PERANCANGAN APLIKASI *SINGLE TRACK AUDIO PROCESSOR* PADA *SMARTPHONE* BERBASIS *ANDROID*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

Memperoleh gelar Sarjana Teknik

OLEH:

ANGGAKARA YUDHA PRADANA

NIM. 0610630015 - 63

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2012

PERANCANGAN APLIKASI *SINGLE TRACK AUDIO PROCESSOR* PADA *SMARTPHONE* BERBASIS *ANDROID*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

Memperoleh gelar Sarjana Teknik

Disusun oleh:

ANGGAKARA YUDHA PRADANA

NIM. 0610630015 - 63

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	**Dosen Pembimbing II**
Adharul Muttaqin, ST., MT.	**Waru Djuriatno, ST., MT.**
NIP. 19760121 200501 1 001	**NIP. 19690725 199702 1 001**

PENGANTAR

Dengan nama Allah SWT Yang Maha Pengasih dan Penyayang. Segala puji bagi Allah SWT karena atas rahmat dan hidayahNya-lah penulis dapat menyelesaikan Skripsi yang berjudul "**Perancangan Aplikasi *Single Track Audio Processor* Pada *Smartphone* Berbasis *Android*"**. Shalawat dan salam atas junjungan besar kita Nabi Muhammad S.A.W. beserta keluarga dan para sahabat sekalian. Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Program Studi Teknik Informatika dan Komputer Fakultas Teknik Universitas Brawijaya Malang.

Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih penulis yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan baik bantuan moril maupun materiil selama penulisan tugas akhir ini. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan rasa hormat dan terima kasih penulis kepada :

1. Kedua Orang Tua penulis (Indra Wahyudi dan Kismi Sulistyowati) dan seluruh keluarga yang senantiasa tiada henti-hentinya memberikan do'a dan restu demi terselesaikannya skripsi ini.
2. Saudari Lussia Mariesti Andriany, SE. yang senantiasa memberikan dorongan moral dan motivasi selama kegiatan akademik di Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
3. Bapak Sholeh Hadi Pramono, DR., Ir., MS. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
4. Bapak M. Azis Muslim, ST., MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
5. Bapak Moch. Rif'an. ST., MT. selaku Ketua Program Studi Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
6. Bapak Waru Djuriatno, ST. MT. sebagai Ketua Kelompok Dosen Keahlian Teknik Informatika dan Komputer sekaligus sebagai dosen pembimbing II yang telah banyak memberikan bimbingan, masukan dan arahan dalam penyusunan skripsi ini.

7. Bapak Adharul Muttaqin ST., MT. selaku dosen pembimbing I yang telah banyak memberikan bimbingan, masukan dan arahan dalam penyusunan tugas akhir ini.
8. Bapak Ir. Choiri (alm), Bapak Panca Mudjirahardjo, ST., MT, dan Bapak Wijono, Ir., MT, PhD selaku dosen pendamping akademik yang telah memberikan motivasi dan bimbingan selama kegiatan akademik.
9. Bapak dan Ibu Dosen, Staff Administrasi Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.
10. Semua Asisten, Ka. Lab serta Laboran dari Laboratorium Teknik Informatika dan Komputer yang telah memberikan banyak bantuan dan dukungan dalam menyelesaikan tugas akhir ini.
11. Saudara Aflahlana Septian Habibina, ST. , Andik Nur Achmad, Yulius Candra Widyanto ST., Kuncoro Adi P., ST., Norman Natanael, Eka Prakarsa Mandyarta atas bantuan, dukungan serta penciptaan suasana persaingan yang kompetitif selama studi di kampus ini.
12. Teman-teman angkatan 2006 (Ge-Force) dan teman-teman Home Band Teknik segala bantuan serta motivasinya selama menjadi mahasiswa.
13. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Hanya doa yang bisa penulis berikan, semoga Allah SWT tidak pernah melepaskan segala RahmatNya yang tak terhingga untuk kita selama-lamanya. Amin.

Penulis menyadari bahwa tugas akhir ini masih banyak kekurangan dan masih jauh dari sempurna. Untuk itu, saran dan kritik yang membangun sangat penulis harapkan. Semoga tugas akhir ini membawa manfaat bagi penyusun maupun pihak lain yang menggunakannya.

Malang, 10 Februari 2012

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

ANGGAKARA YUDHA PRADANA. 2012 : Perancangan Aplikasi *Single Track Audio Processor* Pada *Smartphone* Berbasis *Android*. Skripsi Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya. Dosen Pembimbing: Adharul Muttaqin, ST., MT. dan Waru Djuriatno, ST., MT.

Sebuah s*martphone* memiliki *processor, memory*, media penyimpanan data yang cukup besar, dan sistem operasi sendiri sehingga mampu melakukan beberapa tugas komputasi, seperti *word processing*, *spreadsheet*, memutar video, dan merekam suara. Kelebihan dari sistem operasi Android adalah digunakannya Dalvik Virtual Machine yang membuat *smartphone* dapat melakukan komputasi kompleks dan penanganan data yang besar. Salah satu pemanfaatan dari *smartphone* adalah sebagai perekam suara bergerak. Kekurangan dari perekam suara yang ada pada perangkat bergerak adalah kualitas perekaman yang kurang memadai sehingga hasil rekaman memiliki detil yang rendah.

Dalam perancangan perangkat lunak s*ingle track audio processor*, sistem dibagi menjadi dua subsistem yaitu subsistem *audio recording* dan *audio processing*. Pada subsistem *audio recording* perancangan difokuskan pada pemanfaatan *library* API Android untuk perekaman suara yang berasal dari mikrofon. Sedangkan subsistem *audio processing* perancangan difokuskan pada pemanfaatan *library* API Android untuk pengolahan suara baik hasil perekaman maupun pemilihan berkas suara yang sudah ada.

Pada subsistem *audio recording* pada *smartphone* Android, didapatkan hasil yang tidak maksimal karena tanggapan frekuensi dari mikrofon internal yang terbatas pada frekuensi tertentu. Penambahan fitur pemilihan file dapat menggantikan kekurangan dari mikrofon yang dapat diolah sama dengan fitur perekam yang disediakan. Subsistem *audio processing* dapat bekerja dengan baik dimana pada pengolahan sinyal suara berupa *equalizer*, didapatkan penguatan frekuensi pada pita frekuensi yang ditentukan. Pada pengolahan sinyal suara berupa *reverb*, didapatkan hasil pembangkitan efek *reverb* menimbulkan gema dari sumber suara yang membuat suara seolah-olah berada pada sebuah ruangan.

Kata Kunci : *Audio Recording, Equalizer, Reverb, Android.*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi komputasi yang pesat saat ini memacu inovasi dan kemajuan industri *software* atau perangkat lunak. Di mulai dengan perangkat lunak yang hanya berbasis teks (*text based application*) sampai dikembangkannya perangkat lunak visual atau perangkat lunak yang memiliki GUI (*Graphical User Interface*). Semula distribusi perangkat lunak tersebut hanya untuk komputer, namun sekarang telah berkembang ke peralatan portable seperti *handphone, PDA (Personal Digital Assistance),* atau bahkan ditanamkan pada peralatan rumah tangga. Pengembangan perangkat portabel menjadi nilai tambah tersendiri di kalangan masyarakat.

Seperti telah disebutkan di atas, bahwa *handphone* merupakan salah satu distribusi inovasi perangkat lunak yang sedang marak saat ini. Dengan dikembangkannya *handphone* dengan konsep *smartphone*, maka sebuah *handphone* dapat berfungsi layaknya sebuah komputer. *Smartphone* memiliki *processor*, *memory*, media penyimpanan data yang cukup besar, dan sistem operasi sendiri sehingga mampu melakukan beberapa tugas komputer, seperti *word processing*, *spreadsheet*, memutar video, dan merekam suara. Salah satu sistem operasi *smartphone* yang ada saat ini adalah *Android*. Kelebihan dari sistem operasi ini adalah digunakannya *Dalvik Virtual Machine* yang membuat *smartphone* dapat melakukan komputasi kompleks dan penanganan data yang besar.

Sebuah perekam suara digital merupakan sebuah bentuk perkembangan teknologi komputasi yang memungkinkan manusia menangkap suara yang ditangkap oleh mikrofon yang diubah menjadi data digital untuk keperluan tertentu seperti dokumentasi, sarana publikasi, maupun untuk media komunikasi. Salah satu kekurangan dari perekam suara yang ada pada perangkat portabel adalah kualitas perekaman yang kurang memadai sehingga hasil rekaman tidak terlalu jelas dan memiliki detil yang rendah.

Dalam skripsi ini akan digunakan sebuah *smartphone* berbasis *Android* yang didalamnya dirancang sebuah aplikasi perekam suara digital dengan fitur *processing* sederhana. Penggunaan *handset Android* di sini karena kemampuan dari sistem operasi *Android* yang memungkinkan penanganan data besar yang nantinya dihasilkan oleh proses penangkapan suara. Suara yang ditangkap oleh mikrofon akan diubah menjadi data digital untuk mendapatkan hasil maksimal dari perekaman suara. Alternatif lain juga diberikan pemilihan berkas suara dari dalam perangkat untuk dapat diolah. Dengan kualitas data yang baik diharapkan proses ekualisasi dan reverberasi yang akan diberikan memberikan hasil yang maksimal.

Pemberian proses ekualisasi dapat memberikan pengaruh pada lebar pita dari suara masukan. Hal ini dapat meningkatkan detil khusus seperti rasa pukulan pada frekuensi rendah hingga tingkat kejelasan artikulasi pada suara yang berupa percakapan. Pemberian proses reverberasi dapat memberikan kesan ruangan yang mempengaruhi tingkat kelembutan dari sumber suara. Menurut Wikipedia,

> “ Rooms used for speech typically need a shorter reverberation time so that speech can be understood more clearly. If the reflected sound from one syllable is still heard when the next syllable is spoken, it may be difficult to understand what was said. "Cat", "Cab", and "Cap" may all sound very similar. If on the other hand the reverberation time is too short, tonal balance and loudness may suffer. “

Penggunaan reverberasi yang tidak tepat akan mengurangi kejelasan atau artikulasi dari kata – kata yang diucapkan. Keluaran yang diharapkan adalah sebuah audio yang memiliki karakteristik *Hi-Fidelity* dari sebuah perangkat portabel.

Berdasarkan penjelasan di atas, maka penulis tertarik untuk menulis skripsi dengan judul "PERANCANGAN APLIKASI SINGLE TRACK AUDIO PROCESSOR PADA SMARTPHONE BERBASIS ANDROID".

1.2 Rumusan Masalah

1. Bagaimana merancang sebuah *Single Track Audio Recorder* pada sistem operasi *Android*?
2. Bagaimana implementasi *Audio Processor* dengan menerapkan filter ekualisasi dan reverberasi pada *Single Track Audio Recorder*?
3. Bagaimana pengujian dan analisis kerja sistem *Audio Processor* dalam *Smart-Phone* berbasis *Android*?

1.3 Tujuan

Merancang dan membangun suatu aplikasi *Single Track Audio Recorder* yang menerapkan filter ekualisasi dan efek reverberasi pada *platform Android.*

1.4 Batasan Masalah

1. Pembahasan difokuskan pada pembuatan *single track audio processor*.
2. Sistem operasi yang digunakan adalah *Android* versi 2.3 (*Gingerbread*).

3. IDE (*Integrated Development Environment*) yang digunakan adalah Eclipse 3.4.2 (Helios).
4. Platform pengembangan yang digunakan adalah JSE 6 (Java *Standard Edition*6).
5. *SDK (Software Development Kit)* yang digunakan adalah SDK Platform *Android* 2.1-update1, API 7, revision 2.
6. ADT (*Android Development Tools*) yang digunakan adalah versi 0.9.9.v201009221407-60953.
7. *Audio Processing* yang diterapkan adalah filter ekualisasi dan efek reverberasi.
8. Pengujian performa sistem meliputi fungsi *capture* dan penyimpanan audio atau pemilihan file audio, analisa karakter suara hasil sampling, analisa karakter setelah diberi efek dan proses ekualisasi.
9. Pengujian performa sistem menggunakan emulator *Android* dan sebuah *Smart-Phone Android*.
10. Implementasi yang dihasilkan dari sistem berupa sebuah aplikasi untuk *audio processing*.

1.5 Manfaat

1. Bagi penulis
   a. Menerapkan ilmu yang telah diperoleh dari Teknik Elektro Konsentrasi Teknik Informatika dan Komputer Universitas Brawijaya.

b. Mendapatkan pemahaman tentang perancangan dan pengembangan *Audio Processor* yang diimplemantasikan pada sistem *Android*.

2. Bagi pengguna

a. Menyediakan sarana *Audio Recording dan Processing* pada sebuah *smart-phone Android*.

b. Meningkatkan detil suara masukan dari pengolahan yang dilakukan oleh perangkat.

1.6 Sistematika Penulisan

Sistematika penulisan dalam skripsi ini sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, manfaat dan sistematika penulisan.

BAB II DasarTeori

Membahas teori dasar dan teori penunjang yang berkaitan dengan *Digital Audio Recording*, *Digital Signal Processor*, *Equalizer*, *Reverb*, sejarah *Android*, dan struktur platform *Android*.

BAB III Metode Penelitian

Membahas metode yang digunakan dalam penulisan yang terdiri dari studi literatur, perancangan perangkat lunak, implementasi perangkat lunak, pengujian dan analisis, serta pengambilan kesimpulan dan saran.

BAB IV Perancangan dan Implementasi

Membahas analisis kebutuhan dan perancangan yang sesuai dengan teori yang ada dan implementasinya.

BAB V Pengujian dan Analisis

Memuat hasil pengujian dan analisis terhadap sistem yang telah direalisasikan.

BAB VI Penutup

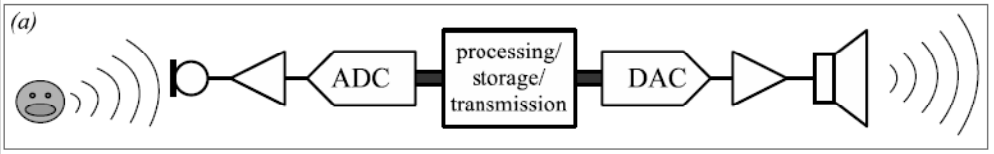
Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian program, serta saran–saran untuk pengembangan lebih lanjut.

BAB II

DASAR TEORI

2.1 *Digital Audio Recording*

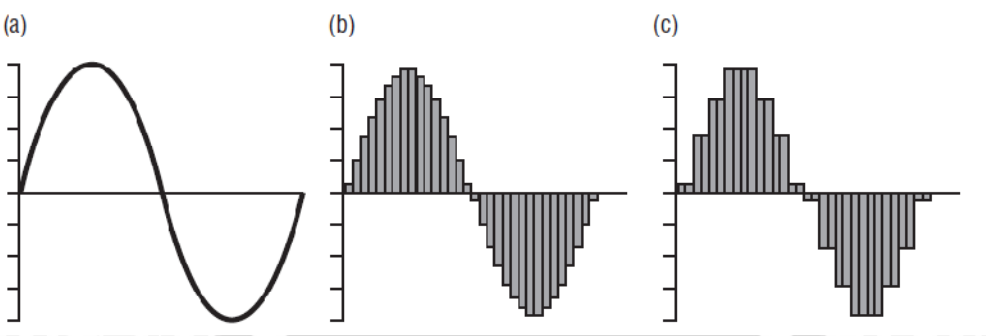
Digital Audio Recording adalah proses menangkap gelombang suara yang dilakukan oleh mikrofon dengan cara menangkap perbedaan tekanan yang disebabkan oleh gelombang suara pada membran tipis, kemudian diubah menjadi tegangan dan arus. Sinyal elektrik tersebut akan diubah menjadi data digital dengan cara melakukan teknik *sampling* menggunakan ADC (*Analog to digital Converter*) [Lou-09:1]. Secara diagram, ditunjukkan pada Gambar 2.1.



Gambar 2.1 Diagram sistem *digital audio processing system* (dari kiri ke kanan), sebuah masukan dari mikropon, sebuah amplifier, ADC, unit pengolahan digital, DAC, amplifier, dan *speaker*

Sumber : [Lou-09:3].

ADC (*Analog to Digital Converter*), sebuah subsistem yang mengubah gelombang analog menjadi data digital dengan melakukan *sampling* [AMA-02:3]. Gelombang yang masuk ke ADC akan dicacah (*sampling*) sesuai frekuensi tertentu yang kemudian diubah menjadi data digital. Gambar 2.2 menampilkan gelombang masukan dan keluaran ADC.



Gambar 2.2 (a) Gambar sinyal analog masukan, (b) Gambar sinyal keluaran yang telah disampling, (c) gambar sinyal hasil *sampling* dengan kerapatan sampling yang berbeda

Sumber : [Kat-06:155].

Sampling merupakan sebuah teknik pencacahan data analog pada waktu tertentu yang seragam untuk menghasilkan data digital. Frekuensi dari waktu pencacahan disebut *sampling rate* [Lou-09:4]. Kualitas dari *sampling* ditentukan oleh 2 faktor, resolusi (dinyatakan dalam bit) dan frekuensi *sampling*. Semakin tinggi resolusi dan semakin tinggi frekuensi maka kualitas data digitalnya semakin mendekati data analog yang berarti tinggi pula kualitas datanya. Pada table 2.1 terdapat beberapa perbandingan *Sample rate* dan resolusi yang sering digunakan.

Application	Sample rate, resolution	Used how
telephony	8 kHz, 8–12 bits	64 kbps A-law or μ-law
voice conferencing	16 kHz, 14–16 bits	64 kbps SB-ADPCB
mobile phone	8 kHz, 14–16 bits	13 kbps GSM
private mobile radio	8 kHz, 12–16 bits	<5 kbps, e.g. TETRA
long-play audio	32 kHz, 14–16 bits	minidisc, DAT, MP3
CD audio	44.1 kHz, 16–24 bits	stored on CDs
studio audio	48 kHz, 16–24 bits	CD mastering
very high end	96 kHz, 20–24 bits	for *golden ears* listening

Tabel 2.1 Beberapa resolusi dan *sample-rate* sampling yang sering digunakan

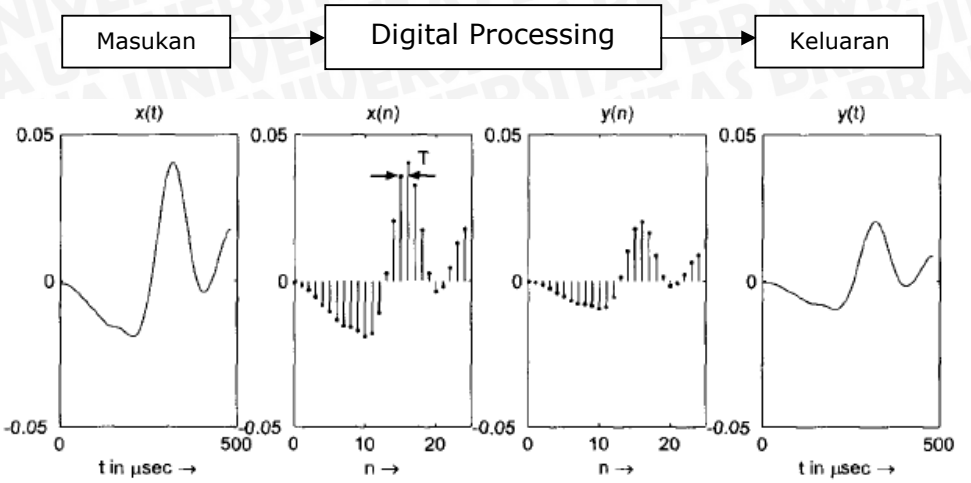
Sumber : [Lou-09:5].

2.2 *Digital Signal Processor (DSP)*

Digital signal processors (DSP) adalah sebuah pengolahan sinyal yang berdasarkan waktu diskrit. Arsitektur dan set instruksinya didesain special untuk melakukan pengolahan sinyal secara *real-time* [Zöl-08:97].

Terdapat tiga blok operasi yang menyusun sebuah *audio processor* yaitu operasi penambahan, perkalian, and waktu tunda (*delay)*. Beberapa efek dapat dihasilkan dari ketiga operasi tersebut. Operasi penambahan digunakan untuk menambahkan dua sinyal masukan. Operasi perkalian dapat digunakan untuk menambah dan mengurangi intensitas sinyal. Penerapan digital filter yang terdiri dari operasi perkalian pada frekuensi tertentu akan menghasilkan sebuah filter ekualiser. Operasi waktu tunda *(delay)* digunakan untuk membuat efek suara berulang pada sebuah sinyal suara. Efek *delay* yang memiliki *feedback* akan menghasilkan suara memantul yang disebut efek *echo.* Efek *echo* yang digunakan secara bersamaan dapat menghasilkan efek reverbrasi (*reverb*) [Kat-06:177].

Pemrosesan sinyal dapat dibagi menjadi 2 macam menurut karakteristik pengolahannya. Pemrosesan pada domain waktu memiliki karakteristik tergantung pada satuan waktu, misalnya *delay, echo*, dan *reverb*. Sedangkan untuk pemrosesan pada domain frekuensi memiliki karakteristik transformasi frekuensi, misalnya *equalizer*. Pada gambar 2.3 digambarkan konsep dasar dari pemrosesan sinyal.



Gambar 2.3 Gambar sinyal masukan, kemudian dilakukan pemrosesan (penurunan *gain*) dan sinyal keluaran

Sumber : [Ama-02:3].

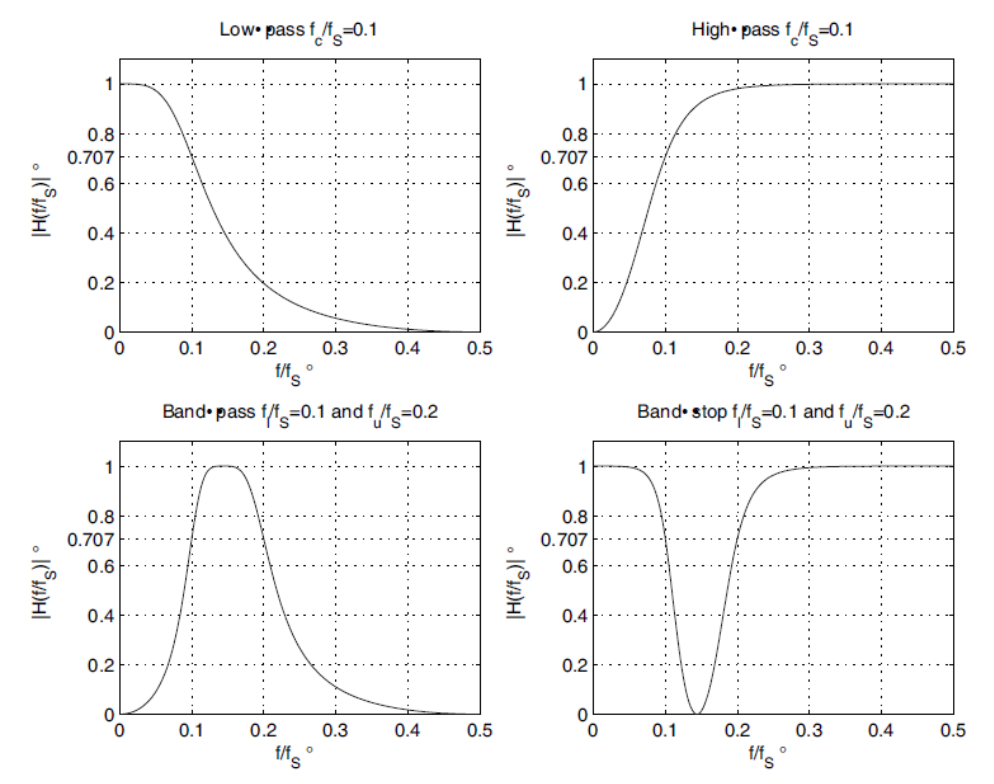
2.2.1 Ekualisasi (*Equalization*)

Ekualisasi adalah sebuah subsistem *audio processing* yang terdiri dari berbagai macam filter dalam domain frekuensi. Semakin detil ekualisasinya, kompleks fungsi filternya [Zöl-08:115]. Dalam penerapannya, sebuah filter sederhana banyak digunakan untuk mendapatkan audio yang berkualitas. Beberapa filter yang sering digunakan untuk membuat *Equalizer* adalah :

a. *Low-pass* dan *high-pass* filter dengan frekuensi *cutoff* f_c (3 dB cutoff frequency) ditunjukkan pada gambar 2.4 bagian pertama. Filter ini melewatkan frekuensi di atas dan bawahnya.

b. *Band-pass* dan *band-stop* filter memiliki frekuensi tengah f_c dan sebuah frekuensi cutoff bawah (f_l) dan frekuensi cutoff atas (f_u).

c. *Octave* filter adalah *band-pass* filter yang memiliki frekuensi cutoff khusus.



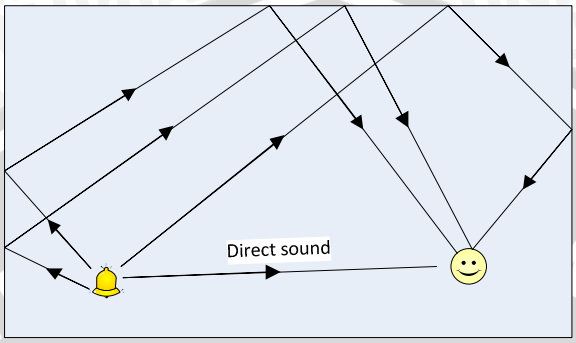
Gambar 2.4 Pola sinyal pada filter

Sumber : [Zöl-08:116].

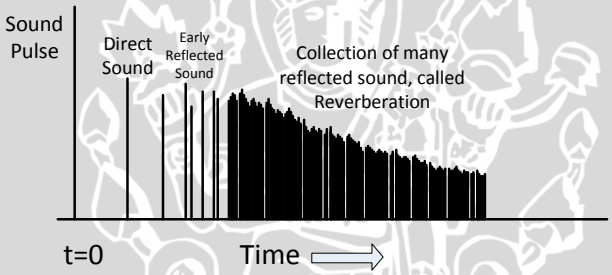
2.2.2 Reverberasi *(Reverberation)*

Reverberasi adalah suara yang masih tersisa pada sebuah ruangan ketika suara aslinya dihilangkan. Reverb terjadi ketika sebuah suara ditimbulkan pada sebuah ruangan tertutup yang menimbulkan efek *echo* yang berlipat dan lama – lama menghilang karena diserap oleh dinding dan udara. Dengan kata lain, ketika sumber suara dihentikan, namun tetap terjadi pantulan yang amplitudonya menurun hingga habis.

Sebuah *Reverberator Digital* dapat dibuat dengan menggunakan sebuah algoritma sederhana, yaitu menggunakan *multiple feedback* dari sebuah *delay* untuk membuat banyak efek *echo* dalam jeda tertentu.



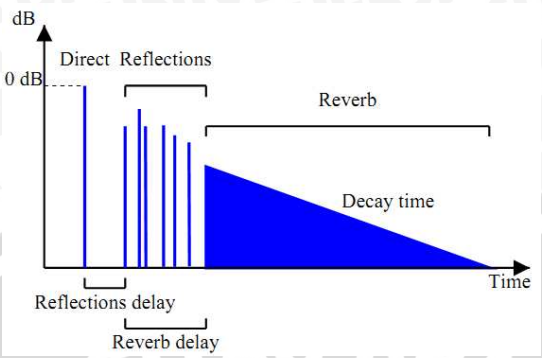
Gambar 2.5 Ilustrasi Reverberasi



Gambar 2.6 Gelombang Reverberasi yang timbul

Sumber : http://hyperphysics.phy-astr.gsu.edu

2.2.2.1 Environment reverbration model



Gambar 2.7 Pemodelan respon dari Reverberasi

Sumber : [3D-99]

Pemodelan pada respon reverberasi dapat menginderakan ruangan dimana pendengar berada. Pemodelan ini dibagi menjadi tiga bagian yaitu : langsung (*direct*), pantulan (*reflections*), dan reverberasi. Pemodelan digambarkan pada gambar 2.7 dengan asumsi pendengar dan sumber suara berada pada lingkungan yang sama, sehingga pantulan dan reverberasinya bergantung pada ruangan dimana pendengar berada. Respon dari reverberasi dikarakteristikkan oleh parameter berikut:

- Energi dari masing – masing bagian (*direct*, *reflections*, and *reverb*).
- *Reflection* dan *reverb* delay.
- "*Direct filter*": sebuah low-pass filter untuk mengurangi energi pada frekuensi tinggi.
- "*Room filter*": sebuah low-pass filter untuk komponen pantulan dan reverb untuk mengurangi energi pada frekuensi tinggi.
- Waktu luruh pada frekuensi rendah dan tinggi.
- *Diffusion* dan *density* dari reverberasi.

Sebuah ruangan sebenarnya adalah Reverberasi (*Reverb*) ditambah pantulan (*reflections*). Difusi (*Diffusion*) sebanding dengan jumlah pantulan per detik di akhir gema. Kepadatan gema dapat berubah seiring peluruhan gema.

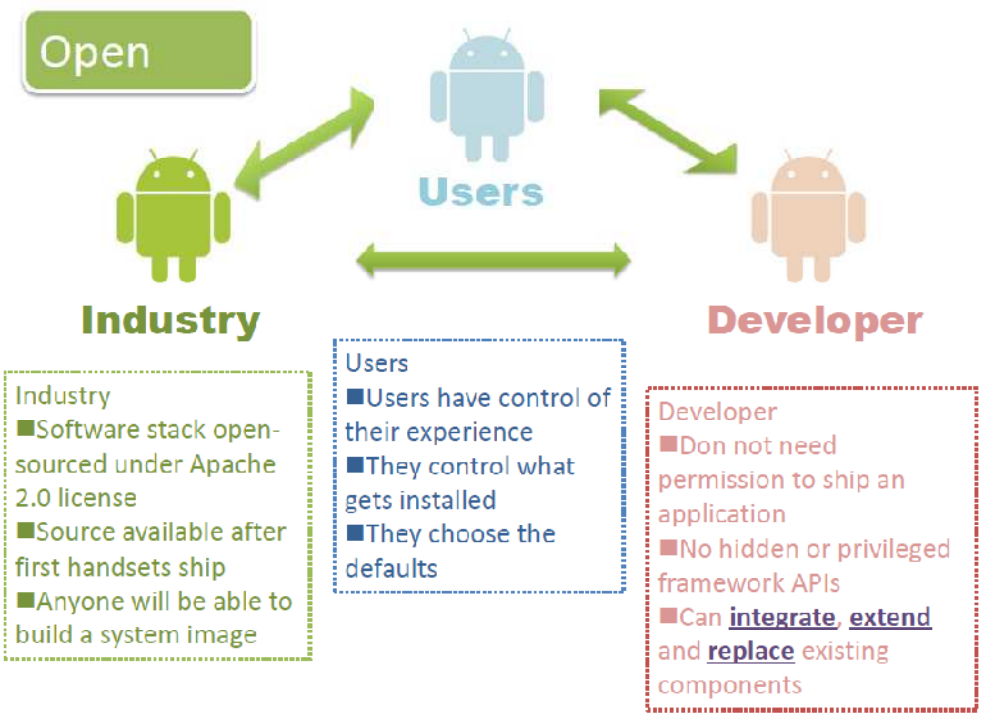
Kepadatan (*Density*) sebanding dengan jumlah resonansi per hertz di akhir gema. Kepadatan rendah menghasilkan suara berongga seperti yang ditemukan di kamar kecil. [Selfon-04:27].

Berdasarkan parameter-parameter yang ada, pengaturan dapat dikelompokkan sebagai berikut :

- Pengaturan waktu : Reflections_delay, Reverb_delay, Decay_time
- Pengaturan volume : Room, Reflections, Reverb, Room_rolloff_factor
- Pengaturan perbandingan frekuensi : HF_reference, Room_HF, Decay_HF_ratio
- Pengaturan Warna Reverb : Diffusion (“graininess”), Density (“hollowness”)

2.3 *Android*

Android adalah sebuah platform yang dikembangkan bersama oleh Google Inc. dan Open Handset Alliance (OHA) yang merupakan sebuah sistem operasi, perangkat perantara (*Middleware*), dan beberapa aplikasi *mobile* utama (*key mobile applications*). Konsep yang dikembangkan adalah sebuah platform yang benar – benar terbuka (*open*) bagi programmer dan pengguna. Pada gambar 2.7 digambarkan hubungan dari ketiga aktor dalam *Android*



Gambar 2.8 Konsep peranan *Open* pada ketiga aktor *Android*

2.3.1 Sejarah *Android*

Google melakukan akuisisi pada sebuah perusahaan bernama *Android* Inc. pada tahun 2005 untuk memulai riset mengenai *Android* Platform. Beberapa pengembang yang berperan dalam *Android* Inc. adalah Andy Rubin, Rich Miner, Nick Sears, dan Chris White.

Pada awal 2007, beberapa perusahaan di bidang telekomunikasi membuat dukungan kepada Platform *Android* dan membentuk Open Handset Alliance (http://www.openhandsetalliance.com). Beberapa anggota dari OHA adalah

- Sprint Nextel
- T-Mobile

• Motorola

• Samsung

• Sony Ericsson

• Toshiba

• Vodafone

• Google

• Intel

• Texas Instruments

Tujuan utama dari persekutuan ini adalah melakukan pengembangan secara berkala dan merespon dengan cepat keinginan consumer. Hal tersebut sesuai dengan tujuan dasar *Android* yang didesain untuk memenuhi kebutuhan operator seluler, produsen perangkat keras, dan pengembang aplikasi.

2.3.2 Struktur Platform *Android*

Android memiliki 5 struktur dasar yaitu lapisan Linux kernel atau lapisan Driver, lapisan Libraries, Lapisan *Android* Runtime yang berisi Dalvik Virtual Machine, sebuah lapisan Application Framework, dan lapisan tertinggi adalah aplikasi. Struktur dasar dari platform *Android* seperti pada gambar 2.8.

Gambar 2.9 Struktur Platform *Android*

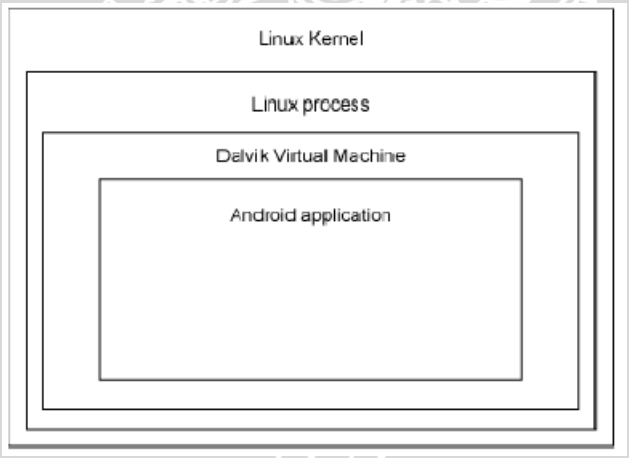
Pada lapisan Kernel, *Android* menggunakan kernel Linux 2.6.x sebagai inti servisnya. Lapisan ini bertanggungjawab pada driver dari perangkat keras, akses dari *resource*, dan *power management.* Driver yang disediakan meliputi Display, Camera, Keypad, WiFi, Flash Memory, Audio, dan IPC (interprocess communication).

Pada lapisan *Libraries*, terdapat berbagai macam *library,* ditulis dalam bahasa C berdasarkan BSD yang disesuaikan untuk *Embedded Linux-based Device*. Lapisan ini bertanggungjawab pada *library* dasar yang berjalan pada kernel Linux.

Pada lapisan *Android Runtime*, merupakan kelebihan dari platform *Android*. *Core Libraries* yang dipanggil memungkinkan berfungsinya bahasa pemrograman Java, namun berjalan pada mesin virtual Dalvik. Mesin Virtual

Dalvik adalah mesin virtual yang bertipe Register-based, beda dengan mesin virtual Java yang bertipe Stack-based. Mesin virtual Dalvik mampu menjalankan beberapa mesin virtual sekaligus. Jika sebuah aplikasi java menggunakan Library, maka setiap aplikasi akan berjalan di atas mesin virtualnya sendiri. Selain itu, sebuah mesin virtual yang bertipe *register-based* dapat memproses data yang lebih besar dan lebih cepat daripada mesin virtual *stack-based*.

Pada lapisan framework memuat beberapa API utama sistem yang mencakup telephony, resources, locations, UI, content providers (data), dan package managers (installation, security, dan lainnya). Dengan menggunakan API tersebut, programmer dapat membuat aplikasi yang setingkat dengan aplikasi standar lainnya. Lingkungan aplikasi *Android* seperti pada gambar 2.9.



Gambar 2.10 Lingkungan aplikasi *Android*

2.3.3 Struktur Aplikasi Android

Arsitektur Android mendorong digunakannya kembali komponen yang memungkinkan kita untuk mempublikasikan dan berbagi aktifitas / kegiatan, layanan dan data antar aplikasi dengan batasan keamanan yang dapat kita definisikan sendiri. Hal ini memungkinkan bagi pengembang untuk memasukkan

suatu komponen lain seperti telepon *dialer* atau manajemen *contact*, atau menambahkan fungsionalitas baru kedalam aplikasi yang telah dibuat. Adapun dasar-dasar didalam aplikasi yang dibangun di Android, yaitu :

- *Activity Manager* : bagian yang mengontrol siklus hidup suatu aktivitas / kegiatan. Suatu kegiatan atau aktifitas dapat dibandingkan dengan bentuk tampilan *windows* atau bentuk web yang diusung oleh suatu control (tampilan) dimana membentuk suatu antarmuka, sebuah aktifitas merepresentasikan satu layar tersendiri (*single screen*).
- *Views* : komponen UI (*User Interface*) yang membangun sebuah antarmuka.
- *Notification Manager* : suatu komponen yang menyediakan mekanisme yang selalu konsisten memberitahukan atau mengingatkan kepada pengguna.
- *Content Providers* : memungkinkan beberapa aplikasi utnuk berbagi data diantara mereka.
- *Resource Manager* : seperti konsep sumber daya yang diusung ASP.NET, aplikasi android juga memungkinkan bagi pengembang untuk menyimpan sumber daya seperti *string* atau gambar.

2.3.4 Struktur Android *Project*

Dalam setiap membangun suatu aplikasi yang diimplementasikan di Android maka akan didapat folder-folder yang berisi :

1. *Src* : berisi semua kode sumber (*source code*) dalam file kelas (*class files*) untuk proyek tersebut.
2. *Gen* : berisi class file R.java. Dimana R.java adalah class file yang secara otomatis dihasilkan untuk memegang referensi bagi setiap sumber daya didalam aplikasi.
3. *Android 2.3* : Namanya berubah sesuai dengan versi SDK yang digunakan (2.3, disini). Berisi class file API (*Application Programming Interface*) Android yang dikemas kedalam file android.jar
4. *Assets* : Didalam folder ini kita dapat menempatkan setiap sumber daya eksternal (file teks, gambar,..) . Lebih jauh lagi dari sekedar folder *res,* folder ini lebih dari sekedar sistem file dimana kita bisa meletakkan file yan gingin kita akses sebagai byte yang baku.
5. *Drawable Folder* : Berisi gambar-gambar yang dipelukan untuk alikasi yang di buat. Didalamnya terdapat tiga buah folder, hal ini dimaksudkan utnuk menyertakan sumber gambar dengan resolusi yang berbeda, sesuai dengan layar telepon yang digunakan.
6. *Layout* : Berisi file *main.xml* yang mendefinisikan pandangan bahwa aplikasi ini membangun suatu *User Interface.*
7. *Values* : Mengandung sumber informasi lainnya yang dapat digunakan untuk aplikasi [Has-10].

2.3.5 API *(Application Programming Interface)Level*

API level adalah tingkatan revisi dari kerangka kerja yang disesuaikan dengan perkembangan sistem operasi Android. API tersebut telah disediakan oleh Google Inc. sebagai pengembang resmi sistem operasi Android dengan tujuan agar pengembang aplikasi mendapatkan kemudahan serta kesetaraan dalam membuat aplikasi.

Kerangka API berisi kumpulan paket *(packages)* dan kelas kelas *(class)*, kumpulan elemen XML dan atribut untuk menyatakan sebuah *file manifest*, kumpulan elemen XML dan atribut untuk menyatakan dan mengakses sumber daya, kumpulan *intents*, serta kumpulan perizinan dalam sistem, serta pengaturan kebijakan dalam sistem.

Beberapa API dan versi sistem operasi Android tampak pada tabel 2.2.

Platform Version	API Level
Android 3.1	12
Android 3.0	11
Android 2.3.4	10
Android 2.3.3	
Android 2.3	9
Android 2.2	8
Android 2.1	7
Android 2.0.1	6
Android 2.0	5
Android 1.6	4
Android 1.5	3
Android 1.1	2
Android 1.0	1

Tabel 2.2 Versi Android dan API Level.
Sumber: **http://developer.android.com/guide/appendix/api-levels.html**

2.4 Kelas Audiofx

Kelas Audiofx adalah kelas dasar yang digunakan untuk mengendalikan efek audio dari library OpenSL ES yang di dalam *Android*. Sebuah aplikasi tidak dapat menggunakan kelas Audiofx secara langsung, namun dengan cara menggunakan turunan kelasnya untuk mengendalikan efek. Turunan kelas dari yang disediakan oleh kelas Audiofx adalah:

- Equalizer
- Virtualizer
- BassBoost
- PresetReverb
- EnvironmentalReverb

2.4.1 Kelas *Equalizer*

Kelas *Equalizer* adalah kelas turunan dari Kelas Audiofx. Kelas *equalizer* memiliki kendali atas Library OpenSL ES pada bagian *equalizer*.

Untuk mengaplikasikan *equalizer* kepada *Audio Track* atau *Media Player* tertentu, dibutuhkan ID *audio session* dari obyek *media player* atau *audio track* yang akan diberi efek. Untuk memberikan efek pada jalur keluaran secara global, *session* dapat diset pada angka 0 pada saat membuat obyek dari *Equalizer*.

Pembuatan obyek dari *equalizer* melalui konstruktor dibutuhkan 2 parameter yaitu prioritas dari *equalizer* itu sendiri dan ID *Audio Session* dari *media player* yang diberi pengaruh oleh *equalizer*.

Beberapa fungsi yang ditangani oleh kelas *Equalizer* tampak pada tabel 2.3.

Tabel 2.3 Fungsi-fungsi dari kelas *Equalizer*

Tipe Fungsi	Fungsi
short	getBand(int frequency) Gets the band that has the most effect on the given frequency.
int[]	getBandFreqRange(short band) Gets the frequency range of the given frequency band.
short	getBandLevel(short band) Gets the gain set for the given equalizer band.
short[]	getBandLevelRange() Gets the level range for use by `setBandLevel(short, short)`.
int	getCenterFreq(short band) Gets the center frequency of the given band.
short	getCurrentPreset() Gets current preset.
short	getNumberOfBands() Gets the number of frequency bands supported by the Equalizer engine.
short	getNumberOfPresets() Gets the total number of presets the equalizer supports.
String	getPresetName(short preset) Gets the preset name based on the index.
Equalizer.Settings	getProperties() Gets the equalizer properties.
void	setBandLevel(short band, short level) Sets the given equalizer band to the given gain value.
void	setParameterListener(Equalizer.OnParameterChangeListener listener) Registers an OnParameterChangeListener interface.
void	setProperties(Equalizer.Settings settings) Sets the equalizer properties.
void	usePreset(short preset) Sets the equalizer according to the given preset.

2.4.2 Kelas *EnvironmentalReverb*

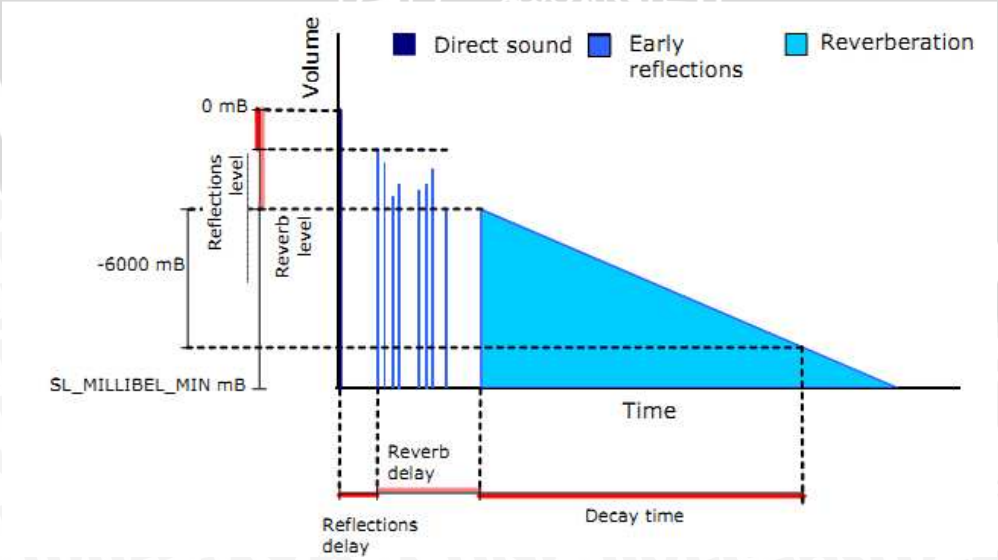
Kelas *EnvironmentalReverb* adalah kelas turunan dari kelas Audiofx. Kelas *EnvironmentalReverb* memiliki kendali atas Library OpenSL ES pada bagian *Environmental Reverb*.

Berbeda dengan equalizer, untuk mengaplikasikan Reverb kepada *Audio Track* atau *Media Player* tertentu diterapkan dengan metode Aux *send* efek pada media player. Pada media player dipanggil attachAuxEffect dengan mengikutkan parameter ID dari obyek reverb yang telah dibuat.

Pembuatan obyek baru dari EnvironmentalReverb melalui konstruktor tidak memerlukan parameter karena EnvironmentalReverb bersifat aux efek. Beberapa variabel yang disediakan oleh API *EnvironmentalReverb* tampak tabel 2.4.

Tabel 2.4 Fungsi-fungsi dari kelas *EnvironmentalReverb*

Tipe	Nama Fungsi	Deskripsi
int	PARAM_DECAY_HF_RATIO	Decay HF ratio.
int	PARAM_DECAY_TIME	Decay time.
int	PARAM_DENSITY	Density.
int	PARAM_DIFFUSION	Diffusion.
int	PARAM_REFLECTIONS_DELAY	Early reflections delay.
int	PARAM_REFLECTIONS_LEVEL	Early reflections level.
int	PARAM_REVERB_DELAY	Reverb delay.
int	PARAM_REVERB_LEVEL	Reverb level.
int	PARAM_ROOM_HF_LEVEL	Room HF level.
Int	PARAM_ROOM_LEVEL	Room level.



Gambar 2.11 Ilustrasi parameter *Reverb*

BAB III
METODE PENELITIAN

Pada bab ini dijelaskan langkah-langkah yang akan dilakukan dalam perancangan, implementasi dan pengujian dari aplikasi perangkat lunak yang akan dibuat. Kesimpulan dan saran disertakan sebagai catatan atas aplikasi dan kemungkinan arah pengembangan aplikasi selanjutnya.

3.1 Studi Literatur

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut meliputi:

a. *Digital Audio Recording*
   - Proses *Capturing* Audio
   - Pembuatan *Waveform* Audio

b. *Audio Efek*
   - Ekualisasi (*Equalization*)
   - Reverbrasi (*Reverberation)*

c. Android
   - Struktur sistem operasi Android
   - Standar *Library* Android
   - Struktur Pemrograman Android

3.2 Analisis Kebutuhan (*Requirement Analysis*)

Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan (*requirements*) sistem (*Single-Track Audio Processor*).

a. Perangkat Keras :
   1. PC atau Laptop
   2. *Smart-phone Android*
      - Processor ARM scale X5 600 MHz
      - ROM 256 MB
      - RAM 512 MB
      - Android v. 2.3 (*Gingerbread*)
      - Layar 3.2” tipe kapasitif dengan resolusi 240x320 pixel

b. Perangkat Lunak :

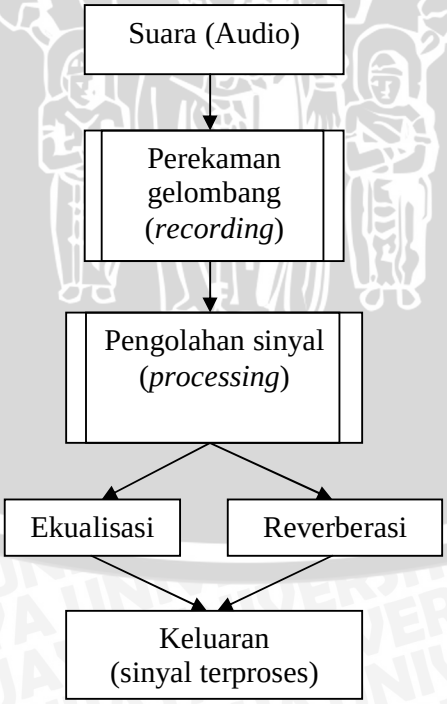
1. Sistem Operasi Windows XP
2. Eclipse IDE
3. JRE (Java *Runtime Environment*)
4. SDK (*Software Development Kit*) Android
5. ADT (*Android Developer Tools*) *plugin for Eclipse*

3.3 Perancangan dan Implementasi

Perancangan aplikasi dilakukan setelah semua kebutuhan sistem didapatkan melalui tahap analisis kebutuhan. Perancangan subsistem *Single-Track Audio Processor* dilakukan dengan mengidentifikasi kelas-kelas dan *interface-interface* yang dibutuhkan yang dimodelkan dalam *class diagram.* Perancangan *Single-Track Audio Processor* ini secara garis besar meliputi :

- Proses *Audio Capturing* untuk merekam suara yang ditangkap dari mikrofon..
- Pembuatan algoritma dan pengaplikasian *Equalizer* dalam *Audio Processor*.
- Pembuatan algoritma dan pengaplikasian *Reverb* dalam *Audio Processor*.

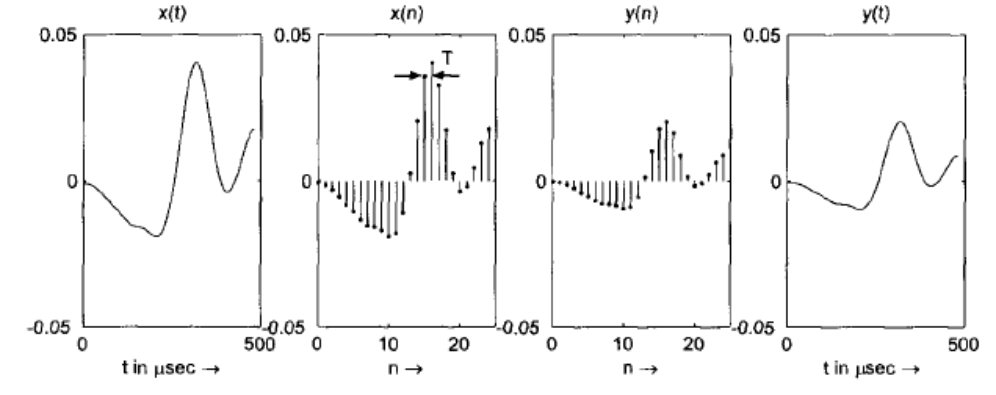
Langkah – langkah program *audio processor* adalah sebagai berikut :



Gambar 3.1 Blok Diagram Sistem

Keterangan:

a. Gelombang suara ditangkap oleh mikrofon untuk selanjutnya di-*sampling* menjadi data digital.
b. Data digital disimpan dalam memory, dan dibuat pola *waveform*-nya.
c. Setelah data digital siap, maka dapat dilakukan proses pemberian ekualisasi ataupun reverberasi pada data digital.



Gambar 3.2 Pengolahan sinyal

3.4 Pengujian dan Analisis

Pengujian perangkat lunak pada skripsi ini dilakukan agar dapat menunjukkan bahwa perangkat lunak telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang melandasinya.

Pengujian yang dilakukan meliputi pengujian performa sistem meliputi analisa fungsi standar perekam suara (*Audio Recorder*) dan analisa hasil perubahan karakteristik suara yang direkam setelah diberi efek.

Pada pengujian fungsi standar perekan suara akan dilakukan alur perekaman suara hingga menghasilkan bentuk gelombang suara yang ditangkap (berupa *waveform*). Kemudian dari hasil tersebut akan diberikan filter ekualiser atau reverberasi yang kemudian hasilnya akan dibandingkan dengan hasil rekaman awal. Perubahan pola gelombang akan menunjukkan pengaruh dari pemberian efek terhadap sinyal masukan.

3.5 Pengambilan Simpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem aplikasi telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibangun. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi selanjutnya.

BAB IV

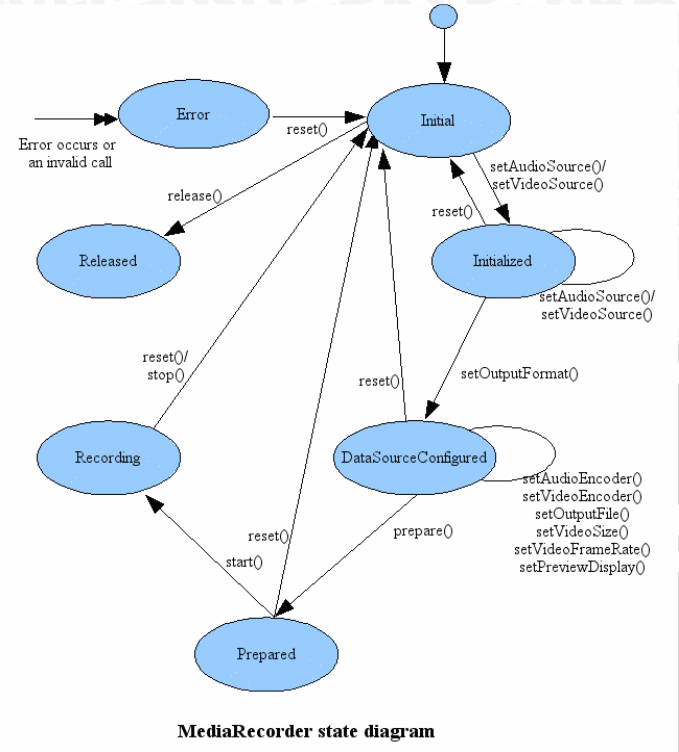
PERANCANGAN DAN IMPLEMENTASI

4.1 Perancangan Umum

Perancangan umum terdiri atas perancangan umum subsistem *audio recorder* dan aplikasi *audio processing* berupa pengolahan sinyal dalam *equalizer* dan *reverb.* Perancangan umum subsistem *audio recorder* menggambarkan relasi antar kelas (*class)* sebagai pemodelan subsistem *audio recorder* secara keseluruhan.

4.1.1 Perancangan Umum *Audio Recorder*

Dalam perancangan *Audio Recorder* pada sistem operasi Android, dimanfaatkan kelas tersedia yang dikemas dalam *Application Programming Interface* (API). API yang digunakan adalah paket Media. API tersebut menangani sistem masukan dan keluaran suara. Di dalam paket tersebut tersedia kelas *MediaRecorder* yang memiliki beberapa parameter fungsi. Beberapa fungsi yang ditangani oleh kelas *MediaRecorder* tampak pada gambar 4.1.



Gambar 4.1 *State Diagram* dari kelas MediaRecorder.

Sumber: http://developer.android.com/reference/android/media/MediaRecorder.html

Langkah – langkah umum perancangan *Audio Recorder*,

1. Membuat sebuah obyek dari kelas *MediaRecorder*
2. Mengatur sumber masukan suara menggunakan fungsi MediaRecorder.setAudioSource(), menggunakan MediaRecorder.AudioSource.MIC jika masukan yang dikehendaki berasal dari mikrofon.
3. Mengatur format *file* keluaran menggunakan fungsi MediaRecorder.setOutputFormat()
4. Mengatur nama hasil rekaman menggunakan MediaRecorder.setOutputFile()

5. Mengatur *encoder* hasil menggunakan MediaRecorder.setAudioEncoder()
6. Mempersiapkan proses perekaman dengan memanggil fungsi MediaRecorder.prepare() dari obyek MediaRecorder.
7. Untuk memulai proses perekaman, memanggil fungsi MediaRecorder.start().
8. Untuk mengakhiri rekaman, memanggil MediaRecorder.stop().
9. Jika selesai menggunakan kelas MediaRecorder, memanggil fungsi MediaRecorder.release() untuk membebaskan *resource*.

4.1.2 Perancangan Umum *Equalizer*

Dalam perancangan *Audio Processing* pada sistem operasi Android, dimanfaatkan paket dan kelas tersedia yang dikemas dalam API *audiofx* yang merupakan anak dari paket *media.*

API *audiofx* tersebut menangani sistem pengolahan suara. Di dalam paket tersebut tersedia kelas yang akan digunakan, yaitu kelas *Equalizer* dan *EnvironmentalReverb* yang memiliki beberapa parameter fungsi. Beberapa fungsi yang ditangani oleh kelas *Equalizer* tampak pada tabel 4.1.

Tabel 4.1 Fungsi-fungsi dari kelas Equalizer

Tipe Fungsi	Fungsi
short	getBand(int frequency) Gets the band that has the most effect on the given frequency.
int[]	getBandFreqRange(short band) Gets the frequency range of the given frequency band.
short	getBandLevel(short band) Gets the gain set for the given equalizer band.
short[]	getBandLevelRange() Gets the level range for use by `setBandLevel(short, short)`.
int	getCenterFreq(short band) Gets the center frequency of the given band.

Tipe Fungsi	Fungsi
short	getCurrentPreset() Gets current preset.
short	getNumberOfBands() Gets the number of frequency bands supported by the Equalizer engine.
short	getNumberOfPresets() Gets the total number of presets the equalizer supports.
String	getPresetName(short preset) Gets the preset name based on the index.
Equalizer.Settings	getProperties() Gets the equalizer properties.
void	setBandLevel(short band, short level) Sets the given equalizer band to the given gain value.
void	setParameterListener(Equalizer.OnParameterChangeListener listener) Registers an OnParameterChangeListener interface.
void	setProperties(Equalizer.Settings settings) Sets the equalizer properties.
void	usePreset(short preset) Sets the equalizer according to the given preset.

Pertama dibuat sebuah objek Equalizer itu sendiri untuk mengawali prosesnya dan mengendalikan *framework* audio. Beberapa *preset* telah disediakan di dalam *framework* tersebut atau dengan memanfaatkan beberapa fungsi yang tersedia untuk mendapatkan parameter yang sesuai dengan keinginan pengguna.

Setelah objek terbentuk, ditetapkan sesi ID dari audio yang akan diolah. ID dapat berupa angka unik yang merepresentasikan objek audio tersebut. Sesi ID 0 digunakan jika Equalizer akan digunakan untuk mengolah sinyal keluaran secara keseluruhan.

4.1.3 Perancangan Umum *Environmental Reverb*

Dalam membuat pemroses sinyal *Reverb*, digunakan kelas *Environmental Reverb* yang merupakan implementasi dari kelas audiofx. *Environmental Reverb* memberikan akses kepada *reverb engine* dari OpenSL ES. Beberapa variabel yang disediakan oleh API *EnvironmentalReverb* tampak tabel 4.2.

Tabel 4.2 Fungsi-fungsi dari kelas Environmental Reverb

Tipe	Nama Fungsi	Deskripsi
int	PARAM_DECAY_HF_RATIO	Decay HF ratio.
int	PARAM_DECAY_TIME	Decay time.
int	PARAM_DENSITY	Density.
int	PARAM_DIFFUSION	Diffusion.
int	PARAM_REFLECTIONS_DELAY	Early reflections delay.
int	PARAM_REFLECTIONS_LEVEL	Early reflections level.
int	PARAM_REVERB_DELAY	Reverb delay.
int	PARAM_REVERB_LEVEL	Reverb level.
int	PARAM_ROOM_HF_LEVEL	Room HF level.
Int	PARAM_ROOM_LEVEL	Room level.

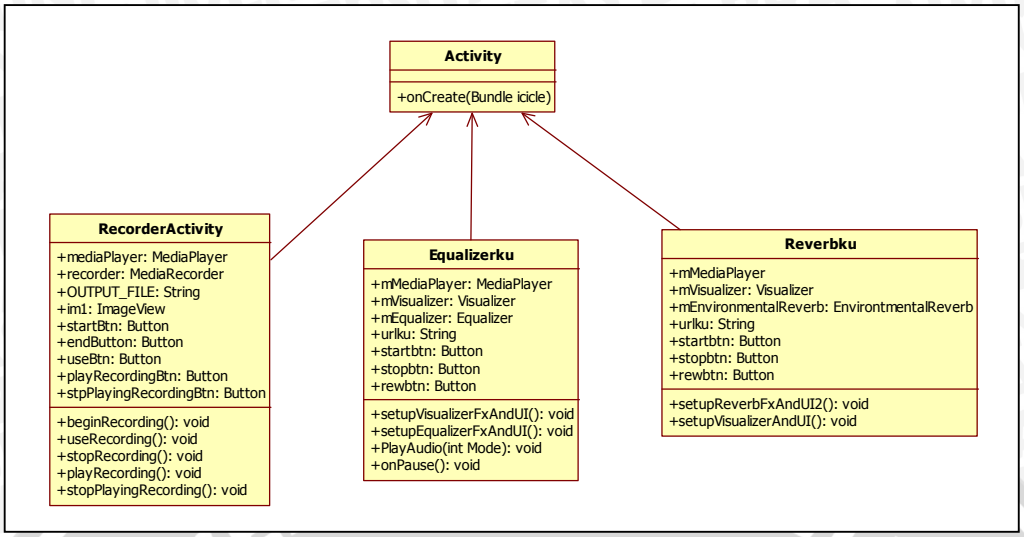
Implementasi yang dilakukan adalah membuat sebuah objek dari *EnvironmentalReverb* untuk *inisiasi* dan mengendalikan *reverb engine* dalam *framework audio*. Untuk mengaplikasikan pada sebuah sinyal masukan, obyek yang telah dibuat diimplementasikan pada kanal keluaran dari Media Player.

4.2 Perancangan Perangkat Lunak

4.2.1 Diagram Klas (*Class Diagram*)

Diagram klas memberikan gambaran pemodelan elemen-elemen klas yang membentuk sebuah sistem perangkat lunak mulai dari atribut-atribut serta method-methodnya dan hubungannya dengan klas-klas lain dalam sebuah sistem. Pada subsistem ini terdapat sejumlah kelas yang saling membentuk relasi. Kelas-kelas yang akan membangun subsistem ini dikelompokkan pada sebuah paket yaitu paket audioprocessing.

Atribut dan operasi dari kelas audioprocessing digambarkan dalam diagram kelas pada Gambar 4.2.



Gambar 4.2 Diagram kelas anggota paket com.audioprocessing
Sumber: Perancangan

4.3 Implementasi Sistem

Setelah tahap perancangan sistem tahap selanjutnya adalah tahap implementasi. Tujuan dari tahap implementasi ini merupakan proses transformasi hasil perancangan perangkat lunak. Pembahasan terdiri dari lingkungan implementasi (spesifikasi perangkat lunak dan perangkat keras), implementasi antarmuka aplikasi dengan sintaks dari bahasa pemrograman yang digunakan.

4.3.1 Spesifikasi Subsistem

Perancangan perangkat lunak yang telah diuraikan menjadi acuan untuk melakukan implementasi menjadi sebuah sistem *single-track audio processing* yang dapat berfungsi sesuai dengan kebutuhan. Spesifikasi subsistem diimplementasikan pada spesifikasi perangkat keras dan perangkat lunak

4.3.1.1 Spesifikasi Perangkat Keras

Pengembangan sistem *single-track audio processing* menggunakan sebuah *smartphone* dengan spesifikasi perangkat keras yang dijelaskan pada Tabel 4.3.

Tabel 4.3 Spesifikasi perangkat keras

Nama Komponen	Spesifikasi
Model	LG P-500
Prosesor	ARM Scale X5 600MHz
Memori (RAM)	512 MB
Penyimpanan Internal (ROM)	256 MB
Display	TFT, capacitive touch screen, 16M colors, 240 x 320 pixels, 2.8 inches

Sumber: Implementasi

4.3.1.2 Spesifikasi Perangkat Lunak

Pengembangan subsistem menggunakan perangkat lunak dengan spesifikasi yang dijelaskan pada Tabel 4.4.

Tabel 4.4 Spesifikasi perangkat lunak komputer

Sistem operasi	Android 2.3 *(Gingerbread)*
Bahasa pemrograman	Java dan Android OOP
Tools pemrograman	ADT 1.6.0_02
Lingkungan pemrograman	Java(TM) 2 Runtime Environment, Standard Edition (build
IDE *(Integrated Development*	Eclipse Helios Version: 3.6.2 Build id: M20110210-1200

Sumber: Implementasi

4.3.1.3 Spesifikasi *Single Track Audio Processor*

Spesifikasi dari aplikasi yang dibuat seperti tampak pada tabel Tabel 4.5.

Tabel 4.5 Spesifikasi aplikasi *Single Track Audio Processor*

Nama Aplikasi	UBU-4035
Kategori	Perekam dan pengolah berkas suara
Platform minimum	Android 2.3 *(Gingerbread)*
Sumber berkas suara	Mikrofon internal dan berkas di dalam perangkat
Tipe berkas yang didukung	MP3, WAV, 3GP, dan AMR
Durasi Perekam	Terbatas pada kapasitas memori
Editor	Memotong awal dan akhir berkas suara yang dibuka
Ekualisasi	5 pita (*bands*) pada frekuensi 60Hz, 250Hz, 1000Hz, 4000Hz, dan 14000Hz
Reverberasi	5 pengaturan, *Room Size*, *Room High Frequency Level, Decay High Frequncy Ratio, Density Level*, dan *Reverb Level*

Sumber: Implementasi

4.3.2 Batasan-Batasan Implementasi

Beberapa batasan dalam mengimplementasikan subsistem adalah sebagai berikut:

1. Spesifikasi pengujian terbatas pada perangkat keras yang disebutkan pada sub-bab sebelumnya.
2. Keterbatasan sinyal masukan dari perangkat keras perekam (mikrofon) dalam menangkap sinyal masukan terbatas pada rentang frekuensi tertentu.
3. Implementasi sistem pengolahan sinyal hanya pada lapisan aplikatif, bukan pada teori pengolahan sinyal.
4. Implementasi efek terjadi secara *real-time*, sehingga hasil implementasinya bersifat *just-in-time*.

4.3.3 Implementasi Klas pada File Program

Setiap klas yang telah dirancang pada proses perancangan direalisasikan pada sebuah *file* program *.java. Tabel 4.6 menjelaskan mengenai pasangan antara klas dengan *file* program yang digunakan untuk mengimplentasikannya.

Tabel 4.6 Implementasi klas pada kode program *.java

No.	Paket	Nama Klas	Nama *File* Program
1.	com.audioprocessing	EditActivity	EditActivity.java
2.		Equalizerku	Equalizerku.java
3.		RecorderActivity	RecorderActivity.java
4.		Reverbku	Reverbku.java
5.		SelectActivity	SelectActivity.java
6.		SelectActivity_list	SelectActivity_list.java

Sumber: Implementasi

4.3.4 Implementasi Algoritma

Aplikasi *Single Track Audio Processing* ini mempunyai beberapa proses utama yaitu membuat merekam suara dari sumber luar, memilih file yang ada sebagai sumber internal, mengubah parameter ekualiser, dan mengubah parameter reverb.

4.3.4.1 Implementasi Algoritma Merekam Suara

Proses perekaman suara diinisiasi ketika menu rekam baru dipilih. Gambar 4.3 merupakan Implementasi Algoritma Merekam Suara.

```
DESCRIPTION:

1     private static String OUTPUT_FILE="/sdcard/temporaryUBU.3gpp";
2     private void beginRecording() throws Exception {
3                   im1.setImageResource(R.drawable.led_on);
4                   killMediaRecorder();
5                   outFile = new File(OUTPUT_FILE);
6                   if(outFile.exists())
7                   {
8                         outFile.delete();
9                   }
10                  recorder = new MediaRecorder();
```

```
11                  recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
12
13          recorder.setOutputFormat(MediaRecorder.OutputFormat.RAW_AMR);
14
15          recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
16                  recorder.setOutputFile(OUTPUT_FILE);
17                  recorder.prepare();
18                  recorder.start();
19          }
20    private void stopRecording() throws Exception {
21                  im1.setImageResource(R.drawable.led_off);
22                  if (recorder != null) {
23                          recorder.stop();
24                          }
25          }
26    private void playRecording() throws Exception {
27                  killMediaPlayer();
28                  im1.setImageResource(R.drawable.green_on);
29                  mediaPlayer = new MediaPlayer();
30                  mediaPlayer.setDataSource(OUTPUT_FILE);
31                  mediaPlayer.prepare();
32                  mediaPlayer.start();
33          }
34
35    private void stopPlayingRecording() throws Exception {
36                  im1.setImageResource(R.drawable.green_off);
37                  if(mediaPlayer!=null)
38                  {
39                          mediaPlayer.stop();
40                  }
41          }
42    private boolean mWasGetContentIntent;
43    private static final int REQUEST_CODE_EDIT = 1;
44
45    private void useRecording() {
46       try {
47         Intent intent = new Intent(Intent.ACTION_EDIT,
48                                    Uri.parse(OUTPUT_FILE));
49         intent.putExtra("was_get_content_intent",
50                         mWasGetContentIntent);
51         intent.setClassName(
52                 "com.audioprocessing",
53                 "com.audioprocessing.EditActivity");
54         startActivityForResult(intent, REQUEST_CODE_EDIT);
55       } catch (Exception e) {
56         Log.e("audioprocessing", "Couldn't start editor");
57       }
58    }
```

Gambar 4.3 Potongan kode program perekam suara
Sumber: Perancangan

Penjelasan dari potongan algoritma perekam suara pada Gambar 4.3 yaitu:

1. Pendeklarasian method beginRecording() yang berisi pemanggilan gambar led_on, memanggil killMediaRecorder(), pengecekan file temporary, jika ada maka diambil aksi menghapusnya.
2. Pembuatan obyek baru recorder yang bertipe MediaRecorder()
3. Melakukan deklarasi parameter seperti memilih MIC pada setAudioSource, RAW_AMR pada setOutputFormat, AMR_NB pada

setAudioEncoder, dan variable OUTPUT_FILE yang berisi "/sdcard/temporaryUBU.3gpp" pada setOutputFile.

4. Pemanggilan fungsi prepare() untuk mempersiapkan perekam.
5. Pemanggilan fungsi start() untuk mulai perekaman.
6. Pendeklarasian fungsi stopRecording() yang berisi pemanggilan gambar led_off, jika obyek recorder ada, memanggil fungsi stop().
7. Pendeklarasian fungsi playRecording() yang berisi pemanggilan fungsi killMediaPlayer() untuk menghentikan media player jika sudah dimulai dan pemanggilan gambar green_on.
8. Dalam playRecording() dibuat sebuah obyek mediaplayer yang diturunkan dari kelas MediaPlayer().
9. Pengaturan parameter setDataSource yang diarahkan kepada variable OUTPUT_FILE.
10. Pemanggilan fungsi prepare() untuk mempersiapkan mediaplayer.
11. Pemanggilan fungsi start() untuk mulai memutar rekaman.
12. Pendeklarasian fungsi stopPlayingRecording() yang berisi pemanggilan gambar green_off, jika obyek mediaplayer ada, memanggil fungsi stop().
13. Mendeklarasikan fungsi useRecording() yang berisi pembuatan intent baru yang bernama intent yang bertipe menampilkan data dan mencantumkan variabel OUTPUT_FILE. Intent dibuat dari kelas EditActivity, dan memanggil aktivitas intent melalui perintah startActivityForResult dan menyimpan hasilnya dalam variable REQUEST_CODE_EDIT.

4.3.4.2 Implementasi Algoritma Equalizer

Implementasi Equalizer memanfaatkan kelas Equalizer pada package audiofx didalam struktur platform Android lapisan *Application Framework*. Equalizer yang diciptakan bersifat *Real-time*, memiliki efek pada keluaran pemutar audio. Hal ini didapatkan dengan cara melewatkan parameter id dari pemutar audio kepada equalizer.

```
1    private void setupEqualizerFxAndUI() {
2            mEqualizer = new Equalizer(0,
3    mMediaPlayer.getAudioSessionId());
4            mEqualizer.setEnabled(true);
5
6            TextView eqTextView = new TextView(this);
7            eqTextView.setText("Equalizer:");
8            mLinearLayout.addView(eqTextView);
9
10           short bands = mEqualizer.getNumberOfBands();
11
12           final short minEQLevel =
13   mEqualizer.getBandLevelRange()[0];
14           final short maxEQLevel =
15   mEqualizer.getBandLevelRange()[1];
16
17           for (short i = 0; i < bands; i++) {
18               final short band = i;
19
20               TextView freqTextView = new TextView(this);
21               freqTextView.setLayoutParams(new
22   ViewGroup.LayoutParams(
23                       ViewGroup.LayoutParams.FILL_PARENT,
24                       ViewGroup.LayoutParams.WRAP_CONTENT));
25               freqTextView.setGravity(Gravity.CENTER_HORIZONTAL);
26               freqTextView.setText((mEqualizer.getCenterFreq(band)
27   / 1000) + " Hz");
28               mLinearLayout.addView(freqTextView);
29
30               LinearLayout row = new LinearLayout(this);
31               row.setOrientation(LinearLayout.HORIZONTAL);
32
33               TextView minDbTextView = new TextView(this);
34               minDbTextView.setLayoutParams(new
35   ViewGroup.LayoutParams(
36                       ViewGroup.LayoutParams.WRAP_CONTENT,
37                       ViewGroup.LayoutParams.WRAP_CONTENT));
38               minDbTextView.setText((minEQLevel / 100) + " dB");
39
40               TextView maxDbTextView = new TextView(this);
41               maxDbTextView.setLayoutParams(new
42   ViewGroup.LayoutParams(
43                       ViewGroup.LayoutParams.WRAP_CONTENT,
44                       ViewGroup.LayoutParams.WRAP_CONTENT));
45               maxDbTextView.setText((maxEQLevel / 100) + " dB");
46
```

```
47              LinearLayout.LayoutParams layoutParams = new
48   LinearLayout.LayoutParams(
49                      ViewGroup.LayoutParams.FILL_PARENT,
50                      ViewGroup.LayoutParams.WRAP_CONTENT);
51              layoutParams.weight = 1;
52              SeekBar bar = new SeekBar(this);
53              bar.setLayoutParams(layoutParams);
54              bar.setMax(maxEQLevel - minEQLevel);
55              bar.setProgress(mEqualizer.getBandLevel(band));
56
57              bar.setOnSeekBarChangeListener(new
58   SeekBar.OnSeekBarChangeListener() {
59                  public void onProgressChanged(SeekBar seekBar,
60   int progress, boolean fromUser) {
61                      mEqualizer.setBandLevel(band, (short)
62   (progress + minEQLevel));
63                  }
64
65                  public void onStartTrackingTouch(SeekBar
66   seekBar) {}
67                  public void onStopTrackingTouch(SeekBar seekBar)
68   {}
69              });
70
71              row.addView(minDbTextView);
72              row.addView(bar);
73              row.addView(maxDbTextView);
74
75              mLinearLayout.addView(row);
76          }
77      }
78
```

Gambar 4.4 Potongan kode program *equalizer*
Sumber: Perancangan

Penjelasan algoritma Equalizer pada `setupEqualizerFxAndUI()`dalam Gambar 4.4 yaitu:

1. Membuat obyek mEqualizer yang terbuat dari kelas equalizer yang memiliki 2 parameter, prioritas yang diberikan nilai 0 dan audio session melewatkan sesi dari media player melalui `mMediaPlayer.getAudioSessionId()` dan mengaktifkannya melalui `setEnabled(true)`.
2. Mendefinisikan parameter yang dibutuhkan seperti pita (*bands*), MinEQLevel, dan MaxEQLevel.
3. Menentukan frekuensi tengah dari lebar pita frekuensi masing – masing pita melalui `mEqualizer.getCenterFreq(band)`.

4. Membuat slider pada setiap pita untuk mengatur nilai penguatan frekuensinya.

4.3.4.3 Implementasi Algoritma Reverb

Implementasi Reverb sama dengan Implementasi Equalizer, yaitu memanfaatkan kelas EnvironmentalReverb, pada kelas audiofx didalam struktur platform Android lapisan *Application Framework*. Berbeda dengan Equalizer yang menangkap identitas sesi dari audio player, Reverb diterapkan dengan metode Aux efek dari media player. Hal ini didapatkan dengan cara memberikan identitas sesi dari reverb kepada media player.

Lima parameter yang dapat disesuaikan yaitu :

- *Room Size*, mengatur tingkat master volume dari efek *environmental reverb.*
- *Room High Frequency Level*, mengatur tingkat volume pada frekuensi 5 kHz relatif tingkat volume pada frekuensi rendah.
- *High Frequncy Decay Ratio*, mengatur perbandingan antara waktu meluruh (decay) frekuensi tinggi (pada 5 kHz) relatif pada waktu meluruh (decay) frekuensi rendah. *High frequency decay ratio* (dalam skala permil) memiliki rentang antara 100 hingga 2000. Perbandingan pada angka 1000 mengindikasikan bahwa semua frekuensi meluruh pada waktu yang sama.
- *Density Level*, mengendalikan kepadatan pantulan pada saat proses peluruhan.

- *Density*, menggunakan skala permil dengan rentang nilai antara 0 hingga 1000. Nilai 1000 ‰ mengindikasikan reverberasi yang natural. Pada tingkat dibawahnya akan menghasilkan efek yang lebih berwarna. Pada saat *density* diatur rendah akan menghasilkan suara yang kosong, dimana hal ini akan berguna untuk membuat simulasi reverberasi yang ringan seperti pada kamar mandi.
- *Reverb Level*, mengendalikan tingkat volume dari pantulan reverberasi yang terjadi.

```
1    mMediaPlayer = new MediaPlayer();
2    mEnvironmentalReverb = new EnvironmentalReverb();
3    mMediaPlayer.attachAuxEffect(mEnvironmentalReverb.getId());
4    mMediaPlayer.setAuxEffectSendLevel(1.0f);
5
6          mEnvironmentalReverb.setEnabled(true);
7
8            LinearLayout.LayoutParams layoutParams = new
9    LinearLayout.LayoutParams(
10                   ViewGroup.LayoutParams.FILL_PARENT,
11                   ViewGroup.LayoutParams.WRAP_CONTENT);
12           layoutParams.weight = 1;
13           SeekBar bar1 = new SeekBar(this);
14           bar1.setLayoutParams(layoutParams);
15           bar1.setMax(maxRSLevel - minRSLevel);
16           bar1.setProgress(mEnvironmentalReverb.getRoomLevel());
17
18   bar1.setOnSeekBarChangeListener(new
19   SeekBar.OnSeekBarChangeListener() {
20               public void onProgressChanged(SeekBar seekBar, int
21   progress, boolean fromUser) {
22                   mEnvironmentalReverb.setRoomLevel((short) (progress
23   + minEQLevel));
24               }
25
26               public void onStartTrackingTouch(SeekBar seekBar) {}
27               public void onStopTrackingTouch(SeekBar seekBar) {}
28           });
29
30           row1.addView(bar1);
31
32           final short minHFLevel = -4000;
33           final short maxHFLevel = 0;
34
35           SeekBar bar2 = new SeekBar(this);
36           bar2.setLayoutParams(layoutParams);
37           bar2.setMax(maxHFLevel - minHFLevel);
38           bar2.setProgress(mEnvironmentalReverb.getRoomHFLevel());
39
40           bar2.setOnSeekBarChangeListener(new
41   SeekBar.OnSeekBarChangeListener() {
42               public void onProgressChanged(SeekBar seekBar, int
43   progress,
```

```
                    boolean fromUser) {
                mEnvironmentalReverb.setRoomHFLevel((short)
(progress + minHFLevel));
            }

            public void onStartTrackingTouch(SeekBar seekBar) {}
            public void onStopTrackingTouch(SeekBar seekBar) {}
        });

        row2.addView(bar2);

        final short minDHFLevel = 0;
        final short maxDHFLevel = 2000;

        SeekBar bar3 = new SeekBar(this);
        Bar3.setLayoutParams(layoutParams);
        Bar3.setMax(maxDHFLevel - minDHFLevel);
        Bar3.setProgress(mEnvironmentalReverb.getDecayHFRatio());

        Bar3.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            public void onProgressChanged(SeekBar seekBar, int
progress,
                    boolean fromUser) {
                mEnvironmentalReverb.setDecayHFRatio((short)
(progress + minDHFLevel));
            }

            public void onStartTrackingTouch(SeekBar seekBar) {}
            public void onStopTrackingTouch(SeekBar seekBar) {}
        });

        Row3.addView(bar3);

        final short minReLLevel = 0;
        final short maxReLLevel = 2000;

        SeekBar bar4 = new SeekBar(this);
        Bar4.setLayoutParams(layoutParams);
        Bar4.setMax(maxReLLevel - minReLLevel);

Bar4.setProgress(mEnvironmentalReverb.getReflectionsLevel());

        Bar4.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            public void onProgressChanged(SeekBar seekBar, int
progress,
                    boolean fromUser) {
                mEnvironmentalReverb.setReverbLevel((short)
(progress + minReLLevel));
            }

            public void onStartTrackingTouch(SeekBar seekBar) {}
            public void onStopTrackingTouch(SeekBar seekBar) {}
        });

        Row4.addView(bar4);

        final short minDenLevel = 0;
        final short maxDenLevel = 1000;


        SeekBar bar5 = new SeekBar(this);
        Bar5.setLayoutParams(layoutParams);
        Bar5.setMax(maxDenLevel - minDenLevel);
```

```
109         Bar5.setProgress(mEnvironmentalReverb.getDensity());
110
111         Bar5.setOnSeekBarChangeListener(new
112  SeekBar.OnSeekBarChangeListener() {
113             public void onProgressChanged(SeekBar seekBar, int
114  progress,
115                     boolean fromUser) {
116                 mEnvironmentalReverb.setDensity((short) (progress +
117  minDenLevel));
118             }
119
120             public void onStartTrackingTouch(SeekBar seekBar) {}
121             public void onStopTrackingTouch(SeekBar seekBar) {}
122         });
123
124         Row5.addView(bar5);
125     }
```

Gambar 4.5 Potongan kode program *Reverb*
Sumber: Perancangan

Penjelasan algoritma Reverb pada `setupReverbFxAndUI2()`dalam Gambar 4.5 yaitu:

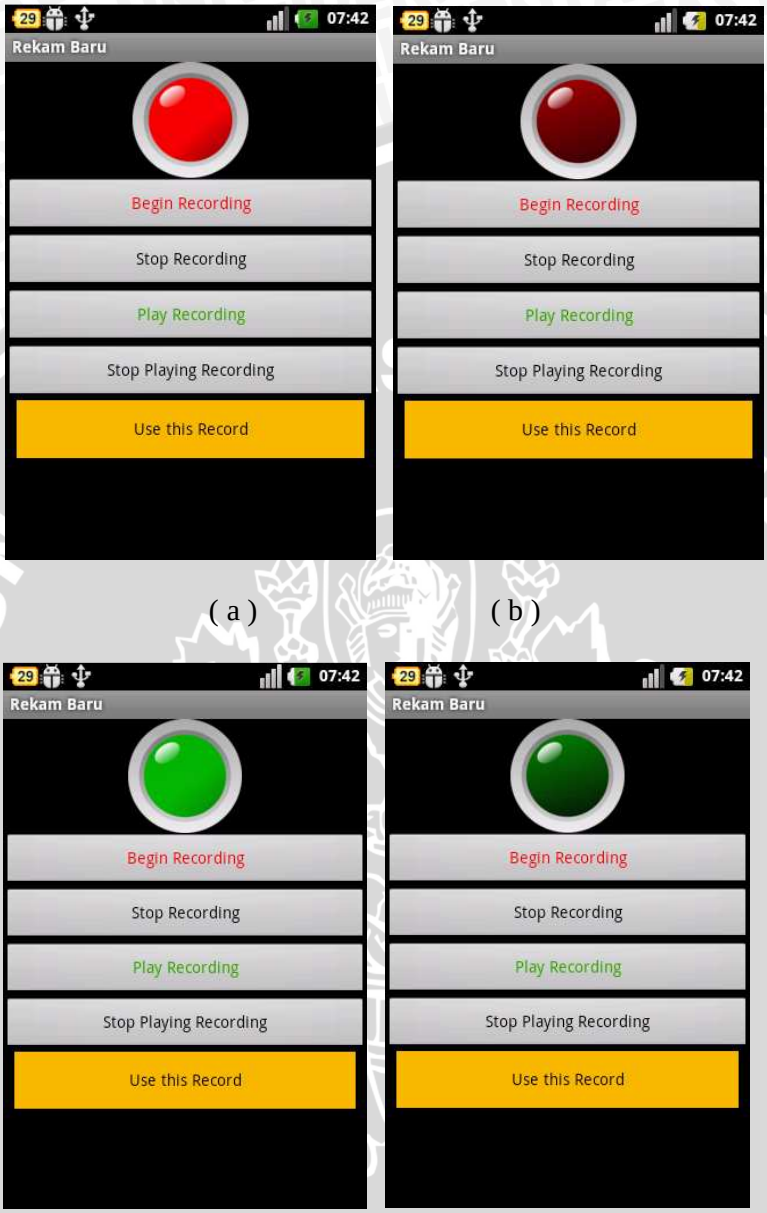
1. Membuat obyek mAudioPlayer dan mReverb.
2. Memberikan efek pada mAudioPlayer dengan fungsi `attachAuxEffect` dengan melewatkan ID dari mEnvironmentalReverb melalui `(mEnvironmentalReverb.getId());`
3. Mengatur besarnya pengaruh efek pada keluaran menggunakan perintah `setAuxEffectSendLevel` dengan rentang nilai 0.0 – 1.0.
4. Mengaktifkan mEnvironmentalReverb.
5. Membuat 5 slider untuk mengatur parameter dari reverb.
6. Parameter yang diatur adalah :
    a. Room Level
    b. Room High Frequency Level
    c. Decay High Frequency Ratio
    d. Reverb Level
    e. Room Density

4.3.5 Implementasi Antarmuka Program

Gambar 4.6 Tampilan antarmuka perekam suara
Sumber: Perancangan

Menu utama pada gambar 4.6 menggunakan beberapa *container* untuk tulisan dan gambar, serta dua buah *button* yang masing-masing mendeskripsikan tombol membuka menu rekam baru, dan pilih *file*.

4.3.5.1 Antarmuka Perekam Suara



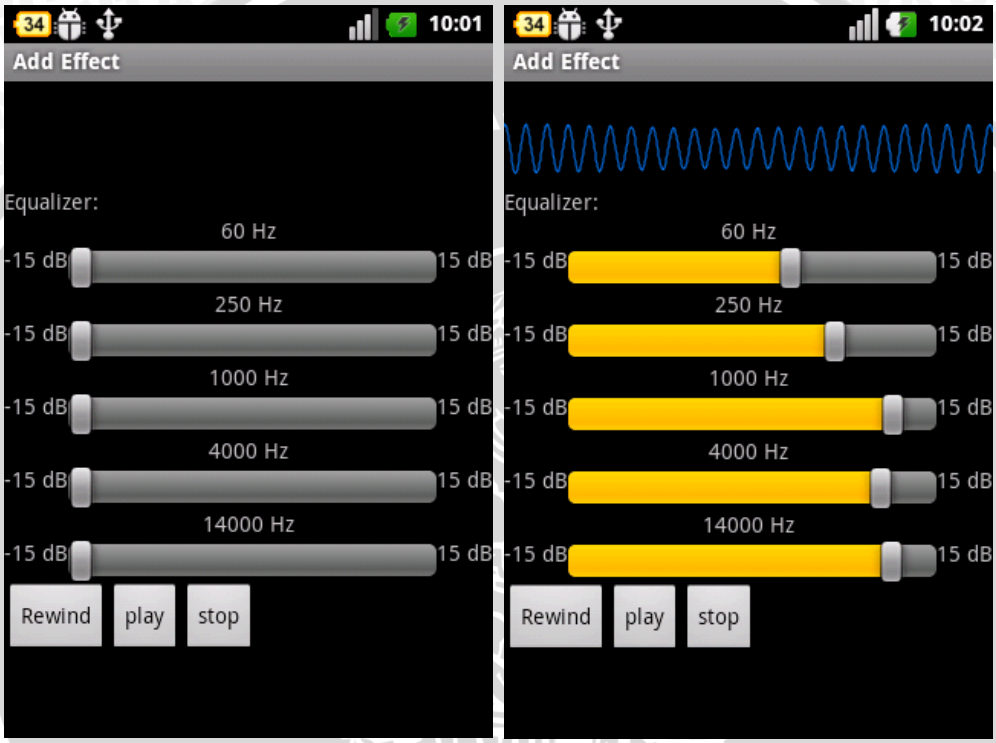
Gambar 4.7 Tampilan antarmuka perekam suara
Sumber: Perancangan

Penjelasan dari potongan antarmuka perekam suara pada Gambar 4.7 yaitu:

1. Sebuah *container* dari gambar led yang akan berubah ketika tombol ditekan.

2. Lima buah *button* yang masing-masing digunakan untuk mendeskripsikan tombol *Begin Recording, Stop Recording, Play Recording, Stop Playing Recording,* dan *Use This Record.*

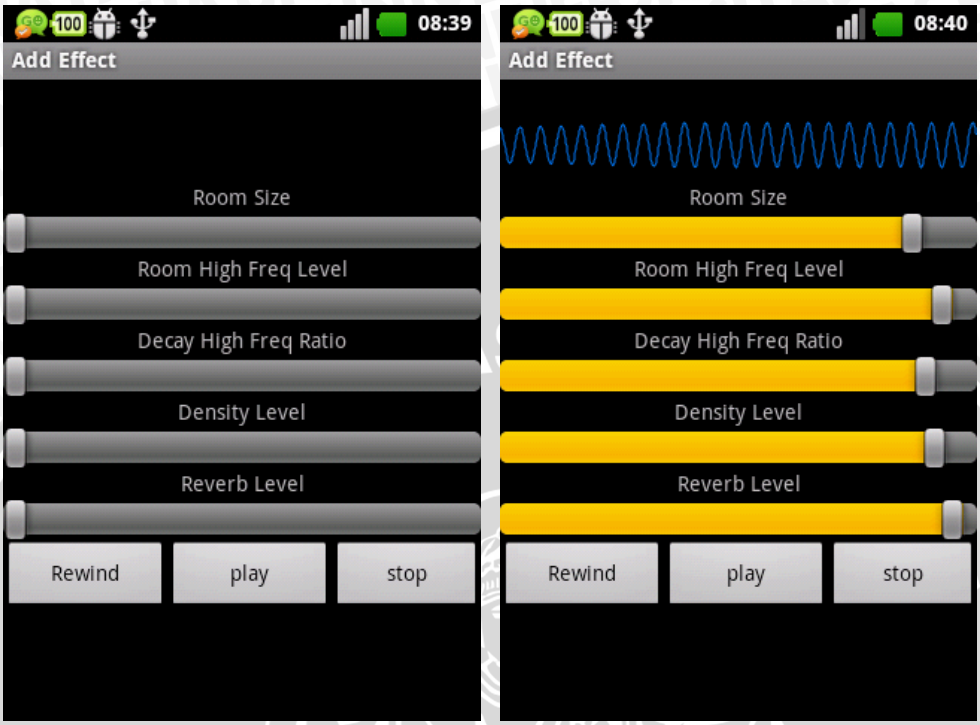
4.3.5.2 Antarmuka *Equalizer*



Gambar 4.8 Tampilan antarmuka *equalizer*
Sumber: **Perancangan**

1. Sebuah *Visualizer* untuk menampilkan spektrum masukan suara.
2. Lima buah SeekBar mendeskripsikan tombol geser (*slider*) untuk mengendalikan data tingkat penguatan frekuensi beserta elemen-elemennya.
3. Tiga buah Button yang masing-masing mendeskripsikan tombol gulung (rewbtn), mulai (startbtn) dan berhenti (stopbtn) beserta elemen-elemennya.

4.3.5.3 Antarmuka *Reverb*



Gambar 4.9 Tampilan antarmuka *Reverb*
Sumber: Perancangan

Penjelasan dari potongan antarmuka *Reverb* pada Gambar 4.9 yaitu:

1. Sebuah *Visualizer* untuk menampilkan spektrum masukan suara.
2. Lima buah SeekBar yang masing-masing mendeskripsikan tombol geser (*slider*) untuk mengendalikan *Room Size, Room High Freq Level, Decay High Freq Ratio, Density Level,* dan *Reverb Level.*
3. Tiga buah Button yang masing-masing mendeskripsikan tombol gulung (rewbtn), mulai (startbtn) dan berhenti (stopbtn) beserta elemen-elemennya.

BAB V

PENGUJIAN DAN ANALISIS

Pada bab ini dilakukan proses pengujian dan analisis terhadap aplikasi *single track audio processor* pada *smartphone* berbasis *android* yang telah dibangun. Proses pengujian dilakukan untuk mengetahui apakah aplikasi yang dibuat sudah bekerja dengan baik dan sesuai dengan perancangan atau bahkan terjadi suatu kegagalan. Pengujian yang dilakukan pada bab ini akan dibagi menjadi beberapa hal berikut :

1. Pengujian mikrofon perangkat
2. subsistem perekam suara
3. Pengujian subsistem *equalizer*
4. Pengujian subsistem *reverb*

5.1 Pengujian Mikrofon

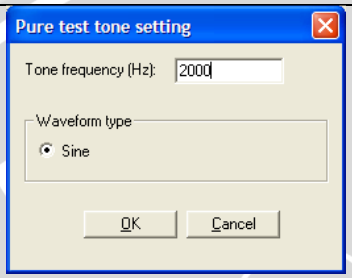
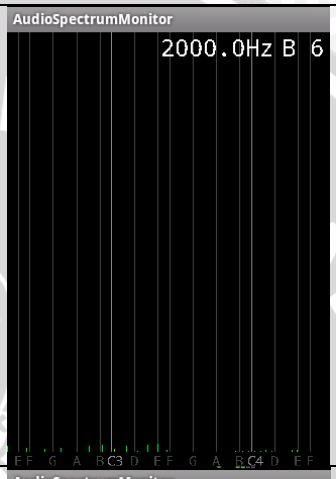
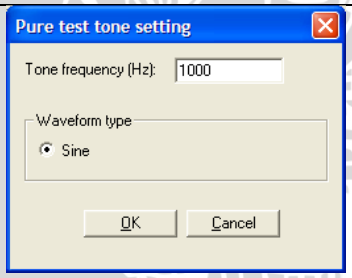
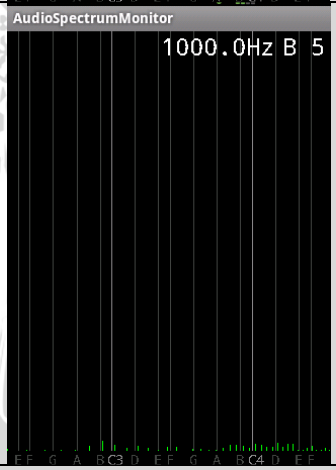
Pengujian ini bertujuan untuk mengetahui tanggapan frekuensi dari mikrofon perangkat bergerak. Pengujian dilakukan dengan cara menggunakan sumber suara dengan frekuensi yang dibangkitkan untuk setiap pengujiannya. Sumber suara berupa gelombang sinus dengan frekuensi tertentu agar mudah untuk mengidentifikasi hasilnya. Pengujian ini secara deskriptif dapat dijelaskan yaitu melakukan pengujian dengan membunyikan sumber suara dan mengetahuinya tanggapannya melalui program pihak ketiga yang berfungsi sebagai osciloskop yang sumber masukannya berasal dari mikrofon.

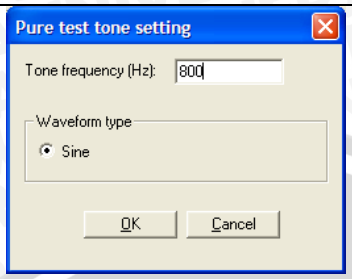
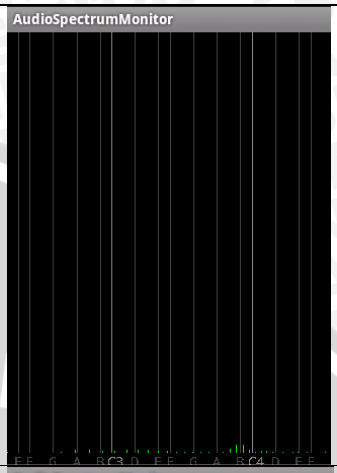
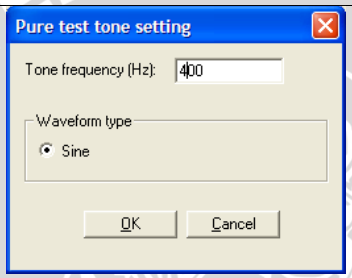
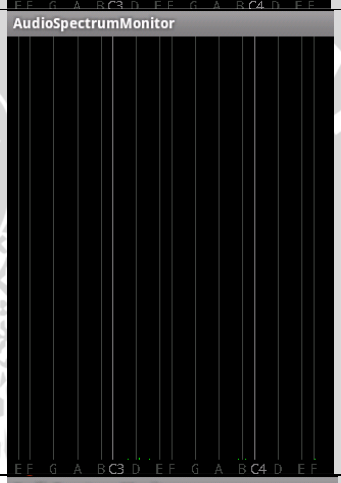
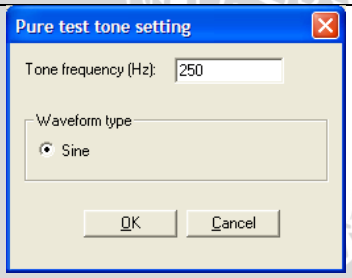
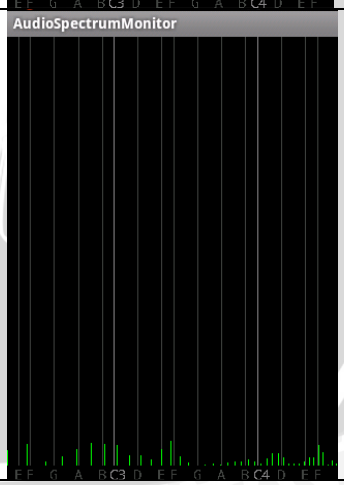
Program osciloskop yang digunakan merupakan aplikasi pihak ketiga yang bisa menerima masukan dari mikrofon. Sumber suara dibangkitkan, kemudian

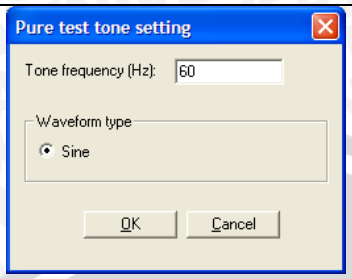
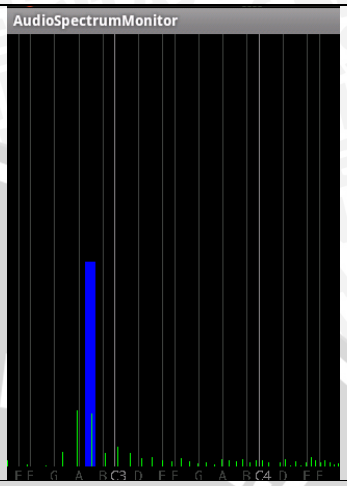
akan menampilkan hasil tanggapan frekuensi mikrofon dari sumber suara yang dimaksud.

Hasil pengujian dapat ditunjukkan didalam tabel 5.1.

Tabel 5.1 Data hasil tanggapan mikrofon

No.	Sampel Frekuensi	Masukan	Tanggapan	Pembacaan
1	2000 Hz			2000.0 Hz
2	1000 Hz			1000.0 Hz

No.	Sampel Frekuensi	Masukan	Tanggapan	Pembacaan
3	800 Hz			N/A
4	400 Hz			N/A
5	250 Hz			N/A

No.	Sampel Frekuensi	Masukan	Tanggapan	Pembacaan
6	60 Hz			N/A

Dari hasil pengujian pada Tabel 5.1 , didapatkan hasil mikrofon tidak dapat merespon masukan dari sumber suara dengan frekuensi di bawah 1000 Hz.

5.2 Pengujian Subsistem Perekam Suara

Pengujian ini bertujuan untuk mengetahui ketepatan fungsi dari perekam suara. Pengujian dilakukan dengan cara menggunakan beberapa sumber suara yang sama untuk setiap pengujiannya. Sumber suara berupa gelombang sinus dengan frekuensi tertentu agar mudah untuk mengidentifikasi hasilnya. Pengujian dapat dibagi menjadi beberapa skenario seperti berikut :

1. Pengujian perekam dan memutar hasilnya secara langsung.
2. Pengujian perekam dan membuka hasilnya di editor suara.

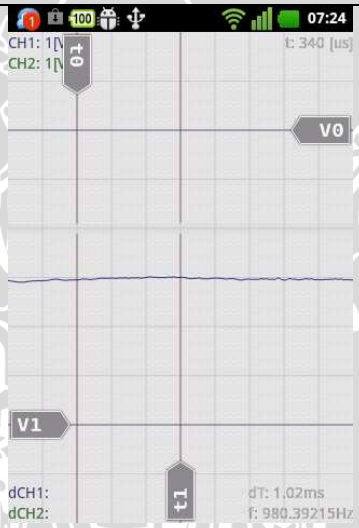
5.2.1 Pengujian Subsistem Perekam Suara Skenario 1

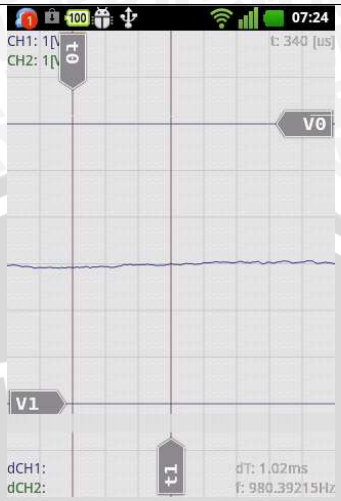
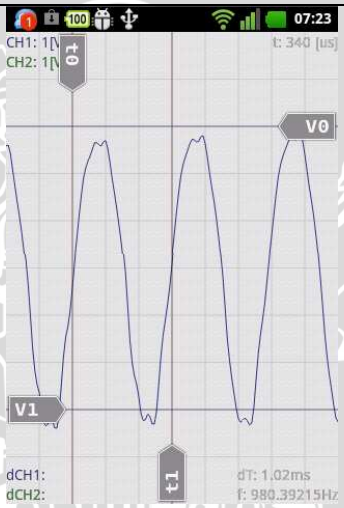
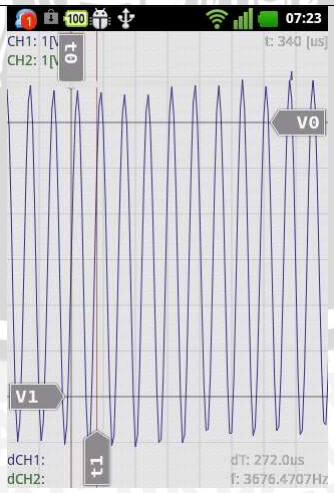
Pengujian aplikasi skenario 1 ini secara deskriptif dapat dijelaskan yaitu melakukan pengujian dengan membunyikan sumber suara dan merekamnya,

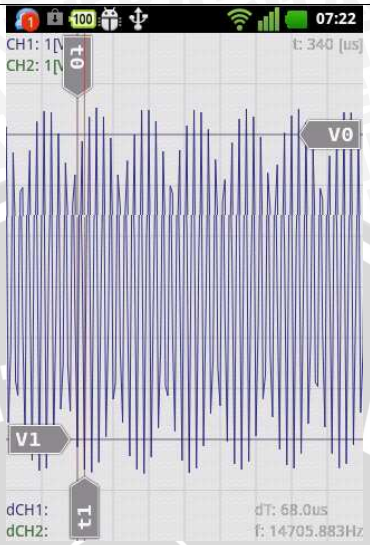
kemudian membandingkan hasil spektrum sumber suara dengan spektrum hasil perekaman. Jadi misalnya menggunakan sumber suara gelombang sinus dengan frekuensi 1000Hz, maka akan dibandingkan dengan tanggapan frekuensi hasil perekaman.

Pengujian dilakukan dengan menggunakan aplikasi osiloskop pihak ketiga pada *Android*. sumber suara dengan frekuensi yang beragam. Hasil pengujian dapat ditunjukkan didalam tabel 5.2.

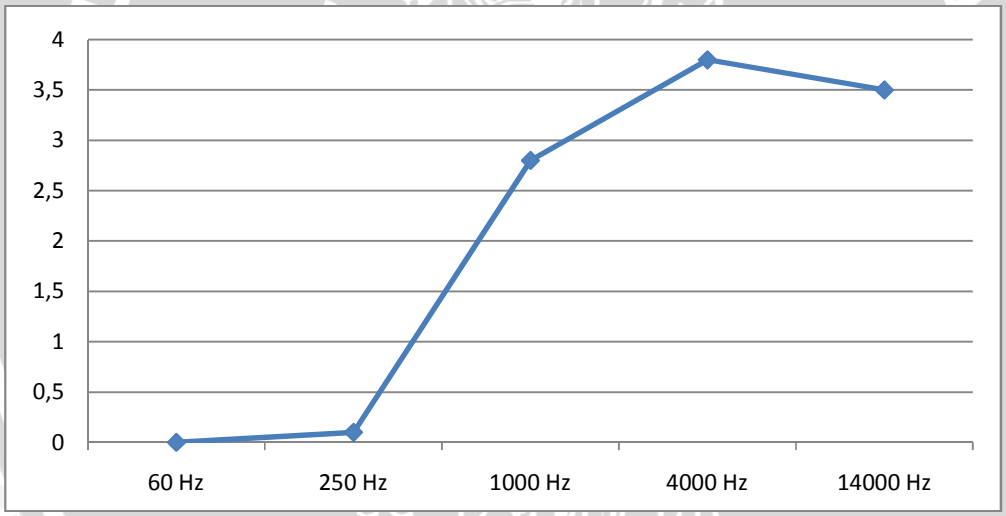
Tabel 5.2 Data hasil perekam suara

No	Frekuensi Sumber Suara	Hasil Rekaman	Frekuensi yang Ditangkap
1	60 Hz		Tidak Terbaca

No	Frekuensi Sumber Suara	Hasil Rekaman	Frekuensi yang Ditangkap
2	250 Hz		Tidak Terbaca
3	1000 Hz		1000 Hz
4	4000 Hz		4000 Hz

No	Frekuensi Sumber Suara	Hasil Rekaman	Frekuensi yang Ditangkap
5	14000 Hz		14000 Hz

Sumber: Pengujian



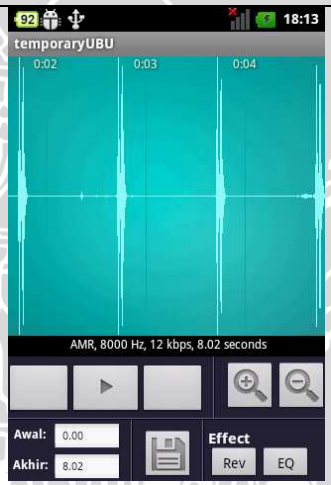
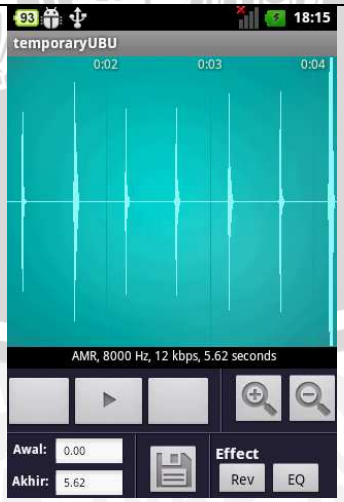
Dari hasil pengujian pada Tabel 5.2 , didapatkan hasil mikrofon tidak dapat merespon masukan dari sumber suara dengan frekuensi di bawah 1000 Hz. Hal ini dikarenakan keterbatasan penangkapan frekuensi dari mikrofon. Untuk frekuensi yang dapat ditangkap oleh mikrofon, dapat diambil kesimpulan bahwa mikrofon dapat melakukan proses perekaman dengan baik dengan tanggapan frekuensi yang mendekati sumber suara aslinya.

5.2.2 Pengujian Subsistem Perekam Suara Skenario 2

Pengujian aplikasi skenario 2 ini secara deskriptif dapat dijelaskan yaitu melakukan pengujian dengan membunyikan sumber suara dan merekamnya kemudian membukanya menggunakan editor dari program, kemudian membandingkan hasil spektrum sumber suara dengan spektrum hasil perekaman.

Pengujian dilakukan dengan menggunakan sumber suara dengan pola yang berbeda. Hasil pengujian dapat ditunjukkan didalam tabel 5.3.

Tabel 5.3 Data Hasil Perekam Suara pada Editor

No	Sumber suara	Hasil Rekaman	Kesesuaian hasil rekaman
1	Metronom 60 bpm		Ya
2	Metronom 120 bpm		Ya

No	Sumber suara	Hasil Rekaman	Kesesuaian hasil rekaman
3	Gelombang sinus 1000 Hz		Ya

Sumber: Pengujian

Dari hasil pengujian pada Tabel 5.3, didapatkan hasil yang menampilkan kesesuaian sumber suara dengan hasil dari perekam. Hal ini dapat diambil kesimpulan bahwa mikrofon dapat melakukan proses perekaman dengan baik dan mampu menangkap suara sesuai dengan ada pada sumber bunyi.

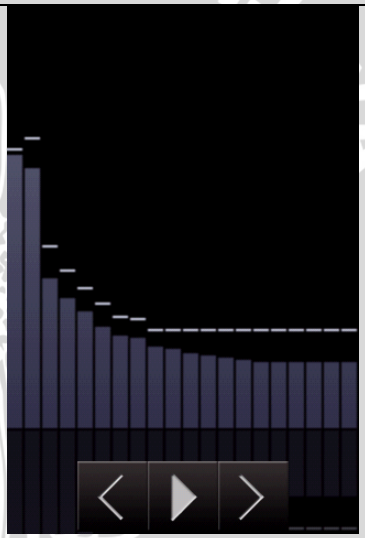
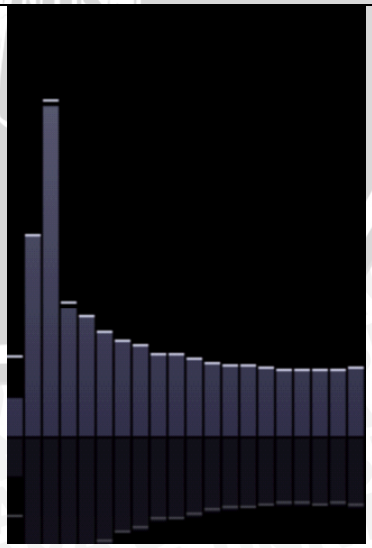
5.3 Pengujian Subsistem *Equalizer*

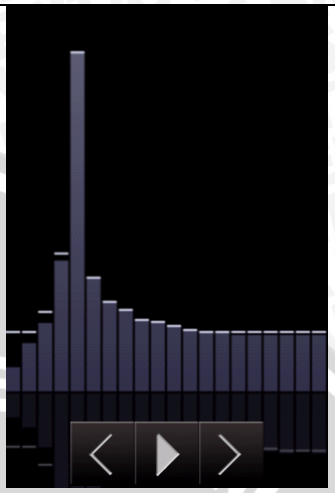
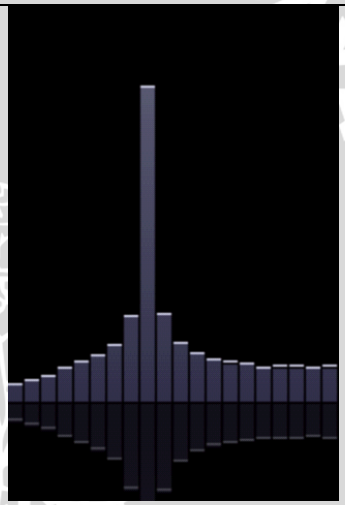
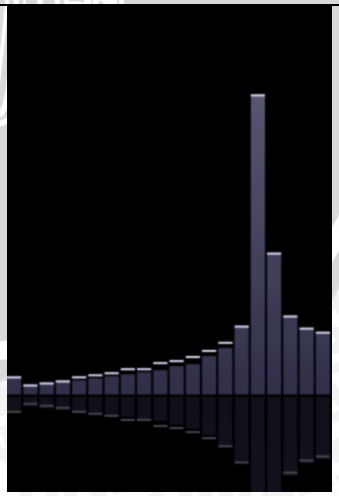
Pengujian ini bertujuan untuk mengetahui pengaruh *equalizer* terhadap file audio. Pengujian dilakukan dengan cara menggunakan beberapa sumber suara yang sama untuk setiap pengujiannya. Sumber suara berupa gelombang sinus dengan frekuensi yang berdekatan dengan *band* dari *equalizer* agar mudah untuk mengidentifikasi hasilnya. Pengujian dilakukan dengan menggunakan 5 file sumber suara yaitu sumber suara dengan frekuensi 60Hz, 250Hz, 1000Hz, 4000Hz, dan 14000Hz.

Pengujian dilakukan melalui 2 skenario. Skenario pertama, sumber suara dari *smartphone*, kemudian menggunakan aplikasi *visualizer* dari *smartphone* untuk menangkap tanggapan frekuensinya. Skenario kedua, sumber suara berasal dari *smartphone*, kemudian gelombang suara ditangkap oleh mikrofon profesional untuk diolah ke dalam komputer.

Hasil pengujian skenario pertama dapat ditunjukkan didalam tabel 5.4.

Tabel 5.4 Data Tanggapan *Equalizer*

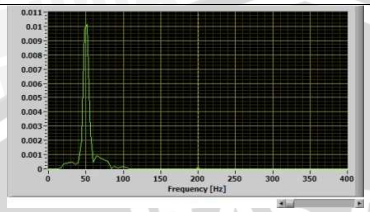
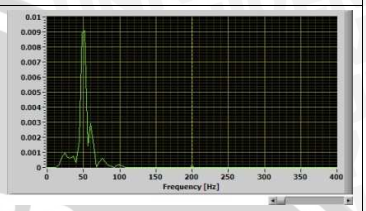
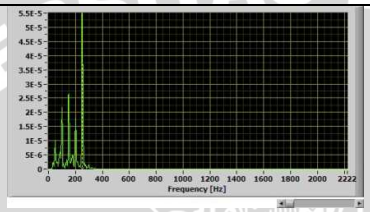
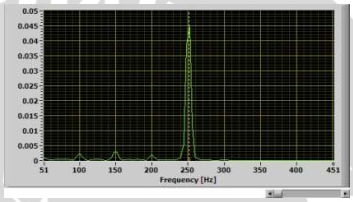
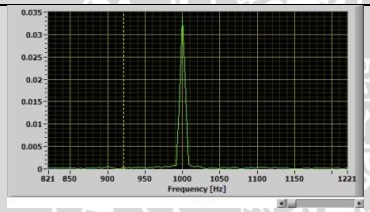
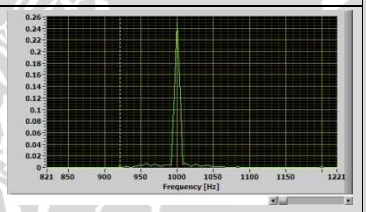
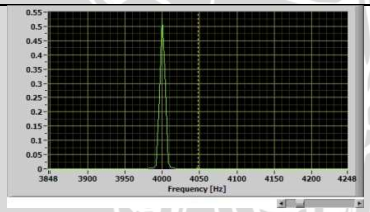
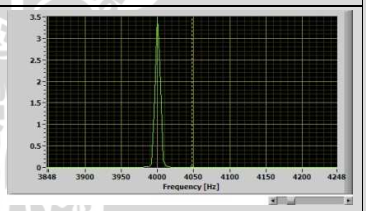
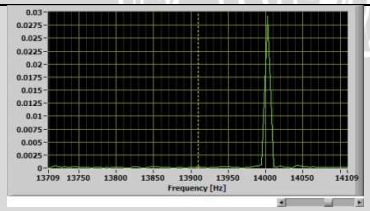
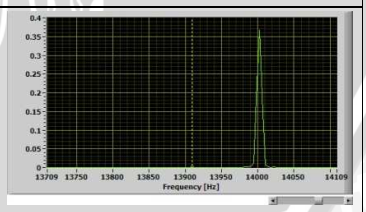
No	Frekuensi Sumber suara	Frekuensi yang dikuatkan	Hasil penguatan Rekaman
1	60 Hz	60 Hz	
2	250 Hz	230 Hz	

No	Frekuensi Sumber suara	Frekuensi yang dikuatkan	Hasil penguatan Rekaman
3	1000 Hz	910 Hz	
4	4000 Hz	3600 Hz	
5	14000 Hz	14000 Hz	

Sumber: Pengujian

Hasil pengujian skenario kedua dapat ditunjukkan didalam tabel 5.5.

Tabel 5.5 Data Tanggapan Frekuensi *Equalizer*

No	Frekuensi Sumber suara	Tanggapan hasil perekaman pada saat frekuensi dilemahkan	Tanggapan hasil perekaman pada saat frekuensi dikuatkan
1	60 Hz		
2	250 Hz		
3	1000 Hz		
4	4000 Hz		
5	14000 Hz		

Sumber: Pengujian

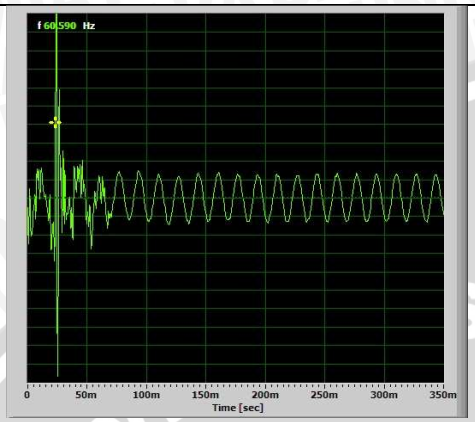
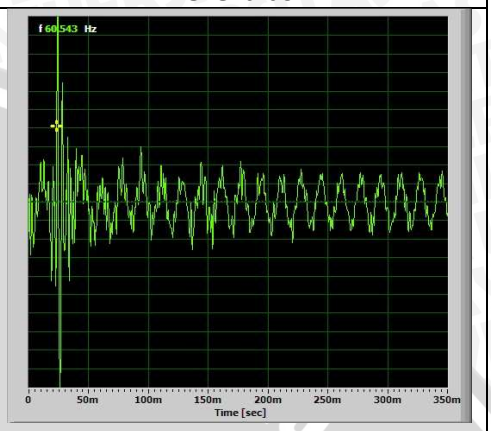
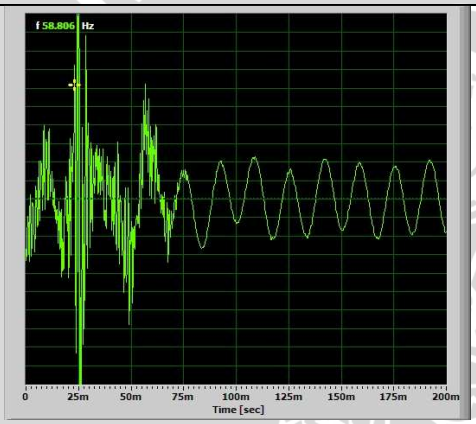
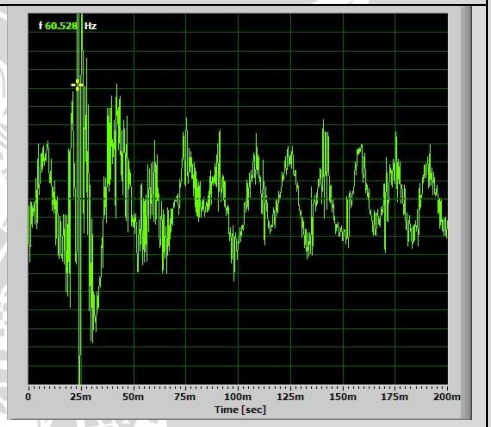
Dari hasil pengujian pada Tabel 5.4, didapatkan hasil peningkatan grafik frekuensi yang signifikan pada frekuensi yang mendekati dengan frekuensi sumber suara. Sedangkan untuk table 5.5, tidak didapatkan hasil yang signifikan pada frekuensi 60 Hz, hal ini dikarenakan keterbatasan pelepasan frekuensi dari speaker *smartphone* dan penangkapan frekuensi dari mikrofon. Untuk frekuensi yang lain, dapat diambil kesimpulan bahwa proses penguatan frekuensi pada *equalizer* berfungsi dengan baik.

5.4 Pengujian Subsistem *Reverb*

Pengujian ini bertujuan untuk mengetahui pengaruh *reverb* terhadap file audio. Pengujian dilakukan dengan cara menggunakan beberapa sumber suara ritmis dengan tempo yang sama untuk setiap pengujiannya. Sumber suara berupa suara ketukan metronome dengan kecepatan 60 ketukan per menit agar mudah untuk mengidentifikasi hasilnya.

Hasil pengujian dapat ditunjukkan didalam tabel 5.6.

Tabel 5.6 Data Hasil Pengujian Reverb

No	Potongan gelombang sebelum reverbrasi	Potongan gelombang sesudah reverbrasi
1		
2		

Sumber: Pengujian

Dari hasil pengujian pada Tabel 5.6, didapatkan hasil perubahan dari keadaan setelah ketukan. Pada keadaan sebelum diberi pengaruh reverberasi, terlihat keadaan gelombang stabil tanpa gangguan setelah terjadi ketukan. Pada keadaan setelah mendapat pengaruh dari reverberasi, tampak keadaan gelombang tidak stabil akibat pengaruh dari efek pantul ruangan yang diberikan oleh reverberasi. Dapat diambil kesimpulan bahwa proses reverberasi berfungsi dengan baik dan memiliki pengaruh pada gelombang suara masukan.

BAB VI

PENUTUP

6.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut:

1. Dalam merancang sebuah *single track audio recorder* pada *smartphone Android,* didapatkan hasil yang tidak maksimal karena setelah dilakukan pengujian, didapatkan bahwa tanggapan frekuensi dari mikrofon handset yang terbatas pada frekuensi tertentu.
2. Penambahan fitur pemilihan file dapat menggantikan kekurangan dari mikrofon melalui sumber suara eksternal yang dapat diolah sama dengan fitur perekam yang disediakan.
3. Dalam menerapkan pengolahan sinyal suara berupa *equalizer,* setelah dilakukan pengujian didapatkan hasil pengolahan frekuensi dapat dilakukan dengan baik. Hal ini dibuktikan dengan terdapat penguatan frekuensi pada pita frekuensi yang telah ditentukan oleh perangkat lunak pada *smartphone Android*.
4. Dalam menerapkan pengolahan sinyal suara berupa *reverb*, didapatkan hasil bahwa pemberian efek ruang dapat dilakukan secara baik oleh *smartphone Android*. Hal ini dibuktikan dengan pembangkitan efek reverberasi dimana pada pengujian menimbulkan gema dari sumber suara yang membuat suara seolah-olah berada pada sebuah ruangan.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan ini antara lain :

1. Untuk pengembangan lebih lanjut diberikan tambahan dukungan dari perangkat luar untuk menambahkan mikrofon eksternal dengan lebar jangkauan frekuensi yang lebih besar sebagai media masukan perekam, untuk menggantikan mikrofon dalam yang hanya dapat menangkap sumber suara pada frekuensi tertentu.
2. Untuk meningkatkan kemampuan *smartphone* sebagai media pemrosesan sinyal suara yang bersifat portabel, dibutuhkan peningkatan kualitas mikrofon internal sebagai media penangkap sinyal suara mengingat perkembangan dari *smartphone* yang mampu melakukan pemrosesan sinyal suara dengan baik.

DAFTAR PUSTAKA

Amatriain, Xavier. 2002. *DAFX-Digital Audio Effect*. USA: Wiley

Hashimi, Sayed dkk. 2010. *Pro Android 2.* NewYork : Springer.

Huang, Ying, 2009. *Begin Android Journey in Hours*.

Katz, David J., Gentile, Rick. 2006. *Embedded Media Processing.* USA : Elsevier

Loughlin, Ian Mc., 2009. *Applied Speech and Audio Processing*. New York: Cambridge University Press

Murphy, Mark L., 2010. *Beginning Android 2*. New York: Spinger

Murphy, Mark L., 2008. *The Busy Coder's Guide to Android Development.* USA: Commonsware

Xuguang, Haung, 2009. *An Introduction to Android*. Database Lab. Inha University

Zölzer, Udo. 2008. *Digital Audio Signal Processing-Second Edition*. Chippenham: JohnWiley & Sons Ltd