikasi MySalak. Lalu jika user berhasil masuk, *frontend* akan merequest FCM Token dari Firebase yang berfungsi untuk menghasilkan *client token* bagi setiap device pengguna. *Token* ini akan digunakan sebagai identifier unik, yang memungkinkan Firebase mengirimkan notifikasi secara spesifik ke device yang dituju melalui aplikasi MySalak. Kemudian, *frontend* juga akan mengenerate UUID (*Universally Unique Identifier*) yang akan digunakan sebagai *indentifier* untuk setiap device *user*. FCM Token dan UUID nantinya akan disimpan pada local storage dan dikirimkan ke *backend* sehingga dapat mengirimkan notifikasi kepada *user*.

3.5.2 Perancangan *Backend*

Pada perancangan *backend*, penulis menggunakan framework Express.js yang menggunakan bahasa pemrograman Javascript dan Sequelize sebagai ORM (*Object Relational-Mapping*). Penggunaan *backend* pada sistem ini adalah untuk menerima, menyimpan dan mengolah data yang diperlukan yaitu FCM token dan UUID. Selain itu, *backend* akan digunakan untuk mengirimkan notifikasi melalui library Firebase. Penulis menggunakan *Firebase Cloud Messaging* sebagai *push notification* karena Firebase merupakan layanan notifikasi gratis dan memiliki banyak dokumentasi yang tersedia serta menjamin keamanan yang tinggi karena dikelola langsung oleh Google. Berikut beberapa *table* dan API yang akan dibuat pada perancangan *backend*:

1. *Table Authorization* (Admin, Petani, Role)
2. *Table Push Notification Token*
3. API *Authorization* (Login dan Logout)
4. API *Notification Management* (Register, Delete, Send Notification)

Langkah awal yang akan dilakukan pada perancangan backend adalah membuat sistem authorization pada aplikasi MySalak untuk membedakan user sesuai dengan role masing-masing. Selanjutnya, penulis akan membuat model menggunakan ORM Sequelize yang nantinya akan menjadi table yang digunakan untuk menyimpan token.

Lalu penulis akan membuat POST API yang bertujuan untuk menyimpan token agar dapat mengirim notifikasi berdasarkan token dan POST API untuk menghapus token yang diperlukan jika *user* melakukan *logout* pada aplikasi MySalak, hal ini bertujuan agar notifikasi tidak terkirim pada user yang telah logout (tidak terverifikasi). Selanjutnya, penulis akan membuat sebuah POST API

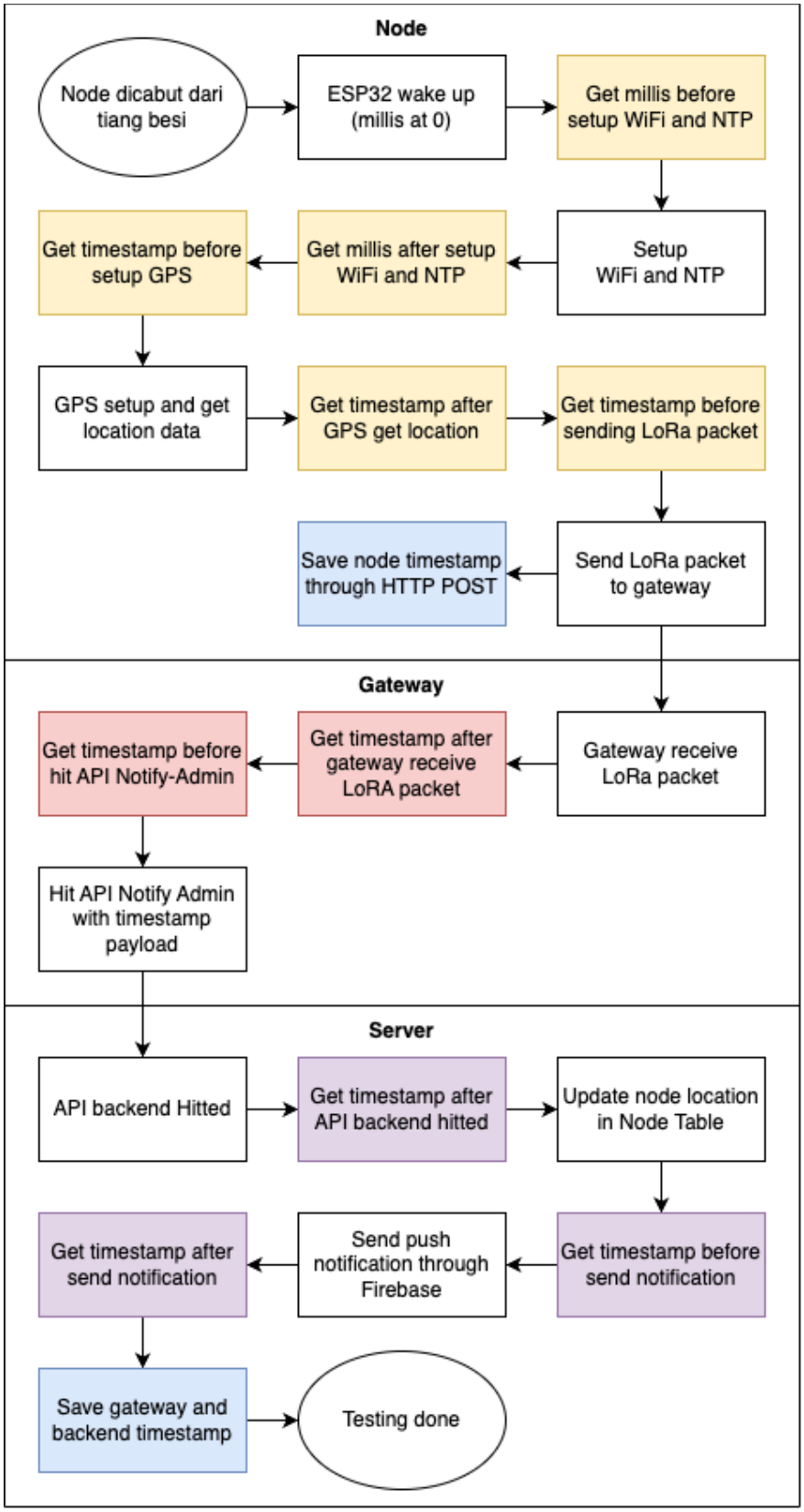
untuk mengirimkan notifikasi ke device *user*. API ini akan digunakan oleh gateway untuk memberitahu sistem bahwa ada perangkat IoT yang telah tercabut dari tiang besinya.

3.6 Metode Evaluasi

Pada penelitian ini, sistem monitoring perangkat IoT akan dilakukan pengujian untuk mengukur performa dan respons sistem dalam beberapa kondisi. Tujuan pengujian adalah untuk menguji sistem saat perangkat IoT dicabut dari tiang besi dan melihat apakah waktu yang diperlukan sistem untuk mengirim notifikasi sesuai yang diharapkan. Waktu ini ditentukan dari penelitian sebelumnya dimana GPS setup time sekitar 7 - 12 detik, pengiriman LoRa dibawah 1 detik dan pengiriman notifikasi sekitar 2 detik. Maka waktu yang diharapkan adalah dibawah 15 detik. Pengujian dilakukan dengan dua kondisi, yaitu:

1. Pengujian pencabutan perangkat IoT di lokasi pemasangan tiang besi yang akan dilakukan 5 kali setiap perangkat IoTnya.
2. Pengujian pencabutan perangkat IoT yang kemudian dibawa dengan cara berjalan secara santai dengan kecepatan rata-rata yaitu 4 km per jam.

Pengujian dilakukan dengan mengambil beberapa data timestamp dari node, *gateway* dan *backend*. Data timestamp tersebut dikirimkan dengan cara HTTP POST sehingga memerlukan setup WiFi dan NTP (*Network Time Protocol*). Berikut ini alur flowchart dalam melakukan pengujian seperti gambar 3.8.



Gambar 3.9 Flowchart Proses Testing

Berikut adalah data yang akan dikumpulkan setelah melakukan testing:

1. Keberhasilan trigger yang memanfaatkan konduktivitas pada besi di perangkat IoT dan tiang besi. Data ini didapatkan dengan menjumlahkan berapa kali trigger dapat bekerja untuk mengaktifkan GPS.
2. Waktu yang diperlukan GPS untuk mendapatkan data lokasi, data ini didapatkan dengan mengukur waktu pada saat pertama kali GPS menyala hingga GPS mendapatkan data lokasi. Pengukuran dilakukan dengan menggunakan NTP (*Network Time Protocol*) pada ESP32 untuk melihat kapan waktu GPS menyala dan kapan waktu GPS berhasil mendapatkan data lokasi. Kedua data tersebut kemudian dikurangi sehingga didapatkan waktu yang diperlukan GPS.
3. Waktu yang diperlukan setiap perangkat IoT untuk mengirim data lokasi ke gateway, data ini didapatkan dengan mengukur waktu sebelum melakukan pengiriman paket LoRa hingga paket LoRa diterima oleh gateway. Pengukuran dilakukan dengan menggunakan NTP (*Network Time Protocol*) dari ESP32 untuk melihat waktu sebelum paket LoRa dikirimkan dan waktu setelah paket LoRa diterima oleh *gateway*. Kedua data tersebut kemudian dikurangi sehingga didapatkan waktu pengiriman paket LoRa dari perangkat IoT ke gateway.
4. Waktu yang diperlukan gateway untuk mengakses API *backend* setelah data LoRa diterima, data ini didapatkan dengan mengukur waktu pada saat paket LoRa sampai pada gateway sampai sebelum gateway mengakses API *backend*. Pengukuran dilakukan dengan NTP untuk melihat waktu sebelum gateway mengakses API *backend* dan waktu kapan backend API *backend* terhit. Kedua data tersebut kemudian dikurangi sehingga didapatkan waktu yang diperlukan gateway untuk mengakses API.
5. Waktu yang diperlukan oleh server untuk mengirimkan notifikasi setelah API diakses, data ini didapatkan dengan cara melihat timestamp sebelum memanggil function *send notification* dari Firebase dan timestamp sesudah memanggil function *send notification*. Pengambilan timestamp dilakukan dengan menggunakan *function* date pada Javascript untuk mendapatkan

waktu server. Kemudian kedua data tersebut dikurangi sehingga didapatkan waktu yang diperlukan oleh server untuk mengirimkan notifikasi setelah API diakses.

6. Waktu yang diperlukan sistem untuk mengirimkan notifikasi setelah perangkat IoT dicabut dari tiang besi. Data ini didapatkan dengan cara mengurangi timestamp setelah notifikasi terkirim dan timestamp pertama kali node menyala lalu dikurangi waktu setup WiFi dan NTP (hal ini dilakukan karena pada kondisi aslinya tidak memerlukan WiFi dan NTP).