Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Innovative Tools for Real-Time Simulation of Dynamic Systems

Gianluca Palli * **Raffaella Carloni** * **Claudio Melchiorri** *

* *DEIS, University of Bologna,*
Viale Risorgimento 2, 40136 Bologna, Italy
Phone: +39 051 20 93034, Fax: +39 051 20 93073
{*gianluca.palli, raffaella.carloni, claudio.melchiorri*}*@unibo.it*

Abstract: In this paper, we present a software architecture, based on RTAI-Linux, for the real-time simulation of dynamic systems and for the rapid prototyping of digital controllers. Our aim is to simplify the testing phase of digital controllers by providing the real-time simulation of the plant with the same interface used for the communication between the control applications and real plant. This unified interface, based on the COMEDI library, allows to switch the controller from the simulated to the real plant without any modification of the control software. Moreover, a set of tools for helping the users in the development of the real-time simulation tasks of the plants have been developed. A great attention has been posed in the automatic generation of symbolic kinematic and dynamic models of robotic manipulators from a description of the robot in terms of kinematic parameters and inertia/center of mass of each link. The system, besides being useful for rapid prototyping of mechatronic control systems, may be used for fault detection, and also as a teaching tool in Mechatronic/Digital Control Courses. A case study, the real-time simulation and control of the PUMA 560 manipulator, is presented and discussed.

Keywords: Real-Time Systems, Programming Environments, Control Systems Design, Teaching Tools, Simulators.

1. INTRODUCTION

Rapid control prototyping represents an indispensable tool for the simulation of complex dynamic systems and the design of digital controllers, since it allows the validation of the modeling and the control algorithms before any practical implementation on a real plant. Within this context and with the aim of improving the reliability and the performance of the controller, real-time simulation environments can be used to test and modify theoretical models and control strategies, which have been developed under the assumption of a perfect knowledge of the system.

In the last years, the issue of rapid control prototyping has been addressed in several works, see (Bonivento et al. [2000], Bona et al. [2002], Chen et al. [2004]) among others, and it is a research topic both in the industrial and academic field. In particular, this kind of simulation/validation techniques may be of great interest in control educational environments for undergraduate/graduate levels of study (Palli and Melchiorri [2006, 2007]).

In Mechatronic and Digital Control Courses, standards CACSD [1] tools (e.g. Scilab/Scicos, Mathematica, Matlab/Simulink) are commonly used for the analysis of dynamic systems and for the synthesis of an appropriate control architecture. CACSD tools also offer packages (e.g. RTW [2] of Matlab/Simulink) that are able to convert the control law from the simulation environment to a real-time implementation written in a standard programming language, like C, with different RTOS [3] as target platform. Moreover, CACSD tools can implement the interface between the simulation environment and an I/O device that directly communicates with the real dynamic system. In general, the I/O device is a Data Acquisition Board (DAQ board) which is basically composed by several sub-devices, like digital/analogical converters (DAC and ADC), digital I/O signals, encoders and others. The control applications can be made independent from the specific DAQ board by using the COMEDI [4] library that provides an unified API [5] for control applications and that supports real-time Linux extensions, i.e. RT-Linux and RTAI [6].

In this paper, we present a software architecture, based on the RTAI-Linux open source environment, that allows the monitoring of the behavior of both the controller and the plant without really connecting the control system to the real process. Our aim is to create a rapid control prototyping system for both industrial applications and educational purposes so that the designer can test the control schemes even in critical or faulty conditions before applying the control to the real process. This architecture has the advantage that, once the controller has been tested, no more changes have to be done on the control software so that no new errors are introduced. Our architecture is based on an extension of the COMEDI library and it

[1] Computer Aided Control System Design.
[2] Real-Time Workshop.
[3] Real-Time Operating Systems.
[4] COntrol and MEasurement Device Interface (Schleef [Online]).
[5] Application Programming Interface.
[6] Real-Time Application Interface (Beal et al. [April 2000], Cloutier et al. [2000]).

is summarized in Fig. 1. In particular, in the left part of the figure, a general scheme of connection between a controller and a real plant is depicted: the controller is implemented in user-space and, through a proper kernel space driver, it communicates with the DAQ board and, then, to the real plant. On the other hand, the scheme in the right part of Fig. 1 depicts the architecture that we are proposing in this work. We have exploited COMEDI to implement the interface between user and kernel space so that, once the control law has been designed for the real-time simulation, it can be directly applied to the real time system without changing the code and/or adding other software parts. This grants that no new error sources are introduced. A similar approach to real-time simulation is proposed in (Ferretti et al. [2005]): in this work, a RTAI application for the simulation of dynamic system has been derived directly from a model of the plant made with Modelica (Fritzson and Engelson [1998]).

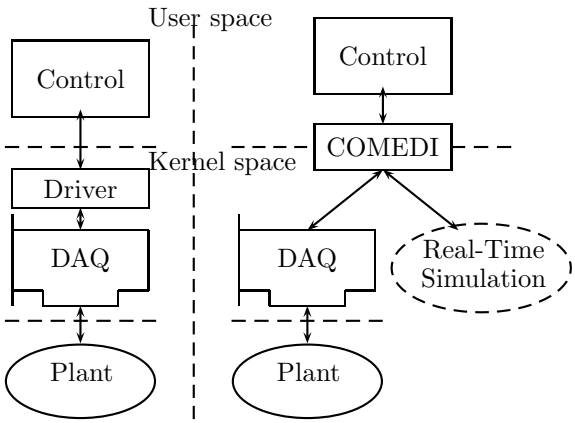

Fig. 1. Left: typical software structure; Right: software structure with the COMEDI library.

A crucial point in the proposed architecture is to ensure the real-time execution of the basic integration algorithm in the simulation task. For this purpose, we use the GNU Scientific Library (Galassi et al. [2003]) since, besides being open-source, it provides the necessary functions for the implementation of the simulation algorithms.

Motivated by the advantages that such a real-time simulation environment can introduce in automatic control educational activities, we have also developed a set of Simulink blocks for helping the students in the design of real-time simulation of generic dynamic systems and, in particular, robotic manipulators.

The paper is organized as follows: the real-time simulation algorithm for linear and non-linear systems is discussed in Sec. 2. Sec. 3 describes the implementation of the COMEDI driver for real-time simulation driver. A case study is discussed in Sec. 5, while conclusions and plans for future activities are reported in Sec. 6.

2. REAL-TIME SIMULATION OF DYNAMIC SYSTEMS

In this work, the real-time simulation of the plant is implemented with a periodic RTAI task, which performs all the necessary operations for the computation of the state evolution over time.

In the linear stationary case, the evolution of the plant can be described by using the standard discrete-time state space representation:

$$x(k+1) = A_d x(k) + B_b u(k), \quad x(0) = x_0 \tag{1}$$

$$y(k) = C_d x(k) + D_d u(k) \tag{2}$$

where A_d, B_d, C_d, D_d are matrices of proper dimensions and k is the discrete-time variable. Eq. (1)-(2) are very simple to be implemented with a periodic real-time task, and the required computational effort is very low.

In the generic non-linear case, the dynamics of the plant can be described by the classic state space representation:

$$\dot{x}(t) = f(x(t), u(t), t), \qquad x(t_0) = x_0 \tag{3}$$

$$y(t) = g(x(t), u(t), t) \tag{4}$$

where x, u, y are, respectively, the state, input and output vectors, t is the time variable, $f(\cdot)$ is the state velocity vector, $g(\cdot)$ is the output function and x_0 is the initial state. In order to compute the time evolution of the plant, the ordinary differential equation system (ODE) defined by Eq. (3) must be solved by means of a numerical integration algorithm over the period of the real-time task. This may require a very high computational effort, especially if a variable step integration algorithm is used.

In general, the integration algorithms can be characterized by either *fixed* or *variable* time step. In the fixed step integration case, the time step has to be chosen by considering the computational capabilities of the calculator and by taking into account that, in general, the error on the solution increases as the step size increases. In our implementation, the period of the simulation task corresponds to the integration time step. The experiments show that the execution time of the algorithm is almost constant during the simulation and that it mainly depends on the order of the system and on the computational capabilities of the computer. On the other hand, in the variable step case, at the beginning of the execution, the integration time step is initialized to the same period of the simulation task in order to estimate the first candidate solution. Then, the step doubling error checking method (Press et al. [1992]) is applied in order to evaluate the relative integration error and, in case, to subdivide the integration time step until the required error is achieved. In this case, the computational effort can be very high, especially if non-smooth dynamic functions are considered, but the reliability of the solution is considerably improved.

Controls on the integration error, on the integration step size and on the execution time are introduced in order to avoid the system starvation and to satisfy the real-time constraints. Therefore, it is possible to require the simulator to solve the dynamic equations with a given precision, in terms of relative or absolute error: if the period of the simulator task is too small to allow the computation of a solution with the prescribed error, the algorithm is stopped and the last available solution is returned. In Fig. 2 the flowchart of the real-time integration algorithm is reported. In general, the period of the simulation task must be grater or, at least, equal than the controller period. In the case of variable step algorithms, the simulation task period can be chosen equal to the period of the controller: the error checking procedure can refine the integration step to satisfy the specifications on

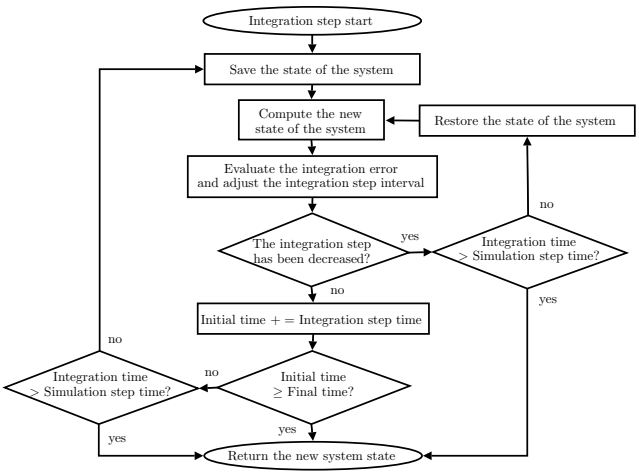

Fig. 2. Flowchart of the real-time integration algorithm.

the integration error. The use of fixed step algorithms can introduce a large error drift in the solution of the system because there is no control on the integration error.

The period of the simulation task has to be chosen large enough to allow the solver to compute at least a possible solution of the ODE system. If necessary, the time scale of the simulation and of the controller can be expanded to fit the computational capabilities of the hardware platform. On the other hand, if the time required by the integration algorithm is small enough with respect to the simulation step, it is possible to compress the time scale to evaluate the behavior of the controller in a shorter time, recovering, in some sense, the features of other simulation environments, but performing the simulation using the final control application.

In this work, we have used different functions of the GNU Scientific Library (GSL) for the implementation of the numeric integration methods for solving the differential equations describing the plant. Since GSL is an user space library, we had to introduce proper modifications to the source code in order to use this library in kernel space together with the COMEDI board drivers and in order to grant the real-time execution. With this changes, we can use all the fixed/variable step integration algorithms available in GSL in the implementation of the plant.

3. THE COMEDI REAL-TIME SIMULATION DRIVER

In order to implement the dynamic equations of the plant in the real-time simulator, it is required to write these equations using a programming language, like C. In general, the generation of the C code for dynamical equation can be easily obtained by using CACSD systems like Mathematica or Modelica. Once the C code is available, it is possible to enclose it into a generic real-time simulation COMEDI driver. For this purpose, a Real-Time Simulator Target for RTW has been created together with a suitable component library to allows the generation of real-time simulation drivers of dynamic systems directly from Simulink models.

The COMEDI library is a standard interface for DAQ boards. It provides a set of functions, the API, that allow the communication with all the different sub-devices present on the data acquisition board, the configuration of these devices and the exchange, the conversion and the management of all the data. COMEDI supports both synchronous and asynchronous data transfers via buffered or direct data acquisition instructions. The COMEDI library is composed by two different parts: a user-space library that implements the API for user space applications, and a set of kernel modules, called COMEDI drivers, for the communication with the hardware. Each kernel modules implements all the specifics functions for a particular board or board family.

The COMEDI interface allows to communicate with the acquisition hardware through characters device files. Since the association between the board and the file is dynamic and is specified via command line during the COMEDI configuration, it is possible to support simultaneously many DAQ boards on the same system by associating each of them to a different file.

In this paper, this mechanism is used to create a COMEDI driver for a virtual board that implements, besides the necessary functions to emulate the acquisition hardware, the real-time task for the simulation of the plant. Each data sent to the DAQ board change the value of the input variables (or some parameters) of the plant simulation task and the data read functions get the value of the plant outputs. Since some properties of the DAQ boards, e.g. setting time and conversion time, are not taken explicitly into account by COMEDI, we have to manage this parameters in order to simulate also the delays introduced by the electronic devices.

4. REAL-TIME SIMULATOR DESIGN TOOLS

One of the most important feature of a simulation environment, from the point of view of the final user, is the availability of suitable tools, for the simplification of the design procedure, and component libraries to hide low level implementation details. Some well-known examples of these tools are Simulink and Dymola.

On the other hand, the requirements in terms of efficiency for the implementation of real-time simulation tasks impose the use of dedicated applications. To help the users in the development of these applications, a set of tools has been developed with the aim of hiding all the implementation details needed by the real-time operating system in which the simulator will run. By exploiting the features and the flexibility of the Matlab/Simulink/RTW environment, the attention is focused only in the modeling of the plant, letting to the underlying system the automatic generation of the related code for real-time simulation.

Usually, students use Matlab/Simulink/RTW to:

- Analyse the plant model and design the control law;
- Simulate the behavior of the system in both continuous and discrete time domain;
- Generate the real-time control application.

With the proposed environment, the users are also able to:

- Generate the code of the plant real-time simulator;
- Test the behavior of the control application on the real-time simulated system;
- Compare the response of the real and the simulated system under the control of the same application.

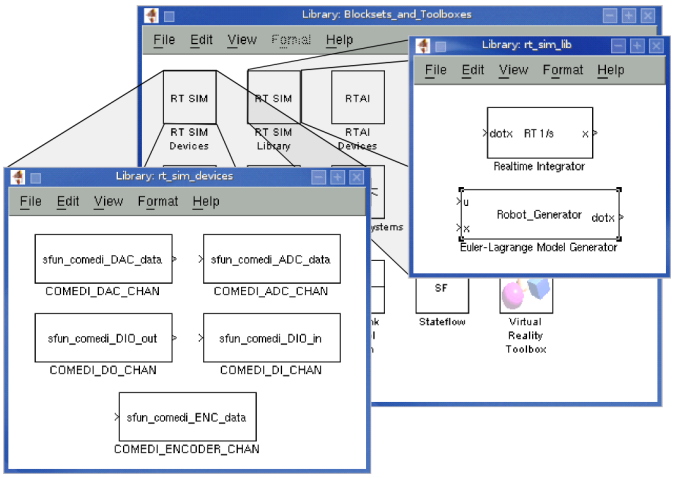

Fig. 3. Real-Time Simulator Simulink blocks.

4.1 Generic Dynamic Systems Modeling

For algebraic or discrete time dynamic systems, described by Eq. (1)-(2), the sub-device blocks shown in Fig. 3 can be used in Simulink to build the real-time simulation model. In particular, these blocks are used to connect the input and output signals to the sub-devices of the virtual DAQ board used to control the plant. The properties of the sub-devices (like channels, resolution, data range) can be set through a dialog box. The real-time simulation application is then generated by RTW using the *Linux Real-Time Simulation target* as target environment.

Continuous time dynamic systems, described by Eq. (3)-(4), can be modeled in Simulink by using standard blocks plus the `realtime_integrator` block (see Fig. 3) in which the real-time integration algorithm described in Sec. 2 is implemented. In the `realtime_integrator` block dialog box, it is possible to choose the desired stepping method among all the available in the GSL library, the initial conditions, the sample time and the amount of time (in percents over the sample time step) that can be used by the integrator to compute the solution.

4.2 Generation of Manipulators Dynamic Models

The real-time simulation of the dynamics of a robotic arm requires the use of a computational-efficient dynamic model. Several software packages for the computation of robotic manipulator dynamics are available, like the Robotic Toolbox (Corke [1995, 1996]) or RobotiCAD (Falconi et al. [2006]) for Matlab/Simulink, or like ROBOOP (Gourdeau [2006]) for the C programming language. These tools use the Newton-Euler recursive method for the computation of the manipulator dynamics. This approach is justified by the difficulties in the definition of the Euler-Lagrange closed form dynamic model of robotic arms when the number of DOF [7] is greater than two.

By using the open source C++ GiNaC [8] library, it is possible to embed, into custom applications, symbolic manipulations capabilities of generic mathematical functions. This library has been used for the generation of the symbolic Euler-Lagrange dynamic model of robotic manipulators given its description in terms of Denavit-Hartemberg parameters (Spong and Vidyasagar [1989]) and the properties of each link (link inertia tensor/mass and position of the center of mass). Also the gravity field can be arbitrarily assigned. The algorithm for the generation of the Euler-Lagrange model is described in (Yoshikawa [1990]).

[7] Degrees of freedom.

[8] GiNaC is an iterated and recursive acronym for GiNaC is Not a CAS, where CAS stands for Computer Algebra System (Bauer et al. [2002]).

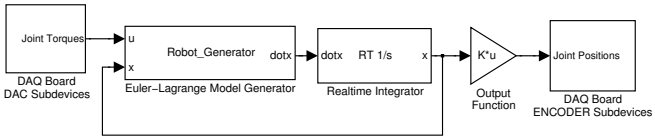

Fig. 4. Simulink scheme of the robot real-time simulator.

Fig. 4 shows the Simulink scheme of a generic robotic arm real-time simulator. In this figure, the `robot_generator` block (see also Fig. 3) receives as inputs the actual manipulator state $x = [q \quad \dot{q}]^T$, where q is the joint positions vector, and the vector of joint torques u, while it provides as output the state time derivative $\dot{x} = [\dot{q} \quad \ddot{q}]^T$. The DACs and the encoders of the DAQ board connected to the robot are collected into two subsystems. The output function selects only the position information as output of the robotic system.

5. A CASE STUDY

The software architecture described in the previous Sections has been used for the simulation and the control design of a number of mechatronic devices. The Unimation PUMA 560 6-DOF manipulator is considered here as a case study to show the ability of the proposed environment in the generation of the dynamic model of the system and to compare the responses of the real-time simulator and of the real plant.

Fig. 5 depicts the PUMA 560 robotic manipulator together with the reference frames of all its links. The parameters and the explicit dynamic model of the PUMA 560 are not here reported for brevity but they can be found, among others, in (Armstrong et al. [1986], Corke and Armstrong-Helouvry [1994]).

5.1 The Control System

The control platform is based on a PC-104 industrial computer, with a AMD Elan520 processor (486 compatible) at 133 MHz, 32 MB of RAM and two Sensoray 526 DAQ board. The operating system is Debian-GNU Linux SID with kernel 2.6.19.5, RTAI 3.5 and GCC 3.3.4. The COMEDI library has been obtained from the CVS [9] archive. Note that this software platform has been chosen because all its components are open source. The control system of the PUMA 560 manipulator has been modified to allows a direct control of the system. The TRC-041 (Robotics and Inc. [2007]) card set has been used to directly connect the DAQ boards to the power unit of the robot. The TRC-041 provides the encoder signals, the potentiometer signals (for absolute position references) and the current control input of each axis of the manipulator.

[9] Concurrent Versions System, an open source tool to manage the distributed development of software projects (Fogel and Bar [2003]).

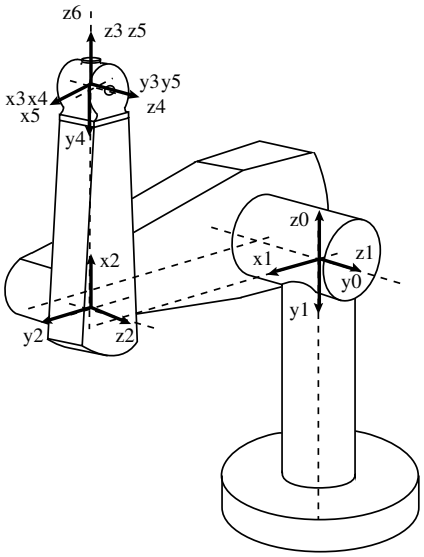

Fig. 5. Scheme of the PUMA 560 robotic arm and link reference frames.

To verify the behavior of the robot, a joint PD controller with gravity compensation (Sciavicco and Siciliano [1996]) has been used. The gravity terms are computed according to the model reported in (Armstrong et al. [1986]). The control application has been generated from the Simulink scheme using RTW. The details about the implementation of the robot controller are not reported here for brevity and because is not the focus of this discussion. Note that this control application has been used, without any modification, for the control of both the real manipulator and the simulated one. The response of these two systems are reported and compared in Fig. 6-7. Fig. 6 shows the positions of the arm joints 2 and 3 (the most affected by the gravity terms) and of the wrist joint 5 while in Fig. 7 the tracking error of the same joints of both the real and the simulated robot. In particular, from these latter plots, it is possible to note that the behavior of the two systems is almost the same. The small differences in the response between the real and the simulated robot can be ascribed to the large uncertainty on the joint static friction effects.

5.2 The Simulation Platform

The simulation platform is based on a Intel Centrino 1.6 GHz Notebook equipped with 512 Mb of RAM. The operating system is Debian-GNU Linux SID with kernel 2.6.17.11, RTAI 3.5 and GCC 3.3.4. The COMEDI library has been obtained from the CVS.

With the aim of making available to students or not-skilled users this software platform without any software installation or configuration requirements, a Knoppix-based Linux Live-CD distribution called RTAI-Knoppix [10] has been developed (Knopper [Online]). This live distribution contains all the necessary tools like IDE [11], editors, libraries, examples and source code for the development of real-time applications and other tools for office, Internet, etc. RTAI-Knoppix provides also an auto-configurable GUI [12] with xrtailab (Bucher and Dozio [2003, 2004]), the graphic interface for RTAI applications.

[10] RTAI-Knoppix is available for download at: http://www-lar.deis.unibo.it/people/gpalli/files/rtai-knoppix.iso
[11] Integrated Development Environment.
[12] Graphic User Interface.

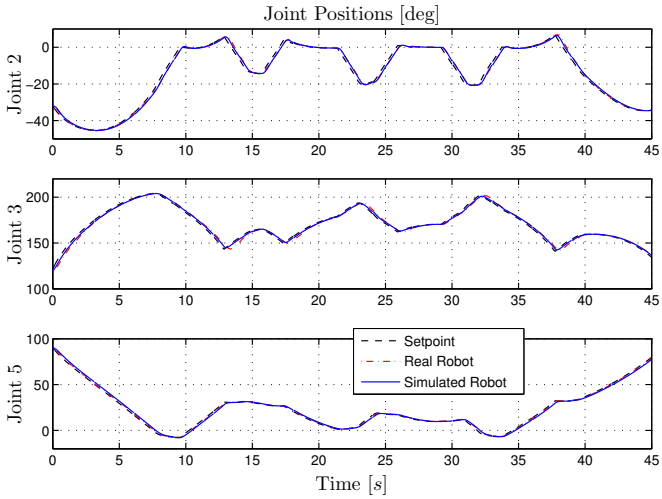

Fig. 6. Joint positions of the PUMA 560 Arm.

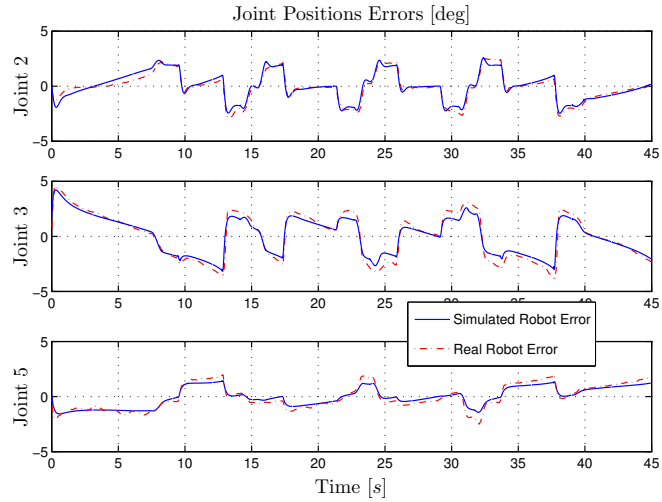

Fig. 7. Joint position errors of the PUMA 560 Arm.

6. CONCLUSIONS AND FUTURE WORK

In this paper, some innovative tools for the real-time simulation of dynamic systems have been presented. This environment, based on RTAI-Linux and on the COMEDI library, allows to the control system designer to execute the real-time control application and to simulate in real-time the plant to be controlled. When the behavior of the system is satisfactory, the control application can be simply switched from the simulated to the real process without any change in the software. Although being somehow limited to the RTAI-Linux environment, the system would facilitate the implementation and rapid prototyping of digital control systems.

The system, besides for design purposes, has also been used in the last three years as a teaching support in digital control courses as an alternative to standard laboratory setups. In this case, the student may be asked to face both standard control design problems and aspects related to implementation of multi-task real-time software.

Note that, if a proper model of the plant is available, with this system it is possible to run simultaneously two identical controllers, one connected to the plant and one to the simulator. This could help in the real time evaluation of the response of the plant to the given commands before their real application, and also for supervision/fault detection purposes.

With respect to our previous works, in this paper a great attention has been posed in the simplification of the design procedure of real-time simulation tasks. The introduction of a C++ symbolic manipulation library like GiNaC gives us the possibility of automatically generating the dynamic model of complex nonlinear systems, like robotic arms, directly form a physical description of the device. Future activities will be focused, other than to improve of these tools, to the integration of this environment with a 3D graphic rendering of the plant evolution.

REFERENCES

B. Armstrong, O. Khatib, and J. Burdick. The explicit dynamic model and inertial parameters of the PUMA 560 arm. In *Proc. of the 1986 IEEE Int. Conf. on Robotics and Automation*, pages 510– 518, San Francisco, CA, 1986.

C. Bauer, A. Frink, and R. Kreckel. Introduction to the GiNaC framework for symbolic computation within the C++ programming language. *J. Symbolic Comput.*, 33 (1):1–12, 2002.

D. Beal, E. Bianchi, L. Dozio, S. Hughes, P. Mantegazza, and S. Papacharalambous. RTAI: Real Time Applications Interface. *Linux Journal*, April 2000.

B. Bona, M. Indri, and N. Smaldone. An experimental setup for modelling, simulation and fast prototyping of mechanical arms. In *Proc. IEEE Int. Symp. on Computer Aided Control System Design*, pages 207–212, 2002.

C. Bonivento, A. Tonielli, and C. Melchiorri. A PC-based rapid prototyping workstation for the design of motion control systems. In *1st IFAC Conf. on Mechatronic Systems*, Darmstadt, Germany, Sept. 18-20 2000.

R. Bucher and L. Dozio. CACSD under RTAI Linux with RTAI-Lab. In *Realtime Linux Workshop*, Valencia, Spain, 2003.

R. Bucher and L. Dozio. Rapid control prototyping with Scilab/Scicos and Linux RTAI. In *Scilab-2004, 1st Scilab International Conference*, December 2-3 2004.

C. Chen, H. Tsai, and J. Tu. Robot control system implementation with rapid control prototyping technique. In *Proc. IEEE Int. Symp. on Computer Aided Control System Design*, pages 278–283, 2004.

P. Cloutier, P. Mantegazza, S. Papacharalambous, I. Soanes, S. Hughes, and K. Yaghmour. DIAPM-RTAI Position Paper. In *2nd Real-Time Linux Workshop 2000*, Orlando, Florida, USA, November 27-30 2000.

P. I. Corke. A computer tool for simulation and analysis: the robotics toolbox for MATLAB. In *Proc. National Conf. Australian Robot Association*, pages 319–330, 1995.

P. I. Corke and B. Armstrong-Helouvry. A search for consensus among model parameters reported for the puma 560 robot. In *Proc. of the 1994 IEEE Int. Conf. on Robotics and Automation*, pages 1608–1613, San Diego, CA, USA, May 8-13 1994.

P.I. Corke. A robotics toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, 3(1):24–32, March 1996.

R. Falconi, C. Melchiorri, A. Macchelli, and L. Biagiotti. RobotiCad, a Matlab toolbox for robot manipulators. In *Proc. of the IFAC Int. Symp. on Robot Control SYROCO 2006*, 2006.

G. Ferretti, M. Gritti, G. Magnani, G. Rizzi, and P. Rocco. Real-time simulation of modelica models under Linux / RTAI. In Gerhard Schmitz, editor, *Proc. of the 4th Int. Modelica Conference*, Hamburg, Germany, March 7-8, 2005.

K. Fogel and M. Bar. *Open Source Development with CVS, Third Edition*. Paraglyph Press, 2003. Avaiable online: `http://cvsbook.red-bean.com/`.

P. Fritzson and V. Engelson. Modelica - A unified object-oriented language for system modeling and simulation. *Lecture Notes in Computer Science*, 1445:67–, 1998.

M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, M. Booth, and F. Rossi. *GNU Scientific Library Reference Manual*. Network Theory Ltd., 2 edition, 2003. Avaiable online: `http://www.gnu.org/software/gsl/manual/`.

R. Gourdeau. ROBOOP: A Robotics Object Oriented Package in C++, 2006.

K. Knopper. Knoppix web page. `http://www.knoppix.org/`, Online.

G. Palli and C. Melchiorri. Realtime hardware emulation for rapid prototyping and testing of control systems. In *Proc. 4th IFAC Symp. on Mechatronic Systems*, 2006.

G. Palli and C. Melchiorri. A realtime simulation environment for rapid prototyping of digital control systems and education. *Automazione e Strumentazione*, 55(2): 98–105, 2007.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992. ISBN 0521437148.

Trident Robotics and Research Inc. TRC041 Puma cable card set, 2007.

D. Schleef. COMEDI - The Control and Measurement Device Interface web page. `http://www.comedi.org/`, Online.

L. Sciavicco and B. Siciliano. *Modeling and Control of Robot Manipulators*. McGraw-Hill, 1996.

M. Spong and M. Vidyasagar. *Robot Dynamics and Control*. John Wiley and Sons, 1989.

T. Yoshikawa. *Foundations of robotics: analysis and control*. MIT Press, Cambridge, MA, USA, 1990. ISBN 0-262-24028-9.