DTMF ENCODER-DECODER

Group Number-4

Submitted to faculty: ASHOK RANADE

Table of Contents

1. Synopsis:

In the modern world automation is an important part of life. Home automation is one of those. Prosaic, house switches to operate distinct appliances are located far off, making it arduous to turn on/off the load. For the most part, it becomes more gruelling for the elderly and impediments to operate the appliances as per their requirements. Furthermore, home automation ensures to overcome the above issues providing higher security and saves a worthwhile time. Home automation systems with the use of DTMF are one of those. How and why use DTMF for home automation? The project makes an effort to answer the above question. The system is established by setting up ARDUINO UNO along with RELAY MODULE which receives digital commands from DTMF signals provided from the mobile keypad. Unlike any other type of control that uses radio frequencies or infrared, DTMF uses sound signals. Owing to the same the system provides flexibility to operate home appliances from any part of the world giving a new face to home automation.

2. Process:

To on/off the desired appliances, the corresponding number of the dial pad of the user mobile device is pressed which is then received by the home mobile device via call, the home mobile device is a part of the home automation circuit installed in the house. The received dualtone is transmitted to a decoder which decodes the frequency into a 4-bit binary and sends it to the MCU Arduino Uno. based on the command set in Arduino, the relay module is triggered letting the desired appliance to function.

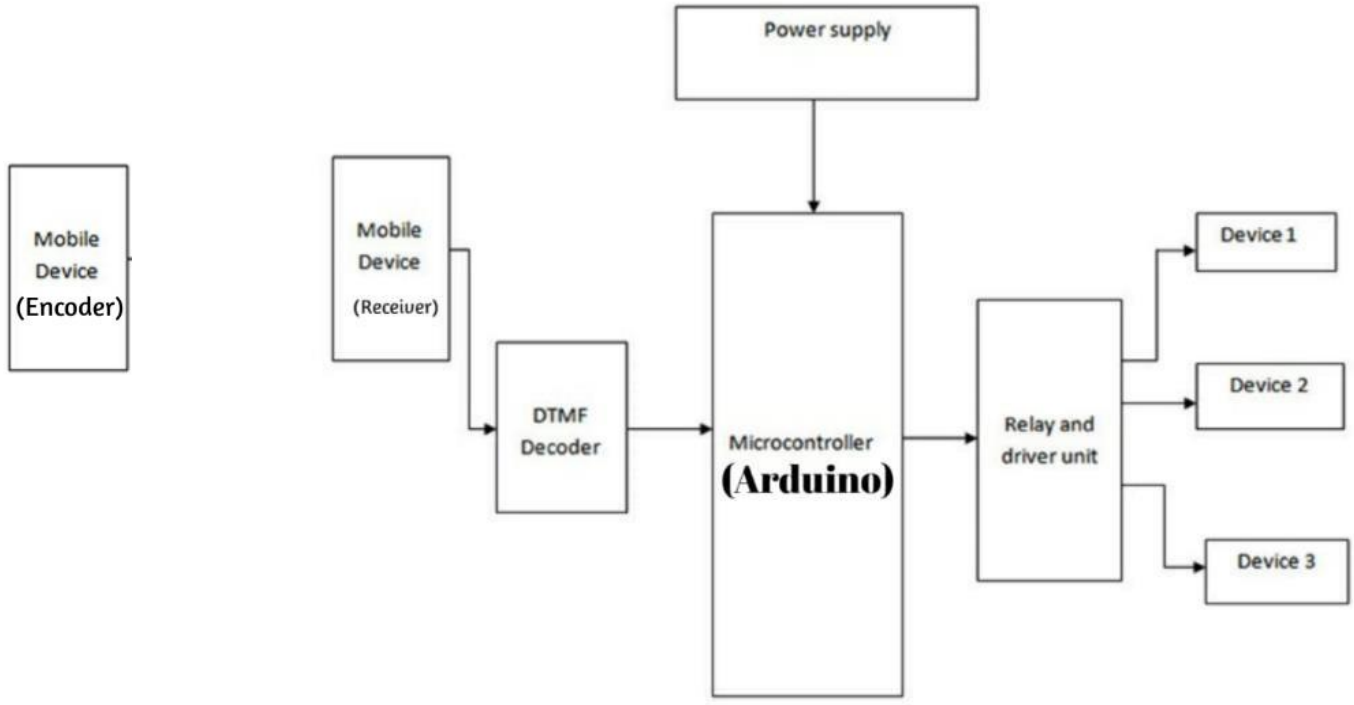
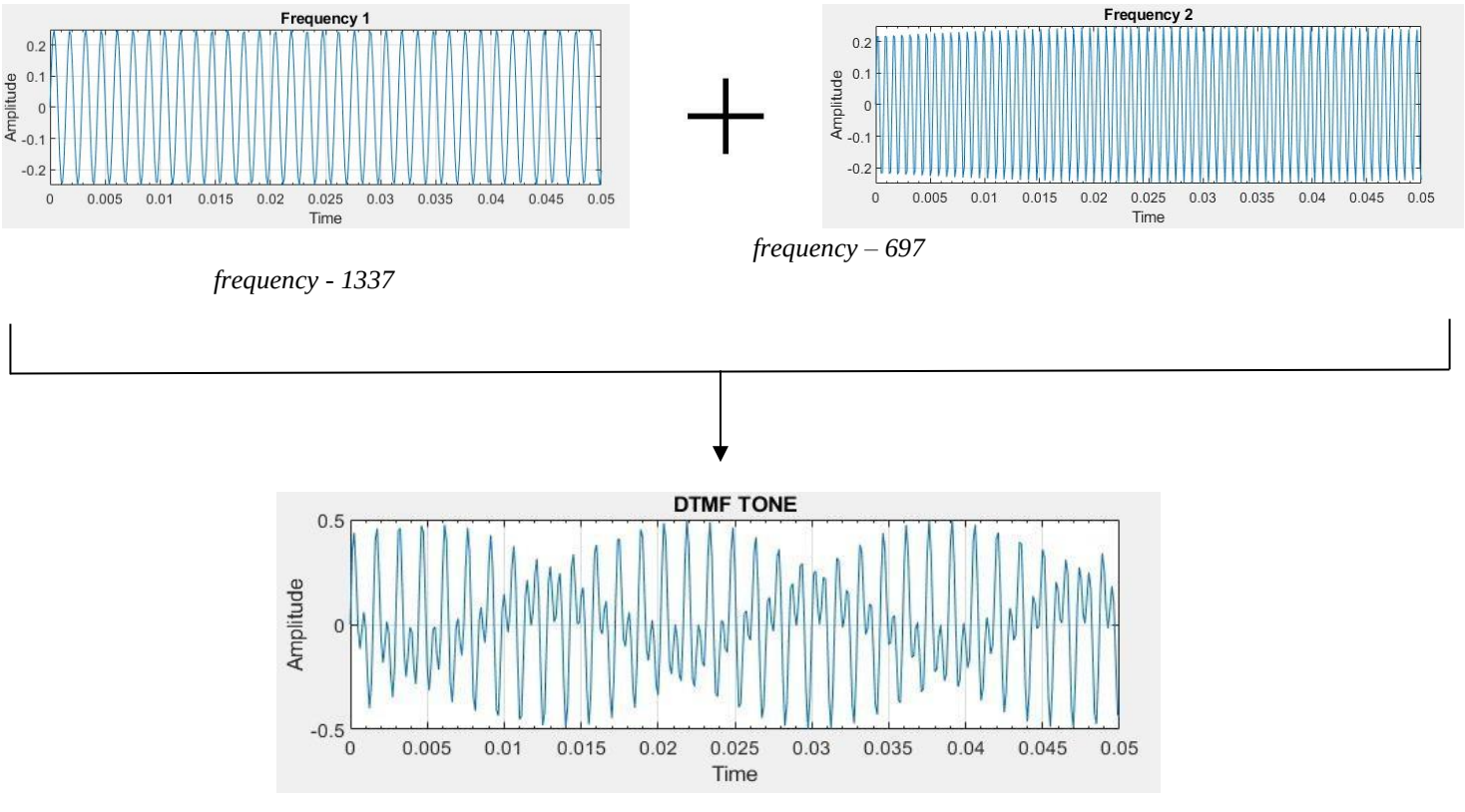


Fig: DTMF flow chart

3. Encoder:

DTMF encoder is the mobile device dial pad of the user. The 4 rows and 4 columns of the keypad form a matrix consisting of low frequency along the rows and high frequency along with the columns. When a key is pressed, a specific frequency is generated which is the result of the combination of the two frequencies: one from low-band and one from high-band.



frequency - 1337

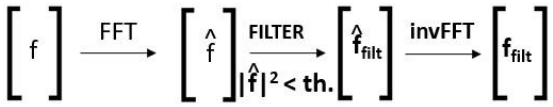
frequency – 697

fig: formation of DTMF signal

4. Decoder:

4.1 Denoising signals

While transferring sound signals from encoder to decoder there is always a possibility of adding up noise along with the required signal. In order to denoise the signal Fast Fourier Transform is used. The signal is Fourier transformed to obtain a Fourier coefficient vector having complex entries, each having a magnitude and phase. The magnitude lets how much of the particular sine and cosine function is present in the signal and phase tells if the signal is more of a sine or cosine constituent. A plot representing the power spectral density VS the frequency of each of the elements of the vectors shows which frequency in the signal has the most power. To filter the noise, we represent a threshold PSD value. A vector of 1's and 0's, it's 1 where every PSD is higher than the threshold and 0 where it is less, is multiplied to the FFT vector, inverse Fourier of which gives a clear signal without noise.



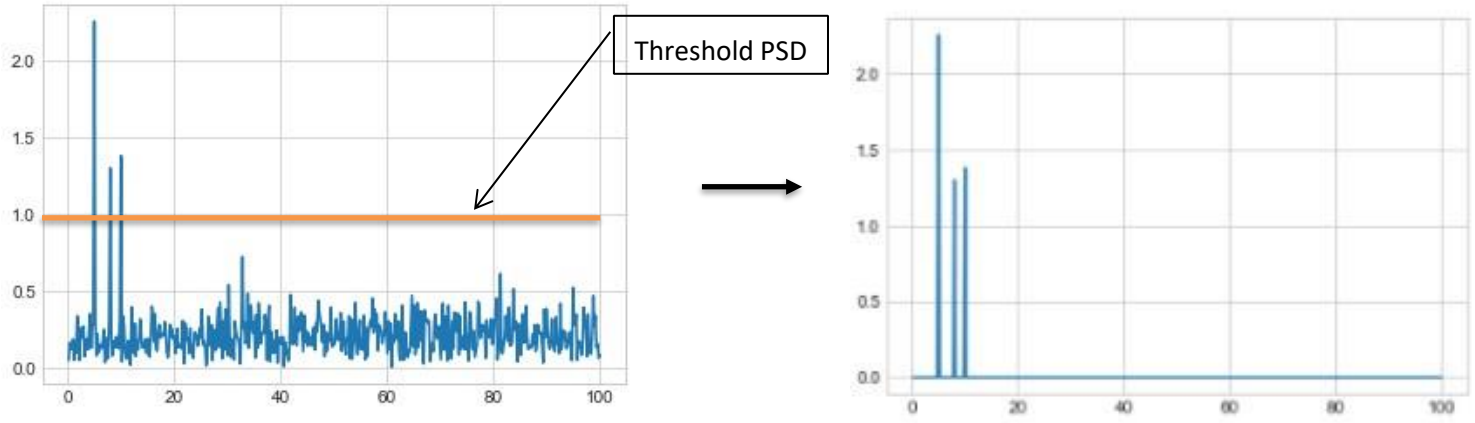


Fig: noise removal

4.2 Bandpass filter

Even after the removal of noise, the possibility of removal of noise having PSD above the threshold resides. To obtain the frequencies required from DTMF and to reject all other frequencies, we use FIR bandpass filters. The centre frequency of the filter is the DTMF frequency which is required. This is done by defining an impulse response similar to the signal. After obtaining a better signal we decode the signal and decompose the signal into two different DTMF frequencies: one low and the other be high. Bypassing the signal through 8 different bandpass filters each with a specific DTMF frequency we determine that two filters have the highest output. Comparing the corresponding data of numbers, we can determine which key was pressed. The binary of the corresponding number is the output of the decoder which is then transferred to the Arduino.

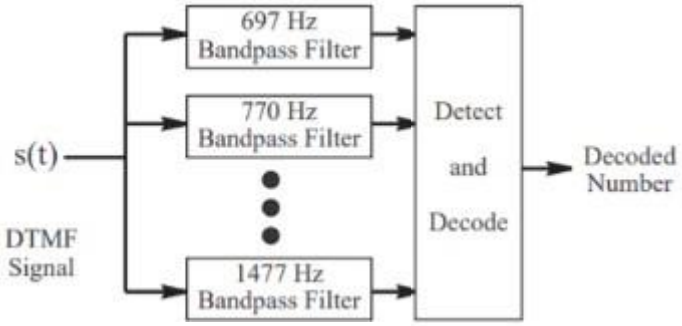


Fig : Decoding DTMF

4.3 Arduino:

Arduino Uno is based on the microcontroller ATmega328 (MCU). Out of the 14-digital input/output pins of Arduino, 4 pins receive the 4-bit input from the decoder. The 4-bit output is directed to the relay module through 4 output pins. The relation between the input and output is dependent on the relation provided through onboard Arduino programming language using Arduino IDE provided through a USB cable connection. For instance, if 0011 is the input bits and the program is set to provide high, low, high, low as output, which acts as an input for the relay module.

Fig : Arduino

4.4 Relay:

The relay module provides the functionality to turn on/off the current. A 4-relay channel module is used to fulfil the requirements of the automation of appliances requiring higher current. Each of these channels reviews high/low voltage from the Arduino. For instance, if a high, low, high, low sequence is received from Arduino, the relay with a high signal will set to operate resulting in the appliance connected to the 1st and the 3rd relay to turn on.

Fig : Relay

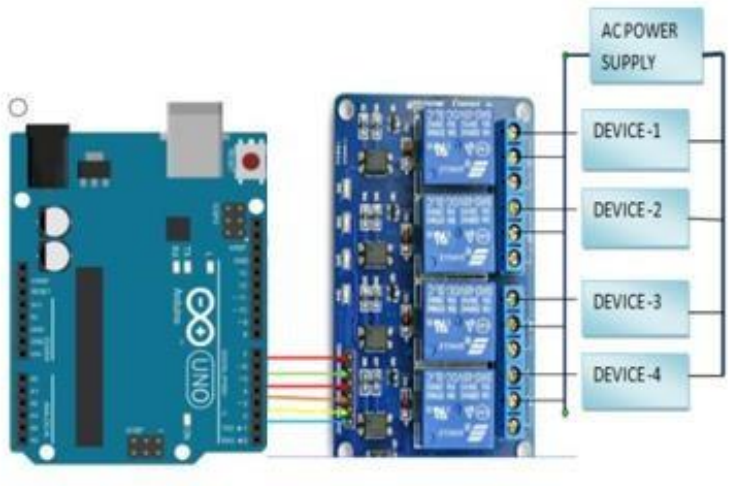


fig: relay and Arduino connection

4.5 Arduino to relay relation

key No.	**Arduino input**	**Relay module input**	**key No**	**Arduino input**	**Relay module input**
1	0001	LLLH	9	1001	HLLH
2	0010	LLHL	A	1010	HLHL
3	0011	LLHH	B	1011	HLHH
4	0100	LHLL	C	1100	HHLL
5	0101	LHLH	D	1101	HHLH
6	0110	LHHL	*	1110	HHHL

7	0111	LHHH	0	0000	LLLL
8	1000	HLLL	#	1111	HHHH

5. User interface:

On-screen mobile dial pad, a 4x4 key matrix, is the provided interface for the user. The Rows are assigned 4 different low frequencies and the 4 high frequencies are assigned to the columns. The assigned frequencies are neither multiple of the others nor the addition or subtraction, that being the case, pressing of each key generates a unique frequency.

Fig: Dial pad

6. Program [Matlab Simulation]

6.1 dtmf

```
function varargout = dtmf(varargin)
% DTMF MATLAB code for dtmf.fig %      DTMF, by itself, creates a new DTMF or raises the existing  % singleton*.
%      H = DTMF returns the handle to a new DTMF or the handle to
%      the existing singleton*.
%      DTMF('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in DTMF.M with the given input arguments.
%      DTMF('Property','Value',...) creates a new DTMF or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before dtmf_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to dtmf_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 dtmf
% Last Modified by GUIDE v2.5 07-Dec-2020 21:45:04
```

```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1; gui_State = struct('gui_Name',
mfilename, ...
                'gui_Singleton',  gui_Singleton, ...
                'gui_OpeningFcn', @dtmf_OpeningFcn, ...
                'gui_OutputFcn',  @dtmf_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 dtmf is made visible. function
dtmf_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 dtmf (see
VARARGIN)

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

% Update handles structure guidata(hObject,
handles);

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

% --- Outputs from this function are returned to the command
line.
function varargout = dtmf_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;


% --- 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)
t=[0:0.000125:.06];
fs=8000; f1=697;f2=1209;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;
sound(y,fs) subcode;


% --- 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)
t=[0:0.000125:.05];
fs=8000;
f1=697;f2=1336;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;sound(y,fs)
subcode;


% --- 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)
t=[0:0.000125:.05];
fs=8000;
f1=697;f2=1663;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;sound(y,fs)
subcode;


% --- 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)
t=[0:0.000125:.05];
fs=8000;
f1=697;f2=1447;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;sound(y,fs)
subcode;


% --- Executes on button press in pushbutton5. function
pushbutton5_Callback(hObject, eventdata, handles) %
hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
t=[0:0.000125:.05];
fs=8000;
f1=770;f2=1209;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;sound(y,fs)
subcode;


% --- Executes on button press in pushbutton6. function
pushbutton6_Callback(hObject, eventdata, handles) %
hObject    handle to pushbutton6 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
t=[0:0.000125:.05]; fs=8000;
f1=770;f2=1336;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;sound(y,fs) subcode;

% --- Executes on button press in pushbutton7. function
pushbutton7_Callback(hObject, eventdata, handles) % hObject
handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
t=[0:0.000125:.05]; fs=8000;
f1=770;f2=1633;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;sound(y,fs)
subcode;


% --- Executes on button press in pushbutton8. function
pushbutton8_Callback(hObject, eventdata, handles) % hObject
handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
t=[0:0.000125:.05]; fs=8000;
f1=770;f2=1477;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;sound(y,fs)
subcode;


% --- Executes on button press in pushbutton9. function
pushbutton9_Callback(hObject, eventdata, handles) % hObject
handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
t=[0:0.000125:.05]; fs=8000;
f1=852;f2=1209;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;sound(y,fs)
subcode;


% --- Executes on button press in pushbutton10. function
pushbutton10_Callback(hObject, eventdata, handles) % hObject
handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
t=[0:0.000125:.05]; fs=8000;
f1=852;f2=1336;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;sound(y,fs)
subcode;


% --- Executes on button press in pushbutton11. function
pushbutton11_Callback(hObject, eventdata, handles) %
hObject    handle to pushbutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
t=[0:0.000125:.05];
fs=8000;
f1=852;f2=1633;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;sound(y,fs)
subcode;


% --- Executes on button press in pushbutton12. function
pushbutton12_Callback(hObject, eventdata, handles) %
hObject    handle to pushbutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
t=[0:0.000125:.05];
fs=8000;
f1=852;f2=1477;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;sound(y,fs)
subcode;


% --- Executes on button press in pushbutton13. function
pushbutton13_Callback(hObject, eventdata, handles) %
hObject    handle to pushbutton13 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
t=[0:0.000125:.05];
fs=8000;
f1=941;f2=1209;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;sound(y,fs)
subcode;


% --- Executes on button press in pushbutton14. function
pushbutton14_Callback(hObject, eventdata, handles) %
hObject    handle to pushbutton14 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
t=[0:0.000125:.05];
fs=8000;
f1=941;f2=1336;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;sound(y,fs)
subcode;


% --- Executes on button press in pushbutton15. function
pushbutton15_Callback(hObject, eventdata, handles) %
hObject    handle to pushbutton15 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
```

```
t=[0:0.000125:.05]; fs=8000;
f1=941;f2=1633;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;sound(y,fs)
subcode;


% --- Executes on button press in pushbutton16. function
pushbutton16_Callback(hObject, eventdata, handles) % hObject
handle to pushbutton16 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
t=[0:0.000125:.05];
fs=8000; f1=941;f2=1477;
y1=.25*sin(2*pi*f1*t);
y2=.25*sin(2*pi*f2*t);
y=y1+y2;sound(y,fs);
subcode;


function op_Callback(hObject, eventdata, handles) %
hObject    handle to op (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 op as text %
str2double(get(hObject,'String')) returns contents of op as
a double

% --- Executes during object creation, after setting all properties.
function op_CreateFcn(hObject, eventdata, handles) %
hObject    handle to op (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 exit. function
exit_Callback(hObject, eventdata, handles) %
hObject    handle to exit (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
close;


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

```
uiwait(msgbox({'Developed by';'Jinil chandarana';'Keyur
Nagar';'Maulikkumar Bhalani';'Umang Patel' }
,'About','none'));
```

6.2 subcode

```
axes(handles.fre1); plot(t,y1);
set(handles.fre1,'XMinorTick'
,'on'); title('Frequency
1');xlabel('Time');
ylabel('Amplitude');grid;
axes(handles.fre2); plot(t,y2);
set(handles.fre2,'XMinorTick'
,'on'); title('Frequency
2');xlabel('Time');
ylabel('Amplitude');grid;


axes(handles.enconv); plot(t,y);
set(handles.enconv,'XMinorTick','on')
; title('convoluted
signal');xlabel('Time');
ylabel('Amplitude');grid;

 np= rand(1,length(t)); % Function to generate random
noise
x=y+np;
axes(handles.convnois
e) plot(x)  title ('noise
added');  grid on;

axes(handles.nois
e) plot(np)  title
('noise signal');
grid on;


%remove        the
noise         dt     =
0.000125;   L     =
length(t);   fhat   =
fft(x,L);
PSD = fhat.*conj(fhat)/L;


% creating frequency
axis freq =
1/(dt*L)*(0:L);
n=1:floor(L/2); indices =
PSD>0.2; PSDclean =
PSD.*indices; fhat =
indices.*fhat;
ffilt = ifft(fhat);

axes(handles.removenois
e) plot(t,y); title('Noise
removed'); xlabel('Time
(s)');
ylabel('Amplitude');


%Filter process
rmain=2048*2;rmag=1024*
2; cn=9;cr=0.5;
cl=.25;ch=.28;
[b,a]=cheby1(cn,cr,cl);
```

```
yfilt1=filter(b,a,y);
h2=fft(yfilt1,rmain);
hmag2=abs(h2(1:rmag));
[b1,a1]=cheby1(cn,cr,ch,'high');
yfilt2=filter(b1,a1,y);

h3=fft(yfilt2,rmain);
hmag3=abs(h3(1:rmag));

axes(handles.fig1); plot(yfilt1);grid;
title('Filtered Low Freq. Signal');
xlabel('Time');ylabel('Amplitude');

axes(handles.fig2); plot(yfilt2);grid;
title('Filtered High Freq. Signal');
xlabel('Time');ylabel('Amplitude');

hlow=fft(yfilt1,rmain); hmaglow=abs(hlow);
axes(handles.fig3); plot(hmaglow(1:rmag));
title('FFT Low Pass');grid;


hhigh=fft(yfilt2,rmain); hmaghigh=abs(hhigh);
axes(handles.fig4); plot(hmaghigh(1:rmag));
title('FFT High Pass');grid;


m=max(abs(hmag2));n=max(abs(hmag3));
o=find(m==hmag2);p=find(n==hmag3); j=((o-1)*fs)/rmain;
k=((p-1)*fs)/rmain;


axes(handles.arduino); img =
imread('arduino.jpg');
imshow(img);


if j<=732.59 && k<=1270.91;
set(handles.op,'String','HALL FAN');
axes(handles.image);    img = imread('fan.jpeg');
imshow(img);

  elseif j<=732.59 & k<=1404.73;
set(handles.op,'String','TV');
axes(handles.image);       img =
imread('TV.jpg');       imshow(img);


  elseif j<=732.59 & k<=1553.04;
set(handles.op,'String','GEYSER');
axes(handles.image);      img =
imread('geyser.jpg');
    imshow(img);

  elseif j<=732.59 & k>1553.05;
set(handles.op,'String','AC');
axes(handles.image);      img =
imread('AC.jpg');
    imshow(img);

  elseif j<=809.96 & k<=1270.91;
set(handles.op,'String','ELECTRICAL MOTOR');
    axes(handles.image);      img = imread('electrical-
motor.jpg');      imshow(img);    elseif j<=809.96 &
k<=1404.73;      set(handles.op,'String','PREHEAT
OVEN');      axes(handles.image);      img =
imread('oven.jpg');      imshow(img);

  elseif j<=809.96 & k<=1553.04;
set(handles.op,'String','WASHING MACHINE');
axes(handles.image);      img = imread('washing-
machine.jpg');      imshow(img);    elseif
j<=809.96 & k>1553.05;
set(handles.op,'String','GARAGE GATE');
axes(handles.image);      img = imread('gate.jpg');
    imshow(img);

  elseif j<=895.39 & k<=1270.91;
set(handles.op,'String','REFRIGERATOR');
axes(handles.image);      img =
imread('fridge.jpg');      imshow(img);    elseif
j<=895.39 & k<=1404.73;
set(handles.op,'String','HALL LIGHT');
axes(handles.image);      img =
imread('HallLight.jpg');
    imshow(img);

  elseif j<=895.39 & k<=1553.04;
set(handles.op,'String','ROOM LIGHT');
axes(handles.image);      img =
imread('bulb1.jpg');
    imshow(img);

  elseif j<=895.39 & k>1553.05;
set(handles.op,'String','ROOM FAN');
axes(handles.image);      img =
imread('fan1.jpg');
    imshow(img);

  elseif j>895.40 & k<=1270.91;
set(handles.op,'String','ALARM');
axes(handles.image);      img =
imread('alarm.jpg');
    imshow(img);

  elseif j>895.40 & k<=1404.73;
set(handles.op,'String','TURN OFF ALL DEVICES');
    axes(handles.image);
img = imread('turnoff.jpg');
    imshow(img);

  elseif j>895.40 & k<=1553.04;
set(handles.op,'String','TURN ON MAIN POWER');
    axes(handles.image);
img = imread('powerOn.jpg');
    imshow(img);

  elseif j>895.40 & k>1553.05;
set(handles.op,'String','TURN OFF MAIN POWER');
    axes(handles.image);
img = imread('poweroff.jpg');
imshow(img); end
```

7. RESULTS:

As soon as the user presses any key the two freqs. signal corresponding to the key is generated. The combination of which is passed to the decoder installed at home. The first step of decoder is to remove the noise using FFT. The denoised signal is filtered using BPF and the output of the filter tells which two frequencies are present. This data is converted to binary, send to arduino, accoring to the Arduino, corresponding relay in the relay module is triggered resulting the functioning of thr appliance.

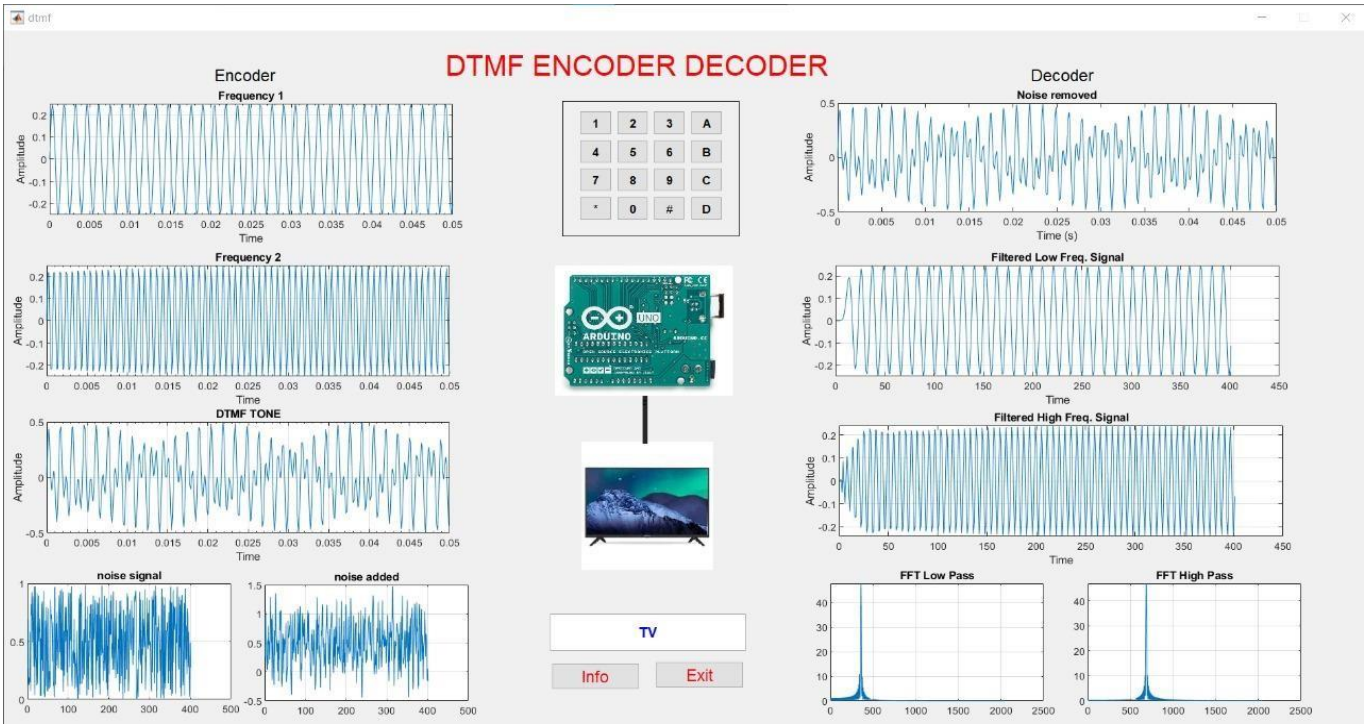


Fig: matlab simulation result

8. Conclusion:

DTMF used for home Automation have high range is highly effective. It has been conceivable to control all home appliances automatically with the help of mobile phones. Nowadays Mobile phones are an essential part of life; hence anybody can operate our system without any prior training. With the help of the project, we can operate home appliances from anywhere in the world. It reduces the wastage of the electricity when we forgot to switch off the lights & fans and gone outside. It is a blessing for handicapped peoples and also saves lots of time.

References

Banik, P. P. (2015). *ieeexplore*. Retrieved from ieeexplore.org:

https://ieeexplore.ieee.org/document/7506843

coskun, m. (2018, April 24). *medium*. Retrieved from medium.org: https://medium.com/swlh/noisehttps://medium.com/swlh/noise-removal-for-a-better-fast-fourier-transformation-284918d4250fremoval-for-a-better-fast-fourier-transformation-284918d4250f

Qiong wang, Jianguo chen, yu Huang. (2018). *ieeexplore*. Retrieved from ieeexplore.org: https://ieeexplore.ieee.org/document/8469726

S.Sircar. (n.d.). *engr*. Retrieved from engr.mun.ca: http://www.engr.mun.ca/~sircar/project1_files/dtmf.pdf seas. (2011). *seas*. Retrieved

from seas.ucla.edu: http://www.seas.ucla.edu/dsplab/ttt/over.html

the universty of mishigun. (n.d.). *eecs*. Retrieved from eecs.umich.edu: http://www.eecs.umich.edu/courses/eecs206/archive/fall03/public/lab/lab7/lab7.pdf