An Overview of the VISIR Open Source Software Distribution 2007

J. Zackrisson, I. Gustavsson, L. Håkansson
Blekinge Institute of Technology/Department of Signal Processing, Ronneby, Sweden

***Abstract*—This paper is intended for people who are interested in creating online laboratories. Blekinge Institute of Technology (BTH) in Sweden has started a project known as VISIR (Virtual Instrument Systems in Reality) together with National Instruments in USA and Axiom EduTech in Sweden to disseminate an online laboratory concept created at BTH using open source technologies in collaboration with other universities and organisations. The VISIR open source distribution 2007 includes software for two implemented examples, one laboratory for low frequency electronics and one for signal processing. The distribution is modular and many modules can be used for other online laboratories. The goal is an international standard, enabling teams worldwide to expand and jointly develop this powerful approach into distributed online laboratories. Each laboratory is a client/server application controlled by an administrative system. These three parts are mainly written in Adobe Flash, C++, and PHP respectively. However, the hardware control module of the equipment server is written in LabVIEW. This paper describes the organisation of the software.**

***Index Terms*—Electronics, online lab, remote lab, signal processing, VISIR project, software development.**

I. INTRODUCTION

Blekinge Institute of Technology (BTH) has started a project known as VISIR (Virtual Instrument Systems in Reality) together with National Instruments in USA and Axiom EduTech in Sweden to disseminate an online laboratory concept [1]. The concept involves adding a remote operation option to traditional instructional laboratories to make them more accessible, regardless of whether the students are on campus or mainly off campus. The BTH solution uses a unique interface enabling students to recognise and be able to operate the equipment found in the local laboratory. Two laboratory environments have been implemented; one for electronics experiments and one for signal processing, with the emphasis on mechanical vibration. The goal of VISIR is to further develop the current software and create distributed online laboratories in collaboration with other universities and organisations. This paper aims to give an overview of the laboratory software system and the interfaces between the components involved. Hopefully it will show the flexibility of the system and how it can be adopted to the suit other remote controlled experiment laboratories. The fact that the experiment environment is released as open source should help making this a collaborative effort.

II. SOFTWARE ORGANIZATION

The software solution is divided in four distinct parts (Fig. 1). A web interface handles administration, user admission and resource scheduling. There is an experiment client through which the users can control the experiments. The measurement server is responsible for handling the experiment requests from the experiment clients. Finally, there is the equipment server, which is a stand alone equipment controller, handling the low level instrument interfaces. Because the system must function in an education environment, external systems such as Learning management systems (LMS) has to be acknowledged and taken into account. This is viewed as an external, but important, part of the system.

A. Web interface

The main purpose of the web interface is the management of resources, keeping track of when and by whom the laboratory is used, to guarantee service and performance.

We use courses as a way to manage students: A course has a start, an end and a limit for the number of registered students. Administrators of the laboratory create it, and therefore they can anticipate usage and allocate the resources that may be needed.

The teachers are registered as responsible for the course and handle all administrative tasks concerning the course, such as registering students and making reservations for the lab sessions. Only students registered to a course can take part of the experiments in the course, but for demonstration purposes, example courses are available for people who want to try the functionality of the laboratory.

One of the basic ideas behind the laboratory is to always be open for experiments if there are adequate resources available. This becomes apparent when staring an experiment. When the user chooses to do so, she/he can start the experiment immediately if capacity is available, i.e. if the laboratory isn't already in full use. To guarantee time for experimentation the student can also make reservations in advance.

1)Starting the experiment

A priority is assigned to the experiment session, depending on a few conditions. Highest priority is given to sessions reserved in advance. Then comes sessions started without a reservation. Lowest priority is given to sessions for demonstration experiments. If the laboratory has available resources no prioritising has to be done, otherwise the experiment sessions with higher priority takes resources from those with lower priority. Thus,

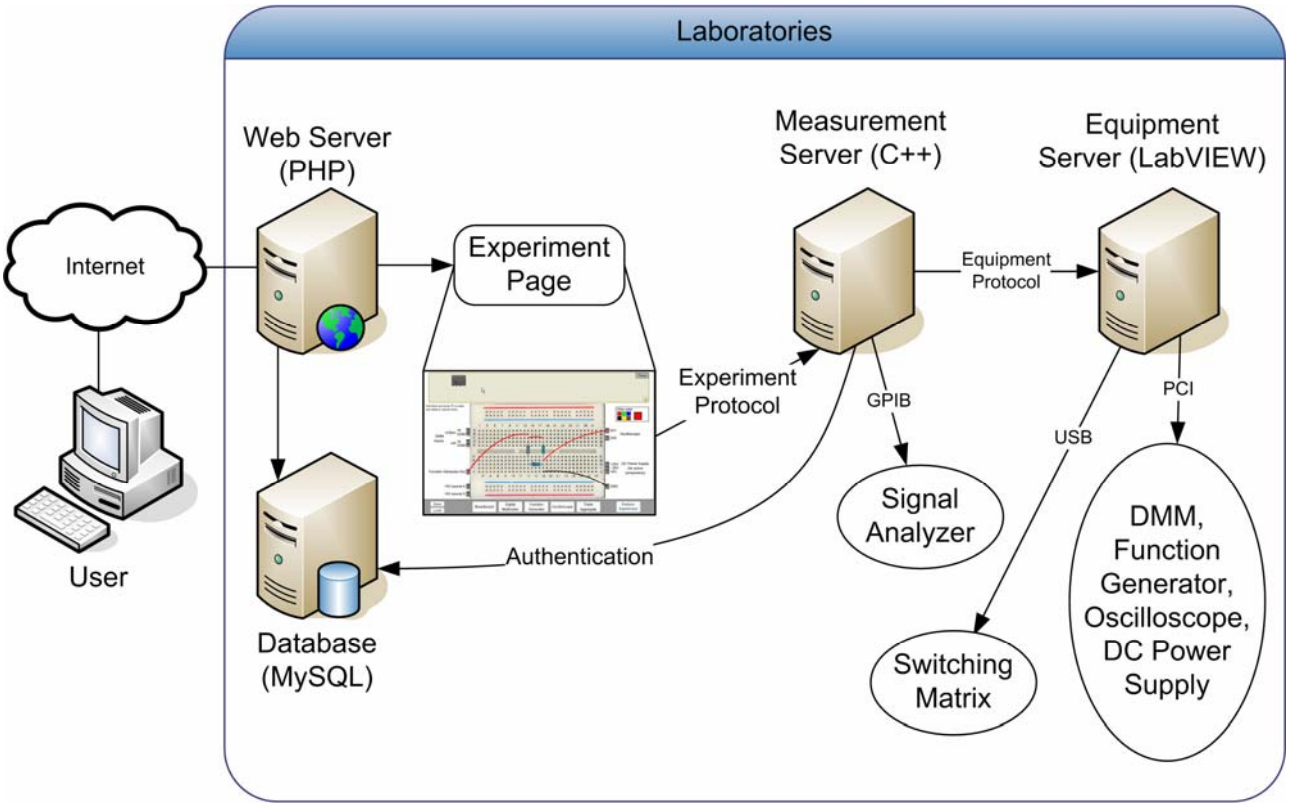

Figure 1. System modules and interfaces.

sessions with low priority are kicked out if the resources are needed by, for example, a supervised lab session.

Information about the started experiment session is stored in a database and is later used to for authentication in the measurement server, better described below.

2)Controlling the experiment

When an experiment session is successfully started, the system presents the means available for the experimentation to the user; the experiment client software. This is done though an HTML page containing the experiment client as an embedded object. Parameters to control the set-up of the experiment can be given to the experiment client, for instance which instruments to display or which components to show by default on the breadboard. Teachers of the course can control these parameters. The web interface itself adds some parameters that can not be controlled e.g. which measurement server to contact. Also, a special cookie parameter identifying the started experiment session is given to the experiment client, this is part of the authentication procedure, described below.

3)Learning Management Systems

Worth noticing is that the web interface does not handle typical Learning Management System (LMS) tasks. Course material and planning etc. has to be handled by an external LMS system. The reason behind this is that every school or university has their own needs and probably runs their own LMS systems already. To make it possible to integrate our web interface into the LMS system, it is possible to use http url:s linking to either courses or experiments.

B. Experiment Client

A life like experiment environment has been developed, containing an assortment of instruments. Each instrument imitates an instrument usually found in a classic laboratory, presenting all the usual functions of the hardware instrument. Photographs of the instruments are used to further improve the perception of the instrument as being real (Fig. 2).

This means that students will get familiar with the classic instruments by using the ones in the remote laboratory. Each instrument is contained in its own module (a separate .swf file), which can be loaded on demand when the client starts. Which instruments to load are decided by the parameters given to the experiment client by the web interface. The modular approach has the advantage of being very flexible when it comes to

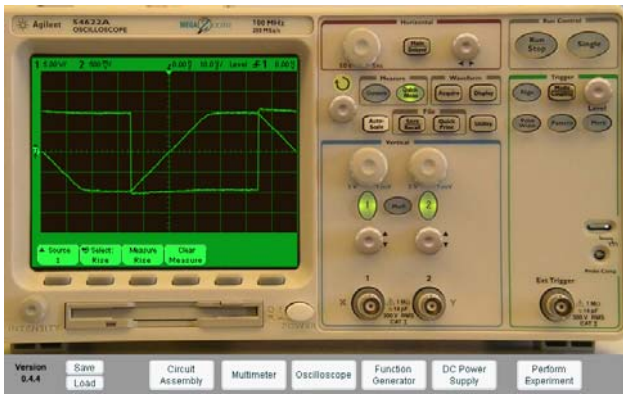

Figure 2. Oscilloscope displayed on the computer screen.

changes. Instruments can change and new ones can be introduced, without the need to change anything other than the instrument module.

Current modules available are:

- A Breadboard for wiring circuits
- Function generator, HP 33120A
- Oscilloscope, Agilent 54622A
- Triple Output DC Power Supple, E3631A
- Digital Multi-meter, Fluke 23
- Signal Analyser, HP 35670A

1)Measurements

Performing measurements and viewing the results is, of course, the most important part of the experiment client. When a user makes a measurement, the experiment client takes the settings from all the instrument modules and compiles them into a measurement request (Fig. 3). This request is subsequently sent to the measurement server, which returns a response that can be read by the instruments to update their displays. The measurement request and response is transmitted using the experiment protocol [2], an XML based protocol describing what settings and functions each instrument type can perform, independent of hardware manufacturer. This makes it possible to implement new modules, for example imitating an instrument that is not available in the current instrument set.

Efforts for expanding the protocol, including new instrument types and/or new instrument modules, can be shared between projects.

C. *Measurement server*

The main responsibility of the measurement server is to serve measurement requests, sent by the experiment clients. These requests are encoded using the experiment protocol, which contains the settings and functions of the equipment used in the experiment.

Before these experiment requests can be served, they are validated; looking for values outside what can be performed or cases that may harm the hardware. The requests then pass the timesharing system and in time the request is handled. This leads to control commands being sent to the equipment hardware and results being read back. The results are then passed back to the client by the experiment protocol.

1)Timesharing

As many clients as possible should be able to use the laboratory at the same time. This is done by queuing all simultaneous requests from the clients and performing them sequentially, using a timesharing scheme. The design goal is to serve requests from 16 simultaneous clients within less than a second. This gives a maximum 1/16 of a second for each request and this has to be taken into account when controlling the hardware.

The maximal number of clients allowed is controllable by a configuration setting. (Because of how it is used, timesharing is not possible when using the signal analyser).

2)Authentication

Because of the limited resources, keeping track of valid clients is important. Verifying the identity of the clients is crucial, for example when keeping track of accounting information. There might be problems concerning clients that do not originate from our web interface that tries to steal laboratory time.

This problem is solved by requiring clients connecting to log-in using an experiment session cookie. This cookie is generated by the web interface and stored in a database. When the measurement server receives this log-in request it validates the cookie against the same database as the

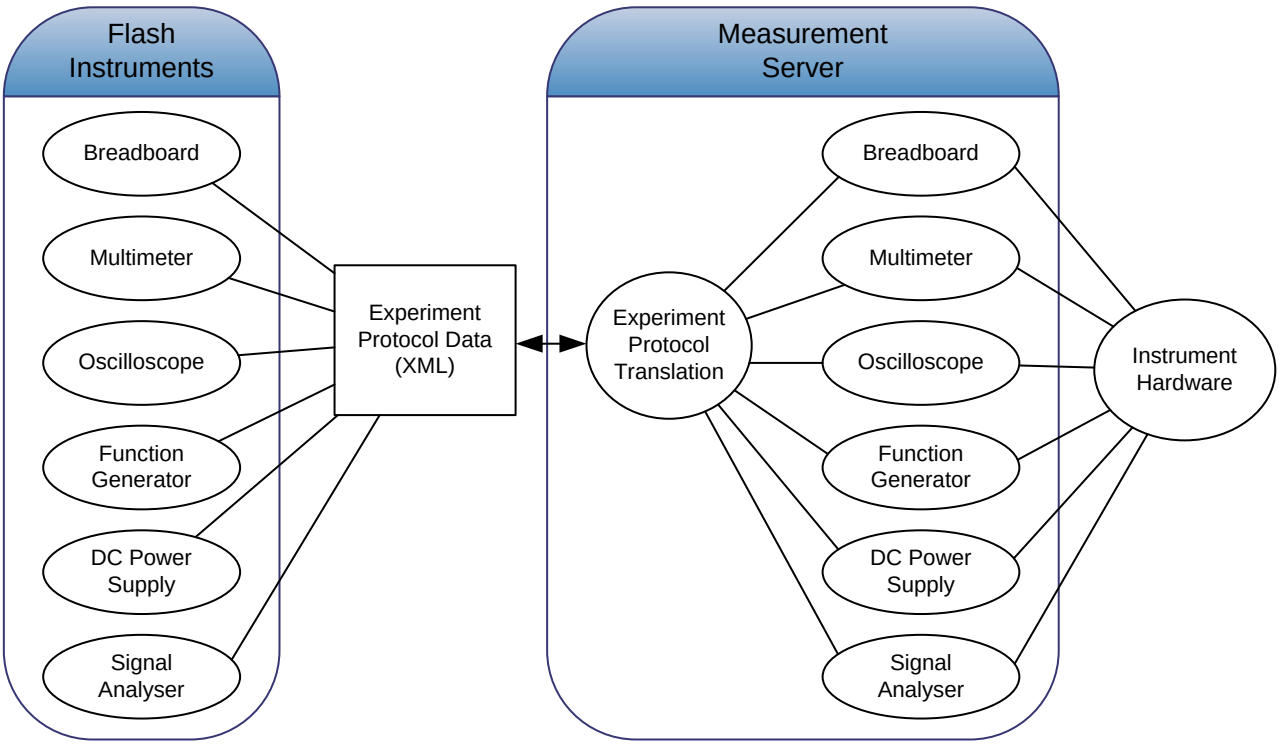

Figure 3. Instrument data transmitted from experiment client to hardware instruments.

web interface uses, and if valid, the client can start experimenting.

3)Experiment protocol

The experiment protocol, previously mentioned, has several advantages that the measurement server can take advantage of. It provides an extra indirection when dealing with low-level hardware instruments, making it possible to provide an interface for controlling equipment independent of manufacturer. Furthermore, because of its design, new instrument types can be added and implemented separately. Thus, making it simpler to expand the kinds of hardware the measurement server can control.

4)Direct or indirect control

Hardware control can be done in two ways. The first is by using directly connected instruments, using a low level interface. This is the case when controlling the spectrum analyser, which uses GPIB to communicate.

Secondly, the measurement server can function as a gateway for other kinds of instrument control, as is the case with equipment servers. Equipment servers handle low-level equipment and instrument interfaces and takes commands sequentially over TCP/IP. This gateway functionality can, if needed, be used for load balancing between multiple equipment servers.

D. Equipment server

The equipment server in Fig. 1 is the server hosting the instrument hardware for electronics experiments, plus a relay switching matrix [1]. This server was earlier acting as a stand-alone online electronics laboratory and it can still be used for applications requiring no administrative service. However, the corresponding old client software supporting the equipment protocol is obsolete and not included in the VISIR Open Source Distribution 2007. If the equipment server is busy executing a request to perform an experiment for a client in stand-alone mode subsequent requests will be queued. The server software is written in LabVIEW and the instrument drivers are IVI (Interchangeable Virtual Instruments) compliant.

III. Deployment

Setting up a new laboratory requires the operator to have good knowledge in the fields where the laboratory software operates. Good knowledge about web applications and administration is crucial for the web interface and experiment client. As for the measurement and equipment servers, experience with instrument control and electronics is needed.

A. Hardware set-up

The modular design of the system gives a substantial freedom when it comes to hardware configuration.

The absolute minimum is a Microsoft Windows machine containing the hardware instruments needed in the laboratory. (PXI or PCI solutions are available). If needed, an equipment server may be needed on this machine to control the instruments. The interface between equipment server and measurement server uses TCP/IP, which enables the measurement server to be run on a separate machine.

Running without the web interface requires that the authentication is disabled (or that you provide your own). Otherwise a machine capable of running a web server supporting PHP and MySQL is required. This could be the same machine again, but the laboratories at BTH run them separately.

It should be possible to run the experiment on any platform supporting Adobe Flash 8 [3].

Communication between experiment client and measurement server is currently done by TCP/IP on port 2324, which has to be connectable through any firewall solution used.

IV. Development

All the software for the laboratory is available as open source, encouraging others to use and to co-operate in the development of the experiment laboratory.

Bug databases and project management information is available on the projects web sites [4][5][6], as well as technical documentation, such as instructions for getting the latest source code from the subversion repositories.

A. Web interface

The web interface is written in PHP against a MySQL database. Some kind of Linux or BSD system is, however, preferable. But the software should be able to run and be developed under other OS such as Windows or Mac OS.

B. Experiment client

The experiment client is written in Adobe Flash Pro 8, but a newer version is already available and will be adopted in a not too distant future. To a great extent the source code is written in external ActionScript files to simplify with the file versioning.

C. Measurement server

The measurement server is written for Microsoft Windows, in C++ using Microsoft Visual C++.

Porting to other operating systems should be a minor effort as the only real dependencies are on the network socket layer. The software is continually developed. An effort has been made to make it as easy as possible to add new instruments and instrument control to the system. The dynamic properties of the XML protocol help both regarding that and for custom adoption if laboratories have special needs. Hopefully others will adopt the same protocol for their instrument control needs, either by using our software or by developing their own.

V. Future work

Because of the experiment protocol, the possibilities exist of using other types of clients beside our experiment client. This can allow other systems, for example web-based scripts, to access the laboratory and display the results as plain web pages. Other types of interfaces, such as web services could be adopted, either as a standalone service accessing the measurement server or as a module in the measurement server.

References

[1] 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.

[2] Experiment protocol, http://svn.openlabs.bth.se/trac/measureserver/wiki/ClientProtocol, 2007.

[3] Adobe Flash system requirements, http://www.adobe.com/products/flashplayer/productinfo/systemreqs/, 2007.

[4] Web interface project page, http://svn.openlabs.bth.se/trac/openlabsweb, 2007.

[5] Experiment client project page, http://svn.openlabs.bth.se/trac/flash, 2007.

[6] Measurement server project page, http://svn.openlabs.bth.se/trac/measureserver, 2007.

AUTHORS

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

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

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

This work was supported by National Instruments, Axiom EduTech, BTH, and VINNOVA (Swedish Governmental Agency for Innovation Systems).