An Introductory Virtual Laboratory for Electrical Engineering

Erwin D'Souza[1] and Mehmet C. Öztürk[2]

North Carolina State University
[1]Department of Computer Science
[2]Department of Electrical and Computer Engineering
Raleigh, NC 27695

Introduction

Educational Java applets are gaining popularity as the number of applets available on the World Wide Web continues to grow. An important advantage of applets is that they can be run on Java enabled internet browsers without the need for storing the actual program in the user's computer. Hence, in educational institutions, an applet stored on a web server can be shared by all the students in a class allowing them to run the applet on their home computers in their own time. When large numbers of students are involved, it is also convenient for an instructor because the software has to be installed in a single computer on the network providing easy maintenance and upgrades.

An excellent site for finding educational applets is the National Science Digital Library (NSDL), which is a software depository sponsored by National Science Foundation. Some schools are also hosting their own sites such as the widely popular *Signals, Systems and Control Demonstrations* for electrical and computer engineering (ECE) students by John Hopkins University[1].

A number of publications are available on the innovative uses of educational software in electrical and computer engineering education[2,3,4]. In general, educational programs are written as self-study tools to help the students. Some of these programs are also suitable for interactive homework assignments. They are especially powerful in introductory classes in which students are introduced to new abstract concepts requiring countless examples to digest the material. Finally, simulation programs can help the instructors in developing effective demonstrations thanks to the widespread availability of projectors in modern class-rooms.

This paper presents an introductory virtual laboratory, which consists of a series of new Java applets suitable for sophomore and junior level ECE classes. The software was designed with the primary objective of creating a tool to enhance the delivery of fundamental concepts in a new sophomore level course and its hardware laboratory at North Carolina State University (NCSU). A detailed description of the course and the laboratory can be found in a previous paper presented at this conference[5]. The new course sharply differs from traditional sophomore level ECE courses on electrical circuits. Its philosophy can be summarized as follows: introduce different ECE specializations and use this medium to teach fundamental concepts with motivating examples. The course assumes no prior background in electrical and computer engineering concepts, but it requires successful completion of the first year classes common to all engineering students, which includes introductory physics and calculus. In the laboratory, students learn how to use the standard bench-top test instruments consisting of the power supply, function generator, oscilloscope and spectrum analyzer. The experiments are based on dedicated laboratory hardware designed to enable the beginning students to experiment with fairly complex but motivating systems based on real-life applications such as music amplification, transmission and reception of radio signals, sampling and reconstruction of analog signals. The new laboratory hardware developed for this course is now available for purchase from Tricounty Industries, a non-profit organization based in Rocky Mount, North Carolina.

After offering the course at NCSU to approximately two thousand sophomores in two consecutive years, we were able to conclude that the new course/laboratory concept proved to be successful. The students were highly motivated by the experiments and it was clear that the experiments greatly helped learning of the material introduced in the lectures. Nevertheless, we felt that some improvements were still necessary to establish a stronger link between the experiments and the course material. To assess our success in several key areas, carefully planned student surveys were conducted and specific assessment questions were included in the exams. First and foremost, these studies indicated that spending three hours in the laboratory every week was not sufficient for the beginning students to master the modern bench-top instruments. We saw that students' struggles with the test equipment (especially during the first half of the semester) could result in frustration in the laboratory creating an artificial barrier in performing the experiments and learning the fundamental concepts. Extra time on the equipment could certainly help; however, limited resources available for laboratory instructors did not allow us to keep the laboratory open outside the regularly scheduled laboratory sessions. The problem became worse when one of the students in a laboratory group assumed the primary operator role, while the others remained relatively passive. A passive student could still record all the measurement results to include in the report and receive good grades in spite of his/her lack of active involvement in the laboratory. Finally, a student did not get the full benefit of running an experiment when he/she came to the laboratory without reviewing the background material properly. Assigning a few homework problems related to the next experiment helped but it often did not guarantee satisfactory preparation for the experiment.

We believe that similar concerns are also shared by other institutions offering introductory laboratories and they are expected to be amplified even further when large enrollment numbers, limited laboratory facilities and long list of subjects to cover during a four year ECE curriculum force the institutions to reduce the hours dedicated to traditional hardware laboratories.

The virtual laboratory concept presented in this paper emerged as a potential solution to address the above challenges. The original idea was simple: create a software tool that simulates the operation of standard bench-top equipment and use the homework assignments to revisit and reinforce some of the key experimental concepts from previous week's experiment. Similar attempts to simulate bench-top equipment can be found in the literature[6,7,8]. The virtual laboratory presented in this paper differs from these tools because it combines simulations of the test equipment with a system under test (SUT), which can be a simple RC circuit, an amplifier or an AM modulator. As such, using the virtual laboratory, the students can practice using the test instruments, but they can also make measurements similar to those carried out in the hardware laboratory. The difference is that in the virtual laboratory all the connections are in place, there are no broken wires or blown fuses or noise on the signals. While these may indeed be positive attributes, the authors are of the opinion that the virtual laboratory cannot be a replacement for a hardware laboratory and its primary objective is to provide support to the new course/laboratory described in previous paragraphs.

Virtual Test Instruments

The Virtual Laboratory is a set of Java applets, each applet corresponding to a different experiment on topics such as simple circuits with resistors and capacitors, amplifiers, filters and multipliers, which emphasize interpretation of signals in both time and frequency domains. The experiments are centered around three *virtual test instruments* that simulate the operation of standard bench-top equipment including the function generator, oscilloscope and spectrum analyzer. The features and controls on the virtual instruments closely resemble those available on the actual equipment. In the following paragraphs, we present the basic features of the virtual test instruments.

Function Generator:

The function generator shown in Figure 1 is the primary instrument for generating the signals. It can produce three different waveforms: sinusoidal, square wave and random. The user can switch between these signals by clicking on one of the waveform buttons on the left. For each signal, the user can set the frequency, peak-to-peak and average value using the rotary knobs. In the square wave generator, it is also possible to set the duty cycle. The random signal is constructed by adding ten sinusoids with random phase angles and amplitudes. Each click on the random signal button generates a new waveform.

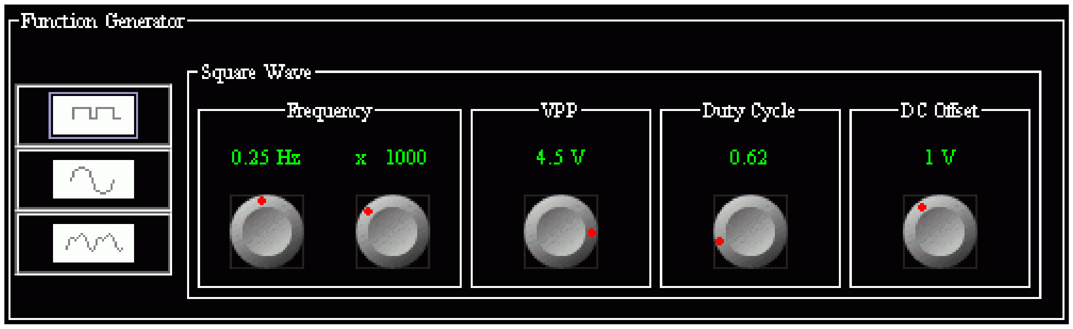


Figure 1 - Function Generator

Oscilloscope

The oscilloscope shown in Figure 2 is used to display the waveforms in the time domain. It is a two-channel oscilloscope, with all the basic controls of an oscilloscope except triggering. The user can turn off one of the channels, choose AC or DC coupling, set the vertical and horizontal scales and move the signals up and down. A digital meter also allows the user to measure the frequency, average value (V_{DC}) and root-mean-square value (V_{RMS}) of the signal displayed on a channel.

Spectrum Analyzer

The spectrum analyzer shown in Figure 3 is used to display the signals in frequency domain. The user may choose the signal channel to display, adjust the start frequency, horizontal and vertical scales by turning rotary knobs. The analyzer can display both amplitude and power spectrum. The vertical axis for the power spectrum can be signal power in Watts or decibel-Watts (dBW). Since the signals are known, the code computes the Fourier series coefficients instead of employing a Fast Fourier Transform (FFT) routine. This provides fast and accurate display of the spectra while losing the ability of displaying the spectra of unknown signals.

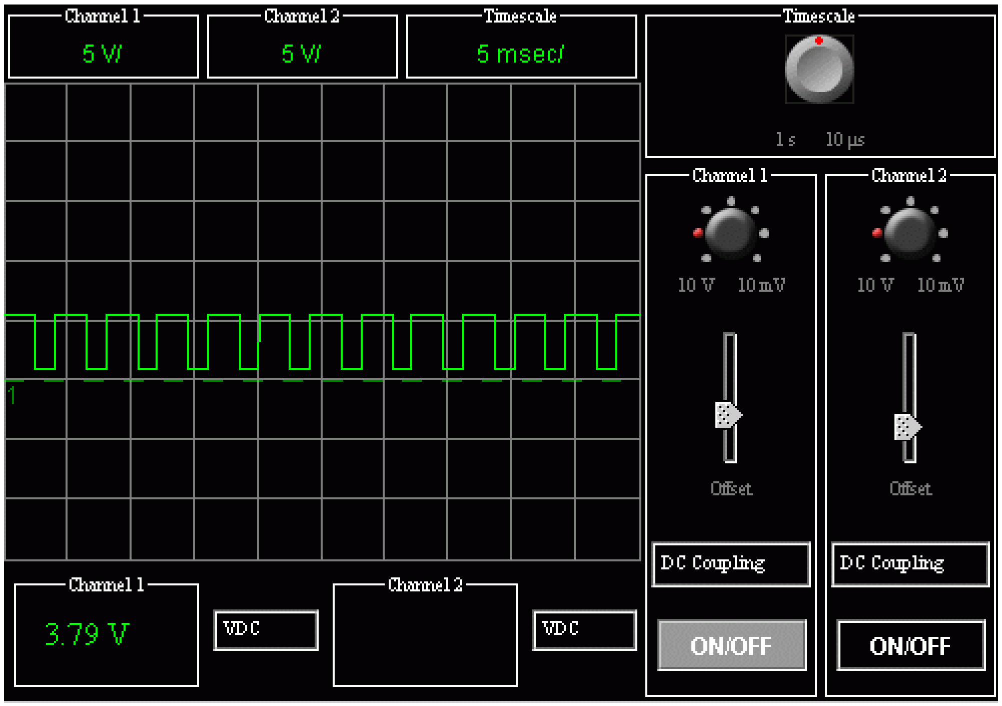
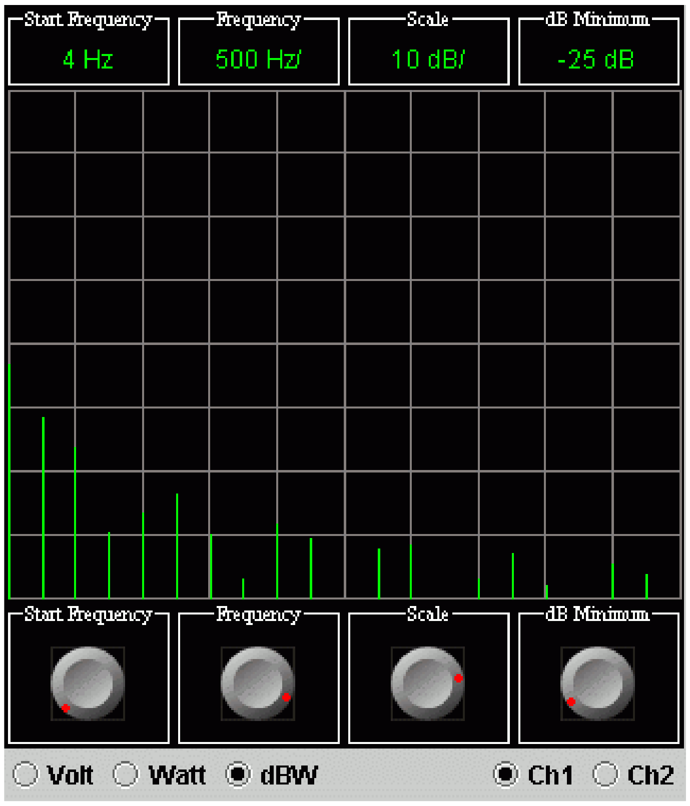

Channel 1: 5 V/
Channel 2: 5 V/
Timescale: 5 msec/
Timescale: 1 s 10 µs
Channel 1: 10 V 10 mV Offset DC Coupling ON/OFF
Channel 2: 10 V 10 mV Offset DC Coupling ON/OFF
Channel 1: 3.79 V VDC
Channel 2: VDC

Figure 2 - Oscilloscope

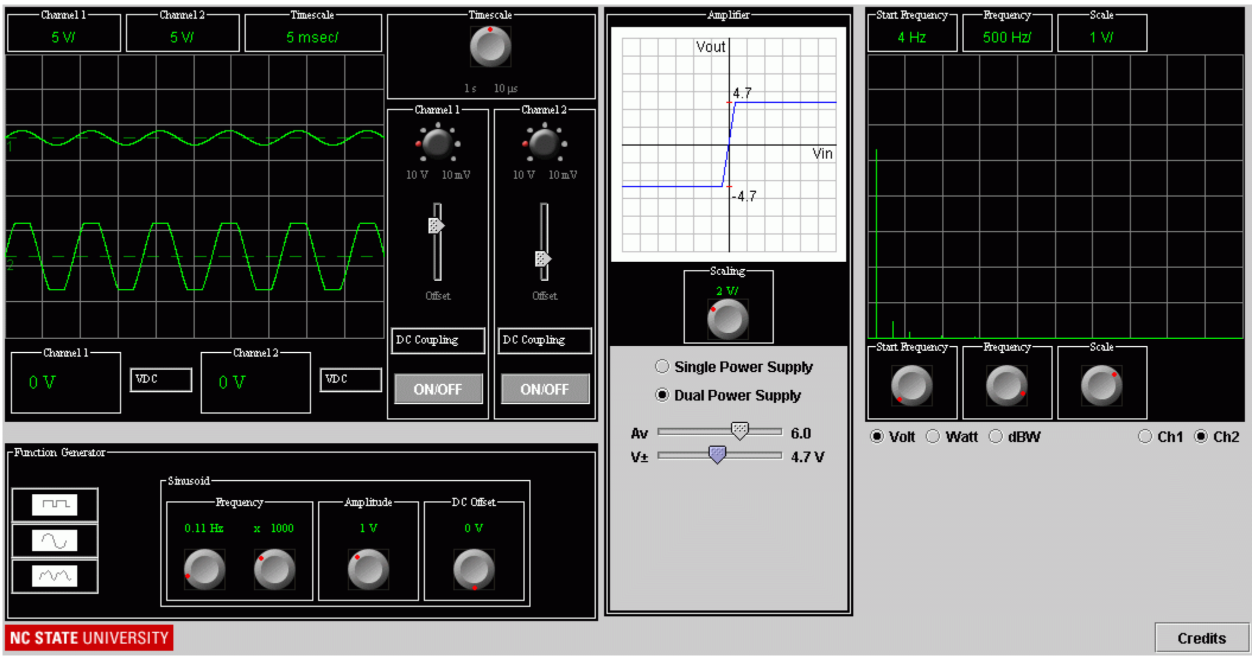

Start Frequency: 4 Hz
Frequency: 500 Hz/
Scale: 10 dB/
dB Minimum: -25 dB

Volt Watt [✓ dBW] [✓ Ch1] Ch2

Figure 3 - Spectrum Analyzer

Examples of Virtual Laboratory Experiments

To illustrate the operation of the virtual laboratory, we shall use the amplifier applet shown in Figure 4 as our example. This particular applet employs all three test instruments introduced in the previous section. The amplifier section (displayed in the center of Figure 4) allows the user to choose between a single or dual power supply, change the power supply voltage and the voltage gain. The transfer characteristic (V_{out} versus V_{in}) of the amplifier is also plotted for reference. As soon as the user picks an input signal and sets the waveform parameters, the oscilloscope can display the input and output signals simultaneously. However, this operation requires that the user is familiar with the oscilloscope basics including setting the vertical and horizontal scales, AC/DC coupling and vertical positioning. An "autoscale" button has been omitted from the oscilloscope on purpose to enforce this learning. The spectrum analyzer is used to monitor either one of the displayed signals in frequency domain. In Figure 4, a 1100 Hz sinusoidal waveform is applied to the input-port of the amplifier. We note that the output waveform is clipped because with the chosen voltage gain of 6 the output signal exceeds the maximum possible range of +/- 4.7 V set by the power supply. The clipping can also be observed on the spectrum analyzer as additional harmonics appear at multiples of 1100 Hz.



Channel 1: 5 V/
Channel 2: 5 V/
Timescale: 5 msec/
Channel 1: 0 V VDC
Channel 2: 0 V VDC
Channel 1: DC Coupling
Channel 2: DC Coupling
Scaling: 2 V/
[✓ Dual Power Supply]
Av 6.0
V± 4.7 V
Sinusoid: Frequency 0.11 Hz x 1000; Amplitude 1 V; DC Offset 0 V
Start Frequency: 4 Hz
Frequency: 500 Hz/
Scale: 1 V/
[✓ Volt] Watt dBW
Ch1 [✓ Ch2]

Figure 4 Amplifier applet.

Figure 5 shows a homework problem based on the operational amplifier (op-amp) applet, which is very similar to the amplifier applet described in the previous example. In this case, instead of setting the amplifier gain, the user is first asked to choose one of the four op-amp circuits. The middle section provides slider potentiometers to set the values of the resistors in the amplifier circuit. As the user changes the resistors or the power supply voltage, the amplifier output can be observed in both time and frequency domains. The homework problem focuses on distortion due to output clipping and the user is asked to calculate the noise power using the spectrum display. The steps followed in making the measurements are almost identical to those followed in the hardware laboratory. The virtual lab provides the student a second chance to go through the same procedure and focus on the concepts missed in the laboratory perhaps due to the pressure of finishing the experiment in a given time period. Because the student already ran a similar experiment with real hardware during the previous week, the virtual experiment is believable.

Homework Problem:

Go to the virtual lab at the following URL:
http://www.ece.ncsu.edu/virtuallab/JAVA/applets/opamp.html

Use the following parameters:

Operational Amplifier Circuit:
V_{supply} = +/- 6 V, R_f = 80 kΩ, R_n = 10 kΩ, R_g = 25 kΩ, R_p = 50 kΩ

Input Signal:
f = 200 Hz, VDC = 1 V, V_{p-p} = 500 mV, Type = Random

1. Measure the DC value of the output signal as you change the DC Value of the input signal from 0 to 1 V. Take five measurements and plot V_{DC}(in) versus V_{DC}(out).
2. Use the virtual-short concept to derive an equation for the output voltage and use this equation to verify your measurement result.
3. Set the DC value of the input signal to 0 V. Try different random signals (each click on the button generates a new signal) with different peak-to-peak values to experimentally determine the peak-to-peak value of the largest input signal the amplifier can amplify without clipping.
4. Use the transfer characteristic of the amplifier and the equation derived in part (2) to verify your measurement result.
5. Set the DC value of the input signal to 1 V. Repeat parts (3) and (4).
6. Change the input signal to a sinusoid with f = 200 Hz, V_{DC} = 0 V, Vp-p = 6 V. The output signal should be clipped. Use the spectrum analyzer to compute the signal-to-noise ratio of the output signal in dB.
7. Without changing the amplitude of the input signal, determine how you adjust the value of R_f such that a signal with V_{p-p} = 6 V can be amplified without clipping.

Figure 5 An example homework problem using the operational amplifier applet.

Software Implementation

Although each applet is different in functionality, the code and libraries shared are substantial, and only a small portion of the code is actually different. Hence, the Virtual Lab can be considered as a single application package and each enclosed applet simply a chosen view or facade to this package. Java was chosen as the language due to the following reasons:

1. Availability of support for Java applets and platform independence
2. Object oriented methodology
3. Support for multi-threading
4. Convenient exception handling

Our design goals were:

1. Reusability: To be able to reuse existing components easily and without modification, thereby saving development time and effort.

2. Modifiability: To be able to easily locate and modify the functionality or value of a desired component. To be able to make the required changes with minimal rework.
3. Decoupling (Abstraction): To be able to design and code a component independent of the components it may interact with.
4. Generality: To be able to use a component in a wide variety of scenarios without any modification.
5. Efficiency: To have an implementation that displays signals reasonably fast on an average computer and responds quickly to the user on any computer.
6. Compactness: To minimize the size of the project executable code in order to facilitate fast downloading.

The individual interfaces of the virtual lab are homogeneous, thus facilitating effortless addition and interconnection of the virtual test instruments. Furthermore, as is the case with real life instruments, every virtual instrument is complete in itself with a defined behavior that does not depend on the other instruments it may connect to. Hence, to create a new applet, all that is required from the programmer is to write the code for the SUT and connect the virtual instruments.

An applet begins by initializing the required instruments, positioning them on the screen and connecting them as desired. Each instrument may have signals incoming, outgoing or both. The signal flow in the amplifier example is shown in Figure 6. In this applet, the function generator generates a signal, which is received by the amplifier and by the first oscilloscope channel for display. The amplifier generates, based on this input signal, a new amplified signal for output. This output signal is received by the second oscilloscope channel, which displays it on the screen. The spectrum analyzer can accept signals from either one of the oscilloscope channels for displaying the signal in the frequency domain.

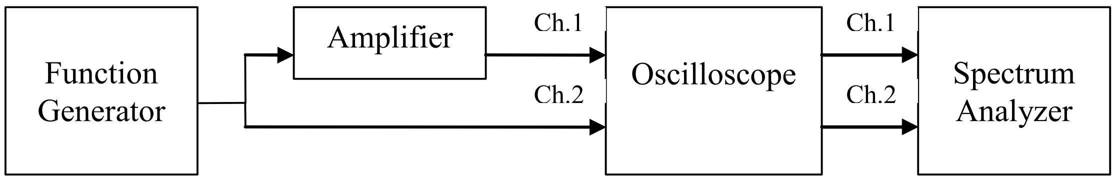


Figure 6 Signal flow in the amplifier applet

Whenever the user modifies one of the parameters of the signal or the amplifier, the instrument needs to generate a new output signal based on the new set of property values. These property values, in conjunction with the required input signals (if any) can completely define the output signal. By design, a change in a virtual instrument's properties will affect only that instrument and all *downstream* instruments. The instruments further *upstream* are not aware of the change in this instrument's properties.

The applet design follows the ***Observer*** design pattern (also known as the *Subscriber/Publisher* model)[9], which aims to decouple the generator of a message from the receiver. Every virtual instrument takes upon itself the responsibility of *listening* (*subscribing*) to an instrument whose output signal it depends on. When the output signal changes, the listener is notified by the instrument that generated the signal (the *Publisher*), and can then proceed to receive the new signal. The destination instruments (S*ubscribers*) for a signal, therefore, need not be directly established by the publisher instrument until it is indirectly informed during runtime.

This greatly improves the flexibility of the virtual instruments, allowing the applets to configure new experiments easily.

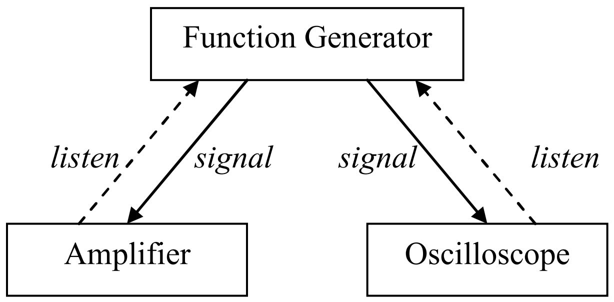


Figure 7 Subscribers register with the Publisher during initialization, publisher notifies subscribers of events during execution

Every virtual instrument in the Virtual Lab is designed as per the ***Model-View-Controller* (MVC)** design pattern[9] as shown in Figure 8. For every instrument, there are two classes defined – one that represents the *Model* and the other that represents the User Interface (which combines the *View* and the *Controller*.) The *Model* class maintains the properties of the component, computes the signal and handles the signal-change notifications. The User Interface (UI) class manages the layout and visual appearance of the instrument (the *View*) and also accepts the user's inputs (the *Controller*.) This separation of concerns helps abstract out the computation of signals from details of visual appearance, thereby allowing easy modification of display parameters without affecting the underlying signal synthesis.

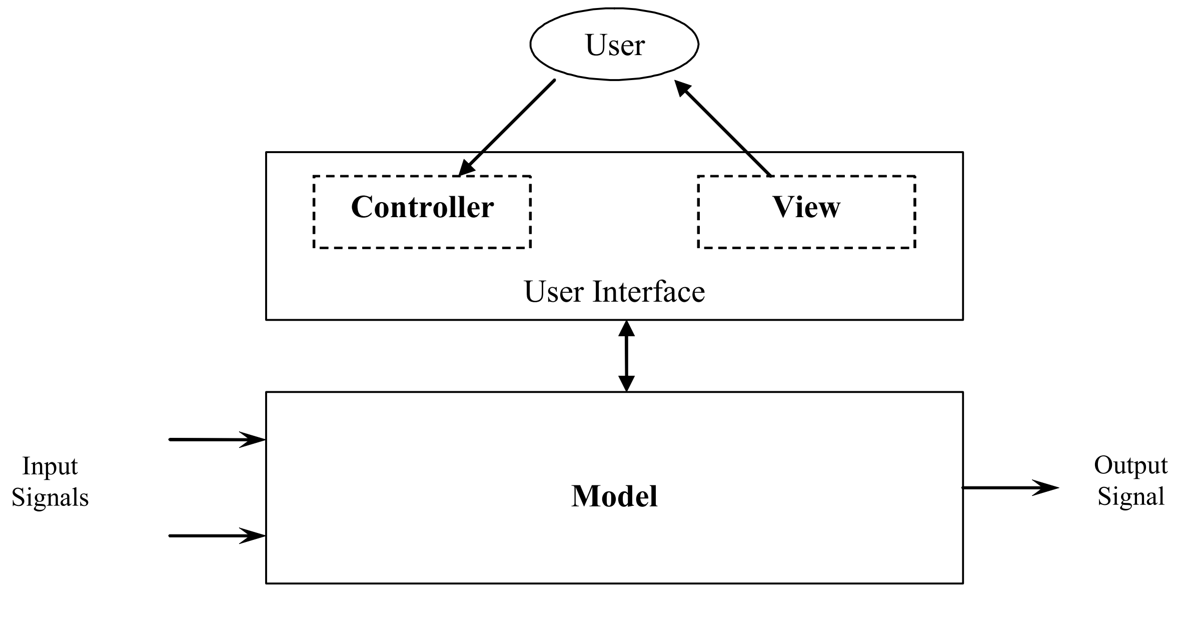


Figure 8 MVC architecture in virtual instruments

Signals in Virtual Laboratory

In the Virtual Lab, signals are represented in the form of objects. Since a continuous representation of arbitrary waveforms is not possible on a digital computer, the signal object carries only a discrete sampling of the waveform. The waveform is uniformly sampled, and the signal object stores the time-axis value and the corresponding amplitude value for each of the sampled points. The object may additionally carry other information such as the frequency of the signal.

The sampling rate and the number of samples collected are crucial to the functionality and accuracy of the experiment. However, they cannot be arbitrarily large due to limitations in computational power. For this reason, the following optimization has been implemented: When the signal object is first created, the oscilloscope component initializes it so as to make it compute samples only at points that are actually displayed on the oscilloscope screen. In other words, the signal is not sampled at points beyond the oscilloscope's display width. Since the width (in pixels) of the oscilloscope screen is fixed and limited, the signal object stores only as many points as there are pixel columns. However, the oscilloscope has several possible timescale values, and a different set of samples is required for each timescale. Instead of having the signal to be recomputed whenever the timescale value is changed, the signal pre-computes and stores samples for all possible timescale values. The first approach would have saved computational power but would have involved a reverse dependency between the signal creator (function generator) and the signal displayer (oscilloscope), and hence this approach was dismissed.

When sampling an arbitrary signal, there is always the risk of under-sampling, that is, the samples may miss out on important components of the signal. Although the sampling rate as obtained with the above method is sufficient to display a value at each pixel column, there is still the possibility of the number of samples being insufficient, especially at higher frequencies, and the displayed signal may be inaccurate. Therefore, in order to improve the accuracy of the displayed signal, more samples need to be taken. However, this increase in computation would result in a slower response time for the user's actions.

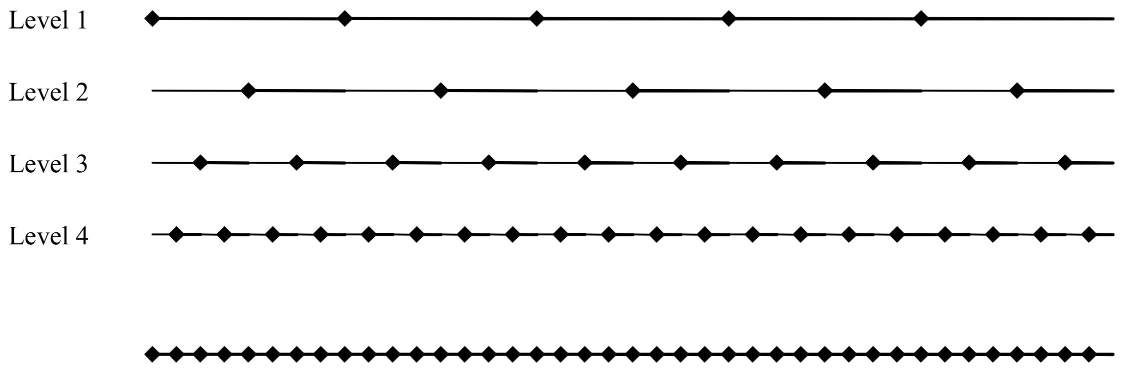


Figure 9 Signal levels and effective sampling

In the Virtual Lab, this situation is remedied by performing the signal computations incrementally, in a series of increasing **levels**. At each higher level, samples are computed at points that "fill in the gaps" of the previous level, thereby effectively sampling at a higher rate. When any level is completely computed, the next higher level is initiated only if the user has not changed any property values. If anyone of the property values is changed, the computation is halted and is restarted from the lowest level with the new property values. Thus, user response time is shortened while at the same time achieving a higher sampling rate.

A further improvement to response time can be achieved by performing the intensive signal computations in a background thread instead of using the event dispatch thread, which can now be left free for the user's action events. In order to incorporate both incremental computations and background threads, and still maintain independence and self-sufficiency of virtual instruments, each virtual instrument needs to be provided with its own separate thread. This thread begins signal computation whenever a property of the instrument is modified. It performs signal computations not only at its virtual instrument but also at all the downstream instruments. When there is a contention between this instrument's thread and an upstream thread, preference is always given to the upstream thread so that it does not have to wait before returning.

The spectrum analyzer places its own demands on sampling, and hence needs to be treated specially. A uniformly distributed set of samples is taken from a single time-period of the signal, and is used by the analyzer to compute the frequency components up to a certain range. This frequency range is iteratively extended by obtaining more samples in the higher levels in the same way as it is done with the oscilloscope samples.

Conclusions

The virtual laboratory was first introduced to the students at NCSU during Spring 2004 as an optional tool to enhance students' learning of some of the basic concepts. In Fall 2004, we began to assign virtual laboratory experiments such as the one shown in Figure 5 in weekly homework assignments. During this semester, the virtual laboratory was also used in the class-room to demonstrate the operation of test instruments before the students even saw the real equipment in the hardware laboratory. Our preliminary assessment results based on questions asked in the final exam indicate that the virtual laboratory resulted in a measurable improvement in teaching the basic equipment skills. A detailed effort to assess the impact of the virtual laboratory on student learning is underway. In the mean time, we are continuing to add new applets to the virtual laboratory. The existing applets are available for public use at http://courses.ncsu.edu/ece/ece200/.

Acknowledgments

The authors wish to thank Daniel Hoag for writing the original oscilloscope applet and many contributions during the start up of the virtual laboratory effort. The virtual laboratory was supported by National Science Foundation through a DUE-CCLI grant.

References

1 W. J. Rugh, "Signals, systems and control demonstrations," http://www.jhu.edu/ signals/, 1995.

2 J. M. R. Frederic de Coulon, Eddy Forte, "Kirchhoff: An educational software for learning the basic principles and methodology in electrical circuits modeling," IEEE Transactions on Education, vol. 36, no. 1, pp. 19–22, February 1993.

3 D. Y. Northam, “Introducing computer tools into a first course in electrical engineering,” IEEE Transactions on Education, vol. 38, no. 1, pp. 13–16, February 1995

4 J. Svajger and V. Valencic, “Discovering electricity by computer-based experiments,” IEEE Transactions on Education, vol. 46, no. 4, pp. 502–508, November 2003

5 M. C. Ozturk, J. Trussell, C. Townsend, G. Byrd, A. Mortazavi, M. Baran, T. Conte, B. O’Neal, G. Bilbro and J. Brickley, A New Introductory Laboratory for Electrical and Computer Engineering, Proceedings of the ASEE Annual Conference and Exposition 2003, June 22 - 25, Nashville, Tennessee

6 R. Giannetti, “An analog oscilloscope simulator with internet interaction capability for on-line teaching,” IEEE Transactions on Education, vol. 41, no. 4, pp. 344–348, November 1998

7 J.M. G. Palop and J.M. A. Teruel, “Work bench for electronic instrumentation teaching,” IEEE Transactions on Education, vol. 43, no. 1, pp. 15–18, February 2000

8 H.-P. Huang, “Java based distance learning environment for electronic instruments,” IEEE Transactions on Education, vol. 46, no. 1, pp. 88–95, February 2003

9 Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, “Design Patterns – Elements of Reusable Object-Oriented Software,” Addison Weasley, October 1994

Erwin D’ Souza is a graduate student in the Department of Computer Science at NC State University working toward his Master of Science degree. His expected graduation date is May 2005.

Mehmet C. Öztürk is a professor of Electrical and Computer Engineering at NC State University. His research interests include advanced processes for silicon based CMOS integrated circuits and novel processes for nanostructures. Lately, he has been fairly busy experiencing the joys of developing a new introductory course and laboratory for ECE sophomores.