A Five-layers Open-Architecture Robot Controller Applied to Interaction Tasks

Article · October 2008

DOI: 10.1109/LARS.2008.9 · Source: InTech

4 authors, including:

Edson Roberto De Pieri
Federal University of Santa Catarina
142 PUBLICATIONS **1,008** CITATIONS

SEE PROFILE

Daniel Martins
Federal University of Santa Catarina
260 PUBLICATIONS **1,190** CITATIONS

SEE PROFILE

Ubirajara Moreno
Federal University of Santa Catarina
91 PUBLICATIONS **386** CITATIONS

SEE PROFILE

5

An Open-architecture Robot Controller applied to Interaction Tasks

A. Oliveira[1], E. De Pieri[2] and U. Moreno[2]
[1]*Mechanical Engineering Department*
University of Caxias do Sul (UCS)
[2]*Department of Automation and Systems*
Federal University of Santa Catarina (UFSC)
[1-2]*Brazil*

1. Introduction

Many current robotic applications are limited by the industry state of art of the manipulators control algorithms. The inclusion of force and vision feedbacks, the possibility of cooperation between two or more manipulators, the control of robots with irregular topology will certainly enlarge the industrial robotics applications. The development of control algorithms to this end brings the necessity of using open-architecture controllers.
Generally the robotic controllers are developed for position control, without accomplishing integrally the requirements of tasks in which interactions with the environment occur. Therefore, this is currently one of the main research areas in robotics, e.g., in (Abele et al., 2007) is presented the identification of characteristics to an industrial robot to execute machining applications. To consider this interaction the robot controller has to give priority to the force control time response, because in the instant of end-effectors contact with the surface, several forces act on the system. Depending on the speeds and the accelerations involved in the process, damages or errors can occur. To avoid these effects, compliances are inserted in tool or in surface of operation.
A new reference model for a control system functional architecture applied to open-architecture robot controllers is presented. Where, this model is applied for integrally development of a five-layer based open-architecture robotic controller for interaction tasks, which uses parallel and distributed processing techniques, avoiding the necessity of compliance in system, allowing a real-time processing of the application and the total control of information. This architecture provides flexibility, the knowledge of all the control structures and allows the user to modify all controller layers. The used controller conception aims to fulfill with the following requirements: high capacity of processing, low cost, connectivity with other systems, availability for the remote access, easiness of maintenance, flexibility in the implementation, integration with a personal computer and programming in high level.

This chapter is organized as follows. Section 2 overviews the most relevant categories, definitions and requirements of robot controllers. Section 3 details the reference model for open-architecture controller development. Section 4 describes the robot retrofitting for interactions tasks. Section 5 presents and discusses the experimental setup. Finally, Section 6 concludes the chapter and outlines future research and development directions.

2. Open-architecture Robot Controllers

Various open control architectures for industrial robots have already been developed by robot and control manufacturers as well as in research labs. In (Lippiello et al., 2007) is presented an open architecture for sensory feedback control of a dual-arm industrial robotic cell for cooperation tasks. In (Macchelli & Melchiorri, 2007) is presented a real-time control system based on RTAI-Linux operating system and developed for coupling of an advanced end-effector. (Hong et al., 2001) develop a system of robot open control based on a reference model OSACA. (Bona et al., 2001) propose a real-time architecture for robot control system development based in real-time operating system for embedded systems, RTOS. In (Donald & Dunlop, 2001) present a retrofitting of a path control system for a hydraulic robot based on a FPGA executing the embedded operating system RTSS. The inexistence of a standard methodology for architecture controller project difficult the development of high-openness degree control system.

Most of the existing robot control open architectures are based on a standard PC hardware and a standard operating system, because I/O boards and communication boards for robots have a higher cost in relation to the similar boards for PCs. Another reason is the lack of standardization of robot peripherals, with each manufacturer developing its own protocols and interfaces, forcing the users to buy all the components of a single manufacturer (Lages et al., 2003). Additionally, a PC based controller can be integrated more easily with many commercially available add-on peripherals such as mass storage devices, Ethernet card and other I/O devices. So, the facility to integrate other functionalities is a strong reason to use a standard PC hardware in robot control open architectures.

Another reason is that the robot programming languages are, at low level, more similar to the Assembly languages than to the modern high level languages and this may difficult implementations (Lages et al., 2003). In a PC based controller standard software development tools (e.g., Visual C++, Visual Basic or Delphi) can be used.

2.1 Definitions

The definition of open system, according to Technical Committee of Open Systems of IEEE is "An open system provides capabilities that enable properly implemented applications to run on a variety of platforms from multiple vendors, interoperate with other system applications and present a consistent styler of interaction with the user". A open-architecture control system has the capacity to operate with the best components of different manufacturers. What makes possible the easy integration of new system functionalities.

From user point of view, the "openness" of the systems consists in capabilities to integrate, extend and reuse software modules in control systems (Lutz & Sperling, 1997). In (Pritschow & Altintas, 2001) and (Nacsa, 2001) the "degree of openness" of a system is defined by some criteria, as:

- ***Extendibility***: A variable number of modules can be executed simultaneously in a same platform, without causing conflicts, i.e., this characteristic depends mainly on the operating system, that should accomplish a multi-task processing, and also of modules coupling level, that should allow those operations.
- ***Interoperability***: The modules work together efficiently and they can interchange data in a defined way through logical and physical communication buses.
- ***Portability***: The modules can be executed in different platforms without modifications, maintaining their functionalities, i.e., they should conform software and hardware standards to maintain the system compatibility with other platforms.
- ***Scalability***: Depending on the user requirements, the module functionalities and performance and size of the hardware, software and firmware can adapt for the system optimization.

Those characteristics define the "degree of openness" of a system, how more extended and refined, major will be the level of openness. For open-architecture controllers, one more characteristic should be considered, the modularity.

- ***Modularity***: The system is divided in specialized subsystems, called modules, that can be substituted without significant modifications in system. This characteristic consists of logical and physical system decomposition in small functional units.

2.2 Categories

The controllers are characterized by the freedom of access information or simply for "degree of openness". Usually, the control of several system modules (e.g., unit power and low level control) is proprietary and cannot be modified by user, other levels are considered open (e.g., communication interface and high-level control), i.e., they are based on hardware and software standards with specifications of open interface.

In (Pritschow & Altintas, 2001), (Lutz & Sperling, 1997) and (Ford, 1994), the "degree of openness" of a system is defined in agreement with access concept to controller layers, like this, they can be classified in three categories:

- ***Proprietary***: That system modality allows the access just to application layer, being therefore, a closed system. In those systems is extremely difficult or impossible the integration of external modules.
- ***Hybrid or Restricted***: That category makes available the access to application layer and a controlled access to operating system module. The operating system has a fixed topology, however, allows small changes in control system modules (e.g., gains and parameters).
- ***Open***: Open-architecture systems allow integral access of application layers and operating system modules, supplying a homogeneous vision of the system, allowing the manipulation and modification of all modules that compose the system. Its offers interchangeability, scalability, portability and interoperability.

2.3 Requirements

One of main requirements for a system to be characterized with open-architecture is the necessity of the control functionalities be subdivided in small functional units with a solid relationship among the subsystems. Consequently, the modularity becomes fundamental for a control system to have an open-architecture (Pritschow & Altintas, 2001).

The determination of module complexity should consider factors as the "degree of openness" wanted and integration cost. Small modules supply a high-level openness, but they increase the complexity and integration costs. A low modularity can drive for a high demand of resources and to deteriorate the system performance, not allowing real-time data articulation (Nacsa, 2001).
The system structuring through a modular interaction requests a detailed group of relationship methods, composed by Application Programming Interfaces (i.e., these are a group of routines and software standards for extern access of their functionalities). In open control systems these interfaces need to be standardized (Pritschow & Altintas, 2001).
The modular platforms encapsulate the operation system specific methods absorbing the hardware, operating system and communication characteristics. What promotes a high level data exchange, this abstraction requests a data mediation module, called middleware. These data concatenation and adaptation points increase the portability and interoperability of distributed applications in heterogeneous environments.

3. The Reference Model for Open-Architecture Robot Controllers

The reference model for a control system functional architecture presented in (Sciavicco & Siciliano, 2000) has a priority focus in the control structure, little exploring the other levels of robot controllers.
This work proposes a new reference model for a control system functional architecture applied to open-architecture robot controller. The model is based on model of (Sciavicco & Siciliano, 2000), however, it expands the approach for all controller levels, adapts their layers in agreement with the standard ISO 7498-1 and considers the definition, categories, requirements and tendencies for open-architecture controllers. The structure of the proposed reference model is represented in Fig. 1, where the five hierarchical levels are illustrated. To proceed, those layers will be described individually.

3.1 Task Layer

The task layer is responsible for industrial robot control tasks grouped in three categories: trajectory planning, supervisory system and control law. Those operations are processed in the central equipment of the system, usually a personal computer (PC). In remote control operations, the operations can be divided in two software modules with relationship client-server. The trajectory planning and supervisory system will be processed with smaller time requirements in client, while the control structure will be processed in real time of application in server.

3.2 Integration Layer

The adopted functional architecture hierarchical structure, together with its articulation into different modules, suggests a hardware implementation that exploits distributed computational resources interconnected by means of suitable communication channels. At the integration layer, the information adaptation is accomplished (i.e., concatenation and organization) incoming from several processors that compose the distributed system.

These operations supply to superior layer a heterogeneity vision of the system to sharing resources. In this level, peripherals with high-level of abstraction (e.g., exteroceptive sensors) are also appropriate in this level.

3.3 Communication Layer

At the communication layer, the interconnection of information among the system processors is accomplished, usually using high-speed data transmission buses. The network topology is indifferent, however, it is important the use of redundant ways for connection among all intermediate points and the net central knot through the main bus and the embedded systems interconnection by an alternative communication bus. Every system interconnection accomplished in this layer is based in International Standard ISO/IEC 7498-1.

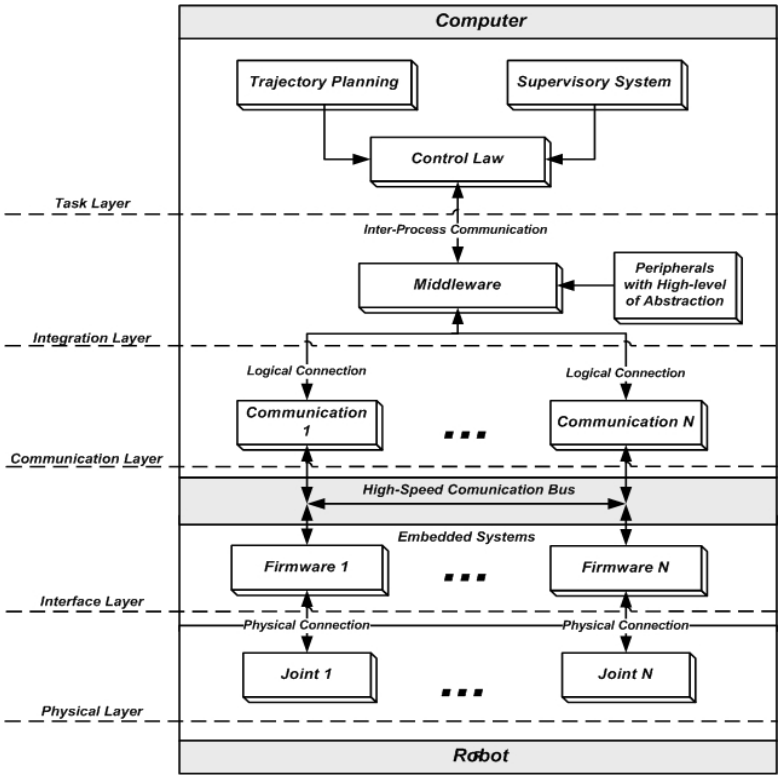

Fig. 1. Functional architecture of proposed reference model.

3.4 Interface Layer

The interface layer is composed by the embedded systems, i.e., dedicated hardware's to process specific task software (called firmware) encapsulated in internal storage memories. This organizational structure divides the system in small hardware modules and consequently, distributes the system processing. The processing distribution degree is

proportional at the utilization level of dedicated processors in system. The system decomposition in task dedicated processors guarantees a fixed and minimum response time.

3.5 Physical Layer

The industrial manipulator physical access (i.e., actuators and proprioceptive sensors) occur in physical layer, composed only by the input and output robot data channels. Usually, the actuator activation is realized indirectly, because, the controller signs only access the unit power that adapts this signs for the motors.

4. Reference Model applied to Interaction Tasks

Special requirement for robot controllers that includes force control

Generally the robotic controllers are developed for applications that require only position control, and the robot end effector doesn't contact the workspace during its movement. The interaction concept is related with the contact between robot and environment, where generated force and torque profiles need to be controlled. In applications that need force control, the end effector contacts some surface in its workspace and this interaction generates contact forces that must be controlled in a way to fulfill the task correctly, without damaging both, robot tools and the working objects.

The contact force intensities, originated by tool movements commanded by the robot controller, depend on both, the tool rigidity and the object surface rigidity, and they must be also controlled. A small tool movement could originate large force intensities in case the tool and the object surface rigidity are large. It should be noted that by introducing compliance to the tool we generate a delay in the application of the forces and this could be unacceptable in some applications. Consequently, the system should have a small time response to these forces, to prevent tool, robot or object damages. The use of high performance systems is a requisite of controllers for application of force control.

Therefore, the reference model proposed was applied, considering the interaction tasks requirements, for retrofit of old industrial manipulator. The resultant functional structure for controller is presented in Fig. 2 and described as follows.

4.1 Task Layer for interaction tasks

The task layer has a mathematical environment prepared to make operations with matrices in which the control law is stored. The information proceeding from the n joints are available in matrices *nx1* corresponding to the position vector q, and the velocity vector $\dot{q}$, where the lines represent the joints. The force sensor data are stored in a matrix *6 x 1* called $h = [f_x\ f_y\ f_z\ \mu_x\ \mu_y\ \mu_z]^T$, which contains forces and moments data. The information to be directed to the motors and encoders is stored in an *n x 3* control matrix u. In this layer the user develops the control laws of position and/or force of the manipulator and it is possible to carry through the task simulation.

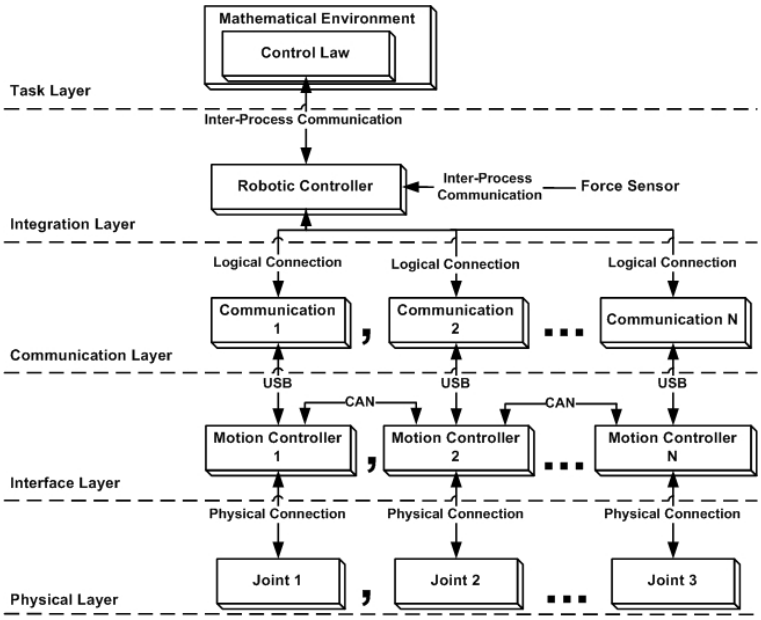

Fig. 2. Functional architecture of proposed reference model applied to interaction tasks.

4.2 Integration Layer for interaction tasks

In the integration layer the concatenation and the organization of all the information coming from the sensors and to be sent to the superior layer are done. In case of the inclusion of a new hardware to the system, it is necessary to add its control structure to this layer. This is carried through by a high-level application that manages the power unit and control unit. Preventing any irregular movements and danger situations and controlling the components of the lower level. In this software the controller's components can be activated or disabled independently.

4.3 Communication Layer for interaction tasks

The communication layer controls the data transfer by managing the interface USB (Universal Serial Bus 2.0) and the industrial protocol CAN (Campus Area Network), both high performance communication devices. The USB makes a system interconnection through a star form topology, which has the computer as a central knot. Each USB door supports up to 127 devices and, in this manner, it is possible to connect a great quantity of joints to the controller. The protocol CAN form the bus between the secondary knots (motion controllers) and the result structure is a redundant net architecture. The implementation of this bus is still being explored and intends to introduce the possibility of a joint to access information of another joint without passing through the central knot. This will increase the performance of the net and it gives the opportunity to an implementation of the system of control without the central knot: a totally embedded control. The resultant architecture communication is presented in Fig. 3.

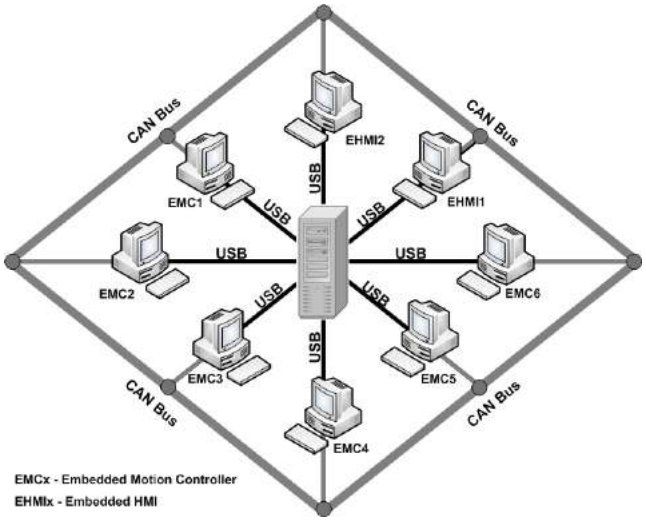

Fig. 3. Communication architecture for interaction tasks.

4.4 Interface Layer for interaction tasks

The interface layer comprises the embedded systems that carry out the control of the robotic joints, named motion controllers. Each of these motor digital controllers decodes the corresponding encoder signal and generates the modulation width pulse (PWM) to the control of the respective motor. Each of these systems has an optical isolated interface to prevent any inadequate return to the processor. It possesses a great amount of expansion doors, which allows the connection of other tools.

We developed the controller with a modular architecture to have an independent control for each joint and so, divide the mathematical complexity among the processors of the system. This results in a distributed processing organized by the central knot (computer), where the operations occur in parallel. This methodology facilitates the expansion and maintenance of the system.

Currently the system operates with a medium tax of update of the signals of 1 ms, only for a convention of literature. In case of necessity this largeness can be diminished.

4.5 Physical Layer for interaction tasks

The most inferior layer, here denominated physical layer, is the power unit of the motors and the angular position sensors.

5. Experimental Environment

The retrofitting methodology was validated with the adaptation of an old anthropomorphic manipulator, model *Rv15*, produced by the *REIS Robotics*, for interaction tasks. Where was substituted the proprietary controller by the new open-architecture controller and coupled a force sensor in system.

The *REIS Rv15* robot has six rotating joints acted by electric motors and the angular positions measurement are done using incremental optical encoders. It is a manipulator with a topology that is very common in industry applications, which constitutes an anthropomorphous arm (joints 1, 2 and 3) with a spherical wrist (joints 4, 5 and 6).
The Fig. 4 presents a complete diagram of the embedded five-layer open-architecture robotic controller for an industrial manipulator, containing it data flow and the systems interconnections

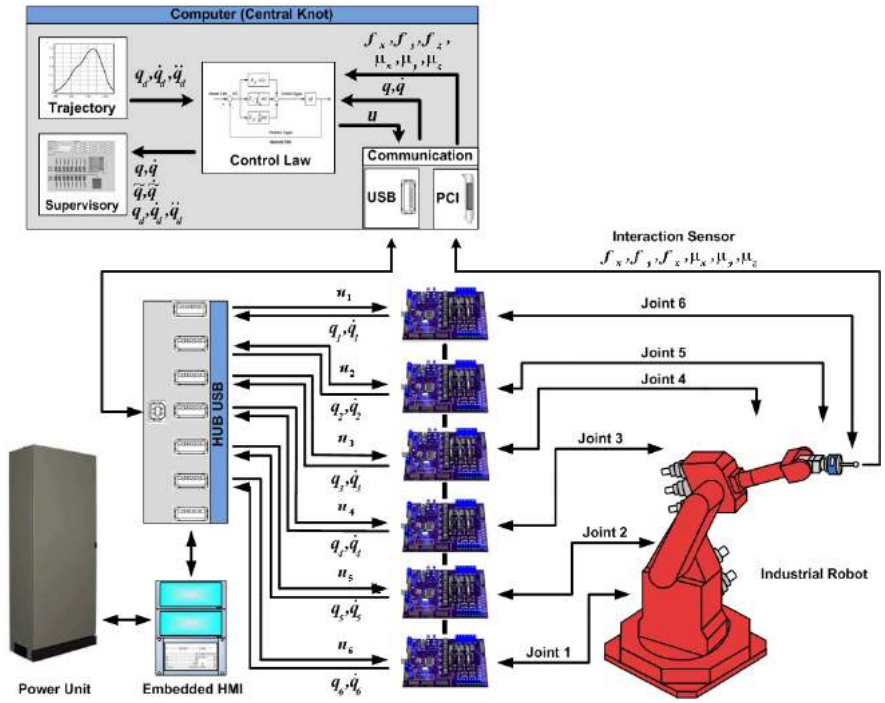

Fig. 4. Experimental environment for interaction tasks.

5.1 Hardware architecture description

The system's hardware was developed and built using high performance and reliability, low cost and easiness to be found in the market components. The Fig. 5 shows the diagram of internal blocks used in the motion controllers.
The main component is a digital signal controller (*DSC*) produced by the *Microchip Technology Inc*. named dsPIC30F6010A. It operates with 16-bits, in a 120 MHz frequency with a package TQFP of 80 pins, and is one integrant of the family of the motors control. It possesses a great amount of well differentiated modules including an ample program memory with a 144K and a non-volatile memory with 4096 bytes for information storage. It has 16 ways for A/D conversions and the necessary modules of communication. For the communication through USB we used a component which carries through the conversion of module UART for the bus. This component supports transference taxes up to 3 Megabaud and is manufactured by the *FTDI* (*Future Technology Devices International Ltd*).

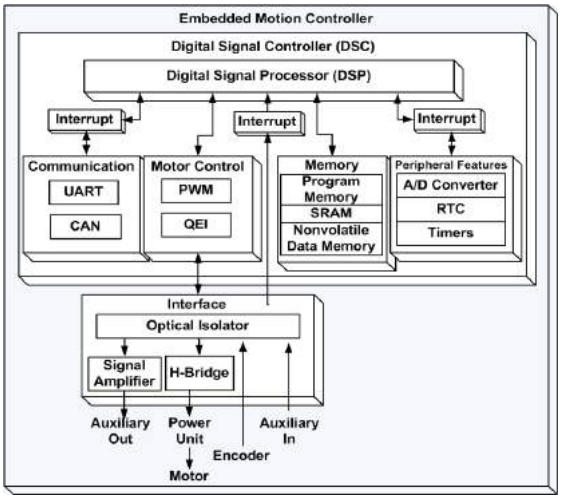

Fig. 5. Interface architecture for interaction tasks.

Moreover, it possesses other functionalities, including the generation of a digital external signal oscillator with changeable frequencies. Besides this, the same manufacturer produces available Royalty-Free drivers for many operational systems, for this form of implementation. To implement the requirements for the physical layer defined by the ISO-11898, we connect the CAN industrial protocol to a transceiver of high speed, which supports until 1Mb/s.

The system firmware implementation uses the high level language C. This is completely modulated and organized in units, to facilitate modifications. All the modules operate with interruptions of the processor with distinct priorities, such that an operation of less priority doesn't delay a higher importance process.

The module of motion controller is also composed by a 16 bits PWM generator and by a module to read the quadrature encoder (named QEI), which we extend for 32 bits. Connected to it there is an optical decoupling barrier and an H-bridge for the control of the power unit that supports 100V and 8A. There are also amplified auxiliary output channels, which operate until 100V and 6A. To protect the system, the encoder entrances and the auxiliary inputs also have been connected to optical decoupling barrier. Internally there is still a great amount of resources that had not been used and that can be useful in future upgrades of the system.

The experimental validation of the proposal was accomplished with implementation of a indirect force control strategy, the impedance control that will be presented to follow.

5.2 Firmware architecture description

The real time processing is obtained with the modularity of the embedded controller. These modules communicate only through hardware interrupts with eight priority levels. The software architecture and realistic physical modelling of the sensors and actuators provided to system a high response time.

The servo motors are controlled through a embedded self-tuning PID controller that uses the linear actuator dynamic model. The Fig. 6 presents the validation of dynamic model.

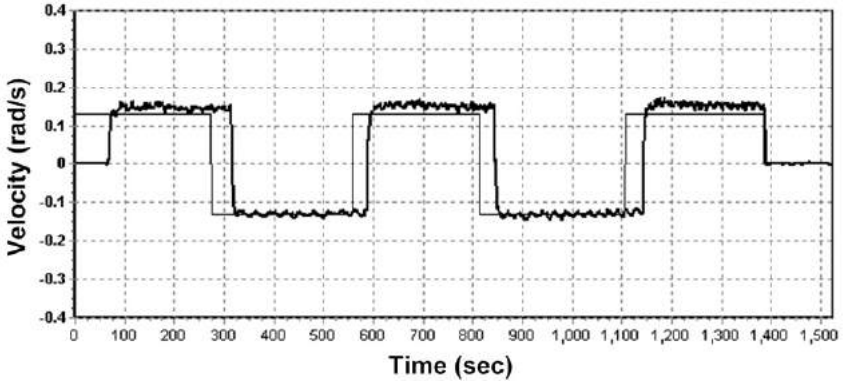

Fig. 6. Validation of the dynamic model behaviour.

5.3 Software architecture description

The software was developed using a high level object-oriented language (C++), for the Linux operating system recompiled for real-time application interface (RTAI). The control system monitors the processor activity, because most processes works through threads.
Like this, when the processor activity reaches a critical level, the threads priorities are altered favouring controller essential tasks.

5.4 Impedance control

The manipulator control strategies for interaction tasks are grouped usually in two categories: indirect force control and direct force control. The first approaches the movement with an implicit force feedback only based in movement control, the other supplies the possibility of force control for a wanted value, through an explicit force feedback (Sciavicco & Siciliano, 2000).
The classic impedance control approaches contact force indirectly modeling the interaction as a mass-spring-damper system. The indirect relationship is a consequence of force sign influence on control law, the objective is to adapt the manipulator dynamic behavior in contact with the environment and not to fulfill a position and/or force trajectory. This way, an explicit force feedback doesn't exist in system, because, this signal just supplies the system impedance in contact with a surface.
Therefore, the fundamental philosophy of impedance control, in agreement with (Hogan, 1985), is that the control system regulates the manipulator impedance, that is defined by relationship between the speed and applied force, Fig. 7 (Zeng & Hemami, 1997). The formulation of this control strategy is presented to follow, in the equation 1.

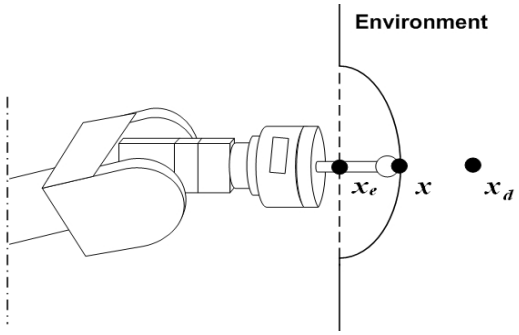

Fig. 7. Impedance force control.

$$M_D\ddot{\tilde{x}} + K_D\dot{\tilde{x}} + K_P\tilde{x} = h_A \tag{1}$$

Where, M_D is the mass matrix, K_D is the damping matrix, K_P is the stiffness matrix and h_A is the interaction force matrix.

Equally to position control for inverse dynamics, where the impedance control is based, the integral knowledge of manipulator dynamics is admitted. In this way, the accurate knowledge of object elasticity characteristics or contact environment is not necessary in this control strategy (Yoshikawa, 2000).

5.5 Results

The implemented control strategy uses force feedback only to regulate the manipulator impedance, assuming that the manipulator is in contact with operation surface. In this way, when some force be detected the control law only will regulate the impedance to establish the system.

The application used to validate the developed controller for interaction tasks is based in this characteristic of the impedance control, however, in this case, the end-effector isn't in contact with the surface. Therefore, the manipulator is immobile, admitting to be in wanted impedance profile, and when detects external force controls the system impedance (Fig. 8). The joint speed profiles generated in experiments are present in Fig. 9.

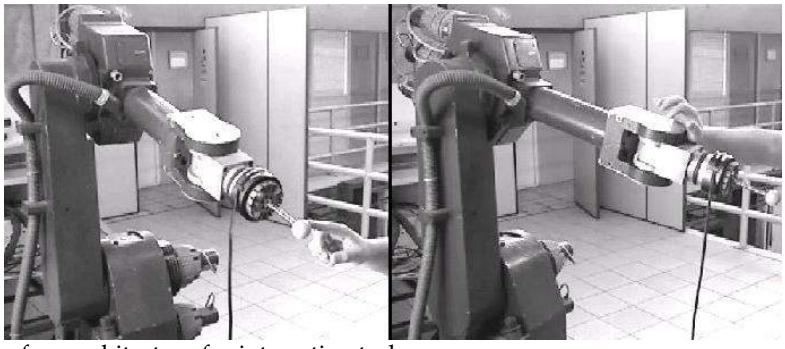

Fig. 8. Interface architecture for interaction tasks.

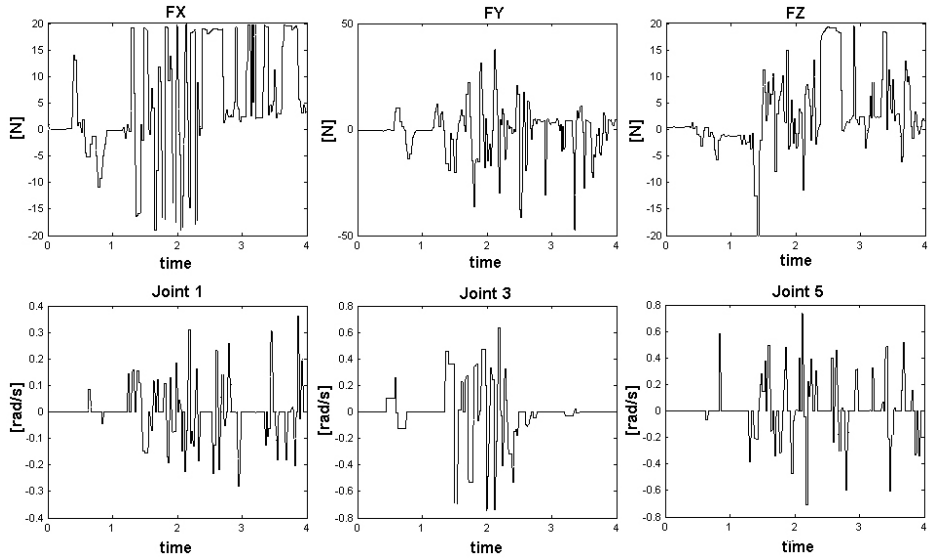

Fig. 9. Force and joint velocities profile of the impedance control.

The system validation experiment accomplished consists in a parallel manipulation on robot workspace limits. The force profiles, detected by *JR3* force and moments sensor, only adapt manipulator dynamic behaviour with environment dynamic characteristics. In this way, a profile of force control is used to drive the joints velocity behaviour to a desired impedance profile.

6. Conclusion

In this work was considered a new reference model for a control system functional architecture applied to open-architecture robot controllers. The proposed approach was applied for integrally developing of a five-layer based open-architecture robotic controller considering interaction tasks. The architecture uses parallel and distributed processing techniques and circumvents the necessity of compliance in system, allowing a real-time processing of the application and the total information control.

Old manipulator retrofitting considers the problem of including controllers with new functionalities as force control. The main characteristics of these systems are high-stiffness and position control. These characteristics restrict response time of the system. Therefore, an open-architecture system can be projected to operate in real-time.

The proposed reference model for open-architecture robot controllers was experimentally validated including the implementation of an indirect force control strategy in the robot controller. Practical tests have shown the interest of the proposed architecture in terms of controller flexibility, costs and maintenance and high capacity of processing.

This reference model clarifies the concept of robot controller and explains the internal modules that compose robot control unit. The system decomposition makes possible the optimization of internal modules for a specific task, e.g., interaction tasks. In this way, it is

possible to include new functionalities to the system, e.g., other feedback signals, new actuators or dedicated processors for a specific problem, e.g., resolution of redundancy or inverse kinematics.
In the actual stage, the researchers have been focused on the theoretical aspects of the problem. Further works will consider the model validation and experimental applications.

7. References

Abele, E.; Weigold, M. & Rothenbücher, S. (2007). Modeling and identification of an industrial robot for machining applications, *CIRP Annals- Manufacturing Technology*, Vol. 56, No. 1, page numbers 387–390.

Bona, B.; Indri, M. & Smaldone, N. (2001). Open system real time architecture and software design for robotcontrol, *Proceedings 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Vol. 1.

Donald, S. & Dunlop, G. (2001) Retrofitting path control to a unimate 2000b robot, *Proccedings 2001 Australian Conference on Robotics and Automation*, Vol. 14, page number. 15.

Ford, W. (1994). What is an open architecture robot controller?, *Proceedings of the 1994 IEEE International Symposium on Intelligent Control*, page numbers 27–32.

Hong, K.; Choi, K.; Kim, J. & Lee, S. (2001). A pc-based open robot control system: PC-ORC, *Robotics and Computer Integrated Manufacturing*, Vol. 17, No. 4, page numbers 355–365.

Hogan, N. (1985). Impedance Control: An Approach to Manipulation, Parts I-Ill, *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 107, No. 1, page numbers 1–24.

Lages, W.; Henriques, R. & Bracarense, A. (2003). Arquitetura aberta para retrofitting de robôs, *Manet Notes Workshop*, Bragança Paulista, SP, Brazil.

Lippiello, V.; Villani, L. & Siciliano, B. (2007). An open architecture for sensory feedback control of a dual-arm industrial robotic cell, *Industrial Robot: An International Journal*, Vol. 34, No. 1, page numbers 46–53.

Lutz, P. & Sperling, W. (1997). Osaca the vendor neutral control architecture, *Proceedings European Conference Integration in Manufacturing*, page numbers 247–256.

Macchelli, A. & Melchiorri, C. (2002). Real time control system for industrial robots and control applications based on real time Linux, *15th IFAC World Congress*, page numbers 21-26, Barcelona, Spain.

Nacsa, J. (2001). Comparison of three different open architecture controllers, *Proceedings of IFAC MIM*, page numbers 2–4, Prague.

Pritschow, G. ; Altintas, Y.; et al. (2001). Open controller architecture–past, present and future, *CIRP Annals-Manufacturing Technology*, Vol. 50, No. 2, page numbers 463–470.

Sciavicco, L. & Siciliano, B. (2000). Modelling and Control of Robot Manipulators. Springer.

Yoshikawa, T. (2000). Force control of robot manipulators, *Proceedings. ICRA '00 IEEE International Conference on Robotics and Automation*, Vol. 1, page numbers 220–226.

Zeng, G. & Hemami, A. (1997). An overview of robot force control, *Robotica*, Vol. 15, No. 05, page numbers 473–482.