graded by Mark Williams

86/100

Experiment 2. Data Acquisition With Labwindows

Yi-Chun Hung[1,*] and Almostafa Mohamed[1,2]
[1]*Department of Physics, Northeastern University, Boston, Massachusetts,* 02115, *USA*
[2]*My partner*
(Dated: February 6, 2023)

In this section, we collect the signal of different kinds of wave from a wave generator through lab windows and its corresponding chip made by National Instrument. Then, we perform the fast Fourier transform (FFT) on the data with different sampling frequencies in a given time to study the Nyquist theorem. Eventually, a benchmark of the results is conducted through Excel.

Keywords: *lab windows,C,Graphic User Interface,Fast Fourier Transform,Data Acquisition,Python,Matlab*

I. INTRODUCTION

instead of "different waves" state which ones -2

This section of Principle of Experimental Physics is structured to study how to use lab windows to acquire data from its corresponding instrument and perform the data analysis [1]. More specifically, we have to use lab windows to read the signal of different waves from a wave generator from a chip made by National Instrument. Then, we have to use the fast Fourier transform (FFT) to analyze our data obtained in a given time with different sampling frequencies to study the Nyquist theorem [2, 3]. The analysis still cannot be done by just running the program because each major process is linked to a button on a graphical user interface (GUI), including acquiring data and plotting the data with its Fourier transformation, saving the Fourier transformed results into a .txt file, and quitting the program. The button to quit the program is the most important button because the program windows cannot be stopped by clicking the X-symbol in the up-right corner. Not only buttons, but we also need to input the time interval for acquiring data and the sampling frequency through input boxes. Eventually, we benchmarked our results by importing the saved .txt file into excel. The methods used in this work are introduced in chapter II. The results of this work and its discussion are given in chapter III and chapter IV, respectively. Last but not least, a summary with needed future works is made in chapter V.

discussion of buttons should be in methods.

II. METHODS

We use the lines in Fig. 12 to Fig. 14 to acquire data from the wave generator and do the fast Fourier transform using GUI generated by lab windows. This chapter contains sub-chapters that briefly introduce the **functions** and important *variables* used in this section in an ordered sequence. Nevertheless, within seconds, more detailed and completed descriptions of these functions can be easily found in [4]. Please find details of some basic functions, such as **main**, in [5].

[*] hung.yi@northeastern.edu

A. quit_st

The most important function generates a button on the GUI through **QuitUserInterface**, which quits the user interface after clicking on it to avoid possible crashes caused by closing the windows improperly.

what is the daq card model? how is it wired? how is th

B. acquire_st

The function that acquires the data from the chip and plots the time domain spectrum and the FFT spectrum on the frequency domain. To do this, it first obtains the time interval to be observed and the sampling frequency from two different input boxes by **GetCrtVal**. Then, it creates the data acquisition (DAQ) tasks by **DAQmxCreateTask**, which generates the *data_st* to be used later. Then, the *data_st* is assigned to the DAQ task by **DAQmxCreateAIVoltageChan**. After that, the task is started by **DAMQStartTask** while the clock starts to count through using **DAMQmxCFgSampClktiming**. Eventually, the acquired data is assigned to *bst_data* through **DAQmxReadAnalogF64**.

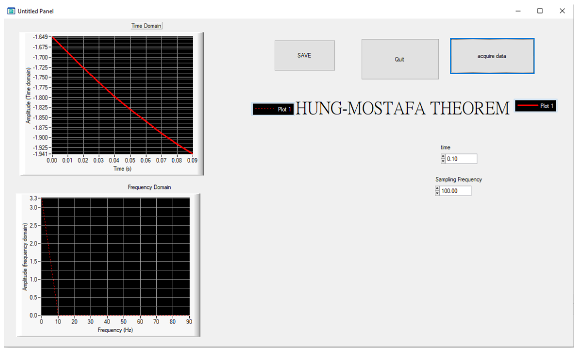
Then, a for loop is used to construct the time axis to be used in the plot of the time domain spectrum. After plotting the time domain spectrum through **PlotXY**, the *bst_data* is assigned to its FFT results through **Spectrum**. Similarly, a for loop is used to construct the frequency axis in the plot of the frequency domain spectrum. After all, the frequency domain spectrum is plotted through **PlotXY**.

C. save_st & save_ax_st

The functions that generate buttons to save the acquired data with the FFT results and the time axis with frequency axis into two files, respectively. Please find details in [5].

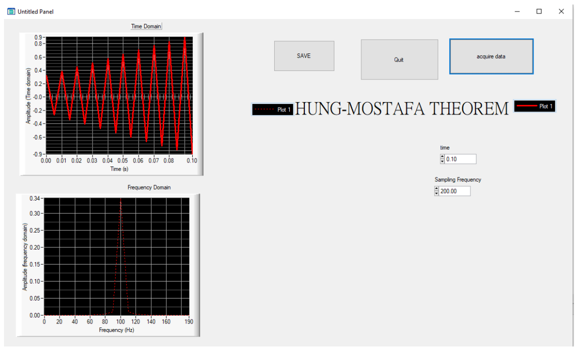
III. RESULTS

The results we obtained through the steps in chapter. II are shown in the figure below, of which the caption indicates the detailed descriptions of each result. The frequencies of the wave we used are all 100Hz.



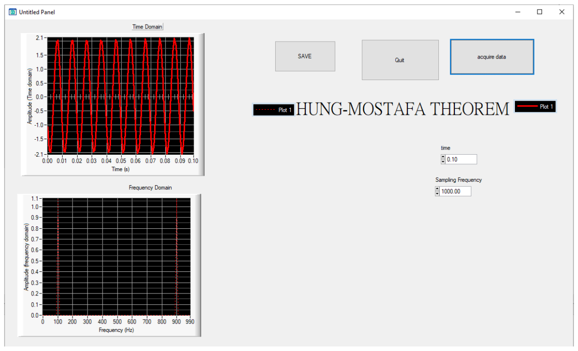
time 0.10
Sampling Frequency 100.00

FIG. 1. The time domain and frequency domain spectrum of a sine wave with sampling frequency f_{sam} are 100Hz obtained from the wave generator within 0.1 seconds.



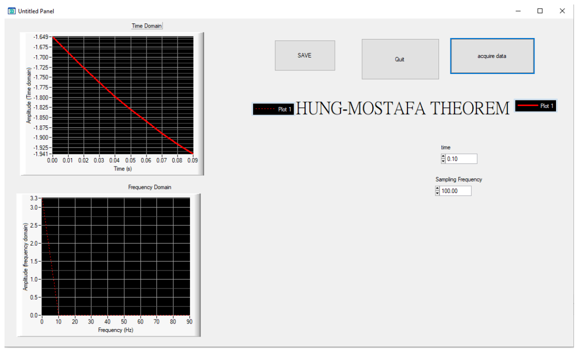
time 0.10
Sampling Frequency 200.00

FIG. 2. The time domain spectrum and frequency domain spectrum of a sine wave with sampling frequency f_{sam} are 200Hz obtained from the wave generator within 0.1 seconds



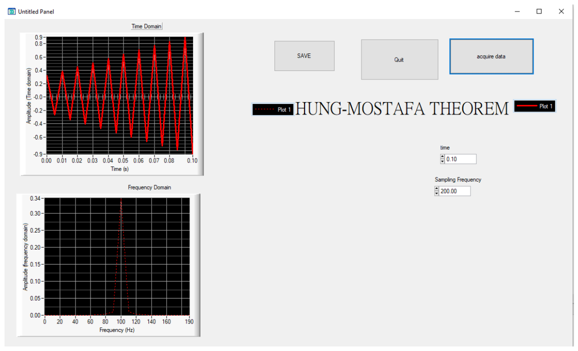
time 0.10
Sampling Frequency 1000.00

FIG. 3. The time domain spectrum and frequency domain spectrum of a sine wave with sampling frequency f_{sam} are 1000Hz obtained from the wave generator within 0.1 seconds.



time 0.10
Sampling Frequency 100.00

FIG. 4. The time domain and frequency domain spectrum of a triangular wave with sampling frequency f_{sam} are 100Hz obtained from the wave generator within 0.1 seconds.



time 0.10
Sampling Frequency 200.00

FIG. 5. The time domain and frequency domain spectrum of a triangular wave with sampling frequency f_{sam} are 200Hz obtained from the wave generator within 0.1 seconds.

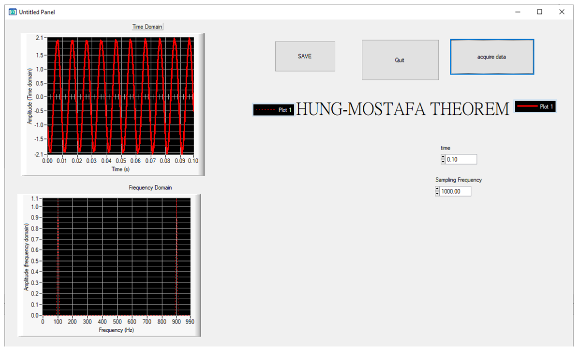
I really cannot read these tiny figures. use single columr

IV. DISCUSSION

According to the Nyquist theorem, the sampling frequency should be high enough to restore more details about the signal. The criteria for acquiring the correct FFT spectrum is called the Nyquist frequency $f_{Nyq} = \frac{1}{2} f_{sam}$, where f_{sam} is the sampling frequency. If the frequency of the signal $f \geq 2f_{Nyq}$, neither the time domain spectrum nor the frequency domain spectrum can be correctly captured. If $f = f_{Nyq}$, the frequency domain can be captured correctly, but only one peak will be observed in the FFT spectrum for a single sine wave since only half period of the wave is sampled. To have a complete FFT spectrum, $f < f_{Nyq}$ is needed, in which two peaks being mirror symmetric to f_{Nyq} will be observed in the FFT spectrum for a single sine wave due to the frequency folding of a real signal [2, 3]. Alternatively, the position of the peak in the FFT spectrum can be explained by an intuitive relation $f = f_{Nyq} + |f_{Nyq} - f_{observed}|$.

In this work, the wave's frequency is $f = 100$Hz. Therefore, in Fig. 1, Fig. 4, and Fig. 7, neither the time domain spectrum nor the frequency domain spectrum can be correctly captured since $f > f_{Nyq} = 50$Hz in these cases. In Fig. 2, Fig. 5, and Fig. 8, only the frequency domain spectrum can be correctly captured,

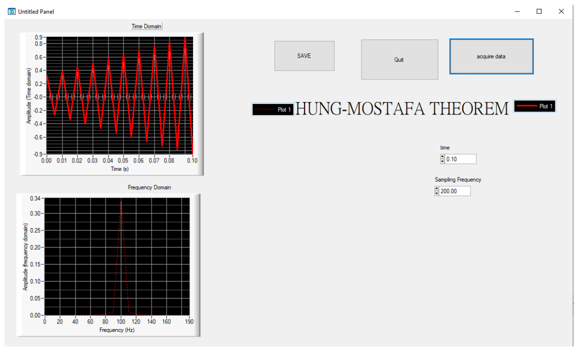
aliasing discussion is good.



time 0.10

Sampling Frequency 1000.00

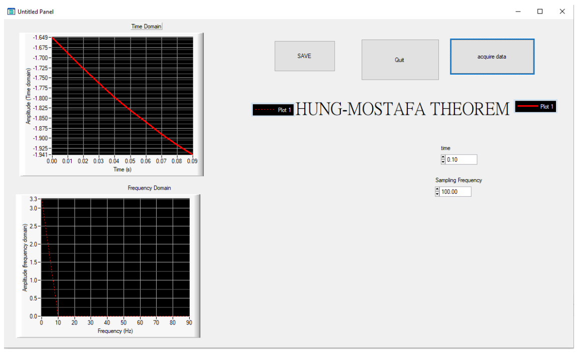
FIG. 6. The time domain and frequency domain spectrum of a triangular wave with sampling frequency f_{sam} are 1000Hz obtained from the wave generator within 0.1 seconds.



time 0.10

Sampling Frequency 200.00

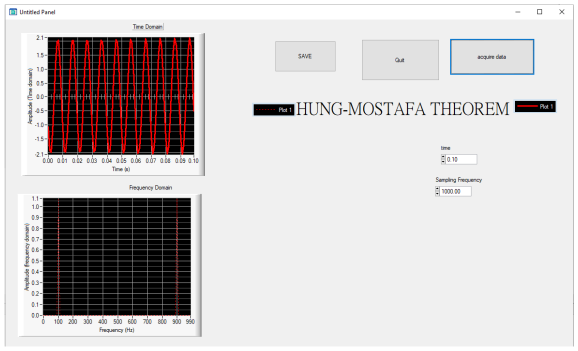
FIG. 8. The time domain and frequency domain spectrum of a square wave with sampling frequency f_{sam} are 200Hz obtained from the wave generator within 0.1 seconds.



time 0.10

Sampling Frequency 100.00

FIG. 7. The time domain and frequency domain spectrum of a square wave with sampling frequency f_{sam} are 100Hz obtained from the wave generator within 0.1 seconds.



time 0.10

Sampling Frequency 1000.00

FIG. 9. The time domain and frequency domain spectrum of a square wave with the sampling frequency f_{sam} are 1000Hz obtained from the wave generator within 0.1 seconds.

and there is only one peak in the FFT spectrum since $f = f_{Nyq} = 100$Hz in these cases. In Fig. 3, Fig. 6 and Fig. 9, both the time domain spectrum and the frequency domain spectrum can be correctly captured while two peaks appearing in the FFT spectrum since $f < f_{Nyq} = 500$Hz in these cases. Eventually, we benchmark our results through excel as demonstrated in Fig. 10 and Fig. 11. In brief, our results successfully demonstrate the Nyquist theorem, and our codes function as expected.

V. SUMMARY

In this section, we study how to acquire signal from a wave generator through lab windows and its corresponding chip made by National Instrument. Then, fast Fourier transform (FFT) analysis is performed for different kinds of waves, such as square wave, triangular wave, and sine wave, within a given time with various sampling frequencies to study the Nyquist theorem. More specifically, we use sampling frequencies 100Hz, 200Hz, and 1000Hz to observe a wave with its frequency be 100Hz within 0.1 second. The results agree with the Nyquist theorem that the FFT with 100Hz sampling frequency cannot correctly obtain the spectrum in both the time

missing discussion of harmonics for square and triangle

domain and frequency domain, which agrees with the fact that if the sampling frequency is smaller than the Nyquist frequency (200Hz in this case); on the other hand, the FFT with 200Hz sampling frequency can correctly obtain the frequency domain spectrum but not the time domain spectrum, which agrees with the fact that the sampling frequency should be larger than the Nyquist frequency to capture the time domain spectrum correctly. Last but not least, the FFT with 1000Hz sampling frequency can correctly obtain both the time domain and frequency domain spectrum as expected. Due to the frequency folding in FFT, two peaks are observed in the frequency domain.

Needed future work for this section is to switch everything to Python, which is free with powerful and intuitive data analysis packages and has a more extensive and stronger community in contrast to lab windows. Most importantly, National Instrument has provided free Python packages that support their data acquisition (DAQ) devices since 2017 [6–8]. By doing so, the technique learned in these sections can be applied to a broader range of physics. Besides, using Matlab is also another long-existing feasible option to switch in [9–11].

python daq packages are ported and much slower than C. C or C++ is used by more than half the experimental r

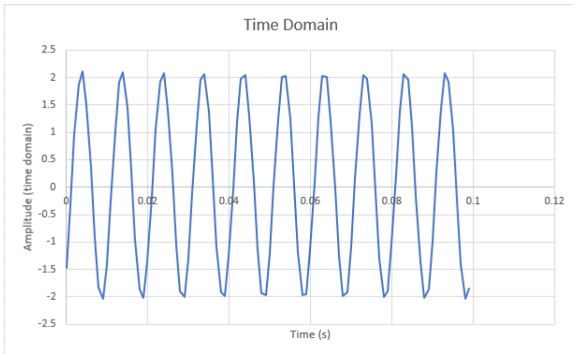


FIG. 10. The time domain spectrum plotted by excel, where the vertical line is the amplitude, and the horizontal line is time.

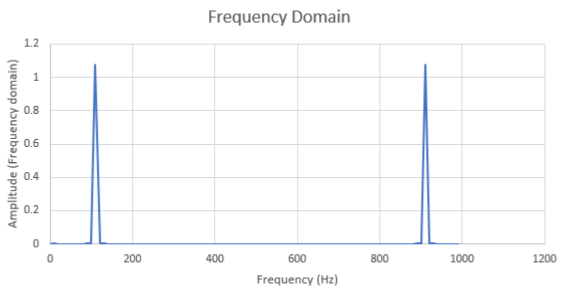


FIG. 11. The frequency domain spectrum plotted by excel, where the vertical line is the amplitude, and the horizontal line is frequency.

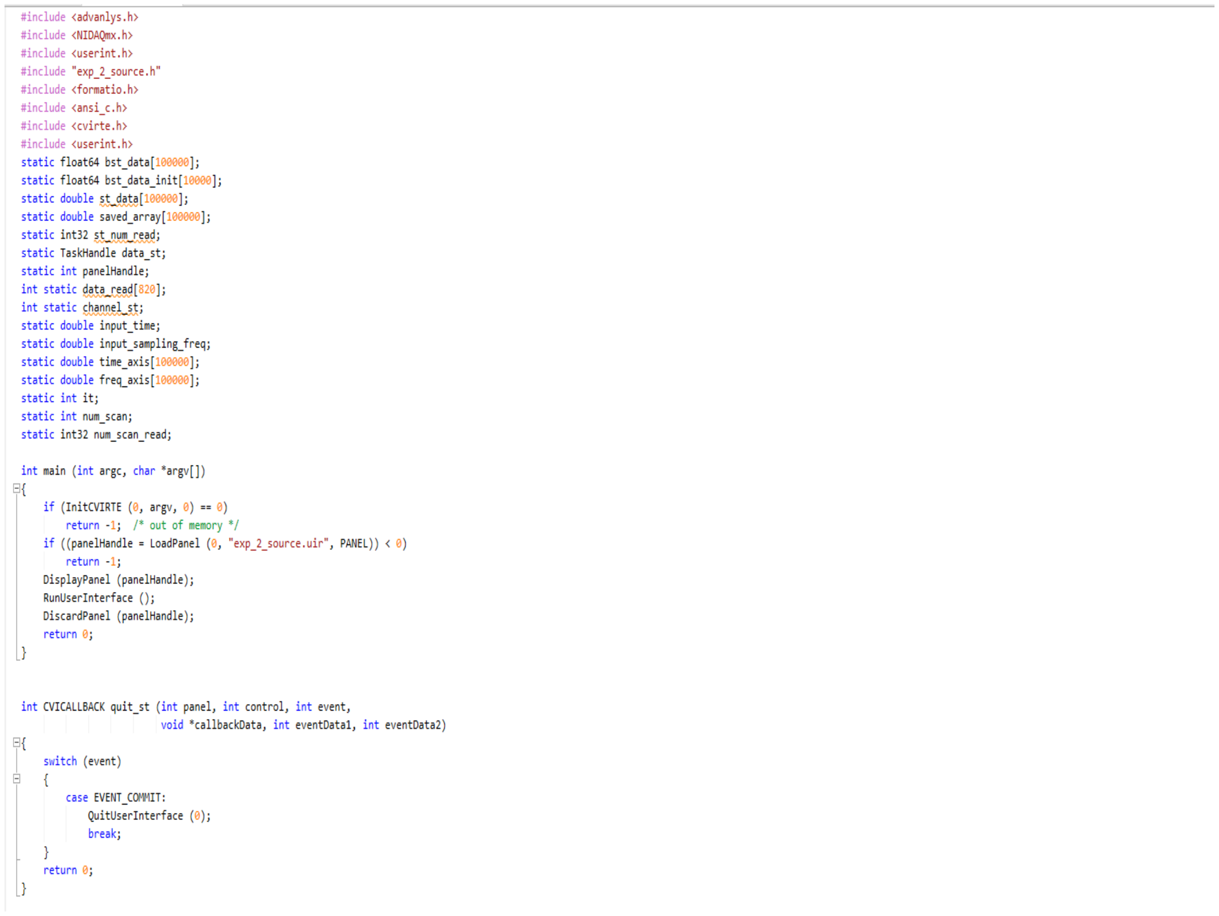
ACKNOWLEDGEMENT

The authors thank **Professor Mark Williams**, **Ben Cashen**, and **Joha Joshi**, the teaching assistant in this class, who helped us to debug our codes and were patient in answering our problems. The authors also appreciate themselves and each other for sustaining considerable hours doing something they were, are, and will not be interested in since it is never useful for theorists.

Appendix A: Codes Used In This Work

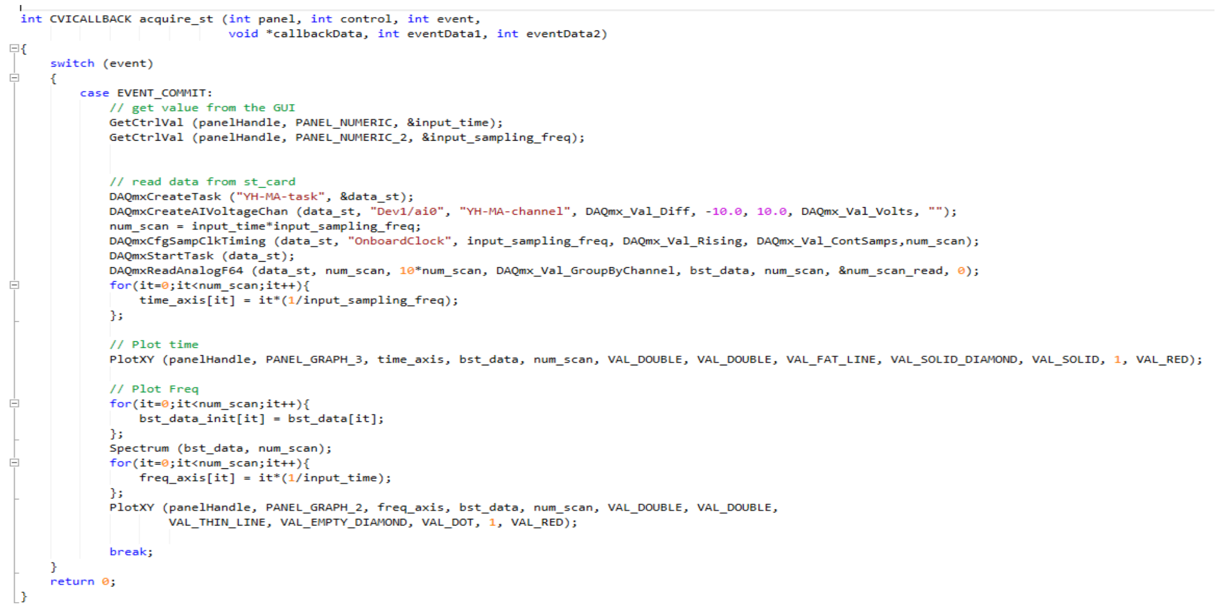
Although the link to the GitHub storing my codes is not required, I'll just put it here like normal people usually do in a journal paper `https://github.com/lengentyh/ST_Lab-Windows/tree/main` because normally people who really care about the codes want to see the codes in their editor instead of figures. Still, the screenshots of the codes I used in this section are listed below.

[1] N. U. Phys5318, Experiment 2: Data acquisition with labwindows (2023).
[2] H. Nyquist, Certain topics in telegraph transmission theory, Transactions of the American Institute of Electrical Engineers **47**, 617 (1928).
[3] L. Tan and J. Jiang, Chapter 4 - discrete fourier transform and signal spectrum, in *Digital Signal Processing (Third Edition)*, edited by L. Tan and J. Jiang (Academic Press, 2019) third edition ed., pp. 91–142.
[4] https://www.ni.com/docs/en-us/bundle/labwindowscvi/page/cvi/intro.htm.
[5] https://github.com/lengentyh/st˙lab-windows/blob/main/annotated-exp.1-3.pdf.
[6] https://www.ni.com/en-us/support/documentation/supplemental/16
resources-for-ni-hardware-and-software.html ().
[7] https://nidaqmx-python.readthedocs.io/en/latest ().
[8] https://github.com/ni/nidaqmx-python ().
[9] https://www.mathworks.com/help/daq/getting-started-with-session-based-interface-using-ni-devices.html ().
[10] https://www.mathworks.com/help/daq/acquire-continuous-and-background-data-using-ni-devices.html ().
[11] https://www.mathworks.com/help/daq/acquire-data-using-ni-fielddaq-device.html ().



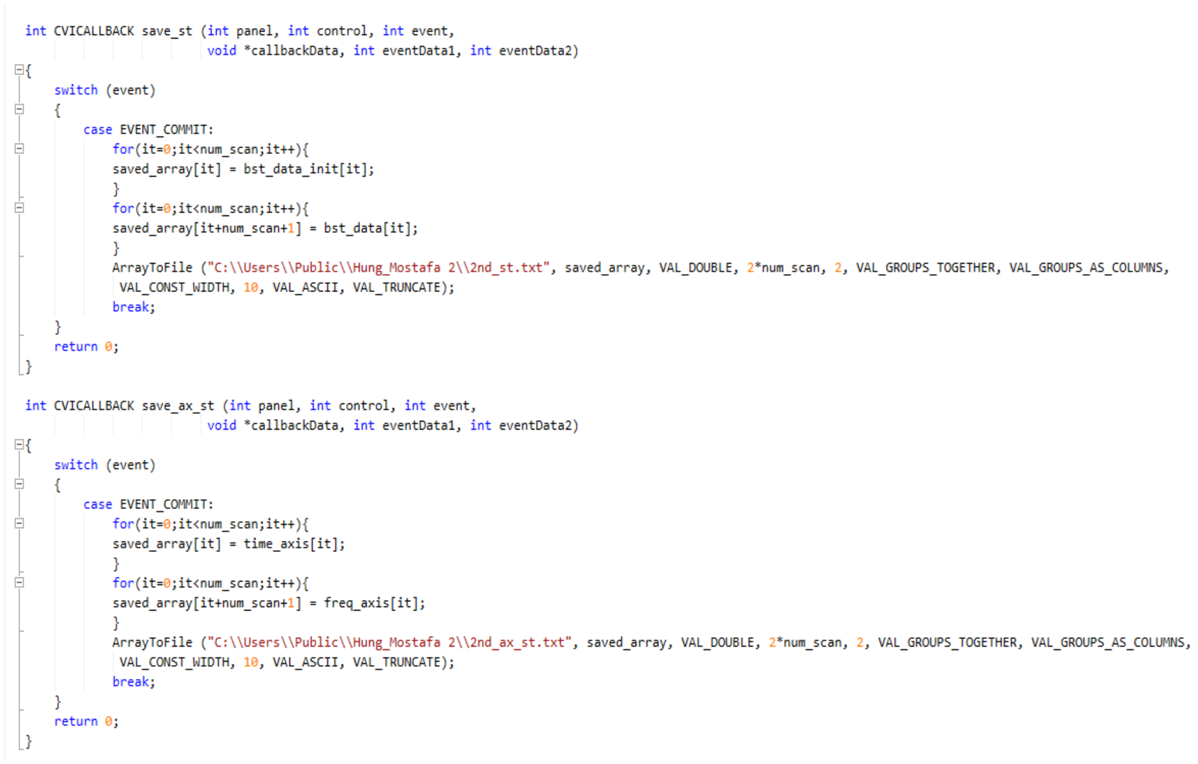
```c
#include <advanlys.h>
#include <NIDAQmx.h>
#include <userint.h>
#include "exp_2_source.h"
#include <formatio.h>
#include <ansi_c.h>
#include <cvirte.h>
#include <userint.h>
static float64 bst_data[100000];
static float64 bst_data_init[10000];
static double st_data[100000];
static double saved_array[100000];
static int32 st_num_read;
static TaskHandle data_st;
static int panelHandle;
int static data_read[820];
int static channel_st;
static double input_time;
static double input_sampling_freq;
static double time_axis[100000];
static double freq_axis[100000];
static int it;
static int num_scan;
static int32 num_scan_read;

int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0)
        return -1;  /* out of memory */
    if ((panelHandle = LoadPanel (0, "exp_2_source.uir", PANEL)) < 0)
        return -1;
    DisplayPanel (panelHandle);
    RunUserInterface ();
    DiscardPanel (panelHandle);
    return 0;
}


int CVICALLBACK quit_st (int panel, int control, int event,
                         void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            QuitUserInterface (0);
            break;
    }
    return 0;
}
```

FIG. 12. The first paragraph of our lab windows source code.



```c
int CVICALLBACK acquire_st (int panel, int control, int event,
                            void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            // get value from the GUI
            GetCtrlVal (panelHandle, PANEL_NUMERIC, &input_time);
            GetCtrlVal (panelHandle, PANEL_NUMERIC_2, &input_sampling_freq);


            // read data from st_card
            DAQmxCreateTask ("YH-MA-task", &data_st);
            DAQmxCreateAIVoltageChan (data_st, "Dev1/ai0", "YH-MA-channel", DAQmx_Val_Diff, -10.0, 10.0, DAQmx_Val_Volts, "");
            num_scan = input_time*input_sampling_freq;
            DAQmxCfgSampClkTiming (data_st, "OnboardClock", input_sampling_freq, DAQmx_Val_Rising, DAQmx_Val_ContSamps,num_scan);
            DAQmxStartTask (data_st);
            DAQmxReadAnalogF64 (data_st, num_scan, 10*num_scan, DAQmx_Val_GroupByChannel, bst_data, num_scan, &num_scan_read, 0);
            for(it=0;it<num_scan;it++){
                time_axis[it] = it*(1/input_sampling_freq);
            };

            // Plot time
            PlotXY (panelHandle, PANEL_GRAPH_3, time_axis, bst_data, num_scan, VAL_DOUBLE, VAL_DOUBLE, VAL_FAT_LINE, VAL_SOLID_DIAMOND, VAL_SOLID, 1, VAL_RED);

            // Plot Freq
            for(it=0;it<num_scan;it++){
                bst_data_init[it] = bst_data[it];
            };
            Spectrum (bst_data, num_scan);
            for(it=0;it<num_scan;it++){
                freq_axis[it] = it*(1/input_time);
            };
            PlotXY (panelHandle, PANEL_GRAPH_2, freq_axis, bst_data, num_scan, VAL_DOUBLE, VAL_DOUBLE,
                    VAL_THIN_LINE, VAL_EMPTY_DIAMOND, VAL_DOT, 1, VAL_RED);

            break;
    }
    return 0;
}
```

FIG. 13. The second paragraph of our lab windows source code.



```
int CVICALLBACK save_st (int panel, int control, int event,
                         void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            for(it=0;it<num_scan;it++){
            saved_array[it] = bst_data_init[it];
            }
            for(it=0;it<num_scan;it++){
            saved_array[it+num_scan+1] = bst_data[it];
            }
            ArrayToFile ("C:\\Users\\Public\\Hung_Mostafa 2\\2nd_st.txt", saved_array, VAL_DOUBLE, 2*num_scan, 2, VAL_GROUPS_TOGETHER, VAL_GROUPS_AS_COLUMNS,
             VAL_CONST_WIDTH, 10, VAL_ASCII, VAL_TRUNCATE);
            break;
    }
    return 0;
}

int CVICALLBACK save_ax_st (int panel, int control, int event,
                         void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            for(it=0;it<num_scan;it++){
            saved_array[it] = time_axis[it];
            }
            for(it=0;it<num_scan;it++){
            saved_array[it+num_scan+1] = freq_axis[it];
            }
            ArrayToFile ("C:\\Users\\Public\\Hung_Mostafa 2\\2nd_ax_st.txt", saved_array, VAL_DOUBLE, 2*num_scan, 2, VAL_GROUPS_TOGETHER, VAL_GROUPS_AS_COLUMNS,
             VAL_CONST_WIDTH, 10, VAL_ASCII, VAL_TRUNCATE);
            break;
    }
    return 0;
}
```

FIG. 14. The third paragraph of our lab windows source code.