IMPLEMENTASI GOOGLE MAPS API PADA IOT PLATFORM UNTUK PELACAK SUATU OBJEK MENGGUNAKAN GPS

GOOGLE MAPS API IMPLEMENTATION ON IOT PLATFORM FOR TRACKING AN OBJECT USING GPS

Achmad Mustofa Luthfi[1], Dr. Nyoman Bogi A. K., S. T., MSEE[2], Ratna Mayasari, S. T., M. T.[3]
[1,2,3] Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[1]**mustofaluthfi@student.telkmouniversity.ac.id**, [2]**aditya@telkomuniversity.ac.id**,
[3]**ratnamayasari@telkomuniversity.ac.id**

Abstrak

Di era modern ini tentunya sudah tidak asing lagi dengan yang namanya *Global Positioning System* (GPS). Dimana untuk mengetahui suatu letak atau posisi dari suatu objek hanya diperlukan GPS, yang mana dari GPS tersebut mengirimkan titik koordinat yang kemudian dari titik koordinat tersebut dapat diketahui lokasinya, dengan memanfaatkan fitur geolokasi tentunya. GPS dapat dimanfaatkan untuk perangkat cerdas *Internet of Things* (IoT) sehingga dapat digunakan untuk mengetahui tempat suatu objek dengan menggunakan GPS dan mikrokontroler.

GPS akan digunakan sebagai penentu lokasi yang akan terintegrasi dengan NodeMCU sebagai *processor*-nya. Pada penelitian sebelumnya IoT *platform* masih belum memiliki fitur Google Maps untuk menampilkan lokasi dari objek tersebut, maka dilakukanlah implementasi Google Maps API pada IoT *platform* tersebut. Pada sistem modul GPS yang aktif akan mengirimkan titik koordinat. Pada IoT *platform* operator dapat melihat lokasi dari objek dan juga dapat mengetahui informasi yang terdapat pada IoT *platform* tersebut seperti nilai sensor suhu, temperatur dan lainnya.

Hasil dari pengujian alat yang terdiri dari NodeMCU sebagai *processor*, DHT-11 sebagai sensor suhu, dan modul GPS Neo-6m sebagai penentu lokasi maka didapatkan hasil sebagai berikut. Pembacaan modul GPS Neo-6m memiliki perbedaan ±0,000003 berdasarkan pembacaan titik koordinat pada Google Maps secara langsung, ataupun dengan *mobile phone*. Hasil *delay* pada kondisi jaringan baik didapatkan rata-rata *delay* yaitu 0,326 detik, serta didapatkan nilai *throughput* pengiriman data menggunakan NodeMCU dari pengujian tersebut adalah 140,4 *Bytes/s* dalam selang waktu 150 detik.

Kata kunci : NodeMCU, IoT platform, GPS, Google Maps API

Abstract

In this modern era, of course already familiar to what is called the Global Positioning System (GPS). Where to find out the location or position of an object GPS is only needed, which of the GPS sends a coordinate point which is then located from the coordinate point, using the geolocation feature of course. GPS can be used for intelligent Internet of Things (IoT) devices so that it can be used to track an object using GPS and a microcontroller.

GPS will be used to determine the location that will be integrated with RFID, and NodeMCU as the processor. In previous research IoT platforms still did not have the Google Maps feature to display the location of the object to be tracked, so the implementation of the Google Maps API on IoT platform was carried out. In this system RFID detects cards that have been registered before, then the GPS module will be active when RFID detects the registered card, the GPS module will send coordinates. On IoT platforms operators can see the location of objects and can also find out information from those using objects based on RFID that have been detected before.

The results of the testing tool consisting of NodeMCU as a processor, DHT-11 as a temperature sensor, and GPS module Neo-6m as a location determinant, the results are as follows. The reading of the GPS Neo-6m module has a difference of ± 0.000003 based on reading coordinates on Google Maps directly, or with a mobile phone. The results of delay on good network conditions obtained an average delay of 0.326 seconds, and the data transmission throughput value obtained using NodeMCU from the test is 140.4 Bytes / s in an interval of 150 seconds.

Keywords: NodeMCU, IoT platform, GPS, Google Maps API

1. Pendahuluan

IoT *platform* merupakan sebuah teknologi *multi-layer* yang memungkinkan untuk langsung me-manajemen, mengontrol dan otomatisasi segala perangkat yang terhubung dengan IoT itu sendiri. Pada dasarnya IoT *platform* ini menghubungkan perangkat keras yang bermacam-macam ke *cloud* atau ke server dengan menggunakan konektivitas internet sehingga memiliki pemrosesan data yang luas dan mudah di akses. Secara garis besar IoT

platform bisa dalam tampilan web maupun aplikasi [1]. Saat ini beberapa IoT *platform* sebenarnya masih perlu pengembangan untuk mewujudkan IoT *platform* yang lebih baik lagi, namun dalam mewujudkan itu juga terdapat beberapa kendala dari segi infrastruktur jaringan yang masih terbatas [2].

Pada penelitian sebelumnya IoT *platform* masih belum memiliki fitur menampilkan lokasi menggunakan Google Map. Sehingga pada tugas akhir ini penulis akan mengembangkan *platform* pada penelitian sebelumnya yaitu, pengimplementasian Google Maps untuk *platform* agar *platform* tersebut bisa digunakan untuk mengetahui lokasi yang mampu memenuhi kebutuhan pengguna. Berikut adalah beberapa orang yang telah melakukan penelitian terkait implementasi Google Maps API, seperti "*Design and implementation of an accurate real time GPS tracking System*" [3], "*Real Time Vehicle Tracking System Based on ARM7 GPS and GSM Technology*" [4]. "Implementasi Metode Link Aggregation Dengan Mode Balance-RR Pada Operational Server Berbasis IOT Platform" [5].

Pada penelitian ini akan menggunakan metode *Extreme Programming* (XP) untuk pengimplementasian Google Maps pada IoT *platform*. Metode ini memiliki beberapa tahapan yang dilakukan yaitu *planning, design, coding, testing* [6]. Dengan dikembangkannya IoT *platform* ini penulis dapat memanfaatkan geolokasi untuk mengetahui lokasi sebuah kendaraan berdasarkan titik koordinat. Yang mana titik koordinat tersebut dikirimkan dengan menggunakan perangkat NodeMCU dan sensor GPS, dan kemudian dimunculkan ke web pada IoT *platform* yang sudah diimplementasikan Google Maps secara *real-time*.

2. Konsep Dasar

2.1 IOT Platform

Sistem *Internet of Things* (IoT) merupakan alat-alat seperti sensor, *processor* yang akan menghubungkan perangkat dan juga sensor yang sudah terintegrasi hingga ke internet. IoT *platform* merupakan sebuah layanan yang berperan sebagai penghubung antara sensor dan *processor* dengan web ataupun aplikasi *mobile* yang terhubung ke server [1]. Contoh IoT *platform* yang sudah sering dijumpai adalah Geeknesia, Antares, dan Thinkspeak, dll. Yang mana masing-masing dari *platform* tersebut memiliki kelebihan dan kelemahannya.

Saat ini IoT *platform* yang dikembangkan merupakan IoT *platform* yang telah dibuat sebelumnya dan masih dalam tahap pengembangan. Pada IoT *platform* ini alat-alat IoT sudah bisa terhubung dan bisa digunakan secara langsung bagi yang ingin membuat alat berbasis IoT. IoT *platform* ini berfungsi sebagai suatu layanan berbasis website yang bisa digunakan untuk memantau kondisi dengan menggunakan sensor maupun mengontrol alat-alat tersebut dengan mengakses *website*. Seperti pada Gambar 2.1 merupakan tampilan awal dari IoT *platform*. IoT *platform* sendiri memiliki 3 bagian utama dalam sistem kerjanya yaitu:

2.1.1 *Server*

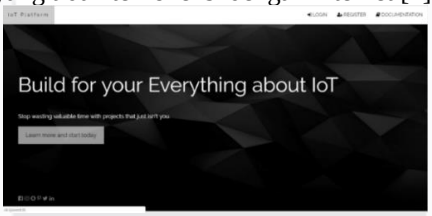
Server merupakan perangkat yang berfungsi untuk menyimpan keseluruhan trafik data maupun *request*, dan juga berfungsi sebagai pengatur juga menganalisa dari sistem IoT *platform*. *Server* sendiri terdiri dari *storage* atau berupa *database*, *Application Programming Interface* (API) *libraries*, dan *management server* [1].

2.1.2 *Network Gateway Devices*

Network Gateway Devices terdiri dari *security*, *Applicaton Programming Interface* (API) *libraries*, dan *management gateway*. *Network Gateway Devices* berfungsi sebagai penghubung antara *things* dengan *third party cloud*, *network interface*, dan *server* [1].

2.1.3 *Things*

Things atau biasa disebut sebagai pengguna (*end user*) atau juga *devices* yang terkoneksi dengan internet, dan terhubung dengan IoT *platform* tersebut. Pada *things* juga terdapat *Application Programming Interface* (API) *libraries*, *management end user*, dan juga seluruh perangkat atau sensor-sensor yang sudah terintegrasi dan terkoneksi dengan internet maupun yang tidak terkoneksi dengan internet [1].



Gambar 1. Tampilan awal iot platform.
Sumber: www.skripsweet.tk

2.2 Keterangan Tabel dan Gambar

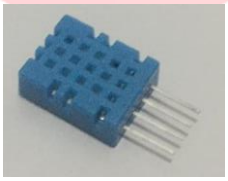
Mikro WiFi NodeMCU merupakan salah satu *board* yang sudah *compatible* dengan arduino yang digunakan dan dirancang untuk kebutuhan IoT. Mikro Wifi NodeMCU ini menggunakan SoC WiFi yaitu ESP8266. Sudah *compatible* dengan arduino ini adalah mikrokontroler ini dapat diprogram menggunakan *software* Arduino IDE, dengan menggunakan *syntax* dan *library* khusus NodeMCU itu sendiri. Mikro WiFi ini perangkat yang dapat terhubung ke internet karena didalamnya sudah terdapat modul WiFi sehingga dapat langsung digunakan ketika terdapat koneksi internet berupa WiFi [7].

Gambar 2. Nodemcu.

2.3 Sensor Suhu DHT-11

Sensor DHT-11 adalah sensor yang dapat mengukur dua parameter lingkungan, yaitu suhu dan kelembaban udara. Sensor ini tergolong komponen yang memiliki tingkat stabilitas yang baik, dan juga memiliki respon pembacaan data yang cepat. Sensor DHT-11 ini memiliki ukuran yang relatif kecil sehingga cocok digunakan untuk alat-alat yang minimalis [8].

Sensor ini akan dihubungkan dengan NodeMCU sebagai *processor* untuk mengolah data dari sensor ini. Yang kemudian data tersebut akan dikirimkan ke IoT *platform* untuk ditampilkan. Sensor ini memiliki kemampuan transmisi sinyal hingga 20 meter, sehingga sangat cocok untuk mendeteksi keadaan dalam suatu ruangan. Berikut merupakan gambar dari sensor DHT-11 [8].



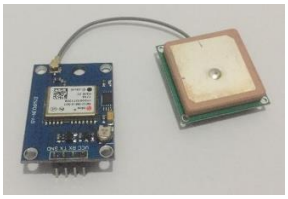
Gambar 3. Sensor dht-11.

2.4 Google Maps API

Google Maps merupakan sebuah layanan peta seluruh dunia yang bisa diakses seacara online dan gratis yang disediakan oleh Google. Layanan ini memberikan beberapa *view* berupa peta jalan, satelit, kondisi lalu lintas, dan pencarian rute dalam berpergian baik menggunakan kendaraan ataupun tidak. Google sendiri sebenarnya memberikan layanan berupa Google Maps API yang bisa digunakan untuk dimasukkan kedalam situs web pihak ketiga yang mana bisa dimanfaatkan untuk kepentingan web itu sendiri. Google Maps API merupakan suatu aplikasi *interface* yang dapat diakses menggunakan *javascript*, python, html, dan sebagainya, sehingga Google Maps dapat ditampilkan. Sebelumnya juga perlu dilakukan pendaftaran untuk mendapatkan API *key* agar bisa menggunakan Google Maps API tersebut [9].

2.5 GPS

Global Positioning System (GPS) merupakan sistem navigasi atau penentuan lokasi. Sistem GPS sendiri digunakan untuk memperoleh lokasi dan juga bisa berupa kecepatan serta memberikan informasi waktu, secara terus menerus tanpa bergantung pada waktu dan cuaca tertentu. Lokasi pada GPS sendiri dinyatakan dalam suatu titik koordinat yang mana nantinya titik koordinat tersebut dapat dimanfaatkan untuk menampilkan lokasi dimana keberadaan GPS tersebut [10].



Gambar 4. Modul GPS NEO-6M

2.5 Wireshark

Wireshark merupakan suatu perangkat lunak untuk menganalisa paket-paket dalam suatu jaringan dan menampilkan paket-paket tersebut secara detail, dan Wireshark ini bersifat *open-source*. Wireshark sendiri memiliki beberapa fungsi diantaranya yaitu dapat memeriksa keamanan jaringan, dapat digunakan untuk men-debug implementasi protokol, dan juga dapat mempelajari tentang protokol TCP/IP [11].

2.6 Parameter Pengujian

2.6.1 Akurasi

Akurasi merupakan kedekatan suatu nilai yang diukur dengan nilai yang sebenarnya, semakin dekat nilai nya maka akan semakin akurat. Selain akurasi terdapat satu lagi yang disebut presisi yang mana ketika diambil beberapa sampel nilai dan dilakukan secara berulang. Tingkat presisi suatu data dapat dilihat dari hasil deviasi dari pengukuran. Dan juga terdapat nilai koefisien variansi yang berfungsi untuk menentukan nilai *error* pada data tersebut dan tentunya berhubungan dengan tingkat akurasi [12]. Berikut merupakan rumus untuk menentukan akurasi suatu data:

Rumus Variansi:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1} \quad (1)$$

Rumus Standar Deviasi:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \quad (2)$$

Rumus Koefisien Variasi:

$$KV = \frac{s}{\bar{x}} \times 100\% \quad (3)$$

Keterangan:

s^2 = Varians
s = Simpangan Baku (Standar Deviasi)
KV = Koefisien Variasi
x_i = Nilai x ke-i
n = Ukuran Sampel
$\bar{x}$ = Rata-rata baca sensor

Koefisien keruncingan atau juga disebut sebagai koefisien kurtosis merupakan koefisien yang digunakan untuk mengetahui data uji tersebut homogen (mengelompok) atau tidak. Pada Gambar 2.6 merupakan koefisien keruncingan yang terbagi berdasarkan keruncingannya, kurva tersebut terbagi menjadi tiga macam yaitu:

1. Leptokurtik, merupakan kurva yang memiliki puncak relatif tinggi. Dimana nilai dari α4 lebih dari 3 (α4 > 3).
2. Platikurtik, merupakan kurva yang memiliki puncak hampir mendatar. Dimana nilai dari α4 lebih kecil dari 3 (α4 < 3).
3. Mesokkurtik, merupakan kurva yang memiliki puncak tidak tinggi dan tidak mendatar. Dimana nilai dari α4 sama dengan 3 (α4 = 3).

Rumus Koefisien Keruncingan:

$$\alpha_4 = \frac{\frac{1}{n}\sum(n-1)^4}{S^4} \quad (4)$$

Keterangan:
α_4 = Koefisien Keruncingan
n = Ukuran Sampel
s = Simpangan Baku (Standar Deviasi)

2.6.2 Delay

Delay merupakan waktu yang terjadi mulai dari pengiriman data dikirim sampai data tersebut diterima di tujuan. Pada suatu jaringan *delay* dapat menentukan kualitas suatu jaringan, yang mana semakin kecil nilai *delay* maka kualitas jaringan tersebut akan semakin bagus pula. Berikut merupakan rumus untuk menghitung delay [16]:

$$delay = waktu\ paket\ diterima - waktu\ paket\ dikirimkan \quad (4)$$

2.6.3 Throughput

Throughput merupakan kecepatan rata-rata yang diterima atau dikirimkan pada suatu titik dalam selang waktu tertentu. *Throughput* biasa disebut juga nilai *bandwidth* aktual saat sedang dilakukan koneksi. Nilai *throughput* juga dapat berpengaruh sewaktu-waktu apabila ada yang menghambat koneksi tersebut seperti *latency* dan sebagainya. Satuan dari *throughput* sendiri sama seperti *bandwidth* yaitu *Byte/s.* Berikut merupakan rumus untuk mecari nilai *throughput,* dan mengetahui berapa persen besaran paket yang terpakai terhadap *bandwidth* internet yang digunakan [13]:

$$Throughput = \frac{Jumlah\ data\ yang\ dikirim}{Lamanya\ waktu\ pengiriman\ data} \quad (5)$$

$$\%Terpakai = \frac{Throughput}{Bandwidth\ internet\ yang\ digunakan} \quad (6)$$

3. Model Sistem dan Perancangan

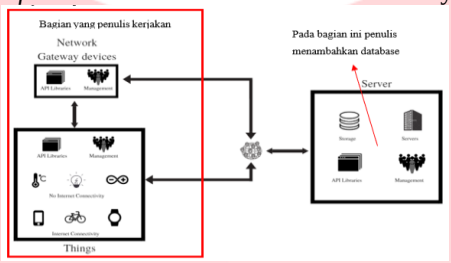
3.1. Desain Sistem

Pada IoT platform terdapat 3 bagian yang akan dikerjakan yaitu:

1. Things
   Pada bagian *things* dibuat perangkat keras sistem pelacakan yang sudah terintegrasi, mulai dari modul GPS neo-6, sebagai penentu lokasi, sensor suhu sebagai nilai sensor yang akan dikirimkan ke IoT *platform*, dan NodeMCU sebagai *processor.*
2. Network Gateway Devices
   Pada *Network Gateway Devices* diimplementasikan API Google Maps dengan cara mengkonfigurasi pada *backend* dan juga *frontend* agar dapat digunakan sebagai pelacak lokasi suatu objek.
3. Server

Pada server akan ditambahkan database untuk GPS yang nantinya akan diolah Google Maps API untuk menampilkan lokasi dari alat tersebut.

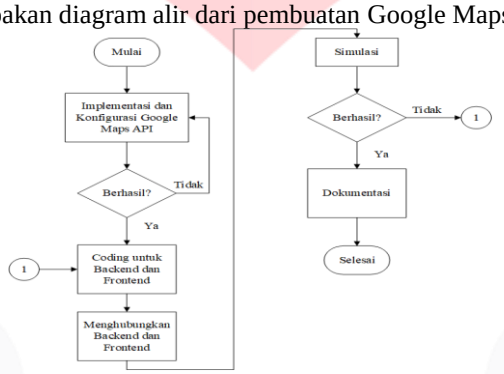
Dari tiap bagian sistem IoT *platform* sudah tersusun dalam blok sistem yang ditunjukkan pada Gambar 5.



Gambar 5. Blok sistem IoT platform

3.1.1 Perancangan Google Maps API Pada IOT Platform

Perancangan Google Maps API pada IoT *platform* yaitu dengan menambahkan *database* untuk menyimpan data koordinat yang akan dihubungkan dengan IoT *platform*. Kemudian akan dilakukan coding pada bagian *front-end* dan *back-end*. Berikut merupakan diagram alir dari pembuatan Google Maps API pada IoT *platform*.

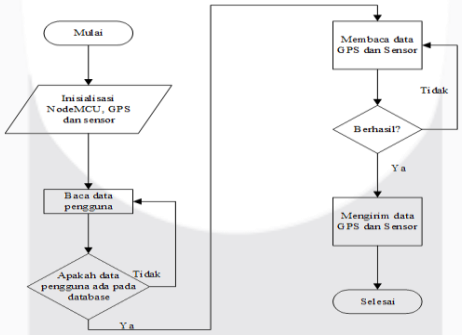


Gambar 6. Diagram alir implementasi Google Maps API

Dari Gambar 6 menunjukkan diagram alir dari pembuatan Google Maps API pada IoT *platform*, yang dimulai dengan menambahkan *script* Google Maps API *token* pada *platform*, dan dengan mendaftarkan API pada situs Google Maps Developers. Selanjutnya ketika sudah berhasil maka dilakukan *coding* pada bagian *back-end* yaitu dengan menambahkan kolom pada *database* untuk menyimpan data koordinat, lalu *coding* pada bagian *front-end* agar Google Maps dapat ditampilkan pada situs IoT *platform*. Selanjutnya adalah dengan menhubungkan antara *back-end* dengan *front-end*. Lalu dilakukan simulai yang memperlihatkan apakah koordinasi *back-end* dan *front-end* yang apabila berhasil dapat terlihat pada Google Maps dengan menampilkan titik koordinat.

3.1.2 Perancangan Sistem Pelacakan

Perancangan sistem pelacakan terdiri dari NodeMCU sebagai *processor* untuk mengolah data-data dari sensor dan modul yang digunakan. Modul GPS Neo-6m digunakan untuk mengirimkan informasi berupa titik koordinat berdasarkan lokasi sebenarnya menggunakan GPS. Berikut merupakan diagram alir dari perancangan sistem pelacakan.

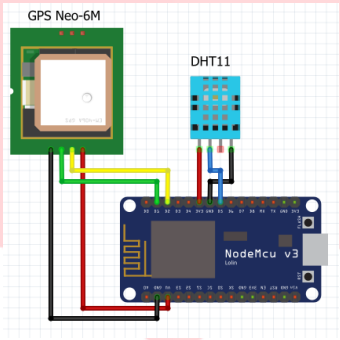


Gambar 7. Perancangan sistem pelacakan.

Dari Gambar 3.3 menunjukkan diagram alir dari perancangan sistem pelacakan yang menggunakan nodeMCU sebagai *processor* dan juga sensor untuk mengetahui perubahan fisik atau kimia. Sistem diawali dengan menginisialisasi nodeMCU agar dapat terhubung dengan IoT *platform* yang kemudian menginisialisasi modul GPS dan sensor. Kemudian data pengguna yang telah diinisialisasi pada nodeMCU diperiksa dengan data yang ada pada database. Apabila data pengguna tidak terdapat pada database maka sistem akan mengulang kembali untuk menginisialisasi data di nodeMCU. Apabila berhasil maka sensor dan modul GPS akan membaca perubahan data berupa titik koordinat pada saat itu juga. Apabila sistem berhasil membaca data perubahan maka data tersebut akan dikirimkan ke IoT *platform*.

3.2 Perangkat Keras

Dalam pembuatan sistem ini dibutuhkan beberapa perangkat keras agar dapat mendukung kinerja dari sistem, dan diharapkan berjalan dengan sesuai yang diharapkan. Gambar 3.4 merupakan rangkaian dari perangkat keras yang akan dibuat, perangkat keras terdiri dari sensor suhu dht-11, modul GPS neo-6m, dan nodeMCU sebagai *processor*.



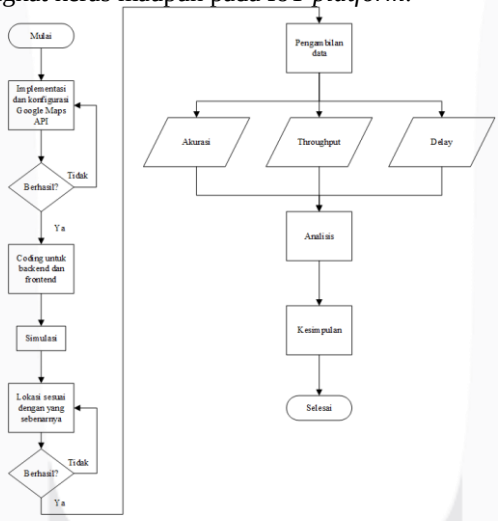
Gambar 8. Rangkaian skematik perangkat keras.

3.3 Pembuatan *Library* HTTP Pada IOT *Platform*

Pembuatan *library* HTTP pada IoT *platform* dilakukan agar tidak terjadi kesalahan ketika pengkodingan dan mempermudah pengkodingan pada perangkat keras nodeMCU dan sensor agar format yang telah diberikan oleh IoT *platform*. Sehingga akan lebih praktis ketika sudah menggunakan *library* ini.

3.4 Skenario Pengujian

Pada skenario pengujian memiliki tujuan yang akan dicapai dan sesuai dengan harapan sehingga dilakukan pengujian dari masing-masing perangkat dalam sistem. Apakah perangkat berjalan sesuai dengan baik atau tidak, serta menguji sistem secara keseluruhan. Pengujian yang dilakukan adalah simulasi dan pengujian pada perangkat keras, kemudian pengujian pada IoT *platform*. Sehingga pada simulasi deng pengujian tersebut dapat dilakukan pengambilan data baik dari perangkat keras maupun pada IoT *platform*.



Gambar 9. Diagram alir tahapan implementasi google maps API.

Pada proses implementasi Google Maps API dijelaskan secara singkat sesuai dengan Gambar 3.8. Tahap pertama yaitu dengan menyiapkan IoT *platform* agar bisa di implementasikan Google Maps API. Mengimplementasikan Google Maps API pada IoT platform yaitu dengan melakukan beberapa *coding* pada *backend* dan *frontend*. Setelah dilakukan *coding* maka dilakukan simulasi untuk melihat apakah Google Maps sudah berhasil diimplementasikan pada IoT *platform*. Dari hasil simulasi dapat dilihat apakah titik koordinat yang diberikan oleh perangkat keras yang dirangkai sudah sesuai dengan titik koordinat aslinya. Pengambilan data dilakukan untuk melihat kualitas dari sistem tersebut seperti akurasi ketepatan dari titik koordinat, *throughput*, dan *delay*. Selanjutnya adalah dengan melakukan analisis data untuk mendapatkan kesimpulan akhir.

4 Pengujian dan Analisis

Pengujian yang dilakukan pada Tugas Akhir kali ini dilakukan dua pengujian yaitu pengujian pada perangkat keras seperti sensor yang telah terhubung dengan nodeMCU dan juga modul GPS, yang kedua adalah pengujian pengiriman data dari perangkat keras ke IoT *platform*.

4.1 Pengujian Akurasi GPS

Pengujian akurasi GPS dilakukan untuk mengetahui seberapa akurat dari modul GPS neo-6m dengan keadaan lokasi yang sebenarnya. Berikut merupakan pengujian akurasi GPS berdasarkan lokasi yang sebenarnya didapat

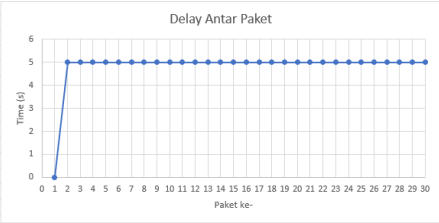
dari Google Maps dan dicocokkan dengan koordinat yang didapat modul GPS neo-6m. Dari beberapa percobaan data bahwa perbedaan koordinat tidak jauh berbeda yaitu dengan nilai standar deviasi *latitude* 0,0000031 dan *longitude* 0,0000035 yang berarti pada titik *latitude* dan *longitude* pembacaan modul GPS neo-6m memiliki perbedaan ±0,000003 berdasarkan pembacaan titik koordinat pada Google Maps secara langsung. Dan dapat dilihat bahwa tingkat akurasi hampir mencapai 100% yang mana dapat dilihat baik *latitude* dan *longitude* memiliki tingkat akurasi mencapai 99,99%.

4.2 Pengujian Delay

Pengujian *delay* merupakan perhitungan selisih waktu antara pengiriman data menggunakan nodeMCU sampai data diterima di server. Selanjutnya adalah dilakukan perhitungan *delay* pada saat nodeMCU berhasil mengirimkan data ke server sampai nodeMCU menerima respon balasan dari server. Dan yang terakhir adalah melakukan pengukuran *delay* pengiriman tiap paket dalam suatu waktu.

Gambar 10. *Delay respons data.*

Pada Gambar 10 merupakan delay respons data yang didapatkan ketika NodeMCU berhasil mengirimkan data ke server sampai NodeMCU menerima respon dari server. Pengujian tersebut dilakukan dalam 3 kondisi yaitu ketika jaringan dalam kondisi jaringan baik (biru), sedang (oranye), dan sibuk (abu-abu). Pengujian dilakukan dengan mengirimkan 30 data pada masing-masing kondisi dalam selang waktu tertentu. Pada kondisi jaringan baik didapatkan rata-rata delay yaitu 0,326 detik, pada kondisi jaringan sedang didapatkan rata-rata delay 0,472 detik, dan pada kondisi jaringan buruk didapatkan rata-rata delay 0,784 detik. Merujuk pada Tabel 2.1 maka delay pada kondisi jaringan baik termasuk kedalam standarisasi kategori cukup, sedangkan pada kondisi jaringan sedang dan sibuk masuk kedalam kategori buruk.



Gambar 11. *Delay* antar paket.

Setiap paket yang dikirmkan masing-masing memiliki jeda waktu 5 detik sama seperti yang sudah dikoding pada program nodeMCU, seperti pada Gambar 11 dapat dilihat dari 30 paket yang dikirimkan setiap paket memiliki delay antar paket yaitu 5 detik.

4.3 Pengujian Throughput

Pengujian throughput kali ini dilakukan pada 3 kondisi jaringan yaitu kondisi jaringan baik, sedang, dan sibuk dengan mengirimkan 30 kali percobaan menggunakan NodeMCU yang dikirimkan ke server, dan dengan bantuan software Wireshark untuk mengetahui nilai dari throughput. Dari total paket yang dikirimkan memiliki ukuran paket 21.060 Bytes dan dalam selang waktu 150 detik sehingga:

$$Throughput = \frac{Ukuran\ Paket}{Lama\ Waktu\ Pengiriman}$$
$$= \frac{21.060}{150} = 140{,}4\ Bytes/s$$
$$\%Terpakai = \frac{Throughput}{Bandwidth\ internet\ yang\ digunakan}$$
$$= \frac{0{,}0001404}{20} \times 100 = \ 0{,}0007\%$$

Didapatkan nilai throughput pengiriman data menggunakan NodeMCU dari pengujian tersebut adalah 140,4 Bytes/s dalam selang waktu 150 detik. Dan dari bandwidth yang disediakan 20 MBps pengujian ini hanya memakai 0,0007%. Dan pengukuran throughput tersebut diukur berdasarkan pengiriman data pada NodeMCU sehingga tidak akan mempengaruhi nilai throughput pada bandwidth berapapun.

4. Simpulan

Dari hasil pengujian sistem pengiriman data dari NodeMCU sampai ke IoT *platform* yang telah dilakukan pada Tugas Akhir ini, didapatkan beberapa kesimpulan sebagai berikut:

1. Untuk memperoleh lokasi dari suatu perangkat dilakukan dengan cara memanfaatkan modul GPS Neo-6m yang berfungsi untuk mendapatkan titik koordinat diintegrasikan dengan NodeMCU sebagai pengolah data.

2. Dilakukan pengkodingan *REST-API* pada *backend* IoT *platform* sehingga komunikasi antara GPS pada NodeMCU dapat terhubung dengan IoT *platform*.
3. Pembuatan *library* dilakukan sesuai dengan format yang diperlukan IoT *platform*, sehingga NodeMCU dapat mengirimkan informasi dan nilai sensor yang diperlukan oleh IoT *platform*, dan pada IoT *platform* dapat ditampilkan nilai dan informasi yang sudah dikirimkan.
4. Dilakukan pengkodingan pada *backend* IoT *platform* yang menggunakan *framework* Laravel untuk mengolah data berupa titik koordinat dan juga pada *frontend* sehingga pada tampilan web dapat dilihat menggunakan Google Maps.
5. Analisa performansi sistem kerja disimpulkan menjadi beberapa poin yaitu:
   a) Didapatkan pada perbandingan Neo-6m dengan Google Maps nilai koefisien keruncingan *latitude* 2,9 dan *longitude* 3,2. Pada perbandingan Neo-6m dengan *mobile phone* nilai koefisien keruncingan *latitude* 2,2 dan *longitude* 2,8. Maka pada koefisien keruncingan yang mendekati angka 3 membuktikan bahwa perbandingan data tersebut bersifat homogen atau data pada pembacaan sensor dengan data sebenarnya seragam.
   b) Dari 30 percobaan data dapat terlihat bahwa perbedaan koordinat tidak jauh berbeda seperti dapat dilihat pada nilai standar deviasi *latitude* 0,0000031 dan *longitude* 0,0000035 yang berarti pada titik *latitude* dan *longitude* pembacaan modul GPS Neo-6m memiliki perbedaan ±0,000003 berdasarkan pembacaan titik koordinat pada Google Maps secara langsung ataupun dengan *mobile phone*.
   c) *Delay* pada pengujian ini dilakukan dengan menghitung waktu mulai dari NodeMCU mengirimkan data hingga NodeMCU menerima respon dari server. *Delay* yang didapatkan yaitu 0,326 detik. Dengan nilai deviasi 0,064 yang berarti dari rata-rata *delay* yang didapatkan akan terjadi persimpangan ± 0,064 detik.
   d) Pada pengujian ini didapatkan nilai *throughput* 2.389 *Bytes/s* yang menggunakan *bandwidth* internet 40 MBps. Sehingga dari kapasitas jaringan yang disediakan pengujian ini hanya menggunakan sekitar 0,006%.

Daftar Pustaka:

[1] P. Jamborsalamati, E. Fernandez, M. J. Hossain and F. H. M. Rafi, "*Design and implementation of a cloud-based IoT platform for data acquisition and device supply management in smart buildings*," Australasian Universities Power Engineering Conference (AUPEC), Melbourne, VIC, pp.1-6, 2017.

[2] M. W. Habibi, A. Bhawiyuga and A. Basuki, "Rancang Bangun IoT Cloud Platform Berbasis Protokol Komunikasi MQTT." Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2(2), pp.479-485, 2017.

[3] H. A. Abdallah Dafallah, "*Design and implementation of an accurate real time GPS tracking System*," *Third International Conference on e-Technologies and Networks for Development (ICeND)*, pp.183-188, 2014.

[4] P. V. Mistary and R. H. Chile, "*Real time Vehicle tracking system based on ARM7 GPS and GSM Technology.*" *Annual IEEE India Conference (INDICON)*, pp.1-6, 2015.

[5] O. Rukmananda, "Implementasi metode link aggregation dengan mode balance-rr pada operational server berbasis iot platform", Telkom University, 2018.

[6] R. S. Pressman, Software Engineering: A Partitioner's Approach, Vol. 7, pp.72-79, 2010.

[7] M. Electronics, "NodeMCU V3 ESP8266 ESP-12E," Available at: https://www.makerlab-electronics.com/product/nodemcu-v3-esp8266-esp-12e/ [Accessed 3 Oktober 2018, 22:30:00 WIB]

[8] M. Y. E. Adiptya and H. Wibawanto, "Sistem Pengamatan Suhu dan Kelembaban Pada Rumah Berbasis Mikrokontroller ATmega8," Jurnal Teknik Elektro Vol. 5 No. 1, 2013.

[9] I. A. Gufroni, N. Hiron, A. N. Rachman and Y. A. Malik, "Implementasi Google Mpas API Dalam Aplikasi Mobile Penghitung Jarak Aman Dari Dampak Kemungkinan Letusan Gunung Galunggung," Seminar Nasional Aplikasi Teknologi Informasi (SNATI), 2013.

[10] E.Susanti and J. Triyono. "Pengembangan sistem pemantau dan pengendali kendaraan menggunakan raspberry pi dan firebase," Konferensi Nasional Teknologi Informasi dan Komunikasi (KNASTIK), 2016.

[11] S. Wang, D. Xu and S. Yan, "Analysis and application of Wireshark in TCP/IP protocol teaching," *2010 International Conference on E-Health Networking Digital Ecosystems and Technologies (EDT)*, Shenzhen, pp. 269-272, 2010.

[12] N. Kesumawati, A. Retta, dan N. Sari, "Pengantar Statistika Penelitian", pp.71-89, 2017

[13] R. Wulandari, "Analisis QoS (*Quality of Service*) Pada Jaringan Internet (Studi Kasus : UPT Loka Uji Teknik Penambangan Jampang Kulon – Lipi)," Jurnal Teknik Informatika dan Sistem Informasi Vol. 2, 2016