Flexible Software for Industrial Robots

António Ferrolho[1, 2] and Manuel Crisóstomo[2]

[1] Department of Electrical Engineering
Superior School of Technology of Viseu
Polytechnic Institute of Viseu
Campus Politécnico de Repeses
3504-510 Viseu – Portugal
antferrolho@elect.estv.ipv.pt

[2] Institute of Systems and Robotics
Department of Electrical and Computer Engineering
University of Coimbra
Polo II
3030-290 Coimbra – Portugal
mcris@isr.uc.pt

***Abstract*-This paper describes flexible software for industrial robots. *WinRS232ROBOTcontrol* and *winEthernetROBOTcontrol* software were developed to be used in industrial robots. With this software, industrial robots can be integrated in modern production systems, in an easy and efficient way. A Robotic Bar and a Flexible Manufacturing Cell (FMC) were developed with the objective of showing the potentialities of the developed software.**

I. Introduction

This paper presents and describes several flexible software applications developed for industrial robots: *winRS232ROBOTcontrol* and *winEthernetROBOTcontrol*. The developed flexible software uses the original capacities of the control system of the robots, in a distributed client/server environment. A Robotic Bar and a Flexible Manufacturing Cell (FMC) were developed with the aim of testing the potentialities of the proposed software in industrial applications.

The work described in this paper allows:

- Remotely controlling and monitoring all of the functionalities of the robots;
- Developing a distributed software architecture based on personal computers using Ethernet networks;
- Developing applications that allow remote exploration of robots using always the same programming languages (e.g., C++);
- Using a personal computer as programming environment;
- Developing graphical user interfaces (GUI) for robots in a simple and efficient way.

II. Industrial Robots Software

Nowadays, the industrial robot controllers have the following important limitations [1]:

- Some robot controllers only can use RS232 connections, and others can support Ethernet networks;
- Robots manipulators have closed controllers and are essentially position-controlled devices that can receive a trajectory and run it continuously;
- Many robot controllers do not allow remote control from an external computer;
- Robot programming environments are not powerful to develop tasks requiring complex control techniques (e.g., integration and control of robots in Flexible Manufacturing Cells);
- Robot manipulators from different manufacturers have their own programming language and environments, making the task of integration and cooperation difficult.

To reduce several of these limitations, the *winRS232ROBOTcontrol* and *winEthernetROBOTcontrol* integration software for industrial robots were developed. This flexible software for industrial robots allows [1]:

- Developing a distributed software architecture based on personal computers using Ethernet networks;
- Developing applications that allow remote exploration of robots manipulators using always the same programming languages (e.g., C++);
- Using a personal computer as programming environment, taking advantage of the huge amount of programming and analysis tools available on these platforms;
- Developing a GUI, in a simple and efficient way.

A. *winRS232ROBOTcontrol*

The *winRS232ROBOTcontrol* was developed to be used in robots manipulators where the communication with the controller is only possible through the RS232 serial port. With the *winRS232ROBOTcontrol* it is possible to develop robot programs in C++, and run them at PC level, not at robot's controller level. Nevertheless, the *winRS232ROBOTcontrol* also allows the development of programs to be executed at a robot's controller level. Furthermore, it is also possible to have programs running in the PC and in the robot's controller (mix solution), individually or simultaneously. The *winRS232ROBOTcontrol* was developed for industrial robot manipulators, and was implemented and tested in two different robots: Scorbot ERVII robot and Mitsubishi RV-3SB robot.

ROBOCOMM 2009, 31st Mar – 2nd Apr 2009, Odense, Denmark.
Copyright © 2011 – 2012 ICST ISBN 978-963-9799-51-6
DOI 10.4108/ICST.ROBOCOMM2009.5963

Nevertheless, *winRS232ROBOTcontrol* was designed to work with all robots that support RS232. But, of course, whenever we use different robots, it is necessary to make some upgrades in the *winRS232ROBOTcontrol*, according to the used robot.

The developed API, for the *winRS232ROBOTcontrol* application, is based on a thread process running simultaneously with the main program. The RobotControlAPI library was developed with access to the controller's functions in mind, as shown in Fig. 1. This API allows us to communicate with the robot's controller. The available functions are divided into two groups. The first contains public methods and the second contains methods representing events. Table I shows some of these functions.

An Ethernet server was developed for the computer that controls the robot. Thus, it is possible to control the robot remotely, in a distributed client/server environment, as shown in Fig. 1.

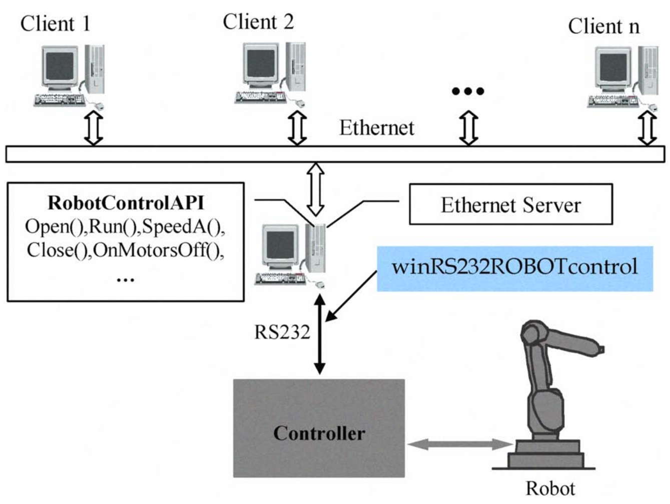


Fig. 1. winRS232ROBOTcontrol.

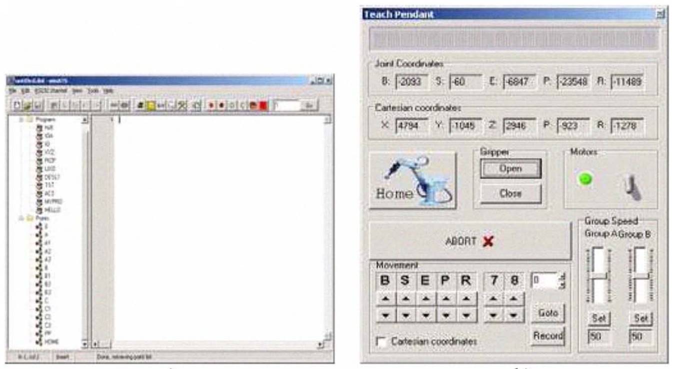


Fig. 2. winRS232ROBOTcontrol windows and teach pendant.

Fig. 2(a) shows the *winRS232ROBOTcontrol* window and Fig. 2(b) presents the developed teach pendant. The *winRS232ROBOTcontrol* window has some toolbars with a set of icons. These icons can be used, for example, to switch the motors on and off, to open and close the gripper, to home the robot, to open and create files, and so on. The teach pendant has multiple functions, such as, abort programs, moving the robot, switching the motors on and off, saving the robot's locations, homing, and so on.

TABLE I
SOME AVAILABLE FUNCTIONS

	Function	Description		Function	Description
Public Methods	SpeedA()	Changes group A speed	Events	OnEndHoming()	This event activates after homing
	Home()	Homing the robot		OnClose()	This event activates after the execution of the Close() function
	MotorsOff()	Switches off the robot's motors		OnOpen()	This event activates after the execution of the Open() function
	MotorsOn()	Switches on the robot's motors		OnMotorsOff()	This event activates after the motors are switched off
	Close()	Closes the gripper		OnMotorsOn()	This event activates after the motors are switched on
	Open()	Opens the gripper			

B. *winEthernetROBOTcontrol*

The *winEthernetROBOTcontrol* was developed to be used in robots manipulators where the communication with the controller is possible through an Ethernet network. The *winEthernetROBOTcontrol* has a Dynamic Link Library (DLL) and allows development of simple and fast applications for industrial robots. The developed flexible software uses the original capabilities of the robot's controllers, in a distributed client/server environment. Fig. 3 shows the PC1 and PC4 communicating with the robots through *winEthernetROBOTcontrol*.

Table II presents same procedures, functions and events developed for the *winEthernetROBOTcontrol*. The procedure "Robot" allows making a connection between one PC (with *winEthernetROBOTcontrol*) and one or more robots. For that, it is necessary to create and configure one connection between the PC and the robot controller (Add New Alias), as shown in Fig. 4(a). After that, it is possible to communicate and control the robot remotely through the connection (Alias).

Fig. 4(b) shows one small example developed in C++ with the objective of showing the potentiality of the *winEthernetROBOTcontrol*. This code allows running "Load_M.prg" program in the IRB-1400 robot. The use of the developed *winEthernetROBOTcontrol* allows the integration and control of robots in modern systems (e.g., FMC), and uses the original capacities of the control systems of the robots. For example, the "Load_M.prg" program is done with RAPID software (ABB software to program only ABB robots) and the *winEthernetROBOTcontrol* can work together with this program. With *winEthernetROBOTcontrol* it is possible to develop a software control, for one or more robots, in any PC. With this PC, it is possible to communicate and control the robot (or robots) through an Ethernet [see Fig. 3].

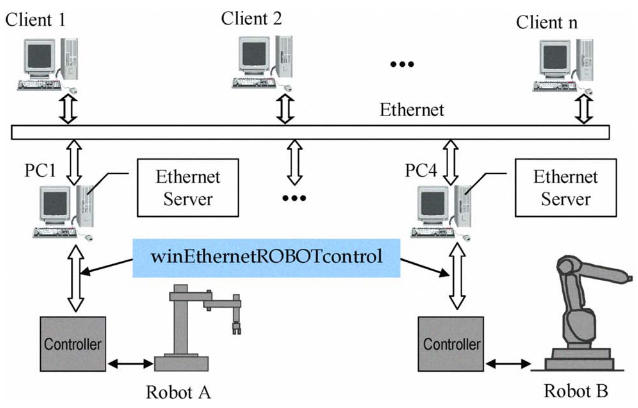
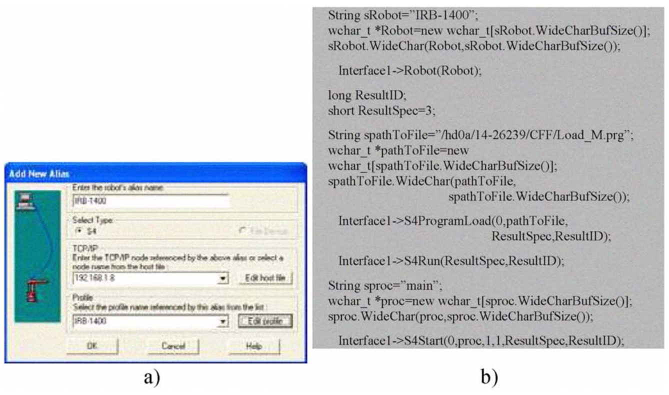


Fig. 3. winEthernetROBOTcontrol.



Enter the robot's alias name: IRB-1400
Select Type: (•) S4
TCP/IP
Enter the TCP/IP node referenced by the above alias or select a node name from the host file: 192.168.1.8
Profile
Select the profile name referenced by this alias from the list: IRB-1400

```
String sRobot="IRB-1400";
wchar_t *Robot=new wchar_t[sRobot.WideCharBufSize()];
sRobot.WideChar(Robot,sRobot.WideCharBufSize());

  Interface1->Robot(Robot);

long ResultID;
short ResultSpec=3;

String spathToFile="/hd0a/14-26239/CFF/Load_M.prg";
wchar_t *pathToFile=new
wchar_t[spathToFile.WideCharBufSize()];
spathToFile.WideChar(pathToFile,
                        spathToFile.WideCharBufSize());

  Interface1->S4ProgramLoad(0,pathToFile,
                                  ResultSpec,ResultID);

  Interface1->S4Run(ResultSpec,ResultID);

String sproc="main";
wchar_t *proc=new wchar_t[sproc.WideCharBufSize()];
sproc.WideChar(proc,sproc.WideCharBufSize());

  Interface1->S4Start(0,proc,1,1,ResultSpec,ResultID);
```

Fig. 4. winEthernetROBOTcontrol.

TABLE II
SOME AVAILABLE PROCEDURES, FUNCTIONS AND EVENTS

Procedures	Description	Functions	Description
Robot()	Add New Alias	Add_Variable()	Add a new variable
S4ProgamLoad()	Load a program in the robot controller	Remove_Variable()	Remove one variable
S4Run()	Switches ON the robot's motors	ControlID()	Gives the ID controller
S4Standby()	Switches OFF the robot's motors	InerfaceState()	Gives the interface state.
S4Start()	Start program execution		
S4Stop()	Stop program execution	**Events**	**Description**
S4ProgramNumVarRead()	Reads a numerical variable in the robot controller	StatusChanged()	This event activates after something changed in the robot controller. For example: power off, emergency stop, executing sate, stopped state, run, and so on.
S4ProgramNumVarWrite()	Writes a numerical variable in the robot controller	BoolVariableChanged()	This event activates after one Boolean variable changed.
S4ProgramStringVarWrite()	Writes a string in the robot controller	NumVariableChanged()	This event activates after one numeric variable changed.
S4ProgramStringVarRead()	Reads a string in the robot controller		

The *winEthernetROBOTcontrol* library was developed for industrial robot manipulators, and was implemented for the special case of ABB robots (with the new S4 control system). Nevertheless, this very flexible library was designed to work with all robots that support Ethernet. However, of course, if we use a different robot it is necessary to make some upgrades in this library, according to the used robot. The basic idea is to define a library of functions for any specific equipment that in each moment define the basic functionality of that equipment for remote use.

III. ROBOTIC BAR

Co-existence of robots and humans in shared workspace is one of the major characteristics of service robot applications. Furthermore, robots and humans will not only co-exist, but collaborate and interactively communicate with their human users. All this does require appropriate man-machine interaction technologies [2] and [3].

With the aim of testing the potentialities of the developed *winRS232ROBOTcontrol* in industrial applications, a Robotic Bar was developed. Fig. 5 shows the layout of the developed Robotic Bar. The Robotic Bar layout consists of a general purpose Mitsubishi RV-3SB robot and its controller unit, an Automated Guided Vehicles (AGV), optical and infrared sensors, conveyor, drinks machine (beer, juice and cola), two robotics interface, one electronic control module for the AGV, two tables, a black track and a personal computer.

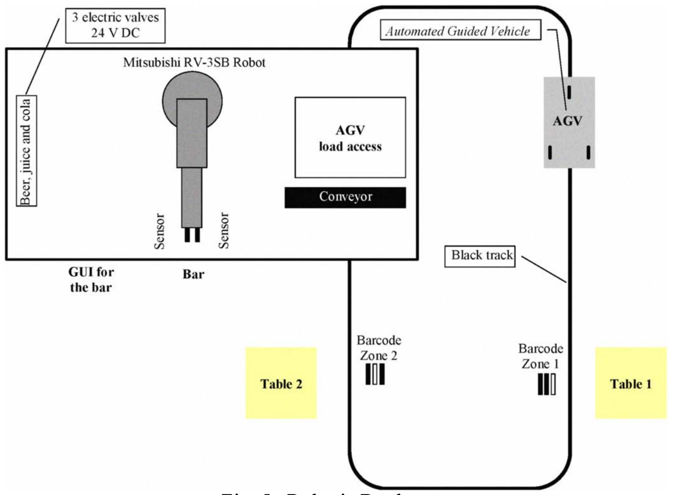


Fig. 5. Robotic Bar layout.

A. *Automatic Guided Vehicle*

Automatic Guided Vehicle (AGV) was designed and built to perform some operations in the bar without direct human guidance (e.g., take drinks to the tables). Fig. 6 shows the general diagram of the developed AGV and fig. 7 shows the AGV photograph. The AGV moves on the plant floor of the bar without need of an onboard operator or driver. On the plant floor, black tracks (paths) were defined for the AGV navigation.

The developed AGV has the following characteristics: weight, 6 Kg; dimensions, 64x38x12 cm; two batteries (12 V DC); more or less 10 hours of autonomy; 3 safety sensors; 6 guidance sensors; 3 zone sensors; left and right motors (12 V DC).

Digital Object Identifier: 10.4108/ICST.ROBOCOMM2009.5963
http://dx.doi.org/10.4108/ICST.ROBOCOMM2009.5963

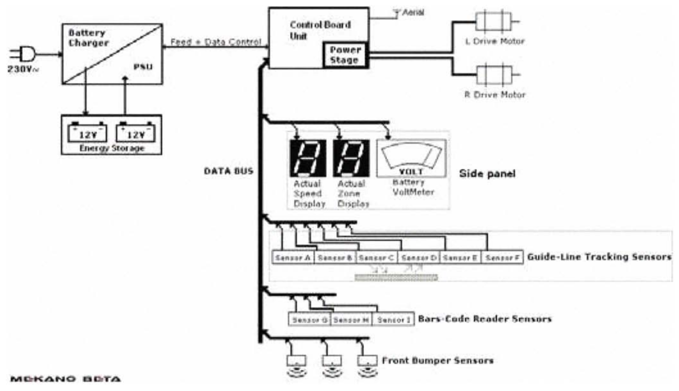


Fig. 6. General diagram of the AGV.

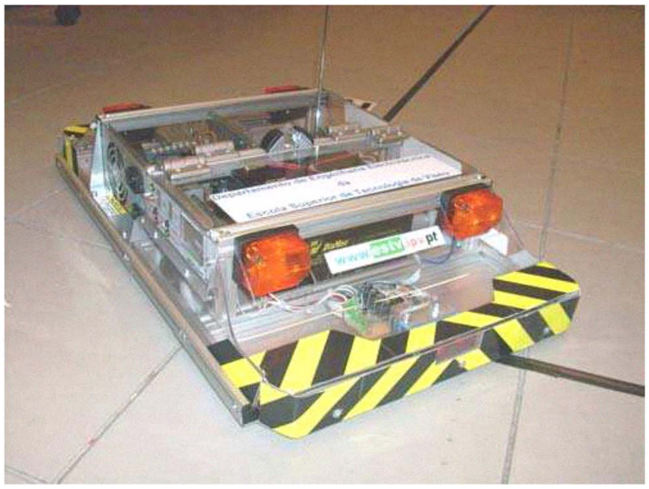


Fig. 7. Photograph of the AGV.

The communication system developed for the AGV and tables uses 4 bits for transmission, working frequency of 433.92 MHz and the maximum distance of operation is 250 meters. Optical guidance system was developed for the AGV but, if necessary, others guidance systems can be used.

B. *Electronic control modules*

Fig. 8 presents the position of the electronic control modules hardware: robotic interface 1, robotic interface 2 and electronic control module for the AGV.

The robotic interface 1 has the following functions:

- To receive human drinks requests done with the GUI for the bar (see Fig. 5) and the signals about all the sensors used (e.g., the signal of the sensor used in conveyor to detect the glass, and so on);
- To send requests to the robot controller (e.g., open the electric valve of the juice, and so on).

The robotic interface 2 has the following functions:

- To receive orders coming by the robot controller (e.g., open the electric valve of the juice, turn on/off the conveyor, and so on);
- To send the power signals necessary for the orders execution (e.g., turn on the conveyor motor with a constant speed, and so on).

The AGV electronic control module has the following functions:

- To receive the stop zone where the AGV needs to take the drinks (e.g., to leave the drinks at the table 1);
- To send the barcode zone number where the AGV should leave the drinks.

C. *Control and monitoring*

Fig. 8 also shows the control and monitoring of the Mitsubishi RV-3SB robot carried out by *winRS232ROBOTcontrol* presented in section II. An Ethernet server was developed for the computer (PC bar) that controls the Mitsubishi RV-3SB robot [4], [5] and [6]. Thus, it is possible to control the robot remotely, in a distributed client/server environment, as shown in Fig. 8.

The *winRS232ROBOTcontrol* was developed for industrial robot manipulators, and was implemented in the Mitsubishi RV-3SB robot, used in the Robotic Bar. The *winRS232ROBOTcontrol* allowed the development of programs for the controller of the Mitsubishi RV-3SB robot.

The software developed for the PC bar allows:

- Coordination, integration and control of all the tasks in Robotic Bar;
- Controlling and supervision in real time all the operations in Robotic Bar;
- Guaranteeing tolerance of failures, safety and coherence of the data in Robotic Bar.

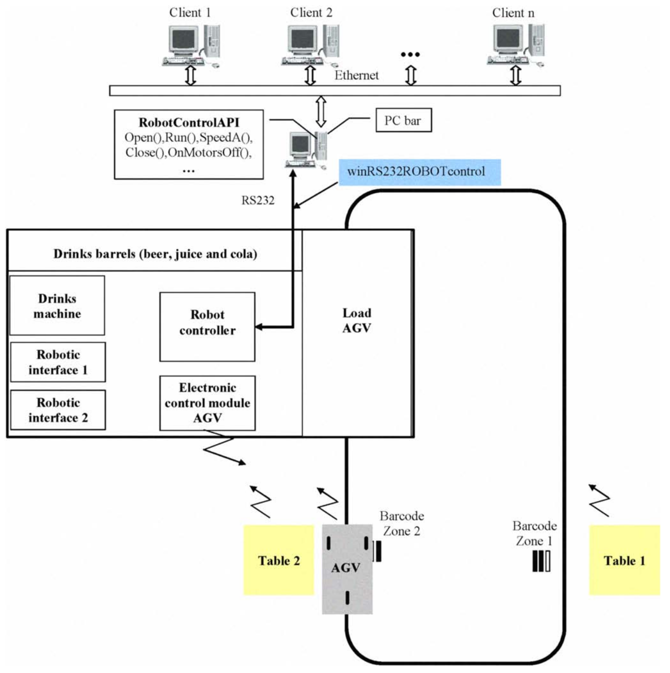


Fig. 8. Electronic control modules.

IV. FLEXIBLE MANUFACTURING CELL

With the aim of testing the potentialities of the developed software in industrial applications, a FMC was developed. The developed FMC is comprised of four sectors and the control of existing equipment in each sector is carried out by one of four computers: PC1 – manufacturing sector, PC2 – assembly sector, PC3 – handling sector and PC4 – storage sector. The coordination, synchronization and integration of the four sectors are carried out by the FMC manager computer [7] and [8].

Fig. 9 presents the layout of developed FMC, composed by four sectors:

- Manufacturing sector, made up of two CNC machines (mill and lathe), one ABB IRB140 robot and one buffer;
- Assembly sector, made up of one Scorbot ER VII robot, one small conveyor and an assembly table;
- Handling sector, made up of one big conveyor;
- Storage sector, made up of five warehouses and one robot ABB IRB1400.

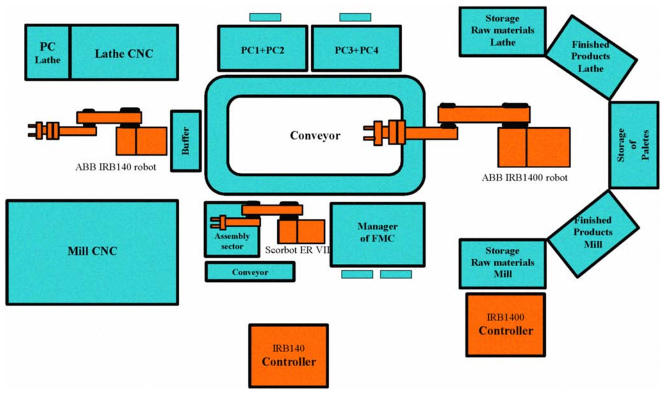


Fig. 9 Layout of Flexible Manufacturing Cell.

A. Manufacturing Sector

The manufacturing sector has two CNC machines (CNC Mill and CNC Lathe) responsible for the manufacturing of the jobs. In order to optimize the manufacturing jobs there is a buffer in this sector where small amounts of raw-materials for both machines are available and where finished-products are kept.

The load and unload operations of the machines and the buffer are executed by the ABB IRB140 robot. Fig. 10 shows the control of the ABB IRB140 robot is carried out by *winEthernetROBOTcontrol* presented in section II. Control of the CNC machines (Mill and Lathe) is carried out by Direct Numerical Control (DNC), and the robotics interfaces were used as redundancy.

B. Assembly Sector

The assembly sector is made up of a Scorbot ER VII robot [9], [10], [11], and [12], a conveyor and an assembly table. Responsibility for the control and coordination of this sector falls on PC2, as shown in Fig. 11. In this sector, the *winRS232ROBOTcontrol* presented in section II was used.

C. Storage Sector

Control of the storage sector is carried out by PC4, as shown in Fig. 12. The functions of the developed software for this sector are the administration of the database of the whole warehouse, and controlling the ABB IRB1400 robot [13], [14], and [15]. Control of the ABB 1400 robot is carried out by *winEthernetROBOTcontrol* presented in section II.

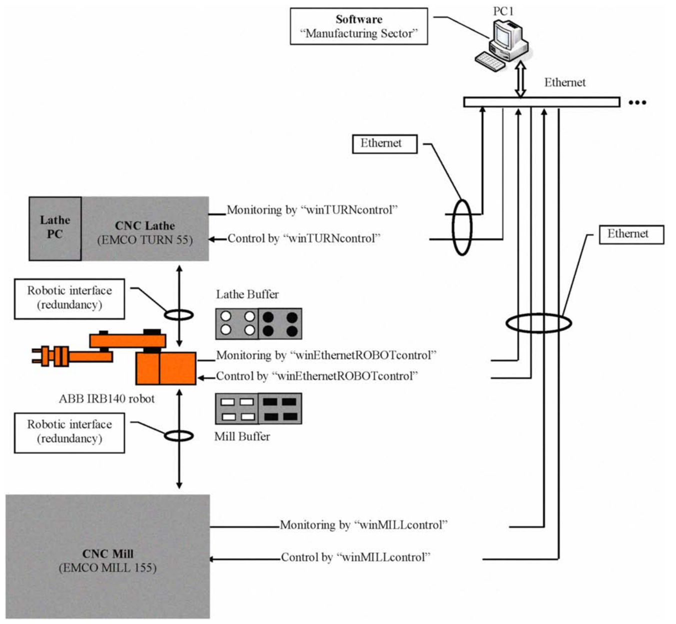

Software "Manufacturing Sector"
PC1
Ethernet
Ethernet
Ethernet
Lathe PC
CNC Lathe (EMCO TURN 55)
Monitoring by "winTURNcontrol"
Control by "winTURNcontrol"
Robotic interface (redundancy)
Lathe Buffer
Monitoring by "winEthernetROBOTcontrol"
Control by "winEthernetROBOTcontrol"
ABB IRB140 robot
Robotic interface (redundancy)
Mill Buffer
Monitoring by "winMILLcontrol"
Control by "winMILLcontrol"
CNC Mill (EMCO MILL 155)

Fig. 10 Control and monitoring of the manufacturing sector.

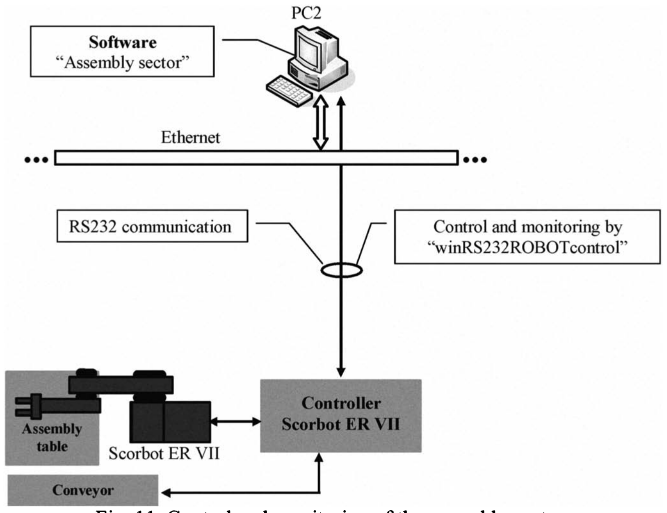

PC2
Software "Assembly sector"
Ethernet
RS232 communication
Control and monitoring by "winRS232ROBOTcontrol"
Controller Scorbot ER VII
Assembly table
Scorbot ER VII
Conveyor

Fig. 11 Control and monitoring of the assembly sector.

Digital Object Identifier: 10.4108/ICST.ROBOCOMM2009.5963
http://dx.doi.org/10.4108/ICST.ROBOCOMM2009.5963

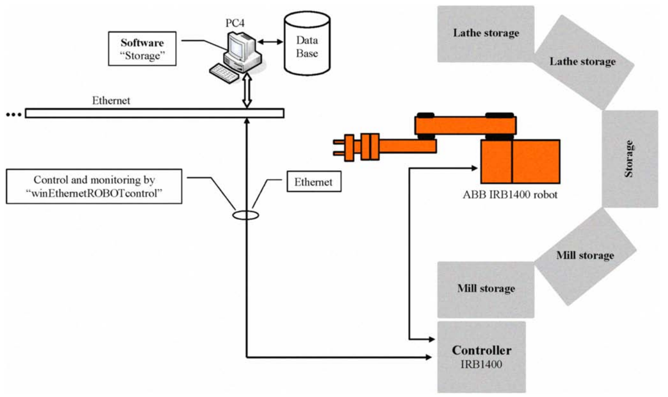


Fig. 12 Control and monitoring of the storage sector.

V. Conclusion

In this paper, a collection of flexible software tools for industrial robots was developed: *winRS232ROBOTcontrol*, *winEthernetROBOTcontrol*. These software tools allow the development of industrial applications of monitoring and controlling robotic networks inserted in modern systems (e.g., FMC, Robotic Bar, and so on). All the software developed has operation potentialities in Ethernet.

Implementation of the proposed software architecture was presented for the special cases of the robots we have at our laboratory (robots: ABB IRB 140 robot, ABB IRB 1400 robot, Scorbot ER VII robot and Mitsubishi RV-3SB robot). The proposed software architecture was tested and validated in the developed Robotic Bar and in the FMC. As demonstrated, it was very effective and functional in the resolution of coordination problems, integration and control of several Robotic Bar and FMC equipments.

The proposed software architecture may be extended and applied to other robots from different manufacturers. However, of course, if we use different robots it is necessary to make some upgrades in these proposed software architecture. The basic idea is to define a library of functions for any specific equipment that in each moment define the basic functionality of that equipment for remote use.

The main advantages of the developed flexible software are:

– Using these software applications in industrial robots of different manufacturers;
– Developing intelligent control and integration software for FMC and Robotic Bar using an only programming language (e.g., C++);
– Remote programming using the referred mechanism, together with the library of generic functions defined for each piece of equipment;
– Loading programs in the robots' controllers from any computer in the network;
– Administration of programs and files, including operation registration files;
– Monitoring of status, accountancy and error registration;
– Recovering of stop situations, placing the robots in safety positions;
– Intelligent control and integration of FMC and Robotic Bar. It is possible to coordinate, synchronize and integrate different equipments from different manufacturers in the development of industrial and didactic applications.

References

[1] António Ferrolho and Manuel Crisóstomo, "Intelligent Control and Integration Software for Flexible Manufacturing Cells," *IEEE Transactions on Industrial Informatics*, vol. 3, no. 1, ISSN: 1551-3203, pp. 3-11, February 2007.
[2] U. Rembold, B. O. Nnaji and A. Storr, *Computer Integrated Manufacturing and Engineering*. Addison-Wesley, 1993.
[3] Mikell P. Groover, *Automation, Production Systems, and Computer Integrated Manufacturing*. Prentice Hall, New Jersey, 2001.
[4] Mitsubishi Industrial Robot, CR1/CR2/CR3/CR4/CR7/CR8/CR9 Controller, Instruction Manual, Melfa BFP-A5992-K, 2005.
[5] Mitsubishi Industrial Robot, RV-3SB/3SJB Series, Standard Specifications Manual, Melfa BFP-A8408-A, 2005.
[6] Mitsubishi Industrial Robot, CR1/CR1B Controller, Instruction Manual, Melfa BFP-A8054-H, 2004.
[7] António Ferrolho and Manuel Crisóstomo, "Control and Scheduling in Flexible Manufacturing Cells," *Proceedings of the IEEE International Conference on Industrial Technology (ICIT06)*, Mumbai, India, December 15-17, 2006, ISBN: 1-4244-0726-5, pp. 1241-1246, 2006.
[8] António Ferrolho e Manuel Crisóstomo, "Control and Scheduling Software for Flexible Manufacturing Cells", *Industrial Robotics: Programming, Simulation and Applications*, ISBN: 3-86611-286-6, pp. 315-340, Advanced Robotic Systems, 2007.
[9] Scorbot ER VII, *User's Manual. Eshed Robotec*, 2ª Edição, 1996.
[10] Advanced Control Language, *Reference Guide*. Eshed Robotec, 4th Edition, 1995.
[11] ACLoff-line, *User's Manual. Eshed Robotec*, 2ª Edição, 1995.
[12] Advanced Terminal Software, *Reference Guide*. Eshed Robotec, 1st Edition, 1995.
[13] ABB RAPID Reference Manual, ABB Robotics AB, 2003.
[14] ABB Advanced User's Guide, RAP Protocol Specification TRP-1. ABB Automation Technology Products AB, 2003.
[15] ABB User's Guide, *RAP Service Specification*. ABB Automation Technology Products AB, 2003.

Digital Object Identifier: 10.4108/ICST.ROBOCOMM2009.5963
http://dx.doi.org/10.4108/ICST.ROBOCOMM2009.5963