A Versatile Internet-Accessible Electronics Workbench with DC Domain Experimentation and Troubleshooting Capabilities

by

Rahul Shroff

S.B., Massachusetts Institute of Technology (2008)

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

June 2009

Author . . . .
Department of Electrical Engineering and Computer Science
May 22, 2009

Certified by . . . .
V. Judson Harward
Associate Director, Center for Educational Computing Initiatives
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

A Versatile Internet-Accessible Electronics Workbench with DC Domain Experimentation and Troubleshooting Capabilities

by

Rahul Shroff

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 2009, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

iLabs are online laboratories that give students access to various experimental setups enabling them to conduct experiments based on real equipment via the Internet, remotely from any part of the world. The MIT iLab Project is dedicated to the proposition that iLabs can enrich science and engineering education by greatly expanding the range of experiments that students are exposed to. Using iLabs students complement their theoretical calculations and results with real data, providing them with a better understanding of a wide range of engineering concepts. Most recently, the iLab Project has focussed on building remote laboratories around the National Instruments Educational Laboratory Virtual Instrumentation Suite (ELVIS), a cost-effective, all-in-one electronics workstation. This thesis documents my efforts in extending the ELVIS iLab framework by enabling the investigation of the Direct Current domain through the addition of a new instrument, the Digital Multimeter. Using an augmented version of switching, this new instrument provides students with real-time, dynamic circuit testing and troubleshooting capabilities, unprecedented in an iLab. This significantly enhances an iLab's value as a versatile educational tool and represents a considerable step forward in bridging the gap between conventional and remote laboratories.

Thesis Supervisor: V. Judson Harward
Title: Associate Director, Center for Educational Computing Initiatives

Acknowledgments

I would like to begin by expressing my gratitude towards my thesis advisor, Dr. Judson Harward for giving me the opportunity to work on this project. His invaluable guidance, insightful suggestions and constant encouragement were instrumental in ensuring that my research reached its full potential. I would also like to thank Professor Jesus A. del Alamo whose contribution to the MIT iLab Project over the last eleven years has been monumental. His vision and technical expertise were indispensible in shaping the final product.

I am extremely grateful to James Hardison for the inspiration and support he has given me during the last year. His experience and willingness to help greatly facilitated my progress. I would also like to thank the rest of the MIT iLabs team - Phil Bailey, Kimberly DeLong, Meg Westlund and Maria Karatzas. Their motivation and assistance made my M. Eng. year thoroughly enjoyable.

The iLabs Project has enjoyed the generosity of the Carnegie Corporation. I would like to thank them for having faith in our ability to steadily deliver.

I would like to thank my predessor Adnaan Jiwaji, for introducing me to the project and for being a great mentor and friend during this last year. Bryant Harrison also helped me transition into the project. The help, advice and encouragement I received from Hamidou Soumare, my partner on the project, guaranteed this thesis' timely completion.

The iLabs teams at Makerere University and the University of Dar-es-Salaam have my sincerest thanks for being extremely animated and hospitable hosts. The ideas and challenges they presented always kept me inspired and motivated.

The last five years would have been extremely different without the company of my closest friends. I thank Cankut Durgun, Firat Ileri, Kaan Karamanci, Asish Misra and Spyros Zoumpoulis for the camaraderie, moral support and innumerable memories they have provided me with during my tenure at MIT.

Finally, I am deeply indebted to my parents and brother for the unconditional love and support they have shown me throughout my life. My parents have inculcated in me the importance of hard work, dedication and discipline. These values and my parents' unflagging belief in my ability to succeed have stood me in good stead throughout my MIT career.

Contents

1 Introduction **13**

1.1 Background on the iLabs Project 14

1.2 iLabs in Sub-Saharan Africa 15

1.3 Frequent Concerns about iLabs 16

1.4 Overview of Thesis 17

2 Description of Components **19**

2.1 The iLabs Shared Architecture 19

2.1.1 Lab Client 20

2.1.2 Service Broker 21

2.1.3 Lab Server 23

2.1.4 Data flow within the iLabs Shared Architecture 23

2.2 National Instruments Educational Laboratory Virtual Instrumentation Suite (NI-ELVIS) 24

2.3 LabVIEW 26

3 Previous Versions of ELVIS iLab and Motivation for ELVIS v4.0 **29**

3.1 ELVIS v1.0 30

3.2 ELVIS v2.0 31

3.3 Unification of parallel development efforts in the iLabs project 33

3.4 Motivation for ELVIS v4.0 35

4 ELVIS v4.0 Detailed Design **37**

4.1 XML Specification Documents 37
4.1.1 Lab Configuration 38
4.1.2 Experiment Specification 38
4.1.3 Experiment Result 39
4.2 Lab Server 39
4.2.1 LabVIEW 41
4.2.2 Experiment Engine 46
4.2.3 Adapting Switching for use with the Digital Multimeter 47
4.2.4 Validation Engine 53
4.3 Weblab Client 53
4.3.1 General Structure of the Client 54
4.3.2 Changes made to client to incorporate the DMM 54
4.4 Testing and Deployment 57

5 Closing Remarks **63**
5.1 Contributions 63
5.2 Recommendations for Future Development 64
5.3 Status of iLabs in Africa 65

A Resource Conflicts **67**

B Lab Configuration XML document **69**

C Lab Configuration DTD File **73**

D Experiment Specification XML Document **75**

E Experiment Specification DTD File **77**

F Experiment Result XML Document **79**

G Experiment Result DTD File **81**

List of Figures

1-1 Timeline of iLabs development 15

2-1 iLabs Shared Architecture 19

2-2 ELVIS v1.0 Lab Client 20

2-3 Screenshot of MIT iLab Service Broker. This page indicates the groups a user is a member of and the iLab clients the user has access to. . . 21

2-4 XML transfer within the ISA 24

2-5 The National Instruments Educational Laboratory Virtual Instrumentation Suite 25

2-6 Screenshot of the NI-ELVIS Instrument Launcher which provides users access to the 12 instruments on the ELVIS 26

3-1 ELVIS iLab Development Trajectory 29

3-2 Screenshot of ELVIS v2.0 Lab Client. Each red box represents a switchable component 32

3-3 OpAmp circuit schematic simplified using ELVIS v2.0 switching capability 33

3-4 Screenshot of ELVIS v3.5 Client representing all exposed functionality. The COM dialogue box is seen on the right and represent how a user chooses a switchable component. 34

4-1 ELVIS iLab Development Trajectory 40

4-2 Flowchart illustrating code heirarchy in ELVIS v3.5 `Main2.vi` 41

4-3 Screenshot of GUI for Digital Multimeter 42

4-4 Screenshot of LabVIEW code for `DMM.vi` 43

4-5 Flowchart illustrating code heirarchy in ELVIS v4.0 `Main2.vi` 45

4-6 A simple RLC circuit 48

4-7 RLC circuit with switching to enable multipoint measurement 49

4-8 Illustrative switch string 50

4-9 Switch string creation in the Lab Server 53

4-10 RLC Circuit Schematic 55

4-11 Dialog Box for DMM 58

4-12 Phase plot for the frequency characterisation of an RLC circuit 59

4-13 Result for measurement of resistance across the capacitor in DC mode. The capacitor behaves as an open circuit in DC and hence the measured value is infinity. 60

4-14 Result for the measurement of AC current in the series circuit. Since there is no time varying source, the value is zero. 61

4-15 Result for measurement of DC voltage across the capacitor. The input voltage is $10V$. 62

A-1 Resource Conflicts on ELVIS I [9] 67

List of Tables

4.1 Description of each bit in Switch String 51

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

A student's curiosity is often left unsatisfied with a classroom education. In most cases, science and engineering courses in schools and colleges are taught with a simultaneous laboratory component. These laboratory experiments provide students with important hands-on, practical experience to give more credibility to the theories and methods taught in class. Further, they equip students with a variety of skills necessary for them to be successful in the real world. A purely theoretical education would shield students from the inevitable discrepancies they would face when applying that knowledge to solve actual problems they wish to tackle after their education. These inconsistencies are often a result of noisy data, inefficiencies of components and sometimes even a failure to setup the problem correctly. An education's laboratory component provides students with real data and enables them to deal with these issues in an amicable setting so that they can overcome these problems more competently in the future.

However, students often don't have sufficient access to state of the art laboratories. This is because traditional lab facilities are expensive to set up and maintain. Further, large class sizes and limitations in class time and equipment availability have led to many institutions around the world neglecting this necessary component of their students' education.

Moreover, traditional laboratories where available, rarely reach their full potential. This is because, for the most part, they are available to students only during regular business hours. This presents a considerable inconvenience for students, who are often so engaged in setting up and troubleshooting an experiment that they are not left with enough time to absorb the crucial concepts the assignment was designed to demonstrate.

1.1 Background on the iLabs Project

In 1998, Professor Jesus A. del Alamo created the concept of iLabs - an attempt to bridge the challenges and inefficiencies of using traditional laboratories. iLabs are online laboratories that give students access to various experimental setups thus enabling them to conduct experiments based on real equipment via the Internet, remotely from any part of the world. Such a setup bypasses a large number of the typical problems of conventional laboratories. Most importantly, iLabs greatly reduce the cost of setting up a laboratory for each user because only one piece of equipment that can be shared by multiple users is required. As a result, students don't have to wait in long queues to use equipment because they can access the setups at their convenience from their own homes. Thus, while iLabs provide unlimited access, only a few institutions need to invest in costly equipment. This immense scalability of the iLabs framework lends itself to the project's vision: A global network of institutions creating and sharing cross-disciplinary iLabs for use by each others'students and faculty.

Starting with the Microelectronics Device Characterisation lab in 1998, several iLabs in diverse disciplines have been developed at MIT (Figure 1-1). These and other remote labs have been used in 18 universities throughout Africa, Asia, Europe and the United States [4]. In addition to using iLabs developed at MIT, many of our partner universities have created iLabs better suited for use in their own curricula. With the help of these universities, the remote laboratory concept is spreading throughout the world as a versatile educational tool, especially in areas where traditional lab facilities are lacking.

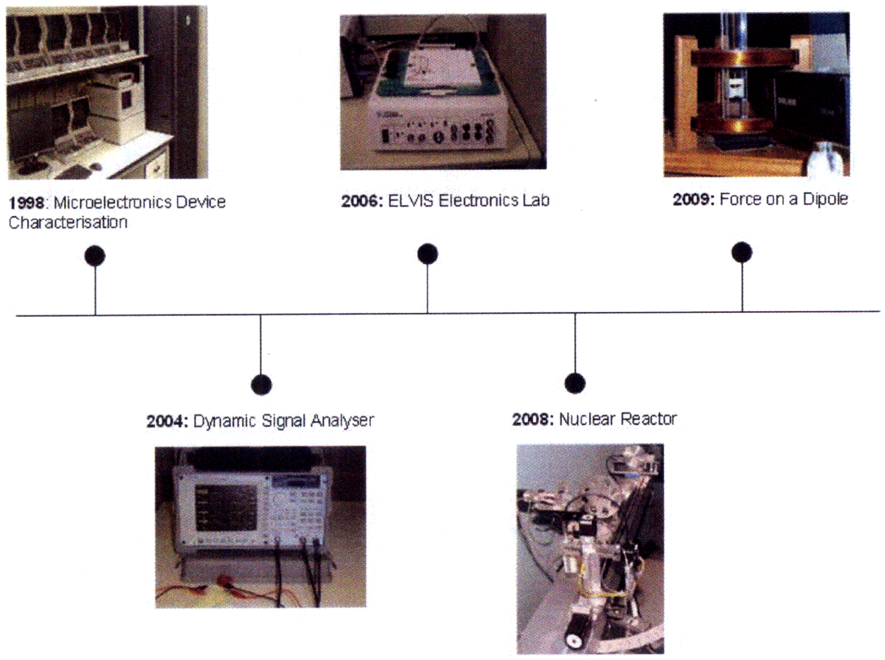

1998: Microelectronics Device Characterisation
2006: ELVIS Electronics Lab
2009: Force on a Dipole
2004: Dynamic Signal Analyser
2008: Nuclear Reactor

Figure 1-1: Timeline of iLabs development

1.2 iLabs in Sub-Saharan Africa

It is easy to see that the scalability and economically efficient nature of iLabs makes it an ideal concept for students in developing countries around the world, specifically in Africa [2]. Several African countries are on the verge of an internet revolution with some (e.g. Ghana) even encouraging the purchase and retail of Internet bandwidth. Moreover, African institutions are eager to forge partnerships with foreign establishments of higher education. This tremendous growth in the accessibility of the internet and enthusiasm to foster new relationships is conducive to setting up iLabs in these countries. Institutions in many African countries are unable to provide their students with the extent of lab experience that is necessary for effective learning. This is mostly due to large class sizes, financial impediments and sometimes even because of the inability to overcome the bureaucracy of the government. In these cases, iLabs can substitute for physical laboratories and provide students with some degree of familiarity with real experiments and results.

With this in mind the MIT iLabs Project, in conjunction with the Carnegie Corporation of NY, formed a partnership with three leading African Universities: Makerere

University (MUK) in Kampala, Uganda, Obafemi Awolowo University (OAU) in Ile-Ife, Nigeria, and University of Dar es Salaam (UDSM) in Dar es Salaam, Tanzania. Although this partnership began with the sharing of the Microelectronics Laboratory and other iLabs where the actual setup was housed at MIT, gradually our partner universities were able to duplicate some of these experimental setups at their own campuses to cope with the lack of a high bandwidth fibre-optic cable connection to the Internet. This in turn led to the replacement of the high-end equipment used by the Microelectronics Lab with a relatively low cost, all-in-one electronics workbench: the National Instruments Educational Laboratory Virtual Instrument Suite (ELVIS). An unintended, though highly beneficial side-effect of this transition was the emergence of the ELVIS as a common platform for development between these and other universities, leading to a more systematic, modular and somewhat generic apporach towards creating iLabs.

Work on the iLabs project in Africa has been extremely successful with students and faculty at the various partner universities taking a great interest in its development. Starting with Albert Lumu's visit in 2004, numerous developer exchanges have been made between the four universities resulting in the training of new team members and in gaining a general understanding of the needs and challenges of each university

1.3 Frequent Concerns about iLabs

The iLabs project is a means of non-traditional education that was conceived strictly as a complement to more traditional teaching methods. Like many other varieties of alternative education, it has inevitably faced its share of criticism. I would like to take this opportunity to clear some misconceptions about the iLabs concept, which from my conversations and interactions with people in the field, seem fairly widespread.

Although an iLab is web-based, the graphcial user interface (GUI) an iLab user interacts with is nothing but a virtual representation of a *real* experiment. Students use this GUI to configure and troubleshoot a *real*, physical apparatus and once the experiment is performed, can view *real* results. It is imperative to realise that iLabs do *not* perform any simulations. Users are presented with actual results, replete with real world considerations. Thus, iLabs represent a significant value-added over simulating programmes such as SPICE or ModelSim which merely provide textbook-like simulations of engineering concepts under ideal conditions. These programmes are indeed valuable in reinforcing concepts already learnt in a classroom, but they do little to expose students to how experiment components behave in the presence of imperfect conditions.

It is important to note, however, that iLabs are not a perfect replacement for the traditional laboratory. Students using solely iLabs are unable to physically experience wiring a circuit on a prototyping board, or gain the valuable skills of setup configuration. However, these skills can be acquired in a relatively cost effective manner. Once students have acquired the necessary proficiency, teachers can use iLabs as a substitute for the costly equipment required to analyse these setups. Thus iLabs are best suited as a supplement to exercises that may still include elementary hands-on laboratory assignments and as a replacement *only* in cases where no other options are available.

1.4 Overview of Thesis

The focus of my research has been to improve the ELVIS-iLab platform to provide students and faculty with a richer pedagogical experience. In this thesis I document my efforts in introducing DC-domain experimentation and troubleshooting capabilities into the iLabs framework.

In Chapter 2, I provide a description of various components constituting the iLab framework. These include the iLab Shared Architecture, the NI-ELVIS hardware and the LabVIEW programming environment.

Chapter 3 provides a high-level description of past developments on the ELVIS iLab. In this chapter I also examine what piece of functionality is exposed in each new version and how every iteration of the ELVIS iLab overcomes the limitations of its predecessors. Finally, this chapter provides some motivation for my contribution to the project and details how it improves the iLab user experience.

In Chapter 4 I provide a detailed description of the design for ELVIS v4.0. This chapter also discusses the testing and deployment of this new version.

Chapter 5 gives an udpate on the current status of the iLabs Africa partnership and the creation of the iLab Consortium. Finally, I enlist my contributions to the iLab Project and express some of the limitations that still exist in ELVIS v4.0. This sets the stage for a discussion on the future of iLabs and also recommendations for future work.

Chapter 2

Description of Components

2.1 The iLabs Shared Architecture

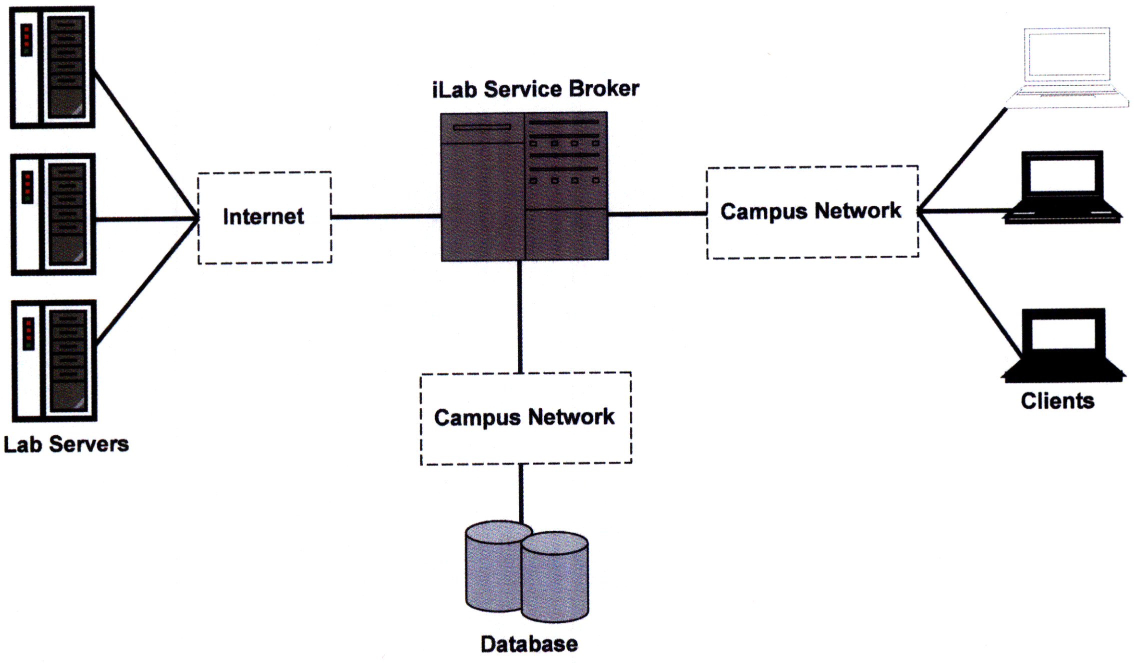


Figure 2-1: iLabs Shared Architecture

Early in the iLab development effort, a Java based web Client communicated directly with a server connected to the hardware being investigated. This server also provided an administrative interface, managing student accounts and logins. However, as the iLab project grew and more labs were created, this approach led to a divergence in development efforts with each lab having its own unique code structure. The iLabs shared architecture was introduced in 2002 in an effort to create a standardised infrastructure which could be used by developers around the world. [6]

As can be seen in Figure 2-1, the iLab architecture is split into three major components: a lab client, a service broker and a lab server which interact with each other using web-based service calls [3]. Each of these components is described in detail in this section.

2.1.1 Lab Client

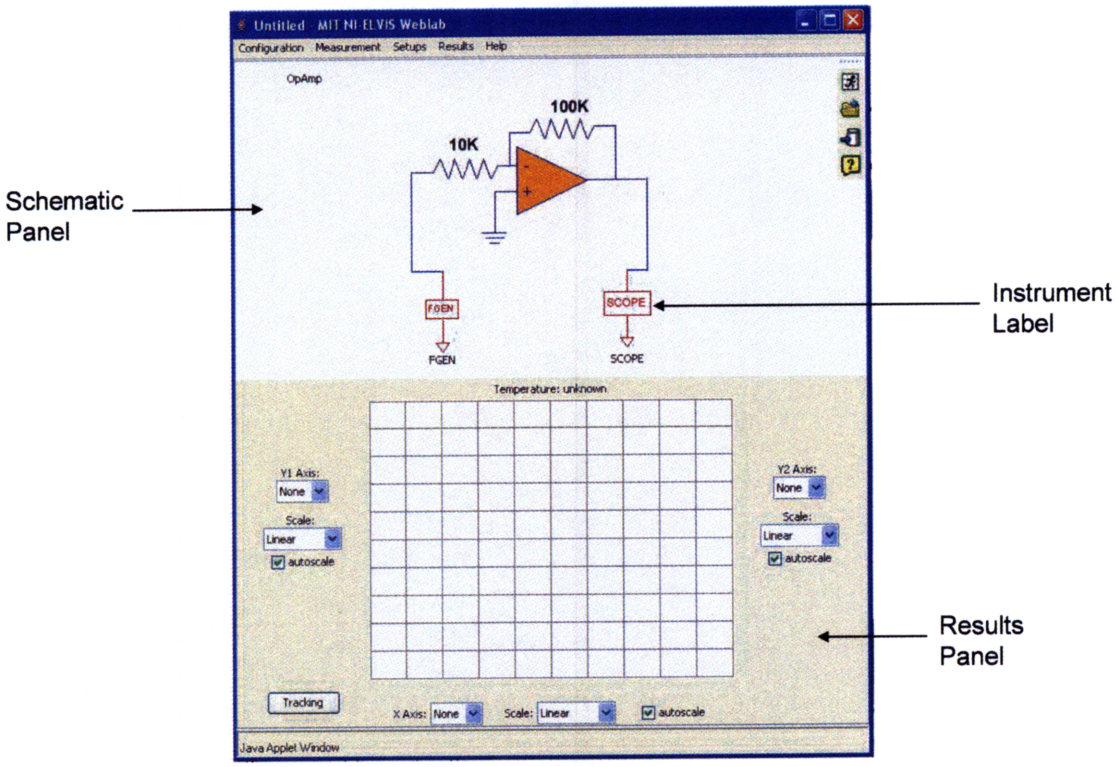


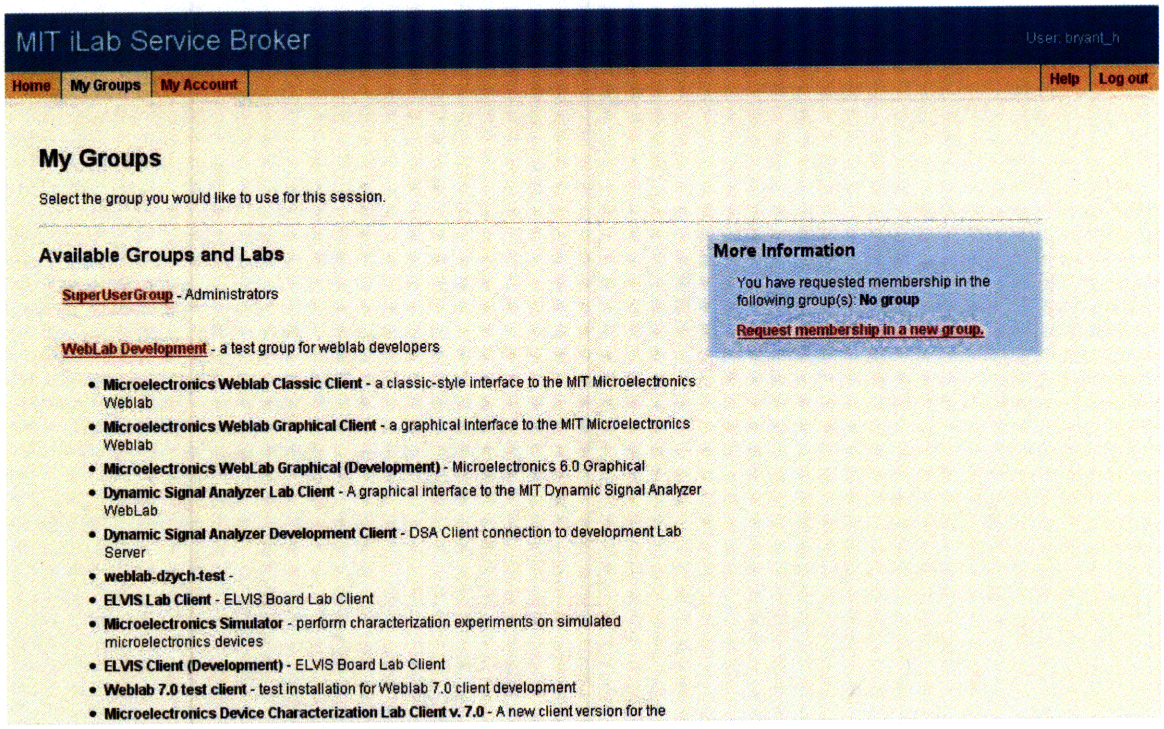
Figure 2-2: ELVIS v1.0 Lab Client

The client provides a graphical user interface through which a student accesses and interacts with an iLab. In essence, the client is nothing but a *virtual* representation of a *real* setup. As seen in Figure 2-2, the Client consists of primarily two components: the Schematic Panel and the Results Panel.

The Schematic Panel presents a graphical representation of the experiment being run. Students can configure the experiment to their liking by entering appropriate parameter values for each instrument visible on the panel.

The Results Panel plots the data vectors returned from a particular run of the assigned experiment. The graphing utility on the Results Panel allows students to concurrently plot two data vectors enabling easy comparison and analysis. Students are also provided with scaling and tracking facilities. In case users wish to further scrutinise the results, the Client provides an option to export the actual data vectors as a list of comma separated values.

2.1.2 Service Broker



My Groups

Select the group you would like to use for this session.

Available Groups and Labs

SuperUserGroup - Administrators

WebLab Development - a test group for weblab developers

- **Microelectronics Weblab Classic Client** - a classic-style interface to the MIT Microelectronics Weblab
- **Microelectronics Weblab Graphical Client** - a graphical interface to the MIT Microelectronics Weblab
- **Microelectronics WebLab Graphical (Development)** - Microelectronics 6.0 Graphical
- **Dynamic Signal Analyzer Lab Client** - A graphical interface to the MIT Dynamic Signal Analyzer WebLab
- **Dynamic Signal Analyzer Development Client** - DSA Client connection to development Lab Server
- **weblab-dzych-test** -
- **ELVIS Lab Client** - ELVIS Board Lab Client
- **Microelectronics Simulator** - perform characterization experiments on simulated microelectronics devices
- **ELVIS Client (Development)** - ELVIS Board Lab Client
- **Weblab 7.0 test client** - test installation for Weblab 7.0 client development
- **Microelectronics Device Characterization Lab Client v. 7.0** - A new client version for the

More Information

You have requested membership in the following group(s): **No group**

Request membership in a new group.

Figure 2-3: Screenshot of MIT iLab Service Broker. This page indicates the groups a user is a member of and the iLab clients the user has access to.

The Service Broker forms the core of the ISA and is the main point of differentiation between this and previous versions of the iLab architecture. The service broker is a web-based portal which plays the role of an intermediary between labs servers and clients connected to it. As we will see in Section 2.1.4 communication between lab servers and clients is limited to the exchange of XML documents. The service broker negotiates these exchanges, ensuring that each dispatched document reaches its intended destination. This is guaranteed by the fact that within the service broker, each client must be coupled to a particular lab server. Due to the fact that the service broker is laboratory independent, several lab servers and clients maybe be linked to one service broker.

In addition to overseeing these exchanges, the service broker also manages student accounts and groups which usually correspond to classes using the iLabs. An administator is provided with complete flexibility in handling group logistics. He may categorise students by section, course or year and specify permissions such that each iLab is accessible only to its intended audience. Finally, the service broker also stores experiment data such as setups, results, time and date information etc. which can be retrieved by administrators and students as needed. Although theoretically, one service broker is sufficient to manage every combination of lab server and client, typically, each institution (or course) provides its own service broker for greater efficiency and autonomy.

Two types of service brokers have been developed at MIT: the batched and interactive service brokers. The batched architecture is distinct in that users are presented with an experimental setup and asked to input the required parameters without actually utilising hardware resources until the definition is complete. Each student's experiment specification is then queued to be executed in the lab server, following a firt-in-first-out discipline. After execution, the results are conveyed to the client for display. In this model each student only utilises lab equipment for a couple of seconds and thus the batched service broker is ideal for significantly large classes. Due to the fact that a student's interaction with the physical equipment is limited in this version of the service broker, its bandwidth requirement per experiment run is low. However, this limited interaction is also a potential drawback of using the Batched Service Broker.

On the other hand, the interactive service broker gives students exclusive, real-time, dynamic access to laboratory equipment. Users shedule time-blocks during which solely they can access the lab equipment. While this unlimited control and flexibility can be a considerable improvement over the batched model it is not viable in situations involving large classes or when investigating time-intensive experiments. Furthermore, since the interactive service broker provides continuous access to equipment for students, its use is bandwidth intensive. These issues have led to the deployment of the Batched service broker with the ELVIS iLabs in East Africa, where low-bandwidth and unreliable power have made the use of the interactive architecture impractical.

2.1.3 Lab Server

The Lab Server is the part of the ISA which interacts directly with laboratory equipment. Specifications for each experiment, entered in the client by students, are sent to the lab server via the service broker. Analogously, once the experiment is performed the lab server sends the results back to the client through the service broker.

Another important function of the lab server is to serve as an administrative interface to the lab equipment. Lab administrators enter the desired configuration for each experiment by choosing which instruments they wish to include in a particular setup. Administrators are also given the opportunity to specify the allowable range of input values for each required parameter. Any values entered by the user outside this range are rejected. This ensures the safety of laboratory equipment and helps prevent potentially hazardous situations.

To facilitate the sharing of labs between institutions each lab server can interact with several service brokers. It must, however, be noted that each lab server is equipment specific, and as such, we need a different lab server to interact with each piece of hardware.

2.1.4 Data flow within the iLabs Shared Architecture

Information is exchanged between the various components of the ISA using three unique XML documents namely, `LabConfiguration.xml`, `ExperimentSpecification.xml` and `ExperimentResult.xml`. One complete run of an iLab experiment involves the creation, transfer and deciphering of each of these documents. This exchange is depicted in Figure 2-4.

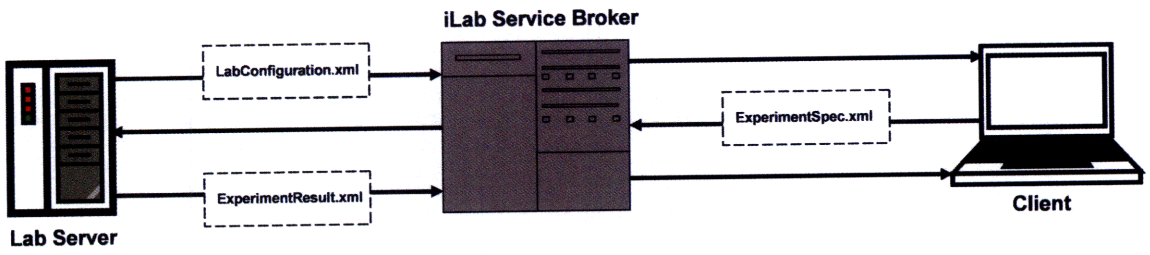


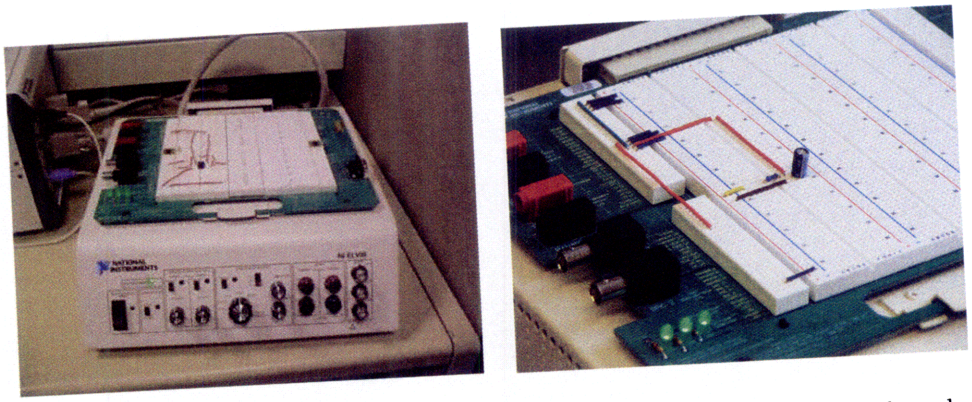
Figure 2-4: XML transfer within the ISA

Initially, when a user selects the lab he wishes to run in the service broker, the `LabConfiguration.xml` document is created by the lab server and sent via the service broker to the client. This document encapsulates information about which instruments on the ELVIS are required in the present setup. On parsing `LabConfiguration.xml`, the client retrieves the data it carries and displays the appropriate instruments on its Schematic Panel.

Once the assignmnent is displayed to the student, he configures the various instruments visible on the panel by inputting appropriate parameter values. These values are compiled into the `ExperimenSpecification.xml` document which is sent to the lab server where it is parsed. Once the experiment is conducted as per the user's specifications, the results are stored in vectors in `ExperimentResult.xml`. This document is then sent to the client where the relevant information is extracted and graphed.

2.2 National Instruments Educational Laboratory Virtual Instrumentation Suite (NI-ELVIS)

The NI-ELVIS as seen in Figure 2-5(a) is an electronic workbench which can be used to design and test circuits. This all-in-one device allows students to perform electronic experiments on a supplied (detachable) prototyping board (Figure 2-5 (b)) using a


(a) NI-ELVIS Workbench

(b) Close-up of Prototyping board on ELVIS

Figure 2-5: The National Instruments Educational Laboratory Virtual Instrumentation Suite

suite of twelve Virtual Instruments (VIs) which include a function generator (`FGEN`), oscilloscope (`SCOPE`), Digital Multimeter (`DMM`), and others. The ELVIS hardware is connected to a computer via a Data Aquistion (DAQ) card.

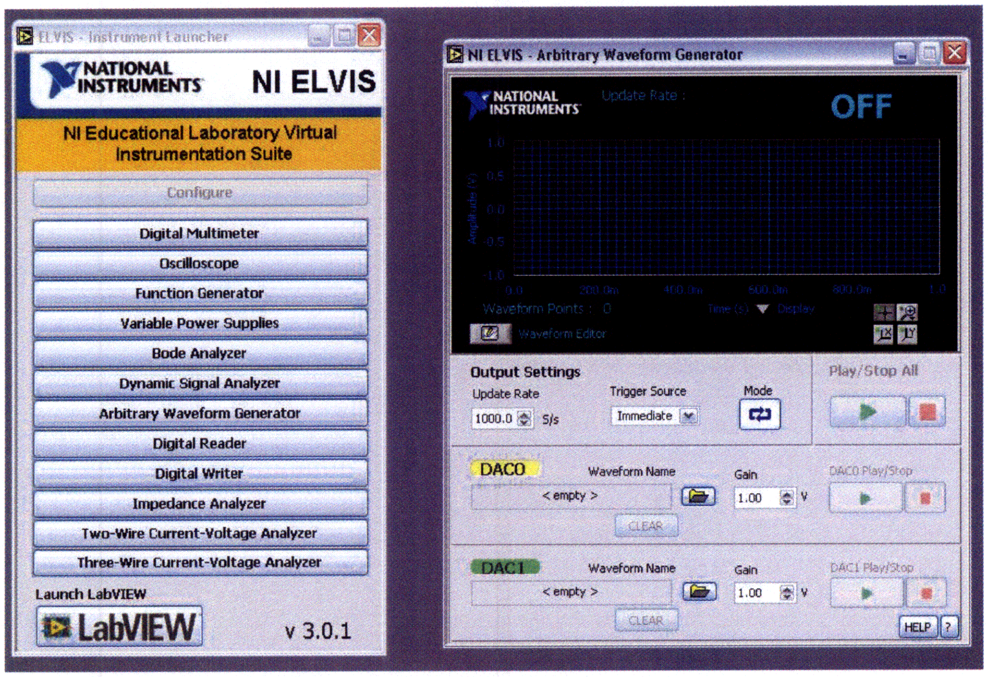
Instruments on the ELVIS can be accessed using the knobs and switches on its face. The ELVIS is also highly customisable using the LabVIEW programming environment, which also allows its features to be accessed remotely. Thus this workbench fits nicely into the iLab context.

The NI-ELVIS was designed to be used primarily in a university setting. It combines the 12 most commonly used instruments in electrical engineering experiments. Further, students are not limited to experiments involving circuits. National Instruments provides several boards which can be used to replace the standard prototyping board seen in Figure 2-1(b). These include the EMONA board for telecommuncation experiments and the Quanser Rotary Inverted Pendulum Bundle for experiments involving control design.

As mentioned earlier, slow internet speeds as a direct result of low bandwidth and high latency proved to be an impediment to the use of iLabs in East Africa. The cost effective ELVIS provided a solution. Since the ELVIS retails at a relatively cheap $2000, each university can be provided with its own system. Students can then use the high-speed campus intranet to access iLabs, which can also be specifically designed to suit each university's curricular needs. This obviates the need for our partners to access equipment housed at MIT over slow, low-bandwidth networks. Further, since this all-in-one device is sufficient to perform a wide array of engineering experiments, other, expensive equipment need not be bought. Owing to the highly scalable nature of iLabs, this one piece of hardware can cater to an entire department and thus overcomes the problem of creating the infrastructure to house several sets of identical equipment.

Although low internet speeds and high costs were a major factor in the introduction of the ELVIS at our African partner universities, its use led to a framework that provided for easy exchange and reusability in iLab development efforts. This has inspired the creation of an MIT managed iLab forum which, in addition to the several National Instrument controlled fora that already exist, provides iLab developers with extensive technical assistance and support.

2.3 LabVIEW



Update Rate 1000.0 S/s
Trigger Source Immediate
Gain 1.00 V

Figure 2-6: Screenshot of the NI-ELVIS Instrument Launcher which provides users access to the 12 instruments on the ELVIS

The National Instruments Laboratory Virtual Instrumentation Engineering Workbench (LabVIEW) is a platform and development environment for a visual, dataflow programming language. LabVIEW provides programmers with an intuitive approach to programming by directly linking the creation of user interfaces into the development process.

LabVIEW programs and subroutines are called virtual instruments (VIs). Each VI consists of three distinct components: a back-end block diagram, a front panel and a connector panel. The front panel is essentially a graphical user interface that allows users to input data into controls and extract information from indicators in a VI. The connector panel is used to represent the virtual instrument in the block diagrams of other, parent VIs. LabVIEW is so powerful because it allows each Virtual Instrument not only to be run as an autonomous program, but also permits its easy incorporation into higher level, parent programs. This concept of encapsulation allows for the reuse of code and the creation of modular, robust programmes.

LabVIEW provides users with access to the twelve instruments on the ELVIS. An Instrument Launcher (Figure 2-6) enables users to control each instrument individually. If developers wish to create more involved programs they may make use of LabVIEW's Express VIs as a basis. An ExpressVI is a high level API which encompasses most of the common functionality for any given instrument while automatically handling lower level functions such as device initialisation.

While using ExpressVIs can make development quicker and easier, they are not always the most suitable option. Using lower level VIs, though potentially tedious, can have the advantage of providing more flexibility in implementation and design. A balanced approach, developing with ExpressVIs along with lower level VIs as appropriate, should provide the best of both cases. Overall development can be reasonably quick while taking advantage of as much of the ELVISs functionality as possible.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Previous Versions of ELVIS iLab and Motivation for ELVIS v4.0

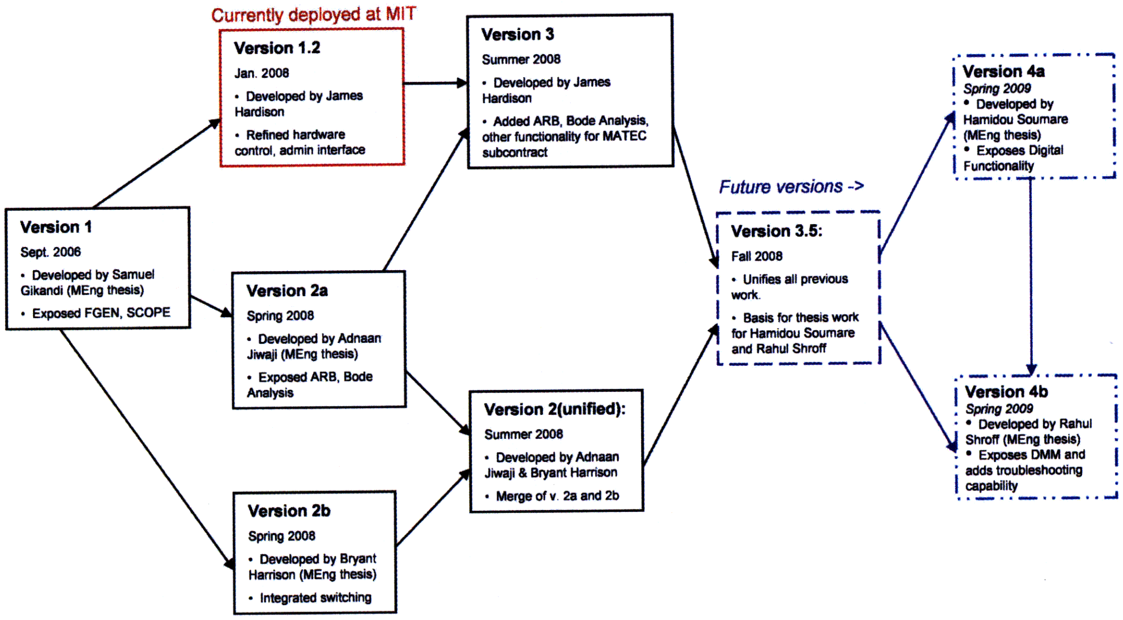

Currently deployed at MIT
Version 1.2
Jan. 2008
• Developed by James Hardison
• Refined hardware control, admin interface
Version 3
Summer 2008
• Developed by James Hardison
• Added ARB, Bode Analysis, other functionality for MATEC subcontract
Version 4a
Spring 2009
• Developed by Hamidou Soumare (MEng thesis)
• Exposes Digital Functionality
Future versions ->
Version 1
Sept. 2006
• Developed by Samuel Gikandi (MEng thesis)
• Exposed FGEN, SCOPE
Version 3.5:
Fall 2008
• Unifies all previous work.
• Basis for thesis work for Hamidou Soumare and Rahul Shroff
Version 2a
Spring 2008
• Developed by Adnaan Jiwaji (MEng thesis)
• Exposed ARB, Bode Analysis
Version 4b
Spring 2009
• Developed by Rahul Shroff (MEng thesis)
• Exposes DMM and adds troubleshooting capability
Version 2(unified):
Summer 2008
• Developed by Adnaan Jiwaji & Bryant Harrison
• Merge of v. 2a and 2b
Version 2b
Spring 2008
• Developed by Bryant Harrison (MEng thesis)
• Integrated switching

Figure 3-1: ELVIS iLab Development Trajectory

Since the adoption of the NI-ELVIS as a cost-effective platform for the development of iLabs in 2006, there have been several, distinct releases of the software-suite used to power ELVIS-based iLabs. Each of these versions has provided faculty at MIT and our partner universities with greater flexibility in designing meaningful assignments for use in engineering curricula. At every iteration, particular attention was paid towards overcoming limitations of previous versions of the ELVIS, as exposed by students and faculty alike. The following section details the ELVIS iLab development trajectory (Figure 3-1) and also provides some motivation for my contribution.

3.1 ELVIS v1.0

The first version of the ELVIS iLab was completed in 2006 by Samuel Gikandi in fulfilment of his Master of Engineering thesis at MIT [5]. Gikandi used the Microelectronics iLab's general framework as a basis for its development. However, to make ELVIS-based iLabs more natural to use, a specifically tailored client was developed. The lab server was also appropriately modified to effectively communicate with the new hardware while the original service broker was used. ELVIS v1.0 exposed the `SCOPE` and `FGEN` functionalities of the ELVIS enabling users to study single input circuits. A screen shot of the ELVIS v1.0 client is seen in Figure 2-2.

Owing to the fact that only two of the ELVIS's instruments were exposed in v1.0, it did not support a very wide range of experiments. More specifically, ELVIS v1.0 could only handle time-domain experimentation. No frequency- or digital-domain analysis could be performed. This was a considerable limitation of this version, given the fact that these concepts form an integral part of most undergraduate level electrical engineering courses. Almost all iLabs using ELVIS v1.0 consisted of simple operational amplifier characterisation experiments involving a single input/output configuration. Further, due to its inability to handle more than one input or output, or switch between existing setups, v1.0 only supported a single circuit configuration per board. This meant that if several courses at the same university wished to use iLabs as part of their curriculum, a new set of hardware would be needed for each course. This, in a way, belied the ELVIS's reputation as a low cost replacement for higher end equipment. Another implication of the inability to switch between points on the proto-board was that every setup was fixed, i.e., a user could only analyse a previously wired circuit. For example, if he wished to measure input voltage instead of output voltage, it would not be possible. This would require the physical circuit to be rewired by an administrator. In other words, a student had absolutely no control over the apparatus.

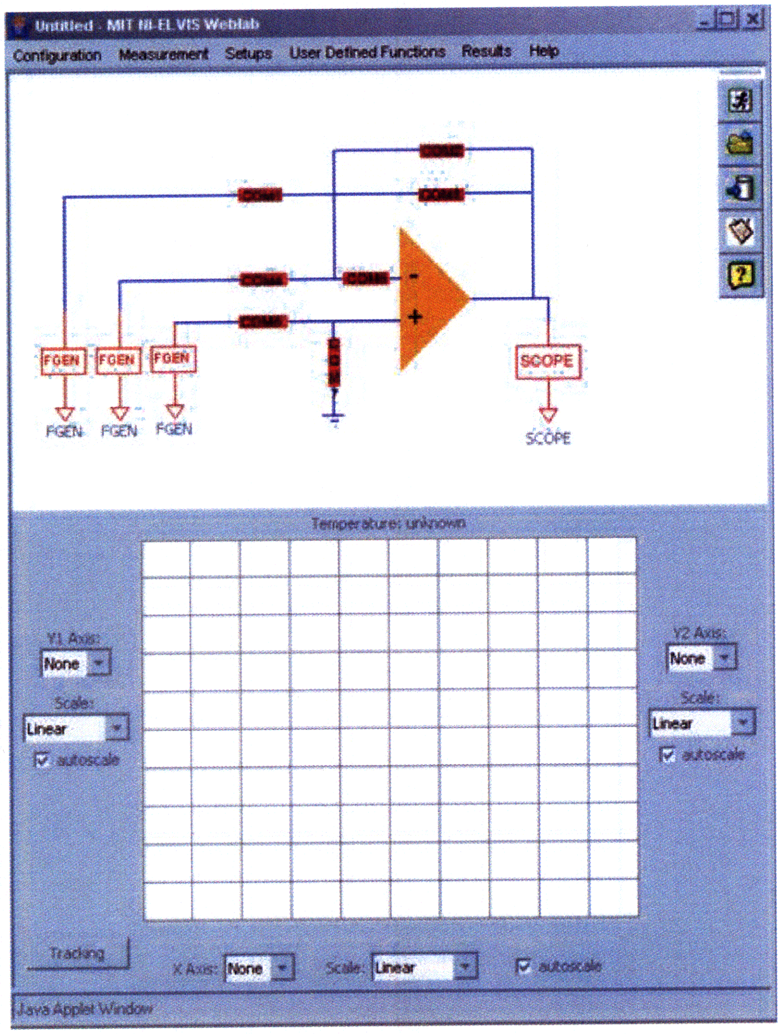
Despite all these limitations, EVLIS v1.0 was a great success. Its cost effective nature implied that our African partners would no longer have to rely on slow, low-bandwidth networks to access equipment housed at MIT. iLabs based on this version were extensively used at MIT and other universities. ELVIS v1.0 also provided a robust platform conducive to subsequent development.

3.2 ELVIS v2.0

This version of the ELVIS provided iLab developers with greater flexibility by exposing three new instruments: The Arbitrary Waveform Generator (`ARB`) which allowed students to explore the response of circuits to arbitrary input waveforms, the Bode Analyser (`BODE`) which enabled frequency domain study, and the Variable Power Supply (VPS) which could produce an output voltage between -12V and 12V. Thus, this version took significant steps in overcoming the limitations of its predecessor.

A considerable limitation of the older ELVIS v1.0 was its inability to perform frequency-domain analysis. This was overcome by the addition of the BODE instrument in v2.0, which allowed the creation of iLabs to study transfer functions of simple LTI systems.

Through the addition of the Single Pull Single Throw switching hardware, ELVIS v2.0 provided iLabs with an important component switching capability. Users could now study the behaviour of a particular setup as a function of its components. For example, consider the schematic shown in Figure 3-3 which shows an inverting operational amplifer. We see that by throwing appropriate switches, the value of the feedback resistor can be set to $10K$, $20K$ or $200K$. In the client, a user can select which resistor he wishes to use and by studying the transfer function of the circuit for different values of this resistor can determine how output voltage changes with respect to feedback resistance. Thus, the addition of the switching matrix provided students with a certain degree of real-time control over the configuration of a circuit,



Y1 Axis: None
Scale: Linear
☑ autoscale
Y2 Axis: None
Scale: Linear
☑ autoscale
X Axis: None
Scale: Linear
☑ autoscale

Figure 3-2: Screenshot of ELVIS v2.0 Lab Client. Each red box represents a switchable component

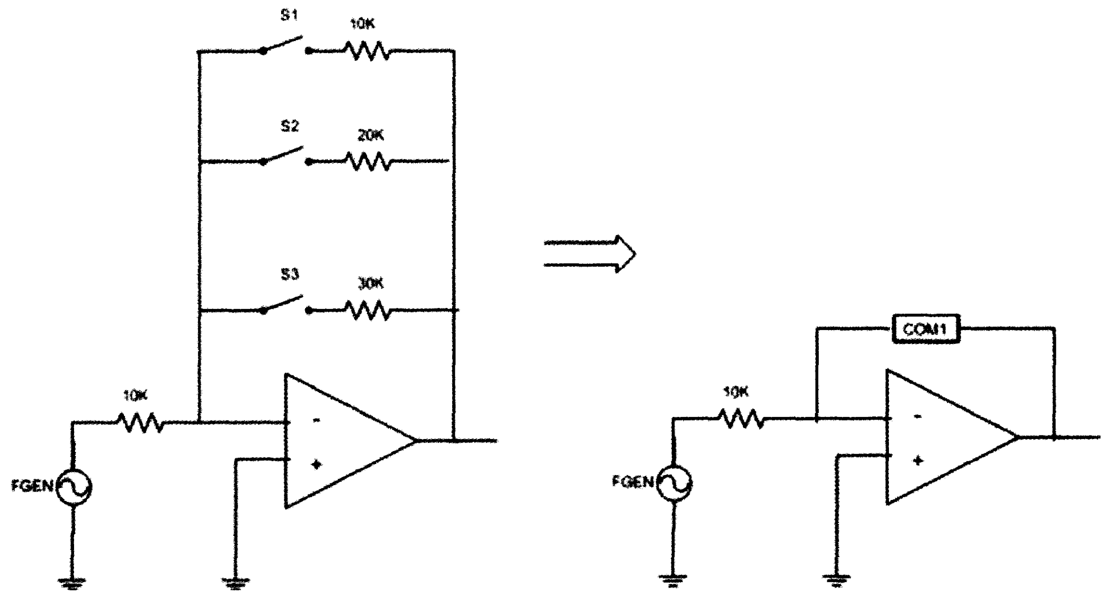


Figure 3-3: OpAmp circuit schematic simplified using ELVIS v2.0 switching capability

previously unavailable in an iLab. Further, from a financial perspective, the integration of switching facilitated greater hardware utilisation by allowing numerous setups per board.

The ARB and BODE features were added in Summer 2008 and formed the bulk of Adnaan Jiwaji's M. Eng. Thesis [8]. The switching component was added by Bryant J. Harrison in 2008 in completion of his M. Eng. Thesis [6].

We can see an example of a v2.0 ELVIS Lab Client in Figure 3-2. Note the COM boxes in the circuit. Each of these represents a switchable component.

3.3 Unification of parallel development efforts in the iLabs project

Branches in the development effort led to the emergence of two separate versions of the ELVIS iLab, each with its own functionality and distinct approach to the underlying LabView Virtual Instrument (VI) structure. For example, James Hardison's MATEC version did not incorporate any switching capabilities. However, this version used

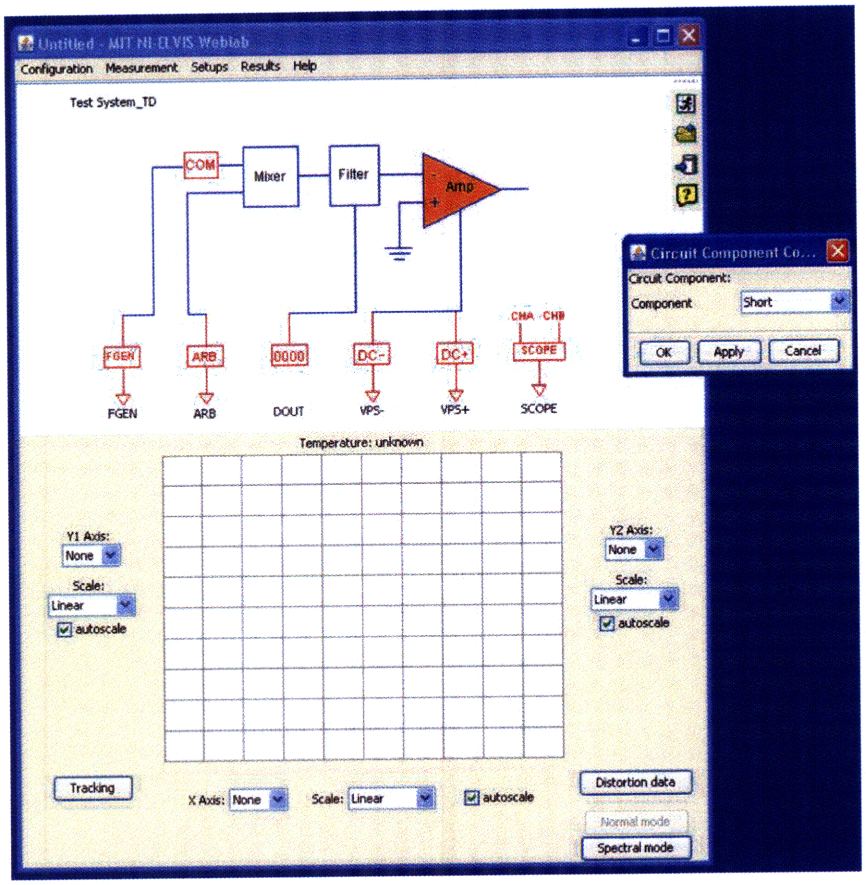


Figure 3-4: Screenshot of ELVIS v3.5 Client representing all exposed functionality. The COM dialogue box is seen on the right and represent how a user chooses a switchable component.

ExpressVIs to drive the hardware, as opposed to the other versions which used lower level virtual instruments. Thus, to prevent further divergence and inconsistencies, the creation of a consolidated codebase was imperative.

Working with fellow M. Eng. student, Hamidou Soumare, my initial task was to create a new version of the ELVIS iLabs, v3.5, which packaged all previous work into one new version, thus creating a robust release from which further development could be undertaken. This new version enabled instructors and developers to use the entire breadth of exposed ELVIS functionality to create new and more meaningful labs. Further, ELVIS v3.5 also enabled our partner institutions to benefit from work done for individual cases.

A screenshot of the the ELVIS v3.5 Lab Client is seen in Figure 3-5.

3.4 Motivation for ELVIS v4.0

To enable a greater variety of experiments to be run on the ELVIS iLabs, the next step was to increase the number of the instruments available on the ELVIS that could be accessed by iLab developers. One such functionality still not exposed was the Digital Multimeter (DMM). The DMM instrument can perform a wide range of basic measurments - DC voltage, AC voltage, current (AC and DC), resistance, capacitance, and inductance as well as perform diode and continuity tests.

The Digital Multimeter is extremely useful in the testing and debugging of an electronic circuit and is most often used in physical labs for this purpose by connecting it to various points in a circuit. Within the iLabs context, its use would only be justified if we were able to move its position within the circuit. Now with switching integrated into the latest version of the ELVIS iLab, this usage model could be implemented, enabling a variety of remote experiments that were previously unavailable.

It is important to realise that although the SCOPE instrument gives us some measurement capability, it is limited in the sense that it deals only with time-domain, voltage-related measurements. Measuring current for example requires the tedious procedure of having to fit a small resistor in the circuit and programme the ELVIS control software with its value in order to measure the Ohmic drop across it. Performing the same measurement with the DMM is much more streamlined. While integrating the DMM increases the complexity of the overall system, it enables a variety of measurements in a more robust fashion.

My work on the ELVIS v4.0, described in depth in the next chapter, introduces the Digital Multimeter, and by combining it with an adapted version of the switching capabilities introduced in v2.0, enables students to examine dynamically configured circuits.

Chapter 4

ELVIS v4.0 Detailed Design

This chapter gives a detailed outline of ELVIS v4.0 and sequentially describes its development process. Owing to the deliberate compartmentalisation of components in the ISA, the process of adding the Digital Multimeter instrument into the already existing framework was very modular. The entire operation was extremely iterative, with several preliminary versions being tested and subsequently fine-tuned to provide the optimum user experience.

4.1 XML Specification Documents

We saw in Section 2.1.4 that components in the ISA communicate with each other solely through the use of three XML documents: `LabConfiguration.xml`, `ExperimentSpecification.xml` and `ExperimentResult.xml`. All information relevant to a particular run of an experiment is relayed between the client and lab server via the service broker using these documents. Each of these specification files has a unique structure which dictates what and how information is represented, accessed and stored in the client and server. To preserve this structure, each document has associated with it a Document Type Definition (DTD). A DTD is primarily used to define contraints on the format of an XML document. Specifically tailoring each DTD file ensures that only valid XML files are exchanged by components of the ISA. In this section, I describe the changes that were made to these XML documents and their corresponding DTD files to account for the addition of a new instrument, the Digital Multimeter.

4.1.1 Lab Configuration

`LabConfiguration.xml` (Appendix B), encapsulates all information pertaining to a laboratory setup. More specifically, it contains a list of all terminals that will be present in that setup along with each terminal's associated information (such as name, icon pixel location and serial number etc.). It also provides a brief description of the setup and a file path for its schematic representation. `LabConfiguration.xml` is generated in the lab server after querying the database to see which instruments will be used in a given experiment.

`LabConfiguration.xml` needed to be modified to account for the new DMM instrument. As a result, `Labconfiguration.dtd` (Appendix C) was also updated to allow for the creation of a new terminal block corresponding to the DMM. It should be noted that `LabConfiguration.xml` segments instruments into modes corresponding to domains of operation of the ELVIS. As we will discuss in more detail later, the DMM creates a new mode of experimentation on the ELVIS: the DC mode. Accordingly, the DTD file was also changed to allow for DC experimentation.

4.1.2 Experiment Specification

`ExperimentSpecification.xml` (Appendix D), packages all the information a user enters into the client while configuring an experiment to his liking. This XML document contains several blocks of input parameter values for each terminal in an active setup.

Once again, the DTD file associated with experiment specifications (Appendix E) needed to be updated to account for the input parameters of the Digital Multimeter. The generated DMM block from `ExperimentSpeciciation.xml` is seen below.

```
<terminal instrumentType="DMM" instrumentClass="input" instrumentNumber="7" setupTermID = "7">
<function type="DMMFunction">
```

```
<udm>Resistance</udm>
<rangeid>3</rangeid>
<position>1010101010101010101010101</position>
</function>
</terminal>
```

We see that the DMM has three inputs: a user desired measurement, a range ID and a position. Each of these is explained in great detail in the following sections.

4.1.3 Experiment Result

`ExperimentResult.xml` (Appendix F), contains data vectors representing output values and measurements for an experiment. These data vectors are not specific to a particular instrument, and as such, only one set of vectors is required. Due to its generic nature, no changes needed to be made to this document to account for the addition of the Digital Multimeter.

4.2 Lab Server

The lab server plays a crucial role in the creation and execution of experiments. The flowchart in Figure 4-1 illustrates its various components and portrays their mutual interaction. Briefly, the lab server is composed of:

- Back-end code, which can be further categorised into:
  - a `Validation Engine`, which ensures the validity of input parameters both in value and in format,
  - an `Experiment Engine`, which continuously runs in the background during the execution of an experiment, interfacing with the service broker on one end and a DLL Wrapper function on the other,
  - the DLL Wrapper function which gets parameter values from the `Experiment Engine` and presents them to the corresponding LabVIEW VI for execution,

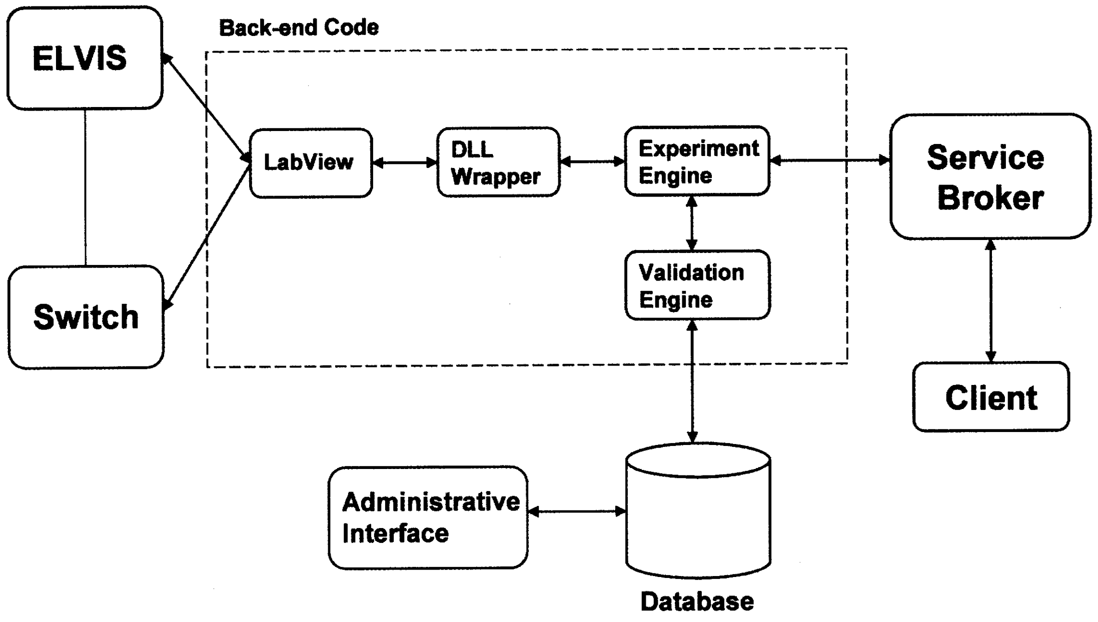


Figure 4-1: ELVIS iLab Development Trajectory

  - a LabVIEW virtual instrument which interacts directly with laboratory hardware (ELVIS and Switch) to carry out an experiment. Once the experiment has been executed, empirical results are sent via the DLL Wrapper to the Experiment Engine from where it is conveyed to the client for display.

- an *Administrative Interface* where faculty can design and activate labs.

- a comprehensive *Database* which stores all information pertinent to the execution of experiments in the lab server.

Due to its equipment specific nature, the lab server is arguably the least generic of the three components in the ISA. In this section I elaborate on how each of the above components of the lab server was changed to incorporate the Digital Multimeter instrument.

4.2.1 LabVIEW

As explained in Section 2.3, LabVIEW is a graphical programming environment which directly, or through the (adapted) use of Express Virtual Instruments gives developers access to the EVLIS instrumentation suite.

4.2.1.1 ELVIS v3.5 `Main2.vi`

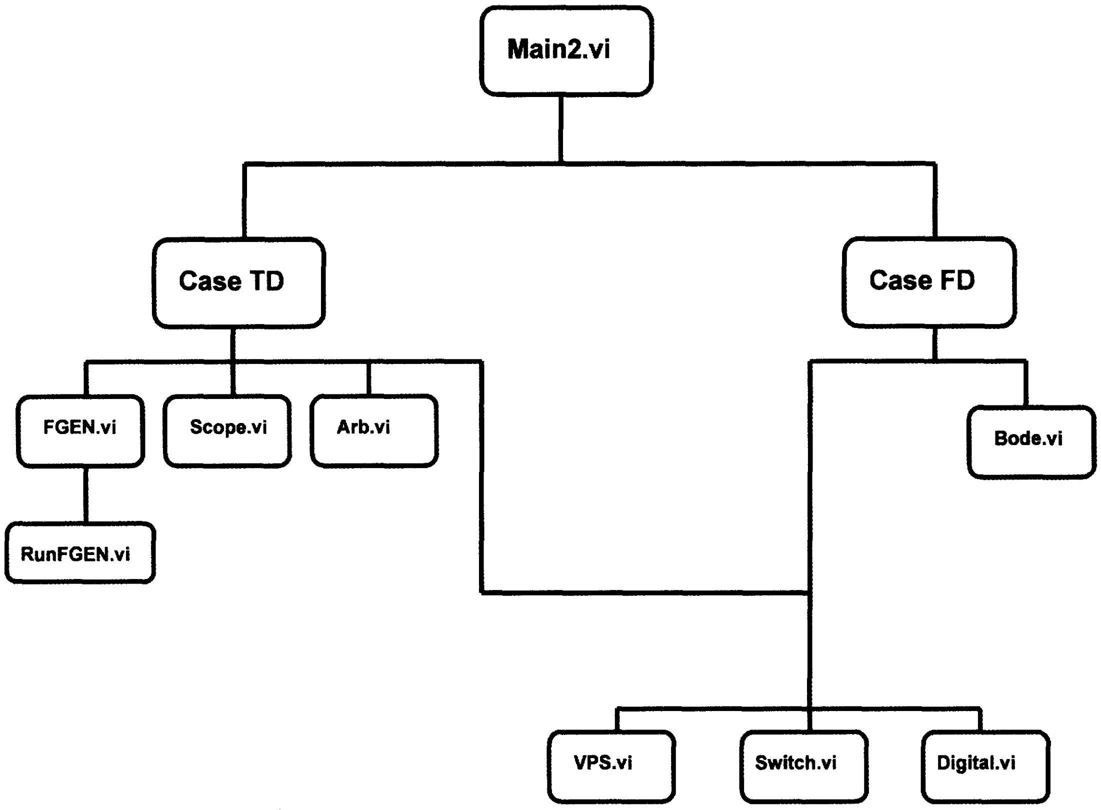


Figure 4-2: Flowchart illustrating code heirarchy in ELVIS v3.5 `Main2.vi`

Taking advantage of LabVIEW's ability to represent a VI using its associated input and output channels (as defined on each VI's connector panel), ELVIS v3.5 was powered by an all inclusive program, `Main2.vi`, containing instances of all virtual instruments that had previously been developed for ELVIS-based iLabs. The schematic shown in Figure 4-2 illustrates the code heirarchy in `Main2.vi`. Each VI within `Main2.vi` is categorised as being either part of the frequency-domain , the time-domain or common to both. This separation into mutually exclusive cases was necessary to prevent any resource conflicts on the ELVIS. For example, `Bode.vi` had to be placed in a separate case within the code because its Express VI uses the FGEN and SCOPE to input a time-varying signal and to read a circuit's frequency response respectively.

4.2.1.2 `DMM.vi`

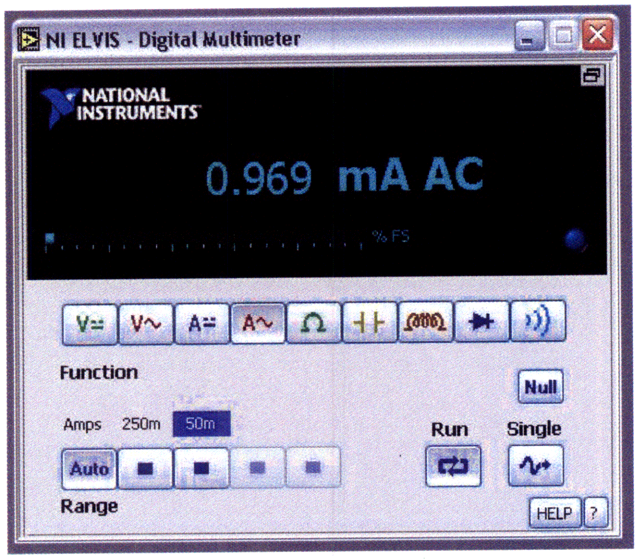
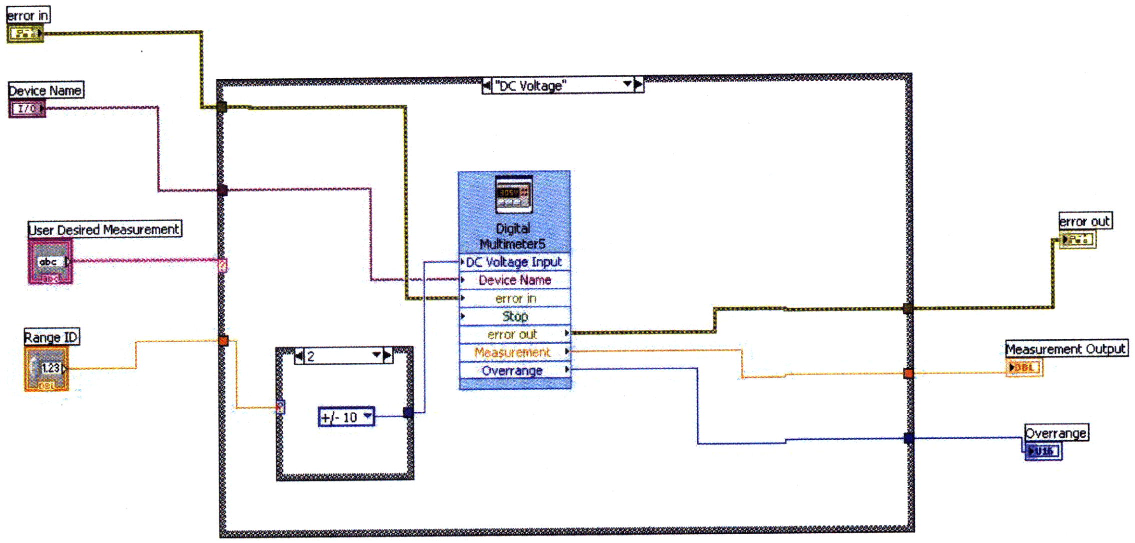


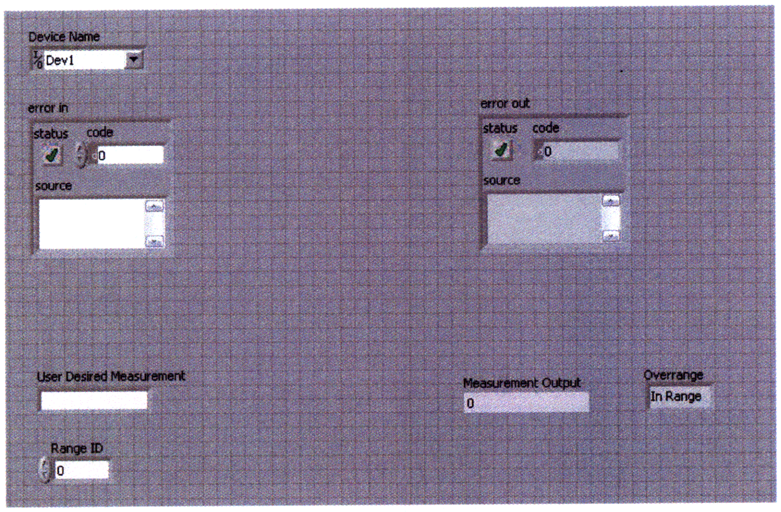
Figure 4-3: Screenshot of GUI for Digital Multimeter

`DMM.vi` was designed keeping its end use in mind. Simply put, by itself, `DMM.vi` allows users to input a measurement type and performs that desired measurement on the ELVIS. This is accomplished using seven different instances of the DMM Express VI, each conducting the following seven, distinct measurements: Resistance, Capacitance, Inductance, DC Voltage, DC Current, AC Voltage amd AC Current. I used a simple case structure, triggered by the *User Desired Measurement* control variable to separate each kind of measurement. LabVIEW code for the case controlling a DC Voltage measurement is seen in Figure 4-4(a).

`DMM.vi` has two inputs relevant to its use: a *User Desired Measurement* and a *Range ID.* The *Range ID* is used to specify the range of values a user expects his desired measurement to fall within. For simplicity, I represent this *Range ID* as an integer number, which once inputted is translated into a valid range corresponding



(a) DMM.vi Back Panel



(b) DMM.vi Front Panel

Figure 4-4: Screenshot of LabVIEW code for DMM.vi

to each measurement type.

Analogously, `DMM.vi` has two related outputs: *Measurement* and *Overrange.* Once the underlying VI performs the desired measurement on the ELVIS, its result is stored in indicator variable *Measurement. Overrange* is used to indicate to the user whether or not the conducted measurement is within the earlier specified range.

It may be noted that when accessing the DMM instrument directly through LabVIEW's instrument launcher, a user is given the option to allow the ELVIS to automatically determine each measurement's range. However, this feature is not available when using the DMM Express VI and thus needed to be added as an input. From my experience, for most measurement types (with the possible exception of Voltage), even inputting an incorrect range gives an extremely accurate measurement. *Range ID* was made visible to the user more as a learning tool, enabling a student to reasonably estimate what range a measurement might fall within based on other information he is provided with.

4.2.1.3 Resource Conflicts and ELVIS v4.0 `Main2.vi`

Once `DMM.vi` had been created and tested thoroughly, it had to be placed in the larger framework of iLab LabVIEW code. The placement of `DMM.vi` in `Main2.vi` was influenced by resource conflicts on the ELVIS I. These resource conflicts are a result of the fact that some analogue input channels on the ELVIS are used by the internal circuitry for other instruments. Further, some instruments depend on others to function. A list of resource conflicts can be found in Appendix A. We see that all DMM component type measurements conflict with the Function Generator. All other DMM measurements conflict with the Oscilloscope and Bode Analyser on the ELVIS I.

To avoid these resource conflicts, I placed `DMM.vi` in its own case within `Main.vi`, separate from the time- and frequency-domain cases where it would conflict with FGEN and BODE respectively. I called this case the DC (Direct Current or Static) domain. This nomenclature is a bit of a misnomer in the sense that `DMM.vi` does not only perform DC measurements. However, once placed in this framework, to avoid

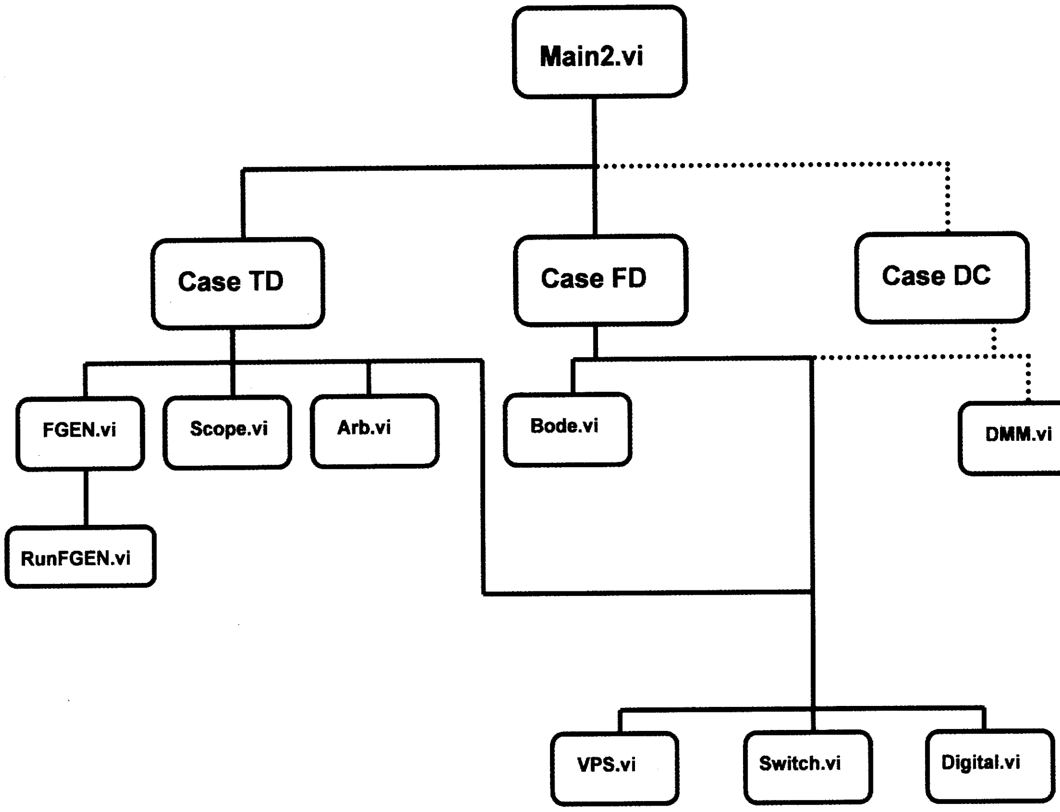


Figure 4-5: Flowchart illustrating code heirarchy in ELVIS v4.0 `Main2.vi`

resource conflicts only DC power sources can be used in conjunction with the DMM. The only time-varying source the ELVIS provides is the FGEN with which the DMM conflicts. I did however choose to expose the AC measurement capabilities of the DMM in `DMM.vi`. What prompted me to do this was the fact that the next logical step in the iLab development cycle is to upgrade our hardware to the newer ELVIS II. This revamped workbench uses completely different underlying circuitry and, as a result, does not present any resource conflicts with the DMM. Once adapted for use with the ELVIS II, the DMM could be used to perform an even wider array of measurements.

To faciliate the creation of a new case, and to preserve the inherent structure of `Main2.vi` I created another virtual instrument, `DMM_parse.vi`. This helper module bundles the two DMM inputs *User Defined Measurement* and *Range ID* into an output cluster of comma separated values which is then inputted into `Main2.vi`. LabVIEW only allows each VI to have thirty nodes on its connector panel and by condensing the number of inputs into the VI, nodes can be preserved for future use.

A flowchart depicting the various modules that comprise the latest version of `Main2.vi` is seen in Figure 4-5.

4.2.2 Experiment Engine

The Experiment Engine is an executable that runs in the background of the lab server, waiting for activity from the service broker. Its operation can be broadly categorised into three distinct steps executed in its `Main()` method.

1. When the service broker passes a request from a user for a particular laboratory, the `loadjob()` function is called. This function queries the databases for active setups, and, using the information it retrieves , it creates `LabConfiguration.xml`. As mentioned earlier, the lab configuration file is separated into cases corresponding to modes of operation in the ELVIS. This segmentation is necessary to avoid any resource conflicts. When modifying `loadjob` to handle the DMM instrument, I added a new case, the DC case, to enable the creation of a new mode in the generated XML document.

2. Once `loadjob` creates `LabConfiguration.xml` it is submitted to the client. The experiment engine then waits for configured experiments to be submitted by users. These submitted experiments are then de-queued from the SQL server and must be parsed in the experiment engine to translate each user's experiment specification. This is accomplished using the `ParseExperimentSpec()` method in the experiment engine. This method reads `ExperimentSpecification.xml` and stores all the information it contains in two matrices, *terminfoTable* and *functinfoTable*. *terminfoTable* contains information about the terminals present in a particular setup, while *functinfoTable* matches each terminal with its input parameter values. Both these tables needed to be extended to account for the DMM instrument. I also added a block to the actual parsing subroutine enabling it to identify the DMM instrument from a list of terminals, and load its three parameter values into the matrix.

3. Once `ExperimentSpecification.xml` has been successfully parsed, the execution engine calls `RunExperiment()`. This method uses the information stored in *functinfoTable* to create a list of user defined input parameters. Once this has been created it calls RunELVISExp.exe, an executable file, to instantiate the ELVISWrapper class which subsequently calls a LabVIEW DLL to run the experiment on the ELVIS hardware. Once the experiment has been executed the DLL returns an interleaved array of output data from the EVLIS. Using this data, `RunExperiment` compiles the `ExperimentResult.xml` file summarising all the results. This file will eventually be sent to the client where it is parsed to reveal the experiment's results to the user.

   I added a new case in `RunExperiment` to handle the execution of DC-domain experiments. This allows me to effectively separate instruments used in time- and frequency-domain study while using the DMM and thus prevents any resource conflicts. Within this case I added functionality to extract DMM related input information from the abovementioned tables and to create a parameter list to be sent to the DLL . I also handled the deinterleaving of data once it had been received from the DLL and the creation of `ExperimentResult.xml`.

4.2.3 Adapting Switching for use with the Digital Multimeter

The machinery described in the preceding two sections allows users of ELVIS v4.0 to take a wide array of elementary, point-to-point measurements. However, due to the nature of iLabs, students are only exposed to a diagrammatic representation of the circuit schematic they are examining. They do not have the opporunity to physically change the actual configuration of a circuit. Given this limitation, in the version described before, the probes of the DMM had to be connected to two fixed points in a circuit. Using this model, students would only be able to take measurements across two previously selected circuit points. Consider the simple RLC circuit seen in Figure 4-6. In the existing model, an administrator would connect the DMM across, say the capacitor, and students would only be able to make voltage or current measurements across it. For example, if a student wanted to find the value of R as labelled in the figure, it would be impossible without physically reconfiguring the circuit.

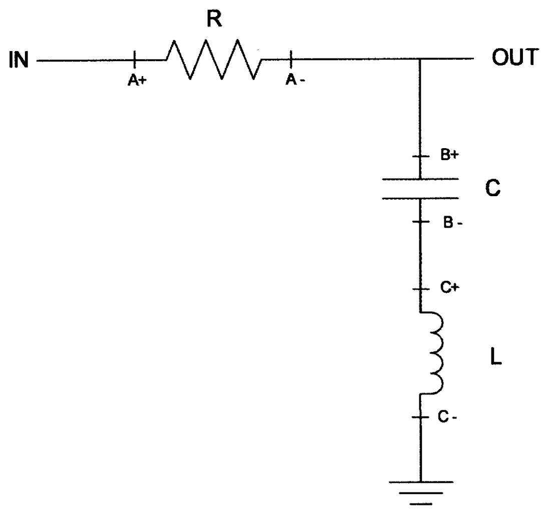


Figure 4-6: A simple RLC circuit

I chose to expose the ELVIS's DMM functionality with the aim of providing students with troubleshooting capabilities previously unavailable on ELVIS-based iLabs. This implies that students should have complete flexibility in dynamically changing the location of the DMM in a circuit. This would, to a certain degree, replicate what he or she would do while examining a circuit in a traditional laboratory setting.

To overcome this limitation of single point investigation, I combined the switching introduced by Harrison [6] in ELVIS v2.0 with the DMM functionality. Due to the fact that ELVIS v2.0's switching focussed primarily on multiple component use, it needed to be modified for use with the DMM.

4.2.3.1 Switching Layers

To facilitate switching withing the DMM framework I introduced the concept of *switching layers*. As depicted in Figure 4-7 there are four switching layers that assist with DMM measurements:

1. As seen in the figure, the first layer consists of switches defining measurement

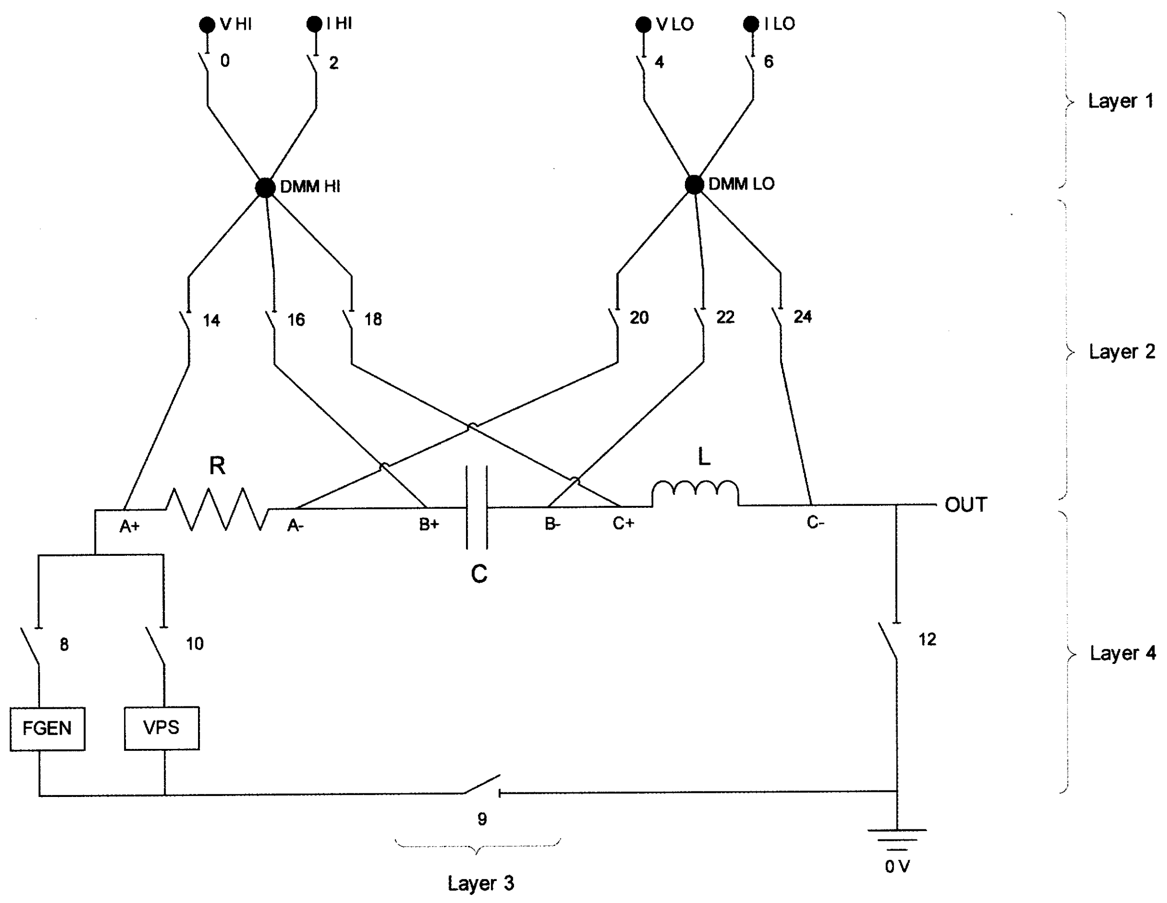


Figure 4-7: RLC circuit with switching to enable multipoint measurement

category. This level is necessary because the ELVIS uses different resources to conduct, what it calls, 'Current Type' and 'Voltage Type' measurements. Simply put, when conducting voltage measurements of any sort, the points under investigation must be connected to the `VOLTAGE HI` and `VOLTAGE LO` terminals of the DMM on the ELVIS bread board. For all other measurements we connect to the `CURRENT HI` and `CURRENT LO` terminals on the proto-board.

2. The next layer of switching is related to measurement position. This is the layer of switching which is used to specify where exactly the user wishes to place the DMM. Points of interest are marked on the circuit schematic displayed in the client to the user. Points labelled with a + or − indicate connections for the DMM's positive or negative terminal respectively. For example, if the user wishes to connect the DMM across the resistor, he would close switch #14 and

#20 as seen in Figure 4-7.

3. Another layer of switching is used to facilitate branch current measurements. Typically, while taking current measurements, the branch being investigated is disconnected from the circuit. The terminals of the DMM are then used to bridge the resulting open circuit. This third layer of switching is used to simulate this behaviour as illustrated in Figure 4-7.

4. The final layer of switching deals with peripheral instruments. As seen in the figure these are switches connected to various sources and grounds which are part of the circuit. Correctly switching these components is extremely necessary. For example, it can be seen from Figure 4-6 that the inductor, L, is connected to ground. If we did not open this connection to ground while taking an Inductance measurement across $C+$ and $C-$, we could get an inaccurate value.

The actual switch hardware consists of a chasis which houses a switch module. This module is connected to a terminal block containing several pairs of nodes, each of which represents a typical switch. Circuits like the one seen in Figure 4-7 can be created by connecting each node pair on the terminal block to points on the ELVIS prototyping board.

Thus, by wiring a given circuit in a manner analogous to the schematic shown in Figure 4-7 and by throwing the appropriate switches, users can take any type of measurement, across any combination of points using the DMM.

4.2.3.2 Switch String

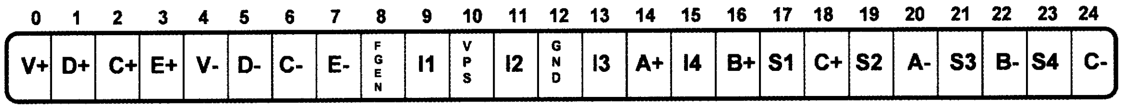

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
V+ D+ C+ E+ V- D- C- E- FGEN I1 VPS I2 GND I3 A+ I4 B+ S1 C+ S2 A- S3 B- S4 C-

Figure 4-8: Illustrative switch string

ELVIS v2.0 used a switch string, i.e., a continous string of 25 characters, to represent the 25 switches (switch 0 to 24) contained in the switching module. Each

Bit #	Content	Interpretation
0	V+	`DMM VOLTAGE HI` on ELVIS protoboard
1	D+	Connection point for `DMM HI` probe as defined on circuit schematic
2	I+	`DMM CURRENT HI` on ELVIS protoboard
3	E+	Connection point for `DMM HI` probe as defined on circuit schematic
4	V-	`DMM VOLTAGE LO` on ELVIS protoboard
5	D-	Connection point for `DMM LO` probe as defined on circuit schematic
6	I-	`DMM CURRENT LO` on ELVIS protoboard
7	E-	Connection point for `DMM LO` probe as defined on circuit schematic
8	FGEN	ELVIS Function Generator connection to circuit
9	I1	Status of branch in circuit (for branch current measurement)
10	VPS	ELVIS Variable Power Supply connection to circuit
11	I2	Status of branch in circuit (for branch current measurement)
12	GND	$0V$ connection to circuit
13	I3	Status of branch in circuit (for branch current measurement)
14	A+	Connection point for `DMM HI` probe as defined on circuit schematic
15	I4	Status of branch in circuit (for branch current measurement)
16	B+	Connection point for `DMM HI` probe as defined on circuit schematic
17	S2	Switchable component 1
18	C+	Connection point for `DMM HI` probe as defined on circuit schematic
19	S3	Switchable component 2
20	A-	Connection point for `DMM LO` probe as defined on circuit schematic
21	SC4	Switchable component 3
22	B-	Connection point for `DMM LO` probe as defined on circuit schematic
23	SC4	Switchable component 4
24	C-	Connection point for `DMM LO` probe as defined on circuit schematic

Table 4.1: Description of each bit in Switch String

character takes a 0 or 1 value representing the open and closed positions of each switch respectively. To prevent confusion and maintain consistency, I introduced a standard whereby each bit of the switch string represents a particular switch in each switching layer. This standard should be adhered to when wiring circuits containing switches for use with the DMM. A figurative switch string, with labels for each bit, specifying the standard is seen in Figure 4-8. Table 4-1 provides a detailed explanation of what each bit in the switch string represents. Thus, an administrator may specify up to five points of connection for each of the DMM's terminals allowing 25 distinct points of measurement.

It may be noted that this standard only needs to be maintained when an active setup contains the DMM instrument.

4.2.3.3 Changes made to the Lab Server to incorporate Switching

The primary aim of adapting switching for use with the Digital Multimeter is to allow for complete flexibility in the placement of the virtual DMM probes within a circuit. Since we are trying to replicate a conventional laboratory experience it is important to keep users unaware of the underlying switching mechanism used to accomplish this. Thus, a student is presented with a circuit schematic similar to the one in Figure 4-6 and asked to choose the points between which he wishes to place the DMM. These coordinates are translated into a switch string in the client and sent to the lab server (in the *position* field of `ExperimentSpecification.xml`) where it is stored in *functinfoTable.* The `RunExperiment()` method in the experiment engine of the lab server then updates the already existing switch string with the new information using two procedures (Figure 4-9):

- *resetSwitchstr* which resets all DMM related bits to the open position before the switch string is updated. This ensures that only the switches needed for this run of the experiment are closed. It must be noted that simply resetting all bits to zero does not accomplish this because each switch string also stores information related to switchable components. This information must be preserved to make an accurate measurement.

- *updateSwitchstr*, which parses the information received from the client and copies it into the DMM related bits of the switch string. Care was taken not to overwrite the bits corrresponding to switchable components.

Finally, an administrator can specify the branching factor for each terminal of the DMM i.e., to how many points in a circuit he wishes to connect each of its terminals. To enable this, the database and some of its stored procedures needed to be updated. This information is stored in the *dmmConnections* column of the database.

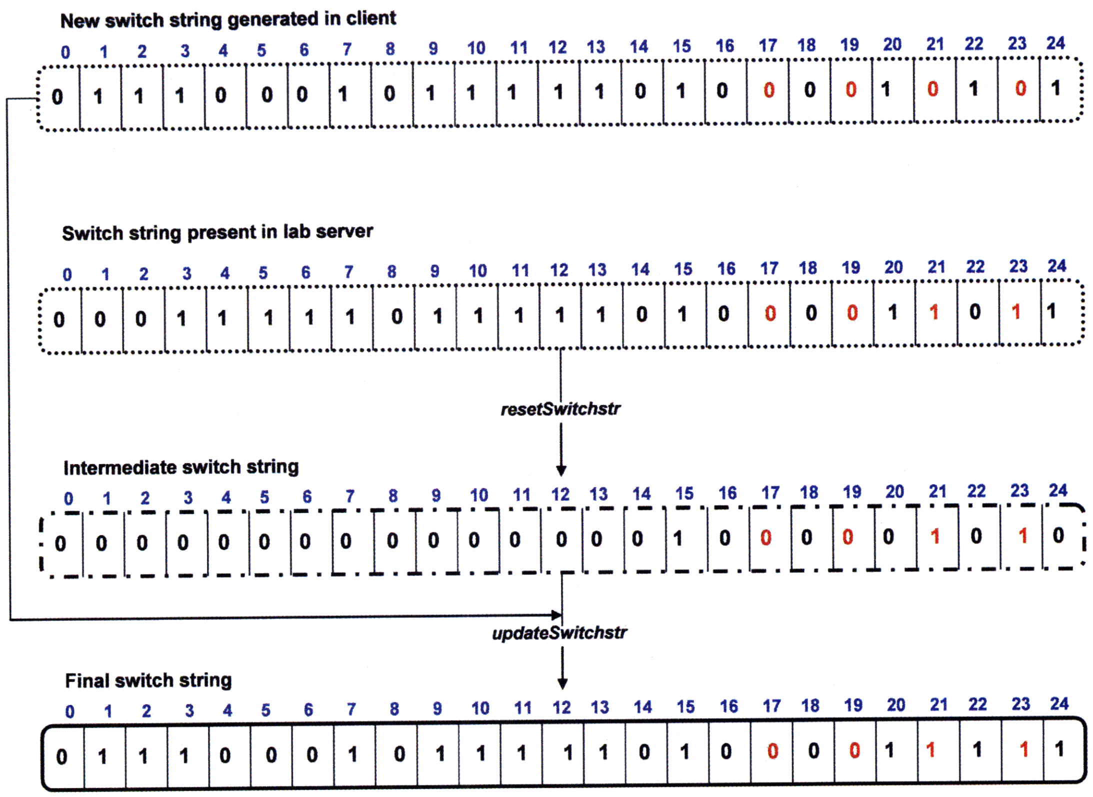


Figure 4-9: Switch string creation in the Lab Server

4.2.4 Validation Engine

The Validation Engine's primary function is to parse an experiment specification and ensure its correctness. It contains an `experimentValidate()` method which performs this function. The validation engine needed to be updated to reflect the addition of the DMM instrument. Since the DMM cannot be used in either the time- or frequency-domains, I created a new DC-domain case for it. This ensures that if an administrator mistakenly designs an experiment using the DMM in the time-domain, execution would stall before the lab is sent to the ELVIS. This final check guarantees that no resource conflicts will occur once the lab has been submitted to be run.

4.3 Weblab Client

For ELVIS-based iLabs using the batched architecture, a user's interaction with an experiment is limited to his communication with the client. Thus, from a user-experience point of view, the client is probably the most important part of the iLabs Shared Architecture. The original v1.0 weblab client was inspired greatly by the GUI on the microelectronics weblab. As we have seen in Chapter 3, this client has transformed over the years to reflect added functionality and make the iLab experience as intuitive to a student as possible.

4.3.1 General Structure of the Client

The client is written in Java and has an extremely modular structure. It has generic graphing, xml and server interfaces which were inspired by the Microelectronics weblab client. Each instrument in the client is characterised by its name, information parameters, schematic panel icon and associated dialog box. When a user launches the client, a Java Applet window pops up showing a schematic and results panel. Users can configure each instrument by clicking its icon on the schematic panel and entering appropriate values into the dialog box that appears. Once the student is satisfied with his specifications, he submits the experiment for execution to the server. After the experiment has been run to completion, the client displays the data on its results panel. The user may then use the client's graphing utilities to scale the graph or track individual points. Users are also given the option to save or retrieve experiments or to export results for further analysis.

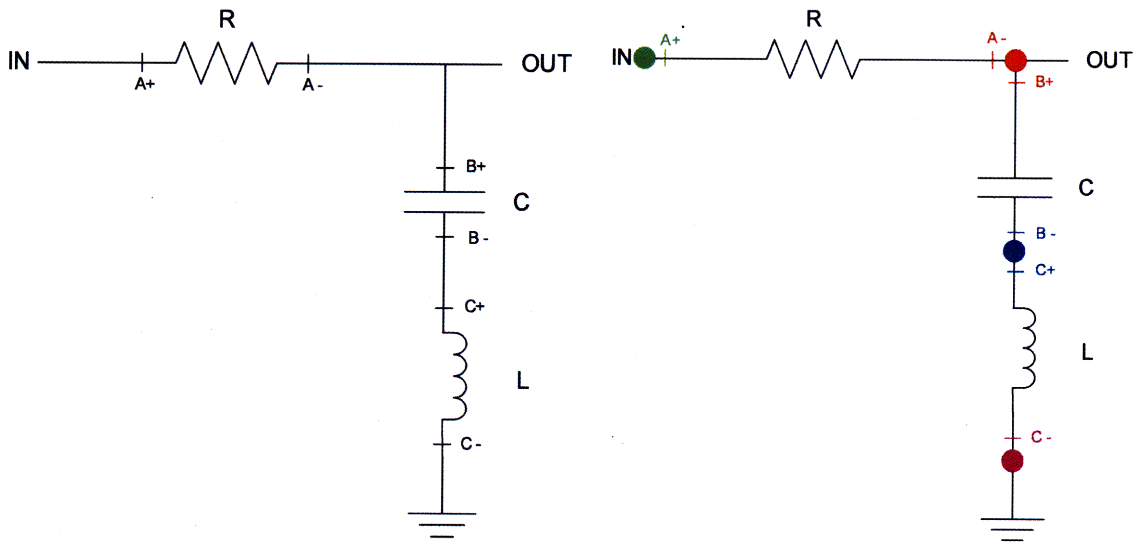
4.3.2 Changes made to client to incorporate the DMM

The client weblab package which contains a majority of the user interface code, needed to be modified to include the DMM. Classes such as *DMM*, *DMMFunction*, *DMMLabel* and *DMMDialog* were added to expose the new functionality. Further, superclasses such as *Instrument*, *Terminal* etc. were appropriately modified.

We saw in Section 4.1 that each of the three XML documents were updated during the addition of the DMM into the system. Two of these namely, `LabConfiguration.xml` and `ExperimentResult.xml`, are sent by the lab server and parsed in the client. The two Java classes accomplishing this needed to be amended to account for the changed strcuture of these documents. The third document, `ExperimentSpecification.xml`, is generated within the client before it is sent to the server. The class responsible for this, *ExperimentSpecification* was also extended to enable the conversion of user inputted parameter values for the DMM into legitimate XML blocks, in adherence with the schema dictated in `ExperimentResult.dtd` (Appendix G).

4.3.2.1 Representation of Circuit Schematics

The above changes although significant, only created the skeletal framework needed to support DMM use in the client. A considerable amount of time was spent making Digital Multimeter based iLabs more user friendly. The DMM provides students with a slightly different user experience as compared to other instruments on the ELVIS. For the first time, users are given complete flexibility regarding where they wish to connect an instrument. Further, the results sent back after computation on the ELVIS are scalar values which need to be displayed in the client rather than plotted, as had previously been the case.



(a) Previous RLC Circuit

(b) RLC Circuit with coloured clusters representing ekectrical nodes

Figure 4-10: RLC Circuit Schematic

In addition to making iLabs involving the DMM natural to use, the client's design was greatly influenced by the need to always present an *electrically accurate* circuit to the user. Several improvements were made to facilitate this. Consider the circuit schematic seen in Section 4.2.3, which has been replicated in Figure 4-10(a). The added switching functionality enables users to choose which points ($A+$, $A-$, $B+$ etc) to investigate on this circuit. However, a student is completely oblivious to the machinery used to achieve this. Due to this, the schematic seen in (Figure 4-10(a)) might be potentially confusing for students. For example, the wire connecting one end of the resistor to the capacitor in the circuit represents *one* electrical node. However, to facilitate flexibility in measurement there exist *two* distinct points ($A-$ and $B+$) on this wire to which the DMM terminals can connect. From the DMM point of view, having these two points is absolutely necessary since we must be able to physically connect both its positive and negative terminals to an electrical node for analysis. However, since the user only sees a virtual representation of the circuit and is unaware of the switching mechanism used to take multipoint measurements, seeing nodes labelled as in Figure 4-10(a) might be disconcerting.

Doing away with these point labels would greatly reduce the flexibility of taking measurements with the DMM. Users would now only be able to take measurements across points previously selected by an administrator. To prevent this, I chose to represent these electrical nodes as distinctly coloured clusters. As seen in Figure 4-10(b), each cluster consists of three points (a '+' and '−' representing connection points for the DMM's virtual probes, and a coloured circle representing the actual node) and together they represent an electrical node. This approach preserves the inherent usability of the system while making clear that each wire can indeed be represented by a single node. To make it even more clear to students, once the DMM has been configured in the client, its icon morphs to reflect which points on the circuit it is connected to and also what type of measurement it is taking.

I also udpated the Variable Power Supply (VPS) instrument in the client to correctly change its representation in the schematic panel based on its value and use. A screenshot of the ELVIS v4.0 client is seen in Figure 4.

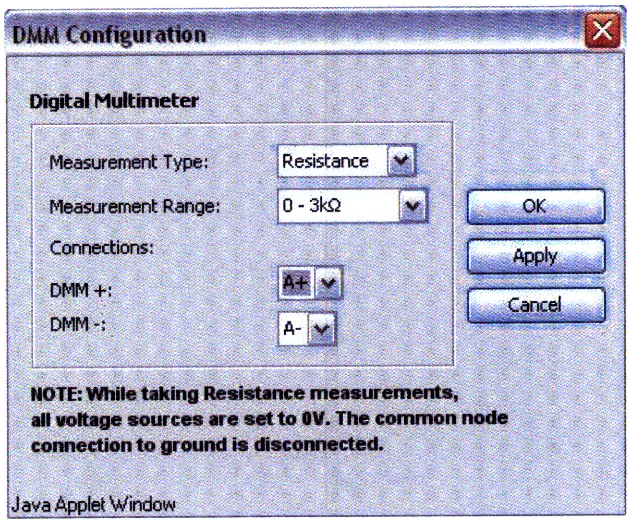
4.3.2.2 Dialog Box for User Configuration

The dialog box created for the DMM is seen in Figure 4-11(a). It contains several drop-down menus from which users can select desired measurement type, range and position of the DMM terminals. Although users are ignorant of switch use in taking measurements, it is important that they are made conscious of the changes in the configuration of the circuit required to take certain readings (Switch Layers 3 & 4). To realise this, I include a note in the dialog box apprising students of exactly how the circuit has been altered. This note is displayed as soon as the user selects a measurement type.

4.4 Testing and Deployment

Several tests were performed using single/multiple circuit components (resistors, capacitors etc.) as well as simple circuits to ensure that the DMM was indeed making accurate measurements. The circuit depicted in Figure 4-10(b) was conceived to introduce students to frequency- and DC-domain analysis. The setup was thoroughly tested for both modes of use. The entire iLabs team contributed to making the DMM user experience as streamlined and enjoyable as possible. Further, the iLab was testing by members of the MIT undergraduate community, whose inputs helped improve the overall user interface.

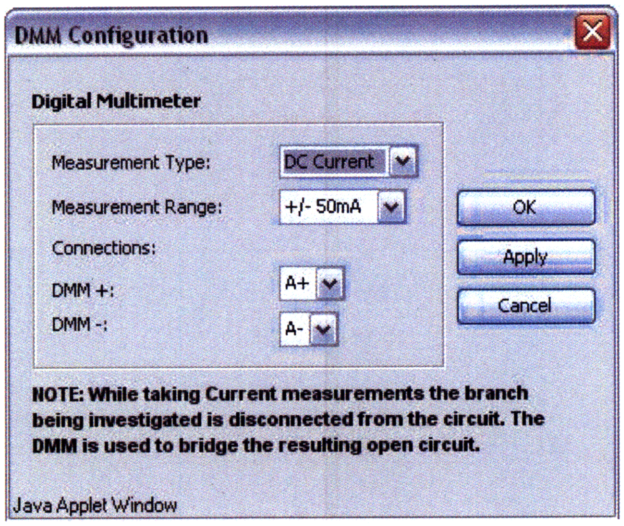
Extensive testing revealed several issues with preliminary versions of the iLab. For example, it was noticed that some sequences of measurement types led to readings that were extremely inaccurate. This was later attributed to the fact that the switch chasis was not reinitialising in time. This prompted the addition of the resetting mechanism described in Section 4.2.2.3 which rectified the problem. Further, it was seen that the client was not effectively handling infinity measurement values. These and several other problems were identified and corrected. The resulting product is seen in Figure 4-12 - Figure 4-15.



Measurement Type: Resistance
Measurement Range: 0 - 3kΩ
Connections:
DMM +: A+
DMM -: A-

NOTE: While taking Resistance measurements, all voltage sources are set to 0V. The common node connection to ground is disconnected.

(a) Dialog Box showing configuration of a Resistance measurement

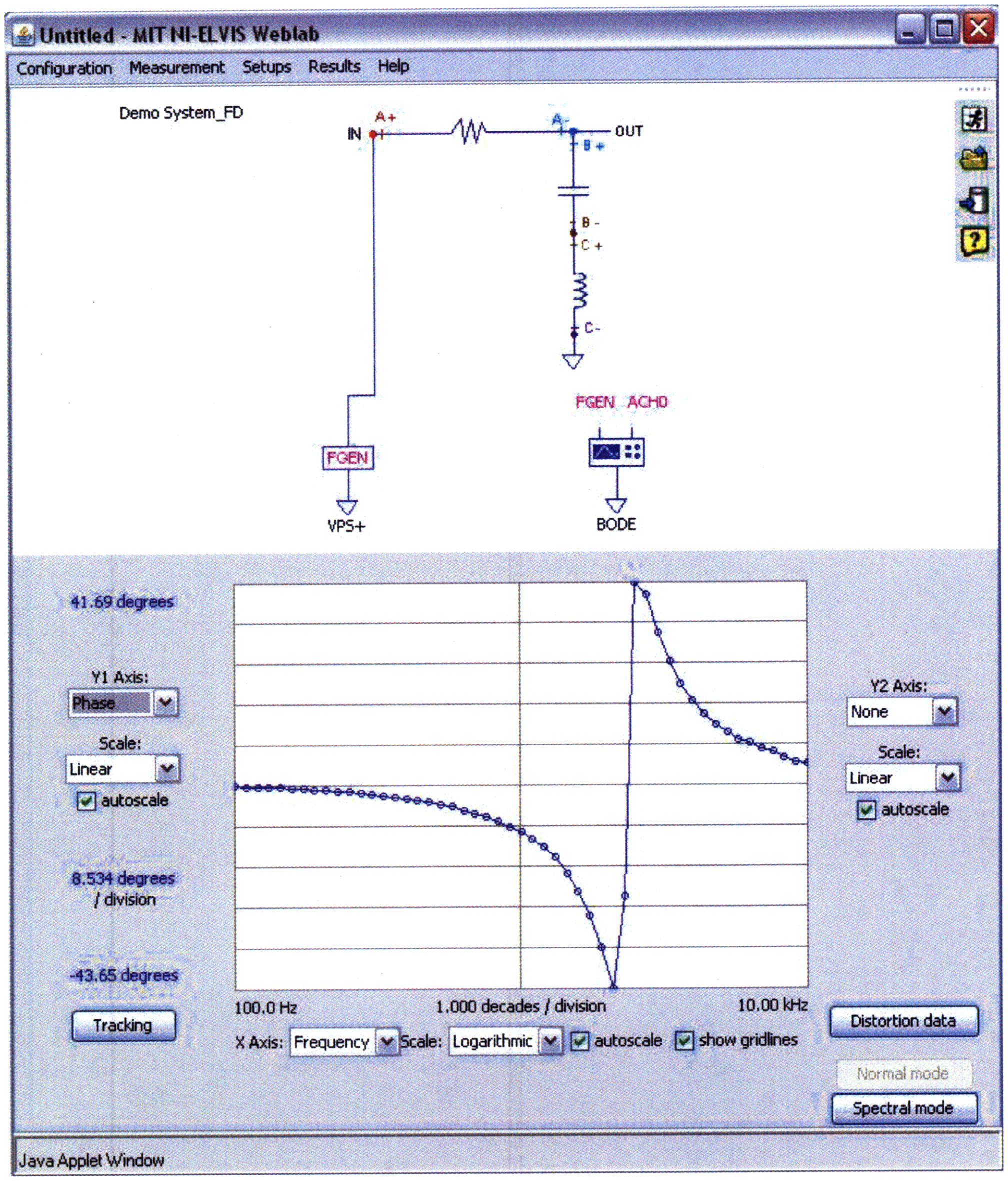


Measurement Type: DC Current
Measurement Range: +/- 50mA
Connections:
DMM +: A+
DMM -: A-

NOTE: While taking Current measurements the branch being investigated is disconnected from the circuit. The DMM is used to bridge the resulting open circuit.

(b) Dialog Box showing configuration of a Current Measurement

Figure 4-11: Dialog Box for DMM



41.69 degrees

8.534 degrees / division

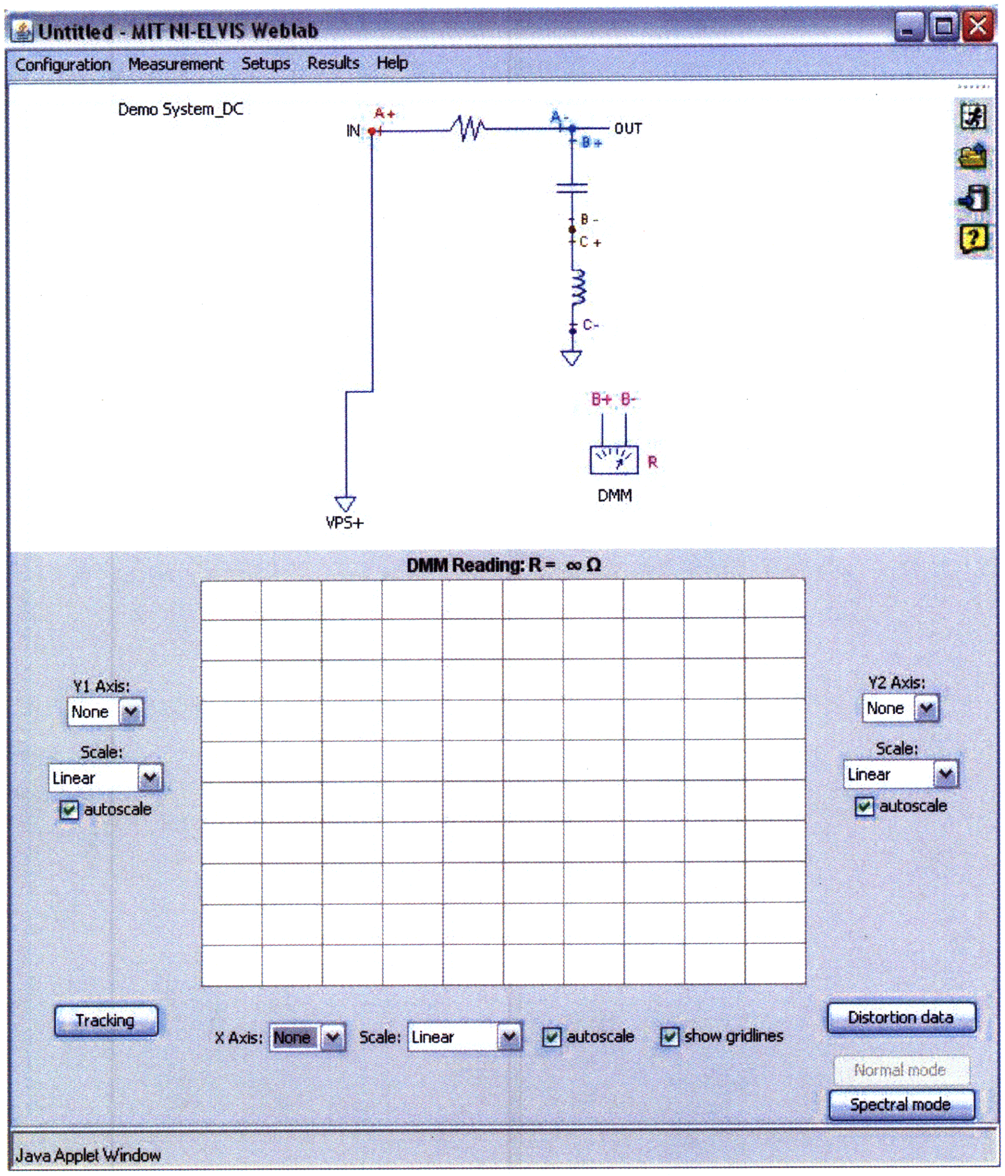
-43.65 degrees

100.0 Hz

1.000 decades / division

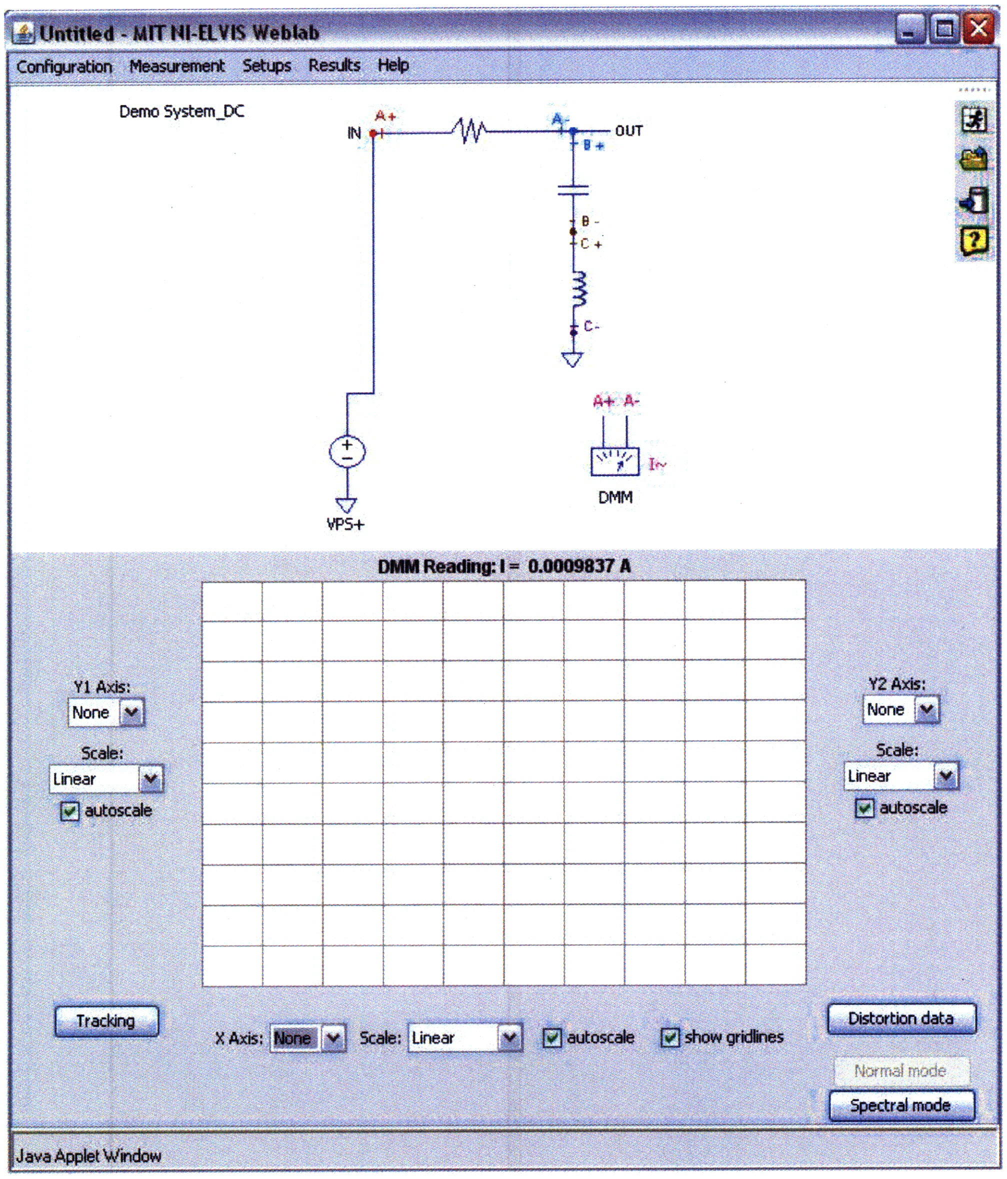
10.00 kHz

Figure 4-12: Phase plot for the frequency characterisation of an RLC circuit



DMM Reading: R = ∞ Ω

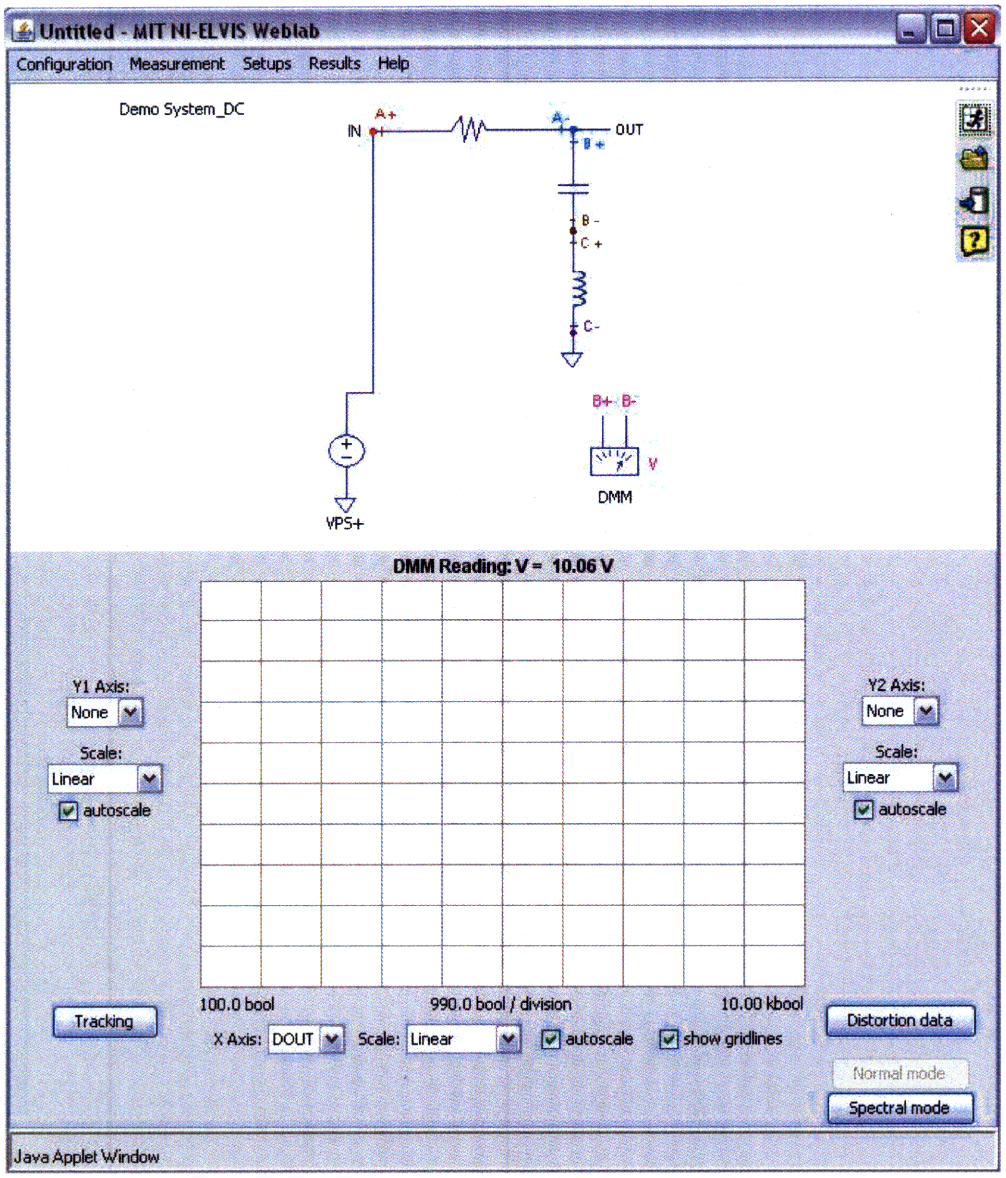
Figure 4-13: Result for measurement of resistance across the capacitor in DC mode. The capacitor behaves as an open circuit in DC and hence the measured value is infinity.



DMM Reading: I = 0.0009837 A

X Axis: None Scale: Linear ✓ autoscale ✓ show gridlines

Figure 4-14: Result for the measurement of AC current in the series circuit. Since there is no time varying source, the value is zero.



DMM Reading: V = 10.06 V

Y1 Axis: None
Scale: Linear
✓ autoscale

Y2 Axis: None
Scale: Linear
✓ autoscale

100.0 bool   990.0 bool / division   10.00 kbool

X Axis: DOUT   Scale: Linear   ✓ autoscale   ✓ show gridlines

Figure 4-15: Result for measurement of DC voltage across the capacitor. The input voltage is $10V$.

Chapter 5

Closing Remarks

5.1 Contributions

Detractors of the iLab concept have always been quick to point out that using iLabs, students are unable to test or troubleshoot a circuit. Although conceived as a complement to hands-on laboratory exercises, given their cost-effective nature it is inevitable that in certain situations iLabs will be used to completely replace their traditional counterparts. In these situations students using earlier versions of ELVIS-based iLabs would never exposed to these invaluable skills. ELVIS v4.0 takes a step forward in overcoming this disability by introducing the Digital Multimeter and coupling its use with an augmented version of the already existing switching. However, the multimeter's use is not limited to debugging. A wide range of laboratories, ranging from a simple experiment involving series resistance to much more complex circuits can be designed around this instrument.

I would like to take this opportunity to summarise my contributions to the iLab Project:

- Created an all-encompassing, consolidated codebase to facilitate further development.

- Extended the iLabs Shared Architecture to include the Digital Multimeter.

- Used switching to enable real-time, dynamic circuit testing and troubleshooting, unprecedented in an iLab.
- Made changes to the client to improve the schematic representation of circuits.
- Co-taught a training programme at Makerere University, Kampala introducing new members to the iLab concept.

5.2 Recommendations for Future Development

ELVIS-based iLabs have greatly matured since their introduction in 2006. Constant development has seen a two instrument setup trying to mimic the Microelectronics Weblab transform into a compelling platform supporting time-, frequency-, and DC-domain analysis, digital experimentation and now, troubleshooting mechanisms.

However, in a bid to add new functionality some aesthetic details have been overlooked by developers. Although I have done some work in trying to improve circuit representation, what the iLab client really needs is the ability to change the way a circuit looks in real-time. For example consider the physical changes in a circuit that occur when taking measurements with the DMM. Ideally, these changes (such as sources being shorted when taking component measurements, or branches being opened when taking current measurements) should be made visible to users on the client's schematic panel. A further improvement would be to allow the position of the DMM icon to dynamically change based, on a user's selection. Changes of this nature require a complete revamp of the weblab package. Although this will take a significant effort, I believe its completion will be vital in furthering the iLab notion.

A different approach towards future development could involve incorporating one of the many third party plug-in boards that can be used with the ELVIS. Telecommunication experiments using the EMONA board are already being developed by students at Makerere University. Several other boards exist which could potentially be explored. This would greatly increase the breadth of ELVIS-based iLabs.

As of now, eight of the twelve instruments on the ELVIS workstation have been exposed. This leaves us with the Two and Three wire voltage analyser, the Dynamic Signal Analyser and the Impedance Analyser which are still to be made available to educators. It may be noted that although highly sophisticated versions of these instruments are already part of the iLabs framework, their addition is merited by the ELVIS's cost effective nature.

As mentioned earlier, National Instruments has released a much enhanced version of the ELVIS, the ELVIS II. This new workbench provides more accurate measurements at low voltages (a major flaw in the ELVIS I) and has a frequency range twenty times greater than its predecessor. Most importantly, it uses an entirely different underlying circuitry that prevents many of the resource conflicts inherent in the ELVIS I [10]. Some of our African partners have already experimented with the new ELVIS. A huge step forward would be to rework the entire ISA so that it can be used with this new hardware. While this may not be something visible to users directly, its successful implementation would in my opinion be of greatest value to the iLab Project.

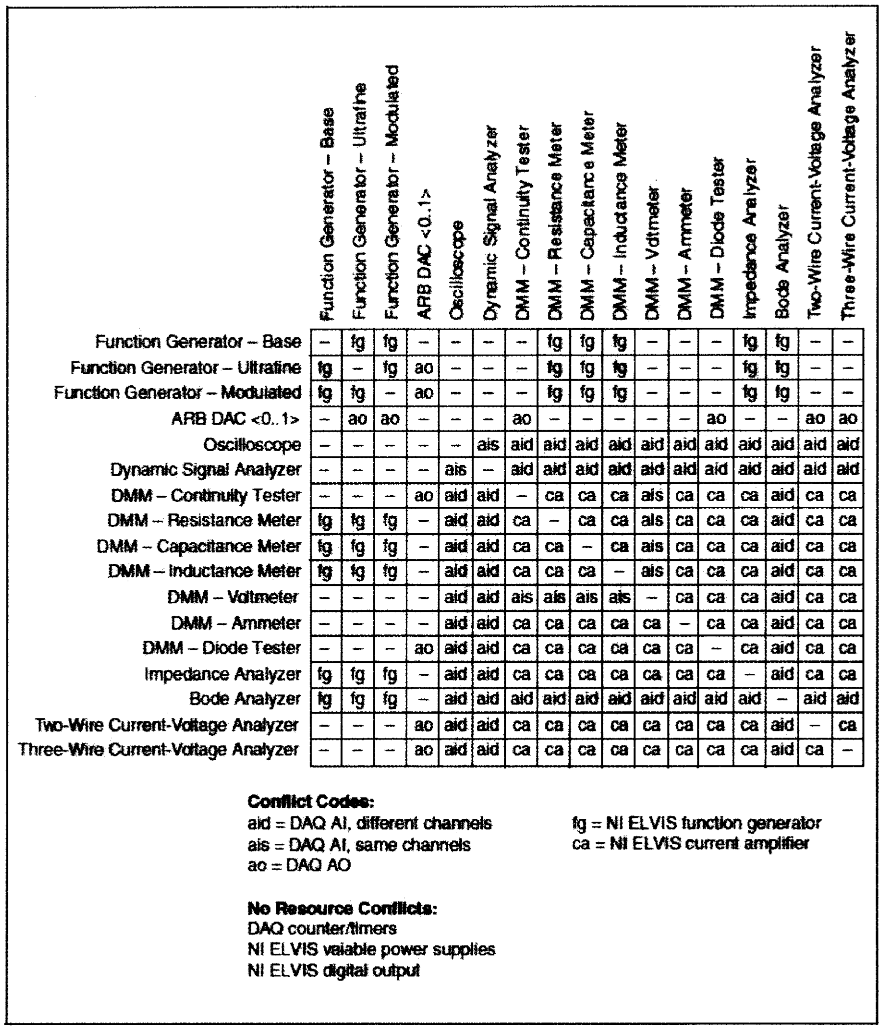
5.3 Status of iLabs in Africa

The iLabs Africa partnership has flourished under the benevolence of the Carnegie Foundation and National Instruments. Bi-annual developer exchanges have further strengthened our relationship and contributed to our partners' progress.

In January 2009, Hamidou Soumare and I visited Makerere University in Kampala to conduct a training programme for new developers. We were astonished to find that in the period of six months since the annual iLab Conference, the MUK team had grown from a three member unit to one that had fourteen energetic students yearning to be exposed to iLabs. While the training programme itself was a great success, probably a greater achievement was the fact that we were able to facilitate an exchange of students between Makerere University and the University of Dar es Salaam (UDSM). Three students from UDSM attended a week of the two week long training. Further, two students from MUK will travel to UDSM this June. This helped to initiate a system whereby these universities no longer need to rely on MIT for technical assistance and support. Helping each other will not only save expensive trips overseas, but will give them the confidence to make even more valuable contributions to the project. The next step would be for our partners to represent MIT within the African community and help spread the notion of iLabs by recruiting new partners. Given the respect, enthusiasm and commitment shown by both these teams I have reason to believe that the future of iLabs in Africa is extremely promising.

Appendix A

Resource Conflicts



	Function Generator – Base	Function Generator – Ultrafine	Function Generator – Modulated	ARB DAC <0..1>	Oscilloscope	Dynamic Signal Analyzer	DMM – Continuity Tester	DMM – Resistance Meter	DMM – Capacitance Meter	DMM – Inductance Meter	DMM – Voltmeter	DMM – Ammeter	DMM – Diode Tester	Impedance Analyzer	Bode Analyzer	Two-Wire Current-Voltage Analyzer	Three-Wire Current-Voltage Analyzer
Function Generator – Base	–	fg	fg	–	–	–	–	fg	fg	fg	–	–	–	fg	fg	–	–
Function Generator – Ultrafine	fg	–	fg	ao	–	–	–	fg	fg	fg	–	–	–	fg	fg	–	–
Function Generator – Modulated	fg	fg	–	ao	–	–	–	fg	fg	fg	–	–	–	fg	fg	–	–
ARB DAC <0..1>	–	ao	ao	–	–	–	ao	–	–	–	–	–	ao	–	–	ao	ao
Oscilloscope	–	–	–	–	–	ais	aid	aid	aid	aid	aid	aid	aid	aid	aid	aid	aid
Dynamic Signal Analyzer	–	–	–	–	ais	–	aid	aid	aid	aid	aid	aid	aid	aid	aid	aid	aid
DMM – Continuity Tester	–	–	–	ao	aid	aid	–	ca	ca	ca	ais	ca	ca	ca	aid	ca	ca
DMM – Resistance Meter	fg	fg	fg	–	aid	aid	ca	–	ca	ca	ais	ca	ca	ca	aid	ca	ca
DMM – Capacitance Meter	fg	fg	fg	–	aid	aid	ca	ca	–	ca	ais	ca	ca	ca	aid	ca	ca
DMM – Inductance Meter	fg	fg	fg	–	aid	aid	ca	ca	ca	–	ais	ca	ca	ca	aid	ca	ca
DMM – Voltmeter	–	–	–	–	aid	aid	ais	ais	ais	ais	–	ca	ca	ca	aid	ca	ca
DMM – Ammeter	–	–	–	–	aid	aid	ca	ca	ca	ca	ca	–	ca	ca	aid	ca	ca
DMM – Diode Tester	–	–	–	ao	aid	aid	ca	ca	ca	ca	ca	ca	–	ca	aid	ca	ca
Impedance Analyzer	fg	fg	fg	–	aid	aid	ca	ca	ca	ca	ca	ca	ca	–	aid	ca	ca
Bode Analyzer	fg	fg	fg	–	aid	aid	aid	aid	aid	aid	aid	aid	aid	aid	–	aid	aid
Two-Wire Current-Voltage Analyzer	–	–	–	ao	aid	aid	ca	ca	ca	ca	ca	ca	ca	ca	aid	–	ca
Three-Wire Current-Voltage Analyzer	–	–	–	ao	aid	aid	ca	ca	ca	ca	ca	ca	ca	ca	aid	ca	–

Conflict Codes:
aid = DAQ AI, different channels
ais = DAQ AI, same channels
ao = DAQ AO
fg = NI ELVIS function generator
ca = NI ELVIS current amplifier

No Resource Conflicts:
DAQ counter/timers
NI ELVIS vaiable power supplies
NI ELVIS digital output

Figure A-1: Resource Conflicts on ELVIS I [9]

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix B

Lab Configuration XML document

```
<?xml version='1.0' encoding='utf-8' standalone='no' ?>
<!DOCTYPE labConfiguration SYSTEM 'http://localhost/labserver/xml/labConfiguration.dtd'>
<labConfiguration lab='MATEC ESyst iLab' specversion='0.1'>
<setup id='9'>
<name>Audio System</name>
<description>multistage audio system</description>
<imageURL>http://olid.mit.edu/images/setups/9RLcircuit.PNG</imageURL>
<mode type='TD' enabled='true'>
<terminal instrumentType='FGEN' instrumentClass='input' instrumentNumber='1' setupTermID='1'>
<label>FGEN Input</label>
<pixelLocation>
<x>123</x>
<y>43</y>
</pixelLocation>
</terminal>
<terminal instrumentType='ARBO' instrumentClass='input' instrumentNumber='2' setupTermID='2'>
<label>ARB Input</label>
<pixelLocation>
<x>45</x>
<y>34</y>
</pixelLocation>
<enabledModes>SINE,SQUARE,SAWTOOTH,TRIANGULAR,WAVEFORM,FILE</enabledModes>
<file WAVID='1'>
<name>Speech Sample</name>
<description>this is a sample of speech, 2 seconds long</description>
<URL>[WAVfile location on LS]</URL>
<length>2</length>
<recSamplingRate>20000</recSamplingRate>
<recTotalSamples>40000</recTotalSamples>
</file>
<file WAVID='2'>
<name>Music Sample</name>
<description>this is a sample of a song, 4 seconds long</description>
<URL>[WAVfile location on LS]</URL>
<length>4</length>
<recSamplingRate>20000</recSamplingRate>
<recTotalSamples>80000</recTotalSamples>
</file>
</terminal>
<terminal instrumentType='SCOPE' instrumentClass='output' instrumentNumber='3' setupTermID='3'>
<label>Oscilloscope</label>
```

```
<source name='Circuit Output' channel='ACH0'>
<pixelLocation>
<x>45</x>
<y>34</y>
</pixelLocation>
</source>
<source name='Amp Stage Out' channel='ACH1'>
<pixelLocation>
<x>23</x>
<y>56</y>
</pixelLocation>
</source>
<source name='FGEN' channel='FGEN'>
<pixelLocation>
<x>123</x>
<y>43</y>
</pixelLocation>
</source>
<source name='ARB0' channel='ARB0'>
<pixelLocation>
<x>45</x>
<y>34</y>
</pixelLocation>
</source>
<source name='Mixer Stage Out' channel='ACH2'>
<pixelLocation>
<x>78</x>
<y>457</y>
</pixelLocation>
</source>
<postProcessOptions>SPEC,DIST</postProcessOptions>
</terminal>
<terminal instrumentType='VPSPos' instrumentClass='control' instrumentNumber='4' setupTermID='4'>
<label>Var Power Supply +</label>
<pixelLocation>
<x>166</x>
<y>84</y>
</pixelLocation>
</terminal>
<terminal instrumentType='VPSNeg' instrumentClass='control' instrumentNumber='5' setupTermID='5'>
<label>Var Power Supply -</label>
<pixelLocation>
<x>345</x>
<y>23</y>
</pixelLocation>
</terminal>
<terminal instrumentType='DOUT' instrumentClass='control' instrumentNumber='6' setupTermID='6'>
<label>Digital Out</label>
<pixelLocation>
<x>2</x>
<y>45</y>
</pixelLocation>
</terminal>
<terminal instrumentType='COM' instrumentClass='control' instrumentNumber='7' setupTermID='0'>
<label></label>
<pixelLocation>
<x>100</x>
<y>150</y>
</pixelLocation>
<subCOM subCOMType ="horizR" instrumentNumber='7' setupTermID='7'>
<label>100 K</label>
```

```
 </subCOM>
<subCOM subCOMType ="horizR" instrumentNumber='7' setupTermID='8'>
<label>200 K</label>
 </subCOM>
 </terminal>

</mode>
<mode type='FD' enabled='true'>
<terminal instrumentType='BODE' instrumentClass='output' insturmentNumber='8' setupTermID='9'>
<label>Bode Analyzer</label>
<pixelLocation>
<x>64</x>
<y>78</y>
</pixelLocation>
</terminal>
<terminal instrumentType='VPSPos' instrumentClass='control' instrumentNumber='4' setupTermID='4'>
<label>Var Power Supply +</label>
<pixelLocation>
<x>166</x>
<y>84</y>
</pixelLocation>
</terminal>
<terminal instrumentType='VPSNeg' instrumentClass='control' instrumentNumber='5' setupTermID='5'>
<label>Var Power Supply -</label>
<pixelLocation>
<x>345</x>
<y>23</y>
</pixelLocation>
</terminal>
<terminal instrumentType='DOUT' instrumentClass='control' instrumentNumber='6' setupTermID='6'>
<label>Digital Out</label>
<pixelLocation>
<x>2</x>
<y>45</y>
</pixelLocation>
</terminal>



<terminal instrumentType='COM' instrumentClass='control' instrumentNumber='8' setupTermID='0'>
<label></label>
<pixelLocation>
<x>100</x>
<y>150</y>
</pixelLocation>
<subCOM subCOMType ="horizR" instrumentNumber='8' setupTermID='7'>
<label>100 K</label>
 </subCOM>
<subCOM subCOMType ="horizR" instrumentNumber='8' setupTermID='8'>
<label>200 K</label>
 </subCOM>
 </terminal>
</mode>
<mode type='DC' enabled='true'>
<terminal instrumentType='DMM' instrumentClass='output' insturmentNumber='8' setupTermID='9'>
<label>Digital Multimeter</label>
<pixelLocation>
<x>64</x>
<y>78</y>
</pixelLocation>
<numConnections> 2 </numConnections>
```

```
</terminal>
<terminal instrumentType='VPSPos' instrumentClass='control' instrumentNumber='4' setupTermID='4'>
<label>Var Power Supply +</label>
<pixelLocation>
<x>166</x>
<y>84</y>
</pixelLocation>
</terminal>
<terminal instrumentType='VPSNeg' instrumentClass='control' instrumentNumber='5' setupTermID='5'>
<label>Var Power Supply -</label>
<pixelLocation>
<x>345</x>
<y>23</y>
</pixelLocation>
</terminal>
<terminal instrumentType='DOUT' instrumentClass='control' instrumentNumber='6' setupTermID='6'>
<label>Digital Out</label>
<pixelLocation>
<x>2</x>
<y>45</y>
</pixelLocation>
</terminal>



<terminal instrumentType='COM' instrumentClass='control' instrumentNumber='8' setupTermID='0'>
<label></label>
<pixelLocation>
<x>100</x>
<y>150</y>
</pixelLocation>
<subCOM subCOMType ="horizR" instrumentNumber='8' setupTermID='7'>
<label>100 K</label>
 </subCOM>
<subCOM subCOMType ="horizR" instrumentNumber='8' setupTermID='8'>
<label>200 K</label>
 </subCOM>
 </terminal>
</mode>

</setup>
</labConfiguration>
```

Appendix C

Lab Configuration DTD File

```
<!ELEMENT labConfiguration (setup*)>
<!ATTLIST labConfiguration    lab    CDATA    #REQUIRED
                    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
                        enabled (true | false)     #REQUIRED>
            <!ELEMENT terminal (label, pixelLocation?,subCOM*, ((source+, postProcessOptions?) |(enabledModes, file*)))>
            <!ATTLIST terminal  instrumentType    (FGEN | SCOPE | ARBO | VPSPos | VPSNeg | DOUT | COM | BODE |DMM)    #REQUIRED
                       instrumentClass    (input | output | control | switch)    #IMPLIED
                       instrumentNumber    CDATA      #REQUIRED
                       setupTermID         CDATA       #REQUIRED>

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

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix D

Experiment Specification XML Document

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE experimentSpecification SYSTEM "http://localhost/xml/ExperimentSpecification.dtd">
<experimentSpecification lab="MATEC ESyst iLab" specversion="0.1">
<setupID mode='DC'>10</setupID>
<terminal instrumentType="DMM" instrumentClass="input" instrumentNumber="7" setupTermID = "7" >
<function type="DMMFunction">
<udm>Resistance</udm>
<rangeid>3</rangeid>
<position>1010101010101010101010101</position>
</function>
</terminal>
<terminal instrumentType="VPSPos" instrumentClass="control" instrumentNumber="4" setupTermID = "4"
<function type="VPSFunction">
<value>5.0</value>
</function>
</terminal>
<terminal instrumentType="VPSNeg" instrumentClass="control" instrumentNumber="5" setupTermID = "5"
<function type="VPSFunction">
<value>-5.5</value>
</function>
</terminal>
```

```
<terminal instrumentType="DOUT" instrumentClass="control" instrumentNumber="6" setupTermID = "6">
<function type="DOUTFunction">
<byte>01100111</byte>
</function>
</terminal>
<terminal instrumentType="COM" instrumentClass="control" instrumentNumber="7" setupTermID = "0">
<function type="subCOM" setupTermID = "73">
<subCOM>horizR</subCOM>
<label>100k</label>
</function>
</terminal>
</experimentSpecification>
```

Appendix E

Experiment Specification DTD File

```
<!ELEMENT experimentSpecification (setupID, terminal+)>
<!ATTLIST experimentSpecification lab CDATA #REQUIRED
specversion CDATA #REQUIRED>
<!ELEMENT setupID (#PCDATA)>
<!ATTLIST setupID mode (TD | FD |DC) #REQUIRED>
<!ELEMENT terminal (function+)>
<!ATTLIST terminal instrumentType (FGEN | SCOPE | ARBO | VPSPos | VPSNeg | DOUT | DMM | BODE | COM) #REQUIRED
instrumentClass (INPUT | OUTPUT | CONTROL ) #IMPLIED
instrumentNumber CDATA #REQUIRED
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,input, output) |
    (udm, rangeid) |
    (subCOM, label)|
    (startFreq, stopFreq, stepsPerDec, inputAmp))>
<!ATTLIST function type (WAVEFORM | SAMPLING | ARB | BODE | VPSFunction | DOUTFunction | DMMFunction | subCOM) #REQUIRED>
<!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 source (#PCDATA)>
<!ELEMENT slope (#PCDATA)>
<!ELEMENT level (#PCDATA)>
<!ELEMENT mode (#PCDATA)>
<!ELEMENT arbSamplingRate (#PCDATA)>
<!ELEMENT arbSamples (#PCDATA)>
<!ELEMENT frequency (#PCDATA)>
<!ELEMENT amplitude (#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 input (#PCDATA)>
<!ELEMENT output (#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)>
```

Appendix F

Experiment Result XML Document

```
"<?xml version='1.0' encoding='utf-8' standalone='no' ?>
<!DOCTYPE experimentResult SYSTEM 'http://localhost/xml/experimentResult.dtd'>
<experimentResult lab='MATEC ESyst iLab' specversion='0.1'>
<datavector name='Measurement' units='meas' scalable='true' type='vector'>9848.46525562889 0</datavector>
<datavector name='Range?' units='range' scalable='false' type='vector'>1 0</datavector>
<datavector name='DMMExtra' units='none' scalable='false' type='vector'></datavector>
<datavector name='DOUT' units='bool' scalable='false' type='vector'></datavector></experimentResult>";
```

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix G

Experiment Result DTD File

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

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] iCampus: The MIT-Microsoft Alliance. iLab: Remote Online Laboratories. http://icampus.mit.edu/projects/ilabs.shtml.

[2] Jesus del Alamo. Realizing the Potential of iLabs in sub-Sahara Africa, June 2005. http://www-mtl.mit.edu/ alamo/del%20Alamo.pdf.

[3] V. Harward et al. iLabs A Scalable Architecture for Sharing Online Experiments. *Whitepaper International Conference on Engineering Education*, 2004.

[4] V. Harward et al. The iLab Architecture: A Web Services Infrastructure to Build Communities of Internet Accessible Laboratories. *Proceedings of the IEEE*, 2007.

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

[6] Bryant J. Harrison. Expanding the Capabilities of the Elvis iLab Using Component Switching". Master's thesis, Massachusetts Institute of Technology, 2008.

[7] V. Harward. Service Broker to Lab Server API, May 2008. http://icampus.mit.edu/iLabs/architecture/downloads/protectedfiles/service broker to lab server api.doc.

[8] Adnaan Jiwaji. Modular Development of an Educational Remote Laboratory Platform for Electrical Engineering: the ELVIS iLab. Master's thesis.

[9] National Instruments. *NI ELVIS*, April 2008. http://www.ni.com/academic/ni_elvis/.

[10] National Instruments. *NI ELVIS II*, April 2008. http://digital.ni.com/manuals.nsf/websearch/DCFAB6F3A71636F58625743200519759.