Relative Deadline Analysis in Multitasking RTS using RM & EDF Scheduling

Sandeep S Chapalkar, K. Karibasappa

Abstract: ***In embedded systems the time required for any process to complete its execution in multitasking environment is an important factor to understand the performance of Real Time System (RTS) and its ability fulfill the deadline requirement of each process under different process load conditions. Even though some non-critical systems provide flexibility over deadlines, the hard real time systems are to be designed to meet the deadline requirement of all processes under peak process load condition. The number of processes available in scheduling queue may vary with time, the dynamic load on processing unit also changes proportionately which in turn affects the relative deadlines of each process. The scheduling policies considered are widely used scheduling policies like Rate Monotonic (RM) and Earliest Deadline First (EDF) to analyze and understand the impact on relative deadline with respect to number of scheduled processes. The real time execution timings of each process is observed on Raspberry Pi 3b+ processing unit operating at standard frequency of 700 MHz in multitasking mode of operation. The results obtained will decisively conclude the suitable scheduling policy for a set of processes under different process load conditions.***

Keywords: Embedded, Dynamic load, load estimation, Multitasking, Optimization, Rate monotonic, Earliest Deadline First, Real time systems, Absolute deadline, Relative deadline

I. INTRODUCTION

Embedded systems have been an integral part of life from few decades and is expected to play more prominent role in future days. The ever changing requirement and needs of end users leads more complex systems and in turn is going to increase in number of processes to be executed. And new applications demands systems capable of handling more complex operations. Therefore it's important to identify the most efficient scheduling method for implementation of efficient product. As the processes associated with the application may be event triggered or user initiated, the number of processes scheduled in the system may vary continuously with time. This change in the number of processes significantly affects the relative deadlines of each process and it's important to design systems adopt the right scheduling methods to deliver efficient performance.

Manuscript received on May 17, 2021.
Revised Manuscript received on May 24, 2021.
Manuscript published on May 30, 2021.
* Correspondence Author
Sandeep S Chapalkar*, Assistant Professor, Department of Electronics and Communication Engineering, Dayananda Sagar College of Engineering, Bengaluru (Karnataka), India, E-mail: sandeepsc.ece@gmail.com
K. Karibasappa, Professor, Department of Electronics and Communication Engineering, Dayananda Sagar Academy of Technology and Management, Bengaluru (Karnataka), India, E-mail: karibasappably@gmail.com

The objective of this work is to understand the effect of scheduling policies on relative deadlines of different processes in the system. The relative deadlines needs to be recalculated with every new process in the system. In multitasking systems, the processes can be triggered by different methods such as events triggered, human interfaces or periodic. Therefore, the number of active processes in the system keeps varying with respect to time and condition.

In some systems the processes which will be executing continuously or periodically are priory known. Depending on execution time of these processes developer can estimate the static load on the processing unit and select appropriate processing unit to meet the deadline requirements. However, in some systems the processes may be triggered during operational phase depending on event or human interaction, under such circumstances the relative deadlines of the processes need to be recalculated. The hard-real time systems need to be designed to fulfil all the deadline requirements under its peak load.

To observe the real time execution period of the processes we are using Raspberry pi 3b+ which has an ARM11J6JZF-S-v6 operating at a standard frequency of 700MHz. Structural unit in a processor are analyzed for efficient usage and power utilization. Various peripherals are interfaced to operate in multitasking mode to note the change in relative deadlines for selected scheduling methods.

The microphone and camera interfaced with raspberry pi samples the respective inputs repeatedly to evaluate the execution time of each process at different frequencies. The camera is programmed to capture both still image as well as video sample, hence three processes are instantiated multiple times for experimental purpose.

The system supports an SD card interface to load required operating system and programs. 16GB SD card is installed to make adequate memory availability for system and system has 512MB SDRAM for runtime memory requirement along with two levels of cache memory system for better performance.

Display module is interfaced through HDMI port available on Raspberry board. The installed operating system provides GUI for working. The OS provides and interface to program using python programming language, however some programs are executed through terminal and output is observed.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication

II. LITERATURE SURVEY

A sensitivity analysis is provided for task deadlines in the context of dynamic-priority, pre-emptive, uniprocessor scheduling. The paper present a deadline minimization method that computes the shortest deadline of a periodic task. Real-time systems are often designed using a set of periodic tasks whose periods are usually governed by the application requirement and system is developed accordingly, the system performance can be improved by modifying deadlines wherever possible while still satisfying the requirements. Sensitivity analysis in real-time systems can be focused on changes in task computation times, using fixed priority analysis [1]. The paper proposes sensitivity analysis for deadline modification under EDF scheduling for periodic tasks in real time systems. In case of event driven applications sporadic tasks are well suited that periodic as they have an upper bound on their rate of occurrences [2].

Liu and Layland introduced Earliest Deadline First scheduling algorithm in 1973 which is widely used in systems with dynamic priority allocation [3]. Quick convergence Processor demand Analysis (QPA) algorithm determines an optimal system parameter by a single run. QPA algorithm provides efficient and exact sensitivity analysis real-time systems and using various starting values intends to improve the implementation of this sensitivity analysis [4].

The EDF algorithm is more efficient if the preemption is allowed compared to non-preemptive systems. The EDF scheduling is similar to FCFS method if the deadlines are deterministic. An analytical method for approximating the fractions of job or processes missing the deadline is useful to define the boundary within which the system can be operated using respective scheduling algorithm. If the system is user interfaced, then number of customers operating at any given time affects the rate at which the processes may exceed the defined deadline. In such cases and optimality property of EDF algorithm can be used to estimate the loss rate [5]. The feasibility analysis in EDF is most often based on processor demand. Once the processes are scheduled according to deadlines then at end of every deadlines processor demand of all processes in queue needs to be checked again to identify any new instances. Under multitasking situations the processes are executed based on their priority, hence there is possibility of missing the deadline of lower priority processes in uniprocessor system, Hence the processor handover the higher priority processes to respective accelerator which are usually designed to perform a specific task at much higher rate and core processor makes itself free for lower priority process establishing an parallel execution environment. This method of implementation is used only if the cost of development and implementation has enough flexibility as the introduction of accelerator proportionately increases the cost of the system [6].

The CPU utilization factor may continuously vary in case of multitasking systems bounded to be less than unity. There will always be a trade-off between flexibility and system overheads as implementation of flexible systems includes more exception handling modules and a greater number of service routines, the most common overhead under such situation is context switching time which increases overall turnaround time of the processes. An experimental using polynomial time complexity algorithm show increase in acceptance ratio. Both backward and forward iterations need to be carried out in order to check deadline requirements as process requests may be random in real time [7]. In many multitasking systems the number of processes in to be scheduled may change continuously. However the hard real time systems are to be designed to satisfy the deadline requirement under worst case execution time [8].

John Lehoczky, Lui Sha and Ye Ding states that Rate monotonic scheduling is optimal fixed priority algorithm in which the processes having least period is assigned highest priority. There are many advantages of Rate Monotonic scheduling algorithm in practical implementation and have gained importance accordingly [9] [10] [11]. The comparison of RM and EDF reveal the misconceptions related to EDF and RM policies [12]. The detailed comparison shows that the performance of EDF is better in many scenarios. Statistical study done by Lehoczky in 1989 shows the processor utilization in RM is about 88% for randomly generated task parameters [13].

Cervin proves that EDF scales the period if there is permanent overload in the system but RM may block lower priority tasks or processes [14]. The software embedded inside the system should support optimal scheduling for the application. Along with the hardware, programming part also plays important role in deciding the performance of the system. A good platform and application can improve the efficiency and aids envelopment of better systems. Programming languages are important for both control over real time aspects along with performance enhancement [15].

III. SYSTEM BLOCK DIAGRAM AND WORKING PRINCIPLE

The real time multitasking system developed to understand the performance and operation under different load conditions. Raspberry Pi3b+ processing unit is used as hardware platform to develop porotype. This is an ARM7 core-based processor having CPU operating at 700 Mhz standard frequency. It also consists and separate GPU and other supporting I/O peripheral units required for the application.

The Raspberry Pi has standard Raspbian operating system to maintain the performance efficiently which recommends minimum of 8GB SD card for OS installation. But having 16GB provides enough space and flexibility to upgrade and system and also install additional required tools & packages. The system is interfaced with peripherals required to execute the intended processes. Often it is operated as headless computer with prior configuration, but it can also be operated using generic USB computer keyboard or mouse. With appropriate device driver installed on underlying operating system virtually any other device / component compatible with USB can be interfaced.

Retrieval Number: 100.1/ijrte.A59000510121
DOI: 10.35940/ijrte.A5900.0510121

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication

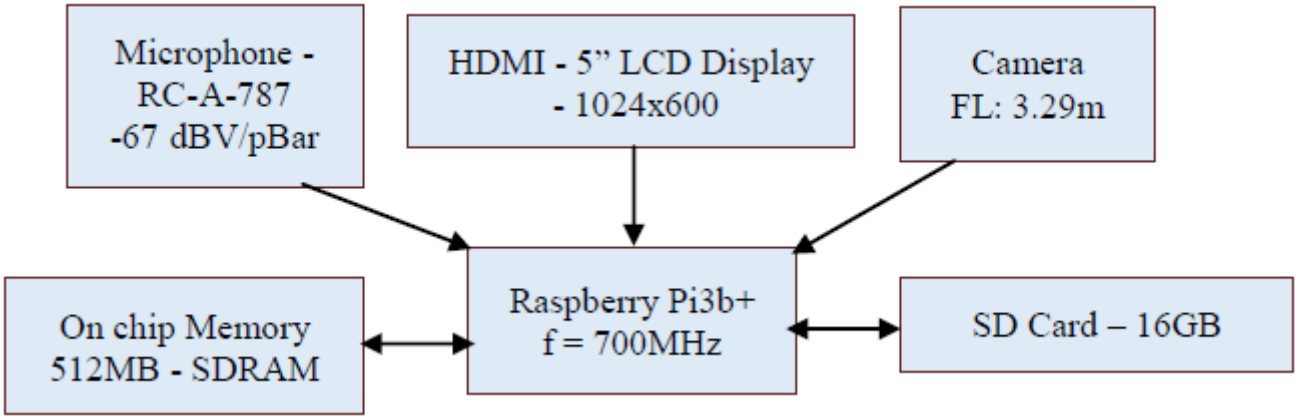

Figure 1: Raspberry Pi3b+ Block Diagram

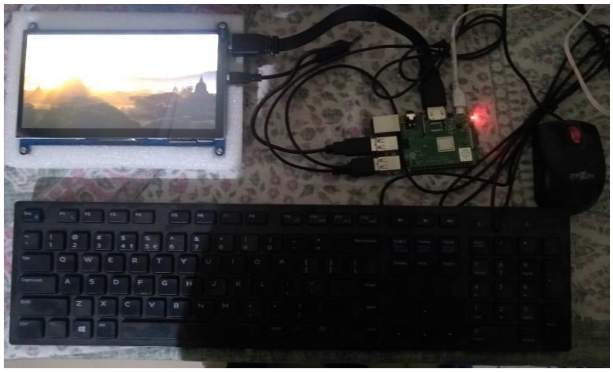

Figure 2: Hardware setup

New Out of Box Software (NOOBS) is used as an operating system platform to implement the code required for the application. The coding is done using Python and executed using Thonny compiler which is an in build cross compiler integrated along with Raspbian operating system or NOOBS.

A real time embedded system consisting N number of processes with TiExe execution time of any process X. The absolute deadline TAD of any process is defined as the time consumed by the process from the instance when process is arrived in the process queue i.e the time difference between the arrival time TA to completion of the execution, which includes expected waiting period TW and execution period TiExe as shown in equation 1. The waiting period is introduced to include context switching time in multitasking environment and can be excluded otherwise.

$$T_{AD}=T_W+T_{iExe} \quad (1)$$

Then the relative deadline of any process depends on the absolute deadlines of all higher priority processes in the process scheduling queue of the system as shown in equation 2. The relative deadline T_{RD} is defined as the summation of T_{AD} of all higher priority processes in the queue at any given instant of time.

$$T_{RD}=\sum T_{iAD} \quad \text{for i=0 to N - (X-1)} \quad (2)$$

$$T_{RD}=\sum (T_{iW} + T_{iExe}) \quad \text{for i=0 to N – (X-1)} \quad (3)$$

Average relative deadline T_{RD} is defined as the arithmetic mean of relative deadlines of each process.

$$T_{RDavg} = \frac{(\sum (N-(X-1))+ T_{iExe})}{(N-(X-1))} \quad \text{for i=0 to X-1} \quad (4)$$

Where N – (X-1) are the number of processes having higher priority in the scheduled process queue. Hence to calculate relative deadline, the arithmetic mean of all the scheduled processes are considered.

The worst-case scenario is considered for the system design i.e when all processes available in the system raise the request and scheduled in system process queue. The processing unit is expected to work at its best performance to meet the system requirements and satisfy the deadlines of every process. The scheduling algorithm implemented plays a crucial role in determining if all the processes are executed within the desired time duration. To satisfy this requirement, the turnaround time of every process should be less than its absolute deadline. The total turnaround time TT for the process is given by cumulative addition of individual process turnaround time T_{Ti} having higher priority in scheduling queue.

$$T_T=\sum T_{Ti} \quad \text{where i=0 to X-1} \quad (5)$$

Rate Monotonic (RM) Scheduling Analysis: The priorities in RM scheduling is decided depending on their frequency of arrival [13]. Hence the processes may take more time than their actual execution time due to preemption by higher priority processes. Accordingly the total turnaround time will determine if the processes are executed within the specified deadline.

Earliest Deadline First (EDF) Scheduling Analysis: The priority of the process is decided based on the absolute deadlines defined for the processes [7].

The five processes considered for the analysis are listed in the table 1 below with respective time slots in seconds and deadlines and related details required for scheduling. The scheduling period of 30 time slots is decided based on LCM of all deadlines.

The actual LCM is 24 but scheduling is done for 1.25 times of their LCM to accommodate more number of processes. The execution time shown in table 1 indicated time consumed by the process when it's executed independently and no other process in scheduled. However the total turnaround time of process changes when these processes are executed in multitasking environment.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication

The total turnaround time depends on type of scheduling, hence both RM and EDF scheduling methods are used to illustrate the effect of scheduling on process turnaround time.

IV. RESULTS AND DISCUSSION

For real time implementation and experimental analysis five processes are considered and their respective significance is explained below in this application. The scheduling process is carried out for both scheduling algorithm i.e. RM & EDF for comparative analysis, based on the results the better algorithm can be concluded under specific dynamic process load. The scheduling time slots considered for scheduling is 1 second under both the policies.

Video Capturing (P1): The camera is used for capturing still images as well as video capturing is RCM0M SY101S-A-E305654 -V2.1. CSI-2 (Camera Sensor Interface) is an interface between camera and host processor that allows the use of specially designed Raspberry Pi Camera Module to capture both image and video. A 15 cm flat ribbon cable (FRC) is used to connect the Camera module to CSI port of Raspberry pi module. Once connected then the data can be accessed using Multi Media Abstraction Layer (MMAL) or Video for Linux (V4L) Application programmer Interface (APIs) which runs over OpenMAX.

The video capture duration is 5 seconds with sufficient high resolution. The average execution time (4 executions) of the code is V(TEavg) = 5.1255 seconds when executed independently under single process execution mode and in multitasking mode the average execution time is V(TEavg) = 5.1299 seconds, the process takes little more time due to context switch and other overheads.

Video Play (P2): The video_play process is invoking video player at the background as another independent process. The video captured for 5 secs is played back using the display interfaced through HDMI port. NOOBS operating system by default includes omx player and same is used in application. The omx player compatible with .h264 format is invoked as sub process to share the display for play time.

The 5 seconds video captured for experimental analysis and execution time of this process is 0.0078 seconds and hence the total process duration is calculated by adding period of video and process execution time. The video_play process is consuming 5.0078 seconds to complete 5 seconds video playing.

Image Capturing (P3): The camera interfaced is enabled to capture still image on every run in .jpg format. The camera used for video capturing and still image are of same specifications and are used by both the processes on shared resource concept. Average execution time (3 executions) for single image capturing under independent execution mode I(TEavg) = 2.0216 seconds.

Audio Capture (P4): Audio is recorded for 2 seconds using the microphone interfaced through USB connector. Sampling frequency for recording is configured to fs = 44100 through program. The average execution time of this process observed is ACap(TEavg) = 3.5204 seconds. Therefore audio capture time for any duration can be calculated by adding 0.5204 to actual capturing duration.

Audio Play (P5): The recorded audio is played using 3.2mm jack connector. The configuration for audio capturing is done using pcm.mic API and audio play can be configured using pcm.speaker. On board speaker is used instead of another peripheral interface. Audio is played at 44000 hzwith 4096 buffer memory which is used to store the data to be played for seamless streaming.

The average execution time (4 executions) for this process is Aplay (TEavg) = 2.1646 seconds. The execution is for audio play of 2 Seconds duration. It's observed that the additional execution time taken by process is 0.1646 secs. Therefore we can calculate the execution time for any audio play duration by just adding 0.1646 secs to actual play time. If 3 seconds of audio play is considered then the total execution time will be 3.1646 seconds.

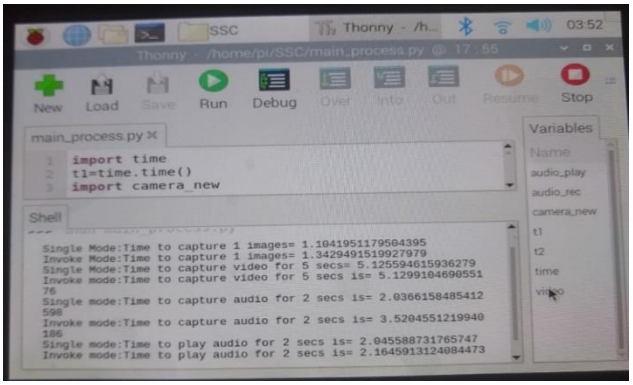

Figure 3: Real time measurement of execution time of processes

Table 1: Process Description

Process Process	T_{Exe} (Seconds)	T^*_{Req}	T_{AD}	Fp
P1	5.1255	06	12	30
P2	5.0078	06	12	15
P3	1.3442	02	04	05
P4	3.5204	04	08	10
P5	3.1646	04	08	30

T^*_{Exe} – Actual execution time ceiled to nearest integer to include context switching and other overheads time.

T_{Req} – Time slots required to execute

D – Deadline in Seconds

Fp – Frequency of the Process

Rate Monotonic (RM) Scheduling Analysis: The process scheduling is implementation using RM scheduling algorithm is shown in **table** 2.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication

Table 2: RM - Process Turnaround Time

Process	Priority	T_{Req}	T_{AD}	T_{RD}	T_{Exe}(RM)	Comment
P1	5	06	12	22	NS*	Miss
P2	3	06	12	12	30	Miss
P3	1	02	04	02	02	Meet
P4	2	04	08	06	08	Miss
P5	4	04	08	16	NS*	Miss

NS*: Non-schedulable as they cannot complete the execution within the total scheduling period of 30 time slots. The higher priority processes will continuously interrupt and these processes will never get sufficient time to complete their execution. The only way to address such issue is to reduce the frequency of higher priority process while operating hardware processing unit and its frequency is assumed to be constant. It's can be observed that for a required deadline periods the processes are not completely schedulable. Three processes out of five are missing the deadline as they are not getting sufficient number of slots for execution due to preemption of higher priority processes.

Earliest Deadline First (EDF) Scheduling Analysis: The process scheduling is implemented using EDF scheduling method is shown in table 3.

Table 3: EDF Process Turnaround Time

Process	Priority	T_{Req}	T_{AD}	T_{RD}	T_{Exe}(EDF)	Comment
P1	5	06	12	22	NS*	Miss
P2	4	06	12	16	PS#	Miss
P3	1	02	04	04	02	Meet
P4	2	04	08	06	08	Miss
P5	3	04	08	10	19	Miss

PS#: Partially scheduled

In EDF scheduling as well it can be observed that the processes are not able to satisfy the required deadlines limitations. Only three out of five has met the deadline and one is not able to schedule due to lack of time slots in total scheduling period.

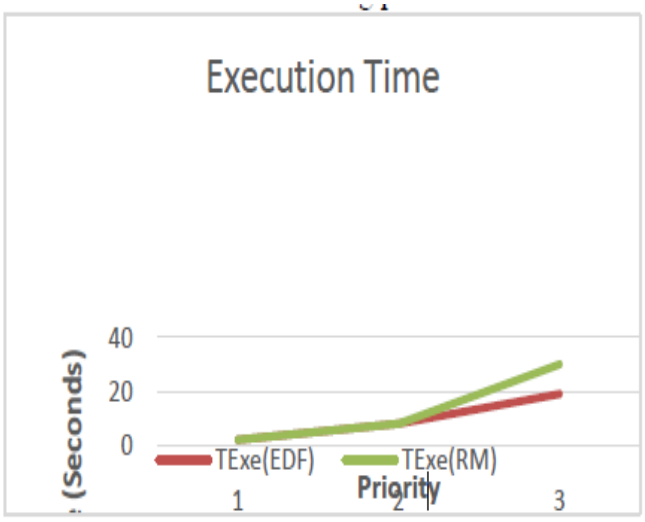

Figure 4: Comparison of Execution times using RM & EDF algorithms

V. CONCLUSION

In this work it can be observed that even though processes included in the multitasking embedded system are not able to satisfy the deadline requirements under both RM and EDF scheduling methods. But EDF performance is better compared to RM as it's able to successfully schedule 60% of the total processes and also partially complete fourth process. Hence the deadline relaxation required in case of EDF scheduling is lesser than that of RM.

The performance of the system can be improved and required deadlines can be met by implementing dynamic frequency scaling in processing unit. The Raspberry Pi also facilitates frequency scaling up to 1.4GHz. However it also important to note that the power consumption also increases in proportionate to operating frequency. Therefore proper cooling system need to be incorporated when processors are operated at its highest permitted operating frequency.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication

REFERENCES

1. Patricia Balbastre, Ismael Ripoll, Alfons Crespo "Minimum deadline calculation for periodic real-time tasks in dynamic priority systems", IEEE Transactions on Computers 2007 IEEE.
2. Kevin Jeffay, David Becker, David Bennett, Shaun Bharrat, Timothy Gramling, Mark Housel "The Design, Implementation and Use of a Sporadic Tasking Model", 27599-3175 USA April 1994.
3. C. L. Liu, James W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard- Real-Time Environment", Journal of the Association for Computing Machinery, Vo|, 20, No. 1, January 1973, pp. 46-61
4. Fengxiang Zhang, Alan Burns, Sanjoy Baruah "Sensitivity analysis of arbitrary deadline real-time systems with EDF scheduling", Springer Science Business Media, LLC 2011
5. Mehdi Kargahi, Ali Movaghar"A Method for Performance Analysis of Earliest-Deadline-First Scheduling Policy", Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN'04) 0-7695-2052-9/04 © 2004 IEEE.
6. Andrew Morton and Wayne M. Loucks "EDF Feasibility Analysis of Accelerated Tasks", 0840-7789/07©2007 IEEE
7. Michael Short Electronics & Control Group Teesside University, Middlesborough, UK "Improved schedulability analysis of implicit deadline tasks under limited preemption EDF scheduling" IEEE ETFA 2011.
8. Fengxiang Zhang · Alan Burns · Sanjoy Baruah "Sensitivity analysis of arbitrary deadline real-time systems with EDF scheduling", Springer Science Business Media, LLC 2011.
9. J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior", in IEEE Real-Time Systems Symposium, 1989, pp. 166–171.
10. Sprunt B, Kirk D, and L.Sha "Priority Driven Preemptive I/O Controllers for Real Time Systems", Proceedings of 15th International Symposium on Computer Architecture (1998), 152-159.
11. Stronsnider J K, Marchok T and J Lehoczky "Advanced Real Time Scheduling using the IEEE 802.5 Token Ring", Proceedings of the IEEE Real Time Systems Symposium (1988), 42-52.
12. Giorgio C. Buttazzo "Rate Monotonic vs. EDF: Judgment Day", 2005 Springer Science Business Media, Inc. Manufactured in the Netherlands.
13. Lehoczky, J. P., Sha, L., and Ding, Y. "The rate-monotonic scheduling algorithm: exact characterization and average case behavior", Proceedings of the IEEE Real-Time Systems Symposium 1989, pp. 166–171.
14. Chen Yao, Li Qiao, Li Zheng, Xiong Huagang "Efficient schedulability analysis for mixed-criticality systems under deadline-based scheduling", Chinese Journal of Aeronautics 2014 Ref CJA 305.
15. K. J. Ransom, C. D. Marlin and Wei Zhao, "An integrated environment for the development and analysis of hard real-time systems", Design of Real-Time Systems, Copyright © IFAC Real Time Programming 1991.

AUTHORS PROFILE

Sandeep S Chapalkar, Ph.D Perusing in Embedded design and development area, Visvesvaraya technological University, Belagavi, Karnataka, India. M.Tech in Digital Electronics and Communication, Visvesvaraya technological University Belagavi, Karnataka, India. B.E in Electronics and Communication Engineering, Karnatak University Dharwad, Karnataka, India. IEEE member.

K. Karibasappa, Ph.D -Machine Learning and Perception Using Cognitive Method, University College of Engineering, Burla, Orissa Sambalpur University, 2004, India • M.E-Master of Electronics and Telecommunication Engineering, Jadavapur University, 1998, Calcutta, West Bengal, India • B.E-Electronics and Communication Engineering, Malnad College of Engineering, Hassan, Karnataka, India. Lifetime Member if ISTE