Universidad Católica
"Nuestra Señora de la Asunción"

Teoria y Aplicación de la Informática II

Sistemas Operativos de Tiempo Real

Autor:
Ramon Insaurralde

Docente:
Ing. Juan de Urraza

Resumen

Entrega de trabajo práctico de la materia de Teoria y aplicación de la informática II el cual tiene como objetivo la investigación y posterior analisis de diferentes temas de ambito cientifico. En este caso particular el trabajo se centra en los *Sistemas Operativos de Tiempo Real* los cuales son muy útiles en las mediciones y control de aplicaciones, y como de alguna manera difieren de los Sistemas Operativos de objetivo general como Windows.

Índice

1. Introducción

La mayoria de los sistemas embebidos estan limitados por restricciones de tiempo real. En los controles de producción las maquinas deben recibir sus ordenes al tiempo justo para garantizar el buen funcionamiento de la fabrica y para satisfacer las necesidades de los clientes.O en los vuelos, los sistemas de control de vuelos contienen muchisimos artefactos cuyas operaciones dependen en exclusiva del tiempo, como por ejemplo, los controles de combustion de las turbinas, etc.

"Tiempo Real"Significa que el sistema informático ya no controla su propio dominio de tiempo, ahora el progreso del tiempo del ambiente es el que dicta como el tiempo progresa dentro del sistema.

Las secciones siguientes mostraran los conceptos basicos y las terminologias relacionadas a los Sistemas operativos de Tiempo Real asi como diferentes aplicaciones actuales que necesitan de este tipo de sistema operativo.

2. Sistemas Operativos de Tiempo real (RTOS)

2.1. Concepto

En general, un sistema operativo (OS) es responsable de la gestión de los recursos de hardware de una computadora asi como el alojamiento de aplicaciones que se ejecutan en el equipo. Un RTOS realiza estas actividades, pero también está especialmente diseñado para ejecutar aplicaciones con una sincronización muy precisa y un alto grado de fiabilidad. Esto puede ser muy importante en los sistemas de medición y automatización en donde el tiempo de inactividad es costoso o un retraso de algún programa podría causar un riesgo de seguridad.

Para ser considerado .[en] tiempo real", un sistema operativo debe tener un tiempo máximo conocido para cada una de las operaciones críticas que realiza (o por lo menos ser capaz de garantizar el máximo la mayor parte del tiempo). Algunas de estas operaciones incluyen llamadas al OS e interrumpciones. Los sistemas operativos que pueden garantizar un tiempo máximo para estas operaciones se refieren comúnmente como "tiempo real duro", mientras que los sistemas operativos que sólo puede garantizar un máximo la mayoría de las veces son referidos como "soft. en tiempo real .[En] la práctica, estas categorías estrictas tienen utilidad limitada - cada solución RTOS demuestra características rendimiento único.

Para comprender plenamente estos conceptos, es útil considerar un ejemplo. Imagínese que usted está diseñando un sistema de airbag para un nuevo modelo de coche. En este caso, un pequeño error en el tiempo (haciendo que el airbag se despliegue demasiado pronto o demasiado tarde) podría ser catastrófico y causar lesiones. Por lo tanto, se necesita un sistema de tiempo real; que garantize que ninguna sola operación excederá ciertas limitaciones de tiempo.

2.2. Terminologías

Terminologías importantes:

- Determinismo: Una aplicación (o una pieza crítica de una aplicación) que se ejecuta en un sistema operativo de tiempo real se conoce como determinista si su temporización se puede garantizar dentro de un cierto margen de error.
- Soft vs Hard en tiempo real: Un sistema operativo que absolutamente puede garantizar un tiempo máximo para las operaciones que realiza se conoce comoen tiempo real duro. En contraste, un sistema operativo que por lo general pueden realizar las operaciones en un determinado tiempo se denomina tiempo real suave.

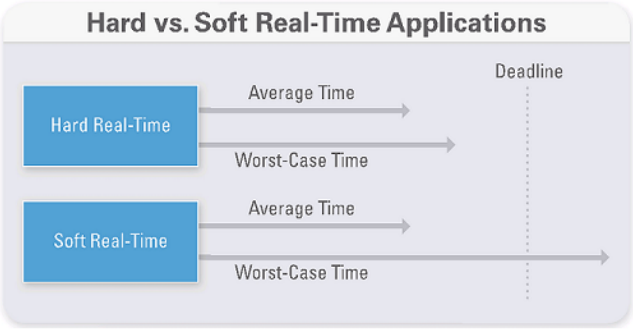

Figura 1: Hard and Soft

- Jitter: La cantidad de error en la temporización de una tarea sobre iteraciones subsiguientes de un programa o de bucle se conoce como jitter. Los sistemas operativos de tiempo real están optimizados para proporcionar una baja cantidad de jitter cuando se programa correctamente.

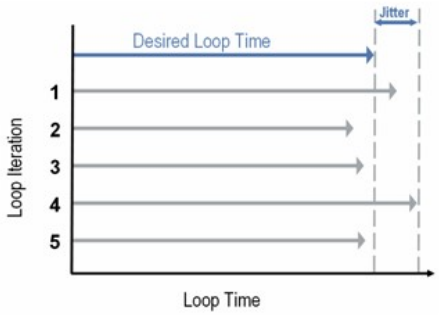

Figura 2: Jitter

2.3. Caracteristicas Generales

Usado típicamente para aplicaciones integradas, normalmente tiene las siguientes características:

- No utiliza mucha memoria
- Cualquier evento en el soporte físico puede hacer que se ejecute una tarea
- Multi-arquitectura (código portado a cualquier tipo de CPU)
- Muchos tienen tiempos de respuesta predecibles para eventos electrónicos

Se caracterizan por presentar requisitos especiales en cinco áreas generales:

- Determinismo
- Sensibilidad
- Control del usuario
- Fiabilidad
- Tolerancia a los fallos
- Son de tiempo compartido

En la actualidad hay un debate sobre qué es tiempo real. Muchos sistemas operativos de tiempo real tienen un planificador (en inglés conocido como scheduler), diseños de controladores que minimizan los periodos en los que las interrupciones están deshabilitadas, un tiempo finito conocido (casi siempre calculado para el peor de los casos, término que en inglés se conoce como worst case) de la duración de interrupción. Muchos incluyen también formas especiales de gestión de memoria que limitan la posibilidad de fragmentación de la memoria y aseguran un límite superior mínimo para los tiempos de asignación y retiro de la memoria asignada.

2.4. RTOS: Generalidades

En contraste con los sistemas operativos en tiempo real, los sistemas operativos más populares para el uso personal (como Windows) son llamados sistemas operativos de propósito genera (SOPG). Mientras más información técnica detallada sobre cómo los sistemas operativos en tiempo real difieren de los sistemas operativos de propósito general se da en una sección de abajo, es importante recordar que hay ventajas y desventajas de ambos tipos de sistema operativo. Los sistemas operativos como Windows están diseñados para mantener la capacidad de respuesta del usuario con muchos programas y servicios que se ejecutan (garantizar la "justicia"), mientras que los sistemas operativos en tiempo real están diseñados para ejecutar aplicaciones críticas en forma fiable y con una sincronización precisa (presta atención a las prioridades del programador).

2.5. RTOS: Manejo de Prioridades

En la programación de una aplicación, la mayoría de los sistemas operativos (de cualquiertipo) permite al programador especificar una prioridad para la general aplicación e incluso para diferentes tareas dentro de la aplicación (hilos). Estas prioridades sirven como una señal para el sistema operativo, que dicta que operaciones son los más importantes.

El objetivo es que si dos o más tareas están listas para funcionar al mismo tiempo, el sistema operativo ejecuta la tarea con la prioridad más alta.

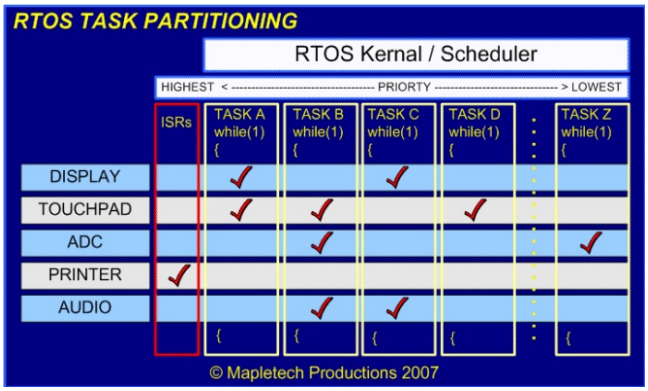

Figura 3: Manejo de Tareas

En la práctica, los sistemas operativos de propósito general no siempre siguen estas prioridades programadas estrictamente. Debido a que los sistemas operativos de propósito general están optimizados para ejecutar una variedad de aplicaciones y procesos de forma simultánea, por lo general trabajan para asegurarse de que todos tareas reciben al menos algún tiempo de procesamiento. Ademas, en los sistemas operativos de propósito general las tareas de baja prioridad pueden en algunos casos aumentar su prioridad por encima de otra tareas de mayor prioridad. En la mayoria de los RTOS, si una tarea esta usando el cien por ciento de el procesador, ninguna tarea de menor prioridad se ejecutará hasta que la tarea de mayor prioridad termine. Todos los programadores de aplicaciones en RTOS deben tener esto muy en cuenta. Esto asegura una cierta cantidad de tiempo de ejecución para cada tarea, pero significa que los deseos del programador no siempre se cumplen.En contraste con las (SOPG), los RTOS siguen las prioridades del programador en forma mas estricta.

2.6. RTOS: Clases de Aplicaciones

Los sistemas operativos en tiempo real fueron diseñados para dos clases generales de aplicaciones: de respuesta evento y de control de bucle cerrado.

Las aplicaciones de respuesta de eventos, tales como la inspección visual automatizada de piezas de la línea de montaje, requieren una respuesta a un

estímulo dado una cierta cantidad de tiempo. En este sistema de inspección visual, por ejemplo, cada parte debe ser fotografiado y analizado antes de la línea de montaje se mueva. Al programar cuidadosamente una aplicación que se ejecuta en un sistema operativo de tiempo real, los diseñadores que trabajan en la respuesta evento aplicaciones pueden garantizar que la respuesta va a suceder de manera determinista (dentro de una cierta cantidad máxima de tiempo). En vista de el ejemplo de inspección partes, utilizando un sistema operativo de propósito general podría resultar que una parte no se este inspeccionando al tiempo - por lo tanto, se tendria que retrasar la línea de montaje,haciendo que esa parte analizada se deseche, o se envíe como una parte potencialmente defectuosa.

Por el contrario, los sistemas de control en bucle cerrado, como el de un sistema de control de velocidad cuzero del automóvil, continuamente procesan datos de feedback para ajustar uno o mas outputs . Debido a que cada valor de Output depende de procesamiento de los datos de entrada en una cantidad fija de tiempo, es fundamental que los plazos de bucle se cumplan con el fin de asegurar que se producen las outputs correctos. ¿Qué pasaría si un sistema de control de crucero no ha podido determinar que el ajuste del acelerador debe estar en un punto dado en el tiempo? Una vez más, el sistema operativo de tiempo real puede garantizar que los datos de entrada del sistema de control se procesen en la misma cantidad de tiempo (con un worst case fijo máximo).

También debe notarse que muchas aplicaciones que deben funcionar durante largos períodos de tiempopueden beficiarse de la fiabilidad de que una RTOS puede proporcionar. Ya que los sistemas operativos en tiempo real suelen ejecutar un conjunto mínimo de software en lugar de muchas aplicaciones y procesos al mismo tiempo, estas son muy adecuadas para sistemas que requieren de 24-7 de operación o donde el tiempo de inactividad es inaceptable o caro.

2.7. RTOS: Latencia de Interrupción

La latencia de interrupción se mide como la cantidad de tiempo entre cuando un dispositivo genera una interrupción y cuando se atiende ese dispositivo. Mientras los sistemas operativos de propósito general pueden tener una cantidad variable de tiempo para responder a una interrupción determinada, los sistemas operativos en tiempo real deben garantizan que todas las interrupciones serán atendidos dentro de una cierta máxima cantidad de tiempo. En otras palabras, la latencia de interrupción de las RTOS deben ser delimitadas. La latencia comprende dos componentes deterministas y al menos dos no deterministas:

- Determinista: - Conmutación de Hardware y entrar en el manejador de interrupción - Ejecución de código del controlador. Mientras que el tiempo de ejecución depende de el número de instrucciones, el tiempo en un controlador específico es constante (si no se interrumpe!)
- No determinista:

  Una interrupción no se sirve si: a) el sistema de interrupción esta desactivado (permitido por una RTOS) o b) durante el mantenimiento de una

interrupción con mayor prioridad de hardware.

2.8. RTOS: Gestión de Procesos

Gestión de proceso: Realización de tareas cuasi paralelo en un procesador utilizando procesos o hilos (proceso ligero) por:

• Mantener estados de procesos, gestión de colas de proceso.

• Tareas preventivas (cambio de contexto rápido) y rápida manejo de interrupciones.

•Planificación de la CPU (garantizando plazos, minimizando proceso de los tiempos de espera, la equidad en la concesión de recursos comopotencia de cálculo).

•Proceso de sincronización (secciones críticas, semáforos,monitores, de exclusión mutua).

•Comunicación entre procesos (buffering).

•Apoyo de un reloj de tiempo real como una referencia de tiempo interna.

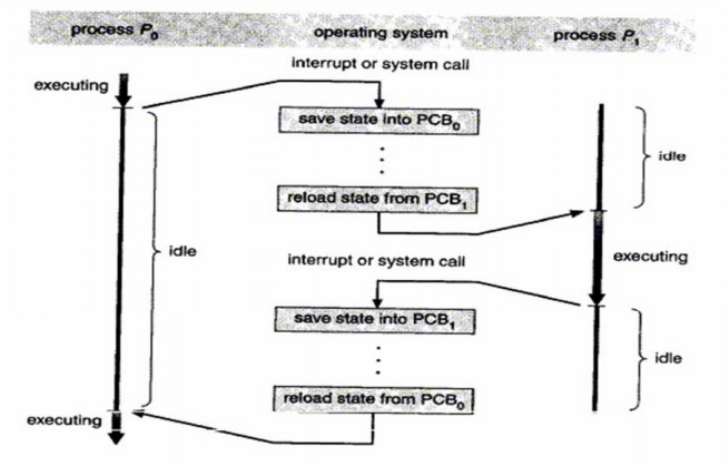

Figura 4: Gestión de Porcesos

2.9. RTOS: Estado de Procesos

Run:

• Una tarea entra en este estado, ya que comienza a ejecutarse en el procesador.

Ready:

•Estado de aquellas tareas que están listos para ejecutar, pero no pueden ser ejecutadas debido a que el procesador está asignado a otra tarea.

Wait:

•Una tarea entra en este estado cuando se realiza una sincronización primitiva para esperar un evento, por ejemplo, una primitiva espera en un semáforo. En este caso,la tarea se inserta en una cola asociada con el semáforo. las tareas en la cabeza se reanudan cuando el semáforo está desbloqueado por una señal primitiva.

Idle (inactivo):

•Un trabajo periódico entra en este estado cuando finaliza su ejecución y tiene para esperar el comienzo de la siguiente periodo.

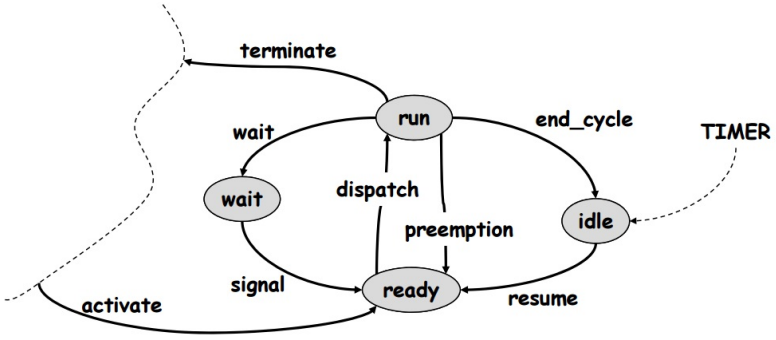

Figura 5: Estado de Porcesos

2.10. RTOS: Performance

Un error común es que los sistemas operativos de tiempo real tienen mejor rendimiento que otros sistemas operativos de propósito general. Mientras que los sistemas operativos en tiempo real pueden proporcionar un mejor rendimiento en algunos casos debido a la menor multitarea entre aplicaciones y servicios, esto no es una regla. El rendimiento real de una aplicación dependerá de la velocidad de la CPU, la arquitectura de memoria, las características del programa, y otras caracteristicas. Aunque los sistemas operativos de tiempo real pueden o no aumentar la velocidad de ejecución, estos proporcionan mucha mas precisión y predictivilidad en las características de tiempo de ejecución que las de propósito general.

3. RTOS mas utilizados y sus aplicaciones

3.1. Windows CE

Windows CE (conocido oficialmente como Windows Embedded Compact y anteriormente como Windows Embedded CE,1 también abreviado como Win-CE) es un sistema operativo desarrollado por Microsoft para sistemas embebidos. Windows CE no debe confundirse con Windows Embedded Standard, que

es un sistema basado en Windows NT; Windows CE está desarrollado independientemente.

La versión actual de Windows Embedded Compact funciona en procesadores Intel x86 y compatibles, además de los tipos MIPS y ARM.

Se puede encontrar en teléfonos inteligentes, notebook, hasta en pocket pc y gps.

El último sistema operativo creado a base de Windows CE fue Windows Phone 7 y sus actualizaciones (7.5 y 7.8)

3.2. Symbian

Symbian es un sistema operativo propiedad de Nokia, y que en el pasado fue producto de la alianza de varias empresas de telefonía móvil, entre las que se encontraban Nokia, Sony Mobile Communications, Psion, Samsung, Siemens, Arima, Benq, Fujitsu, Lenovo, LG, Motorola, Mitsubishi Electric, Panasonic, Sharp, etc. Sus orígenes provenían de su antepasado EPOC32, utilizado en PDA's y Handhelds de PSION.

El objetivo del Symbian era crear un sistema operativo para terminales móviles que pudiera competir con el de Palm o el Windows Mobile de Microsoft y posteriormente Android de Google , iOS de Apple, Windows Phone de Microsoft y BlackBerry OS de Blackberry.

3.3. RTAI

RTAI (Real Time Application Interface) Interface para Aplicaciones en Tiempo Real, es una implementación de Linux para tiempo real basada en un principio en RTLinux, y actualmente en ADEOS, no es un sistema operativo tal como VXworks o QNX. Se basa en el núcleo Linux, proporcionando la capacidad de hacerla completamente requisable (preemptable). RTAI añade un pequeño núcleo Linux de tiempo real bajo el núcleo estándar de linux y trata al núcleo linux como una tarea de menor prioridad. RTAI además proporciona una amplia selección de mecanismos de comunicación entre procesos y otros servicios de tiempo real.

Adicionalmente, RTAI proporciona un módulo llamado LXRT para facilitar el desarrollo de aplicaciones de tiempo real en el espacio de usuario.

3.4. VxWorks

3.4.1. Descripción

Fue desarrollado por la empresa VxWorks WRS (Wind River System Inc.) tiene un microkernel, alto rendimiento, sistema operativo escalable y en tiempo real que soporta una amplia gama de protocolos de red, y pueden combinarse de acuerdo a las necesidades del usuario, y su abierta la arquitectura y la compatibilidad con los estándares de la industria para que los desarrolladores sólo tienen que hacer un mínimo de trabajo para diseñar sistemas eficaces adaptados

a las diferentes necesidades de los usuarios. Además de la destacada actuación del sistema operativo, la compañía también ofrece excelentes herramientas de desarrollo de sistemas WRS sistema operativo en tiempo real. Puede apoyar casi toda la plataforma de trabajo Tornado Tornado y procesador objetivo, las herramientas proporcionadas pueden ser utilizados para todos los equipos de destino, y tiene dos modos de depuración: El modo del sistema y el modo misión. Sistema de procesamiento de datos de telemetría en tiempo real utiliza en tiempo real multitarea plataforma de sistema operativo basado en x86 de Intel y de alta fiabilidad VxWorks bus industrial. Plataforma de desarrollo compatible con el lenguaje de programación de alto nivel y la aguja sistema de compilación CompactPCI VxWorks C para una variedad de procesadores, puede proporcionar un mejor control en tiempo real, la ejecución multitarea, y la asignación y gestión de los recursos.

3.4.2. Historia

VxWorks iniciadó a finales de 1980 como un conjunto de mejoras para un simple RTOS llamados VRTX vendidos por Sistemas Ready (convirtiéndose en un producto de Mentor Graphics en 1995). Wind River adquirió los derechos para distribuir VRTX y significativamente mejorada añadiendo , entre otras cosas, un sistema de archivos y un entorno de desarrollo integrado. En 1987, anticipándose a la terminación de su contrato de distribuidor de Sistemas Ready, Wind River desarrolló su propio núcleo para reemplazar VRTX dentro VxWorks.

VxWorks hitos clave son:

1980: (procesamiento de 32 bits) - VxWorks añade soporte para los procesadores de 32 bits

1990: (Internet) - VxWorks 5 se convierte en la primera RTOS con una pila de red

2000: (multi-core) - VxWorks 6 soporta SMP y añade plataformas específicas de la industria de derivados

2010s: (procesamiento de 64 bits y el Internet de las Cosas) - VxWorks añade soporte para el procesamiento de 64 bits e introduce VxWorks 7 de la IO.

VxWorks es propiedad de Intel.

3.4.3. Caracteristicas

Las características distintivas de VxWorks son:

- la compatibilidad POSIX
- el tratamiento de memoria
- las características de multi-procesador
- una shell de interfaz de usuario
- monitor de rendimiento y depuración de código fuente y simbólico.

VxWorks se usa generalmente en sistemas embebidos. Al contrario que en sistemas nativos como Unix, el desarrollo de vxWorks se realiza en un "host"que ejecuta Unix o Windows.

En la actualidad, vxWorks puede ejecutarse en prácticamente todas las CPU modernas del mercado de sistemas embebidos.Esto incluye la familia de CPUs x86, MIPS, PowerPC, SH-4, ARM, StrongARM y xScale.

3.4.4. Ejemplos de uso de VXWORKS

1-El MARS Curiosity Rover's: Funciona con 2,5 millones de líneas de C sobre un procesador fabricado por BAE RAD750. El sistema operativo que utiliza es de Wind River VxWorks RTOS. El RTOS en cuestión puede ser programado en C, C ++, Ada o Java. Sin embargo, sólo el C y C ++ son estándar en el sistema operativo, Ada y Java son compatibles con extensiones. Wind River proporciona una enorme cantidad de detalles en cuanto a los cómos y los porqués de VxWorks. El chipset subyacente es absurdamente robusto. Sus características pueden no parecer mucho al principio, pero se les permite tener una y sólo una "pantalla azulçada 15 años.Tener en cuenta, esto es bajo el bombardeo de la radiación que mataría a un ser humano muchas veces. En el espacio,la robustez prevalece sobre la velocidad. Por supuesto, la solidez de esa manera tiene un costo. En este caso, se trata de unos 200,000a 500.000.

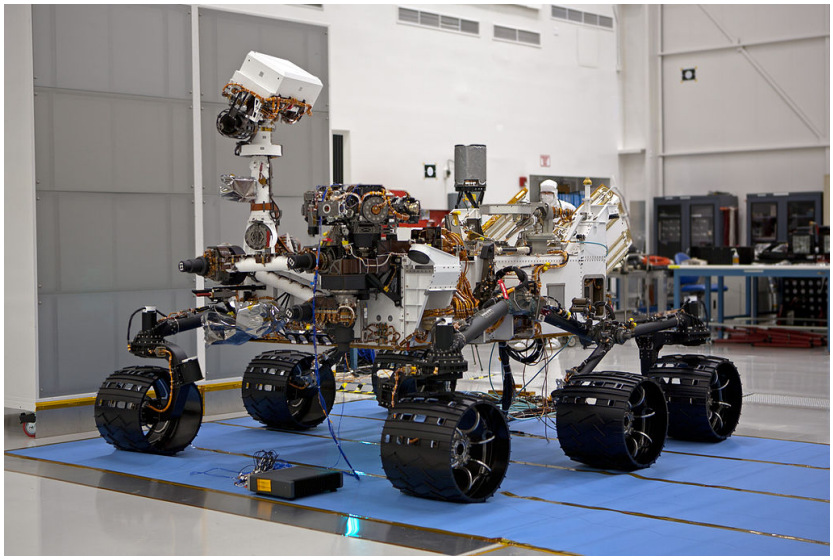

Figura 6: Curiosity

2-ASIMO: Un acrónimo de Paso Avanzado en Movilidad Innovadora, es un robot humanoide diseñado y desarrollado por Honda. Introducido el 21 de octubre de 2000, ASIMO fue diseñado para ser un asistente móvil multifuncional. Con aspiraciones de ayudar a los que carecen de plena movilidad, ASIMO

se utiliza con frecuencia en las manifestaciones en todo el mundo para fomentar el estudio de la ciencia y las matemáticas.

Figura 7: Robot ASIMO

4. El futuro de los RTOS

Impulsado por la convergencia de la tecnología en la nube, los volúmenes de datos de rápido crecimiento y la creciente cantidad de dispositivos conectados, el Internet de las Cosas (IoT) plantea nuevos retos y presenta una serie de nuevas oportunidades que las empresas de todos los tamaños y sectores pueden aprovechar ahora. Este sistema de sistemas es fundamental para la realización de negocios de valor para desbloquear la visión oculta en los datos, la identificación y la creación de nuevos servicios, la mejora de la productividad y la eficiencia, para mejorar la toma de decisiones en tiempo real, la solución de problemas críticos, y el desarrollo de nuevas e innovadoras experiencias de usuario . Miles de millones de dispositivos y sistemas inteligentes conforman el Internet de las Cosas. La mayoría de estas çosas"son sistemas, muchas de las cuales ejecutan un sistema operativo de tiempo real (RTOS) incrustado.

Para aprovechar al máximo la oportunidad que ofrece la Internet de las cosas, los fabricantes de sistemas embebidos deben cumplir varios retos:

- Llevar los dispositivos conectados al mercado más rápidamente
- Diferenciar los productos con características de vanguardia y capacidades
- Seguridad en cuento a los riesgos que la conectividad omnipresente de la Internet de las cosas implica
- Construir la flexibilidad en los productos existentes con el fin de ser capaz de aprovechar las nuevas oportunidades de mercado a medida que surgen.
- Asegúrarse de que la oferta de productos sigue siendo pertinente y competitivo ya que los mercados evolucionan
- Reducir los costos y los riesgos de desarrollo de sistemas

Para ayudar a los fabricantes de dispositivos integrados frente a estos retos, un RTOS debe evolucionar para ofrecer la escalabilidad, modularidad, conectividad, seguridad y un conjunto de características de vanguardia que son demandados por el nuevo, super conectado, conciente de la seguridad y manejado remotamente por las redes M2M (machine to machine) y el IoT.

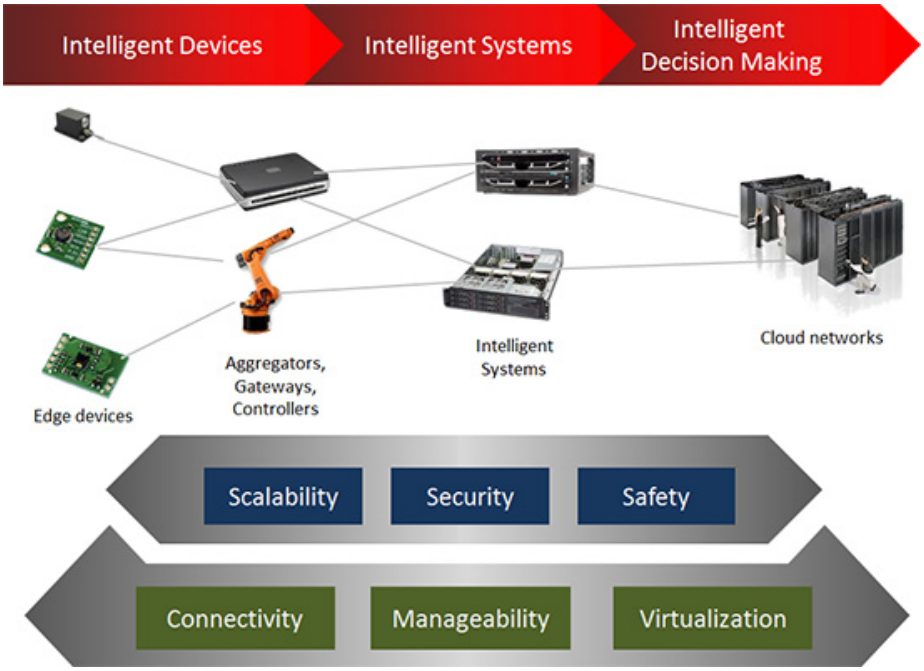

Figura 8: Future RTOS

4.1. Escalabiliadad

El Internet de las cosas puede crear un incentivo para que los fabricantes de dispositivos integrados puedan mantener una cartera de productos más amplia que incluye diferentes clases de dispositivos que van desde pequeños, los dispositivos simples, de una sola aplicación a gran escala de sistemas complejos, multi-aplicaciónes. Una sola RTOS que puede escalar para satisfacer las necesidades únicas de funcionalidad y potencia de procesamiento de múltiples clases de productos puede ayudar a los fabricantes de sistemas integrados a aumentar el retorno de su inversión, reducir los costos de desarrollo mediante el aprovechamiento de las economías de alcance y reducir el tiempo de llegada al mercado.

4.2. Modularidad

La arquitectura modular de un RTOS ayudará a los fabricantes de dispositivos integrados a poder mejor diferenciar sus productos y mantenerlos competitivos durante períodos más largos de tiempo enriqueciendolos con nuevas características y capacidades sin cambiar el núcleo del sistema ya que las normas y exigencias del mercado evolucionan. El nuevo RTOS también permitirá a los fabricantes extender la vida útil del Core del sistema para varias generaciones de productos, lo que aumenta el retorno de su inversión ROI.

4.3. Conectividad

Un RTOS reinventado para el IoT debe ser compatible con los estándares de comunicación líderes en la industria y protocolos tales como CAN, Bluetooth, Continua, ZigBee, Wi-Fi y Ethernet, y entregar capacidades de red de alto rendimiento. Además, una naturaleza modular del nuevo RTOS puede ayudar a dispositivos existentes con las opciones de conectividad necesarios para que muchos de los dispositivos anteriormente desconectados pueden ser llevados en línea de nuevo sin volver a trabajar en el núcleo de su software integrado.

4.4. Seguridad

Un aspecto crítico del IoT es la seguridad, y sistemas integrados de próxima generación deben ser diseñados pensando en la seguridad. Un RTOS ganador para el IoT sería dar a los clientes la flexibilidad para diseñar su sistema embebido con el nivel necesario de seguridad mediante el aprovechamiento de un amplio conjunto de funciones integradas en la cobertura. Un buen RTOS necesita para tener funciones de seguridad no sólo para proteger contra malwares y aplicaciones no deseadas o delincuentes, sino también para ofrecer almacenamiento seguro de datos y la transmisión y prueba de falsificaciones en diseños. El apoyo a nivel de sistema operativo de estas características es fundamental ya que la adición de ellos a nivel de usuario o aplicación es ineficaz, costoso y arriesgado.

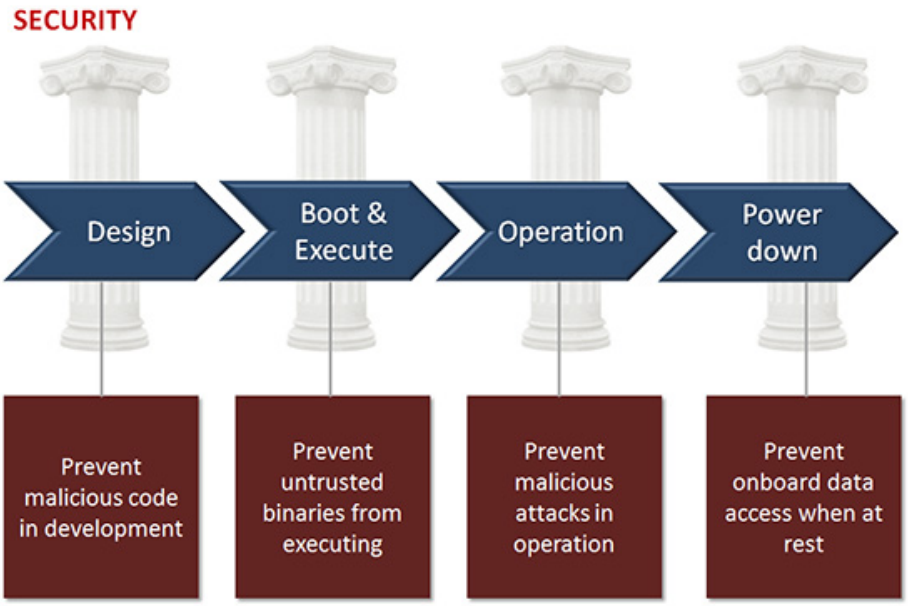

Figura 9: Seguridad RTOS del futuro

5. Conclusion

Este trabajo intentó reflejar de manera rápida y sencilla las caracteristicas, aplicaciones y escalabilidad de los sistemas operativos de tiempo real llevando al lector a entender los aspectos tecnicos y aplicativos de el mismo. Asi como hacer entender las diferencias que existen entre los RTOS y los SOPG los cuales difieren en caracteristicas y aplicaciones funcionales.

Los sistemas operativos de tiempo real son extremadamente necesarios ya que nos ayudan a tener perspectivas tecnológicas que ayudan en todos los aspectos inclusive los más basicos de las personas del mundo.

La era en la que nos encontramos require soluciones modulares, configurables y escalables de RTOS, todo progreso que se realize en mejorar los aspectos que citamos más arriba ayudará a un RTOS del futuro que mejore la mayoría de los procesos que hoy en dia aún se deja en la capa de aplicación la cual le rinde propensa a errores y elevados costos a los implementadores.

6. Bibliografía

Referencias

[1] Website, Octubre 2015.
http://www.tik.ee.ethz.ch/education/lectures/ES/slides/6_RTOS.pdf.

[2] Website, Octubre 2015.
http://www.cis.upenn.edu/~lee/06cse480/lec-RTOS_RTlinux.pdf.

[3] Website, Octubre 2015.
http://www.springer.com/cda/content/document/cda_downloaddocument/9781402094354-c2.pdf?SGWID=0-0-45-699603-p173865105.

[4] Website, Octubre 2015.
www.ni.com/white-paper/3938/en/pdf.

[5] Website, octubre 2015.
http://eecatalog.com/intel/2014/01/14/the-internet-of-things-defines-the-future-of-the-rtos/.

[6] Website, Octubre 2015.
https://es.wikipedia.org/wiki/Sistema_operativo_de_tiempo_real.

[7] Website, octubre 2015.
http://www.uio.no/studier/emner/matnat/fys/FYS4220/h12/undervisningsmateriale/forelesninger-rt/2012-3_realtime_operating_systems.pdf.