A New Client Interface for the ELVIS iLab with Enhanced Capabilities

by

Olayemi A. F. Oyebode

S.B. Electrical Engineering and Computer Science, M.I.T., 2009

S.B. Management Science, M.I.T., 2009

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

February 2011

Author_____

Department of Electrical Engineering and Computer Science
February 7, 2011

Certified by_____

Jesús A. del Alamo
Professor of Electrical Engineering
Thesis Supervisor

Accepted by_____

Dr. Christopher J. Terman
Chairman, Department Committee on Graduate Theses

A New Client Interface for the ELVIS iLab with Enhanced Capabilities

by

Olayemi A. F. Oyebode

Submitted to the

Department of Electrical Engineering and Computer Science

February 7, 2011

In Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The MIT iLab project has developed online laboratories called iLabs that allow students to conduct real experiments on real equipment over the Internet. The ELVIS iLab, developed using National Instruments' Educational Laboratory Virtual Instrument Suite, has made it possible for students to develop a better understanding of engineering concepts by obtaining real data from an electronics lab. In the latest version of the ELVIS iLab, developed in this thesis, the user interface is redesigned to incorporate new features that will improve the experience of students and create new learning opportunities for students. As a result, students can enter values for parameters in the experiment more intuitively, have multiple experiments open simultaneously, view more variables in the graphs by adding new axes and defining new plots, retrieve earlier experiments and choose whether to connect or disconnect circuit elements.

Thesis Supervisor: Jesús del Alamo

Title: Professor of Electrical Engineering

Acknowledgements

I would like to acknowledge several people who have made the completion of this project possible. I am grateful to my supervisor, Professor Jesús del Alamo, and Jud Harward for the opportunity to be part of a project that has been making significant impact in tertiary education in Africa. I am indebted to Jim Hardison for explaining to me about iLabs, creating project goals, setting up meetings, assisting me with proposals and submissions since my first days with iLabs as an undergraduate student.

I appreciate my other colleagues at MIT's Center for Educational Computing Initiatives for their contributions too. I thank Kirky DeLong and Philip Bailey for solving and addressing technical problems that I could neither solve nor understand. I am grateful to Meg Westlund for organizing so many aspects of my iLabs experience and to Maria Karatzas for making sure I had the funds to work and travel for this project.

My gratitude also goes out to Hamidou Soumare and Rahul Shroff - two predecessors - for introducing me to iLabs and providing advice on project ideas. I would be remiss to omit Edison Achelengwa whose explanation of the switching module enabled me to meet my project goals.

Finally, I extend my thanks to my African iLab colleagues. I thank Dr. Alfred Mwambela, Josiah Nombo, Baraka Maiseli and the entire iLabs team at the University of Dar es Salaam; Cosmas Mwikirize, Arthur Asiimwe and the entire iLabs team at Makerere University; and the iLabs team at Obafemi Awolowo University for their feedback on ELVIS iLab 5.0, their hospitality and showing me their impressive iLab experiments. This has been a rewarding experience and I am delighted that my work will be contribute to the education of students across Africa, North America and the rest of the world.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Internet technology has made it possible for schools, universities and other educational institutions to offer Web-based online laboratories as an alternative to the traditional laboratory experience. Online laboratories are remotely-operated experiments accessed in real-time over the internet. Educational institutions that would have struggled or been unable to provide a traditional laboratory experience - such as distance-learning institutions and impoverished universities - use online laboratories to incorporate laboratory experiences in their course curriculum [1, 2]. Consequently, more students in technical and science courses are able to participate in valuable laboratory experiences. Even universities without such concerns can benefit from online laboratories as a complement to their existing hands-on facilities.

Online laboratories have several benefits over traditional laboratories. Online laboratories are flexible and can be accessed from anywhere at anytime. The flexibility lets students conduct their experiments at a convenient time and location. It also allows institutions to keep their laboratories open much longer than typically possible with the traditional hands-on laboratory. Furthermore, universities can use online laboratories to expand their offerings of available laboratory experiments with minimal additional cost as an entire class of students

can perform the same experiment on shared equipment. Certainly, online laboratories are not a perfect replacement for traditional laboratories. Students do not get hands-on experience with setting up experiments and have limited debugging abilities.

1.1 The iLab project

Prior to 1998, students in semiconductor courses at MIT did not have a laboratory component to their courses. In the fall of 1998, a web-based client and server were created that allowed students to take measurements on semiconductor devices on an Agilent 4155B semiconductor parameter analyzer over the internet [3]. This online lab, the Microelectronics WebLab, was very successful and expanded to other courses and accessed by other educational institutions. The Microelectronics WebLab became the precursor to the iLabs project.

The iLab project is the effort to develop scalable online laboratories using real equipment at educational institutions. Students access a web-based client to configure settings for the experiment while a lab server communicates with the lab equipment to both execute labs and retrieve results. The online laboratories in the iLab project have similar benefits as other online laboratories such as reduction in cost, remote access and 24/7 availability. The remote 24/7 access allows lab administrators to open up spare lab time on their iLab experiment to other external academic communities. Therefore, a single iLab experiment and its equipment can be shared between other institutions and groups.

The iLab project involves MIT and partners such as schools, universities, academic communities and other educational institutions in North America, Europe, Africa, Asia and Australia. Together, iLab members have developed new iLabs utilizing other equipment and in new disciplines such as physics, chemistry, civil engineering and mechanical engineering.

1.2 The iLab-Africa Initiative

The expansion and development of the iLabs project has been helped by strategic partnerships between MIT and other educational institutions. One such major partnership is the iLab-Africa program involving MIT and three sub-Saharan African universities using iLabs - Obafemi Awolowo University (OAU) in Ile-Ife, Nigeria; Makerere University (MAK) in Kampala, Uganda; and the University of Dar es Salaam (UDSM) in Dar es Salaam, Tanzania - supported financially by the Carnegie Corporation of New York [4]. The purpose of the iLab-Africa initiative is to explore the benefits and challenges to establishing iLabs in sub-Saharan Africa as a proxy for developing countries.

The iLab-Africa program has helped the three African universities include laboratory experiences in their technical course curricula. These universities often lack the state-of-the-art equipment and infrastructure resources available to institutions like MIT. With iLabs, they can overcome these limitations by saving on purchasing costs - taking advantage of economies of scale that iLabs makes possible - or by accessing iLab experiments at other institutions. In the early days of the initiative, students from these universities accessed iLabs developed by MIT [3, 5]. Now, the universities have their own development teams creating iLab experiments that are better suited for their individual curricula.

1.3 iLabs on the ELVIS Workstation

The four universities involved in the iLab-Africa program - MIT, OAU, UDSM and MAK - have primarily collaborated on developing electronic laboratories. These electronics laboratories have mostly been developed on the National Instruments (NI) Educational Laboratory Virtual Instrument Suite (ELVIS) (Figure 1.1). By developing on a common hardware

platform, the partner institutions have been able to easily share technology, ideas and expertise.

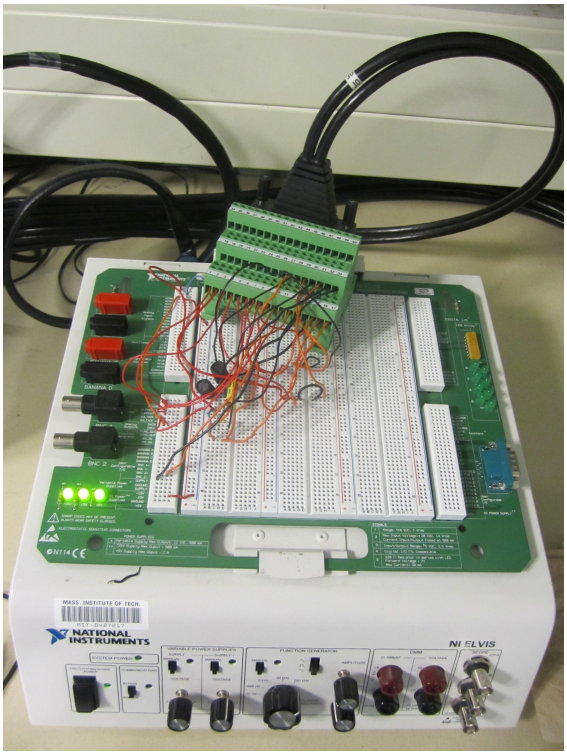


Figure 1.1: The ELVIS I

The ELVIS is a low cost, small-footprint, all-in-one electronics workstation that performs the function of instruments typically found in a basic electronics lab. Its variety of instruments and versatility has made it a suitable platform for development of electronics laboratories among the participating universities of the iLab-Africa program. It contains 12 instruments including an oscilloscope, a digital multimeter, a function generator, a bode analyzer, a variable power supply and an arbitrary waveform generator [6]. The ELVIS has a removable prototyping board where users can construct electrical circuits and connect the ELVIS's instruments. Additional third-party modules can be attached to the mainframe to create new types of experiments to be performed on the instrument.

The hardware is connected to a computer via NI's PCI-6251 digital acquisition (DAQ) card [7]. The ELVIS can be programmatically accessed using the NI LabVIEW, a graphical development environment containing and integrating a multitude of hardware devices and libraries. Since LabVIEW contains controls for other hardware, added third-party modules can be programmatically controlled in the same development environment as the ELVIS.

Previous versions of the ELVIS-based iLabs improved the student learning by providing access to more of the ELVIS's instruments and adding third-party modules to create more varied experiments. However, the ELVIS iLab's web-based client has remained nearly un-

changed since its initial version. Users have not been able to take full advantages of these new changes by being able to view more experiments, load previously executed experiments and view more output variables. Furthermore, the client could take advantage of the ELVIS iLab's flexibility to change the representation of the ELVIS's digital multimeter in the client so that students could learn more. These changes would create new ways for students to learn, but also improve the user experience.

1.4 Overview of Thesis

This thesis chronicles the development of version 5.0 of MIT's ELVIS-based iLabs. ELVIS iLab 5.0 has significantly modified the Lab Client used for MIT's ELVIS-based iLabs to improve the user experience and create new avenues for further development.

Chapter 2 details the responsibilities of the three major components of the ELVIS iLab's three-tier architecture and the communication between them. Following that, there will be an overview of the achievements that have been made in the previous ELVIS versions: from ELVIS iLab 1.0 to ELVIS iLab 4.0. Chapter 2 concludes with an explanation of why the user interface needs to be improved and an overview of the improvements in the ELVIS iLab 5.0.

Chapter 3 focuses on the design of the major improvements that are part of ELVIS iLab 5.0. There will be a detailed description of each improvement, the changes made to the web-based client and lab server with diagrams illustrating the changes. Some important limitations and issues will also be highlighted.

Finally, Chapter 4 states some of the important conclusions from the latest version of MIT's ELVIS iLab. Recommendations and suggestions about the ELVIS iLab's potential improvements and uses will also be included.

Chapter 2

Overview of the ELVIS iLab

Before discussing the changes that are desired in the ELVIS iLab platform, it is important to understand the iLab architecture that forms the structure of the ELVIS iLab. Since there is already existing information about these three components in other references [3, 8, 9], this chapter focuses on the aspects of the three components that are integral to this work and presents an overview of how they interact.

Then, this chapter will shift focus to presenting a profile of each major version of the ELVIS iLab prior to the latest version. This profile will highlight the major achievements and their impact on the experimental setups available within iLabs. Finally, the chapter will conclude by describing the five goals for the ELVIS iLab 5.0. These new goals have been inspired by annoyances from the user's perspective, iLabs on other hardware platforms and improving functionality added in previous versions of the ELVIS iLab.

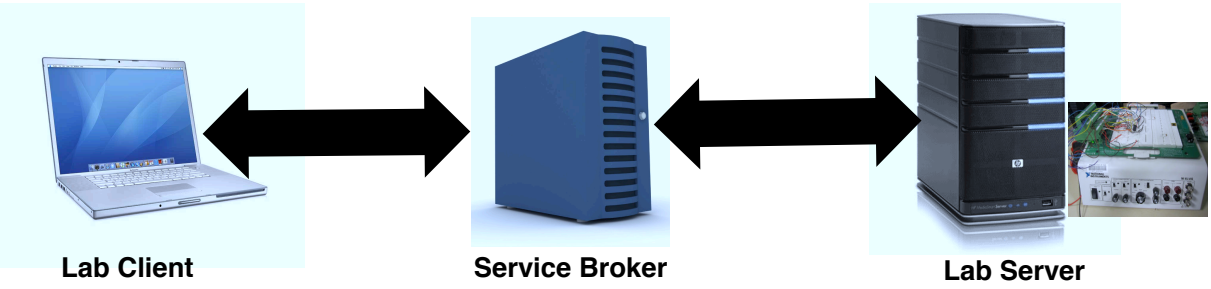


Figure 2.1: The three-tiered iLab Shared Architecture

2.1 The Architecture

In the MIT iLab Shared Architecture, iLabs are designed using a three-tiered infrastructure built on top of Web Services [3]. The three-tiered infrastructure consists of three key components: the Lab Client, the Service Broker and the Lab Server (Figure 2.1). This is known as the iLab Shared Architecture (ISA) because the Service Broker can be shared by multiple Lab Clients and Lab Servers. The iLab Shared Architecture uses Web Services based on the SOAP protocol.

2.1.1 Lab Client

The Lab Client is a web-based application that students use to set parameters for an experiment, send it to the Lab Server for execution and view results after execution. Students access the Lab Client after downloading it from the Service Broker. ELVIS-based iLabs use a Java applet as its Lab Client (Figure 2.2). The applet has four visual components which are described next.

Visual Components

The four visual components are the menubar, the toolbar, the Schematic Panel and the Results Panel. The menubar contains menu items that let users perform actions such as

selecting an experiment, executing an experiment, saving an experiment, viewing data and exporting data. The toolbar lets users quickly access four functions from the menubar: execute the current experiment, open a saved experiment, save an experiment and open up a help window.

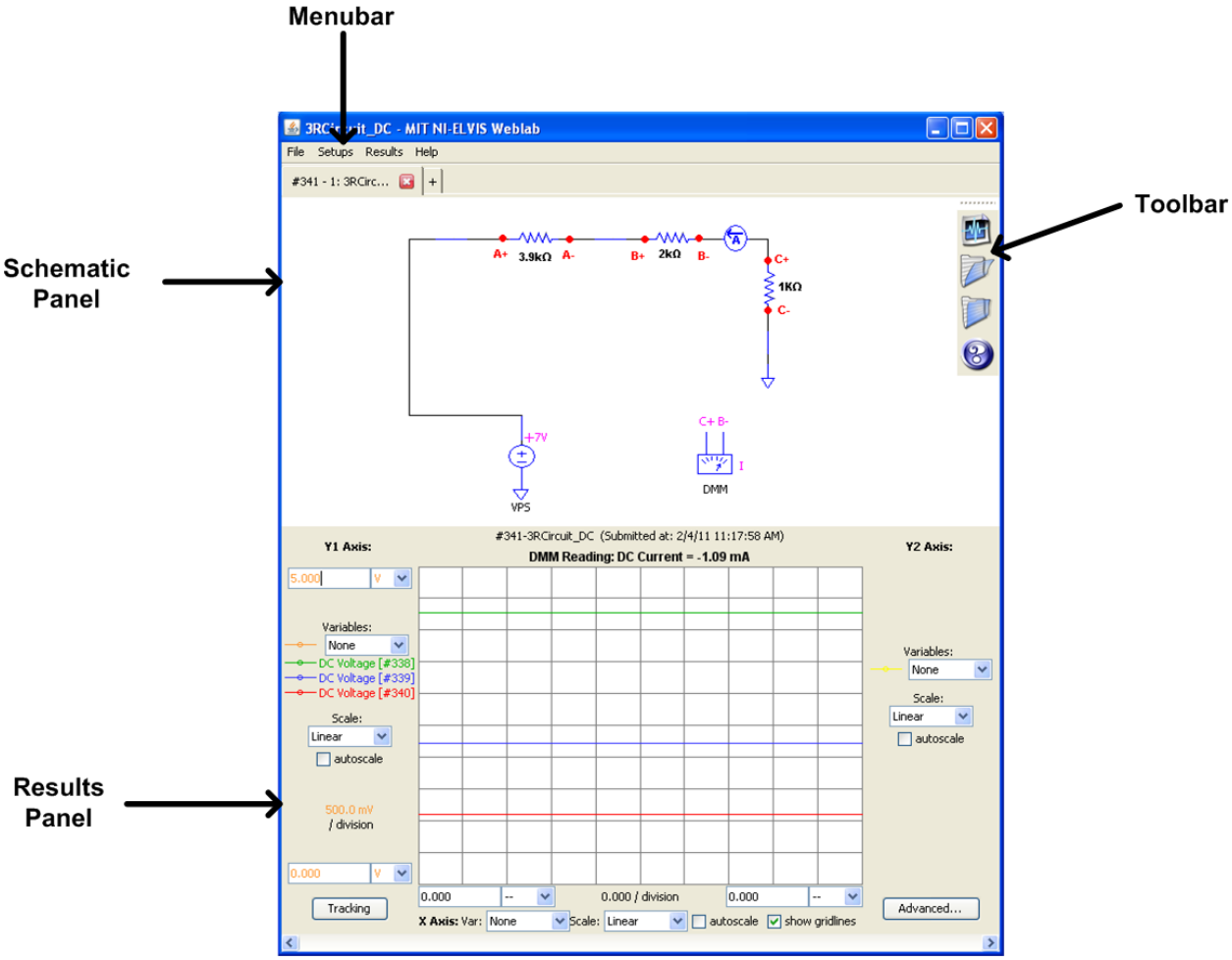


Figure 2.2: The ELVIS iLab 5.0 Lab Client

The Schematic Panel is an interactive display of an experiment's circuit and instruments. Images of the instruments, known as *instrument labels*, are either drawn below or within the circuit. To configure an instrument, the user clicks on its label to open an *instrument dialog* and enters values for the instrument's parameters. Once an instrument has been configured, its label will change to reflect its configured state.

The Results Panel displays the results of an executed experiment. It consists of a grid capable of plotting two dependent variables and one independent variable. It contains controls for choosing the variables, switching between linear and logarithmic scales and tracking values on the grid.

Structural Components

The three major structural components for the client are the user interface (UI) layer, the functional core layer and the server interface layer.

The UI layer consists of the classes that collectively govern the look-and-feel of the interface. Among these classes are those for the main frame for the applet, the schematic panel, the results panel, the instrument labels, the instrument dialogs, the grids and the axes.

The functional core layer consists of the classes that perform the functional logic occurring behind the background of the user interface. Within this collection of classes are those that store the internal state of each instrument, the data vectors for the variables that will appear in the results panel and the information contained in the messages passed between the Lab Client and Service Broker. These classes are also responsible for translating messages from the Service Broker that the UI layer will display to the user and conversely, taking the settings from the UI layer and generating messages for the Service Broker.

The server interface layer is an interchangeable layer containing the classes needed for the Lab Client to communicate via Web Services to the iLab Service Broker [8]. The classes within this layer are responsible for making the appropriate SOAP calls to either retrieve or send information.

2.1.2 The Service Broker

The Service Broker is a middleware system with several roles related to handling general management of iLabs and helps maintain modularity in the architecture. It manages user accounts, experiments, Lab Clients and Lab Servers. It also stores data for each experiment submitted by the user for execution. Logging into the Service Broker is the first step in using an iLab: the user accesses the ELVIS iLab by signing in at the Service Broker and launching the Lab Client. Unlike the Lab Client and Lab Server, the Service Broker is general to all iLab experiments; therefore, it can be shared by multiple iLabs.

2.1.3 The Lab Server

The Lab Server is a system that communicates with the laboratory hardware to both perform experiments and retrieve results. The Lab Server has a website that is the interface where lab administrators can create, modify and delete experiments as well as configure the connection settings to Service Brokers.

The Lab Server controls the execution of the experiment. It has a validation engine that analyzes each submitted experiment for correctness. If the validation fails, the Lab Server returns an error message; otherwise, the submission enters an experiment queue. Another Lab Server component, the execution engine, constantly and independently polls the experiment queue, executes the first item on the queue on the lab hardware and returns the results.

2.1.4 Communication

The communication between an iLab's Lab Client, Service Broker and Lab Server depends on its type. There are two types of iLabs: batched and interactive. In batched iLabs, the Lab

Client and Lab Server do not directly communicate. Instead, the Service Broker becomes their middleman. Interactive iLabs do not have this limitation as the Lab Client and Lab Server can directly communicate. This is necessarily because unlike batched iLabs, the user has real-time control of the experiment instead of submitting parameters to execute into a queue.

Since the ELVIS iLab is a batched iLab, its communication processes resemble those of a typical batched iLab. The Lab Client, Service Broker and Lab Server send messages to each other using SOAP and XML. There are three important types of messages sent via XML that are used to transfer information between the Lab Client, Service Broker and Lab Server: the Lab Configuration, Experiment Specification and Experiment Result (Figure 2.3).

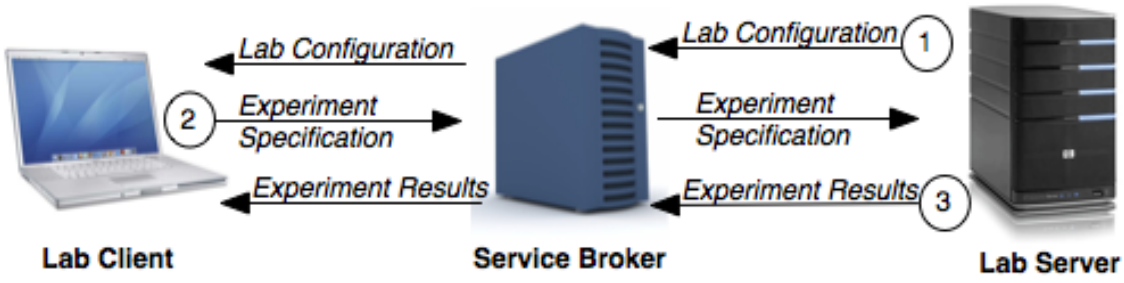


Figure 2.3: Transfer of XML documents between the tiers of the ISA

The Lab Configuration contains detailed information on each available experimental setup and their configurable instruments. Immediately after being launched from the Service Broker, the Lab Client sends a request to the Service Broker for the iLab's lab configuration. The Service Broker then makes a request for the lab configuration to the Lab Server. The Lab Server generates a Lab Configuration XML document and returns it to the Service Broker. The Service Broker transfers it to the Lab Client. The Lab Configuration Document Type Definition (DTD) and a sample lab configuration document are listed in A.1 and A.2 respectively.

The Experiment Specification contains information on an experiment's instruments and their

values. When the user submits an experiment for execution, the Lab Client generates the experiment specification for that experiment and sends it to the Service Broker, which passes it on to the Lab Server. The Lab Server uses the information from the experiment specification to determine what instruments to use and what parameters values to use. The Experiment Specification DTD and a sample experiment specification XML document and are listed in A.3 and A.4 respectively.

After the experiment has finished executing, the Lab Server stores the results for each output variables in an Experiment Result document. Meanwhile the Lab Client periodically polls the Service Broker for the status of the submitted experiment. If the Service Broker does not have the results for the submitted experiment, it will convey the same request to the Lab Server. The Lab Server will respond with *pending* until its execution engine has executed the submission and written the results back to the Lab Server. The Lab Server will then respond to the next request for the experiment's status with a *job complete* and the experiment result. The Service Broker will then send it to the Lab Client. The Experiment Result DTD and a sample experiment result document are listed in A.5 and A.6 respectively.

2.2 ELVIS iLab Development History

The inaugural ELVIS iLab was built in 2006 by Samuel Gikandi and borrowed source code from another iLab, the Microelectronics Device Characterization (Microelectronics) iLab [9]. In the debut version, only two of the twelve ELVIS' instruments - the function generator and oscilloscope - were exposed. The function generator produces a sinusoidal, square or triangle waveform with specified amplitude, DC offset, start frequency, stop frequency and step frequency [10]. The oscilloscope measures voltage waveforms on two different channels [10]. With these two instruments, the ELVIS iLab could only do experiments with time-

varying voltage signals.

Adnaan Jiwaji and Bryant Harrison added three new instruments - the arbitrary waveform generator, bode analyzer and variable power supply - in ELVIS iLab 2.0 [7, 11]. The arbitrary waveform generator produces a sinusoidal, triangular, sawtooth or square waveform with a specified amplitude, DC offset, start frequency, stop frequency, and step frequency specified by the user on two independent channels. Alternatively, the user can upload a custom waveform by entering a MATLAB formula or an external wave file [10]. The Bode analyzer measures the frequency response characteristics - gain and phase - of the circuit [10]. The variable power supplies produce a constant DC voltage not exceeding +/- 12V.

Also, ELVIS iLab 2.0 integrated a switching module into the hardware to add flexibility to the circuits used in the experiments. Lab administrators could now create experiments that asked the students to choose between different resistor, capacitor, inductor values or open/close a connection within the circuit.

ELVIS iLab 3.0 was developed by James Hardison for the Maricopa Advanced Technology Education Center (MATEC) to provide a remote laboratory experience that would enhance the curriculum for electronics technician students at American community colleges [5, 12]. This version forked from an incomplete version of the ELVIS iLab 2.0; it did not have the switching capability and its interaction to the instruments in the ELVIS hardware was substantially different. An intermediate version, version 3.5, was later created to reconcile the differences between 2.0 and 3.0. ELVIS iLab 3.5 was released to our iLab-Africa partners.

ELVIS iLab 4.0, developed by Hamidou Soumare and Rahul Shroff, combined versions 3.0 and 3.5 and brought in the ELVIS's digital instruments by exposing the digital multimeter, digital in and digital out [12, 13]. The digital multimeter measures AC/DC voltages and currents, resistance, capacitance and inductance, checks continuity of the circuit and performs diode

testing [10, 12]. Since the digital multimeter only measures one static branch, it was combined with the switching module in the ELVIS iLab so that it could measure up to 25 different combinations of branches to measure. The digital in and digital out were amalgamated together in the ELVIS iLab to create a new "instrument", called DOUT that writes data to and reads data from the digital channels. The DOUT could generate all the inputs necessary to create a 3-input truth table.

A timeline of the major versions of the ELVIS iLab has been provided in Figure 2.4.

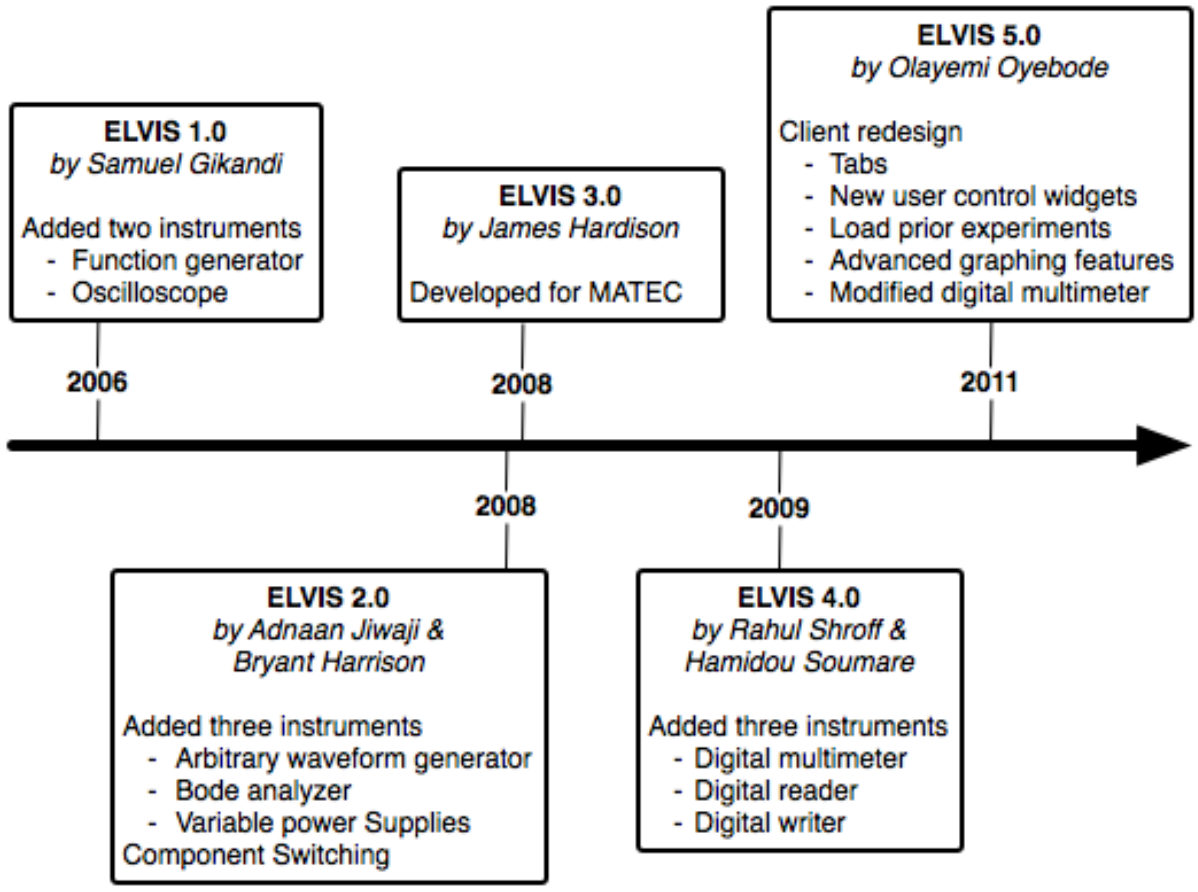


Figure 2.4: The ELVIS iLab Development Timeline

2.3 ELVIS 5.0 Development Overview

The development of the ELVIS iLab 5.0 was inspired by three concerns. Firstly, it was important to improve how users enter data in the Lab Client. Secondly, new features already existing in the Microelectronics Lab Client needed to be included in the ELVIS Lab Client to take advantage of the progress made by prior development efforts. Finally, there was a desire to revamp the implementation of the digital multimeter in the ELVIS iLab 4.0 to give more control to students.

2.3.1 Replace text boxes with spinners

The only user controls present in the ELVIS iLab 4.0 in the instrument dialog were text boxes and drop down lists. In the instrument dialog for a function generator instrument, the student uses a drop-down list to select a **Waveform** for the instrument (Figure 2.5). To enter the values for the function generator's **Frequency**, **Amplitude** and **Offset**, the student enters the numerical value in a text box and selects the scale of the unit to associate with that value. The drop-down lists are intuitive and user-friendly controls for selecting a single value out of a limited set of possible values such as units and waveform types. The text boxes, however, are neither intuitive nor user-friendly for entering numerical values.

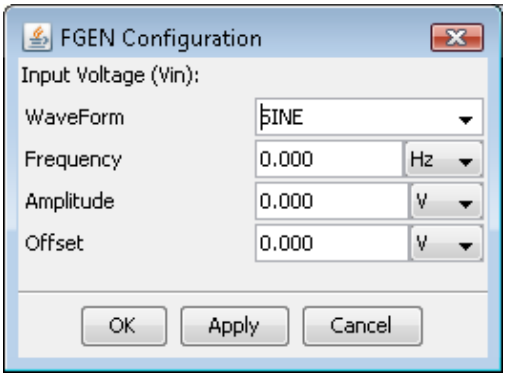


Figure 2.5: A dialog box for the function generator in the ELVIS iLab 4.0

A text box does not indicate the minimum or maximum value of a parameter or prevent accidental mistakes. Furthermore, if the value for a parameter is invalid, the user is unaware until after submitting the experiment and receiving an error message from the validation engine.

The ELVIS iLab 5.0 utilizes spinner controls instead of text boxes for entering numerical data to addresses these concerns and inconveniences.

2.3.2 Importing features from the Microelectronics iLab

Initially the ELVIS iLab could only support one experiment. Consequently, users could only load a single experiment at a time. However, when it became possible to have to have different experiments available, the Lab Client had not been changed to allow users to load multiple experiments at a time. This contrasted with the Microelectronics iLab, whose users could load and view multiple devices at a time in different tabs. One of the goals of ELVIS iLab 5.0 was to adopt this capability of multiple tabs so that its users could also have multiple experiments open inside the Lab Client.

In order to later retrieve an executed experiment's instrument settings within the Lab Client, a user must save the experiment prior to exiting the Lab Client. To retrieve the results from the same experiment, the user must visit the Service Broker to manually retrieve the results; it was not possible to retrieve it from within the Lab Client. A goal for ELVIS iLab 5.0 was to make it possible for users to retrieve the instrument settings *and* the results of a previously executed experiment entirely within the Lab Client by importing this feature from the Microelectronics iLab.

Finally, in the ELVIS iLab 4.0 Lab Client, the results panel displayed one independent variable and up to two dependent variables at a time. Meanwhile, students using the Microelectronic iLab could display up to ten dependent variables and create custom graphs. Another goal for the ELVIS iLab 5.0 Lab Client is to import this feature so that students would have more flexibility in viewing data.

2.3.3 Adapting the digital multimeter

To measure the voltage across a component or branch, the circuit must be fully connected and the digital multimeter's probes measure the terminals of the branch. To measure resistance, capacitance or inductance of a component or branch in a circuit, the component or branch must be disconnected from the circuit and its terminals measured by the digital multimeter. To measure current through a branch, the branch must be opened and then the digital multimeter's probes will bridge the circuit. Failure to set up the connections properly would result in an incorrect measurement.

In the ELVIS iLab 4.0, students only had to select a measurement mode, a range and the terminals of the branch being measured. The Lab Client then determined the connections necessary for a correct measurement and displayed a message alerting the user to the connections in the circuit (Figure 2.6) . A final goal for the ELVIS iLab 5.0 is to let students enhance their laboratory skills and experience by being able to set the connections.

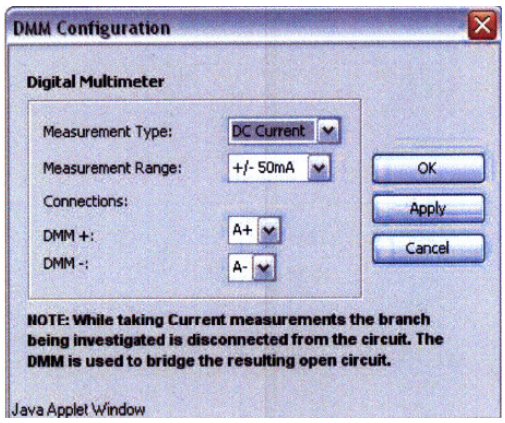


Figure 2.6: The old dialog window for digital multimeter

2.4 Summary

The ELVIS iLab's architecture consists of three components: the Lab Client, the Service Broker and the Lab Server. The Lab Client is the interface that allows students to execute experiments. While previous versions of the ELVIS iLab have improved the platform by providing more functionality, the latest version targets only the Lab Client for improvement. There are five goals slated for the ELVIS iLab 5.0 in order to continue to provide students

a better iLabs experience. The next chapter will present the modifications that have taken place in the Lab Client's visual and structural components in order to achieve these five goals.

Chapter 3

ELVIS iLab 5.0 Design

This chapter presents the modifications and additions made to the source code of the ELVIS iLab platform in order to achieve the five goals stated in section 2.3.3 and create the Lab Client in Figure 3.1. The achievement of each goal is profiled with a descriptive overview of the work necessary, details of any classes and/or functions in the Lab Client and Lab Server that had to be modified or introduced and a demonstration of each feature illustrated with screenshots.

3.1 Replace text boxes with spinners

3.1.1 Descriptions

In order to give students more intuitive controls for entering numerical data, spinners were added into the ELVIS iLab Lab Client. The source code was modified so that adding the spinners would occur in a modular framework. Because of the modularity, future developers could enhance the Lab Client by adding other controls in the future.

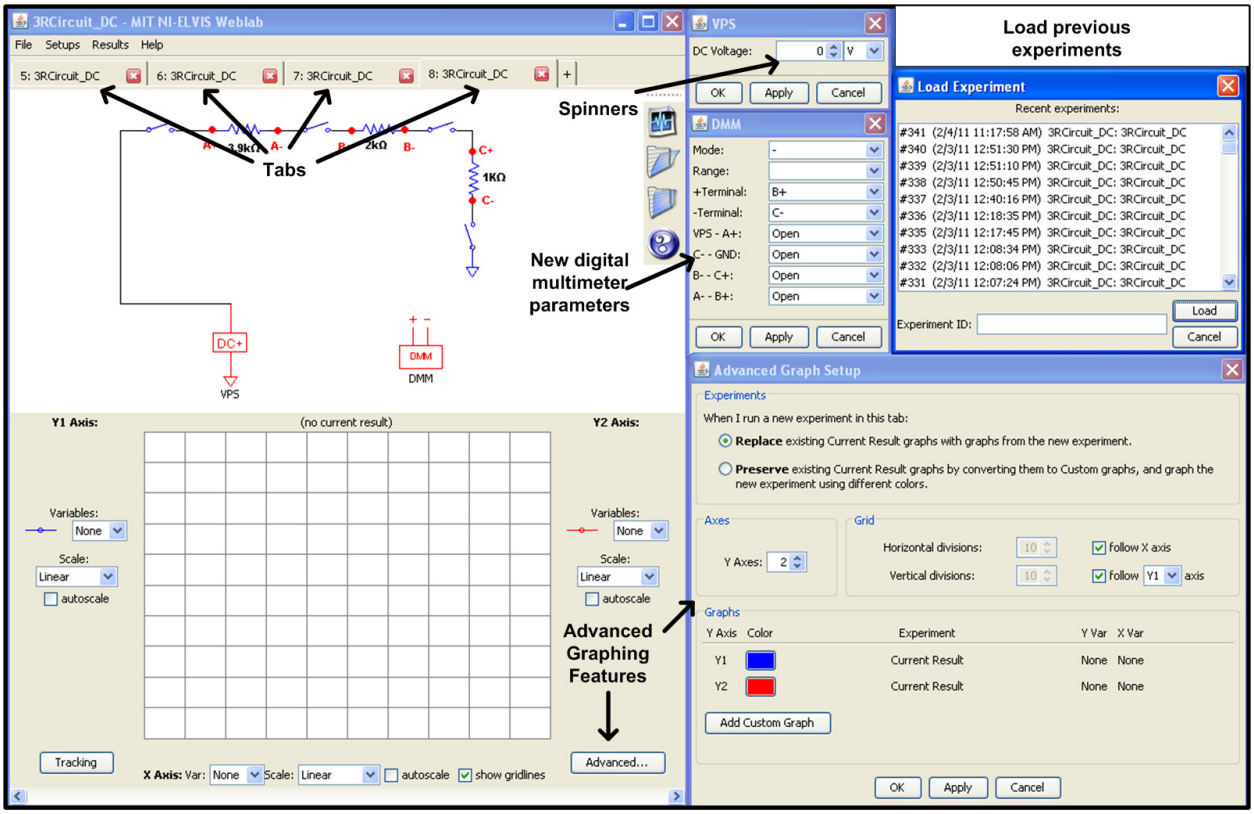


Figure 3.1: The ELVIS iLab 5.0 Lab Client can display tabs, spinner controls, new digital multimeter parameters, previous experiment data and custom made axes and graphs.

3.1.2 Lab Client Code Modification

In the ELVIS iLab 4.0, the combined text field and drop-down list used to enter numerical data and select units are grouped together into an `EngValueField` object. In the new design, `EngValueField` was changed into an abstract class and two child classes were created: `EngValueTextField` and `EngValueSpinnerField`.

The source code for `EngValueField` was slightly rewritten so that it used a generic user interface component instead of a text field. Its child classes would be responsible for using specific components such as text fields and spinners. The methods for retrieving and setting the value contained in a component, `getValue` and `setValue` respectively, were turned into abstract functions. The source code of the previous version of `EngValueField` was moved into `EngValueTextField`.

`EngValueSpinnerField` uses a spinner as its component. It must be supplied with a minimum and a maximum value for the spinner box. These minimum and maximum values are used by event handlers that trigger whenever the value in the spinner or units box has changed. These event handlers check to make sure that the spinner only contains valid numerical values. If the student enters a value that exceeds the maximum, the event handlers will set the spinner's value to the maximum value; if the value falls below the minimum, the event handlers set the spinner's value to the minimum value.

Both `EngValueTextField` and `EngValueSpinnerField` implement the `getValue` and `setValue` methods for accessing the values in their respective components. This approach minimized the disruption to other parts of the applet that use the previous `EngValueField` and also set up a template to allow for other user control widgets such as check boxes and radio buttons to be used in the instrument dialogs in the future.

3.1.3 Demonstration

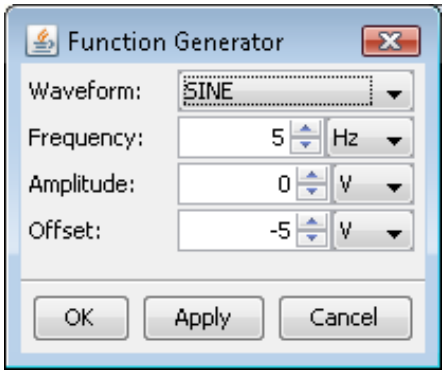


Figure 3.2: New dialog for function generator

To configure a function generator in the ELVIS iLab 5.0, the user clicks on the instrument label to open its configuration dialog. Similar to the ELVIS iLab 4.0 Lab Client, the user selects an appropriate waveform in the **Waveform** field by using a drop-down list (Figure 3.2). To enter the values for **Frequency**, **Amplitude** and **Offset**, the student enters the numerical value in a spinner or uses the up-/down arrows to increment/decrement the values. The valid range of values for the function generator's **Frequency**, **Amplitude** and **Offset** are 0.005 - 500 kHz, 0 - 2.5 V and -5 - 5V respectively. When the user enters a number below 5Hz for the **Frequency**, such as entering *3* in the spinner and selecting *Hz* as the unit, the value is corrected to be 5 Hz. If the user

enters a number above 5V for the **Voltage**, the value is corrected to be 5V.

By adding support for spinners, students now have a more intuitive way to enter numerical data when they configure the parameters for instruments. In addition, future developers now have a template for easily introducing newer controls into the Lab Client.

3.2 Tabs in the Lab Client

3.2.1 Description

Since it is possible to have multiple experimental setups, the Lab Client needed to be upgrade to handle multiple experiments. The solution, inspired by the Microelectronics 7.0 Lab Client, was to make it possible for the Lab Client to have multiple tabs and have instances of the experimental setups loaded into a tab.

Previously, the applet's frame could only interact with one set of client elements - Schematic Panel, Results Panel, lab configuration, experiment specification and experiment result. In ELVIS iLab 5.0, the applet's frame interacts with a list of tabs and each tab maintains its own set of client elements.

The user interface was further modified so that users were aware of the tabs and make use of them. The user interface for tabs borrows functionality, menus, shortcuts, layouts and other UI elements common to tabbed web browsers. Users can create, delete, reload or copy tabs in the applet.

3.2.2 Lab Client Code Modification

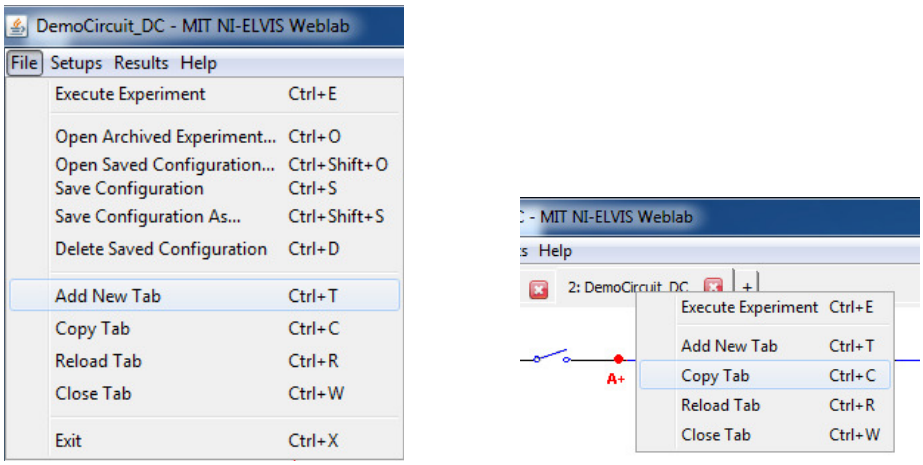
In previous versions of the ELVIS iLab, the `MainFrame` class directly created and displayed the `SchematicPanel` and `ResultsPanel` objects in the Lab Client. This organization created a one-to-one relationship between the applet and the schematic and results panels. Furthermore, the `MainFrame` object had a one-to-one relationship with the `WeblabClient` object that stores handles communication between the Service Broker and the Lab Client. The `WeblabClient` object also manages the `LabConfiguration`, `ExperimentSpecification` and `ExperimentResult` objects that store the information contained in the lab configuration, experiment specification and experiment result XML documents respectively.

In ELVIS iLab 5.0, the `ExperimentTab` and `Experiment` classes were added so that the client elements could be abstracted away from the `MainFrame` and consequently the applet. Each tab is an `ExperimentTab` object and maintains its own `SchematicPanel` and `ResultsPanel`. In addition, each `ExperimentTab` has a unique one-to-one relationship with an `Experiment` object, which maintains a set of `LabConfiguration`, `ExperimentSpecification` and `ExperimentResult` that corresponds to an experiment's lab configuration, experiment specification and experiment result XML documents respectively.

However, the `MainFrame` and `WeblabClient` still need to interact with the active experiment's schematic and results panel as well as XML documents for actions such as submitting an experiment for execution and updating the instruments in an experiment. The solution was for the `MainFrame` to maintain a dynamic `ExperimentTab` object that points to the active `ExperimentTab`. The `MainFrame` and `WeblabClient` then retrieve the active `Experiment` via the `ExperimentTab`'s `getExperiment` function.

The **File** menu was modified so that users could utilize the tabs (Figure 3.3a). Four new menu items were added:

- **Add a New Tab** creates a new tab with the first experiment automatically selected
- **Copy Tab** duplicates the active tab along with its schematic panel, results panel and all internal state into a new tab
- **Reload Tab** resets all instruments in the experiment to their default settings
- **Close Tab** discards the active tab; if the active tab is the last remaining tab, a new experiment tab is automatically created



(a) The new file menu (b) The right-click menu

Figure 3.3: Menu items for accessing tab functions

Four additional features that not present in the Microelectronics iLab Client were added so that the behavior of tabs in the ELVIS iLab could mimic the behavior found in multi-tabbed web browsers and applications. First, students can right-click on a tab to access the **Add a New Tab**, **Copy Tab**, **Reload Tab** and **Close Tab** functions (Figure 3.3b). Second, there is a small tab with a "+" at the end of the tabbed pane that when clicked triggers **Add a New Tab**. Third, each tab has a close button to the right of its title that when clicked triggers **Close Tab**. Finally, the applet supports keyboard shortcuts and accelerators similar to those commonly used in multi-tabbed web browsers.

3.2.3 Demonstration

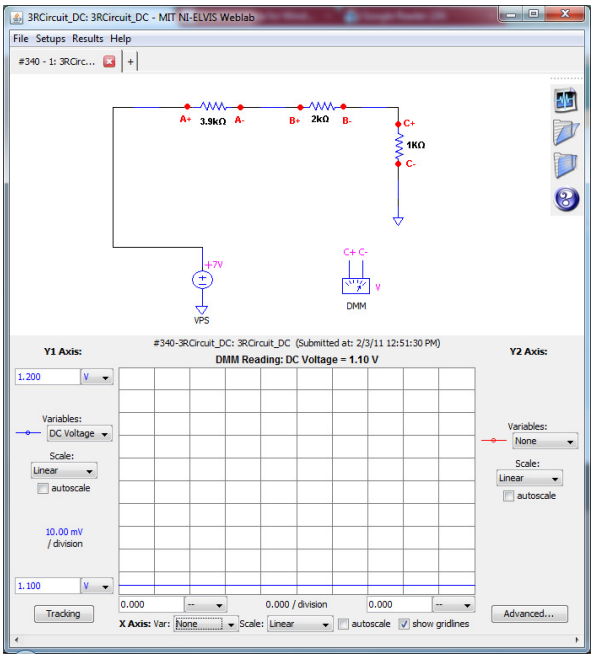
In this demonstration, a student has already measured the voltage across a resistor in a circuit and will proceed to add a new tab to the applet, make a copy of the instrument configuration settings of the original tab, reset the instrument settings in the original tab and finally remove the original tab from the applet.

After executing the experiment, the student can add by a new tab by clicking the tab with a "+" icon, using the operating system-specific keyboard shortcut for adding a new tab, clicking **Add a New Tab** in either the **File** menu, right-clicking on a tab and clicking **Add a New Tab**. The newly created tab is added to the end of tabbed pane, becomes the active tab and also loads the first experimental setup. The new tab has its own instruments, configuration settings and display elements (Figure 3.4).

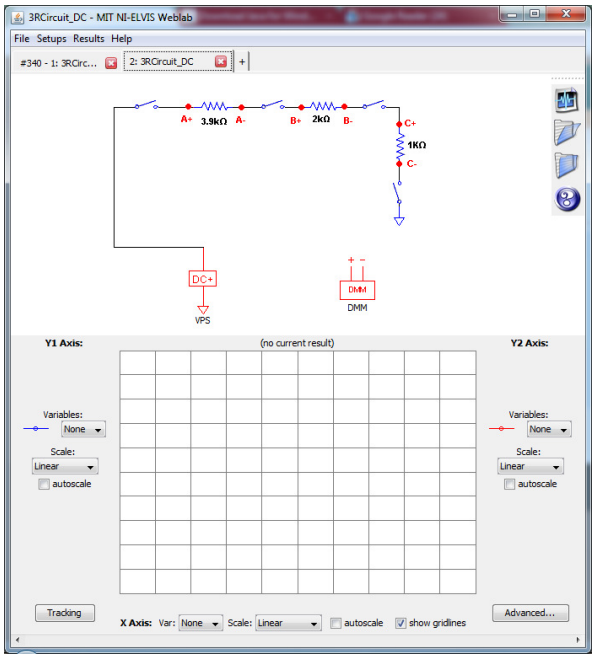
To duplicate all instruments and configuration settings in the active tab, the student clicks **Copy Tab** in the **File** menu, clicks **Copy Tab** in a right-click menu or uses the appropriate keyboard shortcut for **Copy Tab**. The circuit schematic and instrument settings of the copy exactly mirror those of the original tab, but the results panel is not copied (Figure 3.5). The duplicated tab also becomes the active tab.

Then, when the student desires to reset all instruments to their default values, the student clicks **Reload Tab**. All parameters in the instruments within the original tab will return to their initial unconfigured values. The icon of the instrument will also revert to its initial image at launch (Figure 3.6).

To close the active tab, the student can again go to either of the menus and click **Close Tab**. In addition, the student could also use a keyboard shortcut or click the close Icon to the right of the tab's title. When a tab is closed, it is remove from the tabbed pane and the leftmost tab becomes the new active tab (Figure 3.7).

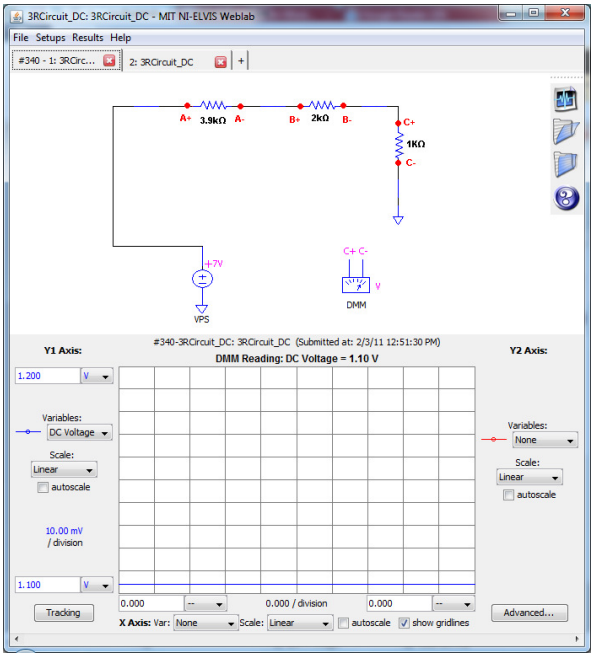


(a) Applet with one tab

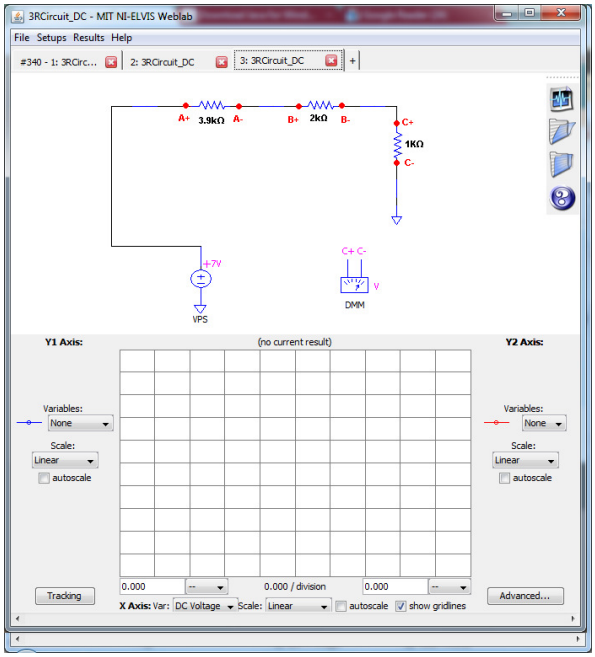


(b) Applet with a newly added tab

Figure 3.4: Add a New Tab

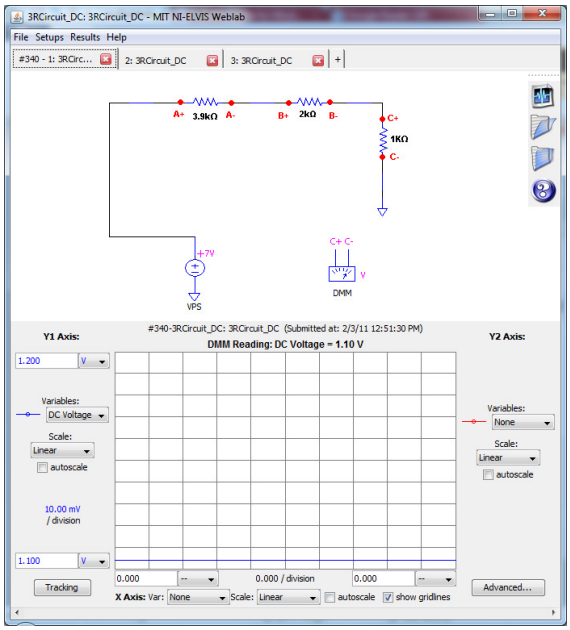


(a) The original tab

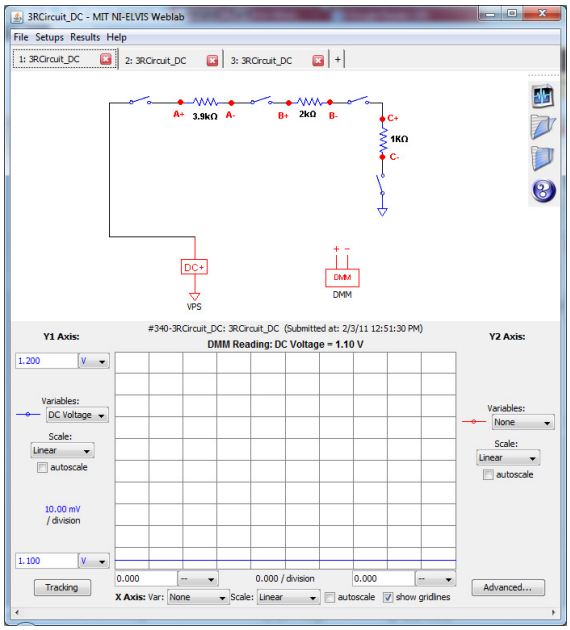


(b) The duplicate tab

Figure 3.5: Copy Tab

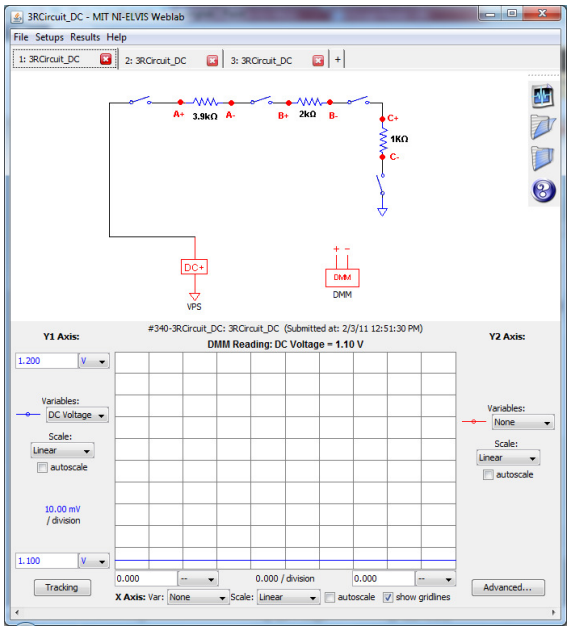


(a) A tab configured with instruments

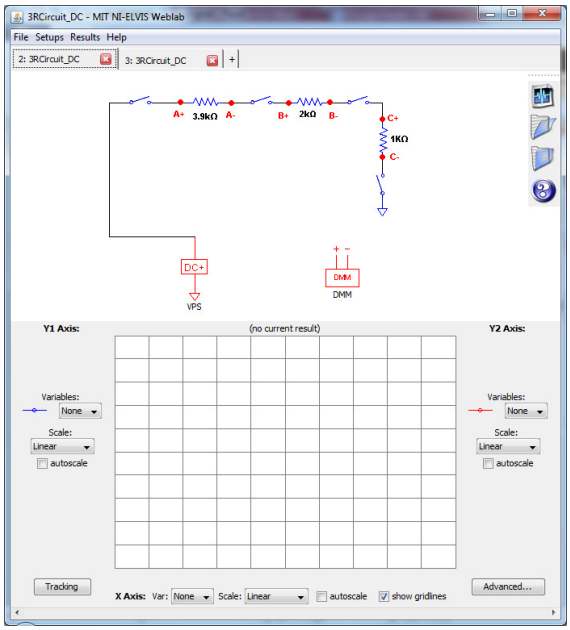


(b) The tab reset to its default settings

Figure 3.6: Reset Tab



(a) The first tab is about to be removed



(b) After removing the first tab, the second tab is now the active tab

Figure 3.7: Close Tab

3.3 Load previous experiment data

3.3.1 Description

Though students could retrieve their experiment data directly from the Service Broker, it would be more valuable for students to be able to load and retrieve them within the Lab Client. Therefore, the students could execute an experiment and compare with previous executions. In addition, the user would be able to view both the configuration settings of the instrument and the results in a single place.

The ability of the Lab Client to retrieve and load previous experiments was implemented using two methods. In the first method, the Lab Client maintains a local copy of each experiment successfully executed during the current session. When a user requests to load an experiment executed during the same session, the Lab Client would access its stored copies and load the appropriate copy. The second method would be used for experiments executed from previous sessions. The Lab Client would retrieve the experiment's information - lab configuration, experiment specification and experiment result - from the Service Broker after providing the appropriate parameters.

3.3.2 Lab Client Code Modification

To store a local copy of each executed experiment, the `MainFrame` copies each `Experiment` once its experiment specification has been successfully executed. The duplicated `Experiment` objects are maintained in a map indexed by the ID of the original experiment.

The Service Broker Web services do not include a method for the Lab Client to search and list all experiment IDs from a user's sessions on an iLab. The Service Broker could not be modified because it would require that all Service Brokers around the world be modified.

To circumvent the problem, the Lab Client saves lists of experiment IDs as *client items* on the Service Broker. Client items are general purpose pieces of information saved on the Service Broker by the Lab Client; they can be retrieved at a later time or session by the client. For example, an experiment's experiment specification can be saved to the Service Broker as a client item.

The Lab Client uses the `Server` interface and `SBServer` object for interacting with the service broker. The `Server` interface defines service broker-related methods such as saving and retrieving client items while the `SBServer` class implements these methods. In the ELVIS iLab 4.0, the only client item with defined methods in the `Server` interface and `SBServer` object was the experiment specification. Consequently, the `SBServer` could only save experiment specifications. The `Server` interface in the ELVIS iLab 5.0 includes support for generic client items by adding four new methods to `Server` and `SBServer`.

- `listAllClientItems` - returns a list of the name of existing client items
- `deleteClientItem` - deletes a client item from the Service Broker
- `loadClientItem` - loads a client item from the Service Broker
- `saveClientItem` - saves a client item on the Service Broker

Three changes were made to the `WeblabClient` object. First, after each experiment has successfully executed, `WeblabClient` adds the experiment's ID to a list maintained as a client item on the Service Broker using `SBServer`'s `saveClientItem` method. This list is specific to the user account. Also, only IDs of experiments executed after this change would ever be stored in such a list. The second change was to add the `getRecentExperimentIDs` method for retrieving this list. The final change was to add the `loadArchivedExperiment` method that would retrieve an experiment from the Service Broker.

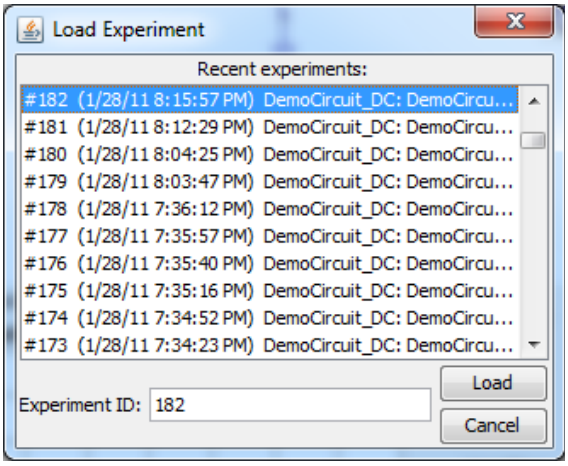
A new class, `LoadExperimentDialog`, was created to show a dialog window for selecting

experiments to the user. The dialog window contains a listbox to display the experiments, a text box to enter the ID of an experiment and **Load** or **Close** buttons. The IDs in the listbox are generated by using the `getRecentExperimentIDs` method of the `WeblabClient`. If an experiment and its ID do not appear in the listbox, the user can enter it in the text box.

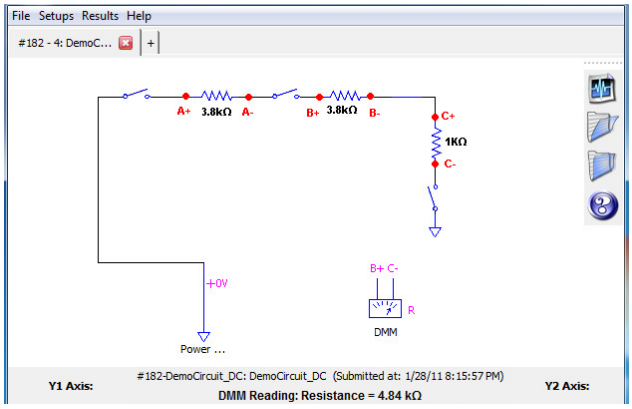
A menu item, **Load Archived Experiment**, was added to the **File** menu so that the user could access this dialog window. Clicking this menu item will present a dialog window, the `LoadExperimentDialog` to the user. After the user selects an experiment or enters its ID, the ID of that experiment will be passed to the `MainFrame`. The `MainFrame` checks whether it has stored the experiment ID in its map. If the experiment ID is found in the map of experiments from the current session, the `MainFrame` returns a clone of the `Experiment` and all of its settings including the schematic panel, results panel, instruments, data variables and XML documents. However, if the experiment ID not found, the `MainFrame` uses the `WeblabClient` to retrieve and parse the lab configuration, experiment specification and experiment result XML documents of the experiment from the Service Broker. The display elements are then recreated on the Lab Client.

3.3.3 Demonstration

To access a previous experiment, the student goes to the **File** menu and clicks the menu item **Load Archived Experiment**. A dialog window will pop-up and the student can either select an experiment in the listbox or manually enter an experiment's ID (Figure 3.8a). Once the user clicks **Load**, the experiment will be retrieved and loaded into the active tab (Figure 3.8b). In the example in Figure 3.8, the student previously measured the resistance between B+ and C- using the digital multimeter and has retrieved it.



(a) Load experiment dialog



(b) Experiment #182 is loaded into a new tab

Figure 3.8: Loading a previously executed experiment

3.4 Modify DMM features

3.4.1 Description

The goal of modifying the representation of the digital multimeter is to expose students to the same decisions that they would have to make in a traditional hands-on laboratory setting. In the previous implementation of the digital multimeter, the Lab Client adjusted the relays between circuit elements as needed for an accurate measurement. Each relay represents a connection between two elements of a circuit.

In the new implementation, the connections have been exposed graphically to students. Therefore, the student must now directly set the state of the relay to get accurate measurements. For each relay, the user has the option of deciding to connect it, have it stay open or open and bridged with an ammeter.

3.4.2 Lab Client Code Modification

To allow for the updates to the ELVIS iLab's implementation of the digital multimeter, two new classes - `DMMCONN` and `DMMLabConfigInfo` - were added to the Lab Client. A `DMMCONN` represents a relay between components in the circuit on the ELVIS or the connection between neighboring circuit elements such as the variable power supply and the first 3.8KΩ resistors (Figure 3.9). The `DMMLabConfigInfo` is a static class that stores and retrieves `DMMCONN` objects for all digital multimeter instruments in the applet.

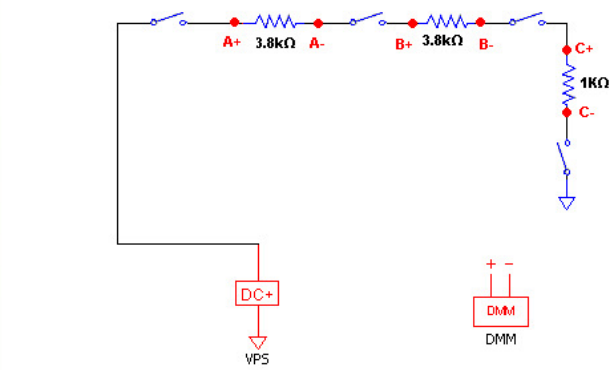


Figure 3.9: A circuit with all relays opened

For each relay specified in the Lab Configuration, a `DMMCONN` object is created. Since the digital multimeter uses a 25-character array to set the status of the 25 switches integrated into the ELVIS platform, the `DMMCONN` object stores the index number of the switch corresponding to each relay (see Appendix B).

In the digital multimeter's dialog window, a label and a drop-down are dynamically added for each `DMMCONN` object. The label displays the name of the relay specified in the lab configuration XML document and the drop-down list allows the user to set the state of the relay to *Open*, *Short* or *Ammeter*. An *open* connection indicates that the relay is unconnected. A *short* connection indicates that the relay is connected. An *ammeter* connection indicates that the relays are not connected to each other but bridged by the multimeter's ammeter.

Similar to the previous implementation, the Lab Client generates a 25-character string indicating the state of the relays based on the settings for the digital multimeter. If the user selected *Short* for a `DMMCONN`, the state of its relay will be to '1' whereas if the user selected *Open* or *Ammeter* for a `DMMCONN`, the state of the relay will be set to '0'. Furthermore, the state of each relay will be displayed graphically on the Lab Client (Figure 3.10).

(a) short (b) open (c) ammeter

Figure 3.10: Connection states and their symbols

3.4.3 Lab Server Code Modification

The Lab Server's database was modified so that it could store the name, orientation and switch index of each relay as well as whether it has a terminal that connects to ground. The Lab Configuration XML document that the Lab Server uses to pass the information to the Lab Client has also been modified (Figure 3.11).

Though all measurements from the digital multimeter have noise (Table 3.1), an adjustment was necessary for current measurements. The ammeter used in the ELVIS hardware is ground referenced which causes a common mode voltage measurement error when one of the ends is not directly connected to ground. Empirical measurements on the particular ELVIS hardware used for this thesis had measurement error ranging between 650 μA and 800 μA. Therefore, the current measurements made on this device are adjusted by 725 μA. This adjustment value will be different for each ELVIS hardware and a permanent workstation-independent solution must be sought.

```
<terminal instrumentType='DMM' instrumentClass='output' insturmentNumber
    ='8' setupTermlD='9'>
  <label>Digital Multimeter</label>
  <pixelLocation>
    <x>64</x>
    <y>78</y>
  </pixelLocation>
  <numConnections> 2 </numConnections>
</terminal>
```

(a) ELVIS iLab 4.0

```
<terminal instrumentType="DMM" instrumentClass="control" instrumentNumber
    ="2" setupTermID="18">
  <label>DMM</label>
  <pixelLocation>
    <x>2</x>
    <y>2</y>
  </pixelLocation>
  <connection name="VPS - A+" x="-30" y="9" horizontal="True" switch="24"/>
  <connection name="C- - GND" x="268" y="110" horizontal="False" switch
      ="21"/>
  <connection name="B- - C+" x="240" y="9" horizontal="True" switch="22"/>
  <connection name="A- - B+" x="120" y="9" horizontal="True" switch="23"/>
</terminal>
```

(b) ELVIS iLab 5.0

Figure 3.11: Declaring a digital multimeter in the Lab Configuration XML document

3.4.4 Demonstration

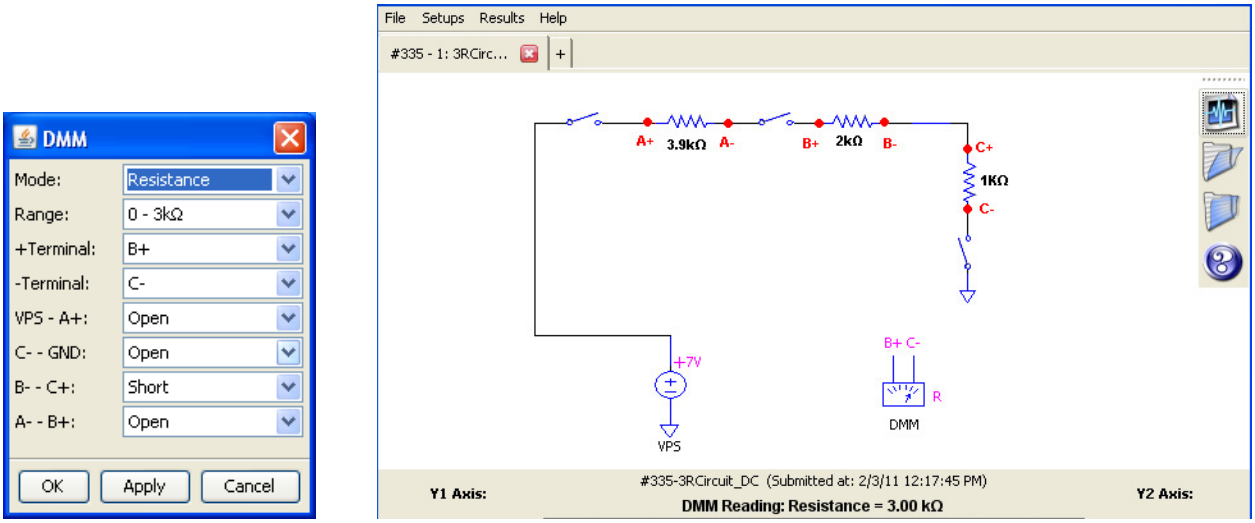
In order to measure using the digital multimeter, the user must set the following parameters for the digital multimeter - **Mode**, **Range**, +**Terminal**, **-Terminal** - and in addition, also set whether a given wire should be shorted, opened or opened and bridged with an ammeter. **Mode** is the list of measurements that the digital multimeter can make: *Resistance*, *Capacitance*, *Inductance*, *DC Voltage*, *AC Voltage*, *DC Current* and *AC Current*. The **Range** is the range within which the user expects the returned value to fall. The range affects the accuracy of the measurement. If the true value of a measurement exceeds the selected **Range** or is outside the range indicated in Table 3.1, the digital multimeter will return a value of

Measurement	Range	Accuracy
Resistance	5 Ω - 3 MΩ	1%
Capacitance	50 pF - 500 μF	2%
Inductance	100 μH - 100 mH	1%
AC Voltage	-14 V_{RMS} - 14V_{RMS}	0.3%
DC Voltage	-20 V - 20 V	0.3%
AC Current	250 mA	0.25% ±3 mA; reduces to 200 μA when properly null corrected
DC Current	250 mA	0.25% ±3 mA; reduces to 200 μA when properly null corrected

Table 3.1: The range and accuracy of the digital multimeter's measurements

infinity. The +**Terminal** and **-Terminal** respectively define the "+" and "-" labeled nodes that define the ends of the circuit branch being measured.

Measuring resistance, inductance and capacitance



(a) Set to measure resistance

(b) The resistance between B+ - C- is 3.0 kΩ

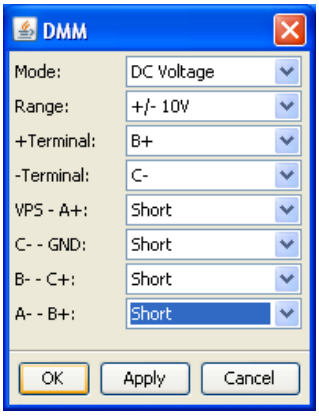
Figure 3.12: Measuring resistance with the digital multimeter in ELVIS iLab 5.0

In a hands-on laboratory, to measure resistance, capacitance or inductance, the student must isolate the branch or component being measured from the circuit.

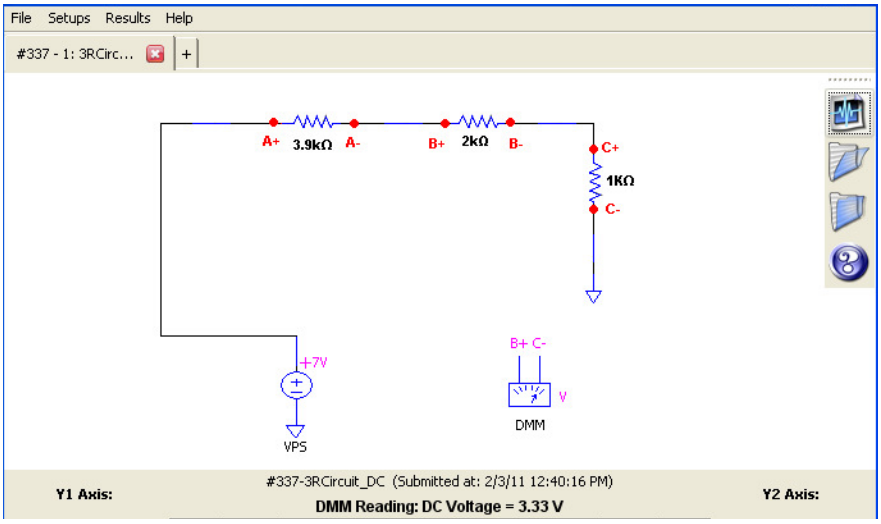
In the ELVIS iLab 5.0 to measure resistance across a component or branch, the student

will set **Mode** to *Resistance* and **Range** set to the expected range. Meanwhile +**Terminal** and **-Terminal** must be set to define the two ends of the branch whose resistance is being measured. Any connection that falls within the circuit branch must have its status set to *Short* while all other connections must have their status set to *Open* (Figure 3.12a). The student will obtain an appropriate reading (Figure 3.12b). Similar steps would be followed to measure the capacitance or inductance of a component or branch.

Measuring voltage



(a) Set to measure voltage

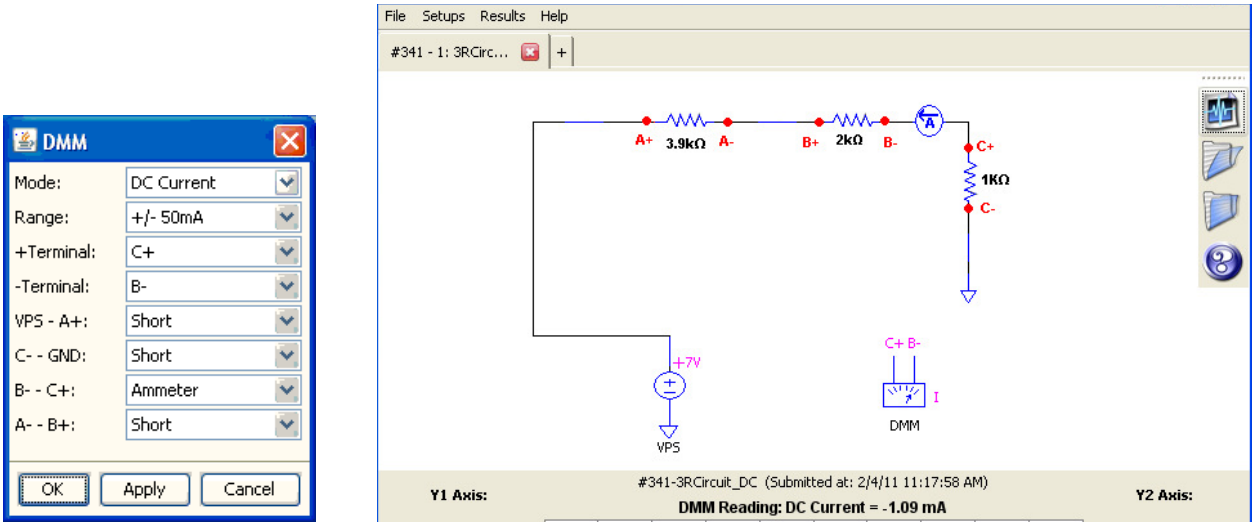


(b) The voltage drop across B+ - C- is measured to be 3.33 V

Figure 3.13: Measuring voltage with the digital multimeter in ELVIS iLab 5.0

To measure the voltage drop across a component or branch, the student must set **Mode** to *Voltage*, **Range** to an appropriate range and +**Terminal** and **-Terminal** to the two ends that define the component or branch being measured. In voltage measurements, the entire circuit must be connected to get an accurate reading; therefore, the student will need to set all the status of all connections to *Short* (Figure 3.13a). In this example, the voltage drop across B+ - C- is approximately 3.33 V (Figure 3.13b).

Measuring current



(a) Set to measure current

(b) The current is measured to be -1.09 mA

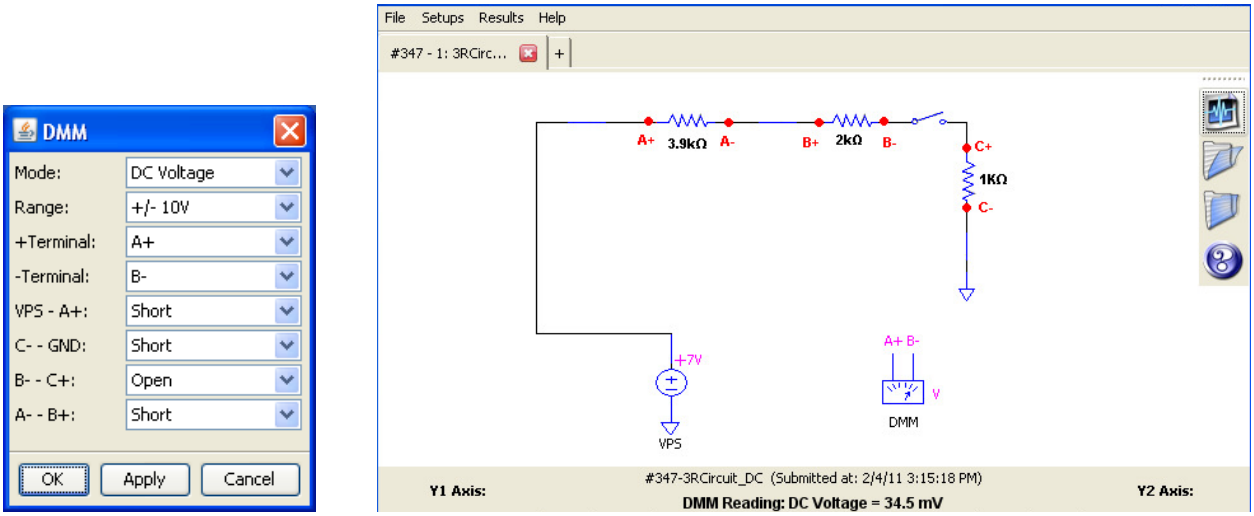
Figure 3.14: Measuring current with the digital multimeter in ELVIS iLab 5.0

For current measurements, the student must set **Mode** to *Current* and **Range** to an appropriate range. Furthermore, the student must break one of the connections and insert an ammeter there. The student must set +**Terminal** and **-Terminal** to the '+' and '-' ends of this connection respectively and also make sure to set this connection to *Ammeter* while setting the status of all connections to *Short* (Figure 3.14a). In this example, the current through the circuit as measured between terminals C+ and B- is approximately -1.09 mA (Figure 3.14b).

Wrongly configured circuits

The setup for the digital multimeter mimics the same behavior as a hands-on laboratory experience. Therefore, the Lab Client does not correct or generate alerts for any parameters set by the student. The student is responsible for checking that the digital multimeter parameters are correct. In the example show in Figure 3.15, the student has tried to measure

the voltage drop across A+ and B-. However, the connection between B- and C+ is *Open* so the entire circuit is open and the digital multimeter returns a voltage of nearly 0 V.



(a) Incorrect parameters (b) An erroneous measurement

Figure 3.15: A measurement with incompatible parameters

The ELVIS hardware already guards against a short-circuit and high voltages through an internal protection board [10] so no additional checks were built into the Lab Client or Lab Server.

3.5 Advanced graphs

3.5.1 Description

The ability to create additional axes and custom graphs in the ELVIS iLab Lab Client ensures that students will be able to visualize data from similar experiments on the same graph within the iLab platform. The Results Panel no longer has direct control over the axes, graphs and the grid; instead, a new class has been created to manage these components. This new manager allows for axes and user-created graphs to be removed and added. The Lab

Client also gets additional UI enhancements - a button and a dialog window - so that users can choose and control which axes to display and create their own custom graphs.

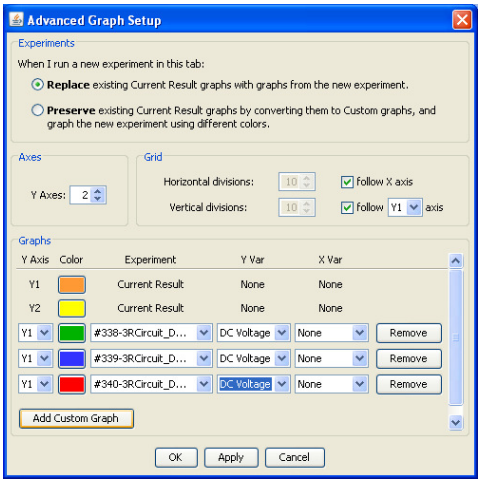
3.5.2 Lab Client Code Modification

The `GraphCanvasManager` is the new handler and manager for the axes, graphs and grids of the Results Panel. The axes are each represented as an `Axis` object, the graphs as `Graph` objects and the grid as a `Grid` object. A `Graph` consists of one independent axis and one dependent axis. In the ELVIS iLab 4.0, the Results Panel had two dependent axes, one independent axis and two graphs - each using the same independent axis and a different dependent axis. In the ELVIS iLab 5.0, the `GraphCanvasManager` maintains separate lists of `Axis` and `Graph` objects. Therefore, the `GraphCanvasManager` can maintain more than two dependent axes and two graphs. The `GraphCanvasManager` has methods for retrieving, adding and deleting an axis from and to a graph as well as removing axes and graphs from its lists altogether.

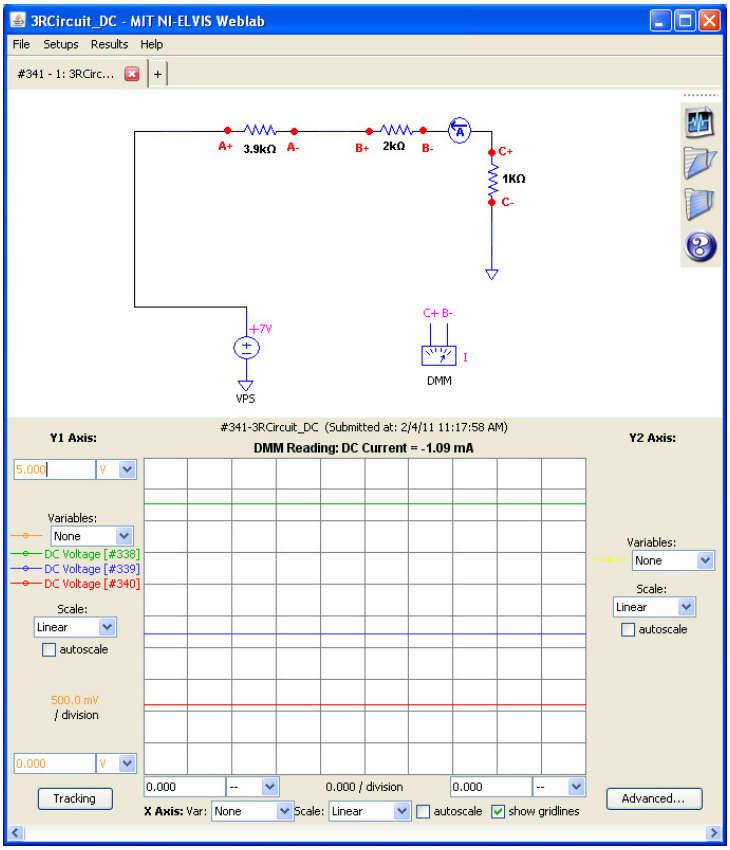
The `GraphSetupDialog` provides a dialog window as the user interface to the new capabilities. Users are able to select how many dependent axes to display, create custom graphs and configure graphing options. When creating custom graphs, users select an independent variable, a dependent variable and the Y-axis on which to graph the dependent variable. If the user creates custom graphs whose independent variables have different units, the user will receive a warning that the grid will not display the units next to the independent axis. The user can select variables from other experiments executed within the same session or from previous sessions. This dialog window appears when the user clicks a newly added button labeled **Advanced** in the bottom-right corner of the results panel.

3.5.3 Demonstration

In this example, the student has measured the current in Section 3.4.4 and also the voltage across each resistor. The student would like to plot all three voltages on the same grid. The student clicks the **Advanced** button to launch the **Advanced Graph Setup** window (Figure 3.16a). Under **Axes**, the user can increase the number of Y-axes up to ten. Under **Graphs**, the user can create custom graphs by clicking **Add Custom Graph** and then select an experiment, a dependent and an independent variable from that experiment, a color for the plot and the Y-axis to use to graph the relationship. The student selects the voltage measurement experiments and selects *DC Voltage* under **Y Var**. After the student clicks **Apply** or **OK**, the new axes and graphs are added to the results panel (Figure 3.16b).



(a) Advanced Graph Setup dialog window



(b) Grid with custom added graphs

Figure 3.16: Adding additional axes and custom graphs in the ELVIS iLab 5.0

3.6 Summary

ELVIS iLab 5.0 has incorporated five new features that improve both the student experience and the learning capabilities of the platform. Students have access to more intuitive controls for entering numerical data, manipulate experiments in a tabbed environment, access experiments executed earlier, create custom graphs using variables from multiple experiments and make decisions on the status of connections for a digital multimeter experiment.

The next chapter will present conclusions from the development of the ELVIS iLab 5.0 and recommendations for future development of the ELVIS-based iLabs and improvements in their usage.

Chapter 4

Conclusions and Recommendations

This chapter will begin with conclusions gained from the development of the ELVIS iLab 5.0 and highlight the impact of the new features to the experience and learning opportunities of students that will use this new client. Then, there will be a presentation of recommendations for further growth of the ELVIS iLab. These recommendations will address improving features from the ELVIS iLab 5.0, adding to the experiments currently available on the ELVIS iLab platform, preparing the latest version for use in courses and finally expansion of the iLab-Africa initiative.

4.1 Conclusions

ELVIS iLab 5.0 has improved the user experience and learning opportunities available within the Lab Client of the ELVIS-based iLabs.

The spinner widgets are an improvement on the user experience. Users are able to interact with a more flexible widget that they are already familiar with from other user interfaces.

The tabbed environment of the ELVIS leads to a better user experience and more learning opportunities. With tabs, users can create and execute multiple experiments and have them all open. A user could also replicate a slightly different version of an experiment by duplicating the tab and modifying the settings of the instruments in the experiment.

The Lab Client's new ability to retrieve a prior experiment spares the user from having to go to the Service Broker directly to retrieve the results of a past experiment. The user is able to view the result and experiment configuration together within the Lab Client. By combining this new feature with tabs, a user could load a previous experiment in a tab and execute the same or different experiment in another tab, allowing the user to compare and contrast results.

The new set of graphing features on the ELVIS iLab give users more ways to visualize their results. Users can now view more dependent variables and also create additional custom graphs to display on the grid.

The changes to the digital multimeter allow the ELVIS iLabs to better approximate the traditional hands-on laboratory experience. By bearing the responsibility of disconnecting sources and branches, students on the ELVIS iLab now learn the skills and have the experiences they were to learn from traditional hands-on laboratory experiences.

With these five new features, the ELVIS iLab is poised to be a better online laboratory for students.

4.2 Recommendations

There are five areas to target for improvement of the ELVIS-based iLabs: obtain accurate current measurements without hardware-dependent fixes, add needed functions to the Service

Broker, expand the range of experiments, deploy the latest version and have the iLab-Africa partners share their iLab experiments with local communities.

Firstly, this thesis has used a hardware-dependent offset to correct for the common mode voltage measurement error that occurs when measuring current. This is inadequate because this offset varies for each ELVIS workstation. It is important that a permanent hardware-independent solution be found so that the ELVIS iLab can be used on other ELVIS workstations.

Secondly, the Service Broker's web service methods should be upgraded to include a method to retrieve the IDs of a user's previous experiments without having to save them as client items. This would allow the Lab Client to list experiment IDs that were not saved as client items to the Service Broker.

Thirdly, expanding the range of experiments allows the ELVIS iLab to continue to enrich students. There are two ways to expand the range of experiments. The first way is to incorporate the four remaining instruments on the ELVIS - the dynamic signal analyzer, impedance analyzer, and the two-wire & three-wire current-voltage analyzers. Alternatively, third-party boards could be attached to the ELVIS mainframe to generate new types of experiments. A larger set of experiments allows educators to create complex experiments than may not be possible in a traditional hands-on laboratory.

Fourthly, the last version of the ELVIS deployed at MIT was version 1.2, which consisted only of a function generator and an oscilloscope. Since then, newer versions have added six new instruments - bode analyzer, arbitrary waveform generator, variable power supplies, digital multimeter, digital reader and digital writer - as well as switching capability and the new user interface features in the ELVIS iLab 5.0. The ELVIS iLab 5.0 is the latest version and therefore the ideal candidate for what should be deployed to students. Before it can be deployed, experiments from potential courses must first set up on the hardware and tested.

The tests will need to include load testing and a trial run by potential student users.

Finally, the partners in the iLab-Africa initiative have provided positive experiences as proxies for universities in developing countries. To generate more success of the iLab-Africa initiative, these universities need to share their ELVIS iLabs and other iLabs in their local communities and become local champions of iLabs. It would signify that iLabs has had a strong positive impact on student learning at these sub-Saharan institutions and suggest that iLabs in developing countries is a viable idea. Makerere University has already started by working with another Ugandan university, Busitema University while OAU in Nigeria has been in communication with two Nigerian universities about sharing its iLabs. More efforts to reach out into local communities by the African partners would testify to the ability of iLabs to provide an excellent alternative to the traditional laboratory experience in educational institutions around the world.

Appendix A

XML Specification Documents

The Document Type Definition (DTD) is useful for declaring the valid markup for any type of XML document. There is a DTD for each of the three XML documents used by iLabs: the Lab Configuration, Experiment Specification and Experiment Result. For each XML document, the DTD and a sample valid file have been included.

A.1 Lab Configuration

Listing A.1: The Lab Configuration Document Type Definition

```
<!ELEMENT labConfiguration (setup*)>
<!ATTLIST labConfiguration lab CDATA #REQUIRED>
<!ATTLIST labConfiguration specversion CDATA #REQUIRED>
<!ELEMENT setup (name, description, imageURL, mode+)>
<!ATTLIST setup id CDATA #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT imageURL (#PCDATA)>
<!ELEMENT mode (terminal*)>
<!ATTLIST mode type (TD|FD|DC) #REQUIRED>
<!ATTLIST mode enabled (true | false) #REQUIRED>
<!ELEMENT terminal (label, pixelLocation?,subCOM*,connection*, ((source+,
   postProcessOptions?) |(enabledModes, file*))?)>
<!ATTLIST terminal instrumentType (FGEN | SCOPE | ARB0 | VPSPos | VPSNeg |
   DOUT | COM | BODE |DMM) #REQUIRED>
<!ATTLIST terminal instrumentClass (input | output | control | switch) #
   IMPLIED>
<!ATTLIST terminal instrumentNumber CDATA #REQUIRED>
<!ATTLIST terminal setupTermID CDATA #REQUIRED>
<!ATTLIST subCOM subCOMType CDATA #REQUIRED>
<!ATTLIST subCOM instrumentNumber CDATA #REQUIRED>
<!ATTLIST subCOM setupTermID CDATA #REQUIRED>
```

```
<!ATTLIST connection name CDATA #REQUIRED>
<!ATTLIST connection x CDATA #REQUIRED>
<!ATTLIST connection y CDATA #REQUIRED>
<!ATTLIST connection horizontal CDATA #REQUIRED>
<!ATTLIST connection switch CDATA #REQUIRED>
<!ELEMENT subCOM (label)>
<!ELEMENT connection (#PCDATA)>
<!ELEMENT label (#PCDATA)>
<!ELEMENT pixelLocation (x, y)>
<!ELEMENT x (#PCDATA)>
<!ELEMENT y (#PCDATA)>
<!ELEMENT source (pixelLocation)>
<!ATTLIST source name CDATA #REQUIRED>
<!ATTLIST source channel CDATA #REQUIRED>
<!ELEMENT postProcessOptions (#PCDATA)>
<!ELEMENT enabledModes (#PCDATA)>
<!ELEMENT file (name, description, URL, length, recSamplingRate,
   recTotalSamples)>
<!ATTLIST file WAVID CDATA #REQUIRED>
<!ELEMENT URL (#PCDATA)>
<!ELEMENT length (#PCDATA)>
<!ELEMENT recSamplingRate (#PCDATA)>
<!ELEMENT recTotalSamples (#PCDATA)>
```

Listing A.2: An example An example Lab Configuration XML

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE labConfiguration SYSTEM "http://localhost/xml/labConfiguration.
   dtd">
<labConfiguration lab="NI ELVIS iLab" specversion="5.0">
  <setup id="4">
    <name>DemoCircuit</name>
    <description>A demonstration circuit supporting VPS and DMM</
       description>
    <imageURL>http://maui.mit.edu/images/setups/43mycircuit.GIF</imageURL>
    <mode type="TD" enabled="false"/>
    <mode type="FD" enabled="false"/>
    <mode type="DC" enabled="true">
      <terminal instrumentType="VPSPos" instrumentClass="control"
         instrumentNumber="1" setupTermID="12">
        <label>VPS</label>
        <pixelLocation>
          <x>0</x>
          <y>17</y>
        </pixelLocation>
      </terminal>
      <terminal instrumentType="DMM" instrumentClass="control"
         instrumentNumber="2" setupTermID="18">
        <label>DMM</label>
        <pixelLocation>
          <x>2</x>
```

```
          <y>2</y>
        </pixelLocation>
        <connection name="VPS - A+" x="-30" y="9" horizontal="True" switch=
           "24"/>
        <connection name="C- - GND" x="268" y="110" horizontal="False"
           switch="21"/>
        <connection name="B- - C+" x="240" y="9" horizontal="True" switch="
           22"/>
        <connection name="A- - B+" x="120" y="9" horizontal="True" switch="
           23"/>
      </terminal>
    </mode>
  </setup>
</labConfiguration>
```

A.2 Experiment Specification

Listing A.3: The Experiment Specification Document Type Definition

```
<!ELEMENT experimentSpecification (setupID, terminal+)>
<!ATTLIST experimentSpecification lab CDATA #REQUIRED>
<!ATTLIST experimentSpecification specversion CDATA #REQUIRED>
<!ELEMENT setupID (#PCDATA)>
<!ATTLIST setupID mode (TD|FD|DC) #REQUIRED>
<!ELEMENT terminal (function+)>
<!ATTLIST terminal instrumentType (FGEN|SCOPE|ARB0|VPSPos|VPSNeg|DOUT|DMM|
   BODE|COM) #REQUIRED>
<!ATTLIST terminal instrumentClass (input| output| control ) #IMPLIED>
<!ATTLIST terminal instrumentNumber CDATA #REQUIRED>
<!ATTLIST terminal setupTermID CDATA #REQUIRED>
<!ELEMENT function ((waveformType, frequency, amplitude, offset) |
                    (scope+, samplingRate, samples, trigger) |
                    (mode, arbSamplingRate, arbSamples, ((frequency,
                       amplitude, offset, phase, dutyCycle?) |
                    (waveform) |
                    (fileID))) |
                    (value) |
                    (byte,cycles,inputbits, outputbits) |
                    (udm, rangeid,position) |
                    (subCOM, label)|
                    (startFreq, stopFreq, stepsPerDec, inputAmp))>
<!ATTLIST function type (WAVEFORM | SAMPLING | ARB | BODE | VPSFunction |
   DOUTFunction | DMMFunction | subCOM) #REQUIRED>
<!ATTLIST function setupTermID CDATA #IMPLIED>
<!ELEMENT waveformType (#PCDATA)>
<!ELEMENT frequency (#PCDATA)>
<!ELEMENT amplitude (#PCDATA)>
<!ELEMENT offset (#PCDATA)>
<!ELEMENT scope (name?, source, paSpec, paDist)>
```

```
<!ATTLIST scope channel (A | B) #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT source (#PCDATA)>
<!ELEMENT paSpec EMPTY>
<!ATTLIST paSpec perform (true | false) #REQUIRED>
<!ELEMENT paDist EMPTY>
<!ATTLIST paDist perform (true | false) #REQUIRED>
<!ELEMENT samplingRate (#PCDATA)>
<!ELEMENT samples (#PCDATA)>
<!ELEMENT trigger (source, slope?, level?)>
<!ELEMENT slope (#PCDATA)>
<!ELEMENT level (#PCDATA)>
<!ELEMENT mode (#PCDATA)>
<!ELEMENT arbSamplingRate (#PCDATA)>
<!ELEMENT arbSamples (#PCDATA)>
<!ELEMENT phase (#PCDATA)>
<!ELEMENT dutycycle (#PCDATA)>
<!ELEMENT waveform (#PCDATA)>
<!ATTLIST waveform dt CDATA #REQUIRED>
<!ELEMENT fileID (#PCDATA)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT byte (#PCDATA)>
<!ELEMENT inputbits (#PCDATA)>
<!ELEMENT outputbits (#PCDATA)>
<!ELEMENT cycles (#PCDATA)>
<!ELEMENT udm (#PCDATA)>
<!ELEMENT rangeid (#PCDATA)>
<!ELEMENT position (#PCDATA)>
<!ELEMENT startFreq (#PCDATA)>
<!ELEMENT stopFreq (#PCDATA)>
<!ELEMENT stepsPerDec (#PCDATA)>
<!ELEMENT inputAmp (#PCDATA)>
<!ELEMENT subCOM (#PCDATA)>
<!ELEMENT label (#PCDATA)>
```

Listing A.4: An example Experiment Specification XML for a digital domain experiment

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE experimentSpecification SYSTEM "http://maui.mit.edu/xml/
   ExperimentSpecification.dtd">
<experimentSpecification lab="MIT NI-ELVIS Weblab" specversion="0.1">
  <setupID mode="DC">4</setupID>
  <terminal instrumentType="VPSPos" instrumentNumber="1">
    <name download="true">VPS</name>
    <function type="VPSFunction">
      <value>5.000</value>
    </function>
  </terminal>
  <terminal instrumentType="DMM" instrumentClass="input" instrumentNumber="
     2">
    <name download="true">DMM</name>
```

```
    <function type="DMMFunction">
      <udm>DC Voltage</udm>
      <rangeid>2</rangeid>
      <position>11011000011000000000001100</position>
    </function>
  </terminal>
</experimentSpecification>
```

A.3 Experiment Result

Listing A.5: The Experiment Result Document Type Definition

```
<!ELEMENT experimentResult (datavector+)>
<!ELEMENT datavector (#PCDATA)>
<!ATTLIST experimentResult lab CDATA #REQUIRED>
<!ATTLIST experimentResult specversion CDATA #REQUIRED>
<!ATTLIST datavector name CDATA #REQUIRED>
<!ATTLIST datavector units CDATA #REQUIRED>
<!ATTLIST datavector scalable (true | false) #IMPLIED>
<!ATTLIST datavector type (vector | scalar) #IMPLIED>
```

Listing A.6: An example Experiment Result XML for a digital domain experiment

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE experimentResult SYSTEM "http://localhost/xml/experimentResult.
    dtd">
<experimentResult lab="NI ELVIS iLab" specversion="5.0">
  <datavector name="DC Voltage" units="V" scalable="true" type="scalar">
     2.43260338624319</datavector>
  <datavector name="Range?" units="range" scalable="false" type="vector">0
     0</datavector>
  <datavector name="DMMExtra" units="none" scalable="false" type="vector">0
      0</datavector>
  <datavector name="DOUT" units="bool" scalable="false" type="vector"/>
</experimentResult>
```

Appendix B

Digital Multimeter Switching

The digital multimeter uses an integrated switching module composed of relays in order to be able to take measurements across multiple points in a circuit. Currently, 25 different combinations of points can be measured for voltage, resistance, capacitance, inductance or current.

The digital multimeter conducts measurements through two pairs of terminals: `VOLTAGE HI` and `VOLTAGE LO` for voltage measurements while `CURRENT HI` and `CURRENT LO` is used for all other measurements. Consequently, only one pair may be on at a time.

The exact configuration of which relays are connected and which are disconnected depends on both the type of measurement the digital multimeter will be making and which points in the circuit it will be measuring. Figure B.1 and Table B has been provided to summarize each connection and an interpretation of how it controls the relays in the circuit as well as a full table on all the switches in the circuit.

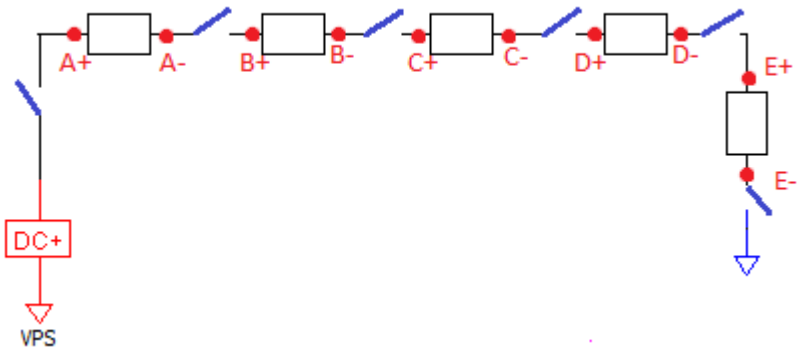


Figure B.1: A circuit supporting five components with all relays set to open

Index	**Label**	**Interpretation**
0	GND	use ground node
1	VPS	use Variable Power Supply
2	*unused*	
3	VLO	use `VOLTAGE LO`
4	VHI	use `VOLTAGE HI`
5	ILO	use `CURRENT LO`
6	IHI	use `CURRENT HI`
7	GND+	connect ground to `VOLTAGE HI` or `CURRENT HI`
8	VPS-	connect VPS to `VOLTAGE LO` or `CURRENT LO`
9	A+	connect A+ to `VOLTAGE HI` or `CURRENT HI`
10	A-	connect A- to `VOLTAGE LO` or `CURRENT LO`
11	B+	connect B+ to `VOLTAGE HI` or `CURRENT HI`
12	B-	connect B- to `VOLTAGE LO` or `CURRENT LO`
13	C+	connect C+ to `VOLTAGE HI` or `CURRENT HI`
14	C-	connect C- to `VOLTAGE LO` or `CURRENT LO`
15	D+	connect D+ to `VOLTAGE HI` or `CURRENT HI`
16	D-	connect D- to `VOLTAGE LO` or `CURRENT LO`
17	E+	connect E+ to `VOLTAGE HI` or `CURRENT HI`
18	E-	connect E- to `VOLTAGE LO` or `CURRENT LO`
19	I6	connect E- to ground
20	I5	connect D- to E+
21	I4	connect C- to D+
22	I3	connect B- to C+
23	I2	connect A- to B+
24	I1	connect VPS to A+

Table B.1: Interpretation of the 25 switches used by the digital multimeter

Bibliography

[1] S. Dillon. (2006, October 20) No Test Tubes? Debate on Virtual Science Classes. [Online]. Available: http://www.nytimes.com/2006/10/20/education/20online.html

[2] A. Maiti, "Automatic Evaluation of Student's Performance in Online Laboratories," *International Journal of Online Engineering*, vol. 5, pp. 1861–2121, 2010.

[3] V. J. Harward, J. A. del Alamo, S. R. Lerman, P. H. Bailey, J. Carpenter, K. DeLong, C. Felknor, J. Hardison, B. Harrison, I. Jabbour, P. D. Long, T. Mao, L. Naamani, J. Northridge, M. Schulz, D. Talavera, C. Varadharajan, S. Wang, K. Yehia, R. Zbib, and D. Zych, "The iLab Shared Architecture: A Web Services Infrastructure to Build Communities of Internet Accessible Laboratories," *Proceedings of the IEEE*, vol. 96, no. 6, pp. 931–950, June 2008.

[4] K. Theroux, "Linking African Universities with MIT iLabs," *Carnegie Reporter*, vol. 3, no. 4, Spring 2006. [Online]. Available: http://carnegie.org/publications/carnegie-reporter/single/view/article/item/153/

[5] A. Jiwaji, J. Hardison, A. K. P., S. S. Tickodri-Togboa, A. Mwambela, V. J. Harward, J. A. del Alamo, B. Harrison, and S. Gikandi, "Collaborative Development of Remote Electronics Laboratories: The ELVIS iLab," in *Proceedings of the ASEE Annual Conference and Exposition.* ASEE, June 2009.

[6] National Instruments. (2009, April) What is NI ELVIS? [Online]. Available: http://zone.ni.com/devzone/cda/tut/p/id/8599

[7] B. J. Harrison, "Expanding the Capabilities of the ELVIS iLab Using Component Switching," Master's thesis, Massachusetts Institute of Technology, 2008.

[8] J. Hardison, D. Zych, del Alamo J. A., H. V. J., S. Lerman, S. M. Wang, K. Yehia, and C. Varadharajan, "The Microelectronics WebLab 6.0–An Implementation Using Web Services and the iLab Shared Architecture," in *International Conference on Engineering Education and Research 2005.* Tainan, Taiwan: INEER, March 2005.

[9] S. Gikandi, "ELVIS iLab: A Flexible Platform for Online Laboratory Experiments in Electrical Engineering," Master's thesis, Massachusetts Institute of Technology, 2006.

[10] National Instruments. (2006, September) NI Educational Laboratory Virtual Instrumentation Suite (NI ELVIS). [Online]. Available: http://zone.ni.com/devzone/cda/tut/p/id/3711

[11] A. Jiwaji, "Modular Development of an Educational Remote Laboratory," Master's thesis, Massachusetts Institute of Technology, 2008.

[12] R. Shroff, "A Versatile Internet-Accessible Electronics Workbench with DC Domain Experimentation and Troubleshooting Capabilities," Master's thesis, Massachusetts Institute of Technology, 2009.

[13] H. Soumare, "Introduction of Digital Experimentation Capabilities on the ELVIS iLab Platform," Master's thesis, Massachusetts Institute of Technology, 2009.