OpenLabs Electronics

A Remote Electronics Laboratory

André van Schoubroeck

September 2008 until January 2009, Ronneby (Sweden)

INTERNSHIP REPORT FOR FONTYS HOGESCHOOL ICT

Information about the student:

Name	A.H.L. Van Schoubroeck
Student ID Fontys	2076950
Student ID BTH	anvf08
Course	Engineering and Computing Technology, Full-Time (Dutch: Technische Informatica)
Internship Period	September 2008 until January 2009

Information about the institution:

Name	Blekinge Tekniska Högskola
Department	OpenLabs Electronics Lab
Project Leader	Ingvar Gustavsson
Host College Supervisor	Johan Zackrisson
Home College Supervisor	Dick van Schenk Brill

Information about this report:

Title	OpenLabs Electronics A Remote Electronics Laboratory
Date	10 January 2009

Read and signed by host college supervisor and project leader

Date: Date:

Johan Zackrisson, Ingvar Gustavsson,

Preface

This rapport was written in context of my internship at Blekinge Tekniska Högskola, a University College in Sweden.

During my internship I've been working on various tasks within the OpenLabs project. I have been programming in ActionScript, C and C++. I've been exploring possibilities to enhance the project and realising it. In this report I will write about my work during my internship.

I would like to thank Dick van Schenk Brill (from the Fontys University of Applied Sciences in Eindhoven, the Netherlands) and Ingvar Gustavsson (from Blekinge Tekniska Högskola in Ronneby, Sweden) for giving me the possibility to do my internship at Blekinge Tekniska Högskola.

I would also like to thank my technical supervisor and colleague Johan Zackrisson for giving me the opportunity to work on this project. I also thank my other colleague Kristian Nilsson. It has been a pleasure to work with both of my colleagues.

Index

Summary

The project has been executed at Blekinge Tekniska Högskola, a Swedish University College. Blekinge Tekniska Högskola has developed a project to support distant students with on-line experiments, which is called the OpenLabs project. The OpenLabs project is an umbrella project, and contains different projects, all offering on-line experiments to students in different subject.

This report will focus on the OpenLabs Electronics project, a project that allows students to perform electronic experiments on-line.

The interface the students can use to operate the measurement equipment designed according to real equipment, like a multimeter, power supply, function generator and oscilloscope. Using this interface, students can learn how to operate the real instruments, but this interface is not optimised for being used as a computer interface. Therefore a new interface has been created, designed after a software-based interface designed by National Instruments.

Not only the interface has been improved, the software performing the measurements has been upgraded as well. A new proxy software has been written to enhance the performance. The software that performs the actual experiments has also been improved. The initial software depended on LabView to control the measurement equipment and the circuit the measurement is performed on. LabView is a graphical programming environment. Because of this, it's hard to keep track of changes in the code, making hard to maintain. A LabView license is required to run the server.

This dependency has been removed by integrating the control into the software that performs the experiments. The requirement to have a LabView license has been removed, the code is easier to maintain. Because the entire processing of the measurement request is handled by one single program, the integration results also in a performance improvement.

Samenvatting

Het project heeft plaatsgevonden aan Blekinge Tekniska Högskola, een hogeschool in Zweden. Deze hogeschool heeft een project ontwikkeld dat thuis studerende studenten in staat stelt om practica uit te voeren via het internet. Dit project heet het OpenLabs project. Het OpenLabs project is een overkoepelend project en bestaat uit verschillende projecten, die elk online practica aanbiedt op verschillende vakgebieden.

Dit verslag gaat over een van die projecten, namelijk het OpenLabs Elektronica Project. Met dit project kunnen studenten online elektronische circuits bouwen en doormeten.

Studenten kunnen een voeding, een multi meter, een functie generator en een oscilloscoop bedienen. De interface is ontworpen met bestaande meetinstrumenten als uitgangspunt, zodat studenten kunnen leren hoe deze meetinstrumenten bediend moeten worden. Een fysiek meetinstrument nabootsen op een computerscherm is niet ideaal, want een fysiek meetinstrument is niet ontworpen met bediening via een computerscherm in gedachten. Daarom zijn er nieuwe bedieningspanelen ontworpen, met een bestaande computergebaseerde interface als uitgangspunt, namelijk de Soft Panels van National Instruments.

Niet alleen de bediening van het OpenLabs Elektronica project is verbeterd, ook in de software die de metingen uitvoert zijn verbeteringen doorgevoerd. Er is een nieuw proxy programma ontwikkeld, dat betere prestaties levert dan de oudere variant. Daarnaast is het programma dat de experimenten verwerkt verbeterd. De oude implementatie is afhankelijk van LabView om met de meetinstrumenten te communiceren. LabView is een grafische programmeertaal. Dit heeft tot gevolg dat het lastig is om wijzigingen bij te houden, en daarom is onderhoud aan LabView code niet gemakkelijk. Daarnaast is voor een in LabView geschreven programma, LabView zelf ook vereist, en moet deze dus ook aangeschaft worden om het project te kunnen gebruiken.

Deze afhankelijkheid is verwijderd door de communicatie met de meetapparatuur te integreren in de server programmatuur. Hierdoor is de code makkelijker te onderhouden, en hoeft er geen LabView meer aangeschaft te worden. Deze integratie levert ook een prestatieverbetering op omdat alles nu binnen één programma wordt geregeld.

Sammanfattning

Projekten har utförts på Blekinge Tekniska Högskola. Högskolan har utvecklat flera projekt för att erbjuda experiment till distansstudenter. Dessa tillhör OpenLabs, som är det övergripande projektet. OpenLabs innehåller olika projekt. Varje projekt har online experiment i annat subjekt.

Rapporten beskriver ett av projekten, OpenLabs elektronik labb, där studenter kan bygga elektroniska kretsar och mäta på dessa.

Studenter kan använda mätinstrument som imiterar verkliga instrument. Verkliga instrument är inte alltid lämplig att använda på en dataskärm. Därför har nya instrumentfrontpaneler utvecklats. Instrumentfrontpanelerna efterliknar National Instruments Soft Panels, vilka lämpar sig för dataskärmen.

Förutom frontpanelerna har även interna funktioner förbättrats. Ny proxy programvara har skrivits. Den nya proxy programvaran är snabbare än den gamla. Programvaran som utföra experimenten har också förbättrats. Den gamla programvara som utföra experiment beror på LabView som kommunicerar med mätinstrumenten. LabView är grafisk programmeringsspråk. Detta försvårar underhåll. Underhåll är inte lätt. Den som använder programvara skriver i LabView, måste köpa en LabView licens.

Beroendet av LabView har tagits bort, genom att integrera kommunikation med mätinstrumenten från programvaran som utför experiment. Det gör koden lättare att underhålla och man behöver inte köpa LabView. Eftersom allt integreras i en mjukvara blir den snabbare.

Terms

ActionScript	Programming Language used by Flash.
Base-64	Encoding to encode binary data into text
Daemon	Program without user interface that runs in the background performing a specific task.
Equipment Server	Server that interfaces with the measurement equipment.
Flash	Program to create web applications.
Flash Client	User Interface for OpenLabs Electronics written in Flash
Front Panel	Component of the Flash Client to operate a specific instrument
LabView	Graphical programming language with interfaces to measurement equipment
Measurement Server	Server that receives and verifies measurement requests
OpenLabs	Projects for remote controlled laboratories
Proxy	A program that receives data and retransmits it
Socket	Interface for network programming
Soft Panel	Software Interface to control National Instruments Equipment
Switching Matrix	Stackable boards that build up electronic circuits

Abbreviations

API	Application Programming Interface
BTH	Blekinge Tekniska Högskola
CGI	Common Gateway Interface
DC	Direct Current
GPIB	General Purpose Interface Bus
HTTP	Hypertext Transfer Protocol
IVI	Interchangeable Virtual Instrument
NI	National Instruments
PHP	PHP: Hypertext Preprocessor
PXI	PCI eXtensions for Instrumentation
TCP	Transmission Control Protocol
UNIX	Uniplexed Information and Computing System
USB	Universal Serial Bus
VISIR	Virtual Systems in Reality
XML	eXtensible Markup Language

1. Introduction

Blekinge Tekniska Högskola is a University College in Sweden. Blekinge Tekniska Högskola does not only offer courses at it's campuses, but also distance courses.

Blekinge Tekniska Högskola developed a project to allow distant student to perform real laboratory experiments. This is the OpenLabs project.

This rapport focusses on the OpenLabs Electronics Project and the improvements made to this project. For example the user interface has been upgraded to support different ways of interacting with the software.

Better performance has been realised by exploring and implementing new http proxy software. The new http proxy is not the only performance enhancement made to the project. The software that processes measurement requests has been improved as well. A new software module has been realised to enable the measurement software to communicate with the hardware directly, without the need of yet another program, as was the case in the old implementation.

In Chapter 2 information about the institution will be provided. Chapter 3 will explain the assignment. In the next chapters the different aspect of the assignment will be explained. Chapter 4 will discuss creation of new front panels. Chapter 5 will discuss a CGI based proxy and chapter 6 will discuss the IVI and USB drivers and their integration into the measurement server.

2. The Institution

This chapter will provide information about Blekinge Tekniska Högskola.

2.1 Introduction

Blekinge Tekniska Högskola (BTH) is a Swedish *University College*.
(Swedish: *Högskola*)

Image 2.1
Blekinge län

Its official English name is “Blekinge Institute of Technology”
It is located in Blekinge län, a province in the South-East of Sweden.
BTH has three campuses: in Karlskrona *Campus Gräsvik*,
in Ronneby *SoftCenter* and in Karlshamn *Campus Karlshamn*.
BTH employs five hundred forty five (545) people.

2.2 History

BTH was founded in 1989. It used to have a different name when it was founded: *Högskolan I Karlskrona-Ronneby (HK/R).*
In 1999 the Institute was granted the right to run Ph.D. programmes in Engineering. In that year the institute also merged with The Baltic International School of Health. In 2000 the institute opened a third campus, Campus Karlshamn and got renamed to *Blekinge Tekniska Högskola.*
In 2001 BTH got officially validated to award the Degree of Master of Science in Engineering, and in 2002 the general right to award Master degrees.

Image 2.2
BTH logo

2.3 Education at BTH

Blekinge Tekniska Högskola offers about forty (40) full-time study programmes and about four hundred seventy (470) single subject courses. Blekinge Tekniska Högskola offers sixteen (16) Master Degree Study Programmes.

At Blekinge Tekniska Högskola there are approximately seven thousand three hunded (7300) registered students. One thousand (1000) of those students are international students. About one thousand six hundred (1600) students take their courses via Internet based teaching.

The courses Blekinge Tekniska Högskola offers cover different areas.
Three percent (3%) of the students at BTH study *medicines*.
Four percent (4%) if the students study *human science*.
Eight percent (8%) of the students study *physics*.
Eleven percent (11%) of the students study *health care*.
Fifteen percent (15%) of the students study *economics*.
Fifty-nine percent (59%) of the students study the area of *telecommunication*.

2.4 Research at BTH

BTH is authorised to do research in engineering and award post-graduate degrees in that area. Research account about one-third of the institutes activities. Research programmes are carried out in co-operation with other universities.

There are fourteen (14) doctoral degree study programmes at BTH, one hundred and twelve (112) doctoral students registered at BTH and twenty six (26) professors working at BTH.

2.5 OpenLabs

OpenLabs is an umbrella project that covers four different projects that share a common goal: Providing resources for distant students.

Within the OpenLabs Project there are four projects: *Antenna theory*, *Electronics*, *Security* and *Vibration Analysis*. These projects offer students to perform experiments in these areas over the internet.

To understand a theory, it's important to apply the theory in practice. Although there are many software programs that can simulate experiments, their results can sometimes be inadequate because they only simulate a mathematical model.

Running a real laboratory, for students to perform experiments in, is expensive, and the resources are limited. The number of students is increasing, and it will be costly to expand the laboratories to meet the increasing number of students.

A remote laboratory can be used to offer experiments to distant students. It can also be used by students on the campus to prepare for a real lab session, so students can use the limited time they have in the real laboratories more efficient.

Opposed to a normal laboratory, the equipment in a remote laboratory can be used my multiple people at the same time. Therefore a remote laboratory is more efficient then a normal laboratory.

2.6 Virtual Systems in Reality (VISIR)

The Electronics Lab started in 1999 at Blekinge Tekniska Högskola as a supplement to local laboratories, to make experiments more accessible for students. The OpenLabs project initiated a big interest worldwide.

In 2006 the Virtual Systems in Reality (VISIR) Initiative was created, a co-operation between Blekinge Tekniska Högskola, National Instruments and Axiom EduTech. Together they work on creating remote laboratories, and release open source software to run them.

Many universities are interested in the OpenLabs project. Among them are:
Carinthia Universitiy of Applied Sciences in Austria,
Gunadarma University in Indonesia,
ISEP in Portugal
Princess Sumaya University for Technology in Jordan,
UNINOVA in Portugal
University of Genoa in Italy,
University Transilvania of Brasov in Romania.

FH Campus Wien in Austria and *University of Deusto* in Spain are already using the OpenLabs Electronics project.

2.7 OpenLabs Electronics Lab

The OpenLabs Electronics Lab is the project within OpenLabs that offers distance students to run experiments on electronic circuits through a web interface. The assignment discussed in this rapport was executed in context of the OpenLabs Electronics Lab.

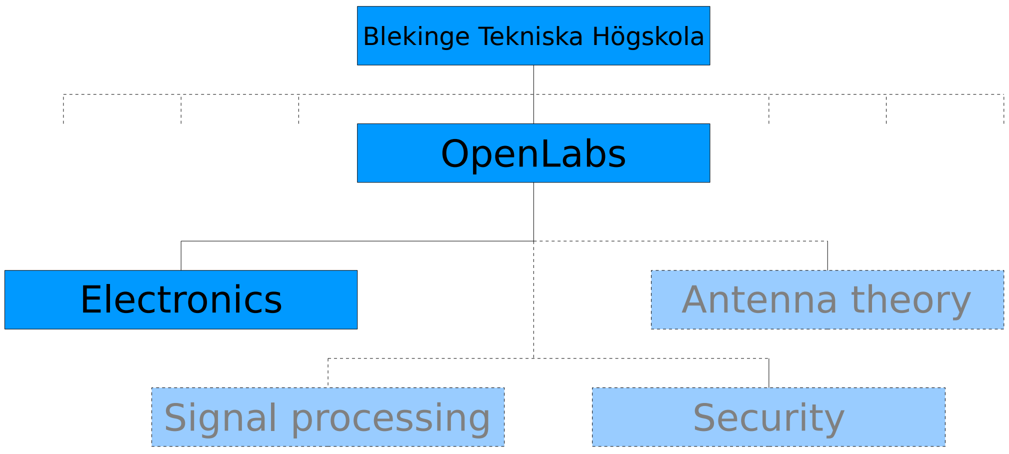


Image 2.7: Organogram

3. The assignment

This chapter will explain the initial situation, the project goals and the assignment.

3.1 Initial situation

The OpenLabs electronics laboratory is currently running and serving many students with measurements.

Students can compose a circuit on a breadboard and do measurements on it. Measurements can be performed using a digital multimeter, a DC power supply, a function generator and an oscilloscope.

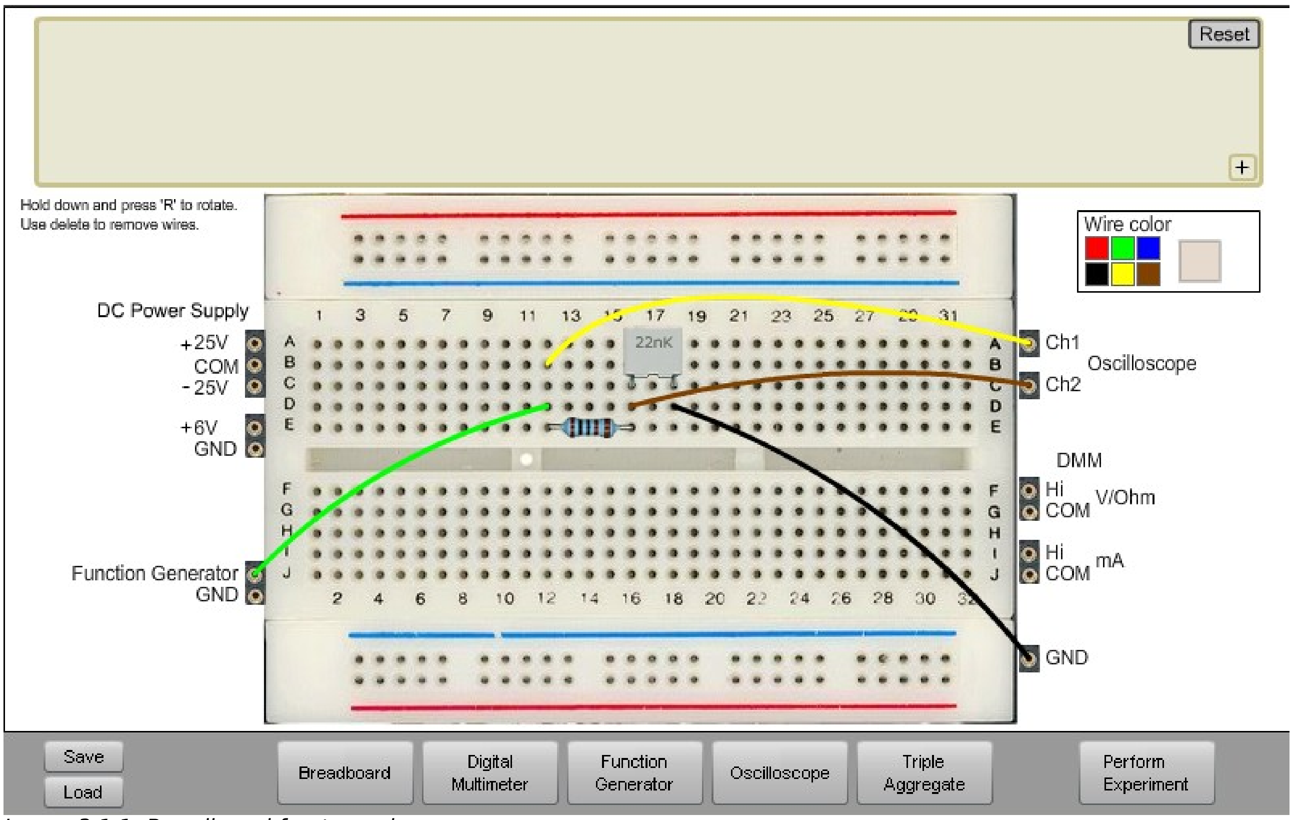


Image 3.1.1: Breadboard front panel

The OpenLabs Electronics Laboratory is operated through a web interface, displaying an image of measurement equipment students work with in real laboratories as well. This way the students can operate equipment they are familiar with. The instruments used in the laboratories are a Fluke 23 multimeter, an Hewlett Packard 33120A function generator, an Hewlett Packard 54622A oscilloscope, and an Agilent E3631A power supply.

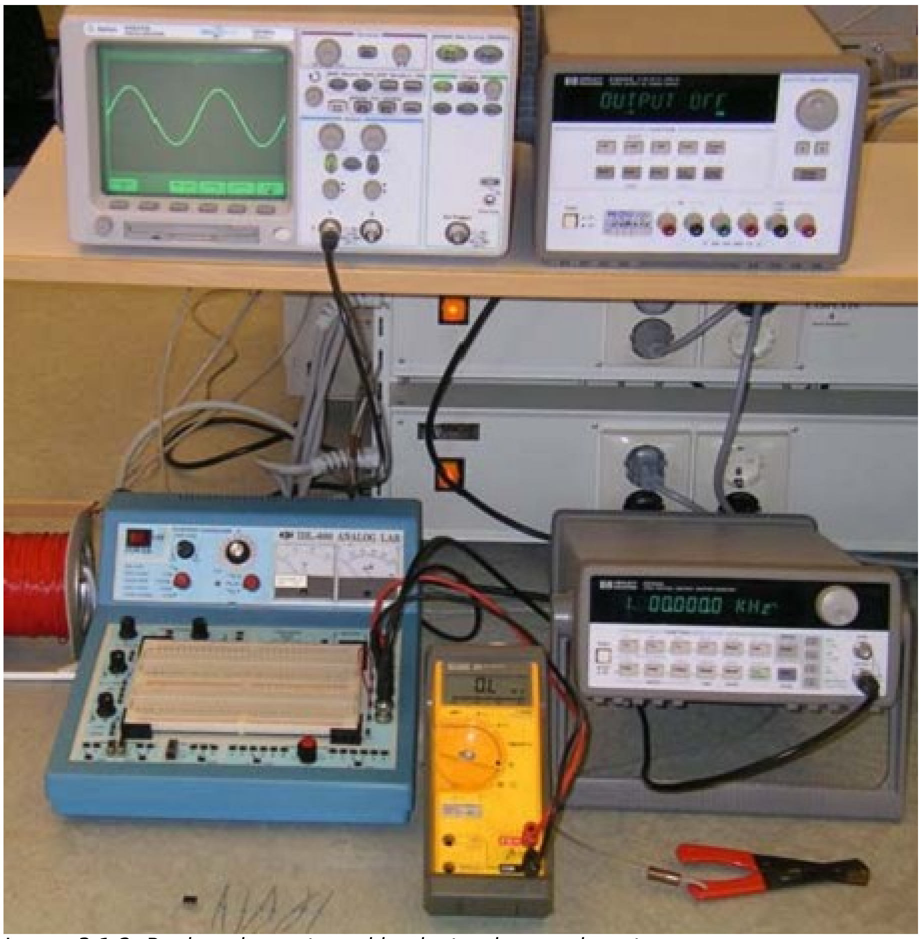


Image 3.1.2: Real equipment used in electronic experiments

The circuits students compose and measure on will be built and connected to the measurement equipment.

Currently the system doing so consists of three components, a web interface that allows students to compose circuits and set up measurement equipment. This component is called the Flash Client. The Flash Client consist of Front Panels, each controlling a different measurement equipment or the circuit. The Flash Client will send the information about the circuit and equipment set-up to the Measurement Server using an XML-based protocol. The Measurement Server verifies the circuit and sends it to the the Equipment Server using a serialised string based protocol.

The Equipment Server will then configure the equipment and receive the measurement results from the equipment. After that the result is returned to the Measurement Server, which will send the result to the Front Panel, and the result will be visible on the Flash Client.

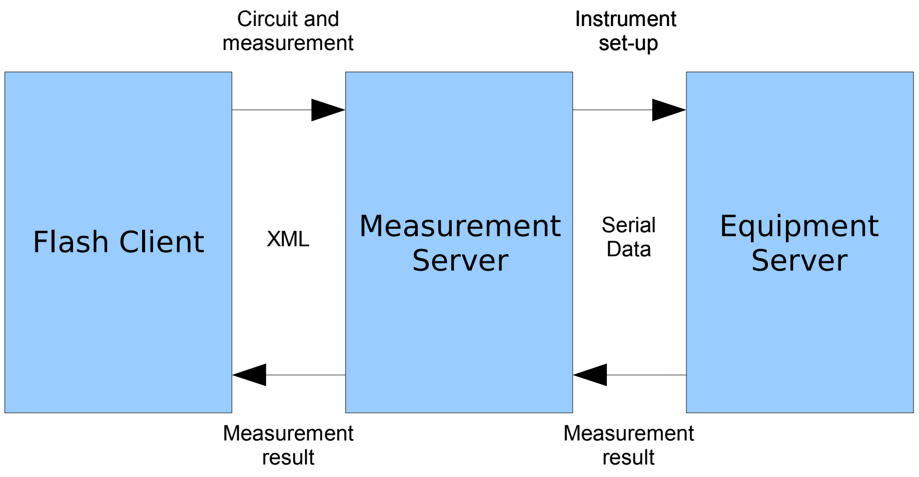


Image 3.1.4: Overview of components of OpenLabs Electronics

The Flash Client is written in ActionScript 3.0 (AS3). This is the coding language used by Adobe Flash.
The Flash program will run within a web browser, and is embedded in a website. The Measurement Server is written in C++. The server runs on a Microsoft Windows Operating System.
The Equipment Server is written in LabView. It will run within the LabView environment.

Some students desire to communicate using HTTP, because their firewall might block the traffic otherwise. For this reason it's possible to connect through a proxy. The current HTTP proxy is implemented as a PHP script on the web-server.

In order to do measurements on the circuit the students compose, it will have to be built. Building the circuits is done by a *USB controlled switching matrix.* This is a matrix consisting of several cards. Each card contains a number of relays. There are two major types of cards. Component cards and Equipment cards. Depending on the card type, the card contains electronic components sush as resistors, capacitors and coils, or connections for measurement equipment.

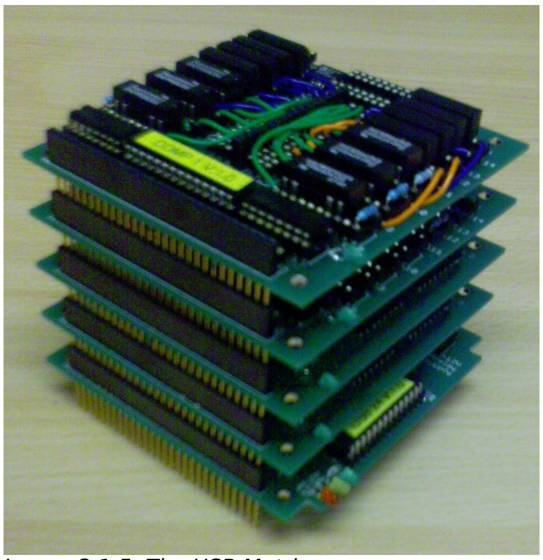


Image 3.1.5: The USB Matrix

The measurements in the remote laboratory are performed using equipment that differs from what the students would use a real laboratory. The used hardware to perform the measurements are *National Instruments* (NI) devices, positioned in a *PXI box* connected to the measurement computer. The used multimeter is an NI-4072, the function generator is an NI-5402, the oscilloscope is an NI-5112 and the DC power supply is an NI-4110

Image 3.1.6: PXI Box

3.2 Objectives

There are several objectives in the OpenLabs project. Creating new Front Panels, creating a new proxy implementation and integrating the equipment server into the measurement server.

3.2.1 Front Panels

One of the objectives is to have multiple front panels, so the students have a choice which front end they wish to use. This way the students have the freedom of choosing how to operate the emulated measurement equipment they shall be using in real laboratories.

The goal is to create new front panels that behave similar to the Soft Panels provided by National Instruments. Soft Panels are software interfaces to measurement equipment. National Instruments is the supplier of the used measurement equipment, and also supports the project. Therefore front panels with similar interface to the Soft Panels are desired, to promote National Instruments.

3.2.2 New proxy implementation

Another objective is to create a new proxy implementation. The current proxy implementation is a simple PHP script. PHP is an interpreted language, therefore it's not as fast as binary code.

Therefore a new proxy written in C should be created. Code written in C compiles to binary code, and so it should improve performance.

3.2.3 Integration of equipment and measurement servers

Another goal to the project is to research and create alternative ways to control the measurement equipment and the USB Matrix, without the need for a LabView server. Since LabView is a graphical programming environment, it is hard to maintain the code. As opposed to text based programming, it is not easy to compare different revisions of the software, and upgrading the code could take up a long time. This problem could be solved by writing the code in a text based programming language.

Another reason to research alternative ways to control the measurement equipment and the USB matrix is a possible speed improvement. In the initial situation, the measurement and equipment server are separate servers communicating using a TCP connection. Because of this the performance is reduced. LabView code does not compile into a binary program, and therefore is not as fast as binary code.

Integrating the code that controls the measurement equipment and the USB matrix into the measurement server may result in a performance improvement.

Next to the performance improvement and the benefits of being able to control the source in a more flexible way, an integrated server will also reduce the costs for running the system, since the dependency on LabView will be removed, and so a LabView license will no longer be required.

3.3 Assignment

From the objectives follow the assignments as creating new front panels that look and behave like the National Instruments Soft Panels, creating a new proxy implementation and researching for alternative ways to control the measurement equipment and USB matrix.

3.4 Development Method

An Agile Software Development Method is used during this project. This method of software development requires frequent face-to-face communication with the other team members. At least once a week, but often more frequent, the progress will be discussed with the other team members.

During Agile software development is a flexible development method and changed to the product can be applied without much trouble. Progress is discussed frequently and team members can adjust to changed requirements.

Agile Software Development only work for people with a certain attitude to the work. People who work using an Agile development method need to the urge to achieve their goals, and the creativity to come up with a good solution for the problem at hand. Agile software development will not work for people who lack this attitude.

Because Agile Software Development needs frequent face-to-face communication with the other members it can only work for small teams. This can also be a disadvantage of this method. If one team member gets ill for a long time it can drastically slow down the project.

4. Front Panels

The first objectivewas to create new front panels that look and behave like the soft panels supplied by National Instruments. In this chapter the creation of those front panels will be elaborated on.

4.1 Flash CS3

The front panels in the OpenLabs Electronic project are created using Flash CS3. To create a Front Panel using flash, a Flash File containing widgets and code written in ActionScript 3.0 is needed. ActionScript 3.0 is an ECMA-derived scripting language. The ActionScript language is close to Java.

4.2 Soft Panels

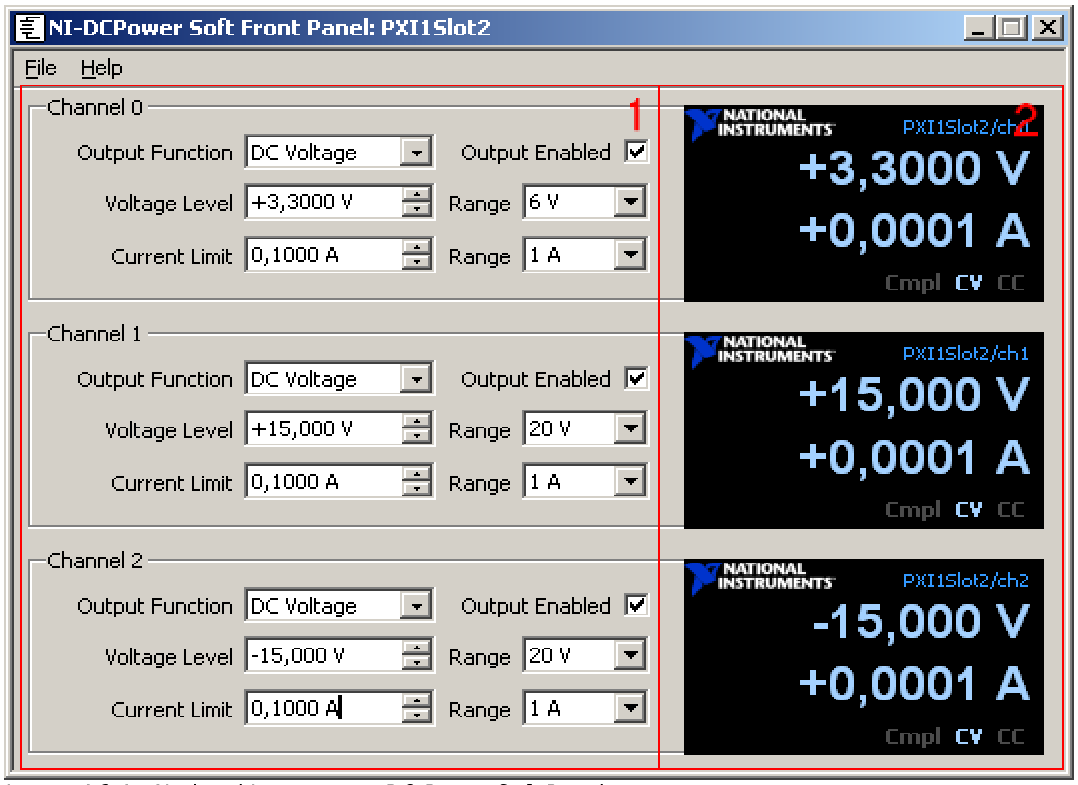
The Front Panels were designed after the Soft Panels from National Instruments, so the Soft Panels had to be analysed first. Soft Panels are software interfaces to control the measurement equipment. The Soft Panels from National Instruments come with the drivers for their measurement equipment, and communicate directly with the hardware. The Soft Panels also support an emulation mode, and can be used without hardware attached to the system. However, the emulation mode has no option to alter the signals emulated, so it's not suitable as a replacement for analysing it's behaviour in reality.

4.3 NI DC-Power

The NI DC-Power Soft Panel is a Soft Panel for a DC Power Supply. It is shown in image 4.3.1. It consists of three output channels, which voltage and current can be set independently.

The Soft Panel from National Instruments supports the following configuration of channels

Channel	Voltage Range	Current Range
Channel 0	0 till +6 Volts	0 till 1 Ampere
Channel 1	0 till +20 Volts	0 till 1 Ampere
Channel 2	0 till -20 Volts	0 till 1 Ampere



Channel 0
Output Function: DC Voltage — Output Enabled: ✓
Voltage Level: +3,3000 V — Range: 6 V
Current Limit: 0,1000 A — Range: 1 A
PXI1Slot2/ch0: +3,3000 V, +0,0001 A, Cmpl CV CC

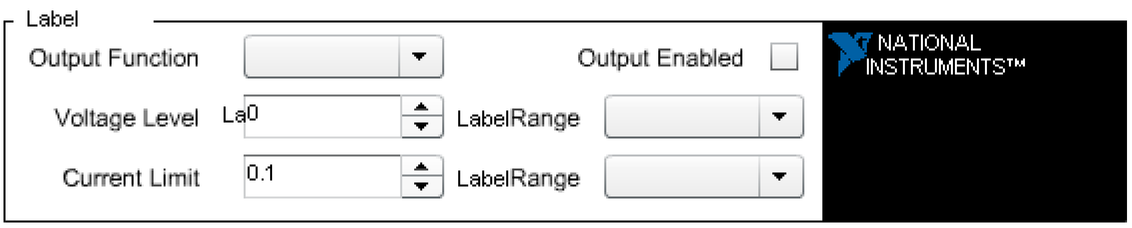
Channel 1
Output Function: DC Voltage — Output Enabled: ✓
Voltage Level: +15,000 V — Range: 20 V
Current Limit: 0,1000 A — Range: 1 A
PXI1Slot2/ch1: +15,000 V, +0,0001 A, Cmpl CV CC

Channel 2
Output Function: DC Voltage — Output Enabled: ✓
Voltage Level: -15,000 V — Range: 20 V
Current Limit: 0,1000 A — Range: 1 A
PXI1Slot2/ch2: -15,000 V, +0,0001 A, Cmpl CV CC

Image 4.3.1 : National Instruments DC-Power Soft Panel

At the left (1) the channel can be configured and at the right (2) the current output will be displayed.

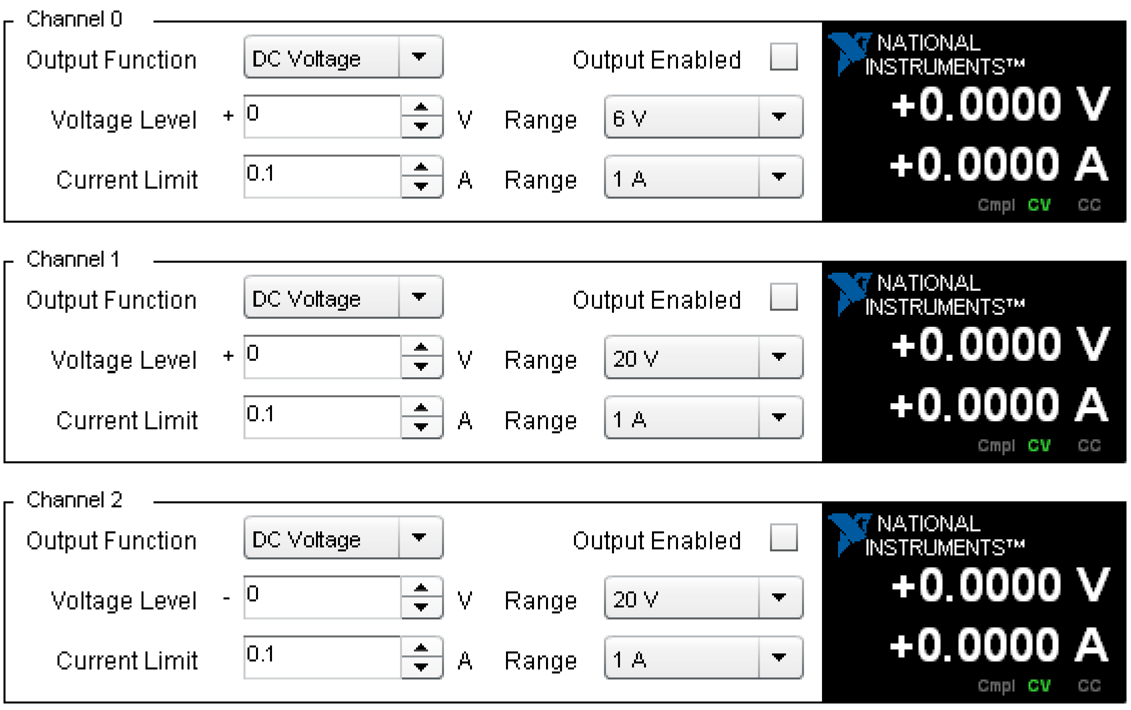
Since all thee channels are the same, except for the allowed ranges, it is possible to create one channel and re-use it. This way the same work doesn't need to be done three times, and if a change is required, it only needs to be done once.



Label
Output Function | Output Enabled
Voltage Level La0 | LabelRange
Current Limit 0.1 | LabelRange
NATIONAL INSTRUMENTS™

Image 4.3.2 : Flash Front Panel, one DC-Power Channel

A single channel without values is designed. When the Flash Clients runs, it will display the channel three times with the values filled in.



Channel 0
Output Function DC Voltage | Output Enabled
Voltage Level + 0 V | Range 6 V
Current Limit 0.1 A | Range 1 A
NATIONAL INSTRUMENTS™
+0.0000 V
+0.0000 A
Cmpl CV CC

Channel 1
Output Function DC Voltage | Output Enabled
Voltage Level + 0 V | Range 20 V
Current Limit 0.1 A | Range 1 A
NATIONAL INSTRUMENTS™
+0.0000 V
+0.0000 A
Cmpl CV CC

Channel 2
Output Function DC Voltage | Output Enabled
Voltage Level - 0 V | Range 20 V
Current Limit 0.1 A | Range 1 A
NATIONAL INSTRUMENTS™
+0.0000 V
+0.0000 A
Cmpl CV CC

Image 4.3.3 : Flash DC-Power Front Panel

The resulting front panel looks similar to the original National Instruments Soft Panel, but it still differs due the use of different widgets. The Front Panel uses the widgets native to Flash.

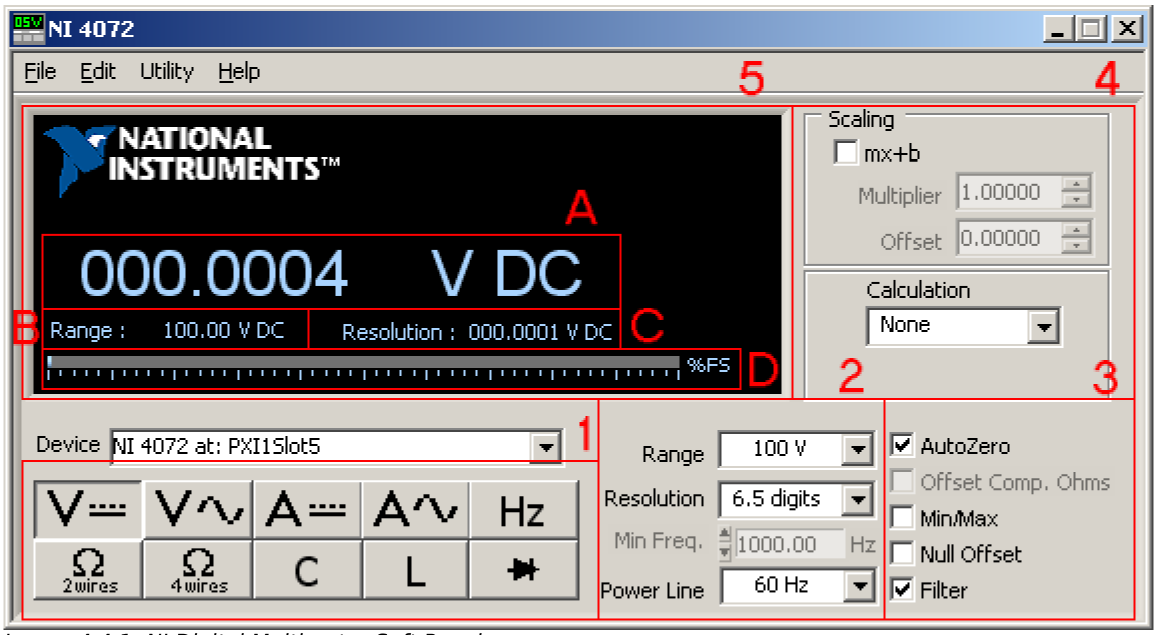
Another thing that's different is the behaviour when values are changed. On the original Soft Panel, the output values are immediately adjusted when the user changes the values in the interface. This behaviour requires a direct connection to the hardware. Since the Flash Client is designed to run on a remote system, it cannot behave like this. In stead, the output values will only be updated upon a new measurement.

4.4 NI Digital Multimeter

The NI Digital Multimeter Soft Panel is a Soft Panel for a Digital Multimeter. It is shown in image 4.4.1 and will be explained below.

At the bottom (1) left there are buttons to select which function the user desires to measure. Possible measurements are DC and AC voltage, DC and AC current, Frequency, 2-wire and 4 wire resistance, capacitance, inductance and diode.

At the right of the measurement buttons (2), the user can select the desired measurement range and resolution. At the rightmost (3) there are options like AutoZero, Min/Max and Null Offset.


NI 4072

File Edit Utility Help

Scaling

mx+b

Multiplier 1.00000

Offset 0.00000

Calculation None

000.0004 V DC

Range : 100.00 V DC

Resolution : 000.0001 V DC

Device NI 4072 at: PXI1Slot5

Range 100 V

Resolution 6.5 digits

Min Freq. 1000.00 Hz

Power Line 60 Hz

[✓] AutoZero

[ ] Offset Comp. Ohms

[ ] Min/Max

[ ] Null Offset

[✓] Filter

Image 4.4.1: NI Digital Multimeter Soft Panel

At the upper right there are scaling options.(4) The user can select a multiplier and an offset that will be applied to the measured value. Below the scaling options there is a calculation option. This option can be used to display the values in dB and dBm.

At the top left (5), the measurement result will be displayed (A), along with the current scaling (B) and resolution(C) information and an indication of the measurement value compared to the selected measurement range (D).

The Flash Front Panel differs in a few points from the original National Instruments Soft Panel. Some options have been disabled due the configuration of the system. For example the “Power Line” and “Filter” options, to select a compensation for the power line, are not available, because the frequency of the power line the measurement equipment is connected to will never change, so it's unnecessary to change this value. The “Offset Compensation” option, to compensate the cables used during the measurement, has also been disabled because it's not possible to apply this option in combination with the matrix, since it needs to be performed on the cables which cannot be disconnected from the matrix in a running system. The “Minimal Frequency” option has been disabled as well, because applying high values to this option can drastically slow down the measurement.

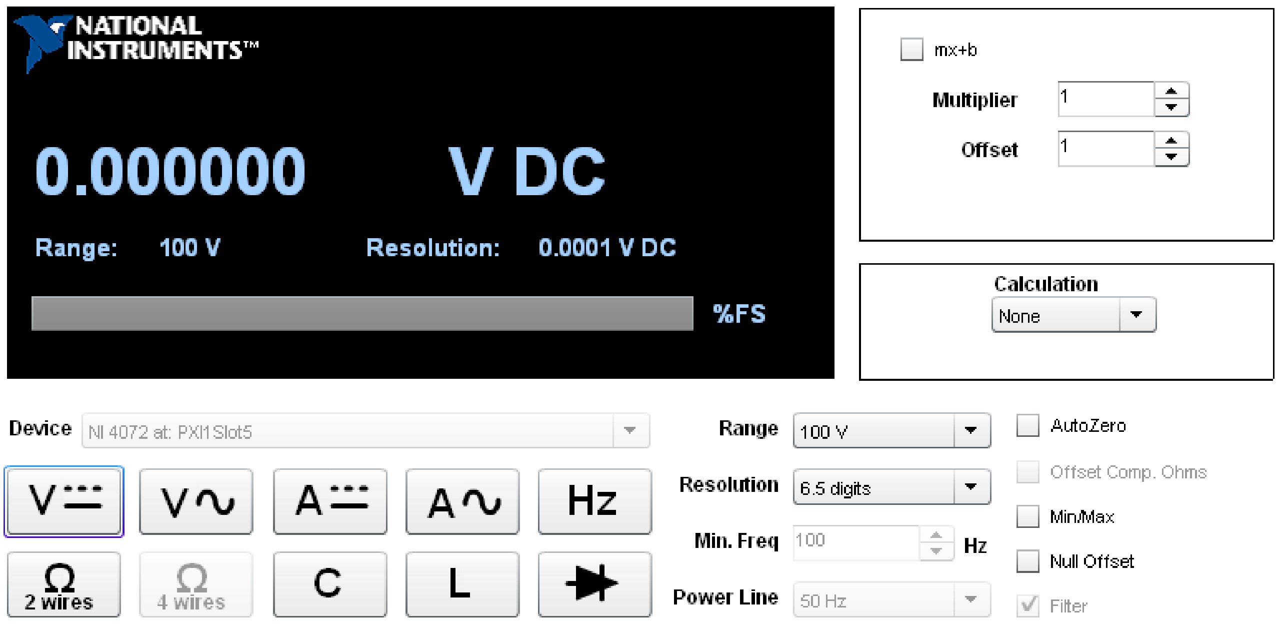
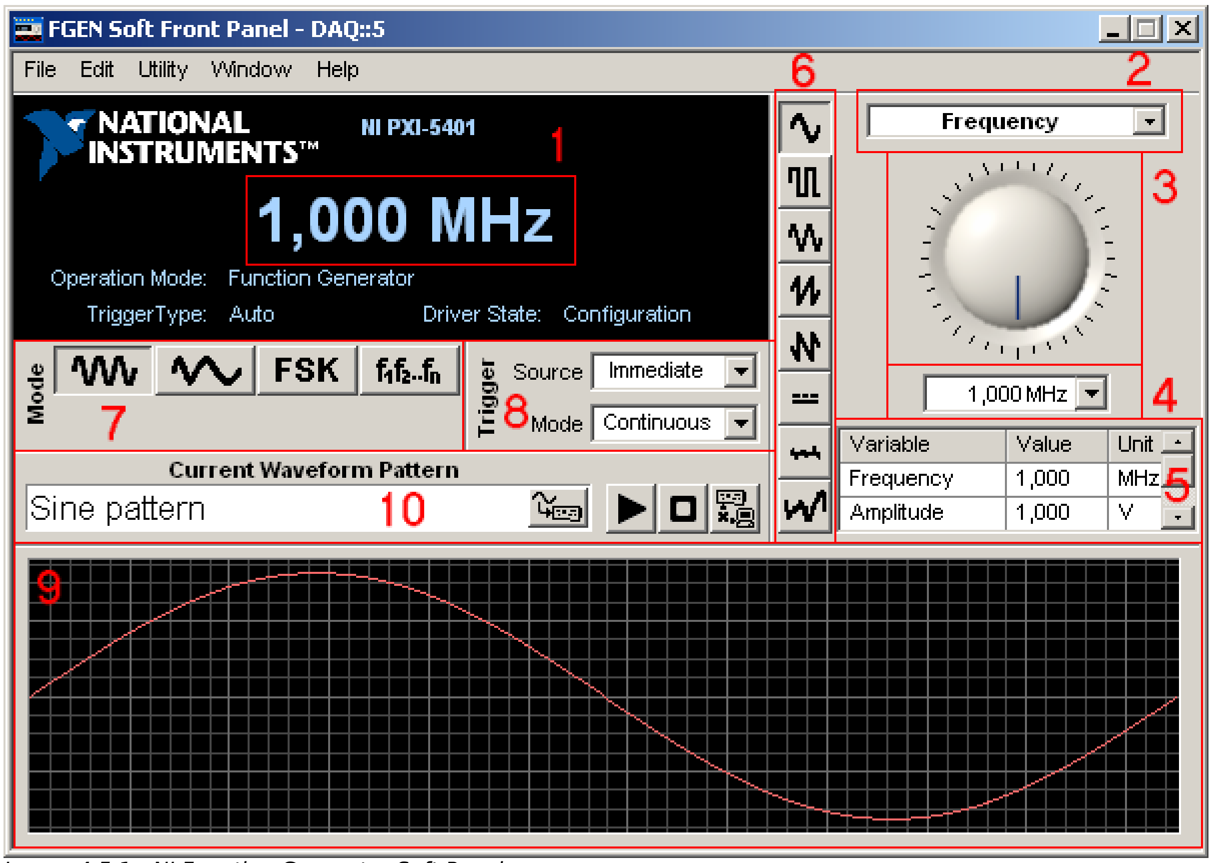


Image 4.4.2: Flash Digital Multimeter Front Panel

4.5 NI Function Generator

The NI Function Generator Soft Panel is a Soft Panel to control a Function Generator. It is shown in image 4.5.1 and will be explained below.

In the black area of the window (1), the current output setting is displayed. This can either be frequency, amplitude, DC-offset or start phase, depending on the option chosen from a drop down list-box at the top right of the window(2). Below this drop- down list-box there is a wheel (3) the user can turn until the desired value is reached. The value will be displayed in the black area and in a box below the wheel(4).


FGEN Soft Front Panel - DAQ::5

NI PXI-5401

1,000 MHz

Operation Mode: Function Generator

TriggerType: Auto

Driver State: Configuration

Trigger Source: Immediate

Trigger Mode: Continuous

Current Waveform Pattern: Sine pattern

Frequency

1,000 MHz

Variable	Value	Unit
Frequency	1,000	MHz
Amplitude	1,000	V

Image 4.5.1 : NI Function Generator Soft Panel

The desired value can also be entered directly in the box (4) below the wheel.

Below the box is a table (5) where all settings are displayed, regardless which setting the user is currently editing.

Between the black area and the wheel there are buttons (6)to select the output function. Possible output functions are sine wave, square wave, triangle, rising ramp, falling ramp, DC voltage level, noise and a user-defined waveform.

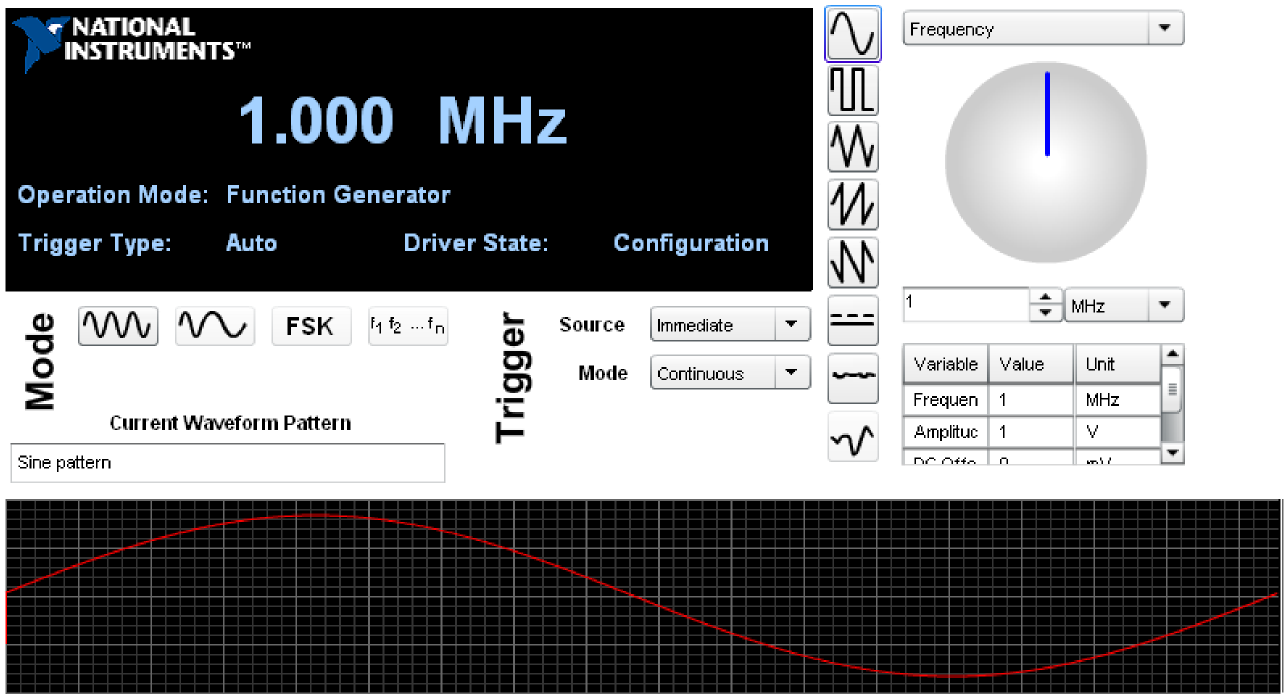
Below the black area displaying the current selected value there are mode buttons(7) and trigger boxes(8). The mode buttons allow the user to select standard function output, frequency sweep of standard functions, frequency shift keying using standard functions and frequency list generation of standard functions. The trigger boxes allow the user to select a desired trigger.

At the bottom of the Soft Panel a graph (9) of the selected waveform is displayed.This graph will not change when the user changed frequency, amplitude, DC offset or phase. It will only change when the user selects a different waveform. Above the graph the name of the current selected waveform is displayed.(10)

There are a few differences between the original Soft Panel and the Flash Front Panel. In Flash there is no widget for a wheel, so a custom wheel had to be implemented. The wheel looks at the position of the mouse pointer compared to the centre of the wheel and determinates the angle that way. It also counts how many turns the wheel has made, and together with the angle, this results in the value the wheel is set to.

Another difference between the original Soft Panel and the Flash Front Panel is the box below the wheel. The original Soft Panel has a box where the user can enter values integrated with a drop-down menu. There is no such thing available in Flash, therefore two standard Flash components have been used to create the same functionality.

The Flash Front Panel only supports standard waveform generation. The other options the Soft Panel supports are left out in the Front Panel since this functionality is currently not supported by the OpenLabs Electronics project.



1.000 MHz

Operation Mode: Function Generator

Trigger Type: Auto

Driver State: Configuration

Current Waveform Pattern: Sine pattern

Source: Immediate

Mode: Continuous

Frequency

1 MHz

Variable	Value	Unit
Frequen	1	MHz
Amplituc	1	V
DC Offs	0	mV

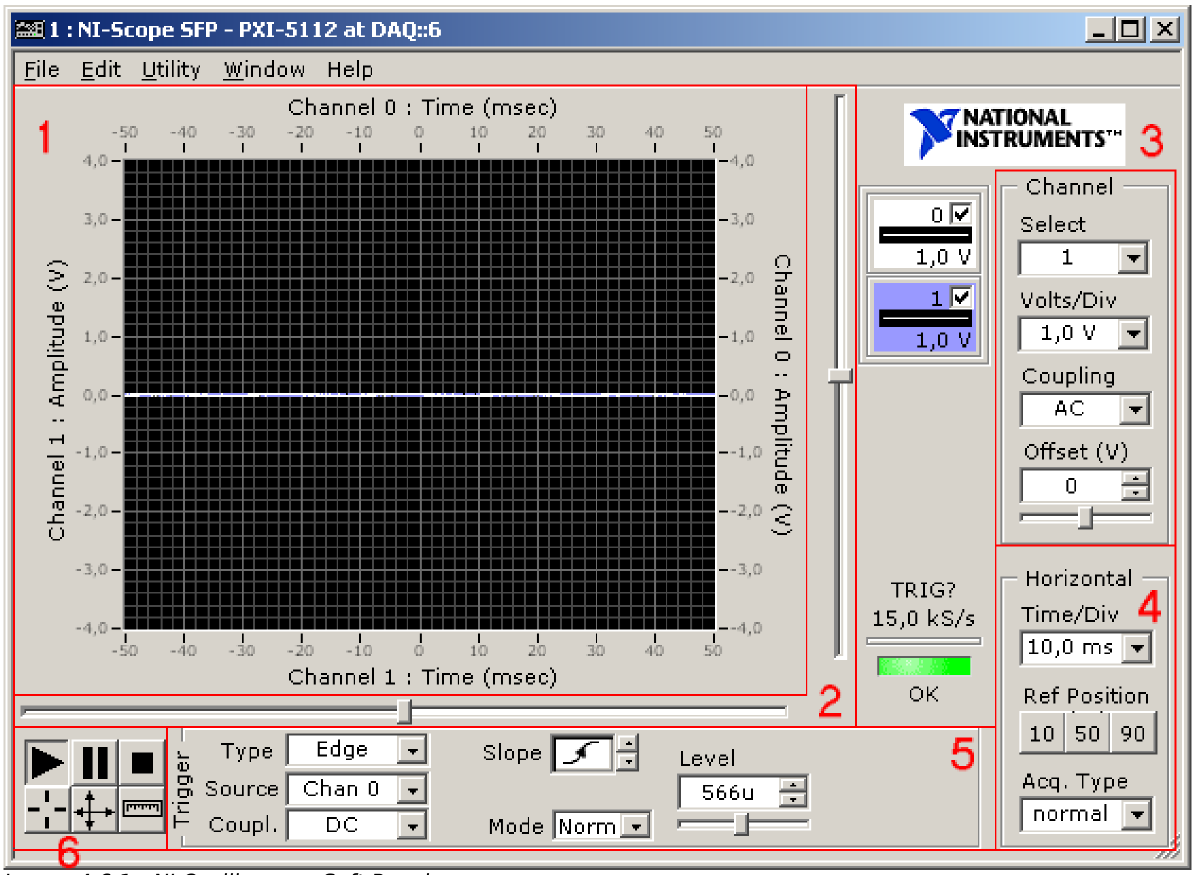
Image 4.5.2 : Flash Function Generator Front Panel

4.6 NI Oscilloscope Front Panel

The NI Oscilloscope Soft Panel is a Soft Panel for an Oscilloscope.

The National Instruments Oscilloscope Soft Panel shows a graph(1) at the top left. The graph is the result from a measurement performed by the oscilloscope. At the right and the bottom of the graph there are sliders(2) to move the graph. At the top right there are configuration options for the channel(3). There the user can select which channel he desires to set up, and set the volts per division, channel coupling and channel offset.

Below the channel settings there are the Horizontal settings(4). There the user can select the time per division and reference position. There is also an option for Acquisition Type, but there is only one option available.


1 : NI-Scope SFP - PXI-5112 at DAQ::6

Channel: Select 1; Volts/Div 1,0 V; Coupling AC; Offset (V) 0

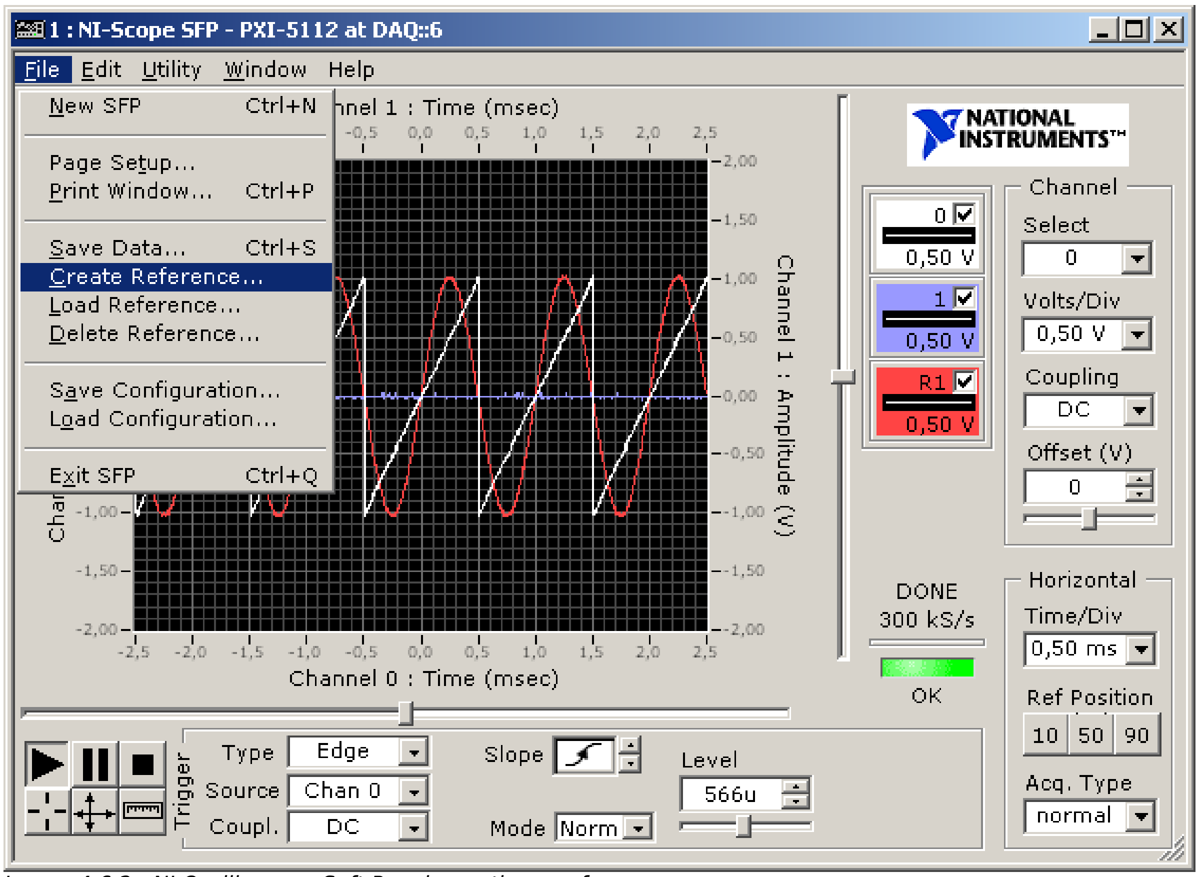
Horizontal: Time/Div 10,0 ms; Ref Position 10 50 90; Acq. Type normal

Trigger: Type Edge; Source Chan 0; Coupl. DC; Slope; Level 566u; Mode Norm

TRIG? 15,0 kS/s OK

Image 4.6.1 : NI Oscilloscope Soft Panel

At the bottom of the Soft Panel, the user can configure the trigger(5). The user can select a type, a source, coupling, slope, level and mode. Above the trigger set-up, next to the horizontal set-up, there is an indicator which shows if the configured trigger was received. Next to the Channel set-up the user can enable or disable the displaying of the channels. Through the File → Create Reference option, the user can save the current waveform as a reference graph.



0 ✓ 0,50 V
1 ✓ 0,50 V
R1 ✓ 0,50 V

Channel
Select: 0
Volts/Div: 0,50 V
Coupling: DC
Offset (V): 0

Horizontal
Time/Div: 0,50 ms
Ref Position: 10 50 90
Acq. Type: normal

Type: Edge
Source: Chan 0
Coupl.: DC
Slope
Mode: Norm
Level: 566u

Image 4.6.2 : NI Oscilloscope Soft Panel, creating a reference

The trigger cursor can be enabled through the Utility → Trigger Cursor Option. This will show a trigger cursor, a line, in the graph at the selected trigger level.

At the left bottom there are buttons(6) with play, pause and stop symbols. The start button will connect to the hardware and the stop button disconnect. The pause button will pause the acquisition. That buttons below will show cursors, perform auto set-up and display the measurement window.

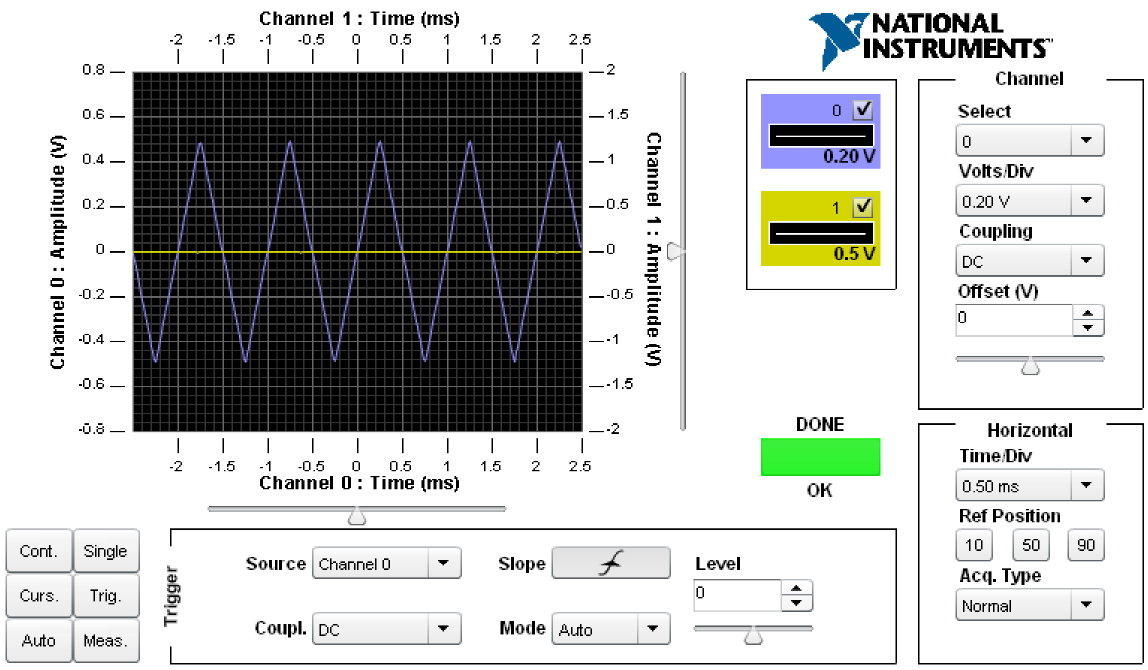
In the Flash Oscilloscope Front Panel, a few changes where made to match the requirements by the OpenLabs Electronics project and because of limitations caused by flash and running on a remote system.

The buttons to connect and disconnect from the hardware have been removed, since there is no connection to the real oscilloscope. Instead there are buttons to do a single measurement or continuous measurements. The options to view the trigger cursor has been added as a button into the interface, since there is no menu-bar in the flash interface.

The options available at the trigger settings are also different to match the options provided by the OpenLabs protocol the Flash Front Panels use to communicate.

The option to display a reference graph has also been left out, since there is no need for that option in the OpenLabs project.

The look of the Front Panels slightly differs because of the use of Flash Widgets. Different colours have been chosen to display the graphs to give a better image to the user.



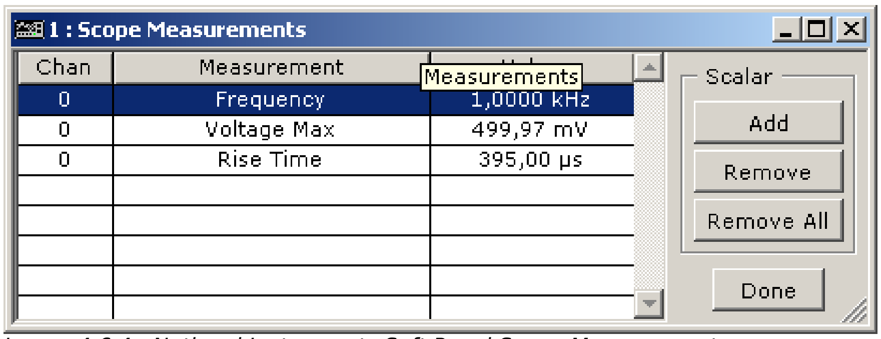
Channel 0 ✓ 0.20 V
Channel 1 ✓ 0.5 V

Channel
Select: 0
Volts/Div: 0.20 V
Coupling: DC
Offset (V): 0

Horizontal
Time/Div: 0.50 ms
Ref Position: 10 50 90
Acq. Type: Normal

Trigger
Source: Channel 0
Slope
Level: 0
Coupl.: DC
Mode: Auto

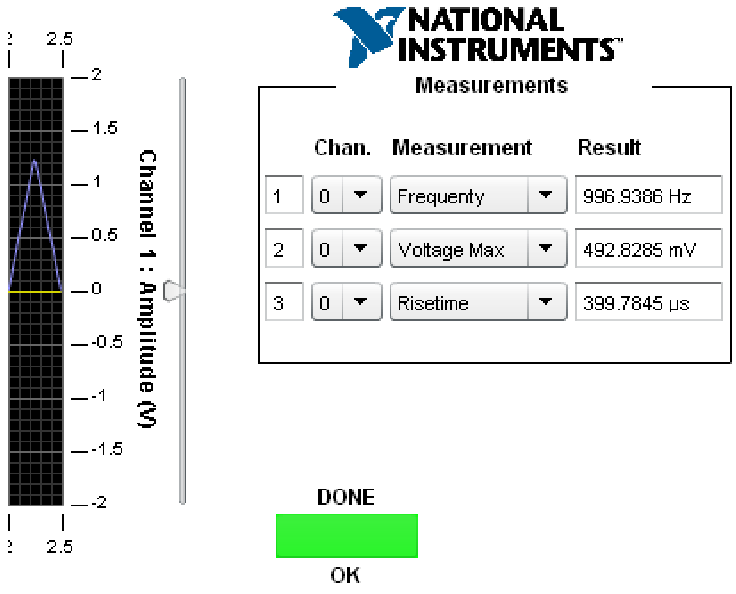
Image 4.6.3 : Flash Oscilloscope Front Panel



Chan	Measurement	
0	Frequency	1,0000 kHz
0	Voltage Max	499,97 mV
0	Rise Time	395,00 µs

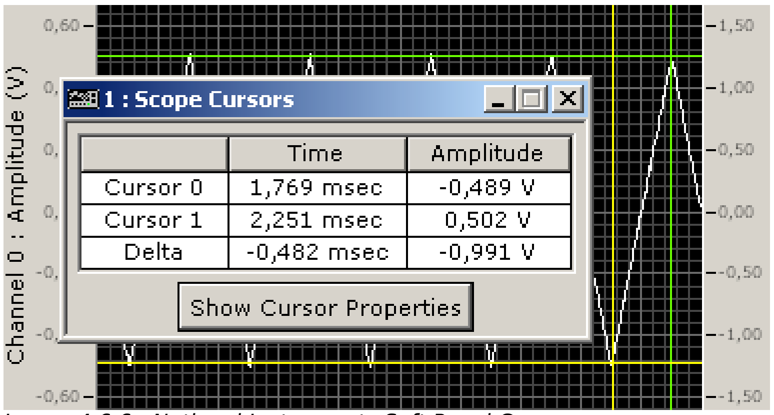
Image 4.6.4 : National Instruments Soft Panel Scope Measurements

Another difference between the National Instruments Soft Panel and the Flash Front Panel is the way cursors and measurements are displayed. The National Instruments Soft Panel displays them in a separate window. Flash has limited support for creating windows. Only overlaying windows are supported, but no windows hosted by the operating system. Because of this limitation it's only possible to draw windows within the main window. Because of this limitation, the cursors and measurements within the flash client will be displayed on a fixed position. When they are activated, the channel and horizontal set-up will be replaced by the requested option. The number of measurements has also been limited to three measurements since that is the maximal supported number of measurements by the protocol the Front Panels use to communicate.



	Chan.	Measurement	Result
1	0	Frequenty	996.9386 Hz
2	0	Voltage Max	492.8285 mV
3	0	Risetime	399.7845 µs

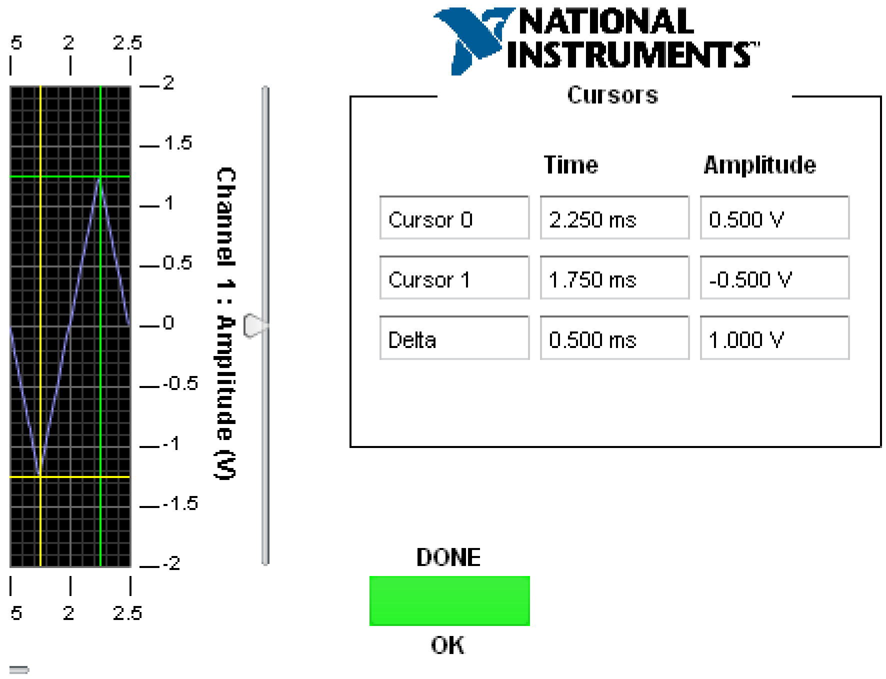
Image 4.6.5 : Flash Front Panel Oscilloscope Measurements



	Time	Amplitude
Cursor 0	1,769 msec	-0,489 V
Cursor 1	2,251 msec	0,502 V
Delta	-0,482 msec	-0,991 V

Image 4.6.6 : National Instruments Soft Panel Cursors

The cursors information in the Flash Front Panel will be displayed next to the graph, at the same position as the measurements. The cursors themselves behave the same as in the Soft Panel. The user can drag the cursors displayed in the graph to the desired position using a mouse and the values next to the graph will be updated to match the current cursor position.



	Time	Amplitude
Cursor 0	2.250 ms	0.500 V
Cursor 1	1.750 ms	-0.500 V
Delta	0.500 ms	1.000 V

Image 4.6.7 : Flash Front Panel Cursors

This Front Panel communicates, like all others, using an XML-based protocol. Since the oscilloscope Front Panel receives a graph, and not a single value like the other front panels, only the graph data reception will be explained.

The graph is sent as an array of five hundred (500) samples. The samples are transferred base-64 encoding. The data is base-64 encoded because the XML protocol does not allow binary data. Decoding the base-64 data will result in samples stored as a signed eight (8) bit value. This means they will have a value between -128 and 127. Since the graph area on the screen is only 240 pixels high, this resolution is more than adequate to display a good image.
To convert their values to the real values, a gain and an offset value is sent along with the samples. Each sample will be multiplied with the gain and the offset added to the result of the multiplication.

5. Proxies

As stated in the previous chapter, the Front Panels communicate using an XML protocol. The server can communicate in this protocol over a TCP connection. Some firewalls might block this traffic, so some students prefer to send data over an HTTP connection in stead.

Since the server does not support HTTP connections, a proxy is required. The proxy will receive the data over an HTTP connection and re-transmit it over a TCP connection to the measurement server. The second goal in this project was to create such a proxy, and which will be discussed in this chapter.

5.1 Direct CGI proxy

The current solution uses a basic proxy implementation in PHP. Since PHP is a interpreted language, it will be interpreted at run-time. This results in slower performance then when the proxy was binary code.

To solve this problem, a proxy implementation in C was written. CGI is an interface that allows an application written in C to be run within a web-server. This way, the application can be accessed using the HTTP protocol. The web-server runs on a FreeBSD system. This means the program must use the POSIX-API. Since the development machine runs Microsoft Windows, the development was done using CYGWIN, a UNIX-like environment for Microsoft Windows. This way the code could be developed and tested on the local workstation before uploading and testing it on the real web-server. It also is a way to ensure the code is portable to other UNIX-like environments sush as Linux and Solaris, with no or minor modifications.

5.2 Blocking sockets

The initial version waits for an incoming connection from the Flash Client, opens a connection to the measurement server, sends the data, and waits for the response. When all involved systems work correctly, this implementation works fine. In case there is a problem with the measurement server, the proxy would wait an infinite time for a response. This behaviour is not desired since in case the measurement server is down, many instances of the proxy would clog the web-server, eventually, bringing down the web-server as well, since every proxy instance would keep a connection open, and there is a limitation on the maximum number of allowed simultaneous connections.

5.3 Time-outs on sockets

It is possible to add time-outs to the send and receive calls on sockets, eliminating problems that might be caused by non-responding servers. However, the time-out for connecting is a hard coded value in the Operating System.

One solution to this problem is to use an alarm. An alarm is a timer that, when it expires, sends a signal to the program. When the program receives the signal, it will be interrupted, and execute the signal handler for the received signal. Using this method, a time-out shorter then the Operating Systems default can be used. The problem when using this method is the fact the signal to interrupt the connection can be sent by an other process running on the server, and there is no way to verify whether it was our own alarm that sent the signal.

By default, a socket is set to blocking mode. This means that a call will only return on success, failure or time-out. It's not possible to set a self-defined connection time-out on blocking sockets. An alternative to blocking sockets are non-blocking sockets. Non-blocking sockets will return immediately after they are called. When it returns, it's result might not be known yet. The result can be obtained by a select call to the socket. This select call does offer the possibility to set a time-out. This way it is possible to create a self-defined time-out on a socket connection.

The socket will be switched back to blocking mode after the connection was successful. Since we can define the time-outs for sending and receiving data, it's not necessary anymore to use the socket in non-blocking mode.
That would only need more code to wait for the data to be available and adds no functionality that's required for the proxy.

5.4 Using a daemon

The proxy implementation described in the paragraphs above work well, but has one disadvantage. The proxy will have to connect to the measurement server for each request. Making a new connection each for every request takes time, and even though it's a rather short time, when handling many requests, it might decrease the performance.

In order to solve that problem, a daemon can be used. A daemon is a program that runs in the background, and has no user interface, also called a service. The daemon will keep a connection to the measurement server open, and the CGI-program will send its request to the daemon in stead of directly to the measurement server. Since the daemon and the proxy run on the same machine, the time they need to communicate can be neglected.

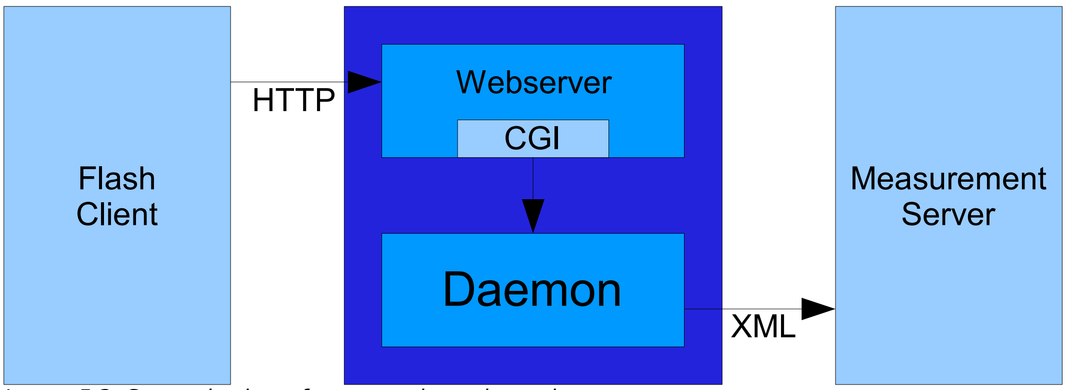


Image 5.3: Scematic view of a connection using a daemon

Apart from the connecting sockets, described in the previous paragraphs, the daemon also uses listening sockets. The listening socket will wait for an incoming connection. A listen call to a blocking socket will only return when there is an incoming connection. This means the program would wait forever if there is no incoming connection.

5.5 Keeping the connection alive

The daemon also has one connection open to the measurement server. During the waiting for an incoming connection, there will be no activity on this connection. When there is no activity over a socket connection, it is unsure if the connection is still alive. It's possible to set a keep-alive option to the socket. This option will send keep-alive packets,the other end of the connection will respond to those packets. This way it can be verified the connection is still alive. However, the time between the connection verifications is hard coded to two hours. This is a rather big timespan, and we would like to check the connection status more frequently.

To be able to check the status more frequently, the protocol has implemented a heart-beat. To send this heart-beat we need to interrupt the listen call to the socket. This can be done using an alarm, but the use of a signal has some unwanted side effects, as described in previous paragraph. Other processes can send this signal as well, and, in the case of the heart-beat, this may result in flooding the measurement server with heart-beat packets.

Instead, we can use non-blocking sockets here too. After setting the socket in non-blocking mode, a listen call is performed on the socket. The listen call will return immediately. Using the select call with a time-out, it's possible to wait for an incoming connection that will be interrupted when the time-out has expired. The status code the listen call returns indicated if there was an incoming connection or a time-out. In case there was a time-out, the heart beat will be sent to the server. The server will reply to the heart-beat, and the waiting for an incoming connection will be resumed.

This way it's possible to ensure that the connection to the measurement server is still alive, without risking the server to be flooded by external processes running on the same machine.

5.6 UNIX sockets

The communication between the proxy and the daemon runs over a TCP connection to the local host. Since the software runs on a FreeBSD machine, it's also possible to use UNIX sockets.

Unlike TCP sockets, UNIX sockets are file based. In a traditional UNIX environment everything is a file. Because UNIX sockets are file-based, an advantage is the possibility to limit the access to the socket. UNIX files can have access right depending on the user who is running the task. On UNIX platforms, a user does not necessarily mean a real person. One can create a UNIX user for a specific application, and set rights to files so that file can only be accessed by the specific application.

Apart from the advantage of controlling the access to the socket, UNIX sockets guarantee both ends of the socket run on the same machine. Therefore it's not necessarily to perform any checksum calculations or apply routing information. Fewer operations have to be performed on the data before it's sent, so UNIX sockets provide better performance than TCP sockets.

The implementation of the proxy and daemon only need minor modifications to use UNIX sockets. The final implementation of the daemon/proxy implementation supports both TCP and UNIX sockets.

6. Direct Instrument Control

The third objective was to create a way of directly controlling the instruments using the IVI API. This was also the biggest assignment. This chapter discusses the different aspects of the controlling the instruments directly.

6.1 IVI API

There exist an API to directly control measurement instruments. This API is created by the IVI Foundation and could be used to control any measurement equipment from any manufacturer that supplies IVI drivers for their products. This way, a program that uses the IVI API is not dependant on any specific hardware.

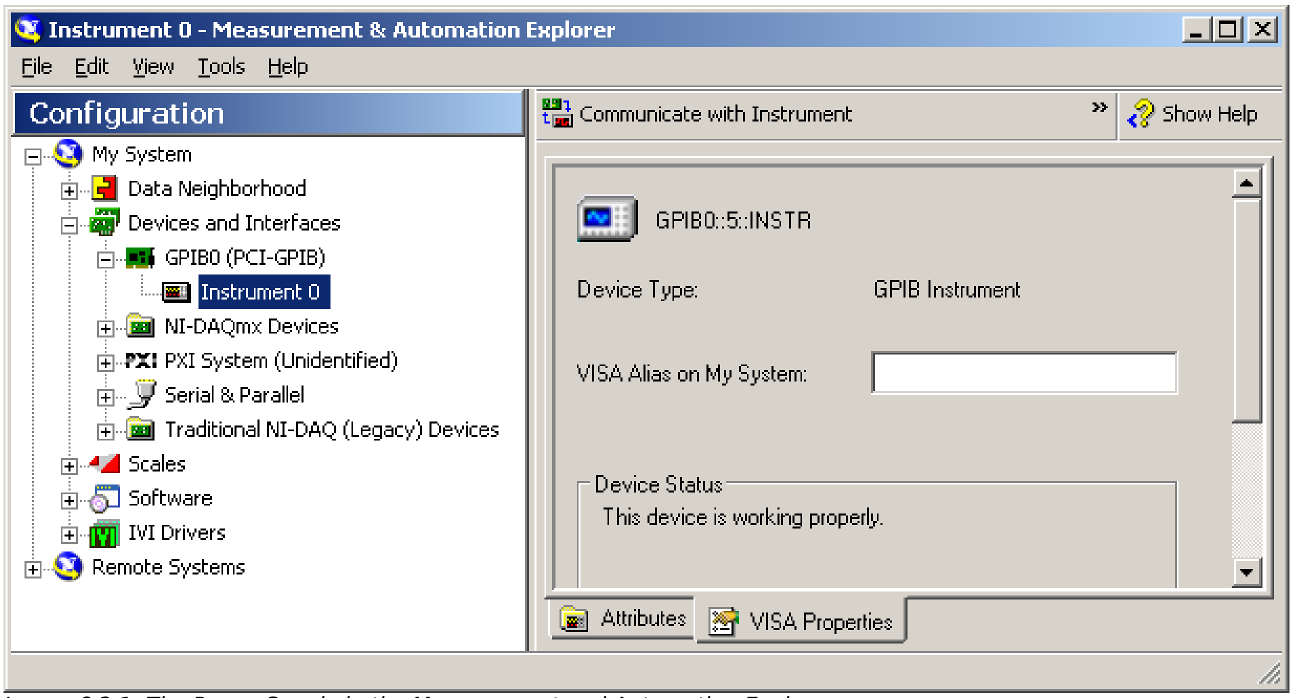
The first task within this assignment was to analyse the API and to see if it is possible to set an output signal or fetch a measurement. A simple test application was created for this purpose.

The API is simple,but there were a few unclarities that had to be solved. To use a driver, it shuld know which hardware it should use. The driver is to be told during the initialisation, but the API documentation does not specify how the equipment has to be identified. It seems this identification is manufacturer specific.

The hardware used during development differs from the hardware that's used in the actual OpenLabs Electronics Laboratory. The development system is using an older PXI box that has been used in a previous version of the OpenLabs Electronics laboratory. The power supply is connected through a GPIB interface in stead of PXI, while the Power Supply in the actual OpenLabs Electronics Laboratory is connected using PXI. The different interface is irrelevant to the development since the software interface is the same.

6.2 Hardware Identification

During the initialisation of the driver, the hardware has to be identified. The first equipment to be implemented was the power supply. This device is connected through the GPIB interface. The identification of the power supply is “GPIB0::5::INSTR”. This is interpreted as Instrument at GPIB bus 0, address number 5. This address appears in the “Measurement and Automation Explorer” software supplied by National Instruments.



GPIB0::5::INSTR

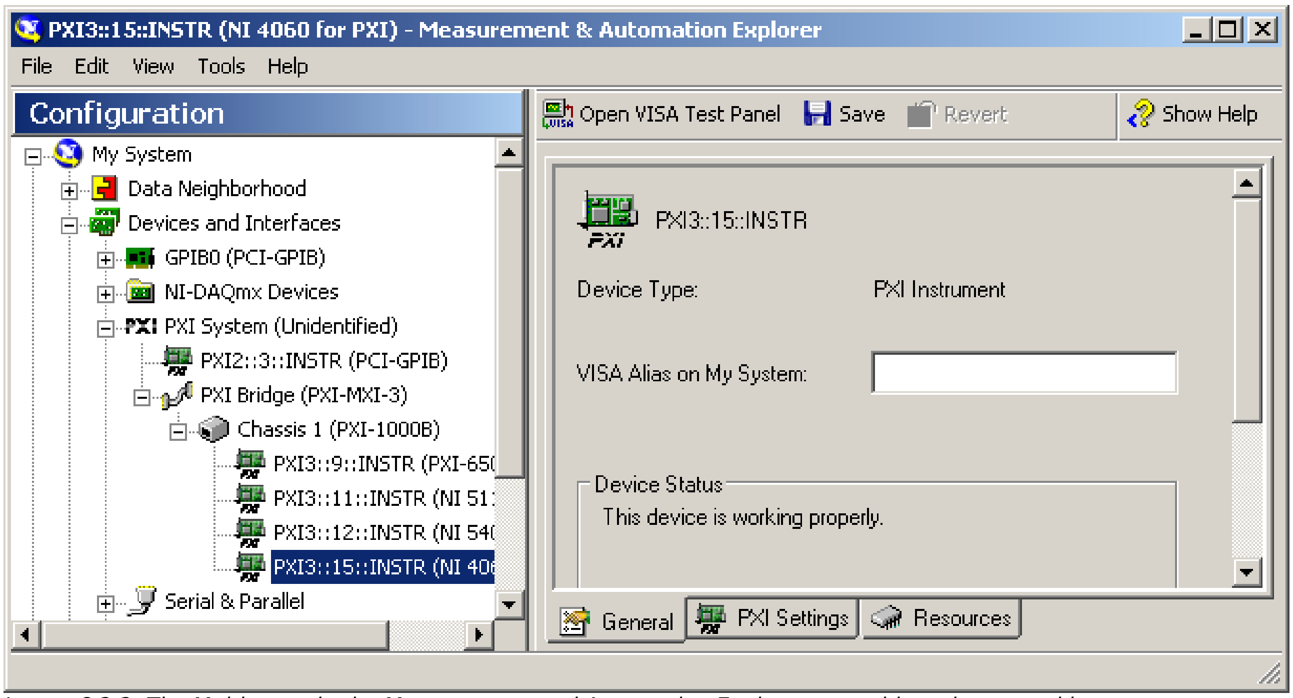
Device Type: GPIB Instrument

VISA Alias on My System:

Device Status
This device is working properly.

Image 6.2.1: The Power Supply in the Measurement and Automation Explorer

After the power supply was implemented, the multimeter was next to implement. The multimeter is connected using the PXI interface. This device appears as “PXI3::15::INSTR” in the “Measurement and Automation Explorer”



PXI3::15::INSTR

Device Type: PXI Instrument

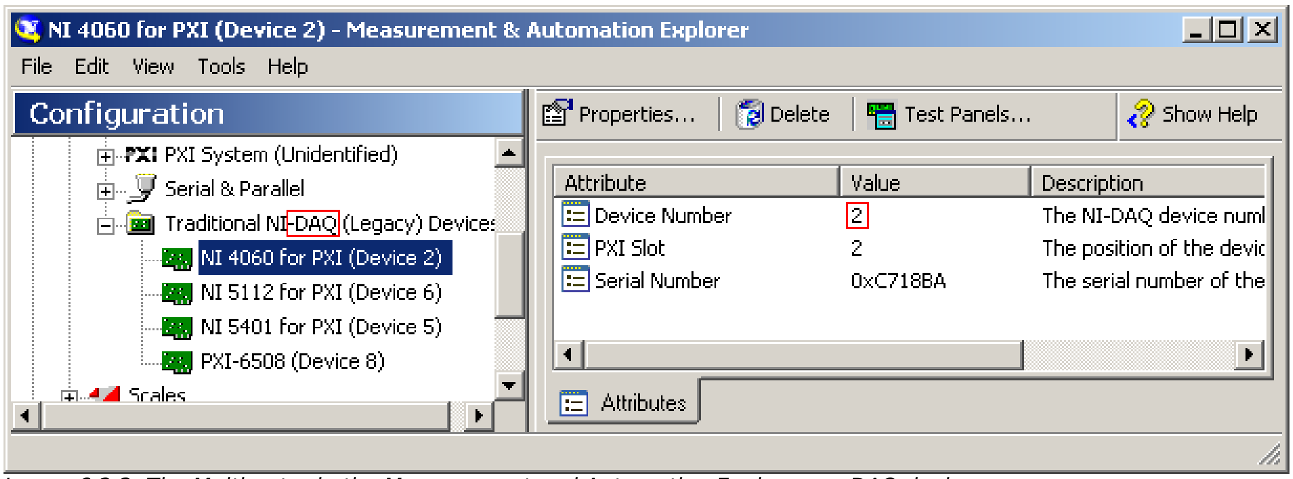
VISA Alias on My System:

Device Status

This device is working properly.

Image 6.2.2: The Multimeter in the Measurement and Automation Explorer, yet address is not working

Yet, trying to initialise the driver using this address resulted in a “hardware not found” error. It turns out, the drivers from National Instruments require a different addressing method. The correct value to initialise the National Instruments driver turns out to be it's DAQ address, which can be found in the Measurement and Automation Explorer too. (Image 6.2.3) The driver initialised correctly when using address “DAQ::2” in stead.


NI 4060 for PXI (Device 2) - Measurement & Automation Explorer

File Edit View Tools Help

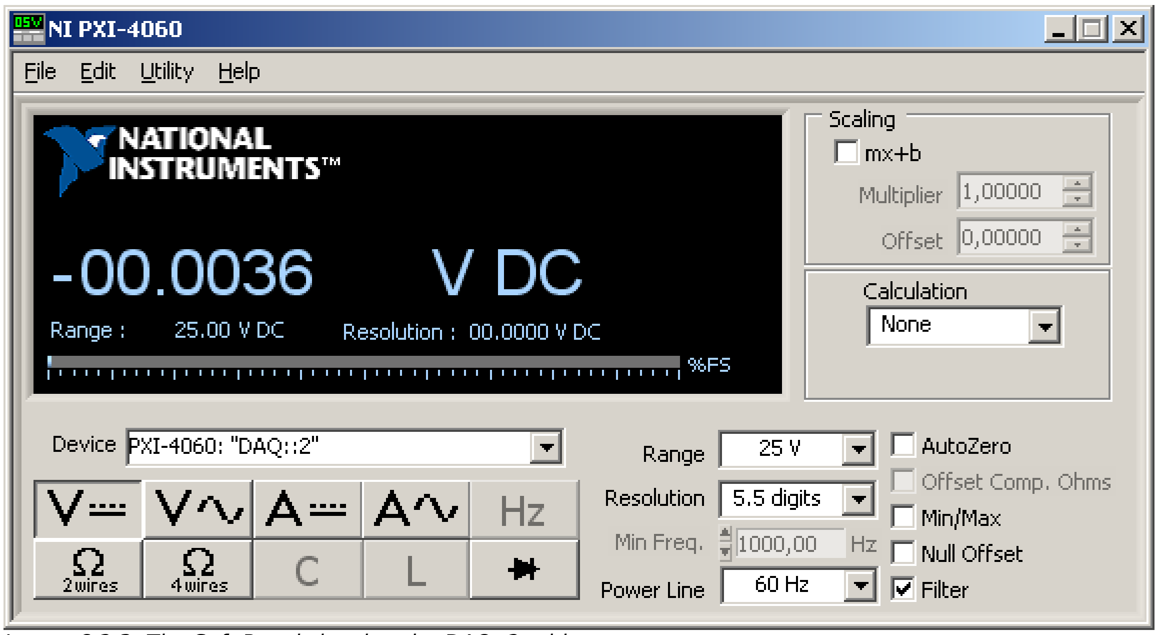
Configuration

- PXI System (Unidentified)
- Serial & Parallel
- Traditional NI-DAQ (Legacy) Devices
  - NI 4060 for PXI (Device 2)
  - NI 5112 for PXI (Device 6)
  - NI 5401 for PXI (Device 5)
  - PXI-6508 (Device 8)
- Scales

Attribute	Value	Description
Device Number	2	The NI-DAQ device numl
PXI Slot	2	The position of the devic
Serial Number	0xC718BA	The serial number of the

Image 6.2.3: The Multimeter in the Measurement and Automation Explorer, as DAQ device

The correct addressing for National Instruments equipment is also visible in the National Instruments Soft Panels.


NI PXI-4060

File Edit Utility Help

-00.0036 V DC

Range : 25.00 V DC Resolution : 00.0000 V DC

Device PXI-4060: "DAQ::2"

Scaling: mx+b [ ]; Multiplier 1,00000; Offset 0,00000

Calculation: None

Range: 25 V

Resolution: 5.5 digits

Min Freq. 1000,00 Hz

Power Line: 60 Hz

AutoZero [ ]; Offset Comp. Ohms [ ]; Min/Max [ ]; Null Offset [ ]; Filter [✓]

Image 6.2.3: The Soft Panel showing the DAQ::2 address

After the addressing problem was solved, it was easy to get this and the other drivers from NI working

6.3 Loading the driver

The first implementations of the interface with the driver loaded the specific driver for the device directly. This driver implements the IVI API, but only supports one specific device. This would result in a program that only supports one specific type of hardware.

It is possible to load a driver for a specific type of hardware through a main driver, which only implements the API but does not support any hardware. This feature obtains pointers to all the IVI functions that exist in the specific driver. Every function pointer has to be obtained one by one and results in a lot of extra code in the program. Therefore this solution is not the desired solution.

Fortunately, National Instruments offers another option to load a specific driver from a main IVI driver. To use this option, a number of settings need to be made within their Measurement and Automation software.

Using the Measurement and Automation software, a user defined identification for the measurement equipment can be defined. The main IVI driver can be initialised using this identification.

6.4 From C to C++

After the basic driver handling was working, the code should be integrated into the measurement server. Since the measurement server is written in C++, so should the code that is to be integrated. Therefore a class for each driver had to be created.

Each class should initialise the measurement equipment it's going to be used with the equipment's identification, perform equipment set-up and fetch the resulting data. After use the equipment should be unloaded correctly.

It's possible to send the initialisation parameters to the constructor of the class, however, a constructor cannot return a value, therefore it's hard to indicate possible errors. In stead, an initialisation function is used, that's to be called after the creation of the class. This function can return a value and so the status can be checked easier.

All measurement equipment, with exception of the multimeter, support multiple channels. Each channel works independent. For example a power supply can generate different voltages on each of it's output channels.

6.5 Power Supply Class

Upon initialisation of the Power Supply Class, a Power Supply Driver will be loaded. If the driver is loaded successfully, the channels will be requested. For each channel, a channel object will be created. Within the IVI-API channels are identified by a string, which usually contains a number. Since we want to identify the channels by their output range (+6 Volt, +20 Volt or -20 Volt) the driver will request the maximal output voltage from each channel and assign the correct range to each channel. This value will also be stored within the channel objects. This way, it is possible to address each channel by it's type, not depending on it's name used internally by the IVI driver.

For each channel it is possible to set an output voltage and a maximal output current within the limits supported by the equipment. It's also possible to request the current output voltage and current and to check the current limitation has been reached or not.

6.6 Multimeter Class

There is only once function available to perform a measurement. This function accepts the desired measurement, measurement range and resolution as parameters. Supported measurement by the IVI standard can be seen in the IVI API Documentation [IVIAPI] Chapter 4.2 page 21. Note that not all of there measurements have to be supported by the measurement equipment. The driver will generate an error code when an unsupported measurement is requested. For example, the National Instruments multimeter does not support a temperature measurement.

National Instrument has added an extension to their driver to also support AC volts DC coupled, Diode measurement, Waveform Voltage, Waveform Current, Capacitance and Inductance. Support for these extensions can be enabled and disabled during compilation of the code.

6.7 Function Generator Class

A function generator can support multiple channels, and therefore channels must be requested during initialisation, even though the used function generator only has one channel.

The function generator class has a few functions to set up the output signal. The properties of the output signal that can be defined are the Amplitude, DC Offset, Duty Cycle, Frequency, Output Impedance and the Waveform. The waveform can be a sine wave, a square wave, a triangle, a rising ramp, a falling ramp, or a DC Voltage. National Instruments also added some extensions, and added support for noise and user defined waveforms.

6.8 Oscilloscope Class

The oscilloscope has two channels, and during initialisation their names will be requested.

To obtain a waveform from the oscilloscope takes two steps; the configuration and the acquisition. The configuration consists of two parts, the general configuration and a per channel configuration.

The general configuration takes the following parameters: the time the acquisition is supposed to take, the number of samples the result will be, and the time the acquisition is supposed to start relative to the trigger. The trigger has to be configured with the trigger coupling, trigger level and trigger slope.

The channel configuration takes the following parameters: range, offset, channel coupling, probe attenuation. The channel will also be configured with the input impedance and a maximum frequency. When the maximum frequency parameter is set, it will apply a low-pass filter to the input signal.

After configuration the actual acquisition of the channel can be initialised. There are several ways of acquiring and obtaining a waveform from a channel. When the ReadWaveForm function is called, it will acquire the waveforms from both channels, but return the data from the channel it was called on. Since the ReadWaveForm function acquired the waveform from the other channel as well, it's not necessary to call it again. This would only waste the time it takes to do another acquisition. In stead the FetchWaveForm function is called on the other channel to obtain the waveform for the other channel. This way the waveform for both channels are obtained with only one acquisition.

Apart from obtaining waveform from the channels, the oscilloscope also has the ability to perform measurements on the signal. The measurement function is called with the requested measurement. Possible measurements can be seen in the IVI API Documentation [IVIAPI] Chapter 4.1, page 122 and following.

The measurement function is called with one of these measurements and the channel to measure on as parameters and will return it's value.

The OpenLabs Electronics software requires a feature that's not supported by IVI, therefore this feature has to be implemented elsewhere.

This feature is auto configuration. This means setting a suitable time and voltage per division depending on the signals offered on the oscilloscope channels.

In order to determine the values that are best suitable for the offered signals, the AutoSetup function will perform measurements on the signal to obtain the minimal and maximal voltage of the signal. Using these values the voltage per division and offset voltage will be determined. When the voltage levels have been determined another measurement will be performed to obtain the period of the signal. Using the result from this measurement the time per division will be determined.

6.9 Different Hardware

The development machine that was used to develop the IVI Driver classes contains hardware that differs from the machine currently in use for the OpenLabs Electronics Project. The used hardware is shown in the table below.

Hardware type	Development hardware	Production hardware
Power supply	Agilent E3631A	NI-4110
Multimeter	NI-4060	NI-4072
Function Generator	NI-5401	NI-5402
Oscilloscope	NI-5112	NI-5112

In theory, using different hardware should not cause a problem, since the software interface is the same. In practice, there were some problems.

During the initialisation of the power supply, the maximal output voltage per channel is requested per channel. While requesting the maximal voltage, a current should be included in the request. On the Agilent power supply the correct value was returned when this parameters was set to 0 Ampère. On the National Instruments power supply this request would result in a return value of 0 Volts. A solution to this problem is to request the maximal current first, with a parameter of 0 Volts, however on the Agilent power supply, this would result in an out-of-range error in the return value. To make the initialisation work correctly on both power supplies, the maximal current is requested and divided by two. Using this current value as parameter when requesting the maximal voltage returns the correct result for both power supplies.

The NI-5401 Function Generator allows the output signal to be changed when the output signal is still being generated. Changing the output signal when the output signal is still being generated is allowed according the IVI specifications. However, when using the NI-5402 Function Generator, trying to change the output signal when a signal is still being generated resulted in an error being returned saying it is not allowed to change the output signal when the signal is still being generated. The solution is to stop the generation before changing the signal. Stopping the generation before changing the signal causes no problems for the NI-5401 Function Generator.

Even though the same oscilloscope is used in both systems, their behaviour is different. Possibly through different driver version. When no trigger was received on the development machine the oscilloscope would configure a suitable trigger value itself, however on the running production machine an error would occur. The oscilloscope driver class has to handle both cases correctly. When the driver returns a waveform, it will validate the channel against the configured trigger level. Depending on the requested trigger mode, appropriate action will be taken. If the trigger mode was set to 'normal' the signal will be abandoned, in other cases the signal will be returned. When a no-trigger-error is received, and the trigger mode was not set to 'normal', a measurement will be performed on the signal to determine a valid trigger level and the signal acquisition will be requested again. Besides the 'normal' trigger mode it's possible to have an 'auto' and an 'auto-level' trigger mode. Both will return the signal, even when the trigger is invalid. The difference is that the 'auto-level' will also return a suitable trigger level for the current signal, while the 'auto' trigger will only return the signal.

6.10 The Matrix

For the LabView server to be replaced by code that interfaces with the hardware directly, the Matrix has also to be controlled from the measurement server.

The Matrix has an USB interface and uses a simple string based protocol. In order to communicate with the Matrix, libusb is used. Libusb provides a simple interface to USB devices.

Like the measurement equipment, the code to communicate with the matrix has also to be written in the form of a C++ Class.

Next to the initialisation the class has a function to send and receive data from the matrix. The current matrix only supports sending data to it. A function to receive data has been implemented for future use.

The matrix consists of a few stacked boards, which either are connected to measurement equipment or contain components. The strings sent to the matrix specify which relays are to be opened and closed on which boards. Opening and closing relays will create the electronic circuit the student has sent to the server to measure on.

In order to decide which relays should be opened and closed, a list of the components available on the boards is supplied to the server. This data is processed elsewhere in the server and will provide a list with relays to open and close to the module.

With the list of relays, it will create the commands that are to be sent to the matrix. During the circuit build-up, its relays will be opened and closed per card. An incomplete circuit will contain incorrect connections. When power is available in the circuit this can lead to a short circuit, possibility damaging the equipment. Therefore it's important to guarantee the circuit is not powered during set-up. To guarantee the circuit is not powered during set-up, all instruments will be disconnected during set-up. They will only be attached to the circuit after it's completed. Since the component cards have an address below 16 and the equipment cards have an address above 16, setting them up in increasing order guarantees then equipment to be attached after the component cards.

7. Conclusions

The additional flash Front Panels that have been created offer students a choice on how to operate the OpenLabs Electronics interface. The flash modules are based upon the National Instruments Soft Panels. The Soft Panels are designed to be used on a computer screen, therefore the Front Panels based upon these Soft Panels are easier to operate then the original Front Panels based upon equipment that is not computer based.

The new CGI based proxy has a performance advantage over the older PHP based proxy, and when this proxy is used, it will increase the overall performance of the OpenLabs Electronics system.

The integrated IVI and USB drivers into the measurement server will make the need of a LabView server obsolete. Controlling the equipment directly from the measurement server results in a performance advantage, and removes the dependency of LabView. Less factors the system depends on will result in a more reliable system, and it reduces the cost for running the system since a LabView license is not required.

Evaluation

The time I've spent in Sweden doing my internship at Blekinge Tekniksa Högskola has been a great time. During my internship I've met some new technologies I hadn't used before.

During my internship an Agile software development has been used. I've liked this way of working since this development method doesn't have much bureaucracy interfering with creativity and slowing down development.

During my first assignment I've learned and written software using ActionScript 3.0 in Adobe Flash CS3. This is a coding language and environment I haven't used before my internship. I've learned about some possibilities of this development environment and language.

My second assignment has given me more experience with BSD Socket programming in C. Before my internship I hadn't coded for a FreeBSD system before. Although it's a POSIX compatible system, there are some differences with for example Linux when it comes to socket programming. I've also been using some more advances features of sockets then I've done before my internship. I've learned about some features sockets offer I didn't know of before.

My third and last assignment has given me more experience with Microsoft Visual Studio Express 2008. I hadn't used this development environment before my internship. Working on the project provided me with more experience in the programming language C++ and the use of libraries. I've also learned about the IVI API to control measurement equipment. The IVI API is an API to control measurement instruments. Even though it's a standard, there are implementation differences between different drivers.

List of References

Books:

UNIX Network Programming, third edition:
ISBN: 0-13-141155-1

Websites:

GNU C Library Documentation:
http://www.aquaphoenix.com/ref/gnu_c_library

IVI API Documentation [IVIAPI]:
http://ivifoundation.org/specifications/default.aspx

OpenLabs Client Protocol documentation
http://svn.openlabs.bth.se/trac/measureserver/wiki/ClientProtocol

OpenLabs Equipment Protocol Documentation:
http://svn.openlabs.bth.se/trac/equipmentserver/browser/branches/usbmatrix/Docs/Protv4.pdf

Virtual Instrument Systems in Reality (VISIR):
http://www.bth.se/tek/asb/research/visir

Delivered source code:

Since all code produced in this project is released under the GPL License, the code is public available. The code can be browsed using a trac interface using the links below.

CGI Proxy and Daemon:
http://svn.openlabs.bth.se/trac/openlabsweb/browser/trunk/sites/electronics/cgi_proxy

Flash Front End modelled after National Instruments Soft Panels:
http://svn.openlabs.bth.se/trac/flash/browser/trunk/NI_DCPower
http://svn.openlabs.bth.se/trac/flash/browser/trunk/NI_DMM
http://svn.openlabs.bth.se/trac/flash/browser/trunk/NI_FGEN
http://svn.openlabs.bth.se/trac/flash/browser/trunk/NI_OSC

IVI & USB control code for the measurement server:
http://svn.openlabs.bth.se/trac/measureserver/browser/trunk/src/ivicontrol

OpenLabs Electronics

A Remote Electronics Laboratory

Project Plan

Name	André van Schoubroeck
Date	18 November 2008
Version	0.3

Index

1. Introduction to the Project

The OpenLabs project is an umbrella project for several projects that share the similar goals. The OpenLabs project is a project for distance students that allows them to do experiments over the internet. At the moment the project within the OpenLabs project include Antenna theory, Electronics, Security and Vibration Analysis.

During my internship I will be a part of the electronics laboratory team. I will investigate new possible features and methods to improve the laboratories. For example create alternative front panels in Flash, and create new ways of accessing the measurement hardware.

The electronics lab consists of three parts: an interface to the hardware written in LabView, a measurement server written in C++, and a front panel written in ActionScript 3 (The language Adobe Flash uses).

2. Project Statements

2.1 Formal Client

The OpenLabs project is a project at Blekinge Tekniska Högskola, therefore BTH is the formal client.

2.2 Project Leader

The project leader is Ingvar Gustavsson, and the supervisor is Johan Zackrisson.

2.3 Initial Situation

The initial situation of the OpenLabs electronics project is a running project that's currently serving many students with measurements.

The project contains three components. An *equipment server*, written in LabView. It communicates with the measurement equipment. A *measurement server*, written in C++. This server manages the measurements and requests the measurements at the equipment server. The measurement server evaluated the circuit and makes sure it cannot do any harm to the school's equipment before sending it to the equipment server. The final part is a flash client. This is the part the student sees, it is written in ActionScript 3.0 / Flash CS3 and this piece of software allows the student to compose an electronics circuit on the screen, attach measurement devices to it and perform a measurement on the circuit.

2.4 Project Justification

The OpenLabs electronics project is currently serving many students with measurements. Students on a distance learning course, who might be used to different measurement equipment. For this reason, I will create additional front panels to give the students a choice which front panel they would like to use. A different front panel would not change the measurement results, but the controls and results will be presented in a different way.

The option to have different front end available is to make the use of the system easier for the student, since he can choose for an interface he has used before, and also when the student operated the real measurement equipment he already knows how to operate it since it works the same as the on-line version.

The front panels I will create are front panels that look like the soft panels used by National Instruments. The choice to create front panels that look like the soft panels from National Instruments is also because the used measurement equipment is from National Instruments and showing their name is also a way of promoting them, since National Instruments is supporting the project.

I will implement a new proxy written in C. This proxy will run on a FreeBSD machine. In order to write this proxy, the possibilities of POSIX sockets will have to be explored and applied in this proxy.

I will also explore new ways to communicate with the measurement equipment. The current solution uses LabView to communicate with the equipment, but it would be interesting to investigate the possibilities to access the equipment in C++. If that can be done, it would be possible to integrate the equipment and measurement servers into one program.

2.5 Project Product

During my internship I will create front panels looking similar to the soft panels by National Instruments. These will be written in Flash (ActionScript 3). If necessary I will extend the measurement server (written in C++) to support the features the National Instruments soft panels support and are desired to be supported by the front panels.

The Measurement Instruments I will work at are:

- Power Supply
- Multimeter
- Oscilloscope
- Function Generator.

I will create a proxy implementation written in C to replace the current proxy written in PHP. This will be done because C offers better performance then PHP.

I will also investigate the possibilities to control the measurement equipment directly from C or C++. The measurement equipment uses an industry standard API: IVI. IVI is short for “Interchangeable Virtual Instrument”, and is supported by almost all measurement devices.

2.6 Project deliverables and non-deliverables

The deliverables in this project will be, among others, new front panels for the electronics labs web (Flash) interface.

Information and solution about the use of IVI drivers directly from C or C++ code, without using LabView. A prototype implementation will be provided and a prototype of a measurement server with integrated with the IVI driver classes I will create.
I will also provide a guide how to configure a system to be used with this software in combination with different types of measurement equipment (GPIB or PXI) .

All source code generated within this project will be released under a GPL license.

An internship report and presentation will be provided according to the requirements of Fontys University Of Applied Sciences requires for a internship report. The report and presentation will be given to both Fontys University of Applied Sciences and Blekinge Institute of Technology. Blekinge Tekniska Högskola will apply the same requirements as Fontys University of Applied Sciences.

2.7 Project constraints

BTH will have to provide me with a computer supplied with all software needed to preform my activities. This will include the Microsoft Windows XP operating system, the Microsoft Windows Visual Studio 2008 Express IDE and compiler, Adobe Flash CS3, National Instruments Soft Panels, drivers and the Measurement and Automation Explorer.

The computer will need to be equipped with a PXI and a GPIB interface to communicate with the measurement equipment, and an internet connection is desired to supply information.

The required hardware will need to be provided, including a National Instruments PXI box with a PXI Multimeter, a PXI Power Supply, a PXI Function Generator and a PXI Oscilloscope. Alternative hardware with a GPIB interface can be used as well.

2.8 Project risks

A possible risk during this project is possible data loss. The minimize the risk of data loss, source code will be submitted to an subversion (svn) server. This way the code will always be on-line and it will always be possible to revert to an older version of the code in case it stops working.

3. Development Method

During this project an agile software development methodology will be used. Therefore there will be no traditional long-term planning.

Agile software development is not a methodology, but rather a family of methodologies that share many similarities. Agile software development is a method of software development best suitable for small developing teams.

In agile software development the software will be developed in small pieces with minimal planning. Such a piece is called an iteration, and it usually takes a few weeks.

After each iteration the result will be reviewed and the priorities will be re-evaluated. Because of this agile software development is a flexible way of developing software. It's easy to adapt to changes in the requirements.

Agile software development emphasizes face-to-face communication over written documents. Progress is mainly being monitored by delivered working pieces of software. A working piece of software will be delivered frequently.

4. Communication Plan

As stated in the previous chapter, during we will use agile software development. This methodology of software development requires frequent face-to-face communication with the involved people.

Apart from the daily contact with the other team members I will make a weekly report of my activities the past week, and an overview of my planned activities the coming week and email it to my teacher at Fontys (Dick van Schenk Brill).

A Flexible Electronics Laboratory with Local and Remote Workbenches in a Grid

I Gustavsson[1], J. Zackrisson[1], K. Nilsson[1], J. Garcia-Zubia[2], L. Håkansson[1], I. Claesson[1], and T. Lagö[3]
[1] Blekinge Institute of Technology/Signal Processing, Ronneby, Sweden
[2] University of Deusto/Faculty of Engineering, Bilbao, Spain
[3] Axiom EduTech AB, Falkenberg, Sweden

Abstract— **The Signal Processing Department (ASB) at Blekinge Institute of Technology (BTH) has created two online lab workbenches; one for electrical experiments and one for mechanical vibration experiments, mimicking and supplementing workbenches in traditional laboratories. For several years now, the workbenches have been used concurrently with on-site ones in regular, supervised lab sessions. The students are encouraged to use them on a 24/7 basis for example, in preparation for supervised sessions. The electronic workbench can be used simultaneously by many students. The aim of a project known as VISIR (Virtual Systems in Reality) founded by ASB at the end of 2006, is to disseminate the online lab workbenches using open source technologies. The goal is to create a template for a grid laboratory where the nodes are workbenches for electrical experiments, located at different universities. This paper focuses on standards, pedagogical aspects, and measurement procedure requirements.**

Index Terms—**Electronics, Grid, Remote labs, Workbench.**

I. INTRODUCTION

For centuries, scientists have performed physical experiments in order to verify and test theories, and to create proper mathematical models, to describe reality well enough. Such experiments are the only way to "communicate" with nature and to learn its principles. Only recently has it become evident that mankind must live in symbiosis with nature and focus on sustainability and understanding. Thus, the demand for experimenters will increase. However, during recent decades, the amount of hands-on laboratory work, for example, in engineering education has been reduced. The prime cause is clearly due to the task of handling the dramatically increased number of students, whilst staff and funding resources have scarcely changed [1].

Reducing the number of lab sessions is easy because laboratory work is seldom evaluated, and the cost reduction obtained is often considerable. However, for example, ABET (Accreditation Board for Engineering and Technology) in the USA has demonstrated that learning objectives for laboratory work must exist and subsequently, be evaluated [2, 3]. Thus, the amount of hands-on laboratory work in a course must be correlated to its learning objectives. Unfortunately, a substantial rise in base funding resources is unlikely to manifest itself in a real life environment.

It is, of course, fundamental for students to understand theories and mathematical models. Appropriate and low cost tools are often hand calculations and simulations. The use of computer simulations has increased very much in engineering education in the last few decades however, to properly assess differences between mathematical models and the real world, experiments are clearly indispensable [4, 5]. On the other hand, traditional laboratories have limited accessibility and high running costs.

Nowadays, students want extended accessibility to learning resources and increased freedom to organize their learning activities, which is also one of the main objectives of the Bologna Process. From a technological perspective, such flexible education corresponds to an adequate exploitation of information, communication devices and infrastructures, especially the Internet. Today, many academic institutions offer a variety of web-based experimentation environments, so called remote laboratories, that support remotely operated physical experiments [6-9]. This is one way to compensate for the reduction of lab sessions with face-to-face supervision.

The remote, or online laboratories around the world, are used in a variety of disciplines. However, the wide range of user interfaces is a problem for students and teachers. Efforts are being made to address this situation. The iLabs project at Massachusetts Institute of Technology in the USA, for example, has developed a suite of software tools that facilitates online complex laboratory experiments, and provides an infrastructure for user management [10]. A somewhat different approach would be to create a grid laboratory where the nodes are online lab workbenches, distributed among a number of universities or other organizations. In such a laboratory, intended for the same type of experiments, it would be possible to organize supervised lab sessions with as many students, or student teams working concurrently as are optimal, for one instructor. Such supervised lab sessions could, for example, take place in a traditional laboratory where some students could use the local lab workbenches and others could perform the experiments remotely, on distant grid nodes. Then it should be possible for each university to offer more time in the laboratory for its students.

In 1999, ASB began a remote laboratory project. Today, ASB has two online lab workbenches; one for electrical experiments and one for mechanical vibration experiments, based on the BTH Open Laboratory concept [11]. The concept is about providing new possibilities for students to do laboratory work and become experimenters, by adding online lab workbenches to traditional instructional laboratories to make them more accessible for students, whether they are on campus or mainly off campus. These workbenches are equipped with a unique interface enabling students to recognize, on their own computer screen, the instruments and other equipment that most of them have previously used in the local laboratory.

At the end of 2006, ASB started the VISIR project together with National Instruments in the USA and Axiom EduTech in Sweden, to disseminate the online laboratories at BTH using open source technologies. Axiom EduTech is a supplier of education, technical software, and engineering services for noise and vibration analysis. The project is financially supported by BTH and by VINNOVA (Swedish Governmental Agency for Innovation Systems).

What type of instructional laboratory would be feasible for creating a template for a grid laboratory? There are reasons for starting with a grid laboratory for electrical experiments:

- There are electronics laboratories at most universities around the world containing the same equipment, (oscilloscopes, waveform generators, multi-meters, power supplies, and solderless breadboards) although models and manufacturers may vary. Such laboratories are already in a way, a de facto standard.
- There are standards defining the functionality for instruments common in an electronics laboratory. The IVI Foundation is a group of end user companies, system integrators, and instrument vendors, working together defining standard instrument programming interfaces [12].
- Today, BTH has an online electronics laboratory running in regular education where the software produced is released as open source code [13].

This template can be used for designing grid laboratories for other areas of interest. ASB has identified a laboratory for mechanical vibration experiments as a strategic and appropriate candidate, because those lab workbenches are very expensive and mathematical models generally provide a too simplified picture of the reality, even for introductory courses. Measured vibration signals frequently exhibit complicated properties compared to the vibration signal models frequently utilized in education. Selecting appropriate estimators and estimator settings, enabling extraction of different accurate estimates of vibration quantities from measured vibration signals, generally provides a substantial challenge for the inexperienced person. Moreover, the dynamic properties of a mathematical model of a structure and the actual dynamic properties of said structure, generally differ. Of significance; an online mechanical vibration laboratory provides the opportunity for engineering students to access the practical and theoretical knowledge advancement in experimental vibration analysis that is highly attractive for the industry.

II. THE OPEN ELECTRONICS LAB AT BTH

An experiment is a set of actions and observations, performed in the context of solving a particular problem. Experiments are cornerstones in the empirical approach to acquire a deeper knowledge of the physical world but also an important approach to verify that a model is accurate enough. The experimenter sets up, and operates, the experiment with his/her hands and/or with actuators. As an example, a lab workbench in an instructional laboratory for low-frequency analog electronics at BTH is shown in Fig. 1. The student wires a test circuit on the breadboard using his/her fingers and uses instruments to measure what s/he cannot perceive directly with human senses as, for example, the electrical current. Experiments that are possible to perform in this environment are mainly limited by the set of components provided by the instructor.

In instructional laboratories at most universities, there are a number of lab workbenches where the same number of students, or usually a pair of students, perform experiments supervised by an instructor. The students are permitted to be in the laboratories only during lab sessions when an instructor is present. The number of lab workbenches in a laboratory is usually selected, considering how many students an instructor can supervise if a workbench is not too expensive. Typically, electronics instructional laboratories are equipped with eight identical workbenches. Fewer lab workbenches mean more teaching hours per course but less investment. It is a pedagogical advantage if the lab workbenches are identical because the students can then perform the same number of experiments in each session and in the correct order as required by the syllabus. Alternatively, it implies larger investments i.e. more duplicates of each instrument [9].

In electronics, it is possible to perform the same experiment in different time scales by selecting the values of the components controlling the time constants properly. This "feature" is used in the online electronics laboratory at BTH containing only one workbench to allow simultaneous access by time sharing. A single workbench can replace a whole laboratory with many workbenches. The maximum duration of a single experiment i.e. circuit creation and measurement procedure is currently set to 0.1 second to get a reasonable response time even with a large number of experimenters. The experiments are set up locally in each client computer. Only by pressing a *Perform Experiment* button the experimenter sends a message containing a description of the desired circuit and the instrument settings to the workbench (server). If the workbench is not occupied, the experiment procedure is performed in a predefined order, and the result or an error

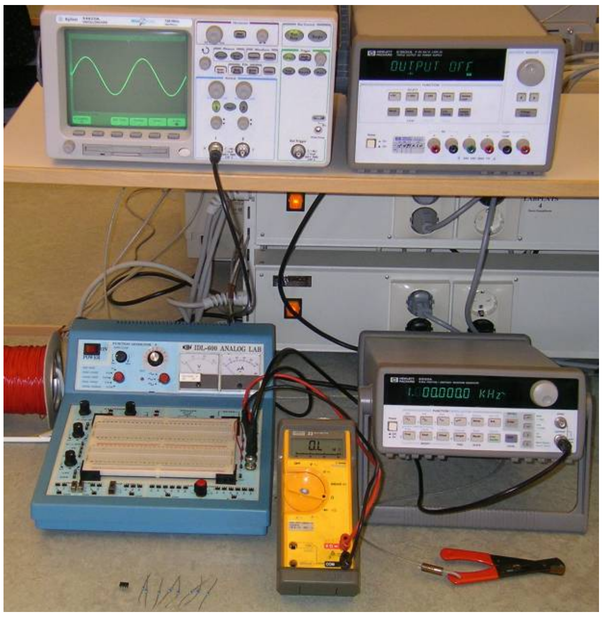


Figure 1. Workbench in a local electronics laboratory at BTH

message is returned to the requesting client computer. Otherwise, the request is queued.

The online lab workbench at BTH is different when compared with the traditional one in Fig.1. It is, of course, not possible for students to manipulate the components and to remotely wire a desired circuit on the breadboard using their fingers. A telemanipulator e.g. a relay switching matrix must be used. The instruments are plug-in boards installed in a PXI chassis connected to a host computer as shown in Fig. 2. This chassis and its contents are manufactured by National Instruments. The corresponding virtual front panels are photographs of the front panels of the instruments in Fig. 1. As an example, a screen-dump, displaying the oscilloscope, is shown in Fig. 3. The card stack on the top of the PXI chassis in Fig. 2 is the switching matrix. A subset of the components a teacher or the laboratory staff has installed in the matrix is displayed on the client computer screen adjacent to a virtual breadboard where the student wires the desired circuit to control the matrix. It is possible to assemble a circuit with up to 16 nodes by engaging a number of relays in the matrix. Apart from a controller board, the card stack contains two types of board: one with component sockets and one for connecting instruments. The nodes passing all boards can be connected to sources, instruments, and/or components installed in the sockets via relay switches. The online electronics laboratory at BTH is used in three ways:

- In supervised lab sessions in the local laboratory where students can select if they want to perform the experiments locally or remotely. However, in the first lab session, it is mandatory to do the wiring on the real breadboard.
- In supervised lab sessions for distance learning courses, where the students are scattered all over the country. Remote desktop software and MS Messenger has been used to communicate between the students themselves and between the students and the instructor. More advanced means of communication will be adopted [14].
- Students can prepare supervised lab sessions and perform the experiments at home, knowing that the equipment in the traditional laboratory looks and behaves in a similar fashion. They can also repeat experiments afterwards! Inexperienced or less confident students requiring more time, appreciate these possibilities. A student wanting, for example, to master the oscilloscope, can practice in the privacy of his/her own home.

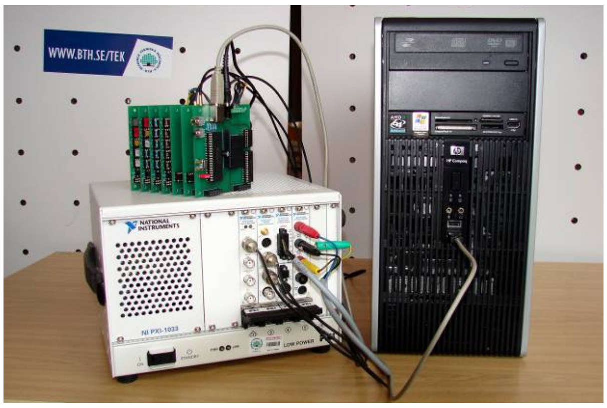


Figure 2. Equipment Server

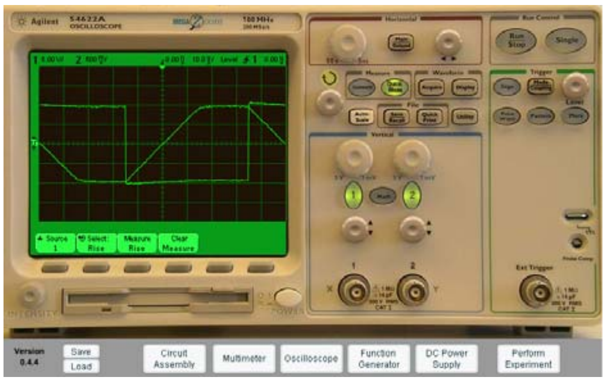


Figure 3. Screen dump showing the oscilloscope

So far, the research has been focused on recreating as accurately as possible, the laboratory experience for a remotely based learner.

III. THE VISIR PROJECT

The aim of the VISIR project is to form a group of cooperating universities and other organizations, a VISIR Consortium, creating/modifying software modules for online laboratories using open source technologies and setting up online lab workbenches [15]. A number of such scattered lab workbenches may be nodes in a grid laboratory. The VISIR Initiative is not confined to electronics laboratories but the VISIR project has started with lab workbenches for electrical experiments, since this is an easy and straightforward application to demonstrate the powerful concept. So far, the following universities are participating, or are interested in participating in the project; FH Campus Wien in Austria, University of Deusto in Spain, University of Genoa in Italy, Princess Sumaya University for Technology in Jordan, Carinthia University of Applied Sciences in Austria, Gunadarma University in Indonesia, UNINOVA (Institute for the Development of New Technologies) in Portugal, and ISEP (Instituto Superior de Engenahria do Porto) in Portugal. The first two universities have already implemented online workbenches using the currently released software. BTH will act as a hub for the development and maintain a server from which the current version of the software can be downloaded.

The overall goal of the VISIR project is aimed at increasing access to experimental equipment in many areas for students, without raising the running cost per student significantly for the universities. The means, are shared online laboratories created by universities in cooperation and supported by instrument vendors. Sharing of laboratories may lead to sharing of course material. The ultimate goal of the research at BTH is ubiquitous physical experimental resources, accessible 24/7 for everyone, gender neutral, as a means of inspiring and encouraging children, young people and others to study engineering and become good professionals or to be used as a means of life-long learning.

IV. A GRID LABORATORY FOR ELECTRICAL EXPERIMENTS

Grid computing has emerged as a way to harness and take advantage of computing resources across geographies and organizations. Grid architecture for an electronics laboratory similar to the BTH one has already been published [16, 17]. In this grid-based laboratory, a measurement workflow execution service takes care of executing the measures according to the rules and sequence described in a measurement workflow repository. It invokes instrument services and manages multi-user concurrent sessions on the same physical test bench. The composition of measurement workflows is in charge to teachers, who provide the description of the measurement process in terms of, for example, instruments activation process. On the other hand, knowing how to handle the measurement process is an important part of lab assignments. To display a transient on the oscilloscope, for example, the oscilloscope must first be armed and then the transient is activated. Each student, or student team, in front of a client computer should have a workbench at their own disposal for exclusive access, as in the local laboratory. Then the *Perform Experiment* button in the BTH laboratory is no longer required.

It should be possible to organize a grid laboratory distributed among universities around the world. The workbenches should be the proper grid nodes. Smaller nodes are not feasible because the instruments and the circuit under test must be located closely together. The instruments and the circuit creation manipulator would be device services accessible by the lab clients via virtual front panels or a virtual breadboard, Fig. 4, 5. Web services prescribe XML-based messages conveyed by Internet protocols such as SOAP. However, real time performance requires protocols without significant latencies and overhead. For example, the oscilloscope display should be updated at least every second.

It is possible to combine a virtual front panel representing a particular instrument from one manufacturer with the corresponding hardware from another, as long as the performance of the hardware matches that of the displayed instrument. The VISIR client software package is modular and it is recommended that every university creates virtual front panels representing the instruments they have in their local laboratories to preserve the student's context.

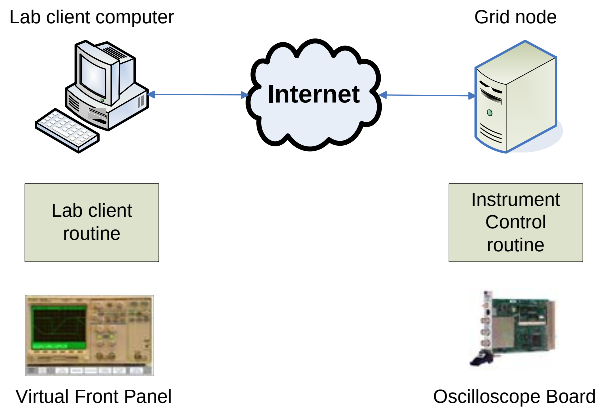


Figure 4. Oscilloscope service

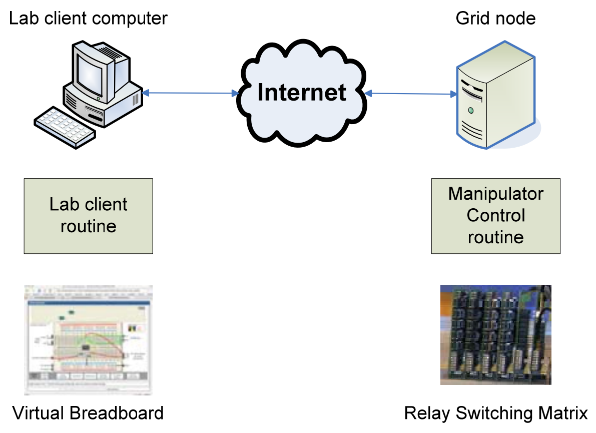


Figure 5. Wring service

Instrument I/O is a well-studied domain with established industrial standards. Most commercial products follow the Virtual Instrument System Architecture (VISA) or the Interchangeable Virtual Instrument (IVI) standards [18]. The IVI foundation creates instrument class specifications. There are currently eight classes, defined as DC power supply, Digital multi-meter (DMM), Function generator, Oscilloscope, Power meter, RF signal generator, Spectrum analyzer, and Switch. Within each class, a base capability group and multiple extension capability groups are defined. Base capabilities are the functions of an instrument class that are common to most of the instruments available in the class. For an oscilloscope, for example, this means edge triggering only. Other triggering methods are defined as extension capabilities. For example, the functions supported by the VISIR oscilloscope are listed in Table 1. The goal of the IVI Foundation is to support 95% of the instruments in a particular class.

It is not necessary to use IVI drivers, but to enable interchangeability between grid nodes VISIR recommends functions and attributes defined by the IVI Foundation to be used to describe the capabilities of the lab hardware. In this way it should be possible to create a standardized

TABLE I.
THE VISIR OSCILLOSCOPE CAPABILITIES

Group Name	Description
`IviScopeBase`	Base Capabilities of the IviScope specification. This group includes the capability to acquire waveforms using edge triggering.
`IviScopeWaveformMeas`	Extension: IviScope with the ability to calculate waveform measurements, such as rise time or frequency.
`IviScopeTriggerModifier`	Extension: IviScope with the ability to modify the behavior of the triggering subsystem in the absence of a expected trigger.
`IviScopeAutoSetup`	Extension: IviScope with the automatic configuration ability.

approach which is easy to adopt.

V. CONCLUSIONS AND FUTURE WORK

BTH is disseminating software for an online workbench, comprising the same equipment as a workbench in a traditional electronics laboratory. The equipment used in the BTH workbenches form robust references for universities who are interested in implementing similar remote or online laboratories. A number of students can perform experiments on such a workbench simultaneously by time sharing. This will be a way for universities to provide free access to experimental equipment for their students in order to produce true experimenters without increased running cost per student. Two universities have already implemented such workbenches using the VISIR software, and are now using them in their own courses. However, each remote student, or student team, should have a workbench at their own disposal to be able to control each step of the measurement process. An approach to reach this more ideal situation would be constituted by increasing the number of online workbenches and organizing them in a grid. Further research is required to accomplish real time performance comparable with that of the local workbench, when a web service approach is to be adopted. The goal is to offer a lab experience that is as genuine as possible, despite the lack of direct contact with the actual lab hardware. Other research groups have developed advanced communication methods, appropriate for a grid laboratory. Such methods will be adopted.

REFERENCES

[1] D. Magin and S. Kanapathipillai, “Engineering Students’ Understanding of the Role of Experimentation”, *European Journal of Engineering Education*, 2000, Vol. 25, no. 4, pp. 351-358.

[2] L. D. Feisel and A. J. Rosa, “The Role of the Laboratory in Undergraduate Engineering Education”, *Journal of Engineering Education*, January 2005, pp 121-130.

[3] Cooper, M., “Remote laboratories in teaching and learning – issues impinging on widespread adoption in science and engineering education”, *International Journal of Online Engineering*, Vol. 1 No. 1, 2005.

[4] Nedic, Z., Machotka, J., and Nafalski, A., “Remote Laboratories Versus Virtual and Real Laboratories”, *Proceedings of the 33rd ASEE/IEEE Frontiers in Education Conference*, Bolder, USA, November 5 – 8, 2003.

[5] J. Ma, and J. V. Nickerson, "Hands-on, simulated, and remote laboratories: A comparative literature review", *ACM Computing Surveys*, 2006.

[6] D. Gillet, A. V. N. Ngoc, and Y. Rekik, “Collaborative Web-Based Experimentation in Flexible Engineering Education”, *IEEE Transactions on Education*, Vol. 48, No. 4, November 2005.

[7] Z. Nedic and J. Machotka, “Remote Laboratory NetLab for Effective Teaching of 1st Year Engineering Students”, *Proceedings of the REV 2007 Conference*, Porto, Portugal, June 25 – 27, 2007.

[8] A. M. Scapolla, A. Bagnasco, D. Ponta, and G. Parodi, “A Modular and Extensible Remote Electronic Laboratory”, *International Journal of Online Engineering*, Vol. 1 No. 1, 2005.

[9] J. Garcia-Zubia et al., ”WebLab-GPIB at the University of Deusto”, *Proceedings of the REV 2007 Conference*, Porto, Portugal, June 25 – 27, 2007.

[10] iLabs: Internet access to real labs - anywhere, anytime, http://icampus.mit.edu/iLabs/, 2007-07-20.

[11] L. Gomes and J. Garcia-Zubia (eds), Chapter 11 in *Advances on remote laboratories and e-learning experiences*, University of Deusto, Bilbao, Spain, 2007, pp. 247 – 267, ISBN 978-84-9830-077-2.

[12] http://www.ivifoundation.org/, 2007-12-15.

[13] I. Gustavsson et al., “An Instructional Electronics Laboratory Opened for Remote Operation and Control", *Proceedings of the ICEE 2006 Conference,* San Juan, Puerto Rico, July 23 - 28, 2006.

[14] MJ. Callaghan, J. Harkin, TM. McGinnity and LP. Maguire, “Paradigms in Remote Experimentation”, *International Journal of Online Engineering*, Vol. 3 No. 4, 2007.

[15] I. Gustavsson et al., “The VISIR project – an Open Source Software Initiative for Distributed Online Laboratories”, *Proceedings of the REV 2007 Conference*, Porto, Portugal, June 25 – 27, 2007.

[16] A. Bagnasco, A. Poggi, A. M. Scapolla, “A Grid-based Architecture for the Composition and the Execution of Remote Interactive Measurements, ”*2nd IEEE International Conference on e-Science and Grid Computing*, Amsterdam, the Netherlands, Dec. 2006.

[17] A. Bagnasco, A. Poggi, A. M. Scapolla, “Computational GRIDSs and Online Laboratories”, *1st International ELeGI Conference on Advanced Technology for Enhanced Learning*, 2005.

[18] Y. Yan, Y. Liang, X. Du, H. Saliah-Hassane, and A. Ghorbani, ”Putting Labs Online with Web Services”, *IT Pro*, March|April 2006, Published by the IEEE Computer Society.

AUTHORS

I. Gustavsson is with Blekinge Institute of Technology, Ronneby, Sweden (e-mail: ingvar.gustavsson@ bth.se).

J. Zackrisson is with Blekinge Institute of Technology, Ronneby, Sweden (e-mail: johan.zackrisson@ bth.se).

K. Nilsson is with Blekinge Institute of Technology, Ronneby, Sweden (e-mail: kristian.nilsson@ bth.se).

J. Garcia-Zubia is with University of Deusto, Bilbao, Spain (email: zubia@eside.deusto.es).

L. Håkansson is with Blekinge Institute of Technology, Ronneby, Sweden (e-mail: lars.hakansson@ bth.se).

I. Claesson is with Blekinge Institute of Technology, Ronneby,Sweden (e-mail: ingvar.claesson@ bth.se).

T. Lagö is with Acticut International AB (owner of Axiom EduTech), Falkenberg, Sweden (e-mail: thomas.lago@axiom-edutech.com).

Manuscript received 29 March 2008. The VISIR project is supported by VINNOVA (Swedish Governmental Agency for Innovation Systems).

Sektionen för teknik

Supported by

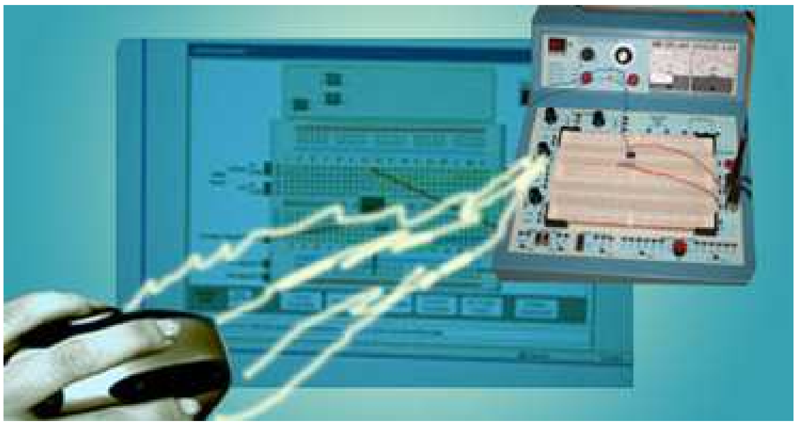
VINNOVA

Virtual Instrument Systems in Reality - VISIR

An Open Source Software Initiative for Distributed Online Laboratories

Background

The VISIR Initiative started at the department of Signal Processing in 1999 as a supplement to local instructional laboratories. The concept is about providing new possibilities for students to do laboratory work and become experimenters by adding a remote operation option to traditional instructional laboratories to make them more accessible for students, irrespective of whether they are on campus or mainly off campus. This option is equipped with a unique interface enabling students to recognize on their own computer screen instruments (oscilloscopes, waveform generators, multi-meters, power supplies, and solderless breadboards) which students have previously used in the local laboratory. This kind of new pedagogical tools supplement experiments in the local laboratory. Students are provided the opportunity to try their skills in new and less stressful environments where they may execute commands as many times as needed for understanding. This option provides the unlimited opportunity for repetition of experiments, training and understanding in remote control labs before an examination. The project focuses on physically real equipment in the lab and has many benefits compared to simulation programs. Students can make real mistakes and come to a certain conclusion by their occurrence while simulators do not allow for such kind of conclusions. Physical experiments are necessary parts in many educational and professional activities, but practical issues of those experiments may create certain restrictions. Many technical institutes in the world are not able to provide the same high level of technical and experimental means for their students. Financial positions are often the most prominent problems. Many steps towards utilizing remote experiments have been made in simulated environments. Blekinge Institute of Technology was among the first to promote the new conception of real world experiments in remotely controlled education, and initiated a big interest worldwide. Now there is an opportunity to practically make use of the open source software and provide full access to experimental resources in a standardized way to students and others which may utilize web-based education.Today the department of Signal Processing has two online laboratories one electronics lab and one signal processing lab for mechanical vibration experiments based.



The focus of the project

The focus of virtual instruments in education is to increase the use of information- and communication-technologies in the education system as well as in the industry. VISIR aims at forming a group of cooperating universities and other organizations, a VISIR Consortium, creating software modules using open source technologies for online laboratories and/or setting up online lab stations. During the recent decades the amount of laboratory work in engineering education, etc. has been reduced. Basically the number of students has increased, while staff and funding resources have diminished. The VISIR project is an initiative to provide means for sustainable development in the education system as well as in the industry. By enabling access to experimental equipment for everyone everywhere as well as providing new tools, makes marketing and installation of new education facilities more effective. The VISIR project creates opportunities enabling the development of competitive actors on a global education market and it is based on an initiative to create open source software to be used for distributed online laboratories.

Since most engineering disciplines needs to provide means for practical experience in hand, the ideal outcome from the project would be a complete standardization of technical remote laboratories. Teachers, or other professional developers, can then develop individual experiments, even where manual interactivity is needed. BTH will act as a hub for the development and maintain a server from which the current version of the software can be downloaded. The server is already in place loaded with the software presently used in the electronics lab and the signal processing lab,

http://svn.openlabs.bth.se/trac.

In the VISIR project these two laboratories will be disseminated, enhanced, and further developed to distributed laboratories where the lab stations will be set up and maintained by a cooperating universities or other organizations. The Consortium will seek EU funding for common online laboratories.

The overall goal is increasing the access to experimental equipment worldwide. The means are shared online lab stations created by universities in cooperation and supported by instrument vendors. Sharing of laboratories may lead to sharing of course material. The ultimate goal of our research at the department of Signal Processing is ubiquitous physical experimental resources accessible 24/7 for everyone as a means of inspiring and encouraging children, young people and others to study engineering and become professionals or to be used as a means of life-long learning.

Blekinge Institute of Technologys conception is that VISIR offers a high degree of variability from a pedagogic point of view and can also be expanded into other scientific spheres, such as mechanical engineering, physic, chemistry etc.

Recommended standards

Standards for instrument drivers are already available. VISIR recommends the standard IVI (Interchangeable Virtual Instrument). The IVI Foundation,

http://www.ivifoundation.org/,

is a group of end-user companies, system integrators, and instrument vendors, working together to define standard instrument programming APIs (Application Programming Interface). The IVI standards define open driver architectures, a set of instrument classes, and shared software components. Hardware platforms such as PXI (PCI eXtensions for Instrumentation),

http://www.pxisa.org/

or LXI (LAN eXtensions for Instrumentation),

http://www.lxistandard.org/home,

are recommended.

The concept will be expanded to laboratories for physical experiments in other areas which are feasible to perform experiments using remote operation.

A VISIR Consortium chaired by BTH will be formed to release new extended future versions.

Project organization and invitation to participate

The partners who have created the VISIR project are Axiom EduTech in Sweden, BTH, and National Instruments in USA. So far FH Campus Wien in Austria and University of Deusto in Spain are participating in the project. University of Genoa in Italy, Carinthia University of Applied Sciences in Austria, Gunadarma University in Indonesia, UNINOVA (Institute for the Development of New Technologies) in Portugal, and University Transilvania of Brasov in Romania are interested to join the consortium. Other universities and organizations are most welcome to participate in the project. Please contact the project leader.

Project leader: Ingvar Gustavsson

VISIR

Axiom EduTech

Postadress: Blekinge Tekniska Högskola, 371 79 Karlskrona
Telefon: 0457 - 38 57 02 | Fax: 0457 - 279 14
Ansvarig för sidan: Ingvar Gustavsson | Sidan ändrad: 26-05-2008