Escola Politècnica Superior
d'Enginyeria de Vilanova i la Geltrú
UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROJECTE FI DE CARRERA

TÍTOL: Diseño e Implementación de una Herramienta para la Planificación de Ejecutivos Cíclicos

AUTOR: José Yépez Castillo

TITULACIÓ: Enginyeria en automàtica i Electrònica Industrial

DIRECTOR: Pau Martí Colom

PONENT: Manel Velasco García

DEPARTAMENT: Enginyeria de Sistemes, Automàtica I Informàtica Industrial

DATA: 03 de Julio de 2007

TÍTOL: Diseño e Implementación de una Herramienta para la Planificación de Ejecutivos Cíclicos

COGNOMS: Yépez Castillo **NOM: José**

TITULACIÓ: Enginyeria en automàtica i Electrònica Industrial

ESPECIALITAT: **PLA:**

DIRECTOR: **Pau Martí Colom**

DEPARTAMENT: Enginyeria de Sistemes, Automàtica I Informàtica Industrial

QUALIFICACIÓ DEL PFC

TRIBUNAL

PRESIDENT	SECRETARI	VOCAL
Francisco Villasevil Marco	Francisco Ruiz Vegas	Enric Ferrer Bardem

DATA DE LECTURA: 09 de Julio de 2007

Aquest Projecte té en compte aspectes mediambientals: □ Sí □ No

PROJECTE FI DE CARRERA

RESUM (màxim 50 línies)

Uno de los planteamientos válidos para realizar la implementación de la gestión de tareas en sistemas de tiempo real periódicos lo constituye la utilización de planificadores cíclicos. Este método de planificación goza de un alto grado de determinismo, previsibilidad, fiabilidad y sencillez de implementación. Además, es un método bien conocido y ampliamente utilizado en entornos industriales. El principal inconveniente de este tipo de planificador es la falta de flexibilidad, ya que cualquier cambio en el conjunto de tareas o en sus características temporales obliga a rehacer el plan de ejecución. Para solventar estos inconvenientes es necesario, y muchas veces indispensable, contar con herramientas que ayuden a los diseñadores del sistema en la creación de la planificación cíclica de forma automática. En este trabajo se describe una herramienta denominada CICLIC, que sirve de soporte en el diseño o regeneración de planificaciones cíclicas. El mecanismo de diseño se basa en algoritmos de búsqueda exhaustiva que utilizan reglas heurísticas para optimizar el camino de búsqueda.

Paraules clau (màxim 10):

Planificación Cíclica	Sistemas de Tiempo Real	Simulación

Tabla de Contenido

CAPITULO 1

Introducción, Motivación y Objetivos

1.1. Introducción

Una de las áreas de aplicación de los computadores es la supervisión y el control de procesos industriales. En estos casos, el computador suele estar integrado en un sistema de ingeniería más complejo y su objetivo es gobernar el comportamiento de dichos sistemas.

Estos sistemas de control por computador están presentes en un gran número de sistemas de ingeniería y realizan labores de complejidad muy variable. Desde sencillos computadores para controlar el proceso de inyección de combustible en un motor de explosión hasta sofisticadas redes de computadores para controlar una planta de fabricación, una central nuclear o un avión.

Una característica que diferencia estos sistemas es la obligación de completar sus actividades en determinados plazos de tiempo, de otro modo el sistema controlado no funcionará correctamente.

Estos sistemas reciben la denominación de sistemas de tiempo real. El requisito fundamental de los sistemas de tiempo real es que su correcto funcionamiento depende tanto de proporcionar las salidas adecuadas como de hacerlo dentro del intervalo de tiempo requerido [28]. Este requisito conduce a uno de los problemas fundamentales en el diseño de sistemas de tiempo real, que es la planificación del tiempo de ejecución, de forma que las respuestas se produzcan dentro del plazo fijado.

Como consecuencia de este requisito, el desarrollo de estos sistemas es diferente al de los sistemas informáticos convencionales donde no existen estas restricciones temporales. Un tipo especial de sistemas de tiempo real son los sistemas de tiempo real crítico. En este tipo de

sistemas el incumplimiento de algún plazo de respuesta es intolerable, puesto que se pueden producir graves consecuencias en el sistema controlado.

Por otra parte, los sistemas de tiempo real actualmente juegan un rol muy importante en nuestra sociedad utilizándose en un gran número actividades complejas. Específicamente, los sistemas de planificación de tiempo real constituyen un área de vital importancia para todos los ingenieros de control. Actualmente casi todos los controladores son implementados de forma digital en un computador. Por ejemplo, un sistema de gestión de motor para coches depende cada vez más del control realimentado y cálculo de tiempo real para mejorar el rendimiento, reducir el consumo de combustible y minimizar la cantidad de contaminación que emite. Por lo tanto, son necesarias herramientas de ingeniería que permitan desarrollar sistemas de tiempo real más eficientes, fiables, económicos y de fácil mantenimiento.

Los sistemas de tiempo real se suelen estructurar en un conjunto de tareas concurrentes. El problema fundamental reside en planificar los recursos computacionales de forma que los resultados se produzcan en el plazo de tiempo previsto.

Existen dos aproximaciones para realizar la planificación, calcular la planificación off-line, esto es antes de poner en marcha el sistema, o realizar la planificación on-line, es decir en tiempo de ejecución. La primera aproximación se conoce como planificación estática y es el método clásicamente utilizado. La segunda aproximación exige la asignación de prioridades de forma que el planificador tenga un criterio para decidir a quién asigna los recursos en cada momento.

El planteamiento clásico de planificador de tiempo real off-line, lo constituye el planificador cíclico. Sus ventajas principales residen en un menor uso de los recursos computacionales para llevar a cabo la planificación, así como su determinismo y predecibilidad. Esto hace que sea el planificador más utilizado en entornos donde los recursos computacionales son limitados, como las aplicaciones espaciales [7] o donde el grado de fiabilidad exigido es muy alto, como los sistemas críticos respecto a la seguridad [18].

Sin embargo, su principal desventaja es que el diseño de los planes es muy laborioso, poco flexible y difícil de mantener, ya que si existe un cambio en los requisitos temporales de alguna de las tareas o se añade una nueva tarea de debe rehacer toda la planificación. Además, aunque puede alcanzar un alto grado de utilización del procesador existen muchos sistemas que no admiten una planificación cíclica. Esto se debe a que el método exige cierta uniformidad entre las características temporales de los procesos del sistema, de otro modo o no es posible la planificación o se hace muy compleja.

Las primeras contribuciones teóricas sobre planificación fueron propuestas hace alrededor de 30 años [20]. Esta teoría ha evolucionado para hacer frente a las diferentes exigencias de las aplicaciones [17, 30] y ha sido utilizada satisfactoriamente en muchos proyectos [24].

Esta teoría ayuda al diseñador del sistema a estudiar y predecir el comportamiento temporal de un conjunto de tareas de tiempo real a través de test y simulación de la planificación. El *test*

de planificabilidad establece si es factible llevar a cabo la planificación de un conjunto de tareas mediante un método especificó sin necesidad de determinar dicha planificación. La *simulación de la planificación* consiste en determinar una planificación en un intervalo de tiempo, y observar las propiedades temporales en la planificación calculada.

Sin embargo, pocas herramientas han sido desarrolladas para ayudar a los diseñadores del sistema a generar planificadores de forma automática. Algunas de estas herramientas son brevemente descritas:

Rapid-RMA [26], es una de las herramientas de análisis más completas para estudiar la planificabilidad bajo condiciones de peor caso. Soporta la mayoría de los algoritmos de planificación clásicos y test de planificabilidad. No es de libre acceso.

TimeSys [25] es un entorno para el modelado, análisis y simulación del comportamiento temporal de sistema de tiempo real. Esta herramienta provee la mayoría de los algoritmos de planificación clásicos.

YASA [5] y la herramienta de la Universidad Libre de Bruxelles [9] están enfocadas únicamente a la simulación de la planificación. No ofrecen un test de planificabilidad.

MAST [14] es una herramienta de trabajo ADA y proporciona algunas pruebas de factibilidad y servicios de simulación para planificadores con prioridades fijas.

CHEDDAR [8] provee un marco de trabajo de libre acceso y flexible, el cual implementa la mayoría de los métodos de la teoría de planificación clásica. Sin embargo no trata planificadores cíclicos.

TRUETIME [27] es un simulador basado en Matlab/Simulink para sistemas de control de tiempo real. Esta herramienta provee la mayoría de los algoritmos de planificación clásicos.

La gran mayoría de estas herramientas no permiten la determinación de planes de ejecución de forma automática para planificación cíclica, excepto Rapid-RMA. Ésto es debido en gran parte a la naturaleza del problema a resolver y la carencia de algoritmos que generen automáticamente planes de ejecución en un tiempo razonable.

1.2. Motivación

El presente proyecto nace de un proyecto realizado con la empresa Lear Corporation empresa fabricante de componentes para el sector de la automoción y el grupo de investigación consolidado de la UPC: *GRINS – Grup de Recerca en Robòtica Intel·ligent i Sistemas* y específicamente al grupo de Sistemas Distribuidos de Control del Departamento de ESAII cuyo responsable científico es el Profesor Catedrático Josep M. Fuertes i Armengol.

Uno de los objetivos de este proyecto es el de brindar herramientas que permitan a priori determinar los niveles de carga de cada uno de los procesadores, de tal forma que se pueda evaluar objetivamente hasta que punto es posible descargar el recurso procesador y hasta que punto es posible sobrecargar el recurso procesador para mejora el rendimiento de las tareas.

Después de llevar a cabo un análisis de los sistemas de planificación implementados en los procesadores de Lear y de profundizar en los fundamentos teóricos de los planificadores o ejecutivos cíclicos, se detecto la necesidad de una herramienta que sirva de apoyo a los diseñadores de sistemas de tiempo real en la construcción y mantenimiento de planificadores cíclicos. Así, el desarrollo e implementación de la herramienta que se describe en este documento está motivado por la carencia de herramientas flexibles para construir planificadores cíclicos.

Por lo tanto, este trabajo describe el diseño y la implementación de una herramienta llamada CICLIC que permite la construcción y mantenimiento de planificadores cíclicos. El mecanismo de diseño de CICLIC se basa en algoritmos de búsqueda exhaustiva que utilizan reglas heurísticas para optimizar el camino de búsqueda. CICLIC se caracteriza por ser una herramienta portable, fiable y fácil de utilizar.

1.3. Objetivos

El objetivo principal de este proyecto es diseñar e implementar una herramienta que sirva de apoyo, en el análisis, diseño, desarrollo y mantenimiento de planificadores cíclicos de sistemas de tiempo real con un procesador.

Para llevar a cabo este objetivo general se desprenden los siguientes objetivos específicos:

1. Estudiar en profundidad el método de planificación cíclico.
2. Desarrollar e implementar algoritmos para la generación de planificadores cíclicos de forma automática y en un tiempo razonable.
3. Aplicar la herramienta obtenida a un caso real.

1.4. Metodología

Para cumplir con los objetivos planteados se plantea la siguiente metodología de trabajo:

1. *Revisión bibliográfica y documental:* En esta etapa se realizó una búsqueda bibliográfica relacionada con los sistemas de tiempo real, con el fin de establecer un marco conceptual.

2. *Análisis de los diferentes métodos de planificación:* Esta actividad consiste en investigar acerca de las diferentes técnicas de planificación con especial énfasis en el método de planificación de ejecutivos cíclicos y herramientas existentes que permiten la determinación de ejecutivos cíclicos.

3. *Diseño e implementación de la herramienta*: En esta etapa se determinan los diferentes algoritmos utilizados para la determinación de ejecutivos cíclicos. Esta etapa contempla la implementación de estos algoritmos sobre un lenguaje de programación.

4. *Aplicación práctica:* Este paso consiste en el uso de la herramienta desarrollada sobre un problema real.

5. *Presentación y discusión de resultados*: Su objetivo fue mostrar los resultados de la investigación con el fin de realizar posteriormente su análisis, discusión e interpretación.

6. *Elaboración de conclusiones:* Esta etapa consistió en presentar las conclusiones que se desprendieron del análisis de los resultados.

1.5. Contenido

Este documento esta organizado en seis capítulos cuyo contenido son descritos brevemente a continuación.

Capitulo 1. *Introducción, Motivación y objetivos*. En este capitulo se describe el motivo del proyecto, los objetivos que se persiguen, metodología de trabajo y la estructura de este documento.

Capitulo 2. *Sistemas de Tiempo Real*. En este capitulo se describen los conceptos básicos de los sistemas de tiempo real, Se describe en que consiste la planificación de tareas y se describen los principales algoritmos de planificación.

Capitulo 3. *Ejecutivos cíclicos.* En este capitulo se presenta el método de planificación cíclica enunciado por Baker [2], complejidad del problema y se describen los algoritmos de búsqueda utilizados para generar la planificación cíclica de un conjunto de tareas.

Capitulo 4. *Herramienta para generar ejecutivos cíclicos*. En este capitulo se describe en detalle la herramienta CICLIC para la generación de ejecutivos cíclicos.

Capitulo 5. *Resultados experimentales*. En este capitulo se describe en detalle el conjunto de tareas en estudio, se presentan los resultados obtenidos por simulación en PC

utilizando la herramienta CICLIC y los resultados obtenidos por emulación en tarjeta electrónica.

Capitulo 6. *Estudio económico*. En este capitulo se presenta un estudio económico del proyecto.

Capitulo 7. *Conclusiones*. En este capitulo se resumen las conclusiones obtenidas en la realización de este proyecto y se describen posibles trabajos futuros.

CAPITULO 2

Sistemas de Tiempo Real

2.1. Introducción

Los sistemas de tiempo real son sistemas de computación que interaccionan repetidamente con su entorno físico y responde a los estímulos que recibe del mismo dentro de un plazo determinado. Para que el funcionamiento sea correcto no basta con que las acciones sean correctas, sino que tiene que ejecutarse dentro del intervalo de tiempo especificado.

Típicamente un sistema de tiempo real consiste (Figura 2.1) de subsistemas que controlan (el computador de control) y subsistemas controlados (el ambiente físico) y está caracterizado por la necesidad de restricciones de tiempo. Las interacciones entre los dos subsistemas están descritas por tres operaciones: muestreo, procesamiento y respuesta. Los subsistemas de computación continuamente procesan los datos muestreados del ambiente físico y producen una apropiada respuesta que es enviada al ambiente físico. Las tres operaciones deben ser realizadas dentro de tiempos específicos; esto es lo que constituye las restricciones de tiempo. Si una reacción ocurre demasiado tarde podría ser peligrosa para el sistema. Hoy en día, la computación en tiempo real juega un papel importante en nuestra sociedad, puesto que un gran número de sistemas complejos depende en parte o completamente del control por computadora. Algunos ejemplos de aplicaciones que requieren computación de tiempo real son:

- Control de automóviles
- Control de procesos de producción complejos
- Control de plantas nucleares y químicas
- Control de tráfico aéreo
- Sistemas multimedia
- Automatización industrial
- Sistemas de telecomunicación

- Sistemas de misiones espaciales
- Robótica
- Sistemas militares

La mayoría de los sistemas de tiempo real son componentes de otros sistemas, en los que realizan funciones de control, en este caso se habla de sistemas empotrados o sistemas embebidos (embedded systems).

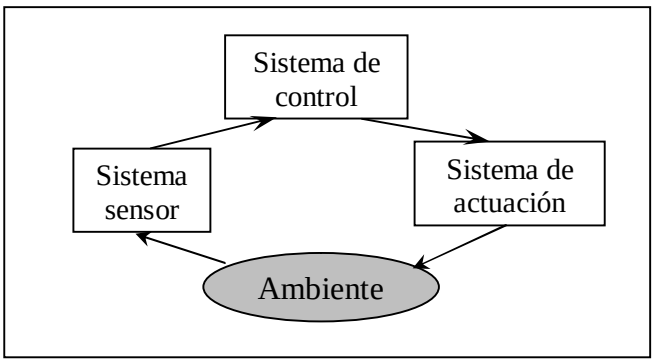


Figura 2.1. Entorno de un sistema de tiempo real

2.1.1. Características deseables para los sistemas de tiempo real

Las características deseables para un sistema de tiempo real son:

- **Puntualidad.** Los resultados deben ser correctos tanto en su valor como en el tiempo en el cual se producen.
- **Alta utilización de recursos.** Poder alcanzar utilizaciones altas de los recursos, en especial de la CPU, a la vez que se garantiza el cumplimiento de los plazos.
- **Diseñados para soportar picos de carga.** Los sistemas de tiempo real deben ser diseñados para que soporten picos de carga, es decir los sistemas de tiempo real no deben colapsar cuando están sujetos a condiciones de pico de carga y deben poder manejar todos los escenarios anticipadamente.
- **Fiabilidad.** Los sistemas de tiempo real deben proporcionar el servicio especificado.
- **Previsibilidad.** El sistema debe garantizar un mínimo nivel de rendimiento y ser capaz de predecir las consecuencias de alguna decisión de planificación.

  El sistema debe ser capaz de predecir la evolución de las tareas y garantizar anticipadamente que todas las restricciones de tiempo crítico sean cumplidas. Sin embargo esto depende de varios factores, los cuales tienen que ver con las características de la arquitectura de hardware (DMA, memoria cache, etc.), los mecanismos y políticas

adoptadas por el kernel (semáforos, llamadas del sistema, interrupciones, etc.) y el lenguaje de programación utilizado en la implementación.

- **Tolerante a fallos.** Simples fallos en el hardware o en el software no deben causar que el sistema colapse. Así, los componentes críticos de los sistemas de tiempo real tienen que ser diseñados para tolerar fallos.
- Producir un bajo aumento de la sobrecarga (overhead), es decir, del trabajo extra que la CPU dedica para realizar la planificación.
- **Mantenibilidad.** La arquitectura del sistema de tiempo real debe ser modular para asegurar que posibles modificaciones del sistema sean realizadas fácilmente.

2.1.2. Limitaciones de los sistemas de tiempo real actuales

La mayoría de los sistemas de computación de tiempo real usados para soportar aplicaciones de control están basados en kernels [1, 15], los cuales son versiones modificadas de sistemas operativos de tiempo compartido. Como consecuencia ellos tienen las mismas características básicas encontradas en los sistemas de tiempo compartido, tales como multitarea, programación concurrente, planificación basada en prioridades, kernel pequeño, uso de semáforos como mecanismos para comunicación de procesos y sincronización, etc.; la mayoría de estas características introducen retardos no acotados en los tiempos de ejecución de las tareas, lo cual afecta la previsibilidad del sistema.

2.1.3. Clasificación de los sistemas de tiempo real

Una característica común de los sistemas de tiempo real es que el computador está conectado al ambiente con el cual esta trabajando por una amplia gama de dispositivos a través de los cuales envía y recibe una variedad de señales. Los procesos del ambiente operan en su propia escala de tiempo y el computador se dice que opera en tiempo real si las acciones llevadas acabo en el computador se relacionan con las escalas de tiempo de los procesos externos o del ambiente.

La sincronización entre los procesos externos y las tareas llevadas a cabo por el computador pueden ser definidas en términos:

- Del paso del tiempo, o la hora actual del día, en este caso el sistema se dice que el sistema esta *dirigido por tiempo* (time-triggered systems) y las operaciones son cumplidas de acuerdo a una planificación en el tiempo.
- Definido en términos de eventos, por ejemplo el cierre de un interruptor, en el cual se dice que el sistema esta *dirigido por eventos* (event-triggered systems).

2.1.3.1. Tareas basadas en tiempo (periódicas)

Una característica importante de una planta que opera en tiempo real es la constante de tiempo de la planta (una constante de tiempo es una medida del tiempo tomado por una planta para responder a un cambio en la entrada). La constante de tiempo puede ser medida en horas para algunos procesos químicos o en milisegundos para un sistema de avión. El control realimentado requiere una tasa de muestreo la cual depende de la constante de tiempo de los procesos a ser controlados. La constante de tiempo del proceso mas corta, es la tasa de muestreo mas rápida que se requiere. El computador que es usado para controlar la planta debe por lo tanto ser sincronizado a tiempo real y debe ser capaz de cumplir todas las operaciones requeridas - medición, control y actuación – dentro de cada intervalo de muestreo. La sincronización es usualmente obtenida, añadiendo un reloj al sistema del computador, normalmente referido como un reloj de tiempo.

Las tareas basadas en tiempo típicamente son referidas como tareas cíclicas o periódicas donde el término puede implicar que la tarea se ejecute una vez por período *T*, o esta se ejecute en exactamente *T* intervalos.

2.1.3.2. Tareas basadas en eventos (aperiódicas)

Existen muchos sistemas donde las acciones no son realizadas en un tiempo particular o en un intervalo de tiempo pero si en respuesta a algún evento. Por ejemplo: el apagado de una bomba o el cierre de una válvula cuando el nivel de líquido en un tanque alcance un valor determinado, etc. Sistemas basados en eventos son también usados extensamente para indicar condiciones de alarma e iniciar acciones de emergencia. Las especificaciones de sistemas basados en eventos usualmente incluyen un requerimiento que el sistema debe responder dentro de un máximo tiempo dado para un evento particular.

Los sistemas basados en eventos normalmente emplean interrupciones para informar al sistema de computación que acción es requerida. Los eventos no ocurren en intervalos deterministicos y las tareas basadas en eventos son frecuentemente referidos como una tarea aperiódica o esporádica. Tales tareas pueden tener plazos expresados en términos de tiempo de iniciación o tiempo de finalización (o ambos). Por ejemplo, una tarea puede ser requerida para que comience su ejecución dentro de 5 segundos de la ocurrencia de un evento, o alternativamente esta debe producir una salida a los 5 segundos del evento.

2.2. Conceptos básicos

Una de las más importantes entidades de software tratada por un sistema operativo es el proceso. Un *proceso* es un Cálculo que es realizado por el CPU de forma secuencial. Una *tarea* es una ejecución secuencial de código que no se suspende así misma durante su ejecución, un

proceso es una actividad computacional más compleja, que puede estar compuesta por varias tareas.

Cuando un procesador tiene que ejecutar un conjunto de tareas concurrentes – esto es, que pueden solaparse en el tiempo – las tareas tienen que ser asignadas a la CPU de acuerdo a un criterio predefinido llamado *política de planificación*. El conjunto de reglas que en algún tiempo determina el orden en el cual las tareas son ejecutadas es llamado un *algoritmo de planificación*. Un *método de análisis* que permite comprobar el comportamiento temporal del sistema acompaña al algoritmo de planificación, en general este estudia el peor comportamiento posible. La operación específica de reparto de CPU a una tarea seleccionada por el algoritmo de planificación es referida como *despacho*.

Una tarea que puede potencialmente ejecutarse sobre el procesador, independientemente de su disponibilidad, es llamada una *tarea preparada o lista*, cuando la tarea se esta ejecutando es llamada una *tarea en ejecución*. Todas las tareas listas que esperan por el procesador son guardadas en una cola, llamada *cola de preparados*. Los sistemas operativos que manejan diferentes tipos de tareas, puede tener mas de una cola.

En muchos sistemas operativos que permiten activación de tareas dinámicas, las tareas en ejecución pueden ser interrumpidas en algún momento, tal que una tarea más importante que llegue al sistema pueda inmediatamente tomar el procesador y no necesite esperar en la cola de preparados. En este caso la tarea en ejecución es interrumpida e insertada en la cola de preparados, mientras que la CPU es asignada a la tarea lista más importante que acaba de llegar. La operación de suspensión de tareas que están ejecutándose y de inserción de esta en la cola de preparados es llamada *desalojo* [23].

Un *planificador con desalojo* es un planificador en el cual tareas ejecutándose pueden ser arbitrariamente interrumpidas en algún instante de tiempo, para asignar la CPU a otra tarea de acuerdo a una política de planificación predefinida.

Cuando una tarea esta ejecutándose, esta puede ser *suspendida* si la tarea requiere algún recurso el cual no esta disponible o porque la misma esta esperando por alguna señal del sistema. Todas las tareas suspendidas sobre un mismo recurso son almacenadas dentro de una cola asociada al recurso. La Figura 2.2, ilustra los estado y el proceso de planificación de las tareas.

Un planificador se dice que es *factible* si todas las tareas pueden ser completadas de acuerdo a un conjunto de restricciones específicas. Un conjunto de tareas se dice que es *planificable* si existe al menos un algoritmo que pueda producir un planificador factible.

2.2.1. Tipos de restricciones en las tareas

Las restricciones típicas que pueden especificarse en tareas de tiempo real son restricciones de tiempo, restricciones de exclusión mutua sobre recursos compartidos y relaciones de precedencia.

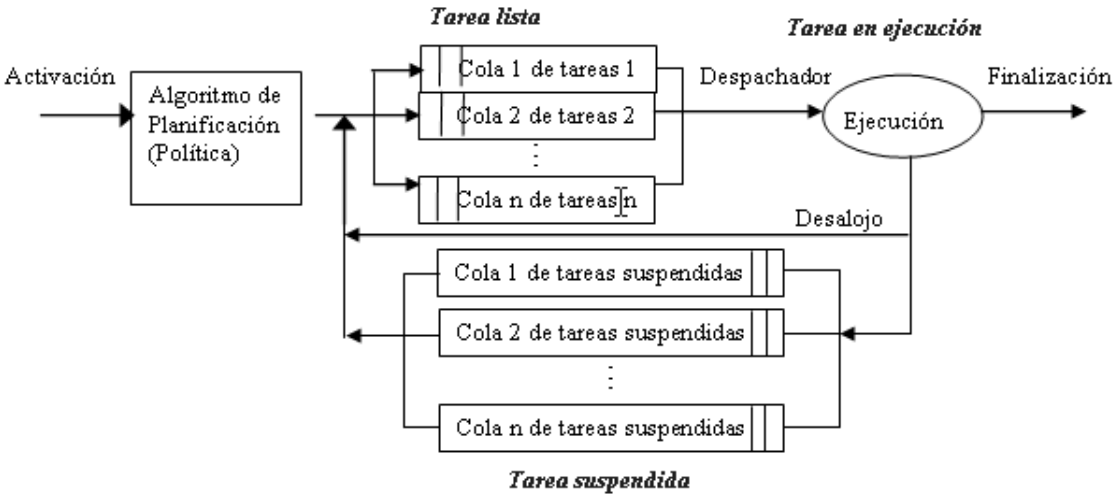


Figura 2.2. Estado y proceso de planificación de las tareas

2.2.1.1. Restricciones de tiempo

Los sistemas de tiempo real se caracterizan por tener actividades computacionales con estrictas restricciones que deben ser cumplidas en orden para alcanzar el comportamiento deseado. Una típica restricción tiempo sobre una tarea es el *plazo*, el cual representa el tiempo antes del cual un proceso debe completar su ejecución. Dependiendo de las consecuencias de la pérdida de un plazo, las tareas de tiempo real son clasificadas en:

- **Críticas.** Una tarea se dice que es crítica si la pérdida de su plazo puede causar catastróficas consecuencias sobre el sistema. *Sistemas de tiempo real críticos* son sistemas capaces de manejar tareas de tiempo real críticas. Típicamente los sistemas de tiempo real críticos envuelven vidas humanas, tal como sistemas de control de vuelo, sistemas de control de procesos químicos, sistemas de monitoreo de pacientes, o ambientes críticos en tiempo tal como sistemas de control de robots y sistemas de conmutación telefónica.
- **Acríticas.** Una tarea se dice que es acrítica si la pérdida de su plazo decrementa el rendimiento del sistema pero no pone en peligro su correcto comportamiento. *Sistemas de tiempo real acríticos* son sistemas capaces de manejar tareas de tiempo real acríticas. Ejemplo de sistemas de tiempo real acríticos son sistemas de adquisición de datos remotos, los sistemas de reservación de boletos aéreos, maquinas contadoras automáticas.

Los parámetros que en general caracterizan una tarea de tiempo real J_i se muestran en la Tabla 2.1.

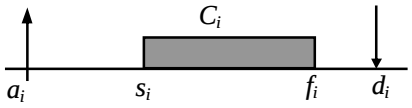
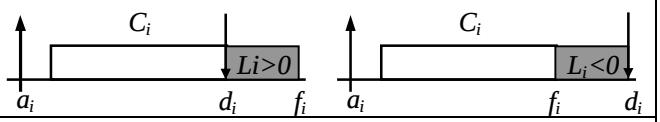
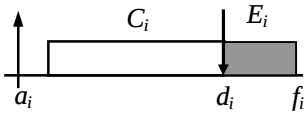
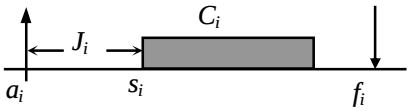
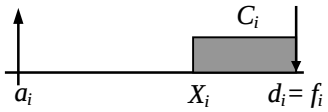
Parámetro		Descripción
Tiempo de llegada	a_i	Tiempo en el cual la tarea esta lista para su ejecución, también es referido como el tiempo de liberación o tiempo de solicitud indicado por r_i.
Tiempo de Cálculo o computo	C_i	Tiempo necesario de procesador para ejecutar la tarea sin interrupción.
Plazo	d_i	Tiempo antes del cual una tarea deberá ser completa.
Tiempo de comienzo	s_i	Tiempo en el cual la tarea comienza su ejecución.
Tiempo de finalización	f_i	Tiempo en el cual la tarea finaliza su ejecución.
Tiempo de respuesta máximo	R_i	Tiempo de respuesta máximo de una tarea o tiempo de respuesta de la tarea en el peor caso.
		
Criticidad	K_i	Parámetro relacionado a las consecuencias de perder el plazo-típicamente puede ser crítica o acrítica.
Prioridad	P_i	Representa la importancia relativa de una tarea con respecto a otra en el sistema.
Retardo	L_i	Representa el retardo de una tarea completada con respecto a su plazo. $L_i = f_i - d_i$ 
Tardanza o Tiempo excedente	E_i	Tiempo que una tarea permanece activa después de su plazo. $E_i = \max(0, L_i)$ 
Variación (Jitter)	J_i	Tiempo que una tarea gasta desde su liberación hasta su tiempo de comienzo. $J_i = a_i - s_i$ 
Latencia o tiempo de holgura	X_i	Es el máximo tiempo que la activación de una tarea puede ser retardada tal que pueda completarse dentro de su plazo. $X_i = d_i - a_i - C_i$ 

Tabla 2.1. Parámetros que describen una tarea de tiempo real J_i.

A cada tarea J_i le es asignada una prioridad P_i que indica la importancia o urgencia que tiene J_i con respecto a las otras tareas en el sistema. Las prioridades pueden ser asignadas a las tareas estáticamente o dinámicamente. Si en un tiempo *t*, $p_a > p_b$ significa que la ejecución de J_a es mas importante que la de J_b; así, J_b puede ser retardada a favor de J_a.

Otra característica importante que puede ser especificada sobre una tarea de tiempo real se refiere a la regularidad de su activación. En particular las tareas pueden definirse en:

- **Tareas periódicas** consisten de una secuencia infinita de actividades idénticas, llamadas instancias que son activadas regularmente a una razón constante. La activación de la primera instancia es llamada fase. Si ϕ_i es la fase de la tarea periódica τ_i, el tiempo de activación de la k-esima instancia esta dada por $\phi_i + (k\text{-}1)T_i$, donde T_i es el período de la tarea. La figura 2.3, muestra un ejemplo de una secuencia de tareas periódicas.

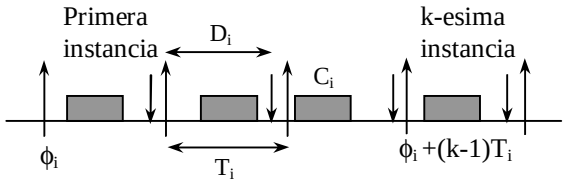


Figura 2.3. Secuencia de tareas periódicas

- **Tareas aperiódicas** consisten de una secuencia infinita de instancias cuyas activaciones no son regulares. La figura 2.4, muestra un ejemplo de una secuencia de tareas periódicas.

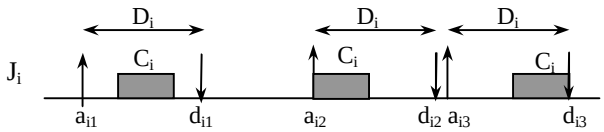


Figura 2.4. Secuencia de tareas aperiódicas

2.2.2. Restricciones de precedencia

En ciertas aplicaciones, las actividades computacionales no pueden ser ejecutadas en orden arbitrario y debe respetarse algunas relaciones de precedencia. Usualmente estas restricciones son descritas a través de gráficos acíclicos, donde las tareas están representadas por nodos y las relaciones de precedencia indicadas por flechas. Un gráfico de precedencia induce a un orden parcial sobre el conjunto de tareas.

2.2.3. Restricciones de recursos

Un recurso es una estructura de software que puede ser usada por un proceso para avanzar en su ejecución. Un recurso puede ser:

- Una estructura de datos
- Un conjunto de variables
- Un área de memoria principal
- Un archivo
- Una pieza de programa
- Un conjunto de registros
- Un dispositivo periférico

2.2.3.1. Tipos de recursos

- **Recurso privado**: es un recurso dedicado a un proceso en particular.
- **Recurso compartido**: es un recurso que puede ser usado por varias tareas.
- **Recurso exclusivo**: es un recurso compartido que no permite acceso simultáneo manteniendo mutua exclusión entre las tareas que compiten por el recurso, de manera de mantener la consistencia de los datos. Una pieza de código que es ejecutado bajo restricciones de exclusión mutua es llamada sección crítica.

Una tarea que necesite entrar a una sección crítica deberá esperar mientras que otra tarea mantenga el recurso. Una tarea esperando por un recurso exclusivo se dice estar *bloqueada* por el recurso, de otra manera este procede a entrar a la sección crítica y retiene el recurso. Cuando la tarea abandona la sección crítica, el recurso asociado con la sección crítica es liberado y este puede ser asignado a otra tarea que espera por el mismo.

Para asegurar el acceso secuencial a recursos exclusivos, el sistema operativo usualmente provee un mecanismo de sincronización, tal como los semáforos [11, 21, 22], los cuales pueden ser usados por las tareas para construir secciones críticas.

Todas las tareas bloqueadas sobre un mismo recurso son almacenadas dentro de una cola asociada con el semáforo que protege al recurso (Figura 2.5). Cuando una tarea libera el estado de espera, este no va al estado de ejecución inmediatamente, pasa al estado listo, tal que el algoritmo de planificación asigne al CPU la tarea con más alta prioridad.

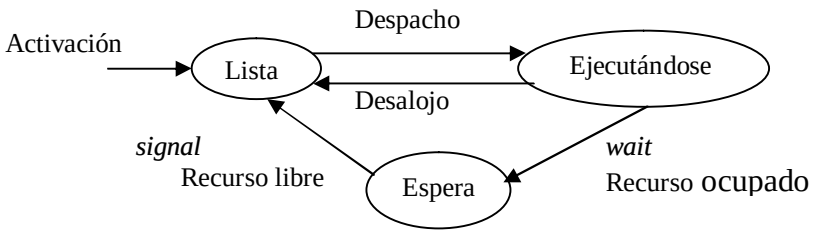


Figura 2.5. Bloqueo de tareas por restricciones de recursos

2.2.4. Problema de planificación

Dado un conjunto de *n* tareas $J = \{J_1, J_2, \mathbf{K}\, J_n\}$, un conjunto de *m* procesadores $P = \{P_1, P_2, \mathbf{K}, P_m\}$ y un conjunto de *s* tipos de recursos $R = \{R_1, R_2, \mathbf{K}, R_s\}$, el problema de planificación se puede definir como asignar procesadores desde *P* y recursos desde *R* a tareas desde *J* en orden para completar todas las tareas bajo las restricciones impuestas [4]. Las relaciones de precedencia entre tareas pueden ser especificadas a través de un diagrama acíclico, y las restricciones de tiempo están asociadas a cada tarea. Este problema en su forma general es NP-completo [13] y por lo tanto es intratable computacionalmente.

Para reducir la complejidad de la construcción de planificadores factibles, un conjunto de suposiciones son hechas tales como utilizar un solo procesador lo cual simplifica la arquitectura computacional, utilizar prioridades fijas, utilizar activación simultánea de tareas, etc. Las suposiciones hechas sobre el sistema o sobre el conjunto de tareas son típicamente usadas para clasificar los diferentes algoritmos de planificación propuestos en la literatura.

2.2.5. Clasificación de los algoritmos de planificación

La siguiente tabla muestra las principales clases de algoritmos de planificación entre la gran variedad de algoritmos propuestos para planificar tareas de tiempo real.

Algoritmo	Características
Con desalojo	Las tareas en ejecución pueden ser interrumpidas en algún instante de tiempo para asignarle al procesador otra tarea activa de acuerdo a una política de planificación predefinida.
Sin desalojo	Una vez que una tarea es iniciada, es ejecutada por el procesador hasta su finalización.
Estático	Son aquellos algoritmos en los cuales las decisiones están basadas sobre parámetros fijos, asignados a las tareas antes de su activación.
Dinámico	Son aquellos algoritmos en los cuales las decisiones están basadas sobre parámetros dinámicos que pueden cambiar durante la evolución del sistema.
Fuera de línea	Un algoritmo de planificación se dice que es usado fuera de línea si este es ejecutado sobre el conjunto entero de tareas antes de la activación de la tarea. El planificador generado es almacenado en una tabla y ejecutado después por un despachador.
En línea	Se dice que un algoritmo planificador es usado en línea si las decisiones de planificación son tomadas en tiempo de ejecución cada vez que una nueva tarea entra al sistema o cuando una tarea que este ejecutándose termine.
Suficientes en recursos	Los recursos del sistema son suficientes para garantizar el comportamiento temporal con máxima carga y posibles fallos.
Insuficientes en recursos	Uso razonable recursos en el sistema desde un punto de vista económico, solo sirve para sistemas acríticos.
Sistemas con respuesta garantizada	Comportamiento temporal garantizado analíticamente. La carga máxima y los posibles fallos deben ser caracterizados con precisión.
Sistemas que	Comportamiento temporal de tipo “lo mejor que se pueda”. No se hace una

Algoritmo	Características
hacen lo que pueden	caracterización precisa de carga y fallos, solo sirve para sistemas acríticos.
Optimo	Un algoritmo es óptimo si este minimiza alguna función de costo definida sobre el conjunto de tareas.
Heurístico	Un algoritmo es heurístico si este tiene la tendencia a encontrar un planificador óptimo pero el mismo no se puede garantizar.

Tabla 2.2. Clasificación de los algoritmos de planificación.

2.2.6. Medidas para la evaluación del rendimiento

El rendimiento de los algoritmos de planificación es típicamente evaluado a través de una función de costo definida sobre el conjunto de tareas. Algunas de las funciones de costo más comúnmente usadas [3] son descritas en la tabla 2.3.

Función de costo	Descripción matemática
Tiempo de respuesta promedio	$\bar{t}_r = \frac{1}{n}\sum_{i=1}^{n}(f_i - a_i)$
Tiempo total de completación	$t_c = \max_i(f_i) - \min_i(f_i)$
Suma con peso de los tiempos completados	$t_w = \sum_{i=1}^{n} w_i f_i$
Máximo retardo	$L_{max} = \max_i(f_i - d_i)$
Máximo número de tareas pérdidas	$N_{tp} = \sum_{i=1}^{n} per(f_i)$ donde $per(f_i) = \begin{cases} 0 & \text{si} \quad f_i \le d_i \\ 1 & \text{en otro caso} \end{cases}$

Tabla 2.3. Funciones de costo.

Las métricas adoptadas en los algoritmos de planificación tienen fuerte implicaciones sobre el rendimiento del sistema de tiempo real [29] y este debe ser cuidadosamente seleccionado de acuerdo a las especificaciones de la aplicación a desarrollar.

2.3. Planificación de tareas

Cada algoritmo presenta una solución para un problema de planificación particular, el cual es expresado a través de un conjunto de suposiciones sobre el conjunto de tareas y por el criterio de optimalidad que es usado por el planificador.

Existen principalmente dos paradigmas de algoritmos de planificación: planificadores cíclicos y planificadores basados en prioridad (Figura 2.6).

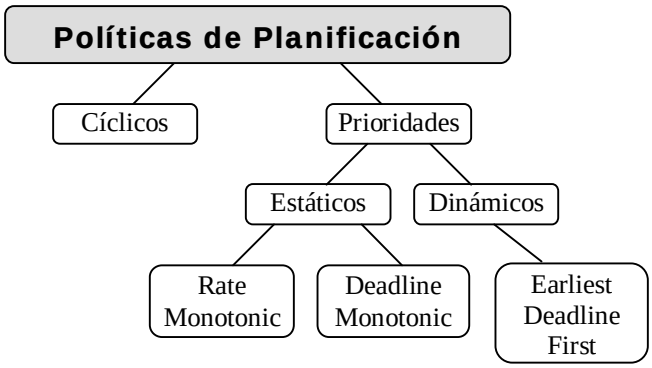


Figura 2.6. Políticas de planificación

El primero, el que más se ha utilizado tradicionalmente, se conoce por el nombre de planificador cíclico y consiste en construir en la fase de diseño del sistema una tabla con todas las acciones que tenía que llevar a cabo el planificador durante la ejecución. Durante la ejecución, el planificador únicamente consulta la tabla y hace que el procesador ejecute las tareas siempre en el mismo orden.

El principal problema que presentan es la poca flexibilidad a la hora de modificar alguno de los parámetros de las tareas, pues ello conlleva la re-elaboración de todo el plan. Aún hoy este tipo de planificación se utiliza en la industria. El capitulo 3 describe mas en detalle este tipo de planificación.

Otro gran grupo de planificadores son los basados en prioridades. Los cuales se dividen en prioridades estáticas y prioridades dinámicas. Nuestro estudio se centrará principalmente en los planificadores estáticos. En los planificadores estáticos, durante la fase de diseño a cada tarea se le asigna una prioridad. Después, durante la ejecución, el algoritmo de planificación simplemente tiene que ordenar la ejecución de las tareas en función de la prioridad asignada.

2.3.1. Factor de utilización

Para un conjunto de n tareas periódicas, el factor de utilización del procesador U es la fracción de tiempo de procesador gastada en la ejecución del conjunto de tareas [20]. Puesto que

C_i/T_i es la fracción de tiempo de procesador gastada en la ejecución de la tarea τ_i, el factor de utilización para un conjunto de *n* tareas esta dado por

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

Este factor provee una medida de la carga computacional sobre el CPU debido al conjunto de tareas periódicas, por lo tanto existe un máximo valor de U bajo el cual el conjunto de tareas es planificable y por arriba del cual no es planificable. Tales límites dependen del conjunto de tareas y el algoritmo usado para la planificación.

2.3.2. Planificación con prioridades fijas

Los dos principales algoritmos de asignación de prioridades fijas son:

- Dar más prioridad a la tarea más frecuente o *Rate Monotonic* (RM). Cuanto menor es el periodo de una tarea, mayor es su prioridad.
- Dar más prioridad a la tarea más urgente o *Deadline Monotonic* (DM). Cuanto menor es el plazo de una tarea, mayor es su prioridad.

En cualquiera de los dos casos, la prioridad se mantiene fija durante la ejecución del sistema.

2.3.2.1. Rate Monotonic

El algoritmo de planificación Rate Monotonic asigna prioridades a las tareas de acuerdo a su tasa de solicitud, es decir tareas con periodos cortos tendrán una alta prioridad (esto es, como una función monótona de la tasa de solicitud). Puesto que los periodos son constantes RM es un asignador de prioridades fijo, pero una tarea que llega con un periodo mas corto puede desalojar y adelantar una tarea que este ejecutándose.

Liu y Layland [20] demostraron que RM es óptimo entre todas las asignaciones con prioridad fija, y además derivaron un mínimo límite superior para el factor de utilización del procesador para un conjunto de n tareas periódicas.

Test de garantía (factor de utilización)

Un conjunto de *n* tareas será planificable bajo el Rate-Monotonic si se cumple que el factor de utilización del conjunto de tareas es menor que la expresión:

$$\sum_{i=1}^{n} \frac{C_i}{D_i} \leq n(2^{1/n} - 1)$$

2.3.2.2. Deadline Monotonic

Propuesto en 1982 por Leung y Whitehead [19] como una extensión de Rate Monotonic (DM) donde las tareas pueden tener un plazo relativo menor que su período.

De acuerdo al algoritmo DM, cada tarea es asignada una prioridad inversamente proporcional a su plazo relativo. Así, en algún instante, la tarea con el plazo relativo mas corto es ejecutado. Puesto que el plazo relativo es constante, DM es un planificador de prioridad estático. Como RM, DM usa adelanto en su planificación.

DM es un planificador óptimo, esto es si un algoritmo de planificación con prioridad estático puede planificar un conjunto de tareas con plazos diferentes a sus periodos, entonces DM también planificara este conjunto de tareas.

La factibilidad de un conjunto de tareas con plazo menor a su plazo puede ser garantizada usando el algoritmo de planificación Rate-Monotonic, reduciendo los periodos de las tareas a su plazo relativo.

2.3.3. Planificación con prioridades dinámicas

2.3.3.1. Earliest Deadline First

El principal algoritmo de asignación de prioridades dinámico lo constituye el algoritmo propuesto por Horn en 1974 [16], y llamado Earliest Deadline First (EDF), el cual permite planificar un conjunto de n tareas independientes sobre un procesador, cuando las tareas tienen tiempo de llegada dinámica y el adelanto es permitido. Este resultado puede ser expresado por el siguiente teorema.

Teorema 2 (Horn) *Dado un conjunto de n tareas independientes con tiempo de llegada arbitrario, un algoritmo que en algún instante ejecute la tarea con plazo absoluto más pequeño entre todas las tareas listas, es óptimo con respecto a la minimización del máximo retraso.* Prueba [10]

Para garantizar la factibilidad del planificador producido por el algoritmo EDF, se debe verificar que todas las tareas deben ser completadas antes de su plazo, lo cual esta dado por las siguientes *n* condiciones:

$$\forall i = 1, \mathbf{K}, n \qquad \sum_{k=1}^{i} c_k \leq d_i$$

donde c_i es el tiempo restante de ejecución en el peor caso de la tarea J_i, note que tiene un valor inicial igual a c_i y debe ser actualizado cuando la misma es desalojada.

EDF es un planificador dinámico que selecciona las tareas de acuerdo a su plazo absoluto, utiliza adelanto y puede ser usado para planificación de tareas periódicas como aperiódicas ya que no hace uso de la periodicidad de las tareas para la planificación de las mismas.

Para este caso la menor cota para el factor de utilización del procesador es 1, por lo tanto las tareas pueden utilizar el 100% del procesador y aún ser planificable. En particular el siguiente teorema se mantiene [20, 29]:

Teorema 3. Un conjunto de tareas es *planificable* con EDF si y sólo si

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

Prueba [3].

CAPITULO 3

Ejecutivos Cíclicos

3.1. Introducción

Los ejecutivos cíclicos o planificadores cíclicos ejecutan las tareas en secuencia según unas tablas de planificación o planes de ejecución. Estas tablas o planes se construyen antes de poner en marcha el sistema (off-line) a partir del conjunto de tareas y de sus restricciones temporales.

Un planificador cíclico [2, 12] es un procedimiento iterativo que permite planificar la ejecución de un conjunto de procesos periódicos en un procesador. La planificación es hecha de forma determinista tal que los tiempos de ejecución son predecibles.

A partir del conjunto de tareas y sus requisitos temporales se construye un plan principal. Un plan principal define la secuencia de tareas que deben ejecutarse durante un periodo fijo de tiempo llamado ciclo principal. El plan principal se divide en uno o más planes secundarios los cuales describen la secuencia de tareas que deben ejecutarse durante un periodo fijo de tiempo llamado ciclo secundario. En general los planes secundarios de un mismo plan principal contendrán secuencias de distintas tareas, dependiendo de las características temporales de las tareas.

El instante en que acaba un ciclo secundario y empieza el siguiente se sincroniza con los impulsos producidos por el reloj de ciclo secundario. Esta sincronización permite garantizar el cumplimiento de los requisitos temporales de las tareas. Cada impulso del temporizador indica el fin de un ciclo secundario, de modo que la secuencia de las tareas correspondiente a dicho

ciclo secundario debe haber acabado su ejecución. Cada plan secundario se divide, a su vez, en uno o más marcos. Dentro de cada marco se ejecuta una tarea de la secuencia correspondiente al plan secundario.

La duración de los marcos puede ser controlada mediante un temporizador. Esta temporización se activa al iniciar la ejecución de cada tarea, con un valor igual a su tiempo máximo de cómputo, y se desactiva tan pronto como el proceso acaba su ejecución. Por lo tanto, los marcos son de duración variable dependiendo del tiempo de cómputo de las tareas.

El problema de encontrar una planificación admisible es NP-Completo [13] de modo que el único procedimiento para encontrarla o bien asegurar que no existe es mediante búsqueda exhaustiva.

La secuencia de procesos correspondiente a cada ciclo secundario y la secuencia de planes secundarios correspondiente al plan principal, así como la duración de los marcos, ciclos secundarios y, por tanto, ciclo principal se reflejan en tablas que definen cada plan principal. El planificador cíclico utiliza estas tablas para llevar a cabo la planificación.

3.2. Planificador con reloj de ciclo secundario

Un planificador con reloj de ciclo secundario es aquel que produce un pulso de reloj cada vez que analiza un ciclo secundario.

3.2.1. Determinación de los periodos de los ciclos

Sea un conjunto n de tareas periódicas τ representadas por el conjunto $\{(C_1,T_1,D_1), (C_2,T_2,D_2), \ldots(C_n,T_n,D_n)\}$, con $C \leq D \leq T$, donde C representa el tiempo de ejecución, D el plazo y T el periodo. La duración del ciclo principal M, denominado también hiperperiodo, es el mínimo común múltiplo de los periodos de las tareas,

$$M = \text{mcm}(T_i) \tag{3.1}$$

En cuanto a la duración del ciclo secundario m, puede haber varias soluciones admisibles. Si solo existe un temporizador que controla la duración de los ciclos secundarios y, por lo tanto, el planificador solo controla la ejecución de la secuencia de tareas al final de cada ciclo secundario. Entonces el valor de m debe cumplir las siguientes condiciones [2]:

1. m debe ser menor o igual que el plazo de ejecución de cualquier tarea:

$$\forall i : m \leq D_i \tag{3.2}$$

Esta condición es necesaria para que entre el instante de activación de cada tarea y su plazo límite pueda haber un ciclo secundario completo.

2. *m* debe ser mayor o igual que el mayor de los tiempos de computo máximos de las tareas:
3.

$$m \geq \max(C_i) \tag{3.3}$$

Es decir, cada tarea debe ejecutarse completamente en un ciclo secundario.

4. *m* debe dividir la duración del ciclo principal *M*:

$$\exists k : M = k.m \tag{3.4}$$

Dicho de otro modo, todos los ciclos secundarios son de igual duración Teniendo en cuenta las condiciones (3.1) y (3.2) esto equivale a que *m* divida algún T_i:

$$\exists i, k : T_i = k.m \tag{3.5}$$

5. *m* debe cumplir:

$$\forall i : m + (m - mcd(m, T_i)) \leq D_i \tag{3.6}$$

Esta condición incluye a la (3.2), y es necesaria y suficiente para que entre el instante de activación de cada tarea y su plazo límite pueda haber un ciclo secundario completo. De este modo, establece una duración del ciclo secundario que permite garantizar la periodicidad de las tareas y, a la vez, detectar posibles pérdidas en el plazo de respuesta.

3.2.2. Inclusión de ejecuciones de tareas en ciclos secundarios

A partir de las condiciones (3.2), (3.3), (3.4) y (3.6) se obtiene el conjunto de valores que puede tomar la duración del ciclo secundario *m*. Ya que la duración de *M* es fija, según el valor de *m* elegido se tienen $n_{cs} = M / n$ ciclos secundarios en el plan principal. Así que un plan principal esta definido por n_{cs} planes secundarios. Por otra parte, cada plan principal, incluye

$$n_{e_i} = \frac{M}{T_i} \tag{3.7}$$

ejecuciones de cada tarea τ_i. De modo cada plan secundario está constituido por una secuencia $s : \{t_{i_k}, t_{j_l}, \mathbf{K}, t_{r_s}\}$ de ejecuciones de tareas. Para definir un plan principal, es necesario definir las secuencias de tareas de todos los ciclos secundarios, de tal forma que las tareas se ejecuten con los periodos y plazos de ejecución especificados.

A partir de la condición (3.6) se infiere que la ejecución k-esima, $\forall k \in [1\mathbf{K}\, n_{e_i}]$, de un tarea τ_i, caracterizada por (C_i,T_i,D_i), podrá ser incluida en los ciclos secundarios j-esimos, $\forall j \in [1\mathbf{K}\, n_{c_s}]$, tal que:

$$(k-1) \times T_i \leq (j-1) \times m \leq (k-1) \times T_i + D_i - m \quad (3.8)$$

Es decir, podrá ser incluido en aquellos ciclos secundarios que empiecen simultáneamente o después del instante de su k-esima activación y finalicen antes o a la vez que su k-esimo plazo de respuesta. Debido a que el incumplimiento del plazo de respuesta de la tarea se detecta mediante el desbordamiento del ciclo secundario donde se incluye, debe haber dos pulsos del reloj de ciclo secundario entre cada activación y el correspondiente plazo de respuesta.

Cada ciclo secundario m_j que verifique (3.8) tendrá asociado, en una etapa de la construcción del plan de ejecución, una secuencia de ejecuciones de tareas $s : \{t_{i_k}, t_{j_l}, \mathbf{K}, t_{r_s}\}$.

Para que la ejecución k-esima de P_i, P_{i_k} pueda ser incluida en $s(m_j)$, de forma que se garanticen sus restricciones temporales, m_j ha de cumplir, además de (3.8) la siguiente condición:

$$C_i \leq m - \sum_{\forall l | t_l \in s(m_j)} C_l \quad (3.9)$$

Es decir, debe quedar tiempo libre en dicho ciclo secundario para que pueda ejecutarse la tarea. Si m_j verifica (3.8) y (3.9), t_{i_k} se incluye en $s(m_j)$ obteniéndose $s(m_j) : \{t_{f_l} \mathbf{K} t_{r_s}, t_{i_k}\}$.

3.2.3. Búsqueda de una planificación admisible

Una planificación admisible es aquella que permite que el conjunto de tareas se ejecute de forma que se cumplan sus requisitos temporales.

Las condiciones que debe cumplir un ciclo secundario para que pueda incluirse en él la ejecución de una tarea han sido establecidas anteriormente. Pero, el problema de encontrar una planificación admisible es NP-Completo [13] y, por lo tanto, no se conocen algoritmos eficientes para encontrar una solución de forma general.

El problema de encontrar una planificación admisible para *n* tareas tiene similitud al problema clásico de situar n-reinas en un tablero de ajedrez, si se hace corresponder tareas con reinas y situaciones de reinas con planificaciones de tareas. En el problema de las n-reinas, una situación es admisible si no existen dos reinas en la misma columna, fila o diagonal. En nuestro caso, una planificación es admisible si las ejecuciones de las tareas satisfacen sus restricciones temporales, tanto sus periodos como sus plazos de ejecución.

Para resolver este tipo de problemas se pueden utilizar algoritmos dé retroceso (bach-traching). Estos algoritmos buscan una solución recorriendo exhaustivamente el espacio de soluciones del problema. La búsqueda se facilita organizando en árbol el espacio de soluciones.

En este caso, la raíz de éste árbol Π_o es la planificación trivialmente admisible para 0 tareas. Cada nodo $\Pi_{i,j}$ de este árbol contiene una solución parcial al problema consistente en la planificación de *i* tareas, donde *i* es la profundidad del nodo. Cada nodo $\Pi_{i,j}$ tiene *h* posibles sucesores, estos nodos sucesores contemplan todos los modos de incluir en la planificación uno de las *n*-1 tareas que quedan por planificar. El espacio de soluciones está definido por todos los caminos desde el nodo raíz a los nodos terminales. Cada nodo del árbol del espacio de soluciones representa un estado del problema, así a la organización en árbol del espacio de soluciones se le llama árbol del espacio de estados.

3.2.4. Complejidad del problema

El espacio de soluciones está constituido por todas y cada una de las maneras de incluir las *n* tareas en la planificación, la cual esta dado por la siguiente ecuación:

$$\prod_{i=1}^{n}\binom{n_{cs}}{n_{e_i}} = \frac{(n_{cs}!)^n}{\Pi_{i=1}^{n} n_{e_i} \times (n_{cs} - n_{e_i})!} \tag{3.10}$$

De donde se deduce que la complejidad del problema aumenta con el número de tareas y con el número de ciclos secundarios. Por lo tanto, si existen varios valores de *m* que verifiquen las ecuaciones (2), (3), (4) y (6), se deben elegir primero los mayores valores para intentar encontrar una planificación admisible.

3.2.5 Algoritmo de retroceso

El algoritmo de retroceso recorre en profundidad el árbol del espacio de estados hasta encontrar una solución admisible al problema. Esto se corresponde con la generación en profundidad de todos los estados posibles. Para evitar generar todos estos estados se utiliza una función de poda (bounding function) que identifique aquellos nodos que no conducen a una solución admisible sin necesidad de generar el subárbol correspondiente.

En el problema de encontrar una planificación admisible, si en un determinado estado del problema $\Pi_{i,j}$ no es posible incluir la planificación de una tarea τ_i cualquiera, entonces no podrá ser incluido en la planificación para cualquier estado del problema $\Pi_{i+f,g}$ derivado de $\Pi_{i,j}$.

3.2.6 Optimización del camino de búsqueda

En el peor de los casos se debe recorrer exhaustivamente el árbol del espacio de soluciones para encontrar una planificación admisible para el conjunto de las *n* tareas, o bien, para asegurar que no existe. Debido a la gran complejidad del espacio de soluciones (3.10), conviene aplicar

reglas heurísticas que permitan obtener soluciones en la mayoría de los casos con una menor complejidad.

Estas reglas deben permitir elegir la mejor opción a la hora de expandir un nodo con alguno de sus sucesores, de forma que se pueda esperar razonablemente llegar por este caminó a un nodo de profundidad *n*, generando un menor número de nodos del árbol y, por lo tanto, reduciendo, en la mayoría de los casos, la complejidad del problema.

Elección de la tarea a incluir

Existen diferentes criterios para elegir la tarea a incluir algunas de las cuales son:

1. Añadir la tarea con el plazo de ejecución más corto entre los restantes. Este criterio se basa en el método de planificación monótona con respecto al plazo de respuesta (deadline monotonic).
2. Añadir la tarea con el período más corto entre los que restan por planificar. Este criterio está basado en el método de planificación monótona en frecuencia (rate monotonic).
3. Añadir la tarea con el mayor tiempo de cómputo entre los que restan por planificar. Este criterio está basado en la regla de elegir primero los las tareas con mayor tiempo de cómputo (largest processing time).

Estos tres criterios seleccionan primero a los procesos con menos posibilidades o más complicados de incluir en la planificación, de forma que se pueda esperar llegar a la solución con un menor número de retrocesos que si se elige la tarea a incluir de forma arbitraria.

En los algoritmos desarrollados, se eligen las tareas según el orden que establece la primera regla, utilizando las otras dos para seleccionar uno entre aquellos que tengan el mismo plazo de respuesta o período.

Elección del ciclo secundario donde incluir la ejecución de una tarea

Una vez elegida una tarea τ_i, hay que incluir sus n_{ei} ejecuciones en los planes secundarios. En el caso general, para cada ejecución τ_{ij} existirá más de un ciclo secundario donde pueda incluirse.

El problema de añadir tareas a una planificación parcial se puede asimilar al problema de empaquetar bultos (bin parking) [13]. Este problema es NP-Hard y consiste en empaquetar *n* bultos en cajas de igual capacidad *L*. Cada bulto *i* ocupa l_i.

Para elegir un ciclo secundario, donde incluir la ejecución τ_{ij} se pueden aplicar las siguientes reglas heurísticas que se utilizan en el problema de empaquetar cajas:

1. Incluir la tarea en el primer ciclo secundario, que verifique sus requisitos temporales, en el qué quede un tiempo de cómputo mayor o igual al tiempo máximo de cómputo de la tarea. Esto equivale a empaquetar el bulto en la primera caja en la que quepa (first fit).

2. Encajar la tarea en el ciclo secundario en el que quede menor tiempo de cómputo libre y se garanticen los requisitos temporales. Esto equivale a empaquetar el bulto en la caja que tenga la menor capacidad libre y esta sea mayor o igual a la que necesita el bulto (best fit).

Por otra parte, se conoce de antemano el número de ciclos secundarios, se puede utilizar otra regla que realiza un empaquetado simétrico al de la primera.

3. Incluir la tarea en el último ciclo secundario en el que quepa y se garanticen sus requisitos temporales. Esto equivale a empaquetar el bulto en la última caja en la que quepa (last fit).

Puesto que la primera y tercera regla, no exigen cálculos adicionales, es recomendable incluir la ejecución de la tarea usando primero dichas reglas, y si no es posible, o se alcanza un nodo muerto, insertar la tarea utilizando la segunda hasta completar todos los casos posibles.

3.2.7. Algoritmo

La estructura del programa principal de la herramienta desarrollada puede observarse en la Figura 3.1. Este procedimiento se encarga de calcular los parámetros iniciales de las posibles planificaciones basadas en las reglas dadas en sección 3.2.1, inicializar las estructuras de datos y ordenar el conjunto de tareas en base a las reglas heurísticas enunciadas en la sección A.2.6. Posteriormente, se intenta encontrar una planificación admisible comenzando por aquella que posee un menor número de ciclos secundarios. La planificación se construye recursivamente, realizando una llamada; al procedimiento Planificar-Tarea, cuyo parámetro es el identificador de la primera tarea.

La estructura del procedimiento Planificar-Tarea se observa en la Figura 3.2. Para planificar un tarea es necesario planificar todas y cada una de sus ejecuciones. Si alguna de ellas no es planificable no lo es la tarea y, por tanto, se ha alcanzado un nodo muerto. En este caso, se debe retroceder en el árbol para planificar de otro modo la tarea que fue planificado anteriormente.

En general, cada ejecución de la tarea podrá ser planificada en varios ciclos secundarios. El orden en la elección del ciclo secundario donde la tarea va a ser planificado se realiza en base a las reglas heurísticas dadas en la sección 3.2.6. Si la complejidad del problema es muy elevada y no se quiere realizar una búsqueda exhaustiva, sólo se intenta planificar la ejecución en los ciclos secundarios seleccionados por las reglas heurísticas.

Por último, se genera un nuevo nodo del árbol incluyendo todas las ejecuciones de la tarea en la planificación. Para generar este nuevo nodo, se realiza la llamada al procedimiento

Planificar-Ejecución dando como parámetros el identificador de la tarea y su primera ejecución. Antes de esto, se comprueba si se ha alcanzado un nodo terminal y se ha construido una planificación.

El procedimiento Planificar_Ejecución se muestra en la Figura 3.3. Este procedimiento incluye la ejecución en un ciclo secundario de los calculados en Planificar_Tarea, respetando el orden establecido. Una vez hecho esto se procede a planificar la siguiente ejecución llamando recursivamente a Planificar_Ejecución con el identificador de la tarea y el de la siguiente ejecución. Si al finalizar esta llamada se han planificado todos las tareas, se ha construido una planificación, en caso contrario, se ha alcanzado un nodo muerto y es necesario incluir en otro ciclo secundario la ejecución. Si se han agotado los ciclos secundarios, entonces se deberá planificar de otro modo la ejecución anterior.

Finalmente, se comprueba si se han incluido todas las ejecuciones de las tareas. Si es así, se ha acabado la generación del nodo y se procede a generar uno nuevo mediante una llamada a Planificar_Tarea con la tarea siguiente.

Procedimiento Principal:

1. Calcular la duración del ciclo principal utilizando la ecuación (3.1),
2. Calcular el número de ejecuciones de cada tarea utilizando la ecuación (3.7);
3. Calcular el conjunto de duraciones posibles del ciclo secundario utilizando las condiciones (3.2), (3.3), (3.4) y (3.6);
4. Ordenar este conjunto de tareas de mayor a menor;
5. Ordenar el conjunto de tareas según las reglas heurísticas dadas en la sección 3.2.6;
6. Mientras el conjunto de valores para el ciclo secundario no sea el vacío y no se encuentre una planificación:
   - 6.1. Elegir el primer valor *m* para la duración del ciclo secundario;
   - 6.2. Construir la planificación vacía para M/m ciclos secundarios;
   - 6.3. Planificar_Tarea(primero);
   - 6.4. Si se han planificado todas las tareas:
     Se ha encontrado una planificación;
     Si no:
     Quitar el primer valor de ciclo secundario del conjunto de valores posibles;
7. Si no se ha encontrado una planificación:
   El conjunto de tareas no admite una planificación cíclica;
8. Fin Principal.

Figura 3.1. Procedimiento Principal

Procedimiento Planificar_Tarea (i-ésima):

1. Si la tarea i-ésima es posterior a la última: Se han planificado todos las tareas;
2. Si no:
   2.1. Para toda ejecución j de la tarea i-ésima:
      2.1.1. Calcular el conjunto de ciclos secundarios que verifican las condiciones (3.8) y (3.9);
      2.1.2. Si el conjunto de ciclos secundarios es vacío:
         2.1.2.1. La tarea no es planificable;
         2.1.2.2. Salir del bucle;
      2.1.3. Ordenar este conjunto de ciclos según las reglas heurísticas dadas en la sección 3.2.6.;
      2.1.4. Si no se va a realizar búsqueda exhaustiva:
         2.1.4.1. Truncar el conjunto de ciclos secundarios y conservar sólo los primeros;
   2.2. Si la tarea es planificable:
      2.2.1. Planificar-Ejecución (i-ésima, primera);
3. Fin Planificar_Tarea

Figura 3.2. Procedimiento Planificar-Tarea

Procedimiento Planificar-Ejecución (i,j):

1. Si j es posterior a la última ejecución de la tarea i:
   1.1. Planificar-Tarea (i + 1);
2. si no:
   2.1. Para todo k en el conjunto de ciclos secundarios de $\tau_{i,j}$:
      2.1.1. Incluir $\tau_{i,j}$ en ciclo secundario k;
      2.1.2. Planificar-Ejecución (i,j+l);
      2.1.3. Si se han planificado todos las tareas:
         2.1.3.1. Salir del bucle;
      2.1.4. Si no:
         2.1.4.1. Quitar $\tau_{i,j}$ de ciclo secundario k;
3. Fin Planificar-Ejecución.

Figura 3.3. Procedimiento Planificar-Ejecución

La Figura 3.4 muestra el diagrama de flujo en el cual se basa la herramienta CICLIC para determinar y simular planificadores cíclicos.

El algoritmo esta formado por varios procesos. El primero y basado en las especificaciones temporales para el grupo de tareas (Proceso A en la Figura 3.4), verifica las condiciones de

planificabilidad (Proceso B en la Figura 3.4) a través de las reglas dadas en la sección 3.2.1. Si estas condiciones se cumplen, se determina tanto el ciclo primario y el ciclo secundario de la planificación (Proceso D en la Figura 3.4). A continuación el grupo de tareas es ordenado de acuerdo a las reglas heurísticas dadas en la sección 3.2.6. y se determinan los tiempos de ejecución de cada tarea (Proceso D en la Figura 3.4). Posteriormente se insertan las tareas en los ciclos secundarios basados en las reglas dadas en la sección 3.2.6. (Proceso E en la Figura 3.4). . Este proceso construye la planificación cíclica del conjunto de tareas de forma recursiva, hasta que una planificación cíclica factible es encontrada (todas las tareas son insertadas en ciclos secundarios y todos los requisitos temporales se cumplen) (Proceso G en la Figura 3.4)o hasta que el conjunto de tareas no pueda ser insertado (Proceso F en la Figura 3.4) con lo cual una planificación cíclica factible para este conjunto de tareas no puede ser encontrado (Proceso H en la Figura 3.4).

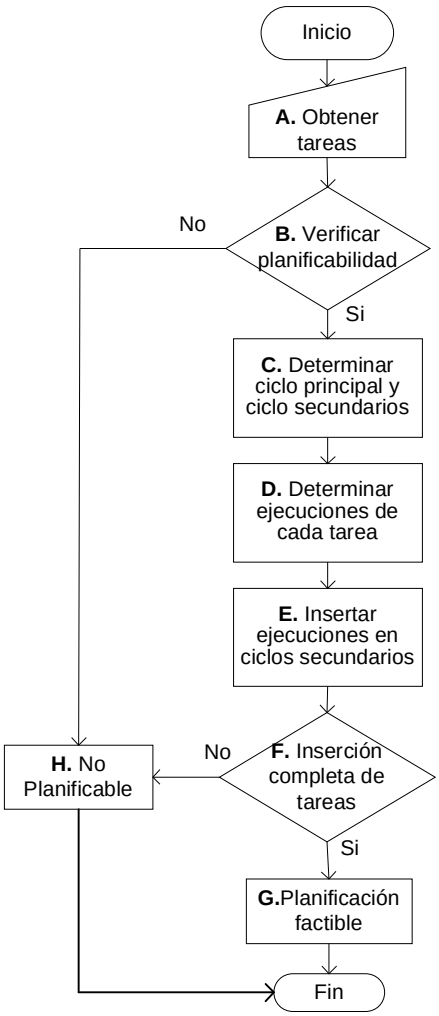


Figura 3.4. Diagrama flujo del algoritmo de la herramienta CICLIC

CAPITULO 4

Herramienta para generar ejecutivos cíclicos

4.1. Introducción

CICLIC es una herramienta Java para el diseño y desarrollo de planificadores cíclicos de sistemas de tiempo real sobre un procesador, el cual provee facilidades de análisis y simulación. Así, CICLIC permite establecer planificaciones cíclicas, que pueden ser implementadas posteriormente sobre cualquier procesador y sistema operativo de tiempo real. Los fundamentos teóricos en los cuales se basa la herramienta CICLIC y los algoritmos de las principales funciones implementadas están descritas en el capitulo 3.

Los usuarios de CICLIC especifican las tareas del sistema. Todas las especificaciones son leídas desde los archivos de especificaciones proveídas por el usuario y compiladas en una representación interna. Estas descripción puede ser fácilmente vista y modificada a través de interfaces graficas con el usuario (GUI). El usuario puede asignar a cada tarea un color, para posteriormente determinar fácilmente y verificar la ocurrencia de eventos relevantes, medir intervalos de tiempo y verificar rápidamente el cumplimiento de restricciones temporales. Después de leer las características de las tareas del sistema, se pueden emplear utilidades de análisis y simulación. CICLIC muestra a través de diferentes ventanas:

- Especificaciones del conjunto de tareas,
- Análisis de planificación,
- Construir planificadores cíclicos basados en reloj de ciclo secundario
- Determinar el factor de utilización del conjunto de tareas a planificar.
- Código C de la planificación encontrada
- Simulación de planificación.

- Para un planificador cíclico encontrado y en el caso de que el factor de utilización sea menor que uno:
  - La herramienta permite determinar el tiempo de cómputo máximo que una tarea debe tener para que pueda ser añadida a esta planificación.
  - La herramienta permite determinar el tiempo de cómputo máximo que una de las tareas puede alcanzar en la planificación cíclica.

De esta manera, CICLIC permite reducir el impacto de actualizaciones y/o modificaciones de las tareas, facilitando su realización.

4.2. Requerimientos de software

Es posible instalar y ejecutar CICLIC en ordenadores que usen las plataformas Windows, MAC OS o Linux. CICLIC fue desarrollado usando el lenguaje de programación Java por lo que requiere del entorno Java para funcionar.

Para ejecutar CICLIC se deben copiar los archivos necesarios en una carpeta y ejecutar el archivo planificador, a partir del cual se ejecutará el ambiente gráfico de CICLIC el cual es descrito en la sección 4.5. El programa ejecutable y el código fuente de esta herramienta se encuentran en el CD anexo a este documento.

4.3. Análisis de Planificabilidad

CICLIC verifica las condiciones suficientes de planificabilidad para planificadores cíclicos (sección 2.1) y determina el factor de utilización de la CPU. Si no existe ningún valor que verifique el conjunto de condiciones, entonces el conjunto de tareas no puede ser planificado a través un planificador cíclico.

Si una planificación cíclica es encontrada para un conjunto de tareas y en el caso en el que el factor de utilización sea menor que uno, CICLIC permite determinar:

- el tiempo de cómputo máximo que una tarea debe tener para que pueda ser añadida a esta planificación.
- el tiempo de cómputo máximo que una de las tareas puede alcanzar en la planificación cíclica.

4.4. Código C

CICLIC genera el conjunto de instrucciones en código C correspondientes a la

planificación encontrada.

4.5. Área de trabajo de CICLIC

Una vez haya ingresado a CICLIC, aparecerá el espacio de trabajo del mismo (figura 4.1), el cual esta compuesto por los siguientes elementos:

- Barra de herramientas
- Área de datos: donde se visualizan las características (Nombre, tiempo de cómputo, periodo y plazo) del conjunto de tareas a ser planificadas.
- Área de resultados: donde se visualizan las características de la planificación cíclica en el caso que exista, del conjunto de tareas especificadas en el área de datos.

A continuación se describen cada uno de estos elementos en detalle.

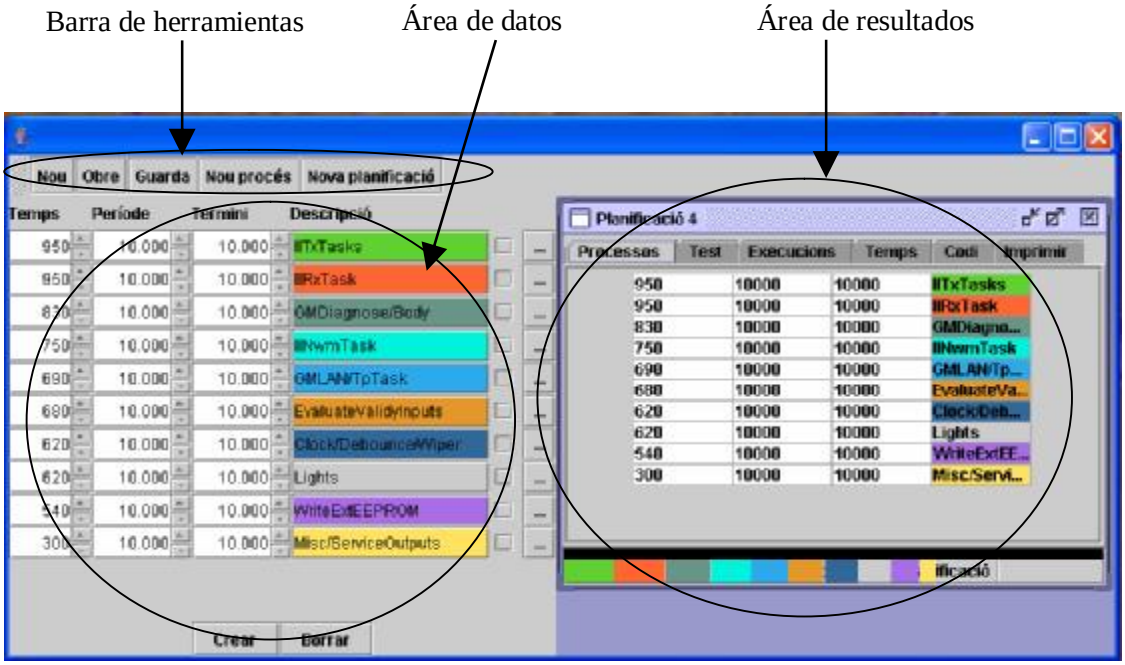


Figura 4.1. Área de trabajo herramienta CICLIC v 1.0.

4.5.1. Barra de herramientas

CICLIC dispone de una barra de herramienta ubicada en la parte superior derecha de la ventana, la cual contiene los siguientes botones:

4.5.1.1 Botón Nou:

El botón Nou permite introducir un nuevo conjunto de tareas, eliminando todas las tareas existentes en el área de datos en el caso de que exista alguna.

4.5.1.2 Botón Obre:

Al presionar el botón Obre aparecerá la ventana que se muestra en la Figura 4.2, desde la cual se puede seleccionar el archivo que contiene las especificaciones del conjunto de tareas a planificar. Si existen tareas en el área de datos, los datos de las tareas contenidos en el archivo se anexaran a las ya existentes.

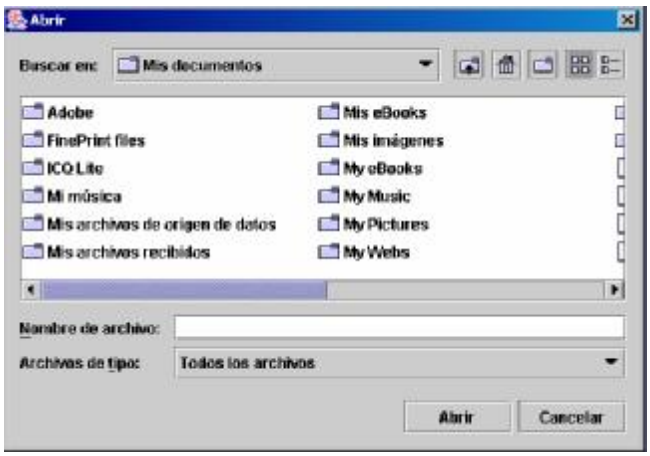


Figura 4.2. Ventana Abrir.

4.5.1.3 Botón Guarda:

Al presionar este botón Guarda aparecerá la ventana que se muestra en la Figura 4.3, la cual permite guardar el conjunto de tareas que se encuentran en el área de datos en un archivo.

4.5.1.4 Botón Nou procés:

El botón Nou procés permite añadir una nueva tarea o proceso al conjunto de tareas en el área de datos. Al presionar este botón la Figura 4.4 es desplegada en pantalla.

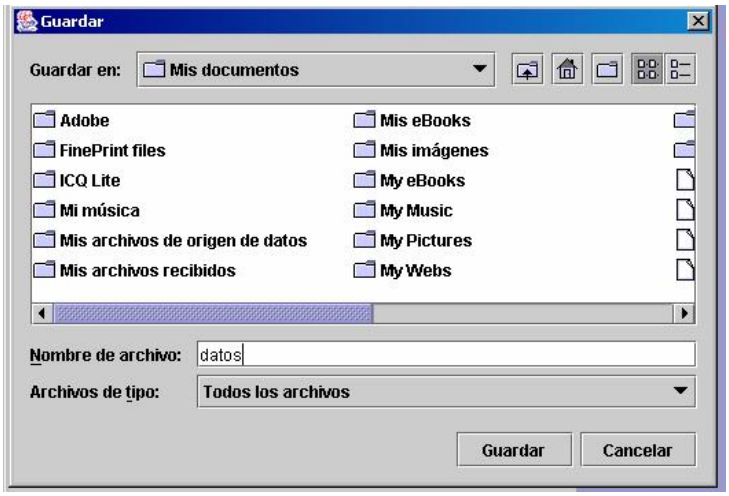


Figura 4.3. Ventana Guardar

Figura 4.4. Parámetros de la nueva tarea a insertar.

En la Figura 4.4 se debe especificar el tiempo de cómputo, período, plazo y nombre o descripción de la tarea insertada. Además del nombre, cada tarea esta identificada por un color, el cual es asignado de forma arbitraria al crear la tarea. Si se desea cambiar este color, se debe hacer doble clic sobre el área coloreada en el campo descripción, lo cual abrirá la ventana que se muestra en la Figura 4.5, a partir de la cual se puede asignar un nuevo color a la tarea.

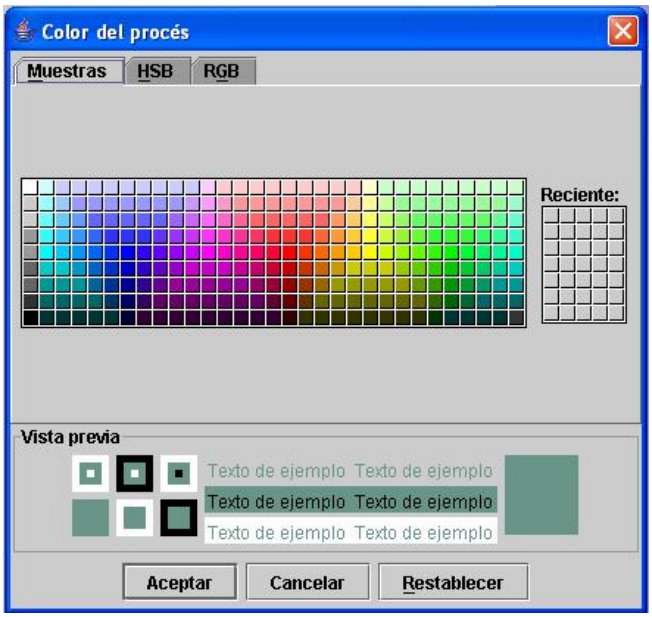


Figura 4.5. Asignación de color a una tarea.

4.5.1.5 Botón Nova planificació:

El botón 

permite verificar si es posible encontrar una planificación cíclica para el conjunto de tareas especificadas en el área de datos. Si existe una planificación cíclica, aparecerá la siguiente ventana donde se deberá escoger el tiempo del ciclo secundario:

Figura 4.6. Selección del tiempo del ciclo secundario.

Por otro lado, CICLIC permite encontrar el tiempo máximo de cómputo que una tarea puede tener para que sea planificadle. En este caso se debe especificar la tarea a ser maximizada, a través de la ventana que se muestra en la Figura 4.7.

Figura 4.7. Selección del tiempo del ciclo secundario.

Una vez seleccionado el ciclo secundario y el proceso a maximizar, CICLIC mostrará la planificación cíclica encontrada y sus características en el área de resultados.

En el caso de que no exista una planificación el mensaje que se muestra en la Figura 4.8 será mostrado por pantalla:

Figura 4.8. Mensaje de planificación cíclica no encontrada.

4.5.2. Área de datos

Esta área esta compuesta por diferentes casillas en las cuales se puede:

- **Introducir nuevas tareas:** existen dos maneras de introducir tareas, la primera es a través del botón Nou procés como se describió en la sección 4.5.1.1. La segunda es utilizando el botón Crear ubicado en la parte inferior del área de datos, ver Figura 4.9.

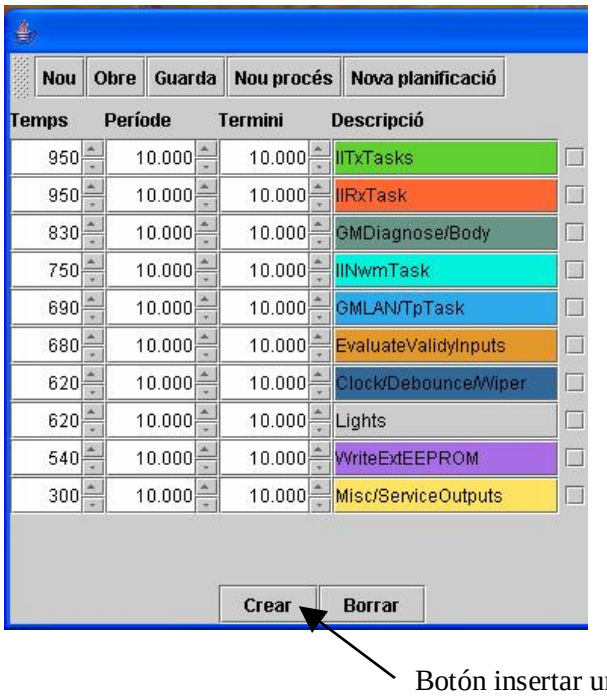


Figura 4.9. Botón para crear una nueva tarea.

- **Modificar tareas existentes:** para modificar las características temporales de una tarea (tiempo de cómputo, período, plazo, descripción), simplemente se debe seleccionar con el ratón la casilla correspondiente y sobrescribir sobre ella los nuevos valores.
- **Eliminar tareas existentes:** para eliminar una tarea se debe seleccionar la casilla de verificación que se encuentra a la derecha de la casilla de descripción de la tarea y presionar el botón Borrar ubicado en la parte inferior del área de datos, ver Figura 4.10.

4.5.3. Área de resultados

Al presionar el botón Nova planificació y en el caso de que exista una planificación cíclica, aparecerá una ventana en el área de resultados la cual está compuesta por las siguientes fichas.

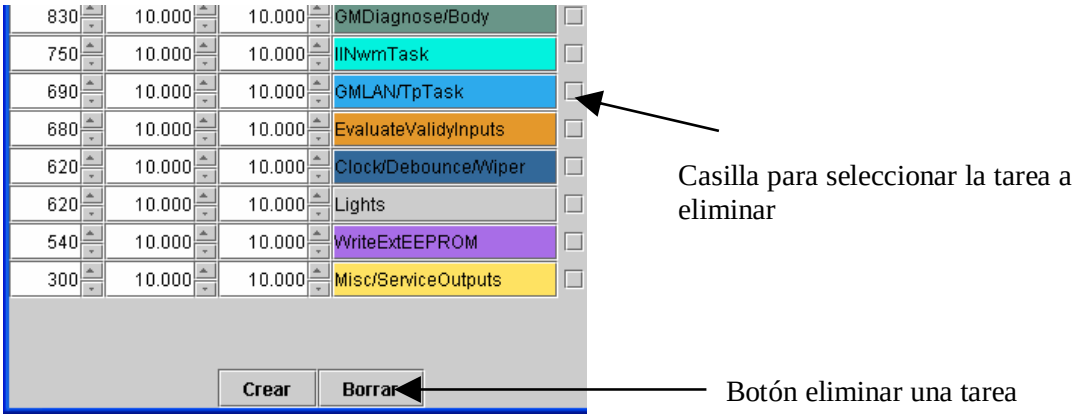


Figura 4.10. Botón para crear una nueva tarea.

4.5.3.1. Proceso

En esta ficha (ver Figura 4.11) se muestra el conjunto de tarea planificadas y en la parte inferior de la misma una barra que ilustra la planificación obtenida.

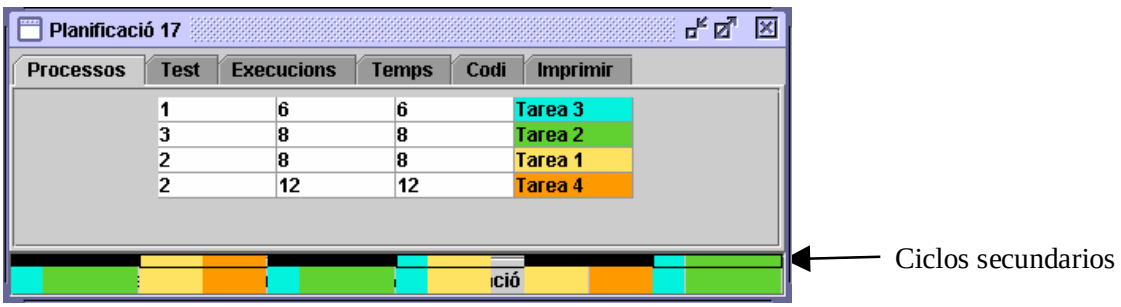


Figura 4.11. Planificación cíclica del conjunto de tareas.

La barra que ilustra la planificación cíclica obtenida, contiene una pequeña barra en la parte superior, la cual esta compuesta por combinación de bloques de color negro y de color, que indican el comienzo y el final de cada ciclo secundario. Específicamente en este ejemplo se tiene 6 ciclos secundarios.

4.5.3.2. Execucions

En esta ficha (ver Figura 4.12) se muestra el tiempo de inicio de ejecución de cada una de las tareas. Esta información puede ser copiada y pegada en otro documento a través del uso del ratón y del teclado.

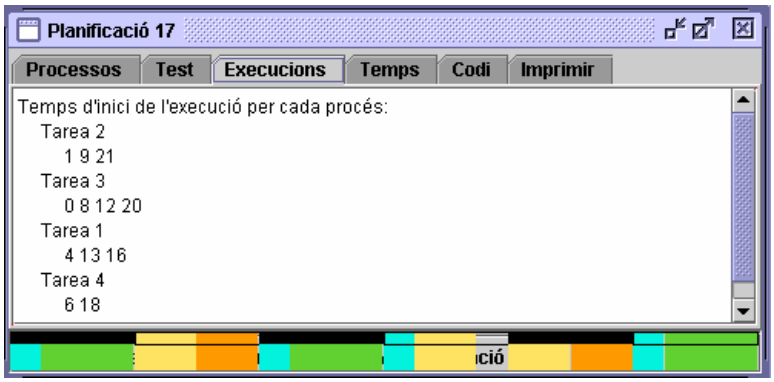


Figura 4.12. Tiempo de inicio de ejecución de cada una de las tareas.

4.5.3.3. Temps

En esta ficha (ver Figura 4.13) se muestra:

1. El orden de ejecución de las tareas y el tiempo de inicio de ejecución de cada una de ellas tiempo de inicio de ejecución de cada una de las tareas.
2. El tiempo de duración del ciclo principal.
3. El tiempo de duración de los ciclos secundarios.
4. Factor de utilización del recurso.

Esta información puede ser copiada y pegada en otro documento a través del uso del ratón y del teclado.

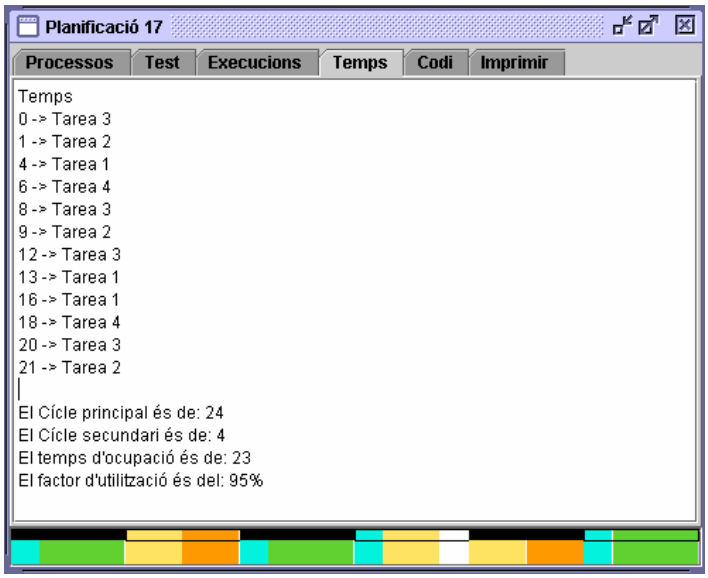


Figura 4.13. Parámetros característicos de la planificación cíclica.

4.5.3.4. Codi

En esta ficha (ver Figura 4.14) se genera el código en el lenguaje C de tal manera que esta información pueda ser utilizada para la programación de un micro controlador. Esta información puede ser copiada y pegada en otro documento a través del uso del ratón y del teclado.

4.5.3.5. Imprimir

En esta ficha (ver Figura 4.15) se muestra la representación gráfica de la planificación obtenida, en la misma puede agregarse el texto que se desee. Esta ficha contiene un botón Imprimir el cual permite imprimir el contenido de esta ficha.

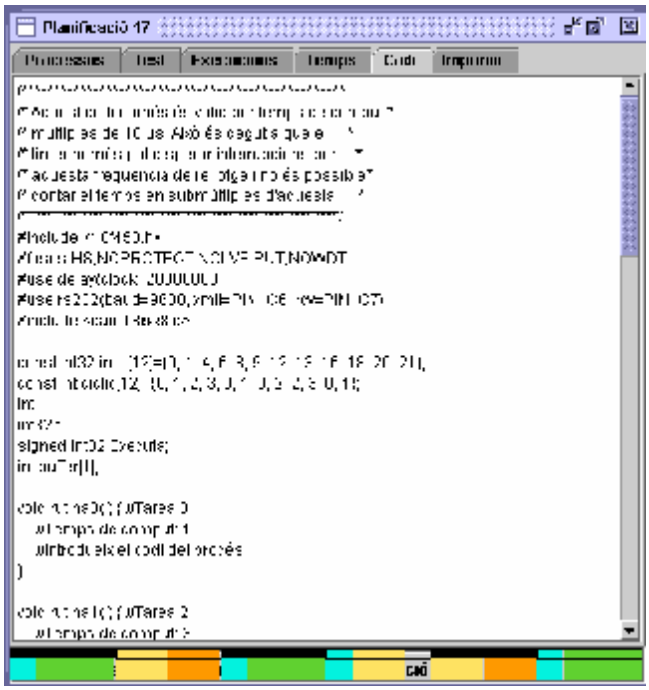


Figura 4.14. Código C de la planificación cíclica.

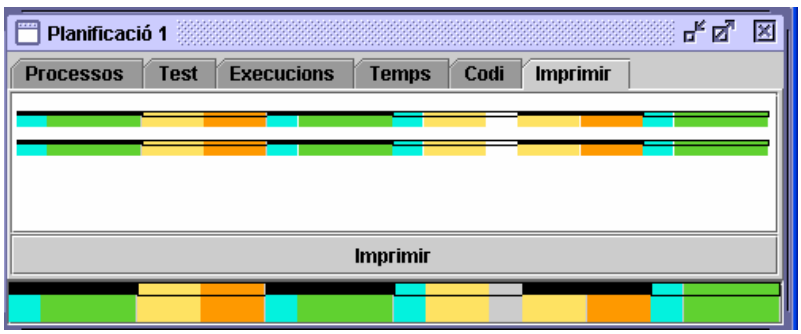


Figura 4.15. Simulación de la planificación cíclica.

CAPITULO 5

Resultados Experimentales

5.1. Caso de estudio

El caso de estudio a tratar en este proyecto se basa en los diferentes procesos que Lear tiene implementado sobre uno de sus microcontrolador: REC – Rear Electrical Center. Las especificaciones temporales de las tareas implementadas sobre este microcontrolador se muestran en la tabla 5.1 2 a partir de las cuales se tienen las siguientes observaciones:

- El microcontrolador está compuesto por 10 tareas (algunas de estas tareas agrupan a otras tareas).
- Se especifican los tiempos de cómputo mínimos, máximos y más frecuentes en la ejecución de cada una de las tareas. Para el estudio nos basaremos en el tiempo de computo máximo, es decir, tiempo de computo en el peor de los casos.
- Los periodos impuestos a cada una de las tareas es igual al ciclo de tiempo seleccionado para el reloj del planificador, es decir, 10 mseg.

#	Tarea	Wcet (µs)	Periodo (µs)
1	Clock/Debounce/Wiper	720	10000
2	Lights	620	10000
3	Misc/ServiceOutputs	300	10000
4	IITxTasks	950	10000
5	IINwmTask	740	25000
6	GMLAN/TpTask	690	10000
7	IIRxTask	950	25000
8	GMDiagnose/Body	830	10000
9	EvaluateValidyInputs	680	10000
10	WriteExtEEPROM	540	10000

Tabla 5.1. Especificaciones temporales para el conjunto de tareas en estudio

5.2. Experimento de simulación sobre PC

5.2.1. Planificación cíclica

Basados en el conjunto de tareas especificadas en la tabla 5.1. y utilizando la herramienta CICLIC desarrollada se obtuvo un planificador cíclico para este conjunto de tareas. La Figura 5.1 muestra la especificación del conjunto de tareas en CICLIC.

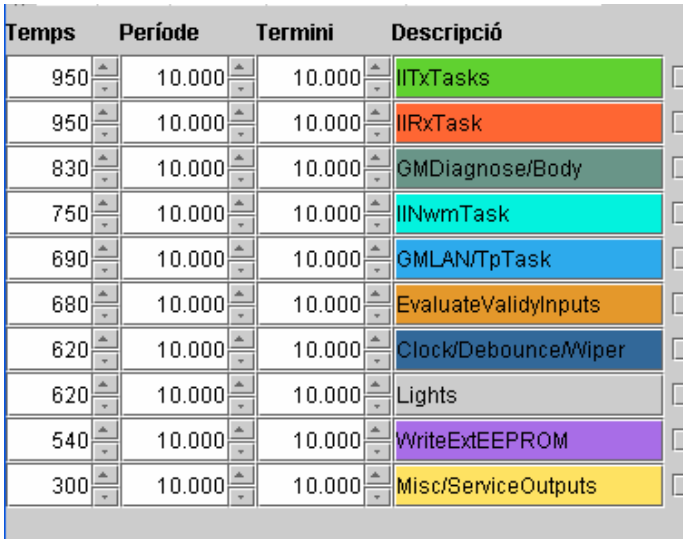


Figura 5.1. Especificación del conjunto de tareas en CICLIC.

Los parámetros de planificación obtenidos de la herramienta CICLIC para este conjunto de tareas se muestran en la Tabla 5.2.

Datos de planificación	
Factor de utilización	0.60
Hiperperiodo (M)	50000
Periodo secundario (m)	1000,1250,2000,2500,3125,5000,10000
Periodo secundario seleccionado	10000
Ciclos secundarios	5

Tabla 5.2. Parámetros del planificador cíclico.

La figura 5.2 muestra la planificación cíclica obtenida para un periodo de ciclo secundario igual a 10000 µseg.

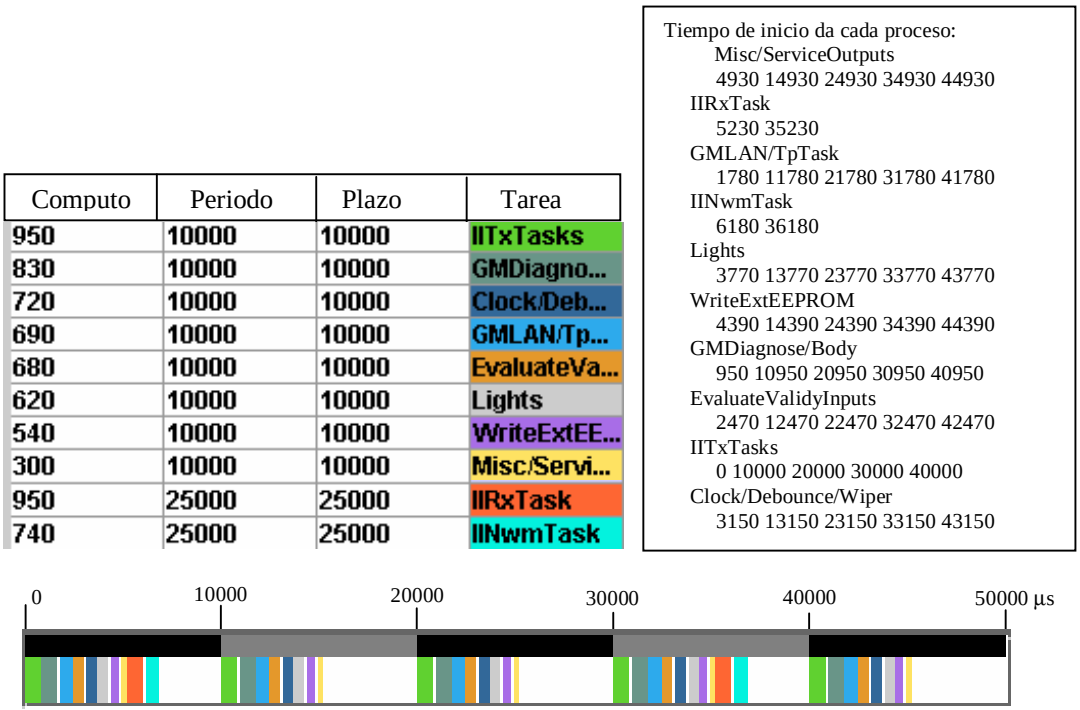


Figura 5.2. Planificación cíclica con reloj de ciclo secundario m = 10000µs.

5.2.2. Estudio de la inserción de nuevas tareas en el microcontrolador

Basados en la planificación cíclica obtenida en el apartado anterior y haciendo uso de la herramienta desarrollada, se obtiene el tiempo de computo máximo que puede tener una tarea de 10 ms., que quiera ser insertada en este planificador, esta nueva tarea tiene como nombre *MAX*. En la tabla 5.3, se muestra el tiempo de cómputo máximo para esta tarea para diferente factor de utilización.

Tiempo de computo C (µs)	Factor de Utilización U (%)
500	65
1000	70
2000	80
3000	90
3720	97

Tabla 5.3. Factor de utilización para diferentes tiempos de cómputo de la nueva tarea.

La figura 5.3, muestra la planificación cíclica obtenida para un periodo de ciclo secundario de 10000 µs, con la inserción de la nueva tarea *MAX* con un tiempo de cómputo máximo de 3720 µs y un factor de utilización del 97%.

Computo	Periodo	Plazo	Tarea
950	10000	10000	IITxTasks
830	10000	10000	GMDiagno...
720	10000	10000	Clock/Deb...
690	10000	10000	GMLAN/Tp...
680	10000	10000	EvaluateVa...
620	10000	10000	Lights
540	10000	10000	WriteExtEE...
300	10000	10000	Misc/Servi...
3720	10000	10000	Max
950	25000	25000	IIRxTask
740	25000	25000	IINwmTask

Tiempo de inicio da cada proceso:
IINwmTask
19050 49050
IIRxTask
9050 39050
GMDiagnose/Body
950 10950 20950 30950 40950
Misc/ServiceOutputs
4930 14930 24930 34930 44930
IITxTasks
0 10000 20000 30000 40000
Clock/Debounce/Wiper
3150 13150 23150 33150 43150
EvaluateValidyInputs
2470 12470 22470 32470 42470
GMLAN/TpTask
1780 11780 21780 31780 41780
WriteExtEEPROM
4390 14390 24390 34390 44390
Lights
3770 13770 23770 33770 43770
Max
5230 15230 25230 35230 45230

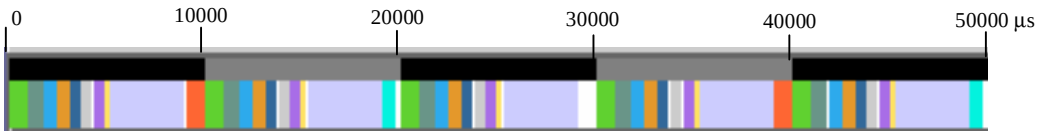


Figura 5.3. Planificación cíclica con inserción de la nueva tarea MAX, m = 10000µs.

5.3. Experimento de emulación sobre tarjeta electrónica

En esta sección, se describe las características técnicas de la tarjeta electrónica utilizada para emular los resultados obtenidos en la sección anterior. Posteriormente se detalla la implementación del conjunto de tareas y la planificación cíclica sobre dicha tarjeta.

5.3.1. Características técnicas de la tarjeta electrónica

Las características técnicas de la tarjeta electrónica utilizada son:

Microprocesador: 18F458

Nº de bits: 8

Velocidad: 20 Mhz

CAN integrado.

La Figura 5.4, muestra la tarjeta electrónica utilizada.

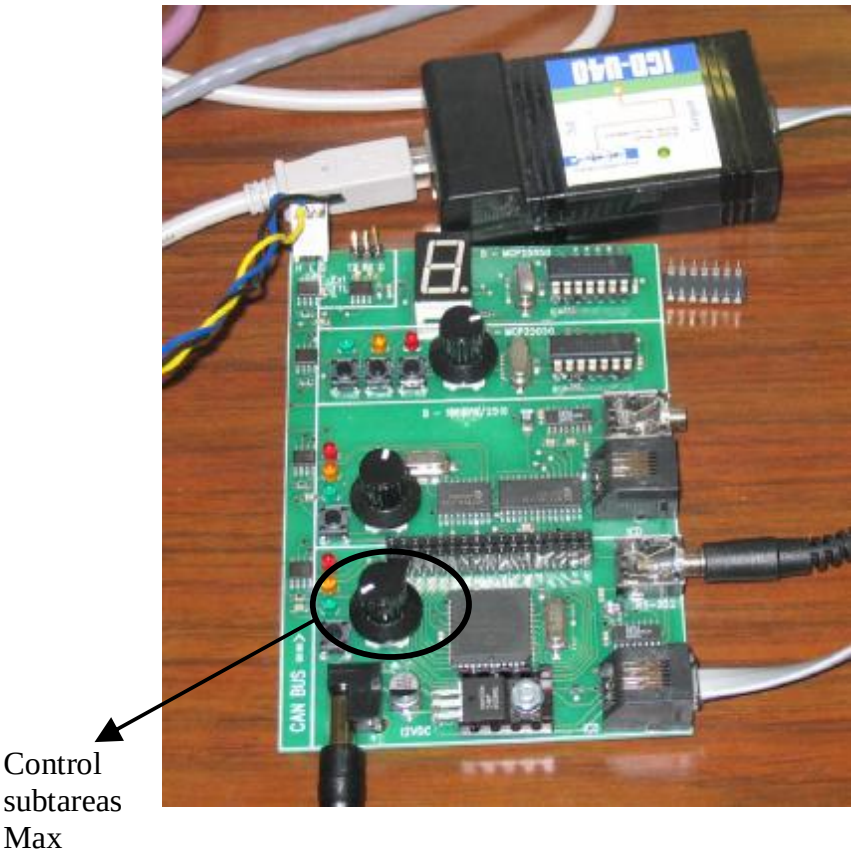


Figura 5.4. Tarjeta electrónica PIC18F458 con CAN integrado.

5.3.2. Tareas emuladas

El conjunto de tareas y la planificación cíclica emulada, estan descritas en la sección 5.2. Específicamente, la Figura 5.3, muestra las especificaciones del conjunto de tareas y la planificación cíclica obtenida para el conjunto de tareas del microcontrolador REC de Lear.

Específicamente la planificación cíclica de este conjunto de tareas fue tomada en cuenta para programar y controlar los mensajes a través de CAN.

La tarea agregada al conjunto de tareas (tarea Max), esta constituida por tres subtareas: OK, Warning, y Danger. Cada una de estas tareas es activada, por valores predeterminados, los cuales son controlados en tiempo de ejecución por el potenciómetro que se señala en la Figura 5.4.

5.3.3. Comprobación de resultados

Para verificar el comportamiento del sistema, se diseño y programo una herramienta en Visual Basic (ver Figura 5.5), a través de la cual en tiempo de ejecución se puede observar el

tiempo de computo de cada tarea, factor de utilización de cada tarea y factor de utilización total.

De las mediciones observadas se pudo verificar y constatar el correcto funcionamiento del conjunto de tareas, lo cual verifica y valida los resultados obtenidos al aplicar la metodología propuesta.

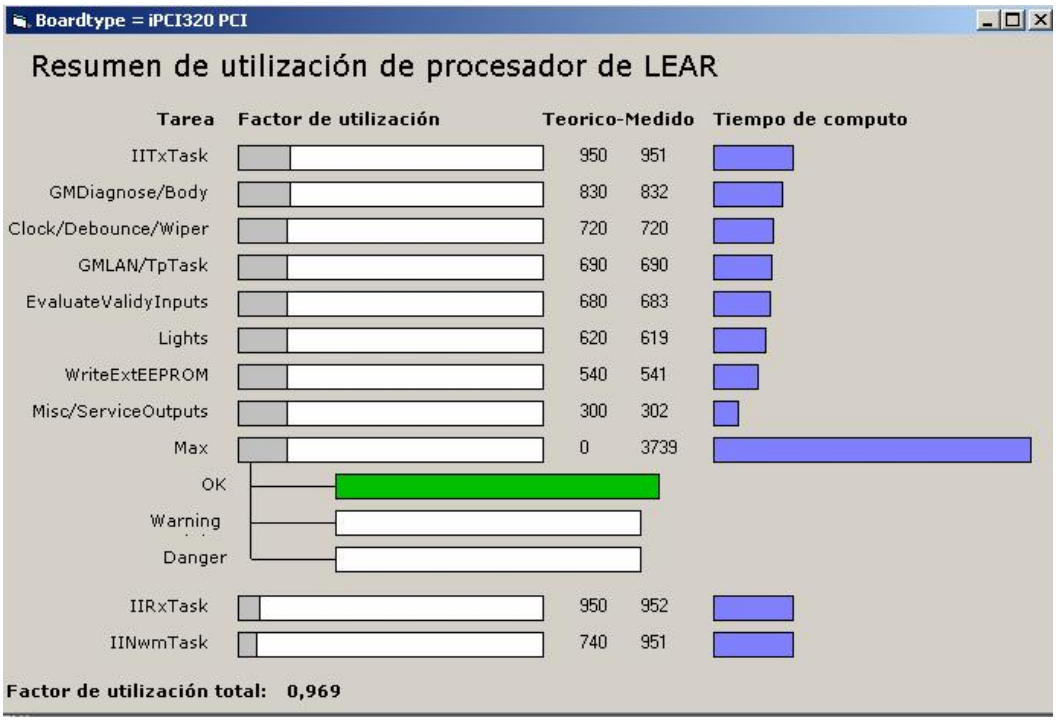


Figura 5.5. Interfaz - tarjeta electrónica PIC18F458.

CAPITULO 6

Estudio Económico

En este Capítulo se muestra el estudio económico y la planificación estimada de las diferentes tareas que componen el proyecto. El proyecto esta dividido en diferentes fases cada una compuesta por un conjunto de tareas.

Esta planificación estima el tiempo de dedicación de cada una de las fases, etapas y tareas que conforman el proyecto, proveyendo la duración total del mismo y el presupuesto esperado. También se incluye el costo de los elementos hardware utilizados en el proyecto, tales como ordenador, tarjetas de emulación, etc.

6.1. Fases del proyecto

La Figura 6.1 muestra las diferentes etapas que conforman este proyecto. A continuación se describe brevemente cada una de las tareas que componen cada una de estas fases.

Fase de análisis: Estudio y análisis del problema. Búsqueda bibliográfica relacionada con los sistemas de tiempo real y métodos de planificación de tiempo real con especial énfasis en los planificadores cíclicos. Investigación de las herramientas existentes actualmente para la planificación de sistemas de tiempo real.

Fase de diseño: Búsqueda de soluciones al problema planteado. Definición de los diferentes algoritmos utilizados para la determinación de ejecutivos cíclicos.

Fase de implementación: Implementación de los algoritmos para la determinación de ejecutivos cíclicos sobre un lenguaje de programación, específicamente el lenguaje utilizado es JAVA.

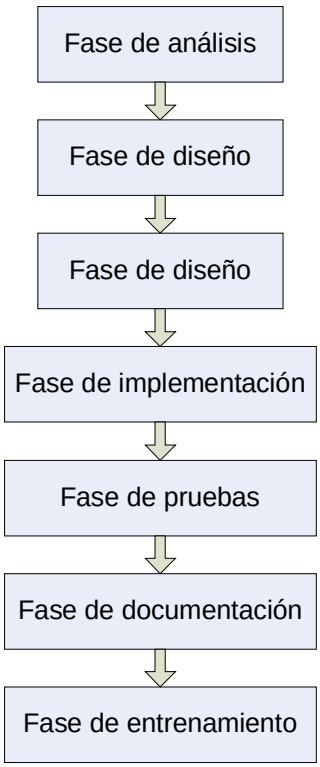


Figura 6.1. Etapas del proyecto.

Fase de pruebas: Selección y descripción de un problema real. Uso de la herramienta desarrollada sobre el problema real. Simulación del planificador cíclico utilizando la herramienta. Emulación del planificador cíclico sobre una tarjeta electrónica. Evaluación de los resultados obtenidos.

Fase de documentación: Documentación de cada una de las fases anteriores.

Fase de entrenamiento: Impartir curso de utilización de la herramienta.

6.2. Estimación del número de horas del proyecto

La Tabla 6.1 muestra el número de horas estimadas para cada una las fases realizadas.

6.3. Planificación del proyecto

La planificación del proyecto se ha realizado teniendo en cuenta una jornada laboral de 5 horas de lunes a viernes sin incluir festivos con una sola persona para desarrollar el conjunto de tareas que comprenden el proyecto.

La Figura 6.2 muestra la planificación de las diferentes tareas que componen el proyecto.

Fase	Número de Horas
Fase de análisis	90
Fase de diseño	90
Fase de implementación	90
Fase de pruebas	125
Fase de documentación	100
Fase de entrenamiento	30
Nº total de horas:	**525**

Tabla 6.1. Número de horas estimadas para el proyecto.

Id	Nombre de tarea	Duración
1	**Fase de análisis**	**18 días**
2	Estudio y análisis del problema	6 días
3	Revisión bibliográfica	12 días
4	**Fase de diseño**	**18 días**
5	Búsqueda de solucion problema planteado	9 días
6	Desarrollo de algoritmos	9 días
7	**Fase de implementación**	**18 días**
8	Implementación de los algoritmos	18 días
9	**Fase de pruebas**	**25 días**
10	Definición caso de estudio - Proceso Lear	7 días
11	Simulación del planificador cíclico	2 días
12	Emulación planificador cíclico sobre tarjeta electrónica	12 días
13	Evaluación resultados	4 días
14	Fase de documentación	20 días
15	Fase de entrenamiento	6 días

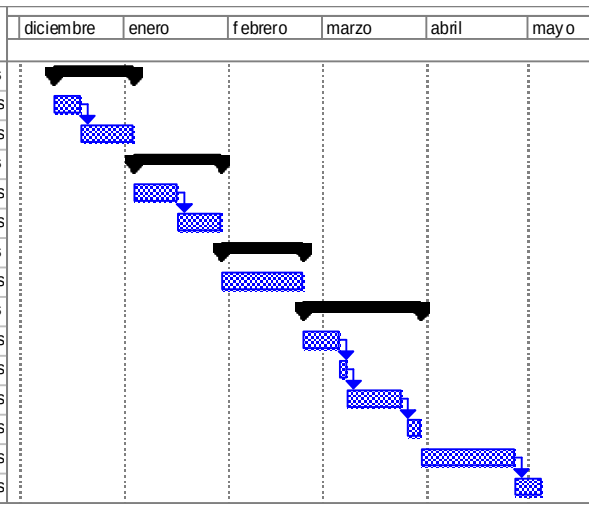


Figura 6.2. Diagrama Ghant del proyecto.

6.4. Presupuesto del proyecto

La planificación del proyecto se ha realizado teniendo en cuenta una jornada laboral de 5 horas de lunes a viernes sin incluir festivos con una sola persona para desarrollar el conjunto de tareas que comprenden el proyecto.

6.4.1. Presupuesto Hardware

La Tabla 6.2 muestra el hardware necesario para la realización del proyecto y el coste de cada uno de ellos.

Cantidad	Hardware	Presupuesto (€)
1	Ordenador Portátil	1250
2	Tarjeta electrónica PIC18F458	5000 x 2 = 1000
Coste total hardware:		2250

Tabla 6.2. Presupuesto de hardware.

6.4.2. Presupuesto recurso humano

La Tabla 6.3 muestra el presupuesto asociado al recurso humano para la realización del proyecto para cada fase. El precio estipulado por hora para un Ing. Técnico según el Departamento de ESAII de Barcelona es de 12 €/hora.

Fase	Número de Horas	Presupuesto (€)
Fase de análisis	90	1080
Fase de diseño	90	1080
Fase de implementación	90	1080
Fase de pruebas	125	1500
Fase de documentación	100	1200
Fase de entrenamiento	30	360
Coste total recurso humano:		7950

Tabla 6.2. Coste asociado al recurso humano.

6.4.3. Presupuesto total

La Tabla 6.4 muestra el presupuesto total asociado a este proyecto. Los costes indirectos relacionados con el proyecto tales como suministro eléctrico, calefacción y mantenimiento de laboratorios son tomados en cuenta en el recurso *Otros* en la Tabla 6.4.

Recurso	Presupuesto (€)
Hardware	2250
Recurso humano	7950
Otros	300
Total:	10500

Tabla 6.2. Presupuesto total del proyecto.

CAPITULO 7

Conclusiones

El desarrollo de este trabajo ha permitido estudiar el método de planificación cíclica, evaluar sus ventajas y desventajas frente a otros métodos de planificación de sistemas de tiempo real crítico y diseñar e implementar una herramienta que sirva de soporte a los diseñadores de sistemas de tiempo real.

Se ha podido comprobar que el método de planificación cíclico es un método determinista, predecible, de fácil realización y requiere una sobrecarga muy pequeña para llevar a cabo la planificación. Estas características han hecho que este método sea ampliamente usado para desarrollar sistemas de tiempo real crítico.

Sin embargo, el principal inconveniente de este método es que es poco flexible y difícil de mantener, ya que si existe un cambio en los requisitos temporales de alguna de las tareas o se añade una nueva tarea se debe rehacer toda la planificación.

En este proyecto se ha diseñado e implementado una herramienta que sirve de apoyo, en el análisis, diseño, desarrollo y mantenimiento de planificadores cíclicos de sistemas de tiempo real con un procesador. El mecanismo de diseño de esta herramienta llamada CICLIC se basa en planificadores de ciclo secundario y en algoritmos de búsqueda exhaustiva que utilizan reglas heurísticas para optimizar el camino de búsqueda de un planificador factible.

Una de las principales características de CICLIC es el uso complementario de componentes analíticos y de simulación de la herramienta durante el proceso de diseño. En primer lugar CICLIC verifica si se cumplen las condiciones suficientes de planificabilidad del conjunto de tareas, para luego determinar la planificación y la simulación de las mismas. Otra característica importante de esta herramienta es que la misma permite determinar el tiempo de cómputo máximo que una tarea puede tener, tal que la misma pueda ser añadida en una planificación factible.

Los resultados experimentales obtenidos utilizando la herramienta CICLIC son muy satisfactorios ya que se encontraron planificaciones de manera directa en la mayoría de los casos. Existen conjuntos de procesos que no admiten una planificación cíclica factible y en este caso se debe proceder a dividir algunas tareas en subtareas con tiempos de cómputo menores.

Una línea de trabajo futuro consiste en extender la herramienta para tratar conjuntos de tareas con relaciones de precedencia e introducir y evaluar nuevas reglas heurísticas que permitan aumentar la eficacia de la herramienta.

El desarrollo de este proyecto se ha basado principalmente en los conocimientos adquiridos en las siguientes asignaturas impartidas durante el segundo ciclo de Enginyeria en automàtica i Electrònica Industrial: Enginyeria de Control (ENCO), Modelat i Simulació de Sistemes (MOSS), Sistemes Informàtics en Temps Real (SITR) y Proyectes.

Bibliografía

[1] L. Alger and J.Lala. Real-time operating system for a nuclear power plant computer. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 1986

[2] T. P. Baker and A. Shaw. The Cyclic Executive Model and Ada Real Time Systems 1(1), 1989.

[3] G. Buttazzo. *Hard Real-Time Computer Systems. Predictable Scheduling Algorithms and Applications.* Kluwer Academic Publishers, 1997.

[4] J. Blazewicz et al. Scheduling in Computer and Manufacturing Systems. Springer – Verlag, 1993.

[5] J. Blumenthal, F. Golatowski, J. Hildebrandt, and D. Timmermann. *Framework for validation and Analysis of Real time Scheduling Algorithms and scheduler implementations*. University of Rostock, Technical report available from http://yasa.e-technik.uni-rostock.de/, 2003.

[6] A. Burns N. Hayes and M.F. Richardson, "Generating Feasible Cyclic Schedules," *Control Eng. Practice*, vol. 3, no. 2, pp. 151-162, 1995.

[7] Gene D. Carlow. The Architecture of the Space Shuttle Primary Avionics Software System Communications of the ACM, 27(9), 1984.

[8] The Cheddar project: A free real time scheduling analyzer, http://beru.univ-brest.fr/~singhoff/cheddar/

[9] S. Devroey, J. Goossens, and C. Hernalsteen. A generic simulator of real-time scheduling algorithms, pp. 242–249. Proceedings of the 29th Annual Simulation Symposium, New Orleans, Louisiana, April 1996.

[10] M. L. Dertouzos. Control robotics: the procedural control of physical processes. Information Processing, 74, 1974.

[11] E.W. Dijkstra. Cooperating sequential processes. In F. Genuys, editor, Programming Languages

[12] J. Flores Zamorano, "Planificación Estática de Procesos en Sistemas de Tiempo Real Crítico", Tesis Ph. D., Universidad Politécnica de Madrid, 1995.

[13] M. R. Garey and D. Johnson Computers and Intractability. A Guide to the Theory of NP-Completeness Freeman, 1979.

[14] M. G. Harbour, J. G. García, J. P. Gutierrez, and J. D. Moyano. MAST: Modeling and Analysis Suite for Real Time Applications, pp. 125–134. Proc. of the 13th Euromicro Conference on Real-Time Systems, Netherlands, June 2001.

[15] V.P Holmes, D. Harris, K. Piorkowski, and G. Davidson. Hawk: An operating system kernel for a real-time embedded multiprocessor. Technical report, Sandia National Laboratories, 1987.

[16] W. Horn. Some simple scheduling algorithms. Naval Research Logistics Quarterly, 21, 1974.

[17] M. H. Klein, T. Ralya, B. Pollak, R. Obenza, and M. G. Harbour. *A Practitioner's Handbook for Real Time Analysis*. Kluwer Academic Publishers, 1994.

[18] H. Kopetz. A Solution to an Automotive Control System Benchmark. In *IEEE Real-Time Systems Symposium*. Diciembre 1994.

[19] J. Leung and J.W. Whitehead. On the complexity of fixed priority scheduling of periodic real-time tasks. Performance Evaluation, 2(4), 1982.

[20] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of ACM*, Vol. 20, No. 1, pp. 46-61, 1973.

[21] Per Brinch Hansen. Operating System Principles. Prentice–Hall, 1973.

[22] J. Peterson, and A. Silberschatz. Operating Systems Concepts. Addison – Wesley, 1985.

[23] K. Schwan, W. Bo, and P. Gopinath. A high performance, object-based operating system for real-time robotics application. In Proceedings of the IEEE Real – Time Systems Symposium, December, 1986.

[24] SEI. The Rate Monotonic Analysis. Technical report, In the Software Technology Roadmap. http://www.sei.cmu.edu/str/descriptions/rma body.html, September 2003.

[25] TimeSys. *Using TimeWiz to Understand System Timing before you Build or Buy*. White paper, http://www.timesys.com/index.cfm?bdy=home bdylibrary.cfm, 2002.

[26] Tri-Pacific. *Rapid-RMA: The Art of Modeling Real-Time Systems.* http://www.tripac.com/html/prod-fact-rrm.html, 2003.

[27] TrueTime: *Simulation of Networked and Embedded Control Systems*. http://www.control.lth.se/truetime/

[28] J. A. Stankovic Misconceptions about real-time programming IEEE Computer. Octubre 1988

[29] J.A. Stankovic, M. Spuri, M. Di Natale, and G. Buttazzo. Implications of classical scheduling results for real-time systems. IEEE Computer, 28(6), June 1995.

[30] J. Yépez, P. Martí and J. M. Fuertes. "Control Loop Scheduling Paradigm in Distributed Control Systems." 29th Annual Conference of the IEEE Industrial Electronics Society (IECON03), Roanoke, USA, November, 2003.