International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-10 Issue-5, June 2021

A Survey on Different Real Time Operating Systems

Santhosh M S, Nagaraja G S

Abstract: ***To minimise development costs and enhance dependability, modern embedded system development is increasingly emphasising on software modularity and reuse. Microcontrollers are extensively employed in embedded applications that have a very specific and specialised job to complete. The embedded applications are always resource constraint which requires efficient utilization of available resources. A Real Time Operating System (RTOS) is frequently used in this context to plan task execution as well as enable intertask communication and synchronisation. This paper provides the survey of different RTOS available in market and their applications. Several open source RTOS such as Free RTOS, VxWorks, SmallRTOS and TinyOS are compared with respect to the scheduling algorithms used.***

Keywords: RTOS, Scheduler, Event Objects

I. INTRODUCTION

An RTOS (Real-Time Operating System) is a piece of software that allows users to easily swap between jobs, creating the impression that numerous programmes are running at the same time on a single processing core. Response time to external events is the parameter that differentiates an operating system such as Windows or Unix and the RTOS used in embedded systems. Usually, OS's have a non-deterministic, soft real-time response, where there are no guarantees as to when each assignment will be completed, but they will try to remain responsive to the user. An RTOS varies in that it usually provides a hard real-time response, providing a swift, highly deterministic response to external events to serve as a real-time application that processes data as it comes in, often without buffer delay.

The components of an RTOS is shown in fig 1.

The Scheduler: The scheduler component of RTOS determines the order in which the tasks are executed. The parameters considered for scheduling is decided by the scheduling algorithm used.

Symmetric Multiprocessing (SMP): Also known as multitasking, it is the number of concurrent tasks that can be handled by an RTOS for processing.

Function Library: It is an interface using which the kernel interacts with the application code. The application sends requests to and receives response from kernel using the function library.

Memory Management: this element takes care of memory allocation for all the process.

Fast dispatch latency: It is the time interval between suspending the current task and execution of next task from the ready queue, also known as context switch time.

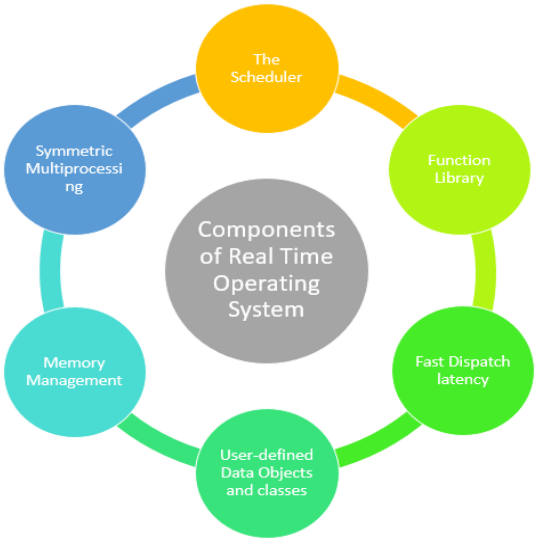

Fig 1 : Components of an RTOS

In today's embedded system applications, the idea of RTOS is critical, as it is responsible for everything from task scheduling to supporting high-level languages like C and Python.

It aims to improve efficiency, support idle processing and priority based scheduling. The factors that need to be considered while selecting RTOS are performance, middleware, embedded system usage, maximum consumption, task shifting, responsiveness and available system resources.

II. LITERATURE REVIEW

Paper [1] explains the basic parts of an RTOS and different scheduling algorithms present in an RTOS such as co-operative scheduling, round robin scheduling and preemptive scheduling It also describes the development of SmallRTOS, used mainly for PIC 18 series microcontrollers.

SmallRTOS is developed using C programming language and is based on round robin scheduling algorithm, where every task created is given the same amount of execution time called quantum.

Manuscript received on June 04, 2021.
Revised Manuscript received on June 11, 2021.
Manuscript published on June 30, 2021.
* Correspondence Author
Santhosh M S*, Student, Department of Computer Science and Engineering, Rashtreeya Vidyalaya College of Engineering, Bengaluru (Karnataka), India. Email: anushal.scn19@rvce.edu.in
Dr. Nagaraja G S, Professor & Associate Dean, Department of Computer Science and Engineering, Rashtreeya Vidyalaya College of Engineering, Bengaluru (Karnataka), India. Email: nagarajags@rvce.edu.in

SmallRTOS is simple, effective and can be used in real time applications which requires smaller memory. of the journal. There are two email address. It is compulsory to send paper in both email address.

Paper [2] describes the design of a coprocessor for dual core systems which performs conflict-free task scheduling. The study provides a solution based on two algorithms: the priority-based FIFO method, which is appropriate for non-real-time activities, and the Earliest Deadline First (EDF) approach, which has been shown to always discover an optimal ordering of difficult real-time jobs. Even though the mentioned algorithms uses different parameters for scheduling, the proposed RTOS can handle both types of tasks efficiently. The suggested coprocessor is designed to work with dual-core CPUs, allowing real-time embedded systems to perform better.

The use of FreeRTOS, an open source RTOS extensively utilised for low-cost applications, is explained in Paper [3]. The FreeRTOS operating system was chosen for the project because it offers a reasonable balance of API richness, execution-time overheads, and memory consumption. This work evaluates the basic performance of RTOS by considering various parameters such as context switching, semaphore synchronization overheads and jitter induced by low priority tasks. According to the results of the tests, synchronisation primitives have a tiny fixed overhead. A minor amount of jitter time was also discovered for the highest priority job, demonstrating that priority inversion is dependent on the operating system's internal implementation.

For FreeRTOS-based Embedded systems, the paper [4] evaluates several scheduling strategies. Dynamic priority real-time scheduling (DPS), such as Earliest-Deadline First (EDF), is one of the algorithms under consideration for review. It provides high levels of system schedulability. The overheads associated with two alternative EDF implementations are compared to Rate Monotonic Scheduling, a fixed priority scheme. The two EDF implementations differ in how priority queues are built, with one based on min-heap (EDF-H) and the other on multiple linked lists (EDF-L).For various sorts of task sets and system loads, runtime overheads and schedulability are taken into account. Experiments consistently show that EDF-L outperforms EDF-H in all experimental scenarios.

TinyOS is an embedded based operating system for low power wireless devices, such as those used in wireless sensor networks, as detailed in paper [5]. Tiny OS has limitations in terms of CPU scheduling has less memory which makes it suitable for WSN applications. Because of the First Come First Serve (FCFS) scheduling technique of TinyOS, its performance is hindered. This work analyses the scheduling mechanism used in TinyOS and discusses its disadvantages. It then suggests an enhanced design for the priority-based soft real-time task scheduling method as a result of the study. Experiments on sensor nodes reveal that the novel strategy can increase the communication performance of wireless sensor networks significantly.

The implementation of a least slack time first dynamic scheduling approach to serve a number of real-time applications on a single processor is presented in papers [6] and [7]. The suggested method implementation covers a variety of real-time applications and can be customised for a variety of tasks, including periodic, aperiodic, and sporadic jobs. The designed scheduler makes use of mutex and semaphore objects to support synchronization between the tasks and sharing of resources. The hardware used for implementing the algorithm is ARM Cortex M4 based processor. The simulation is tested and results are verified, which meets the desired requirements of a robust scheduler.

The quantitative and qualitative results of the analysis of real-time operating systems are presented in Paper [8]. Windows CE, VxWorks, Linux, QNX Neutrino and RTAI-Linux, which are widely used in industrial and academic environments, are among the systems examined for analysis. The response time, worst case response times for latency and latency jitter are the metrics used in the evaluations.

Table 1 shows the experimental results for different RTOS which considers the measured value for response time, interrupt latency and latency jitter in micro seconds.

Table1 : Worst times measured during experiment (in Micro seconds)

	Win CE	Neutrino	mC/OS II	Linux	RTAI	VxWorks
Response Time	20	20	1.92	13.89	5	3.85
Latency	99	35.2	3.2	98	11.4	13.4
Latency Jitter	88.8	32	2.32	77.6	7.01	10.4

III. CONCLUSION

There are already several ports available for popular Real Time Operating Systems such as FreeRTOS. But the actual Real Time OS behavior shall not be emulated because in most of the available ports, the different tasks are mapped to the threads offered by the specific platform (either Windows Threads or Posix Threads for Linux and MAC).

Most of the Real Time Operating Systems considered for survey implements busy waiting solution. Use of busy waiting solutions in embedded systems leads to increase in power consumption and more CPU utilization. One way to effeciently utilize the CPU and available resources is to use the windows event objects for synchronization.

REFERENCES

1. Sonia Zouaoui, Lotfi Boussaid and Abdellatif Mtibaa, “SmallRTOS: Microcontroller-based embedded multitasking”, International Conference on Engineering & MIS (ICEMIS), Monastir, Tunisia, 2017
2. Lukáš Kohútka and Viera Stopjaková, “Task scheduler for dual-core real-time systems”, MIXDES - 23rd International Conference Mixed Design of Integrated Circuits and Systems, 2016
3. Ivan Cibrario Bertolotti and Gilda Ghafour Zadeh Kashani, “On the performance of open-source RTOS synchronization primitives”, IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2015

4. Gessé Oliveira and George Lima, “Evaluation of Scheduling Algorithms for Embedded FreeRTOS-based Systems”, Brazilian Symposium on Computing Systems Engineering (SBESC), 2020
5. Anita Patil and Rajashree V. Biradar, “Scheduling techniques for TinyOS: A review”, International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), 2016
6. Rakesh Belagali, Sushant Kulkarni, Vinayak Hegde and Geetishree Mishra, “Implementation and validation of dynamic scheduler based on LST on FreeRTOS”, International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT), 2016
7. Yan Zhao, Qianping Wang, Wei Wang, Dong Jiang and Yiwen Liu, “Research on the Priority-Based Soft Real-Time Task Scheduling in TinyOS”, International Conference on Information Technology and Computer Science, 2019
8. Prasanna Hambarde, Rachit Varma and Shivani Jha, “The Survey of Real Time Operating System: RTOS”, International Conference on Electronic Systems, Signal Processing and Computing Technologies, 2014

AUTHORS PROFILE

Santhosh M S, is a MTech student at Department of Computer Science and Engineering, Rashtreeya Vidyalaya college of Engineering, Bengaluru, Karnataka. anushal.scn19@rvce.edu.in

Dr. Nagaraja G S, is working as professor and Associate Dean at Department of Computer Science and Engineering, Rashtreeya Vidyalaya College of Engineering, Bengaluru, Karnataka, India. nagarajags@rvce.edu.in