SIMULASI PENGONTROLAN NYALA LAMPU BERBASIS SISTEM PENGENALAN SUARA (*VOICE RECOGNITION*) DENGAN METODE *FAST FOURIER TRANSFORM* DAN *K-NEAREST NEIGHBOR*

SKRIPSI
KONSENTRASI KOMPUTASI CERDAS DAN VISUAL

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :

RENALDI PRIMASWARA PRASETYA
0810680057

PROGRAM STUDI TEKNIK INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2012

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT yang telah memberikan taufik dan hidayah-Nya kepada penulis sehingga penulis dapat menyelesaikan Laporan Tugas Akhir ini dengan judul "Pengontrolan Nyala Lampu Berbasis Sistem Pengenalan Suara (voice recognition) Dengan Menggunakan Metode Fast Fourier Transform dan K-Nearest Neighbor" dengan baik dan tepat.

Banyak sekali kesulitan dan hambatan yang penulis hadapi dalam penyusunan Laporan Tugas Akhir ini, akan tetapi berkat bantuan, bimbingan dan dorongan dari banyak pihak, akhirnya Laporan Tugas Akhir ini dapat penulis selesaikan sebagaimana mestinya. Untuk itu penulis mengucapkan terima kasih dan penghargaan yang setinggi-tingginya kepada banyak pihak yang telah memberikan bantuan baik pikiran maupun tenaga, waktu, sehingga Laporan Tugas Akhir ini dapat diselesaikan. Dalam kesempatan ini penulis mengucapkan banyak terima kasih kepada :

1. Allah SWT yang telah memberikan anugerah kekuatan, keselamatan dan kesehatan serta bimbingan-Nya sehingga penulis bisa menyelesaikan Laporan Tugas Akhir ini.
2. Ibu dan Ayah sebagai Orang Tua yang selalu memberikan semangat serta doa yang tiada henti-hentinya.
3. Bapak Ir. Sutrisno, M.T., selaku Ketua Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya (PTIIK UB) yang telah menyediakan berbagai fasilitas dalam penyusunan dan penyelesaian tugas akhir ini.
4. Ibu Rekyan Regasari MP, ST., MT. serta Ibu Dewi Yanti Liliana, S.Kom., M.Kom selaku dosen pembimbing I dan dosen pembimbing II yang telah banyak memberikan bimbingan, masukan dan arahan dalam penyusunan skripsi ini.
5. Sahabat dan seluruh teman-teman angkatan 2008 (TIF 08) yang selalu bersama dalam keceriaan.
6. Rekan-rekan Laboratorium Komputasi Cerdas dan Visual.

7. Semua pihak yang telah membantu penyelesaian skripsi ini baik secara langsung dan tidak langsung yang tidak dapat penulis sebutkan satu per satu.

Penulis membuka kesempatan yang seluas-luasnya untuk memperoleh kritik dan saran yang bersifat membangun demi perbaikan tugas akhir yang penulis buat. Akhirnya semoga tugas akhir yang penulis buat ini dapat bermanfaat bagi orang banyak.

Malang, Februari 2013

Penulis

ABSTRAK

Renaldi Primaswara Prasetya. 2013. Simulasi Pengontrolan Nyala Lampu Berbasis Sistem Pengenalan Suara (Voice Recognition) Dengan Metode Fast Fourier Transform dan K-Nearest Neighbor. Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang. Pembimbing: Rekyan Regasari MP., ST., MT. dan Dewi Yanti Liliana S.Kom., M.Kom.

Pengenalan suara merupakan cara interaksi antara manusia dengan sebuah sistem dimana interaksi user dengan sistem dapat dilakukan dengan memberikan inputan suara. Salah satu metode yang digunakan pada analisis spektrum pengenalan suara adalah *Fast Fourier Transform (FFT)*, yang melakukan pengolahan sinyal secara digital. *Fast Fourier Transform* adalah suatu algoritma yang digunakan untuk mewakili sinyal diskrit dalam domain waktu dan domain frekuensi dengan memanfaatkan sifat periodikal (konjugasi) dari transformasi fourier. Kemudian dari sinyal hasil proses FFT diambil dua ciri fitur yang mempresentasikan bentuk sinyal suara yaitu Root Mean Square dan Average Power Sprectum untuk dicocokkan menggunakan algoritma KNN. Sistem ini diimplementasikan menggunakan bahasa pemrograman C# yang terintegrasi dengan database MySQL. Pengujian yang digunakan yaitu pengujian kotak hitam dan pengujian akurasi. Hasil pengujian blackbox menunjukkan bahwa fungsionalitas sistem dapat berjalan dengan baik sesuai dengan daftar kebutuhan. Hasil pengujian akurasi terbaik terhadap nilai k yang berbeda yaitu 60,5% yang menunjukkan bahwa sistem berjalan dengan baik pada nilai k = 5 dan hasil pengujian akurasi terhadap jenis pembicara yang berbeda yaitu 60,5 % dan 56,67%, yang menunjukan bahwa dalam speech recognition, proses pengenalan kata yang diucapkan tidak mempedulikan identitas orang terkait.

Kata kunci : sinyal digital, pengenalan suara, *Fast Fourier Transform, KNN*.

ABSTRACT

Renaldi Primaswara Prasetya. 2013. *Flash Light Control Simulation Based Of Voice Recognition System (Voice Recognition) Using Fast Fourier Transform Method and K-Nearest Neighbor. Thesis of Informatic Engineering Study Program, Information Technology and Computer Science Program, Brawijaya University, Malang. Advisor:* Rekyan Regasari MP., ST., MT. *and* Dewi Yanti Liliana S.Kom., M.Kom.

Voice recognition is a way of interaction between people and a system where the user interaction with the system can be done by providing voice input. One of the methods used in spectrum analysis of speech recognition is the Fast Fourier Transform (FFT), which performs digital signal processing. Fast Fourier Transform is an algorithm that is used to represent a discrete signal in the time domain and the frequency domain by utilizing the properties of periodicals (conjugation) of the Fourier transformation. Then the results of the FFT of the signal taken two characteristic features of the present form of the sound signal is Root Mean Square and Average Power Sprectum to be matched using KNN algorithm.The system is implemented using the C # programming language integrated with MySQL databases. The test that used are blackbox testing and accuracy testing. Blackbox testing results show that the system's functionality can be well fit with a list of needs. The results of testing the best accuracy for different values of k is 60.5%, which indicates that the system is running well on the value of k = 5 and the results of testing the accuracy of the different types of speakers are 60.5% and 56.67%, which shows that the speech recognition, the recognition of the spoken word does not care about the identity of the person concerned.

***Key words** : digital signal, voice recognition, Fast Fourier Trasform, KNN.*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Manusia merupakan makhluk sosial yang memerlukan komunikasi dengaan sesamanya dalam kehidupan sehari – hari. Suara merupakan salah satu media komunikasi yang paling sering dan umum digunakan oleh manusia. Manusia dapat memproduksi suaranya dengan mudah tanpa memerlukan energi yang besar. Selain itu, dengan suara manusia dapat memberikan informasi maupun perintah. [JER-11:1]. Proses pengenalan kata atau suara merupakan salah satu fungsi dari *voice recognition. Voice recognition* dibagi menjadi dua jenis, yaitu *speech recognition* dan *speaker recognition*. Berbeda dengan *speaker recognition* yang merupakan pengenalan identitas yang diklaim oleh seseorang dari suaranya (ciri khusus dapat berupa intonasi suara, tingkat kedalaman suara, dan sebagainya), *speech recognition* adalah proses yang dilakukan komputer untuk mengenali kata yang diucapkan oleh seseorang tanpa mempedulikan identitas orang terkait. [KHR-11:1].

Pemanfaatan Perintah Suara (voice recognition) telah lama dikembangkan dan berlangsung sampai sekarang. Berbagai metode telah diterapkan dan masih terus dikembangkan oleh para peneliti di seluruh dunia. Banyak metode yang diterapkan dalam memanfaatkan voice recognition , salah satunya yaitu menggunakan metode *divide and conquer*. Pada penelitian aplikasi perintah suara oleh Melissa, Khrisna, yang menghasilkan akurasi sebesar 48%, digunakan metode *divide and conquer* yang merupakan algoritma yang berprinsip memecah-mecah permasalahan yang terlalu besar menjadi beberapa bagian kecil. [KHR-11:25].

Namun Algoritma penelusuran seperti devide and conquer memiliki keakuratan yang kurang serta memiliki kelemahan seperti lambatnya proses perulangan[KHR-11:129]. Proses pemanggilan *sub-rutin* yang berlebih, sehingga *call stack* penuh dari beberapa sub-rutin tersebut. Hal ini dapat menjadikan beban yang cukup signifikan pada prosesor. Kemudian lebih rumit untuk masalah yang

sederhana. Untuk pemecahan masalah yang relatif sederhana, algoritma sekuensial terbukti lebih mudah dibuat daripada algoritma divide and conquer.

Metode yang akan dibahas dalam penelitian ini adalah pencocokan pola suara menggunakan algoritma *Fast Fourier Transform* dan *K-Nearest Neighbor*. *Fast Fourier Transform (FFT)* merupakan salah satu metode untuk transformasi sinyal suara menjadi sinyal frekuensi. Artinya proses perekaman suara disimpan dalam bentuk digital berupa file WAV. *KNN* merupakan algoritma klasifikasi yang dalam pemrosesannya melakukan pencarian terhadap beberapa nilai terdekat dengan kelas yang ada, tidak seperti devide and conquer yang hanya memperhitungkan satu kelas saja.

Dengan suara manusia dapat memberikan informasi maupun perintah. Namun banyak persoalan yang terjadi ketika suara dimanfaatkan oleh sebuah sistem karena setiap orang memiliki ciri suara yang berbeda-beda. Suara merupakan modal utama yang dimiliki manusia untuk berkomunikasi dengan orang lain. Manusia dapat mengenali seseorang selain dari wajah (face recognition) juga dari suaranya (voice recognition).

Berdasarkan uraian permasalahan tersebut maka diperlukan dan dibuatlah sebuah aplikasi pengenalan perintah suara sebagai pengontrol nyala lampu dengan metode *Fast Fourier Transform* dan *K-Nearest Neighbor*.

1.2 Rumusan Masalah

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

1. Bagaimana membangun aplikasi perintah suara yang dapat menangkap sinyal suara dan kata yang diucapkan ?
2. Bagaimana mengimplementasikan algoritma FFT dan KNN sebagai pengolah sinyal suara dan proses pencocokan fitur suara dalam proses pengenalan suara ?
3. Bagaimana pengujian aplikasi *pengontrol nyala lampu* ?

1.3 Batasan Masalah

Agar permasalahan yang dirumuskan lebih terfokus, maka penelitian ini dibatasi oleh hal – hal sebagai berikut :

1. Ada 4 kata yang akan diteliti, dan dikenali dalam perintah suara ini, yaitu : Nyala, Mati, Terang, Redup.
2. File suara yang diproses adalah file berekstensi Waveform Audio (Wav).
3. Data suara disampling sebanyak 50 data.
4. Perekaman dilakukan dengan durasi waktu 2-3 detik.
5. Pengujian perintah suara dibatasi oleh beberapa faktor yaitu intonasi suara yang sedang, jarak bicara 1 cm tepat didepan mikropon eksternal, dan jenis pembicara yaitu seorang pria dan seorang wanita.
6. Pengujian *aplikasi* meliputi :
    a. Pengujian perangkat lunak yang terdiri dari pengujian *blackbox.*
    b. Pengujian kinerja algoritma KNN dengan melihat accuracy system berdasarkan penentuan jumlah tetangga terdekat dan jenis pembicara.

1.4 Tujuan

Adapun tujuan dari penelitian ini adalah :

1. Menerapkan metode *Fast Fourier Transform* sebagai pengolah sinyal suara pada aplikasi pengenalan kata.
2. Menerapkan dan mengetahui hasil pencocokan pola suara dengan metode *K-Nearest Neighbor* pada aplikasi pengenalan perintah suara.

1.5 Manfaat

Penelitian ini diharapkan dapat bermanfaat untuk berbagai pihak. Manfaat dari penelitian ini adalah sebagai berikut:

1. Bagi penulis:
    a. Mengaplikasikan ilmu yang didapat selama mengikuti perkuliahan di Teknik Informatika Universitas Brawijaya.

b. Mendapatkan pengetahuan tentang pembuatan *aplikasi* simulasi pengontrol nyala lampu dengan memanfaatkan teknologi Voice Recognition.

2. Bagi pengguna
   a. Sebagai media pembelajaran voice recognition yang menarik.
   b. Membantu para pengguna *aplikasi* dalam mengembangkan *aplikasi pengontrol nyala lampu* dengan menggunakan metode algoritma *Fast Fourier Transform* dan *KNN* atau metode algoritma yang lainnya.

1.6 Sistematika Pembahasan

Sistematika isi dan penulisan dalam skripsi ini antara lain :

Bab I Pendahuluan

Berisi tentang latar belakang masalah, perumusan masalah dan pokok-pokok bahasan, tujuan dan manfaat dari penelitian serta sistematika pembahasan.

Bab II Tinjauan Pustaka

Bab ini berisi tentang teori-teori yang mendasari tugas akhir ini, meliputi pengenalan suara, pengolahan sinyal suara, metode pencocokan pola suara, analisis terstruktur, komunikasi data, pengenalan mikrokontroler dan pengenalan program yang digunakan.

Bab III Metodologi Penelitian

Bab ini berisi tentang gambaran umum perancangan perangkat lunak / aplikasi simulasi pengontrol nyala lampu meliputi: studi literatur, analisis kebutuhan, perancangan perangkat lunak, implementasi perangkat lunak, pengujian perangkat lunak dan pengambilan kesimpulan dan saran.

Bab IV Perancangan

Membahas tentang analisis kebutuhan dan perancangan perangkat lunak yang sesuai dengan teori yang ada.

Bab V **Implementasi**

Membahas tentang implementasi dari sistem aplikasi.

Bab VI **Pengujian dan Analisis**

Memuat hasil pengujian dan analisis terhadap sistem yang telah direalisasikan.

Bab VII **Penutup**

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

1.7 Jadwal Penelitian

Tabel 1.1. Jadwal Pelaksanaan Penelitian

Kegiatan	Bulan 1	Bulan 2	Bulan 3	Bulan 4	Bulan 5	Bulan 6
Studi Literatur						
Analisis Kebutuhan						
Perancangan Perangkat Lunak						
Implementasi Perangkat Lunak						
Pengujian Perangkat Lunak						
Penulisan Laporan						

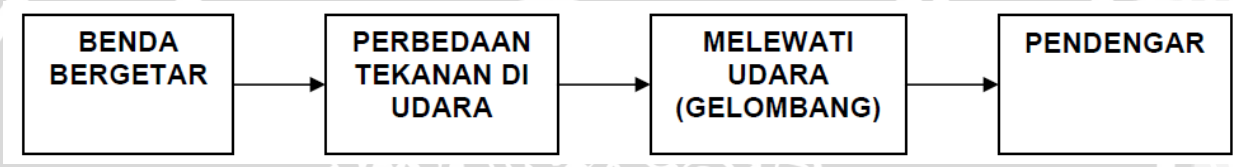
BAB II
DASAR TEORI

Pada bab ini akan diuraikan mengenai teori-teori dasar pembuatan perangkat lunak *aplikasi simulasi pengontrol nyala lampu* diantaranya sebagai berikut.

2.1 Landasan Teori

2.1.1 Suara

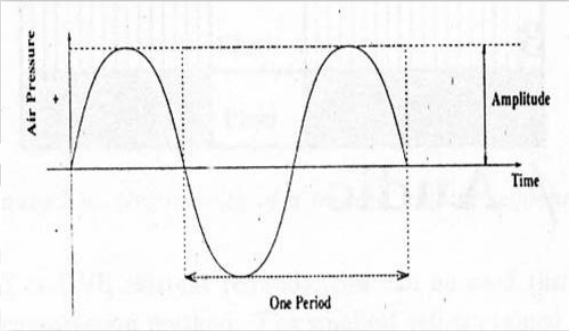
Suara adalah sesuatu yang dihasilkan oleh getaran yang berasal dari benda bergerak, sesuatu yang menghasilkan bunyi, dan suara adalah sebuah tekanan gelombang udara, maka memiliki nilai kontinu terhadap waktu (*analog*).



Gambar 2.1 Skema Suara

Sumber : [JER-11:8]

Suara berhubungan erat dengan rasa "mendengar". Suara/bunyi biasanya merambat melalui udara. Suara/bunyi tidak bisa merambat melalui ruang hampa. Gelombang suara bervariasi dalam tingkatan tekanan suara (*amplitudo*) dan dalam frekuensi. Jumlah waktu yang diperlukan untuk terjadinya suatu getaran atau gelombang dinamakan perioda (T). Sedangkan jumlah gelombang yang terjadi setiap detik dinamakan ferkuensi (f) dengan satuan m/dt (Hz). [JER-11:8].



Gambar 2.2. Gelombang suara

2.1.2 Konversi Sinyal Analog Ke Sinyal Digital

Sinyal yang terdapat pada kehidupan sehari-hari disebut dengan sinyal analog. Untuk dapat memproses sinyal analog tersebut, maka harus diubah terlebih dahulu menjadi sinyal digital atau sinyal diskrit. Sinyal analog memiliki sifat kontinu pada domain waktu dan amplitudo, sedangkan sinyal digital memiliki sifat diskrit pada domain waktu dan amplitudo. Untuk mengubah sinyal analog menjadi sinyal digital maka diperluas proses "digitalisasi". Proses ini terdiri dari 3 tahap, yaitu :

1) Sampling

Sinyal suara merupakan sinyal yang tidak terbatas dalam domain waktu (infinite time interval). Suara manusia akan menghasilkan sinyal analog yang terus kontinu. Gelombang suara analog tidak dapat langsung direpresentasikan pada komputer. Terlebih dahulu komputer mengukur amplitudo pada satuan waktu tertentu untuk menghasilkan sejumlah angka. Tiap satuan pengukuran dinamakan " Sampel". Untuk keperluan pemerosesan dalam transformasi Fourier sinyal suara harus dibentuk dalam potongan-potongan waktu. Proses sampling dilakukan dengan didasarkan asumsi bahwa sinyal percakapan berada pada daerah frekuensi 300-3400 Hz. Teori sampling Nyquist meyebutkan bahwa frekuensi sampling (sampling rate) minimal harus dua kali lebih tinggi dari frekuensi maksimum yang akan disampling. [NUR-09:23].

$$f_s \geq 2xf_h$$

Keterangan:

F_s = Frekuensi sampling

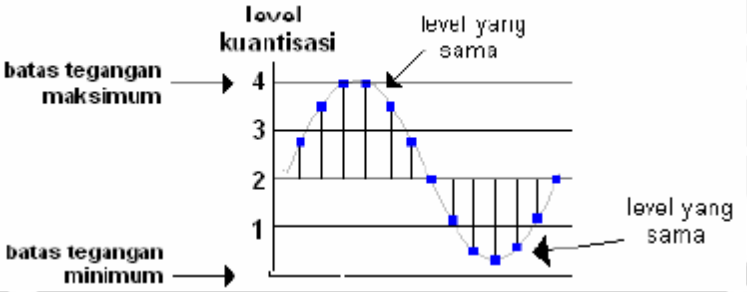
F_h = Frekuensi sinyal analog tertinggi

Jika sinyal sampling kurang dari dua kali frekuensi maksimum sinyal yang akan disampling, maka akan timbul efek aliasing. Aliasing adalah suatu efek di mana sinyal yang dihasilkan memiliki frekuensi yang berbeda dengan sinyal aslinya[NUR-09:24].

2) *Kuantisasi*

Sinyal digital merupakan sebuah deretan angka (sampling) yang diwakili oleh beberapa digit untuk menentukan keakuratan. Proses melakukan konversi sinyal

yang telah disampling menjadi sinyal digital yang diwakili oleh sebuah nilai dengan jumlah digit tertentu disebut kuantisasi. [NUR-09:25].



Gambar 2.3 Ilustrasi Proses Kuantisasi

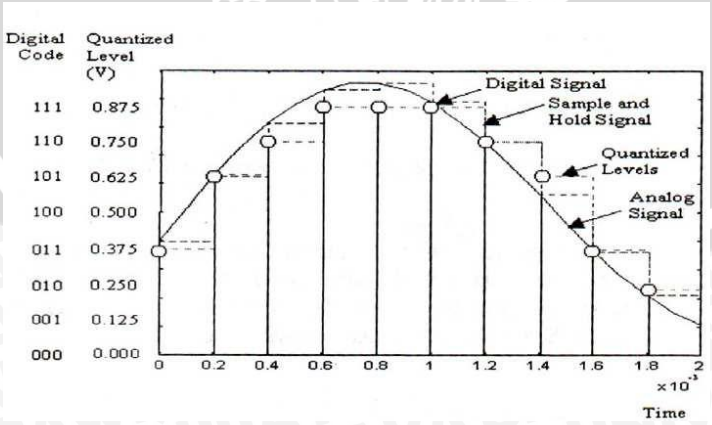
Sumber : [NUR-09:25]

Semakin banyak level yang dipakai semakin akurat pula data sinyal yang disimpan tetapi akan menghasilkan ukuran data yang besar dan proses yang lama. Gambar 2.5 adalah contoh proses kuantisasi yang menggunakan empat level. Pada level 4 terdapat empat buah sinyal yang menempati level yang sama, artinya keempat sinyal tersebut dikelompokkan menjadi level yang sama walupun tingginya berbeda. Demikian pula pada level 1. Selisih antara nilai kuantisasi dengan sinyal sebenarnya disebut kesalahan kuantisasi (error quantization).

$$e_q(n) = x_q(n) - x(n)$$

3) *Coding / Proses Pengkodean*

Proses pengkodean adalah proses pemberian kode untuk tiap-tiap data sinyal yang telah terkuantisasi berdasarkan level yang ditempati.



Gambar 2.4 Pengkodean

Sumber : [NUR-09:25]

Dengan level kuantisasi 0.000 sampai 0.875 dengan interval 0.125 didapati 8 buah nilai, jadi ada 8 nomor kode, yang dimulai dengan 0,1,…sampai 8. Dalam menuliskan 8 buah nilai kode dibutuhkan 3 buah digit. Karena satu digit, hanya bisa menuliskan kondisi 0 dan 1. Dua digit, bisa menuliskan empat kondisi 00, 01, 10 dan 11. Tiga digit bisa menuliskan delapan kondisi dan empat digit bisa menuliskan kondisi. [NUR-09:26]. Sehingga dapat dituliskan:

$$L = 2^n$$

Keterangan :

L = jumlah nilai kuantisasi yang digunakan

n = jumlah digit

Apabila Gambar 2.6 nomor-nomor kode dituliskan secara digital, maka diperoleh tabel sebagai berikut :

Tabel 2.1 Kuantisasi

Nilai kuantisasi	**Nomor Kode**	**Kode**
0.000	0	000
0.125	1	001
0.250	2	010
0.375	3	011
0.500	4	100
0.625	5	101
0.750	6	110
0.875	7	111

Sumber : [NUR-09:26]

Dengan Tabel 2.1 dapat dilakukan proses pengkodean terhadap sinyal yang telah dikuantisasi. Hasilnya dapat dilihat pada Tabel 2.2.

Tabel 2.2 Pengkodean

iT	Nilai Kuantisasi	Nomor Kode	Kode Biner
0	0.375	3	011
0.2	0.625	5	101
0.4	0.750	6	110
0.6	0.875	7	111
0.8	0.875	7	111
1.0	0.875	7	111
1.2	0.750	6	110
1.4	0.625	5	101
1.6	0.375	3	011
1.8	0.250	2	010
2.0	0.125	1	001

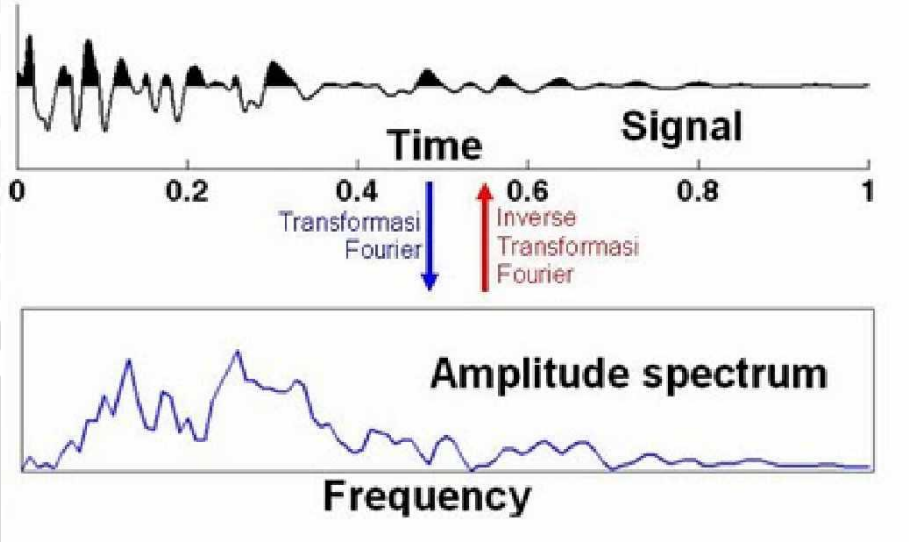
Sumber : [NUR-09:27]

4) *Banyak bit (bit depth)*

Byte merupakan unit informasi digital dalam komputasi dan telekomunikasi. 1 byte terdiri dari 8 bit. Banyaknya bit yang digunakan untuk merepresentasikan level kuantisasi atau nilai amplitudo dinamakan bit depth. Semakin besar jumlah bit yang digunakan, maka jangkauan level amplitudo akan semakin halus dengan nilai interval yang semakin kecil. Untuk bit depth sebesar 8 bit, jumlah level yang memungkinkan adalah 256 level, dengan jangkauan nilai level 0 s/d 255. Semakin besar nilai sampling rate dan bit depth, maka tingkat akurasi sinyal digital terhadap sinyal asal (sinyal analog) menjadi semakin besar dan kualitas sinyal digital semakin baik. Akan tetapi pemrosesannya membutuhkan penyimpanan data dalam jumlah yang besar. Jika pemrosesan sinyal dilakukan pada perangkat yang canggih, penyimpanan data dalam jumlah besar tidak akan menjadi masalah.[SAM-12:14]

2.1.3 FFT (Fast Fourier Transform)

Transformasi *Fourier* adalah suatu model transformasi yang memindahkan domain spasial atau domain waktu menjadi domain frekuensi.



Gambar 2.5 Transfrormasi Fourier

Sumber : [KHR-11:24]

Transformasi *Fourier* merupakan suatu proses yang banyak digunakan untuk memindahkan domain dari suatu fungsi atau obyek ke dalam domain frekuensi. Di dalam pengolahan citra digital atau sinyal suara, transformasi *fourier* digunakan untuk mengubah domain spasial pada citra/sinyal menjadi domain frekunsi. Analisa-analisa dalam domain frekuensi banyak digunakan seperti filtering. Dengan menggunakan transformasi *fourier*, sinyal suara atau citra dapat dilihat sebagai suatu obyek dalam domain frekuensi. FFT (*Fast Fourier Transform*) adalah teknik perhitungan cepat dari DFT (*Discrete Fourier Transform*). Transformasi fourier diskrit atau disebut dengan *Discrete Fourier Transform* (DFT) adalah model transformasi fourier yang dikenakan pada fungsi diskrit, dan hasilnya juga diskrit. FFT adalah DFT dengan teknik perhitungan yang cepat dengan memanfaatkan sifat periodikal (konjugasi) dari transformasi *fourier*. [KHR-11:24]

$$F(u) = \frac{1}{N}\sum_{x=0}^{x=N-1} f(x)\,(\cos(\frac{2\pi ux}{N}) - jsin(\frac{2\pi ux}{N}))$$

2.1.4 Normalisasi

Dalam setiap proses sampling sinyal suara akan diperoleh hasil dimana setiap sprektum akan mempunyai bentuk yang berbeda-beda dan level amplitudo yang berbeda-beda pula. Oleh karena itu untuk menyamakan nilai amplitudo tertinggi dan terendah dari tiap tiap sprektum makan digunakan proses normalisasi.

(n-mindata / maxdata –mindata) x (max – min)

2.1.5 Algoritma K-Nearest Neighbor (KNN)

Dalam mengklasifikasikan sekumpulan data sangat banyak cara dan algoritma yang bisa digunakan. Salah satu algoritma yang paling sering digunakan adalah KNN (K-Nearest Neighbor).

KNN adalah sebuah metode klasifikasi terhadap sekumpulan data berdasarkan pembelajaran data yang sudah terklasifikasikan sebelumya. KNN termasuk dalam golongan supervised learning, dimana hasil query instance yang baru diklasifikasikan berdasarkan mayoritas kedekatan jarak dari kategori yang ada dalam KNN. Nantinya kelas yang baru dari suatu data akan dipilih berdasarkan grup kelas yang paling dekat jarak vektornya.

Tujuan dari algoritma ini adalah mengklasifikasikan obyek baru berdasarkan atribut dan training sample. Diberikan titik query, akan ditemukan sejumlah k obyek atau (titik training) yang paling dekat dengan titik query. Klasifikasi menggunakan voting terbanyak diantara klasifikasi dari k obyek. Algoritma k-nearest neighbor (KNN) menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari query instance yang baru.

Dekat atau jauhnya tetangga biasanya dihitung berdasarkan Euclidean Distance. Jarak euclidean berfungsi menguji ukuran yang bisa digunakan sebagai interpretasi kedekatan jarak antara dua obyek. yang direpresentasikan sebagai berikut :

$$D(a,b) = \sqrt{\sum_{k=1}^{d} (a_k - b_k)^2}$$

Dimana D(a,b) adalah jarak skalar dari dua buah vektor a dan b dari matrik berukuran D dimensi. Data training dalam perintah suara adalah sebanyak 5 kali tiap kata yang disimpan ke dalam beberapa kelas yaitu nyala, mati, terang, dan redup.

Langkah-langkah untuk menghitung metode K-Nearest Neighbor :

1. Menentukan parameter K (jumlah tetangga paling dekat).
2. Menghitung kuadrat jarak euclid (query instance) masing–masing obyek terhadap data sampel yang diberikan.
3. Kemudian mengurutkan objek–objek tersebut kedalam kelompok yang mempunyai jarak euclidian terkecil.
4. Mengumpulkan kategori Y (Klasifikasi nearest neighbor).
5. Dengan menggunakan kategori nearest neighbor yang paling mayoritas maka dapat dipredisikan nilai query instance yang telah dihitung.

2.1.6 Proses Pengolahan Dan Pencocokan Suara

Secara umum pemrosesan sinyal suara dimulai dengan inputan berupa suara dengan menggunakan media berupa microphone. Sinyal yang dihasilkan oleh suara manusia merupakan suatu sinyal analog. Agar sinyal analog dapat diolah oleh komputer, maka terlebih dahulu harus dilakukan perubahan menjadi sinyal digital. Perubahan dari sinyal digital menjadi sinyal analog melalui 3 tahapan yaitu sampling, kuantisasi, dan pengkodean. Kemudian dengan menggunakan Fast Fourier Transform, sinyal digital diproses agar berubah dari domain waktu menjadi domain frekuensi, kemudian dilakukan ekstrasi fitur yang bertujuan mendapatkan suatu gelombang yang dapat mewakili keseluruhan gelombang dan dapat mempercepat proses perhitungan. Kemudian, nilai suatu sinyal akan dicocokkan menggunakan algoritma KNN agar dapat menjalankan sebuah masukan perintah suara.

2.2 Kajian Pustaka

2.2.1 Ekstrasi Fitur

Untuk mendapatkan data yang akurat dan konsisten dari setiap sampel, digunakan suatu metode ekstraksi fitur/ciri sinyal suara dan untuk mencapai suatu *feature extraction*, akan digunakan metode analisis akustik untuk mengurangi suara menjadi beberapa set parameter dan teknik statistik untuk mengambil suara dan mengklasifikasikannya. Berikut adalah cara mengukur nilai-nilai parameter fitur akustik dari berbagai suara masing-masing yang diambil dari *feature extraction* dan dijadikan sebagai sample pengenalan kata yaitu *Loudness* (Kekerasan Suara), dan Average Power Spectrum (AVG).

2.2.1.1 *Loudness* (Kekerasan Suara)

Loudness diambil dari suara untuk mengukur tingkat kekerasan suara tiap-tiap data sampel suara. Sinyal didekati dengan perhitungan *Root Mean Square* (RMS). Dalam matematika, RMS dikenal sebagai rata-rata kuadrat, adalah ukuran statistik besarnya suatu kuantitas yang bervariasi. RMS ini berguna ketika terdapat variasi yang positif dan juga negatif, misalnya sinusoid. RMS digunakan dalam berbagai bidang, dan paling sering digunakan pada bidang sinyal. RMS dalam fitur ini menghitung RMS dalam domain frekuensi/FFT dengan perhitungan.

$$\mathrm{RMS}_j = \sqrt{\frac{\sum_{i=1}^{M} |U_{ij}|^2}{M}}, 1 \leq j \leq N$$

2.2.1.2 Average Power Sprectum (AVG)

Average Power Spectrum adalah proses untuk mengukur rata-rata daya dari sebuah sinyal periodik deterministik. Jenis sinyal kontinu dalam domain waktu, tetapi menghasilkan discrete power spectrum. Sebuah sinyal terdiri dari sinusoid contohnya sinyal listrik yang memiliki energy yang tak terbatas, tetapi rata-rata dayanya terbatas. Pertama-tama mengukur rata-rata daya menggunakan periodogram spectrum object dengan metode window hamming, dan menghitung rata-rata daya spektrum. Berikut rumus untuk metode window hamming pada sample ke- N.

$$W(n) = 0.54 - 0.46 \cos(\frac{2\pi n}{N-1}) , 0 \leq n \leq N-1$$

Setelah proses *windowing* menggunakan *window hamming,* konversi nilai tersebut kedalam suatu nilai logaritmik.

$$Avgpower = 10 * \log10 (\frac{W(n)}{2})$$

2.2.2 Algoritma Divide and Conquer

Pada penelitian yang dilakukan oleh Melissa dan Khrisna, proses pencocokan sinyal suara dilakukan menggunakan algoritma *Divide and Conquer*. *Divide and Conquer* merupakan algoritma yang berprinsip memecah-mecah permasalahan yang terlalu besar menjadi beberapa bagian kecil sehingga mudah untuk diselesaikan. Langkah-langkah umum algoritma *Divide and Conquer* :

- *Divide* : membagi masalah menjadi beberapa sub-masalah yang memiliki kemiripan dengan masalah semula namun berukuran lebih kecil (idealnya berukuran hampir sama).
- *Conquer* : memecahkan atau menyelesaikan masing-masing sub-masalah secara rekursif.
- *Combine* : menggabungkan solusi masing-masing sub-masalah sehingga membentuk solusi masalah semula.

Objek masalah yang dibagi adalah masukan (input) atau instances yang berukuran n: tabel (array), matriks, dan sebagainya, bergantung pada masalahnya. Pada implementasinya metode devide and conquer dimanfaatkan untuk proses pencocokan suara melalui dua fitur suara yaitu Root Mean Square dan Average Power Sprectum. Dari pencocokan sinyal suara menggunakan metode devide and conquer terhadap dua fitur tersebut, didapatkan nilai akurasi yang cukup rendah yaitu sebesar 48%.

2.3 Perancangan Perangkat Lunak

Perancangan merupakan penghubung antara spesifikasi kebutuhan dan implementasi. Tujuan perancangan adalah menghasilkan model atau representasi entitas yang akan dibangun. Perancangan dibagi dua yaitu [HAR-04:406] :

1. *Preliminary design* biasa disebut dengan perancangan sistem. Tahap ini mentranformasikan kebutuhan menjadi arsitektur perangkat lunak.
2. *Detail design,* pada pendekatan berorientasi objek disebut perancangan objek. Tahap ini memfokuskan pada perbaikan menjadi representasi arsitektur yang menuntun ke representasi struktur data dan algoritma rinci.

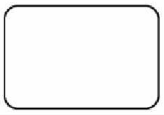
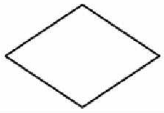
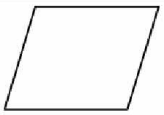
Pada tahap perancangan di skripsi ini, digunakan pemodelan dengan menggunakan *sequence diagram* dan *class diagram.*

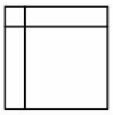
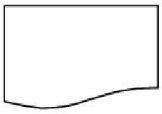
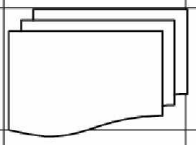
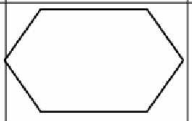
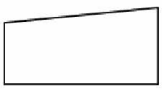
2.3.1 Bagan Alir Sistem *(Flow Chart)*

Bagan alir sistem (*flowchart*) merupakan bagan yang menunjukkan arus pekerjaan dari sistem secara keseluruhan, menjelaskan urutan dari prosedurprosedur yang ada di dalam sistem serta menunjukkan apa yang dikerjakan di dalam sistem. [KHR-11:31]

Tabel 2.3. Simbol - simbol Flowchart

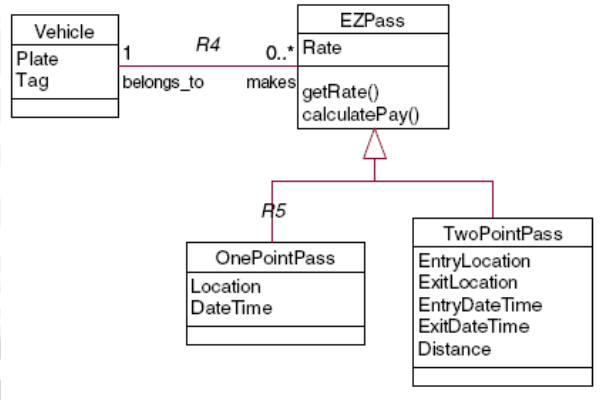
Sumber : [KHR-11:31]

No	Simbol	Nama	Deskripsi
		- Process - Proses	- An operation or action step - Langkah operasi atau tindakan -
		- Alternate process - Proses alternatif	- An alternate to the normal process step -
		- Decision	- A question or branch in the process - Proses pertanyaan atau cabang - kondisi yang akan menghasilkan beberapa kemungkinan jawaban/aksi
		- Data I/O	- Indicates data inputs and outputs to and from process - Proses memasukkan atau mengeluarkan data
		- Predifined process	- A formally defined sub-process - Sub proses yang ditetapkan

		- Internal storage	- Data storage in memory - Penyimpanan data
		- Document	- A document or report - Sebuah dokumen atau laporan - symbol yang menyatakan input berasal dari dokumen dalam bentuk kertas atau output dicetak ke kertas
		- Multi-Document	- Same as document, except, well, multiple documents - Dokumen yang sama, dokumen rangkap
		- Terminator	- A strart or stop point in process - Permulaan atau akhir proses
		- Preparation	- A preparation or set-up process step - Langkah persiapan atau proses set-up
		- Manual Input	- Manually input into a system - Memasukkan data secara manual

2.3.2 Diagram kelas

Diagram kelas merupakan diagram paling umum dipakai di semua pemodelan berorientasi objek. Pemodelan kelas menunjukan kelas- kelas yang ada di sistem dan hubungan antar kelas – kelas itu, atribut – atribut dan operasi – operasi di kelas – kelas. Diagram kelas menunjukan aspek statik sistem terutama untuk mendukung kebutuhan fungsionalitas sitem. Kelas di diagram kelas dapat secara langsung diimplementasikan ke dalam bahasa pemrograman berorientasi objek yang secara langsung mendukung bentukan kelas [HAR-04:277]. Contoh diagram kelas ditunjukan pada gambar 2.6.



Gambar 2.6 Diagram Kelas

Sumber : [LIU-06:19]

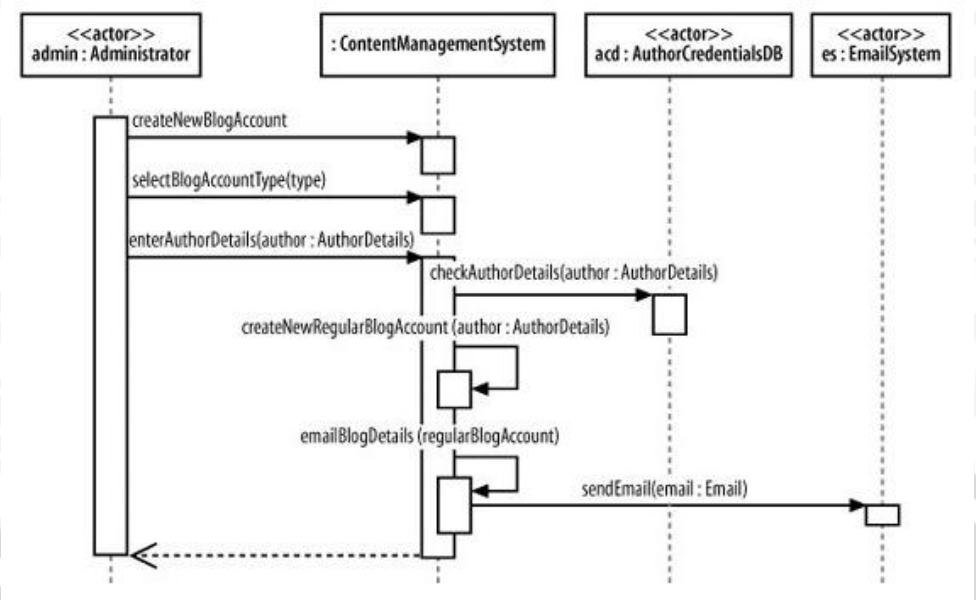
Elemen – elemen esensi di diagram kelas adalah sebagai berikut :

1. Kelas
2. Antarmuka (*Interfaces*)
3. Kolaborasi
4. Hubungan (*Relationship*) seperti kebergantungan, generalisasi dan asosiasi.

2.3.3 Diagram Sekuen

Diagram sekuen digunakan untuk memodelkan skenario penggunaan. Skenario penggunaan adalah barisan kejadian yang terjadi selama satu eksekusi sistem. Diagram sekuen menunjukan objek sebagai garis vertikal dan tiap kejadian sebagai panah horisontal dari objek pengirim ke objek penerima [HAR-04:309]. Contoh diagram sekuen ditunjukan pada gambar 2.7. Diagram sekuen terdiri dari beberapa elemen yaitu [ALH-03:146] :

1. ***Class roles***, yaitu elemen yang digunakan untuk level spesifikasi dari kolaborasi (*specification-level collaborations*).
2. ***Specific objects***, yaitu objek yang sesuai dengan class roles dan objek lain yang digunakan untuk *instance-level collaborations*.



Gambar 2.7 Diagram Sekuen

Sumber : [HAM-06]

3. ***Lifeline***, yaitu elemen yang merepresentasikan keberadaan elemen dari waktu ke waktu.
4. ***Activations***, yaitu elemen yang merepresentasikan waktu sebuah elemen beroperasi.

2.4 Implementasi

Implementasi perangkat lunak dilakukan untuk merealisasikan desain dari perangkat lunak menggunakan bahasa pemrograman berorientasi objek (*Object Oriented Programming Languages/OOP*).

2.5 Pengujian Perangkat Lunak

Pengujian adalah proses pemerikasaan atau evaluasi sistem atau komponen sistem secara manual atau otomatis untuk memverifikasi apakah sistem memenuhi kebutuhan – kebutuhan yang dispesifikasikan atau mengidentifikasi perbedaan – perbedaan antara hasil yang diharapkan dengan hasil yang terjadi.Awalnya, pengujian diartikan sebagai aktivitas yang dapat atau hanya dilakukan setelah pengkodean (Kode program selesai). Namun, pengujian seharusnya dilakukan

dalam skala lebih luas. Pengujian dapat dilakukan begitu spesifikasi kebutuhan telah dapat didefinisikan [HAR-04:569].

Terdapat dua teknik pengujian berdasarkan ketersediaan logik sistem, yaitu *black box testing dan white box testing* [HAR-04:577]. Pada skripsi ini menggunakan salah satu teknik pengujian tersebut yaitu *blackbox testing*.

2.5.1 *Black Box Testing*

Black box testing adalah teknik pengujian yang menguji hanya berdasarkan kebutuhan dan spesifikasi. Black box testing juga disebut sebagai behavioral testing dan berfokus pada kebutuhan fungsi dari perangkat lunak [PRE-01]. Proses umum yang terjadi pada black box testing yaitu :

1. Kebutuhan atau spesifikasi dianalisa terlebih dahulu.
2. Penentuan input valid terpilih berdasarkan spesifikasi untuk menentukan perangkat lunak berjalan dengan benar. Input yang tidak valid juga harus dipilih untuk memverifikasi bahwa perangkat lunak dapat mendeteksinya dan menanganinya dengan baik.
3. Penentuan output yang diharapkan sesuai dengan input yang telah dipilih.
4. Pengujian dibuat dengan input yang telah dipilih.
5. Pengujian dijalankan.
6. Output yang sebenarnya dibandingkan dengan output yang diharapkan.
7. Penentuan dibuat menyangkut perangkat lunak berfungsi sesuai dengan spesifikasi yang telah ditentukan.

2.5.2 Pengujian Akurasi

Sistem pendukung keputusan / decision support system (DSS) adalah bagian dari sistem informasi berbasis komputer (termasuk sistem berbasis pengetahuan (manajemen pengetahuan) yang dipakai untuk mendukung pengambilan keputusan. DSS dapat juga dikatakan sebagai sistem komputer yang mengolah data menjadi informasi untuk mengambil keputusan dari masalah semi-terstruktur yang spesifik.

Pengujian akurasi sistem dilakukan dengan aturan *True Positif, True Negatif* berdasarkan hasil keputusan yang bersifat actual atau keputusan decision maker. Nilai akurasi dihitung menggunakan rumus accuracy.[HLA-12]

Jika dimisalkan *True Positive* adalah jumlah kata perintah yang berhasil dikenali oleh sistem dan sesuai dengan hasil decision maker, *False Positive* adalah jumlah kata perintah yang berhasil dikenali oleh sistem tetapi tidak relevan/tidak sesuai dengan hasil decision maker dan *False Negative* adalah jumlah kata perintah yang relevan/sesuai dengan decision maker tetapi tidak dikenali oleh sistem, serta *True Negatif* adalah jumlah kata perintah yang tidak dikenali oleh sistem dan decision maker, maka rumus *accuracy* ditunjukkan pada persamaan berikut:

TP : *True Positive* (Jika prediksi positif dan benar)

TN : *True Negative* (Jika prediksi negatif dan benar)

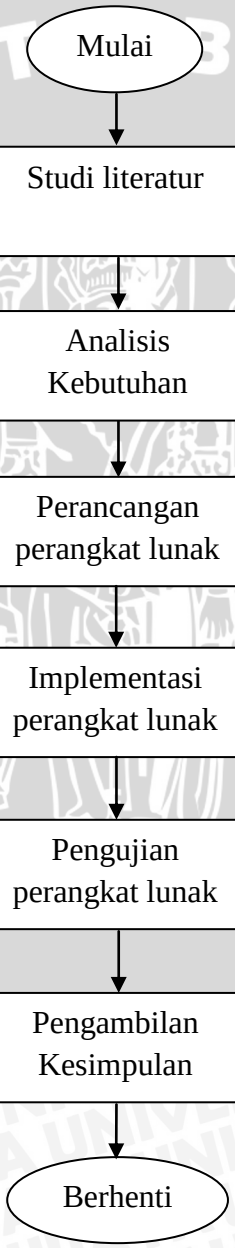
FP : *False Positive* (Jika prediksi positif dan salah)

FN : *False Negative* (Jika prediksi negatif dan salah)

Accuracy = (TP + TN) / (TP + TN + FP + FN)

BAB III
METODOLOGI PENELITIAN

Langkah – langkah yang dilakukan dalam penelitian ini yaitu : studi literatur, analisis sistem, perancangan perangkat lunak, implementasi perangkat lunak dan pengujian perangkat lunak. Gambar 3.1 menunjukan diagram alir metode penelitian.



Gambar 3.1 Diagram Alur Metode Penelitian

Penjelasan dari masing – masing proses pada diagram alir Metode Penelitian adalah sebagai berikut :

3.1. Studi Literatur

Studi literatur dilakukan untuk mempelajari teori yang digunakan dalam pembuatan perangkat lunak simulasi *Pengontrol Nyala Lampu*. Teori – teori yang dipelajari yaitu tentang :

1. *Suara*
2. Pengenalan Suara Pada Manusia
3. Proses Pengolahan Dan Pencocokan Suara
4. Konversi Sinyal Analog ke Sinyal Digital
5. FFT
6. Ekstrasi Ciri
7. Algoritma KNN clasification

3.2. Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dalam pembuatan perangkat lunak *Pengontrol Nyala Lampu*. Langkah - langkah analisis kebutuhan perangkat lunak *Pengontrol Nyala Lampu* yaitu (1) pembuatan *aplikasi* secara global, (2) identifikasi aktor, (3) identifikasi kebutuhan, (4) pembuatan *use case* diagram.

3.2.1. Pembuatan *Aplikasi*

Proses pengenalan kata dan simulasi pada sistem harus melalui beberapa tahap proses agar kata dapat dikenali oleh sistem. Suara yang telah direkam berupa sinyal analog diolah menjadi sinyal digital untuk diubah menjadi data diskrit. Data suara hasil rekaman kembali diproses dalam pengolahan sinyal suara, di ekstraksi ciri fiturnya untuk mengetahui perbedaan dan ciri-ciri tiap suara agar memudahkan saat proses pengenalan kata. Setelah itu barulah sistem pengenalan kata akan mengenali ucapan dari pengguna setelah sistem memiliki *data training* yang akan dijadikan perbandingan. Hasil pengenalan kata dapat diproses lebih lanjut atau di implementasikan ke berbagai aplikasi.

3.2.2. Identifikasi Aktor

Aktor adalah pemakai sistem, dapat berupa orang atau sistem terotomastisasi lain [HAR-04:268]. Tahapan identifikasi aktor dilakukan untuk mengetahui aktor - aktor yang terlibat dalam sistem perangkat lunak.

3.2.3. Identifikasi Kebutuhan

Identifikasi kebutuhan terdiri dari identifikasi 2 kebutuhan yaitu :

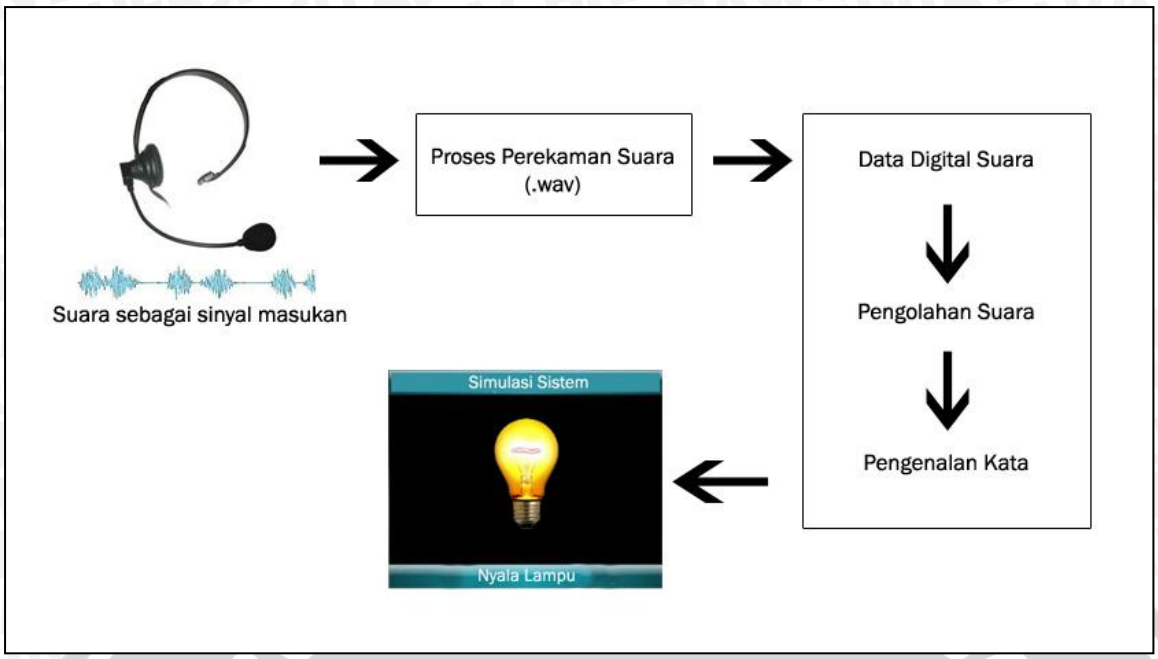
a. Kebutuhan fungsional
b. Kebutuhan non fungsional

3.2.4. Pembuatan *Use Case* Diagram

Fungsi – fungsi yang disediakan oleh sistem dan interaksi aktor dengan sistem selanjutnya akan dimodelkan dalam *Use case* diagram.

3.3. Perancangan Perangkat Lunak

Perancangan merupakan tahap lanjutan dari analisis sistem dimana pada perancangan sistem digambarkan rancangan sistem yang akan dibangun sebelum dilakukan pengkodean kedalam suatu bahasa pemograman. Dalam perancangan sistem tidak lepas dari hasil analisis, karena dari hasil analisis sistem baru dapat dibuat suatu perancangan sistem. Desain umum yang akan diaplikasikan bertujuan untuk memberikan gambaran secara umum kepada penggunaan tentang sistem yang akan dibangun.



Gambar 3.2 Desain Umum Rancangan Sistem

Fungsi masing-masing bagian dalam rancangan sistem di atas adalah sebagai berikut.

1. Suara merupakan sinyal *input* yang digunakan sistem.
2. *Microphone* digunakan sebagai media *recording*, yang langsung dihubungkan dengan komputer.
3. Sinyal suara hasil perekaman kemudian diubah dari bentuk analog ke digital menggunakan bantuan soundcard dalam *accessories windows* kemudian disimpan dalam bentuk *waveform (.wav).*
4. Pengolahan suara dilakukan untuk mendapatkan sinyal suara dalam domain frekuensi menggunakan algoritma *Fast Fourier Transform* dan Ekstrasi Fitur suara yaitu kekerasan (RMS) dan daya (AVG).
5. Setelah diolah , suara akan dikenali oleh sistem sebagai suatu perintah dimana cara pengenalan kata tersebut menggunakan algoritma K-Nearest Neighbor.
6. Hasil pengenalan perintah suara ditampilkan dalam bentuk simulasi di dalam aplikasi.

3.4. Implementasi Perangkat Lunak

Setelah melakukan perancangan maka Aplikasi Pengontrol Nyala Lampu akan dibuat mengacu pada rancangan perangkat lunak dan antarmuka pengguna. Pembuatan *aplikasi* menggunkan bahasa pemrograman berorientasi objek, yaitu menggunakan bahasa pemrograman C# dan mengintegrasikannya ke database MySQL.

3.5. Pengujian Perangkat Lunak

Pengujian perangkat lunak dilakukan untuk memastikan bahwa perangkat lunak telah sesuai dengan spesifikasi dari kebutuhan yang telah ditentukan sebelumnya. Pengujian perangkat lunak meliputi :

1. Pengujian validasi

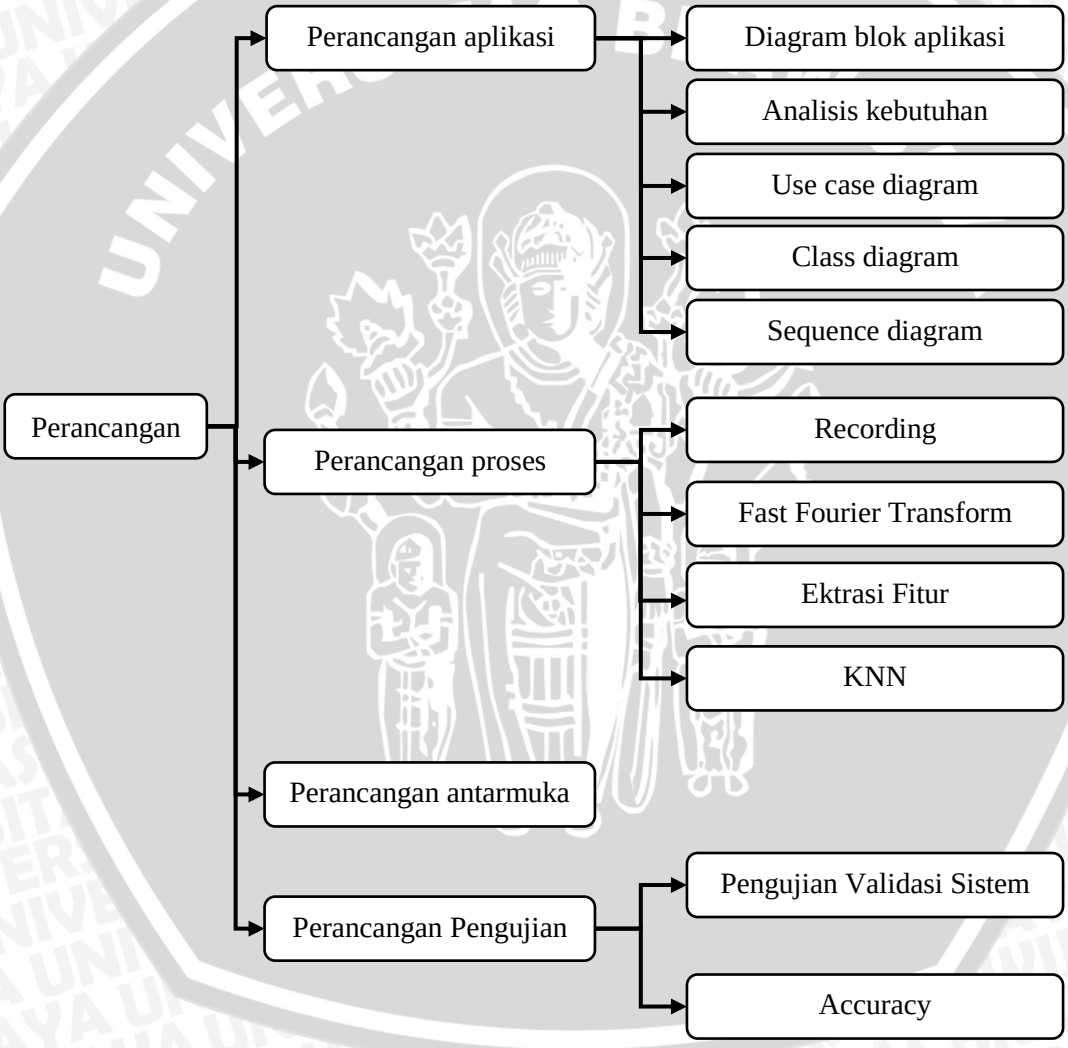
   Pengujian validasi menggunakan teknik *Black Box testing* karena tidak berkosentrasi pada alur jalannya algoritma program. Pengujian dilakukan untuk mengetahui apakah sistem yang dibangun telah sesuai dengan daftar kebutuhan. Ada 3 (tiga) validasi utama sistem yang akan dilihat yaitu : proses perekaman suara, proses perhitungan data suara dan insert data, serta proses pengenalan perintah suara.
2. Pengujian kinerja atau tingkat keakuratan sistem dilakukan dengan cara menghitung nilai accuracy tiap kata yang diteliti yang diperoleh dengan menentukan nilai tetangga terdekat yang berbeda pada algoritma KNN dan jenis pembicara yang berbeda.

3.6. Pengambilan Kesimpulan

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

BAB IV
PERANCANGAN

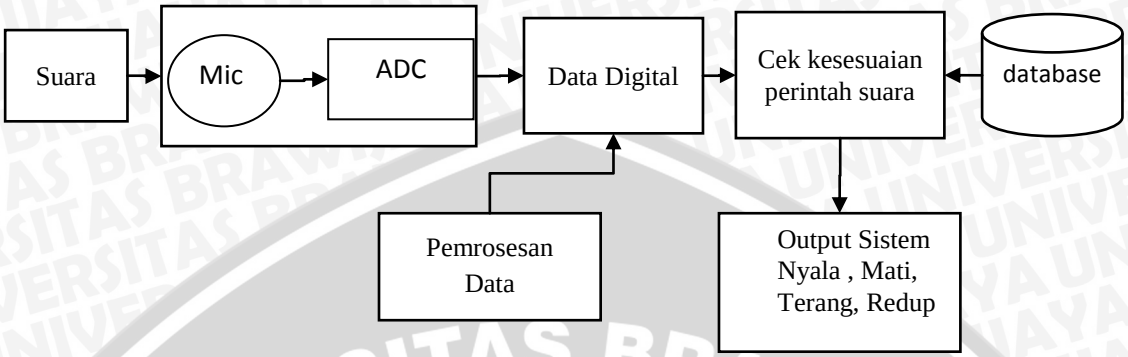
Pada bab ini dijelaskan langkah-langkah yang akan dilakukan dalam tahap perancangan. Perancangan yang dilakukan meliputi perancangan aplikasi, perancangan proses, perancangan pengujian, dan perancangan antarmuka atau tampilan aplikasi. Pohon perancangan aplikasi dapat dilihat pada gambar 4.1.



Gambar 4.1 Pohon Perancangan

Sumber : [perancangan]

4.1 Diagram Blok Aplikasi



Gambar 4.2 Diagram Blok Perancangan Aplikasi
Sumber: [Perancangan]

Penjelasan diagram blok perancangan aplikasi pada gambar 4.2 dapat dijelaskan sebagai berikut:

1. Pengguna/ *user* menginputkan sebuah perintah suara melalui eksternal microphone.
2. Sinyal suara hasil perekaman kemudian diubah dari bentuk analog ke digital menggunakan bantuan soundcard dalam *accessories windows.*
3. Data suara digital kemudian diproses menggunakan algoritma *FFT* dan ekstrasi fitur suara untuk digunakan dalam pencocokan perintah suara.
4. Hasil data digital suara yang sudah diproses akan dicocokan dengan hasil inputan perintah suara *user* yang baru menggunakan algoritma *KNN*.
5. Jika perintah suara hasil inputan *user* sesuai dengan basis data hasil pelatihan perintah suara sebelunya, maka aplikasi akan menampilkan sebuah output lampu menyala, mati, terang, atau redup.

4.2 Analisis Kebutuhan

Pengembangan sebuah perangkat lunak bertujuan untuk menghasilkan perangkat lunak yang dapat memenuhi kebutuhan *user*. Setiap pengembangan

sebuah sistem perangkat lunak memerlukan adanya dokumentasi terhadap kebutuhan-kebutuhan *user* agar tujuan tersebut tercapai.

Proses analisis kebutuhan dilakukan dengan acuan pengumpulan dan penetapan kebutuhan-kebutuhan dari aplikasi yang akan dibangun. Analisis kebutuhan dibagi menjadi dua tahap yaitu tahap identifikasi aktor dan tahap penjabaran daftar kebutuhan.

4.2.1 Identifikasi Aktor

Tahap identifikasi aktor dilakukan untuk mengidentifikasi aktor yang berinteraksi dengan aplikasi. Identifikasi aktor akan dijelaskan pada tabel 4.1

Tabel 4.1 Identifikasi Aktor

Aktor	Deskripsi Aktor
Pengguna/ *User*	Pengguna atau *User* merupakan aktor yang menggunakan fasilitas aplikasi perintah suara untuk menyalakan lampu, mematikan lampu, membuat lampu menjadi lebih terang dan lebih redup.

Sumber: [Perancangan]

4.2.2 Identifikasi Kebutuhan

Pada tahap daftar kebutuhan akan dijelaskan kebutuhan fungsional yang diperlukan oleh pengguna atau *user* dalam menggunakan aplikasi pengontrol nyala lampu berbasis perintah suara. Spesifikasi kebutuhan fungsional pengguna atau *user* ditunjukkan pada tabel 4.2

Tabel 4.2 Daftar Kebutuhan Fungsional *User*

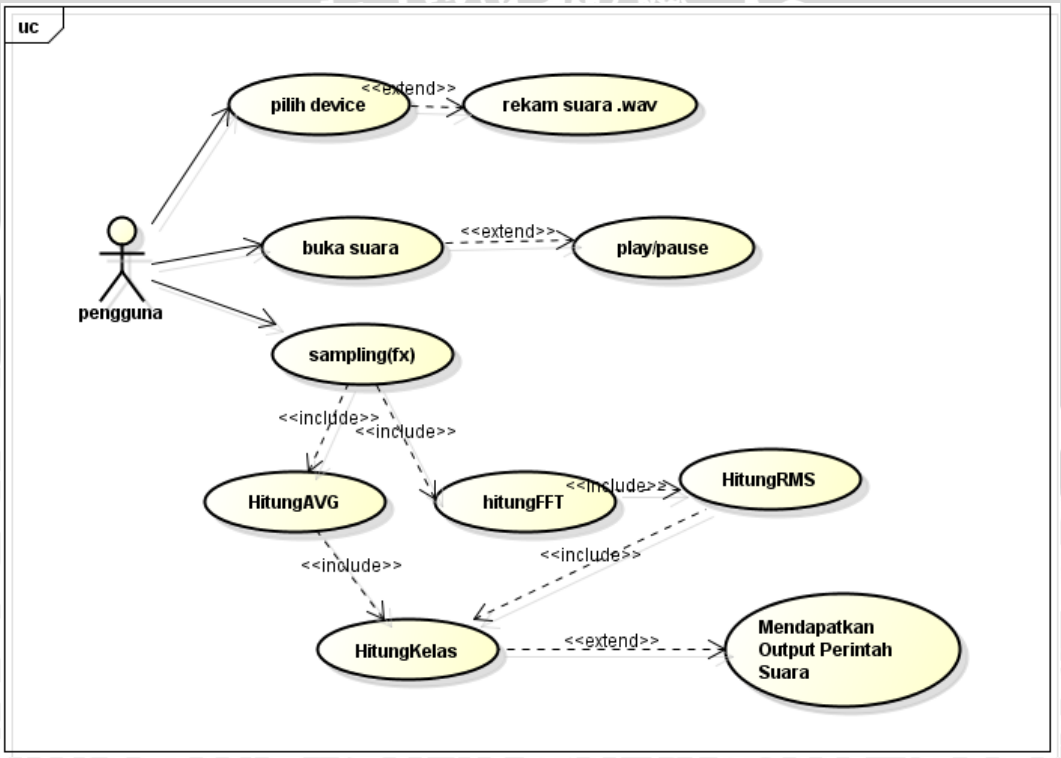
No	Kebutuhan
01	Aplikasi pengontrol nyala lampu berbasis perintah suara harus dapat memberikan fasilitas untuk menampilkan hasil inputan perintah suara berupa nyala lampu, mematikan lampu, menerangkan lampu, dan meredupkan lampu
02	Aplikasi pengontrol nyala lampu berbasis perintah suara harus dapat digunakan merekam perintah suara dalam format .wav
03	Aplikasi pengontrol nyala lampu berbasis perintah suara harus dapat digunakan untuk mensampling data digital

	suara yang sudah direkam.
04	Aplikasi pengontrol nyala lampu berbasis perintah suara harus dapat digunakan untuk melakukan perhitungan Fast Fourier Transform sebagai pengolah perintah suara.
05	Aplikasi pengontrol nyala lampu berbasis perintah suara harus dapat digunakan untuk melakukan perhitungan RMS dan AVG sebagai fitur ciri perintah suara dan melakukan pencocokan suara menggunakan algoritma KNN

Sumber: [Perancangan]

4.2.3 *Use Case Diagram*

Diagram *use case* adalah salah satu diagram untuk memodelkan aspek perilaku aplikasi. Diagram *use case* menunjukkan sekumpulan *use case*, aktor, dan hubungannya. *Use case* merupakan fungsionalitas aplikasi yang diinisialisasi oleh aktor. Pemodelan diagram *use case* untuk aplikasi pengontrol nyala lampu berbasis perintah suara dapat dilihat pada gambar 4.3



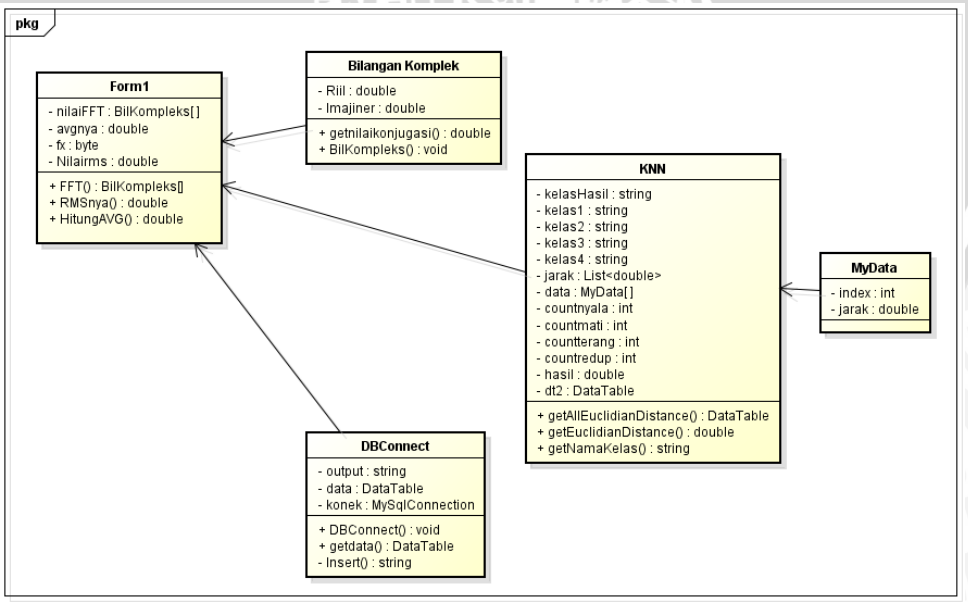
Gambar 4.3 Diagram *Use Case* Aplikasi
Sumber: [Perancangan]

Diagram use case pada gambar 4.3 di atas dapat dijelaskan sebagai berikut:

- Pengguna / *User* memilih device sound untuk melakukan perekaman, kemudian user dapat merekam suara dan menghentikan perekaman suara.
- Pengguna / *User* juga dapat melakukan pemutaran suara yang telah direkam untuk mendengarkan hasil rekaman suara.
- Hasil perekaman suara kemudian diproses melalui perhitungan sampling suara, perhitungan FFT, perhitungan fitur suara yaitu RMS dan AVG, kemudian melakukan pencocokan inputan perintah suara dengan melakukan perhitungan kelas. Jika perintah suara yang diinputkan oleh pengguna cocok dengan data base, maka aplikasi akan memunculkan output sistem berupa lampu yang menyala, mati, bertambah terang, atau redup.

4.2.4 *Class Diagram*

Class diagram adalah diagram kelas yang akan diimplementasikan pada aplikasi pengontrol nyala lampu berbasis perintah suara. Pada kelas diagram ini terdapat 5 buah kelas yaitu *main Form1 class*, kelas Bilangan Kompleks, kelas KNN, kelas DBConnect, dan kelas MyData. Pemodelan *class diagram* untuk aplikasi pengontrol nyala lampu berbasis perintah suara adalah sebagai berikut.



Gambar 4.4 Diagram *Class* Aplikasi
Sumber: [Perancangan]

Diagram *class* pada gambar 4.4 di atas dapat dijelaskan seperti di bawah ini:

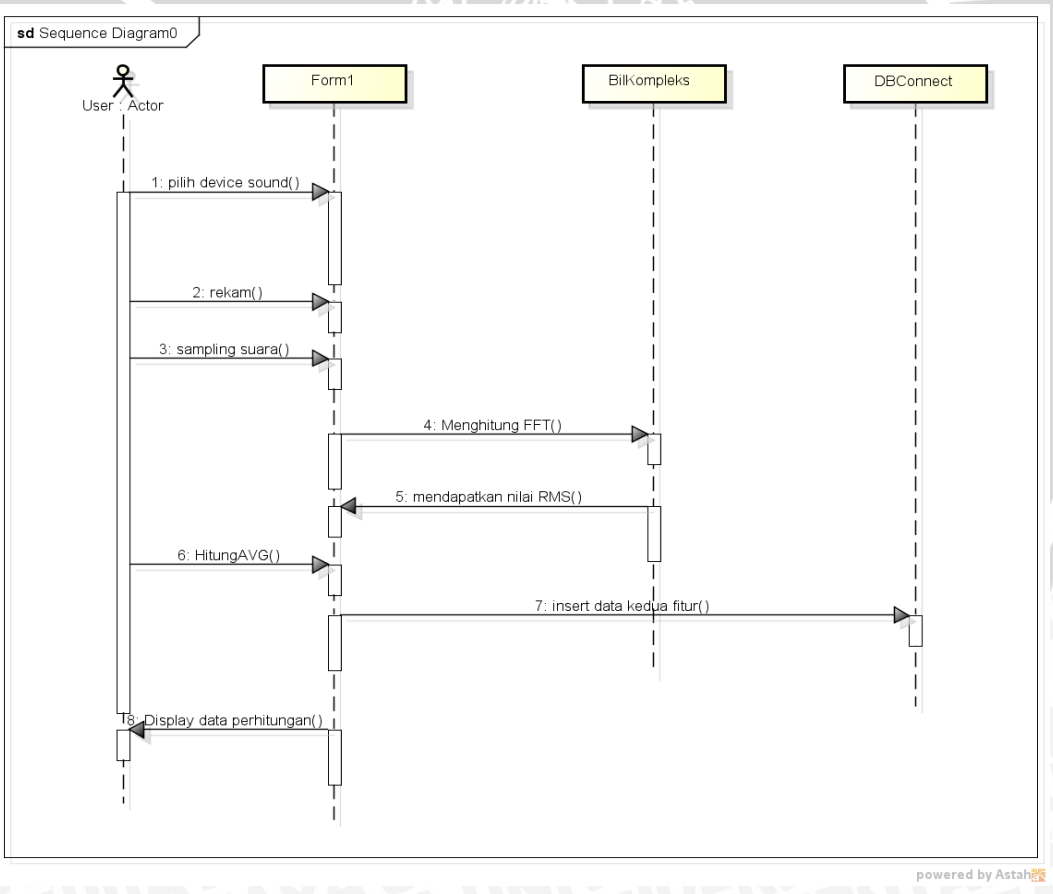
1. Kelas DBConnect merupakan kelas yang digunakan untuk menghubungkan sistem dengan *database* MySQL. Kelas ini terdiri dari 3 *method* yaitu *DBconnect(), Insert(),* dan *getdata(). Method DBConnect()* digunakan untuk membuka koneksi dengan *database* MySQL, *method Insert()* digunakan untuk memasukkan data hasil pelatihan fitur RMS dan AVG kedalam database, sedangkan *method getdata()* digunakan untuk mengambil dan menampilkan data yang ada dalam *database* MySQL.
2. Kelas *Bilamgan Kompleks* merupakan kelas yang digunakan untuk perhitungan *Fast Fourier Transform*. Kelas ini terdiri dari 2 *method* yaitu *BilKompleks()* dan *getnilaikonjugasi(). Method BilKompleks()* digunakan untuk mendeklarasikan variable riil dan imaginer yang nantinya digunakan sebagai tipe data dalam perhitungan FFT yang mengandung nilai riil dan imaginer, sedangkan *method getnilaikonjugasi()* digunakan untuk melakukan perhitungan konjugasi dari hasil perhitungan FFT.
3. Kelas *Mydata* merupakan kelas yang digunakan untuk mendeklarasikan variable index dan jarak dalam perhitungan algoritma KNN.
4. Kelas *KNN* merupakan kelas yang digunakan untuk mengerjakan proses algoritma *K-Nearest Neighbor*. Kelas ini memiliki 3 *method* yaitu *getNamaKelas(), getEcluidianDistance(),* dan *getAllEcluidianDistance(). Method getNamaKelas()* digunakan untuk menampung atau mendapatkan jarak tiap data dan mengembalikan nilai kembalian berupa kelas hasil yang memiliki jarak ecluidian terkecil. *Method getEcluidianDistance()* digunakan untuk menghitung jarak data testing terhadap data training. Method *getAllEcluidianDistance()* digunakan untuk menghitung semua jarak ecludiandistance tiap data.
5. Kelas *main Form1* merupakan kelas utama yang akan melakukan pemanggilan ke kelas-kelas yang lainnya.

4.2.5 *Sequence Diagram*

Sequence diagram adalah diagram yang menunjukkan alur atau urutan jalannya aplikasi. Terdapat 2 *sequence diagram* pada aplikasi pengontrol nyala lampu berbasis perintah suara yaitu *sequence diagram pemrosesan suara dan ektrasi fitur*, dan *sequence diagram pencocokan suara melalui KNN*. Pemodelan masing-masing *sequence diagram* dapat dijelaskan sebagai berikut.

4.2.5.1 *Sequence Diagram Pemrosesan Suara dan Ektrasi Fitur*

Sequence diagram pemrosesan suara menampilkan urutan jalannya aplikasi pengontrol nyala lampu berbasis perintah suara ketika berhubungan dengan algoritma *FFT* untuk mendapatkan suatu ektrasi fitur. Pemodelan *sequence diagram caverphone* dapat dilihat pada gambar 4.5



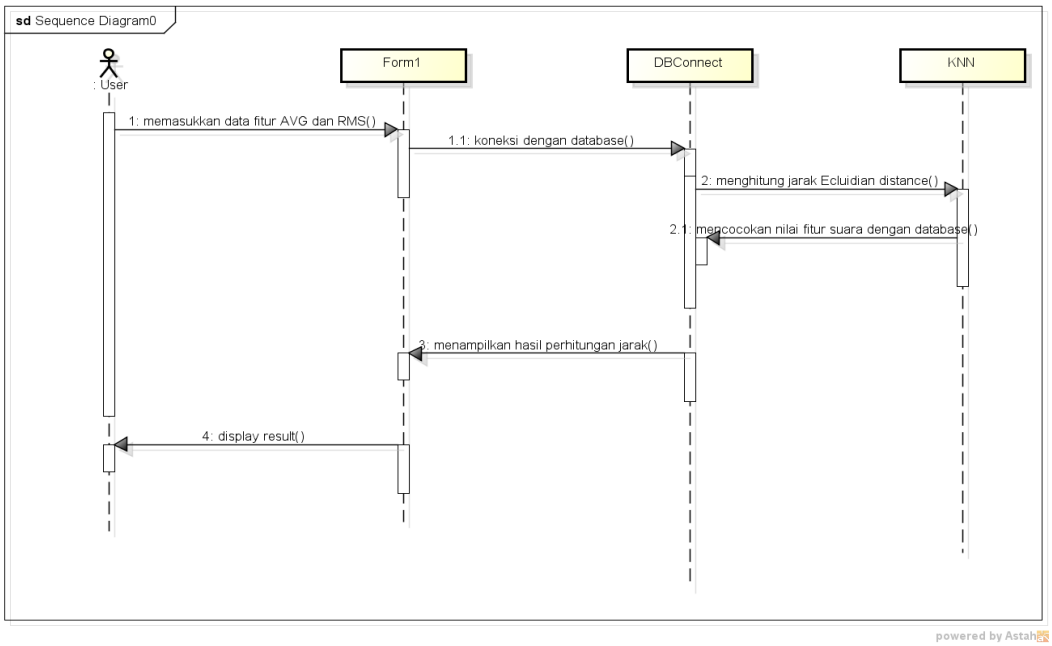
Gambar 4.5 Sequence Diagram Pengolahan Suara Dan Ektrasi Fitur
Sumber: [Perancangan]

Diagram *sequence* pada gambar 4.5 di atas dapat dijelaskan seperti di bawah ini:

1. User memilih device dan merekam suara sebagai data inputan.
2. Aplikasi akan mengolah data inputan suara dengan men-sampling suara menjadi beberapa bagian data.
3. Data hasil sampling suara akan dihitung dengan menggunakan algoritma *FFT* yang akan menghasilkan suatu bilangan kompleks dan dihitung kembali melalui konjugasi sehingga didapatkan suatu fitur RMS
4. Aplikasi akan memproses hasil sampling untuk menghitung nilai fitur yang kedua yaitu fitur AVG melalui pendekatan window hamming
5. Aplikasi akan menampilkan hasil dari masing-masing perhitungan data dan memasukkan nilai fitur AVG dan RMS ke dalam suatu database.

4.2.5.2 *Sequence Diagram Pencocokan Suara*

Sequence diagram pencocokan suara menampilkan urutan jalannya aplikasi pengontrol nyala lampu berbasis perintah suara ketika berhubungan dengan algoritma *KNN* untuk menghitung ecluidian distance. Pemodelan *sequence diagram caverphone* dapat dilihat pada gambar 4.5



Gambar 4.6 Sequence Diagram Pencocokan Suara
Sumber: [Perancangan]

Diagram *sequence* pada gambar 4.6 di atas dapat dijelaskan seperti di bawah ini:

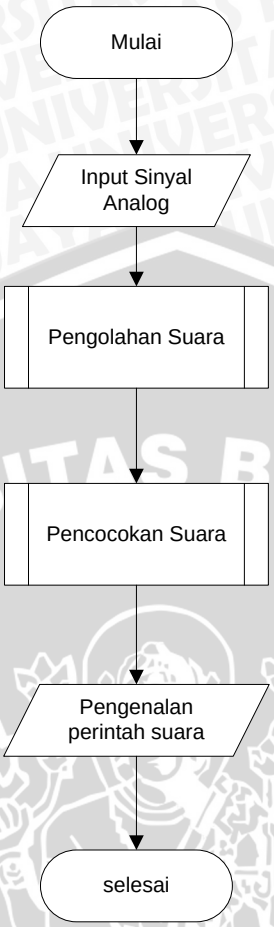
1. User memasukkan data dua fitur pada halaman user dan melakukan koneksi dengan database
2. Setelah terhubung dengan database, aplikasi akan menghitung jarak ecluidian antara data testing dan data training terhadap dua fitur suara.
3. Data dua fitur suara akan dicocokkan dengan data training dengan mengambil nilai terkecil dari jarak ecluidian masing-masing data kemudian hasil perhitungan ditampilkan.
4. Aplikasi akan menampilkan *output sistem* sesuai dengan inputan perintah suara yaitu berupa lampu yang menyala, mati, terang, atau redup.

4.3 Perancangan Proses

Perancangan proses dari aplikasi pengontrol nyala lampu berbasis perintah suara dengan menggunakan metode *FFT* dan *KNN*, dilakukan dalam beberapa tahap meliputi perekaman suara (*recording*), proses *algoritma FFT*, ektrasi fitur, dan proses *algoritma KNN*.

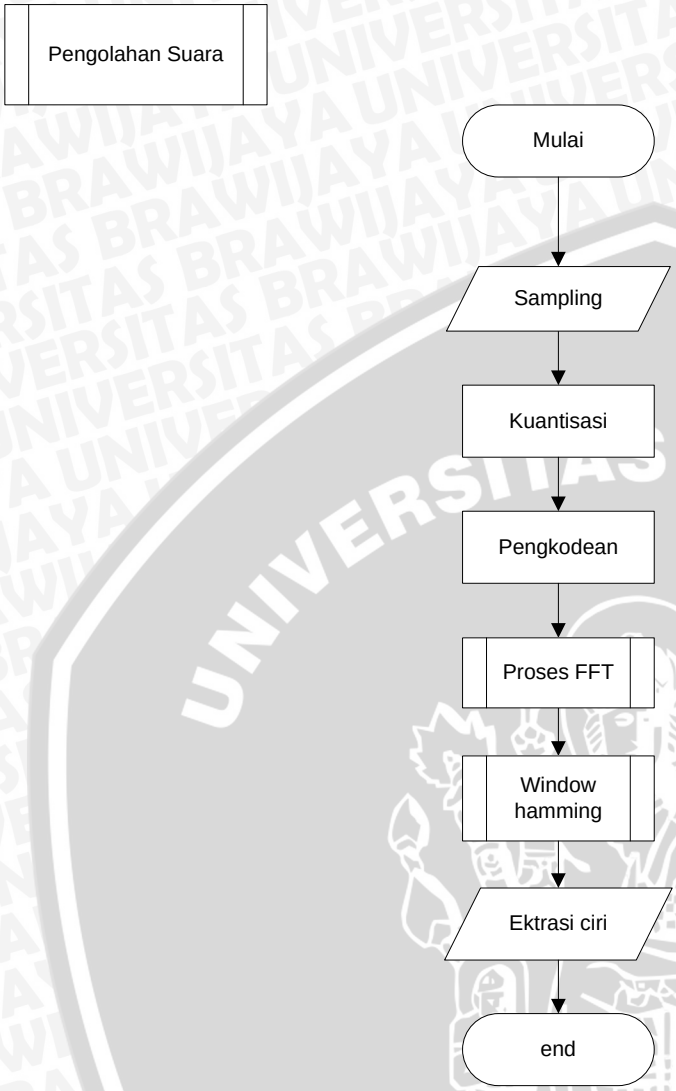
4.3.1 Perancangan Secara Umum

Desain umum yang akan diaplikasikan bertujuan untuk memberikan gambaran secara umum kepada penggunaan tentang sistem yang akan dibangun.



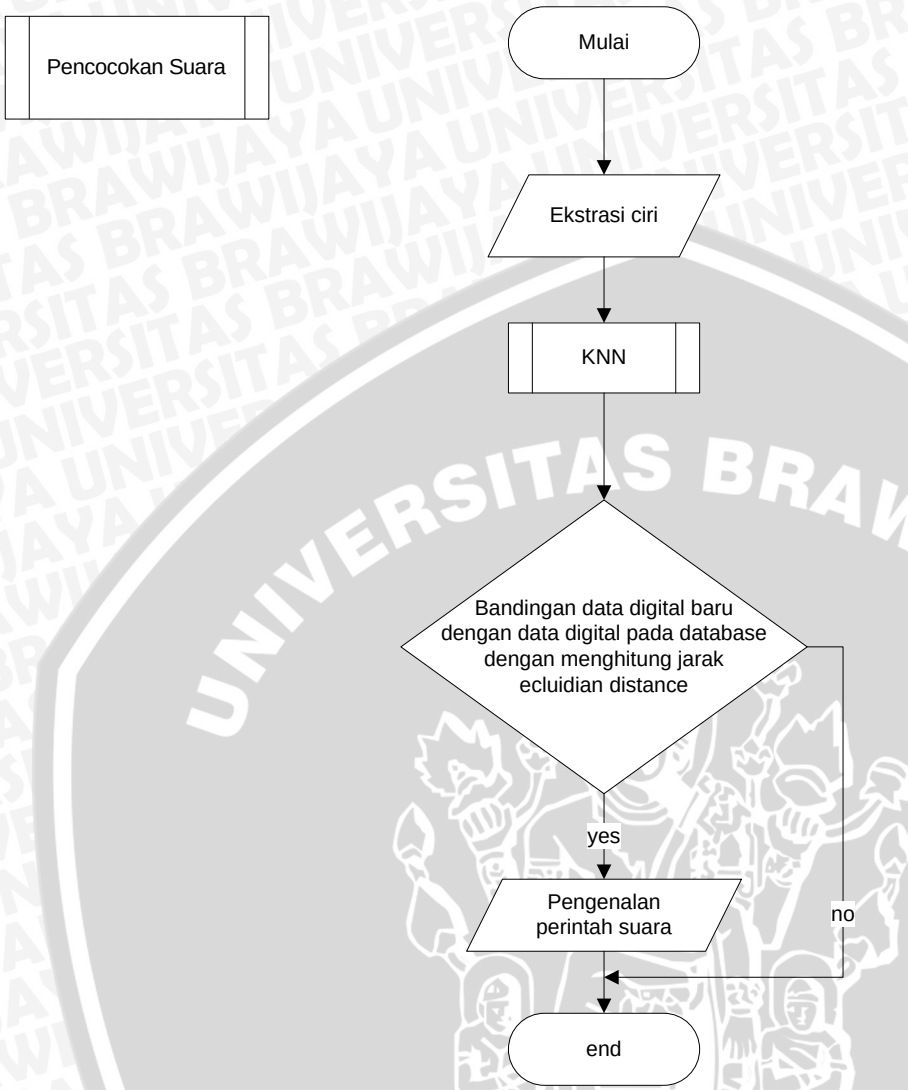
Gambar 4.7 Diagram Alur Sistem Pengenalan Perintah Suara

Sinyal yang dihasilkan oleh suara manusia merupakan suatu sinyal analog. Agar sinyal analog dapat dikenali oleh komputer, maka terlebih dahulu harus dilakukan perubahan menjadi sinyal digital. Perubahan dari sinyal analog menjadi sinyal digital dilakukan melalui proses *pengolahan suara*. Kemudian sinyal digital diproses untuk mendapatkan dua buah fitur yaitu RMS dan AVG yang nantinya digunakan untuk proses *pencocokan perintah suara*. Kemudian dari hasil pencocokan perintah suara, didapatkan suatu pengenalan perintah suara.



Gambar 4.8 Flowchart Pemrosesan Sinyal Suara

Proses pengolahan suara dari sinyal analog menjadi sinyal digital dilakukan melalui 3 tahapan yaitu *sampling*, *kuantisasi*, dan *pengkodean.* Kemudian data sinyal digital diolah untuk mendapatkan suatu sinyal periodik dalam domain frekuensi melalui metode *Fast Fourier Transform* dan *Window Hamming* yang nantinya akan didapatkan data-data fitur/ciri dari sinyal suara tersebut yaitu fitur RMS dan fitur AVG.



Gambar 4.9 Flowchart Pencocokan Perintah Suara

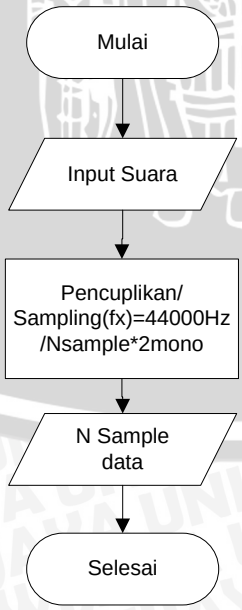
Proses pencocokan dua data fitur suara dilakukan dengan menggunakan metode KNN. Proses pencocokan dilakukan dengan cara menghitung jarak ecluidian distance dari data testing fitur RMS dan AVG terhadap data training fitur RMS dan AVG yang ada dalam database. Kemudian data dua fitur tersebut dibandingkan lalu dicocokkan untuk mendapatkan suatu pengenalan perintah suara.

4.3.2 Perancangan Perangkat Lunak

Perangkat lunak dibangun dengan menggunakan *software Visual Studio* dengan bahasa pemrograman C#. Perancangan sistem dilakukan dengan langkah-langkah seperti berikut.

4.3.2.1 Recording

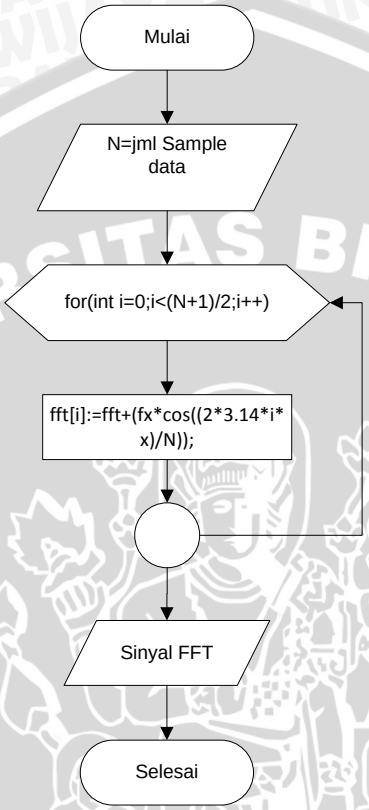
Data berupa sinyal suara diperoleh dengan cara merekam suara melalui mikrofon yang dihubungkan dengan sebuah komputer. Perekaman suara dilakukan dengan membuat fungsi *wave record* di dalam aplikasi dengan menggunakan bantuan program *accessories windows*, yaitu *sound recorder* dengan frekuensi sampling 44100Hz 16 bit stereo. Frekuensi sampling sebesar 44100Hz sering digunakan untuk merekam suara manusia karena dinilai sudah cukup, dan kualitas suaranya kira-kira setara dengan lawan bicara ketika bicara di telepon. Durasi suara yang akan direkam pada aplikasi adalah 2-3 detik. Suara direkam dengan intonasi sedang oleh seorang laki laki dan seorang perempuan masing-masing sebanyak 10 kali tiap perintah kata. Sinyal suara disampling dengan frekuensi sampling 44100 Hz sehingga menghasilkan sekitar 1411200 titik sampel data yang kemudian diambil sebanyak 50 data sampel.



Gambar 4.10 Flowchart Recording

4.3.2.2 *Fast Fourier Transform* (FFT)

Data sinyal yang dihasilkan dari proses sampling kemudian diolah menggunakan FFT untuk diambil data sinyal berdasarkan domain frekuensi. Berikut diagram alir dari proses FFT.



Gambar 4.11 Flowchart FFT

Pada proses FFT input N adalah jumlah data sample hasil pencuplikan sinyal suara. Dimulai dengan membagi sinyal menjadi bagian lebih kecil dengan (N+1)/2 bagian, karena sinyal bersifat mirror / terdiri dari bagian imajiner dan riel. Kemudian melakukan perhitungan FFT, dilakukan langkah secara terus menerus sesuai dari perulangan sampai mendapat kombinasi dari nilai FFT.

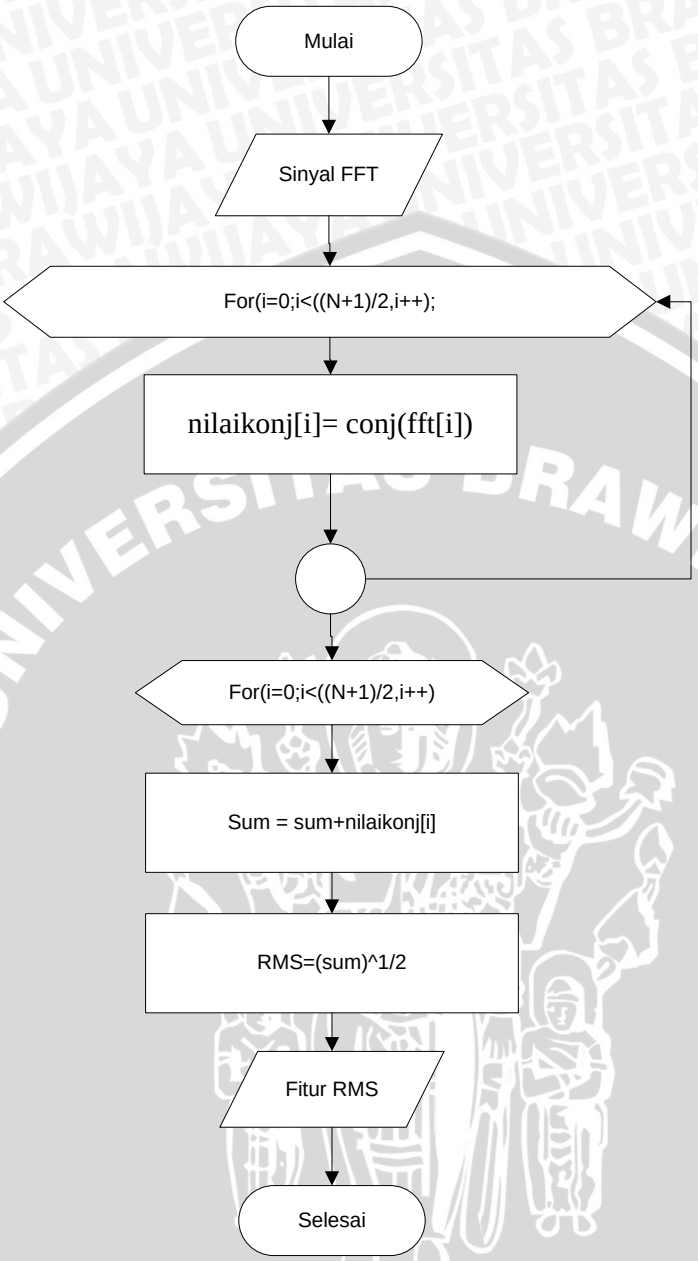
4.3.2.3 Tahap Ektrasi Fitur

File suara yang sebelumnya direkam berekstensikan *wav* dan selanjutnya diproses kedalam suatu proses *sampling* dan proses FFT sehingga dapat dilakukan proses *feature extraction*, yaitu sebuah proses yang mengkonversi sinyal suara

menjadi beberapa parameter yang dapat diambil untuk proses selanjutnya yaitu identifikasi pola suara.

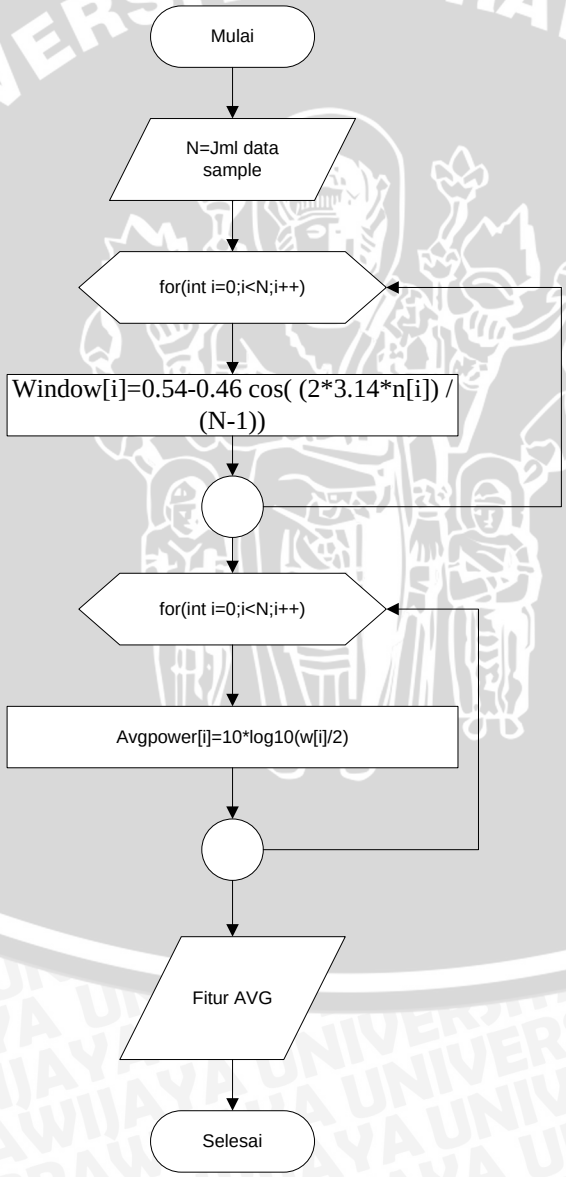
Parameter yang memungkinkan dari suatu *file* suara adalah nilai amplitudo. Karena nilai amplitudo didapatkan berdasarkan domain waktu sehingga untuk pengambilan sampel data akan didapatkan waktu yang mungkin berbeda-beda. Hal inilah yang menyebabkan suatu *feature extraction* tidak konsisten untuk dijadikan suatu acuan atau masukan pada proses selanjutnya. Untuk itulah parameter dari suatu *file* suara tersebut harus diubah terlebih dahulu kedalam domain frekuensi dengan menggunakan *Fast Fourier Transform* karena pada manusia memiliki batas frekuensi tertentu sehingga data sampel bisa lebih konsisten untuk dijadikan suatu masukan pada proses selanjutnya. Proses inilah yang merupakan bagian dari pengolahan sinyal yang akan menghasilkan suatu *feature extraction* berupa nilai *magnitude* terhadap domain frekuensi.

Untuk mendapatkan fitur *Root Mean Square* yang perlu dilakukan adalah menghitung konjugasi bilangan imajiner hasil perhitungan FFT sehingga didapatkan suatu bilangan real. Kemudian bilangan-bilangan hasil konjugasi dijumlahkan agar mendapatkan satu nilai rata rata yang dapat mewakili nilai fitur dari RMS. Proses dari fitur RMS dapat dilihat pada gambar 4.12 berikut.



Gambar 4.12 Flowchart *Root Mean Square*

Untuk mendapatkan fitur *Average Power Sprectum* yang perlu dilakukan adalah mengukur rata-rata daya menggunakan *periodogram spectrum object* dengan metode *window hamming*, dan menghitung rata-rata daya spektrum. Tahapan windowing berfungsi untuk mengurangi efek diskontinuitas pada masing-masing sinyal yang telah diperoleh dari proses sampling sebelumnya. Fungsi *window hamming* ini menghasilkan *sidelobe level* yang tidak terlalu tinggi (kurang lebih -43 dB), selain itu *noise* yang dihasilkan pun tidak terlalu besar (kurang lebih 1.36 BINS). [JER-11:17]

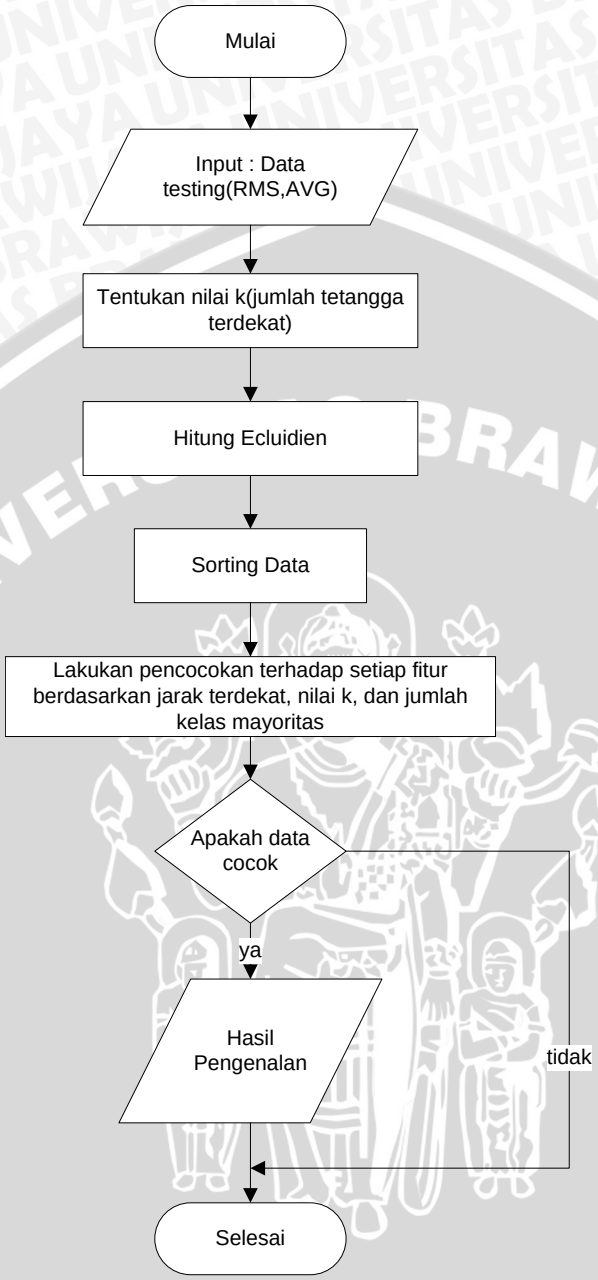


Gambar 4.13 Flowchart *Average Power Sprectum*

4.3.2.4 Pengenalan Perintah Suara

Pengenalan perintah suara menggunakan metode *K-Nearest Neighbor* pada program yang akan dirancang digunakan untuk proses klasifikasi terhadap inputan yang berupa data suara dengan cara membagi proses ekstraksi sinyal suara menjadi 2 bagian fitur suara, bagian pertama yaitu *Root Mean Square* dan bagian kedua *Average Power Sprectum.* Proses pencocokan pertama kali dilakukan dengan menentukan jumlah tetangga terdekat (nilai k) yang nantinya digunakan untuk menentukan nilai mayoritas yang paling mendekati dengan data testing fitur suara.

Kemudian pada setiap fitur suara tersebut, dilakukan proses perhitungan jarak ecluidian distance data testing suara terhadap masing-masing data training yang ada dalam database. Setelah melakukan perhitungan jarak ecluidien distance, data-data fitur tersebut disorting dari nilai jarak yang terkecil hingga yang terbesar. Kemudian dengan memperhatikan nilai jarak yang terdekat, dan jumlah kelas mayoritas yang didapatkan berdasarkan nilai tetangga yang telah ditentukan, data fitur perintah suara tersebut dicocokkan dengan data fitur perintah suara yang ada pada database yang akhirnya dikenali sebagai suatu perintah suara.



Gambar 4.14 Flowchart Pengenalan Perintah Suara Dengan KNN

4.4 Contoh Perhitungan

Pada sub bab ini akan dibahas tentang contoh perhitungan manual. Misalkan perintah suara yang digunakan adalah kata *"mati"*. Sinyal suara disampling sebanyak 4 sampel. Nilai tiap sampel untuk sinyal suara tersebut ditunjukkan pada tabel 4.3.

Tabel 4.3 Data Sampling Suara

f(0)	1
f(1)	0.21551724137931
f(2)	0
f(3)	0.396551724137931

4.4.1 Perhitungan Fast Fourier Transform

Tahap selanjutnya adalah perhitungan FFT. Pada perhitungan FFT, hasil data sampling diubah menjadi data sinyal suara dalam domain frekuensi dengan memanfaatkan rumus konjugasi. Berikut contoh perhitungan untuk indeks ke-0.

FFT(0) = ¼ [1(cos(2*pi*0*0/4) – jsin(2*pi*0*0/4)) + 0.21551724137931(cos(2*pi*0*1/4) – jsin(2*pi*0*1/4)) + 0(cos(2*pi*0*2/4) – jsin(2*pi*0*2/4)) + 0.396551724137931(cos(2*pi*0*3/4) – jsin(2*pi*0*3/4))]

= ¼ [1(1-0) + 0.21551724137931(1-0) + 0(1-0) + 0.396551724137931(1-0)]

= ¼ [1+ 0.21551724137931 + 0 + 0.396551724137931]

= 0.40301724137931025

Conjugate FFT(0) = 0.40301724137931025*conj(0.40301724137931025) = 0.162422896848989

Untuk indeks yang lainnya, perhitungannya sama dengan diatas hanya nilai tiap indeksnya diganti. Data sinyal suara FFT ditunjukkan pada tabel 4.4.

Tabel 4.4 Data Sinyal FFT

Hasil FFT
FFT(0) = 0.162422896848989
FFT(1) = 0.0645483427467301
FFT(2) = 0.00940565546967896
FFT(3) = 0.0645483427467301
FFT(4) = 0.162422896848989

4.4.2 Perhitungan Fitur RMS

Perhitungan fitur RMS dilakukan dengan menghitung rata-rata data sinyal dari hasil perhitungan FFT. Perhitungan fitur RMS adalah sebagai berikut.

1. RMS = $\sqrt{\sum FFT}$

   RMS = 0.162422896848989 + 0.0645483427467301 + 0.00940565546967896 + 0.0645483427467301 + 0.162422896848989

   RMS = 0.680696800830677

4.4.3 Perhitungan Fitur AVG

Fitur AVG didapatkan dengan menggunakan pendekatan rumus window hamming kemudian dikonversi ke dalam suatu nilai logaritmatik. Berikut merupakan contoh perhitungan window hamming untuk data sampel indeks ke-0.

W(0) = (0.54 – 0.46 cos(2*pi*0/3) * 1)) = 0.08

Untuk indeks yang lainnya, perhitungannya sama dengan diatas hanya nilai tiap indeksnya diganti. Data Window Hamming ditunjukkan pada tabel 4.5

.**Tabel 4.5** Data Window Hamming

W(0) = 0.08
W(1) = 0.1659482758620687
W(2) = 0
W(3) = 0.03172413793103448
W(total) = 0.27767241379310318

Kemudian total perhitungan window hamming dihitung menggunakan rumus AVG sebagai berikut.

AVG = 10 * log10 (Wtotal / 2)

= -8.57497260071208

4.4.4 Perhitungan KNN

Pengenalan perintah suara didapatkan melalui proses pencocokan menggunakan metode KNN. Berikut merupakan contoh perhitungan menggunakan metode KNN. Dimisalkan ada sebuah data testing perintah kata “mati” dengan data RMS = 0.803807685860088 dan data AVG = 8.42262035984051. Terdapat 12 buah data training yang ditunjukkan pada tabel 4.6.

Tabel 4.6 Data Training Fitur Suara

ID	RMS	AVG	Kelas
1	0.82865	8.43333	Nyala
2	0.81089	8.34524	Nyala
3	0.77696	8.26405	Nyala
4	0.67319	7.25656	Terang
5	0.88051	8.82825	Terang
6	0.78409	8.31066	Terang
7	0.72772	8.41041	Mati
8	0.77555	8.4926	Mati
9	0.81417	8.47138	Mati
10	0.8582	8.60782	Redup
11	0.7629	7.54585	Redup
12	0.78551	8.33961	Redup

Kemudian dilakukan proses perhitungan jarak ecluidian distance sebuah data testing terhadap masing-masing data training. Ditentukan nilai tetangga terdekat / nilai k = 5. Berikut merupakan contoh perhitungan data testing terhadap data training id ke-1.

Ecluidian Disance =

$$\sqrt{(0.82865 - 0.803807685860088)^2 + (8.43333 - 8.42262035984051)^2}$$

Ecluidian Distance = 0.0270524853603479

Untuk fitur dengan ID yang lainnya, perhitungannya sama dengan diatas hanya nilai data training tiap fiturnya diganti sesuai dengan nomer ID. Data ecluididan distance yang telah disorting dari yang terkecil hingga terbesar ditunjukkan pada tabel 4.7.

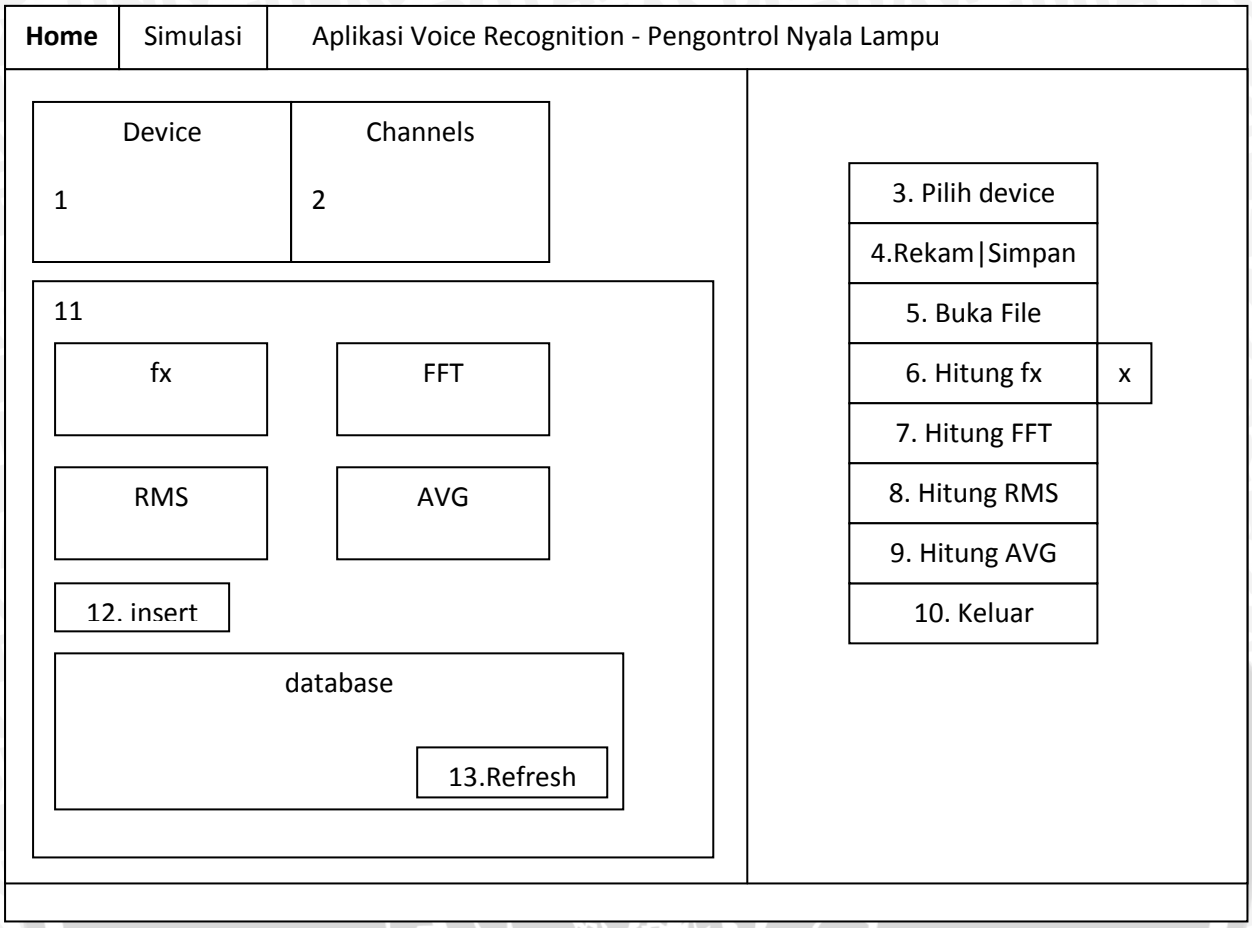
Tabel 4.7 Data Ecluidian Distance

ID	Ecluidian Distance	Kelas
1	0.0270524853603479	Nyala
9	0.0498485713217259	Mati
8	0.0754695093863681	Mati
7	0.0770612018267179	Mati
2	0.0777037918162498	Nyala
12	0.0850030890537825	Redup
6	0.113683373064373	Terang
3	0.160827103611272	Nyala
10	0.193021839574437	Redup
5	0.412817938043317	Terang
11	0.877724160916905	Redup
4	1.17335320455983	Terang

Dari tabel diatas, berdasarkan nilai k = 5 maka didapatkan 5 nilai ecluidian distance dengan jarak terdekat dengan nama kelas masing-masing jarak yaitu berupa kata "mati" sebanyak 3 buah dan kata "nyala" sebanyak 2 buah. Berdasarkan nilai mayoritas yang didapatkan, maka perintah suara yang masuk dikenali sebagai perintah suara "mati".

4.5 Perancangan Antar Muka Sistem

Perancangan antarmuka menu utama atau tampilan pertama kali pada aplikasi perintah suara pengontrol nyala lampu terdapat tujuh pilihan menu, yaitu menu pilih device,rekam/simpan suara, buka file,play/pause,hitung,keluar dan tombol refresh.



Gambar 4.15 Tampilan Awal Sistem

Sumber : [Perancangan]

Keterangan (Gambar 4.15) :

1 dan 2 : berfungsi menampilkan device soundcard suara yang disediakan oleh windows.

3 : memanggil/merefresh device soundcard yang ada pada computer.

4 : melakukan perekaman dan penyimpanan data file perintah suara dalam format .wav

5 : *view* dan *file open* data untuk mendengarkan perintah suara yang akan diproses.

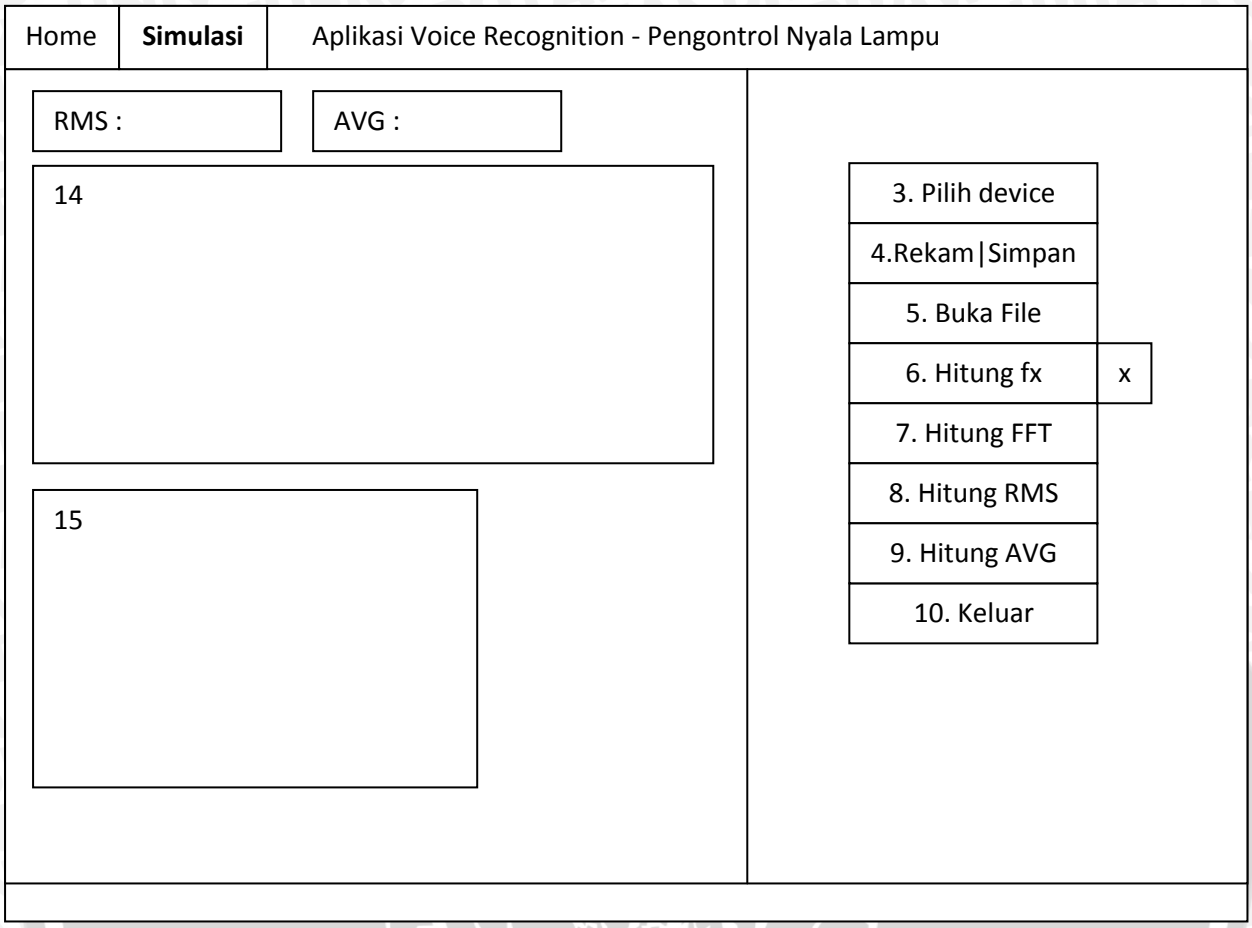
6 : berfungsi sebagai proses sampling file perintah suara.

7 : berfungsi sebagai proses perhitungan *Fast Fourier Transform (FFT).*

8 : berfungsi sebagai proses perhitungan fitur RMS

9 : berfungsi sebagai proses perhitungan fitur AVG

10 : keluar dari sistem aplikasi



Gambar 4.16 Tampilan Simulasi Sistem

Keterangan (Gambar 4.15 dan 4.16) :

11 : menampilkan hasil perhitungan sampling data, FFT sinyal suara, fitur RMS, dan fitur AVG.

12 : berfungsi memasukkan data fitur RMS dan AVG

13 : melakukan refresh isi database

14 : menampilkan jarak *ecluidian distance* masing masing data perintah suara

15 : tampilan simulasi perintah suara

4.6 Perancangan Pengujian

Proses perancangan pengujian merupakan tahapan akhir setelah perangkat lunak selesai dikembangkan. Perancangan pengujian yang nantinya akan dilakukan meliputi uji coba validasi dan keakuratan terhadap sistem yang telah

dibangun dengan melakukan proses pembelajaran dan proses pengenalan pengenalan suara. Dari 4 macam suara dari 2 orang yang berbeda, akan digunakan untuk data pelatihan dan data pengujian pengenalan suara. Ada beberapa parameter yang digunakan untuk mengetahui seberapa besar persentase keberhasilan sistem terhadap pengenalan suara. Hasil pengujian tersebut kemudian dihitung keakuratannya dengan menggunakan perhitungan *accuracy.*

4.6.1 Pengujian Validasi Sistem Perintah Suara

Pengujian sistem perintah suara ini dilakukan dengan pengujian objektif, cara menguji validasi inputan pengguna dengan menggunakan pengujian *blackbox.*

Tabel 4.8 Skenario Uji Validasi

No	Skenario Uji	Metode
1	Pengujian Perekam Suara	Blackbox
2	Pengujian perhitungan data suara dan memasukkan data dalam database	Blackbox
3	Pengujian Tampilkan Perintah Suara	Blackbox

1. Pengujian Perekam Suara

Tabel 4.9 Pengujian Perekam Suara

Kasus dan Hasil Uji (data yang dimasukan benar)			
Data Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Sudah memilih device soundcard dan melakukan perekaman suara menggunakan microphone.	Dapat melakukan perekaman suara.	-	-

Kasus dan Hasil Uji (data yang dimasukan salah)			
Data Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Belum memilih device soundcard dan melakukan perekaman suara menggunakan microphone.	Tidak dapat melakukan perekaman suara.	-	-

2. Pengujian perhitungan data suara dan memasukkan data dalam database

Tabel 4.10 Pengujian Perhitungan Data Suara dan Insert Database

Kasus dan Hasil Uji (data yang dimasukan benar)			
Data Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Memilih file data suara untuk melakukan proses perhitungan dan memasukkan data.	Data suara diproses dan masuk ke dalam database	-	-

Kasus dan Hasil Uji (data yang dimasukan salah)			
Data Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Batal memilih file data suara.	Data suara tidak diproses dan tidak masuk ke dalam database	-	-

3. Pengujian Tampilkan Perintah Suara

Tabel 4.11 Pengujian Tampilkan Perintah Suara

Kasus dan Hasil Uji (data yang dimasukan benar)			
Data Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Memasukkan data dua fitur suara.	Dapat menampilkan perintah suara	-	-

Kasus dan Hasil Uji (data yang dimasukan salah)			
Data Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Belum Memasukkan data dua fitur suara.	Tidak dapat menampilkan perintah suara	-	-

4.6.2 Pengujian Akurasi

Berikut merupakan tabel pengujian akurasi pengenalan perintah suara yang dilakukan dengan intonasi suara yang sedang atau tidak terlalu tinggi, dan terdapat dua jenis pembicara yaitu pria dan wanita serta jarak bicara kurang lebih 1 cm didepan microphone dengan durasi rekaman 2-3 detik.

Tabel 4.12 Pengujian akurasi

No	Pembicara 1 / Pembicara 2			
	k = x			
	Nyala/Terang		Mati/Redup	
	Sistem	Actual	Sistem	Actual
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
Jumlah TP				
Jumlah TN				
Jumlah FN				
Jumlah FP				
Akurasi				

Sumber : [perancangan]

Pengujian melibatkan 4 kata yang akan diteliti yaitu nyala, mati, terang, dan redup. Pengujian dilakukan oleh dua pembicara untuk mengetahui pengaruh jenis pembicara terhadap perintah suara. *Pembicara 1* berjenis kelamin laki laki dan *pembicara 2* berjenis kelamin perempuan. Kemudian pengujian juga

memperhatikan nilai k yang berbeda(x), untuk mengetahui pengaruh nilai k (tetangga terdekat) terhadap keakuratan perintah suara.

BAB V
IMPLEMENTASI

Tahap implementasi dilaksanakan setelah tahap perancangan sistem. Tahap implementasi merupakan tahap meletakkan sistem agar siap untuk dioperasikan dan dapat dipandang sebagai usaha untuk mewujudkan sistem yang telah dirancang. [KHR-11:105]

5.1 Lingkungan Implementasi

Aplikasi dibangun dengan menggunakan bahasa pemrograman C#. Sistem diimplementasikan menggunakan komputer dengan spesifikasi sebagai berikut.

1. Perangkat Keras:
   - Prosesor : Intel(R) Core(TM) Duo CPU T5750
   - RAM (Memori) : 2.00 GB
   - HDD : 320 GB
   - *Recording Device* : Eksternal Mic
2. Perangkat Lunak:
   - Sistem Operasi : Microsoft Windows 7 Ultimate
   - Bahasa Pemrograman : C#
   - IDE : Microsoft Visual Studio 2010 Express Edition

5.2 Kegiatan Implementasi

Kegiatan Implementasi ini antara lain meliputi :

1. Implementasi Antarmuka Sistem
2. Pemrograman
3. Pengujian Sistem

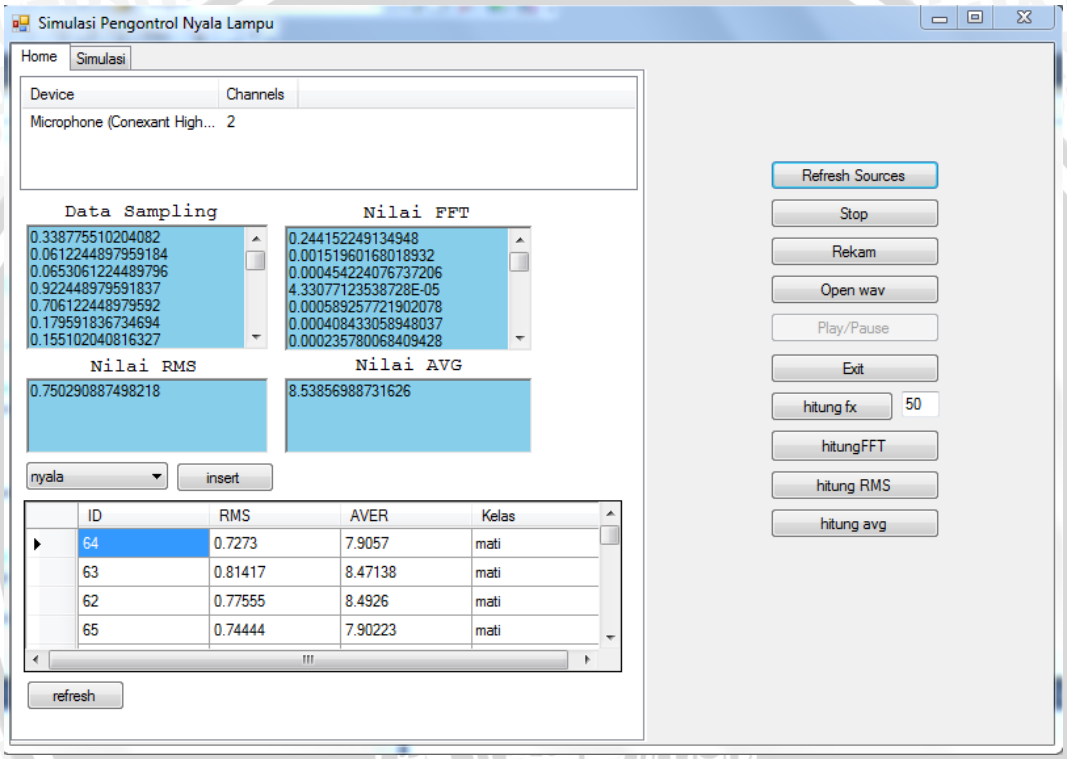
5.2.1 Implementasi Antarmuka Sistem

Dari hasil analisis perancangan antarmuka yang telah dilakukan sebelumnya, diimplementasikan dalam suatu antarmuka pada aplikasi perintah suara. Tampilan antarmuka sistem terdiri dari dua bagian yaitu bagian halaman

utama dan bagian simulasi. Berikut adalah tampilan antar muka pada aplikasi sistem perintah suara sebagai pengontrol nyala lampu :

5.2.1.1 Form Antarmuka Menu Utama Aplikasi Perintah Suara

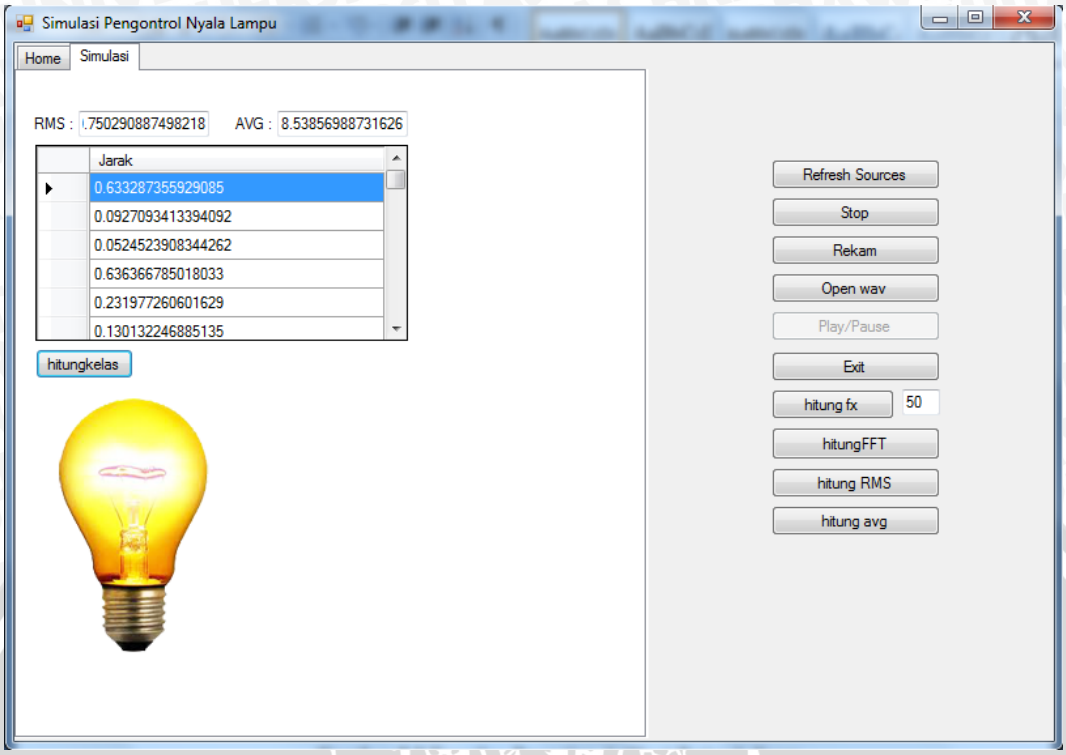
Form ini terdiri dari fungsi rekam suara dari kata yang diinginkan, dan menampilkan urutan proses pengolahan suara serta ekstraksi ciri suara. Dimulai dari perhitungan sampling suara, hitung FFT, fitur RMS dan AVG.



Gambar 5.1 Tampilan Utama Aplikasi Perintah Suara

5.2.1.2 Form Pengujian

Form ini terdiri dari fungsi perhitungan kelas perintah suara untuk menentukan data testing yang dimasukkan merupakan lampu nyala, mati, terang, atau redup.



Gambar 5.2 Tampilan Pengujian Aplikasi Perintah Suara

5.2.2 Implementasi Pemrograman

Berdasarkan perancangan proses yang terdapat dalam bab 4, maka pada bab ini akan dijelaskan implementasi proses-proses tersebut. Tahap pemrosesan awal dilakukan untuk menyetarakan semua sinyal suara dan mengubah sinyal suara dari analog menjadi digital.

5.2.2.1 Perekaman Suara (Recording)

Perekaman suara diimplementasikan dengan menggunakan komponen-komponen yang disediakan oleh *Waveform NAudio (Audio API)*. *Waveform NAudio* adalah audio API open source untuk .NET ditulis dalam C# oleh Mark Heath, dengan kontribusi dari pengembang lain. Hal ini dimaksudkan untuk memberikan seperangkat kelas utilitas berguna dari mana Anda dapat membuat aplikasi audio Anda sendiri. Komponen pada *Waveform Audio API* berupa *struct* dan *function* yang digunakan untuk implementasi perekaman suara ditunjukkan pada Tabel 5.1

Tabel 5.1 Struct dan function pada Waveform NAudio

Nama	Deskripsi
WaveIn	Fungsi input file suara
WaveFileWriter	Tujuan WaveFileWriter adalah untuk memungkinkan Anda untuk membuat file suara standar. WAV. File
WaveFileReader	Membaca file suara .wav
DirectSoundOut	Fungsi keluaran file suara
WaveFormat	Berfungsi sebagai ketentuan format file .wav yang akan direkam.
PlayBackStage	Menghentikan proses pemutaran suara ataupun memulai proses pemutaran suara.

Source code perekaman suara ini digambarkan pada Gambar 5.3

```
private void button5_Click(object sender, EventArgs e)
        {

            if (sourceList.SelectedItems.Count == 0) return;

            SaveFileDialog save = new SaveFileDialog();
            save.Filter = "Wave File (*.wav|*.wav;";
            if (save.ShowDialog() != System.Windows.Forms.DialogResult.OK)
return;

            int deviceNumber = sourceList.SelectedItems[0].Index;

            sourceStream = new NAudio.Wave.WaveIn();
            sourceStream.DeviceNumber = deviceNumber;
            sourceStream.WaveFormat = new NAudio.Wave.WaveFormat(44100,
NAudio.Wave.WaveIn.GetCapabilities(deviceNumber).Channels);

            sourceStream.DataAvailable += new
EventHandler<NAudio.Wave.WaveInEventArgs>(sourceStream_DataAvailable);
            waveWriter = new NAudio.Wave.WaveFileWriter(save.FileName,
sourceStream.WaveFormat);

            sourceStream.StartRecording();
        }
```

Gambar 5.3 Source code Recording

5.2.2.2 Proses Sampling

Sampling merupakan proses awal dalam pengenalan suara setelah merubah sinyal *analog* menjadi sinyal *digital*. Proses sampling bertujuan untuk membaca *file* recording kemudian dihitung dalam bentuk byte suara yang mewakili amplitudo suara tersebut. Dalam proses ini digunakan *openFilter* pada C# sebagai mengambil *file* format (WAV), dan WaveFileReader sebagai pengambilan data. *Source code sampling* ini digambarkan pada Gambar 5.4

```
private void button7_Click(object sender, EventArgs e)
        {

            OpenFileDialog open = new OpenFileDialog();
            open.Filter = "Wave File (*.wav|*.wav;";
            if (open.ShowDialog() != DialogResult.OK) return;

            wave = new NAudio.Wave.WaveFileReader(open.FileName);

            byte[] audData = new byte[wave.Length];
            wave.Read(audData, 0, (int)wave.Length);

            fx = new byte[0];
            int panjang = Convert.ToInt32(textBox1.Text);
            for (int j = 0; j < wave.Length; j += ((int)(wave.Length / panjang)))
            {
                Array.Resize(ref fx, fx.Length + 1);
                fx[fx.Length - 1] = audData[j];

            }


            for (int u = 0; u < fx.Length; u++)
            {

                richTextBox4.AppendText(fx[u].ToString());
                richTextBox4.AppendText("\n");
            }


        }
```

Gambar 5.4 Source code Sampling

5.2.2.3 Normalisasi

Proses Normalisasi digunakan untuk membuat kuat rendah sinyal dan juah dekatnya sinyal dari sumber sinyal tidak mempengaruhi ketika dilakukan proses sampling. Dalam proses normalisasi ini digunakan jangkuan sebesar 0 sampai 1.

```
        public double[] normalisasi(double[] source, double minVal, double
maxVal)
        {
            double max = source[0];
            double min = source[0];

            for (int i = 0; i < source.Length; i++)
            {
                if (source[i] < min)
                {
                    min = source[i];
                }

                if (source[i] > max)
                {
                    max = source[i];
                }
            }

            for (int i = 0; i < source.Length; i++)
            {
                source[i] = (source[i] - min) * (maxVal - minVal) / (max - min);
            }

            return source;
        }
```

Gambar 5.5 Source code Normalisasi

5.2.2.4 Proses Fast Fourier Transform

Pada implementasi algoritma FFT digunakan suatu tipe data baru yaitu *BilKompleks* karena setelah proses sampling, FFT melakukan suatu perhitungan bilangan kompleks. Dalam bilangan kompleks, nilai pertama merepresentasikan komponen *real* dan nilai kedua merepresentasikan komponen *imaginer*.

```
BilKompleks[] FFT()
        {
            BilKompleks[] fft = new BilKompleks[0];

            for (int u = 0; u <= fx.Length; u++)
            {
                //MessageBox.Show("fx ke-" + u + ": " + fx[u]);
                Array.Resize(ref fft, fft.Length + 1);
                fft[fft.Length - 1] = new BilKompleks();

                for (int x = 0; x < fx.Length; x++)
                {

                    fft[u].Riil += fx[x] * (Math.Cos((2 * Math.PI * u * x) /
fx.Length));
                    fft[u].Imaginer += (-1) * fx[x] * (Math.Sin((2 * Math.PI *
u * x) / fx.Length));
                }
                fft[u].Riil = fft[u].Riil / fx.Length;
                fft[u].Imaginer = fft[u].Imaginer / fx.Length;
            }
            return fft;
        }
```

Gambar 5.6 Source code FFT

Setelah melakukan perhitungan FFT, dan untuk mendapatkan satu nilai yang real, dilakukan suatu proses konjugasi agar bilangan kompleks hasil perhitungan FFT dapat digunakan dalam fitur RMS.

```
public double getNilaiKonjugasi()
        {
            double val = 0;

            val = Math.Pow(Riil, 2) + Math.Pow(Imaginer, 2);//untuk konjugasi

            return val;
        }
```

```
private void button8_Click(object sender, EventArgs e)
        {
            nilaiFFT = FFT();
            for (int i = 0; i < nilaiFFT.Length; i++ )
            {
                //MessageBox.Show("nilai fft ke-" + i + ": " +
nilaiFFT[i].getNilaiKonjugasi());

richTextBox1.AppendText(nilaiFFT[i].getNilaiKonjugasi().ToString());
                richTextBox1.AppendText("\n");
            }
```

Gambar 5.7 Source code Konjugasi

5.2.2.5 Proses Ektrasi Fitur RMS

Seteleh dilakukan proses FFT selanjutnya dilakukan proses ekstrasi ciri Root Mean Square dengan menghitung rata2 akar dari nilai perhitungan FFT.

```
double RMS(BilKompleks[] fftinput)
        {
            double Nilairms = 0;
            for (int i = 0; i < fftinput.Length; i++)
            {
                Nilairms = Nilairms + fftinput[i].getNilaiKonjugasi();
            }
            Nilairms = Math.Sqrt(Nilairms);


            return Nilairms;
        }
```

Gambar 5.8 Source code Ektrasi Fitur RMS

5.2.2.6 Proses Ektrasi Fitur AVG

Untuk mendapatkan nilai fitur AVG dilakukan suatu pendekatan dengan perhitungan Hamming Window untuk mengurangi efek diskontinu dari proses sampling. Kemudian Setelah proses windowing menggunakan window hamming, konversi nilai tersebut kedalam suatu nilai logaritmik.

```
double HitungAVG()

        {
            double avg = 0;
            for (int u = 0; u < fx.Length; u++)
            {

                avg = avg + (0.54 - (0.46 * Math.Cos((2 * Math.PI * u) /
(fx.Length - 1)))) * fx[u];
            }

                avg = 10 * Math.Log10(avg / 2);

            return avg;

        }
```

Gambar 5.9 Source code Ektrasi Fitur AVG

5.2.2.7 Proses Koneksi Database

Hasil perhitungan dua fitur yaitu RMS dan AVG disimpan ke dalam suatu database yang diikuti dengan pemberian nama kelas perintah suara.

```
class DBConnect
    {
        public MySqlConnection konek;

        public DBConnect()
        {
            string conStr =
"server=localhost;user=root;database=pengenalansuara;port=3306;password=''
";
            konek = new MySqlConnection(conStr);
            try
            {
                Console.WriteLine("Connecting to MySQL...");
                konek.Open();

            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.ToString());
            }

        }

public string Insert(double rms, double avg, string kls)
        {
            string output = "";
            konek.Close();
            konek.Open();
            DataTable data = new DataTable();
            DataSet ds = new DataSet("ggg");
            MySqlCommand a = konek.CreateCommand();
            string _sql = "call insdata(" + rms + "," + avg + ",'" + kls +
"');";
            //MessageBox.Show("" + _sql);
            a.CommandText = _sql;
            MySqlDataAdapter adap = new MySqlDataAdapter(a);
            adap.Fill(ds, "ggg");
            data = ds.Tables["ggg"];
            konek.Close();
            output = data.Rows[0][0].ToString();

            return output;
        }
```

```
        public DataTable getdata(String namaTabel)
        {
            konek.Close();
            konek.Open();
            DataTable data = new DataTable();
            DataSet ds = new DataSet("ggg");
            MySqlCommand a = konek.CreateCommand();
            a.CommandText = "select * from " + namaTabel;
            MySqlDataAdapter adap = new MySqlDataAdapter(a);
            adap.Fill(ds, "ggg");
            data = ds.Tables["ggg"];
            konek.Close();
            return data;
        }

}
```

Gambar 5.10 Source code Koneksi Database

5.2.2.8 Proses Pencocokan Suara Dengan Metode KNN

Setelah dua fitur yaitu fitur RMS dan AVG didapatkan, kemudian dilakukan pencocokan suara melalui dua fitur tersebut. Data testing perintah suara dicocokkan dengan data training yang sudah dilakukan sebelumnya dengan cara menghitung jarak ecluidian distance tiap data dan dengan menentukan nilai tetangga terdekatnya, dipilih suatu kelas yang memiliki bobot terbanyak (mayoritas).

```
class KNN
    {
        public String getNamaKelas(DataTable dataTraining, double rms,
double avg)
        {
            string kelasHasil = "", kelas1 = "", kelas2 = "", kelas3 = "",
kelas4 = "";

            List<double> jarak = new List<double>();
            MyData[] data = new MyData[0];
            int countnyala = 0;
            int countmati = 0;
            int countterang = 0;
            int countredup = 0;

            for (int i = 0; i < dataTraining.Rows.Count; i++)
            {
                double temp = getEuclidianDistance(rms, avg,
Convert.ToDouble(dataTraining.Rows[i][1]),
Convert.ToDouble(dataTraining.Rows[i][2]));
                Array.Resize(ref data, data.Length + 1);
                data[data.Length - 1].index = i;
                data[data.Length - 1].jarak = temp;
            }
```

```
        for (int i = 0; i < data.Length - 1; i++)//sorting data
            {
                for (int j = i + 1; j < data.Length; j++)
                {
                    if (data[j].jarak < data[i].jarak)
                    {
                        MyData temp = data[i];
                        data[i] = data[j];
                        data[j] = temp;
                    }
                }
            }

            for (int i = 0; i < 5; i++)
            {

                if
(dataTraining.Rows[data[i].index][3].ToString().Equals("nyala"))
                {
                    kelas1 = dataTraining.Rows[data[i].index][3].ToString();
                    countnyala++;
                }
                else if
(dataTraining.Rows[data[i].index][3].ToString().Equals("mati"))
                {
                    kelas2 = dataTraining.Rows[data[i].index][3].ToString();
                    countmati++;
                }
                else if
(dataTraining.Rows[data[i].index][3].ToString().Equals("terang"))
                {
                    kelas3 = dataTraining.Rows[data[i].index][3].ToString();
                    countterang++;
                }
                else
                {
                    kelas4 = dataTraining.Rows[data[i].index][3].ToString();
                    countredup++;
                }

            }

            if (countnyala > countmati && countnyala > countterang &&
countnyala > countredup)
                kelasHasil = kelas1;
            else if (countmati > countnyala && countmati > countterang &&
countmati > countredup)
                kelasHasil = kelas2;
            else if (countterang > countnyala && countterang > countmati &&
countterang > countredup)
                kelasHasil = kelas3;
            else if (countredup > countnyala && countredup > countmati &&
countredup > countterang)
                kelasHasil = kelas4;
            else
                MessageBox.Show("gak ada perintah");

            return kelasHasil;
        }
```

```
public double getEuclidianDistance(double rms1, double avg1, double rms2,
double avg2)
        {
            double hasil = 0;

            hasil = Math.Sqrt(((Math.Pow((rms2 - rms1), 2)) +
(Math.Pow((avg2 - avg1), 2))));

            return hasil;
        }

public DataTable getAllEuclidianDistance(DataTable dt, double rms, double
avg)
        {
            double[] hasil = new double[0];
            DataTable dt2 = new DataTable();
            dt2.Columns.Add("Jarak");

            for (int i = 0; i < dt.Rows.Count; i++)
            {
                Array.Resize(ref hasil, hasil.Length + 1);
                hasil[hasil.Length - 1] = getEuclidianDistance(rms, avg,
Convert.ToDouble(dt.Rows[i][1]), Convert.ToDouble(dt.Rows[i][2]));
                dt2.Rows.Add(getEuclidianDistance(rms, avg,
Convert.ToDouble(dt.Rows[i][1]), Convert.ToDouble(dt.Rows[i][2])));
            }

            return dt2;
        }

    }
```

Gambar 5.11 Source code metode KNN

BAB VI
PENGUJIAN DAN ANALISIS

Untuk mengetahui apakah sistem bekerja dengan baik dan sesuai dengan perancangan, maka diperlukan serangkaian pengujian. Pengujian yang dilakukan antara lain : pengujian sistem perintah suara, pengujian akurasi, dan analisis sistem.

6.1 Pengujian Sistem Perintah Suara

Pengujian sistem perintah suara ini dilakukan dengan pengujian objektif, cara menguji validasi inputan pengguna dengan menggunakan pengujian *black box*. Tabel hasil pengujian sistem dari berbagai kemungkinan proses input yang dilakukan oleh pengguna selangkapnya dapat dilihat sebagai berikut :

Tabel 6.1 Skenario Uji Validasi

No	Skenario Uji	Metode
1	Pengujian Perekam Suara	Blackbox
2	Pengujian perhitungan data suara dan memasukkan data dalam database	Blackbox
3	Pengujian Tampilkan Perintah Suara	Blackbox

6.1.1 Pengujian Perekam Suara

Perekaman suara sebagai data latih maupun data uji dilakukan sebanyak 10 kali dalam durasi 3 detik dengan memilih device soundcard yang tersedia dalam *accessories windows.*

Tabel 6.2 Pengujian Perekam Suara

Kasus dan Hasil Uji (data yang dimasukan benar)			
Data Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Sudah memilih device soundcard dan melakukan perekaman suara menggunakan microphone.	Dapat melakukan perekaman suara.	Pengguna dapat melakukan perekaman suara sebanyak 10 kali sesuai dengan yang diharapkan	Valid

Kasus dan Hasil Uji (data yang dimasukan salah)			
Data Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Belum memilih device soundcard dan melakukan perekaman suara menggunakan microphone.	Tidak dapat melakukan perekaman suara.	Pengguna harus memilih device soundcard terlebih dahulu sebelum melakukan perekaman.	Valid

6.1.2 Pengujian Perhitungan Data Suara dan Insert Database

Proses perhitungan data suara dimulai dari menghitung data sampling, fast fourier transform, kemudian menghitung data fitur suara yaitu RMS dan AVG yang dimasukkan ke dalam database.

Tabel 6.3 Pengujian perhitungan data suara dan insert database

Kasus dan Hasil Uji (data yang dimasukan benar)			
Data Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Memilih file data suara untuk melakukan proses perhitungan dan memasukkan data	Data suara diproses dan masuk ke dalam database	Pengguna dapat melakukan perhitungan data suara secara urut dan masuk ke dalam database	Valid

Kasus dan Hasil Uji (data yang dimasukan salah)			
Data Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Batal memilih file data suara.	Data suara tidak diproses dan tidak masuk ke dalam database	Pengguna harus memilih file suara terlebih dahulu untuk melakukan proses perhitungan data suara	Valid

Sumber : [Pengujian]

6.1.3 Pengujian Tampilkan Perintah Suara

Untuk menampilkan perintah suara sebagai simulasi sistem, pengguna harus memasukkan data fitur suara yaitu Root Mean Square (RMS) dan Average Power Sprectum (AVG).

Tabel 6.4 Pengujian Tampilkan Perintah Suara

Kasus dan Hasil Uji (data yang dimasukan benar)			
Data Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Memasukkan data dua fitur suara.	Dapat menampilkan perintah suara	Pengguna dapat melihat hasil tampilan perintah suara sesuai dengan yang diharapkan.	Valid

Kasus dan Hasil Uji (data yang dimasukan salah)			
Data Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Belum Memasukkan data dua fitur suara.	Tidak dapat menampilkan perintah suara	Pengguna tidak dapat melihat hasil tampilan perintah suara dan harus memasukkan data fitur terlebih dahulu	Valid

Sumber : [Pengujian]

Proses analisis terhadap hasil pengujian validasi dilakukan dengan melihat kecocokan antara fitur data masukan sistem dengan hasil pengamatan pengguna / user. Berdasarkan hasil pengujian validasi, dapat disimpulkan bahwa implementasi dan fungsionalitas sistem pengenalan perintah suara telah memenuhi kebutuhan yang telah dijabarkan pada tahap pengujian validasi sistem. Pada fitur perkaman suara, ketika pengguna telah memilih device soundcard maka pengguna dapat melakukan perekaman perintah suara sesuai dengan yang diharapkan. Dan ketika pengguna belum memilih device soundcard, maka pengguna tidak dapat melakukan perekaman perintah suara. Kemudian pada fitur perhitungan data suara dan insert database, ketika user telah memilih file suara yang akan diproses, maka pengguna dapat melihat hasil perhitungan data perintah suara secara urut mulai dari sampling suara, perhitungan fast fourier transform, fitur RMS, dan fitur AVG untuk dimasukkan ke dalam database. Dan ketika pengguna belum memilih file suara maka pengguna tidak dapat melakukan proses perhitungan dan tidak ada data yang dimasukkan ke dalam database.

Pada fitur tampilkan perintah suara, sistem dapat berjalan dengan benar ketika data fitur-fitur suara sudah dimasukkan, maka hasil perintah suara dapat ditampilkan oleh sistem, dan ketika data fitur-fitur suara belum dimasukkan oleh pengguna maka sistem tidak dapat menampilkan hasil perintah suara.

6.2 Pengujian Akurasi

Pengujian akurasi pengenalan perintah suara yang dilakukan dengan intonasi suara yang sedang atau tidak terlalu tinggi, dan terdapat dua jenis pembicara yaitu pria dan wanita serta jarak bicara kurang lebih 1 cm didepan microphone. Nilai akurasi dihitung menggunakan rumus *accuracy* [HLA-12].

Accuracy = *(TP + TN) / (TP + TN + FP + FN)*

= *(TP + TN) / total*

Keterangan :

TP = jika *prediksi* dan *actual* bernilai benar atau positif

TN = jika *prediksi* bernilai salah dan *actual* bernilai benar

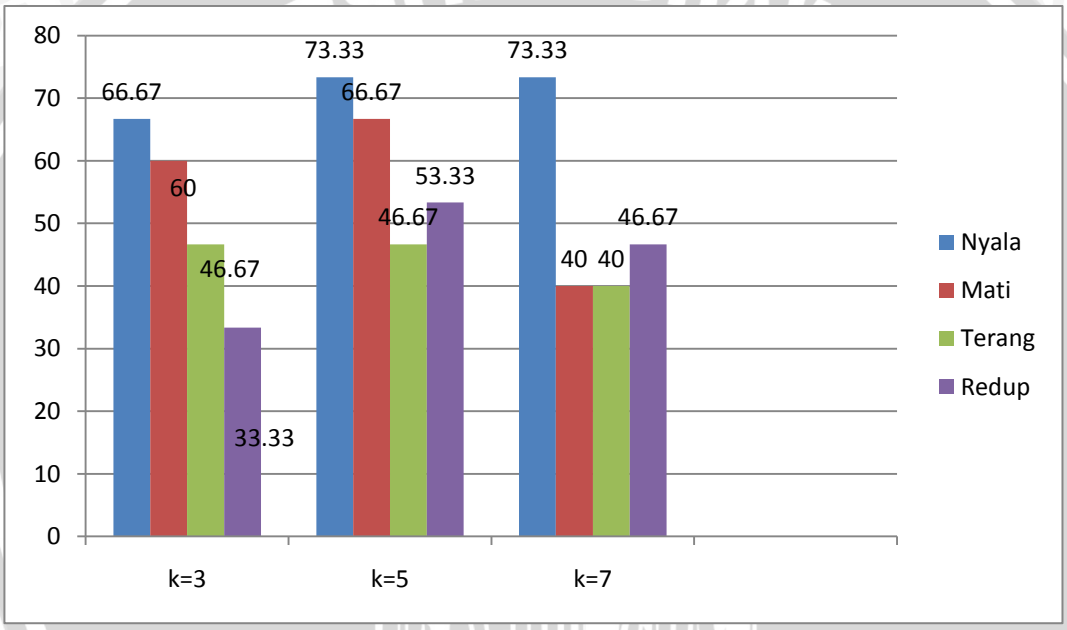
FP = jika *prediksi* bernilai benar dan *actual* bernilai salah

FN = jika *prediksi* dan *actual* bernilai salah atau negative

prediksi = perintah kata yang berhasil dikenali oleh sistem

actual = hasil pengenalan / keputusan decision maker

Berdasarkan pengujian akurasi yang telah dilakukan maka diperoleh tingkat kinerja sistem pada studi kasus nilai tetangga terdekat dengan rata- rata akurasi sebesar 51,67% untuk nilai k = 3, akurasi 60,5% untuk nilai k = 5, dan akurasi 50% untuk nilai k = 7. (Hasil percobaan dapat dilihat pada halaman lampiran). Pengujian akurasi dapat digambarkan dalam grafik 6.1 berikut ini.



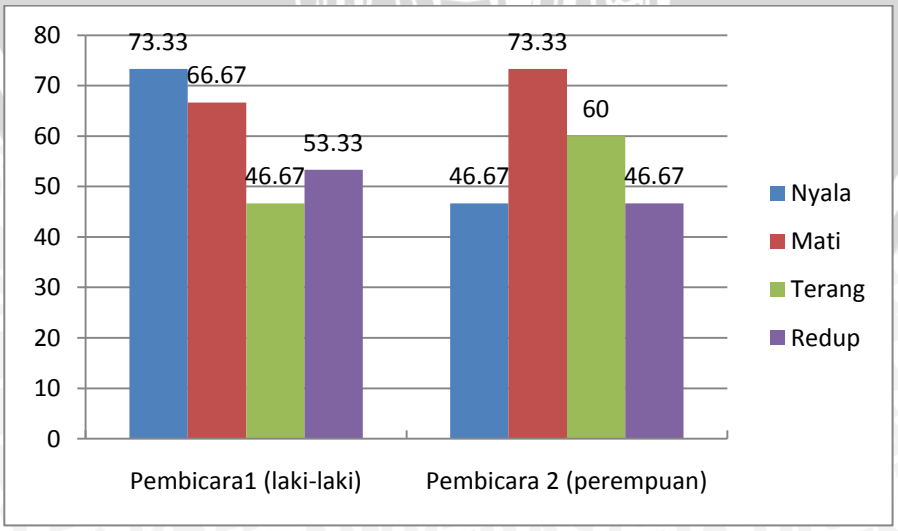
Gambar 6.1 Diagram Pengaruh Nilai k Terhadap Akurasi Sistem

Berdasarkan hasil pengujian akurasi dengan beberapa studi kasus di atas, didapatkan suatu analisa yaitu aplikasi perintah suara sebagai pengontrol nyala lampu dapat berjalan dengan baik pada nilai k = 5 dengan rata-rata akurasi 60,5% dan berjalan kurang baik pada nilai k = 3 dan k = 7. Hal ini bisa terjadi karena dalam algoritma KNN, penentuan kelas didapatkan dari proses klasifikasi berdasarkan jumlah terbanyak dari data yang memiliki kemiripan dengan data aslinya.

Pada kasus ini, dengan jumlah data yang diteliti sebanyak 4 kata, maka apabila ditentukan nilai k=3 dapat menimbulkan proses klasifikasi berjalan kurang efektif karena masih dimungkinkan ada satu atau dua data yang tidak masuk dalam urutan klasifikasi 3 data terdekat. Begitu juga apabila ditentukan nilai k=7, masih dimungkinkan sistem pengklasifikasian memiliki banyak pilihan sehingga sistem berjalan kurang efektif dan tidak mudah bagi sistem untuk menentukan jenis kelas perintah suara. Dan apabila ditentukan nilai k = 5, kemungkinan data yang yang tidak masuk dalam urutan klasifikasi akan kecil dan karena ada 4 data yang digunakan maka sistem mudah untuk menentukan tipe kelas.

Pada proses pengujian ditemukan beberapa kesalahan dan kelemahan yaitu sistem tidak cukup baik untuk mengenali perintah suara dengan kata *terang* dan *redup*. Akurasi keberhasilan tertinggi pada tiap kata perintah yang diteliti terjadi pada kata *nyala* sebesar 73,33% dan pada kata mati sebesar 67,67% sehingga dapat disimpulkan bahwa perintah suara dengan kata yang memiliki akhiran nada vocal lebih mudah dikenali oleh sistem dari pada perintah suara dengan akhiran nada mati.

Berdasarkan pengujian akurasi yang telah dilakukan maka diperoleh tingkat kinerja sistem pada studi kasus jenis pembicara dengan akurasi sebesar 60,5% untuk pembicara1 seorang laki-laki dan 56,67% untuk pembicara2 seorang perempuan.



Gambar 6.2 Diagram Pengaruh Jenis Pembicara Terhadap Akurasi Sistem

Berdasarkan hasil pengujian akurasi dengan beberapa studi kasus di atas, didapatkan suatu analisa yaitu perintah suara yang dilakukan dengan jenis/ kuat lemahnya suara yang berbeda yaitu oleh pembicara 1 (laki-laki) dan oleh pembicara 2 (perempuan) tidak berpengaruh terhadap akurasi sistem dalam mengenali perintah suara dengan perbandingan akurasi rata rata 60,5% dan 56,67%. Hal ini dikarenakan sesuai sifat dalam speech recognition dimana proses dalam pengenalan kata yang diucapkan tidak mempedulikan identitas orang terkait.

BAB VII
PENUTUP

Pada bab ini akan membahas mengenai kesimpulan dan saran yang dapat diambil dari pembuatan aplikasi pengenalan perintah suara menggunakan algoritma *Fast Fourier Transform* dan algoritma *K-Nearest Neighbor*.

7.1 Kesimpulan

Berdasarkan penelitian yang dilakukan dalam pembuatan aplikasi pengenalan perintah suara menggunakan algoritma *Fast Fourier Transform* dan algoritma *K-Nearest Neighbor* dapat ditarik beberapa kesimpulan sebagai berikut:

1. Dalam sistem pengenalan perintah suara berbasis voice recognition, algoritma Fast Fourier Transform kurang dapat berfungsi dan berjalan dengan baik hal ini disebabkan karena dalam pengenalan suara tidak hanya ditentukan dari tinggi rendahnya atau kuat lemahnya amplitudo suara saja, namun juga harus memperhatikan beberapa faktor seperti jeda waktu perintah suara, dan faktor noise.
2. Sistem pengenalan perintah suara dengan menggunakan metode K-Nearest Neighbor sebagai metode pencocokan suara masih kurang baik, karena dilihat dari hasil pengujian akurasi metode ini yang memiliki rata-rata akurasi rendah. Hal ini disebabkan karena pada algoritma KNN masih dimungkinkan kondisi dimana jumlah bobot tiap kata dapat bernilai sama.
3. Akurasi yang didapatkan dari pengujian aplikasi pengenalan perintah suara didapatkan sebesar 60,5% untuk nilai k = 5 dan 56,67%. untuk jenis pembicara yang berbeda.

7.2 Saran

Berdasarkan penelitian yang dilakukan dalam pembuatan aplikasi pengenalan perintah suara menggunakan algoritma *Fast Fourier Transform* dan algoritma *K-Nearest Neighbor* dapat diberikan beberapa saran sebagai berikut:

1. Peningkatan dan pengoptimalan akurasi pada proses pengolahan perintah suara menggunakan algoritma Fast Fourier Transform dapat lebih ditingkatkan lagi. Untuk itu dibutuhkan kombinasi teknik-teknik tertentu dalam membangun suatu sistem pengenalan suara seperti wavelat, penambahan jumlah sampling data, pattern matching menggunakan JST backpropagation, hidden markov model, dan dynamic time wraping.
2. Peningkatan dan pengoptimalan akurasi pada proses pencocokan perintah suara menggunakan algoritma K-Nearest Neighbor dapat lebih ditingkatkan lagi, untuk itu dibutuhkan metode klasifikasi seperti weighted KNN atau dapat juga menggunakan Algoritma Bayesian.
3. Penggunaan teknik filtering, noise reduction, end point detection sehingga sinyal suara digital yang dihasilkan akan lebih baik dari segi kualitas.
4. Penggunaan alat-alat audio (mikrofon dan kartu suara) yang lebih baik sehingga data audio yang diperoleh akan lebih baik kualitasnya.

DAFTAR PUSTAKA

[ALH-03] Alhir, Sinan Si. 2003. *Learning UML*. O'Reilly Media, Inc.

[HAM-06] Hamilton, Kim & Miles, Russell. 2006. *Learning UML 2.0*. O'Reilly Media, Inc.

[HAR-04] Hariyanto, Bambang. 2004. *Rekayasa Sistem Berorientasi Objek*. Informatika Bandung.

[LIU-06] Liu, Liping & Roussev, Boris. 2006. *Management of The Object-Oriented Development Process*. Idea Group Publishing.

[PRE-01] Pressman, Roger S. 2001. *Software Engineering: a practitioner's approach,7th edition*. Mc Graw Hill.

[KHR-11] Nugraha, Khrisna. 2011. *Aplikasi Perintah Suara Dengan Metode Fast Fourier Transform Dan Devide And Concuer Pada Simulasi Rumah Pintar*. Unikom Jakarta.

[JER-11] Rianto, Jeri. 2011. *Perangkat Lunak Pengenalan Suara (Voice Recognition) Untuk Absensi Karyawan Dengan Menggunakan Metode DTW*. Unikom Jakarta.

[NUR-09] Nurlaily. 2009. *Pencocokan Pola Suara Dengan Algoritma FFT dan DC*. Universitas Sumatra Utara Medan.

[WAR-11] Wardana, Jaya. 2011. *Pengenalan Suara Menggunakan Algoritma Fast Fourier Transform (FFT) Dengan Algoritma Mel Frequency Cepstrum Coeffisient (MFCC) Sebagai Ekstrasi Ciri*. Brawijaya Malang.

[HLA-12] Hlavac, Vaclav. 2012. *Classifier Performance Evaluation"*. Center for Machine Perception, Departmen of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Replubic.

[SAM-12] Samsiar, Asri. 2012. *Aplikasi Penganalisis Spektrum Sinyal Digital Secara Real-Time Menggunakan Algoritma Fast Fourier Transform (FFT) Pada Perangkat Mobile*. Brawijaya Malang.

[GIA-09] Giawa Tutorial. 2009. Naudio Codeplex.

LAMPIRAN

Tabel 6.5 Hasil Pengujian akurasi (studi kasus nilai tertangga terdekat/nilai k=5)

No	Pembicara 1 (pria)			
	k = 5			
	Nyala		Mati	
	Sistem	Actual	Sistem	Actual
1	Benar	Benar	Benar	Benar
2	Benar	Benar	Benar	Benar
3	Benar	Benar	Benar	Benar
4	Benar	Benar	Benar	Benar
5	Benar	Benar	Benar	Benar
6	Salah	Salah	Benar	Benar
7	Salah	Benar	Benar	Benar
8	Benar	Benar	Benar	Salah
9	Benar	Benar	Benar	Salah
10	Benar	Benar	Benar	Salah
11	Salah	Benar	Salah	Benar
12	Benar	Benar	Salah	Benar
13	Benar	Salah	Salah	Salah
14	Salah	Salah	Salah	Benar
15	Salah	Salah	Salah	Salah
Jumlah TP	9		7	
Jumlah TN	2		3	
Jumlah FN	3		2	
Jumlah FP	1		3	
Akurasi	73,33%		66,67%	

No	Pembicara 1 (pria)			
	k = 5			
	Terang		Redup	
	Sistem	Actual	Sistem	Actual
1	Benar	Benar	Benar	Benar
2	Benar	Benar	Benar	Benar
3	Benar	Salah	Benar	Salah
4	Benar	Salah	Benar	Salah
5	Benar	Benar	Benar	Benar
6	Benar	Salah	Benar	Benar
7	Benar	Salah	Benar	Benar
8	Benar	Benar	Benar	Benar
9	Benar	Benar	Benar	Salah
10	Benar	Salah	Benar	Salah
11	Salah	Benar	Salah	Benar
12	Salah	Benar	Salah	Benar
13	Salah	Salah	Salah	Salah
14	Salah	Salah	Salah	Salah
15	Salah	Salah	Salah	Salah
Jumlah TP	5		6	
Jumlah TN	2		2	
Jumlah FN	3		3	
Jumlah FP	5		4	
Akurasi	46,67%		53,33%	

Tabel 6.6 Hasil Pengujian akurasi (studi kasus nilai tertangga terdekat/nilai k=3)

No	Pembicara 1 (pria)			
	k = 3			
	Nyala		Mati	
	Sistem	Actual	Sistem	Actual
1	Benar	Benar	Benar	Benar
2	Benar	Benar	Benar	Benar
3	Benar	Benar	Benar	Benar
4	Benar	Benar	Benar	Benar
5	Benar	Benar	Benar	Benar
6	Benar	Salah	Benar	Salah
7	Benar	Benar	Benar	Benar
8	Benar	Salah	Benar	Salah
9	Benar	Benar	Benar	Salah
10	Benar	Benar	Benar	Salah
11	Salah	Benar	Salah	Benar
12	Salah	Salah	Salah	Benar
13	Salah	Benar	Salah	Salah
14	Salah	Salah	Salah	Benar
15	Salah	Salah	Salah	Salah
Jumlah TP	8		6	
Jumlah TN	2		3	
Jumlah FN	3		2	
Jumlah FP	2		4	
Akurasi	66,67%		60%	

No	Pembicara 1 (pria)			
	k = 3			
	Terang		Redup	
	Sistem	Actual	Sistem	Actual
1	Benar	Benar	Benar	Benar
2	Benar	Benar	Benar	Benar
3	Benar	Benar	Benar	Salah
4	Benar	Salah	Benar	Salah
5	Benar	Salah	Benar	Salah
6	Benar	Benar	Benar	Salah
7	Benar	Benar	Benar	Salah
8	Benar	Salah	Benar	Benar
9	Benar	Salah	Benar	Benar
10	Benar	Salah	Benar	Salah
11	Salah	Benar	Salah	Salah
12	Salah	Salah	Salah	Benar
13	Salah	Benar	Salah	Salah
14	Salah	Salah	Salah	Salah
15	Salah	Salah	Salah	Salah
Jumlah TP	5		4	
Jumlah TN	2		1	
Jumlah FN	3		4	
Jumlah FP	5		6	
Akurasi	46,67%		33,33%	

Tabel 6.7 Hasil Pengujian akurasi (studi kasus nilai tertangga terdekat/nilai k=7)

No	Pembicara 1 (pria)			
	k = 7			
	Nyala		Mati	
	Sistem	Actual	Sistem	Actual
1	Benar	Benar	Benar	Benar
2	Benar	Benar	Benar	Benar
3	Benar	Benar	Benar	Benar
4	Benar	Benar	Benar	Salah
5	Benar	Benar	Benar	Benar
6	Benar	Benar	Benar	Salah
7	Benar	Benar	Benar	Salah
8	Benar	Benar	Benar	Salah
9	Benar	Benar	Benar	Salah
10	Benar	Salah	Benar	Salah
11	Salah	Benar	Salah	Salah
12	Salah	Benar	Salah	Salah
13	Salah	Salah	Salah	Benar
14	Salah	Salah	Salah	Benar
15	Salah	Salah	Salah	Salah
Jumlah TP	9		4	
Jumlah TN	2		2	
Jumlah FN	3		3	
Jumlah FP	1		6	
Akurasi	73,33%		40%	

No	Pembicara 1 (pria)			
	k = 7			
	Terang		Redup	
	Sistem	Actual	Sistem	Actual
1	Benar	Benar	Benar	Benar
2	Benar	Benar	Benar	Benar
3	Benar	Salah	Benar	Benar
4	Benar	Salah	Benar	Benar
5	Benar	Salah	Benar	Benar
6	Benar	Salah	Benar	Salah
7	Benar	Salah	Benar	Salah
8	Benar	Salah	Benar	Benar
9	Benar	Benar	Benar	Benar
10	Benar	Benar	Benar	Salah
11	Salah	Benar	Salah	Salah
12	Salah	Salah	Salah	Salah
13	Salah	Benar	Salah	Salah
14	Salah	Salah	Salah	Salah
15	Salah	Salah	Salah	Salah
Jumlah TP	4		7	
Jumlah TN	2		0	
Jumlah FN	3		5	
Jumlah FP	6		3	
Akurasi	40%		46,67%	

Tabel 6.8 Pengujian akurasi (studi kasus jenis pembicara)

No	Pembicara 1 (pria)			
	k = 5			
	Nyala		Mati	
	Sistem	Actual	Sistem	Actual
1	Benar	Benar	Benar	Benar
2	Benar	Benar	Benar	Benar
3	Benar	Benar	Benar	Benar
4	Benar	Benar	Benar	Benar
5	Benar	Benar	Benar	Benar
6	Salah	Salah	Benar	Benar
7	Salah	Benar	Benar	Benar
8	Benar	Benar	Benar	Salah
9	Benar	Benar	Benar	Salah
10	Benar	Benar	Benar	Salah
11	Salah	Benar	Salah	Benar
12	Benar	Benar	Salah	Benar
13	Benar	Salah	Salah	Salah
14	Salah	Salah	Salah	Benar
15	Salah	Salah	Salah	Salah
Jumlah TP	9		7	
Jumlah TN	2		3	
Jumlah FN	3		2	
Jumlah FP	1		3	
Akurasi	73,33%		66,67%	

No	Pembicara 1 (pria)			
	k = 5			
	Terang		Redup	
	Sistem	Actual	Sistem	Actual
1	Benar	Benar	Benar	Benar
2	Benar	Benar	Benar	Benar
3	Benar	Salah	Benar	Salah
4	Benar	Salah	Benar	Salah
5	Benar	Benar	Benar	Benar
6	Benar	Salah	Benar	Benar
7	Benar	Salah	Benar	Benar
8	Benar	Benar	Benar	Benar
9	Benar	Benar	Benar	Salah
10	Benar	Salah	Benar	Salah
11	Salah	Benar	Salah	Benar
12	Salah	Benar	Salah	Benar
13	Salah	Salah	Salah	Salah
14	Salah	Salah	Salah	Salah
15	Salah	Salah	Salah	Salah
Jumlah TP	5		6	
Jumlah TN	2		2	
Jumlah FN	3		3	
Jumlah FP	5		4	
Akurasi	46,67%		53,33%	

No	Pembicara 2 (wanita)			
	k = 5			
	Nyala		Mati	
	Sistem	Actual	Sistem	Actual
1	Benar	Benar	Benar	Benar
2	Benar	Benar	Benar	Salah
3	Salah	Salah	Salah	Benar
4	Salah	Salah	Benar	Benar
5	Benar	Benar	Benar	Salah
6	Salah	Salah	Benar	Salah
7	Salah	Benar	Benar	Benar
8	Benar	Benar	Benar	Salah
9	Benar	Benar	Benar	Benar
10	Benar	Benar	Benar	Salah
11	Salah	Benar	Benar	Benar
12	Benar	Benar	Benar	Benar
13	Benar	Salah	Benar	Benar
14	Salah	Salah	Benar	Salah
15	Salah	Salah	Benar	Salah
Jumlah TP	5		8	
Jumlah TN	2		3	
Jumlah FN	3		2	
Jumlah FP	5		2	
Akurasi	46,67%		73,33%	

No	Pembicara 2 (wanita)			
	k = 5			
	Terang		Redup	
	Sistem	Actual	Sistem	Actual
1	Benar	Benar	Salah	Salah
2	Benar	Benar	Benar	Benar
3	Salah	Salah	Benar	Salah
4	Salah	Salah	Benar	Salah
5	Benar	Benar	Benar	Salah
6	Salah	Salah	Salah	Salah
7	Salah	Benar	Salah	Benar
8	Benar	Benar	Benar	Benar
9	Benar	Benar	Benar	Benar
10	Benar	Benar	Benar	Salah
11	Salah	Benar	Benar	Salah
12	Benar	Benar	Benar	Benar
13	Benar	Salah	Salah	Salah
14	Salah	Salah	Benar	Salah
15	Salah	Salah	Salah	Benar
Jumlah TP	7		5	
Jumlah TN	2		2	
Jumlah FN	3		3	
Jumlah FP	3		5	
Akurasi	60%		46,67%	