Rancang Bangun Aplikasi Efek Gitar Dengan Kendali Pergerakan Menggunakan Sensor Akselerometer Pada Smartphone Android

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Billy Febiar
NIM: 115060800111059

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

RANCANG BANGUN APLIKASI EFEK GITAR DENGAN KENDALI PERGERAKAN MENGGUNAKAN SENSOR AKSELEROMETER PADA SMARTPHONE ANDROID

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Billy Febiar
NIM: 115060800111059

Skripsi ini telah diuji dan dinyatakan lulus pada
7 Januari 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Agi Putra Kharisma, S.T., M.T.	Sabriansyah Rizqika Akbar, S.T., M.Eng.
NIK: 2013048604301001	NIP: 198208092012121004

Mengetahui
Ketua Program Studi Informatika

Drs. Marji, M.T
NIP: 19670801 199203 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 7 Januari 2016

Billy Febiar

NIM: 115060800111059

KATA PENGANTAR

Puji Syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena hanya dengan rahmat dan karuniaNya, penulis dapat menyelesaikan skripsi dengan judul "Rancang Bangun Aplikasi Efek Gitar dengan Kendali Pergerakan Menggunakan Sensor Akselerometer pada Smartphone Android". Melalui kesempatan ini, penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan batuan dan dukungan selama penulisan skripsi ini, diantaranya:

1. Kedua orang tua penulis Ir Febri Gumelar dan Maria Sagita Rosalina dan adik penulis serta seluruh keluarga besar atas segala doa, nasehat dan dukungan yang diberikan hingga terselesainya skripsi ini.

2. Bapak Agi Putra Kharisma, S.T., M.T. dan Bapak Sabriansyah Rizqika Akbar, S.T., M.Eng. selaku dosen pembimbing I dan dosen pembimbing II yang telah banyak memberikan bimbingan, ilmu dan saran dalam penyusunan skripsi ini.

3. Bapak Aswin Suharsono, S.T., M.T. selaku dosen pembimbing akademik yang telah memberikan bimbingan dan saran selama masa perkuliahan.

4. Seluruh Civitas Akademika Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Prodi Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.

5. Sahabat-sahabat penulis dari awal perkuliahan Herdian Adi Winarno S.Kom, Nur Muhammad Rasyid S.Kom, dan Alfian Mukmin Ali S.Kom atas bantuan dan dukungannya.

6. Teman-teman informatika kelas I dan teman-teman seperdotaan penulis Rio, Adi, Heddy, Irfandi, Tadho, Eka dan Ando yang telah menghibur penulis dari kesepian di malam hari.

7. Semua pihak yang tidak cukup untuk disebutkan satu persatu atas segala keramahan, kebaikan, bantuan, doa dan peran sertanya dalam penyelesaian skripsi ini.

Semoga jasa dan amal baik dari semua pihak mendapat balasan dari Tuhan Yang Maha Esa. Dengan segala kerendahan hati, penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan sebagai karya penelitian yang baik. Untuk itu, saran dan kegiatan pengembangan penelitian akan sangat berguna untuk kemajuan penulis. Akhirnya, semoga skripsi ini bermanfaat dan berguna bagi pembaca terutama mahasiswa Fakultas Ilmu Komputer Universitas Brawijaya.

Malang, 7 Januari 2016

Penulis

Email: billy7sign@gmail.com

ABSTRAK

Sebagian besar pemain gitar elektrik menggunakan efek gitar untuk menghasilkan karakter suara pada gitarnya. Efek gitar merupakan perangkat elektronik yang dapat memodifikasi nada, *pitch*, atau suara dari gitar elektrik. Karena fungsinya yang sangat penting, membuat harga dari perangkat ini menjadi relatif mahal. Dengan adanya perangkat smartphone yang fiturnya sangat kompleks kita dapat membuat aplikasi efek gitar dan menanamkannya pada perangkat android kita sehingga kita tidak perlu membeli efek gitar yang harganya cukup mahal.

Aplikasi-aplikasi efek gitar yang sudah ada di play store masih belum dapat memaksimalkan fungsi dari smartphone android itu sendiri. Penelitian ini bertujuan untuk merancang dan mengimplementasikan aplikasi efek gitar dengan memanfaatkan sensor akselerometer yang ada pada smartphone android dimana smartphone nantinya akan diletakkan pada bagian *headstock* gitar sehingga diharapkan dapat mendeteksi dan merespon pergerakan dari pemain gitar.

Dari hasil penelitian yang dilakukan, Aplikasi efek gitar yang dibangun ini berhasil diimplementasikan pada smartphone android dan telah sesuai dengan perancangan dan kebutuhan pengguna. Hasil pengujian fungsionalitas menunjukkan sistem telah sesuai dengan spesifikasi kebutuhan fungsional. Hasil pengujian usabilitas berdasarkan 5 aspek *usability testing* mencapai rata-rata sebanyak 81.9% yang menunjukkan aplikasi layak diterima dan memudahkan pengguna dengan hasil sangat memuaskan.

Kata kunci: Efek Gitar, Akselerometer, dan Android

ABSTRACT

Most of the electric guitar players use a guitar effect to produce sound character on the guitar. Guitar effect is an electronic device that can modify the tone, pitch, or the sound of an electric guitar. Because its function is very important, make the price of these devices to be relatively expensive. With the feature of smartphone device that are very complex we can make guitar effect application and installing in our android device so we do not need to buy a guitar effect that are quite expensive.

Guitar effects applications that already exist in the play store still has not been able to maximize the functionality of the android smartphone itself. This study aims to design and build guitar effect application by utilizing the accelerometer sensor on android smartphone where smartphone will be place on guitar headstock which is expected to detect and respond to the movement of a guitar player.

From the result conducted, guitar effect application has been built and implemented successfully on android smartphone and in accordance with specification of functional requirements. System functionality test indicate that system in accordance with specification of functional requirements. Usability testing result based on five aspects of usability testing reached an average of 81.9% which shows the application accepted and easier for users with very satisfactory result.

Keyword : Guitar Effect, Accelerometer, and Android.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Gitar merupakan alat musik yang dibuat dari kayu yang dimainkan dengan cara memetik keenam buah tali atau senarnya. Keenam senar terikat pada *pegs* atau pemutar senar yang ditarik sepanjang badan gitar. *Pegs* ini digunakan untuk menyetel (menyetem) gitar. Secara umum, gitar terbagi atas dua jenis, yaitu akustik dan elektrik (Chandra, 2012).

Gitar elektrik merupakan salah satu jenis gitar yang menggunakan alat yang disebut juga *pickup* untuk mengubah bunyi atau getaran dari senar gitar menjadi arus listrik yang kemudian akan dikuatkan kembali menggunakan *speaker*. Gitar elektrik pertama kali dibuat pada tahun 1932 oleh Adolphus Ricken Backer. Suara gitar listrik dihasilkan oleh getaran senar gitar yang mengenai kumparan yang ada di bagian *body* gitar yang biasa disebut *pick up* (Nugroho, 2009).

Sebagian besar pemain gitar elektrik menggunakan efek gitar untuk menghasilkan karakter suara pada gitarnya. Efek gitar merupakan perangkat elektronik yang dapat memodifikasi nada, *pitch*, atau suara dari gitar elektrik. Terdapat 2 macam jenis efek gitar, yaitu analog dan digital. Efek gitar analog hanya memuat satu atau paling banyak dua jenis suara dalam satu unitnya sedangkan efek gitar digital dapat memuat banyak jenis suara dalam satu unit (Noviyandhika, 2009).

Karena fungsinya yang sangat penting, membuat harga dari perangkat/alat ini menjadi relatif mahal. Bagi mereka yang berkecukupan, hal ini mungkin bukanlah menjadi suatu masalah. Tetapi bagi mereka yang memiliki pendapatan ekonomi menengah kebawah, hal ini merupakan masalah yang sangat serius terlebih lagi diperlukan cukup dana untuk membeli *Sound Out* (*Amplifier*) dan gitar elektrik itu sendiri.

Akan tetapi dengan kemajuan teknologi, hal ini tidaklah menjadi suatu masalah. Terutama dengan adanya perangkat smartphone android yang fiturnya sangat kompleks sehingga kita dapat membuat aplikasi efek gitar kita sendiri dan menanamkannya pada perangkat android kita sehingga kita tidak perlu lagi membeli efek gitar yang harganya cukup mahal.

Sebenarnya aplikasi efek gitar sudah banyak tersedia. Akan tetapi aplikasi-aplikasi tersebut belum dapat memaksimalkan fungsi dari smartphone itu sendiri. Aplikasi tersebut hanya mengubah suara gitar dan pengguna perlu menekan tombol secara manual untuk berinteraksi dengan aplikasi.

Oleh karena itulah, penulis mempunyai gagasan untuk memanfaatkan fungsi akselerometer yang ada pada smartphone Android. Akselerometer merupakan suatu sensor yang terdapat pada smartphone yang dapat mengukur percepatan yang dialami oleh sensor (Graham, 2012). Dengan adanya sensor akselerometer ini diharapkan dapat mendeteksi dan merespon pergerakan dari pemain gitar

sehingga pemain gitar dapat mengganti jenis efek gitar dan mengatur intensitas tiap jenis efek hanya dengan kendali dari pergerakan.

1.2 Rumusan masalah

Berdasarkan pada permasalahan yang telah dijelaskan pada bagian latar belakang, maka rumusan masalah dapat disusun sebagai berikut :

1. Bagaimana merancang aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer pada smartphone android?
2. Bagaimana mengimplementasikan aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer pada smartphone android?
3. Bagaimana tingkat usabilitas aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer pada smartphone android yang telah diimplementasikan?

1.3 Tujuan

Tujuan dari penelitian ini adalah untuk merancang dan mengimplementasi aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer yang dapat menghasilkan efek suara dengan intensitas tertentu dan mengganti jenis efek suara hanya dengan menggunakan pergerakan dari pemain gitar.

1.4 Manfaat

Manfaaat yang dapat diperoleh dari penelitian ini adalah :

1. Membantu penggunanya (pemain gitar) untuk menghasilkan suara gitar yang lebih variatif.
2. Membantu penggunanya (pemain gitar) untuk berinteraksi secara langsung dengan aplikasi hanya dengan menggunakan pergerakan.
3. Sebagai alternatif dari penggunaan hardware efek gitar yang harganya relatif mahal.

1.5 Batasan masalah

Agar permasalahan yang dirumuskan dapat lebih terfokus, maka pada penelitian ini dibatasi dalam hal :

1. Terdapat 3 jenis efek suara yang akan digunakan yaitu wah-wah, *tremolo* dan distorsi.

1.6 Sistematika pembahasan

Penyusunan skripsi ini menggunakan kerangka pembahasan yang tersusun sebagai berikut :

Bab I **Pendahuluan**

Bab ini menjelaskan latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan.

Bab II **Tinjauan Pustaka**

Bab ini berisi kajian pustaka, referensi atau sumber-sumber yang berhubungan dengan permasalahan dalam skripsi baik mengenai software maupun hardware serta teori yang diperlukan dalam perancangan aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer

Bab III **Metode Penelitian**

Bab ini menjelaskan bagaimana metodologi untuk perancangan dan pembuatan aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer.

Bab IV **Analisis dan Perancangan**

Bab ini menjelaskan analisis dan perancangan aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer yang dapat menjawab permasalahan yang telah diuraikan pada rumusan masalah.

Bab V **Implementasi**

Bab ini berisi tentang implementasi aplikasi yang dibuat, meliputi penjelasan mengenai aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer.

Bab VI **Analisis dan Pengujian**

Bab ini berisi tentang teknik atau metode pengujian yang di lakukan pada aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer serta melakukan analisa terhadap hasil pengujian.

Bab VII **Penutup**

Bab ini menjelaskan kesimpulan yang dapat diambil dari penelitian skripsi disertai saran yang dapat dijadikan masukan untuk penelitian lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Efek Gitar

Efek Gitar merupakan perangkat elektronik yang memodifikasi nada, *pitch*, atau suara dari gitar listrik. Efek elektronik dan pemrosesan sinyal merupakan bagian penting dari nada gitar listrik yang digunakan dibanyak genre seperti *rock*, *pop*, *blues*, dan *metal*. Semua ini dimasukkan kedalam jalur sinyal antara instrumen listrik dan *amplifier* (Brewster, 2010).

Efek Gitar memodifikasi sinyal yang datang dari instrumen, kemudian menambahkan efek yang mengubah suara tersebut menjadi lebih menarik. Efek gitar juga digunakan oleh instrumen lainnya di music *rock*, *pop*, *blues* dan *metal*, seperti *keyboard* elektronik. Pemain *bass* yang menggunakan *bass* elektrik juga menggunakan efek *bass* yang dirancang untuk bekerja dengan nada frekuensi rendah dari bass.

Terdapat beberapa jenis efek gitar yang sering digunakan oleh para gitaris yaitu (Noviyandhika, 2009):

1. Wah-wah

   Efek ini menghasilkan suara "waoo..waoo" dan biasanya dimainkan dengan *foot pedal*, tetapi untuk beberapa efek digital menyediakan fitur *autowah* yang secara otomatis akan menghasilkan efek wah tanpa *foot pedal* dan untuk tempo *wah* dapat kita atur sesuai dengan tempo seperti saat kita menggunakannya dengan *foot pedal*.

2. *Compresor*

   Fungsi dari efek ini adalah menghasilkan *gain/volume* menjadi lebih stabil. *Compressor* ini akan memperbesar sinyal yang kurang kuat dan menahan sinyal yang terlalu kuat, sehingga sinyal keluar dengan *gain/volume* yang seimbang.

3. *Overdrive* dan *Distortion*

   Kedua efek ini adalah dua hal yang serupa tetapi tidak sama. Sesuai dengan namanya, *overdrive* adalah sinyal clean yang dikuatkan secara berlebih, sehingga menimbulkan suara yang sedikit terdistorsi. Sedangkan distorsi merupakan pemotongan, penguatan, dan penghancuran sinyal secara ekstrim, sehingga menghasilkan sound yang benar-benar rusak dan sangat terdistorsi.

4. *Equalizer* (EQ)

   Equalizer merupakan piranti yang mengolah dan bermain dengan frekuensi. Dengan menggunakan efek ini, kita bisa menambah dan memotong (untuk mengurangi *noise*) frekuensi-frekuensi tertentu, membuat *tone sound* menjadi lebih terasa dan juga dapat menambah dan mengurangi *high*, *middle* dan *low* dari *sound* yang kita inginkan.

5. *Modulation* (modulasi)

Terdapat beberapa efek yang termasuk dalam modulasi ini, antara lain *chorus*, *fianger*, *phaser*, *tremolo*, *pitch shifter* dan lain-lain.

6. Ambience

Yang termasuk didalam efek ini adalah *delay* dan *reverb*. *Delay* merupakan gema panjang yang berulang-ulang sedangkan *reverb* merupakan kumpulan *delay* yang pendek, banyak dan hampir tidak terdengar sehingga menciptakan efek gema di dalam ruangan.

Efek yang digunakan dalam penelitian ini hanya beberapa jenis, yaitu efek suara wah-wah, efek suara *tremolo* dan efek suara distorsi. Masing-masing jenis efek tersebut nantinya juga dapat diatur intensitas suaranya.

2.2 Sistem Operasi Android

Android adalah sistem operasi untuk telpon seluler yang berbasis Linux. Android menyediakan *platform* yang bersifat *open source* bagi para pengembang untuk menciptakan sebuah aplikasi (Andi, 2013). Perusahaan yang pertama kali mengembangkan sistem operasi mobile ini adalah Android Inc. Nama perusahaan itulah yang akhirnya digunakan sebagai nama dari sistem operasi Android ini.

Sistem operasi android memiliki beberapa fitur didalamnya sehingga memberikan kemudahan bagi *developer* untuk melakukan modifikasi dalam aplikasinya. Fitur-fitur android digaris besarkan sebagai berikut:

8. Dukungan *hardware*: Sensor akselerometer, kamera, kompas digital, sensor *proximity*, dan GPS (*Global Positioning System*).
9. *Multi touch*: Mendukung layar dengan dukungan *multi touch.*
10. *Multitasking*: Kemampuan untuk melaksanakan tugas secara bersamaan.

Dalam penelitian ini fitur android yang digunakan ialah sensor akselerometer. Sensor akselerometer digunakan untuk mendeteksi pergerakan yang telah dilakukan oleh pemain gitar.

2.2.1 Akselerometer

Akselerometer adalah alat yang digunakan untuk mengukur percepatan, mendeteksi dan mengukur getaran (vibrasi), dan mengukur percepatan akibat gravitasi (inklinasi) (Rahman, 2011). Sensor accelerometer mengukur percepatan akibat gerakan benda yang melekat padanya.Akselerometer dapat digunakan untuk mengukur dua jenis percepatan, yaitu percepatan *static* dan *dynamic*. Pengukuran *static* digunakan untuk mengukur percepatan terhadap gravitasi bumi, sedangkan percepatan d*ynamic* digunakan untuk mengukur percepatan dari objek yang bergerak (Haryanti, 2012).

Percepatan merupakan keadaan dimana terjadi perubahan kecepatan terhadap waktu. Bertambahnya suatu kecepatan dalam suatu rentang waktu

disebut juga percepatan (acceleration). Jika kecepatan semakin berkurang daripada kecepatan sebelumnya, disebut deceleration. Percepatan juga bergantung pada arah/orientasi karena merupakan penurunan kecepatan yang merupakan besaran *vector*. Berubahnya arah pergerakan suatu benda akan menimbulkan percepatan pula.

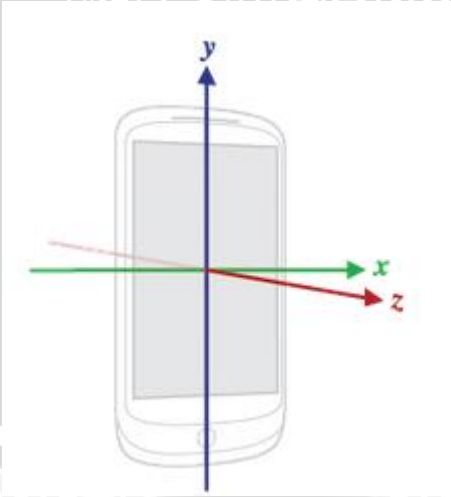
Pada Penelitian ini, sensor akselerometer pada smartphone android digunakan untuk mendeteksi pergerakan dari pemain gitar sehingga dapat mengganti jenis efek suara dari jenis yang satu ke jenis yang lainnya serta dapat mengatur intensitas dari setiap jenis efek suara.

2.2.2 Sistem Koordinat Sensor

Secara umum, sensor menggunakan standar sistem koordinat 3 sumbu standar seperti pada Gambar 2.1 untuk mengekspresikan nilai-nilai data (Developer, 2015). Ketika perangkat diposisikan dalam orientasi *default*, sumbu X adalah *horizontal* dan mengarah kekanan, sumbu Y adalah vertikal dan mengarah keatas, dan sumbu Z mengarah kearah luar dari layar. Pada sistem operasi android, koordinat kearah belakang layar memiliki nilai Z negatif.

Hal yang paling penting untuk dipahami tentang sistem koordinat ini adalah bahwa sumbu tidak bertukar saat orientasi layar perangkat dirubah. Perilaku ini adalah sama dengan perilaku pada sistem OpenGL.

Pada penelitian ini, ketiga sumbu ini digunakan sebagai acuan terjadinya pergerakan pada pemain gitar. Apabila sumbu Y berubah, maka suara intensitas efek gitar akan berubah sesuai dengan nilai sumbu Y saat ini. Sedangkan apabila terjadi perubahan pada sumbu Z, maka terjadi perubahan jenis suara pada efek dari suara yang satu ke suara yang lainnya.



Gambar 2.1 Sistem koordinat sensor

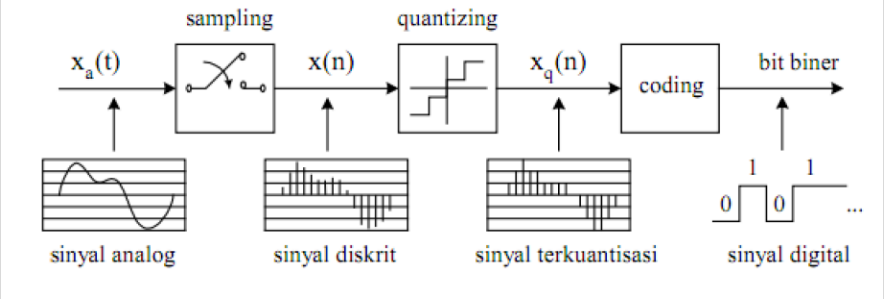
Sumber: (Developer, 2015)

2.3 Pengolahan Suara

Suara merupakan sesuatu yang dihasilkan oleh benda yang bergetar sehingga akan dapat menghasilkan suatu gelombang (Nurhayati D. , 2010). Pengolahan suara/*audio* merupakan suatu proses aritmatik yang dilakukan terhadap suatu sinyal suara sehingga diperoleh informasi yang berguna (Kurniawan, 2010).

Suara termasuk kedalam jenis sinyal analog. Sinyal analog merupakan sinyal yang berupa gelombang *electromagnetic* yang bekerja dengan mentransmisikan suara dan gambar dalam bentuk gelombang *kontinu* (Hartono, 2014). Dengan menggunakan sinyal analog, maka jangkauan transmisi data dapat mencapai jarak jauh, akan tetapi sinyal analog akan mudah diperngaruhi oleh *noise*. Oleh karena itu sinyal analog ini perlu ditransformasikan menjadi sinyal digital. Sinyal digital merupakan hasil teknologi yang dapat merubah sinyal menjadi bit-bit biner sehingga tidak mudah terpengaruh oleh derau.

Untuk dapat mentranformasi sinyal analog menjadi sinyal digital dapat menggunakan ADC. ADC (*Analog to Digital Converter*) mengubah *amplitude* sebuah gelombang kedalam waktu *interval* (*sample*) sehingga menghasilkan representasi digital dari suara. Sedangkan untuk menampilkan suara/sinyal digital kedalam suara analog (speaker) menggunakan DAC (*Digital go Analog Converter*) untuk mengkonversikannya.



Gambar 2.2 Proses konversi sinyal analog ke sinyal digital

Sumber: (Hartono, 2014)

Terdapat 3 langkah yang dilakukan dalam proses ADC seperti pada Gambar 2.2 yaitu (Nurhayati D. , 2010):

1. *Sampling*

   Proses perubahan waktu yang berjalan diubah kedalam bentuk diskrit disebut dengan sampling. Frekuensi dari waktu biasanya disebut dengan *sampling rate* atau frekuensi *sampling*.

2. Kuantisasi

   Merupakan proses perubahan bentuk yang berkelanjutan kedalam bentuk diskrit. Dalam proses ini dilakukan pembagian *range* sinyal kedalam bentuk interval angka yang disepakati. Ukuran dari interval kuantisasi disebut dengan langkah kuantisasi.

3. Pengkodean

Merupakan proses merepresentasikan isi digital kuantisasi. Ini dapat terjadi saat menggunakan *digital to analog converter* (DAC) untuk melakukan rekontruksi kembali sinyal analog yang berasal dari data digital.

2.3.1 USB OTG

USB OTG (*On The Go*) merupakan suatu spesifikasi yang memungkinkan perangkat smartphone yang memiliki *port* USB untuk bertindak sebagai *host*, yang memungkinkan perangkat USB lainnya seperti USB *flashdrive*, *mouse*, atau *keyboard* dapat terpasang. Tidak seperti sistem USB konvensional, USB OTG dapat bertindak sebagai perangkat USB normal ketika terpasang ke host lain (Wijaya, 2013).

Sebuah perangkat android kemungkinan tidak menyediakan daya yang cukup untuk mengoperasikan beberapa perangkat yang ada seperti *flashdisk* ataupun *hardisk external*. Oleh karena itu diperlukan kabel OTG yang dimodifikasi yaitu OTG Y Power Hub seperti Gambar 2.3 untuk juga dapat menerima *input* daya sehingga kebutuhan daya dapat terpenuhi.

Gambar 2.3 USB OTG Y

Sumber: (Kurniawan, 2010)

2.3.2 *Audio Interface*

Audio interface merupakan hal yang penting untuk dimiliki sebagai media utama perekaman. *Interface* atau juga disebut *soundcard external* ini bertugas menerima sinyal dari *instrument* lalu meneruskannya menuju komputer untuk diproses secara lebih dalam (audio, 2014). Audio interface ada yang menggunakan usb atau *firewire* sebagai media untuk menyalurkan suara. *Firewire* mampu mengirimkan data sinyal suara lebih cepat dibandingkan dengan usb sehingga resiko *latency* dapat dihindari. Akan tetapi dengan adanya usb 3, kini proses transfer dari *audio interface* yang menggunakan usb mampu dilakukan lebih cepat.

Pada penelitian ini, digunakan Guitar link seperti pada Gambar 2.4 sebagai *audio interface* yang akan mengalirkan sinyal suara dari gitar menuju ke smartphone. Guitar link ini menggunakan usb sebagai media untuk menyalurkan suara. *Output* usb ini nantinya akan dihubungkan dengan usb OTG sehingga guitar link dapat terhubung dengan smartphone android.

Gambar 2.4 Guitar link UCG102

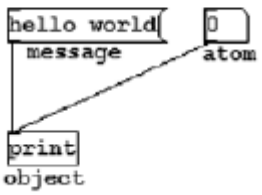
Sumber: (UCG102, 2015)

2.4 Pure Data

Pure data merupakan bahasa pemograman visual untuk suara digital atau lainnya secara *real-time* dan bersifat *open-source* yang dikembangkan oleh Miller Puckette pada tahun 1990 untuk menciptakan karya multimedia yang interaktif (Brinkman, 2012). Pure data memungkinkan musisi, seniman visual, peneliti, dan pengembang untuk dapat membuat perangkat lunak grafis, tanpa menulis baris kode. Pd digunakan untuk memproses dan menghasilkan suara, video, grafis 2D/3D, sensor antarmuka, perangkat *input*, dan MIDI.

Selain itu, Pure data juga merupakan representasi pemograman grafis. Baris kode yang menggambarkan fungsi program dan bagaimana mereka berinteraksi telah digantikan dengan objek visual yang dapat dimanipulasi secara langsung pada layar. Pengguna dapat membuat program-program baru (*patch*) dengan menempatkan fungsi (*object*) di layar. Pengguna dapat mengubah cara objek-objek ini berprilaku dengan mengirim pesan (*message*) dan dengan menghubungkan mereka bersama dengan menggambarkan garis diantara objek-objek tersebut. *Patch* akan dieksekusi secara *real time* dan apabila terjadi perubahan pada salah satu objek didalamnya, maka perubahan pada *patch* akan diberlakukan saat itu juga.

Terdapat 4 tipe dari objek dalam pure data yaitu *message, atom, object,* dan *comment* (Gruber, 2006). *Message* merespon klik dari mouse dengan mengirimkan konten mereka ke satu atau lebih tujuan. Tujuan biasanya ditunjukkan melalui *outlet* yang berada di pojok kiri dari kotak. *Atom* merespon dari pergeseran *mouse* ke atas dan ke bawah sehingga dapat mengubah isinya dan mengirim hasilnya keluar melalui *outlet*. Sedangkan *object* memiliki fungsi masing-masing tergantung apa yang ketik. Seperti contohnya *object* "print" yang ada pada Gambar 2.5 berfungsi untuk menampilkan pesan yang dikirimkan.



Gambar 2.5 Contoh *patch pure data*

Sumber: (Gruber, 2006)

Dalam melakukan pemrosesan sinyal suara dalam pure data, digunakan beberapa objek atau perintah yang dijelaskan pada Tabel 2.1 berikut.

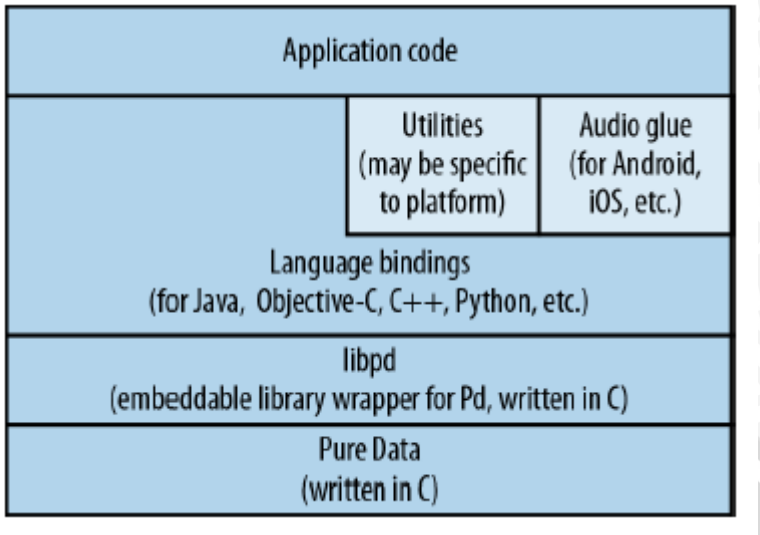
Tabel 2.1 Komponen pure data

No	Objek	Keterangan
1	adc~	Menerima inputan dari *soundcard*
2	dac~	Mengalirkan output ke *soundcard*
3	[+]	Operasi aritmatika untuk melakukan proses pertambahan terhadap sinyal
4	[-]	Operasi aritmatika untuk melakukan proses pengurangan terhadap sinyal
5	[*]	Operasi aritmatika untuk melakukan proses perkalian terhadap sinyal
6	[/]	Operasi aritmatika untuk melakukan proses pembagian terhadap sinyal
7	osc~	Menghasilkan suara dalam bentuk gelombang cosinus
8	clip~	Membuat batasan terhadap sinyal dengan nilai maksimum dan minimum
9	vcf~	Salah satu filter dalam pure data yang berfungsi untuk menghapus beberapa frekuensi yang tidak sesuai nilai dari parameter yang ditentukan

2.4.1 LibPd

Libpd adalah sebuah wrapper yang mengubah Pure Data menjadi *library* pemrosesan sinyal (Brinkman, 2012). Libpd merupakan cara baru dalam menggunakan Pure Data yang memungkinkan kompatibilitas dengan pengembangan aplikasi mobile, pengembangan game, embedding kedalam alat-alat canggih visualisasi 3d, dan banyak aplikasi lainnya (Kirn, 2012).

Dalam model aplikasi berbasis libpd seperti pada Gambar 2.6, terdapat 5 komponen utama yaitu Pure Data itu sendiri, Libpd, Java Binding untuk *library* ini, beberapa *Audio Glue* untuk mengikat Java Binding kedalam arsitektur *audio* android, dan kode aplikasi. Setiap layer komponen hanya berkomunikasi dengan lapisan setelahnya. Libpd ditulis dengan bahasa C, tapi APInya menggunakan tipe data standar.



Gambar 2.6 Layer model aplikasi berbasis LibPd

Sumber: (Brinkman, 2012)

Dalam menginisialisasi aplikasi yang berbasis libpd, harus dilakukan dengan urutan yang benar yaitu:

1. Inisialisasi komponen *audio*.
2. Membuat *dispatcher* dan mendaftarkannya dengan PdBase.
3. Tambahkan *listener* jika dibutuhkan.
4. Ambil *patch* yang ingin dijalankan.
5. Aktifkan komponen *audio*.

2.5 UML

Pemodelan adalah suatu proses merancang perangkat lunak atau aplikasi sebelum melakukan pengkodean (coding) (Darwiyanti 2003). *Unified Modeling Language* (UML) sudah menjadi standar bahasa untuk melakukan visualisasi, merancang dan mendokumentasikan sistem aplikasi dalam industri perangkat lunak. Dengan menggunakan UML, sebuah sistem dapat dimodelkan menjadi diagram-diagram yang merepresentasikan sistem tersebut. Model inilah yang kemudian menjadi *blueprint* untuk tahap implementasi perangkat lunak dan dokumentasi.

Berikut adalah kelebihan UML (Hamilton, 2006) :

1. UML adalah bahasa formal

   Tiap elemen dari UML memiliki makna tersendiri sehingga tidak akan terjadi kesalah pahaman.
2. UML komprehensif

   UML dapat digunakan untuk memodelkan seluruh aspek-aspek penting dari sebuah sistem.
3. UML memiliki skalabilitas yang tinggi

   UML dapat digunakan untuk memodelkan mulai dari sistem yang sederhana hingga sistem yang sangat besar.

4. UML memiliki banyak referensi

UML telah dikembangkan lebih dari 15 tahun dan memiliki banyak sumber atau referensi yang dapat dijadikan bahan pembelajaran.

5. UML adalah standar

UML adalah standar untuk pemodelan yang dikontrol oleh open standart group yang memiliki kontribusi aktif dari berbagai macam vendor dan akademisi dari seluruh dunia sehingga transformabilitas dan interoperabilitas UML terjamin.

2.5.1 *Use case* Diagram

Use case diagram adalah kumpulan gambaran relasi antara *use case* dengan aktor (Nugraha, 2011). Diagram ini sangat penting untuk mengorganisasi suatu sistem. *Use case* diagram digunakan untuk melihat sudut pandang sistem secara statik (Nugraha, 2011).

Use case diagram terdiri dari *actor, use case*, dan *communicate*. *Actor* adalah segala sesuatu yang berinteraksi dengan *use case*. *Use case diagram* menggambarkan deskripsi aktifitas yang dapat dilakukan oleh sistem. Sedangkan *communicate* adalah garis yang menghubungkan *actor* dan *use case*.

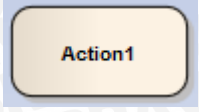
2.5.2 *Activity* Diagram

Activity Diagram adalah salah satu diagram UML yang digunakan untuk memodelkan alur dari proses pada sebuah sistem. *Activity diagram* menampilkan aksi-aksi yang saling berhubungan yang mewakili suatu proses yang terjadi dalam suatu sistem yang dimodelkan (Hamilton, 2006).

Activity diagram baik digunakan untuk memodelkan proses bisnis. Selain itu, *activity diagram* menggunakan simbol-simbol yang sama dengan *flowchart* sehingga diagram ini cocok digunakan untuk menggambarkan suatu proses kepada *audience* yang luas (Hamilton, 2006).

Tabel 2.1 merupakan tabel simbol yang digunakan dalam *Activity diagram:*

Tabel 2.2 Tabel simbol *activity* diagram

NO	Notasi	Keterangan
1		Notasi untuk memulai dan mengkahiri suatu *activity* diagram
2		*Activity* adalah proses yang dimodelkan.
3		*Action* adalah bagian dari *activity* atau langkah-langkah yang membentuk suatu *activity*

4		*Decision* adalah percabangan dalam *activity* diagram untuk mengeksekusi urutan action tertentu berdasarkan suatu kondisi
5		*Fork* digunakan untuk menunjukkan *action-action* yang dieksekusi secara bersamaan

2.5.3 *Sequence* Diagram

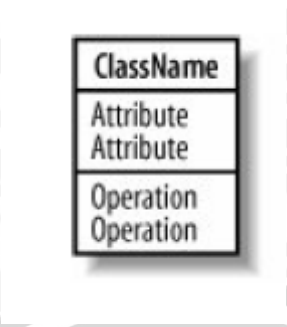
Sequence Diagram adalah diagram interaksi yang menekankan pada pengiriman waktu pesan. *Sequence* diagram menunjukkan sekumpulan objek dan pesan yang dikirim dan diterima oleh objeknya. *Sequence* diagram menggambarkan interaksi dari setiap partisipan dengan garis alir secara *vertical* dan pengurutan pesan dari atas ke bawah (Nugraha, 2011).

Terdapat beberapa notasi pada *sequence* diagram seperti *lifeline, message, guards, alternative, option* dan *loop. Lifeline* mewakili obyek yang ikut berinteraksi dalam sistem. *Lifeliness* digambarkan sebagai kotak yang memiliki nama di dalamnya dengan garis putus-putus ke bawah. *Message* digambarkan dengan garis yang memiliki anah panah di ujungnya. *Message*/nama *method* ditempatkan diatas garis. *Message* yang dikirim mewakili operasi/*method* yang diimplementasikan oleh obyek tujuan. *Guard* digunakan ketika suatu kondisi harus dipenuhi sebelum sebuah message dapat dikirim. *Guard* diwakili dengan notasi [kondisi]. Alternatif pada *sequence* diagram digunakan jika terdapat dua atau lebih alternatif *sequence* yang harus dieksekusi berdasarkan kondisi tertentu. Sama halnya dengan percabangan pada pemrograman. Sedangkan *option*, hampir sama dengan *guard* namun *option* melibatkan *sequence* yang mengandung lebih dari 2 pengiriman *message*. Yang terakhir adalah *loop*. *Loop* adalah perulangan yang terjadi pada *sequence* ketika memenuhi kondisi tertentu.

2.5.4 *Class* Diagram

Class adalah sebuah cetak biru dari suatu objek (Darwiyanti, 2003). Sebuah objek dimodelkan dengan melakukan abstraksi. Abstraksi adalah proses menghilangkan informasi yang tidak penting dari sebuah objek dan hanya menggunakan informasi yang relevan dan cukup untuk mewakili suatu objek (Hamilton, 2006). Hasil dari abstraksi suatu objek adalah *class*. Sedangkan *Class Diagram* adalah sebuah diagram yang menggambarkan *class* yang dibutuhkan oleh suatu sistem serta relasi antar *class* tersebut (Hamilton, 2006).

Sebuah *Class* dalam *Class* diagram dimodelkan dengan kotak yang berisi Nama kelas, Attribut dan Behavior. Gambar 2.7 menunjukkan gambar dari sebuah *class.*



Gambar 2.7 Contoh *class diagram*

Dalam sebuah *class* diagram, *class* yang ada didalamnya saling berhubungan. Ada beberapa macam jenis relasi atau hubungan pada *class* diagram. Gambar 2.8 menunjukkan jenis-jenis relasi pada *class* diagram.

Gambar 2.8 Jenis garis dalam *class diagram*

Dependensi adalah relasi dari *class* yang digunakan ketika suatu *class* membutuhkan *class* yang lain ketika ingin menggunakan objeknya (Hamilton, 2006). Asosiasi adalah relasi dari *class* yang mengandung referensi kepada objek dari kelas lain dalam bentuk attribute (Hamilton, 2006). Aggregasi adalah versi yang lebih kuat dari asosiasi. Relasi ini berarti bahwa suatu *class* memiliki dan dapat berbagi objek dari *class* yang lain. Komposisi adalah versi yang lebih kuat dari aggregasi. Suatu *class* dapat mengandung *class* lainnya dan tidak dapat dishare kepada bagian lain dari sistem. Komposisi berguna untuk memodelkan bagian internal yang membuat suatu *class*. Generalisasi atau *inheritance* adalah relasi terkuat untuk menunjukkan bahwa suatu *class* adalah bagian dari *class* yang lain. Dalam relasi ini, terdapat *superclass* dan *subclass* dimana *subclass* akan mewarisi atau memiliki attribut dan *method/*operasi dasar dari *superclass.*

2.6 Pengujian Perangkat Lunak

Pengujian perangkat lunak merupakan suatu tahapan penting dalam pembangunan perangkat lunak. Pengujian dilakukan dengan mengevaluasi konfigurasi perangkat lunak yang terdiri dari spesifikasi kebutuhan, deskripsi perancangan dan program yang dihasilkan. Hasil evaluasi kemudian dibandingkan dengan hasil uji yang diharapkan. Jika ditemukan kesalahan, maka perbaikan perangkat lunak harus dilakukan untuk kemudian diuji kembali.

Pengujian perangkat lunak adalah suatu proses menjalankan dan mengevaluasi perangkat lunak untuk menguji apakahsudah memenuhi

persyaratan atau belum, atau untuk menentukan perbedaan antara hasil yang diharapkan dengan hasil sebenarnya (Nurhayati A. , 2010).

2.6.1 Teknik Pengujian

Hanya terdapat satu jenis teknik pengujian yang digunakan dalam penelitian ini yaitu *black box testing*.

2.6.1.1 *Black Box Testing*

Pengujian *black box* digunakan untuk menguji fungsi-fungsi khusus yang ada dari perangkat lunak. Pada teknik ini hanya menekankan kepada keluaran yang dihasilkan dari data atau kondisi masukan yang diberikan tanpa melihat bagaimana proses untuk mendapatkan keluaran tersebut. Beberapa jenis kesalahan yang dapat diidentifikasi (Nurhayati A. , 2010):

1. Fungsi tidak benar atau hilang.
2. Kesalahan antar muka.
3. Kesalahan pada struktur data.
4. Kesalahan inisialisasi dan akhir program
5. Kesalahan performasi.

2.6.2 Strategi Pengujian

Digunakan untuk mngintegrasikan metode-metode perancangan kasus pengujian perangkat lunak menjadi satu langkah-langkah terencana dengan tujuan mendapatkan perangkat lunak yang sukses. Setiap strategi pengujian harus meliputi perencanaan pengujian, perancangan kasus-kasus uji, eksekusi pengujian, pengumpulan data, serta evaluasi.

2.6.2.1 Pengujian Validasi

Pengujian ini dimulai jika tahap integrasi tidak ditemukan kesalahan. Suatu validasi dikatakan sukses jika perangkat lunak berfungsi pada suatu cara yang diharapkan oleh pemakai (Nurhayati A. , 2010).

2.6.2.2 Pengujian *Usability*

Usability adalah sejauh mana suatu produk digunakan oleh pengguna tertentu untuk mencapat target yang ditetapkan dengan efektivitas, efesiensi dan mencapai kepuasan penggunaan dalam konteks tertentu. Berdasarkan definisi tersebut *usability* diukur berdasarkan komponen (Rahadi, 2014):

1. *Learnability*

   Mendefinisikan seberapa cepat pengguna mahir dalam menggunakan sistem serta kemudahan dalam penggunaan menjalankan suatu fungsi serta apa yang pengguna inginkan.
2. *Efficiency*

   Mendefinisikan sebagai sumber daya yang dikeluarkan guna mencapai ketepatan dan kelengkapan tujuan.

3. *Memorability*
Mendefinisikan bagaimana kemampuan pengguna mempertahankan pengetahuannya setelah jangka waktu tertentu.
4. *Errors*
Mendefinisikan berapa banyak kesalahan-kesalahan yang dibuat pengguna, kesalahan yang dibuat pengguna mencangkup ketidaksesuaian apa yang pengguna pikirkan dengan apa yang sebenarnya disajikan oleh sistem.
5. *Satisfaction*
Mendefinisikan kebebasan dari ketidaknyamanan, dan sikap positif terhadap penggunaan produk atau ukuran subjektif sebagaimana pengguna merasa tentang penggunaan sistem.

Proses analisis terhadap hasil pengujian *usability* dilakukan dengan menggunakan skala Linkert. Skala Linkert adalah suatu skala psikometrik yang biasanya digunakan dalam kuisioner, dan merupakan skala yang paling banyak digunakan dalam riset berupa survei (Fitrianto, 2014).

Dalam penelitian ini digunakan pengujian *usability* untuk mengukur tingkat kenyamanan user terhadap sistem. Pengujian *usability* dalam penelitian ini menggunakan metode kuisioner dengan membagikan beberapa pertanyaan kepada sejumlah responden. Kemudian hasil jawaban responden akan dianalisis menggunakan skala Linkert.

Untuk mendapatkan total skor dari hasil kuisioner yang kemudian akan digunakan untuk menghitung indeks atau hasil interpretasi dari pengujian *usability*, menggunakan rumus (Fitrianto, 2014):

$$\text{Total Skor} = S_{STS}\times 1 + S_{TS}\times 2 + S_{N}\times 3 + S_{S}\times 4 + S_{SS}\times 5 \quad (2.1)$$

Kemudian dari perhitungan Total Skor, maka dapat ditentukan nilai indeksnya atau hasil interpretasi yang merupakan tingkat *usability* sistem yaitu dengan menggunakan rumus (Fitrianto, 2014):

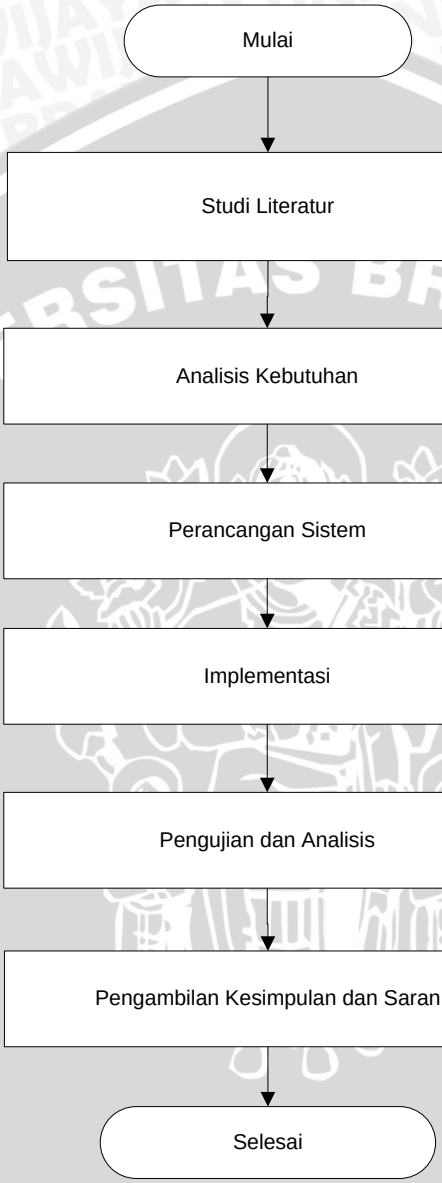
$$\text{Index}(\%) = (\text{Total Skor}/Y)100 \quad (2.2)$$

Keterangan:

STS	=	Sangat Tidak Setuju	N	=	Netral	SS	=	Sangat Setuju
TS	=	Tidak Setuju	S	=	Setuju	Y	=	Skor linkert tertinggi × jumlah responden

BAB 3 METODOLOGI PENELITIAN

Pada bab ini akan dijelaskan tentang langkah-langkah yang digunakan dalam penelitian. Metodologi penelitian ini terdiri dari enam tahap. Runtutan pengerjaan penelitian dapat dilihat pada diagram alir berikut.



Gambar 3.1 Diagram alir penelitian

3.1 Studi Literatur

Studi literature berisi dasar teori yang digunakan sebagai sumber acuan untuk penulisan skripsi. Studi literatur dilakukan dengan membaca buku, jurnal, paper dan artikel-artikel di internet. Teori dan pustaka yang berkaitan dengan tugas akhir ini meliputi:

1. Efek Gitar
2. Android
   - Sensor Akselerometer
   - Sistem Koordinat Sensor
3. Pengolahan Suara
   - USB OTG
   - *Audio Interface*
4. UML
   - *Use case* Diagram
   - *Activity* Diagram
   - *Sequence* Diagram
   - *Class* Diagram
5. Pengujian Perangkat Lunak
   - Teknik Pengujian
   - Strategi Pengujian

3.2 Analisis Kebutuhan

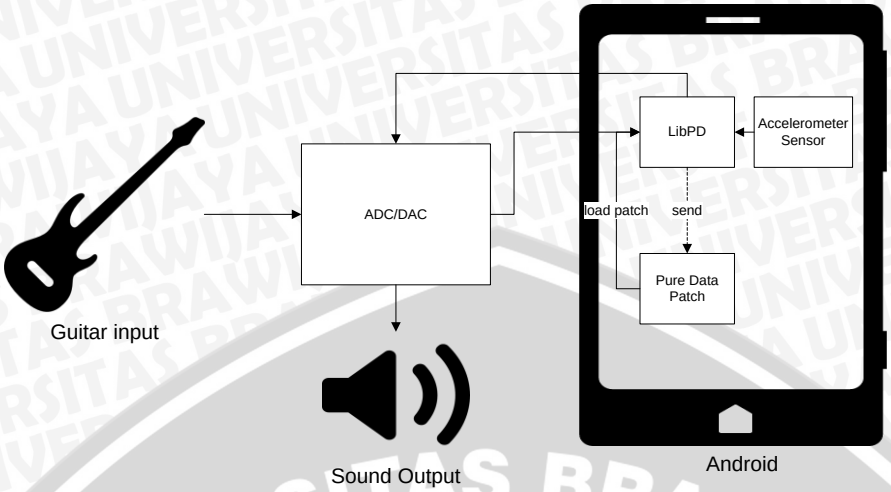
Kegiatan analisis kebutuhan dalam penelitian ini meliputi analisis spesifikasi perangkat lunak. Metode analisis dalam penelitian ini menggunakan bahasa pemodelan UML (*Unified Modeling Language*). *Use case Diagram* digunakan untuk mendeskripsikan kebutuhan-kebutuhan dan fungsionalitas sistem dari perspektif user. Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan (*requirements*) sistem yang kemudian akan dimodelkan dalam diagram *use case*. Kebutuhan fungsional yang nantinya akan disediakan oleh aplikasi ini antara lain adalah:

1. Aplikasi ini dapat mengubah suara gitar menjadi efek suara wah-wah, *tremolo*, dan distorsi.
2. Aplikasi ini dapat mengganti efek suara yang diinginkan dengan kendali pergerakan dari pemain gitar.
3. Aplikasi ini dapat mengubah intensitas suara setiap efek gitar berdasarkan pergerakan pemain gitar.

3.3 Perancangan Sistem

Perancangan dilakukan setelah tahap analisis kebutuhan dilakukan. Pada tahap perancangan, dilakukan identifikasi terhadap *class-class* yang dibutuhkan, dan kemudian dimodelkan dalam bentuk *class* diagram. Tahap selanjutnya hubungan dan interaksi antar objek diidentifikasi dan kemudian dimodelkan menggunakan *sequence* diagram yang menggambarkan urutan waktu dari interaksi antar objek. Kemudian tahap berikutnya merupakan perancangan antar muka pengguna.

Arsitektur sistem aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer pada smartphone android dijelaskan pada Gambar 3.2 berikut.



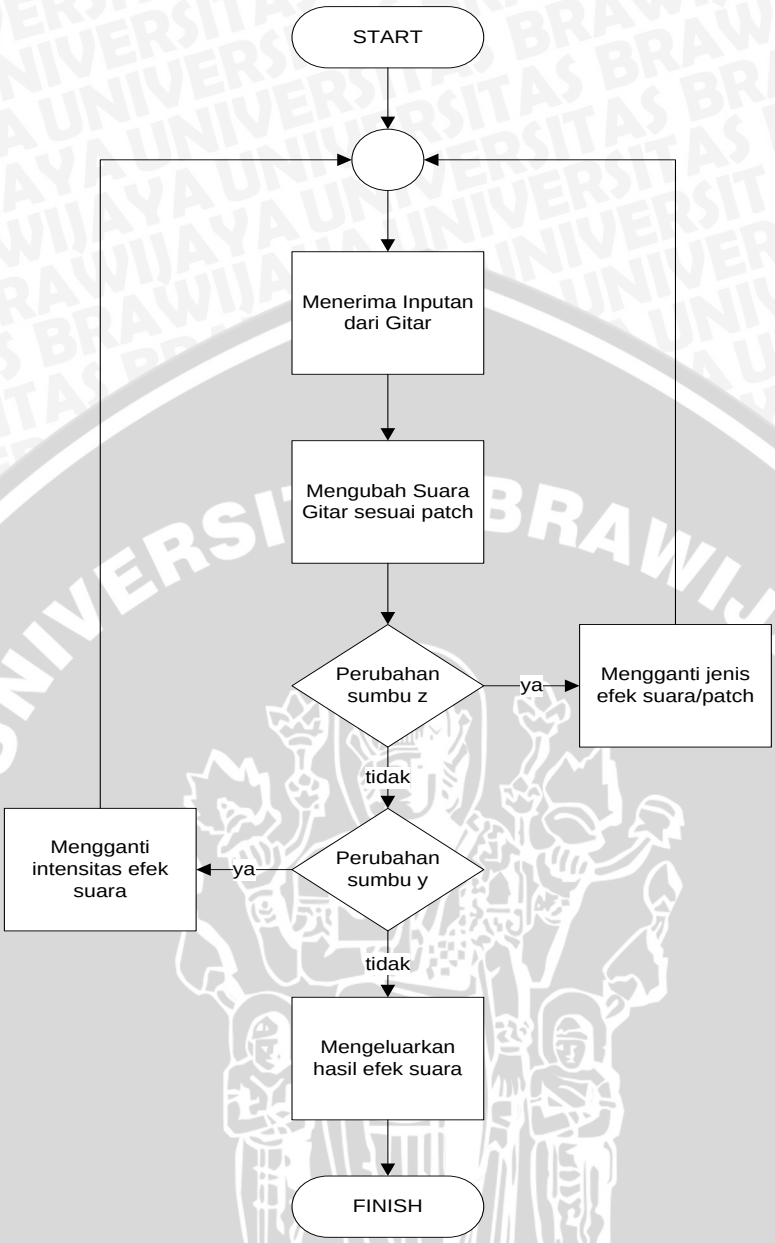
Gambar 3.2 Arsitektur sistem

Gambar 3.2 mengilustrasikan sebuah siklus sistem. Pertama-tama suara *input* dari gitar yang berupa sinyal analog akan dikonversikan oleh ADC (*Analog to Digital Converter*) menjadi sinyal digital agar dapat diproses oleh smartphone. Kemudian masukan tadi akan diproses oleh Libpd yang kemudian akan dimodifikasi sesuai dengan *patch* pure data yang digunakan. Apabila terjadi perubahan nilai pada sensor akselerometer, maka nilai dari sensor akselerometer tersebut akan digunakan oleh Libpd untuk mengirimkan pesan ke *patch* dan mengubah beberapa nilai dari *patch* tersebut. Kemudian hasil modifikasi masukan gitar yang sudah dimodifikasi berupa sinyal digital tadi akan dikonversikan lagi menjadi sinyal analog oleh DAC dan hasilnya akan dikeluarkan melalui *speaker*.

3.4 Implementasi

Implementasi dilakukan dengan mengacu pada perancangan yang telah dilakukan pada tahap sebelumnya. Implementasi sistem akan diterapkan menggunakan bahasa pemrograman java. Pembuatan aplikasi dilakukan dengan menggunakan IDE (Integrated Development Environment) Eclipse serta *library* Libpd untuk mempermudah pengolahan pure data. Sedangkan IDE PD-extended digunakan untuk membuat *patch* dari pure data yang akan digunakan.

Gambaran umum dari cara kerja aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer pada smartphone android dapat dijelaskan pada Gambar 3.3 berikut.



Gambar 3.3 Diagram alir aplikasi

Program dimulai dari penerimaan masukan dari gitar elektrik. Kemudian masukan tersebut dimodifikasi sesuai dengan *patch* dari pure data. Apabila terjadi perubahan pada nilai sumbu z dari sensor akselerometer, maka *patch* pure data diganti menjadi *patch* yang lainnya sehingga menghasilkan jenis efek suara yang berbeda dari yang sebelumnya. Dan apabila terjadi perubahan pada nilai sumbu y dari sensor akselerometer, maka libpd akan mengirim pesan kepada *patch* pure data dan mengubah beberapa parameter pada *patch* tersebut sehingga terjadi perubahan intensitas pada efek suara tersebut. Kemudian hasil dari modifikasi *input* gitar tersebut akan dikeluarkan melalui *sound output*.

3.5 Pengujian dan Analisis

Pengujian sistem dilakukan agar dapat menunjukkan bahwa perangkat lunak mampu bekerja sesuai dengan spesifikasi dan kebutuhan penggunanya dan mengetahui kinerja dan performa aplikasi. Strategi pengujian yang digunakan yaitu pengujian validasi menggunakan teknik *black box testing* dan pengujian *usability* menggunakan metode kuisioner.

Langkah selanjutnya setelah melakukan proses pengujian akan dilakukan analisa terhadap hasil pengujian sehingga didapatkan kesimpulan dari pengembangan perangkat lunak yang telah dilakukan.

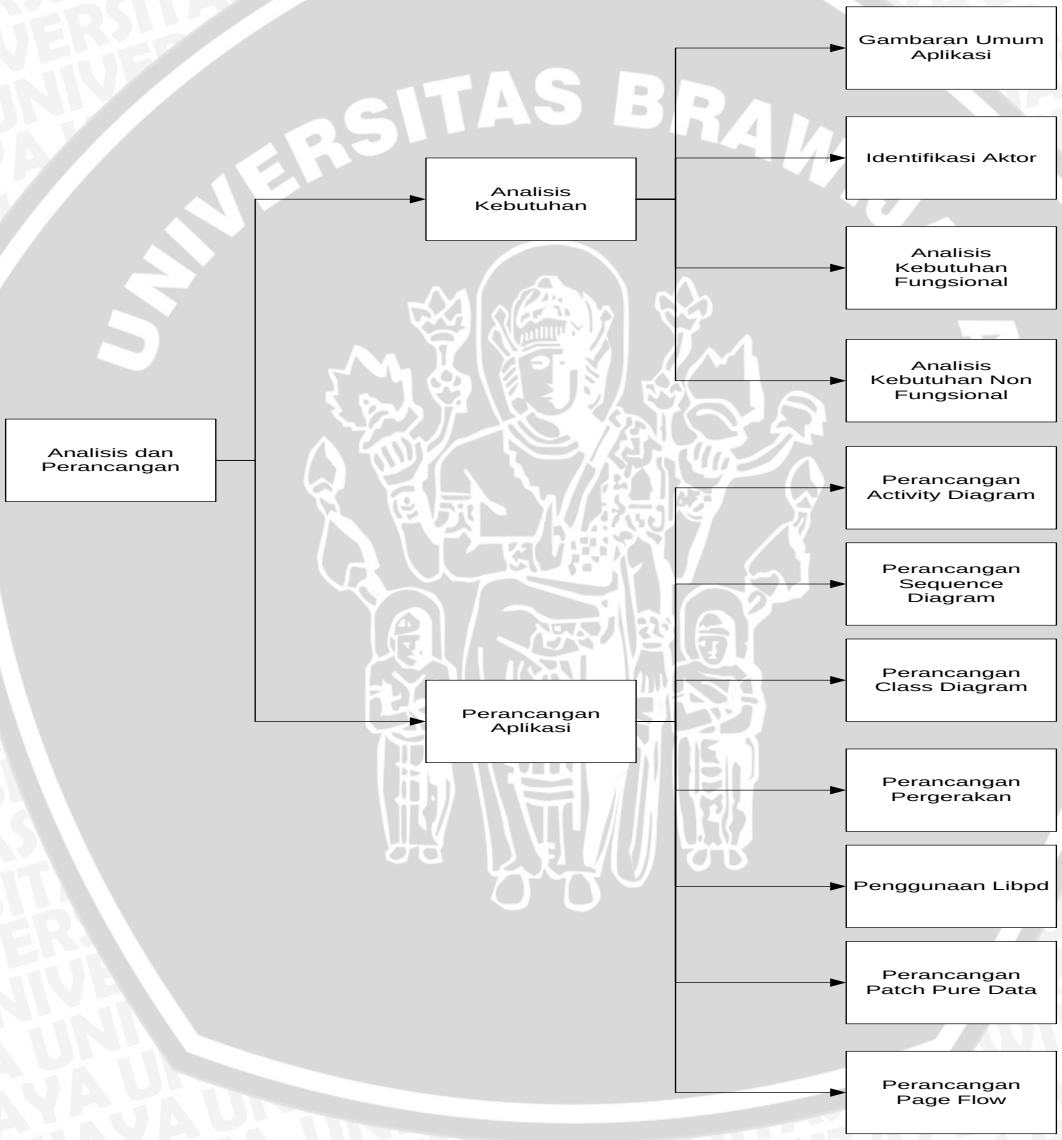
3.6 Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem aplikasi telah selesai dilakukan dan didasarkan pada kesesuaian antara teori dan praktik. Kesimpulan diambil untuk menjawab rumusan masalah yang telah ditetapkan sebelumnya. 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 4 ANALISIS DAN PERANCANGAN

Bab ini membahas tentang perancangan aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer pada smartphone android. Tahap perancangan dibagi menjadi dua tahap. yaitu analisis kebutuhan dan perancangan aplikasi.

Gambar 4.1 merupakan langkah-langkah analisis dan perancangan aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer pada smartphone android.



Gambar 4.1 Diagram pohon analisis dan perancangan

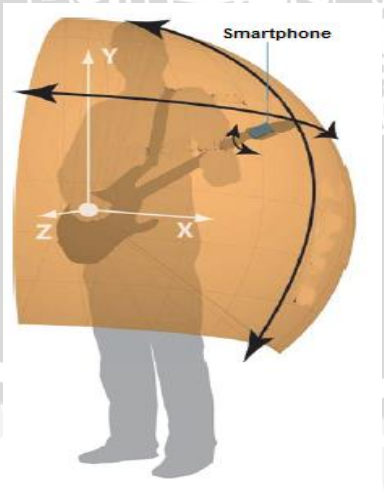
4.1 Analisis Kebutuhan

Tahap analisis kebutuhan ini akan melakukan identifikasi semua kebutuhan aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer pada smartphone android. Terdapat beberapa tahap dalam analisa kebutuhan ini antara lain gambaran aplikasi, identifikasi aktor, daftar kebutuhan fungsional dan non-fungsional dengan menggunakan *use case* diagram. Analisa kebutuhan ini bertujuan untuk memberikan analisa secara jelas tentang apa-apa saja kebutuhan yang disediakan oleh sistem demi memenuhi kebutuhan pengguna.

4.1.1 Gambaran Umum Aplikasi

Aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer pada smartphone android ini merupakan aplikasi yang bertujuan untuk membantu para pemain gitar agar dapat mengubah suara gitar elektriknya menjadi lebih variatif. Berbeda dengan efek *stompbox* yang harus diinjak atau diputar, pada aplikasi ini menggunakan pergerakan. Selain itu aplikasi ini mempunyai fitur yang dapat mengubah intensitas dari suara gitar sesuai pergerakan dari pemain gitar. Sensor akseleromater pada smartphone digunakan untuk berinteraksi dengan aplikasi.

Dalam aplikasi ini terdapat tiga macam jenis efek suara gitar yang dapat digunakan oleh pengguna, yaitu efek suara wah-wah, efek suara *tremolo*, dan efek suara distorsi. Untuk mengganti dari satu efek suara ke efek suara yang lainnya pengguna hanya perlu merotasi badan gitar ke atas. Hal ini dapat mentrigger sensor akselerometer pada smartphone yang diletakkan pada bagian *headstock* gitar seperti pada Gambar 4.2 sehingga apabila terjadi perubahan nilai dari akselerometer, maka secara otomatis jenis efek suara akan berganti menjadi jenis yang lainnya. Sedangkan untuk mengubah intensitas suara pada setiap jenis efek, pengguna hanya perlu menggerakkan bagian *headstock* gitar kearah atas.



Gambar 4.2 Visualisasi gambaran umum aplikasi

Sumber: (Willet, 2008)

4.1.2 Identifikasi Aktor

Pada tahap ini dilakukan proses identifikasi terhadap aktor-aktor yang berperan atau berinteraksi dengan aplikasi ini. Pada Tabel 4.1 memperlihatkan aktor-aktor yang terlibat berserta penjelasannya masing-masing.

Tabel 4.1 Tabel aktor

Aktor	Deskripsi
Pengguna	Pengguna merupakan pengguna dari aplikasi yang bertujuan untuk mengubah suara gitarnya menjadi lebih variatif dengan intensitas tertentu sesuai pergerakan dari pengguna

4.1.3 Analisis Kebutuhan Fungsional

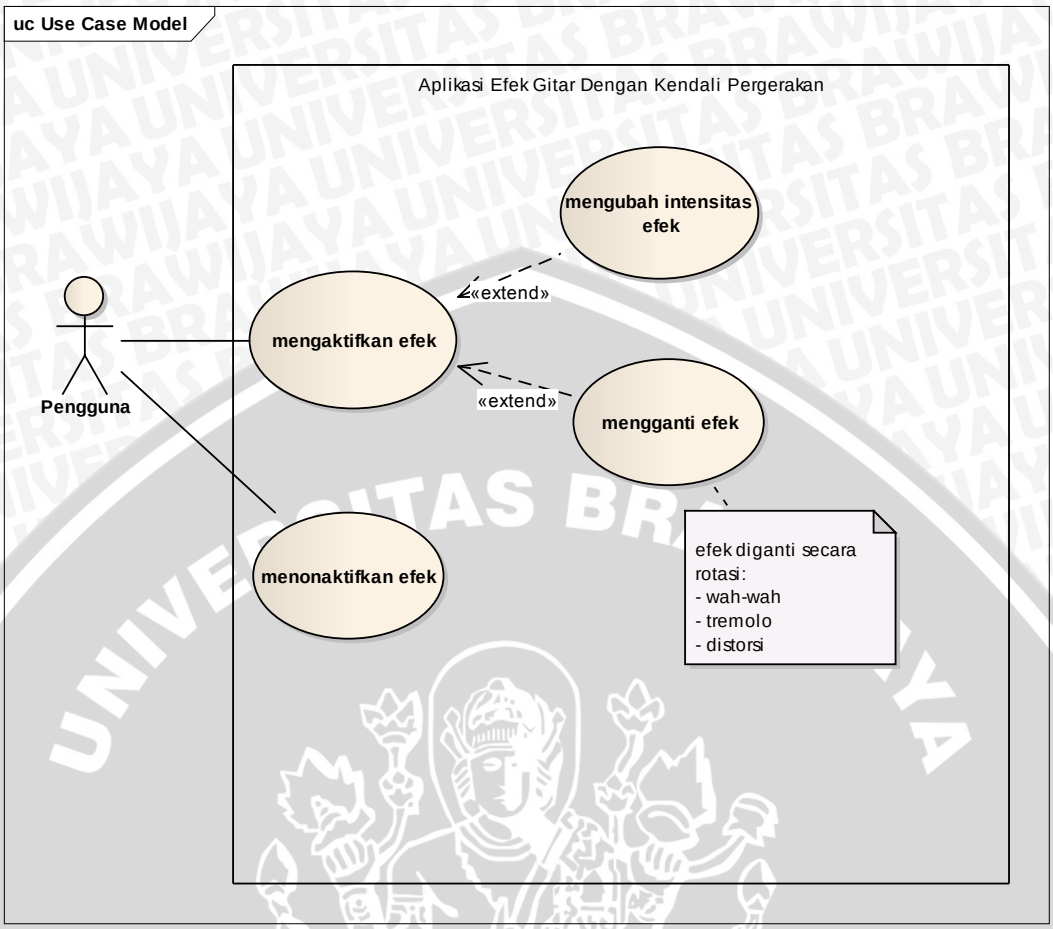
Tahap ini merupakan identifikasi kebutuhan fungsional dari aplikasi. Kebutuhan-kebutuhan dari sistem harus dirancang dengan baik agar aplikasi dapat memenuhi kebutuhan pengguna. Pada tabel berikut memuat fitur utama yang dimiliki oleh aplikasi.

Tabel 4.2 Daftar kebutuhan fungsional

Id	Kebutuhan	*Use case*
UC-01	Aplikasi harus mampu mengubah suara gitar menjadi suara wah-wah, *tremolo* dan distorsi	Mengaktifkan Efek
UC-02	Aplikasi harus mampu mengubah intensitas pada setiap jenis efek suara gitar	Mengubah Intensitas Efek
UC-03	Aplikasi harus mampu mengganti efek suara gitar dari jenis yang satu menjadi jenis lainnya	Mengganti Efek
UC-04	Aplikasi harus mampu menonaktifkan efek suara gitar menjadi suara gitar standar	Menonaktifkan Efek

4.1.3.1 Diagram *Use case*

Gambar 4.3 merupakan *use case* diagram dari sistem yang ada di aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer pada smartphone android.



Gambar 4.3 *Use case* Diagram

4.1.3.2 Skenario *Use case*

Pada skenario *use case* ini terdapat penjelasan yang lebih detail dari tiap-tiap *use case* pada Gambar 4.3. Skenario *use case* menjabarkan tentang nama *use case*, aktor yang terlibat, tujuan, deskripsi, langkah-langkah *use case*, kondisi awal yang harus dipenuhi (*pre-condition*) dan kondisi akhir yang diharapkan (*post condition*).

4.1.3.2.1 Skenario 1 Mengaktifkan Efek

Use case ini bertujuan untuk mengaktifkan efek suara pada gitar.

Prekondisi: Untuk melakukan *use case* ini, efek harus dalam keadaan non-aktif.

Postkondisi: Efek suara berhasil diaktifkan.

Alur Utama:

1. Pengguna menekan tombol on/off.
2. Sistem merespon dengan mengaktifkan efek suara wah-wah.

4.1.3.2.2 Skenario 2 Mengubah Intensitas Efek

Use case ini bertujuan untuk mengubah intensitas efek suara pada tiap-tiap jenis efek suara.

Prekondisi: Efek suara dalam keadaan aktif.

Postkondisi: Terjadi perubahan intensitas pada efek suara.

Alur Utama:

1. Pengguna menggerakkan *headstock* gitarnya kearah atas sehingga memicu sensor akselerometer.
2. Sistem mendeteksi perubahan nilai dari akselerometer dan mengirimkan nilai tersebut ke *patch* pure data.

4.1.3.2.3 Skenario 3 Mengganti Efek

Use case ini bertujuan untuk mengganti jenis efek suara gitar yang telah diaktifkan.

Prekondisi: Efek suara dalam keadaan aktif.

Postkondisi: Jenis efek suara berbeda dengan efek suara sebelumnya.

Alur Utama:

1. Pengguna merotasi badan gitarnya ke atas sehingga memicu sensor akselerometer.
2. Sistem mendeteksi perubahan nilai sensor akselerometer dan memanggil *patch* pure data setelahnya.

4.1.3.2.4 Skenario 4 Menonaktifkan Efek

Use case ini bertujuan untuk menonaktifkan efek suara pada gitar

Prekondisi: efek dalam keadaan aktif

Postkondisi: efek suara gitar dinonaktifkan

Alur Utama:

1. Pengguna menekan tombol on/off.
2. Sistem memanggil *patch* pure data gitar standar.

4.1.4 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional merupakan kebutuhan aplikasi yang tidak diminta oleh pengguna namun dibutuhkan agar aplikasi dapat berjalan dengan baik dalam memenuhi kebutuhan pengguna.

Analisis kebutuhan non fungsional dari aplikasi ini dijelaskan pada Tabel 4.3 berikut ini.

Tabel 4.3 Spesifikasi kebutuhan non fungsional

Parameter	Deskripsi Kebutuhan
Usability	Desain antarmuka dirancang agar mudah untuk dipelajari dan digunakan oleh pengguna serta mengetahui tingkat *latency* yang terdapat dalam aplikasi

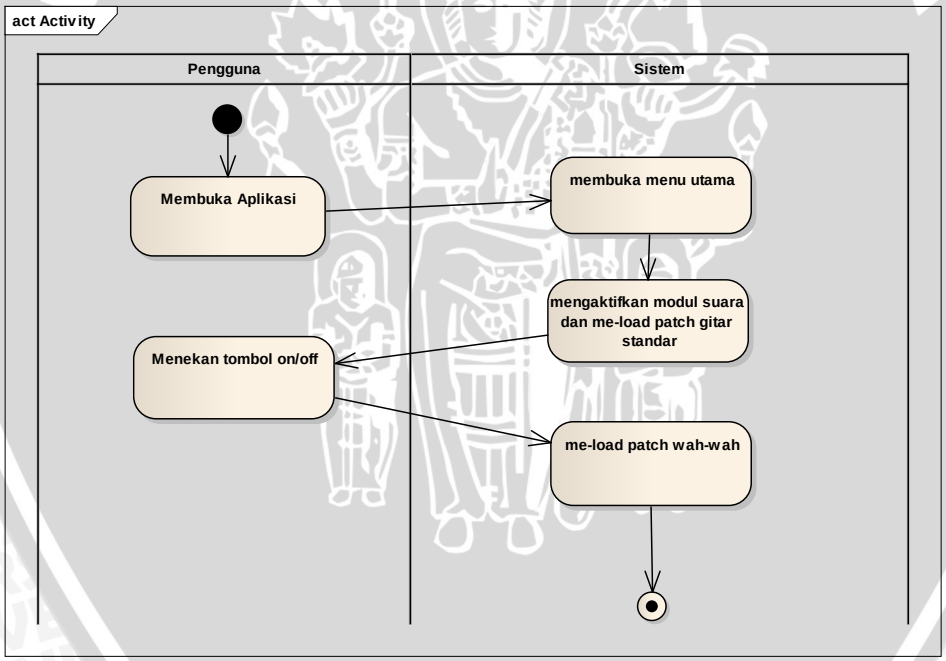
4.2 Perancangan Aplikasi

Pada perancangan aplikasi terdiri dari beberapa tahapan, yaitu perancangan *activity* diagram, perancangan *sequence* diagram, perancangan *class* diagram, perancangan pergerakan, perancangan *patch* pure data dan perancangan antar muka aplikasi.

4.2.1 Perancangan *Activity* Diagram

Perancangan *Activity* Diagram merupakan representasi grafis berupa diagram dari interaksi antara pengguna dan sistem berdasarkan pada *use case*.

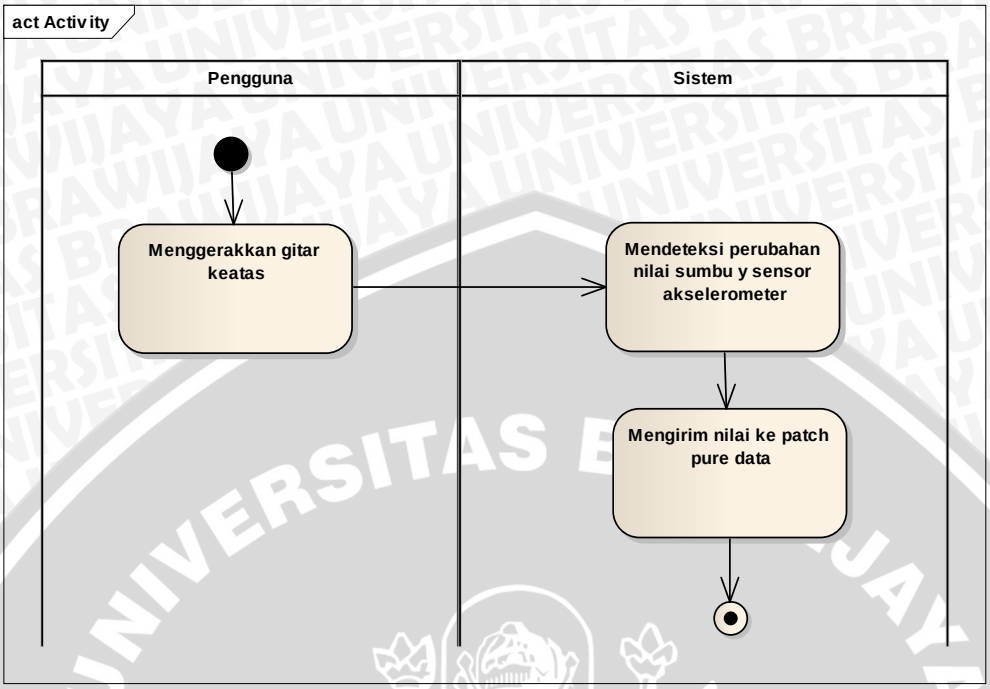
4.2.1.1 *Activity* Diagram Mengaktifkan Efek



Gambar 4.4 *Activity* diagram mengaktifkan efek

Pada Gambar 4.4 menampilkan interaksi antara pengguna dengan sistem pada pada saat pengguna aplikasi ingin menggaktifkan penggunaan dari aplikasi. Pertama-tama pengguna membuka aplikasi kemudian menekan tombol on/off pada menu utama yang telah ditampilkan oleh sistem. Setelah itu sistem akan memanggil *patch* wah-wah yang merupakan *patch default* saat pertama kali mengaktifkan aplikasi. Sinyal suara gitar akan menjadi masukan bagi *patch* yang kemudian akan dimodifikasi dan hasil modifikasi akan dikeluarkan lagi melalui *sound output*.

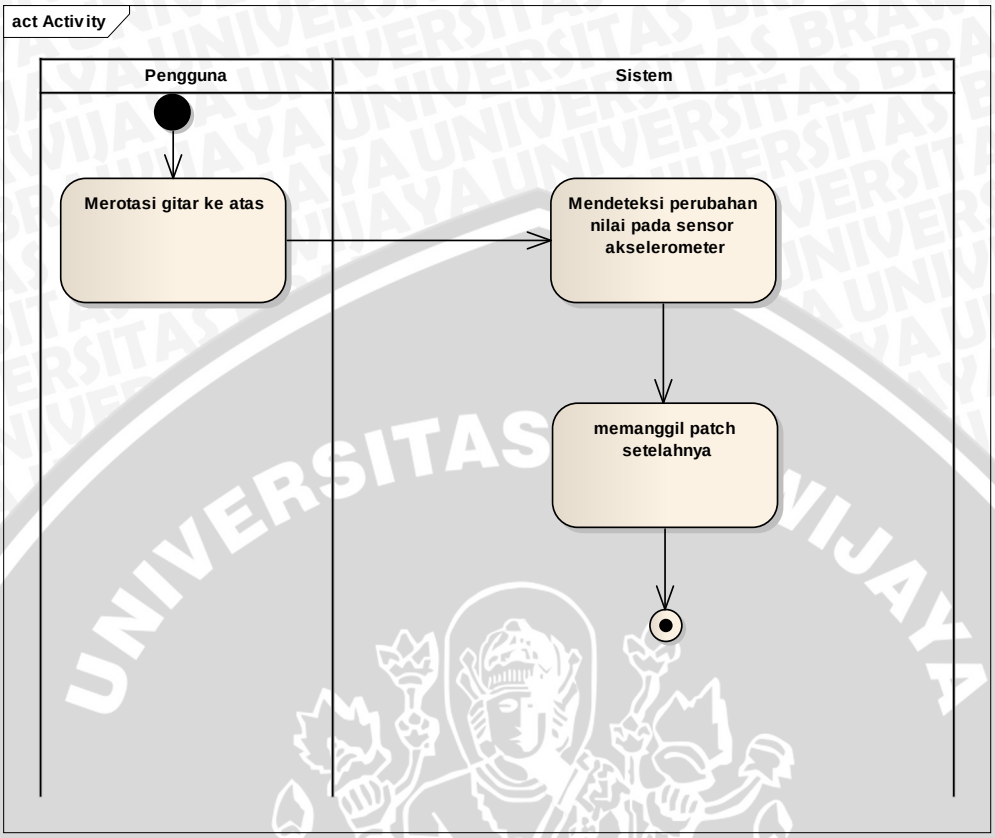
4.2.1.2 *Activity* Diagram Mengubah Intensitas Efek



Gambar 4.5 *Activity* diagram mengubah intensitas efek

Pada Gambar 4.5 menampilkan interaksi antara pengguna dengan sistem pada saat pengguna aplikasi ingin mengganti intensitas pada salah satu jenis efek suara. Pengguna menggerakan bagian *headstock* gitar yang menjadi tempat diletakkannya smartphone kearah atas sehingga memicu perubahan sumbu y pada sensor akselerometer. Nilai dari sumbu y ini yang nantinya dikirimkan ke *patch* pure data sehingga dapat mengubah nilai dari salah satu fungsi yang terdapat pada *patch* dan menyebabkan perubahan pada efek suara.

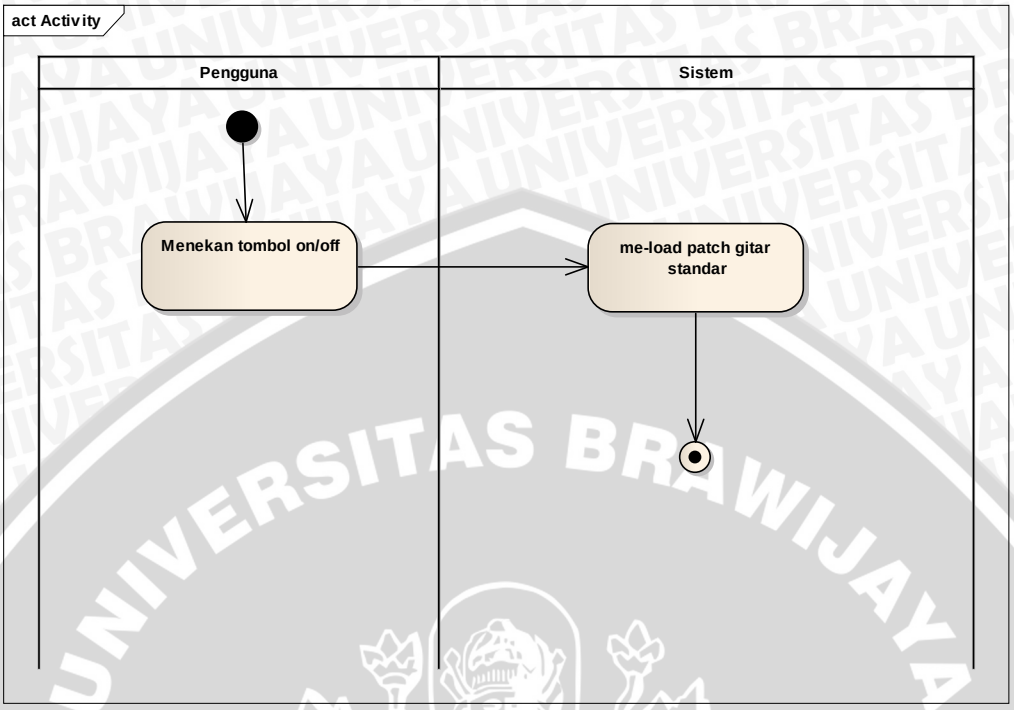
4.2.1.3 *Activity* Diagram Mengganti Efek



Gambar 4.6 *Activity* diagram mengganti efek

Pada Gambar 4.6 menampilkan interaksi antara pengguna dengan sistem pada saat pengguna aplikasi ingin mengganti efek suara yang ingin digunakan. Pertama-tama pengguna merotasi gitarnya ke atas sehingga nantinya akan memicu sensor akselerometer yang terdapat pada smartphone. Kemudian sistem akan memanggil patch pure data setelahnya. Urutan efek gitar yang digunakan ialah wah-wah, *tremolo* dan distorsi. Jadi apabila efek yang aktif sebelumnya ialah wah-wah, maka efek akan berganti menjadi *tremolo*, demikian seterusnya hingga efek kembali menjadi wah-wah.

4.2.1.4 *Activity* Diagram Menonaktifkan Efek Suara



Gambar 4.7 *Activity* diagram menonaktifkan efek suara

Pada Gambar 4.7 menampilkan interaksi antara pengguna dengan sistem pada saat pengguna menonaktifkan penggunaan efek suara dari aplikasi. Pertama-tama pengguna menekan tombol on/off saat efek suara sedang aktif. Kemudian sistem akan mengganti *patch* suara gitar yang sedang aktif dengan *patch* suara gitar standar sehingga modifikasi suara dihentikan.

4.2.2 Penggunaan *Library* LibPd

Dalam merancang aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer ini dibutuhkan *library* Libpd untuk menangani proses dari hasil analisis kebutuhan sistem. Dalam aplikasi ini *library* LibPd berperan sebagai modul pemrosesan sinyal digital secara *real time* sekaligus melakukan pertukaran informasi secara langsung antara pure data dengan *client* aplikasi. Dalam mengimplementasi aplikasi ini dibutuhkan beberapa kelas dari *library* libPd yang dapat dilihat pada Tabel 4.4.

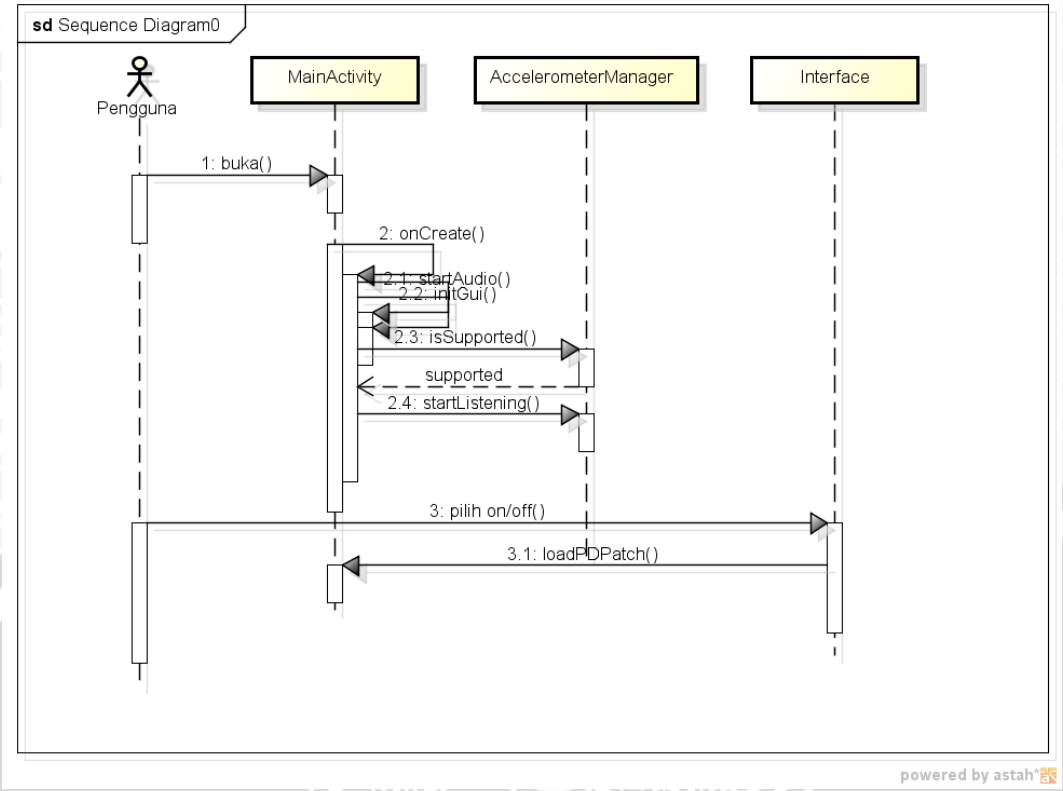
Tabel 4.4 Penggunaan *library* LibPd

Nama Kelas	Fitur Umum	Method yang digunakan
PdBase	Merupakan kelas utama dari *library* yang bertanggung jawab mengenai fitur-fitur utama dari aplikasi.	• `openPatch()`: berfungsi untuk membuka dan menyimpan *patch* dari file. • `sendFloat()`: berfungsi mengirimkan float ke object pada *patch* pure data. • `Release()`: berfungsi membersihkan segala kondisi dari pure data termasuk *patch* yang telah dibuka. • `startAudio()`: berfungsi untuk menjalankan/mengaktifkan modul suara.

4.2.3 Perancangan *Sequence* Diagram

Sequence Diagram menggambarkan perilaku objek pada *use case* dengan cara mendeskripsikan waktu hidup objek dan pesan yang dikirim dan diterima antar objek.

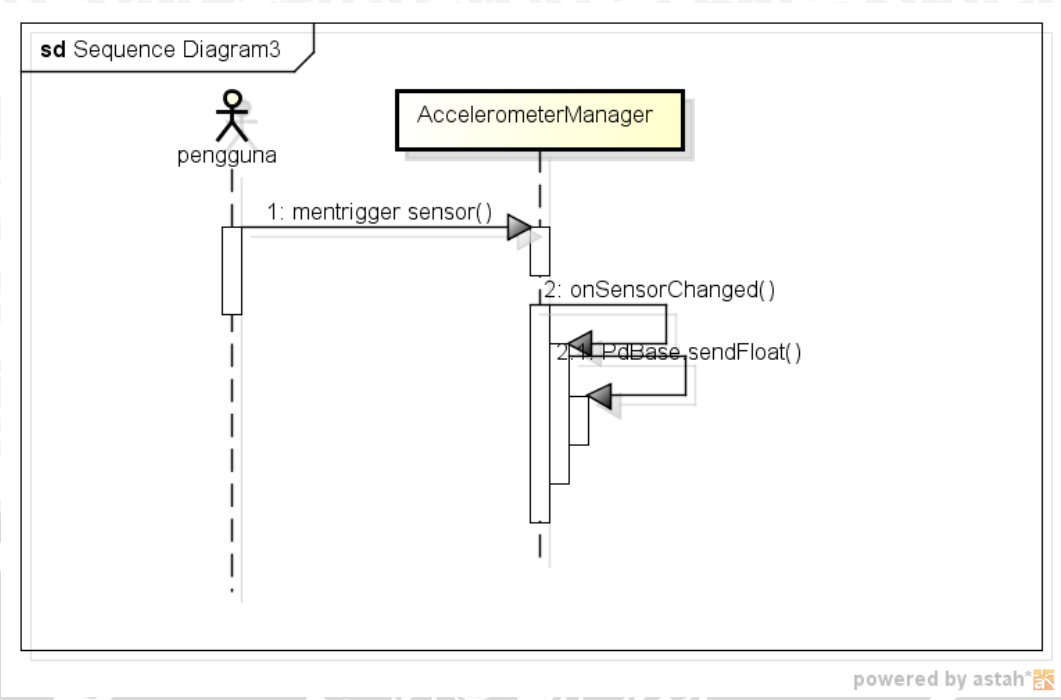
4.2.3.1 Sequence Diagram Mengaktifkan Efek



Gambar 4.8 *Sequence* diagram mengaktifkan efek

Gambar 4.8 merupakan *sequence* diagram yang digunakan pengguna saat pengguna ingin menggaktifkan penggunaan dari aplikasi. Setelah pengguna membuka aplikasi, maka akan dijalankan fungsi onCreate() untuk menginisialisasi fitur-fitur dari aplikasi dan mengaktifkan mode *listening* pada sensor akselerometer. Kemudian pengguna menekan tombol on/off untuk mengambil *patch* pure data wah-wah yang merupakan *patch default* saat aplikasi pertama kali diaktifkan.

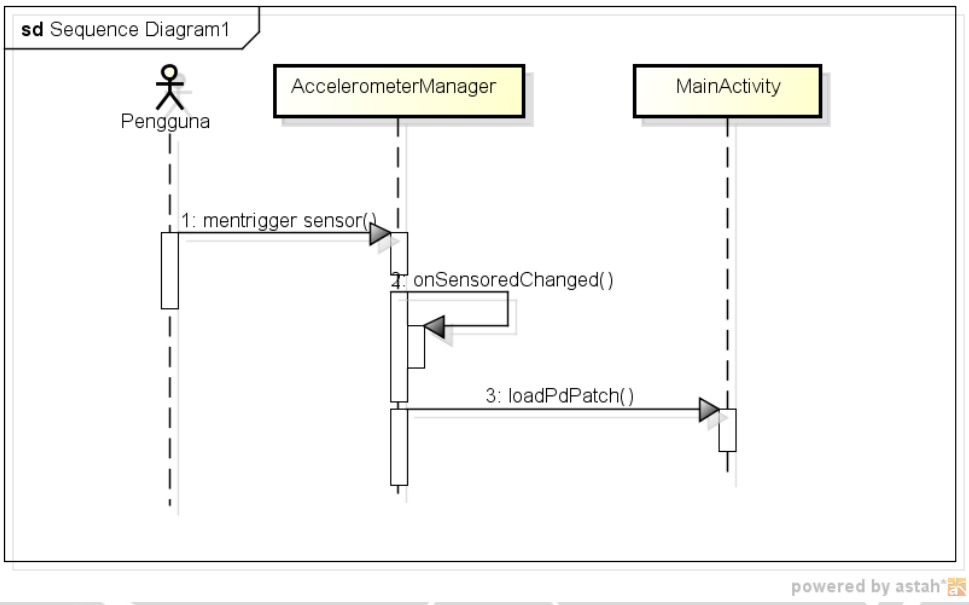
4.2.3.2 Sequence Diagram Mengubah Intensitas Efek



Gambar 4.9 *Sequence* diagram mengubah intensitas efek

Gambar 4.9 merupakan *sequence* diagram yang digunakan pengguna saat mengubah intensitas efek suara. Pengguna menggerakkan gitarnya kearah atas untuk memicu perubahan sensor akselerometer. Jika terjadi perubahan pada sensor, maka akan dipanggil fungsi onSensorChanged() dan karena pergerakan gitar mengarah keatas maka sumbu y pada sensor mengalami perubahan nilai. Kemudian digunakan fungsi PdBase.sendFloat untuk mengirimkan perubahan nilai y tadi ke *patch* pure data.

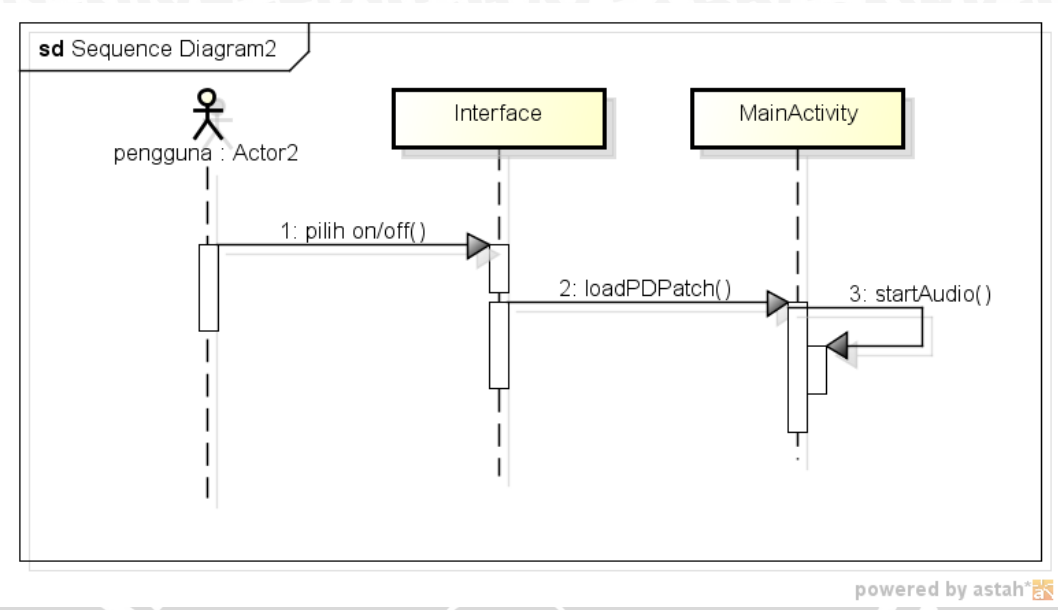
4.2.3.3 Sequence Diagram Mengganti Efek



Gambar 4.10 *Sequence* diagram mengganti efek

Gambar 4.10 merupakan *sequence* diagram yang digunakan pengguna saat pengguna ingin mengganti jenis efek suara yang digunakan. Sesudah pengguna menekan tombol on/off dan *patch* wah-wah dijalankan, pengguna dapat merotasi badan gitarnya kearah atas untuk memicu sensor akselerometer pada smartphone. Kemudian fungsi onSensorChanged() dipanggil karena telah terjadi perubahan nilai pada sensor. Karena pengguna melakukan gerakan rotasi badan gitar, maka sumbu z pada akselerometer mengalami perubahan dan akan dilakukan proses *increment* pada sebuah variabel, misalkan h. Nilai h yang awalnya bernilai 1 akan menjadi 2 sehingga pada fungsi loadPDPatch() akan dipanggil *patch* pure data *tremolo* yang akan digunakan untuk memodifikasi suara gitar. Urutan *patch* pure data yang digunakan ialah wah-wah, *tremolo* dan distorsi. Sehingga apabila efek yang digunakan saat ini adalah wah-wah, maka efek berikutnya yang akan digunakan adalah efek *tremolo* dan seterusnya hingga efek suara wah-wah digunakan kembali.

4.2.3.4 Sequence Diagram Menonaktifkan Efek

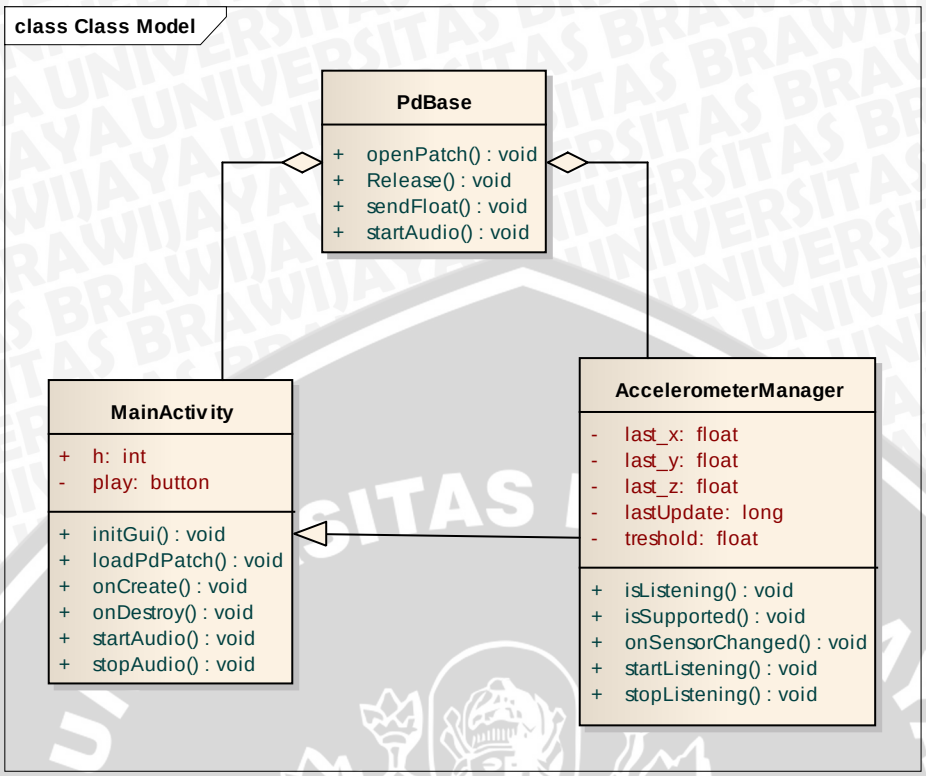


Gambar 4.11 *Sequence* diagram menonaktifkan efek

Gambar 4.11 merupakan *sequence* diagram yang digunakan pengguna saat pengguna ingin menonaktifkan penggunaan efek suara dari aplikasi. Pengguna menekan tombol on/off yang terdapat pada interface, kemudian aplikasi akan memanggil fungsi loadPDPatch untuk memanggil *patch* standar yang akan menghasilkan suara gitar tanpa modifikasi.

4.2.4 Perancangan *Class* Diagram

Class diagram merupakan diagram yang memodelkan elemen-elemen class yang membentuk sebuah sistem. *Class* diagram juga menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berhubungan. Gambar berikut ini merupakan *class* diagram dari aplikasi.



Gambar 4.12 *Class* diagram

Pada gambar 4.12, digambarkan perancangan kelas berupa *class diagram* pada aplikasi efek gitar dengan kendali pergerakan ini. Pada *class* diagram tersebut terdapat relasi *inheritance* yang ditunjukkan pada relasi antara kelas MainActivity dengan kelas AccelerometeManager. Relasi tersebut menyatakan bahwa kelas MainActivity dapat menurunkan *property* atau *method* dari kelas AccelerometerManager sehingga kelas AccelerometerManager sebagai *subclass* akan mewarisi *attribute* dan *method* dari MainActivity sebagai *superclass* dan dapat menggunakannya. Sedangkan relasi lainnya yang ada pada *class diagram* sistem ini adalah relasi *aggregation* yang salah satunya ditunjukkan pada relasi antara kelas MainActiviy dengan kelas PdBase. Relasi tersebut menyatakan bahwa kelas PdBase dapat berbagi objek dengan kelas MainActivity. Untuk penjelasan dari tiap-tiap *method class diagram* pada sistem ini, dapat dilihat pada Tabel 4.5 dan Tabel 4.6.

Tabel 4.5 Penjelasan method pada *class* MainActivity

Nama *Class*: `MainActivity`
Deskripsi: merupakan kelas utama dari aplikasi yang berfungsi untuk menjalankan modul pemrosesan suara pada aplikasi
Nama Method: `initGui()` **Fungsi:** Method ini berfungsi untuk menginisialisasi layout dan button yang akan digunakan pada *class*
Nama Method: `loadPDPatch()` **Fungsi:** Method ini berfungsi untuk memanggil *patch* pure data yang ingin digunakan oleh aplikasi
Nama Method: `onCreate()` **Fungsi:** Method ini merupakan method yang pertama kali dijalankan oleh aplikasi yang berisi deklarasi komponen-komponen utama yang digunakan oleh aplikasi
Nama Method: `onDestroy()` **Fungsi:** Method ini berfungsi untuk menghapus secara permanen proses yang sedang berjalan
Nama Method: `startAudio()` **Fungsi:** Method ini berfungsi untuk menjalankan modul audio pada aplikasi
Nama Method: `stopAudio()` **Fungsi:** Method ini berfungsi untuk memberhentikan modul audio yang sedang berjalan

Tabel 4.6 Penjelasan method pada *class* AccelerometerManager

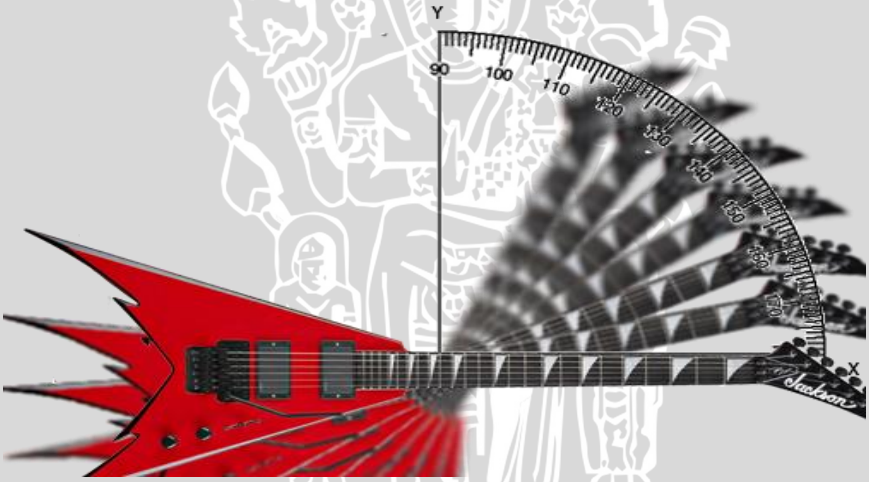
Nama *Class*: `AccelerometerManager`
Deskripsi: kelas yang digunakan untuk menjalankan sensor akselerometer yang akan digunakan oleh aplikasi
Nama Method: `isListening()` **Fungsi:** Method ini berfungsi untuk mendeteksi apakah sensor sudah berada dalam kondisi listening
Nama Method: `isSupported()` **Fungsi:** Method ini berfungsi untuk mendeteksi apakah perangkat yang digunakan sudah mendukung sensor akselerometer
Nama Method: `onSensorChanged()` **Fungsi:** Method yang berfungsi menjalankan event tertentu apabila terjadi perubahan pada sensor
Nama Method: `startListening()` **Fungsi:** Method ini berfungsi menjalankan proses listening pada sensor
Nama Method: `stopListening()` **Fungsi:** Method ini berfungsi memberhentikan proses listening

4.2.5 Perancangan Pergerakan

Pergerakan pemain gitar digunakan sebagai kontrol dan masukan dari aplikasi. Pada penelitian ini terdapat dua perancangan dari pergerakan yang digunakan yaitu perancangan pergerakan untuk mengatur intensitas suara setiap jenis efek dan perancangan pergerakan untuk mengganti jenis efek.

4.2.5.1 Pergerakan Untuk Mengatur Intensitas

Kebanyakan gitaris dapat mengarahkan gitarnya sekitar 85° dari garis *horizontal* (Willet, 2008). Pada aplikasi ini, smartphone android akan diletakkan di bagian *headstock* gitar dengan orientasi *landscape*, sehingga berdasarkan sistem koordinat sensor yang ada pada Gambar 2.1 apabila *headstock* gitar diarahkan keatas maka akan mempengaruhi sumbu y dari akselerometer. Pada Gambar 4.13 Pengguna dapat mengatur intensitas suara efek gitar dengan menggerakkan gitarnya kearah atas. Apabila posisi gitar berada di titik 10° maka nilai dari sumbu y sama dengan 1, apabila berada di titik 20° maka nilai dari sumbu y sama dengan 2 begitu seterusnya sampai nilai y sama dengan 9. Nilai sumbu y dari akselerometer ini nantinya akan menjadi masukan dari *patch* pure data sehingga intensitas suara efek yang dihasilkan berbeda setiap gerakannya.



Gambar 4.13 Perancangan pergerakan untuk mengatur intensitas

4.2.5.2 Perancangan Untuk Mengganti Jenis Efek

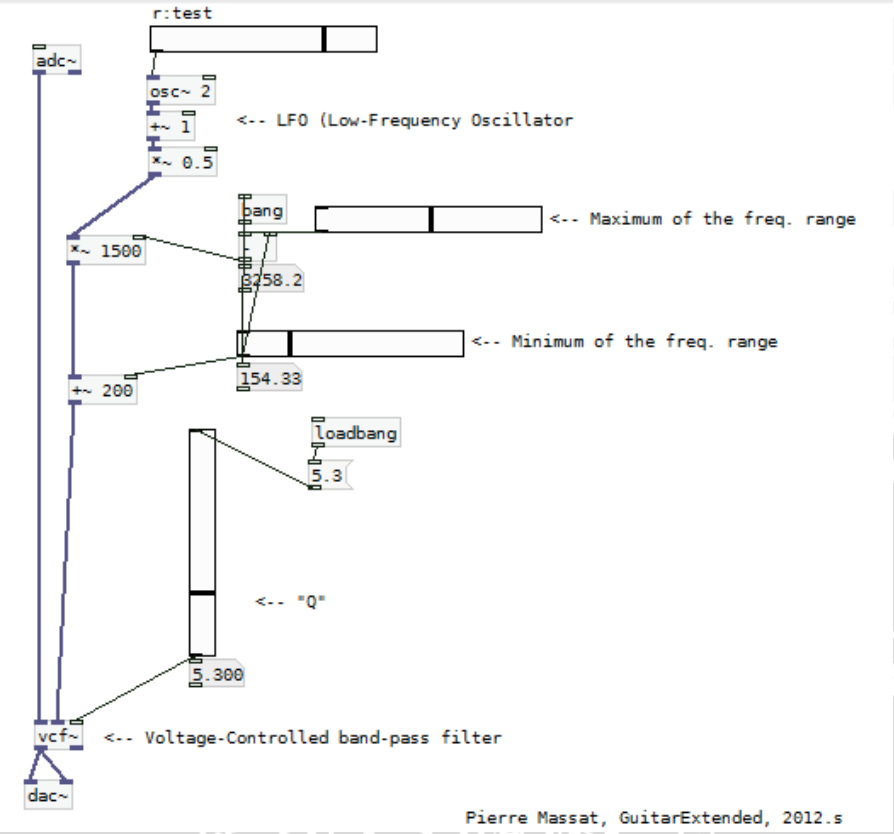
Pemain gitar juga dapat merotasi badan gitarnya sebanyak 45° pada garis *horizontal* dengan cara melengkung punggungnya (Willet, 2008). Pada Gambar 4.14 menjelaskan pengguna dapat merotasi badan gitarnya kearah atas pada garis *horizontal*. Merotasi gitar kearah atas akan mempengaruhi nilai sumbu z dari akselerometer sehingga nantinya apabila terjadi perubahan nilai pada sumbu z dan nilai perubahannya melebihi dari nilai *threshold* yang ditentukan, maka akan dipanggil *patch* berikutnya yang akan menggantikan *patch* sebelumnya sehingga gitar menghasilkan efek suara yang berbeda dari efek suara sebelumnya.

Gambar 4.14 Perancangan pergerakan untuk mengganti jenis efek

4.2.6 Perancangan *Patch* Pure Data

Patch pure data merupakan program hasil jadi dari pure data yang berisi representasi grafis dari aliran sinyal audio dan pesan kontrol yang akan dieksekusi secara *real time*. *Patch* pure data pada aplikasi ini digunakan untuk memodifikasi *input* suara dari gitar elektrik menjadi lebih bervariasi. Dalam penelitian ini terdapat 4 jenis *patch* pure data yang digunakan, yaitu *patch* wha-wha, *patch tremolo*, *patch* distorsi dan *patch* standar. *Patch* untuk mengubah efek suara ini diambil/bersumber pada Pierre Massat, guitarextended.wordpress.com.

4.2.6.1 Perancangan *Patch* Wha-Wha

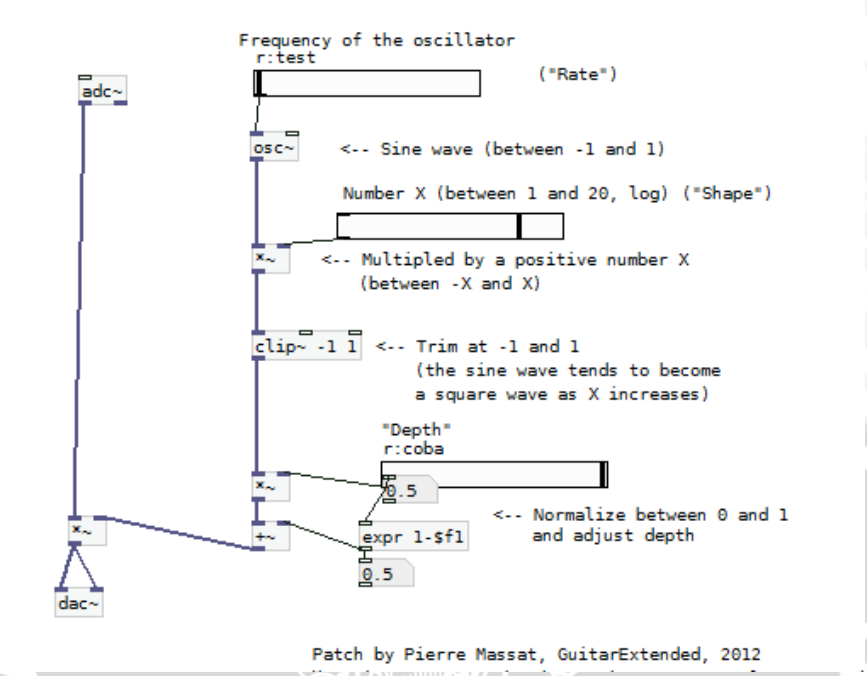


Gambar 4.15 Wha-wha *patch*

Sumber: (Massat, 2012)

Pada Gambar 4.15 masukan suara dari gitar elektrik yang berupa sinyal analog akan diubah menjadi sinyal digital oleh objek adc. Kemudian sinyal tersebut akan difilter oleh objek vcf. Frekuensi tengah pada vcf dapat diatur oleh sinyal audio, yang berarti bahwa frekuensi tengah ini dapat dikendalikan oleh LFO (Low Frequency Oscillator). Sinyal digital nantinya akan terus-menerus berjalan melalui frekuensi yang ditentukan melalui slider. Dan masukan pada objek osc digunakan sebagai kecepatan tempo dari sinyal dan juga masukan dari nilai sumbu y dari akselerometer.

4.2.6.2 Perancangan *Patch Tremolo*

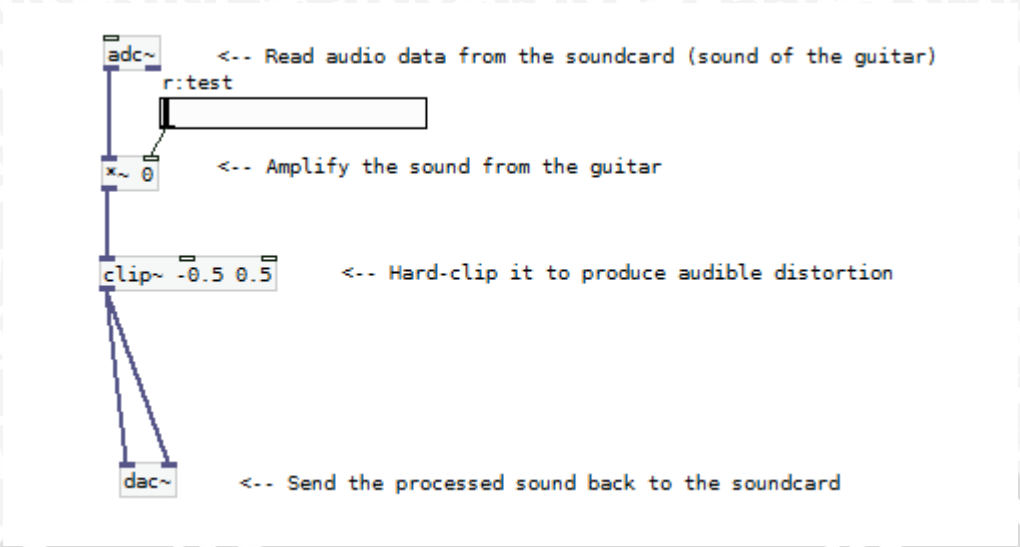


Gambar 4.16 *Tremolo patch*

Sumber: (Massat, 2012)

Pada Gambar 4.16 masukan suara dari gitar elektrik yang berupa sinyal analog akan diubah menjadi sinyal digital oleh objek adc. Kemudian sinyal tersebut dikalikan oleh *output* dari oscillator. *Output* ini disesuaikan sehingga sinyal dari gitar dikalikan dengan angka antara 0 dan 1 (kedalaman maksimal). Apabila nilai oscillator mendekati minimum, sinyal dikalikan dengan nilai yang lebih kecil dan akan mengurangi volume. Dan apabila nilai bertambah menjadi maksimal, maka volume akan meningkat kembali. Masukan objek osc akan dijadikan masukan dari nilai sumbu y dari akselerometer yang akan mempengaruhi tempo dari suara yang dihasilkan.

4.2.6.3 Perancangan *Patch* Distorsi

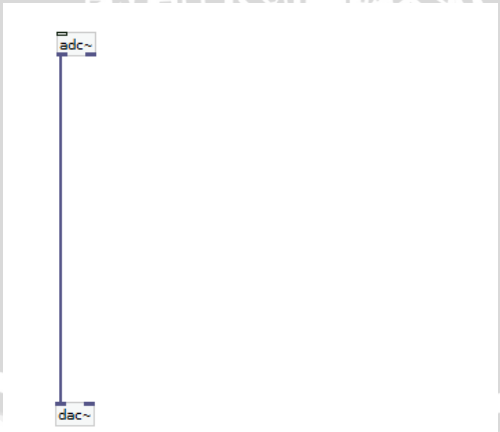


Gambar 4.17 Distortion *patch*

Sumber: (Massat, 2012)

Pada Gambar 4.17 suara gitar elektrik yang menjadi masukan berupa sinyal analog akan diubah menjadi sinyal digital agar dapat diproses. Kemudian dilakukan pengerasan suara berdasarkan *input* dari akselerometer. Kemudian masukan sinyal akan dipotong (*clipping)* agar menghasilkan suara distorsi. Sinyal hasil pemrosesan kemudian akan diubah (*convert*) kembali menjadi sinyal analog dan dikeluarkan melalui *sound output*.

4.2.6.4 Perancangan *Patch* Standar

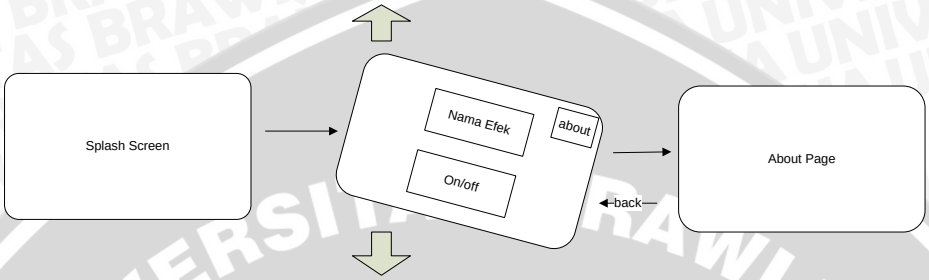


Gambar 4.18 Standar *patch*

Pada patch standar, suara masukan gitar dari adc akan langsung dikeluarkan melalui dac sehingga akan menghasilkan suara gitar murni tanpa adanya modifikasi efek suara.

4.2.7 Perancangan Page Flow

Perancangan *page flow* menjelaskan tentang alur yang terdapat pada antarmuka aplikasi. Perancangan *page flow* harus mengacu hasil proses analisis kebutuhan. Perancangan ini juga dilakukan untuk memudahkan dalam proses implementasi, sehingga dapat diketahui *layout* apa saja yang diperlukan agar tampilan sesuai dengan perancangan *page flow* ini. Perancangan *page flow* dapat diagram dapat diamati pada Gambar 4.19.



Gambar 4.19 *Page flow* diagram

Pertama kali ketika pengguna membuka aplikasi maka aplikasi akan menampilkan halaman *splash screen*, setelah itu aplikasi akan menampilkan halaman utama dari aplikasi yang berisi tombol on/off, tombol about dan sebuah *text view* yang akan menampilkan nama dari jenis efek yang dihasilkan. Saat pengguna menekan tombol on/off maka fitur utama dari aplikasi dijalankan dan pada halaman inilah pemrosesan sinyal suara dari gitar akan dilakukan. Pada halaman ini pengguna hanya perlu menggerakkan smartphone untuk memilih jenis suara efek maupun mengubah intensitas suara efek. Sedangkan tombol *about* digunakan untuk menampilkan halaman *about* yang berisi cara menggunakan aplikasi.

BAB 5 IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dari aplikasi berdasarkan analisis kebutuhan dan proses perancangan yang telah dilakukan. Pembahasan pada bab ini terdiri dari penjelasan tentang spesifikasi sistem, batasan-batasan dalam implementasi, dan implementasi kode program.

5.1 Spesifikasi Sistem

Dalam pembuatan aplikasi efek gitar dengan kendali pergerakan ini membutuhkan perangkat keras dan perangkat lunak. Berikut ini akan dijelaskan spesifikasi perangkat keras dan perangkat lunak yang digunakan.

5.1.1 Spesifikasi Perangkat Keras

Spesifikasi lingkungan perangkat keras yang digunakan dalam proses pengembangan aplikasi ini ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi perangkat keras komputer

Nama Komponen	Spesifikasi
System Model	Asus A46CB
Processor	1,8 GHz Core i3 3217U
Memory	4GB
Display	NVIDIA GT740M

Dalam proses implementasi dan pengujian, perangkat keras yang digunakan adalah smartphone android dengan spesifikasi perangkat keras pada Tabel 5.2 berikut.

Tabel 5.2 Spesifikasi perangkat keras Android

Nama Komponen	Spesifikasi
System Model	Asus Zenfone 2
Processor	Quadcore 1.8 GHz
Memory	2GB
Sensor	*Accelerometer*, *proximity* dan *compass*
Usb	microUSB v2.0, USB Host

5.1.2 Spesifikasi Perangkat Lunak

Spesifikasi lingkungan perangkat lunak komputer yang digunakan dalam proses pengembangan aplikasi ini ditunjukkan pada Tabel 5.3.

Tabel 5.3 Spesifikasi perangkat lunak komputer

Nama Komponen	Spesifikasi
Sistem Operasi	Windows 8.0
Bahasa Pemrograman	Java dan Pure Data
Library	Libpd
IDE (*Integrated Development Environment*)	Eclipse dan PD-extended

Sedangkan perangkat lunak android yang digunakan memiliki spesifikasi pada Tabel 5.4 berikut.

Tabel 5.4 Spesifikasi perangkat lunak Android

Nama Komponen	Spesifikasi
OS mobile	Android OS 5.0 (Lollipop)

5.2 Implementasi Kode Program

Pada aplikasi efek gitar dengan kendali pergerakan ini memiliki beberapa fungsi utama yang ada dibeberapa kelas. Pada penulisan laporan skripsi ini hanya dicantumkan algoritma dari beberapa proses saja. Pada subbab berikut akan dijelaskan mengenai algoritma pendeteksian perubahan sensor akselerometer dan algoritma memanggil *patch* pure data.

5.2.1 Algoritma Pendeteksian Perubahan Sensor Akselerometer

```
1  public void onSensorChanged(SensorEvent sensorEvent) {
2  Sensor mySensor = sensorEvent.sensor;
3  if (mySensor.getType() == Sensor.TYPE_ACCELEROMETER) {
4          float x = sensorEvent.values[0];
5          float y = sensorEvent.values[1];
6          float z = sensorEvent.values[2];
7          long curTime = System.currentTimeMillis();
8
9          if (lastUpdate == 0 && z > 6) {
10             last_z = 6;
11             lastUpdate = curTime;
12         } else {
13         if ((curTime - lastUpdate) > 600) {
14         last_z = 0;
```

```
15              lastUpdate = curTime;
16
17               float dif= Math.abs(last_y-y);
18              if(dif>1){
19                  last_y = y;
20                PdBase.sendFloat("test", last_y);
21               }
22
23
24          Float force = Math.abs(z - last_z);
25          if (Float.compare(force, threshold) >0 ){
26              if(h==3){
27                 h=1;
28                 PdBase.release();
29                 listener.loadPDPatch(h);
30                 listener.startAudio();
31
32              }
33          else{
34               PdBase.release();
35                h++;
36              listener.loadPDPatch(h);
37              listener.startAudio();
38                              }
39          }
40          }
41  }
42  }
```

Gambar 5.1 Algoritma pendeteksian perubahan sensor

Pada Gambar 5.1, ditampilkan implementasi kode program algoritma pendeteksian perubahan sensor yang akan menjadi acuan fitur efek gitar yang akan digunakan. Berikut beberapa penjelasan mengenai kode program pada Gambar 5.1:

1. Baris 1 merupakan nama fungsi yang terletak pada kelas AccelerometerManager yang dijalankan apabila sensor mengalami perubahan.
2. Baris 4-6 merupakan inisialisi variabel x,y,z yang digunakan untuk mendapatkan nilai dari tiap-tiap sumbu pada akselerometer.
3. Baris 7 merupakan inisialisi variabel Curtime yang digunakan untuk menyimpan waktu saat ini dalam satuan *millisecond* sehingga fungsi tidak dijalankan terus-menerus.
4. Baris 9-11 merupakan proses yang akan dilakukan saat pertama kali aplikasi dijalankan.
5. Baris 17-20 merupakan proses yang dilakukan jika terjadi perubahan nilai sumbu y pada akselerometer sehingga nantinya nilai y inilah yang dikirimkan ke *patch* pure data sehingga dapat mengubah intensitas suara efek yang digunakan.
6. Baris 24-37 merupakan proses yang akan dilakukan jika terjadi perubahan nilai sumbu z pada akselerometer dan digunakan untuk mengganti jenis efek suara. Proses ini akan dilakukan jika variable force lebih besar dari pada nilai *threshold* yang ditentukan. Apabila nilai h sama dengan 3 maka nilai h akan diganti menjadi 1 sehingga *patch* yang akan diambil kembali

menjadi *patch* wah-wah. Jika nilai h tidak sama dengan 3, maka variable h akan diincrementkan sehingga akan mengambil *patch* selanjutnya.

5.2.2 Algoritma Memanggil *Patch* Pure Data

```
1  void loadPDPatch(int h){
2       Resources res = getResources();
3       File patchFile = null;
4       try {
5        PdBase.subscribe("android");
6        if (h==0){
7         InputStream in = res.openRawResource(R.raw.standar);
8         patchFile = IoUtils.extractResource(in, "standar.pd",
9  getCacheDir());
10        PdBase.openPatch(patchFile);
11         }
12
13       else if (h==1){
14         InputStream in = res.openRawResource(R.raw.wha);
15        patchFile = IoUtils.extractResource(in, "wha.pd",
16 getCacheDir());
17        PdBase.openPatch(patchFile);
18        }
19
20       else if (h==2){
21         InputStream in = res.openRawResource(R.raw.tremolo);
22         patchFile = IoUtils.extractResource(in, "tremolo.pd",
23 getCacheDir());
24        PdBase.openPatch(patchFile);
25        }
26
27      else if (h==3){
28        InputStream in = res.openRawResource(R.raw.fuzz);
29        patchFile = IoUtils.extractResource(in, "fuzz.pd",
30 getCacheDir());
31        PdBase.openPatch(patchFile);
32        };
33 } catch (IOException e) {
34  // TODO Auto-generated catch block
35   Log.e(TAG, e.toString());
36   finish();
37   }
38 finally {
39        if (patchFile != null) patchFile.delete();
40       }
41 }
42
```

Gambar 5.2 Algoritma memanggil *patch* Pure Data

Pada Gambar 5.2, ditampilkan implementasi kode program algoritma memanggil *patch* pure data yang akan digunakan untuk memilih *patch* mana yang akan digunakan oleh aplikasi. Berikut beberapa penjelasan mengenai kode program pada Gambar 5.2:

1. Baris 1 merupakan nama fungsi yang terletak di kelas MainActivity.
2. Baris 6-11 merupakan proses yang akan dijalankan jika nilai h yang dikirimkan sama dengan 0 dan digunakan untuk memanggil *patch* pure data standar yang akan menghasilkan suara gitar asli tanpa modifikasi.

3. Baris 13-18 merupakan proses yang akan dijalankan jila nilai h sama dengan 1 sehingga nantinya akan dipanggil *patch* pure data wah-wah yang menjadi urutan nomor 1 dari aplikasi ini.
4. Baris 20-25 merupakan proses yang akan dijalankan jika nilai h sama dengan 2 sehingga nantinya akan dipanggil *patch* pure data *tremolo* yang menjadi urutan nomor 2 dari aplikasi ini.
5. Baris 27-32 merupakan proses yang akan dijalankan jika nilai h sama dengan 3 sehingga nantinya akan dipanggil *patch* pure data fuzz/distorsi yang menjadi urutan terakhir dari aplikasi ini.

5.3 Implementasi Antarmuka Pengguna

Pada tahap ini akan ditampilkan hasil implementasi antarmuka pengguna aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer. Tampilan dari implementasi antarmuka pengguna aplikasi ini ditunjukkan pada Gambar 5.3 sampai dengan 5.6 berikut.

5.3.1 Halaman *Splash Screen*

Gambar 5.3 merupakan implementasi antarmuka *splash screen* yang muncul ketika aplikasi ini pertama kali dijalankan.

Gambar 5.3 Halaman *splash screen*

5.3.2 Halaman Utama

Gambar 5.4 merupakan implementasi halaman utama aplikasi yang ditampilkan setelah *splash screen* dijalankan. Pada halaman ini terdapat sebuah *button* yang digunakan untuk mengaktifkan atau menonaktifkan efek suara gitar. Pada halaman utama inilah fitur mengubah intensitas efek suara dan mengganti jenis efek suara dapat dijalankan hanya dengan menggerakkan smartphone. Kemudian pada pojok kanan atas terdapat menu *action bar* untuk menjalankan halaman *about*.

Gambar 5.4 Halaman utama aplikasi

5.3.3 Halaman *about*

Gambar 5.5 merupakan implementasi antarmuka halaman *about* yang ditampilkan dengan cara menekan tombol about yang berada di dalam *action bar* pada halaman utama aplikasi. Halaman ini berisi cara-cara menggunakan aplikasi.

Gambar 5.5 Halaman *about* aplikasi

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini dilakukan proses pengujian dan analisis terhadap aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer pada smartphone android. Proses pengujian dilakukan melalui 2 tahapan, yaitu pengujian validasi dan pengujian *usability*. Pengujian validasi dilakukan dengan menggunakan teknik pengujian *black-box* (*black-box testing*). Sedangkan pengujian *usability* dilakukan dengan metode kuisioner. Selanjutnya dilakukan analisis terhadap sistem untuk mengetahui hasil dari pengujian.

6.1 Pengujian

Proses pengujian pada aplikasi menggunakan pengujian validasi dan pengujian *usability.* Pengujian validasi dilakukan untuk menguji sisi fungsional sistem sedangkan pengujian *usability* digunakan untuk menguji sisi non-fungsional sistem.

6.1.1 Pengujian Validasi

Pengujian validasi dalam penelitian ini digunakan untuk mengetahui kesesuaian antara daftar-daftar kebutuhan yang merupakan hasil dari analisis kebutuhan dan kinerja sistem, pada setiap kebutuhan sistem dilakukan proses pengujian kasus masing-masing akan ditunjukkan pada Tabel 6.1 sampai dengan Tabel 6.4.

Tabel 6.1 Kasus uji validasi mengaktifkan efek

Nomor Kasus Uji	VAL_01
Nama Kasus Uji	Mengaktifkan Efek
Objek Uji	Kebutuhan Fungsional(UC-01)
Tujuan Pengujian	Memastikan aplikasi mampu mengubah suara gitar.
Prosedur Pengujian	Menekan tombol on/off untuk mengaktifkan efek.
Hasil yang Diharapkan	Efek suara berhasil diaktifkan sehingga mengubah/memodifikasi suara gitar dari pengguna.

Tabel 6.2 Kasus uji validasi mengubah intensitas efek

Nomor Kasus Uji	VAL_02
Nama Kasus Uji	Mengubah Intensitas Efek
Objek Uji	Kebutuhan Fungsional(UC-02)
Tujuan Pengujian	Memastikan aplikasi mampu mengubah intensitas suara pada setiap jenis efek suara.
Prosedur Pengujian	Menggerakkan gitar kearah atas.
Hasil yang Diharapkan	Terjadi perubahan intensitas pada efek suara.

Tabel 6.3 Kasus uji validasi mengganti efek

Nomor Kasus Uji	VAL_03
Nama Kasus Uji	Mengganti Efek
Objek Uji	Kebutuhan Fungsional(UC-03)
Tujuan Pengujian	Memastikan aplikasi mampu mengganti jenis efek gitar menjadi jenis lainnya
Prosedur Pengujian	Merotasi badan gitar ke atas
Hasil yang Diharapkan	Jenis efek suara berganti menjadi jenis efek suara lainnya.

Tabel 6.4 Kasus uji validasi menonaktifkan efek

Nomor Kasus Uji	VAL_04
Nama Kasus Uji	Menonaktifkan Efek
Objek Uji	Kebutuhan Fungsional(UC-04)
Tujuan Pengujian	Memastikan aplikasi mampu menonaktifkan fitur efek suara.
Prosedur Pengujian	Menekan tombol on/off saat aplikasi dalam keadaan aktif.
Hasil yang Diharapkan	Efek suara gitar dinonaktifkan sehingga menjadi suara gitar asli tanpa modifikasi.

Berdasarkan pada kasus uji yang dilakukan akan diperoleh hasil pengujian. Hasil pengujian validasi akan ditampilkan pada Tabel 6.5 berikut.

Tabel 6.5 Hasil pengujian validasi

Nomor Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan	Status Validasi
VAL_01	Efek suara berhasil diaktifkan sehingga mengubah/memodifikasi suara gitar dari pengguna.	Efek suara berhasil diaktifkan sehingga mengubah/memodifikasi suara gitar dari pengguna.	Valid
VAL_02	Terjadi perubahan intensitas pada efek suara.	Terjadi perubahan intensitas pada efek suara.	Valid
VAL_03	Jenis efek suara berganti menjadi jenis efek suara lainnya.	Jenis efek suara berganti menjadi jenis efek suara lainnya.	Valid
VAL_04	Efek suara gitar dinonaktifkan sehingga menjadi suara gitar asli tanpa modifikasi.	Efek suara gitar dinonaktifkan sehingga menjadi suara gitar asli tanpa modifikasi.	Valid

6.1.2 Pengujian *Usability*

Pengujian *usability* pada penelitian ini menggunakan metode kuisioner dengan cara mengajukan beberapa pertanyaan kepada 20 orang responden yang berasal dari kalangan musisi (pemain gitar) dan orang-orang lainya yang sudah tidak asing lagi dengan dunia musik. Kemudian dari data kuisioner tersebut diambil kesimpulannya terhadap sistem yang telah dibuat. Kategori jawaban menggunakan lima rentang, diantaranya:

1. Sangat tidak setuju
2. Tidak setuju
3. Biasa saja
4. Setuju
5. Sangat setuju

Pertanyaan yang diberikan kepada responden dibuat berdasarkan metode usability testing yaitu, mudah dipelajari (*learnability*), efisiensi (*efficiency*), mudah diingat (*memorability*), perlindungan terhadap kesalahan (*errors*), dan kepuasan (*satisfaction)* (Saputra, 2014).

Pada komponen *learnability*, diberikan pertanyaan dengan tujuan menjelaskan tingkat kemudahan pengguna atau *user* dalam berinteraksi saat pertama kali melihat atau berhadapan dengan sistem yang ada. Kasus uji pada komponen *learnability* dapat dilihat pada Tabel 6.6 berikut.

Tabel 6.6 Kasus uji *learnability*

No.	Pertanyaaan
1.	Apakah tulisan teks yang digunakan pada sistem mudah dan jelas?
2.	Apakah simbol-simbol gambar mudah dipahami?

Pada komponen *efficiency*, diberikan pertanyaan dengan tujuan menjelaskan seberapa cepat pengguna berinteraksi saat pertama kali mempelajari sistem. Kasus uji pada komponen *efficiency* dapat dilihat pada Tabel 6.7 berikut.

Tabel 6.7 Kasus uji *efficiency*

No.	Pertanyaaan
1.	Apakah suara gitar dapat didengar secara *real time* (tidak terdapat *delay*)?
2.	Apakah saat berpindah dari efek yang satu ke yang lainnya dapat dilakukan dengan cepat (tidak terdapat *delay*)?

Pada komponen *memorability,* diberikan pertanyaan dengan tujuan menjelaskan tingkat kemudahan pengguna dalam menggunakan sistem dengan

baik, setelah beberapa lama tidak menggunakannya. Kasus uji pada komponen memorability dapat dilihat pada Tabel 6.8 berikut.

Tabel 6.8 Kasus uji *memorability*

No.	Pertanyaaan
1.	Apakah sistem mudah dioperasikan?
2.	Apakah halaman pada sistem mudah diingat?
3.	Apakah icon yang ada telah lazim dan sudah anda kenal?

Pada komponen *errors*, diberikan pertanyaan dengan tujuan menjelaskan kemungkinan terjadinya *error* atau kesalahan yang dilakukan oleh pengguna dan seberapa mudah mereka dapat mengatasinya. Kasus uji pada komponen *errors* dapat dilihat pada Tabel 6.9 berikut.

Tabel 6.9 Kasus uji *errors*

No.	Pertanyaaan
1.	Apakah semua tombol dapat berfungsi dengan baik (tidak terdapat *error*)?
2.	Apakah nama efek suara yang ditampilkan sudah sesuai dengan efek suara yang dijalankan?

Pada komponen *satisfaction*, diberikan pertanyaan dengan tujuan menjelaskan tingkat kepuasan pengguna dalam menggunakan sistem yang telah dibuat. Kasus uji pada komponen *satisfaction* dapat dilihat pada tabel 6.10 berikut.

Tabel 6.10 Kasus uji *satisfaction*

No.	Pertanyaaan
1.	Apakah semua fitur yang ada telah mencakup kebutuhan yang diharapkan?
2.	Apakah anda merasa terbantu dengan adanya aplikasi ini dalam memainkan instrument gitar anda?

Berdasarkan hasil pengujian *usability* melalui kuisioner yang disebarkan kepada 20 orang responden, didapatkan akumulasi hasil pada tabel 6.11 berikut.

Tabel 6.11 Hasil pengujian *usability*

No.	Pertanyaan	1	2	3	4	5
1.	Apakah tulisan teks yang digunakan pada sistem mudah dan jelas?	0	0	4	12	4
2.	Apakah simbol-simbol gambar mudah dipahami?	0	0	1	14	5
3.	Apakah suara gitar dapat didengar secara *real time* (tidak terdapat *delay*)?	0	2	6	9	3
4.	Apakah saat berpindah dari efek yang satu ke yang lainnya dapat dilakukan dengan cepat (tidak terdapat *delay*)?	0	1	5	10	4
5.	Apakah sistem mudah dioperasikan?	0	0	0	9	11
6.	Apakah halaman pada sistem mudah diingat?	0	0	1	8	11
7.	Apakah icon yang ada telah lazim dan sudah anda kenal?	1	0	4	8	7
8.	Apakah semua tombol dapat berfungsi dengan baik (tidak terdapat *error*)?	0	0	4	9	7
9.	Apakah nama efek suara yang ditampilkan sudah sesuai dengan efek suara yang dijalankan?	0	0	1	12	7
10.	Apakah semua fitur yang ada telah mencakup kebutuhan yang diharapkan?	0	1	2	15	2
11.	Apakah anda merasa terbantu dengan adanya aplikasi ini dalam memainkan instrument gitar anda?	0	0	2	12	6

Keterangan:

1 = Sangat Tidak Setuju 3 = Netral 5 = Sangat Setuju

2 = Tidak Setuju 4 = Setuju

6.2 Analisis

Proses analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer ini. Proses analisis yang dilakukan adalah analisis pada pengujian validasi dan pengujian *usability*.

6.2.1 Analisis Hasil Pengujian Validasi

Proses analisis terhadap hasil pengujian validasi dilakukan dengan cara melihat kesesuaian antara hasil kinerja sistem dengan daftar kebutuhan fungsional. Berdasarkan dengan hasil pengujian validasi pada Tabel 6.5 dapat

disimpulkan bahwa implementasi dan fungsionalitas aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer telah memenuhi kebutukan yang telah dijabarkan pada tahap analisis kebutuhan. Analisis kebutuhan yang dimaksud adalah dapat menjalankan semua fitur aplikasi, yaitu mengaktifkan efek suara, mengubah intensitas efek, mengganti jenis efek, dan menonaktifkan efek suara.

6.2.2 Analisis Hasil Pengujian *Usability*

Proses analis terhadap hasil pengujian *usability* dilakukan dengan menggunakan skala linkert. Hasil perhitungan index presentase dari setiap pertanyaan ditunjukkan pada Tabel 6.12 dengan menggunakan rumus untuk menghitung total skor yang ditunjukkan pada Persamaan 6.1 dan rumus untuk melakukan perhitungan index persentase ditunjukkan pada Persamaan 6.2.

Tabel 6.12 Index persentase

No.	Pertanyaan	STS	TS	N	S	SS	Total Skor	Index (%)
Learnability								
1.	Apakah tulisan teks yang digunakan pada sistem mudah dan jelas?	0	0	4	12	4	80	80%
2.	Apakah simbol-simbol gambar mudah dipahami?	0	0	1	14	5	84	84%
Efficiency								
3.	Apakah suara gitar dapat didengar secara real time (tidak terdapat *delay*)?	0	2	6	9	3	73	73%
4.	Apakah saat berpindah dari efek yang satu ke yang lainnya dapat dilakukan dengan cepat (tidak terdapat *delay*)?	0	1	5	10	4	77	77%
Memorability								
5.	Apakah sistem mudah dioperasikan?	0	0	0	9	11	91	91%
6.	Apakah halaman pada sistem mudah diingat?	0	0	1	8	11	90	90%
7.	Apakah icon yang ada telah lazim dan sudah anda kenal?	1	0	4	8	7	80	80%
Errors								
8.	Apakah semua tombol dapat berfungsi dengan baik (tidak terdapat *error*)?	0	0	4	9	7	83	83%
9.	Apakah nama efek suara yang ditampilkan sudah sesuai dengan efek suara yang dijalankan?	0	0	1	12	7	86	86%

Satisfaction								
10.	Apakah semua fitur yang ada telah mencakup kebutuhan yang diharapkan?	0	1	2	15	2	78	78%
11.	Apakah anda merasa terbantu dengan adanya aplikasi ini dalam memainkan instrument gitar anda?	0	0	2	12	6	84	84%

Keterangan:

STS = Sangat Tidak Setuju N = Netral SS = Sangat Setuju

TS = Tidak Setuju S = Setuju

$$\text{Total Skor} = S_{STS} \times 1 + S_{TS} \times 2 + S_N \times 3 + S_S \times 4 + S_{SS} \times 5 \quad (6.1)$$

$$\text{Index}(\%) = (\text{Total Skor}/Y)100 \quad (6.2)$$

Y = Skor Likert Tertinggi × Jumlah Responden

Untuk dapat menentukan status dari Index persentase dalam berbagai aspek penilaian digunakan interpretasi skor likert pada tabel 6.13 berikut.

Tabel 6.13 Interpretasi skor Likert

Skor likert	Interpretasi Skor dengan interval = 20	Pilihan
1	0%-19.99%	Sangat Tidak Memuaskan
2	20%-39.99%	Tidak Memuaskan
3	40%-59.99%	Biasa
4	60%-79.99%	Memuaskan
5	80%-100%	Sangat Memuaskan

Berdasarkan dengan interpretasi skor likert maka dapat ditentukan status dari hasil pengujian *usability* yang ditunjukkan pada tabel 6.14 berikut.

Tabel 6.14 Status pengujian *usability*

Aspek Penilaian	Rata-rata persentase(%)	Status
Learnability	82%	Sangat Memuaskan
Efficiency	75%	Memuaskan
Memorability	87%	Sangat Memuaskan
Errors	84.5%	Sangat Memuaskan
Satisfaction	81%	Sangat Memuaskan
Rata-rata	81.9%	Sangat Memuaskan

Dari hasil pengujian usability yang telah dilakukan, menyatakan bahwa index persentase (%) yang didapatkan pada setiap aspek penelitian, mendapatkan hasil rata-rata sebanyak 81.9% yaitu dengan status sangat memuaskan. Dengan adanya hasil tersebut menunjukkan aplikasi layak diterima dan memudahkan pengguna dengan hasil sangat memuaskan.

BAB 7 PENUTUP

7.1 Kesimpulan

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

1. Perancangan aplikasi efek gitar dengan kendali pergerakan menggunakan akselerometer pada smartphone android pertama-tama dilakukan dengan membuat daftar kebutuhan. Dari daftar kebutuhan tersebut, didapatkan skenario yang digambarkan dari *use case* diagram. Kemudian *use case* diagram dikembangkan menjadi *activity* diagram, *class* diagram, dan *sequential* diagram. Kemudian dilakukan perancangan pergerakan dan perancangan *patch* pure data untuk memodifikasi masukan suara gitar elektrik.
2. Sistem telah diimplementasikan sesuai dengan hasil perancangan pada smartphone android menggunakan bahasa pemrograman java dengan memanfaatkan fitur akselerometer dan menggunakan *library* libpd untuk menghubungkan android dengan pure data sehingga aplikasi dapat memodifikasi suara gitar elektrik dan dapat dikendalikan menggunakan pergerakan pemain gitar.
3. Berdasarkan hasil pengujian validasi dengan metode *black box testing*, menunjukkan bahwa semua fitur atau kebutuhan fungsional telah sesuai dengan yang dibutuhkan.
4. Tingkat *usability* sistem berdasarkan aspek pengujian *usability* yang meliputi aspek penilaian *learnability*, *efficiency*, *memorability*, *errors*, dan *satisfaction* menunjukkan rata-rata sebesar 81.9% yaitu dengan status sangat memuaskan.

7.2 Saran

1. Perlu ditambahkan jenis efek suara lainnya seperti *delay*, *chorus*, dll.
2. Dapat dilakukan pengembangan OS mobile lainnya seperti Ios yang mendukung *low latency audio* sehingga diharapkan dapat mengurangi *latency* pada aplikasi.
3. Dapat dilakukan pengembangan menggunakan sensor lainnya pada smartphone seperti sensor *gyrometer*.

DAFTAR PUSTAKA

audio. 2014. *About audio interface*. [online] Tersedia di: <http://www.sweetwater.com/computer-audio/audio_interfaces/> [Diakses 20 Oktober 2015]

Developer. 2015. *Android Developer*. [online] Tersedia di: <http://developer.android.com/guide/topics/sensor/sensor_overview.html> [Diakses 15 September 2015]

Source Android. 2015. *The Android Source Code*. [online] Tersedia di: <https://source.android.com/devices/audio/usb.html> [Diakses 15 September 2015]

UCG102. 2015. *Audio Interface UCG102*. [online] Tersedia di: <http://www.music-group.com/Categories/Behringer/Computer-Audio/Audio-Interfaces/UCG102/p/P0198> [Diakses 20 Okteber 2015]

Andi. 2013. *Step by Step Menjadi Programmer Android.* Yogyakarta: Andi Offset.

Brewster, D. 2010. Introduction to Guitar Tone and Effects. *A Manual for Getting the Sound from Electric Guitars*.

Brinkman, P. 2012. Making Musical Apps. Sebastopol: O'Reilly Media.

Chandra, R. 2012. *Langsung Jago Gitar Otodidak.* Depok: Pustaka Makmur.

Fitrianto, Rahmanda., 2014. *Sistem Aplikasi Mobile E-Voting Menggunakan Platform Android*. S1. Universitas Brawijaya.

Graham, B. 2012. *Using an Accelerometer Sensor to Measure Human Hand Motion.* Massachusetts Institute of Technology.

Gruber, L. 2006. *Bang Pure Data.* Hofheim: Wolke Verlag.

Dharwiyanti, Sri. 2003. Pengantar Unified Modeling Language (UML). Kuliah Umum IlmuKomputer.Com

Hamilton, K. 2006. *Learning UML 2.0.* O'Reilly Media.

Hartono, S. S. 2014. *Pengembangan Voice Recognition System Pada Robot Guidance Di Pasar Swalayan.* [pdf] Tersedia di: <eprints.binus.ac.id/30155/> [Diakses 18 September 2015]

Haryanti, M. 2012. *Aplikasi Accelerometer 3 Axis untuk Mengukur Sudut Kemiringan(TILT) Engineering Model Satelit Di Atas Air Bearing.* Teknik Elektro, Universitas Kristen Petra.

Kirn, P. 2012. *About Libpd*. [online] Tersedia di: <http://libpd.cc/about/> [Diakses 20 September 2015]

Kurniawan, I. 2010. *Pengolahan Sinyal.* Politeknik Jambi.

Massat, Pierre. 2012. *Audio Effect for Guitar with Pure Data.* [online] Tersedia di:

<https://guitarextended.wordpress.com> [Diakses 18 September 2015]

Noviyandhika, A. 2009. *Jago GItar Elektrik.* Yogyakarta: Indonesia Cerdas.

Nugraha, A. 2011. *Metode Perancangan Program UML.* Bina Nusantara University.

Nugroho, A. 2009. *Belajar Otodidak Menjadi Gitaris Andal.* Yogyakarta: Andi Offset.

Nurhayati, A. 2010. *Analisis Pengujian Perangkat Lunak Augmented Reality.* Teknik Elektro, Intitut Teknologi Sepuluh November.

Nurhayati, D. 2010. *Pengolahan Audio.* Sistem Komputer. Universitas Diponegoro.

Rahadi, D. R. 2014. *Pengukuran Usability Sistem Menggunakan Use Questionnaire Pada Aplikasi Android.* Program Pascasarjana Universitas Bina Darma Palembang.

Rahman, V. 2011. *Aplikasi Sensor Accelerometer Pada Deteksi Posisi.* Jurusan Teknik Elektro, Universitas Diponegoro.

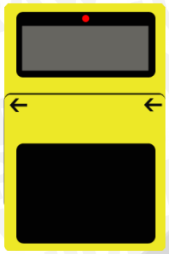
Saputra, E. 2014. *Usability Testing Untuk MEngukur Penggunaan Website Inspektorat Kota Palembang.* Universitas Bina Darma.

Wijaya, R. 2013. *Design and Implementation Audio Processing in Angstorm Beagleboard.* Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

Willet, W. 2008. *Motion Based Control For Guitar Effects Processing.* Barkeley Institute of Design.

LAMPIRAN A KUISIONER

A.1 Lembar Kuisioner *Usability Testing*



Aplikasi Efek Gitar dengan Kendali Pergerakan

Nama : .....

No. Identitas (KTP/SIM/NIM) : .....

Email : .....

No. HP : .....

-FEEDBACK-

Berilah tanda check(V) pada masing-masing pertanyaan dengan jawaban yang menurut Anda paling sesuai!

1 = Sangat Tidak Setuju
2 = Tidak Setuju
3 = Netral
4 = Setuju
5 = Sangat Setuju

No.	Pertanyaan	1	2	3	4	5
1.	Apakah tulisan teks yang digunakan pada sistem mudah dan jelas?					
2.	Apakah simbol-simbol gambar mudah dipahami?					
3.	Apakah suara gitar dapat didengar secara *real time* (tidak terdapat delay)?					
4.	Apakah saat berpindah dari efek yang satu ke yang lainnya dapat dilakukan dengan cepat(tidak terdapat delay)?					
5.	Apakah sistem mudah dioperasikan?					

6.	Apakah halaman pada sistem mudah diingat?					
7.	Apakah icon yang ada telah lazim dan sudah anda kenal?					
8.	Apakah semua tombol dapat berfungsi dengan baik (tidak terdapat *error*)?					
9.	Apakah nama efek suara yang ditampilkan sudah sesuai dengan efek suara yang dijalankan?					
10.	Apakah semua fitur yang ada telah mencakup kebutuhan yang diharapkan?					
11.	Apakah anda merasa terbantu dengan adanya aplikasi ini dalam memainkan instrument gitar anda?					

Kritik :

……

……

……

……

……

……

……

……

Saran :

……

……

……

……

……

……

……

……

Malang, November 2015

(……)

A.2 Hasil Kuisioner *Usability Testing*

No	Nama	Pertanyaan										
		1	2	3	4	5	6	7	8	9	10	11
1	Luthfi Anshori	5	5	3	4	5	5	5	5	4	5	5
2	Muh. Munim Sabirin	4	4	3	5	5	5	4	4	5	4	4
3	I Dewa Bagus Gde Khisma	3	4	3	4	5	4	5	4	4	4	5
4	Herdian Adi Winarno	4	4	2	3	4	4	4	4	4	4	3
5	I Made Riko Adi S.	3	4	2	4	4	4	5	4	4	4	4
6	I Gusti Ngurah Ersania	4	4	3	4	4	5	3	4	5	3	4
7	Bagus Putra Hardika	4	4	3	3	4	4	4	5	4	4	4
8	Rio Trilaksono Putro	5	5	4	5	5	5	4	4	4	4	5
9	Muhammad Murtadho	4	5	4	4	5	5	5	4	5	4	4
10	Heddy Sebastian E. P.	4	4	5	4	5	5	1	5	5	4	4
11	Adi Wiratama	5	5	4	4	5	5	4	5	5	5	5
12	Mareta Rizki Amelia	4	4	4	4	5	5	5	4	4	4	5
13	Tito V M	4	4	4	3	5	5	4	3	3	3	3
14	Alfian Mukmin Ali	3	3	5	5	4	4	5	5	4	4	4
15	Rohbi Visdya Harris	4	4	3	5	5	4	3	3	4	4	4
16	Nidia	5	4	4	3	4	5	3	5	4	4	5
17	Dhavin Putra A.	4	4	4	4	4	4	3	3	4	2	4
18	Alfiansyah Wikananda A	4	4	4	3	5	5	4	5	5	4	4
19	Sendy Deva Pastia	4	4	4	4	4	4	4	4	4	4	4
20	Eka Cahya Putra	3	5	5	2	4	3	4	3	5	4	4