RTAI Based Real-Time Control of Robotic Wheelchair

Chong Hui Kim, Seong Jin Kim and Byung Kook Kim
Department of Electrical Engineering & Computer Science
Korea Advanced Institute of Science and Technology
373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Korea
{chkim, sjkim}@rtcl.kaist.ac.kr, bkkim@ee.kaist.ac.kr

Abstract

In order to provide autonomous mobility, a robotic wheelchair requires various tasks such as sensing, localization, obstacle detection, and motor control, which is a very complex control system. In this paper, we present a real-time control system design for robotic wheelchairs based on RTAI. We developed a multiprocessor-based robotic wheelchair platform for the disabled . Based on discussions on real-time requirements of robotic wheelchairs, we designed an efficient software architecture for autonomous real-time control of robotic wheelchairs. The performance is revealed by experimental results carried out at the *Intelligent Sweet Home* in KAIST.

1 Introduction

As an aging society is coming and the disabled population is on the increase, social demands for better life have been increasing. Recent advances in research areas such as robotics, artificial intelligence, and sensor technology lead them to expect that the desires of overcoming their handicap will become true in the near future. The aim of rehabilitation engineering is to improve the functions of them [1] and assist them to make a living without assistance of another person as far as possible [2].

Powered wheelchair is a necessary locomotive system to provide mobility aid for the people with motor disabilities. However, it is still difficult to drive a conventional powered wheelchair for the severely-handicapped people who have spinal cord injury at the cervical, quadriplegia, tremors, and so forth. Hence they have desired development of robotic/intelligent wheelchair to provide independent mobility [3].

As a robotic wheelchair carries a person and operates close to human, its malfunction can lead to unwanted results or vital accidents. Hence it should be considered as a real-time system [4].

Robotic wheelchair is a very complex control system. In order to provide mobility, robotic wheelchair requires various functions such as localization, navigation, and obstacle avoidance as well as several sensors such as encoder, sonar, vision, and laser range finder (LRF) to cope with various changes of external world. Robotic wheelchair consists of many tasks related to sensors and functions, which must be performed with prespecified deadlines. Since each task has diverse sample rates, computational loads, and time constraints, it is difficult to ensure achievement in-time execution of all tasks. This problem can be solved with the help of a real-time operating system.

In this paper, we present a real-time control system design for robotic wheelchairs by retrofitting a commercial powered wheelchair. In order to provide real-time capability, we adopted a Linux real-time extension called Real-Time Application Interface (RTAI).

The ultimate goal of robotic wheelchair is to take the user automatically and safely to the destination. In order to achieve this goal, many robotic wheelchair projects have been emerged in recent years. As representative research projects, there are NavChair [5], SIAMO [6], SENARIO [7], RobChair [8], MAid [9], FRIEND [10], Rolland [11], VAHM [12], and KARES [13].

Many research groups [6, 10, 13] developed a robotic wheelchair under Windows operating system. Windows is not designed for a real-time system, but widely used since it provides easy development environment, abundant device drivers and multitasking. However, it requires a lot of resources to maintain a graphic display so that it cannot guarantee whether task is completed satisfying the time constraints.

Moreover, writing device drivers is not straightforward and cumbersome.

Some groups [9, 11] use commercially available real-time operating systems such as QNX and LynxOS. While these systems have good real-time features, they are too expensive, have less device drivers, and lack expandability.

There are few groups to develop a robotic wheelchair using Linux. There are many advantages in using Linux because it is a free software with open architecture and provides comprehensive development environment including debugger and graphical user interface. In addition, device driver is easy to write and its multitasking is better than windows multitasking. Linux was not designed to be a real-time system, but real-time extension is available such as RTAI. Hence Linux with RTAI can be a good solution for the real-time control of robotic wheelchair, which is a complex real-time system.

The remainder of this paper is organized as follows. In Sections 2, we describe our developed robotic wheelchair. Section 3 deals with software architecture for real-time control of developed robotic wheelchair. In Section 4, some experimental results are shown. Section 5 is for concluding remarks.

2 System Description

2.1 Hardware Architecture with Multiprocessor

Our robotic wheelchair is based on a commercial powered wheelchair *Chairman* manufactured by Permobil, which is powered by two 12V batteries.

Figure 1 shows our retrofitted robotic wheelchair platform. Its controller is divided into three components: PC, joystick module, and power module. The PC provides high-speed processing for control and data acquisition from sensors [14]. It is mounted on the back of the wheelchair and powered by Linux with RTAI for real-time processing. The joystick module receives user's intention via joystick and converts them to a corresponding control commands. The power module converts the control command to power signal for driving wheelchair and monitor the current status of robotic wheelchair. These two modules contain a H8 microprocessor and are linked to the PC via a RS232 serial port.

In order to provide autonomous mobility, our robotic wheelchair is equipped with sensors such as LRF and two incremental encoders to localize its own position and to detect obstacles. Incremental encoders are used to obtain odometric data which provide relative position and orientation of wheelchair.

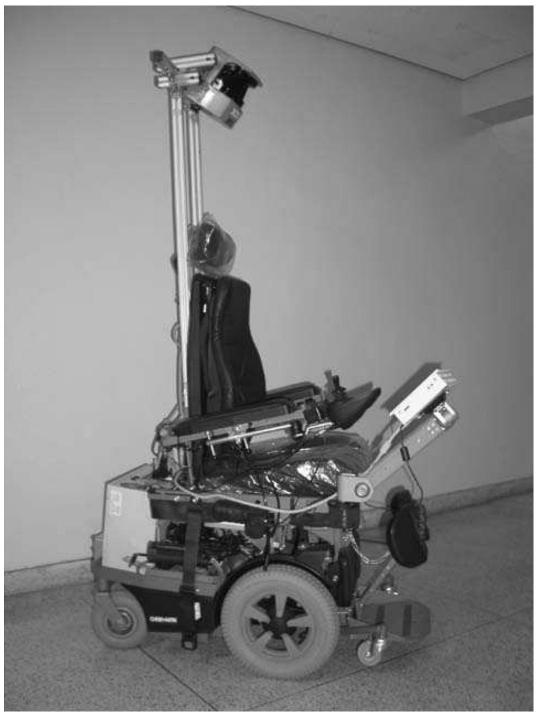
FIGURE 1: *Robotic Wheelchair Platform*

An LRF (SICK LMS 200) is mounted on a custom designed aluminum props at a height of 176cm and scan plane titled at an angle of 25 degrees to the ground as shown in Figure 1 so that the wheelchair can perceive the environment and detect objects below the height of the LRF. The LRF provides a selectable angular resolution of 0.25°, 0.5°, or 1°. In our system, it scans the 180° coverage at 0.5° angular resolution in every complete scan period T_L (about 26ms). Since a data packet for complete scan is 733 bytes long, it is linked to the PC via RS422 serial port for high speed data transmission at 500K baud rate.

There is also bumper switches which consist of four microswitches. It is attached to the footplate of robotic wheelchair and used to detect and avoid collision.

Since many heavily handicapped people employing a wheelchair suffer from controlling a powered wheelchair with universal joystick, LCD with touch screen is attached to the neighborhood of the right armrest as a visual interactive human-machine interface.

2.2 Real-Time Requirements of Robotic Wheelchair

Since robotic wheelchair carries a person and interacts with environment through sensors, consideration of safety is very important and inevitable. For example, if the sensor information for detecting obstacle or control task is missed, the rider can be faced with dangerous situation such as a collision with a static or moving object. It means that safety depends not only upon functional correctness but also upon temporal exactness in which each task is executed. In our robotic wheelchair tasks related to sensors and actuators have real-time constraints and are created as real-time tasks by the real-time kernel.

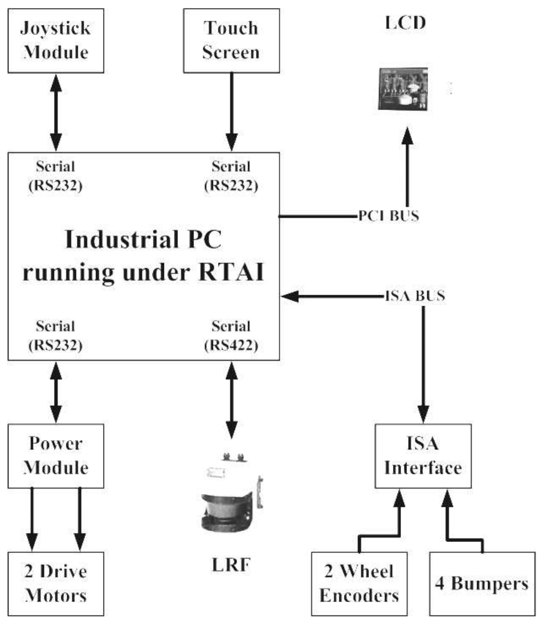

FIGURE 2: *Hardware Architecture of Developed Robotic Wheelchair's Control System*

Figure 2 shows the hardware architecture of our robotic wheelchair's control system. In this system, joystick module periodically issues a command and requires a status feedback every T_J (about 10ms). During this operation, motor control task implemented in PC must determine the control signal from sensor data and user's intention through joystick, and prepare the control packet for the power module every T_J. There is a data dependency because motor control task uses the command packet from the joystick module.

For an efficient real-time processing, two different kinds of real-time tasks are required: periodic task performed at a given rate, and aperiodic/sporadic task performed as a consequence of particular event [15]. The motor control task is usually implemented as a periodic task. In our robotic wheelchair, joystick module acts as a master controller and the motor control task should be periodically executed every T_J. If the motor control task is executed with different period, in-time response can be missed. Since we use a commercial joystick module, T_J is not known. Although we can precisely measure the period T_J, there is a small measurement error. Hence it is difficult to create the motor control task as a periodic task with the exact period of T_J.

In order to resolve data dependency and periodic task creation problem, we create the motor control task as an aperiodic task synchronized with status packet from the power module. Initially, the motor control task is created in a suspended state. If the status packet is incoming, the execution of motor control task is resumed by **rt_task_resume** called in serial interrupt service routine (ISR). After completing execution, it is suspended by **rt_task_suspend** as shown in Figure 3.

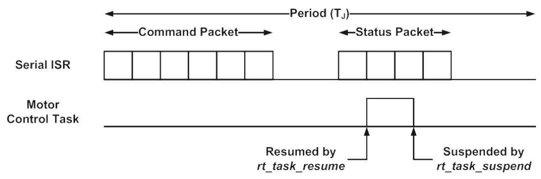

FIGURE 3: *Execution scheme of motor control task*

Since LRF transmits continuously scan data every complete scan period T_L, localization and object detection should be completed within time constraints T_L.

3 Software Design for Robotic Wheelchair

3.1 Hierarchical Control Architecture

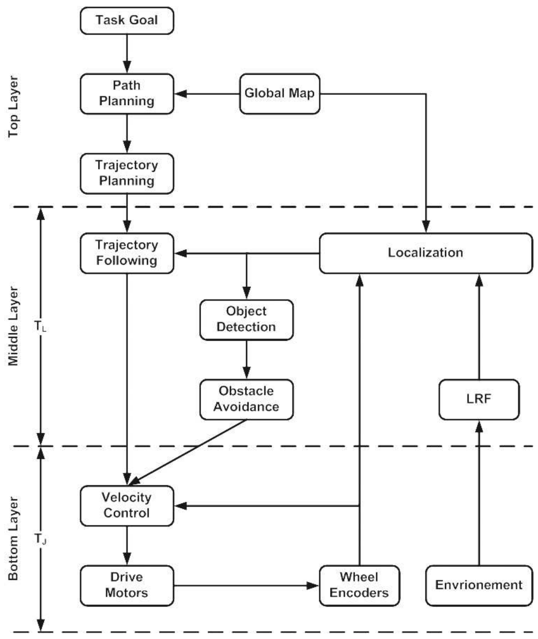

FIGURE 4: *Hierarchical control architecture of our system*

Figure 4 shows the hierarchical control architecture of our system. Hierarchical control is divided into three layers with respect to characteristic of tasks.

The bottom layer consists of low-level motor control and position feedback from encoders which are executed at every T_J and have light computation load. The middle layer consists of tasks related to some tactical components such as environment sensing, localization, and object detection. These tasks are executed at every T_L and require heavier computational loads than tasks of bottom layer. The top layer consists of goal selection, path and trajectory planner which are executed aperiodically by particular request of user or other task.

All tasks are divided into two layer as shown in Figure 5. Real-time (RT) control layer consists of tasks with real-time requirements such as motor control and localization including object detection. Non-real-time (Non-RT) control layer consists of tasks with soft or no timing constraints such as graphical user interface. Tasks between two layers communicate by means of shared memory and RT-FIFO as shown in Figure 5.

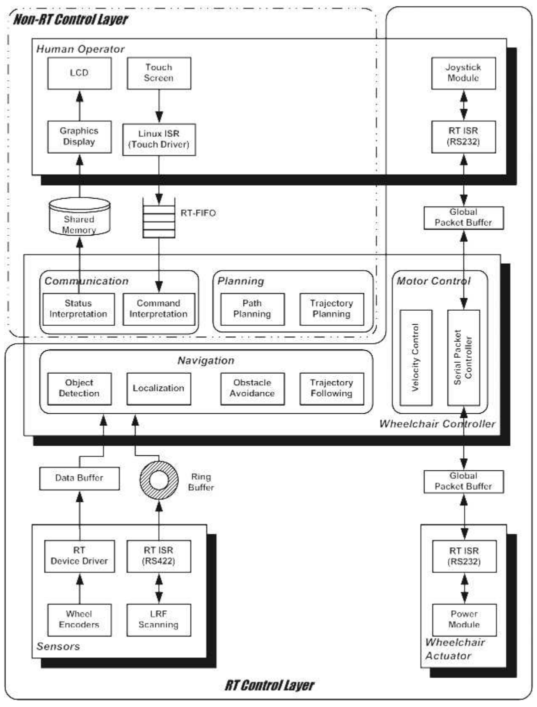

FIGURE 5: *Software Architecture of Our System*

3.2 Real-Time Control Layer

RT control layer consists of two real-time control tasks: motor control and navigation. These tasks are implemented as real-time kernel modules.

```
while(TRUE) {
        rt_get_time();              /* read current time for dead-reckoning */
        <Read encoder>;             /* sensor acquisition */
        <estimate current position using dead-reckoning>;
        <get user's intention from command packet of joystick>;
        <control law>;              /* velocity control */
        <convert control input to control packet sent to power module>
        rtai_task_suspend();        /*suspend till next status packet incoming */
}
```

FIGURE 6: *Pseudocode of Motor Control Task*

The pseudocode of motor control task is shown in Figure 6. Its period is equal to the communication period of joystick module and power module. For synchronization, it is created as aperiodic task and its execution is resumed by serial ISR when status packet header is arrived. Although it is created as aperiodic task, it is periodically executed every T_J. It permits to move the robotic wheelchair to the desired position considering user's intention.

```
while(TRUE) {
        rt_get_time();              /* read current time for dead-reckoning */
        <Read encoder>;             /* sensor acquisition */
        <estimate current position using dead-reckoning>;
        <feature extraction>;       /* extract line feature from point data of LRF */
        <map matching>;
        <object detection>;
        if (ObjectFound)            /* if object is found */
                ObstacleAvoidance();
        else                        /* if object is not found */
                TrajectoryFollwoing();
        if (PositionUpdate)         /* if localization result is available */
                <update position from localization result>;
        rtai_task_suspend();        /* suspend till next complete scan */
}
```

FIGURE 7: *Pseudocode of Navigation Task*

Figure 7 shows the pseudocode of navigation task. Navigation task performs localization and obstacle detection using LRF and encoders. For high speed processing, LRF is linked to PC via RS422 serial port and transmits continuously sensed data every T_L. To avoid synchronization and data dependency problem with LRF, navigation task is also created as aperiodic task. Its execution is resumed by **rtai_task_resume** after a complete data transmission of LRF.

Our localization algorithm is based on line feature and map matching. Since the LRF provides clear line features for an indoor environment, lines provide strong and accurate information, but with far less number than that of points. Therefore, line based algorithms are more appropriate for the real-time system compared to algorithms that are point based.

For object detection, we defined that object is one not on the given map. Object detection is performed using unused point data in localization from LRF scan data.

3.3 Non-Real-Time Control Layer

Non-RT control layer consists of communication, planning, and graphical user interface and is executed at user space.

Since a robotic wheelchair is highly interactive system, graphical user interface is an important issue. The graphical user interface is built on Linux using GTK which is a multi-platform toolkit for creating graphical user interfaces. Figure 8 shows the graphical user interface which is composed of mode selection button, a motion pad, and a display window.

There are three control modes: FREE, SEMI, and AUTO. The motion pad is used to direct the wheelchair. Translation velocity is mapped to vertical axis and rotation velocity is mapped to the horizontal axis. Since some handicapped people can not precisely touch the intended point, we assigned a large pixel area in the center for clear stop command. It is very easy and intuitive to read the console and view status.

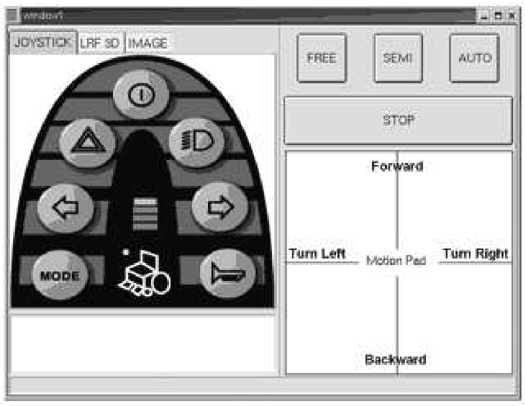

FIGURE 8: *Graphical User Interface*

Using a communication module based on TCP/IP based on wireless network, our robotic wheelchair can control home appliances such as lights, curtain, and TV through the home network server. Also the user can control the robotic wheelchair using his/her voice through the home network server.

4 Experimental Result

We tested our developed robotic wheelchair at *Intelligent Sweet Home* in KAIST. Test objective is to transfer the disabled from a bed to the robotic wheelchair without assistance of another person. For accomplishing this objective, the robotic wheelchair moves autonomously to the predefined docking ready position and performs docking with the robotic transfer system. During docking operation, bumpers of robotic wheelchair detect collision with robotic transfer system and help safe docking.

Figure 9 shows the robotic wheelchair is moving toward to the predefined position. During autonomous moving, robotic wheelchair performs localization, object detection, and motor control task.

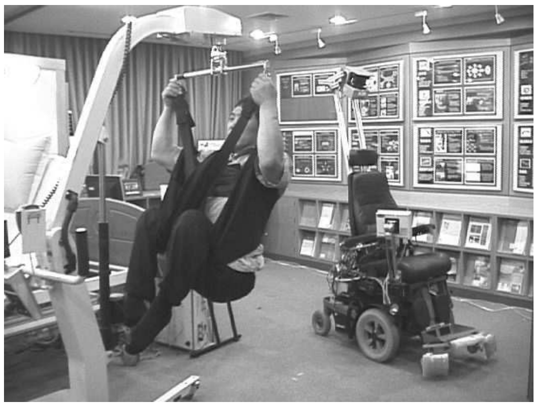

FIGURE 9: *Autonomous Moving to the Predefined Position*

Figure 10 shows the docking operation. During this operation, robotic wheelchair detects collision with robotic transfer system using bumpers for safe docking. When a bumper is pressed, it is quickly serviced by a real-time ISR. If right side switch is pressed, robotic wheelchair moves toward to the left-front direction. If left side switch is pressed, robotic wheelchair moves toward to the right-front direction. If one of two font switches is pressed, robotic wheelchair stops immediately.

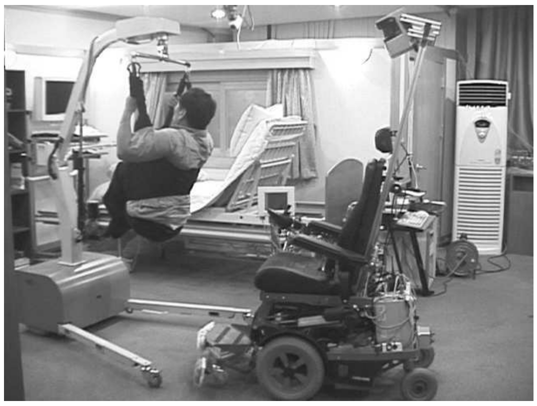

FIGURE 10: *Docking Operation*

Table 1 shows the required basic tasks to provide mobility and worst-case execution time of each task. The processing time of navigation module from data transfer of LRF raw data to location update including obstacle detection is less than 18ms, which are done in real-time without missing since the full scan period of LRF is equal to T_L (26ms).

Module	Task	Worst Ex. (ms)
Motor	Motor Control	0.46
Navigation	RS422 Data Tx.	16.5
	Feature Extraction	0.49
	Map Matching	0.35
	Obstacle Detection	0.35
	Location Update	0.08
Bumper	Bumper Handling	0.05

TABLE 1: *Tasks of Robotic Wheelchair*

Also the bumper interrupt is processed with low latency and processing time so that the robotic wheelchair can quickly react for the collision.

5 Concluding Remarks

We developed a sensor-based robotic wheelchair for the people with motor disabilities. Since robotic wheelchair requires real-time capability, we adopted Linux with RTAI. We discussed real-time requirements of our robotic wheelchair and software architecture is realized under RTAI. We carried our experiments in *Intelligent Sweet Home* at KAIST so that our developed robotic wheelchair could navigate and dock with robotic transfer system, and hence can transfer the disabled from bed to robotic wheelchair safely . During operation, all tasks of robotic wheelchair are done in real-time without missing as shown in Table 1.

Further work will focus on docking control with battery charging station for autonomous battery charging and human-robot cooperative control.

Acknowledgment

This work is supported by KOSEF through HWRS-ERC at KAIST.

References

[1] S. Fioretti, T. Leo, and S. Longhi, 2000, *A Navigation System for Increasing Autonomy and the Security of Powered Wheelchairs*, IEEE Transactions on Rehabilitation Engineering, Vol. 8, pp 490-498.

[2] Kwang-Hyun Park and Z. Zenn. Bien, 2003, *Intelligent Sweet Home for Assisting the Elderly and the Handicapped*, Proceedings of the 1st International Conference on Smart Homes and Health Telematics, pp 151-158.

[3] Chong Hui Kim, Jik Han Jung, and Byung Kook Kim, 2003, *Design of Intelligent Wheelchair for the Motor Disabled*, Proceedings of the 8th International Conference on Rehabilitation Robotics, pp 92-95.

[4] Jane W. S. Liu, 2000, *Real-Time Systems*, Prentice Hall, ISBN 0130996513.

[5] Simon P. Levine, David A. Bell, Lincoln A. Jaros, Richard C. Simpson, Yoram Koren, and Johan Borenstein, 1999, *The NavChair Assistive Wheelchair Navigation System*, IEEE Transactions on Rehabilitation Engineering, Vol. 7, No. 4, pp 443-451.

[6] Manuel Mazo et al., 2001, *An Integral System for Assisted Mobility*, IEEE Robotics & Automation Magazine, Vol. 8, No. 1, pp 46-56.

[7] N.I. Katevas, N.M. Sgouros, S.G. Tzafestas, G. Papakonstantinow, P. Beattie, J.M. Bishop, P. Tsanakas, and D. Koutsouris, 1997, *The autonomous mobile robot SENARIO: a sensor aided intelligent navigation system for powered wheelchair*, IEEE Robotic & Automation Magazine, Vol. 4, No. 4, pp 60-70.

[8] G. Pires, R. Araujo, U. Nunes, and A.T. de Almeida, 1998, *RobChair-a powered wheelchair using a behaviour-based navigation*, 5th International Workshop on Advanced Motion Control, pp 536-541.

[9] Erwin Prassler, Jens Scholz, and Paolo Fiorini, 2001, *A Robotics Wheelchair for Crowded Public Environments*, IEEE Robotics & Automation Magazine, Vol. 8, No. 1, pp 38-45.

[10] Christian Martens, Nils Ruchel, Oliver Lang, Oleg Ivlev, and Axel Gräser, 2001, *A FRIEND for Assisting Handicapped People*, IEEE Robotics & Automation Magazine, Vol. 8, No. 1, pp 57-65.

[11] Axel Lankenau, and Thomas Röfer, 2001, *A Versitile and Safe Mobility Assistance*, IEEE Robotics & Automation Magazine, Vol. 8, No. 1, pp 29-37.

[12] G. Bourhis, O. Horn, O. Habert, and A. Pruski, 2001, *An Autonomous Vehicle for People with Motor Disabilities*, IEEE Robotics & Automation Magazine, Vol. 8, No. 1, pp 57-65.

[13] W. -K. Song, H. Lee and Z. Bien, 1999, *KARES: Intelligent Wheelchair-Mounted Robotic Arm System Using Vision and Force Sensor*, Robotics and Autonomous Systems, Vol. 28, No. 1, pp 83-94.

[14] C.J. Lee and C. Mavroidis, 2001, *PC-Based Control of Robotic and Mechatronic Systems Under MS-Windows NT Workstation*, IEEE/ASME Transactions on Mechatronics, Vol. 6, No. 3, pp 311-321.

[15] Maurizio Piaggio, Antonio Sgorbissa, and Renato Zaccaria, 1999, *ETHNOS: a Light Architecure for Real-Time Mobile Robots*, Proceedings of the 1999 IEEE/RSJ Internation Conference on Intelligent Robots and Systems, pp 1292-1297.