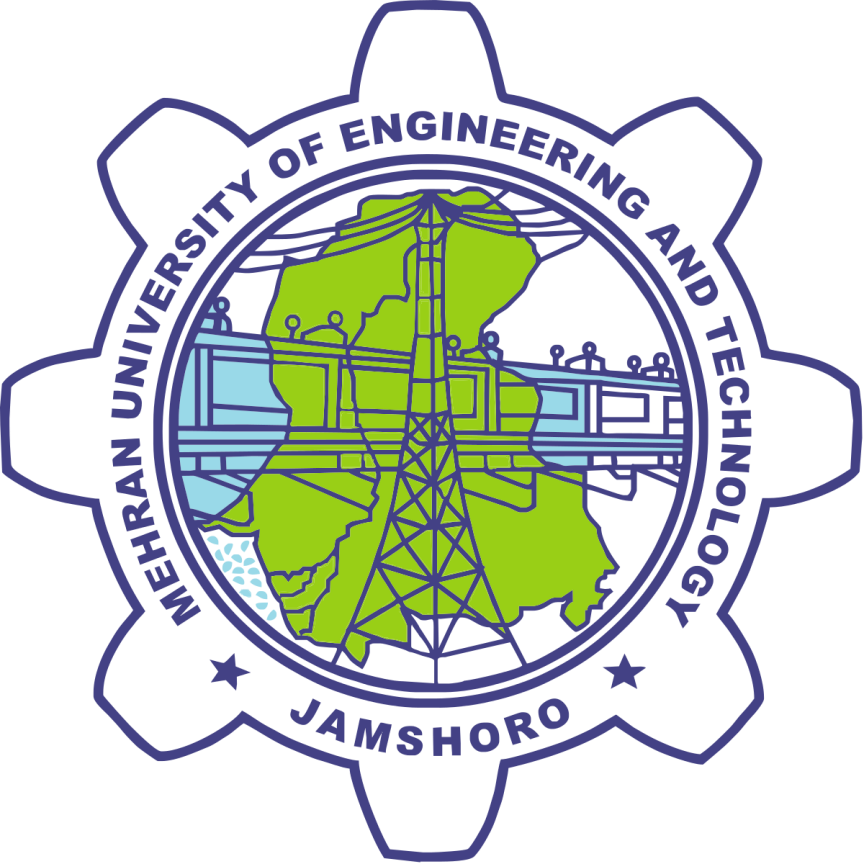
**Mehran University of Engineering and Technology, Jamshoro**

**Department of Computer System Engineering(4th semester, 2nd year)**



**PROJECT REPORT: DTMF Decoder**

**Name:**

**Roll no:**

**Subject: Signals and Systems**

**Project title: DTMF (Dual-Tone Multi-Frequency) Decoder using MATLAB**

**Submitted to: Mr. Umair Khorie**

**Introduction**

This project focuses on the design and implementation of a DTMF (Dual-Tone Multi-Frequency) decoder using MATLAB. DTMF is a signaling method used for telecommunication signaling over analog telephone lines, in voice-frequency bands between telephone handsets and other communication devices. This system allows users to identify pressed keys on a keypad based on unique frequency tones.

The main goal is to develop a MATLAB GUI that takes DTMF signals, processes them using Chebyshev filters, and decodes the input to display the corresponding key press.

**Objective**

The objective of this project is to:

* Design a MATLAB GUI to simulate and decode DTMF signals.
* Apply filtering and signal processing techniques to isolate frequency pairs.
* Decode and display the corresponding key based on the input signal.

**Tools and Techniques**

* MATLAB: A high-performance language used for technical computing.
* Chebyshev Filters: Applied to filter the high and low frequencies in DTMF signals.
* Fast Fourier Transform (FFT): Used to convert the signal into its frequency domain for further analysis.

**Methodology**

**DTMF Signal Generation**

A DTMF signal consists of two tones, one from a low-frequency group and the other from a high-frequency group. Each button on a telephone keypad corresponds to a combination of one low-frequency and one high-frequency signal.

**DTMF Signal Processing**

The signal is processed by passing it through two Chebyshev filters:

* **Low-Pass Filter:** To isolate the low-frequency components of the signal.
* **High-Pass Filter:** To isolate the high-frequency components of the signal.

**Decoding Mechanism**

The frequency components of the processed signal are analyzed using FFT. The system then determines which button was pressed based on the frequency pairs.

**DTMF Frequency Mapping Table**

The following table maps the frequency ranges to the respective Dual-Tone Multi-Frequency (DTMF) key presses. The frequencies j and k represent the low and high frequency components, respectively. This mapping helps determine which key was pressed based on the frequency components detected during signal processing. The table is structured as follows:

|  |  |  |
| --- | --- | --- |
| **Low Frequency (j)** | **High Frequency (k)** | **Key Pressed** |
| ≤ 732.59 Hz | ≤ 1270.91 Hz | 1 |
| ≤ 732.59 Hz | ≤ 1404.73 Hz | 2 |
| ≤ 732.59 Hz | ≤ 1553.04 Hz | 3 |
| ≤ 732.59 Hz | > 1553.05 Hz | A |
| ≤ 809.96 Hz | ≤ 1270.91 Hz | 4 |
| ≤ 809.96 Hz | ≤ 1404.73 Hz | 5 |
| ≤ 809.96 Hz | ≤ 1553.04 Hz | 6 |
| ≤ 809.96 Hz | > 1553.05 Hz | B |
| ≤ 895.39 Hz | ≤ 1270.91 Hz | 7 |
| ≤ 895.39 Hz | ≤ 1404.73 Hz | 8 |
| ≤ 895.39 Hz | ≤ 1553.04 Hz | 9 |
| ≤ 895.39 Hz | > 1553.05 Hz | C |
| > 895.40 Hz | ≤ 1270.91 Hz | \* |
| > 895.40 Hz | ≤ 1404.73 Hz | 0 |
| > 895.40 Hz | ≤ 1553.04 Hz | # |
| > 895.40 Hz | > 1553.05 Hz | D |

**Interface:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Code Implementation**

Two major MATLAB scripts were used for the project:

**Subdecode.m**

This script processes the DTMF signals, applies Chebyshev filters, and displays filtered low and high-frequency signals along with the FFT results. It identifies the key pressed by comparing the FFT output with the expected frequencies of each DTMF tone.

|  |
| --- |
| axes(handles.fig1); |
| plot(t,y); |
| set(handles.fig1,'XMinorTick','on'); |
| title('DTMF Input');xlabel('Time'); |
| ylabel('Amplitude');grid; |
|  |
| 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.fig2); |
| plot(yfilt1);grid; |
| title('Filtered Low Freq. Signal'); |
| xlabel('Time');ylabel('Amplitude'); |
|  |
| axes(handles.fig3); |
| plot(yfilt2);grid; |
| title('Filtered High Freq. Signal'); |
| xlabel('Time');ylabel('Amplitude'); |
|  |
| hlow=fft(yfilt1,rmain); |
| hmaglow=abs(hlow); |
| axes(handles.fig4); |
| plot(hmaglow(1:rmag)); |
| title('FFT Low Pass');grid; |
| xlabel('Time');ylabel('Amplitude'); |
|  |
| hhigh=fft(yfilt2,rmain); |
| hmaghigh=abs(hhigh); |
| axes(handles.fig5); |
| plot(hmaghigh(1:rmag)); |
| title('FFT High Pass');grid; |
| xlabel('Time');ylabel('Amplitude'); |
|  |
| 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; |
|  |
| if j<=732.59 & k<=1270.91; |
| disp('---> Key Pressed is 1'); |
| elseif j<=732.59 & k<=1404.73; |
| disp('---> Key Pressed is 2'); |
| elseif j<=732.59 & k<=1553.04; |
| disp('---> Key Pressed is 3'); |
| elseif j<=732.59 & k>1553.05; |
| disp('---> Key Pressed is A'); |
| elseif j<=809.96 & k<=1270.91; |
| disp('---> Key Pressed is 4'); |
| elseif j<=809.96 & k<=1404.73; |
| disp('---> Key Pressed is 5'); |
| elseif j<=809.96 & k<=1553.04; |
| disp('---> Key Pressed is 6'); |
| elseif j<=809.96 & k>1553.05; |
| disp('---> Key Pressed is B'); |
| elseif j<=895.39 & k<=1270.91; |
| disp('---> Key Pressed is 7'); |
| elseif j<=895.39 & k<=1404.73; |
| disp('---> Key Pressed is 8'); |
| elseif j<=895.39 & k<=1553.04; |
| disp('---> Key Pressed is 9'); |
| elseif j<=895.39 & k>1553.05; |
| disp('---> Key Pressed is C'); |
| elseif j>895.40 & k<=1270.91; |
| disp('---> Key Pressed is \*'); |
| elseif j>895.40 & k<=1404.73; |
| disp('---> Key Pressed is 0'); |
| elseif j>895.40 & k<=1553.04; |
| disp('---> Key Pressed is #'); |
| elseif j>895.40 & k>1553.05; |
| disp('---> Key Pressed is D'); |
| end |

**Decode.m**

This script creates the GUI that enables user interaction. Users press buttons on a virtual keypad, and the corresponding DTMF signals are generated and processed.

|  |
| --- |
| function varargout = decode(varargin) |
| gui\_Singleton = 1; |
| gui\_State = struct('gui\_Name', mfilename, ... |
| 'gui\_Singleton', gui\_Singleton, ... |
| 'gui\_OpeningFcn', @decode\_OpeningFcn, ... |
| 'gui\_OutputFcn', @decode\_OutputFcn, ... |
| 'gui\_LayoutFcn', [], ... |
| 'gui\_Callback', []); |
| if nargin && isstr(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 |
|  |
| function decode\_OpeningFcn(hObject, eventdata, handles, varargin) |
| handles.output = hObject; |
| guidata(hObject, handles); |
|  |
| function varargout = decode\_OutputFcn(hObject, eventdata, handles) |
| varargout{1} = handles.output; |
|  |
| function b1\_Callback(hObject, eventdata, handles) |
| t=[0:0.000125:.05]; |
| 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) |
| subdecode; |
|  |
| function b2\_Callback(hObject, eventdata, handles) |
| 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) |
| subdecode; |
|  |
| function A\_Callback(hObject, eventdata, handles) |
| 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) |
| subdecode; |
|  |
| function b3\_Callback(hObject, eventdata, handles) |
| 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) |
| subdecode; |
|  |
| function b4\_Callback(hObject, eventdata, handles) |
| 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) |
| subdecode; |
|  |
| function b5\_Callback(hObject, eventdata, handles) |
| 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) |
| subdecode; |
|  |
| function B\_Callback(hObject, eventdata, handles) |
| 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) |
| subdecode; |
|  |
| function b6\_Callback(hObject, eventdata, handles) |
| 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) |
| subdecode; |
|  |
| function b7\_Callback(hObject, eventdata, handles) |
| 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) |
| subdecode; |
|  |
| function b8\_Callback(hObject, eventdata, handles) |
| 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) |
| subdecode; |
|  |
| function C\_Callback(hObject, eventdata, handles) |
| 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) |
| subdecode; |
|  |
| function b9\_Callback(hObject, eventdata, handles) |
| 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) |
| subdecode; |
|  |
| function ba\_Callback(hObject, eventdata, handles) |
| 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) |
| subdecode; |
|  |
| function b0\_Callback(hObject, eventdata, handles) |
| 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) |
| subdecode; |
|  |
| function D\_Callback(hObject, eventdata, handles) |
| 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) |
| subdecode; |
|  |
| function bn\_Callback(hObject, eventdata, handles) |
| 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) |
| subdecode; |
|  |
| function info\_Callback(hObject, eventdata, handles) |
| msgbox('SIGNAL AND SYSTEM PROJECT Prepared by 22CS 032 || 078 Mehran University of Engineering Technology (CSE Department)','Info','warn') |
|  |
| function close\_Callback(hObject, eventdata, handles) |
| close; |
|  |

The code implementation involves plotting the signals and filters using axes() commands, processing the signals through filters, and matching the output to predefined DTMF frequency ranges.

**Results**

After processing the signal using Chebyshev filters and FFT, the system successfully decoded the pressed buttons based on the frequency components. The GUI allows the user to interact by pressing buttons, and the decoded output is displayed correctly.

**Conclusion**

The project successfully demonstrates the use of signal processing techniques to decode DTMF signals. MATLAB's GUI capabilities provide an interactive way to simulate the behavior of a DTMF decoder, making this project a practical and educational tool for understanding signal processing.