2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)
August 19~22, 2016 at Sofitel Xian on Renmin Square, Xian, China

An EtherCAT-based Real-time Motion Control System in Mobile Robot Application

Raimarius Delgado[1], Shin-Young Kim[2], Bum-Jae You[2], and Byoung-Wook Choi[1*]
[1] Department of Electrical and Information Engineering, Seoul National University of Science and Technology,
232 Gongneung-ro, Nowon-gu, Seoul 01811, South Korea
(Tel : +82-2-970-6412; E-mail: {raim223, bwchoi}@seoultech.ac.kr)
[2] Center of Human-centered Interaction for Coexistence, 5 Hwarang-ro 14-gil, Seongbuk-gu,
Seoul 02792, Republic of Korea
(Tel : +82-2-958-7391; E-mail: {sini13, ybj}@chic.re.kr)

Abstract – This paper presents a real-time motion control system using EtherCAT protocol and its application on a differential drive mobile robot. The motion controller is designed from open-source components consisting of dual kernel approach using standard Linux and real-time extension of Xenomai, and the EtherCAT Master stack, IgH. In order to validate feasibility of the real-time system, timing analysis between the master and the slaves is performed in terms of periodicity of the cyclic task, jitter, and in-control execution time as test metrics. Furthermore, we conducted a convolution based trajectory planning algorithm that considers the physical limits of the mobile robot to generate periodic velocity commands following a curved path. Encoder data from each wheels is evaluated to guarantee the accuracy of the motion control system in Cartesian space.

Keywords – EtherCAT, Mobile robot, Xenomai, CANopen-over-EtherCAT, Real-time

1. Introduction

Nowadays, Ethernet-based fieldbus system as the standard physical communication layer is radically increasing worldwide in the fields of automation and control technology. In comparison with other fieldbuses, Ethernet is not optimized to send subsequent short messages and requires microprocessors at each node that entirely slows down the whole system [1].

EtherCAT is a real-time Ethernet protocol that is gaining popularity in rigorous automation. It offers various appealing features such as optimal usage of the Ethernet bandwidth for data transfers, short cycle times, and full compatibility with the standard Ethernet protocol [2-3].

Real-time performance is vital in controlling intelligent and dynamic systems, especially in the field of robotics. For example, a mobile robot driven with velocity commands is critical to failure in meeting deadlines between the task that generates velocity commands to travel from an initial position to the next sampling point and the task that sends these commands to the actuator [4].

In order to ensure successful embracement of EtherCAT in real-world applications, benchmarking the protocol using real-time mechanisms in cyclic tasks with bounded time constraints is required. Currently, there are different approaches in analyzing EtherCAT-based control systems. Cereia et.al [5] presented a comparative evaluation of a PC-based real-time EtherCAT Master on the fully pre-emptible Linux kernel and RTAI assessing the ability to support concurrent best-effort tasks without compromising real-time performance. Sung et.al [6] analyzed the end-to-end delay for EtherCAT control processes taking into account the time for in-controller processing, message delivery, and slave-local handling. However, these researches were not able to produce results regarding actual automation workload and focused more on analyzing communication timing rather than control application aspect.

The main purpose of this paper is to verify the expected timing accuracy from an EtherCAT-based real-time motion control system using open-source software to actuate a differential drive mobile robot along a planned path. This measurement is significant to serve as criterion for testing the efficiency of an EtherCAT system when integrated into a practical application, which is time critical. We developed a software architecture based on Xenomai [7], which is a dual-kernel real-time extension running alongside the standard Linux kernel to satisfy real-time requirements. The latest version of an open-source EtherCAT Master, IgH EtherCAT [8], was integrated on top of the real-time operating system to establish deterministic communication with servo drives supporting CANopen-over-EtherCAT (CoE) protocol.

Real-time response of the system is validated by performing a timing analysis with respect to the periodicity of the cyclic task determined through its actual cycle time, jitter, and execution time.

Finally, an established trajectory planning algorithm in [9] is conducted to generate velocity commands to actuate the mobile robot along a curved path. The velocity commands are sent periodically to the slaves and the encoder values are analyzed to validate accuracy of the mobile robot to track the reference path articulated in the Cartesian coordinate plane.

978-1-5090-0821-6/16/$31.00 ©2016 IEEE

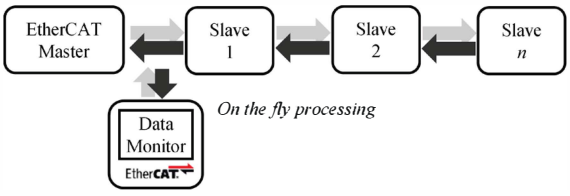

Fig. 1. Line topology of an EtherCAT automation system

2. EtherCAT Protocol

In comparison to other Real-time Ethernet Protocols, EtherCAT offers very high real-time performance and determinism developed by Beckhoff Automation [2]. The typical EtherCAT network supports single-master network configurations as shown in the linear topology in Figure 1, where the master communicates with the slaves by sending them suitable telegrams.

The EtherCAT master sends a telegram that passes through each node. Each EtherCAT slave device reads the data addressed to it on the fly, and inserts its data in the frame, as the frame is moving downstream. The frame is delayed only by hardware propagation delay. The last node in a segment detects an open port and sends the message back to the master using Ethernet technology's full duplex feature. The telegram's maximum effective data rate increases to over ninety percent, and due to the utilization of the full duplex feature, the theoretical effective data rate is even higher than 100 Mbit/s.

EtherCAT offers efficient clock synchronization for slaves down to hundreds of nanoseconds through special timing functionality, known as the Distributed Clock (DC) mechanism. The first slave connected to the EtherCAT Master serves as the reference clock for the entire network. Before activating the master, the controller calculates the offsets between the reference clock and the other DC-enabled slaves. The calculated offset is written to the slaves with which the value of the global clock is calculated based on the measurement at the local clock. The controller periodically distributes the value of the reference clock to all the slaves for the compensation of the clock skew caused by the local clock drift [5].

Although the DC mechanism is easily implemented, it can accurately synchronize the clocks of numerous slaves with a difference of less than one microsecond.

3. EtherCAT-based Real-time Motion Control System

3.1 System Structure

A typical motion control system is used in industrial processes to actuate and carry task to move a specific load to a target position. In mobile robot application, the motion control system should provide a main controller that generates motion paths from an initial point to a target position and send commands, a drive that links the main controller and the actuators, and actuators that convert the commands into mechanical motion. The choice of actuator type is based on the specified power, speed, precision, or torque.

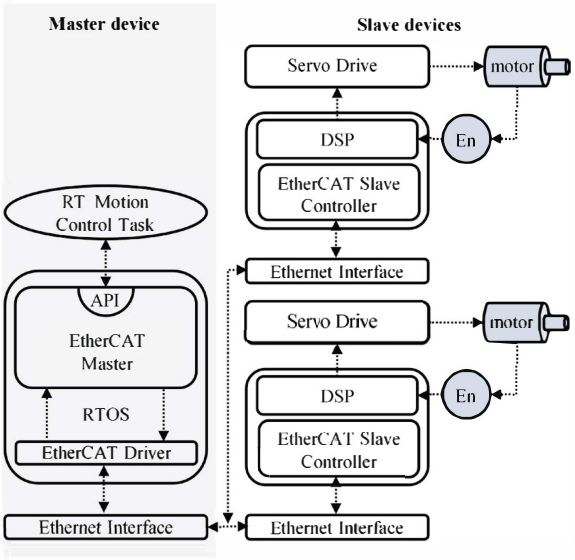

Fig. 2. System structure of a motion control system for a mobile robot

In order to meet such requirements, Figure 2 shows the overall system structure of a motion control system to actuate a mobile robot. The developed system consists of a main controller based on an EtherCAT Master supported device, EtherCAT slave devices with digital servo drives, and motors connected to absolute encoders, represented by the encircled "En" within the figure. Encoders are essential to ensure performance in feedback loop control. The dotted connectors represent the flow of data within the system, where a single-ended arrow signifies one-way communication and double-ended arrow both directions.

The main controller is responsible for generating and sending commands sent to the drives, running an EtherCAT Master instance under a real-time operating system (RTOS). The RTOS helps maintaining proper scheduling of different tasks handled by the main controller such as, receiving feedback from the drives, generating target commands, and sending these commands back to the drive. The servo drive contains an EtherCAT Slave Controller (ESC) that handles EtherCAT protocol by processing frames on the fly. An ESC provides interfaces for data exchange between the EtherCAT Master and the local application processed by a digital signal processor (DSP). These controllers are commonly implemented using field-programmable gate arrays or application-specific integrated circuit provided by Beckhoff Automation.

In our developed system, we used two servo drives supporting EtherCAT manufactured by LS Mecapion, L7NA004B, which uses CoE for communication. For each slave, process data objects are divided into 12 Bytes each for transmission and receipt, totaling to 24 Bytes of required data packets at each cycle. In addition, the servo drive supports processing of 19-bit serial encoder and provides three different control modes: velocity, position, and torque control. For the actuator, we attached a 400-Watt AC motor at each drive to serve as the left and right wheels of the mobile robot.

To avoid incompatibility with the servo drive, we have chosen a motor model from the same manufacturer, APM-FB04AMK, with a rated maximum speed of 3000 revolutions per minute. Each motor is equipped with a built-in 19-bit absolute encoder that is used for accurate feedback loop control required to ensure that the load reaches the target command.

3.2 Open-source Software Architecture

In the presented system structure, the main controller is developed to support EtherCAT protocol. The primary role of the main controller is to receive data from EtherCAT datagrams, process the received data to generate target commands, and send these commands back to the slaves. In order to perform such tasks, the main controller must be equipped with proper computing devices. We selected an i386 architecture-based Intel Atom D2700 single board computer (SBC) as the main controller. The SBC is assembled with two gigabytes of DDR3 SDRAM and a Realtek RTL8168 as the network interface controller. In order to ensure deterministic communication between the master and the slaves, real-time mechanisms are required by the EtherCAT Master stack.

In recent years, radical improvement of computing power of hardware coincides with the trend of shifting into using free general-purpose operating systems, especially Linux, opening new solutions for real-time applications. Although using the standard Linux kernel shows no difficulty in implementing applications with soft real-time requirements, operations with tighter timing constraints such as feedback loop control for trajectory tracking pose the opposite results. Techniques that aim to enhance the standard Linux to meet real-time requirements include pre-emptible kernel approach and dual-kernel approach.

Figure 3 shows the detailed software development environment for the main controller of the designed motion control system using the dual-kernel approach, Xenomai [7].

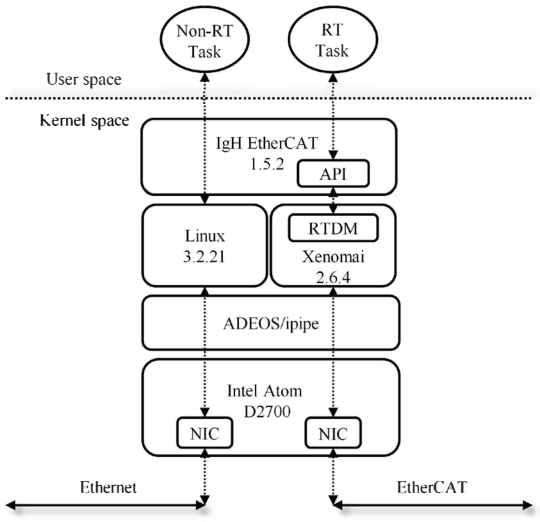

Fig. 3. Software architecture based on open-source EtherCAT Master on Real-time Linux

In contrast to previous researches in [5-6], we implemented later versions of each software components. First, the standard Linux kernel, version 3.2.21, was implemented to handle non real-time application. Since the Linux scheduler performs tasks and processes based on fairness rather than priority, Xenomai 2.6.2.1 is developed to run alongside the standard kernel using the adaptive domain environment for operating systems (ADEOS). To finalize the whole system, the latest version of an open-source solution, IgH EtherCAT Master 1.5.2 [8], is stacked on top of the real-time Linux kernel to be able to c communicate with the EtherCAT slave devices.

3.3 CANopen-over-EtherCAT

CANopen is a communication protocol and device profile specification for embedded systems, used in automation. In terms of the open systems interconnection model, CANopen implements the layers above and including the network layer.

The standard consists of an addressing scheme, several small communication protocols and an application layer defined by a device profile. The communication protocols have support for network management, device monitoring and communication between nodes, including a simple transport layer for message segmentation and de segmentation. The lower level protocol implementing the data link and physical layers is usually Controller Area Network. In this paper, we use slaves that incorporate CANopen profiles into EtherCAT resulting to CoE protocol.

Figure 4 shows the sequence diagram in developing a control application for a CANopen-based slave using the CoE protocol with IgH EtherCAT Master. First, the CoE protocol enables the complete The SDO protocol is used directly, so that existing CANopen stacks can be used practically unchanged. The sum of all registered PDO entries defines the PDO map.

An application can register PDO entries for exchange. Every PDO entry and its parent PDO is part of a memory area in the slave's physical memory, which is protected by a sync manager for synchronized access.

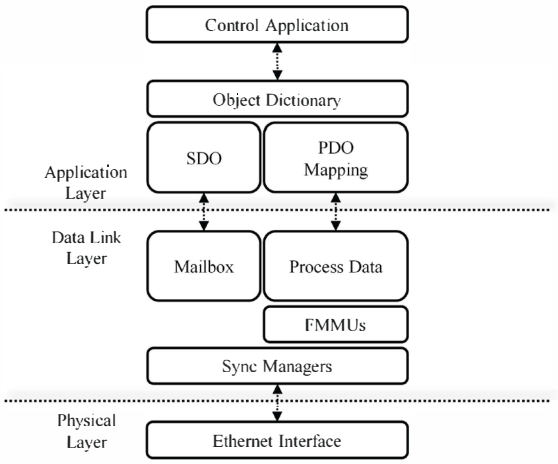

Fig. 4. Architecture of CANopen-over-EtherCAT device

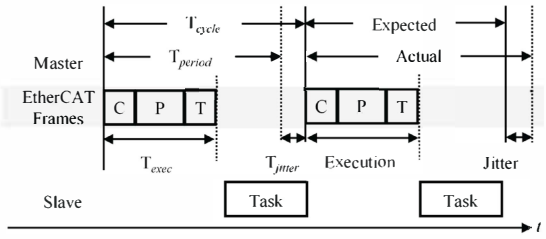

Fig. 5. Timing diagram for timing analysis of the EtherCAT-based control system

4. Experimental Results

4.1 Timing Analysis

In this paper, timing analysis is discussed by evaluating the periodicity of the cyclic control task using actual cycle time, jitter, and the execution time of transferring an EtherCAT frame from the master to the slaves within the network as graphically shown by the timing diagram in Figure 5. The time duration that it takes to acquire current data stored in the process image from the slave is represented by the block, C. Block P denotes processing time of the next command for the slave and T is the point when the processed command is transferred to the slave.

Execution time, T_{exec}, is defined as the time duration to receive the stored process image from the buffer of the network interface card, calculate the received data to generate the next command, and transmit the generated data from the master to the slaves. Overall execution time of the system varies on the delay that occurs depending on the number of slaves and the size of the process data objects denoted by k_n and k_s, respectively.

$$T_{exec} = k_n P + k_s (C + T) \quad (1)$$

The cycle of the periodic task is calculated using the timer management service, rt_timer_read, provided by the Xenomai Native skin API. This service serves as a probe that returns the system time at the calling point. The actual cycle time, T_{period}, is calculated by subtracting the stored value of the previous iteration to the acquired value of the current instance. The task release jitter, T_{jitter}, is calculated as the difference between the expected cycle time, T_{cycle}, and the actual cycle time, or:

$$T_{jitter} = \left| T_{cycle} - T_{period} \right| \quad (2)$$

Our experimental system is consisted of two EtherCAT-based servo drives with a total of 24 bytes of process data objects for each slave. The control task is bounded to a period of 10 ms for the timing analysis. The measurement is performed on the user space with the control task being the highest priority.

The experiment was conducted for 7.62 seconds producing 762 samples. The results of the test are shown in Table 1 classified into values of the average, maximum, minimum, and standard deviation of each timing metric shown in milliseconds.

Table 1 Summary statistics of the timing analysis

T_{period}	Timing Analysis (ms)
Average	10.00001
Maximum	10.01313
Minimum	9.991378
St. D.	0.003067
T_{exec}	
Average	0.043994
Maximum	0.109150
Minimum	0.034169
St. D.	0.010902
T_{jitter}	
Average	0.001775
Maximum	0.013128
Minimum	0.000002
St. D.	0.002501

Based on the results, the jitter of the system and the overall execution time for the EtherCAT Master is statistically under tolerable range in this particular development environment as compared to delay analysis of previous researchers. Although the results were sufficient to prove that, the system does not impose delay problems in communication between the master and the slave, performance in application for a mobile robot control system has not yet been verified.

4.2 Trajectory Tracking Control

Figure 6 shows the central position of the mobile robot in Cartesian space is represented by P_c (x_c, y_c, θ_c), where (x_c, y_c) denotes the coordinates of the position of the center and θ_c defines the heading angle measured counter clockwise from the x-axis. The radii of the wheels are represented by r and D denotes the linear distance between the wheels passing through the center of the mobile robot.

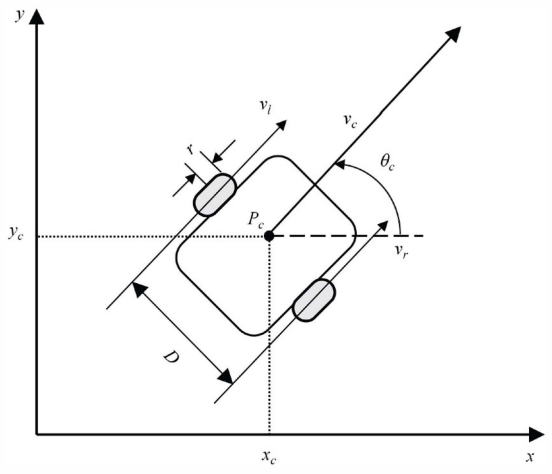

Fig. 6. Kinematics of a two-wheeled mobile robot

The kinematic equation of a two-wheeled mobile robot is obtained using the following equation:

$$P_c = \begin{bmatrix} \frac{r}{2}\cos\theta_c & \frac{r}{2}\cos\theta_c \\ \frac{r}{2}\sin\theta_c & \frac{r}{2}\sin\theta_c \\ \frac{r}{D} & -\frac{r}{D} \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \tag{3}$$

In this equation, the angular velocities which are the actual commands carried out by each of the motors of the differential drive mobile robot are represented by ω_r and ω_l for the right and left wheel, respectively.

From the central velocity, v_c, actual velocity commands for the mobile robot to track a planned path requires the angular velocities for each wheel. Solving Eq. (3) produces the equation:

$$\begin{aligned} \omega_r &= \frac{1}{r}\left(v_c + \frac{D}{2}\cdot\omega_c\right) \\ \omega_l &= \frac{1}{r}\left(v_c - \frac{D}{2}\cdot\omega_c\right) \end{aligned} \tag{4}$$

where ω_c represents the central angular velocity taken from the derivative of the heading angle at each point of the planned path with respect to a given sampling time, or:

$$\omega_c = \frac{d\theta_c}{dt} \tag{5}$$

Generated commands given to the main controller are in the form of the linear velocities for each wheel:

$$\begin{aligned} v_r &= r\omega_r \\ v_l &= r\omega_l \end{aligned} \tag{6}$$

An established study in [9] suggested a method based on convolution to generate velocity commands that could track a curved path. In this study, we generated a trajectory for the developed mobile robot to follow an S-curve.

The experiment was performed on an ideal environment where the wheels of the mobile robot are lifted to impose freewheeling motion to avoid disturbances that conveys uncertainties such as slip and friction. The velocity profile is generated in the user space and these commands are sent to the mobile robot using CoE protocol on a bounded period of ten milliseconds.

In this example, we generated a path from an initial point (0, 0, 0°) to the desired terminal point (4, 4, 0°), with both initial and terminal velocities at 0 m/s. The physical limits of the mobile robot required to generate velocity commands are based on the actual constraints of the actuators. According to the specifications, these limits are stated as 1 m/s for the maximum velocity limit, 0.4 m/s^2 for the maximum tangential acceleration, and 0.5 m/s^3, for the maximum jerk.

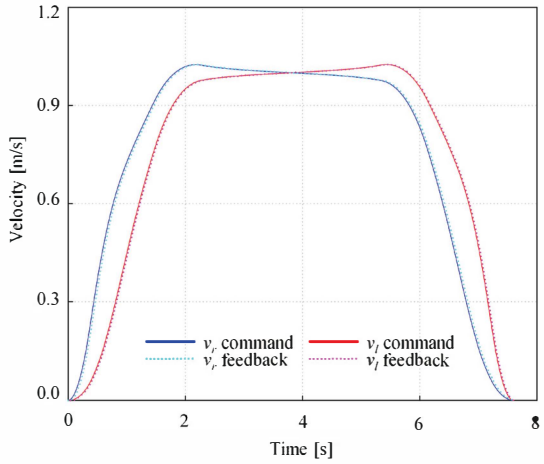

Fig. 7. Velocity commands and feedback along an S-curve

Using the given physical limits, the central velocity is produced using a convolution operator. Actual velocity commands for a differential mobile robot require velocity profiles for both wheels, which are calculated using Eq. (4) and Eq. (6) with the radius of wheels measured at 0.2 m and the length of the line between the wheels that also passes through the center of the main body is measured at 0.4 m.

Figure 7 shows the velocity profile for each wheel of the mobile robot that could track an S-curve. The solid blue and red lines labeled as commands denote the generated velocity for the right and left wheels, respectively. The corresponding encoder values from the mobile robot is acquired and compared with the calculated velocity shown as the dotted cyan and pink lines for the right and left wheels labeled as feedback.

The encoder results are analyzed using the robot kinematics to produce a trajectory in Cartesian coordinate plane in comparison to the reference path as shown in Figure 8. The planned path is represented by the solid blue line labeled as “Reference” while the results of the analysis are shown in dotted pink line labeled as “EtherCAT feedback”.

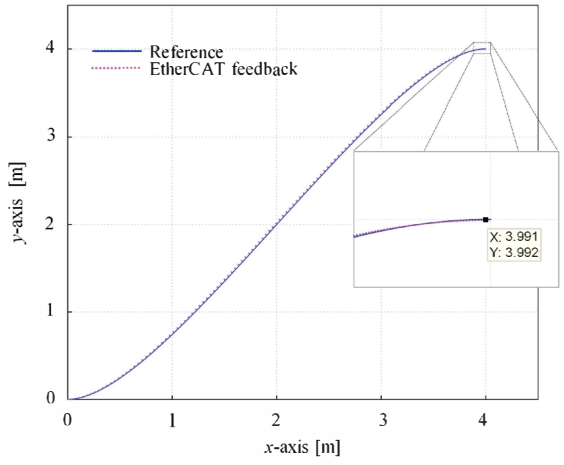

Fig. 8. Trajectory tracking in Cartesian space

Analysis of the encoder data shows that the mobile robot was able to track the planned path accurately with an error margin of less than 1 millimeter on both the x and y axes at the terminal point, (3.991, 3.992). While moving along the path, the calculated travel time of the mobile robot is obtained as 7.62 seconds and the integrated actual length of the planned path is calculated as 5.76 meters.

As shown in the results, some sampling points at the velocity profile for both the right and left wheels have reached values over the given maximum velocity limit, which could result to drift and tracking issues from the original planned path in practical navigation. These problems are beyond the scope of this paper since these would not occur when the mobile robot is exhibiting freewheeling motion. Thus, an accurate trajectory that could track the planned path was possible to be generated although the maximum velocity limit was not met. Moreover, we note that feedback control is also not adopted within this experiment since the focus of the test is to guarantee the real-time characteristics in using the EtherCAT protocol from open source software rather than performance of the control algorithm.

5. Conclusion and Future Work

In this paper, the performance of EtherCAT protocol in motion control application for a differential drive mobile robot has been evaluated by timing analysis and application of a proven trajectory generation technique.

First, an open source EtherCAT Master was implemented on an i386-based embedded board to send velocity commands to two EtherCAT-supported servo drives using CoE protocol. Then, timing analysis has been carried out aimed at evaluating the periodicity of the cyclic task on the EtherCAT master while connected to the two slaves using actual cycle time, jitter, and execution time as test criteria. Finally, performance analysis for control applications has been performed by generating velocity commands that could produce a trajectory to track a curved planned path.

Results of the timing analysis show that an open source EtherCAT Master could be operated as a motion control system with minimal jitter. Execution time is also tolerable in comparison with results from the analyses of other researches.

Validation of the proposed system structure in mobile robot application was conducted by performing experiment with two slaves that serve as the actuators of the mobile robot. Velocity commands are generated to track a curved path and the actual values from encoder results are compared to reference path in Cartesian space. The results of the analysis were able to validate that the EtherCAT-based motion controller was able to follow the reference path accurately with minimal error at the target point.

Further detailed analysis regarding the validity of the system in more complicated control application such feedback loop control is left as future work. Other performance metrics are also considered in future studies, for example, transaction time of the EtherCAT protocol.

Acknowledgement

This work was supported by the Human Resources Development of the Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Korea government Ministry of Trade, Industry & Energy. (NO. 20154030200720) and by the Global Frontier R&D Program on <Human-centered Interaction for Coexistence> funded by the National Research Foundation of Korea grant funded by the Korean Government(MSIP)(2010-0029759).

References

[1] T. Hu, P. Li, C. Zhang, and R. Liu, “Design and application of a real-time industrial Ethernet protocol under Linux using RTAI,” International Journal of Computer Integrated Manufacturing, Vol. 26, No. 5, pp. 429-439, 2012.

[2] http://www.ethercat.org/default.htm, EtherCAT Technology Group, 2016

[3] D. Orfanus, R. Indergaard, G. Prytz, and T. Wien, “EtherCAT-based platform for distributed control in high-performance industrial applications,” Proc of the IEEE Conf. on Emerging Technologies & Factory Automation, pp. 1-8, 2013.

[4] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, Robotics: Control, Sensing, Vision, and Intelligence, McGraw-Hill, New York, 1987.

[5] M. Cereia, I. C. Bertolotti, and S. Scanzio, “Performance of a real-time EtherCAT Master under Linux,” IEEE Transactions on Industrial Informatics, Vol. 7, No. 4, pp. 679-687, 2011.

[6] M. Sung, I. Kim, and T. Kim, “Toward a holistic delay analysis of EtherCAT synchronized control processes,” International Journal of Computers Communications and Control, Vol. 8, No. 4, pp. 608-621, 2013.

[7] http://xenomai.org/, Xenomai Project, 2016

[8] http://www.etherlab.org/en/ethercat/, IgH EtherCAT Master, 2016

[9] G. J. Yang, R. Delgado, and B. W. Choi, “Convolution-based time optimal path planning for a high-curvature Bezier curve considering possible physical limits,” International Journal of Control and Automation, Vol. 8, No. 9, pp. 325-336, 2015.