Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Real-Time Control with Linux: A Web Services Approach

Michele Basso, Roberto Bucher, Marco Romagnoli and Massimo Vassalli

Abstract— **The aim of the paper is to present an implementation of the already mature web services technology to the real-time world based on the project Linux RTAI which includes advanced tools for the integration of Computer Aided Control System Design environments. Within this context, RTAI-XML is a new server component executing within a RTAI domain that works as a bridge between a remote client application and a real-time process. The server exposes its methods and properties to the Internet using XML (eXtensible Markup Language) and a remote procedure call framework. The application contexts of the new platform range from control education to industrial applications.**

I. INTRODUCTION

In designing a Real-Time oriented Human-Machine Interface (HMI) application, it is natural to approach separately Hard (HRT) and Soft (SRT) real-time components, the former being more closely related to hardware communication and control algorithms implementation, the latter being associated with user interface design and data analysis and manipulation. The strong separation between these two aspects of an RT-oriented application is also clearer if we compare the requirements the programmers take into account during the design phase: whereas working times and jitter are ever in mind of the RT process programmer, the GUI (Graphical User Interface) developer needs flexibility and friendliness. This duality is intrinsic to the whole process of building an RT-oriented application, so that it is substantially impossible to choose development tools and strategies reliable both for hard and soft real-time domains.

The duality problem presented for RT-oriented application design is only one side of the more general scenario where an office application developer has to deal both with business logic and presentation graphics. In the last years this problem has become more and more actual, mainly due to the development of web technologies, facing not only the logic difference between the two aspects of an application, but also the physical separation, in the server - client architecture. After a first set of trials, oriented to patch the problem taking into account the specificity of a web application (for example a web portal), the Internet community drastically changed its point of view, moving coherently toward a new approach, where the definition of standard object access protocols over a network allows two remote pieces of code to interact properly, sharing only the communication language but none of the implementation details. That is the world of web services [1].

M. Basso and M. Romagnoli are with Dipartimento di Sistemi e Informatica, Università di Firenze (Italy) {basso,romagnoli}@dsi.unifi.it

R. Bucher is with the Department of Innovative Technologies, University of Applied Sciences of Southern Switzerland, CH-6928 Lugano-Manno, Switzerland roberto.bucher@supsi.ch

M. Vassalli is with Istituto Nazionale di Ottica Applicata, Firenze (Italy) vassalli@inoa.it

The aim of this paper is to present an implementation of the already mature web services technology to the real-time world based on the open-source project Linux RTAI (Real-Time Application Interface) [2]. Besides being a fully featured RT operating system, RTAI includes advanced tools for the integration of Computer Aided Control System Design (CACSD) environments and data acquisition systems, providing a flexible platform for rapid control prototyping. Within this framework, we have started the project RTAI-XML [3]. RTAI-XML is a server component executing within a RT domain that works as a bridge between one or more RT processes (Targets) and the remote procedure call framework provided by standard protocols like, for example, SOAP (Simple Object Access Protocol)[4]. Once the Target is able to expose its methods and properties to the Internet using XML, the RT programmer has no longer to do with presentation needs. In the same way, the user interface designer can use whatever application development tool he/she prefers, creating executables for a specific platform or even multi-platform, forgetting problems related to the criticality of an RT control and following only presentation and functionality requirements.

The application contexts of the new platform range from control education, where the birth of remote control labs suitably fits in a web services framework, to industrial applications, where a strong separation of the real-time core from the GUI development is always required.

The paper is organized as follows: in Section II the main features of RTAI and its integration with CACSD environments are described; in Section III the basic architecture of RTAI-XML is shown from a web services perspective; in Section IV various applications exploiting the RTAI-XML paradigm are presented. Finally, in Section V main results are briefly discussed.

II. RAPID CONTROL PROTOTYPING AND LINUX RTAI

Rapid Controller Prototyping (RCP) requires two components: a CACSD software and a dedicated hardware with a hard real-time operating system. The software should allow to perform all phases of control system design (specification, modelling and identification, control design, simulation, implementation and verification) in the same environment. A graphical application allows the user to create models for dynamic systems simply by connecting blocks from available libraries. Among others, some blocks implement linear systems given as transfer functions or state-space realizations both in continuous and discrete time. An integrated code generator allows to directly create code from the graphical scheme. The COMEDI project [5] provides the drivers to interface the system with the real plants.

0-7803-9568-9/05/$20.00 ©2005 IEEE

A. Linux RTAI

The proposed solution is based on Linux RTAI, a hard real-time extension of the Linux Operating System developed at the Dipartimento di Ingegneria Aerospaziale del Politecnico di Milano (DIAPM) [2].

Paolo Mantegazza started the RTAI (Real Time Application Interface) project in year 1999. Since version 3.1, Linux RTAI is based on Philippe Gerum's Adeos (Adaptive Domain Environment for Operating Systems) nanokernel [6]. Adeos provides a simple layer that is inserted between the hardware and a running OS (called *domain*) and thereafter provides the primitives and the mechanisms to allow multiple OSes to share the same hardware environment. Adeos uses an interrupt pipe to propagate interrupts through the different domains running on the hardware. One of this domain is the normal Linux OS. When the real-time modules of RTAI are loaded, a new domain *RTAI* is registered and inserted in the pipe by considering its priority. A more detailed description of Adeos is given in [7]. The RTAI extension was created as an environment for implementing low cost data acquisition and digital controller systems. Linux RTAI provides a tools called RTAI-Lab [8], which contains a set of utilities for the integration of two CACSD software, the commercial Matlab/Simulink/RTW and the open source Scilab/Scicos suites, into the Linux RTAI environment.

B. Matlab/Simulink/RTW

The Matlab/Simulink/RTW suite [9] is a commercial product widespread in universities and industries. Near to an accurate graphical interface it gives the possibility to create C code, using the Real Time Workshop Toolbox. The generated code can be easily adapted and downloaded to different targets.

The core of the implementation for the Linux RTAI environment is represented by the target files "rtai.tlc" and "rtai.tmf" and by the different C-MEX S-Functions that implement the data transfer to the RTAI-Lab environment and to the drivers of the DAQ boards (COMEDI or user specific). Fig. 1 shows the current Simulink RTAI Library.

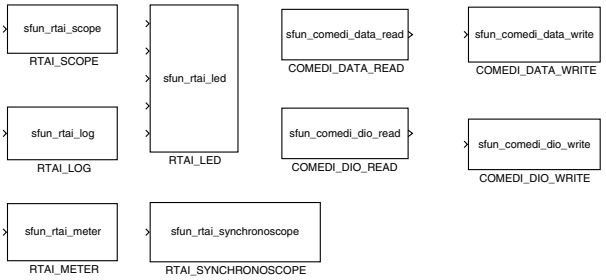

Fig. 1. Simulink RTAI library.

The Matlab/Simulink/RTW suite allows to generate code from a graphical scheme which may contain continuous blocks. In this case a numerical integration algorithm is added to the generated code. Multiple sampling rates can be handled by starting a real-time thread for each sampling time.

The main disadvantage of this suite is given by its cost (this solution requires licenses for Matlab, Simulink, Control System Toolbox and Real-time Workshop). Furthermore the last release of the suite needs a high amount of system resources.

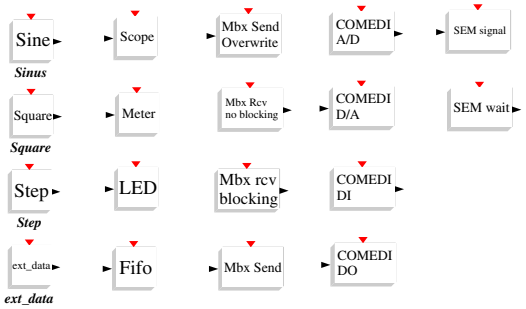

Fig. 2. Scicos RTAI library.

C. Scilab/Scicos

Scilab [10] is a scientific software package for numerical computations providing a large set of functions for engineering and scientific applications. It has been developed since 1990 by researchers from INRIA (Institut National de Recherche on Informatique et on Automatique, [11]) and ENPC (École Nationale des ponts et chaussée, [12]) and it can be freely downloaded from the Internet. Scilab is currently used in educational and industrial environments around the world.

Scilab already contains all the functions and toolboxes needed for control applications. A tool called "Scicos" allows the implementation of block diagrams in a graphical mode. The simulation of the designed scheme can be performed directly from the Scicos window.

The RTAI environment has been extended with the new code generator *RTAICodegen_sci*. This is a modified version of the *CodeGeneration_sci* file provided by Scicos.

A new Scicos block library specific for the RTAI environment has been implemented. Fig. 2 shows the blocks contained in this library.

The RTAI library *libsciblk.a* contains the code of the Scicos blocks which need specific RTAI resources.

The code generator can only work with discrete time blocks. Multiple sampling rates must be implemented using distributed controllers. Asynchronous systems can be realized using semaphores or blocking mailboxes.

The main advantage of the Scilab/Scicos suite is given by the fact that all the software can be freely downloaded from the web.

D. Code generation

A simple example shows the three distinct steps needed to create a stand-alone executable for the controller.

1. Model definition: The designer creates an appropriate Simulink or Scicos model using blocks from the standard built-in libraries (see Fig. 3 top). The plant is implemented as mathematical model including the non linearities of the system (Fig. 4 top). Using this scheme the designer can implement and simulate different control strategies and algorithms.

2. Code generation: The second step involves the C-code generation. The mathematical plant is now substituted by specific I/O blocks in order to implement the interfaces to the acquisition boards (see Fig. 4 bottom). Scopes must be substituted with the ones provided by the RTAI-Lab libraries (Fig. 3 top). In Matlab/Simulink the code generation is performed by the Real Time Workshop toolbox (RTW). In Scilab/Scicos the user must create a superblock first, and then he can start the code generation by calling the new provided function "RTAICodeGen_sci".

3. Code execution: Finally, the stand-alone executable can be directly started or transferred to any target machine running the same version of RTAI used for compilation.

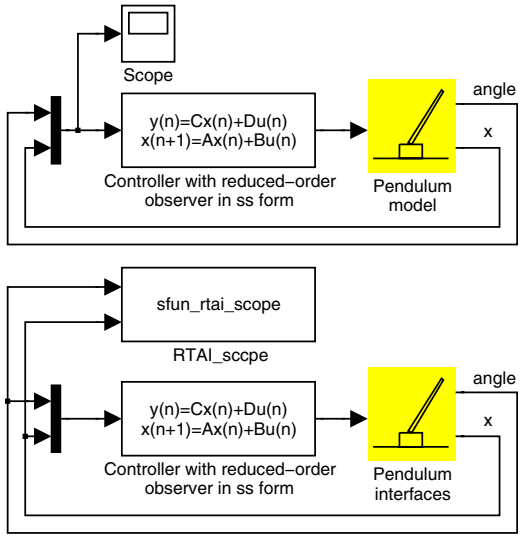

Fig. 3. Scheme for simulation (top) and code generation (bottom).

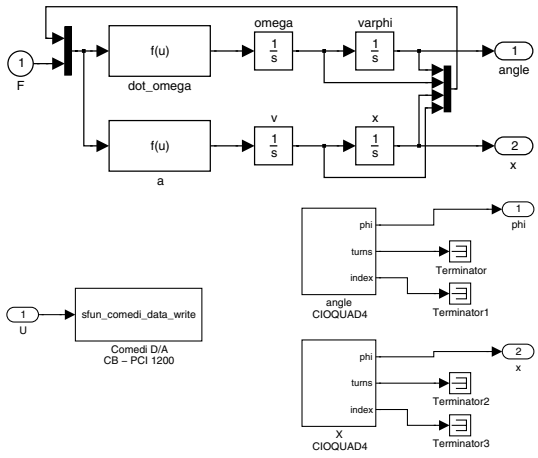

Fig. 4. Pendulum block for the simulation (top) and code generation (bottom).

This procedure is similar in both Matlab/Simulink/RTW and Scilab/Scicos environments. Both systems contain a main file called *rtmain.c* which starts the real-time thread for the control task. The main file starts also a communication thread between the real-time task and an external monitoring GUI application.

E. Distributed control

The same mechanism used to communicate between the real-time task and the RTAI-Lab GUI can be used to establish a communication between different hard real-time tasks. This makes it possible to create a distributed control system within a single PC or in a LAN, using the net_rpc layer and the rtnet hard real-time network driver [13].

Data can be exchanged using the RTAI *mailbox* structure. This allows also to integrate real-time tasks generated by Matlab/Simulink with tasks generated by Scilab/Scicos.

III. RTAI-XML ARCHITECTURE

RTAI-XML provides a remoting approach to RT control applications design, based on XML abstraction layer. The aim of RTAI-XML is thus to provide a service as depicted in Fig. 5, where the real-time control process (the Target) and the user interaction related procedures (the Host) run independently, typically on two different CPUs, communicating over the network (Internet). The main difficulties in facing a real-time application are visible immediately. On one hand there is the need to work in a real-time framework with stringent time constraints, on the other hand the user has to interact with the application through a human friendly interface. These two requirements are – in principle – conflicting.

The solution provided by the RTAI-XML server defines an operation domain called RTAI-XML domain, as shown in Fig. 6. Three different objects related to their position inside the RTAI-XML domain can be defined:

1) the *real-time targets* are situated inside the domain and work in HRT. They can be generated using RCP techniques based on RTAI-Lab, or implemented directly using the API (Application Programming Interface) of RTAI. In both cases the rigorous RT communication interfaces with the RTAI-XML server must be observed.
2) the *RTAI-XML server* is located on the real-time domain border. The server manages the available targets, connecting and disconnecting them from the domain, obtaining the structure and changing, if required, their parameters in real-time. Overall, it supervises the communication with the RT processes. On the other side, it provides an external interface that allows one to interact with the different targets inside the domain. Such an interface is based on XML over HTTP and is structured as a web service providing information about the RTAI-XML domain, the state of every registered target, and a set of remote procedure calls that are used to control and monitor the target execution.
3) the *external applications* are clients of the RTAI-XML server services. Such applications are not developed in a real-time framework and can be implemented in any programming languages. They communicate with the RTAI-XML server using XML remote procedure calls as those detailed in Table I.

A framework developed according to this logic shows the necessary flexibility to adapt to many application contexts, as shown in Section IV. The flexibility of the RTAI-XML domain is ensured both at the programming level and the interface level. At the programming level the RTAI-XML server can communicate with targets that have been specifically written using the RTAI-XML APIs, or automatically

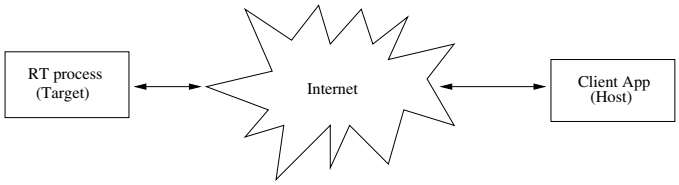

Fig. 5. Basic client-server architecture.

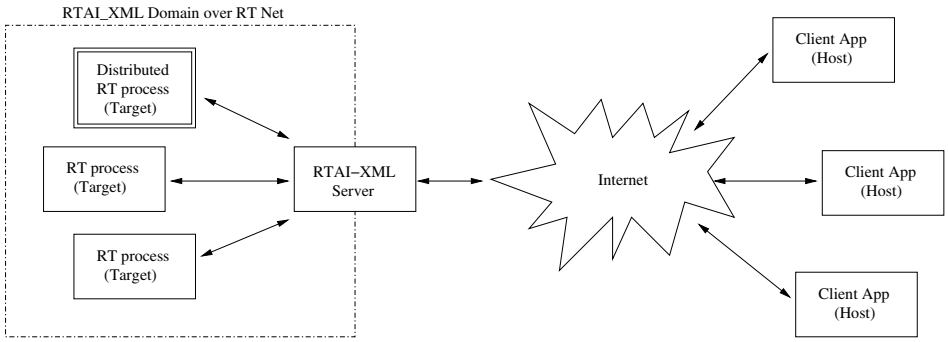

Fig. 6. Advanced client-server architecture.

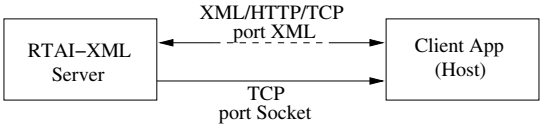

Fig. 7. Details of the client-server interaction.

generated by RCP tools (see Section II). At the interface level it allows the definition of a communication protocol between the target and the RTAI-XML server. On the external side the basic features of a web services approach are provided. Such an architecture is completely independent from the hardware outside the RTAI-XML domain. At the same time it allows to communicate and interact between generic external applications and the RTAI-XML domain.

In this framework one can find a high interoperability among the various components of the global system, and the so-called "making data available from the plant floor to the top floor", satisfying at best the different requirements at several organization levels: "at the plant floor" the possibility to design distributed applications to control and monitor RT processes with the power and flexibility of a Linux RTAI real-time domain. At this level every source of troubles in dealing with an intuitive data presentation is removed. Instead, "at the top floor" this approach allows the developers to focus on data processing without any complication due to the real-time domain.

Since the introduction of web services there have been conflicting positions concerning advantages and disadvantages of this approach to real-time applications [14]. The main advantage can be summarized in the increase of interoperability among different platforms, removing the restrictions due to the hardware limits. The disadvantage is generally concerned with the overhead present in data transmission. The interface provided by the RTAI-XML server is based on XMLRPC [15] that uses XML to represent data and HTTP/TCP (Transmission Control Protocol) as communication protocol (see Fig. 7). The overhead due to XML and HTTP is sent over the network increasing the required bandwidth without increasing the delivered information. For this reason, web services are often not suited to communicate the amount of data an application might require in a real-time domain.

RTAI-XML tackles the latter problem exploiting two separate communication levels as depicted in Fig. 7. At the first level only web services are employed, allowing remote clients to communicate via XML/HTTP/TCP. When the RTAI-XML server needs to answer a client request, it provides two alternatives: if the request involves general information about the domain state or, for instance, retrieving only the signal and/or parameter structure for a given target, then the response uses the same request technology via XML/HTTP/TCP. Conversely, if a high data rate is involved, then the RTAI-XML server switches to the next communication level. The latter is designed exploiting a specific protocol directly over TCP, such that the overhead is reduced and the transmission is optimized for the available bandwidth.

IV. APPLICATIONS

A. Generic Client Application: JRTAILAB

JRTAILAB is a generic application implemented in the Java programming language. It represents a simple implementation of a client supporting the communication with RTAIXML. This client shows the possibility of interfacing with RT processes without any direct real-time implication. The developed software is multi-platform and can be executed on any combination of operating systems and hardware supported by JAM (Java Virtual Machine). The main goal of this application is to control a target registered in the RTAIXML domain. It allows to establish a connection to the RT target, choosing the signals to observe, retrieving and

TABLE I
RTAI-XML EXTERNAL INTERFACE.

XMLRPC Server Method	Description
Info	Retrieve the execution state of a target in the RT domain.
Connect	Insert a target in the RT domain.
Disconnect	Remove a target from the RT domain.
Start	Start the execution of a target.
Stop	Stop the execution of a target.
Get_Param	Retrieve the target parameters.
Send_Param	Update the target parameters.
Get_Signal_Structure	Retrieve the signal structure of a target.
Start_Data	Select a signal to send data over the TCP communication channel.
Stop_Data	Stop a signal transmission.

updating the target parameters and, in general, supervising the target execution (see Fig. 8 for a screenshot).

B. Remote Control Lab: ARTIST

The fast development of web-based technologies has led to the new concept of remote or virtual labs, i.e. laboratory facilities that can be accessed at any time from any place through an internet connection, providing the user of an effective way to perform physical experiments remotely, while observing the results on his/her computer, locally. Interesting examples of this approach can be found in [16] and references therein. In this framework, ARTIST [17] has been designed as a prototype of a remote and shared control lab based on RTAI-XML, providing the following features:

1) The designed controller (either a Simulink or a Scicos file) is automatically converted as a real-time executable and uploaded on a remote PC connected to one or more processes, guaranteeing hard real-time performance.
2) The experiments are controlled by a standard web browser exploiting RTAI-XML calls, allowing a full modification of controller parameters and observing signals on real-time plots.
3) For each experiment, a supervisor can be employed in order to let users deal with unstable plants and/or to prevent them to force the system with wrong control actions.
4) Since a centralized server is employed to handle the remote PC's, experiments can be easily shared among different laboratories or even different departments/universities, so reducing costs and facilitating the birth of laboratory consortia.

C. Industrial Application: AFM

Atomic Force Microscopy (AFM) is a measuring technique in which an elastic micrometric sized cantilever, with a sharp tip on the top, is scanned over the sample, allowing to image the surface with nanometric resolution. Based on this simple working principle, reminding of the macroscopic mechanical profilometers or feeler pins, AFM is a really flexible tool, able to image samples ranging from nanostructured materials (nanotechnology) to living cells (biotechnology). Switching from one context to another is clearly a matter of designing the proper measuring head, but also it is fundamental to optimize the control logic and to develop a dedicated software interface to the machine.

In order to obtain a flexible AFM system, implementing different working modes inside a unique framework, we adopted the RTAI-XML paradigm. The standard AFM controller was substituted by a Personal Computer, powered by Linux-RTAI, interfaced with the mechanical AFM head using AD/DA boards (PCI 6036E from National Instruments [18]) dedicated to acquire the required signals (e.g. the deflection of the cantilever) and to drive the piezo actuation of the scanning (through an external HV amplifier). The main task of the controller is to implement a feedback ring conditioning the behavior of the tip during the surface scanning. Using the RTAI-Lab approach we were able to design and test different control logics, from simple PI algorithms to model based H_∞ controllers, simply rebuilding a Simulink scheme, without changing any of the hardware implementation details. In an AFM system, the interface should clearly allow the user to access the signals and the control parameters (e.g. the gains of the feedback), but it is also important to have advanced image processing and data analysis routines. Therefore we decided to run the RTAI-XML server on the controller and to migrate the user interface on a remote machine, being free to choose the development platform and operating system. In particular, We have started the development of the code on the basis of a commercial AFM system, SPMagic by ElbaTech [19]. The user interface of this product is released under the GPL (General Public License) license. Using the remoting functionalities offered by RTAI-XML, we are able to inherit the user interface of the SPMagic system, re-coding only the control communications routines, and substituting direct access to the hardware with XML-RPC calls. Fig. 9 shows a screenshot of the SPMagic GUI.

V. CONCLUSIONS

This paper has presented a new software platform for the integration of web services and CACSD environments for the open-source Linux RTAI operating system. The new paradigm allows for a strong separation between the real-time core of the application and its Graphical User Interface which can now be designed with specific tools without any RT constraint. In the *Interactive Session* the main features of the proposed approach will be shown through a remote session with one of our Labs.

REFERENCES

[1] W3C. Web Services Activity. [Online]. Available: http://www.w3.org/2002/ws/
[2] RTAI. Real Time Application Interface. [Online]. Available: http://www.rtai.org

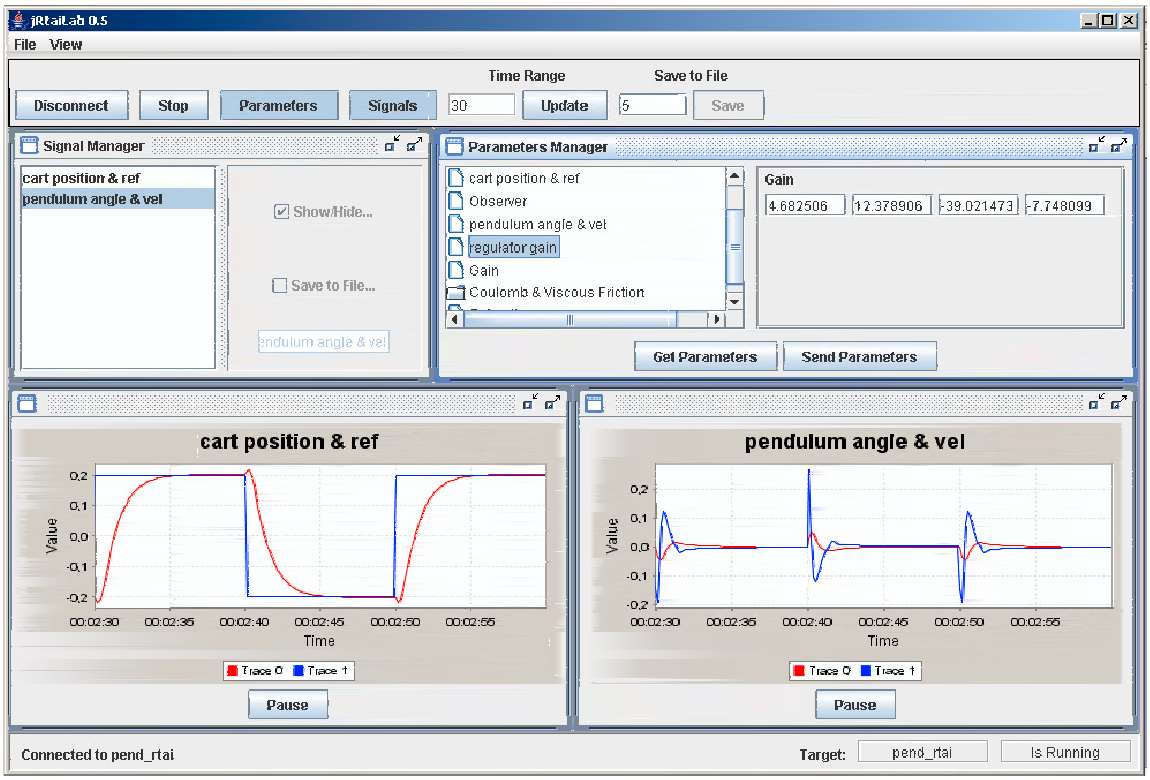

Fig. 8. JRTAIlab application interface.

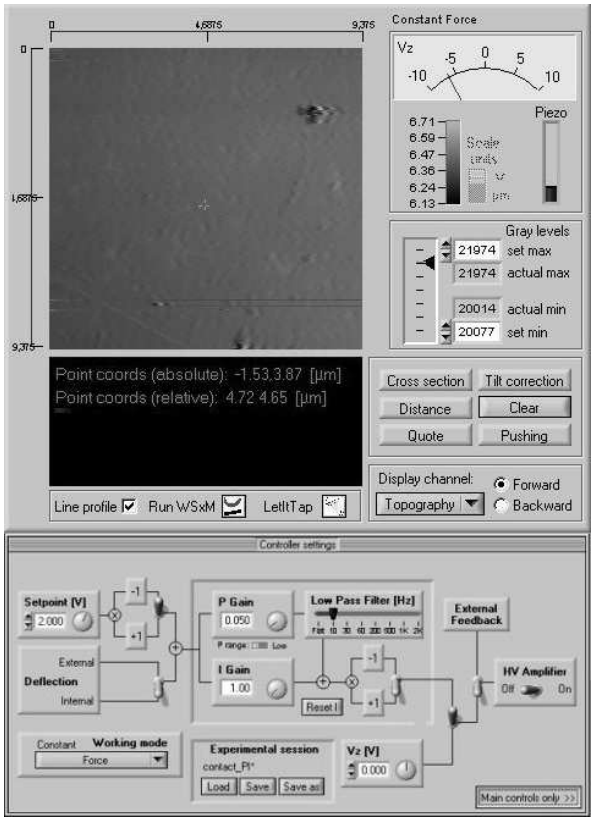

Fig. 9. AFM SPMagic interface.

[3] M. Basso, M. Romagnoli, and M. Vassalli. RTAI-XML Project. [Online]. Available: http://artist.dsi.unifi .it/rtaixml/

[4] W3C. SOAP Version 1.2 Part 0: Primer. [Online]. Available: http://www.w3.org/TR/soap12-part0/

[5] COMEDI. Linux Control and Measurement Device Interface. [Online]. Available: http://www.comedi.org

[6] ADEOS. Adaptive Domain Environment for Operating Systems. [Online]. Available: http://home.gna.org/adeos/

[7] K. Yaghmour. Adaptive Domain Environment for Operating Systems. [Online]. Available: http://www.opersys.com/ftp/pub/Adeos/adeos.pdf

[8] R. Bucher and L. Dozio. (2003) CACSD with Linux RTAI and RTAI-Lab. Valencia. [Online]. Available: ftp://ftp.realtimelinuxfoundation.org/pub/events/rtlws-2003/proc/bucher.pdf

[9] Mathworks. The MathWorks. [Online]. Available: http://www.mathworks.com

[10] Scilab. A Free Scientifi c Software Package. [Online]. Available: http://www.scilab.org

[11] Inria. Insitut National de Recherce on Informatique et on Automatique. [Online]. Available: http://www.inria.fr

[12] Enpc. École nationale des ponts et chaussee. [Online]. Available: http://www.enpc.fr

[13] RTNET. Hard Real-Time Networking for Linux/RTAI. [Online]. Available: http://www.rts.uni-hannover.de/rtnet

[14] J. Strothman. (2002) Will Web services replace HMI? [Online]. Available: http://www.isa.org/intech

[15] XML-RPC, Simple cross-platform distributed computing. [Online]. Available: http://www.xmlrpc.org

[16] M. Casini, D. Prattichizzo, and A. Vicino, "The automatic control telelab: A web based technology for distance learning," *IEEE Control Systems Magazine*, vol. 24, no. 3, pp. 36–44, 2004.

[17] M. Basso and G. Bagni, "ARTIST: A Real-Time Interactive Simulink-based Telelab," in *Proceeding of the 2004 IEEE Conference on Computer Aided Control System Design*, Taipei, Taiwan, Sept. 2004, pp. 196–201.

[18] National Instruments. National Instruments. [Online]. Available: http://www.ni.com

[19] ElbaTech s.r.l. SPMagic AFM System. [Online]. Available: http://www.spmagic.it