The Intelligent Universal Virtual Laboratory (UVL)

Michael Duarte
Sr. Software Engineer, Temple University, Intelligent Systems Application Center, Electrical & Computer Eng. Department
Philadelphia, PA 19122 USA

Annapoorna Mahalingam[1], and Brian P. Butz[2]

Abstract - **The objective of this project was to create a realistic, real-time, electrical engineering virtual laboratory. This project targets individuals who do not have adequate mobility of their upper bodies to perform laboratory experiments. To provide a more realistic and enhanced learning experience, the users of the virtual laboratory are allowed the freedom to build and test a wide variety of realistic electrical circuits, and be able to perform curriculum-based experiments. The main goal is to create an environment similar to a real electrical engineering laboratory, and to offer the user a way to learn the different aspects of instrumentation and circuitry.**

Index Terms - Circuit Comparer, Intelligent Tutor, Interactive Software, Virtual Laboratory

INTRODUCTION

According to the Center for Disease Control [1], there are 13.6 million individuals who have limited hand use and another 16.3 million who have mobility limitations. In the field of science and engineering, there are approximately 109,700 persons with motor disabilities employed in the United States [2]. Also, approximately 31,300 students with motor disabilities were registered in science and engineering programs in 1995 [3]. This project is intended to encourage and assist individuals with such mobility disabilities to enter the field of electrical and computer engineering. Presently, disabled individuals with motor disabilities have a difficult time with the laboratory portion of the curriculum. At most, individuals with limited or no use of their arms and hands could only watch their lab partners perform the laboratory experiments. While better than nothing, this is not good enough for a quality laboratory experience.

The purpose of the Universal Virtual Laboratory (UVL) is to provide a disabled student with motor disabilities a realistic laboratory experience that can be done at the student's pace while providing a good, solid, curriculum-based background in circuit experimentation, as well as a virtual lab assistant to guide and assist the student.

Recent advancements in computer technology and availability have allowed the computer industry to develop hardware and software applications that address the needs of the physically disabled. Circuit simulation software has existed for some time, with the very first simulators being DOS text based programs such as *PSpice*. With the introduction of operating systems with graphical user interfaces (GUI), better laboratory simulation software became available, such as *Electronics Workbench*. However, these programs contain an interface that is difficult for the physically disabled to use, such as small buttons and an unfriendly breadboard. In addition, because the instruments and components do not look realistic, it can feel like a simulation, instead of a laboratory.

IMPLEMENTATION

This section describes the factors considered in the development of the UVL. The *User Interface* section describes the user interface and why it is designed the way it is. A detailed explanation of the various programs used to develop the UVL is discussed in section II, *The Design Process*. Section III, the *System Architecture*, gives an overall view of the components that result in the present UVL. The mechanism facilitating communication among the UVL's application programs is described in section IV. Finally, sections V and VI give an overview of the intelligent laboratory assistant.

I. User Interface

The main goal in the design of the user-interface is to present the user with a realistic environment as well as an environment that a disabled user can manipulate without difficulty. To do this, the laboratory was designed to allow many different types of assistive technology to work with the environment. Assistive technology allows disabled individuals the ability to manipulate a computer. Some of the devices that were focused on are: switches, large keyboards, and voice recognition. The UVL has also been developed to accept traditional mouse and keyboard manipulation. To make the user interface friendly to the above mentioned

[1] Annapoorna Mahalingam, Temple University, Intelligent Systems Application Center, Electrical & Computer Eng. Department
[2] Brian P. Butz, Professor, Temple University, Electrical & Computer Eng. Department

devices, a particular scheme of design was followed so that simple commands would perform major laboratory functions.

The user-interface consists of a breadboard with miniature instruments, with a variety of electrical components (see Fig. 1). The components available are: resistors, capacitors, inductors, diodes, zener diodes, potentiometers, variable capacitors, transistors, and jumper wires. At the workstation, the user has the freedom to build any type of circuit configuration possible. Typically, the student is given an experiment to complete, which includes circuit schematics to build and test circuit theory. The instruments available to the user are two DC power-supplies, a function generator, oscilloscope, spectrum analyzer, and a digital multimeter.

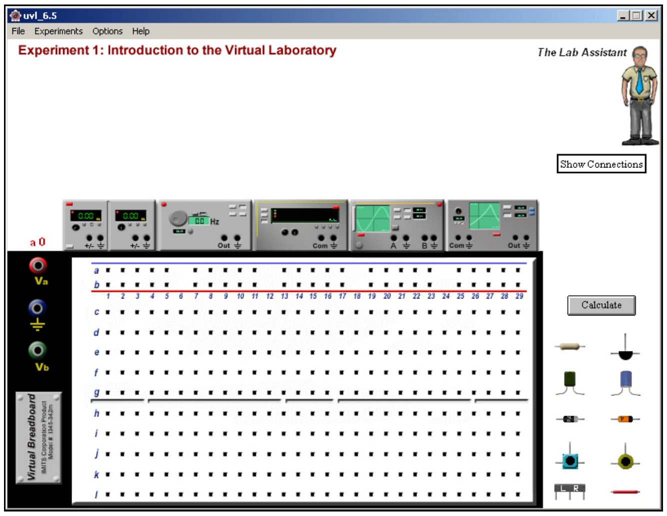

uvl_6.5
File Experiments Options Help
Experiment 1: Introduction to the Virtual Laboratory
The Lab Assistant
Show Connections
Calculate
Virtual Breadboard
Va
Vb

FIGURE 1
USER WORKSPACE / INTERFACE

The instruments are connected to the holes on the breadboard when the user selects the instrument of his/her choice and enters the coordinate of the hole the user wishes to place a cable. The components connect to the breadboard in a similar way. The coordinates are the letters and numbers seen on the breadboard (see Fig. 2). The user enters the letter and number corresponding to the hole on the breadboard where he/she wishes to place a wire, cable or component. An example of the interaction with the voice recongnition device, might be: "Move mouse right", "Click", "C, 2, 2000". This would move the cursor to the right, over a particular component, and place it on coordinate "C 2" with a value of 2000 ohms for a resistor.

This design scheme, to connect instruments and components to the breadboard, was determined to be the most effective way a disabled person with an assistive technology device could easily control the interface. For a traditional mouse and keyboard setup, the interface allows a user to simply drag a component or wire onto a particular node on the breadboard.

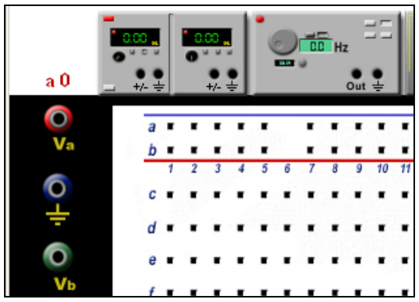


FIGURE 2
COORDINATE VIEW OF THE USER WORKSPACE

Each instrument in the workstation has a corresponding larger version that is used to change the parameters of that instrument. With the two input instruments (DC power-supply and function generator), the user can change the settings and control the type of signal being put through the circuit. The output instruments (multimeter and oscilloscope) can be used to measure the voltage and current of particular parts of the circuit. The user can manipulate the settings on these instruments to view the signal similar to actual instruments. Fig. 3 shows the larger version of the function generator and the oscilloscope respectively.

The specific instruments that were modeled are fully adjustable to a range of values found on the equipment in an ordinary electrical engineering laboratory. As can be seen from Fig. 3, the buttons, knobs and displays have a large area. This was done so that the disabled user could more adequately manipulate the instruments. If the user is inclined to use the cursor with the voice recognition software, he or she can more readily navigate the cursor over the large buttons and displays.

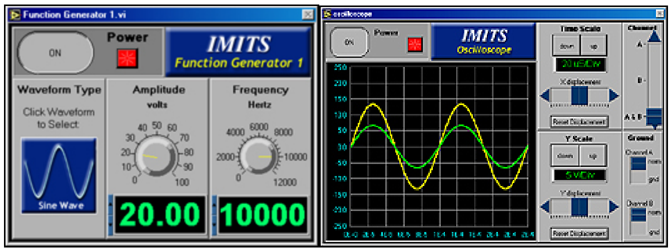


FIGURE 3
FUNCTION GENERATOR AND OSCILLOSCOPE

II. The Design Process

In the process of designing the UVL, the major issue encountered was: what software applications can be used that will make the system run and perform like a real laboratory. The laboratory had to give the user a way to produce a virtual layout of a circuit configuration. Once this layout was completed, it had to be analyzed to extract certain information about the circuit. Finally, this information had to be displayed on instruments that looked and functioned like real instruments found in a real laboratory.

Macromedia's *Authorware* was chosen to give the user a way to layout a circuit or build a circuit on a breadboard.

Authorware is an interactive multimedia development package that has been used as an authoring tool for computer-based training, because it makes it easy to handle a wide variety of media and precisely track and respond to users' actions. It can display images and text, as well as track and store user movements, and has the ability to communicate with other packages. Since users of the UVL needed the ability to move components, and create a circuit that seemed real, a package that could make interaction simple, and at the same time record the interactions was an important aspect for this project.

Authorware is designed so that a developer can easily add graphics and mobility to a screen. Also, innate to *Authorware* is the user tracking mechanism. This mechanism is the most important key to the UVL, because of the need to "know" what the user is moving, connecting, or changing on the screen.

If a user builds a circuit, *Authorware* can record the interactions and with a programmed algorithm, develop a description of the circuit. For example, if an image of a resistor is on the screen, the user could move the resistor to a particular part of another image, say the breadboard. Once the user has moved the image of the resistor to his/her desired spot, *Authorware* can store a numeric location of this image. This numeric location can then be used later to perform an analysis of the circuit.

Although *Authorware* can be programmed to do many different calculations and even execute C-based code, it cannot easily analyze a circuit. This brought up the issue of how could a circuit built or laid out in *Authorware* be analyzed to extract the appropriate information the user needs to see. *PSpice*, which is a circuit analysis tool, was an obvious choice in this case.

PSpice is a popular circuit analysis program used by many electrical engineers as a tool to analyze and test circuitry [5]. To use *PSpice* to analyze a circuit, one has to generate a text version of the circuit in a text file. This "text circuit" is called a "netlist," which is essentially a component-by-component "text" diagram of the circuit written in a specific format for *PSpice*. Knowing that *PSpice* needs this netlist file, *Authorware* had to generate a text file with the netlist of the circuit built by the user. Using *Authorware's* ability to track movements of images by the users, a translation of where the user puts an image of a component to a node location was plausible. Therefore, with *PSpice's* ability to analyze the circuit and store the analyzed data in another text file, there had to be a way to display this information.

National Instruments' *LabView* is a well-known package that is used in industry for instrumentation analysis [4]. This program seemed ideal, because of its ability to be customized through its native G-code (graphical code) language. Within *LabView* are dials, buttons, switches and the ability to process mathematical data easily. Knowing that *LabView* could look and perform like real instruments, it seemed like even more of an ideal choice for the virtual laboratory. These instruments created in *LabView* needed the ability to process information coming out of the *PSpice* output file. *LabView* can easily send information to other programs and analyze data from actual real instruments or from other programs. Essentially, the output from *PSpice* would be used as a data stream continuously sent to *LabView*. This stream would then have to be parsed for the appropriate information using a search algorithm.

III. The System Architecture

Fig. 4 shows the system architecture of the UVL. For simplification purposes, this architecture uses the voice recognition device as the interface to the laboratory. The user, via the voice interface *Dragon Dictate*, interacts with a packaged *Authorware* executable, which contains the user interface. *Dragon Dictate* is an off the shelf software package that allows users to manipulate the *Windows* environment, via voice recognition. The user also controls the instruments that are packaged as separate executables, which were created in *LabView*. The user cannot change or manipulate the code developed in *Authorware* or *LabView*. Instead, these packaged versions only run in the *Windows* environment, and allow the user to control only what is displayed to them. *PSpice,* which is used to calculate all the necessary information of the circuit the user builds runs autonomously and hidden from the user. At no point does the user interact with *PSpice* in any way.

The Intelligent laboratory tutor is a C++ natural language interface, developed specifically for this project. It accepts questions or a comment posed by the user and determines what information the user is requesting. The Intelligent tutor can also determine if a circuit the user has built on the breadboard is the same as the circuit they are required to build (this will be discussed further in the section VI). This C++ executable runs hidden from the user and seems, to the user, as a natural part of the user workspace.

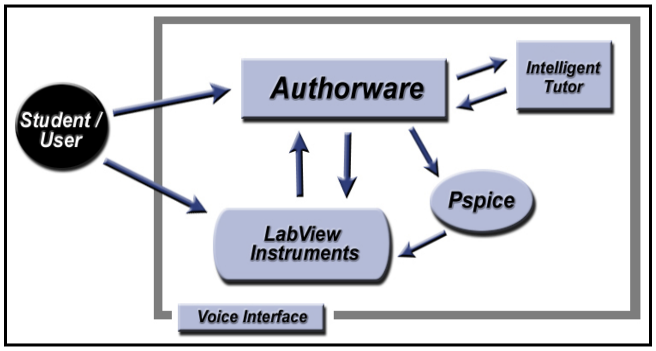
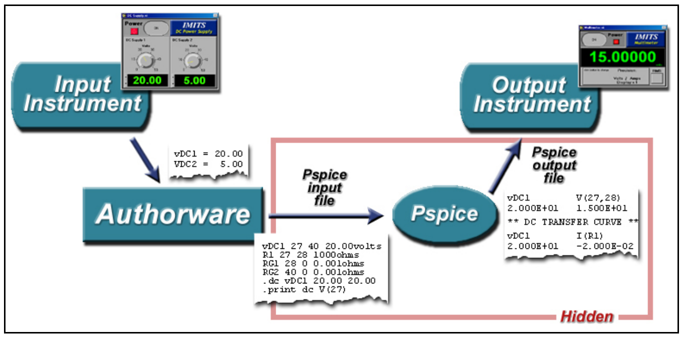


FIGURE 4
UVL SYSTEM ARCHITECTURE

IV. The Communications Channel

The architecture discussed above needed to work quickly and efficiently, without the user seeing what is happening in order for the UVL to seem like it is functioning in real-time. Through extensive testing of *Authorware, LabView*, and *PSpice's* input and output capabilities, it was concluded that using text files is a fast, efficient, and reliable method to create a constant communications channel (see Fig. 5).

This communications channel allows the virtual instruments to update in real-time (although slower than in a real laboratory). The input instruments (function generator, DC power supply, and sweep generator) send *Authorware* the settings the user has set, through a text file. *Authorware* then takes these settings along with the netlist it has created from the virtual breadboard and sends it to *PSpice*. *PSpice* then creates an output file with all the mathematical information of the circuit and sends it to the output instruments (oscilloscope, multimeter, and spectrum analyzer) for display. The creation and exchanging of files, as well as the *PSpice* calculation, is completely hidden from the user (see Fig. 5). It runs in a hidden DOS window within the *Windows* operating system.



```
vDC1 = 20.00
VDC2 =  5.00
```

```
vDC1 27 40 20.00volts
R1 27 28 1000ohms
RG1 28 0 0.001ohms
RG2 40 0 0.001ohms
.dc vDC1 20.00 20.00
.print dc V(27)
```

```
vDC1         V(27,28)
2.000E+01    1.500E+01
** DC TRANSFER CURVE **
vDC1         I(R1)
2.000E+01    -2.000E-02
```

FIGURE 5
THE UVL COMMUNICATIONS CHANNEL

V. The Intelligent Tutor/Laboratory Assistant

The Intelligent tutor found in UVL consists of two parts; a natural language parser and an intelligent circuit comparer, that we call the circuit recognizer. The circuit recognizer will be discussed in detail in the next section. The natural language parser portion of the tutor is used to answer questions that the user might pose during a laboratory session. When the user of the laboratory wishes to post a question or comment to the intelligent tutor, he/she calls the tutor by clicking the tutor icon. Once the tutor is activated, the user can either type or dictate their question or comment. This question is stored to another text file and the C++ executable containing the natural language parser loads this file and makes the necessary assessment of what the user wants to know. An example could be: "How do I measure the voltage across the resistor?"

With this information, the tutor parses the sentence and identifies the key words in order to make an assessment of what the user wants to know. The key words above are: "How", "measure", "voltage", and "resistor". These words are used to search the tutor's knowledge base and make a decision on what the user should see; which could be a simple tutorial on how to connect the digital multimeter across a resistor to measure its voltage. For more information on how the natural language parser performs the above operation, please refer to [6].

VI. The Intelligent Tutor/Laboratory Assistant – The Circuit Recognizer

Part of bringing the user a natural and helpful learning environment involves providing a facility that can evaluate their work, identify mistakes and provide appropriate advice and suggestions. The circuit recognizer is an intelligent circuit analysis module incorporated in UVL to examine and analyze the student's work, identify errors and provide explanations and suggestions to guide the user to the right answers.

As part of the lab, UVL provides the user with a set of experiments along with instructions on how to build and simulate circuits provided in each experiment. The user is given the freedom to build the circuit in any way that he/she wants, adhering however to the "essence" of the template. Therefore, when a student builds a circuit on the breadboard, the circuit may be connected in a manner physically different from the connections shown in the circuit schematic; and yet the circuits might be equivalent. In a real laboratory setting, if a student had a problem with an experiment/circuit he or she would approach the laboratory-teaching assistant (TA). Ordinarily, the TA would examine the student's circuit keeping in mind the objective of the experiment. The TA would check to see if the student's circuit has all the components that are required. Next the TA would examine if the required components are properly connected as illustrated in the experimental schematic. Based on observations, a conclusion would then be made about the causes and nature of the problem. The CR provides this functionality in UVL.

The CR has knowledge of circuit theory concepts, and is capable of using this knowledge to assess student work. The objective of the CR is to verify if the circuit that is created by the student is a valid representation of the laboratory's circuit schematic found in the experiments. If a student wishes to confirm that the circuit created on the breadboard is indeed equivalent to the schematic in the experiment, they have the option to call the CR by hitting the "check my circuit" button provided near the breadboard. The CR compares the student's circuit with the appropriate laboratory schematic. The result of the analysis is recorded in a text file, to be displayed to the user through *Authorware*.

ARCHITECTURE AND IMPLEMENTATION OF CR

The main objective of the CR is to function as a circuit configuration comparator and determine if the student created circuit on the breadboard is equivalent to the experimental schematic. The CR determines circuit equivalence based on the rule that, "If the components in the student's circuit are the same in value and type as the laboratory schematic, AND the manner in which they are connected is the same; then it can be said that the two circuits are component-wise and topologically equivalent. If any two given circuits are component-wise similar and topologically similar, then the two circuits are considered to be equivalent." The following sections will describe the various modules in the CR and their role in the determination of circuit equivalence.

I. Inputs and Preprocessing

UVL uses a computer-aided circuit analysis program, *Pspice,* to simulate circuits created in the lab. *Pspice* analyzes the circuit, and generates a file referred to as the output netlist. The netlist contains information about each component (type and values), connecting wires and measuring devices used in the circuit along with the nodes across which they are connected. Each time the user clicks on the "calculate" button after hooking up a circuit, *Pspice* is activated in the background and generates an output netlist of the circuit on the breadboard. This circuit output netlist is the input to the CR. In order to analyze the equivalence of two circuits, the CR requires the *Pspice* generated netlist of both circuits under consideration.

Before the CR proceeds to analyze the components and topologies of the circuits, there is some preprocessing that needs to be done. The circuit is condensed by removing wires and measuring devices, which are not considered to be part of the main circuit but serve only as adjunct or additional components. The CR begins by separating the components and connecting wires in each circuit in different files (wire file and component file). Routines are written which traverse the wire file and remove each of these wires. The removal routine effectively equates the two nodes across which the wire is connected, thereby reducing the wire to a single node/point. If a wire is connected in parallel across a component, the component is removed from the circuit as it amounts to a short circuit. A log of the components that are eliminated and the wires that shorted them out is maintained. If the elimination of wires resulted in a hanging node, then the circuit is identified to be open. The nodes that are hanging are noted down. In case of an open circuit or short circuit, the CR stops analyzing the circuit and communicates the error to the user. Once the circuit is stripped of all the wires, the measuring devices are removed. If a device is used in parallel across a component the device is merely removed from the circuit. If the device is used in series with a component, the device is removed and the nodes across which the device is connected is effectively equated. Separate modules determine if a measuring device is connected in series or parallel with a component. After preprocessing is done, the nodes of the circuit are identified and placed in an array. The CR then proceeds to analyze the circuit in the following two steps.

II. Component Check Module

The component check module verifies if all the components required by the experimental schematic is present in the circuit created by the student. From the component files (refer I) of the student and template circuits, components of a particular type (Resistor, Capacitor, etc) are picked out and placed in separate arrays. The tag with which it is represented in the *Pspice* netlist identifies each component. Each array (Resistor array, Capacitor array, inductor array) is then sorted in ascending order using selection-sorting algorithm [7]. Once this is done for all the components, the corresponding arrays of the two circuits are compared for order and individual values. If any disparity is detected, the missing or additional component is noted in the result file to be communicated to the student when the analysis is complete.

III. Topology Check Module

The second phase involved in determining equivalence, is to see if the two circuits are topologically similar. We begin by devising a topological model of a circuit that uniquely represents the topology of an electrical circuit. After much research and help from graph theory and state variable approaches to circuit solutions, the topological representation that was chosen for the CR is the *Proper tree* representation of a circuit. This section deals with the rationale behind this decision and the details of topological evaluation in UVL.

We begin our topological analysis by viewing an electrical circuit as a Graph [8]. An electrical network consisting of N_b branches and N_n nodes can be represented by a node branch matrix of order $(N_n\text{-}1) * N_b$ called the *Basic Incidence Matrix* (BIM). This matrix represents the node-component incidence pattern of an electric circuit network. The BIM will be used as the base to create the topological model.

Our interest is now directed towards a certain tree of an electric circuit graph, which we believe is the topological essence of the entire network. A network tree is a sub graph connecting all the network nodes but having no closed paths. This sub graph called the *proper tree* is defined as a tree of an electrical circuit network that contains (i) All the voltage sources as tree branches, (ii) all the current sources as links (iii) as many capacitors as tree branches as possible (iv) as many inductors as links as possible [10]. A link is an element when added to a network forms a closed path. Our contention is that two electrical circuit networks that have the same proper tree/trees have similar topologies. This is verified using the state variable approach to circuit analysis, which is a popular method to compute the solution of a circuit.

According to the state variable approach [9], the solution of a linear time invariant system can be obtained by computing the values of its independent system parameters. Once the independent state variables are computed the dependant can be determined from them and the circuit solution can be obtained. In a similar way the topology of a network can be obtained, if the position independent topological variables are known. That is if the position of certain key components in the network is known, the position of the rest can be obtained from them.

For the computer aided generation of state variables, a tree of a particular configuration is picked out so that it ensures that as many independent variables as possible are computed. The order for this tree is voltage sources, capacitor voltage, as few inductors as possible and the required resistors. Link currents and branch voltages are of prime interest. This tree configuration also represents an independent topological model from which the entire network can be generated, as the position of all other elements in the network is dependant on the above tree.

Having established that the proper tree/trees of a network accurately represent an electrical network topology, it is now implemented in UVL. The topology check module examines a circuit and picks out its proper trees. Once this is done for both circuits at hand, their proper trees are compared. If the independent tree/trees (from which the entire network topology can be extracted) for two networks is the same, the topology check module decides that the networks have equivalent topologies. The proper tree(s) of an electric circuit network is picked by performing matrix manipulations on the BIM using a specific algorithm [10]. Once it is established the circuits are equivalent, the user is told that the circuit is hooked-up correctly and told to evaluate their methodology. If the circuit is not equivalent, the user is told what is wrong with their circuit and is told to correct it.

CONCLUSION

The Universal Virtual Laboratory has so far been accurate, reliable, and easy to use. It gives the user enough freedom to create a feeling of a real laboratory environment. Consequently, this program has the potential to be beneficial and educational for a disabled student who is in the electrical engineering field. In addition, the UVL could be widely distributed and used in a wide variety of other educational environments. It could reduce the cost of equipment and perhaps even reduce the time spent in a structured laboratory curriculum.

On-going testing and modifications are being performed on the intelligent tutor part of the virtual laboratory. Although preliminary tests have shown that the intelligent tutor is functioning adequately, future work will be required to make it as close to accurate as a real laboratory assistant.

All the assistive technology devices discussed in this paper have been thoroughly testing by an assistive technology specialist, and have been proven to work effectively with the virtual laboratory. Development of algorithms to assist in the use of eye detectors and headset pointers is anticipated.

It is recommended that the UVL be used on a computer equal too or faster than a 750 MHz computer with at least 32 Mbytes of RAM. One aspect that takes a considerable amount of processor time is displaying the iterative calculations from the *PSpice* output file on the oscilloscope or spectrum analyzer. Using *Labview's* graphical programming language, the data is "stripped" from the *PSpice* output text files and then loaded into arrays within *LabView*. Code optimization within the *LabView* instruments could help the performance of the laboratory and is currently being investigated.

Full fledge testing of the virtual laboratory is anticipated to begin in the fall of 2005. Testing will consist of having students within the Department of Computer and Electrical Engineering at Temple University sit down with the UVL and perform experiments from the department's curriculum. This testing encompasses both the usability of the software, the accuracy of the intelligent tutor's understanding of the questions posed, and the accuracy of the circuit recognizer.

The UVL can run off of a CD-ROM, or over the Internet. Testing of the UVL online has shown no considerable loss of efficiency or any other problems. For more information, please visit http://www.temple.edu/IMITS.

ACKNOWLEDGMENT

Partial support for this work was provided by the National Science Foundation's Division of Undergraduate Education through grant DUE #9952291 and Human Resource Development through grant HRD-0004292.

REFERENCES

[1] Center for Disease Control (1995). *National Health Interview Survey*, 1994.

[2] National Science Foundation (1997). Award Abstract #9710548, *SBIR Phase II: Computer Simulation of Science and Technical Laboratory Exercises for Physically-Disabled Students*, Http://www.fastlane.nsf.gov/servlet/showaward?award=9710548.

[3] National Center for Education Statistics (1996). *The 1996 National Postsecondary Student Aid Study data system*, 1996.

[4] Wells, Lisa, and Jeffrey Travis. *LabView: for Everyone*. Upper Saddle River: Prentice, 1997. 6.

[5] Monssen, Franz. *MicroSim Pspice with Circuit Analysis*. Upper Saddle River: Prentice Hall, 1996. 1-2.

[6] Krishnasamy, Kaushik. "Intelligent Natural Language Interface", *Proceedings of the Seventeenth International Florida AI Research Society Conference*, Miami Beach, FL, 2004.

[7] Press William, Flannery Brian, Teukolsky Saul, Vetterling T. William, *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press; 2 edition, October 30, 1992.

[8] Vago Istvan, G*raph Theory: Applications to the calculation of Electrical Networks*. Elsevier: New York, N.Y., 1985.

[9] Brian Butz, "Computer-Aided Circuit Analysis using State variable techniques", *Masters thesis*, Drexel University, 1968.

[10] Chua, Leon & Lin, Pen-Min, *Computer-Aided Circuit Analysis of Electric Circuits: Algorithms and Computational Techniques.* Prentice Hall 1975