ARQUITECTURA DE SISTEMAS DE TIEMPO REAL

Optativa 1° Cuatrimestre. Quinto Curso de Ingeniería Informática

Facultad de Informática y Estadística.

Profesores: D. Antón Civit Balcells.

PROGRAMA (1999/2000)

Condicionantes.

El alumno de Segundo Ciclo debe formarse con vistas a dos posibilidades: el acceso al mundo laboral y la continuación de estudios en el Tercer Ciclo. Para este último caso, la formación debe ser generalista, orientada a los conceptos y bases teóricas que permitan una visión académica y científica de la Informática. Para el primer caso, la formación del Segundo Ciclo debe incluir las arquitecturas, haciendo énfasis en las más extendidas y útiles en el mundo real, que no ha sido posible estudiar en Primer Ciclo sin el conocimiento que se estudió en las arquitecturas avanzadas de las asignaturas de Cuarto curso Arquitectura de Sistemas Paralelos I y II. Se supone una base en los conceptos de estructura de computadores y en la interconexión de procesadores a sistemas de memoria o entrada/salida, y también se supone que el alumno está bien familiarizado con los conceptos básicos de diseño de sistemas operativos.

Por otra parte, el diseño de cualquier sistema de control por computador eficiente y fiable o, como caso particular, el diseño de cualquier controlador de robot implica el conocimiento de los conceptos fundamentales de los sistemas en tiempo real. Esto implica, en nuestra opinión, que la asignatura de Arquitectura de Sistemas en Tiempo Real sería fundamental para los alumnos que deseen formarse en estos campos.

Objetivos.

Esta asignatura trata de estudiar los requisitos que introduce el tiempo real en el diseño de un sistema informático. Puede definirse sistema de tiempo real (STR) como aquel que responde ante un evento en un tiempo predecible, determinista, es decir menor que cierta cota temporal. Esta definición de sistema de tiempo real parece, a priori, no tener relación con la arquitectura de un sistema, pero condiciona definitivamente su estructura tanto a desde el punto de vista software como hardware.

Además, y lo que es más importante, **la mayoría de los sistemas informáticos que existen son *necesariamente* de tiempo real**. Para muchos alumnos de esta carrera esta afirmación les parecerá increíble, dado que están acostumbrados a máquinas de uso general, es decir, empleadas dentro de un centro de cálculo, de una oficina o de una clase, donde no se requiere que el ordenador esté preparado para responder en un tiempo determinista ante nada (o casi nada, pues comentaremos ejemplos donde se requieren tareas de tiempo real). Pero, si se sale a la calle, uno puede entender que casi todos los sistemas informáticos son de tiempo real: un controlador de tráfico, un horno microondas, un cajero automático, cualquier sistema de control de un vehículo, un "parkímetro", un robot o cualquier control por computador, cualquier sistema empotrado y casi la totalidad de sistemas con microcontroladores, etc., es decir, **cualquier sistema con cierto grado de autonomía**, (que no esté directamente a las órdenes de un operador humano), **es automáticamente un STR**. Esto debe ser así, porque su autonomía obliga a que el sistema debe controlar más o menos síncronamente tanto los eventos como las acciones exteriores. Por ende y es un hecho primordial, ha de controlar si ocurre cualquier tipo de fallo, en cuyo caso el sistema debería tomar la decisión más adecuada para subsanarlo. En este sentido, la manera más clásica de determinar que está ocurriendo un error es definir contadores de tiempo ("timers") internos o externos, tal que si un determinado tiempo prefijado es superado se considere que está ocurriendo un error. Así, toda acción o tarea implica una sincronización con tal timer o con el resto de tareas, es decir un sistema de tiempo real.

A los sistemas antes mencionados hay que añadir **toda la gama de sistemas multimedia** que necesariamente son de tiempo real, porque dependen de la correcta transmisión de una señal de imagen o sonido: un teléfono móvil, un videojuego, una videoconferencia, el procesado de imágenes de un reproductor de vídeo, un televisor digital, etc.

Pero aún más, existen tareas de tiempo real en un ordenador de sobremesa: todos **los programas que son interactivos con el usuario** deben de ser de tiempo real, para que el usuario "vea" que el tiempo está corriendo de forma real; por ejemplo, un juego de matar marcianos no puede ser indeterminista con el movimiento de las balas, porque si una bala se detuviera en mitad de su recorrido porque otra tarea del sistema entrase a ejecutarse, se perdería la sensación de realidad y el usuario quedaría inmediatamente decepcionado. Por supuesto estos últimos sistemas no son de **tiempo real "duro"** porque si ocurre un retraso en alguna de las señales o movimientos, no se produce ninguna "desgracia" o catástrofe, pero hay

que hacer todo lo posible por que el sistema sea STR, o sea debe existir sincronización entre las tareas del sistema y debe intentarse, en la medida de lo posible, satisfacer unos plazos.

Por último no debe confundirse lo que el usuario "ve" de un sistema con lo que realmente está ejecutándose en él; por ejemplo, todos sabemos que la red Internet hoy en día no se "ve" como de tiempo real, porque el envío o recepción de cualquier paquete nadie sabe cuando llegará (¡tal vez en los fines de semana podría intentarse un STR!). Sin embargo muchos de las tareas y sistemas que envían la información requieren tiempo real, con objeto de sincronizar el envío y gestión de las colas de paquetes. Este es el caso del "router", o en general como antes mencionamos de cualquier sistema que no lleve ningún operador controlándolo.

En consecuencia, al finalizar la asignatura, el alumno debe:

- Conocer los conceptos fundamentales de los sistemas en tiempo real.
- Saber diseñar sistemas básicos en tiempo real utilizando los "núcleos" disponibles para tal fin.
- Saber aplicar estas técnicas al diseño de los sistemas multimedia.
- Saber diseñar sistemas distribuidos sencillos en tiempo real.

Metodología y Evaluación.

Dado el carácter eminentemente práctico de esta materia, y al ser una asignatura terminal de la Licenciatura se pretende formar al alumno con el contacto directo en el laboratorio con los sistemas de tiempo real. Por tanto los contenidos y evaluación van a contener de **teoría solamente la base necesaria** para que en el resto del curso se puedan llevar a cabo las prácticas de laboratorio. Éstas consistirán en una **serie inicial de diseños obligatorios y comunes** a todo el alumnado, con objeto de que se asienten los conocimientos en la implementación de los STR mientras el alumno es dirigido por los profesores. Finalmente se pretende que un grupo de alumnos desarrolle un **"miniproyecto" de elaboración propia** en el que los alumnos tomen decisiones propias de diseño e implementación (por supuesto siempre tuteladas por los profesores) para que se enfrenten ellos mismos a los problemas reales. Un objetivo secundario del mencionado "miniproyecto" es la formación de una base para la elaboración del proyecto fin de carrera (en el caso de que el alumno esté interesado en realizarlo con alguno de los profesores del departamento). En este sentido el departamento incluye el Grupo de Investigación de Robótica y Tecnología de Computadores Aplicada a la Rehabilitación, con más de 15 años de experiencia en este campo, un laboratorio con abundante material y robots y financiación propia en base a proyectos y convenios de investigación con entidades públicas y privadas[f1].

Temario de la asignatura Arquitectura de Sistemas de Tiempo Real.

. Tema 0. Presentación del curso.

1. Profesorado de la asignatura.
2. Metodología y actividades docentes.
3. Criterios y sistema de evaluación.
4. Programación de la asignatura.
5. Bibliografía recomendada.

. *Descripción y objetivos:*

En este tema se exponen las normas que rigen la docencia de la asignatura, y se informa de horario y lugar de tutorías, documentación y modo de acceso, temario, etc.

. Tema 1: Conceptos básicos.

1. Clasificación de los STR.
2. Características intrínsecas de un STR.
3. Parámetros influyentes en los STR.
4. "Benchmarks" de evaluación.
5. Técnicas de especificación básicas para STR.

6. Capas en el diseño de STR: Nivel de Arquitectura, Nivel de S.O. (núcleos o "kernels" de tiempo real), Nivel de Aplicaciones.

. ***Descripción y objetivos:***

En este tema se exponen los conceptos más básicos sobre STR. En concreto se establecen los conceptos de Sistema en Tiempo Real, STR estricto (o "duro"), STR blando, STR firme, tareas periódicas y aperiódicas, plazos, deslizamiento temporal, tiempo de respuesta, tiempo de respuesta normalizado, utilización de ruptura, etc. Mapas de estado generalizados (Harel), Redes de Petri generalizadas, etc.

Tema 2: Nivel de Sistema Operativo: Núcleos ("Kernels") de tiempo Real.

1. "Scheduling", concurrencia y sincronización de tareas.
2. Gestión de prioridades. Técnicas "Rate Monotonic" y sus variaciones.
3. Gestión de recursos.
4. Ejemplos de kernels reales. µC/OS y RTKernel32.
5. Kernels compatibles POSIX.

. ***Descripción y objetivos:***

Aunque sería más sistemático introducir el nivel de arquitectura previamente al de Sistema Operativo se ha optado por invertir ese orden dado el carácter eminentemente práctico de la asignatura. El orden propuesto permite comenzar las prácticas fácilmente en las primeras semanas del cuatrimestre.

En este tema se estudian las herramientas básicas para poder realizar implementaciones sencillas y eficientes de sistemas en tiempo real. En primer lugar se repasan algunos conceptos básicos de sistemas operativos como secuenciamiento de tareas con expulsión, semáforos (binarios, generales, de recurso, de evento), buzones, mensajes, etc.

Posteriormente se estudian las técnicas de asignación de prioridades que permiten garantizar el cumplimiento de plazos fundamentalmente las "rate monotonic" que asignan prioridades de forma estática en función del período. También se estudian las variaciones de estas técnicas que permiten tratar las tareas aperiódicas como los servidores aperiódicos y esporádicos y las soluciones para los problemas de inversión de prioridad asociados a los semáforos (fundamentalmente los protocolos de herencia de prioridades).

El siguiente apartado estudia los problemas asociados a la gestión de recursos. Se estudian fundamentalmente los bloqueos, las condiciones necesarias para que se den y las técnicas más básicas para evitarlos.

A continuación se estudia como los núcleos (o "Kernels") típicos implementan las primitivas explicadas anteriormente. Se estudian dos núcleos portables básicos. Por una parte el µC/OS del que existen versiones, de dominio público, para una amplia gama de microcontroladores y procesadores de propósito general (M68HC11, 8051, 80x86 modo real, M68000, Alpha, etc.) y por otra el RT-Kernel32, más sofisticado, pero limitado (de momento) a procesadores 80x86 en modo protegido 32 bits con modelo de memoria plano. Estos núcleos serán los empleados en las practicas de la asignatura.

Tema 3: Nivel de Aplicaciones.

1. Adaptación de una aplicación a un STR.
2. Lenguajes de tiempo real: librerías de los Kernels, lenguajes específicos (ADA 95).
3. Sistemas multimedia de tiempo real.
4. Control de procesos de tiempo real.

. ***Descripción y objetivos:***

En este tema se exponen las técnicas para implementar aplicaciones empleando los métodos descritos en los temas anteriores. Inicialmente se estudian las técnicas de implementación y depuración de aplicaciones mediante librerías para lenguajes convencionales (C++, Pascal). Posteriormente se repasan los conceptos básicos de concurrencia implementados en el ADA95 y la implementación concreta de éstos en la versión GNU.

Se estudian los requisitos básicos de las aplicaciones multimedia y la forma de satisfacerlos mediante técnicas TR. Se introducen así las soluciones a emplear en sistemas donde lo esencial es

satisfacer los requisitos TR en la inmensa mayoría de las ocasiones (sistemas TR firmes). Se aplican estas técnicas al diseño de aplicaciones interactivas con requisitos TR como los videojuegos.

En el último apartado se introducen las características del grupo de aplicaciones más clásico dentro del tiempo real: los sistemas de control. Se exponen los conceptos más básicos y se estudia como satisfacer las restricciones que imponen mediante las técnicas de programación en tiempo real.

Tema 4: Nivel de Arquitectura.

1. Sistemas empotrados básicos. Formas de Temporización. .
2. Problemática de los procesadores avanzados para el diseño de STR: procesadores segmentados, interrupciones, jerarquía de memoria.
3. Sistemas distribuidos de tiempo real: redes deterministas.

. ***Descripción y objetivos:***

En este tema se expone la influencia en la arquitectura del sistema debida a su carácter STR. Por una parte son necesarios elementos que permitan temporizar el sistema y detectar errores básicos en el mismo. Por otra existe una problemática ligada a las técnicas de optimización de prestaciones, esencialmente estadísticas, empleadas por los procesadores actuales. Técnicas como la implementación encadenada (o superencadenada/superescalar), el secuenciamiento dinámico, la ejecución especulativa, los cachés o la memoria virtual llevan a sistemas en los que la predicción del tiempo de ejecución es difícil. En el tema se hará un estudio básico de los métodos para permitir emplear, de forma correcta, estos procesadores en STR:

Para finalizar el tema se realizará un estudio de las redes locales que permiten establecer cotas máximas para los plazos de transmisión de mensajes. Estas redes son absolutamente necesarias en el caso de aplicaciones tiempo real distribuidas como pueden ser los sistemas de control de vehículos y muchos controladores industriales. La creciente importancia de las aplicaciones multimedia implicará una influencia de los protocolos usados en estas redes en otras de propósito general.

Tema 5: Fiabilidad de un STR.

1. Conceptos básicos de fiabilidad y seguridad.
2. Técnicas Básicas de Detección de fallos. Autocomprobación, Temporizadores de Guardia, Protección de recursos, Procesadores de Guardia.
3. Introducción a las Técnicas básicas de Tolerancia a fallos.

. ***Descripción y objetivos:***

En este tema se exponen los conceptos asociados a la fiabilidad y seguridad en los sistemas en tiempo real. En el caso de la mayoría de sistemas lo que se pretende es que sean seguros, es decir, que tengan una probabilidad muy baja de sufrir averías catastróficas. Para este fin suele ser suficiente asegurar la detección de estados que nos puedan llevar a estas situaciones y rutinas de emergencia que permitan actuar en consecuencia. Para la detección de los fallos se estudia la implementación de las soluciones más extendidas en los sistemas tiempo real: los temporizadores de guardia y la protección de recursos (zonas de memoria, dispositivos, etc.). También se estudian algunas soluciones más avanzadas como los procesadores de guardia que monitorizan la actividad del procesador o procesadores donde se ejecuta la aplicación. Como culminación del tema se introducen las técnicas a emplear en el caso en que sea necesario que el sistema continúe operativo incluso después de sufrir una avería.

Tema 6: Análisis cuantitativo.

1. Viabilidad.
2. Acotación del retraso y tiempo de ejecución de tareas.
3. Acotación de tareas aperiódicas.
4. Frecuencia de muestreo y accionamiento de señales.

. ***Descripción y objetivos:***

En este tema se exponen con mayor profundidad los conceptos introducidos en el tema 2 para garantizar de forma formal el cumplimiento de los plazos en cualquier sistema tiempo real. Se presta especial atención al caso en que la aplicación este distribuida entre diversos procesadores conectados

mediante una red o bus tiempo real. En el tema también se discuten de forma más formal que en los anteriores los requisitos impuestos por las características de las señales y estímulos externos con los que nuestro sistema deba interactuar.

Prácticas de Arquitectura de Sistemas de Tiempo Real.

La asistencia al laboratorio y la entrega de las memorias correspondientes a cada prácticas son requisitos indispensables para evaluar la asignatura.

Todas las prácticas incluyen una explicación previa (aprox. 30 minutos) de los conceptos necesarios y de la metodología a seguir. La duración completa, incluyendo la explicación previa, elaboración del software y realización de pruebas es de alrededor de cuatro a cinco horas, pudiéndose dividir en partes de dos horas si el horario del centro así lo aconseja.

Finalmente cada grupo de alumnos deberá desarrollar un "miniproyecto" de elaboración propia en el que los alumnos tomen decisiones propias de diseño e implementación (por supuesto siempre tuteladas por los profesores) para que se enfrenten ellos mismos a los problemas reales.

Se impartirán las siguientes prácticas a lo largo del cuatrimestre:

- Práctica 1. Desarrollo de una aplicación básica TR empleando librerías del nucleo RT-Kernel 32. La aplicación permitirá que las tareas existentes en un procesador puedan gestionar los mensajes y buzones en otro mediante una conexión serie entre ellos.
- Practica 2. Implementación de un simulador en tiempo real de un motor con entradas y salidas analógicas.
- Practica 3. Implementación de un controlador para motor. En una sesión la mitad de grupos realizarán la práctica 2 y la otra mitad la 3. Como prueba final se comprobará que el controlador realizado por un grupo es capaz de controlar el "motor" simulado por otro.
- Practica 4. Compresión/Descompresión de audio en tiempo real. Se dará a los alumnos un algoritmo de compresión/descompresión y sus requisitos temporales. Deberá implementarse empleando C++ y RT-Kernel 32 garantizando el cumplimiento de los tiempos.
- Miniproyecto: Se propondrá una lista de proyectos relacionados con el diseño de sistemas multimedia y sistemas de control. Los alumnos elegirán el que deseen y lo implementarán en las sesiones de prácticas restantes del cuatrimestre. Los alumnos podrán proponer sus propios miniproyectos si lo desean.

BIBLIOGRAFIA BASICA.

Burns, A., Wellings, A. "Real Time Systems and Programming Languages". Segunda Edición, Addison-Wesley, 1996.

Laplante, "Real Time Systems Design and Analysis". IEEE Press, 1993.

Klein et al. "A Practitioner Handbook for Real Time Analysis". Ed. Kluwer, 1993.

N. Nissanke. "Real Time Systems: An Introduction". Ed. Prentice Hall. 1997.

BIBLIOGRAFIA COMPLEMENTARIA.

Se utilizarán artículos de revistas científicas especializadas en la materia.