ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

MATLAB BASED SIGNALS AND SYSTEMS TRAINER

Group Members: Abebe Woldetsadik ATR/4066/08

Alexander Gebreyohannes ATR/6146/08

Amanuel Taye ATR/3020/06

Advisor: Dr. Ing Dereje Hailemariam

Signature: ……………………

Date of submission: **12/12/2020**

Abstract

Aside from advanced mathematics, an effective way of communication is crucial for teachers and students in dealing with courses involving higher level of mathematical equations and calculations. Dry lectures and equations make students study engineering in a traditional manner regardless of their previous knowledge. Although the course, signals and systems, provides students with a basic understanding of main types of signals and systems, the limitation of students' capability to realize signals and systems is always there. As a result, students face a tough time trying to fully grasp the points behind those equations. In this paper, MATLAB is used to develop a comprehensive demonstration using Graphical User Interface. MATLAB Based Signals and Systems trainer aims at providing students with an often simpler and more user-friendly explanations of signals and systems.

Key words: Signals and Systems, MATLAB, Graphical User Interface (GUI)

Acknowledgment

First of all, we are grateful to the Almighty God. Then we wish to express our sincere thanks and gratitude to our advisor Dr. Ing Dereje Hailemariam for advising us.

Table of Contents

List of tables

List of figures

Chapter One

1.1 Introduction and background

Digital Signal Processing (DSP) is a required education in learning any signal involving technology. Digital Signal Processing is the most prevailing technology that has been shaping science and engineering of the 21st century. Ground-breaking changes have been brought to a broad range of fields: Communication, radar and sonar, medical imaging, music and video production, to name just a few. As each of these areas involve of its own deep and thorough algorithmic and mathematical techniques, students find it tough to deal with mathematical equations and stay focused without losing motivation. Although learning general concepts of Digital Signal Processing (DSP) is crucial to understanding the specialized techniques that a specific field may require, if an effective means of learning is not employed, students will continue to be consumed spending most of their time working on unrewarding elementary problems. And the powerful tool that is being used in achieving this goal efficiently is MATLAB.

MATLAB is a programming platform designed specifically for engineers and scientists. It has been used as a teaching aid in many subjects such as mathematics, computation, algorithm development, electronics, circuit design, communication theory, data analysis, exploration and visualization and many other applications. It also provides users with a powerful integrated GUI development environment. By inputting GUIDE commands in to the development environment through the command window, users can develop their application. [1]MATLAB is more preferable over other programming languages such as C++, Visual Basic or Java for several reasons. It is quite easy to program in these environments for a computer science or information technology student but for other science and engineering students this pose problem as they are not used to these environments. A flexible software structure of MATLAB, powerful calculation and visualization means of the package in MATLAB, a wide selection of Toolboxes, predefined functions for solving application-specific problems are among the several pros of MATLAB. One important feature of MATLAB is GUIDE. MATLAB-GUIDE presents information; variable-changing buttons, concept-demonstrating panels, in a unified framework. A concise MATLAB-GUID is implemented to simulate the generation of several functions such as sine-wave, cosine-wave, sinc-function and many others; triangle, rectangular, unit functions and perform continuous-time domain and discrete-time domain.

DSP education roots in signals and systems. Signals are a unique type of data carrying information that varies with time or with any independent variable such as audio signal and video signal. And a system processes an input signal to produce a desired output signal.

Signals and systems come in different types of classifications, properties and operations. Hence, they are frequently discussed without knowing the exact parameters being represented. This is the same as using x and y in algebra, without assigning a physical meaning to the variables. Systems also need to be understood as they have to be designed in a specific way to obtain the desired output. This leads to a very sophisticated mathematical equations and calculations. What most students struggle is then to understand those techniques encountered in solving the equations. As a result, they get lost in the midst and pay little or no attention to the final results which in fact, is more important than what it takes to get there. For instance, if a speech audio signal has some sort of noise, the desirable output signal would be a signal with much less noise. To achieve this, several system design techniques are deployed.

In such case, visualization and interactive demonstrations of signals and systems plays a great role in the teaching and learning system of Digital Signal Processing (DSP). In this paper, main signal functions are made to be plotted in both continuous and discrete time domain. These functions can be sampled, quantized, analogously or digitally modulated and demodulated. Audio Signal Processing, Image Signal Processing and main types of Filters are also included in the MATLAB based signals and systems trainer project.

1.2 Problem Statement

Signals and systems is one of the fundamental courses in every electrical and electronic engineering program, and all students taking these courses are expected to have an understanding of the behavior of complex analogue and digital signals and systems. Signals and systems are highly mathematical concepts and teaching this course in general involves the traditional way of teaching the theory in the classroom, where the lecturer presents the complex theory and students attempt to learn it. To begin to work with these concepts however, a student should have a high ability in mathematics, and good skills of retaining the learned material.

Without employing mathematics at high level, the teaching of signals and systems become dull and not meaningful. Consequently, at a level just above the normal comfort level, students learn the very basic methods which are not in general sufficient for understanding the behavior of complex signals in real life. In engineering education, computer simulations and laboratory experiments support the theory learned in the classroom and enable students to see the applications of the theory into practice.

In this project we are going to create a large set of exploratory demonstrations and applications programmed in MATLAB GUI to teach essential concepts, principles and applications of signal and system and communication systems course.

1.3 Study limitation and Scope

The overall objective of this study is to give cover to basic signals and systems rather than to cover all, which could only be unrealistic and time consuming. As a result, one may not find the specific signal or system of importance for a specific matter of purpose as the signals and systems included here are few and basic. Some of the systems-windows however, gives the user the opportunity to type in the function and examine through the system. For instance, in the case of continuous time convolution, the user can either select a signal from the predefined signals or fill in once own input signal and impulse signal to be convolved. But this is not true for all of the other systems. All in all, we assume, the project successfully provides the profound concepts of signals and systems and signal processing as well.

1.4 Objective

1.4.1 General objective

To develop a MATLAB-Based Signals and Systems Trainer package

1.4.2 Specific objective

- To create a user-interactive MATLAB-Based Digital Signal Processing (DSP) platform
- To enable students grab visual understanding of Signals and Systems
- To keep students motivated in the way of studying Signals and Systems

1.5 Literature Review

The subject matter of this project is to familiarize engineering students with the concepts of numerous signals and systems using MATLAB-GUIDE. Many scholars have done researches demonstrating typical signal functions and systems. Some of these researches are [1]Signals and Systems Using MATLAB, [2]Signal Analysis Based on MATLAB-GUI, [3]Teaching Signals and Systems at Undergraduate level. Although these researches have shown that CBE specially deploying MATLAB has great importance in teaching signals and systems, they could have been improved by combining them together and adding some other important features. This review will demonstrate what each of these researches is missing in comparison to MATLAB-based Signals and Systems Trainer.

Computer has been used in teaching signal processing for years. [4]Martti Rahkila in his paper, Computer Based Education (CBE) for signal processing, describes that computers easily provide some elements in education that are difficult to achieve using standard methods. Martti Rahkila also says that with signal processing, especially with Digital Signal Processing (DSP), Computer Based Education (CBE) can be fruitful. [5]Radki and Kulkarni had also designed an integrated MATLAB suit for introductory DSP Education. Thus, using computers for educating signal processing is proven to be effective.

The objective of the research; Signal Analysis Based on MATLAB-GUI was to design a concise MATLAB-GUI which simulates the generation of multiple signals. This paper has shown time-domain and frequency-domain analysis of signals. However, as it is mentioned in the conclusion, this particular paper didn't include the two-dimensional image processing and signal processing algorithms such as convolution and Fourier series. Similarly, [6]Fahreddin Sadikoglu and Dogan Ibrahim in their research under the title Teaching Signals and Systems at Undergraduate Level, developed simulation and lab session using MATLAB. In the same manner, this research could also be improved by adding other signals processing such as audio and image.

Bob L. Sturm and Jerry D. Gibson had employed MATLAB to illustrate sophisticated concepts of signals and systems. They created a MATLAB-GUIDE platform that has 37 programs including waveforms, modulation, sampling and interpolation, aliasing the frequency domain, convolution and filtering, pole-zero diagrams, analysis and synthesis, signal features and many others. Although the developed SSUM application was relatively easy, it is costly and students could not access the code behind the application as the approach used hides it away. On the other hand, the approach used in MATLAB-Based Signals and Systems Trainer let students see into the code and modify it if it is needed.

All in all, although the researches that are mentioned above have their own respective strengths, at the same time, they are found to be missing important aspects that concerns anybody who would want to study and understand signals and systems in an easy way.

Chapter Two

2.1 Methodology

MATLAB-Based Signals and Systems Trainer is a suite of explanatory demonstrations coded in MATLAB. These demonstrations are designed in MATLAB-GUIDE to support the traditional way of teaching and learning system of signals and systems. Several programs such as convolution, typical signal functions, modulation and demodulation of signals, carrier and signal messages, audio signal processing, image signal processing, different noise adding effect, filters and many others are included. MATLAB-Based SST is designed in a way that students can access the code and it will be available to the public for free.

GUIDE (Graphical User Interface Development Environment) is a development environment that provides a set of tools for creating user interfaces (UIs). User Interface (UI) is a graphical display in MATLAB-windows containing controls, called components; that enable a user to perform interactive tasks. UI components include menu, toolbars, push buttons, radio buttons, list boxes, text fields, sliders, panels, axes-just to name a few. UIs creating MATLAB tools can also perform any type of computation, read and write data files, communicate with other UIs, and display data as tables or as plots.

- Guide provides the tools to design user interface and to create custom apps.
- To launch guide, there are two possible ways. One is to the command window and type GUIDE and a window which has options with common layouts will pop up. GUIDE provides several templates to create your own UIs. The templates are fully functional apps.
- To start from scratch, choose the blank GUI.
- When you hit ok, the new window that pops up is called the GUIDE layout editor.
- From here, it can be designed the layout of the GUI by dragging and dropping components from the left on to the canvas. But before you start doing that, go to file---preferences---and check the top box to see what each icon represents which specific component.

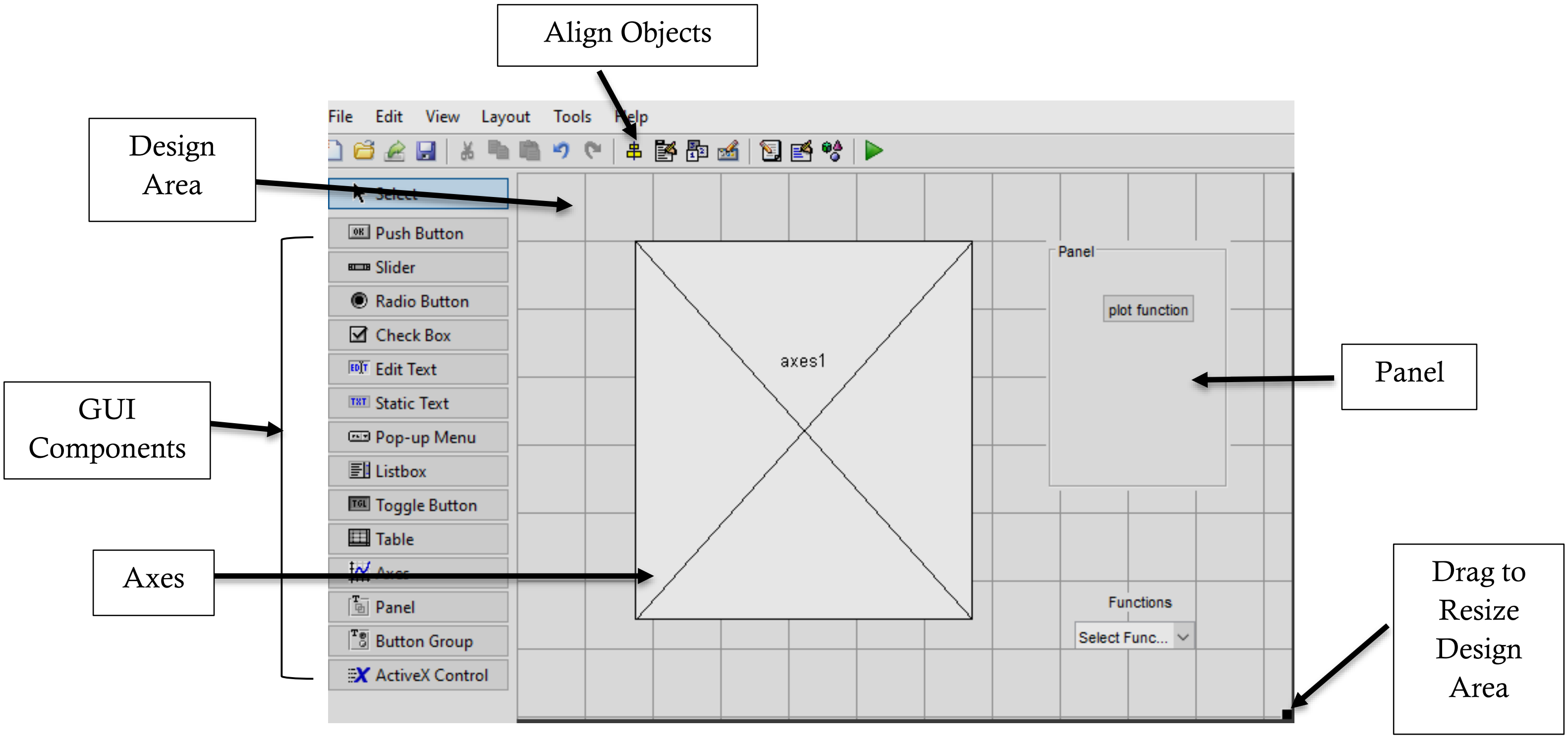

Figure 2.1 Procedure on how to setup the GUI

- The commonly used components are push button, slider, radio button, check box, edit text, static text, pop-up menu, list box and toggle button.

Some Basic GUI Components

Element	**Description**
Push button:	Triggers a call-back when clicked with a mouse.
Toggle button:	A toggle button is either "on" or "off," and it changes state each time that it is clicked. Each mouse button click also triggers a call-back.
Radio button:	A radio button is a type of a toggle button that appears as a small circle with a dot in the middle when it is "on." Groups of radio buttons are used to implement mutually exclusive choices. Each mouse click on a radio button triggers a call back.
Check box:	Check box is a type of a toggle button that appears as a small square with a check mark in it when it is "on." Each mouse click on a check box triggers a call-back.
Edit box:	An edit box displays a text string and allows the user to modify the information displayed. A call-back is triggered when the user presses the *Enter* key.
List box:	A list box is a graphical control that displays a series of text strings. A user can select one of the text strings by single- or double-clicking on it. A call-back is triggered when the user selects a string.
Popup menus:	A popup menu is a graphical control that displays a series of text strings in response to a mouse click. When the popup menu is not clicked on, only the currently selected string is visible.
Slider:	A slider is a graphical control to adjust a value in a smooth, continuous fashion by dragging the control with a mouse. Each slider change triggers a call-back.

Table 2.1 Some Basic GUI Components

Static Elements

Frame:	Is a rectangular box within a figure. Frames are used to group sets of controls together. Frames never trigger call-backs.
Text field:	Creates a label, which is a text string located at a point on the figure. Text fields never trigger call-backs

Table 2.2 Some Basic GUI Static Elements

Menus and Axes

Menu items:	Creates a menu item. Menu items trigger a call-back when a mouse button is released over them.
Context menus:	Creates a context menu, which is a menu that appears over a graphical object when a user right-clicks the mouse on that object.
Axes:	Creates a new set of axes to display data on. Axes never trigger call-backs.

Table 2.3 Some Basic GUI Menus and Axes

- Add an axis. Then add a panel on which you are going to add some push buttons. A panel helps to manipulate the push buttons as a group. Buttons can be aligned and distributed by using the alignment tool. All components have property inspectors. The property inspector allows you to view and set object property. To assign a unique identifier for each UI control, alter the Tag property of each object. GUIDE uses the Tag property to name the automatically generated MATLAB functions. Once you save and run the layout, it generates a MATLAB fig. file which displays the GUI. GUIDE generates two files: the FIG-file, with extension .fig, which contains layout information and each component such as push buttons, axes, panel, menus and so on and a code m-file, with extension .m, which contains implementation code and templates for some call-backs that control behaviour. Once MATLAB generates the code automatically, custom code can be added to add functionality for when the user interacts with the GUI. These custom codes are called call-back functions.

The trainer package is designed in a way that the user can easily access a specific process. When the run button is clicked over, a window pops up with five buttons on it. This window mainly consists of five major processes: Basic functions' plot, Modulation and Demodulation, Audio and Image processing, Convolution and Fourier Transform. In this methodology, these will be explored and the respective developing process will be shown as well. When each button is clicked, it leads you to the specific layout which has several buttons, axes and other components significant to that specific system. The first window gives the user an overall image of the trainer package and clear direction to where to go.

The GUI has three main blocks:-

Figure 2.2 The main GUI

2.1.1 <u>Basic Signal function block</u>: - it consists of basic signal functions plot and sampling and quantization.

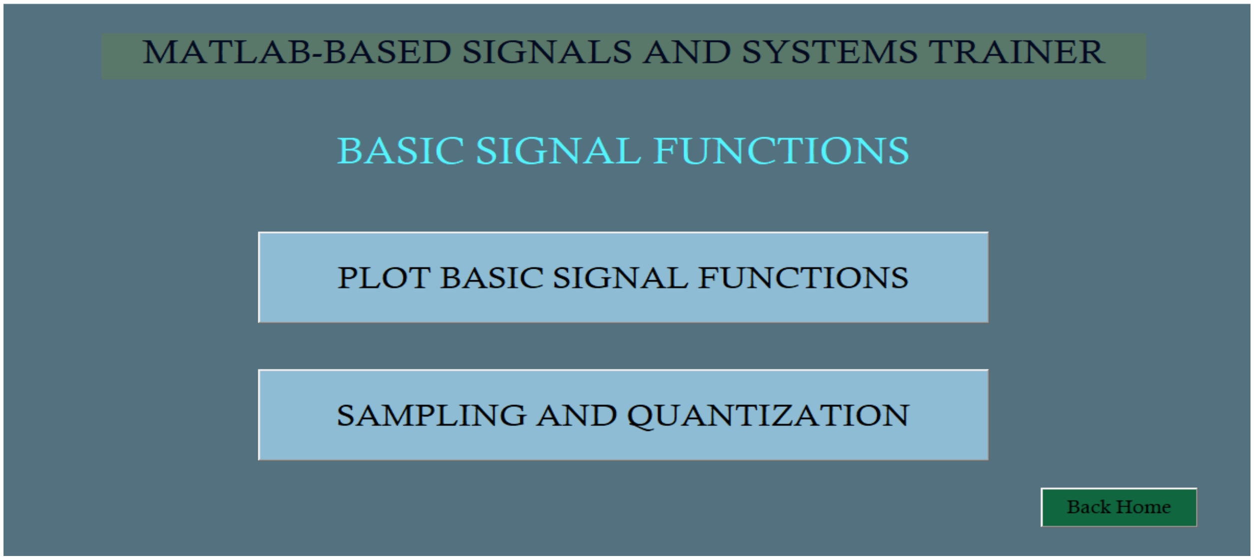

Figure 2.3 Basic Signal function block

<u>Basic signal functions plot</u>:- a system designed to perform a particular task often uses measurements obtained from the environment and/or input from the user. These in turn may be converted into other forms. The physical variables of interest are generally called signals. In an electrical system the variable of interest could be voltage, current, etc. In communication system, signal is a codified message.

In this project some of the typical signal functions such as sine, cosine, exponential, rectangular pulse, sinc, unit step, impulse, ramp, tan and triangular pulse are defined.

Basically, there are five panels in the Plot-Basic-Signal-Functions-window; Plot-a-Function, Type-of-Signal, Time-Adjustment, Display-the-Function and Values. The Plot-a-Function panel has a popup menu where all the basic above-mentioned signal functions are listed. The Type-of-Signal panel has the time domains. When the continuous time-box is checked, the continuous function of the chosen signal is displayed in the first axes. And the discrete signal is displayed in the other axes when the discrete time-box is checked.

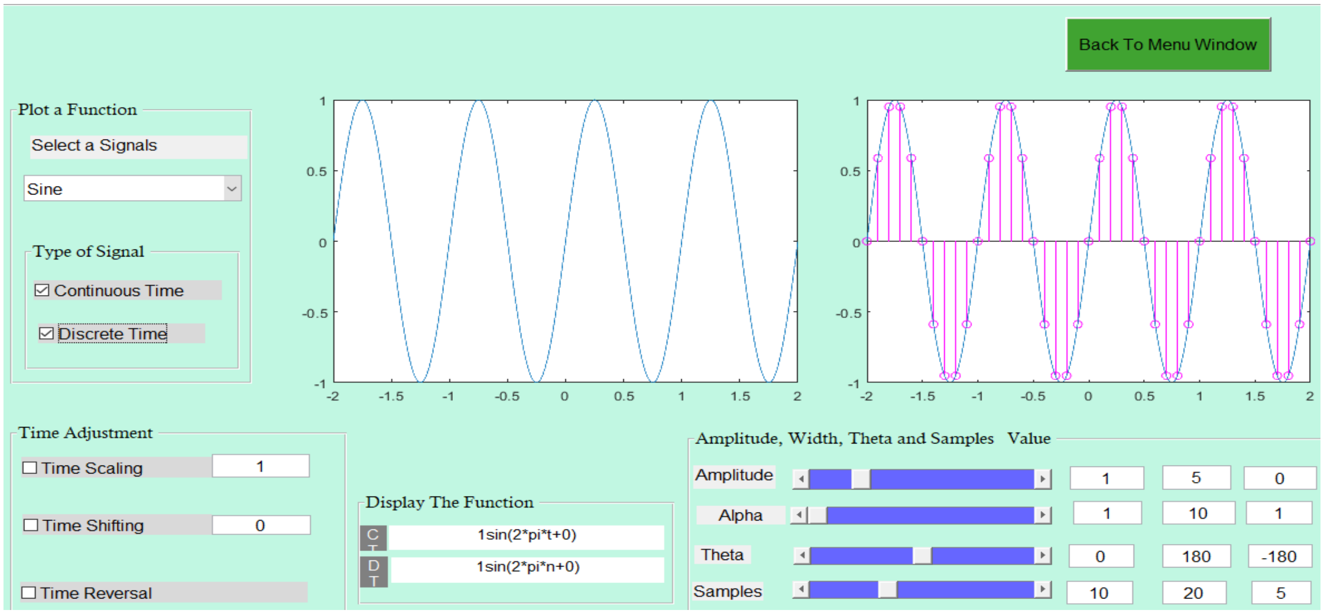

Figure 2.4 Basic Signal function plot

The other boxes in the signal time scaling, shifting and reversal-panel can be edited by the user to see how a signal is changed when it is scaled, shifted or reversed.

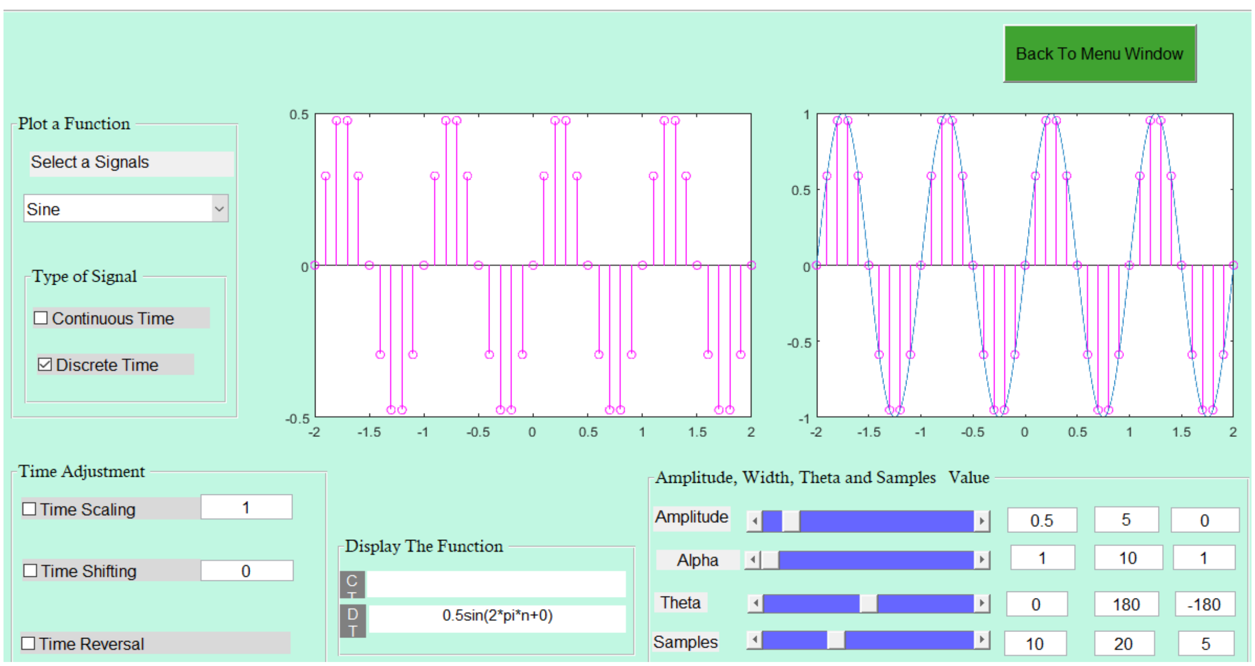

Figure 2.4.1 Basic Signal function plot

If the 'continuous-Time box' is unchecked, the sampled signal is displayed on the axes to the left and the function is displayed in the 'display the function'.

Sampling and Quantization

Sampling is the process of deriving a discrete-time sequence from a continuous time function. The samples are evenly spaced in time. The time interval between samples is called sampling interval. The minimum sampling rate of (2W) samples per second, for an Analog signal bandwidth of W Hz, is called the Nyquist theorem.

Quantization: is the process of approximating (rounding –off) the sampled values to a finite number of discrete amplitude levels.

When the "SAMPLING AND QUANTIZATION" button is pressed, the following window pops up. This window has all necessary edit-text, axes and push-buttons to define, edit and plot signals according to the user's choice. Apparently, there is a defined signal in the 'Define Message Signal' Text-edit. This can be used by the user to plot and change 'quantization level and frequency' to observe how the signal changes with the variation of these values. The message signal ' @(t)sin(20*pi*t) + sin(100*pi*t) + sin(200*pi*t)' is plotted in the axes to the left. And the sampled signal is plotted on the axes to right when the 'plot sampled signal' button.

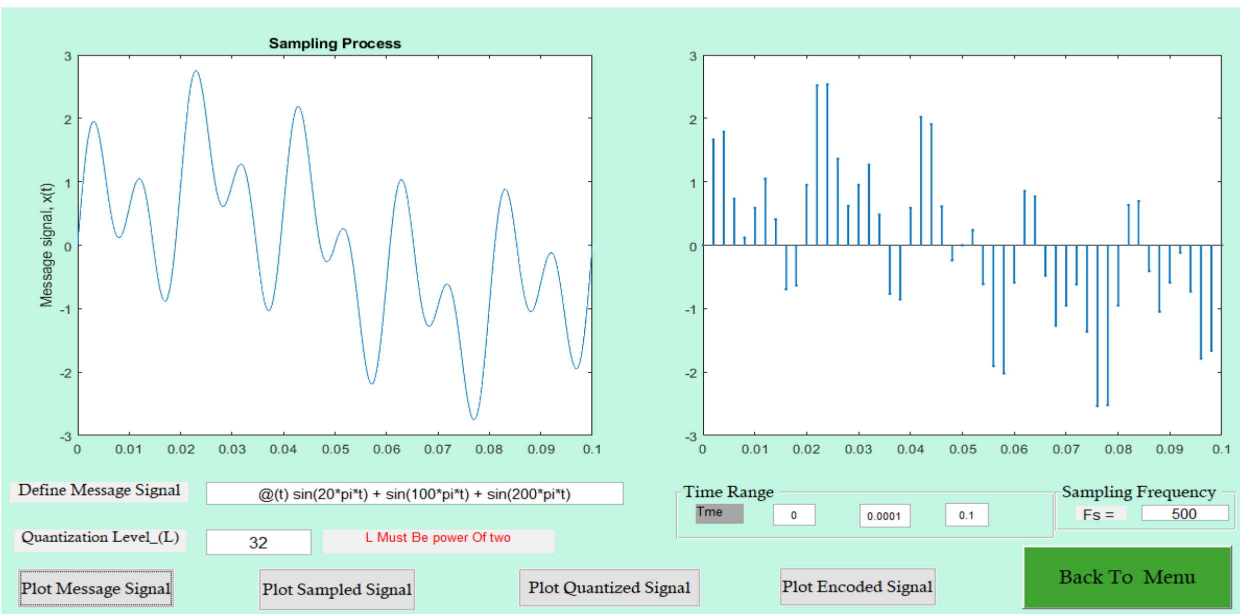

Figure 2.5 Sampling and Quantization

If the frequency on which the signal is sampled is increased, the signals starts to look smoother and almost similar to the original signal. The above Fig. show that the given signal is sampled at Fs=500. Let's increase the sampling frequency (Fs) to 1500 and see how it changes the sampled signal. The higher the Fs the smoother the signal gets.

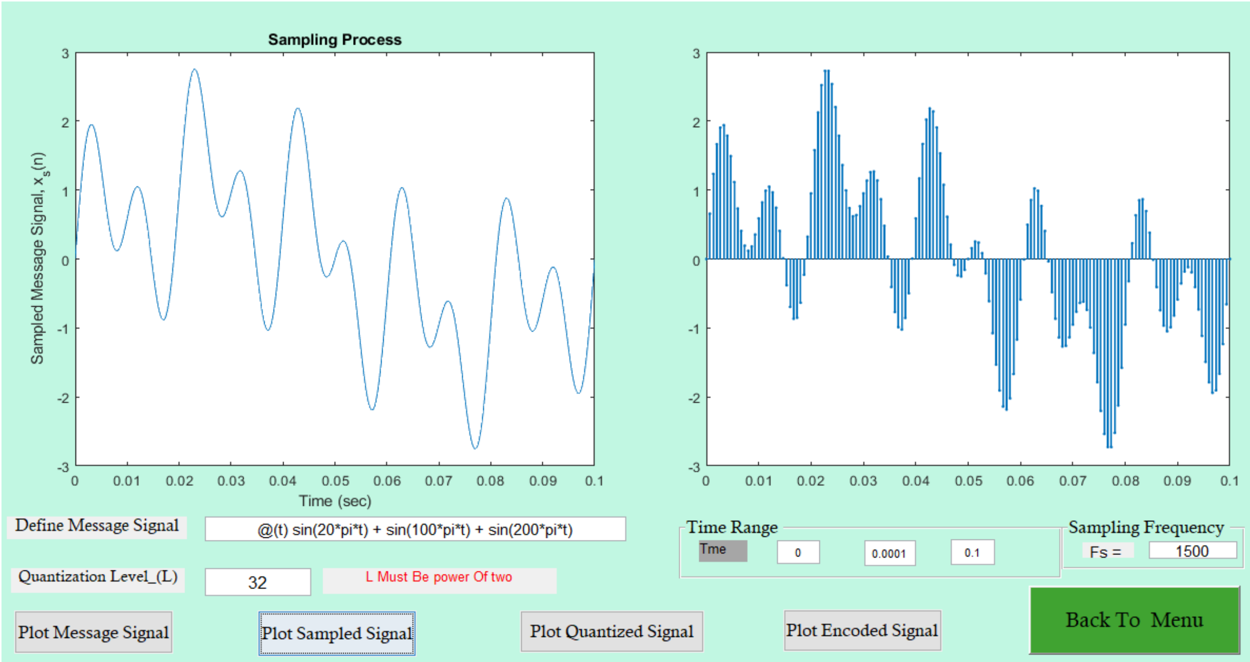

Figure 2.5.1 Sampling and Quantization

2.1.2 <u>Basic Signal systems block</u>: - it consists of convolution (both Discrete-time and Continuous-time, modulation (both analog and digital) and filters.

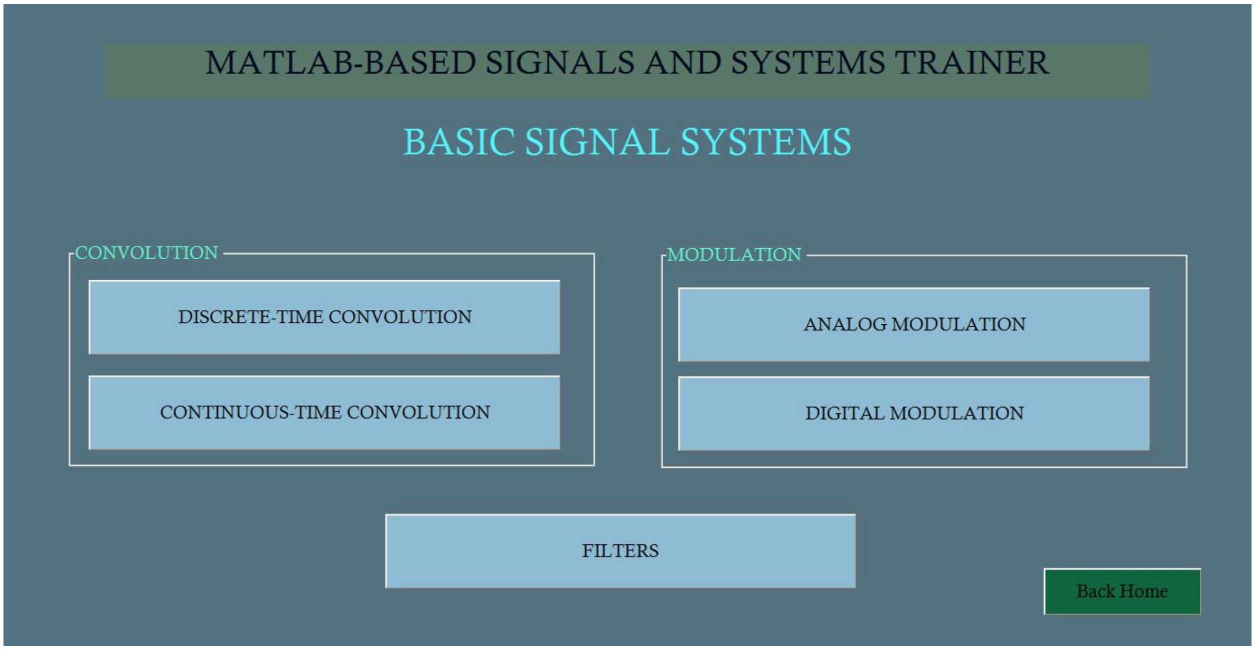

Figure2.6 Basic Signal systems block

Convolution

The idea behind convolution is that if we know the impulse response of the linear system, then we can find the output for any input. From the above definition, we see that there are two basic terms that needs to be defined before proceeding any farther.

- Linear System:- A linear system has a property that the response to a linear combination of inputs is the same linear combination of the individual responses.
- Impulse Signal:- Is a mathematical function that describes the output waveform that results when the input is excited by a unit impulse. The impulse function is just a function that is zero at all times except for at t=0.

With the linearity property of the system fulfilled, convolution tells how an input signal is modified as it passes through the given Linear Time Invariant (LTI) system. The reason LTI systems are incredibly useful is because of a key fact: if you know the response of the system to an impulse, then you can calculate the response of the system to ANY input. This gives you an enormous amount of predictive power. Convolution can be either discrete or continuous. The layout lets the user inter his own input function to be shifted.

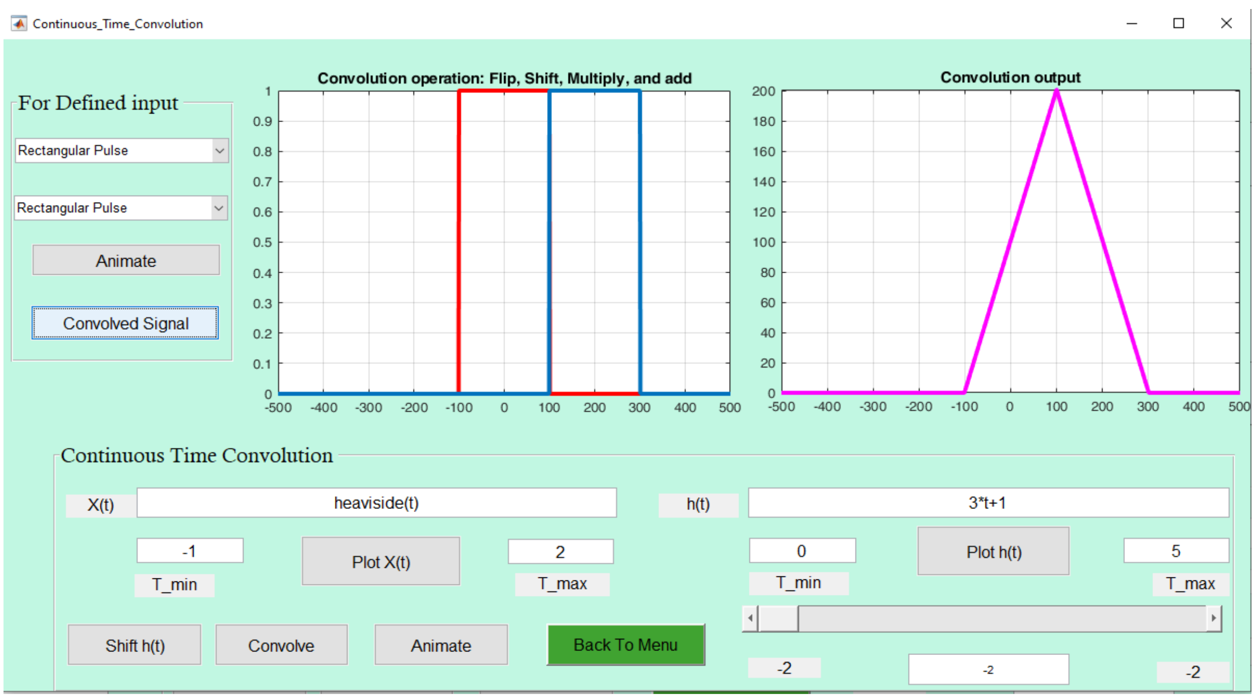

Figure 2.6.1 Continuous time Convolution

Modulation

changing one or more of the characteristics of the high frequency signal (known as the carrier signal) based on the value of another signal (known as the information or modulating signal) to produce a modulated signal.

- The message or modulating signal may be either: analogue or digital (*i.e.* sequences of 1's and 0's)
- The carrier could be a 'sine wave' or a 'pulse train'.

Analog Modulation: A higher frequency signal is generated by varying some characteristic of a high frequency signal (carrier) on a continuous basis. The three types of analog modulation are:-

- Amplitude Modulation (AM):- varying the amplitude of the carrier based on the information signal as done for radio channels that are transmitted in the AM radio band.
- Frequency Modulation (FM):- varying the frequency of the carrier based on the information signal as done for channels transmitted in the FM radio band.
- Phase Modulation (PM):- varying the phase of the carrier based on the information signal.

Digital Modulation: Signals are converted to binary data, encoded, and translated to higher frequency.

- Amplitude shift-keying (ASK):- the carrier frequency and carrier are both maintained constant, while the in which carrier amplitude is keyed between the two possible values used to represent symbols 0 and 1.
- Phase-shift keying (PSK):- in which the carrier amplitude and carrier frequency are both maintained constant, while the carrier phase is keyed between the two possible values (e.g., 0° and 180°) used to represent symbols 0 and 1.
- Frequency-shift keying (FSK):- in which the carrier amplitude and carrier phase are both maintained constant, while the carrier frequency is keyed between the two possible values used to represent symbols 0 and 1.
- BPSK (Binary Phase Shift Keying):- shifts the carrier sine wave 180° for each change in binary state, a very popular digital modulation scheme.
- 8-PSK(Phase Shift Keying):- uses eight symbols with constant carrier amplitude 45° shifts between them, enabling three bits to be transmitted for each symbol.
- Quadrature amplitude modulation (QAM):- is a combination of ASK and PSK so that a maximum contrast between each signal unit (bit, dibit, tribit, and so on) is achieved.

- ➢ <u>Quadriphase-shift keying (QPSK)</u>:- as with BPSK, information carried by the transmitted signal is contained in the phase of a sinusoidal carrier. In particular, the phase of the sinusoidal carrier takes on one of four equally spaced values.
- ➢ <u>Differential phase shift keying (DPSK)</u>:-_both amplitude-shift keying and frequency-shift keying lend themselves naturally to noncoherent detection whenever it is impractical to maintain carrier-phase synchronization of the receiver to the transmitter. But in the case of phase-shift keying, we cannot have noncoherent detection in the traditional sense because the term “noncoherent” means having to do without carrier-phase information. To get around this difficulty, we employ a “pseudo PSK” technique known as *<u>differential phase-shift keying</u>* <u>(DPSK)</u>, which, in a loose sense, does permit the use of noncoherent detection.
- ➢ <u>On Off Keying (OOK)</u>:- OOK is modified version of ASK modulation. In OOK modulation there is no carrier during the transmission of logic zero. The carrier is transmitted during the transmission of logic one.

<u>Filter</u>: - is a frequency-selective system that is used to limit the spectrum of a signal to some specified band of frequencies. Its frequency response is characterized by a passband and a stopband. The frequencies inside the passband are transmitted with little or no distortion, whereas those in the stopband are rejected. The filter may be of the low-pass, high-pass, band-pass, or band-stop type, depending on whether it transmits low, high, intermediate, or all but intermediate frequencies, respectively.

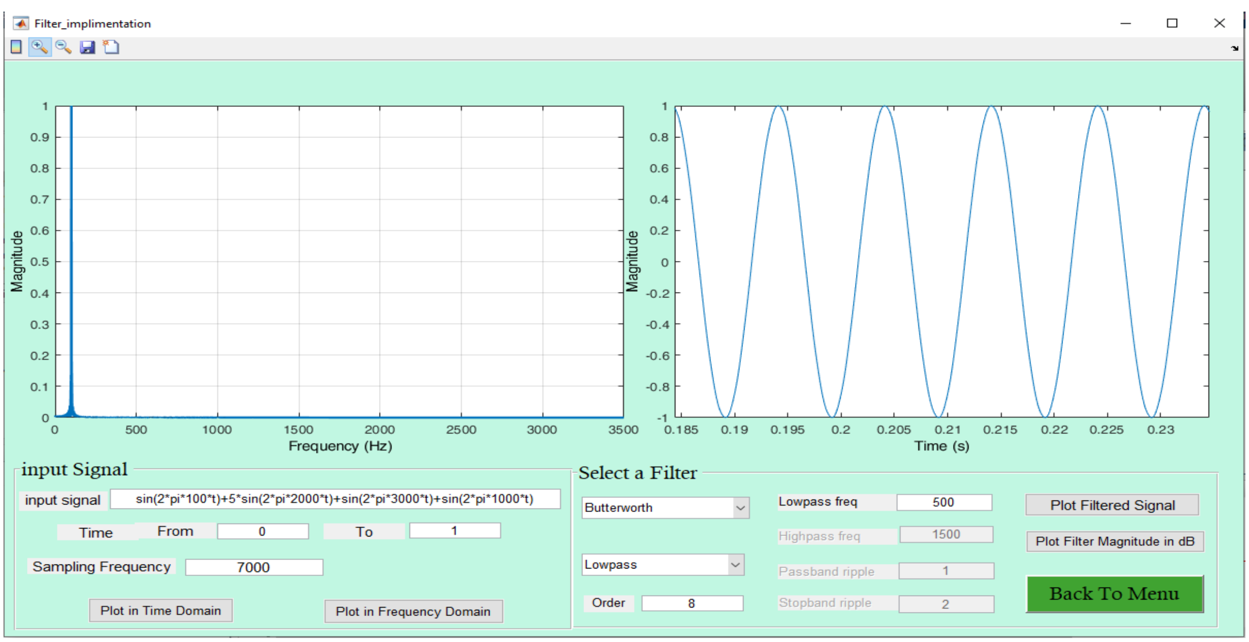

Figure 2.6.1 Example of low pass filter

2.1.3 Signal processing block:- it consists of audio signal processing, image signal processing and Fourier series and transform.

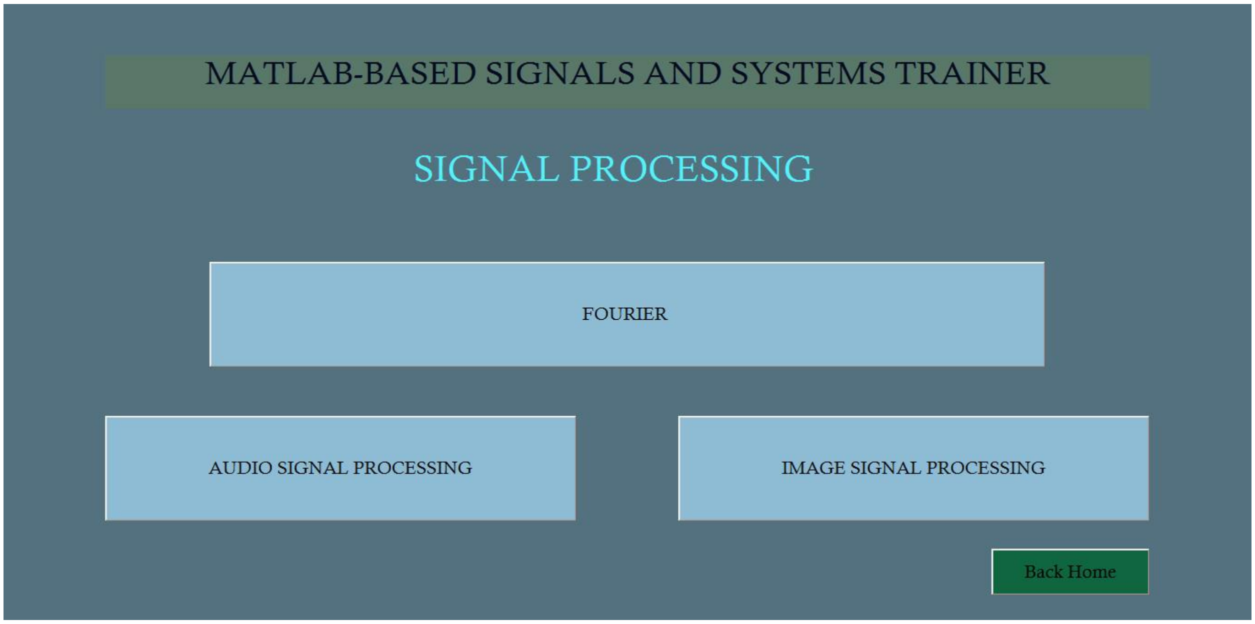

Figure2.7 Signal processing block

Image signal processing:- is a method to perform some operations on an image. In image processing one might be interested in image enhancement, then in image compression and decompression. There are several kinds of noise in digital image. Digital image differs from a photo in that the values are all discrete. A digital image can be considered as a large array of discrete dots, each of which has a brightness associated with it. These dots are called picture elements, or more simply pixels. This project assumes Gaussian, speckle and salt and pepper noises.

As it is mentioned earlier in the objectives, the main purpose of the project MATLAB-Based Signals and Systems Trainer in this specific process is to let students apply the different effects to a digital image and see how those filters modify the image by eliminating the effects. There are three main aspects of image processing: Image enhancement, image restoration and image segmentation. These three aspects can be seen in the image signal processing provided in this project. Of course, the main idea behind those three aspects is real application to digital images. But in our case, as the possible effects and noises that can be added are limited in type and number, the image process is also limited somehow.

The other important feature is types of digital image. Binary, Grayscale and True colour are the three types of digital image.

- Binary Image:- Each pixel is just black or white. Since there are only two possible values for each pixel (0,1), only one bit per pixel is needed.
- Grayscale:- Each pixel is a shade of grey, normally from 0(black) to 255(white). This range means that each pixel can be represented by 8 bits, or exactly one byte.
- True colour, or RGB: Each pixel has a particular colour; that colour is described by the amount of Red, Green or Blue in it.

Grayscale images can be transformed into a sequence of binary images by breaking them up into their bit-plans. The grey value of each pixel of an 8-bit image is considered as an 8-bit binary word. This is called **Bit plane**.

Spatial Resolution:- is the density of pixels over the image. The greater the spatial resolution, the more pixels are used to display the image.

Histogram:- Given a greyscale image its histogram consists of the histogram of its grey levels; that is, a graph indicating the number of times each grey level occurs in the image. A great deal of information of an image can be inferred from its histogram.

Noise:- is any degradation in the image signal, caused by external disturbance.

- Salt and pepper noise:- are caused by sharp sudden disturbance in the image signal; it is randomly scattered white or black pixels. It can be modelled by random values added to an image.
- Gaussian noise:- is an idealized form of white noise, which is caused by random fluctuations in the signal.
- Speckle noise:- It is a major problem in radar applications. It can be modelled by random values multiplied by pixel values.

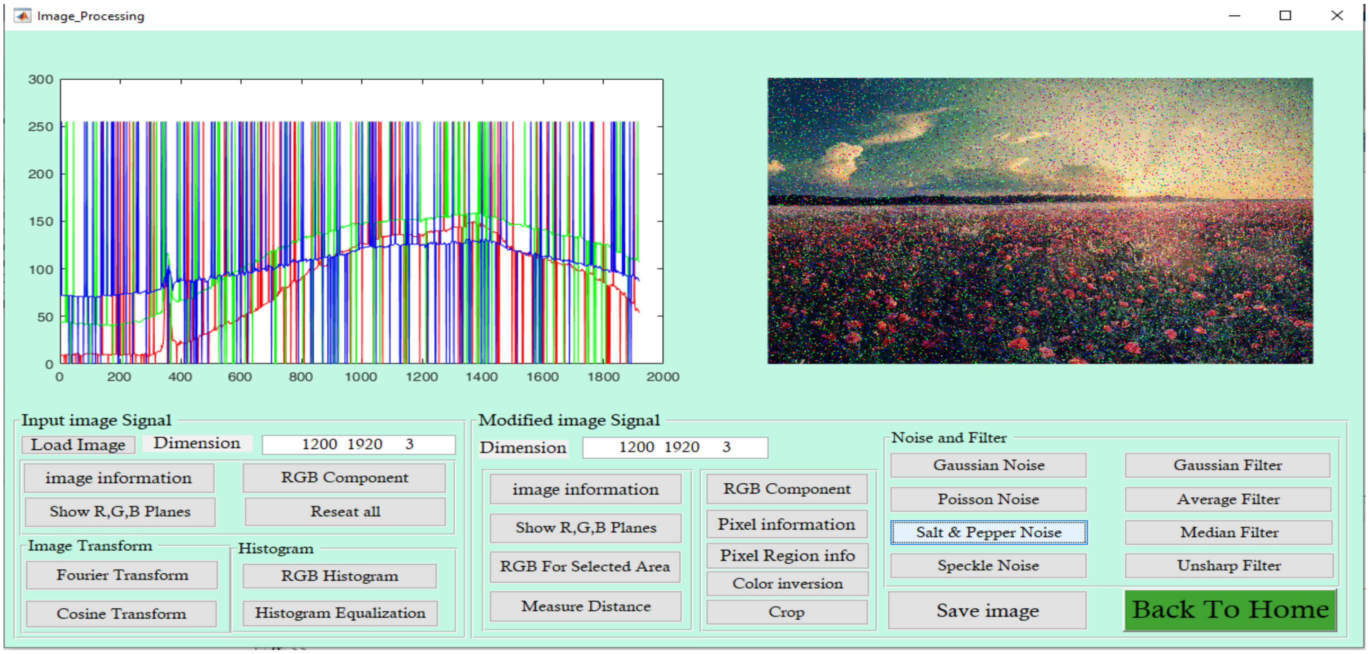

Figure 2.7.1 Image Signal processing block

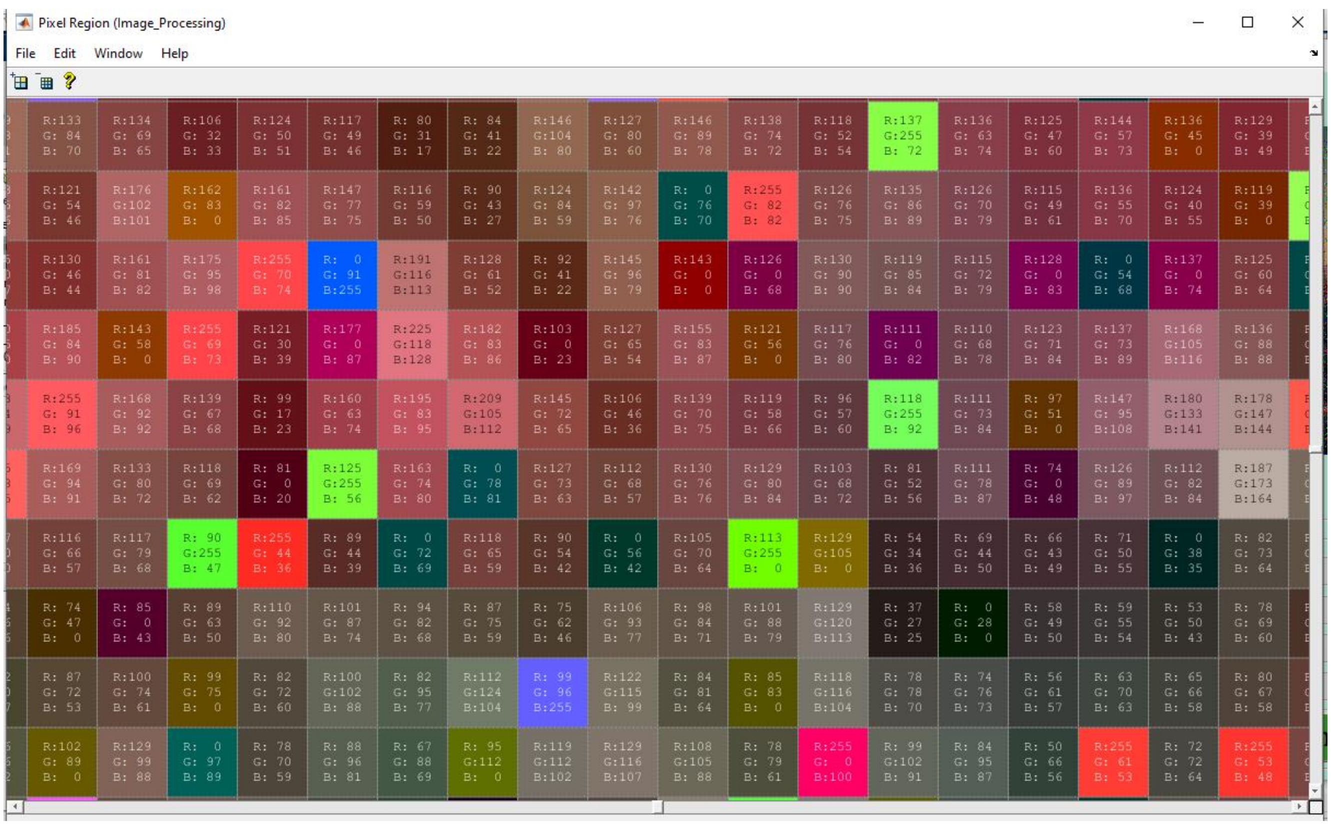

Figure 2.7.2 RGB format for the above image

Audio Signal Processing: - is a subfield of signal processing that is concerned with the electronic manipulation of audio signals. Audio signals are electronic representations of sound waves—longitudinal waves which travel through air, consisting of compressions and rarefactions. The energy contained in audio signals is typically measured in decibels. As audio signals may be represented in either digital or analog format, processing may occur in either domain. Analog processors operate directly on the electrical signal, while digital processors operate mathematically on its digital representation.

Audio signal processing is all about operating audio signals using different methods. Effects can be added to an audio signal once it is loaded. Those effects include white noise and two other defined frequency signal noises; Signal frequency noise and Multi-frequency noise. The other major effect is Signal to Noise Ratio (SNR). By selecting the type of noise and specifying the SNR value, the noise added audio signal can be generated on the axes and the sound heard. This demonstrates the nature of the original audio signal, the noise signal separately on two different axes and the new generated noise added signal.

In audio signal processing, audio electronics and radio communications filters serve a critical role. In this process three classic analogue filters; Butterworth, Chebyshev and Elliptic are deployed.

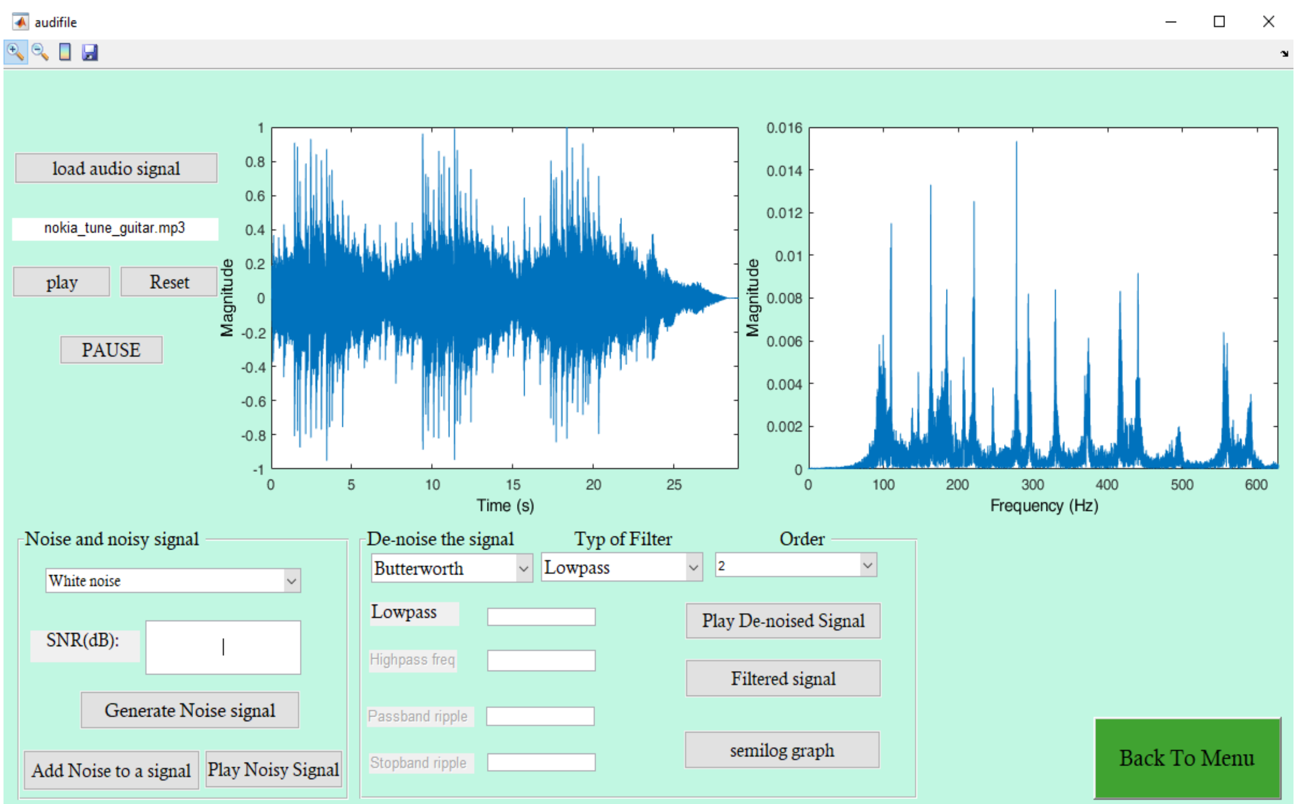

Figure2.7.2 Audio Signal processing block

Fourier

Fourier series:- Fourier series is a branch of Fourier analysis and it was introduced by Joseph Fourier. Fourier series is that any periodic waveform can be represented with a sum of harmonically related sinusoids (i.e sines and cosines with different frequencies and amplitudes).

- It uses sine and cosine because cosine and sine form an orthogonal basis for the space of continuous, periodic functions.
- Bandwidth Concerns: - It's important to note that most periodic signals are composed of *an infinite sum* of sinusoids, and therefore require an infinite bandwidth to be transmitted without distortion. Unfortunately, no available communication medium (wire, fiber optic, wireless) have an infinite bandwidth available. This means that certain harmonics will pass through the medium, while other harmonics of the signal will be attenuated. Engineering is all about trade-offs. The question here is "How many harmonics do I *need* to transmit, and how many can I safely get rid of?" Using fewer harmonics leads to reduced bandwidth requirements, but also results in increased signal distortion. These subjects will all be considered in more detail in the future.

Fourier Transform:- is a mathematical operation that breaks a signal in to its constituent frequencies. The original signal that changed over time is called the time domain representation of the signal. The Fourier transform is called the frequency domain representation of a signal since it depends on the frequency. Both the frequency domain representation of a signal and the process used to transform that signal in to the frequency domain are referred to as the Fourier transform.

Chapter Three

3.1 Result

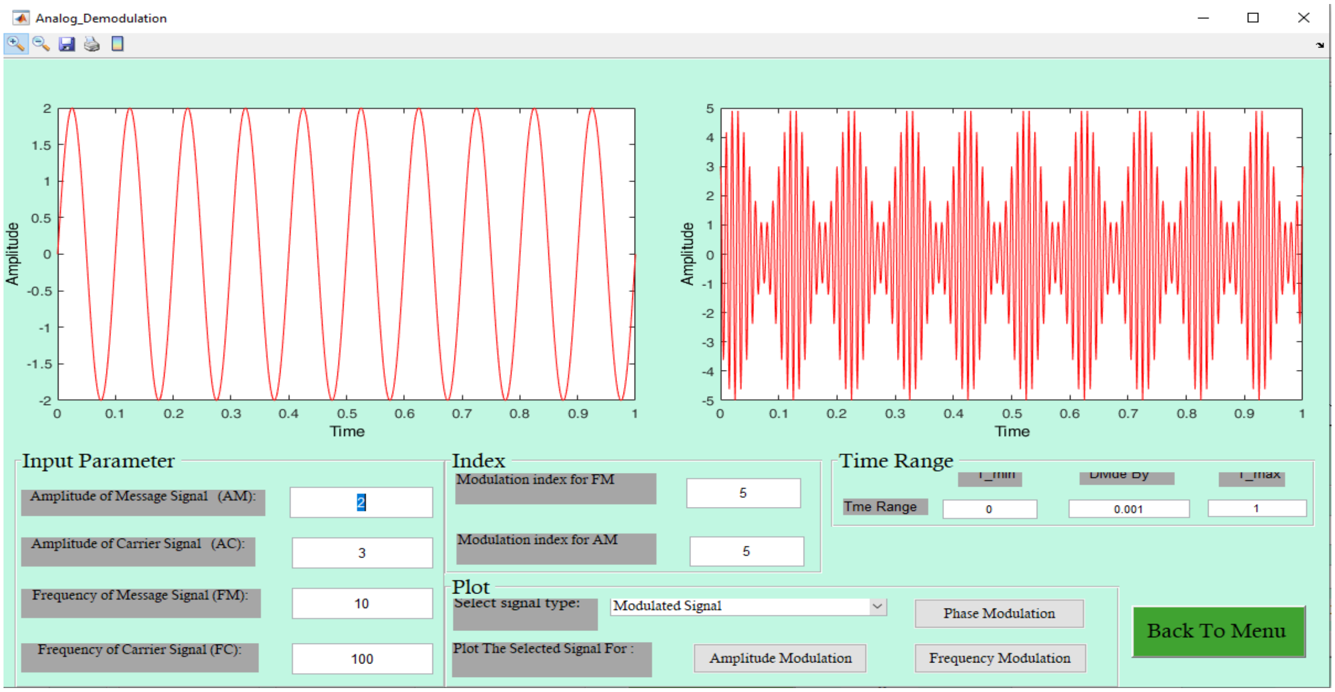

Figure 3.1 Analog modulation

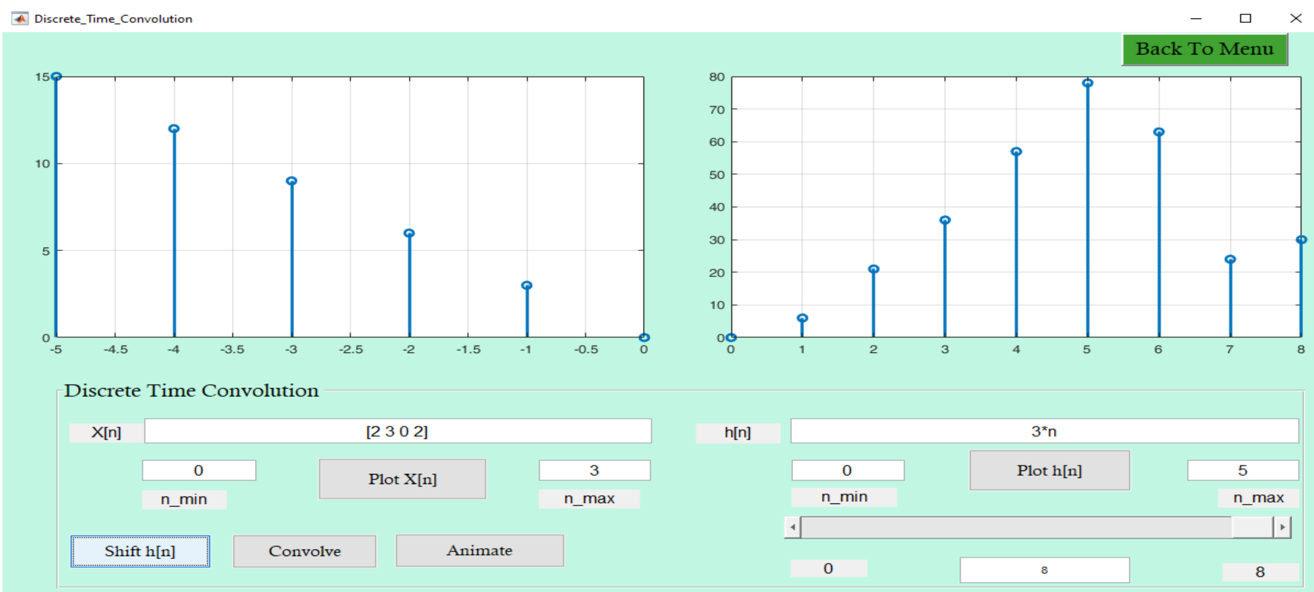

Figure 3.2 Discrete time convolution

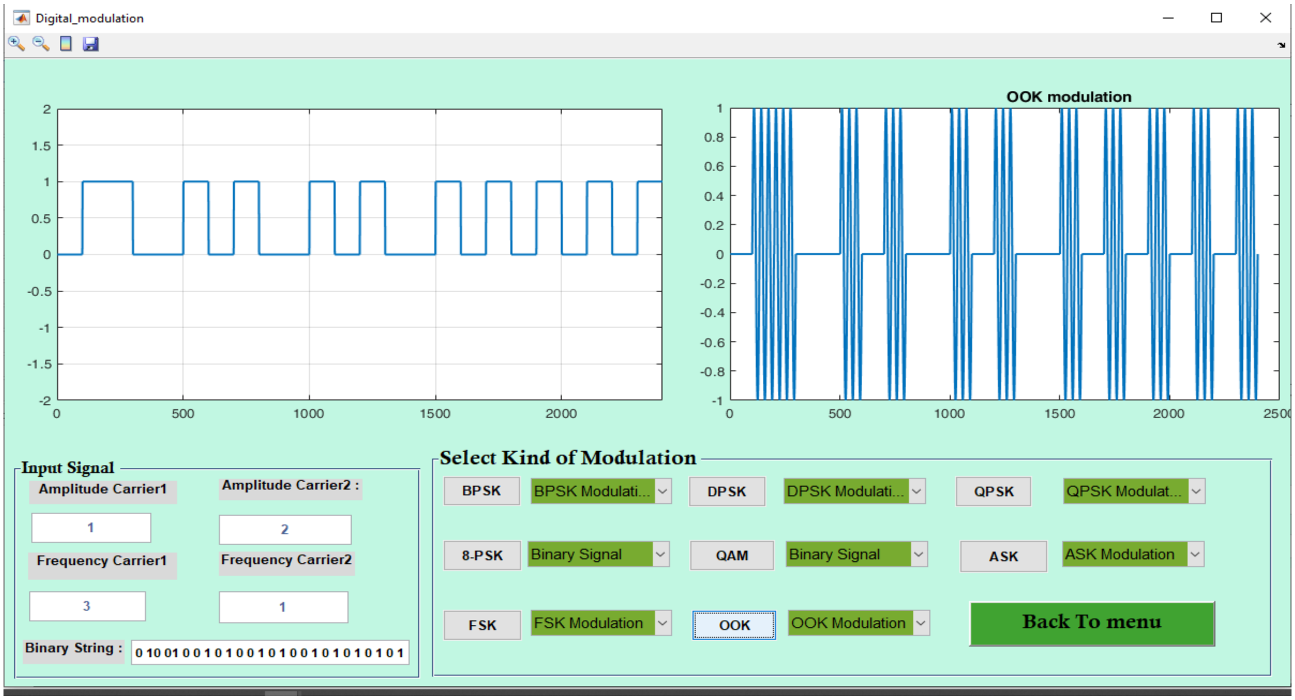

Figure 3.3 Digital Modulation

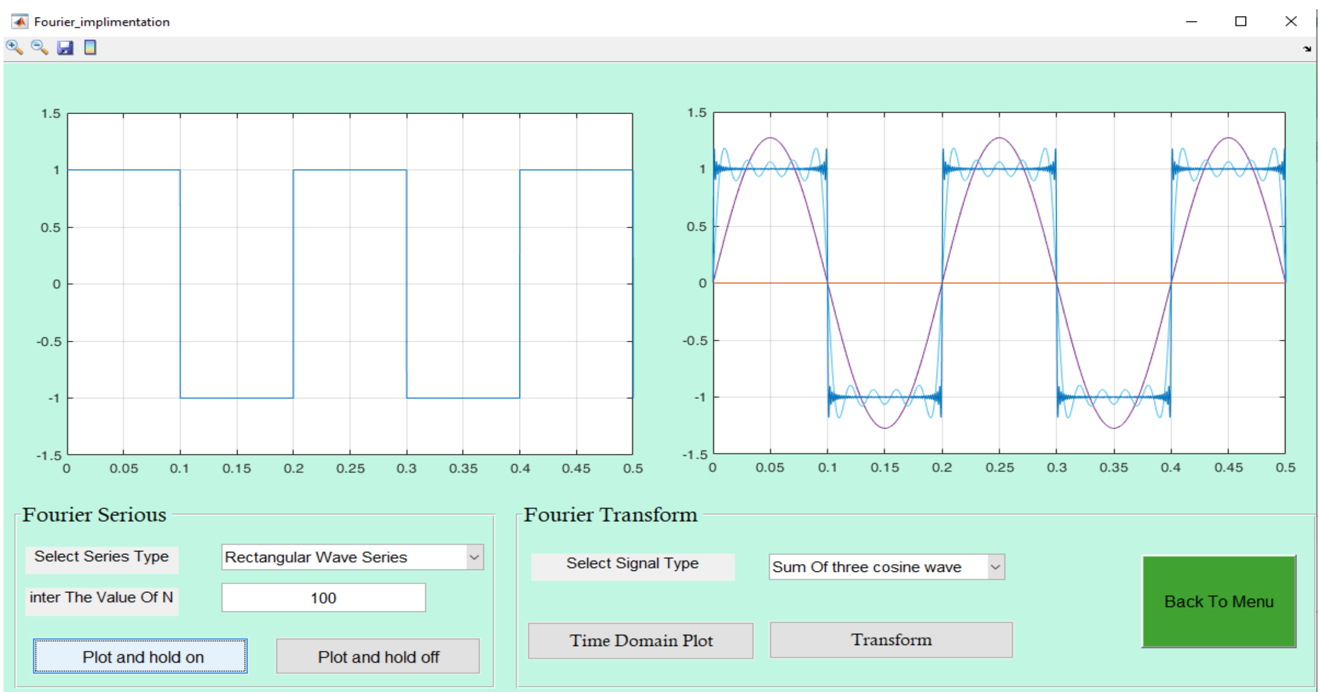

Figure 3.4 Fourier implementation

3.2 Conclusion and recommendation

MATLAB-Based Signals and Systems Trainer has met both the general and specific objectives of the study. It provides quick feedback to sophisticated concepts of signals and systems with less time spent on basics. The results observed from each of the signals, systems and signal processing outputs were satisfying. MATLAB-Based Signals and Systems Trainer will be available publicly on Github.com to everyone to access and modify and for further study. However, in this study, some problems were also encountered. Apparently, Fourier series requires an infinite bandwidth, which is unrealistic to achieve and hence, some of the signals are attenuated. In this matter, it was difficult to accurately identify the value of n where the attenuated signal does not considerably affect the Fourier series to be generated.

3.3 References

[1] A. Nasiruzzaman, A. Nasiruzzaman and A. Nasiruzzaman, “Using MATLAB to Develop a Standalone Graphical User Interface (GUI) Software Packages for Educational Purposes,” in *Matlab-Modeling, Programming and Simulations*, Rajshahi, Afganistan, Rajshahi University of Engineering and Technology, Oct, 2010, pp. 17-40.

[2] L. S. Bob and D. G. Jerry, “Signals and Systems Using MATLAB,” in *Frontiers in Education Conferrence*, 2005.

[3] H. Jiahao, “Signal Analysis Based on MATLAB-GUI,” in *Journal of Physics*, 2019.

[4] B. Mohammed, S. Yassir and L. Abderrahmane, “Teaching Digital and Analog Modulation to Undergradute Information Technology Students,” in *The Future of Global Learning Engineering Education*, Madrid, 2010.

[5] R. Martti, A Computer based education system for signal processing, Espoo: HELSINKI UNIVERSITY OF TECHNOLOGY Faculty of Electrical Engineering, 1996.

[6] R. Richard and K. Sanjeev, “An Integrated MATLAB Suit for Introductory DSP Education,” Princeton University, New Jersey, 2000.

[7] F. Sadikoglu and D. Ibrahim, “Teaching Signals and Systems at Undergraduate Level,” in *9th International Conference on Theory and Application of Soft Computing, Computing with Words and Perception, ICSCCW 2017, 24-25 August 2017*, Budapest, Hungary, 2017.

Appendix

Sample MATLAB code for Audio Processing

```
if (handles.fileLoaded==1)
      if isfield(handles, 'SNR')
            SP=norm(handles.x).^2; % Input signal power
NP=sqrt (SP*10^-(handles.SNR/10)/length(handles.x))*3.455; % Desired noise
      else
            msgbox('Missing SNR! Please fill in the missing values and try
again.','Missing SNR','error');
      end
      S = get (handles. popupmenu1,'Value');
      Switch (S)
            case (1) % white noise
                  handles.Noise = NP*(0.5-rand(size(handles.x))); % adding noise
                  %handles.Noise = wgn(1, length(handles.x), NP);
                  DONE = 1;
            case (2) % signal freq noise
                  f = 300;
                  handles.Noise = NP*cos(2*pi*f*handles.Time');
                  DONE = 1;
            case (3) % multi-freq noise
                  f1 = 200;
                  f2 = 300;
                  f3 = 12000;
                  f4 = 15000;
                  handles.Noise =
0.25*NP*(cos(2*pi*f1*handles.Time')+cos(2*pi*f2*handles.Time')+cos(2*pi*f3*ha
ndles.Time')+cos(2*pi*f4*handles.Time'));
                  DONE = 1;
      end
      if (DONE==1)
            handles.fileNoise = 1;
            handles.x_noise = handles.Noise + handles.x;
            axes(handles.axes1)
            plot(handles.Time, handles.Noise);
            axis([0 max(handles.Time) -1 1]);
            xlabel('Time (s)')
            ylabel('Time')
            axes(handles.axes2);
            Fn = handles.Fs/2;
```

```
        Fy = fft(handles.Noise)/length(handles.Noise);
        Fv = linspace(0, 1, fix(length(handles.Noise)/2) + 1)*Fn;
        Iv = 1:length(Fv);
        plot(Fv, abs(Fy(Iv,1))*2)
        xlabel('Frequency (Hz)')
        ylabel('Magnitude')
    end
end
```