AC 2008-1017: BUILDING HARDWARE-BASED LOW-COST EXPERIMENTAL DSP LEARNING MODULES

A. Uluagac, Georgia Institute of Technology

A. Selcuk Uluagac is a Ph.D. student in the School of Electrical and Computer Engineering at Georgia Institute of Technology , Atlanta, GA as a member of the Communications Systems Center. He received his B.Sc. in Computer Engineering from Turkish Naval Academy and M.Sc. degrees in Electrical and Computer Engineering from Carnegie Mellon University in PA,USA, in 1997 and 2002, respectively. He is a member of IEEE and ASEE.

Douglas Williams, Georgia Institute of Technology

Douglas B. Williams received the BSEE, MS, and PhD degrees in electrical and computer engineering from Rice University, Houston, Texas, in 1984, 1987,and 1989, respectively. In 1989, he joined the faculty of the School of Electrical and Computer Engineering at the Georgia Institute of Technology, Atlanta, Georgia, where he is currently Professor and Associate Chair for Undergraduate Affairs. There he is also affiliated with the Center for Signal and Image Processing and the Arbutus Center for Distributed Engineering Education.

Dr. Williams has served as an Associate Editor of the IEEE Transactions on Signal Processing and the EURASIP Journal of Applied Signal Processing. He is currently on the IEEE Signal Processing Society's SPTM Technical Committee and Education Technical Committee, and he has been a member of the Society's Board of Governors. He is currently Area Editor--Special Issues for the IEEE Signal Processing Magazine. Dr. Williams was co-editor of the Digital Signal Processing Handbook published in 1998 by CRC Press and IEEE Press. He is a member of the Tau Beta Pi, Eta Kappa Nu, and Phi Beta Kappa honor societies.

Building Hardware-Based Low-Cost Experimental DSP Learning Modules

Abstract

Students often face difficulties in grasping and understanding fundamental digital signal processing (DSP) concepts in the classroom environment. Most DSP courses do not have labs associated with them, and the ones that do typically depend on the usage of text-based programming languages (e.g., MATLAB) that do not allow real-time external hardware interaction. Although the availability of external DSP platforms (e.g. National Instruments Speedy-33, Texas Instruments DSP Student Kits, etc.) for educational purposes has increased because of recent advances in embedded processors and sensor technology, they are rarely used in early DSP courses because of the programming sophistication needed for real-time processing.

In this paper we discuss the development of low-cost experimental DSP learning modules for classes that do not usually have a lab component. These modules use National Instruments' LabVIEW for their programming and development platform with the Speedy-33 DSP board and LEGO Mindstorms NXT Brick as the hardware platforms. Many of the modules can also be run completely on the host computer's sound card. Modules have been developed for examining different aspects of topics such as sampling, aliasing, and filtering, while working with data that has been captured and processed in real-time. Students are able to interact with the hardware and data through GUIs, thus obviating the need to first develop real-time programming skills. This paper will describe these modules and how they are designed to be used both in lectures and as part of homework assignments.

1. Introduction

Recent advances in embedded processors and sensor technology have made it possible to use external hardware platforms, such as National Instruments (NI) Speedy-33 DSP Board,[1] Texas Instruments DSP Student Kits,[2] LEGO NXT Brick,[3] etc., for educational purposes. Similarly, we have begun to see the proliferation of graphic programming languages, which utilize icons or graphics to implement programming language constructs (e.g., if statements) or functions, such as NI's LabVIEW,[4] Visual Application Builder (VAB) Software,[5] or LEGO Mindstorms software.[3] Students often find it easier to learn and implement programs using these graphical languages. One other nice feature of these programming languages is that they allow for real-time interaction with external hardware platforms. Thus, a combined use of external hardware platforms and graphical programming languages can facilitate the need of both educators and students in teaching and learning fundamental digital signal processing concepts in the undergraduate curriculum, which can be relatively difficult topics for students to grasp.

However, currently neither the external DSP hardware platforms nor the graphical programming languages are commonly facilitated in the classroom settings with this understanding. This failure could be attributed to the programming sophistication needed for real-time processing and the lack of the knowledge of the presence of such platforms and programming environments.

In this paper we introduce the development of low-cost hardware-based experimental DSP learning modules. We use the Speedy-33 external DSP board from NI and the LEGO

Mindstorms NXT Brick as our hardware platforms, and NI's LabVIEW as our software environment. The modules teach students fundamental concepts, such as sampling, aliasing, FFT, and filtering, via interactive GUIs acting on data that is captured and processed in real-time. The modules have been designed to be used in early DSP classes that do not necessarily have associated labs and/or classes that typically utilize only textual programming languages (e.g., MATLAB), which lack the real-time external hardware interaction. Moreover, the modules can be extended to teach other advanced DSP concepts and would be useful beyond the undergraduate curriculum, as well.

The learning modules can be prepared and distributed to students as pre-compiled modules, where students need only run the applications without further programming. Alternatively, the modules can be given to students as partially or wholly implemented source codes, where the students would be expected to program or modify the provided code. The methods of preparation and distribution are a matter of choice.

There are several benefits of employing experimental DSP learning modules. First of all, use of a graphical programming language like LabVIEW can ease the implementation of relatively harder DSP concepts. Second, the modules can be extended easily for the implementation of other more advanced concepts. Third, the modules may be easily incorporated into classes that are taught online[6,7] as well; for instance, the hardware can be sent to students at remote locations while the pre-compiled or wholly/partially implemented modules can be downloaded from the web. Lastly, the associated cost of having the external boards or brick is small, being only on the order of several hundred dollars. However, in many cases where the available budget is insufficient for the external hardware, the PC card of the laptops/desktops can be utilized for capturing and processing the real-time data.

The paper proceeds as follows. Applicability of LabVIEW for the experimental DSP modules is given in the next section. Typical hardware and the software setups of the modules are articulated in Section 3. A list of modules built to date is explained in Section 4. Finally, Section 6 concludes the paper.

2. Applicability of LabVIEW for Experimental DSP Modules

In this section, we articulate the suitability of LabVIEW software and our approach in utilizing it for our low-cost experimental DSP modules. First and foremost, LabVIEW is a graphical programming language that has many icon-based built-in, ready-to-use functions and controls that can be easily extended for almost any purpose. LabVIEW also has controls and functions that facilitate the interface with external hardware platforms like DSP boards (e.g., Speedy-33), PC sound cards, and robotic platforms (e.g., NXT Brick). With minimal effort, users can fully utilize the software for many courses in undergraduate curriculum, starting from a basic signals and systems course to DSP courses. LabVIEW can even be used in graduate classes covering advanced signal processing. For these reasons, we have explicitly opted to utilize LabVIEW in our experimental DSP module designs.

Screen shots from a typical programming environment are shown in Figures 1 and 2. In the LabVIEW jargon the applications are referred to as Virtual Instruments (VIs). There are two important sub-components in a VI that are utilized when creating applications under LabVIEW.

One is called the Front Panel (FP), and the other is a Block Diagram (BD). These are tightly integrated with each other. [8]

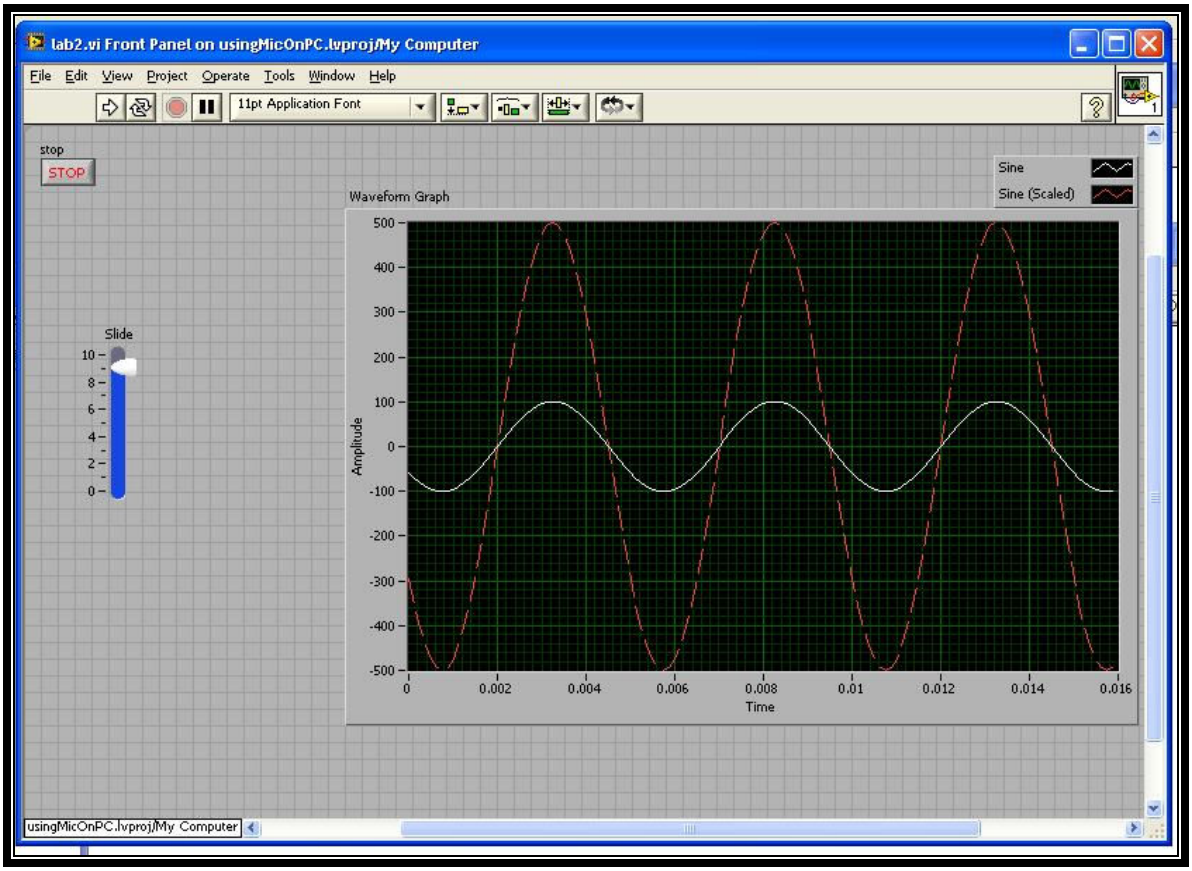

Figure 1: Front Panel on a sample Virtual Instrument

The developer defines the interactions of the application with the user or the outside world in the FP. In other words, the FP becomes the graphical user interface to the application. Albeit dependent on the specific application, the FP typically includes controls and indicators, which are the interactive input and output terminals of the VI. For instance, controls may consist of knobs, push buttons, dials, and other input mechanisms while indicators may consist of displays such as graphs, LEDs, and other output displays.

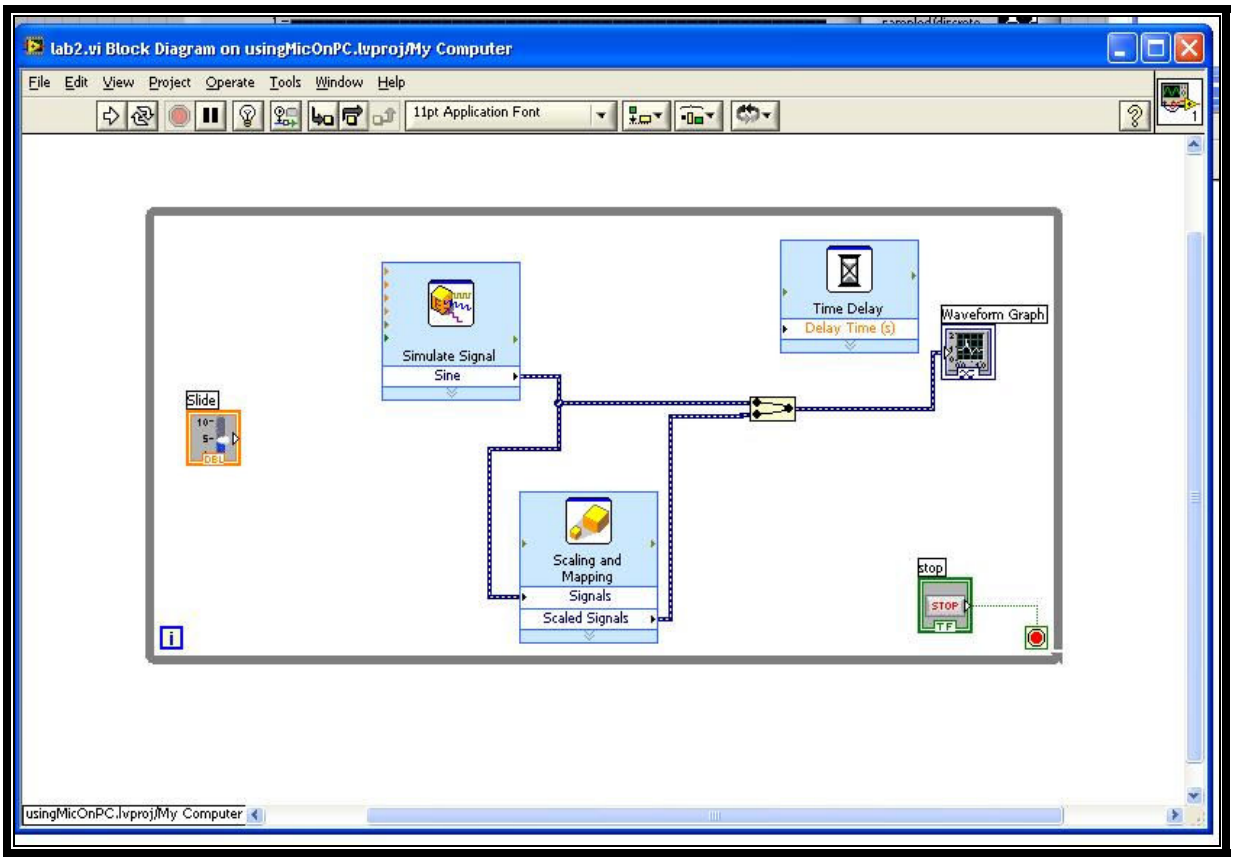

Figure 2: Block Diagram on a sample Virtual Instrument

On the other hand, the developer designs and implements the application's functionality inside the BD. The BD is a window where the developer inserts graphical code (i.e., icons) into the application. A BD integrates the FP into the actual flow of the program. BDs can be nested, and often these diagrams are composed of icons representing other BDs. There are many built-in ready-to-use graphical representations of functions that are themselves written as BDs.

Within the capabilities of LabVIEW, we have determined two ways of preparing and distributing the DSP learning modules as depicted in Figure 3.

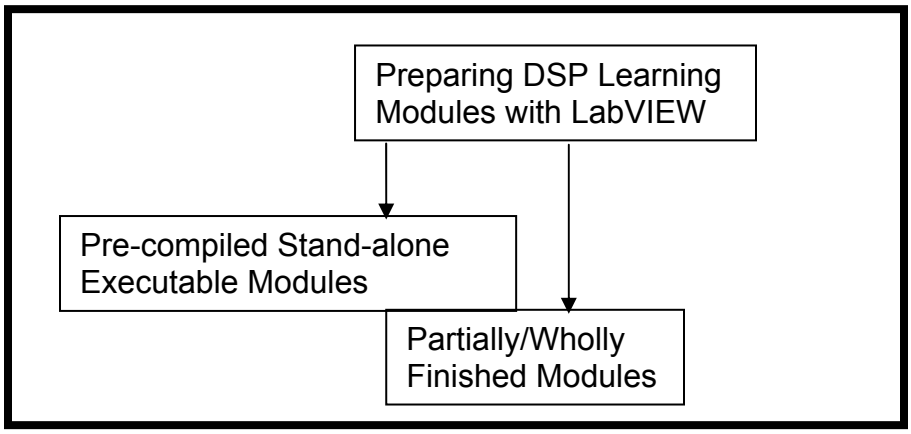

Figure 3: Preparing and distributing DSP Modules to students

One of the ways to prepare applications for students is to create *pre-compiled stand-alone executable modules* that can later be downloaded from the class web page or taken from the class instructor. Some of the modules that are developed under LabVIEW can be run as stand-alone executable files. For these programs students would not need to have their own copies of LabVIEW, but, unless the students are given the module source code, they could only interact with the executables. Overall, this may be a good option because students would not need to learn the details of the programming language. However, the students would also not be able to observe how the concepts have been implemented. The LabVIEW programming environment provides an embedded software tool called the *application builder* to create such stand-alone executable applications (e.g., EXE type applications). The user does not need to install or have a copy of the LabVIEW software to run these exe files. The only necessary component is the LabVIEW Run-time Engine,[9] which can be downloaded from the NI web site freely. Stand-alone applications can run on the host PC or on the external hardware device. Nonetheless, this feature is not fully supported by all the external hardware platforms. For instance, when using the Speedy-33, the LabVIEW software is still needed for some interactive modules where students are expected to draw plots of sound waves captured via the analog input.

The students could be also be provided with all or part of the implementation, which we refer to as *Partially/Wholly Finished Modules*. Most of the applications that play with the real-time data captured via the external hardware platforms can be distributed to the students with this way. The students are expected to interact more with the module with this approach. If the module is partially implemented, students can complete the module as an assignment. Alternatively, if the module is already finished, students may have the chance of reviewing and modifying the actual implementation. Thus, they may learn and grasp some of the implementation techniques and gain hands-on experience as well.

We further identify two methods of creating real-time experimental modules using LabVIEW. Both stand-alone and partially/wholly finished modules can involve real-time modules.

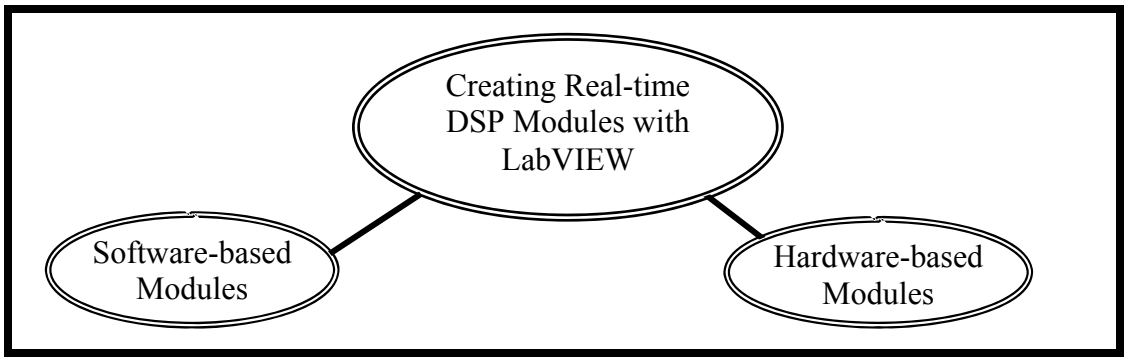

Figure 4: Creating real-time DSP Modules

One is acquiring the real-time data via external/internal hardware platforms. This approach constitutes our *hardware-based modules* (or applications). For this case, DSP boards (e.g., Speedy-33, TI-C6713), sound cards in the PCs, or the NXT brick of Lego Mindstorms can be utilized. The executables of these types of stand-alone applications can be hosted either by the PC or the external device.

The second method is to simulate the data acquiring process with computer generated data, which constitutes the *software-based modules*. This approach is basically simulation of the real world with the already present functions under LabVIEW on the host PC platform. LabVIEW contains built-in functions for operations such as signal creation to simplify this process.

Nonetheless, as mentioned earlier, in this work, we keep the focus only on the hardware-based experimental modules. Throughout this work, three different hardware platforms have been explored in the sample modules created. Two of them are completely external to the PC: NI's Speedy-33 DSP Board and Lego Mindstorm's NXT Brick. The third is the sound card, which is found on the host laptop. Out of these three platforms, we have given an emphasis on the use of the Speedy-33 external DSP board.

3. Typical Setup of Experimental Modules

This section gives the specifics of the setup of the experimental modules. The available hardware and software components for the experimental modules include one laptop, one external DSP Board, which is NI-Speedy-33, one LEGO NXT Brick, one webcam, and the sound card on the laptop. As has been specified earlier, the emphasis was on utilizing the DSP board whenever and wherever possible. Figure 5 show the different setups.

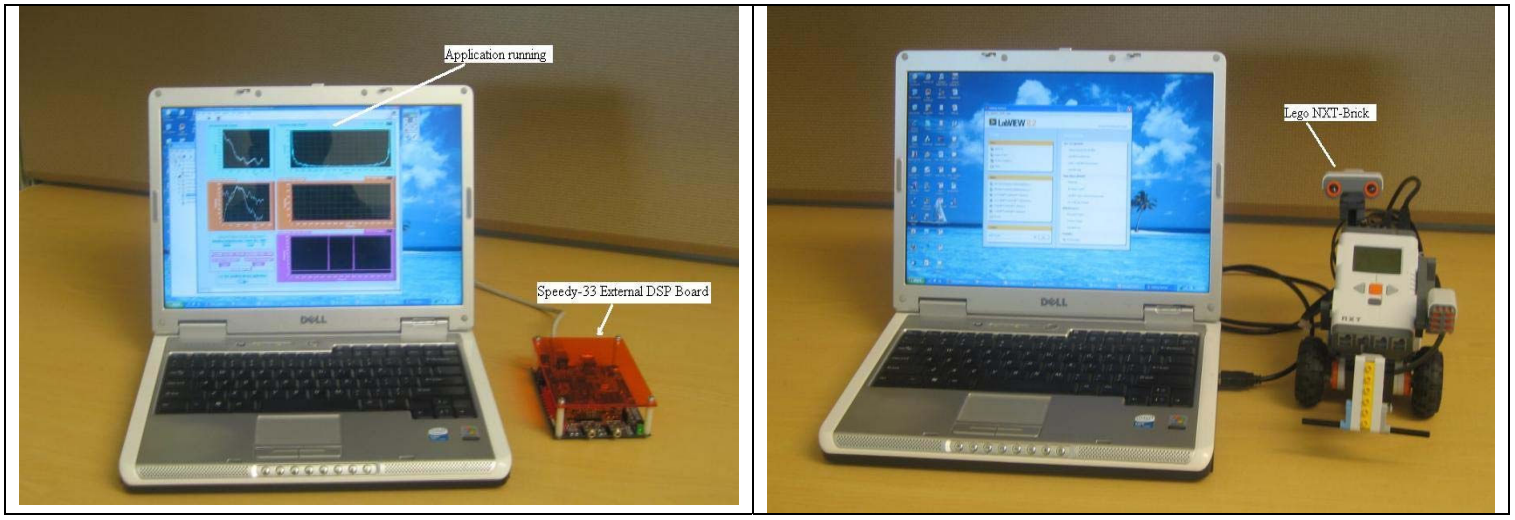

Figure 5: On the left, the NI Speedy-33 DSP board and laptop; on the right, the LEGO Mindstorms NXT Brick and laptop

We have primarily used the NI-Speedy-33 for the basis of our modules. As seen on the left part of the figure, the Speedy-33 DSP board is connected to the PC or laptop via the USB port. Specifically, the Speedy-33 is a high-performance floating-point TMS320VC33-based DSP system,[11] having an on-chip RAM with a size of 34 K × 32 words and 512 K × 8 onboard flash memory. The board features two input/output analog channels that support the sampling rates of 8 kHz, 18 kHz, 24 kHz, 36 kHz, and 48 kHz, and onboard 8 digital I/O lines for controlling motors/servos. The onboard flash memory allows storing of application programs, data such as tables, and sound waveforms. The flash memory can also be programmed to run applications in stand-alone mode when unplugged from the host system, PC, or laptop.

4. List of Experimental DSP Learning Modules Built with NI-Speedy-33

A variety of simple or complicated hardware and software based modules could be built with LabVIEW. In this section, we explain which modules we have built using LabVIEW 8.2 and its tool DSP Module 2. Specifically, we have developed 13 NI-Speedy-33 modules, 2 NXT modules, and 4 software-based modules. We elaborate more on the hardware-based ones utilizing the NI-Speedy-33 to reflect the real-time examples as much as possible. However, most aspects of signal processing and DSP classes could be implemented with software-based modules, but they would be based on non-real-time data.

When building hardware-based modules, some of the signal processing concepts such as signal generation, FFT, convolution, and filtering can be implemented directly by the students with the functions and tools provided in LabVIEW for the Speedy-33. After students learn to implement these fundamental concepts, assignments can be given to them where they are expected to build relatively more complicated programs combining several of these basic ones. The following is the list of such hardware-based modules we have built using the NI-Speedy-33, which are

basically the implementations of widely used or widely studied general DSP concepts or sample applications in textbooks. Please note that the explanations or limitations related to LabVIEW functions or tools mentioned below only refer to the particular cases where the NI-Speedy-33 is used, not to LabVIEW in general.

Sampling/Aliasing: The goal of this module is to allow students to examine the sampling theorem and the aliasing effect. The external board samples a sound source using its analog input. The sound source is another module, consisting of a software-based stand-alone executable module that creates the sound waves at given frequencies. In the sampling/aliasing module, both time and frequency domain representations of the captured signals are plotted on graphs. Additionally, students are expected to enter their guesses for the main frequency components of the input signal. The NI-Speedy-33 was a better choice than the sound card for this module as the sound card has a built-in anti-aliasing filter and would not allow student to observe the aliasing effect on the input signal. This module is shown on screen-shots in Figure 6 and 7. Figure 6 shows the front panel, while Figure 7 shows the block diagram.

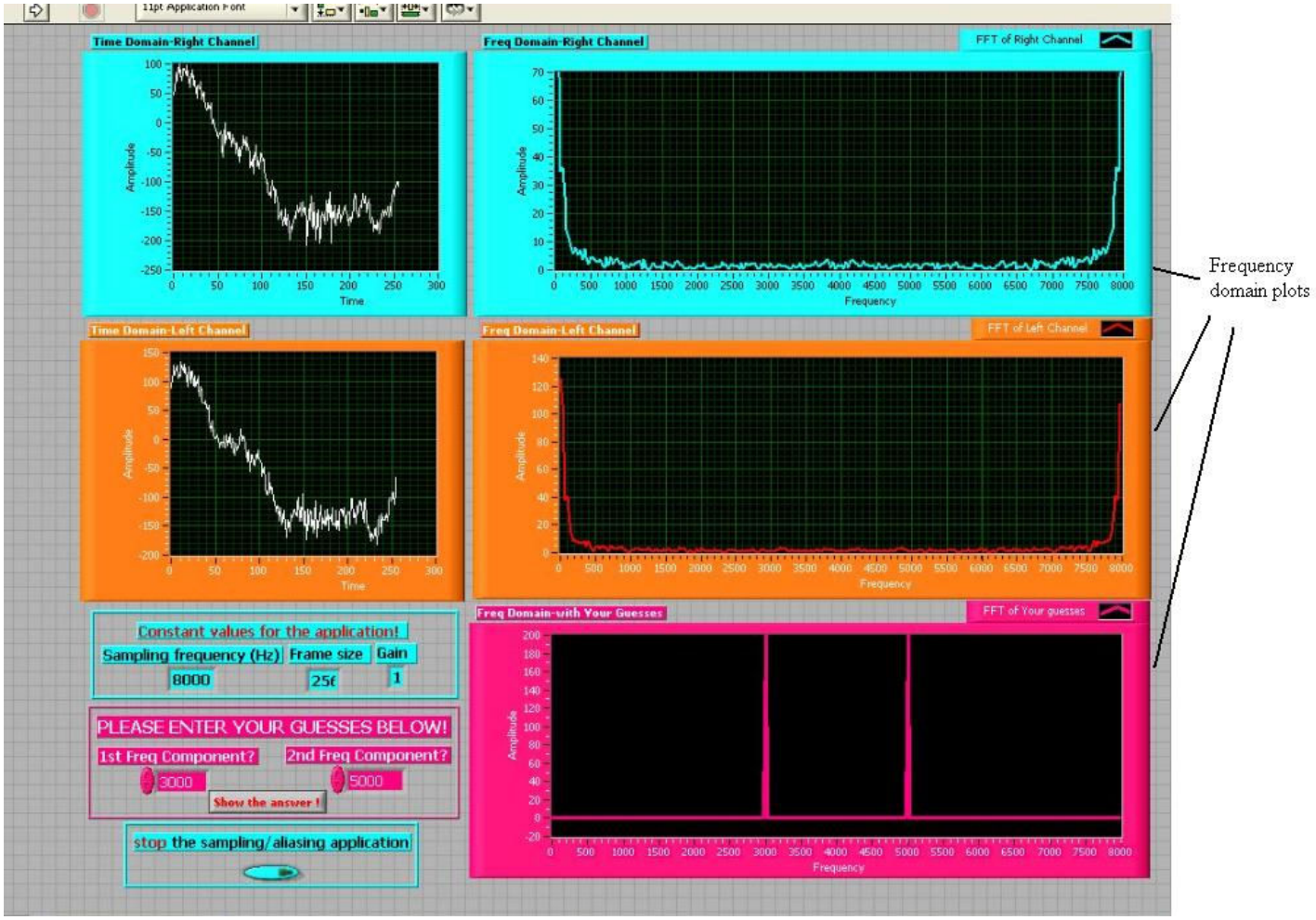

Figure 6: Front panel of the Sampling/Aliasing module

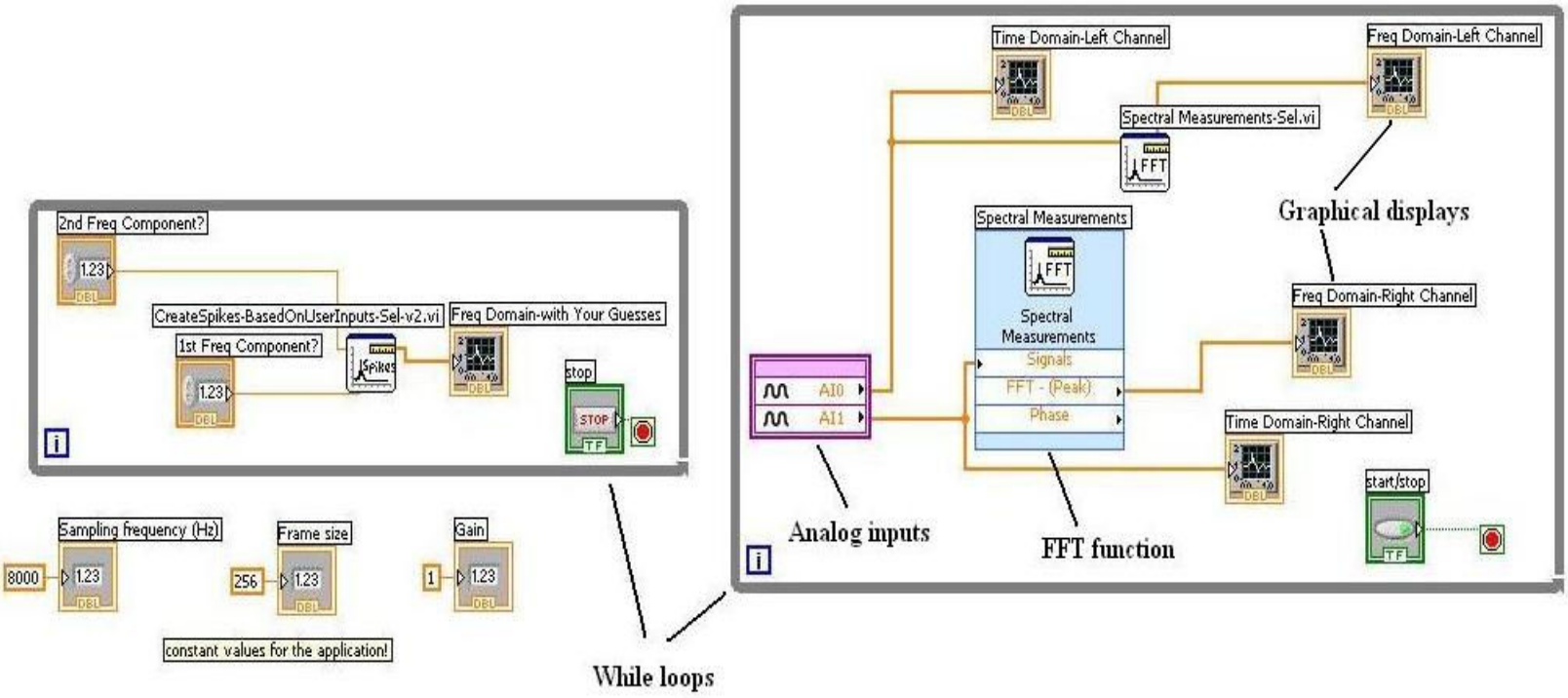

Figure 7: Block diagram of the Sampling/Aliasing module

Frequency Response of FIR Filters: The LabVIEW DSP Module 2.0 for the NISpeedy- 33 does not provide functions or tools to directly show the frequency response of the filters. We have developed two different modules for this purpose. One implements the Dirichlet function to calculate the response of an averaging filter, whereas the other computes the response of linear-phase FIR filters. In this module students are able to provide the length of the filter as an input parameter. The resultant frequency response and the filter output are both plotted in the module.

Removing Zeros from an FIR Filter: In this module, we have extended the previous module and provided the student with the necessary interactions to remove zeros from the filter transfer function. First, the students are able to see both the frequency response and the transfer function of the filter and then, based on their input to the module, some of the zeros are removed. Finally, the resultant frequency response is plotted.

Cascaded Filters: The LabVIEW built-in filtering function for Speedy-33 only implements lowpass or highpass filters. In order to implement bandpass or a bandstop filters in a module, cascading of more basic filtering functions is needed. Thus, in this module, we have developed a bandpass filter by cascading a high and low pass filter for a signal generated with the embedded sine generator function. The student is able to provide the frequency and the amplitude of the signal to be generated, and observe what happens when the signal is not within the limits of the filter.

Removing a Sinusoidal Interference: This is another module where filtering is used. In this module, while an auxiliary software-based stand-alone module plays a noisy sound file, the main

module filters the noise out with a bandstop IIR filter. After removing the noise from the main wave file, the main module plays the output signal through the board's analog output channels. With the help of a speaker connected to the output, the student is able to listen to either the original or the filtered sound.

7-band Equalizer: Utilizing the previously built cascaded filters module, an equalizer module can easily be developed. Thus, in this module we have provided a 7- band equalizer with associated student interactions. The students are able choose which frequency components to increase or decrease.

Additionally, based on the capabilities of the NI-Speedy-33 and our experience with it, more DSP applications involving Echo, Reverberation, Phone tone generating, AM Transmitter, AM Receiver, FSK Modulator/Demodulator are also being implemented.

5. Discussion of Hardware-based DSP Learning Modules vs. Other Comparable Labs

In this section, we shed light on how our proposed method of hardware-based low-cost experimental DSP modules compares to other, comparable labs. To date, the usual experiential approach for teaching these concepts is largely based on the use of MATLAB and Simulink[11,12,13,14,15]. Although there are other recent efforts to teach DSP concepts that lie outside of these cases being considered, these approaches that use other high-level languages are still an avenue for further development and research[15,16,17,18]. Thus, in this section we limit our comparisons only to other labs utilizing MATLAB. We show that the proposed method of teaching DSP concepts with hardware-based learning modules in this paper can be compared in several ways with tools based on MATLAB or Simulink.

First and foremost, the hardware-based DSP learning modules work with data that is captured or processed in real-time. Moreover, LabVIEW, since its inception, has been designed with external hardware platforms in mind and there are several platforms currently available that can interface with LabVIEW, including the NI-Speedy-33, TI-C6713 and LEGO Mindstorms' NXT Robot. The number of different platforms that can be used with other software is increasing, as well. However, the ability for real-time data processing and interfacing with all these different external hardware platforms is limited in MATLAB.

The construction and design of hardware-based DSP modules are accomplished here via utilization of the graphical programming language LabVIEW. The LabVIEW environment provides a rich set of ready-to-use, built-in functions and tools, whereby users can easily implement both fundamental and advanced DSP concepts by clicking and inserting these functions into the program. This style of programming does not necessitate acquiring a great deal of prior conventional programming experience and is a better match to the block diagram descriptions commonly used by engineers. However with MATLAB, even though it is also a powerful high-level programming language, the users or students need to learn conventional programming techniques and styles to a certain level of expertise in order to simulate the same DSP labs.

Finally, LabVIEW allows the development of some labs or learning modules as stand-alone executable labs. The modules that are developed as stand-alone executables can be run with the LabVIEW Run-time Engine, which can be downloaded freely from NI's web site. This way, the students would not require a copy of the LabVIEW software. However, with MATLAB labs and GUI-based demos a copy of MATLAB is necessary.

6. Acknowledgements

This work was supported by the National Science Foundation's Course, Curriculum, and Laboratory Improvement Program under contract number DUE-0618645.

7. Conclusions

In this paper, we have discussed the development of experimental DSP learning modules. These experimental DSP modules target the facilitation of teaching and learning fundamental DSP concepts such as sampling, aliasing, FFT, and filtering, which are often considered by the students as relatively difficult concepts to grasp and understand.

As opposed to other efforts throughout academia, the modules described here are based on using real-time captured data via an external DSP board. Students are able to interact with this external hardware and play with the captured data through GUIs. Instead of text-based programming languages like MATLAB, the modules have been implemented using a graphical programming language that has built-in functions and controls to access the external hardware and obviates the need to develop real-time programming skills. Specifically, we have used NI's LabVIEW, which is a graphical programming language, as the software platform and the NI-Speedy-33 DSP Board as the main hardware platform for our experimental modules.

The combined use of external platforms (e.g., the Speedy-33 DSP board and LEGO Mindstorms NXT Brick[19]) and a graphical programming language is trivial under LabVIEW and does not require a great deal of sophistication. The associated cost for acquiring the hardware is on the order of several hundred dollars. Even if the budget is insufficient for this hardware, the modules can be implemented as software modules based on non-real-time data generated by the program. The graphical programming language eases the implementation of real-time DSP algorithms. The flexibility of the programming language also allows for the easy extension of the modules beyond the fundamental DSP concepts in the undergraduate curriculum. These learning modules can easily be incorporated into classes that have been primarily or entirely lecture-based. For instance, the modules and associated boards may be checked out by the students for after-class hours or may even be sent to remote locations for online classes.

With today's technology, we already see several institutions and projects[11, 13, 14, 15, 17, 18] that aim to teach DSP concepts easily with the help of either high-level programming languages (e.g., Java) or external DSP boards. The primary goal of these initiatives is to fill the gap between the theory and the hands-on experience. We believe that future improvements in technology will further enable the entry of external hardware platforms (e.g., external DSP boards, Robots, etc.)

into the classes that do not traditionally have labs or teach real-time concepts in the undergraduate ECE curriculum.

Our future work will include implementations of more experimental DSP learning modules. Completed modules will be tested in our undergraduate classes, and their effectiveness and performance will be measured, including the students' and instructors' feedbacks.

Bibliography

1. NI SPEEDY-33 User Manual, http://digital.ni.com/manuals.nsf/websearch/104061C41B6A2362862570460052AEE9
2. Digital Signal Processing (DSP) Development Tools, Texas Instruments http://focus.ti.com/dsp/docs/dsphome.tsp?sectionId=46
3. LEGO.com MINDSTORMS NXT Home http://mindstorms.lego.com/
4. http://www.ni.com/labview/
5. NI Acquires Digital Signal Processing Software Developer Hyperception, http://zone.ni.com/devzone/cda/tut/p/id/3667
6. Jackson, J., Barnwell, T., Williams, D., Anderson, D., and Schafer, R., "DSP for Practicing Engineers: An online course for continuing DSP education," In *Proceedings of the IEEE International Conference Acoustics, Speech, and Signal Processing* (ICASSP)*, 2001.*
7. P.S. Hong, D.V. Anderson, D.B. Williams, J.R. Jackson, T.P. Barnwell, M.H. Hayes, R.W. Schafer, and J.D. Echard, "DSP for Practicing Engineers: A case study in internet course delivery," *IEEE Trans. on Education*, vol. 47, no. 3, pp. 301-310, August 2004.
8. LabVIEW User Manual, http://www.ni.com/pdf/manuals/320999b.pdf
9. LabVIEW Run-time Engine, http://zone.ni.com/devzone/cda/epd/p/id/4850
10. NI Speedy-33 User Manual, http://digital.ni.com/manuals.nsf/websearch/104061C41B6A2362862570460052AEE9
11. Sharon Gannot and Vadim Avrin, "A Simulink© and Texas Instruments C6713® based Digital Signal Processing Laboratory", The European Association for Signal Processing (Eusipco) 2006, Florence Italy, Sep. 2006.
12. Chiang, K.H; Evans, B.L.; Huang, W.T.; Kovac, F.; Lee, E.A.; Messerschmitt, D.G.; Reekie, H.J.; Sastry, S.S.; "Real-time DSP for sophomores"**,** ICASSP-96. Conference Proceedings., Volume 2, 7-10 May 1996 Page(s):1097 - 1100 vol. 2
13. Wright, C. H. and Welch, T. B., "Teaching DSP concepts using MATLAB and the TMS320C31 DSK," *Proceedings of the IEEE International Conference Acoustics, Speech, and Signal Processing* (ICASSP), 1999
14. Lisa G. Huettel, "A DSP Hardware-Based Laboratory for Signals and Systems,**"** *12th Signal Processing Education Workshop*, 4th Volume, Issue, Sept. 2006 Page(s):456 - 459
15. Lisa G. Huettel, "Integration of a DSP Hardware-Based Laboratory into an Introductory Signals and Systems Course," *Proceedings of the American Society for Engineering Education (ASEE), Annual Conference of Composition and Exhibition*, 2006
16. Asif, A. "Multimedia learning objects for digital signal processing in communications," *Proceedings of the 2003 International Conference on Multimedia and Expo - Volume 1* (July 06 - 09, 2003), 157-130.
17. Spanias, A. Berisha, V. Ho Min Kwon, Chih-Wei Huang, Natarajan A., Ferzli, R., "Using the Java-DSP Real-Time Hardware Interface in Undergraduate Classes," Proc. of the *36th Annual Frontiers in Education Conference***,** 2006
18. Yoder, M.A. Black, B.A., Work in Progress: A Study of Graphical vs. Textual Programming for DSP, Proc. of The *36th Annual Frontiers in Education Conference,* 2006
19. R. Butera, M. Clark, A. Deck, M. Torba, S. Trahan, S. Uluagac, D. Williams, "Cooperative University/Industry Development of a Freshman 'Introduction To ECE Design' Course", *Proceedings of the American Society for Engineering Education (ASEE), Annual Conference of Composition and Exhibition*, 2007