A Survey on The Implementation of Real Time Systems for Industrial Automation Applications

Vicent Rutagangibwa
PG Student
Electronics & Communications Engineering
Gujarat Technological University (GTU) Ahmedabad – India

Babu Krishnamurthy
Senior Technical Officer
C-DAC Pune

Abstract

Real Time Systems are at the heart of industrial automation applications and thus, a time response mechanism is required to be able to implement such systems. This paper presents a survey of Implementing Real Time Systems for industrial automation applications. Recently, PLCs have dominated industrial automation implementations but however, they do present some challenges especially in meeting real time constraints due to its centralized control and cyclically scanned program execution mechanisms. This paper proposes an alternative implementation approach using FreeRTOS platform that can act as a benchmark for time bound services. This would help in having a hybrid system that can work with PLCs and/or where possible replace PLCs for deterministic service delivery.

Keywords: FreeRTOS, Industrial Automation, Real Time Systems.

I. INTRODUCTION

Real Time Systems are required in industries in order to have a deterministic response to real world events. These events have deadlines to meet and can either be Soft real time deadlines or hard real time deadlines. Soft real time systems may somet imes miss deadlines without causing catastrophic consequences whereas hard real time systems must complete within its deadlines, otherwise the system fails completely and may cause catastrophic consequences to the machines and/or humans.

Industrial control systems are normally engaged with real-time operating systems to guarantee meeting real-time constraints [1]. IEC 61499 standard introduced the Event-Based Execution order for the program functional unit, but the drawback of this approach is the step-learning curve required for this implementation as it is very different from the earlier standard (IEC 61131) [1]. According to [1], when the functional program unit is invoked, it executes its code sequentially from beginning to end, this normally results in delay in execution of real time tasks. In FreeRTOS implementation, a program is organized in terms of task blocks independent of each other and each task can be configured to control the execution of the program functional unit. This technique helps execute the task as per requirements, thanks to the priority based scheduling mechanism of FreeRTOS implementations and thus, achieve the determinism nature of the industrial application requirements.

II. METHODS USED IN IMPLEMENTATITING REAL-TIME SYSTEMS

This survey is structured according to the key approaches on the implementation of real time systems in industries. These approaches consists of a Service-Oriented approach, an Event-Driven approach, Object-Oriented approach and Supervisory controls. This section finally presents the implementation of the proposed new approach based on FreeRTOS platform. Industrial controllers work basically by cyclically reading and storing inputs, executing user program(s), and finally writing the output [1]. The centralized programmable control model of PLCs may cause non-deterministic results and overheads when two or more controllers communicate via a network, and its centralized, cyclically scanned program execution model according to [2] limits the reuse of the components in case of reconfiguration.

The IEC 61499 industrial standard reference architecture facilitates the development of distributed automation systems with decentralized logic in order to solve the problem of centralized traditional approach of PLCs. This standard addresses the trend of increasing importance of software in automation systems design by improving portability, configurability, and interoperability of automation systems according to [2]. The PLCs are often considered as the main program implementations for industrial automation systems. They are connected with the plant peripherals, such as sensors and actuators, via direct electric wiring to their input/output (I/O) ports, or via remote I/O modules which communicate with the PLC via field area networks (fieldbuses) [2].

A. Event - Triggered Approach

Event-Based execution approach requires that the operating system implements an event triggered execution of task blocks as in IEC 61499 standard, and the [1] showed that this can be achieved by keeping activated functional blocks only when required. However, it was shown that achieving determinism using this approach is not easy as function blocks must be executed concurrently and also be able to meet logical constraints (i.e. resource sharing). According to the discussion in [3], Event-driven solution can widely be used by assuring better behavior of complex Operating system constructs, e.g. by using a native RTOS instead of Linux with real-time improvements.

B. Service - Oriented Approach

This [1] presents a Service-Oriented alternative that is used to link the program execution unit and the real time constraints. Here, the supervisor enables a service request only when the required resources are available and the program execution units implements the service request that is associated with a certain task algorithm. The resource is then only releases when it is no longer required. In FreeRTOS Implementation, the release of a resource is done automatically by the idle task and the developer does not need to worry of any memory issues. The advantage of this approach is that programming of functional units is made as simple as possible. In industrial automation, the purpose of a real-time program in any industry is to know the occurred events and produce results in a manner that a desired behavior is obtained. The desired behavior can be expressed in terms of sequences and constraints on these sequences [1]. Service requests are assumed to be controllable events and the service completion events are modeled by uncontrollable events since its occurrence cannot be disabled. In [3], a service oriented approach with real-time capabilities was proposed and a temporal behavior of each activity is isolated as much as possible by using dedicated hardware and software. In this model, Linux kernel is modified to provide temporal isolation. The solution [3] aimed at designing the entire real-time complex industrial automation systems with existing real devices and non- existing-but-will-exist devices (i.e. virtual devices).

C. Mathematical Methods

According to [2], Formal models in industrial automation softwares create a firm foundation in creating reliable softwares with properties that can be guaranteed by design to meet real time constraints. The specifics of any automation systems is that, they are control system where the dynamic changes of the plant must be taken into considerations when control software is designed in order to have predictable performance of the industrial real time systems. The progress in formal synthesis of control algorithms based on specifications and constraints has been reviewed and it was found out in [2], that the difficulty of formal synthesis is high computational complexity of the process. T. Winkler [4] presented a new methodology to enhance the synthesis process by application of new structural analysis methods.

D. Object - Oriented Approach

Object Oriented Programming (OOP) according to [2] is a popular programming method using “Objects” to design applications and computer programs. This method has the advantage of efficient code reuse, increased safety and stability of system‘s software. According to [1], OOP technologies fail to provide all the expected benefits, particularly in handling the complexity and performance of a given system. In Industrial automation, objects are often regarded as physical devices built in a specific industrial domain contexts, thus OOP are expected to be more robust than it has been in software engineering. Industrial control systems are not like general computer systems and hence, OOP tools used for industrial automation should satisfy some additional requirements [5] like; I/O configuration and direct access to I/O signals, Multi-paradigm programming. The overall recommendation of using OOP, is that it should be optional and act as a stepwise and reversible transition [1] instead of using it on its own.

III. INDUSTRIAL AUTOMATION

Currently, Industrial automation systems are experiencing a paradigm shift due to incorporation of new technologies, such as embedded real time devices and communication networks [3]. The demand of automation systems that utilize hardware, software and communication infrastructure to meet real time constraints are highly needed in today‘s industrial system‘s implementations. This calls for alternative approaches in real time systems implementation for industrial automation applications from the traditional PLC dominated systems that may result in unpredictable results. Intelligent industrial automation systems rely heavily on a distributed computer-based infrastructure, where smart sensors and actuators, intelligent machines, robots, and other automation deices can interact using industrial protocols and can also be to take decisions in real time [3]. In these systems, there are some challenging issues like; system-level communication, devices synchronization, and the integration of new devices to the system. Hence there is a huge demand for sophisticated tools to design and implement intelligent complex industrial automation systems [3]

IV. REAL TIME SYSTEMS

A Real Time System any computer system where the correctness of the system behavior depends not only on the logical results of the computations, but also on the physical time when these results are produced. The system behavior means the sequence of outputs in time of a system. For any physical plant to be deterministic, it requires a controller capable of taking some input signals and producing some specific output signals within a specific time frame (i.e. Deadline). If the controller outputs occur outside such time frame, those outputs will no longer be valid, and will produce malfunctions or even a catastrophic failure in the physical plant (i.e. for hard real time systems). Real time properties have to be handled via embedded controllers that are introduced to the system by encapsulation.

There has been a proposal on handling real-time issues in functional blocks (FBs), but it's not yet implemented. According to [3]. There are a lot of studies in the fields of increasing computation power of processors and increasing execution speed via parallel execution or optimization. Real time embedded operating system can effectively manage embedded processor resources, shorten the design cycle of application software, meet real-time needs in embedded. Applications and improve the reliability of embedded systems. The industry has already recognized the importance of providing safe and reliable real-time operating systems for industrial applications according to [6] and the academic community is actively working on tools and methods that can improve the current standards of software quality.

A. FreeRTOS

FreeRTOS is a real-time, multitasking, preemptive operating system for embedded devices. Task is the most important concept in FreeRTOS and each executing program is a task under the control of a real-time operating system [6]. When Compared with other real-time embedded operating systems, FreeRTOS supports time slice rotation priority scheduling, open source code, and can easily be migrated to different processors such ARM, AVR, MSP430 etc. [7]. The reference [8] describes FreeRTOS operating system on the LPC2292‘s portable file and code changes. This showed how to migrate FreeRTOS to a different processor for a specific application. However, according to [7], the LPC2292 is not suitable for domestic industrial automation applications due to the task switching and stack overflow issues in industrial architectural set up. It was showed in [7] that to solve this problem of task switching and stack overflow for industrial real-time embedded systems, a redesigned migration method of resource scheduling, equipment management and memory allocation for FreeRTOS system based processors is required.

FreeRTOS kernel provides a relatively straightforward interface and specification by replacing traditional messaging module for upper application development. Each application coding is written based on FreeRTOS kernel, and implemented by a system call [9]. The task communicates with other tasks by a standard interface (i.e. queues, semaphores, etc.). Upper multitasking applications are implemented based on real-time kernel task management and device management. FreeRTOS embedded system‘s kernel also provides a system protection about the environment of the interrupt according to [7].

With the introduction of Real-Time OS in embedded systems field, specific controllers usually used to control single processes are now being replaced by shared controllers which are ported with Real-Time OS running multiple control algorithms concurrently. In this [10], the control system can be viewed as a mapping from plant outputs to actuation commands in order to achieve specific control objectives, with learning as the process of modifying this mapping to improve future closed-looped system performance.

B. FreeRTOS Scheduling Mechanism

The scheduler task starts when the *vTaskStartSchedule()* API is called. The kernel can decide to suspend and later resume a task many time during its lifetime. Tasks do not know when they are suspended or resumed by the kernel, the scheduler has to guarantee that upon waking up, and a task has a context identical to that immediately prior to its previous execution to avoid scheduling errors which may result to scheduling latencies. The process of saving the context of a task being suspended and restoring the context of a task being resumed is called context switching and the context of a task is saved in its own stack [6]. Each time a tick generated interrupt occurs, FreeRTOS saves the context of the current running task and executes the API function *vTaskSwitchContext ().* This function invokes the scheduler which selects the highest priority task that is ready to run and then executes it.

V. DISCUSSIONS AND FUTURE WORK

With industrial automation application showing tremendous growth towards real-time dependent system, FreeRTOS system implementations can be prototyped to pave way for industrial research. The proposed system implementation is shown in the fig 1. In this implementation, the LPC1769 board shall be used as the process controller which is based on ARM Cortex-M3 processor. The sensors S1, S2 and S3 represent the industrial sensor network. PLCs based industrial controllers can thus be interfaced with this proposed system via the industrial Ethernet protocol as shown in fig 1. This implementation is an ongoing research at GTU Ahmedabad – India, and hopes to achieve the deterministic performance of industrial applications by serving as a benchmark for time bound industrial applications.

ACKNOWLEDGMENT

Great thanks to Mr. Babu Krishnamurthy, Mr. Rajesh Sola for their outstanding contributions and great expertise towards this research work.

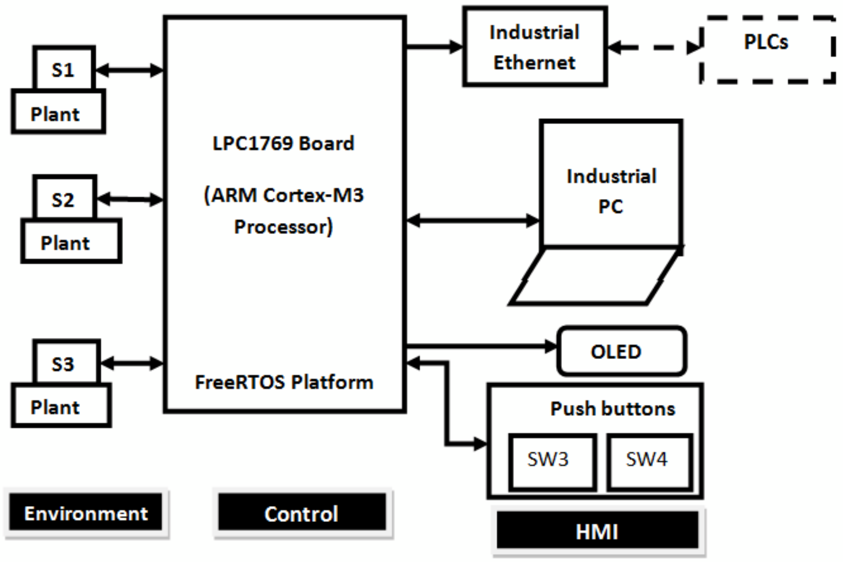

Fig. 1: Block Diagram of The Proposed Implementation

References

[1] F. Basile, P. Chiacchio, and D. Gerbasio, "On the implementation of industrial automation systems based on plc," in IEEE, Transaction science and Engineering, vol. 10, no. 4, October 2013, pp. 990 – 1003.
[2] I. S. M. Valeriy Vyatkin, "Software engineering in industrial automation: State-of-the-art review," in IEEE, Transaction on industrial informatics, vol. 9, no. 3, August 2013, pp. 1234 – 1249.
[3] D. Fennibay, A. Yurdaku, and A. Sen, "A heterogeneous simulation and modeling framework for automation systems," in IEEE Transactions on Computer-Aided Design of Integrated circuits and systems, vol. 31, no. 11, November 2012, pp. 1642 – 1655.
[4] H. L. T. Winkler and H. M. Hanisch, "A new model structure based synthesis approach for distributed discrete process control," in Proc, 9th IEEE int. Conf. Ind. Inf, Lisbon, Portugal, 2011, pp. 527 – 532.
[5] U. Schunemann, "Programming plcs with an object-oriented approachl," In Autom. Technol. Practice, no. 2, November 2007, pp. 59 – 63.
[6] F. Ferreira, G. He, and S. Qin., "Automated verification of the freertos scheduler in hip/sleek," in IEEE Sixth International Symposium on Theoretical Aspects of Software Engineering, 2012, pp. 51 – 58.
[7] S. Niu, J. Zhang, B. Wang, and X. Fu, "Analysis and implementation of migrating real-time embedded operating system freertos kernel based on s3c44b0 processor," in Fourth International Symposium on Information Science and Engineering, 2012, pp. 430 – 433.
[8] P. Huang, "Implementation of porting rtos FreeRTOS to arm7," in J. Chinese electronic business communications market, June 2009, pp. 59 – 64.
[9] R. Barry, "Real time engineering ltd: FreeRTOS main website, http://www.freertos.org."
[10] A. J. Pankaj Sagar, Naveen N, A. N. K. H, and L. M., "Implementation of cmnn based industrial controller using vxworks rtos ported to mpc8260," in Third International Conference on Advances in Computing and Communications, 2013, pp. 239 – 242.