**TUGAS AKHIR - KI141502**

# Rancang Bangun Sistem Pemantauan Detak Jantung Pasien Pengguna Kendaraan berbasis Mikrokontroler Arduino

ANGGARDA DIAJENG PRAMESWARI
NRP 5111100126

Dosen Pembimbing
Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Baskoro Adi Pratomo, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

*[Halaman ini sengaja dikosongkan]*

**FINAL PROJECT - KI141502**

# Design and Implementation of Heart Rate Monitoring System for Car User Patient based on Arduino Microcontroller

ANGGARDA DIAJENG PRAMESWARI
NRP 5111100126

Advisor
Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Baskoro Adi Pratomo, S.Kom., M.Kom.

INFORMATICS DEPARTMENT
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2015

*[Halaman ini sengaja dikosongkan]*

## LEMBAR PENGESAHAN

## RANCANG BANGUN SISTEM PEMANTAUAN DETAK JANTUNG PASIEN PENGGUNA KENDARAAN BERBASIS MIKROKONTROLER ARDUINO

# TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
**ANGGARDA DIAJENG PRAMESWARI**
NRP. 5111100126

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr.Eng. Radityo Anggoro, S.Kom., M.Sc. .....
   NIP: 198410162008121002 (pembimbing 1)

2. Baskoro Adi Pratomo, S.Kom., M.Kom. .....
   NIP: 198702182014041001 (pembimbing 2)

**SURABAYA**
**JUNI 2015**

*[Halaman ini sengaja dikosongkan]*

**RANCANG BANGUN SISTEM PEMANTAUAN DETAK JANTUNG PASIEN PENGGUNA KENDARAAN BERBASIS MIKROKONTROLER ARDUINO**

**Nama Mahasiswa :Anggarda Diajeng Prameswari**
**NRP :5111100126**
**Jurusan :Teknik Informatika FTIf-ITS**
**Dosen Pembimbing I :Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**
**Dosen Pembimbing II :Baskoro Adi Pratomo, S.Kom., M.Kom.**

## ABSTRAK

*Kesehatan merupakan satu hal penting dalam hidup. Kondisi kesehatan seseorang dapat dipantau dari perubahan yang terjadi pada bagian vital tubuh, yang salah satunya adalah jantung. Pentingnya peran jantung yang berfungsi untuk menyebarkan oksigen ke seluruh tubuh, mendorong perlunya pemantauan kondisi jantung secara intensif khususnya terhadap pasien kelainan jantung. Seiring dengan berkembangnya teknologi dan kebutuhan akan mobilitas pasien, perlu dirancang sistem pemantauan detak jantung jarak jauh bagi pasien yang sedang berkendara.*

*Tugas akhir ini akan dibangun dengan ide untuk memantau pasien pengguna kendaraan secara berkelanjutan dengan menggunakan mikrokontroler Arduino, sensor detak jantung dan sebuah modul komunikasi untuk memberi notifikasi kepada keluarga dan informasi rumah sakit terdekat berdasarkan posisi dari pengguna ketika itu, jika tindakan darurat diperlukan.*

*Sistem ini diharapkan dapat menjadi solusi bagi para penderita kelainan jantung untuk terus dapat melakukan aktifitasnya dalam berkendara dengan tetap terpantau kondisi kesehatan jantungnya dari jarak jauh. Hasil yang ditunjukkan pada sistem yaitu responsifitas sistem dari data dikirim hingga notifikasi diterima rata – rata sebesar 3.041 menit atau 3 menit 2 detik dan tingkat akurasi sistem yang mencapai 90% dalam*

*memproses data di lapangan dengan metode pengambilan keputusan Fuzzy*.

***Kata kunci: detak jantung, kesehatan, mikrokontroler Arduino, pemantauan, responsif*.**

**DESIGN AND IMPLEMENTATION OF HEART RATE MONITORING SYSTEM FOR CAR USER PATIENT BASED ON ARDUINO MICROCONTROLLER**


| | |
|---|---|
| **Name** | **:Anggarda Diajeng Prameswari** |
| **NRP** | **:5111100126** |
| **Jurusan** | **:Teknik Informatika FTIf-ITS** |
| **Advisor I** | **:Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.** |
| **Advisor II** | **:Baskoro Adi Pratomo, S.Kom., M.Kom.** |


## ABSTRACT


*Health is the most important aspect of life. A person's health condition can be monitored by the changes that happen in the vital organs, which one of them is heart. The importance role of heart is to distribute oxygen by pumping blood to all organs, enhancing the needs of monitoring heart condition intensively, especially anyone who suffers heart failures. With the growing of technology and patients' mobility, the needs of being able to monitor patients' heart rate remotely while driving are increasing.*

*This research study conducted based on the idea of having the ability to monitor patients while driving continuously using the Arduino microcontroller, a heart rate sensor and a communication device that is able to notify the family where the location of the nearest hospital is from the current position if an emergency situation emerged.*

*This system is expected to be the solution for anyone who suffers heart failures so that they are able to do their daily activities, in which driving a vehicle is included, as normal while their heart conditions are monitored remotely simultaneously. The outcome of this system which is its responsivity from the data being delivered to the notification being accepted is measured in the average of 3.041 minutes which is 3 minutes and 2 seconds*

*and its accuracy is measured to be 90% in processing the data by using Fuzzy as the decision making method.*

***Keywords: Arduino microcontroller, health, heart rate, monitoring, responsive.***

# KATA PENGANTAR

Dengan menyebut nama Allah SWT yang Maha Pengasih lagi Maha Panyayang, kami panjatkan puja dan puji syukur atas kehadirat-Nya, yang telah melimpahkan rahmat, hidayah, dan inayah-Nya kepada kami, sehingga penulis dapat menyelesaikan tugas akhir ini dengan baik.

Penulis ingin menyampaikan rasa hormat dan terima kasih yang setinggi-tingginya kepada pihak-pihak yang telah membantu penulis dalam penyelesaian tugas akhir ini, terutama kepada:

1. Bapak Abdul Wahab dan Ibu Lucia Fransisca Latupeirissa, orang tua penulis, yang menjadi motivasi penulis dalam membangun sistem monitoring detak jantung dan yang selalu memberikan dukungan dan semangat baik dalam bantuan finansial maupun dorongan positif agar penulis mampu untuk menyelesaikan tugas akhir dengan benar dan tepat waktu, serta memberikan doa yang terus dikirimkan agar penyelesaian tugas akhir berjalan dengan lancar.
2. Paramita Siswari Putri, saudara penulis, yang selalu memberikan semangat dan memberikan doa agar penulis mampu menyelesaikan tugas akhir.
3. Ibu Henning Titi Ciptaningtyas, S.Kom., M.Kom. sebagai dosen wali penulis yang turut memberikan saran dan menuntun penulis dalam menentukan mata kuliah yang akan diambil pada pembelajaran di Jurusan Teknik Informatika ITS ini.
4. Bapak Dr.Eng. Radityo Anggoro, S.Kom., M.Sc. dan Bapak Baskoro Adi Pratomo, S.Kom., M.Kom. yang telah bersedia untuk menjadi dosen pembimbing tugas akhir sehingga penulis dapat mengerjakan tugas akhir dengan arahan dan bimbingan yang baik dan jelas.

5. Teman-teman Mahasiswa Teknik Informatika 2011 yang telah berjuang bersama-sama selama menempuh pendidikan di Jurusan ini.
6. Danang Prawira Nugraha yang telah membantu penulis dalam menguasai banyak materi selama kuliah dan memberikan inspirasi tersindiri bagi penulis.
7. Nabilla Sabbaha Audria yang telah membantu penulis berbagi ilmu tentang mikrokontroler Arduino dan sahabat seperjuangan selama kuliah.
8. Amanda Tiara Averousi sebagai teman seperjuangan kerja praktek di PT. Bio Farma di kota Bandung, pada tahun 2014 yang telah berbagi banyak materi perangkat lunak.
9. Sahabat – sahabat seperjuangan dari Madiun, Bayu Faliandra, yang telah menjadi mentor dan memberikan banyak materi elektronika dasar dan mikrokontroler, Nurul Aini dan Fishatania Nirwesthi yang selalu setia memberi semangat dan hiburan kepada penulis selama pengerjaan tugas akhir.
10. Serta pihak-pihak lain yang turut membantu penulis baik secara langsung maupun tidak, yang namanya tidak penulis sebutkan disini.

Penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, mohon maaf apabila ada kesalahan dan kata-kata yang dapat menyinggung perasaan. Penulis berharap tugas akhir ini dapat menjadi solusi terhadap pemantauan detak jantung bagi kerabat yang memiliki permasalahan jantung di Indonesia.

Surabaya, Juni 2015

Penulis

# DAFTAR ISI

*[Halaman ini sengaja dikosongkan]*

# DAFTAR TABEL

# DAFTAR GAMBAR

*[Halaman ini sengaja dikosongkan]*

# BAB I
# PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan tugas akhir, dan sistematika penulisan.

## 1.1. Latar Belakang

Kesehatan merupakan hal yang penting bagi manusia. Dengan kesehatan kita dapat melakukan berbagai kegiatan dan berpikir dengan baik. Menjaga kesehatan dapat dilakukan dengan banyak cara, salah satunya memantau kondisi fisik secara berperiodik. Dengan memantau konfisi fisik secara berperiodik, maka dapat dicegah hal – hal yang tidak diinginkan terjadi. Salah satu bagian dari *monitoring* kesehatan adalah pengecekan detak jantung. Hal tersebut dikarenakan jantung merupakan organ yang mengontrol aliran darah ke seluruh tubuh. Dengan mendeteksi jantung seseorang kita dapat mengetahui kondisi kesehatan seseorang sedang dalam kondisi yang baik atau tidak. Kondisi jantung seseorang menjadi lebih vital untuk dipantau ketika seseorang berada dalam kondisi yang tidak sehat, khususnya bagi penderita penyakit jantung.

Dewasa ini mulai banyak dikembangkan sistem pemantau detak jantung. Tetapi sebagian besar hanya merupakan sistem yang ditargetkan kepada pembacaan detak jantung, sehingga jika pun ada pembacaan dari sensor lain digunakan sebagai dua parameter yang berbeda tanpa saling diintegrasikan [1]. Oleh karena itu dibutuhkan suatu sistem yang tidak hanya melakukan pembacaan detak jantung, tetapi juga menggunakan parameter lain dalam suatu sistem pemantauan. Pada tugas akhir ini ini, sistem yang dibuat adalah sistem yang dapat memantau jumlah detak jantung seseorang per satuan waktu yang diintegrasikan dengan kecepatan dan percepatan seseorang ketika sedang

berkendara. Untuk membantu integrasi sistem ini diperlukan bantuan teknologi sensor dan mikrokontroler yang dapat mengukur jumlah detak jantung per satuan waktu berupa *Pulse* Sensor dengan bentuk *ear clip* dan GPS/GPRS/GSM *Shield* V3.0 untuk mendapatkan kecepatan, percepatan dan lokasi dari modul GPS untuk dapat memberikan notifikasi.

Sistem ini diharapkan mampu mengintegrasikan jumlah detak jantung per satuan waktu dengan percepatan dan kecepatan seseorang ketika berkendara untuk kemudian mengirimkan notifikasi berupa SMS kepada keluarga ketika terjadi perubahan jumlah detak jantung per satuan waktu dengan disertai perubahan kecepatan dan percepatan dalam kurun waktu tertentu. Dengan adanya integrasi dengan percepatan dan kecepatan diharapkan mampu membuat pengiriman notifikasi menjadi lebih akurat terhadap kondisi perubahan jumlah detak jantung per satuan waktu seseorang yang sedang berkendara.

## 1.2. Rumusan Permasalahan

Rumusan masalah yang ingin diselesaikan penulis dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana membuat sistem pemantau detak jantung bagi pasien pengguna kendaraan yang responsif berdasarkan detak jantung, kecepatan kendaraan dan percepatan yang dialami?
2. Bagaimana menbuat sistem notifikasi kepda kerabat dan pengguna sendiri tentang posisi dan kondisi pengguna?
3. Bagaimana sistem dapat membantu pengguna dalam mencari rumah sakit atau fasilitas kesehatan terdekat?

## 1.3. Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Sistem digunakaan saat pengguna berkendara dengan mobil.

2. Data yang dikirim ke server adalah detak jantung, kecepatan, dan posisi koordinat pengguna.
3. Notifikasi yang dikirimkan berupa pesan teks dan notifikasi pada Android pengguna.

## 1.4. Tujuan

Tujuan dari pembuatan tugas akhir ini adalah:

1. Membuat sistem yang dapat memantau kondisi detak jantung pengguna ketika berkendara.
2. Membuat sistem yang dapat memberikan notifikasi kepada keluarga dan pengguna sendiri.

## 1.5. Manfaat

Manfaat dari tugas akhir ini adalah sebagai sarana untuk memantau kondisi pasien penderita penyakit jantung ketika berkendara untuk kemudian akan dikirimkan notifikasi berdasarkan kondisi detak jantung, kecepatan dan percepatan kepada keluarga dan lokasi rumah sakit terdekat melalui *smartphone* pengguna.

*[Halaman ini sengaja dikosongkan]*

# BAB II
# DASAR TEORI

Pada bab ini akan dibahas mengenai dasar teori yang menjadi dasar pembuatan tugas akhir ini. Pokok permasalahan yang akan di bahas adalah tentang teknologi yang mendukung pembuatan tugas akhir dari segi perangkat lunak maupun perangkat keras dan juga informasi lanjut mengenai jantung dan hasil penelitian terdahulu yang relevan dengan topik yang diangkat dalam tugas akhir ini.

## 2.1. Jantung

Jantung merupakan organ vital manusia yang dapat digunakan untuk mengukur kondisi kesehatan seseorang. Salah satu cara untuk mengetahui kondisi jantung seseorang adalah dengan mengukur jumlah detak jantung per menit dan pada tugas akhir akan dibuat sebuah sistem yang melakukan pembacaan detak jantung.

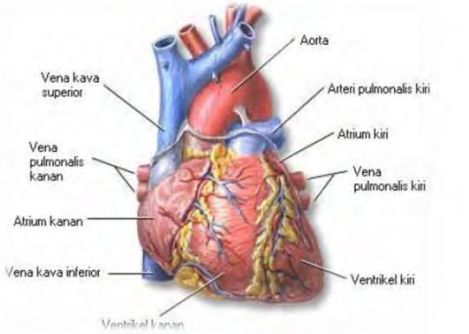
### 2.1.1. *Monitoring* Detak Jantung

Sebagai salah satu indikator kesehatan, jantung sangat penting untuk diperiksa secara berkala [2]. Selama ini pemeriksaan detak jantung yang berkembang di dunia kedokteran Indonesia adalah dengan pemeriksaan elektrokardiografi (EKG). Pemeriksaan menggunakan mesin EKG diperuntukkan bagi pasien yang ingin memeriksakan kondisi jantung mereka. EKG adalah suatu metode mempelajari kerja otot jantung sehingga dapat membantu diagnosis abnormalitas jantung dan kecendurungan atau perubahan fungsi jantung. Prinsip kerja EKG adalah merekam signal elektrik yang berkaitan dengan aktivitas jantung dan menghasilkan grafik rekaman tegangan listrik terhadap waktu [3].

### 2.1.2. Sistem Kardiovaskuler

Sistem kardiovaskuler adalah suatu sistem tubuh yang berkaitan dengan jantung dan urat-urat (pembuluh) darah.

Kardiovaskuler tersusun dari jantung dan pembuluh-pembuluhnya seperti aorta, arteri, arteriola, vena dan venula. Jantung merupakan suatu organ otot berongga yang terletak di pusat dada. Bagian kanan dan kiri jantung masing – masing memiliki ruang sebelah atas (atrium) yang mengumpulkan darah dan ruang sebelah bawah (ventrikel) yang mengeluarkan darah [4]. Ventrikel terdiri dari ventrikel kiri dan ventrikel kanan, begitu pula atrium terdiri dari atrium kanan dan atrium kiri, seperti yang terlihat pada Gambar 2.1.



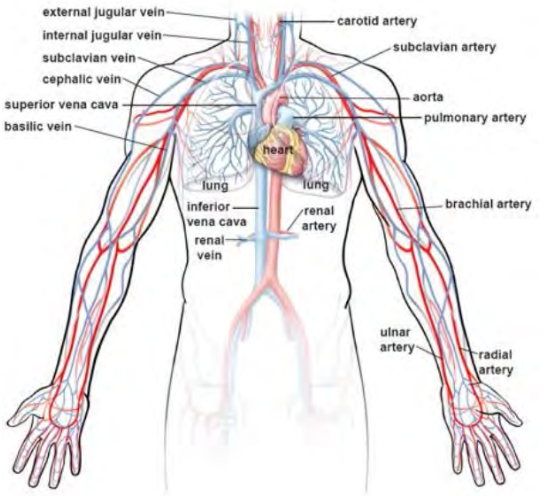

**Gambar 2.1 Jantung Tampak Depan [4]**

Fungsi utama jantung adalah menyediakan oksigen ke seluruh tubuh dan membersihkan tubuh dari hasil metabolisme (karbondioksida). Jantung melaksanakan fungsi tersebut dengan mengumpulkan darah yang kekurangan oksigen dari seluruh tubuh dan memompanya ke dalam paru-paru dimana darah akan mengambil oksigen dan membuang karbondioksida. Jantung kemudian mengumpulkan darah yang kaya oksigen dan memompanya ke jaringan di seluruh tubuh. Saat melakukan kerja jantung memiliki tiga periode, yaitu periode kontstriksi, periode dilatasi dan periode istirahat. Periode konstriksi adalah keadaan di mana jantung bagian ventrikel dalam keadaan bekerja dengan kondisi menguncup, periode dilatasi adalah ketika jantung sedang mengambang di mana darah dari seluruh tubuh akan masuk ke jantung untuk dibersihkan dan kemudian dipompa lagi ke seluruh tubuh dan periode istirahat yaitu waktu antara periode konstriksi dan periode dilatasi di mana jantung

berhenting sekitar 1/10 detik [5]. Pada saat berdetak, jantung mengendur dan terisi darah, untuk selanjutnya berkontraksi dan memompa darah keluar dari jantung. Detak yang dihasilkan oleh jantung inilah yang kemudian diukur untuk pemeriksaan kesehatan [4].

### 2.1.3. *Heart Rate* Manusia

Akibat adanya kerja jantung terutama ventrikel dan atrium, maka dihasilkan detak jantung. Dengan dihasilkan detak jantung tersebut kemudian melahirkan *heart rate* [4]. Saat ventrikel kiri berkontraksi, darah dipompa ke seluruh tubuh dengan tekanan yang besar dan kemudian terbagi ke seluruh pembuluh arteri dan pembuluh arteri kecil [6]. Peredaran darah dari jantung ke tubuh bagian atas terlihat pada Gambar 2.2.




**Gambar 2.2 Peredaran Darah Jantung Ke Atas Tubuh [7]**

Periode kerja jantung saat memompa dan menarik darah tersebut merupakan satu kali jantung berdetak. Jumlah detak jantung

per menit normal dari orang dewasa, baik yang sehat atau tidak, tidak jauh berbeda yakni berkisar antara 60 sampai 100 kali per menit [8]. *Heart rate* merupakan salah satu indikator yang bisa digunakan untuk mengukur kondisi kesehatan seseorang dan bahkan resiko seseorang menderita penyakit jantung [9]. Bertambahnya jumlah detak jantung per menit bagi seseorang jika diukur dalam kurun waktu tertentu dapat menunjukkan resiko terkena penyakit jantung, dibandingkan dengan seseorang dengan *heart rate* yang stabil [10].

## 2.2. Hubungan Antara Jantung, Kecepatan dan Percepatan

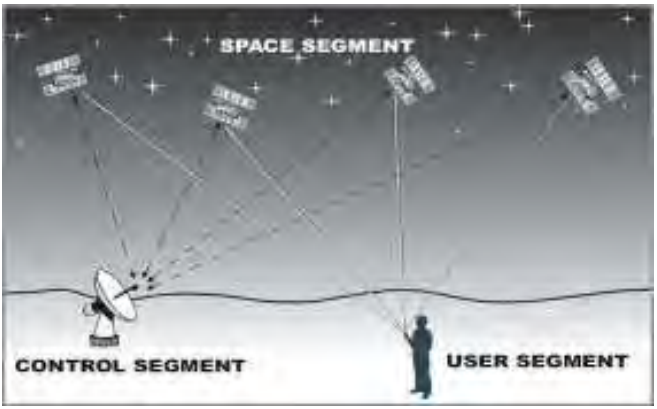
Pada tugas akhir ini akan dibangun sebuah sistem pemantau detak jantung yang memantau kondisi detak jantung pengguna disaat menggunakan kendaraan dan menggunakan kecepatan dari kendaraan dan percepatan yang dialami sebagai parameter yang saling mempengaruhi satu sama lain. Kondisi jantung di saat beristirahat atau tidak beraktifitas berbeda dengan kondisi jantung di saat berkendara, hal ini dikarenakan di saat berkendara banyak faktor – faktor dari luar yang berpotensi untuk mempengaruhi kondisi detak jantung. Faktor tersebut beberapa diantaranya adalah kecepatan dan percepatan. Dalam mengemudi, semakin tinggi kecepatan yang ditempuh dan semakin besar percepatan yang dialami, detak jantung seseorang juga akan semakin meningkat [11]. Tabel 2.1 berikut menunjukkan hasil studi yang menunjukkan bahwa detak jantung akan terpacu untuk menjadi lebih tinggi ketika terjadi penambahan kecepatan dan percepatan, studi dilakukan terhadap 14 orang pengemudi dan dilakukan selama 3 menit.

**Tabel 2.1 Hasil Studi Efek Kecepatan pada Fisik (*Heart Rate)* Pengemudi [11]**

| | | | | | |
|---|---|---|---|---|---|
| Kecepatan (km/jam) | 80 | 90 | 100 | 110 | 120 |
| Percepatan ($m/s^2$) | 0.26 | 0.30 | 0.32 | 0.36 | 0.43 |
| Rata – rata | 0.05 | 0.06 | 0.03 | 0.05 | 0.08 |

## 2.3. *Global Positioning System* (GPS)

Sistem GPS adalah sebuah sistem navigasi yang menggunakan sejumlah satelit yang berada di orbit bumi, yang memancarkan sinyal ke bumi dan ditangkap oleh sebuah alat penerima. Terdapat tiga bagian penting dari sistem ini, yaitu bagian kontrol, bagian angkasa dan bagian pengguna [12]. Skema dari sistem GPS yang terdiri dari tiga bagian terlihat pada Gambar 2.3.




**Gambar 2.3 Skema Sistem GPS [12]**

Pesawat penerima GPS menggunakan sinyal satelit untuk melakukan triangulasi posisi yang hendak ditentukan dengan cara mengukur lama perjalanan waktu sinyal dikrimkan dari satelit, kemudian mengalihkannya dengan kecepatan cahaya untuk menentukan secara tepat seberapa jauh pesawat penerima GPS dari setiap satelit. Dengan mengunci sinyal yang ditransmit oleh satelit, minimum 3 sinyal dari satelit yang berbeda, pesawat penerima GPS menghitung posisi tetap sebuah titik, yaitu posisi lintang (*latitude*) dan posisi bujur (*longitude*) atau sering disebut dengan *2D fix*. Penguncian sinyal satelit yang keempat membuat pesawat penerima GPS dapat menghitung posisi ketinggian titik tersebut terhadap muka

laut rata-rata (*Mean Sea/Level*) atau disebut dengan *3D fix* dan keadaan ini ideal untuk navigasi. Pada tugas akhir ini, teknologi GPS akan digunakan untuk mendapatkan informasi posisi pengguna.

## 2.4. NMEA

*National Marine Electronic Association* atau NMEA dikembangkan secara spesifik untuk standar industri sebagai antar muka bermacam-macam alat kelautan yang diperkenalkan sejak tahun 1983. Salah satu peralatan yang mengeluarkan NMEA adalah GPS. NMEA berisi informasi yang berhubungan dengan data geografis seperti waktu, *latitude, longitude*, ketinggian, kecepatan dan masih banyak lagi. Untuk menampilkan informasi yang lebih dimengerti oleh pengguna data NMEA perlu diolah lebih lanjut lagi [13]. Pada tugas akhir ini, informasi koordinat dan kecepatan akan didapatkan dari hasil ekstrasi kalimat NMEA yang didapatkan melalui teknologi GPS. Daftar beberapa kalimat NMEA terlihat pada Tabel 2.1.

**Tabel 2.2 Kalimat NMEA**

| Kalimat | Deskripsi |
|---|---|
| $GPGGA | *Global positioning system fixed data* |
| $GPGLL | *Geographic position (latitude/longitude)* |
| $GPGSA | *GNSS DOP and active satellites* |
| $GPGSV | *GNSS satellites in view* |
| $GPRMC | *Recommended minimum specific GNSS data* |
| $GPVTG | *Course over ground and ground speed* |

## 2.5. *Global System for Mobile Communication* (GSM)

GSM adalah sebuah teknologi komunikasi selular yang bersifat digital. Teknologi GSM banyak diterapkan pada komunikasi bergerak, khususnya telepon genggam. Teknologi ini memanfaatkan gelombang mikro dan pengiriman sinyal yang dibagi berdasarkan waktu, sehingga sinyal informasi yang dikirim akan sampai pada

tujuan. GSM dijadikan standar global untuk komunikasi selular sekaligus sebagai teknologi selular yang paling banyak digunakan orang di seluruh dunia. Pada tugas akhir ini teknologi GSM akan digunakan untuk pengiriman data dan operator yang digunakan adalah Telkomsel dengan frekuensi 900Mhz – 907 Mhz (7,5 Mhz) [14].

Jaringan GSM adalah sebuah *Public Land Mobile Network* (PLMN) yang terbentuk dari empat bagian utama sistem yang saling terhubung. Bagian utama sistem jaringan GSM adalah sebagai berikut:

1. *Mobile Station* (MS) merupakan perangkat yang digunakan oleh pelanggan untuk melakukan pembicaraan, yang terdiri dari:
   a. *Mobile Equipment* yang berfungsi sebagai pengirim dan penerima sinyal untuk berkomunikasi dengan perangkat GSM lain.
   b. *Subsriber Identity Module* (SIM) atau kartu SIM yang berisi seluruh informasi pelanggan dan pelayanan.
2. *Base Station System* (BSS) yang terdiri dari:
   a. *Base Transceiver Station* atau BTS yang merupakan perangkat GSM yang berhubungan langsung dengan MS dan berfungsi sebagai pengirim sinyal.
   b. *Base Station Controller* yaitu perangkat yang mengontrol kerja BTS-BTS yang berada di bawahnya dan sebagai penghubung BTS dan MSC.
3. *Network Sub System* (NSS) yang terdiri dari:
   a. *Mobile Switching Center* atau MSC, yaitu inti dari jaringan seluler, dimana MSC berperan untuk interkoneksi hubungan pembicaraan, baik antar selular maupun dengan jaringan kabel PSTN, ataupun dengan jaringan data.
   b. *Home Location Register* atau HLR, yang berfungsi sebagai sebuah basis data untuk menyimpan semua data dan informasi mengenai pelanggan agar tersimpan secara permanen.
   c. *Visitor Location Register* atau VLR, yang berfungsi untuk menyimpan data dan informasi pelanggan.

d. *Authentication Center* atau AuC, yang diperlukan untuk menyimpan semua data yang dibutuhkan untuk memeriksa keabsahaan pelanggan.
e. *Equipment Identity Registration* atau EIR, yang memuat data-data pelanggan.

4. *Operation and Support System* (OSS) yang merupakan sub sistem jaringan GSM yang berfungsi sebagai pusat pengendalian, yang berupa *fault management*, *configuration management*, *performance management* dan *inventory management*.

Pada tugas akhir ini, teknologi GSM akan digunakan pada modul komunikasi dan juga untuk mengaktifkan modul GPS yang diperlukan dalam pembacaan data lokasi dengan mendaftarkan GSM terlebih dahulu ke jaringan.

## 2.6. *General Packet Radio Service* (GPRS)

GPRS adalah teknologi yang sebenarnya adalah bentuk upgrade dari fitur – fitur dasar GSM, yang memungkinkan pengiriman dan penerimaan data yang lebih cepat. Teknologi GPRS saat ini sudah diimplementasikan oleh seluruh operator yang ada di Indonesia. GPRS dapat digunakan untuk melakukan transfer data dalam bentuk paket data yang berkaitan dengan email, data gambar, *Wireless Application Protocol* (WAP) dan *World Wide Web* (WWW) [15]. Komponen–komponen utama dalam jaringan GPRS antara lain:

1. *Gateway GPRS Support Node* (GGSN) yang berfungsi sebagai antarmuka ke PDN (*Public Data Network*), *information routing*, *network screening*, *user screening*, dan *address mapping*.
2. *Serving GPRS Support Node* (SGSN) yang berfungsi untuk mengantarkan paket data ke MS, *update* pelanggan ke HLR, registrasi pelanggan baru.
3. PCU yaitu komponen di level BSS yang menghubungkan terminal ke jaringan GPRS.

Pada tugas akhir ini, teknologi GPRS akan digunakan pada modul komunikasi untuk mengirimkan paket data yang berupa hasil pembacaan sensor ke server.

## 2.7. AT *Command*

AT *command* adalah perintah-perintah yang digunakan dalam komunikasi dengan *serial port*. Dengan AT *command* maka dapat diketahui vendor dari handphone yang digunakan, kekuatan sinyal, membaca pesan yang ada pada SIM card, mengirim pesan, mendeteksi pesan SMS baru yang masuk secara otomatis, menghapus pesan pada SIM card, dan masih banyak lagi [12]. Pada tugas akhir ini AT *command* digunakan untuk komunikasi antara GSM, GPRS dan juga GPS. Contoh beberapa AT *command* dapat dilihat di Tabel 2.2.

**Tabel 2.3 Daftar AT *Command***

| AT *Command* | Keterangan |
|---|---|
| AT | Mengecek apakah alat sudah terhubung |
| AT+CREG | Mendaftar ke jaringan GSM |
| AT+CGPSIPR | Me-*reset* GPS mode (*cold/warm/hot*) |
| AT+CGPSPWR | Mengatur *power* dari GPS |

## 2.8. MySQL

MySQL adalah sebuah perangkat lunak yang bersifat *open-source* untuk manajamen basis data SQL atau DBMS yang *multi-thread* dan *multi-user*. SQL adalah sebuah konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis. Pada tugas akhir ini, teknologi MySQL digunakan sebagai basis data untuk menyimpan data jumlah detak jantung per menit, kecepatan dan percepatan serta lokasi dari pengguna [16].

## 2.9. PHP

PHP adalah singkatan dari *Hypertext Prepocessor*, yaitu bahasa pemrograman yang digunakan secara luas untuk penanganan pembuatan dan pengembangan sebuah situs web dan bisa digunakan bersamaan dengan HTML. Pada tugas akhir ini, PHP digunakan untuk dalam pembangunan aplikasi server [16].

## 2.10. Gammu

Gammu adalah suatu perangkat lunak yang digunakan untuk membuat SMS *Gateway* yang tidak berbayar. Gammu SMS *Gateway* dapat dibangun dengan banyak program dan *platform*, baik itu *web-based* dengan PHP ataupun *desktop* dengan Delphi maupun VB. Pada tugas akhir ini teknologi Gammu yang digunakan adalah *web-based* dengan PHP [17].

## 2.11. *Google Cloud Messaging* (GCM)

*Google Cloud Messaging* (GCM) untuk Android adalah sebuah *service* yang mengijinkan adanya pengiriman data dari server ke pengguna Android yang *compatible*, dan menerima pesan dari perangkat yang memiliki koneksi yang sama dengan GCM *service* dengan melakukan antrian pesan dan dikirimkan ke target aplikasi Android yang sedang berjalan di perangkat target. GCM seutuhnya gratis sebesar apapun ukuran pesan yang akan dikirimkan maupun diterima dan tidak ada batasan kuota dalam penggunaannya. Untuk menggunakan GCM, setiap perangkat yang akan dilibatkan harus terdaftar dalam *web server* GCM. Hal ini berguna untuk menyimpan registrasi GCM ID masing-masing perangkat ke *webserver* dan satu perangkat hanya terdaftar satu registrasi GCM ID saja. Setiap aplikasi Android yang menggunakan GCM, di dalam kode sumber aplikasi tersebut telah dimasukkan fungsi GCM yang berisi inisialisasi registrasi GCM ID jika perangkat tersebut belum pernah didaftarkan

sebelumnya, dan jika sudah terdaftar maka akan menyimpan nomor registrasi GCM ID perangkat tersebut menjadi session selama aplikasi tersebut berjalan. Tujuannya adalah untuk memanipulasi notifikasi pada tata kelola profil pengguna. Jika pengguna ingin mendapatkan notifikasi, maka dapat mengubah inisialisasi GCM notifikasi pengguna menjadi true, sedangkan jika pengguna tidak ingin mendapatkan notifikasi, maka dapat mengubah inisialisasi GCM notifikasi pengguna menjadi *false* [18].

Pada tugas akhir ini, *Google Cloud Messaging* akan digunakan untuk memberikan *push notification* kepada *smartphone* Android pengguna untuk menginformasikan lokasi rumah sakit terdekat dari lokasi pengguna sekarang.

## 2.12. *Google Maps Android* API v2

*Google Maps Android* API v2 adalah sebuah *service* yang menyediakan layanan peta dari Google untuk dapat digunakan pada aplikasi yang dikembangkan. *Google Maps Android* API v2 ini secara otomatis akan mengatur akses ke server dari *Google Maps*, mengunduh data, menampilkan peta dan mengatur interaksi pengguna dengan peta. Pada tugas akhir ini, *Google Maps Android* API v2 ini digunakan untuk menampilkan lokasi rumah sakit terdekat dari lokasi pengguna sekarang ketika pengguna diperlukan untuk diberikan notifikasi [19].

## 2.13. jFuzzyLogic

*Fuzzy logic* merupakan suatu metode yang digunakan dalam proses pengambilan keputusan dengan cara memetakan suatu ruang input ke dalam ruang output. Sistem ini diperkenalkan dimana pada saat itu *boolean logic* hanya mengenal dua keadaan yaitu: *True/False* atau hanya mempunyai logika 0 dan 1 saja. Sedangkan kondisi nyata di alam ini bukan hanya ya (1, *true*) atau tidak (0, *false*) tetapi seluruh kemungkinan diantara 0 dan 1, sehingga untuk mengenal

kondisi ini kita tidak dapat menggunakan *boolean logic* tetapi dengan menggunakan *fuzzy logic*. Pada tugas akhir ini *Fuzzy Logic* digunakan sebagai algoritma pengambilan keputusan dari data sensor. Pada tugas akhir ini *Fuzzy Logic* akan diimplementasikan dengan menggunakan *library* bahasa pemrograman Java yaitu jFuzzyLogic [20]. *Library* jFuzzyLogic adalah pustaka pemrograman yang memiliki fitur untuk melakukan perhitungan algoritma *fuzzy* yang sederhana, dengan mengatur masing – masing masukkan yang digunakan dan mengatur batas masing – masing. Pada *library* ini Secara garis besar, proses pada sistem *fuzzy* meliputi tiga tahap, yaitu:

### 2.13.1. Tahap *Fuzzification*

Proses ini berfungsi untuk mengubah masukan-masukan berupa nilai analog atau yang nilai kebenarannya bersifat pasti (*crisp input*) menjadi nilai *fuzzy*, yang digunakan sebagai *fuzzy* input. *Fuzzy* input ini berupa nilai linguistik yang semantiknya ditentukan berdasarkan fungsi keanggotaan tertentu. Jika terdapat suatu nilai analog yang menjadi input pada proses *fuzzy* maka input tersebut dimasukkan pada batas domain sehingga didapatkan suatu nilai fungsi keanggotaan. Nilai fungsi keanggotaan inilah yang menentukan proses pengambilan keputusan selanjutnya.

### 2.13.2. Tahap *Rule Evaluation*

Proses ini digunakan untuk mencari nilai *fuzzy* output dari *fuzzy* input. Jika terdapat suatu nilai fuzzy input dari proses *fuzzification* nilai tersebut akan dimasukkan ke dalam *rule* yang telah dibuat untuk mendapatkan nilai *fuzzy* output. Pada proses inilah suatu sistem dapat dikatakan pintar atau tidak. Jika *rule* yang dibuat tidak pintar maka sistem yang dikontrol menjadi kacau dan objek yang seharusnya dikenali menjadi tidak dapat dibaca. Dengan *rule* yang ada diperoleh nilai *fuzzy* yang digunakan dalam membantu pengambilan keputusan.

### 2.13.3. Tahap *Defuzzification*

Pada tahap inilah pengambilan keputusan dilakukan. Nilai yang didapatkan berupa nilai *crisp*, yaitu 0 atau 1. Proses *defuzzification* melakukan suatu fungsi output yang memproses nilai *fuzzy* yang berasal dari *rule evaluation* sehingga keputusan akhir dapat dilakukan. Fungsi output ini dilakukan dengan mencocokan nilai-nilai yang ada dengan *threshold* yang ditentukan. Sebuah input pada proses *fuzzy* akan diterima sebagai anggota himpunan *fuzzy* jika memiliki nilai keanggotaan yang melewati batas *threshold* yang ada.

## 2.14. Mikrokontroler Arduino Uno

Arduino Uno yang ditunjukkan pada Gambar 2.4 adalah salah satu jenis mikrokontroler berbasis ATmega 328. Mikrokontroler ini menggunakan bahasa pemrograman *processing* yang mengkombinasikan bahasa pemrograman C++ dan Java, sehingga penggunanya akan lebih mudah menggunakan jika telah terbiasa dengan menggunakan kedua bahasa pemrograman tersebut. Arduino memiliki 14 pin digital input/output, 6 pin digunakan sebagai PWM output, 6 analog input, 16 MHz sumber *clock signal* untuk sirkuit digital, sebuah konektor USB, sebuah colokan listrik, sebuah konektor ICSP, dan tombol *reset*. Arduino Uno dapat diberikan *power* melalui konektor USB atau dengan *power supply* eksternal [21]. Pada tugas akhir ini, Arduino Uno digunakan sebagai mikrokontroler untuk pembacaan data dari sensor agar dapat saling terintegrasi dan dikirim ke server.


**Gambar 2.4 Mikrokontroler Arduino Uno [21]**

## 2.15. *Pulse* Sensor

*Pulse* Sensor seperti yang ditunjukkan pada Gambar 2.5 adalah sensor untuk mendeteksi detak jantung yang mudah untuk digunakan. *Pulse* Sensor ini mengkombinasikan sensor detak jantung optik yang sederhana dengan *amplification* dan *noise cancellation* yang memungkinkan pembacaan data yang tepat dan *reliable*. Pulse Sensor ini menggunakan *power* 4mA pada tegangan 5V sehingga cocok untuk digunakan pada aplikasi *mobile* [22].

Pada tugas akhir ini, *Pulse* Sensor digunakan untuk membaca jumlah detak jantung per menit dengan dipasangkan di telinga pengguna menggunakan *ear clip* yang juga merupakan bagian dari paket pulse sensor.

**Gambar 2.5 *Pulse* Sensor**

## 2.16. DFRobot GPS/GPRS/GSM *Shield* V3.0

GPS/GPRS/GSM *Shield* V3.0 seperti yang ditunjukkan pada Gambar 2.6 adalah modul yang berkemampuan komunikasi *Quad-band* GSM/GPRS pada frekuensi EGSM 900MHz/DCS 1800MHz dan GSM 850 MHz/PCS 1900MHz. Selain itu, ditanamkan pula kemampuan GPS untuk navigasi satelit. Pengontrolan via AT *command* (GSM07.07 ,07.05 dan SIMCOM *enhanced* AT *Commands*). Desain *shield* ini memungkinkan untuk mengendalikan fungsi GSM & GPS langsung dari komputer dan Arduino. Untuk antena GPS dan GSM yang digunakan adalah antena *high-gain* SMD

yang sudah terpasang pada *shield* [23]. Pada tugas akhir ini, alat ini digunakan untuk mendapatkan data GPS dan kecepatan serta sebagai pendukung Arduino untuk pengiriman data ke server melalui internet.

**Gambar 2.6 DFRobot GPS/GPRS/GSM *Shield* V3.0**

*[Halaman ini sengaja dikosongkan]*

# BAB III
# PERENCANAAN SISTEM

Bab ini membahas metodologi yang digunakan pada pengerjaan tugas akhir. Metodologi yang akan dijelaskan meliputi deskripsi umum dan arsitektur sistem yang merupakan bagian dari analisis dan perancangan sistem yang akan dibangun.

## 3.1. Deskripsi Sistem

Pada subbab ini akan dibahas mengenai penjelasan umum mengenai sistem dan model basis data yang dipakai dalam pengembangan sistem.

### 3.1.1. Deskripsi Umum Sistem

Pada tugas akhir ini akan dibangun sebuah sistem yang digunakan sebagai media pantau detak jantung dari pasien pengguna kendaraa, untuk menghindari hal-hal yang tidak diinginkan ketika berkendara. Pasien yang dimaksudkan disini adalah semua pengguna kendaraan yang memiliki kondisi fisik yang tidak dalam keadaan sehat. Parameter tidak sehat diukur pada pengguna yang memiliki penyakit terkait dengan jantung dan sistem kardiovaskuler.

Sistem akan dibangun dengan menggunakan integrasi mikrokontroler Arduino Uno R3 antara modul sensor yang terdiri dari sensor detak jantung, *Pulse* Sensor, yang terhubung secara fisik dengan pengguna dan sensor GPS mendapatkan data lokasi dan kecepatan pengguna dengan modul komunikasi yang menggunakan modul GSM/GPRS untuk mengirimkan data ke server. Modul yang digunakan untuk membaca data sensor dan mengirimkan data terdapat pada satu modul yaitu GPS/GPRS/GSM *Shield* V3.0. Di server nantinya data akan disimpan ke dalam basis data dan akan dihitung percepatan per satuan waktu, untuk kemudian digunakan dalam algoritma

pengambilan keputusan *Fuzzy*, dimana terdapat tiga jenis parameter yang akan diolah, yaitu detak jantung per menit, kecepatan pengguna per satuan waktu dan percepatan pengguna per satuan waktu.

Pengambilan keputusan akan dimasukkan kedalam tiga kategori yaitu bahaya, berpotensi dan normal. Pengambilan keputusan ini akan didasarkan dari penelitian sebelumnya yang terkait dengan hubungan kondisi detak jantung pasien dan tingkat mortalitas dari jumlah detak jantung per menit [8] dan bagimanaa percepatan dan kecepatan ketika berkendara mempengaruhi kondisi detak jantung [11]. Berdasarkan keputusan ini pengguna akan mendapatkan respons dari server yang berupa notifikasi beberapa fasilittas kesehatan terdekat dari lokasi pengguna saat itu pada *smartphone* berbasis Android pengguna dan pesan teks yang akan dikirimkan kepada kerabat yang nomornya sudah terdaftar.

### 3.1.2. Model Fisik Basis Data

Pada sistem ini data sensor akan disimpan dalam basis data MySQL. Tabel yang digunakan pada basis data terlihat pada Tabel 3.1.

**Tabel 3.1 Struktur Tabel Sensor**

| Nama | Tipe Data | Atribut | Deskripsi |
|---|---|---|---|
| id_sensor | Integer | Primary Key | ID dari tiap data sensor yang dikirimkan. |
| timestamp | Timestamp | | Keterangan waktu dari data (detik, menit, jam, tanggal, bulan, tahun). |
| latitude | Varchar | | Koordinat *latitude* dari posisi terakhir yang terbaca oleh modul GPS. |
| longitude | Varchar | | Koordinat *longitude* |

| | | | |
|---|---|---|---|
| | | | dari posisi terakhir yang terbaca oleh modul GPS. |
| kecepatan | Float | | Kecepatan terakhir yang terbaca oleh modul GPS. |
| percepatan | Float | | Percepatan terakhir dari berdasarkan waktu dan kecepatan. |
| heart_rate | Integer | | Jumlah detak jantung per menit yang terbaca oleh modul sensor. |
| status | Integer | | Status pengguna untuk notifikasi |
| condition | Varchar | | Kondisi akhir hasil dari *Fuzzy* |

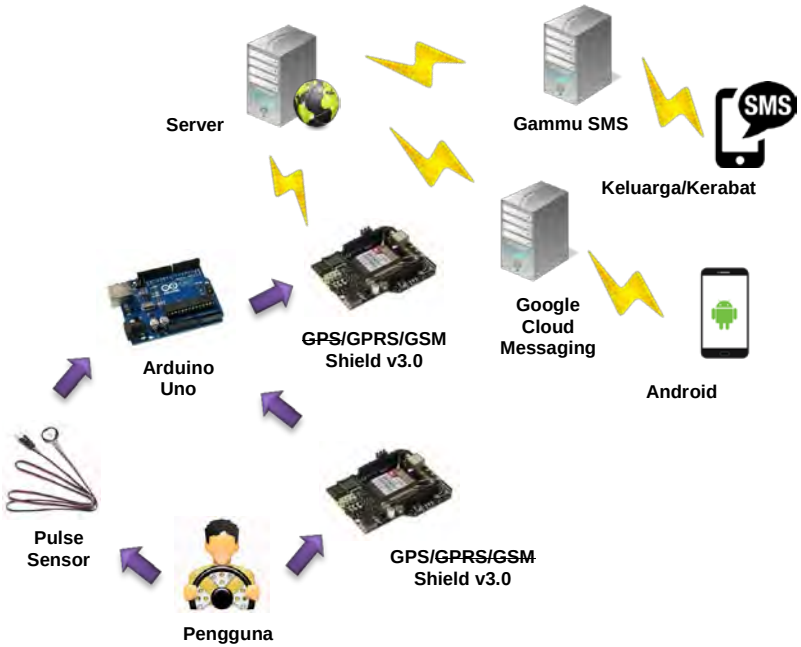
## 3.2. Arsitektur Sistem

Subbab ini membahas rancangan sistem, meliputi: Arsitektur Umum Sistem dan Rancangan Proses Sistem.

### 3.2.1. Arsitektur Umum Sistem

Berdasarkan hasil analisa deskripsi umum sistem dan secara garis besar arsitekur sistem dapat dibagi menjadi dua bagian, yakni pada mikrokontroler dan server. Modul mikrokontroler terdiri dari *Pulse* Sensor dan GPS/GPRS/GSM *Shield* V3.0 yang terintegrasi dengan mikrokontroler Arduino Uno. Sedangkan untuk server terdiri dari *web server* yang menangani penerimaan dan pengiriman *request* dengan modul komunikasi serta melakukan pengolahan data pada basis data dan komponen pendukung pengiriman notifikasi yang berupa Gammu SMS *Gateway* untuk notifikasi berupa pesan teks dan *Google Cloud Messaging* untuk notifikasi informasi rumah sakit terdekat

pada *smartphone* berbasis Android dari pengguna. Arsitektur sistem secara keseluruhan terlihat pada Gambar 3.1




**Gambar 3.1 Arsitektur Sistem**

Pengguna akan terhubung dengan mikrokontroler Arduino melalui sensor detak jantung yang dijepitkan oleh sensor di bagian tubuh pengguna, yaitu bagian daun telinga, untuk mendapatkan data detak jantung tanpa mengganggu pengguna dalam berkendara. Sensor detak jantung terhubung ke mikrokontroler Arduino dengan kabel. Dalam beroperasi, mikrokontroler Arduino akan secara bergantian menggunakan modul komunikasi GSM dan GPRS dengan modul GPS untuk mengambil data koordinat posisi pengguna dan kecepatan kemudian mengirimkan data tersebut ke server yang sudah disiapkan. Pengiriman data akan melalui jaringan GPRS dimana

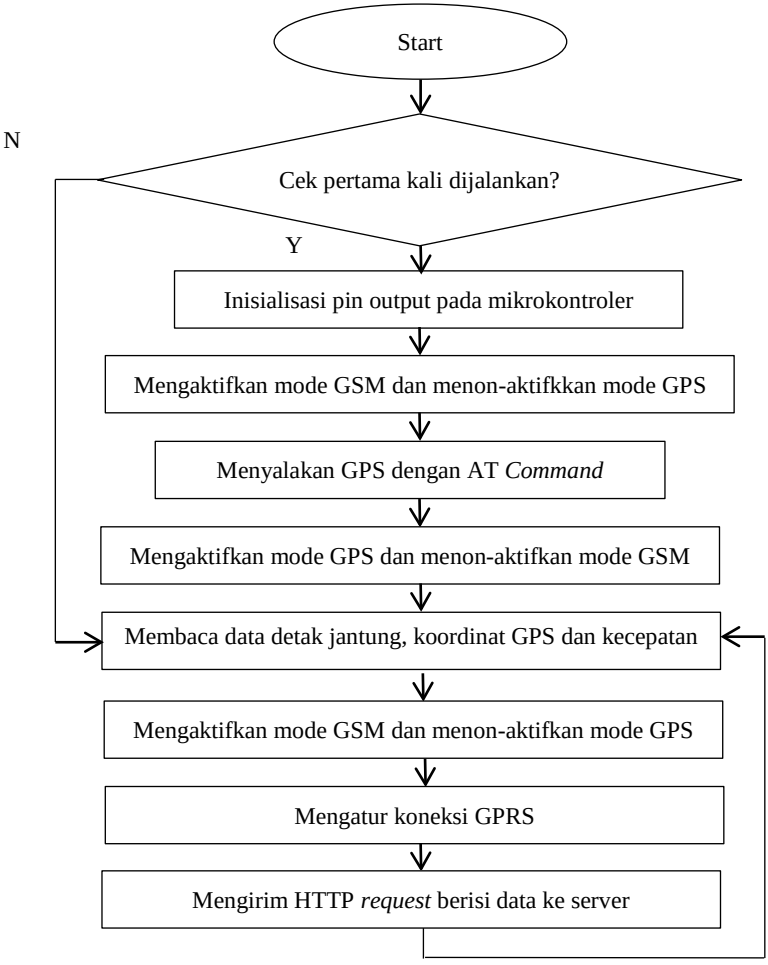
modul GSM dan GPRS akan mengirimkan HTTP *request* ke server dengan parameter data yang berupa koordinat posisi pengguna dan kecepatan pengguna akan dikirim setiap 2 menit ke server untuk dimasukkan ke dalam basis data dan dihitung percepatan yang dialami pengguna. Setelah data masuk ke dalam basis data, setiap 3 menit akan dilakukan pengecekan data oleh server apakah berdasarkan data pengguna yang dikirimkan pengguna masih dikategorikan dalam keadaan normal atau tidak. Pengecekan atau proses pengambilan keputusan ini diproses dengan *Fuzzy Logic* pada server dengan mengambil tiga parameter masukkan yaitu detak jantung, kecepatan dan percepatan pengguna. Apabila kondisi akhir yang didapatkan termasuk dalam keadaan yang tidak tergolong normal, maka notifikasi akan dikirim oleh sistem kepada pengguna dan kepada kerabat pengguna. Bentuk notifikasi dari sistem disediakan dalam dua bentuk, yaitu dalam bentuk pesan singkat yang berisi posisi terakhir pengguna dan notifikasi pada *smartphone* pengguna. Pesan singkat akan dikirimkan dengan mengunakan Gammu sebagai SMS *Gateway* yang berisi posisi terakhir pengguna dalam bentuk alamat yang didapatkan dari hasil *geocoding* dari *Google Geocoding* API. Sedangkan notifikasi pada *smartphone* pengguna didapat dari aplikasi yang sudah terpasang didalamnya yang memiliki fitur dari *Google Cloud Messaging* untuk mengirimkan *push notification*, yang apabila dibuka akan menampilkan peta dari lokasi pengguna saat itu dan beberapa fasilitas kesehatan terdekat yang ada dengan menggunakan fitur dari *Google Map* Android API untuk tampilan peta dan *Google Places* API untuk mendapatkan fasilitas kesehatan terdekat.

### 3.2.2. Rancangan Proses Sistem

Pada rancangan proses sistem akan dijelaskan mengenai proses yang berjalan sistem untuk memenuhi fungsionalitas dengan bentuk diagram alur.

#### 3.2.2.1. Rancangan Proses pada Perangkat Keras

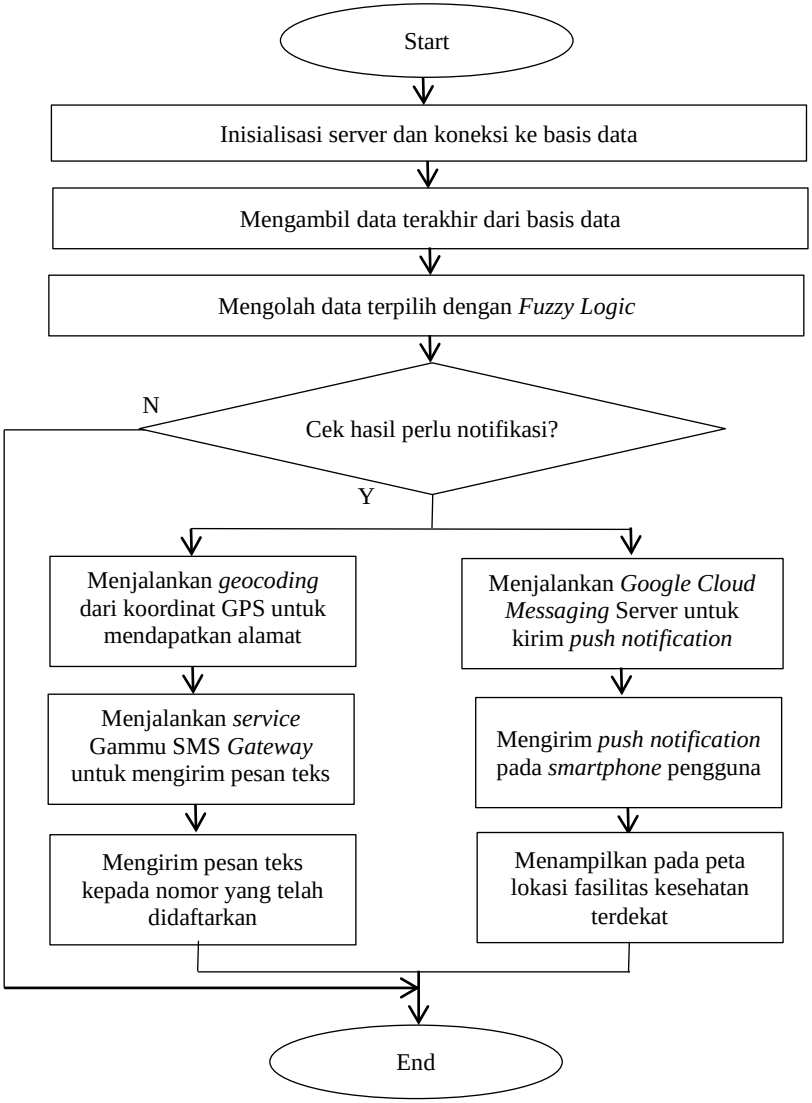
Proses ini memuat alur pembacaan data sensor yang berupa detak jantung dan koordinat posisi pengguna. Proses dimulai saat mikrokontroler Arduino dinyalakan dengan catu daya eksternal berupa baterai. Ketika dijalankan pertama kali, mikrokontroler akan melakukan proses inisialisasi berupa inisialisai pin – pin mana yang digunakan dalam proses pembacaan data yang diperlukan oleh modul GSM dan GPRS serta GPS untuk berfungsi. Agar modul GPS dapat dijalankan, maka harus digunakan AT *Command* untuk mengirim perintah dari mikrokontroler ke modul, sehingga perlu diaktifkan terlebih dahulu mode GSM untuk mengirim AT *Command* lalu kemudian AT *Command* untuk menyalakan GPS dinyalakan. Kemudian agar modul GPS dapat berfungsi, maka mode GSM harus dinon-aktikan. Setelah ini proses akan masuk ke fungsi perulangan Arduino. Lalu data detak jantung akan dibaca oleh sensor dengan fungsi yang sudah disediakan oleh sensor dan kecepatan serta koordinat GPS akan dibaca berdasarkan ekstrasi kalimat NMEA $GPRMC yang memiliki informasi titik latitude, titik longitude dan kecepatan. Setelah melewati proses pembacaan data, maka data siap untuk dikirim oleh modul GPRS. Proses ini dimulai dengan menon-aktifkan modul GPS dan mengaktifkan kembali modul GSM, lalu dilakukan pengaturan koneksi GPRS yang berupa APN (*Access Point Name*). Setelah itu dilakukan HTTP *request* ke server yang juga berfungsi sekaligus sebagai metode pengiriman data. Proses pembacaan data sampai pengiriman data akan terus dijalankan secara berulang oleh mikrokonrtoler Arduino. Keseluruhan rancangan proses ini dapat terlihat pada Gambar 3.2.




**Gambar 3.2 Diagram Alur Proses pada Mikrokontroler**

#### 3.2.2.2. Rancangan Proses pada Server

Proses ini memuat proses pengolahan dan pengiriman notifikasi yang dilakukan pada server. Proses ini dijalankan oleh server setiap 3 menit dimana dimulai dengan inisialisasi koneksi ke basis data untuk memastikan server dapat mengakses basis data dengan MySQL ODBC (*Open Database Connectivity*). Data yang diproses merupakan data yang terakhir didapatkan di basis data dan masih belum diproses dengan mengecek status data saat melakukan *query*. Kemudian data yang terpilih akan diolah dengan *Fuzzy Logic* untuk menentukan kondisi pengguna berdasarkan data terakhir termasuk dalam kategori hasil yang memerlukan notitfikasi atau tidak. Proses pengolahan ini dilakukan dengan bantuan *library* jFuzzyLogic untuk mengolah masukkan yang berupa detak jantung, percepatan dan kecepatan. Apabila data termasuk dalam kategori yang tidak memerlukan notifikasi maka data tidak akan diolah lebih lanjut dan keseluruhan proses ini berhenti, tapi sebaliknya jika termasuk maka data akan diteruskan untuk dilakukan pengiriman notfikasi berdasarkan atribut koordinat GPS yang dimiliki. Terdapat dua notifikasi yang akan dikirim, yang pertama adalah notifikasi berupa pesan teks. Konten dari pesan teks ini berupa data alamat dari koordinat GPS data yang merupakan posisi terakhir pengguna yang didapat dari hasil fitur *geocoding* oleh *Google Maps* API. Kemudian data ini dikirim dengan Gammu SMS *Gateway* ke nomor kerabat yang sudah didaftarkan. Notifikasi yang kedua adalah berupa notifikasi ke pengguna sendiri melalui *smartphone* pengguna yang bersistem operasi Android. Proses ini akan menjalankan *Google Cloud Messaging* Server untuk mengirim *push notification* dan akan menampilkan peta dengan lokasi fasilitas kesehatan terdekat dari posisi pengguna saat itu dengan menggunakan fitur pencarian lokasi terdekat dari *Google Places* API. Keseluruhan Rancangan proses dapat terlihat pada diagram alur padaGambar 3.3.



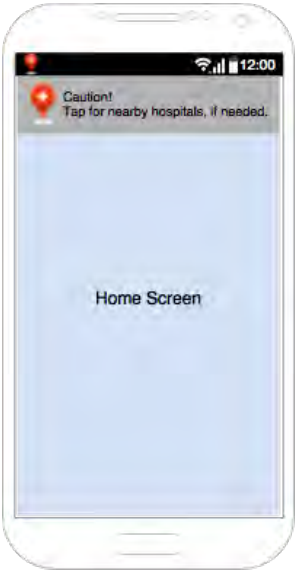

**Gambar 3.3 Diagram Alur Proses pada Server**

### 3.2.3. Rancangan Antar Muka Notifikasi

Rancangan antarmuka aplikasi diperlukan untuk memberikan gambaran umum kepada pengguna bagaimana sistem yang ada dalam aplikasi ini berinteraksi dengan pengguna.

#### 3.2.3.1. Rancangan Antar Muka Notifikasi di Halaman Utama di Android

Fungsionalitas dari aplikasi Android ini hanya berjalan ketika mendapat *push notification* dari *Google Cloud Messaging* Server. Rancangan antar muka dari *push notification* terlihat pada Gambar 3.4.

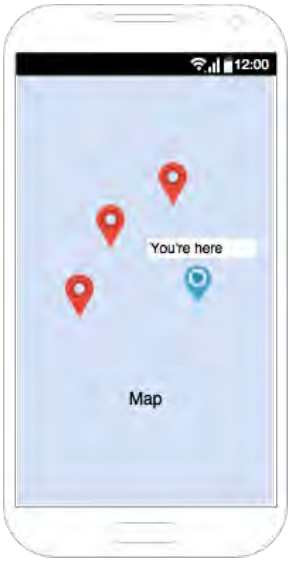



**Gambar 3.4 Rancangan Utama Notifikasi di Halaman Utama**

Pada tahapan ini *push notification* hanya mengirimkan notifikasi yang dapat terlihat halaman utama Android. *Push notification* akan memberi tahu pengguna bahwa kondisi terakhir sesuai hasil pengolahan di server , jika dibutuhkan perlu untuk ditindak lanjuti dengan bantuan lokasi fasilitas kesehatan terdekat.

### 3.2.3.2. Rancangan Antar Muka Peta Fasilitas Kesehatan Terdekat di Android

*Push notification* akan merujuk pengguna untuk melakukan interaksi dan membuka aktifiitas aplikasi ke peta yang memuat lokasi fasilitas kesehatan terdekat. Rancangan antar muka notifikasi lokasi terlihat pada Gambar 3.5.




**Gambar 3.5 Rancangan Antar Muka Lokasi *Maps* di Android**

Pada tahapan ini aplikasi akan menampilkan peta dari *Google Maps* Android API dan menampilkan lokasi dari beberapa fasilitas kesehatan terdekat dari posisi pengguna saat itu.

### 3.2.3.3. Rancangan Antar Muka Notifikasi Pesan Teks

Pada halaman ini nomor yang terdaftar di server akan menerima notifikasi informasi pengguna. Rancangan antar muka notifikasi pesan teks terlihat pada Gambar 3.6.


**Gambar 3.6 Rancangan Antar Muka Notifikasi Pesan Teks**

# BAB IV
# IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem yang telah dijabarkan pada bab sebelumnya.

## 4.1. Lingkungan Pembangunan

Dalam membangun sistem ini digunakan beberapa perangkat pendukung baik perangkat keras maupun perangkat lunak. Lingkungan pembangunan dijelaskan sebagai berikut.

### 4.1.1. Lingkungan Pembangunan Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan sistem ini adalah sebagai berikut:

- 1 Laptop dengan prosesor Intel(R) Core i5 CPU @ 2,40GHz dan memori (RAM) 4,00 GB.
- 1 Mikrokontroler Arduino Uno R3
- 1 Sensor detak jantung *Pulse* Sensor untuk pembacaan detak jantung.
- 1 DFRobot GPS/GPRS/GSM *Shield* v3.0 untuk komunikasi GSM/GPRS dan pembacaan posisi koordinat.
- 1 Modem GSM Huawei E173 untuk mendukung SMS *Gateway*.
- 1 *Smartphone* Android dengan sistem operasi Jellybean 4.3 untuk notifikasi pada aplikasi Android.

### 4.1.2. Lingkungan Pembangunan Perangkat Lunak

- Windows 8.0 64 bit sebagai sistem operasi.

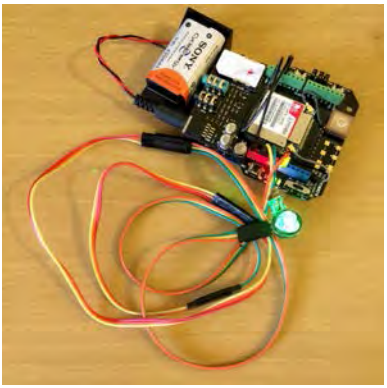
- Arduino IDE versi 1.6.4 untuk melakukan pemrograman mikrokontroler.
- NetBeans IDE versi 7.4 untuk melakukan pemrograman server Java.
- Android Studio versi 1.2.1.1 untuk melakukan pemrograman aplikasi Android.
- Gammu versi 1.33.0 sebagai SMS *Gateway* untuk notifikasi.
- XAMPP versi 5.6.8 untuk menyediakan layanan *web server*.
- MySQL versi 4.3.11 untuk menyediakan layanan basis data.
- PHP versi 5.6.8 untuk mendukung layanan web server.

## 4.2. Implementasi Sistem

Pada sistem ini perancangan sistem *diimplementasikan* ke dalam beberapa bagian yaitu implementasi perangkat keras, *web server*, *fuzzy logic*, server dan implementasi notifikasi yang berupa notifikasi pesan teks dan aplikasi Android.

### 4.2.1. Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam sistem ini merupakan rangkaian dari Mikrokontroler Arduino, GPS/GSM/GPRS *Shield* dan juga sensor detak jantung yang dirangkai dalam sebuah box. Tampilan dari modul – modul sebelum di masukkan ke dalam box dapat dilihat pada Gambar 4.1 dan hasil akhir dari perangkat keras yang sudah dikemas dalam box terlihat pada Gambar 4.2



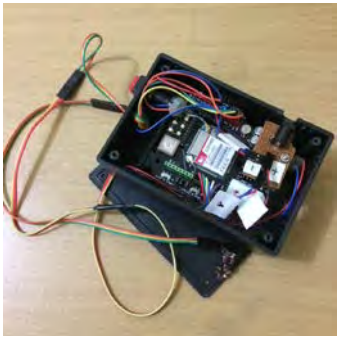
**Gambar 4.1 Modul Perangkat Keras Sebelum Dikemas**

**Gambar 4.2 Hasil Gabungan Mikrokontroler dan Sensor**

Didalam box ini disimpan rangkaian mikrokontroler Arduino yang sudah terhubung dengan GSM/GPRS/GPS *Shield* dan *Pulse* Sensor. Pada box ini, diberikan 2 tombol yang dapat diraih yaitu tombol untuk menyalakan dan mematikan *power* eksternal dan tombol untuk melakukan reset apabila diperlukan. Untuk gambar lebih dekat dari tombol *power* dan tombol *reset* dapat dilihat di Gambar 4.3, dan untuk indikator proses yang berupa LED dapat dilihat lebih dekat pada Gambar 4.4. Untuk isi rangkaian di dalam box ini dapat dilihat pada Gambar 4.5.

**Gambar 4.3 Tombol *Power* dan *Reset* pada Box**

**Gambar 4.4 Lampu Indikator LED pada Box**



**Gambar 4.5 Rangkaian dari Mikrokontroler Arduino dalam Box**

Perangkat ini dikembangkan dengan menggunakan tambahan catu daya eksternal berupa baterai dengan tegangan 9V dan arus 1A. Perangkat ini menggunakan serial untuk menjalankan AT *Command* untuk digunakan menjalankan fungsi – fungsi dari GSM dan GPRS *shield*. Dalam mendapatkan data detak jantung, sensor ini membaca detak jantung dalam bentuk frekuensi yaitu jumlah detak jantung per menit. Agar data sensor dan posisi koordinat dapat dibaca dan dikirim ke server, maka penggunaan GSM, GPRS dan GPS harus dijalankan secara bergantian dan diberi jeda waktu antar perintah. LED berwarna putih diatur sebagai indikator yang menandakan bahwa rangkaian ini sudah dijalankan. Kode sumber untuk bagian *setup* dari mikrokontroler Arduino terdapat pada Gambar 4.6

```
void setup() {
Serial.begin(9600);
pinMode(redPin, OUTPUT);
pinMode(greenPin, OUTPUT);
pinMode(yellowPin, OUTPUT);
pinMode(bluePin, OUTPUT);
pinMode(whitePin, OUTPUT);
digitalWrite(whitePin, HIGH);
interruptSetup();
pinMode(3,OUTPUT);
pinMode(4,OUTPUT);
pinMode(5,OUTPUT);
digitalWrite(5,HIGH);
delay(1500);
digitalWrite(5,LOW);
enableGSM_disableGPS();
delay(2000);
Serial.begin(9600);
delay(5000);
start_GPS();
enableGPS_disableGSM();
delay(2000);
fix = false;
reachedEnd = false;
}
```

**Gambar 4.6 Kode Sumber Bagian *Setup* Arduino**

Pada fungsi `interruptSetup()`, sensor akan membaca sinyal yang didapat dari pin analog 0 yang terhubung dengan sensor, untuk kemudian diatur pembacaan sinyal sebagai frekuensi. Pin 3, 4 dan 5 diatur sebagai pin *driver* yang digunakan oleh GSM/GPRS dan GPS. Arduino membutuhkan registrasi kartu sim ke jaringan *provider* untuk mendapatkan layanan GSM dan GPRS, oleh karena itu diatur untuk mengaktifkan mode GSM. Arduino juga diatur untuk langusng mendapatkan posisi koordinat ketika dijalankan, sehingga dibutuhkan mode GPS untuk diaktifkan. Pengaktifan mode GSM dan GPS dilakukan dengan mengatur tegangan pada pin 4 dan 3, seperti terlihat pada Gambar 4.7 berikut.

```
void enableGSM_disableGPS() {
digitalWrite(3,LOW);
digitalWrite(4,HIGH);
}
void enableGPS_disableGSM() {
digitalWrite(4,LOW);
digitalWrite(3,HIGH);
}
```

**Gambar 4.7 Kode Sumber Aktifasi Mode GSM dan GPS**

Posisi koordinat dari pengguna didapatkan dari kalimat – kalimat NMEA yang hanya bisa didapatkan ketika modul GPS sudah dinyalakan dan dilakukan *reset* secara otomatis dengan AT *Command* seperti pada Gambar 4.8.

```
void start_GPS() {
Serial.println("AT");
delay(2000);
Serial.println("AT+CGPSPWR=1");
delay(1000);
Serial.println("AT+CGPSRST=1");
delay(1000);
}
```

**Gambar 4.8 Kode Sumber Untuk *Reset* GPS**

Sistem dibutuhkan untuk mendapatkan data detak jantung dan posisi koordinat dari pengguna secara terus – menerus ketika perangkat dijalankan selama menggunakan kendaraan, oleh karena itu proses pengambilan data posisi dan detak jantung diulang terus menerus, dengan diatur juga LED warna merah sebagai indikator bahwa Arduino sedang melakukan pengambilan data GPS. Kode sumber Arduino bagian *loop* seperti pada Gambar 4.9 berikut.

```
void loop(){
delay(10000);
serialOutput() ;
if (QS == true){
serialOutputWhenBeatHappens();
QS = false;
} else  {
digitalWrite(blinkPin,LOW);
}
delay(20);
fix = false;
if(reachedEnd == true){
delay(5000);//GPS ready
start_GPS();
enableGPS_disableGSM();
delay(2000);
}
while(fix==false) {
Serial.print("Lat:");
latitude();
digitalWrite(whitePin, LOW);
digitalWrite(redPin, HIGH);
Serial.print("Lon:");
longitude();
Serial.print("Spd:");
speedotg();
if( !isinf(lattmp) && !isinf(oke2) && (spdtmp>=0)) {
fix=true;
}
delay(20);
digitalWrite(redPin, LOW);
}
Serial.println("Sending data:");
```

```
delay(2000);
enableGSM_disableGPS();
delay(20000);
start_GSM();
delay(30000);
send_GPRS();
reachedEnd = true;
delay(1000);
}
```

**Gambar 4.9 Kode Sumber Bagian *Loop* Arduino**

Dalam proses akuisisi data, Arduino akan membaca detak jantung terlebih dahulu dengan melihat ada tidaknya sinyal yang terbaca untuk kemudian diolah sesuai dengan waktu antar sinyal. Untuk proses pembacaan posisi koordinat dilakukan dengan ekstrasi kalimat $GPRMC yang ada pada kalimat NMEA. Kalimat $GPRMC dipilih karena terdapat atribut pada bagian isinya yang menandakan apakah posisi yang didapatkan sudah *valid* atau belum. Berikut struktur dari kalimat NMEA tipe $GPRMC terlihat pada Gambar 4.10.

```
$GPRMC,165432.000,A,0717.336410,S,11248.880943,E,0.0
00,0.0,170615,,,A*72

165432      : 165432.00  adalah format waktu UTC
A           : Menunjukkan bahwa data GPS valid
0717.336410 : Titik koordinat latitude
S           : Arah koordinat latitude (South)
11248.880943: Titik koordinat longitude
E           : Arah koordinat longitude (East)
0.000       : Kecepatan pengguna
```

**Gambar 4.10 Isi Kalimat NMEA $GPRMC**

Setelah data detak jantung, posisi koordinat dari pengguna didapat maka data tersebut akan dikirim ke server dengan menggunakan modul GPRS dan mengirimkan HTTP *Request* GET ke alamat *webserver*. LED berwarna hijau diatur untuk menjadi indikator ketika jaringan GPRS sedang diatur dan LED berwarna kuning diatur ketika data akan dikirimkan. Kode

program dari proses pengaturan modul GPRS dan pengiriman data terlihat pada Gambar 4.11.

```
void start_GPRS() {
digitalWrite(greenPin, HIGH);
Serial.println("AT+CREG?");
delay(2000);
Serial.println("AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\"");
delay(2000);
Serial.println("AT+SAPBR=3,1,\"APN\",\"telkomsel\"")
;
delay(2000);
Serial.println("AT+SAPBR=3,1,\"USER\",\"\"");
delay(2000);
Serial.println("AT+SAPBR=3,1,\"PWD\",\"\"");
delay(2000);
Serial.println("AT+SAPBR=1,1");
delay(5000);
Serial.println("AT+HTTPINIT");
delay(2000);
digitalWrite(greenPin, LOW);
}
void send_GPRS()
{
digitalWrite(yellowPin, HIGH);
Serial.print("AT+HTTPPARA=\"URL\",\"bey0nd.ddns.net:
8090/sensor/index.php?latitude=");
Serial.print(lattmp,7);
Serial.print("&longitude=");
Serial.print(lngtmp,7);
Serial.print("&bpm=");
Serial.print(BPM);
Serial.print("&speed=");
Serial.print(spdtmp,2);
Serial.println("\"");
delay(2000);
Serial.println("AT+HTTPPARA=\"CID\",1");
delay(2000);
Serial.println("AT+HTTPACTION=0");
delay(2000);
}
```

**Gambar 4.11 Kode Sumber Pengaturan GPRS**

### 4.2.1. Implementasi *Webserver*

Server yang dibangun pada sistem ini berfungsi sebagai komponen utama sistem. Pada server dibangun basis data yang berguna untuk menampung dan mengolah data. Server dibangun pada komputer yang terhubung dengan ISP, sehingga mendapat IP *public* dan dapat diakses dari internet. *Port* komunikasi yang digunakan yaitu pada *port* HTTP di *forward*, agar dapat diakses dari luar melalui *Dynamic* DNS. Selain itu pada server juga dilakukan proses penerimaan data dari mikrokontroler Arduino, proses penambahan dan penyimpanan data ke dalam basis data, serta proses pengambilan keputusan dengan menggunakan *Fuzzy Logic*. Server yang dibangun merupakan gabungan dari beberapa fungsionalitas utama yang dibangun pada beberapa lingkungan bahasa pengembangan untuk mendukung masing – masing kebutuhan. Bagian yang menjadi penghubung antara perangkat keras dan server dikembangkan dalam bentuk *webserver* dengan menggunakan XAMPP dan PHP. *Webserver* ini berfungsi untuk membaca data yang dikirimkan dari mikrokontroler Arduino, melakukan perhitungan dan pengambilan informasi dari data kecepatan sebelumya dari basis data untuk kemudian dimasukkan ke dalam basis data seperti pada Gambar 4.12.

```
<?php
$dbhost = 'bey0nd.ddns.net:8096';
$dbuser = 'root';
$dbpass = '';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
function convert2Degrees($str) {
$numbers = explode('.', $str);
$deg = (int)$numbers[0];
$minutes = (int)$numbers[1];
$minutes = $minutes / 10000000;
if ($deg < 100) {
$minutes = $minutes+$deg;
$deg = 0;
} else { $minutes += (int)($deg % 100);
$deg = (int)($deg / 100); }
```

```
$deg= $deg+($minutes/60);
return $deg; }
if ( !empty($_GET['latitude']) &&
!empty($_GET['longitude']) && !empty($_GET['bpm'])
&& !empty($_GET['speed'])  ) {
function getParameter($par, $default = null){
if (isset($_GET[$par]) && strlen($_GET[$par]))
return $_GET[$par];
elseif (isset($_POST[$par]) && strlen($_POST[$par]))
return $_POST[$par];
else return $default; }
$lat = getParameter("latitude");
$lat = round(convert2Degrees($lat), 7);
$lat = "-".$lat;
$lon = getParameter("longitude");
$lon = round(convert2Degrees($lon), 7);
$bpm = getParameter("bpm");
$vtmp = getParameter("speed");
$v2 = $vtmp * 1.852;
$vms = $v2 * 0.2777777777778 ;
$status = 0;
if(!$conn) { die('Could not connect: ' .
mysql_error()); }
$sql_v1 = 'SELECT timestamp, velocity FROM tbl_data
ORDER BY id DESC LIMIT 1 ';
mysql_select_db('ta_trial');
$mbambleh = mysql_query( $sql_v1, $conn );
$row = mysql_fetch_array($mbambleh, MYSQL_NUM);
$now = date("Y-m-d H:i:s");
$time1 = strtotime($row[0]); $time2 =
strtotime($now); $diff = ($time2 - $time1);
$a = ($vms - $row[1]) / $diff;
$a = round($a, 2);
if(!$conn) { die('Could not connect: ' .
mysql_error()); } $sql = 'INSERT INTO tbl_data '.
'(timestamp,latitude,longitude,velocity,acceleration
,heartrate,status) '. 'VALUES ( "'.$now.'", '.$lat.'
, '.$lon.', '.$v2.', '.$a.', '.$bpm.', '.$status.'
);'; mysql_select_db('ta_trial');
$retval = mysql_query( $sql, $conn );
if(! $retval ) { die('Could not enter data: ' .
mysql_error()); }} ?>
```

**Gambar 4.12 Kode Sumber *Webserver***

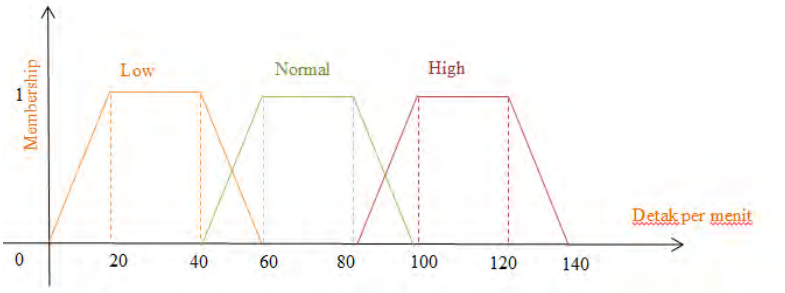
### 4.2.2. Implementasi *Fuzzy Logic*

Pada sistem ini terdapat 3 parameter yang menjadi masukkan untuk pengambilan keputusan atau yang biasa disebut dengan *Linguistic Variable* dalam pengolahan data dari mikrokontroler Arduino dengan *Fuzzy Logic* seperti yang dijelaskan pada Tabel 4.1 berikut.

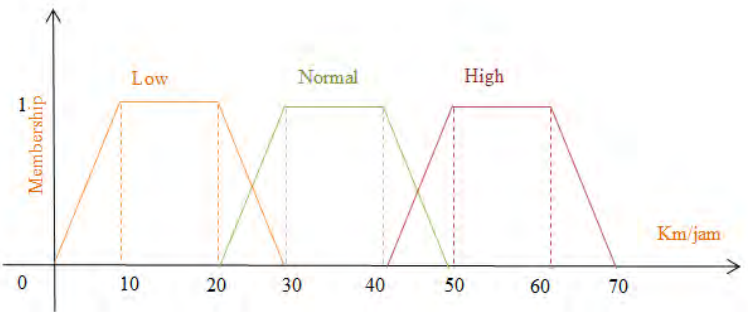
**Tabel 4.1 *Linguistic Variable* dan *Terms***

| No. | *Linguistic Variable* | *Terms* |
|---|---|---|
| 1 | Detak Jantung (hr) | *Low, Normal High* |
| 2 | Kecepatan (v) | *Low, Normal High* |
| 3 | Percepatan (a) | *Decreasing, Increasing* |

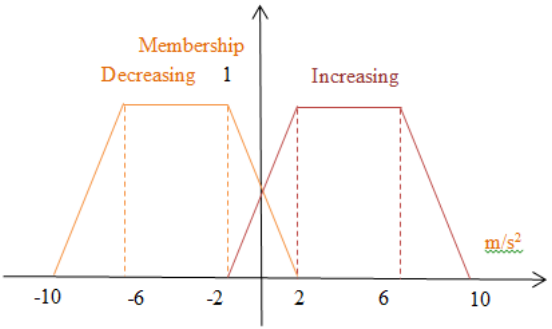
Setiap *Linguistic Variable* memiliki *Membership Function* yang menjadi penentuan masing – masing batas dan sebagai salah satu faktor yang menentukan keluaran dari perhitungan Masing – masing *Membership Function* akan dijelaskan dengan Grafik Keanggotaan pada Gambar 4.13, Gambar 4.14 dan Gambar 4.15.




**Gambar 4.13 Grafik Keanggotaan dari Detak Jantung**




**Gambar 4.14 Grafik Keanggotaan dari Kecepatan**




**Gambar 4.15 Grafik Keanggotaan dari Percepatan**

Masing – masing *Linguistic Variable* memiliki batas yang berbeda, agar kinerja dari sistem dapat optimal maka dilakukan normalisasi batas yang memetakan masing – masing batas ke dalam rentang 0 sampai 1. Untuk proses normalisasi ini akan digunakan *Min-Max Normalization* dengan rumus berikut.

$$\text{NewValue} = \frac{(\text{OldValue} - \text{OldMin})}{(\text{OldMax} - \text{OldMin})} * (\text{NewMax} - \text{NewMin}) + \text{NewMin} \qquad (4.1)$$

Perhitungan masing – masing *Linguistic Variable* dengan *Min-Max Normalization* akan dijelaskan pada Tabel 4.2, Tabel 4.3 dan Tabel 4.4.

**Tabel 4.2 Tabel Normalisasi Detak Jantung**

| OldValue | Old | | New | | NewValue |
|---|---|---|---|---|---|
| | Max | Min | Max | Min | |
| 0 | 140 | 0 | 1 | 0 | $= \frac{(0-0)}{(140-0)} * (1-0) + 0$ $= 0$ |
| | | | | | $= \frac{(20-0)}{(140-0)} * (1-0) + 0$ $= 0.142857143$ |
| | | | | | $= \frac{(40-0)}{(140-0)} * (1-0) + 0$ $= 0.285714286$ |
| | | | | | $= \frac{(60-0)}{(140-0)} * (1-0) + 0$ $= 0.428571429$ |
| | | | | | $= \frac{(80-0)}{(140-0)} * (1-0) + 0$ $= 0.571428571$ |
| | | | | | $= \frac{(100-0)}{(140-0)} * (1-0) + 0$ $= 0.714285714$ |
| | | | | | $= \frac{(120-0)}{(140-0)} * (1-0) + 0 =$ $0.857142857$ |
| | | | | | $= \frac{(140-0)}{(140-0)} * (1-0) + 0$ $= 1$ |

**Tabel 4.3 Tabel Normalisasi Kecepatan**

| OldValue | Old | | New | | NewValue |
|---|---|---|---|---|---|
| | Max | Min | Max | Min | |
| 0 | 70 | 0 | 1 | 0 | $= \frac{(0-0)}{(70-0)} * (1-0) + 0$ $= 0$ |

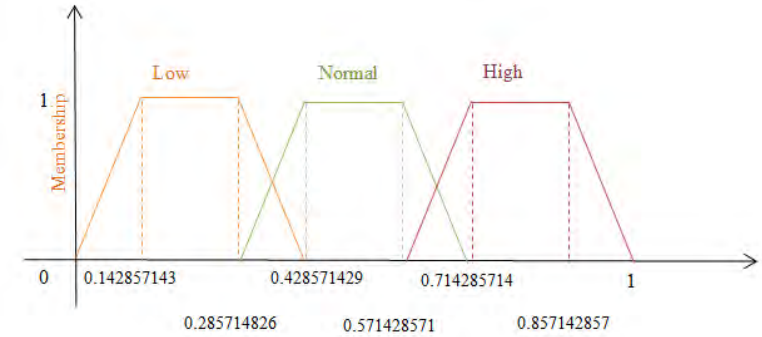
| | | | | | |
|---|---|---|---|---|---|
| | | | | | $= \frac{(10-0)}{(70-0)} * (1-0) + 0$ $= 0.142857143$ |
| | | | | | $= \frac{(20-0)}{(70-0)} * (1-0) + 0$ $= 0.285714286$ |
| | | | | | $= \frac{(30-0)}{(70-0)} * (1-0) + 0$ $= 0.428571429$ |
| | | | | | $= \frac{(40-0)}{(70-0)} * (1-0) + 0$ $= 0.571428571$ |
| | | | | | $= \frac{(50-0)}{(70-0)} * (1-0) + 0$ $= 0.714285714$ |
| | | | | | $= \frac{(60-0)}{(70-0)} * (1-0) + 0$ $= 0.857142857$ |
| | | | | | $= \frac{(70-0)}{(70-0)} * (1-0) + 0$ $= 1$ |

**Tabel 4.4 Tabel Normalisasi Percepatan**

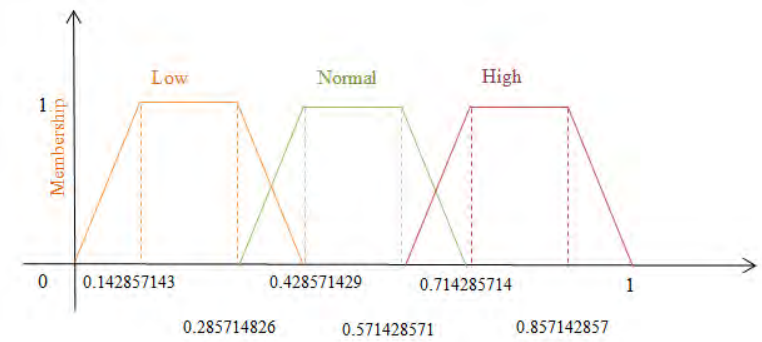
| OldValue | Old | | New | | NewValue |
|---|---|---|---|---|---|
| | Max | Min | Max | Min | |
| 0 | 6 | -6 | 1 | 0 | $= \frac{(-6-(-6))}{(6-(-6))} * (1-0) + 0$ $= 0$ |
| | | | | | $= \frac{(-4-(-6))}{(6-(-6))} * (1-0)$ $= 0.166667$ |
| | | | | | $= \frac{(-2-(-6))}{(6-(-6))} * (1-0) + 0$ $= 0.333333$ |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | $= \frac{(0-(-6))}{(6-(-6))} * (1-0) + 0$ $= 0.5$ |
| | | | | | $= \frac{(2-(-6))}{(6-(-6))} * (1-0) + 0$ $= 0.666667$ |
| | | | | | $= \frac{(4-(-6))}{(6-(-6))} * (1-0) + 0$ $= 0.833333$ |
| | | | | | $= \frac{(6-(-6))}{(6-(-6))} * (1-0) + 0$ $= 1$ |

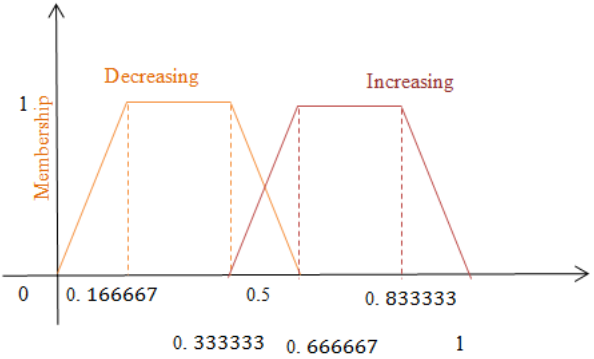
Grafik Keanggotan darin masing – masing *Linguistic Variable* setelah dilakukan normalisasi berubah mengikut nilai maksimum dan minimum baru, yaitu 1 dan 0. Masing – masing Grafik Keanggotaan setelah dilakukan proses normalisasi terlihat pada Gambar 4.16, Gambar 4.17 dan Gambar 4.18.




**Gambar 4.16 Grafik Keanggotaan dari Detak Jantung setelah Normalisasi**




**Gambar 4.17 Grafik Keanggotaan dari Kecepatan setelah Normalisai**




**Gambar 4.18 Grafik Keanggotaan dari Percepatan setelah Normalisasi**

Dengan nilai yang sudah diperoleh dari hasil normalisasi ini, parameter – parameter ini akan dievaluasi ke dalam hasil akhir melalui aturan – aturan yang seperti terlihat pada Tabel 4.5.

**Tabel 4.5 Tabel *Fuzzy Rules***

| |
|---|
| RULE 1 : IF heartrate IS low AND velocity IS low AND acceleration IS decreasing THEN condition IS normal. |
| RULE 2 : IF heartrate IS low AND velocity IS low AND |

| |
|---|
| acceleration IS increasing THEN condition IS normal. |
| RULE 3 : IF heartrate IS low AND velocity IS normal AND acceleration IS decreasing THEN condition IS normal. |
| RULE 4 : IF heartrate IS low AND velocity IS normal AND acceleration IS increasing THEN condition IS normal. |
| RULE 5 : IF heartrate IS low AND velocity IS high AND acceleration IS decreasing THEN condition IS potential. |
| RULE 6 : IF heartrate IS low AND velocity IS high AND acceleration IS increasing THEN condition IS potential. |
| RULE 7 : IF heartrate IS normal AND velocity IS low AND acceleration IS decreasing THEN condition IS normal. |
| RULE 8 : IF heartrate IS normal AND velocity IS low AND acceleration IS increasing THEN condition IS normal. |
| RULE 9 : IF heartrate IS normal AND velocity IS normal AND acceleration IS decreasing THEN condition IS normal. |
| RULE 10 : IF heartrate IS normal AND velocity IS normal AND acceleration IS increasing THEN condition IS normal. |
| RULE 11 : : IF heartrate IS normal AND velocity IS high AND acceleration IS decreasing THEN condition IS normal. |
| RULE 12 : IF heartrate IS normal AND velocity IS high AND acceleration IS decreasing THEN condition IS normal. |
| RULE 13 : IF heartrate IS normal AND velocity IS high AND acceleration IS increasing THEN condition IS normal. |
| RULE 14 : IF heartrate IS high AND velocity IS low AND acceleration IS decreasing THEN condition IS potential. |
| RULE 15 : IF heartrate IS high AND velocity IS low AND acceleration IS increasing THEN condition IS potential. |
| RULE 16 : IF heartrate IS high AND velocity IS normal AND acceleration IS decreasing THEN condition IS potential. |

| |
|---|
| RULE 17 : IF heartrate IS high AND velocity IS normal AND acceleration IS increasing THEN condition IS potential. |
| RULE 18 : IF heartrate IS high AND velocity IS high AND acceleration IS decreasing THEN condition IS dangerous. |
| RULE 19: IF heartrate IS high AND velocity IS high AND acceleration IS increasing THEN condition IS dangerous. |

### 4.2.3. Implementasi Server

Pada sistem ini server yang digunakan berbasis Java dan dijalankan secara berperiodik, yaitu setiap 3 menit untuk melakukan pengecekan terhadap data terakhir yang ada pada basis data. Server ini bertugas untuk mendapatkan data terakhir dari basis data, mengolah data dengan *Fuzzy Logic* dan kemudian apabila memenuhi kondisi yang sudah ditentukan akan melakukan pengiriman notifikasi yang berupa notifikasi berupa pesan teks dan notifikasi pada aplikasi Android. Server utama berjalan pada kelas “Server” dan pertama kali berjalan akan mengambil data terakhir dari basis data dan di proses dengan menggunakan fungsi dari kelas “Fuzzy” yang memuat library dan semua fungsi dari *Fuzzy Logic*. Pemanggilan fungsi dari kelas “Fuzzy” yaitu `getFuzzyResult()` akan memproses masing - masing data yang memiliki 3 parameter masukkan, yaitu detak jantung, kecepatan dan percepatan. Apabila didapati hasil dari pengolah dengan menggunakan *Fuzzy* termasuk dalam kondisi “Potential” atau “Dangerous” maka hasil akan ditampung sementara untuk kemudian dilanjutkan dengan proses pengiriman notifikasi. Pada setiap data yang sudah diproses atribut status yang merupakan atribut dari tabel basis data akan diubah dengan yang baru dimana nilai tersebut menunujukkan bahwa data tersebut sudah diolah dan tidak perlu diolah lagi jika belum ada data terbaru lagi dari sensor. Kode sumber pengambilan data dapat dilihat pada Gambar 4.19.

```
String query = "SELECT * FROM tbl_data WHERE status
= 0 ORDER BY id DESC LIMIT 1 ";
con =
DriverManager.getConnection(url, user, password);
stmt = con.createStatement();
rs = stmt.executeQuery(query);
ArrayList<ArrayList<String>> data =
new ArrayList<>();
while (rs.next())
{
ArrayList<String> row = new ArrayList<>();
int id = rs.getInt("id");
String timestamp = rs.getString("timestamp");
String latitude = rs.getString("latitude");
String longitude = rs.getString("longitude");
float velocity = rs.getFloat("velocity");
float acceleration = rs.getFloat("acceleration");
float heartrate = rs.getFloat("heartrate");
int status = rs.getInt("status");
row.add(Integer.toString(id));
row.add(timestamp);
row.add(Float.toString(heartrate));
row.add(Float.toString(velocity));
row.add(Float.toString(acceleration));
row.add(latitude);
row.add(longitude);
data.add(row);
}
ArrayList<ArrayList<String>> resultList =
new ArrayList<ArrayList<String>>();
for(ArrayList<String>            result            :
Fuzzy.getFuzzyResult(data))
{
ArrayList<String> selectedResult =
new ArrayList<String>();
if(result.get(2).equalsIgnoreCase("Potential"))
{
selectedResult.add(result.get(0));
selectedResult.add(result.get(1));
selectedResult.add(result.get(2));
resultList.add(selectedResult);
System.out.println("Potential");
}
```

```
else if(result.get(2).equalsIgnoreCase("Dangerous"))
{
selectedResult.add(result.get(0));
selectedResult.add(result.get(1));
selectedResult.add(result.get(2));
resultList.add(selectedResult);
System.out.println("Dangerous");
} }
```

**Gambar 4.19 Kode Sumber Pengambilan Data dan Hasil Pengolahan *Fuzzy* pada Server**

### 4.2.4. Implementasi Notifikasi SMS

Bentuk keluaran dari sistem ini adalah adanya notifikasi yang ditujukan pada nomor keluarga yang terdafttar untuk memberitahukan posisi pengguna ketika masuk dalam kategori "Dangerous" atau "Potential" yang merupakan hasil dari pengambilan keputusan *Fuzzy Logic*. Hal ini diimplementasikan dengan menggunakan *service* dari Gammu untuk mengirimkan pesan teks secara otomatis. Modem Huawei E173 digunakan sebagai alat pendukung untuk menjalankan Gammu. Agar Gammu dapat dijalankan maka dibutuhkan pengaturan modem yang tepat di file "smsdrc" yang meruapakan file konfigurasi *service*. Yang perlu diperhatikan dan diubah nilanya dalam file konfigurasi tersebut yaitu "device" yang merupakan *port* dari modem yang sedang terhubung pada fisik server, "connection" merupakan tipe koneksi yang disediakan oleh Gammu yang tersedia di website Gammu, "service" merupakan tipe basis data yang digunakan untuk menampung basis data default dari Gammu, pengaturan basis data seperti "user" untuk nama pengguna, "password" kode rahasia dari basis data jika ada, "pc" merupakan *host* tempat basis data disimpan, "database" merupakan nama dari basis data dan "driver" yang merupakan jenis *service* basis data yang digunakan. Kode sumber pengaturan file konfigurasi "smsdrc" yang disesuaikan dengan tipe modem yang dipakai dapat dilihat pada Gambar 4.20.

```
# Gammu configuration, this section is like section
"gammu" in "gammurc" file,
# see gammurc(5) for documentation.
[gammu]
device = com10:
#model = 6110
connection = at115200
# SMSD service to use, one of FILES, MYSQL, PGSQL,
DBI
service = SQL
# Database backends congfiguration
user = root
password =
pc = localhost
# pc can also contain port or socket path after
colon (eg. localhost:/path/to/socket)
database = ta_trial
# DBI configuration
driver = native_mysql
```

**Gambar 4.20 Kode Sumber Konfigurasi Gammu**

Untuk menjalankan proses pengiriman notifikasi pesan teks dengan Gammu, perlu dipastikan *service* Gammu sudah dijalankan di sistem operasi, kemudian proses pengiriman sistem ini dengan cara menambah baris data pada tabel “outbox” yang ada pada basis data, yang mana secara otomatis pada *trigger* yang sudah dijalankan dari basis data SQL *default* dari Gammu akan mengirim pesan teks sesuai dengan file konfigurasi. Konten dari pesan teks yang dikirimkan merupakan alamat posisi pengguna terakhir yang merupakan hasil dari pemanggilan *Google Maps* API yang dilakukan di server utama dengan menggunakan parameter masukkan posisi koordinat dari data yang diproses. Kode sumber dari proses *geocoding* dan eksekusi perintah pada server dapat dilihat pada Gambar 4.21.

```
lat = data.get(sel_idx).get(5);
lng = data.get(sel_idx).get(6);
String api_address =
"https://maps.googleapis.com/maps/api/geocode/json?"
;
```

```
String param_location =
"latlng=" + lat + "," + lng;
String param_location_type =
"&location_type=GEOMETRIC_CENTER|RANGE_INTERPOLATED|
APPROXIMATE|ROOFTOP";
String param_result_type =
"&result_type=street_address|postal_code|country";
String param_key =
"&key=AIzaSyDfffxaKhH2Eds1IPS5oiS1XiQcZ69VQoY";
String apiURL =
api_address + param_location + param_location_type +
param_result_type + param_key;
String jsonString = callURL(apiURL);
JSONArray tagResults;
try
{
JSONObject jsonResponse =
new JSONObject(jsonString);
tagResults = jsonResponse.getJSONArray("results");
JSONObject resultObj = tagResults.getJSONObject(0);
addr = resultObj.getString("formatted_address");
System.out.println(addr);
}
catch (JSONException e)
{
e.printStackTrace();
}
text = "Caution at " + addr ;
String SMSquery =
"INSERT INTO outbox (DestinationNumber, TextDecoded,
CreatorID)"
+ " VALUES ('" + phoneNumber + "', '" + text + "',
'Gammu 1.33.0')";
stmt.executeUpdate(SMSquery);
```

**Gambar 4.21 Kode Sumber Proses *Geocoding* dan Eksekusi Pengiriman Notfikasi Pesan Teks**

Hasil dari implementasi notifikasi pesan teks ketika dieksekusi dapat dilihat pada Gambar 4.22.


**Gambar 4.22 Tampilan Notifikasi SMS**

### 4.2.5. Implementasi Notifikasi Android

Pada sistem ini juga diimplementasikan notifikasi pada *smartphone* dengan sistem operasi Android pengguna. Tujuan dari notifikasi ini untuk memberitahu pengguna akan kondisi terakhir pengguna dan memberi informasi kepada pengguna lokasi beberapa fasilitas kesehatan terdekat dari posisi pengguna saat itu. Notifikasi pada *smartphone* Android ini diimplementasikan dengan *Google Cloud Messaging* untuk mengirim *push notification* ke *smartphone* pengguna dan *Google Maps Android* API untuk mengimplementasikan peta pada *smartphone* dan mendapatkan lokasi fasilitas kesehatan terdekat.

*Google Cloud Messaging* diimplementasikan di server yang tersedia dan digunakan untuk mengirim pesan lewat GCM. Beberapa hal yang perlu diperhatikan dalam proses pengiriman pesan yaitu URL dari web API untuk mengirim pesan lewat *Google Cloud Messaging*, *array fields* yang akan diparse sebagai JSON yang berisi token registrasi dari aplikasi Android dan pesan yang dikirimkan, API *key* untuk server di project pada *web developer* Google. Pesan JSON dikirim ke *Google Cloud Messaging* web API dengan “curl”. Kemudian variabel `$result`

akan berisi file JSON yang memuat informasi apakah pengiriman berhasil atau tidak. Kode sumber dari *Google Cloud Messaging* yang berada di server dapat dilihat pada Gambar 4.23.

```
<?php
function  sendPushNotificationToGCM($registatoin_ids,
$message) {
$url = 'https://gcm-http.googleapis.com/gcm/send';
$target = $registatoin_ids[0];
$fields = array(
'to' => $target,
'data' => $message,
);
define("GOOGLE_API_KEY","AIzaSyDfffxaKhH2Eds1IPS5oiS1
XiQcZ69VQoY");
$headers = array(
'Authorization: key=' . GOOGLE_API_KEY,
'Content-Type: application/json'
);
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch,CURLOPT_POSTFIELDS,json_encode($field
s));
$result = curl_exec($ch);
if  ($result  ===  FALSE)  {  die('Curl  failed:  '  .
curl_error($ch)); }
curl_close($ch);
return $result;
}?>
```

**Gambar 4.23 Kode Sumber *Google Cloud Messaging Web* API**

Dalam implementasinya, pengiriman dapat dilakukan dengan dua cara yaitu melalui *interface* yang disediakan di web atau langsung mengirimkan *form* POST ke halaman PHP yang ada di server. Untuk prosesnya, pertama akan dicek pada URL apakah ada variabel GET ‘push’, kemudian server akan

mengambil token registrasi di file “GCMRegID” yang memuat token registrasi dan mengecek data *form* POST dengan nama ‘message’ yang merupakan pesan yang ingin dikirim. Jika token registrasi sudah ada dan ‘message’ sudah memiliki konten, maka dilakukan pemanggilan fungsi untuk pengiriman *push notification*. Kode sumber untuk fungsi sendPushNotification() dapat dilihat pada Gambar 4.24.

```
$pushStatus = "";
if(!empty($_GET["push"])) {
$gcmRegID  = file_get_contents("GCMRegId.txt");
$pushMessage = $_POST["message"];
if (isset($gcmRegID) && isset($pushMessage)) {
$gcmRegIds = array($gcmRegID);
$message = array("m" => $pushMessage);
$pushStatus  =  sendPushNotificationToGCM($gcmRegIds,
$message);
}}
```

**Gambar 4.24 Kode Sumber Fungsi Pengiriman *Push Notification***

Agar *Google Cloud Messaging* pada *webserver* dan aplikasi Android dapat saling berkomunikasi, dibutuhkan token registrasi yang didapapatkan ketika aplikasi Android dijalankan pertama kali. Pada tahapan ini *webserver* akan melakukan pengecekan apakah pada URL terhadap variabel GET ‘shareRegId’. Jika ada, maka akan dilakukan pengecekan data POST pada variabel ‘regId’ dan nilainya disimpan di file “GCMRegId.txt”. Kode sumber dari penerimaan token registrasi yang dilakukan oleh *we server* dapat dilihat pada Gambar 4.25.

```
if(!empty($_GET["shareRegId"])) {
$gcmRegID  = $_POST["regId"];
file_put_contents("GCMRegId.txt",$gcmRegID);
echo "Ok!";
exit;
}
```

**Gambar 4.25 Kode Sumber Penerimaan Token Registrasi Aplikasi**

Aplikasi Android dalam implementasi *Google Cloud Messaging* ini berlaku sebagai *client* yang menerima pesan dari server. Pertama yang dilakukan adalah mengunduh file konfigurasi yang disediakan di *web developer* Android dan diletakkan di dalam folder '`/app`' dari project atau aplikasi. Kemudian agar *Google Cloud Messaging* dapat berjalan dibutuhkan penambahan *dependencies* dari Gradle untuk modul aplikasi yaitu *Google Play Services*. Penambahan pengaturan agar memiliki akses dari aplikasi untuk menggunakan fitur Android dilakukan di "AndroidManifest.xml". Pemberian akses `android.permission.internet` digunakan agar aplikasi dapat mengakses internet untuk mengirim token registrasi dari *Google Cloud Messaging* ke *webserver*, pemberian akses `android.permission.wake_lock` bertujuan agar pesan tetap dapat diterima ketika aplikasi sedang tidak dijalankan dan *smartphone* tidak dalam kondisi *standby*, deklarasi "GcmReceiver" pada tag <`receiver`> bertujuan untuk mengatur penerimaan pesan terkirim dari *Google Cloud Messaging* ke aplikasi Android, deklarasi "GcmListenerService" pada tag <`service`> mengacu ke kelas "MyGcmListenerService" yang ada pada aplikasi yang berperan sebagai *service* untuk mengolah setiap pesan yang diterima dari *Google Cloud Messaging*, seperti menampilkan notifikasi, deklarasi "InstanceIDListener" pada tag <`service`> mengacu pada kelas "MyInstanceIDListenerService" yang berperan sebagai *service* untuk memantau status registrasi aplikasi Android dengan *Google Cloud Messaging* server, dimana nantinya jika diperlukan registrasi maka *service* ini akan mengirim notifikasi ke *intent* "RegistrationIntentService". Penambahan pengaturan dapat dilihat pada Gambar 4.26.

```
<uses-permission
android:name="android.permission.INTERNET"/>
<uses-
permissionandroid:name="android.permission.WAKE_LOCK
"/>
<uses-permission
```

```
android:name="com.google.android.c2dm.permission.REC
EIVE" />
<receiver
android:name="com.google.android.gms.gcm.GcmReceiver
"
android:exported="true"
android:permission="com.google.android.c2dm.permissi
on.SEND">
<intent-filter>
<action
android:name="com.google.android.c2dm.intent.RECEIVE
" />
<category
android:name="gcm.play.android.samples.com.gcmquicks
tart" />
</intent-filter>
</receiver>
<service
android:name=".MyGcmListenerService"
android:exported="false" >
<intent-filter>
<action
android:name="com.google.android.c2dm.intent.RECEIVE
" /></intent-filter>
</service>
<service
android:name=".MyInstanceIDListenerService"
android:exported="false" >
<intent-filter>
<action
android:name="com.google.android.gms.iid.InstanceID"
/></intent-filter>
</service>
```

**Gambar 4.26 Kode Gambar Pengaturan AndroidManifest.xml**

Agar *Google Cloud Messaging* dapat berjalan dengan lancar pada aplikasi Android maka perlu dilakukan pengecekan terhadap *Google Play Service* APK setiap kali *Google Play Services* SDK akan diakes oleh *smartphone* Android penguna. Implementasi pengecekan ini berada pada kelas “MainActivity” dan dapat dilihat pada Gambar 4.27.

```
private boolean checkPlayServices() {
int resultCode =
GooglePlayServicesUtil.isGooglePlayServicesAvailable
(this);
if (resultCode != ConnectionResult.SUCCESS) {
if
(GooglePlayServicesUtil.isUserRecoverableError(resul
tCode)){
GooglePlayServicesUtil.getErrorDialog(resultCode,
this, PLAY_SERVICES_RESOLUTION_REQUEST).show();
} else {
Log.i(TAG, "This device is not supported.");
finish();
} return false;
} return true;
}
```

**Gambar 4.27 Kode Sumber Pengecekan *Google Play Service* APK**

Pada aplikasi Android terdapat kelas "QuickstartPreferences" yang berisi variabel-variabel statis yang digunakan untuk menyimpan konfigurasi dari aplikasi. Kelas ini akan digunakan untuk membuat sebuah *instance* "PreferenceManager" yang digunakan untuk menyimpan konfigurasi dari aplikasi. Konfigurasi yang disimpan adalah nilai *boolean* dari *string* `REGISTRATION_COMPLETE` yang menandakan apabila aplikasi sudah terdaftar ke *Google Cloud Messaging connection server*, `SENT_TOKEN_TO_SERVER` yang menandakan apakah token registrasi sudah dikirimkan ke *web server* yang disediakan untuk mengirim pesan lewat GCM ke aplikasi android ini. *String* `APP_SERVER_URL` mendefinisikan URL yang digunakan untuk mengirimkan token registrasi. Implementasi kelas "QuickstartPreferences" dapat dilihat pada Gambar 4.28.

```
public class QuickstartPreferences {
public static final String SENT_TOKEN_TO_SERVER =
"sentTokenToServer";
public static final String REGISTRATION_COMPLETE =
"registrationComplete";
```

```
static final String APP_SERVER_URL =
"http://bey0nd.ddns.net:8090/server126/register.php?
shareRegId=1";
}
```

**Gambar 4.28 Kode Sumber Kelas *QuickstartPreferences***

Aplikasi Android yang ingin menerima pesan harus mendaftarkan diri dulu ke *Google Cloud Messaging connection server*. Ketika sebuah aplikasi Android berhasil registrasi, sebuah token registrasi akan dikirimkan dari *Google Cloud Messaging connection server* ke Aplikasi android. Token registrasi inilah yang digunakan sebagai nomor alamat untuk mengirim pesan ke aplikasi ini. Token registrasi ini lalu dikirimkan ke *webserver* yang ingin mengirimkan pesan lewat *Google Cloud Messaging* ke aplikasi Android. Proses registrasi ke GCM *connection* server dilakukan pada kelas "RegistrationIntentService" pada fungsi `onHandleIntent()`. Fungsi `onHandleIntent()` ini akan dijalankan setiap kali *intent* ini dibuat pada pemanggilan fungsi `onTokenRefresh()` di kelas "MyInstanceIDListenerService". Implementasi dari fungsi `onTokenRefresh()` dapat dilihat pada Gambar 4.29.

```
public void onTokenRefresh() {
Intent intent = new Intent(this,
RegistrationIntentService.class);
startService(intent);
}
```

**Gambar 4.29 Kode Sumber Fungsi *onTokenRefresh()***

Pada kelas "RegistrationIntentService", terdapat "sharedReferences" yang berisi keterangan apakah aplikasi sudah berhasil registrasi dan apakah token registrasi sudah dikirim ke *webserver* aplikasi yang digunakan untuk mengirim pesan. Potongan kode sumber dari "sharedReferences" yang memuat informasi token registrasi sudah didapatkan untuk pengiriman pesan dapat dilihat pada Gambar 4.30.

```
sharedPreferences=
PreferenceManager.getDefaultSharedPreferences(this);
InstanceID instanceID = InstanceID.getInstance(this);
String token=
instanceID.getToken(getString(R.string.mygcm_senderid
),
GoogleCloudMessaging.INSTANCE_ID_SCOPE, null);
Log.i(TAG, "GCM Registration Token: " + token);
```

**Gambar 4.30 Kode Sumber *sharedPreferences***

Pengecekan pada konfigurasi diperlukan untuk melihat apakah token registrasi sudah pernah dikirim ke *web server* untuk mengirim pesan, jika belum maka fungsi `sendRegistrationtoServer()` akan dipanggil. Setelah itu konfigurasi `SENT_TOKEN_TO_SERVER` akan diubah nilainya menjadi *true*. Proses pengecekan ini dapat dilihat pada Gambar 4.31.

```
if(sharedPreferences.getBoolean(QuickstartPreferences
.SENT_TOKEN_TO_SERVER, false) == false) {
String result = sendRegistrationToServer(token);
Log.i(TAG, "sending token to app server..");
Toast.makeText(getApplicationContext(), result,
Toast.LENGTH_LONG).show();
}
sharedPreferences.edit().putBoolean(QuickstartPrefere
nces.SENT_TOKEN_TO_SERVER, true).apply();
```

**Gambar 4.31 Kode Sumber Pengecekan Token Registrasi**

Setelah semua proses diatas sudah dijalankan, maka hal yang terakhir pelu dilakukan agar aplikasi Android dan *Google Cloud Messaging pada web server* dapat saling terhubung melalui token registrasi adalah dengan mengirim token tersebut ke *web server*. Yang harus dilakukan dalah menyiapkan *POST form* data ke URL yang sudah dispesifikasikan pada "QuickstartPreferences". Kode sumber dari pengiriman token registrasi ke server agar antara aplikasi dan server dapat saling terhubung dapat dilihat pada Gambar 4.32.

```
private String sendRegistrationToServer(String token)
{
String result = "";
Map paramsMap = new HashMap();
paramsMap.put("regId", token);
try { URL serverUrl = null;
try { serverUrl = new
URL(QuickstartPreferences.APP_SERVER_URL);
} catch (MalformedURLException e) {
Log.e("AppUtil", "URL Connection Error: "
QuickstartPreferences.APP_SERVER_URL, e);
result = "Invalid URL: " +
QuickstartPreferences.APP_SERVER_URL;
}
StringBuilder postBody = new StringBuilder();
Iterator iterator = paramsMap.entrySet().iterator();
while (iterator.hasNext()) {
Map.Entry param = (Map.Entry) iterator.next();
postBody.append(param.getKey()).append('=').append(pa
ram.getValue());
if (iterator.hasNext()) {
postBody.append('&');
} }
String body = postBody.toString();
byte[] bytes = body.getBytes();
HttpURLConnection httpCon = null;
try {
httpCon = (HttpURLConnection)
serverUrl.openConnection();
httpCon.setDoOutput(true);
httpCon.setUseCaches(false);
httpCon.setFixedLengthStreamingMode(bytes.length);
httpCon.setRequestMethod("POST");
httpCon.setRequestProperty("Content-
Type","application/x-www-form-urlencoded;charset=UTF-
8");
OutputStream out = httpCon.getOutputStream();
out.write(bytes);
out.close();
int status = httpCon.getResponseCode();
if (status == 200) { result = "Token shared with
Application Server. Token: " + token; } else {
result = "Post Failure." + " Status: " + status;
} } finally {
```

```
if (httpCon != null) {
httpCon.disconnect();
} }
} catch (IOException e) {
result = "Post Failure. Error in sharing with App
Server.";
Log.e("AppUtil", "Error in sharing with App Server: "
+ e);
} return result;
}
```

**Gambar 4.32 Kode Sumber Pengiriman Token Registrasi**

Setelah token registrasi dari aplikasi Android sudah terhubung dengan *Google Cloud Messaging* di *webserver* maka diimplentasikan kelas "MyGCMListenerService" yang merupakan implementasi dari kelas abstrak "GcmListenerService" yang digunakan unttuk menerima pesan dari GCM. Proses penerimaan pesan dilakukan pada fungsi `onMessageReceived()`. Pada funsgi ini, setiap pesan yang masuk akan diolah dalam fungsi `sendNotification()`. Untuk setiap pesan yang masuk, aplikasi akan membuat notifikasi dengan judul dan teks yang sudah diatur untuk mengingatkan pengguna bahwa pengguna dalam kondisi yang perlu dilakukan tindak lanjut. Ketika notifikasi tersebut di klik, maka akan dibuka kelas *activity* baru yang berisi peta dari *Google Maps* Android API yang menunjukkan posisi pengguna dan beberapa lokasi fasilitas kesehatan terdekat. Kode sumber untuk pengiriman notifikasi dapat dilihat pada Gambar 4.33.

```
private void sendNotification(String message) {
Intent intent = new Intent(this,
MainActivity2Activity.class);
intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
PendingIntent pendingIntent =
PendingIntent.getActivity(this, 0 /* Request code
*/, intent, 0);
Log.i(TAG, "GCM Message: " + message);
Uri defaultSoundUri=
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_N
```

```
OTIFICATION);
NotificationCompat.Builder notificationBuilder = new
NotificationCompat.Builder(this).setSmallIcon(R.mipm
ap.ic_launcher).setContentTitle("Caution!")
            .setStyle(new
NotificationCompat.BigTextStyle().bigText(message))
            .setContentText("Tap for nearby health
facilities, if needed.")
            .setSound(defaultSoundUri)
            .setContentIntent(pendingIntent);
NotificationManager notificationManager =
(NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
notificationManager.notify(0 /* ID of notification
*/, notificationBuilder.build());}
```

**Gambar 4.33 Kode Sumber Pengiriman Notifikasi**

*Push notification* yang dikirimkan dapat diaktifkan dengan interaksi pengguna yaitu dengan melakukan klik, yang mana akan membuka *activity* pada aplikasi yang mengambil posisi pengguna saat ini dan mengambil data dari beberapa fasilitas kesehatan terdekat untuk kemudian dalam bentuk peta. Pembacaan posisi pengguna pada saat itu dilakukan oleh Android dengan menggunakan layanan lokasi yang diberikan oleh jaringan dari kartu. Pada inisialisasi awal *activity* ini akan mencari lokasi tetap dari pengguna yang diimplementasikan pada fungsi `onLocationChanged()` yang dieksekusi oleh "locationManager" ketika pengguna membuka *activity*. Pada fungsi ini akan ditepatakan titik koordinat *latitude* dan *longitude* dari posisi pengguna saat itu. Kode sumber dari pencarian posisi pengguna dapat dilihat pada Gambar 4.34.

```
@epublic void onLocationChanged(Location location)
{
if(alreadySet == false)
{
alreadySet = true;
currLat = location.getLatitude();
currLng = location.getLongitude();
currLocation = new Location("Current Location");
```

```
currLocation.setLatitude(location.getLatitude());
currLocation.setLongitude(location.getLongitude());
currLatLng = new
LatLng(currLocation.getLatitude(),currLocation.getLon
gitude());
}
}
```

**Gambar 4.34 Kode Sumber Untuk Menetapkan Posisi Pengguna**

Setelah posisi pengguna yang tepat didapatkan, maka informasi tersebut digunakan sebagai parameter dari API yang disediakan oleh *Google Maps* Android API untuk mendapatkan data beberapa fasilitas kesehatan terdekat. Fungsi ini dieksekusi ketika *activity* dijalankan pada fungsi `onCreate()`. Parameter yang dibutuhkan dalam pemanggilan API ini adalah posisi koordinat yang berupa *latitude* dan *longitude*, radius dari cakupan area pencarian fasilitas dari posisi pengguna, yang mana diimplementasikan untuk melakukan pencarian dalam radius 5 km, tipe fasilitas yang dicari yaitu 'hospital' dan kemudian API *key* dari layanan *Google Maps Android* API. Hasil dari pemanggilan API ini dalam bentuk JSON, yang kemudian akan diekstrak sesuai dengan masing – masing atribut yang dibutuhkan, yaitu nama, alama, serta koordinat *latitude* dan *longitude*. Daftar hasil dari ekstrasi ini kemudian akan disimpan dalam *hash map*. Kode sumber proses pemanggilan API *Google Maps* Android dan ekstrasi dapat dilihat pada Gambar 4.35.

```
String api_address =
"https://maps.googleapis.com/maps/api/place/nearbysea
rch/json?";
String api_lat =
Double.toString(currLocation.getLatitude());
String api_lng =
Double.toString(currLocation.getLongitude());
String param_location = "location=" + api_lat + "," +
api_lng;
String param_radius = "&radius=5000";
String param_types = "&types=hospital";
String param_key =
```

```
"&key=AIzaSyDuc4ab9iT7VPSZMCdKI416yQOeLprV5ME";
String apiURL = api_address + param_location +
param_radius + param_types + param_key;
ServiceHandler serviceHandler = new ServiceHandler();
String jsonResponse =
serviceHandler.makeServiceCall(apiURL,ServiceHandler.
GET);
Log.d("Response: ", "> " + jsonResponse);
if(jsonResponse!=null) {
try {
JSONObject jsonObj = new JSONObject(jsonResponse);
tagResults = jsonObj.getJSONArray(TAG_RESULTS);
for(int i=0; i<tagResults.length(); i++) {
JSONObject resultObj = tagResults.getJSONObject(i);
String id = resultObj.getString(TAG_ID);
String name = resultObj.getString(TAG_NAME);
String vicinity = resultObj.getString(TAG_VICINITY);
JSONObject geometryObj =
resultObj.getJSONObject(TAG_GEOMETRY);
JSONObject locationObj =
geometryObj.getJSONObject(TAG_LOCATION);
String lat = locationObj.getString(TAG_LAT);
String lng = locationObj.getString(TAG_LNG);
HashMap<String, String> hospital = new
HashMap<String, String>();
hospital.put(TAG_ID, id);
hospital.put(TAG_NAME, name);
hospital.put(TAG_VICINITY, vicinity);
hospital.put(TAG_LAT, lat);
hospital.put(TAG_LNG, lng);
hospitalList.add(hospital);
} } catch (JSONException e) { e.printStackTrace();
} } else {
Log.e("ServiceHandler", "Couldn't get any data from
the url");
}
```

**Gambar 4.35 Kode Sumber Pencarian Lokasi Fasiltas Kesehatan Terdekat**

Setelah informasi lokasi dari fasilitas kesehatan didapat, berdasarkan data lokasi yang mencakup titik koordinat *latitude*, *longitude* dan nama fasilitas, masing – masing lokasi akan

ditampilkan ke peta dengan *Google Maps* Android API. Proses menampilkan masing – masing lokasi ke peta dapat dilihat pada Gambar 4.36.

```
if (mMap == null) {
mMap = ((SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map
)).getMap();
if (mMap != null) {
for(int i=0; i<MAX_PLACES; i++)
{
mMap.addMarker(new
MarkerOptions().position(currLatLng)
               .title("You are here now.")
.icon(BitmapDescriptorFactory.defaultMarker(BitmapDes
criptorFactory.HUE_AZURE)));
mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(cur
rLatLng, 14));
mMap.addMarker(new MarkerOptions().position(new
LatLng(Float.parseFloat(hospitalList.get(i).get(TAG_L
AT)),
Float.parseFloat(hospitalList.get(i).get(TAG_LNG))))
.title(hospitalList.get(i).get(TAG_NAME))
.snippet(hospitalList.get(i).get(TAG_VICINITY))
);
} } }
```

**Gambar 4.36 Kode Sumber Proses Menampilkan Lokasi**

Pada kelas “MainActivity” yang merupakan bagian utama dari aplikasi Android ini, pada fungsi `onCreate()`, “MainActivity” akan menerima *broadcast* dari *service* “RegistrationIntentServic”e bahwa proses registrasi ke *Google Cloud Messaging connection server* untuk aplikasi ini sudah selesai. “MainActivity” lalu menampilkan *string* yang mengatakan registrasi berhasil dan token sudah terkirim ke server jika nilai dari *sharedPreference* untuk string `SENT_TOKEN_TO_SERVER` adalah *true*. Jika tidak, *MainActivity* akan menampilkan *string* token *error*. Kemudian dilakukan pengecekan terhadap *Google Play Services*. Kode sumber dari

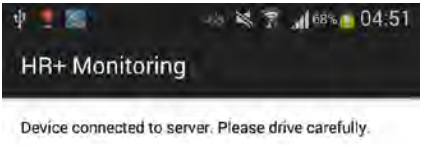
fungsi inisialiasi `onCreate()` pada kelas “MainActivity” ini dapat dilihat pada Gambar 4.37.

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
mRegistrationProgressBar       =       (ProgressBar)
findViewById(R.id.registrationProgressBar);
mRegistrationBroadcastReceiver =
new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent)
{
mRegistrationProgressBar.setVisibility(ProgressBar.GO
NE);
SharedPreferences sharedPreferences =
PreferenceManager.getDefaultSharedPreferences(context
);
boolean sentToken = sharedPreferences
.getBoolean(QuickstartPreferences.SENT_TOKEN_TO_SERVE
R, false);
if (sentToken) {
mInformationTextView.setTextColor(Color.BLACK);
mInformationTextView.setText(getString(R.string.bgtxt
));
} else {
mInformationTextView.setTextColor(Color.BLACK);
mInformationTextView.setText(getString(R.string.bgtxt
_error));
}}
};
mInformationTextView = (TextView)
findViewById(R.id.informationTextView);
if (checkPlayServices()) {
Intent intent =
new Intent(this, RegistrationIntentService.class);
startService(intent);
}}
```

**Gambar 4.37 Kode Sumber Fungsi *onCreate() MainActivity***

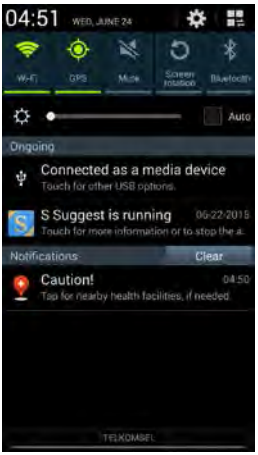
Hasil dari implementasi notifikasi berupa aplikasi Android dengan nama “HR+ Monitoring” yang sudah berjalan dan berhasil

memiliki token registrasi dengan *Google Cloud Messaging* server dapat dilihat pada Gambar 4.38.




**Gambar 4.38 Tampilan Halaman Utama Aplikasi Android**

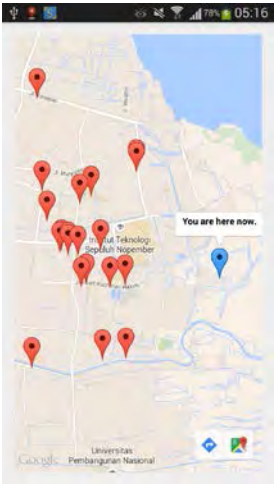
Tampilan dari fitur *push notification* yang diterima oleh *smartphone* Android pengguna dari *Google Cloud Messaging* server ketika selesai mengolah data dan memenuhi kondisi untuk pengiriman notifikasi dapat diliha pada Gambar 4.39.




**Gambar 4.39 Tampilan *Push Notification* Aplikasi Android**

Tampilan dari aplikasi Android yang memuat peta dengan informasi posisi pengguna saat itu dan beberapa lokasi fasilitas kesehatan terdekat, yang didapat dari hasil interaksi pengguna pada *push notification* dapat dilihat pada Gambar 4.40 dan

Gambar 4.41, dengan *marker* warna biru untuk posisi pengguna dan *marker* warna merah untuk fasilitas kesehatan terdekat.




**Gambar 4.40 Tampilan Peta Posisi Pengguna Aplikasi Android**


**Gambar 4.41 Tampilan Peta Lokasi Fasilitas Kesehatan Terdekat Aplikasi Android**

# BAB V
# PENGUJIAN DAN EVALUASI

Pada bab ini akan dibahas mengenai hasil pengujian dan analisa kinerja yang telah dilakukan. Uji coba akan dibagi menjadi tiga bagian yaitu uji coba fungsionalitas, uji coba skenario dan uji coba performa.

## 5.1. Lingkungan Pengujian

Dalam membangun aplikasi ini digunakan beberapa perangkat pendukung baik perangkat keras maupun perangkat lunak. Perangkat keras yang digunakan dalam pembuatan aplikasi ini adalah sebuah laptop *Apple Macbook* Pro Retina 13,3-inch yang memiliki spesifikasi sebagai berikut.

- Prosesor Intel(R) Core i5 CPU @ 2,40GHz
- Memori (RAM) 4,00 GB
- Intel Iris *Graphics*
- Mikrokontroler Arduino Uno R3
- GPS/GSM/GPRS *Shield* v3
- *Pulse* Sensor
- Modem Huawei E173

## 5.2. Pengujian Fungsionalitas

Pada pengujian fungsionalitas akan diuji fungsi yang telah dibuat dalam sistem. Fungsi yang diuji dalam sistem dimulai dari fungsi pengambilan data detak jantung, posisi koordinat dan kecepatan pengguna, fungsi pengiriman data ke dalam basis data pada server melalui mikrokontroler Arduino yang sudah dirangkai. Pada server, data akan diolah secara berperiodik untuk digunakan sebagai data tes pada proses klasifikasi. Selanjutnya, apabila memenuhi kondisi akan dilakukan fungsi pengiriman

notifikasi yang meliputi dua jenis notifikasi, yaitu notifikasi berupa pesan teks kepada nomor kerabat yang didaftarkan dan notifikasi kepada pengguna pada *smartphone* Android.

### 5.2.1. Pengujian Fungsionalitas Pembacaan Data

Fungsionalitas utama dari perangkat keras mikrokontroler dalam sistem ini adalah melakukan pembacaan data. Uji coba fungsionalitas ini akan mengikuti prosedur yang ada pada Tabel 5.1 berikut.

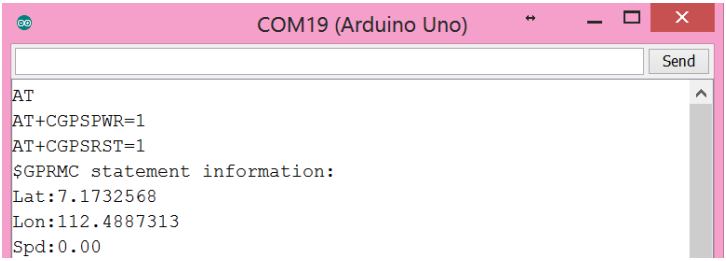
**Tabel 5.1 Prosedur Uji Coba Fungsionalitas Pengiriman Data**

| | |
|---|---|
| Nama Skenario Pengujian | Uji Coba Fungsionalitas Pengambilan Data |
| Kode | UJ-01 |
| Tujuan Pengujian | Menguji fitur pengambilan data |
| Kondisi Awal | Mikrokontroler dijalankan |
| Data Input | - |
| Prosedur Pengujian | 1. Pengguna memasang *Pulse* Sensor ke daun telinga dengan *ear clip* yang sudah disediakan<br>2. Pengguna melakukan aktivitas berkendara di mobil dengan membawa mikrokontroler untuk diletakkan di sebelah pengguna<br>3. Pengambilan data berupa data detak jantung, posisi koordinat dan kecepatan pengguna |
| Hasil yang Diharapkan | Detak jantung, kecepatan dan posisi koordinat pengguna. |
| Hasil yang Diperoleh | Berhasil |

Pengujian ini ditujukan untuk menguji fungsionalitas utama yaitu melakukan pengambilan data dari mikrokontroler Arduino terhadap pengguna. Proses pemasangan *Pulse* Sensor ditunjukkan pada ilustrasi yang ditampilkan pada Gambar 5.1.

**Gambar 5.1 Ilustrasi Pemakaian *Pulse* Sensor**

Tampilan serial monitor untuk memastikan pengambilan data dari mikrokontroler Arduino berhasil dilakukan dapat dilihat pada Gambar 5.2 berikut.




**Gambar 5.2 Tampilan Pengujian Serial Monitor Ketika Akuisisi Data**

Pada mikrokontroler Arduino terdapat indikator yang berupa lampu LED yang menandakan adanya proses pengambilan

data koordinat pengguna dengan LED warna merah dan detak jantung dengan LED warna biru. Ilustrasi indikasi yang ditunjukkan dengan lampu LED ini dapat dilihat pada Gambar 5.3 untuk indikasi pembacaan data GPS.

**Gambar 5.3 Ilustrasi Indikasi Pengambilan Data Sukses dengan LED**

### 5.2.2. Pengujian Fungsionalitas Pengiriman Data

Setelah data berhasil didapatkan melalui uji coba pembacaaan data, fungsionalitas sistem yang akan di uji adalah saat pengiriman data dari mikrokontroler Arduino ke server. Prosedur pengujian uji coba fungsionalitas ini dijelaskan pada Tabel 5.2 berikut.

**Tabel 5.2 Prosedur Uji Coba Fungsionalitas Pengiriman Data**

| | |
|---|---|
| Nama Skenario Pengujian | Uji Coba Fungsionalitas Pengiriman Data |
| Kode | UJ-02 |
| Tujuan Pengujian | Menguji fitur untuk mengirimkan data ke server |
| Kondisi Awal | Posisi koordinat, kecepatan dan |

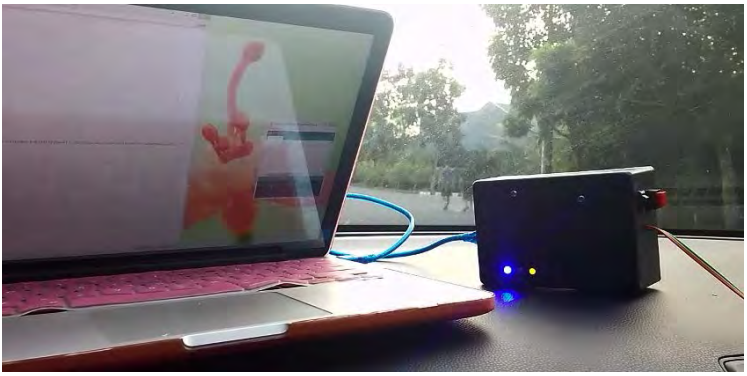
| | |
|---|---|
| | data detak jantung sudah terbaca |
| Data Input | Posisi koordinat, kecepatam dan data detak jantung |
| Prosedur Pengujian | 1. Mengirimkan data ke server dengan AT *Command*<br>2. Mengecek pada basis data apakah data yang dikirimkan sudah berhasil masuk |
| Hasil yang Diharapkan | Data berhasil masuk |
| Hasil yang Diperoleh | Data tersimpan dalam basis data |

Pengujian ini dilakukan dengan tujuan untuk menoba fungsionalitas pengiriman data dari mikrokontroler ke server dengan metode pengiriman HTTP *Request* melalui jarigan GPRS. Pengujian ini dilakukan dengan mencoba pengaturan APN (*Access Point Name*) yang meliput tipe koneksi, nama APN, nama pengguna, nama kode rahasia. Setelah pengaturan selesai maka dilakukan pemanggilan HTTP *Request* dan pemanggilan variabel yang menampung data untuk dikirimkan ke server. Tampilan serial monitor untuk memastikan mikrokontroler berhasil melakukan pengaturan koneksi dan melakukan pengiriman dapat dilihat pada Gambar 5.4.

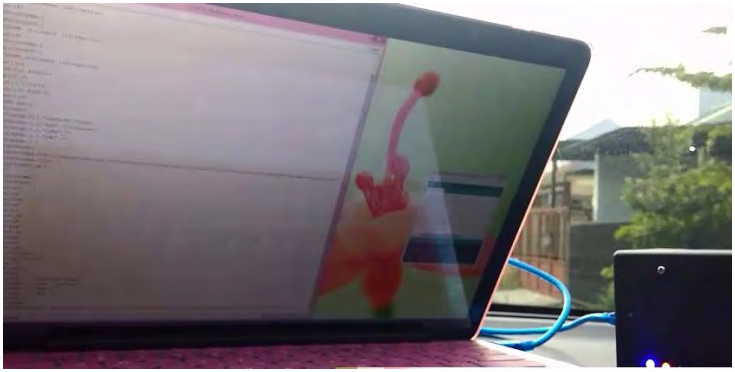

**Gambar 5.4 Tampilan Pengujian Serial Monitor Ketika Pengiriman Data**

Pada mikrokontroler Arduino terdapat indikator yang berupa lampu LED yang menandakan adanya proses pengaturan jaringan GPRS dengan LED warna hijau dan LED warna kuning yang mengindikasikan proses pengiriman data tengah dilakukan. Ilustrasi indikasi yang ditunjukkan dengan lampu LED ini dapat

dilihat pada Gambar 5.5 untuk indikasi pengaturan jaringan GPRS dan Gambar 5.6 untuk indikasi pengiriman data ke basis data server.



**Gambar 5.5 Ilustrasi Indikasi Pengaturan Jaringan GPRS Sukses dengan LED**



**Gambar 5.6 Ilustrasi Indikasi Pnegiriman Data Sukses dengan LED**

Data yang berhasil dikirim akan disimpan ke dalam basis data dengan atribut yang diperlukan adalah “timestamp” untuk waktu dari data dikirim, “latitude” untuk titik koordinat *latitude*, “longitude” untuk titik koordinat *longitude*, “velocity” untuk kecepatan pengguna, “acceleration” merupakan percepatan

pengguna dengan perbandingan waktu dari data sebelumnya, "heartrate" untuk menyimpan data detak jantung dan status sebagai *flag* yang menandakan apakah data ini sudah diproses oleh server atau belum dan "condition" yang berupa informasi kondisi hasil pengolahan data oleh server. Tampilan data yang berhasil masuk ke dalam basis data ditampilkan pada Gambar 5.7.

| id | timestamp | latitude | longitude | velocity | acceleration | heartrate | status | condition |
|---|---|---|---|---|---|---|---|---|
| 1 | 2015-06-15 13:30:49 | -7.288514 | 112.814390 | 20 | 0 | 60 | 0 | NULL |
| 2 | 2015-06-15 13:32:49 | -7.290137 | 112.812797 | 20 | 0 | 85 | 0 | NULL |

**Gambar 5.7 Tampilan Pengujian Basis Data untuk Data yang Berhasil Disimpan**

### 5.2.3. Pengujian Fungsionalitas Pengambilan Keputusan dengan *Fuzzy Logic*

Setelah data berhasil masuk ke basis data di server maka fungsionalitas selanjutnya yang akan diuji adalah fungsionalitas dalam melakukan pengambilan keputusan dengan *Fuzzy Logic* yang dilakukan dengan prosedur yang dijelaskan pada Tabel 5.3.

**Tabel 5.3 Prosedur Uji Coba Fungsionalitas Pengambilan Keputusan dengan *Fuzzy Logic***

| | |
|---|---|
| Nama Skenario Pengujian | Uji Coba Fungsionalitas Pengambilan Keputusan dengan *Fuzzy* |
| Kode | UJ-03 |
| Tujuan Pengujian | Menguji fitur untuk menjalankan proses pengambilan keputusan dengan *Fuzzy* |
| Kondisi Awal | Server mengambil data terakhir yang belum diproses basis data untuk kemudian diproses |
| Data Input | Posisi koordinat, kecepatam dan data detak jantung dari data terakhir yang ada pada basis data |

| | |
|---|---|
| Prosedur Pengujian | 1. Server mengambil data terakhir dari basis data yang belum diproses<br>2. Server menjalankan *library* jFuzzyLogic untuk mengolah data |
| Hasil yang Diharapkan | Data diklasifikasikan dengan benar |
| Hasil yang Diperoleh | Data berhasil diklasifikasikan dengan tepat |

Proses pengujian fungsionalitas pengambilan keputusan dengan *Fuzzy* berlangsung di server yang berjalan secara periodik yaitu setiap 3 menit untuk mendapatkan data dari basis data dan diolah dengan menggunakan *library* jFuzzyLogic yang sudah disesuaikan dengan parameter masukkan dan keluarannya. Parameter masukkan yang diperlukan ada tiga yaitu detak jantung, kecepatan dan percepatan. Tampilan *console* dari server yang menunjukkan bahwa proses pengujian berhsail dapat dilihat pada Gambar 5.8 berikut.


**Gambar 5.8 Hasil Ujicoba Pengambilan Keputusan dengan *Fuzzy* di Server**

Berdasarkan data diatas dapat dilihat bahwa data sudah berhasil diklasifikasikan dengan benar dengan menggunakan algoritma pengambilan keputusan *Fuzzy*. *Sample* data yang diproses dapat dilihat pada Gambar 5.9 berikut.

| id | timestamp | latitude | longitude | velocity | acceleration | heartrate | status | condition |
|---|---|---|---|---|---|---|---|---|
| 95 | 2015-06-22 12:57:43 | -7.1733284 | 112.4887465 | 80 | 3 | 135 | 0 | NULL |

**Gambar 5.9 Sample Data Untuk Pengujian Fuzzy**

### 5.2.4. Pengujian Fungsionalitas Pengiriman Notifikasi SMS

Setelah data sudah melalui proses pengambilan keputusan di server, maka akan dilakukan pengujian fungsionalitas dalam mengirimkan notifikasi berupa pesan teks atau SMS. Prosedur pengujian pengiriman notifikasi SMS ini dijelaskan pada Tabel 5.4 berikut.

**Tabel 5.4 Prosedur Uji Coba Pengiriman Notifikasi SMS**

| | |
|---|---|
| Nama Skenario Pengujian | Uji Coba Fungsionalitas Pengiriman Notifikasi SMS |
| Kode | UJ-04 |
| Tujuan Pengujian | Menguji fitur untuk mengirim notifikasi SMS |
| Kondisi Awal | Server mendapatkan nilai yang memenuhi persyaratan untuk notifikasi |
| Data Input | Posisi koordinat dan detak jantung pengguna |
| Prosedur Pengujian | 1. Server mendapatkan kondisi pengiriman berdasarkan hasil pengambilan keputusan<br>2. Server menjalankan *service* Gammu untuk dapat mengirimkan pesan teks |
| Hasil yang Diharapkan | Data dikirim dengan tepat |
| Hasil yang Diperoleh | Data berhasil dikirim berupa pesan teks dengan tepat |

Dalam pengujian ini data yang digunakan oleh server dalah data pada Gambar 5.9 yang mana berdasarkan hasil pengujian fungsionalitas sebelumnya yaitu pengujian fungsionalitas pengambilan keputusan mendapati hasil akhir kondisi yang termasuk dalam kondisi yang memerlukan notifikasi. Konten dari pesan teks yang dikirimkan berdasarkan lokasi koordinat dari data yang diolah dan kemudian diproses dengan *geocoding* untuk mendapatkan estimasi alamat dari *Google Maps* API. Hasil dari uji coba ini berupa notifikasi pesan teks yang didapati oleh nomor kerabat yang sudah terdaftar di server. Tampilan dari notifikasi SMS yang diterima dengan format alamat dari posisi pengguna ini dapat dilihat pada Gambar 5.10 berikut.


**Gambar 5.10 Tampilan Pengujian Notifikasi SMS dari Server**

### 5.2.5. Pengujian Fungsionalitas Pengiriman Notifikasi Aplikasi Android

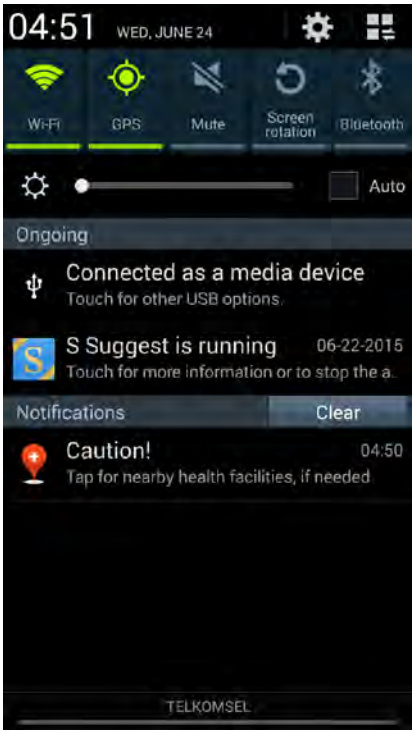
Dalam sistem ini bentuk notifikasi yang dikirimkan juga meliputi pengiriman *push notification* dengan bantuan *service* dari *Google Cloud Messaging* dari server ke aplikasi Android pada *smartphone* pengguna. Prosedur pengiriman notifikasi berupa *push notification* ke aplikasi Android ini dapat dilihat pada Tabel 5.5 berikut.

**Tabel 5.5 Prosedur Uji Coba Pengiriman Notifikasi Aplikasi Android**

| | |
|---|---|
| Nama Skenario Pengujian | Uji Coba Fungsionalitas Pengiriman Notifikasi Android |
| Kode | UJ-05 |
| Tujuan Pengujian | Menguji fitur untuk mengirim notifikasi Android |
| Kondisi Awal | Server mendapatkan nilai yang memenuhi persyaratan untuk notifikasi |
| Data Input | Posisi koordinat pengguna dari Jantung |
| Prosedur Pengujian | 1. Server mendapatkan kondisi pengiriman berdasarkan hasil pengambilan keputusan<br>2. Server menjalankan *service* fitur GCM untuk *downstream* message ke aplikasi Android terdaftar |
| Hasil yang Diharapkan | Data dikirim dengan tepat dan dapat dibuka dengan benar berisi informasi rumah sakit atau layanan kesehatan terdekat. |
| Hasil yang Diperoleh | Aplikasi Android menerima *push notification* dan mampu menampilkan peta |

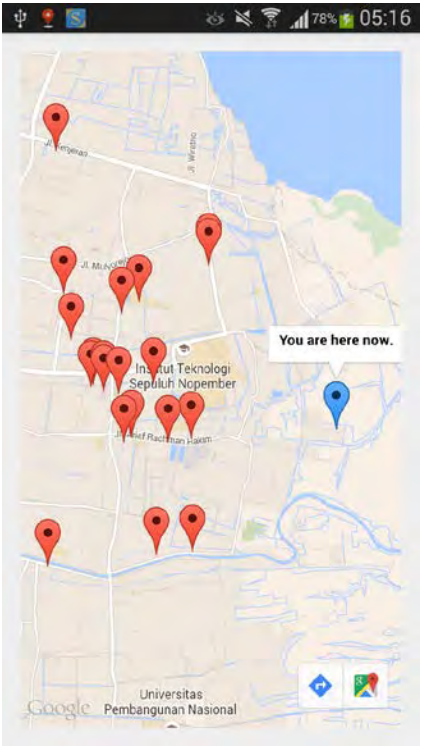
Pengujian ini dilakukan saat server melakukan pengolahan data, dimana data yang digunakan ada data pada Gambar 5.9 yang menunjukkan bahwa pengguna termasuk dalam kategori yang membutuhkan tindak lanjut yang berupa notifikasi ini. Pengujian ini juga ditujukan untuk menguji fungsionalitas aplikasi Android sebagai *client* dari *Google Cloud Messaging* server yang berhasil mengirimkan token registrasi yang berhasil

disimpan di server. Token registrasi ini penting untuk melakukan pengiriman *push notification* dari *Google Cloud Messaging web* API. Berhasil atau tidaknya pengujian ini didasari dari dua buah hasil, yaitu aplikasi Android berhasil menerima *push notification* dari server dan kemudian berhasil menampilkan peta dari posisi pengguna saat itu dan lokasi dari beberapa fasilitas kesehatan terdekat. Tampilan pada *smartphone* Android ketika berhasil menerima *push notification* dari server dapat dilihat pada Gambar 5.11




**Gambar 5.11 Tampilan Pengujian *Push Notification* pada Aplikasi Android**

Sedangkan tampilan aplikasi Android yang berhasil menampilkan posisi pengguna dan beberapa fasilitas kesehatan ketika pengguna melakukan interaksi untuk membuka *push notification* dapat dilihat pada Gambar 5.12 berikut.

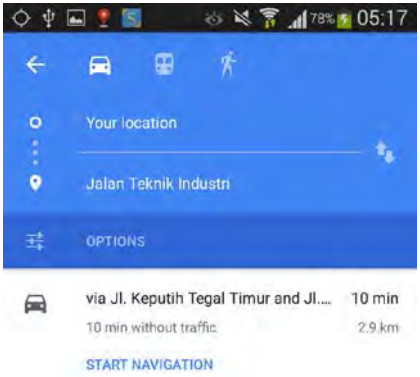



**Gambar 5.12 Tampilan Pengujian Peta pada Aplikasi Android**

Dari pengujian aplikasi Android diatas yang sudah berhasil menampilkan peta posisi pengguna dengan *marker* warna biru dan beberapa fasilitas kesehatan dengan *marker* warna merah, dapat pula dilakukan interaksi untuk mengetahui informasi dari fasilitas yang dapat dilihat pada Gambar 5.9.


**Gambar 5.13 Tampilan Pengujian Peta dengan Lokasi Fasilitas Kesehatan Terdekat pada Aplikasi Android**

Dengan mengetahui lokasi fasilitas kesehatan terdekat, pengguna dapat memanfaatkan aplikasi *Google Maps* yang sudah ada di *smartphone* Android, untuk melakukan navigasi ke fasilitas kesehatan yang diipilih dengan melakukan interaksi berupa klik pada *icon* warna biru kecil di bawah layar. Tampilan dari aplikasi *Google Maps* dengan tujuan dari fasilitas kesehatan yang dipilih dapat dilihat pada Gambar 5.10.




**Gambar 5.14 Tampilan Pengujian Navigasi ke Lokasi Fasilita Kesehatan Terdekat dengan Google Maps**

## 5.3. Pengujian Skenario

Pengujian kali ini dilakukan pada sebuah skenario dimana sistem akan diuji dengan skenario yang sudah ditentukan. Uji coba ini bertujuan untuk mengetahui apakah sistem dapat melakukan pembacaan data riil kepada subjek dengan penyakit jantung dengan tiga skenario pengambilan data yang berbeda. Pengujian dilakukan dengan prosedur yang dapat dilihat pada Tabel 5.6.

**Tabel 5.6 Status Pengguna Uji Coba**

| | |
|---|---|
| Subjek | Pengguna |
| Umur | 48 tahun |
| Jenis Kelamin | Pria |
| Penyakit Jantung | Jantung koroner |
| Alat Bantu | - |

Pengujian dilakukan dengan menguji pengguna dalam mengendarai mobil dan data dikirim setiap 2 menit ke server. Skenario pengujian ini dilakukan berdasarkan prosedur yang ada pada Tabel 5.7.

**Tabel 5.7 Prosedur Pengujian Skenario**

| | |
|---|---|
| Nama Skenario Pengujian | Uji Coba Skenario pada Pengguna |
| Kode | UJ-06 |
| Tujuan Pengujian | Untuk melihat bagaimana pembacaan data pada subjek riil yaitu penderita penyakit jantung |
| Kondisi Awal | Pengguna memasang perangkat mikrokontroler dan mulai berkendara |
| Skenario 1 | Pengguna berkendara dengan kecepatan normal atau sedang |
| Masukkan | Detak jantung, kecepatan dan posisi koordinat |
| Keluaran | Data detak jantung, koordinat pengguna dan kecepatan terbaca dan terkirim |
| Hasil Uji Coba | Berhasil |
| Skenario 2 | Pengguna menaikkan kecepatan kendaraan |
| Masukkan | Detak jantung, kecepatan dan posisi koordinat |
| Keluaran | Data detak jantung, koordinat |

| | |
|---|---|
| | pengguna dan kecepatan terbaca dan terkirim |
| Hasil Uji Coba | Berhasil |
| Skenario 3 | Pengguna menurunkan kecepaatan kendaraan |
| Masukkan | Detak jantung, kecepatan dan posisi koordinat |
| Keluaran | Data detak jantung, koordinat pengguna dan kecepatan terbaca dan terkirim |
| Hasil Uji Coba | Berhasil |

Berdasarkan dari prosedur pada Tabel 5.7, maka hasil dari pengujian ini dapat dilihat pada Tabel 5.8

**Tabel 5.8 Hasil Pengujian Skenario**

| Latitude | Longitude | Kecepatan (km/jam) | Percepatan ($m/s^2$) | *Heartrate* (detak per menit) |
|---|---|---|---|---|
| -7.254316 | 112.799929 | 46.3 | 0.00 | 88 |
| -7.256263 | 112.800704 | 38.43 | -0.04 | 79 |
| -7.257618 | 112.804207 | 25.05 | -0.05 | 81 |
| -7.258970 | 112.805800 | 71.76 | 0.23 | 105 |
| -7.262205 | 112.806991 | 61.58 | -0.06 | 101 |
| -7.264769 | 112.807615 | 62.5 | 0.01 | 83 |
| -7.267134 | 112.808372 | 43.15 | -0.09 | 89 |
| -7.270543 | 112.809823 | 24.3 | -0.09 | 76 |
| -7.272225 | 112.811459 | 15.28 | -0.03 | 80 |
| -7.275801 | 112.811428 | 21.76 | 0.03 | 87 |

Dari hasil pengujian pada Tabel 5.8 dapat dilihat bahwa sistem dapat melakukan pembacaan pada target riil subjek yaitu

pasien penderita penyakit jantung dengan menggunakan skenario kecepatan yang berbeda dimana nilainya tetap dapat terbaca oleh sistem dan terkirim ke server dan disimpan di basis data di server.

## 5.4. Pengujian Performa

Pengujian performa dilakukan dengan tujuan mengetahui kemampuan sistem. Pengukuran performa dilakukan dengan menguji akurasi *Pulse* sensor, *successful rate* pada proses pengiriman data ke suatu server. Selain itu, pengujian juga dilakukan dalam mengukur performa akurasi metode *Fuzzy Logic* dalam mengambil keputusan.

### 5.4.1. Pengujian Performa Akurasi *Pulse* Sensor Sebagai Alat Sensor Detak Jantung

Pengujian performa akurasi *Pulse* sensor sebagai alat sensor detak jantung dilakukan dengan tujuan mengetahui tingkat akurasi sensor jika dibandingkan dengan pembacaan detak jantung dari alat tensimeter ABN DW-200 yang juga dilengkapi dengan fitur pembacaan detak jantung dengan tingkat akurasi yang tertulis pada spesifikasi alat sebesar ± 5 detak jantung. Perbandingan *Pulse* sensor dengan tensimeter ABN DW-200 ini dilakukan untuk mengetahui performa *Pulse* sensor sebagai alternatif. Prosedur pengujian performa akurasi *Pulse* sensor sebagai alat sensor detak jantung ini dapat dilihat pada Tabel 5.9.

**Tabel 5.9 Prosedur Pengujian Performa Akurasi *Pulse* Sensor Sebagai Alat Sensor Detak Jantung**

| | |
|---|---|
| Nama | Pengujian Performa *Pulse* Sensor Sebagai Alat Sensor Detak Jantung |
| Kode | UJ-08 |
| Tujuan Pengujian | Menguji akurasi *Pulse* sensor sebagai alat sensor detak jantung untuk mengetahui performanya sebagai |

| | |
|---|---|
| | alternatif alat sensor detak jantung |
| Kondisi Awal | Disiapkan alat tensimeter ABN DW-200 HEM 6-111 sebagai perbandingan alat medis |
| Prosedur Pengujian | Pengujian dilakukan dengan membaca masing – masing hasil pembacaan detak jantung alat, yaitu *Pulse* sensor dan alat tensimeter ABN DW-200 secara bergantian dilakukan sebanyak 10 kali |
| Masukkan | 10 buah data masukkan hasil pembacaan dari *Pulse* sensor dan 10 buah data masukkan hasil pembacaan alat tensimeter ABN DW-200 |
| Keluaran | Rata – rata selisih hasil pembacaan data dari kedua alat *Pulse* sensor dan alat tensimeter ABN DW-200 |
| Hasil Uji Coba | Hasil pembacaan *Pulse* sensor tidak jauh berbeda dengan alat tensimeter ABN DW-200 yang merupakan alat dengan standar medis, rata – rata selisih yang didapati dari 10 kali percobaan hanya sebesar 2.6 |

Hasil pembacaan berulang sebanyak 10 kali oleh *Pulse* sensor dan alat tensimeter ABN DW-200 dapat dilihat pada Tabel 5.10 berikut.

**Tabel 5.10 Perbandingan Hasil Pembacaan Data Detak Jantung *Pulse* Sensor dan Tensimeter ANM DW-200**

| Percobaan Ke | *Pulse* Sensor (detak per menit) | ABN DW-200 (detak per menit) | Selisih |
|---|---|---|---|
| 1 | 78 | 77 | 1 |
| 2 | 81 | 76 | 5 |
| 3 | 74 | 77 | 3 |

| | | | |
|---|---|---|---|
| 4 | 71 | 75 | 4 |
| 5 | 66 | 68 | 2 |
| 6 | 77 | 78 | 1 |
| 7 | 71 | 67 | 4 |
| 8 | 63 | 65 | 2 |
| 9 | 84 | 81 | 3 |
| 10 | 82 | 81 | 1 |
| Rata - rata | 74.7 | 74.5 | 2.6 |

Berdasarkan data dari Tabel 5.10, didapati bahwa dari 10 kali percobaan, hasil pengambilan data dari *Pulse* sensor rata – rata pada jumlah detak jantung 74.7 detak per menit, sedangkan pada alat tensimeter ABN DW-200 berada pada rata – rata 74.5 detak per menit dimana rata – rata dari selisih kedua data tiap percobaan adalah berbeda sebesar 2.6. Dari data ini dapat dianalisa bahwa *Pulse* sensor memiliki pembacaan data yang cukup akurat dengan rata – rata selisih pembacaan data tiap percobaan sebesar 2.6 dengan pembanding alat tensimeter ABN DW-200 yang memiliki spesifikasi akurasi ±5 detak jantung. Dengan demikian *Pulse* sensor dapat digunakan sebagai alat pembacaan detak jantung yang cukup akurat pada sistem ini.

### 5.4.2. Pengujian Performa Akurasi GPS dari GPS/GPRS/GSM *Shield* v3.0 Sebagai Alat Pembaca Kecepatan

Pengujian performa akurasi modul GPS dari GPS/GPRS/GSM *Shield* v3 sebagai alat pembaca kecepatan pengguna yang sedang dilakukan dengan tujuan mengetahui tingkat akurasi GPS jika dibandingkan dengan pembacaan dari *speedometer* kendaraan. Kendaraan yang digunakan merupakan mobil dengan batasan *speedometer* 0 hingga 200 km/jam. Perbandingan modul GPS dengan pembacaan manual dari speedometer ini dilakukan untuk mengetahui apakah hasil

pembacaan kecepatan oleh GPS memilki perbedaan yang cukup besar atau tidak jika dibandingkan dengan pembacaan manual dari *speedometer*. Prosedur pengujian performa akurasi GPS sebagai pembaca kecepatan ini dapat dilihat pada Tabel 5.11.

**Tabel 5.11 Prosedur Pengujian Performa Akurasi GPS Sebagai Alat Pembaca Kecepatan**

| | |
|---|---|
| Nama | Pengujian Performa Akurasi GPS Sebagai Alat Pembaca Kecepatan |
| Kode | UJ-09 |
| Tujuan Pengujian | Menguji akurasi modul GPS dari GPS/GPRS/GSM *Shield* v3.0 sebagai alat pembaca kecepatan |
| Kondisi Awal | Dikondisikan untuk pembacaan manual *speedometer* kendaraan |
| Prosedur Pengujian | Pengujian dilakukan dengan membaca masing – masing hasil pembacaan kecepatan, yaitu dari GPS dan *speedometer* dilakukan sebanyak 10 kali |
| Masukkan | 10 buah data masukkan hasil pembacaan dari GPS dan 10 buah data masukkan hasil pembacaan *speedometer* |
| Keluaran | Rata – rata selisih hasil pembacaan data dari kedua alat, yaitu GPS dan *speedometer* |
| Hasil Uji Coba | Hasil pembacaan GPS tidak jauh berbeda dengan *speedometer*, rata – rata selisih yang didapati dari 10 kali percobaan hanya sebesar 1.65 |

Hasil pembacaan kecepatan yang dilakukan berulang sebanyak 10 kali oleh GPS dan *speedometer* dapat dilihat pada Tabel 5.12 berikut.

**Tabel 5.12 Perbandingan Hasil Pembacaan Kecepatan oleh GPS dan *Speedometer* Kendaraan**

| Percobaan Ke | GPS (km/jam) | *Speedometer* (km/jam) | Selisih |
|---|---|---|---|
| 1 | 46.3 | 45 | 1.3 |
| 2 | 38.43 | 35 | 3.43 |
| 3 | 25.05 | 25 | 0.05 |
| 4 | 71.76 | 70 | 1.76 |
| 5 | 61.58 | 60 | 1.58 |
| 6 | 62.5 | 60 | 2.5 |
| 7 | 43.15 | 40 | 3.15 |
| 8 | 24.3 | 25 | 0.7 |
| 9 | 15.28 | 15 | 0.28 |
| 10 | 21.76 | 20 | 1.76 |
| Rata - rata | | | 1.65 |

Berdasarkan data dari Tabel 5.12, didapati bahwa dari 10 kali percobaan dimana data dari GPS diperoleh setiap GPS sudah mendapatkan posisi koordinat yang fix dan pembacaan manual *speedometer*, rata – rata selisih perbedaan yang terjadi tidak terlalu besar dimana dari masing – masing selisih tiap pembacaan rata – rata selisih perbedaan antar kedua data hanya sebesar 1.65.

### 5.4.3. Pengujian Performa Rasio Keberhasilan Pengiriman Data

Pengujian performa menghitung *successful rate* pada proses pengiriman data dilakukan dengan tujuan mengetahui tingkat keberhasilan pengiriman data ke basis data yang berada di server. Saat pengambilam sampling data dilakukan secara berulamg, maka pengiriman data ke server juga dilakukan secara berulamg. Untuk itu yang menjadi permasalahan adalah apakah proses pengiriman sebuah sampling data telah berhasil, sebelum

memulai proses pengambilan dan pengiriman sampling data yang berikutnya. Untuk itu diperlukan adanya pengujian untuk menganalisa nilai rasio keberhasilan (*successful rate*) pengiriman data yang masuk ke dalam basis data di server. Untuk prosedur pengujian yang akan dilakukan dapat dilihat pada Tabel 5.13.

**Tabel 5.13 Prosedur Pengujian Performa *Successful Rate* Pengiriman Data**

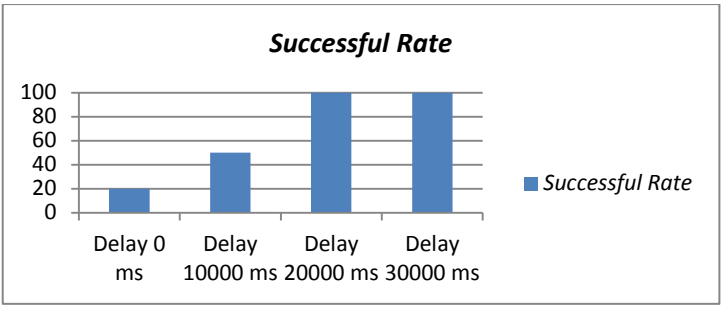
| | |
|---|---|
| Nama | Pengujian Performa Rasio Keberhasilan Pengriman Data ke Basis Data |
| Kode | UJ-10 |
| Tujuan Pengujian | Menguji rasio keberhasilan masuknya data yang dikirim mikrokontroler Arduino ke basis data di server |
| Kondisi Awal | - |
| Prosedur Pengujian | Pengujian dilakukan dengan 4 macam skenario, yaitu dengan mengatur *delay* yang berbeda sebelum proses pengiriman data dilakukan, yaitu sebagai berikut:<br>1. *Delay* = 0 ms<br>2. *Delay* = 10000 ms<br>3. *Delay* = 20000 ms<br>4. *Delay* = 30000 ms |
| Masukkan | 1 buah paket data yang berisi 4 parameter, yaitu titik koordinat *latitude*, titik koordinat *longitude*, detak jantung dan kecepatan pengguna dilakukan berulang sebanyak 8 kali |
| Keluaran | Dari salah satu skenario uji coba yang dilakukan menghasilkan presentasi nilai rasio keberhasilan yang tinggi |
| Hasil Uji Coba | Semakin besar nilai counter bilangan sebagai *delay* maka persentasi nilai rasio keberhasilan pengriman data ke basis data server semakin tinggi |

Dari prosedur pengujian tersebut didapatkan hasil pengujian performa *successful rate* pengiriman data yang dapat dilihat pada Tabel 5.14Tabel 5.14 berikut.

**Tabel 5.14 Hasil Pengujian Performa *Succesful Rate* Pengiriman Data**

| Waktu Delay | Data Terkirim | *Succesful Rate* ke Basis Data Server |
|---|---|---|
| 0 ms | 2 dari 8 | 25% |
| 10000 ms | 4 dari 8 | 50% |
| 20000 ms | 8 dari 8 | 100% |
| 30000 ms | 8 dari 8 | 100% |

Dari pengujian performa yang telah dilakukan, maka dapat dianalisa bahwa semakin besar *delay* yang diberikan setelah pengambilan data dan sebelum pengiriman data berikutnya, maka semakin besar juga rasio keberhasilan pengiriman data ke basis data ke server, hal ini dikarenakan pembacaan serial pada Arduino dalam pengiriman AT *Command* membutuhkan waktu untuk berjalan dan penggunaan dua modul, yaitu modul GPS dan modul GPRS/GSM sebagai modul komunikasi pada satu *shield* membuat dibutuhkan mekanisme pemanggilan AT *Command* secara bergantian yang mengakibatkan penggunaan delay yang tepat dalam hal ini cukup lama yaitu sebesar minimal 20000 ms atau 20 detik agar setiap data dari total 20 data dapat terkirim ke server. Gambar 5.15 berikut mempemperlihatkan hasil pengujian dalam bentuk grafik.




**Gambar 5.15 Grafik *Succesful Rate* Pengiriman Data**

### 5.4.4. Pengujian Performa Responsifitas Pengiriman Notifikasi

Pengujian performa responsifitas pengiriman notifikasi ini dilakukan dengan tujuan untuk mengetahui berapa lama waktu (*delay*) yang dibutuhkan oleh server untuk melakukan pengiriman notifikasi sampai notifikasi diterima. Notifikasi yang dikirim berupa notifikasi dalam bentuk pesan teks atau SMS dan *push notification* ke aplikasi pada *smartphone* Android pengguna. Pengujian ini dilakukan agar diketahui berapa lama rata – rata waktu yang dibutuhkan dari saat data dikirim dan diterima oleh server lalu kemudian dikirim notifikasinya apabila diperlukan untuk mengetahui tingkat responsifitas dari sistem. Prosedur pengujian performa responsifitas pengiriman notifikasi dapat dilihat pada Tabel 5.15.

**Tabel 5.15 Prosedur Pengujian Performa Responsifitas Penerimaan Notifikasi**

| | |
|---|---|
| Nama | Pengujian Peforma Responsifitas Pengiriman Notifikasi |
| Kode | UJ-11 |
| Tujuan Pengujian | Menguji responsifitas dari pengiriman notifikasi |
| Kondisi Awal | Data sudah didapat dan dikirim ke server |

| | |
|---|---|
| Prosedur Pengujian | Pengujian dilakukan dengan melakukan perhitungan waktu dari ketika data dikirim hingga data diterima dalam bentuk notifikasi yang dilakukan secara berulang sebanyak 10 kali |
| Masukkan | Waktu ketika data dikirim dan ketika notifikasi yang pertama kali diterima |
| Keluaran | Rata – rata waktu yang dibutuhkan untuk notifikasi diterima |
| Hasil Uji Coba | Rata – rata waktu yang diperlukan untuk pengiriman nofitikasi dari ketika data dikirim hingga notifikasi pertama kali diterima sebesar 3.041 menit atau sekitar 3 menit 2 detik |

Dari prosedur pengujian tersebut didapatkan hasil pengujian performa responsifitas pengiriman data yang dapat dilihat pada Tabel 5.16 berikut.

**Tabel 5.16 Hasil Pengujian Responsifitas Pengiriman Notifikasi**

| Percobaan Ke | Timestamp | Status Kondisi | Timestamp Notifikasi | Selisih (menit) |
|---|---|---|---|---|
| 1 | 5:20:52 | Dangerous | 5:24:10 | 3.3 |
| 2 | 5:24:26 | Potential | 5:28:04 | 3.9 |
| 3 | 5:28:40 | Potential | 5:32:03 | 3.8 |
| 4 | 5:31:59 | Dangerous | 5:35:11 | 3.2 |
| 5 | 5:34:46 | Dangerous | 5:38:00 | 3.23 |
| 6 | 5:37:23 | Dangerous | 5:40:44 | 3.35 |
| 7 | 5:41:04 | Dangerous | 5:44:19 | 3.25 |
| 8 | 5:44:29 | Normal | 0:00:00 | 0 |
| 9 | 5:48:53 | Potential | 5:52:01 | 3.13 |
| 10 | 5:52:12 | Potential | 5:55:27 | 3.25 |
| Rata - rata | | | | 3.041 |

Berdasarkan data hasil pengujian pada Tabel 5.16, didapati bahwa rata – rata waktu yang diperlukan dari data dikirim dari Arduino ke server hingga notifikasi dikirimkan dan diterima adalah 3.041 menit atau 3 menit 2 detik yang mana waktu ini cukup responsif dan efisien apabila digunakan ketika berkendara karena data yang dikirim antara waktu 3 menit sudah pasti akan memiliki perbedaan, baik dari lokasi maupun kecepatan.

### 5.4.5. Pengujian Performa Akurasi Sistem dengan Metode Pengambilan Keputusan *Fuzzy*

Pengujian performa akurasi sistem dilakukan dengan tujuan mengetahui tingkat akurasi dari sistem yang menggunakan metode pengambilan keputusan *Fuzzy*. Pengujian ini dilakukan dengan melakukan perbandingan hasil keluaran *Fuzzy* dari server terhadap data yang diolah dengan keadaan di lapangan. Pada pengujian ini *threshold* dari detak jantung diperkecil, sehingga batas atas detak jantung maksimal menjadi 100 detak jantung per menit, hal ini dilakukan dalam pengujian agar dapat mendapatkan pengiriman notifikasi dengan data detak jantung yang tinggi. Pengujian ini dilakukan dengan prosedur pengujian pada Tabel 5.17 berikut.

**Tabel 5.17 Prosedur Pengujian Performa Akurasi Sistem dengan Meotde Pengambilan Keputusan *Fuzzy***

| | |
|---|---|
| Nama | Pengujian Akurasi Sistem dengan Metode Pengambilan Keputusan *Fuzzy* |
| Kode | UJ-12 |
| Tujuan Pengujian | Menguji akurasi sistem dengan metode pengambilan *Fuzzy* apabila dibandingkan dengan keadaan di lapangan |
| Kondisi Awal | - |
| Prosedur Pengujian | Pengujian dilakukan oleh pengguna |

| | |
|---|---|
| | dengana menggunakan *Pulse* sensor dan berkendara hingga 10 data berhasil dikirim, bersamaan itu pula dilakukan pengamatan kondisi pengguna dengan hasil pengolahan data yang didapat dari *Fuzzy* |
| Masukkan | 10 buah data sensor beserta lokasi koordinat pengguna dan 10 buah data hasil pengamatan kondisi di lapangan |
| Keluaran | Tingkat akurasi dari sistem dengan menggunakan metode pengambilan keputusan *Fuzzy* yang merupakan hasil perbandingan dari kondisi di lapangan |
| Hasil Uji Coba | Sistem dengan metode pengambilan keputusan *Fuzzy* sudah cukup akurat dengan 1 data tidak cocok dengan keadaan sebenaranya di lapangan, sehingga akurasi dari 10 data yang diproses adalah 90% |

Berdasarkan data yang diperoleh dari pengujian skenario sebelumnya pada Pengguna A dan B, diperoleh hasil dari proses pengambilan keputusan pada masing – masing data yang dapat dilihat pada Tabel 5.18 berikut.

**Tabel 5.18 Hasil Pengujian Performa Akurasi Sistem dengan Metode Pengambilan Keputusan *Fuzzy***

| Uji Coba Ke | v (km/ jam) | a ($m/s^2$) | *hr* (detak per menit) | Hasil *Fuzzy* | Kondisi di Lapangan |
|---|---|---|---|---|---|
| 1 | 46.3 | 0.00 | 88 | Dangerous | Normal |
| 2 | 38.43 | -0.04 | 79 | Potential | Potential |
| 3 | 25.05 | -0.05 | 81 | Potential | Potential |
| 4 | 71.76 | 0.23 | 105 | Dangerous | Dangerous |

| | | | | | |
|---|---|---|---|---|---|
| 5 | 61.58 | -0.06 | 101 | Dangerous | Dangerous |
| 6 | 62.5 | 0.01 | 83 | Dangerous | Potential |
| 7 | 43.15 | -0.09 | 89 | Dangerous | Potential |
| 8 | 24.3 | -0.09 | 76 | Normal | Normal |
| 9 | 15.28 | -0.03 | 80 | Potential | Potential |
| 10 | 21.76 | 0.03 | 87 | Potential | Potential |

Berdasarkan data yang didapat pada Tabel 5.18, didapati bahwa terdapat 1 buah data yang tidak cocok dengan kondisi di lapangan, hal ini terjadi karena pengambilan data baru dilakukan dan percepatan yang terhitung memerlukan data sebelumnya dimana apabila berbeda jam maka percepatan dihitung nol. Sehingga berdasarkan hasil pengujian ini dimana terdapat 1 percobaan yang tidak cocok dengan kondisi di lapangan pada 20 kali percobaan, maka dapat dihitung bahwa tingkat akurasi dari sistem dengan menggunakan metode pengambilan *Fuzzy* ini cukup tinggi yaitu 90%.

*[Halaman ini sengaja dikosongkan]*

# BAB VI
# KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan tugas akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

## 6.1. Kesimpulan

Dari proses pengerjaan selama perancangan, implementasi, dan proses pengujian aplikasi yang dilakukan, dapat diambil kesimpulan sebagai berikut.

- Sistem ini sudah cukup akurat dalam melakukan pembacaan detak jantung yang menggunakan *Pulse* sensor yang mana rata – rata selisih pembacaaannya dengan alat standar medis seperti tensimeter ABN DW-200 cukup kecil yaitu selisih sebesar 2.96 dimana tingkat akurasi alat ini dengan detak jantung sebenarnya adalah ±5 detak jantung.
- Sistem ini berhasil melakukan pembacaan kecepatan dalam proses pemantauan pengguna, dengan modul GPS pada GPS/GPRS/GSM *Shield* v3.0 yang mana rata – rata selisih pembacaannya dengan *speedometer* dari kendaraan cukup kecil yaitu sebesar 1.65 km/jam dari kecepatan kendaraan sebenarnya, sehingga pembacaan kecepatan oleh sistem sudah cukup akurat.
- Rata – rata lama waktu yang dibutuhkan oleh sistem untuk mendapatkan data dan mengirimkan notifikasi sampai notifikasi diterima adalah 3.041 menit atau 3 menit 2 detik, dimana dalam sistem ini cukup responsif dan efisien karena data diproses setiap 3 menit sehingga terdapat beberapa waktu untuk menempuh jarak tertentu dalam berkendara.
- Penggunaan metode *Fuzzy Logic* dalam pengambilan keputusan sudah cukup membantu sistem untuk menjadi

cukup akurat dengan tingkat akurasi 90% terhadap total 10 data yang diproses jika dibandingkan dengan kondisi nyata di lapangan.

## 6.2. Saran

Berikut saran-saran untuk pengembangan dan perbaikan sistem di masa yang akan datang. Di antaranya adalah sebagai berikut:

- Fungsionalitas sistem dapat dikembangkan lagi dengan menggunakan jenis sensor lainnya untuk memantau kondisi pengguna yang lain, seperti sensor suhu dan kelembapan.
- Komunikasi antara sensor dan *shield* dapat dikembangkan dengan menggunakan *soft serial* pada Arduino untuk mengurangi *delay*.
- Ketepatan dan kecepatan pembacaan koordinat GPS dapat dikembangkan lagi dengan menggunakan modul GPS tipe lain dan antena eksternal.

# DAFTAR PUSTAKA


[1] A. Ramalingan, P. Dorairaj, and S. Ramamoorthy. Personal Safety Triggering System on Android. [Online]. http://arxiv.org/ftp/arxiv/papers/1208/1208.3138.pdf

[2] J. M. Arnold MD FRCPC, D. H. Fitchett MD FRCPC, J. G. Howlett MD FRCPC, E. M. Lonn MD FRCPC, and J.-C. Tardif MD FRCPC, "Resting heart rate: A modifiable prognostic indicator of cardiovascular risk and outcomes?," *The Canadian Journal of Cardiology*, vol. 24, pp. 3A-8A, 2008.

[3] (2015, May) EKG. [Online]. http://prodia.co.id/pemeriksaan-penunjang/ekg

[4] R. Sanjoyo, "Sistem Kardiovaskuler," Universitas Gajah Mada Laporan Tugas Akhir, 2005.

[5] Syaifuddin, *Anatomi Fisiologi Untuk Siswa Perawat Edisi 2*. Jakarta: Penerbit Buku Kedokteran EGC, 1997.

[6] J. Cameron, *Physics of the Body Second Edition*. Medison: Medical Physics Publisher, 1999.

[7] S. Cweiner. (2015, May) The Circulatory System. [Online]. http://www.glogster.com/seamcweiner/the-circulatory-system/

[8] K. Fox, et al., "Resting Heart Rate in Cardiovascular Disease," *Journal of the American College of Cardiology*, vol. 50, no. 9, 2007.

[9] T. K. Mishra and P. K. Rath, "Pivotal role of heart rate in health and disease," *Journal of Indian Academy of Clinical Medicine*, vol. 12, no. 4, pp. 297-302, 2011.

[10] A. Nonaka, H. Shiotani, K. Kitano, and M. Yokoyama, "Determinants of Heart Rate Recovery in Patients with Suspected Coronary Artery Disease," *Kobe Journal of Medical Sciences*, vol. 53, no. 3, pp. 93-98, 2007.

[11] M. Uchikune, "Physiological and Psychological Effects of High Speed Driving on Young Male Volunteers," *Journal of Occupational Health*, vol. 44, pp. 203-206, 2002.

[12] S. P. Wijaya, Y. Christiyono, and S. Unknown, "Alat Pelacak Lokasi Berbasis GPS Via Komunikasi Seluler," *TRANSMISI*, vol. 12, no. 2, pp. 82-86, 2010.

[13] (2015, May) Penjelasan GPS NMEA 0183. [Online]. http://www.mikron123.com/index.php/Aplikasi-GPS/Penjelasan-GPS-NMEA-0183.html

[14] Wikipedia. (2015, May) Global System for Mobile Communications. [Online]. http://id.wikipedia.org/wiki/Global_System_for_Mobile_Communications

[15] Wikipedia. (2015, May) GPRS. [Online]. http://id.wikipedia.org/wiki/GPRS

[16] L. Welling, *PHP and MySQL Web Development*. Indianapolis, United States of America: Sams Publishing, 2001.

[17] M. Čihař, M. Wiącek, and Gnokii. (2015, Jan.) Gammu. [Online]. 4

[18] Android. (2014, Jan.) Android Developers. [Online]. https://developer.android.com/google/gcm/gcm.html

[19] Android. (2015, May) Android Developers. [Online]. https://developers.google.com/maps/documentation/android/

[20] P. Cingolani. (2015, Jun.) jFuzzyLogic. [Online]. http://jfuzzylogic.sourceforge.net/html/index.html

[21] (2014, ) Arduino Uno Board. [Online]. http://arduino.cc/en/Main/arduinoBoardUno

[22] SparkFun Electronics. (2014, Jan.) sparkfun. [Online]. https://www.sparkfun.com/products/11574

[23] DFrobot. (2014, ) GPS/GPRS/GSM Module V3.0 (SKU:TEL0051). [Online].

http://www.dfrobot.com/wiki/index.php/GPS/GPRS/GSM_Module_V3.0_%28SKU:TEL0051%29

[24] N. K. Kasabov, *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*. London, England: The MIT Press, 1998.

*[Halaman ini sengaja dikosongkan]*

# BIODATA PENULIS

Penulis, Anggarda Diajeng Prameswari lahir di Magetan, Jawa Timur, pada tanggal 11 Juli 1993. Penulis adalah anak ke 2 dari 2 bersaudara dan dibesarkan di Madiun.

Penulis menempuh pendidikan formal di SD Negeri 09 Madiun Lor (1999-2005), SMP Negeri 1 Madiun (2005-2007) dan SMA Negeri 2 Madiun (2007-2011). Pada tahun 2011, penulis menempuh pendidikan S1 jurusan Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember, Surabaya, Jawa Timur.

Di jurusan Teknik Informatika, penulis mengambil bidang minat Komputasi Berbasis Jaringan dan aktif mengikuti kegiatan seminar yang diadakan oleh Jurusan dan Himpunan Mahasiswa Teknik Computer-Informatika (HMTC). Penulis dapat dihubungi melalui alamat email ajprameswari@gmail.com