FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Development of LabVIEW based DSP-application

Bachelor Thesis

Nguyen Viet Dung

This page is intentionally left blank

FRANKFURT
UNIVERSITY
OF APPLIED SCIENCES

Development of LabVIEW based DSP application

by

NGUYEN VIET DUNG

Submitted to Faculty 2: Engineering and Computer Science

In partial fulfillment of the requirement for the degree of

Bachelor of Engineering in Electrical Engineering and Information Technology

at the

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

08 June to 30 August 2021

Author: ....

Nguyen, Viet Dung

Matriculation number: 1273 548

Certified by:...

Manfred Jungke

Professor of Technical Information and Measurement

Thesis supervisor

Certified by:...

Robert Michalik

Master of Science; Lab Engineer

Thesis supervisor

This page is intentionally left blank

Development of LabVIEW based DSP application

by

NGUYEN VIET DUNG

Submitted to Faculty 2: Engineering and Computer Science

In partial fulfillment of the requirement for the degree of

Bachelor of Engineering in Electrical Engineering and Information Technology

at the

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

August 2021

Abstract

In this project, a LabVIEW based program was written for teaching supportive purpose. This program gives a compact way for my supervisor, Professor Jungke, to use in his lectures and deliver it to the students and they can also use the program and have some knowledge about their course of Signal Processing in general. In this thesis, a group of DSP characteristics are reviewed with realistic models, such as windowing functions and digital filters and then how to implement them via LabVIEW.

Keywords: LabVIEW, Window Function, Digital Filter, Spectral Leakage.

Thesis Supervisor: Manfred Jungke; Professor of Technical Information and Measurement

Thesis Co-Supervisor: Master of Science, Laboratory Engineer

To my father, mother, elder brother and Dr. Hien

Disclaimer

I confirm that I have written this thesis on my own. No other sources were used except those referenced. Content which is taken literally or analogously from published or unpublished sources is identified as such. The drawings or figures of this work have been created by myself or are provided with an appropriate reference. This work has not been submitted in the same or similar form or to any other examination board.

30 August 2021

Dung Nguyen Viet, Author

Acknowledgement

This is the report for the last module of my six-semester bachelor's degree. During my first five semesters in the VGU, I would like to give a special thanks to all of my lecturers in Vietnam, especially Dr. Hien Nguyen Minh, who give me enthusiastic lectures and motivation to me to keep studying in the field of Digital Signal Processing. Dr. Hien had cited and introduced many documents from Prof. Jungke, Senior lecturer in the FRA-UAS, who is my first supervisor for this last semester in Germany. I also give gratefulness to M. Sc. Michalik, my second supervisor, who support me in my senior project and this thesis, who enlighten and assistance me in the preparation of this thesis. It is hard to express my appreciation for them because of the opportunity for me to do this project "Development of LabVIEW based DSP application", which I can learn a lot from this topic.

Any achievement in any level cannot be satisfactorily finished without the support and guidance of my parents, my friends in Vietnam and also Germany.

I would like to give a thanks to Thien Nguyen Minh, my homie in FRA-UAS, the monitor of the class, who give me helpful advice in report formation and daily life in Germany. Thien and me have studied together since our first special topic class in 2015, and until now, we have learnt, received many achievements together. A lovely thanks to my girlfriend Phuong Do Truc Lam, despite the different time zone and a worrisome situation of Corona pandemic in Vietnam, she still mentally support me in the thesis period. Another thanks to my eight-year friend Ninh Vuong Cam, my best eight-year friend, who listen to me, give advice and support me a lot when I have my first experience in Germany. Last but not least, give a respect to my big family at my home country. They give me motivation and cover my expense first when I go to Germany to start this exchange semester. Without physical and mental support any of those people, I cannot finish my last module of the undergraduate and have many memorial moments in Germany.

Contents

Table of figures, tables and appendices

This page is intentionally left blank

1. Introduction

1.1 Motivation

Professor Jungke has taught many years on the field of Digital Signal Processing at university. At the Bachelor level, it necessitate some supportive material to visualize the features to students instead of many theoretical classes with equations and formulae. In his lectures of fundamental on filters and window functions, he had an old version program Analysis Advisor for his presentation. However, the problem comes from the incompatibility of the program that it cannot execute in Window 10, which is a common operating system nowadays. Therefore, it is essential to develop a new program with equivalent functions as the old one.

1.2 Program description

The goal of the program is to be a supportive program for Professor Jungke with teaching purpose. In this new program, there some modules in related to the topic of windows and digital filters.

The Discrete Fourier Transform (DFT) is vulnerable to spectral leakage. Spectral leakage is a phenomenon which occurs when a non-integer number of periods of a signal is applied to the DFT. Because of spectral leakage, a single-tone signal (signal that contain only one value of frequency) is spread out among other neighbor frequencies after the DFT is computed. The situation of spectral leakage can solve by applying window function. Several window functions are discussed in the thesis.

In addition, properties of some types of digital filter are presented: median filter, FIR and IIR filter. Each type of filter has their drawbacks and advantages in some special features that the other cannot replace totally.

After discussing about the theoretical background, the next part give information about the implementation into the program via the LabVIEW environment. Each module in the program constructs some experiments to highlight the key points in those aforementioned topics with explanation, so that students after using this program will have ideas about what they are learning.

2. Time schedule

This is the part to report the schedule that been accepted by the co-supervisor, M.Sc. Michalik and applied for the whole thesis and the following table describe clearly about this.

Milestone description	Start day (2021)	Duration (day(s))
Install LabVIEW and virtual machine	07-06	4
LabVIEW basis tutorial	09-06	6
State-machine project	10-06	7
Theoretical review on Windowing function	12-06	10
Theoretical review on FFT	19-06	10
Progress report 1	22-06	1
Old program revision	28-06	10
Report theoretical background (draft)	05-07	6
Program structure	09-07	17
Progress report 2	13-07	1
Progress report 3	20-07	1
Report result (draft)	26-07	5
Program UI and executable file	30-07	10
Report discussion and conclusion	11-08	5
Report writing and correction	15-08	14
Progress report 4	16-08	1
Program demo	19-08	1
Reconstruct of the UI	19-08	5
Progress report 5	24-08	1
Report final formation and correction	29-08	1
Report printing	29-08	1
Report submission	30-08	1

Table 1: Thesis schedule

3. Old version application

The goal of this part is to have an overview and limit the scope of the program. To execute the old-version program, Oracle VM VirtualBox was used to emulate the operating system Window XP 32-bit. The figure below shows the user interface of the program Analysis Advisor, which was developed also by LabVIEW.

The goal of the project is to demonstrate some features in Signal Processing, so the topic of Smoothing Windows, Digital Filters and Statistics was chosen.

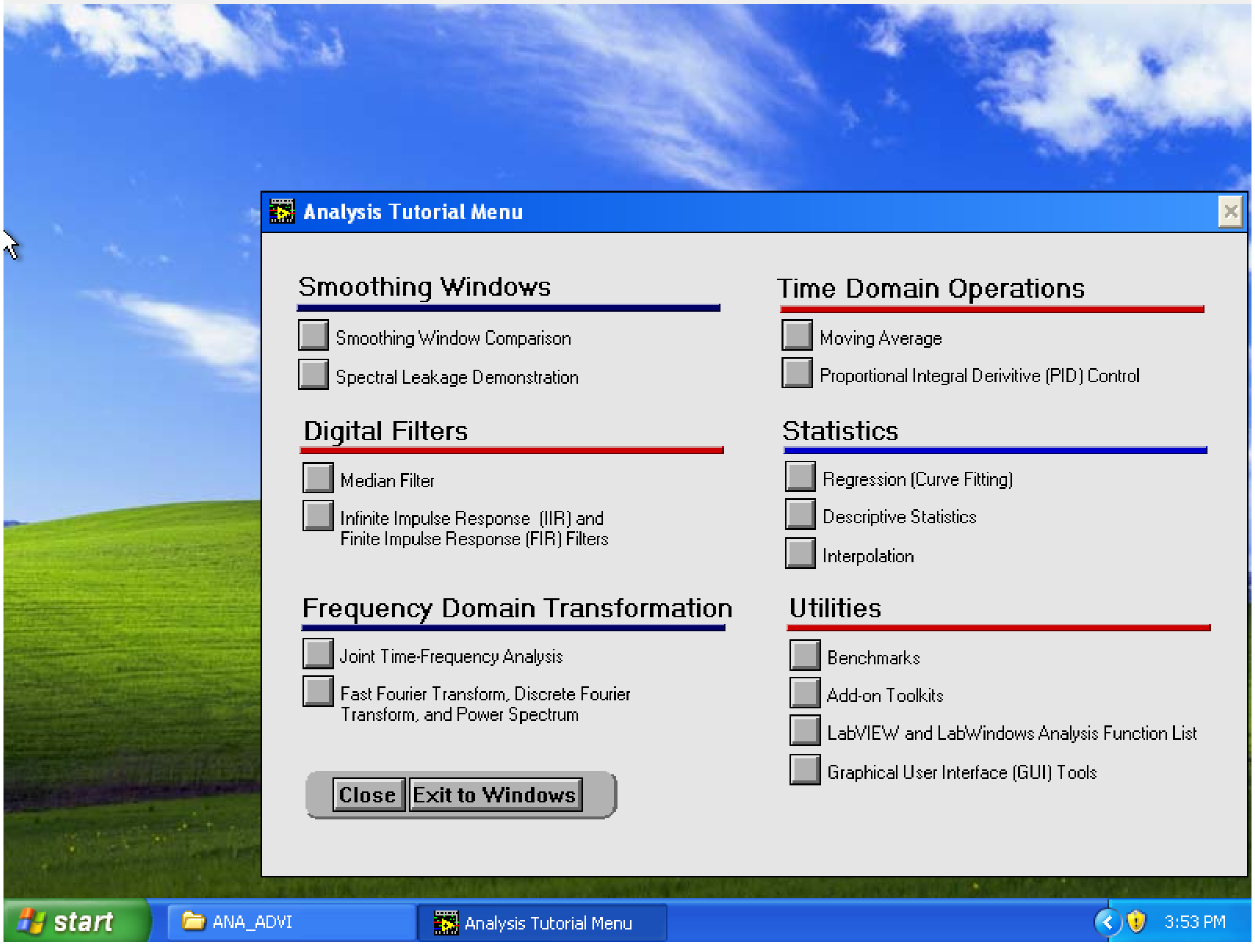

Analysis Tutorial Menu
Smoothing Windows
Smoothing Window Comparison
Spectral Leakage Demonstration
Digital Filters
Median Filter
Infinite Impulse Response (IIR) and Finite Impulse Response (FIR) Filters
Frequency Domain Transformation
Joint Time-Frequency Analysis
Fast Fourier Transform, Discrete Fourier Transform, and Power Spectrum
Close
Exit to Windows
Time Domain Operations
Moving Average
Proportional Integral Derivitive (PID) Control
Statistics
Regression (Curve Fitting)
Descriptive Statistics
Interpolation
Utilities
Benchmarks
Add-on Toolkits
LabVIEW and LabWindows Analysis Function List
Graphical User Interface (GUI) Tools
start
ANA_ADVI
Analysis Tutorial Menu
3:53 PM

Figure 3-1: User interface of old version program

4. Software overview

Laboratory Virtual Instrument Engineering Workbench (LabVIEW) is a high-level programming based on graphical coding language G provided by National Instrument. LabVIEW is used with educational and industrial purposes.

4.1 Why LabVIEW?

LabVIEW provides built-in functions and supportive tools for data acquisition and instrumental controlling, that why it is viral in the field of Signal Processing.

Furthermore, LabVIEW with its high-level programming decrease the difficulties of coding structure and architecture for the users because it brings the visual way of coding based on dataflow and graphical programming, which is mentioned later.

In addition, the user interface (UI) of LabVIEW is a type of WYSIWYG: What You See Is What You Get, helps the developers accelerates the coding procedure and diminish the stress for engineer in any aspect of an engineering project: hardware and software synchronization, data measurement, debugging, and UI customizing.

Last but not least, any executable files in LabVIEW require only LabVIEW Run Time Engine to perform, which is compact in size and easy for lecturers if they want to distribute their educational implements to students, as same as in this project.

4.2 Graphical Coding

People are familiar with textual programming. In the field of Engineering, MATLAB is also a supportive tool which are comparable to LabVIEW. MATLAB programming language and many others are written ones, based on geometrical design with many regulations by indentation (like MATLAB), syntactic order and highlighting. LabVIEW programming is even a higher-level program than MATLAB, because it brings visualization and users simply drag and drop function boxes and link them together in a logical way which is similar to textual programming.

4.3 Basics of LabVIEW and Dataflow Programming Language

In this part, some basic detail on how dataflow programs are operated with numerous examples.

4.3.1 Layout

LabVIEW program is called an VI, which stands Virtual Instrument, which including two windows: Front Panel (the grey one) and the Block Diagram (the white one) in Figure 4-1. Front Panel is where the users interact with the program, give inputs and receive the results. Block Diagram is the zone to manage the code by dragging built-in blocks in an appropriate way.

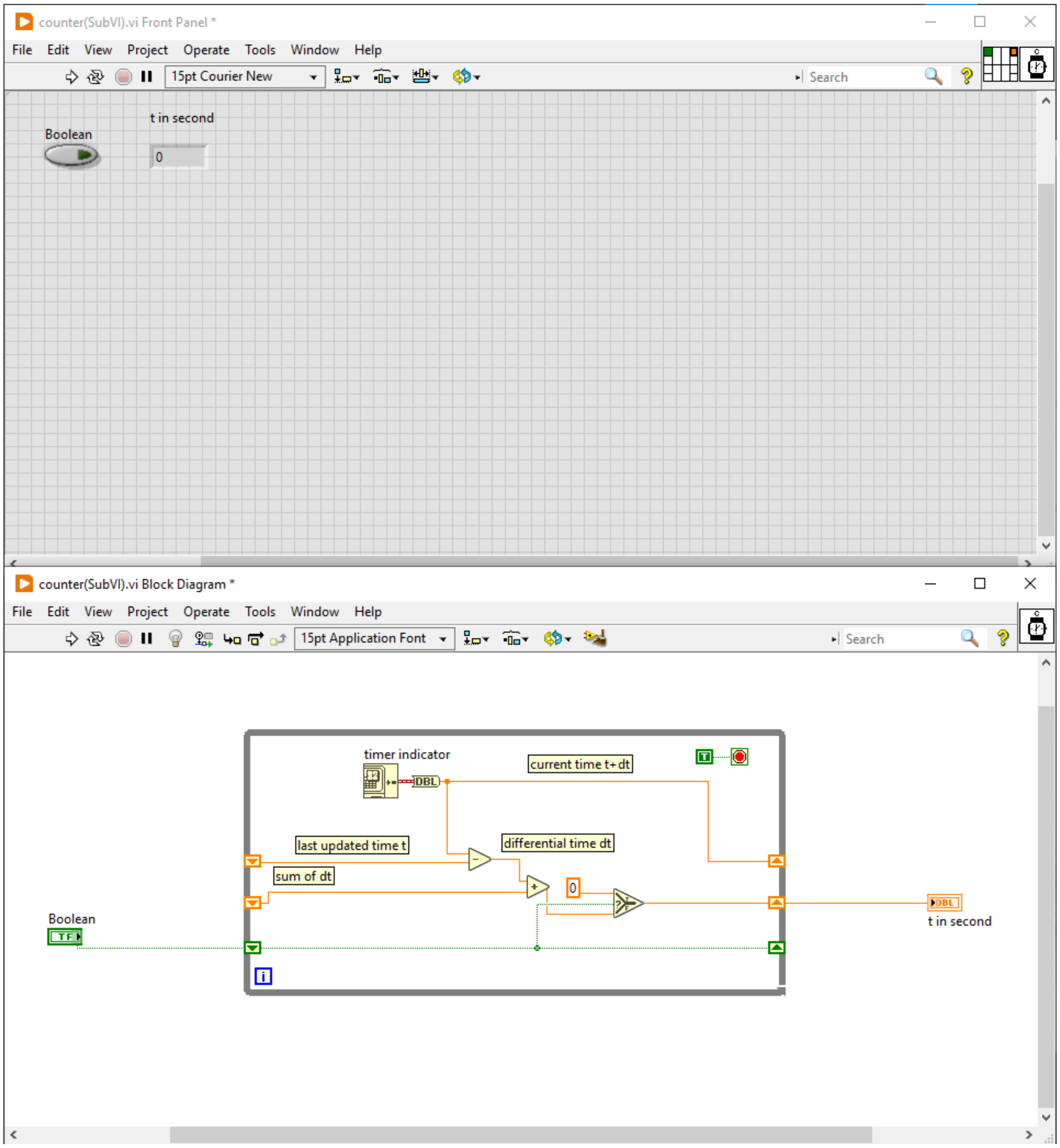


Figure 4-1: Example for a VI in LabVIEW

4.3.2 Structure

Similar to textual programming, dataflow programming also has basic loops. In this thesis, for loop, while loop and case structure (switch-case loop) are mentioned. Figure 4-2 show the options for structure building in Block Diagram.

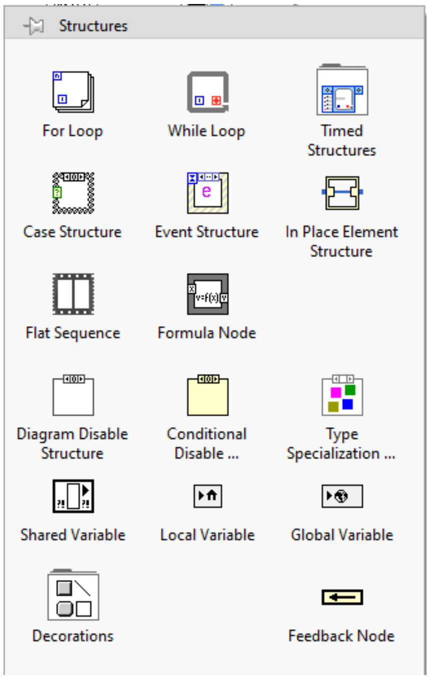


Figure 4-2: Structure palette in Block Diagram

- Mathematical function implementation using For loop

This example shows how the function $w\ k\ = 0.54 - 0.46\cos(2\pi k/N)$ is described in LabVIEW (Figure 4-3). The flagging index (i) represents k. In this example, the for loop is used to generate an N-dimension array output.

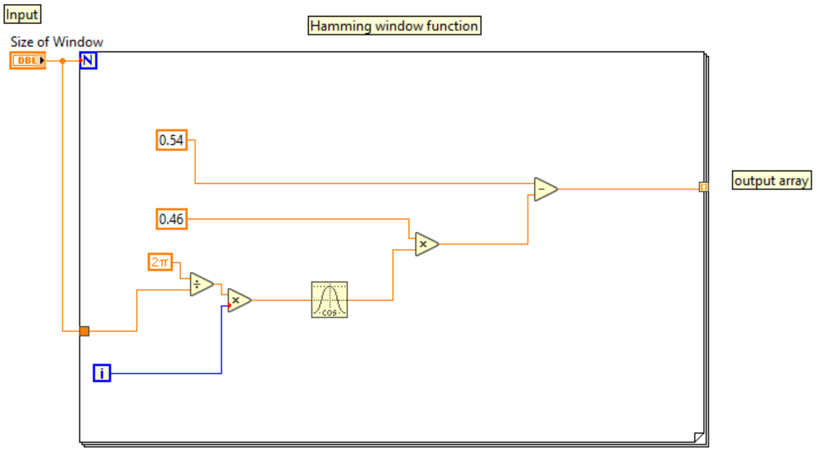


Figure 4-3: Illustration of simple mathematical function in LabVIEW

- Time counter using While loop

In this example, the input is a button (TF). Whenever the button is normal (Boolean value = 0), the time indicator shows the running time with initial value equals 0. When the button is triggered (Boolean value = 1), the time indicator always stay at 0.

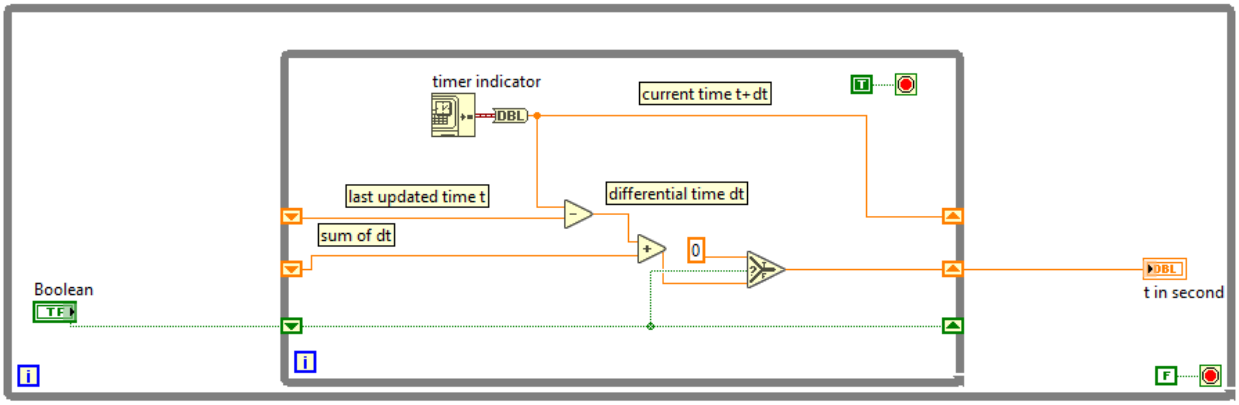


Figure 4-4: Time counter flow diagram

- State-machine using Case Structure and sub VI

State machine is very important in managing any program, while sub VI is useful for simplify the graphic flow diagram. In this small project, a set of lights and buttons represents the two-light traffic light system. The light has three modes: GREEN (G) (allow to go), YELLOW (Y) (slow down), RED (R) (stop), and the button (BUTTON) is setup for the walker to get the priority when crossing the roads.) Figure 4-5 and Figure 4-6 give information about the graphical design of a traffic light system and its state-machine diagram; while Figure 4-7 shows the UI of the program.

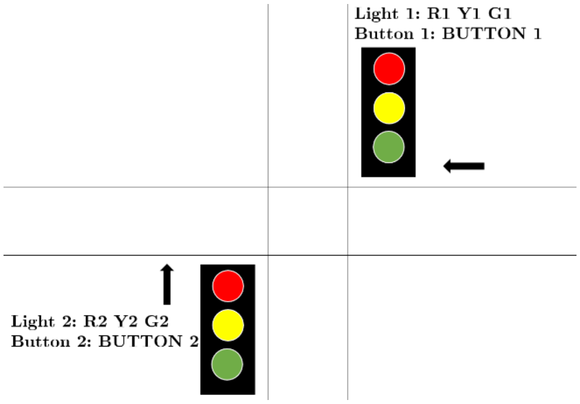


Figure 4-5: Illustration of traffic light system

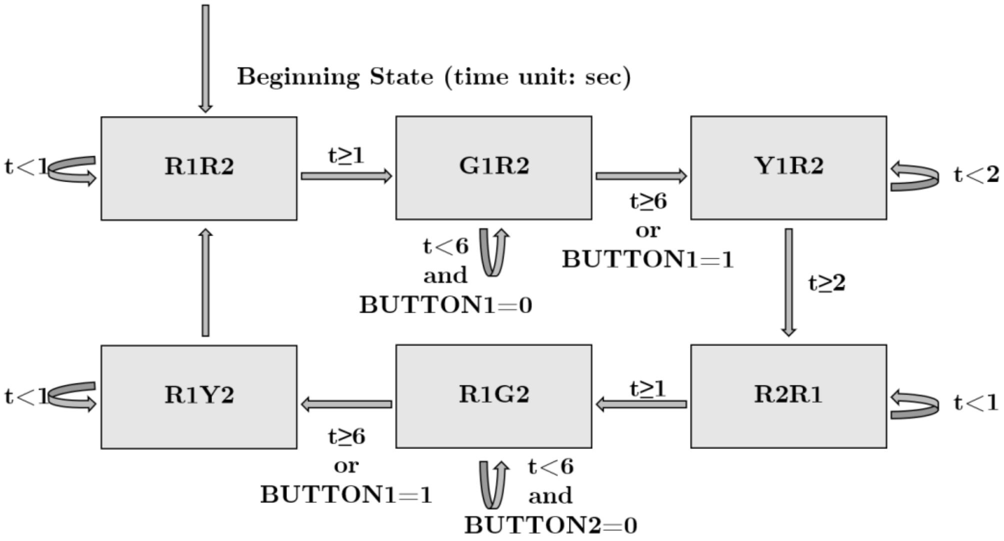
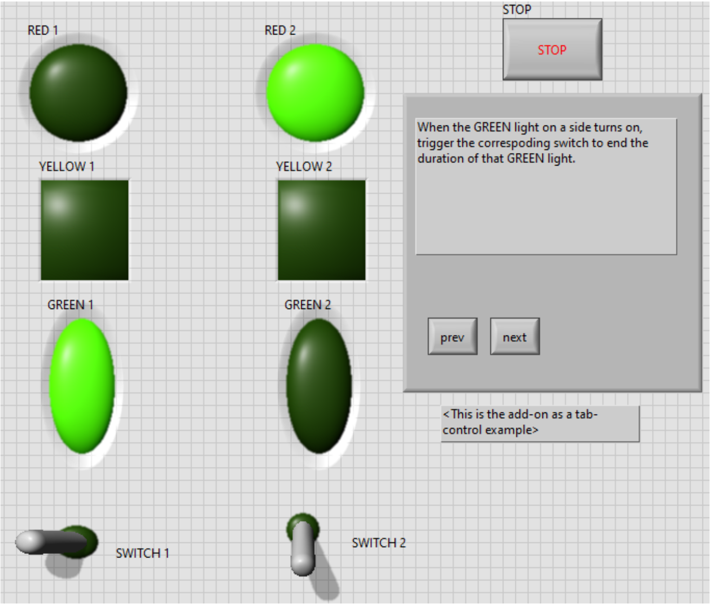


Figure 4-6: State-machine diagram of the system

Because of the necessity of a precise timer, a sub VI was created for timing function. Figure 4-1 shows the detail for the sub VI. Note that the graphical code in Figure 4-1 is different from Figure 4-4 because a while loop must be added to the sub VI to make it operates again and again.

Figure 4-7 and Figure 4-8 show the front panel and the block diagram of the program. In Figure 4-8, the shift register (▼) lets the data (in this case is the enum constant R1R2) run through the loop to the next working frame. In this example, "R1R2" run through the case structure and identify the case. If the time duration is larger than 1 (sec) , "G1R2" run out of the loop and go to the next working frame and the next case is "G1R2", elsewhere "R1R2" would run through the loop and the next case remains the same. With the next states of the program, the method is similar to this.

In addition, the tab control is for illustrating the introduction of the program with many tabs in control of "next" and "prev" button.



When the GREEN light on a side turns on, trigger the correspoding switch to end the duration of that GREEN light.

<This is the add-on as a tab-control example>

Figure 4-7: Front panel of traffic light system project

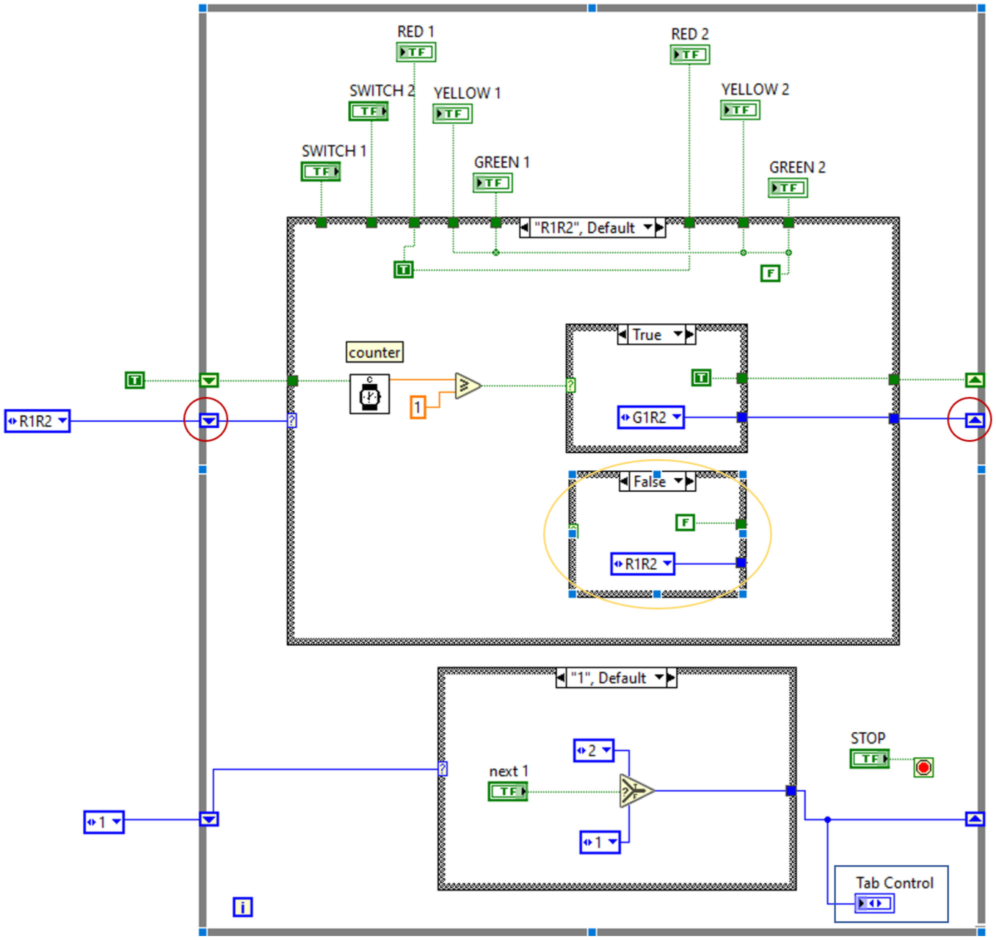


Figure 4-8: Block diagram of traffic light system project

5. Theoretical background

Before read through the theoretical background, note that Appendix 1 explains the notations are used in this report.

5.1 Fourier transform

Fourier transform is a mathematical mapping from a time-domain function $x(t)$ to a frequency domain function $X(\omega)$. In the field of Digital Signal Processing, spectrum analysis (in frequency domain) is a critical concern and therefore the Fourier transform is powerful tool. A continuous-time Fourier transform (CTFT) is defined by:

$$X(\omega) = \int_{t=-\infty}^{t=+\infty} x(t)e^{j\omega t}dt \tag{1}$$

For the discrete-time Fourier transform (DTFT):

$$X(\omega) = \sum_{n=-\infty}^{n=+\infty} x[n]e^{j\omega n} \tag{2}$$

5.1.1 Discrete Fourier Transform

In reality, digital signal analysis is more commonly perform. According to Proakis and Manolakis [1], to perform frequency analysis on a discrete-time signal $x[n]$, there is necessary to compute a conversion from the time-domain sequence into a corresponding frequency representation. However, $X(\omega)$ is a continuous function and therefore, it is not an equivalent mapping of $x[n]$. Intentionally, a frequency domain sampling of $X(\omega)$ with Q points, which called Discrete Fourier Transform (DFT), is:

$$X(q) \equiv X\big(\omega = \tfrac{2\pi q}{Q}\big) = \textstyle\sum_{q=0}^{Q-1} x[n]e^{-j2\pi q/Q} \;\; ; \text{ where } q = 0, 1, 2 \ldots Q-1 \tag{3}$$

5.1.2 Fast Fourier Transform

Because the direct DFT is operated inefficiently, the workload of a computation would be decreased significantly with the help of fast Fourier transform [1]. Radix-2 FFT algorithm ($Q = 2^p$, with p is a natural number), which are the most common used FFT algorithm. Therefore, for an efficient manipulation, the size of the transformation Q is chose to be a power of two.

5.1.3 Power Spectrum with FFT

For visualization, compute and plot a power spectrum is essential in Signal Processing. To have an overview about the construction of a power spectrum of any waveform, let us start with a pseudocode example in MATLAB and its equivalent code.

```
% real time signal with length N (signal)
%FFT with length Q
Freq=(-Q/2:Q/2-1)/Q; % Frequency bins array
X=(1/length(signal))*fft(signal,Q); % compute FFT, length(signal)=N
X1=fftshift(X); %
PS_dB=20*log10(abs(X1/max(X1))) % power spectrum with dB scale
%Plot the Power Spectrum%
```

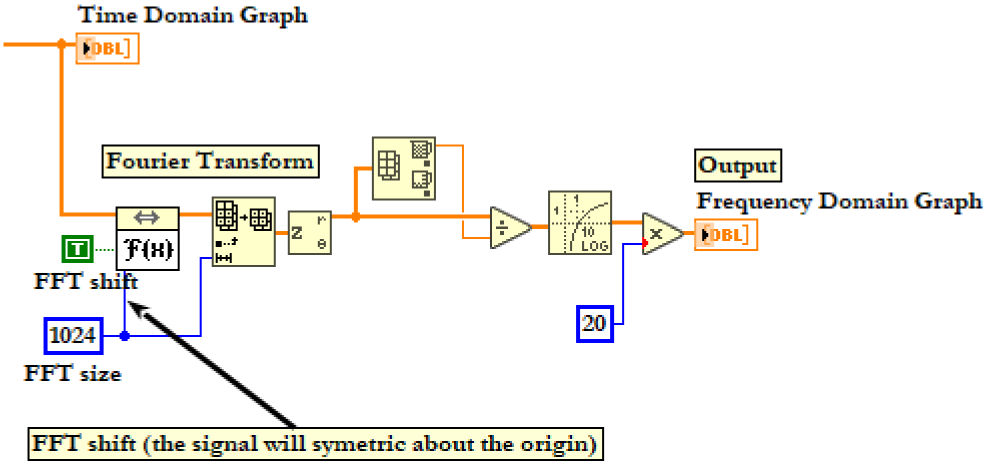


Figure 5-1: concept for FFT

As aforementioned, FFT is a mapping from an N-sample signal $x[n]$ to an Q-sample frequency response $X(q)$. The demonstration of the result will be in the next chapter. Please note that Power spectrum $= 20\log_{10}|\bullet|$ but not $10\log_{10}|\bullet|$ because Power $\sim x^2[n]$; leads to $Power = 10\log_{10} \bullet\ ^2 = 20\log_{10}|\bullet|$.

5.2 Spectral leakage phenomena

Although DFT brings analytic method, it is susceptible with spectral leakage and therefore an inaccurate estimation of the result may occur [2, 3, 4]. According to the documentation, the spectral leakage phenomena occurs when the acquiring interval is finite, and that observation duration contains a non-integer number of signal period.

Explain to this, the observed signal may not be adequate with its original period because of discontinuities at the boundaries of the captured signal (Figure 5-2).

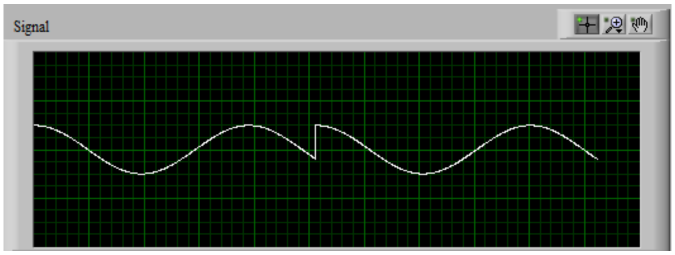


Figure 5-2: Discontinuity between a recirculated signal with its original signal

Therefore, to prevent the discontinuities, it necessitates to satisfy some criteria:

_ The end point of the signal is the same as its beginning and begin to zero, otherwise the recirculated signal is unmatched with the original.

_ Furthermore, the slope of the initial and final state of the acquiring signal must equal to zero as well. In general, aperiodic and random signals violate these regulations.

5.3 Window functions

Window functions are weighting functions applied to the signal before using the DFT to mitigate the spectral contribution [2, 3]. The window function (with the shape of a ring bell) is used to taper the borders of the observing signal to be the same at both edges (equal zero, otherwise it cannot be sure that two edges have the same value) of the window (to satisfy the first criteria as aforementioned) and match their derivatives (as much as possible) to zero or approximately zero (to meet the second criteria). Thus, windowed data have smoothly edges and the DFT of this windowed data can diminish the spectral leakage.

Research of a number of window functions were conducted in [2] and further investigation in [4]. The general form of a weight cosine window is as follow [4]:

$$w[k] = \begin{cases} \sum_{m=0}^{M} -1\ ^{m} a_m \, cos\left(\frac{2\pi}{K} mk\right); & k = 0; 1; \ldots K-1 \\ 0; & \text{otherwise} \end{cases} \tag{4}$$

with the following constraint:

$$\sum_{m=0}^{M} a_m = 1 \tag{5}$$

Notice that K is the window's size. Usually, this size is equal to the size of samples in the captured duration, i.e. $K = N$. For the convenience and simplification, the rest of the thesis will use N to note for both quantities.

We should evaluate if the upper function (4) is satisfied the criteria in chapter 5.2, but in the analog form:

$$w(t) = \begin{cases} \sum_{m=0}^{M} -1\ ^{m} a_m \, cos\left(\frac{2\pi mt}{T}\right); & t \leq T \\ 0; & \text{otherwise} \end{cases} \tag{6}$$

At two boundaries:

$$w(0) = w(T) = \sum_{m=0}^{M} (-1)^m a_m \tag{7}$$

$$\frac{dw}{dt}(0) = \frac{dw}{dt}(0) = 0; \tag{8}$$

To conclude, the equations (8) meets the requirements in chapter 5.2, but equation (7) generally do not. In the rest of the chapter, some evaluating parameters are presented to analyze the efficiency of some famous window functions.

According to [2] and further explanation in [5, 6, 7]; if any window function has a nonzero n-th derivative ($n = 0$ represent its own function), its sidelobe falloff slope equals to $6(n + 1)$ decibel per octave. The clarification of the optimization according to this parameter would be in the following.

5.3.1 Hanning window

Hanning window function is given by the following formular:

$$w[k] = A - (1 - A)\, cos\left(\frac{2\pi k}{N}\right) \tag{9}$$

For Hanning window, $A = 0.5$.

This function exist a nonzero derivative at the power of 3, means that its sidelobe oblique asymptote is 18 (dB/oct). In general, a window function has a higher sidelobe falloff rate will emphasize the drop of the sidelobes with respect to the main lobe and minimize the effect of spectral leakage phenomena.

Figure 5-3 shows the time-domain function and its frequency response of Hanning window with the length of $N = 51$.

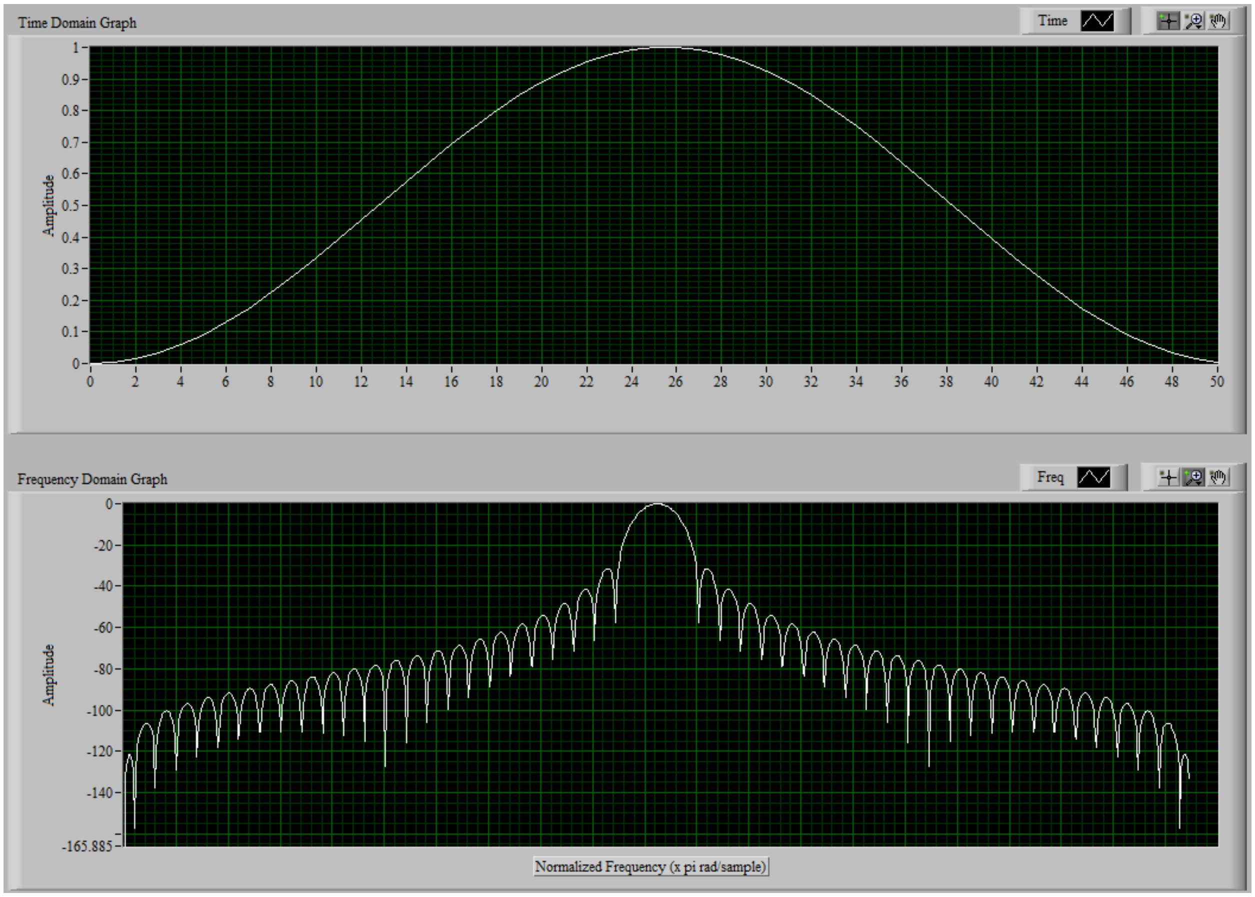


Figure 5-3: Hanning window and its frequency response

5.3.2 Hamming window

The Hamming window is an optimization of the Hanning window. According to [2], by analyze the behavior of sidelobe attenuation (the difference between the mainlobe and the first sidelobe), $A \approx 0.54$ results the largest sidelobe attenuation:

$$w[k] = 0.54 - 0.46\, cos\left(\frac{2\pi k}{N}\right) \tag{10}$$

Figure 5-4 shows the time-domain function and its frequency response of Hamming window with the length of $N = 51$.

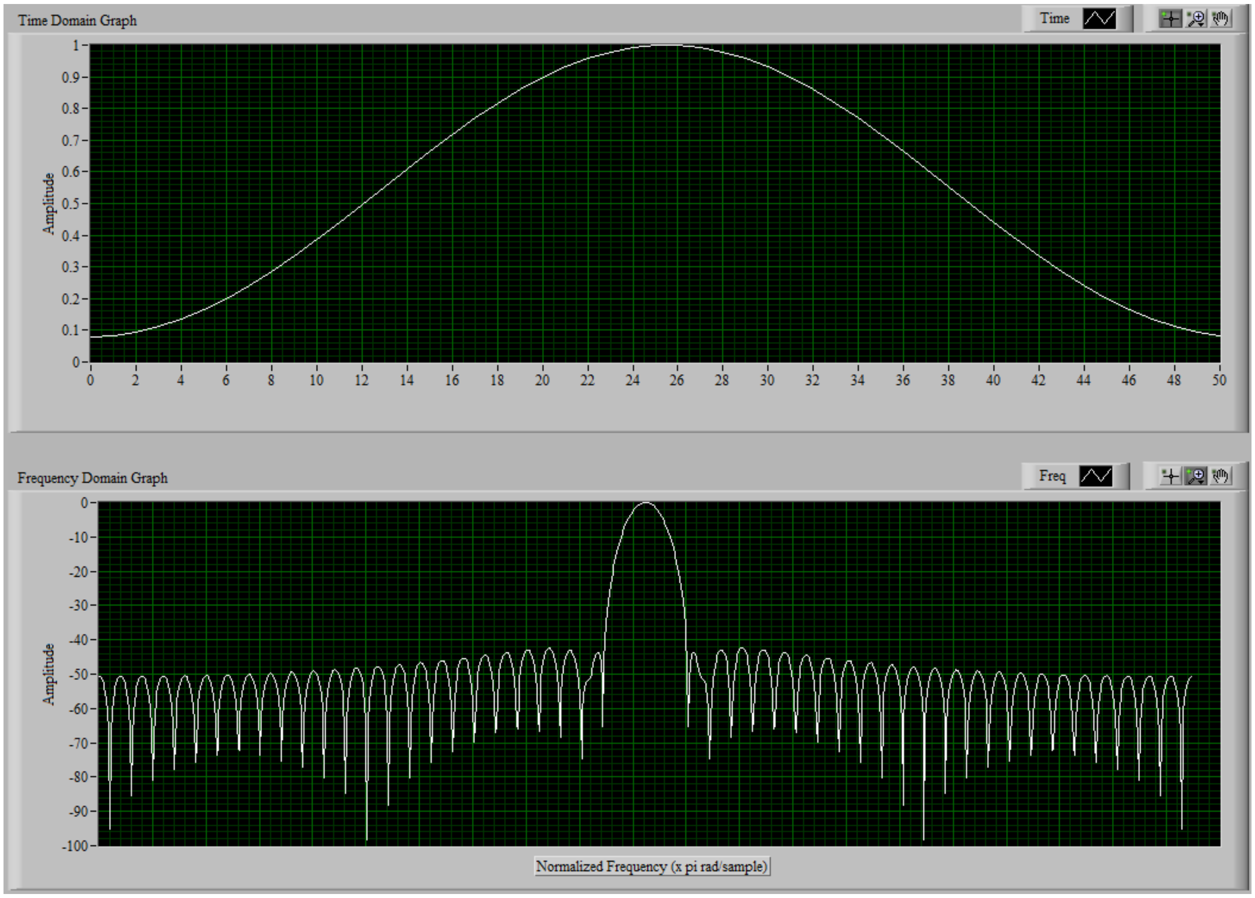


Figure 5-4: Hamming window in time and frequency domain

Using MATLAB to re-proof the work of Harris [2] is a good way to illustrate how the parameter weight A was chose. The idea is to test A in its domain with the increment of 0.0001 using for loop and compare the maximum height of the sidelobe in dB scale (sidelobe attenuation) to figure out which A results the minimum sidelobe level. The MATLAB code for this is in Appendices chapter,

Appendix 2.

The result for this is:

The largest sidelobe attenuation:

```
x =

 -42.805090097366673
```

The corresponding weight A:

```
A_xmax =

0.538400000000000
```

Compare the result with the original work, $A = 0.53856$, it is around 1% error. It means that the approach's algorithm of the re-proof is correct.

However, it is noticeable that the boundary's values are not equal to zero:

$$w[k=0] \equiv w[k=N-1] = 0.54 - 0.46 = 0.08 \tag{11}$$

This entails the sidelobe falloff rate equals to 6 (dB/oct). To be noticed that the absence of window function (or can be known as rectangular window) results the same decrease level. However, the Hamming window forces the sidelobe level to be dramatical lower (-43(dB)) than without a window (-13 (dB)). This brings the meaning of visualization in the sub-chapter 5.3.3 below.

5.3.3 Overview of windows

This part illustrates the advantages of Hamming window and Hanning window comparing to the original power spectrum.

Figure 5-5, Figure 5-6 and Figure 5-7 show the power spectrum of an arbitrary cosine wave with and without applying windows. While the original one has the small mainlobe level and the power has leaked into other frequency bins; the windowed spectrums highlight the benefits: the mainlobe level raises considerably and the separation between the main frequency bin and the neighbors is existed.

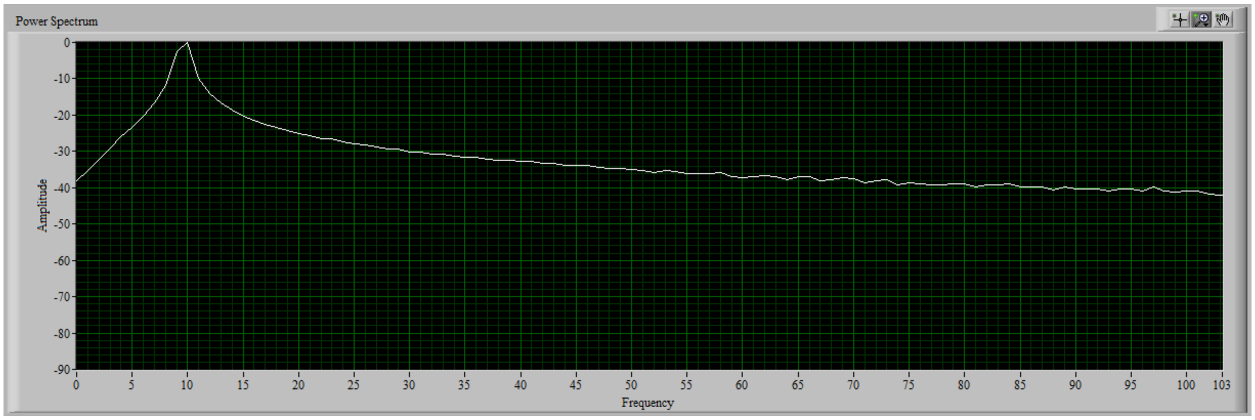


Figure 5-5: Original power spectrum of a signal

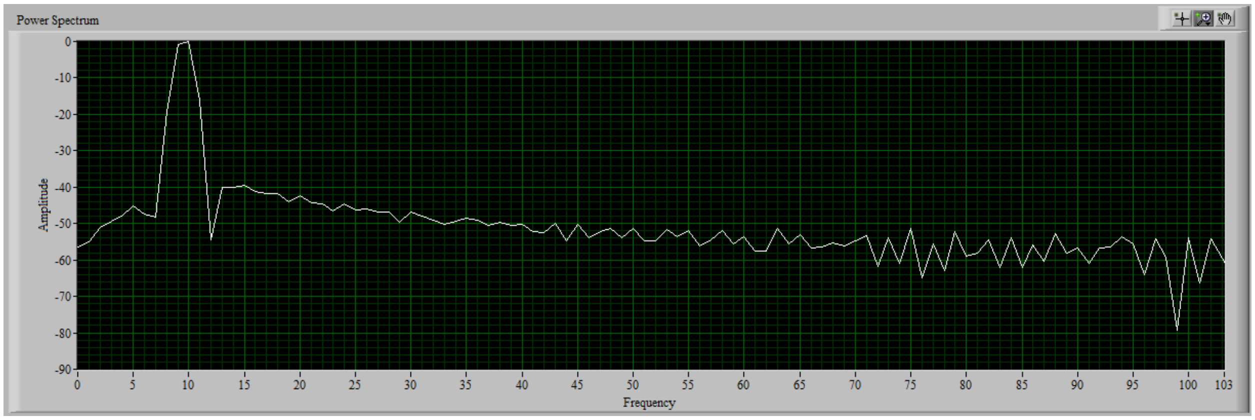


Figure 5-6: Power spectrum of the same signal after applying Hamming window

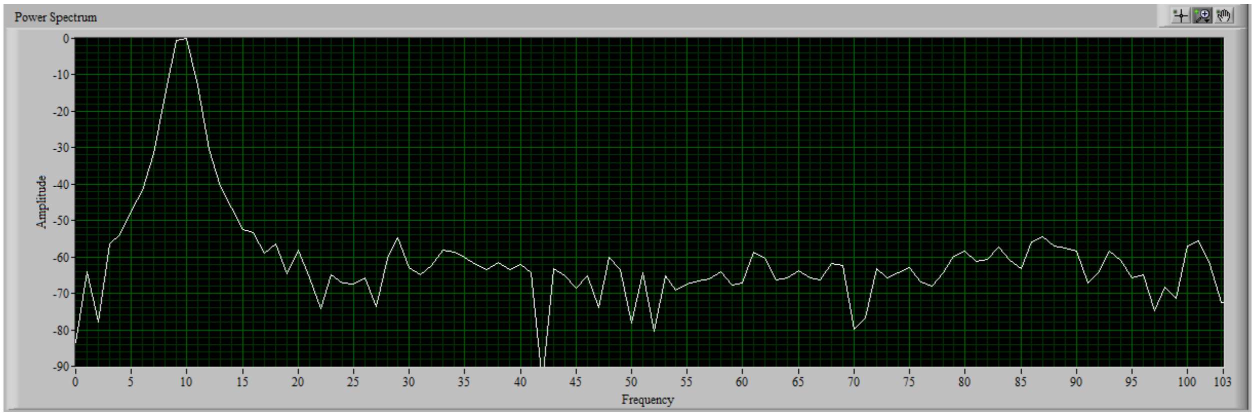


Figure 5-7: Power spectrum of the same signal after applying Hanning window

5.4 Window application in spectrum detection

In the field of DSP, window functions emphasize the spectrum by separating the main frequency bin with the neighbor and increase the fall-off rate. In application, for example, a duo-tone signal with two nearby frequencies is analyzed as in Figure 5-8 and Figure 5-9. In both frequency and time domain, it is complicated to figure out if the signal is duotone and calculate the second component of the signal. However, when applying Hamming window function as in

Figure 5-10, the main spectrum of both components are isolated from the neighbor and easy to determine second element of the signal.

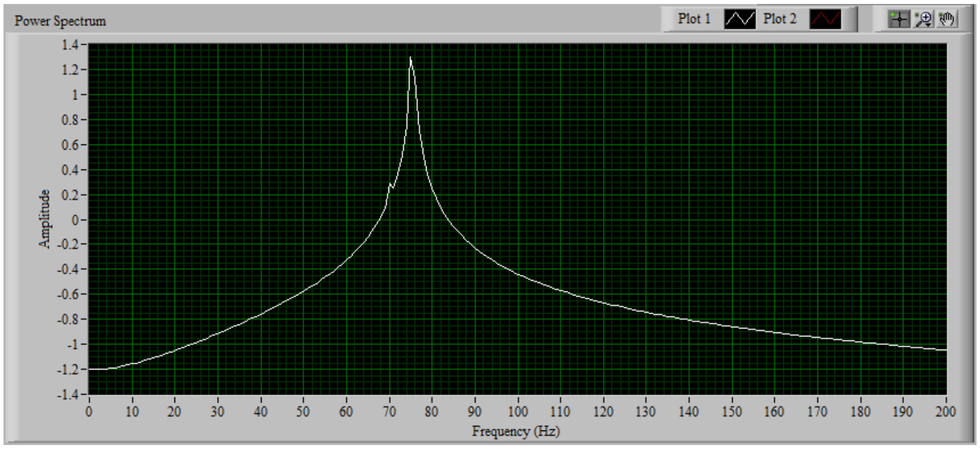


Figure 5-8: Original signal in frequency domain

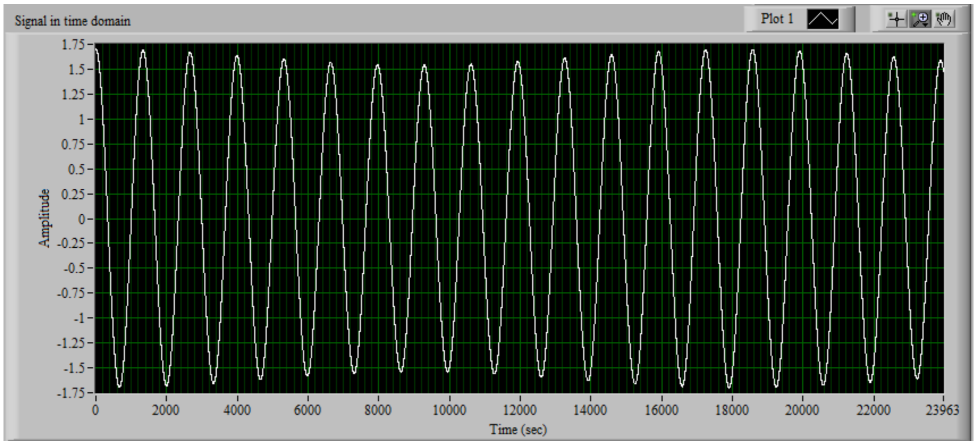


Figure 5-9: Original signal in time domain

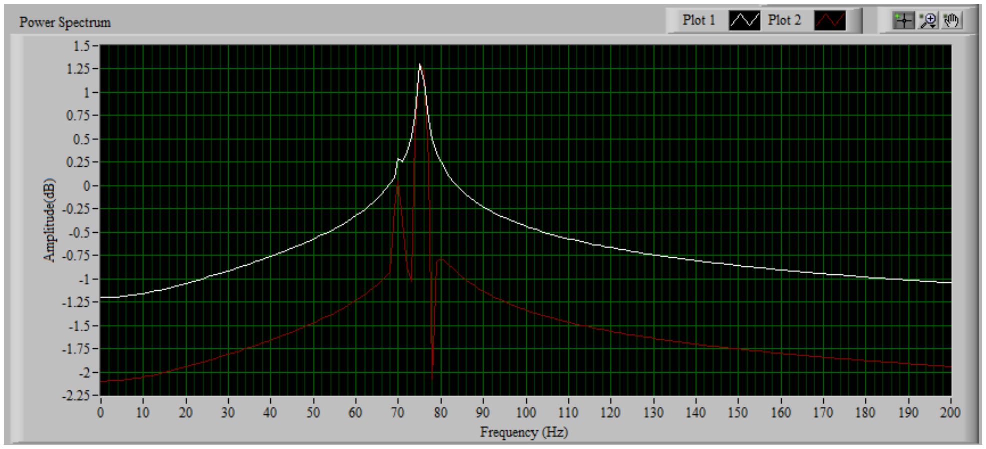


Figure 5-10: Comparison between original and windowed signal

5.5 Median filter

In general, median filter is a non-linear technique to eliminate the noise from a signal. The key algorithm of the median filter is to replace every element of the signal with the median of the neighboring elements which are in the rank. To be noticed that median of a set is the number that separates the larger and lower parts of that set.

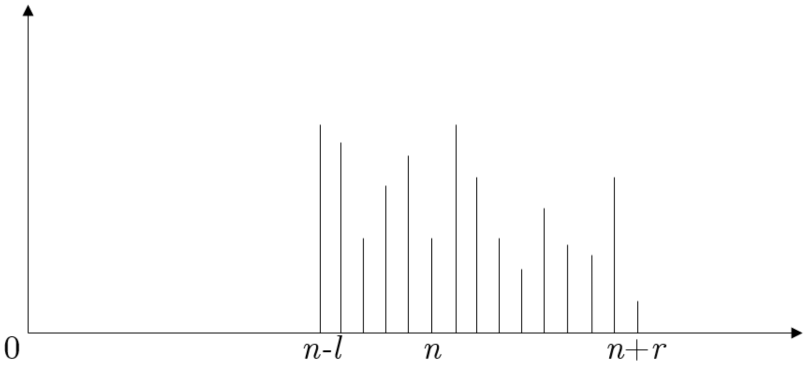


Figure 5-11: Time domain of random signal

With the input signal $x[n]$, the filtered output $y[n] = \text{median}\ \{x[n-l]; \ldots x[n+r]\}$, with l and r are the left and right filter rank. Because a set of values have a median value when its size is odd, so that $l + r + 1$ must be an odd number. To simplify the constraint, lets $l = r$.

Figure 5-12 shows an example of implement the median filter in signal smoothing. In the figure, the "green" noisy signal is a combination of a rectangular signal, Gaussian white noise with a spike noise. After filtering, the median filter can preserve the shape of the rectangular, also reduce the noise (rather than a fluctuated signal) and diminish the spike.

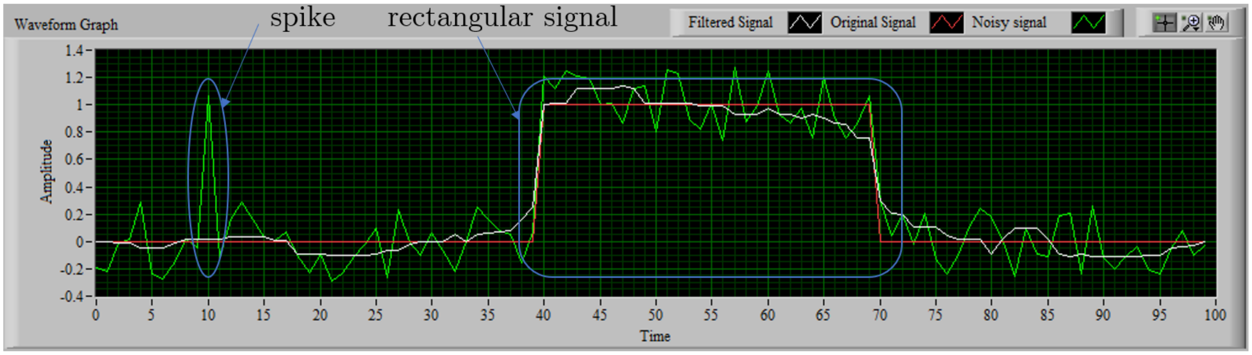


Figure 5-12: Using median filter for data smoothing

5.6 FIR and IIR filters

In this part, method to compute the coefficients of the FIR and IIR filters are not mentioned but the properties of them are shown and compared.

According to [1], for a linear-time invariant system, the general output-input equation:

$$y[n] = \sum_{\kappa=0}^{\kappa=B} b_\kappa x[n-\kappa] - \sum_{\kappa=0}^{\kappa=A} a_\kappa y[n-\kappa] \tag{12}$$

With b_κ and a_κ are **forward coefficients** and **reverse coefficients**.

The transfer function of the equation (12) is as below:

$$H\ \ z^{-1}\ \ = \frac{\sum_{\kappa=0}^{\kappa=B} b_\kappa z^{-\kappa}}{1 + \sum_{\kappa=1}^{\kappa=A} a_\kappa z^{-\kappa}} \tag{13}$$

5.6.1 FIR filters

- Stability

In case of FIR filters, all reverse coefficients a_κ equal to 0, the output-input equation becomes:

$$y[n] = \sum_{\kappa=0}^{\kappa=B} b_\kappa x[n-\kappa] \tag{14}$$

The impulse response of the FIR filter can be obtained when substitute $x[n-\kappa]$ by impulse function $\delta[n-\kappa]$:

$$h[n] = \sum_{\kappa=0}^{\kappa=B} b_\kappa \delta[n-\kappa] = \begin{cases} b_n & 0 \le n \le B \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

This means the impulse response of any FIR filter is finite and it is always stable.

- Linear phase response

Although FIR filter's performance is generally less than IIR filter's; it has a highlight characteristic: the phase response of the desired zone is linear (i.e. the low frequency band of the lowpass filters). As comparison to the IIR filters, their frequency response are non-linear (there exist methods to linearize the phase response of IIR filters, but it is complicate in comparison to the FIR filter).

5.6.2 IIR filters

- Fast execution and less memory storage

Because the IIR filters can generate the same attenuation slope (and cutoff frequency) with a lower order in comparison to the corresponding number of taps of FIR filters, they require less memory for storage. Moreover, because of the fast execution, IIR filters are appropriate in real-time control that needs high rate of manipulation.

5.6.3 Comparison between both type of filters

In this part, some small program (which are not included in the main program) is built for re-validation the properties of both IIR and FIR filters.

The following figure depicts the phase delay of FIR and IIR lowpass filters (cutoff frequency is $100(Hz)$), prove that the FIR filter's phase characteristic is linear, while for IIR is not.

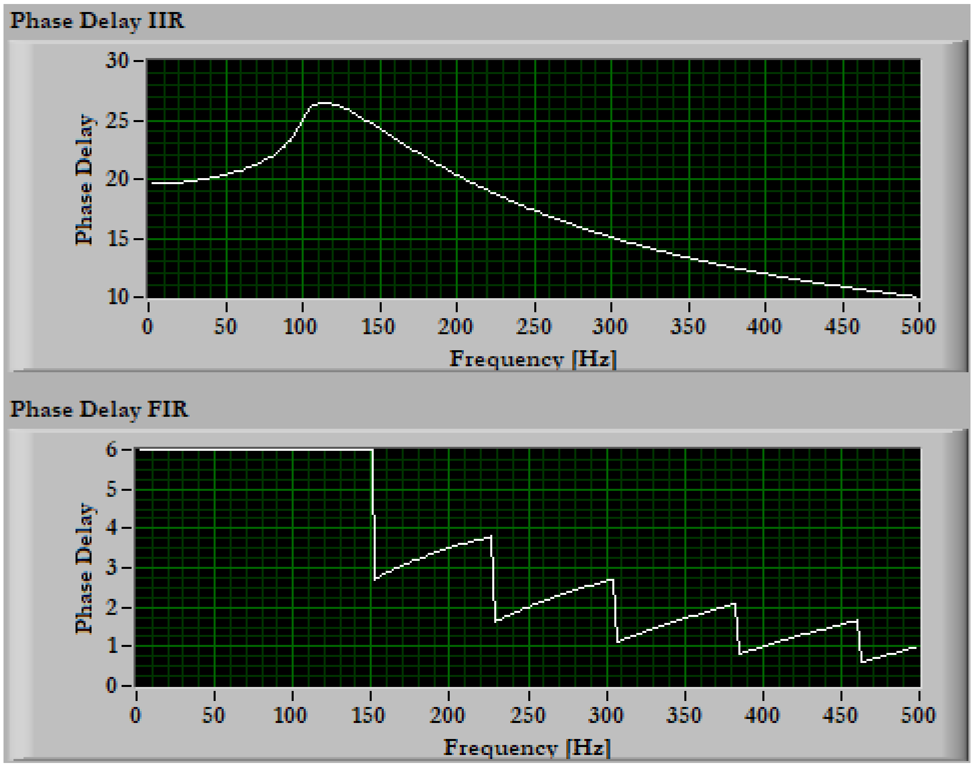


Figure 5-13: Phase delay of filters

The following figures illustrate the differences between FIR and IIR lowpass filters. In Figure 5-14, the FIR filter's tap and the IIR filter's order equal 10. In Figure 5-15, the IIR filter's order remains the same but the FIR filter's tap increase to 30 to have a similar attenuation rate.

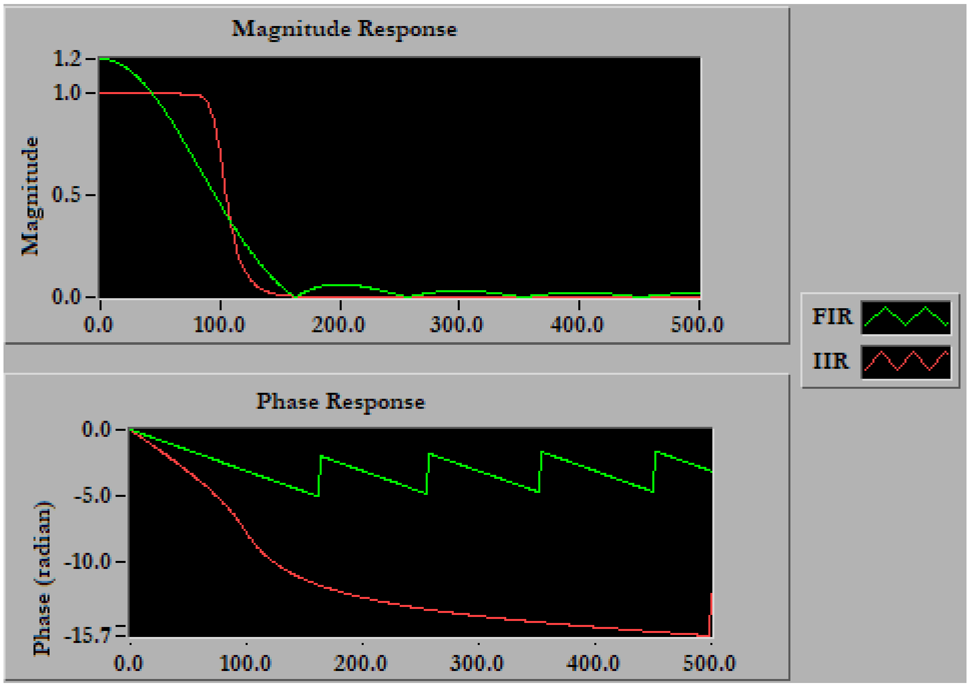


Figure 5-14: Comparison between FIR (10 taps) and IIR (10-th order) filter

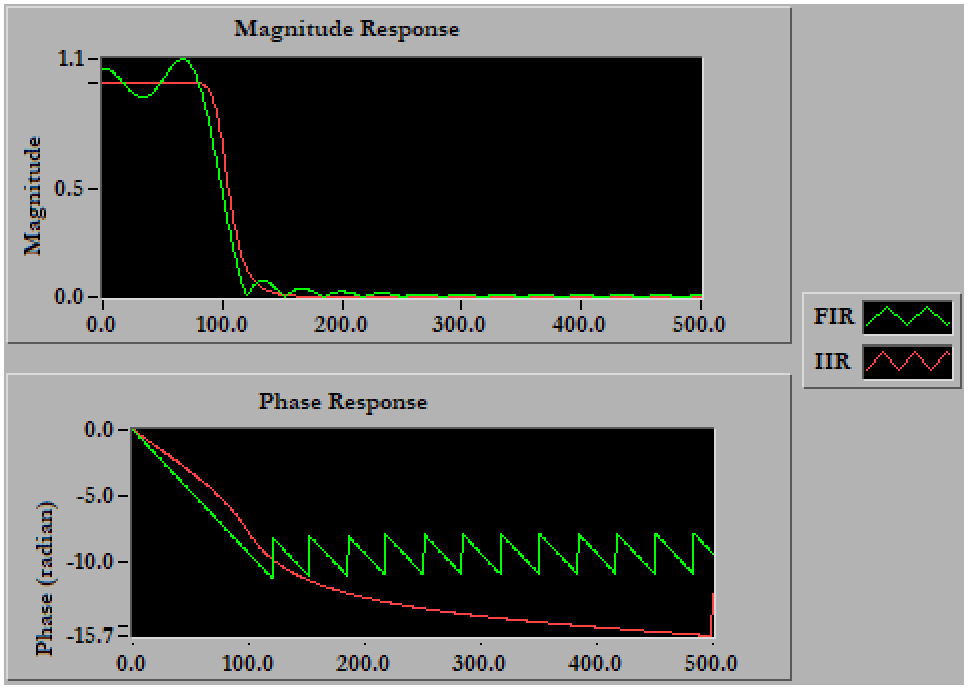


Figure 5-15: Comparison between FIR (30 taps) and IIR (10-th order) filter

The following table summarize the advantages and drawbacks of FIR and IIR filters.

FIR filter	IIR filter
Do not contain poles: always stable • Easy constructing method	Not always stable (the poles need to be inside the unit circle) • More constraints
Linear phase characteristic	Non-linear phase characteristic (necessitate the usage of all-pass phase equalizers)
need more coefficient for the same magnitude characteristics with respect to IIR filters • More memory storage • Slower executing rate	but require relatively less coefficients for the same operation requirements with respect to FIR filters • Less memory storage • Faster computation

Table 2: Comparison between FIR and IIR filters

6. LabVIEW program illustration

In this chapter, a brief description of the figure (represents the graphical code for each module) is explained in the flowing direction of the data.

6.1 Spectral leakage and spectrum analysis

1. Set the input of the sine wave signal noise and add them together.

2. Let the noisy signal run through the window (with the input “Window Option” for choosing the type of listed window function)

3. Take the FFT of the windowed signal

4. Display the Spectrum.

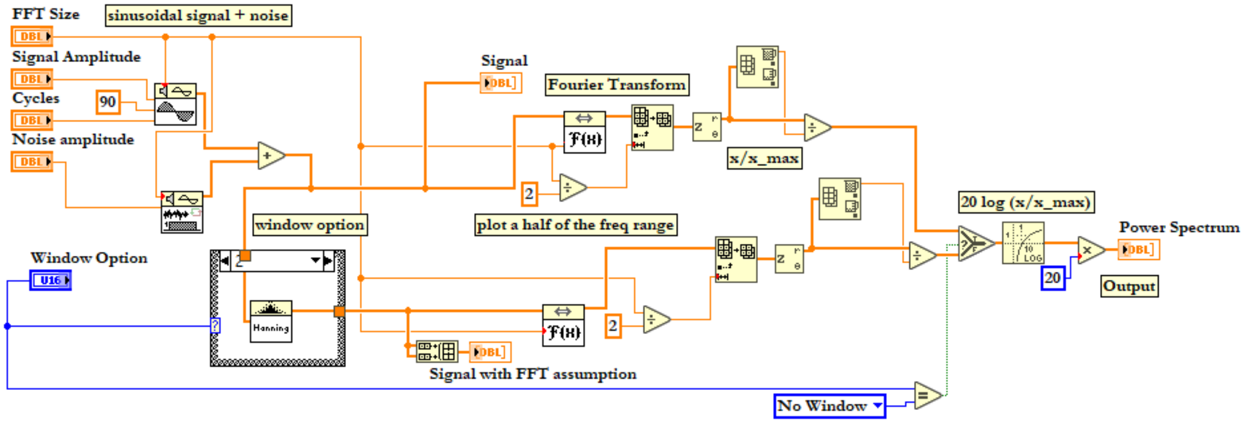


Figure 6-1: Spectral leakage and spectrum analysis tab

6.2 Smoothing window comparison

The signal is included of two sine wave signals, and the structure of the module is similar to the former module.

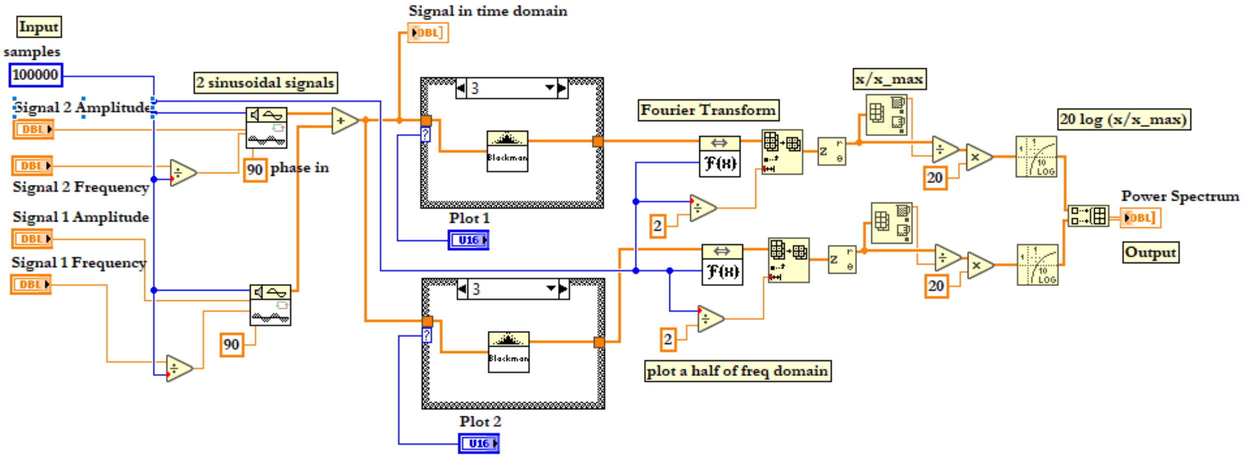


Figure 6-2: Smoothing window comparison tab

6.3 Window function generator

1. Set the size of the window function and choose the type of window function.

2. Generate the window function with mathematical elements and display it.

3. Take the FFT of the window function and display it.

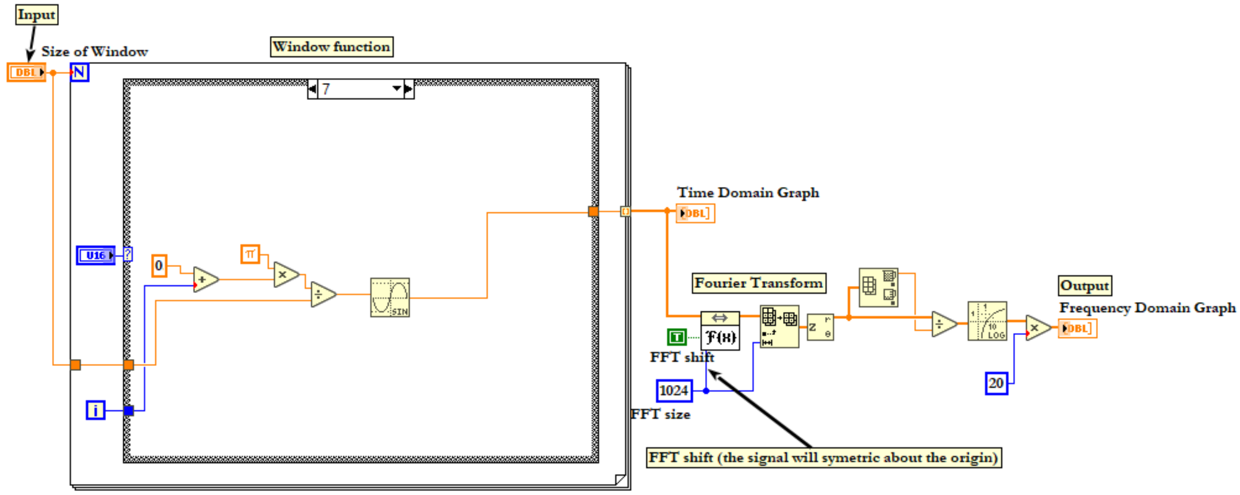


Figure 6-3: Window function generator tab

6.4 Median filter

1. Build up a mixture of signal consist of: a rectangular wave, a “white” uniform noise with a spike.

2. Let the noisy signal run through the median filter with the left and right rank equal to each other.

3. The user can choose the existing plot via the “Plotting option” loop and display them.

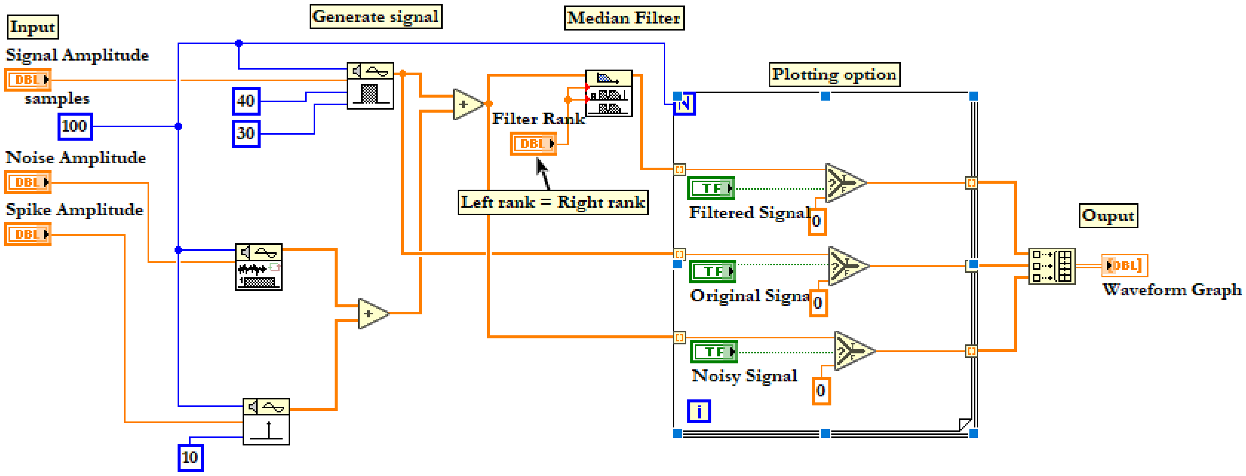


Figure 6-4: Flow diagram of median filter tab

6.5 FIR and IIR Filter design

1. Construct the FIR and IIR coefficients with the corresponding control input.

2. Take the Frequency Response of the filters (including Magnitude and Phase Response) and display them in the same graph. (Have options for normal scale and *dB* scale)

3. Plot the zeros and poles of the LTI systems in the Z-Plane.

4. Take the Phase delay of the systems and display them.

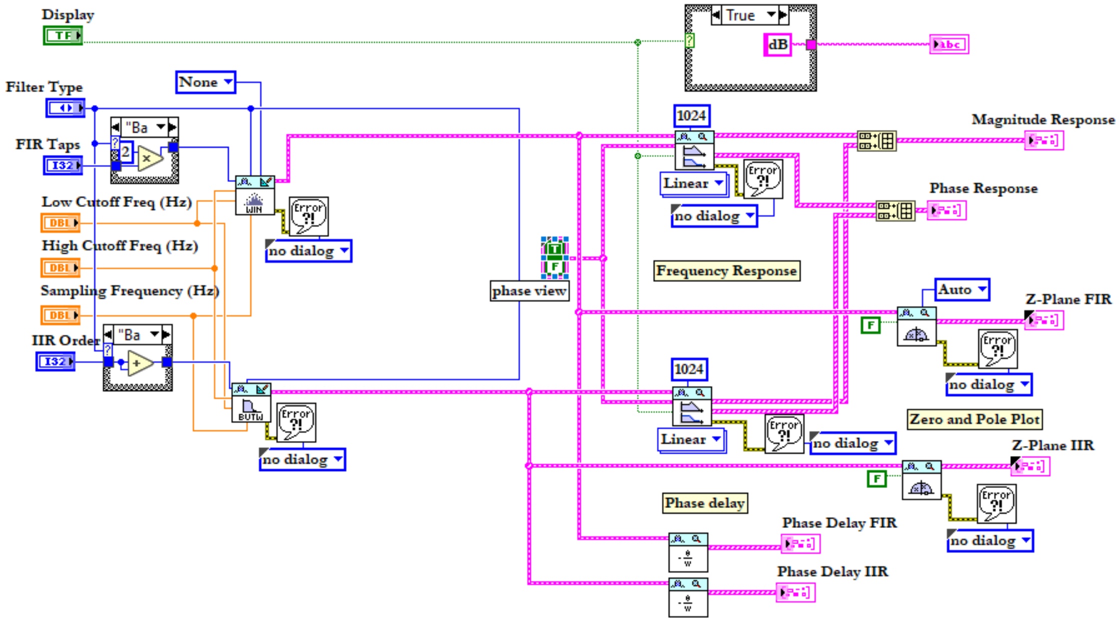


Figure 6-5: FIR and IIR filter design tab

7. Problem encounter and future plans

In this chapter, numerous problems are listed and the solution that the author found to deal with those difficulties. After that is the realistic plan for any deeper improvement of this program.

7.1 Phase 1: 7 June to 22 June

In this phase, the problems came from Window capability and the unfamiliarity with the new environment. At first, the author try to use the virtual machine with Window XP 64 bit. However, the ADVISOR program worked well in Window XP 32 bit, but it did not run in Window 64 bit.

There are some features that did not exist in the textual programming but did in the graphical programming, for instance, enum constant and shift register are the typical ones of LabVIEW.

7.2 Phase 2: 23 June to 13 July

The problems mostly come from the theoretical proof. Because the thesis's approaches are theoretical and practical parallelly, it necessitate a harmonious structure between equations and their illustration in LabVIEW or MATLAB.

7.3 Phase 3: 14 July to 16 August

At first, in the tab of FIR and IIR filters, there was a method to plot the magnitude and phase response of the FIR and IIR filter using an impulse signal as an input of an arbitrary (FIR or IIR) filter, and display them as in the draft in Figure 7-1.

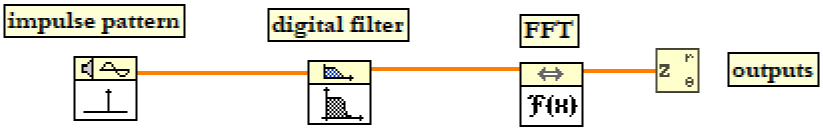


Figure 7-1: symbolic graph for digital filter analysis

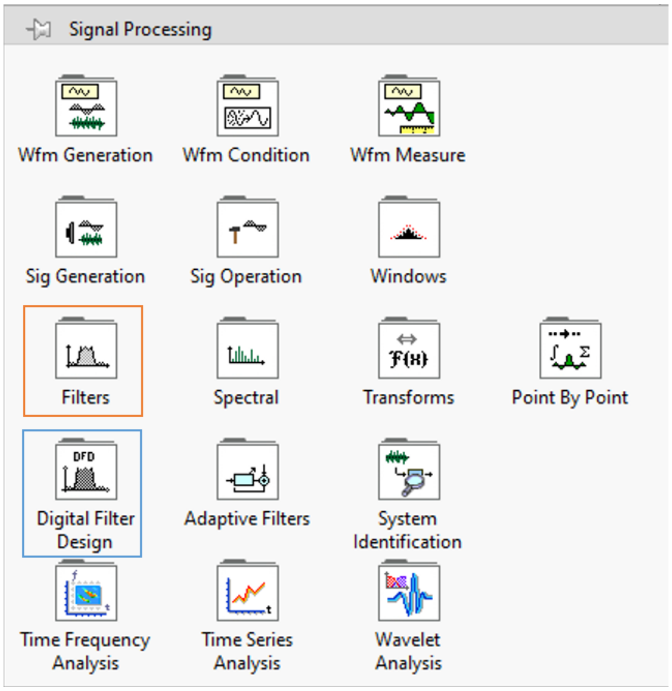


Figure 7-2: Signal Processing Palette

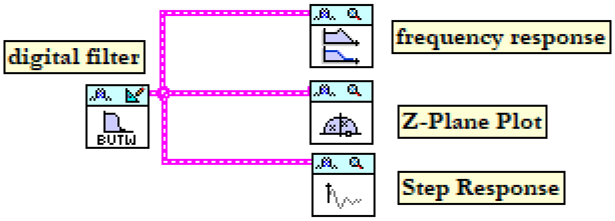


Figure 7-3: symbolic graph for digital filter design

However, because the item are not provide the option to show off the coefficients of the filter, therefore cannot make the pole-and-zero graph of the corresponding filter, as a constraint of the thesis.

In advance, there another built-up items for investigating the digital filter in a more detail approach, as in Figure 7-2. The selection for “Digital Filter Design” provides libraries for specific analysis as illustrated in Figure 7-3, which are not able to generate in the selection for “Filters” in Figure 7-1.

7.4 Phase 4: 17 August to 30 August

After the realistic demonstration in Professor Jungke's personal laptop, there existed a UI problem: his laptop had a display resolution of 2000 × 3000 (pixels × pixels) and a scale of 200%, while the full scale of the program window is 1080 × 1920 (pixels × pixels), and the program cannot display fully in his screen, also the program is blurring. This leads to the change in the scale of the application window: spacing optimization in a compact way with multiple tab controller, as the following Figure 7-4.

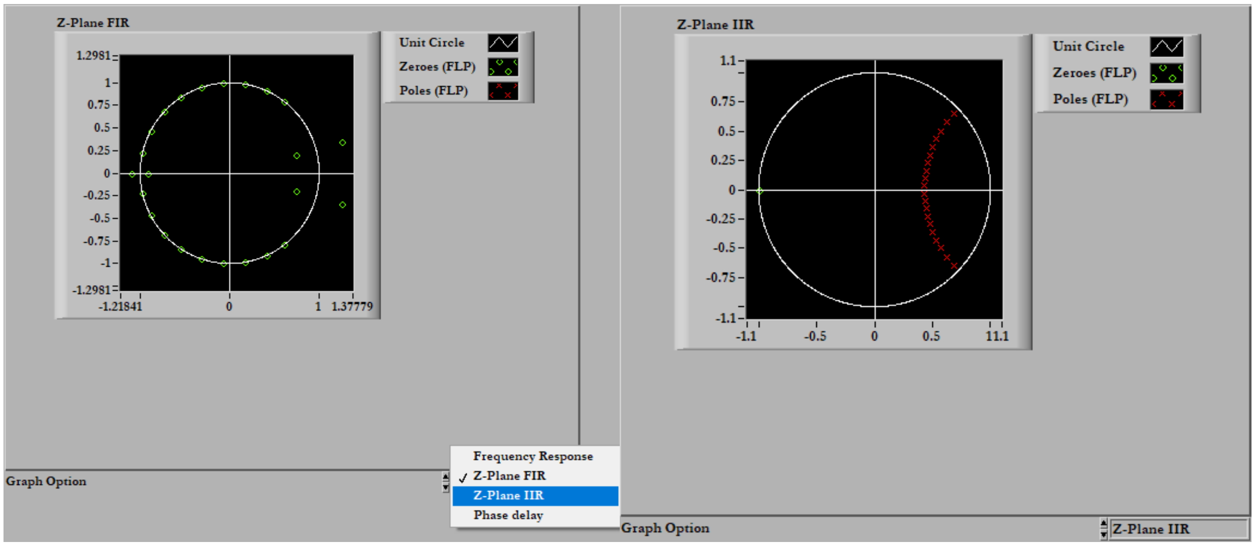
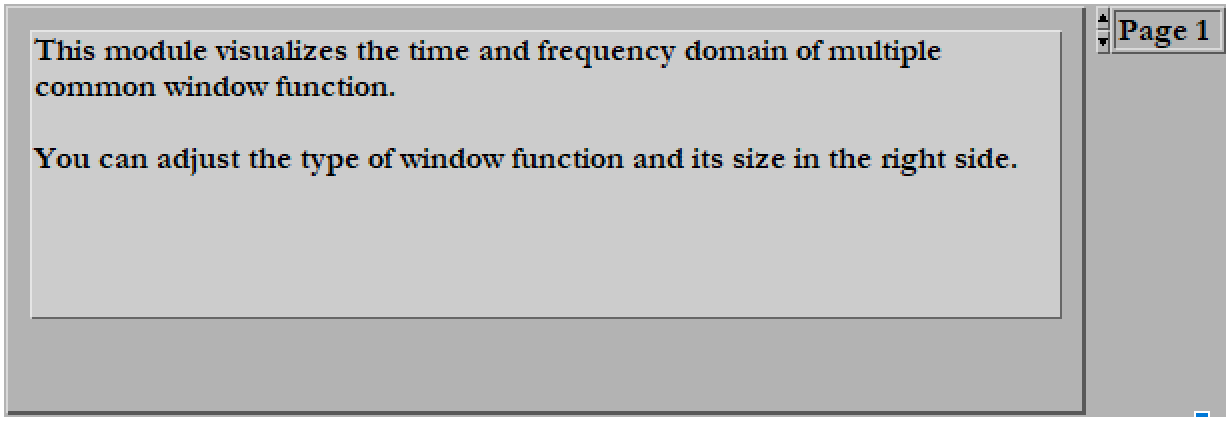


Figure 7-4: Use tab controller to utilize the window size of the program

As can be seen from Figure 7-4, the tab controller can categorize many visual display or input controllers together. Also, tab controller also was used to build the main palette of the program and for the manual instruction, as in Figure 7-5.


This module visualizes the time and frequency domain of multiple common window function.

You can adjust the type of window function and its size in the right side.

Figure 7-5: Use tab controller for user's instruction

7.5 Further insight for the program

In the field of filtering, while the FIR and IIR filters are analyzed clearly, the space for median filter and their application in reality is not mentioned in this thesis and this is a possibility for any further expansion of this project in the future. Moreover, a more user-friendly UI needs to be consider because the program's scale are currently cannot be change. Last but not least, because the programming approach is for teaching support as many case studies with unchangeable setup, that's why it is not interactive with the students as they cannot change or add more inputs and constraints to what they are unclear about the field of studying but they can only choose what are built up in the program.

8. Conclusion

To conclude, the topic of processing signal in frequency domain is an indispensable topic in Digital Signal Processing, as to work in frequency domain is the only choice to analysis the characteristic of the signal rather than in the time (space) domain. In this thesis, two main branches on windowing and digital filtering are discussed with theoretical and programming-base confirmations. A mathematical transformation, Fourier Transform, are used as a mapping between time (space) and frequency domain and FFT as it efficient approximative manipulating method to reduce the order of processing period. However, despite the importance place in DSP field, FFT is sensitive to spectral leakage, as defined former, and leads to inaccurate calculation. To diminish these miscalculations, it is proved that window functions can deal with this phenomenon. Hamming window and Hanning window are the chosen cases for characteristic comparison and what is the difference between the original leaking spectrum with the windowed spectrum. Another topic of reviewing is about digital filters: median filters, FIR and IIR filters. While median filter are used commonly in imaging processing because of its extraordinary characteristic; FIR and IIR filters are in the wider usage, every type has their highlight trade-off, which cannot be replace by the others.

To emphasize the mentioned topics and illustrate them as an academic way, LabVIEW is the environment for developing this application. With their strength in DSP solutions and high-level programming language, the implementation of the idea into the real program is easier than comparing to another comparable developing environments. With a potential upgradation, this program can handle more hot topics in the field of DSP.

9. Bibliography

[1] J. G. Proakis and D. G. Manolakis, Digital Signal Processing: Principles, Algorithms, and Applications, New Jersey: Prentice-Hall, 1996.

[2] F. J. Harris, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *Proceeding of IEEE,* vol. 66, pp. 51-83, 1978.

[3] A. S. a. N. Stergiou, "Power spectrum and filtering," in *Biomechanics and Gait Analysis*, Academic Press, 2020, pp. 99-148.

[4] I. Reljin, B. Reljin and V. P. a. P. Kostic, "New Window functions generated by means of time convolution-spectral leakage error," *Mediterranean Electrotechnical Conference,* vol. 2, pp. 878-881, 1998.

[5] G. V. Zaystev and A. Khzmalyan, "A Family of Optimal Window Functions for Spectral Analysis with the Spectrum Sidelobe Falloff Rate Multiple of 12 dB per Octave," *Journal of Communications Technology and Electronics,* vol. 65, pp. 502-515, 2020.

[6] K. Prabhu, "Review of Window Functions," in *Window functions and their applications in Signal Processing*, CRC Press, 2014.

[7] J. O. Smith, Spectral Audio Signal Processing, W3K Publishing, 2011.

[8] K. Fahy and E.Perez, 3 August 2020. [Online]. Available: https://www.ni.com/en-gb/innovations/white-papers/06/using-fast-fourier-transforms-and-power-spectra-in-labview.html.

10. Appendices

Appendix 1: Notation definition

				Description
$f(t)$	$T = \text{duration}(t)$	$F(\omega)$		Arbitrary continuous-time function and its CTFT
$f[n]$	$N = \text{size}[n]$	$F[q]$	$Q = \text{size}[q]$	Arbitrary discrete-time function and its DFT
m	$M = \text{size}[m]$			Number of terms of a cosine weighted window function
$w[k]$	$K = \text{size}[k]$			Discrete-time Window function
$h[n]$		$H(z)$		Impulse response (transfer function) of a LTI system
a_κ	$A = \text{size}(a_\kappa)$	b_κ	$B = \text{size}(b_\kappa)$	Reverse and forward coefficients of a LTI system
l		r		Left and right rank of a median filter

Appendix 2: <MATLAB code> Find the appropriate value for weight parameter A.

```
M = 51; n = 0:M-1; x=0; A_xmax=0; NFFT = 2048;
format long;
for A=0:0.0001:1
    window = A - (1-A) * cos (2 * pi * n / M);
    X1 = fftshift (fft (window, NFFT) / length (window));
    Freq = (-NFFT / 2: NFFT / 2-1) / NFFT;
    Y1 = X1 (1: length (Freq)) / max (X1);
    mag_dB =  20*log10(abs(Y1));
    lobe = mag_dB(islocalmax(mag_dB));
    if mod(length(lobe) , 2) == 1
    lobe((length(lobe)+1)/2)=[];
        if x>max(lobe)
        x=max(lobe);
        A_xmax=A;
        end
    end
end
x
A_xmax
```