Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 10(4): 372-377, Dec. 2012 **Regular Paper**

OSEK/VDX Porting to the Two-Wheel Mobile Robot Based on the Differential Drive Method

Duy Le Nguyen and Myung-Eui Lee*, *Member, KIICE*

Department of Electrical, Electronics and Communication, Korea University of Technology and Education, Cheonan 330-708, Korea

Abstract

In this paper, we propose an implementation of a real-time operating system for the two-wheel mobile robot. With this implementation, we have the ability to control the complex embedded systems of the two-wheel mobile robot. The advantage of the real-time operating system is increasing the reliability and stability of the two-wheel mobile robot when they work in critical environments such as military and industrial applications. The real-time operating system which was ported to this implementation is open systems and the corresponding interfaces for automotive electronics (OSEK/VDX). It is known as the set of specifications on automotive operating systems, published by a consortium founded by the automotive industry. The mechanical design and kinematics of the two-wheel mobile robot are described in this paper. The contributions of this paper suggest a method for adapting and porting OSEK/VDX real-time operating system to the two-wheel mobile robot with the differential drive method, and we are also able to apply the real-time operating system to any complex embedded system easily.

Index Terms: Mobile Robot, OSEK/VDX, Real-time operating system

I. INTRODUCTION

To manage sophisticated systems and increase stability, it is desirable to use a real time operating system (RTOS) in robots. An RTOS will improve the predictability of applications, do the right thing at the right time, and increase stability of the system. Unfortunately, designing an RTOS for robotics is very difficult because the scale and scope of robotics continues to grow. Different types of robots can have wide varieties of hardware and platforms, making code reuse nontrivial. The current trends in robotics software are the development of the frameworks that are implemented in algorithms running in many operating system platforms. A number of robotics frameworks, such as robot operating system (ROS), Dave's ROS (DROS), Orocos, open robotics automation virtual environment (OpenRAVE), are widely available. These factors force developers to still depend heavily on the robot hardware that they program.

Observing industrial vehicles, with the development of automotive functions, the number of engine control units in a whole car is increasing correspondingly, the software development of automotive electronics is becoming more complicated, and automotive electronics now has high real-time requirements. In 1993, the European automobile industry provided the open systems and the corresponding interfaces for automotive electronics (OSEK/VDX) [1] standard to resolve the above problems. From the analysis of the favorable aspects of automotive operating system, it is clear that we

Received 30 July 2012, **Revised** 05 September 2012, **Accepted** 14 September 2012
***Corresponding Author** E-mail: mlee@koreatech.ac.kr

Open Access **http://dx.doi.org/10.6109/jicce.2012.10.4.372** print ISSN: 2234-5973 online ISSN: 2234-8883

can use an automotive operating system for robotics. Since this implementation of the robot by using OSEK/VDX is entirely feasible, we demonstrate how to implement a two-wheel mobile robot using OSEK/VDX RTOS.

II. BACKGROUND OF THE OSEK/VDX

Nowadays, automotive standards have become international standards and are used in other areas such as robotics. OSEK/VDX and AUTomotive Open System ARchitecture (AUTOSAR) [2] are known as two popular automotive operating systems. Between OSEK/VDX and AUTOSAR, we choose OSEK/VDX for implementation in a robot because AUTOSAR has a large number of standards from the AUTOSAR consortium that are very complex, does not have open source implementation, and requires commercial licenses. Furthermore, OSEK/VDX is the foundation of AUTOSAR. Therefore, OSEK/VDX is the best option for this implementation.

OSEK/VDX is a multitask operating system that is becoming a standard in the European automotive industry. This standard is primarily composed of four standards: the OSEK/VDX Operating System, OSEK/VDX Communication, OSEK/VDX Network Management, and OSEK/VDX OSEK Implementation Language. Three additional standards are in progress: the OSEK/VDX Run Time Interface, OSEK-Time, and OSEK/VDX Fault-Tolerant Communication. The OSEK standard specifies interfaces to multitasking functions (generic I/O and peripheral access) and thus remains architecture dependent. OSEK systems are expected to run on chips without memory protection. Features of an OSEK implementation can usually be configured at compile-time. The number of application tasks, stacks, mutual exclusion (mutexes), etc., is statically configured; it is not possible to create more at run time. The specification uses ISO/ANSI-C-like syntax; however, the implementation language of the system services is not specified. These OSEK/VDX standards are now candidates for standardization by the International Standards Organization (ISO) under standard number ISO 17356.

Since the publication of OSEK/VDX standards, many implementations have been designed by RTOS vendors or automotive engineering societies. Most of them are certified and these licenses are commercial. The following open-source initiatives should be mentioned: PICos18 [3], open OSEK [4], Free OSEK [5], EMERALDS-OSEK [6], nxt OSEK [7], TOPPERS/OSEK [8], ERIKA [9], and Trampoline [10]. These are innovative RTOSs for small microcontrollers based on an application programming interface (API) similar to those proposed by the OSEK/VDX Consortium. In this paper, we would like introduce Trampoline, which we used as an OSEK/VDX implementation.

III. THE MOBILE ROBOT MODEL

A trajectory-tracking controller is needed for the robots to be able to follow a planned trajectory [11].

In this section, we present a controller based on the kinematic model of mobile robots. The kinematics equations of the mobile robot are written as (Fig. 1):

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases} \tag{1}$$

These new variables are listed as following:

$$\begin{cases} z_1 = x \\ z_2 = \tan\theta \\ z_3 = y \end{cases} \tag{2}$$

The new control inputs are listed below:

$$\begin{cases} u_1 = v\cos\theta \\ u_2 = \omega(1+\tan^2\theta) \end{cases} \tag{3}$$

A new form for the kinematic equations of motion:

$$\begin{cases} \dot{z}_1 = u_1 \\ \dot{z}_2 = u_2 \\ \dot{z}_3 = z_2 u_1 \end{cases} \tag{4}$$

The desired trajectory for the new configuration variables:

$$\begin{cases} z_1^d(t) = x^d(t) \\ z_2^d(t) = \frac{\dot{y}^d(t)}{\dot{x}^d(t)} \\ z_3^d(t) = y^d(t) \end{cases} \tag{5}$$

The new desired configuration variables:

$$\begin{cases} u_1^d = \dot{x}^d(t) \\ u_2^d = \frac{d}{dt}\left(\frac{\dot{y}^d(t)}{\dot{x}^d(t)}\right) = \frac{\ddot{y}^d(t)\dot{x}^d(t) - \ddot{x}^d(t)\dot{y}^d(t)}{(\dot{x}^d(t))^2} \end{cases} \tag{6}$$

We write the nonlinear error equations as:

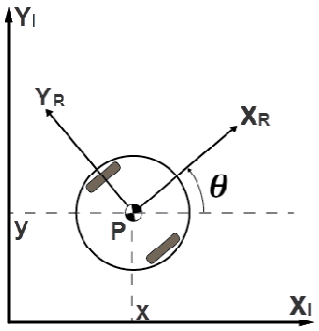

Fig. 1. The global reference frame and robot local reference frame.

$$\begin{cases} \dot{\tilde{z}}_1 = \tilde{u}_1 \\ \dot{\tilde{z}}_2 = \tilde{u}_2 \\ \dot{\tilde{z}}_3 = z_2^d \tilde{u}_1 + \tilde{z}_2 u_1^d + \tilde{z}_2 \tilde{u}_1 \end{cases} \tag{7}$$

Now, we linearize the nonlinear system

$$\begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \end{bmatrix} = \begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & k_3 / u_1^d \end{bmatrix} \begin{bmatrix} \tilde{z}_1 \\ \tilde{z}_2 \\ \tilde{z}_3 \end{bmatrix} \tag{8}$$

in which k_1, k_2, k_3 are constant controller gains. The following linear time-variant closed loop system is obtained:

$$\begin{bmatrix} \dot{\tilde{z}}_1 \\ \dot{\tilde{z}}_2 \\ \dot{\tilde{z}}_3 \end{bmatrix} = \begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & k_3 / u_1^d \\ k_1 z_1^d & u_1^d & 0 \end{bmatrix} \begin{bmatrix} \tilde{z}_1 \\ \tilde{z}_2 \\ \tilde{z}_3 \end{bmatrix} \tag{9}$$

Select the controller gains as:

$$\begin{cases} k_1 = -\lambda_1 \\ k_2 = -2\lambda_2 \\ k_3 = -(\lambda_2^2 + \lambda_3^2) \end{cases} \tag{10}$$

Finally, the robot linear and angular velocity:

$$\begin{cases} v = \frac{\tilde{u}_1 + u_1^d}{\cos\theta} \\ \omega = \frac{\tilde{u}_2 + u_2^d}{1 + \tan^2\theta} \end{cases} \tag{11}$$

IV. SYSTEM DESCRIPTION AND IMPLEMENTATION

In this section, we describe the system to implement OSEK/VDX. We divided the mobile robot into two parts (Fig. 2): low level and high level parts. The low level part (Fig. 3) controls the mobile robot with the desired velocity when it receives instructions from the high level part. It shows the current velocity of the mobile robot and also transmits the robot's real-time velocity to the high level part. The high level part (Fig. 4) is the user application that multitasks. Fig. 2 presents the system description.

These tasks are the trajectory algorithm, keypad scanning, liquid crystal display (LCD) updating, Bluetooth communication, and sending command messages to the first part.

The mission of the low level part is to control the desired velocity of the mobile robot. This part is implemented with OSEK/VDX, programmed in multitasks, and uses resources to protect the global variable. For driving a mobile robot, we must control the two direct current (DC) motors on the mobile robot.

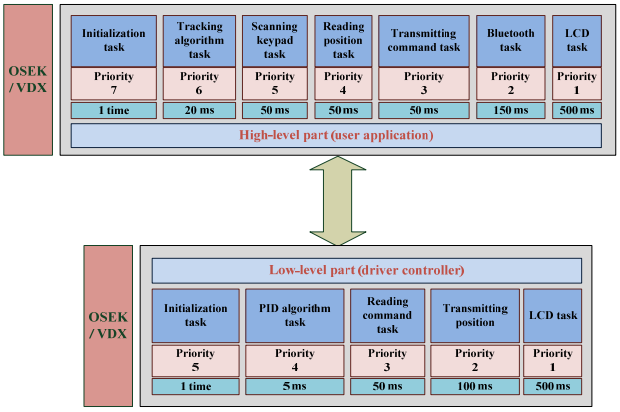

Fig. 2. System description. OSEK/VDX: open systems and the corresponding interfaces for automotive electronics, PID: proportional-integral-derivative, LCD: liquid crystal display.

We applied the proportional-integral-derivative (PID) velocity algorithm for two DC motors. Therefore, the PID algorithm task is the core task, has high priority, and is activated every 5 ms. This task follows two control logics: read the velocity command and execute the PID velocity algorithm. The second task receives the velocity commands via the universal asynchronous receiver/transmitter (UART) and is activated every 50 ms. Another task sends the real-time velocity to the high level part, and it is activated every 50 ms. The LCD task is activated every 500 ms to present the current velocities of the mobile robot. Obviously, we need the initialization task for setting the system before running. Most of the periodic tasks in the low level part are executed by the OSEK Alarm Counter. Only the initialization task is defined by AUTO-START.

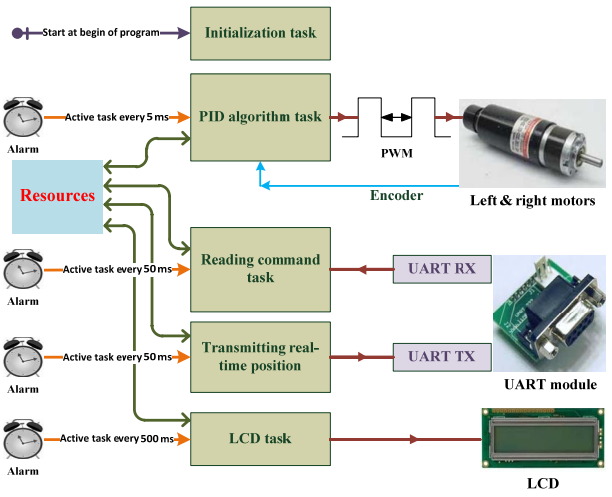

Fig. 3. Application diagram of low-level part. UART: universal asynchronous receiver/transmitter, LCD: liquid crystal display, PWM: pulse width modulation, PID: proportional-integral-derivative.

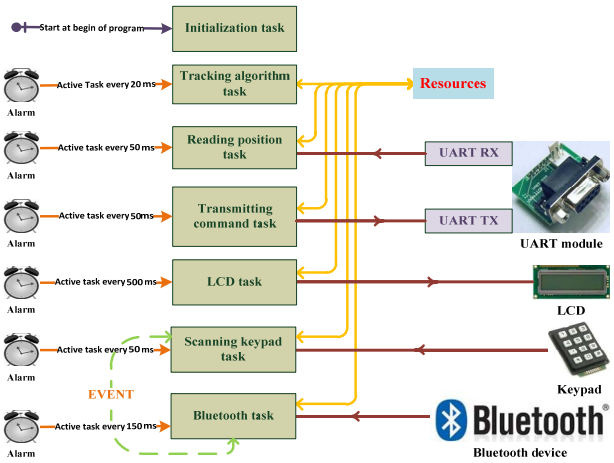

Fig. 4. Application diagram of high-level part. UART: universal asynchronous receiver/transmitter, LCD: liquid crystal display.

Thus, it is activated automatically and initially at the beginning of the program. After completing the system setup, this task is suspended and not triggered by any events, alarms, or tasks; it is only activated again when the system is rebooted to reestablish the parameters.

The high level part is designed for user applications. In this paper, this section is primarily used for trajectory tracking. This implementation is integrated with OSEK/VDX and programmed in multitasks. It has seven tasks in total: initialization task, tracking algorithm task, reading position task, transmitting command task, Bluetooth task, scanning keypad task, and LCD task. The initialization task is run automatically at the beginning to initialize all of the modules. The other tasks can be executed sequentially by the alarm counter after the initialization task finishes. The user application is implemented in the task that has the second priority. This two mode selections that the user can select for the task are the trajectory and remote control. The selection mode is recognized at the beginning of this task.

In the first mode, it reads the velocity commands from the Bluetooth task, and then saves them in global variables. The Bluetooth task only reads the velocity commands of the user via the Bluetooth protocol when the event is set from the scanning keypad task. The scanning keypad task reads the states of the buttons and sets the event for the Bluetooth task every 50 ms. The second mode is the desired trajectory. This mode reads the defined trajectory and then executes the algorithm. In this mode, it is necessary to know the current position of the mobile robot. Therefore, it must read the current positions from the global variables, which are calculated by the position task. The position task calculates the current position of the mobile robot from the real time velocity of the low level part, and then saves these values in global variables. After finishing the algorithm, the tracking

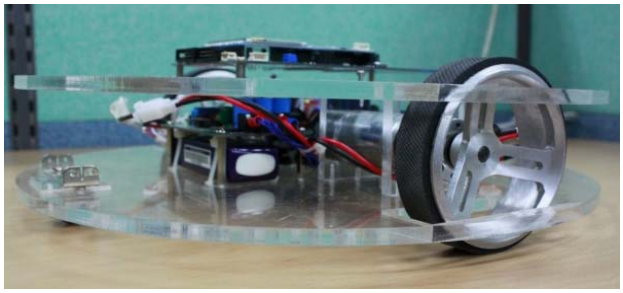

Fig. 5. Left view of the actual mobile robot.

algorithm task saves the velocity commands in global variables. The transmitting command task takes these commands from the global variables and sends them to the low level part every 50 ms via the UART. The execution time of the tracking trajectory algorithm task is 20 ms, which is short enough for smooth motion. Synchronizing between tasks, the resources and events are used to share and protect the global values between them. The LCD task is a background task which has the lowest priority and shows the state of the keypad, selection mode, etc.

V. SIMULATION AND EXPERIMENTAL RESULTS

The mobile robot in this paper has a two wheel differential drive, as shown in Fig. 5. This design has two motors, which are mounted on the left and right side and two passive wheels on the front and rear of the mobile robot for stability reasons.

In this section, we simulate and show the results of the experiment on the low level part using a LM3S8962 platform for experimental research. In this paper, we simulated and experimented on the low level part with constant inputs. Parameters listed below show the response of the DC motor used in the simulation (Table 1) and in the experiment on the actual physical device (Table 2).

In Fig. 6, the fast response of the velocity when using the PID algorithm can be seen. Within 0.5 sec, the mobile robot reaches the desired velocity.

Table 1. Simulation parameters of low level part

Parameter	Value
Wheel radius (m)	0.05
Half of the wheel distance (m)	0.17
$K_p0 = K_p1$	100
$K_d0 = K_d1$	10
$K_i0 = K_i1$	200
Linear velocity of mobile robot (m/sec)	0.39
Angular velocity of mobile robot (rad/sec)	0.46

Table 2. Real-time test parameters of low level part

Parameter	Value
$K_p0 = K_p1$	14.49
$K_d0 = K_d1$	1.25
$K_i0 = K_i1$	2.32
Linear velocity of mobile robot (m/sec)	0.39
Angular velocity of mobile robot (rad/sec)	0.46
Angular velocity of the right DC motor (rad/sec)	3π
Angular velocity of the left DC motor (rad/sec)	2π
Frequency of the DC motor (kHz)	15

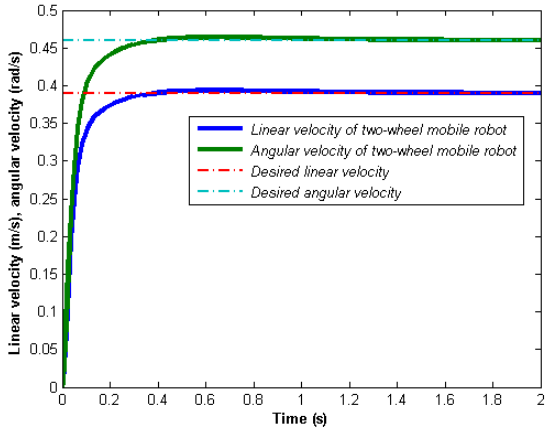

Fig. 6. The simulation results of the linear and angular velocities of the two-wheel mobile robot with the desired velocities.

The parameter for the PID algorithm is set differently for the real-time experiments than for the simulation because in the real environment, many factors affect the operation of the mobile robot. The environmental impact of friction creates a small vibration in the results of the linear and angular velocities of the mobile robot, as shown in Fig. 7.

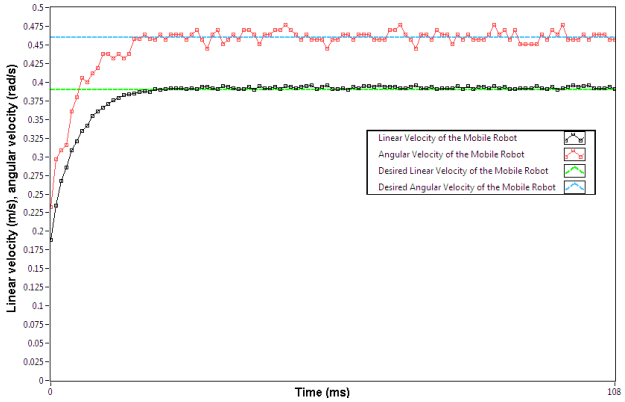

Fig. 7. The real-time results of the linear and angular velocities of the two-wheel mobile robot with the desired velocities.

VI. CONCLUSIONS

We successfully implemented the tracking of the desired trajectory by the linear and angular velocity of a mobile robot. The missions of this implementation were performed in an exactly defined time to ensure the speed of each wheel, and this mobile robot can run with minimizing errors to maintain the velocity of the mobile robot at the desired values. This research project provides a solution for some complex applications by dividing the system into several subsystem parts. Each part can be implemented by using the OSEK/VDX RTOS to execute a control algorithm in the same way.

Through the results of the simulations and real-time experiments above, this method is shown to be well executed, demonstrating the validity of the implementation on the mobile robot using OSEK/VDX RTOS. The contributions of this paper suggest a method for adapting and porting OSEK/VDX RTOS to a two-wheeled mobile robot with a differential drive. The implementation of the high level part and the improvement of the tracking performance require further research.

REFERENCES

[1] OSEK-VDX [Internet], Available: http://www.osek-vdx.org/.

[2] AUTOSAR: AUTomotive Open System ARchitecture [Internet], Available: http://www.autosar.org/.

[3] PICos18 [Internet], Available: http://www.picos18.com/index_us.htm.

[4] open OSEK [Internet], Available: http://www.openosek.org/tikiwiki/tiki-index.php.

[5] FreeOSEK [Internet], Available: http://opensek.sourceforge.net.

[6] K. M. Zuberi, P. Pillai, K. G. Shin, T. Imai, W. Nagaura, and S. Suzuki, “EMERALDS-OSEK: a small real-time operating system for automotive control and monitoring,” SAE International, Warrendale: PA, Technical Paper 1999-01-1102, 1999.

[7] nxtOSEK/JSP [Internet], Available: http://lejos-osek.sourceforge.net.

[8] TOPPERS [Internet], Available: http://www.toppers.jp/osek-os.html.

[9] ERICA Enterprise and RT-Druid [Internet], Available: http://erika.tuxfamily.org/.

[10] J. L. Bechennec, M. Briday, S. Faucou, and Y. Tringuet, “Trampoline: an open source implementation of the OSEK/VDX RTOS,” in *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation*, Prague, Czech, pp. 62-69, 2006.

[11] J. P. Laumond, *Robot Motion Planning and Control*, New York, NY: Springer, 2003.

Duy Le Nguyen
received a B.S. degree in Automotive Engineering from Hochiminh University of Technical Education, Hochiminh, Vietnam, in 2007, and an M.S. degree in Information and Communication Engineering from Korea University of Technology and Education, Cheonan, Korea, in 2011. He is currently pursuing a Ph.D. degree in Information and Communication Engineering at the Korea University of Technology and Education, Cheonan, Korea. His research interests include embedded systems, automotive systems, robotics, and real-time operating systems.

Myung-Eui Lee
received B.S., M.S., and Ph.D. degrees in Electrical Engineering in 1985, 1987, and 1991, respectively, from Inha University, Inchon, Korea. From December 1986 to September 1995, he served as a research engineer at Hyundai Electronics Industries Co., Ltd., Inchon, Korea. He joined Korea University of Technology and Education, Cheonan, Korea, in 1995, where he is presently a professor of Information and Communication Engineering. His research interests include digital control systems, system software design, and communication satellite systems.