Development in Real Time Operating System-Case Study: Free RTOS Vs Windows CE

Vinayak D. Shinde¹ Jyoti Vaibhav Jadhav² Pallavi Kamalakar Bhoir³
¹H.O.D. ²M.E. Scholar
³Department of Computer Engineering
Shree L.R. Tiwari College of Engineering, Mira Road, Thane-401107.

Abstract— An application is said to be Real Time if it is required to complete it’s work & deliver it’s services on time. Real time applications usually runs on real time operating system (RTOS). It processes the data as soon as it comes without delay. Internet of things (IOT) devices are embedded systems which receive and transmit information over the network. Real time operating system is preferred option for such devices. This paper focuses on the features and architecture of RTOS. It gives comparative study of FreeRTOS and Windows CE which are the two among top real time operating systems used in embedded systems market.

Key words: RTOS, IOT, embedded systems

I. INTRODUCTION

There is an evolution of embedded system in modern times. Nowadays embedded systems are ubiquitous. It is on smartphones, air conditioners, cars etc. These systems are contribution in making our life safe and comfortable. Safety is one crucial aspect in which we are depending on these embedded systems. The core of the system is the operating system. Most of these systems use RTOS.

An RTOS is an operating system specialized for real time operations. An RTOS must have following characteristics:

1) Response time should be predictable
2) It should be deterministic

There are other qualities like speed, features set, small size etc which are important as a whole operating system but are not what really characterize an RTOS.

Wikipedia defined RTOS as “A real-time operating system (RTOS) is an operating system (OS) intended to serve real-time application process data as it comes in, typically without buffering delays. Processing time requirements (including any OS delay) are measured in tenths of seconds or shorter”[1].

Real time systems are classified as hard Real time systems and soft Real time systems. A Soft real time system is a system in which if a deadline does not meet it can have undesirable effects. But the effects are not catastrophic. There is a performance degradation. Examples are computer games, multimedia transmission and reception. Hard Real Time system is a system in which if a deadline does not meet it can have catastrophic effects. Examples are air traffic control, nuclear plant control and vehicle subsystem control.

II. THE INTERNET OF THINGS AND RTOS

A. IoT Device:

The Internet of Things (IoT) is the network of physical objects—devices, vehicles, buildings and other items—embedded with electronics, software, sensors, and network connectivity that enables these objects to collect and exchange data [4].

There is a great demand of data-driven world. An IoT device will require robust and reliable software and this requirement is satisfied by a real-time operating system (RTOS).

B. RTOS Key Features for IoT:

Scalable: IoT can different classes of devices ranging from simple to complex, single application to multi-application. Single RTOS can be scaled to meet requirements of multiple class products. Historically 8 bit and 16 bit MCUs are used but nowadays price of 32 bit MCUs price has dropped down. So popularity of embedded devices is increased. IoT devices can be of small and large MCUs. A scalable RTOS runs on a variety of 16 and 32-bit MCUs. Scalable RTOS meet memory requirements and reduce processor demands.

Modular: The IoT and machine to machine networks are evolving very fast nowadays. So RTOS must be built on modular, upgradable structure which will separate the core kernel from middleware, protocols, applications, and other packages. As RTOS has stable core, middleware, new packages can be upgraded or added without modifying the core. Modular architecture of RTOS allow manufacturers to enrich their products with new features and enhancements without changing the core of the system.

Connected: Internet of things should be connected over network. In any industry we want embedded devices to connect to each other and communicate with each other. RTOS should support transfer data in and out of the device via Wi-Fi, Ethernet, USB, or Bluetooth. RTOS will allow to select particular protocol to save memory for communication.

Reliable: Many IoT devices work in safety critical environment where repair or replacement is almost impossible. So RTOS should have a safety critical certification. This is crucial for safety and reliability of the device. If the RTOS is already certified, then the building the device in such environment would be easy.

III. RTOS ARCHITECTURE

For simpler applications, RTOS is mostly a kernel. As per requirement if the complexity increases then various modules like networking protocol stacks debugging facilities, device I/Os are included to the kernel. The general architecture of RTOS is shown in the fig. [1]
RTOS Kernel and Its Services:

Kernel is the smallest and central part of an operating system. It has following services: managing memory and devices, providing an interface for software applications. Additional services such as managing protection of programs and multitasking.

1) Task Management

Task management facilitates programmers to design their software which can be a number of separate programs each handling a unique goal. This service has mechanisms of scheduler and dispatcher that creates and maintain task objects.

2) Synchronization & Intertask Communication:

Synchronization is important for tasks to share mutually exclusive resources like devices, buffers, etc. It allows multiple tasks to be executed concurrently. Task synchronization is achieved using two types of mechanisms: 1) Event Objects and 2) Semaphores.

Intertask communication is sharing of data among tasks through sharing of memory space, transmission of data and etc. Few of mechanisms available for executing intertask communications are 1) Message queues 2) Pipes 3) Remote procedural calls (RPC)

3) Memory Management:

An embedded RTOS usually struggle to achieve the functionality needed for the user’s applications. Two types of memory management are included in RTOSs. They are 1) Stack 2) Heap managements.

4) Timer Management:

In RTOS, system and user tasks are always scheduled to perform after a specified duration. To provide such scheduling, there is a need for a periodical interrupt. This interrupt keep track of time delays and timeout. Most RTOSs nowadays provide both relative timers and absolute timers. Relative timers work in units of ticks. Absolute timers work with actual date and time. For each timer, RTOSs provide services like task delay and task alert. Another timer service is provided to determine whether tasks have met or missed their real-time deadlines.

5) Interrupt Handling:

An interrupt is a hardware mechanism used to inform the CPU that an asynchronous event has occurred. A fundamental challenge in RTOS design is to support interrupts and because of that allow asynchronous access to internal RTOS data structures. Data integrity is achieved by restricting interrupts from occurring when modifying a data structure. Interrupts are disabled when RTOS is performing critical operations.

6) Device I/O Management:

An RTOS kernel is often provide device I/O management service. It provides a uniform framework and supervision facility to an embedded system for organizing and accessing large numbers of different hardware device drivers. However, most device driver APIs and supervisors are standard.

Top RTOSs available in market:

1) Integrity-178B: Integrity-178B has a unique feature of an EAL6 rating by the NSA [2]. It is used in several military jets as well as the commercial airframes Airbus.

2) VxWorks: VxWorks is designed for use in embedded systems requiring real-time, deterministic performance. It provides safety and security certification, for industries, like aerospace and defense, medical devices, industrial equipment, robotics, energy, transportation, network infrastructure, automotive, and consumer electronics [3].

3) QNX: QNX was one of the first commercially successful microkernel operating systems and is used in a variety of devices including cars and mobile phones. It is very small in size and practically run on any cpu which is used in modern embedded systems.

4) FreeRTOS: FreeRTOS is small and simple. The kernel consists of only three or four C files. It makes the code readable, easy to port, and maintainable. There are a few assembly functions which are included wherever needed. Its famous for compactness and speed of execution. FreeRTOS can be thought of as a 'thread library' rather than an 'operating system'...

5) Windows CE: It is developed by windows based on windows NT used for embedded devices. It is optimized for devices which has very less memory. It even works with device having 1 megabyte memory.

FreeRTOS:

(pronounced “Free-arr-toss”) FreeRTOS is an open source real time operating system for embedded System.

FreeRTOS supports many different Architecture[5]. FreeRTOS is a popular real time operating system, which supports about 35 microcontroller architectures. The FreeRTOS project was founded by Richard Barry from Real Time Engineers Ltd. which are the owners and maintainers of the project. FreeRTOS started circulation in 2003 its latest version is 8.2.2 released on 12th August 2015. FreeRTOS is Designed to be small & simple. the kernel consists of three and four C files. The mostly FreeRTOS written in C[6].

Architecture: FreeRTOS can be supports many different architectures ARM7, ARM cortex-m3, version PICs silicon Labs 8051,x86 etc. FreeRtos designed to be small, simple and easy to use the FreeRtos

Hardware-dependent and Hardware-independent code to get system up and running. FreeRTOS is only three source (.c) files and a handful of header files. the code lines 9000 including comments and blank lines and binary code Image less than 10KB. FreeRTOS code breaks down into three main areas: tasks, communication, and hardware interfacing[7].
B. Top RTOSS Available in Market:

1) Integrity-178B: Integrity-178B has a unique feature of an EAL6 rating by the NSA [2]. It is used in several military jets as well as the commercial airframes Airbus.

2) VxWorks: VxWorks is designed for use in embedded systems requiring real-time, deterministic performance. It provides safety and security certification, for industries like aerospace and defense, medical devices, industrial equipment, robotics, energy, transportation, network infrastructure, automotive, and consumer electronics [3].

3) QNX: QNX was one of the first commercially successful microkernel operating systems and is used in a variety of devices including cars and mobile phones. It is very small in size and practically run on any cpu which is used in modern embedded systems.

4) FreeRTOS: FreeRTOS is small and simple. The kernel consists of only three or four C files. It makes the code readable, easy to port, and maintainable. There are a few assembly functions which are included wherever needed. It’s famous for compactness and speed of execution. FreeRTOS can be thought of as a ‘thread library’ rather than an ‘operating system’.

5) Windows CE: It is developed by windows based on windows NT used for embedded devices. It is optimized for devices which have very less memory. It even works with device having 1 megabyte memory.

IV. FREERTOS

(pronounced “Free-arr-toss”) FreeRTOS is an open source real time operating system for embedded System FreeRTOS supports many different Architecture[5]. FreeRTOS is a popular real time operating system, which supports about 35 microcontroller architectures. The FreeRTOS project was founded by Richard Barry from Real Time Engineers Ltd. which are the owners and maintainers of the project. FreeRTOS started circulation in 2003 its latest version is 8.2.2 released on 12th August 2015. FreeRTOS is Designed to be small & simple the kernel consists of three and four C files. The mostly FreeRTOS written in C[6].

A. Architecture:

FreeRTOS can be supports many different architectures ARM7, ARM cortex-m3, version PICs silicon Labs 8051, x86 etc. FreeRtos designed to be small, simple and easy to use the FreeRtos Hardware-dependent and Hardware-independent code to get system up and running. FreeRTOS is only three source (.c) files and a handful of header files. The code lines 9000 including comments and blank lines and binary code Image less than 10KB. FreeRTOS code breaks down into three main areas: tasks, communication, and hardware interfacing[7].

B. Features:

1) The FreeRtos is very small memory footprint.
2) low overhead.
3) Fast execution
4) semaphores to manage resource sharing between multiple tasks
5) priority-based multitasking capability
6) Tick-less option for low application
7) choose of scheduling policy
8) pre-emptive(RR)
9) cooperative

C. Scope:

1) The FreeRTOS support Multitasking task.
2) Easy to use
3) Small footprint
4) FreeRTOS is a popular real time operating system. which supports about 35 microcontroller architectures.
5) complex application can be set of smaller and manageable tasks.

D. Limitation:

FreeRTOS support uses standalone board support package (BSP) library and same limitations as standalone mode. lower RAM usage when compared to an equivalent task comes at the cost of some restrictions on how a coroutine can be used. Coroutines are more restrictive and complex to use than tasks.

V. WINDOWS CE (COMPACT EDITION):

Microsoft announced Windows CE at the COMDEX expo in 1996. It was demonstrated on stage by Bill Gates and John McGill [8]. It supports x86, MIPS and 32-bit ARM Platforms Windows ce is a very popular embedded operating system. For PDAS and mobiles. Window can easily develop application for windows CE.Windows CE 5.0 is the latest version. Windows mobile software for pocket PC and smartphone are also based on the core of windows CE.Windows CE uses Memory management ROM and RAM. The ROM Read Only Memory is a Windows CE system is like a small read only hard disk. The Random access memory In a windows CE system is divided into two area: Program memory & object store. Windows CE is a 32-bit operating system so it support 4GB virtual address space. Windows CE designed specifically for handheld device.
A. Architecurer:
Windows CE is 32-bit Operating system, so it supports 4GB virtual address space. Windows CE is designed specifically for embedded developers who need to bring new devices in the market quickly and at the lowest possible cost. Windows CE architecture breaks down into four parts: User Mode, Operating System, Kernel Mode, and Hardware[9].

B. Features:
1) Handheld and similar mobile device.
2) Data exchange between stationary and mobile computer.
3) Windows CE is designed for devices that have minimal memory; one megabyte of memory is enough for a Windows CE kernel to run.
4) Windows CE OS image creation and integration of operating system designs based on CE.

C. Scope:
Windows CE is based on Windows 95 with the user interface, for small devices. Specially designed for microcomputers. These microcomputers are mostly used as handheld computer or personal digital assistant (PDA).

D. Limitation:
The Microsoft file in Windows CE do not provide all of the functionality that is available on the corresponding Windows desktop applications. There is no support for files created in the Microsoft Office 97. There is no support for the Microsoft Visual Basic® and Macro programming language. Windows CE does not provide the same set of fonts that is available on Windows-based desktop operating systems.

VI. COMPARISON BETWEEN THE FREERTOS AND WIN CE

<table>
<thead>
<tr>
<th></th>
<th>FreeRTOS</th>
<th>Win CE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>License</td>
<td>Modified GNU GPL</td>
</tr>
<tr>
<td>2</td>
<td>Source Model</td>
<td>Open Source</td>
</tr>
<tr>
<td>3</td>
<td>Target Uses</td>
<td>embedded</td>
</tr>
<tr>
<td>4</td>
<td>Status</td>
<td>active</td>
</tr>
<tr>
<td>5</td>
<td>Platform(s)</td>
<td>ARM, AVR, AVR32, ColdFire</td>
</tr>
</tbody>
</table>

Table 1:

VII. CONCLUSION
So RTOS is an operating system that supports real-time applications by giving logically correct output within the specified deadline. Basic Structure is exactly similar to traditional OS but, in addition, it provides mechanisms to allow real-time scheduling of tasks. It may possible that real-time operating systems can not increase the speed of execution but they can provide much more precise and predictable timing characteristics than general-purpose OS. Nowadays majority of devices are intelligent devices which make up IoT. And majority of these devices are embedded devices which are running under RTOS. RTOS has the characteristics to realize the enormous opportunities of IoT.

REFERENCES