TUGAS AKHIR - KI141502

ALAT BANTU DENGAR BERBASIS *SMARTPHONE* DENGAN MENGIMPLEMENTASIKAN KOMPRESI MEMBRAN TIMPANI DAN MEMBRAN BASILIAR

ARIANTO WIBOWO
NRP 5113100037

Dosen Pembimbing I
Dr. Chastine Fatichah, S.Kom., M.Kom.

Dosen Pembimbing II
Rully Soelaiman, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2017

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR - KI141502

ALAT BANTU DENGAR BERBASIS *SMARTPHONE* DENGAN MENGIMPLEMENTASIKAN KOMPRESI MEMBRAN TIMPANI DAN MEMBRAN BASILIAR

ARIANTO WIBOWO
NRP 5113100037

Dosen Pembimbing I
Dr. Chastine Fatichah, S.Kom., M.Kom.

Dosen Pembimbing II
Rully Soelaiman, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2017

(Halaman ini sengaja dikosongkan)

FINAL PROJECT - KI141502

HEARING AID BASED ON SMARTPHONE BY IMPLEMENTING TYMPANIC MEMBRANE AND BASILIAR MEMBRANE COMPRESSION

ARIANTO WIBOWO
NRP 5113100037

Supervisor I
Dr. Chastine Fatichah, S.Kom., M.Kom.

Supervisor II
Rully Soelaiman, S.Kom., M.Kom.

Department of INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2017

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

ALAT BANTU DENGAR BERBASIS *SMARTPHONE* DENGAN MENGIMPLEMENTASIKAN KOMPRESI MEMBRAN TIMPANI DAN MEMBRAN BASILIAR

TUGAS AKHIR
Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
ARIANTO WIBOWO
NRP: 5113100037

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Dr. Chastine Fatichah, S.Kom., M.Kom. .....
NIP: 197512202001122002 (Pembimbing 1)

Rully Soelaiman, S.Kom., M.Kom. .....
NIP: 197002131994021001 (Pembimbing 2)

SURABAYA
Juni 2017

(Halaman ini sengaja dikosongkan)

ALAT BANTU DENGAR BERBASIS *SMARTPHONE* DENGAN MENGIMPLEMENTASIKAN KOMPRESI MEMBRAN TIMPANI DAN MEMBRAN BASILIAR

Nama	:	**ARIANTO WIBOWO**
NRP	:	**5113100037**
Jurusan	:	**Teknik Informatika FTIf**
Pembimbing I	:	**Dr. Chastine Fatichah, S.Kom., M.Kom.**
Pembimbing II	:	**Rully Soelaiman, S.Kom., M.Kom.**

Abstrak

Gangguan pendengaran adalah salah satu gangguan kesehatan yang umumnya disebabkan oleh faktor usia atau karena sering terpapar suara yang keras. Gagalnya sinyal suara untuk mencapai otak merupakan penyebab utama gangguan pendengaran. Penderita gangguan pendengaran biasanya merasa kesulitan untuk mendengarkan suara yang terlalu kecil, di tempat yang ramai, ataupun kesulitan untuk mendengar pada frekuensi tertentu. Salah satu cara untuk mengatasi masalah tersebut adalah dengan menggunakan alat bantu dengar yang dipasang pada telinga penderita. Sayangnya, alat tersebut masih mahal dan belum terjangkau oleh sebagian masyarakat.

Alat bantu dengar berbasis smartphone merupakan salah satu solusi untuk mengatasi masalah gangguan pendengaran. Sistem memanfaatkan smartphone untuk digunakan sebagai alat bantu dengar. Penderita gangguan pendengaran dapat meng- unduh aplikasi ini di smartphone mereka, dan dapat segera digunakan menggantikan alat bantu dengar.

Tugas akhir ini bertujuan untuk membuat sebuah aplikasi alat bantu dengar menggunakan algoritma yang didesain untuk menirukan cara kerja telinga manusia. Aplikasi akan menerima

input suara smartphone dan mengeluarkan suara dengan kulitas yang telah diperbaiki ke earphone penderita gangguan telinga. Setelah diuji coba terhadap penderita gangguan telinga, aplikasi dapat membantu penderita untuk dapat mendengarkan suara dengan akurasi rata-rata sebesar 79.3%. Hasil pengujian ini cukup memuaskan karena penderita menjadi dapat mendengarkan suara tanpa perlu menggunakan alat bantu dengar.

***Kata-Kunci**: Gangguan Pendengaran, Smartphone*

HEARING AID BASED ON SMARTPHONE BY IMPLEMENTING TYMPANIC MEMBRANE AND BASILIAR MEMBRANE COMPRESSION

Name	:	**ARIANTO WIBOWO**
NRP	:	**5113100037**
Major	:	**Informatics FTIf**
Supervisor I	:	**Dr. Chastine Fatichah, S.Kom., M.Kom.**
Supervisor II	:	**Rully Soelaiman, S.Kom., M.Kom.**

Abstract

Hearing Impairment is one of the health problems usually caused by age or exposed to loud sound continuously. Failure of the sound signal to reach the brain is the main cause of hearing impairment. Hearing impaired often cannot hear soft sound, cannot hear in noise, and cannot hear sound in certain frequency. One of the solution is to wear hearing aids on the impaired ear. However, hearing aids are pretty expensive and some people cannot afford it.

Hearing Aids based on smartphone is an inexpensive solution for hearing impairments. System will use smartphone as the hearing aids. Hearing impaired can download the appli- cations in their smartphone so they can use the smartphone as the hearing aids.

This final project intend to create a hearing aids application on smartphone using algorithm designed to mimic how the human ear works. Application will receive sound input from the microphone and output sound with better quality to the earphone of the hearing impaired. After testing the system to hearing impaired, proposed system can help hearing impaired listening to sound with 79.3% accuracy. This result is satisfying as

hearing impaired will be able to understand speech without wearing hearing aids.

***Keywords**: Hearing Impairment, Smartphone*

KATA PENGANTAR

Puji Syukur kepada Tuhan Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul **Alat Bantu Dengar Berbasis *Smartphone* Dengan Mengimplementasikan Kompresi Membran Timpani dan Membran Basiliar**.

Harapan dari penulis semoga apa yang tertulis di dalam buku Tugas Akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini, serta dapat memberikan kontribusi yang nyata.

Dalam pelaksanaan dan pembuatan Tugas Akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Tuhan Yang Maha Esa.
2. Papa, Mama, adik, dan keluarga yang selalu memberikan dukungan, baik moral maupun material, secara penuh untuk menyelesaikan Tugas Akhir ini.
3. Bapak Rully Soelaiman yang telah meluangkan waktunya untuk membimbing penulis selama masa kuliahnya dalam perkuliahan, pembinaan kompetisi, serta Tugas Akhir ini.
4. Bu Chastine Fatichah selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberikan petunjuk selama proses pengerjaan Tugas Akhir ini.
5. Bapak, Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak memberikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
6. Seluruh staf dan karyawan FTIF ITS yang banyak memberikan kelancaran administrasi akademik kepada penulis.
7. Teman-teman penghuni laboratorium proyek Badak yang selalu memberi dorongan dan inspirasi kepada penulis.
8. Teman-teman 2013 Jurusan Teknik Informatika ITS yang telah menemani perjuangan selama 4 tahun ini atas saran,

masukan, dan dukungan terhadap pengerjaan Tugas Akhir ini.

9. Serta pihak-pihak lain yang tidak dapat disebutkan disini yang telah banyak membantu penulis dalam penyusunan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, 14 Juli 2017

Arianto Wibowo

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB 1

PENDAHULUAN

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

1.1 Latar Belakang

Perkembangan teknologi informasi dalam beberapa dekade terakhir sangat pesat. Teknologi informasi ini tidak hanya dapat digunakan sebagai sarana mencari informasi., tetapi juga dapat digunakan untuk membantu kehidupan sehari-hari. Salah satu teknologi yang sedang berkembang pesat adalah *Assistive Technology* [1]. *Assistive Technology* merupakan teknologi yang dapat digunakan untuk membantu penderita gangguan untuk melakukan hal-hal yang sebelumnya tidak dapat dilakukan dengan sempurna.

Salah satu gangguan yang sering dialami oleh manusia adalah gangguan pendengaran. Penderita gangguan pendengaran seringkali tidak dapat mendengarkan suara dengan intensitas rendah dengan jelas, kesulitan mendengar di kondisi yang ramai, dan tidak dapat menangkap suara dengan frekuensi tertentu [2]. Gangguan pendengaran ini dapat berpengaruh terhadap kemampuan untuk berbahasa yang mengakibatkan kesulitan dalam pekerjaan sehari-hari. Gangguan pendengaran dapat disebabkan oleh beberapa factor, antara lain genetik (keturunan), usia yang sudah tua, terkena polusi suara secara terus-menerus, ataupun terdapat infeksi pada alat pendengaran.

Gangguan pendengaran dapat diatasi menggunakan bahasa tubuh. Salah satu metode yang paling banyak digunakan agar penderita gangguan telinga dapat mengerti pembicaraan orang lain adalah dengan membaca gerak bibir pembicara. Dengan demikian, penderita dapat mengerti konteks pembicaraan

pembicara. Namun, terdapat kelemahan dalam Bahasa tubuh ini, yaitu penderita tidak dapat mendengarkan melalui telepon, tidak dapat menonton film, bioskop, ataupun tidak dapat mendengar dalam kondisi gelap.

Cara lain dalam mengatasi gangguan pendengaran adalah menggunakan alat bantu dengar. Alat bantu dengar tersebut menerima masukan suara, lalu melakukan proses perbaikan kualitas suara dan mengeluarkan suara yang sudah jernih dan sesuai dengan karakteristik pendengaran penderita [3]. Melalui alat ini, penderita gangguan dapat mendengarkan suara pembicara layaknya manusia dengan pendengaran normal. Namun, harga untuk alat bantu dengar cukup mahal dan kurang dapat dijangkau sebagian masyarakat. Rata-rata harga untuk sepasang alat bantu dengar tersebut adalah 60 juta rupiah dan dapat bervariasi sesuai kebutuhan penderita. Alat tersebut juga memiliki rata-rata harapan hidup selama 5 tahun saja. Padahal, berdasarkan data dari *World Health Organization* (WHO), pada tahun 2015 terdapat 360 jtua penderita gangguan telinga [4]. Penderita gangguan pendengaran tersebut didefinisikan sebagai orang yang tidak dapat mendengarkan suara pada 40 desibel. Di antara 360 juta penderita tersebut, terdapat sekitar 313 juta penderita yang berasal dari golongan menengah ke bawah. Melihat harga alat bantu dengar yang cukup mahal tersebut, alat tersebut tidak dapat dijangkau oleh sebagian besar penderita gangguan pendengaran. Alat tersebut hanya dapat mengatasi masalah sebagian penderita saja.

Untuk mengatasi masalah tersebut, penulis menerapkan sebuah metode perbaikan kualitas suara berbasis aplikasi *smartphone*. Pada beberapa dekade terakhir, perkembangan *smartphone* sangat pesat dan sebagian penduduk dunia telah memiliki minimal sebuah smartphone. Penulis berencana untuk membuat aplikasi *smartphone* yang menerima input dari audio pada *smartphone*, lalu melakukan proses perbaikan kualitas suara

pada *smartphone* tersebut. Suara yang sudah jernih kemudian akan dikeluarkan melalui *headset* atau earphone pengguna. Dengan demikian, diharapkan para pengguna *smartphone* tidak perlu membeli alat bantu dengar, tetapi cukup hanya dengan menggunakan aplikasi tersebut pada smartphone masing-masing.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut :

1. Membangun aplikasi berbasis *smartphone* untuk membantu penderita gangguan pendengaran yang menerima input suara dari *microphone* dan mengeluarkan suara yang telah diproses ke *earphone*.
2. Mengimplementasikan algoritma perbaikan suara sesuai dengan cara kerja pendengaran manusia.
3. Melakukan uji coba fungsionalitas aplikasi terhadap penderita gangguan pendengaran dan melakukan perbandingan terhadap aplikasi sejenis.

1.3 Batasan Masalah

Dari permasalahan yang telah diuraikan di atas, terdapat beberapa batasan masalah pada tugas akhir ini, yaitu:

1. Bahasa pemrograman yang akan digunakan adalah Bahasa pemrograman *Swift* untuk aplikasi perangkat bergerak
2. Aplikasi yang dikembangkan hanya dapat berjalan pada perangkat bergerak berbasis *iOS*.
3. Aplikasi hanya menerima masukan suara dari audio *smartphone*.
4. Aplikasi mengeluarkan suara yang telah diproses ke *earphone/headset* yang dipasang pada *smartphone*

1.4 Tujuan

Tujuan dari pengerjaan Tugas Akhir ini adalah membuat aplikasi berbasis *iOS* yang dapat digunakan untuk melakukan perbaikan suara dan mengeluarkan suara tersebut ke *earphone* pengguna yang menderita gangguan pendengaran.

1.5 Manfaat

Manfaat dari hasil pembuatan tugas akhir ini adalah menyediakan aplikasi *smartphone* sebagai solusi yang praktis yang dapat membantu penderita gangguan pendengaran sehingga dapat mendengarkan suara dari lingkungan sekitar layaknya manusia dengan pendengaran normal.

1.6 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

1.6.1 Penyusunan Proposal Tugas Akhir

Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat dari hasil pembuatan tugas akhir. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Sub bab metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

1.6.2 Studi Literatur

Tahap ini merupakan tahap pengumpulan informasi yang diperlukan untuk pengerjaan Tugas Akhir sekaligus mempelajarinya. Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan aplikasi, yaitu mengenai *tympanometry*, *acoustic reflex*, *butterworth filter*, *medial olivocochlear compression*, dan *noise gate*.

1.6.3 Analisis dan Perancangan Sistem

Pada tahap ini dilakukan analisa awal dan pendefinisian kebutuhan sistem untuk mengetahui permasalahan yang sedang dihadapi. Selanjutnya, dirumuskan rancangan sistem yang dapat memberi solusi terhadap permasalahan tersebut. Langkah yang akan digunakan pada tahap ini adalah sebagai berikut:

1. Pencarian dan pendataan algoritma yang akan digunakan dalam aplikasi ini.
2. Perancangan sistem dan mekanisme aplikasi.
3. Analisis kebutuhan non fungsional.
4. Analisis algoritma dan formula yang digunakan dalam aplikasi

1.6.4 Implementasi

Aplikasi ini akan dibangun dengan bahasa pemrograman *Swift* dengan *IDE XCode* dan dengan bantuan framework *AudioKit* untuk *Swift*. Aplikasi yang dibangun berbasis perang- kat bergerak dengan sistem operasi *iOS*.

1.6.5 Pengujian dan Evaluasi

Aplikasi akan diuji setelah selesai diimplementasikan menggunakan skenario yang sudah dipersiapkan. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan

perencanaan. Dengan melakukan pengujian dan evaluasi, dimaksudkan juga untuk mengevaluasi jalannya program, mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

1.6.6 Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam Tugas Akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini merupakan laporan secara lengkap mengenai Tugas Akhir yang telah dikerjakan baik dari sisi teori, rancangan, maupun implementasi sehingga memudahkan bagi pembaca dan juga pihak yang ingin mengembangkan lebih lanjut. Sistematika penulisan buku Tugas Akhir secara garis besar antara lain:

1.7.1 Pendahuluan

Bab ini berisi penjelasan latar belakang, rumusan masalah, batasan masalah dan tujuan pembuatan Tugas Akhir. Selain itu, metodologi pengerjaan dan sistematika penulisan laporan Tugas Akhir juga terdapat di dalamnya.

1.7.2 Dasar Teori

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

1.7.3 Analisis dan Perancangan Sistem

Bab ini berisi penjelasan tentang rancangan dari sistem yang akan dibangun.

1.7.4 Implementasi

Bab ini berisi penjelasan implementasi dari rancangan yang telah dibuat pada bab sebelumnya. Implementasi disajikan dalam bentuk pseudocode disertai dengan penjelasannya.

1.7.5 Pengujian dan Evaluasi

Bab ini berisi penjelasan mengenai data hasil percobaan dan pembahasan mengenai hasil percobaan yang telah dilakukan.

1.7.6 Kesimpulan dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

(Halaman ini sengaja dikosongkan)

BAB 2

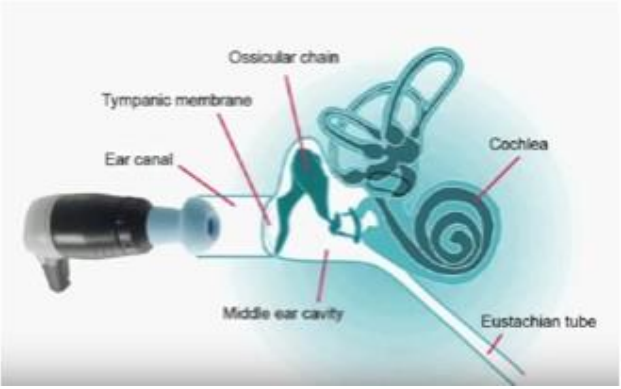
TINJAUAN PUSTAKA

Sistem yang akan dibangun menirukan cara kerja pendengaran pada manusia. Pada pendengaran manusia, suara akan melalui *tympanic membrane* yang dikendalikan oleh otot stapedius di dalam telinga. Pada otot stapedius tersebut, terdapat mekanisme *acoustic reflex* yang dapat diuji menggunakan *tympanometry*. Setelah itu, sinyal suara akan masuk ke *cochlea* (rumah siput) sebelum sinyal diteruskan ke otak. Di dalam rumah siput, terdapat *basiliar membrane* yang melakukan mekanisme dispersi frekuensi dari sinyal suara yang masuk. Proses dispersi frekuensi ini dapat diimplementasikan menggunakan *butterworth filter*. Selain itu, *basiliar membrane* juga melakukan proses *medial olivocochlear compression*. Setelah sinyal melalui rumah siput ini, sinyal baru akan diteruskan ke otak manusia. Untuk mengurangi *noise* yang terdapat pada suara, digunakan metode *noise gate*. Terakhir, suara akan dikeraskan pada tiap *range* frekuensi agar dapat didengarkan oleh penderita gangguan sesuai dengan karakteristik gangguan pendengaran yang dialami.

Dalam bab ini akan dijelaskan mengenai teori-teori yang merupakan dasar dari pembangunan sistem tersebut. Selain itu terdapat penjelasan yang menunjang pengerjaan Tugas Akhir ini sehingga dapat memberikan gambaran secara umum sistem yang akan dibangun.

2.1 Tympanogram

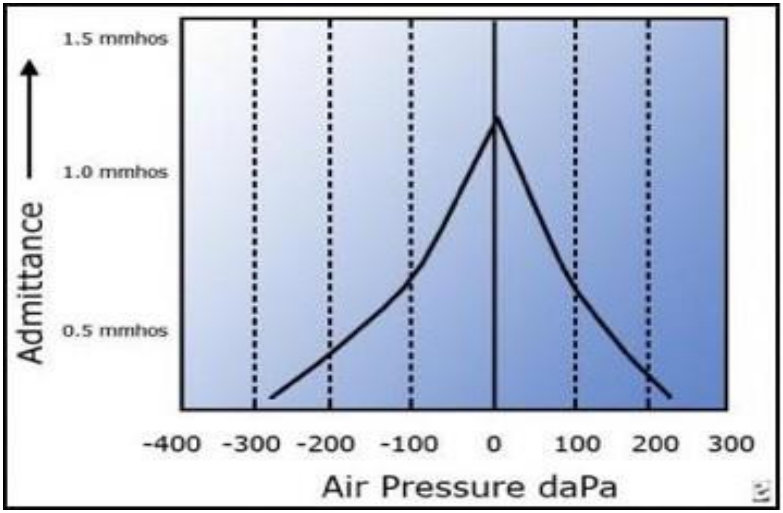
Tympanogram adalah sebuah tes yang digunakan untuk memeriksa keadaan di dalam telinga bagian tengah, tepatnya pada *tympanic membrane*. Tes ini dilakukan dengan meletakkan sebuah probe yang mengeluarkan suara di dalam telinga. Lalu, suara yang masuk ke dalam kanal telinga akan direkam untuk mengetahui cara kerja *tympanic membrane*.



Gambar 2.1: Rangkaian Tympanogram [5]

Gambar 2.1 menunjukkan rangkaian tes pada *tympanogram*. *Probe* akan mengeluarkan suara dengan frekuensi 226 Hz. Sebagian suara akan dipantulkan oleh *tympanic membrane*, yang akan diterima kembali oleh *probe*. Besarnya suara yang dipantulkan dapat digunakan untuk mengukur *compliance* (kemudahan suara untuk dapat diteruskan oleh *tympanic membrane*).

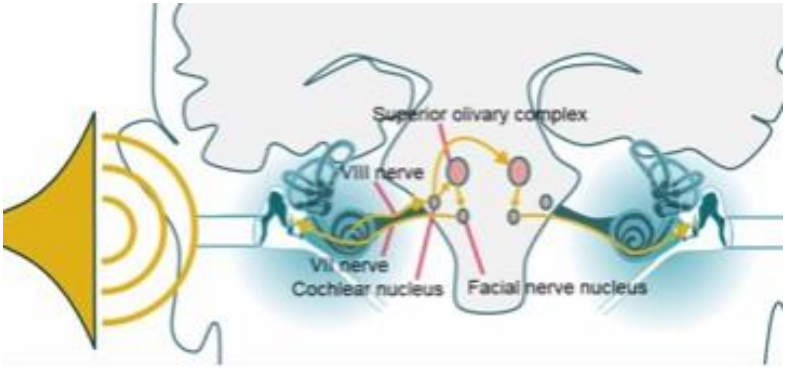
Pada pengujian, telinga akan diberikan suara dengan frekuensi 226 Hz dan tekanan (*pressure*) yang berbeda-beda. Telinga akan diberi tekanan dari 0 hingga batas tertentu yang menyebabkan nilai *compliance* stagnan, lalu tekanan akan dikurangi secara perlahan. Hal ini digunakan untuk menghi- langkan efek dari volume di dalam kanal telinga yang berbeda-beda untuk tiap pengujian. Hasil dari *tympanogram* adalah sebuah grafik yang menjelaskan *compliance* (mmho) terhadap *pressure* (dekapascal). Gambar 2.2 merupakan contoh hasil *tympanogram* pada pendengaran manusia yang normal. Nilai *compliance* maksimal tidak harus terdapat pada tekanan netral karena bergantung dengan keadaan tuba eustachius



Gambar 2.2: Hasil Tympanogram [5]

2.2 Acoustic Reflex

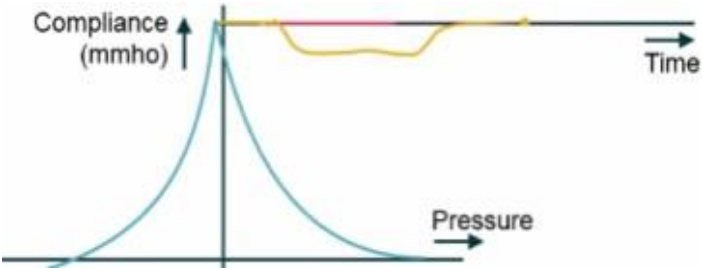
Acoustic Reflex adalah sebuah mekanisme dalam pendengaran manusia yang dilakukan oleh terjadinya kontraksi pada *stapedius muscle* (otot stapedius). Ketika terdapat suara dengan intensitas besar, suara akan masuk ke *cochlear* dan diterima oleh otak bagian *cochlear nucleus*. Suara akan dite- ruskan ke *superior olivary complex* dan diteruskan ke *facial nerve nucleus* yang dapat mengatur keadaan otot stapedius. Ketika suara yang diterima *facial nerve nucleus* terlalu keras, *nucleus* akan memberikan perintah kepada otot stapedius untuk berkontraksi. Kontraksi ini menyebabkan mengerasnya *tympanic membrane* (membran timpani) yang menyebabkan berkurangnya *compliance* dari suara yang masuk ke otak. Mekanisme ini secara lengkap dapat dilihat pada gambar 2.3



Gambar 2.3: Mekanisme Acoustic Reflex [6]

Gambar 2.3 menunjukkan mekanisme *acoustic reflex* secara lengkap di dalam telinga manusia. Otot stapedius ditunjukkan dengan tulang kecil berwarna kuning dalam gambar tersebut. Panah berwarna kuning merupakan alur sinyal suara keras yang masuk ke dalam pendengaran. Suara awalnya akan masuk ke otak yang diterima oleh *cochlear nucleus*, lalu suara keras akan memicu *facial nucleus* untuk melakukan kontraksi pada otot stapedius. Dapat dilihat bahwa suara keras yang diterima salah satu telinga (kiri saja dalam gambar) juga akan diteruskan ke *superior olivary complex* dari telinga kanan, sehinga otot stapedius telinga kanan juga akan berkontraksi terhadap suara yang diterima oleh telinga kiri, dan sebaliknya [7].

Untuk melakukan pengukuran *acoustic reflex* pada telinga, dibutuhkan bantuan *tympanometry*. Pertama, dilakukan proses tympanogram untuk mendapatkan tekanan dengan nilai *compliance* paling tinggi. Lalu, akan diberikan suara keras selama 1-2 sekon dengan mempertahankan tekanan di dalam kanal telinga. Hasil *tympanogram* pada pendengaran normal akan terlihat seperti pada gambar 2.4.



Gambar 2.4: Tympanogram Acoustic Reflex [6]

Nilai *compliance* memiliki *delay* penurunan pada saat suara keras diberikan pada telinga (warna merah muda pada garis waktu) dan juga terdapat *delay* kenaikan pada saat suara keras sudah tidak diberikan pada telinga. Pada telinga, pemberian suara keras akan menghasilkan mekanisme seperti pada gambar 2.4 dimana panah berwarna merah muda adalah suara keras dan panah kuning adalah suara yang tidak keras.

Gambar 2.5 menjelaskan mekanisme di dalam telinga saat terjadi pengukuran *acoustic reflex*. Warna merah muda menggambarkan alur suara yang keras, sedangkan warna kuning merupakan suara yang biasa saja. Dapat dilihat bahwa suara keras yang diterima oleh telinga kiri juga akan mempengaruhi otot stapedius dari telinga kanan, dan sebaliknya. Hal ini menyebabkan suara keras yang diterima oleh salah satu telinga akan menyebabkan telinga lainnya juga mengalami reduksi sinyal.

Gambar 2.5: Alur suara saat pengujian Acoustic Reflex [6]

Acoustic Reflex ini merupakan mekanisme proteksi yang dimiliki oleh telinga. Mekanisme ini melindungi organ dalam telinga saat telinga menerima suara yang sangat keras [8]. Organ telinga yang terpapar suara keras dapat mengalami kerusakan yang merupakan salah satu penyebab gangguan pendengaran. Selain itu, mekanisme ini juga dapat mereduksi suara dari pita suara individu tersebut. Pita suara manusia terletak sangat dekat dengan telinga, dan suara yang dihasilkan akan diterima oleh telinga manusia itu sendiri dengan sangat keras. Dengan adanya mekanisme ini, manusia tidak mendengar suaranya sendiri dengan terlalu keras, sehingga dapat mendengarkan suara dari lingkungan sekitar ketika sedang berbicara.

2.3 Butterworth Filter

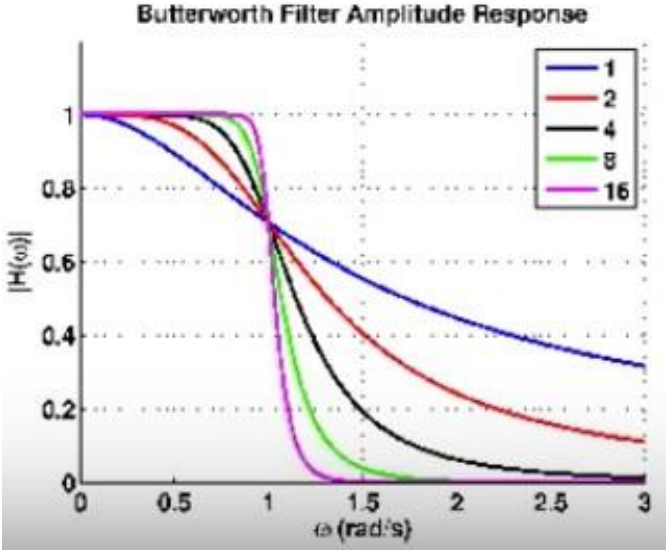
Butterworth Filter merupakan salah satu jenis *filter* yang sering digunakan dalam pemrosesan sinyal. Filter ini pertama ditemukan pada tahun 1930 oleh ilmuwan Inggris bernama Stephen Butterworth. Filter ini awalnya merupakan *low pass filter*, yaitu filter yang meneruskan sinyal dengan frekuensi di bawah *cutoff frequency* dan melemahkan frekuensi di atasnya [9]. Berikut adalah fungsi transfer pada *butterworth filter*:

$$|H(j\omega)| = \frac{1}{\sqrt{1 + (\frac{\omega}{\omega_c})^{2n}}} \tag{2.1}$$

ω_c merupakan *cutoff frequency*, yang mana frekuensi di bawah nilai tersebut akan diteruskan, sedangkan frekuensi di atasnya akan dikurangi. *n* merupakan *order* dari *butterworth filter*. Nilai n yang besar menyebabkan transisi *cutoff frequency* akan semakin curam, dan sebaliknya. Nilai n yang mendekati tak hingga akan menyerupai dengan desain filter ideal, yang mana meneruskan semua sinyal dengan frekuensi sesuai dengan *pass*

band yang ditentukan, dan menolak semua frekuensi pada *stop band*. Namun, filter ideal ini tidak dapat diimplementasikan karena memiliki *impulse response* yang jumlahnya tak hingga. *Butterworth Filter* dapat dimodifikasi untuk menjadi *Pass Band Filter*, yaitu filter yang hanya meneruskan sinyal dengan frekuensi $center_f requency \pm bandwidth$.

Gambar 2.6 merupakan grafik pengaruh nilai order butterworth terhadap respons frekuensi. Pada gambar tersebut, nilai dari *cutoff frequency* adalah 1, sehingga frekuensi kurang dari sama dengan 1 akan diteruskan, sedangkan frekuensi di atasnya akan dikurangi. Dari grafik dapat dilihat bahwa dengan naiknya nilai n, maka redaman terhadap *stop band* akan semakin curam. Dengan demikian, bertambahnya nilai n akan membuat filter menjadi lebih sempurna atau ideal.



Gambar 2.6: Pengaruh nilai n terhadap respons *butterworth* [10]

Salah satu keunggulan dari *butterworth filter* dibandingkan filter yang lain (*Bessel Filter*, *Chebyshev Filter*) adalah filter ini dapat bekerja dengan baik pada seluruh frekuensi [11]. Selain itu, pengurangan pada *stop band* bersifat monoton, sehingga sinyal yang dihasilkan lebih halus.

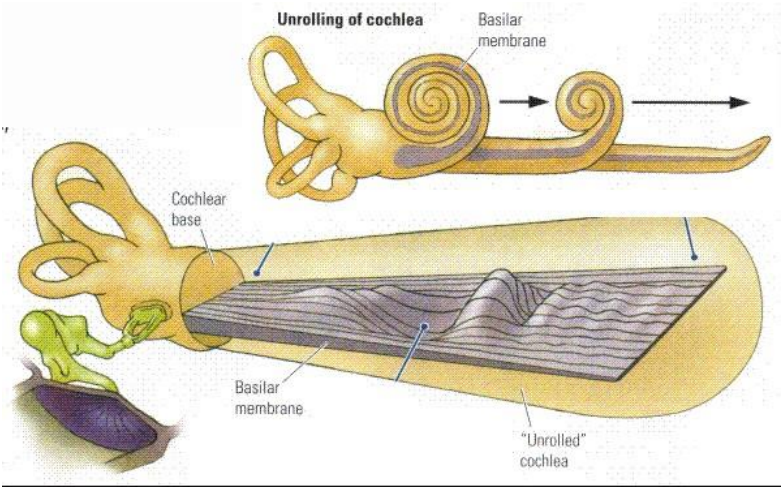
Pada telinga manusia, proses pemecahan frekuensi terjadi di dalam *cochlea* atau rumah siput. Dalam rumah siput, terdapat *basiliar membrane* yang memiliki tingkat sensitivitas yang berbeda-beda terhadap frekuensi suara yang masuk, sebelum akhirnya sinyal tersebut diteruskan ke otak. *Butterworth Filter* ini dipilih dengan harapan untuk dapat menirukan cara kerja rumah siput dalam memecah frekuensi yang masuk.

2.4 Channel gain

Channel gain adalah penambahan intensitas suara agar penderita gangguan pendengaran dapat mendengarkan suara tersebut. *Gain* akan diberikan pada tiap channel, sehingga konfigurasi nilai *gain* dapat berbeda-beda untuk tiap *channel*. Hal ini membantu penderita gangguan untuk dapat menye- suaikan sistem yang dibangun sesuai dengan karakteristik pendengarannya. Gangguan pendengaran dapat terjadi pada frekuensi yang berbeda-beda, ada yang dapat mendengar frekuensi rendah tetapi tidak dapat mendengar frekuensi tinggi, dan sebaliknya [12].

2.5 Medial Olivocochlear Compression

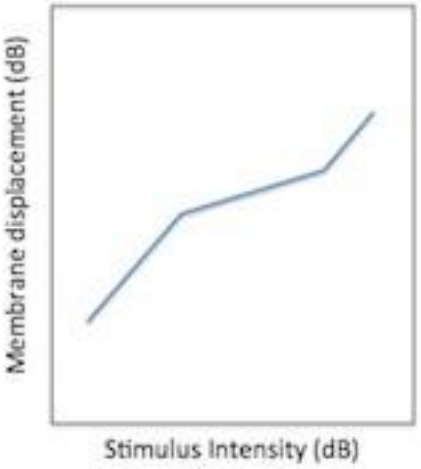
Medial Olivocochlear Compression merupakan proses yang terjadi di dalam *basiliar membrane* yang terdapat di dalam telinga. *Basiliar Membrane* sendiri merupakan selaput yang berada sepanjang *cochlea* dan berfungsi untuk melakukan pemecahan frekuensi dan meredam suara.



Gambar 2.7: Membran Basiliar [13]

Gambar 2.7 menggambarkan posisi membran basiliar yang berupa untaian panjang di dalam *cochlea* yang memiliki lebar, ketebalan, massa, dan kelembapan yang berbeda-beda pada tiap bagiannya. Membran ini memiliki fungsi utama yaitu untuk melakukan pemecahan sinyal berdasarkan frekuensi. Selain itu, membran ini juga berfungsi untuk mengurangi intensitas suara pada *range* tertentu.

Gambar 2.8 menjelaskan pergerakan membran terhadap stimulus yang berbeda-beda. Grafik tersebut menjelaskan terdapat suatu pola grafik yang patah sehingga kerap disebut "*broken stick*". Hal ini disebabkan pada intensitas rendah (kurang lebih 20 dB hingga 80 dB), stimulus suara akan diteruskan secara linear oleh membran [14]. Sementara itu, untuk tiap kenaikan intensitas 1 dB di atas *range* tersebut, membran hanya akan meneruskan sebesar 0.2 dB ke otak. Membran kembali meneruskan sinyal secara linear pada intensitas yang sangat tinggi.



Gambar 2.8: Respons membran terhadap stimulus [15]

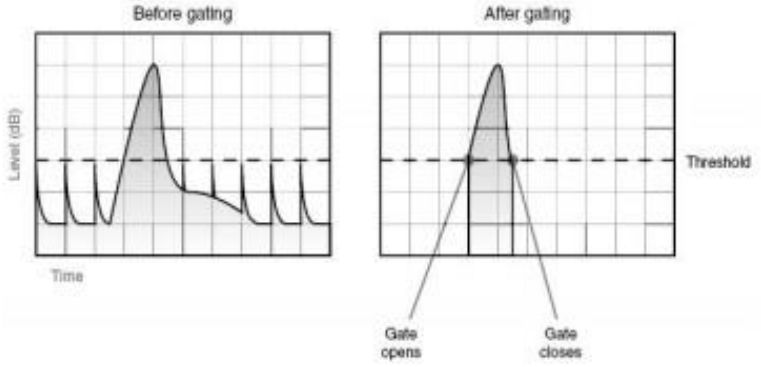
Untuk menirukan kinerja telinga tersebut, maka akan dilakukan kompresi (peredaman suara) terhadap suara di atas 80 dB. Proses ini bertujuan untuk mensimulasikan respons pada *basiliar membrane* sesuai dengan grafik pada gambar 2.8 yang menyerupai bentuk *broken stick*. Namun, proses *medial oli- vocochlear compression* ini tidak akan membuat respons stimulus linear kembali pada intensitas yang sangat tinggi. Hal ini disebabkan karena adanya proses *Acoustic Reflex Compression* yang diharapkan telah melakukan kompresi pada suara dengan intensitas tinggi tersebut. *Medial Olivocochlear Compression* memiliki perbedaan dengan *Acoustic Reflex Compression*, yang mana proses *acoustic reflex* dilakukan sebelum adanya filter terhadap frekuensi yang berbeda-beda (sinyal suara akan melalui otot stapedius terlebih dahulu, baru rumah siput), sedangkan *medial olivocochlear compression* dilakukan bersamaan dengan proses filter frekuensi pada *cochlea*. Kedua jenis kompresi ini juga bekerja pada *range*

intensitas suara yang berbeda.

2.6 Noise Gate

Noise adalah sinyal yang tidak diinginkan dalam sinyal digital. *Noise* dapat disebabkan oleh sirkuit elektrik, kuantisasi digital, ataupun *noise* dari lingkungan sekitar pada saat proses perekaman. Adanya noise tersebut di dalam suara dapat menye- babkan berkurangnya pemahaman pendengar terhadap suara yang disajikan dan juga mengurangi kenyamanan pendengar. Oleh karena itu, metode ini berusaha untuk menghilangkan *noise* yang terdapat dalam sinyal suara.

Dalam sudut pandang alat bantu dengar, yang membedakan *speech* dan *noise* adalah intensitas suara. *Speech* cenderung memiliki intensitas yang besar, sedangkan *noise* baik dari *microphone* maupun lingkungan sekitar memiliki intensitas yang relatif kecil [16]. Oleh karena itu, metode *noise gate* berusaha untuk meredam sinyal-sinyal yang memiliki intensitas kecil untuk mengurangi *noise*, dan meneruskan sinyal-sinyal yang memiliki intensitas besar.



Gambar 2.9: Noise Gate [16]

Noise gate merupakan suatu algoritma untuk mengurangi jumlah *noise* yang terdapat pada sinyal. Algoritma ini bekerja dengan memberikan suatu nilai *threshold* pada intensitas, yang mana suara dengan intensitas lebih dari *threshold* akan dianggap sebagai sinyal, sedangkan suara dengan intensitas lebih rendah dari *threshold* akan dianggap sebagai *noise*.

Gambar 2.9 merupakan ilustrasi cara kerja *noise gate* saat terdapat sinyal dengan intensitas berubah. Sinyal dengan intensitas di atas *threshold* akan diteruskan sementara sinyal dengan intensitas di bawah *threshold* akan dilemahkan. Untuk mengatur kemulusan suara, maka *noise gate* memiliki mekanisme *attack and release*, yaitu *delay* yang ditambahkan sebelum sinyal diteruskan (*attack*) dan *delay* yang diberikan sebelum sinyal dilemahkan (*release*). Diperlukan pemilihan parameter *attack* dan *release* yang baik agar percakapan dapat dipahami dengan mudah. Nilai *attack* dan *release* yang terlalu kecil menyebabkan percakapan sulit dipahami dikarenakan suara yang masuk akan terdengar seperti terputus-putus. Umumnya, nilai jeda waktu *release* lebih besar daripada jeda waktu *attack*.

2.7 Metode Evaluasi

Evaluasi kuantitatif akan dilakukan dengan mengujikan aplikasi terhadap penderita gangguan pendengaran. Penderita akan diminta untuk menggunakan aplikasi yang telah dibuat. Lalu, penguji akan mengucapkan beberapa kata untuk mengetes apakah penderita dapat mendengarkan atau tidak. Pendengar kemudian akan diminta untuk mengulangi kata yang telah diucapkan oleh penguji. Dari pengujian tersebut, didapatkan akurasi pengujian menggunakan rumus berikut:

$$akurasi = \frac{jumlah\_kata\_benar}{total\_kata} \tag{2.2}$$

Diharapkan akurasi ini dapat mewakili evaluasi apakah sistem yang dibangun dapat membantu penderita gangguan untuk dapat mendengar atau tidak.

2.8 Bahasa Pemrograman *Swift*

Swift merupakan bahasa pemrograman yang ditujukan untuk membangun aplikasi berbasis *macOS*, *iOS*, *watchOS*, dan *tvOS*. Bahasa pemrograman ini dikembangkan oleh *Apple* dan mulai diluncurkan ke publik pada tahun 2014 menggantikan bahasa pemrograman *Objective-C* yang sebelumnya digunakan untuk pemrograman *iOS* [17]. Bahasa pemrograman ini sangat inter- aktif dan mudah untuk dipelajari. *Compiler Swift* telah diopti- masi untuk meningkatkan performa, dan beberapa algoritma yang memiliki kompleksitas cukup tinggi ditulis dalam bahasa *C* untuk mengoptimasinya.

Untuk menulis kode dalam bahasa pemrograman *Swift*, diperlukan *IDE Xcode* yang juga dikembangkan oleh *Apple* dalam rangka membantu para developer untuk membangun aplikasi berbasis *macOS*, *iOS*, *watchOS*, dan *tvOS*. *IDE* ini menawarkan pemrograman yang interaktif antara kode dengan antarmuka yang dibangun. Developer dapat mendesain aplikasi dengan mudah dan menghubungkannya dengan kode *(logic)* yang diinginkan.

(Halaman ini sengaja dikosongkan)

BAB 3

DESAIN DAN PERANCANGAN

Pada bab ini diuraikan mengenai perancangan sistem perangkat lunak untuk mencapai tujuan Tugas Akhir. Dalam pembuatan suatu perangkat lunak, perancangan secara teknis tentang bagaimana aplikasi akan berjalan merupakan bagian awal yang amat penting dan harus dicermati. Sehingga bab ini secara khusus dibuat untuk menjelaskan perancangan sistem yang dibuat dalam tugas akhir ini, mulai dari deskripsi umum aplikasi hingga perancangan proses, alur, dan implementasinya.

3.1 Deskripsi Umum Sistem

Pada Tugas Akhir ini akan dibangun sebuah aplikasi yang berfungsi untuk memperbaiki kualitas suara yang dapat didengarkan oleh penderita gangguan pendengaran. Pengguna aplikasi dapat melakukan setelan terhadap keras-kecilnya suara yang akan dikeluarkan sesuai dengan karakteristik pendengaran masing-masing. Aplikasi akan menerima *input* suara dari *micro-phone* pengguna dan mengeluarkannya melalui *earphone* atau *headset* pengguna

Pertama, suara yang masuk akan melalui proses *acoustic reflex compression*. Proses ini akan mereduksi intensitas suara jika suara yang masuk sebelumnya terlalu keras. Setelah itu, suara akan dipisah berdasarkan frekuensinya menggunakan *butterworth filter*. Aplikasi akan membuat *filter bank* (memecah suara menjadi 9 sesuai dengan frekuensinya).

Setelah sinyal terpecah ke 9 area frekuensi, akan dilakukan proses *medial olivocochlear compression, channel gain*, dan *noise gate* pada masing-masing *channel*. Medial Olivocochlear merupakan algoritma untuk mengurangi intensitas suara yang diteruskan jika melebihi suatu *threshold*. Mekanisme ini menyerupai mekanisme *broken stick* yang terdapat pada telinga. Lalu, sinyal akan melalui proses *gain*, yaitu penambahan

intensitas suara agar dapat didengarkan oleh penderita gangguan pendengaran. Pengguna dapat mengkonfigurasikan setelan *gain* pada tiap *channel* sesuai dengan karakteristik pendengaran masing-masing. Suara yang sudah diberi nilai *gain* akan masuk ke *noise gate*, yang mana akan menghilangkan *noise* yang berasal dari perangkat keras maupun lingkungan sekitar.

Sinyal dari 9 *channel* kemudian akan digabungkan menjadi 1 lagi. Penggabungan dilakukan dengan melakukan penambahan sinyal dari keluaran masing-masing *channel*. Sebelum dikeluarkan ke *earphone* pengguna, dilakukan perhitungan nilai *acoustic reflex* terlebih dahulu, sehingga jika suara yang akan dikeluarkan terlalu keras, aplikasi dapat menguranginya untuk sinyal yang akan datang. Lalu, sinyal dikeluarkan ke *earphone* pengguna untuk didengarkan.

3.2 Arsitektur Umum Sistem

Aplikasi dibuat menggunakan bahasa pemrograman *Swift*. Bahasa ini dikeluarkan oleh *Apple* sebagai bahasa pemrograman untuk *iOS* yang menggantikan *Objective-C*. IDE yang digunakan adalah *XCode*, yaitu IDE yang disediakan oleh *Apple* yang memiliki antarmuka yang *friendly*. Bahasa dan IDE yang digunakan sesuai dengan kebutuhan *platform iOS*.

3.3 Perancangan Data

Perancangan data merupakan bagian yang penting dalam proses pengoperasian perangkat lunak. Data yang benar mem- buat perangkat lunak dapat beroperasi dengan benar. Data yang dibutuhkan dalam proses pengoperasian perangkat lunak adalah data masukan (*input*), serta data keluaran (*output*) yang memberikan hasil dari operasi perangkat lunak bagi pengguna.

3.3.1 Data Masukan

Data masukan adalah data yang akan diolah oleh sistem untuk mendapatkan hasil keluaran yang sudah ditentukan sistem. Pada aplikasi ini, data masukan berupa sinyal suara yang berasal dari *microphone* perangkat *iOS*. Dari sinyal suara yang diterima, aplikasi akan mendapatkan nilai frekuensi dan intensitas dari sinyal masukan tersebut. Aplikasi juga menerima data masukan berupa setelan *gain* untuk tiap *channel* frekuensi.

3.3.2 Data Proses

Data proses adalah data yang diproses ketika pengoperasian aplikasi. Data proses yang digunakan dalam proses ini dapat dilihat pada Tabel 3.1

Tabel 3.1: Data Proses

No	Nama Data	Tipe Data	Keterangan
1	input	AKNode	Sinyal suara input
2	output	AKNode	Sinyal suara output
3	frequency	float	Frekuensi sinyal
4	dBspl	float	Intensitas sinyal
5	ARbuff	float[]	Acoustic Reflex Buffer
6	cntrFreq	float[]	Nilai tengah frekuensi untuk tiap bandpass
7	bandwidth	float[]	Bandwidth dari tiap bandpass filter
8	chanGain	float[]	Nilai gain untuk tiap channel
9	ARthresh	float	Nilai threshold untuk Acoustic Reflex
10	NGthresh	float	Nilai threshold untuk Noise Gate
11	NGdelay	integer[]	Delay mekanisme *attack and release noise gate*

3.3.3 Data Keluaran

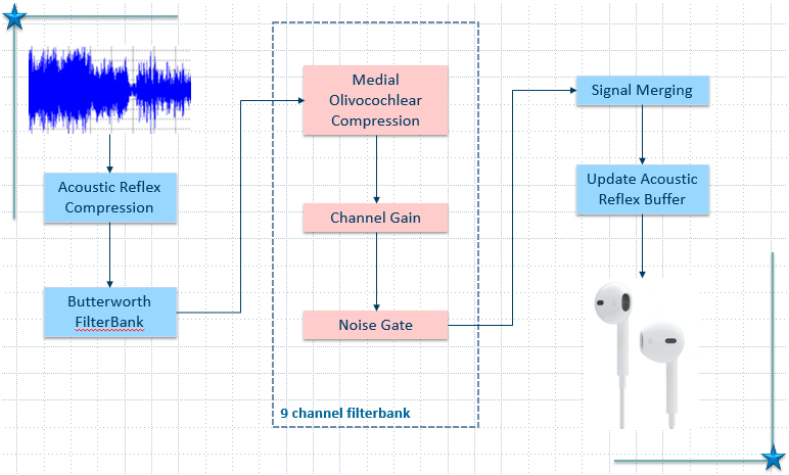
Data keluaran adalah data yang dihasilkan dari proses yang berjalan pada aplikasi. Pada aplikasi ini, data yang dihasilkan berupa sinyal suara yang telah melalui proses *acoustic reflex compression*, *butterworth filter*, *medial olivocochlear compression*, *channel gain*, dan *noise gate*. Sinyal suara tersebut dikeluarkan ke *earphone* pengguna aplikasi.

3.4 Metode Evaluasi

Evaluasi dilakukan untuk mengetahui performa aplikasi. Metode evaluasi yang dilakukan terdapat 2 jenis, yaitu pengujian secara fungsionalitas dan pengujian perbandingan aplikasi sejenis. Kedua jenis pengujian dilakukan dengan melakukan uji coba terhadap penderita gangguan pendengaran. Pada pengujian pertama, partisipan akan diminta untuk menggunakan sistem yang dibangun untuk mengetahui apakah sistem sudah dapat berfungsi untuk membantu para penderita gangguan pen- dengaran. Pada pengujian kedua, partisipan akan diberikan be- berapa jenis alat bantu dengar sejenis, dan diminta untuk memilih aplikasi mana yang lebih baik pada beberapa kategori yang telah ditentukan.

3.5 Diagram Alir Sistem Utama

Proses utama dalam aplikasi ini membutuhkan masukan berupa suara dari *microphone* beserta setelan yang sesuai dengan karakteristik pendengaran pengguna. Setelan tersebut dapat di- ubah oleh pengguna untuk menentukan konfigurasi yang paling baik bagi gangguan pendengaran pengguna. Sistem kemudian akan melakukan proses perbaikan suara sesuai dengan diagram alir pada gambar 3.1



Gambar 3.1: Diagram Alir Sistem

Pada gambar 3.1, dapat dilihat bahwa sinyal akan dipecah ke dalam 9 *channel* melalui *filter bank*. Filter-filter ini memiliki parameter *range* frekuensi yang berbeda-beda. Setelah itu, sinyal yang telah dipecah akan mengalami proses *medial olivocochlear compression*, *channel gain*, dan *noise gate* yang diimplementasikan pada tiap sinyal yang sudah dipecah. Setelah itu, baru sinyal akan digabung kembali dan siap untuk dikeluarkan.

3.5.1 Sinyal Input

Pertama, aplikasi akan menerima input dari *microphone* pengguna. Sinyal input yang diambil menggunakan *sample rate* sebesar 44100 Hertz. Frekuensi sampling ini dipilih karena berdasarkan *Nyquist-Shannon sampling theorem*, frekuensi ini harus diambil sebesar minimal 2 kali frekuensi maksimum yang akan diproduksi. Pendengaran manusia maksimum dapat mendengar suara sebesar 20000 Hz, sehingga *sampling rate* minimum adalah 40000 Hz. Selain itu, untuk mencegah *aliasing*, digu-

nakan *low-pass filter*. Filter yang cukup besar (2.05 kHz) dapat digunakan untuk melakukan *anti-aliasing* pada 44.1 kHz.

Nilai *sample rate* yang lebih besar dinilai kurang mem- berikan efek positif dikarenakan kompleksitas dalam melakukan pemrosesan sinyal juga akan meningkat. *Sample rate* yang lebih besar juga memunculkan *noise* yang lebih besar, sehingga kurang optimal dalam aplikasi ini. *Hardware* yang digunakan untuk melakukan proses sendiri adalah *smartphone*, yang mana tidak memiliki kapasitas seperti laptop atau PC. Selain itu, nilai 44.1 kHz sudah cukup bagi pendengaran manusia untuk mengerti isi percakapan.

Sinyal yang diambil dari *microphone* bersifat mono. Hal ini dikarenakan input hanya berasal dari 1 tempat, yaitu *microphone* pengguna, sehingga tidak memungkinkan untuk memproduksi sinyal stereo.

3.5.2 Acoustic Reflex

Acoustic Reflex adalah proses yang berusaha untuk meredam suara dengan intensitas terlalu besar. Pada diagram alir, dapat dilihat bahwa proses ini terbagi jadi 2, yaitu *Acoustic Reflex Compression* dan *Update Acoustic Reflex Buffer*. Proses ini akan dikalkulasi tiap 0.01 sekon dikarenakan kompleksitas yang menjadi terlalu besar jika dilakukan kalkulasi untuk tiap sample.

3.5.2.1 Acoustic Reflex Compression

Proses ini dilakukan terhadap sinyal input. Sinyal akan dimodifikasi sesuai dengan rumus berikut:

$$S = in/c \tag{3.1}$$

in merupakan sinyal input yang didapat dari *microphone*, dan *c* adalah konstanta *Acoustic Reflex Compression*. Konstanta ini diperoleh dari proses *update acoustic reflex buffer* yang di-

jalankan sebelum sinyal hasil dikeluarkan. Hasil dari proses ini merupakan sinyal suara yang telah melalui peredaman jika intensitas yang diterima sebelumnya terlalu besar yang dapat mengganggu kesehatan telinga [8].

3.5.2.2 Update Acoustic Reflex Buffer

Proses ini melakukan kalkuklasi nilai *buffer acoustic reflex* dan dilakukan di akhir sebelum sinyal dikeluarkan.

$$c = \frac{\sqrt{opf(in^2)}}{t}$$
$$c = max(1, c) \tag{3.2}$$

in merupakan sinyal yang dihasilkan dari proses sebelumnya. *t* merupakan *Acoustic Reflex Threshold*, yaitu threshold yang digunakan sebagai pembatas suara keras pada pendengaran. *opf* merupakan *one pole filter*, yang digunakan untuk memperhalus sinyal. Filter ini memiliki fungsi transfer sebagai berikut:

$$|H(z)| = \frac{b_0}{1 + a_1 * z^{-1}}$$
$$b_0 = 1/(sr * 0.06)$$
$$a_1 = (1/(sr * 0.06)) - 1 \tag{3.3}$$

Nilai konstanta c kemudian akan dimaksimalkan dengan angka 1. Hal ini ditujukan agar ketika intensitas suara di bawah threshold, tidak terjadi pengerasan suara (konstanta 1 berarti tidak ada perubahan intensitas).

Acoustic Reflex pada pendengaran manusia memiliki jeda waktu (0.1 sekon) sebelum dapat beradaptasi terhadap intensitas suara [7]. Dalam aplikasi ini, jeda waktu diimplementasikan dengan memiliki sebuah buffer yang berisi nilai konstanta *c* yang dihitung tiap 0.01 sekon. Dengan buffer ini, pada pada saat

proses kompresi, konstanta yang digunakan adalah konstanta yang dihitung pada 0.1 sekon sebelumnya. Proses ini tidak mengubah sinyal suara, hanya melakukan kalkulasi konstanta *acoustic reflex*.

3.5.3 Butterworth Filter Bank

Butterworth Filter Bank merupakan kumpulan dari beberapa *butterworth bandpass filter* yang memiliki parameter yang berbeda-beda. *Butterworth filter* yang digunakan menggunakan *order* 2. *Order* 2 ini dipilih karena kompleksitasnya yang tidak terlalu besar, dan cukup akurat karena gradien atenuasinya cukup besar [9]. Nilai order yang lebih tinggi akan me- nyebabkan komputasi menjadi lebih lama dan pada beberapa *device* dikhawatirkan aplikasi menjadi *laggy* (tidak *real-time*). Respons pada *butterworth order* 2 dapat dilihat pada gambar 2.6 yang diwakili dengan garis berwarna merah. Pada aplikasi ini, ter- dapat 9 filter yang akan digunakan pada *range* frekuensi yang berbeda dan tidak *overlapping*. Berikut adalah keterangan me- ngenai masing-masing filter:

Tabel 3.2: Data *Butterworth BandPass Filter*

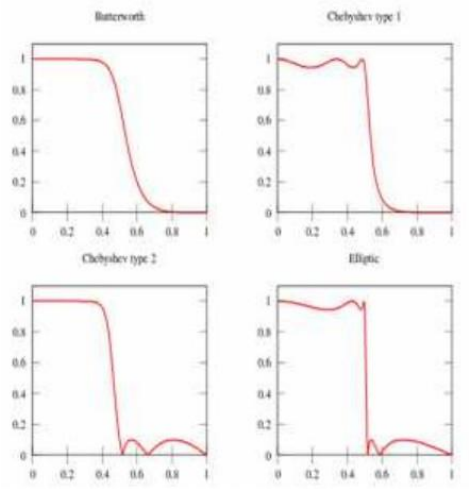
No	Center Frequency (Hertz)	Bandwidth (Centz)
1	250	600
2	500	600
3	1000	600
4	1414	300
5	2000	300
6	2828	300
7	4000	300
8	5656	300
9	8000	300

Konfigurasi frekuensi sesuai dengan keterangan di atas dapat merepresentasikan frekuensi yang diterima pada rumah siput pada telinga. Pembagian frekuensi tersebut diharapkan dapat mencakup kerusakan pada pendengaran manusia [12]. Meskipun demikian, konfigurasi tersebut bukanlah konfigurasi yang paling baik dan dapat dilakukan eksperimen terhadap konfigurasi filter untuk menghasilkan *filterbank* yang lebih baik. *Centz* merupakan satuan untuk *bandwidth* yang memiliki rumus sebagai berikut:

$$¢ = 1200 * log_2(f_2/f_1) \tag{3.4}$$

Pada rumus 3.4, dapat dilihat bahwa satuan *centz* dipilih untuk menentukan *bandwidth* tiap *butterworth filter*. *Centz* me- rupakan rasio terhadap *center frequency* untuk menentukan frekuensi passband. Satuan centz dipilih karena lebih repre- sentatif dalam menggambarkan *bandwidth* yang dibutuhkan untuk tiap filter. Hal ini dikarenakan *bandwidth* yang digunakan merupakan rasio terhadap *center frequency*, sehingga jika menggunakan satuan *Hertz* tidak sesuai.

Aplikasi ini menggunakan *butterworth filter* untuk membuat *filter bank* dikarenakan filter ini tidak memiliki *ripple* (respons berbentuk gelombang) sebelum dan sesudah *cutoff frequency* seperti yang terdapat pada jenis filter lainnya [11]. Gambar 3.2 menjelaskan perbandingan respons pada *butterworth filter* dan filter lainnya. Dapat dilihat bahwa respons pada *butterworth* lebih *smooth* dibandingkan dengan filter sejenis. Butterworth filterbank ini sendiri merupakan implementasi dari proses dispersi frekuensi yang terdapat pada *cochlea* atau rumah siput sebelum sinyal suara masuk ke otak.



Gambar 3.2: Perbandingan Respons Filter [11]

3.5.4 Medial Olivocochlear Compression

Medial Olivocochlear Compression merupakan proses yang terjadi di dalam *basiliar membrane* dengan melakukan kompresi terhadap suara di atas threshold. Proses ini merupakan proses pertama yang dilakukan pada tiap *channel* filter. Pada sistem yang dibangun, *threshold* yang dipilih adalah 80 desibel yang mana merupakan *threshold* rata-rata yang dimiliki oleh pendengar normal. Penderita gangguan biasanya tidak memiliki perbedaan *threshold* yang signifikan terhadap *threshold* pen- dengar normal [14]. Jika sinyal terdeteksi memiliki intensitas di atas 80 desibel, maka sistem akan memberikan nilai *gain* sebesar
0.2 desibel untuk tiap kenaikan 1 desibel pada sinyal tersebut. Nilai *threshold* dan nilai *gain* tidak mengalami perubahan pada *channel* yang berbeda. Proses reduksi suara tersebut diimple- mentasikan menggunakan formula di bawah ini:

$$S = in^g * t^{1-g} \tag{3.5}$$

Pada rumus tersebut, *in* merupakan sinyal input dari proses *medial olivocochlear compression*. Sinyal input ini merupakan sinyal hasil dari proses *filter bank*. *t* merupakan nilai threshold pada proses ini, yaitu 80 desibel. *g* merupakan nilai gain untuk tiap kenaikan sebesar 1 desibel jika intensitas sinyal berada di atas threshold. Pada aplikasi, nilai *g* yang dipilih adalah 0.2.

3.5.5 Channel Gain

Channel Gain merupakan penambahan intensitas suara yang dilakukan pada tiap *channel*. Input dari proses ini merupakan sinyal hasil dari proses *medial olivocochlear compression*. Penambahan intensitas akan disesuaikan dengan setelan dari pengguna pada tiap *channel*. Tiap *channel* memiliki nilai *gain* minimum sebesar 0 desibel (suara tidak dikeraskan sama sekali) hingga 50 desibel. Tujuan dari proses ini adalah agar pengguna dapat mendengar suara pada frekuensi-frekuensi yang mana pendengaran mereka terganggu.

3.5.6 Noise Gate

Noise gate yang digunakan dalam aplikasi ini memiliki *threshold* sebesar 50 db. Dengan demikian, sinyal dengan intensitas yang kurang dari 50 db akan dilemahkan karena sinyal tersebut akan dianggap sebagai *noise*, sedangkan sinyal dengan intensitas lebih dari 50 db akan diteruskan. Pemilihan nilai *threshold* ini berdasarkan eksperimen yang telah dilakukan berkali-kali pada berbagai macam lingkungan sekitar dan dipilih berdasarkan nilai *threshold* yang paling baik.

Selain itu, *noise gate* memiliki 2 parameter lainnya, yaitu *attack* dan *release time*, yaitu jeda waktu yang diberikan sebelum sinyal akan diteruskan atau diberhentikan. *Noise Gate* didesain

untuk memiliki waktu *attack* sebesar 0.01 sekon. Waktu *attack* yang sangat kecil ini diharapkan membuat penderita pen- dengaran dapat dengan segera mendengar suara dengan intensitas di atas *threshold*. Sebaliknya, nilai *release* akan diberi jeda lebih besar daripada nilai *attack*, yaitu 0.3 sekon. Pemberian nilai yang agak besar ini ditujukan agar *noise* tidak muncul dan berhenti secara terlalu mendadak. Hal ini juga membantu pen- dengar memahami suara setelah percakapan selesai [16].

BAB 4

IMPLEMENTASI

Bab ini menjelaskan tentang implementasi Tugas Akhir berdasarkan rancangan perangkat lunak. Proses implementasi mengacu pada rancangan perangkat yang telah dilakukan sebelumnya, namun juga dimungkinkan terjadinya per- ubahan-perubahan jika dirasa perlu. Implementasi dilakukan dalam bahasa *Swift*

4.1 Lingkungan Implementasi

Sub bab ini menjelaskan tentang lingkungan implementasi perangkat lunak yang dibangun. Lingkungan selama proses implementasi aplikasi *Hear Me* adalah sebagai berikut:

Tabel 4.1: Spesifikasi Lingkungan Implementasi

Perangkat Keras	Prosesor 2.5 GHz Intel Core i5 Memori 4 GB 1600 MHz DDR3 Grafik Intel HD Graphics 4000 1536 MB Apple Earphone
Perangkat Lunak	Sistem Operasi Macintosh El Capitan versi 10.11.6 XCode versi 8.2.1 AudioKit iOS versi 3.6

4.2 Implementasi Perubahan Intensitas Suara

Bab ini menjelaskan implementasi sebuah kelas bernama *AKBooster* yang berfungsi untuk meredam dan memperkeras intensitas suara. Kelas ini dapat menerima 2 jenis parameter, yaitu *dB* - penambahan intensitas suara dalam desibel, dan *gain* - penambahan intensitas suara dalam bentuk rasio terhadap intensitas suara sekarang. Untuk melakukan peredaman inten-

sitas, dapat dengan memberi parameter desibel bernilai negatif, atau parameter *gain*. Kelas ini memanfaatkan kode yang telah disediakan oleh *Swift* dan *AudioKit* untuk merubah intensitas sinyal.

```
class AKBooster: AKNode, AKToggleable, AKComponent {
    public AKAudioUnitType = AKBoosterAudioUnit

    private var internalAU: AKAudioUnitType?
    private var token: AUParameterObserverToken?

    private var gainParameter: AUParameter?
    private var lastKnownGain: Double = 1.0

    /// rasio gain
    open dynamic var gain: Double = 1 {
        willSet {
            if gain != newValue {
                if internalAU?.isSetUp() ?? false {
                    if let existingToken = token {
                        gainParameter?.setValue
                        (Float(newValue),
                        originator: existingToken)
                    }
                } else {
                    internalAU?.gain = Float(newValue)
                }
            }
        }
    }

    /// desibel
    open dynamic var dB: Double {
        set {
            gain = pow(10.0, Double(newValue / 20))
        }
        get {
            return 20.0 * log10(gain)
        }
    }
```

```
36      public init(
37      _ input: AKNode?,
38      gain: Double = 1) {
39
40          self.gain = gain
41          _Self.register()
42          super.init()
43
44          AVAudioUnit._instantiate(with:
45          _Self.ComponentDescription)
46          { [weak self] avAudioUnit in
47
48              self?.avAudioNode = avAudioUnit
49              self?.internalAU = avAudioUnit.auAudioUnit
50
51              input?.addConnectionPoint(self!)
52          }
53
54          gainParameter = tree["gain"]
55
56          token = tree.token (byAddingParameterObserver:
57          { [weak self] address, value in
58
59              DispatchQueue.main.async {
60                  if address == self?.gainParameter?.
61                  address {
62                      self?.gain = Double(value)
63                  }
64              }
65          })
66          internalAU?.gain = Float(gain)
67      }
68
69      open func start() {
70          if isStopped {
71              gain = lastKnownGain
72          }
73      }}
```

Kode Sumber 4.1: Implementasi AKBooster

4.3 Implementasi *Acoustic Reflex*

Pada bab ini akan dibahas mengenai implementasi *Acoustic Reflex Compression* yang melakukan reduksi intensitas terhadap sinyal suara yang terlalu keras.

4.3.1 Implementasi *Acoustic Reflex Compression*

Subbab ini akan menjelaskan implementasi dari kompresi *acoustic reflex* dengan meredam sinyal sekarang sesuai dengan nilai *acoustic reflex constant* yang telah dikomputasi dan disimpan di dalam buffer.

```
1  var ARconst: Double = ARbuff.dequeue()
2  var ARcompress: AKBooster!
3
4  ARcompress = AKBooster(input, gain: ARconst)
```

Kode Sumber 4.2: Proses *Acoustic Reflex Compression*

4.3.2 Implementasi *Update Acoustic Reflex Buffer*

Subbab ini akan membahas mengenai kalkulasi konstanta *Acoustic Reflex* yang akan dibuat untuk meredam suara yang terlalu keras. Nilai tersebut akan dimasukkan ke dalam buffer untuk dapat digunakan sesuai dengan delay yang diinginkan.

```
1  var curr_dB: Double = amptracker.amplitude
2  curr_dB = power2db(power: curr_dB)
3
4  var ARconst: Double = opf(pow(curr_dB, 2))
5  ARconst = ARconst.squareRoot() / ARthresh
6  ARconst = max(ARconst, 1)
7
8  ARbuff.enqueue(ARconst)
```

Kode Sumber 4.3: Proses *update Acoustic Reflex Buffer*

4.4 Implementasi *Butterworth Filter Bank*

Pada bab ini akan dibahas mengenai implementasi *Butterworth Filter Bank* yang bertugas untuk memecah frekuensi sinyal.

4.4.1 Implementasi *Butterworth Filter*

Subbab ini akan menjelaskan implementasi kelas untuk melakukan pemrosesan *butterworth filter*.

```
1  class AKBandPassFilter: AKNode, AKComponent {
2      public typealias AKAudioUnitType =
3      AKBandPassButterworthFilterAudioUnit
4
5      private var internalAU: AKAudioUnitType?
6      private var token: AUParameterObserverToken?
7
8      private var centerFrequencyParameter: AUParameter?
9      private var bandwidthParameter: AUParameter?
10
11     /// Center frequency. (in Hertz)
12     dynamic var centerFrequency: Double = 2_000.0 {
13         willSet {
14             if centerFrequency != newValue {
15                 if internalAU?.isSetUp() ?? false {
16                     if let existingToken = token {
17                         centerFrequencyParameter?.
18                         setValue(Float(newValue),
19                         originator: existingToken)
20                     }
21                 } else {
22                     internalAU?.centerFrequency =
23                     Float(newValue)
24                 }
25             }
26         }
27     }
28     /// Bandwidth. (in Centz)
```

```
29      open dynamic var bandwidth: Double = 100.0 {
30          willSet {
31              if bandwidth != newValue {
32                  if internalAU?.isSetUp() ?? false {
33                      if let existingToken = token {
34                          bandwidthParameter?.setValue
35                          (Float(newValue),
36                          originator: existingToken)
37                      }
38                  } else {
39                      internalAU?.bandwidth =
40                      Float(newValue)
41                  }
42              }
43          }
44      }
45
46      public init(
47      _ input:  AKNode?,
48      centerFrequency: Double = 2_000.0,
49      bandwidth: Double = 100.0) {
50
51          self.centerFrequency = centerFrequency
52          self.bandwidth = bandwidth
53          _Self.register()
54          super.init()
55
56          AVAudioUnit._instantiate(with:
57          _Self.ComponentDescription)
58          { [weak self] avAudioUnit in
59
60              self?.avAudioNode = avAudioUnit
61              self?.internalAU = avAudioUnit.
62              auAudioUnit
63
64              input?.addConnectionPoint(self!)
65          }
66
67          guard let tree = internalAU?.parameterTree
68          else {
```

```
69                 return
70             }
71
72             ctrFreqParameter = tree["centerFrequency"]
73             bandParameter = tree["bandwidth"]
74
75             token = tree.token (byAddingParameterObserver:
76             { [weak self] address, value in
77
78                 DispatchQueue.main.async {
79                     if address == self?.ctrFreqParameter?.
80                     address {
81                         self?.ctrFrequency = Double(value)
82                     }
83                     else if address == self?.bandParameter? .
84                     address {
85                         self?.bandwidth = Double(value)
86                     }
87                 }
88             })
89
90             internalAU?.centerFrequency =
91             Float(ctrFrequency)
92             internalAU?.bandwidth = Float(bandwidth)
93         }
94
95         open func start() {
96             internalAU?.start()
97         }
98     }
```

Kode Sumber 4.4: Implementasi *Butterworth Filter*

4.4.2 Implementasi *Filter Bank*

Subbab ini akan membahas mengenai implementasi filter bank dengan memanfaatkan kelas *butterworth filter* yang telah dibuat.

```
var filter = [AKBandPassFilter]()
for i in 1...nBands {
    filter.append(AKBandPassFilter(ARcompress))
}

filter[0].centerFrequency = 250
filter[0].bandwidth = 600

filter[1].centerFrequency = 500
filter[1].bandwidth = 600

filter[2].centerFrequency = 1000
filter[2].bandwidth = 600

filter[3].centerFrequency = 1414
filter[3].bandwidth = 300

filter[4].centerFrequency = 2000
filter[4].bandwidth = 300

filter[5].centerFrequency = 2828
filter[5].bandwidth = 300

filter[6].centerFrequency = 4000
filter[6].bandwidth = 300

filter[7].centerFrequency = 5656
filter[7].bandwidth = 300

filter[8].centerFrequency = 8000
filter[8].bandwidth = 300

/// Mix the signal together
var mixer = AKMixer()
for currBand in 1...nBands {
    mixer.connect(noise_gate[currBand - 1])
}
```

Kode Sumber 4.5: Proses membuat *filter bank*

4.5 Implementasi *Medial Olivocochlear Compression*

Pada bab ini dibahas mengenai implementasi *medial olivocochlear compression*, yaitu reduksi suara pada intensitas di atas *medial olivocochlear threshold*. Proses ini akan meneruskan intensitas sebesar 0.2 dB untuk tiap kenaikan 1 dB. Perubahan nilai akan dilakukan setiap 0.01 sekon agar pemrosesan yang terjadi di dalam *smartphone* tidak terlalu berat.

```
1  for i in 1...nBands {
2      MOC.append(AKBooster(filter[i - 1]))
3
4      if amp_now > MOCt {
5          amp_now = power(amp_now, MOCg) *
6          power(MOCt, (1-MOCg))
7
8          var tmp_gain: Double = amp_now / curr_amp
9          MOC[i - 1].gain = tmp_gain
10     }
11 }
```

Kode Sumber 4.6: Implementasi *Medial Olivocochlear Compression*

4.6 Implementasi *Channel Gain*

Pada bab ini akan dibahas mengenai implementasi *channel gain*, yaitu pengerasan suara yang dilakukan pada sinyal pada tiap *filter*. Besar tidaknya nilai *gain* pada tiap *channel* dipe- ngaruhi oleh setelan pengguna. Tiap *channel* akan memiliki nilai *gain* sebesar 20 desibel pada saat aplikasi pertama kali dibuka. Pengguna dapat mengatur nilai tersebut dengan nilai minimal sebesar 0 desibel dan nilai maksimal sebesar 50 desibel. Pengerasan suara menggunakan bantuan AKBooster yang terdapat pada bab 4.2

```
var chan_gain = [AKBooster]()

for i in 1...nBands {

    gain_dB.append(20)
    chan_gain.append(AKBooster(MOC[i - 1]))
    chan_gain[i - 1].dB = gain_dB[i - 1]
}

stackView.axis = .vertical
stackView.distribution = .fillEqually
stackView.alignment = .fill
stackView.translatesAutoresizingMaskIntoConstraints =
false
stackView.spacing = 3

for i in 1...nBands {
    stackView.addArrangedSubview(AKPropertySlider(
        property: "Gain " + i,
        format: "%0.2f",
        value: self.gain_dB[i - 1],
        minimum: 0, maximum: 50,
        color: UIColor(red:1 - (0.1 * (i-1)), green:1,
         blue:0) ) {

        sliderValue in self.gain_dB[0] = sliderValue
    })
}
```

Kode Sumber 4.7: Implementasi *Channel Gain*

4.7 Implementasi *Noise Gate*

Pada bab ini akan dibahas mengenai implementasi *noise gate* pada aplikasi. Proses ini berfungsi untuk meredam suara jika hanya terdapat *noise* sehingga pendengar tidak perlu men- dengarkan *noise*. *Noise gate* yang diimplementasikan tidak mengalami penurunan intensitas suara pada saat sedang terjadi

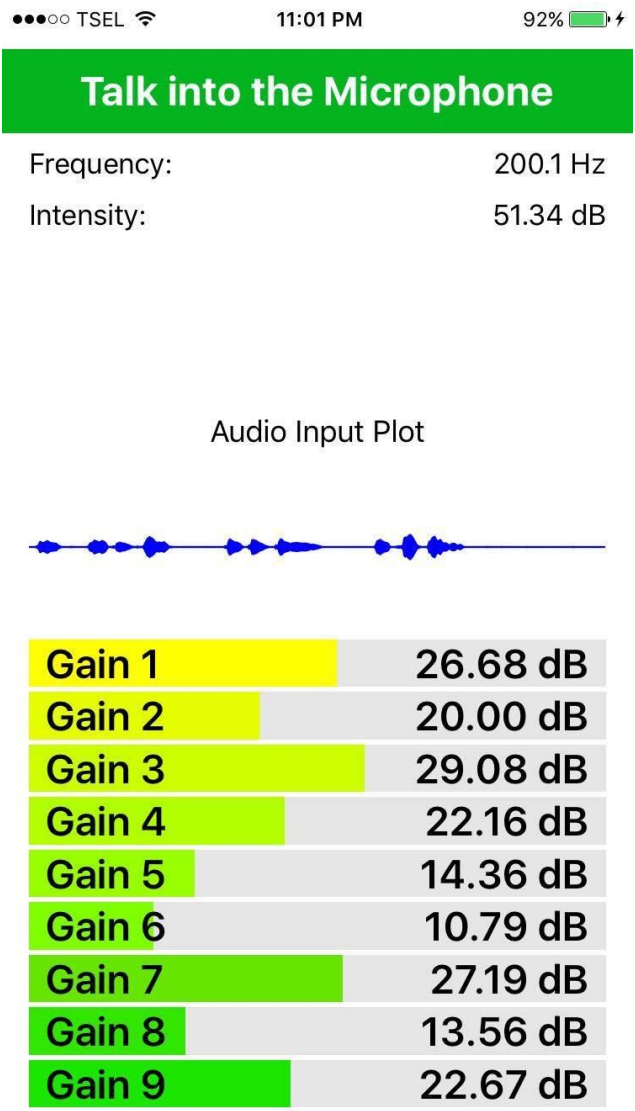
mekanisme *release*. Implementasi modul ini masih menggunakan bantuan kelas AKBooster

```
var noise_gate = [AKBooster] ()
var NGdelay = [Integer]()

for currBand in 1...nBands{
    noise_gate.append(AKBooster(chan_gain[currBand-1],
     gain:1))

    var curr_dB: Double = chan_gain[currBand-1].
    amplitude
    curr_dB = power2db(power: curr_dB)

    if curr_dB < NGthresh {
        NGdelay[currBand-1] = delay[currBand-1] - 1
        if delay[currBand-1] == 0 {
            noise_gate[currBand-1].gain = 0
        }
    } else {
        NGdelay[currBand - 1] = 30
        noise_gate[currBand-1].dB = gain_dB[currBand-1]
    }
}
```

Kode Sumber 4.8: Implementasi *Noise Gate*

4.8 Implementasi Antarmuka Aplikasi

Aplikasi ini hanya memiliki 1 halaman, yaitu halaman utama yang berisi informasi mengenai frekuensi, amplitudo, dan intensitas suara yang masuk ke dalam microphone pengguna. Dalam halaman utama ini, juga terdapat setelan *gain* untuk tiap *channel* yang dapat diubah-ubah oleh pengguna. Nilai *gain* minimum adalah 0 desibel, yang berarti tidak terdapat pe- ngerasan suara sama sekali, dan nilai maksimumnya adalah 50 desibel. Tampilan antarmuka halaman utama aplikasi tersebut dapat dilihat pada gambar 4.1.



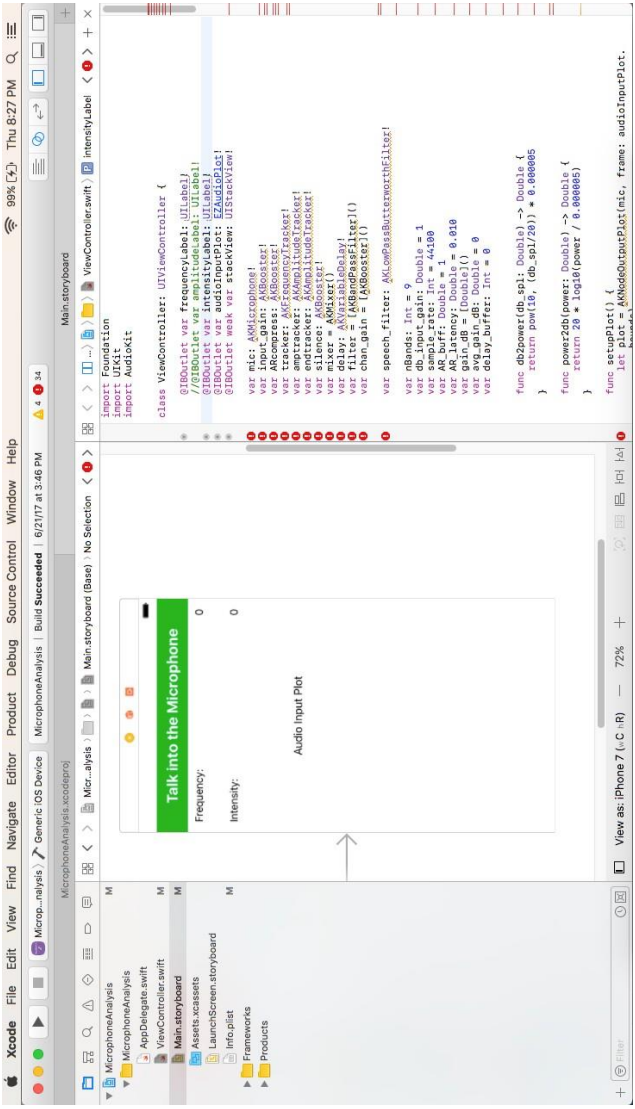
Gambar 4.1: Antarmuka Halaman Utama Aplikasi

4.9 Implementasi Deployment Aplikasi

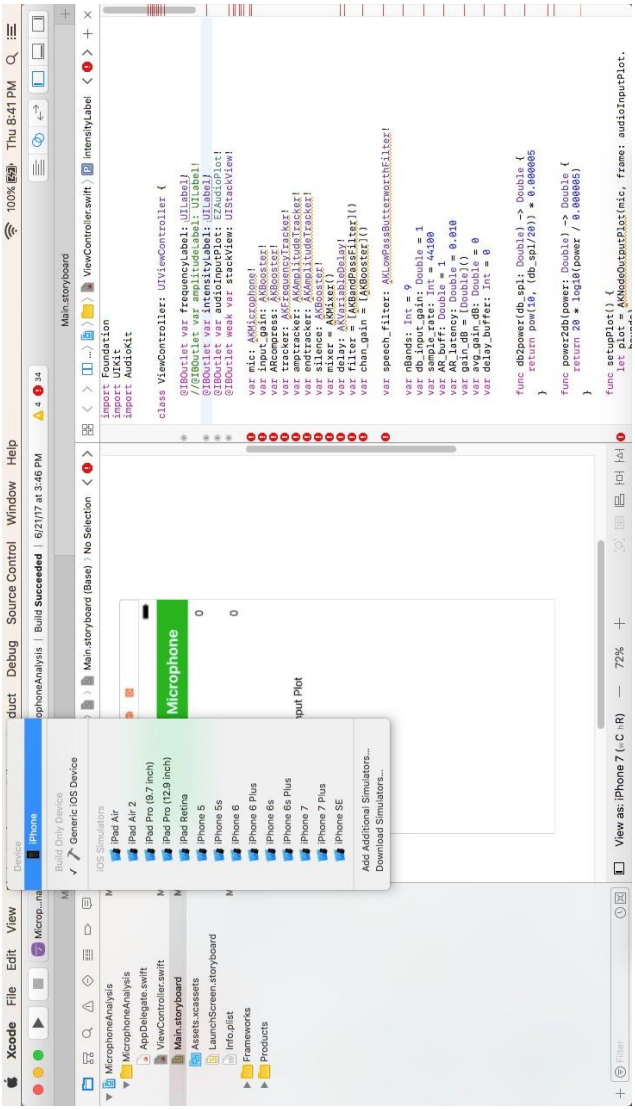
Subbab ini akan membahas cara untuk melakukan deployment aplikasi menggunakan bahasa *Swift* dengan *IDE Xcode* menuju smartphone berbasis *iOS* yang diinginkan. Gambar 4.2 menunjukkan tampilan antarmuka XCode yang digunakan untuk melakukan *development* aplikasi berbasis *iOS*. *Panel* yang terdapat di paling kiri merupakan struktur projek aplikasi yang sedang dibuat. Pada *panel* yang di tengah me- rupakan antarmuka aplikasi yang diinginkan. Proses edit antar- muka dapat dilakukan pada bagian ini. Setelah itu, *panel* paling kanan merupakan area untuk kode program. Pada bagian ini, setiap elemen antarmuka aplikasi dapat dihubungkan ke dalam kode untuk diberikan logika pemrograman yang diinginkan.

Untuk melakukan uji coba *deployment* aplikasi ke *smart- phone* yang diinginkan, *smartphone* harus terlebih dahulu dihubungkan ke *PC* atau *laptop* menggunakan kabel USB. Setelah *smartphone* tertancap pada *laptop*, maka akan muncul pilihan *smartphone* yang digunakan di kotak pilihan bagian atas seperti pada gambar 4.3 (pada gambar digunakan *iPhone* sebagai *smartphone* yang akan digunakan).

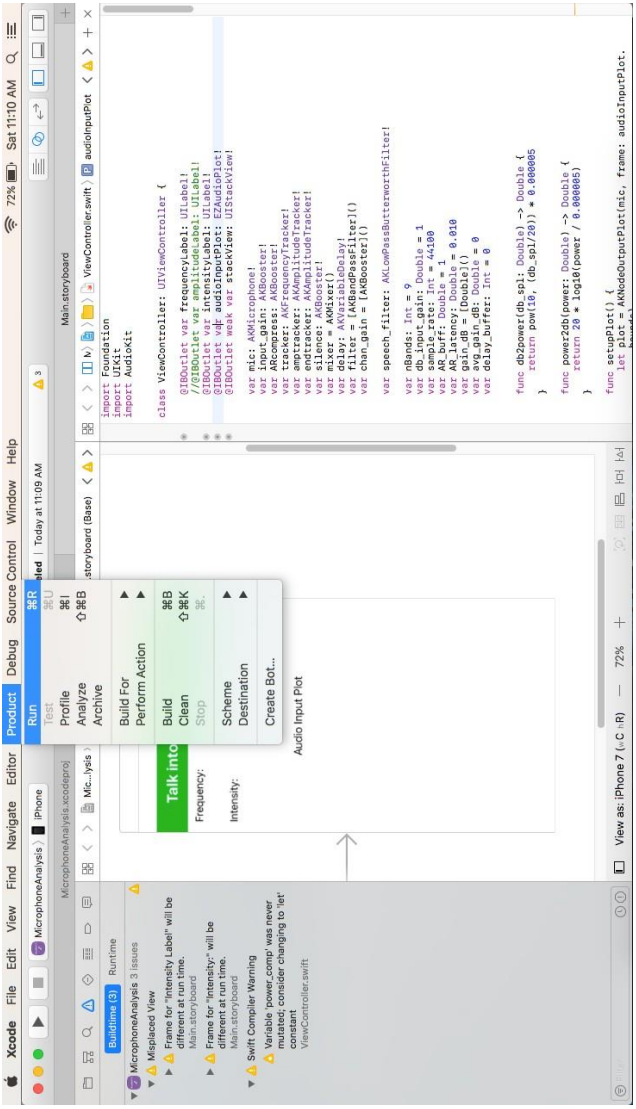
Setelah selesai memilih target *deployment*, aplikasi sudah siap untuk dijalankan pada perangkat yang dipilih. Pastikan bahwa perangkat yang dipilih sudah dalam keadaan *unlocked*, karena jika perangkat masih terkunci aplikasi tidak dapat di-*install* pada perangkat tersebut. Untuk menjalankan aplikasi, pilih menu *Product* pada *XCode*, lalu pilih *Run* seperti nampak pada gambar 4.4. Aplikasi juga dapat dijalankan dengan me- nekan tombol *Run* yang terletak di sebelah kiri menu pilihan target *deployment*. *XCode* akan secara otomatis melakukan instalasi aplikasi pada perangkat yang dipilih, dan aplikasi dapat dijalankan pada aplikasi tersebut.



Gambar 4.2: Antarmuka IDE XCode



Gambar 4.3: Pilihan Target Deployment



Gambar 4.4: Cara Menjalankan Aplikasi

BAB 5

PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada aplikasi yang dikembangkan. Pengujian yang dilakukan terbagi menjadi 2, yaitu pengujian terhadap kebutuhan fungsionalitas sistem dan kegunaan sistem, serta pengujian perbandingan terhadap produk lain yang sejenis. Pengujian fungsionalitas dan kegunaan pro- gram dilakukan dengan mengetahui tanggapan dari pengguna terhadap sistem. Pengujian perbandingan sistem dilakukan de- ngan membandingkan aplikasi yang dibangun dengan aplikasi alat bantu dengar yang juga bekerja dalam sistem operasi *iOS*. Hasil evaluasi menjabarkan tentang rangkuman hasil pengujian pada bagian akhir bab ini.

5.1 Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan Tugas Akhir ini dilakukan pada lingkungan dan alat kakas seperti yang tertera pada Tabel 5.1.

Tabel 5.1: Spesifikasi Pengujian Sistem

Perangkat Keras	Device: iPhone6 64 GB Prosesor: Dual Core 1.4 GHz Typhoon Memori: 1 GB DDR3 Apple Earphone
Perangkat Lunak	Sistem Operasi: iOS 10.0.2

5.2 Skenario Uji Coba Fungsionalitas

Uji coba ini melibatkan beberapa pengguna yang mengalami gangguan pendengaran. Skenario uji coba yang akan dilakukan adalah dengan pengguna diberikan *smartphone* beserta *earphone*. Pengguna kemudian diminta untuk menjalankan apli-

kasi *HearMe* dan mendengarkan suara melalui *earphone* yang sudah disediakan. Setelah itu, pengguna akan melakukan setelan pada aplikasi sesuai dengan karakteristik gangguan pendengaran masing-masing dan kenyamanan masing-masing pengguna. Setelah melakukan setelan pada aplikasi, penulis akan mengatakan sebuah kata pada jarak 1 meter. Pendengar akan diminta untuk menuliskan atau mengatakan kembali kata yang telah diucapkan oleh penulis. Pengujian seperti ini mirip dengan pengujian yang dilakukan di lab pendengaran untuk menguji kemampuan pendengaran penderita. Pengujian ini dilakukan sebanyak dua kali, pertama di lingkungan yang sepi (tidak terdapat *background noise*) dan yang kedua di lingkungan yang ramai (terdapat *noise*). Simulasi lingkungan yang ramai dila- kukan dengan menyalakan televisi ataupun kipas angin yang menghasilkan suara yang cukup keras. Kata yang dipilih me- rupakan kata yang hanya memiliki 1-2 suku kata. Hal ini disebabkan otak manusia mampu untuk mengkira-kira sebuah kata jika diberikan konteks yang sesuai. Jika diberikan kata dengan suku kata yang sedikit, diharapkan pengujian ini benar-benar mengecek kemampuan pengguna untuk mendengar melalui aplikasi *HearMe*.

Penulis kemudian akan melakukan tanya jawab seputar penggunaan alat bantu dengar berbasis *smartphone* tersebut untuk mendapatkan *feedback* mengenai kelebihan dan kekurangan aplikasi. Pada sesi ini, pengguna juga akan tetap menggunakan aplikasi untuk merasakan efek dari aplikasi terhadap pembicaraan normal.

5.3 Hasil Uji Coba Fungsionalitas

Uji coba telah dilakukan dengan melibatkan tujuh pengguna yang terdiri dari dua laki-laki dan lima perempuan. Uji coba ini dilakukan untuk mengetahui bagaimana kepuasan pengguna

terhadap metode perbaikan kualitas suara untuk membantu penderita gangguan pendengaran. Tabel 5.2 menjelaskan data tiap pengguna yang telah mencoba aplikasi tersebut.

Tabel 5.2: Data Pengguna Aplikasi

No. Partisipan	Jenis Kelamin	Usia (tahun)	Lama Pemakaian (tahun)
1	Laki-Laki	22	4
2	Perempuan	89	9
3	Perempuan	25	2
4	Perempuan	57	-
5	Perempuan	33	3
6	Laki-Laki	38	4
7	Perempuan	23	1

Pengujian dilakukan dengan penulis mengucapkan 10 kata satu-persatu dan meminta pengguna untuk mengucapkan kembali kata tersebut. Kata-kata yang dipilih adalah sebagai berikut: *cat, hands, bells, king, car, tree, book, chair, dog, leg*. Kata-kata ini dipilih dikarenakan jumlah suku kata yang sedikit (hanya terdapat 1 suku kata), dan pelafalannya mirip-mirip, sehingga pengujian dapat memberikan gambaran yang lebih baik mengenai apakah pengguna dapat mendengar dengan baik atau tidak. Hasil uji coba terhadap pengguna dapat dilihat pada tabel 5.3

Akurasi lingkungan sepi merupakan akurasi pendengaran partisipan pada lingkungan yang sepi. Sedangkan akurasi lingkungan ramai merupakan akurasi pendengaran partisipan pada lingkungan yang diberikan background noise (suara pem- bicaraan, suara TV, suara mesin, dll). Masing-masing kata diucapkan sebanyak sekali, sehingga terdapat 10 kali uji coba pada lingkungan sepi dan ramai

Tabel 5.3: Data Pengguna Aplikasi

No. Partisipan	Akurasi Lingkungan Sepi	Akurasi Lingkungan Ramai
1	100%	100%
2	50%	60%
3	90%	90%
4	70%	70%
5	100%	80%
6	80%	60%
7	80%	90%

5.4 Skenario Pengujian Perbandingan Aplikasi

Aplikasi yang akan digunakan sebagai perbandingan adalah *i-Hear* dan *Petralex*. Kedua aplikasi merupakan aplikasi berbasis *iOS* yang juga berfungsi untuk membantu penderita gangguan pendengaran. Kemudian, partisipan akan diberikan ketiga ma- cam aplikasi (beserta dengan sistem *HearMe* yang telah dibangun). Partisipan kemudian diminta untuk memberikan pe- nilaian terhadap beberapa kriteria penilaian, yaitu kejernihan suara, fleksibilitas sistem, kemudahan penggunaan, dan aplikasi yang lebih disukai. Partisipan diminta untuk memilih salah satu dari ketiga aplikasi yang memiliki performa paling baik untuk tiap kriteria penilaian.

5.5 Hasil Pengujian Perbandingan Aplikasi

Pengujian perbandingan aplikasi dilakukan terhadap 4 kategori sistem, yaitu kejernihan suara, fleksibilitas sistem, kemudahan penggunaan, dan aplikasi yang lebih disukai secara keseluruhan. Tabel 5.4 menunjukkan berapa partisipan yang paling menyukai sebuah aplikasi dalam keempat kategori.

Tabel 5.4: Hasil Perbandingan Aplikasi

Kategori	HearMe	i-Hear	Petralex
Kejernihan Suara	5	1	1
Fleksibilitas Sistem	7	0	0
Kemudahan Penggunaan	2	2	3
Keseluruhan Sistem	5	0	2

Dari hasil pengujian perbandingan, banyak partisipan menilai suara yang dihasilkan *HearMe* lebih jernih pada saat tidak terdapat pembicaraan. *HearMe* juga lebih fleksibel dikarenakan terdapat setelan untuk mengubah nilai *gain* untuk tiap *channel*. Untuk kemudahan penggunaan, partisipan lebih menyukai *Petralex* dikarenakan terdapat test pada awal aplikasi dan profil pendengaran dibuat berdasarkan hasil test. Namun, beberapa partisipan tetap lebih menyukai *HearMe* dikarenakan test tersebut memakan waktu sehingga kurang optimal. Secara ke- seluruhan, partisipan tetap lebih menyukai aplikasi *HearMe*.

5.6 Evaluasi Pengujian

Berdasarkan hasil evaluasi fungsionalitas, sistem yang dibangun telah berhasil untuk membantu penderita gangguan pendengaran untuk dapat mendengarkan dalam lingkungan yang sepi dan ramai. Terdapat sebuah pengecualian terhadap par- tisipan kedua, dikarenakan partisipan tersebut tidak mengerti bahasa inggris dan sudah tua (umurnya 89 tahun). Selain par- tisipan kedua, akurasi pendengaran cukup memuaskan. Keadaan lingkungan tidak memiliki efek yang signifikan terhadap akurasi pendengaran dari para partisipan. Hal ini membuktikan bahwa keadaan lingkungan sekitar tidak terlalu mempengaruhi kemampuan pendengaran partisipan menggunakan sistem yang telah dibangun. Dengan demikian, aplikasi ini dapat digunakan dalam keadaan sepi ataupun ramai.

Dalam proses pengujian, terdapat beberapa kelebihan dan kekurangan yang sempat disampaikan oleh para partisipan. Kelebihan yang utama adalah kemampuan sistem untuk membantu penderita gangguan tanpa adanya suara mendengung seperti yang diproduksi oleh alat bantu dengar. Alat bantu dengar mengeluarkan suara mendengung dengan intensitas cukup tinggi sehingga para penderita harus menyesuaikan pendengarannya terlebih dahulu saat pertama kali menggunakan aplikasi. Namun, pada sistem yang dibangun terdapat suara gemerisik yang dihasilkan oleh microphone. Suara ini juga mengganggu pendengaran, tetapi intensitasnya tidak sebesar suara dengung yang dihasilkan alat bantu dengar. Kekurangan lain adalah suara yang dihasilkan terdengar seperti suara *speaker* atau robot, sehingga tidak terdengar natural seperti suara manusia. Selain itu, penderita juga tidak dapat mengenali asal suara selama menggunakan aplikasi yang dibangun. Alat bantu dengar memiliki fitur untuk mengenali asal suara, misalnya jika ada suara yang berasal dari kanan penderita, maka suara keluaran alat bantu dengar sebelah kanan akan lebih keras dari yang kiri. Hal ini menyebabkan penderita mengerti bahwa suara berasal dari kanan. Fitur ini belum dimiliki oleh sistem yang dibangun.

Hasil pengujian perbandingan aplikasi juga menunjukkan bahwa partisipan rata-rata lebih menyukai sistem *HearMe*. Hal ini disebabkan oleh beberapa faktor, antara lain keras suara yang dapat disesuaikan oleh pengguna dan jernihnya suara. Kejer- nihan suara sistem dianggap lebih unggul dibandingkan aplikasi sejenis dikarenakan terdapat peredaman suara pada saat tidak terdapat sinyal terdeteksi. Peredaman suara tersebut sangat membantu untuk mengurangi suara gemerisik yang dihasilkan sistem saat tidak terdapat suara keras. Salah satu faktor yang dianggap kurang dari sistem yang dibangun adalah kemudahan penggunaan. Beberapa partisipan mengaku bahwa aplikasi *Petralex* lebih unggul dikarenakan terdapat tes pendengaran

terlebih dahulu sebelum dapat menggunakan aplikasi. Dari hasil tes pendengaran tersebut, profil pendengaran akan disimpan secara otomatis dan setelan aplikasi akan diatur sesuai dengan profil tersebut. Sebaliknya, ada beberapa partisipan yang me- ngaku bahwa tes tersebut lebih ribet, sehingga lebih mudah menggunakan sistem yang dibangun.

(Halaman ini sengaja dikosongkan)

BAB 6

KESIMPULAN DAN SARAN

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

6.1 Kesimpulan

Dari hasil selama proses perancangan, implementasi, serta pengujian dapat diambil kesimpulan sebagai berikut:

1. Sistem yang dibangun mampu membantu penderita gangguan pendengaran untuk dapat mendengarkan layaknya pendengaran normal dibuktikan dengan akurasi pendengaran rata-rata partisipan mencapai 79.3
2. Kondisi lingkungan sekitar tidak berpengaruh secara signifikan terhadap kemampuan penderita menggunakan sistem yang dibangun. Terbukti dengan rata-rata selisih akurasi lingkungan sepi dengan lingkungan ramai hanya 4.29
3. Secara keseluruhan, sistem yang dibangun lebih disukai dibandingkan dengan sistem sejenis yang ada dikarenakan 5 dari 7 partisipan lebih menyukai sistem HearMe

6.2 Saran

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

- Terdapat kebutuhan aplikasi untuk dapat memodifikasi suara yang diterima melalui panggilan masuk dari *smartphone*, ataupun memodifikasi lagu yang sedang diputar oleh pengguna agar penderita tetap dapat berbicara menggunakan *smartphone*nya ketika sedang ditelpon.

- Suara yang diterima oleh *microphone* memiliki kualitas yang tidak terlalu baik, sehingga pengguna merasa sedang mendengarkan *speaker* (kurang natural seperti suara manusia).
- Pengguna tidak dapat mengetahui asal suara seperti pada alat bantu dengar dikarenakan suara yang masuk ke dalam aplikasi bersifat mono, bukan stereo.
- Aplikasi hanya dapat berjalan dengan baik ketika *microphone* telah ditancapkan ke dalam *smarphone*. Jika tidak, maka suara yang diterima akan mendengung karena suara *output* dari aplikasi akan masuk lagi sebagai *input*. Hal ini dapat diatasi dengan menambahkan modul *feedback cancellation*.

DAFTAR PUSTAKA

[1] "Tennessee science standards," Apr. 2017.

[2] J. M. Lasak, P. Allen, T. McVay, and D. Lewis, "Hearing loss: diagnosis and management," vol. 41, pp. 19–31, Elsevier, 2014.

[3] H. Dillon, A. James, J. Ginis, *et al.*, "Client oriented scale of improvement (cosi) and its relationship to several other measures of benefit and satisfaction provided by hearing aids," vol. 8, pp. 27–43, 1997.

[4] "Who deafness and hearing loss," Apr. 2017.

[5] "Tympanogram an introduction," May 2017.

[6] "Acoustic reflex - introduction," May 2017.

[7] W. Niemeyer, "Relations between the discomfort level and the reflex threshold of the middle ear muscles," vol. 10, pp. 172–176, Taylor & Francis, 1971.

[8] T. Brask, "The noise protection effect of the stapedius reflex," vol. 86, pp. 116–117, Taylor & Francis, 1978.

[9] S. Butterworth, "On the theory of filter amplifiers," vol. 7, pp. 536–541, 1930.

[10] W. M. Laghari, M. U. Baloch, M. A. Mengal, and S. J. Shah, "Performance analysis of analog butterworth low pass filter as compared to chebyshev type-i filter, chebyshev type-ii filter and elliptical filter," 2014.

[11] L. Himanshu and K. Wason, "Butterworth filter," vol. 1, pp. 1612–1614, 2014.

[12] R. Meddis, W. Lecluyse, C. M. Tan, M. R. Panda, and R. Ferry, "Beyond the audiogram: Identifying and modeling patterns of hearing deficits," in *The neurophysiological bases of auditory perception*, pp. 631–640, Springer, 2010.

[13] R. Port, “Audition for linguists,” May 2017.

[14] S. Kortlang, G. Grimm, V. Hohmann, B. Kollmeier, and S. D. Ewert, “Auditory model-based dynamic compression controlled by subband instantaneous frequency and speech presence probability estimates,” vol. 24, pp. 1759–1772, IEEE, 2016.

[15] T. Jurgens and et al, “Hearing aid fitting using individual computer models,” 2013.

[16] Y. Yun, “Noise reduction & gate plug-ins in audio mixing process,” vol. 9, pp. 49–56, Citeseer, 2014.

[17] “The swift programming language - apple developer,” 2017.

BIODATA PENULIS

Arianto Wibowo, lahir di Surabaya tanggal 10 Juli 1995. Penulis merupakan anak pertama dari 2 bersaudara. Penulis telah menempuh pendidikan formal TK St. Maria Surabaya, SD St. Maria Surabaya (2001-2007), SMP St. Maria Surabaya (2007-2010), dan SMA St. Louis 1 Surabaya (2010-2013). Penulis melanjutkan studi kuliah program sarjana di Jurusan Teknik Informatika ITS. Selama masa kuliah, penulis aktif dalam membantu kegiatan Pelatihan Nasional untuk Tim Olimpiade Komputer Indonesia. Penulis juga aktif menjadi problem setter pada acara National Programming Contest yang diselenggarakan oleh ITS dan Agricode yang diselenggarakan oleh IPB.

Selama kuliah di Teknik Informatika ITS, penulis mengambil bidang minat Komputasi Cerdas dan Visualisasi (KCV). Penulis pernah menjadi asisten dosen dan praktikum untuk mata kuliah Dasar Pemrograman, Struktur Data, Aljabar Linear, Perancangan dan Analisis Algoritma 1, Perancangan dan Analisis Algoritma 2, serta Pengolahan Citra Digital. Selama menempuh perkuliahan, penulis juga aktif mengikuti kompetisi dan berhasil menjadi juara 2 COMPFEST UI (2014), juara 1 GEMASTIK Data Mining (2016), juara 1 Bukalapak Programming Contest (2016), dan beberapa lomba pemrograman lainnya tingkat nasional. Penulis juga selalu menjadi finalis ICPC Regional Asia-Jakarta (2013,2014,2015,dan 2016). Penulis dapat dihubungi melalui surel di `louis.arianto@gmail.com`

(Halaman ini sengaja dikosongkan)