BAB III
METODE PENELITIAN

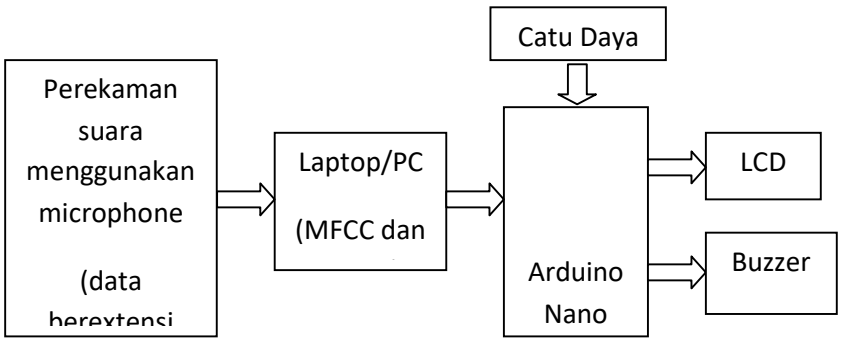
3.1 Studi Literatur

Sebelum melangkah ke dalam pengerjaan sistem dari Tugas Akhir ini hal yang perlu dilakukan adalah observasi, karena dengan melakukan observasi terlebih dahulu dapat memudahkan apa saja yang harus dikerjakan dalam membuat Tugas Akhir. Observasi tersebut berupa studi literatur yang dilakukan untuk menguatkan gagasan, pemahaman konsep, teori dan teknologi yang akan digunakan dalam pengerjaan Tugas Akhir. Dalam studi literatur hal – hal yang dilakukan antara lain mempelajari dan memahami berbagai referensi dari buku, Internet, jurnal atau paper, kondisi nyata di masyarakat tentang semua yang berhubungan dengan rumusan masalah yang telah ditentukan dalam Tugas Akhir ini.

3.2 Perancangan Sistem

Perancangan sistem Klasifikasi Huruf Vokal dengan Pengolahan Audio Untuk Sistem Notifikasi Ruang Perawat dengan Metode *Mel-Frequency Cepstrum Coefficient* dan *Backpropagation-Neural Network* menggunakan sistem kendali Arduino Uno dibagi menjadi 2 bagian yaitu perancangan perangkat keras (h*ardware)* dan perancangan perangkat lunak (*software)*. Perangkat keras terdiri dari modul *Arduino Uno*, Microphone sebagai alat untuk merekam suara, laptop sebagai prosesing perekaman suara, LCD sebagai penampil notifikasi dan Buzzer sebagai notifikasi suara. Sedangkan perangkat lunak mengunakan aplikasi MATLAB, yang berfungsi sebagai aplikasi programing dari Arduino Uno supaya sistem ini bisa mengenali perintah suara sesuai dengan batasan huruf vokal.

Gambar perancangan sistem *hardware* ditunjukkan pada Gambar 3.1



Gambar 3.1 Diagram Blok Sistem Mikrokontrol

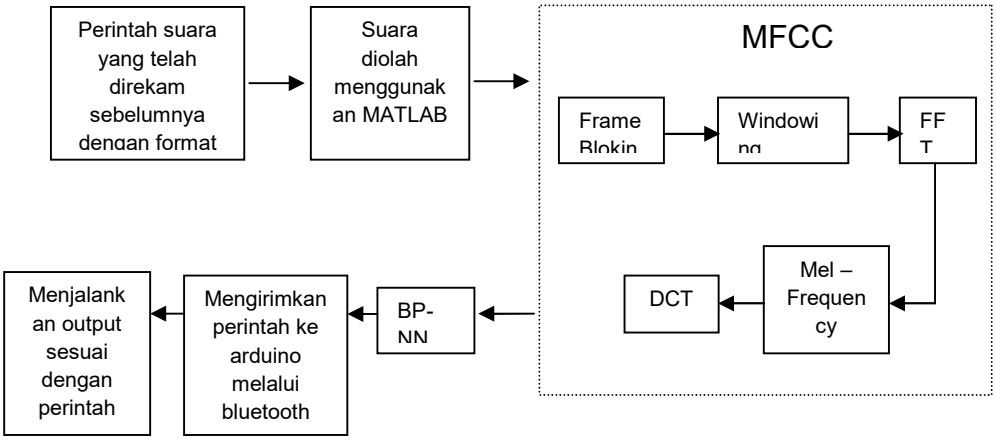
Dibawah ini menjelaskan tentang perancangan kerja hardware. Dimana pada hardware dalam tugas akhir meliputi:

1. Laptop sebagai prosesing gelombang suara yang di rekam
2. Microphone sebagai alat untuk merekam suara
3. Catu daya 3.3 VDC dan 5 VDC digunakan sebagai sumber bagi mikrokontroller, serta pendukung lainnya.
4. LCD dan Buzzer sebagai penampil notifikasi Tulisan dan Suara
5. Prototipe alat

Rancangan kerja hardware menjelaskan proses keseluruhan dari alat yang akan dibuat dalam tugas akhir ini. Alat ini menggunakan mikrofon untuk mendapatkan gelombang suara yang berikutnya akan diproses melalui ekstraksi metode MFCC untuk pengenalan pola suara dan setelah hasil MFCC diketahui maka akan menjadi input bagi BP-NN. Setelah BP-NN mengenali perintah maka hasilnya akan dikirimkan ke mikrokontroler melalui wireless untuk menjalankan perintah tersebut.

3.2.1 Perancangan Sistem Pengenalan Perintah Suara

Perancangan sistem pengenalan perintah suara dapat ditunjukkan pada gambar 3.2 :



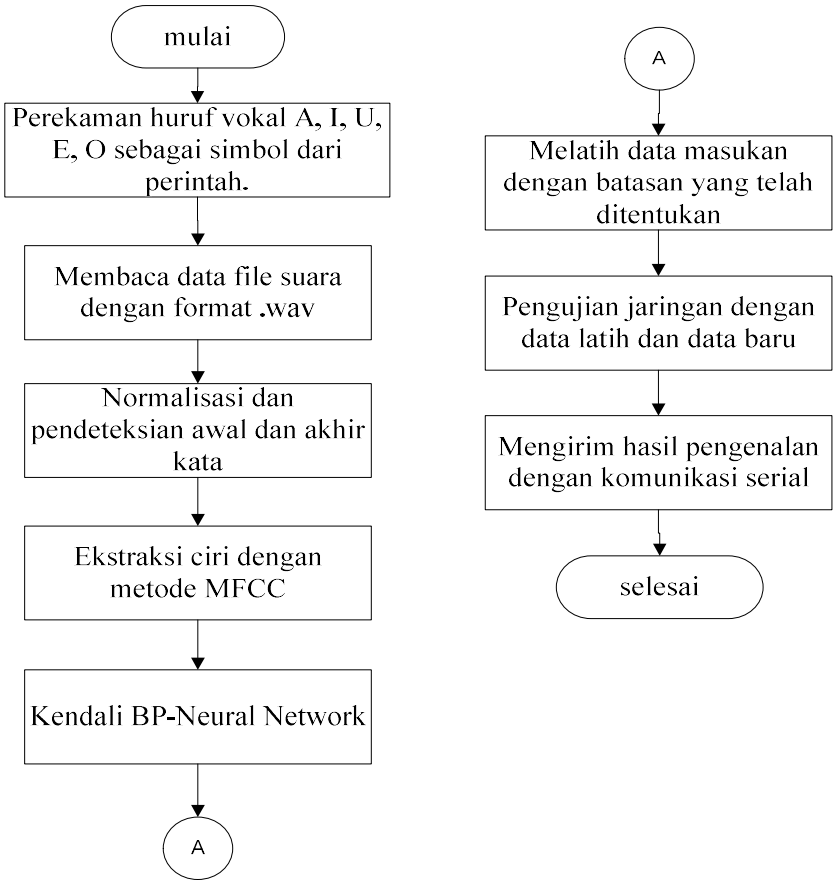
Gambar 3.2 Perancangan sistem pengenalan perintah suara

Pada Gambar 3.2 menjelaskan konsep awal dalam perancangan sistem pengenalan suara untuk sistem notifikasi ruang perawat. Pada awalnya sistem ini menggunakan mikrofon yang dihubungkan pada PC

untuk mendapatkan sinyal suara dengan format wav. Namun karena penulis tidak mendapatkan hasil perekaman suara yang bagus secara real time akhirnya sistem ini diubah dengan cara membaca perintah suara dengan format .wav yang telah direkam terlebih dahulu. Sinyal suara berupa wav tersebut akan diproses oleh software MATLAB menggunakan metode MFCC, metode MFCC ini terdiri dari beberapa proses. Dari proses MFCC ini akan menghasilkan sebuah output yang akan diproses dengan metode BP-NN sehingga mendapatkan keluaran yang bisa dibaca oleh arduino untuk menjalankan perintah berdasarkan huruf vocal yang diucapkan oleh user.

3.2.2 Perancangan Flow chart kerja sistem

Prinsip kerja sistem dalam tugas akhir ini ditunjukkan pada Gambar 3.3 yaitu flowchart kerja sistem.



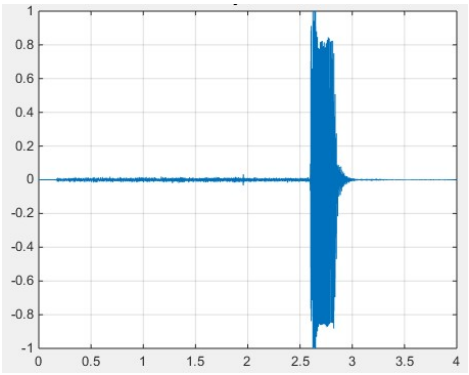
Gambar 3.3 *Flowchart* kerja sistem

Dari flowchart diatas, prinsip kerja sistem pada tugas akhir ini adalah :

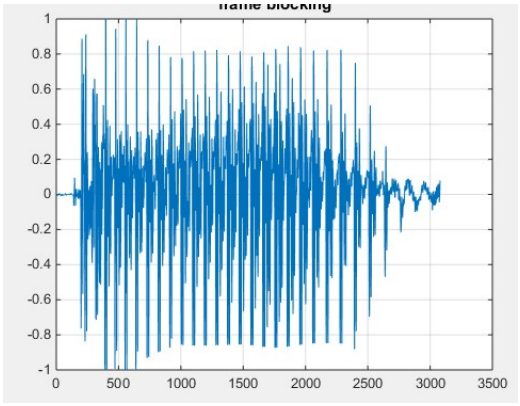
Pada kondisi awal dilakukan perekaman suara dengan huruf vokal A, I, U, E, O sebagai simbol masing-masing perintah. Data yang telah didapat kemudian dibaca dengan format .wav yang kemudian akan dilakukan normalisasi dan pembacaan awal akhir kata.Deteksi awal-akhir digunakan pada proses untuk mendeteksi mulainya sinyal ucapan awal dan berakhir ketika sudah tidak diucapkan. Setelah itu akan dilakukan pengekstraksian ciri menggunakan *metode MFCC (Mel-Frequency Cepstrum Coefficient)*, proses ini bertujuan untuk mendapatkan parameter-parameter dari sinyal suara. Parameter inilah yang nantinya akan membedakan karakteristik dari masing masing suara. Setelah nilai MFCC didapatkn kemudian dijadikan masukan bagi *Back PropagationNeural Network* dan akan diproses sehingga mendapatkan error paling kecil. Setelah data yang diproses oleh BP-NN terkenali, maka sistem akan mengeluarkan output yang akan menampilkan perintah pada LCD dan buzzer. Dengan demikian sistem selesai dalam melakukan proses pengenalan perintah suara.

3.2.3. Pemrosesan suara menggunakan metode MFCC

Berikut merupakan conton pemrosesan suara dengan menggunakan kata “A”. Sesuai dengan Tahapan MFCC sinyal suara terlebih dahulu direkam menggunakan mikrophone. Sinyal suara direkam selama 4 detik dengan fs sebesar 8000Hz. Sinyal suara yang telah direkam, kemudian ditampilakan dikomputer, seperti pada Gambar 3.4 Sinyal suara yang direkam kemudian dipotong untuk dideteksi bagian suara saja. *Silencedetection* digunakan pada bagian ini. Gambar 3.5 merupakan potongan suara kata “A”.

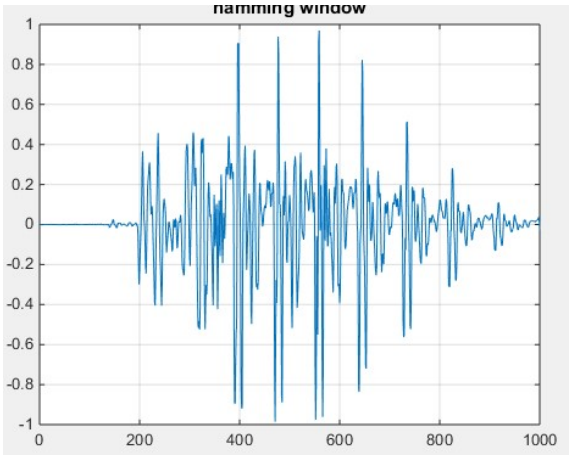


Gambar 3.4 Sinyal suara A



Gambar 3.5 Silence Detection

Tahap selanjutnya adalah *hamming window* atau yang sering disebut dengan *windowing*. *Windowing* digunakan sebagai bentuk jendela dengan mempertimbangkan blok atau kata berikutnya dalam fitur rantai pengolahan ekstraksi dan mengintegrasikan semua lini frekuensi yang paling dekat. Contoh dari proses *windowing* ada pada Gambar 3.6.



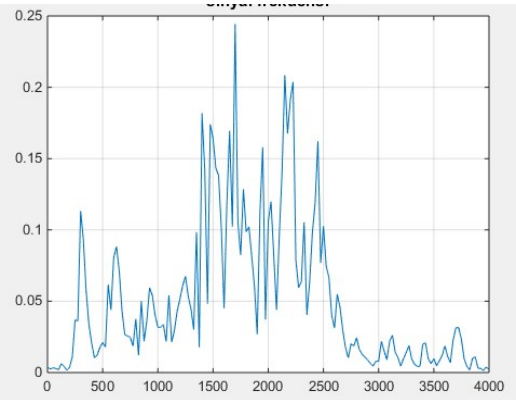
Gambar 3.6 Windowing

$$Y\,[n] = X\,(n).W\,(n) \qquad 0 \le n \le N-1 \tag{3.1}$$

$$W\,(n) = 0{,}54 - 0{,}46\cos\frac{2\pi n}{N-1} \tag{3.2}$$

Setelah itu dilakukan proses *Fast Fourier Transform (FFT)* digunakan untuk mengkonversi sinyal dari domain waktu ke domain frekuensi. FFT dapat

mengetahui besarnya respon frekuensi setiap frame. FFT ditunjukan pada Gambar 3.7.



Gambar 3.7 FFT

Fast Fourier Transformation (FFT) pada dasarnya sama dengan algoritma *Discreet Fourier Transformation* (DFT), namun FFT menggunakan teknik perhitungan yang lebih cepat. Persamaan 3.3 merupakan dasar DFT.

$$F(k) = \sum_{n=0}^{N-1} f(n) W_N^{k.n}, 0 \leq k \leq N-1 \quad (3.3)$$

Dimana $Wn = e^{j\frac{2\pi}{N}}$

Hasil perhitungan DFT berjumlah N^2, contoh terdapat 4 data maka dibutuhkan 16 kali perhitungan. Agar lebih mudah dipahami perhitungan tersebut dibentuk kedalam sebuah matriks seperti Persamaan 3.4

$$\begin{bmatrix} F\,[0] \\ F\,[1] \\ F\,[2] \\ F\,[3] \end{bmatrix} = \begin{bmatrix} W_N^{0.0} & W_N^{0.1} & W_N^{0.2} & W_N^{0.3} \\ W_N^{1.0} & W_N^{1.1} & W_N^{1.2} & W_N^{1.3} \\ W_N^{2.0} & W_N^{2.1} & W_N^{2.2} & W_N^{2.3} \\ W_N^{3.0} & W_N^{3.1} & W_N^{3.2} & W_N^{3.3} \end{bmatrix} . \begin{bmatrix} f\,[0] \\ f\,[1] \\ f\,[2] \\ f\,[3] \end{bmatrix} \quad (3.4)$$

FFT mempercepat perhitungan DFT dengan setengah kali perhitungan DFT. DFT membutuhkan n^2 kali perhitungan, namun FFT hanya memerlukan (n/2 + 1) x n + n/2 kali perhitungan (Adhitya, et al., 2016). Misalkan jumlah data n = 50 dengan menggunakan FFT cukup dilakukan ((26x50)+25) = 1.325 kali perhitungan, sedangkan jika menggunakan DFT dibutuhkan sejumlah 2.500 kali perhitungan.

Perhitungan FFT dapat dilihat pada Persamaan 3.5 dibawah ini

$$F(k) = \sum_{n=0}^{N-1} f(n) \cos\left(\frac{2\pi nk}{N}\right) - j \sum_{n=0}^{N-1} f(n) \sin\left(\frac{2\pi nkT}{N}\right) \quad (3.5)$$

Perhitungan DFT dapat disederhanakan dengan cukup menghitung setengah periode saja, setengah periode berikutnya dapat dihitung dengan menggunakan Persamaan 3.6 (Adhitya, et al., 2016):

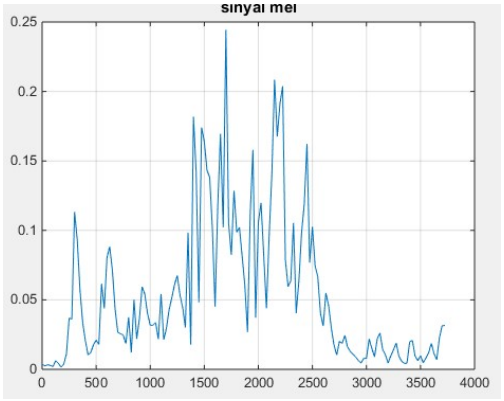
$$F = x + \left(\frac{T}{2}\right) = Real\left\{F\left(\frac{T}{2} - x\right)\right\} - j\, Im\{F(x)\} \quad (3.6)$$

Langkah berikutnya setelah didapatkan data FFT maka dilakukan perhitungan *Mel-Frequency.* Untuk setiap nada dengan frekuensi f, diukur dalam Hz, *pitch* subjektif diukur pada skala yang disebut "mel" skala. Skala mel frekuensi adalah frekuensi linier berada dibawah 1000 Hz dan logaritmik diatas 1000Hz (Muldayani, Purwanto, & Arief, 2015) .

Untuk menghitung mels untuk f frekuensi dalam Hz, dapat menggunakan rumus Persamaan 3.7 berikut:

$$F(mel) = 1127\, log\left[1 + \frac{f}{700}\right] \quad (3.7)$$

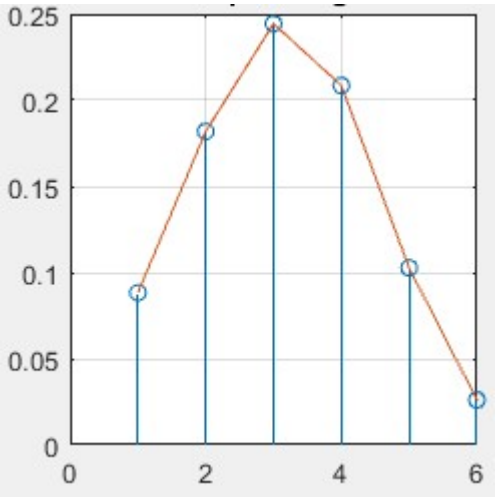
Gambar 3.8 merupakan Mel-Frequency Warping



Gambar 3.8 Mel-Frequency Warping

langkah terakhir ini, spektrum log mel seharusnya dirubah kembali ke domain waktu menggunakan *Discrete cosine transform* (DCT), namun karena tidak mendapatkan hasil yang baik untuk input *Neural Network* maka dilakukan pengambilan magnitude yang memiliki nilai dominan diantara magnitude lainnya.Hasilnya disebut bandpass signal. Pada Gambar 3.9 ini

merupakan hasil dari pola bandpass signal yang digunakan sebagai input *Neural Network* yang selanjutnya disebut sebagai pola MFCC.



Gambar 3.9 pola MFCC

MFCC yang dihasilkan dari sinyal suara "A". setiap sinyal suara akan menghasilkan pola MFCC yang berbeda-beda. Pola yang dihasilkan terdiri dari 6 sampel atau element setiap huruf. Pola dari MFCC ini disimpan yang nantinya digunakan sebagai inputan dari NN.

Berikut adalah *Source code* dari pemrosesan sinyal suara dengan metode MFCC sekaligus digunakan untuk membuat data *training* untuk input *Neural Network.*

```
clc;
fs = 8000;
figure(1)
filewav = 420; % masukkan nama file .wav disini
urutan = 100; % urutan data untuk keperluan training
% kalau sudah selesai mengurutkan pada command window
% ketikkan save('25','datatrain')
cek = [num2str(filewav) '.wav'];
[y,Fs] = audioread(cek);
sound(y,Fs);
t = 1/fs: 1/fs: length(y)/fs ;
subplot(231)
plot(y);
grid on;
title('sinyal asli')
```

```
a1 = detectVoiced(y,1);
segmen = size(a1)
ukuran = segmen(1,2);

if ukuran == 1
a = [a1{1}];
elseif ukuran == 2
a = [a1{1};a1{2}];
elseif ukuran > 2
a = [a1{ukuran-1};a1{ukuran}];
end

subplot(232)
plot(a);
grid on;
title('frame blocking')
w = a(1:1000).*hamming(1000);

subplot(233)
plot(w);
grid on;
title('hamming window')

T = 1/fs;
L = 320;
sip = fft(w);
P2 = abs(sip/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = fs*(0:(L/2))/L;

subplot(234)
plot(f,P1) ;
grid on;
title('sinyal frekuensi')

fmel = 1127*log(1+f/700);
```

```
subplot(235)
plot(f(1:150),P1(1:150));
grid on;
title('sinyal mel')

high1 = max(P1(21:41,1));
high2 = max(P1(41:61,1));
high3 = max(P1(61:81,1));
high4 = max(P1(71:101,1));
high5 = max(P1(101:121,1));
high6 = max(P1(121:141,1));

simpan = [high1 high2 high3 high4 high5 high6];
datatrain(:,urutan) = simpan'

subplot(236)
stem(simpan);grid on;hold on;
plot(simpan);
grid on;
title('bandpass signal');
```

Berikut adalah data nilai yang didapatkan dari proses ekstraksi dengan metode MFCC yang akan digunakan sebagai data *training* untuk input jaringan BP-NN

Tabel 3.1 Nilai data *training*

A1	**A2**	**A3**	**A4**	**A5**
0,088323	0,087066	0,217916	0,128752	0,127721
0,181863	0,210328	0,192037	0,227764	0,105367
0,244354	0,543065	0,139679	0,206449	0,377077
0,208492	0,330144	0,116501	0,266799	0,324406
0,102674	0,086901	0,029322	0,035432	0,125431
0,026119	0,035652	0,014024	0,014561	0,023506
I1	**I2**	**I3**	**I4**	**I5**
0,2376	0,080432	0,114462	0,163483	0,845004
0,066525	0,029679	0,040551	0,057875	0,03406

0,039588	0,022238	0,021951	0,039538	0,022872
0,02272	0,012928	0,021951	0,022796	0,009867
0,005976	0,008941	0,007361	0,019836	0,009008
0,014743	0,007035	0,011125	0,035382	0,023714
U1	**U2**	**U3**	**U4**	**U5**
0,395628	0,521852	0,293472	0,905905	0,245638
0,062112	0,189451	0,057215	0,221398	0,045036
0,043204	0,064464	0,122188	0,018171	0,013212
0,017362	0,023757	0,058201	0,010396	0,007975
0,002874	0,006718	0,005321	0,002171	0,002238
0,003197	0,003548	0,003359	0,001551	0,001221
E1	**E2**	**E3**	**E4**	**E5**
0,493404	0,356536	0,467838	0,568212	0,466841
0,155973	0,060531	0,0745	0,116067	0,062811
0,042702	0,018635	0,010798	0,020603	0,011039
0,042702	0,012484	0,011004	0,015431	0,01217
0,03106	0,0111	0,007435	0,017512	0,016633
0,076631	0,04958	0,009464	0,048727	0,166779
O1	**O2**	**O3**	**O4**	**O5**
0,631153	0,370088	0,57378	0,497584	0,427115
0,216038	0,158736	0,57378	0,497584	0,104
0,196646	0,06431	0,276969	0,112645	0,115356
0,040994	0,008826	0,01408	0,02979	0,043337
0,003589	0,002668	0,001803	0,00299	0,001355
0,003749	0,0037	0,002584	0,001741	0,00148

Dengan data target yang digunakan sebagai data target jaringan BPNN sebagai berikut

Tabel 3.2 Data Target

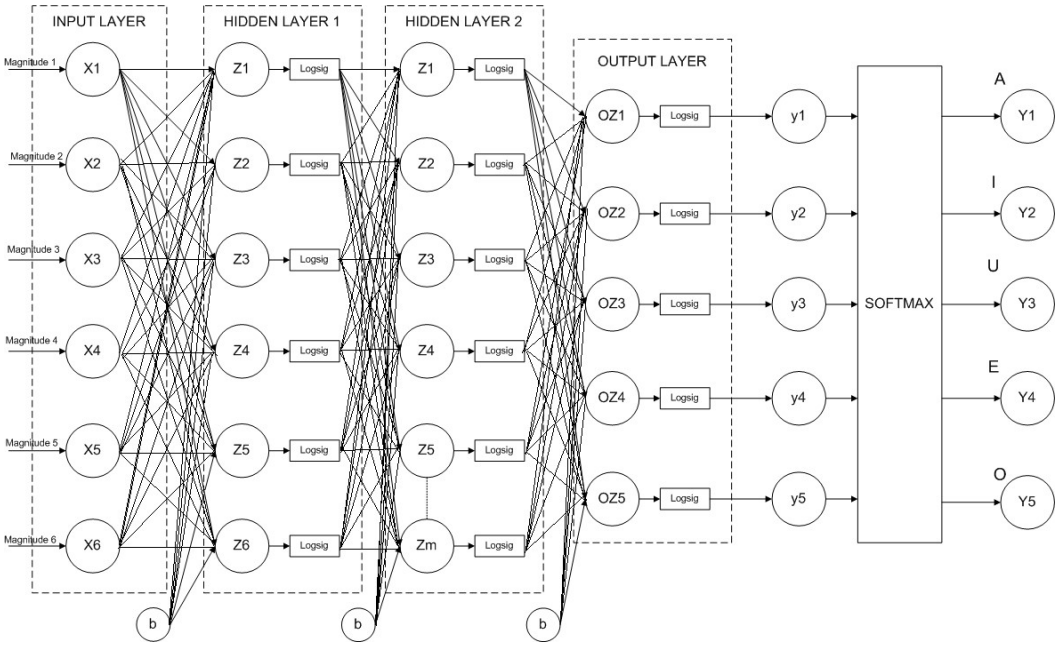
A	**I**	**U**	**E**	**O**
1	0	0	0	0
1	0	0	0	0

1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
0	1	0	0	0
0	1	0	0	0
0	1	0	0	0
0	1	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1

Sedangkan untuk data uji ditentukan sebanyak 20 % dari jumlah data *training* yaitu jika data *training* sebanyak 25 data, maka untuk data uji adalah sebanyak 5 data atau masing – masing perintah suara adalah 1 data (suara “A” 1 data, suara “I” 1 data, suara “U” 1 data, suara “E” 1 data, suara “O” 1 data) .

3.2.4. Pemodelan Back Propagation Neural Network

Pemodelan *Back Propagation Neural Network* dengan softmax function ditunjukkan oleh Gambar 3.10 dibawah ini.



Gambar 3.10. Pemodelan *Back Propagation Neural Network*

Hasil dari MFCC berupa data dalam bentuk domain waktu selanjutnya dijadikan *input* untuk *training* BPNN. Jumlah vektor *input* tergantung dari hasil percobaan atau jumlah sampel yang didapat dari MFCC. Pada Gambar 3.10 vektor *input* yang disimbolkan dengan X1-X6 berisi sampel hasil MFCC, jika sampel berjumlah 6 maka vektor *input* adalah X1-X6. *Hidden layer* 1 sebanyak 6 neuron disimbolkan dengan Z1-Z6, *hidden layer 2* sebanyak 17 neuron disimbolkan Z1-Zm, dan OZ1-OZ5 merupakan *output* dari *hidden layer* 2 yang kemudian diproses kembali untuk mendapatkan *output* yang disimbolkan y1-y5. Denormalisasi dilakukan untuk mengembalikan data sesuai dengan *input* sebelum dilakukan normalisasi. Fungsi yang digunakan adalah fungsi *softmax*, dimana *softmax* ini berfungsi mencari nilai terbesar dari hasil *output* prediksi untuk menentukan vektor *output*. Dalam permasalahan klasifikasi, *softmax* sangat baik untuk mendapatkan peluang nilai yang tinggi dalam memprediksi kelas target. *Range* nilai yang dihasilkan *softmax* adalah 0 sampai 1, *range* tersebut akan mewakili tiap-tiap output. Hasil dari *output* NN akan diproses dengan fungsi *softmax* sehingga mendapatkan *output* baru. Vektor *output* pada BP-NN ini berjumlah 5

disimbolkan dengan Y1-Y5 yang mewakili dari perintah A, I, U, E, dan O. *Output* inilah yang nantinya akan terbaca oleh kontroler untuk mengambil keputusan dan menampilkan notifikasi pada layar lcd sesuai dengan output yang telah ditentukan.

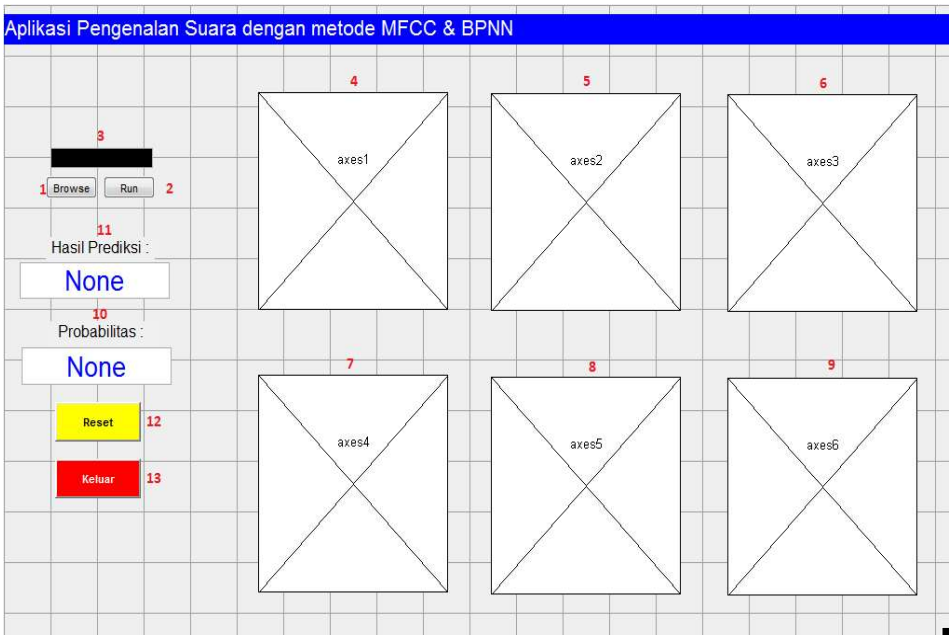
Berikut adalah *Source code* yang digunakan untuk training jaringan *Back Propagation-Neural Network*

```
clc;
a = load('25'); %load nama file data input
b = a.datatrain; %nama variabel penyimpanan data input
c = load('25t'); %load nama file data target
d = c.target; %nama variabel penyimpanan data target

INPUT = b;
OUTPUT = d';

% Batasan error sebagai target trainning
    error_max=1e-5;

% Membentuk jaringan NN
    net=newff(minmax(INPUT),        [6        17        5],
{'logsig','logsig','logsig'},'trainlm')
% trainlm,traingd,traingda,traingdm, traingdx

% Define parameters
    net.trainParam.epochs = 100;
    net.trainParam.goal = error_max;
    net.trainParam.max_fail = 6000;

%Train network
    netWr = train(net, INPUT, OUTPUT);

% Simulate result
    Wr= sim(netWr,INPUT);
    prediksi = Wr';
    hasil = [OUTPUT' prediksi]

% plot target vs prediksi
```

```
% ke=1;plot(hasil(:,ke));hold on;stem(hasil(:,100+ke));grid
on;
save ceksj
```

3.2.5. Design Interface

Desain tampilan yang dibuat untuk Tugas Akhir ini dikerjakan menggunakan GUI *(Graphic User Interface)* MATLAB. Gambar 3.11 merupakan tampilan yang digunakan pada Tugas Akhir ini.



Gambar 3.11. Desain tampilan tugas akhir

Terdapat beberapa bagian yang digunakan untuk mempermudah akses antara *user* dengan aktuator yang digunakan. Beberapa fungsi bagian-bagian tersebut antara lain:

1. Tombol browse berfungsi untuk melakukan load data suara yang teah direkam.
2. Tombol *run* berfungsi untuk menjalankan program.
3. Kolom data berfungsi menampilkan data yang sudah dipilih untuk diproses.
4. Grafik *origin signal* menunjukkan hasil grafik perekaman suara.
5. Grafik *Frame Blocking* menunjukkan grafik hasil *frame blocking*

sinyal suara.

6. *Grafik windowing* menunjukkan grafik hasil *hamming window* sinyal suara
7. Gafik FFT merupakan grafik hasil dari proses perubahan fungsi dari waktu ke fungsi frekuensi.
8. Grafik *mel-frequency* merupakan grafik hasil skala mel.
9. Grafik Bandpass signal merupakan grafik hasil pengolahan data yang berbentuk fungsi frekuensi yang dilakukan pengambilan data magnitude yang dominan.
10. Kolom hasil dari klasifikasi BP-NN.
11. Kolom hasil perintah yang terdeteksi.
12. Tombol Reset untuk mereset ulang pogram ketika akan melakukan proses data baru
13. Tombol Keluar untuk keluar dari aplikasi

Berikut adalah *Source code* program utama yang digunakan aplikasi pengenalan suara dengan metode MFCC dan BPNN

```
function varargout = Utama(varargin)
% UTAMA MATLAB code for Utama.fig
%      UTAMA, by itself, creates a new UTAMA or raises the
existing
%      singleton*.
%
%      H = UTAMA returns the handle to a new UTAMA or the
handle to
%      the existing singleton*.
%
%           UTAMA('CALLBACK',hObject,eventData,handles,...)
calls the local
%       function named CALLBACK in UTAMA.M with the given
input arguments.
%
%      UTAMA('Property','Value',...) creates a new UTAMA or
raises the
```

```
%         existing singleton*.   Starting from the left,
property value pairs are
%        applied to the GUI before Utama_OpeningFcn gets
called.  An
%       unrecognized property name or invalid value makes
property application
%      stop.  All inputs are passed to Utama_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Utama

% Last Modified by GUIDE v2.5 03-Jun-2019 01:58:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Utama_OpeningFcn, ...
                   'gui_OutputFcn',  @Utama_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}]     =     gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Utama is made visible.
function Utama_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to Utama (see VARARGIN)

% Choose default command line output for Utama
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Utama wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the
command line.
function varargout = Utama_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
 function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles      structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as
text
%        str2double(get(hObject,'String')) returns contents
of edit1 as a double

 % --- Executes during object creation, after setting all
properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles     empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if    ispc    &&    isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

 % --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles      structure with handles and user data (see
GUIDATA)
global filename1
[filename1,pathname] = uigetfile('*.wav');
file = fullfile(pathname,filename1);
```

```
parseng = strsplit(filename1,'.');
tangkap = char(parseng(1,1));
set(handles.edit1,'string',filename1);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
global filename1
load ceksj
fs = 8000;
ke = 50;
[y,Fs] = audioread(filename1);
sound(y,Fs);
t = 1/fs: 1/fs: length(y)/fs ;
a1 = detectVoiced(y,1);
segmen = size(a1);
ukuran = segmen(1,2);
if ukuran == 1
a = [a1{1}];
elseif ukuran == 2
a = [a1{1};a1{2}];
elseif ukuran > 2
a = [a1{ukuran-1};a1{ukuran}];
end

w = a(1:1000).*hamming(1000);
T = 1/fs;
L = 320;
sip = fft(w);
P2 = abs(sip/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = fs*(0:(L/2))/L;
fmel = 1127*log(1+f/700);
```

```
high1 = max(P1(21:41,1));
high2 = max(P1(41:61,1));
high3 = max(P1(61:81,1));
high4 = max(P1(71:101,1));
high5 = max(P1(101:121,1));
high6 = max(P1(121:141,1));
simpan = [high1 high2 high3 high4 high5 high6];

axes(handles.axes1);
plot(y);
grid on;
title('origin signal')

axes(handles.axes2);
plot(a);
grid on;
title('frame blocking')

axes(handles.axes3);
plot(w);
grid on;
title('hamming window')

axes(handles.axes4);
plot(f,P1) ;
grid on;
title('fft signal')

axes(handles.axes5);
plot(f(1:150),P1(1:150));
grid on;
title('mel signal')

axes(handles.axes6);
stem(simpan);grid on;hold on;
plot(simpan);
grid on;
title('bandpass signal');
```

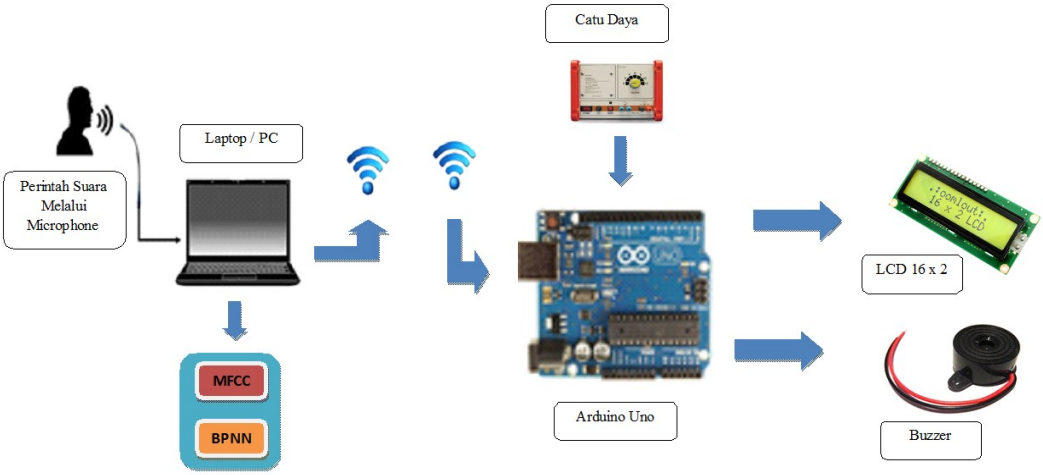
```
simpan = [high1 high2 high3 high4 high5 high6];
INPUT = simpan'
Wr= sim(netWr,INPUT);
prediksi = Wr';
probabilitas = max(prediksi);
[maxNumCol, maxIndexCol] = max(prediksi);
[maxNum, col] = max(maxNumCol);
row = maxIndexCol(col);
switch row
case 1
        tampil = 'a'
        jalan = '*%a#';
case 2
        tampil = 'i'
        jalan = '*%i#';
case 3
        tampil = 'u'
        jalan = '*%u#';
case 4
        tampil ='e'
        jalan = '*%e#';
case 5
        tampil ='o'
        jalan = '*%o#';
end
set(handles.edit2,'string',tampil);
set(handles.edit3,'string',probabilitas);

ser=serial('COM4','BAUD', 9600); % Make sure the baud rate
and COM port is
fopen(ser);
pause (1);                                  %the number of
executions
fprintf(ser,jalan);                %This command will send
entered value to Arduino
pause (1);
fclose(ser);
```

```
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
axes(handles.axes1);
cla;
axes(handles.axes2);
cla;
axes(handles.axes3);
cla;
axes(handles.axes4);
cla;
axes(handles.axes5);
cla;
axes(handles.axes6);
cla;
set(handles.edit2,'string','');
set(handles.edit3,'string','');

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
close;

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit2 as
text
%        str2double(get(hObject,'String')) returns contents
of edit2 as a double

% --- Executes during object creation, after setting all
properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if    ispc    &&    isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles      structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as
text
%        str2double(get(hObject,'String')) returns contents
of edit3 as a double

% --- Executes during object creation, after setting all
properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version
of MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if    ispc    &&    isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

3.3. Perancangan dan pembuatan hardware



Gambar 3.12. Rancang Kerja Hardware

Gambar 3.12 menjelaskan rancangan kerja *hardware* . Proses keseluruhan dari alat yang akan dibuat dalam Tugas Akhir ini. Alat ini menggunakan mikrofon untuk mendapatkan gelombang suara yang berikutnya akan diproses melalui ekstraksi metode MFCC untuk pengenalan pola suara dan setelah hasil MFCC diketahui maka akan menjadi input bagi BP-NN. Setelah BP-NN mengenali perintah maka hasilnya akan dikirimkan ke *hardware* melalui *bluetooth* untuk menjalankan perintah tersebut.