Web – based Virtual Instruments for Remote Data Acquisition

Vasil Janov Vasilev, Vassiliy Platonovitch Tchoumatchenko
Department of Electronics, Technical University of Sofia
1756 Sofia, Bulgaria, vvasilev@lark.vmei.acad.bg

A. Bagnasco
Biophphysical and Electronic Engeneering Department, University Of Genoa
Via'All Opera Pialla – 16145 Genoa – Itally, bagnasco@dibe.unige.it

Abstract. The paper describes a software environment for distributed instrumentation ISILab. The focus of this presentation is on the client-side technologies. Basic principles and structure of the system is considered. Then TCP/IP based protocol for data interchange is discussed. Two example applets are included: virtual multimeter and virtual oscilloscope.

1. Introduction

The possibility of controlling instrumentation by using a personal computer and the wide diffusion of computer networks have led to the development of software environments where the physical presence of the experimenter in the laboratory is not required any more. Virtual interfaces to instrumentation and computers networks provide the link between users and the laboratory equipment, allowing a large learning community to share practical activities. The experimental set up can be distributed in different real laboratories, spread on a wide area network, and controlled by local computers. Users can carry out experiments through the network and practice transparently to real location and appearance of involved hardware.
Researches carried-on at University of Genoa during the latest years, has led to the definition of a model [1] to share laboratories in Internet, and to the implementation of a prototype (Internet Shared Instrumentation Laboratory – ISILab – pronunciation: easy-lab). An innovative aspect of ISILab [2] is its modular and scalable structure, so that different developers can contribute to increase the number of experiments. The remote laboratory has to be regularly updated in order to meet students and teacher needs: new experiments are added and new instruments controlled. The insertion of a new experiment implies to build a certain number of objects, such as web pages for the experiment explanation, graphical user's interface for instrument control, instrument's drivers, etc. The success and the effectiveness of the environment are strongly conditioned by the way we can add new components and reuse existing.

This paper focuses on the graphical user interfaces (GUI) fitting the ISILab model. It is developed in co-operation between the University of Genoa and the Technical University of Sofia within the framework of the NetPro II EC Leonardo da Vinci Project.

2. ISILab: an overview

ISILab is based on a distributed software environment (see Fig. 1) consisting of a main Virtual Laboratory Server (VLS), one or more Real Laboratory Servers (RLS) and user/client stations. Internet links all these components.

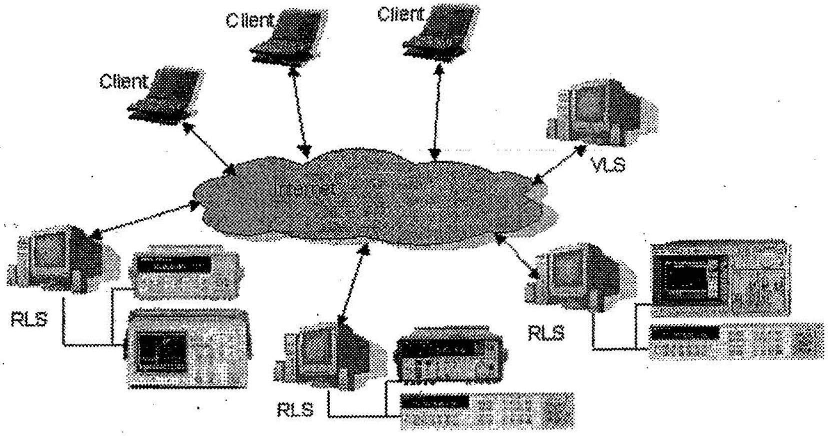

Figure 1: The system architecture.

The VLS is a network node hosting a web server, which delivers web pages introducing users to the laboratory. Also, VLS applies the access control policy, and logs users' activities. The access is restricted on the basis of login/password credentials and only authorized users can get into the laboratory. RLSs manage the interactions between users and experimental set-up. When a user asks for executing an experiment, VLS initializes the communication between the client and the RLS, by allocating available resources, and delivers user's interfaces. These user's interfaces are Java applets that run on client and establish a direct TCP connection with the RLS.

RLSs run a specific server-application (written in LabVIEW from National Instruments [3]) that receives the inputs from users via TCP, applies them to the instruments, which are connected via local bus (i.e., IEEE488), retrieves the results and sends them back to users.

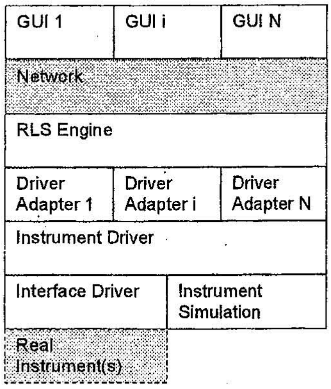

Figure 2: RLS's software architecture

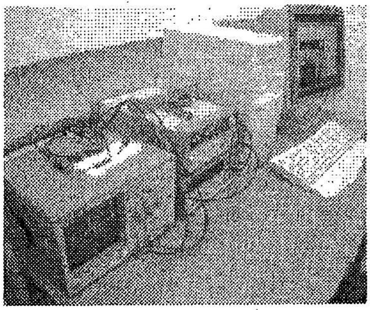

Figure 3: RLS photograph.

Figure 2 shows the software architecture of the RLS. The Engine takes care of separating the different user data spaces (Contexts), in order to manage concurrent access to instruments. The Context is the data structure that caches information about the current configuration of each instrument involved in a specific experiment, carried out by a specific user. When a user sends a command (by acting on the virtual instrument panel), his/her Context is updated and applied to the appropriate instrumentation set. Instruments' responses are recorded in the Context and changes are sent back to the user. A Context identifier is assigned to each user when an experiment is selected; this assures the coherency among context data, virtual instrumentation and user identification during the experiment execution. In order to communicate with the RLS, applets must send an identifier that contains both the Context and the specific instrument IDs. Driver Adapters convert the user controls (knobs, switches and so on) to Instrument Driver inputs/outputs. So, they are software modules that manage the communication between the GUI and the instrument driver. The status of the instrument, set by the user, is elaborated and translated to a specific call to the instrument driver.

Instrument drivers are general-purpose drivers that are usually provided by vendors. They transform numeric values into instruments' specific commands. Instrument drivers should be able to exchange data with both real and simulated instrumentation.

This layered model facilitates the development of different GUIs and adapters for the same instrument. In this way, using ISILab approach, we can control the same instrument using different type of GUI. On Figure 3 a photograph of real RLS is presented.

3. Web-based client front-end

In ISILab the remote laboratory experiences have to be carried out via the most popular web browsers. HTML pages, simple and portable carrier of information, are used for introducing and explaining experiments. Java applets are the natural choice for instrument control, because of the flexibility in design, facilities in network programming and platform independence [4].

Often, actual instruments have very hostile control panels. This is because they have a small amount of controls (switches, knobs and so on), which are used to set a large number of parameters. Since software layers make the user interface independent from the controlled hardware, with ISILab we are free to design panels with every shape we want. In this way, we can develop task-oriented applets, able to show to the user only the setting that are strictly necessary to execute the assigned task. This kind of interface takes advantages from the rationality of graphical objects like menus, forms, list boxes, etc. This is very useful for implementing didactical panel, oriented to generic concepts teaching, because it lets to override the complexity of the actual control panel. On the other hand, if we want to teach exactly how the particular instrument works, we need a detailed, high interactive, and realistic soft-copy of the real control panel. Figure 4 shows examples of both task-oriented and realistic interfaces.

Task-oriented applets for ISILab have been developed using AppletVIEW™ from Nacimiento™ [5]. This is a commercial framework for developing instrument control panels. It offers a graphical editor that automatically generated applets without requiring Java programming, and a set of library that makes these applets able to communicate with LabVIEW™ using a TCP-based protocol called VITP (Virtual Instrument Transfer Protocol). AppletVIEW™ is very useful for realizing simple applet in a very short time, but it is not practicable where we need more realism. In this case the applet becomes a complex application and we need to face its development in different way.

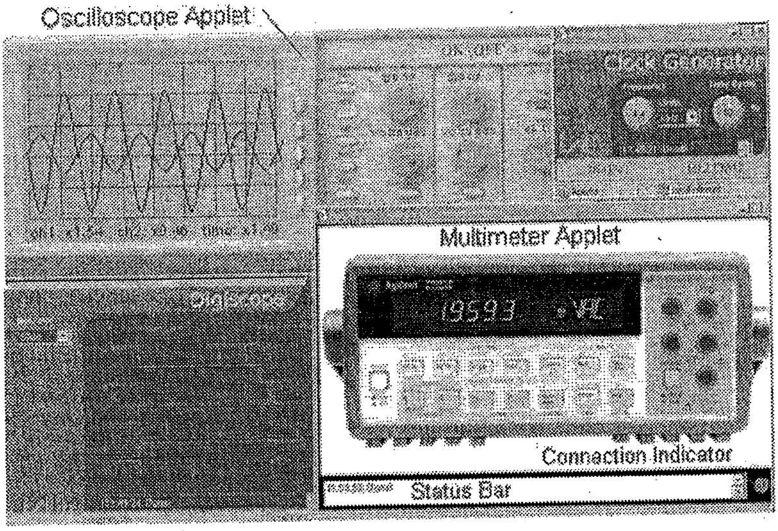

Figure 4: The web-based environment.

4. Applets for realistic device functionality

Two applets that provide realistic device functionality are developed. One is virtual multimeter ant the other is virtual oscilloscope. These applets and all other that can be developed use VITP protocol to communicate through the network. For this reason special Application Programming Interface (API) for Java is created which to implement VITP over TCP/IP connection. The architecture of this API is shown on Figure 5.

In central position here is the LVCConnector class, which provides all needed methods for sending and receiving data of all types. It generates events when data are received and propagates them to all registered listeners (classes that implement DataReceivedListener. These are usually applets). There is one additional class named DataTransformer which is responsible for converting data types from VITP to Java and vice-versa.

On Figure 4 a screenshot from the multimeter applet is provided at front of the other applications. The status bar in the bottom of the applet informs about the status of the connection. The little circle right to the status area can have three colors: yellow - means waiting for connection to the server; red - connection broken; green - connection established.

The Graphical User Interface (GUI) of the applet provides all necessary number of elements to control multimeter's behavior (power on/off button, measurement type selection buttons, range and resolution selectors and a display).

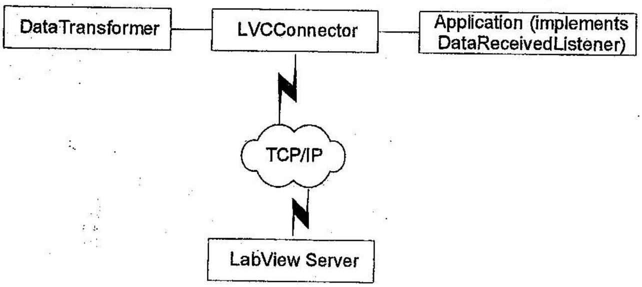

Figure 5: VITP implementation API

The oscilloscope front-end applet provides all needed elements to control two-channel oscilloscope. This includes channel 1 and channel 2 coupling and resolution and time resolution. X and Y axis offsets can also be set.

5. Conclusions

The developed system frees the experimenter from physical presence in the laboratory. It also allows multiple users all over the world to participate in common experiment with device equipment not necessarily situated in one laboratory.

By using Java Applet technology, specific task-oriented GUI can be created to focus attention of the experimenter only on the information user needs for the experiment. Graphical front-ends that reproduce the full functionality of the real device can also be created.

References

[1] A. Bagnasco, M. Chirico, G. Parodi, A. Sappia, A.M. Scapolla, "A Virtual Laboratory for Remote Electronic Engineering Education", in International Perspective on Tele-education and Tele-learning, Ashgate Book, 2000, pp. 1-14.
[2] A. Bagnasco, M. Chirico, A.M. Scapolla, "XML Technologies to Design Didactical Distributed Measurement Laboratories", IMTC2002 – IEEE Instrumentation and Measurement Technology Conference, Anchorage, AK,USA, May 2002
[3] www.ni.com
[4] Chung Ko, Chi et Al., "A Web-Based Virtual Laboratory on a Frequency Modulation Experiment", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 31, No. 3, August 2001.
[5] Travis, Jeffrey, "Internet Application in LabVIEW", Prentice Hall, 2000