DISCRETE ENVELOPE DETECTOR USING MATLAB

SEMESTER PROJECT: 2018

Group No: 4
Submitted by:
Muhammad Shahab(2016353)
Rehan(2016404)
Zeeshan(2016544)

Windows User


بسم الله الرحمن الرحيم

DEDICATION

To Our Parents

Acknowledgement

We are deeply thankful to Almighty Allah, who is highly Merciful and Graceful, who bestowed us with courage and knowledge to complete this study.

We acknowledge and appreciate the scholarly guidance, valuable suggestions, positive criticism and kind advice given to us by **Dr. Shahab Ansari** under whose supervision this project has been completed. His guidance on the discussions and suggestions invoked our thinking process. It generated a great deal of interest in the research work, giving us self-believe and feeling of responsibility.

Contents

Envelop Detector:

Problem Statement:

AM envelope detector MATLAB Consider an envelope detector that is used to detect the message sent in the AM system shown in the examples. The envelope detector as a system is composed of two cascaded systems one that computes the absolute value of the input (implemented with ideal diodes), and a second that low-pass filters its input (implemented with an RC circuit). The following is an implementation of these operations in the discrete time so we can use numeric MATLAB. FIGURE 2.24 Problem 2.18: sample and hold circuit. Let the input to the envelope detector be where P is the minimum of p(nTs) scaled. Use MATLAB to solve numerically this problem.

Objective:

To design a discrete envelop detector on Matlab using Low Pass filter

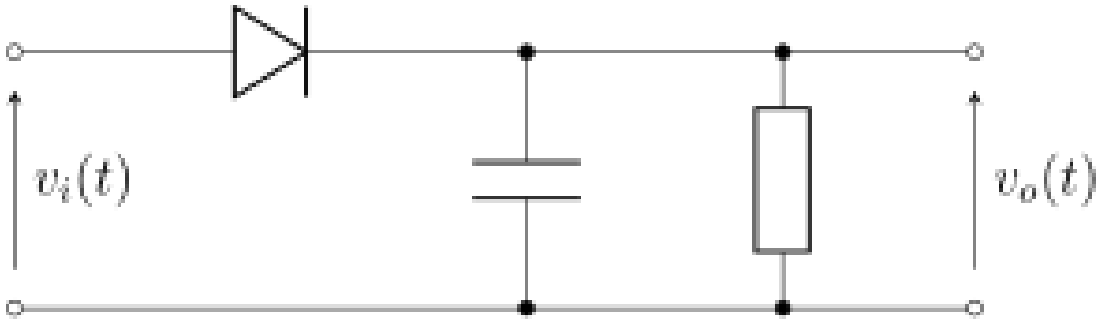
An envelope detector is an electronic circuit that takes a (relatively) high-frequency amplitude modulated signal as input and provides an output which is the envelope of the original signal.

Introduction:

Most practical envelope detectors use either half-wave or full-wave rectification of the signal to convert the AC audio input into a pulsed DC signals. Filtering is then used to smooth the final result. This filtering is rarely perfect and some "ripple" is likely to remain on the envelope follower output, particularly for low frequency inputs such as notes from a bass instrument. Reducing the filter cutoff frequency gives a smoother output, but decreases the high frequency response. Therefore, practical designs must reach a compromise.

Circuit operation:

The capacitor in the circuit above stores charge on the rising edge and releases it slowly through the resistor when the input signal amplitude falls. The diode in series rectifies the incoming signal, allowing current flow only when the positive input terminal is at a higher potential than the negative input terminal.



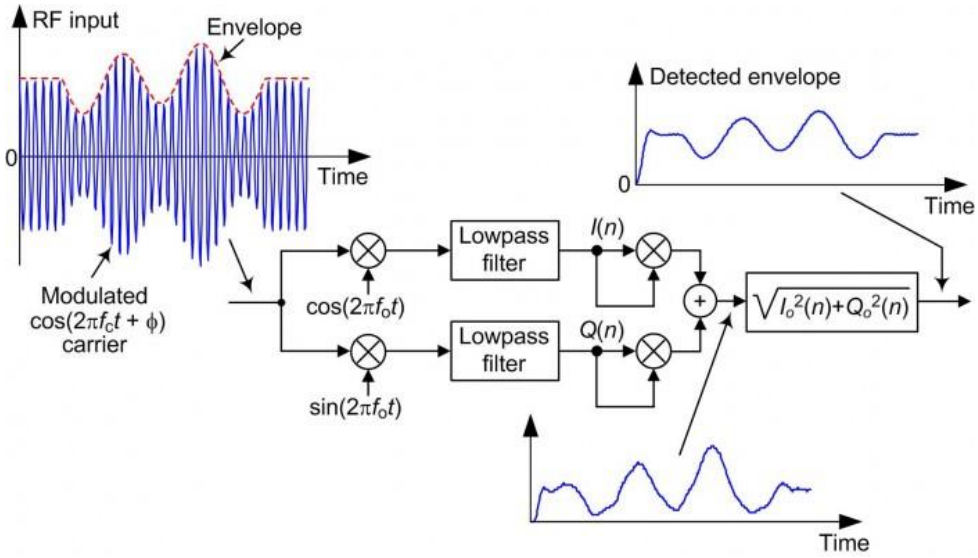
Advantages:

1. **Diode detector:**

The simplest form of envelope detector is the diode detector which is shown above. A diode detector is simply a diode between the input and output of a circuit, connected to a resistor and capacitor in parallel from the output of the circuit to the ground. If the resistor and capacitor are correctly chosen, the output of this circuit should approximate a voltage-shifted version of the original (baseband) signal. A simple filter can then be applied to filter out the DC component.

2. **Precision detector**
   An envelope detector can also be constructed to use a precision rectifier feeding into a low-pass filter.

3. **Demodulation of signals**
   An envelope detector can be used to demodulate a previously modulated signal by removing all high frequency components of the signal. The capacitor and resistor form a low-pass filter to filter out the carrier frequency. Such a device is often used to demodulate AM radio signals because the envelope of the modulated signal is equivalent to the baseband signal.



Drawbacks

The envelope detector has several drawbacks:

- ✓ The input to the detector must be band-pass filtered around the desired signal, or else the detector will simultaneously demodulate several signals.
- ✓ The filtering can be done with a tunable filter or, more practically, a super heterodyne receiver.
- ✓ It is more susceptible to noise than a product detector if the signal is over modulated, distortion will occur.

Most of these drawbacks are relatively minor and are usually acceptable tradeoffs for the simplicity and low cost of using an envelope detector.

Additional Features:

- ✓ Sound recording has been done of GUI and results of recorded sound can be seen on the plot.
- ✓ A recorded sound file is been enveloped and their envelop sounds can be listen by pressing buttons available.

Matlab Code:

```
function varargout = project_2016544(varargin)
% PROJECT_2016544 MATLAB code for project_2016544.fig
%      PROJECT_2016544, by itself, creates a new PROJECT_2016544 or
raises the existing
%      singleton*.
%
%      H = PROJECT_2016544 returns the handle to a new PROJECT_2016544
or the handle to
%      the existing singleton*.
%
%      PROJECT_2016544('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in PROJECT_2016544.M with the given
input arguments.
%
%      PROJECT_2016544('Property','Value',...) creates a new
PROJECT_2016544 or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before project_2016544_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to project_2016544_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 project_2016544

% Last Modified by GUIDE v2.5 03-Dec-2018 15:23:53

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @project_2016544_OpeningFcn, ...
                   'gui_OutputFcn',  @project_2016544_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 project_2016544 is made visible.
function project_2016544_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 project_2016544 (see VARARGIN)

% Choose default command line output for project_2016544
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes project_2016544 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = project_2016544_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)
global N;
N = 100;
global n;
n = 0:N - 1;
global Ts;
Ts = 0.01;
global p;
p=heaviside(n*Ts) - heaviside(n*Ts - 20*Ts) + heaviside(n*Ts - 40*Ts)-
heaviside(n*Ts - 60*Ts);
axes(handles.axes1);
plot(n,p)
title("Given Signal");
grid on;
legend('P*Ts');


% --- 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 N;
N = 100;
global n;
n = 0:N - 1;
global Ts;
Ts = 0.0001;
global x;
global p;
x = cos(2000*pi*n*Ts);
axes(handles.axes2);
plot(n,x)
grid on;
title("Cosine Function");
legend('Cos(200 * PI*n*Ts)');
```

```
% --- 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)
global N;
N = 100;
global n;
n = 0:N - 1;
global Ts;

global x;
global y;
global p;
y = p.*x;
axes(handles.axes3);
plot(n,y)
grid on;
title("Product of 1st and 2nd Signal");
legend('P(nTs)*Cos(200 * PI*n*Ts)');



% --- 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)

global N;
N = 100;
global n;
n = 0:N - 1;
global Ts;

global y;
global u;
u = abs(y);

axes(handles.axes4);

plot(n,u);
grid on;
title("Absoulte value of Signal");
legend('Abs');


% --- 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)
global N;
N = 100;
global n;
n = 0:N - 1;



global u;

axes(handles.axes5);
z = averager(15,u);
plot(n,z);
grid on;
title("Envelope detection and Low Pass Filtering");
legend('LPF');



% --- 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)
global r;
r=audiorecorder(22050,16,1);
record(r);



% --- 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)
global r;
stop(r);

% --- 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)
global r;
b=getaudiodata(r);

[lo up]=envelope(b);
axes(handles.axes6);
plot(b); hold on;
plot(lo); hold on;
```

```
plot(up); hold on;
grid on;
title("Envelope detection of Recorded sound");


% --- 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)
global r;
play(r)


% --- 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)
global x;
global fs;
global lo;
global up;
[x fs]=audioread('SpeechDFT-16-8-mono-5secs.wav');
[up lo]=envelope(x);
axes(handles.axes7);
plot(up); hold on;
plot(lo); hold on;
sound(x,fs);
grid on;
title("Envelope detection of Recorded sound");



% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton1


% --- Executes on button press in togglebutton1.
function togglebutton1_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton1
```

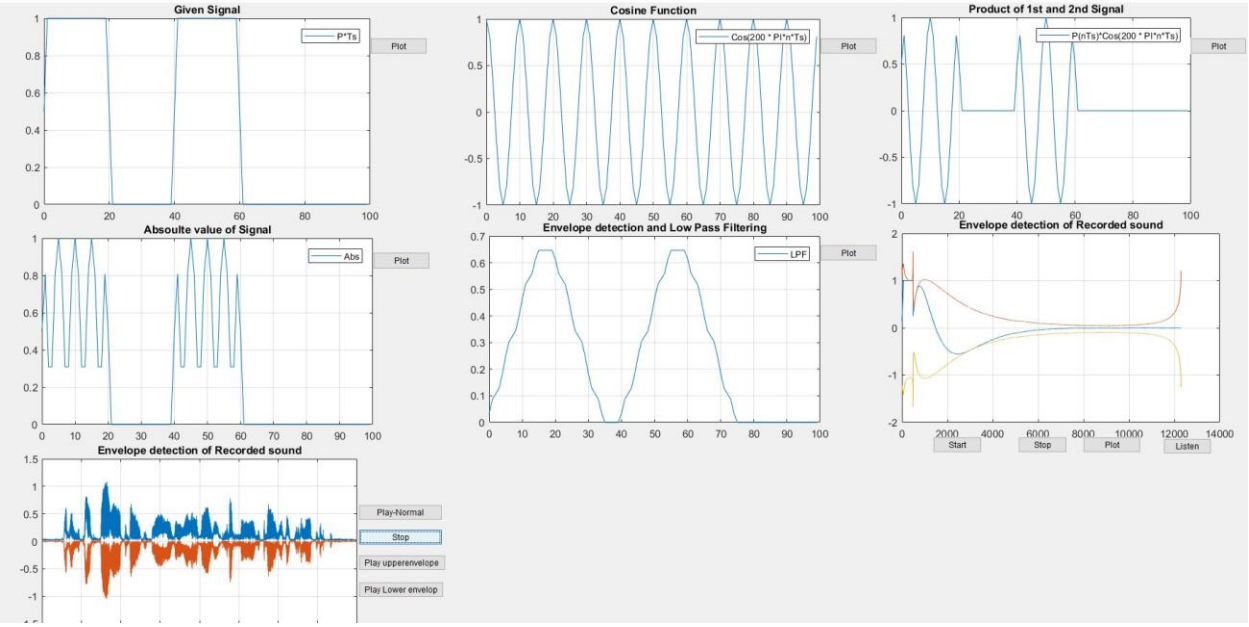
```
% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global x;
clear sound;


% --- 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)
global lo;
global fs;
sound(lo,fs);

% --- 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)
global up;
global fs;
sound(up,fs);

//This function must me declared separately in .m file and save
it in the same directory of the project
function y = averager(M,x)
b = (1/M)*ones(1, M);
y = filter(b, 1, x);
end
```

Results:



Bibliography:

https://www.electronics-notes.com/articles/radio/modulation/am-diode-detector-demodulator.php

https://www.dsprelated.com/showarticle/938.php

https://www.mathworks.com/help/audio/ug/sample-audio-files.html