International Journal of Control and Automation
Vol.7, No.9 (2014), pp.189-200
http://dx.doi.org/10.14257/ijca.2014.7.9.16

Implementation of Joint Space Trajectory Planning for Mobile Robots with Considering Velocity Constraints on Xenomai

Gil Jin Yang and Byoung Wook Choi*

Department of Electrical and Information Engineering
Seoul National University of Science and Technology, Seoul, South Korea
{gjyang, bwchoi}@seoultech.ac.kr

Abstract

This paper presents an implementation of trajectory planning method while considering velocity constraints for a two-wheeled mobile robot (TMR) on Xenomai. A Bezier curve is utilized to make a smooth path. A convolution operator was used to generate a velocity profile for the travel distance of the planned path while also giving consideration of the velocity constraints of the TMR. The trajectory planning are performed on Xenomai with multitasking structure, which is a real-time Linux system developed by open-source projects. Finally, the periodicity and schedulability of real-time tasks and trajectory tracking performance were analyzed. Moreover, the efficiency of real-time operating system was also examined.

Keywords: *Mobile robot, Real-time Operating System, Xenomai, Bezier curve, Trajectory*

1. Introduction

Embedded systems can control certain hardware using a microprocessor. Embedded systems, together with the use of a general-purpose operating system, are rarely capable of achieving expected outputs within predetermined time constraints [1]. On the other hand, real-time systems are capable of doing tasks within a predefined time frame [2].

A navigation system for a TMR consists of a path planner, a trajectory generator and a tracking controller. Path planning is about generating smooth paths while maintaining the desired position in workspaces. The trajectory generator's task is to generate a velocity profile for the planned paths as a function of time. The tracking controller and the driving controller allow a TMR to travel along a predefined trajectory at a desired time while staying within its physical limits [3, 4, 5, 10].

Usually, a mobile robot gives commands to a controller periodically. If a system could not handle these periodic commands then the output is not expected to show proper performance. In this paper, one of the real-time development frameworks named Xenomai, was used to build a navigation system for the TMR rather than a general-purpose operating system. Xenomai is appropriate to achieve expected outputs within predetermined time constraints through giving periodic velocity commands to the robot controller [6]. In Xenomai, the periodic tasks and multitasking systems are easily developed with real-time mechanisms. The periodicity of real-time tasks and trajectory tracking performance was very important on the overall system performance so these results were analyzed.

* Corresponding Author

ISSN: 2005-4297 IJCA

In this research, real-time preemptive priority-based multitask scheduling was designed. Preemptive priority-based multitask scheduling system permits preemption of tasks wherein tasks are timely constrained and which can be suspended for a while based on the predefined priority of the task that could be resumed afterwards [7].

2. Trajectory Planning Considering the Velocity Constraints

When planning a path for TMR, the position and direction angle at its initial and target point should be considered. In this research, a path is generated using a Bezier curve to make a smooth path. Previously, Bezier curves were used in path planning and the convolution operator was also used for considering the physical limits of the TMR [3-5].

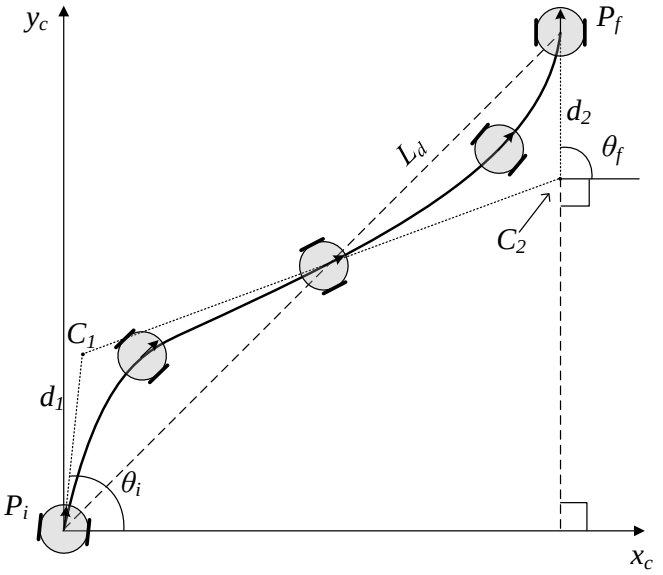

Figure 1. Bezier Curve

As shown in Figure 1, a Bezier curve consists of an initial point $P_i(A_0, B_0)$, a target point $P_f(A_3, B_3)$, and control points $C_1(A_1, B_1)$ and $C_2(A_2, B_2)$. An equation for the Bezier curve is calculated using C_1 and C_2. The equation of Bezier curve is given below in equation (1). Formerly, Bezier curve-based path planning were studied to determine the control points [3-5].

$$\begin{aligned} x(u) &= A_0(1-u)^3 + 3A_1u(1-u)^2 + 3A_2u^2(1-u) + A_3u^3 \\ y(u) &= B_0(1-u)^3 + 3B_1u(1-u)^2 + 3B_2u^2(1-u) + B_3u^3 \end{aligned} \quad (1)$$

In equation (1), u is an arbitrary value where $0 \leq u \leq 1$ and can be used to generate a smooth curve from a starting point to a target point: a more precise Bezier curve with a smaller increase. The path given by equation (1) does not consider velocity and is only parameterized by u.

To include the velocity of the physical limits, a convolution operator was used to create a central velocity $v_c(t)$ of TMR [4]. However, the velocity constraints of both wheels are not considered for the TMR. In other words, for any position $(x(u_i), y(u_i))$,

the robot travels with velocity $v_c(t_i)$ as a center velocity of TMR. In order to consider the positions in task space that depend on velocities in paths and direction angles, the parameter $u(t)$ of Bezier curve for the distance during the sampling time should be determined and calculated using equation (2) [5].

$$S_b = \sum_{u=0}^{1} \Delta\rho(u) = \sum_{u=0}^{1} \sqrt{(x(u+\Delta u) - x(u))^2 + (y(u+\Delta u) - y(u))^2}$$
$$u(t) = \frac{\sum_{t=0}^{T} v_c(t)}{S_b} \quad (2)$$

The trajectory for the TMR was generated so that it satisfies the physical limits described above and allows the TMR to travel along the curved path using its central velocity. An actual command for actuating TMR is the angular velocity for both wheels. Additionally, it can generate wheel velocity commands in joint space using the equations in [5].

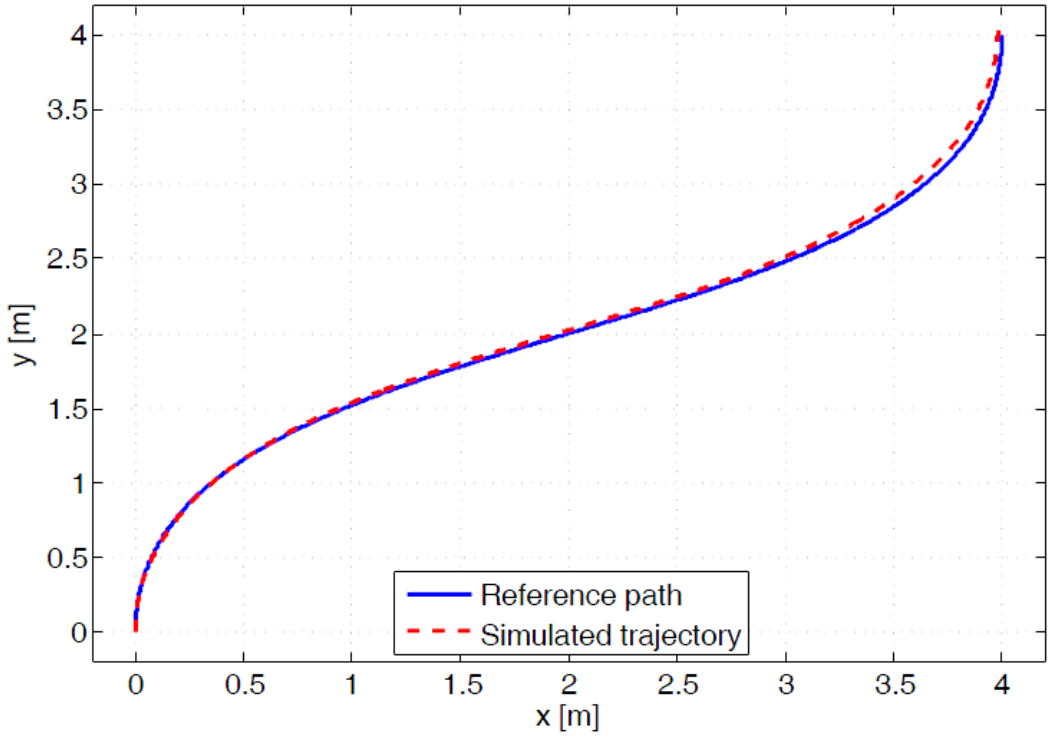

Figure 2. Reference Path and the Simulated Trajectory

Figure 2 shows the Bezier curve with starting point (0, 0, 90°) and target point (4, 4, 90°) and simulated trajectory which is recalculated using the two wheels' velocity profile in Figure 3.

Figure 3 shows the central velocity and velocity commands of the two wheels that satisfies the physical constraints using the proposed trajectory generator, v_{max} = 0.5 m/s, a_{max} = 0.2 m/s^2, j_{max} = 0.2m/s^3 following the Bezier curve in the Figure 2.

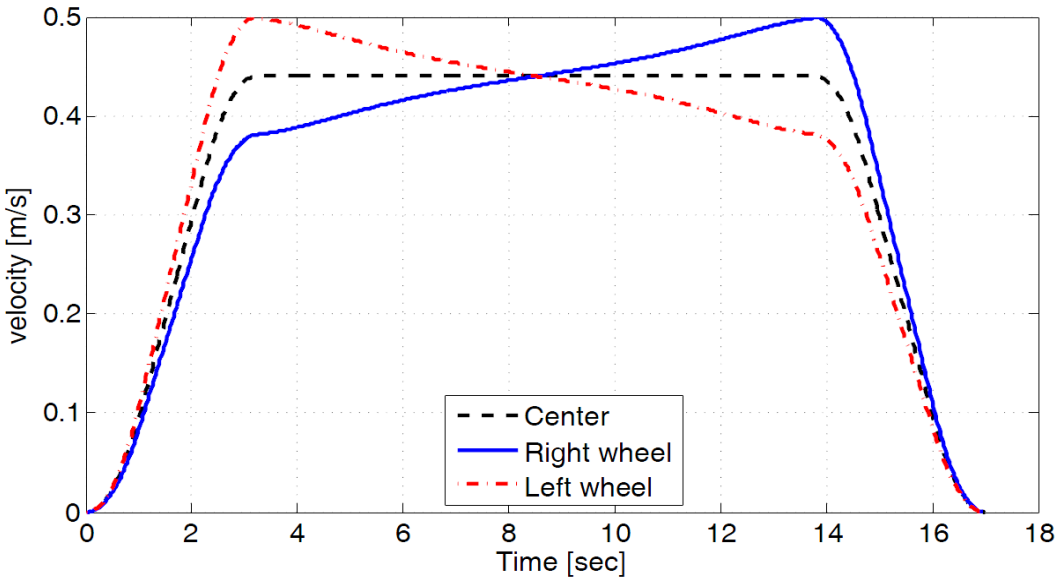

Figure 3. Joint Velocity Commands Following Curved Trajectory

3. System Architecture

The research was implemented on top of Xenomai which interfaces real-time tasks. Therefore, an OS is needed to use it. In Xenomai, the Linux OS kernel is treated as an idle task, and it only executes when there are no other real-time tasks to run. Figure 4 shows the architecture and versions of the real-time Linux used in this paper. Xenomai uses a general-purpose Linux kernel and a real-time API.

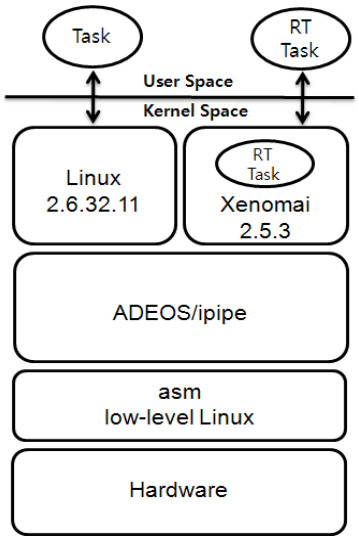

Figure 4. Architecture of Real-time Linux: Xenomai [2]

An application to control the TMR is performed with three tasks— Servo_Task, Velocity_profile_Task and Odometer_Task, which are real-time tasks on Xenomai. Servo_Task gives the velocity commands computed by velocity profile to the motors

and Odometer_Task gets the encoder data. Each task has its own priority. The highest priority is Servo_Task, followed by Velocity_profile_Task and Odometer_Task respectively. Velocity_profile_Task and Servo_Task communicate through an event flag group which is one of inter-task communication mechanisms. Servo_Task is executed after completing the computation for the velocity profile. The Odometer_Task and Servo_Task use mutex to avoid deadlock because for using the same control line to access the device.

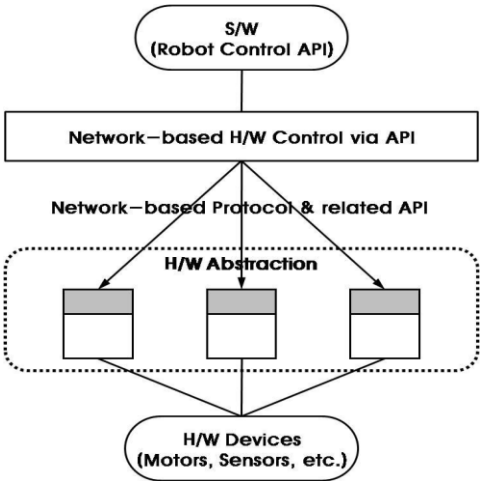

Figure 5. Architecture of DSSP-HAL Services for Control the TETRA DS-III

TETRA DS-III, a mobile robot developed by Dongbu Robot Co., Ltd. was used in this paper [11]. Figure 5 shows a control system of TETRA DS-III which is called "DSSP-HAL". DSSP-HAL is an XML-type network based abstract API that features independence over other hardware and operating systems that can access the devices that are installed in the mobile robot. Thus all devices are accessed through a TCP/IP network. The motor driver is implemented with velocity commands. Figure 6 represents hardware platform wherein Xenomai is installed on top of DSSP-HAL.

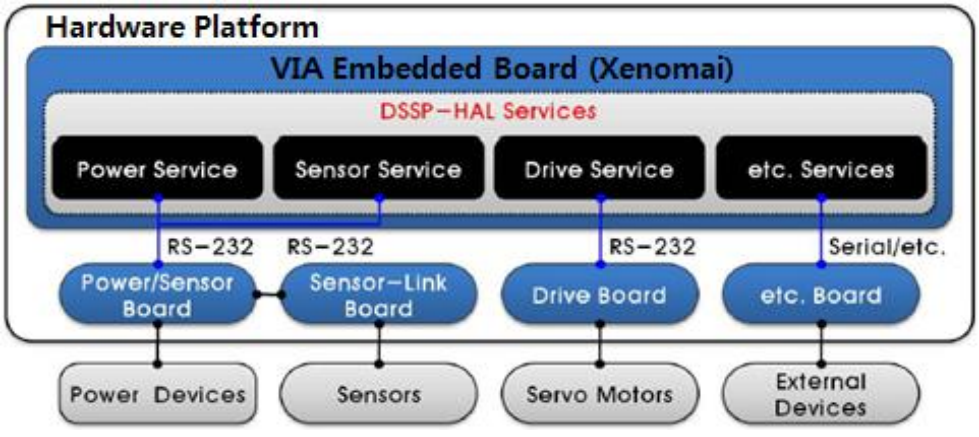

Figure 6. Hardware Platform of TETRA DS-III

The control of the mobile robot is operated by sending RS-232 commands to each device boards using the DSSP-HAL services located at the main board *VIA Embedded Board*.

```
set period and make periodic
Procedure Servo_Task
  while(1)
    read start time;
    grab the mutex lock;
    set velocity using the DSSP-HAL;
    release the mutex;
    read end time;
  end
  wait period;
end
```

Figure 7. Pseudo Code for *Servo_Task*

In case of controlling the mobile robot through its drive system that receives velocity command inputs, periodic velocity commands should be generated. Furthermore, these commands are also needed to be input to the given respective period. Figure 7 shows a periodic task being created for this research. In this process, DSSP-HAL service was used to send velocity commands to the drive system. Mutex was also used to prevent Priority Inheritance, one of the single network data transmission processes.

```
set period and make periodic
Procedure Delay_Task
  set EXECTIME to 5 ms;
  set SPINTIME to 1 ms;
  while(runningtime < EXECTIME)
    read start time;
    grab the mutex lock;
    rt_timer_spin(SPINTIME);  // spin cpu doing nothing;
    set runningtime = runningtime + SPINTIME;
    release the mutex;
    read end time;
  end
  wait period;
end
```

Figure 8. Pseudo Code for *Dummy_Task*

Figure 8 shows a Dummy Task that was used to analyze the real-time characteristics of the real-time operating system. The total execution time of the Dummy Task is 5ms. Therefore, an arithmetic operation that takes 1ms was performed five times. The CPU is kept busy by the function rt_timer_spin.

4. Experimental Results

The aim of this experiment is to analyze the performance for the implemented trajectory planning method on real-time operating system. In the experiment, a friction force between the ground and wheels and slip of the wheels were ignored.

The time required computing the path and trajectory planning at the Velocity_profile_Task takes approximately 3ms. The number of sample is about 870 and the sampling time is set to 20ms. The period of the Servo_Task is the same with the sampling time. The path is generated from (0, 0, 90°) to (4, 4, 0°) along the Bezier curve while considering the two wheels' velocity constraints. Figure 9 shows the result of the experiment. This is the outcome when the joint periodic velocity commands were input to the TMR drive system that uses the velocity profiles of both the mobile robot's wheels in real-time framework, Xenomai and non-real time Linux-based system experiments. The average period of the Servo_Task on the Xenomai is approximately 19.995ms and the average delay is approximately -5.43us. On the other hand, the average period of the Servo_Task on the Non-real-time is 25.32ms and the average delay is approximately +5.32ms. In Xenomai, a real time system, preemptive scheduling is considered to be an ordinary task. Linux, on the other hand, provides this scheduling only under certain conditions.

Non provision of preemptive scheduling could result to the occurrence of priority inversion as shown in Figure 9. In this graph, an error took place in the travel trace of the TMR following the delay of Servo_Task.

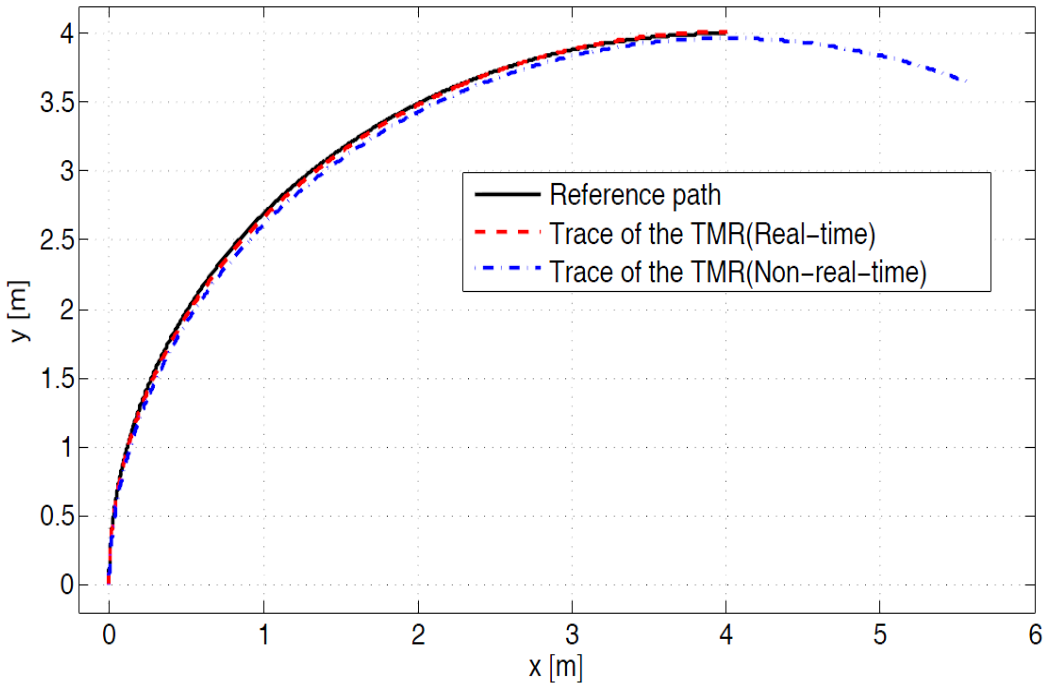

Figure 9. Reference Path and the Travel Trace of the TMR

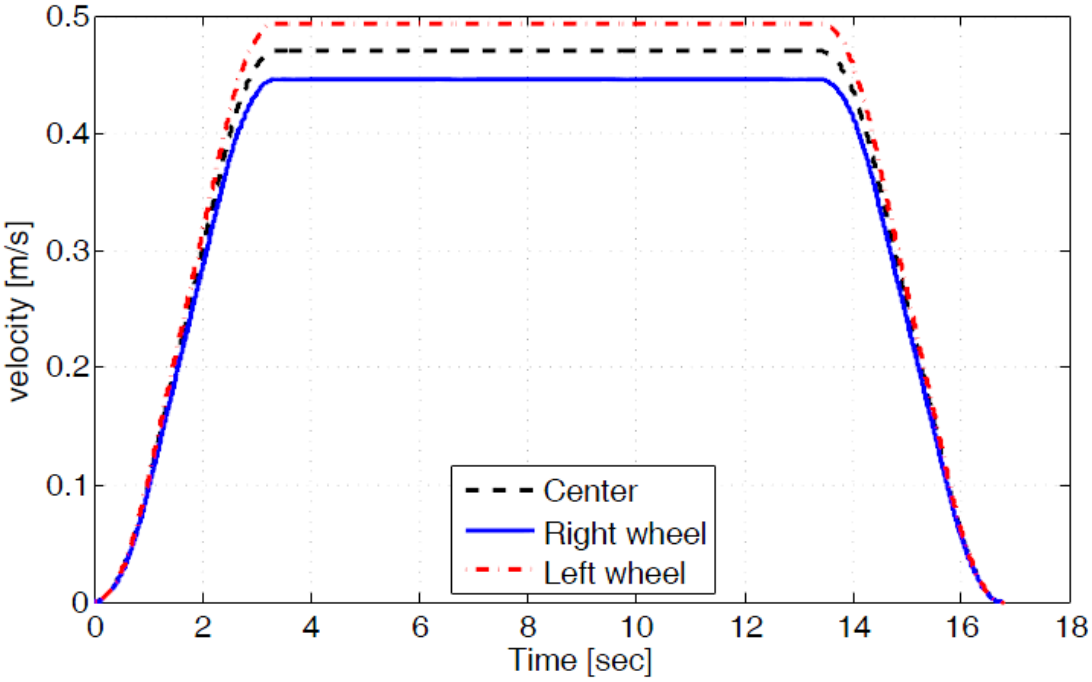

Figure 10. Joint Velocity Commands Following Curved Trajectory

Table 1. Task Computation Time and CPU Utilization

Task	Period	Average computation time	Average CPU utilization
Servo	20 ms	3.04 ms	0.152
Odometer	20 ms	8.02 ms	0.401
Delay	30 ms	5.04 ms	0.168
		Total	0.721

Figure 11 displays the task status of the mobile robot from 0.05s to 0.15s upon movement. In this experiment, Servo_Task was set to have the highest priority and Odometer_Task was set next in line. Also, in order to analyze the periodic characteristic of the task priority, Dummy_task, which is a busy wait burning CPU cycle, was set to the lowest. Figure 8 shows the pseudo code for the Dummy_task. In here, the period is set to 30ms and in an approximately 5ms execution time, the CPU was spun with Mutex every 1ms. Therefore, in Figure 11, one period of operation ended every time the Dummy_task was switched 5 times.

Servo_Task will start running upon computation of the velocity command for both wheels of the TMR. The resulting period of the Servo_task and Odometer_task was close to the set period but, the lowest priority Dummy_task shows data delay due to preemptive task. This result shows that in 0. 063003sec and 0. 0930sec timeframe, the Dummy_task was not able to spin the CPU and the context switching was done while in the pending status.

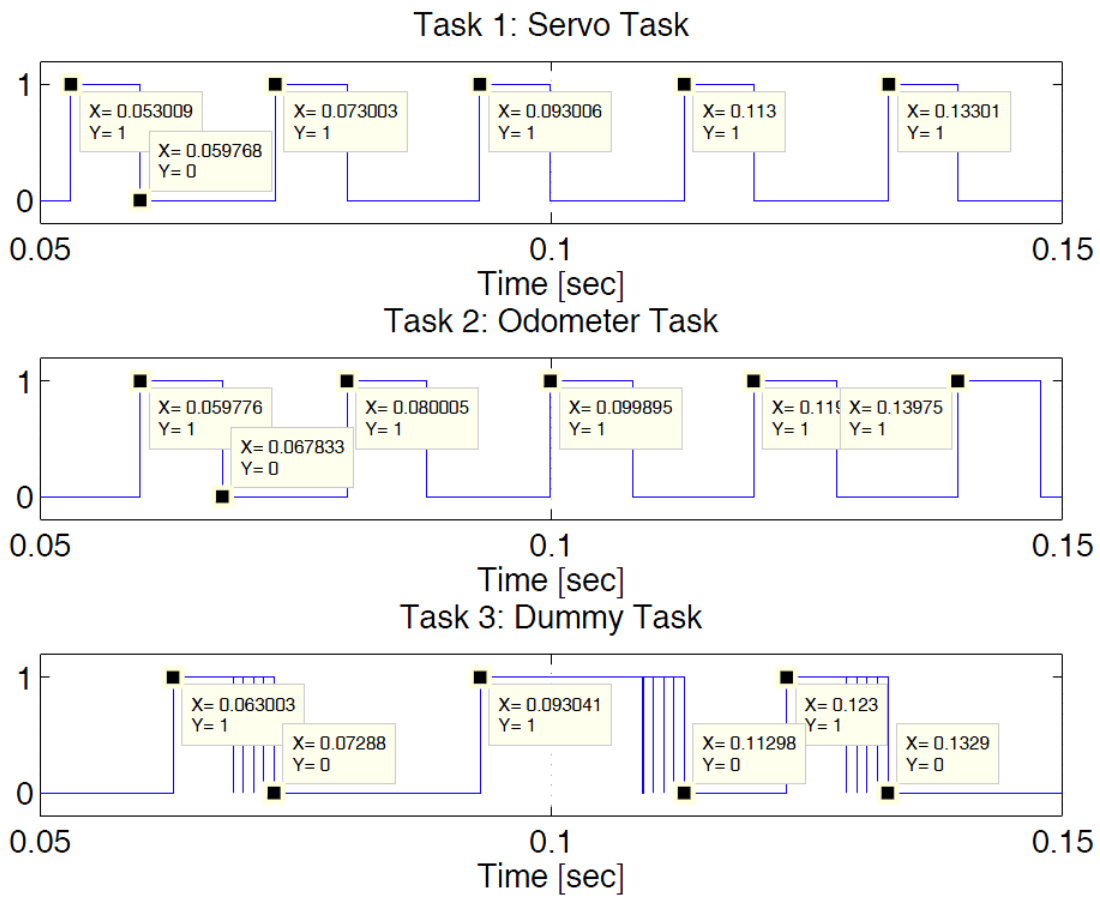

Figure 11. Task Status

5. Real-time Schedulability

Real-time systems have deterministic guarantees with regard to response times. Various method of the schedulability test of a real-time system is being researched as of the moment [7-9]. In this study, the Rate-monotonic scheduling (RMS) and Response Time (RT) Test were used to analyze schedulability and verify the real-time characteristic of the real-time system.

$$U = \sum_{i=0}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1) \qquad \textbf{(3)}$$

Liu & Layland proved that for a set of n periodic tasks with unique periods, a feasible schedule that will always meet deadlines exists if the CPU utilization is below a specific bound [8]. The schedulability test for RMS is represented in equation (3). Where C_i is the computation time, T_i is the period, and n is the number of task to be scheduled. In previous experiment three tasks were used for control the mobile robot on Xenomai. The CPU utilization U was computed. From the result U = 0.721 and the RMS bound for the three task is 0.77976, the system is said to be schedulable since U < RMS bound.

Table 2. Task Computation Time and CPU Utilization

Task	Period	Average computation time	Average CPU utilization
Servo	20 ms	3.04 ms	0.152
Odometer	20 ms	8.02 ms	0.401
Dummy	30 ms	10.04 ms	0.334
		Total	0.887

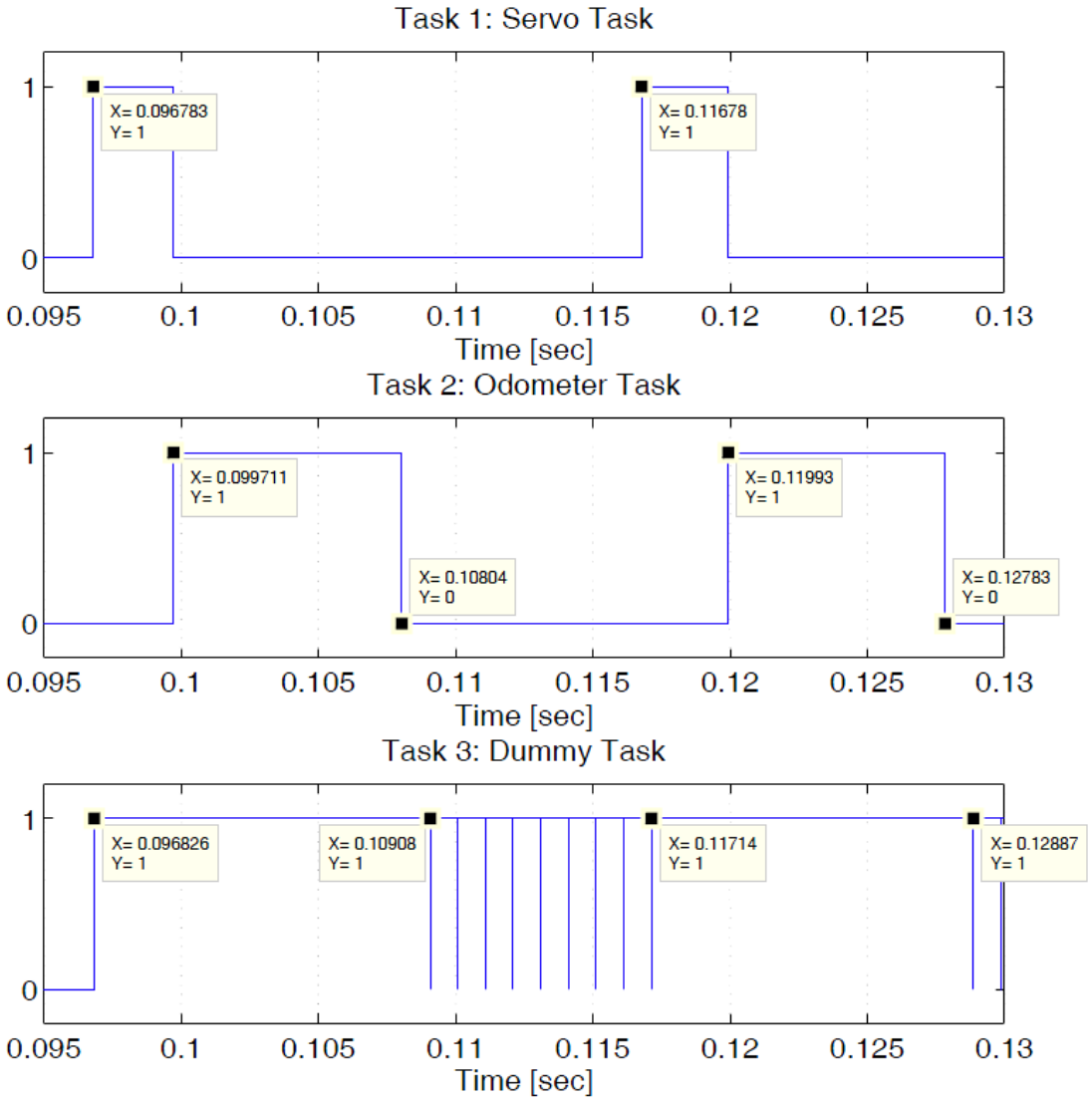

Figure 12. Task Status

Table 2 shows the computation time of the Dummy_Task that was increased by 10ms as opposed to Table 1. The SPINTIME of the Dummy_Task was executed by switching 10 times within 1ms that ends the operation. The RMS test shows that U = 0.887 and that U > RMS bound which made the system unschedulable. Figure 12 shows that when the TMR was driven using the tasks in Table 2. It also portrays the task status of the mobile robot from 0.095s to 0.13s upon movement.

Although the RMS test result is unschedulable and that in the Dummy_Task, the supposed operation was not performed in the given 30ms period, the Servo_Task and Odometer_Task performed close, if not the same with each task's preset period. At the same time, RT Test also, referred to as "Exact Schedulability Test" or "Completion Time Test" is needed to be carried out in order to determine the schedulability of each task.

RT test can be used for computing response times with any fixed-priority preemptive scheduling scheme and a_k^i is the worst-case response time for task τ_i and a_k^i of τ_i may be computed by the following equation (4):

$$a_{k+1}^i = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{a_k^i}{T_j} \right\rceil C_j \quad where\ a_0^i = \sum_{j=1}^{i} C_j \tag{4}$$

The test terminates when $a_{k+1}^i = a_k^i$ and the task i is schedulable if its response time is before its deadline, $a_k^i \leq D_i$. The result of RT test for table 2 is shown in equation (5).

$$a_0^3 = \sum_{j=1}^{3} C_j = 3.04 + 8.02 + 10.04 = 21.10$$

$$a_1^3 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{a_0^3}{T_j} \right\rceil C_j = 10.04 + \left\lceil \frac{21.10}{20} \right\rceil 3.04 + \left\lceil \frac{21.10}{20} \right\rceil 8.02 = 32.16 \tag{5}$$

$$a_2^3 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{a_1^3}{T_j} \right\rceil C_j = 10.04 + \left\lceil \frac{32.16}{20} \right\rceil 3.04 + \left\lceil \frac{32.16}{20} \right\rceil 8.02 = 32.16$$

The result of RT Test is $a_2^3 = a_1^3$ and in the second period for task 3, the deadline is 30. Therefore, the schedulability condition, $a_1^3 \leq D_3$, is not satisfied meaning that task 3, hereon `Dummy_Task`, is unschedulable. However, Task 1, `Servo_Task`, is schedulable and in spite of the Dummy Task, it keeps well in period with the velocity command. This means that velocity command is able to appropriately follow the predefined path of TMR.

6. Conclusion

In this paper, we implemented the proposed trajectory planning with consideration of the physical limits on real-time Linux, Xenomai. Through experiments we verified that the proposed trajectory planning was useful to control the TMR within physical constraints. The effect of periodicity of velocity commands was examined and we conclude that the proposed system is practical and shows outstanding performance in the sense of following a trajectory. By adding low priority tasks, the system is extendable without losing the periodicity of `Servo_Task`. Currently, the developed control system is an open loop system so further research should be done to consider obstacles where the trajectory should be modified on-line. Each tasks' performance should be evaluated whether the tasks meet or miss the deadline in that case we have to develop any feedback system to consider encoder data corresponding to the velocity commands.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea Grant funded by the Ministry of Education (NRF-2012R1A1A2006057).

References

[1] P. A. Laplante, "Real-time System Design and Analysis", Wiley-IEEE Press, **(2004)**.
[2] J. H. Koh and B. W. Choi, "Real-time Performance of Real-time Mechanisms for RTAI and Xenomai in Various Running Conditions", International Journal of Control and Automation, vol. 6, no. 1, **(2013)**, pp. 235-245.
[3] K. G. Jolly, R. S. Kumara and R. Vijayakumara, "A Bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits", Robotics and Automation Systems, vol. 57, **(2009)**, pp. 23-33.
[4] G. Lee, J. Kim and Y. Choi, "Convolution-Based Trajectory Generation Methods Using Physical System Limits", J. Dyn. Sys., Meas., **(2013)**.
[5] G. J. Yang and B. W. Choi, "Smooth Trajectory Planning Along Bezier Curve for Mobile Robots with Velocity Constraints", International Journal of Control and Automation, vol. 6, no. 2, pp. 225-234, **(2013)**.
[6] G. J. Yang and B. W. Choi, "Joint Space Trajectory Planning on RTOS", J. Korean, Inst. Intell. Syst., vol. 24, no. 1, **(2014)**, pp. 52-57.
[7] A. Abdelli, "A Much Compact Abstraction of the State Space of Real time Preemptive Systems", International Journal of Advanced Science and Technology, vol. 27, **(2011)**, pp. 45-58. [8] C. L. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment", Journal of the ACM, vol. 20, no. 1, **(1973)**, pp. 46–61.
[8] R. I. Davis, A. Zabos and A. Burms, "Efficient Exact Schedulability Tests for Fixed Priority Real-Time Systems", IEEE transactions on computers, vol. 57, no. 9, **(2008)**, pp. 1261-1276.
[9] G. J. Yang and B. W. Choi, "Trajectory Planning for Mobile Robots with Considering Velocity Constraints on Xenomai", Advanced Science and Technology Letters, vol. 49, **(2014)**, pp. 1-5.
[10] Dongbu Robot Co., Ltd., "http://www.dongburobot.com/"

