



UNIVERSITAS INDONESIA

PROTOTYPE SISTEM TERINTEGRASI PENDETEKSIAN DINI
DAN *MONITORING* PENYAKIT JANTUNG BERBASIS SINYAL
ELEKTROKARDIOGRAM

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Ilmu
Komputer

MUHAMMAD ANWAR MA'SUM

0906510382

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
DEPOK
JANUARI 2013

HALAMAN PERNYATAAN ORISINALITAS

**Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.**

Nama : Muhammad Anwar Ma'sum

NPM : 0906510382

Tanda Tangan : _____

Tanggal : 10 Januari 2013

HALAMAN PERSETUJUAN

Judul : *Prototype* Sistem Terintegrasi Pendeteksian Dini dan *Monitoring*

Penyakit Jantung Berbasis Sinyal Elektrokardiogram.

Nama : Muhammad Anwar Ma'sum

NPM : 0906510382

Laporan Skripsi ini telah diperiksa dan disetujui.

10 Januari 2013

Dr. Eng. Wisnu Jatmiko S.T., M.Kom
Pembimbing Skripsi

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh:

Nama : Muhammad Anwar Ma'sum

NPM : 0906510382

Program Studi : Ilmu Komputer

Judul Skripsi : *Prototype* Sistem Terintegrasi Pendeteksian Dini dan *Monitoring* Penyakit Jantung Berbasis Sinyal Elektrokardiogram.

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana S.Kom pada program Studi Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Dr. Eng Wisnu Jatmiko M. Kom (.....)

Penguji : M. Ivan Fanany, S.Si, M.Kom, Ph.D. (.....)

Penguji : Samuel Louvan, S.Kom, M.Sc (.....)

Ditetapkan di : Fakultas Ilmu Komputer

Tanggal : 10 Januari 2013

KATA PENGANTAR

Penulis mengucapkan puji dan syukur kehadiran Allah SWT, dengan izin-Nya penulis dapat menyelesaikan kegiatan penelitian dan penyusunan laporan tugas akhir dengan judul: *Prototype Sistem Terintegrasi Pendeteksian Dini dan Monitoring Penyakit Jantung Berbasis Sinyal Elektrokardiogram*.

Pada kesempatan ini, penulis ingin menyampaikan terima kasih yang kepada semua pihak yang telah ikut serta memberikan dukungan serta bantuan mencakup dorongan semangat dan moral, sehingga akhirnya kegiatan penelitian tugas akhir ini dapat berjalan lancar seperti sebagaimana mestinya. Penulis mengucapkan terimakasih kepada:

1. Kedua orang tua penulis Imam Ma'sum dan Sumarni, serta adik penulis yaitu Muhammad Zainal Abidin dan Hani'atul Khoiriah yang selalu memberikan dukungan moril maupun materil serta menyertai langkah penulis dalam setiap doa mereka.
2. Bapak Ir. Wisnu Jatmiko, M.Kom., Dr.Eng., selaku pembimbing tugas akhir yang telah mengarahkan penulis dalam melaksanakan kegiatan penelitian ini hingga proses penyusunan laporan.
3. Ir. Adhi Yuniarto L.Y. M.Sc., selaku Pembimbing Akademik yang telah memberikan arahan kepada penulis selama masa studi di Fakultas Ilmu Komputer, Universitas Indonesia.
4. Seluruh Staf Dosen Fasilkom yang telah membimbing dalam setiap mata kuliah yang penulis ambil.
5. Mas Iqbal, Mas Alvis, Mas Indra, Mas Zaki (Bang Jek), Mas Ali, Mas Andri, Mas Ari, Mas Tommy, Mas Ali selaku asisten riset lab 1231 yang telah menyalurkan ilmu dan membantu penulis dalam penelitian ini.
6. Rekan-rekan seperjuangan di Lab 1231. yang telah berbagi pengetahuan dan pengalaman selama kegiatan penelitian.
7. Mahasiswa Fasilkom Angkatan 2009 yang tidak bisa disebutkan satu persatu yang selama 3.5 tahun ini bersama-sama, terima kasih atas segala pengalaman baik suka dan duka yang telah diberikan

8. Tim Robotika Universitas Indonesia (TRUI) yang menjadi wadah bagi penulis dalam belajar ilmu pengetahuan dan teknologi serta mengembangkan diri dalam berbagai aspek.
9. Semua pihak yang telah membantu penulis dalam pengerjaan tugas akhir ini.

Penulis menyadari masih terdapat kekurangan pada penyusunan laporan ini. Oleh karena itu, penulis dengan tangan terbuka bersedia menerima kritik dan saran yang berguna. Semoga karya ini bermanfaat bagi pembaca semua.

Depok, 10 Januari 2013

Muhammad Anwar Ma'sum

**HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI KARYA
ILMIAH UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademik Universitas Indonesia, saya yang bertandatangan di bawah ini:

Nama : Muhammad Anwar Ma'sum
NPM : 0906510382
Program Studi : Ilmu Komputer
Fakultas : Ilmu Komputer
Jenis Karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty-Free Right)** atas karya ilmiah saya yang berjudul:

Prototype Sistem Terintegrasi Pendeteksian Dini dan *Monitoring* Penyakit Jantung Berbasis Sinyal Elektrokardiogram.

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalih media/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 10 Januari 2013
Yang menyatakan

Muhammad Anwar Ma'sum

ABSTRAK

Nama : Muhammad Anwar Ma'sum

Program Studi : Ilmu Komputer

Judul : *Prototype Sistem Terintegrasi Pendeteksian Dini dan Monitoring Penyakit Jantung Berbasis Sinyal Elektrokardiogram.*

Penyakit jantung merupakan penyakit mematikan nomor satu di dunia dan juga di Indonesia. Salah satu penyebab utama penyakit jantung yang akut adalah tidak terdeteksinya gejala penyakit sejak awal. Untuk mencegah bertambahnya korban kematian akibat penyakit jantung dibutuhkan suatu sistem pendeteksian dini dan *monitoring* penyakit jantung. Dengan sistem pendeteksian dini dan *monitoring* penyakit jantung berbasis elektrokardiogram, gelombang detak jantung dapat visualisasikan, diklasifikasikan dalam kategori sehat atau tidak. Dengan demikian gejala penyakit dapat diketahui sejak awal. Sistem ini diimplementasikan dalam perangkat *smartphone Android*. Selain itu agar dokter dapat melakukan verifikasi data detak jantung yang sudah diambil dengan alat, perlu dibangun suatu sistem terintegrasi dengan memanfaatkan teknologi komunikasi dan informasi. Setelah dilakukan penelitian, sistem terintegrasi ini dapat diwujudkan, dan dapat berjalan dengan baik. Selain itu tingkat responsivitas dalam pengiriman atau pengunduhan data ke / dari *server* pada *ptototype* ini cukup cepat, sehingga dapat dikembangkan lebih lanjut menjadi produk nyata.

Kata Kunci:

Sistem terintegrasi pendeteksian dini dan *monitoring* penyakit jantung, elektrokardiogram, *smartphone Android*, *server*

ABSTRACT

Name : Muhammad Anwar Ma'sum

Study Program : Ilmu Komputer

Title : *Prototype* of Integrated Sistem for Early Detection and *Monitoring* of Heart Disease Based on Electrocardiogram Signal.

Heart disease is the number one cause of death in the world and Indonesia. One of the reason of acute heart disease is because the disease is not detected in early stage. To prevent a great number victim, an early detection and *monitoring* sistem for heart is needed. With early detection and *monitoring* sistem, heartbeat can be visualized. Therefore, the heart disease can be detected earlier. This sistem is implemented in Android smartphone device. To accommodate the doctor to verify the acquired heartbeat signals, we need to build an integrated sistem by using information communication technology. This sistem has been realized in this thesis. The responsitivity for sending and receiving data from or to *server* is quite high, therefore this *prototype* can be developed become real product.

Key words:

Integrated Sistem, Early Detection and *Monitoring*, Heart Disease, Electrocardiogram, Android Smartphone, *Server*

DAFTAR ISI

HALAMAN PERNYATAAN ORISINALITAS	ii
HALAMAN Persetujuan	iii
HALAMAN PENGESAHAN	iv
KATA PENGANTAR	v
HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS	vii
ABSTRAK	viii
ABSTRACT	ix
DAFTAR ISI	x
DAFTAR GAMBAR	xiii
DAFTAR TABEL	xvii
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.1. Rumusan Masalah	4
1.2. Tujuan Penelitian	4
1.3. Ruang Lingkup Penelitian	5
1.4. Posisi Penelitian	5
1.5. Metodologi Penelitian	7
1.6. Sistematika Penulisan	8
BAB 2 landasan teori	10
2.1. Elektrokardiogrami	10
2.2. Penyakit Jantung	11
2.2.1. Fuzzy Learning Vector Quantization (FLVQ)	13
2.3. Sistem Operasi Android	14
2.4. Perangkat Keras Elektrokardiogram <i>Single Lead</i>	16
2.4.1. Transduser	18
2.4.2. Penguat awal	18
2.4.3. High Pass Filter (HPF)	20
2.4.4. Low Pass Filter (LPF)	20
2.4.5. Adder	21
2.5. Aplikasi Sistem Pendeteksian Dini Penyakit Jantung (E-Cardio)	22
2.5.1. E-Cardio versi <i>Smartphone</i> Android	22

2.5.2.	E-Cardio versi <i>Personal Computer (PC) Desktop</i>	24
2.5.3.	Teknologi Java <i>Enterprise Edition (Java EE)</i>	25
BAB 3	Rancangan Penelitian	28
3.1.	Rancangan Arsitektur Sistem	28
3.2.	Rancangan Modifikasi Aplikasi E-Cardio	32
3.2.1.	Use case Diagram	33
3.2.2.	Use case Specification	34
3.3.	Rancangan Implementasi Database	46
3.4.	Rancangan Implementasi <i>Server</i>	47
3.5.	Rancangan Interaksi Antar Komponen	49
3.6.	Data Detak Jantung yang Digunakan	50
3.6.1.	Kelas Detak Jantung Normal	50
3.6.2.	AFib	51
3.6.3.	2-BLK 1	52
3.6.4.	RBBB	53
3.6.5.	PAC	53
3.6.6.	PVC EARLY	54
3.6.7.	PVC R-On-T	54
3.6.8.	MF PVC	55
3.6.9.	BIGEMINY	55
3.6.10.	RUN 5 PVC	56
3.6.11.	V-TACH	56
3.6.12.	V-FIB	57
3.6.13.	PACED	57
BAB 4	Implementasi	59
4.1.	Perangkat Yang Digunakan	59
4.1.1.	Elektrokardiogram	59
4.1.2.	Mikrokontroller	59
4.1.3.	<i>Bluetooth</i>	61
4.1.4.	Kabel serial	62
4.1.5.	Konektor Elektroda	62
4.1.6.	Baterai	63
4.2.	Implementasi <i>Database</i>	63
4.3.	Implementasi Aplikasi <i>Web</i> pada <i>Server</i>	67
4.4.	Implementasi Modifikasi Aplikasi E-Cardio	73
4.4.1.	Implementasi Use Case Spesification : Mendaftarkan akun	73
4.4.2.	Implementasi Use Case Spesification : Menyimpan <i>History</i> Detak Jantung	74
4.4.3.	Implementasi Use Case Spesification : Melihat <i>History</i> Detak Jantung	75
4.4.4.	Implementasi Use Case Spesification : Setting	75

4.4.5. Implementasi Use Case Spesification : Mengirimkan <i>History</i> Detak Jantung Ke <i>Server</i>	76
4.4.6. Implementasi Use Case Spesification : Melihat Informasi Dokter	77
4.4.7. Implementasi Use Case Spesification : Melihat Informasi Rumah Sakit dan Klinik.....	78
4.4.8. Implementasi Use Case Spesification : Melakukan Verifikasi Data Detak Jantung	79
4.4.9. Implementasi Use Case Spesification : Mendaftarkan Data Rumah Sakit dan Klinik.....	80
4.4.10. Implementasi Use Case Spesification : Mendaftarkan Afiliasi Dokter	81
4.5. Implementasi Pembuatan Restful <i>Webservice</i>	82
4.6. Implementasi Sinkronisasi Data.....	83
BAB 5 hasil dan evaluasi	84
5.1. Rancangan Uji Coba	84
5.2. Hasil Uji Coba.....	85
5.3. Analisis.....	86
BAB 6 PENUTUP.....	87
6.1. Kesimpulan	87
6.2. Saran.....	88
DAFTAR PUSTAKA	89

DAFTAR GAMBAR

Gambar 1-1. Rasio penyebab kematian di Indonesia akibat berbagai penyakit menurut data WHO tahun 2002.	2
Gambar 1-2. Peta persebaran dokter spesialis penyakit jantung di Indonesia berdasarkan provinsi.	3
Gambar 1-3. Diagram posisi penelitian.	7
Gambar 2-1. Posisi keenam elektroda dada untuk elektrokardiogrami standar	11
Gambar 2-2. Arsitektur FLVQ.	13
Gambar 2-3. Pencarian similaritas.	14
Gambar 2-4. Diagram arsitektur perangkat lunak sistem operasi Android.	15
Gambar 2-5. EKG <i>single lead</i>	17
Gambar 2-6. Modul – modul penyusun rangkaian elektrokardiogram (EKG).	18
Gambar 2-7. Penguat instrumentasi.	19
Gambar 2-8. Rangkain <i>Right Leg Driver Loop</i> untuk meminimalkan <i>noise</i>	19
Gambar 2-9. Rangkaian HPF.	20
Gambar 2-10. <i>Low Pass Filter orde 3 butterworth</i>	21
Gambar 2-11. Rangkaian <i>Adder</i>	21
Gambar 2-12. Tampilan aplikasi E-Cardio di smartphone Android.	23
Gambar 2-13. Tampilan aplikasi E-Cardio di PC desktop.	24
Gambar 2-14. Arsitektur Java EE.	25
Gambar 2-15. Contoh kode sumber servlet.	26
Gambar 2-16. Contoh kode program halaman <i>web</i> dengan JSP.	27

Gambar 3-1. Arsitektur sistem pendeteksian dini dan <i>monitoring</i> penyakit jantung.....	28
Gambar 3-2. Sistem kerja komponen <i>hardware</i> pada <i>prototype</i> system.	29
Gambar 3-3. Sistem kerja komponen <i>smartphone android</i> pada <i>prototype</i> sistem.	31
Gambar 3-4. Diagram fungsional <i>server</i> pada <i>prototype</i> sistem.....	32
Gambar 3-5. <i>Use Case</i> diagram pada aplikasi E-Cardio versi 3.0.....	33
Gambar 3-6. Diagram <i>entity relationship</i> untuk <i>prototype</i>	47
Gambar 3-7. Diagram interaksi komponen-komponen <i>prototype</i> system.	49
Gambar 3-8. Gambar lima kelas contoh gelombang detak jantung manusia normal	51
Gambar 3-9. Contoh detak jantung manusia penderita AFib yang diambil menggunakan pasien simulator.	52
Gambar 3-10. Contoh detak jantung manusia penderita 2-BLK 1 yang diambil menggunakan pasien simulator.	52
Gambar 3-11. Contoh detak jantung manusia penderita RBBB yang diambil menggunakan pasien simulator.	53
Gambar 3-12. Contoh satu detak jantung manusia penderita PAC yang diambil menggunakan pasien simulator.....	53
Gambar 3-13. Contoh satu detak jantung manusia penderita PVC <i>Early</i> yang diambil menggunakan pasien simulator.....	54
Gambar 3-14. Contoh detak jantung manusia penderita PVC R-On-T yang diambil menggunakan pasien simulator.....	54
Gambar 3-15. Contoh detak jantung manusia penderita MF PVC yang diambil menggunakan pasien simulator.	55

Gambar 3-16. Contoh detak jantung manusia penderita <i>Bigeminy</i> yang diambil menggunakan pasien simulator.	55
Gambar 3-17. Contoh detak jantung manusia penderita RUN 5 PVC yang diambil menggunakan pasien simulator.	56
Gambar 3-18. Contoh satu detak jantung manusia penderita V-TACH yang diambil menggunakan pasien simulator.	56
Gambar 3-19. Contoh detak jantung manusia penderita V-FIB yang diambil menggunakan pasien simulator.	57
Gambar 3-20. Contoh satu detak jantung manusia penderita PACED yang diambil menggunakan pasien simulator.	58
Gambar 4-1. Elektrokardiogram yang digunakan pada <i>prototype</i> system.	59
Gambar 4-2. Mikrokontroler ATmega8L.	60
Gambar 4-3. Diagram konversi data menggunakan fitur ADC.	61
Gambar 4-4. Format pengiriman dan penerimaan data serial.	61
Gambar 4-5. Modul <i>bluetooth</i> to serial converter.	62
Gambar 4-6. Kabel serial.	62
Gambar 4-7. Konektor elektroda, dari kiri kekanan terhubung dengan elektroda (L, GND, dan R).	63
Gambar 4-8. Baterai berseta charger yang digunakan pada <i>prototype</i>	63
Gambar 4-9. Tabel <i>patient</i> pada <i>database</i>	64
Gambar 4-10. Tabel <i>doctor</i> pada <i>database</i>	65
Gambar 4-11. Tabel <i>hospital</i> pada <i>database</i>	65
Gambar 4-12. Tabel <i>hospital</i> pada <i>database</i>	66
Gambar 4-13. Tabel <i>affiliation</i> pada <i>database</i>	66

Gambar 4-14. Contoh Kode <i>Database</i> Versi <i>Lite</i>	67
Gambar 4-15. Contoh kode program servlet.....	68
Gambar 4-16. Arsitektur servlet pada <i>server</i>	69
Gambar 4-17. Contoh kode program servlet.....	71
Gambar 4-18. Contoh kode program servlet.....	73
Gambar 4-19. Implementasi <i>use case</i> mendaftarkan akun pada aplikasi E- Cardio.....	74
Gambar 4-20. Implementasi <i>use case</i> menyimpan <i>history</i> detak jantung.....	75
Gambar 4-21. Implementasi <i>use case</i> melihat <i>history</i> detak jantung	75
Gambar 4-22. Implementasi <i>use case</i> setting.....	76
Gambar 4-23. Implementasi <i>use case</i> mendaftarkan <i>history</i> detak jantung ke <i>server</i>	77
Gambar 4-24. Implementasi <i>use case</i> melihat informasi dokter.....	78
Gambar 4-25. Implementasi <i>use case</i> melihat informasi rumah sakit dan klinik.	79
Gambar 4-26. Implementasi verifikasi data detak jantung	80
Gambar 4-27. Implementasi mendaftarkan data rumah sakit dan klinik	81
Gambar 4-28. Implementasi mendaftarkan afiliasi dokter.....	82
Gambar 4-29. Contoh keluaran <i>web service</i> dalam sintaks XML.	82

DAFTAR TABEL

Tabel 4-1. Parameter <i>Request</i> dan Keluaran Servlet.....	69
Tabel 5-1. Parameter uji coba tingkat responsivitas <i>server</i>	84
Tabel 5-2. Hasil uji coba tingkat responsivitas <i>server</i>	85

BAB 1 PENDAHULUAN

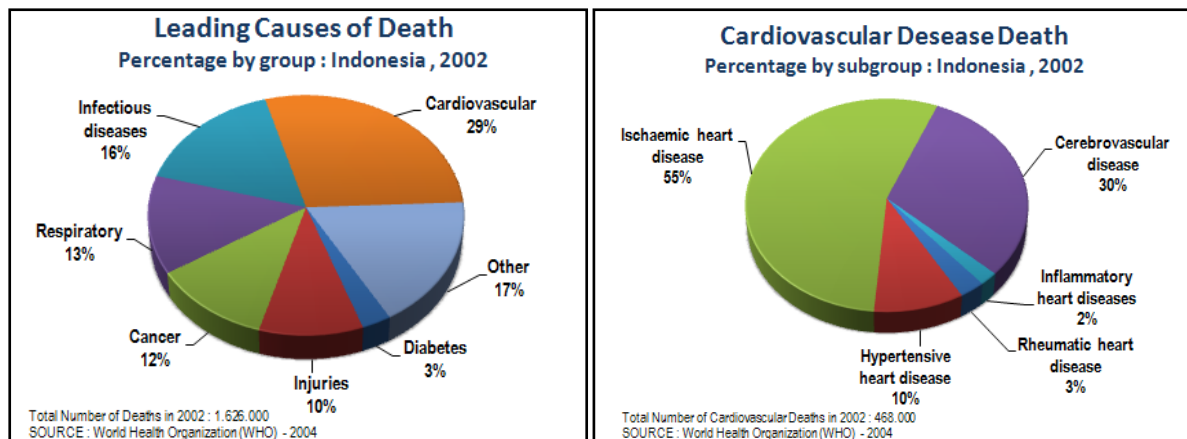
Bab pendahuluan merupakan bab yang memaparkan tentang latar belakang dari penelitian yang dilakukan, rumusan masalah yang diajukan, tujuan dari penelitian ini, posisi penelitian, ruang lingkup penelitian, metodologi penelitian, dan sistematika penulisan dokumen laporan penelitian ini.

I.1 Latar Belakang

Jantung merupakan salah satu organ yang sangat penting untuk kelangsungan hidup manusia. Jantung berperan untuk memompa darah agar bisa mengalir ke seluruh tubuh manusia. Dalam sistem organ, darah berfungsi untuk mengedarkan sari-sari makanan ke seluruh tubuh manusia. Apabila jantung mengalami gangguan atau disfungsi, maka proses pengedaran sari-sari makanan oleh darah akan terganggu. Hal ini tentu saja menimbulkan dampak yang buruk untuk tubuh manusia, karena sumber energi yang digunakan untuk proses metabolisme tersendat. Oleh sebab itu gangguan pada jantung akan mengakibatkan gangguan pada metabolisme sel-sel tubuh maupun kinerja organ-organ tubuh manusia.

Menurut data dari World Health Organisation (WHO) penyakit kardiovaskular (CVDs) merupakan penyakit yang menyebabkan kematian nomor satu di dunia. Pada tahun 2008, diperkirakan sekitar 17,3 juta orang meninggal akibat penyakit kardiovaskular[1]. Dari sekian juta orang yang menjadi korban CVDs tersebut 80% di antaranya merupakan warga negara dengan tingkat pendapatan menengah ke bawah. Diperkirakan, penderita penyakit jantung koroner naik sampai 137% untuk pria dan 120% untuk wanita dalam kurun waktu tahun 1990 – 2020 [2][3]. Selain itu WHO juga memperkirakan bahwa pada tahun 2030 hampir 23.6 juta orang meninggal karena menderita CVDs [1].

Di Indonesia sendiri penyakit jantung dan pembuluh darah juga merupakan penyakit mematikan nomor satu. Hasil catatan WHO, pada tahun 2002 penderita kardiovaskular mencapai 29% di antara penderita penyakit mematikan lainnya. Di antara seluruh kategori penyakit kardiovaskular, penyakit jantung iskemik merupakan penyakit yang paling banyak diderita masyarakat Indonesia yaitu mencapai 55%. Diagram presentase penderita penyakit jantung dan kardiovaskular serta penyakit mematikan lainnya dapat dilihat pada gambar 1-1.

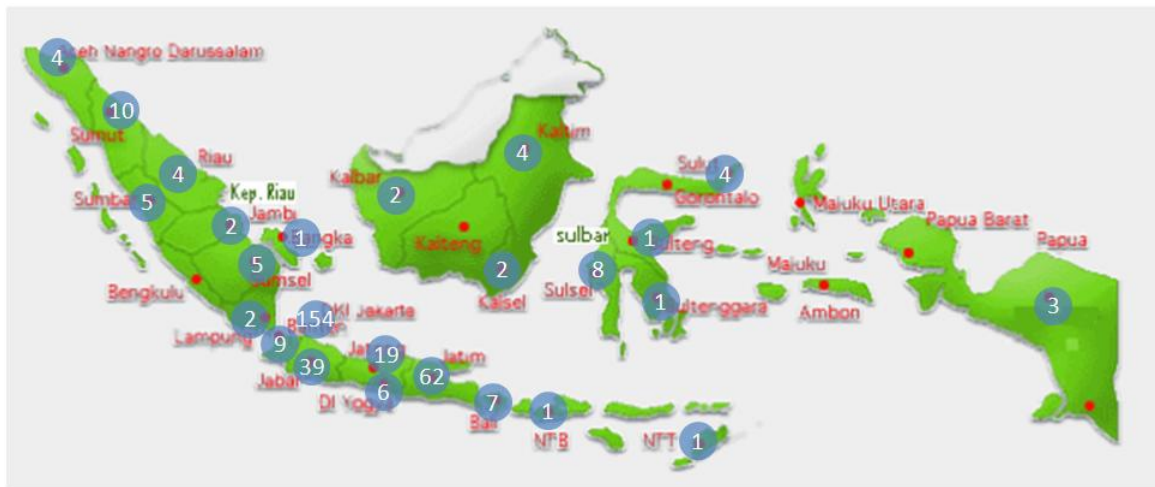


Gambar 1-1. Rasio penyebab kematian di Indonesia akibat berbagai penyakit menurut data WHO tahun 2002, (a) Rasio penderita penyakit jantung dibanding penyakit lainnya (b) Rasio penderita penyakit jantung iskemik dibandingkan penyakit kardiovaskular lainnya.

Berdasarkan fakta ini, diketahui bahwa penyakit jantung merupakan ancaman yang berbahaya bagi masyarakat Indonesia dan juga seluruh penduduk di dunia.

Di Indonesia, timbulnya penyakit kardiovaskular diakibatkan karena pola konsumsi dan aktivitas yang tidak sehat. Makanan yang mengandung kolesterol dalam kadar yang tinggi masih menjadi pilihan banyak orang. Selain itu, masyarakat juga masih susah menghindari makanan berlemak jenuh tinggi. Faktor lain penyebab penyakit jantung adalah minimnya aktivitas olah raga yang dilakukan setiap hari. Akan tetapi, penyebab utama banyaknya penderita penyakit jantung dan kardiovaskular di Indonesia adalah karena perawatan kesehatan jantung yang kurang serius. Kebanyakan dari masyarakat Indonesia tidak memerikasakan kondisi kesehatannya secara berkala. Hal inilah yang membuat penyakit tidak dapat diketahui sejak dini. Fenomena ini memang tidak sepenuhnya merupakan kesalahan dari masyarakat, karena jumlah kardiologis dan pusat-pusat pemeriksaan dan perawatan jantung di Indonesia masih sangat minim. Berdasarkan data yang diperoleh dari Perhimpunan Dokter Spesialis Kardiovaskular Indonesia (PERKI) jumlah dokter spesialis jantung di Indonesia hanya berjumlah 356 orang[2]. Sedangkan jumlah penduduk di Indonesia pada tahun 2010 lebih dari 273 orang[4]. Berdasarkan data tersebut dapat diketahui bahwa jumlah dokter spesialis jantung jauh lebih sedikit dibandingkan dengan jumlah penduduk di Indonesia. Perbandingannya kurang lebih 1 :665.730. Ditambah fakta lagi bahwa kebanyakan dokter spesialis jantung di Indonesia tidak tersebar merata di seluruh provinsi dan daerah di Indonesia. Kebanyakan dari mereka bertempat tinggal di pulau jawa,

terutama di kota – kota besar. Penyebaran dokter spesialis jantung di Indonesia dapat dilihat pada gambar 1-2.



Gambar 1-2. Peta persebaran dokter spesialis penyakit jantung di Indonesia berdasarkan provinsi [1].

Berdasarkan fakta-fakta di atas diperlukan suatu metode untuk mencegah penyakit jantung. Dengan pemeriksaan dan *monitoring* secara berkala gejala-gejala penyakit jantung dapat diketahui sejak dini. Dengan demikian penyakit jantung yang lebih serius dapat dihindari. Penelitian tentang jantung memberikan data bahwa data detak jantung seseorang sangat bergantung pada kondisi kesehatan tubuh, kondisi psikologi, dan kondisi-kondisi lainnya[5]. Penelitian lainnya menyebutkan bahwa terdapat hubungan yang erat antara detak jantung seseorang dengan berbagai penyakit dan juga ketidak normalan dalam tubuh seseorang[6]. Dengan demikian ada tidaknya penyakit atau kelainan jantung dapat diketahui melalui data detak jantung orang yang bersangkutan.

Perkembangan sains dan teknologi telah menghasilkan alat yang dapat digunakan untuk mengambil data detak jantung manusia, yaitu elektrokardiogram. Di Indonesia, elektrokardiogram masih merupakan alat yang masih langka, sekalipun di rumah sakit atau pusat kesehatan. Hal ini dikarenakan alat tersebut didatangkan dari negara lain dengan harga yang mahal. Selain itu pada umumnya elektrokardiogram yang digunakan di Indonesia berukuran sangat besar dan membutuhkan sumber tenaga listrik yang cukup besar. Ditambah lagi, kebanyakan elektrokardiogram jenis ini hanya mampu melakukan visualisasi gelombang dan pencetakan gelombang detak jantung. Elektrokardiogram ini belum mampu memprediksi kondisi jantung pasien apakah pasien menderita penyakit jantung atau tidak, atau dengan kata lain belum memiliki kecerdasan.

Berdasarkan latar belakang fakta-fakta di atas, penulis berinisiatif untuk merancang suatu sistem terintegrasi pendeteksian dini dan *monitoring* penyakit jantung yang cerdas dan praktis. Sistem ini akan menggunakan elektrokardiogram untuk mengambil sinyal detak jantung dari manusia. Selanjutnya sistem ini dapat memprediksi kondisi jantung seseorang berdasarkan sinyal elektrokardiogram yang diambil dari orang tersebut. Akan tetapi agar dapat digunakan oleh masyarakat umum, sistem ini harus dibuat dengan praktis. Selain itu, sistem ini juga akan dibuat dengan kebutuhan konsumsi daya seminimal mungkin. Lebih jauh lagi, sistem yang diajukan ini juga dirancang agar sinyal detak jantung yang telah diambil dari elektrokardiogram dapat diverifikasi oleh dokter. Proses verifikasi oleh dokter dirancang dengan teknologi komunikasi, sehingga dokter dan pengguna sistem ini tidak harus bertemu langsung. Artinya, data dikirimkan oleh pasien kepada dokter melalui jaringan internet menggunakan suatu *server* maupun jaringan telekomunikasi lainnya. Dengan demikian, sinyal detak jantung pengguna yang berada di daerah kota kecil tetap dapat diperiksa oleh dokter spesialis jantung.

1.1. Rumusan Masalah

Adapun rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana bentuk rancangan *prototype* sistem terintegrasi pendeteksian dini dan *monitoring* penyakit jantung berbasis sinyal elektrokardiogram?
2. Bagaimanakah sistem komunikasi antar komponen dalam *prototype* sistem terintegrasi pendeteksian dini dan *monitoring* penyakit jantung berbasis sinyal elektrokardiogram?
3. Bagaimanakah implementasi struktur basis data yang tepat agar kebutuhan penyimpanan data untuk keperluan verifikasi oleh dokter?
4. Bagaimanakah implementasi sinkronisasi data di perangkat milik pasien, dokter, dan *server* yang efektif dan efisien?
5. Seberapa cepat responsivitas *server* untuk pengiriman data dari pasien ke *server* dan dari *server* ke dokter untuk keperluan verifikasi sinyal detak jantung?

1.2. Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Mengimplementasikan suatu *prototype* sistem terintegrasi pendeteksian dini dan *monitoring* penyakit jantung berbasis sinyal elektrokardiogram.

2. Mengimplementasikan sistem komunikasi antar komponen dalam *prototype* sistem terintegrasi pendeteksian dini dan *monitoring* penyakit jantung berbasis sinyal elektrokardiogram untuk mencapai seluruh fungsi sistem yang diinginkan.
3. Mengimplementasikan struktur basis data yang tepat agar kebutuhan penyimpanan data untuk keperluan verifikasi oleh dokter.
4. Mengimplementasikan metode sinkronisasi dalam pengelolaan basis data yang dibuat.
5. Mengukur performa responsivitas *server* untuk pengiriman data dari pasien ke *server* dan dari *server* ke dokter untuk keperluan verifikasi sinyal detak jantung.
6. Memperkaya pengetahuan dan pengalaman dalam pembuatan perangkat lunak.

1.3. Ruang Lingkup Penelitian

Berdasarkan tujuan yang ingin dicapai dalam penelitian, maka penelitian ini dibatasi pada ruang lingkup sebagai berikut:

1. Penggunaan elektrokardiogram (EKG) yang dikembangkan di laboratorium *Architecture, Network and High Performance Computing* sebagai komponen perangkat keras (*hardware*).
2. Pembuatan basis data untuk penyimpanan data-data yang dibutuhkan baik secara *online* maupun secara *offline*.
3. Pembuatan *server* untuk menghubungkan dokter dan pasien.
4. Modifikasi aplikasi pendeteksian dini penyakit jantung (E-Cardio) yang telah dikembangkan sebelumnya, dengan fitur-fitur yang menunjang verifikasi oleh dokter, penyimpanan *online*, dan sinkronisasi dengan *server*.
5. Uji coba implementasi yang dilakukan dengan beberapa skenario.

1.4. Posisi Penelitian

Penelitian ini merupakan tahap lanjutan dari penelitian – penelitian sebelumnya. Posisi penelitian ini dapat dilihat pada gambar 1-3. Penelitian ini dapat dikatakan adalah fase integrasi dan penyempurnaan dari penelitian tentang pembuatan sistem pendeteksian dini penyakit jantung yang telah dilaksanakan pada tahun 2011. Pada penelitian yang diadakan tahun 2011 tersebut sudah diimplementasikan *prototype* sistem versi pertama. Komponen *prototype* versi pertama tersebut terdiri dari *hardware* yang berupa sensor EKG (ECG), dan aplikasi yang berjalan di *personal computer* (PC) desktop, dan di *smartphone* Android. Aplikasi yang dibuat di *smartphone* Android tersebut diberi nama E-Cardio. Kedua aplikasi

terebut dapat dikatakan dua aplikasi berbeda karena tidak dapat saling tukar *platform*. Meskipun demikian kedua aplikasi tersebut memiliki fitur- fitur yang sama. Fitur- fitur inti aplikasi versi pertama adalah visualisasi gelombang detak jantung, klasifikasi detak jantung, dan penyimpanan data detak jantung pada file di dalam sitem operasi.

Selanjutnya, pada tahun 2012 *hardwrae* yang berupa sensor EKG digunakan untuk penelitian lain dan di-*upgrade* menjadi sistem baru, yaitu alat untuk *monitoring* detak jantung dan *sleep stages*. Sistem baru tersebut menggunakan *single board computer* jenis *beagleboard* sebagai *main controller* dari alat tersebut. Selain itu, untuk keperluan *user interface* digunakan *liquid crystal display* (LCD) *touch screen* berukuran 4 inchi. Di sisi lain, pada aplikasi yang berjalan di *smartphone* Android juga dilakukan perbaikan *graphical user interface* (GUI) untuk membuat pengguna lebih nyaman saat menggunakan aplikasi. Aplikasi dengan GUI yang telah di-*upgrade* tersebut diberi nama E-Cardio versi 2.0.

Pada penelitian ini, aplikasi akan dibuat suatu *server* sehingga pasien dan dokter dapat saling berkomunikasi dan bertukar data. Selain itu, akan dibuat juga *database* untuk menyimpan data – data sinyal detak jantung, data pasien, data dokter, dan data data rumah sakit dan klinik kesehatan jantung. *Database* akan dibuat baik secara *online* yang terletak di *server*, maupun secara offline yang dibuat pada *smartphone* Android. Selanjutnya, pada alikasi E-Eardio 2.0 juga akan ditambahkan dengan menu-menu untuk keperluan pengiriman data dan verifikasi data antara pasien dengan dokter. Selain itu, untuk menjamin penyimpanan data yang tepat, akan dibuat juga sinkronisasi data antara *smartphone* dan *server*.



Gambar 1-3. Diagram posisi penelitian.

1.5. Metodologi Penelitian

Tahapan – tahapan yang dilakukan dalam penelitian ini adalah sebagai berikut :

1. Mengumpulkan dan mempelajari bahan-bahan studi pustaka dan literatur yang berkaitan dengan permasalahan dalam penelitian ini. Bahan- bahan yang dimaksud berupa dokumentasi penelitian-penelitian sebelumnya, metode-metode yang berhubungan dengan permasalahan yang diteliti, maupun teknologi – teknologi yang sesuai dengan penelitian ini.
2. Membuat basis data sesuai dengan kebutuhan data yang disimpan baik secara *online* (terletak di *server*) ataupun secara *offline* (terletak di *smartphone*)
3. Membuat suatu *server* untuk menghubungkan pasien dan dokter serta mengimplementasikan layanan untuk penyimpanan, dan pengiriman / penerimaan data.
4. Menambahkan menu pada aplikasi E-Cardio untuk keperluan verifikasi data, termasuk komunikasi dengan *server*

5. Melakukan uji coba performa pengiriman data dari dan ke *server* dengan beberapa skenario
6. Memberikan kesimpulan dari hasil penelitian dan saran untuk pengembangan lebih lanjut dari *prototype* yang dibuat.

1.6. Sistematika Penulisan

Sistematika penulisan laporan tugas akhir ini adalah sebagai berikut :

1. Bab 1 Pendahuluan, berisi penjelasan tentang latar belakang dilaksakannya penelitian dalam tugas akhir ini, rumusan masalah, tujuan pelaksanaan penelitian, ruang lingkup penelitian yang akan membatasi penelitian dalam tugas akhir ini, tahapan serta metodologi yang penulis gunakan dalam melaksanakan penelitian, dan terakhir menjelaskan tentang sistematika penulisan laporan.
2. Bab 2 Landasan Teori merupakan bab yang berisi teori – teori yang digunakan sebagai landasan dalam melaksanakan penelitian. Penelitian – penelitian sebelumnya, produk – produk, maupun teknologi yang berkaitan dengan penelitian ini juga akan dipaparkan dalam bagian tersebut. Paparan dalam bab tersebut bermanfaat untuk memberikan pemahaman yang lebih mendalam kepada penulis maupun pembaca mengenai teori-teori terkait.
3. Bab 3 Rancangan Penelitian merupakan bagian yang menjelaskan tentang rancangan atau arsitektur sistem yang akan dibuat, komponen-komponen yang digunakan dalam membuat sistem, rancangan komunikasi antar komponen pada sistem, dan juga data-data detak jantung yang digunakan dalam percobaan.
4. Bab 4 Implementasi merupakan bab yang menjelaskan bagaimana seluruh rancangan yang telah disusun pada bab sebelumnya diimplementasikan. Selain itu bab ini juga menjelaskan tentang hasil – hasil dari proses implementasi, serta perbedaan antara hasil akhir penelitian maupun rancangan awal yang digunakan
5. Bab 5 Hasil dan Pembahasan memaparkan penjelasan-penjelasan mengenai hasil uji coba sistem yang telah berhasil diimplementasikan dengan berbagai skenario yang telah dibuat. Dari hasil uji coba, penulis akan melakukan evaluasi terhadap kinerja sistem dan juga membuat suatu analisis tentang fenomena yang muncul.
6. Bab 6 Penutup berisi penjelasan mengenai kesimpulan yang diperoleh dari pelaksanaan penelitian tugas akhir ini yang berupa suatu rangkuman dari pemecahan masalah yang dilakukan. Selain itu bab ini juga menjelaskan saran-saran yang mungkin dapat diusulkan

sebagai ide yang bagus pada masa selanjutnya, baik berupa pengembangan lebih lanjut sistem yang telah dibuat ini, maupun pembuatan sistem baru yang memiliki persamaan karakteristik dengan sistem yang telah dibuat.

BAB 2 LANDASAN TEORI

Bab landasan teori merupakan bab yang berisi teori – teori yang digunakan sebagai landasan dalam melaksanakan penelitian. Penelitian – penelitian sebelumnya, produk – produk, maupun teknologi yang berkaitan dengan penelitian ini juga akan dipaparkan dalam bab ini. Paparan dalam bab ini bermanfaat untuk memberikan pemahaman yang lebih mendalam kepada penulis maupun pembaca mengenai teori-teori terkait.

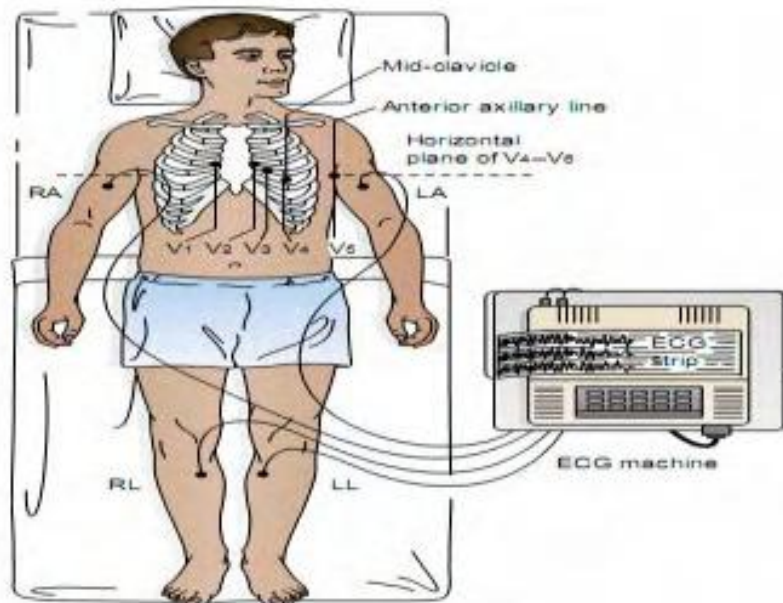
2.1. Elektrokardiogrami

Elektrokardiogrami adalah salah satu cara yang dilakukan untuk mendapatkan, mendiagnosis, dan memantau hasil uji klinis dari pasien guna mengetahui ada tidaknya sindrom penyakit jantung. Secara etimologi elektrokardiogrami dapat diartikan sebagai sebuah pencatatan aktivitas listrik pada jantung. Menurut Daniel Lee Kulick,, elektrokardiogrami adalah sebuah tes bersifat *non-invasive* yang digunakan untuk menentukan kondisi jantung dengan mengukur aktivitas listrik pada jantung [7]. Aktivitas listrik yang diukur tersebut kemudian digambarkan sebagai gelombang atau kurva.

Pada setiap EKG ada banyak bentuk gelombang yang dapat mengindikasikan keadaan jantung yang normal maupun yang tidak normal. Kemiripan lebar interval pola yang terlihat pada elektrokardiogram dan pemahaman tentang ketidak benaran efek *non-kardiak* adalah prasyarat untuk penafsiran yang akurat [8]. Pada umumnya EKG yang lengkap terdiri dari 10 elektroda. Elektroda merupakan bagian yang dipasangkan pada tubuh manusia. Sepuluh elektroda tersebut yaitu RL, RA, LA, LL, V1,V2,V3,V4,V5, dan V6. Sinyal keluaran dari EKG biasanya berupa sinyal sebanyak 12 macam (12 *lead*). Meskipun demikian ada juga jenis EKG yang jumlah *lead*-nya tidak sampai 12, melainkan hanya 1 *lead*, 3 *lead*, atau 6 *lead* saja.

Setiap sinyal yang dikeluarkan oleh EKG 12 *lead* memiliki representasi masing-masing. Setiap *lead* sinyal dari ke-12 sinyal ini merupakan kombinasi dari 3 elektroda yang dipasang pada tubuh manusia. Oleh sebab itu ada pembagian pemasangan elektroda pada tubuh manusia. Enam penunjuk dada (V1 sampai V6) yang menggambarkan jantung pada bidang horizontal. Informasi dari kombinasi keenam elektroda lainnya dikombinasikan untuk menghasilkan enam dahan *lead* (I, II, III, aVR, aVL, dan aVF), yang menggambarkan jantung pada bidang vertikal. Informasi dari 12 *lead* menghasilkan hubungan anatomi berikut: *lead* II, III, dan aVF menggambarkan permukaan bawah jantung; *lead* V1 sampai V4 menggambarkan permukaan

depan; *lead* I, aVL, V5, dan V6 menggambarkan permukaan samping; *lead* V1 dan aVR menggambarkan serambi kanan sampai ke rongga pada bilik kiri. Gambar 2-1 menunjukkan posisi dari enam elektroda dada untuk elektrokardiogram dengan standar 12 *lead* dimana V1: tepi sternum kanan, ruang interkostal keempat; V2: tepi sternum kiri, ruang interkostal keempat; V3: diantara V2 dan V4; V4: baris mid-klavikularis, ruang kelima; V5: baris axial anterior, secara horizontal pada baris dengan V4; V6: baris axial tengah, secara horizontal pada baris dengan V4.



Gambar 2-1. Posisi keenam elektroda dada untuk elektrokardiogrami standar dengan 12 penunjuk (di adaptasi dari Smeltzer & Bare, 2008).

2.2. Penyakit Jantung

Saat ini banyak sekali jenis penyakit jantung yang dikenal, dan hampir semuanya mengancam kehidupan manusia. Penyakit jantung adalah berbagai macam kesalahan fungsi yang dialami jantung. Istilah penyakit jantung (*heart disease*) sering disalah gunakan sebagai sinonim dari *Coronary Artery Disease (CAD)* oleh kebanyakan masyarakat awam maupun oleh petugas kesehatan. *Heart disease* sama maknanya dengan *Cardiac Disease*, akan tetapi tidak sama dengan *Cardiovascular Disease*. *Cardiovascular Disease* mencakup seluruh penyakit jantung dan penyakit pembuluh darah, sedangkan penyakit jantung merupakan gejala abnormal. Beberapa penyakit yang termasuk penyakit jantung adalah sebagai berikut:

1. ***Coronary Artery Disease (CAD)***

CAD timbul karena pengendapan kolesterol pada arteri koroner. CAD juga disebut *Coronary Artherosclerosis*. Hal ini merujuk kepada *atherosleroses* yang berarti keras dan menyempit pada arteri koroner.

2. ***Coronary Hearth Disease (CHD)***

CHD adalah penyakit yang disebabkan karena berkurangnya pasokan darah ke otot jantung dari *atherosclerosis* koroner. Penyakit yang tergolong dalam CHD antara lain:

- ***Heart attack – Myocardial Infraction (MI)***

MI adalah kondisi matinya otot jantung akibat penyumbatan arteri koroner secara tiba-tiba oleh gumpalan darah.

- ***Angina***

Angina adalah kondisi nyeri pada dada yang disebabkan pasokan oksigen ke otot jantung yang tidak memadai. Nyeri dada akibat *angina* biasanya parah dan berbahaya.

- ***Arithmia or Cardilac dysythmia***

Arithmia or Cardilac dysythmia adalah kondisi ritme jantung yang tidak normal.

- ***Heart Failure***

Heart Failure adalah ketidak mampuan jantung untuk memenuhi tuntutan organ, terutama kegagalan memompa darah dengan efisiensi normal.

- ***Congenital Heart Disease***

Congenital Heart Disease adalah kelainan struktur jantung sejak lahir.

- ***Dilated Cardiomyopath***

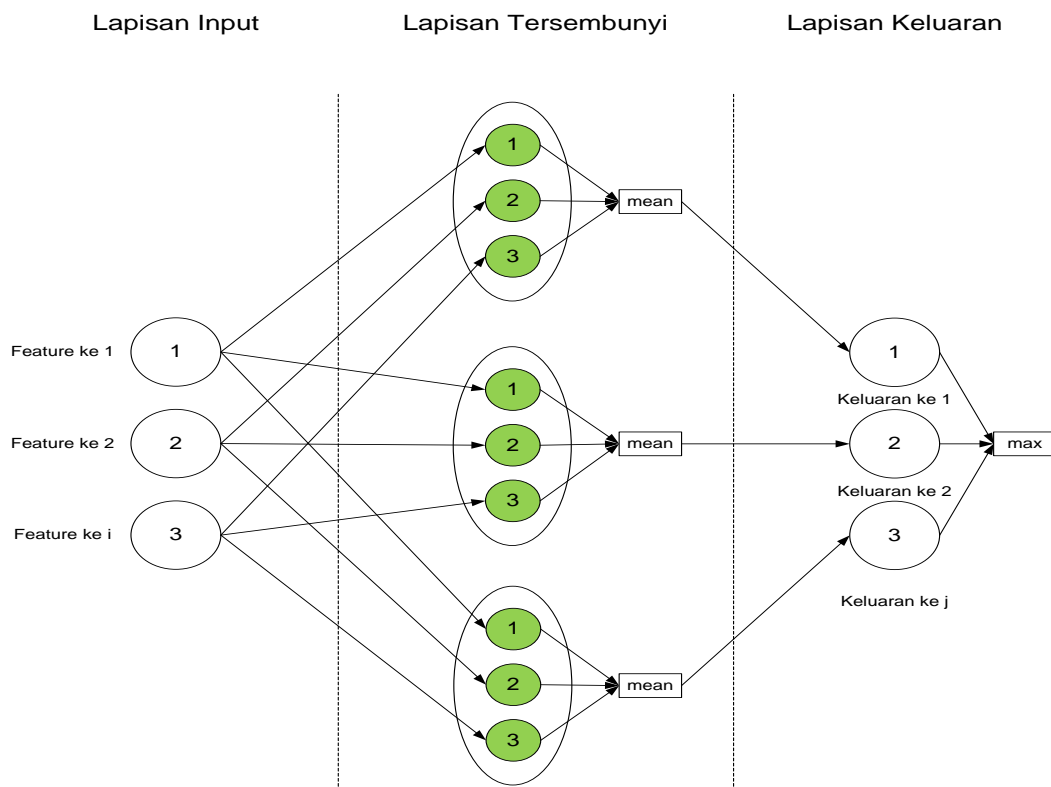
Dilated Cardiomyopath adalah deviasi membesarnya jantung karena otot jantung yang melemah, sehingga tidak bisa memompa darah secara efektif.

- ***Hypertropic Cardiomyopathy (HCM)***

HCM adalah kelainan genetik pada jantung yang ditandai dengan menebalnya dinding bilik kiri.

2.2.1. Fuzzy Learning Vector Quantization (FLVQ)

Fuzzy Learning Vector Quantization (FLVQ) adalah pengembangan dari algoritma jaringan saraf tiruan *Learning Vector Quantization* (LVQ) yang memanfaatkan *fuzzy* pada vektor masukan, proses pelatihan, dan penentuan vektor masukan sehingga dapat menentukan vektor perwakilan pada lapisan *cluster* yang paling merepresentasikan seluruh data masukan. *Learning Vector Quantization* (LVQ) sendiri adalah metode klasifikasi pola dimana terdapat dua *layer* dan setiap unit keluaran merepresentasikan suatu kelas. FLVQ menggunakan prinsip *winner takes all*, dimana penentuan hasil pelatihan ditentukan oleh neuron pemenang. Penentuan vektor perwakilan awal dilakukan dengan cara mengambil salah satu vektor sebagai nilai vektor perwakilan secara acak (inisialisasi acak) atau dengan menentukan vektor masukan pertama sebagai vektor perwakilan (inisialisasi awal).

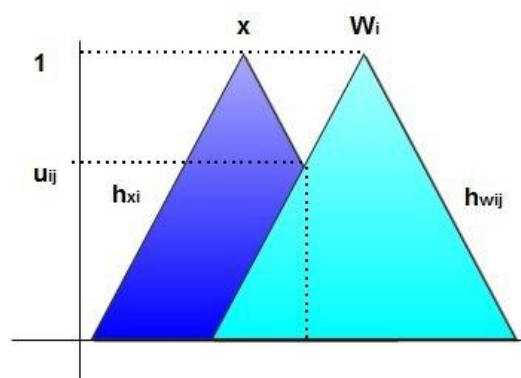


Gambar 2-2. Arsitektur FLVQ.

Arsitektur FLVQ terdiri atas 3 lapisan yaitu lapisan masukan, lapisan keluaran, dan lapisan tersembunyi yang berfungsi menghitung nilai similaritas vektor masukan dan vektor perwakilan. Lapisan masukan terdiri dari neuron-neuron yang menerima pola data yang dimasukkan. Lapisan input ini menggambarkan fitur-fitur data dan banyaknya jumlah neuron pada lapisan input adalah sebanyak jumlah fitur dari data. Lapisan keluaran terdiri dari neuron-neuron

yang merupakan banyaknya kelompok pada data, sedangkan lapisan tersembunyi atau *cluster* terdiri dari neuron-neuron yang banyaknya diperoleh dari perkalian antara banyaknya neuron pada lapisan masukan dengan banyaknya neuron pada lapisan keluaran. Arsitektur FLVQ dapat dilihat pada gambar 2-2

Setelah didapatkan vektor pewakilnya, maka selanjutnya kita mencari nilai similiaritasnya. Nilai similiaritas dapat diperoleh dengan mencari nilai maksimal dari irisan antara vektor masukan dengan vektor pewakil. Nilai similaritas ini nantinya digunakan untuk mencari target kelas keluaran sehingga nanti dapat dikenali data masukannya. Proses pencalian nilai similaritas dilakukan dengan metode segitiga *fuzzy* seperti pada gambar 2-3.



Gambar 2-3. Pencarian similaritas.

Proses pelatihan FLVQ dilakukan dengan melihat nilai similaritasnya di vektor keluaran. Setelah semua vektor pewakil dihitung nilai similaritasnya dengan vektor masukan, maka setiap elemen di lapisan *cluster* dicari nilai similaritas terkecilnya untuk dijadikan masukan pada lapisan keluaran. Penentuan vektor pewakil pemenang dilakukan dengan cara mencari nilai similaritas terbesar yang ada pada tiap-tiap elemen lapisan keluaran tersebut.

2.3. Sistem Operasi Android

Android merupakan sebuah sistem perangkat lunak terintegrasi yang terdiri atas sistem operasi, *middleware*, dan aplikasi utama yang dirancang khusus untuk perangkat *mobile*. Android berdiri sebagai perangkat lunak terbuka dan diproduksi oleh perusahaan Google dan Open Handset Alliance. Saat ini, Android telah terdapat pada jutaan telepon seluler dan perangkat *mobile* lainnya di seluruh dunia. Perkembangan Android meningkat secara pesat sehingga menjadikannya sebagai *platform* yang populer dalam pengembangan aplikasi *mobile*.

Android merupakan lingkungan perangkat lunak untuk perangkat *mobile*. Sistem Android terdiri atas kernel dari sistem operasi Linux, aplikasi *user interface*, aplikasi *end-user*, *code libraries*, *driver* dan *application framework*. Sistem operasi pada Android dibangun dengan bahasa C atau C++, sedangkan untuk pengembangan berbagai aplikasi, tidak terkecuali aplikasi yang secara default sudah ada di Android, digunakan bahasa Java dan dibangun dengan menggunakan *Android Standard Development Kit (SDK)*. Arsitektur sistem operasi Android dapat dilihat pada gambar 2-4.



Gambar 2-4. Diagram arsitektur perangkat lunak sistem operasi Android[13].

Android memiliki arsitektur sistem operasi dengan 4 tingkatan. Pada lapisan paling bawah terdapat kernel Linux yang menjembatani komunikasi antara perangkat keras dengan perangkat lunak. Hubungan tersebut dilakukan oleh driver yang bersesuaian dengan fungsinya, contohnya *display driver*, *camera driver*, *keypad driver*, *flash memory driver*, *WiFi driver*, dan *audio driver*. Pada lapisan kedua, terdapat elemen penting dalam pengembangan aplikasi Android, yaitu pustaka kode yang ditulis dalam bahasa C dan C++. Pada lapisan tersebut terdapat juga Dalvik Virtual Machine, yakni lingkungan tempat dimana aplikasi Android dieksekusi.

Dalvik Virtual Machine (DVM) dan *Java Virtual Machine* (JVM) pada umumnya cukup berbeda. DVM menggunakan arsitektur berbasis register, sedangkan JVM menggunakan arsitektur berbasis stack. Sistem berbasis register tersebut dipilih sebab sangat dengan karakteristik mesin yang sederhana. Tak heran jika aplikasi dalam bahasa Java dapat dieksekusi dalam sistem Android. Selain itu, DVM mengedepankan optimalisasi kode dan minimalisasi ukuran aplikasi. Eksekusi kode diharapkan dapat lebih efisien pada perangkat keras yang berskala kecil.

Lapisan berikutnya adalah *application framework*, berperan sebagai referensi bagi aplikasi menjalankan fungsionalitas fitur yang telah disediakan oleh Android. Pada lapisan paling atas, terdapat aplikasi yang berinteraksi langsung dengan pengguna. Kedua lapisan ini dibangun dengan menggunakan bahasa Java. Pada lapisan teratas tersebut, penulis mengembangkan aplikasi yang digunakan dalam penelitian ini.

2.4. Perangkat Keras Elektrokardiogram *Single Lead*

Seperti yang di jelaskan pada bagian sebelumnya bahwa elektrokardiogram merupakan alat yang berfungsi menangkap sinyal jantung manusia. Prinsip kerja alat ini adalah menguatkan sinyal detak jantung dan mengurangi *noise* yang ada pada sinyal tersebut sehingga kurva sinyal detak jantung dapat divisualisasikan dengan jelas dan mulus. EKG *single lead* ini memiliki 6 pin masukan dan 2 pin keluaran. Tiga pin dari 6 pin masukan adalah pin V+, V-, dan GND (ground). Tiga pin masukan lainnya adalah pin yang terhubung dengan elektroda yang di pasang pada tubuh manusia, yaitu RL, RA, dan LA. Elektroda dipasang pada kedua lengan tangan dan kaki kanan. Elektroda yang dipasang pada kaki kanan bertugas sebagai *grounding*. Gambar elektrokardiogram yang digunakan dalam penelitian ini dapat dilihat pada gambar 2-5. Perangkat ini dikembangkan pada tahun 2011 di Fakultas Ilmu Komputer Universitas Indonesia.

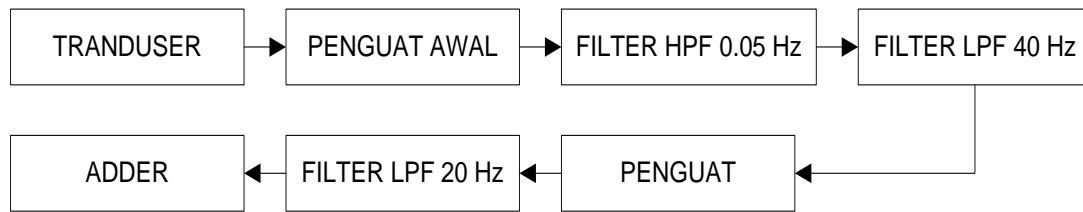


Gambar 2-5.EKG single lead.

Secara umum rangkaian EKG elektronik di atas memiliki spesifikasi sebagai berikut :

- Input sinyal didapatkan dari transduser EKG berbahan metal dari 3 M.
- Penguat awal menggunakan IC tipe INA 118 dengan penguatan 10 kali.
- Menggunakan *High Pass Filter* dengan frekuensi 0,05 Hz.
- Menggunakan *Low Pass Filter* dengan frekuensi 40 Hz
- Menggunakan penguat tambahan dengan penguatan sebesar 100 kali.
- Menggunakan *low pass filter* tambahan untuk meredam *noise* dengan frekuensi sebesar 20 Hz.
- Menggunakan rangkaian *adder* untuk menggeser amplitudo tegangan.

Setiap bagian dari EKG di atas memiliki peran masing-masing. Selain itu spesifikasi yang dimiliki bagian-bagian tersebut juga ditentukan dan dipertimbangkan dengan perhitungan – perhitungan matematis, berdasarkan karakteristik masing – masing komponen. Selanjutnya diperlukan penyesuaian ukuran antara satu komponen elektronika dan komponen lainnya agar alat dapat bekerja dengan baik. Secara umum spesifikasi dan diagram alur rangkaian EKG di atas dibentuk dalam modul-modul yang ditunjukkan pada gambar 2-6.



Gambar 2-6. Modul – modul penyusun rangkaian elektrokardiogram (EKG).

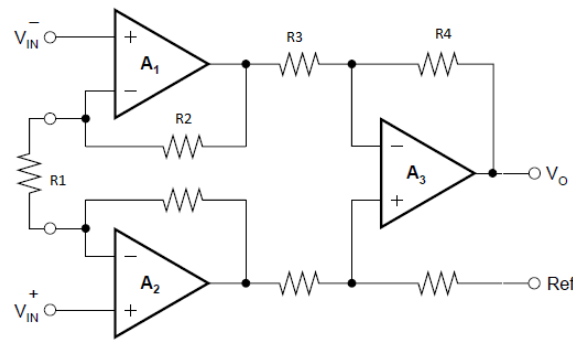
Penjelasan detail mengenai masing-masing bagian dari lektrokardiogram tersebut akan dipaparkan selanjutnya.

2.4.1. Transduser

Transduser merupakan suatu perangkat yang digunakan untuk mengubah sinyal detak jantung manusia dari bentuk alamiah menjadi bentuk sinyal listrik. Pada EKG tersebut *transduser elektroda* (selanjutnya disebut sebagai elektroda) digunakan untuk mengubah ion-ion pada permukaan tubuh pasien menjadi sinyal elektrik sehingga dapat diproses dalam rangkaian EKG elektrik. *Transduser* yang digunakan adalah tipe *transduser* yang mempunyai kontak langsung dengan kulit. *Transduser* terbuat dari bahan metal seperti *silver* atau *stainless steel*. Pada proses pengambilan sinyal ini menggunakan *transduser* dari 3M. Jika sinyal diambil dari alat pasien simulator, maka tidak diperlukan transduser, melainkan sinyal dapat diambil menggunakan konektor elektrik seperti penjepit buaya.

2.4.2. Penguat awal

Penguatan awal digunakan untuk menguatkan sinyal awal EKG yang diambil oleh *elektroda transduser*. Penguatan di tahap awal ini bertujuan agar sinyal detak jantung dapat dibaca dengan lebih mudah. Pada penguatan awal ini diperlukan suatu penguat instrumentasi. Ilustrasi gambar rangkaian penguat instrumentasi dideskripsikan seperti yang ditunjukkan pada gambar 2-7. Untuk mendapatkan fungsi kerja yang optimal dari penguat instrumentasi dan untuk memperkecil rangkaian, maka digunakan suatu penguat instrumentasi yang telah terintegrasi dalam satu IC. IC dengan spesifikasi ini biasa disebut dengan *instrumentation amplifier*. Penelitian ini menggunakan *instrumentation amplifier* tipe INA118. Penguatan dari INA118 dapat diatur dengan menggunakan satu resistor eksternal.

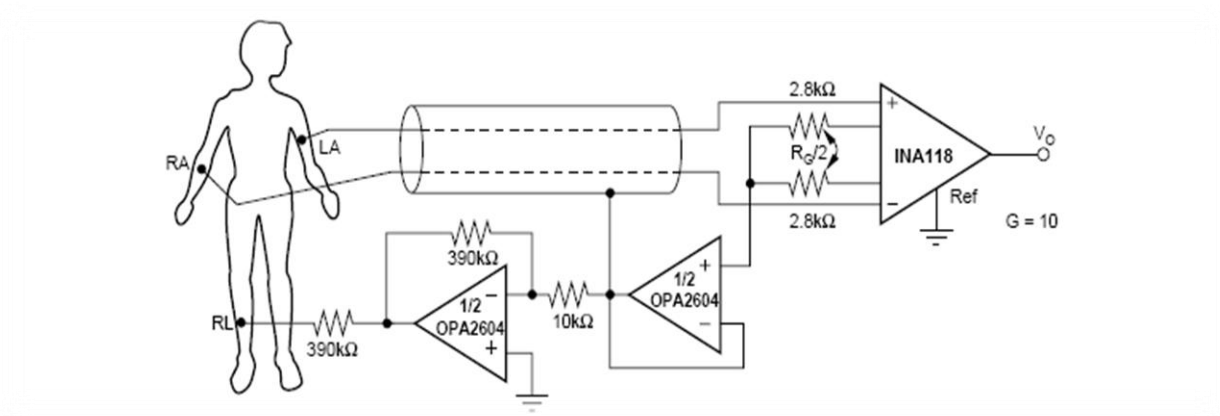


Gambar 2-7. Penguat instrumentasi.

Untuk menghindari efek saturasi dimana simpangan sinyal menjadi rusak dari penguat instrumentasi INA118 maka penguat awal EKG ini menggunakan penguatan yang relatif kecil yaitu 10 kali. Untuk mendapatkan nilai penguatan sebesar 10 kali, maka nilai R_G yang digunakan adalah $5.6K\Omega$. Nilai tersebut dihitung dengan persamaan berikut :

$$G = 1 + \frac{50K\Omega}{R_G}$$

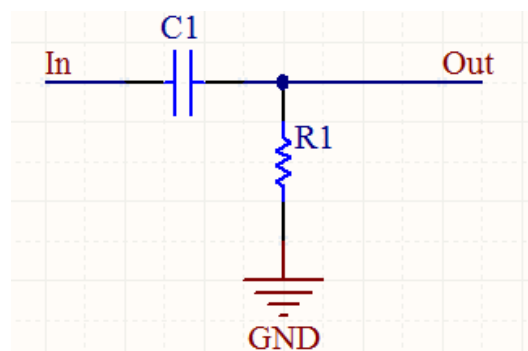
Dalam pembuatan penguat awal digunakan suatu rangkaian *right-leg-driven-loop* yang berfungsi untuk mengurangi *noise* yang berasal dari tubuh. Selain itu jenis rangkaian ini juga berguna sebagai sistem pengamanan pasien. Rangkaian *right-leg-driven-loop* yang digunakan diperoleh dari *datasheet* INA118 yang mana dapat dilihat pada gambar 2-8. Rangkaian *right-leg-driven-loop* ini digunakan untuk *inverted version common mode interference* pada kaki kanan pasien, yang berfungsi untuk menghilangkan interferensi.



Gambar 2-8. Rangkain Right Leg Driver Loop untuk meminimalkan noise.

2.4.3. High Pass Filter (HPF)

High Pass Filter (HPF) pada rangkaian EKG digunakan untuk mengurangi *noise* dengan frekuensi rendah yang dihasilkan oleh tubuh. Konsep dari HPF adalah menghilangkan atau meniadakan sinyal dibawah frekuensi tertentu (frekuensi *cut off*). Tubuh manusia juga mengeluarkan *noise* pada frekuensi antara 0 sampai 1 Hz. Selain itu juga terdapat *noise* dari catuan DC pada frekuensi rendah. HPF akan melewatkan frekuensi di atas frekuensi *cut-off*-nya. Dalam alat ini frekuensi *cut-off* yang digunakan adalah sebesar 0,05Hz. Nilai *cut-off* diambil 0,05 Hz karena frekuensi *monitoring* EKG berkisar antara 0,05 – 20 Hz. Dengan demikian sinyal – sinyal di bawah frekuensi batas minimal EKG akan dihilangkan oleh HPF. Rangkaian HPF yang digunakan dapat dilihat pada gambar 2-9.



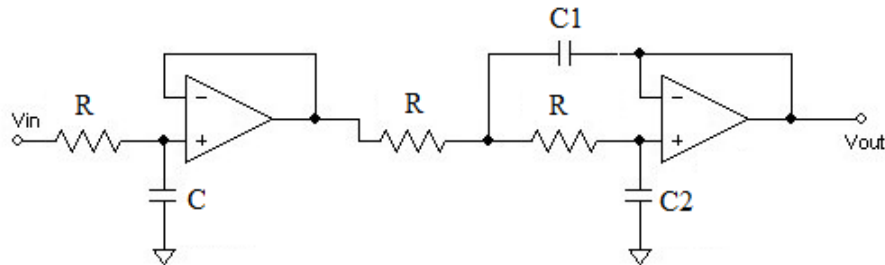
Gambar 2-9. Rangkaian HPF.

Untuk untuk mendapatkan frekuensi *cut-off* sebesar 0,05 Hz dapat digunakan rumus $F_c = \frac{1}{2\pi RC}$. Untuk mendapatkan nilai *cut-off* yang diinginkan dapat dilakukan dengan cara mengubah nilai R dan nilai C. Pada pembuatan alat ini digunakan nilai R=33k Ω dan nilai C=100 μ F. Dengan demikian, berdasarkan perhitungan yang diperoleh dengan rumus frekuensi *cut off*, diperoleh nilai frekuensi yang dihasilkan dari nilai R dan C tersebut adalah 0.048Hz.

2.4.4. Low Pass Filter (LPF)

Setelah mendapatkan frekuensi sinyal yang bersih dari *noise* berfrekuensi rendah, rangkaian *Low Pass Filter* (LPF) perlu ditambahkan pada rangkaian. LPF berfungsi untuk menghilangkan *noise* pada sinyal EKG yang frekuensinya besar. Prinsip kerja dari LPF adalah melewatkan sinyal dengan frekuensi di bawah frekuensi *cut off* nya. Dengan demikian jika ada sinyal *noise* dengan frekuensi di atas batas atas frekuensi sinyal EKG akan

dihilangkan. Pada pengembangan prototipe ini digunakan LPF aktif dengan orde , yaitu jenis *filter butte rworth* dan topologi rangkaian *Sallen Key*. Rangkaian *low pass filter* yang digunakan dapat dilihat pada gambar 2-10.

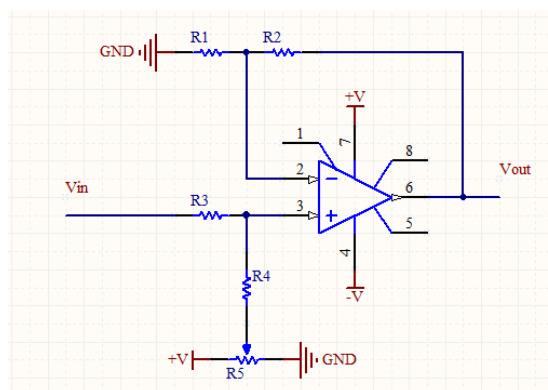


Gambar 2-10. Low Pass Filter orde 3 butterworth.

Untuk mempermudah perhitungan dalam mencari nilai-nilai komponen yang diinginkan, maka nilai R dibuat sama yaitu 50 k Ω . Sehingga nilai kapasitor yang dibutuhkan dapat dihitung melalui persamaan $C = \frac{1}{2\pi fR}$. Dengan nilai R dan *frekuensi cut off* yang diketahui, maka nilai C dalam *low pas filter* ini adalah 159,2 nF

2.4.5. Adder

Rangkaian terakhir dari deretan modul-modul penyusun EKG adalah rangkaian *adder*. Rangkaian ini berfungsi untuk menghilangkan nilai simpangan negatif pada sinyal EKG yang merupakan dikeluarkan rangkaian rangkaian LPF kedua. Rangkaian *adder* ini dipakai untuk menggeser amplitudo ke sisi positif sehingga keseluruhan sinyal EKG memiliki simpangan (voltase) lebih dari atau sama dengan 0 volt. Gambar skematik rangkaian *adder* dapat dilihat pada gambar 2-11.



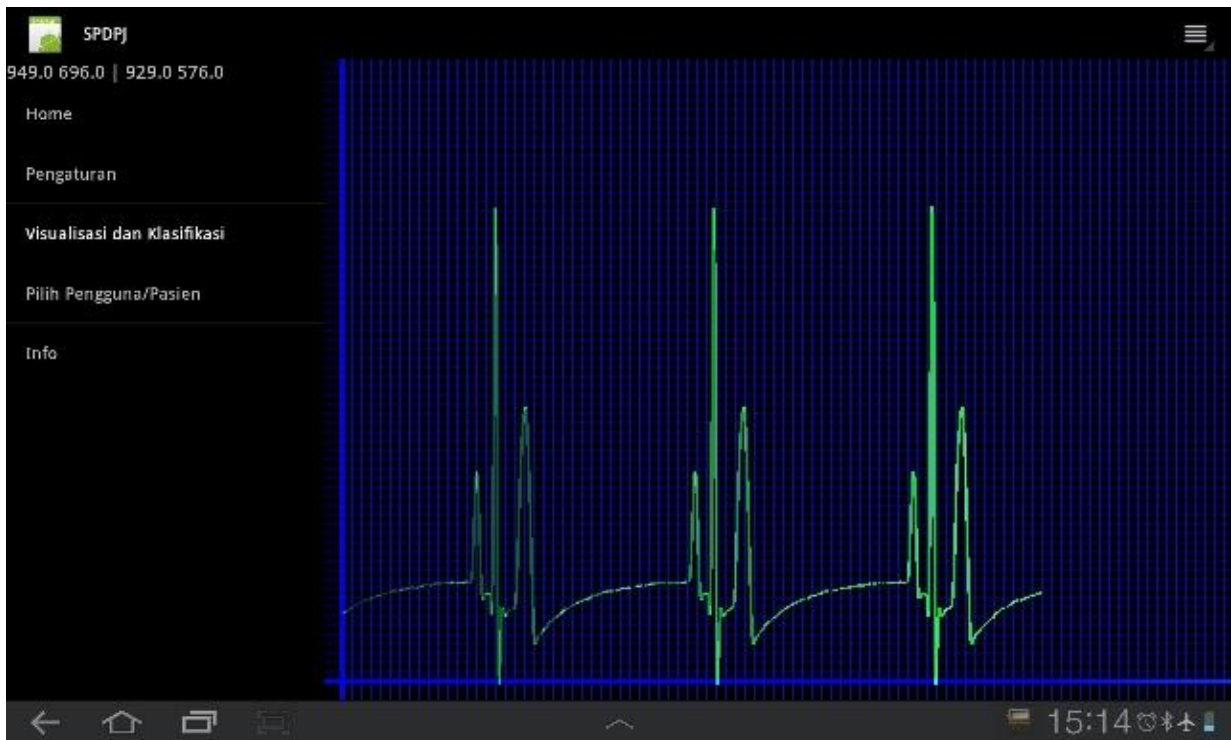
Gambar 2-11. RangkaianAdder.

2.5. Aplikasi Sistem Pendeteksian Dini Penyakit Jantung (E-Cardio)

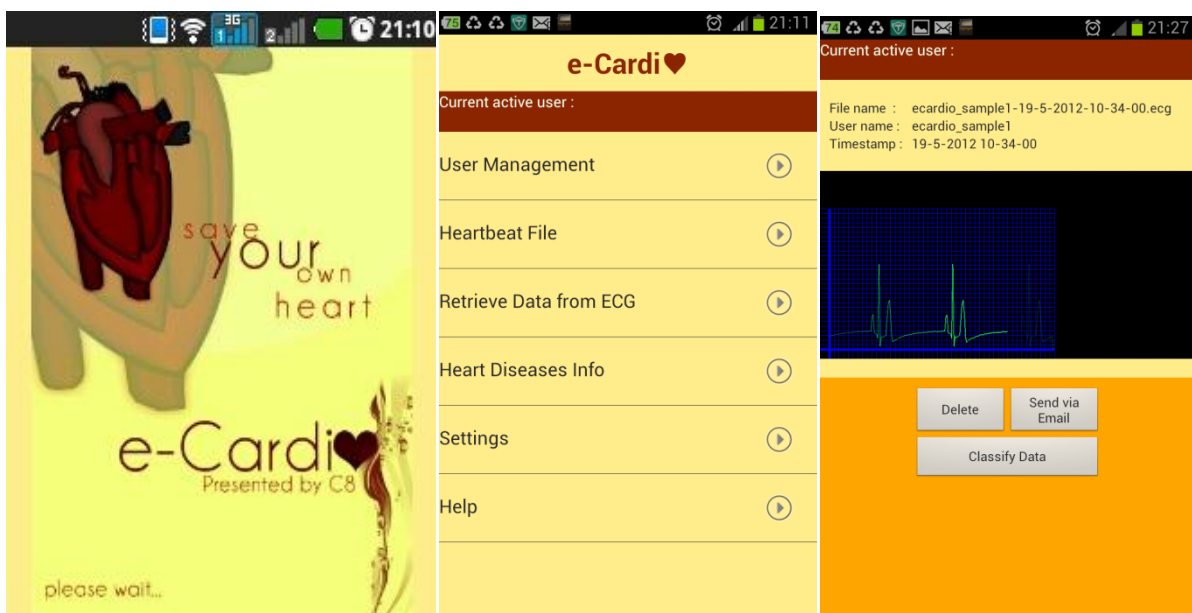
2.5.1. E-Cardio versi *Smartphone* Android

E-Cardio merupakan salah satu aplikasi berbasis *smartphone* Android yang dikembangkan di Fakultas Ilmu Komputer Universitas Indonesia. Aplikasi ini merupakan aplikasi yang digunakan untuk memvisualisasikan data detak jantung yang diambil dari sensor *hardware* maupun dari file data EKG yang telah disimpan. Aplikasi ini dapat berkomunikasi dengan sensor EKG (*hardware*) melalui koneksi jaringan *bluetooth*. Melalui jaringan inilah data sinyal detak jantung ditransmisikan oleh *hardware* ke *smartphone* Android. Pengiriman data tersebut menggunakan frekuensi *baudrate* sebesar 9600 bit per detik. Selanjutnya data detak jantung yang telah diterima oleh *smartphone* Android tadi dapat diklasifikasikan apakah ada gejala penyakit jantung atau tidak. Data detak jantung tadi juga dapat disimpan untuk dilihat pada waktu yang akan datang. Tampilan gambar aplikasi E-Cardio dapat dilihat pada gambar 2-12.

Selain memberikan fitur utama yaitu visualisasi dan pendeteksian dini penyakit jantung, aplikasi E-Cardio juga dilengkapi dengan fitur-fitur tambahan seperti pengaturan pengguna dan informasi penyakit jantung. Dengan demikian selain dapat memberikan layanan untuk mendeteksi penyakit jantung, aplikasi ini juga memberikan fungsi edukasi kepada pengguna. Selain itu E-Cardio versi 2.0 juga menyediakan fitur untuk pengaturan koneksi dengan *hardware* melalui jaringan *bluetooth*.



(a)

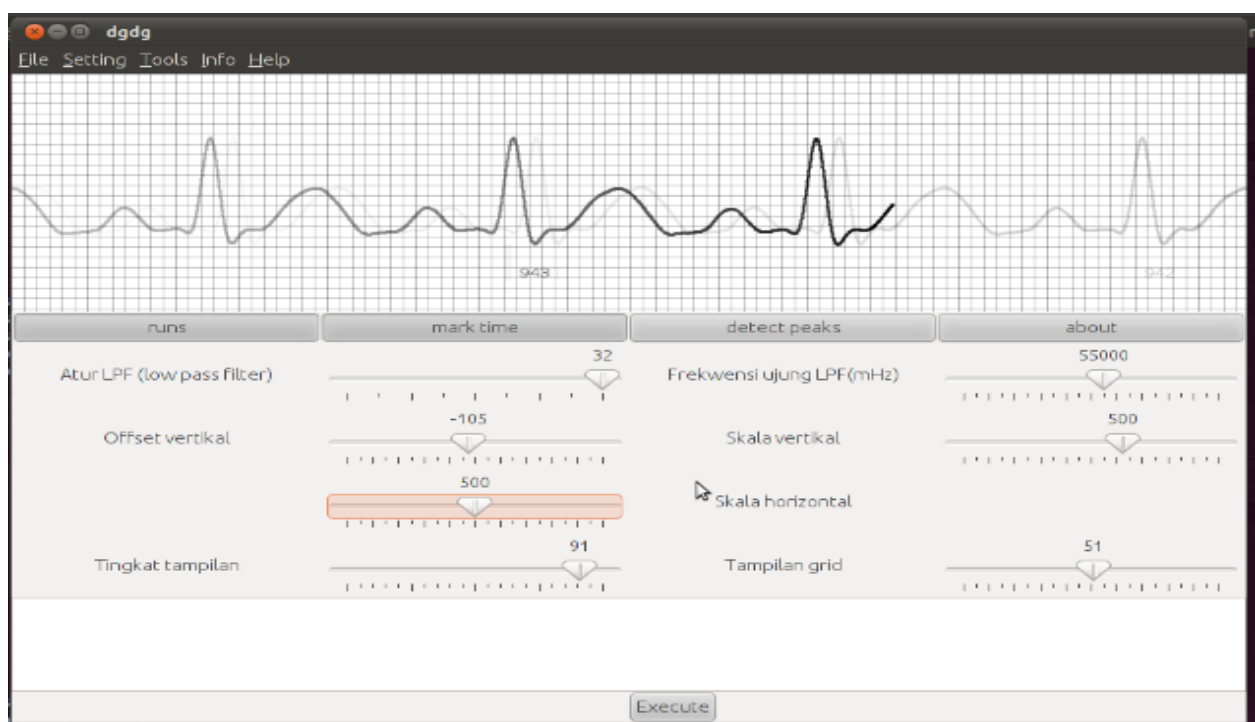


(b)

Gambar 2-12. Tampilan aplikasi E-Cardio di smartphone Android (a). Versi 1.0, (b). Versi 2.0.

2.5.2. E-Cardio versi *Personal Computer (PC) Desktop*

Selain dikembangkan untuk *platform* sistem operasi Android, aplikasi ini juga dikembangkan untuk perangkat *personal computer (PC) Desktop*. Aplikasi yang berjalan di PC dibuat dengan platform Java. Java menjadi pilihan dalam pengembangan karena tujuan dari pembuatan aplikasi versi PC desktop adalah agar aplikasi dapat berjalan di semua platform sistem operasi komputer desktop. Oleh sebab itu, Java menjadi *tools* yang solutif untuk mencapai tujuan yang diinginkan tersebut. Selain itu dengan memilih Java untuk pengembangan aplikasi *desktop* juga diperoleh keuntungan, yaitu beberapa kelas yang sudah diimplementasikan pada aplikasi Android dapat digunakan, tanpa memodifikasi. Hal ini tentu saja mempercepat proses pengembangan aplikasi. Secara keseluruhan fungsi dan fitur-fitur aplikasi pada versi PC desktop dan versi *smartphone* Android kurang lebih sama. Tampilan aplikasi E-Cardio versi PC Desktop dapat dilihat pada gambar 2-13.



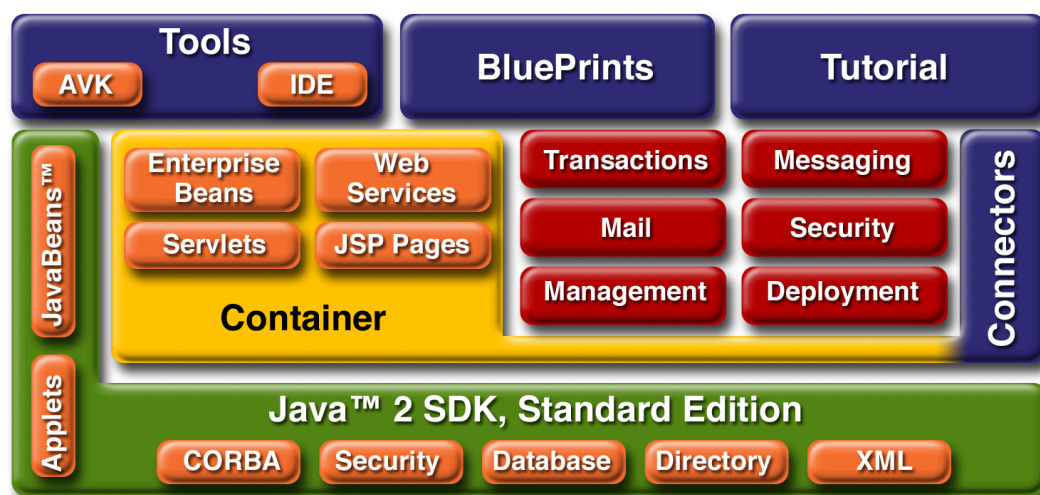
Gambar 2-13. Tampilan aplikasi E-Cardio di PC desktop.

Meskipun sudah dapat memberikan layanan dan fungsi di atas, aplikasi ini masih memiliki berbagai macam kelemahan. Pertama, jaringan komunikasi dari aplikasi ini sering terputus saat digunakan. Saat dicoba disambungkan kembali terkadang koneksi gagal dilakukan. Dalam keadaan yang seperti ini pengguna harus melakukan *unpairing* lalu melakukan *pairing* kembali dengan *bluetooth* yang di-embed pada sensor EKG. Kelemahan kedua dan

yang paling penting adalah aplikasi ini tidak menyediakan fitur untuk mengirimkan data dari pasien ke dokter. Dengan demikian sementara sementara ini, dokter tidak bisa memberikan verifikasi data pasien tanpa harus bertemu dengan pasien secara langsung (jarak jauh). Selain itu aplikasi ini juga belum didukung dengan *server* basis data secara *online* yang dapat digunakan untuk menyimpan data secara *online*. Oleh sebab itu penting dilakukannya penambahan *server* dan modul untuk pengiriman data secara *online* agar dokter dapat memverifikasi data detak jantung pasien dari jarak jauh.

2.5.3. Teknologi Java *Enterprise Edition* (Java EE)

Java Enterprise Edition (Java EE) merupakan teknologi yang dibuat oleh perusahaan Oracle untuk keperluan pengembangan perangkat lunak *enterprise*. Jenis perangkat lunak yang dimaksud dapat berupa aplikasi berbasis *web*, *database*, dan sistem yang besar. Platform Java EE memberikan kemudahan dengan berbagai macam dukungan yang diberikan. Tools yang paling utama dari Java EE yang digunakan untuk mengembangkan aplikasi berbasis *server* antara lain adalah JSP Pages, Servlet, Enterprise Beans, dan Web Services. Selain menyediakan *application programming interface* (API) untuk membangun perangkat lunak, Java EE juga memberikan *tools - tools* lain seperti IDE, konektor untuk berbagai transaksi seperti email, *security*, messaging, dan juga kompatibilitas dengan platform java yang lain. Arsitektur platform Java EE dapat dilihat pada gambar 2-14.



Gambar 2-14. Arsitektur Java EE.

Seperti yang sudah disebutkan sebelumnya bahwa dukungan dari Java EE yang memegang peranan penting dalam pelaksanaan penelitian ini adalah Servlet, JSP Pages, *Web Service*, dan Enterprise Beans. Servlet merupakan *tools* yang dapat digunakan sebagai manajemen *unified resource locator* (URL) pada aplikasi berbasis *web* yang berjalan di *server*. Dengan servlet, pengembang aplikasi dapat mengimplementasikan metode *model view controller* (MVC) dengan lebih mudah, karena setiap menu atau sub menu dapat ditangani oleh satu servlet. Contoh kode sumber suatu servlet dapat dilihat pada gambar 2-15. Pada servlet terdapat argument berupa request dan response. Dari kedua argumen tersebut bisa diperoleh parameter – parameter yang diberikan pada URL.

```

/*
 * @author anwar
 */
@WebServlet(name = "RegisterPatient", urlPatterns = {"/RegisterPatient"})
public class RegisterPatient extends HttpServlet {

    /**
     * Processes requests for both HTTP
     * <code>GET</code> and
     * <code>POST</code> methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {...}
}

```

Gambar 2-15. Contoh kode sumber servlet.

JSP Pages merupakan *tools* untuk mengelola halaman *web* aplikasi yang dibuat. JSP juga mendukung tag-tag html yang biasa digunakan dalam membangun suatu halaman *web*. Pada kode program halaman *web* JSP dapat juga disipkan kode java standar, yaitu dengan cara diapit dengan tag “<%” dan “%>” (tanda “ ” hanya sebagai penanda). JSP merupakan *tools* yang menggabungkan fitur halaman *web* dengan html dan juga teknologi Java standar. Contoh kode program pada halaman *web* JSP dapat dilihat pada gambar 2-16.

```

<%--
Document    : index
Created on  : Jan 5, 2013, 11:04:56 PM
Author      : anwar
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

```

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>JSP Page</title>
</head>
<body>
  <h1>Hello World!</h1>
</body>
</html>
```

Gambar 2-16. Contoh kode program halaman *web* dengan JSP.

Enterprise java beans (EJB) merupakan tools yang digunakan untuk menggabungkan kelas-kelas yang ditulis dengan java standar ke dalam *package* dimana aplikasi berbasis *web* dibuat. Manfaat dari tools ini adalah kita bisa tetap menerapkan paradigma *object oriented programming* (OOP) dalam pengembangan aplikasi. Selain itu dengan adanya EJB, dapat dilakukan *object relational mapping* dari *database* ke kelas java. Dengan demikian proses penegelolaan data pada *database* dapat dilakukan dengan lebih mudah. Untuk keperluan *mapping* ke *database* diperlukan API yang lazim disebut dengan Java Persistence API (JPA).

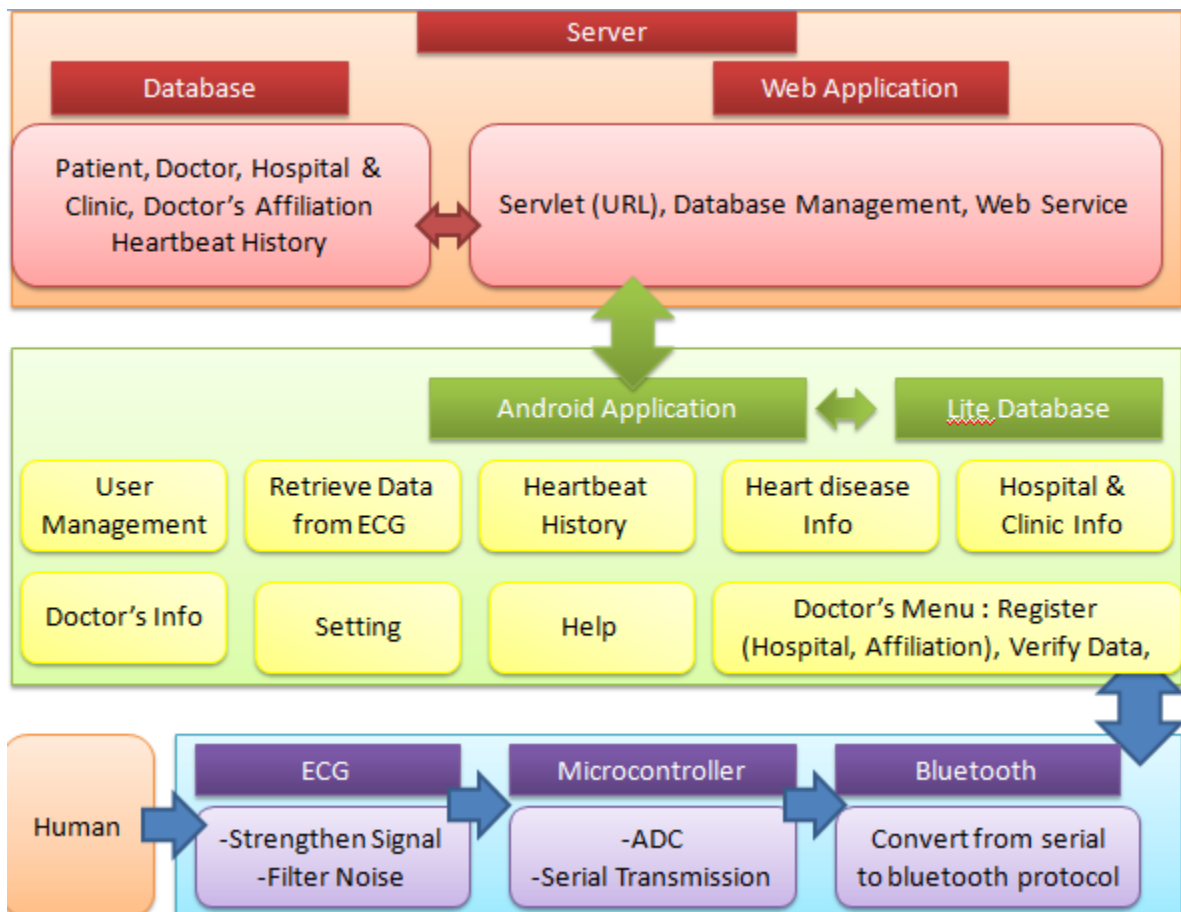
Selain *tools* yang dijelaskan pada paragraf sebelumnya, ada *tools* yang penting pada Java EE untuk mengembangkan aplikasi berbasis *web*, yaitu *tools* untuk membuat *web service*. Seperti yang kita ketahui bahwa ada dua macam *web service* yaitu SOAP *web service* dan Restful *web service*. Java EE menyediakan *tools* yang dapat digunakan untuk membuat kedua macam *web service* tersebut. Dengan demikian pengembang perangkat lunak cukup fokus pada implementasi pembuatan fungsi pada *web service*.

BAB 3 RANCANGAN PENELITIAN

Bab rancangan penelitian merupakan bab yang menjelaskan mengenai rancangan penelitian yang akan dilakukan. Penjelasan tersebut mencakup rancangan arsitektur sistem, rancangan modifikasi aplikasi E-Cardio, Rancangan implementasi *database*, rancangan implementasi *server*, rancangan interaksi komponen dalam sistem, data detak jantung yang digunakan dalam eksperimen.

3.1. Rancangan Arsitektur Sistem

Arsitektur prototipe sistem terintegrasi pendeteksi dini dan *monitoring* penyakit jantung terdiri dari tiga komponen besar seperti yang terlihat pada gambar 3-1.



Gambar 3-1. Arsitektur sistem pendeteksian dini dan *monitoring* penyakit jantung.

Komponen pertama pada *prototype* sistem adalah *hardware*. *Hardware* pada *prototype* sistem ini merupakan sensor berbasis EKG. Komponen kedua adalah *smartphone* Android. Komponen ketiga dalam *prototype* sistem tersebut adalah *server*.

Hardware yang berupa sensor EKG ini bertugas untuk mengambil data detak jantung manusia lalu mengubahnya menjadi data digital sedemikian hingga dapat diterima oleh komponen kedua. Sensor ini terdiri dari tiga komponen, yaitu EKG, mikrokontroller, dan *bluetooth device*. Sistem kerja *hardware* ini dapat dilihat pada gambar 3-2.



Gambar 3-2. Sistem kerja komponen *hardware* pada *prototype system*.

Tahap masukan merupakan tahap pengambilan data detak jantung dari tubuh manusia. Sebenarnya tanpa menggunakan elektrokardiogram data detak jantung ditampilkan pada alat osiloskop. Akan tetapi pola dari gelombang sinyal elektrik yang dihasilkan terlalu lemah sehingga susah dibaca. Selain itu gelombang ini juga mengandung banyak *noise*, sehingga tidak dapat digunakan keperluan analisis kondisi jantung seseorang. Pada dasarnya elektrokardiogram bertugas melakukan penguatan pada sinyal detak jantung manusia sekaligus mereduksi *noise*. Dengan demikian pola gelombang yang dihasilkan mudah dibaca. Dengan menganalisis pola gelombang tersebut dapat disimpulkan kondisi jantung seseorang apakah ada gejala penyakit jantung atau tidak.

Seperti yang dijelaskan sebelumnya bahwa elektrokardiogram yang akan digunakan pada prototipe ini adalah elektrokardiogram *single lead* dimana data output dari sinyal detak jantung dari EKG jenis ini hanya ada satu jenis sinyal. Selain itu data yang dihasilkan masih dalam bentuk data analog. Data analog merupakan data yang masih dalam bentuk besaran voltase listrik. Bentuk fisik dari elektrokardiogram ini adalah rangkaian elektronika yang terdiri dari kombinasi rangkaian penguat (*amplifier*), rangkaian *high pass filter*, *low pass filter* dan rangkaian *adder*, dilengkapi dengan elektroda yang akan dipasang pada anggota badan. Untuk mengamati sinyal detak jantung yang dihasilkan oleh alat ini dapat dilakukan dengan menggunakan osiloskop.

Data sinyal detak jantung yang dikeluarkan oleh elektrokardiogram masih dalam bentuk data analog. Oleh sebab itu sebelum dikirim ke *smartphone*, perlu diubah terlebih dulu menjadi data digital. Pengubahan atau konversi data analog menjadi digital tersebut dilakukan oleh perangkat atau modul yang disebut dengan *Analog to Digital Converter* (ADC). Komponen elektronika yang bertugas untuk melakukan tugas ADC dalam *prototype* ini adalah mikrokontroler. Konversi data analog ke data digital oleh perangkat ADC merupakan konsep yang sederhana. Data analog adalah data yang masih dalam bentuk voltase. Setelah melalui konversi oleh ADC data tersebut akan diinterpretasikan dengan nilai digital tertentu. Untuk keperluan interpretasi ini dibutuhkan voltase referensi VCC dan ground (GND). Interval nilai yang digunakan bergantung pada perangkat yang digunakan. Nilai 0 diberikan jika input memiliki voltase yang sama dengan *ground*, sedangkan nilai maksimal diberikan jika input memiliki voltase yang sama dengan VCC. Ada beberapa hal yang harus diperhatikan dalam menggunakan modul ADC. Pertama, harus diperhatikan pemberian tegangan referensi. Tegangan yang diberikan untuk referensi VCC pada ADC dan tegangan VCC untuk *chip* yang digunakan biasanya tidak boleh berbeda jauh (ada batas selisih maksimal yang diizinkan). Hal ini diperlukan untuk menghilangkan *noise* dan menjaga konsistensi dalam proses konversi. Pada suatu perangkat, biasanya ada beberapa pin ADC yang memiliki akurasi berbeda. Misalnya pada mikrokontroler AVR rumpun atmega, ada yang memiliki akurasi 8 bit dan ada yang memiliki akurasi sampai 10 bit. Berikutnya yang harus diperhatikan adalah pemakaian frekuensi *clock*. Pada umumnya semakin tinggi *clock* yang digunakan, maka konversi semakin bagus dan konsisten.

Setelah sinyal detak jantung yang dikonversi menjadi data digital akan dikirimkan data digital ke *smartphone* untuk proses selanjutnya. Pengiriman data dari perangkat yang memiliki modul ADC ke perangkat *smartphone* dapat dilakukan dengan berbagai cara. Pertama, pengiriman data dapat dilakukan menggunakan kabel (*wire*). Hal ini tergantung dari antarmuka (*interface*) perangkat yang melakukan konversi tersebut. Jika perangkat yang melakukan konversi memiliki antarmuka USB, maka data digital dapat dilakukan dengan mode USB to USB. Akan tetapi kebanyakan *smartphone* tidak mendukung USB *host*. Dengan demikian pengiriman data menggunakan metode *wire* tidak dapat dilakukan ke *smartphone* Android. Alternatif lain yang dapat digunakan untuk mengirimkan data digital ke *smartphone* adalah dengan cara nirkabel (*wireless*). Untuk keperluan ini kita dapat memanfaatkan modem *bluetooth* yang bisa ditambahkan ke perangkat yang melakukan

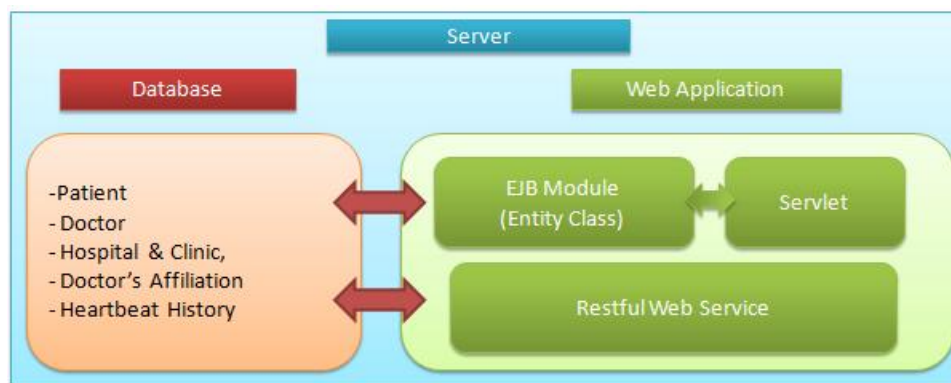
konversi. Sehingga perangkat tersebut dapat mengirimkan data via *bluetooth* ke komponen kedua yang sudah dilengkapi dengan modul *bluetooth*.

Komponen kedua dalam *prototype* sistem pendeteksian dini dan *monitoring* penyakit jantung ini adalah *smartphone* Android. Komponen kedua ini memiliki beberapa fungsi. Urutan kerja fungsi-fungsi komponen *smartphone* Android ini dapat dilihat pada Fungsi pertama adalah untuk menerima data (*raw*) data detak jantung yang dikirimkan oleh komponen pertama (*hardware*). Pada tahap *raw* data ini, juga dilakukan filter untuk menghilangkan *noise* yang ditimbulkan oleh perangkat keras. Perangkat keras sensor EKG juga dapat memberikan efek *noise* pada gelombang sinyal detak jantung. Dengan demikian ketika data sampai di *smartphone* bentuk gelombangnya tidak mulus (*smooth*). Oleh sebab itu dilakukan *filter* lagi untuk membuat gelombang detak jantung *smooth*. Fungsi kedua dari komponen *smartphone* ini adalah pemrosesan dan pengelolaan data. Data detak jantung yang sudah diterima tadi diproses untuk klasifikasi guna mengetahui ada tidaknya gejala penyakit jantung pada pengguna. *History* data detak jantung tersebut juga akan disimpan dalam *database* *smartphone* Android dalam bentuk *lite database*. Pemilihan *database* diambil karena *database* memiliki banyak kelebihan dibandingkan dengan penyimpanan data pada *file*. Dengan menggunakan *database* manajemen pengelolaan data lebih mudah digunakan. Selain itu, jika suatu saat sistem dikembangkan lagi, maka penggunaan *database* sangat mendukung dan tidak perlu dimodifikasi. Selain data detak jantung, data tentang pasien, dokter, serta data rumah sakit dan klinik juga disimpan dalam *lite database* ini. Fungsi ketiga dari komponen *smartphone* Android ini adalah pengiriman data dan pengunduhan data ke dan dari *server*. Data yang dikirimkan tersebut adalah data *history* detak jantung, data pasien, data dokter, serta data rumah sakit dan klinik. Sistem kerja komponen kedua dapat dilihat pada gambar 3-3.



Gambar 3-3. Sistem kerja komponen *smartphone android* pada *prototype* sistem.

Komponen ketiga dalam *prototype* sistem pendeteksian dini dan *monitoring* penyakit jantung ini adalah *server*. *Server* berguna untuk menyimpan data pada *database* secara *online*. Dengan adanya *server* ini pasien (pengguna) dapat mengirimkan *data* detak jantungnya untuk diverifikasi oleh dokter. Selanjutnya, dokter bisa melakukan verifikasi pada detak jantung pasien yang dikirimkan ke *server*. Pada *server* terdapat dua komponen, yaitu *database*, dan aplikasi *web*. *Database* digunakan untuk menyimpan data pasien, dokter, *history* data detak jantung, data rumah sakit dan klinik, dan data afiliasi dokter. Aplikasi *web* digunakan untuk menerima permintaan (*request*) dari pengguna aplikasi E-Cardio. Pada aplikasi *web* sendiri ada beberapa komponen. Komponen pertama adalah *servlet*. *Servlet* digunakan untuk menerima *request* dalam bentuk URL tertentu. Komponen kedua dalam aplikasi *web* adalah modul EJB. Modul EJB digunakan untuk keperluan *object relational mapping* ke *database*. Dengan demikian data pada *database* dapat direpresentasikan dengan objek dari suatu kelas. Komponen ketiga adalah *Restful Web Service*. *Web service* digunakan untuk maintenance *database* dalam jangka panjang. Dengan adanya *restful web service* ini *database* dapat diakses oleh *server* lain. Diagram komponen dan fungsional *server* dapat dilihat pada gambar 3-4.



Gambar 3-4. Diagram fungsional *server* pada *prototype* sistem.

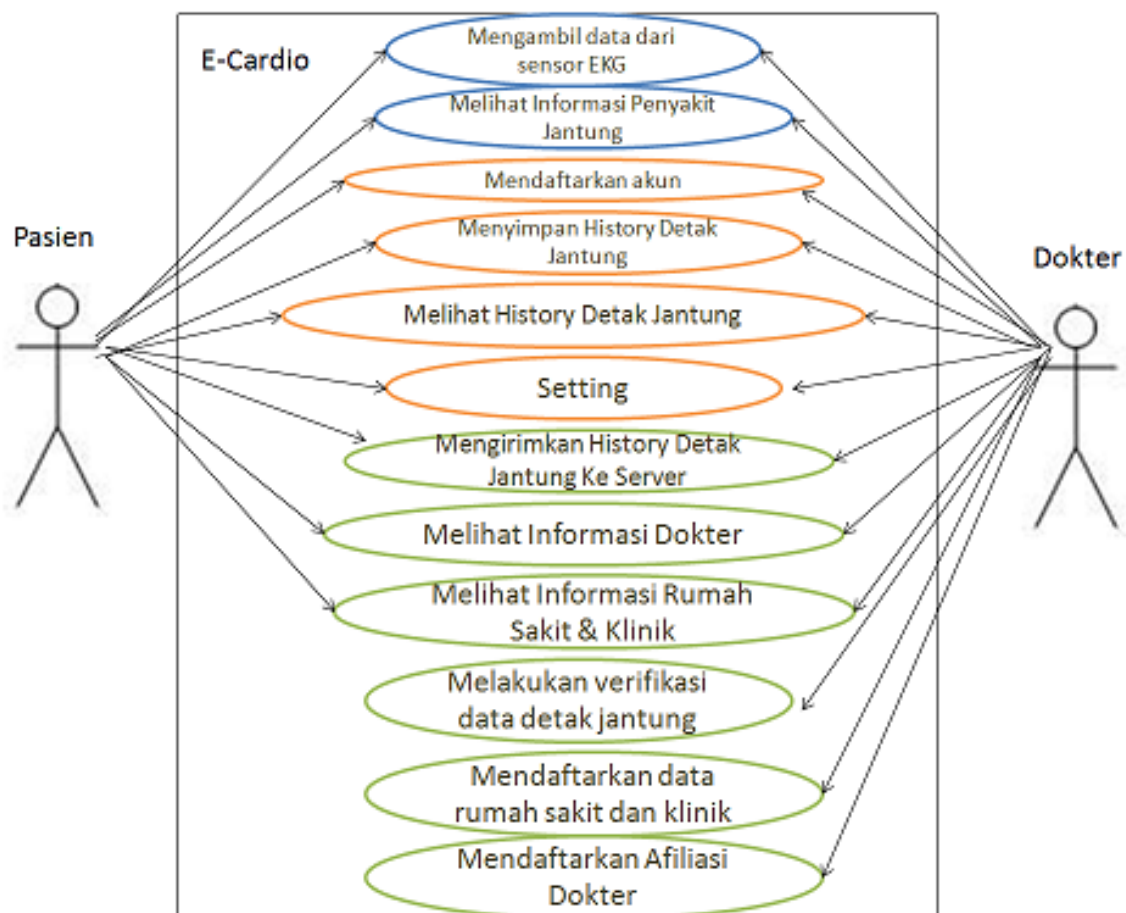
3.2. Rancangan Modifikasi Aplikasi E-Cardio

Seperti yang sudah dijelaskan pada bab sebelumnya, bahwa aplikasi E-Cardio masih memiliki berbagai macam kekurangan. Oleh sebab itu, perlu dilakukan modifikasi dan penambahan fitur agar dapat menjalankan tugas sesuai dengan harapan. Memang ada beberapa fitur yang sudah terimplementasi dengan baik dan tidak perlu diubah lagi pada aplikasi E-Cardio. Akan tetapi, ada fitur yang perlu dimodifikasi atau ditambahkan guna

mencapai hasil yang yang diinginkan. Hasil modifikasi aplikasi E-Cardio pada pelaksanaan penelitian ini akan diberi nama E-Cardio versi 3.0.

3.2.1. Use case Diagram

Kebutuhan *prototype* sistem dirancang dalam suatu *use case*. *Use case* yang akan digunakan dalam aplikasi E-Cardio versi 3.0 ini dapat dilihat pada gambar 3-5. Pada diagram *use case* dapat dilihat bahwa ada beberapa macam warna. Warna – warna tersebut hanya digunakan sebagai penanda saja dalam penegambangan aplikasi E-Cardio ini. *Use case* yang berwarna biru merupakan *use case* yang sudah diimplementasikan dan tidak akan diubah lagi pada modifikasi yang dilakukan nanti. *Use case* yang berwarna jingga merupakan *use case* yang sudah diimplementasikan, akan tetapi akan dimodifikasi lebih lanjut untuk memenuhi tujuan aplikasi yang diinginkan. Terakhir, *use case* yang berwarna hijau merupakan *use case* yang belum diimplementasikan, dan oleh sebab itu akan diimplementasikan dari awal.



Gambar 3-5. Use Case diagram pada aplikasi E-Cardio versi 3.0.

Untuk memudahkan proses implementasi yang akan dilakukan dari setiap *use case* akan dibuat *use case specification*. Keterangan lengkap mengenai spesifikasi setiap *use case* yang diimplementasikan dalam pelaksanaan penelitian ini akan dijelaskan pada bagian selanjutnya.

3.2.2. Use case Specification

Suba bab ini berisi penjelasan lengkap mengenai spesifikasi *use case* yang akan diimplementasikan. Dalam penjelasan spesifikasi *use case* ini akan dijelaskan mengenai deskripsi *use case*, *flow of event* yang terdiri dari *basic flow* dan *alternative flow*, *pre-condition* dan *post condition*. Keterangan lain mengenai *use case* tidak dijelaskan dalam laporan ini.

3.2.2.1. Use Case Specification : Mendaftarkan akun

Description

Use case “Mendaftarkan akun” merupakan salah satu kebutuhan fungsional pada aplikasi E-Cardio. Pengguna aplikasi baik dokter maupun pasien harus mendaftar diri mereka sebelum menggunakan sistem ini. Penyimpanan akan dilakukan pada *lite database* yang dibuat pada *smartphone* yang digunakan.

Flow of Event

Basic Flow

Aktor (Pasien / Dokter)	Respon dari Sistem
1. Memilih menu untuk membuat akun pengguna	
	2. Menampilkan form isian biodata pengguna
3. Mengisi form lalu mensubmit	
	4. Menyimpan data pengguna di <i>database</i>

Alternative Flow

Tidak ada *alternative flow* untuk *use case* ini.

Special Requirement

Tidak ada *special requirement* untuk *use case* ini

Pre-Condition

Pengguna telah membuka menu untuk pengaturan pengguna

Post-Condition

Data pengguna disimpan dalam *lite database*

3.2.2.2. Use Case Specification : Menyimpan *History* Detak Jantung

Description

Use case “Menyimpan *history* detak jantung” merupakan salah satu kebutuhan fungsional pada aplikasi E-Cardio. Pengguna aplikasi baik dokter maupun pasien dapat menyimpan data detak jantung yang telah mereka ambil dari sensor EKG. Penyimpanan akan dilakukan pada *lite database* yang dibuat pada *smartphone* yang digunakan.

Flow of Event

Basic Flow

Aktor (Pasien / Dokter)	Respon dari Sistem
1. Memilih menu untuk menyimpan data detak jantung	
	2. Menyimpan data detak jantung pada <i>lite database</i>

Alternative Flow

Tidak ada *alternative flow* untuk *use case* ini.

Special Requirement

Special requirement untuk *use case* ini adalah pengguna telah mendaftarkan (membuat) akun pada aplikasi. Selain itu akun pengguna juga harus menjadi *active user* saat itu.

Pre-Condition

Pengguna telah membuka menu untuk mengambil data detak jantung dari sensor EKG dan sinyal tersebut sudah dapat divisualisasikan.

Post-Condition

Data detak jantung disimpan dalam *lite database*.

3.2.2.3. Use Case Spesification : Melihat *History* Detak Jantung

Description

Use case “Melihat *history* detak jantung” merupakan salah satu kebutuhan fungsional pada aplikasi E-Cardio. Pengguna aplikasi baik dokter maupun pasien dapat melihat *history* data detak jantung yang telah disimpan dalam *lite database*. Selain itu pengguna dapat mengambil update terbaru dari *server* apakah data tersebut sudah diverifikasi oleh dokter atau belum.

Flow of Event

Basic Flow

Aktor (Pasien / Dokter)	Respon dari Sistem
1. Memilih menu untuk untuk melihat data <i>history</i> detak jantung	
2. Memilih salah satu data dari semua data <i>history detak jantung</i>	
3. Memilih opsi untuk melihat detail data detak jantung	
	4. Memberikan data detail tentang detak jantung yang diminta pengguna

5. Memilih opsi untuk meminta update data detak jantung terbaru dari <i>server</i>	
	6. Memberikan data detak jantung terupdate dari <i>server</i>

Alternative Flow

Alternative flow dari tahap nomor 6, jika tidak ada update terbaru untuk data yang diminta maka sistem akan memberikan pesan bahwa tidak ada update terbaru .

Special Requirement

Special requirement untuk *use case* ini adalah pengguna menjadi *active user* saat itu. Selain itu data detak jantung yang ingin dilihat oleh *user* sudah disimpan.

Pre-Condition

Pengguna telah membuka menu utama, dan akan memilih menu untuk melihat data *history* detak jantung.

Post-Condition

Pengguna dapat melihat detail data detak jantung dan update terbaru apakah ada verifikasi dari dokter.

3.2.2.4. Use Case Spesification : Setting

Description

Use case “Setting” merupakan salah satu kebutuhan fungsional pada aplikasi E-Cardio. Pengguna aplikasi baik dokter maupun pasien dapat mengatur berbagai konfigurasi yang dibutuhkan oleh aplikasi. Pengaturan yang dapat dilakukan antara lain adalah pengaturan email, pengauran *bluetooth*, dan pengaturan *server*. Pengaturan *bluetooth* dan pengaturan email sudah diimplementasikan pada aplikasi versi 2.0 sebelumnya. Oleh sebab itu pada penelitian kali ini yang akan diimplementasikan adalah pengaturan *server*.

Flow of Event

Basic Flow

Aktor (Pasién / Dokter)	Respon dari Sistem
1. Pengguna memilih menu pengaturan	
2. Pengguna memilih sub menu pengaturan <i>server</i>	
	3. Memberikan form pengisian nama <i>server</i>
4. Pengguna memasukkan nama <i>server</i>	
5. Pengguna memilih opsi untuk menyimpan nam <i>server</i>	
	6. Sistem menyimpan nama <i>server</i>

Alternative Flow

Alternative flow dari tahap 5, jika pengguna memilih opsi untuk *cancel*, maka sistem tidak jadi menyimpan nama *server*.

Special Requirement

Tidak ada *special requirement* untuk *use case* ini.

Pre-Condition

Pengguna telah membuka menu utama .

Post-Condition

Data nama *server* dalam system

3.2.2.5. Use Case Specification : Mengirimkan *History* Detak Jantung Ke *Server*

Description

Use case “Mengirimkan *history* detak jantung ke *server*” merupakan salah satu kebutuhan fungsional pada aplikasi E-Cardio. Pengguna aplikasi baik dokter maupun pasien dapat mengirimkan *history* data detak jantung yang telah disimpan dalam *lite database* ke *server*. Dengan demikian data detak jantung pengguna dapat diverifikasi oleh dokter.

Flow of Event

Basic Flow

Aktor (Pasien / Dokter)	Respon dari Sistem
1. Memilih menu untuk untuk melihat data <i>history</i> detak jantung	
2. Memilih salah satu data dari semua data <i>history detak jantung</i>	
3. Memilih opsi untuk melihat detail data detak jantung	
	4. Memberikan data detail tentang detak jantung yang diminta pengguna
5. Memilih opsi untuk mengirimkan data detak jantung terbaru ke <i>server</i>	
	6. Data detak jantung terupdate dikirimkan ke <i>server</i>
	7. Data detak jantung disimpan di <i>server</i> (<i>online database</i>)

Alternative Flow

Alternative flow dari tahap nomor 8. Jika data detak jantung sudah pernah disimpan di *server*, maka sistem akan memberi tahu kepada pengguna bahwa data detak jantung sudah pernah disimpan di *server*.

Special Requirement

Special requirement untuk *use case* ini adalah pengguna menjadi *active user* saat itu. Selain itu data detak jantung yang ingin dilihat oleh *user* sudah disimpan.

Pre-Condition

Pengguna telah membuka menu utama .

Post-Condition

Data detak jantung disimpan di *server*.

3.2.2.6. Use Case Spesification : Melihat Informasi Dokter

Description

Use case “Melihat informasi dokter” merupakan salah satu fitur fungsional pada aplikasi E-Cardio. Pengguna aplikasi baik dokter maupun pasien dapat mengetahui siapa sajakah dokter spesialis jantung yang memberikan kontribusinya dalam meberikan verifikasi detak jantung. Selain itu pengguna dapat melihat data detail setiap dokter, termasuk data diri dan afiliasi dokter yang bersangkutan

Flow of Event

Basic Flow

Aktor (Pasien / Dokter)	Respon dari Sistem
1. Memilih menu untuk untuk melihat data dokter	
	2. Sistem meberikan list seluruh dokter yang tergabung dalam aplikasi
3. Memilih salah satu dokter pada list	
	4. Memberikan data detail tantang dokter

Alternative Flow

Alternative flow dari tahap nomor 5, jika pengguna memilih untuk mengunduh update data dokter dari *server*, maka sistem akan mengunduh data dokter *terupdate* dari *server*.

Special Requirement

Tidak ada *special requirement* untuk *use case* ini.

Pre-Condition

Pengguna telah membuka menu utama.

Post-Condition

Pengguna dapat melihat detail data dokter yang diinginkan.

3.2.2.7. Use Case Spesification : Melihat Informasi Rumah Sakit dan Klinik

Description

Use case “Melihat informasi rumah sakit dan klinik” merupakan salah satu fitur fungsional pada aplikasi E-Cardio. Pengguna aplikasi dapat mengetahui rujukan rumah sakit atau klinik yang disarankan oleh dokter. Di sisi lain dokter dapat memberikan rujukan rumah sakit dan klinik kepada pasien saat memberikan verifikasi detak jantung. Dengan demikian jika keadaan pasien dirasa perlu diperiksa di rumah sakit atau klinik, informasi ini sangat membantu.

Flow of Event

Basic Flow

Aktor (Pasien / Dokter)	Respon dari Sistem
1. Memilih menu untuk untuk melihat data rumah sakit dan klinik	
	2. Sistem meberikan list rumah sakit dan klinik yang tergabung dalam aplikasi

3. Memilih salah satu rumah sakit atau klinik pada list	
	4. Memberikan data detail tentang rumah sakit atau klinik

Alternative Flow

Alternative flow dari tahap nomor 5, jika pengguna memilih untuk mengunduh update data dokter dari *server*, maka sistem akan mengunduh data rumah sakit dan klinik *ter-update* dari *server*.

Special Requirement

Tidak ada *special requirement* untuk *use case* ini

Pre-Condition

Pengguna telah membuka menu utama.

Post-Condition

Pengguna dapat melihat detail data dokter yang diinginkan.

3.2.2.8. Use Case Specification : Melakukan Verifikasi Data Detak Jantung

Description

Use case “Melakukan verifikasi data detak jantung” merupakan salah satu kebutuhan fungsional pada aplikasi E-Cardio. Pengguna aplikasi baik dokter maupun pasien dapat mengetahui keadaan jantungnya setelah detak jantung yang telah dia ambil dari EKG diverifikasi oleh dokter. Oleh sebab itu perlu diimplementasikan fitur verifikasi detak jantung oleh dokter.

Flow of Event

Basic Flow

Aktor (Dokter)	Respon dari Sistem
1. Memilih menu untuk verifikasi data detak jantung	
	2. Memberikan list data detak jantung yang belum terverifikasi
3. Memilih salah satu data dari semua data detak <i>detak jantung</i>	
	4. Memberikan form verifikasi
5. Menulis pesan tentang kondisi detak jantung dan rumah sakit atau klinik sebagai rujukan bila perlu	
6. Mengirim verifikasi ke <i>server</i>	
	7. <i>Server mengupdate data history detak jantung yang ada di database</i>

Alternative Flow

Alternative flow dari tahap nomor 5. Jika dokter membatalkan pengiriman verifikasi ke *server*, maka akan kembali pada form pengisian

Special Requirement

Special requirement untuk *use case* ini adalah pengguna menjadi *active user* saat itu.

Pre-Condition

Pengguna telah membuka menu utama .

Post-Condition

Verifikasi suatu data detak jantung tersimpan di *server*.

3.2.2.9. Use Case Spesification : Mendaftarkan Data Rumah Sakit dan Klinik

Description

Use case “Mendaftarkan Rumah Sakit dan Klinik” merupakan salah satu kebutuhan fungsional pada aplikasi E-Cardio. Dokter perlu meberikan rujukan ke rumah sakit atau klinik kepada pasien yang keadaan jantungnya harus segera diperiksa. Di sisi lain pasien juga perlu mengetahui rumah sakit atau klinik yang disarankan oleh dokter. Dalam sistem ini hanya dokter yang dapat mendaftar data rumah sakit dan klinik ke *database*.

Flow of Event

Basic Flow

Aktor (Dokter)	Respon dari Sistem
1. Memilih menu untuk untuk mendaftar rumah sakit dan klinik	
	2. Memberikan form untuk pengisian data rumah sakit dan klinik
3. Mengisi form data rumah sakit dan klinik lalu mensubmit	
	4. Menyimpan data rumah sakit dan klinik di <i>server</i>

Alternative Flow

Alternative flow dari tahap nomor 4 Jika *server* memberitahukan bahwa data rumah sakit dan klinik yang didaftarkan sudah terdaftar yang di maka akan kembali pada form pengisian

Special Requirement

Special requirement untuk *use case* ini adalah pengguna menjadi *active user* saat itu.

Pre-Condition

Pengguna telah membuka menu utama .

Post-Condition

Data rumah sakit dan klinik jantung tersimpan di *server*.

3.2.2.10. Use Case Spesification : Mendaftarkan Afiliasi Dokter

Description

Use case “Mendaftarkan Afiliasi Dokter” merupakan salah satu fitur tambahan pada aplikasi E-Cardio. Dokter perlu memberikan verifikasi detak jantung dan rujukan ke rumah sakit atau klinik kepada pasien yang keadaan jantungnya harus segera diperiksa. Verifikasi dan rujukan tersebut diberikan kepada pasien. Akan tetapi agar pasien lebih yakin dengan kredibilitas dokter yang memberikan verifikasi, maka pasien perlu mengetahui afiliasi dari dokter tersebut. Dengan demikian kepercayaan kepada dokter dari pasien akan lebih kuat.

Flow of Event

Basic Flow

Aktor (Dokter)	Respon dari Sistem
1. Memilih menu untuk untuk mendaftarkan afiliasi dokter	
	2. Memberikan form untuk pengisian data afiliasi dokter
3. Mengisi form data afiliasi dokter lalu mensubmit	
	4. Menyimpan data afiliasi dokter di <i>server</i>

Alternative Flow

Alternative flow dari tahap nomor 4 Jika *server* memberitahukan bahwa data afiliasi dari dokter sudah terdaftar maka akan kembali pada form pengisian.

Special Requirement

Special requirement untuk *use case* ini adalah pengguna menjadi *active user* saat itu.

Pre-Condition

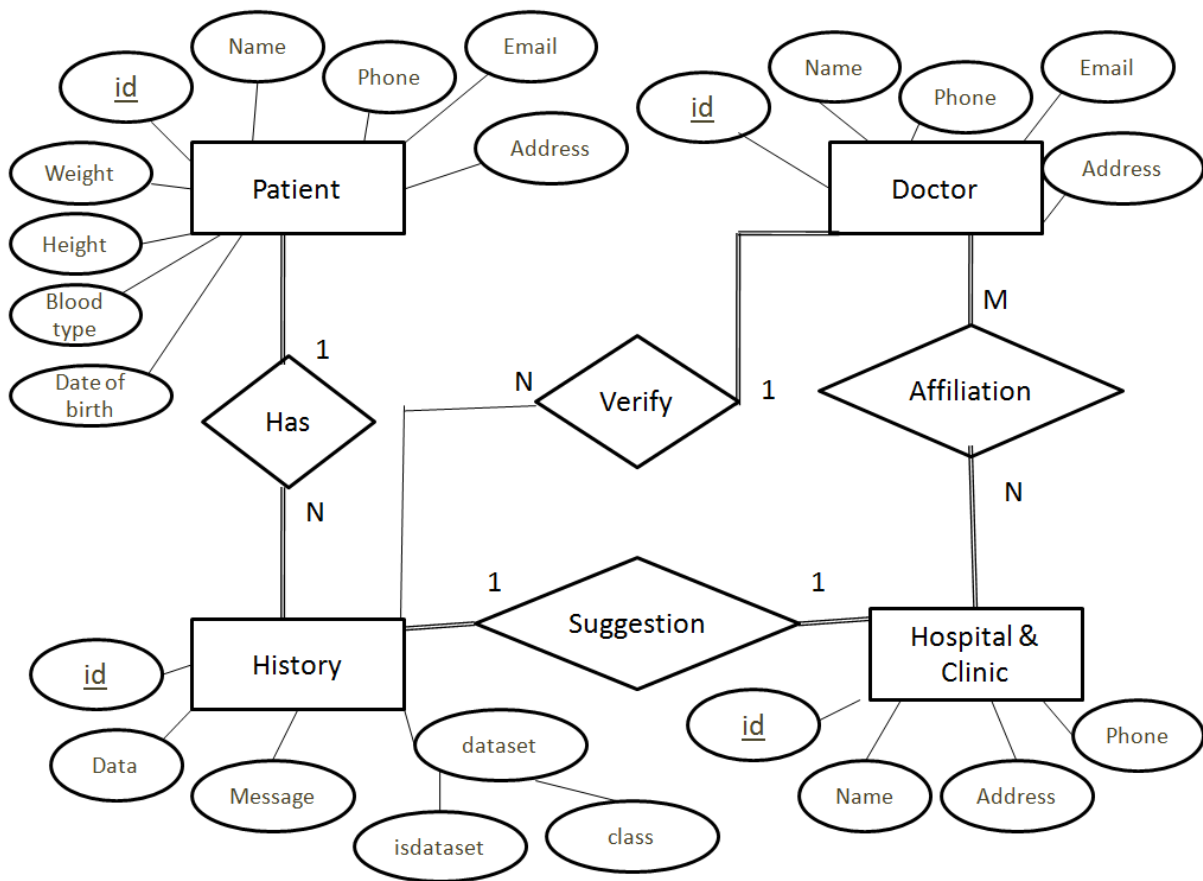
Pengguna telah membuka menu utama .

Post-Condition

Verifikasi suatu data detak jantung tersimpan di *server*.

3.3. Rancangan Implementasi Database

Seperti yang sudah dijelaskan dalam paparan sebelumnya, bahwa dalam implementasi *prototype* sistem terintegrasi pendeteksian dini dan *monitoring* penyakit jantung ini dibuat suatu *database* untuk penyimpanan data. Data – data yang disimpan dalam *database* ini adalah data pasien, data dokter, data rumah sakit. Selain itu data *history* detak jantung pasien juga disimpan dalam *database*. Untuk keperluan referensi afiliasi dokter yang dibutuhkan oleh pasien, maka data afiliasi dokter juga disimpan dalam *database*. Diagram *entity relationship database* yang digunakan dalam *prototype* ini dapat dilihat pada gambar 3-6



Gambar 3-6. Diagram *entity relationship* untuk *prototype*.

3.4. Rancangan Implementasi Server

Server merupakan komponen yang menghubungkan pasien dan dokter. Melalui *server* pasien dan dokter dapat bertukar data. Dokter juga dapat memberikan *verifikasi* data detak jantung yang dikirimkan oleh pasien. Seperti yang sudah dijelaskan pada awal bab ini bahwa pada *server* ada dua komponen, aplikasi *web* dan *database*. Struktur *database* yang akan dibuat sudah dijelaskan pada subbab berikutnya. Komponen-komponen yang ada pada *server* juga sudah dijelaskan pada subbab arsitektur sistem. *Server* mengakomodasi aplikasi E-Cardio untuk mengambil data atau mengunggah data untuk disimpan. Oleh sebab itu *server* dirancang dengan fungsi – fungsi yang termodularisasi. Fungsi – fungsi tersebut selanjutnya dibagi pada setiap URL pada *server*. Pembagian URL tersebut adalah sebagai berikut :

1. *Server/RegisiterPatien/*

Merupakan URL pada *server* yang berfungsi untuk mendaftarkan akun pasien.

2. *Server/ RegsiterDoctor/*

Merupakan URL pada *server* yang berfungsi untuk mendaftarkan akun dokter.

3. *Server/UploadHistory/*

Merupakan URL pada *server* yang berfungsi untuk mengunggah data *history* detak jantung untuk disimpan di *database* yang ada di *server*.

4. *Server/LookHistory/*

Merupakan URL pada *server* yang berfungsi untuk mengunduh data *history* detak jantung untuk disimpan di *database* yang ada di *server*.

5. *Server/GetDoctorData/*

Merupakan URL pada *server* yang berfungsi untuk mengambil data lengkap seorang dokter.

6. *Server/GetHospitalData*

Merupakan URL pada *server* yang berfungsi untuk mengambil data suatu rumah sakit atau klinik.

7. *Server/GetUnverifiedHistoryData*

Merupakan URL pada *server* yang berfungsi untuk mengambil data- data detak jantung di *database* yang belum diverifikasi oleh dokter.

8. *Server/VerifyHistory*

Merupakan URL pada *server* yang berfungsi untuk mengunggah hasil verifikasi data data detak jantung oleh dokter untuk disimpan di *database*.

9. *Server/RegsiterHospital*

Merupakan URL pada *server* yang berfungsi untuk mendaftarkan data rumah sakit dan klinik untuk disimpan di *database*.

10. *Server/RegsiterAffiliation*

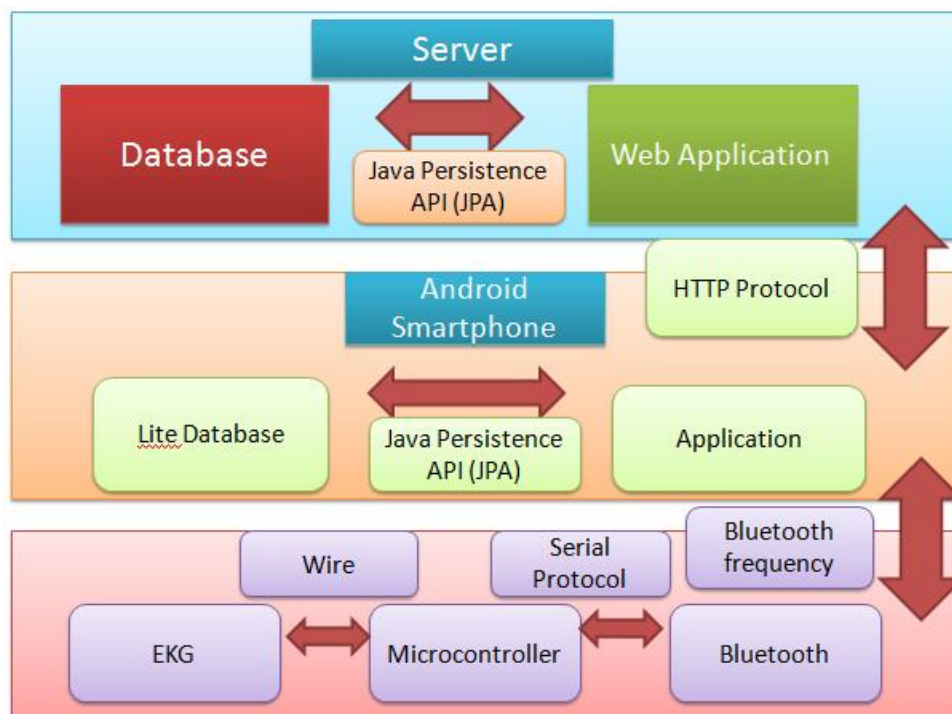
Merupakan URL pada *server* yang berfungsi untuk mendaftarkan afiliasi dokter.

11. *Server/GetDoctorAffiliation*

Merupakan URL pada *server* yang berfungsi untuk mengambil data afiliasi dokter.

3.5. Rancangan Interaksi Antar Komponen

Prototype ini memiliki tiga komponen seperti yang sudah dijelaskan pada bab sebelumnya. Agar ketiga komponen tersebut dapat bekerja dan berinteraksi satu sama lain dengan benar, maka perlu disusun suatu rancangan interaksi antar komponen. Rancangan Interaksi antar komponen pada *prototype* ini dapat dilihat pada gambar 3-7. Pada komponen *hardware* EKG dan *microcontroller* berinteraksi dengan sambungan kabel (*wire connector*). Antara komponen mikrokontroler dan *bluetooth* berkomunikasi melalui protokol *serial*. Selanjutnya anatar *Bluetooth* dan *smartphone* Android berkomunikasi melalui jaringan *bluetooth* pada frekuensi tertentu. Pada *smartphone* Android dibangun aplikasi E-Cardio dan juga *lite database*. Komunikasi anantara aplikasi dan *lite database* ini berjalan dengan penghubung *java database connector* untuk Android. Selanjutnya *smartphone* Android dan *server* berkomunikasi dengan protokol HTTP. Berikutnya, pada *server* juga ada dua komponen, yaitu *database* dan aplikasi *web*. Aplikasi *web* dan *database* yang dibangun di *server* berkomunikasi menggunakan API dari *java persistence* atau yang lebih dikenal sebagai JPA.



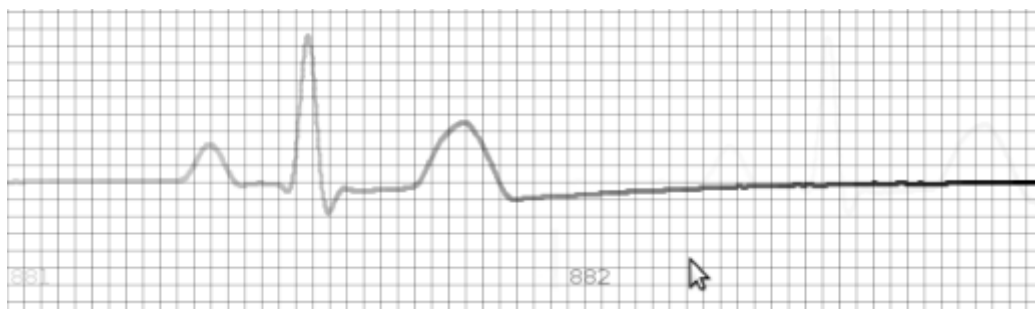
Gambar 3-7. Diagram interaksi komponen-komponen *prototype* system.

3.6. Data Detak Jantung yang Digunakan

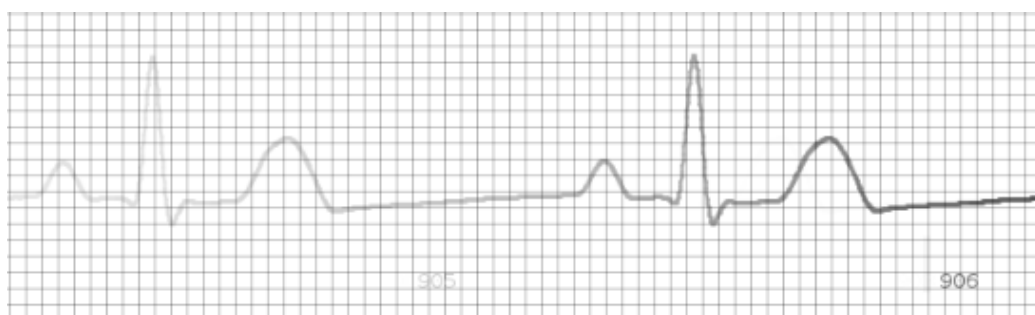
Penelitian ini bertujuan untuk membuat suatu perangkat cerdas untuk pendeteksian dini dan *monitoring* penyakit jantung. Dalam pengujian alat ini akan digunakan beberapa macam tipe (kelas) sinyal detak jantung. Kelas untuk detak jantung ini dibagi menjadi tujuh belas (17) kelas yaitu, dua belas (12) kelas untuk detak jantung yang menandakan indikasi penyakit dan lima (5) detak jantung yang menandakan detak jantung normal. Contoh detak jantung diambil menggunakan pasien simulator dimana pasien simulator ini dapat menghasilkan gelombang detak jantung manusia. Di bawah ini adalah contoh gambar gelombang detak jantung beserta penjelasan masing-masing kelas penyakit jantungnya.

3.6.1. Kelas Detak Jantung Normal

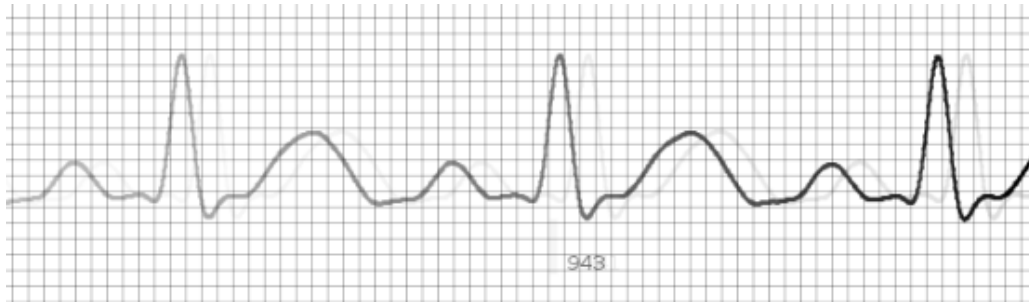
Kelas detak jantung normal merupakan kelas yang terdiri dari data-data detak jantung yang normal dalam berbagai variasi frekuensi detakan (*beat*). Contoh gelombang detak jantung normal dapat dilihat pada gambar 3-8.



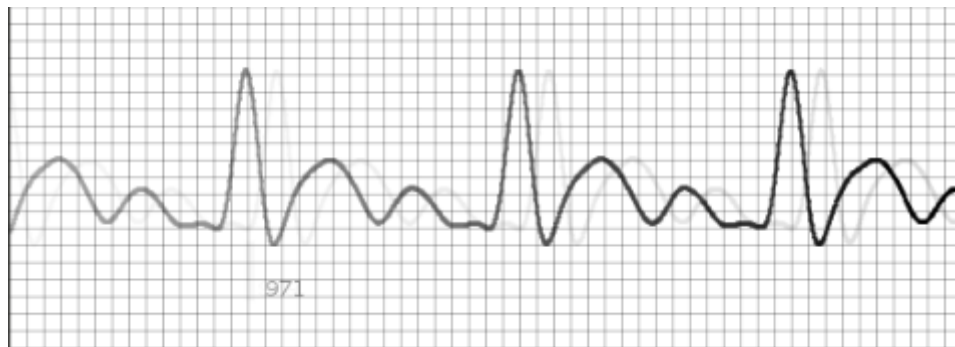
(a)



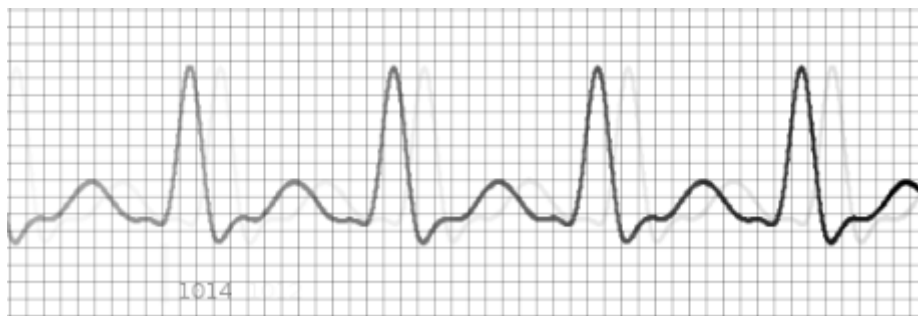
(b)



(c)



(d)



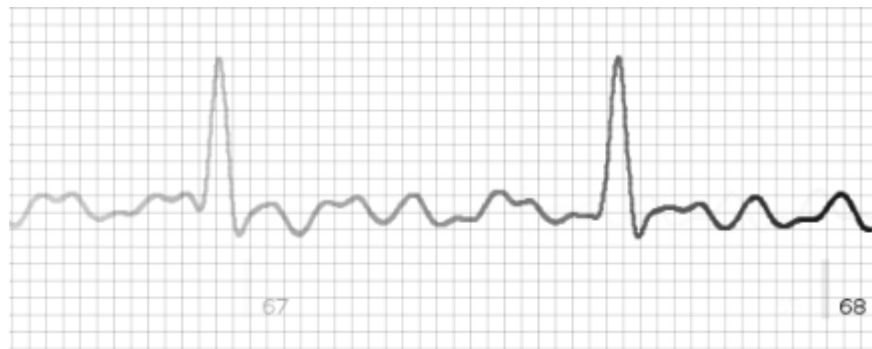
(e)

Gambar 3-8. Gambar lima kelas contoh gelombang detak jantung manusia normal saat dalam kondisi 30 bpm (a), 60 bpm (b), 120 bpm (c), 180 bpm (d), dan 240 bpm (e) yang diambil menggunakan pasien simulator.

3.6.2. AFib

Salah satu gangguan jantung yang paling umum terjadi dan disebabkan adanya gangguan pada atrium jantung adalah *Atrial fibrillation* (AF atau AFib). Walaupun AFib dianggap tidak mengancam jiwa penderita dan dianggap sebagai gangguan yang bersifat aman, *arrhythmia* tipe ini merupakan *arrhythmia* yang merepotkan karena banyaknya komplikasi gangguan kesehatan yang dihasilkan seperti detak jantung yang tidak teratur sehingga darah tidak mengalir dengan lancar. Selain itu, penyakit ini sulit diatasi dikarenakan kurangnya terapi

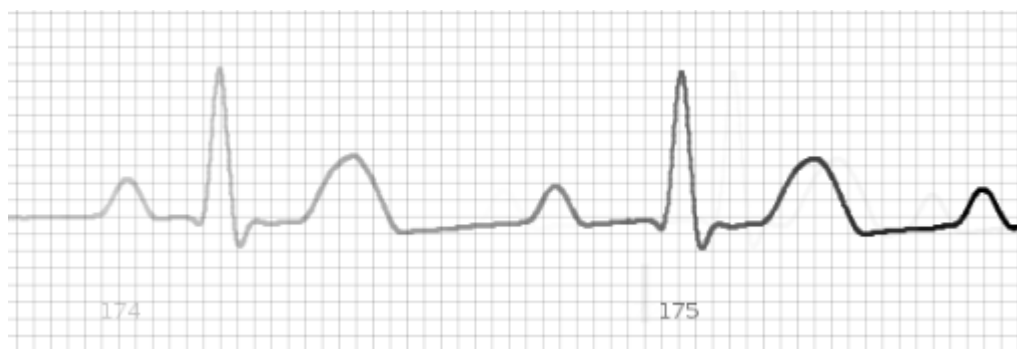
yang efektif dan aman. Pada gambar 3-9 diperlihatkan detak jantung *Atrial fibrillation* yang dihasilkan oleh pasien simulator.



Gambar 3-9. Contoh detak jantung manusia penderita AFib yang diambil menggunakan pasien simulator.

3.6.3. 2-BLK 1

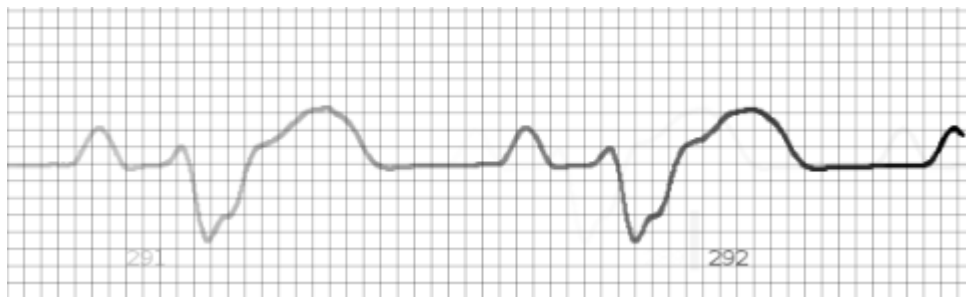
Adanya keterlambatan atau total blok impuls listrik saat perjalanan dari titik sinus ke ventrikel biasanya akan menyebabkan gangguan jenis ini. Jantung menjadi berdetak secara tidak teratur dikarenakan tingkat blok dan keterlambatan mungkin terjadi pada katup aorta atau *His-Purkinje* sistem. Apabila gangguan ini dirasakan mulai serius, dapat ditangani dengan pengawasan dari tim medis khusus dan alat pacu jantung. Contoh detak jantung manusia penderita penyakit 2-BLK 1 yang dihasilkan pasien simulator diperlihatkan oleh gambar 3-10.



Gambar 3-10. Contoh detak jantung manusia penderita 2-BLK 1 yang diambil menggunakan pasien simulator.

3.6.4. RBBB

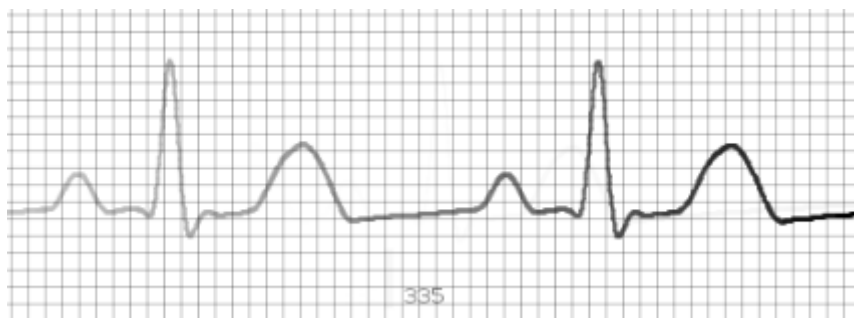
Kerusakan pada konduksi elektrik pada jantung sering disebut dengan penyakit RBBB atau *Right Bundle Branch Block*. Gejala dari RBBB berupa, ventrikel kanan tidak teraktifasi secara langsung oleh impuls lewat lapisan pembuluh atas jantung sebelah kanan namun ventrikel kiri tetap normal. Walaupun dapat terjadi pada orang yang sehat-sehat saja namun umumnya RBBB biasanya disebabkan oleh penyakit lain. Contoh detak jantung manusia penderita penyakit RBBB yang dihasilkan pasien simulator diperlihatkan oleh gambar 3-11



Gambar 3-11. Contoh detak jantung manusia penderita RBBB yang diambil menggunakan pasien simulator.

3.6.5. PAC

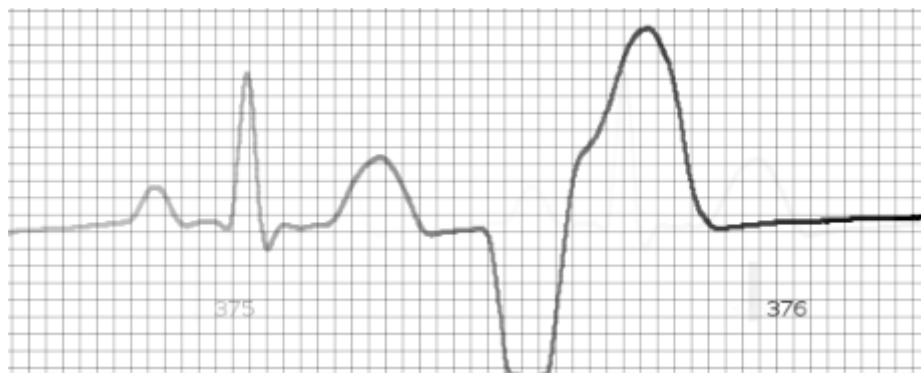
Prematur Atrial Contraction(PAC) adalah *arrhythmia* yang bersifat tidak mematikan dan umum. PAC adalah gangguan jantung yang dikarenakan detak jantung yang dihasilkan berasal jauh dari titik sinus yang seharusnya, yaitu titik yang mengirimkan sinyal listrik melalui atrium jantung. Penderita PAC dapat merasakan adanya detak jantung yang terlewatkan pada selang waktu sesaat. Penyebab terkena PAC dan meningkatkan potensi terjadinya PAC adalah kadar kafein, nikotin, alkohol serta tingkat stress yang tinggi. Satu detak jantung manusia penderita penyakit PAC yang dihasilkan pasien simulator diperlihatkan oleh gambar 3-12.



Gambar 3-12. Contoh satu detak jantung manusia penderita PAC yang diambil menggunakan pasien simulator.

3.6.6. PVC EARLY

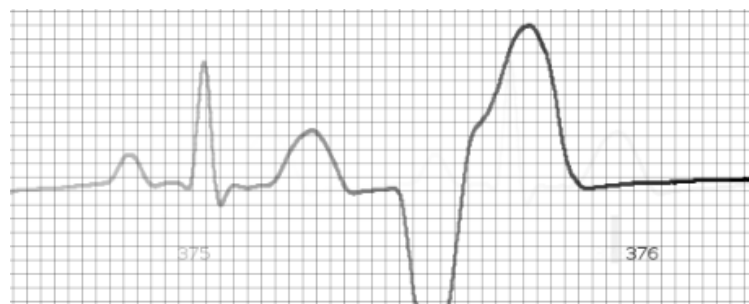
Salah satu penyakit *arrhythmia* yang paling sering terjadi pada manusia adalah PVC *Early*. Stress berlebihan, terlalu banyak kandungan kafein dalam tubuhnya ataupun nikotin merupakan penyebab terjadinya penyakit ini. Terkadang serangan jantung atau ketidakseimbangan elektrolit dalam tubuh dapat menyebabkan PVC terjadi. Kategori PVC *Early* biasanya jarang memerlukan penanganan medis dan tidak membahayakan. Satu detak jantung manusia penderita penyakit PVC *Early* yang dihasilkan pasien simulator diperlihatkan oleh gambar 3-13.



Gambar 3-13. Contoh satu detak jantung manusia penderita PVC *Early* yang diambil menggunakan pasien simulator.

3.6.7. PVC R-On-T

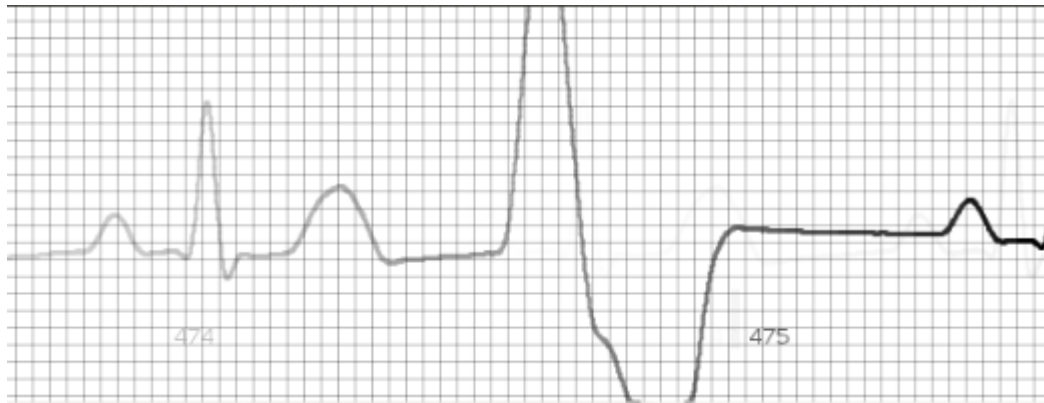
PVC R-On-T dalam beberapa kasus dapat menyebabkan *arrhythmia*. Penyakit ini terjadi ketika jantung menghasilkan detak elektrik yang berbahaya. Jenis *arrhythmia* yang terkait adalah *Ventricular Tachycardia* (V-TACH). *Arrhythmia* ini digolongkan sebagai *arrhythmia* yang cukup fatal dan membahayakan pasiennya. Contoh detak jantung manusia penderita penyakit PVC R-On-T yang dihasilkan pasien simulator diperlihatkan oleh gambar 3-14.



Gambar 3-14. Contoh detak jantung manusia penderita PVC R-On-T yang diambil menggunakan pasien simulator.

3.6.8. MF PVC

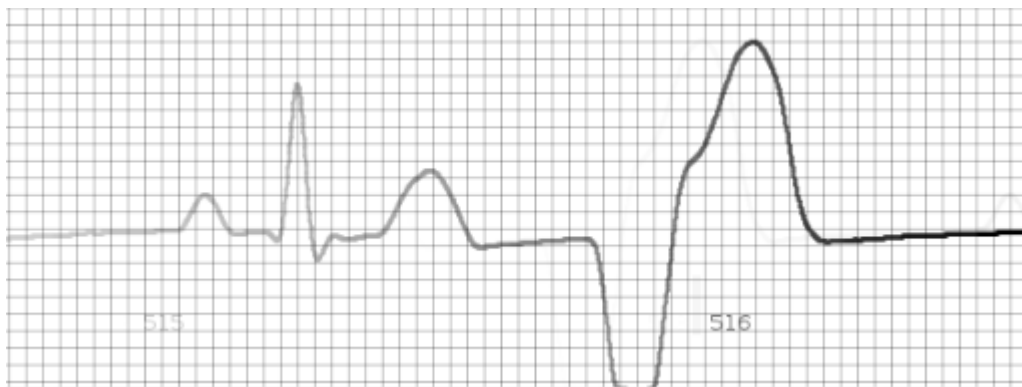
Multifocal PVC (MF PVC) tergolong sebagai *arrhythmia* yang berbahaya. Contoh detak jantung manusia penderita penyakit MF PVC yang dihasilkan pasien simulator diperlihatkan pada gambar 3-15.



Gambar 3-15. Contoh detak jantung manusia penderita MF PVC yang diambil menggunakan pasien simulator.

3.6.9. BIGEMINY

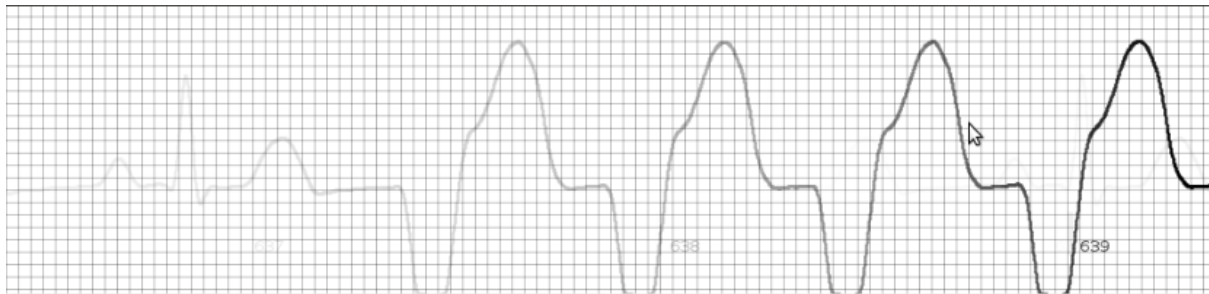
Bigeminy merupakan gangguan *arrhythmia* yang berciri-ciri detak jantung yang abnormal. Detak jantung penderita *Bigeminy* ini biasanya dibawah 60 BPM (*beat per minute*). Asap rokok, alkohol, kafein, kekurangan magnesium dan potassium, kelelahan yang ekstrem serta keracunan CO₂ dapat menyebabkan *Bigeminy*. Contoh detak jantung manusia penderita penyakit *Bigeminy* yang dihasilkan pasien simulator diperlihatkan oleh gambar 3-16.



Gambar 3-16. Contoh detak jantung manusia penderita *Bigeminy* yang diambil menggunakan pasien simulator.

3.6.10. RUN 5 PVC

RUN 5 PVC merupakan varian dari Premature Ventricular Contraction (PVC). Contoh detak jantung manusia penderita penyakit RUN 5 PVC yang dihasilkan pasien simulator diperlihatkan oleh Gambar 3-17.



Gambar 3-17. Contoh detak jantung manusia penderita RUN 5 PVC yang diambil menggunakan pasien simulator.

3.6.11. V-TACH

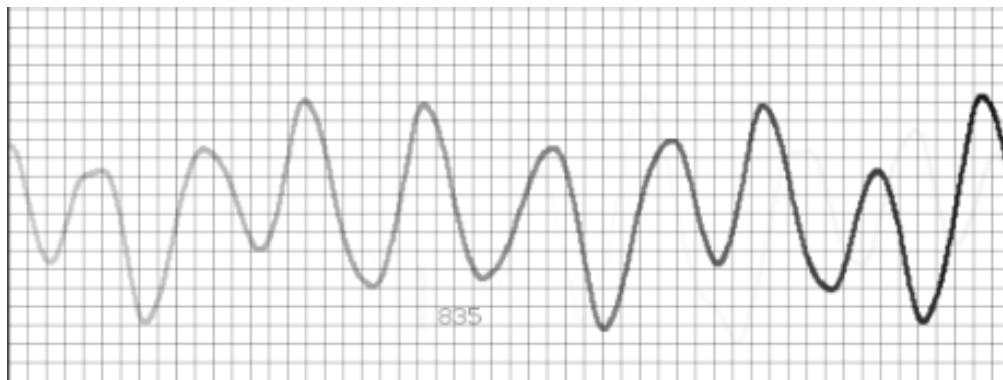
Ventricular tachycardia (V-TACH) adalah penyakit *arrhythmia* yang menyebabkan pasiennya menderita detak jantung sangat cepat. Karena dapat memicu terjadinya kematian mendadak dan asistol (berhentinya detak jantung, biasanya pada monitor jantung, grafik menjadi datar), *Arrhythmia* jenis ini termasuk sebagai salah satu *arrhythmia* yang dapat membahayakan nyawa penderitanya. Satu detak jantung manusia penderita penyakit V-TACH yang dihasilkan pasien simulator dapat diperlihatkan oleh gambar 3-18.



Gambar 3-18. Contoh satu detak jantung manusia penderita V-TACH yang diambil menggunakan pasien simulator.

3.6.12. V-FIB

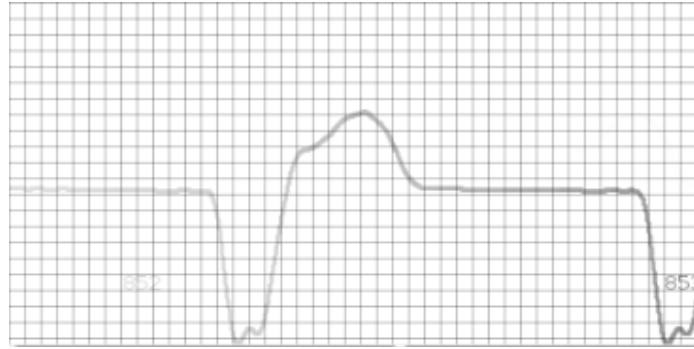
Ventricular fibrilasi (V-FIB) adalah *arrhythmia* yang banyak dialami oleh penderita gangguan jantung dan bersifat cukup membahayakan. Penyebab dari gangguan ini adalah adanya suatu momen dimana kontraksi dari otot jantung tidak terkoordinasi dengan baik. Karena jika kejadian diatas dibiarkan beberapa detik kemudian, akan menyebabkan asistol dan berefek pada kematian sang pasien, maka kejadian tersebut perlu ditangani secepatnya. Selain itu *ventricular fibrilasi* juga dapat menyebabkan *shock* jantung yang berakibat jantung berhenti mendadak dan akan menyebabkan kematian dalam beberapa menit kedepannya. Contoh detak jantung manusia penderita penyakit V-FIB yang dihasilkan pasien simulator diperlihatkan oleh gambar 3-19.



Gambar 3-19. Contoh detak jantung manusia penderita V-FIB yang diambil menggunakan pasien simulator.

3.6.13. PACED

PACED merupakan salah satu gangguan *arrhythmia* yang berciri-ciri detak jantung yang abnormal. Satu detak jantung manusia penderita penyakit PACED yang dihasilkan pasien simulator diperlihatkan oleh gambar 3-20



Gambar 3-20. Contoh satu detak jantung manusia penderita PACED yang diambil menggunakan pasien simulator.

BAB 4 IMPLEMENTASI

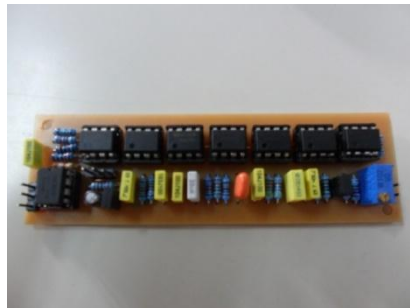
Bab implementasi merupakan bab yang menjelaskan mengenai implementasi yang dilakukan pada penelitian. Penjelasan tersebut mencakup perangkat yang digunakan, implementasi *database*, implementasi aplikasi *web* pada *server*, implementasi modifikasi E-Cardio, implementasi *web service*, dan implementasi sinkronisasi data.

4.1. Perangkat Yang Digunakan

Dalam membuat *prototype* ini diperlukan perangkat-perangkat yang menyusun sistem. Perangkat –perangkat yang digunakan dalam pembuatan *prototype* ini akan dijelaskan pada sub bab berikutnya.

4.1.1. Elektrokardiogram

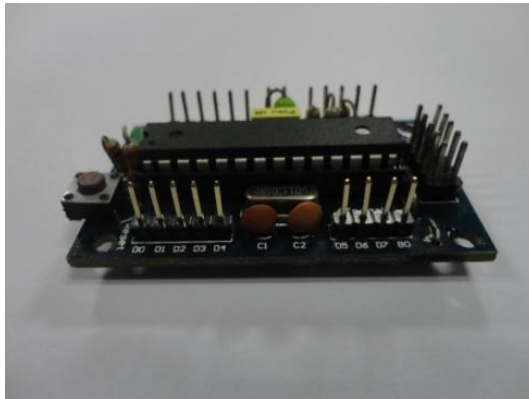
Elektrokardiogram yang akan digunakan pada prototipe ini adalah elektrokardiogram sederhana, dimana data output dari sinyal detak jantung dari EKG ini disajikan dalam satu macam sinyal (*single lead*) . Penjelasan lengkap mengenai elektrokardiogram ini sudah dijelaskan pada bab landasan teori. Bentuk fisik dari elektrokardiogram ini dapat dilihat pada gambar 4-1.



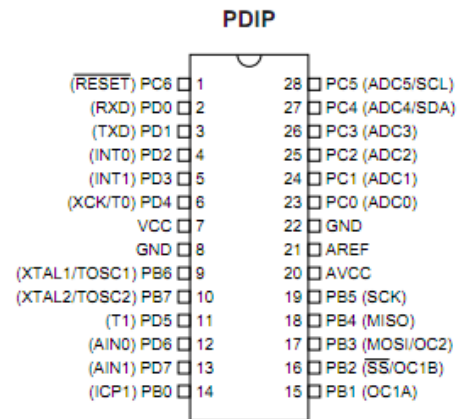
Gambar 4-1. Elektrokardiogram yang digunakan pada *prototype* system.

4.1.2. Mikrokontroller

Mikrokontroller dalam *prototype* sistem ini bertugas untuk mengubah data detak jantung yang berbentuk data analog menjadi data digital. Selain itu komponen ini bertugas mengirimkan data detak jantung yang sudah dalam bentuk analog ke perangkat *smartphone*. Jenis mikrikontroller yang dipakai pada prototipe ini adalah ATMEL Atmega8L . Bentuk fisik dari mikrokontroller yang digunakan dapat dilihat pada gambar 4-2



(a)

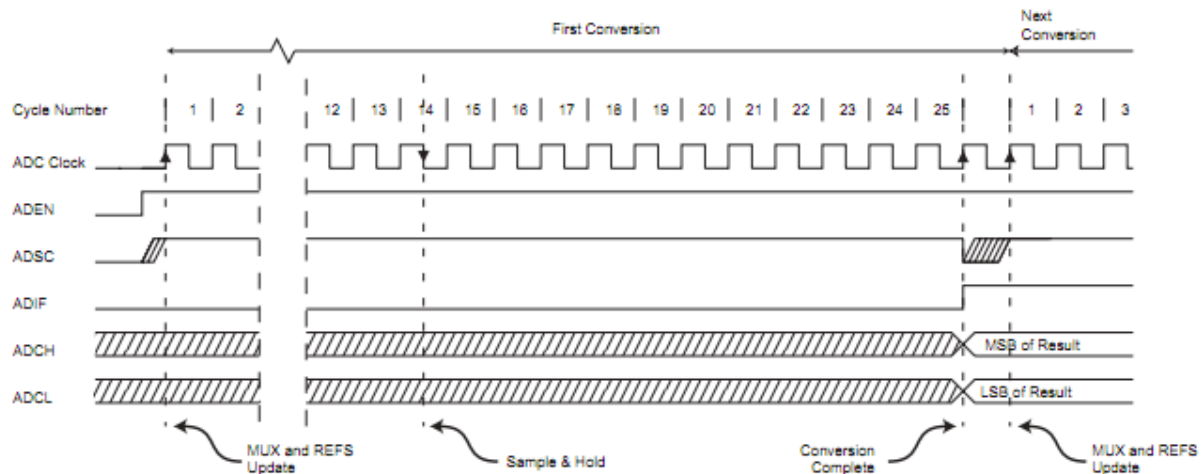


(b)

Gambar 4-2. Mikrokontroler ATmega8L. (a) Sistem minimum (b). Konfigurasi pin (kanan).

Mikrokontroler berperan mengubah data-data detak jantung yang dikirimkan oleh elektrokardiogram dalam bentuk analog menjadi data-data dalam bentuk digital. Mikrokontroler dapat menjalankan fungsi tersebut karena memiliki fitur/utilitas *analog to digital converter* (ADC). Cara kerja utilitas ADC adalah dengan membandingkan voltase input yang masuk ke suatu pin pada mikrokontroler dengan nilai voltase maksimal yang diijinkan dioperasikan pada mikrokontroler (VCC). Suatu fitur ADC memiliki ketelitian tertentu, misalnya 8 bit atau 10 bit. Suatu ADC dengan ketelitian 8 bit dapat mengkonversi suatu voltase yang dibaca ke suatu bilangan asli dari 0 sampai $255(2^8 - 1)$. ADC dengan ketelitian 10 bit dapat mengkonversi voltase masukan dari 0 sampai $1023(2^{10} - 1)$.

Untuk melakukan pembandingan ADC menggunakan berbagai macam clock untuk *prescaler*. Clock yang memiliki frekuensi berbeda-beda ini dihasilkan dari clock mikrokontroler. Untuk melakukan suatu konversi ADC memerlukan 13 siklus (*cycle*) clock mikrokontroler. Sebelum melakukan konversi dibutuhkan 25 siklus untuk melakukan inisialisasi. Diagram konversi data menggunakan fitur ADC dapat dilihat pada gambar 4-3



Gambar 4-3. Diagram konversi data menggunakan fitur ADC.

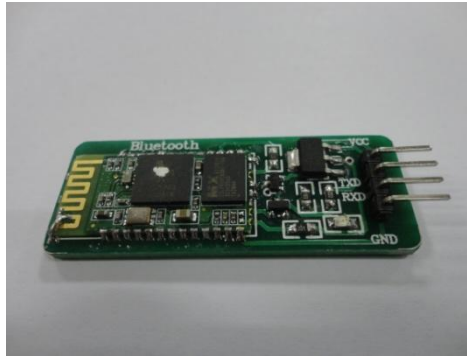
Setelah melakukan konversi data analog ke data digital tugas kedua yang harus dilakukan mikrokontroller adalah mengirim data digital tersebut ke perangkat cerdas. Mikrokontroller dapat melakukan fungsi tersebut karena memiliki modul komunikasi data serial (USART). Diagram format pengiriman data serial dapat dilihat pada gambar 4-4. Dalam protokol serial ada berbagai macam parameter, yaitu jumlah bit data dalam satu transmisi, *stopbit*, *paritybit*, dan kecepatan transmisi(baudrate). Dalam prototipe ini nilai parameter yang digunakan adalah 8 bit data, 1 *stopbit*, *no parity bit*, dan baudrate 9600.



Gambar 4-4. Format pengiriman dan penerimaan data serial.

4.1.3. Bluetooth

Komponen ini merupakan komponen yang bertugas sebagai antarmuka pengiriman data. Alat ini berfungsi mengubah data yang dikirim dengan protokol USART melalui gelombang radio (protokol *bluetooth*). Dengan menggunakan alat ini, mikrokontroller dapat mengirimkan data ke perangkat *smartphone* mengingat *smartphone* tidak menyediakan protokol komunikasi data secara kabel dengan mikrokontroller. Modul *bluetooth* to serial converter yang digunakan pada prototipe ini dapat dilihat pada gambar 4-5.



Gambar 4-5. Modul *bluetooth* to serial converter.

4.1.4. Kabel serial

Komponen ini merupakan komponen yang dapat digunakan untuk mengirimkan data dari mikrokontroller ke PC. Komponen ini diperlukan pada PC yang tidak memiliki modul perangkat keras *bluetooth*. Kabel serial yang digunakan pada *prototype* ini merupakan kabel serial yang bisa dilakukan *switch mode* menjadi *downloader*. Kabel ini bisa digunakan untuk konektor pengiriman data melalui serial dan juga bisa digunakan untuk mengunduh program dari PC ke mikrokontroller. Kabel serial yang digunakan pada prototipe ini dapat dilihat pada gambar 4-6.



Gambar 4-6. Kabel serial.

4.1.5. Konektor Elektroda

Komponen ini bertugas menghubungkan pin masukan pada EKG dengan elektroda yang di pasang pada badan pasien. Konektor elektroda pada modul perangkat keras ini dapat dilihat pada gambar 4-7. Ujung dari konektor ini dijepitkan pada elektroda yang dipasang pada anggota badan pasien.



Gambar 4-7. Konektor elektroda, dari kiri kekanan terhubung dengan elektroda (L, GND, dan R).

4.1.6. Baterai

Komponen ini bertugas untuk memberikan tegangan masukan pada modul perangkat keras, baik untuk EKG maupun untuk komponen-komponen lainnya. Baterai yang digunakan pada modul tersebut adalah dua Baterai GP 200 mAh, 8,4 volt. Baterai ini merupakan Baterai yang bisa di *recharge*, sehingga nantinya pengguna nantinya tidak perlu membeli Baterai berulang kali. Baterai beserta charger yang dipakai pada prototipe ini dapat dilihat pada gambar 4-8.

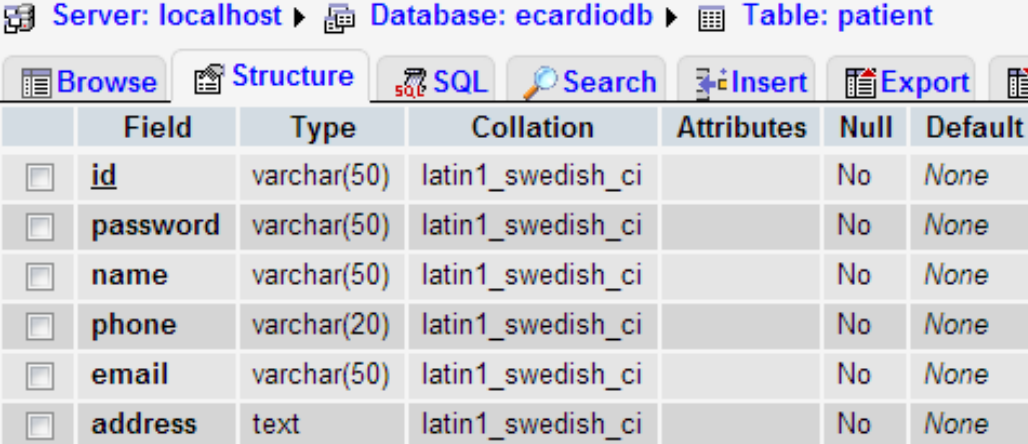


Gambar 4-8. Baterai beserta charger yang digunakan pada *prototype*.

4.2. Implementasi Database

Setelah melakukan desain, *database* kemudian diimplementasikan. Yang dimaksud dengan implemntasi *database* adalah pembuatan *database* pada *server* dan juga *smartphone* Android sebagai versi *lite*. Dari hasil pemetaan ER *diagram* yang telah dibuat, pada *database* dibuat lima tabel berdasarkan kebutuhan sistem. Kelima tabel tersebut adalah tabel *patient*, *doctor*, *hospital*, *history* dan *affiliation*. Database manajemnt sistem (DBMS) yang digunakan pada *prototype* ini adalah MySQL.

Tabel *patient* merupakan tabel yang menyimpan data pasien pengguna sistem pendeteksian dan *monitoring* penyakit jantung ini. Pada implementasi *database* yang dibangun di *server* tabel ini terdiri dari enam kolom (*field*). Keenam kolom tersebut adalah id pasien, password pasien, nama pasien, nomor telepon pasien, email pasien dan alamat pasien. Pada *database* yang diimplementasikan di *server* ini data data tanggal lahir, berat badan, tinggi badan , golongan darah tidak disimpan, karena informasi yang disimpan di *database* yang terletak di *server* ini sementara fokus pada informasi – informasi yang berhubungan dengan komunikasi antara pasien dan dokter. Deskripsi tabel *patient* pada *server* dapat dilihat pada gambar 4-9.



	Field	Type	Collation	Attributes	Null	Default
<input type="checkbox"/>	<u>id</u>	varchar(50)	latin1_swedish_ci		No	None
<input type="checkbox"/>	password	varchar(50)	latin1_swedish_ci		No	None
<input type="checkbox"/>	name	varchar(50)	latin1_swedish_ci		No	None
<input type="checkbox"/>	phone	varchar(20)	latin1_swedish_ci		No	None
<input type="checkbox"/>	email	varchar(50)	latin1_swedish_ci		No	None
<input type="checkbox"/>	address	text	latin1_swedish_ci		No	None

Gambar 4-9. Tabel *patient* pada *database*.

Tabel *doctor* merupakan tabel yang menyimpan data dokter pada sistem pendeteksian dan *monitoring* penyakit jantung ini. Pada implementasi *database* yang dibangun di *server*, tabel ini terdiri dari enam kolom (*field*). Keenam kolom tersebut adalah id dokter, password dokter, nama dokter, nomor telepon dokter, email dokter dan alamat dokter. Pada *database* yang diimplementasikan di *server* ini memang tidak ada kolom tanggal lahir, berat badan, tinggi badan golongan darah, karena informasi tersebut memang dirancang untuk tidak disimpan pada *database* dokter. Deskripsi tabel *doctor* pada *server* dapat dilihat pada gambar 4-10.

Server: localhost ▶ Database: ecardiodb ▶ Table: doctor

	Field	Type	Collation	Attributes	Null	Default	Extra
<input type="checkbox"/>	<u>id</u>	varchar(50)	latin1_swedish_ci		No	None	
<input type="checkbox"/>	password	varchar(50)	latin1_swedish_ci		No	None	
<input type="checkbox"/>	name	varchar(50)	latin1_swedish_ci		No	None	
<input type="checkbox"/>	phone	varchar(20)	latin1_swedish_ci		No	None	
<input type="checkbox"/>	email	varchar(50)	latin1_swedish_ci		No	None	
<input type="checkbox"/>	address	text	latin1_swedish_ci		No	None	

Gambar 4-10. Tabel *doctor* pada *database*.

Tabel *hospital* merupakan tabel yang menyimpan data rumah sakit dan klinik pada sistem pendeteksian dan *monitoring* penyakit jantung ini. Pada implementasi *database* yang dibangun di *server*, tabel ini terdiri dari lima kolom (*field*). Kelima kolom tersebut adalah id rumah sakit atau klinik, nama rumah sakit atau klinik, nomor telepon dan alamat rumah sakit atau klinik tersebut. Informasi rumah sakit dan klinik ini nantinya digunakan sebagai rujukan oleh dokter kepada pasien untuk memerikasakan kondisi kesehatan jantungnya lebih lanjut. Deskripsi tabel *hospital* pada *server* dapat dilihat pada gambar 4-11.

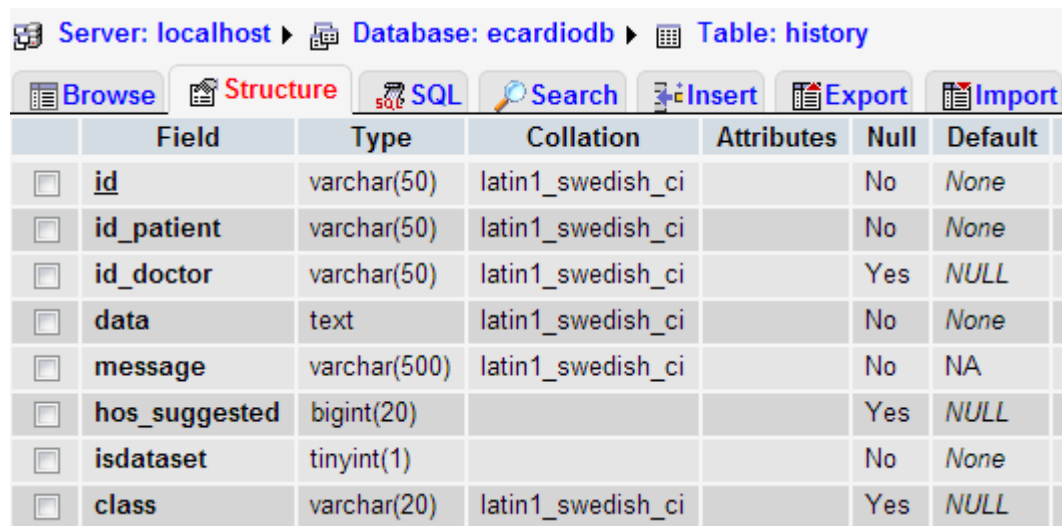
Server: localhost ▶ Database: ecardiodb ▶ Table: hospital

	Field	Type	Collation	Attributes	Null	Default	Extra
<input type="checkbox"/>	<u>id</u>	bigint(20)		UNSIGNED	No	None	auto_increment
<input type="checkbox"/>	name	varchar(50)	latin1_swedish_ci		No	None	
<input type="checkbox"/>	phone	varchar(20)	latin1_swedish_ci		No	None	
<input type="checkbox"/>	address	text	latin1_swedish_ci		No	None	

Gambar 4-11. Tabel *hospital* pada *database*.

Tabel *history* merupakan tabel yang menyimpan data detak jantung pengguna (pasien) dalam sistem pendeteksian dan *monitoring* penyakit jantung ini. Pada implementasi *database* yang dibangun di *server*, tabel ini terdiri dari delapan kolom (*field*). Delapan kolom pada tabel tersebut adalah, id data, kemudian id pasien yang memiliki data tersebut, kemudian id dokter yang nantinya memverifikasi data, kolom data yang berisi data detak jantung, pesan verifikasi, rumah sakit rujukan dan kolom untuk keperluan dataset. Sewaktu pertama kali data disimpan

dalam *database* nilai id dokter, rumah sakit rujukan dan kelas dataset tidak ada isinya (kosong). Sebenarnya untuk kolom pesan, pertama kali data disimpan nilainya juga kosong. Akan tetapi untuk memudahkan proses pengelolaan, nilai *default* atau awalnya adalah “NA”. Deskripsi tabel *history* pada *server* dapat dilihat pada gambar 4-12.



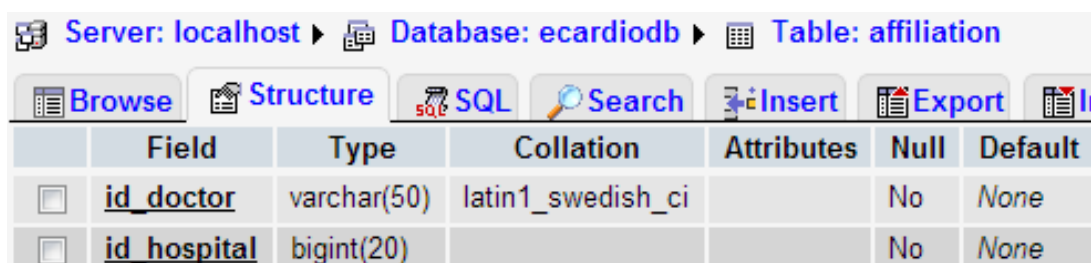
Server: localhost ▶ Database: ecardiodb ▶ Table: history

Buttons: Browse, Structure, SQL, Search, Insert, Export, Import

	Field	Type	Collation	Attributes	Null	Default
<input type="checkbox"/>	<u>id</u>	varchar(50)	latin1_swedish_ci		No	None
<input type="checkbox"/>	<u>id_patient</u>	varchar(50)	latin1_swedish_ci		No	None
<input type="checkbox"/>	<u>id_doctor</u>	varchar(50)	latin1_swedish_ci		Yes	NULL
<input type="checkbox"/>	<u>data</u>	text	latin1_swedish_ci		No	None
<input type="checkbox"/>	<u>message</u>	varchar(500)	latin1_swedish_ci		No	NA
<input type="checkbox"/>	<u>hos_suggested</u>	bigint(20)			Yes	NULL
<input type="checkbox"/>	<u>isdataset</u>	tinyint(1)			No	None
<input type="checkbox"/>	<u>class</u>	varchar(20)	latin1_swedish_ci		Yes	NULL

Gambar 4-12. Tabel *hospital* pada *database*.

Tabel *affiliation* merupakan tabel yang menyimpan data afiliasi dokter dalam sistem pendeteksian dan *monitoring* penyakit jantung ini. Pada implementasi *database* yang dibangun di *server*, tabel ini hanya memiliki dua kolom (*field*). Kedua kolom tersebut adalah id dokter, dan id rumah sakit atau klinik yang menjadi afiliasinya. Dalam implementasi hal ini memang tidak ada field lain, karena data yang dibutuhkan sudah tersimpan di tabel lain (*doctor* dan *hospital*). Sedangkan data yang disimpan pada tabel ini adalah id yang juga *primary key* dari kedua tabel tersebut. Deskripsi tabel *affiliation* pada *server* dapat dilihat pada gambar 4-13.



Server: localhost ▶ Database: ecardiodb ▶ Table: affiliation

Buttons: Browse, Structure, SQL, Search, Insert, Export, Import

	Field	Type	Collation	Attributes	Null	Default
<input type="checkbox"/>	<u>id_doctor</u>	varchar(50)	latin1_swedish_ci		No	None
<input type="checkbox"/>	<u>id_hospital</u>	bigint(20)			No	None

Gambar 4-13. Tabel *affiliation* pada *database*.

Selain diimplementasikan dalam versi *online*, database juga diimplementasikan dalam versi *lite*, yaitu dalam versi *smartphone* Android. Pada versi *lite database*, tabel yang digunakan sama. *Database* versi *lite* juga menggunakan lima table seperti versi *online*. Hanya saja ada sedikit perbedaan dari sisi struktur secara utuh. Setelah dilakukan percobaan, *database* versi *lite* ini tidak mengijinkan adanya dua table dalam satu nama *database*. Dengan demikian, ada lima *database* yang masing-masing terdiri dari satu table. Dalam implementasi *database* versi *lite*, dibuat suatu kelas sebagai penghubung ke *database* asli. Contoh kelas yang mewakili *database* pasien dapat dilihat pada gambar 4-14.

```
public class PatientDBHelper extends SQLiteOpenHelper {

    public static final String DB_NAME = "dbpatient";
    public static final int DB_VERSION = 1;
    public static final String TABLE_NAME = "patient";

    private static final String DB_CREATE =
        "CREATE TABLE " + TABLE_NAME +

        "(id TEXT PRIMARY KEY, " +
        "password TEXT NOT NULL, " +
        "name TEXT NOT NULL, " +
        "phone TEXT NOT NULL, " +
        "email TEXT NOT NULL, " +
        "address TEXT NOT NULL, " +

        "dateOfBirth TEXT NOT NULL, " +
        "bloodType TEXT NOT NULL, " +
        "weight INTEGER NOT NULL, " +
        "height INTEGER NOT NULL, " +
        "gender TEXT NOT NULL);";

    /**
     *
     * @param context bertipe Context
     */
    public PatientDBHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }
}
```

Gambar 4-14. Contoh Kode Database Versi Lite.

4.3. Implemetasi Aplikasi Web pada Server

Komponen kedua pada *server* adalah aplikasi *web*. Aplikasi ini bertugas memberikan layanan yang diminta oleh pengguna melalui aplikasi E-Cardio. Pada bagian perancangan sudah

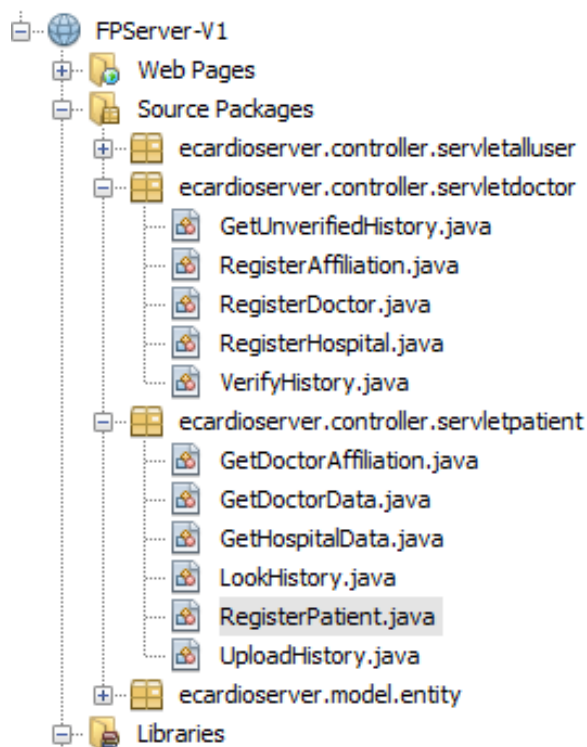
dijelaskan bahwa aplikasi *web* akan dibuat secara modular sesuai dengan fungsi dari masing – masing modul. Setiap modul memiliki fungsi tersendiri dan alamat URL tersendiri. Untuk mengimplementasikan hal itu digunakan servlet pada *web* aplikasi. Servlet merupakan file java yang dapat melayani *request* pada aplikasi *web*. Servlet menerima parameter berupa *request* dan memberikan keluaran berupa response. *Request* menampung berbagai data-data yang diberikan oleh *client* kepada *server*. Contoh kode program suatu servlet dapat dilihat pada gambar 4-15.

```
/**
 * @author anwar
 */
@WebServlet(name = "RegisterPatient", urlPatterns = {"/RegisterPatient"})
public class RegisterPatient extends HttpServlet {
    /**
     * Processes requests for both HTTP
     * <code>GET</code> and
     * <code>POST</code> methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse
    response)
        throws ServletException, IOException {
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        try {
            //Ambil parameter
            //Validasi input parameter
            //Simpan data pasien pada database

        } finally {
            out.close();
        }
    }
}
```

Gambar 4-15. Contoh kode program servlet.

Sesuai dengan rancangan yang dilakuakn, pada apliaksi *web* dibuat suatu *package* untuk memisahkan fungsional dari servlet. Dengan demikian proses implementasi leih mudah dilakukan. Secara fungsional, ada servlet yang berfungsi untuk melayani *request* dari pasien, dan ada servlet yang melayani *request* dari dokter saja. Oleh sebab itu dilakukan pembagian servlet ke *package-package* sesuai fungsionalnya. Arsitektur pengelolaan servlet pada aplikasi *web* dapat dilihat pada gambar 4-16.



Gambar 4-16. Arsitektur servlet pada server.

Dalam implementasi aplikasi *web* yang dilaksanakan, setiap servlet memiliki dua parameter, yaitu *request* dan *response*. *Request* digunakan untuk mengambil parameter parameter yang diberikan oleh *client* saat meminta layanan dari aplikasi *web* ini. Sedangkan *response* adalah parameter yang digunakan untuk menyimpan objek parameter HTTP saat servlet mengembalikan permintaan dari *client* tadi. Dalam implementasi yang dilakukan, parameter – parameter yang diminta oleh pengguna disisipkan pada objek *request* tersebut. Dengan demikian kriteria data yang diminta oleh pengguna dapat diketahui oleh *server*. Keterangan lengkap mengenai parameter yang digunakan oleh setiap servlet pada aplikasi *web* ini dapat dilihat pada tabel 4-1.

Tabel 4-1. Parameter *Request* dan Keluaran Servlet

No.	Servlet	Parameter <i>request</i>	output
1	/RegisterPatient	<ul style="list-style-type: none"> - Id - Name - Phone - Email - address 	Berhasil / pesan gagal
2	/RegisterDoctor	<ul style="list-style-type: none"> - Id - Name 	Berhasil / pesan gagal

		<ul style="list-style-type: none"> - Phone - Email - address 	
3	<i>/UploadHistory</i>	<ul style="list-style-type: none"> - id - id_patient - pass_patient - data 	Berhasil / pesan gagal
4	<i>/LookHistory</i>	<ul style="list-style-type: none"> - id - id_patient - pass_patinet 	Data detak jantung lengkap sesuai id / pesan gagal
5	<i>/GetDoctorData</i>	<ul style="list-style-type: none"> - id_doctor - id_patient - pass_patient 	Data dokter lengkap / pesan gagal
6	<i>/GetHospitalData</i>	<ul style="list-style-type: none"> - id_hospital - id_patient - pass_patient 	Data rumah sakit atau klinik lengkap / pesan gagal
7	<i>/GetUnverifiedHistory</i>	<ul style="list-style-type: none"> - id_doctor - pass_doctor - number 	List data detak jantung yang belum terverifikasi / pesan gagal
8	<i>/VerifyHistory</i>	<ul style="list-style-type: none"> - id_doctor - pass_doctor - id - message - hos_suggested 	Berhasil / pesan gagal
9	<i>/RegisterHospital</i>	<ul style="list-style-type: none"> - id_doctor - pass_doctor - name - phone - address 	Berhasil / pesan gagal
10	<i>/RegisterAffiliation</i>	<ul style="list-style-type: none"> - id_doctor - pass_doctor - id_hospital 	Berhasil / pesan gagal
11	<i>/GetDoctorAffiliation</i>	<ul style="list-style-type: none"> - id_patient - pass_patient - id_doctor 	Data afiliasi dokter yang diminta / pesan gagal

Seperti yang dijelaskan sebelumnya bahwa setiap servlet akan menjalankan suatu fungsi tertentu. Setelah mencari parameter-parameter yang dia butuhkan, setiap servlet akan melakukan pengolahan data. Setiap servlet melakukan pengolahan data yang berbeda – beda. Servlet */RegisterPatient* misalnya, setelah mendapatkan nilai dari parameter – parameter yang

dibutuhkan, dia akan melakukan proses penyimpanan data pasien pada *database*. Contoh implementasi kode pada servlet untuk menyimpan data pasien pada *database* dapat dilihat pada gambar 4-17.

```
protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
    response.setContentType("text/plain");
    PrintWriter out = response.getWriter();
    try {
        String id = request.getParameter("id");
        String password = request.getParameter("password");
        String name = request.getParameter("name");
        String phone = request.getParameter("phone");
        String email = request.getParameter("email");
        String address = request.getParameter("address");

        if(id!= null && password!= null && name !=null && phone!=null
            && email != null && address != null) {

            EntityManagerFactory emf =
Persistence.createEntityManagerFactory("FPServer-V1PU");
            EntityManager em = emf.createEntityManager();

            Patient p = em.find(Patient.class, id);

            if(p==null) {
                EntityTransaction tx = em.getTransaction();
                Patient p2 = new Patient(id, password, name, phone, email,
address);
                tx.begin();
                em.persist(p2);
                tx.commit();
                out.println("berhasil");
            }
            else {
                out.println("Akun sudah terdaftar");
                out.println("Silakan gunakan akun yang berbeda");
            }
        }
        else {
            out.println("Data tidak lengkap");
            out.println("Silakan lengkapi data anda");
        }
        /* TODO output your page here. You may use following sample code. */
    } finally {
        out.close();
    }
}
```

Gambar 4-17. Contoh kode program servlet

Untuk memudahkan pengelolaan data pada *database* dibuat suatu *object relational model* dari *database* ke kelas java (java bean). Pada aplikasi *web* ini dibuat suatu kelas entitas yang mencerminkan tabel pada *database*. Objek dari kelas tersebut merepresentasikan suatu *record* (baris data) yang ada dalam *database*. Dengan menggunakan fitur ini maka proses manajemen data pada *database* lebih mudah dilakukan. Dari lima tabel yang ada pada *database* yang digunakan pada *prototype* ini dihasilkan enam kelas entitas. Kelas Patient.java merepresentasikan tabel patient, kelas Doctor.java merepresentasikan tabel doctor, kelas History.java merepresentasikan tabel history, kelas Hospital.java merepresentasikan tabel hospital, dan kelas Affiliation.java merepresentasikan tabel affiliation. Selain itu ada tambahan kelas AffiliationPK. AffiliationPK merupakan kelas yang menjadi objek *primary key* dari kelas Affiliation. Secara fisik ada penambahan anotasi tertentu pada kelas entitas tersebut. Potongan kode sumber pada kelas entitas Patient.java dapat dilihat pada gambar 4-18.

```
@Entity
@Tabel(name = "patient")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Patient.findAll", query = "SELECT p FROM Patient p"),
    @NamedQuery(name = "Patient.findById", query = "SELECT p FROM Patient p WHERE
p.id = :id"),
    @NamedQuery(name = "Patient.findById", query = "SELECT p FROM Patient p
WHERE p.id = :id"),
    @NamedQuery(name = "Patient.findByName", query = "SELECT p FROM Patient p
WHERE p.name = :name"),
    @NamedQuery(name = "Patient.findByPhone", query = "SELECT p FROM Patient p
WHERE p.phone = :phone"),
    @NamedQuery(name = "Patient.findByEmail", query = "SELECT p FROM Patient p
WHERE p.email = :email"))
public class Patient implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 50)
    @Column(name = "id")
    private String id;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 50)
    @Column(name = "password")
    private String password;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 50)
    @Column(name = "name")
    private String name;
    // @Pattern(regexp="^(\\d{3})\\d{3}[- ]?(\\d{3})[- ]?(\\d{4})$",
message="Invalid phone/fax format, should be as xxx-xxx-xxxx")//if the field
```

```
contains phone or fax number consider using this annotation to enforce field
validation
}
```

Gambar 4-18. Contoh kode program servlet

4.4. Implementasi Modifikasi Aplikasi E-Cardio

Seperti yang dijelaskan pada bab randangan penelitian, bahwa dalam implementasi *prototype* sistem ini salah satunya akan diimplementasikan modifikasi aplikasi E-Cardio Seluruh *use case* yang dirancang sudah diimplementasikan dalam bentuk *aplikasi* siap pakai. Keterangan mengenai modifikasi akan dijelaskan pada penjelasan selanjutnya.

4.4.1. Implementasi Use Case Spesification : Mendaftarkan akun

Use case mendaftar akun merupakan *use case* yang dimodifikasi dari fitur aplikasi yang sudah ada sebelumnya. *Use case* ini dibuat dengan tujuan untuk mengelola data pengguna aplikasi. Perbedaan dari versi sebelumnya adalah pada pembedaan jenis pengguna dan data pengguna yang disimpan. Dalam aplikasi versi 3.0 ini, ada pembedaan *role* pengguna antara dokter dan pasien. Selain itu, data nomor telepon, email, dan alamat pasien juga disimpan di *database*. Setelah mengisi dan menekan tombol *save* Implementasi *use case* mendaftar akun dapat dilihat pada gambar 4-19. Untuk mendaftar akun Pertama kali pengguna harus memilih menu *user management*. Setelah sistem membuka layar seperti pada gambar 4-19 (a), pengguna tinggal menekan tombol *create new user*. Selanjutnya, aplikasi akan menyediakan *form* isian untuk mendaftar akun.

(a)

(b)

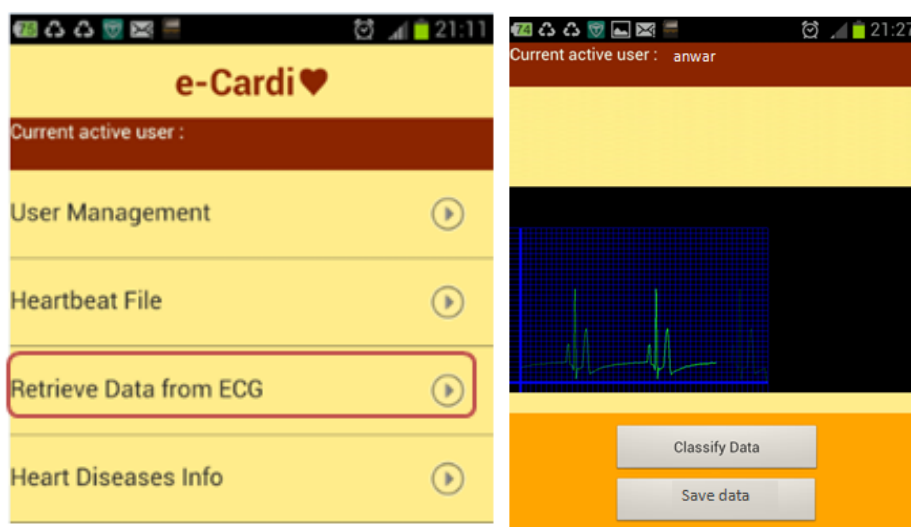
(c)



Gambar 4-19. Implementasi *use case* mendaftarkan akun pada aplikasi e-cardio (a) Tampilan awal (b)-(e) Form untuk mendaftarkan akun.

4.4.2. Implementasi Use Case Specification : Menyimpan *History* Detak Jantung

Use case Menyimpan *history* detak jantung merupakan *use case* yang dimodifikasi ulang dari fitur yang sudah ada sebelumnya. Perbedaan dari versi seelumnya adalah bahwa dalam versi baru ini data detak jantung disimpan di *database*, bukan di *file*. Implementasi *use case* menyimpan *history* detak jantung dapat dilihat pada gambar 4-20. Pertama kali, pengguna harus memilih menu untuk mengambil data dari sensor EKG seperti pada gambar 4-20 (a). Setelah data dari *sensor ECG* ditampilkan pada aplikasi (seperti gambar 4-20 (b)), selanjutnya pengguna tinggal menekan tombol *save data*.



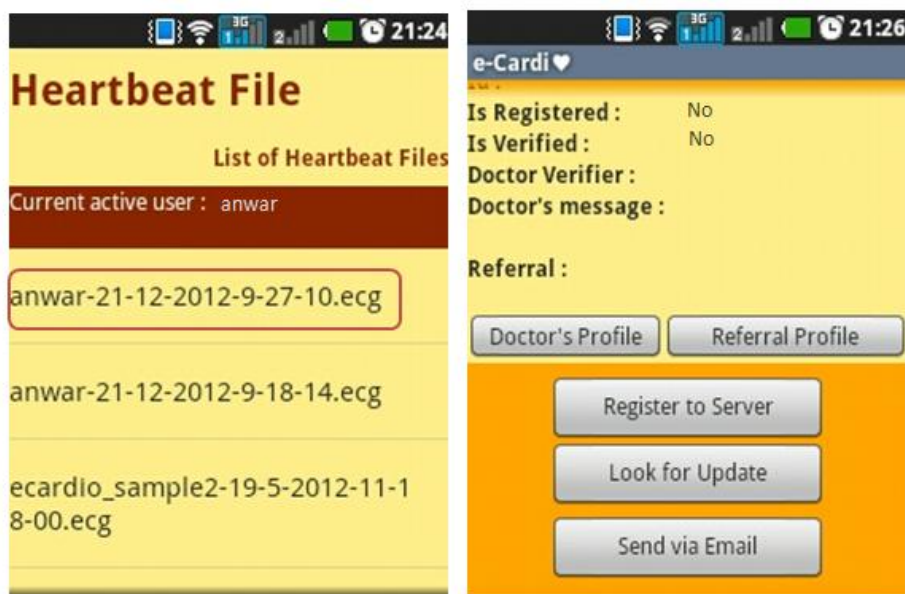
(a)

(b)

Gambar 4-20. Implementasi *use case* menyimpan *history* detak jantung (a) Tampilan awal (b)-(e) Tampilan saat sinyal detak jantung sudah divisualisasikan.

4.4.3. Implementasi Use Case Spesification : Melihat *History* Detak Jantung

Use case Melihat *history* detak jantung merupakan *use case* yang sudah diimplementasikan sebelumnya. Akan tetapi, dalam pelaksanaan penelitian ini, *use case* tersebut dimodifikasi. Modifikasi yang dilakukan adalah setelah *history* detak jantung ditampilkan secara detail, akan ada status apakah *history* tersebut sudah diverifikasi atau belum, berikut pesan verifikasi, serta dokter yang memverifikasi. Jika belum mendapat verifikasi, data *history* tersebut bisa didaftarkan / dikirimkan ke *server*. Implementasi *use case* melihat *history* detak jantung dapat dilihat pada gambar 4-21.

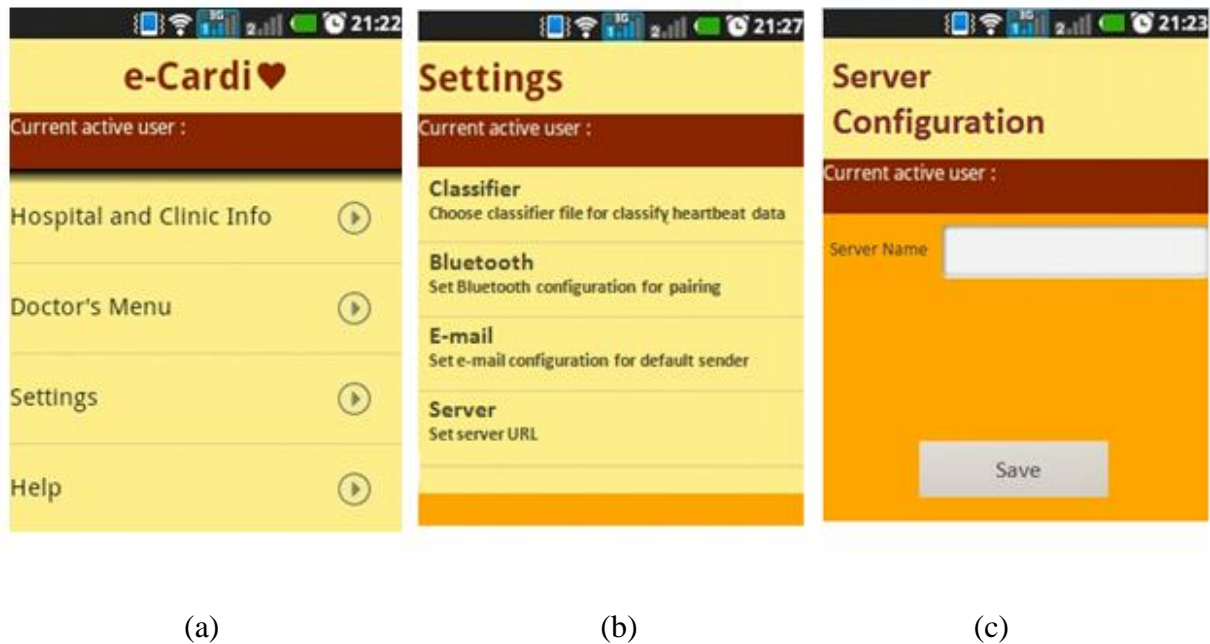


Gambar 4-21. Implementasi *use case* melihat *history* detak jantung (a) Tampilan saat memilih salah satu data(b)-(e) Tampilan detail data detak jantung.

4.4.4. Implementasi Use Case Spesification : Setting

Use case setting merupakan *use case* yang sudah diimplementasikan sebelumnya. Akan tetapi, dalam pelaksanaan penelitian ini, *use case* tersebut dimodifikasi. Modifikasi yang

dilakukan adalah dengan ditambahkan pengaturan untuk *server*. Implementasi *use case* setting dapat dilihat pada gambar 4-22. Untuk melakukan pengaturan *server*, pertama kali pengguna harus memilih menu *setting* pada menu utama, seperti gambar 4-22 (a). Setelah muncul pilihan *setting* seperti pada gambar 4-22 (b), pengguna harus memilih *sub menu server*. Selanjutnya pengguna tinggal memasukkan nama *server* lalu menyimpannya seperti pada Gambar 4-22 (c).



Gambar 4-22. Implementasi *use case* setting (a) Tampilan menu utama (b) Tampilan pilihan setting (c) Tampilan Pengaturan server.

4.4.5. Implementasi Use Case Spesification : Mengirimkan *History* Detak Jantung Ke *Server*

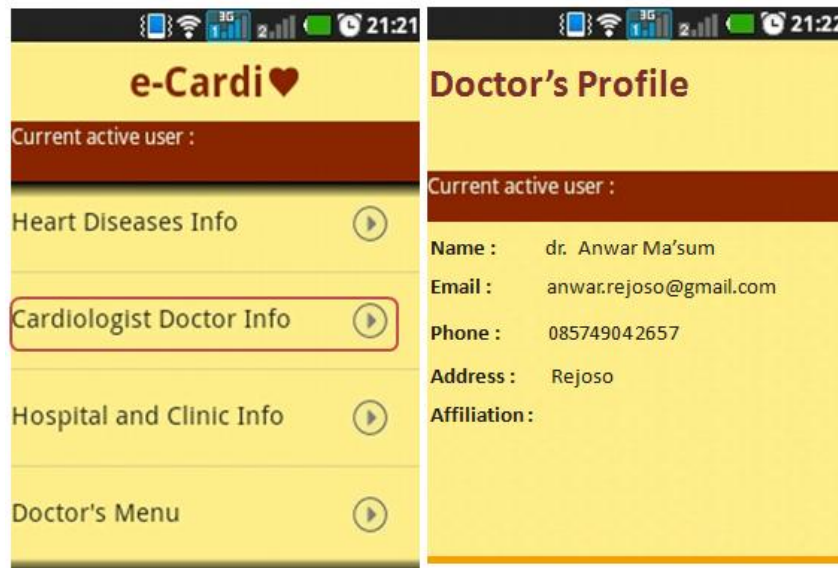
Use case mengirimkan *history* detak jantung ke *server* merupakan *use case* yang belum diimplementasikan sebelumnya. *Use case* ini merupakan lanjutan dari *use case* melihat *history* detak jantung. Setelah data *history* detak jantung ditampilkan secara detail, selanjutnya data tersebut dapat dikirimkan ke *server* jika belum didaftarkan sebelumnya. Implementasi *use case* mengirimkan *history* detak jantung ke *server* dapat dilihat pada gambar 4-23.



Gambar 4-23. Implementasi *use case* mendaftarkan *history* detak jantung ke *server*.

4.4.6. Implementasi Use Case Spesification : Melihat Informasi Dokter

Use case melihat informasi dokter merupakan *use case* baru diimplementasikan pada *prototype* sistem versi ketiga ini. *Use case* ini diimplementasikan menjadi suatu menu tersendiri pada aplikasi. Ilustrasi implemetasi dari *use case* ini dapat dilihat pada gambar 4-24. Pertama kali pengguna memilih menu *cardiologist doctor* info pada menu utama seperti gambar 4-24 (a). Selanjutnya pengguna bisa memilih salah satu data di antara *list* data dokter yang ada. Dengan demikian, aplikasi akan menampilkan data lengkap tentang dokter yang diminta seperti gambar 4-24 (b).



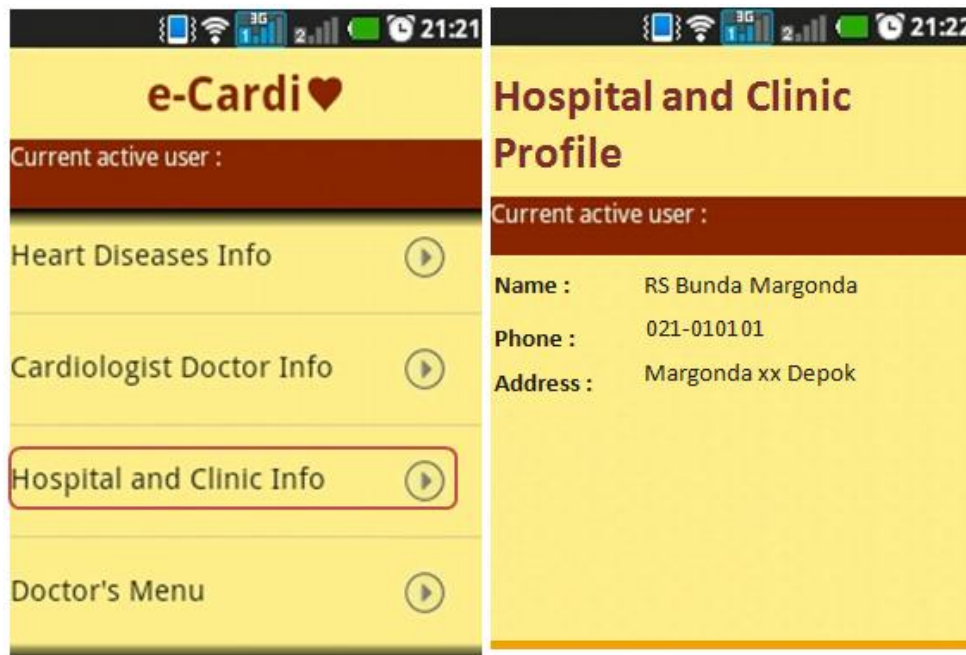
(a)

(b)

Gambar 4-24. Implementasi *use case* melihat informasi dokter. (a) Tampilan menu awal. (b) Tampilan data dokter

4.4.7. Implementasi Use Case Spesification : Melihat Informasi Rumah Sakit dan Klinik.

Seperti halnya dengan *use case* melihat informasi dokter, *use case* melihat informasi rumah sakit dan klinik merupakan *use case* baru diimplementasikan pada *prototype* sistem versi ketiga ini. *Use case* ini juga diimplementasikan dalam suatu menu tersendiri. Ilustrasi implementasi *use case* ini dapat dilihat pada gambar 4-25. Pertama kali pengguna memilih menu untuk melihat data rumah sakit dan klinik pada menu utama, seperti gambar 4-25 (a). Selanjutnya, pengguna tinggal memilih salah satu data rumah sakit dari list yang ada. Setelah aplikasi menampilkan data detail tentang rumah sakit atau klinik, tampilan aplikasi akan tampak seperti gambar 4-25 (b).



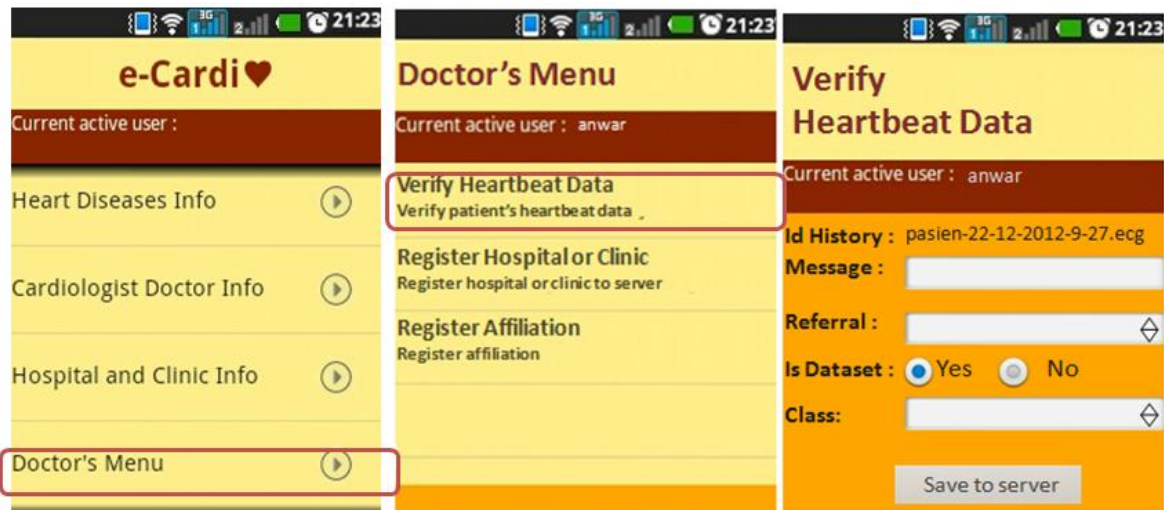
(a)

(b)

Gambar 4-25. Implementasi *use case* melihat informasi rumah sakit dan klinik. (a) Tampilan menu awal. (b) Tampilan data rumah sakit atau klinik

4.4.8. Implementasi Use Case Spesifikasi : Melakukan Verifikasi Data Detak Jantung

Use case melakukan verifikasi data detak jantung merupakan *use case* yang paling penting dalam sistem terintegrasi ini. *Use case* ini diimplementasikan dalam suatu menu khusus. Dalam aplikasi ini dibuat satu menu khusus untuk dokter, yaitu melakukan verifikasi detak jantung, mendaftarkan rumah sakit dan klinik, dan mendaftarkan afiliasi dokter. Implementasi *use case* ini diilustrasikan pada gambar 4-26. Pertama kali dokter memilih menu khusus dokter seperti gambar bagian (a). Selanjutnya, dokter memilih sub menu untuk verifikasi data detak jantung seperti pada gambar (b). Selanjutnya dokter memilih salah satu data detak jantung dari list yang ada untuk dia verifikasi. Selanjutnya dokter bisa mengisi *form* isian verifikasi data detak jantung untuk dikirimkan ke *server* seperti gambar bagian (c).



(a)

(b)

(c)

Gambar 4-26. Implementasi verifikasi data detak jantung (a) Menu utama (b) Sub menu khusus dokter (c) Isian form untuk verifikasi data detak jantung

4.4.9. Implementasi Use Case Spesification : Mendaftarkan Data Rumah Sakit dan Klinik

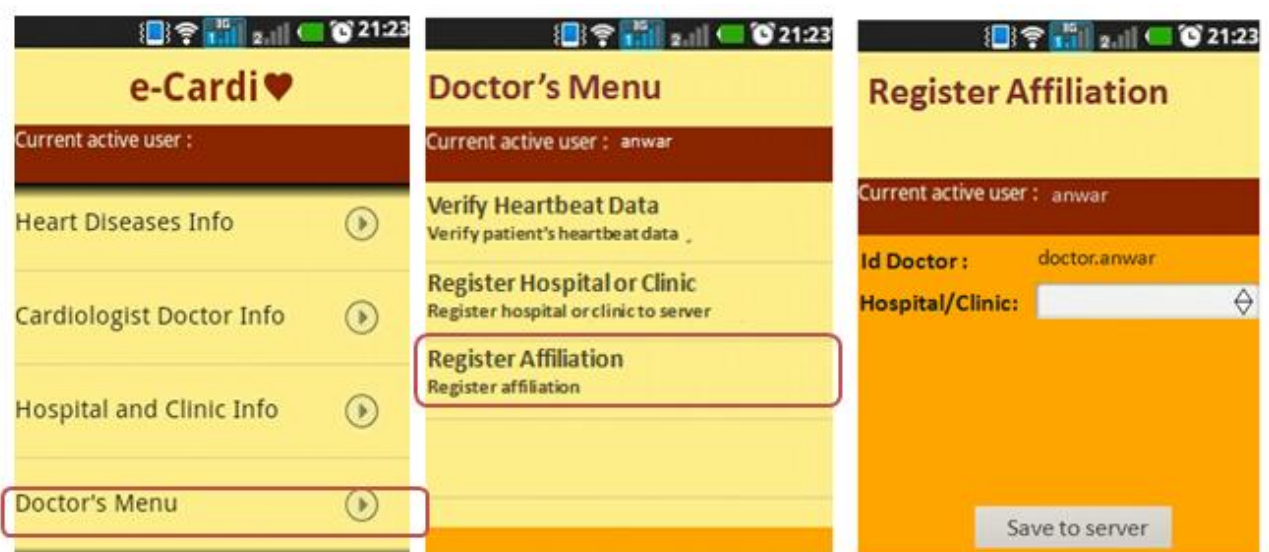
Use case mendaftarkan data rumah sakit dan klinik merupakan *use case* yang hanya dapat dilakukan oleh dokter. *Use case* ini diimplementasikan sebagai sub menu yang hanya dapat dilakukan oleh dokter. Ilustrasi implementasi *use case* ini dapat dilihat pada Gambar 4-27. Tahap pertama dan kedua sama seperti pada *use case* verifikasi detak jantung, yaitu membuka menu khusus dokter, dan memilih sub menu untuk *use case* ini. Selanjutnya, dokter tinggal mengisi isian *form* untuk mendaftarkan data rumah sakit dan klinik di *database*



Gambar 4-27. Implementasi mendaftarkan data rumah sakit dan klinik (a) Menu utama (b) Sub menu khusus dokter (c) Isian *form* untuk mendaftarkan rumah sakit.

4.4.10. Implementasi Use Case Specification : Mendaftarkan Afiliasi Dokter

Use case mendaftarkan afiliasi dokter merupakan menu khusus dokter yang terakhir dalam aplikasi ini. Dua langkah pertama untuk menggunakan menu ini sama dengan menu khusus dokter yang lain. Selanjutnya untuk langkah ketiga, dokter tinggal memilih rumah sakit atau klinik yang menjadi afliasinya. Setelah itu dokter tinggal menekan tombol untuk menyimpan di server. Ilustrasi implementasi *use case* mendaftarkan afiliasi dokter dapat dilihat pada gambar 4-28.



Gambar 4-28. Implementasi mendaftarkan afiliasi dokter (a) Menu utama (b) Sub menu khusus dokter (c) Isian form untuk mendaftarkan afiliasi dokter

4.5. Implementasi Pembuatan Restful Webservice

Pada sistem ini dapat juga dibuat restful web service agar data dapat pada database bisa diakses dari server lain. Restful web service dalam implementasi prototype ini berhasil dilakukan. Meskipun demikian ada beberapa perbedaan dengan rancangan. Pada rancangan pembuatan, web service akan dibuat menjadi salah satu modul dari aplikasi web yang dibangun di server. Akan tetapi, setelah diimplementasikan ada berbagai masalah yang muncul. Masalah tersebut dikarenakan penggunaan JPA dalam waktu yang bersamaan antara entity class dan web service. Oleh sebab itu dilakukan modifikasi pada arsitektur server. Pada server dibuat dua aplikasi web. Aplikasi pertama adalah aplikasi yang melayani request untuk client. Aplikasi kedua adalah aplikasi yang dibuat secara khusus memuat modul web service. Dengan cara ini, kedua aplikasi dapat berjalan bersamaan dan simultan tanpa ada error. Contoh keluaran dari web service yang dibuat pada prototype ini dalam bentuk sintaks xml dapat dilihat pada gambar 4-29.

```
<?xml version="1.0" encoding="UTF-8"?>
  <histories>
    <history>
      <data>snajdbakjbcakjcbcsjbvsjbvj</data>
      <hosSuggested>1</hosSuggested>
      <id>lalalalal</id>
      <idPatient>a</idPatient>
      <isdataset>>false</isdataset>
      <message>sakit berat</message>
    </history>
    <history>
      <data>snajdbakjbcakjcbcsjbvsjbvj</data>
      <id>lalalaldadal</id>
      <idPatient>a</idPatient>
      <isdataset>>false</isdataset>
      <message>NA</message>
    </history>
  </histories>
```

Gambar 4-29. Contoh keluaran web service dalam sintaks XML.

4.6. Implementasi Sinkronisasi Data

Untuk menjamin sinkronisasi data antara data yang disimpan di *server* dan data yang disimpan di *smartphone* diperlukan suatu aturan (*rule*) yang tepat. Tujuan dari sinkronisasi adalah agar data yang disimpan di kedua sisi sama. Sesuai dengan desain sistem yang dibuat bahwa alur pertukaran data ada di antara dua komponen yaitu *smartphone* dan *server*. Semua data yang ada di *server* berasal (diunggah) dari data yang dibuat di *smartphone*, meskipun nantinya data yang ada di *server* akan diunduh oleh pengguna menggunakan *smartphone*. Oleh sebab itu bisa dibuat suatu validasi sebelum data masuk maupun diambil dari *server*. Sebelum data disimpan di *server*, akan dicek terlebih dahulu apakah data yang akan disimpan tersebut sudah pernah disimpan. Jika data yang diminta sudah dapat sudah pernah disimpan, maka sistem akan menolak penyimpanan data tersebut. Selain itu, juga dilakukan pengecekan atribut data yang bersangkutan. Misalnya pada data *history* detak jantung ada id pasien. Jika id pasien pada tuple *history* tidak ada (tidak valid), maka penyimpanan data akan ditolak oleh sistem. Dengan demikian validasi dilakukan pada dua aspek, yaitu data yang bersangkutan harus bersifat unik, dan data-data terkait sudah tersimpan di *database*.

Dalam proses penyimpanan data, ada dua macam *rule* yang digunakan, yaitu *client first rule* dan *server first rule*. Client first rule merupakan aturan dimana data disimpan di sisi *client* terlebih dahulu. Apabila data ini ingin didaftarkan pada *server*, maka pasien atau dokter perlu mendaftarkan dahulu. Jika *server* memperbolehkan baru data ini disimpan di *server*. Jika *server* menolak, berarti akun yang sama (unik) sudah pernah disimpan di *server*. Data-data yang disimpan dengan *client first rule* ini adalah data pasien, data dokter, dan data *history*. Sedangkan *server first rule* merupakan *rule* dimana data harus disimpan di *server* dulu baru boleh disimpan di sisi *client*. Jadi, sebelum pengguna menyimpan data ini, maka *server* harus lebih dulu menyimpannya. Data – data yang disimpan dengan *rule* ini adalah data rumah sakit dan klinik, serta data afiliasi..

Untuk menjaga kevalidan data yang diinput juga dilakukan autentifikasi terhadap pengguna yang meminta layanan ke *server*. Dengan kata lain, setiap transaksi untuk mengambil data dari *server* maupun penyimpanan data di *server* dilakukan autentifikasi terhadap *username* (id) dan *password* pengguna. Dengan demikian, seorang hanya bisa melakukan pengelolaan data yang terkait dengan dirinya sendiri.

BAB 5 HASIL DAN EVALUASI

Bab hasil dan evaluasi merupakan bab yang menjelaskan hasil penelitian serta evaluasi yang dilakukan. Pada bab ini akan dijelaskan rancangan uji coba, hasil uji coba, dan analisis yang dilakukan dalam menguji sistem yang dibuat.

5.1. Rancangan Uji Coba

Uji coba pada penelitian ini akan dilakukan untuk mengetahui tingkat responsivitas *server* dalam menangani *request* dari *client*. Percobaan akan dilakukan dengan mengukur waktu dari mulai *request* dikirimkan oleh *client* sampai respon diterima oleh *client* kembali. Pengukuran waktu dilakukan pada semua pemrosesan *request* yang dieksekusi oleh setiap URL pada *server*. Untuk setiap URL, pengukuran waktu akan dilakukan sebanyak sepuluh kali. *Server* dan *client* terhubung melalui jaringan yang dibangun tersambung dengan suatu router. Alamat IP *server* dalam hal ini adalah 192.168.1.4. Kasus uji coba merupakan kasus yang melibatkan data individual. Artinya setiap transaksi yang digunakan hanya melibatkan satu paket data. Daftar kasus uji coba yang digunakan dalam uji coba ini dapat dilihat pada table 5-1.

Tabel 5-1. Parameter uji coba tingkat responsivitas *server*

URL	Kasus Uji (Parameter)
/RegisterPatient	http://192.168.1.4:8080/FP <i>Server</i> - V1/RegisterPatient?id=a&password=a&name=anwar&phone=007&email=gaada&address=lab
/RegisterDoctor	http://192.168.1.4:8080/FP <i>Server</i> - V1/RegisterDoctor?id=a&password=a&name=anwar&phone=007&email=gaada&address=lab
/Upload <i>History</i>	http://192.168.1.4:8080/FP <i>Server</i> - V1/Upload <i>History</i> ?id=lalalalal&id_patient=a&pass_patient=a&data=snajd bakjbcakjcbjsjbcjsjbvsjbvj
/Look <i>History</i>	http://192.168.1.4:8080/FP <i>Server</i> -

	V1/LookHistory?id=lalalalal&id_patient=a&pass_patient=a
/GetDoctorData	http://192.168.1.4:8080/FPServer-V1/GetDoctorData?id_doctor=a&id_patient=a&pass_patient=a
/GetHospitalData	http://192.168.1.4:8080/FPServer-V1/GetHospitalData?id_hospital=1&id_patient=a&pass_patient=a
/GetUnverifiedHistory	http://192.168.1.4:8080/FPServer-V1/GetUnverifiedHistory?id_doctor=a&pass_doctor=a&number=1
/VerifyHistory	http://192.168.1.4:8080/FPServer-V1/VerifyHistory?id_doctor=a&pass_doctor=a&id=lalalalal&message=sakit berat&hos_suggested=1
/RegisterHospital	http://192.168.1.4:8080/FPServer-V1/RegisterHospital?id_doctor=a&pass_doctor=a&name=RSanwar&phone=007&address=deketsini
/RegisterAffiliation	http://192.168.1.4:8080/FPServer-V1/RegisterAffiliation?id_doctor=a&pass_doctor=a&id_hospital=1
/GetDoctorAffiliation	http://192.168.1.4:8080/FPServer-V1/GetDoctorAffiliation?id_patient=a&pass_patient=a&id_doctor=1

5.2. Hasil Uji Coba

Setelah dilakukan uji coba diperoleh data berupa waktu yang dibutuhkan oleh *client* yang dalam hal ini adalah aplikasi E-Cardio untuk melakukan *request ke server*. Hasil uji coba dari eksperimen yang dilakukan dapat dilihat pada tabel 5-2.

Tabel 5-2. Hasil uji coba tingkat responsivitas server

URL	Waktu (ms) untuk masing masing percobaan										
	1	2	3	4	5	6	7	8	9	10	rata-rata
/RegisterPatient	75	194	255	83	89	46	136	42	248	128	129.6
/RegisterDoctor	814	140	113	118	279	92	144	100	114	275	218.9
/UploadHistory	781	245	127	246	270	126	240	251	3108	170	556.4
/LookHistory	294	221	268	270	120	191	201	175	166	160	206.6

/GetDoctorData	520	119	186	197	178	142	126	90	171	194	192.3
/GetHospitalData	1256	75	176	131	57	120	48	15	279	160	231.7
/GetUnverifiedHistory	662	235	107	122	111	146	197	200	99	399	227.8
/VerifyHistory	561	202	96	79	48	97	133	121	130	140	160.7
/RegisterHospital	348	233	231	145	297	214	208	252	159	63	215
/RegisterAffiliation	531	151	228	177	160	143	242	123	137	126	201.8
/GetDoctorAffiliation	2666	293	238	187	195	59	326	44	111	120	423.9

5.3. Analisis

Jika dilihat dari hasil eksperimen yang diperoleh dapat ditarik informasi bawah secara umum tingkat responsivitas dari *server* dalam percobaan ini adalah kurang dari setengah detik (500 ms). Selain itu dari pengujian sistem yang dilakukan ini tidak memberikan informasi adanya tren bahwa fungsi-fungsi pengambilan data pada *database (read)* lebih cepat dari penyimpanan data (*insert*) atau sebaliknya. Selain itu, hasil pengujian tersebut memberikan informasi bahwa pengiriman *request* pada pertama kali, lebih lambat dari pada *request-request* selanjutnya. Selama pelaksanaan penelitian ini belum diketahui penyebabnya. Oleh sebab itu hal ini dapat diteliti lebih lanjut. Berdasarkan hasil ini, dapat dikatakan bahwa tingkat responsivitas *server* dapat dikatakan cukup cepat.

BAB 6 PENUTUP

Bab penutup merupakan bab yang berisi tentang kesimpulan dan saran yang diperoleh dari hasil kegiatan penelitian tugas akhir ini. Pada bab ini penulis juga memberikan saran untuk pelaksanaan penelitian selanjutnya.

6.1. Kesimpulan

1. *Prototype* sistem terintegrasi pendeteksi dini dan *monitoring* penyakit jantung berbasis sinyal elektrokardiogram berhasil diimplementasikan. Sistem ini terdiri dari tiga elemen. Elemen pertama adalah *hardware* yang berupa sensor EKG. *Hardware* bertugas untuk mengambil sinyal detak jantung dari manusia. Komponen kedua adalah *smartphone* Android. Komponen kedua berfungsi menampilkan gelombang detak jantung, klasifikasi, serta mengirim data ke *server*. Selain itu perangkat Android dapat digunakan oleh dokter untuk memberikan verifikasi data detak jantung. Komponen ketiga adalah *server*. *Server* berfungsi untuk melayani penyimpanan dan pertukaran data baik oleh pasien maupun dokter.
2. Komponen pertama yaitu *hardware* EKG dan komponen kedua (*Smartphone* Android) dapat berkomunikasi melalui jaringan frekuensi *bluetooth*. Protokol yang digunakan dalam interaksi ini adalah protokol serial. Komponen kedua dan komponen ketiga (*server*) dapat saling berkomunikasi melalui jaringan *local area network* (LAN). Untuk proses pertukaran data antara dua komponen ini dilakukan dengan protocol HTTP.
3. Implementasi struktur basis data yang cocok untuk keperluan sistem ini adalah dibuat dalam dua versi. Versi pertama adalah *database* yang dibangun di *server*. Versi kedua, *database* diimplementasikan pada perangkat Android dalam bentuk *lite*. Field atau data-data yang disimpan pada kedua *database* tersebut tidak harus sama persis. Akan tetapi jenis data (*field*) yang disimpan pada *database* di *server* juga harus disimpan pada *database* yang dibangun di perangkat *smartphone*.
4. Untuk implementasi sinkronisasi data pada *prototype* dapat diimplementasikan dengan sistem *rule* atau aturan yang menjamin setiap data memiliki identifier yang unik. *Rule* yang digunakan adalah *server first rule* dan *client first rule*. Dalam implementasi dilakukan pengecekan data saat akan disimpan pada *database*. Sebelum disimpan, dicek terlebih dahulu apakah data yang bersangkutan sudah disimpan. Dengan demikian data dijamin memiliki identifier yang unik dan sinkron antara *server* dan *smartphone* Android.

Selain itu, digunakan juga autentifikasi pengguna. Sebelum mengunduh maupun mengunggah perlu dicek akun pengguna apakah merupakan pengguna yang aktif dan terdaftar dalam sistem.

5. Tingkat responsivitas *server* dalam memberikan layanan pengelolaan data pada aplikasi ini tergolong cukup cepat, karena setiap URL kurang lebih membutuhkan waktu 0.5 s untuk menjalankan fungsi yang diminta pada percobaan yang dilakukan.

6.2. Saran

1. Pada penelitian selanjutnya dapat dilakukan percobaan dengan beberapa macam perangkat *smartphone* Android yang berbeda.
2. Sebaiknya dilakukan *stress* tes, yaitu tes dengan ukuran input yang sangat besar dan beberapa *client* melakukan *request* dalam waktu yang bersamaan.
3. Sebaiknya dilakukan tes untuk berbagai macam kemungkinan *input* misalnya yang terkait dengan kasus-kasus *sql injection*.
4. Sebaiknya dilakukan enkripsi pada pengiriman *request* sehingga menjamin keamanan informasi.

DAFTAR PUSTAKA

- [1] <http://www.who.int/>. Diakses pada tanggal 30 Desember 2012.
- [2] <http://www.inaheart.org/>. Diakses pada tanggal 30 Desember 2012.
- [3] <http://www.ugm.ac.id/>. Diakses pada tanggal 30 Desember 2012.
- [4] <http://www.bps.go.id/>. Diakses pada tanggal 30 Desember 2012.
- [5] Soumya, N. (2000). *The Heart: An Indicator Of Your Well-being*. The Internet Journal of Academic Physician Assistants , 2.
- [6] Mayou R, S. D. (2003). *Characteristics of patients presenting to a cardiac clinic with palpitation*. QJM , 6: 115–123.
- [7] Daniel Lee Kulick, M. (2010, 03). *electrocardiogram_ecg_or_ekg/article.htm*. <http://www.medicinenet.com>. Diakses tanggal 30 Desember 2012.
- [8] Meek, S. &. (2002). *ABC of clinical electrocardiography*. BMJ 324:415-418.
- [9] Kuppuswamy, V. C. (2006). *Meeting the NSF targets for door to needle time in acute myocardial infarction – the role of a bolus thrombolytic*. Br J Cardiol, 13:41–2.
- [10] Moser, D. K. (2006). *Reducing Delay in Seeking Treatment by Patients With Acute Coronary Syndrome and Stroke*. A Scientific Statement From the American Heart Association Council on Cardiovascular Nursing and Stroke Council , Circulation 114.
- [11] Wei Jiang, S. G. (2005). *ECG Signal Classification using Block-based Neural Networks*. Proceedings of International Joint Conference on Neural Networks, Montreal, Canada, .
- [12] Jalal A. Nasiri I*, M. N. (2009). *ECG Arrhythmia Classification with Support Vector Machines and Genetic Algorithm*. European Symposium on Computer Modeling and Simulation.
- [13] *What is Android*. <http://developer.Android.com/guide/basics/what-is-Android.html> diakses 4 Juni 2011.
- [14] Burnette, Ed. *Hello Android*. Dallas: The Pragmatic Bookshelf. 2008 <http://kronox.org/documentacion/Hello.Android.new.pdf> diakses 10 Desember 20112.
- [15] W. Frank Ableson, Charlie Collins, dan Robi Sen. *Unlocking Android : Developer's Guide*. Greenwich: Manning, 2009.
- [16] Fajar, Muhamad. 2011. *Implementasi Algoritma Jaringan Saraf Tiruan Backpropagation pada FPGA dan Sistem Operasi Android untuk Mendeteksi Sleep Apnea Menggunakan Fitur Data ECG*. Universitas Indonesia.

- [17] Rambe, Ruliyanto Syafaadillah. 2011. *Implementasi Algoritma Fuzzy Learning Vector Quantization pada Field Programmable Gate Array untuk Pendeteksian Aritmia*. Universitas Indonesia.