International Journal of Internet, Broadcasting and Communication
Vol.13 No.1

ISSN : 2288-4920(Print) 2288-4939(Online)

A Probabilistic Analysis for Periodicity of Real-time Tasks

Raimarius Delgado, Byoung Wook Choi

To cite this article : Raimarius Delgado, Byoung Wook Choi (2021) A Probabilistic Analysis for Periodicity of Real-time Tasks, International Journal of Internet, Broadcasting and Communication, 13:1, 134-142

International Journal of Internet, Broadcasting and Communication Vol.13 No.1 134-142 (2021)
http://dx.doi.org/10.7236/IJIBC.2021.13.1.134

IJIBC 21-1-16

A Probabilistic Analysis for Periodicity of Real-time Tasks

[1]Raimarius Delgado, [2]Byoung Wook Choi

[1]*Ph.D. Candidate, Department of Electrical and Information Engineering, Seoul National University of Science and Technology, South Korea*
[2]*Professor, Department of Electrical and Information Engineering, Seoul National University of Science and Technology, South Korea*
[1]*raim223@seoultech.ac.kr,* [2]*bwchoi@seoultech.ac.kr*

Abstract

This paper proposes a probabilistic method in analyzing timing measurements to determine the periodicity of real-time tasks. The proposed method fills a gap in existing techniques, which either concentrate on the estimation of worst-case execution times, or do not consider the stochastic behavior of the real-time scheduler. Our method is based on the Z-test statistical analysis which calculates the probability of the measured period to fall within a user-defined standard deviation limit. The distribution of the measured period should satisfy two conditions: its center (statistical mean) should be equal to the scheduled period of the real-time task, and that it should be symmetrical with most of the samples focused on the center. To ensure that these requirements are met, a data adjustment process, which omits any outliers in the expense of accuracy, is presented. Then, the Z-score of the distribution according to the user-defined deviation limit provides a probability which determines the periodicity of the real-time task. Experiments are conducted to analyze the timing measurements of real-time tasks based on real-time Linux extensions of Xenomai and RT-Preempt. The results indicate that the proposed method is able to provide easier interpretation of the periodicity of real-time tasks which are valuable especially in comparing the performance of various real-time systems.

Keywords: *Periodic real-time systems, measurement-based analysis, probabilistic analysis, Z-Test, real-time*

1. Introduction

Real-time systems are essential to execute jobs which are safety-critical especially in industrial control systems, smart-grids, and robotics [1-3]. Such systems are expected to share their working environment with human beings. Missing a deadline may result in catastrophic results, whether severe malfunction of the system, or may cause human accidents. It is therefore essential that these applications are bounded with stringent real-time requirements. This means that aside from the logical correctness, these applications should adhere to strict temporal deadlines. Development of real-time applications usually utilizes real-time operating systems (RTOS) owing to their multi-tasking environment and priority-based scheduler. Since the semantics and behavior of the scheduler varies for each RTOS, the same real-time applications may produce varying performance as well.

To ensure that real-time applications can perform the required task functionalities without violating temporal deadlines, real-time performance evaluation should be conducted. In this context, schedulability of the real-time tasks should be guaranteed. Real-time tasks are schedulable if their worst-case response times (WCRT) do not violate the respective temporal deadlines [4]. The response time is defined as the interval from the start

Manuscript Received: December. 15, 2020 / Revised: December. 19, 2020 / Accepted: December. 24, 2020
Corresponding Author: bwchoi@seoultech.ac.kr
Tel: +82-2-970-6412, Fax: +82-2-978-2754
Professor, Department of Electrical and Information Engineering, Seoul National University of Science and Technology, South Korea

of the task, until it finishes executing the expected task functionality.

Existing performance evaluation methods focus on estimating the worst-case execution time (WCET) to calculate for the WCRT. The WCET bounds the execution time of the real-time task in all of the task iteration. For example, Cucu-Grosjean et al., [5] presented a combination of actual measurements and probabilistic analysis using extreme value theory (EVT) to estimate for the WCET. Santinelli et al. [6] and Guet et al. [7] have presented results which showed that EVT can be sustainably and reliably applied in practical cases.

However, these approaches did not consider the stochastic behavior of the scheduler. This behavior results to the jitter. It is the difference between the expected release point and the actual starting point of the real-time task. There are numerous sources for the jitter and searching for the exact one is extremely difficult. In practice, the real-time system is benchmarked for the variation of the jitter to determine the lower-bound for the configurable period of real-time tasks. In a Linux system, for example, the most common approach to measure the scheduling latency is through the benchmarking tool called cyclictest [8]. To this end, analysis of the jitter should be performed for each real-time application to determine the periodicity of real-time tasks.

This paper propose a probabilistic method to analyze the periodicity of real-time tasks. In an ideal scenario, measured period real-time tasks should be equal to the scheduled period in all of its iterations. The inevitable presence of the jitter, however, limits this behavior. Hence, the measured period should result to a distribution with the center equal to the scheduled period, with minimal deviation due to jitter. In the proposed method, the timing measurements are analyzed using the Z-test statistical analysis in which the probability of the measured period to be within a given standard deviation limit is calculated. Since actual timing measurements may produce results that are not approximately normal, a data adjustment process is also presented to ensure that the distribution is symmetric and center-focused to perform the statistical analysis. To validate the feasibility of the method, it has been applied to analyze the periodic behavior of an experimental taskset implemented on the real-time Linux extensions of Xenomai and RT-Preempt [9]. The results indicate that the proposed method can provide easier interpretation of the periodicity of real-time tasks which are valuable especially in comparing the performance of various real-time systems.

2. Probabilistic analysis of the periodicity of real-time tasks

In this paper, we focus on analyzing the periodicity of periodic real-time tasks. The distribution of the measured period is analyzed for its ability to execute cyclically with minimal deviations from the given period. In an ideal environment, the actual period should be equal to the scheduled period of the real-time task. In practical applications, however, there is a difference between the actual and expected release points due to the stochastic behavior of the real-time scheduler. This difference is called the jitter. The jitter should be kept as small as possible to ensure that the real-time task can meet its deadline and show a deterministic behavior.

This means that the measured period should be an approximately normal distribution with its center approximately equal to the expected period, P_i, and the standard deviation of the distribution should be kept minimal. With these assumptions, the measured period should satisfy two conditions to be considered deterministic: The center is equal to the scheduled period, and the distribution should be symmetrical. From these conditions, the distribution of the measured period, p_i, is analyzed to determine periodicity of the real-time task. The periodicity is calculated as the probability of the measured period falling within the interval of a user-defined three-sigma standard deviation limit, $\hat{\sigma}$, or:

$$\text{periodicity} = \text{Prob}(P_i - 3\hat{\sigma} \le p_i \le P_i + 3\hat{\sigma}) \quad (1)$$

This means that the periodicity will result to a value between 0 and 1. The process starts by calculating the length of the measured period, which is denoted by X. $\bar{X}$ denotes the number of omitted samples when an adjustment occurs, and accuracy refers to the trade-off factor which is the ratio of $\bar{X}$ over X. To ensure that the center of the distribution is approximately equal to P_i, the statistical mean (μ) is calculated. For flexibility, a user-defined tolerance level is added to allow a margin of error. To determine if the distribution is symmetric, the skewness, μ_3 , should be within twice of its standard error, or $-2\mu_{3\_error} \le \mu_3 \le 2\mu_{3\_error}$.

```
Procedure PeriodicityAnalysis(P,p,sigma_limit,tolerance)
  X = length(p); % original sample size
  X_bar = 0 ;    % size of deleted samples
  accuracy = 1; % initialize at 100%
  periodicity = 0;
  while 1
    if X_bar >= X
      periodicity = 0;
      break;
    endif
    accuracy = 1 - (X_bar / X);
    mu = mean(p); % calculate center
    if (P-tolerance <= mu && mu <= P+tolerance )
      mu_3 = get_skew(D); % skewnesss
      mu_3_error = sqrt((6*X*(X-1))/((X-2)*(X+1)*(X+3)));
      if (-2 * mu_3_error <= mu_3 <= 2* mu_3_error)
        sigma = std(p);
        if sigma <= sigma_limit
          periodicity = accuracy;
          break;
        else
          periodicity = get_score(mu, sigma, sigma_limit*3) * accuracy;
      else % data adjustment procedure
        p = adjust_data(p); X_bar = length(p);
      endif
    else % data adjustment procedure
      p = adjust_data(p); X_bar = length(p);
    endif
  endwhile
endprocedure
```

Figure 1. Pseudocode of the periodicity analysis

Figure 1 shows the graphical representation of the periodicity analysis. The skewness and its standard error are calculated as [10]:

$$\mu_3 = \frac{\sum_{x=1}^{X}(p_i^{(x)} - \mu)^3}{\sigma^3 (X-1)}$$
$$\mu_{3\_error} = \sqrt{\frac{6X(X-1)}{(X-2)(X+2)(X+3)}} \quad (2)$$

The skewness measures the asymmetry of the distribution with respect to the center, or in this case, the statistical mean, μ. If the skewness is positive, it means that the distribution is asymmetric and most of the samples are leaning to the left side of the center. On the other hand, negative skew denotes that the right side from the center is denser. As mentioned above, the skewness should be within the allowable range of its standard error to ensure that the distribution is symmetrical.

If so, the standard deviation of the distribution, σ , is verified whether it is lesser than the user-defined limit, $\hat{\sigma}$. In this case, the procedure will end, and the resulting periodicity is equal to the accuracy factor. The accuracy factor is the trade-off of data adjustment due to the outliers that can affect symmetry and central tolerance.

```
Procedure adjust_data(p)
  p_mean = mean(p);
  p_minus_mean = abs(p_mean - p);
  % find maximum value and its position
  [p_max, p_maxpos] = max(p_minus_mean);
  p(p_maxpos) = [] % delete the maximum value from p
  return p;
endprocedure
```

Figure 2. Pseudocode of the data adjustment process

Figure 2 shows the pseudocode of the data adjustment process. If the distribution of the measured period does not satisfy the central tolerance and symmetry requirements, the dataset is adjusted by omitting any outlier. An outlier is defined as the sample farthest from the center of the distribution. In other words, the outlier is determined as the maximum value of the difference between all of the samples and the statistical mean. The outlier is omitted from the original distribution and this procedure returns a data set without the outlier. As mentioned above, adjustment of the data comes with a trade-off in accuracy. The accuracy factor is calculated as the ratio of the number of the omitted samples over the actual data size, or $1-\bar{X}/X$. This process ensures that the distribution of the measured period is symmetrical with the center, and the statistical mean is approximately equal to scheduled period with the user-defined central tolerance. If all of the samples are omitted, or $\bar{X} \geq X$, the process will end, and the periodicity is equal to zero.

Otherwise, the distribution can therefore be defined as approximately normal. Thus, statistical analysis for normal distributions such as the Z-test [11] can be performed for the distribution of the measured period. The periodicity is calculated as the probability of p_i within the interval of the user-defined standard deviation limit as denoted in (1). This is graphically shown in Figure 3.

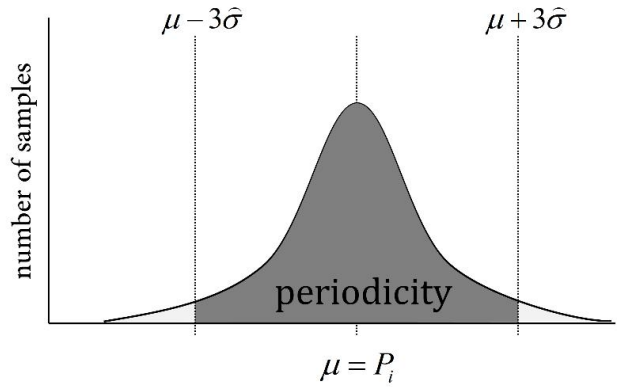

Figure 3. Graphical representation of the periodicity analysis

In this figure, the periodicity is defined as the area within three standard deviation limits ($3\hat{\sigma}$). The three-sigma limit is a good measure to determine the periodicity of the real-time task. In case that the calculated periodicity is 1, this means that 99.7% of the measured period are within the user-defined limit. Note that samples beyond this limit can be neglected since they are inconsistent with the tested model. This is in accordance with a presentation in ROSCON 2019 entitled Safety in Time [12], there is a difficulty to ensure real-time characteristics in modern embedded systems, and they should be given in probabilities. This notion has been proven in our previous study in [9], which states that multicore deployments and CPU features related to power saving greatly affects the real-time performance of modern embedded systems.

```
Procedure get_score(mu, sigma, sigma_limit)
  z_plus  = ((mu + sigma_limit)- mu) / sigma;
  z_minus = ((mu - sigma_limit)- mu) / sigma;
  Z_plus  = normcdf(z_plus);
  Z_minus = normcdf(z_minus);
  periodicity = Z_plus - Z_minus;
  return periodicity;
endprocedure
```

Figure 4. Pseudocode of periodicity calculation

Figure 4 shows the pseudocode to calculate the periodicity shown in Figure 3. This assumes that the measured period results to an approximately normal distribution with the center equal to the scheduled period, and the calculated standard deviation is greater than the standard deviation limit. To determine the probability in (1), the corresponding standard score, or Z-score is calculated [11]:

$$z^{+} = \frac{(\mu + \hat{\sigma}) - \mu}{\sigma} = \frac{\hat{\sigma}}{\sigma}$$
$$z^{-} = \frac{(\mu - \hat{\sigma}) - \mu}{\sigma} = -\frac{\hat{\sigma}}{\sigma} \tag{3}$$

These equations measure the distance, or the position, of the standard deviation limit from the statistical mean depending on the calculated standard deviation. To calculate for the periodicity, these measurements should be associated to its respective probability and (1) becomes:

$$\text{periodicity} = (Z(z^{+}) - Z(z^{-})) \cdot \text{accuracy} \tag{4}$$

Herein, the $Z(.)$ function converts the Z-scores from (3) into standard normal probabilities. These values can be found in a Z-score table [13]. In Matlab, these are obtained using the normcdf() function. Note that the resulting values increase with higher Z-scores. This means that the calculated periodicity has an inverse relationship with the measured standard deviation, σ . The periodicity increases as σ approaches $\hat{\sigma}$. The results are multiplied to the accuracy factor, which is 1 if no data adjustment has been made. To this end, the periodicity interprets the distribution of the measured period, and ensures that it satisfies the standard deviation limits, which then determines the deterministic behavior of real-time tasks.

3. Experimental Results

This section focuses on validating the proposed method through implementation on an actual real-time system. In this work, an Intel-based embedded system called MIO-5272 manufactured by Advantech [9], is utilized to conduct all experiments. It is implemented with Linux 4.14.134 and two extensions of real-time Linux, Xenomai 3.0.9 and RT-Preempt 4.14.134-rt127. An experimental taskset with two real-time tasks is designed to run simulated load by busy waiting the CPU according to the configured execution time. Task parameters for each real-time task are shown in Table 1.

Table 1. Task parameters for the experimental taskset (in milliseconds)

Task Name	Period	Deadline	Execution Time	Priority
τ_1	10	10	3	99
τ_2	20	20	5	89

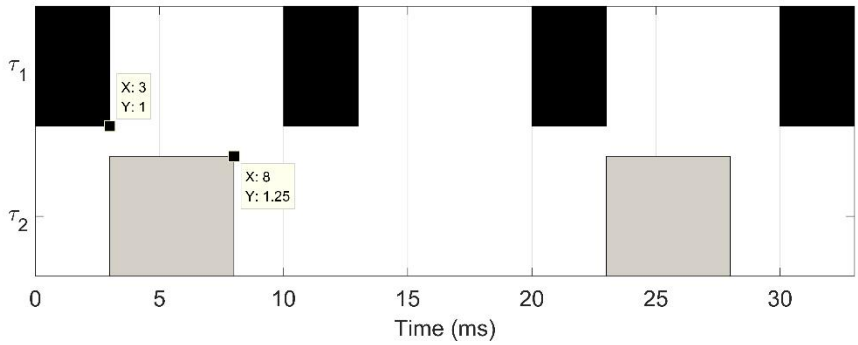

Figure 5. Execution timeline of the experimental taskset

Herein, we follow rate-monotonic scheduling [4], which states that the highest priority task should have the fastest period, and all of the deadlines are equal to the period. This is depicted by τ_1 having the period of 10 milliseconds with the highest priority of 99. Note that in both Xenomai and RT-Preempt, the highest priority for a real-time task is 99, while 1 is the lowest. The lower priority task, τ_2 is configured with to run periodically every 20 milliseconds and lower priority of 89. Both of the tasks are expected to busy wait the CPU for 3 milliseconds and 5 milliseconds, respectively. Based on rate-monotonic analysis [4], both of the real-time tasks are schedulable, with the calculated worst-case response times (WCRT) are 3 milliseconds and 8 milliseconds, respectively. Note that for τ_1 , the WCRT is equal to the configured execution time since it has the highest priority. It should execute without any blocking or interference from lower real-time tasks. τ_2, on the other hand, may only run after τ_1 has finished its execution. The execution timeline of the taskset is shown in Fig. 5.

The experiments were performed on each real-time Linux extension for 10 minutes to verify whether they can show schedulability in comparison to the values above. During the experiment, the real-time system is kept isolated to avoid any unwanted interruptions that could affect the performance of the entire system. For this reason, all the measured values are stored in a buffer for offline processing and analysis. The results of the timing analysis are shown in Table 2 and Table 3 for Xenomai and RT-Preempt, respectively. These are composed of the statistical mean (avg.), maximum (max.), minimum (min.), and standard deviation (st.d.). From the timing measurements, the proposed periodicity analysis method in the previous section were conducted and the results are also shown in the table denoted as periodicity. The accuracy refers to the number of omitted samples due to the violation of the symmetrical and central tolerance conditions. Note that the accuracy is reflected on the final value of periodicity as denoted in (4).

Table 2. Timing measurements and periodicity for Xenomai (in milliseconds)

Task	τ_1(10 ms, 99)		τ_2 (20 ms, 89)	
Metric	Period	Response Time	Period	Response Time
avg.	10.000000	3.004998	19.999968	5.002689
max.	10.071585	3.076599	20.061001	5.047895
min.	9.929621	3.003334	17.093339	5.001750
st. d.	0.002581	0.002545	0.001392	0.000868
periodicity	0.7806, accuracy=100%		0.9291, accuracy=99.8%	

For the experimental taskset, the standard deviation limit is configured as 0.001 milliseconds and the tolerance is kept to a minimum value of 0.0001 milliseconds. As shown in the results for Xenomai in Table 2, the average of the measured period for both real-time tasks are equal to their respective scheduled periods of 10 and 20 milliseconds, respectively.

All of the tasks are schedulable with the measured maximum response times that are lesser than the period and equal to the calculated WCRT. For the periodicity using (3) and (4), τ_1, has a value of 0.7806 with an accuracy of 100%. This means that the distribution of the measured period did not require any adjustments. However, the low score is accountable for the high standard deviation of 0.002581, which is greater than the standard deviation limit. This means that only 78% percent of the measured period are within the range of the user-defined standard deviation limit. For τ_2, however, approximately 120 samples were omitted from the distribution to satisfy the symmetrical and central tolerance requirements. This results to an accuracy of 99.8%. The total periodicity score in this case is 0.9291. Notice that the lower priority task has shown better performance than the higher priority task. This is a peculiar finding which can be attributed to this particular version of Linux and Xenomai for the MIO-5272 hardware.

Table 3. Timing measurements and periodicity for RT-Preempt (in milliseconds)

Task	τ_1(10 ms, 99)		τ_2 (20 ms, 89)	
Metric	Period	Response Time	Period	Response Time
avg.	10.000000	3.004008	20.000015	5.004178
max.	10.063427	3.065291	21.352405	5.060112
min.	9.938631	3.003075	19.937828	5.003306
st. d.	0.001315	0.000940	0.001285	0.002474
periodicity	0.9516, accuracy=100%		0.9958, accuracy=99.6%	

Table 3 shows the results for RT-Preempt. The same standard deviation limit and central tolerance is implemented to compare the performance of both real-time Linux extension. As in Xenomai, the average of the measured period for all real-time tasks are equal to the respective scheduled period. There are also no issues regarding schedulability with all maximum response times lesser than the period. In contrast to Xenomai, the standard deviation of the real-time tasks in RT-Preempt show lesser difference with the user-defined standard deviation limit.

Hence, the periodicity of the τ_1 is 0.9516 which is almost 20% higher than the results from the same task in Xenomai. The same trend has been observed with τ_2. The calculated periodicity is 0.9958 with the accuracy of 99.6%. Even in RT-Preempt, the lower priority task has shown better performance in comparison to the higher priority one. The results have shown that the periodicity of τ_2 is higher, but with lower accuracy. This means that approximately 240 samples were omitted to generate an approximately normal distribution. To visualize the effects of the data adjustment process, the histograms of the original and adjusted data are graphically shown in Figure 6.

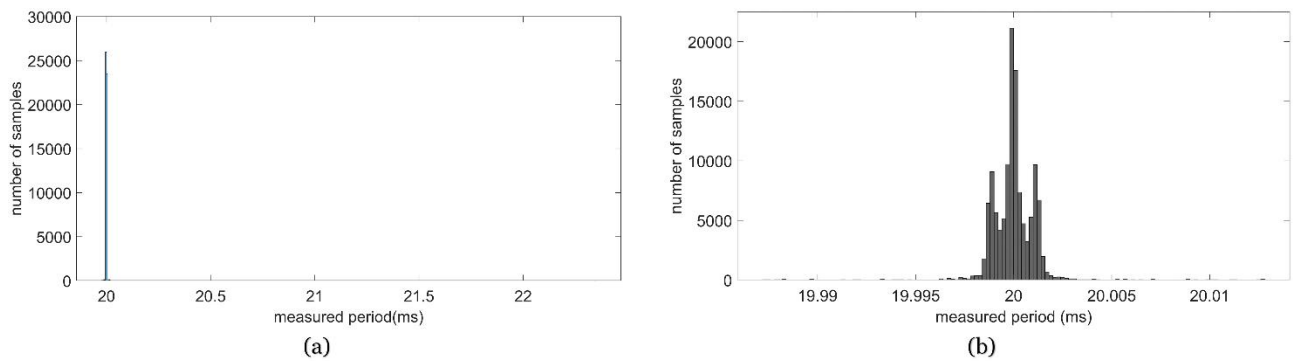

Figure 6. Example of data adjustment of the measured period by omission of outliers; (a) original data (b) adjusted data

As shown in Fig. 6(a), the original timing measurements has shown a concentrated distribution at the scheduled period of 20 ms. However, there are some outliers located far from the center at the right side of the distribution. Using the data adjustment process in Figure 4, the distribution clearly becomes symmetrical as shown in Fig. 6(b). From the results in Table 2 and Table 3, it can be concluded that for this particular version of the Linux kernel, Xenomai, and RT-Preempt, RT-Preempt has shown better periodicity on the Intel-based Advantech MIO-5272.

The results have shown that using the proposed periodicity analysis method, interpretation of the timing measurements has become easier, which can be described using a scale of 0 and 1. This is very convenient especially when comparing various real-time systems such as the conducted experiment for Xenomai and RT-Preempt.

4. Conclusion

This paper proposes a probabilistic method in analyzing timing measurements to determine the periodicity of real-time tasks. It has been observed that existing techniques are more focused either on the estimation of worst-case execution times, or do not consider the stochastic behavior of the real-time scheduler. The proposed method filled this gap by presenting a probabilistic method based on the Z-test statistical analysis.

Herein, actual measurements from the real-time tasks are acquired and analyzed to calculate for the probability within a given standard deviation limit. There are two contingencies for this calculation: 1) The center (statistical mean) of the distribution should be equal to the scheduled period of the real-time task, and 2) it should be symmetrical with most of the samples focused on the center. The symmetry of the distribution is determined by analyzing its skewness, which should be between twice of its standard error. In case these conditions are not met, a data adjustment process has also been presented. This omits any outliers that may affect the central tolerance and symmetry of the distribution, in the expense of accuracy. The periodicity is then calculated depending on the Z-score of the distribution with accordance to the user-defined standard deviation limit.

Experiments were conducted on actual real-time systems based on the real-time Linux extensions of Xenomai and RT-Preempt. An experimental taskset of two real-time tasks with harmonic periods were generated to busy-wait the CPU according to the configured execution time. The actual timing measurements were analyzed using the proposed method, and it has been observed that RT-Preempt has shown better periodic results than Xenomai. As we are more focused on the analysis method, rather than the comparative results from both real-time Linux extensions, it can therefore be concluded that the proposed method can simplify the interpretation of the timing measurements. This is very valuable especially when comparing various real-time systems such as the conducted experiment for Xenomai and RT-Preempt.

There are various directions for future works. Extensive experiments with multiple tasks and multiple platforms are being considered. Also, aside from the periodicity, analysis methods for the response times should also be conducted. Moreover, the analysis method should be extended to multi-tasking environments [14], where each task should be weighed on their contribution to the overall performance of the real-time system.

Acknowledgement

This work has been financially supported by the SeoulTech(Seoul National University of Science and Technology).

References

[1] R. Delgado, and B.W. Choi, “Safe and Policy Oriented Secure Android-Based Industrial Embedded Control System,” Applied Sciences, Vol. 10, No. 8, p. 2796, January 2020. DOI: https://doi.org/10.3390/app10082796

[2] W.Y. Lee, and Y.-S. Choi, "Energy-efficient Scheduling of Periodic Real-time Tasks on Heterogeneous Grid Computing Systems," International Journal of Internet, Broadcasting, and Communication, Vol. 9, No. 2, pp. 78-86, May 2017. DOI: https://doi.org/10.7236/IJIBC.2017.9.2.78

[3] J. Lee, and Y.-S. Choi, "Design and Development of a Monitoring System based on Smart Device for Service Robot Applications," International Journal of Internet, Broadcasting, and Communication, Vol. 10, No. 3, pp. 35-41, August 2018. DOI: https://doi.org/10.7236/IJIBC.2018.10.3.35

[4] L.C. Liu, and J.W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," Journal of the ACM, Vol. 20, No. 1, pp. 46-61, January 1973. DOI: https://doi.org/10.1145/321738.321743

[5] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Verdanega, L. Kosmidis, J. Abella, E. Mezzeti, E. Quinones, and F.J. Cazorla, "Measurement-Based Probabilistic Timing Analysis for Multi-path Programs," in Proc. 24th Euromicro Conference on Real-Time Systems, pp. 91-101, July 11-13, 2012. DOI: https://doi.org/ 10.1109/ECRTS.2012.31

[6] L. Santinelli, J. Morio, G. Dufour, and D. Jacquemart, "On the Sustainability of the Extreme Value Theory for WCET Estimation," in Proc. 14th International Workshop on Worst-Case Execution Time Analysis, pp. 21-30, July 8, 2014. DOI: https://doi.org/10.4230/OASIcs.WCET.2014.21

[7] S. Milutinovic, J. Abella, and F.J. Cazorla, "On the assessment of probabilistic WCET estimates reliability for arbitrary programs," Journal of Embedded Systems, Vol. 28, No. 2017, pp. 1-16, April 2017. DOI: https://doi.org/10.1186/s13639-017-0076-8

[8] D. B. de Oliveira, D. Casini, R. S. de Oliveira, and T. Cucinotta, "Demystifying the real-time linux scheduling latency," in Proc. 32nd Euromicro Conference on Real-Time Systems, pp. 1-23, July 7-10,2020. DOI: 10.4230/LIPIcs.ECRTS.2020.9

[9] R. Delgado, and B.W. Choi, "New Insights Into the Real-Time Performance of a Multicore Processor," IEEE Access, Vol. 8, 186199-186211, October 2020. DOI: https://doi.org/10.1109/ACCESS.2020.3029858

[10] A.J. Bishara, and J.B. Hittner, "Confidence intervals for correlations when data are not normal," Behavior research methods, Vol. 49, No. 1, pp. 294-309, February 2017. DOI: https://doi.org/10.3758/s13428-016-0702-8

[11] A.E. Curtis, T.A. Smith, B.A. Ziganshin, and J.A. Elefteriades, "The mystery of the Z-score," AORTA Journal, Vol. 4, No. 4, pp. 124-130, August 2016. DOI: https://doi.org/10.12945/j.aorta.2016.16.014

[12] Z-score table, http://www.z-table.com/

[13] G. Biggs, Safety in Time: Real-Time and Safety-Critical Software Development, https://www.apex.ai/roscon2019

[14] D-.H. Gong, and S.-J. Shin, "Comparative Analysis between Super Loop and FreeRTOS Methods for Arduino Multitasking," The Journal of the Institute of Internet, Broadcasting and Communication(JIIBC), Vol. 18, No. 6, pp. 133-137, December 2018. DOI: https://doi.org/10.7236/JIIBC.2018.18.6.133