Remotely Controlled Electronics Laboratory Workbench

A Project Report

submitted by

MADAKASIRA MANJUNATH

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

JUNE 2011

CERTIFICATE

This is to certify that the thesis titled **Remotely Controlled Electronics Laboratory Workbench**, submitted by **Madakasira Manjunath**, to the Indian Institute of Technology Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Nagendra Krishnapura
Advisor,
Associate Professor,
Dept. of Electrical Engineering,
IIT-Madras, 600 036

Place: Chennai
Date: 27 May 2011

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my project advisor Prof. Nagendra Krishnapura for giving me an opportunity to work on an exciting project which can have an impact on the students of our department and for his support and guidance throughout the project. His lectures in Analog Circuits and Analog IC Design courses have always left me in awe and inspired me. His dedication, meticulousness, enthusiasm and work ethic are some of the qualities that I hope to pick up. I would also like to thank Prof. Nitin Chandrachoodan whose help with setting up the web server was invaluable.

Back in Sweden, I would like to thank Prof. Ingvar Gustavsson for helping me understand the overall structure of the VISIR project and tirelessly replying to our mails and providing all the help needed. I would also like to thank Johan Zackrisson for his patience in answering every question regarding the installation of software packages and Mohammed Tawfiq for providing the installation guide that was very helpful for installing web server.

I would like to thank my wingmate Anmol Kachroo for all the insightful talks we had on various issues and motivating me with his infinite enthusiasm in analog circuits. I would like to thank Shravan, Rajesh and others for providing a conducive and joyous atmosphere in the testing lab with their witty jokes. I would also like to thank my buddy friends Yashwanth and Chaitanya for helping me out with certain server issues.

Finally I would like to evince my gratitude to my parents, whose support and guidance throughout these years cannot be expressed in measurable terms.

ABSTRACT

KEYWORDS : VISIR ; Switching Matrix .

This project involves understanding and implementing the Virtual Instrument Systems In Reality (VISIR). VISIR is a remote laboratory for undergraduate electronic circuits practice. It allows a student to wire a real circuit remotely and get results from real instruments on their PC screen. It's purpose is to supplement the regular supervised lab sessions and enable students to perform experiments round the clock. The advantage of the system is in the fact that the same set of instruments can be used by many students through time-sharing.

It is realized by using two hardware components: a National Instruments PXI chassis that generates and measures signals and a Switching Matrix that accomodates circuit components which are connected through relays. These hardware components are controlled using various servers installed on the host computer.

This work reflects the acquired experience during the set up and the installation process of VISIR. The guide contains all the stages of the installation and the necessary configurations required for the VISIR start-up, correct usage and administration.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

API	Application Programming Interface
DPST	Double Pole, Single Throw
GPIB	General Purpose Interface Bus
GND	Ground
HTML	Hyper Text Markup Language
IP	Internet Protocol
IVI	Interchangeable Virtual Instruments
LabVIEW	Laboratory Virtual Instrumentation Engineering Workbench
LXI	LAN eXtensions for Instrumentation
NI	National Instruments
PXI	PCI eXtensions for Instrumentation
PC	Personal Computer
SSL	Secure Socket Layer
SPST	Single Pole, Single Throw
TCP	Transmission Control Protocol
TLS	Transport Layer Security
USB	Universal Serial Bus
VISA	Virtual Instrument Software Architecture
VISIR	Virtual Instrument Systems In Reality

CHAPTER 1

Introduction

In most of the undergraduate universities around the world, there are a number of electronic laboratory workbenches where students , mostly in groups perform experiments under the supervision of instructors. In most cases, students are not allowed to enter the lab outside lab hours and the time cost of students learning the basics and conducting experiments in supervised sessions is enormous. In order to help the students understand the concepts while simultaneously taking significant load off the instructors, the VISIR project was born in Blekinge Tekniska Hogskola (BTH) which aims to create a grid laboratory where the nodes are online lab workbenches, distributed among a number of universities or other organizations. My goal would be to understand this project, implement it and write a manual laying the groundwork for future work on this project.

1.1 Motivation

Though there are a number of remote laboratories in the world that are used in different disciplines, many of them have interfaces that are difficult to understand. VISIR project has the advantage that the interface used has a high resemblance to any normal undergraduate student's workbench. It can handle more than one users at a time and once installed offers a very low cost per student added. This laboratory can help students to prepare for supervised learnings in a big way. It also offers the advantage of community effort and can be scaled up for performing more demanding tasks.

1.2 Organization

Chapter 2 discusses the hardware components.

Chapter 3 discusses the software design.

Chapter 4 explains the installation process.

Chapter 5 explains the operation of the entire system.

Appendix A describes the control of hardware using LabVIEW software in standalone mode.

CHAPTER 2

HARDWARE DESCRIPTION

2.1 National Instruments PXI-Platform

Common undergraduate laboratory equipment like Oscilloscope, Function Generator, DMM and DC power supply are replaced with an equipment platform which is suitable for remote control using serial communication like PXI (PCI eXtension for Instrumentation), LXI (LAN eXtension for Instrumentation), or GPIB(General Purpose Interface Bus). In the current case we are using a PXI platform. The PXI platform consists of instrument module cards, a controller card and a chassis in which all the cards are placed which are all manufactured by National Instruments.

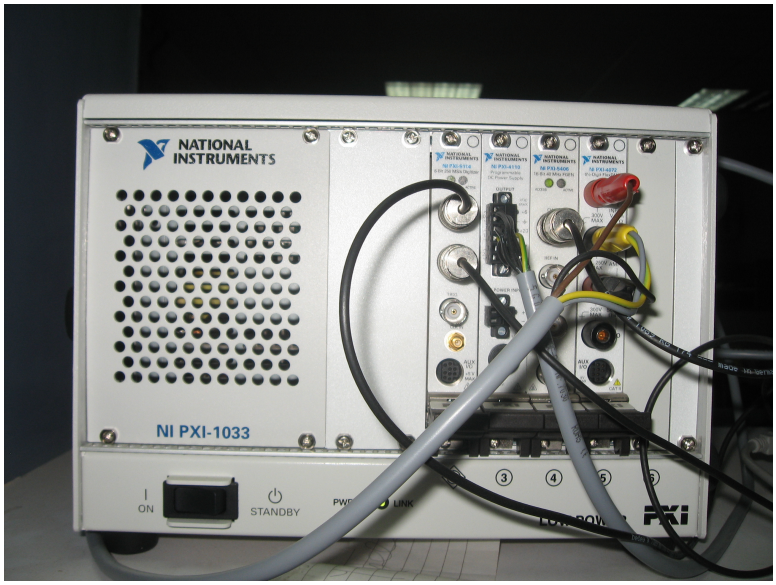

Figure 2.1: NI PXI Platform

Table 2.1: Components of NI PXI platform

	NI PXI-Chassis	NI PXI-Modules	NI PXI-Controller
Function:	It is the backbone of the PXI system in which all the instrument cards(NI PXI-modules) are plugged into.	These module cards substitute the instruments and are digitally controlled. They are plugged into the PXI-Chassis and can be added or removed as per the demand.	The PXI is controlled by the PC using NI PCIe host card. The PCIe host card is plugged into PCI Express slot of PC and connected to PXI through MXI-Express x1 cable.
Model Used:	NI PXI-1033	DC Power Supply(NI PXI-4110) Digital Multimeter(NI PXI-4072) Function-Generator(NI PXI-5412) Oscilloscope(NI PXI-5114)	NI PCIe host card

Various components of this PXI platform and their functions are listed in the table 2.1.

2.2 Relay Switching Matrix

A Relay Switching Matrix can simply be assumed to be a robot making connections and wiring circuits according to instructions received through serial bus. It has been wired to components and instruments through relays which can be controlled It has been manufactured in BTH. It consists of a stack of PCI/104 sized boards and can be used for low frequency electronics circuit experiments. These boards are modular and many such boards can be plugged into the matrix as per convinience. This switching matrix should be handled with extreme care and the pins on the board should not be touched directly as it might cause ESD (Electro Static Discharge).

2.2.1 Types Of Boards

Broadly speaking there are two types of matrix boards: one for connecting instruments and one for holding components or connecting external circuits. Figure 2.2 shows various boards that comprise the current relay switching matrix.

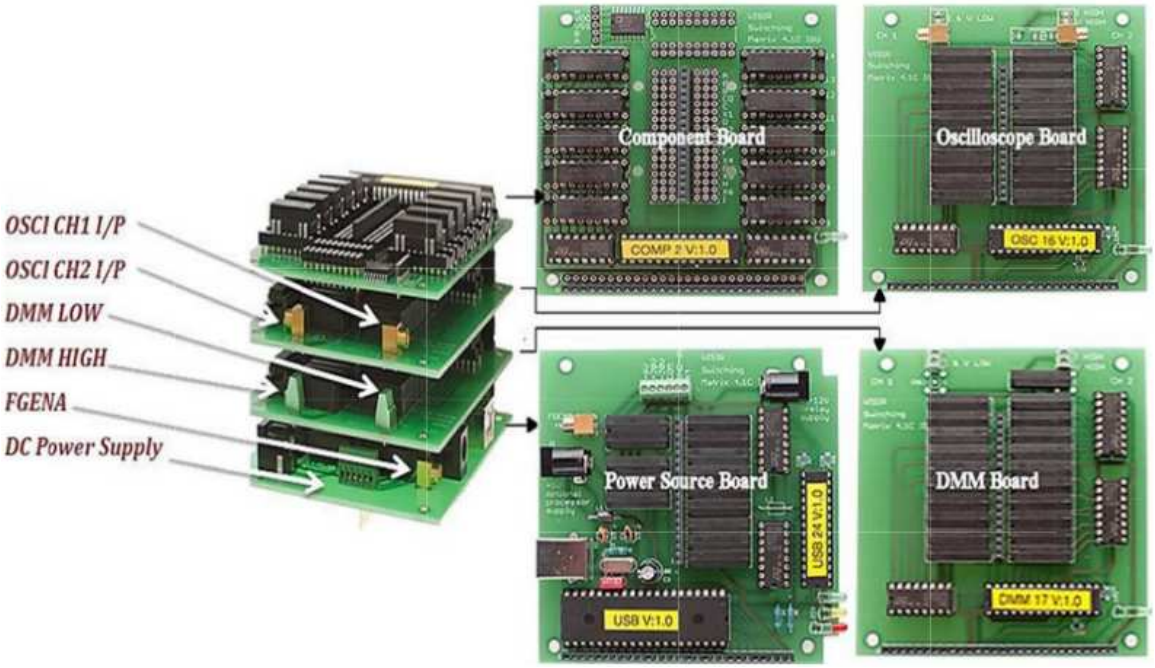

Figure 2.2: Various components of Relay Switching Matrix at a glance

The instrument boards are connected to the PXI Modules in the chassis through coaxial cables or cords, source board to power supply, DMM board to multimeter and oscilloscope board to oscilloscope in NI PXI-1033. The component boards have sockets in which components can be installed. One component board can carry 10 components with two leads or as many components with more than two leads as can be installed in the on- board 20 pin IC socket. The component board also contains a digital potentiometer AD7376 which can be used if needed.

2.2.2 Nodes

There are two stacking connectors on each board which connect the boards. One of the connectors(the central one) called node connector propagates nodes to every board. The notation Node refers to the fact that each conductor created by

the stacked node connectors can be a node in a desired circuit. Only nodes A to I and node 0 can be used to serve as the nodes of the circuit to be built which limits the maximum number of nodes of any circuit that can be built to 10. Node 0 is connected to digital GND. Furthermore, this node bus contains seven auxiliary nodes denoted X1 to X6 and COM. These nodes are connected internally to the power supply through relay switches and they are not to be directly connected to components.

2.2.3 Relays

These relays regulate the connection between the nodes and instruments or components. In the matrix that is being used, two types of relays are used: DPST(Double Pole Single Throw) and SPST(Single Pole Single Throw). Figure shows a two-lead component connected to two nodes through a DPST relay.

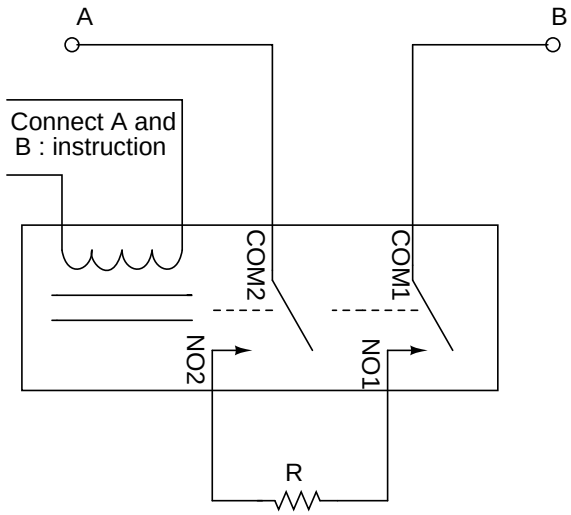

Figure 2.3: DPST Relay Connecting Two Nodes

2.2.4 Internal Connections

A close-up view of the component board and its internal connections are shown in figure 2.4 and figure 2.5 respectively. As shown the leads of the components are to be connected to the sockets on the outer side of the relays in the component board.

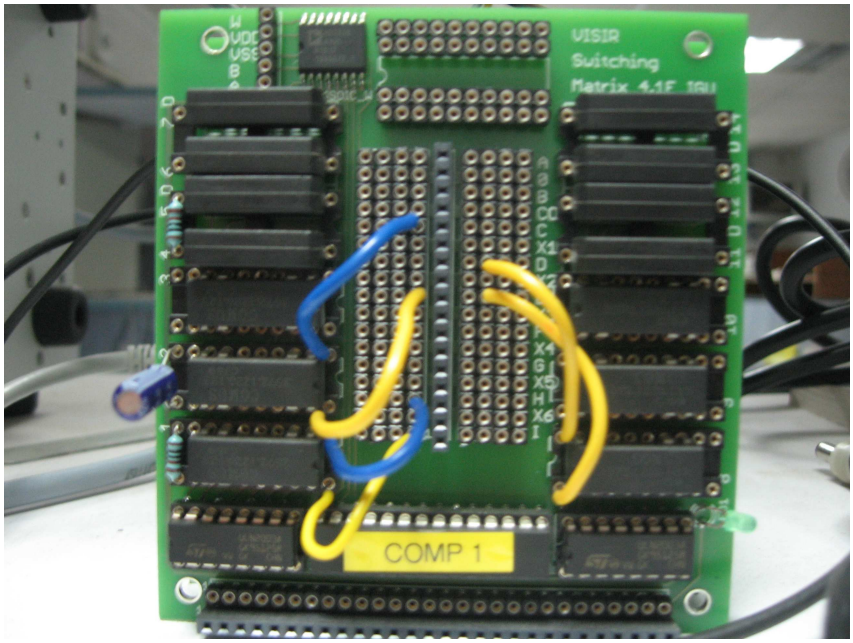

Figure 2.4: Component Board

Figures 2.6, 2.7 and 2.8 depict the internal connections of the source board, oscilloscope board and DMM board respectively. The ground terminals of function generator and oscilloscope are hardwired to node 0. The function generator signal can only be connected to node A. Oscilloscope channels and DMM channels are dynamically connected to any node depending on the circuit design. The power supply channels (0,COM,+6,+20,-20,AUX) are connected to the nodes (0,COM,X1,X2,X3,X4) and X1-X6 internally. Channel 0 is hardwired to the digital ground node 0 while the others are connected through relays in source board.

The maximum number of component boards one can plug into the matrix is 15 and hence the maximum number of components one can use is 150. But in the matrix that we have, there are four component boards, which would mean a maximum of 40 components. Also, the maximum current allowed through the relay is 2A and its minimum life expectency is 3×10^8 operations which is approximately 2 operations per second for 5 years.

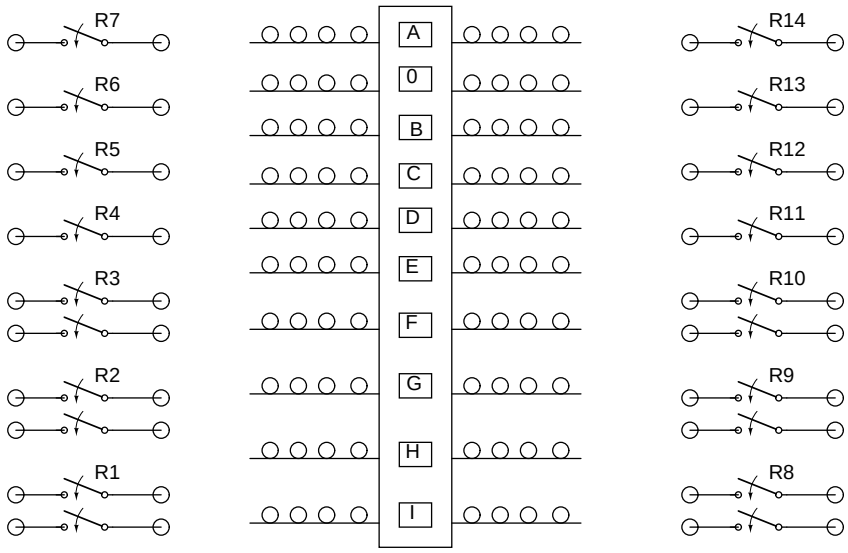

Figure 2.5: Component Board connections showing only the cicuit nodes A to I and node 0

2.2.5 Control Of Matrix

The matrix is microprocessor controlled. A Matrix Controller (PIC18F4550) hosted on the source board communicates with Board Controllers (PIC16F767) on each board via an I2C bus. The I2C address scheme is listed in Table 2.2. Instructions for switching the relays on and off are given to the microprocessor from the personal computer through a USB cable. This switching matrix is treated as a USB device by the computer after the appropriate driver is installed.

Appendix 1 describes how to test this Switching Matrix using LabVIEW software.

Table 2.2: I^2C Address Scheme

Board Type	I^2C Address (Board Number)	Label On The Microprocessor
Component Board 1	1	COMP 1
Component Board 2	2	COMP 2
Etc.		
Oscilloscope Board	16	OSC 16
DMM Board	17	DMM 17
Etc.		
Source Board	24	SRC 24

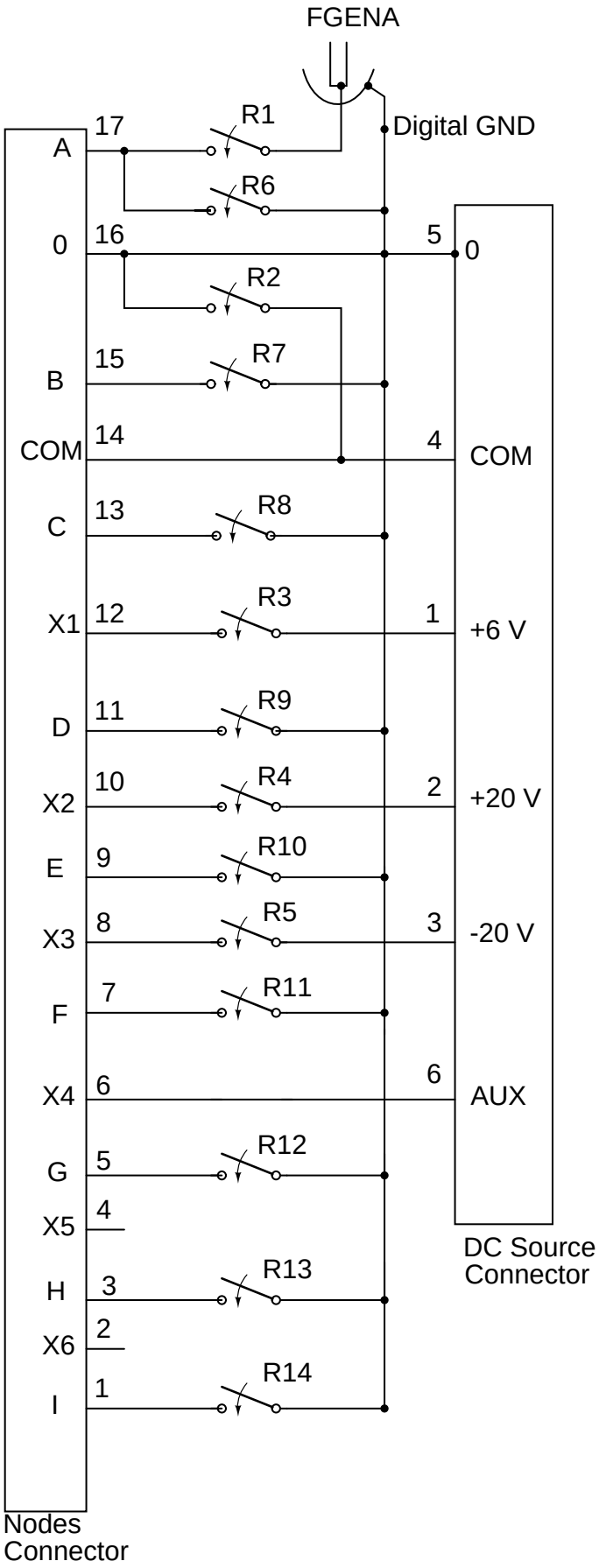

Figure 2.6: Internal Connections of Source Board

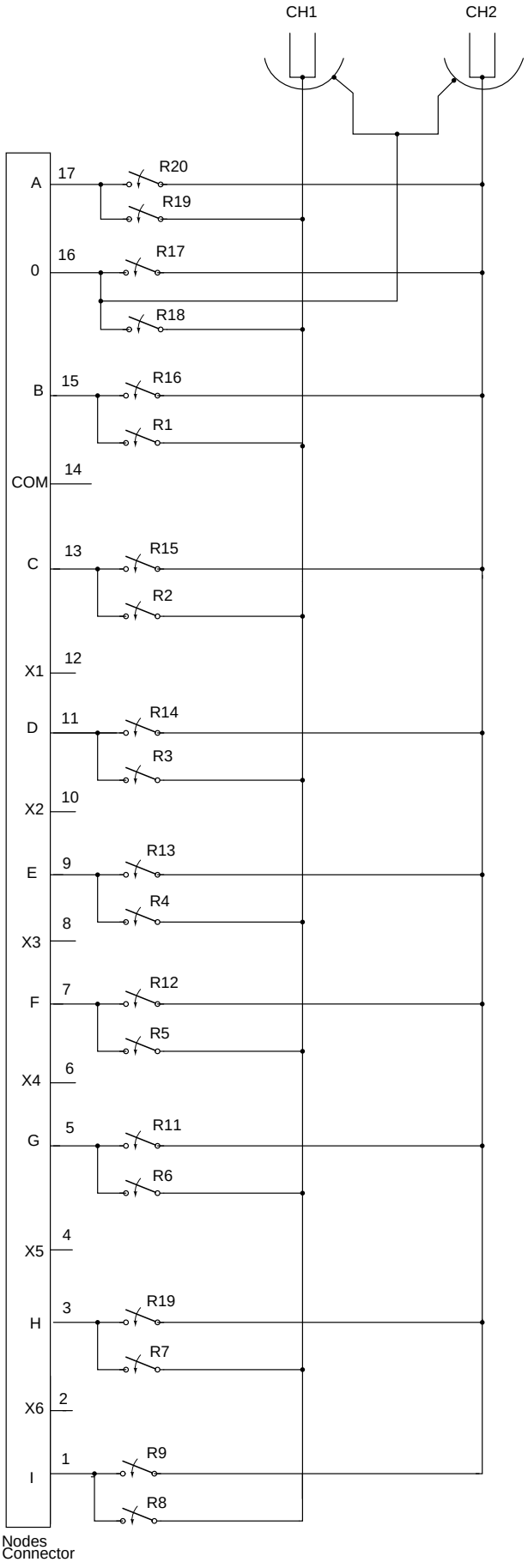

Figure 2.7: Internal Connections of Oscilloscope Board

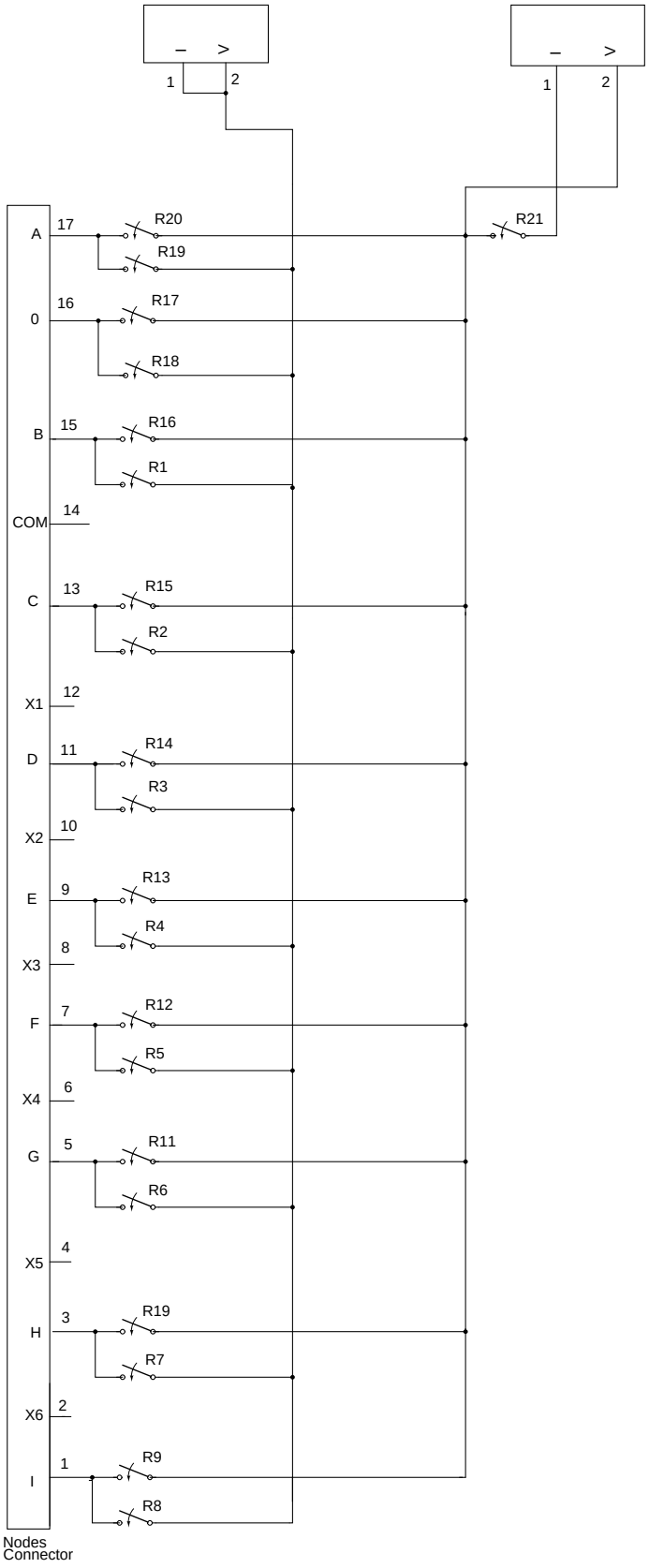

Figure 2.8: Internal Connections of DMM Board

CHAPTER 3

SOFTWARE DESCRIPTION

The hardware which was described in the previous chapter is controlled by a software written in LabVIEW through a serial cable. There are a set of other softwares which communicate with it and provide services to the user who sends requests from his personal computer.

The software section of this project can be divided into three parts: *equipment server*, *measurement server* and *web server*. These three servers sort of form three layers over the hardware equipment. The user first interacts with the *web server*, which authenticates him and then sends the experiment information in the form of XML requests to the *measurement server*, which then does some processing and then checks if the requested circuit can be built. If it is allowed, the information is sent in a XML file to the *equipment server* which instructs the hardware to perform the desired experiment and give back the results which are sent back to the user via *measurement server* and *web server*.

All these softwares are distributed under GNU-GPL as part of the VISIR project. It can be downloaded from the Openlabs developmental webpage of the BTH university or from a subversion server also maintained by the same university for a more recent and updated version.

3.1 Equipment Server

This server has been written in LabVIEW. It is the first layer around the equipment handling low level instrument interfaces and hosting controlling all the hardware comprising PXI platform and Relay Switching Matrix. All the instrument drivers are installed in LabVIEW which are IVI (Interchangeable Virtual Instruments) compliant.

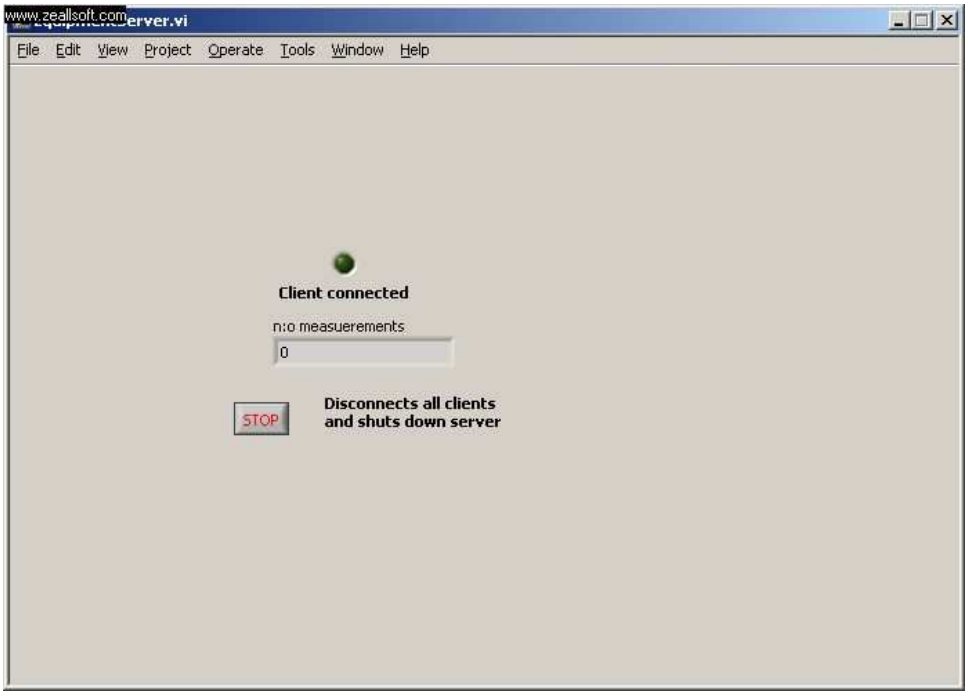

Figure 3.1: *Equipment server* Window

3.1.1 IVI Standards

Most of the undergraduate electronic laboratories in the world have common equipment with common functionalities like oscolloscope, function generator, multimeter, DC power supply and breadboard (some of which are IVI instrument classes) regardless of their make and model. The VISIR project currently uses the PXI platform, though other universities might want to use other platforms like LXI or GPIB. To clear it all up, VISA, IVI and SCPI are all various organizations which set standards for interface languages or APIs to communicate with test instruments from computers. Currently IVI maintains all the standards which essentially means IVI is setting the industry standards.

IVI drivers can be custom specific which are unuique to the instrument and not standardized, class drivers which can comply with an instrument class specification, or class-compliant specific drivers which contain base class capabilities as well as class-extension capabilities. Base class capabilities are common to most instruments in a class (e.g., edge triggered acquisition on a scope) and class extension capabilities represent more specialized features of an instrument (e.g., width trigger on a scope).

On the other hand PXI (PCI eXtensions for Instrumentation), LXI (LAN eXtensions for Instrumentation) and VXI (VME eXtensions for Instrumentation) are all modular electronic instrumentation platforms which are used to build test equipment or automated systems like the VISIR project.

Figure 3.2 shows the IVI architecture describing how the user softwares communicate with Instrumentation hardware. VISA library is the predominantly used I/O library which are used by drivers to communicate with the hardware. VISIR project recommends using instruments that comply with IVI-C class compliant specific drivers to realise interchageability so that all the workbenches in various universities can be linked in the future to form a grid.

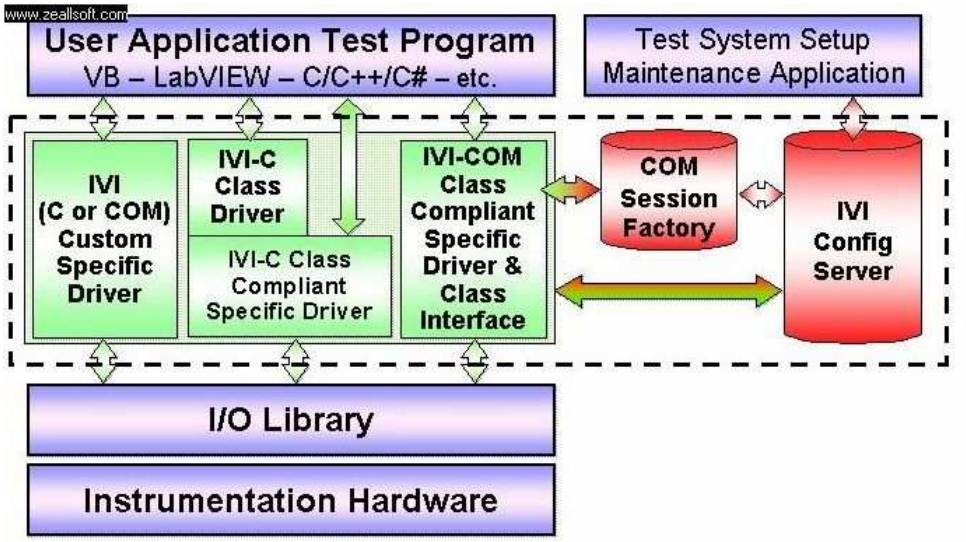

Figure 3.2: IVI System Architecture

3.1.2 Components List

The components installed on the Relay Switching Matrix have to be informed to the *equipment server* and *measurement server* so that they can take requests and instruct the matrix to wire them in. This is done using a *.list* file called *Components.list*. *.list* along with *.max* (described in subsequent seection) are file formats of configuration files for the switching matrix. Components.list contains all the components' types, values, board label numbers, relay numbers, node connections and descriptions which are plugged in the switching matrix. In addition to the

components, it also contains information of some instruments like function generator and power supply while that of DMM and oscolloscope are not needed as they are not altering the circuit per se.

The basic format for listing two-lead components and instruments is as follows:

(Component Type)_(Board No.)_(Relay No.) (Node1) (Node2) (Value)

Here the component type does not strictly mean component type, rather it means the code given to the actual component type by us in the library.xml file in flash client, which will be discussed later.

Here are a few examples:

- R_1_2 A D 10k : This means a resistor of resistance 10k ohm is connected between the nodes A and D through the relay number 1 on board 1 in the switching matrix.
- SHORTCUT_3_4 E F : This means a short or wire is connected between the nodes E and F through the relay number 4 on board 3.

As we have seen that some relays on the switching matrix are single pole relays for a two-lead component to be connected to the nodes we need to use two such relays. Generally speaking if two or more relays are to be switched on to enable a component to connect to the specified nodes then we seperate the two relay connections by a ':' sign. For example,

R_1_6:1_7 A C 5k

would mean that the relays 6 and 7 on board 1 have to be activated to connect the 5 ohm resistor between A and C.

Table shows some predefined types of components and instruments.

These defined types are used to map the real components on the switching matrix to the virtual ones in the flash module. Again, it is to be noted that these types are defined in the library.xml file in the flash module in *web server* where new component types can be defined. In addition to these, PROBE and DMM_DMM are the other two types which represent the oscilloscope probe and DMM probe respectively and are not mentioned in the *components.list* but appear

Table 3.1: Some Predefined Types of Components

Type	Component
R	Resistande
C	Capacitor
L	Inductance
OP	Operation Amplifier
POT	Potentiometer
Q	NPN Transistor
VFGENA	Function Generator
SHORTCUT	a Short or wire
VDC+20	DC power 20 V terminal

in the solved netlist of *measurement server*.

Regarding the instruments, function generator and power supply terminals are connected to specific relays on the board 24. The ground of function generator is hardwired to 0 node and the signal lead is connected to the node A internally through the relay number 1. Hence we have to represent it in the *components.list* as

VFGENA_24_1 A 0

The auxiliary nodes X1 to X6 and COM are not supported to be included in *components.list*. Some of them are connected to the power supply terminals internally through relays on board 24. Table describes the connections among them.

Table 3.2: Internal Connections Between Auxiliary Nodes And Power Supply Terminals

Power Supply Terminal	Auxiliary Node	Connecting Relay
+6V	X1	3
+20V	X2	4
-20V	X3	5
GND	0	hardwired
COM	0	2
COM	COM	hardwired

There are two ways to connect power supply terminals to actual circuit nodes (A to I) :

1. By shorting the appropriate power supply node (X1, X2 or X3) to any node from A to I nodes. Node COM is already connected to node o through the relay number 2. For instance, an entry in *components.list* corresponding to such a short connecting X1 to B would look like:

   ```
   VDC+6V_24_3  B
   ```

2. By connecting the appropriate power supply node (X1, X2 or X3) to any node from A to I nodes through a relay (preferably a single pole relay) on component board. For instance, an entry in the *components.list* corresponding to such a connection between X1 and B through the relay number 7 on board 1 would like:

   ```
   VDC+6V_24_3:1_7  B
   ```

In case of components having more than two pins, we use a different format:

(Component Type)_(List of Board_Relay Nos) (List of Nodes in Increasing Order of Pin No.) (Value)

For example, in the case of an Op-Amp, for the connections in figure 3.3 described by Table (here all the relays are on board number 1), the entry in *components.list* would be

OP_1_14:1_11:1_10:1_7:1_6 NC1 B D F NC2 C E NC3 UA741

Here, NC stands for Not Connected.

Table 3.3: Connections of Op-Amp on Component Board 1

Op-Amp Pin No.	Relay No.	Node Connected To
1	NC	NC
2	14	B
3	11	D
4	10	F
5	NC	NC
6	7	C
7	6	E
8	NC	NC

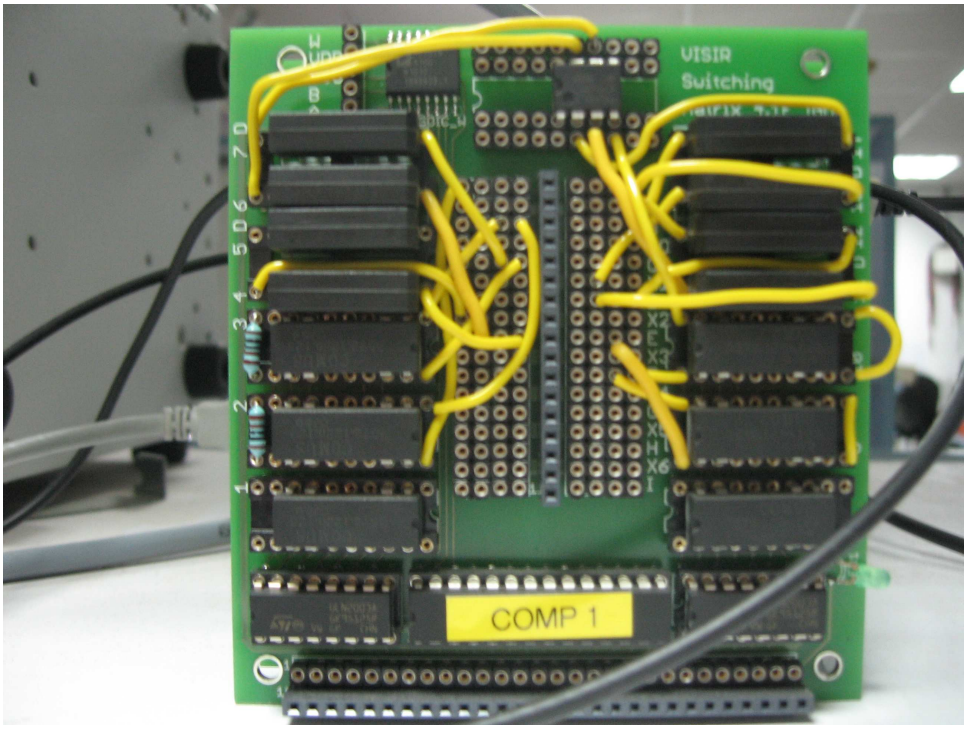

Figure 3.3: Component board showing opamp

3.1.3 Process Flow

While starting *measurement server*, *equipment server* sends *components.list* to it which is compared with maxlists to check for errors. The *equipment server* then receives a validated sequential experiment protocol (XML based format) from the *measurement server* over TCP/IP through the port 5005 and executes it through the connected instruments. After that, the results which contain measurement values are sent back to the client through the same port.

3.2 Measurement Server

It is an executable software program written in C++ using Microsoft Visual C++ that forms a layer on the *equipment server* layer. Figure 3.4 shows the *measurement server* window. It communicates with the *web server* using a unique XML based *client protocol* through the port 2324. This protocol is based on requests and responses. The client builds an experiment string, a request using *web server* with cookie information in it and sends it to the server.

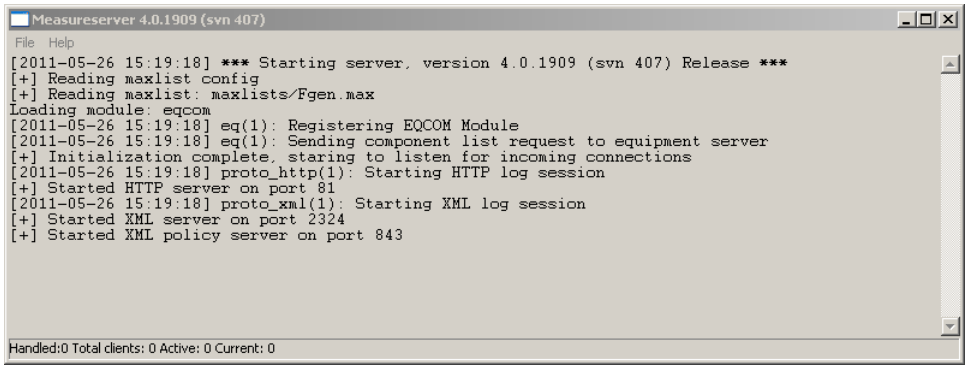

Figure 3.4: *Measurement server* window

The *measurement server* then handles this request and produces a response, which contains a unique-id, which signifies the establishment of a session, that is sent back to the user. This unique-id is used in all the subsequent requests by the user. Each request is sent in a seperate TCP session which means a connection is established, request sent and then it is closed. The size of this request cannot exceed 64kb.

For example a function generator description in a client request is as follows:

```
 <functiongenerator>
    <fg_waveform value="sine" />
    <fg_amplidute value="1000.0" />
    <fg_frequency value="1000.0" />
    <fg_offset value="0.0" />
    <fg_startphase value="0.0" />
    <fg_triggermode value="continous" />
    <fg_triggersource value="immediate" />
    <fg_burstcount value="0" />
    <fg_dutycycle value="0.5" />
    <fg_userdefinedwave length="20" encoding="BASE64">ABCD1234ABCD1234ABCD
</fg_userdefinedwave>
</functiongenerator>
```

3.2.1 Maxlists

While the *components.list* specifies all the components installed in the switching matrix and all the instruments, maxlists specifies all the safe or allowed circuits that can be built by a user. Thus it should be a subset of the components list. These files having a file extension of .max are used by the *measurement server* to confirm that the requested circuit is safe before passing on the request to *equipment server*. It also lists all the components and instruments and how they are connected to each other. Each entry has a format

(Component Type)_(Serial No.) (Node1)(Node2)(Node3)(etc.) (value)

The serial number can be any value which is chosen to name the particular component or instrument.

3.2.2 Process Flow

As soon as the client starts the session with the *web server* a cookie is generated and stored in the database. It is used in all the subsequent requests made to the *measurement server*. The *measurement server* compares the request-cookie with the stored-cookie and authenticates him.

It analyzes the received circuit connections and solves it using a program called circuit-solver written in C++ to get the netlists. This circuit-solver essentially maps the virtual circuit made on breadboard in *web server* flash client to actual netlists used to wire the circuit on the matrix. It then uses the maxlists to check if the circuit is permitted to be wired and then sends the processed client request specifying the connections to be made and instruments settings to the *equipment server* through the port 5005 over TCP/IP, which runs the experiment.

After the experiment is run the results are sent back as response to the user via *measurement* and *web server*.

If more than one user sends a request at a time then the *measurement server* puts them in a queuing system and sequentially executes them with regard to

reservations and priority etc. 1/16 second is the maximum time allocated for each request. To send the request to *equipment server* through the port 5005 over TCP/IP after the above two steps and send the reveived response back to the client via *web server*.

3.3 Web Server

A *web server* is a software which is most directly associated with a user. It provides him with a graphical user interface, handles authentication and sends the user request in a XML based format to the *measurement server*. Figure 3.5 shows the basic structure of *web server*.

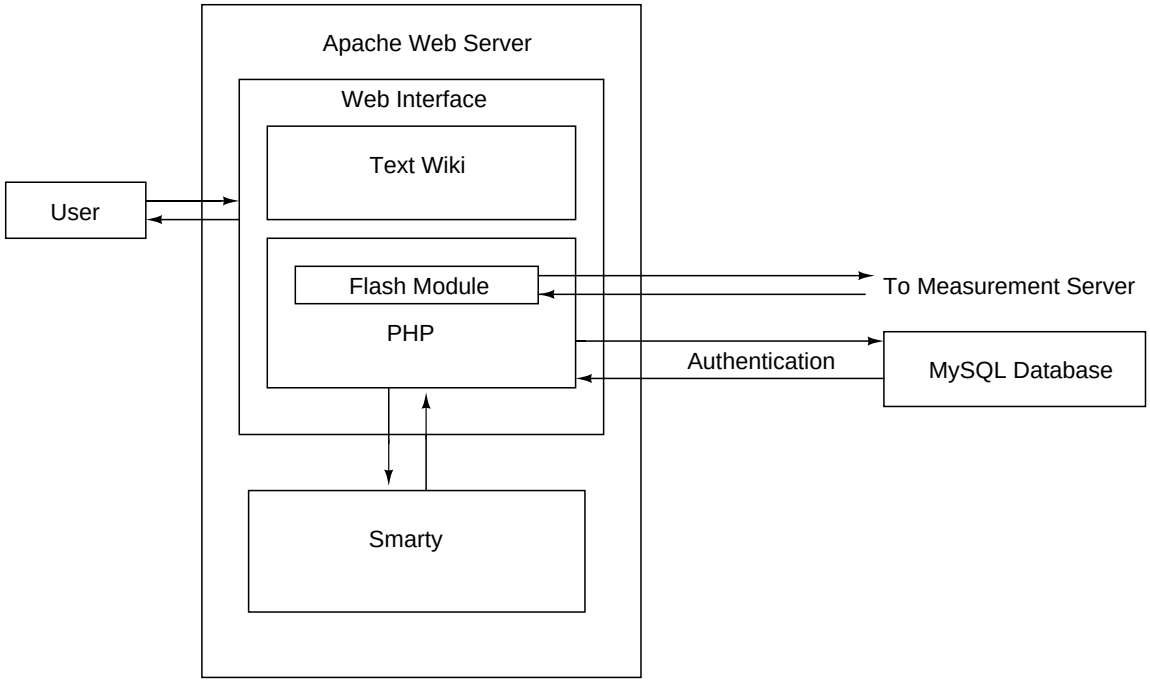

Figure 3.5: General idea of how the *web server* works

In this project, we use Apache server, MySQL database, PHP, Text_Wiki and Smarty. Apache server hosts the entire website, MySQL stores the user details and cookies that are used for authentication purposes and Text_Wiki and PHP using Smarty, create the web interface.

3.3.1 Web Interface

Here, by web interface, we mean the pages that the user is actually able to see. In simple terms,php scripts generate some variable values and put them in Smarty templates and create webpages. If the URL has 'page=' in it, then it is a wiki formatted page and if it has 'sel=' then it is a function in a php script, though the reverse is not true in all the cases.

Figure 3.6 shows the login screen. After logging in the user can see various options depending on his authentication type, which currently are guest, student, teacher and admin. More on this is explained in the Operation Cycle chapter.

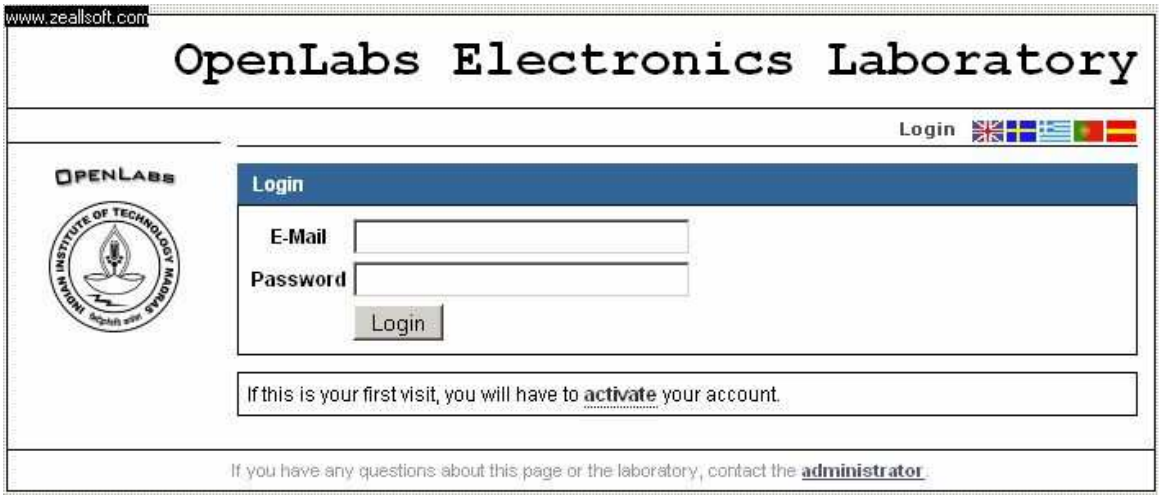

Figure 3.6: Login Page for Web Interface

3.3.2 Flash Module

The flash module is embedded in a php script which provides the user with the virtual workbench. It is used to create a schematic of the circuit intuitively which is then converted into a XML based request and sent to *measurement server* and then to equipment server where it is really wired up. This flash module has been written in Adobe Flash. Figure 3.7 shows the flash module .

Currently the following instruments have been installed in flash module:

- Breadboard
- Multimeter - Fluke 23

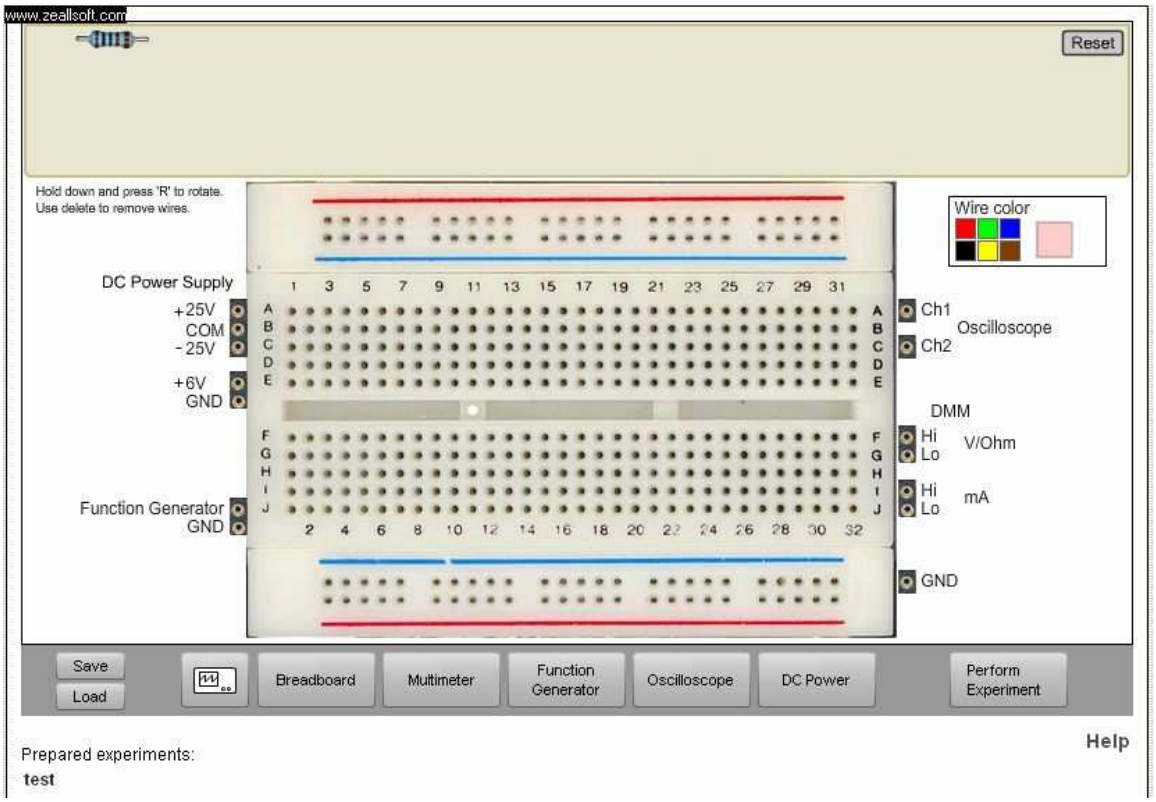

Figure 3.7: Flash Module Showing Virtual Breadboard

- Function Generator - HP33120A
- Oscilloscope - Agilent 54622A
- DC Power - Agilent E3631A
- National Instruments models of Oscilloscope, DC Power Supply, Multimeter and Function Generator.

Additional instruments can be included by designing their flash modules. The instruments provided are included in the instruments.xml file and breadboard components in library.xml. New components can be added easily as described in the operation cycle chapter.The modules are really intuitive and one can change the values using just his mouse and dragging of components, buttons and knobs.

For testing purposes one can use loader.html or loader.swf and change the config.xml settings as desired. But the preferred mode is always through the web interface after logging as admin and starting the client in teacher mode.

After understanding the hardware and software, one can broadly represent the entire project as shown in figure 3.8.

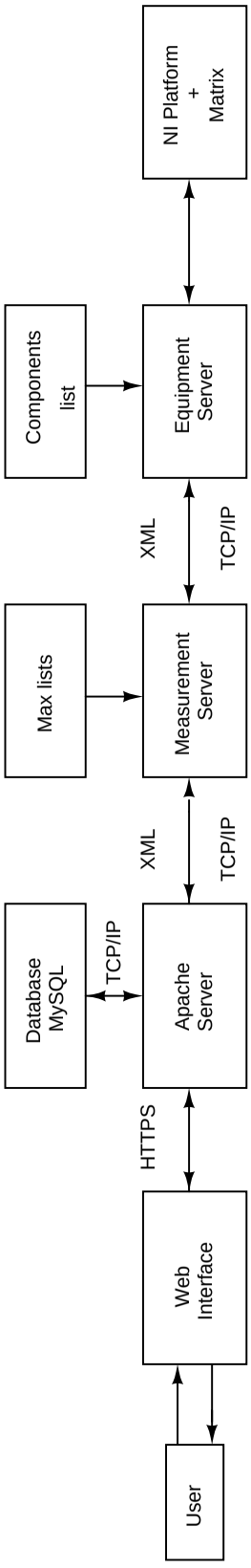

Figure 3.8: Block diagram for the remotely controlled electronics laboratory workbench

CHAPTER 4

SOFTWARE INSTALLATION

In this chapter we describe the installation process of the three server softwares along with the packages they require.

The server softwares can be downloaded either as binaries from the Openlabs development pages or as source files from their subversion server. To do this, we have installed TortoiseSVN which is a subversion client. A subversion is a software versioning and revision control software that maintains and logs various versions of the source files which is generally used in software development. For downloading the files, just right click on the destination folder and select 'svn checkout' and enter the url of the svn server. To update all the source files, you can use 'svn update' option in the same way.

4.1 Equipment Server

Firstly, before installing the NI PXI Chassis, we have to install their drivers. In the case of the Switching Matrix, when we first plug in the USB cable, we are asked for a .ini file which is provided along with it.

4.1.1 LabVIEW

Since the current *equipment server* has been written in LabVIEW 8.6 and the subsequent versions will written in 2009 version, it is better to install LabVIEW 2009 or 2010 version. We have installed the 2009 version with all the needed drivers as listed in the Chassis manual.

The server software, written as LabVIEW .vi files have been downloaded from the openlabs development website. The following are a few important points about LabVIEW:

- LabVIEW is a graphical programming language where there are a number of predefined blocks with different functions. It is modular and execution is determined by the structure of the graphical block diagram.
- When you open a vi file, you will see the front panel. If you press 'Control + T' it will show the block diagram.
- You can double click on a connector pane (block) of a subVI in the block diagram and enter its front panel.
- To see all the interconnections and the structure of the entire project containing various subVIs, you can select 'VI Hierarchy' in 'View' option in the menubar of front panel. If you want to see only the relationships of that particular vi file, then select 'Browse Relationships' in the same 'View' option.
- You can get the entire list of functions and VIs and search through them by right clicking in either front panel or block diagram.
- While debugging you can always see the error code and search for a text in the description or labels etc. of all the VIs in the server using 'Find and Replace' (Ctrl+F).

4.1.2 Measurement and Automation eXplorer - MAX

We can verify that the drivers have been correctly installed by opening measurement and Automation Explorer or MAX and checking under 'Devices and Interfaces' to see the chassis modules and under USB devices to see the Relay Switching Matrix as shown in the Figure **??**. We can also self test the equipment there.

4.1.3 Settings and Configurations

Open 'EquipmentServer.ini' and

- set Port=5005
- set Log File = C:/EquipmentServer.log
- check if the instrument addresses (e.g. PXI1Slot4) are same as mentioned in the MAX.
- edit component types which are used in *components.list* and maxlists and subsequently make the same edits in the component.types file in *measurement server* and .

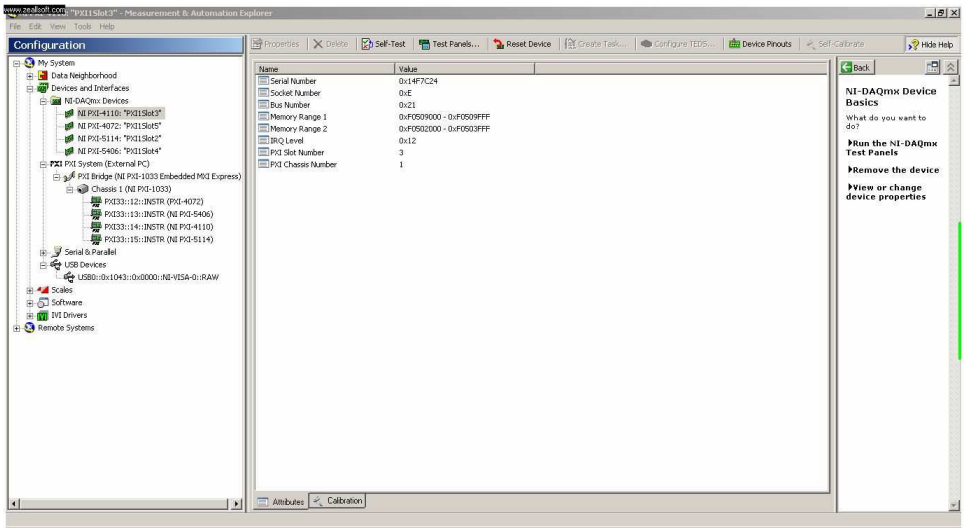

Figure 4.1: Measurement and Automation Explorer

- enter the data in the *components.list* as mentioned in the previous chapter.

4.1.4 Errors

When we initially executed the *EquipmentServer.vi* file, there was a minor error in DMM_control.vi where a default case had to be included in the inner case structure which has been corrected in the subsequent versions. Excepting this there were no other errors.

4.2 Measurement Server

We are using the downloaded binaries from the Openlabs Development pages as these are working fine without giving errors. There is an executable file measure-server_win.exe which can be run. Since it is built with Microsoft compiler, Microsoft runtime libraries (Microsoft Visual C++ Redistributable Package) should be installed before running it.

4.2.1 Compiling Source Files

We can also download the source files from the Openlabs website and compile them using Microsoft Visual C++ 2010 Express which is available for free. Some of the steps taken:

- Open the solution file and build the main project measureserver.
- For all the missing files errors add the file paths to the 'library path' and 'include path' boxes in the VC++ option.
- For all the LNK errors, add the file paths of the corresponding files in the 'linker' section in the properties of the particular module.

Unfortunately the error 'corrupt libusb.lib' could not be sorted out, though compiling the source files is not needed now and the binaries are working fine.

4.2.2 Settings and Configurations

Follow these steps to configure the *measurement server* :

- Firstly, write the appropriate .max file and store it in "/conf/maxlists/" folder and make sure that the maxlists are a subset of the *components.list*.
- Now include all the maxlists files created in the file *maxlists.conf* in "/conf/" folder.
- All the configuration details of the *measurement server* are present in the file 'measureserver.conf' in "/conf/" folder.
- 'Port' is set to 2324 which indicated the port through which it communicates with the *equipment server*.
- Set 'LogLevel' to 5 to enable detailed logging of all the processes.
- Set the 'BypassAuth' to 1 which means that we are bypassing the authentication of the user's cookie by checking it with the one stored in the database. If this authentication is to be enabled, then set it to 0 and add the following lines to the file 'measureserver.conf' :

```
LoadModules eqcom, dbmysql
DB.Host localhost
DB.Port 3306
DB.User root
DB.Password <password>
DB.Database electronics
```

Then copy the file 'libmySQL.dll' from the wamp server folder to the directory "/conf/" in the measurement server folder.

4.3 Web Server

The *web server* is implemented using the following packages (with version numbers in brackets):

1. WAMPServer (2.0i)
2. Text_Wiki (1.2.1)
3. Smarty (2.6.26)
4. Adobe Flash Professional - Flash Module (CS3)
5. Web Interface

The following sections detail the installation procedure for these packages which should be followed in the same order as mentioned:

4.3.1 WAMPServer

'WAMP' in WAMPServer stands for Windows, Apache, MySQL and PHP. Apache is the most widely used HTTP server, MySQL is for creating and managing databases and PHP is a scripting language that can manipulate information held in a database and generate web pages dynamically each time content is requested by a browser. The versions in use are 2.0 for WAMPServer, 2.2.11 for Apache, 5.1.36 for MySQL and 5.3.0 for PHP. It is preferred to use these versions because of certain compatibility issues. It has been installed in the "C:/wamp" folder. The default values for the installed packages are: server port is 80; server address is 'localhost'; server remote address is '127.0.0.1'; database port is 3306 and database user is 'root@localhost'. All the files that are kept in the "C:/wamp/www/trunk/sites/electronics/public" folder are accessible to public.

4.3.2 Text_Wiki

Text_Wiki transforms Wiki and BBCode markup into XHTML, LaTeX or plain text markup. We have to install Text_Wiki using PEAR installer. To install PEAR installer,

- Open Command Prompt
- Go to the directory "C:/wamp/bin/php/php5.3", where the file 'pear.go.bat' exists.
- Type the command "php -d phar.require_hash=0 PEAR/go-pear.phar" and press enter.

  Answer the questions that follow and then pear installer will be installed. Now to install Text_Wiki package,

- Open Command Prompt
- Go to the directory "C:/wamp/bin/php/php5.3.0"
- Type the command "pear config-set http_proxy

  http://username:password@proxy.iitm.ac.in:80 " and press enter to specify the proxy settings. Alternatively you could just download the .tgz file from the project page to the php5.3.0 folder and follow the remaining steps.
- Type the command: "pear install Text_Wiki" or "pear install Text_Wiki-1.2.1.tgz" in case of downloaded file and press enter.

4.3.3 Smarty

Smarty is a template engine for PHP facilitating the seperation of application logic from presentation.

- Download the package from the Smarty project website and extract it to "C:/wamp/Smarty".
- Open to edit the file "C:/wamp/bin/apache/Apache2.2.11/bin/php.ini"
- Under "Paths and Directories" type the line:

  include_path = "C:/wamp/bin/php/php5.3.0/PEAR;C:/wamp/Smarty/libs"

To check that Smarty has been installed properly:

- Create two folders: “C:/wamp/www/smarty/configs”
  and “C:/wamp/www/smarty/templates”
- Create a template file ”index.tpl” in the folder “C:/wamp/www/smarty/templates” with the contents:

```
<html>
<body>
Hello, {$name}!
</body>
</html>
```

- Open to edit the file “index.php”, which is located in the folder “C:/wamp/www” and add the following lines in the end:

```
 <?php
// load Smarty library
require ('Smarty.class.php');
$smarty = new Smarty;
$smarty->template_dir = 'C:\wamp\www\smarty\templates';
$smarty->config_dir = 'C:\wamp\www\smarty\configs';
$smarty->cache_dir = 'C:\wamp\Smarty\cache';
$smarty->compile_dir = 'C:\wamp\Smarty\templates_c';
$smarty->assign ('name','World!');
$smarty->display ('index.tpl');
?>
```

- Now start the WampServer and go to its homepage at “http://127.0.0.1/” or at “http://localhost/”
- If you see “Hello, World!!” at the end of the page below the WampServer options it is installed correctly.

4.3.4 Flash Professional

The flash module which offers the virtual workbench environment is created using Adobe Flash Professional CS3. It is written as a flash project (.flp), which is not supported by the latest CS5 version and hence using CS3 is recommended. The source files can be downloaded from the openlabs subversion server. Follow the steps listed below to set up the flash module in *web server*:

- Before building the project, run the batch file "prepare_output.bat" for creating the directory structure in a folder named 'output'. After building, all the .swf files are stored in the respective folders in "output".

- In the output folder where are flash files are built, create a folder 'images' and move all the images of the instruments to this folder as the loader.swf seeks the images in that location.

- Copy the entire output folder to the location

  "C:/wamp/www/trunk/sites/electronics/public".

- Open to edit the "C:/wamp/www/trunk/sites/electronics/public/flash/config.xml" file. Replace its contents with the following;

```
 <configuration>
<!-- <hostname>localhost</hostname> -->
<!-- <port>2324</hostname> -->
<hostname>10.7.9.101</hostname> # ip address of the host machine
<port>2324</port>
<http>0</http> #this lets requests to be made on http
<teacher>1</teacher> #this lets the user to add

 components to the breadboard
<ignorecookie>0</ignorecookie>
<httpurl>http://10.7.9.101:8080/measureserver</httpurl> #in case

of http, url of measurement server
</configuration>
```

- If the flash module is run as a standalone program by opening either loader.html or loader.swf without giving it a session cookie, it will display a login screen for the system as shown in figure 4.2. This is when we want to run the flash module on pages other than our website by embedding it in a LMS (Learning Management System), with access to the php scripts in the ".../public/flash" folder. The ignorecookie option should only be enabled while testing the flash module. The teacher mode allows us to add components to the breadboard. These components are predefined and present in a drop down box in the flash module as shown in the figure 4.4.

- There are two ways in which a new component can be added to the aforementioned menu: a new component of an existing type like a resistor of a new value or an entirely new component with a new type.

  In the former case, we just need to have an exact cropped picture of the component and add it to "...flash/breadboard/images/" folder and add the data to "...flash/breadboard/library.xml" as shown below (in the case of a 270 ohm resistor):

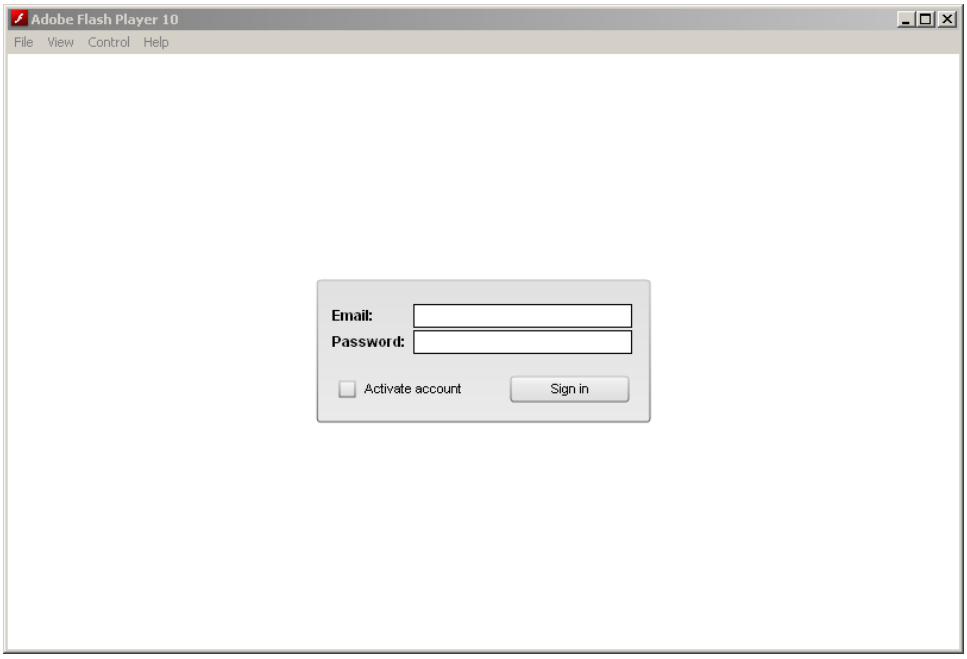

Figure 4.2: Login screen of flash in standalone mode

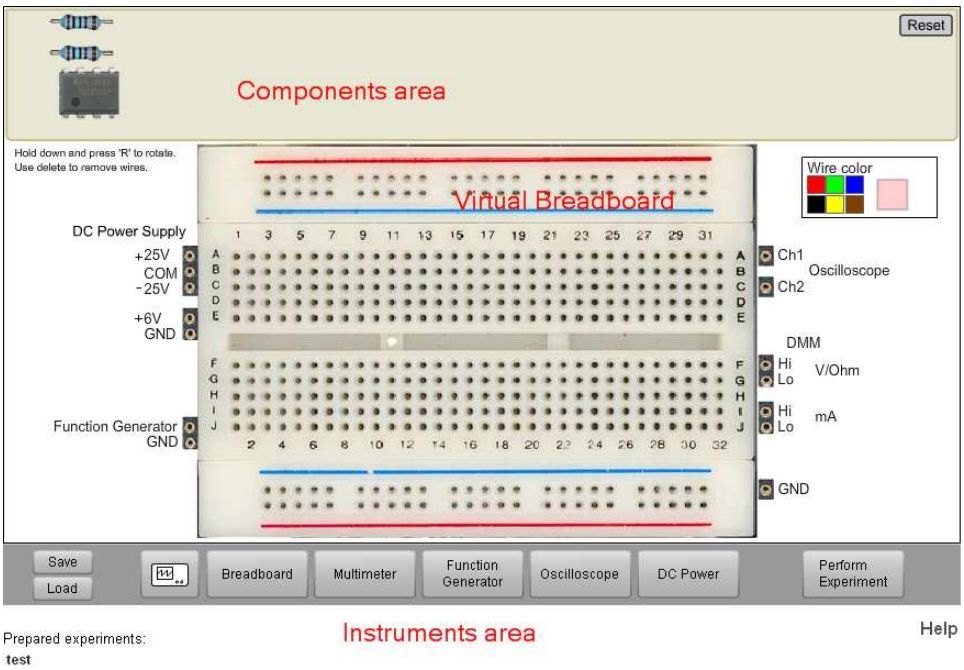

Figure 4.3: After logging in, the virtual breadboard of flash module looks like this

```
<component type="R" value="270" pins="2">
        <rotations>
          <rotation ox="-27" oy ="-6" image="r_270.png" rot="0">
             <pins><pin x="-26" y="0" /><pin x="26"  y="0" /></pins>
          </rotation>
          <rotation ox="-7" oy ="-27" image="r_270.png" rot="90">
             <pins><pin x="0" y="-26" /><pin x="0" y="26" /></pins>
```

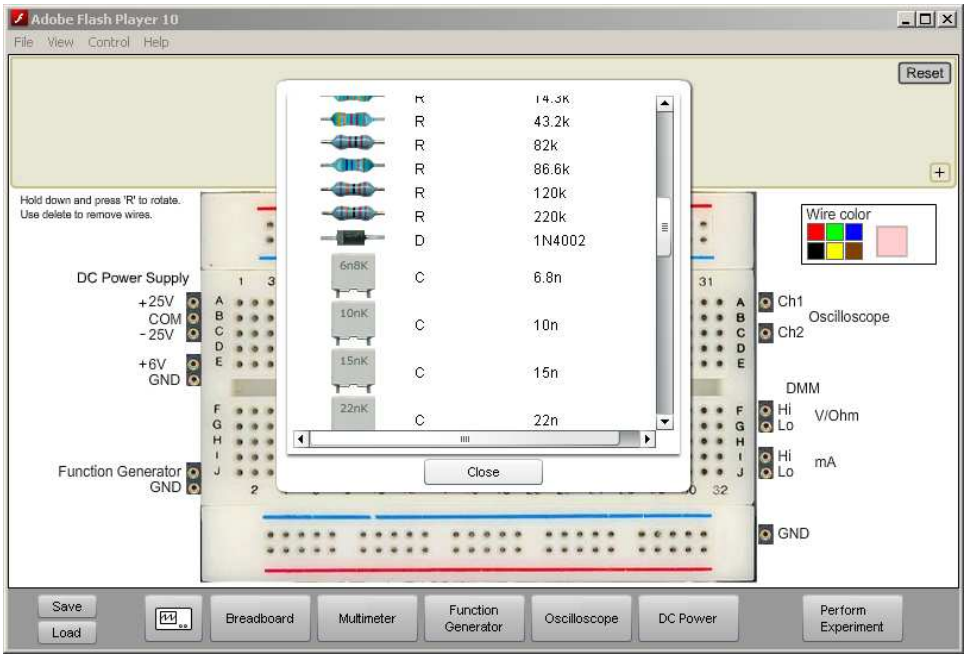

Figure 4.4: Components drop down menu in flash module

```
            </rotation>
          </rotations>
</component>
```

In the latter case, we need to :

1. Open to edit the file "EquipmentServer.ini" in the equipmentserver folder and add the "desired type:number of pins" under the heading "Component type".
2. Open to edit the file "component.types" in "measurement server/conf/" folder and add the desired type with the appropriate parameters.
3. Now follow the same steps as mentioned for the previous case.

4.3.5 Web Interface

This web interface software is written in PHP and is hosted by the Apache server installed as mentioned above. It can be downloaded from the Openlabs subversion server. The downloaded 'trunk' folder should be placed in the 'C:/wamp/www' folder. Now a list of configurations is to be done to get the *web server* running.

Database

Follow these steps to create the database which stores user information:

- Go to the phpmyadmin (http://localhost/phpmyadmin/)
- Go to "Run SQL query//queries"
- Type the command "create database electronics" and execute it by clicking on 'Go' to create a database called 'electronics'.
- Now populate it with contents by running 'create_database.php' by opening the link "http://localhost/trunk/sites/electronics/dbscripts/create_database.php"

To create an admin account:

- Copy the file "C:/wamp/www/trunk/sites/electronics/dbscripts/create_admin.php" to the folder "C:/wamp/bin/php/php5.3.0".
- Open the file "C:/wamp/bin/php/php5.3.0/create_admin.php" and change the code line 3 from

  ```
  require_once ''..config.php''
  ```

  To:

  ```
  require_once ''C:\wamp\www\trunk\sites\electronics\config.php''
  ```

- Open the command prompt and go the the directory "C:/wamp/bin/php/php5.3.0"
- Type "php create_admin.php testinglab.iitm@gmail.com password" and press enter.
- If any error appears because of a missing file, search for the file and include its path in 'require_once' line in its calling file.

Now to check if the administrator account is created correctly,

- Go to phpmyadmin (http://localhost/phpmyadmin/)
- Open the database 'electronics' and look in the table 'anv'. The created admin account should be present there.

SSL/TLS

The connection between the user's computer and the *web server* is SSL encrypted. Therefore, the SSL/TLS protocol should be installed to allow HTTPS (Hyper Text Transfer Protocol Secure) by following the steps mentioned below:

Add an environment variable 'OPENSSL_CONF'

- Go to Control Panel >System > Advanced > Environment Variables
- Add a new system variable: 'OPENSSL_CONF' ,
  Value: "C:/wamp/bin/apache/Apache2.2.11/conf/openssl.cnf"
- Log out and then log into windows.
- Go to command prompt and type 'set | more' and press enter.
- Make sure that the variable "OPENSSL_CONF=
  C:/wamp/bin/apache/Apache2.2.11/conf/openssl.cnf"
  exists among the listed variables

The following steps lead to the creation of security certificates for website.

Open command prompt and go to directory

"C:/wamp/bin/apache/Apache2.2.11/bin"

To generate key:

- Type the command "openssl req -new > webserver.csr" and press enter.
- Now you are prompted to enter a PEM pass phrase and verify it. Remember it because you will need it later. It will then ask you to enter information that will be incorporated into your certificate request. When the command finishes, it has created several files, including "privkey.pem", in "C:/wamp/bin/apache/Apache2.2.11/bin".

To write RSA key:

- Type "openssl rsa -in privkey.pem -out webserver.key" and press enter.
- Enter the pass phrse obtained from the previous step. Now, the RSA key is written and 'webserver.key' is created in the folder.

To create signed certificate:

- Type “openssl x509 -in webserver.csr -out webserver.cert -req -signkey webserver.key -days 365”
- We have just created a certificate that is valid for 365 days.

Now, we have to store the generated certificate files.

- Create the folder “C:/wamp/OpenSSL” with the following subfolders: ’certs’, ’crl’, ’newcerts’, ’private’.
- Copy the files ’webserver.cert’, ’webserver.csr’ and ’webserver.key’ from the folder “C:/wamp/bin/apache/Apache2.2.11/bin”
  to the folder “C:/wamp/OpenSSL/certs”
- Copy the files ’.rnd’ and ’privkey.pem’ from the folder
  “C:/wamp/bin/apache/Apache2.2.11/bin” to the folder “C:/wamp/OpenSSL/private”.

Create error log files for SSL connections: ’ssl_ErrorLog.txt’ and

’ssl_TransferLog.txt’ in the folder ”C:/wamp/logs”.

Now we have to modify the configuration file of SSL. Go to the folder

“C:/wamp/bin/apache/Apache2.2.11/conf/extra” and make a backup of ’httpd-ssl.conf’ and modify it as listed below (replace the first line entries with the second line ones):

1. DocumentRoot “C:/Program files/Apache Software Foundation/Apache2.2/htdocs”
   DocumentRoot “C:/wamp/www”
2. <Directory “C:/Program Files/Apache Software Foundation/Apache2.2/cgi-bin”>
   <Directory “C:/wamp/www/”>
3. SSLSessionCache “shmcb:C:/Program Files/Apache Software
   Foundation/Apache2.2/logs/ssl_scache(512000)”
   SSLSessionCache “shmcb:C:/wamp/logs”
4. ServerAdmin me@localhost.
   ServerAdmin Admin-email
5. ErrorLog “C:/Program Files/Apache Software Foundation/Apache2.2/logs/error.log”
   ErrorLog “C:/wamp/logs/ssl_ ErrorLog.txt”

6. TransferLog “C:/Program Files/Apache Software Foundation/Apache2.2/logs/access.log”
   TransferLog “C:/wamp/logs/ssl_TransferLog.txt”
7. SSLCertificateFile “C:/Program Files/Apache Software Foundation/Apache2.2/conf/server.crt”
   SSLCertificateFile “C:/wamp/OpenSSL/certs/webserver.cert”
8. SSLCertificateKeyFile ”C:/Program Files/Apache Software Foundation/Apache2.2/conf/server.key”
   SSLCertificateKeyFile “C:/wamp/OpenSSL/certs/webserver.key”
9. #SSLCARevocationPath “C:/Program Files/Apache Software Foundation/Apache2.2/conf/ssl.crl”
   SSLCARevocationPath “C:/wamp/OpenSSL/crl”
10. CustomLog “C:/Program Files/Apache Software Foundation/Apache2.2/logs/ssl_request.log” \
    “%t %h %SSL_PROTOCOLx%SSL_CIPHERx \”%r\“ %b”
    CustomLog “C:/wamp/logs/ssl_request.log” \
    “%t %h %SSL_PROTOCOLx %SSL_CIPHERx \”%r\“ %b”

We have to modify the base directory in ’openssl.cnf’ configuration file:

- Open to edit the file “C:/wamp/bin/apache/Apache2.2.11/conf/openssl.cnf”
- Replace the line “dir=./demoCA” with “dir=C:/wamp/OpenSSL”

We now have to modify the http daemon configuration file ’httpd.conf’ as explained:

- Open to edit the file “C:/wamp/bin/apache/Apache2.2.11/conf/httpd.conf”
- Uncomment the following three lines:
  LoadModule rewrite_module modules/mod_rewrite.so
  LoadModule ssl_module modules/modules/mod_ssl.so
  Include conf/extra/httpd-ssl.conf

Finally, test if the configurations made are correct by

1. typing “httpd -t” in command prompt in the directory “C:/wamp/bin/apache/Apache2.2.11/bin”. It should return OK.

2. restarting Apache server and typing “netstat -an | more”. The port 443 should be open.

If the output is contrary to any of the above, then some of the above steps might have been missed.

Configuring Web Server

Copy the contents of the file 'Config.php.dist' and create a new file 'Config.php' with those contents in the folder “C:/wamp/www/trunk/sites/electronics/”. The administrator account is “testinglab.iitm@gmail.com” and its password is “remotelabs”, while a user “electronics” with password “electronics” has been given all privileges on “electronics” database. The following lines in 'Config.php' have to be changed to suit the settings of other packages. The first line should be replaced with the second one in each of the following:

1. $openlabs_dir=“/usr/home/zeta/dav/openlabsweb/trunk”;
   $openlabs_dir=“C:/wamp/www/trunk”;
2. ini_set(“include_path”, $incpath.“:”.$distpath);
   ini_set(”include_path”, $incpath.”;”.$distpath);
3. $site_db_password = “X”;
   $site_db_password = “password of user ”; //electronics in this case
4. $log_dir = $src_dir . “/logs”;
   $log_dir = $src_dir . “/logs”;
   The logs file should be created manually in the folder
   “C:/wamp/www/trunk/sites/electronics”
5. $site_admin = “staff@some.domain”;
   $site_admin = “testinglab.iitm@gmail.com”;
6. $site_mailfrom = “staff@some.domain”;
   $site_mailfrom = “testinglab.iitm@gmail.com”;
7. $site_root = “ ”;
   $site_root = ”/trunk/sites/electronics/public”;
8. $smarty_compile_dir = “/path/to/webcache/electronics”;
   $smarty_compile_dir = “C:/wamp/Smarty/templates_c”;

9. $smarty-¿assign(”site_url”, ”http://dev.openlabs.bth.se”);
   $smarty-¿assign(“site_url”, “http://10.7.9.101/trunk/sites/electronics/public”);
   10.7.9.101 is the ip-address of the machine on which the entire system is installed.
10. $wiki_upload_dir = “/usr/home/zeta/dav/openlabsweb/trunk/sites/electronics/public/wiki_upload”;
    $wiki_upload_dir = “C:/wamp/www/trunk/sites/electronics/public/wiki_upload”;

We need a smtp server to send automated mails regarding account activation and password resetting among other needs. We are currently using the electrical engineering department's smtp server *volt.ee.iitm.ac.in* which can send mails only to email addresses with 'ee.iitm.ac.in' domain. To enable the *web server* to use this smtp server, open *php.ini* file (from WAMPServer system tray icon and PHP option) and modify the following lines under '[mail function]':

- SMTP = volt.ee.iitm.ac.in
- smtp_port = 25
- sendmail_from = testinglab.iitm@gmail.com

Now, we have to install multiviews to enable the *web server* to find scripts with different file extensions. Open to edit the file “C:/wamp/bin/apache/Apache2.2.11/conf/httpd” and modify the appropriate lines by replacing them with the following code:

```
 # If the electronics site is the root of this webserver
DocumentRoot "<source_directory>/trunk/sites/electronics/public"

# If your site is not the root document,

#you can create an alias for the site like
#Alias /electro <source_directory>/sites/electronics/public

<Directory "<source_directory>/trunk/sites/electronics/public">
  # Basic php setup, may not be needed
  AddType application/x-httpd-php .php
```

```
  AddType application/x-httpd-php-source .phps
  DirectoryIndex index.php index.html

  # Don't allow indexing of the contents
  # MultiViews is needed to guess file suffixes for load/save
  Options -Indexes +MultiViews
  AllowOverride None
  Order Allow,Deny
  Allow from all
</Directory>

# To disable access to .svn directories
<DirectoryMatch "/\.svn/">
  Order allow,deny
  Deny from all
</DirectoryMatch>
```

After modifying the 'httpd.conf' file, restart the apache server.

4.4 Debugging

Here, I have mentioned some of the errors encountered during the installation of the above softwares.

- The *equipment server* gives "The session handle is not valid" error when the resource names of the PXI modules do not match the values given in 'EquipmentServer.ini' file. The 'NiFgen Initialize' creates a new IVI instrument driver session by taking in the resource name as one of the inputs. When there is a mismatch in them, this function shows error. It can be rectified by modifying the contents of 'EquipmentServer.ini' to match the values shown in the MAX.
- Use prefixes for values of components (like k, M, etc.) in the *components.list* and maxlists to prevent error messages reporting a mismatch between these two. Such errors citing that maxlists are not subset of the *components.list* are best handled by looking into the *equipment server* log file "eq.log" present in ".../measurement server/logs/". Set Log Level=5

in the EquipmentServer.ini. It records detailed account of the interaction between *equipment server* and *measurement server*.

- A host of #2032 errors can occur if the flash module cannot connect to the *measurement server*. Open the "config.xml" in flash folder and check if the address of the *measurement server* is matching what has been specified in the previous sections. It can also occur due to some missing files in the flash module folder, which can be rectified by copying the required files from the flash source files folder.

- If the *web server* returns the error "Access denied for user 'electronics'@'electronics' (using password: YES)" when we try to open the url

  "https://localhost/trunk/sites/electronics/public/", then there is a problem with the password details of this account in the 'config.php' file in ".../trunk/sites/electronics/" folder. To set an account for the database 'electronics', we enter the following command in SQL commandline in phpmyadmin:

  GRANT ALL PRIVILEGES TO 'electronics' ON 'electronics' IDENTIFIED BY 'electronics';

  This grants all privileges to the user 'electronics' with password 'electronics' on the database 'electronics'. This information has to be updated in the 'config.php' file.

- If you get the error "Deprecated: Assigning the return value of new by reference", it means that the Text_Wiki is an old wiki markup engine and still relies on older version of PHP which used to allow this. To remove this error, in all the files mentioned in the error dialog, remove the '&' after the '=' in the assigning equation.

- When you run the *web server* for the first time, you will get a lot of missing files errors, which you can sort out by modifying the path of the included files. You can use Dev-PHP IDE which is extremely helpful for debugging such include-path errors as it allows us to open a large number of files in tabs simultaneously and explore the code.

- If the *web server* is citing missing files error even when the files are present in the folder, then we might not have installed multiviews module in the server. It allows the *web server* to search for files whose file type is not known. For example, given 'signin' it can search for 'signin.php' or 'signin.cpp' etc.

- Never reactivate admin user account from the web interface as the password cannot be sent to a gmail account and hence we can never know the changed password. In such a case we have to delete the account from phpmyadmin and again add it following the steps mentioned in the installation section.

- If after installing openssl, when we enter 'openssl' in command prompt ,it shows that it cannot find the 'openssl.cnf' file, then most likely the environment variable defining its location is not defined properly and has to be redefined. This can be confirmed by entering

openssl -config C:\wamp\bin\apache\Apache2.2.11\conf\openssl.cnf

in the command line after going to the directory

"C:/wamp/bin/apache/Apache2.2.11/bin". If it works fine then the environment variable has not been declared properly.

- From the web interface, when we try to click on the link "Start experimenting" we get and error stating experiment.php was not found on the *web server*. It is because the the flash client is configured to start only in secure connection (url should have 'https' in it) while the link is taking us to a 'http://' url. It can be solved by opening "occasion.tpl" in the *web server* and changing the 'http' urls in line 5 and 7 to 'https' urls.

- The web interface that we see is actually constructed by smarty templates which take variables from the back-end php scripts. So to change the look of the website we have to change the templates. The general procedure would be to browse through the php files and its functions and then go to the template files which are called by these functions. The scripts supporting wiki pages are meant to be used by all the projects under VISIR which is why they are present in the 'common' folder. 'menubar.tpl' constructs the the menu in the left side of the page, 'contentheader.tpl' constructs the login and language preferences links in the upper side. The contents of the page can be hidden from from type of users like guests or students. The 'wiki pages' link has been made available to students for viewing content by by including the line:

```
<div class="menu-item"><a href="{$wiki_index}?sel=wiki_pages">{i18n

tag="admin.wiki_pages"}</a></div>
```

Now the function wiki_pages has to be accessible to the authentication level of students, so we modify line 10 ofwiki_index.php:

```
function_register("wiki_pages","wiki_pages", AUTH_GUEST);
```

Similarly, any desirable modifications can be made to the appearance of the web interface.

- Initially when we click on a prepared experiment in flash module it gave an error :

```
[IOErrorEventtype="ioError" bubbles=false cancelable=false eventPhase=2

text="Error #2032"]

Failed to load instruments:

Error #2032
```

When we look at the url of any prepared experiment it has "../experiment.php?sel=experiment_immediate&id=1#". Hence, we look for the function experiment_immediate staring from 'experiment.php' and finally arrive at 'client2.tpl' where the code that loads prepared experiment at line 89 was commented. It was uncommented and te line at 96 has been commented to make it work.

CHAPTER 5

OPERATION CYCLE

In this chapter we go through the entire cycle that occurs between having a circuit schematic to running it from a remote computer using the setup that we have installed.

5.1 Circuit Schematic

Draw the circuit schematic and assign labels to various nodes. We shall build a simple inverting amplifier using a UA741 opamp and two equal resistances. The schematic is shown in Figure 5.1.

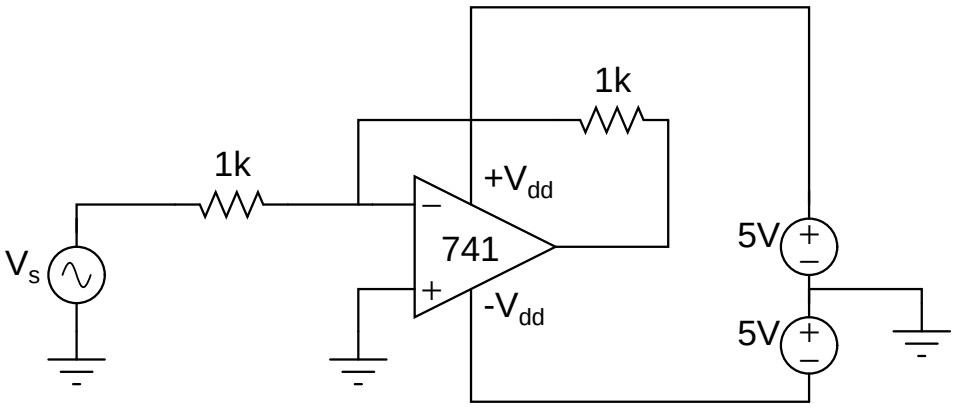

Figure 5.1: Schematic of inverting amplifier

The only constraint while labelling the nodes is that the function generator is connected only to node A internally through a relay on source board. The remaining nodes can be labelled in any order. Every node in the circuit has to be labelled and accounted for. In the case of opamp, even the power supplies have to be assigned nodes.

5.2 Components list and Maxlists

Now we try to fit in the relays in the schematic. Follow the steps below:

1. A resistor is always connected between two nodes through two relays, i.e., a DPST or two SPSTs.

2. Always include a relay between any instrument terminal and a component node.

3. If the DMM has to be used to measure current in a line between two nodes, we have to include a SHORTCUT between those nodes. A SHORTCUT is nothing but a short wire connecting the two nodes through a relay.

4. In the case of an opamp, connect each of the five terminals to a node through a relay.

   The schematic with relays included is shown in Figure 5.2. R_c indicate component board relays and R_s indicate source board relays.

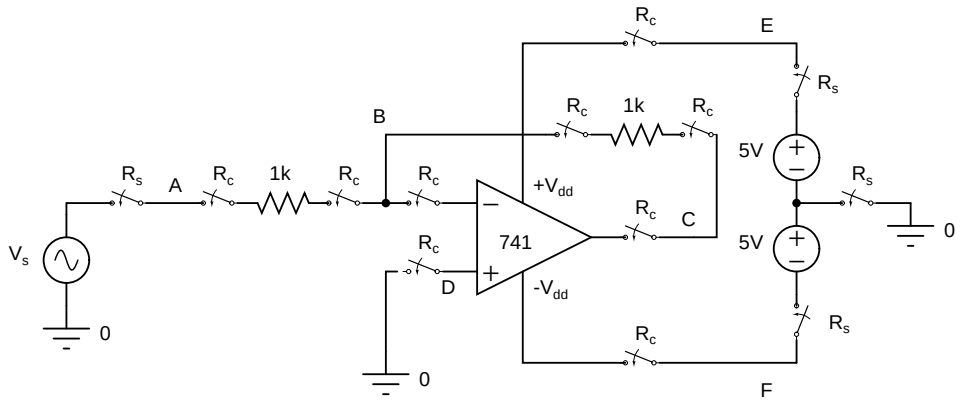

Figure 5.2: Schematic of inverting amplifier with relays

5. The corresponding *components.list* is :

```
VFGENA_24_1    A 0
VDC+6V_24_3:1_13  A    vmax: 4 imax:0.5

VDC+25V_24_4:1_4    E    vmax:15 imax:0.5

VDC-25V_24_5:1_9    F    vmax:-15 imax:0.5
VDCCOM_24_2 0

OP_1_14:1_11:1_10:1_7:1_6    NC1 B D F NC2 C E NC3 UA741

R_1_2    A B    1k
R_1_3    B C    1k
```

```
SHORTCUT_1_12   D 0
```

and the maxlist *Fgen.max* is:

```
VFGENA_24_1   A 0
VDC+6V_1  A   vmax: 4 imax:0.5
VDC+25V_2   E   vmax:15 imax:0.5
VDC-25V_3   F   vmax:-15 imax:0.5
VDCCOM_4   0
OP_1   NC1 B D F NC2 C E NC3 UA741
R_1   A B   1k
R_2   B C   1k
SHORTCUT_1   D 0
```

6. After the maxlist file is created, add it in the *maxlists.conf* file:

```
maxlists/Fgen.max
```

Now, plug in the components and wires on the component boards of switching matrix as described in the *components.list* file. The physical connections between relays and components on the Relay Switching Matrix is shown in Figure 5.3 While making the connections, the matrix should be handled with great care. Do not touch the pins on the boards directly as it might cause ESD. The microprocessors and the relay drivers must not be removed from their sockets. If a relay is to be replaced on a component board, be careful when installing the new relay and install it with pin 1 in the correct position otherwise the internal transient suppression diode will be destroyed and most likely the relay driver as well.

5.3 Web Interface

Now that the hardware and its configuration files are ready, we have to prepare an experiment and add it to a course so that students can perform the experiment. The following steps describe how to prepare an experiment.

1. Log into the openlabs webpage at "https://10.7.9.101/trunk/sites/electronics/public/" using admin account.

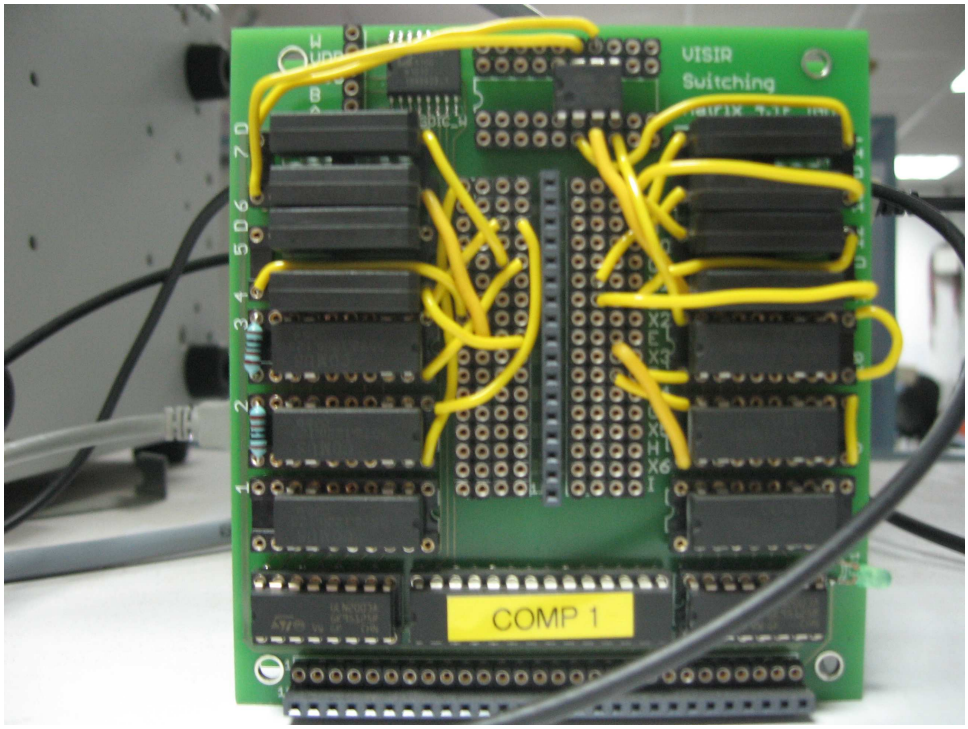

Figure 5.3: Component board of matrix for the above schematic

2. Go to “Admin Courses” > “Add Course” > add a course ’test’ and enter the details.

3. Now, go to “Admin Courses” > test > “View as teacher”. In this page, you can add users to the course and specify their type.

4. Now click on Add prepared experiment > “Start client in teacher mode”. This is the recommended method for testing purposes instead of the standalone file ’loader.html’.

5. You can now add the required components and even make the necessary connections and save the circuit as a .cir file.

6. The above .cir file is then used to create an experiment in the “Add prepared experiment” page that we arrived at previously.

The student can now log in the website and see the prepared experiments in his flash module. It is recommended not to let the students add components themselves by setting the teacher mode in the flash ’config.xml’ file but rather prepare an experiment and include the necessary components.

A word of caution here, while preparing a circuit and testing it for any shorts or such error messages, it is adviced to disconnect the 12V DC power supply to

the switching matrix so that the relays cannot be activated preventing any future damage.

The administrator can also edit the wiki pages to provide information on courses to the students. Some of the screenshots of the web interface are shown below:

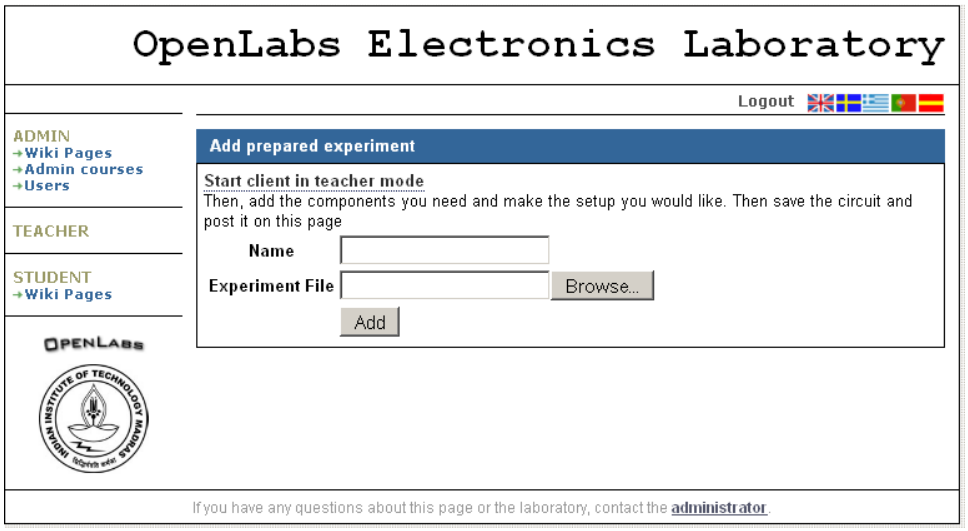

Figure 5.4: Interface for adding a prepared experiment

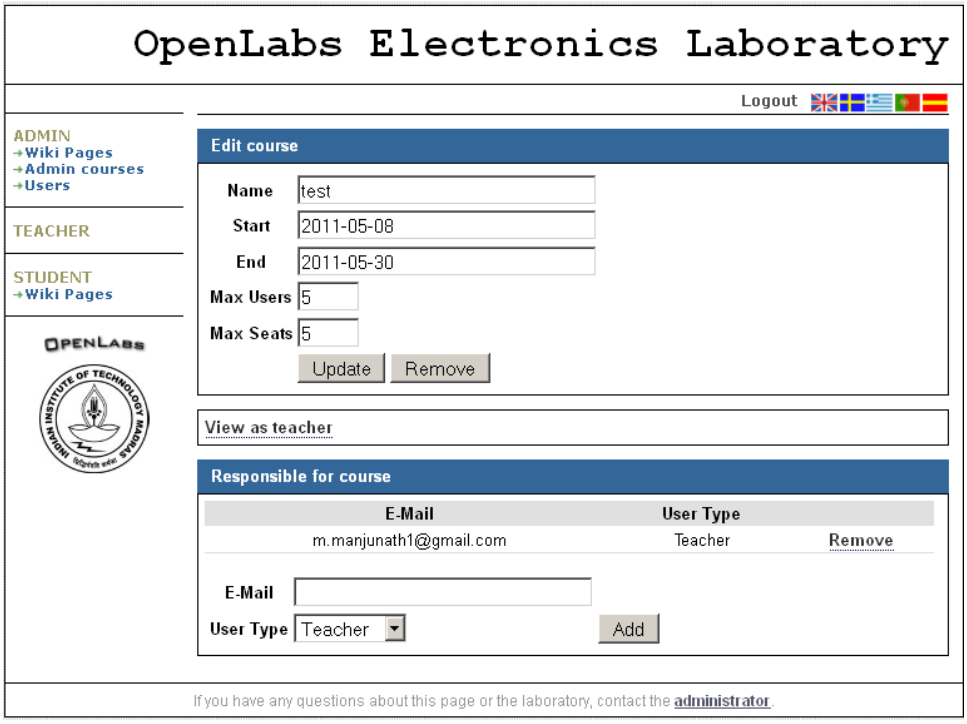

Figure 5.5: Administrator's view of courses

5.4 Results

The student after logging in, goes to the course he is enrolled in and then clicks on "Start Experimenting" to see the flash module. After setting the power supply and function generator signal, the input and output are connected to the two channels of oscilloscope and they are symmetric about x axis as expected as shown in figure 5.7.

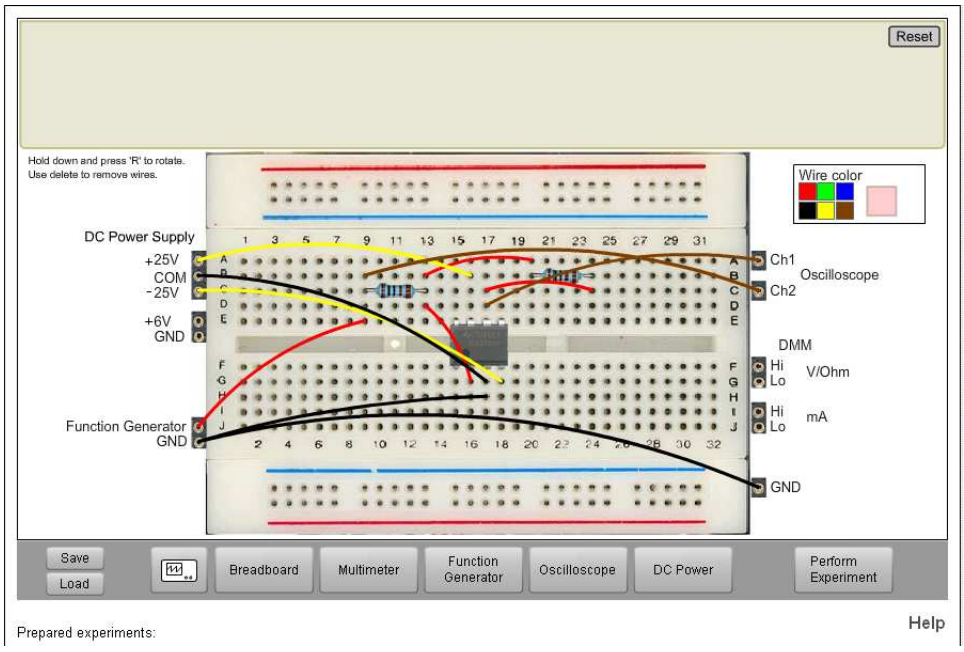

Figure 5.6: Virtual breadboard showing Inverting Amplifier circuit

5.5 Conclusion and Future Work

We have been able to run simple circuits containing resistors and opamps remotely and the results are as good as that of real experiments in the lab. However, there are certain serious limitations to this:

- The number of nodes is strictly limited to 10. This prevents us from running many circuits used in 'analog circuits lab'.
- We cannot make simple mistakes like swapping of connections of opamp terminals as they are prevented by the *measurement server*. We need to laboriously specify even the mistakes that can be attempted by the user in the maxlists and component list.

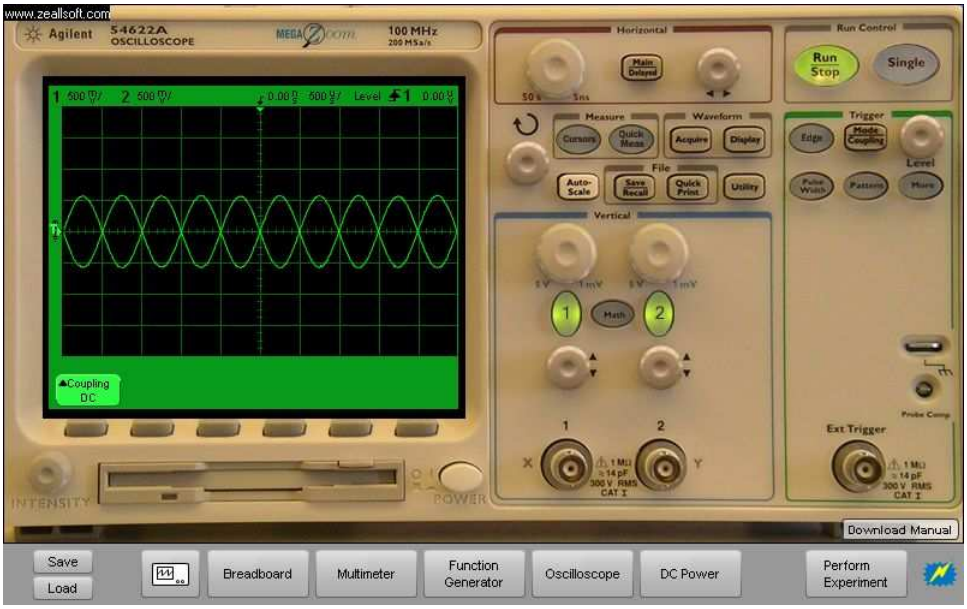

Figure 5.7: Oscilloscope output

- There are a few loopholes in the virtual breadboard such as the shorting of +6V and function generator signal terminals does not generate any error and is allowed.

As part of the future work, we can increase the number of nodes by adding more hardware (central node pins, microcontrollers and relays) and make appropriate changes to the server softwares, which means we have to modify the entire system. The loopholes mentioned can be removed by modifying and compiling the source code of *measurement server*. We can also add a backend simulator to the openlabs interface to enable more functionalities like more number of nodes and allowing wrong connections among others.

APPENDIX A

HARDWARE TESTING

This chapter deals with testing the NI PXI modules and the Relay Switching Matrix using LabVIEW software. After we install all the required NI device drivers, we have front-end soft panels for each instrument present in the National Instruments in 'All Programs' which we can use to control them from the computer. VISIR project also provided a standalone LabVIEW program called "Send circuit to matrix.vi" which can be used to switch on/off the relays present on the boards. We will first describe how to control the Switching Matrix.

A.1 Controlling Switching Matrix

The front panel of "Send circuit to matrix.vi" is shown in the Figure **??**.

As we have seen from the Measurement and Automation Explorer that the resource name of this usb matrix is "USB::0x1043::0x0000::NI-VISA-0::RAW". We either enter this value manually in the "VISA resource name" box, or right click "VISA resource name" > "Select VISA Class" > "I/O Session" > "USB Raw" after which the resource name appears in the drop down menu.

Messages in USB raw format are sent from the PC to the matrix controller through the "Array of board numbers and bit masks" array.The matrix controller then sends this message to the board concerned. A message is 4 bytes in size. Table shows the format of this message.

The first byte is always used to specify the board number. For the remaining bytes, when they are used to specify the relay numbers , the binary representation of the number used indicates the switch states of the relays.

If the binary representation of the byte1 is b7 b6 b5 b4 b3 b2 b1 b0, b0 denotes the state of relay number 1. If b0=1 the relay is switched on and if b0=0 the relay

Table A.1: Board Controller Messages

Board Type	Byte0	Byte1	Byte2	Byte3
Source	Board Number	Relay 1-7	Relay 8-14	Not Used
Instrument	Board Number	Relay 1-7	Relay 8-14	Relay 15-21
Component	Board Number	Relay 1-7	Relay 8-14	Potentiometer Ratio

is switched off and so on. b7 is always zero. In case of byte2, b0 denotes 8th relay, b1 denotes 9th relay and so on and in the case of byte3 b0 denotes 15th relay etc.

If the byte is being used to indicate the potentiometer ratio, then b0 to b6 is potentiometer ratio and b7 is always 1.

A.2 Soft Front Panels

Figures A.1 and A.2 show the soft front panels of DC Power Supply and DMM respectively.

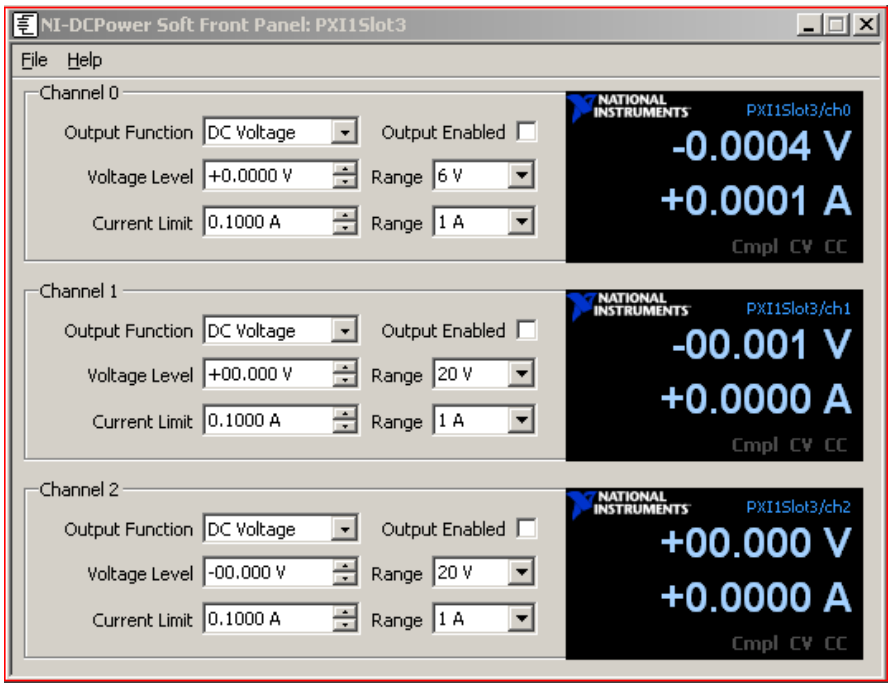

Figure A.1: Soft Front Panel of DC Power Supply

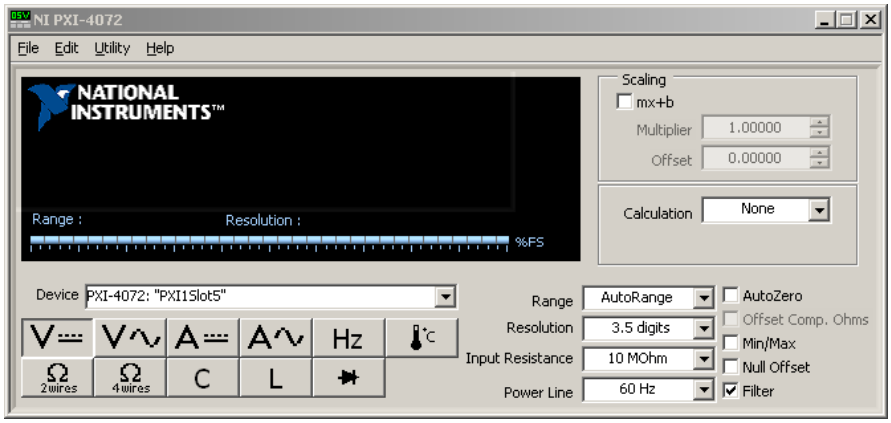

Figure A.2: Soft Front Panel of DMM

A.3 Testing

Here we will try to use the internal relays (on instrument boards) to connect the function generator and oscilloscope. Function generator can connect to only node A and Oscilloscope can connect to any node. So, we send the message as shown in the Figure A.3. We activate relay 1 on source board and relay 19 on Oscilloscope board. The waveform generated by the function generator is shown in oscilloscope as shown in Figure A.4.

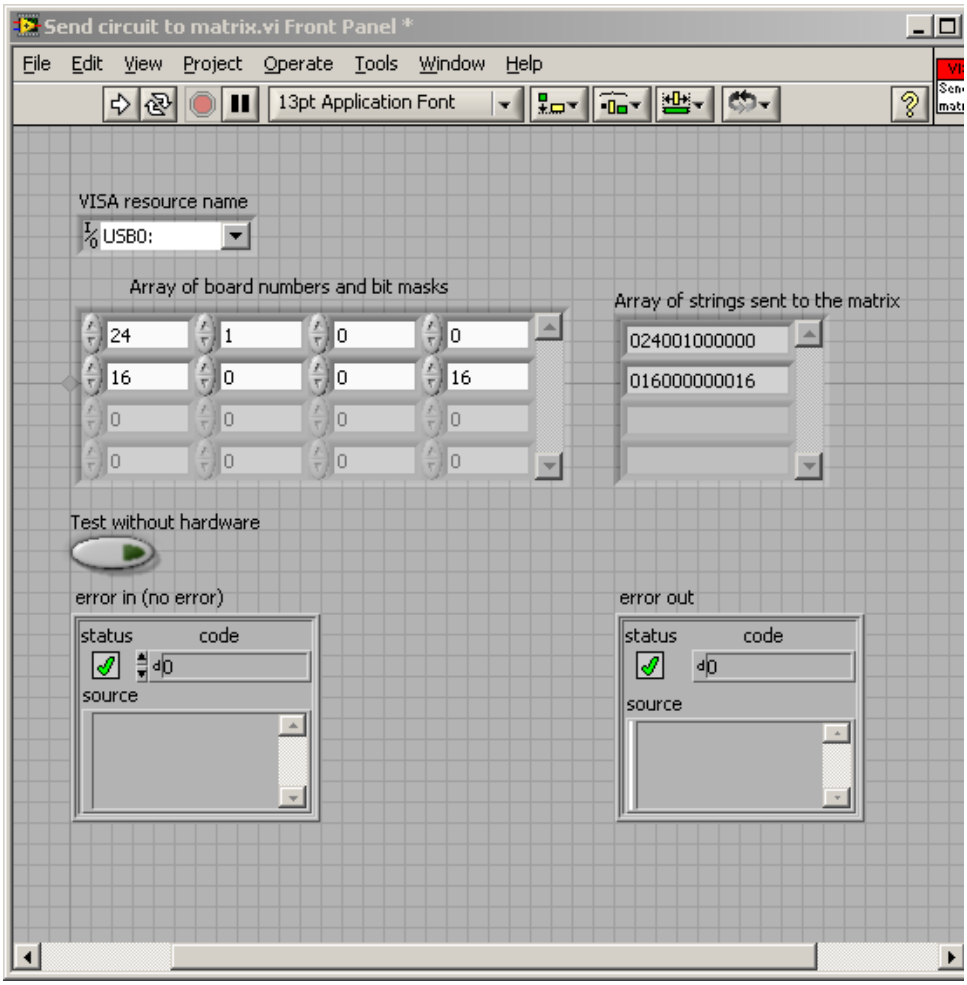

Figure A.3: Front Panel of 'Circuit to matrix.vi'

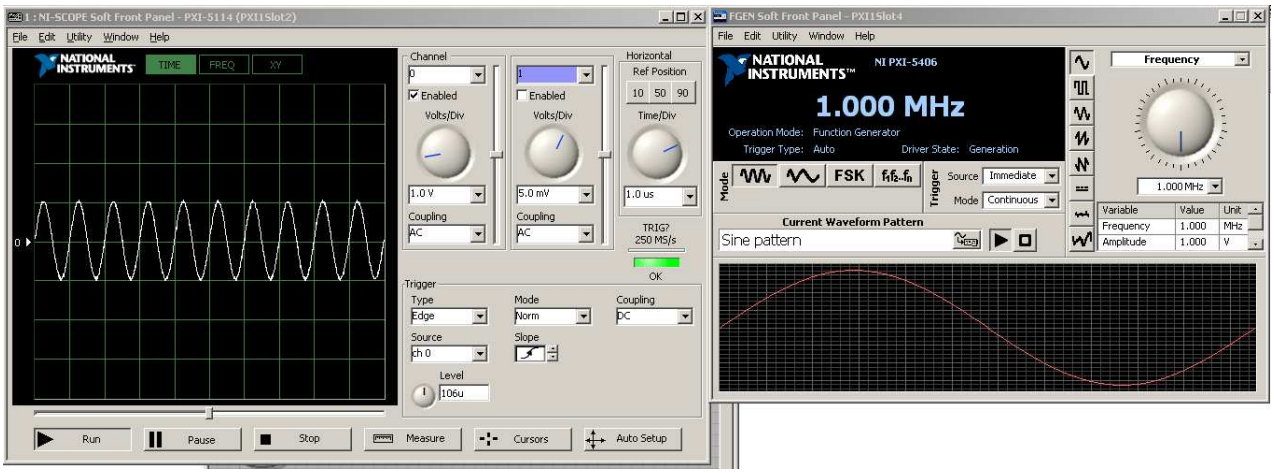

Figure A.4: Soft Front Panel of Function Generator and Oscilloscope

REFERENCES

[1] I. Gustavsson, J. Zackrisson, K. Nilsson, J. G. Zubia, L. Hakansson, I. Claesson, and T. Lagö, "A flexible electronics laboratory with local and remote workbenches in a grid," vol. 4, no. 2, pp. 12–16, 2008.

[2] I. Gustavsson, 2010, teacher manual 6.

[3] ——, 2010, student manual 11.

[4] ——, 2010, VISIR relay switching matrix version 4.1 – user's manual.

[5] M. Tawfik, 2011, Visir Installation and Start-up Guide V.1.

[6] Openlabs home page, http://openlabs.bth.se/.

[7] VISIR project website, http://svn.openlabs.bth.se/trac/.

[8] IVI Foundation, http://www.ivifoundation.org/default.aspx.

[9] National Instruments, http://www.ni.com/.

[10] Wampserver : Apache, MySQL and PHP on Windows, http://www.wampserver.com/en/.

[11] Smarty: PHP template engine, http://www.smarty.net/.

[12] Text_Wiki project webpage, http://pear.php.net/package/Text_Wiki/.

[13] Microsoft Visual C++ 2010 Express, http://www.microsoft.com/express/Downloads/.