Practical approach of digital filtering applications invariant to time, for Digital Signal Processing course

Pedro Huamaní-Navarrete, Dr
Ricardo Palma University, Peru, phuamani@urp.edu.pe

***Abstract*– In this article, it is presented a practical approach to time-invariant digital filtering applications, through the design and development of a graphic user interface using the GUIDE tool and the Toolbox Signal Processing from the MATLAB software (version 2018). The importance of this work will be to provide students and professors of the Digital Signal Processing course, in the Electronics Engineering and Mechatronics Engineering careers, of the Ricardo Palma University in Lima-Peru, with an interactive tool to improve the teaching and understanding of the subject of digital filtering, allowing to choose from a total of 20 real signals sampled at 44100 Hz., such as voice, medical signal, music, among others, and using multiple options of filter types, cut-off frequencies, noise types, graphic representation of time and frequency, up to the sound reproduction of the signals before and after the filtering stage.**

***Keywords*— digital filters FIR, digital filter IIR, real signals, MATLAB GUIDE.**

I. Introduction

Digital Signal Processing is a discipline that has become one of the main subjects of many engineering careers at the undergraduate and graduate levels. This is due to the need to digitally process signals coming from the real world such as voice, music, various animal sounds, videos, medical images, sonar and radar signals, seismic signals, medical signals such as EEG (Electroencephalogram), ECG (Electrocardiogram), EMG (Electromyogram), etc.; where, in most situations, the objective is the application of a digital filter to reduce the noise, separate signals in frequency bands, store and transmit digital signals, apply frequency transformation algorithms such as FFT (Fast Fourier Transform), DCT (Discrete Cosine Transform), DWT (Discrete Wavelet Transform), among many other applications that today are part of our daily basis usage of technology. To give an example, a cellphone which is able to perform voice recognition, face recognition, fingerprint recognition, as well as store audio, images and video files in compressed formats, plus additional applications that make life easier for the user.

Taking into consideration that the number of applications of digital signal processing is high, in this article it was decided to show the design procedure of a graphical interface for the application of time-invariant digital filters of the recursive type (IIR) and non-recursive type (FIR), and use it on a group of 16 signals digitized using the same sampling frequency. To do this, we used the MATLAB software (version 2018), functions and code examples from the Toolbox Signal Processing [7], the GUI Graphical Interface feature of the software itself, and the vast experience of this article's author on the subject of digital filtering; all of this with the objective of better showing the results of applying filters on real signals, and help the students achieve a greater understanding of the subject during the hours of laboratory class with the MATLAB. In contrast, there are other works done with the LabVIEW software [10] that are improving access to the real world, but depending on an independent hardware and with a certain resemblance to the programming made in MATLAB's Simulink.

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2019.1.1.293
ISBN: 978-0-9993443-6-1 ISSN: 2414-6390

While it is true that the technology that makes use of digital signal processing depend on hardware devices such as FPGA (Field-Programmable Gate Array), DSP (Digital Signal Processor) and Raspberry Pi, it is also important to note that its implementation requires a medium to advance level of knowledge in a programming language, specifically C++. This can be confirmed in works [2] and [9]. For this reason, this article shows the procedure of implementing a graphical interface developed in the MATLAB software, which would be most suitable for professors to present during any simulation classes performed in a laboratory, as it would also imply ease for the students at the moment of choosing the type of filter, the type of signal, and to appreciate the respective frequency spectrums and sound reproduction after the filtering stage. In this way, a greater interest, understanding and performance in the students of the subject of Digital Signal Processing in the Electronics Engineering and Mechatronics Engineering careers of the Ricardo Palma University in Lima-Peru would be achieved.

Therefore, for the understanding of this work, three important sections have been included. The first one referring to the selection of digital signals and filters used in the developed graphical interface, where the programming codes for re-sampling of the signals are included, the design and frequency representation of the invariant filters at the time. Then, the second section refers to the development of the graphical interface using the Matlab software GUI, where the programming used for the three designed panels is described. And, as a final session, the results of the simulation are shown, with four particular cases.

II. Selection of signals and digital filters

A. Selection of digital signal

In order to show the practical application of the digital filtering operation in this article, we chose a group of 20 digitized signals with different sampling frequencies: 8000, 11025, 22050 and 44100 samples per second, and compressed

in two file formats: WAV and MP3. Also, this group of signals was chosen from different internet websites but prioritizing the variety among them. However, due to the different sampling rate that was used in the digitization of each of them, we proceeded to standardize them at a sample rate of 44100 samples per second due to it being the highest frequency of the selected group, and to avoid the presence of aliasing. This signals are: sine/cosine 2 Khz, man's voice, music, happy birthday, ECG signal, EEG signal, sonar signal, bird sound, cockcrow, applause sound, dolphin sound, siren noise, cars noise, bells noise, telephone ring, helicopter propellers, ambulance siren, crying baby, train sound and barking dog.

Additionally, to read the content and sampling frequency of the WAV and MP3 files, commonly used lines of code were called [7]; afterwards, the sampling frequency of some signals was changed with the RESAMPLE command [7], and then stored in a single file SENHALES.MAT together with the single sampling frequency and the time variable that corresponded to 3 seconds of duration. To do this, in some situations the size of the data vector had to be reduced.

```
>> [ telephone, Fs1 ] = audioread('telephone.wav');
>> [ bells , Fs2 ] = audioread('bells.mp3');
>> [ sonar , Fs3 ] = audioread('sonar.wav');
>> [ cockcrow , Fs4 ] = audioread(cockcrow.mp3');
>> [ voice , Fs5 ] = audioread('voice.wav');
Fs5 =
 8000
>> size(voice)
ans =
 24000  1
>> voice = resample( voice , 44100 , Fs5 );
>> size(voice)
ans =
 132300  1
>> save senhales telephone bells sonar cockcrow voice
```

B. Selection of time-invariant digital filters

The time-invariant digital filters were selected of the FIR type (finite impulse response) or non-recursive, and IIR (infinite impulse response) or recursive. This choice was made due to the ease of its design and based on the commands or functions offered by MATLAB's Toolbox Signal Processing.

In the case of the design of the FIR filters, the FIR1 command was used, which allowed it to be designed through the windowing method. That means, starting from the fact of knowing the order of the filter, the sampling frequency and the type of window. Also, from the point of view of functionality, the option of selecting any of the following types of non-recursive filters was added: Low-Pass, High-Pass, Band-Pass and Band-Stop, all using an order number equal to 80, a window Hamming type, and a Sampling Frequency of 44100 Hz as used in the digitized signals. The mathematical expression (1) represents, in a general way, the transfer function of the FIR filter [1], [4]. Where b_0, b_1, b_2, b_3, …, b_{80}, are the transfer function coefficients of non-recursive digital filter. For this, FIR1 command of the Signal Processing Toolbox was used; thus, filter design is Window-based finite impulse response [7].

$$H(z) == b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + ... + b_{80} z^{-80} \quad (1)$$

```
>> help fir1
        Coeff = fir1( N , Fcuttoff , 'ftype' , window )
```

Where "N" variable is the number of filter delays, "Fcutoff" is cut-off frequency as scalar variable or vector variable, and "ftype" is filter's type name. In the case "window" variable, hamming window used by default. And "coeff" variable is a row vector containing the N+1 coefficients of an order "N" FIR filter. In addition, it is necessary to clarify that the filter designed is Type I, because N is odd and "coeff" vector is symmetric.

Following, as an example, the code used to design and display the magnitude of a Low-Pass filter with a cut-off frequency of 2 kHz, and a Band-Pass filter with cut-off frequencies of 4 kHz and 8 kHz are shown; using the content of the website as a guide and reference [7]. This are shown in Fig. 1 and 2.

```
>> Fs = 44100;
>> N = 2^17;
>> order = 80;
>> Fcutt1 = 2000;
>> window = hamming( order + 1 );
>> B1 = fir1( order , Fcutt1 / (Fs/2) , window );
>> [ H1 , F ] = freqz( B1 , 1 , 'whole' , N , Fs );
>> mH1 = 20*log10( abs(H1) );
>> figure(1),
>> plot( F(1:N/2000) , mH1(1:N/2) )
>> Fcutt2 = [ 4000  8000 ];
>> B2 = fir1( order , Fcutt2 / (Fs/2) , window );
>> [ H2 , F ] = freqz( B2 , 1 , 'whole' , N , Fs );
>> mH2 = 20*log10( abs(H2) );
>> figure(2),
>> plot( F(1:N/2000) , mH2(1:N/2) )
>> xlabel(' Frequency (KHz) '),
>> ylabel(' Magnitude (dB) '), grid
```

On the other hand, for the case of the design of the IIR filters, the BUTTER command was called, which uses a designing based on the approximation of the Butterworth analog function. That is, starting from the fact of knowing the order of the filter and the sampling frequency. Also, from the functionality point of view, the option to select any of the following types of recursive filters was added: Low-Pass, High-Pass, Band-Pass and Band-Stop, all using the order number equal to 8, and a Sampling Frequency of 44100 Hz.

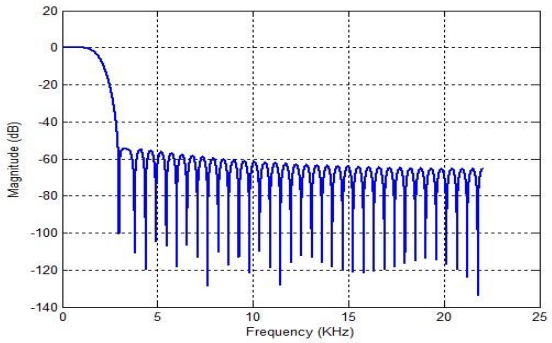

Fig. 1. Magnitude of the Low Pass filter with a cut-off frequency of 2 KHz.

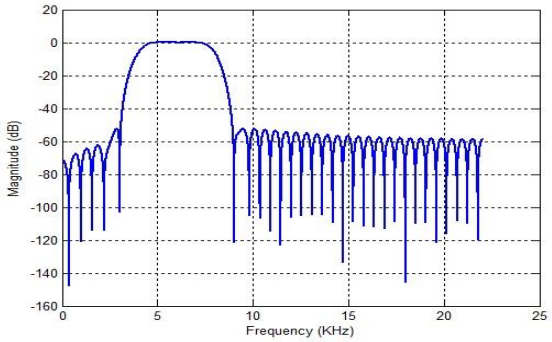

Fig. 2. Magnitude of the Band-Pass filter with cut-off frequency of 4 and 8 KHz.

The mathematical expression (2) represents, in a general way, the transfer function of the IIR filter [1], [4]. Where b_0, b_1, b_2, b_3, …, b_8, a_1, a_2, a_3, …, a_8, are the transfer function coefficients of recursive digital filter.

$$H(z)=\frac{Y(z)}{X(z)}=\frac{\left(b_0+b_1z^{-1}+b_2z^{-2}+b_3z^{-3}+...+b_8z^{-8}\right)}{\left(1+a_1z^{-1}+a_2z^{-2}+a_3z^{-3}+...+b_8z^{-8}\right)} \quad (2)$$

For this, BUTTER command of the Signal Processing Toolbox was used. This Matlab function designs a Butterworth IIR digital filter using the specifications supplied [7].

```
>> help butter
    [ NUM , DEN ] = butter( N , Fcutoff , 'ftype' );
```

Where "N" variable is the number of filter delays, "Fcutoff" is cut-off frequency as scalar variable or vector variable, "ftype" is filter's type name. In addition, "NUM" and "DEN" variables return the filter coefficients in length "N+1" vectors for numerator and denominator, respectively. These coefficients are listed in descending powers of Z.

Likewise, two types of filters were added, called All-Pass filter and Notch filter, both of order 2 and with a straight design from a mathematical expression [8]. In the mathematical expressions (3), a_1 and a_2 are positive coefficients smaller and equal to 1. These allow to place the poles inside the unit circle and to have stable All-Pass filters. On the other hand, in the mathematical expression (4), the G variable is chosen so that gain at low frequencies is unity, w_o variable is the center frequency for Notch filter design, and r variable is radial distance $0 \leq r < 1$. If it is close to the unit circle, the frequency response in deeper and narrower in width [13].

$$H(z)=\frac{Y(z)}{X(z)}=\frac{\left(-a_1+z^{-1}\right)*\left(-a_2+z^{-1}\right)}{(1-a_1\,z^{-1})*(1-a_2\,z^{-1})} \quad (3)$$

$$H(z)=\frac{Y(z)}{X(z)}=G\frac{\left(1-2\cos(w_0)\,z^{-1}+\,z^{-2}\right)}{\left(1-2r\cos(w_0)\,z^{-1}+\,r^2z^{-2}\right)} \quad (4)$$

Then, as an example, the code used to design and display the magnitude of the following recursive filters [7] is shown: High-Pass with cut-off frequency of 7 kHz, All-Pass and Notch filter with a center frequency of 5 kHz. This is shown in Figs. 3, 4 and 5, respectively.

```
>> Fs = 44100;
>> N = 2^17;
>> order = 8;
>> Fcutt1 = 7000;
>> [B1 , A1] = butter( order , Fcutt1 / (Fs/2) , 'high' );
>> [ H1 , F ] = freqz( B1 , A1 , 'whole' , N , Fs );
>> mH1 = 20*log10( abs(H1) );
>> figure(3),
>> plot( F(1:N/2000) , mH1(1:N/2) )
```

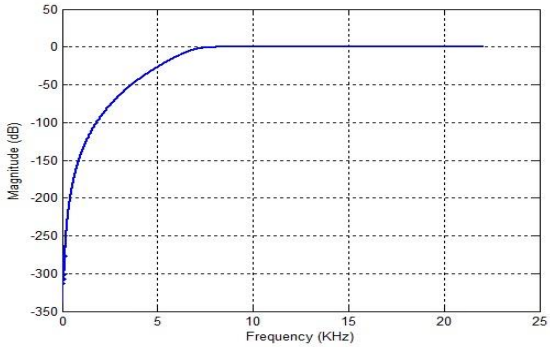

Fig. 3. Magnitude of the High-Pass filter with a cut-off frequency of 7 KHz.

```
>> a1 = 0.2;
>> a2 = 0.7;
>> B2 = conv( [ -a1   1 ] , [ -a2   1 ] );
>> A2 = conv( [  1 -a1 ] , [  1 -a2 ] ) ;
>> [ H2 , F ] = freqz( B2 , A2 , 'whole' , N , Fs );
>> mH2 = 20*log10( abs(H2) );
>> figure(4),
>> plot( F(1:N/2000) , mH2(1:N/2) )
>> axis([ 0 Fs/2000  -350 50])

>> Fc = 5000;
>> G = 0.85;
>> r= 0.85;
>> wo = (2*pi)*( Fc / Fs);
>> B3 = G*[ 1   -2*cos(wo)  1 ];
```

```
>> A3 = [ 1  -2*r*cos(wo)   r*r];
>> [ H3 , F ] = freqz( B3 , A3 , 'whole' , N , Fs );
>> mH3 = 20*log10( abs(H3) );
>> figure(5),
>> plot( F(1:N/2000) , mH3(1:N/2) )
>> xlabel(' Frequency (KHz) '),
>> ylabel(' Magnitude (dB) '), grid
>> axis([ 0 Fs/2000  -100 10])
```

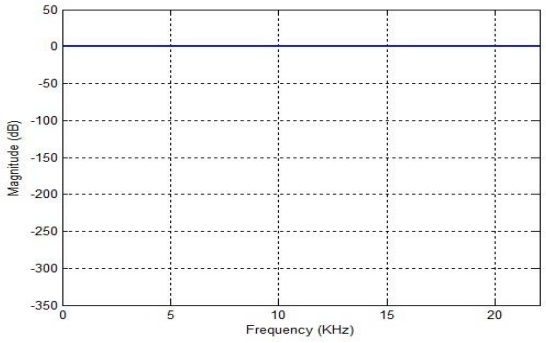

Fig. 4. Magnitude of the All-Pass filter.

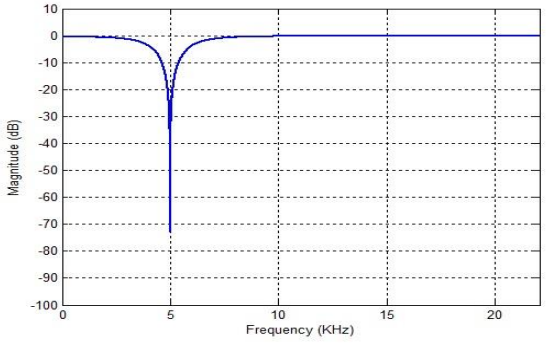

Fig. 5. Magnitude of the Notch filter with a center frequency of 5 KHz.

As shown in Fig. 4, the All-Pass filter passes the frequencies in their entirety. While in Fig. 5 the Notch filter passes all frequencies except that of 5 KHz.

III. Development of the Graphical Interface

For the design and development of the graphic interface, the GUIDE tool (Graphical User Interface Development Environment) was used, which improves the interaction with the student by using object-oriented programming and friendly interfaces composed of buttons, text fields, selection boxes, among others. This paper has taken as reference a work of graphic interface with Matlab for images [11], [12].

In this way, the GUIDE tool made it possible to design a GUI (Graphical User Interface) in an easy and appropriate way, reducing the programming task to the degree of selecting, dragging and customizing properties [5]. Therefore, to develop the graphic interface of this proposal, objects such as Push Button, Radio Button, Button Group, Check Box, Pop-up Menu, Axes, Edit Text, Static Text, Toggle Button and Panel were used [6]. Next, a block diagram of the simulation graphical interface is shown. In this diagram, there is 3 blocks mains that working one after one. Then, every block main has some options. The first block allows to select one of the 20 signals, as well as to add or not two types of noise. In addition, in this block, there is the option of playing the audio of the chosen signal and see time or frequency graph. The second block allows filter select and see graphics on the frequency. And, the third block allows to select the graph of the filtered signal in the time domain or the frequency. See Figs. 6a, 6b and 6c.

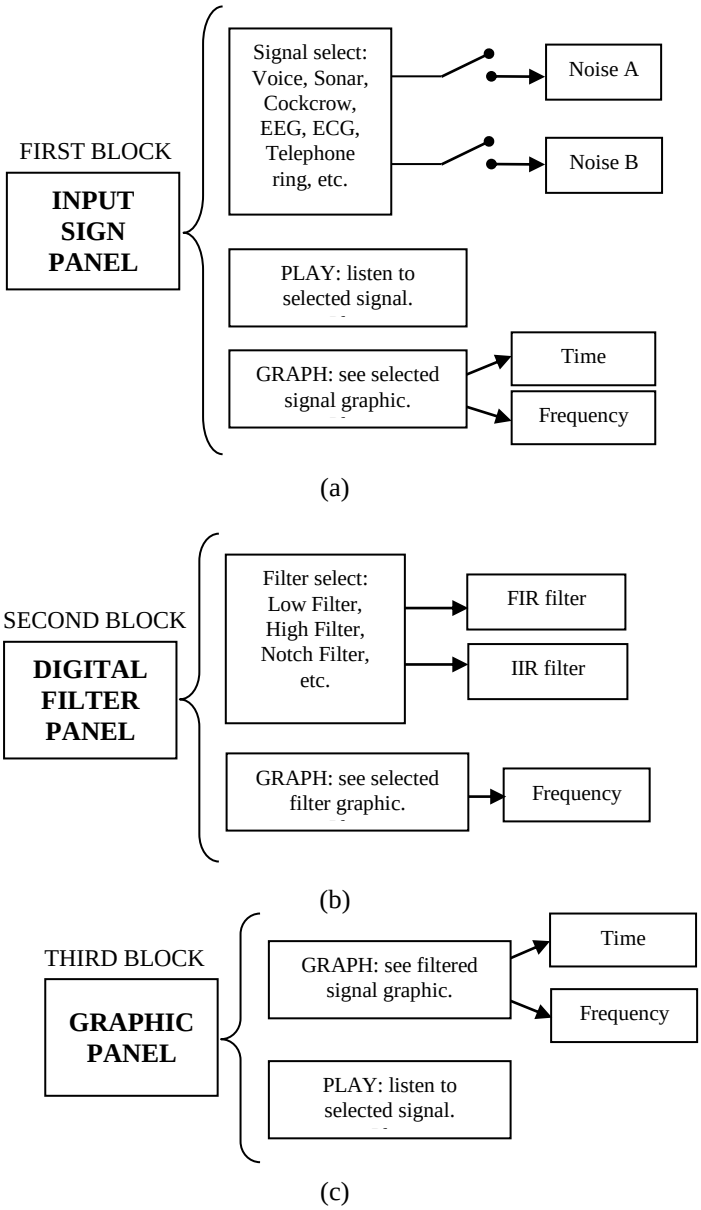

Fig. 6. Diagram of the Graphical Interface for Filtering Digital. a) First block. b) Second block. c) Third block.

A. *Input Signal Panel*

In this panel three elements were inserted, a Pop-up Menu and two Check Boxes. The Pop-up Menu was loaded to list the 16 real signals sampled at 44100 Hz, with the purpose of choosing one of them for the application of digital filtering. Once the desired signal was chosen, the interface proceeds to plot it as a time function in the first Axes. On the other hand, the Check Boxes were used to choose the options of White Noise and/or 1 kHz Tone that can be added to the real signal, chosen from the Pop-up Menu. This allowed observing the performance of the digital filters in the presence of two noises mixed with the original signal, one of the random types and the other deterministic. An example of the SINE/COSINE selection with White Noise is shown in Fig. 7a, and Fig. 7b shows the graph of that signal, which in turn can be reproduced by clicking on the Push Button labeled PLAY.

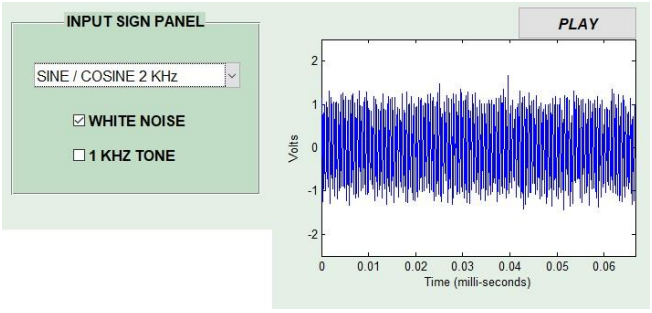

Fig. 7. a) Input Signal Panel of the GUI developed in MATLAB. b) Graph of the chosen signal in time.

Prior to the real signal being selected via the Pop-up Menu, it is necessary to open and read the file SENHALES.MAT. Then, the variable corresponding to the chosen signal is assigned to a global variable. To do this, the variables "x" (filter input signal), "y" (filter output signal), "B" (vector with the numerator coefficients of the filter transfer function), and "A" (vector with the coefficients of the denominator of the filter transfer function) were globally declared in the function OpeningFcn. Next, the MATLAB code used for such a case is shown.

```
function P_OpeningFcn(hObject, eventdata, handles, varargin)
    global x;
    global y;
    global A;
    global B;
```

Then, the following program code consisting of 20 "case" was entered into the Callback function of the Pop-up Menu.

```
global x
Fs = 44100;
N = 2^17;
F = linspace( 0 , Fs , N );
inf =  get(hObject,'Value');
axes(handles.axes1);
load senhales
RB = get(handles.checkRB,'Value');
TONE = get(handles.checkTONE,'Value');
switch inf
  case 1
    x=tone+RB*wgn(length(t),1,0)/5+ …
    TONE*cos(2*pi*1000*t); x = x(1:1470);
    plot( t(1:1470) , x ), axis([ 0 1470/Fs -2.5 2.5])
    xlabel('Time (milli-seconds)'), ylabel('Volts')
  case 2
    x=voice+RB*wgn(length(t),1,0)/5+ …
    TONO*cos(2*pi*1000*t);
    plot( t , x ),
    xlabel('Time (seconds)'), ylabel('Volts')

    . . .

  case 20
    x=train+RB*wgn(length(t),1,0)/5+ …
    TONO*cos(2*pi*1000*t);
    plot( t , x ),
    xlabel('Time (seconds)'), ylabel('Volts')
end
```

B. *Digital Filter Panel*

In this panel eight objects were inserted, a Pop-up Menu, two Static Text, two Edit Text, two Radio Button and a Button Group that participated as a container for the Radio Button. In this way, the Pop-up Menu is able to list the 6 types of filters, in order to choose one of them and apply it to the original signal selected in the previous panel. Then, we continued with the input of the cut-off frequency or frequencies, depending on the case. To represent the text "cut-off frequencies" two Static Text were used, and to enter the numerical values of such cut-off frequencies two Edit Text were used. Likewise, two Radio Buttons were used to allow the selection of the type of filter between FIR and IIR.

Therefore, in the case of the FIR filter, the choice of the Low Pass, High Pass, Bandpass and Bandpass filters were allowed, while in the case of the IIR filter, two more known filters, such as All-Pass and Band-Stop, were additionally allowed.

Once the type of filter was selected from the Pop-up Menu of the DIGITAL FILTER Panel, the frequency response in magnitude in the central Axes was plotted. Said graph was made from 0 to half the sampling frequency. Fig. 8 shows the choice of a FIR Band-Pass filter with cut-off frequencies of 4 kHz and 7 kHz, from the DIGITAL FILTER Panel; and, Fig. 9 represents the frequency spectrum of the selected filter magnitude.

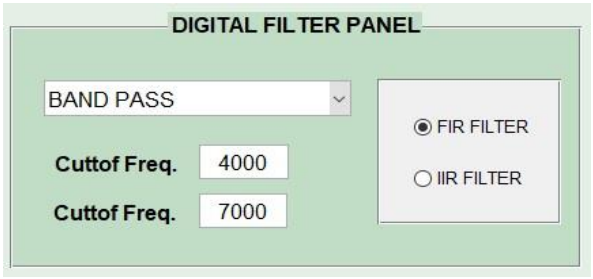

Fig. 8. Digital Filter Panel GUI developed in MATLAB.

Then, with the choice of the input signal and the type of filter, the filtering operation will proceed. Afterwards, the result of said operation will be visualized as a function of time in an Axes type object; and in turn, it will be possible to reproduce said signal by clicking on the Push Button labeled PLAY and located on the time graphic. See Fig. 10.

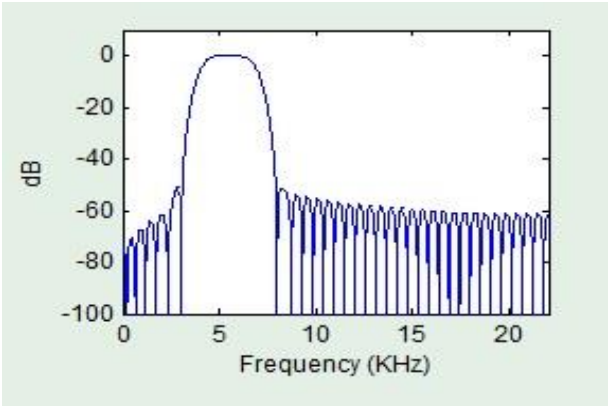

Fig. 9. Frequency spectrum of the magnitude of the selected filter.

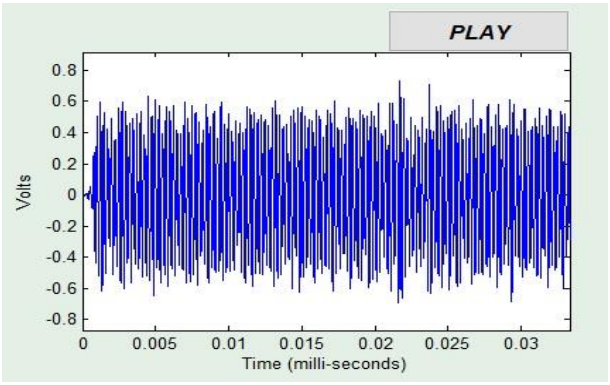

Fig. 10. Temporal representation of the filtered signal.

Next, part of the program code used in the Callback function of the POPUPMENU labeled as POPURMENU3 is shown. As extensive as this code is, it is only showing the design cases of the low pass and high pass filters of both FIR and IIR.

```
function popupmenu3_Callback(hObject, eventdata, handles)
    global A
    global B
    Fs = 44100;
    N = 2^17;
    F = linspace( 0 , Fs , N );
    inf =  get(hObject,'Value');
    Fc1 =  get(handles.editFc1 ,'String'); Fc1 = str2double(Fc1);
    Fc2 =  get(handles.editFc2 ,'String'); Fc2 = str2double(Fc2);
    axes(handles.axes3);
    switch inf
      case 1
          if get(handles.radioButtonFIR, 'Value' ) == 1
            order = 80;
            B = fir1( order , Fc1/(Fs/2) , 'low' ); A = 1;
          else
            order = 8;
            [B,A] = butter( order , Fc1/(Fs/2) , 'low' );
          end
      case 2
          if get(handles.radioButtonFIR, 'Value' ) == 1
            order = 80;
            B = fir1( order , Fc1/(Fs/2) , 'high' ); A = 1;
          else
            order = 8;
            [B,A] = butter( order , Fc1/(Fs/2) , 'high' );
          end
      case 3
        . . .
    end
   [H,F] = freqz( B , A , 'whole' , N , Fs );
   tH = 20*log10( abs(H) );
   plot( F(1:N/2)/1000 , tH(1:N/2) )
   xlabel(' Frequency (KHz)' ), ylabel('dB')
   axis([ 0 Fs/2/1000 -100 10 ])
```

C. *Graphics Panel*

In this panel, two Toggle Buttons elements were inserted to choose the type of graphic representation: time or frequency, as long as the input signal and the digital filter to be used have been selected. See Fig. 11.

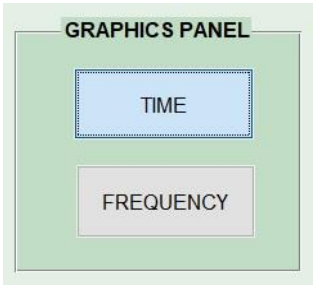

Fig. 11. Graphics Panel of the GUI developed in MATLAB.

Next, the program code used in the Callback function of the Radio Button labeled as TIME is shown.

```
function radioButtonT_Callback(hObject, eventdata, handles)
    global B;    global A;
```

```
global x;     global y;
set( handles.radioButtonF, 'Value' ,0 );
Fs = 44100;          N = 2^17;
y = filter(B,A,x);
axes(handles.axes2);
 M = max(y);  m = min(y);
t = linspace( 0 , length(y)/Fs , length(y) );
plot( t , y )
axis([ 0 length(y)/Fs  1.25*m 1.25*M ])
xlabel('Time (milli-seconds)')
ylabel('Volts')
axes(handles.axes1);
M = max(x);  m = min(x);
plot( t , x ),
axis([ 0 length(x)/Fs  1.25*m 1.25*M ])
xlabel('Time (milli-seconds)'),
ylabel('Volts')
```

On the other hand, the program code used in the Callback function of the second Radio Button labeled as FREQUENCY was the following:

```
function radioButtonF_Callback(hObject, eventdata, handles)
    global B;              global A;
    global x;              global y;
    set( handles.radioButtonT, 'Value' ,0 );
    Fs = 44100;
    N = 2^17;
    y = filter(B,A,x);
    ty = 20*log10(abs(fft(y,N)/N));
    F = linspace( 0 , Fs, N);
    axes(handles.axes2);
    plot( F(1:N/2000) , ty(1:N/2) ),
    axis([ 0 Fs/2000 -100 2 ])
    xlabel('Frequency (KHz)'),
    ylabel('dB')
    tx = 20*log10(abs(fft(x,N)/N));
    F = linspace( 0 , Fs, N);
    axes(handles.axes1);
    plot( F(1:N/2000) , tx(1:N/2) ),
    axis([ 0 Fs/2000 -100 2 ])
    xlabel('Frequency (KHz)'), ylabel('dB')
```

IV. Simulation Results

In order to visualize the results of the application of digital filters type FIR and IIR, 5 signals were chosen random: SONAR SIGNAL, TELEPHONE RING, COCKCROW, APPLAUSE SOUND and TRAIN SOUND, then the program allows to choose the signal to be filtered with or without added noise, followed by the choice of the type of filter with its respective cut-off frequency or frequencies, and finishing with the selection of the graphic form between time or frequency. Additionally, the chosen signal can be reproduced before and after being filtered.

A. First simulation

To the chosen SONAR SIGNAL, a white noise is added. Then, a Low-Pass IIR type filter is selected with a cut-off frequency of 4 kHz, and the Push Button labeled FREQUENCY is clicked to visualize the result in the frequency domain. See Fig. 12.

B. Second simulation

A TELEPHONE RING signal is chosen with no noise is added. Then, the Band-Pass FIR type filter is selected with cut-off frequencies of 2 and 3 kHz, and a click on the Push Button labeled FREQUENCY to visualize the result in the frequency domain. See Fig. 13.

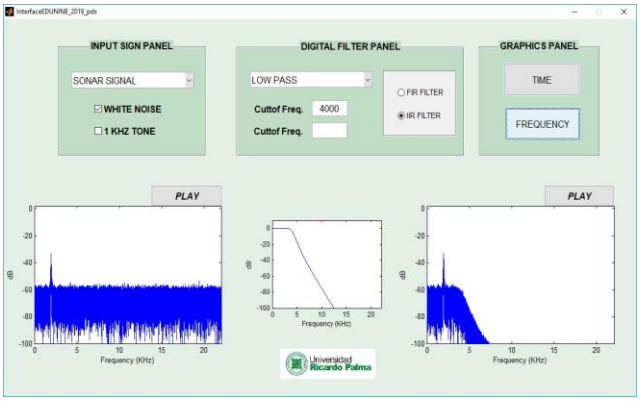

Fig. 12. First simulation complete graphic interface.

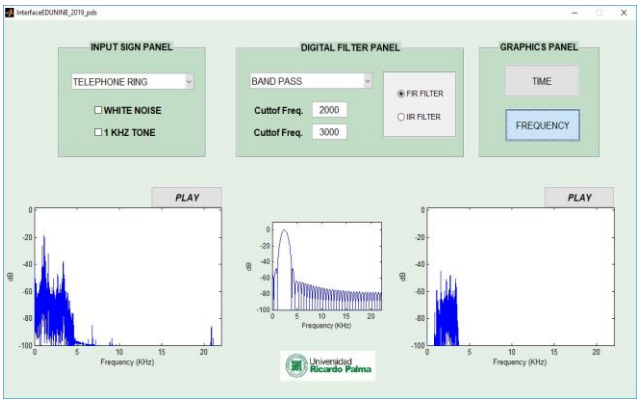

Fig. 13. Second simulation complete graphic interface.

C. Third simulation

To the chosen COCKCROW signal, a one-tone noise is added. Then, a High-Pass FIR type filter with cut-off frequency of 4.5 kHz is selected, and a click on the Push Button labeled TIME shows the result in the time domain. See Fig. 14.

D. Fourth simulation

To the chosen APPLAUSE SOUND signal, a one-tone noise is added. Then, the Notch IIR type filter is selected with a central frequency of 1 kHz, and the Push Button labeled FREQUENCY is clicked to visualize the result in the frequency domain. See Fig. 15.

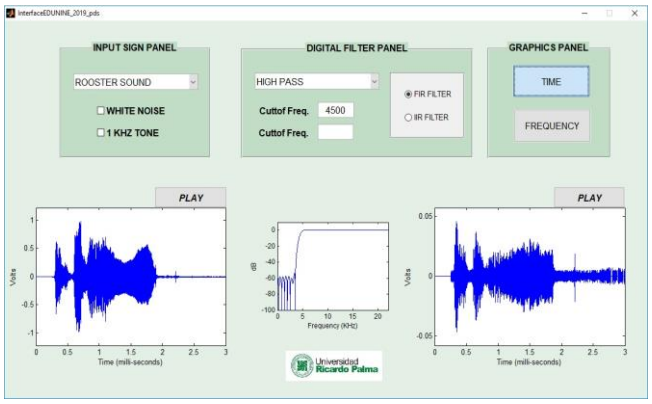

Fig. 14. Third simulation complete graphic interface.

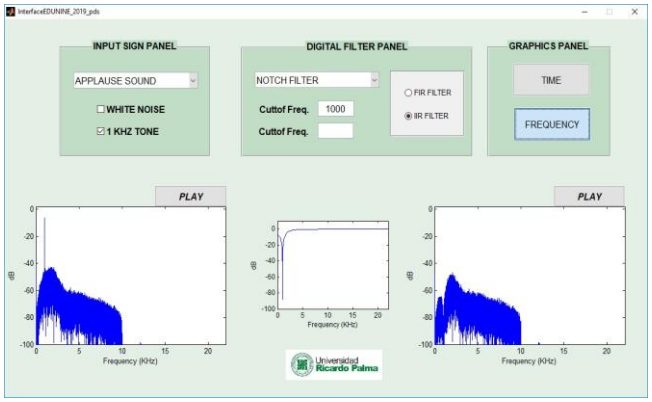

Fig. 15. Fourth simulation complete graphic interface.

E. Fifth simulation

To the chosen TRAIN SOUND signal, a white noise is added. Then, All-Pass type filter is selected, and the Push Button labeled FREQUENCY is clicked to visualize the result in the frequency domain. See Fig. 16.

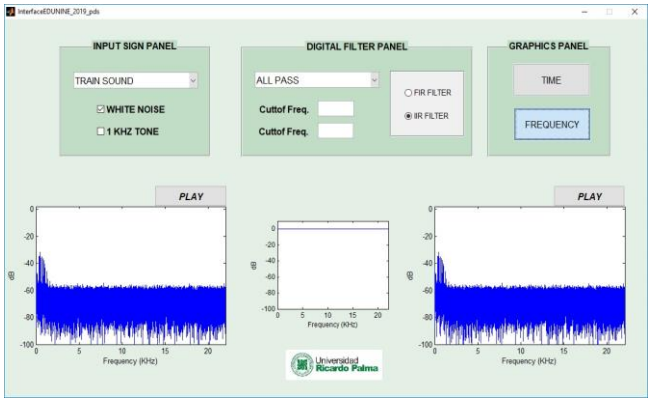

Fig. 16. Fifth simulation complete graphic interface.

Thus, in the hours of laboratory class, the students of the Digital Signal Processing subject of the Electronic Engineering and Mechatronics Engineering careers of the Ricardo Palma University, Lima - Peru, would be able to perform digital filtering simulations with real signals, observing the time and frequency graphs, and listening to the result of the application of the different filters. This would allow to improve the understanding of this topic, as well as giving the users a source code that can be expanded and manipulated by themselves to add new signals of their own interest.

DISCUSSIONS

The graphic interface developed and presented in this article shows certain limitations. Among them, the choice of a fixed order number for the FIR and IIR filters, which restricts the student to observe and compare the performance of digital filtering, when the number of delays increases or decreases. The order 80 for the FIR filter and 8 for the IIR filter were taken randomly, without going over doing the computational effort generated at the time of the filtering operation. Another limitation is related to the designation of a fixed sampling frequency equal to 44100 Hz, this prevents visualizing the presence of aliasing at the time of digital filtering. Also, the design procedure was restricted to using the windowing method for the FIR filters, particularly the Hamming window since it presents an adequate frequency response and is the default option when using the FIR1 function of MATLAB [3], [7]. In the same way, the design procedure for the IIR filters was based on the approximation of the analog function, particularly using the Butterworth function due to its flat response in the passing band and being able to avoid the use of attenuation data at while using the BUTTER function of MATLAB [3], [7].

However, this first version developed by the author can be improved, fixing with the limitations mentioned above, as well as others of great interest that would arise as a recommendation of other teachers, or of the students enrolled.

On the other hand, the use of this computer tool provides substantial benefits in the teaching-learning process, because it will allow students to visualize and hear the result of the application of digital filters using real signals. This in turn will allow, that the evaluations of practices and exams are not only limited to mathematical analysis, but also to a practical observation using graphic representations of the frequency spectrum, both for the digital filter and for the signal of entry and exit of the same. In this way, the understanding of digital filtering applications would be further improved, taught in the laboratory classes of the subject digital signal processing, thus allowing an increase in the level of education and with it a probable increase in the result of the grades.

CONCLUSIONS

The developed graphic interface will allow the teacher of the subject to teach its use in a practical and direct way, with the possibility that he himself or the students themselves, are in the capacity to extend or complement part of the programming code, according to the desired interest.

Likewise, the selection of 20 real signals to apply digital filtering operations was adequate because it will allow the student to observe and hear to the attenuation of a frequency

range or bands, giving the developed interface an interactive characteristic by using appropriate elements to perform the selection of the signal and the digital filter. In addition, due to the simplicity of the programming code in the MATLAB software, it is also possible for the students themselves to add more lines of programming code in order to increase the types of digital filters in the Pop-up Menu of the DIGITAL FILTER Panel, or increase the number of real signals in the Pop-up Menu of the INPUT SIGN Panel.

Finally, this graphical interface will be put to use in the next academic cycle with the purpose of testing and receiving opinions and suggestions from professors and students alike. This can be considered to be a starting point since from this point users can increase more tools, such as zoom, application of filter banks, cascaded and multi-band, visualization of the spectrogram, change of sampling rate, interaction in real time, among countless other applications of digital filtering.

REFERENCES

[1] A. Oppenheim & R. Schafer, Tratamiento de Señales en Tiempo Discreto, 3ra ed. Madrid, España: Editorial Prentice Hall, 2009.

[2] I. Rodríguez, Filtrado Digital de Señal con DSP, Trabajo Fin de Grado, Universidad de la Rioja, España, 2016-2017.

[3] P. Diniz, E. Da Silva e S. Netto, Processamento Digital de Sinais - Projeto e análise de sistemas. 2nd ed. Brasil: Editorial Bookman, 2014.

[4] A. Antoniou, Digital Filters: analysis, design, and signal processings applications, nueva edición. Editorial McGraw-Hill, 2018.

[5] D. Barragán, "Manual de Interfaz Gráfica de Usuario en Matlab", Ecuador 2007. [Online]. Disponible: https://www.dspace.espol.edu.ec/bitstream/123456789/10740/19/%255Bmatlab%255D_MATLAB_GUIDE.pdf

[6] A. Borrero, "Herramienta Software para el Control Remoto de una Fuente de Alimentación mediante Interfaz Gráfica", Proyecto Fin de Carrera, Departamento de Teoría de la Señal y Comunicaciones, Universidad de Sevilla, capítulo 3, p. 56, 2011. [Online]. Disponible: http://bibing.us.es/proyectos/abreproy/11986/fichero/CAP%C3%8DTULO+3%252FCAP%C3%8DTULO+3.pdf

[7] Mathworks R2018b. Signal Processing Toolbox. Funciones. 2018. [Online]. Disponible: https://la.mathworks.com/products/signal/features.html#preprocesamiento-de-se%C3%B1ales

[8] M. Martínez, A. Serrano y J. Gómez, "Introducción al Procesado Digital de Señales", Escuela Técnica Superior de Ingeniería, Departamento de Ingeniería Electrónica, Capítulo 7, 2009-2010. [Onlilne]. Disponible: http://ocw.uv.es/ingenieria-y-arquitectura/1-1/tema7.pdf

[9] J. Ortega y N. Fierro, "Desarrollo de una interfaz interactiva de comunicaciones entre un DSP y Matlab", Síntesis Tecnológica, págs. 39-44, 2004, [Online]. Disponible: http://mingaonline.uach.cl/scielo.php?script=sci_arttext&pid=S0718-025X2004000100007&lng=es&nrm=iso&tlng=es

[10] R. Krneta, D. Damnjanović and M. Doković, "Labview-based laboratory environment for learning of filtering concepts", 6th IEEE International Symposium on Applied Computacional Intelligence and Informatics, Romania, 2011.

[11] Q. Yang, Y. Xv and S.H. Wang, "MATLAB Application in Digital Image Processing Auxiliary Teaching," China Electric Power Education, no. 10, Apr. 2013, pp. 115-116, doi:10.3969/j.issn.1007-0079.2013.10.058.

[12] Yunming Du, Jing Tian, Lina Gai and Wenke Liu, "Digital Image Processing Teaching Auxiliary System based on MATLAB Graphical User Interface" 7th International Conference on Information Technology in Medicine and Education, doi: 10.1109/ITME.2015.67

[13] Wang Chun and Xiao Wei, "Second-order IIR Nothc Filter Design and implementation of digital signal processing system" Proceedings of the 2nd International Symposium on Computer, Communication, Control and Automation (ISCCCA-13), pags 0576-0578, 2013, [Online]. Availabe. https://www.scientific.net/AMM.347-350.729