An Intelligent Universal Virtual Laboratory (UVL)

Michael Duarte
Electrical & Computer Eng. Department
Temple University
Philadelphia, PA 19122 USA

Brian P. Butz
Electrical & Computer Eng. Department
Temple University
Philadelphia, PA 19122 USA

Key Words: Assistive Technology, Interactive Software, Virtual Laboratory, Intelligent Tutor

Abstract — **The objective of this project is 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.**

I. 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.

This project attempts to provide the user with a familiar and realistic environment, using several existing Windows® applications. 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. Furthermore, the laboratory can be run from either a CD-ROM or from a website for distance learning engineering courses.

II. 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. Section B, the *System Architecture*, gives an overall view of the components that result in the present UVL. Specifics of the system architecture are discussed in section C, *The Design Process*. The mechanism facilitating communication among the UVL's application programs is described in section D. Finally, section E gives an overview of the intelligent laboratory assistant.

A. 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 will be designed to allow any type of assistive technology to work with the environment. Assistive technology allows disabled individuals the ability to manipulate a computer. Some of the devices that will be focused on are: eye gaze detectors; head mounted pointing devices and voice recognition software. To date the UVL has only been developed to facilitate the use of voice recognition software as well as traditional mouse and keyboard manipulation. Hence, the remainder of this paper will focus only on the aspects of the laboratory that were designed for voice recognition.

To make the user interface friendly to voice recognition manipulation, a particular scheme of design was followed so that simple voice 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). To date, the components available are: resistors, capacitors, inductors, diodes, zener diodes, potentiometers, variable capacitors, transistors, and jumper wires. At this workstation, the user has the freedom to build any type of circuit configuration possible. Typically, the student is given an experiment to complete. The instruments available to the user are a DC power-supply, a function generator, oscilloscope, spectrum analyzer, and a digital multimeter.

The instruments are connected to the holes on the breadboard when the user selects the instrument of his/her choice and via-voice 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 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.

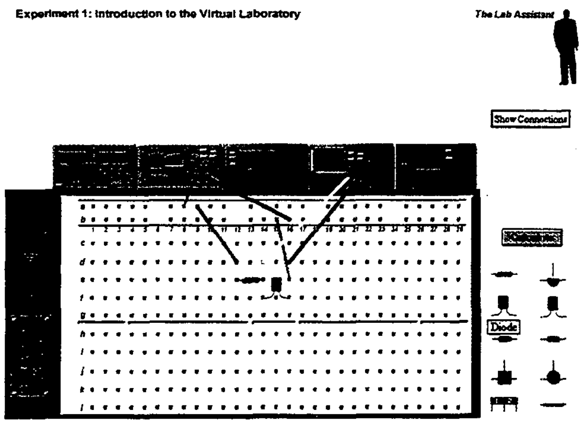


Fig. 1: User Workspace with circuit

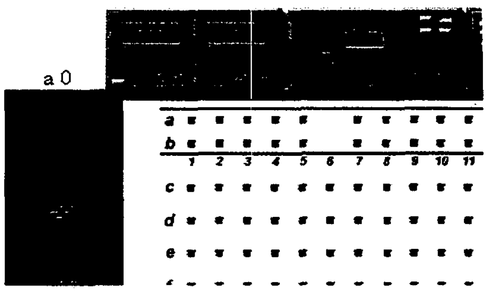


Fig. 2: Coordinate View of the User Workspace

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 similiar 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. The digital multimeter is able to measure the current and voltage of a DC circuit as well as measure the RMS value of the current and voltage of an AC circuit. The two-channel oscilloscope is able to display a current or voltage signal of both a DC and AC circuit. The oscilloscope also has the proper buttons and knobs that control the x-y displacement and the x-y ranges.

The function generator and the DC power supply can be used as inputs to the circuit. The function generator supplies a sinusoid or square wave signal, and has an adjustable amplitude and frequency setting. As for the DC power-supply, it supplies either a positive or a negative voltage to the circuit.

Finally, the spectrum analyzer in actuality has two main functions. It serves as both a frequency sweep generator and spectrum analyzer. The user can use the frequency sweep generator portion of the instrument to apply a frequency sweep to a circuit and measure the output of the circuit with the analyzer portion.

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.

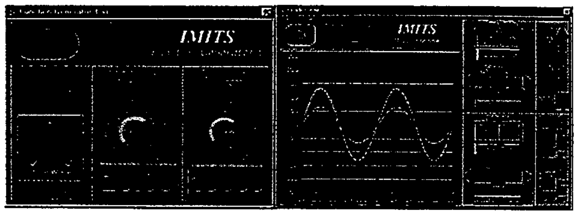


Fig. 3: Function Generator and Oscilloscope

B. The System Architecture

Fig. 4 shows the system architecture of the UVL. The user, via the voice interface *Dragon Dictate*, interacts with a packaged *Authorware* executable. *Dragon Dictate* is an off the shelf software package that allows users to manipulate the *Windows* environment, via voice recognition. In addition, the user controls the instruments that are packaged as separate executables similar to the *Authorware* package. 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. This C++ executable runs hidden from the user and seems, to the user, as a natural part of the user workspace. Further discussion of the intelligent tutor will be discussed in section E.

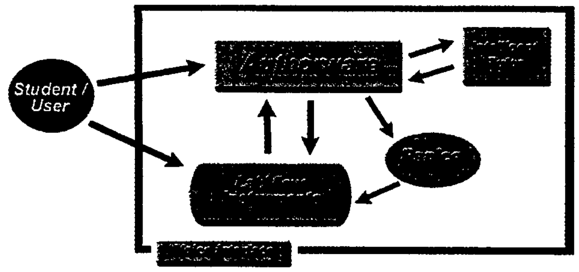


Fig. 4: UVL System Architecture

C. 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 look and function 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 use 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 movablilty to a screen. 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.

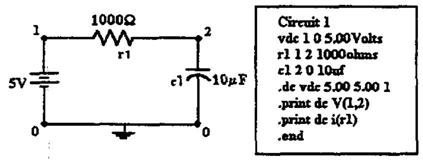
If a user builds a circuit, *Authorware* can record the interactions and with a programmed algorithm, develop a description of the circuit. For example, if an image of a resistor is on the screen, the user could move the resistor to a particular part of another image, say the breadboard. Once the user 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*. Take Fig. 5 as an example. In the figure is a simple circuit diagram and to the right of the diagram is the *PSpice* netlist version of the circuit. As can be seen, each component is labeled and has a numbered location on the diagram with its appropriate value. *PSpice* uses this information to analyze the circuit. Line 3 in the netlist file corresponds to r1 on the diagram. The netlist needs the name of the component, in this case r1, and then needs the location of the component (1 to 2), and finally the value of the component (1000ohms). These numbers or locations of the component are called nodes. In between any component in any given closed circuit are two nodes. These numbered nodes are what give *PSpice* the ability to calculate the circuit.

The last four lines of the netlist tell *PSpice* what analysis to perform on the circuit and what to display as the output. For this case, the input to the circuit is a battery, so *PSpice* will need to do a DC analysis on the circuit. The lines that contain V(1,2) and i(r1), tell *PSpice* to calculate and display the voltage across and the current through r1 respectively. This information is then stored in another text file. 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.



```
Circuit 1
vdc 1 0 5.00Volts
r1 1 2 1000ohms
c1 2 0 10uf
.dc vdc 5.00 5.00 1
.print dc V(1,2)
.print dc i(r1)
.end
```

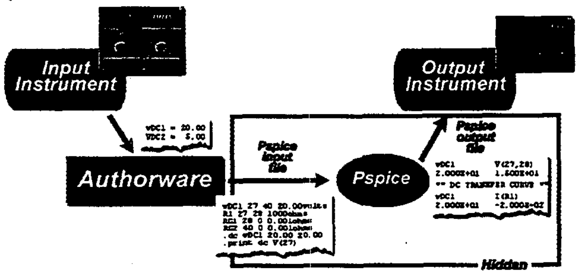
Fig. 5: Circuit diagram with *PSpice* netlist file

With these three application packages, it was concluded that text files were an easy and quite efficient way to have them communicate with each other. Since the computers of today have extremely fast processing times, text files were tested to be more than adequate for the communications link. The time it took to open, write, and close a file of a simple circuit was miniscule when tested.

D. 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. 6).

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. 6). 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
RC1 28 0 0.001ohms
RC2 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
```

Fig. 6: The UVL Communications Channel

E. The Intelligent Tutor/Laboratory Assistant

Fig. 7 shows the intelligent tutor's architecture. When the user of the laboratory wishes to post a question or comment to the intelligent tutor, he/she calls the tutor by either clicking the tutor icon or by saying "t". Once the tutor is activated, it asks the student to "point" to where he/she is having problems. At this point, the user either clicks or selects the component or instrument giving them problems and types or voices their question or comment. This question, along with the name and value of the component or instrument the user selected, is written to another text file. The C++ executable containing the intelligent tutor then 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 it?" The information passed to the tutor would be the question and perhaps "R1, 2000 ohms", if the user clicked on R1.

With this information, the tutor parses the sentence and identifies key words that are used in making the proper decision on what information the user wishes to know. The key words above are: "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.

Preliminary testing of the intelligent tutor has shown this method can work efficiently and quickly. On going testing and modifications are now under way. Testing consists 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 as well as the accuracy of the intelligent tutor's understanding of the questions posed, as well as the accuracy of the virtual laboratory environment compared to a real laboratory.

III. 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 accurate as a real laboratory assistant.

To date only traditional mouse and keyboard as well as voice recognition have been used to test the UVL. Additional testing using other input devices will also encompass future endeavors into the laboratories development. Development of algorithms to assist in the use of eye detectors, switches, joysticks, and headset pointers is anticipated.

The Universal Virtual Laboratory was developed on a Pentium® III 750 MHz computer with 128 Mbytes of memory. Due to the iterative nature of the calculations, the UVL performs only moderately on a 200 MHz computer with 16 Mbytes of memory. When tested on a 400 MHz Pentium II® with 32 Mbytes of memory, the UVL performed flawlessly. Therefore, it is recommend that the UVL be used on a computer faster than 200 MHz 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. For more information, please visit http://www.temple.edu/IMITS.

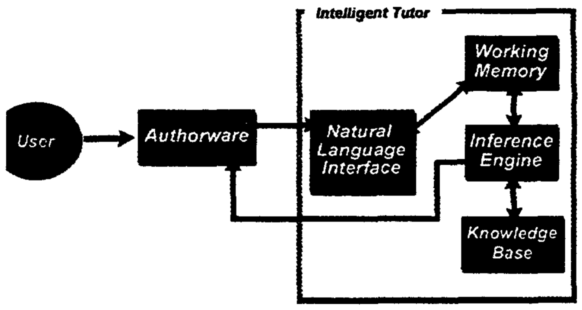


Fig. 7: The UVL Intelligent Tutor Architecture

IV. 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.

V. 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. 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.