Development of robot controllers based on PC hardware and open source software

D. Dallefrate, D. Colombo And L. Molinari Tosatti
ITIA-CNR Institute of Industrial Technologies and Automation
Viale Lombardia 20/A, 20131, Milan, Italy
{d.dallefrate, d.colombo, l.molinari}@itia.cnr.it

Abstract

Researchers, developers and system integrators require more openness in control systems in order to design, develop and rapidly integrate functionalities to fulfill new application requirements. Even if several robot control vendors provide their products with customized development tools, low cost and not proprietary solutions should be preferred to face the rapid market changes and to reduce control life cycle costs. In fact, most important success factors are the use of off-the-shelf hardware and open source and/or free software, as well as a design focused on reconfigurability and portability of the control software is crucial. In this paper a control architecture which complies with the above mentioned requirements is presented. Specifically a modular control portable on different platforms has been designed, implemented and then validated on two different platforms both based on real-time operating systems (RTOS). The porting activity between the two platforms and specifically between QNX 4.25 and RTAI Linux has been done using features compliant to the POSIX standard. As specific target, the proposed architecture has been exploited to control two robots with different kinematics architectures. The first one is a 4 d.o.f parallel robot prototyped at ITIA-CNR while the second one is an industrial 7-d.o.f robot manufactured by Mitsubishi which is based on a serial kinematic chain. The obtained results for the second application case are presented more in detail. Such a robot has an open architecture dedicated to Research & Development that allows the direct control of servo PLC controller through an upper level PC controller. To guarantee the real-time communication, the servo and upper level controller are linked by a dedicated fiber optic network based on Arcnet interface at 100 Mb/s. Procedures have been developed to move the robot in a safe and robust way.

1 Introduction

Nowadays, to be competitive, handling rapidly changes in product design, in product mixes and volumes is crucial. In such a scenario, researchers and developers, system integrators and end users require more and more openness in systems they deal with. Focusing on control systems, during the last decade, the openness issue has been addressed in several ways by worldwide research projects both in the field of machine tools (i.e. OSEC/JOP, OMAC, OSACA) [1] and in the field of Robotics (i.e. OROCOS) [2]. Although a universal agreement on definition of Open Control Systems (OAC) has been not yet achieved, a basic set of common requirements has been drawn and they can be summed up as follows:

- an OAC allows the portability of software on different hardware platforms and operating systems
- an OAC is modular having the capability of easy replacement of modules with new ones, allowing scalability both in performances and functionalities
- an OAC allows easy integration of new functionalities. This means firstly that it is based on standards to let third parties develop hardware and software that meet new requirements; secondly, it means that an OAC provides plug-and-play mechanisms for fast integration of such new functionalities.
- an OAC is reconfigurable providing mechanisms of easy adaptation of its parameters in order to customize the system to different application scenario

- an OAC is economic since it is based on off-the-shelf modules both with reference to hardware and software

Even if many control vendors offer integrated development tools to customize their systems, such tools are often based on proprietary solutions. Then, providing customizations for specific applications, for instance modifications to better exploit robots and machines with new kinematics [3], requires great engineering efforts in term of knowledge, time and cost. On the other hand, controls based on PC hardware cover mostly the requirements of an OAC. Main reasons include the increasing power of PC microprocessors that make them suitable for real-time application, economies of scale of the PC market that reduce their cost, the easy integration of PCs in enterprise network that fulfills the need of accurate manufacturing information at all enterprise level. Another key point for a modern OAC is the use of open source software (OSS) which has attracted recently also the field of numerical controls for machine tools [4]. One of the most promising use of OSS in OAC is at operating system level: first, OSS allows to reduce cost of the whole controller cutting the license fee of commercial RTOS, then OSS has the advantages of full code sources availability that allow developers to provide their own customization [5].

In this paper ITIA approach to robot controllers based on PC hardware and open source software is presented. Specifically, the work is organized as follows: in Section 2 an overall view of the robot control architecture is shown. With reference to such an architecture, in Section 3 the correspondent software implementation is detailed and two different test beds are presented. In Section 4 the specific results obtained controlling a 7 d.o.f industrial robot arm are described; then final conclusions are drawn.

2 Architecture design

The proposed architecture shown in figure 1 is basically divided in two layers: an *application layer* that contains modules needed by a robotic controller; a *system layer* where main functionalities of an RTOS are encapsulated. The application layer is made up by 4 main modules: interpreter, trajectory generator, kinematics, axis controller and axis driver.

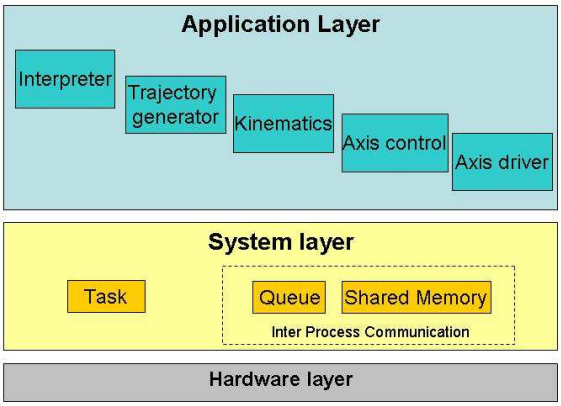

FIGURE 1: *Reference architecture*

The *Interpreter* translates high level motion commands specified by the user into an inner data format (i.e. motion data structure) suitable for further elaborations of the controller described below.

The *Kinematics* includes algorithms to solve transformations from Cartesian to joint space of the controlled robot and vice versa and to check robot workspace and limits. An easy reconfiguration of this module is a basic requirement in order to apply rapidly the controller to different robots achieving one of the goals of an OAC (chapter 1); such a reconfiguration can usually be done in two ways: by means of the definition of a general set of parameters that describe a wide range of kinematics architectures; by means of the implementation of different kinematics modules, each one specific for one kind of architecture. Specifically, this is the design rule followed in our reference architecture. In this case reconfiguration is done by the substitution of a kinematics module with another one. This operation is feasible without effort only if all kinematics modules implement the same interfaces.

The *Trajectory Generator* acts through three different levels: first, it provides the path planning generating a Cartesian trajectory according to the geometrical data specified in the motion data structure prepared by the Interpreter; then, it performs a sampling of such a path according to a given motion law; finally it calls services provided by the kinematics module to transform end effector movements to actuators coordinates set points.

The *Axis Controller* keeps the actual actuators coordinates as close as possible to the corresponding set points by means of proper control algorithms.

The *Axis Driver* is the software interface to the actual I/O board that allows bi-directional data communication between the controller and the robot;

specifically, it allows the reading of sensors signal (encoders, analog and digital inputs) and the writing of command set points.

Beneath the control layer, in order to achieve the software portability on different RTOS platforms, the system layer has been designed. Specifically, it encapsulates some basic functionalities of an RTOS, i.e. task management and inter process communication, providing three modules: task, queue and shared memory.

The *Task* provides functionalities to handle an RT process. Specifically it allows the task configuration setting basic properties such as id, priority, sample time, the task activation and deactivation (i.e make periodic, run, suspend, resume,kill); finally, it provides diagnostic functions to check the task state.

The *Queue* and the *Shared Memory* modules encapsulate respectively the mechanisms of the same name for the communication between different tasks.

3 Implementation

This section presents the implementation of the above mentioned reference architecture. More specially, implementation for the application and system layer are shown in detail by applying the reference model to two different robots, a 4-dof parallel kinematics robot and an industrial robot, and two different RTOS, QNX 4.25 [7] and Linux RTAI[6].

3.1 Control system

Figure 4 shows the five main processes interacting in the robot system control:

- Command interpreter process
- TrjGen process
- pControl process
- MC server process
- HMI Server process

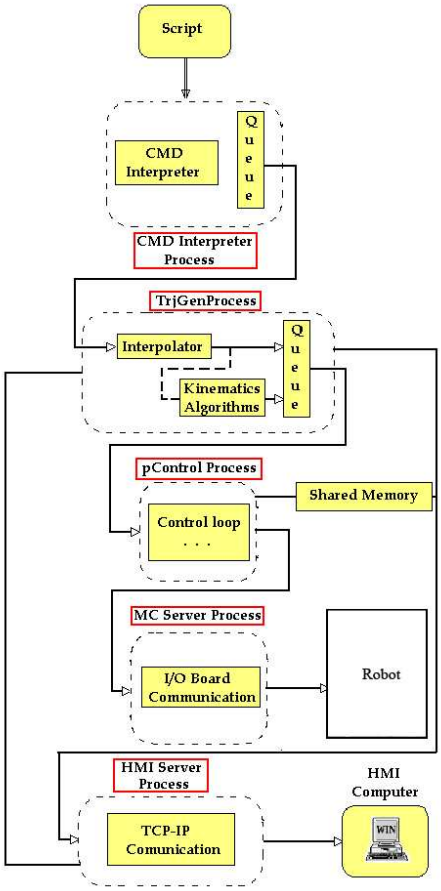

FIGURE 2: *Control Scheme*

The command interpreter process receives from the shell or from the HMI Server process robot scripts. A robot script is an high level command that can be expressed in Cartesian or joint space. The user has to explain the movement command type, the posture to reach, and the time needed. Command movement type can be linear, circular or spline. The main task for the command interpreter process is to make an interpretation of the robot script message and to insert it into a command queue. The Trajectory generator process reads a value from the command queue then, using several interpolation routines (i.e. linear, circular, spline), it generates a set of points with a given sample time (1 ms in our application). If the robot script makes the request of a Cartesian movement the process calls the kinematics functionalities from the kinematics library. Such a library is composed by the following main procedures: position transformations through direct and inverse kinematics algorithms, velocity transformations through Jacobian functions, and work-space verification through Is-Reacheable function. The planned points are filled into a queue that is named reference-queue. Then

pControl, the process with the highest priority in the system, extracts one value from the reference queue at each cycle. Specifically, this process is the heart of the robot controller where axis control algorithms, alarm and synchronization procedures have been implemented. A shared memory is used to make a synchronization of the pControl Process and the TrjGen Process. The pControl process has to manage the five possible states of the machine: initialization when parameters are loaded; the starting which powers the robot on; the control loop when control algorithms are executed; the stopping which powers the robot off; the alarming when an error occurs. In all these machine states the pControl process communicates with the robot through the MC-Server Process, that makes possible the communication with the I/O board encapsulating all hardware dependent functionalities. Furthermore, for monitoring purposes all information related to the machine status (e.g. position, velocity, motor current and so on), is written into a data-structure stored in a shared memory by the pControl process. The HMI Server process, polling such a shared memory with a fixed rate of 10HZ, sends the machine status through the Ethernet LAN using TCP/IP protocol to all client that have been connected to the HMI Server. The HMI Clients are also allowed to send high level commands to the controller using Telnet protocol.

3.2 4-dof parallel kinematic robot

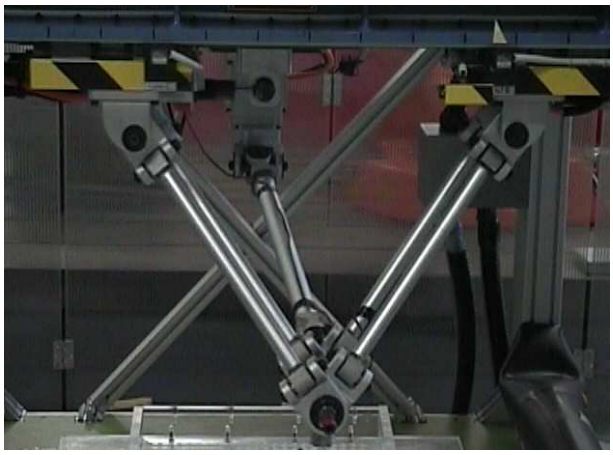

FIGURE 3: *Morpheum Manipulator*

The above mentioned implementation has been first exploited for the control of a parallel robot designed and developed by I.T.I.A. whose name is MorpheuM [8]. Such a robot is based on a particular parallel structure that allows the translation and the rotation of the end effector along its axis. The translation is achieved by means of linear actuators while the rotation is actuated by a brushless torque motor and the movement is transmitted by a double universal joint. Techno-economic advantages of such a robot derive from the symmetry structure and the high modularity and from the possibility to obtain multiple configuration of the end-effector through a structural reconfiguration from a minimum of two till a maximum of six degrees of freedom. Furthermore the work space of the manipulator is adaptable changing the length of the motor's linear guides. The flexibility and modularity qualities let the robot be applied to a large field of tasks[9], ranging from pick & place to assembling, spray painting, laser or water jet cutting. The performance of the robot permits to reach 3.5 m/s of velocity and acceleration above 3g with a maximum load of 5 Kg applied to the end effector. Regarding the controller, the hardware is based on a PC with Intel architecture (i.e. P4 2GHz) and QNX 4.25 as a operating system [10]. For such a robot the implementation of the control system has required the following customizations. Within the application layer a new implementation of the kinematics module to deal with the parallel structure of the robot has been developed; then both the interpreter and the trjGen process have been reconfigured to manage 4D data. At system level, the Task module has been implemented to encapsulate QNX 4.25 functionalities for process management, then OS implementation of POSIX Queues and SharedMemory are used. Regarding the communication with I/O the MC Server module has been develop to hide the use of a driver provided by the manufacturer of the axis control board.

3.3 Industrial robot arm

The second implementation is related to the Mitsubishi PA-10, a commercial robot with an open architecture that makes the manipulator suitable for research and development. The robot is composed by seven links that allow seven d.o.f. (three rotation and four pivot axes); its structure is shown in figure 3. The geometrical characteristics of the PA-10 manipulator are as follow: the length of the manipulator is 1.37 m and the weight is about 35 kg. The robot joints are actuated through three-phase AC servo motors, while the power is given through the domestic 230V AC. The setup that is normally sold, consists of a robot arm, a servo controller, a motion control card, and an upper control computer linked to the servo controller by means of a dedicated Arcnet optic fiber network. Several levels of control are allowed by the standard Mitsubishi axis control card, from an higher level controlling the movement in Cartesian space by using an inner interpolator to a lower level controlling directly the position of the

joints. Using the high-level Mitsubishi libraries provided in the upper controller, the maximum control frequency is 100Hz. In the control mode used in our application the Mitsubishi upper controller has been replaced by ours, bypassing the high level control libraries. In this mode it's possible to obtain a control of the robot with a frequency of 1kHz. The control of the servo may be carried out in Torque mode or in Velocity mode. At the moment, the second modality has been chosen, sending a velocity signal for each joint to the servo. This means that position loops are closed within the upper controller. The servo has a digital PI feedback that allows the joint-torque control of the robot. The upper controller runs the Linux RTAI 3.2 operating system and supports an Intel Pentium 4 processor 2.4 Gbyte. A CControl PCX20020 Arcnet card is mounted on the upper control computer to allow the communication with the servo controller. Specific electrical modifications[11] have been done to make the Arcnet card compatible for PA-10 application. The nominal communication, allowed by a specific optic fiber connection, reaches a maximum of 10 Mb/s. Regarding the upper controller software, the architecture described in chapter 2 has been applied. First some customizations have been done within the Application layer. Specifically, they involve the kinematics, the interpreter and the trjGen in order to deal with a 7 d.o.f. serial robot. Then, at system level a specific RTAI kernel module has been used to provide the implementation of POSIX message queues. Finally a custom implementation of Task and SharedMemory modules have been developed to encapsulate the same Linux RTAI functionalities. The communication with the hardware has been achieved by means of customization of Arcnet driver provided by standard Linux kernel 2.6.10; all such functionalities have been encapsulated in the MC server Process.

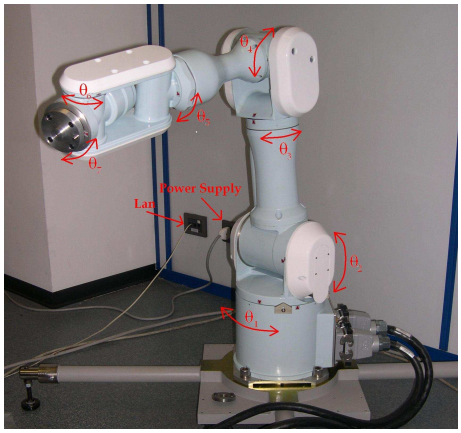

FIGURE 4: *PA-10 Manipulator*

4 Experimental results

With reference to the implementation of the controller for the PA-10 robot, hereunder some experimental results are shown in order to validate both hard real-time and control performance of the system. In detail, first the cycle time of the control process has been measured. The results are drawn in the figure below.

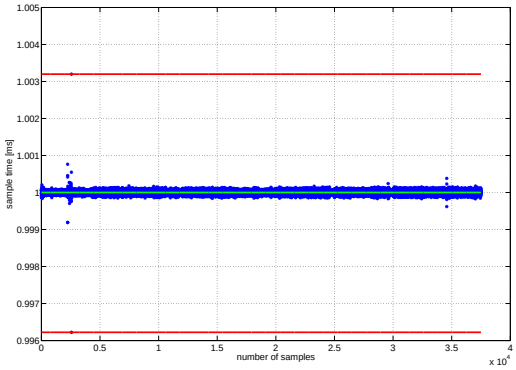

FIGURE 5: *RT performance of PA10 controller*

The test, executed when the controller is running, has shown a maximum error of 3.777 μs with reference to a nominal sample time of 1 ms. Thus, such results confirm the hard real-time performance of the whole system. Regarding the control performance, the trajectory followed by the 6th axis is compared with the reference one as shown in figure 6.

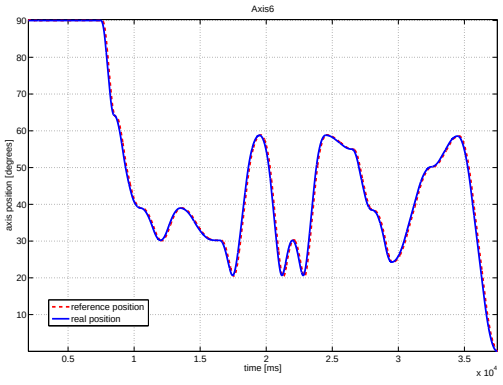

FIGURE 6: *Comparison between planned and followed trajectory*

Specifically, in figure 7 the following error is drawn: it ranges between a minimum of -3.3 to a maximum

of +3.7 degrees.

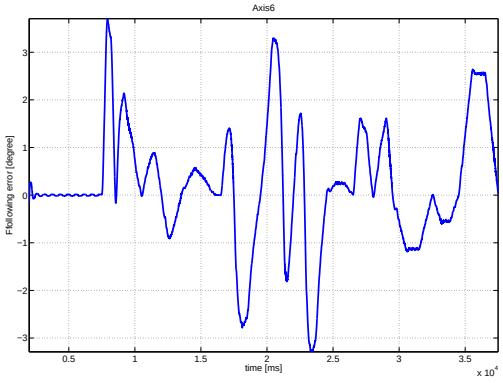

FIGURE 7: *Following error of 6th axis*

.

5 Conclusion and future works

In this paper, a reference architecture for robot controller has been illustrated. Our major goals includes the definition of a modular architecture which can be easily reconfigured with reference to the kinematics of the controlled robot and which can be easily applied to two different RTOS, i.e. QNX 4.25 and Linux RTAI 3.2. Specifically, such an architecture has been implemented with reference to two different robots, the former prototyped by ITIA-CNR, the latter manufactured by Mitsubishi Robot. With reference to the Mitsubishi robot, experimental results are shown to validate both RT and control performance. Future works include the use of the control architecture to test new control algorithms for robots.

References

[1] G. Pritschow, Y. Altintas, F. Jovane, Y. Koren, M. Mitsuishi, S. Takata, H. Van Brussel, M. Weck, K. Yamazaki, 2001, *Open Controller Architecture - Past, Present and Future*, Annals of the Cirp, vol. 50/2/2001.

[2] The Orocos Project. http://www.orocos.org, 2003.

[3] D. Dallefrate, E. Carpanzano, L. Molinari Tosatti, F. Jovane, *Feed rate optimization technique for high-speed CNC machining with parallel manipulators*, The 3rd Chemnitz parallel Kinematics Seminar and 2002 Parallel Kinematic Machines International Conference; April 23-25, 2002, Chemnitz, Germany.

[4] G. Pritschow, G. Rogers, G. Bauer, M. Kremer*Open Controller Enabled by an Advanced Real-Time Network*. CIRP 2nd International Conference on Reconfigurable Manufacturing, Ann Arbor, MI, USA, August 20-21 2003.

[5] L. Dozio, P. Mantegazza, *Linux Real Time Application Interface (RTAI) in low cost high performance motion control*, Motion Control 2003, a conference of ANIPLA, Associazione Nazionale Italiana per l'Automazione (National Italian Association for Automation), Milano, Italy, 27-28 March 2003.

[6] http://www.rtai.org

[7] http://www.qnx.com

[8] S. Negri, *Analysis and Design of a Reconfigurable Machine for Assembly Operations*, 10th International Workshop on Robotica in Alpe Adria Region (RAAD), 16-18 Maggio 2001.

[9] D. Colombo, F. Jatta, L. Molinari Tosatti, *Reconfigurable control strategies applied to a reconfigurable PKM* 3rd CIRP Conference on Reconfigurable Manufacturing, Ann Arbor, USA, May 10-12, 2005

[10] M. Malosio, M. Finardi, S. Negri, L. Molinari Tosatti, F. Jatta, *A Modular Architecture for High-Level Robot Programming and Control in a PC-Based Environment* IEEE International Conference on Robotics and Automation (ICRA), New Orleans, 2004.

[11] http://tech-www.informatik.uni-hamburg.de/personal/westhoff/en/robotics/robotics.html