IMPLEMENTASI ALGORITME SHAZAM UNTUK MENGIDENTIFIKASI HADIS DAN SURAH DALAM AL-QURAN MENGGUNAKAN SUARA

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Bahruddin El Hayat
NIM: 145150200111045

PPROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2017

PENGESAHAN

IMPLEMENTASI ALGORITME SHAZAM UNTUK MENGIDENTIFIKASI HADIS DAN SURAH DALAM AL-QURAN MENGGUNAKAN SUARA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Bahruddin El Hayat
NIM: 145150200111045

Skripsi ini telah diuji dan dinyatakan lulus pada
17 Januari 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Imam Cholissodin, S. Si, M. Kom	Drs.Marji, M. T
NIK. 201201 850719 1 001	NIP. 19670801 199203 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D
NIP. 19710518 200312 1 001

PERNYATAAN ORISINALITAS

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

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

Malang, 2 Januari 2018

Bahruddin El Hayat

NIM: 145150200111045

KATA PENGANTAR

Dengan mengucap *alhamdulillahi rabbil 'alamin*, Segala puji syukur penulis panjatkan kehadirat Allah SWT, karena atas limpahan berkah, rahmat dan hidayan-Nya penulis mampu menyelesaikan laporan skripsi dengan judul "*Implementasi Algoritme Shazam Untuk Mengidentifikasi Hadis Dan Surah Dalam Al-Quran Menggunakan Suara*".

Kemudian tidak lupa juga penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan bantuan baik secara moril maupun secara materil selama proses pelaksanaan dan penulisan laporan skripsi ini. Untuk itu penulis ingin mengucapkan terimakasih sebesar-besarnya kepada:

1. Bapak Imam Cholissodin, S.Si., M.Kom dan Bapak Drs.Marji, M.T. selaku dosen pembimbing skripsi yang telah membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini dengan baik.
2. Bapak Wayan Firdaus Mahmudy,S. Si, M.T, Ph. D. , Bapak Ir. Heru Nurwasito, M.Kom, Bapak Drs. Mardji, M. T, dan Bapak Edy Santoso, S.Si, M.Kom selaku Dekan, Wakil Dekan 1, Wakil Dekan 2 dan Wakil Dekan 3 Fakultas Ilmu Komputer Universitas Brawijaya Malang.
3. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku kepala program studi Teknik Informatika Fakultas Ilmiu Komputer Universitas Brawijaya.
4. Ibu, Ayah, Adik pertama, Adik kedua serta keluarga tersayang atas segala nasehat, semangat, dukungan, perhatian, uang saku dan kasih sayang yang diberikan kepada penulis.
5. GLDev team, Keluarga LINDISANTI Squad dan MountainActor yang senantiasa memberikan dukungan moril dan hiburan kepada penulis selama proses penulisan laporan skripsi ini.
6. Seluruh pihak yang telah membantu kelancaran penulisan laporan skripsi yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa dalam penyusunan laporan skripsi ini masih banyak kekurangan baik format penulisan maupun isinya. Oleh karena itu, saran dan kritik membangun dari pembaca senantiasa penulis harapkan guna perbaikan bagi laporan skripsi selanjutnya. Semoga laporan skripsi ini dapat memberikan manfaat bagi semua pihak, Amin.

Malang, 27 Desember 2017

Penulis

bbandroel@gmail.com

ABSTRAK

Bagi umat muslim mengerti akan Al-Qur'an dan hadis adalah suatu kewajiban karena kedua hal tersebut merupakan dasar ajaran Islam. Dalam proses memahami biasanya akan diawali dengan menghafalkan lafal beserta nama surah atau judul dari hadis. Kemudian setelah menghafal lafal dengan lancar barulah akan dilanjutkan ke arti dan kandungan. Akan tetapi ketika pada tahap menghafal lafal ini yang menjadi masalah utama adalah lupa. Ketika mendengar bacaan salah satu surah atau hadis seseorang tidak mampu mengingat bacaan tersebut merupakan surah atau hadis apa. Oleh karena itu diperlukannya sebuah solusi yang mampu menangani masalah tersebut. Pada penelitian ini peneliti mengusulkan sebuah sistem yang mampu mengidentifikasi hadis dan surah dalam Al-Qur'an dengan masukan berupa file suara menggunakan algoritme *shazam*. Dengan masukan berupa file suara dengan ekstensi WAV identifikasi dilakukan dengan langkah-langkah berikut, yaitu: ekstraksi nilai numerik, konversi, ekstraksi fitur, filtering, dan pencocokan. Hasil yang diperoleh berupa nama beserta informasi mengenai surah atau hadis dari file suara yang menjadi masukan. Hasil akurasi terbaik yang didapatkan pada identifikasi hadis dan surah dalam Al-Qur'an ini mencapai 82% pada pengujian menggunakan data uji berdurasi 15 detik, chunk size = 4096 dan range = 60.

Kata kunci: Al-Qur'an, Hadis, Identifikasi, Suara, Shazam.

ABSTRACT

For muslims, understanding Al-Qur'an and Hadis are an obligation because those two is the basic of Islam. In the learning process, a person usually will begin by memorizing the pronounciation and the surah's or hadis's name. After the person can pronounce the surah or hadis fluently, then they will continue to understand the meaning and the content of the surah or hadis. The problem is sometimes a person can forget the name of surah or hadis when another person says a verse from the surah or hadis. So, a solution is needed to handle the problem. In this research, the writer offers a solution to build a system that can identify the name of surah or hadis in Al-Qur'an by taking an input in the form of a sound file with WAV extension using Shazam algorithm. The identification process is done by doing these following actions: numeric value extraction, conversion, feature extraction, filtering and matching. The result is the name and information of the surah or hadis. The best accuracy from identifying surah and hadis from Al-Qur'an is 82% in the testing phase using a test data with 15 second duration, chunk size=4096 and range=60.

Keywords: Al-Qur'an, Hadis, identification, Sound, Shazam.

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK v
ABSTRACT vi
DAFTAR ISI vii
DAFTAR TABEL x
DAFTAR GAMBAR xi
DAFTAR KODE PROGRAM xii
DAFTAR LAMPIRAN xiii
BAB 1 PENDAHULUAN 1
1.1 Latar belakang 1
1.2 Rumusan masalah 3
1.3 Tujuan 3
1.4 Manfaat 3
1.5 Batasan masalah 3
1.6 Sistematika pembahasan 4
BAB 2 LANDASAN KEPUSTAKAAN 6
2.1 Kajian Pustaka 6
2.2 Landasan Teori 8
2.2.1 Al-Qur'an 8
2.2.2 Hadis 9
2.2.3 Bilangan Kompleks 10
2.2.4 Binary Komplemen Dua 12
2.2.5 Frame Blocking 13
2.2.6 Windowing 13
2.2.7 Fast Fourier Transform 14
2.2.8 Algoritme Shazam 15
2.2.9 File WAV 19
2.2.10 Akurasi 19

BAB 3 METODOLOGI 21
3.1 Studi Pustaka 21
3.2 Pengumpulan Data 21
3.3 Analisis Kebutuhan 22
3.4 Perancangan 22
3.5 Implementasi 22
3.6 Pengujian dan Analisis 23
3.7 Pengambilan Kesimpulan dan Saran 23
BAB 4 PERANCANGAN 24
4.1 Deskripsi Permasalahan 24
4.2 Penyelesaian Permasalahan Identifikasi Hadis dan Surah Dalam Al-Qur'an Menggunakan *Algoritme Shazam* 24
4.2.1 Proses Ekstraksi Nilai Numerik 26
4.2.2 Proses Konversi 28
4.2.3 Proses Ekstraksi Fitur 32
4.2.4 Proses Filtering 53
4.2.5 Proses Pembentukan Hash 57
4.2.6 Proses Pencocokan 62
4.3 Perancangan Antarmuka 66
4.3.1 Perancangan Halaman Test One Data 66
4.3.2 Perancangan Halaman Add Dataset 67
4.3.3 Perancangan Halaman Testing 68
4.4 Pengujian Algoritme 69
4.4.1 Pengujian Panjang Data Uji 70
4.4.2 Pengujian Besar Chunk 70
4.4.3 Pengujian Range 71
BAB 5 IMPLEMENTASI 72
5.1 Implementasi Algoritme 72
5.1.1 Implementasi Algoritme Ekstraksi Nilai Numerik 72
5.1.2 Implementasi Algoritme Konversi 72
5.1.3 Implementasi Algoritme Ekstraksi Fitur 73
5.1.4 Implementasi Algoritme Filtering 76

5.1.5 Implementasi Algoritme Pembentukan Hash 76
5.1.6 Implementasi Algoritme Pencocokan 77
5.2 Implementasi Antarmuka 78
5.2.1 Implementasi Antarmuka *Test One* Data 78
5.2.2 Implementasi Antarmuka *Add Dataset* 79
5.2.3 Implementasi Antarmuka *Testing* 80
BAB 6 PENGUJIAN DAN ANALISIS 82
6.1 Pengujian Tingkat Akurasi Terhadap Panjang Data Uji 82
6.1.1 Skenario Pengujian Panjang Data Uji 82
6.1.2 Analisis Pengujian Durasi Data Uji 82
6.2 Pengujian Tingkat Akurasi Terhadap *Chunk size* 83
6.2.1 Skenario Pengujian *Chunk size* 83
6.2.2 Analisis Pengujian *Chunk size* 84
6.3 Pengujian Tingkat Akurasi Terhadap *Range* 85
6.3.1 Skenario Pengujian *Range* 85
6.3.2 Analisis Pengujian *Range* 85
BAB 7 PENUTUP 87
7.1 Kesimpulan 87
7.2 Saran 87
DAFTAR PUSTAKA 88
LAMPIRAN A DETAIL HASIL PENGUJIAN 89
LAMPIRAN B DETAIL FITUR DAN HASH DATA 1 DETIK SURAH AL-IKHLAS 91
LAMPIRAN C CONSTELLATION MAPS SURAH AN-NAAS DAN HADIS KE-1 94

DAFTAR TABEL

Tabel 2.1 Kajian Pustaka 6
Tabel 2.2 Kajian Pustaka (Lanjutan) 7
Tabel 4.1 Data Numerik File Suara 28
Tabel 4.2 Data Hasil Proses Konversi 32
Tabel 4.3 Data Hasil Proses Hamming Window 36
Tabel 4.4 Data Hasil Proses Hamming Window (Lanjutan) 37
Tabel 4.5 Data Kompleks 39
Tabel 4.6 Hasil Fourier Transform 44
Tabel 4.7 Data Magnitude Dalam Ruang Log 48
Tabel 4.8 40 Point Frekuensi Dan *Magnitude*-nya 49
Tabel 4.9 40 Point Frekuensi Dan *Magnitude*-nya (Lanjutan) 50
Tabel 4.10 Frekuensi Dan *Magnitude* Di atas Rata-Rata 50
Tabel 4.11 Frekuensi Dan *Magnitude* Di atas Rata-Rata (Lanjutan) 51
Tabel 4.12 Data *Point* Frekuensi Dan *Magnitude* 51
Tabel 4.13 *Range* frekuensi 53
Tabel 4.14 *Point* Pada Setiap *Chunk* 55
Tabel 4.15 *Point* Pada Setiap *Chunk* (Lanjutan) 56
Tabel 4.16 Tabel Data Hasil Filtering 57
Tabel 4.17 Target Zone 59
Tabel 4.18 Pasangan *Anchor Point* dan Target *Zone* 60
Tabel 4.19 Hash 61
Tabel 4.20 Hash (Lanjutan) 62
Tabel 4.21 Data *Hash Dataset* Ke-1 dan Ke-2 64
Tabel 4.22 Ilustrasi Pencocokan *Key Hash* 65
Tabel 4.23 Rancangan Pengujian Panjang Data Uji 70
Tabel 4.24 Rancangan Pengujian Chunk Size 71
Tabel 4.25 Rancangan Pengujian Besar Range 71
Tabel 6.1 Hasil Pengujian Panjang Data Uji 82
Tabel 6.2 Hasil Pengujian *Chunk size* 84
Tabel 6.3 Hasil Pengujian *Range* 85

DAFTAR GAMBAR

Gambar 2.1 Ilustrasi Frame Blocking 13
Gambar 2.2 Diagram Alir Algorime Shazam 16
Gambar 2.3 *Constellation Map* 17
Gambar 2.4 Ilustrasi Target Zone dan Anchor Point 17
Gambar 3.1. Metodologi 21
Gambar 4.1 Diagram Alir Proses Identifikasi menggunakan Algoritme Shazam 25
Gambar 4.2 Diagram Alir Proses Ekstraksi Nilai Numerik 27
Gambar 4.3 Visualisasi Data dalam Satuan *Sample* 28
Gambar 4.4 Visualisasi Data Satu *Sample* 28
Gambar 4.5 Diagram Alir Proses Konversi 29
Gambar 4.6 Diagram Alir Proses Ekstraksi Fitur 33
Gambar 4.7 Diagram Alir Proses Hamming Window 35
Gambar 4.8 Diagram Alir Perhitungan FFT 37
Gambar 4.9 Diagram Alir Proses Getpoints 46
Gambar 4.10 Diagram Alir Getindeks 52
Gambar 4.11 Diagram Alir Proses Filtering 54
Gambar 4.12 Diagram Alir Proses Pembentukan Hash 58
Gambar 4.13 Contoh Representasi Target Zone 59
Gambar 4.14 Contoh Anchor Point dan Target Zone 60
Gambar 4.15 Diagram Alir Proses Pencocokan 63
Gambar 4.16 Rangcangan Halaman Test One Data 67
Gambar 4.17 Rancangan Halaman Add Dataset 68
Gambar 4.18 Rancangan Halaman Testing 69
Gambar 5.1 Implementasi Antarmuka Test One Data 79
Gambar 5.2 Implementasi Antarmuka *Add Dataset* 80
Gambar 5.3 Implementasi Antarmuka Testing 81
Gambar 6.1 Grafik Akurasi Hasil Pengujian Panjang Data Uji 83
Gambar 6.2 Grafik Hasil Pengujian *Chunk size* 84
Gambar 6.3 Grafik Hasil Pengujian Range 86

DAFTAR KODE PROGRAM

Kode Program 5.1 Implementasi Algoritme Ekstraksi Nilai Numerik 72
Kode Program 5.2 Implementasi Algoritme Konversi..... 73
Kode Program 5.3 Implementasi Algoritme Ekstraksi Fitur 73
Kode Program 5.4 Implementasi Perhitungan Hamming 74
Kode Program 5.5 Implementasi Proses Perhitungan FFT 74
Kode Program 5.6 Implementasi Algoritme Getpoints..... 75
Kode Program 5.7 Implementasi Algoritme Getindeks 76
Kode Program 5.8 Implementasi Algoritme Filtering 76
Kode Program 5.9 Implementasi Algortime Pembentukan Hash 77
Kode Program 5.10 Implementasi Algoritme Pencocokan 78

DAFTAR LAMPIRAN

LAMPIRAN A DETAIL HASIL PENGUJIAN 89
LAMPIRAN B DETAIL FITUR DAN HASH DATA 1 DETIK SURAH AL-IKHLAS 91
LAMPIRAN C CONSTELLATION MAPS SURAH AN-NAAS DAN HADIS KE-1 94

BAB 1 PENDAHULUAN

1.1 Latar belakang

Dalam agama Islam, al-Qur'an diyakini sebagai kitab suci yang datangnya langsung dari Allah SWT dan Nabi Muhammad sebagai penerima dan sekaligus utusan Allah SWT untuk mengajarkan segala sesuatu yang terdapat di dalamnya. Al-Qur'an merupakan kitab terakhir yang diturunkan oleh Allah SWT yang bertujuan untuk menjadi petunjuk bagi seluruh umat manusia (Daulay, 2014). Sebagai seorang muslim sudah sepantasnya untuk mendekatkan diri dengan al-Qur'an dengan harapan menjadikannya sebagai acuan dalam melakukan kegiatan sehari-hari. Menurut Ahmad Van Denffer dalam tulisan (Daulay, 2014) mengatakan bahwa terdapat tiga tahapan yang harus dilakukan untuk melakukan pendekatan dengan al-Qur'an, pendekatan tersebut yaitu: membaca dan mendengarkan al-Qur'an untuk menerimanya, menghayati dan melakukan kajian makna yang dikandung untuk memahami pesan-pesan yang dikandung al-Qur'an, dan menjalankan pesan-pesan yang terkandung ke dalam kehidupan sehari-hari. Untuk dapat melaksanakan ketiga tahapan tersebut memang tidaklah mudah, seseorang harus memulai dari tahap awal dengan membiasakan diri untuk membaca dan mendengarkan al-Qur'an.

Pada tahap pertama ini yang akan terasa sulit adalah membaca, karena seperti kita tahu al-Qur'an memakai bahasa arab dan tidak semua orang bisa membaca bahasa Arab. Selain itu perlu ilmu khusus yang dibutuhkan untuk dapat membaca al-Qur'an dengan benar, ilmu tersebut seperti ilmu tajwid dan makhraj. Sehingga pada umumnya sebelum seseorang mampu membaca mereka sudah terlebih dahulu dikenalkan dengan al-Qur'an dengan cara mendengarkan seseorang atau rekaman, dengan mendengar secara tidak langsung akan merangsang rasa ingin tahu seseorang sehingga lambat laun dia mulai bisa menirukan bahkan mengingat apa yang biasa ia dengarkan.

Hal yang sama juga berlaku untuk Hadis, dimana Hadis merupakan salah satu sumber hukum dalam islam yang berkedudukan setelah al-Qur'an (Rosidin & Mahfudhoh, 2014). Untuk dapat mempelajari tentang Hadis tidak ada ilmu khusus yang diharuskan dikarenakan pada umumnya sebuah Hadis yang telah dibukukan sudah tersaji lengkap dengan arti dan penjelasannya. Tetapi akan lain ceritanya jika seseorang ingin mempelajari Hadis langsung dari kitab yang tidak memiliki arti dan penjelasan, dia harus mampu membaca tulisan arab dan harus mampu mengartikannya agar bisa mengerti apa isi dan kandungan dari Hadis tersebut. Dalam kasus ini pendekatan untuk mempelajari Hadis sama dengan al-Qur'an yaitu membaca dan mendengarkan, dan yang pertama dilakukan adalah mendengarkan.

Seseorang akan mampu menghafal hanya dengan mendengar secara rutin dan terus menerus. Selain untuk menghafal, mendengar biasa digunakan untuk mengetes hafalan seseorang baik hafalan al-Qur'an, Hadis, maupun yang lainnya. Ketika seseorang mendengar sebuah suara, otak mereka akan mencari informasi

tentang suara tersebut, dan hal itu akan merangsang rasa penasaran terhadap suara yang sedang didengarkan. Dalam menghafal sesuatu baik itu al-Qur'an ataupun Hadis seseorang pasti akan pernah mengalami masalah yang sudah tidak asing lagi, yaitu lupa. Misalkan ketika seseorang mendengarkan sebuah potongan ayat al-Qur'an yang sudah tidak asing baginya, namun dia tidak tahu terdapat pada surah apa potongan ayat tersebut atau ketika mendengar sebuah Hadis yang sedang dibacakan seseorang tidak tahu Hadis tersebut berbicara tentang apa dan diriwayatkan oleh siapa.

Hal seperti inilah yang mengilhami Shazam Entertainment, Ltd saat membuat aplikasi buatannya, dimana aplikasi buatan shazam ini mampu memberikan informasi tentang suara yang sedang didengarkan. Dalam kasus shazam, objek yang digunakan adalah musik atau lagu. Aplikasi shazam mampu mengidentifikasi musik yang sedang didengarkan kemudian akan memberikan informasi yang telah tersedia tentang musik tersebut, mulai dari judul lagu, nama penyanyi, nama album dan lain sebagainya. Aplikasi milik shazam ini mampu melakukan identifikasi sebuah suara dengan sangat cepat, kurang lebih hanya 10 detik (Wang, 2003). Untuk dapat mengidentifikasi sebuah lagu aplikasi shazam menerapkan *audio fingerprint* yang merupakan identitas khusus untuk sebuah suara, terdapat pada setiap suara dan tentu saja bersifat unik untuk setiap suara (Wang, 2003).

Metode yang digunakan oleh shazam merupakan metode buatan sendiri oleh shazam yang mampu memenuhi semua kriteria yang diinginkan oleh shazam itu sendiri (Wang, 2003). Dalam pengidentifikasian sebuah lagu, ada tiga tahap penting yang dilakukan oleh algoritme shazam yaitu ekstraksi fitur, pembentukan audio fingerprint, pencocokan. Pada dunia aplikasi serupa shazam, ada banyak sekali algoritme yang dapat digunakan seperti algoritme Philips' robust hashing, algoritme the Muscle Fish dan algoritme shazam (Wang, 2003). Algoritme-algoritme tersebut dapat digunakan sesuai dengan kebutuhan dan memiliki kelebihan kekurangan masing-masing.

Pada objek yang sama ataupun algoritme yang sama telah banyak dilakukan penelitian sebelumnya, seperti yang dilakukan Wahidah, et al. (2012), dengan objek berupa huruf hijaiyah dan menggunakan MFCC sebagai ekstraksi fitur untuk pengenalan makhraj huruf. Kemudian pada penelitian Ahmed & Abdo (2017) dengan objek al-Qur'an mereka menggunakan metode MFCC untuk ekstraksi fitur dan HMM untuk pencocokan. Razak, et al. (2008) menjelaskan bahwa dalam sistem verifikasi pembacaan al-Qur'an dipengaruhi oleh beberapa hal, salah satunya adalah penggunaan metode dalam ekstraksi fitur dan metode yang digunakan untuk pencocokan hasil ekstraksi fitur. Dan pada penelitian yang dilakukan oleh Nieuwenhuizen, et al. (2009) dengan objek berupa siaran radio dan menggunaan algoritme shazam serta algoritme haitsma & kalker's dalam mendeteksi iklan suara.

Dari uraian di atas, peneliti menawarkan sebuah solusi untuk menyelesaikan permasalahan tentang lupa pada hafalan al-Qur'an atau Hadis dan tentang mempelajari al-Qur'an dan Hadis yang berbentuk sebuah perangkat yang mampu bekerja layaknya shazam akan tetapi dengan basis objek berupa al-Qur'an dan

Hadis. Algoritme yang akan digunakan adalah algoritme shazam, dimana nanti didalam algoritme shazam akan ada beberapa tahapan yang akan dijelaskan pada bab selanjutnya. Alasan peneliti memilih algortima milik shazam adalah karena algoritme shazam mampu mengenali suara dengan cukup baik walaupun mengandung noise, mampu memproses pencocokan data dengan database dengan sangat cepat dan juga memberikan sedikit false positif.

Berdasarkan penjelasan di atas, penulis mengambil judul skripsi 'Implementasi Algoritme Shazam Untuk Mengidentifikasi Hadis Dan Surah Dalam Al-Quran Menggunakan Suara'. Dengan mengimplementasikan algoritme shazam untuk mengidentifikasi Hadis dan surah dalam al-Qur'an menggunakan suara diharapkan dapat membantu seseorang yang ingin mengetahui informasi tentang ayat yang sedang didengarkan atau tentang Hadis yang sedang didengarkan.

1.2 Rumusan masalah

Dari uraian latar belakang di atas peneliti merumuskan beberapa masalah sebagai berikut:

1. Bagaimana mengimplementasikan algoritme shazam untuk identifikasi Hadis dan surah dalam al-Qur'an?
2. Bagaimana menghitung akurasi dari hasil identifikasi yang dilakukan dengan mengunakan algoritme shazam?

1.3 Tujuan

Tujuan yang ingin dicapai oleh penulis adalah:

1. Mengimplementasikan algoritme shazam untuk identifikasi Hadis dan surah dalam al-Qur'an.
2. Mengetahui tingkat akurasi dari hasil identifikasi yang dilakukan dengan menggunakan algoritme shazam.

1.4 Manfaat

Manfaat yang ingin dicapai oleh peneliti:

1. Untuk membantu *user* dalam menemukan informasi surah dalam Al-Qur'an dari potongan ayat yang dia dengarkan atau dia punya.
2. Untuk membantu *user* dalam menemukan data detail suatu Hadis dari potongan Hadis yang dia didengarkan atau dia punya.

1.5 Batasan masalah

Batasan masalah yang ditetapkan oleh peneliti:

1. Surah yang dipakai sebagai database yaitu surah ke 78 sampai surah ke 114 / Juz Amma.
2. Hadis yang digunakan adalah Hadis Arbain/40 hadis.

3. Ekstensi file suara yang digunakan adalah ekstensi WAV.
4. File uji berupa potongan ayat atau Hadis dari rekaman.
5. File suara yang digunakan sebagai *dataset* adalah file rekaman dari 2 orang, untuk juz amma digunakan rekaman dari Hanan Attaqi dan untuk hadis digunakan rekaman dari Sheikh Yasir Al Failakawi.

1.6 Sistematika pembahasan

Berikut sistematika yang digunakan oleh peneliti dalam mengerjakan skripsi ini:

BAB 1 : PENDAHULUAN

Bab ini berisi tentang penjelasan permasalahan yang diusung, latar belakang yang mendasari dilakukannya penelitian, batasan masalah yang ditentukan, tujuan dan manfaat penelitian dan sistematika penulisan.

BAB 2 : LANDASAN KEPUSTAKAAN

Berisi tentang uraian beberapa penelitian serupa, uraian tentang teori-teori yang dipakai dalam pengerjaan penelitian.

BAB 3 : METODOLOGI

Berisi langkah-langkah yang akan dilakukan oleh peneliti dalam melakukan penelitian serta penjelasan singkat pada setiap langkah yang dilakukan.

BAB 4 : PERANCANGAN

Pada bab perancangan, berisi tentang langkah-langkah pembuatan rancangan dari sistem yang nantinya akan di implementasi pada bab selanjutnya lengkap dengan penjelasan.

BAB 5 : IMPLEMENTASI

Bab 5 berisi tentang proses implementasi sistem yang didasari dari perancangan yang telah dilakukan pada bab sebelumnya.

BAB 6 : PENGUJIAN DAN ANALISIS

Pada bab ini membahas tentang proses pengujian dan proses analisis dari hasil pengujian yang dilakukan oleh peneliti.

BAB 7 : PENUTUP

Bab ini berisi kesimpulan dari hasil pengujian dan saran-saran yang diberikan peneliti untuk pengembangan selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Penelitian serupa menggunakan suara dengan objek al-Qur'an atau Hadis ataupun dengan menggunakan algoritme shazam telah banyak dilakukan. Ahmed & Abdo (2017) pada penelitiannya tentang sistem verifikasi pembacaan al-Qur'an dengan metode Mel Frequency Cepstrum Coefficient (MFCC) untuk ekstraksi fitur dan metode Hidden Markov Model (HMM) sebagai metode untuk pencocokan pola yang didapat. Pada penelitian ini sistem memiliki empat proses penting yaitu: pre-prosesing, ekstraksi fitur, training dan testing, dan klasifikasi fitur. Dari hasil penelitian tersebut peneliti menyimpulkan bahwa sistem akan mampu menghasilkan hasil akurasi yang tinggi, hasil tersebut sangat dipengarui oleh algoritme-algoritme yang digunakan.

Kemudian Razak, et al. (2008) juga menjelaskan pada penelitiannya tentang perbandingan beberapa algortime yang cocok dan mendukung untuk kasus sistem j-QAF. j-QAF yang merupakan singkatan dari (Jawi, Quran, Arab Language and Fardu Ain) merupakan sebuah sistem yang di buat dengan tujuan untuk membantu pemahami tentang jawi, cara membaca al-Qur'an, bahasa Arab dan ibadah wajib bagi seorang muslim. Hasilnya, dari beberapa algoritme yang dapat digunakan untuk sistem verifikasi pembacaan al-Qur'an dipilih algoritme MFCC sebagai ekstraksi fitur dan HMM sebagai algoritme untuk pencocokan pola yang telah didapat.

Dan pada penelitian yang dilakukan oleh Nieuwenhuizen, et al. (2009) melakukan perbandingan penggunaan algoritme shazam dan algoritme haitsma & kalker's dalam mendeteksi iklan suara pada sebuah siaran radio. Hasil penelitian tersebut membuktikan bahwa algoritme shazam lebih optimal dalam kasus mendeteksi iklan suara pada sebuah siaran radio. Berikut beberapa penelitian tersebut dalam representasi Tabel 2.1:

Tabel 2.1 Kajian Pustaka

No	Judul	Metode	Objek	Hasil Penelitian
1	Verification Sistem for Quran Recitation Recordings	Hidden markov Model, MFCC	Al-Qur'an	Pada penelitian ini digunakan algoritme MFCC dan HMM untuk implementasi, hasil akurasi tidak dituliskan oleh penulis, penulis hanya menuliskan bahwa hasil akurasi yang didapatkan sangat besar

Tabel 2.2 Kajian Pustaka (Lanjutan)

2	Quranic Verse Recitation Recognition Module for Support in j-QAF Learning: A Review	Hidden Markov Model (HMM), MFCC, LPC, VQ, LPCC, RNN	Al-Qur'an	- untuk kombinasi MFCC dan HMM nilai akurasi berkisar 85%-92% - untuk kombinasi MFCC dan RNN nilai akurasi berkisar 95.9%-98.6% - untuk kombinasi MFCC dan VQ nilai akurasi berkisar 57%-100% - untuk kombinasi LPC dan VQ/HMM nilai akurasi berkisar 62%-96%
3	The Study and Implementation of Shazam's Audio Fingerprinting Algorithm for Advertisement Identification	Algoritme shazam, Haitsma & Kalker's algoritm	Siaran radio	Diberikan perbandingan hasil pengujian dari 2 metode yang digunakan
4	Implementasi Algoritme Shazam untuk Mengidentifikasi Hadis dan Surah dalam Al-Qur'an Menggunakan Suara (Usulan)	Algoritme shazam	al-Qur'an dan Hadis	Akurasi pengujian penggunaan algoritme shazam dalam identifikasi surah dalam al-Qur'an dan Hadis

Dari Tabel 2.1 dan Tabel 2.2 dapat dilihat bahwa telah banyak penelitian-penelitian yang membahas tentang objek Al-Qur'an dengan menggunakan pengolahan suara dan juga penggunaan algoritme *shazam*. Mulai dari penelitian tentang *Verification Sistem for Quran Recitation Recordings* yang dilakukan oleh Ahmed & Abdo (2017) sampai dengan *The Study and Implementation of Shazam's Audio Fingerprinting Algorithm for Advertisement Identification* yang dilakukan oleh Nieuwenhuizen, et al. (2009).

2.2 Landasan Teori

2.2.1 Al-Qur'an

Al-Qur'an memiliki beberapa definisi dari beberapa ulama, jika dilihat dari sudut pandang bahasa Daulay (2014) menjelaskan dalam jurnalnya bahwa al-Qur'an berasal dari kata *qara'a* yang berarti mengumpulan dan menghimpun, qira'ah yang berarti menghimpun huruf-huruf dan kata-kata dalam suatu ucapan. Kata *Qur'an* merupakan isim masdar dari kata qara'a, qira'atan, qur'anan (Daulay, 2014). Sedangkan jika dilihat dari sudut pandang istilah al-Qur'an adalah Kalam Allah yang merupakan mukjizat bagi Nabi Muhammad SAW, diturunkan lewat perantara malaikat Jibril, yang tertulis pada mashahif, diriwayatkan kepada kita secara mutawatir, yang jika dibaca memiliki nilai sebagai ibadah, di awali dengan surah al-Fatihah dan di tutup dengan surah an-Naas (Daulay, 2014).

Al-Qur'an terdiri dari 114 surah dan 6236 ayat yang susunannya ditentukan oleh Allah SWT dengan cara *tauqifi* (Amri, 2013). Al-Qur'an mengandung semua hal dalam kehidupan seorang muslim, dengan kata lain mempelajari, menghafal dan memahami Al-Qur'an merupakan suatu kunci untuk umat muslim untuk mendapat kehidupan yang bahagia dunia dan akhirat (Ahmed & Abdo, 2017).

Amri (2013) menjelaskan dalam tulisannya bahwa untuk mempelajari kandungan dari al-Qur'an pada ulama telah sepakat bahwa terdapat setidaknya empat metode yang dapat digunakan, yaitu:

1. Metode Tahlily (Analisis ayat per-ayat)

   Metode ini digunakan untuk mengetahui kandungan Al-Qur'an dengan memperhatikan urutan ayat dalam Al-Qur'an (Amri, 2013).

2. Metode Ijmaly (Analisis secara global)

   Metode yang digunakan untuk menjelaskan kandungan Al-Qur'an secara global atau garis besar (Amri, 2013).

3. Metode Muqarin (Analisis perbandingan)

   Metode ini melakukan analisis terhadap ayat-ayat Al-Qur'an dengan membandingan ayat satu dengan ayat lainnya. Batasan dari perbandingan adalah sebatas pada persoalan redaksional, bukan sampai pada makna (Amri, 2013).-

4. Metode Mudhu'i / Tematik (Analisis bertolak dari tema tertentu)

   Metode ini memiliki dua bentuk dalam implementasinya, berikut dua bentuk tersebut (Amri, 2013):

   a. Membahas satu surah tertentu secara menyeluruh, kemudian menjelaskan kandungan apa saja yang terdapat dalam surah tersebut.

   b. Mencari dan menghimpun ayat-ayat yang memiliki kesamaan tema kemudian diberi penjelasan kesimpulan untuk tema tersebut.

Kemudian untuk membaca Al-Qur'an seorang diwajibkan untuk terlebih dahulu mengetahui ilmu-ilmu yang berkaitan dengan Al-Qur'an seperti:

1. Ilmu Tajwid

   Ilmu Tajwid adalah cabang ilmu yang mempelajari bagaimana cara membaca al-Qur'an dengan baik dan benar (Razak, et al., 2008).

2. Makharijul huruf

   Makharijul huruf merupakan ilmu yang mempelajari cara pengucapan huruf hijaiyah dengan benar sesuai dengan tempat keluarnya suara (Wahidah, et al., 2012).

Dalam membaca Al-Qur'an selain ilmu-ilmu di atas ada satu lagi ilmu yang bersifat opsional, yaitu murotal al-Qur'an. Dengan murotal dimungkinkan sebuah cara pembacaan dan cara melafalkan bacaan dalam al-Qur'an bisa berbagai macam (Razak, et al., 2008).

2.2.2 Hadis

Hadis secara bahasa berarti muda, cerita, riwayat, baru, ucapan. Sedangkan menurut istilah berarti semua hal yang didasarkan dari Nabi Muhammad Saw baik berupa perkataaan, perbuatan, dan ketetapan beliau (Rosidin & Mahfudhoh, 2014).

2.2.2.1 Unsur-Unsur Hadis

Menurut Rosidin & Mahfudhoh (2014) dalam buku karangannya terdapat tiga unsur Hadis yaitu:

1. Sanad

   Jika dilihat dari segi bahasa sanad berarti yang menjadi sandaran, tempat bersandar. Kemudian jika dilihat dari sudut pandang ilmu Hadis sanad berarti urutan orang–orang yang menjadi sandaran dari suatu Hadis mulai dari Nabi Muhammad sampai ke perawi (Rosidin & Mahfudhoh, 2014).

2. Matan

   Matan adalah isi dari sebuah hadis itu sendiri yang merupakan sabda Nabi Muhammad. Matan biasanya terletak setelah sanad (Rosidin & Mahfudhoh, 2014).

3. Rawi

   Rawi dalam ilmu Hadis berarti orang yang menyampaikan Hadis dari seorang guru kepada orang lain dan membukukan Hadis tersebut. Contoh dari rawi adalah: Imam Ahmad, Imam Muslim, Imam Bukhari, dan lain-lain (Rosidin & Mahfudhoh, 2014).

2.2.2.2 Macam – Macam Hadis

Pengelompokan Hadis dibedakan menjadi dua cara yaitu berdasarkan kuantitas dan kualitas:

1. Berdasarkan Kuantitas

Yang dimaksud kuantitas disini merupakan jumlah perawi yang membawakan sebuah Hadis. Jika dilihat dari segi kuantitas Hadis dikelompokkan menjadi dua yaitu Hadis Mutawatir dan Hadis Ahaad. Hadis Mutawatir merupakan Hadis yang diriwayatkan oleh lebih dari dua perawi perawi, sedangankan Hadis Ahaad merupakan Hadis yang diriwayatkan oleh satu, dua, atau tiga orang tetapi tidak termasuk dalam Hadis mutawatir (Rosidin & Mahfudhoh, 2014).

2. Berdasarkan Kualitas

Jika dilihat dari segi kualitas, Hadis dibedakan menjadi tiga yaitu Hadis Sahih, Hadis Hasan, dan Hadis Da'if. Hadis sahih adalah Hadis yang para perawi yang meriwayatkannya adalah *adil* dan *dabit*, sanadnya bersambung, dan tidak ada kejanggalan dan cacat dalam matan Hadis. Hadis hasan adalah Hadis yang bersambung sanadnya bersambung, matannya tidak terdapat kejanggalan dan cacat tetapi perawinya tidak begitu memiliki ingatan yang kuat. Kemudian untuk Hadis Da'if adalah Hadis yang tidak memenuhi syarat dari suatu Hadis dikarenakan ada beberapa syarat yang hilang (Rosidin & Mahfudhoh, 2014).

2.2.2.3 Hadis Arbain

M. Tohir Rahman menjelaskan dalam penelitian Rohim (2013) bahwa *Al-Arba'in* adalah sebuah kitab karangan dari Imam Nawawi yang berisi 42 hadis sahih tentang agama. Sedangkan untuk Hadis Arbain adalah 42 Hadis yang terdapat pada kitab *Al-Arba'in*.

2.2.3 Bilangan Kompleks

Bilangan komples merupakan panambahan antara bilangan *real* (*real*) dengan bilangan *imaginer* (*imag*), bilangan *imaginer* diberi tanda *j* didepan bilangannya. Dimana jika j^2 akan menjadi -1 (Irwan, 2015). Dari uraian di atas maka dengan kata lain bilangan *real* bisa dikatakan merupakan bilangan kompleks dengan nilai *imaginer* 0. Bilangan kompleks jika dirumuskan akan terlihat seperti pada Persamaan dengan menggunakan Persamaan 2.1.

$$kompleks = real + j(imag) \quad (2.1)$$

imag = bilangan *imaginer.*

j = tanda yang menunjukkan bahwa *imaginer*.

2.2.3.1 Operasi Bilangan Kompleks

Operasi matematis untuk bilangan kompleks sedikit berbeda dari bilangan *real* biasa, hal itu dikarenakan bilangan kompleks memiliki komponen *imaginer.* Berikut beberapa operasi matematis dari bilangan kompleks:

1. Penjumlahan

Penjumlahan dilakukan dengan menjumlahkan masing-masing komponen pada bilangan kompleks, menjumlahkan komponen *real* dengan komponen *real* dan menjumlahkan komponen *imaginer* dengan komponen *imaginer* (Irwan, 2015). Berikut ilustrasi dari operasi penjumlahan bilangan kompleks:

Misal: penjumlahan dari 3 + *j*2 dan 5 + *j*1

= (3+5) + (*j*2+*j*1)

= 8 + *j*3

2. Pengurangan

Operasi pengurangan dilakukan dengan mengurangi antar masing-masing komponen pada bilangan kompleks, mengurangi komponen real dengan komponen *real* dan mengurangi komponen *imaginer* dengan komponen *imaginer* (Irwan, 2015). Berikut ilustrasi dari operasi pengurangan bilangan kompleks:

Misal: pengurangan dari 3 + *j*2 dan 5 + *j*1

= (3-5) + (*j*2-*j*1)

= -2 + *j*1

3. Perkalian

Operasi perkalian bilangan kompleks diawali dengan melakukan perkalian silang antara dua bilangan kompleks yang akan dikalikan, ubah j^2 menjadi -1 dan kemudian lakukan operasi penambahan bilangan kompleks (Irwan, 2015). Berikut ilustrasi dari operasi perkalian bilangan kompleks:

Misal: perkalian dari 3 + *j*2 dan 5 + *j*1

= 15 + *j*3 + *j*10 +2j^2

= 15 + *j*3 + *j*10 – 2

= (15+(-2)) + (*j*3+*j*10)

= 13 + *j*13

4. Pembagian

Operasi perkalian bilangan kompleks dilakukan dengan mengalikan dengan sekawan dari penyebut, kemudian dilakukan operasi penambahan dan pengurangan bilangan kompleks (Irwan, 2015). Berikut ilustrasi dari operasi pembagian bilangan kompleks:

Misal: pembagian dari 3 + *j*2 dan 5 + *j*1

$$= \frac{3+j2}{5+j1} x \frac{5-j1}{5-j1}$$

$$= \frac{(3+j2)(5-j1)}{5^2-j1^2}$$

$$= \frac{15 + j3 + j10 + 2j^2}{25+1}$$

$$= \frac{(15+(-2))+(j3+j10)}{26} = \frac{13+j13}{26} \; jika \; disederhanakan \; menjadi \; \frac{1+1j}{2}$$

2.2.4 Binary Komplemen Dua

Binary adalah salah satu bentuk bilangan yang sering digunakan pada mesin atau komputer. *Binary* hanya terdiri dari 2 komponen, yaitu 0 dan 1. Dalam binary akan sulit diketahui apakah bilangan yang direpresentasikan ke dalam bentuk binary merupakan bilangan positif atau negatif. Oleh karena itu muncul-lah teori tentang *binary signed.* Dimana digit pertama merepresentasikan tanda dari bilangan tersebut, jika digit pertama 0 maka bilangan tersebut adalah positif, tetapi jika digit pertama 1 maka bilangan tersebut adalah negatif (Hussainl & Parvin, 2016).

Dari banyak jenis *binary signed*, terdapat salah satu jenis yang cukup populer yaitu *binary* komplemen dua atau *two's complement binary.* Jika pada konversi bilangan desimal ke bilangan *binary* biasanya dilakukan dengan membagi bilangan desimal dengan 2 sampai bilangan tersebut tidak bisa dibagi lagi. Berikut contoh dari konversi bilangan desimal ke dalam *binary.*

Misal: cari bilangan *binary* dari 7

Pembagian ke-1 → $\frac{7}{2} = 3 \; sisa \; 1$

Pembagian ke-2 → $\frac{3}{2} = 1 \; sisa \; 1$

Pembagian ke-3 → $\frac{1}{2} = 0 \; sisa \; 1$

Binary dari 7 diambil dari semua sisa pembagian yang dilakukan mulai dari pembagian ke-3 sampai pembagian ke-1, jadi *binary* dari 7 adalah 111. Dan untuk cara melakukan konversi dari *binary* ke desimal dilakukan penjumlahan dari perkalian setiap digit dengan 2^n, dimana *n* menyatakan index dari digit tersebut. Berikut contoh konversi binary ke desimal:

Misal: konversi 111 kedalam desimal

Desimal = $(1x2^2)+(1x2^1)+(1x2^0)$

= 7

Kemudian untuk mengkonversi desimal kedalam *binary* komplemen dua, terdapat langkah-langkah tambahan yang harus dilakukan. Berikut akan dijelaskan langkah-langkah melakukan mengkonversi suatu bilangan desimal kedalam *binary* komplemen dua (Hussainl & Parvin, 2016):

1. Tulis hasil *binary* asli dari angka yang akan dicari, tambahkan 0 pada awal hasil tersebut untuk menandakan bilangan tersebut positif.
2. Ubah setiap digit yang telah didapatkan, dari 0 menjadi 1 dan dari 1 menjadi 0.
3. Tambahkan 1 dari hasil langkah ke 2, dan inilah representasi *binary* komplemen dua dari suatu desimal negatif.

Berikut akan diberikan contoh berdasarkan langkah-langkah di atas:

Misal: cari *binary* komplemen dua dari -17_{10}

Langkah ke-1: $17_{10} = 00010001_2$

Langkah ke-2: Ubah setiap digit: 11101110_2

Langkah ke-3: Tambahkan 1: $11101110+1 = 11101111_2$

Jadi *binary* komplemen dua dari -17 adalah 11101111.

Kemudian Untuk mendapatkan kembali nilai desimal dari suatu binary komplemen dua langkah-langkah tersebut dibalik, mulai dari langkah ke-3 ke langkah ke-1

Dengan contoh 11101111, dikarenakan pada digit pertama adalah 1 maka nilai desimalnya adalah negatif, oleh karena itu digunakan langkah-langkah dari *binary* komplemen dua seperti berikut:

Langkah ke-1: Kurangi 1: $11101111-1 = 11101110_2$

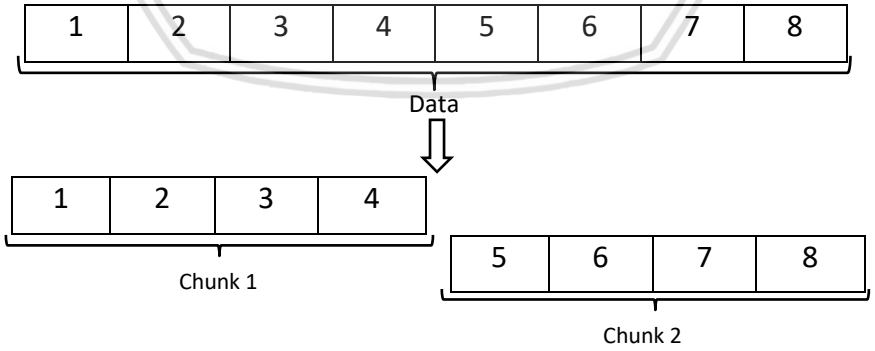
Langkah ke-2: Ubah setiap digit: 00010001_2

Langkah ke-3: $00010001_2 = 17_{10}$

Langkah ke-4: tulis ulang hasil dengan memberikan tanda (-) negatif: -17

2.2.5 Frame Blocking

Frame blocking adalah suatu proses yang dilakukan untuk membagi data suara ke dalam frame-frame (pada penelitian ini peneliti menyebutnya *chunk*) kecil dengan ukuran tertentu (Safaat, 2016). Ukuran dari tiap *chunk* selanjutnya akan disebut sebagai *chunk size*. Ukuran dari tiap *chunk* harus sepanjang mungkin agar bisa melihat semua frekuensi yang ada, akan tetapi harus sependek mungkin agar bisa merepresentasikan informasi waktu dengan baik. Berikut ilustrasi frame blocking dapat dilihat pada Gambar 2.1:



Gambar 2.1 Ilustrasi Frame Blocking

2.2.6 Windowing

Windowing merupakan cara yang digunakan untuk mencegah terjadinya *spectral leakage*, *spectral leakage* adalah munculnya frekuensi yang sebenarnya tidak ada pada data sesungguhnya (Christophe, 2015). *Spectral leakage* pada

umumnya diakibatkan oleh rendahnya jumlah sampling rate atau karena proses frame blocking yang menyebabkan sinyal menjadi discontinue. Terdapat beberapa jenis windowing yang bisa digunakan, berikut adalah beberapa jenis windowing yang bisa digunakan beserta Persamaan 2.2, 2.3 dan 2.4:

1. Hann window

$$w(n) = 0{,}5 - 0{,}5\cos\left(\frac{2\pi n}{N-1}\right) \text{ untuk } n\text{= 0,1,2...}N\text{-1} \quad (2.2)$$

2. Hamming window

$$w(n) = 0{,}54 - 0{,}46\cos\left(\frac{2\pi n}{N-1}\right) \text{ untuk } n\text{= 0,1,2...}N\text{-1} \quad (2.3)$$

3. Backman window

$$w(n) = 0{,}42 - 0{,}5\cos\left(\frac{2\pi n}{N-1}\right) + 0{,}08\cos\left(\frac{4\pi n}{N-1}\right) \text{ untuk } n\text{= 0,1,2...}N\text{-1} \quad (2.4)$$

Pada Persamaan 2.1, 2.2 dan 2.3 *w(n)* merupakan hasil dari windowing, *N* adalah jumlah data dan *n* adalah indeks dari data.

2.2.7 Fast Fourier Transform

Fast Fourier Transform(FFT) adalah salah satu jenis dari *fourier transform,* lebih tepatnya FFT adalah pengembangan dari DFT (*Discrete Fourier Transform)* dengan konsep menurunkan nilai kompleksitas dari DFT itu sendiri. Jika pada DFT memiliki kompleksitas $O(N^2)$ maka untuk FFT memiliki kompleksitas menjadi hanya $O(N\ Log\ N)$ dimana *N* adalah besar dari data. Fungsi utama dari FFT dan DFT adalah mengubah data dari domain waktu ke dalam data dalam domain frekuensi. Data dalam domain frekuensi ini sangat penting dalam proses analisis suara. Persamaan FFT didapatkan dari memecah Persamaan 2.5 yang merupakan persamaan DFT:

$$X(m) = \sum_{n=0}^{N-1} x(n)\, e^{\frac{-j2\pi nm}{N}} \text{ untuk } m \text{ = 0,1,2... } N\text{-1} \quad (2.5)$$

X(m) = data dalam domain frekuensi

x(n) = data dalam domain waktu

n = indeks data domain waktu

m = indeks data domain frekuensi

j = penanda bahwa imajiner

e = eksponensial

π = 180 derajat /3,14 rad

Dari persamaan 2.5 pencarian *X(m)* dibagi menjadi *X(m)* untuk genap dan untuk ganjil sehingga Persamaan 2.5 dapat dituliskan ulang menjadi Persamaan 2.6

$$X(m) = \sum_{n=0}^{N/2-1} x(2n)\, e^{\frac{-j2\pi(2n)m}{N}} + \sum_{n=0}^{N/2-1} x(2n+1)\, e^{\frac{-j2\pi(2n+1)m}{N}}$$

$$= \sum_{n=0}^{N/2-1} x(2n)\, e^{\frac{-j2\pi(2n)m}{N}} + e^{\frac{-j2\pi m}{N}} \sum_{n=0}^{N/2-1} x(2n+1)\, e^{\frac{-j2\pi(2n)m}{N}} \quad (2.6)$$

Persamaan 2.6 biasa dituliskan menjadi

$$X(m) = E_m + e^{\frac{-j2\pi m}{N}} O_m \tag{2.7}$$

Dimana E_m biasa disebut sebagai even dan O_m biasa disebut sebagai odd. Kemudian dengan memanfaatkan periodesasi dari fungsi DFT maka kita dapat tahu bahwa

$$E_{m+N/2} = E_m \text{ dan } O_{m+N/2} = O_m$$

Dan kemudian dapat $e^{\frac{-j2\pi(m+\frac{N}{2})}{N}}$ ditulis ulang menjadi

$$\begin{aligned} e^{\frac{-j2\pi(m+\frac{N}{2})}{N}} &= e^{\frac{-j2\pi m}{N} - \pi j} \\ &= e^{-j\pi} e^{-j2\pi m/N} \\ &= (\cos(\pi) + j\sin(\pi))\, e^{-j2\pi m/N} \\ &= (-1 + j0)\, e^{-j2\pi m/N} \\ &= -e^{-j2\pi m/N} \end{aligned}$$

Dari hasil tersebut kita dapat menulis ulang Persamaan 2.6 menjadi

$$X(m) = E_m + e^{\frac{-j2\pi m}{N}} O_m \tag{2.8}$$

$$X(m + N/2) = E_m - e^{\frac{-j2\pi m}{N}} O_m \tag{2.9}$$

Kemudian untuk nilai eksponen jika diubah menjad trigonometri maka Persamaan 2.8 dan 2.9 akan menjadi

$$X(m) = E_m + (\cos\left(\frac{-2\pi m}{N}\right) - j\sin\left(\frac{-2\pi m}{N}\right))O_m \tag{2.10}$$

$$X(m + N/2) = E_m - (\cos\left(\frac{-2\pi m}{N}\right) - j\sin\left(\frac{-2\pi m}{N}\right))O_m \tag{2.11}$$

Perhitungan dengan menggunakan Persamaan 2.8 dan 2.9 merupakan FFT radix-2, secara sederhana untuk menghitung FFT dengan mengunakan Persamaan 2.10 dan 2.11 memiliki langkah-langkah sebagai berikut:

1. Bagi data menjadi 2, yaitu untuk indeks ganjil dan genap.
2. Ulangi langkah 1 sampai data menjadi berpasang-pasangan indeks ganjil dan genap sebanyak *N*/2 pasang.
3. Hitung FFT untuk setiap pasang dan gabungkan pasangan pertama dengan pasangan setelahnya.
4. Ulangi langkah ke-3 sampai data menjadi data berukuran *N* kembali.

2.2.8 Algoritme Shazam

Algoritme shazam merupakan algoritme yang dibuat dan digunakan oleh Shazam Entertainment, Ltd. algoritme ini digunakan untuk melakukan pengenalan suara sampai pencocokan pola suara *input* dengan *database* yang telah disiapkan

oleh Shazam, algoritme ini dipublikasikan oleh Avery Li-Chun Wang lewat *International Symposium on Music Information Retrieval* pada tahun 2003. Namun yang menjadi sumber utama dari algoritme shazam ini adalah tulisan dari (Christophe, 2015). Dalam tulisannya, Christophe (2015) berhasil menjelaskan secara detail proses algoritme shazam yang tentunya mengacu pada tulisan dari Avery Li-Chun Wang.

2.2.8.1 Cara kerja Algoritme

Untuk melakukan penanganan sampai pengenalan suara, algoritme shazam melakukan beberapa langkah-langkah atau mekanisme, berikut mekanisme yang dilakukan oleh algoritme shazam Christophe (2015) dapat dilihat pada Gambar 2.2:

Gambar 2.2 Diagram Alir Algorime Shazam

1. Ekstraksi Fitur

   Pada tulisannya Christophe (2015) mengimplementasikan FFT untuk mendapatkan fitur dari sebuah file suara. Fitur yang diambil dari hasil FFT adalah point-point frekuensi dengan nilai *magnitude* terbesar. Dalam penjelasannya, sebelum melakukan perhitungan FFT terlebih dahulu Christophe (2015) melakukan *frame blocking* dan *windowing.* Proses *windowing* dan FFT dilakukan untuk setiap *chunk* yang dihasilkan dari proses *frame blocking.* Hasil dari proses FFT adalah koefisien *fourier* dengan nilai bilangan kompleks yang terdiri dari bilangan real dan imaginer. *Koefisien fourier* ini merupakan *koefisien fourier* untuk frekuensi ke-0 sampai frekuensi ke-*N*, dimana *N* adalah *chunk size* - 1. Dari hasil FFT tersebut kemudian akan dicari *magnitude* tertinggi untuk setiap *range*. Nilai *magnitude* dihitung dari *koefisien fourier* yang dihasilkan dengan menggunakan Persamaan 2.12 (Christophe, 2015).

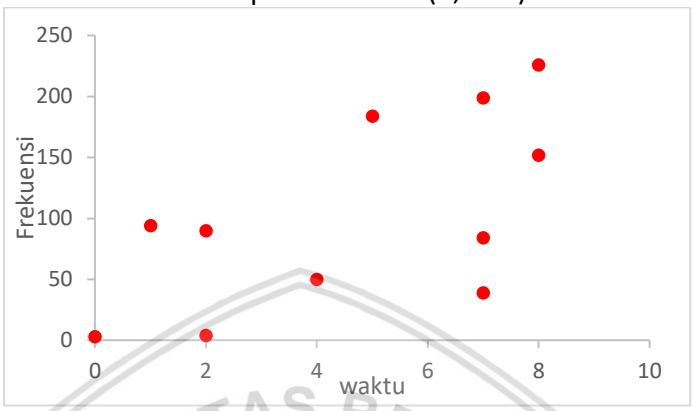
   $$magnitude = \sqrt{real^2 + imag^2} \quad (2.12)$$

   Dimana *real* adalah bilangan real, dan imag adalah bilangan imaginer dari koefisien fourier yang didapatkan.

   Range disini adalah rentang dari frekuensi yang akan dicari nilai *magnitude* terbesarnya. Misalkan data hasil FFT pada sebuah *chunk* adalah frekuensi ke-0 sampai frekuensi ke-100 dan *range* yang digunakan adalah 20 maka untuk setiap 20 frekuensi akan dicari frekuensi dengan nilai *magnitude* terbesar. Maka akan dihasilkan 5 frekuensi dengan nilai magnitude terbesar dari sebuah *chunk*.

2. Pembentukan *hash*/pengalamatan

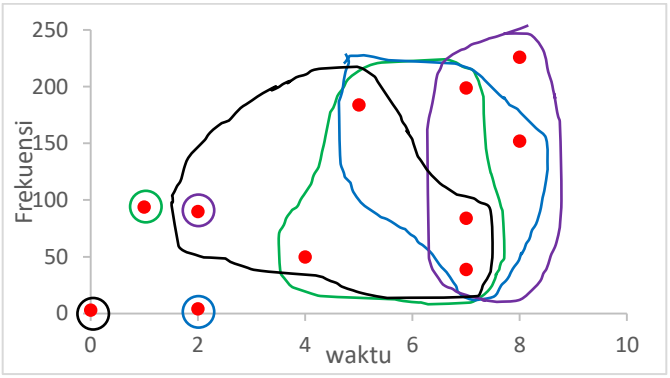
Gambar 2.3 merupakan representasi dari hasil ektraksi fitur dalam bentuk grafik 2 dimensi atau disebut sebagai *constellation map.* Gambar 2.3 didapatkan dari hasil point-point frekuensi dengan *magnitude* terbesar untuk semua *chunk* yang ada. Pada Gambar 2.3, yang dinamakan sebagai waktu adalah indeks dari *chunk*. misalkan sebuah data dengan jumlah *chunk* 10, maka pada data tersebut terdapat 10 waktu (0,1...9).



Gambar 2.3 *Constellation Map*

Dari Gambar 2.3 akan digunakan untuk melakukan pembentukan *hash*. *Hash* disini adalah pasangan *key* dan *value* yang merupakan identitas dari setiap point. Pembentukan *hash* diawali dengan mencari target *zone* dan *anchor point*. Menurut Christophe (2015) dalam tulisannya, yang pertama kali dicari adalah target *zone* dimana target *zone* adalah kumpulan dari beberapa point yang nantinya akan mengacu pada 1 point yang disebut *anchor point.* Aturan pembentukan target *zone* menurut Christophe (2015) adalah kumpulan dari 5 *point* yang berdekatan. Dan untuk *anchor point* adalah *point* ke-3 sebelum point pertama pada sebuah target *zone*. Dari aturan tersebut dapat diketahui bahwa untuk 3 *point* pertama tidak masuk kedalam target *zone* manapun, dikarekan 3 point pertama akan digunakan untuk menjadi *anchor point* dari target *zone* ke-1,2 dan 3.

Setelah mendapatkan target *zone* langkah selanjutnya adalah mencari *anchor point* untuk setiap target *zone* yang telah dibuat. Berikut ilustrasi dari target *zone*, *anchor point* dan hubungan antara keduanya.



Gambar 2.4 Ilustrasi Target Zone dan Anchor Point

Pada Gambar 2.4 setiap warna yang digunakan merepresentasikan pasangan *anchor point* dan target *zone*-nya. Titik dengan lingkaran berwarna hitam melambangkan *anchor point* untuk target *zone* dengan warna hitam, titik dengan lingkaran berwarna hijau melambangkan *anchor point* untuk target *zone* dengan warna hijau, titik dengan lingkaran berwarna ungu melambangkan *anchor point* untuk target *zone* dengan warna ungu dan titik dengan lingkaran berwarna biru melambangkan *anchor point* untuk target *zone* dengan warna biru.

Langkah selanjutnya adalah pembentukan hash dengan cara memberikan alamat untuk setiap *point*s pada target *zone*, pengalamatan ini dilakukan dengan menggunakan pasangan <*key*,*value*> dimana yang menjadi key adalah [frekuensi anchor;frekuensi point;perbedaan waktu] dan untuk valuenya ada 2 jenis pengalamatan yang disesuaikan dengan kondisi, jika kondisi sedang melakukan proses indentifikasi maka pengalamatan yang digunakan adalah [waktu dari anchor], jika sedang melakukan proses *training* data, maka akan digunakan pengalamatan [waktu dari anchor;id dari file suara]. Berikut contoh untuk *hash* pada proses identifikasi dan proses *training*:

1. Identifikasi

   Key = [230;21;3], value = [5] → <[230;21;3], [5]>

2. Training

   Key = [23;211;3], value = [2;1] → <[23;211;3], [2;1]>

3. Pencocokan

   Pencocokan dilakukan dengan mencari jumlah *key hash* antara data uji dengan *key hash* pada setiap data yang terdapat pada *dataset*. Selain itu, juga dicari jumlah kecocokan target *zone* untuk setiap target *zone* yang terbentuk pada data uji (Christophe, 2015). Dari aturan tersebut, sebuah data dalam *dataset* disebut sebagai hasil identifikasi ketika:

   1. Data memiliki jumlah kecocokan target *zone* tebanyak.
   2. Data memiliki jumlah kecocokan *key hash* terbanyak.

   Dari dua kriteria tersebut, kriteria utama yang pakai untuk menentukan hasil dari identifikasi adalah kriteria ke-1, jika keriteria ke-1 tidak dapat dipenuhi baru akan digunakan krit-eria ke-2 sebagai acuan untuk menentukan hasil dari identifikasi.

2.2.8.2 Kelebihan Algoritme Shazam

Berikut kelebihan dari algoritme Shazam menurut uraian dari Avery Li-Chun Wang dalam publikasinya lewat *International Symposium on Music Information Retrieval* pada tahun 2003:

1. Mampu mengenali suara dengan cukup baik walaupun mengandung *noise*.
2. Mampu memproses pencocokan data uji dengan *dataset* dengan sangat cepat.

2.2.9 File WAV

File WAV merupakan sebuah standar file audio yang menyimpan digital audio dalam bentuk gelombang, file WAV ini merupakan file yang digunakan oleh Windows, file yang disimpan dalam bentuk file WAV akan berekstensi .wav (wave) (Safaat, 2016). Dari segi kualitas file WAV bisa memiliki kualitas yang beragam, hal tersebut dikarenakan kualitas dari sebuah digital audio ditentukan dari 3 parameter, berikut parameter tersebut (Safaat, 2016):

1. Jumlah *channel*

Secara sederhana *channel* dapat diartikan sebagai jalur keluar dari suatu digital audio. Jumlah *channel* merupakan parameter yang digunakan untuk menentukan apakah sebuah digital audio tergolong *mono* atau *stereo*. Sebuah file digital audio *mono* hanya memiliki 1 *channel*, sedangakan untuk file digital audio *stereo* memiliki 2 atau lebih *channel* (Safaat, 2016).

2. *Sample Rate/Frame Rate*

Sample rate merupakan sebuah variabel yang menyatakan banyak jumlah sample setiap detiknya, cara kerja *sample rate* ini mirip dengan *frame* pada video dimana semakin tinggi *sample rate* yang digunakan maka kualitas dari sebuah digital audio akan semakin bagus. *Sample rate* yang sering digunakan adalah 8000Hz, 11025Hz, 22050Hz, dan 44100Hz (Safaat, 2016).

3. *Bit Depth/Sample Size*

Bitdepth merupakan sebuah ukuran *bit* untuk setiap *sample*. Ukuran ini meliputi 8 *bits*, 16 *bits*, 24 *bits* dan 32 *bits* (Safaat, 2016). Pengaruh *bit depth* terhadap kualitas sebuah digital audio berbanding lurus, dimana semakin besar nilai *bit depth* maka kualitas akan semakin baik.

Dari ketiga komponen di atas, kita dapat mengetahui jumlah dari data yang bisa ada dari sebuah file suara dalam satuan *byte* dengan menggunakan Persamaan 2.13.

$$N = Sr * C * Bd \quad (2.13)$$

N = jumlah data dalam *byte.*

Sr = *sample rate*

C = jumlah *channel*

Bd = *bit depth* (dalam *byte*)

2.2.10 Akurasi

Nilai akurasi adalah nilai yang digunakan untuk menganalisis tingkat keberhasilan suatu sistem dalam menangani masalah. Nilai akurasi diambil dari proses perbandingan antara jumlah data yang diselesaikan benar oleh sistem dengan jumlah data yang diujikan kepada sistem. Perhitungan akurasi dilakukan dengan menggunakan Persamaan 2.14 (Simanjuntak, et al., 2017).

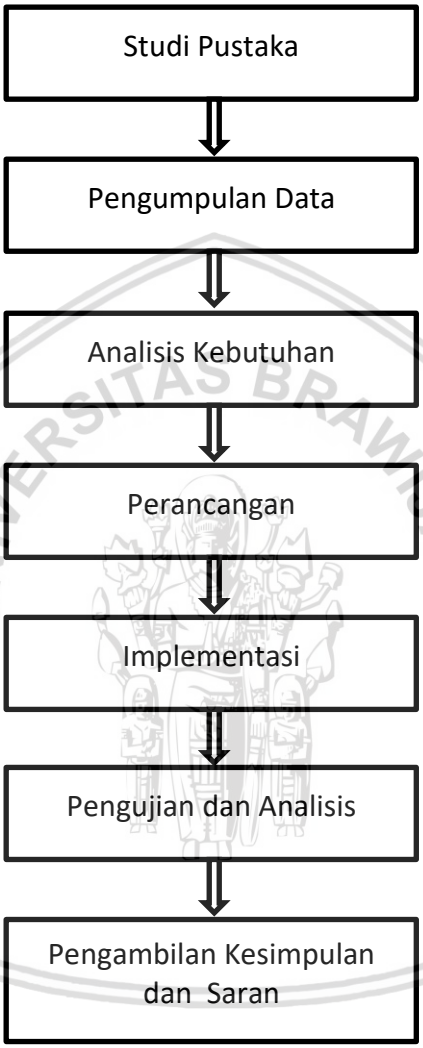
$$Akurasi = \frac{DT}{N} * 100\% \quad (2.14)$$

DT = Jumlah data benar dari hasil sistem

N = Jumlah data yang diujikan pada sistem

BAB 3 METODOLOGI

Pada penelitian ini peneliti menggunakan beberapa tahapan untuk melakukan penelitian, berikut tahapan-tahapan yang dilakukan peneliti dalam proses pengerjaan penelitian ini seperti pada Gambar 3.1



Gambar 3.1. Metodologi

3.1 Studi Pustaka

Pelaksanaan Studi Pustaka dilakukan dengan mengumpulkan data dan kemudian mempelajari referensi-referensi yang ada kaitannya dengan penelitian. Pustaka yang digunakan berupa jurnal, buku dan artikel dengan pustaka utama berupa jurnal terbaru dan jurnal milik Avery Li Chung Wang.

3.2 Pengumpulan Data

Pengumpulan data dilakukan untuk mencari data-data yang akan dipakai dalam penelitian ini, data yang dimaksud adalah *dataset* dan data uji yang berupa

file suara dengan ekstensi wav, pengumpulan data dilakukan dengan cara mencari rekaman yang sudah ada. Cara ini dilakukan dengan mencari rekaman yang sudah ada di internet untuk dijadikan *dataset.* Untuk data uji didapatkan dari memotong file suara *dataset* menjadi potongan kecil (3, 5, 7, 10, 12 dan 15 detik). *Dataset* berupa file suara surah pada juz 30 dan 42 hadis.

3.3 Analisis Kebutuhan

Berikut hardware dan software yang digunakan untuk mengimplementasikan sistem:

1. Hardware yang digunakan, yaitu:
   a. Laptop/komputer

      Processor: Intel(R) Core(TM) i5-7200U CPU @ 2.50Ghz

      Memory (RAM): 4096 MB
2. Software yang digunakan, yaitu:
   a. Sistem Operasi: Windows 10 Pro 64-bit
   b. NetBeans IDE 8.1 sebagai editor
3. Kebutuhan data

   Data yang dibutuhkan pada penelitian ini berupa data file suara surah ke 78 sampai ke 114 atau Juz Amma al-Qur'an dan Hadis Arbain sebagai *dataset*. Kemudian dari dataset tersebut data yang akan digunakan sebagai dataset pada proses pengujian sistem adalah surah An-Naba', surah Al-Fajr, surah Al-Mutaffifin, surah Ad-Duha, surah Al-Buruj, surah An-Naziat, hadis arba'in ke-2: iman, islam, dan ihsan, hadits arba'in ke-24:haramnya berbuat zalim, hadits arba'in ke-10:makan dari rizki yang halal dan hadis arba'in ke-25:bersedekah tidak mesti dengan harta. Data suara surah juz amma didapatkan dari http://media.kangrian.com/2017/06/download-gratis-file-mp3-juz-30-ust-hanan-attaki-googledrive.html yang diakses pada 17 September 2017 dan data suara Hadis Arbain didapatkan dari http://archive.org/download/MurottalMatanHaditsArbainNawawi/MurottalMatanHadistArbainNawawi.zip yang diakses pada 25 November 2017.

3.4 Perancangan

Perancangan sistem merupakan tahap yang akan menjelaskan bagaimana desain dari sistem. Perancangan dilakukan dari hasil tahap pengumpulan data dan analisis kebutuhan.

3.5 Implementasi

Implementasi sistem dilakukan berdasarkan hasil dari perancangan sistem. implementasi sistem pada penelitian ini mengacu kepada batasan perancangan sistem meliputi:

1. Sistem menggunakan pendekatan desktop base *application* dengan bahasa pemrograman Java.
2. Data uji sistem berupa file suara potongan dari suatu surah atau suatu Hadis.

3.6 Pengujian dan Analisis

Pengujian pada penelitian ini dilakukan untuk mengetahui bagaimana kinerja dari sistem yang dibuat. pengujian dilakukan dengan beberapa skenario, untuk penelitian ini skenario yang dipakai adalah sebagai berikut:

1. Pengujian terhadap panjang data uji.
2. Pengujian terhadap *chunk size*.
3. Pengujian terhadap *range*.

3.7 Pengambilan Kesimpulan dan Saran

Proses pengambilan keputusan dilakukan setelah melakukan pengujian dan analisis terhadap hasil pengujian. Hasil analisis tersebut yang akan digunakan sebagai acuan pembuatan kesimpulan. Kemudian untuk kepentingan pengembangan pada penelitian selanjutnya, akan diberikan saran dengan melihat hasil dari pembahasan dan kesimpulan.

BAB 4 PERANCANGAN

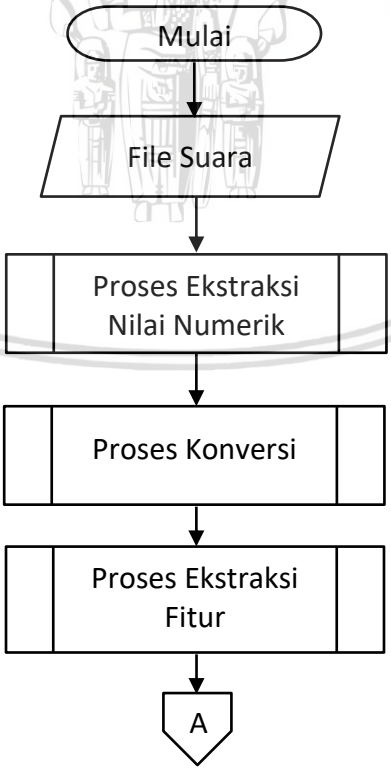
4.1 Deskripsi Permasalahan

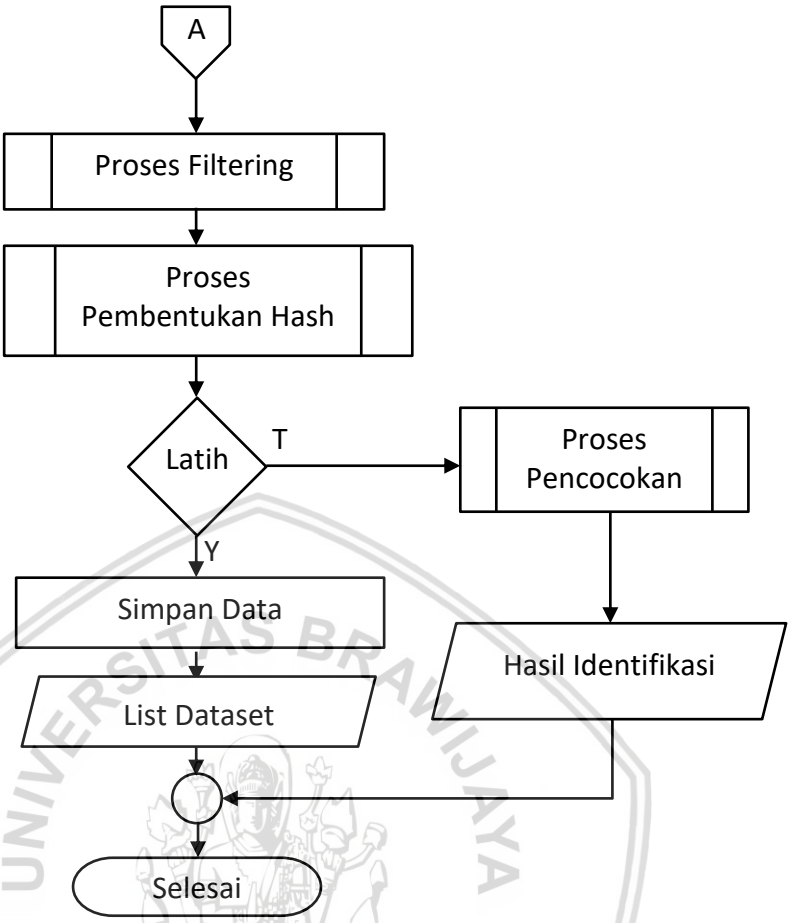
Permasalahan yang akan diselesaikan adalah mengidentifikasi hadis dan surah dalam Al-Qur'an dengan menggunakan *Algoritme Shazam.* Masukan untuk proses identifikasi ini adalah file suara Al-Qur'an juz amma dan hadis arbain yang berekstensi WAV sebagai data latih dan file suara potongan dari suatu surah atau hadis yang juga berekstensi WAV sebagai data uji.

Dari masukan berupa potongan surah atau hadis tersebut kemudian akan diproses sedemikian rupa sehingga nantinya akan mengeluarkan hasil identifikasi terhadap data masukan tersebut berupa nama suah atau hadis dari potongan data suara yang menjadi masukan.

4.2 Penyelesaian Permasalahan Identifikasi Hadis dan Surah Dalam Al-Qur'an Menggunakan *Algoritme Shazam*

Tujuan dari identifikasi hadis dan surah dalam Al-Qur'an ini adalah untuk mengetahui nama dan detail informasi dari potongan surah atau hadis yang menjadi masukan dengan akurasi yang tinggi. Untuk menyelesaikan proses identifikasi ini ada beberapa tahapan yang harus dilakukan. Berikut alir diagram untuk keseluruhan proses indentifikasi hadis dan surah dalam Al-Qur'an:





Gambar 4.1 Diagram Alir Proses Identifikasi menggunakan Algoritme Shazam

Berdasarkan Gambar 4.1, langkah-langkah penyelesaian masalah identifikasi hadis dan surah dalam Al-Qur'an menggunakan algoritme shazam adalah sebagai berikut:

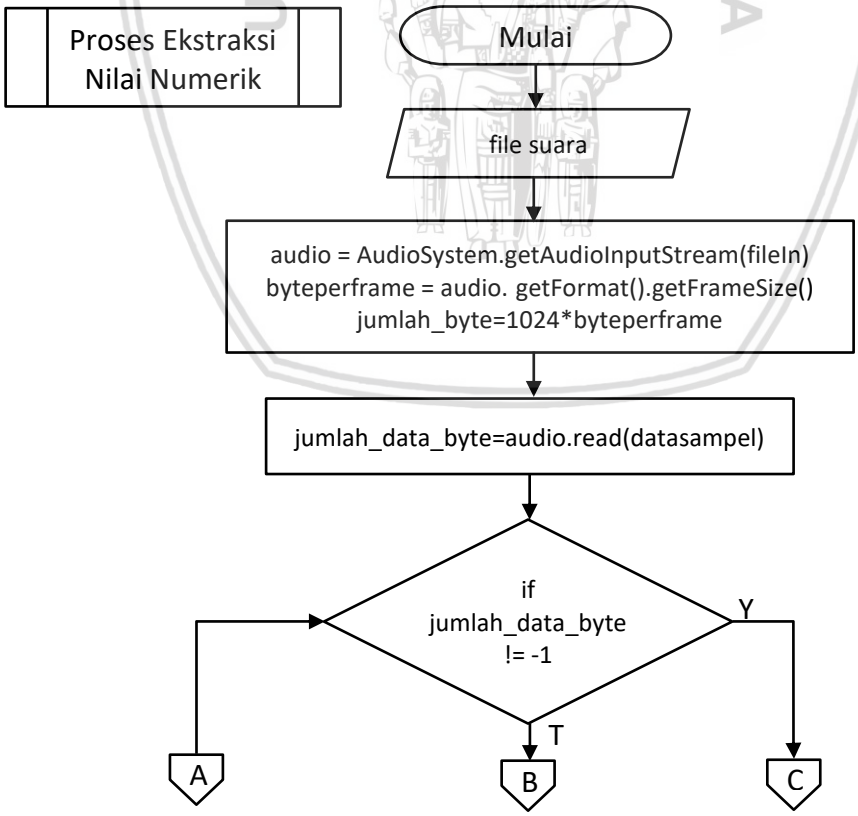
1. Sistem menerima masukan berupa file suara dengan ekstensi WAV.
2. Proses ekstraksi nilai numerik dilakukan dengan mengambil nilai pada byte data yang terdapat pada file suara masukan, Gambar 4.2 merupakan alur dari proses ekstraksi numerik.
3. Proses selanjutnya adalah proses konversi, dimana proses ini akan mengubah data yang awalnya dari data per-byte akan di konversi sehingga data menjadi dalam ukuran sample. Proses ini ditunjukkan pada Gambar 4.5.
4. Kemudian nilai hasil konversi akan diproses untuk mendapatkan fitur. Proses ekstraksi fitur dilakukan dengan menggunakan *fast fourier transform* (FFT). Diagram alir dari ekstraksi fitur ditunjukkan pada Gambar 4.6.
5. Proses selanjutnya adalah prose filtering, dimana proses ini akan melakukan filter mana fitur mana yang dipakai dan fitur mana yang tidak dipakai. Diagram alir dari proses filtering ditunjukkan pada Gambar 4.11.

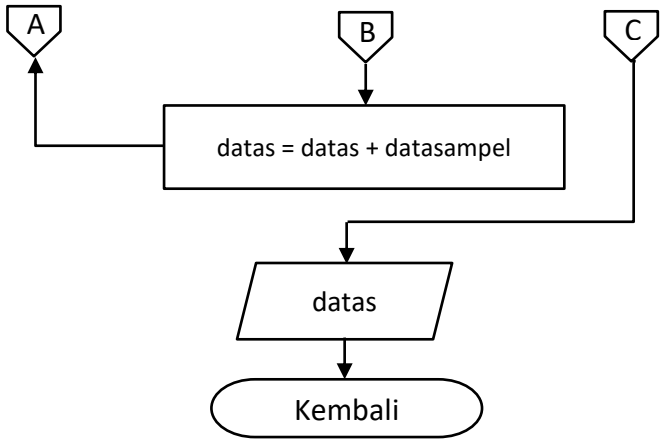
6. Kemudian akan dilanjutkan dengan proses pembentukan *hash*, *hash* ini adalah data berupa pasangan <*key*, *value*> yang akan menjadi identitas untuk tiap file suara. Diagram alir dari proses ini ditunjukkan pada Gambar 4.12.
7. Setelah itu akan masuk ke dalam percabangan, jika proses merupakan proses pelatihan makan akan dilanjutkan ke proses simpan data. Data akan disimpan kedalam file dengan ekstensi **.ser.* Jika tidak, maka akan dilanjutkan ke proses pencocokan.
8. Proses pencocokan ini akan melakukan pencocokan antara hash data latih dengan hash data uji sehingga akan dihasilkan sebuah hasil identifikasi. Alur dari proses pencocokan dapat dilihat pada Gambar 4.15.

Pada pembahasan ini peneliti menggunakan sebuah file suara surah Al-Ikhlas dengan durasi 1 detik yang memiliki *sample rate* 44100Hz, *bit depth* 16 bits dan *channel* sebanyak 2. File suara ini akan diproses sesuai Gambar 4.1 dengan memakai skenario pengujian.

4.2.1 Proses Ekstraksi Nilai Numerik

Proses ekstraksi nilai numerik ini berisi tentang proses pengambilan data numerik dari file suara dengan cara mengambil data yang tersimpan pada tiap *byte* sebuah file suara. Alur proses dari ekstraksi nilai numerik ini digambarkan pada Gambar 4.2.





Gambar 4.2 Diagram Alir Proses Ekstraksi Nilai Numerik

Berdasarkan pada Gambar 4.2 langkah-langkah ekstraksi nilai numerik adalah sebagai berikut:

1. Sub proses ekstraksi numerik menerima masukan berupa file suara.
2. Proses pemberian nilai pada variabel *audio* dengan nilai dari method milik *audioStream* milik java, memberi nilai pada variabel *byteperframe* dengan nilai dari method getsizeframe milik java, dan memberi nilai pada variabel *jumlah_byte* dengan nilai 1024 * *byteframe*.
3. Proses selanjutnya adalah pengambilan nilai pada tiap *byte* data dengan menggunakan *method read* yang disediakan oleh java, data akan disimpan pada variabel *datasampel*.
4. Selanjutnya adalah masuk ke dalam percabangan, dimana jika *jumlah_data_byte* != -1 maka akan masuk ke dalam proses menambah data pada variabel *datas* dari *datasampel*. jika *jumlah_data_byte* == 1 maka sub proses akan mengembalikan nilai yang terdapat pada variabel datas.

Proses ekstraksi nilai numerik dari data suara dilakukan dengan menggunakan *library* yang telah disediakan oleh java. Data numerik ini didapatkan dari nilai setiap 1 *byte* pada 1 sample. Dari file 1 detik yang dipakai data yang bisa diambil adalah sejumlah 176400 *byte* data, jumlah itu didapatkan dari proses perkalian antara *sample rate*, *bit depth* dan *channel* seperti ditunjukkan pada Persamaan 2.13. Berikut contoh perhitungan dari jumlah data yang bisa diambil:

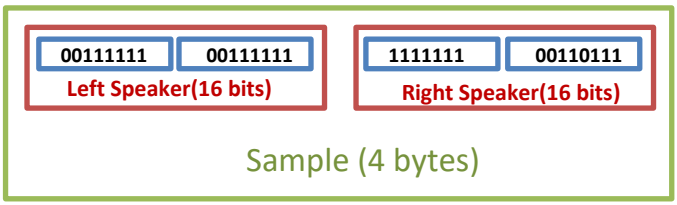
$$N = Sr * C * Bd$$
$$= 44100 * 2 * 2$$
$$= 167400$$

Data numerik pada suara jika di ilustrasikan dalam gambar dengan satuan *sample* akan terlihat seperti pada Gambar 4.3.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	.	.	.	X_{Sr}

Gambar 4.3 Visualisasi Data dalam Satuan *Sample*

X_{Sr} merupakan data pada sample ke-Sr. Dalam satu *sample* menyimpan data sebanyak *Br * C*, pada kasus ini setiap *sample*-nya menyimpan 4 *byte* data yang bisa diambil, yaitu mengalikan 2 *byte bit depth* dengan 2 (jumlah *channel*). Maka, data dalam satu *sample* bisa diilustrasikan seperti pada Gambar 4.4.



Gambar 4.4 Visualisasi Data Satu *Sample*

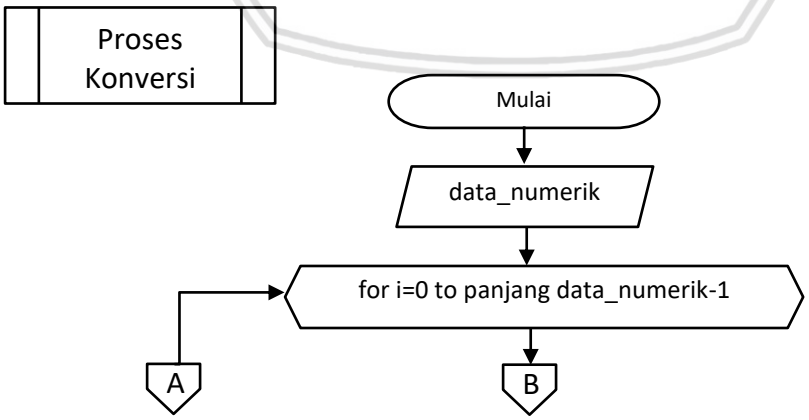
Tabel 4.1 menunjukkan 100 data numerik pertama dari 176400 data yang didapatkan dari file suara surah AL-Ikhlas yang dipakai.

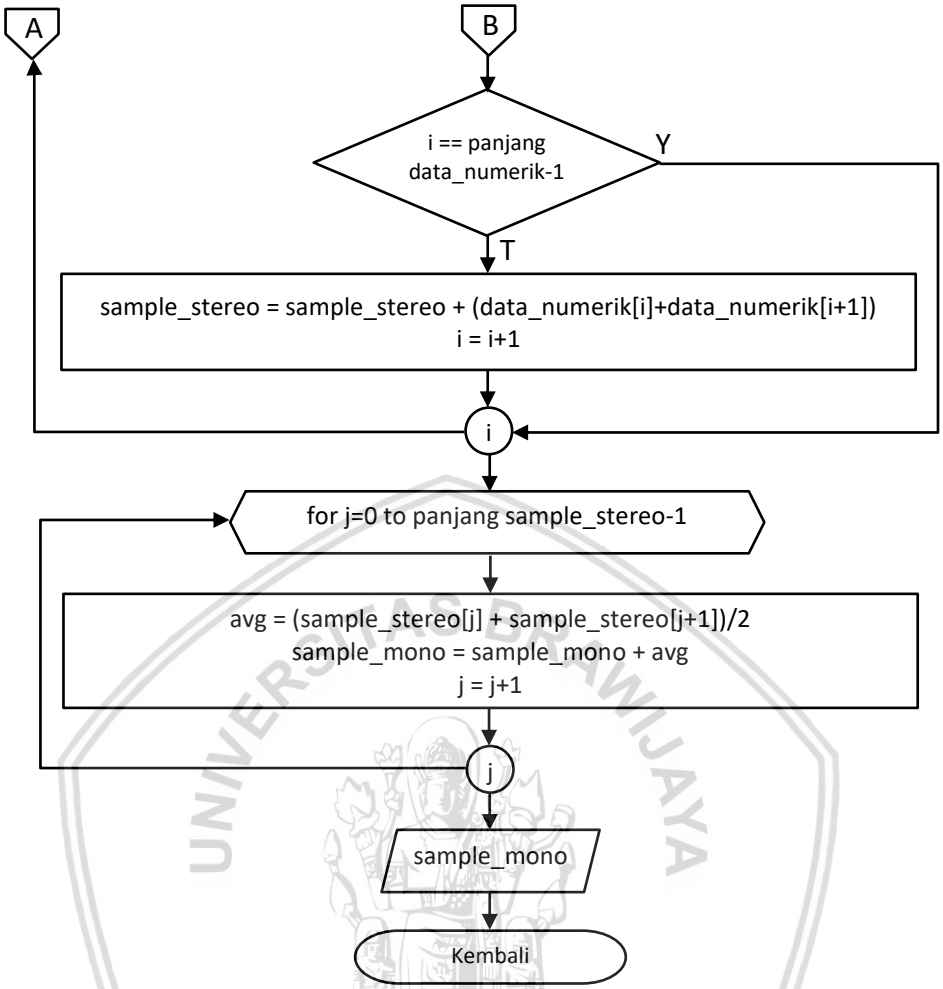
Tabel 4.1 Data Numerik File Suara

100 data pertama dalam byte
72, 13, 70, 13, -7, 14, -6, 14, -105, 16, -105, 16, 87, 18, 84, 18, -45, 19, -41, 19, -109, 20, -110, 20, -59, 20, -61, 20, -128, 20, -125, 20, -113, 19, -115, 19, -97, 18, -98, 18, 56, 18, 61, 18, -57, 17, -64, 17, -64, 16, -58, 16, -124, 15, 127, 15, 77, 14, 81, 14, 14, 13, 10, 13, -57, 11, -54, 11, 84, 10, 82, 10, -85, 8, -84, 8, 23, 7, 23, 7, -105, 5, -106, 5, 62, 4, 63, 4, 18, 3, 19, 3, 94, 1, 92, 1, 66, -1, 69, -1

4.2.2 Proses Konversi

Proses konversi ini dilakukan untuk mengubah setiap *byte* data 1 *sample* data, Diagram alir dari proses ini dapat dilihat pada Gambar 4.5.





Gambar 4.5 Diagram Alir Proses Konversi

Berikut penjelasan dari diagram alir pada Gambar 4.5:

1. Sub proses proses konversi menerima masukan berupa data_numerik.
2. Masuk pada perulangan dari *i* = 0 sampai *i* = panjang *data_numerik*-1 .
3. Masuk pada percabangan, jika *i* = panjang *data_numerik*-1 maka perulangan akan berhenti, dan jika tidak maka akan dilanjutkan ke proses selanjutnya.
4. Menambah nilai pada *sample_stereo* dengan nilai (*data_numerik*[*i*] + *data_numerik*[*i*+1]) dan menambah nilai *i* dengan 1.
5. Akhir perulangan *i*.
6. Masuk pada perulangan dari *j* = 0 sampai *j* = panjang *sample_stereo*-1.
7. Mengisi nilai pada *avg* dengan nilai (*sample_stereo*[*j*] + *sample_stereo* [*j*+1])/2, menambah nilai pada *sample_mono* dengan nilai *avg* dan menambah nilai *j* dengan 1.
8. Akhir perulangan *j.*
9. Mengembalikan nilai dari *sample_mono.*

Pada Gambar 4.5 yang merupakan diagram alir dari proses konversi, masukan dari proses ini adalah *data_numerik* dari hasil proses ekstraksi nilai numerik. Panjang data *data_numerik* ini adalah 167400 *byte* yang diambil dari sebuah file suara dengan *sample rate* 44100, 16 *bit bit depth* dan 2 *channel*/*stereo*. Inti dari proses konversi ini adalah melakukan pengubahan 167400 data dalam *byte* ke dalam 44100 data dalam *sample*.

Dari keterangan mengenai *bit depth* dan jumlah *channel* dari sebuah file suara dapat diketahui bahwa setiap *sample* yang ada pada file suara ini memuat 4 *byte* data, data 1 sample pada file suara ini jika digambarkan maka akan terlihat seperti pada Gambar 4.4. Jadi, dalam 1 buah *sample* yang memuat 4 *byte* data, *byte* pertama dan kedua merupakan data untuk *channel* kiri dan *byte* ke-3 & 4 merupakan data untuk channels kanan. Tiap *channel*-nya mampu menyimpan nilai dari -32,768 sampai 32,767 yang direpresentasikan dalam *signed binary* komplemen dua. Untuk mendapatkan nilai ini akan dilakukan konversi dari bilangan desimal yang tersimpan pada 1 *byte* ke dalam bilangan *binary* dengan aturan dari komplemen dua. Aturan dari binary komplemen dua adalah sebagai berikut:

1. Tulis hasil binary asli dari angka yang akan dicari.
2. Ubah setiap digit yang telah didapatkan, dari 0 menjadi 1 dan dari 1 menjadi 0.
3. Tambahkan 1 dari hasil langkah ke 2, dan inilah representasi *binary* komplemen dua dari suatu desimal negatif.

Berikut ilustrasi proses konversi untuk data ke-5 sampai data ke-8 dari data pada Tabel 4.1:

Data = -7, 14, -6, 14

Data ke-1 = -7_{10}

Karena data adalah negatif maka diperlukan aturan dari binary komplemen dua.

Langkah ke-1: $7_{10} = 00000111_2$

Langkah ke-2: Ubah setiap digit: 11111000_2

Langkah ke-3: Tambahkan 1: $11111000+1 = 11111001_2$

Hasil: 11111001_2

Dengan cara yang sama dilakukan konversi terhadap data ke-2 sampai ke-4 sehingga dihasilkan data sebagai berikut:

Data ke-1 = $-7_{10} \rightarrow 11111001_2$

Data ke-2 = $14_{10} \rightarrow 00001110_2$

Data ke-3 = $-6_{10} \rightarrow 11111010_2$

Data ke-4 = $14_{10} \rightarrow 00001110_2$

Setelah mendapatkan nilai *binary* dari tiap *byte*, maka selanjutnya akan dilakukan penggabungan untuk data ke-1 dengan data ke-2 dan data ke-3 dengan data ke-4 sehingga akan didapatkan 2 *binary* dengan panjang masing-masing 16 digit/ 16 *bits*

$Data_{(1,2)}$ = Data ke-1 & Data ke-2 = 1111100100001110_2

$Data_{(3,4)}$ = Data ke-3 & Data ke-4 = 1111101000001110_2

Kedua *binary* di atas kemudian akan dikonversi ke dalam desimal dengan menggunakan aturan dari *binary* komplemen dua. Aturan sederhananya adalah sebagai berikut:

1. Jika digit pertama 1 maka angka desimalnya adalah negatif, jika digit pertama adalah 0 maka angka desimalnya adalah positif.
2. Untuk angka potisif lakukan konversi binary ke desimal seperti pada umumnya. Contoh: $11001_2 = (1x2^4) + (1x2^3) + (0x2^2) + (0x2^1) + (1x2^0) = 16+8+0+0+1 = 25_{10}$
3. Untuk angka negatif, lakukan langkah-langkah dari aturan *binary* komplemen dua:

   Langkah ke-1: kurangi *binary* dengan 1.

   Langkah ke-2: ubah setiap digit *binary*-nya, 0 menjadi 1 dan 1 menjadi 0.

   Langkah ke-3: konversi *binary* hasil langkah ke-2 kedalam desimal.

   Langkah ke-4: tulis ulang hasil dengan memberikan tanda (-) negatif.

Hasil dari $Data_{(1,2)}$ jika dikonversi adalah sebagai berikut:

$Data_{(1,2)} = 1111100100001110_2$

Karena digit pertama adalah 1 maka hasil desimalnya adalah negatif, oleh karena itu akan dilakukan langkah-langkah seperti pada aturan ke-3:

$Data_{(1,2)} = 1111100100001110_2$

Langkah ke-1: kurangi 1: 1111100100001110 – 1 = 1111100100001101.

Langkah ke-2: ubah setiap digit: 0000011011110010.

Langkah ke-3: konversi binary hasil langkah ke-2 kedalam desimal.

$Data_{(1,2)} = 0000011011110010_2$

$= (0x2^{15}) + (0x2^{14}) + (0x2^{13}) + (0x2^{12}) + (0x2^{11}) + (1x2^{10}) + (1x2^9) + (0x2^8) + (1x2^7) + (1x2^6) + (1x2^5) + (1x2^4) + (0x2^3) + (0x2^2) + (1x2^1) + (0x2^0)$

$= 0+0+0+0+0+1024+512+0+128+64+32+16+0+0+2+0$

$= 1778_{10}$

Langkah ke-4: tulis ulang hasil dengan memberikan tanda (-) negatif.

$Data_{(1,2)} = -1778_{10}$

Dengan cara yang sama dilakukan konversi pada $Data_{(3,4)}$, sehingga dihasilkan data sebagai berikut:

$Data_{(1,2)}$ = -1778_{10}

$Data_{(3,4)}$ = -1522_{10}

Setelah berhasil mendapatkan nilai desimal dari 2 *byte* tiap *channel* maka selanjutnya dilakukan proses konversi dari *stereo channel* ke mono *channel*. Cara yang dilakukan untuk mengkonversi adalah dengan mengambil nilai rata-rata dari data yang terdapat pada masing masing *channel.* Jika $Data_{(1,2)}$ adalah data *channels* sebelah kiri dan $Data_{(3,4)}$ adalah data *channel* kanan maka data mono dapat dihasilkan dengan menghitung rata – rata dari kedua data tersebut.

$Data_{(1,2)}$ = -1778_{10}

$Data_{(3,4)}$ = -1522_{10}

$$data\_mono = \frac{\text{Data}_{(1,2)} + \text{Data}_{(3,4)}}{2}$$

$$= \frac{(-1778) + (-1522)}{2}$$

$$= \frac{-3300}{2} = -1650$$

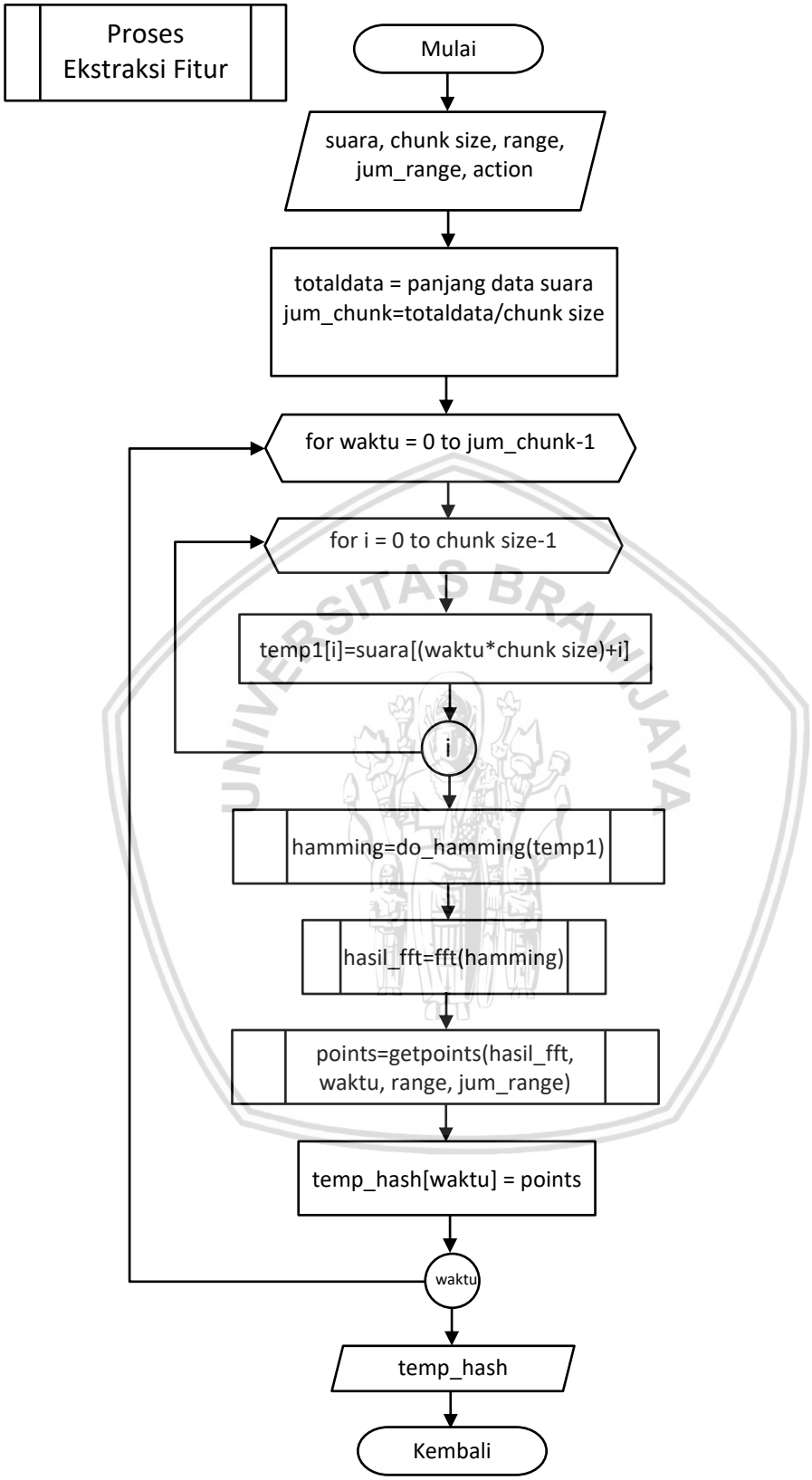
data_mono inilah yang akan digunakan pada proses selanjutnya. Tabel 4.2 adalah 100 data *sample* dari 44100 *sample* yang dihasilkan dari proses konversi:

Tabel 4.2 Data Hasil Proses Konversi

100 Data Pertama Hasil Proses Konversi
18189, -1650, -26864, 21906, -10989, -28012, -15340, -32364, -29165, -24942, 14994, -15471, -15600, 399, 20238, 3085, -14197, 21258, -21624, 5895, -27003, 16004, 4739, 23809, 17535, 21374, 4990, -31235, -3076, -26244, -29317, 13306, 7161, 17783, 2165, 12787, -32143, -1809, -3986, 31213, 17387, 9833, -3738, -22044, -798, 18913, -19361, 13151, 3935, -546.5, -24612, -7846, 9304, -26667, -31533, 22992, -2356, 10059, 6730, 25416, -13242.0, 22854, 26183, 26569, -30133, 16462, -4143, -29099, -1192, -31011, -30366, 18406, 26601, -28052, -13586, -21137, -27665, -531, 18284, 4461, -13329, -26127, -6927, 4978, 24946, -17039, -27793, 28653, 749, -22163, 12143, 13939, 1273, 21117, -19712, 11654, 25485, 2452, 23577, 27293

4.2.3 Proses Ekstraksi Fitur

Proses ekstraksi fitur dilakukan untuk mendapatkan fitur yang nantinya akan dipakai untuk proses selanjutnya. Gambar 4.6 merupakan diagram alir dari proses ekstraksi fitur.



Gambar 4.6 Diagram Alir Proses Ekstraksi Fitur

Dari Gambar 4.6 dapat diketahui bahwa alur dari proses ektraksi fitur adalah sebagai berikut:

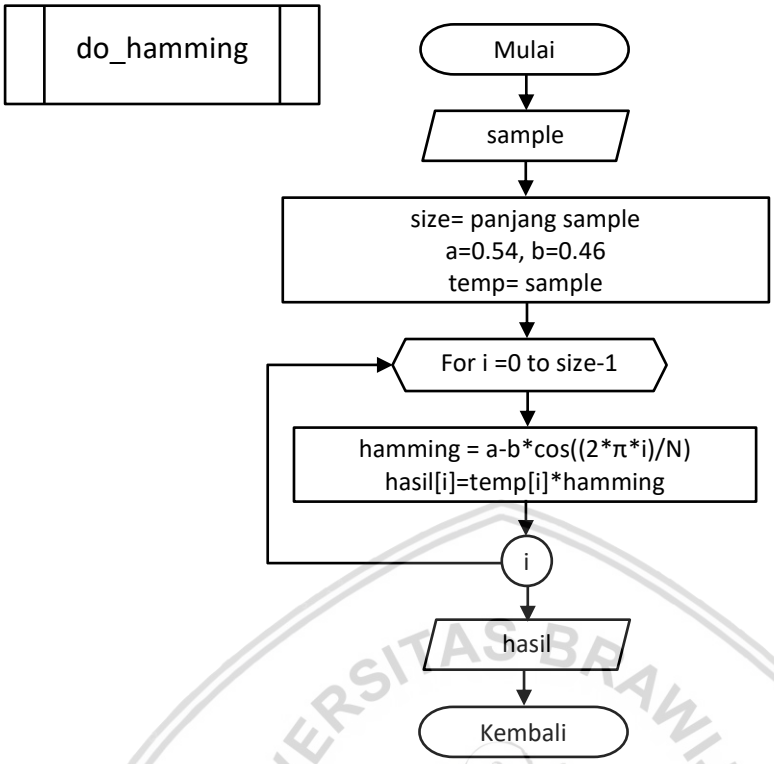
1. Sub proses menerima masukan *suara, chunk size, range, jum_range, action*.
2. Proses pemberian nilai untuk variabel *totaldata, jum_chunk*.
3. Masuk ke perulangan ke-1 dari *waktu* = 0 sampai *waktu* = *jum_chunk*-1
4. Masuk ke perulangan ke-2 dari *i* = 0 sampai *i* = *chunk size*-1
5. Proses memberi nilai *temp1*[*i*] dengan *suara*[(*waktu*chunk size*)+*i*]
6. Akhir perulangan ke-2
7. Proses do_hamming, alur dari proses perhitungan hamming dapat dilihat pada Gambar 4.7.
8. Proses perhitungan FFT, alur dari proses perhitungan FFT dapat dilihat pada Gambar 4.8.
9. Proses getpoints, alur dari proses getpoints dapat dilihat pada Gambar 4.9.
10. Memberi nilai pada variabel *temp_hash*[*waktu*] dengan *points.*
11. Mengembalikan nilai dari *temp_hash.*

Proses ekstraksi fitur diawali dengan membagi data yang menjadi masukan, data tersebut adalah data numerik dari file suara yang di didapatkan pada proses sebelumnya. Pembagian data menjadi potongan-potongan kecil ini bertujuan untuk tetap mengetahui informasi waktu, karena pada proses ini peneliti menggunakan *fast fourier transform* (FFT) yang mana FFT ini akan mengubah data dari domain waktu ke dalam domain frekuensi. Setelah data berada dalam domain frekuensi, akan mengalami kesulitan untuk mengidentifikasi informasi waktu.

Proses pembagian data menjadi potongan-potongan kecil (dalam penelitian ini disebut sebagai *chunk*) ini oleh Safaat (2016) disebut sebagai *frame blocking. Frame blocking* tidak memiliki aturan untuk besar tiap potongan ataupun berapa potongannya. Oleh karena itu ukuran potongan ini akan menjadi salah variabel yang diujikan pada pengujian sistem. Untuk membagi data 1 detik file suara yang digunakan, peneliti menggunakan ukuran sebesar 4096 *sample* per-*chunk* (ukuran per-*chunk* untuk selanjutnya akan disebut sebagai *chunk size*). Dengan begitu akan didapatkan 10 *chunk* dengan tiap *chunk* menyimpan 4096 *sample*. Setelah didapatkan 10 *chunk*, untuk setiap *chunk* tersebut akan dilakukan proses *windowing*, perhitungan FFT dan proses *getpoints.*

4.2.3.1 Proses Windowing

Proses *windowing* merupakan proses yang dilakukan untuk mencegah kesalahan dalam pembacaan data atau kebocoran data karena efek dari proses *frame blocking*. Pada penelitian ini peneliti menggunakan *hamming window* untuk melakukan proses *windowing*. Diagram alir dari proses *windowing* ini dapat dilhat pada Gambar 4.7.



Gambar 4.7 Diagram Alir Proses Hamming Window

Berikut penjelasan dari Gambar 4.7:

1. Sub proses menerima masukan sample.
2. Proses memberikan nilai untuk variabel *a, b, size, temp*.
3. Masuk ke perulangan ke-1 untuk *i* =0 sampai *i* = *size* – 1.
4. Proses perhitungan *hamming*.
5. Akhir perulangan ke-1.
6. Mengembalikan variabel *hasil*.

Proses perhitungan *hamming window* dilakukan dengan menggunakan Persamaan (2.3).

$$w(n) = 0.54 - 0.46 * \cos\left(\frac{2*\pi*n}{(N-1)}\right)$$ untuk *n*= 0,1,2,3... *N*-1

Berikut ilustrasi perhitungan *hamming window* 2 data pertama pada sebuah *chunk* dengan *chunk size* 4096:

$$w(0) = 0.54 - 0.46 * \cos\left(\frac{2*\pi*0}{4095}\right)$$

$$= 0.54 - 0.46 * 1 = 0.08$$

$$w(1) = 0.54 - 0.46 * \cos\left(\frac{2*\pi*1}{4095}\right)$$

$$= 0.54 - 0.46 * 0.9999988228770051 = 0.080000541476577654$$

Kemudian ketika sudah didapatkan hasil dari *hamming*, hasil tersebut akan dikalikan dengan data yang menjadi masukan dan kemudian disimpan dalam variabel *hasil* dengan menggunakan Persamaan 4.1. Berikut ilustrasi perhitungan dari perkalian antara hasil *hamming* dengan data, dimana data ini merupakan 2 data pertama pada *chunk* ke-1 dari keseluruhan data yang digunakan:

Misal:

$hasil(n) = data(n) * w(n)$ dimana $n = 0,1,2....N-1$ (4.1)

$data(0) = 18189$, $data(1) = -1650$

$hasil(0) = data(0) * w(0)$

$= 18189 * 0.08$

$= 1455.12$

$hasil(1) = data(1) * w(1)$

$= -1650 * 0.080000541476577654$

$= -132.00089343635312$

Tabel 4.3 dan 4.4 merupakan 100 data pertama data hasil proses perhitungan *hamming window* pada *chunk* ke-1:

Tabel 4.3 Data Hasil Proses Hamming Window

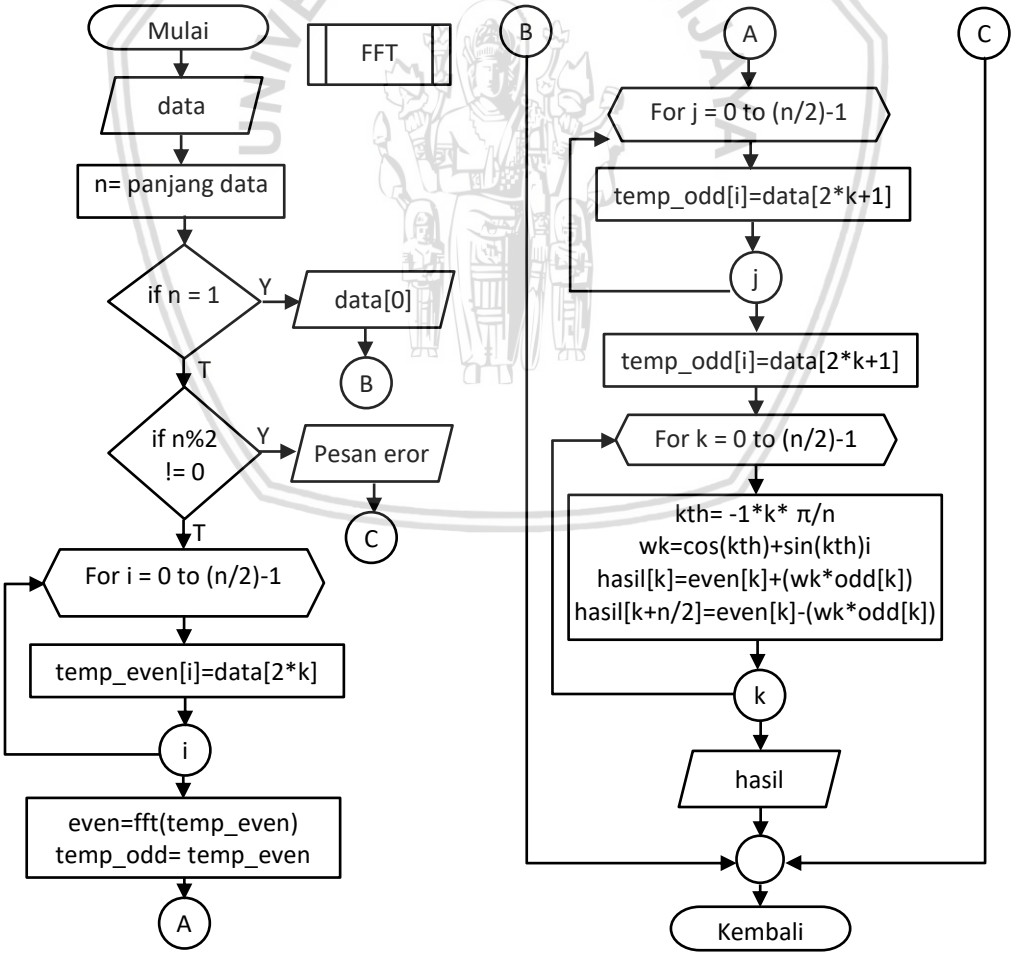
100 Data Pertama Hasil proses hamming window pada *chunk* 1
1455.1200000000003, -132.00089343635312, -2149.1781848728842, 1752.58675410564, -879.2152042976218, -2241.3391942618928, -1227.4990229719792, -2589.978684963734, -2334.2106860288236, -1996.4539290434375, 1200.3318742116985, -1238.6936154167388, -1249.2163388593624, 31.956511103698116, 1621.1877648159395, 247.17583591256874, -1137.727861350472, 1703.966406971631, -1733.7134398093156, 472.75223221491524, -2166.0881390043637, 1284.1412840172281, 380.3618541478983, 1911.5391699577556, 1408.2683831533795, 1717.1525647502028, 401.0262885828281, -2511.1278313736707, -247.38561567094928, -2111.46907063785, -2359.6445022872, 1071.4025927893726, 576.849777274795, 1433.1238278433425, 174.55486804022644, 1031.439693174307, -2593.9907372309235, -146.06061825699268, -321.99574001197345, 2522.738920997455, 1406.0187203487692, 795.5872646225451, -302.609170830896, -1785.5822361027872, -64.67622480429812, 1533.769683324592, -1571.0539422333338, 1067.803384418557, 319.7069389572636, -44.43016212237132, -2002.2607226127382, -638.7245186315242, 757.9352540184562, -2173.8983727953882, -2572.4004277266276, 1876.9977918289107, -192.4781942252365, 822.4050966571052, 550.6507921058186, 2081.153411157669, -1085.1546175040844, 1874.3334280652455, 2149.097161440426, 2182.5755467059257, -2477.417946594656, 1354.5895529299905, -

Tabel 4.4 Data Hasil Proses Hamming Window (Lanjutan)

341.20363170182105, -2398.58834497313, -98.34180903847017, -2560.7507979969423, -2509.770735029142, 1522.6710550288085, 2202.6734989777933, -2325.020275489653, -1127.1209466161904, -1755.2681942367813, -2299.626202573673, -44.18274849260563, 1522.881927432289, 371.9368578115958, -1112.4530252016998, -2182.860111086825, -579.3471873294815, 416.784016054418, 2090.858306884004, -1429.6850152417135, -2334.5829289210296, 2409.4982391367575, 63.05593469528208, -1867.9502947775256, 1024.6141455831494, 1177.520559313176, 107.66455218579225, 1788.0881119056128, -1671.1084498530886, 989.1703170970862, 2165.746722123301, 208.62929419661515, 2008.5376224213567, 2328.0060082633645

4.2.3.2 Proses Perhitungan FFT

Proses perhitungan FFT dilakukan untuk mendapatkan *koefisien fourier* dari file suara yang menjadi masukan, diagram alir dari proses perhitungan FFT dapat dilihat pada Gambar 4.8.



Gambar 4.8 Diagram Alir Perhitungan FFT

Dari Gambar 4.8 dapat dilihat alur dari perhitungan FFT, berikut penjelasan dari alur proses perhitungan FFT:

1. Sub-proses FFT menerima masukan data
2. Langkah selanjutnya, inisialisasi variabel *n* dengan panjang data
3. Kemudian masuk ke percabangan, dimana jika *n* bernilai 1 maka akan mengembalikan data indeks ke 0, jika tidak maka akan lanjut ke proses selanjutnya
4. Prose selanjutnya adalah percabangan lagi dengan syarat jika *n* mod 2 != 0 maka akan menampilkan pesan eror, jika tidak memenuhi syarat maka akan lanjut ke proses berikutnya
5. Masuk ke perulangan ke-1 dengan *i*=0 sampai (*n*/2)-1, akan melakukan pengisian variabel *temp_even* dengan nilai data
6. Selanjutnya akan melakukan rekursif, dan hasil rekursif akan disimpan pada variabel *even*
7. Akhir perulangan ke-1
8. Proses memberi nilai array *temp_odd* dengan *temp_even*
9. Masuk ke perulangan ke-2 dengan *j*=0 sampai (*n*/2)-1, akan melakukan pengisian variabel temp_odd dengan nilai data
10. Akhir perulangan ke-2
11. Selanjutnya akan melakukan rekursif, dan hasil rekursif akan disimpan pada variabel odd
12. Masuk ke perulangan ke-3 dengan *k*=0 sampai (*n*/2)-1
13. Proses mengisi variabel *kth, wk, hasil*[*k*] *dan hasil*[*k*+*n*/2]
14. Akhir perulangan ke-3
15. Mengembalikan nilai hasil

Proses perhitungan FFT diawali dengan mengubah data ke dalam bentuk bilangan kompleks dengan menggunakan Persamaan 2.1. Jadi, jika data pada Tabel 4.3 dan 4.4 diubah menjadi representasi menjadi bilangan kompleks dengan menggunakan Persamaan 2.1 maka akan menjadi seperti pada Tabel 4.5.

Kompleks = real + j(imag)

= 1455.1200000000003*+ j0*

= 1455.1200000000003

Tabel 4.5 Data Kompleks

Data Kompleks
1455.1200000000003, -132.00089343635312, -2149.1781848728842, 1752.58675410564, -879.2152042976218, -2241.3391942618928,- 1227.4990229719792, -2589.978684963734, -2334.2106860288236, - 1996.4539290434375, 1200.3318742116985, -1238.6936154167388, - 1249.2163388593624, 31.956511103698116, 1621.1877648159395, 247.17583591256874, -1137.727861350472, 1703.966406971631, - 1733.7134398093156, 472.75223221491524, -2166.0881390043637, 1284.1412840172281, 380.3618541478983, 1911.5391699577556, 1408.2683831533795, 1717.1525647502028, 401.0262885828281, - 2511.1278313736707, -247.38561567094928, -2111.46907063785, - 2359.6445022872, 1071.4025927893726, 576.849777274795, 1433.1238278433425, 174.55486804022644, 1031.439693174307, - 2593.9907372309235, -146.06061825699268, -321.99574001197345, 2522.738920997455, 1406.0187203487692, 795.5872646225451, - 302.609170830896, -1785.5822361027872, -64.67622480429812, 1533.769683324592, -1571.0539422333338, 1067.803384418557, 319.7069389572636, -44.43016212237132, -2002.2607226127382, - 638.7245186315242, 757.9352540184562, -2173.8983727953882, - 2572.4004277266276, 1876.9977918289107, -192.4781942252365, 822.4050966571052, 550.6507921058186, 2081.153411157669, - 1085.1546175040844, 1874.3334280652455, 2149.097161440426, 2182.5755467059257, -2477.417946594656, 1354.5895529299905, - 341.20363170182105, -2398.58834497313, -98.34180903847017, - 2560.7507979969423, -2509.770735029142, 1522.6710550288085, 2202.6734989777933, -2325.020275489653, -1127.1209466161904, - 1755.2681942367813, -2299.626202573673, -44.18274849260563, 1522.881927432289, 371.9368578115958, -1112.4530252016998, - 2182.860111086825, -579.3471873294815, 416.784016054418, 2090.858306884004, -1429.6850152417135, -2334.5829289210296, 2409.4982391367575, 63.05593469528208, -1867.9502947775256, 1024.6141455831494, 1177.520559313176, 107.66455218579225, 1788.0881119056128, -1671.1084498530886, 989.1703170970862, 2165.746722123301, 208.62929419661515, 2008.5376224213567, 2328.0060082633645

Setelah data sudah direpresentasikan menjadi bilangan kompleks, kemudian akan dilakukan proses perhitungan FFT radix-2 dengan menggunakan Persamaan 2.10 dan 2.11. Berikut ilustrasi langkah-langkah proses perhitungan FFT radix-2 dengan menggunakan 8 data sampel, 8 data sampel ini hanya sebagai contoh untuk menjelaskan langkah-langkah proses perhitungan FFt:

Misal: Data = 2, 3, 4, 5, 6, 3, 4, -5

Ubah data ke dalam bentuk bilangan kompleks.

Data kompleks = 2, 3, 4, 5, 6, 3, 4, -5

2	3	4	5	6	3	4	-5

1. Pisahkan antara yang berindeks genap dan berindeks ganjil.

2	4	6	4

3	5	3	-5

2. Ulangi langkah 1 sampai data terbagi menjadi berpasang-pasangan.

2	6

4	4

3	3

5	-5

3. Hitung FFF dengan Persamaan 2.10 dan 2.11 untuk tiap pasangan.

$$X(m) = E_m + (\cos\left(\frac{-2\pi m}{2}\right) - j\sin\left(\frac{-2\pi m}{2}\right))O_m$$

$$X(m + N/2) = E_m - (\cos\left(\frac{-2\pi m}{2}\right) - j\sin\left(\frac{-2\pi m}{2}\right))O_m$$

- Pasangan ke-1

Data = 2,6

N = 2 (jumlah data)

Even = 2 (even adalah N/2 data pertama)

Odd = 6 (odd adalah N/2 data terakhir)

m = 1,2 ... N-1 (indeks data hasil FFT)

$$X(0) = E_0 + (\cos\left(\frac{-2\pi 0}{2}\right) - j\sin\left(\frac{-2\pi 0}{2}\right))O_0$$
$$= 2 + (\cos(0) - j\sin(0))6$$
$$= 2 + 1(6)$$
$$= 8$$

$$X(0 + 2/2) = E_0 - (\cos\left(\frac{-2\pi 0}{2}\right) - j\sin\left(\frac{-2\pi 0}{2}\right))O_0$$

$$X(1) = 2 - (\cos(0) - j\sin(0))6$$
$$= 2 - 1(6)$$
$$= -4$$

- Pasangan ke-2

Data = 4,4

Even = 4

Odd = 4

N = 2

$$X(0) = E_0 + (\cos\left(\frac{-2\pi 0}{2}\right) - j\sin\left(\frac{-2\pi 0}{2}\right))O_0$$
$$= 4 + (\cos(0) - j\sin(0))4$$
$$= 4 + 1(4)$$
$$= 8$$

$$X(0 + 2/2) = E_0 - (\cos\left(\frac{-2\pi 0}{2}\right) - j\sin\left(\frac{-2\pi 0}{2}\right))O_0$$
$$X(1) = 4 - (\cos(0) - j\sin(0))4$$
$$= 4 - 1(4)$$
$$= 0$$

- Pasangan ke-3

  Data = 3,3

  Even = 3

  Odd = 3

  N = 2

$$X(0) = E_0 + (\cos\left(\frac{-2\pi 0}{2}\right) - j\sin\left(\frac{-2\pi 0}{2}\right))O_0$$
$$= 3 + (\cos(0) - j\sin(0))3$$
$$= 3 + 1(3)$$
$$= 6$$

$$X(0 + 2/2) = E_0 - (\cos\left(\frac{-2\pi 0}{2}\right) - j\sin\left(\frac{-2\pi 0}{2}\right))O_0$$
$$X(1) = 3 - (\cos(0) - j\sin(0))3$$
$$= 3 - 1(3)$$
$$= 0$$

  Hasil = 6, 0

- Pasangan ke-4

  Data = 5,-5

  Even = 5

  Odd = -5

  N = 2

$$X(0) = E_0 + (\cos\left(\frac{-2\pi 0}{2}\right) - j\sin\left(\frac{-2\pi 0}{2}\right))O_0$$

$$= 5 + (\cos(0) - j\sin(0))(-5)$$
$$= 5 + 1(-5)$$
$$= 0$$

$$X(0 + 2/2) = E_0 - (\cos\left(\frac{-2\pi 0}{2}\right) - j\sin\left(\frac{-2\pi 0}{2}\right))O_0$$
$$X(1) = 5 - (\cos(0) - j\sin(0))(-5)$$
$$= 5 - 1(-5)$$
$$= 10$$

Hasil = 0, 10

4. Hitung FFT untuk setiap hasil dari 2 pasangan yang berdekatan.
   - Pasangan ke-1 dan ke-2

   Data = 8, -4, 8, 0

   Even = 8, -4

   Odd = 8, 0

   N = 4

$$X(0) = E_0 + (\cos\left(\frac{-2\pi 0}{4}\right) - j\sin\left(\frac{-2\pi 0}{4}\right))O_0$$
$$= 8 + (\cos(0) - j\sin(0))(8)$$
$$= 8 + 1(8)$$
$$= 16$$

$$X(0 + 4/2) = E_0 - (\cos\left(\frac{-2\pi 0}{4}\right) - j\sin\left(\frac{-2\pi 0}{4}\right))O_0$$
$$X(2) = 8 - (\cos(0) - j\sin(0))(8)$$
$$= 8 - 1(8)$$
$$= 0$$

$$X(1) = E_1 + (\cos\left(\frac{-2\pi 1}{4}\right) - j\sin\left(\frac{-2\pi 1}{4}\right))O_1$$
$$= -4 + (0.25 - 0)(0)$$
$$= -4$$
$$= -4$$

$$X(1 + 4/2) = E_1 - (\cos\left(\frac{-2\pi 1}{4}\right) - j\sin\left(\frac{-2\pi 1}{4}\right))O_1$$

$$X(3) = -4 - (0.25 - j0)(0)$$
$$= -4$$

Hasil = 16, -4, 0, -4

- Pasangan ke-3 dan ke-4

Data = 6, 0, 0, 10

Even = 6, 0

Odd = 0, 10

N = 4

$$X(0) = E_0 + (\cos\left(\frac{-2\pi 0}{4}\right) - j\sin\left(\frac{-2\pi 0}{4}\right))O_0$$
$$= 6 + (\cos(0) - j\sin(0))(0)$$
$$= 6 + 1(0)$$
$$= 6$$

$$X(0 + 4/2) = E_0 - (\cos\left(\frac{-2\pi 0}{4}\right) - j\sin\left(\frac{-2\pi 0}{4}\right))O_0$$
$$X(2) = 6 - (\cos(0) - j\sin(0))(0)$$
$$= 6 - 1(0)$$
$$= 6$$

$$X(1) = E_1 + (\cos\left(\frac{-2\pi 1}{4}\right) - j\sin\left(\frac{-2\pi 1}{4}\right))O_1$$
$$= 0 + ((6.12574E - 17) - j1)(10)$$
$$= 6.12574\text{E} - 16 + \text{j}10$$

$$X(1 + 4/2) = E_1 - (\cos\left(\frac{-2\pi 1}{4}\right) - j\sin\left(\frac{-2\pi 1}{4}\right))O_1$$
$$X(3) = 0 - ((6.12574E - 17) - j1)(10)$$
$$= -(6.12574E - 16) - j10$$

Hasil 6, 6.12574E-16 + j10, 6, -6.12574E-16 – j10

5. Hitung FFT dari hasil gabungan pasangan 1,2 dan pasangan 3,4 sehingga dihasilkan FFT sebanyak *N* data, berikut hasil dari FFT 8 data tersebut.

Hasil = 22, -11.07106781 + *j*7.071067812, 3.67545E-16 + *j*6.0, 3.071067812 + *j*7.071067812, 10, 3.071067812 - *j*7.071067812, -3.67545E-16 - *j*6, -11.071067812 - *j*7.071067812

Dengan langkah-langkah seperti pada ilustrasi dengan menggunakan data sampel di atas, maka akan dilakukan FFT untuk data pada setiap *chunk* yang ada. Tabel 4.6 merupakan 50 data pertama hasil FFT dari data keseluruhan pada *chunk* ke-1 yang mana 100 data awalnya ditunjukkan pada Tabel 4.5:

Tabel 4.6 Hasil Fourier Transform

100 Data Pertama Hasil FFT
26683.667885046685, 108661.40926450328 + *j*465240.64006677864, -1073654.331120927 - *j*572953.4604317981, 1841660.1586953513 + *j*433652.4394148163, -838636.8051285245 - *j*653527.2055507157, 322147.0300782365 + *j*716792.1773081459, -1061584.9179573418 - *j*81364.65124717096, 1467466.9553179261 - *j*564843.8340916476, -1272278.5444388217 + *j*682296.9671511077, 123167.02898792876 - *j*177911.11355412623, 266765.518820553 + *j*126686.37370544014, 311472.014651814 - *j*230615.1888092489, -122536.1321104905 - *j*535622.0517246083, -996881.9065965462 + *j*672770.7885745319, 947527.7644278664 - *j*54042.86298119682, 21208.412722268302 + *j*17028.85700156266, -797155.8239862717 - *j*751708.6168896146, 1201261.0973405843 + *j*315096.46723073884, -801004.4499594638 + *j*776686.5867502568, 7629.755844541709 - *j*1181727.7561691455, -32784.76261937886 + *j*668415.8807624592, -223763.7687753475 - *j*92232.83753409254, 461923.260414473 - *j*14270.66736507928, -397624.54443660384 + *j*438940.4689307632, -34024.169004456824 - *j*414904.3625115652, 363529.37131597684 + *j*327602.86302128667, -192943.1478617881 + *j*316094.9417625364, -117667.69878356197 - *j*544417.8234146059, -513412.7672132976 + *j*47848.78382348044, 736817.2564055105 + *j*287073.16913893074, 113577.71879423551 + *j*166589.4002454966, 362063.1892732442 + *j*135271.6461541909, -391917.05740023917 - *j*886694.9915315902, -156589.75199980952 + *j*382131.35358802817, 132823.71008812584 + *j*531977.8802066557, -191409.1023149368 - *j*372357.3717023367, 231522.86943571316 - *j*54911.66615911339, 639457.2318640206 + *j*342718.4414617985, -1545667.5291511924 - *j*634848.5001775092, 545699.5963874998 + *j*636545.1462851239, 365264.5747919975 - *j*915297.9843863733, -138970.59361708025 + *j*1304769.9514489467, -590921.9634943658 - *j*1092590.5598072964, 622294.5011427109 + *j*533052.6721822976, 167672.86861791904 + *j*282774.08880521206, -749809.6994017089 - *j*551111.8022050193, -53918.92324095662 - *j*192388.1095763986, 67661.07867049499 + *j*279125.37753050757, -122194.37288199455 + *j*66830.98461819175, 537459.5383930779 - *j*398747.55877710244

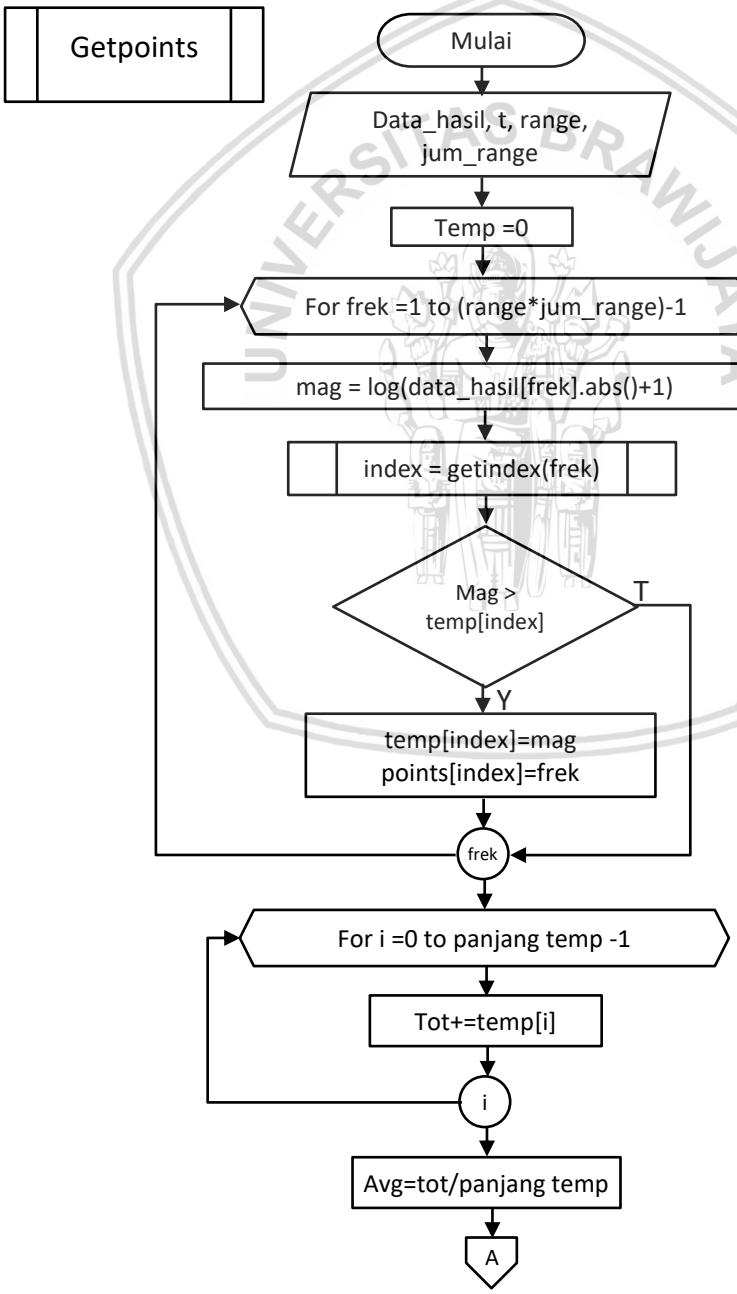
Hasil dari proses perhitungan *fourier* seperti pada Tabel 4.5 adalah data dalam domain frekuensi, yang mana indeks data mengindikasikan frekuensi dan data itu sendiri merupakan sebuah koefisien *fourier*.

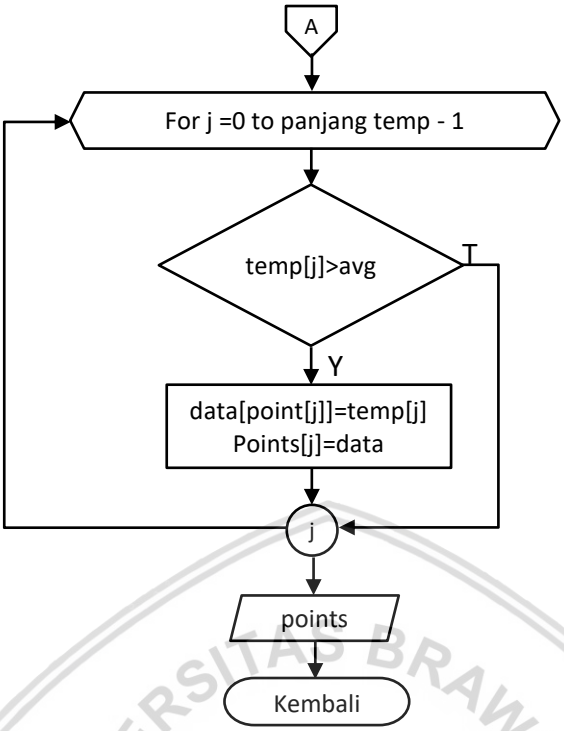
Misal: data(0) = 26683.667885046685

Maka dapat diartikan pada frekuensi ke 0 koefisien *fourier*-nya adalah 26683.667885046685, koefisien inilah yang akan digunakan untuk mencari *magnitude* pada proses getpoints.

4.2.3.3 Proses Getpoints

Proses ini merupakan proses yang dilakukan untuk mencari frekuensi-frekuensi penting dalam suatu file suara. Gambar 4.9 merupakan diagram alir dari proses getpoints.





Gambar 4.9 Diagram Alir Proses Getpoints

Berikut penjelasan dari diagram alir pada Gambar 4.9:

1. Menerima masukan berupa *data_hasil, t, range, jum_range*.
2. Memberi nilai *temp* dengan 0.
3. Masuk perulangan ke-1, untuk *frek* = 1 sampai (*range***jum_range*)-1.
4. Menghitung nilai mag dengan log(*data*[*frek*].abs())
5. Proses getindeks dari *frek*, proses getindeks dapat dilihat pada gambar 4.10.
6. Masuk percabangan, jika *mag* < *temp*[*frek*] maka akan mengisi nilai *points*[*index*]= *frex* dan *temp*[*index*] = *mag*. Jika tidak maka akan menuju proses selanjutnya.
7. Akhir perulangan ke-1.
8. Masuk perulangan ke-2, untuk *i*=0 sampai panjang *temp* -1.
9. Proses mengisi nilai *tot* = *tot*+ *temp*[*i*].
10. Akhir perulangan ke-2.
11. Proses menghitung nilai *avg*.
12. Proses menyimpan *points* dengan *magnitude* di atas rata-rata.
13. Mengembalikan nilai *points*.

Proses getpoints ini adalah proses yang digunakan untuk mencari frekuensi yang mempunyai nilai *magnitude* tertinggi untuk setiap *chunk* pada *range*

frekuensi tertentu. Range yang digunakan merupakan variabel yang akan dilakukan pengujian, pada contoh ini akan digunakan besar range 50. Nilai *magnitude* dicari dengan menggunakan persamaan 2.12. Berikut ilustrasi perhitungan *magnitude* dari 2 data pertama pada Tabel 4.7:

Misal: data(0) = 26683.667885046685,

data(1) = 108661.40926450328+*j*465240.64006677864

$$mag(x) = \sqrt{real^2 + imag^2}$$

Untuk *x* = 0

$$mag(0) = \sqrt{real^2 + imag^2}$$
$$= \sqrt{26683.667885046685^2 + 0^2}$$
$$= \sqrt{712018131.79947182729542462948923 + 0}$$
$$= \sqrt{712018131.79947182729542462948923}$$
$$= 26683.667885046685$$

Untuk *x* = 1

$$mag(1) = \sqrt{real^2 + imag^2}$$
$$= \sqrt{108661.40926450328^2 + j465240.64006677864^2}$$
$$= \sqrt{\begin{matrix}11807301863.347879249805025130758 - \\ 216448853169.74587442831858676025\end{matrix}}$$
$$= 477761.6089987702$$

Hasil *magnitude* yang didapatkan akan dimasukkan ke dalam ruang log, dengan tujuan agar nilai yang dihasilkan memiliki nilai yang lebih kecil dan dapat dengan mudah untuk diamati. Ketentuan dari ruang *log* adalah tidak diperbolehkannya ada *log*(0), oleh karena itu akan dilakukan penambahan 1 sebelum dilakukan operasi *log.* Berikut contoh perhitungan memasukkan hasil *magnitude* ke dalam ruang *log* dari dua data yang telah dihitung nilai *magnitude*-nya:

$$new\_mag(x) = \log(mag(x) + 1)$$

Untuk *x* = 0

$$new\_mag(0) = \log(mag(0) + 1)$$
$$= \log(26683.667885046685 + 1)$$
$$= 10.191844442936302$$

Untuk *x* = 1

$$new\_mag(1) = \log(mag(1) + 1)$$

$$= \log(477761.6089987702 + 1)$$
$$= 13.076869254216348$$

Dengan menggunakan cara yang sama akan dihitung nilai magnitude dalam ruang *log* untuk data keseluruhan hasil FFT pada *chunk* ke-1 yang mana 50 data pertamanya ditunjukkan pada Tabel 4.7. Tabel 4.8 merupakan 100 data pertama dari keseluruhan hasil perhitungan *magnitude* dalam ruang *log* pada *chunk* ke-1:

Tabel 4.7 Data Magnitude Dalam Ruang Log

100 Data Pertama Magnitude Dalam Ruang Log
10.191844442936302, 13.076869254216348, 14.011873283893598, 14.453159884850201, 13.876801476438388, 13.57453018716536, 13.878203090909505, 14.268127667952703, 14.182709035762606, 12.284819269070562, 12.59581460715412, 12.867613313858467, 13.216692736136341, 14.000048194741954, 13.763236464863823, 10.210967615939554, 13.906890223355314, 14.032153308689782, 13.925019048064145, 13.982509814272438, 13.413868759677836, 12.396809143136593, 13.043633209798267, 13.291705697027995, 12.939156865230153, 13.100864151758465, 12.822147964396857, 13.230302094283322, 13.15316148880298, 13.580758485207356, 12.214160991591354, 12.864909553106408, 13.784481851431837, 12.931134120165806, 13.214595839157521, 12.944848925010447, 12.379801748417938, 13.494628398764217, 14.328911868899718, 13.639296747972173, 13.800894279343028, 14.087178283840581, 14.032357145905417, 13.616313402541762, 12.703050415904244, 13.743541113476944, 12.20508209740247, 12.567968863375667, 11.844221094075563, 13.413877401596386, 13.597839955117626, 13.817189687712784, 13.66024521244882, 13.756317449691425, 13.5549713920664, 12.727102642365196, 13.541110753805254, 13.593673640979995, 13.974293254623861, 13.876947730918836, 13.191372596103294, 12.930894323757014, 13.448229548348802, 13.086796421583736, 13.687835146819022, 12.129733462688014, 13.765812911030105, 13.571892723006219, 14.050023377753321, 12.640470343550389, 13.494216011764527, 12.94413227916803, 12.506873261916255, 12.530842854531304, 13.765044023894276, 13.769995738388365, 12.327006034971651, 12.4203924633743, 13.878637180803644, 14.025316861669522, 13.485006835992817, 13.24630069817501, 13.470647426574695, 12.689331005642634, 13.014813881545484, 11.401346470100291, 11.611660242427082, 13.578940652924443, 13.442871513863277, 13.321653025338847, 13.606033260179307, 13.75147218107007, 13.024991233115014, 13.64499541463708, 13.65962423007658, 12.876809997227232, 13.295252269394634, 13.18558757175169, 13.138323902417493, 12.917362847213166, 13.36303419914935

Setelah didapatkan nilai *magnitude* dalam ruang log maka frekuensi dengan nilai *magnitude* tertinggi untuk setiap *range* akan disimpan. Pada pembahasan ini *range* yang digunakan adalah 50. Jadi untuk setiap 50 frekuensi akan didapatkan 1 frekuensi dengan nilai *magnitude* tertinggi. Dari informasi range yang digunakan kita akan mengetahui berapa banyak frekuensi dengan nilai magnitude tertinggi yang akan bisa didapat.

Dengan range 50 dan chunk size 4096 maka jumlah frekuensi dengan nilai magnitude tertinggi yang bisa didapatkan adalah 2048 / 50 = 40, nilai 2048 didapatkan dari setengah *chunk* size, digunakan setengah *chunk* size karena setengah terakhir data hasil FFT merupakan *mirroring* dari setengah data pertama. Setelah mengetahui jumlah *point* frekuensi yang akan dihasilkan, maka untuk mencari setiap *point* tersebut diperlukan proses *getindex* untuk mengetahui suatu frekuensi tersebut masuk kedalam *point* ke berapa. Tabel 4.8 dan 4.9 menunjukkan 40 *point* frekuensi dan magnitude dengan nilai magnitude terbesar dengan menggunakan *range* 50.

Tabel 4.8 40 Point Frekuensi Dan *Magnitude*-nya

Frekuensi	*Magnitude*
3	14.453159884850201
68	14.050023377753321
144	14.071717563014086
159	14.249816179013395
221	14.34907341227222
283	14.253763895178492
332	14.484651313824836
381	14.461691818125004
412	14.242616105318449
499	14.348011911676778
535	14.23840873870423
598	14.138398026968947
632	14.243396662686997
664	14.339384431777049
732	14.137409235699506
778	13.984774850608476
817	14.374450213352585
894	14.049526405134579
940	13.925889486061072
957	14.127576251792862
1005	14.224257925232662
1076	13.99578390836657
1144	14.198163630581778
1181	14.315955224357202
1227	14.136180113103835
1298	14.168903988681969
1339	13.911000779407267

Tabel 4.9 40 Point Frekuensi Dan *Magnitude*-nya (Lanjutan)

1377	14.223007454076777
1436	14.255708050870574
1457	14.163998329476804
1527	14.230454103833544
1596	14.411155717013601
1619	14.47019518255611
1686	14.263320347159938
1731	14.330484081047514
1772	14.039670754468686
1824	14.451123875340661
1897	14.383351204452467
1933	14.264889477295297
1996	14.371864948066078

Kemudian setelah mendapatkan 40 *point* frekuensi, proses selanjutnya adalah proses filtering, proses filtering dilakukan untuk mendapatkan *point* frekuensi yang memiliki nilai *magnitude* di atas rata-rata dari semua *magnitude* yang ada pada sebuah *chunk*. Berikut langkah-langkah filtering:

1. Menghitung nilai rata-rata *magnitude*.
2. Menyimpan frekuensi dengan nilai *magnitude* di atas rata-rata.

Dengan menggunakan langkah-langkah di atas, rata-rata magnitude untuk *chunk* ke-1.

$$rata - rata = \frac{(\sum_{x=0}^{N-1} new\_mag(x))}{N}$$

N = jumlah *point* frekuensi dengan nilai magnitude tertinggi

x = 1,2...*N*-1

$$rata - rata = \frac{(14.453159884850201 + 14.050023377753321 + \cdots + 14.371864948066078)}{40}$$

$$= \frac{569.3332088592023}{40}$$

$$= 14.233330221480056$$

Setelah mendapatkan rata-rata kemudian proses selanjutnya adalah menyimpan *points*= frekuensi beserta *magnitude*-nya yang nilai memiliki nilai *magnitude* di atas rata-rata. Tabel 4.10 dan 4.11 menunjukkan frekuensi beserta *magnitude*-nya yang mana nilai *magnitude*-nya di atas nilai rata-rata.

Tabel 4.10 Frekuensi Dan *Magnitude* Di atas Rata-Rata

Frekuensi	***Magnitude***
3	14.453159884850201
159	14.249816179013395
221	14.34907341227222

Tabel 4.11 Frekuensi Dan *Magnitude* Di atas Rata-Rata (Lanjutan)

283	14.253763895178492
332	14.484651313824836
381	14.461691818125004
412	14.242616105318449
499	14.348011911676778
535	14.23840873870423
632	14.243396662686997
664	14.339384431777049
817	14.374450213352585
1181	14.315955224357202
1436	14.255708050870574
1596	14.411155717013601
1619	14.47019518255611
1686	14.263320347159938
1731	14.330484081047514
1824	14.451123875340661
1897	14.383351204452467
1933	14.264889477295297
1996	14.371864948066078

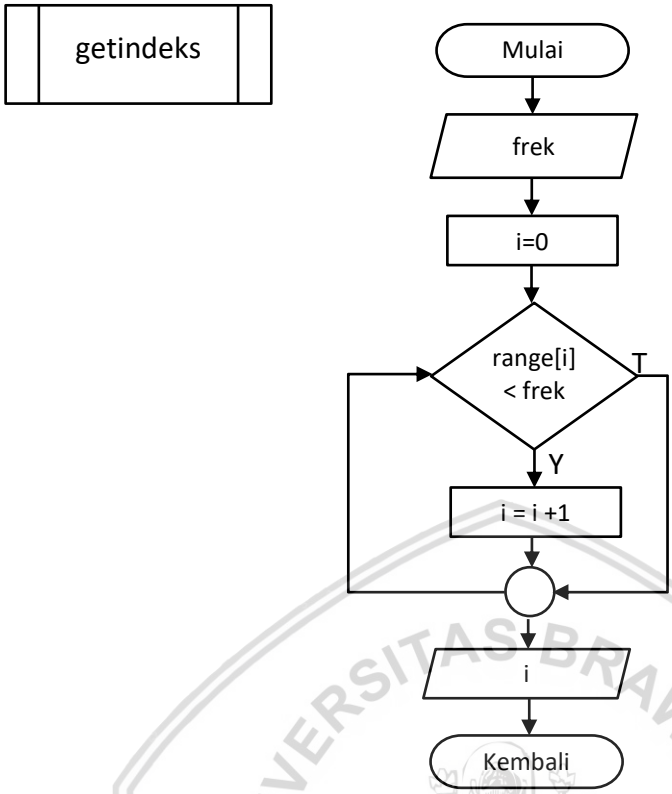
Data pada Tabel 4.10 dan 4.11 inilah yang akan disimpan dalam kombinasi <*key,value*> dimana frekuensinya akan menjadi *key* dan *magnitude*-nya akan menjadi *value*, kombinasi <*key,value*> ini yang nantinya akan dipakai untuk proses selanjutnya. Tabel 4.12 merupakan frekuensi yang lolos proses filtering pada *chunk* ke-1.

Tabel 4.12 Data *Point* Frekuensi Dan *Magnitude*

Data frekuensi beserta *magnitude* hasil filtering pada *chunk* ke-1
{3=14.453159884850201}, {159=14.249816179013395}, {221=14.34907341227222}, {283=14.253763895178492}, {332=14.484651313824836}, {381=14.461691818125004}, {412=14.242616105318449}, {499=14.348011911676778}, {535=14.23840873870423}, {632=14.243396662686997}, {664=14.339384431777049}, {817=14.374450213352585}, {1181=14.315955224357202}, {1436=14.255708050870574}, {1596=14.411155717013601}, {1619=14.47019518255611}, {1686=14.263320347159938}, {1731=14.330484081047514}, {1824=14.451123875340661}, {1897=14.383351204452467}, {1933=14.264889477295297}, {1996=14.371864948066078}

4.2.4.3.1 Proses getindeks

Proses ini bertujuan untuk mendapatkan indeks dari suatu frequensi terhadap range tertentu. Diagram alir dari proses getindeks dapat dilihat pada Gambar 4.10



Gambar 4.10 Diagram Alir Getindeks

Penjelasan dari proses yang terjadi pada Gambar 4.10 adalah sebagai berikut:

1. Menerima masukan berupa *frek.*
2. Memberi nilai *i* dengan 0.
3. Masuk ke perulangan while dengan kondisi jika *range*[*i*] < *freq* maka akan dilanjukan ke langkah selanjutnya.
4. Menambah isi dari variabel *i*.
5. Jika kondisi pada langkah ke-3 tidak terpenuhi maka perulangan selesai.
6. Mengembalikan nilai *i*.

Proses ini bertujuan untuk mencari indeks dari sebuah frekuensi masukan. Proses ini merupakan bagian dari proses *getpoints* dimana tujuan utamanya adalah untuk mengetahui sebuah frekuensi masukan termasuk kedalam 50 (*range*) frekuensi ke berapa. Yang dimaksud sebagai indeks disini adalah sebagai berikut: dengan menggunakan *range* 50, dan *chunk size* 4096, maka akan didapatkan 40 *point* frekuensi dengan *magnitude* terbesar untuk setiap 50 frekuensi. Dimana *point* frekuensi tertinggi untuk 50 frekuensi ke-1 akan menjadi *point* ke-1 dari 40 *point* yang dicari. Kemudian untuk *point* frekuensi tertinggi ke-2 akan menjadi *point* ke-2 dari 40 *point* yang dicari. Hal itu berlaku untuk seterusnya sampai *point* ke-40.

Dengan menggunakan *range* 50, maka Indeks dari sebuah frekuensi dicari dengan melakukan pengecekan frekuensi tersebut berada pada 50 frekuensi ke berapa. Berikut ilustrasi dari *range-range* yang terbuat dari penggunaan range 50:

Tabel 4.13 *Range* frekuensi

50 frekuensi ke-*i*	*Range* frekuensi
1	0-50
2	51-100
3	101-150
4	151-200
5	201-250
6	251-300
.	.
.	.
.	.
40	1951-2000

Dari Tabel 4.13 dapat dilihat bahwa pada frekuensi yang masuk 50 frekuensi ke-1 adalah frekuensi ke-0 sampai ke-50, frekuensi yang masuk 50 frekuensi ke-2 adalah frekuensi ke-51 sampai ke-100 dan seterusnya. Berikut merupakan ilustrasi dari proses getindeks dengan menggunakan masukan 2 data frekuensi pertama dari Tabel 4.9 dan 4.10 dengan mengacu pada aturan yang terdapat pada Tabel 4.14:

frek = 3, 68

untuk frek = 3

jika dilihat dari aturan yang terdapat pada Tabel 4.13 maka frek = 3 termasuk dalam 50 frekuensi ke-1, jadi indeks yang akan dihasilkan untuk frek = 3 adalah 0.

untuk frek = 68

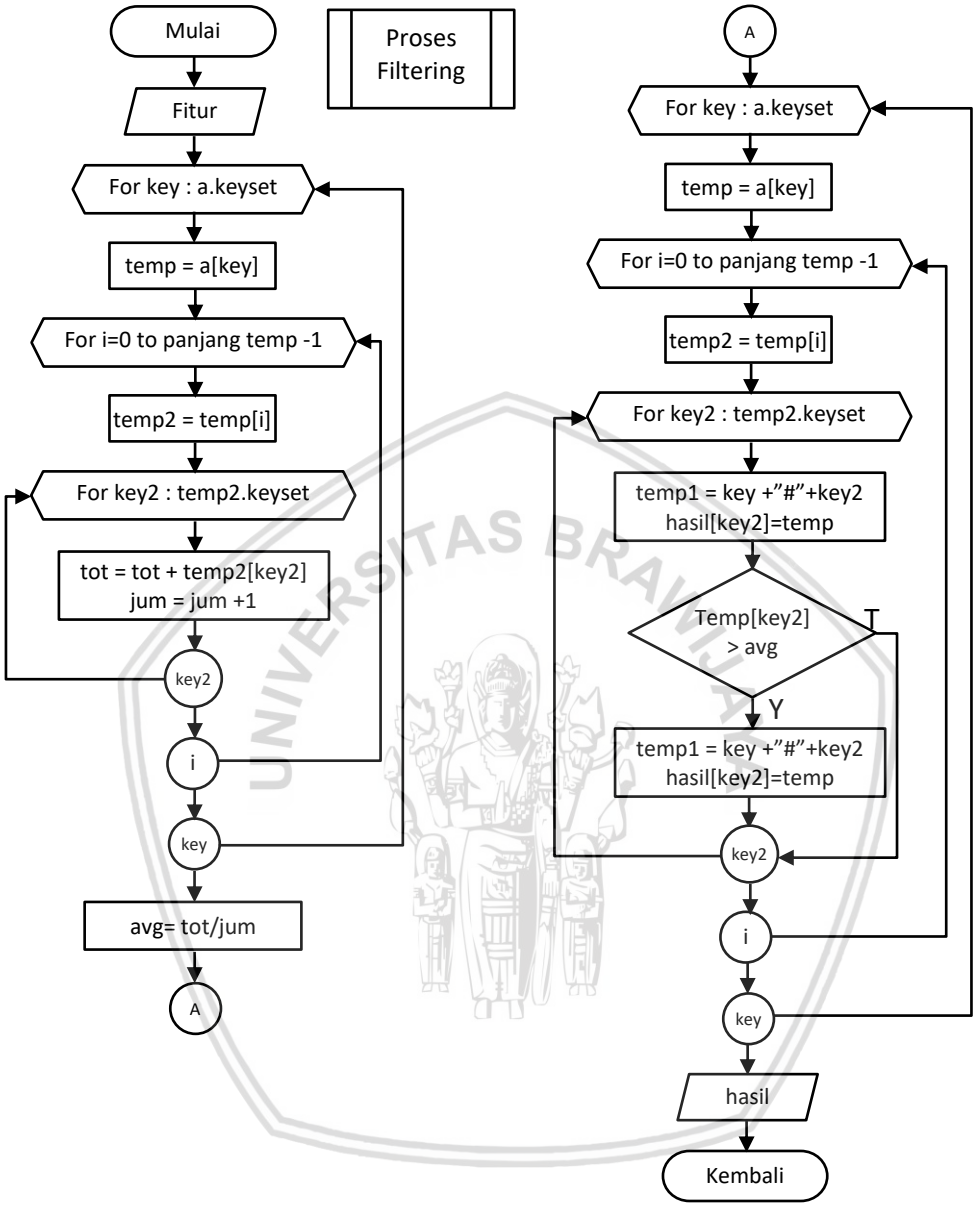
jika dilihat dari aturan yang terdapat pada Tabel 4.13 maka frek = 68 termasuk dalam 50 frekuensi ke-2, jadi indeks yang akan dihasilkan untuk frek = 68 adalah 1.

Dari dua ilustrasi di atas, proses *getindeks* dilakukan untuk semua masukan frekuensi mulai dari 0-4096.

4.2.4 Proses Filtering

Proses filtering merupakan proses yang digunakan untuk melakukan filter dari *point* yang telah didapatkan pada setiap chunk. Filter yang dimaksud adalah

dengan menambil point dengan *magnitude* di atas dari rata-rata. Diagram alir dari prose filtering dapat dilihat pada Gambar 4.11.



Gambar 4.11 Diagram Alir Proses Filtering

Berikut penjelasan dari diagram alir proses filtering pada Gambar 4.11:

1. Proses menerima masukan fitur.
2. Proses menghitung nilai rata-rata dari semua nilai *magnitude* yang ada pada fitur.
3. Proses menyimpan fitur yang memiliki nilai *magnitude* lebih besar dari rata-rata yang didapatkan.

Proses filtering ini bertujuan untuk melakukan filtering terhadap *point* yang telah didapatkan dari proses sebelumnya. Proses filtering ini dilakukan untuk keseluruhan *point* yang ada, tidak terpisah untuk masing-masing *chunk*. Proses filtering dilakukan dengan menyimpan frekuensi dengan nilai *magnitude* di atas dari rata-rata *magnitude* secara keseluruhan.

Tabel 4.14 *Point* Pada Setiap *Chunk*

Chunk ke-*i*	**<key, value>**
0	{3=14.453159884850201} {159=14.249816179013395} {221=14.34907341227222} {283=14.253763895178492} {332=14.484651313824836}
1	{42=14.308964780536432} {94=14.558307204205875} {139=14.279118876409013} {223=14.264164523859511} {427=14.275240429266855}
2	{4=14.443086399539075} {90=14.51325564426334} {123=14.28699085084073} {387=14.4834240481682} {422=14.34392106683155}
3	{192=14.285902804887392} {307=14.337668464291038} {374=14.21574952302697} {493=14.271977192070533} {617=14.23222215994602}
4	{50=14.382829666516633} {238=14.310523837316644} {384=14.36852116264887} {411=14.311489482187643} {661=14.374269705827107}
5	{184=14.351800562838438} {223=14.224393730664476} {284=14.25281112524285} {436=14.236810501521862} {489=14.26592646466421}
6	{65=14.250314648060705} {206=14.259849116517106} {298=14.343964392196824} {311=14.529314631771564} {393=14.557470679013347}
7	{39=14.466921310795128} {84=14.391283457384684}

Tabel 4.15 *Point* Pada Setiap *Chunk* (Lanjutan)

	{199=14.506046823404818} {282=14.57508777412882} {352=14.388564545199216}
8	{41=14.285638199638024} {152=14.379193752481827} {226=14.351020721198852} {557=14.347710496537946} {700=14.315080888392574}
9	{34=14.287587709390587} {145=14.295384646446358} {190=14.271110696021418} {290=14.31886942294809} {360=14.448766650797246}

Tabel 4.14 dan 4.15 merupakan 5 data *point* awal hasil *getpoints* pada semua *chunk* yang ada. Untuk melakukan proses filtering ini terlebih dahulu dilakukan pencarian rata-rata dari keseluruhan *magnitude* yang ada. Dimana *magnitude* merupakan value dari sebuah frekuensi. Berikut ilustrasi perhitungan rata-rata dengan *mag(x)* merupakan nilai magnitude ke-*x* dan *N* adalah jumlah keseluruhan *point* yang ada, pada data yang digunakan nilai *N* =203.

$$rata - rata = \frac{(\sum_{x=0}^{N-1} mag(x))}{N}$$

N = jumlah *point*

x = 1,2...*N*-1

$$rata - rata = \frac{(14.453159884850201 + 14.050023377753321 + \cdots + 14.342365381236005)}{203}$$

$$= \frac{2912.2982891115807}{203}$$

$$= 14.346296990697443$$

Setelah mendapatkan rata-rata, kemudian akan dilakukan filter pada keseluruhan data yang ada. Jika nilai *magnitude* suatu *point* berada di atas rata-rata maka frekuensi-nya akan disimpan beserta dengan indeks *chunk*-nya. Nilai *magnitude* tidak lagi disimpan karena pada proses selanjutnya nilai *magnitude* sudah tidak digunakan. Yang akan digunakan hanyalah frekuensi itu sendiri dan indeks dari *chunk* frekuensi tersebut.

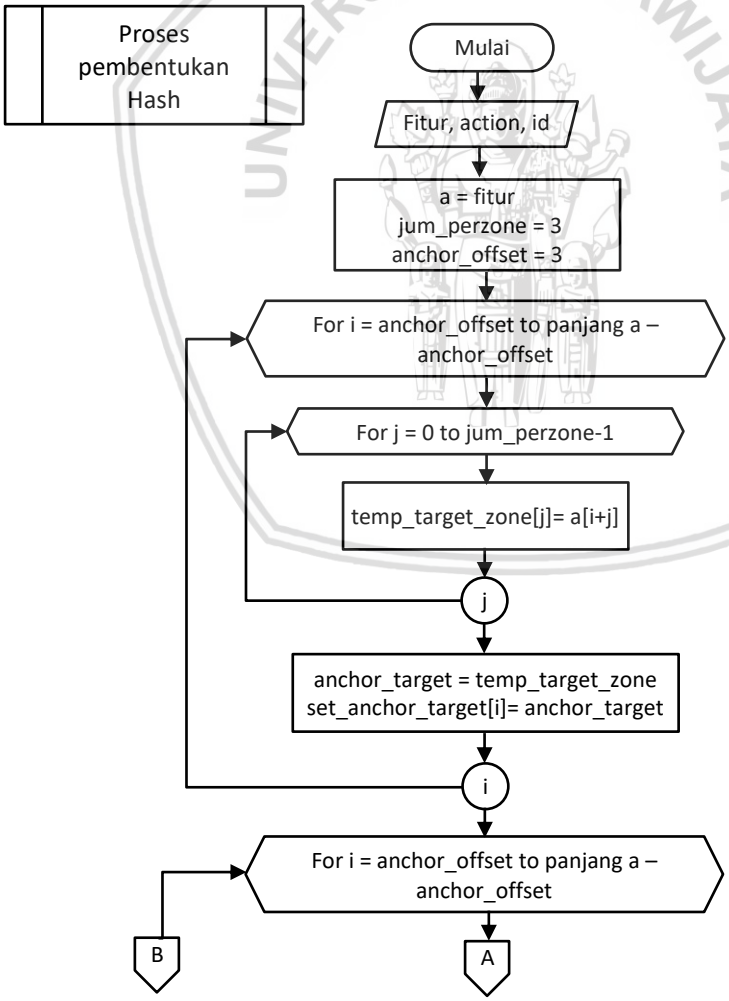
Misalkan frekuensi = 3 pada *chunk* ke-0 memiliki *magnitude* di atas rata-rata, maka nilai yang akan disimpan adalah 3 dan 0, yang mana 3 adalah frekuensinya dan 0 adalah indeks dari chunk frekuensi tersebut (untuk berikutnya akan disebut sebagai waktu). Penyimpanan data menggunakan format waktu#frekuensi, dimana waktu ini adalah indeks dari chunk tempat frekuensi tersebut berada. Tanda # digunakan untuk memisahkan antara waktu dan *frekuensi* dari data. Tabel 4.16 menunjukkan data hasil dari filtering keseluruhan.

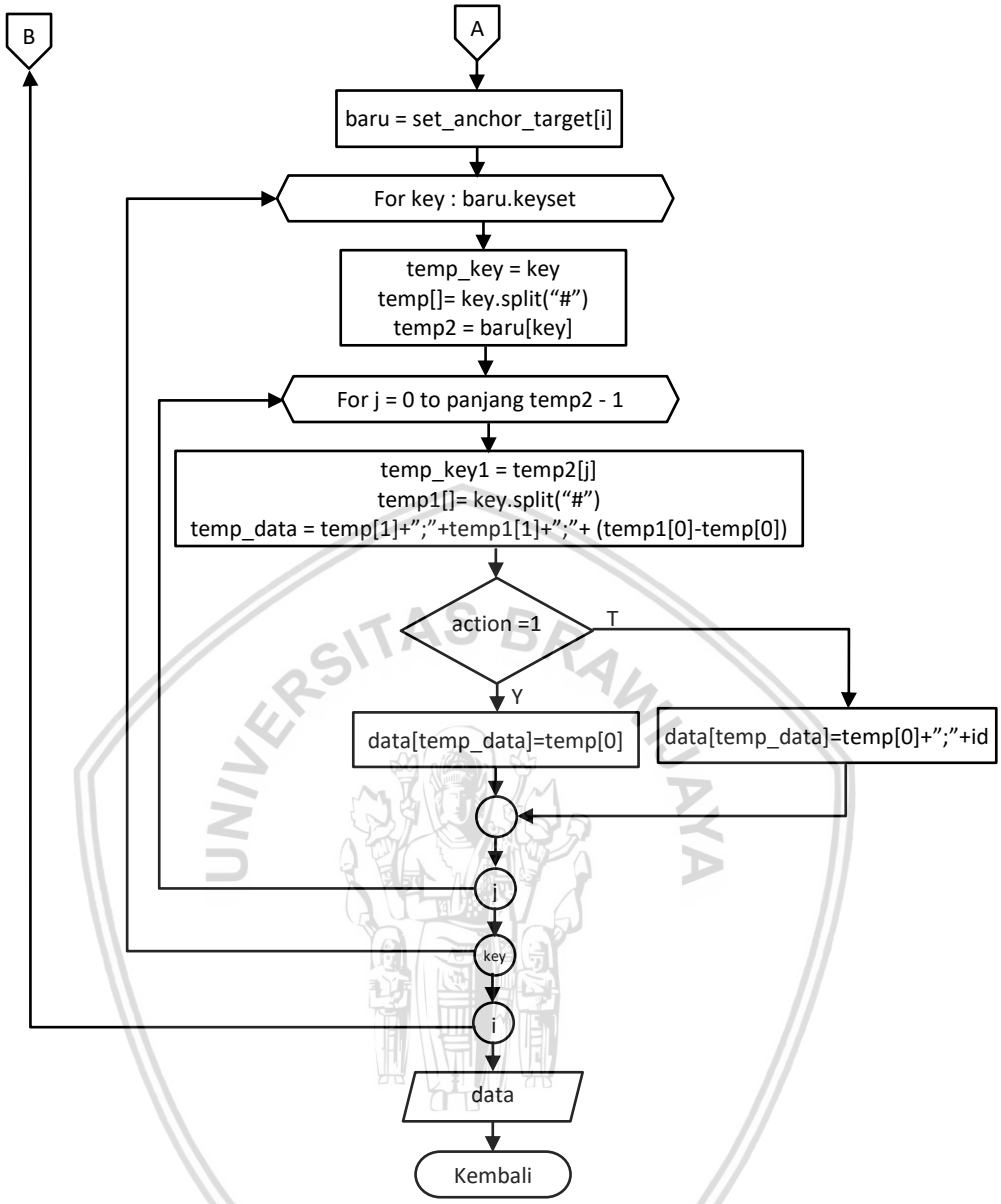
Tabel 4.16 Tabel Data Hasil Filtering

Data hasil filtering
0#3, 0#221, 0#332, 0#381, 0#499, 0#817, 0#1596, 0#1619, 0#1824, 0#1897, 0#1996, 1#94, 1#804, 1#979, 1#1227, 1#1340, 1#1482, 1#1784, 1#1980, 2#4, 2#90, 2#387, 2#509, 2#961, 2#1225, 2#1510, 2#1834, 3#979, 3#1019, 3#1231, 3#1488, 3#1526, 3#1599, 3#1753, 3#1885, 4#50, 4#384, 4#661, 4#865, 4#1363, 4#1807, 4#1946, 5#184, 5#626, 5#692, 5#1278, 5#1686, 5#1802, 6#311, 6#393, 6#587, 6#653, 6#890, 6#1037, 6#1113, 6#1272, 6#1531, 6#1578, 7#39, 7#84, 7#199, 7#282, 7#352, 7#567, 7#611, 7#883, 7#1024, 7#1096, 7#1140, 7#1379, 7#1905, 7#1981, 8#152, 8#226, 8#557, 8#896, 8#964, 8#1246, 8#1337, 8#1373, 8#1530, 8#1577, 8#1624, 8#1902, 9#360, 9#530, 9#612, 9#1099, 9#1197

4.2.5 Proses Pembentukan Hash

Proses pembentukan hash dilakukan untuk memberi alamat dari tiap data yang telah kita dapatkan setelah proses filtering. Gambar 4.12 merupakan diagram alir dari proses pembentukan hash.





Gambar 4.12 Diagram Alir Proses Pembentukan Hash

Berikut penjelasan untuk diagram alir proses pembentukan hash pada Gambar 4.12:

1. Proses menerima masukan *fitur, action* dan *id*
2. Proses membentuk target *zone* dengan cara mengumpulkan *point-point* sebanyak 5 untuk 1 target *zone*.
3. Proses membentuk *anchor point*, *anchor point* diambil dari points yang merupakan berada 3 points sebelum *point* pertama pada target *zone*.
4. Proses melakukan pembentukan *hash*, dimana *hash* terdiri dari pasangan *key* dan *value*, sebagai *key* adalah [frekuensi anchor;frekuensi points;selisih_waktu antara keduanya] dan sebagai *value*-nya akan dilakukan pengkondisian, jika

action = 1 maka yang menjadi *value* adalah waktu dari *anchor*, jika tidak maka yang menjadi *value* adalah [waktu dari *anchor*;*id*].

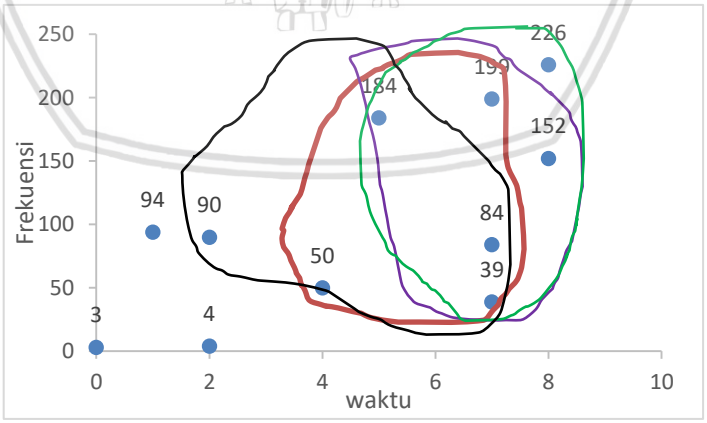
Proses dari pembentukan hash diawali dengan proses pembentukan pasangan target *zone* dan *anchor* point. Target *zone* dibuat dengan mengambil data secara berurutan dari data pada Tabel 4.16, jumlah data pada target zone adalah 5 data. Setelah didapatkan target *zone*, langkah selanjutnya adalah mencari *anchor* point untuk setiap target *zone*, tidak ada aturan yang pasti untuk proses pencarian *anchor* point, menurut Christophe (2015) dalam tulisannya anchor point didapat dari data ketiga sebelum data pertama pada target *zone*. Berikut 5 target *zone* pertama dari keseluruhan pembentukan target *zone* dengan menggunakan data pada Tabel 4.17:

Tabel 4.17 Target Zone

Target_zone ke-*i*	Data
1	0#381, 0#499, 0#817, 0#1596, 0#1619
2	0#499, 0#817, 0#1596, 0#1619, 0#1824
3	0#817, 0#1596, 0#1619, 0#1824, 0#1897
4	0#1596, 0#1619, 0#1824, 0#1897, 0#1996
5	0#1619, 0#1824, 0#1897, 0#1996, 1#94

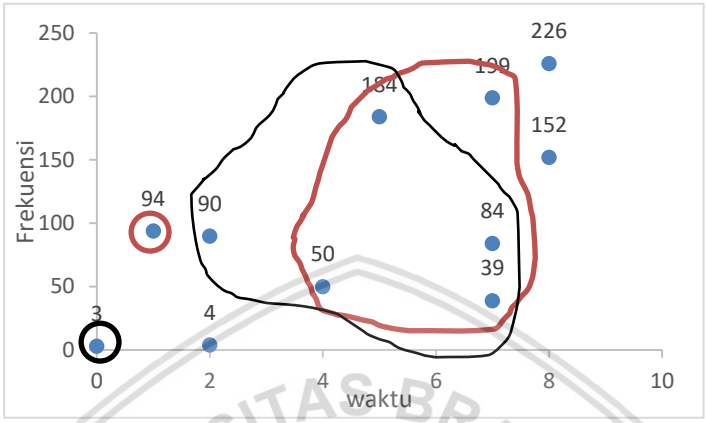
Data yang dipakai untuk membuat target *zone* dimulai dari data ke 4, hal itu dikarenakan data ke-1 akan digunakan sebagai *anchor point* dari target *zone* ke-1. Gambar 4.13 merupakan ilustrasi dari visualisasi target *zone* dengan menggunakan data *dummies* (bukan data sesungguhnya).

Data = 0#3, 1#94, 2#4, 2#90, 4#50, 4#184, 7#39, 7#84, 7#199, 8#152, 8#226



Gambar 4.13 Contoh Representasi Target Zone

Setelah mendapatkan target *zone* seperti pada Tabel 4.17, kemudian akan dilanjutkan dengan melakukan pencarian *anchor* point. Pada penelitian ini peneliti menggunakan aturan yang digunakan oleh Christophe (2015) yaitu *anchor* point adalah data yang terletak sebelum data ke-1 pada target *zone* tepatnya data ke-3 sebelum data ke-1 dari target *zone*. Berikut ilustrasi dari pencarian anchor *point* dari data yang dipakai pada Gambar 4.13:



Gambar 4.14 Contoh Anchor Point dan Target Zone

Dari Gambar 4.14 dapat diketahui bahwa *anchor* point untuk target *zone* dengan garis hitam adalah data yang ditandai dengan lingkaran hitam, yaitu data ke-1 dari keseluruhan data, dan untuk target *zone* dengan garis merah adalah data yang ditandai dengan lingkaran merah juga, yaitu data ke-2 dari keseluruhan data. Berdasarkan aturan di atas, berikut 5 pasang pasangan *anchor* point dan target *zone* dari keseluruhan data yang mana 5 target *zone*-nya ditunjukkan pada Tabel 4.17 akan ditunjukkan pada Tabel 4.18.

Tabel 4.18 Pasangan *Anchor Point* dan Target *Zone*

Pasangan ke-*i*	*Anchor point*	Target *zone*
1	0#3	0#381, 0#499, 0#817, 0#1596, 0#1619
2	0#221	0#499, 0#817, 0#1596, 0#1619, 0#1824
3	0#332	0#817, 0#1596, 0#1619, 0#1824, 0#1897
4	0#381	0#1596, 0#1619, 0#1824, 0#1897, 0#1996
5	0#499	0#1619, 0#1824, 0#1897, 0#1996, 1#94

Langkah selanjutnya adalah memberikan alamat untuk setiap *point* pada target *zone*, pengalamatan ini dilakukan dengan menggunakan pasangan <*key*,*value*> dimana yang menjadi *key* adalah [frekuensi *anchor*;frekuensi *point*;perbedaan waktu] dan untuk *value*-nya ada 2 jenis pengalamatan yang disesuaikan dengan kondisi, jika kondisi sedang melakukan proses indentifikasi maka pengalamatan yang digunakan adalah [waktu dari *anchor*], jika sedang melakukan proses *training* data, maka akan digunakan pengalamatan [waktu dari *anchor*;id dari file suara]. Sebagai contoh, jika kita ingin memberikan alamat pada 2 point awal pada target *zone* berwarna hitam pada Gambar 4.14 dengan menggunakan 2 kondisi, alamat *point* tersebut adalah sebagai berikut:

a. Kondisi identifikasi

- Alamat untuk *point* 90 adalah [frekuensi *anchor*; frekuensi *point*; perbedaan waktu], jadi *key* untuk alamat dari *point* 90 adalah [3;90;2] dan untuk value adalah [waktu dari *anchor*], jadi *value* alamat untuk point 90 adalah [0]
- Alamat untuk *point* 50 adalah, *key* [3;50;4] dan *value* [0]

b. Kondisi training data

Pada kondisi ini yang membedakan hanyalah *value* dari alamat, *value* akan menjadi [waktu dari *anchor*; id dari file suara] maka alamat akan berubah menjadi seperti berikut:

- Alamat point 90 adalah, key [3;90;2] dan value [0;1]
- Alamat point 50 adalah, key [3;50;4] dan value [0;1]

Setelah mendapatkan alamat untuk setiap *point* yang ada dalam target *zone*, maka pasangan <*key*, *value*> ini akan disimpan. Pasangan <*key*,*value*> ini untuk selanjutnya akan disebut sebagai *hash*. Karena pada pembahasan ini skenario yang digunakan adalah pengujian, maka hash yang dihasilkan untuk keseluruhan data pada Tabel 4.16 akan menjadi seperti Tabel 4.19 dan 4.20. Dimana Tabel 4.19 dan 4.20 akan menampilkan 25 *hash* pertama.

Tabel 4.19 Hash

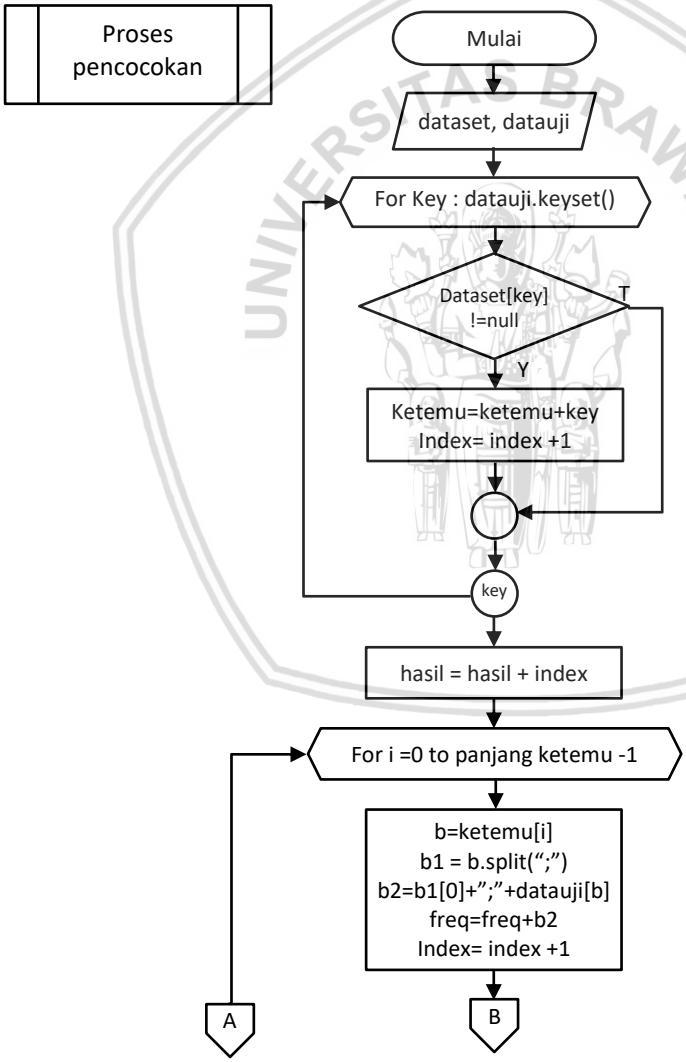
Target zone ke-*i*	Key	Value
1	3;381;0	0
	3;499;0	0
	3;817;0	0
	3;1596;0	0
	3;1619;0	0
2	221;499;0	0
	221;817;0	0
	221;1596;0	0
	221;1619;0	0
	221;1824;0	0
3	332;817;0	0
	332;1596;0	0
	332;1619;0	0
	332;1824;0	0
	332;1897;0	0
4	381;1596;0	0
	381;1619;0	0
	381;1824;0	0

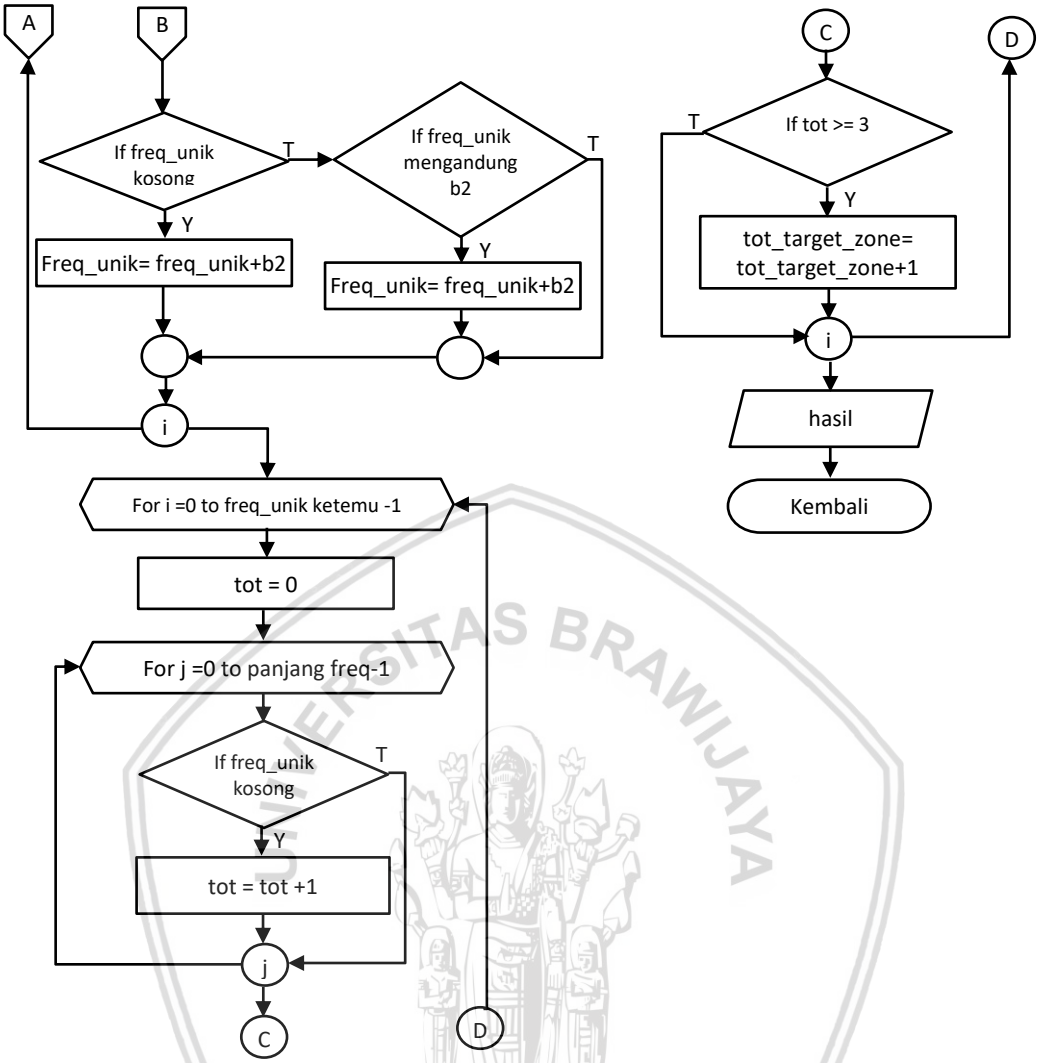
Tabel 4.20 Hash (Lanjutan)

	381;1897;0	0
	381;1996;0	0
5	499;1619;0	0
	499;1824;0	0
	499;1897;0	0
	499;1996;0	0
	499;94;1	0

4.2.6 Proses Pencocokan

Proses pencocokan adalah proses yang dilakukan untuk melakukan identifikasi dengan mencari kesamaan dari hash. Berikut diagram alir dari proses pencocokan akan dijelaskan pada Gambar 4.15.





Gambar 4.15 Diagram Alir Proses Pencocokan

Berikut penjelasan dari diagram alir pada Gambar 4.15:

1. Proses menerima masukan dataset dan data uji.
2. Perulangan pertama adalah proses untuk jumlah mencocokan *key hash* dari data uji dengan *key hash* dari *dataset* yang nantinya akan disimpan dalam variabel hasil.
3. Perulangan ke-2 digunakan untuk mencari frekuensi dari *anchor* yang dinyatakan cocok antara *key hash dataset* dan *key hash dataset* yang nantinya disimpan dalam *freq_unik*.
4. Perulangan ke-3 digunakan untuk mencari kecocokan untuk setiap target *zone*, kumpulan titik dikatakan target *zone* jika ada 3 atau lebih *point* yang memiliki kesamaan *key hash* dengan *freq_unik*.
5. Hasil perulangan ke-4 akan disimpan dalam variabel hasil.
6. Mengembalikan nilai hasil

Proses pencocokan dilakukan untuk mencari berapa *key hash* yang cocok antara data uji dengan *dataset* dan selain itu juga dicari berapa target *zone* yang cocok. Pencocokan dilakukan dengan mengambil *value hash dataset* dengan menggunakan *key* dari data uji, jika hasil pengambilan mendapatkan nilai/tidak null maka akan dilakukan *counter* untuk menyimpan berapa kecocokan yang telah didapatkan. Hasil dari proses ini adalah dua angka yang menunjukan jumlah kecocokan *key hash* dan jumlah kecocokan target *zone*. Dari hasil tersebut akan diketahui sebuah data uji teridentifikasi sebagai surah/hadis apa lewat jumlah kecocokan *key hash* dan jumlah kecocokan target *zone*. Dari dua parameter tersebut parameter utama yang dijadikan acuan adalah jumlah kecocokan target *zone.*Berikut contoh pencocokan dari data uji pada Tabel 4.19 dan 4.20 dengan contoh dari 2 dataset pada Tabel 4.21 dimana dataset ke-1 merupakan data dari surah An-Naas dan *dataset* ke-2 merupakan data dari surah Al-Ikhlas.

Tabel 4.21 Data *Hash Dataset* Ke-1 dan Ke-2

Dataset ke-1			**Dataset ke-2**		
Target zone ke-*i*	**Key**	**Value**	**Target zone ke-*i***	**Key**	**Value**
1	3;381;0	0	1	3;381;0	0
	3;499;0	0		3;499;0	0
	3;117;0	0		3;810;0	0
	3;1396;0	0		3;1596;0	0
	3;1619;0	0		3;1419;0	0
2	152;896;0	0	2	221;499;0	0
	152;964;0	0		221;817;0	0
	152;1246;0	0		221;1596;0	0
	152;1337;0	0		221;1619;0	0
	152;1373;0	0		221;1824;0	0
3	332;817;0	2	3	226;964;0	0
	332;1596;0	2		226;1246;0	0
	332;1619;0	2		226;1337;0	0
	332;1824;0	2		226;1373;0	0
	332;1897;0	2		226;1530;0	0
4	883;1140;0	3	4	381;1596;0	0
	883;1379;0	3		381;1619;0	0
	883;1905;0	3		381;1824;0	0
	883;1981;0	3		381;1897;0	0
	883;152;1	3		381;1996;0	0
5	1379;152;1	0	5	964;1373;0	0
	1379;226;1	0		964;1530;0	0
	1379;557;1	0		964;1577;0	0
	1379;896;1	0		964;1624;0	0
	1379;964;1	0		964;1902;0	0

Tabel 4.22 Ilustrasi Pencocokan *Key Hash*

Key Hash uji	Key Hash dataset ke-1	Key Hash dataset ke-2	Kecocokan dataset ke-1	Kecocokan dataset ke-2
3;381;0	3;381;0	3;381;0	√	√
3;499;0	3;499;0	3;499;0	√	√
3;817;0	3;117;0	3;810;0	-	√
3;1596;0	3;1396;0	3;1596;0	-	√
3;1619;0	3;1619;0	3;1419;0	√	-
221;499;0	152;896;0	221;499;0	-	√
221;817;0	152;964;0	221;817;0	-	√
221;1596;0	152;1246;0	221;1596;0	-	√
221;1619;0	152;1337;0	221;1619;0	-	√
221;1824;0	152;1373;0	221;1824;0	-	√
332;817;0	332;817;0	226;964;0	√	-
332;1596;0	332;1596;0	226;1246;0	√	-
332;1619;0	332;1619;0	226;1337;0	√	-
332;1824;0	332;1824;0	226;1373;0	√	-
332;1897;0	332;1897;0	226;1530;0	√	-
381;1596;0	883;1140;0	381;1596;0	-	√
381;1619;0	883;1379;0	381;1619;0	-	√
381;1824;0	883;1905;0	381;1824;0	-	√
381;1897;0	883;1981;0	381;1897;0	-	√
381;1996;0	883;152;1	381;1996;0	-	√
499;1619;0	1379;152;1	964;1373;0	-	-
499;1824;0	1379;226;1	964;1530;0	-	-
499;1897;0	1379;557;1	964;1577;0	-	-
499;1996;0	1379;896;1	964;1624;0	-	-
499;94;1	1379;964;1	964;1902;0	-	-
Jumlah			8	14

Tabel 4.22 merupakan ilustrasi pencocokan dengan data uji dari Tabel 4.19 dan 4.20 dengan *dataset* pada Tabel 4.21. Dari Tabel 4.22 dapat diketahui bahwa untuk *dataset* ke-1 hasil kecocokannya adalah 1 dan 8, sedangan untuk *dataset* ke-2 hasil kecocokannya adalah 3 dan 14. Nilai 1 dan 3 didapatkan dari kecocokan target

zone. Dinamakan kecocokan target *zone* ketika terdapat 3 atau lebih kecocokan *point* dengan anchor yang sama untuk 3 atau lebih *point* tersebut. Dari hasil pada Tabel 4.22 dapat disimpulkan bahwa hasil identifikasi adalah *dataset* ke-2, hal itu dikarenakan jumlah kecocokan target *zone* dan *key hash* dari dataset ke-2 lebih besar dari *dataset* ke-1. Dari hasil identifikasi di atas didapatkan informasi bahwa proses identifikasi bernilai benar, dikarenakan data uji yang merupakan surah Al-Ikhlas berhasil di identifikasi sebagai surah Al-Ikhlas juga.

Setelah didapatkan hasil untuk data uji maka akan dilakukan perhitungan akurasi, perhitungan akurasi dilakukan dengan mengacu pada Persamaan 2.14 dari bab 2. Berikut contoh perhitungan akurasi dengan jumlah data benar = 45 dan jumlah data yang diujikan = 50:

$$Akurasi = \frac{45}{50} * 100\%$$

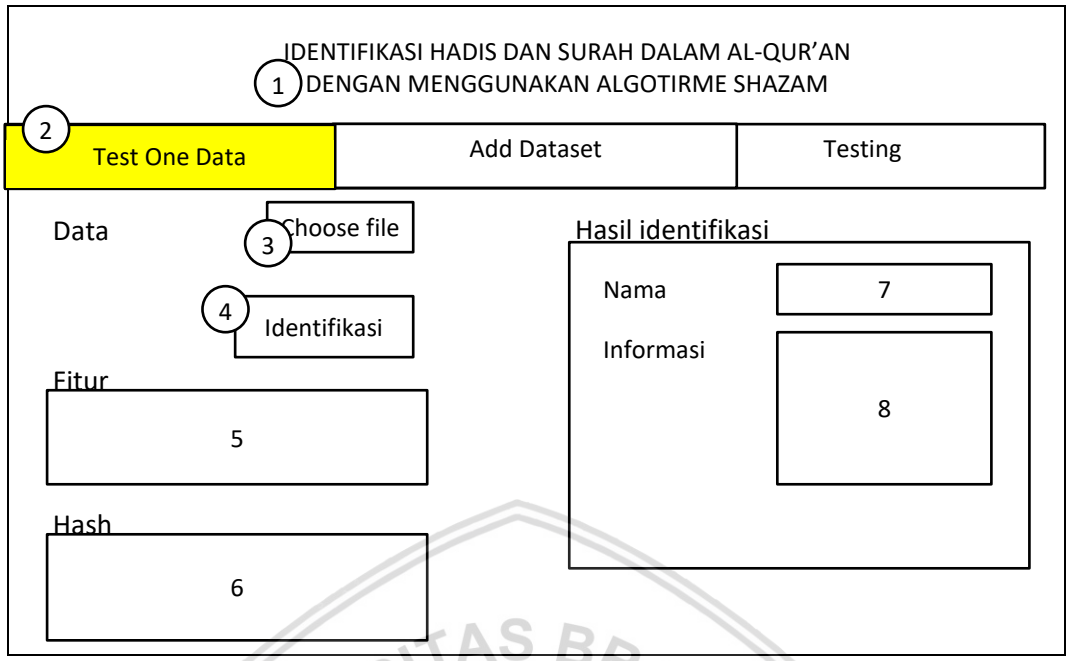
= 0.9*100% = 90%

4.3 Perancangan Antarmuka

Perancangan antarmuka bertujuan untuk memberi gambaran hasil implementasi identifikasi hadis dan surah dalam Al-Qur'an. Adapun antarmuka terdiri dari halaman test one data, add dataset, testing.

4.3.1 Perancangan Halaman Test One Data

Halaman ini digunakan untuk melakukan identifikasi sebuah data, pada halaman ini masukan yang dibutuhkan adalah data suara ekstensi .wav, data akan diproses dengan menggunakan parameter terbaik dari hasil pengujian. Hasil dari identifikasi berupa nama surah / hadis dan informasi akan ditampilkan pada halaman ini juga. Gambar 4.16 merupakan perancangan antarmuka halaman test one data.



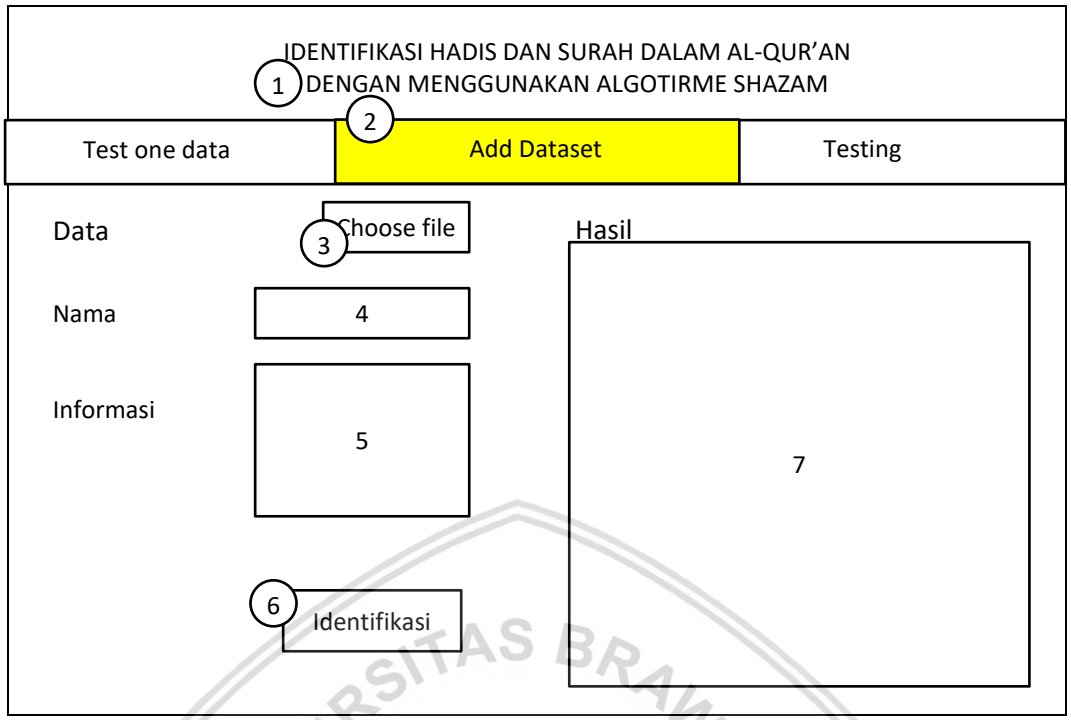
Gambar 4.16 Rangcangan Halaman Test One Data

Keterangan Gambar 4.16:

1. Header aplikasi
2. Tab aktif
3. Button untuk chosse file dengan ekstensi .wav
4. Button untuk menjalankan proses identifikasi
5. Tabel untuk menampilkan fitur yang dihasilkan
6. Tabel untuk menampilkan fingerprint dan hash yang dihasilkan
7. Text field non-editable untuk menampilkan nama hasil nama hadis/surah dari file yang dimasukan
8. Text area non-editable untuk menampilkan informasi hasil dari file yang dimasukan

4.3.2 Perancangan Halaman Add Dataset

Halaman ini digunakan untuk menambahkan dataset yang terdapat pada penyimpanan, masukan dari halaman ini adalah file dengan ekstensi wav, nama surah / hadis dari file, informasi dari file. Pada halaman ini juga akan menampilkan daftar dataset yang telah tersimpan. Rancangan halaman ini bisa dilihat pada Gambar 4.17.



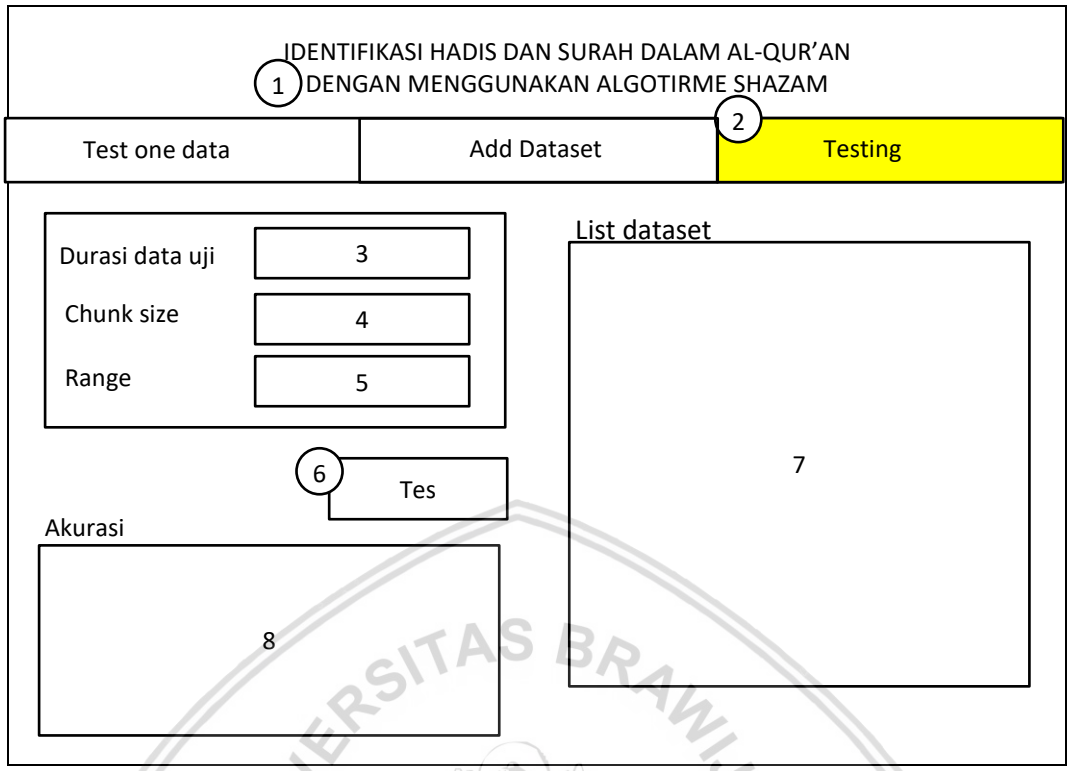
Gambar 4.17 Rancangan Halaman Add Dataset

Keterangan Gambar 4.17:

1. Header aplikasi
2. Tab aktif
3. Button untuk chosse file dengan ekstensi .wav
4. Text file untuk memasukan nama dari data
5. Text area untuk memasukan informasi dari data
6. Button untuk memproses penambahan data baru ke database
7. Table untuk menampilkan daftar dataset yang telah ada pada database

4.3.3 Perancangan Halaman Testing

Halaman ini digunakan untuk melakukan testing sistem, tujuannya adalah untuk mendapatkan nilai akurasi dari sistem yang dibuat, pada halaman ini masukannya adalah berupa parameter-parameter pengujian yang sudah ditentukan, parameternya adalah durasi data uji, besar chunk, dan range. Pada halam ini juga akan ditampilkan hasil dari proses pengujian. Perancangan halaman testing bisa dilihat pada Gambar 4.18.



Gambar 4.18 Rancangan Halaman Testing

Keterangan Gambar 4.18:

1. Header aplikasi
2. Tab aktif
3. Dropdown menu untuk memilih parameter durasi data uji
4. Dropdown menu untuk memilih parameter besar chunk
5. Dropdown menu untuk memilih parameter besar range
6. Button untuk memproses pengujian
7. Table untuk menampilkan hasil dari peroses pengujian

4.4 Pengujian Algoritme

Pengujian algoritme pada penelitian ini digunakan untuk mendapatkan nilai akurasi terbaik dengan menggunakan kombinasi parameter masukan. Pada proses pengujian dataset yang digunakan adalah 10 data, 6 surah dan 4 hadis. Kemudian untuk data uji digunakan 5 potongan kecil untuk masing–masing *dataset* yang digunakan. Berikut pengujian yang digunakan adalah:

1. Pengujian panjang data uji
2. Pengujian *chunk size*
3. Pengujian range

Penghitungan nilai akurasi dilakukan dengan menghitung perbandingan data hasil sistem yang di identifikasi secara benar dengan keseluruhan data yang

kemudian akan dikalikan dengan 100%, Persamaan 2.14 merupakan formulasi yang digunakan untuk menghitung nilai akurasi.

4.4.1 Pengujian Panjang Data Uji

Pengujian panjang data uji dilakukan untuk mendapatkan parameter panjang data uji terbaik yang nantinya akan digunakan untuk mendapatkan nilai akurasi. Pengujian ini menggunakan 6 variasi panjang data uji , yaitu: 3 detik, 5 detik, 7 detik, 10 detik, 12 detik dan 15 detik. Pada pengujian ini akan digunakan semua data uji yang telah disiapkan, menggunakan *chunk size* = 4096 dan *range* = 60. Rancangan pengujian ini dapat dilihat pada Tabel 4.23.

Tabel 4.23 Rancangan Pengujian Panjang Data Uji

Percobaan ke-*i*	Akurasi pada panjang data uji					
	3 detik	5 detik	7 detik	10 detik	12 detik	15 Detik
1						
2						
3						
4						
5						
Rata – rata akurasi						

4.4.2 Pengujian Besar Chunk

Pengujian besar chunk bertujuan untuk mendapatkan parameter *chunk size* terbaik agar didapatkan akurasi yang paling besar. *Chunk size* yang digunakan adalah 512, 1024, 2048, 4096 dan 8192, 16384. Pada pengujian ini digunakan semua data uji yang telah disiapkan, untuk panjang data uji akan digunakan dari hasil pengujian panjang data uji dengan nilai akurasi tertinggi dan besar *range* = 60. Rancangan pengujian ini dapat dilihat pada Tabel 4.24.

Tabel 4.24 Rancangan Pengujian Chunk Size

Percobaan ke-*i*	Akurasi pada chunk size					
	512	1024	2048	4096	8192	16384
1						
2						
3						
4						
5						
Rata – rata akurasi						

4.4.3 Pengujian Range

Pengujian range hash bertujuan untuk mendapatkan parameter *range* terbaik agar didapatkan akurasi yang paling besar. *Range* yang digunakan yang digunakan adalah 20, 30, 40, 60, 70, 80, 90, 100. Pada pengujian ini digunakan semua data uji yang telah disiapkan, panjang data uji yang digunakan adalah dari hasil pengujian panjang data uji dengan nilai akurasi tertinggi, *chunk size* yang digunakan adalah *chunk size* dengan nilai akurasi terbaik dari hasil pengujian sebelumnya. Rancangan pengujian ini dapat dilihat pada Tabel 4.25.

Tabel 4.25 Rancangan Pengujian Besar Range

Percobaan Ke - *i*	Akurasi (%)								
	20	30	40	50	60	70	80	90	100
1									
2									
3									
4									
5									
Rata-rata akurasi									

BAB 5 IMPLEMENTASI

Pada bab ini akan berisi hasil implementasi dari hasil analisis kebutuhan dan perancangan yang telah dilakukan pada bab sebelumnya.

5.1 Implementasi Algoritme

Algortime shazam yang digunakan memiliki beberapa sub proses seperti yang telah dibahas pada perancangan. Berikut hasil implementasi dari algoritme shazam.

5.1.1 Implementasi Algoritme Ekstraksi Nilai Numerik

Algoritme ekstraksi nilai numerik merupakan proses untuk mendapatkan data numerik dari masukan berupa file suara. Kode Program 5.1 merupakan kode program untuk algoritma ekstraksi nilai numerik.

```
1  int numBytes = 1024 * bytesPerFrame;
2  byte[] datasampel = new byte[numBytes];
3  try {
4      while((numBytesRead = audioInputStream.read(datasampel)) != -
5      1) {
6              data.write(datasampel, 0, numBytesRead);
7      }
8  } catch (Exception ex) {}
```

Kode Program 5.1 Implementasi Algoritme Ekstraksi Nilai Numerik

Berikut penjelasan Kode Program 5.1:

1. Baris 1-2 adalah inisialisasi variabel dan array.
2. Baris 3-6 menjelaskan tentang proses pengambilan data dari file suara yang telah dimasukkan dalam variabel audioInputStream.
3. Hasil dari algoritme ini adalah data dalam byte.

5.1.2 Implementasi Algoritme Konversi

Algoritme konversi ini merupakan proses yang dilakukan untuk mengubah data yang awalnya dalam byte dengan jumlah 4 kali sample. Pada proses ini akan dilakukan penggabungan setiap 2 byte data dan kemudian akan dilakukan perhitungan rata-rata untuk tiap 2 data hasil dari proses penggabungan. Proses mencari rata-rata ini digunakan untuk mengubah channel stereo ke mono. Kode Program 5.2 adalah implementasi dari proses konversi.

```
1  for (int i = 0; i < temp_data.length; i++) {
2      if (i == temp_data.length - 1) {
3         break;
4      }
5      sample_stereo.add((double) (temp_data[i] << 8 | temp_data[i +
6      1] & 0xFF));
7      i++;
8  }
9  for (int i = 0; i < sample_stereo.size(); i++) {
10     double avg = (sample_stereo.get(i) + sample_stereo.get(i + 1))
11     / 2;
12     sample_mono.add(avg);
```

```
13      i++;
14  }
```

Kode Program 5.2 Implementasi Algoritme Konversi

Berikut penjelasan dari Kode Program 5.2:

1. Baris ke 1-8 menjelaskan tentang proses penggabungan setiap 2 byte data.
2. Baris ke 9-14 menjelaskan tentang proses perhitungan rata setiap 2 data hasil penggabungan.

5.1.3 Implementasi Algoritme Ekstraksi Fitur

Algoritme ekstaksi fitur ini merupakan proses yang dilakukan untuk mendapatkan fitur dari data. Fitur didapat dari proses perhitungan FFT dan proses getpoint, berikut implementasi ekstraksi fitur dijelaskan pada Kode Program 5.2.

```
1   for (int i = 0; i < temp_range.length; i++) {
2       temp_range[i] = tot;
3       tot += range;
4   }
5   temp_range[temp_range.length - 1] = max + 1;
6   setRange(temp_range);
7   for (int times = 0; times < jum_chunk; times++) {
8       Complex[] temp = new Complex[chunk size];
9       List temp1 = new ArrayList();
10      for (int i = 0; i < chunk size; i++) {
11          temp1.add(audio.get((times * chunk size) + i));
12      }
13      List<Double> hamming = do_hamming(temp1);
14      for (int i = 0; i < chunk size; i++) {
15          temp[i] = new Complex(hamming.get(i), 0);
16      }
17      Complex[] hasil_fft = FFT.fft(temp);
18      points = getpoints(hasil_fft, times, range, jum_range);
19      temp_has.put(times, points);
20  }
```

Kode Program 5.3 Implementasi Algoritme Ekstraksi Fitur

Berikut penjalasan dari Kode Program 5.3:

1. Baris 1-6 merupakan proses pemberian nilai untuk variabel range.
2. Baris 10-12 merupakan proses membagi data menjadi chunk.
3. Baris ke 13 merupakan proses pemanggilan method perhitungan hamming, akan dijelaskan pada Kode program 5.4.
4. Baris ke 14-16 merupakan proses mengubah bilangan real ke bilangan kompleks.
5. Baris ke 17 merupakan proses pemanggilan method FFT, akan dijelaskan pada Kode program 5.5.
6. Baris ke 18 merupakan proses pemanggilan method getpoints, akan dijelaskan pada Kode program 5.6.
7. Baris ke 13 merupakan proses menyimpan data dalam map <*key*,*value*>.

5.1.3.2 Implementasi Algoritme Perhitungan Hamming

Algoritme perhitungan hamming ini adalah proses untuk melakukan *windowing* dengan menggunakan *hamming window*. Berikut penjelasan dari implementasi proses perhitungan *hamming window* akan dijelaskan pada Kode Program 5.4.

```
1   int size = data.size();
2   List<Double> temp = new ArrayList(data);
3   List<Double> hasil = new ArrayList();
4   double a = 0.54, b = 0.46;
5   int N = size - 1;
6   for (int i = 0; i < size; i++) {
7       double hamming = a - b * Math.cos((2 * Math.PI * i) / N);
8       hasil.add(temp.get(i) * hamming);
9   }
```

Kode Program 5.4 Implementasi Perhitungan Hamming

Berikut penjelasan dari Kode Program 5.4:

1. Baris 1- 5 merupakan proses untuk melakukan inisialisasi beberapa variabel
2. Baris 6-8 merupakan proses perhitungan *hamming window* dan perkalian antara data dengan hasil hamming

5.1.3.3 Implementasi Algoritme Perhitungan FFT

Algoritme perhitungan FFT ini digunakan untuk melakukan perhitungan *fourier transform* yang nantinya hasilnya akan berupa koefisien *fourier* untuk tiap frekuensi yang ada. Berikut penjelasan dari algoritme perhitungan FFT akan dijelaskan pada Kode Program 5.5.

```
1   if (N == 1) {
2      return new Complex[]{data[0]};
3   }
4   if (N % 2 != 0) {
5      throw new RuntimeException("jumlah data tidak bisa dibagi 2");
6   }
7   temp_even = new Complex[N / 2];
8   for (int k = 0; k < N / 2; k++) {
9       temp_even[k] = data[2 * k];
10  }
11  even = fft(temp_even);
12  temp_odd = temp_even;
13  for (int k = 0; k < N / 2; k++) {
14      temp_odd[k] = data[2 * k + 1];
15  }
16  odd = fft(temp_odd);
17  for (int k = 0; k < N / 2; k++) {
18      double kth = -2 * k * Math.PI / N;
19      Complex wk = new Complex(Math.cos(kth), Math.sin(kth));
20      hasil[k] = even[k].plus(wk.times(odd[k]));
21      hasil[k + N / 2] = even[k].minus(wk.times(odd[k]));
22  }
```

Kode Program 5.5 Implementasi Proses Perhitungan FFT

Berikut penjelasan dari Kode Program 5.5:

1. Baris 1-3 menjelaskan proses untuk mengembalikan nilai dari data ke 0 jika nilai *N* adalah 1.

2. Baris 4-5 menjelaskan proses untuk memberikan peringatan eror dengan pesan jika *N* mod 2 tidak menghasilkan 0.
3. Baris 7-16 menjelaskan proses untuk memisahkan data antara data dengan indeks genap dan data dengan indeks ganjil, data genap akan disimpan dalam variabel *temp_even*, dan data ganjil akan disimpan dalam variabel *temp_odd*.
4. Baris 17-22 merupakan proses perhitungan FFT.

5.1.3.4 Implementasi Algoritme Getpoints

Algoritme getpoints merupakan proses yang digunakan untuk mencari frekuensi dengan nilai *magnitude* tertinggi untuk setiap *range* yang telah disediakan. Hasil dari algoritme ini adalah pasangan <*key*, *value*> dengan frekuensi sebagai key dan *magnitude* sebagai value. Penjelasan dari implementasi algoritme getpoints dapat dilihat pada Kode Program 5.6

```
1   for (int freq = 1; freq < (range * jum_range) - 1; freq++) {
2       double mag = Math.log(data_hasil[freq].abs() + 1);
3       int indeks = getIndeks(freq);//get indeks dari max magnitude
4       if (mag > temp[indeks]) {
5          temp[indeks] = mag;
6          point[indeks] = freq;
7
8       }
9   }
10  double tot = 0, avg = 0;
11  for (int i = 0; i < temp.length; i++) {
12      tot += temp[i];
13  }
14  avg = tot / temp.length;
15  List points = new ArrayList();
16  for (int j = 0; j < temp.length; j++) {
17      if (temp[j] > avg) {
18         Map data = new HashMap();
19         data.put(point[j], temp[j]);
20         points.add(data);
21      }
22  }
```

Kode Program 5.6 Implementasi Algoritme Getpoints

Berikut penjelasan dari Kode Program 5.6:

1. Baris ke 1- 9 merupakan proses untuk mencari nilai *magnitude* dan frekuensi dengan nilai *magnitude* tertinggi yang mana nilai frekuensi akan disimpan dalam variabel temp dan frekuensinya akan disimpan dalam variabel point.
2. Baris 10-14 merupakan proses untuk mencari rata-rata dari *magnitude* yang telah didapatkan.
3. Baris 15-22 merupakan proses untuk mengambil pasangan frekuensi dan *magnitude* yang nilai *magnitude*-nya di atas nilai rata-rata.

5.1.3.4.1 Implementasi Algoritme Getindeks

Algoritme getindeks digunakan untuk mencari tahu pada range ke berapa sebuah frekuensi. Penjelasan dari implementasi algortime filtering dapat dilihat pada Kode Program 5.7.

```
1   int i = 0;
2   while ((int) RANGE.get(i) < freq) {
3        i++;
4   }
5   return i;
```

Kode Program 5.7 Implementasi Algoritme Getindeks

Berikut penjelasan dari Kode Program 5.7:

1. Baris 2 – 4 merupakan proses untuk mencari indeks dari frekuensi masukan.

5.1.4 Implementasi Algoritme Filtering

Algoritmne filtering digunakan untuk melakukan filtering dari fitur yang telah didapatkan. Dengan menggunakan cara mengambil fitur yang memiliki nilai *magnitude* lebih tinggi dari rata-rata. Penjelasan dari implementasi algoritme filtering dapat dilihat pada Kode Program 5.8.

```
1   for (Object key : a.keySet()) {
2       List temp = (List) a.get(key);
3       for (int i = 0; i < temp.size(); i++) {
4           Map temp2 = (Map) temp.get(i);
5           for (Object key2 : temp2.keySet()) {
6               tot += (double) temp2.get(key2);
7               jum++;
8           }
9       }
10  }
11  avg = tot / jum;
12  List hasil = new ArrayList();
13  for (Object key : a.keySet()) {
14      List temp = (List) a.get(key);
15      String temp1 = "";
16      for (int i = 0; i < temp.size(); i++) {
17          Map temp2 = (Map) temp.get(i);
18          for (Object key2 : temp2.keySet()) {
19              if ((double) temp2.get(key2) > avg) {
20                  temp1 = String.valueOf(key) + "#" +
21                  String.valueOf(key2);
22                  hasil.add(temp1);
23              }
24          }
25      }
26  }
```

Kode Program 5.8 Implementasi Algoritme Filtering

Berikut penjelasan dari Kode Program 5.8:

1. Baris 1-11 merupakan proses untuk mencari nilai rata-rata dari data.
2. Baris 12-26 merupakan proses untuk menyimpan fitur dengan nilai *magnitude* yang lebih tinggi dari nilai rata-rata.

5.1.5 Implementasi Algoritme Pembentukan Hash

Algoritme pembetukan hash dilakukan untuk mendapatkan *anchor point* dan *target zone* yang kemudian akan dijadikan pasangan <*key, value*> yang nantinya akan digunakan untuk proses pencocokan. Penjelasan dari implementasi algoritme pembentukan hash dapat dilihat pada Kode Program 5.9.

```
1   for (int i = anchor_offset; i < a.size() - (anchor_offset + 1);
2       i++) {
3       List temp_target_zone = new ArrayList();
4       Map anchor_target = new HashMap();
5       for (int j = 0; j < jum_perzone; j++) {
6           temp_target_zone.add(a.get(i + j));
7       }
8       anchor_target.put(a.get(a.indeksOf(a.get(i - anchor_offset))),
9       temp_target_zone);
10      set_anchor_target.add(anchor_target);
11  }
12  for (int i = 0; i < set_anchor_target.size(); i++) {
13      Map baru = (Map) set_anchor_target.get(i);
14      for (Object key : baru.keySet()) {
15          String temp_key = String.valueOf(key);
16          String temp[] = temp_key.split("#");
17          List temp2 = (List) baru.get(key);
18          for (int j = 0; j < temp2.size(); j++) {
19              String temp_key1 = String.valueOf(temp2.get(j));
20              String temp1[] = temp_key1.split("#");
21              String temp_data = temp[1] + ";" + temp1[1] + ";" +
22              (Integer.valueOf(temp1[0]) - Integer.valueOf(temp[0]));
23              if (action == 1) {
24                  data.put(temp_data, temp[0]);
25              } else {
26                  String value = temp[0] + ";" + String.valueOf(id);
27                  data.put(temp_data, value);
28              }
29          }
30      }
31  }
```

Kode Program 5.9 Implementasi Algortime Pembentukan Hash

Berikut penjelasan dari Kode Program 5.9:

1. Baris 1-11 digunakan untuk mendapatkan *anchor point* dan *target zone* yang nantinya disimpan dalam variabel *set_anchot_target*.
2. Baris 12-31 menjelaskan proses pembentukan *hash*, *hash* disini berupa padangan *<key, value>* yang disimpan dalam variabel data, hash inilah yang nantinya akan digunakan untuk proses pencocokan.

5.1.6 Implementasi Algoritme Pencocokan

Algoritme pencocokan merupakan proses yang digunakan untuk mengetahui tingkat kecocokan sebuah data uji dengan sebuah dataset. Hasil dari algortime pencocokan ini adalah sebuah list yang menyimpan berapa kecocokan untuk tiap hash dan berapa kecocokan untuk target zone. Penjelasan dari implementasi pencocokan dapat dilihat pada Kode Program 5.10.

```
1   for (Object key : datauji.keySet()) {
2       if (dataset.get(key) != null) {
3          ketemu.add(key);
4          indeks++;
5       }
6   }
7   hasil.add(indeks);
8   for (int i = 0; i < ketemu.size(); i++) {
9       String b = String.valueOf(ketemu.get(i));
10      String b1[] = b.split(";");
11      freq.add(b1[0]);
```

```
12      if (freq_unix.isEmpty()) {
13         freq_unix.add(b1[0]);
14      } else if (!freq_unix.contains(b1[0])) {
15         freq_unix.add(b1[0]);
16      }
17  }
18  for (int i = 0; i < freq_unix.size(); i++) {
19      int tot = 0;
20      for (int j = 0; j < freq.size(); j++) {
21          if (freq.get(j).equals(freq_unix.get(i))) {
22             tot++;
23          }
24      }
25      if (tot >= 3) {
26         tot_target_zone++;
27      }
28  }
29  hasil.add(tot_target_zone);
```

Kode Program 5.10 Implementasi Algoritme Pencocokan

Berikut penjelasan dari Kode Program 5.10:

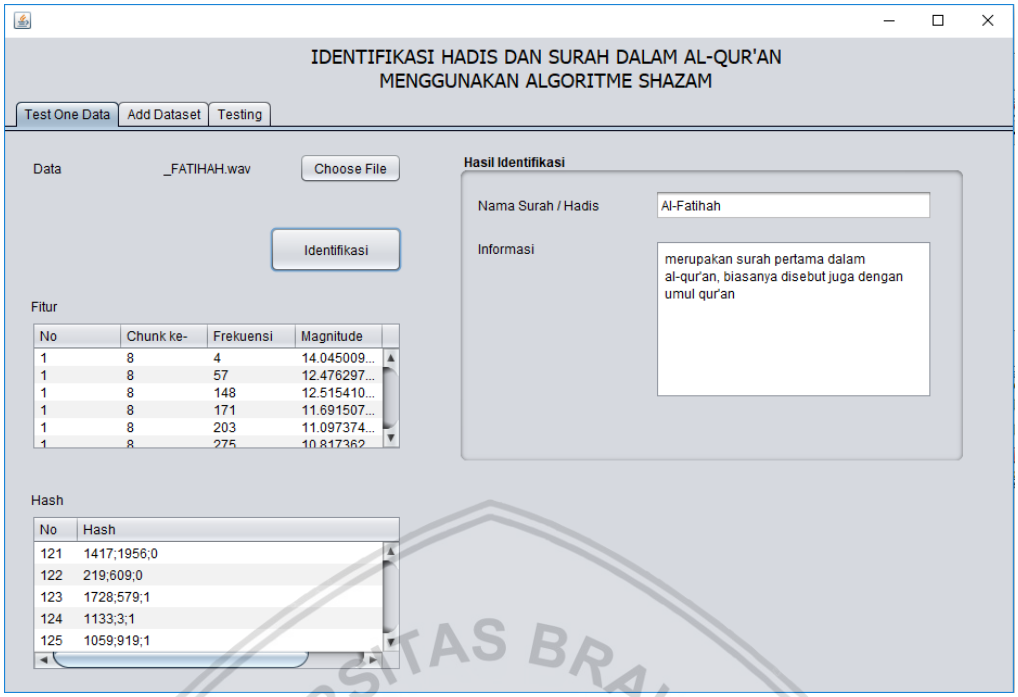
1. Baris 1-6 merupakan proses untuk mencari total kecocokan pada *hash*.
2. Baris 7 merupakan proses untuk menyimpan nilai dari total kecocokan *anchor*.
3. Baris 8-17 merupakan proses untuk menyimpan semua *anchor* yang ada.
4. Baris 18-28 merupakan proses untuk mencari total kecocokan pada target *zone*.
5. Baris 29 merupakan proses untuk menyimpan nilai dari total kecocokan target *zone*.

5.2 Implementasi Antarmuka

Implementasi antarmuka adalah implementasi hasil dari perancangan antar muka pada bab 4. Implementasi antarmuka yang dilakukan pada penelitian ini adalah implementasi antarmuka *tets one* data, add *dataset* dan *testing*.

5.2.1 Implementasi Antarmuka *Test One* Data

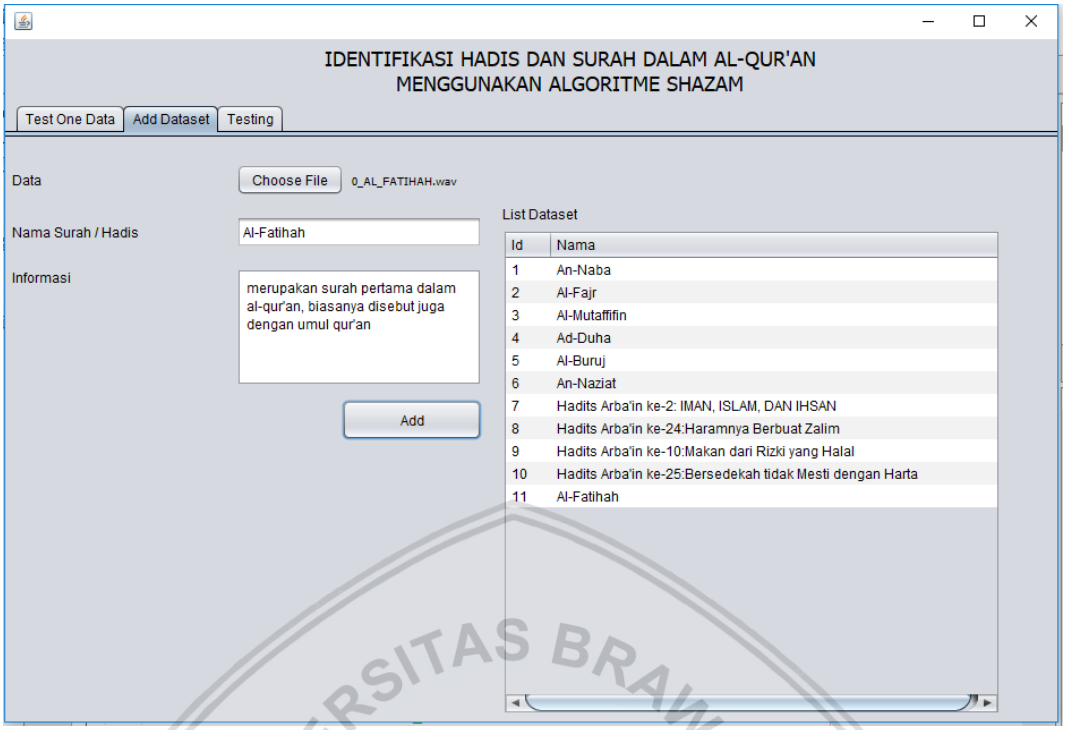
Implementasi antarmuka *test one* data adalah implementasi dari perancangan antarmuka test one data dengan masukan berupa file suara. Antarmuka *test one* data menampilkan hasil identifikasi berupa nama, informasi, fitur dan hash dai file suara yang menjadi masukan. Implementasi antarmuka test one data ditunjukkan pada Gambar 5.1.



Gambar 5.1 Implementasi Antarmuka Test One Data

5.2.2 Implementasi Antarmuka *Add Dataset*

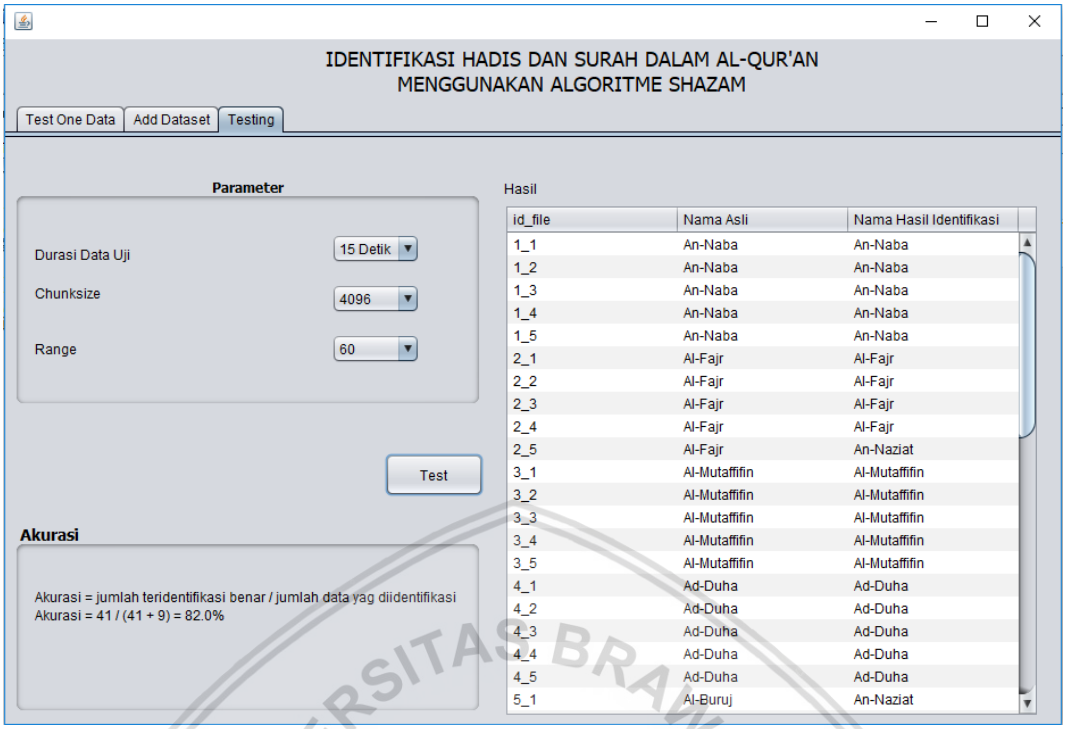
Implementasi antarmuka *add dataset* adalah implementasi dari perancangan antarmuka add dataset pada Gambar 4.17. Masukan pada antarmuka add dataset berupa file suara, nama surah/hadis dan informasi mengenai surah/hadis. Pada antarmuka ini akan ditampilkan list dataset yang sudah tersimpan pada penyimpanan. Gambar 5.2 merupakan hasil dari implementasi antarmuka *add dataset*.



Gambar 5.2 Implementasi Antarmuka *Add Dataset*

5.2.3 Implementasi Antarmuka *Testing*

Implementasi antarmuka testing adalah implementasi dari perancangan antarmuka *testing* pada Gambar 4.18. Masukan pada antarmuka *testing* berupa parameter-parameter pengujian seperti durasi data uji, *chunk size* dan *range* yang digunakan. Pada antarmuka ini akan ditampilkan daftar hasil pengujian dan akurasi dari sistem dengan parameter yang menjadi masukan. Gambar 5.3 merupakan hasil dari implementasi antarmuka *testing*.



Gambar 5.3 Implementasi Antarmuka Testing

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini membahas tentang pengujian serta analisis dari hasil pengujian yang dilakukan terhadap sistem yang telah dibuat. Pengujian yang dilakukan adalah pengujian hubungan antara variabel uji dengan akurasi yang dihasilkan. Pengujian tingkat akurasi dilakukan dengan menggunakan Persamaan 2.14. Pada pengujian ini digunakan 6 surah dan 4 hadis sebagai dataset dan untuk data uji menggunakan 5 data per surah dan hadis, jadi terdapat 50 data uji.

6.1 Pengujian Tingkat Akurasi Terhadap Panjang Data Uji

Pengujian akurasi terhadap panjang data uji ini menjelaskan tentang hasil dari skenario pengujian panjang data uji, dan analisis dari hasil skenario pengujian panjang data uji.

6.1.1 Skenario Pengujian Panjang Data Uji

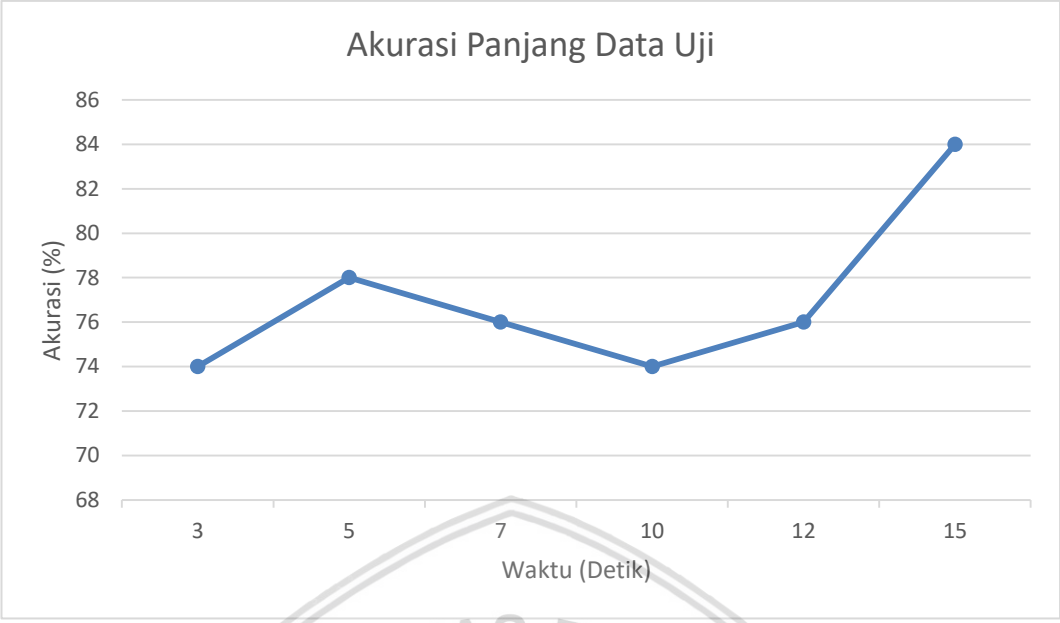
Skenario ini digunakan untuk melihat berapa panjang data uji yang akan menghasilkan akurasi terbaik. Variasi panjang data uji yang dipakai adalah 3, 5, 7, 10, 12, 15. Parameter selain panjang data uji akan dibuat static yaitu 4096 untuk *chunk size* dan 60 untuk besar satiap *range*-nya. Hasil dari pengujian panjang data uji bisa dilihat pada Tabel 6.1.

Tabel 6.1 Hasil Pengujian Panjang Data Uji

Percobaan ke-*i*	**Akurasi pada panjang data uji (%)**					
	3 detik	**5 detik**	**7 detik**	**10 detik**	**12 detik**	**15 Detik**
1	74	78	76	74	76	82
2	74	78	76	74	76	82
3	74	78	76	74	76	82
4	74	78	76	74	76	82
5	74	78	76	74	76	82
Rata – rata akurasi	74	78	76	74	76	82

6.1.2 Analisis Pengujian Durasi Data Uji

Analisis dilakukan dengan membuat grafik seperti pada Gambar 6.1 dengan menggunakan data hasil pengujian pada Tabel 6.1. Pada Gambar 6.1 dapat dilihat bahwa akurasi tertinggi adalah 84% dengan durasi data uji 15 detik.



Gambar 6.1 Grafik Akurasi Hasil Pengujian Panjang Data Uji

Dari Gambar 6.1 dapat diketahui bahwa panjang data 15 detik merupakan panjang data uji yang cukup baik, hal itu ditunjukkan dengan akurasi yang dihasilkan. Dengan kata lain jika melihat hubungan antara hasil akurasi dengan panjang data uji adalah berbanding lurus, semakin lama panjang data uji maka nilai akurasi akan mengalami peningkatan. Hal tersebut dapat terjadi dikarenakan semakin lama waktu data uji maka semakin banyak fitur yang dihasilkan dan akan memperbanyak *hash* yang berdampak pada kemungkinan kecocokan yang dihasilkan akan semakin besar.

6.2 Pengujian Tingkat Akurasi Terhadap *Chunk size*

Pengujian terhadap *chunk size* menjelaskan hasil dari skenario pengujian akurasi terhadap *chunk size* dan analisis dari hasil skenario pengujian.

6.2.1 Skenario Pengujian *Chunk size*

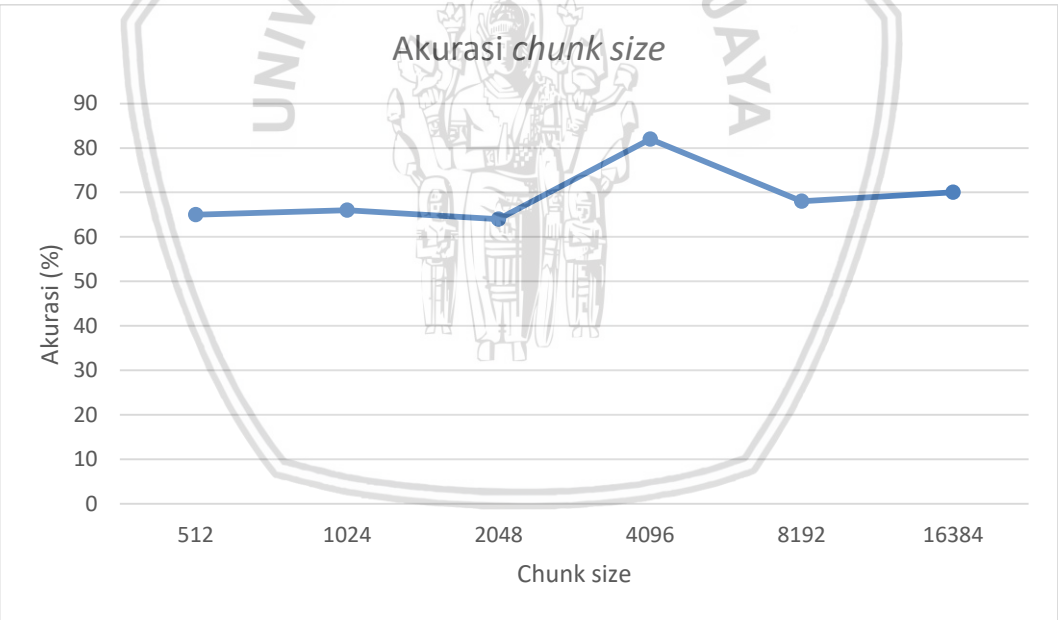
Skenario pengujian ini dilakukan untuk mendapatkan berapa *chunk size* terbaik yang menghasilkan akurasi paling besar. Dalam skenario ini variasi *chunk size* yang digunakan adalah 512, 1024, 2048, 4096, 8192, 16384. Parameter lainnya akan dibuat static dengan menggunakan *range* = 60. Kemudian untuk panjang data uji akan digunakan panjang dengan hasil akurasi paling besar, yaitu 15 detik. Hasil dari skenario pengujian *chunk size* disajikan pada Tabel 6.2.

Tabel 6.2 Hasil Pengujian *Chunk size*

Percobaan ke-*i*	Akurasi pada *chunk size*					
	512	1024	2048	4096	8192	16384
1	56	66	64	82	68	70
2	56	66	64	82	68	70
3	56	66	64	82	68	70
4	56	66	64	82	68	70
5	56	66	64	82	68	70
Rata – rata akurasi	56	66	64	82	68	70

6.2.2 Analisis Pengujian *Chunk size*

Proses analisis dilakukan dengan membuat grafik dari rata-rata akurasi yang telah didapatkan dari Tabel 6.2. Gambar 6.2 merupakan grafik akurasi dari hasil pengujian pada Tabel 6.2.



Gambar 6.2 Grafik Hasil Pengujian *Chunk size*

Dari hasil Tabel 6.2 dapat dilihat bahwa rata-rata akurasi tertinggi adalah 82% dan akurasi terbesar adalah 82% yang diperoleh dari *chunk size* 4096. *Chunk size* ini nantinya akan digunakan untuk melakukan pengujian jumlah *range*. Selain itu dapat dilihat pula bahwa akurasi mengalami peningkatan secara bertahap pada awal dan mengalami pelonjakan pada *chunk size* 4096, akan tetapi mengalami penurunan pada *chunk size* 8192. Dari hasil tersebut dapat disimpulkan bahwa nilai akurasi meningkat seiring meningkatnya *chunk size*, akan tetapi akan ada titik dimana akurasi akan menurun pada *chunk size* tertentu. Hal itu bisa terjadi karena

ukuran dari *chunk size* digunakan untuk mengidentifikasi semua frekuensi pada bagian tertentu dan disisi lain ukuran *chunk size* mempengaruhi banyak *chunk* yang digunakan untuk merepresentasikan informasi waktu. Oleh karena itu *chunk size* harus dibuat sepanjang mungkin untuk dapat melihat frekuensi dengan baik dan ukuran harus cukup pendek untuk dapat melihat informasi waktu dengan baik. *Chunk size* merepresentasikan frekuensi yang dapat diamati dengan baik, semakin besar maka semakin banyak frekuensi yang bisa diamati. Akan tetapi, jika *chunk size* besar maka jumlah *chunk* yang diperoleh semakin kecil. Semakin kecil jumlah *chunk* maka detail informasi waktu yang didapatkan menjadi kurang baik.

6.3 Pengujian Tingkat Akurasi Terhadap *Range*

Pengujian tingkat akurasi terhadap *range* menjelaskan tentang hasil skenario pengujian serta analisis dari hasil yang telah didapatkan.

6.3.1 Skenario Pengujian *Range*

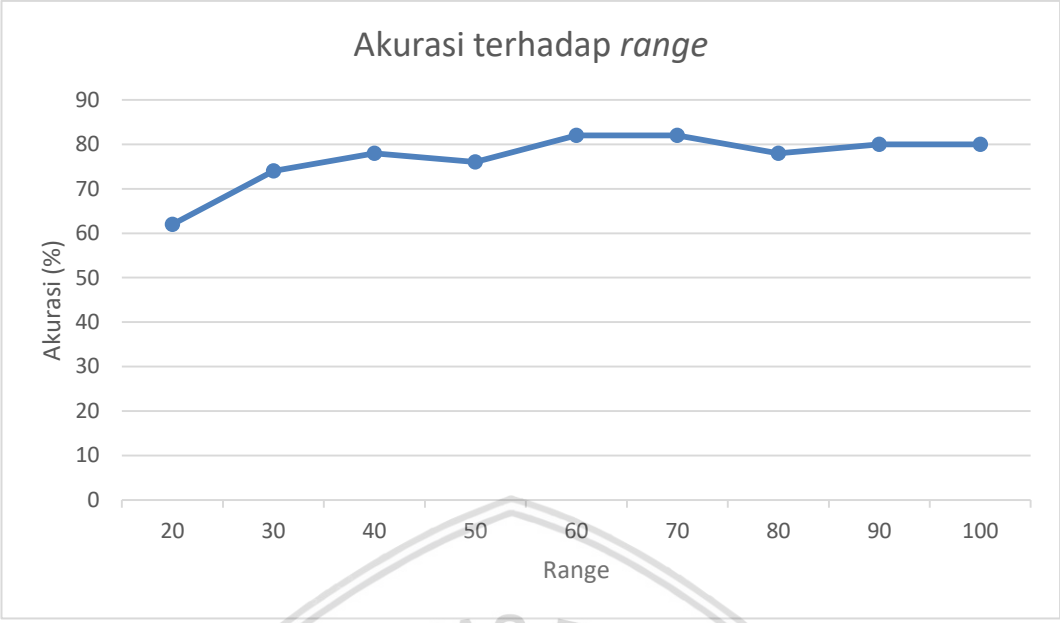
Pada pengujian akurasi terhadap *range* digunakan untuk mendapatkan *range* dengan nilai akurasi terbaik, dalam pengujian ini variasi *range* yang digunakan adalah 20, 30, 40, 50, 60, 70, 80, 90, 100. Kemudian untuk durasi data uji dipakai 15 detik yang merupakan durasi terbaik, dan untuk *chunk size* digunakan 4096 yang juga merupakan *chunk size* terbaik. Hasil dari skenario pengujian ini dapat dilihat pada Tabel 6.3.

Tabel 6.3 Hasil Pengujian *Range*

Percobaan	**Akurasi pada *range* (%)**								
Ke - *i*	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
1	62	74	78	76	82	82	78	80	80
2	62	74	78	76	82	82	78	80	80
3	62	74	78	76	82	82	78	80	80
4	62	74	78	76	82	82	78	80	80
5	62	74	78	76	82	82	78	80	80
Rata-rata akurasi	62	74	78	76	82	82	78	80	80

6.3.2 Analisis Pengujian *Range*

Pada Tabel 6.3 dapat dilihat bahwa rata-rata akurasi dan akurasi tertinggi adalah 82% yang didapat dari *range* 60 dan 70. Gambar 6.3 merupakan grafik yang menunjukkan hasil dari pengujian *range*. Pada Gambar 6.3 dapat dilihat bahwa nilai akurasi untuk dimulai dari *range* 70 cenderung stabil, tidak ada peningkatan atau penurunan yang signifikan. Tetapi dari pola yang ditampilkan pada Gambar 6.3 dapat dilihat bahwa setiap 4 range akan membentuk seperti bukit, dimana ada puncak dan lembah dari bukit tersebut.



Gambar 6.3 Grafik Hasil Pengujian Range

Range memiliki efek pada jumlah point yang didapatkan saat melakukan proses getpoints, semakin besar *range* yang digunakan maka jumlah *point* yang dihasilkan akan semakin sedikit, jika ini terjadi maka *hash* yang dihasilkan pun akan relatif sedikit. Akan tetapi semakin kecilnya *range* juga tidak bisa membuat hasil identifikasi membaik dikarenakan *range* harus mampu mendapatkan frekuensi dengan *magnitude* tertinggi yang tepat agar hasil identifikasi sistem baik.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis yang telah dilakukan pada penelitian identifikasi hadis dan surah dalam Al-Qur'an menggunakan algoritme shazam dengan suara maka diperoleh beberapa kesimpulan. Berikut beberapa kesimpulan yang didapatkan:

1. Implementasi algoritme shazam dapat digunakan untuk mengidentifikasi hadis dan surah dalam Al-Qur'an dengan melalui proses berikut:
   a. Ekstraksi numerik.
   b. Proses konversi.
   c. Prosee ekstraksi fitur.
   d. Proses filtering.
   e. Proses pembentukan hash.
   f. Proses pencocokan.
2. Nilai akurasi didapatkan dari penggunaan variasi kombinasi variabel uji. Variabel yang dijadikan variabel uji adalah durasi data uji, *chunk size* dan *range.* Dari hasil pengujian, nilai akurasi tertinggi sebesar 82%. Hasil tersebut didapatkan dari kombinasi 15 detik durasi data uji, *chunk size* = 4096 dan *range* = 60. Berdasarkan pengujian yang dilakukan, algoritma *shazam* merupakan metode yang cukup baik untuk indentifikasi suara.

7.2 Saran

Dari hasil penelitian, untuk identifikasi hadis dan surah dalam Al-Qur'an didapatkan beberapa saran untuk mengembangkan penelitian ini selanjutnya, berikut saran yang diberikan:

1. Pada penelitian selanjutnya diharapkan masukan data uji dapat berupa suara yang secara langsung diambil secara real time melalui *microphone.* Hal ini agar mempermudah *user* ketika menggunakan sistem ini.
2. Pada penelitian selanjutnya diharapkan melakukan penambahan pada *dataset* yang dipakai, dan menambah variasi pelantun pada *dataset*. Hal ini perlu dilakukan agar sistem mampu mengidentifikasi data yang variatif.
3. Diperlukan pengetahuan mengenai frekuensi-frekuensi penting pada file audio tentang Al-Qur'an dan hadis. Hal ini diperlukan agar tingkat akurasi pada proses identifikasi bertambah

DAFTAR PUSTAKA

Ahmed, A. H. & Abdo, S. M., 2017. Verification System for Quran Recitation Recordings. *International Journal of Computer Applications,* p. 6.

Amri, 2013. Autentisitas dan Gradualitas Al-Quran. *Jurnal substantia,* pp. 168-180.

Christophe, 2015. *How does Shazam work.* [Online] Available at: http://coding-geek.com/how-shazam-works/ [Diakses 10 12 2017].

Daulay, M. R., 2014. Studi Pendekatan Al-Qur'an. *Jurnal Thariqah Ilmiah,* pp. 31-45.

Hussainl, M. Z. & Parvin, K. N., 2016. Q-Format Data Representation and Its Arithmetic. *IJECT,* 7(2), pp. 57-62.

Irwan, M., 2015. Quaternion And It's Properties. *Jurnal MSA,* Volume 3, pp. 16-20.

Razak, Z. et al., 2008. Quranic Verse Recitation Recognition Module for Support in j-QAF Learning: A review. *International Journal of Computer Science and Network Security,* p. 10.

Rosidin, M. F. & Mahfudhoh, H. S., 2014. *Buku Siswa Al-Qur'an Hadis MA kelas X.* 1nd penyunt. Jakarta: Kementerian agama republik indonesia.

Safaat, T., 2016. *Implementasi Fast Fourier Transform Pada Pengenalan Nada Piano Berbasis Andorid,* s.l.: s.n.

Simanjuntak, T. H., Mahmudy, W. F. & Sutrisno, 2017. Implementasi Modified K-Nearest Neighbor Dengan Otomatisasi Nilai K Pada Pengklasifikasian Penyakit Tanaman Kedelai. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* Volume 1, pp. 75-79.

Wahidah, A. N. et al., 2012. Makhraj Recognition Using Speech Processing. *Computing and Convergence Technology (ICCCT),* pp. 689-693.

Wang, A. L.-C., 2003. *An Industrial-Strength Audio Search Algorithm.* [Online] Available at: https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf [Diakses 8 Juni 2017].