PENYELARASAN TINGKAT KENYARINGAN BERKAS SUARA DIGITAL DENGAN PENYAMAAN DAYA SINYAL BERKAS SUARA

Muhammad Ardi Nur Syamsu[*)], Achmad Hidayatno, and Ajub Ajulian Zahra

Jurusan Teknik Elektro, Universitas Diponegoro Semarang
Jl. Prof. Sudharto, SH, kampus UNDIP Tembalang, Semarang 50275, Indonesia

[*)] *Email: ardinursyamsu@yahoo.com*

Abstrak

Berkas suara digital yang biasa digunakan sebagai media penyimpanan musik pada umumnya memiliki tingkat rerata kenyaringan yang berbeda pada setiap berkasnya. Ketika beberapa berkas tersebut dimainkan secara berturut-turut, perubahan tingkat kenyaringan yang terjadi pada setiap berkas dapat menjadi gangguan yang cukup serius bagi sebagian orang sehingga perlu diatur secara manual. Beberapa metode telah digunakan yaitu dengan menyamakan amplitude tertinggi sinyal pada nilai tertentu atau dengan menyamakan aras rerata berkas suara digital pada nilai tertentu. Sinyal pada bentuk apapun baik analog maupun digital memiliki sebuah besaran yang dapat diukur yaitu daya. Pada penelitian ini parameter daya yang ada pada setiap berkas suara digital dihitung untuk dicari tahu apakah nilai besaran daya sebanding dengan aras kenyaringan dari lagu yang bersangkutan. Dari beberapa berkas yang menjadi bahan data uji dihitung nilai dayanya kemudian dikumpulkan untuk dicari nilai tengah yang digunakan sebagai nilai ambang daya. Nilai ambang daya ini selanjutnya digunakan sebagai acuan untuk menentukan penguatan atau pelemahan yang perlu dibobotkan pada amplitude sinyal suara yang dimainkan yang akan mempengaruhi aras kenyaringan keluaran sinyal suara tersebut**.** Dari pengujian yang dilakukan secara kualitatif dengan beberapa responden dihasilkan bahwa daya sinyal pada berkas suara digital berhubungan dengan aras kenyaringan berkas tersebut.

Kata kunci : berkas suara digital, daya sinya digital, aras kenyaringan, python

Abstract

Digital audio file that is generally used as music storage usually have different loudness level. When some of the files are played simultaneously, change in loudness level that occurs can be serious disturbance for some people. Thus they have to change the volume level manually to adjust the loudness. Some methods have been used to linearize loudness between audio file. One way is to find the largest sample amplitude and adjust it to desired level. Another method is to measure mean level and adjust them so that all files get the same mean level. Signal in either analog or digital form has a measurable quantity. One of them is called signal power. In this study, signal power in every audio file is measured so it's relationship with loudness can be found. Files that are used as testing samples are calculated to find their signal power. The signal power is then gathered to find the average. Threshold from average signal power is used as reference value to determine if gain or attenuation should be applied into the file's signal amplitude. From the test that’ve been held quantitatively resulting that signal power in audio files correspond to loudness level of those files.

Keyword : digital audio file, digital signal power, loudness level, python

1. Pendahuluan

Banyak berkas suara digital yang memiliki aras kenyaringan yang berbeda-beda antar berkas. Ketika beberapa berkas tersebut dimainkan secara berturut-turut, perubahan aras kenyaringan yang terjadi pada setiap berkas dapat membuat beberapa pendengar merasa cukup terganggu sehingga merasa perlu untuk mengatur secara manual tingkatan *volume* keluaran suara. Pengaturan ini dapat dilakukan dengan memutar tuas pengatur *volume* pada perangkat keras atau dengan menggeser *slider* pada *master volume* sistem operasi.

Beberapa metode telah digunakan untuk mencoba menyelaraskan kenyaringan antar berkas suara digital. Salah satu cara adalah dengan melakukan normalisasi amplitude puncaknya, yaitu dengan mencari nilai cuplik tertinggi dan mengatur agar setiap cuplikan tertinggi pada setiap berkas suara digital sama[4]. Cara lainnya adalah dengan menghitung aras rerata pada berkas dan

mengaturnya agar setiap berkas memiliki nilai yang sama[5].

Daya sinyal didefinisikan sebagai energi per satuan waktu. Suatu nilai yang tidak bergantung pada durasi sinyal bisa didapatkan dengan menghitung daya sinyal tersebut. Pada penelitian ini akan diteliti hubungan besaran daya sinyal tersebut dengan aras kenyaringan yang mewakili suatu berkas suara digital. Dengan meneliti hubungan tersebut nantinya akan diketahui apakah dengan menyamakan daya pada berkas suara digital dapat menyelaraskan aras kenyaringan antar berkas-berkas tersebut.

Suatu nilai daya ambang ditentukan sebagai acuan perbandingan antara daya berkas dengan daya ambang tersebut. Dari perbandingan tersebut didapatkan nilai penguatan atau pelemahan yang akan dibobotkan pada amplitude sinyal berkas suara digital bersangkutan ketika dimainkan. Dengan menggunakan cara demikian diharapkan dapat terjadi kenyaringan keluaran suara antar berkas yang konsisten.

2. Metode

2.1. Perancangan perangkat lunak

Perancangan perangkat lunak yang digunakan pada penelitian ini menggunakan bahasa pemrograman python. Bahasa ini dipilih karena mudah digunakan, diimplementasikan, dan dapat berjalan pada banyak sistem operasi. Selain itu, bahasa pemrograman ini bersifat kode terbuka yang memiliki *compiler* gratis dan dukungan dari komunitas yang luas.

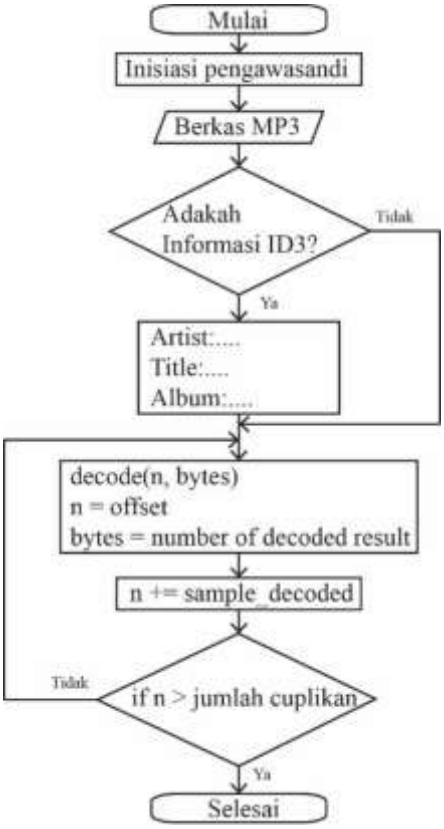
Beberapa modul pendukung juga ditambahkan ke dalam perangkat lunak ini untuk menjalankan fungsi utama yang berkaitan dengan pengawasandian berkas suara termampatkan MP3 menjadi nilai-nilai cuplik PCM dan pengakses perangkat keras keluaran suara. Pengawasandi yang digunakan pada perancangan perangkat lunak ini adalah mpg123 sementara pengakses perangkat keras keluaran suara adalah portaudio.

2.1.1. MP3Dec

MP3Dec merupakan kelas yang terdapat pada modul pendukung mp3dec. Tugas dari MP3Dec adalah melakukan *binding* terhadap libmpg123[4] yang merupakan *dynamic-link-library* (DLL) bagian dari mpg123. Modul ini berfungsi mengawasandikan *frame* MP3 menjadi data PCM tak-termampatkan.

Dalam kelas MP3Dec ini terdapat beberapa fungsi yaitu: `__init__()`, `Decode()`, `SeekSample()`, dan `SeekSecond()` yang berguna untuk mencari dan mendapatkan data PCM dari berkas MP3 sesuai dengan keinginan pemrogram. Fungsi-fungsi dalam kelas ini dijabarkan dalam bagian berikut.

1. Fungsi `__init__()`
   Fungsi ini merupakan *constructor*[1] yang merupakan fungsi yang pertama kali dipanggil secara otomatis ketika objek dibentuk menggunakan *template* dari kelas ini. Satu-satunya parameter yang terdapat pada fungsi ini adalah parameter `filename`. Parameter ini digunakan untuk menghubungkan berkas MP3 dengan pengawasandi melalui fungsi `mpg123_open()` yang merupakan fungsi bawaan dari DLL libmpg123. Dengan fungsi ini juga didapatkan informasi-informasi penting seperti laju pencuplikan, kedalaman bit, dan jumlah kanal.
2. Fungsi `Decode()`
   Fungsi ini adalah fungsi yang digunakan untuk mengawasandikan informasi termampatkan pada *frame* berkas MP3 menjadi data PCM tak termampatkan. Fungsi ini mengambil parameter `bytes_to_be_decoded` yang merupakan nilai untuk menentukan besarnya jumlah *bytes* yang menjadi hasil proses pengawasandian. Nilai ini tidak tergantung dengan jumlah kanal, laju pencuplikan, atau kedalaman bit dari berkas yang diawasandikan.
3. Fungsi `SeekSample()`
   Fungsi ini digunakan untuk menempatkan *offset* pada nilai tertentu dalam berkas. Nilai *offset* ini ditentukann dalam satuan *sample* atau cuplikan dan tidak menunjuk pada *frame* tertentu pada berkas MP3. Dan perlu diingat bahwa nilai ini juga tidak bergantung pada jumlah kanal dan kedalaman bit dari lagu tersebut.
4. Fungsi `SeekSecond()`
   Fungsi ini merupakan pembungkus untuk fungsi `SeekSample()` agar menjadi lebih sederhana dengan menentukan titik *offset* pada ukuran detik dari berkas lagu. Fungsi ini mengambil parameter *offset* berupa detik dan mengubahnya menjadi cuplikan untuk diumpankan ke fungsi `SeekSample()`. Nilai masukan pada parameter ini dapat berupa pecahan dengan pembulatan ke cuplikan terdekat pada saat fungsi ini mengubah dari satuan detik ke satuan cuplikan.



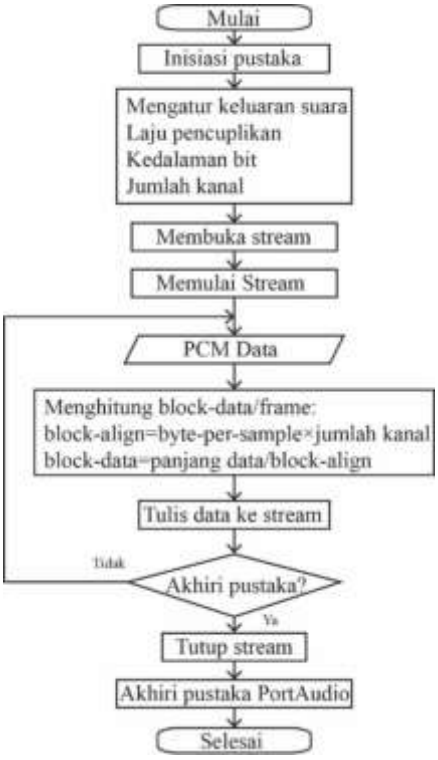
Gambar 1. Diagram-alir kelas MP3Dec

2.1.2. PAudio

PAudio merupakan kelas yang terdapat pada modul pendukung paudio. Tugas dari PAudio adalah melakukan *binding* terhadap portaudio yang merupakan *dynamic-link-library* pengakses perangkat keras keluaran suara. Keunggulan menggunakan portaudio sebagai pengakses perangkat keluaran suara adalah sifatnya yang *blocking* sehingga pemrogram tidak perlu menghitung durasi (*timing*) antara 2 *buffer* data PCM yang cenderung menyebabkan galat saat pengeksekusian kode atau gangguan saat pemutaran berkas suara digital.

Dalam kelas PAudio ini terdapat 3 buah fungsi yang digunakan untuk mengakses perangkat keras keluaran suara. Fungsi-fungsi ini adalah `__init__()`, `write()`, dan `__del__()`. Penjelasan dari fungsi-fungsi tersebut adalah sebagai berikut.

1. Fungsi `__init__()`
   Fungsi *constructor* ini bertugas untuk melakukan persiapan pustaka PortAudio. Pertama-tama, fungsi ini akan menginisiasi pustaka PortAudio agar berjalan pada sistem operasi. Kemudian, sesuai informasi yang baik ditetapkan oleh pengguna secara langsung maupun nilai yang diambil dari pengawasandi, fungsi ini menetapkan jumlah kanal suara, jumlah kedalaman bit setiap cuplikan, dan laju pencuplikan agar suara yang keluar dari perangkat keras nantinya sesuai dengan informasi pada berkas suara digital yang dimainkan. Untuk mengirimkan data dari program ke perangkat keras, fungsi ini menggunakan sebuah peubah yang disebut *stream*. Stream ini merupakan *logical representation* dari perangkat-keras.
2. Fungsi `write()`
   Fungsi ini digunakan untuk menulis data PCM ke perangkat keluaran suara. Pada fungsi ini terdapat sebuah parameter yaitu `data`. Parameter ini harus diisi dengan data PCM tak-termampatkan. Pemrogram tidak perlu menetapkan informas-informasi lainnya karena informasi penting seperti jumlah *block-align* dan *block-data* secara otomatis dihitung oleh fungsi ini sebagai informasi tambahan untuk diumpankan ke perangkat keluaran suara.



Gambar 2. Diagram-alir kelas PAudio

3. Fungsi `__del__()`
   Fungsi ini merupakan *destructor* yang memiliki tugas yang berlawanan dari *constructor*. Fungsi ini hanya dijalankan ketika suatu objek dimusnahkan, dibuang dari memori dan ditangkap oleh *garbage collector*. Pada kelas ini, fungsi ini berguna untuk menghapus jalur yang mengirimkan data dari program ke perangkat keras yaitu *stream*, dan mengakhiri *binding* program dengan pustaka PortAudio.

2.2. Subrutin penghitung daya

Program utama dibagi menjadi dua subrutin yang salah satunya adalah subrutin penghitung daya. Subrutin penghitung daya ini adalah fungsi untuk menghitung daya

sinyal berkas suara digital. Perhitungan menggunakan rumus 1[2] untuk suatu kuantitas diskrit dari nilai PCM tak termampatkan berjumlah N cuplikan. Dari nilai daya tersebut direpresentasikan dalam satuan dBFs yang nilainya dihitung dari rumus 2[3] dengan R adalah kedalaman bit dari cuplikan dan P_x adalah daya hasil perhitungan.

$$P_x = \frac{1}{N}\sum_0^{N-1}|x(n)|^2 \quad (1)$$

$$power(dBFs) = 10.\log\left(\frac{P_x}{2^{2(R-1)}}\right) \quad (2)$$

Karena daya dihitung dalam bentuk bongkahan (*chunk*), maka daya total pada berkas haruslah dihitung dengan rumus 3.

$$P_{total} = \frac{P_0+P_1+\cdots+P_{N-1}}{N} \quad (3)$$

2.2.1. Menghitung gain

Cara menghitung penguatan untuk pembobotan pada saat pemutaran berkas suara digital adalah dengan menggunakan perhitungan berikut.

$$\frac{P_1}{P_2} = \frac{\frac{1}{2}{A_1}^2}{\frac{1}{2}{A_2}^2}$$

$$\frac{P_1}{P_2} = \frac{{A_1}^2}{{A_2}^2}$$

$${A_2}^2 = {A_1}^2\frac{P_2}{P_1}$$

$$A_2 = A_1\sqrt{\frac{P_2}{P_1}} \quad (4)$$

Jika $\sqrt{\frac{P_2}{P_1}}$ adalah nilai penguatan, maka penguatan (*gain*) bisa dihitung dengan.

$$Gain = \frac{1}{2}(P_2 - P_1)_{dB}$$

J

ika P_1 merupakan daya sinyal pada berkas suara digital (P_{file}) sementara P_2 adalah daya acuan (P_{ref}), maka

$$Gain = \frac{1}{2}(P_{ref} - P_{file})_{dBFs}$$

$$Gain = 10^{\frac{(P_{ref}-P_{file})dBFs}{2.10}}$$

$$Gain = 10^{\frac{(P_{ref}-P_{file})dBFs}{20}} \quad (5)$$

Nilai tersebut disimpan ke dalam basis-data agar pada kebutuhan selanjutnya tidak perlu melakukan perhitungan lagi.

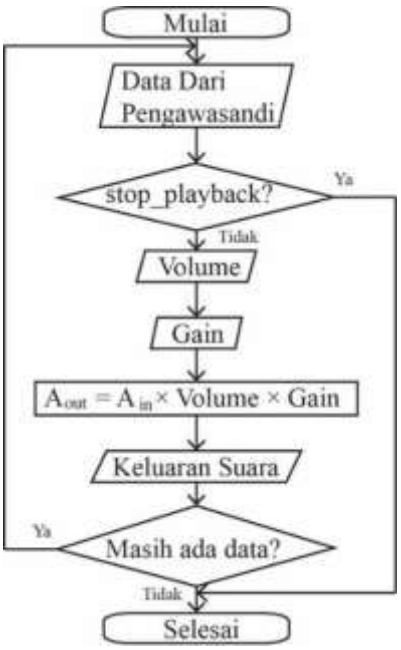
2.3. Subrutin pemutar berkas MP3

Subrutin ini bertugas untuk memainkan berkas suara digital dengan memberikan penguatan jika terdapat data penguatan untuk berkas tersebut, atau tidak jika informasi penguatan yang dibutuhkan tidak tersedia. Pada subrutin ini, pengguna (*user*) diberi kendali untuk menentukan kondisi pemutaran berkas seperti kendali terhadap posisi lagu (*seek*), *play*, *pause*, *stop*, memilih lagu selanjutnya (*next*), memutar lagu sebelumnya (*previous*), mengatur aras kenyaringan (*volume*), serta mengaktifkan fitur penyelarasan kenyaringan berkas suara digital.

1. Kendali *play*
   Kendali ini memainkan berkas yang berada dalam keadaan *stop* atau *pause*. Jika dalam keadaan *stop*, maka berkas dimainkan dari awal sementara dalam keadaan *pause*, berkas dimainkan dari posisi *offset* terakhir ketika di-*pause*. Kendali ini menonaktifkan pemanji `stop_playback`. Secara lengkap diagram alir fitur ini dapat dilihat pada gambar 4.
2. Kendali *pause*
   Kendali tombol *pause* ini akan mengaktifkan pemanji `stop_playback` menjadi bernilai benar. Dengan demikian, sub-proses pemutaran berkas suara digital akan diakhiri sehingga pemutaran berhenti. Pada kendali ini tidak mengubah nilai *offset* pada pengawasandi sehingga ketika kendali *play* diaktifkan kembali, berkas akan diputar tepat dari posisi terakhir berkas berhenti diputar.
3. Kendali *stop*
   Kendali *stop* ini seperti kendali *pause* yang mengaktifkan pemanji `stop_playback` menjadi bernilai benar sehingga mengakhiri sub-proses pemutaran berkas suara. Bedanya pada kendali ini, nilai *offset* pada pengawasandi akan diatur-ulang ke posisi awal atau 0, sehingga ketika pengguna mengaktikan kendali *play*, berkas akan diputar dari awal bukan dari posisi terakhir
4. Kendali aras *volume*
   Kendali ini untuk mengatur kenyaringan berkas yang diputar secara manual. Nilai pembobotan ini berkisar antara 0 sampai 1 yang akan dibobotkan ke amplitude cuplikan sinyal suara.
5. Kendali *next*
   Kendali ini akan menghentikan pemutaran lagu yang sedang diputar untuk kemudian memilih berkas selanjutnya yang terdapat pada daftar putar. Pemutaran berkas ini secara otomatis memutar berkas selanjutnya tanpa pengguna harus mengaktifkan kendali *play*.
6. Kendali *previous*
   Sama seperti *next* hanya saja kendali ini memilih lagu sebelumnya yang terdapat pada daftar putar.
7. Kendali *seek*
   Kendali ini digunakan untuk memindahkan posisi lagu yang diputar ke bagian tertentu. Kendali ini menetapkan nilai *offset* pada pengawasandi dengan memanfaatkan fungsi `SeekSample()` yang terdapat pada pengawasandi.

Tampilan lengkap antarmuka perangkat lunak dapat dilihat pada gambar 3 berikut.

Gambar 3. Tampilan antarmuka perangkat lunak



Gambar 4. Diagram-alir kendali play

3. Hasil dan analisis

Pada bagian ini dibahas hasil dari pengujian perangkat lunak dan fitur penyelarasan daya. Pengujian dibagi mennjadi 2 yaitu pengujian kecepatan dan persepsi. Pengujian kecepatan bertujuan untuk mengetahui kecepatan eksekusi pada prosesor yang berbeda. Pengujian persepsi bertujuan untuk mencari hubungan daya sinyal dengan aras kenyaringan dan keberhasilan fitur untuk menyelaraskan kenyaringan.

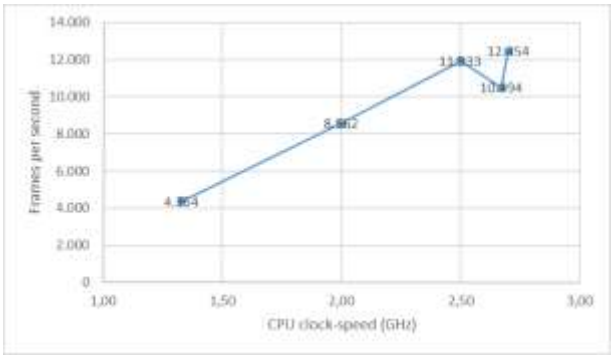
3.1. Kecepatan pengawasandian berkas MP3

Kecepatan pengawasandian yang diukur disini adalah nilai jumlah *frame* yang dapat diawasandikan per detik pada prosesor dengan *clock-speed* yang berbeda-bedaa. Ukuran satu *frame* digunakan karena pada berkas MP3 unit terkecil penyusun berkas adalah *frame*. Dari hasil pengujian didapatkan data berikut.

Tabel 1. Hasil pengujian kecepatan pengawasandian jumlah *frame*/detik berkas MP3 pada prosesor yang berbed

Nama Prosesor	*Clock-speed* (GHz)	Kecepatan Pengawasandian (*frame*/detik)
Intel Atom Z3735F	1,33	4.354
Intel Pentium E2180	2,00	8.562
Intel Core i3 3120M	2,50	11.933
Intel Core i3 390M	2,67	10.494
Intel Pentium G630	2,70	12.454

Dari nilai tersebut kemudian digambarkan pada grafik yang ditampilkan pada gambar berikut.



Gambar 5. Grafik hasil pengujian kecepatan pengawasandian frame/detik pada prosesor yang berbeda

Terlihat pada gambar grafik tersebut bahwa kecepatan pengawasandian berbanding lurus dengan *clock-speed* prosesor. Pada prosesor dengan *clock-speed* yang lebih tinggi, jumlah *frame* yang dapat diawasandikan per detik semakin banyak.

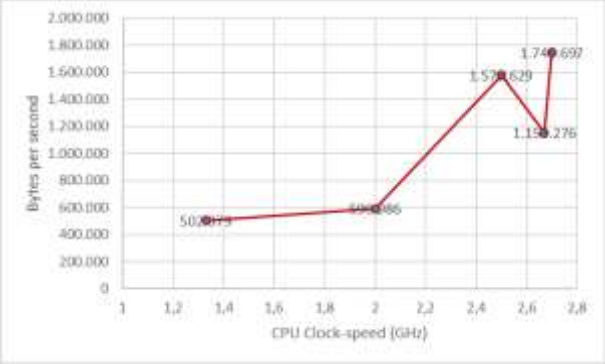
3.2. Kecepatan perhitungan daya berkas MP3

Kecepatan peritungan daya pada berkas suara digital tidaklah berpengaruh terhadap proses pemutaran berkas. Namun demikian, durasi perhitungan daya yang terlalu lama akan membuat pengguna tidak nyaman walaupun proses perhitungan sendiri dijalankan pada *thread* yang terpisah sehingga tidak mengganggu jalannya program utama. Nilai perhitungan kecepatan diukur dalam satuan *bytes*/detik untuk mengetahui hubungan ukuran berkas dengan kecepatan pengukuran daya sinyal. Hasil dari pengujian ditampilkan pada tabel berikut.

Tabel 2. Hasil pengujian kecepatan perhitungan daya pada berkas MP3 dengan prosesor yang berbeda

Prosesor	Clock-speed (GHz)	Kecepatan (*bytes*/detik)
Intel Atom Z3735F	1,33	502.379
Intel Pentium E2180	2,00	590.986
Intel Core i3 3120M	2,50	1.579.629
Intel Core i3 390M	2,67	1.153.276
Intel Pentium G630	2,70	1.745.697

Apabila data disajikan dalam bentuk grafik, akan terlihat bahwa trend kenaikan *clock-speed* prosesor berbanding lurus dengan jumlah *bytes* dari berkas MP3 yang dapat dihitung dayanya per detik seperti yang terlihat pada gambar berikut.



Gambar 6. Grafik hasil pengujian kecepatan perhitungan daya sinyal pada prosesor yang berbeda

3.3. Hubungan daya sinyal dengan persepsi tingkat kenyaringan

Pengujian dilakukan dengan survey terhadap 45 responden yang secara sukarela bersedia melakukan pengujian terhadap hasil akhir perangkat lunak penelitian ini. Pengujian dilakukan untuk menentukan lagu mana saja yang lebih nyaring dari lagu-lagu lainnya tanpa responden mengetahui nilai daya dari lagu-lagu tersebut. Ukuran kenyaringan suatu lagu ditentukan dengan banyaknya responden yang setuju bahwa lagu tersebut lebih nyaring daripada yang lain. Hasil pengujian disajikan pada tabel berikut.

Tabel 3. Data hasil jumlah pilihan responden terhadap lagu yang memiliki aras kenyaringan yang lebih tinggi

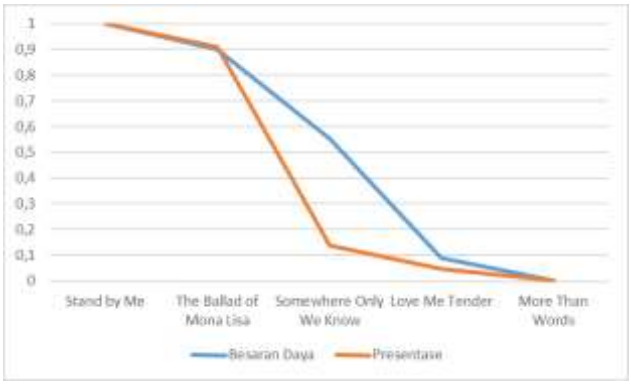
Judul Lagu	Jumlah Responden Memilih	Presentase
Stand by Me	22	48,89%
The Ballad of Mona Lisa	20	44,44%
Somewhere Only We Know	3	6,67%
Love Me Tender	1	2.22%
More Than Words	0	0.00%

Sementara nilai daya dari lagu yang bersangkutan disajikan pada tabel berikut.

Tabel 4. Nilai hasil pengukuran daya pada berkas-berkas suara digital yang digunakan sebagai bahan uji

Judul Lagu	Artis	Nilai Daya (dBFs)
Stand by Me	Oasis	-8,9305
The Ballad of Mona Lisa	Panic! At The Disco	-10,1797
Somewhere Only We Know	Keane	-14,5443
Love Me Tender	Frank Sinatra	-20,3621
More Than Words	Extreme	-21,4841

Dari data-data tersebut dapat disajikan dalam bentuk grafik yang dinormalisasi.

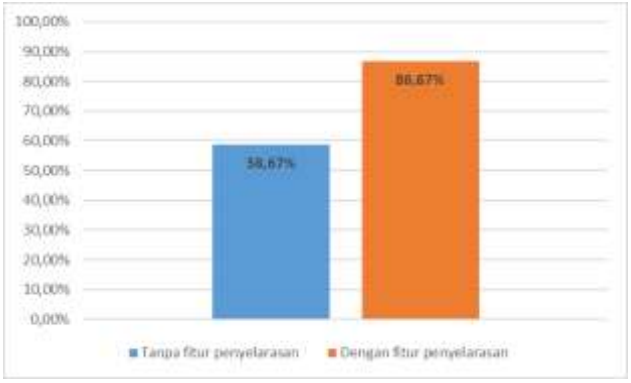


Gambar 7. Grafik Besaran Daya dan Tingkat Kenyarinngan Menurut Responden, Yang Telah Dinormalisasi

Dari grafik tersebut terlihat bahwa antara grafik yang dihasilkan dari pengukuran daya sinyal dengan presentase jumlah responden akan persepsi mereka tentang kenyaringan berkas suara digital tidak benar-benar sama persis. Namun demikian, hasil pengujian aras kenyaringan sudah dapat mewakili hasil pengukuran daya.

3.4. Hubungan penyelarasan daya sinyal dengan keselarasan aras kenyaringan

Pada bagian ini akan diuji apakah nilai daya pada sinyal suara digital dapat digunakan untuk menyelaraskan aras kenyaringan pada lagu-lagu yang memiliki aras kenyaringan yang berbeda. Ukuran yang dipakai pada pengujian ini adalah ketidak-selarasan yang diwakili dengan presentase jumlah lagu yang tidak selaras pada bahan uji. Pengujian dilakukan terhadap 45 responden dengan lagu-lagu bahan uji yang sama. Dari hasil pengujian dihasilkan keselarasan sebelum dan sesudah pengaktifan fitur seperti yang disajikan pada gambar berikut.



Gambar 8. Grafik rerata ukuran keselarasan antar lagu hasil pendapat responden

Dari grafik dapat disimpulkan bahwa sebelum fitur penyelarasan diaktifkan, rerata keselarasan lagu dari setiap responden hanya 58,67% atau hampir 3 lagu selaras aras kenyaringannya. Setelah fitur diaktifkan, presentase keselarasannya naik menjadi 86,67% yang berarti lebih dari 4 lagu menurut responden sudah selaras.

3.5. Pengujian tingkat galat pada pengawasandi dan fitur penyelarasan

Pada bagian ini dilakukan pengujian keandalan pengawasandi dan fitur. Ukuran dari ketidak-andalan pengasanadi adalah derau yang dihasilkan yang terdengar oleh responden pada saat pemutaran baik saat fitur penyelarasan daya diaktifkan maupun tidak. Ukuran dari ketidak-andalan atau kecacatan fitur adalah dampak terjadinya derau pada lagu yang dimainkan dengan fitur penyelarasan diaktifkan. Dari hasil pengujian didapatkan nilai bahwa probabilitas galat pada pengawasandi adalah 2,67% sementara galat akibat pengaktifan fitur adalah 2,22%. Nilai tersebut ditampilkan pada tabel berikut.

Tabel 5. Probabilitas terjadinya galat pada pengawasandi dan pengaktifan fitur penyelarasan daya sinyal suara digital

Jenis galat	Probabilitas Kejadian
Kejadian galat pada pengawasandi	2,67%
Kejadian Galat akibat fitur penyelarasann	2,22%

4. Kesimpulan

Dari hasil pengujian didapatkan bahwa kecepatan pengawasandian sebuah *frame* pada berkas MP3 bergantung pada clock-speed prosesor. Pada prosesor dengan *clock-speed* 1,33 GHz, kecepatan pengawasandiannya 4.354 *frame*/detik sementara pada prosesor dengan *clock-speed* 2,70 GHz, kecepatan pengawasandiannya 12,454 *frame*/detik. Hal ini berarti semakin cepat *clock-speed* prosesor semakin banyak *frame* yang dapat diawasandikan per detik.

Dari hasil pengujian didapatkan bahwa kecepatan perhitungan daya berkas MP3 dipengaruhi oleh besar ukuran berkas dan kecepatan prosesor. Pada prosesor dengan *clock-speed* 1,33 GHz, kecepatan perhitungannya 502.379 *bytes*/detik sementara pada prosesor dengan *clock-speed* 2,70 GHz, kecepatan perhitungannya 1.745.697 *bytes*/detik. Dengan demikian untuk menghitung berkas MP3 yang banyak lebih baik menggunakan prosesor dengan *clock-speed* yang lebih cepat.

Dari hasil pengujian didapatkan bahwa kenyaringan berkas suara digital dapat diwakili dengan nilai dayanya dengan daya yang lebih tinggi berarti berkas tersebut memiliki keluaran suara yang lebih nyaring. Hal ini dapat dilihat pada lagu *Stand By Me* dengan daya yang paling tinggi (−8,93 dBFs) diwakili oleh presentase jumlah responden tertinggi yaitu 48,89% sementara lagu yang memiliki daya terendah (−21,48 dBFs) yaitu *More Than Words* diwakili dengan presentase terkecil yaitu 0%.
Dari hasil pengujian didapatkan bahwa penyelarasan nilai daya pada berkas suara digital dapat digunakan untuk menyelaraskan aras kenyaringan keluaran suara dari berkas-berkas tersebut. Hal ini dapat dilihat dari pengujian sebelum fitur penyelarasan diaktifkan yang menghasilkan presentase keselarasan 58,67% meningkat menjadi 86,67% setelah fitur penyelarasan diaktifkan.

Referensi

[1]. M. Lutz, *Learning Python*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2008.

[2]. M. Bosi dn R. E. Goldberg, *Introduction To Digital Audio Coding Standards.* Dodrecht, Netherlands: 2003.

[3]. K. Ayob, "Power Computations," dalam *Digital Filter Design for FPGA Engineers*, North Charleston, SC, USA: CreateSpace Independent Publishing Platform, 2014, hal. 74.

[4]. E. Vickers, "Automatic Long-Term Loudness and Dynamics Matching," dipresentasikan di 11th Convention, New York, NY, USA, Sep. 21-24, 2001.

[5]. P. Nygren, "Achieving equal loudness between audio files – Evaluation and improvements of loudness algorithms," M.S. thesis, Kungliga Tekniska högskolan at Brinellvägen 8, 114 28 Stockholm, Sweden. 2009.