NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY

Signals and Systems

Project Report

GROUP 5 BEE 11-B

6/17/2021

TEAM MEMBERS

NAME	Registration No
Shehroze Amir	283937
Farheen Gul	290352
Ch. Muhammad Shaheer Yasir	286021
Muhammad Hamza	290951

INTRODUCTION

DTMF (Dual Tone Multiple Frequency) Encoder and Decoder informally named as Touch Tone allocates a unique tone to each button on an appliance (made up of two frequencies high and low). The DTMF Dialing system traces its roots to a technique AT & amp T developed in the 1950s known as MF (Multiple Frequency) which was developed within the AT & ampT telephone network to direct call between switching facilities using in-band signaling.

The DTMF system uses eight different frequency signals transmitted in pairs to represent sixteen different numbers, symbols and letters. The table shows how the frequencies are organized.

Frequencies	1209 Hz	1336 Hz	1477 Hz
697 Hz	1	2	3
770 Hz	4	5	6
852 Hz	7	8	9
941 Hz	*	0	#

Whenever a key is pressed on the Touch Tone, sinusoidal signal is generated containing the sum of two frequencies. These frequencies are chosen by design engineers because the sum or difference of any of the two frequencies gives a unique frequency which is not equal to any of the given frequency. With the help of this, dialed signal can easily be detected. The uniqueness of DTMF is that it is simple to generate and noiseimmune.

DESIGN STEPS AND CODE EXPLANATION

Our Project consists of four main parts:

- First, we have designed a keypad using GUI functionality of MATLAB.
- After designing the keypad, we have assigned a tune to each of these buttons.
- When the tune is produced, we have designed a bandpass filter and decoded the same tune using bandpass filters.
- Then, taking the output of bandpass filters as input, we have determined the same digits pressed using different switch statements.

Now, we will first discuss the global variables used in our code files and then we will discuss each functionality one by one.

GLOBAL VARIABLES

keyNames:
This is a vector which contains all the keys (i.e. digits) we have pressed on our keypad.

tone_all:
This is the encoded signal of the digits pressed. When we pass it to sound function, we will hear all the sounds of multiple frequencies contributing in the digits.

h1:
This is the graph of signal tone i.e. the encoded signal and the signal which has to be decoded again in time domain.

h2:
This is the graph of spectrum for tone i.e. decoded signal passed through bandpass filter in time domain.

h3:
This is the graph of BPF Frequency Response i.e. the fourier transform of bandpass filter.

h4:
This is the graph of decode spectrum i.e. the fourier transform of the decoded signal.

Decode_output:

This is vector which will contain the digits decoded through bandpass filters and several switch statements.

tone:

This is the encoded signal of a single digit. After the digit is pressed first tone signal is generated and then is added to tone_all after a pause of 0.5 seconds.

KEYPAD USING GRAPHICAL USER INTERFACE

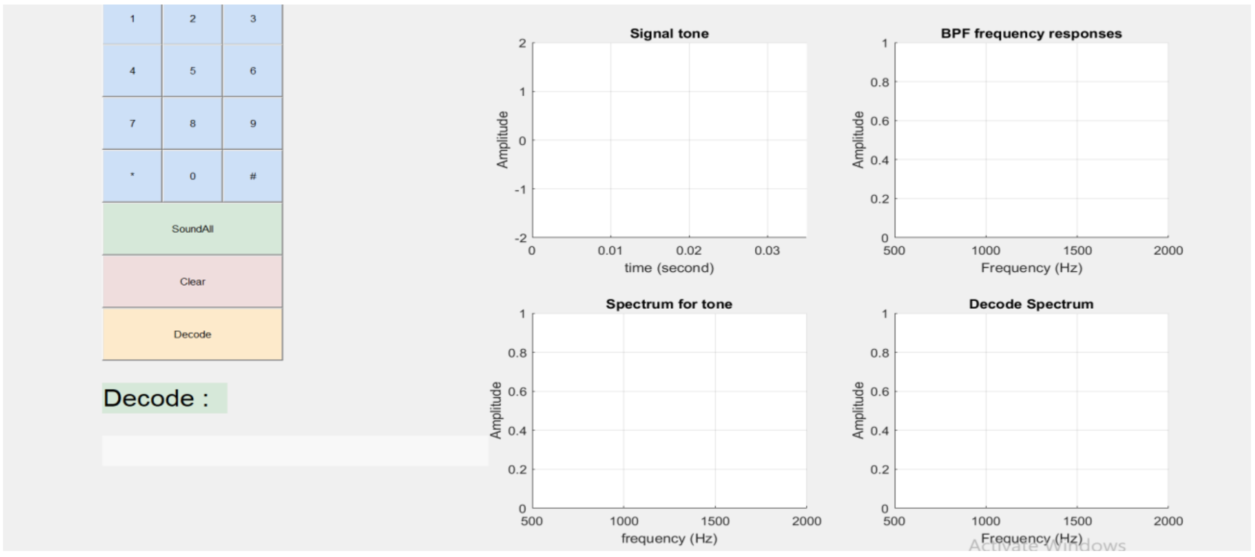
CODE FILE: Main.m

FUNCTIONS USED:

Subplot, Title, Xlabel, Ylabel, Axis, Set, UIControl,

Explanation:

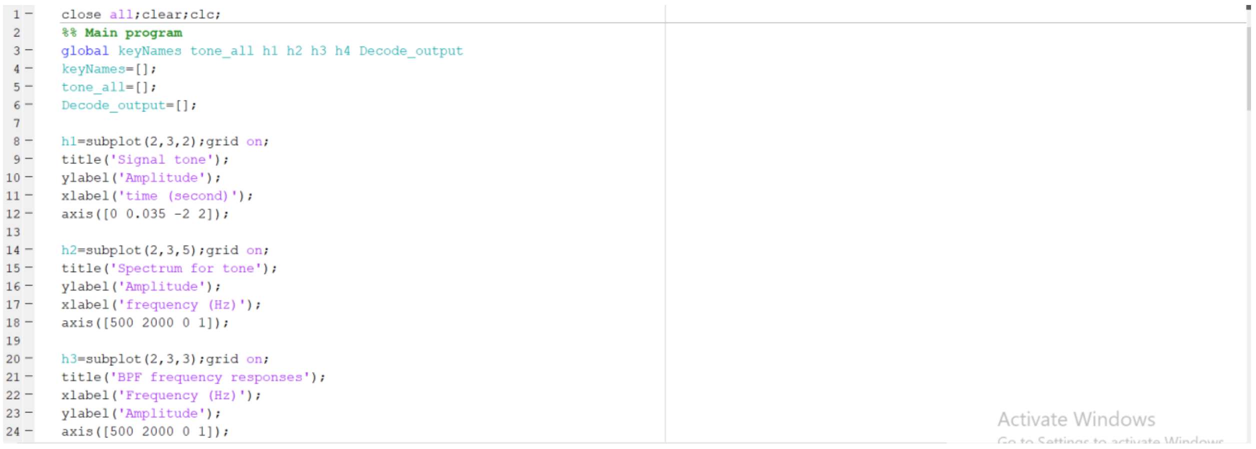
We have designed a keypad using different inbuilt functions of MATLAB for Graphical User Interface. Our User Interface screen consists of two parts. On the left side, we have different buttons having various functionalities while on the right side, we have four graphs which will show the time and frequency domain plots of our encoded and decoded signals.



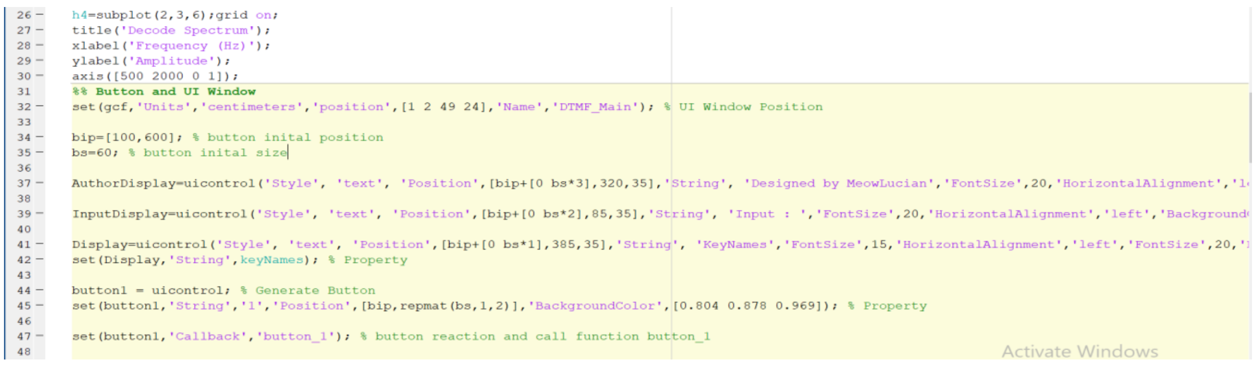
- First, we set the positions and build the four graphs on the right side using the MATLAB function **Subplot.**
- Then we give them titles, x and y labels and specified their axis using **Title, Xlabel, Ylabel** and **Axis** Function.

- Then, we designed our User Interface window using the **set** function and passed several parameters to have the desired window.
- Then, we set the initial position of our buttons and their sizes respectively. These will be passed as parameters to create several buttons.
- Then, we have created different variables AuthorDisplay, InputDisplay and Display showing different kinds of designs (having different fonts and styles) using MATLAB in-built function **UICONTROL.** We passed several parameters to get the design desired.
- Then, we have generated different buttons using Display design. We displayed the strings desired on every button i.e. 1 for button1 and 2 for button2 etc.
- Then as a button is pressed, we have a function for each button which will be called using CallBack as a parameter in **set** function.
- After generating buttons, we have space below for our decoded digits. We have displayed the digits decoded in the Display2 design.

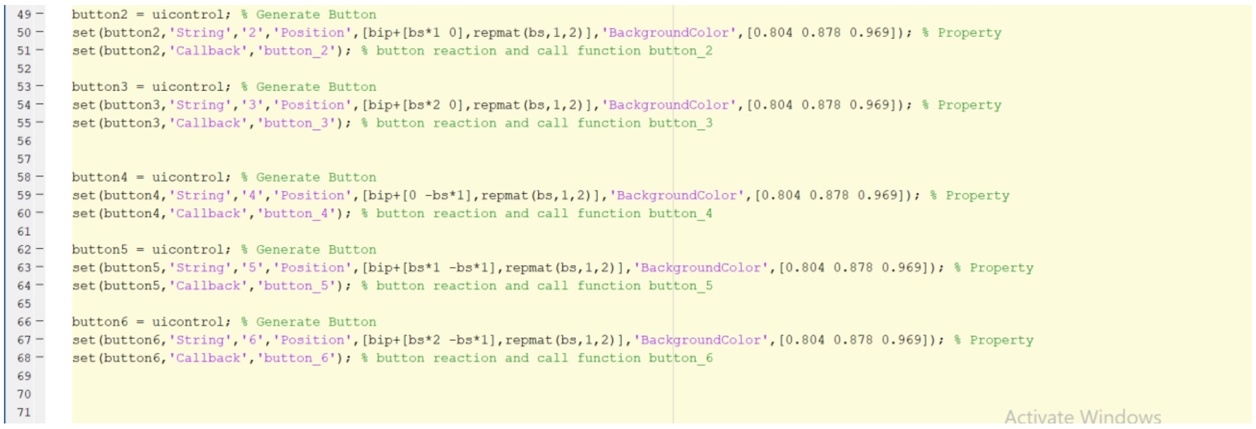
Code:



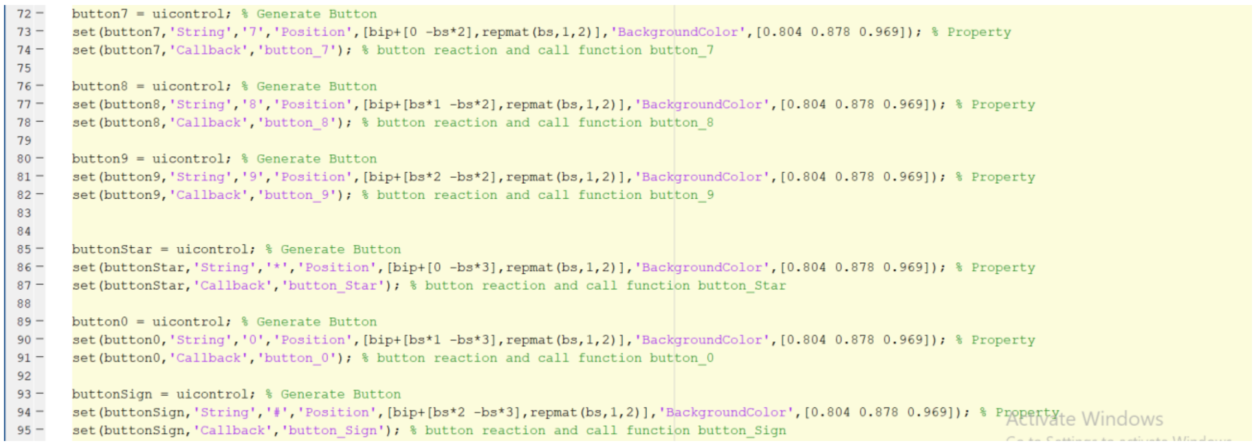
```
close all;clear;clc;
%% Main program
global keyNames tone_all h1 h2 h3 h4 Decode_output
keyNames=[];
tone_all=[];
Decode_output=[];

h1=subplot(2,3,2);grid on;
title('Signal tone');
ylabel('Amplitude');
xlabel('time (second)');
axis([0 0.035 -2 2]);

h2=subplot(2,3,5);grid on;
title('Spectrum for tone');
ylabel('Amplitude');
xlabel('frequency (Hz)');
axis([500 2000 0 1]);

h3=subplot(2,3,3);grid on;
title('BPF frequency responses');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
axis([500 2000 0 1]);
```



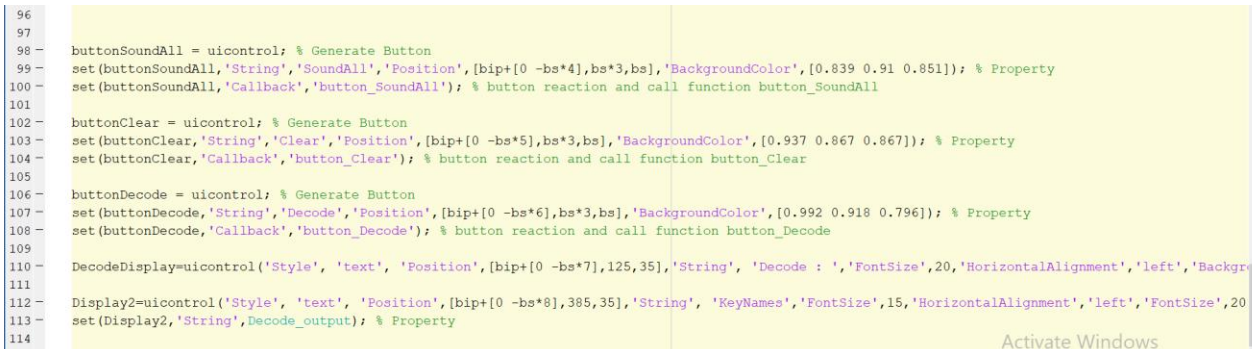
```
h4=subplot(2,3,6);grid on;
title('Decode Spectrum');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
axis([500 2000 0 1]);
%% Button and UI Window
set(gcf,'Units','centimeters','position',[1 2 49 24],'Name','DTMF_Main'); % UI Window Position

bip=[100,600]; % button inital position
bs=60; % button inital size

AuthorDisplay=uicontrol('Style', 'text', 'Position',[bip+[0 bs*3],320,35],'String', 'Designed by MeowLucian','FontSize',20,'HorizontalAlignment','l

InputDisplay=uicontrol('Style', 'text', 'Position',[bip+[0 bs*2],85,35],'String', 'Input : ','FontSize',20,'HorizontalAlignment','left','Background

Display=uicontrol('Style', 'text', 'Position',[bip+[0 bs*1],385,35],'String', 'KeyNames','FontSize',15,'HorizontalAlignment','left','FontSize',20,'

set(Display,'String',keyNames); % Property

button1 = uicontrol; % Generate Button
set(button1,'String','1','Position',[bip,repmat(bs,1,2)],'BackgroundColor',[0.804 0.878 0.969]); % Property

set(button1,'Callback','button_1'); % button reaction and call function button_1
```



```matlab
button2 = uicontrol; % Generate Button
set(button2,'String','2','Position',[bip+[bs*1 0],repmat(bs,1,2)],'BackgroundColor',[0.804 0.878 0.969]); % Property
set(button2,'Callback','button_2'); % button reaction and call function button_2

button3 = uicontrol; % Generate Button
set(button3,'String','3','Position',[bip+[bs*2 0],repmat(bs,1,2)],'BackgroundColor',[0.804 0.878 0.969]); % Property
set(button3,'Callback','button_3'); % button reaction and call function button_3


button4 = uicontrol; % Generate Button
set(button4,'String','4','Position',[bip+[0 -bs*1],repmat(bs,1,2)],'BackgroundColor',[0.804 0.878 0.969]); % Property
set(button4,'Callback','button_4'); % button reaction and call function button_4

button5 = uicontrol; % Generate Button
set(button5,'String','5','Position',[bip+[bs*1 -bs*1],repmat(bs,1,2)],'BackgroundColor',[0.804 0.878 0.969]); % Property
set(button5,'Callback','button_5'); % button reaction and call function button_5

button6 = uicontrol; % Generate Button
set(button6,'String','6','Position',[bip+[bs*2 -bs*1],repmat(bs,1,2)],'BackgroundColor',[0.804 0.878 0.969]); % Property
set(button6,'Callback','button_6'); % button reaction and call function button_6
```



```matlab
button7 = uicontrol; % Generate Button
set(button7,'String','7','Position',[bip+[0 -bs*2],repmat(bs,1,2)],'BackgroundColor',[0.804 0.878 0.969]); % Property
set(button7,'Callback','button_7'); % button reaction and call function button_7

button8 = uicontrol; % Generate Button
set(button8,'String','8','Position',[bip+[bs*1 -bs*2],repmat(bs,1,2)],'BackgroundColor',[0.804 0.878 0.969]); % Property
set(button8,'Callback','button_8'); % button reaction and call function button_8

button9 = uicontrol; % Generate Button
set(button9,'String','9','Position',[bip+[bs*2 -bs*2],repmat(bs,1,2)],'BackgroundColor',[0.804 0.878 0.969]); % Property
set(button9,'Callback','button_9'); % button reaction and call function button_9


buttonStar = uicontrol; % Generate Button
set(buttonStar,'String','*','Position',[bip+[0 -bs*3],repmat(bs,1,2)],'BackgroundColor',[0.804 0.878 0.969]); % Property
set(buttonStar,'Callback','button_Star'); % button reaction and call function button_Star

button0 = uicontrol; % Generate Button
set(button0,'String','0','Position',[bip+[bs*1 -bs*3],repmat(bs,1,2)],'BackgroundColor',[0.804 0.878 0.969]); % Property
set(button0,'Callback','button_0'); % button reaction and call function button_0

buttonSign = uicontrol; % Generate Button
set(buttonSign,'String','#','Position',[bip+[bs*2 -bs*3],repmat(bs,1,2)],'BackgroundColor',[0.804 0.878 0.969]); % Property
set(buttonSign,'Callback','button_Sign'); % button reaction and call function button_Sign
```



```matlab


buttonSoundAll = uicontrol; % Generate Button
set(buttonSoundAll,'String','SoundAll','Position',[bip+[0 -bs*4],bs*3,bs],'BackgroundColor',[0.839 0.91 0.851]); % Property
set(buttonSoundAll,'Callback','button_SoundAll'); % button reaction and call function button_SoundAll

buttonClear = uicontrol; % Generate Button
set(buttonClear,'String','Clear','Position',[bip+[0 -bs*5],bs*3,bs],'BackgroundColor',[0.937 0.867 0.867]); % Property
set(buttonClear,'Callback','button_Clear'); % button reaction and call function button_Clear

buttonDecode = uicontrol; % Generate Button
set(buttonDecode,'String','Decode','Position',[bip+[0 -bs*6],bs*3,bs],'BackgroundColor',[0.992 0.918 0.796]); % Property
set(buttonDecode,'Callback','button_Decode'); % button reaction and call function button_Decode

DecodeDisplay=uicontrol('Style', 'text', 'Position',[bip+[0 -bs*7],125,35],'String', 'Decode : ','FontSize',20,'HorizontalAlignment','left','Backgr

Display2=uicontrol('Style', 'text', 'Position',[bip+[0 -bs*8],385,35],'String', 'KeyNames','FontSize',15,'HorizontalAlignment','left','FontSize',20
set(Display2,'String',Decode_output); % Property

```

ASSIGNING A TUNE TO EACH BUTTON

Code Files:

button_0.m, button_1.m, button_2.m, button_3.m, button_4.m, button_5.m, button_6.m, button_7.m, button_8.m, button_9.m, button_Sign.m, button_Star.m, button_Clear.m, button_Soundall.m

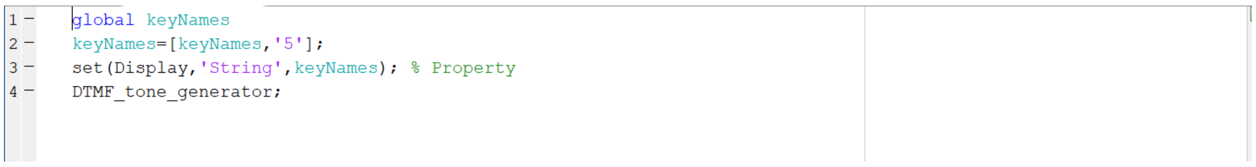
Functions Used:

set, cla, subplot, DTMF_tone_generator, plot, title, xlabel, ylabel, plot, soundsc, axis, pause, audio_write

button:

After designing the button on the keypad, we have written their functionality in all these button files.

Code:



```
global keyNames
keyNames=[keyNames,'5'];
set(Display,'String',keyNames); % Property
DTMF_tone_generator;
```

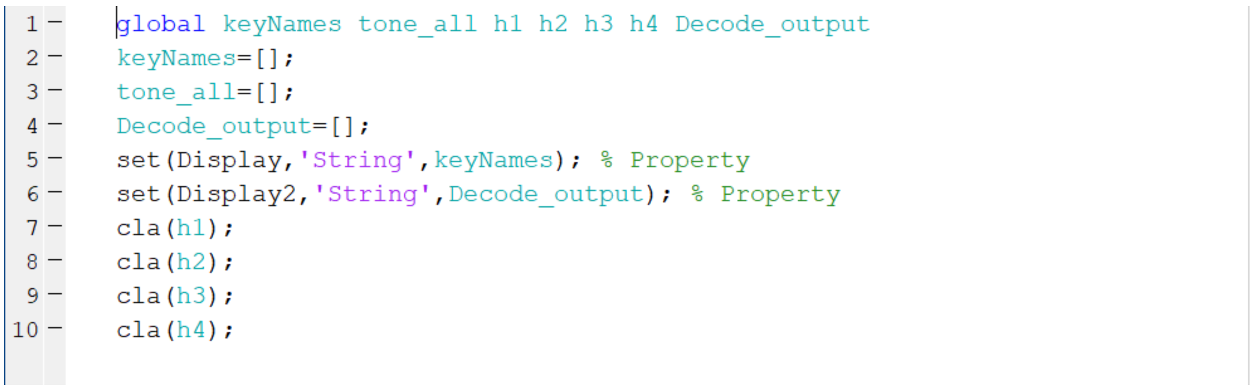
- First when the button is pressed, we have added the key assigned to the button i.e. 5 for button_5 to the vector keyNames. keyNames contains all the keys pressed.
- Then, using Set function of Matlab we have displayed the name of key pressed on the button i.e. restoring the button to original form.
- Then we called the function DTMF_tone_generator. It will create the sum of sinusoidal signals of multiple frequencies (one high and one low for a single key) and then plot the signal and produce the sound. We will discuss it more in the next module.

Similarly, we have created the functionality of buttons from 0 to 9 as well as # and *.

Button_clear:

It will clear all the vectors, tones and reset the system.

Code:



```
global keyNames tone_all h1 h2 h3 h4 Decode_output
keyNames=[];
tone_all=[];
Decode_output=[];
set(Display,'String',keyNames); % Property
set(Display2,'String',Decode_output); % Property
cla(h1);
cla(h2);
cla(h3);
cla(h4);
```

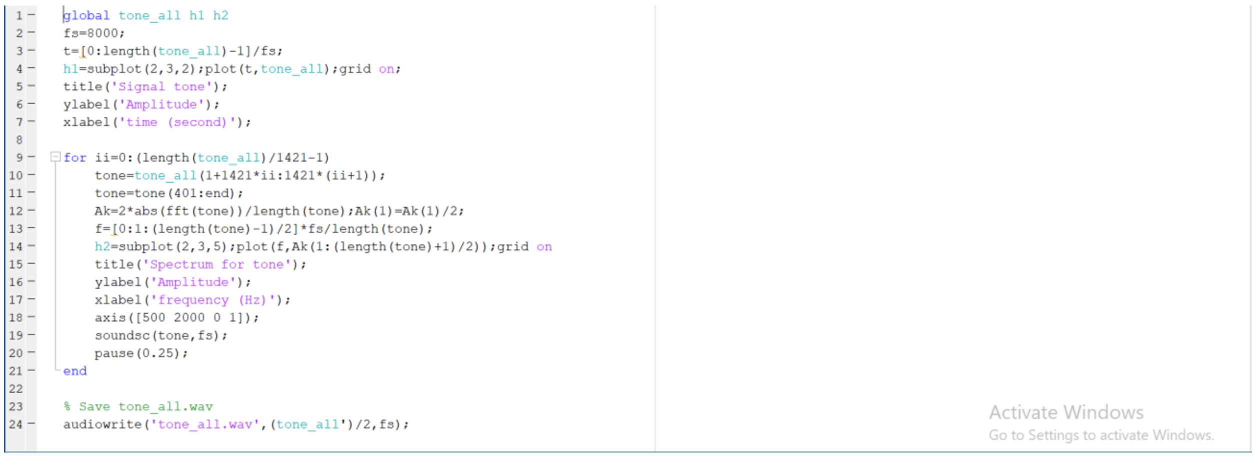
- First we have set keyNames to empty vector i.e. clear all the values. Similarly we have cleared all the tones already stored by assigning tone_all an empty vector.
- Then, we cleared decode_output by assigning it to an empty vector.

- Then we cleared any string printed on UI by printing the cleared vectors of keyNames and Decode_output using **set** function of MATLAB.
- Then, we cleared all the four graphs printed on UI using **cla.**

Button_SoundAll:

This button will plot all the tones stored yet in tone_all vector. First it plot it in time domain and then plot it in frequency domain tone by tone. Then it produced different sounds tone by tone and finally it converts the tone_all signal into a .wav function using audiowrite function.

Code:



```
global tone_all h1 h2
fs=8000;
t=[0:length(tone_all)-1]/fs;
h1=subplot(2,3,2);plot(t,tone_all);grid on;
title('Signal tone');
ylabel('Amplitude');
xlabel('time (second)');

for ii=0:(length(tone_all)/1421-1)
    tone=tone_all(1+1421*ii:1421*(ii+1));
    tone=tone(401:end);
    Ak=2*abs(fft(tone))/length(tone);Ak(1)=Ak(1)/2;
    f=[0:1:(length(tone)-1)/2]*fs/length(tone);
    h2=subplot(2,3,5);plot(f,Ak(1:(length(tone)+1)/2));grid on
    title('Spectrum for tone');
    ylabel('Amplitude');
    xlabel('frequency (Hz)');
    axis([500 2000 0 1]);
    soundsc(tone,fs);
    pause(0.25);
end

% Save tone_all.wav
audiowrite('tone_all.wav',(tone_all')/2,fs);
```

- First we have set our sampling frequency as 8000 as given in the project details.
- Then we defined the time axis and then plotted the tone_all with respect to time axis in the first graph.
- Then, we used a for loop which deals with each tone one by one.
- First we have limit the axis of our tone because the axis was very large and very difficult to handle. Then we defined the frequency axis and defined Ak using the formula:

$$A_K = \frac{2}{N}\sqrt{|X(k)|^2}$$

- Then we plotted our tone in frequency domain and set the axis respectively.
- Then we produced the sound our signal using **soundsc** signal. Then we took a pause of 0.25 secs using **pause** function and moved to next tone.
- Finally we converted our tone_all signal into a .wav file using **audiowrite** function.

DTMF TONE GENERATOR

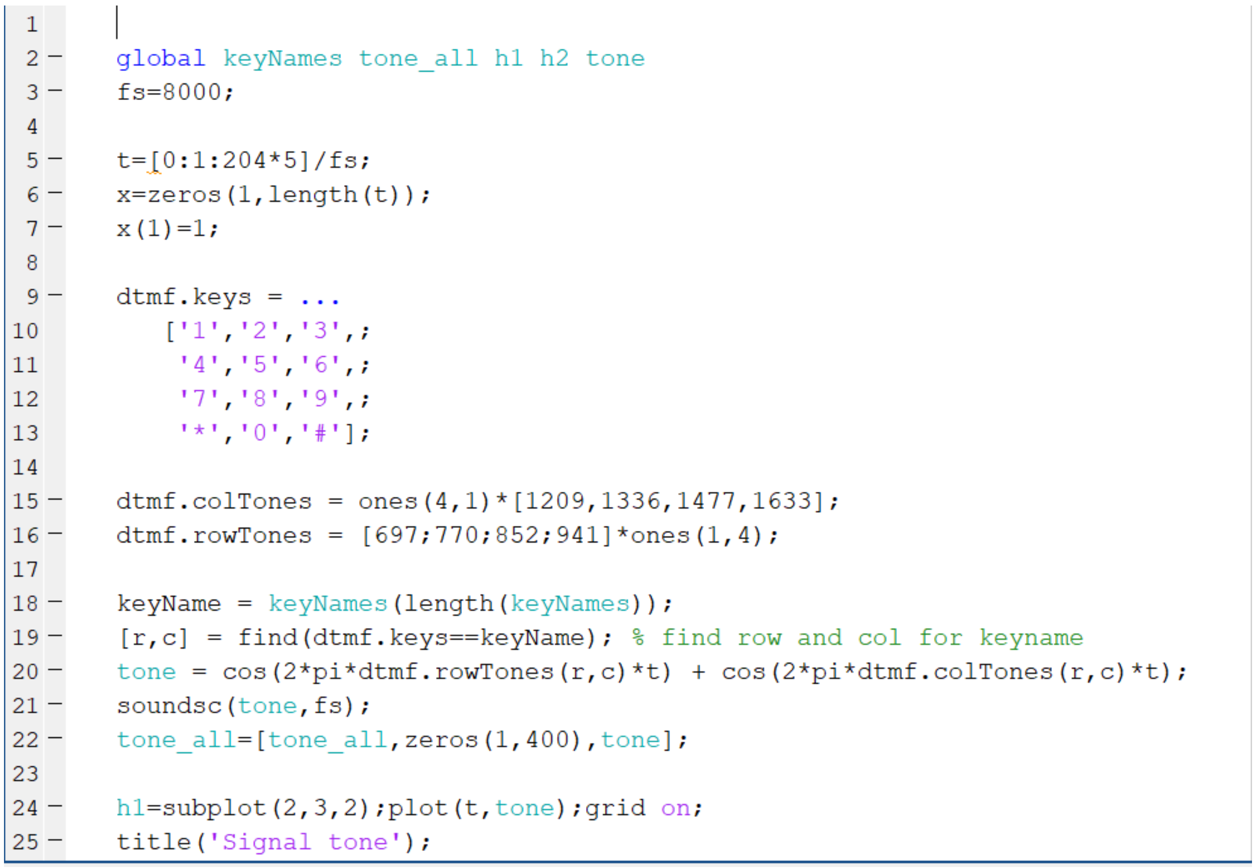
Code Files: **DTMF_tone_generator.m**

Functions Used:

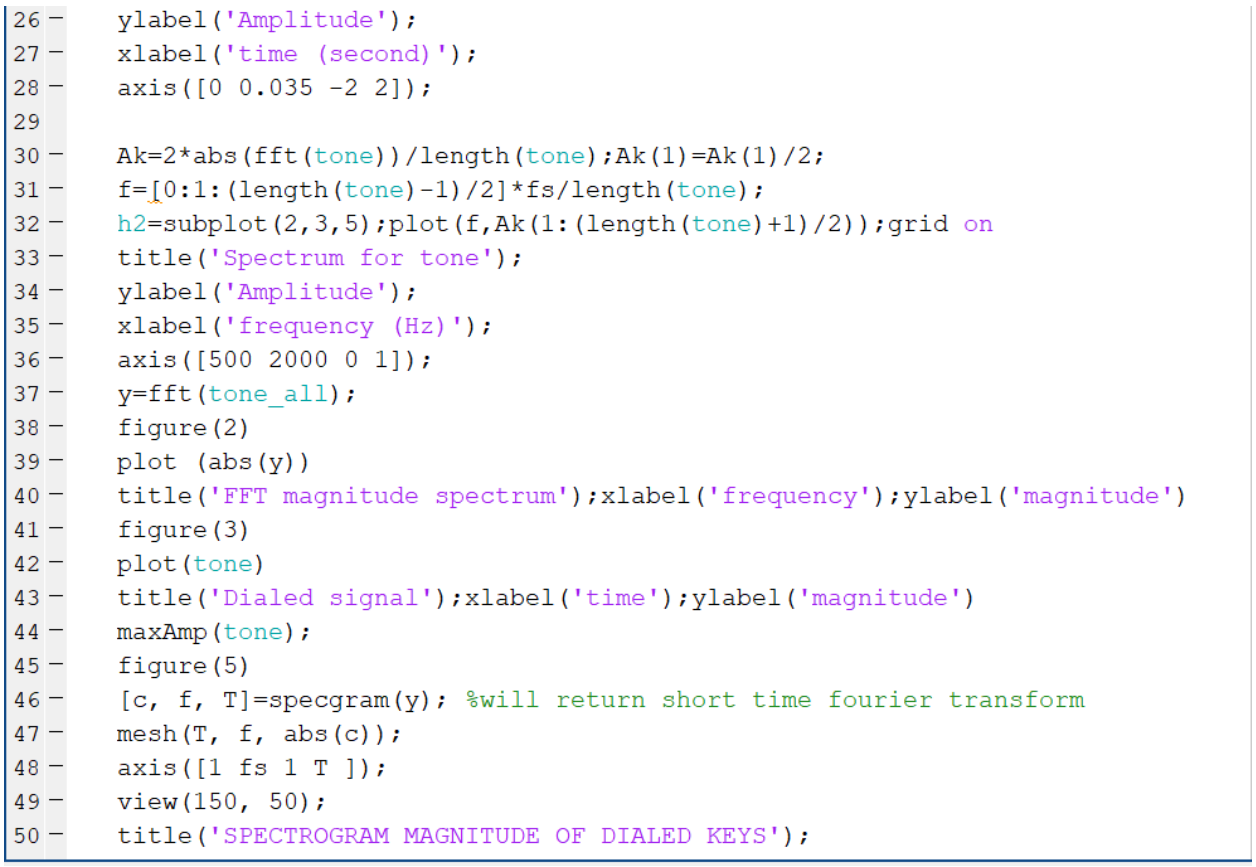
zeros, ones, length, find, cos, soundsc, subplot, plot, title, xlabel, ylabel, zlabel, axis, fft, abs, figure, maxAmp, specgram, mesh, view

It will encode the keys pressed and generate a signal which is the sum of sinusoidal signals of multiple frequencies (one high and one low for each key pressed). Then, it will plot the encoded signal in the first graph and then plotted it in frequency domain in the second graph. Then, it generated multiple figures of the encoded signal which highlights different properties of the signal.

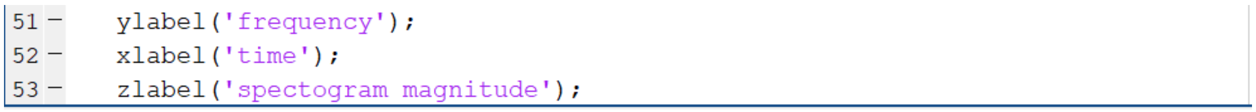
Code:



```

global keyNames tone_all h1 h2 tone
fs=8000;

t=[0:1:204*5]/fs;
x=zeros(1,length(t));
x(1)=1;

dtmf.keys = ...
    ['1','2','3',;
     '4','5','6',;
     '7','8','9',;
     '*','0','#'];

dtmf.colTones = ones(4,1)*[1209,1336,1477,1633];
dtmf.rowTones = [697;770;852;941]*ones(1,4);

keyName = keyNames(length(keyNames));
[r,c] = find(dtmf.keys==keyName); % find row and col for keyname
tone = cos(2*pi*dtmf.rowTones(r,c)*t) + cos(2*pi*dtmf.colTones(r,c)*t);
soundsc(tone,fs);
tone_all=[tone_all,zeros(1,400),tone];

h1=subplot(2,3,2);plot(t,tone);grid on;
title('Signal tone');
```



```
ylabel('Amplitude');
xlabel('time (second)');
axis([0 0.035 -2 2]);

Ak=2*abs(fft(tone))/length(tone);Ak(1)=Ak(1)/2;
f=[0:1:(length(tone)-1)/2]*fs/length(tone);
h2=subplot(2,3,5);plot(f,Ak(1:(length(tone)+1)/2));grid on
title('Spectrum for tone');
ylabel('Amplitude');
xlabel('frequency (Hz)');
axis([500 2000 0 1]);
y=fft(tone_all);
figure(2)
plot (abs(y))
title('FFT magnitude spectrum');xlabel('frequency');ylabel('magnitude')
figure(3)
plot(tone)
title('Dialed signal');xlabel('time');ylabel('magnitude')
maxAmp(tone);
figure(5)
[c, f, T]=specgram(y); %will return short time fourier transform
mesh(T, f, abs(c));
axis([1 fs 1 T ]);
view(150, 50);
title('SPECTROGRAM MAGNITUDE OF DIALED KEYS');
```



```
ylabel('frequency');
xlabel('time');
zlabel('spectogram magnitude');
```

- First, we have set the sampling frequency as 8000 as given to us in the project details.
- Then we defined the time domain with separation of 1 unit.
- Then we assigned x a zero vector using **zeros** function. When this x will contains a value of 1 it means our encoding is successful.
- Then, we defined different keys and stored them in dtmf.keys.
- Then we defined our frequencies and stored them in a matrix using **ones** function.
- Then we extracted the latest key pressed from keyNames vector because everytime a button is pressed, the key will added to the last index and then this function is called.
- Then we will search this key in the keys defined before using **find** function and assigned a row and column according to the frequencies associated with the key pressed.
- Then we generate a sum of sinusoidal signals of the two frequencies extracted before.
- Then, we will produce a sound of this signal using **soundsc** signal and then add it to the tone_all vector will a silence or gap of 400 zero values which almost becomes 0.25 seconds.
- Then we will plot this sinusoidal signal in the first graph in time domain.
- Then we will defined A_K using the equation:

$$A_K = \frac{2}{N}\sqrt{|X(k)|^2}$$

- Then we will define the frequency axis and then plot the fourier transform of our encoded signal in the second graph.
- Then we have generated a number of figures showing different properties of our encoded signal.
- First we have generated fast fourier transform of our tone_all which contains all the tones and then plot its magnitude.
- Then, we have plotted the encoded signal in time domain in the second figure.
- Then we called maxAmp function which detects whether a peak amplitude exists and then plotted the third figure.
- For the fourth figure, we have used **specgram** function to have different parameters to have a 3D view. In order to print the 3D figure, we have used the **mesh** function.
- Then, we labeled the axis of our 3D figure using xlabel, ylabel, and zlabel.

DECODING THE SIGNAL USING BANDPASS FILTERS

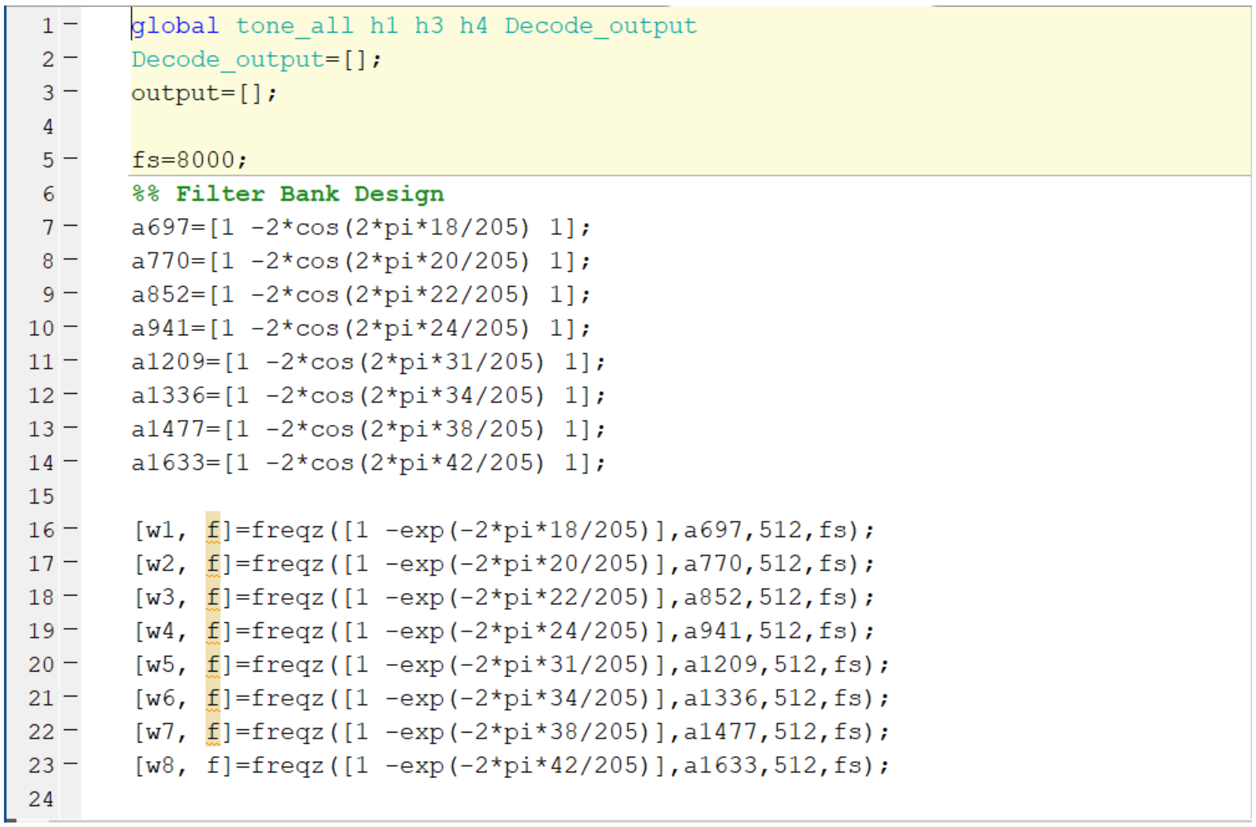
Code Files: **button_Decode.m**

Functions Used:

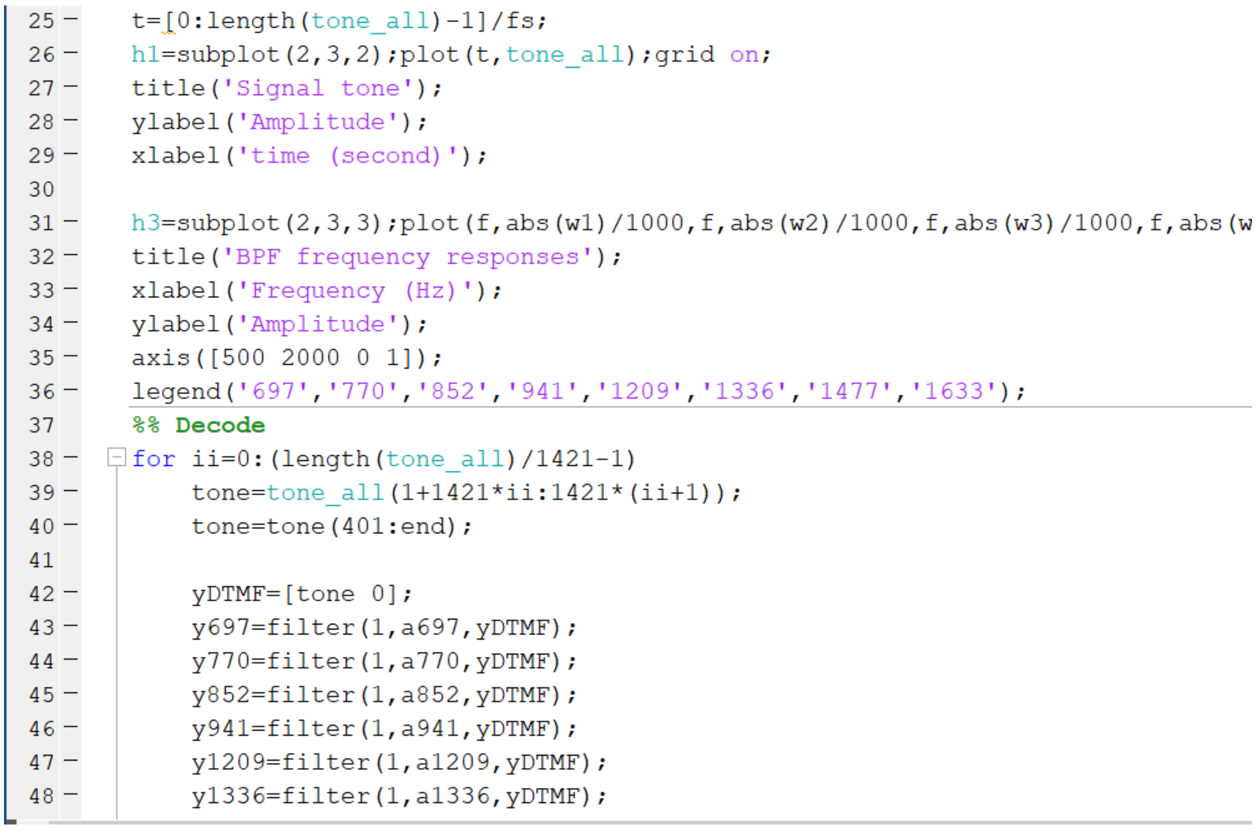
Cos, freqz, exp, length, subplot, plot, xlabel, ylabel, title, axis, legend, filter, sqrt, sum, find, stem, set, soundsc, pause

In this file, we decoded the signal we got from DTMF tone generator using bandpass filters. First, we desgined eight bandpass filters and then used them to isolate individual frequency components. Then, we used different cases with switch staement to find the digits associated with those frequencies and then we have our decoded digits which are the same digits we entered using our keypad.

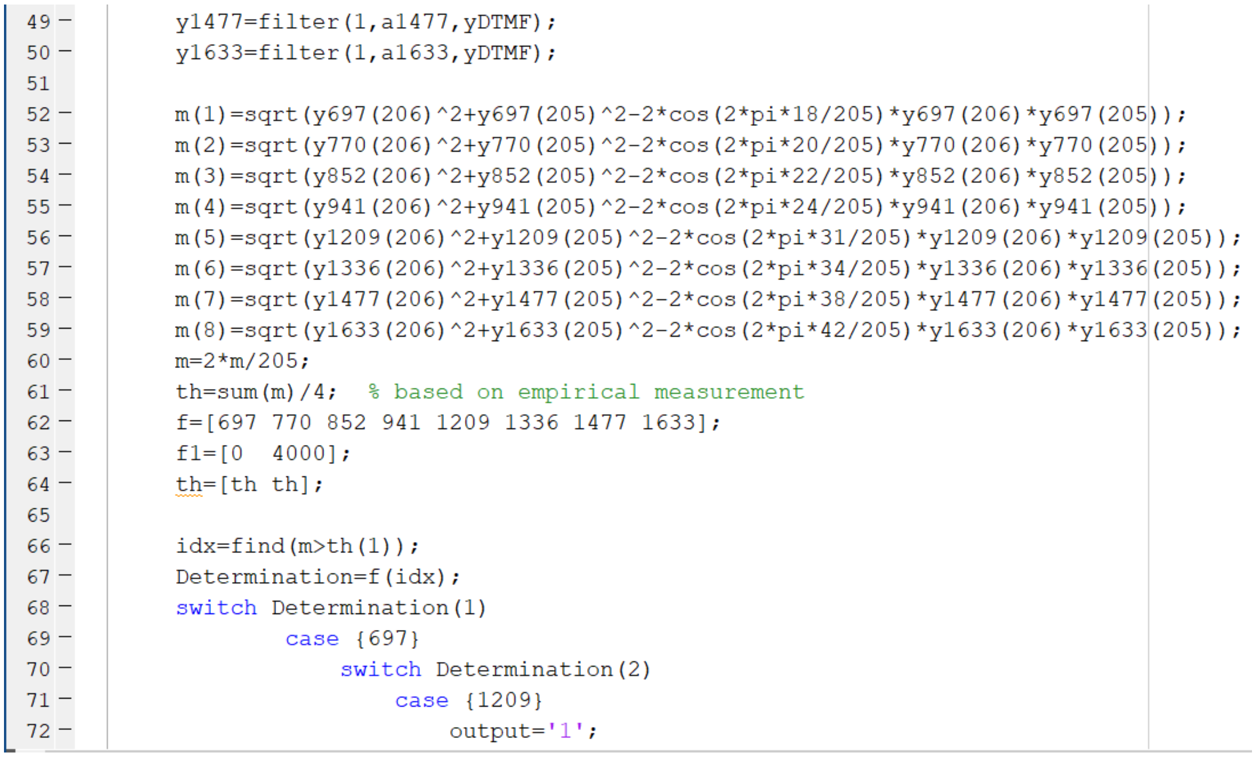
Code:



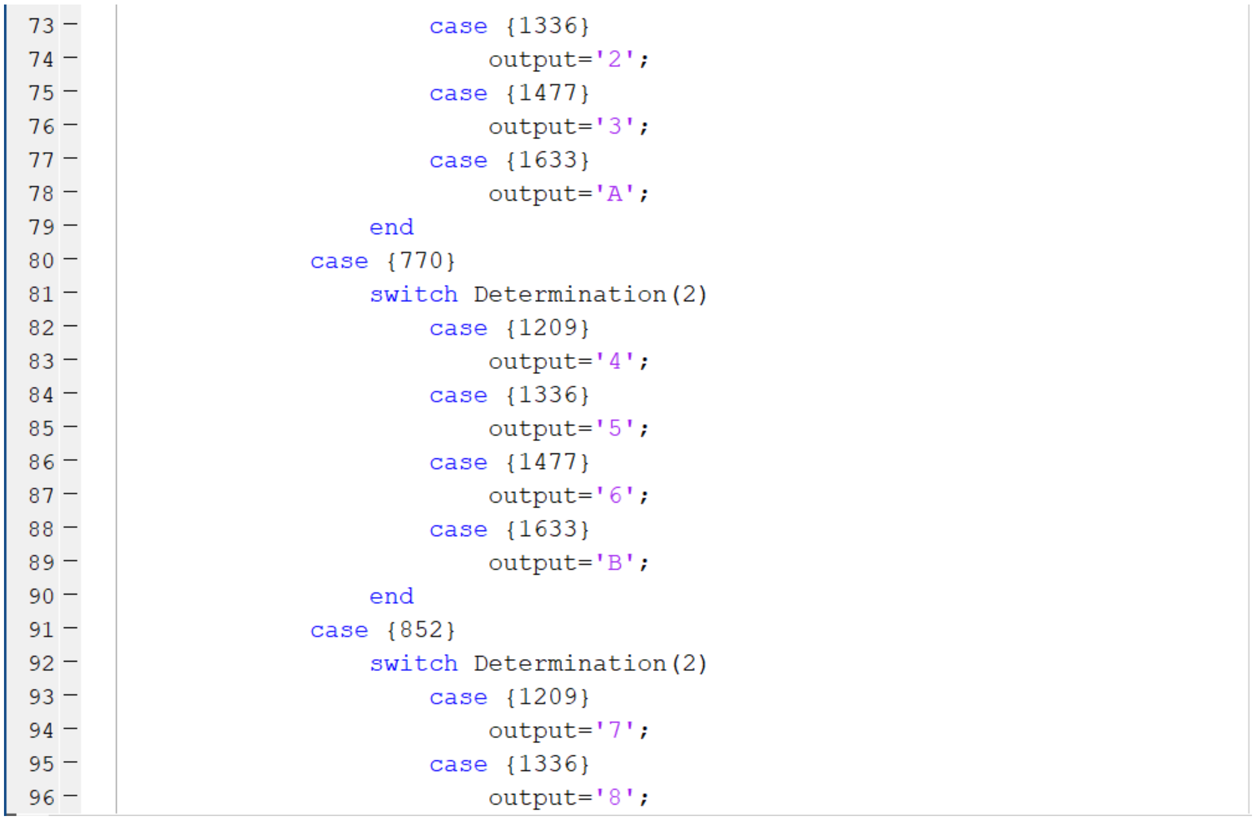
```matlab
global tone_all h1 h3 h4 Decode_output
Decode_output=[];
output=[];

fs=8000;
%% Filter Bank Design
a697=[1 -2*cos(2*pi*18/205) 1];
a770=[1 -2*cos(2*pi*20/205) 1];
a852=[1 -2*cos(2*pi*22/205) 1];
a941=[1 -2*cos(2*pi*24/205) 1];
a1209=[1 -2*cos(2*pi*31/205) 1];
a1336=[1 -2*cos(2*pi*34/205) 1];
a1477=[1 -2*cos(2*pi*38/205) 1];
a1633=[1 -2*cos(2*pi*42/205) 1];

[w1, f]=freqz([1 -exp(-2*pi*18/205)],a697,512,fs);
[w2, f]=freqz([1 -exp(-2*pi*20/205)],a770,512,fs);
[w3, f]=freqz([1 -exp(-2*pi*22/205)],a852,512,fs);
[w4, f]=freqz([1 -exp(-2*pi*24/205)],a941,512,fs);
[w5, f]=freqz([1 -exp(-2*pi*31/205)],a1209,512,fs);
[w6, f]=freqz([1 -exp(-2*pi*34/205)],a1336,512,fs);
[w7, f]=freqz([1 -exp(-2*pi*38/205)],a1477,512,fs);
[w8, f]=freqz([1 -exp(-2*pi*42/205)],a1633,512,fs);

```



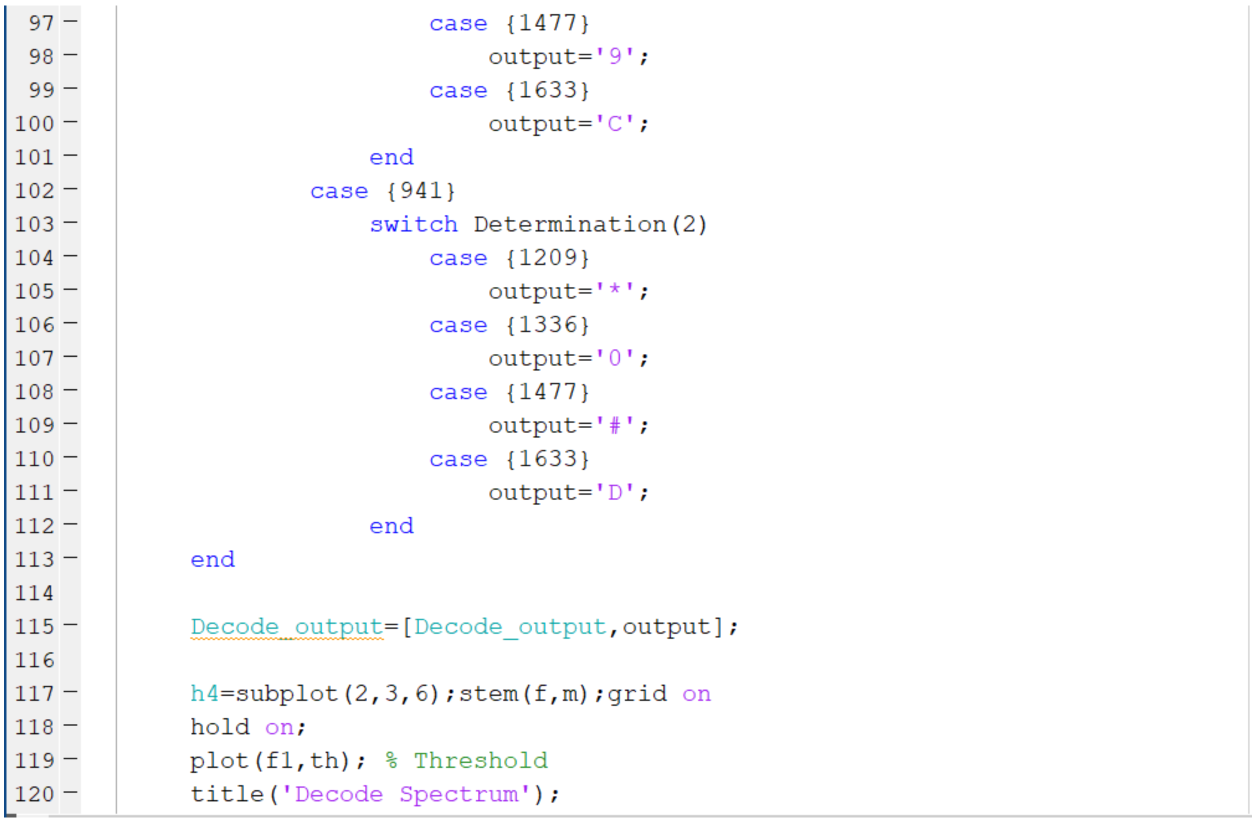
```matlab
t=[0:length(tone_all)-1]/fs;
h1=subplot(2,3,2);plot(t,tone_all);grid on;
title('Signal tone');
ylabel('Amplitude');
xlabel('time (second)');

h3=subplot(2,3,3);plot(f,abs(w1)/1000,f,abs(w2)/1000,f,abs(w3)/1000,f,abs(w
title('BPF frequency responses');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
axis([500 2000 0 1]);
legend('697','770','852','941','1209','1336','1477','1633');
%% Decode
for ii=0:(length(tone_all)/1421-1)
    tone=tone_all(1+1421*ii:1421*(ii+1));
    tone=tone(401:end);

    yDTMF=[tone 0];
    y697=filter(1,a697,yDTMF);
    y770=filter(1,a770,yDTMF);
    y852=filter(1,a852,yDTMF);
    y941=filter(1,a941,yDTMF);
    y1209=filter(1,a1209,yDTMF);
    y1336=filter(1,a1336,yDTMF);
```



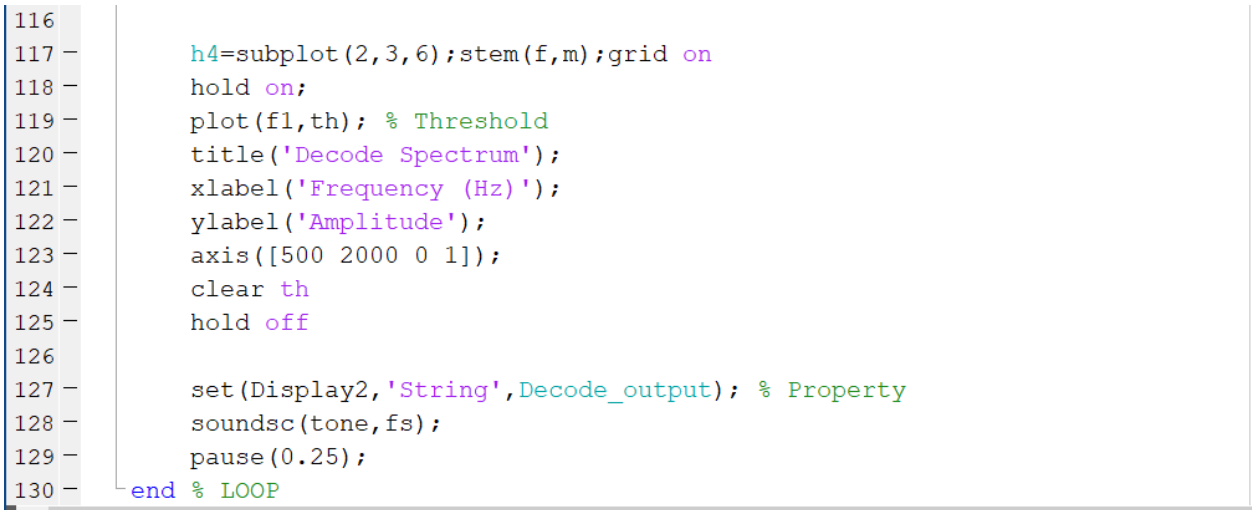
```
y1477=filter(1,a1477,yDTMF);
y1633=filter(1,a1633,yDTMF);

m(1)=sqrt(y697(206)^2+y697(205)^2-2*cos(2*pi*18/205)*y697(206)*y697(205));
m(2)=sqrt(y770(206)^2+y770(205)^2-2*cos(2*pi*20/205)*y770(206)*y770(205));
m(3)=sqrt(y852(206)^2+y852(205)^2-2*cos(2*pi*22/205)*y852(206)*y852(205));
m(4)=sqrt(y941(206)^2+y941(205)^2-2*cos(2*pi*24/205)*y941(206)*y941(205));
m(5)=sqrt(y1209(206)^2+y1209(205)^2-2*cos(2*pi*31/205)*y1209(206)*y1209(205));
m(6)=sqrt(y1336(206)^2+y1336(205)^2-2*cos(2*pi*34/205)*y1336(206)*y1336(205));
m(7)=sqrt(y1477(206)^2+y1477(205)^2-2*cos(2*pi*38/205)*y1477(206)*y1477(205));
m(8)=sqrt(y1633(206)^2+y1633(205)^2-2*cos(2*pi*42/205)*y1633(206)*y1633(205));
m=2*m/205;
th=sum(m)/4;  % based on empirical measurement
f=[697 770 852 941 1209 1336 1477 1633];
f1=[0  4000];
th=[th th];

idx=find(m>th(1));
Determination=f(idx);
switch Determination(1)
        case {697}
            switch Determination(2)
                case {1209}
                    output='1';
```



```
                case {1336}
                    output='2';
                case {1477}
                    output='3';
                case {1633}
                    output='A';
            end
        case {770}
            switch Determination(2)
                case {1209}
                    output='4';
                case {1336}
                    output='5';
                case {1477}
                    output='6';
                case {1633}
                    output='B';
            end
        case {852}
            switch Determination(2)
                case {1209}
                    output='7';
                case {1336}
                    output='8';
```



```
                    case {1477}
                        output='9';
                    case {1633}
                        output='C';
                end
            case {941}
                switch Determination(2)
                    case {1209}
                        output='*';
                    case {1336}
                        output='0';
                    case {1477}
                        output='#';
                    case {1633}
                        output='D';
                end
    end

    Decode_output=[Decode_output,output];

    h4=subplot(2,3,6);stem(f,m);grid on
    hold on;
    plot(f1,th); % Threshold
    title('Decode Spectrum');
```



```

    h4=subplot(2,3,6);stem(f,m);grid on
    hold on;
    plot(f1,th); % Threshold
    title('Decode Spectrum');
    xlabel('Frequency (Hz)');
    ylabel('Amplitude');
    axis([500 2000 0 1]);
    clear th
    hold off

    set(Display2,'String',Decode_output); % Property
    soundsc(tone,fs);
    pause(0.25);
end % LOOP
```

- First, we have initialized decode_output and output with an empty array. Output will contain the key pressed and then it will be added to the decode output.
- Now, we will design 8 bandpass filters which will pass eight desired frequencies only.
- For the purpose of this, first we created eight frequencies from 697 to 1633 Hz. To have their maximum amplitude as 1, we made a vector whose first and last value is 1 and in between them lies the sinusoidal signal of the desired frequency with k defined as :

$$k = \frac{f}{f_s} \times N$$

- Then we designed eight bandpass filters with their center frequencies lying from w1 to w8 using the **freqz** command.
- Then we defined the time axis and plotted the signal received to be decoded in time domain in the first graph.
- Then, we plotted the frequency responses of our bandpass filters in frequency domain in third graph (h3).
- We defined its axis, labelled them and then defined the legends to see which frequency can passed through the filter when a key is pressed.
- After designing the filter, our next goal is to decode the signal using this filter.
- For the purpose of this, we used a for loop to decode each tone one by one.
- As our received signal is very large, we have limited it in our for loop.
- Then, we extracted our desired tone and further limit the axis of our tone to easily decode it.
- Then using **filter** function, we filtered our received tone with the bandpass filter described by eight filters we defined above and maximum amplitude, and stored the filtered data in y697 to y1633.
- Then we defined our transfer functions for the eight filters using Goertzel Equation as follows:

$$H_K(z) = \frac{X_K(z)}{X(z)} = \frac{1}{1 - 2\cos\left(\frac{2\pi k}{N}\right) z^{-1} + z^{-2}}$$

- After defining the transfer functions, we limited and aligned it by multiplying with k.
- Then, we took the average of our transfer function and stored it in th as follows:

$$th = \frac{sum(H_K)}{4}$$

- Then, before using switch staement we have to make the axis of th and f align with the signal received.
- Now, using **find** function whenever a key is pressed, the transfer function will be larger than its average value and we will jump into switch. However, if no key is pressed the transfer function will contain nothing and it will be lower than th therefore it will skip the switch staements.
- In the switch statement, we used the corresponding frequencies as row and column pointers to determine the DTMF code from the keypad and store the key pressed in output.
- Then, we added our key pressed in the vector of decoded output which contains all the decoded digits.
- Then, we plotted our decoded spectrum in the fourth figure.

- Then we cleared th, produce the sound of that single tone, paused for 0.25 seconds and printed the decoded digit on UI and moved on to next tone.
- With this method, we have decoded all the digits.

DETERMINING THE PEAK USING MAXAMP

Code Files: **MaxAmp.m**

Functions Used:

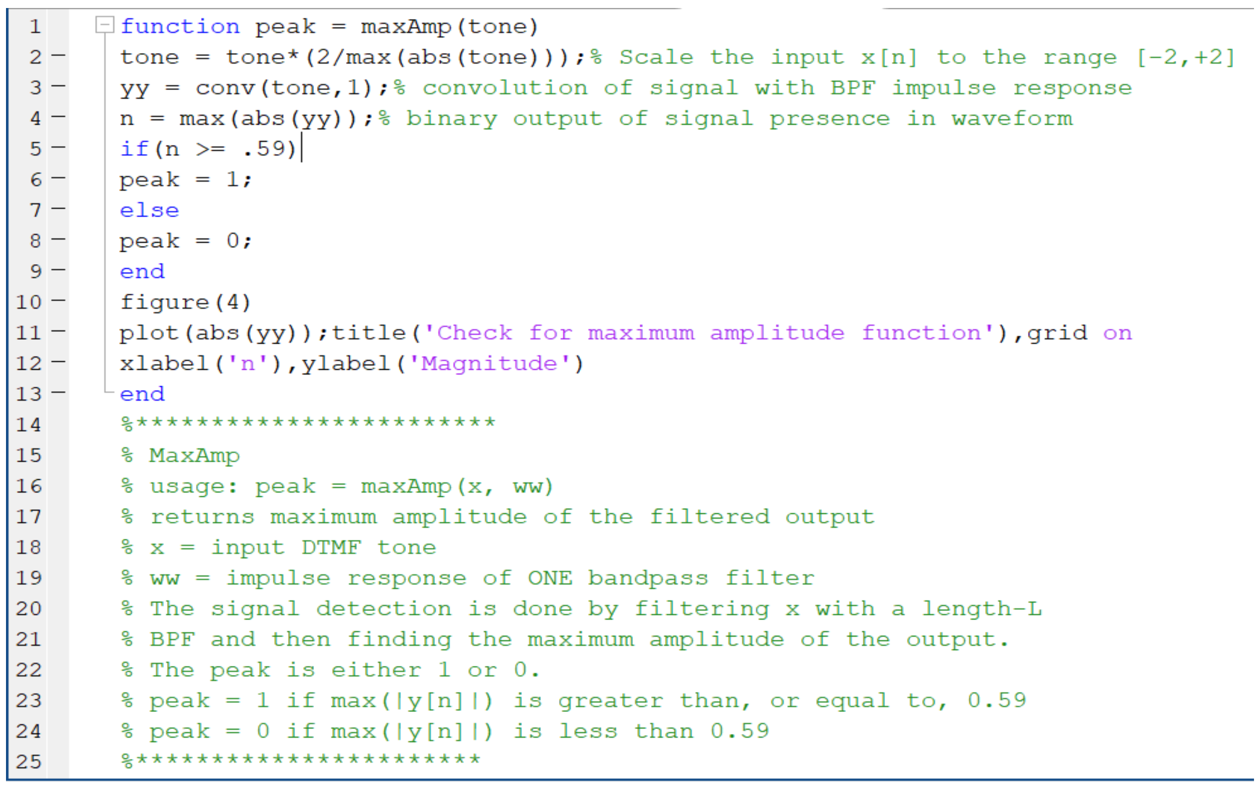
Abs, conv, max, title, xlabel, ylabel, figure

This function will return the maximum amplitude of the filtered output. It takes input as DTMF tone and the impulse response of the bandpass filter. The signal is detected by filtering input tone with length of bandpass filter and then finding the maximum amplitude of the output. The peak is either 0 or 1.

$$peak = 1 \; if \; \max(|y_n|) \geq 0.59$$

$$peak = 0 \; if \; \max(|y_n|) < 0.59$$

Code:



```matlab
function peak = maxAmp(tone)
tone = tone*(2/max(abs(tone)));% Scale the input x[n] to the range [-2,+2]
yy = conv(tone,1);% convolution of signal with BPF impulse response
n = max(abs(yy));% binary output of signal presence in waveform
if(n >= .59)
peak = 1;
else
peak = 0;
end
figure(4)
plot(abs(yy));title('Check for maximum amplitude function'),grid on
xlabel('n'),ylabel('Magnitude')
end
%************************
% MaxAmp
% usage: peak = maxAmp(x, ww)
% returns maximum amplitude of the filtered output
% x = input DTMF tone
% ww = impulse response of ONE bandpass filter
% The signal detection is done by filtering x with a length-L
% BPF and then finding the maximum amplitude of the output.
% The peak is either 1 or 0.
% peak = 1 if max(|y[n]|) is greater than, or equal to, 0.59
% peak = 0 if max(|y[n]|) is less than 0.59
%***********************
```

- Prior to filtering and scoring, make sure that the input signal x[n] is normalized to the range [2; +2]. With this scaling the two sinusoids that make up x[n] should each have amplitudes of approximately 1.01. Therefore the scoring threshold of 0.59 corresponds to a 59% level for detecting the presence of one sinusoid.
- Then, we convoluted the input tone with the impulse response of our bandpass filter which is 1 in this case.
- Then we checked whether the signal is present in waveform and then stored the binary output in n.
- Now, if n is greater than or equal to 0.59 peak is 1 otherwise peak is zero.
- Finally we plotted the convoluted signal in time domain.

RESULTS AND ACHIEVEMENTS

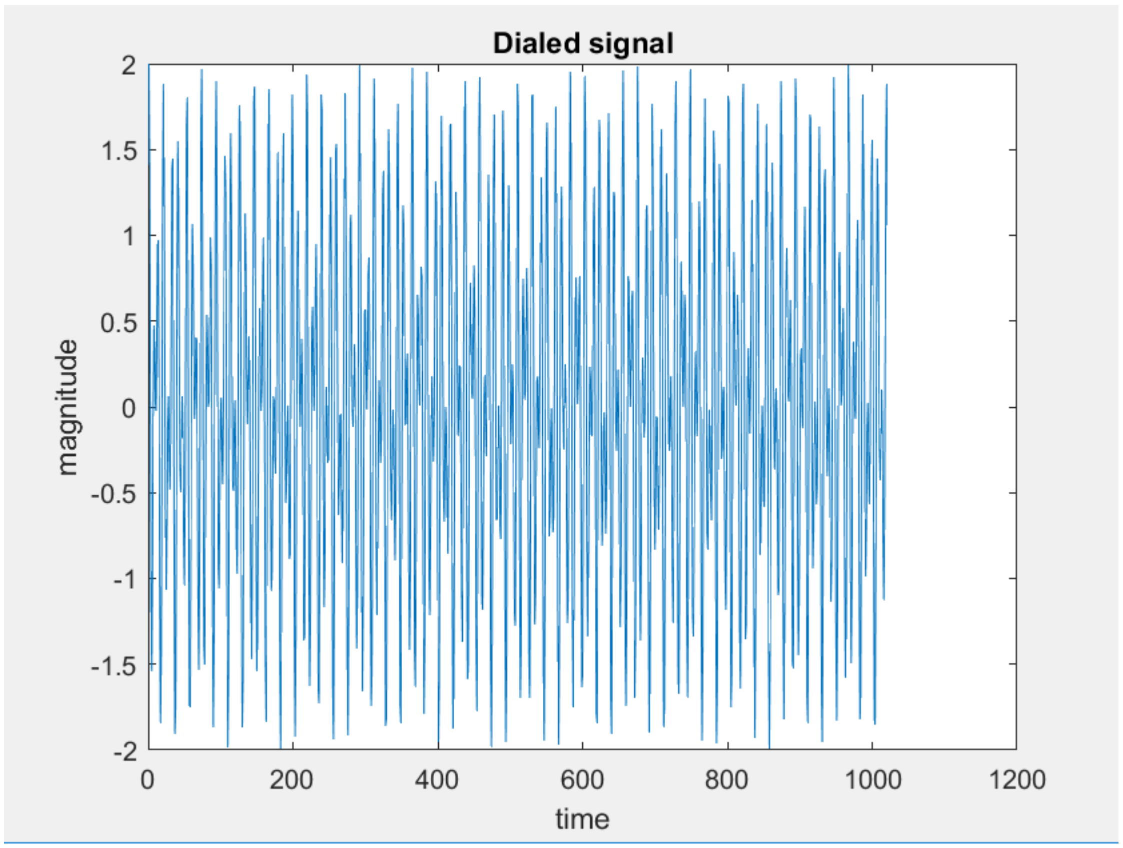
Let us have a demo in which we will plots all the graphs after encoding and decoding.

We will enter a phone number using our keypad and then decode it and plot different figures.

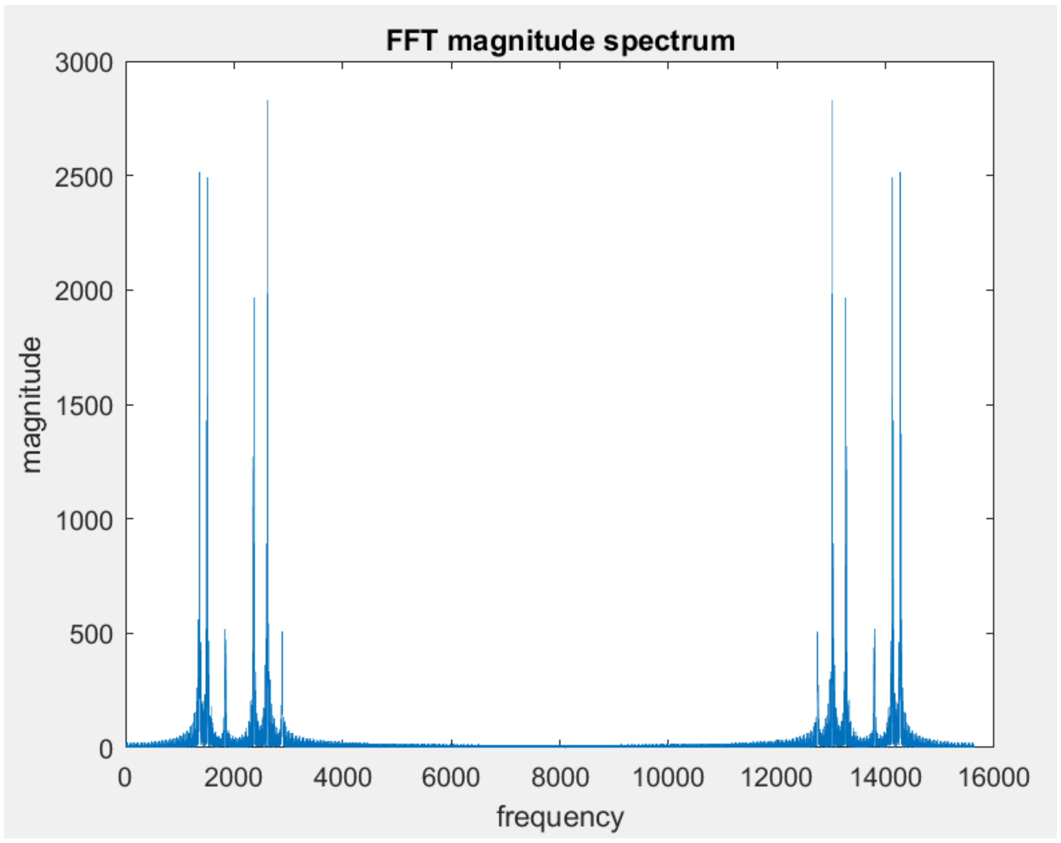
The phone number we have chosen is 03155212454.

When we will enter this phone number using keypad, the system will encode the signal comprising of different frequencies.

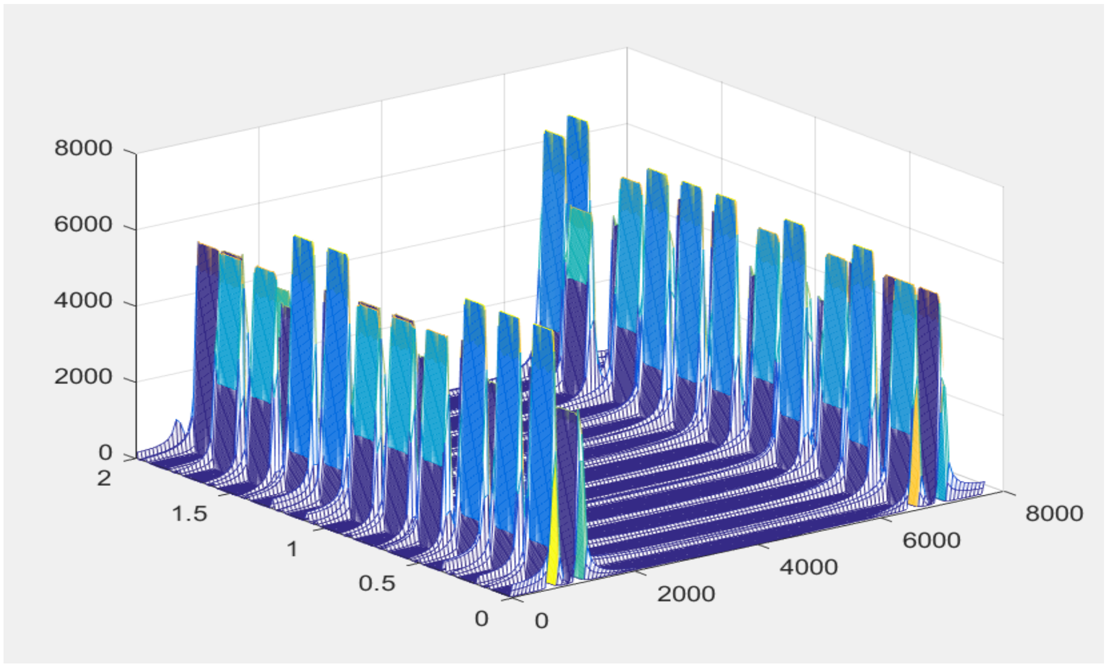
The plot of our encoded signal of the last key pressed in time domain is as follows:



The encoded signal in frequency domain is as follows:

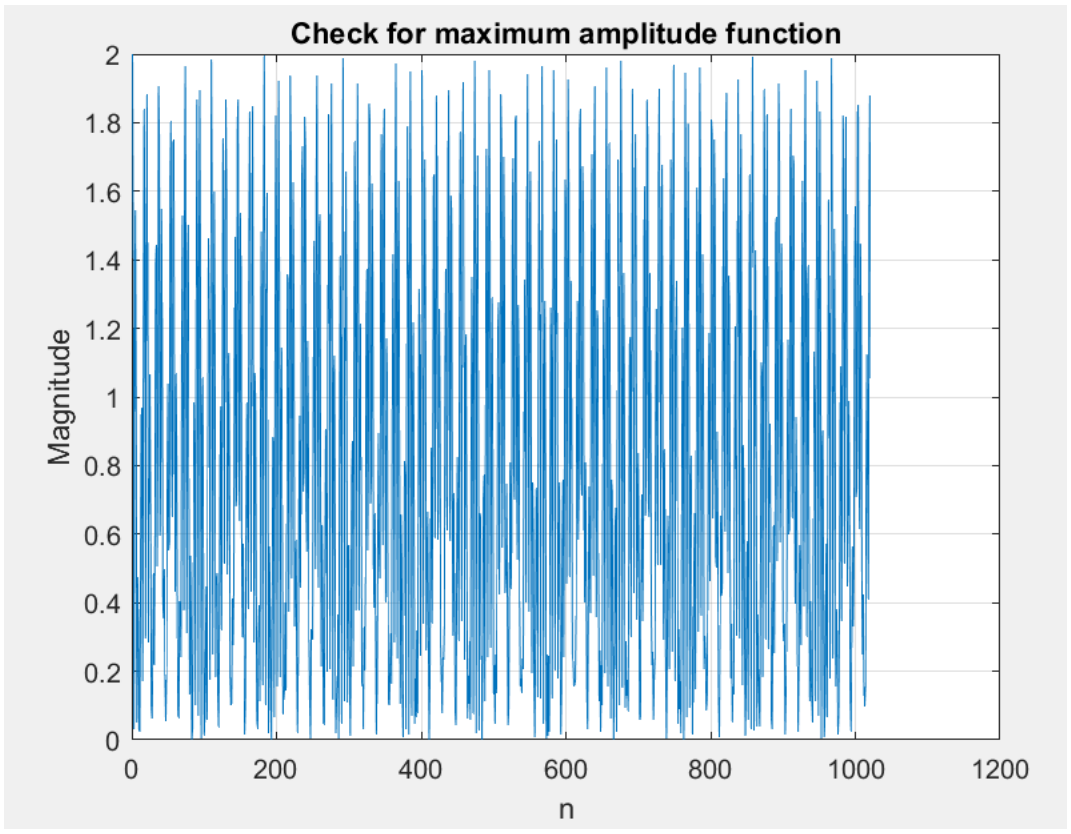


The short time fourier transform of the encoded signal can be more visualized by a 3D graph as follows:

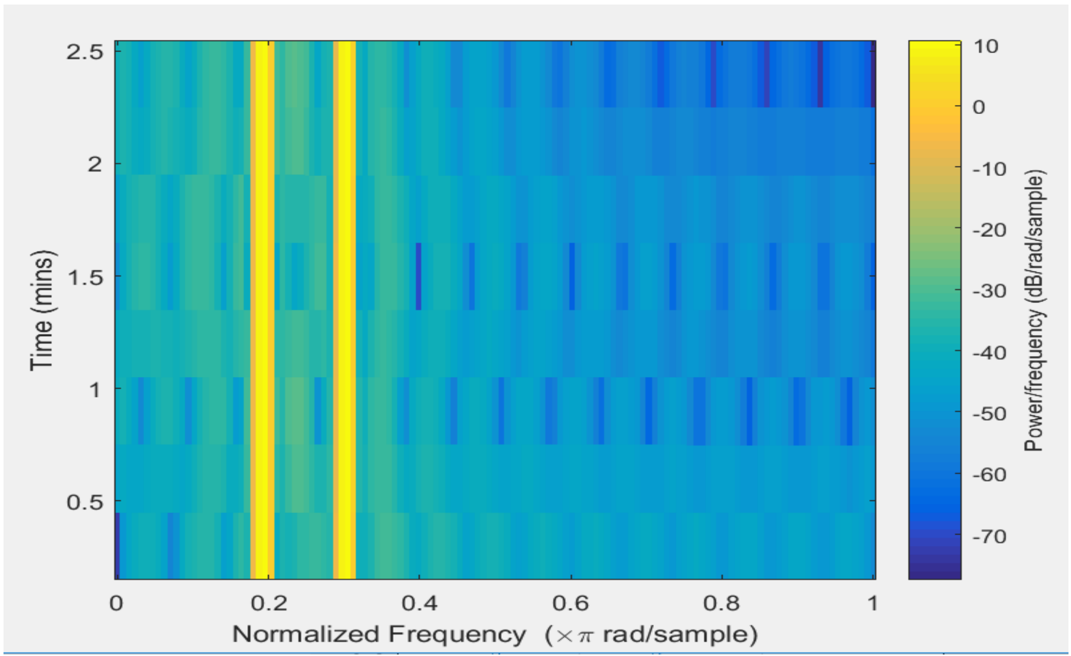


All the frequencies contributing to the digits pressed are respresented beautifully in this 3D graph.

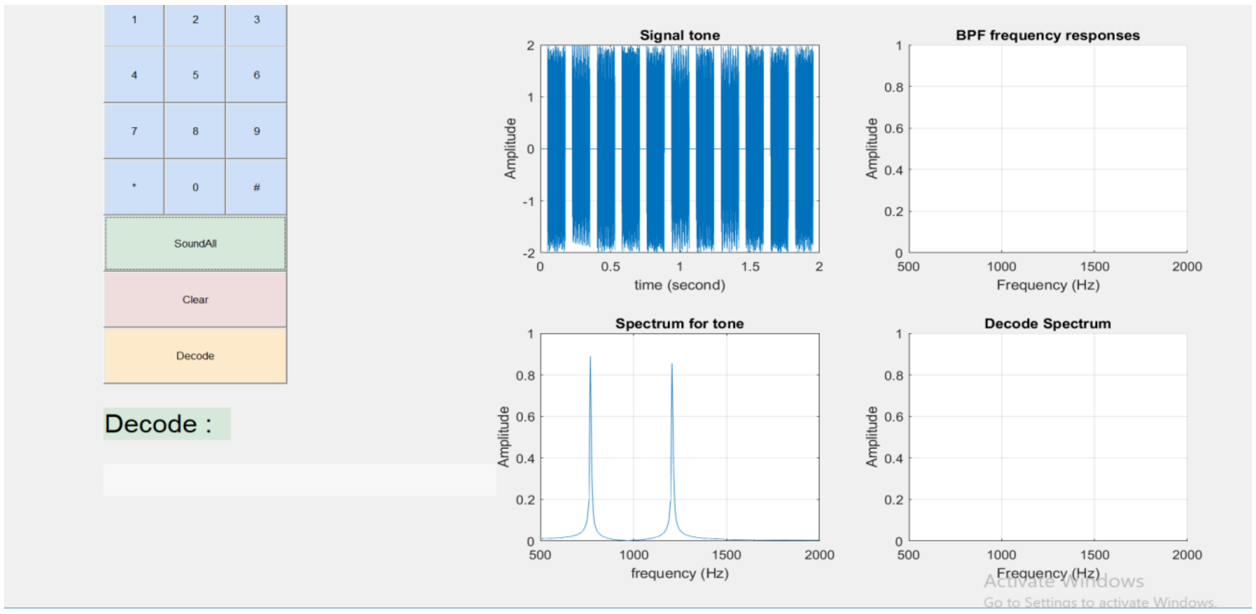
Then to check for maximum amplitude of the encoded signal is represented as follows:



The magnitude of the short fourier transform of encoded signal is plotted as follows:



Now, as all the digits are entered we will press the **sound all** button. It will produce the sound produced by our encoded signal and the entire encoded signal is represented as follows:

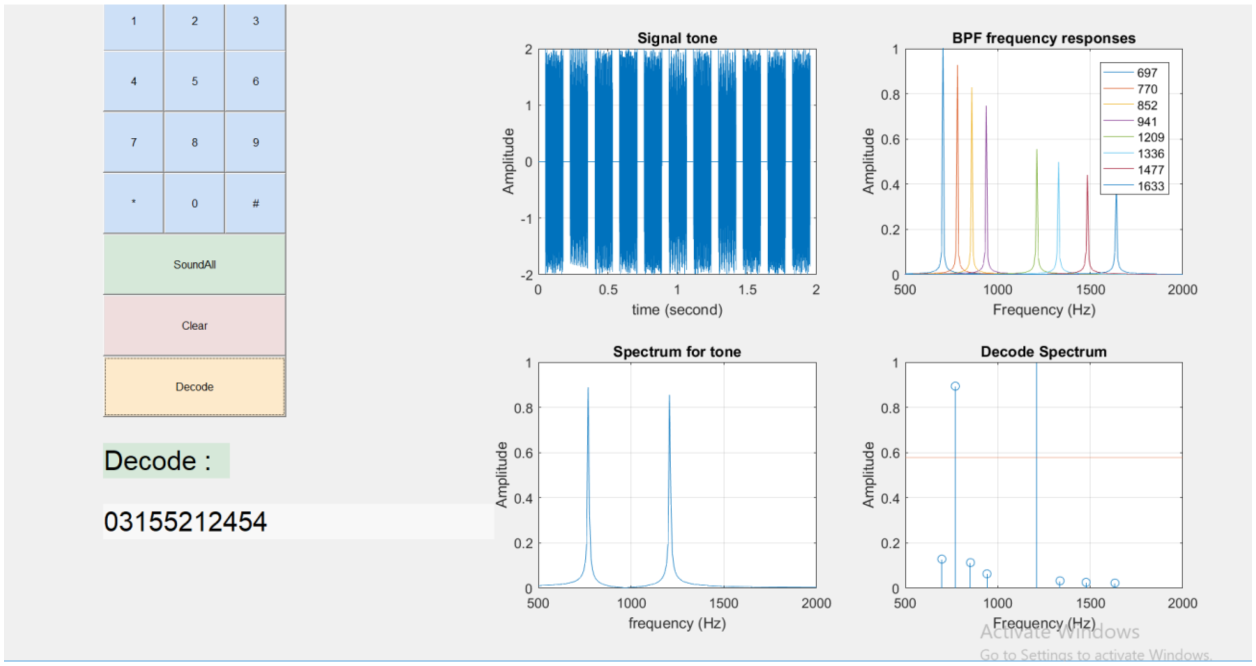


The straight lines in encoded signal represents the silence period or the pause taken between any two tones.

The graphs on the right side are empty because we haven't decoded our signal yet.

To decode the signal, we will press the **decode** button and it will decode this encoded signal back into digits and the same phone number will be printed on the UI screen.

After pressing the **decode** button, we have our UI window as follows:



The Decode spectrum changes tone by tone and will be shown in live demo.

WORK DIVISIONS

The work is divided into group members as follows:

Ch. Muhammad Shaheer Yasir:

He was responsible for making GUI part of the project. He used different functionalities of GUI offered by MATLAB to create the UI window and then designed the keypad. He positioned all the graphs and buttons on keypads. He also created the functionality of buttons from 0 to 9 , # and *. He also checked whether maximum amplitude is present in the signal and assigned a binary output 0 or 1 to peak.

Muhammad Hamza:

He created the functionality of sound_all and clear buttons. When all the keys are entered, and the signal is encoded. He used different techniques like for loops to produce a unified sound created from all the keys pressed. He set all the vectors and tones to empty in the functionality of clear button and also cleared all the graphs present on the screen. He also helped in creating switch statement which decodes the digits.

Farheen Gul:

As soon as the key is pressed, dtmf tone generator created by her starts to create signals of different frequencies. She developed a clear understanding of how the frequencies are associated with the keys pressed and then did all the encoding process using different techniques like generating the sum of sinusoidal signals of the two frequencies (one high and one low for a single key pressed). The encoded signal created by her then produced a sound of multiple frequency.

Shehroze Amir:

Most of the decoding process was done by him. First, he desgined eight bandpass filters and then used those filters to first produced a filtered output. Then, he created the transfer function using Goertzal Equation. He created threshold function which prevents the program for going into switch staement when no key is pressed. After having coressponding frequencies he decoded the digits pressed using switch statement and printed them on the screen.

Applications and Advantages

This system can be easily constructed with an acheivement of fast response. It can control electrical devices wirelessly using low power consumption and less price. It can be used in robotic microcontrollers. It can mainly be used at the telephone switching centers to detect the dialed numbers. They can be used in terrestrial stations to switch on and off the remote transmitters. This system can be used in a lot of industrial applications.

Recommendations and Future Work

Following future work is recommended in this system:

- The user interface of the system can be improved.
- Most of the values our system contains are constant. We can make them variable and gives user the choice to enter different values and have different amazing and beautiful signals.
- Our system contains both encoding and decoding and the same encoded signal is then decoded using different techniques. However, we can separate the both parts and gives user the choice whether he/she wants to encode or decode.

Conclusions

Encoding the signals and then decoding them using bandpass filters included almost all the concepts we have studied in our signals and systems course. We have implemented all the concepts we have learned in our class to make this useful and practical project which can be used in a number of applications. This project enhanced our learning of how to generate signals and then how to deal with them using different techniques. It helped to embedd in us the skills of team work, collaboration, and the ability of analytical and critical thinking. We have faced many difficulties during the project but we helped to overcome them using collaborative thinking. With this, we are all set to implement our concepts of signals and systems in any field of practical and industrial applications.

Citations and References

- https://www.efxkits.us/dtmf-dual-tone-multi-frequency-technology-working-applications/
- http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.402.9538
- https://repository.najah.edu/handle/20.500.11888/12072