# Real-time Performance of Real-time Mechanisms for RTAI and Xenomai in Various Running Conditions

Jae Hwan Koh and Byoung Wook Choi*

*Dept. of Electrical and Information Engineering*
*Seoul National University of Science and Technology, Seoul, Korea*
*go3167@gmail.com, bwchoi@seoultech.ac.kr*

***Abstract***

*For a real-time system, the system correctness depends not only on the correctness of the logical result of the computation but also on the result delivery time. Real-time Operating System (RTOS) is widely accepted in designing real-time systems. The real-time performance is achieved by using real-time mechanisms through data communication and synchronization of inter-task communication (ITC) between tasks. Therefore, benchmarking the response time of real-time mechanisms is a good measure to predict the performance of real-time systems. This paper aims to analyze the response characteristics of real-time mechanisms in kernel and user space for real-time embedded Linux: RTAI and Xenomai. The performance evaluations of real-time mechanisms depending on the changes of task periods and load are also conducted in kernel and user space. Test metrics are jitter of periodic tasks and response time of real-time mechanisms including semaphore, real-time FIFO, Mailbox and Message queue. The results are promising to estimate deterministic real-time task execution in implementing real-time systems using RTAI or Xenomai.*

***Keywords:*** *Real-time Mechanisms, RTAI, Xenomai, Response Time, Various Running Conditions*

## 1. Introduction

Embedded systems can control certain hardware using a microprocessor. They are now used in various fields. However, a general-purpose operating system (OS), which is widely adopted in embedded systems, is rarely capable of achieving expected outputs within predetermined time constraints as it cannot guarantee real-time results. Therefore, embedded systems require a Real-time Operating System (RTOS) that can achieve reliable outputs against asynchronous responses by using the predictable response of methods within defined time constraints. RTOS has two main categories: a commercial RTOS and open-source real-time embedded Linux [1-4].

Typical RTOSes for general use include VxWorks, Nucleus PLUS, QNX, and uC/OS-II. These OSes have proven performance, valuable development environments, and systematic technical support as their advantages, while typical disadvantages are high initial development cost and low development flexibility [5]. In contrast, open-source OSes are widely applied to system development due to high compatibility with wide hardware support and high performance [6-9].

* Corresponding author

RTAI and Xenomai are common real-time embedded Linux systems developed by open-source projects. These two have several benefits. They can be built at a lower cost and provide knowledge from the developers at the users' fingertips since they were developed as open-source. These systems also do not lag behind commercial RTOSes in terms of performance. Despite these advantages, research on open-source OS has been limited due to the unavailability of detailed documentation and a lack of technical support, implying that further dedicated studies should be made for the use of open-source OS.

Real-time systems are generally implemented using multiple tasks. Communication, synchronization, and resource management between tasks are performed through real-time mechanisms [11]. Therefore, the performance of real-time systems can be determined by the time responses of real-time mechanisms. Thus, benchmarking the time characteristics of real-time mechanisms is really important to estimating the deterministic real-time performance. In terms of time characteristics, many studies have examined general-use RTOSes, while research on embedded Linux has focused on the control and handling capabilities in response to interrupts of systems based on Xenomai or RTAI separately [6-10, 12-13].

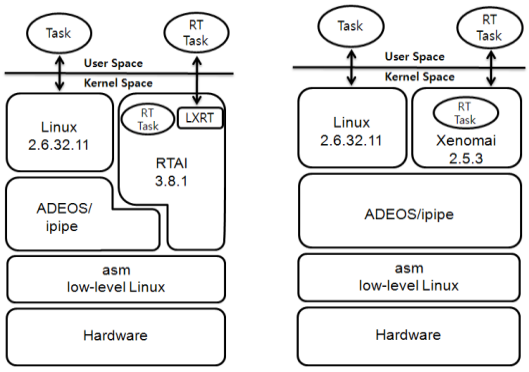
Recently, we studied on the time response of real-time mechanisms in the kernel space for RTAI and Xenomai [14]. Based on the previous results, we extended the time performance analysis of real-time mechanisms to the user space. The researchers want to implement real-time systems in the user space. We analyzed the time performance of real-time mechanisms according to the task period and load. The results are useful since real-time systems should provide deterministic responses within predetermined time constraints.

This paper has five sections. In Section 2, we discuss the implementation procedures of real-time embedded Linux and test environments. Section 3 deals with the time performance of real-time mechanisms in the user and kernel spaces. Section 4 examines how the time performance changes in the kernel/user space when the task periods are changed. The time performance of real-time mechanisms according to load is also analyzed. Finally, we summarize the results of benchmarking the deterministic response to real-time tasks in terms of various real-time mechanisms in real-time embedded Linux, RTAI, and Xenomai.

## 2. Implementation of Real-time Embedded Linux

RTAI and Xenomai are interfaces for real-time tasks rather than real-time operating systems. Therefore, an OS is needed to use them; Linux is most widely used. In RTAI and Xenomai, the Linux OS kernel is treated as an idle task, and it only executes when there are no real-time tasks to run.

Figure 1 shows the architectures and versions of the real-time embedded Linux used in this study. RTAI and Xenomai are conceptually homogeneous, and they both use a general-purpose Linux kernel and real-time API. However, there is a remarkable contrast in the ways they handle interrupts from hardware. To handle interrupts, RTAI and Xenomai both utilize Adaptive Domain Environment for Operating Systems (ADEOS). Hardware interrupts are intercepted by ADEOS and logically forwarded through the pipe structure to other components such as Linux, RTAI, or Xenomai. At this point of transmission, RTAI (Figure 1(a)) can handle interrupts not only by using ADEOS but also by intercepting. In contrast, Xenomai (Figure 2(b)) handles all interrupts using ADEOS. Due to the structural difference, RTAI tends to gain better results than Xenomai in terms of time performance against the interrupt latency [10], while Xenomai, with its simple structure, handles interrupts with higher consistency.




(a) RTAI (b) Xenomai

**Figure 3. Architectures of Real-time Embedded Linux**

### 2.1. Implementation of Real-time Embedded Linux

Through compatibility studies and experiments, we implemented RTAI and Xenomai systems using the latest Linux kernel under the following environmental conditions.

- Ubuntu 10.04 LTS as the host platform
- Intel Core2 Duo CPU E8400 @ 3.00GHz
- Linux kernel 2.6.32.11
- gcc-4.4.3

Prior to patching RTAI, installation of the following package was required to compile the kernel.

- build-essential
- kernel-package
- libncurses5-dev

After installation, the latest version of kernel 2.6.32.11 supported by RTAI-3.8.1 was implemented, and a soft link was configured for compatibility. RTAI was designed to run on a standard Linux kernel, and Ubuntu kernels are widely used as a development platform but can cause unexpected errors. We used the vanilla kernel of version 2.6.32.11. Some kernel configurations were required to use RTAI [15]. Other configurations can be set up depending on the system environment to be developed.

In order to use the same kernel, we adopted Xenomai-2.5.3, which supports Linux kernel 2.6.32. The basic implementation process is the same as that of RTAI. If the version of Xenomai is earlier than 2.5.3, the ADEOS patch is required. For Xenomai, configurations should be disabled so as to not collide with x86 systems [16]. Detailed procedures to implement RTAI and Xenomai are given in [14].

### 2.1. Task Structure of Experiments

Real-time systems transmit or synchronize data between real-time tasks using real-time mechanisms. Therefore, the response time to a real-time mechanism provides an evaluation criterion for expecting the performance of a real-time system. For general-purpose RTOSs, a significant amount of research has focused on the response time [17]. Even though there have been studies on interrupt delays, there have been few reported

results concerning the time performance of real-time mechanisms. Even though previous literatures [8-10] have dealt with the response time of real-time systems, they were all subjected to system responses of kernel. The performance of real-time mechanisms should be analyzed in the user space, where programming can be easily developed with more flexibility.

In the analysis, we programmed real-time tasks in both the kernel and user space. For the programming composition in the user space, Xenomai requires the program to be composed using the same API, while the program for RTAI should be composed using a separate module called LXRT, as shown in Figure 1 [15]. Even though the RTAI API in the kernel space differs from that in the user space, the difference does not affect the entire program structure. In addition to the same structure being shared by the kernel and user spaces, we implemented programs in the kernel space using module structures and excluded the delay time in data processing by confirming the results right after analysis. To prevent scheduling delays, we conducted the analysis by using and running only one module in the kernel. Despite these efforts, there may have been errors due to default processes running on Linux systems. To obtain a more precise comparison of the performances, we repeated the experiment process fifty times. The response times of real-time mechanisms were measured by using the API provided by each RTOS. Figure 2 presents a block diagram of the evaluation tasks.

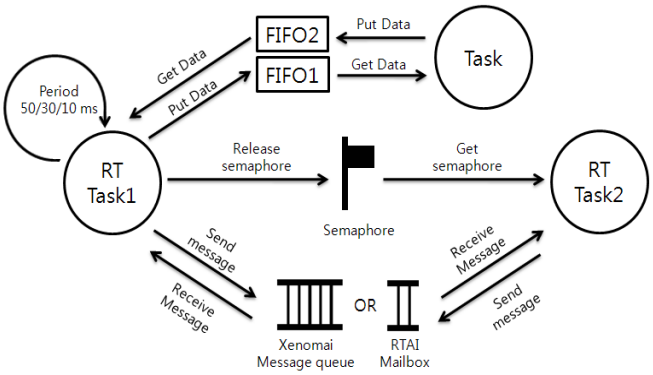



**Figure 4. Block Diagram of Evaluation Tasks for Performance Evaluation**

## 3. Benchmarking Time Response of Real-time Mechanisms

The aim of this study was to analyze the time performance of real-time mechanisms. The performance was evaluated by measuring the API response time provided by RTAI and Xenomai. The analysis focused on the periodicity of the real-time task, which is a fundamental factor for real-time system implementations including semaphores, real-time First-In First-Out (FIFO), and Mailbox and Message queue. The task structures for real-time performance analysis in the kernel and user spaces are similar except for the APIs used. With general-purpose RTOSs, the time tick is usually set to 10 [ms] by default and is implemented in periods of integer multiples of the default value. We measured the response time of the real-time mechanisms in 50 [ms] periods, which are generally used for a typical task. The time response for ITC was measured in nanoseconds.

### 3.1. Periodicity of Periodic Tasks

For performance analysis on the periodicity of the tasks, we used a real-time task with a 50 [ms] period. Tests were performed fifty times to obtain affordable results; the pseudocode for

performance analysis on the periodicity of the tasks is shown in Figure 3. The experimental results are summarized in Figure 4.

```
void Task1()
{
  set period and make periodic;
    …
  while(1)
  {
    start time measurement;
    wait period;
    stop time measurement;
  }
}
```

**Figure 5. Pseudocode for the Periodicity of Periodic Tasks**

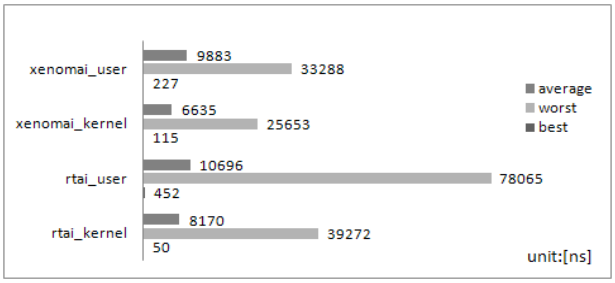



**Figure 6. Performance Test Results for the Periodicity of Periodic Tasks**

The results presented in Figure 4 were obtained for a task with a 50 [ms] period and refer to the error of the task's periodicity. For Xenomai, the mean error (*i.e.*, jitter) was ±9883 [ns] in the user space and ±6635 [ns] in the kernel space. This means that real-time programming in the user space can have more errors in terms of periodicity than that in the kernel space. For RTAI, the mean error was ±10 696 [ns] in the user space and ±8170 [ns] in the kernel space. Errors were more likely to be found in the user space, as was the case for Xenomai. For the total number of errors, Xenomai had fewer errors of periodicity than RTAI, which simply means that tasks were performed with relatively precise periods in Xenomai compared to in RTAI.

### 3.2. Semaphore

A semaphore is a real-time mechanism that abstracts the control of access of shared resources by multiple tasks. A task should wait until a resource unit becomes available. Figure 5 presents the pseudocodes for time performance analysis on a semaphore. We measured the response time for receiving semaphore in Task 2 that was expected to be released it in Task 1. During this process, Task 1 releases the semaphore, and Task 2 waits for the semaphore to execute after context switching. Figure 6 shows the time response of the semaphore in the user and kernel space for RTAI and Xenomai.

```
void Task1()                          void Task2()
{                                     {
  set period and make periodic;         …
    …                                 while(1)
  while(1)                            {
  {                                     get semaphore;
    release semaphore;                  stop          time
    start time measurement;             measurement;
  }                                   }
}                                     }
```

**Figure 7. Pseudocode for the Time Response of Semaphore**

According to the results shown in Figure 6, the average response time for the receiving semaphore between two real-time tasks was 2605 [ns] in the user space and 1558 [ns] in the kernel space. RTAI took 2253 [ns] on average in the kernel space and 8145 [ns] in the user space. Note that the average response time of the semaphore of RTAI in the user space was three times slower than that of Xenomai. Thus, when implementing a real-time system in the user space, better time response characteristics can be achieved by using Xenomai instead of RTAI for the semaphore.

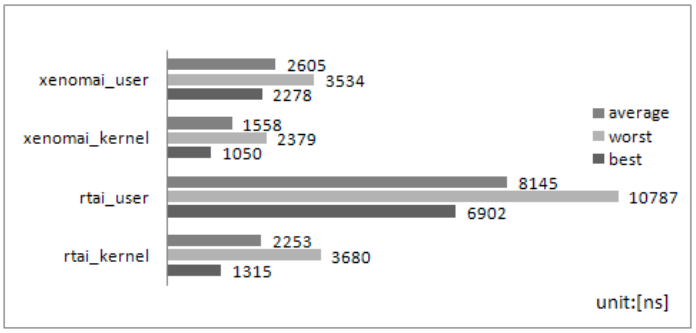



**Figure 8. Performance Test Results for the Time Response of Semaphore**

### 3.3. Real-time FIFO

Real-time FIFO is a method to communicate between tasks. Global variables are a lossy form of communication, so FIFO is usually adopted since writing and reading can be performed at the same time. The advantage of using real-time FIFO is that, even if the reader pauses momentarily and multiple writes to the real-time FIFO occur during that time, data is not lost as long as the reader can catch up and read the elements out of the real-time FIFO before it fills up [18].

Figure 7 shows a pseudocode that represents a structure where data are transferred from Task 1 with a 50 [ms] period to tasks in the user space through real-time FIFO1 and then returned back to Task 1 through real-time FIFO2. We measured the time of the whole process, including sending data and then receiving the returned data. A FIFO cannot simultaneously support reading and writing, so we utilized two FIFOs for bidirectional data transfer. For Xenomai, FIFO was defined as a message pipe.

```
void Task1()
{
  set period and make periodic;
     …
  while(1)
  {
    put data to FIFO1;
    start time measurement;
    get data from FIFO2;
    stop time measurement;
  }
}
```

```
void main()
{
  …
  while(1)
  {
    get data from FIFO1;
    put data to FIFO2;
  }
}
```

**Figure 9. Pseudocode for the Time Response of Real-time FIFO**

The measurements are displayed in Figure 8. For Xenomai, the average response time was 1913 [ns] in the user space and 592 [ns] in the kernel space, while for RTAI, it was 1168 [ns] in the user space and 533 [ns] in the kernel space. In the kernel space, RTAI's FIFO can be expected to have shorter response time characteristics than the message pipe of Xenomai. However, they have similar kernel space performances.

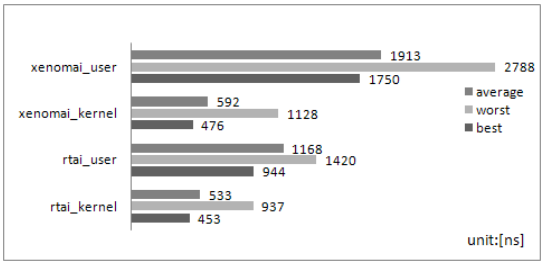



**Figure 10. Performance test Results for the Time Response of Real-time FIFO**

### 3.4. Mailbox and Message Queue

Mailbox and Message queue is a method used in inter-task communication. A queue is used for messaging. RTAI only provides an API for Mailbox while Xenomai has an API for Message queue. We analyzed the performance of RTAI using Mailbox and used Message queue for Xenomai. The analysis was designed similarly to that of Figure 9. We measured the time for Task 1 with a 50 [ms] period to send a message through Mailbox or Message queue to Task 2 and for the message to return back again. Figure 10 shows the measured results.

```
void Task1()
{
  set period and make periodic;
    …
  while(1)
  {
    send message to Task2;
    start time measurement;
    wait for receive message
          from Task2;
    stop time measurement;
  }
}
```

```
void Taks2()
{
  …
  while(1)
  {
    wait for receive
        message from Task1;
    put data to FIFO2;
  }
}
```

**Figure 11. Pseudocode for the Time Response of Mailbox and Message Queue**

As shown in Figure 10, the real-time mechanism provided by Xenomai for Message queue had a better response time than Mailbox of RTAI. In general, Mailbox was a mechanism for task-to-task data transfer, while Message queue was a mechanism used for an N:1 task structure. Thus, it was impossible to directly compare the performances of both mechanisms, but time response analysis can greatly contribute to real-time system implementation.

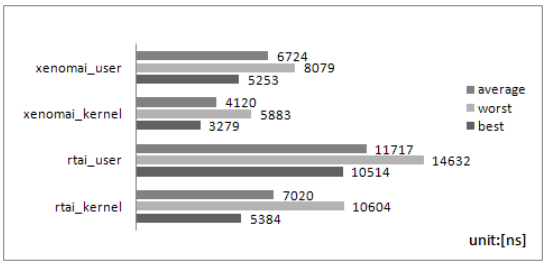



**Figure 12. Performance Test Results for the Time Response of Mailbox and Message Queue**

## 4. Real-time Responses for Various Running Conditions

### 4.1. Benchmarking of Real-time Mechanism with Various Periods

As previously described, real-time systems were implemented with diverse periods. For the general RTOS, the real-time task was implemented with a 10 [ms] period by default with periods increased by n (integer) multiples of the default value. The present study analyzed the difference in performance for different periods of 10, 30, and 50 [ms].

Table 1 displays the average results for the periodic task, semaphore, FIFO, and Mailbox and Message queue according to task periods. According to the results for the periodic task, a shorter task period meant fewer errors. With a 50 [ms] period, Xenomai had fewer errors compared to RTAI in both the kernel and user space. However, with shorter periods, RTAI had fewer errors than Xenomai. Furthermore, at every period, the user space produced more errors than the kernel space.

**Table 1. Experimental Results According to Various Periods**

| RTOS | SPACE | Period [ms] | Periodic Task [ns] | Semaphore [ns] | FIFO [ns] | Mailbox & Message queue [ns] |
|---|---|---|---|---|---|---|
| RTAI | Kernel | 10 | 3837 | 1722 | 443 | 5771 |
| | | 30 | 6146 | 2002 | 481 | 6482 |
| | | 50 | 8170 | 2253 | 533 | 7020 |
| | User | 10 | 6034 | 7207 | 981 | 10550 |
| | | 30 | 9068 | 8037 | 1051 | 11525 |
| | | 50 | 10696 | 8145 | 1168 | 11717 |
| Xenomai | Kernel | 10 | 4840 | 1161 | 461 | 3445 |
| | | 30 | 6541 | 1401 | 522 | 4014 |
| | | 50 | 6635 | 1558 | 592 | 4120 |
| | User | 10 | 6381 | 2420 | 1766 | 6362 |
| | | 30 | 7159 | 2520 | 1876 | 6696 |
| | | 50 | 9883 | 2605 | 1913 | 6724 |

According to the semaphore results, shorter task periods mean shorter durations for both RTAI and Xenomai. The Xenomai semaphore displayed a shorter response time than the RTAI semaphore in every period. RTAI also showed the highest and worst response times in the user space.

With regards to the FIFO results, shorter task periods mean shorter durations for both RTAI and Xenomai. The FIFO mechanism of RTAI showed a shorter duration than the message pipe of Xenomai in every period. In particular, the message pipe of Xenomai had the highest and worst response times in the user space and in every period.

Finally, according to the Mailbox and Message queue results, the message queue mechanism showed a better response time than Mailbox of RTAI in every period. Similar to the previous results, a shorter task period meant a shorter response time for both RTAI and Xenomai.

### 4.2. Benchmarking of Real-time Mechanism with Load

The difference in the performance of real-time mechanisms with changes in load was investigated. As the object of analysis, the task period was 50 [ms]; the load task period was set to 0.1 [ms]. The priority of the load task was set to be lower than the task periods as an object of analysis. Ten load tasks were created. The load task repeatedly carried out arithmetic operations during the period. Table 2 shows the results. RTAI and Xenomai shows no difference in the performance of the real-time mechanism with changes in the load task.

**Table 2. Experimental Results According to Load**

| RTOS | SPACE | Semaphore [ns] | | FIFO [ns] | | Mailbox & Message queue [ns] | |
|---|---|---|---|---|---|---|---|
| | | No Load | Load | No Load | Load | No Load | Load |
| RTAI | Kernel | 2253 | 2277 | 533 | 562 | 7020 | 6989 |
| | User | 8145 | 9043 | 1168 | 2057 | 11717 | 13329 |
| Xenomai | Kernel | 1558 | 1498 | 592 | 566 | 4120 | 4144 |
| | User | 2605 | 2521 | 1913 | 1921 | 6724 | 6675 |

## 5. Conclusions

In this study, we investigated the performance of real-time mechanisms in both the user and kernel space. Moreover, we analyzed the dependence of the time performance of real-time mechanisms on the task period and load. Although several researchers have reported that the performance of real-time embedded Linux is not worse than that of general-use RTOS, no research has been carried out on the response time characteristics of real-time mechanisms, which is a prerequisite for satisfying deterministic real-time system implementation. Generally, RTOS provides the worst performance time on every API, which allows users to understand how the system works in its worst condition. However, research on RTAI and Xenomai was insufficient to help users, even though the systems have increasingly improved their performance through open-source projects. Previous studies on the performance of real-time mechanisms focused only on the controllability of the system, delays in interrupt treatment, or the kernel space [10, 12-14].

The task periodicity showed relatively consistent performance in the kernel space, while Xenomai had better jitter in the user space. For real-time mechanisms, the response times

were faster in the kernel space than in the user space with narrow variations, which implies a deterministic response of real-time systems. Furthermore, in both RTAI and Xenomai, the response time became faster when the task period was shorter. Real-time FIFO of RTAI had faster response characteristics than Xenomai had, while the semaphore showed a superior response in Xenomai than in RTAI in terms of the time response characteristics of the message transfer mechanism. The real-time mechanism showed no difference in performance with changes in the load task.

Therefore, time performance analysis of real-time mechanisms is an important guideline to ensure the deterministic response of real-time tasks when designing real-time systems. The present study investigated the time performance in the user space and was not limited to the kernel space. The results of this paper provide highly useful fundamental principles or information on the deterministic response to the development of real-time systems using real-time embedded Linux. We will continue to study the implementation of optimum real-time mechanisms by studying the source of real-time mechanisms in embedded Linux to produce a promising solution for the best deterministic response of real-time systems when using real-time embedded Linux.

## Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2012-006057)

## References

[1] T. Bird, “Comparing two approaches to real-time Linux”, http://www.linuxdevice.com, **(2002)**.
[2] I. Ripoll, “RTLinux versus RTAI”, http://www.linuxfordevices.com, **(2002)**.
[3] K. Dankwardt, “Comparing real-time Linux alternatives”, http://www.linuxdevice.com, **(2002)**.
[4] W. S. Liu, “Real-Time System”, Prentice Hall **(2000)**.
[5] D. Aboutt, “Linux for Embedded and Real-time Applications”, Elsevier **(2006)**.
[6] M. D. Marieska, A. I. Kistijantoro and M. Subair, “Analysis and Benchmarking Performance of Real Time Patch Linux and Xenomai in Serving a Real Time Application”, Proc. of International Conf. on Electrical Engineering and Informatics, **(2011)**, pp. 1-6.
[7] P. Kadionik, B. L. Gal, H. Levi and A. B. Atitallah, “Performances analysis and evaluation of Xenomai with a H.264/AVC decoder”, Proc. of Int’l Conf. on Microelectronics, **(2011)**,pp. 1-4.
[8] G. Zhang, L. Chen and A. Yao, “Study and Comparison of the RTHAL-based and ADEOS-based RTAI Real-time Solutions for Linux”, Proc. of International MultiSymposium on Computer and Computational Sciences, **(2006)**, pp. 771-775.
[9] M. Liu, D. Liu, Y. Wang, M. Wang and Z. Shao, “On Improving Real-Time Interrupt Latencies of Hybrid Operating Systems with Two-Level Hardware Interrupts”, IEEE Trans. on Computers, vol. 60, no. 7, **(2011)**, pp. 978-991.
[10] A. Barbalace, A. Lunchetta, G. Manduchi, M. Moro, A. Soppelsa and C. Taliercio, “Performance Comparison of VxWorks, Linux, RTAI and XENOMAI in a Hard Real-time Application”, Proc. of Real-Time Conference 2007 15th IEEE-NPSS, **(2007)**, pp. 1-5.
[11] B. W. Choi, “A Review and Outlook of Robotic Software Framework”, J. of Korean Robotic Society, vol. 5, no. 2, **(2010)**, pp. 169-176.
[12] B. W. Choi, D. G. Shin, J. H. Park, S. Y. Yi and S. Gerald, “Real-time control architecture using Xenomai for intelligent service robot in USN environment”, J. of Intelligent Service Robotics, vol. 2, no. 2, **(2009)**, pp. 139-151.

[13] M. Franke, "A Quantitative Comparison of Realtime Linux Solutions", Chemnitz University of Technology, **(2007)**.
[14] J. H. Koh and B. W. Choi, "Performance Evaluation of Real-time Mechanisms for Real-time Embedded Linux", J. of Institute of Control, Robotics and Systems (in Korean), vol. 18, no. 4, **(2012)**, pp. 337-342.
[15] RTAI, http://www.rtai.org.
[16] Xenomai, http://www.xenomai.org.
[17] J. J. Labrosse, "uC/OS-II The Real-Time Kernel", Micrium **(2009)**.
[18] Real-Time FIFO for Deterministic Data Transfer Between Vis, http://www.ni.com/white-paper/3934/en, National Instrument **(2011)**.