Bangladesh University of Engineering and Technology

Course Title: Numerical Technique Laboratory

Implementation of Analog Modulation & demodulation scheme using MATLAB.

Name: **Mine Uddin Chisty**
Id: **1806135** Section: **C1**
Subject Code: EEE 212

Date of Submission: 30-Jul-2021

Objective:

This project mainly modulates a live or pre-recorded audio signal and retrieve the signal after demodulating it.

Modulation is the process of converting data into radio waves by adding information to an electronic or optical carrier signal. A carrier signal is one with a steady waveform -- constant height, or amplitude, and frequency.

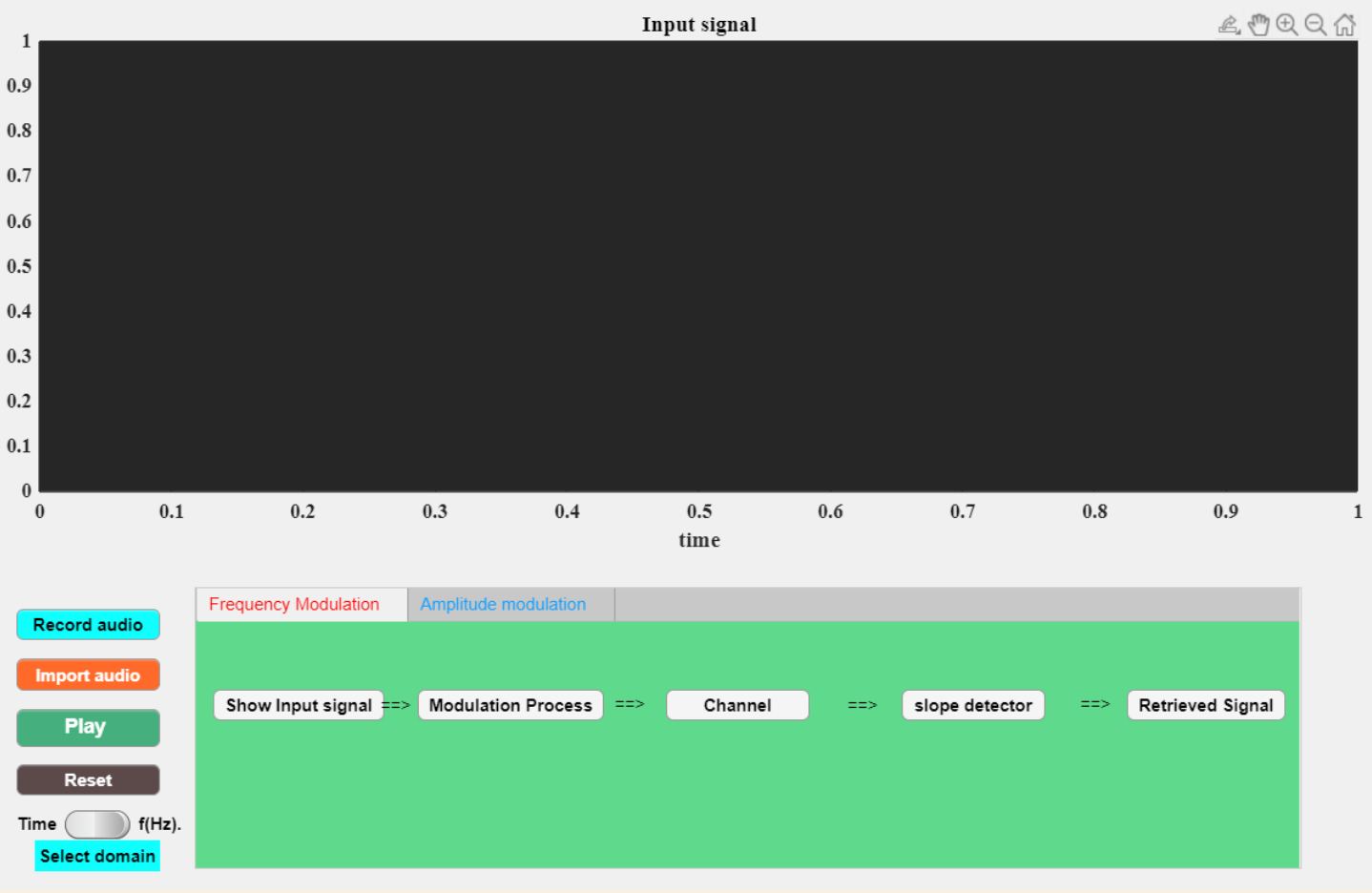
Demodulation is extracting the original information-bearing signal from a carrier wave. A demodulator is an electronic circuit (or computer program in a software-defined radio) that is used to recover the information content from the modulated carrier wave.

Code:

There are two main parts of the project code.

1) Frequency Modulation
2) Amplitude Modulation

GUI INTERFACE

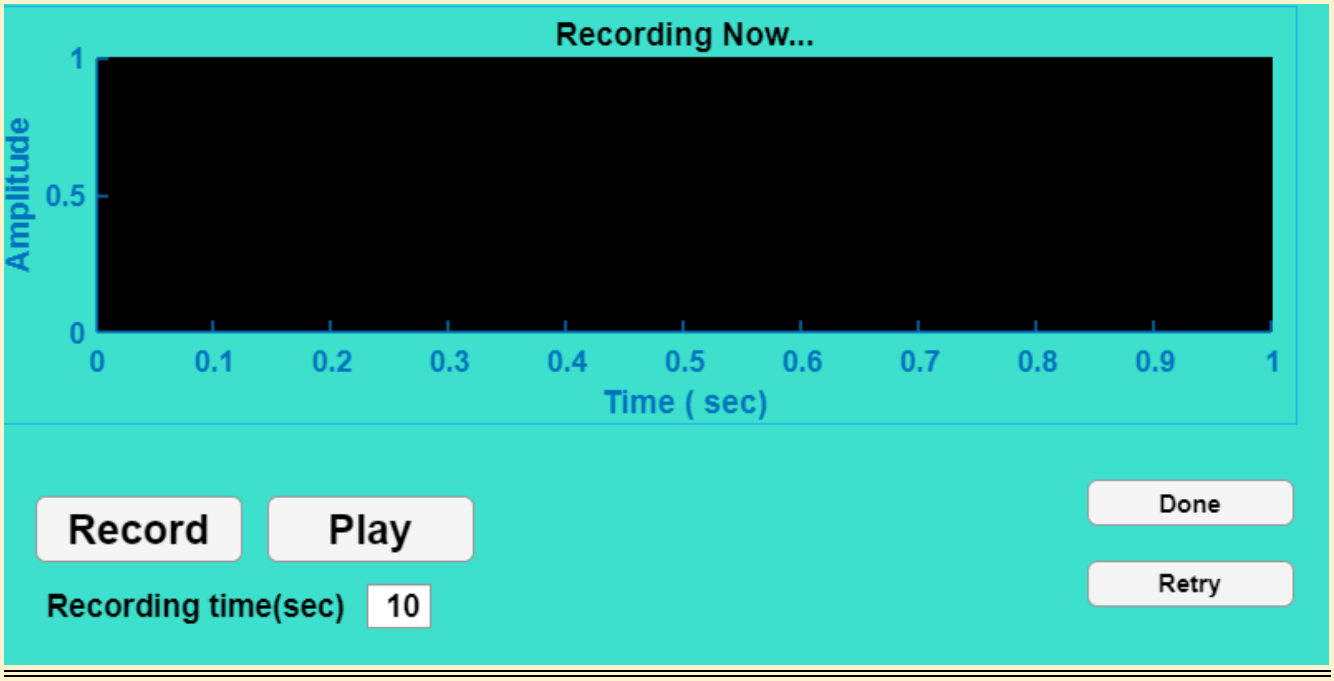


To build the GUI , we wil use properties given below:

```
properties (Access = private)
    t          % time
    tHR       % time vector High Res.
    orgsig  % Original signal
    fmSig   % frequency modulated signal
    amSig   % amplitude modulated signal
    deFmSig % demodulated fm signal
    deAmSig % demodulated am signal

    Y    %current Y
    X    % current X
    domain      % time or freq domain stores
    Audiocapturer % capture live audio
    Len         % length of original sound in second
    ratioM   % ratio of  the sampling rate of output and original signal

    Fs         % Sample frequency
    FsM        % Sample frequency multipler
   aDtype   % amplitude demodulator type
    Title % title of current plot


end
```

Recording audio:

After pressing the “Record audio” button, a pop up opens which takes a time duration and let us record live audio signal.



Properties for audio recording mlapp:

```
properties (Access = private)
    duration
    recording
    callerApp
    input
end
```

Function behind the "Record" button

```
function RecordButtonPushed(app, event)
        rec = audiorecorder;
        recordblocking(rec, app.duration);
        app.recording = getaudiodata(rec);
        t = linspace(0, app.duration, app.duration * 8000);
        plot(app.UIAxes, t, app.recording);
    end
```

Matlab Code:

```
Fs=8000;  %Sampling frequency in hertz
ch=1;
datatype='uint8';
nbits=16;
t=input('enter the time'); %time
%%
% to record audio data from an input device ...
a=audiorecorder(Fs,nbits,ch);
fprintf(" Recording for %.3f second\n",t);

recordblocking(a,t);
disp('End of Recording.');

%Storing recorded audio signal

x=getaudiodata(a,datatype);

%Write audio file

audiowrite('testing no 3.wav',x,Fs);
```

Playing Recorded signal:

```
function PlayButtonPushed(app, event)
        sound(app.recording, 8000);
    end
```

Importing Audio:

```
function ImportaudioButtonPushed(app, event)
        [filename, pname] = uigetfile({'*.wav'}, 'select audio files only');
        disp(strcat(pname, filename)); %pname => pathname
    if pname == 0
            return;
        end
  [app.orgsig, app.Fs] = audioread(strcat(pname, filename));
 signalrenew(app, app.orgsig); %singalrenew
end
```

Frequency Modulation:

```
function ModulationProcessButtonPushed(app, event)
        if isempty(app.orgsig) == 1
            return;
        end
        app.Title = 'FM SignaL (frequency modualated) ';
        Fc = 1600; % carrier freq
        j = 400;
        app.ratioM = 4;
        app.tHR = linspace(0, app.Len, app.Len * app.ratioM * app.Fs);
        Yn = interp1(app.t, app.orgsig, app.tHR);
        frDev = cumsum((app.tHR(2) - app.tHR(1)) * Yn) * j;  %frequency deviation
        app.fmSig = cos( 2*pi*(frDev + Fc * app.tHR));
        app.X = app.tHR;
        app.Y = app.fmSig;
        renewplot(app);  %updates the plot
        ylim(app.UIAxes, [-1, 1]);
        app.ModulationProcessButton.Enable = 'off';
    end
```

Slope detector demodulation:

```
function slopedetectorButtonPushed(app,  event)
        if isempty(app.fmSig)  == 1
            return;
        end
        Dsig = 3 * envelope(diff(app.fmSig));  %Dsig= demodulated signal
        app.deFmSig  = Dsig/max(Dsig);
        app.Y = app.deFmSig;
        app.X = app.tHR(1:end-1);
        app.FsM = 4;
        renewplot(app);
        ylim(app.UIAxes,  [min(app.Y),  max(app.Y)]);
        app.slopedetectorButton.Enable  = 'off';
    end
```

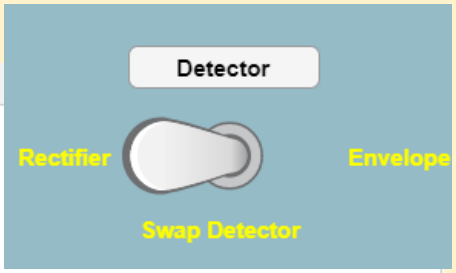
Ploting Retrieved Signal:

```
function RetrievedSignalButtonPushed(app, event)
        if isempty(app.deFmSig) == 1
           return;
        end
        app.FsM = 4;
        app.Y = app.deFmsig;
        app.X = app.tHR(1:end-1);
        app.Title = 'frequnecy DM signal';
        renewplot(app);
    end
```

Amplitude Modulation:

```
function ModulationProcessButton_2Pushed(app, event)
        if isempty(app.orgsig) == 1
          return;
        end
        app.Title = 'Amplitude Modulated Signal';
        Fc = 100;
        app.ratioM = 4;
        app.FsM = 4;
        app.tHR = linspace(0, app.Len, app.Len * app.ratioM * app.Fs);
        Yn = interp1(app.t,app.orgsig, app.tHR);
        app.amSig = Yn .* cos(2 * pi * Fc * app.tHR);
        app.X = app.tHR;
        app.Y = app.amSig;
        renewplot(app);
        app.ModulationProcessButton_2.Enable = 'off';
    end
```

Demodulation Process selection:



GUI for Detector

```
function DetectorButtonPushed(app, event)
    if isempty(app.amSig) == 1
        return;
    end

    app.deAmSig = app.amSig; %deAmsig= demodulated amplitude signal
    app.deAmSig(app.deAmSig < 0) = 0;
    app.Title = 'Demodulated signal for Rectifier ';
    if app.aDtype  == 1  %amplitude demodulation type
        app.deAmSig = envelope(app.deAmSig);
        app.Title = ' Demodulatd signal for Envelope';
    end
    app.FsM = 4;
    app.DetectorButton.Enable = 'off';
    app.X = app.tHR;
    app.Y = app.deAmSig;
    renewplot(app);
end

% Value changed function: SwapDetectorSwitch
function SwapDetectorSwitchValueChanged(app, event)
    value = app.SwapDetectorSwitch.Value;
     switch value
        case 'Envelope'
            app.aDtype = 1;
        case 'Rectifier'
            app.aDtype = 0;
    end
    disp(app.aDtype);
end
```

By this function we can chose the detector type to demodulate the amplitude signal very easily.

Functions used in the GUI:

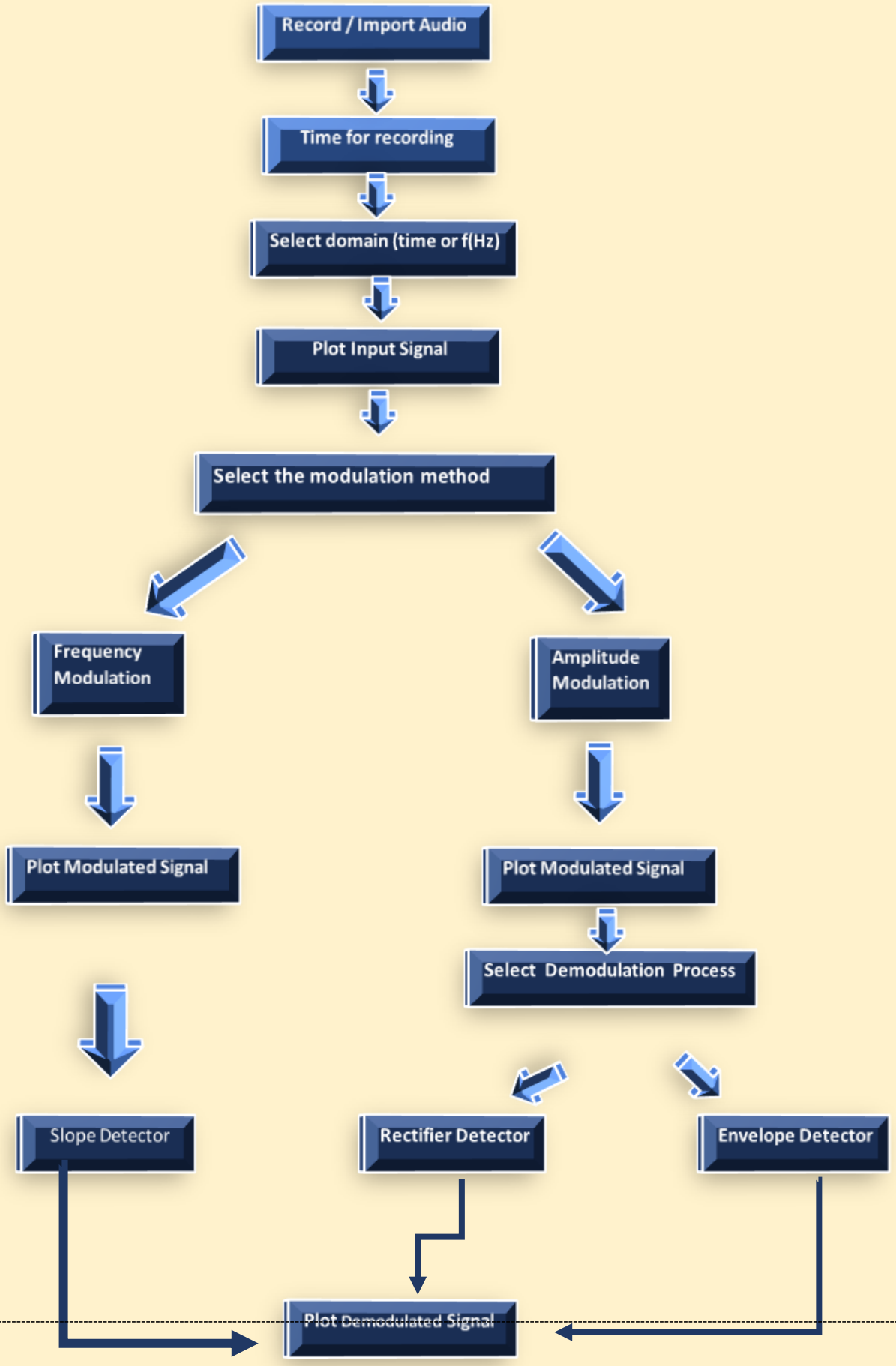
1) renewplot : this function can update the plot information. The code is below

```
function results = renewplot(app)
    if isempty(app.Y)
        return;
    end
    if length(app.X) ~= length(app.Y)
        return;
    end
    if app.domain == true
        plot(app.UIAxes, app.X, app.Y);
        title(app.UIAxes, strcat(app.Title, ' In the time domain'));
        xlabel(app.UIAxes, 'time(sec)');
        xlim(app.UIAxes, [min(app.X), max(app.X)]);
        ylim(app.UIAxes, [min(app.Y) - 0.5, max(app.Y) + 0.05]);
    else
        y = fft(app.Y); %fourier
        L = length(app.Y);
        P2 = abs(y/L);
        P1 = P2(1:L/2+1);
        P1(2:end-1) = 2*P1(2:end-1);
        f = app.Fs*(0:(L/2))/L;

        plot(app.UIAxes, f,P1);
        title(app.UIAxes, strcat(app.Title, ' In the frequency domain'));
        xlabel(app.UIAxes, 'f(Hz)');
        ylabel(app.UIAxes, 'Intensity');
        xlim(app.UIAxes, [min(f), max(f)]);
        ylim(app.UIAxes, [min(P1) - 0.2, max(P1) + 0.2]);
    end
```

2) Reset: resets the modulated plot and audio.

```
function results = reset(app)
    app.ModulationProcessButton.Enable = 'on';
    app.slopedetectorButton.Enable = 'on';
    app.ModulationProcessButton_2.Enable = 'on';
    app.slopedetectorButton.Enable = 'on';
    app.FsM = 1;
    %reset all data
    app.fmSig(1:end) = 0;
    app.amSig(1:end) = 0;
    app.deFmSig(1:end) = 0;
    app.deAmSig(1:end) = 0;
    app.Y(1:end) = 0;
    renewplot(app);
end
```

Flow Chart of Signal modulation and demodulation Project

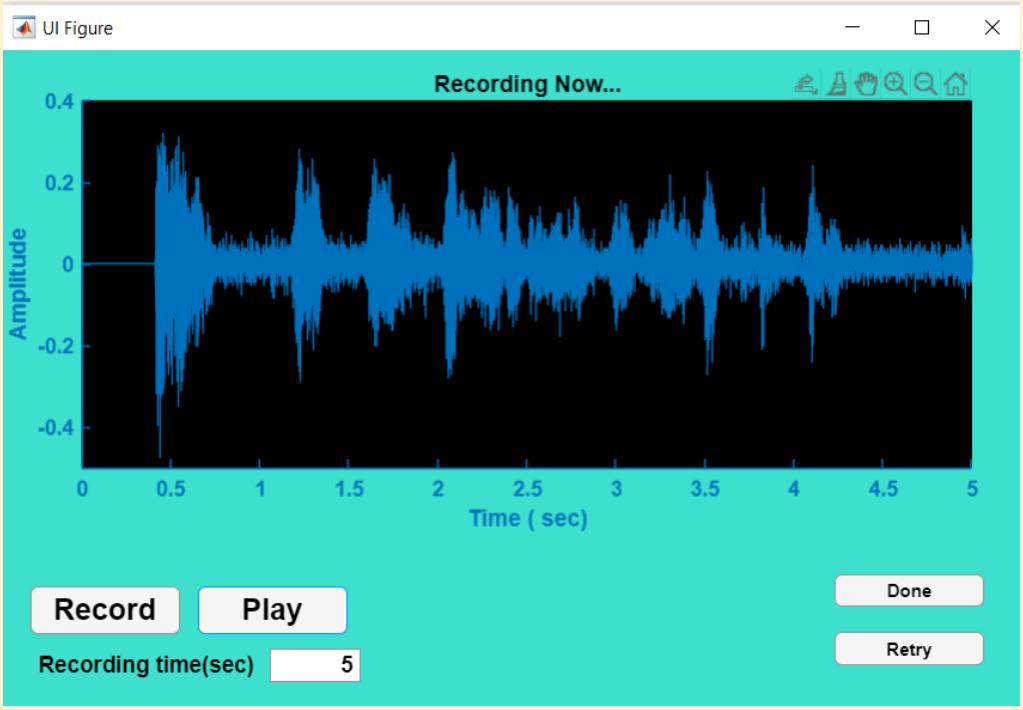


Test cases:

Test case 1:

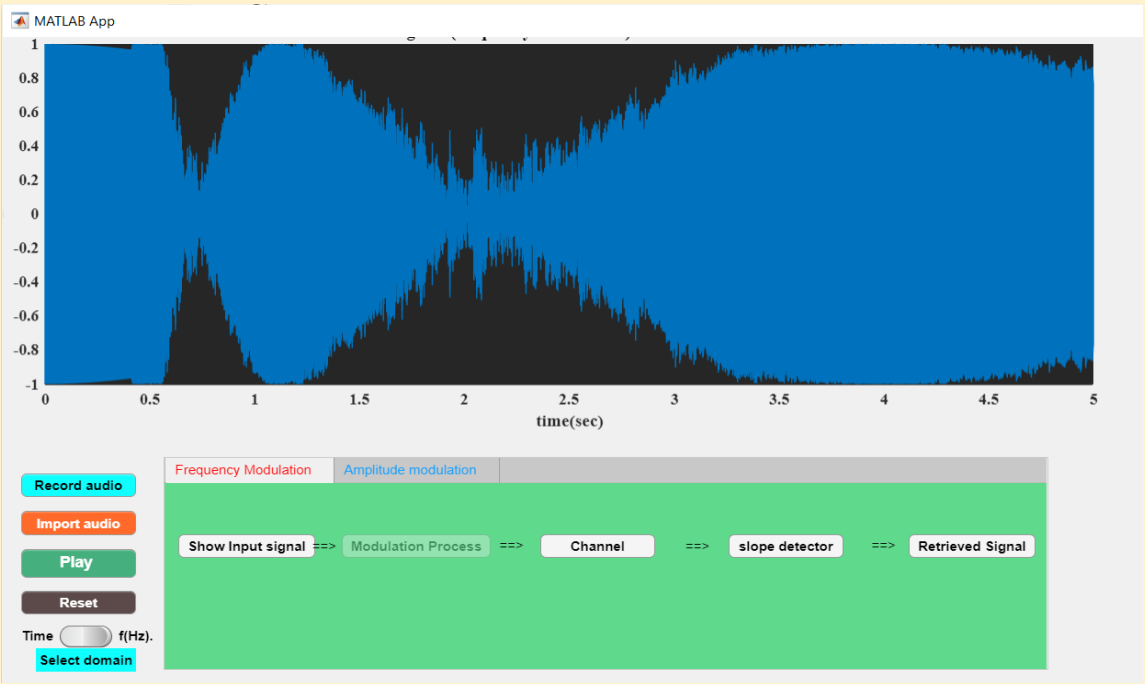
In this case, we will observe a live audio signal and plot its modulated and demodulated signal using GUI.

Taking plot of live audio record

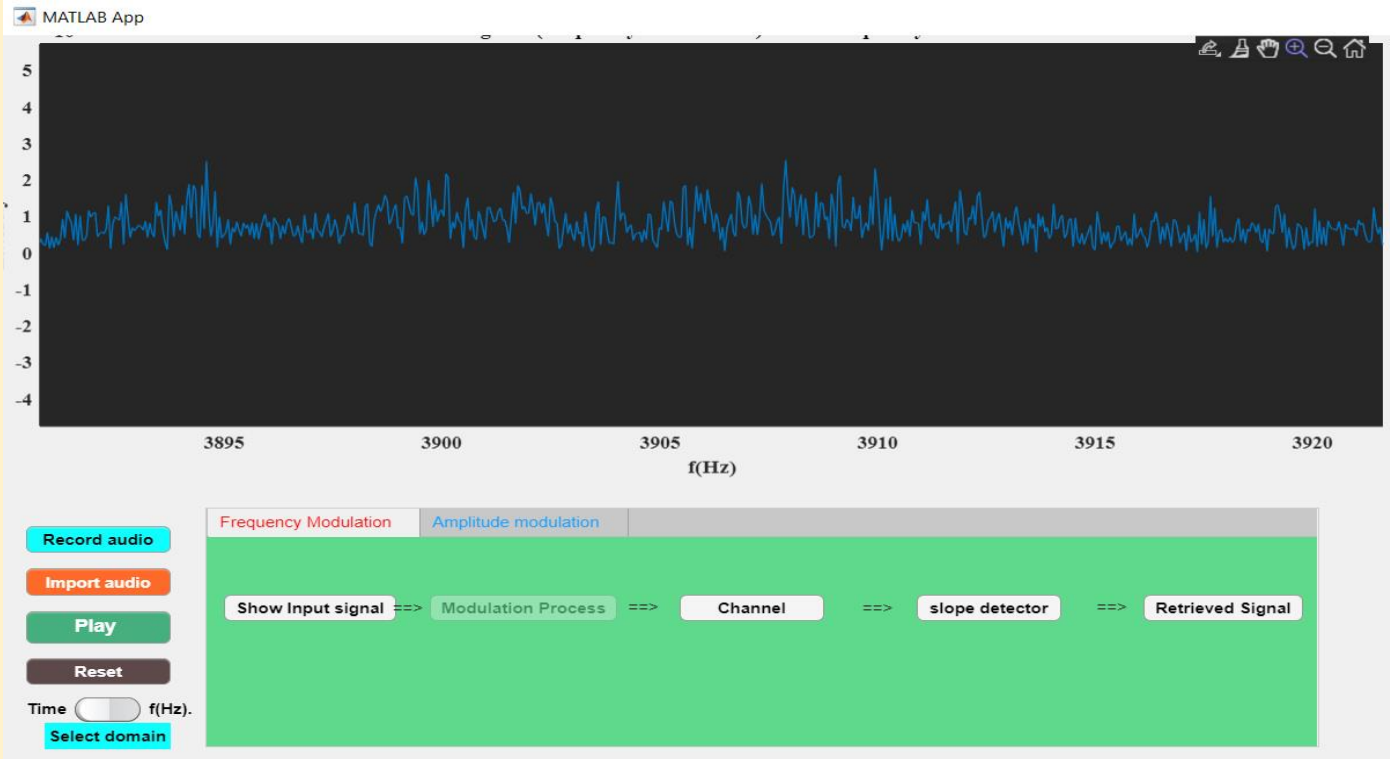


Frequency Modulation and Plotting:

Time domain plot:

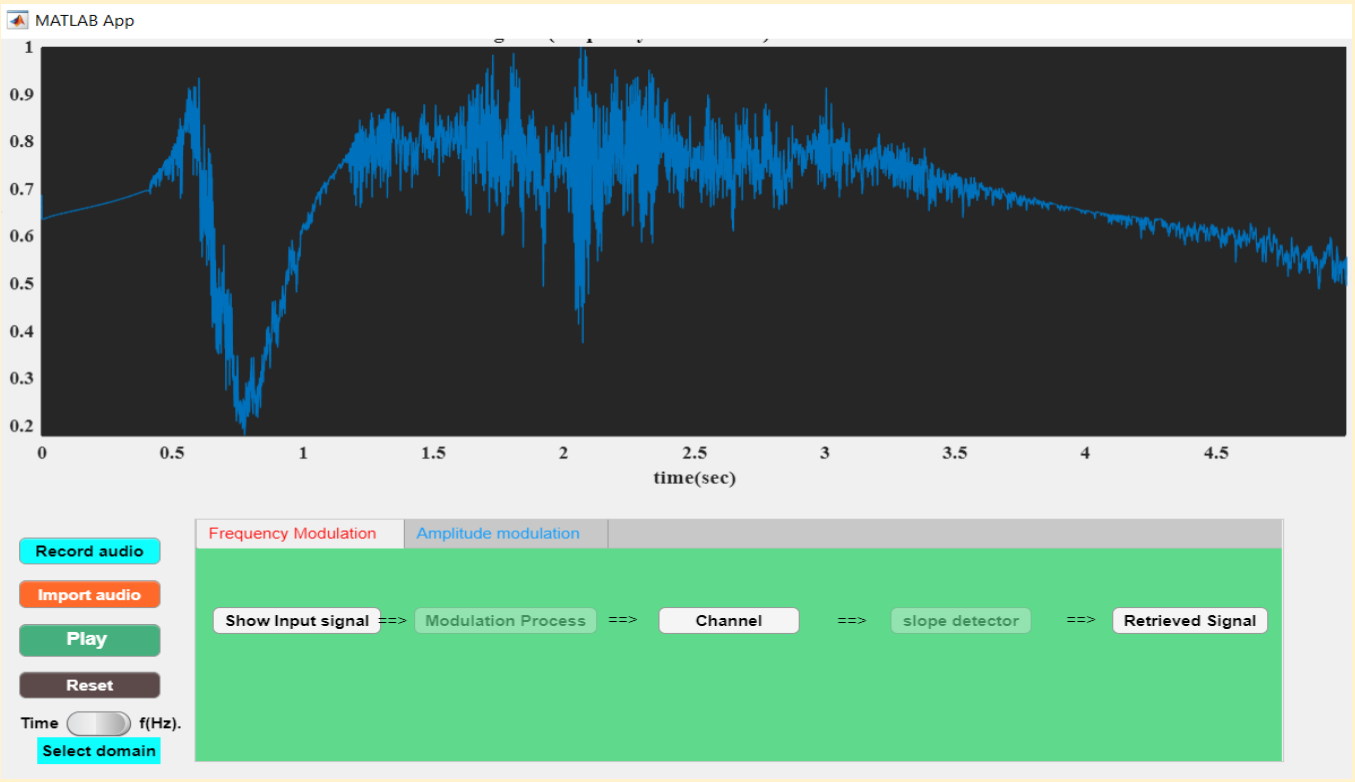


F(Hz) domain plot:

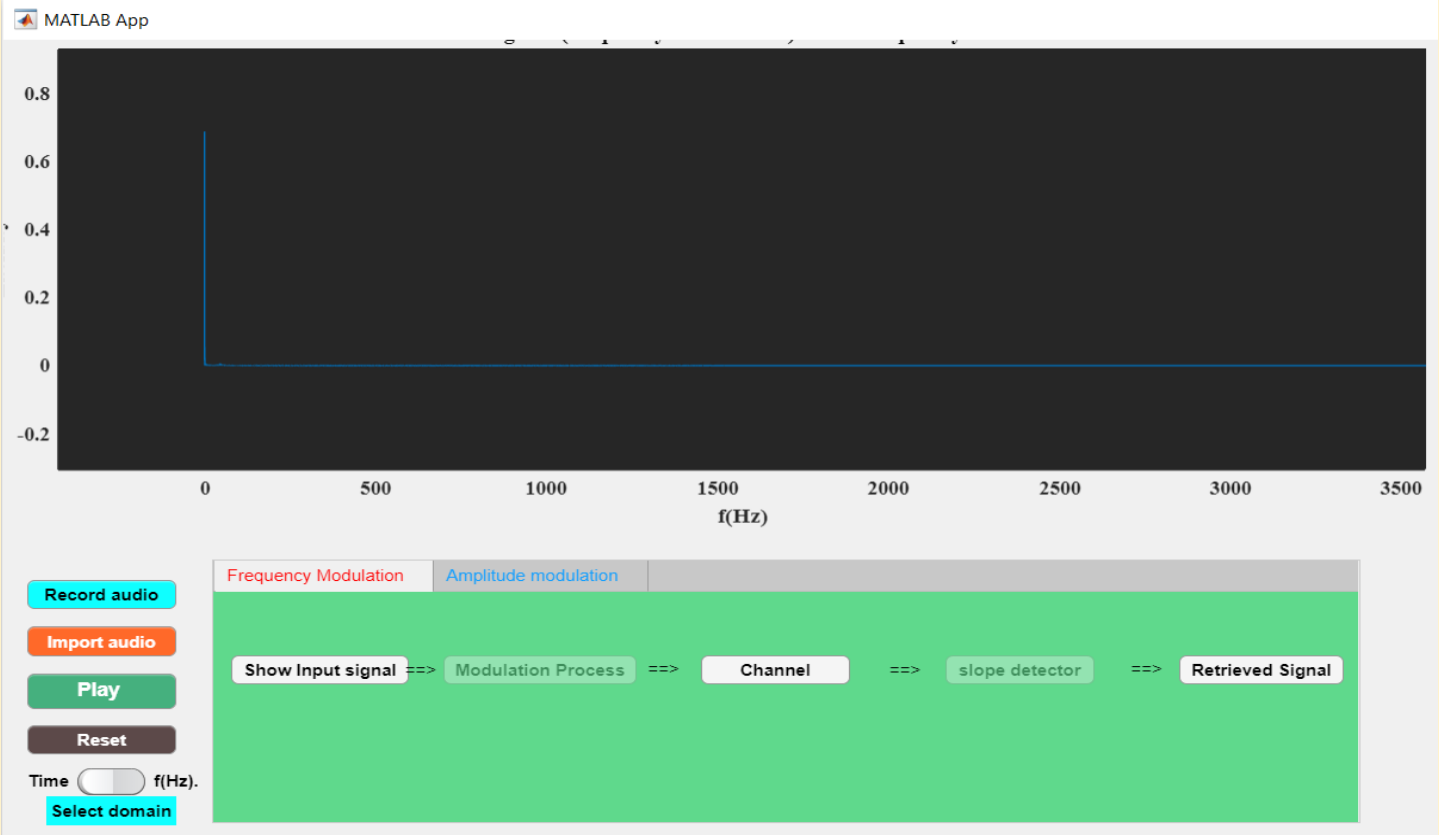


Demodulated Signal (Slope detector):

Time domain Plot:

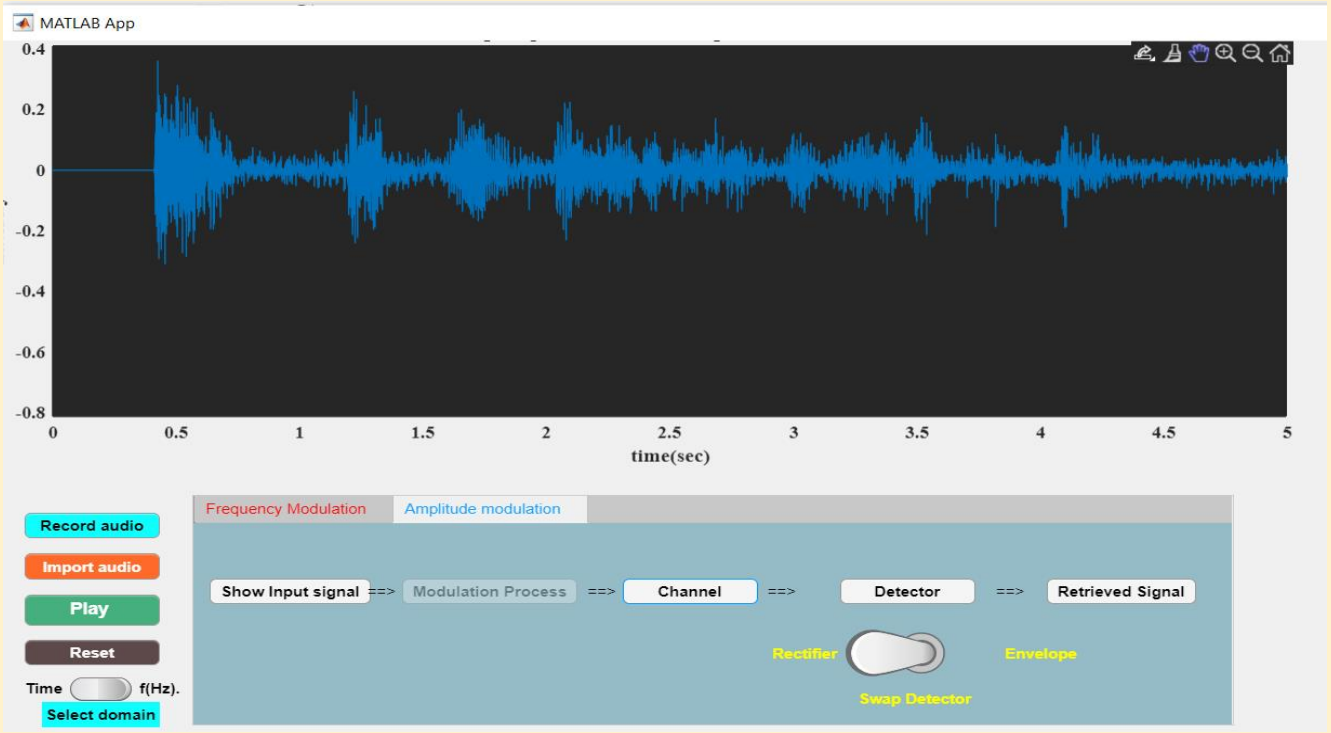


f(Hz) domain Plot:

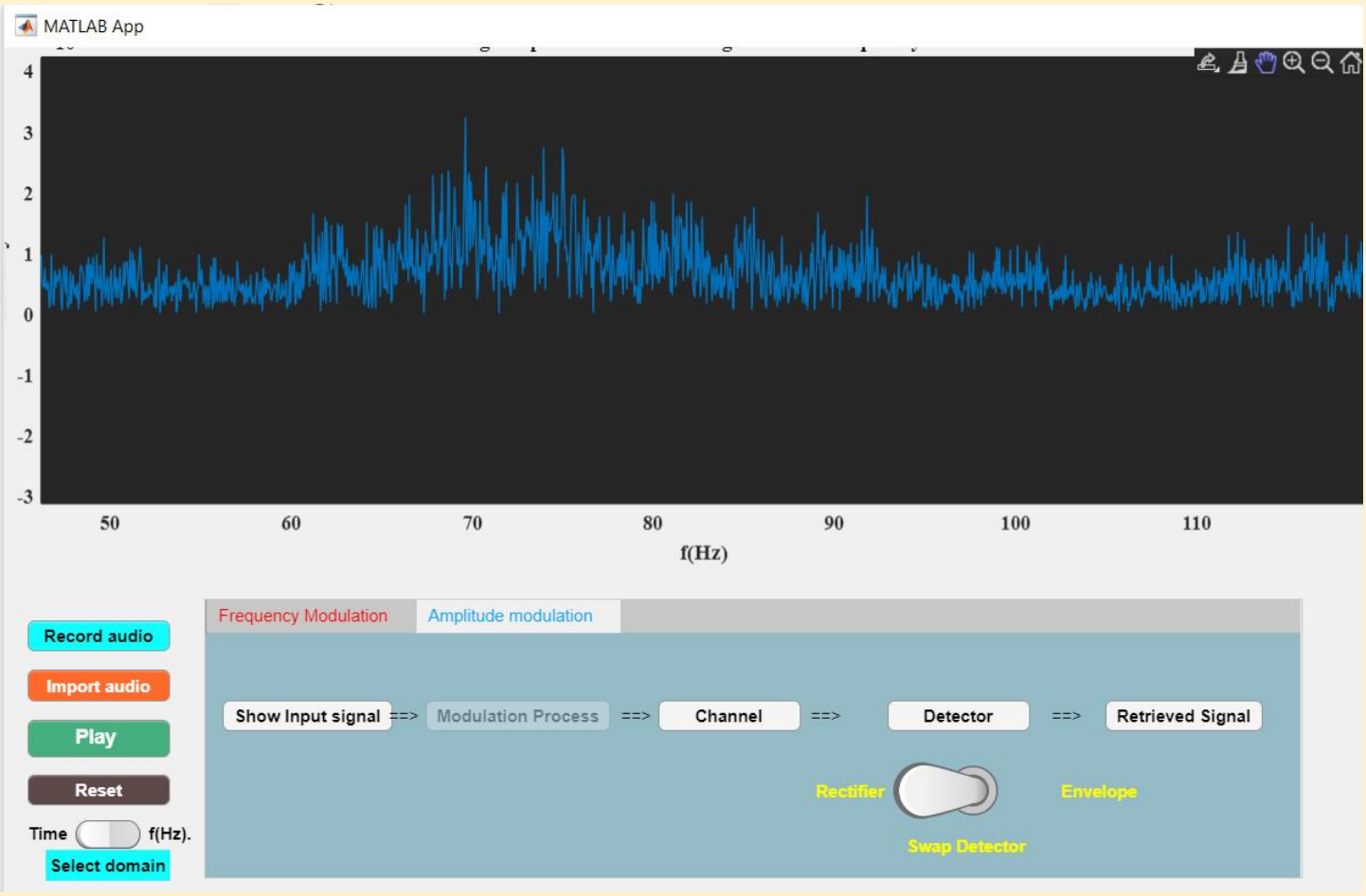


Amplitude Modulation and Plotting:

Time domain plot:

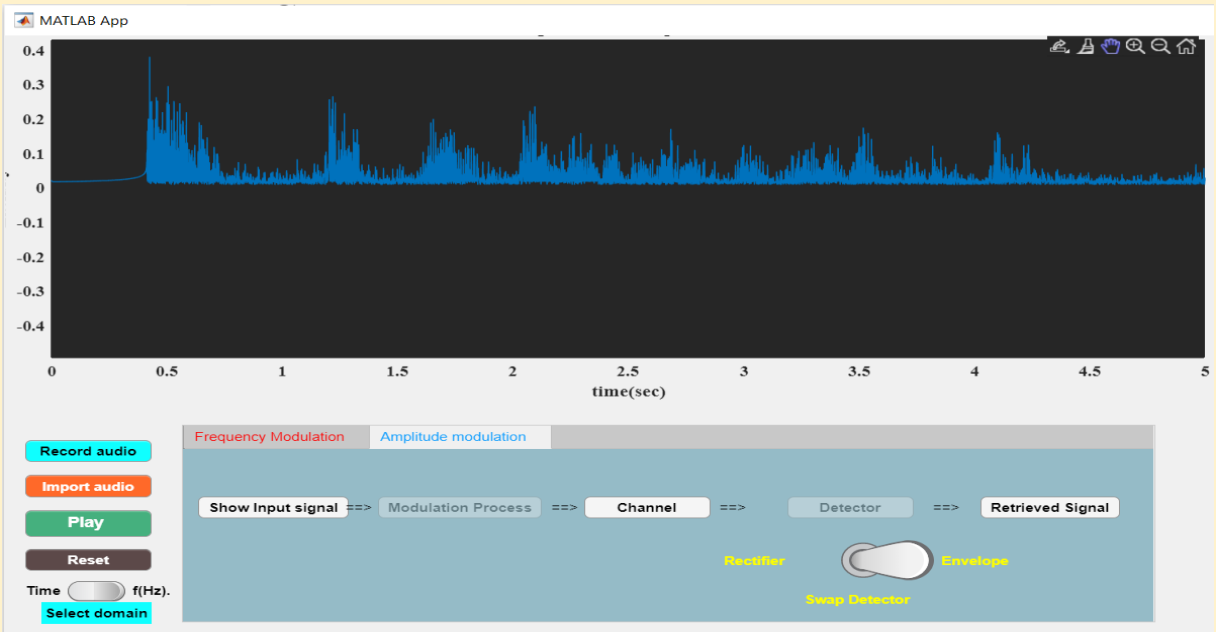


f(Hz) domain plot:



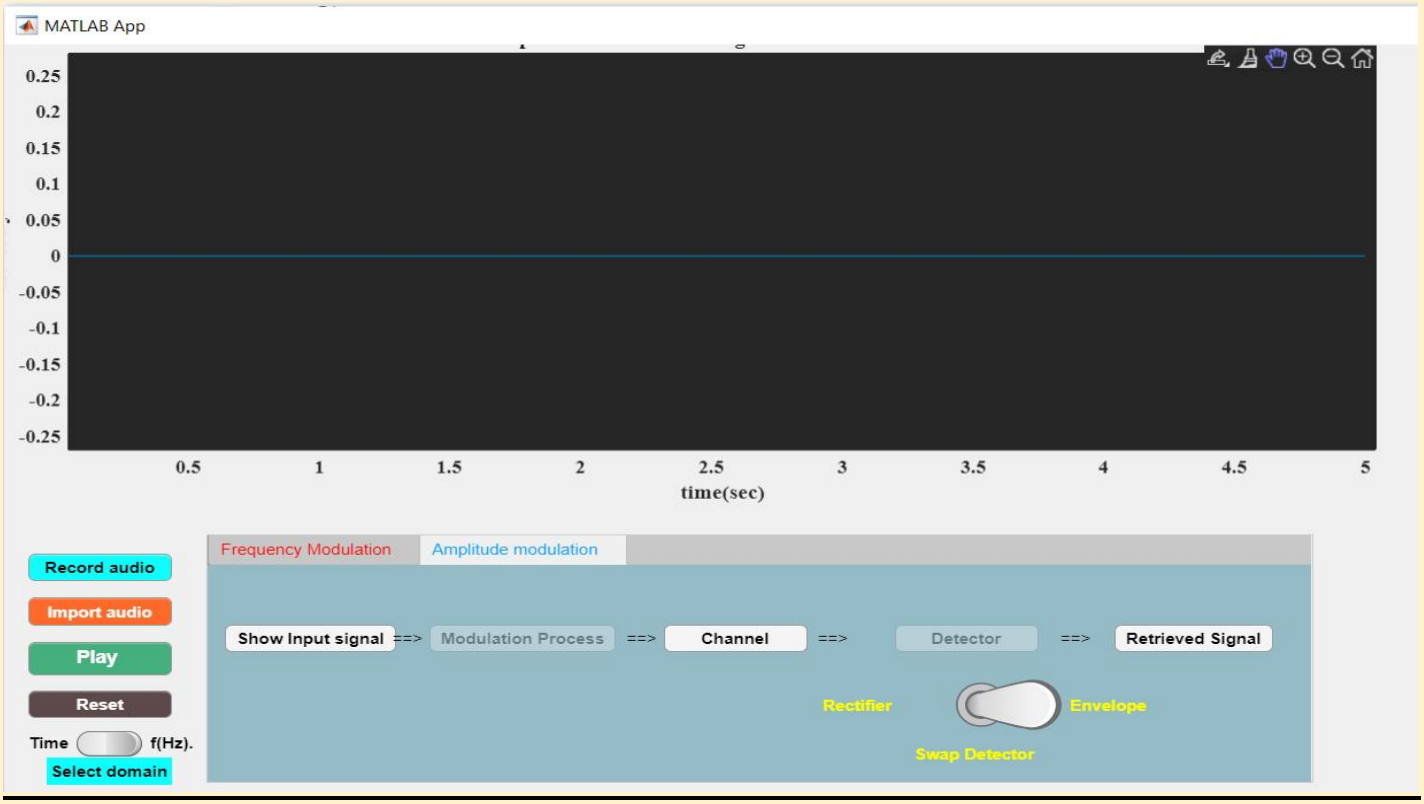
Rectifier Demodulation:

Time domain:



Envelope Modulation:

Time domain:



By using recorded audio, same test cases can be observed.

Discussion:

The project mainly deals with the modulation and demodulation of recorded or real time audio signal. Real time audio is recorded and then adding higher frequency career signal, the signal is modulated. Demodulation is done removing unwanted signals and plotted on the graph.

Future Scope:

- A faster algorithm can be used to modulate and demodulated the signals.

- The GUI can be made more interactive.

- Complexity and iterations can be reduced if further dug in.

- Plotting can be done with more details.