PERANCANGAN SPEAKER RECOGNITION PADA SISTEM KENDALI LAMPU BERBASIS MIKROKONTROLER

SPEAKER RECOGNITION DESIGN BASED ON LAMP CONTROL SYSTEM MICROCONTROLLER

Yulia Nur Utami[1] , R.Rumani[2], Nurfitri Anbaranti[3]

Prodi S1 Sistem Komputer, Fakultas Teknik, Universitas Telkom
[1]yulianurutami08@gmail.com. [2] rumani@telkomuniversity.ac.id. [3] anbarsanti@gmail.com

Abstrak

Teknologi merupakan suatu hal yang sangat bermanfaat bagi kehidupan banyak orang untuk saat ini. Semua aspek kehidupan dapat memanfaatkan teknologi sesuai dengan bidang yang dibutuhkan, salah satunya adalah teknologi kendali rumah. Suatu rumah dapat diautomatisasikan hanya dengan suara saja. Untuk membuat sistem tersebut maka dibutuhkan suatu perangkat yang mendukung untuk memproses suara dan mengontrol perangkat rumah. Pada tugas akhir ini, penulis telah merancang dan mengimplementasikan sebuah sistem lamp u otomatis menggunakan suara yang berbasis mikrokontroler Arduino Uno. Secara garis besar sistem ini menggunakan dua buah metode yaitu MFCCs sebagai ekstraksi ciri dan jaringan saraf tiruan *Backpropagation* sebagai pencocokan cirinya. Keluaran dari sistem ini berupa koefisien MFCCs yang akan diteruskan sebagai masukan untuk proses pencocokan ciri pada jaringan saraf tiruan *Backpropagation*. Pada proses inilah Arduino akan mengaktifkan *relay* untuk menyalakan ataupun mematikan lampu. Dengan sistem ini, dihasilkan 80.23% akurasi kebenaran sistem.

Kata kunci MFCC, *Backpropagation*, Arduino lampu otomatis

Abstract

Technology is something that is very beneficial for many people's lives for the moment. All aspects of life can take advantage of technology in accordance with the required fields, one of which is the control of the house. A house can be in control just by sound alone. To make such a system is needed for a device that supports voice processing and control home devices. In this thesis, the author has designed and implemented an automated lighting system uses voice-based microcontroller Arduino Uno. Broadly speaking, this system uses two methods, namely MFCCs as feature extraction and neural network Backpropagation as matching characteristics. The output of this system in the form of coefficients MFCCs are forwarded input to the process of matching the characteristics of neural network Backpropagation. In the Arduino this process will enable the relay to turn on or turn off the lights. With this system, generated 80.23% of accuracy truth system.

Keywords: *MFCC, Backpropagation, Arduino, automatic lamp*

1. Pendahuluan

Perkembangan ilmu pengetahuan dan teknologi saat ini telah memberikan dampak yang sangat besar dalam kehidupan manusia. Salah satu teknologi yang sekarang sedang banyak dikembangkan adalah teknologi biometrik. Teknologi ini adalah suatu sistem pengenalan seseorang / individu berdasarkan atas ciri khas / sifat biologis yang dimiliki oleh seseorang. Berbagai penelitian telah dilakukan dalam pengidentifikasian individu berdasarkan ciri biologis seperti pola retina, roman muka, sidik jari, telapak tangan, *dental*, dll. Seiring dengan perkembangan ilmu pengetahuan, kemudian dilakukan pengidentifikasian individu melalui suara.

Pada tugas akhir ini, dilakukan perancangan sistem *speaker recognition* yang dapat mengidentifikasi suara pembicara yang akan digunakan dalam menyalakan dan mematikan lampu. Pada penelitian ini dilakukan perekaman suara oleh lima pembicara pada ruang kedap suara yang nantinya akan di jadikan sebagai input pada sistem. Suara pembicara akan direkam dalam ekstensi *wav dan akan menjadi masukan pada sistem ekstraksi ciri dengan metoda *Mel-Frequency Cepstrum Coefficients (MFCC)* dan kemudian ciri dari MFCC akan diteruskan sebagai masukan pada pengklasifikasian jaringan saraf tiruan *backropagation.*

2. Dasar Teori

2.1 Pengenalan Suara[1]

Speaker recognition adalah suatu proses pengenalan pembicara dari informasi yang terkandung dalam gelombang suara yang diinputkan. *Speaker recognition* dibagi menjadi dua bagian, yaitu *speaker verification* dan *speaker identification*. *Speaker verification* adalah proses pemeriksaan ulang seorang pembicara, sehingga perlu diketahui terlebih dahulu identitas pembicara tersebut berdasarkan data yang telah dimasukkan (misalnya *username* dan *password*). *Speaker verification* membandingkan fitur suara dari seorang pembicara secara langsung dengan fitur pembicara tertentu yang ada didalam *database*. Bila hasil perbandingan tersebut lebih besar atau sama dengan batasan tertentu (*threshold*), maka pembicara tersebut diterima, bila tidak maka ditolak. *Speaker identification* adalah proses untuk mencari dan mendapatkan identitas dari seorang pembicara dengan membandingkan fitur suara yang dimasukkan dengan semua fitur suara pembicara yang ada didalam *database*. Berbeda dengan pada *speaker verification*, proses ini melakukan perbandingan *one-to-many* (1:N). Bila dibandingkan dengan *speaker verification*, maka *speaker identification* lebih sulit. Hal ini disebabkan, bila jumlah pembicara yang terdaftar dalam database semakin besar, maka kemungkinan terjadinya kesalahan dalam pengambilan keputusan semakin besar pula.

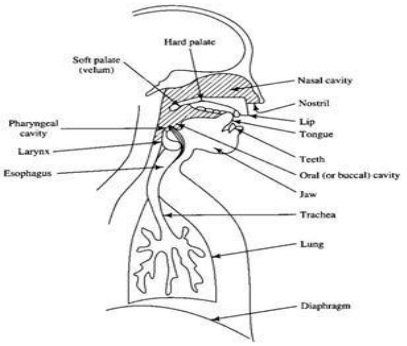
Lebih jauh *speaker recognition* dapat diklasifikasikan menjadi *text-dependent* dan *text-independent*. Pada *text-dependent*, kata-kata yang diucapkan oleh pembicara sudah ditentukan sebelumnya. Sedangkan pada *text-independent*, tidak ada asumsi kata-kata yang akan diucapkan oleh pembicara, sehingga sistem harus memodelkan fitur-fitur umum dari suara seorang pembicara[2].

Pada *speaker recognition* ada beberapa faktor yang dapat menyebabkan kesalahan dalam proses verifikasi dan identifikasi, antara lain :

1. Kesalahan dalam pengucapan (*misspoken*) dan pembacaan (*missread*) frasa.
2. Keadaan emosional yang ekstrim (misalnya stres).
3. Pergantian penempatan mikrofon (*intrasession* atau *intersession*).
4. Kekurangan atau ketidak-konsistenan akustik dari ruangan (misalnya *multipath* dan *noise*).
5. *Channel mismatch* (misalnya penggunaan mikrophon yang berbeda *channel* dalam perekaman dan verifikasi).
6. Sakit (misalnya flu yang dapat merubah *vocal tract*).
7. *Aging* (model *vocal tract* dapat berubah berdasarkan usia).

2.2 Sinyal Suara[4]

Sinyal suara adalah sinyal yang dihasilkan oleh suara manusia dan biasanya mempunyai frekuensi kerja antara 0 sampai 5 KHz. Bentuk gelombang sinyal suara mempunyai bentuk yang unik. Sinyal suara mempunyai unsur bunyi terkecil yang disebut sebagai *pitch*. Panjang *pitch* berkisar 10 ms. *Pitch* manusia berbeda antara satu dengan yang lainnya, terutama ditinjau dari jenis kelamin (laki-laki atau perempuan) serta usia.



Gambar 2.1 Organ Produksi Suara Manusia

Organ tubuh yang terlibat pada proses produksi suara adalah : paru-paru, tenggorokan (*trachea*), laring (*larynx*), faring (*pharynx*), pita suara (*vocal cord*), rongga mulut (*oral cavity*), rongga hidung (*nasal cavity*), lidah (*tongue*), dan bibir (*lips*), seperti dapat dilihat pada gambar diatas.

Organ tubuh ini dapat dikelompokkan menjadi tiga bagian utama, yaitu : *vocal tract* (berawal di awal bukaan pita suara atau *glottis*, dan berakhir di bibir), *nasal tract* (dari velum sampai nostril), dan *source generator* (terdiri dari paru-paru, tenggorokan, dan *larynx*). Ukuran *vocal tract* bervariasi untuk setiap individu, namun untuk laki-laki dewasa rata-rata panjangnya sekitar 17 cm. Luas dari *vocal tract* juga bervariasi antara 0 (ketika seluruhnya tertutup) hingga sekitar 20 cm. Ketika *velum*, organ yang memiliki fungsi sebagai pintu penghubung antara *vocal tract* dengan *nasal tract* terbuka, maka secara akustik *nasal tract* akan bergandengan dengan *vocal tract* untuk menghasilkan suara *nasal*.

Berdasarkan sinyal eksitasi yang dihasilkan pada proses produksi suara, sinyal suara ucapan dapat dibagi menjadi tiga bagian yaitu *silence*, *unvoiced*, dan *voiced*:

a. Sinyal *silence* : sinyal pada saat tidak terjadi proses produksi suara ucapan, dan sinyal yang diterima oleh pendengar dianggap sebagai bising latar belakang.
b. Sinyal *unvoiced* : terjadi pada saat pita suara tidak bergetar, dimana sinyal eksitasi berupa sinyal *random*.
c. Sinyal *voiced* : terjadi jika pita suara bergetar, yaitu pada saat sinyal eksitasi berupa sinyal pulsa kuasi-periodik. Selama terjadinya sinyal *voiced* ini, pita suara bergetar pada frekuensi fundamental – inilah yang dikenal sebagai *pitch* dari suara tersebut.

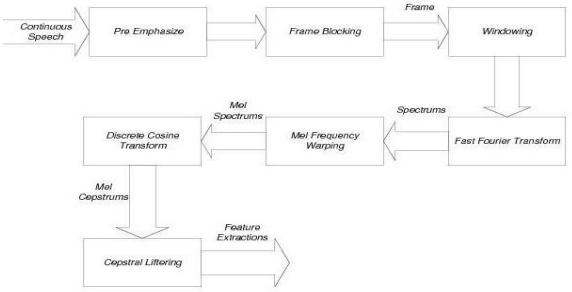
2.3 *Mel Frequency Cepstrum Coefficients* (MFCC) [5]

MFCC (*Mel Frequecy Cepstrum Coefficients*) merupakan salah satu metode yang banyak digunakan dalam bidang *speech technology*, baik *speaker recognition* maupun *speech recognition*. Metode ini digunakan untuk melakukan *feature extraction* yang merupkan sebuah proses yang mengkonversikan sinyal suara menjadi beberapa parameter, keunggulan dari metode ini adalah:

1. Mampu untuk menangkap karakteristik suara yang sangat penting bagi pengenalan suara. Atau dengan kata lain, mampu menangkap informasi-informasi penting yang terkandung dalam sinyal suara..
2. Menghasilkan data seminimal mungkin, tanpa menghilangkan informasi-informasi penting yang ada.
3. Mengadaptasi organ pendengaran manusia dalam melakukan persepsi terhadap sinyal suara.

Perhitungan yang dilakukan MFCC menggunakan dasar dari perhitungan *short-term analysis*. Hal ini dilakukan mengingat sinyal suara yang bersifat *quasi stationary*. Sinyal suara pada periode waktu yang cukup pendek (sekitar 10-30 ms) akan menunjukkan karakteristik sinyal suara yang *stationary*. Tetapi bila dilihat dalam periode waktu yang lebih panjang karakteristik sinyal suara akan terus berubah sesuai dengan kata yang diucapkan.

MFCC *feature extraction* sebenarnya merupakan adaptasi dari sistem pendengaran manusia, dimana sinyal suara akan difilter secara linear untuk frekuensi rendah (dibawah 1KHz) dan secara logaritmik untuk frekuensi tinggi (diatas 1 KHz) [3].

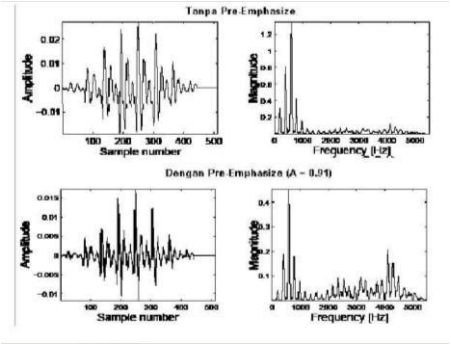


Gambar 2.2 Blok Diagram MFCC

2.3.1 *Pre-emphasize Filtering* [7]

Pre-emphasize Filtering merupakan salah satu jenis filter yang sering digunakan sebelum sebuah sinyal diproses lebih lanjut. Filter ini mempertahankan frekuensi-frekuensi tinggi pada sebuah spektrum yang umumnya tereliminasi pada saat proses produksi suara. Tujuan dari *pre-emphasize* filter ini adalah:

1. Mengurangi *Noise ratio* pada sinyal, sehingga dapat meningkatkan kualitas sinyal
2. Menyeimbangkan spektrum dari *voice sound*. Pada saat memproduksi *voice sound*, glotis manusia menghasilkan sekitar -12 dB *octave slope*. Namun ketika energi akustik tersebut dikeluarkan melalui bibir, terjadi peningkatan sebesar
   +6 dB. Sehingga sinyal yang terekam di *microphone* adalah sekitar -6 db *octave slope*.



Gambar 2.3 Contoh dari *Pre-Emphasize* pada sebuah *Frame*

Perhatikan perbedaan pada frekuensi domain akibat diimplementasikannya *pre-emphasize* filter. Pada gambar diatas terlihat bahwa distribusi energi pada setiap frekuensi menjadi lebih seimbang setelah diimplementasikannya *pre-emphasize* filter. Bentuk yang paling umum digunakan dalam pre-emphasize filter adalah sebagai berikut:

$$H(z) = 1 - \alpha z^{-1}$$

Dimana $0.9 \leq \alpha \leq 1$ dan α R. Formula di atas dapat diimplementasikan sebagai *first order differentiator*, sebagai berikut:

$$y[n] = s[n] - \alpha s[n-1]$$

Keterangan :

y[n] :Sinyal hasil *pre-emphasize* filter

s[n] :Sinyal sebelum *pre-emphasize* filter

Pada umumnya nilai α yang paling sering digunakan adalah antara 0.9 sampai dengan 1.

2.3.2 *Frame Blocking*[7]

Dalam langkah ini sinyal wicara kontinyu diblok menjadi *frame-frame* N sampel. *Frame* pertama terdiri dari *N* sampel pertama. *Frame* kedua dengan *M* sampel setelah *frame* pertama, dan *overlap* dengan N–M sampel. Dengan cara yang sama, *frame* ketiga dimulai 2M sampel setelah *frame* pertama (atau M sampel setelah *frame* kedua) dan *overlap* dengan N–2M sampel. Proses ini berlanjut hingga semua wicara dihitung dalam satu atau banyak *frame*.

Proses *framing* ini dilakukan terus sampai seluruh sinyal dapat terproses. Selain itu, proses ini umumnya dilakukan secara *overlapping* untuk setiap *frame*-nya. Panjang daerah *overlap* yang umum digunakan adalah kurang lebih 30% sampai 50% dari panjang *frame.*

2.3.3 *Windowing*[7]

Proses *windowing* ini dilakukan pada setiap frame hasil dari proses *framing*. Hal ini dilakukan untuk mengurangi terjadinya kebocoran spektral atau aliasing.

Rumus berikut menunjukkan fungsi *window* terhadap sinyal suara yang diinputkan.

$$x(n) = x_1(n)\, w(n)$$

Keterangan :

n : 0,1,....., N-1

x(n) : Nilai Sampel Sinyal

$x_i(n)$: Nilai Sampel dari *Frame* Sinyal ke i

w(n) : Fungsi *window*

Fungsi *window* yang paling sering digunakan dalam aplikasi *speaker recognition* adalah *Hamming Window*. Berikut persamaan dari *hamming window* :

w(n) = 0.54 – 0.46 cos (——)

[7]

2.3.4 *Fast Fourier Transform* (FFT)

Setiap *frame* dikonversi dari domain waktu ke domain frekuensi untuk mendapatkan spektrum frekuensinya. Hal ini dilakukan untuk mempermudah komputasi dan analisa. Metode yang dilakukan untuk mendapatkan spektrum frekuensi adalah *Fast Fourier Transform*.

Berikut adalah rumusan untuk perhitungan FFT :

, n=0,1,2,….,N-1

Keterangan :

: Deretan aperiodik dengan nilai N

N : Jumlah sampel (kelipatan 2)

2.3.5 *Mel Frequency Wrapping*[7]

Mel Frequency Warpping umumnya dilakukan dengan menggunakan *filterbank*. *Filterbank* adalah salah satu bentuk *filter* yang dilakukan dengan tujuan untuk mengetahui ukuran dari *band* frekuensi tertentu dalam sinyal suara, pada MFCC digunakan Mel *Filterbank* yang menggunakan skala mel. Skala mel bersifat linear pada frekuensi dibawah 1000 Hz, dan logaritmik pada frekuensi diatas 1000 Hz. Untuk mencari skala mel, persamaan yang digunakan:

——

2.3.6 *Discrete Cosine Transform*[7]

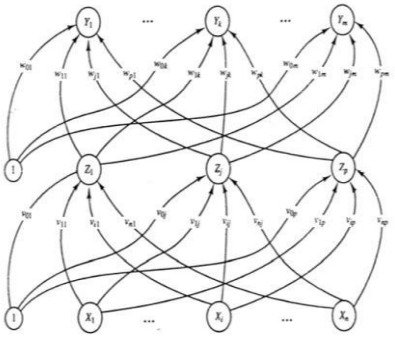
Langkah selanjutnya yaitu mengubah spektrum *log mel* menjadi domain waktu. Hasil ini

disebut *mel frequency cepstrum coefficient* (MFCC). Reprentasi *cepstral* dari spektrum wicara memberikan reprentasi baik dari sifat-sifat spektral lokal sinyal untuk analisis *frame* yang diketahui. Karena koefisien *mel* spektrum adalah bilangan nyata. Dengan mengubahnya menjadi domain waktu menggunakan *discrete cosine transform* (DCT). Jika koefisien spektrum daya *mel* hasilnya adalah , k = 1,2,..., K, sehingga MFCC dapat dihitung adalah

Dimana adalah koefisien cepstrum *mel-frequency* dan adalah koefisien daya *mel.*

2.4 Jaringan Saraf Tiruan (JST) Propagasi Balik (*Backpropagation*) [4]

Jaringan saraf tiruan *backpropagation* merupakan pembelajaran terawasi (*supervised learning*). Didalam *backpropagation*, setiap unit yang berada pada lapisan input terhubung dengan setiap unit yang ada dilapisan tersembunyi. Dan setiap unit pada lapisan tersembunyi terhubung pula dengan setiap lapisan output.



Gambar 2.4 *Backpropagation* dengan 1 *hidden layer*

Algoritma pada *backpropagation* dibagi ke dalam 2 bagian:

1. Algoritma pelatihan
   Terdiri dari 3 tahap yaitu tahap umpan maju pola pelatihan *input*, tahap propagasi balik *error*, dan tahap pengaturan bobot.
2. Algoritma aplikasi
   Yang digunakan hanyalah tahap umpan maju saja.

Algoritma Pelatihan[6]

1. Inisialisasi bobot-bobot.
   Tentukan angka pembelajaran (α) atau *learning rate.*
   Tentukan pula nilai toleransi *error* atau nilai ambang (bila menggunakan nilai ambang sebagai kondisi berhenti); atau *set* maksimal *epoch* (bila menggunakan banyaknya *epoch* sebagai kondisi berhenti).
2. *While* kondisi berhenti tidak terpenuhi *do* langkah ke-2 sampai langkah ke-10.

Tahap Umpan Maju

3. Setiap unit *input* (dari unit ke-1 sampai unit ke-n pada lapisan *input*) mengirimkan sinyal *input* ke semua unit yang ada dilapisan atasnya (ke lapisan tersembunyi): .
4. Pada setiap unit di lapisan tersembunyi (dari unit ke-1 sampai unit ke-p; i=1,…,n; j=1,…,p) sinyal *output* lapisan tersembunyinya dihitung dengan menerapkan fungsi aktivasi terhadap penjumlahan sinyal-sinyal *input* berbobot :
   = f( )
   Kemudian dikirim ke semua unit dilapisan atasnya.
5. Setiap unit dilapisan *output* (dari unit ke-1 sampai unit ke-m; i=1,…,n; k=1,..,m) dihitung sinyal outputnya dengan menerapkan fungsi aktivasi terhadap penjumlahan sinyal-sinyal *input* berbobot bagi lapisan ini:
   = f(+)

Tahap Pemropagasibalikan *Error*

6. Setiap unit *output* (dari unit ke-1 sampai unit ke-m; j=1,..,p; k=1,..,m) menerima pola target lalu informasi kesalahan lapisan *output* ( ) dihitung. dikirim ke lapisan dibawahnya dan digunakan untuk menghitung besar koreksi bobot dan bias (Δ dan Δ) antara lapisan tersembunyi dengan lapisan *output* :
   = (-) f'(+)
7. Pada setiap unit dilapisan tersembunyi (dari unit ke-1 sampai unit ke-p; i=1,…,n; j=1,…,p;k=1,…,m) dilakukan perhitungan

informasi kesalahan lapisan tersembunyi ( ). kemudian digunakan untuk menghitung besar koreksi bobot dan bias (Δ dan Δ) antara lapisan *input* dan lapisan tersembunyi.

= ( ) f'(+)

= α

= α

Tahap Peng-*update*-an Bobot dan Bias

8. Pada setiap unit *output* (dari unit ke-1 sampai unit ke-m) dilakukan peng-*update*-an bias dan bobot (j=0,..,p; k=1,..,m) sehingga bias dan bobot yang baru menjadi :

(baru) = (lama) + Δ

Dari unit ke-1 sampai unit ke-p di lapisan tersembunyi juga dilakukan peng-update-an pada bias dan bobotnya (i=0,…,n;j=1,..,p):

(baru) = (lama) + Δ

9. Tes kondisi berhenti

Algoritma Aplikasi

1. Inisialisasi bobot yang diambil dari bobot-bobot terakhir yang diperoleh dari algoritma pelatihan.
2. Untuk setiap *vector input*, laukan langkah ke-3 sampai ke-5.
3. Setiap unit *input* (dari unit ke-1 sampai unit ke-n pada lapisan *input*; i=1,..,n) menerima sinyal *input* sinyal pengujian dan menyiarkan sinyal ke semua unit pada lapisan di atasnya (unit-unit tersembunyi):
4. Setiap unit di lapisan tersembunyi (dari unit ke-1 sampai unit ke-p; i=1,…,n; j=1,…,p) menghitung sinyal *output*nya dengan menerapkan fungsi aktivasi terhadap penjumlahan sinyal-sinyal *input* . Sinyal *output* dari lapisan tersembunyi kemudian dikirim ke semua unit pada lapisan di atasnya:

= f( )

5. Setiap unit *output* (dari unit ke-1 sampai unit ke-m; j=1,..,p; k=1,..,m) menghitung sinyal *output*nya dengan menerapkan fungsi aktivasi terhadap penjumlahan sinyal-sinyal *input* bagi lapisan ini, yaitu sinyal-sinyal *input* dari lapisan tersembunyi :

= f(+)

2.5 Arduino Uno[8]

Arduino Uno adalah *board* mikrokontroler berbasis AMEGA 328 (*datasheet*). Memiliki 14 pin *input* dari *output* digital dimana 6 pin *input* tersebut dapat digunakan sebagai *output* PWM dan 6 pin *input* analog, 16 MHz osilator kristal, koneksi USB, *jack power*, ICSP *header*, dan tombol *reset*. Untuk mendukung mikrokontroler agar dapat digunakan, cukup hanya menghubungkan *Board* Arduino Uno ke komputer dengan menggunakan kabel USB atau listrik dengan AC yang-ke *adaptor*-DC atau baterai untuk menjalankannya.

Uno berbeda dari semua papan sebelumnya dalam hal itu tidak menggunakan FTDI *chip driver* USB-*to-serial*. Sebaliknya, fitur Atmega16U2 (Atmega8U2 hingga versi R2) diprogram sebagai konverter USB-*to-serial*. Revisi 2 dari dewan Uno memiliki resistor menarik garis 8U2 HWB ke tanah, sehingga lebih mudah untuk dimasukkan ke dalam model DFU .

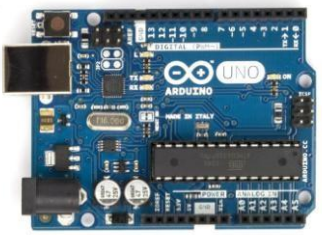
Spesifikasi Arduino Uno :

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (*of which* 6 *provide* PWM *output*)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC *Current* for 3.3V *Pin*	50 mA
Flash Memory	32 KB (ATmega328) *of which* 0.5 KB *used by bootloader*

SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz
Length	68.6 mm

Width	53.4 mm
Weight	25 g

Arduino Uno dapat diaktifkan melalui koneksi USB atau dengan satu daya eksternal. Sumber daya dipilih secara otomatis. Eksternal (*non*-USB) dapat di ambil baik berasal dari AC ke adaptor DC atau baterai. *Adaptor* ini dapat dihubungkan dengan menancapkan *plug jack* pusat-positif ukuran 2.1mm konektor *POWER*. Ujung kepala dari baterai dapat dimasukkan kedalam Gnd dan Vin pin *header* dari konektor *POWER*. Kisaran kebutuhan daya yang disarankan untuk *board* Uno adalah 7 sampai dengan 12 volt, jika diberi daya kurang dari 7 volt kemungkinan pin 5v Uno dapat beroperasi tetapi tidak stabil kemudian jikadiberi daya lebih dari 12V, *regulator* tegangan bisa panas dan dapat merusak *board* Uno.

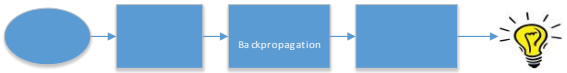


Gambar 2.5 Arduino Uno

3. Analisi dan Perancangan Sistem

3.1 Perancangan Sistem

Secara umum tujuan dari perancangan sistem yang dibuat adalah untuk melakukan kontrol menghidupkan atau mematikan lampu dengan menggunakan suara melalui Arduino Uno.



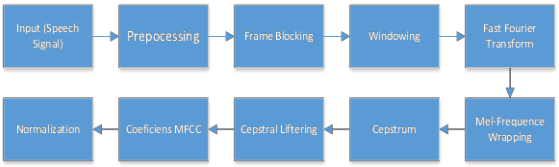
Gambar 3.1 Model Sistem

Cara kerja system :

Gambar 3.2 Diagram Blok Cara Kerja Sistem

3.2 Perancangan Software

Implementasi metode ekstraksi ciri serta klasifikasi dilakukan dengan menggunakan *Microsoft Visual Studio* 2012, secara umum pada ekstaksi ciri MFCC dapat digambarkan sebagai berikut :



Gambar 3.3 Diagram Blok MFCC

3.2.1 *Preprocessing*

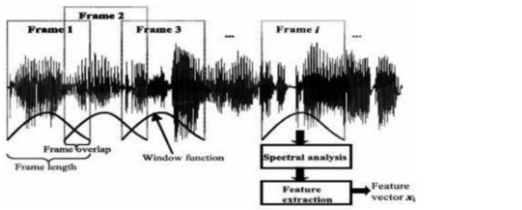
Preprocessing merupakan proses awal yang dilakukan pada *input* yang bertujuan untuk memperbaiki data sebelum dilakukan pengambilan ciri pada data tersebut. Sinyal suara asli pertama akan dilakukan proses normalisasi amplitudo. Dimana proses ini dilakukan untuk menyetarakan semua sinyal suara dalam rentang 1 sampai dengan -1. Hal ini dilakukan agar data yang telah direkam kuat lemahnya suara yang masuk pada sistem berada pada *level* amplitudo yang sama. Setelah dilakukan proses normalisasi amplitudo, dilakukan proses *cropping* sinyal suara.

Proses ini dilakukan untuk membuang sinyal awal dan sinyal akhir. Proses *cropping* ini menggunakan nilai *threshold*, dimana nilai yang berada dibawah nilai *threshold* akan dibuang. Sehingga data yang masuk pada proses selanjutnya lebih sedikit dari pada data asli. Proses *preprocessing* yang terakhir adalah *preemphasis filtering*. Filter *pre-emphasis* sering digunakan pada pengolahan suara sebelum suara diproses lebih lanjut. Hal ini dikarenakan filter ini dapat mempertahankan frekuensi-frekuensi tinggi pada sebuah spektrum, yang umumnya tereleminasi pada saat produksi suara. Filter ini juga dapat digunakan untuk mengurangi *noise rasio* pada sinyal sehingga hal ini dapat meningkatkan kualitas sinyal.

3.2.2 *Frame Blocking*

Karena sinyal suara terus mengalami perubahan akibat adanya pergeseran artikulasi dari organ produksi vokal, sinyal harus diproses secara *short segments (short frame).* Panjang *frame* yang biasanya digunakan untuk pemrosesan sinyal adalah antara 10-30 milidetik. Panjang *frame* yang digunakan sangat mempengaruhi keberhasilan dalam analisa *spektral*. Disatu sisi ukuran *frame*

harus sepanjang mungkin untuk dapat menunjukan resolusi frekuensi yang baik. Tetapi disisi lain, ukuran *frame* juga harus cukup pendek untuk dapat menunjukan resolusi waktu yang baik Penentuan jumlah panjang *frame* sebaiknya adalah kelipatan untuk memfasilitasi proses FFT yang ada pada blok berikutnya. Jika tidak terpenuhi maka panjang *frame* yang tidak mencapai jumlah kelipatan dilakukan *zero padding* pada data setelahnya.



Gambar 3.4 Contoh *Framing* dan *Overlap* pada Sebuah Sinyal

Pada penelitian ini, sinyal dipotong menjadi *frame-frame* dengan nilai 32 ms (256 *sample*) dengan ukuran *overlap* tiap *slot* adalah setengah dari nilai panjang satu *frame*.

3.2.3 *Windowing*

Proses *framing* dapat menyebabkan terjadinya kebocoran spektral (*spektral leakage*) atau aliasing. Aliasing adalah sinyal baru dimana memiliki frekuensi yang berbeda dengan sinyal aslinya. Efek ini dapat terjadi karena rendahnya jumlah *sampling rate*, ataupun karena proses *frame blocking* dimana menyebabkan sinyal menjadi *discontinue*. Untuk mengurangi kemungkinan terjadinya kebocoran spektral, maka hasil dari proses *framing* harus melewati proses *window*. Sebuah fungsi *window* yang baik harus menyempit pada bagian *main lobe* dan melebar pada bagian *side lobe*nya Berikut ini adalah representasi dari fungsi *window* terhadap sinyal suara *input*.

[n]= [n].w[n], n=0,1,…,N

Dimana : [n] = hasil sinyal *windowing* pada *frame* ke-i
[n] = sinyal pada *frame* ke-i
w[n] = fungsi *window* yang digunakan
N = panjang *frame* yang digunakan

Fungsi *window* yang paling sering digunakan dalam aplikasi *speaker recognition* adalah *Hamming Window*. Berikut persamaan dari *hamming window* :

w(n) = 0.54 – 0.46 cos (——)

3.2.4 FFT

Sinyal pada pada langkah-langkah sebelumnya merupakan sinyal yang berada dalam domain waktu. Untuk mempermudah perhitungan dalam proses selanjutnya (*mel-filterbank*) maka sinyal diubah ke dalam domain waktu menggunakan proses FFT. FFT mempunyai tujuan yang sama dengan DFT tetapi dalam komputasi proses perhitungan FFT akan lebih cepat karena FFT akan mereduksi proses *looping* yang terjadi pada DFT. Modifikasi yang dilakukan adalah dengan cara mengelompokan batas n ganjil dan batas n genap, sehingga N *point* DFT menjadi (N/2) *point*

Jika persamaan diatas (DFT) batasnya dibagi menjadi genap dan ganjil sebagai berikut

–

Dimana :

–

,

yang mana

cos(-2) + i sin(-2) = 1

–

–

–

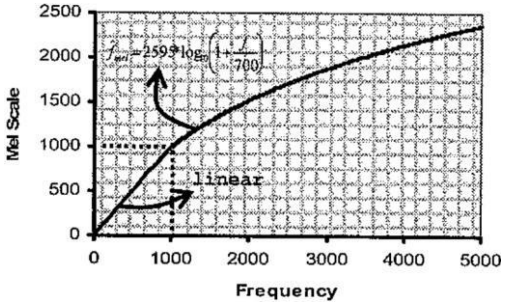
<N/2 *point* DFT genap><N/2 DFT ganjil>

Proses FFT disini merupakan bilangan kompleks yang mempunyai nilai *real* dan *imaginer*. Untuk memperoleh nilai *magnitude* dari *spectrum* dapat dilakukan dengan perhitungan :

Dengan melakukan proses FFT tersebut, maka akan didapatkan nilai x(k), dimana masing-masing nilai tersebut merupakan representasi frekuensi dasar dari sinyal *input*. Nilai ini nantinya akan dikalikan dengan *filterbank* pada proses selanjutnya.

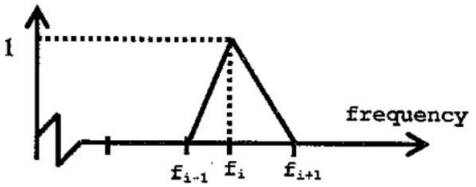
3.2.5 *Mel-Frequency Wrapping*

Tahap ini merupakan proses pengfilteran dari *spectrum* frekuensi setiap *frame* yang diperoleh dari tahapan sebelumnya, menggunakan sejumlah M *filterbank*. Filter ini dibuat dengan mengikuti persepsi telinga manusia dalam menerima suara. Persepsi ini dinyatakan dalam skala *'mel'* (berasal dari *Melody*) yang mempunyai hubungan tidak linear dengan frekuensi suara. Frekuensi (Hz) yang sebenarnya dalam sebuah sinyal akan diukur manusia secara subyektif dengan menggunakan *mel scale*. *Mel frequency scale* adalah linear untuk frekuensi kurang dari 1000 Hz dan logaritmik untuk frekuensi di atas 1000 Hz.



Gambar 3.5 Hubungan Antara Frekuensi dengan Skala Mel

Representasi hubungan tersebut kemudian diterapkan pada fungsi filter segitiga dengan tinggi tertentu yang nantinya akan menjadi faktor pengali pada *frame-frame* sinyal *input* yang berasal dari proses FFT.



Gambar 3.6 Contoh Satu Filter segitiga dengan Tinggi 1

3.2.6 *Cepstrum*

Tahap ini merupakan tahap penentuan ciri MFCC. Skala *mel-frequency*akan dikonversi dari koefisien spektrum mel menjadi domain waktu menggunakan *Discrate Cosine Transform (DCT)*.

Dengan : j = 1,2,3,…,K ;K adalah banyaknya koefisien MFCC yang diinginkan
M = banyaknya filter segitiga
xi= koefisien spektrum *mel* yang diperoleh dari persamaan diatas
Dalam hal ini disebut sebagai koefisien ke j dari *mel frequency cepstralcoefficients* (MFCC).

3.2.7 *Cepstral Liftering*

Hasil dari fungsi DCT adalah *cepstrum* yang sebenarnya sudah merupakan hasil akhir dari proses MFCC. Tetapi untuk meminimalisasi sensitifitas dari koefisien MFCC yang telah didapat, maka *cepstrum* hasil dari DCT akan diolah lagi pada blok *ceptral liftering*. Sensitifitas tersebut adalah *low order* dari *cepstral coefficients* sangat sensitif terhadap *spektralslope*, sedangkan bagaian *high ordernya* sangat sensitif terhadap *noise* *Cepstral liftering* dapat dilakukan dengan mengimplementasikan fungsi *window* terhadap hasil koefisien MFCC.

– —

L = jumlah koefisien MFCC
n = indeks dari koefisien

3.2.8 Normalisasi

Normalisasi dilakukan dengan membagi semua koefisien MFCC dengan nilai maksimum *absolut* yang ada pada vektor koefisien MFCC. Proses ini dilakukan agar nilai-nilai koefisien tidak mempunyai *range* yang terlalu besar sehingga proses komputasi sinyal untuk pengolahan klasifikasi tidak terlalu berat. Pada tahap ini nilai koefisien MFCC berada pada batas 1 dan -1.

Setelah ditentukan proses ekstraksi ciri dan klasifikasi, kemudian hasil klasifikasi dikirim melalui *port serial* FDTI ke Arduino Uno agar dapat menampilkan hasil keluaran yang sudah diproses pada sistem dengan cara membandingkan hasil klasifikasi dengan data yang ada pada *database*. Proses penentuan *database* melibatkan lima *user* dengan masing-masing menyebutkan perintah-perintah yang nanti digunakan pada implementasi di Arduino Uno. Perintah tersebut adalah '*on'* dan '*off'*.

Database diperoleh dengan *Cool Edit Pro 2.1* berupa *file* suara bertipe .wav, dengan panjang tiap suara adalah 0.6 detik. berikut jumlah *database* masing-masing *user* pada setiap perintah :

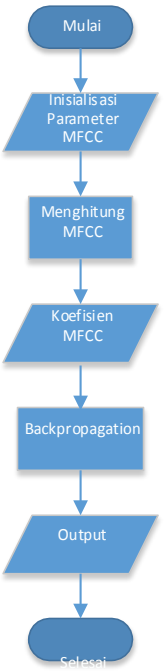
Tabel 3.1 Jumlah *Database*Suara *User*

perintah / user	On	Off
Mirza	30	30
Dedek	28	30
Puput	30	30
Ninun	30	30
Yulia	30	30
Jumlah	148	150

Database latih diatas akan diolah untuk ditentukan koefisien-koefisien ciri MFCC, koefisien ciri ini yang nantinya akan dijadikan vektor acuan pada proses klasifikasi menggunakan *Backpropagation*. Diagram blok dalam proses pengambilan *database* koefisien MFCC adalah sebagai berikut :

Gambar 3.7 Proses Penentuan koefisien *MFCC*

Secara umum proses pengambilan keputusan pada Visual Studio 2012 adalah sebagai berikut :



Gambar 3.8 *Flowchart* Program pada Visual Studio 2012

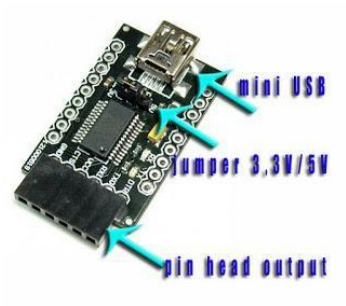
Pada *flowchart* diatas *input* suara akan diinisialisasi oleh proses awal MFCC yang kemudian masuk tahap proses menghitung MFCC dan menghasilkan keluaran MFCC berupa matriks suara yang sudah dikenali cirinya dengan baik. Keluaran MFCC yang berupa matriks suara yang sudah dinormalisasikan akan menjadi masukan bagi proses pengklasifikasian menggunakan *backpropagation*. Matriks suara yang dihasilkan di hitung bobot dan dan diberikan bobot baru sebagai bagian dari proses *backpropagation* yang kemudian hasilnya akan dibandingkan nilainya dengan koefisien yang sudah ditentukan di awal proses. Nilai-nilai yang diperoleh dalam perhitungan tersebut akan saling dibandingkan, sehingga diperoleh satu nilai terkecil.

3.3 Perancangan Hardware

Rangkaian sistem yang penulis gunakan adalah Arduino Uno.

3.3.1 FTDI *Basic Breakout* 3.3/5V

Kabel FTDI adalah USB *to serial* (TTL *level*) konverter yang mengijinkan cara sederhana untuk menghubungkan *interface* TTL ke USB. Pin VCC dari kabel FTDI dikonfigurasi agar beroperasi pada tegangan 5V dengan 3.3V I/O.



Gambar 3.9 *FDTI*

Secara teknikal Arduino menggunakan *chip* FTDI USB *to Serial* (FT232), *chip* ini berkerja mengkonversi USB ke Serial dan banyak digunakan juga sebagai *chip* kabel *converter* USB ke *Serial Port*. Layaknya kabel *converter* anda juga terlebih dahulu harus menginstal *driver* dari *chip* FTDI ini.

3.3.3 Blok Arduino Uno

Ini adalah *capture* dari setingan kodingan sebelum *Board* Arduino Uno disambungkan ke komputer dengan menggunakan kabel FTDI USB .

Gambar 3.10 Program Arduino *Software*

4. Hasil Implementasi dan Pengujian

4.1 Analisis dan Implementasi

Pada bab ini akan dibahas analisa dan Implementasi terhadap *hardware* dan *software* dari sistem kendali lampu berbasis mikrokontoler. Pada bagian pengujian *hardware* akan dilakukan pengujian terhadap *Arduino Uno* dan lampu sebagai *output*. Sedangkan dari bagian *software* akan dilakukan pengujian terhadap komunikasi proses MFCC dan pengklasifikasian jaringan saraf tiruan *backpropagation.*

4.2 Pengujian *Hardware*

4.2.2 Pengujian Arduino

Tegangan masukan pada *board* arduino ketika beroperasi adalah 5v koneksi USB. Sebelum mengkoneksikan arduino uno ke komputer, yang harus dilakukan adalah :

- Instalasi *Software* Arduino
- FDTI kabel sebagai usb .

4.3 Pengujian *Software*

4.3.1 Penentuan Spesifikasi Parameter MFCC

Pengujian pada bagian ini adalah penentuan akurasi sistem metode MFCC dan *backpropagation* dalam mengenali perbedaan perintah dan dijalankan pada *visual studio* C++. Proses penentuan akurasi melibatkan *database* yang telah diambil menggunakan *Cool Edit Pro* 2.1 dengan frekuensi sampling 8000 Hz. Pengujian pada bagian ini bertujuan untuk mendapatkan parameter *input* MFCC yaitu jenis *window*, panjang *frame*, dan jumlah koefisien ciri yang optimal untuk dijadikan acuan pada simulasi sistem. Hasil data lengkap yang digunakan pada simulasi bagian ini ada pada lampiran. Pada perancangan di bab sebelumnya dijelaskan tentang proses ekstraksi suara menggunakan MFCC, untuk lebih jelasnya hasil dari masing-masing blok pada MFCC dapat dilihat pada penjelasan dibawah ini.

a. Sinyal suara yang telah direkam dipotong-potong selama 0,6 detik. Waktu selama 0,6 detik ini dianggap cukup untuk mengucapkan kata-kata perintah (‘*on*’ dan ‘*off*’) pada sistem. Pada penelitian ini digunakan sampling frekuensi sebanyak 8000 Hz, sehingga jumlah data pada proses perhitungan adalah 4800 *sample* untuk waktu sepanjang 0.6 detik.
b. Tahap *preprocessing* dilakukan untuk memperbaiki kualitas data sebelum dilakukan ekstraksi ciri. Tahap *preprocessing* yang digunakan meliputi

normalisasi amplitudo. Pada proses normalisasi amplitudo, nilai amplitudonya berada pada rentang -1 samapai dengan 1. Normalisasi amplitudo ini bertujuan agar proses berikutnya tidak dipengaruhi oleh perubahan amplitudo sehingga sinyal suara masukan memiliki interval amplitudo maksimum yang sama. Selanjtunya pada tahap *preprocessing* akan dilakukan proses *cropping*. Proses ini menggunakan nilai *threshold*, dimana nilai *threshold* yang digunakan adalah 0.02. Nilai amplitudo yang kurang dari *thresholding* akan dibuang. Besar nilai *thresholding* itu sendiri diperoleh dengan melihat karakteristik amplitudo sinyal tersebut.

c. Proses selanjutnya yang dibutuhkan adalah melakukan *filtering preemphasis* pada sinyal hasil *cropping*. Nilai yang digunakan untuk *preemphasis* ini adalah 0.95. Nilai ini didapatkan berdasarkan studi literatur yang telah dilakukan [1].
d. Pada proses *framing*, jumlah sample per*frame* yang digunakan pada proses *frame blocking* adalah 256. Berdasarkan percobaan yang telah dilakukan, jumlah sampel per*frame* yang digunakan yang memberikan hasil yang baik saat nilainya 256. Nilai diatasnya memberikan ciri yang terlalu sedikit, sedangkan nilai dibawah 256 memberikan nilai yang terlalu banyak, sehingga mempengaruhi performansi saat proses pelatihan. Sedangkan nilai *overlapping* yang digunakan adalah 0.3 dari nilai sampel per*frame*nya.
e. Untuk sampel suara '1mirza1.wav' yang diproses pada sistem ini akan menghasilkan *frame* sebanyak 32. Setiap *frame* menghasilkan 256 sampel data. Sehingga matriks yang dihasilkan pada proses ini adalah 256x32. Pada penentuan banyaknya sampel per *frame* yang digunakan, semakin besar jumlah sampel dalam satu *frame*, maka ciri yang dihasilkan pada hasil MFCC akan semakin sedikit. Sebaliknya, jika jumlah per*frame* nilainya sedikit, maka nilai ciri MFCC yang dihasilkan banyak.
f. Pada proses berikutnya proses *windowing* dengan menggunakan *hamming window*. Pemilihan *window* ini dilakukan karena *window* ini paling sering digunakan dalam aplikasi *speaker recognition*. Fungsi window ini menghasilkan *noise* yang tidak terlalu besar sehingga sangat dibutuhkan untuk aplikasi *speaker recognition*. [5]
g. Proses berikutnya adalah pengambilan spektrum dengan FFT. Pada sistem yang dirancang, nilai *point* FFT yang digunakan adalah kelipatan (pada tugas akhir ini digunakan 256 point FFT). Pada proses FFT, sinyal suara akan diproses pada domain frekuensi. Proses ini menghasilkan matriks dengan ukuran 256x32. Proses FFT ini dilakukan pada setiap *frame* yang ada.
h. Setelah sepktrum diambil, dilakukan proses *warping* dengan menggunakan *window triangular*. Berdasarkan dari frekuensi sampling yang digunakan adalah 8000Hz, maka nilai frekuensi bank yang digunakan adalah 24. Sedangkan pada saat frekuensi sampling yang digunakan 16000Hz, nilai frekuensi bank yang dapat digunakan adalah 32. Perkalian dengan menggunakan filter bank ini akan dilakukan untuk setiap *frame*. Hasil dari *filterbank warping* ini adalah matriks dengan ukutan 24 x 32 (24 adalah koefisien *filterbank*, sedangkan 32 adalah jumlah *frame*). Pada proses *warping* ini akan dihasilkan *mel-spektrum*. Prinsip dari proses ini mengadopsi prinsip pendengaran manusia. Dimana pada proses ini, nilai diatas 1000 Hz akan diproses secara linier, sedangkan diatas 1000 Hz diproses secara logaritmik.
i. Proses berikutnya yang dilakukan adalah pengembalian *mel spektrum* ke domain waktu dengan menggunakan DCT. Hasil akhir dari proses ekstraksi ciri ini berupa matriks dengan ukuran 12x32. Dimana 12 adalah jumlah koefisien MFCC yang digunakan, sedangkan 32 adalah jumlah *frame*. Koefisien MFCC yang digunakan untuk *speaker recognition* 12. Pada nilai ini, ciri yang akan dihasilkan oleh MFCC sudah mencukupi, sedangkan jika

penambahan jumlah koefisien yang dilakukan akan mempengaruhi komputasi. Sehingga pada sistem ini dilakukan pembatasan jumlah koefisien menjadi 12. Pada pemilihan koefisien ini juga, jika koefisien MFCC ditambah maka banyaknya nilai yang sama dengan nilai pada ciri sampel yang lainnya. Sehingga dapat mempengaruhi keakurasian sistem. Pada studi literatur yang dilakukan, jumlah koefisien MFCC yang biasanya digunakan untuk proses speech recognition adalah kurang dari 14.[2]

4.3.2 Normalisasi

Normalisasi dilakukan untuk mendapatkan nilai normal suara dengan maksimum amplitudo satu dan minimum minus satu. Dilakukan dengan membagi nilai-nilai koefisein MFCC dengan nilai yang paling maksimum dan telah diabsolutkan.

4.3.3 Klasifikasi

Pada dasarnya, klasifikasi suara dengan menggunakan *backpropagation* adalah suatu proses dimana sistem dilatih untuk mengenali ciri yang ada sehingga nantinya dapat digunakan untuk mengenali masukan yang mirip dengan ciri yang sudah dilatih sebelumnya. Tetapi sebelumnya harus dilakukan proses penelitian dengan parameter yang benar, sehingga nantinya sistem dapat mengenali ciri yang ada dengan baik.

Berdasarkan hasil pengujian yang telah dilakukan, jaringan saraf tiruan *backpropagation* dengan spesifikasi sebagai berikut :

1. Nilai jumlah *hidden layer* = 5 *hidden layer*
2. Nilai konstanta r = 250
3. Nilai *neuron hidden layer* = 13,30,25,10,1
4. Nilai *learning rate* awal ( ) = 0.1
5. Nilai *error* target = 0.0001
6. Iterasi = 1000

4.3.3.1 Performansi Sistem

Percobaan ini merupakan parameter utama yang mengukur keberhasilan dari perancangan sistem. Pada percobaan ini hanya akan dilakukan pengujian sistem *non-realtime*

Sistem *Non-realtime*

Pada pengujian ini data yang akan diujikan sebanyak 150 data. Dimana data tersebut merupakan data 5 responden yang telah dilakukan proses pengklasifikasian sebelumnya.

Pada pengujian ini jumlah sample per *frame* 256. Hasil yang didapatkan dari pengujian ini dengan nilai akurasi 80.23% (dapat dilihat pada lampiran B).

5. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan implementasi dan pengujian yang telah dilakukan seperti pada bab sebelumnya, maka dapat ditarik beberapa kesimpulan sebagai berikut:

1. Performa Arduino Uno dalam sistem ini cukup baik, sehingga sistem dapat berjalan dengan lancar. Baik dalam pengolahan *input* maupun pengolahan *output* yang dihasilkan dari *input* sistem.
2. Akurasi yang dapat dicapai pada sistem *non-realtime* dalam mengenali suara yang ada didalam *database* adalah 80.23%.
3. Parameter-parameter MFCC yang terbaik didapatkan, sehingga diperoleh ciri yang baik saat koefisien masuk pada proses pengklasifikasian jst *bakcpropagation*.
4. Parameter-parameter terbaik jst *backpropagation* di dapatkan dari berbagai uji coba dan di catat parameter yang terbaik.

5.2 Saran

1.Adanya metoda yang digunakan untuk mereduksi *noise*, sehingga sistem ini akan lebih baik dalam mengenali individu terutama untuk *realtime* sistem dan menggunakan metode ekstraksi ciri lain seperti wavelet, LPC, dan lainnya.

2.Menggunakan metode adaptive backpropagation yang lainnya seperti *delta-bar-delta*, *adaptive learning rate by sign change*, RPROP. Atau menggunakan JST yang lain seperti ART-2, SOM kohonen.

3.Dapat diimplementasikan dengan menggunakan bahasa pemograman lain seperti C#, Java, dan lainnya.

4.Harus ditambahkan pengujian *real-time*, agar mengetahui perbandingan ketahanan sistem.

DAFTAR PUSTAKA

[1]. Beigi, Homayoon. (2011). "*Fundamental of Speaker Recognition".* Springer.

[2]. B. Plannerer. (2005). "*An Introduction to speech recognition".*

[3]. Mustofa, Ali. 2007. *Sistem Pengenalan Penutur dengan Metode Mel-frekuency Wrapping*. Jurnal Teknik Elektro". Vol. 7, No. 2, p. 88 – 96.

[4] Putra, Darma (2008). "*Sistem Biometrika".* Yogyakarta : ANDI Yogyakarta.

[5]. Putra, Darma. (2011). "*Verifikasi Biometrika Suara Menggunakan Metode MFCC dan DTW*", Universitas Udayana.

[6]. Rakhman, Arif. (2009). "*Analisis Pengaturan Perubahan Learning Rate Pada Jaringan Syaraf Tiruan (JST) Backpropagation Menggunakan Metode Delta-Bar-Delta*" Institut Teknologi Telkom: Tidak diterbitkan

[7]. Satrya, Ryan. (2010). "*Sistem Identifikasi Suara Pria dan Suara Wanita Berdasarkan Usia Menggunakan Mel Frequency Cepstral Coefficient dan K-Mens Clustering*", Institut Teknologi Telkom.

[8]. Kadir, Abdul. (2013). " *Panduan Praktis Mempelajari Aplikasi Mikrokontroler dan Pemrogamannya Menggunakan Arduino*". Andi Publisher