An Introduction to LabVIEW Fundamentals

Lauren Hernandez, Kathryn Wong

University of Houston

Physics 3313: Advanced Laboratory I

Abstract

Fundamental knowledge of LabVIEW was gained in this experiment through setting up a boolean program, testing and calibrating equipment, working with voltage data aquisition, and processing signals created by voice and tuning forks. The resonant frequency of the unlabeled tuning fork was 325*Hz.*

I. Introduction

In many circumstances, the tasks of a researcher, working-professional or student, may include building complex systems or processing large amounts of information. When these circumstances arise, complicated tasks have the potential to turn into one-step executions with the aid of computer programming. LABVIEW is a graphical programming language with various applications, including data aquisition, test automation, embedded system design, industrial/instrument control, and analysis and signal processing [1].

The goal of this experiment is to gain experience with LabVIEW for assisted data aquisition, instrument testing, and instrument control to ultimately determine the resonant frequency of a tuning fork.

II. Experimental Procedure

Equipment

The equipment we used for this experiment were LabVIEW 2012 software, DAQ Series X Manual, Lenovo Windows Computer, Acer Computer Screen Monitor, EXTECH Instruments True RMS Multimeter EX430A, Nady Audio SMPS - 1X adapter YL1800350, NI CB 68-LP Connection Block, phantom power supply 1971504645, Behringer C-2 condenser microphone J1417034263, a rubber mallet, and three tuning forks of frequency 435 Hz, 256 Hz and one of an unknown frequency.

Part One: Simple Boolean Program

We opened LabVIEW and created a blank VI. In the Front Panel window, we put two switches naming them "Premise A" and "Premise B", and put four indicators, naming them "A", "B", "A and B" and "A or B", shown below in Figure 1.

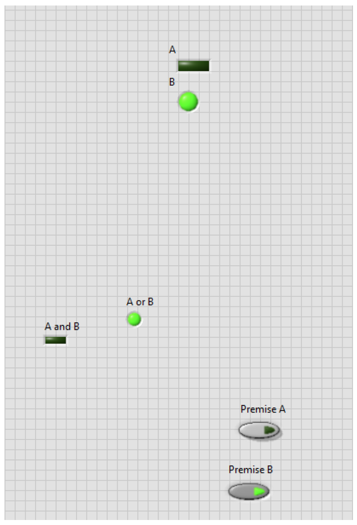


Figure 1. *Two switches and four indicators located in the Front Panel window, used to set up a simple boolean program.*

We went to the Block Diagram window, arranged the switches vertically downward, and wired "Premise A" to "A" and "Premise B" to "B".We then returned to the Front Panel, changed the truth value of a random switch, and then ran the program by clicking "Run". We then clicked "Run Continuously", followed by "Abort Execution". We returned to the Block Diagram window and clicked "Highlight Execution" and then ran the program again. We right clicked in an empty space in the Block Diagram window, pulled up the Functions Palette, placed a boolean "And" to the left of the "A and B" indicator, and placed an "Or" to the left of the "A or B" indicator. We wired "Premise A" and "Premise B" to the "And" conditional, ran the program, received an error, and then made connections to the "Or" conditional. Then, we deselected "Highlight Execution", went to the Front Panel window and ran the program continuously. We returned to the Functions Palette, selected a While Loop and placed all of our code inside of the While Loop. We added a control to the Stop Control terminal on the loop, and placed a "Wait Function" inside the loop in the Block Diagram window. We placed a numeric constant of 100 inside the loop, wiring it to the left input of terminal of the "Wait Function". Then, we returned to the Front Panel window and clicked run, as shown below in Figure 2.

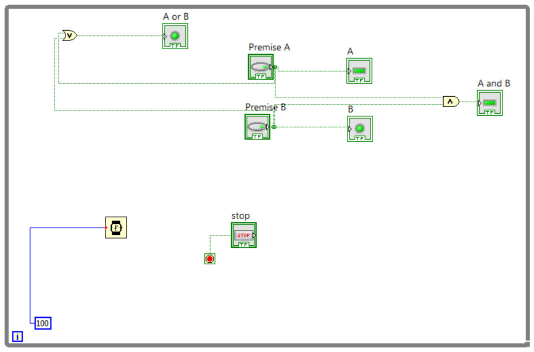
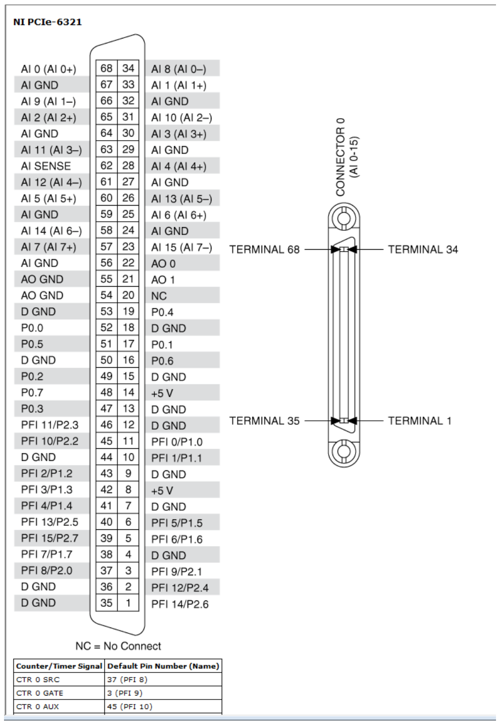


Figure 2. *A simple boolean program displayed in the Block Diagram window of LabVIEW.*

Part Two: The Measurement and Automation Explorer (NI MAX)

We checked the connection between the Connection Block and the DAQ card on the computer, opened the NI-MAX icon on the desktop, clicked "Devices and Interfaces" and selected "NI PCI-6014". We right clicked the device name and selected "Device Pinouts", shown below in Figure 3.


NI PCIe-6321

AI 0 (AI 0+)	68	34	AI 8 (AI 0–)
AI GND	67	33	AI 1 (AI 1+)
AI 9 (AI 1–)	66	32	AI GND
AI 2 (AI 2+)	65	31	AI 10 (AI 2–)
AI GND	64	30	AI 3 (AI 3+)
AI 11 (AI 3–)	63	29	AI GND
AI SENSE	62	28	AI 4 (AI 4+)
AI 12 (AI 4–)	61	27	AI GND
AI 5 (AI 5+)	60	26	AI 13 (AI 5–)
AI GND	59	25	AI 6 (AI 6+)
AI 14 (AI 6–)	58	24	AI GND
AI 7 (AI 7+)	57	23	AI 15 (AI 7–)
AI GND	56	22	AO 0
AO GND	55	21	AO 1
AO GND	54	20	NC
D GND	53	19	P0.4
P0.0	52	18	D GND
P0.5	51	17	P0.1
D GND	50	16	P0.6
P0.2	49	15	D GND
P0.7	48	14	+5 V
P0.3	47	13	D GND
PFI 11/P2.3	46	12	D GND
PFI 10/P2.2	45	11	PFI 0/P1.0
D GND	44	10	PFI 1/P1.1
PFI 2/P1.2	43	9	D GND
PFI 3/P1.3	42	8	+5 V
PFI 4/P1.4	41	7	D GND
PFI 13/P2.5	40	6	PFI 5/P1.5
PFI 15/P2.7	39	5	PFI 6/P1.6
PFI 7/P1.7	38	4	D GND
PFI 8/P2.0	37	3	PFI 9/P2.1
D GND	36	2	PFI 12/P2.4
D GND	35	1	PFI 14/P2.6

NC = No Connect

Counter/Timer Signal	Default Pin Number (Name)
CTR 0 SRC	37 (PFI 8)
CTR 0 GATE	3 (PFI 9)
CTR 0 AUX	45 (PFI 10)

Figure 3. *Device pinout diagram for NI PCI-6014 device.*

We clicked the "Self Test" button, checked to make sure all of the terminals on the Connection Block were open, and then clicked "Self-Calibrate". We connected a DC voltage source of 6 volts in Non-Referenced Single Ended mode (NRSE) to the input channels on the Connection Block. We selected input 68, 62, and 56, which correspond to AChannel0, AISENSE, and AIGround. We clicked "Test Panels", clicked the channel connected to the voltage source, set the Mode to "Continuous", set the configuration to NRSE, and clicked start. We then recorded the voltage when measured by a multimeter. Then, we switched from AChannel0 to DAC0OUT, and measured the voltage with a multimeter.

Part Three: Simple Voltage Measurement Program

We created a new VI, and added a DAQ Assistant to the Block Diagram window. We selected "Aquire Signals", "Analog Input", "Voltage", "ai0", and then "Finish". We set the Signal Input

Range to 0-10, set the Terminal Configuration to NRSE, Aquisition Mode to "N Samples", and clicked "OK". We created controls for "Number of Samples" and "Rate", and then created a "Graph Indicator" for the "data" terminal output. In the Block Diagram window, we created a blank "Array Constant" and put a "Double Numeric Constant" inside of it, and changed it to an indicator. We wired the DAQ Assistant data output to the array, and expanded the indicator for the array to show multiple output values. Then, we ran the program. We returned to the Block Diagram window and added "Array Max & Min", "Add Array Elements", and "Divide" functions. We wired the DAQ Assistant's data output terminal to "Array Max & Min" and "Add Array Elements", and wired "Add Array Elements" and "Numher of Samples" to the "Divide" function. We then made indicators for Max, Min and Divide and ran the program. In the Block Diagram window, we added a "Write to Spreadsheet File" function, wired controls to file path and format terminals, and wired a constant to "transpose". We entered a filepath for the data and wired the array to the "Write Spreadsheet" input and ran the program. Then, we highlighted all of the functions and VI's in the Block Diagram windowNd created a SubVI, and saved it.

Part Four: Introduction to Signal Processing

We connected a microphone to a phantom microphone and wired the phantom power supply to an analog channel on the DAQ Connection Block in NRSE mode. We left the connections at the Connection Block set up as we had it in part three of the experiment. Then, we switched the phantom power unit on. We created a new VI, and put a DAQ Assistant VI in the Block Diagram window. We wired controls to the DAQ Assistant for "Stop", "Number of Samples", and "Sample Rate". We then wired a "Waveform Graph" to the "data" output. We varied the value for "Number of Samples" and "Rate" and noticed the difference in output plots. Then, we placed a "Convert" from the "Dynamic" function into the Block Diagram window and set it to output a "Single Waveform", by wiring the signal output to the conversion function. We put an FFT Power Spectrum VI in the Block Diagram window, and connected the output of the conversion function to the "Time Signal" on the FFT power spectrum. We put 'Unbundle By Name" into the Block Diagram window and wired it to the FFT Power Spectrum. Then we wired the output, "Magnitude" array, to a second waveform graph and created a second graph on the Front Panel window; then we ran the program. Then we used the two labeled tuning forks and our voices while saying the word "Hello" to understand the relationship between the number of samples, sample rate, FFT frequency and the time over which the data was collected. Then, we used this information to interpret the FFT plot of the unknown tuning fork.

III. Results, Analysis, Discussion

Part One: Simple Boolean Program

The resulting boolean program was set up in both the Front Panel window and in the Block Diagram window, as seen in Figures 1 and 2 in the experimental procedure section. When the switches are green, they are in "On" mode, and when the switches are not green, they are in "Off" mode. We set up the program to where one of each switch or indicator pairs was on and one was off. We set "A" to be "Off", and "B" to be "On"; we set "A and B" to be "Off", and "A or B" to be "On"; and we set "Premise A" to be "Off" and "Premise B" to be "On". The program ran seamlessly, so long as "Premise A" and "Premise B" were connected to at least one switch. We connected the premise indicators, each to three switches, so that there could be various outcomes depending on which switch was "On" and which switch was "Off".

Part Two: The Measurement and Automation Explorer (NI MAX)

The voltage source was labeled as having a 6V power supply, and the test pannel produced a voltage reading of $6.00\text{V} \pm 1\text{V}$ for connections to both A1GND and A0GND. The multimeter produced a steady reading of $6.00\text{V} \pm 1\text{V}$ for connections to both A1GND and A0GND. The test pannel reading and multimeter reading for connections to A1GND and A0GND are shown below in Table 1 and Table 2.

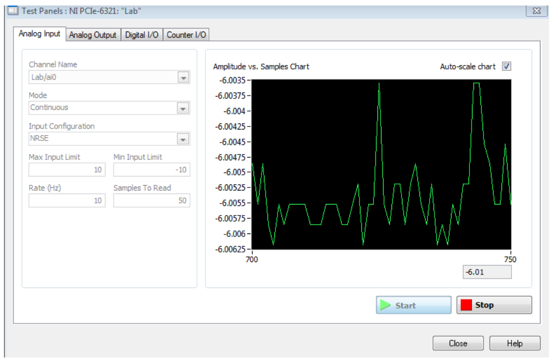
TABLE I. *Test panel and multimeter reading of a 6V voltage source, using a Connection Block's A1GND terminal.*

A1GND	Test Pannel	Multimeter
	6.01 V	6.01 V
	6.00 V	6.01 V
	6.02 V	6.01 V

TABLE II. *Test panel and multimeter reading of a 6V voltage source, using a Connection Block's A0GND terminal.*

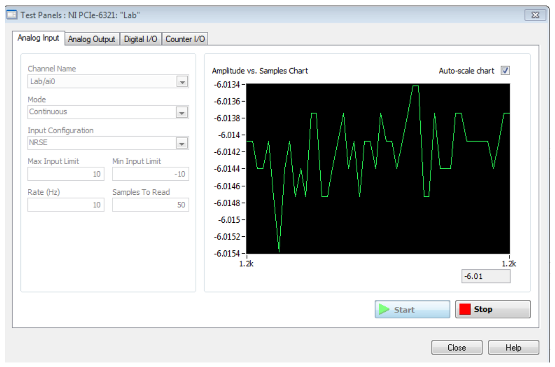
A0GND	Test Pannel	Multimeter
	6.00 V	6.01 V
	6.01 V	6.01 V
	6.00 V	6.01 V

The output waveforms of the voltage (amplitude) vs. samples taken while using terminal A1GND and A0GND are shown below in Figure 4 and Figure 5.


Test Panels : NI PCIe-6321: "Lab"

Channel Name: Lab/ai0
Mode: Continuous
Input Configuration: NRSE
Max Input Limit: 10
Min Input Limit: -10
Rate (Hz): 10
Samples To Read: 50
Auto-scale chart ✓
-6.01

Figure 4. *Voltage (amplitude) vs. Samples of a 6V power source, using a Connection Block's A1GND terminal.*


Test Panels : NI PCIe-6321: "Lab"

Channel Name: Lab/ai0
Mode: Continuous
Input Configuration: NRSE
Max Input Limit: 10
Min Input Limit: -10
Rate (Hz): 10
Samples To Read: 50
Auto-scale chart ✓
-6.01

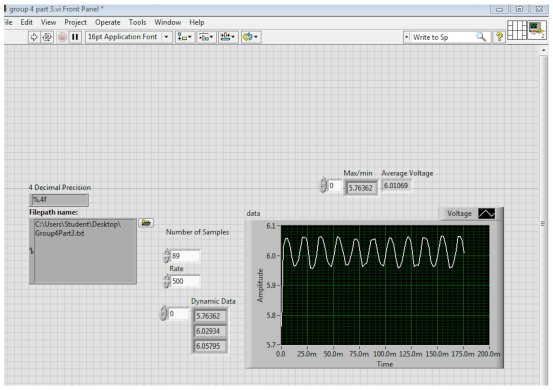
Figure 5. *Voltage (amplitude) vs. Samples of a 6V power source, using a Connection Block's A0GND terminal.*

The sample rate was an important variable in when we were gathering data for the voltage reading, because the higher the rate was, the more data we were collecting per second. This was useful, becuase the more data we had, the more accurately the voltage over time curve represented the actual voltage of the power supply.

There were no complications with this part of the experiment. The test pannel and multimeter readings both confirmed the value of the voltage source. The success of this part of the experiment could be due to the fact that we made sure the connections we placed between the voltage source, the terminals in the connection block, the computer, and mulitmeter were all secure and in-tact throughout the experiment. Additionally, we tested the functions on the multimeter prior to using it, to make sure that it was fully functional for the experiment.

Part Three: Simple Voltage Measurement Program

After we set up the program, the average output reading of the 6V power source was 6.01069 V. The output waveform of voltage over time is shown below in Figure 6.


group 4 part 3.vi Front Panel *
File Edit View Project Operate Tools Window Help
16pt Application Font
Write to Sp
data
Voltage
Amplitude
Time
6.1
6.0
5.9
5.8
5.7
0.0 25.0m 50.0m 75.0m 100.0m 125.0m 150.0m 175.0m 200.0m

4 Decimal Precision: %.4f
Filepath name: C:\Users\Student\Desktop\Group4Part3.txt
Number of Samples: 89
Rate: 500
Max/min: 0, 5.76362
Average Voltage: 6.01069
Dynamic Data: 0, 5.76362, 6.02934, 6.05795

Figure 6. *Voltage measurement function,voltage (amplitude) vs time, shown in the Front Pannel window using a 6V power source. The desired filepath chosen is visible from the window.*

Shown below in Figure 7 is the Block Diagram window after we finished setting up the program and checked it for errors.

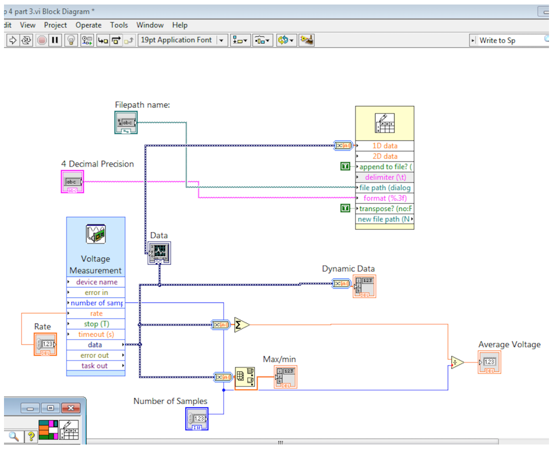
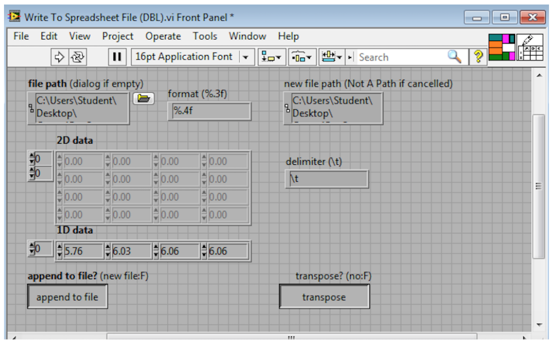


Figure 7. *Voltage measurement program set up in the Block Diagram window using a 6V power source.*

Shown below in Figure 8 is the "Write to Spreadsheet" VI in a sub- Front Panel window. In this widow, we specified the filepath that we wanted to use for data collection, we specified a 4 digit decimal precision, and we chose to not transpose our data.


Write To Spreadsheet File (DBL).vi Front Panel *
File Edit View Project Operate Tools Window Help
16pt Application Font
Search

file path (dialog if empty): C:\Users\Student\Desktop\
format (%.3f): %.4f
new file path (Not A Path if cancelled): C:\Users\Student\Desktop\

2D data

0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00

delimiter (\t): \t

1D data

5.76	6.03	6.06	6.06

append to file? (new file:F): append to file
transpose? (no:F): transpose

Figure 8. *Shown is the "Write to Apreadsheet" command that was set up in the Block Diagram window, with sample data displayed to a 4-point decimal precision, and the chosen filepath displayed.*

We had no unsolvable errors with this part of the experiment. Kathryn and I both have experience with programming, and we set up the program as instructed and the program ran without bugs. We left the Command Block set up securely, as it was in the prior parts of the experiment, and this could have contributed to the success of part three of the experiment.

Part Four: Introduction to Signal Processing

When striking each tuning fork with the rubber mallet, we output two plots. One voltage plot, which is represented as amplitude vs time, and a plot of the Fast Fourier Transform of the voltage plot, represented as intensity vs frequency. The voltage plot was produced by turning sound waves into a volatage wave, and the peaks and pits in the voltage plot represent the rise and fall of how the tuning fork sounds when struck - it sounds like an oscillating ringing, with waves of "loudness" followed by a wave of "quietness". The Fast Fourier Transform of the voltage wave showed us all of the different frequencies that compiled, and superimposed to create the overall sound given off by the respective tuning fork. Each tuning fork's total frequency was made by the summation of several frequencies and those several frequencies are shown in the Fast Fourier Transform plot, showing to what ratio, or quantity, did each coomponent frequency contribute to the overall sound of the tuning fork.

Shown below in Figure 9 are the output voltage and FFT plots of a 256 Hz tuning fork when struck with a rubber mallet.

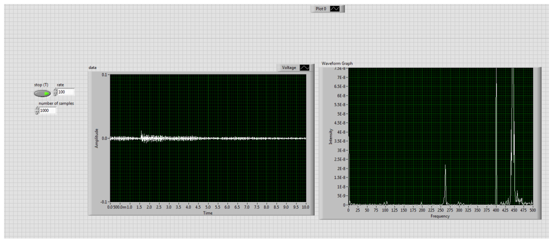


Figure 9. *Shown above on the left is the voltage waveform, and on the right is the FFT of the voltage waveform; both curves represent a 256 Hz tuning fork after it was struck with a rubber mallet.*

Shown below in Figure 10 are the output voltage and FFT plots of a 435 Hz tuning fork when struck with a rubber mallet.

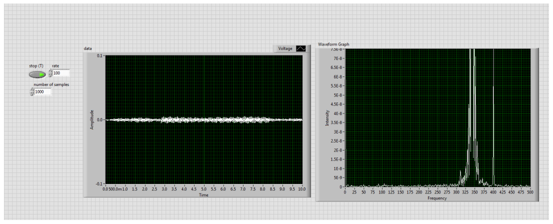


Figure 10. *Shown above on the left is the voltage waveform, and on the right is the FFT of the voltage waveform; both curves represent a 435 Hz tuning fork after it was struck with a rubber mallet.*

Shown below in Figure 11 are the output voltage and FFT plots of a tuning fork of unknown frequency when struck with a rubber mallet.

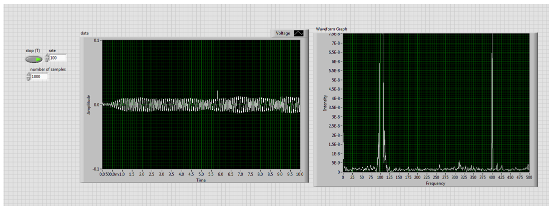


Figure 11. *Shown above on the left is the voltage waveform, and on the right is the FFT of the voltage waveform; both curves represent a tuning fork of unknown frequency after it was struck with a rubber mallet.*

Further, in each voltage plot, the amplitude represents the amount of voltage recorded over time. In each FFT plot of the voltage, the intensity represents the magnitude to which each contributing frequency is present in the overall total frequency of the tuning fork, and the freqency lists range of frequencies present. The frequency of the unknown tuning fork was determined to be 325Hz.

IV. Conclusion

In conclusion, LabVIEW is a program that utilizes programming in a digaram format to process information, and is quite easy to use. The simple boolean program was straightforward and the programming involved helped in understanding how to navigate through LabVIEW. Part two and three of the experiment proved the degree of accuracy to which LabVIEW can test various equipment. Lastly, part four of the experiment gave the results of the frequency of an unknown tuning fork, which was 325 Hz.

References

[1] Wikimedia Foundation. (2021, November 7). LabVIEW. Wikipedia. Retrieved November 7, 2021, from https://en.wikipedia.org/wiki/LabVIEW.

[2] DAQ NI 6013/6014 User Manual, National Instruments,available in Phys 3313 laboratory and at https://kur.web.psi.ch/sans1/manuals/NI%206013%20and%206014%20User%20Manual.pdf, accessed November 7, 2021.

[3] *DAQ E Series User Manual*, National Instruments http://www.ni.com/pdf/manuals/370503k.pdf, accessed November 7, 2021.

[4] Ulinski, A. & Forrest, R. L., "Lab Manual for Advanced Laboratory I", Physics 3313, *A Crash Course in LabVIEW Fundamentals*, pages 1 - 8, (The University of Houston, 2021).