Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica

Optimización del módulo de planificación de procesos del sistema operativo de tiempo real SIWA-RTOS utilizado en los nodos CRTecMote

Informe de Proyecto de Graduación para optar por el título de Ingeniería Electrónica con el grado académico de Licenciatura

Alexander Fco. Valverde Serrano

Cartago, Junio 2010

INSTITUTO TECNOLOGICO DE COSTA RICA

ESCUELA DE INGENIERIA ELECTRONICA

PROYECTO DE GRADUACIÓN

TRIBUNAL EVALUADOR

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Leonardo Rivas A.

Ing. Leonardo Rivas Arce

Profesor lector

Javier

Ing. Javier Pérez Rodríguez

Profesor lector

Ing. William Marín Moreno

Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

Cartago, 23 junio 2010

Declaratoria de Autenticidad

Declaro que el presente documento ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad total por el trabajo realizado y por el contenido del correspondiente anteproyecto.

Cartago, 23 de Junio de 2010.

Firma del autor
Alexander Fco. Valverde Serrano
Cédula: 2-0636-0538

Resumen

Este documento contempla el análisis de diferentes algoritmos de calendarización de procesos utilizados en sistemas operativos de tiempo real, con el fin de diseñar un algoritmo que reduzca el consumo de potencia del sistema operativo SIWA-RTOS utilizado en los nodos de medición CRTecMote.

Se abarca los métodos de planificación por prioridad fija (FPS), turno rotatorio (RRS), cálculo de menor holgura (LLS) y tiempo límite de entrega más próximo primero (EDFS). Además, se presenta la modificación de una herramienta de simulación de algoritmos de planificación denominada TORSHE, logrando obtener parámetros específicos relacionados con la eficiencia de cada uno de las políticas de calendarización implementadas.

La política diseñada lleva por nombre planificación por tiempo de entrega más próximo primero con escalamiento de frecuencia en estados no urgentes (EDF-DFS). Los resultados de la implementación demuestran una reducción aproximada del 11% del consumo de corriente respecto a la versión del SIWA-RTOS sin modificar al ser ejecutada sobre un microcontrolador (MCU por sus siglas en inglés) PIC32MX.

Palabras claves: Sistema Operativo de Tiempo Real (RTOS), Kernel, bajo consumo de potencia, microcontrolador, frecuencia de operación, módulo calendarizador, algoritmo de planificación de procesos.

Summary

This document describes the analysis of different process scheduling algorithms used in real time operating systems, with the purpose of designing an algorithm which reduces the power consumption of the operating system SIWA-RTOS used in the measuring nodes of CRTecMote.

The scope includes Fixed Priority Scheduling (FPS), Round Robin Scheduling (RRS), Least Laxity Scheduling (LLS) and Earliest Deadline First (EDF). Besides, the modification for a simulation tool of algorithms for scheduling named TORSHE is presented, achieving specific parameters related with the efficiency of each one of the scheduling policies implemented.

The designed policy has the name of Earliest Deadline First with Dynamic Frequency Scaling (EDF-DFS). The results of the implementation demonstrate a reduction approximately of 11% in current consumption, compared to the unmodified version of SIWA-RTOS when executing it over a microcontroller unit (MCU) PIC32MX.

Key words: Real Time Operating System (RTOS), Kernel, low power consumption, microcontroller, operation frequency, scheduler, process scheduling algorithm.

Dedicatoria

A mis padres,
que con su apoyo incondicional
me impulsaron a alcanzar esta meta.

Agradecimiento

En primera instancia, deseo agradecer a mis padres por el arduo trabajo que significó ayudarme a sacar adelante mi estudio. Por sus palabras de aliento en los momentos difíciles y por preocuparse semana a semana sobre mi estado de salud y mi avance en los proyecto y exámenes.

Seguidamente quiero agradecer a mis dos compañeros de trabajo durante la carrera Sebastián Badilla Carvajal y Heiner Alvarado Fonseca, quiénes a lo largo de estos cinco años y medio me tendieron una ayuda sólida y me brindaron la confianza para seguir adelante hasta alcanzar finalmente la meta.

Adicionalmente, tengo un profundo agradecimiento por el coordinador del proyecto Lic. Ing. Johan Carvajal Godínez, quién me tendió una mano amiga en un momento difícil de la carrera y me permitió abrir una ventana hacia nuevos ámbitos de conocimiento, y sobre todo gracias al cual hoy veo finalizado el proyecto final de graduación.

Casi terminando, deseo expresar mi gratitud a mis dos compañeros de trabajo durante el proyecto Sebastián López y Miguel Fonseca, quiénes me pusieron al día en la estructuración del sistema y siempre estuvieron a mi lado dispuestos a aclararme cualquier duda que se me presentara. Gracias por que sin su consejo no hubiese terminado el proyecto a tiempo.

Finalmente, y no por ser menos importante, agradezco a Marjorie Loría Morales quién a lo largo de estos cinco años y medio se conformó como mi fortaleza en los peores momentos. Gracias por tu apoyo, tus lindas palabras, tu preocupación, por las largas esperas y sobre todo por el cariño brindado.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capitulo 1. Introducción

1.1 Problema existente e importancia de su solución

Una red distribuida de sensores se compone por un conjunto de dispositivos capaces de realizar la medición de diferentes variables y que se encuentran separados entre sí dentro de un área definida. La ubicación de cada uno de los dispositivos de medición, denominados nodos, dependerá de las variables que se desea monitorizar. Por tanto, es posible que los nodos sean ubicados en lugares de difícil acceso para el encargado de mantenimiento.

En vista de esta situación, un requerimiento esencial de los nodos de medición CRTecMote [6] es que funcionen de forma independiente por el mayor tiempo posible. Esto evidencia el compromiso que debe alcanzarse respecto al consumo de potencia de cada nodo en un periodo determinado de tiempo.

Realizar un manejo optimizado de los recursos de hardware y software de un nodo CRTecMote permite aumentar la eficiencia respecto al uso de la energía y con ello minimizar el consumo energético del dispositivo. Dado que la administración de los recursos de un sistema operativo le corresponde principalmente al módulo de planificación de procesos, también llamado calendarizador, es importante determinar un algoritmo de planificación que minimice el consumo de potencia del sistema operativo.

1.2 Antecedentes relacionados con el proyecto

Por parte del Instituto Tecnológico de Costa Rica, la Escuela de Ingeniería Electrónica se encuentra a cargo de un proyecto de investigación que lleva por nombre “Diseño de un nodo con arquitectura abierta para redes inalámbricas de sensores: CRTecMote” [6] dirigido por el Lic. Ing. Johan Carvajal Godínez y desarrollado por un grupo de estudiantes de la Escuela de Ingeniería Electrónica. Este proyecto consiste en la implementación de una red inalámbrica de sensores (WSN por sus siglas en inglés) tal que un conjunto de dispositivos autónomos de sensado, denominados nodos, logren interactuar entre sí por medio de tecnologías de comunicación inalámbrica.

Durante la segunda mitad del año 2009, se incursionó en la optimización del sistema operativo de tiempo real usado por los nodos CRTecMote. El proyecto fue realizado por el estudiante Alexander Norwing Leiva [14] cuyos resultados permitieron obtener un sistema con estructuras de código reducidas, limitadas a las aplicaciones que serán ejecutadas de forma práctica por los nodos en una futura implementación. Adicionalmente, se implementó una mejora en cuanto al modo de consumo utilizado durante los periodos de ociosidad (idle) del microprocesador, logrando reducir el consumo de potencia en aproximadamente un 47.5% respecto al sistema original.

1.3 Solución seleccionada

La solución seleccionada busca optimizar la política utilizada por el módulo planificador de procesos. Adicionalmente se busca realizar una modificación dinámica del modo de consumo de la unidad de procesamiento (CPU) en determinadas condiciones asumidas dentro de la nueva política de planificación.

El algoritmo de planificación seleccionado fue el tiempo de entrega más próximo con aprovechamiento de estados no urgentes, el cual se obtuvo luego de una investigación comparativa de diferentes políticas utilizadas en sistemas operativos de tiempo real, basando dicha selección en los resultados obtenidos mediante la simulación de los algoritmos de planificación por prioridad fija, por turno rotatorio, por menor holgura y por tiempo de entrega más próximo.

La implementación se efectuó en la tarjeta de evaluación del PIC32MX, modificando el sistema operativo SIWA-RTOS de modo que permitiera la habilitación de ya sea la política original (esquema de prioridades) o la política seleccionada (esquema de tiempos de entrega).

Capitulo 2. Meta y objetivos

2.1 Meta

Optimizar la eficiencia de consumo de potencia de un nodo de sensado CRTecMote con microcontrolador PIC32MX.

2.2 Objetivo general

Modificar el algoritmo de calendarización de procesos del sistema SIWA-RTOS tal que permita aumentar la eficiencia de consumo de potencia de un nodo CRTecMote en un 10%.

2.3 Objetivos específicos

2.3.1 Diseñar un algoritmo de calendarización de procesos, de modo que se posibilite una reducción del consumo de potencia de un sistema operativo de tiempo real.

2.3.2 Modificar el módulo de planificación de procesos del sistema operativo en tiempo real SIWA-RTOS, para incluir el algoritmo de calendarización diseñado.

2.3.3 Determinar la reducción en el consumo de potencia, de forma tal que se pueda caracterizar el consumo energético del nodo de sensado CRTecMote.

Capitulo 3. Marco Teórico

3.1 Red distribuida de sensores

Una red distribuida de sensores (DSN por sus siglas en inglés) consiste en una amplia colección de sensores inteligentes distribuidos espacial o lógicamente en un entorno específico, interconectada a través de una red de alta velocidad, [5]. Dichos sensores pueden ser visuales, acústicos, infrarrojos, lumínicos, térmicos, sensores de humedad, presión, vibración, radioactividad, acidez, entre muchos otros. Estos sensores se encargan de recolectar de manera continua información respectiva al entorno en el que se ubican. De ese modo, la información recopilada puede ser interpretada por un elemento de procesamiento que a su vez transmite la información sobre una red.

La integración de la información recogida por los distintos nodos de medición permite obtener conclusiones apropiadas acerca del entorno en el que los sensores se ubican. Por ejemplo, recopilar información a partir de una monitorización de variables ecológicas en un bosque o incluso en el interior de una planta de energía nuclear para detectar diferentes niveles de radiación, [5].

De forma más específica, una red inalámbrica de sensores (WSN por sus siglas en inglés) corresponde a un conjunto de dispositivos autónomos de sensado, denominados nodos, que logran interactuar entre sí por medio de tecnologías de comunicación inalámbrica, [6]. La figura 3.1 permite observar como la información recopilada se transmite entre los nodos secundarios hasta un nodo principal o sumidero.

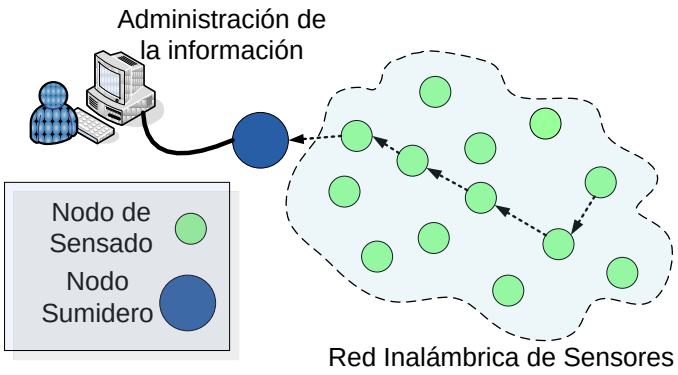


Figura 3.1 Estructuración básica de una red inalámbrica de sensores (WSN).

3.2 Descripción general de un sistema operativo en tiempo real (RTOS)

Un sistema operativo de tiempo real (RTOS por sus siglas en inglés) consiste en un sistema diseñado para mantener una serie de requerimientos dirigidos a la conclusión de tareas específicas en un tiempo máximo estrictamente fijado.

Los procesos o tareas (términos usados indistintamente en este documento) forman lo que se conocen como unidades lógicas de computación en un procesador, [1]. Una aplicación dada típicamente consiste en varios procesos. Cada proceso se maneja mediante un solo hilo de control. Aún así, para muchos procesos ejecutándose sobre un único procesador, la ejecución se realiza de forma intercalada.

En lo que respecta a los procesos o tareas que se ejecutan en un sistema de tiempo real, es posible clasificar las tareas de acuerdo a la periodicidad con la que se presentan. En este sentido, se habla de tareas periódicas (ejecutadas sobre una base de tiempo regular) y las aperiódicas (disparadas por la ocurrencia de un evento en el ambiente externo del nodo), [1].

3.3 Módulo de planificación de procesos de un sistema operativo

El módulo de planificación de procesos, también conocido como calendarizador, es un bloque dentro de la estructura de un sistema operativo que se encarga de proveer de un algoritmo o política que permite determinar el orden en el que se ejecutan las tareas pendientes en el procesador, [13].

La política se establece de acuerdo a criterios que tratan de asignar valor o prioridad a las diferentes tareas según la importancia que poseen en la consumación de los objetivos de aplicación del sistema de tiempo real.

Un algoritmo de planificación puede ser caracterizado ya sea por ser estático o dinámico, [13]. Un algoritmo estático requiere que el conocimiento de las características de las tareas que se ejecutan, como por ejemplo, la duración total de la ejecución y el valor de prioridad asociado de la misma, los cuales suelen ser asignados en el momento de la creación de la tarea. Aún así, utiliza un mínimo de tiempo de ejecución del total con que dispone el sistema.

Por otra parte, un algoritmo de planificación dinámico realiza la programación de las tareas utilizando un método de adaptación a los distintos procesos pendientes, lo cual permite un mejor desempeño en sistemas con generación de tareas no periódicas. Sin embargo, en comparación con el método estático, utiliza una mayor parte del tiempo de procesamiento dispuesto por el procesador.

Un bloque calendarizador se considera óptimo si es capaz de programar la ejecución de una serie de tareas en un tiempo menor al que otros algoritmos de planificación lo permiten [23]. Además, debe conducir a la adecuada distribución de los recursos de hardware y software del sistema entre los diferentes procesos en ejecución.

3.4 Suplantación de procesos en un sistema operativo

Un sistema de procesamiento puede caracterizarse por permitir o no la suplantación de tareas. La suplantación de tareas se refiere a que dicho sistema es capaz de detener la ejecución del proceso actual en caso de que un proceso con mayor prioridad o valor asignado ingrese en la cola de tareas preparadas. En este caso la tarea detenida pasa a la cola de tareas bloqueadas, a la espera de que el nuevo proceso termine su ejecución.

La figura 3.2 muestra el esquema de un sistema con suplantación y sin suplantación de procesos. Se observa que en un sistema con suplantación de tareas, el proceso B puede completar su ejecución después de 15 ciclos de reloj. Mientras que en un sistema sin suplantación, el proceso B debe esperar hasta que se ejecute por completo el proceso A para pasar al estado de ejecución, en este caso le tomaría 19 ciclos de reloj completar el cálculo.

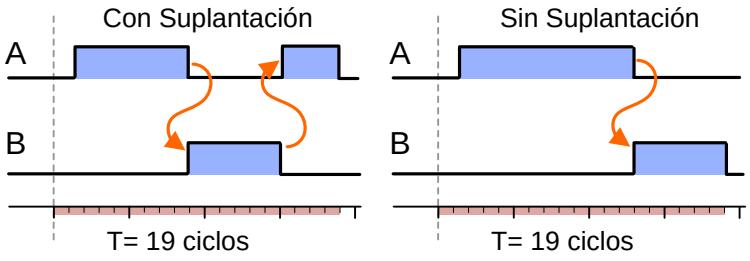


Figura 3.2 Esquema de un sistema con suplantación y sin suplantación de procesos.

3.5 Algoritmos de planificación de procesos

3.5.1 Método de suplantación por prioridad fija

En un sistema que utiliza un esquema de suplantación de procesos es posible programar el orden de ejecución de las tareas según el valor de prioridad asignado a cada una. Dicha prioridad en este caso se asigna de forma estática, es decir que dicho valor de prioridad no puede ser modificado por el núcleo de procesamiento.

El método de suplantación por prioridad fija busca completar la ejecución anticipada de aquellos procesos que poseen un valor de prioridad mayor dentro de la lista de tareas preparadas. En la literatura se le puede encontrar como *fixed priority pre-emptive* o *static priority pre-emptive*, [2, 12].

Usualmente la asignación más alta de prioridad corresponde al valor de 1, mientras que conforme la prioridad decrece el valor asignado aumenta. Si bien este método de programación de procesos permite completar de manera adecuada la ejecución de las tareas con mayor prioridad de ejecución, no prevé que hacer con las tareas que presentan muy baja prioridad. En este sentido, una tarea con muy baja prioridad, respecto a las demás en la lista de procesos preparados, puede nunca pasar al estado de ejecución o bien hacerlo después del tiempo máximo esperado para su finalización.

La figura 3.3 ilustra la planificación de este método de calendarización de procesos. Se presentan tres procesos A, B y C, los cuales posee una prioridad fija de 3, 2 y 1, asignada desde el momento de su creación respectivamente.

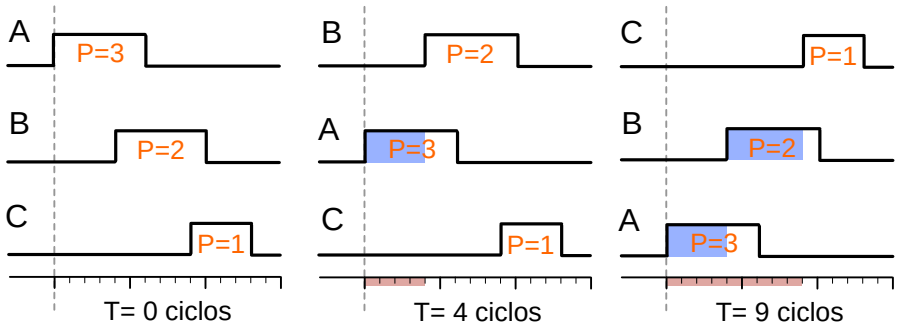


Figura 3.3 Diagrama del método de suplantación por prioridad fija.

Inicialmente, un proceso con baja prioridad (A) inicia su ejecución debido a que no existe un proceso con mayor prioridad asignada en la lista de tareas preparadas. No obstante, 4 ciclos de reloj después, un proceso con prioridad 2 (B) entra en la lista de preparados y suplanta al proceso A, el cual mantiene su final de ejecución pendiente. El proceso B es ejecutado hasta que un proceso de alta prioridad (C) ingresa a la lista de preparados. De ese modo el proceso C pasa al estado de ejecución y los procesos A y B incompletos esperan hasta que termine para continuar su ejecución.

3.5.2 Método de "Tasa de ejecución uniforme"

El método de tasa de ejecución uniforme es una técnica común de asignación dinámica de prioridades a procesos. Es posible hallar información adicional con el nombre de *Rate-Monotonic Algorithm*, [12]. Consiste en asignar una mayor prioridad a las tareas con tasa de ejecución periódica más alta, [4]. La asignación de prioridades mediante esta técnica permite maximizar la planificabilidad de toda una aplicación. Sin embargo, las variaciones en el tiempo de ejecución hacen de los cálculos del valor de prioridad un proceso complejo.

3.5.3 Método de "Tiempo de entrega más próximo primero"

Este método corresponde a un algoritmo dinámico de asignación de prioridad utilizado en sistemas con suplantación de procesos. También se le hace referencia como *Earliest Deadline First (EDF)* o *Deadline Driven Scheduling Algorithm*, [1, 21]. Este método asigna la prioridad correspondiente a un proceso dependiendo de la cercanía respecto al tiempo máximo esperado para su finalización. Es decir, la tarea que amerite una entrega más pronto será la favorecida con una prioridad mayor.

En la figura 3.4 se puede observar el orden en el cual tres procesos, cuyo tiempo máximo de entrega es el único parámetro variante, son ejecutados. En primera instancia, el proceso A es el que debe ser entregado con mayor prontitud por lo que se le asigna la mayor prioridad y es ejecutado luego de 4 ciclos de reloj. Pasados esos 4 ciclos de ejecución, la comparación se realiza respecto a los procesos B y C donde C es el que posee un menor tiempo para el límite de entrega. De ahí que se le asigna la prioridad más alta al proceso C, obligando al proceso B a esperar hasta la finalización del mismo para alcanzar el estado de ejecución.

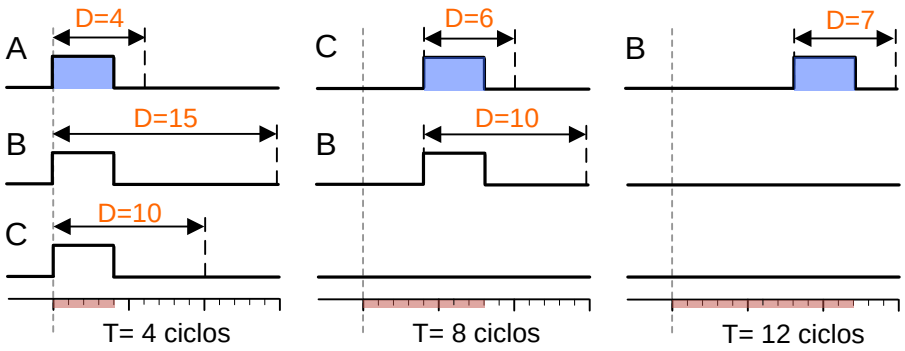


Figura 3.4 Diagrama del método de tiempo de entrega más próximo primero.

Cabe destacar que el ejemplo anterior se asume que la solicitud de ejecución de los tres procesos se realiza en el mismo instante. Aún así, en un sistema con suplantación de tareas, es posible que en cualquier instante salga un proceso del estado de bloqueo con un tiempo límite de entrega menor al del proceso en el estado de ejecución, en cuyo caso el nuevo proceso suplantaría al actual.

Este algoritmo resulta óptimo en el sentido que permite alcanzar una utilización del 100% del tiempo de procesamiento [16]. Es por ello que algunos autores recomiendan el uso de este algoritmo de asignación de prioridad incluso en sistemas sin suplantación de procesos [1].

3.5.4 Método de “Menos holgura primero”

Esta técnica corresponde a un algoritmo de asignación dinámica de prioridades sobre procesos utilizado por sistemas con suplantación de tareas. En la literatura disponible también se le refiere como *Least Slack First* o *Least Laxity First*, [1]. A continuación se encuentra la explicación de la técnica propuesta.

La holgura de un proceso se define como el tiempo máximo esperado para su finalización (“deadline”) menos el tiempo de computación restante para terminar el cálculo. En base al enfoque de menor holgura primero (LSF por sus siglas en inglés), el proceso que tiene la menor holgura se le asigna la mayor prioridad en el sistema y por ende es ejecutado. Mientras un proceso está siendo ejecutado puede ser suplantado por uno cuya holgura ha disminuido por debajo del que se está corriendo actualmente, [25].

La ecuación 3.1 permite determinar el margen de tiempo de holgura $H_i(t)$ de un proceso conocido. Donde $D_i(t)$ representa el tiempo máximo esperado para la finalización de la tarea, $C_i(t)$ el tiempo de computación restante para terminar el cálculo y $T_i(t)$ el tiempo transcurrido desde que se solicita la tarea, [25].

$$H_i(t) = D_i(t) - T_i(t) - C_i(t) \tag{3.1}$$

Como se aprecia en la figura 3.5, se presentan dos procesos A y B tal que el tiempo máximo esperado para su finalización (D) es de 15 ciclos de reloj. Los procesos A y B son conocidos y se sabe que el tiempo total requerido por el procesador para completar el cálculo (C) es de 10 y 5 ciclos de reloj, para los procesos A y B respectivamente. Además, se observa que el proceso A posee una menor holgura (H) respecto al proceso B, por lo que al proceso A se le asigna la prioridad más alta y es ejecutado. Aún así, cuando el tiempo transcurrido (T) alcanza los 6 ciclos de reloj, se observa que la holgura del proceso B ha descendido durante la ejecución del proceso A, por lo que B obtiene una prioridad mayor que A. De ese modo, B suplanta al proceso A y es ejecutado. En este caso, nótese que un proceso en ejecución mantiene tiene una holgura constante puesto que son los demás procesos los que se ven desplazados durante la espera.

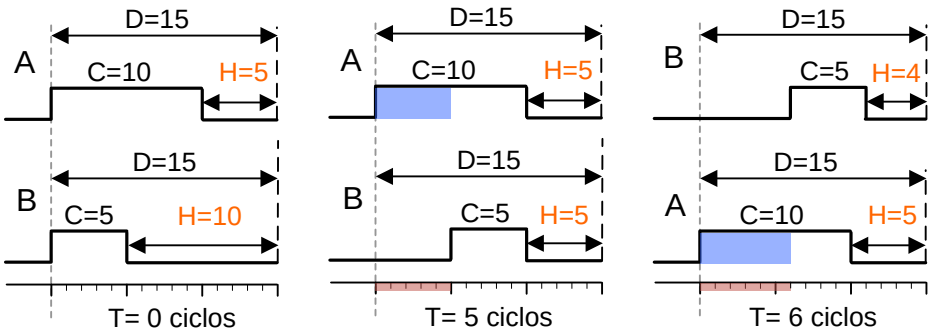


Figura 3.5 Diagrama del método de menos holgura primero.

La calendarización basada en el enfoque de menor holgura es muy útil en sistemas comprendidos principalmente por procesos no periódicos, puesto que no se realizan suposiciones respecto a la tasa de ocurrencia de los eventos. Aún así, al igual que la técnica de "Tiempo de entrega más próximo primero", este algoritmo permite alcanzar una utilización del 100% del procesador, [16].

3.5.5 Método de planificación “primero en llegar primero en salir”

Este método representa una técnica sencilla y de fácil implementación que permite programar la ejecución de una determinada lista de tareas en un sistema operativo. También se encuentra en otros documentos como “First Come First Served”, [18]. Como su nombre lo indica, el método consiste en asignar igual prioridad a todos los procesos y destinar el uso de la unidad de procesamiento en el orden en el que ingresaron a la lista de tareas preparadas para la ejecución, como se aprecia en la figura 3.6.

Entre los beneficios de su utilización, además de lo simple de su implementación, se encuentra que se encuentra libre de inanición, es decir que todos los procesos poseen la posibilidad de ser ejecutados en un tiempo determinado. Aún así, no permite el cumplimiento de los tiempos máximos esperados para la finalización de todos los procesos por lo que no es común que sea utilizado en sistemas operativos de tiempo real.

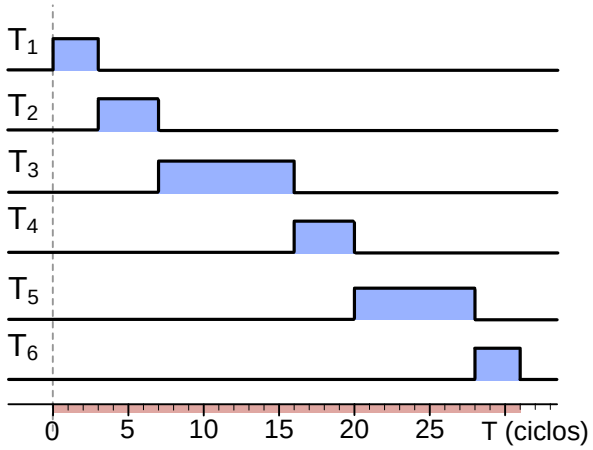


Figura 3.6 Diagrama del método de planificación primero en llegar primero en salir.

3.5.6 Método de planificación por “turno rotatorio”

El método de planificación por turno rotario, al igual que el anterior, es uno de los métodos de calendarización más simples respecto a su implementación. Considera la lista de tareas preparadas como una lista circular de procesos y busca la ejecución de cada tarea por periodos fijos. Es posible hallar información adicional referenciada como planificación “Round Robin”, [14].

La técnica descrita consiste en la ejecución de un conjunto de tareas asignando a cada una un periodo fijo de tiempo de ejecución, denominado *quantum*. El procesador asume que todos los procesos tienen asignada una misma prioridad por lo que ejecuta cada tarea hasta agotar el tiempo predefinido (quantum), [18]. Cuando un proceso agota su quantum es suplantado por el proceso siguiente en la lista de tareas preparadas. Este proceso se realiza de forma continua recorriendo la lista de forma circular.

La figura 3.7 ilustra la aplicación de este método de planificación sobre un conjunto de tareas con diferente duración. Como puede observarse, para el mismo conjunto de tareas de la figura 3.6, se fija un quantum de 3 ciclos de reloj. De este modo, el procesador ejecuta cada proceso según el orden en el que ingresaron a la lista de tareas preparadas (T_1, T_2, T_3, T_4, T_5 y T_6) por un tiempo máximo de 3 ciclos de reloj. Luego del vencimiento del quantum, el proceso siguiente en la lista suplanta al actual y así en lo sucesivo. Es importante notar que cuando un proceso que termina su ejecución antes de agotar el quantum obliga al sistema a continuar con el siguiente proceso en la lista.

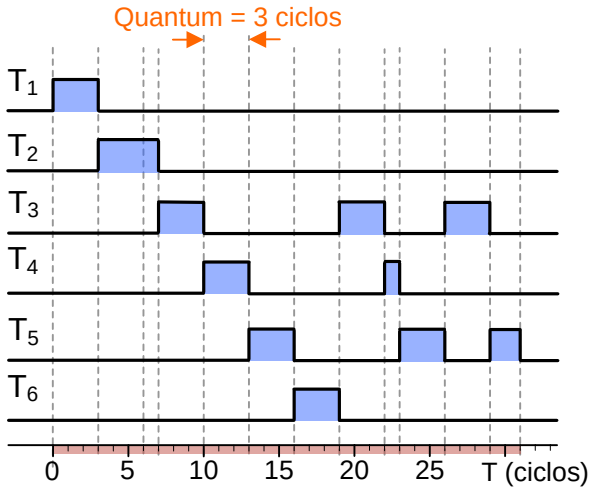


Figura 3.7 Diagrama del método de planificación por turno rotatorio.

3.6 Escalamiento dinámico de modo de consumo del CPU

Esta técnica es ampliamente utilizada para reducir el consumo de energía en sistemas operativos de tiempo real. Es posible hallar referencias como *Dynamic Voltage Scaling* (DVS, por sus siglas del inglés) por diferentes autores [10, 22].

Consiste en variar ya sea la tensión o la frecuencia de trabajo de la unidad de procesamiento en determinados instantes de la ejecución. Sin embargo, si bien al disminuir estos parámetros se logra reducir el consumo de potencia, una variación prolongada provoca un incremento en la latencia de ejecución de las tareas y con ello la pérdida de algunos de los tiempos máximos de entrega establecidos.

La clave de esta técnica recae en elegir adecuadamente los intervalos en los que se producirá la variación del modo de consumo, [22]. Es por ello que suele aplicarse esta técnica en conjunto el algoritmo de planificación de procesos del sistema operativo.

Existen diversas técnicas propuestas para llevar a cabo este proceso. A continuación se detallan dos algoritmos utilizados para llevar a cabo la variación dinámica de la potencia consumida:

1. Método de cálculo de probabilidades de Gorjiara, [10]

2. Método de aprovechamiento de estados no urgentes, [22]

3.6.1 Método de cálculo de probabilidades de Gorjiara (ASG-VTS)

Esta técnica de calendarización de procesos realiza una planificación simultánea tanto del conjunto de tareas presentes como de diferentes modos de consumo asociados a cada tarea separadamente. Se puede encontrar referencias sobre este algoritmo como "*ASG-VTS*".

En este sentido, Bita Gorjiara propone el cálculo de la probabilidad de pasar el sistema a un modo de consumo inferior (SDP, por sus siglas en inglés) basado en las variaciones de la energía total consumida, $E_T(M)$, y un factor de retraso proporcional a la variación del sistema al nuevo modo de consumo supuesto.

De manera complementaria se obtiene la probabilidad de pasar un modo de consumo más alto (SUP, por sus siglas en inglés) debido a la necesidad de aumentar el rendimiento del sistema. La ecuación (3.2) muestra la relación entre las dos probabilidades calculadas para una tarea en específico.

$$SUP(\tau_i) = 1 - SDP(\tau_i) \tag{3.2}$$

Una vez determinada la probabilidad de variar el modo de consumo basado en la tarea ejecutada actualmente, se fija un valor de consumo ya sea mayor o menor de forma aleatoria y se verifican los requerimientos de tiempos de entrega máximos existentes.

Para llevar a cabo el cumplimiento de las limitaciones temporales se calcula la penalización temporal que representa el cambiar a un modo de consumo inferior dado que un modo de consumo inferior implica una reducción de la frecuencia de operación de la unidad de procesamiento. A su vez, un modo de consumo mayor permite aumentar la frecuencia de operación y con ello acelerar el procesamiento del conjunto de tareas.

3.6.2 Método de aprovechamiento de estados no urgentes

Esta técnica es propuesta por [22] y pretende realizar una planificación dinámica de los recursos disponibles en determinados puntos de la ejecución del calendarizador de procesos. El algoritmo propuesto hace uso del método de calendarización "Tiempo de entrega más próximo primero", el cual resulta ser ampliamente utilizado en aplicaciones de tiempo real.

La técnica consiste en aprovechar los estados donde el CPU ejecuta ya sea la tarea de más baja prioridad (idle) o bien la última tarea de la cola de tareas listas para bajar el modo de consumo del sistema. En este sentido el planificador determina el tiempo restante para que la tarea más próxima en la cola de procesos bloqueados pase a la cola de procesos listos. Luego, compara ese tiempo con el tiempo restante de ejecución y tiempo límite de entrega de la tarea actualmente en ejecución, logrando adecuar el nivel de voltaje, o bien de frecuencia, a uno inferior de modo que se logren cumplir los requerimientos de entrega de la tarea actual.

3.7 Sistema operativo de tiempo real SIWA-RTOS

En el sentido de obtener un sistema operativo que se ajustara a las necesidades de cada nodo CRTecMote se utiliza un sistema operativo de tiempo real llamado SIWA-RTOS, el cual resultó de una modificación realizada a un sistema de libre distribución llamado FreeRTOS™, [14].

El FreeRTOS™ es un sistema operativo de código abierto, diseñado para sistemas empotrados adaptable a diferentes plataformas. Se distribuye bajo una Licencia General

Pública de GNU (GNU GPL, por sus siglas en inglés) con una excepción opcional, la cual establece que puede ser utilizado para aplicaciones propietarias siempre y cuando se mantenga el núcleo como fuente abierta, [9].

Actualmente todos los trabajos realizados en el proyecto CRTecMote utilizan el SIWA-RTOS como base para su desarrollo. Es decir, todas las aplicaciones y nuevas funciones se implementan sobre dicho sistema y no sobre la versión original FreeRTOS.

La política de planificación de procesos utilizada por el sistema SIWA-RTOS es conocida como "planificación de turno rotatorio con prioridad fija". Éste algoritmo asigna una prioridad fija a cada tarea desde el momento de su creación, la cual no puede ser modificada por el núcleo de procesamiento (conocido como Kernel). Además, opera sobre un núcleo que permite la suplantación de tareas, es decir que una tarea con alta prioridad que entra al estado de procesos listos para ejecución puede sacar del estado de ejecución actual a cualquier otra que presente una prioridad menor, en este sentido se dice que la nueva tarea "suplanta" a la actual, [4].

El módulo planificador utilizada diferentes colas en el manejo del estado de procesos listos. Cada cola mantiene procesos son un mismo valor de prioridad asignado, lo cual permite ejecutar una política de planificación por turno rotatorio sobre cada una de las listas de forma independiente. Una vez agotados los procesos de una determinada cola el calendarizador inicia la planificación de los procesos en la siguiente cola de menor prioridad asignada. Además, se especifica un valor de quantum de 1ms que determina el intervalo de tiempo entre conmutaciones de contexto de las tareas.

3.8 Herramienta de simulación TORSHE

TORSHE (Time Optimization of Resources, Scheduling) corresponde a un conjunto de herramientas de programación para el entorno de MATLAB que ha sido desarrollado por el Departamento de Ingeniería de Control (Universidad Técnica Checa de Praga, Facultad de Ingeniería Eléctrica) y se distribuye bajo los términos de la GNU GPL (General Public License).

Esta herramienta contiene funciones de generación de gráficos que permiten al diseñador apreciar de forma clara la forma en la que cada algoritmo de planificación se comporta. Adicionalmente, al estar programada en el lenguaje orientado a objetos de MATLAB permite establecer de antemano las características propias asociadas al conjunto de tareas que se desea estudiar.

Una herramienta de este tipo posibilita la obtención de código optimizado que define el comportamiento sobre la administración de recursos de sistemas que se ejecutan sobre arquitecturas de hardware específico. A su vez es útil para realizar el estudio de las cualidades favorables y limitaciones que presentan diferentes algoritmos de planificación de procesos ante un escenario genérico respecto a otros.

En la figura 7 se ilustra las características configuradas para cada proceso, el ancho corresponde al tiempo de procesamiento, la flecha hacia arriba indica el inicio del evento que da inicio al proceso, la flecha hacia abajo es el tiempo límite de entrega de la tarea (deadlines), en el eje vertical se detalla el nombre de las tareas y en el horizontal la escala de tiempo en ciclos de procesamiento.

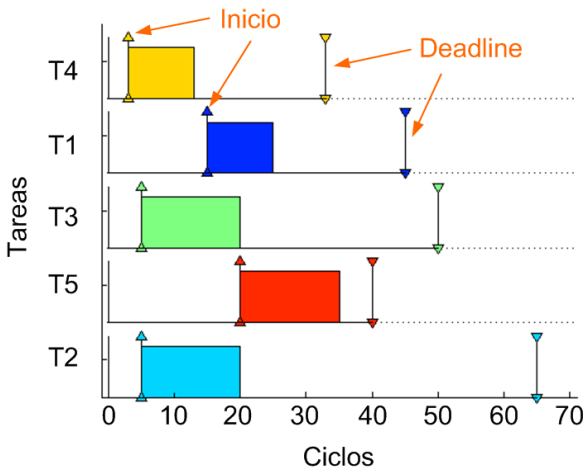


Figura 3.8 Conjunto de tareas ilustrativo del algoritmo FIFO.

En diagrama de la figura 8 presenta la salida obtenida a partir del entorno de simulación TORSHE para el conjunto de 5 tareas de la figura 7, utilizando el algoritmo de planificación "primero en entrar primero en salir (FIFO)". Se observa la incorporación de tareas adicionales para el estado de ociosidad y conmutación de contextos entre tareas.

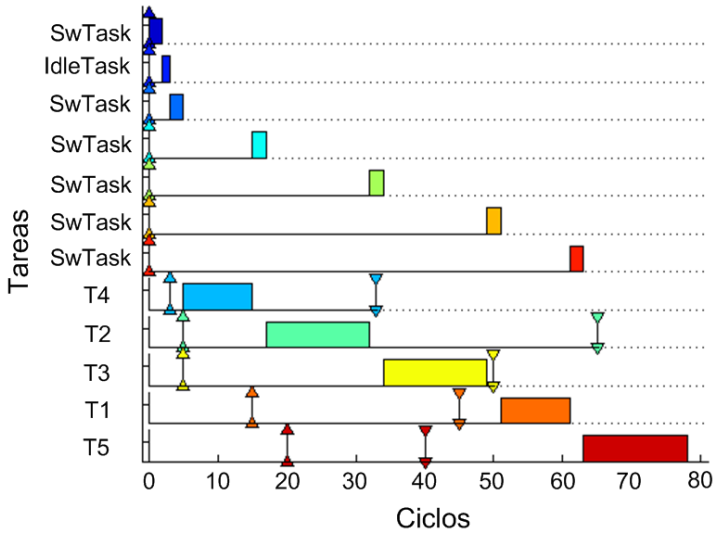


Figura 3.9 Resultado de planificación obtenido usando la herramienta TORSHE según el algoritmo FIFO.

Capitulo 4. Procedimiento Metodológico

4.1 Métodos y actividades

4.1.1 Determinación de las características propias de un sistema operativo de tiempo real

Se abordó el estudio de las características propias de los sistemas operativos de tiempo real, logrando determinar los rasgos distintivos respecto a los sistemas operativos comunes. Además, se indagó acerca de las aplicaciones en las que suelen ser utilizados.

La investigación realizada abarcó material publicado por diferentes autores que han utilizado o bien que actualmente hacen uso de este tipo de sistemas operativos en sus aplicaciones comerciales [3, 8, 19, 20].

4.1.2 Síntesis de algoritmos de planificación de procesos

Se elaboró una investigación detallada respecto a la manera en la que se diseña e implementa el bloque de planificación de procesos en un sistema operativo. Sumado a esto, se estudió la estructuración del módulo calendarizador en el sistema operativo de tiempo real SIWA-RTOS, logrando identificar el algoritmo de planificación utilizado.

Luego, se investigó acerca de diferentes algoritmos de calendarización utilizados en sistemas operativos. En este sentido, se hizo énfasis en los algoritmos utilizados en sistemas de tiempo real.

4.1.3 Modificación de la herramienta TORSHE

Para llevar a cabo una selección válida respecto al algoritmo de planificación óptimo que debería implementarse en el sistema SIWA-RTOS, se indagó acerca de herramientas de simulación [7]. A partir de la investigación se encontró que existía una herramienta de libre distribución llamada TORSHE, la cual es utilizada para simular políticas de calendarización ante diferentes escenarios.

Luego, al estudiar la documentación existente así como las funciones de software provistas por el grupo de diseño de TORSHE, se descubrió que la gama de escenarios propuestos en la herramienta no contenían el utilizado por el sistema SIWA-RTOS. Fue entonces que se procedió a modificar la herramienta TORSHE de modo que se contemplara el esquema de suplantación de tareas en sistemas con una única unidad de procesamiento.

Adicionalmente, se implementó algunos de los algoritmos de calendarización en la herramienta. Primeramente se implementó el método de planificación por turno rotatorio (round robin) y después otros 3 métodos de los expuestos en el marco teórico.

Luego se procedió a depurar las modificaciones efectuadas y se introdujo dentro de la planificación los casos en los que ya sea el módulo planificador o la tarea de más baja prioridad (idle) se ejecutan.

4.1.4 Selección del algoritmo de planificación a implementar

Se llevó a cabo la simulación de los cuatro métodos de planificación más relevantes dentro de los estudiados. Se utilizó para ello un conjunto de 22 tareas que representaban el comportamiento periódico de 3 tareas durante 7 ciclos de procesamiento.

A partir de las simulaciones y la investigación realizada se identificaron los parámetros críticos que afectan el consumo de un sistema operativo basado en el módulo de planificación y la forma en la que éste se diseña. Fue así que se introdujeron elementos de recolección de información estadística dentro de la implementación de los algoritmos en el simulador de modo que se obtuvieran parámetros específicos de comparación.

Se desplegaron estadísticas respecto al desempeño de cada uno de los algoritmos gracias a las modificaciones introducidas en la herramienta TORSHE. A partir de las cuales se logró seleccionar el método de planificación de procesos óptimo para el sistema, quedando en evidencia que el método de “tiempo de entrega más próximo primero” resolvía de forma más adecuada el problema propuesto, bajo el esquema utilizado por el sistema SIWA-RTOS.

4.1.5 Modificación del módulo de planificación de procesos del SIWA-RTOS

Para llevar a cabo la implementación del algoritmo de planificación seleccionado se estudió la documentación correspondiente al FreeRTOS™ provista por el diseñador en [4] ya que este sistema se tomó como base en la implementación previa que dio origen al sistema SIWA-RTOS.

Seguidamente, se abarcó la documentación disponible sobre el SIWA-RTOS, incluido el informe final de proyecto de graduación elaborado por Norwing Leiva, con el fin de adquirir el conocimiento práctico respectivo a la estructuración del sistema actual.

Fue así que se determinó la estructuración de las diferentes rutinas que conforman el bloque de planificación de procesos de SIWA-RTOS así como los algoritmos utilizados, el manejo que se hace de la memoria, las interrupciones, el gestor de recursos, entre otros.

Luego, se procedió con la implementación del algoritmo de planificación de procesos utilizando la herramienta de programación y síntesis MPLAB. En esta fase del proyecto, se hizo uso de las herramientas de depuración provistas por el entorno de desarrollo.

Se diseñó una aplicación que contemplaba un conjunto específico de tareas ejecutables por el SIWA-RTOS original y se inició la etapa de pruebas de funcionamiento del sistema modificado ejecutando la misma aplicación. Esto con el fin de validar que la versión modificada permitía la misma funcionalidad que la original.

Durante la validación de la modificación realizada se hizo uso de las opciones de depuraciones habilitadas en el entorno de desarrollo de MPLAB con el fin de corroborar el correcto funcionamiento del sistema después de modificación así como la ejecución correcta del algoritmo implementado.

4.1.6 Programación de rutina de escalamiento de frecuencia de operación en SIWA-RTOS

En este punto se inició las pruebas de escalamiento dinámico de frecuencia del CPU como complemento al algoritmo de planificación implementado. El algoritmo propuesto

utiliza los estados no urgentes durante la planificación de los procesos para realizar un cambio del modo de consumo de potencia.

Para llevar a cabo el cambio dinámico de frecuencia de operación se debió estudiar detalladamente las especificaciones provistas por el fabricante del PIC32MX. De modo tal que se contemplaran todos los aspectos relevantes para un adecuado ajuste dinámico del sistema.

Primero, se creó una rutina que realizaba la conmutación entre dos frecuencias específicas. Luego, cuando se depuró el comportamiento del sistema ante los cambios de la frecuencia de operación, se migró la rutina a una librería alternativa de modo que el diseñador pueda utilizarla de forma global.

4.1.7 Cuantificación del consumo de potencia de un nodo CRTecMote antes y después de la modificación del algoritmo de planificación

Una vez terminada la fase de implementación se dio inicio a la fase de caracterización del consumo del sistema. Para ello se hizo uso de una unidad de alimentación-medición (SMU, por sus siglas en inglés) provista por la Escuela de Electrónica como parte de un programa incipiente de investigación sobre aplicaciones de nanotecnología.

Para las pruebas se utilizó una tarjeta de evaluación DM320003 que incluye un microcontrolador PIC32MX, específicamente el PIC32MX460F512L, conectada con una tarjeta de expansión PIC32 I/O Expansion Board. El conjunto de elementos conforman la base de hardware de un nodo CRTecMote.

Con respecto al sistema original se realizaron pruebas de consumo fijando diferentes frecuencias de operación, lo cual permitió caracterizar el consumo promedio de corriente en función de la frecuencia. En este sentido, se hizo uso de funciones matemáticas sobre dichos datos, tales como la determinación de la media aritmética y la creación de filtros que permitieron excluir todas aquellas muestras dispersas que no contribuían con la información adquirida de la caracterización. Con los datos recolectados, se realizó un gráfico que muestra claramente el consumo de potencia del sistema cuando ejecutaba el RTOS original.

De forma complementaria, se elaboró la caracterización del consumo promedio de corriente en función de la frecuencia para el sistema con la modificación del módulo planificador. Los resultados se filtran y se muestra gráficamente el comportamiento demostrado por el sistema luego de la modificación.

Finalmente, se calcula el porcentaje de incremento de la duración promedio de descarga de una batería de 9V al servir de alimentación del sistema para cada una de las políticas de planificación soportadas.

Capitulo 5. Descripción detallada de la solución

5.1 Aspectos generales

Para la solución del problema se hizo uso del MCU PIC32MX460F512L, el cual cuenta con 9 modos diferentes para el manejo de potencia así como la capacidad de realizar conmutación de frecuencia durante su estado de operación [17]. De ese modo se posibilitó el obtener un sistema capaz de consumir menor cantidad de corriente en diferentes circunstancias de operación.

En cuanto a la parte de software, se utilizó el sistema SIWA-RTOS 1.0 como sistema base inicial para las modificaciones que se desarrollaron a lo largo de este proyecto con el objeto de obtener un sistema operativo que presentara un menor consumo de potencia.

5.2 Determinación del algoritmo de planificación a implementar

5.1.1 Síntesis de algoritmos de planificación de procesos

Se realizó la síntesis de varios algoritmos utilizados en la planificación de procesos en sistemas operativos basada en la investigación efectuada en la etapa inicial del proyecto. En total se logró sintetizar 6 algoritmos diferentes los cuales se detallan en la sección 3.5 correspondiente al marco teórico de este documento.

La síntesis se realizó con base en diferentes fuentes bibliográficas que permitieron entender en detalle la manera en la que estos se implementan. A continuación se citan los algoritmos sintetizados:

- Método de “suplantación por prioridad fija”
- Método de “Tasa de ejecución uniforme”
- Método de “Tiempo de entrega más próximo primero”
- Método de “Menos holgura primero”
- Método de planificación “primero en llegar primero en salir”
- Método de planificación por “turno rotatorio”

5.1.2 Modificación de la herramienta TORSHE para la simulación de algoritmos de planificación de procesos

Para llevar a cabo la simulación de los algoritmos de planificación de procesos se propuso utilizar herramienta de libre distribución llamada TORSHE. La elección es esta herramienta se debió al amplio conjunto de funciones de graficación y numeroso conjunto de problemas (escenarios) que permite resolver.

Aún así, durante la etapa de estudio de la documentación disponible así como las pruebas realizadas sobre el código fuente brindado por el diseñador de la herramienta, fue posible determinar que el escenario propio del sistema SIWA-RTOS no era representado por ninguno de los abarcados en la herramienta.

Algunas de las limitaciones encontradas en la herramienta TORSHE son:

- No permite realizar la suplantación de procesos durante la ejecución de un conjunto de tareas sobre un único procesador.

- Realiza la planificación completa de forma inmediata del conjunto de tareas de entrada, sin considerar aspectos básicos como el tiempo en el que ocurre el evento que da inicio a la tarea. Es decir, planifica incluso tareas que no han ocurrido aún.

- Durante la planificación no introduce ningún tipo de penalización asociada a la cantidad de veces que debe ejecutarse el módulo calendarizador. Es decir que no impone pérdidas temporales asociadas con la conmutación entre contextos de diferentes tareas.

- No realiza un manejo de colas de procesos. En este caso haciendo referencia a la cola de procesos en estados *listo* y *bloqueado* del sistema SIWA-RTOS.

- No involucra la ejecución de una tarea de baja prioridad durante los ciclos en los que no hay tareas que ejecutar, estado de ociosidad del sistema.

Es así que se modificó el código fuente de la herramienta para incluir el escenario característico de SIWA-RTOS, el cual cuanta con un sistema con una sola unidad de procesamiento capaz de llevar a cabo suplantación de tareas, manejo de colas de procesos, representación del estado de ociosidad del sistema y ciclos de procesamiento consumidos durante la ejecución del módulo planificador.

La programación de la herramienta se encontraba en el lenguaje orientado a objetos de MATLAB por lo fue necesario investigar los manuales disponibles de diferentes autores para llegar a dominar las estructuras programadas y hacer uso de ellas.

Para el manejo de colas de procesos en MATLAB se crearon dos funciones especiales que permiten agregar y remover objetos *tarea* de arreglos de tamaño variable (colas de procesos). En las figuras 5.1 y 5.2 se presentan los diagramas de operación de dichas funciones.

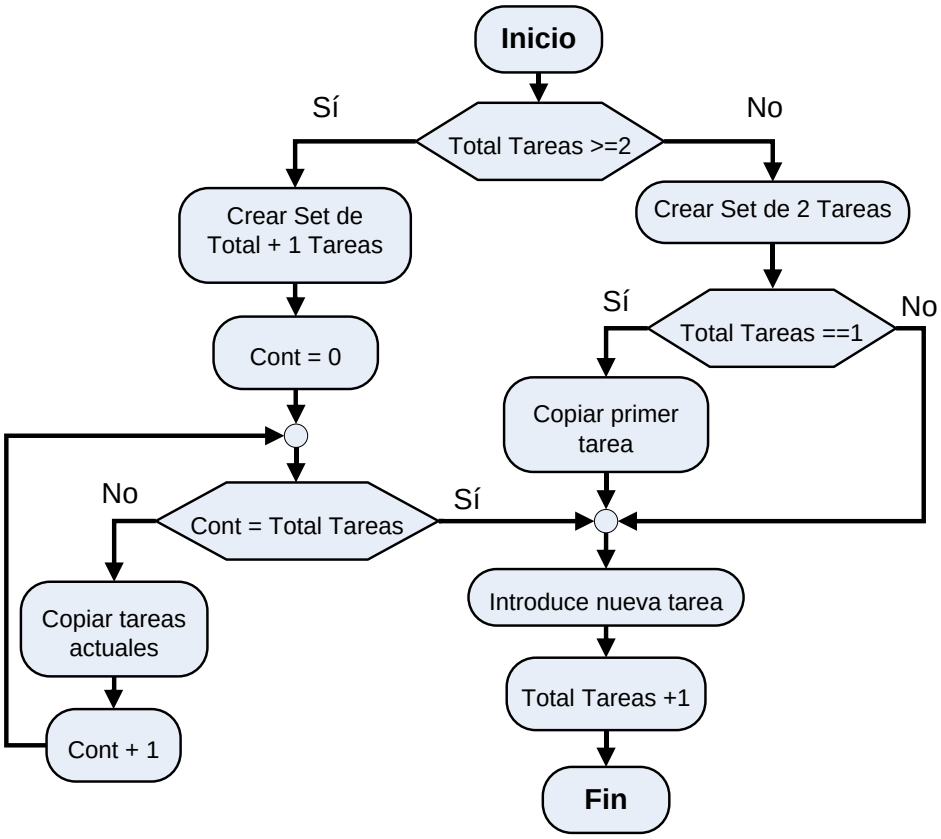


Figura 5.1 Diagrama respectivo a la función de agregar nueva tarea al final, implementada para la herramienta TORSHE.

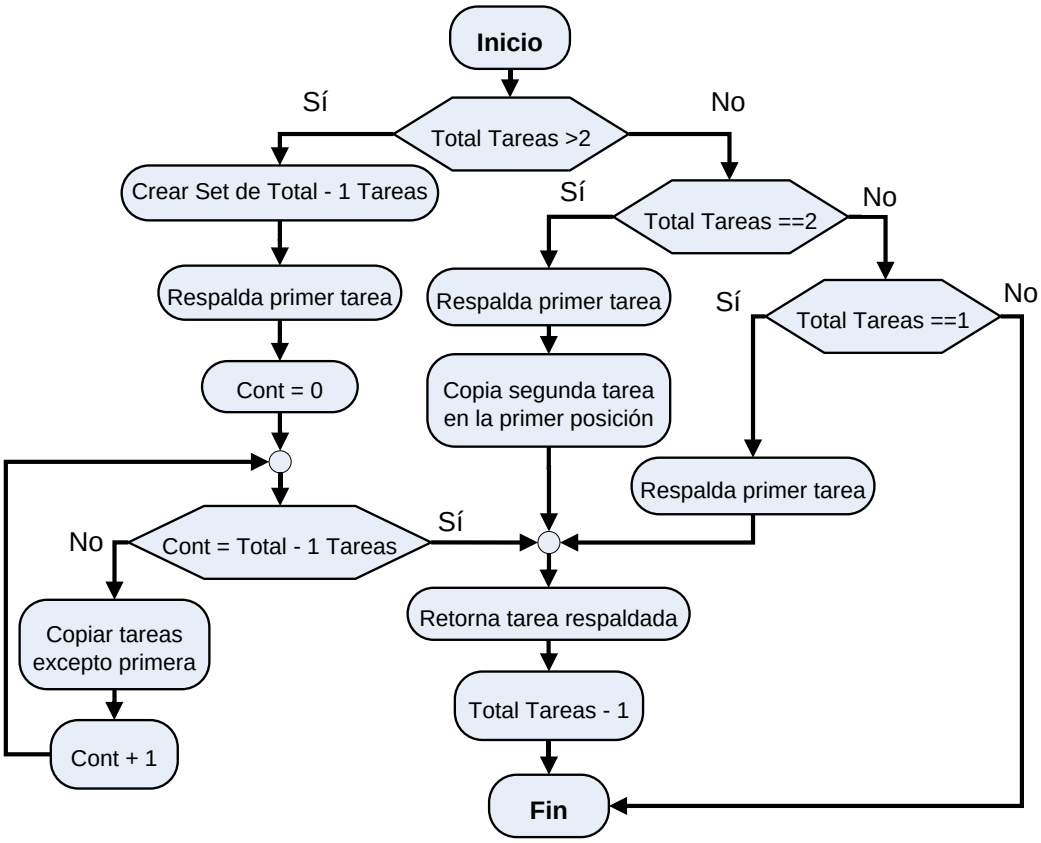


Figura 5.2 Diagrama respectivo a la función de eliminar tarea al inicio, implementada para la herramienta TORSHE.

Con respecto al manejo de colas de procesos *listos* y *bloqueados*, se implementó una función encargada de determinar cuales procesos dentro de la cola de tareas bloqueadas se encuentran habilitados para pasar a la cola de tareas listas. Esta función se ejecuta con cada ciclo de ejecución asegurando que las tareas entren en el estado de listos sin retrasos adicionales lo cual permite que el proceso de suplantación no presente retardos. La figura 5.3 presenta el diagrama ilustrativo de la programación de esta función.

La actualización de colas de procesos permite a su vez una planificación coherente de las tareas en lo que se refiere al tiempo de inicio asignado a cada una de ellas. De ese modo se corrigió el problema relacionado con la planificación de forma inmediata del conjunto de tareas de entrada, considerando ahora el tiempo en el que ocurre el evento que da inicio a la tarea. Es decir, se planifica únicamente aquellas tareas que han sido alcanzadas por el contador de ciclos del sistema.

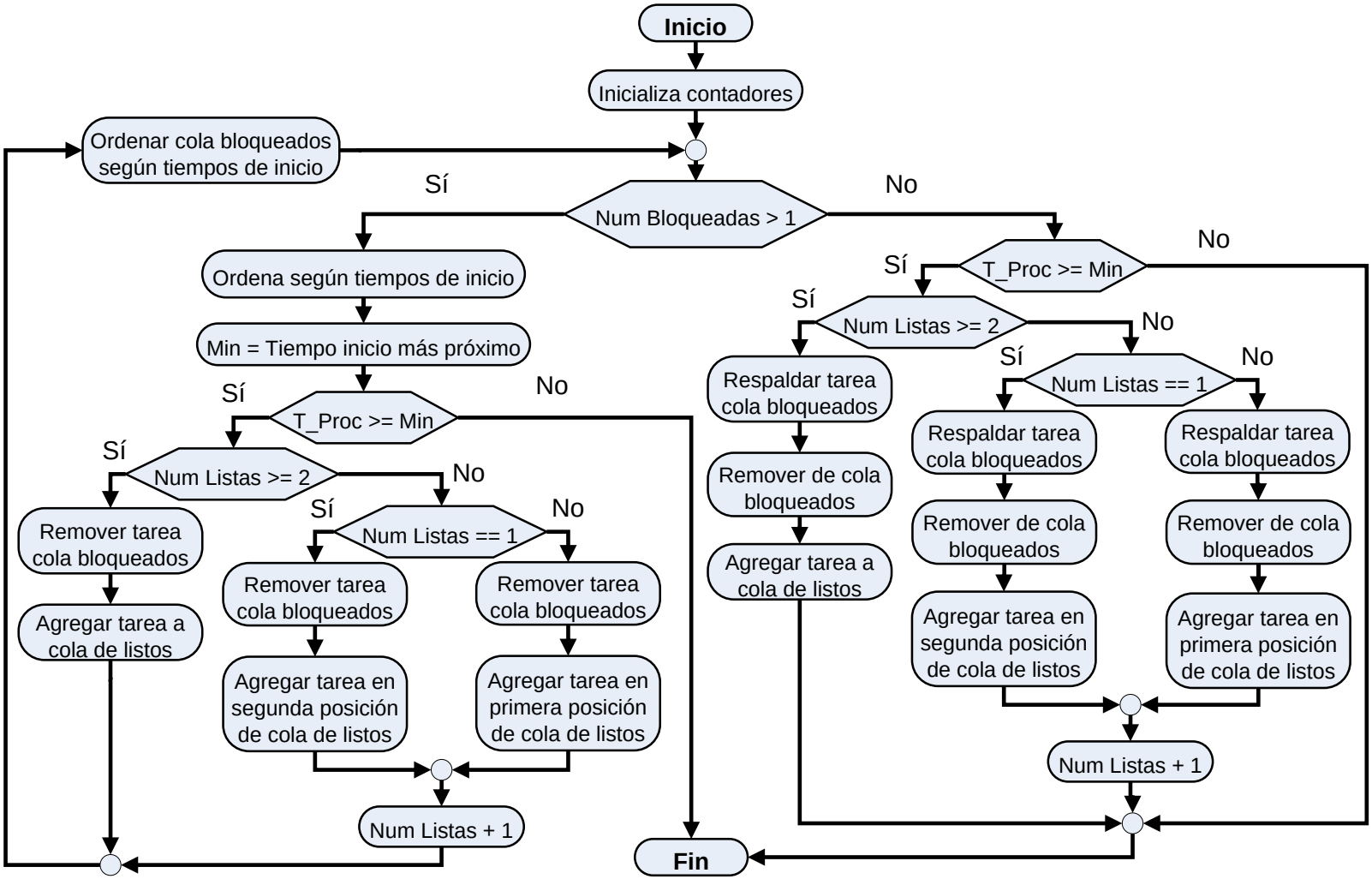


Figura 5.3 Diagrama respectivo a la función de actualización de colas de procesos bloqueados y listos, implementada para la herramienta TORSHE.

5.1.3 Simulación de algoritmos de planificación de procesos

En lo que respecta a la simulación de los algoritmos de planificación de procesos se implementaron en total 4 algoritmos de los sintetizados en la primera fase del proyecto. De estos, dos fueron seleccionados por representar la política utilizada en el sistema SIWA-RTOS original y los otros dos por acentuar las características propias de un sistema de tiempo real como lo es el cumplimiento estricto de los tiempos de entrega. Los algoritmos implementados son los siguientes:

- Método de "suplantación por prioridad fija"
- Método de planificación por "turno rotatorio"
- Método de "Tiempo de entrega más próximo primero"
- Método de "Menos holgura primero"

Se realizó el diseño e implementación dentro del ambiente de desarrollo de MATLAB de los cuatro algoritmos citados anteriormente. Para ello se utilizó las funciones de adición y remoción de tareas así como la función de actualización de colas expuestas en la sección anterior.

Durante la implementación de los algoritmos se tomó en cuenta que la ejecución de todo módulo planificador en un sistema operativo acarrea la pérdida de una determinada cantidad de ciclos de procesamiento. Es por ello que para representar de forma apegada a la realidad el resultado de la planificación, se introdujo una penalización de 2 ciclos de procesamiento por cada ejecución del módulo planificador dentro del algoritmo programado.

Adicionalmente, y como requisito fundamental para representar un escenario similar al del sistema SIWA-RTOS, este algoritmo se estructuró sobre una base funcional que busca habilitar la suplantación de procesos durante la simulación.

El diagrama mostrado en la figura 5.4 pretende ilustrar la programación implementada en la herramienta TORSHE, respectiva al método de planificación por prioridad fija. Puede apreciarse que el algoritmo analiza de forma iterativa el conjunto de tareas presente en las colas de procesos *listos* y *bloqueados* hasta dar por terminada la planificación de los procesos solicitados.

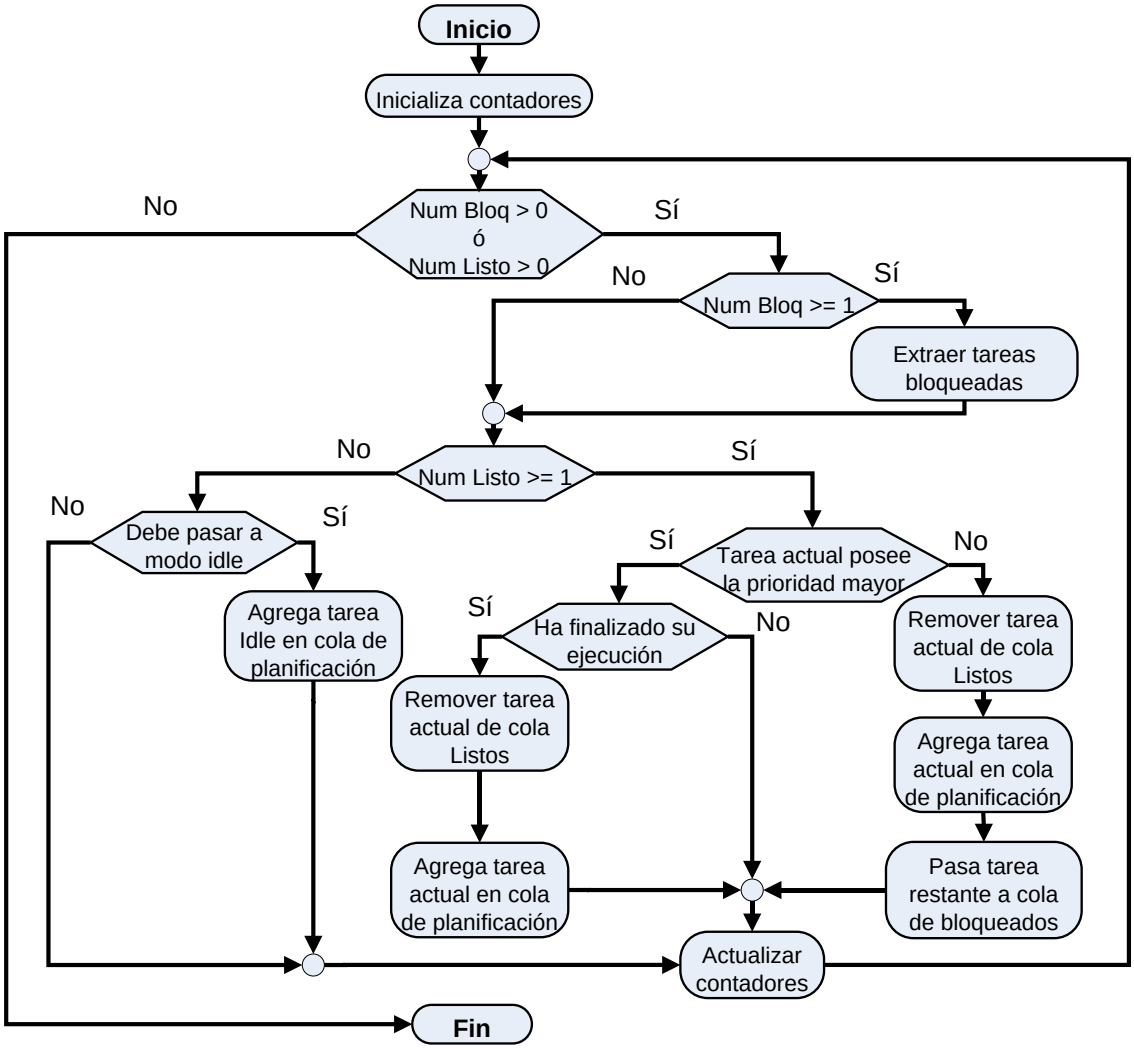


Figura 5.4 Diagrama respectivo al algoritmo de planificación por prioridad fija, implementado para la herramienta TORSHE.

La figura 5.5 presenta el diagrama correspondiente al algoritmo de planificación de procesos por turno rotatorio, más conocido como *round robin*, para la simulación a partir de la herramienta TORSHE.

A diferencia del algoritmo presentado anteriormente, esta implementación no basa su ejecución sobre la base funcional que permite llevar a cabo la suplantación de procesos. La razón recae en la propia naturaleza del algoritmo el cual, al manejar una base constante de conmutación entre tareas (conocida como *quantum*), mantiene una repartición equilibrada del tiempo de procesamiento.

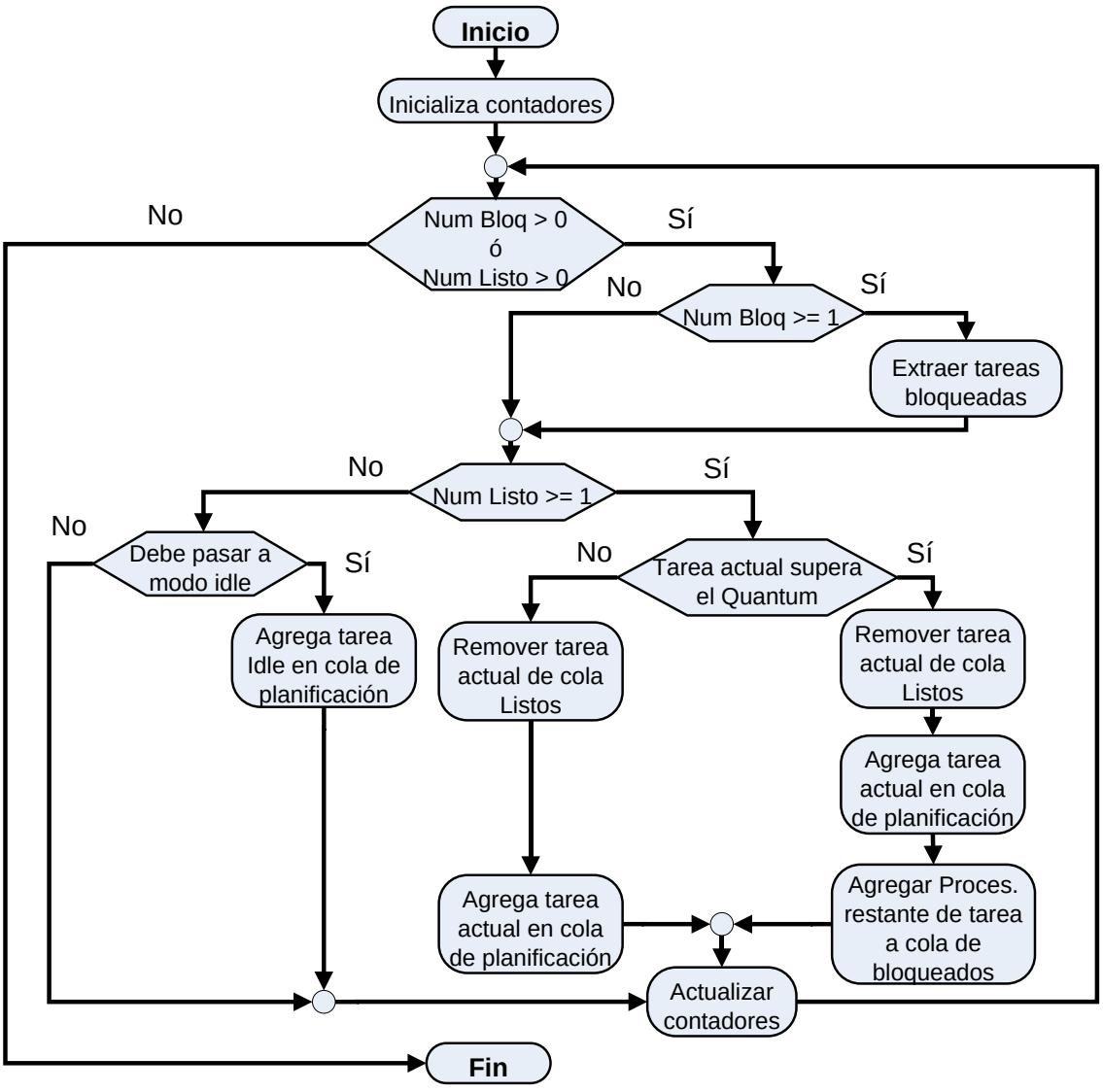


Figura 5.5 Diagrama respectivo al algoritmo de planificación por turno rotatorio, implementado para la herramienta TORSHE.

Los dos métodos anteriores simulan algoritmos de planificación de procesos utilizados por la mayoría de sistemas operativos. Ambos buscan lograr la planificación del total de tareas asignadas. No obstante, al tratarse de un sistema de tiempo real, es necesario abarcar políticas de planificación que tengan como meta el cumplimiento oportuno de los tiempos de entrega asociados a cada proceso.

La figura 5.6 ilustra la implementación del algoritmo de planificación basado en un cálculo de la holgura con la que cada tarea se ejecuta. Al igual que la implementación del método de prioridad fija, este algoritmo utiliza la misma base funcional que permite la suplantación de tarea durante su ejecución.

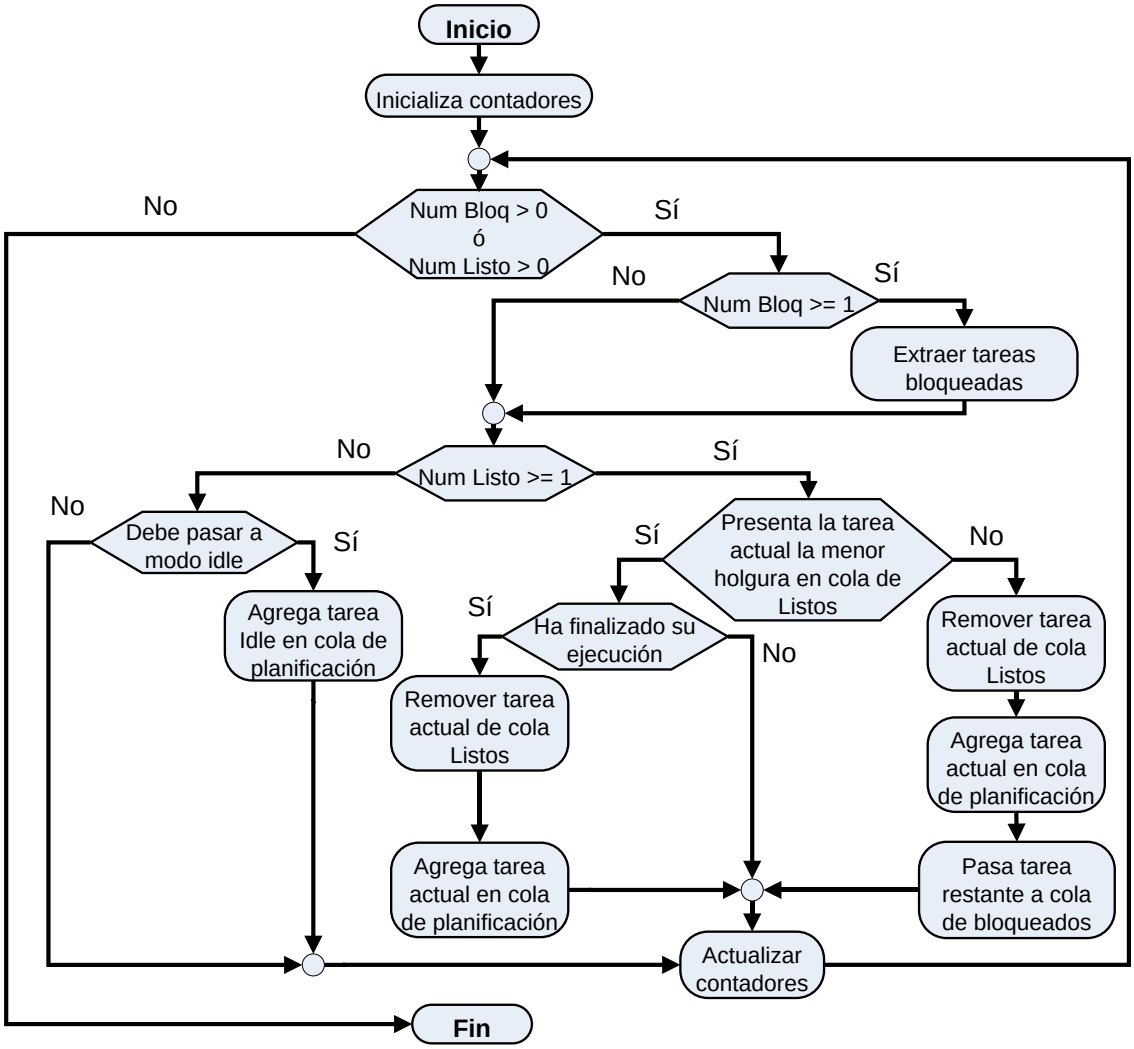


Figura 5.6 Diagrama respectivo al algoritmo de planificación por menor holgura, implementado para la herramienta TORSHE.

El cuarto algoritmo programado en la herramienta TORSHE se basó en el método de planificación del “tiempo de entrega más próximo primero”. Este método al igual que el anterior tiene como objetivo lograr que la planificación contemple primero aquellos procesos cuya fecha límite de entrega (referido como *deadline*) tiene menor margen de vencimiento. La estructura implementada se muestra en la figura 5.7.

Llegado a este punto, es importante resaltar que en todos los algoritmos implementados, la penalización asociada al cambio de contexto entre tareas al ejecutarse el módulo planificador se maneja de manera intrínseca al agregar una tarea en la cola de planificación.

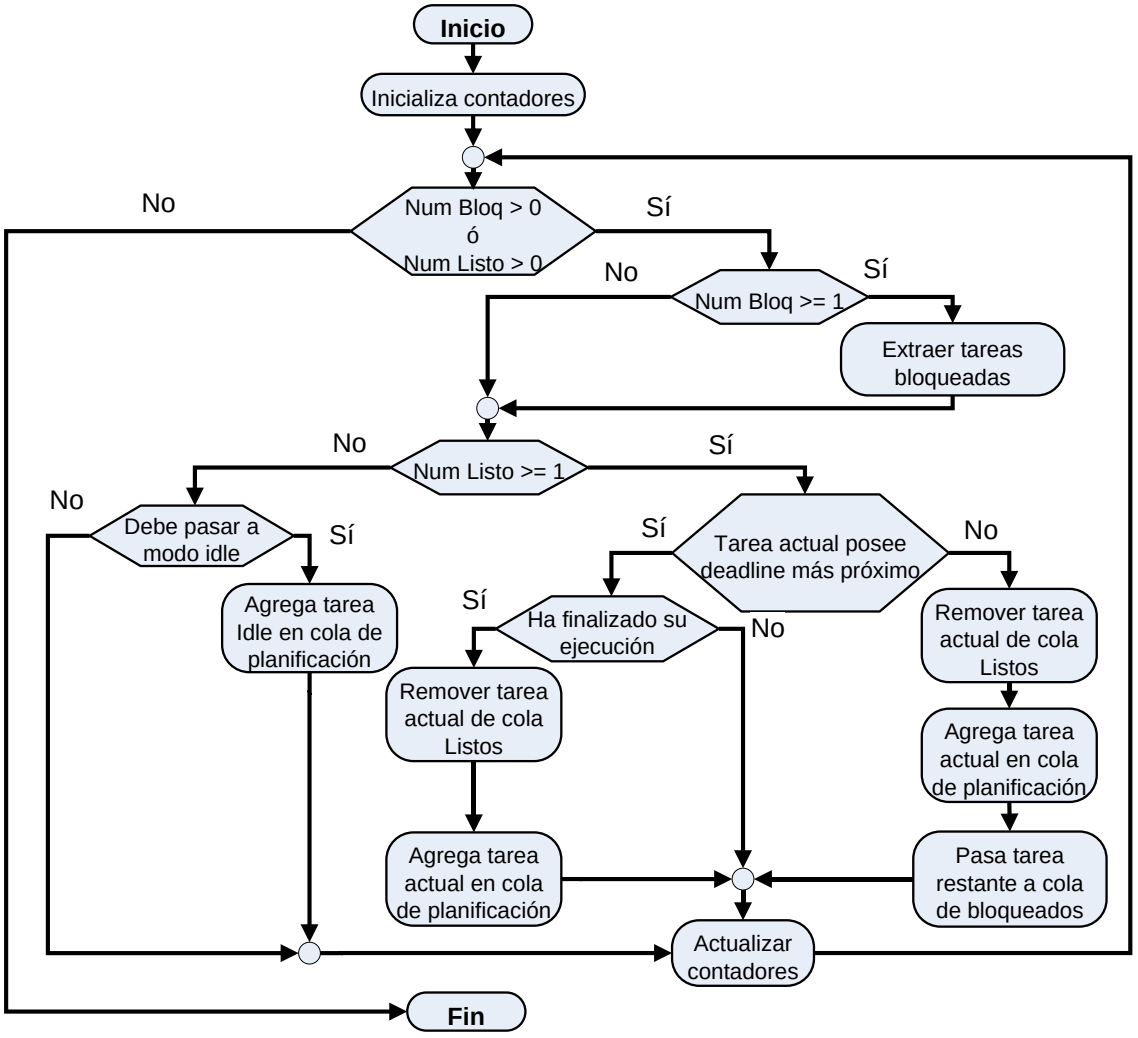


Figura 5.7 Diagrama respectivo al algoritmo de planificación por tiempo de entrega más próximo, implementado para la herramienta TORSHE.

5.1.4 Caracterización del consumo de potencia según la política de calendarización utilizada

La caracterización se realizó con base en los resultados obtenidos mediante la simulación de los cuatro algoritmos implementados en la herramienta TORSHE. La ecuación (5.1) permite aproximar el consumo de potencia del sistema al ejecutar cada una de las políticas de calendarización.

$$P_{normal} \approx (T_{Total} - T_{Idle}) \cdot M_{\text{Alto}} + (T_{Idle}) \cdot M_{\text{Bajo}} + (N_{Conmutaciones}) \cdot P_{\text{Cntx}} \quad (5.1)$$

Los parámetros de la ecuación (5.1) se obtienen de la información estadística recopilada de la simulación, donde T_{Total} se refiere a la cantidad total de ciclos requeridos para llevar a cabo la planificación completa del conjunto de tareas; T_{Idle} es la cantidad de ciclos que se invirtieron en estado de ociosidad del procesador; $N_{Conmutaciones}$ corresponde a la cantidad total de intercambios de contexto provocadas por la ejecución del módulo planificador; M_{Alto} es una constante asociada al consumo promedio de energía al operar el sistema en modo de consumo normal (frecuencia de reloj del CPU al máximo) y M_{Bajo} es una constante asociada al modo de bajo consumo (reloj interno del CPU deshabilitado).

Por otra parte, la ecuación (5.2) permite aproximar el consumo del sistema al implementar la técnica especificada en la sección 3.6.2 respecto a la utilización de un modo de consumo intermedio cuando se ejecuta un estado no urgente en la planificación. Los parámetros de la ecuación (5.2) son los mismos que los especificados en la ecuación (5.1) a excepción de T_{Ultimo} y M_{Med} los cuales corresponden a la cantidad de ciclos consumidos en una condición no urgente y una constante ligada al modo de consumo intermedio (frecuencia de reloj del CPU intermedia), respectivamente.

$$P_{frecuencia} \approx (T_{Total} - T_{Idle} - T_{Ultimo}) \cdot M_{\text{Alto}} + (T_{Ultimo}) \cdot M_{\text{Med}} + (T_{Idle}) \cdot M_{\text{Bajo}} + (N_{Conmutaciones}) \cdot P_{\text{Cntx}} \quad (5.2)$$

El factor P_{Cntx} corresponde a una constante asociada con la sobrecarga debida a la conmutación entre contextos de tareas con cada ejecución del módulo planificador. Una mayor cantidad de conmutaciones, $N_{Conmutaciones}$, provoca un aumento proporcional de la potencia consumida por el sistema.

5.3 Modificación de la estructura del módulo planificador en SIWA-RTOS

Se utilizó el método de planificación por tiempo de entrega más próximo, el cual fue seleccionado a partir de la simulación de diferentes algoritmos al ejecutar un conjunto de tareas periódicas. Para ello se asumió la posibilidad de asociar la ejecución de cada proceso con una periodicidad específica en el sistema SIWA-RTOS. Es así que el valor del periodo es tomado inicialmente como el tiempo límite de entrega de cada tarea, siendo posible modificar el mismo a consideración del futuro diseñador.

En primera instancia, se definió el tipo de política de planificación en la librería de configuración del sistema "*FreeRTOSConfig.h*" de ese modo existe la flexibilidad de elegir el algoritmo de planificación que se desea utilizar en el arranque del sistema. De manera conjunta, se hizo uso de las directivas de preprocesamiento aceptadas por el compilador C32 de Microchip® con la intención de que no generar estructuras de programación inútiles al cambiar de una política de planificación a otra.

La política original realiza el manejo de 5 colas de procesos, una por cada valor de prioridad definido. En este sentido se modificó dicho esquema para manejar una única cola de procesos ordenada a partir de los tiempos de entrega máximos. Para ello fue necesario modificar las funciones de inicialización de colas de procesos así como la forma en la que las tareas son agregadas a las mismas.

La figura 5.8(a) muestra la lógica original de la función de inserción de procesos en una cola específica. Se puede observar que la inserción se realizaba al final de cada cola, esto se debe a que cada cola manejaba procesos con un mismo valor de prioridad asignado por lo que no resultaba necesario establecer ningún orden durante la inserción. Por otra parte, en la figura 5.8(b) se presenta el esquema establecido durante la modificación para llevar a cabo el manejo de una sola cola de procesos junto con el método de ordenamiento respecto al valor de tiempo límite de entrega asignado a cada tarea.

Adicionalmente, se modificó el esquema utilizado por la función de cambio de contexto entre tareas del sistema SIWA-RTOS, como se muestra en la figura 5.9(a), tal que en lugar de buscar los procesos en diferentes colas, se limitara a obtener el primer elemento de una única cola ordenada por tiempos límite de entrega, como se observa en la figura 5.9(b).

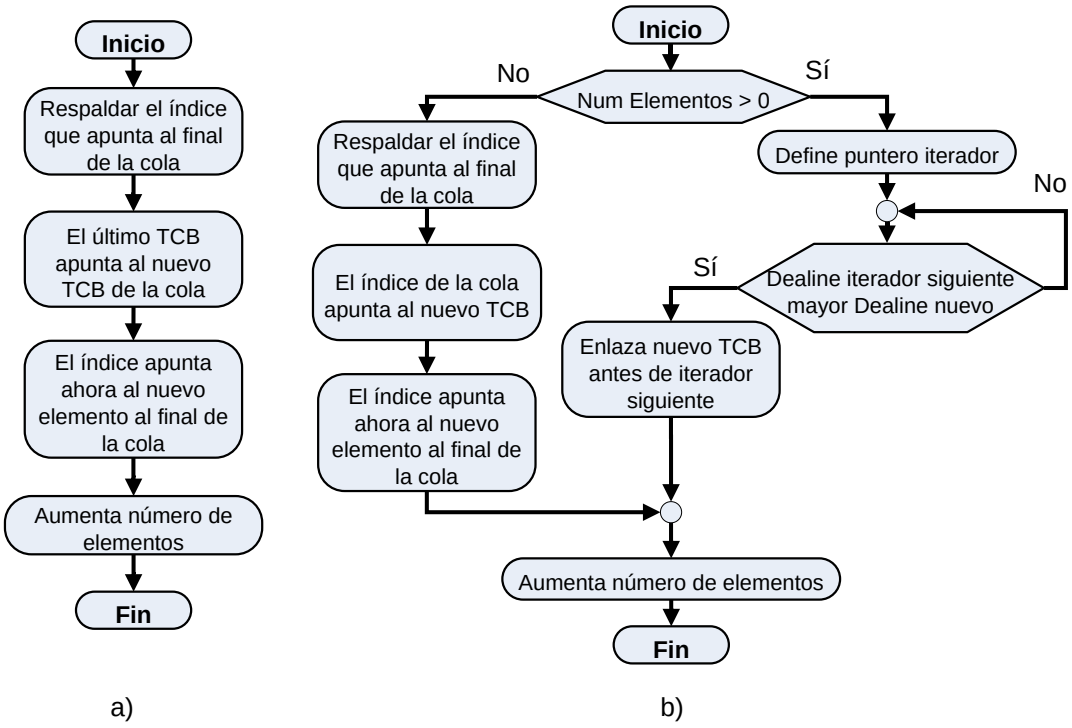


Figura 5.8 Diagrama respectivo a las funciones de inserción de tareas a) al final de la lista y b) ordenada respecto al tiempo límite de entrega.

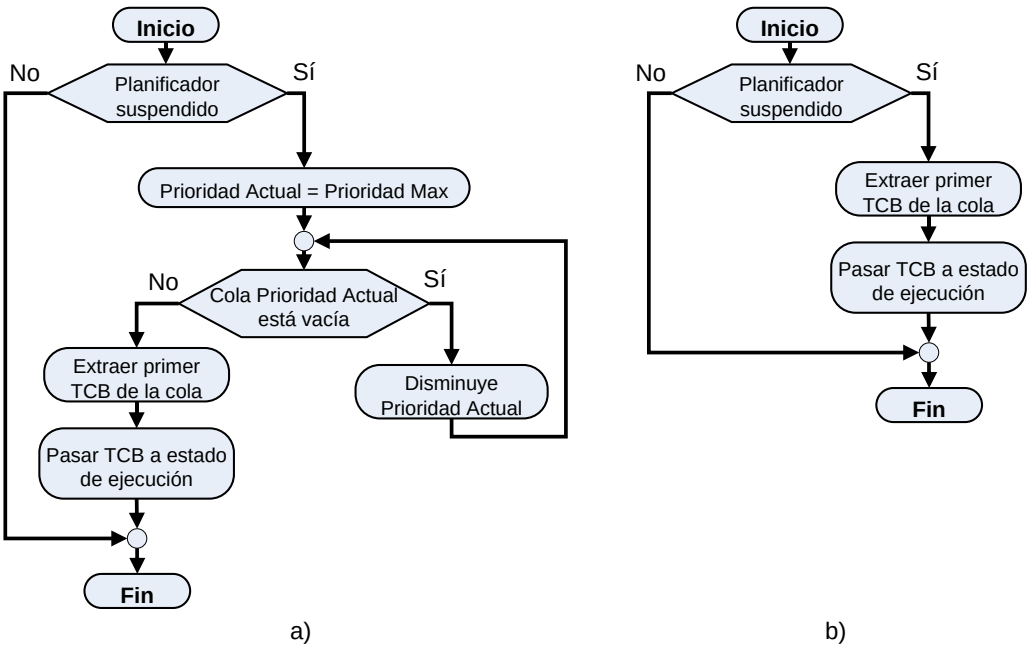


Figura 5.9 Diagrama respectivo a las funciones de conmutación entre contextos para a) varias colas de diferente prioridad y b) única cola ordenada por tiempos límite de entrega.

De forma complementaria, se modificó la función *prvCheckDelayedTasks( )* que verifica la existencia de procesos bloqueados próximos a retornar a la cola de procesos listos. Se insertó una condición adicional que actualiza el tiempo límite de entrega de cada tarea antes de realizar el cambio de una cola de procesos a otra.

Finalmente, se varió las condiciones de suplantación de procesos de modo que se resuelva a partir del valor de tiempo límite asociado en lugar del valor de prioridad. Los cambios se realizaron en la función de creación de tareas así como en la función *xTaskResumeAll( )* la cual reestablece las colas de procesos una vez que se retorna de un estado de suspensión total.

5.4 Manejo de escalamiento de frecuencia en SIWA-RTOS

Cómo se mencionó en la sección 3.6, de forma complementaria con el algoritmo de planificación, es posible disminuir el consumo de potencia de un sistema manejando diferentes modos de consumo en el microprocesador [10. y 23]. La familia PIC32MX permite la utilización de 9 diferentes modos de bajo consumo de energía subdivididos en dos categorías [17]. La primera categoría abarca los modos de consumo que permiten que el reloj del CPU se mantenga en ejecución mientras que el reloj de los periféricos puede ser habilitado o no. Por otra parte, los modos de consumo incluidos en la segunda categoría inhabilitan el reloj del CPU mientras que el de periféricos puede continuar en ejecución o no según el modo seleccionado.

Dado que cualquiera de los modos de consumo puede ser habilitado a través de software, se analizó los modos de consumo disponibles para lograr la adaptación de la propuesta complementaria señalada en la sección 3.6.2. De ese modo se buscó aprovechar los estados no urgentes dentro de la planificación para efectuar un cambio dinámico del modo de consumo.

Los modos incluidos en la segunda categoría fueron descartados puesto que el cambio dinámico de modo de consumo debe efectuarse en intervalos específicos durante la ejecución del módulo planificador y no es factible detener el reloj del CPU en esos momentos. Por otro lado los modos de la primera categoría permiten modificar el consumo sin inhabilitar los relojes del CPU y periféricos.

El método adoptado consistió en escalar tanto la frecuencia de reloj del CPU como la de los periféricos. Esto se consiguió al ajustar el multiplicador y el divisor de salida del PLL del oscilador principal (POSC) junto con el divisor de reloj de periféricos. Para ello se hizo uso de la función *OSCConfig(…)* la cual permite seleccionar la fuente de reloj deseada, el multiplicador del PLL, el valor del divisor de salida del PLL y el divisor de salida del oscilador FRC. Adicionalmente se utilizó la función *mOSCSetPBDIV(…)* para llevar a cabo el ajuste del valor para el divisor de frecuencia del bus de periféricos. De ese modo fue posible mantener el valor del reloj del bus de periféricos por debajo de la frecuencia máxima especificada para el CPU. La figura B.1, ubicada en el anexo 1, muestra el diagrama de los osciladores para la familia PIC32MX.

Se creó una librería especial llamada "*Freq_Optimize.h*" que invoca las funciones de escalamiento dinámico de frecuencia *vConfigOSC_Normal( )* y *vConfigOSC_Low( )*, las cuales permiten variar la frecuencia entre 80MHz y 48MHz respectivamente. La incorporación de estas funciones en una librería independiente permite al diseñador hacer uso de ellas en cualquier parte del código del sistema SIWA-RTOS. Además, se estableció una definición especial en la librería de configuración del sistema "*FreeRTOSConfig.h*" nombrada *configUSE_FREQ_OPTIMIZE*. Esta definición permite al futuro diseñador habilitar o no las funciones de variación dinámica de frecuencia de operación según sean los requerimientos de la aplicaciones.

Las funciones de optimización se frecuencia se acoplaron en dos secciones distintas dentro de la estructuración del sistema. Primero se modificó la función de inicialización del sistema llamada *prvSetupHardware( )* de modo que se estableciera la frecuencia del sistema al máximo configurado (80MHz). Luego se incluyó las funciones de escalamiento dentro de la rutina de cambio de contextos entre tareas debido a que en este punto es posible extraer el número de tareas que existe en cada cola de procesos dinámicamente.

La figura 5.10(b) presenta la modificación realizada a la rutina de intercambio de contextos respecto a la estructura generada anteriormente 5.10(a). Se observa que se permite escalar la frecuencia de operación solo cuando se ejecuta la penúltima tarea dentro de la cola de procesos listos, siendo el último proceso la tarea de ociosidad (idle).

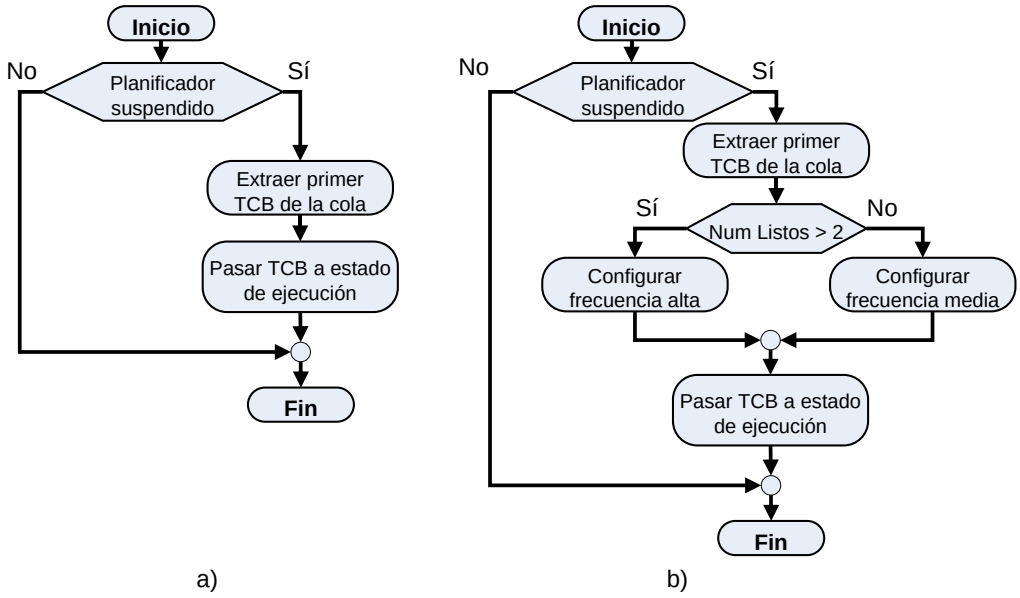


Figura 5.10 Función de conmutación entre contextos a) sin escalamiento de frecuencia y b) con escalamiento de frecuencia en estados no urgentes.

5.5 Cálculo del porcentaje del incremento promedio de la duración de una batería de 9V

Cómo se menciona en [14], es posible determinar la duración promedio de una batería de 9V a partir de los resultados de consumo de corriente promedio para cada uno de los sistemas operativos tratados. Se utiliza una batería para aplicaciones de alto consumo como la Duracell MN1604 puesto que presenta mayor carga para aplicaciones de uso prolongado. La carga de la batería utilizada es de 565mAh, donde se puede calcular la duración promedio de la misma mediante la ecuación (5.3) indistintamente del sistema operativo ejecutado.

$$T \approx \frac{C\,\text{arg}\,a\ Eléctrica\ Batería}{Consumo\ Eléctrico\ del\ PIC32MX} \tag{5.3}$$

El porcentaje de incremento de la duración de la batería se calcula a partir de la ecuación (5.4) donde T_{FPRRS} representa el tiempo de descarga de la batería utilizando el sistema SIWA-RTOS con la política basada en el esquema de prioridades mientras que $T_{EDF\text{-}DFS}$ corresponde a la utilización del esquema de tiempos de entrega con aprovechamiento de estados no urgentes.

$$P.Incremento \approx \frac{(T_{FPRRS} - T_{EDF-DFS})}{T_{FPRRS}} \cdot 100\,\% \tag{5.4}$$

Capitulo 6. Análisis de resultados

6.1 Determinación de los parámetros críticos del módulo planificador de procesos

La determinación de los parámetros críticos que afectan el consumo del sistema y que se encuentran asociados directamente al planificador de procesos se realizó con base en la investigación realizada. Los parámetros encontrados se explican seguidamente:

- Cantidad de conmutaciones entre contextos de tareas: un cambio de contexto se refiere al proceso de almacenar el estado (contexto) actual de ejecución del CPU y restaurar el estado de ejecución de una tarea anterior como respuesta a una solicitud efectuada, en este caso, por el módulo planificador. Con cada conmutación es necesario respaldar el puntero de programa así como todos los aquellos registros especiales utilizados por la tarea en ejecución, lo cual se realiza en un bloque de control de procesos (PCB) que se almacena en la pila del sistema operativo. Es por ello que una mayor conmutación se traduce en sobrecarga de la unidad de procesamiento en acciones no asociadas con la ejecución de las tareas en sí mismas sino inducidas por el módulo de calendarización.

- Rapidez para completar la ejecución de un conjunto de tareas específico: ligado a una menor cantidad de conmutaciones entre tareas, un menor tiempo consumido para terminar la ejecución de los procesos en la cola de listos permite al procesador ejecutar más pronto la tarea de bajo consumo (estado de ociosidad), y de ese modo entrar en el modo reducción de consumo.

- Prevenir inanición: toda tarea sin importar la prioridad asignada debe tener la posibilidad de completar su ejecución al cabo de un tiempo dado. Para ello, la política de planificación utilizada debe asegurar que todos los procesos tengan de oportunidad de entrar en el estado de ejecución en algún momento durante la planificación.

- Cumplimiento de los tiempos límite de entrega: al tratarse de un sistema de tiempo real, uno de los requerimientos esenciales es el cumplimiento estricto de los tiempos límite asignados para la compleción de las tareas. La política de planificación utilizada por el módulo de calendarización debe contemplar que aquellos procesos con tiempo límite de entrega más próximo se trate con una mayor prioridad que los otros dentro de las colas de procesos.

- Frecuencia de trabajo del CPU: a mayor frecuencia de operación es posible completar la ejecución de los procesos en un tiempo menor, aún así el consumo energético del sistema es mayor. Realizar un cambio de frecuencia repercute en el tiempo de compleción de las tareas por lo que el módulo calendarizador es quién debe establecer los momentos adecuados para realizar escalamientos de consumo sin afectar el desempeño del sistema.

6.2 Simulación de algoritmos de planificación

En primera instancia, la simulación de los algoritmos de planificación utilizando la herramienta TORSHE se centró en la corroboración del correcto comportamiento al ejecutarse sobre un conjunto de tareas de tamaño reducido.

La tabla 6.1 presenta las características establecidas para un conjunto de 5 tareas el cual se utilizó para realizar las pruebas sobre los algoritmos implementados. El gráfico de la figura 6.1 pretende ilustrar el conjunto de tareas y las características asociadas en función del tiempo. Puede observarse que luego de 20 ciclos se da la aparición de tareas que requieren procesarse de forma paralela, en estos casos la forma en la que se distribuyen los recursos del sistema es establecida por la política de planificación que se utilice.

Tabla 6.1 Características especificadas para un conjunto de prueba de 5 tareas**.**

N°	Nombre	Tiempo de procesamiento	Tiempo de inicio	Tiempo límite de entrega	Prioridad asociada
1	Tarea 1	15	15	45	2
2	Tarea 2	10	7	37	3
3	Tarea 3	10	37	65	1
4	Tarea 4	15	45	75	2
5	Tarea 5	20	20	80	4

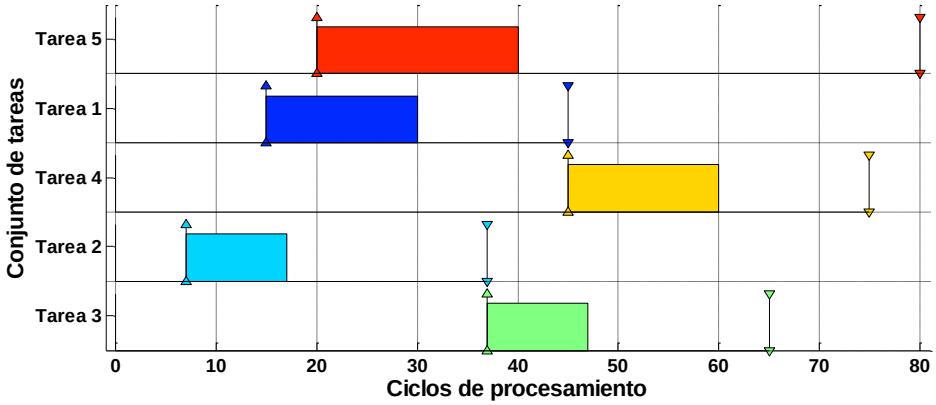


Figura 6.1 Conjunto de tareas correspondiente a las características de la tabla 6.1.

Las figura 6.2 exhibe el resultado obtenido mediante la simulación en la herramienta TORSHE de la planificación del conjunto de tareas de la figura 6.1, utilizando el método de prioridad fija. Se puede observar como el módulo de planificación realiza la transición al estado de ociosidad debido a que no existen procesos en el estado de listos hasta pasados los primeros 7 ciclos de ejecución. Luego la tarea 2 logra completar su ejecución puesto que posee una mayor prioridad asignada que la tarea 1. Es posible notar como la tarea 1 entra en estado de ejecución pero seguidamente es suplantada por la tarea 5 por tener una prioridad mayor. Además, se aprecia como la tarea 3 debe esperar casi 80 ciclos para iniciar su ejecución, aproximadamente 60 ciclos después de arrancado el evento que le dio origen, debido a la baja prioridad asignada.

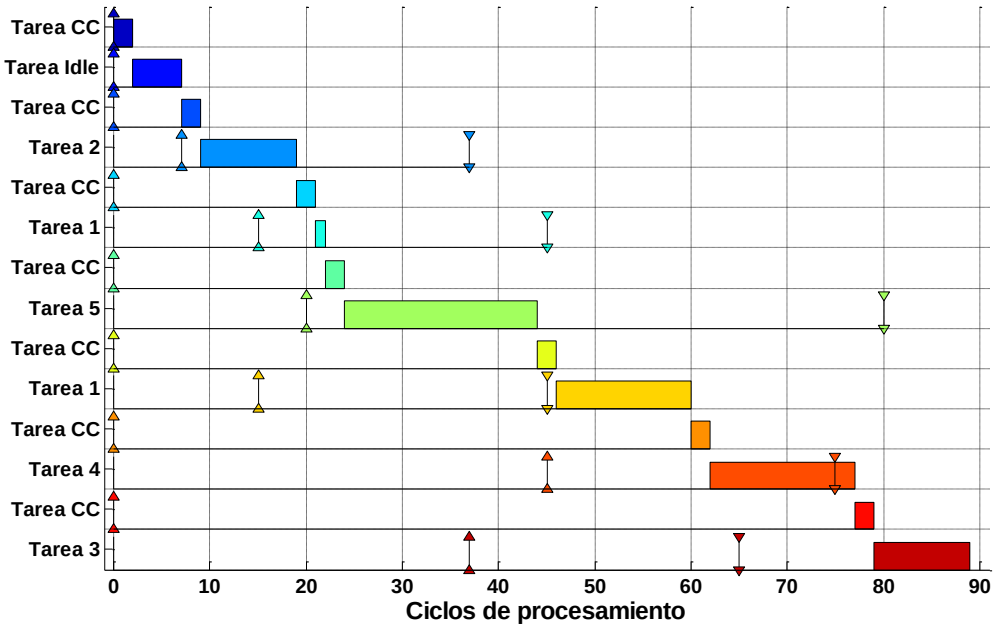


Figura 6.2 Resultado de la planificación del conjunto de tareas de la tabla 6.1 mediante la técnica de prioridad fija.

Por otra parte, la figura 6.3 presenta la planificación del conjunto de tareas de la figura 6.1 obtenida a partir del método de calendarización por turno rotatorio. Se especifica un valor de quantum de 10 ciclos de modo que la máxima cantidad de ciclos que cada tarea ejecuta de forma continua queda limitada a este valor. A diferencia de la técnica anterior, la tarea 3 logra completar su ejecución antes del vencimiento del tiempo límite de entrega fijado mientras que la tarea 4 se extiende por encima del mismo.

Con respecto al valor de quantum fijado, se presenta el impacto de un valor pequeño y uno grande sobre el resultado de la planificación mediante turno rotatorio. En la figura 6.4 se observa como un valor pequeño ocasiona una excesiva conmutación entre contextos, lo cual sumado a la penalización acarreada ocasiona la violación de más límites de tiempo de entrega. Además, un valor de quantum pequeño provoca una extensión en el tiempo requerido para completar la planificación.

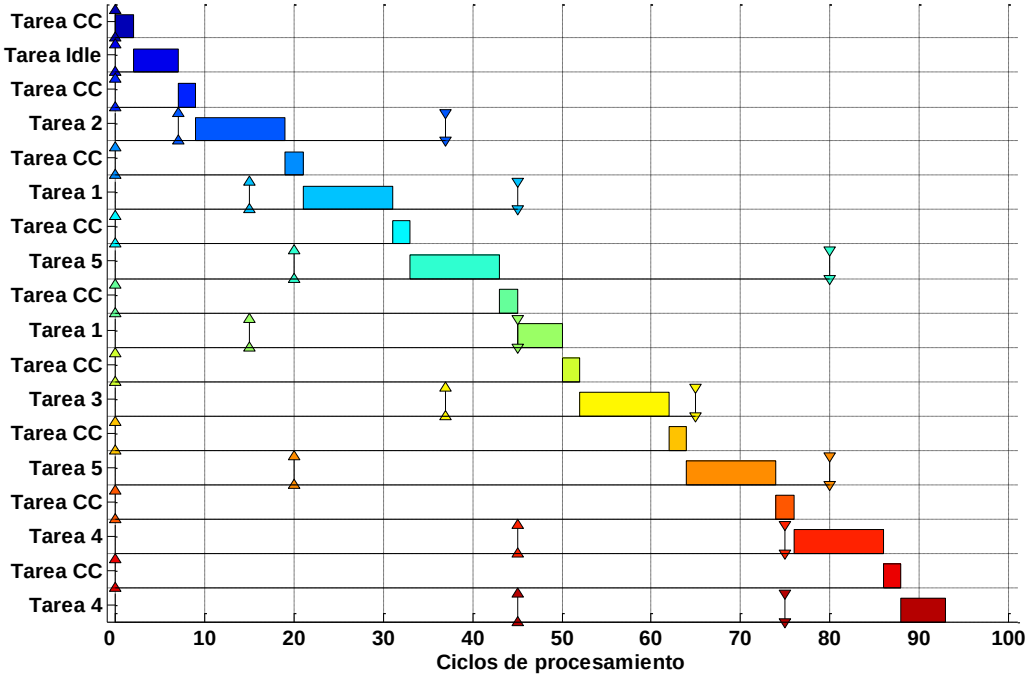


Figura 6.3 Resultado de la planificación del conjunto de tareas de la tabla 6.1 mediante la técnica de turno rotatorio, con un *quantum* de 10 ciclos.

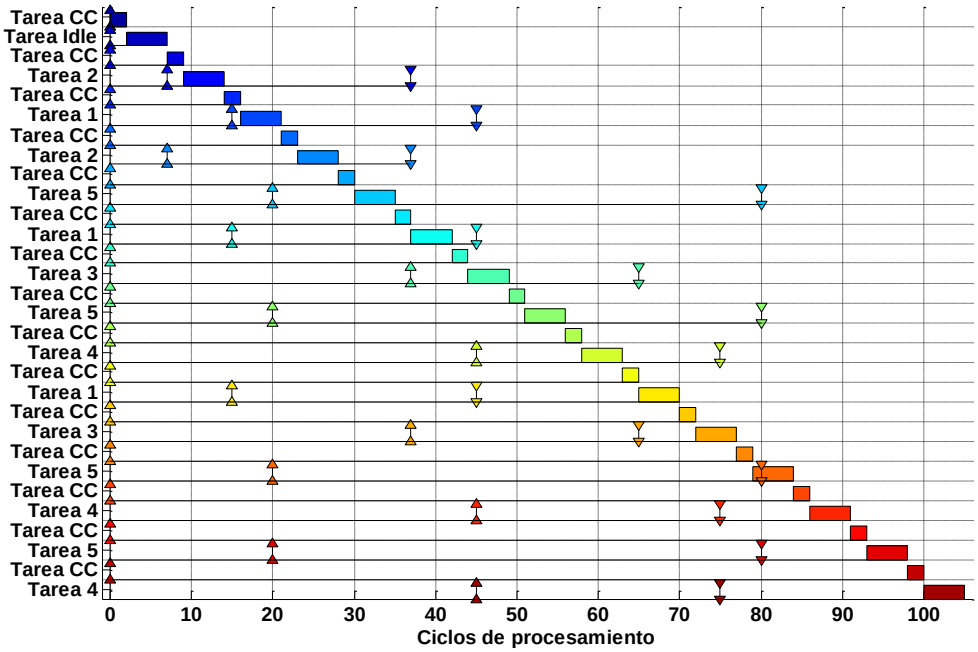


Figura 6.4 Resultado de la planificación del conjunto de tareas de la tabla 6.1 mediante la técnica de turno rotatorio, con un *quantum* de 5 ciclos.

El resultado mediante el método de planificación por tiempo límite de entrega más próximo primero se presenta en la figura 6.5. Se aprecia que el orden de planificación se establece a partir del tiempo de entrega fijado y se logra reducir el número de violaciones de los mismos respecto a los dos métodos presentados anteriormente. Además, se logra completar la planificación en el mismo tiempo que el método por prioridad fija y con igual número de conmutaciones entre contextos. Sin embargo, se viola un solo límite de entrega en comparación con los dos del método de prioridad fija.

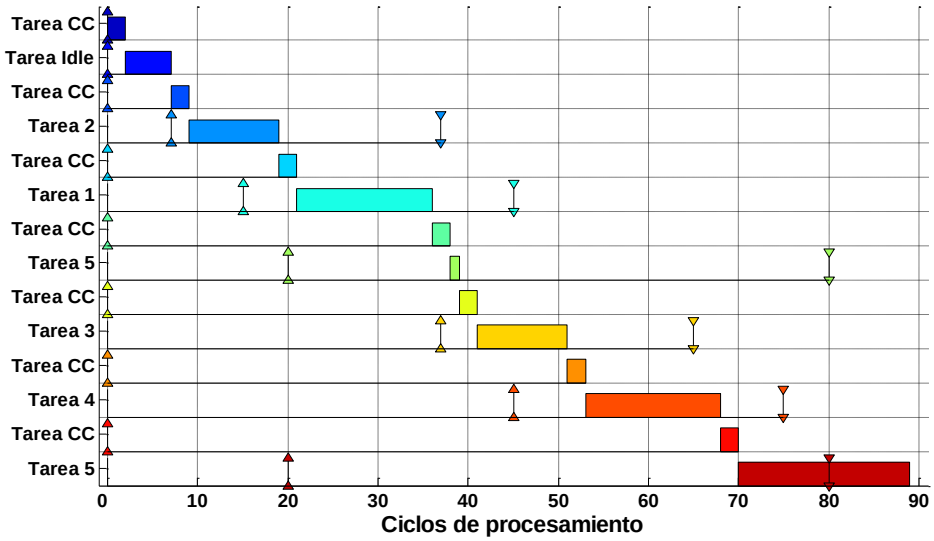


Figura 6.5 Resultado de la planificación del conjunto de tareas de la tabla 6.1 mediante la técnica de tiempo de entrega más próximo primero.

El método de cálculo de menor holgura (figura 6.6) computa de forma continua el tiempo sobrante para entregar la tarea respecto al tiempo de procesamiento requerido para completarla, lo cual se traduce en un mayor tiempo de cálculo por parte del calendarizador. Sin embargo, al igual que el método de tiempo de entrega más próximo, logra completar la planificación con el menor número de límites violados.

En general, es posible notar que la suplantación de procesos provoca que el módulo planificador realice mayor número de conmutaciones que si no se utiliza. No obstante, es una característica que faculta la ejecución de múltiples tareas en una sola unidad de procesamiento de forma paralela.

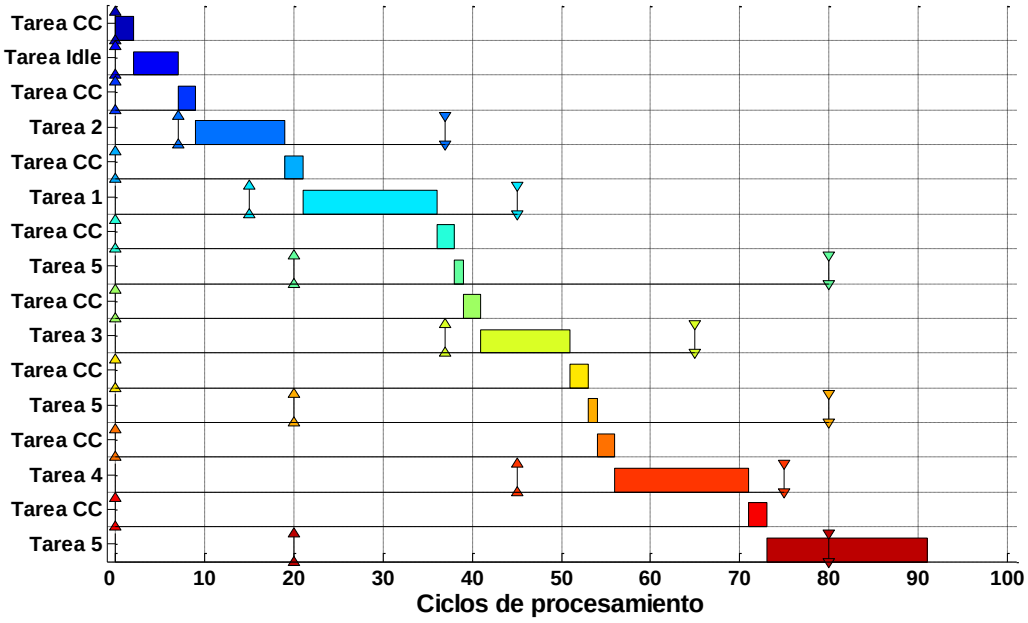


Figura 6.6 Resultado de la planificación del conjunto de tareas de la tabla 6.1 mediante la técnica de cálculo de menor holgura.

6.3 Obtención de estadísticas de comportamiento para diferentes algoritmos de planificación

Una vez concluida la fase de prueba y depuración de los algoritmos implementados en la herramienta TORSHE, se realizó pruebas de comportamiento para conjuntos con diferente cantidad de tareas. Los resultados estadísticos se muestran en las tablas 6.2, 6.3 y 6.4 y 6.5 para conjuntos de 6, 13, 17 y 22 tareas, respectivamente. El resultado completo de la planificación para 22 tareas se detalla en el apéndice A.2 mientras que el conjunto total de tareas se muestra en el apéndice A.1.

Tabla 6.2 Resultado estadístico obtenido en la planificación de un conjunto de prueba de 6 tareas para diferentes algoritmos de calendarización**.**

Método	Total de ciclos requeridos	Cant. Conmutaciones	Ciclos en Idle	Deadlines violados
Prioridad fija	108	10	8	1
Turno rotatorio[1]	108	10	8	0
Menor holgura primero	104	8	8	0
Tiempo de entrega más próximo	104	8	8	0

Tabla 6.3 Resultado estadístico obtenido en la planificación de un conjunto de prueba de 13 tareas para diferentes algoritmos de calendarización**.**

Método	Total de ciclos requeridos	Cant. Conmutaciones	Ciclos en Idle	Deadlines violados
Prioridad fija	232	22	8	4
Turno rotatorio[1]	230	21	8	2
Menor holgura primero	226	19	8	0
Tiempo de entrega más próximo	222	17	8	0

Tabla 6.4 Resultado estadístico obtenido en la planificación de un conjunto de prueba de 17 tareas para diferentes algoritmos de calendarización**.**

Método	Total de ciclos requeridos	Cant. Conmutaciones	Ciclos en Idle	Deadlines violados
Prioridad fija	301	29	8	5
Turno rotatorio[1]	297	27	8	3
Menor holgura primero	293	25	8	0
Tiempo de entrega más próximo	289	23	8	0

Tabla 6.5 Resultado estadístico obtenido en la planificación de un conjunto de prueba de 22 tareas para diferentes algoritmos de calendarización**.**

Método	Total de ciclos requeridos	Cant. Conmutaciones	Ciclos en Idle	Deadlines violados
Prioridad fija	383	35	8	7
Turno rotatorio[2]	383	35	8	5
Menor holgura primero	377	32	8	0
Tiempo de entrega más próximo	374	30	8	0

[1] Se define un valor de quantum de 10 ciclos de procesamiento.
[2] Se define un valor de quantum de 10 ciclos de procesamiento.

Los resultados estadísticos obtenidos permiten diferenciar el comportamiento de los algoritmos implementados. En general se observa como el método de tiempo de entrega más próximo es el que permite realizar la planificación en el menor tiempo, con la menor cantidad de conmutaciones entre contextos y cero tiempos límites violados.

Por otra parte, se aprecia que los métodos de prioridad fija y turno rotatorio se comportan de forma similar respecto al tiempo total consumido en la planificación y el número de conmutaciones efectuadas. No obstante, el método de prioridad fija ocasiona el incumplimiento de una mayor cantidad de tiempo límite de entrega.

A partir de los resultados obtenidos, se destaca que el método de cálculo de menor holgura presenta índices bajos respecto a los métodos de prioridad fija y turno rotatorio. Además, logra alcanzar la totalidad de los límites de tiempo propuestos. Sin embargo, el cálculo de holgura realizado requiere el conocimiento del tiempo total de procesamiento de cada tarea, lo cual ocasiona un aumento de la complejidad del cálculo dado que en un ambiente real resulta delicado determinar este parámetro.

6.4 Relación del consumo de potencia con la política de planificación utilizada según resultados de la simulación

Los resultados se obtienen de la utilización del conjunto de prueba de 22 tareas de la sección anterior, presentado con mayor detalle en el apéndice A.1. La tabla 6.6 recopila la información adquirida a través de la herramienta TORSHE referente a la caracterización del consumo respecto a la política de planificación utilizada.

Tabla 6.6 Relación de consumo respecto al escalamiento de frecuencia, para un conjunto de prueba de 22 tareas y diferentes algoritmos de calendarización**.**

Método	Ciclos como penúltima tarea	Consumo sin escalamiento [3,4] (mW)	Consumo con escalamiento[3,4] (mW)	Reducción estimada (%)
Prioridad fija	63	252,45	242,05	4,12
Turno rotatorio[5]	105	252,45	235,13	6,86
Menor holgura primero	142	248,29	224,86	9,44
Tiempo de entrega más próximo	175	246,18	217,31	11,73

[3] Para el cálculo se usan las Ec. (5.1) y (5.2) junto con los valores de las tablas 6.5 y 6.6.
[4] Se definen M_{Alto} = 0,66mW/ciclo, M_{Med} = 0,495mW/ciclo, M_{Low} = 0,33mW/ciclo y P_{Cntx} = 0,066mW/Cntx.
[5] Se define un valor de quantum de 10 ciclos de procesamiento.

Respecto a los valores detallados en la tabla 6.6 es posible observar el consumo aproximado para cada política de planificación. Se observa que según la ecuación (5.1) el consumo, sin hacer uso de la técnica de escalamiento, de los métodos de prioridad fija y turno rotatorio es equivalente puesto que el cálculo depende de los parámetros de la tabla 6.5. Por otra parte, al calcular el consumo del método de menor holgura es posible apreciar que se reduce en un 1.62% respecto a los dos primeros métodos expuestos en la tabla 6.6. Este caso se repite al calcular este parámetro con el método de tiempo límite de entrega más próximo, logrando obtener una reducción de un 2.48% aproximadamente.

El comportamiento del consumo, sin usar escalamiento, se debe a la dependencia del cálculo con los parámetros presentados en la tabla 6.5, donde al disminuir el número de conmutaciones entre contextos realizadas por cada política se reduce además el tiempo total requerido para la compleción del conjunto de tareas. Según la ecuación (5.1) una dependencia proporcional respecto a estos parámetros provoca la disminución del consumo obtenida.

Es posible comprobar, por medio de la simulación de los algoritmos, como el consumo disminuye al utilizar una política de planificación específica. No obstante, también se comprueba que el simple cambio de la política de planificación no supera el requerimiento de reducción planteado como objetivo general de este proyecto. Es por ello que se simula el efecto de la optimización dinámica de la frecuencia de operación en el consumo del sistema.

La tabla 6.6 contiene la cantidad de ciclos de procesamiento en los que se ejecuta el último proceso dentro de la cola de procesos listos según cada algoritmo implementado. Este parámetro permite obtener el consumo aproximado al aplicar la técnica de aprovechamiento de estados no urgentes como método complementario a la política de planificación seleccionada.

Con respecto a los resultados obtenidos se aprecia que el método de planificación que permite alcanzar el mayor porcentaje de reducción del consumo de potencia, respecto a la implementación sin escalamiento de frecuencia, es el de tiempo de entrega más próximo. Se observa una reducción del 11.73% respecto a sí mismo antes del escalamiento y una disminución de un 10.22% y 7.57% respecto a los algoritmos de planificación por prioridad fija y turno rotatorio, respectivamente, luego de usar la técnica de escalamiento.

Es así que con base en los resultados obtenidos mediante la simulación en la herramienta TORSHE se selecciona el método de planificación de tiempo de entrega más próximo con aprovechamiento de estados no urgentes (EDF-DFS) como el óptimo para el escenario utilizado por el sistema SIWA-RTOS.

6.5 Caracterización del consumo del sistema SIWA-RTOS en función de la frecuencia.

La determinación del consumo se realizó utilizando un sistema SMU 2636A el cual cuenta con la capacidad de realizar medición de tensión, corriente, resistencia y potencia instantánea al mismo tiempo que sirve como fuente de alimentación para el circuito de prueba. En este caso, se configuró el SMU como medidor de corriente y se fijó una tensión de alimentación de 9V. El tiempo de muestreo utilizado se especificó en 1ms y se realizaron 7500 muestras por cada medición.

La figura 6.7 muestra el gráfico de consumo de corriente de un nodo CRTecMote, sin realizar el filtrado de las mediciones obtenidas. Se observa que existen valores que se dispersan respecto a la línea de tendencia de la corriente. Por esta razón, se decide realizar un filtrado de los datos obtenidos haciendo uso de una media regresiva la cual determina el valor final de cada medición como una mediana respecto a los 20 valores anteriores. El resultado del filtrado del consumo de corriente se observa en la figura 6.8, se muestra a su vez una línea de tendencia por media móvil calculada para 20 periodos.

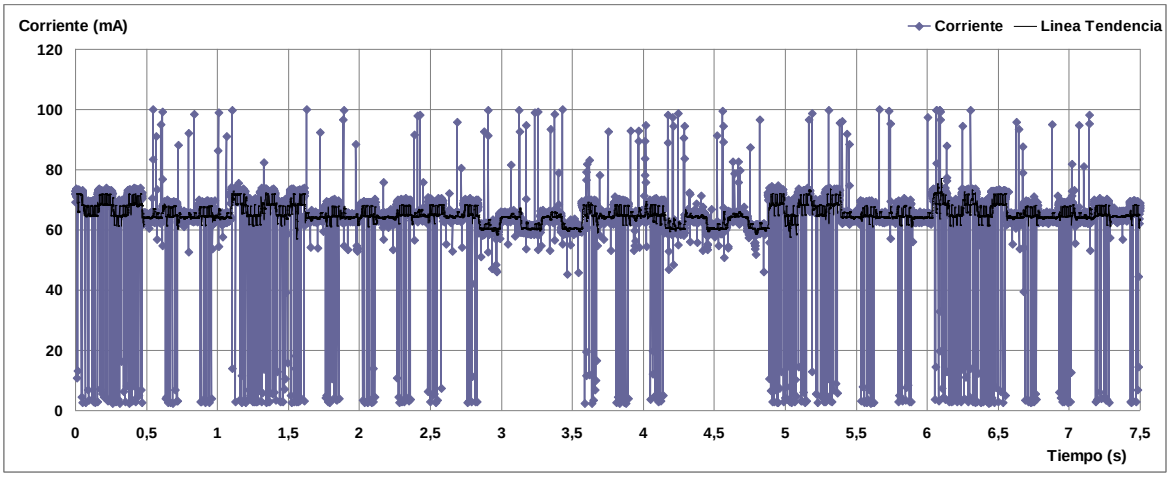


Figura 6.7 Consumo de corriente sin filtrar del sistema SIWA-RTOS a una frecuencia de operación de 80MHz.

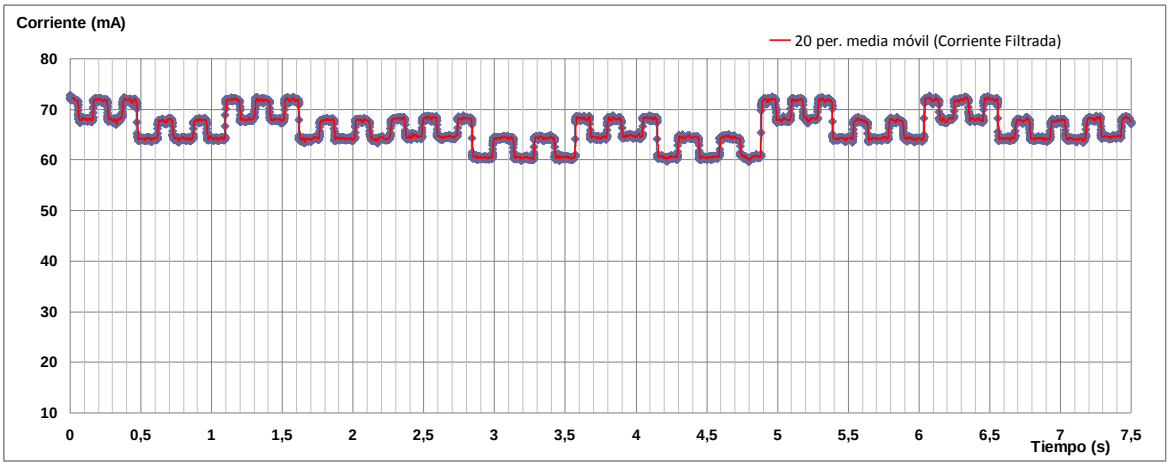


Figura 6.8 Consumo de corriente filtrada del sistema SIWA-RTOS a una frecuencia de operación de 80MHz.

La tabla 6.7 muestra el consumo promedio de corriente, una vez realizado el filtrado de los datos, para diferentes frecuencias de operación y un conjunto de prueba de 6 tareas. El resultado obtenido se grafica en la figura 6.9 donde es posible apreciar que existe un relación proporcional del consumo respecto a la frecuencia de trabajo del CPU. Esto se debe a que la potencia dinámica disipada por un circuito CMOS depende linealmente de la frecuencia de operación tal y como se muestra en la ecuación (6.1), donde C_L es la capacitancia de conmutación de los circuitos CMOS [10].

$$Potencia_{CMOS} = C_L \cdot Frecuencia \cdot (Tensión)^2 \tag{6.1}$$

Tabla 6.7 **C**onsumo promedio de corriente del sistema SIWA-RTOS respecto a la frecuencia de operación del CPU**.**

Frecuencia (MHz)	Corriente promedio filtrada (mA)
10	53,016
20	54,765
30	55,911
40	58,245
48	59,617
60	61,358
72	64,414
80	66,218
96	69,682

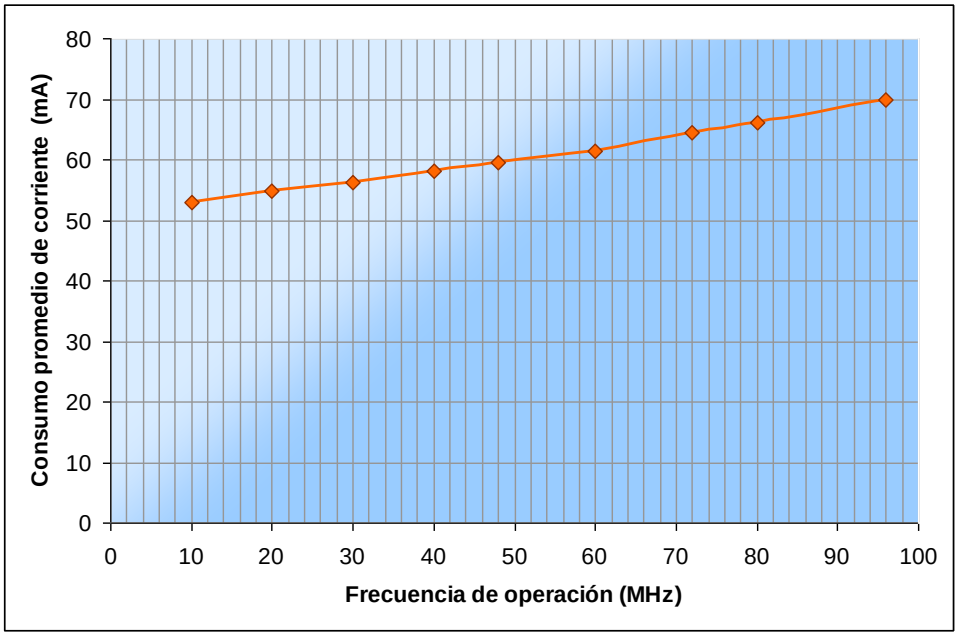


Figura 6.9 Relación entre el consumo de corriente promedio y la frecuencia de operación del sistema SIWA-RTOS original.

Además se observa que la potencia disipada en los elementos de una tarjeta de desarrollo es directamente proporcional al cuadrado de la tensión de alimentación de la misma. Por lo que de manera alternativa es posible modificar el consumo de potencia al aplicar un escalamiento dinámico de la tensión. Sin embargo, escalar la tensión de alimentación tiene efecto intrínsecamente sobre la frecuencia del sistema. En la ecuación (6.2) se observa como la frecuencia depende del nivel de tensión seleccionado, *V(m)*, el valor de tensión umbral entre niveles (V_t) y una constante que depende del circuito utilizado.

$$Frec(m) = K \cdot \frac{(V(m) - V_t)^2}{V(m)} \tag{6.2}$$

De ahí que se observa que sin importar cual de los dos parámetros se varíe dinámicamente, tensión o frecuencia, el resultado obtenido recae en un escalamiento de la frecuencia del sistema y con ello se afecta el consumo energético del circuito en cuestión.

6.6 Comprobación de la implementación de algoritmo EDFS sobre SIWA-RTOS.

El algoritmo de calendarización utilizado por el sistema SIWA-RTOS se le refiere como método de planificación por turno rotatorio con prioridad fija (FPRRS). El algoritmo implementado lleva por nombre planificación de tiempo límite de entrega primero (EDFS).

Se estableció un conjunto de 5 tareas con diferente periodicidad y prioridad asignada. Las características específicas del conjunto de prueba utilizado se muestran en la tabla 6.8. En la implementación se asume que el tiempo límite de entrega se relaciona directamente con el valor de periodicidad de cada tarea, donde este se actualiza cada vez que la tarea

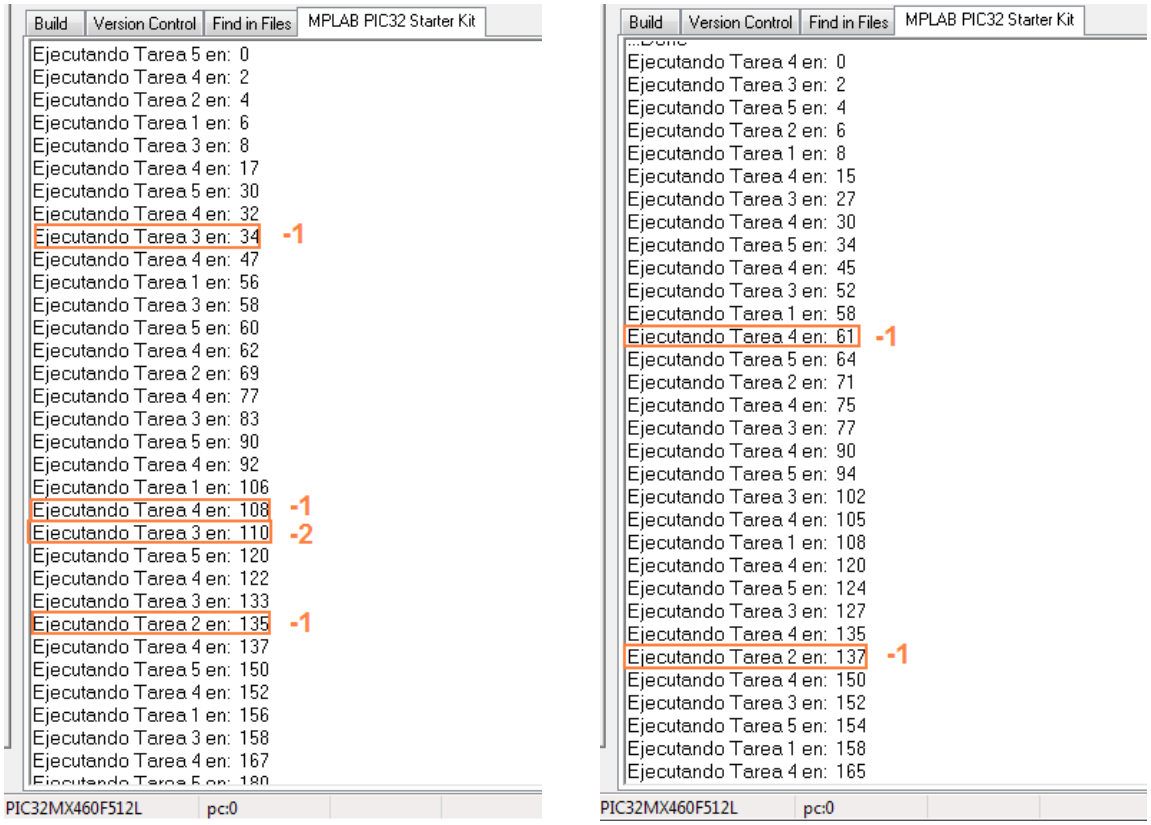
Tabla 6.8 Conjunto de prueba de 5 tareas implementado en SIWA-RTOS**.**

N°	Nombre	Periodo (ms)	Prioridad asociada
1	Tarea 1	50	1
2	Tarea 2	65	2
3	Tarea 3	25	1
4	Tarea 4	15	3
5	Tarea 5	30	4

El resultado de la planificación del conjunto de tareas propuesto se presenta en la figura 6.10. A manera de comparación se muestra en a) la planificación obtenida luego de utilizar el algoritmo FPRRS y en b) la política implementada EDFS. Se observa como en el caso a) se ordena la ejecución de los procesos según el valor de prioridad asignado (Tarea 5, Tarea 4, Tarea 2, Tarea 1, Tarea 3) mientras que el caso b) se ordena por tiempo de entrega más próximo (Tarea 4, Tarea 3, Tarea 5, Tarea 1, Tarea 2). Cabe resaltar que el inicialmente se toma el valor del periodo de cada tarea como especificación de tiempo límite y luego el valor se actualiza de forma dinámica.

Adicionalmente, en la figura 6.10 se enfatiza los casos en los cuales se dan violaciones del tiempo límite fijado para cada tarea. Para el caso de FPRRS se aprecia como suceden 4 incumplimientos dentro de los primeros 165ms mientras que para EDFS se obtienen solo 2 violaciones en el mismo intervalo de tiempo. Por otra parte, si se analiza el retardo producido por cada política de calendarización, es posible notar como FPRRS acarrea el retraso en la entrega de 3 tareas diferentes con un retardo máximo de 2ms en la Tarea 3.

La planificación EDFS permite no solo reducir el número de tareas retrasadas sino que posibilita disminuir la cantidad de ciclos de retardo durante las violaciones de tiempo límite.



a) b)

Figura 6.10 Resultado de la planificación del conjunto de tareas de la tabla 6.8 mediante a) FPRRS y b) EDFS.

6.7 Caracterización del consumo del sistema SIWA-RTOS al utilizar la política de planificación por tiempo de entrega más próximo.

Se realizó la caracterización del consumo en función de la frecuencia de operación al utilizar la política de tiempo de entrega más próximo primero. La tabla 6.9 resume los resultados obtenidos para un conjunto de prueba de 6 tareas. En la figura 6.11 se aprecia como los valores de consumo de corriente de la política de tiempo de entrega más próximo mantienen una dependencia directamente proporcional con la frecuencia de operación del CPU.

Tabla 6.9 **C**onsumo promedio de corriente del sistema SIWA-RTOS respecto a la frecuencia de operación del CPU al utilizar la política de planificación por tiempo de entrega más próximo**.**

Frecuencia (MHz)	Corriente promedio filtrada Planificación EDF (mA)
10	52,919
20	54,750
30	56,352
40	58,156
48	59,609
60	61,475
72	64,382
80	66,030
96	69,834

Al realizar la comparación se demuestra una cercanía entre los valores medidos antes y después de la modificación del módulo de planificación. Es así como el margen de comparación se encuentra por debajo del 0.8%, lo cual no representa un margen válido de contraste efectivo. Por lo que se asume un comportamiento semejante entre ambos métodos de planificación respecto al consumo promedio de corriente.

La dependencia lineal del consumo de consumo del dispositivo respecto a la frecuencia de operación se explica mediante la ecuación (6.1), donde al mantener una tensión de alimentación constante y considerando el parámetro C_L como propio de la tarjeta utilizada, se nota una relación lineal.

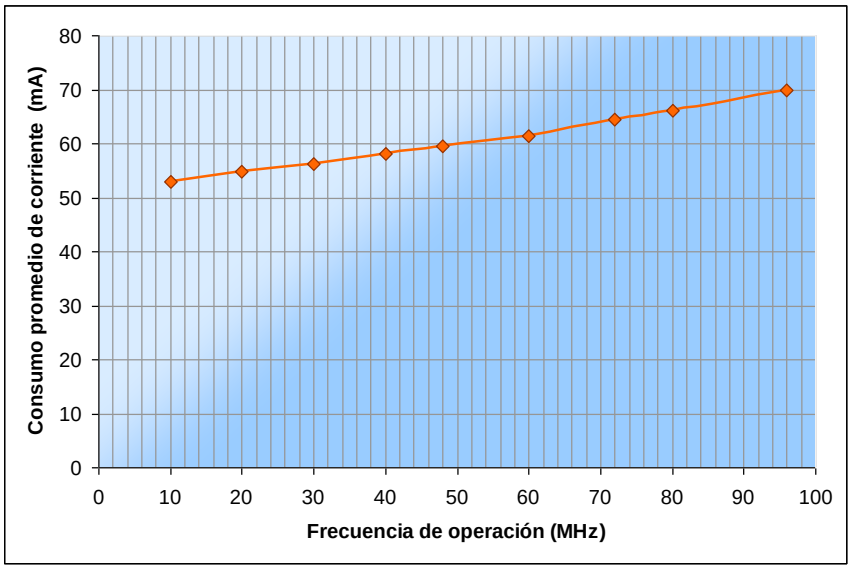


Figura 6.11 Relación entre el consumo de corriente promedio y la frecuencia de operación del sistema SIWA-RTOS con política de planificación por tiempo límite de entrega más próximo.

6.8 Cuantificación del consumo del sistema SIWA-RTOS al utilizar la política de planificación EDF-DFS.

El algoritmo de calendarización utilizado por el sistema SIWA-RTOS se le refiere como método de planificación por turno rotatorio con prioridad fija (FPRR). Por otra parte, el algoritmo seleccionado en la implementación se le denomina planificación de tiempo de entrega más próximo con escalamiento de frecuencia en estados no urgentes (EDF-DFS).

Las pruebas de consumo se realizaron inicialmente utilizando el mismo conjunto de 6 tareas de las secciones anteriores. Los resultados obtenidos se resumen en la tabla 6.10 junto con los valores promedio de la corriente consumida por el sistema al utilizar un conjunto de 3 tareas.

Tabla 6.10 Consumo promedio de corriente del sistema SIWA-RTOS respecto a la frecuencia de operación del CPU al utilizar la política de planificación por tiempo de entrega más próximo**.**

Número de tareas	Corriente filtrada promedio FPRR (mA)	Corriente filtrada promedio EDF-DFS (mA)	Porcentaje de reducción (%)
3	66,176	58,244	11,99
6	66,218	58,273	11,99

Es posible observar como el consumo del sistema se incrementa con el número de tareas, independientemente de la política utilizada. Esto se debe una mayor cantidad de tareas incrementa el número de conmutaciones entre contextos y además reduce los estados de ociosidad del sistema. Entre menos ocasiones se ejecute la tarea Idle entonces menor será el tiempo que permanecerá el sistema en el estado de bajo consumo.

De los resultados de la tabla 6.10 es posible observar como para un conjunto de 6 tareas en ejecución se obtiene el mismo porcentaje de reducción de potencia que para un conjunto de 3 tareas. Esto se justifica con respecto a las características del conjunto de tareas utilizado, puesto que el consumo asociado con cada política de planificación es directamente proporcional al tiempo de procesamiento de cada tarea e inversamente respecto al periodo de las tareas contempladas.

Al establecer un periodo menor se reduce a su vez el tiempo límite de entrega de cada tarea y se incrementa la cantidad de procesos en la cola de tareas listas, reduciendo la oportunidad de generación de estados no urgentes como el estado de ociosidad y la ejecución de la última tarea en la cola de listos.

Con respecto al porcentaje de reducción obtenido, se observa que supera el 10% de la especificación planteada dentro del objetivo general, siendo cierto que posibilita el aumento de la eficiencia respecto al consumo de un nodo CRTecMote.

6.9 Porcentaje de incremento de la duración promedio de una batería de 9V (565mAh).

La tabla 6.11 presenta el resultado del cálculo de la duración promedio de descarga de una batería de larga duración de 9V con una carga aproximada de 565mAh. Se observa que gracias a la modificación del algoritmo de planificación del sistema SIWA-RTOS se obtiene una extensión en el tiempo de descarga de 1 hora con 9 minutos aproximadamente.

Tabla 6.11 Duración promedio de la carga eléctrica de una batería de 9V (565mAh) al ejecutar el sistema SIWA-RTOS con diferentes políticas de planificación**.**

Duración promedio FPRR (h)	Duración promedio EDF-DFS (h)	Porcentaje de incremento (%)
8,532	9,696	13,64

Los cálculos presentados se realizaron mediante las ecuaciones (5.3) y (5.4) utilizando los valores de consumo de corriente presentados en la tabla 6.10 para un conjunto de 6 tareas. De este modo se demuestra que la reducción del consumo de corriente del sistema SIWA-RTOS a partir de la modificación del algoritmo de calendarización de procesos posibilita a su vez alargar el tiempo de descarga de la fuente de alimentación utilizada hasta en un 13,64%.

Capitulo 7. Conclusiones y recomendaciones

7.1 Conclusiones

Se realizó la síntesis de diferentes políticas de planificación de procesos consiguiendo determinar los parámetros relacionados con la eficiencia de cada una de las políticas utilizadas en sistemas de tiempo real.

Se modificó una herramienta de simulación de algoritmos de modo que contemplara el escenario de operación del sistema SIWA-RTOS, con el fin de proveer un medio que sirve como base de comparación respecto al comportamiento de diferentes políticas de planificación.

Se caracterizó el consumo del sistema para diferentes frecuencias, usando tanto la política FPRRS como EDFS, comprobándose además la relación lineal existente entre ambos parámetros.

Se modificó de forma exitosa el sistema SIWA-RTOS para incluir el algoritmo de planificación EDF-DFS diseñado.

Se logró establecer el diseño de un algoritmo de planificación de procesos que posibilita la reducción del consumo efectuado por el sistema SIWA-RTOS, comprobándose una reducción del 11,99% del consumo de corriente del sistema después de la modificación respecto a la versión original.

Se consiguió a partir de las modificaciones realizadas extender la duración promedio (en horas) de una batería de 9V en un 13,64% aproximadamente respecto a la versión de SIWA-RTOS con FPRRS.

7.2 Recomendaciones

Una manera alternativa de reducir el consumo en un nodo CRTecMote es utilizar una técnica de escalamiento dinámico de la tensión de alimentación según las características de la tarea en estado de ejecución actual. Sin embargo debe tomarse en cuenta que al variar dicha tensión se modifica a su vez la frecuencia de operación del sistema y con ello puede verse afectado el desempeño total del mismo. Es así que el uso de esta técnica debe realizarse solo en casos donde los tiempos límite de entrega permitan hacerlo.

El algoritmo original utilizado por el sistema SIWA-RTOS posee un esquema optimizado respecto al manejo de prioridades. En este sentido, incluye diferentes funciones que se encargan de reducir el efecto sobre el desempeño de algunas de las "trampas" asociadas con el esquema de prioridades. Si bien es cierto, se trató de realizar una modificación completa de la estructura del sistema operativo, es necesario establecer un periodo de pruebas y depuración más prolongado para detectar nuevas "trampas" asociadas con el esquema de manejo de tiempos de entrega.

Se recomienda a los futuros diseñadores propiciar una ejecución periódica de las tareas diseñadas con el fin de poseer un parámetro aproximado del tiempo límite de entrega que debe fijarse. A su vez, una ejecución periódica permite aprovechar de forma óptima el esquema establecido (EDF-DFS) y disminuir el problema de inanición del esquema FPRRS.

Es importante enviar cada tarea al estado de procesos bloqueados, mediante una llamada a la función *vTaskDelayUntil (...)*, al final de cada ejecución con el fin de extender la probabilidad de entrar en alguno de los estados no urgentes establecidos y con ello disminuir el consumo de potencia de un nodo CRTecMote.

Los futuros diseñadores podrán establecer un nivel superior de frecuencia intermedia de ser necesario. Esto con el fin de acelerar la ejecución de tareas de alta demanda de procesamiento como por ejemplo un filtro digital.

Bibliografía

[1]. Audsley, Neil. and Burns, Alan. (1990). *Real-Time system scheduling*. Technical Report YCS 134, Department of Computer Science, University of York.

[2]. Audsley, Neil et al. (1995). *Fixed priority pre-emptive scheduling: An historical perspectiva*. Real-Time Systems. Volume 8, numbers 2-3, pages 173-198. Springer Netherlands.

[3]. Baccino, Diego et al. (2008). *Una Experiencia Piloto de Red de Sensores Inalámbricos para Aplicaciones Agronómicas.* IEEE, 7º encuentro de energía, potencia, instrumentación y medidas. Montevideo,URUGUAY. Pag 156.

[4]. Barry, Richard. (2009). "*A Practical Guide: Using The FreeRTOS Real Time Kernel*". Versión 1.0.4. FreeRTOS.org.

[5]. Brooks, R. and Iyengar, S. (2005). *Distributed sensor networks*. Chapman & Hall/CRC. New York.

[6]. CRTecMote. Sistema de Adquisición de datos a través de Redes inalámbricas de sensores de arquitectura abierta (CRTECMOTE). [En Línea]. Última visita: 11 Junio 2010. URL: <http://crtecmote.wikispaces.com/Diseño+de+los+Nodos>.

[7]. Eker, Johan and Cervin, Antón. (1999). *A Matlab Toolbox for Real-Time and Control Systems Co-Design*. IEEE Computer Society Washington, DC, USA.

[8]. Electronics for Bharath. *L.A.T.H.I - Intelligent Helper for The old and Aged People*. [En Línea]. Última modificación: 11 Septiembre 2009. Última visita: 12 Marzo 2010. URL: <http://m8051.blogspot.com/2009/09/pic32-design-challenge-entry.html>.

[9]. FreeRTOS.org. *License Details*. [En Línea]. Última modificación: 27 Febrero 2010. Última visita: 12 Marzo 2010. URL: <http://www.freertos.org/index.html?http://www.freertos.org/a00114.html#exception>.

[10]. Gorjiara, B. Bagherzadeh, N. and Chou, P. (2007). *Ultra-fast and efficient algorithm for energy optimization by gradient-based stochastic voltage and task scheduling.* ACM Transactions on Design Automation of Electronic Systems (TODAES). Volume 12 , Issue 4.

[11]. Gutiérrez, M. y Calvo, N. *Normas de presentación de los informes de prácticas de especialidad, tesis, seminarios y otros del ITCR en formato digital*. Biblioteca José Figueres Ferrer, Instituto Tecnológico de Costa Rica. Versión Junio del 2010.

[12]. Harbour, M., Klein, M. and Lehoczky, J. (1991). *Fixed priority scheduling of periodic tasks with varying execution priority*. Proceedings of the 12th IEEE Real-Time Systems Symposium.

[13]. Lee, Insup et al. (2008). *Handbook of Real-Time and Embedded Systems*. Chapman & Hall/CRC. Boca Raton, Florida.

[14]. Leiva, Norwing. (2009). Informe final de proyecto de graduación. *Sistema de administración en tiempo real de los recursos de hardware y software de un nodo de sensado [SIWA-RTOS]*. Instituto Tecnológico de Costa Rica.

[15]. Lin, Mingjie and Ganjali, Yashar. (2006). *Power-Efficient Rate Scheduling in Wireless Links Using Computational Geometric Algorithms*. International Conference On Communications And Mobile Computing. Vancouver, British Columbia, Canada.

[16]. Liu, C. and Layland, J. (1973). *Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment.* Journal of the ACM, volume 20, issue 1, pages 46-61.

[17]. Microchip Technology Inc©. PIC32MX3XX/4XX Family Data Sheet. Descargado el: 24 Marzo 2010. URL: <http://ww1.microchip.com/ downloads/en/DeviceDoc/61143E.pdf >.

[18]. Mercer, Clifford. (1992). *An Introduction to Real-Time Operating Systems: Scheduling Theory*. School of Computer Science, Carnegie Mellon University. Pittsburgh, Pennsylvania.

[19]. NPTEL, National Programme on Technology Enhanced Learning. *Real-Time Operating Systems and Microkernels.* [En Línea]. Visitado el: 12 Marzo 2010. URL: <http://nptel.tvm.ernet.in/courses/Webcourse-contents/IISc-BANG/ Operating Systems/pdf/Mod 8_LN.pdf>.

[20]. Ramamritham, Krithi and Stankovic, John. (2002). Scheduling Algorithms and Operating Systems Support for Real-Time Systems. IEEE Transactions on Knowledge and Data Engineering. IEEE Educational Activities Department Piscataway, NJ, USA.

[21]. Sha, Lui et al. (2004). Real Time Scheduling Theory: A Historical Perspective. Real-Time Systems. Volume 28, issue 2-3, pages: 101–155. Kluwer Academic Publishers. Norwell, MA, USA.

[22]. Shin, Y. and Choi, K. (1999). Power conscious fixed priority scheduling for hard real-time systems. Proceedings of the 36th annual ACM/IEEE Design Automation Conference. Pages: 134 – 139. New Orleans, Louisiana, United States.

[23]. Tanenbaum, Andrew y Woodhull, Albert. (1997). *Sistemas Operativos: Diseño e Implementación*. 2da Edición. Prentice-Hall, Mexico.

[24]. Yuan, Wanghong and Nahrstedt, Klara. (2003). *Energy-Efficient Soft Real-Time CPU Scheduling for Mobile Multimedia Systems*. ACM Symposium on Operating Systems Principles. Bolton Landing, New York, USA.

[25]. Vahid, S., et. al. (2005) "A Modified Maximum Urgency First Scheduling Algorithm for Real-Time Tasks", Transactions on Engineering, Computing and Technology, pages 19-23

Apéndice A.1. Conjunto de prueba de 22 tareas

Tabla A.1 Características especificadas para un conjunto de prueba de 22 tareas**.**

N°	Nombre	Tiempo de procesamiento	Tiempo de inicio	Tiempo de entrega	Prioridad asociada
1	Tarea 1	15	10	46	2
2	Tarea 2	10	15	66	4
3	Tarea 3	20	30	105	3
4	Tarea 4	10	46	82	4
5	Tarea 5	15	66	117	2
6	Tarea 6	10	82	118	4
7	Tarea 7	20	105	180	3
8	Tarea 8	10	118	154	4
9	Tarea 9	15	117	168	2
10	Tarea 10	10	154	190	4
11	Tarea 11	15	168	219	2
12	Tarea 12	20	180	255	3
13	Tarea 13	10	190	226	4
14	Tarea 14	10	226	262	4
15	Tarea 15	15	219	270	2
16	Tarea 16	20	255	330	3
17	Tarea 17	10	262	298	4
18	Tarea 18	15	270	321	2
19	Tarea 19	10	298	334	4
20	Tarea 20	20	330	405	3
21	Tarea 21	15	321	372	2
22	Tarea 22	10	334	370	4

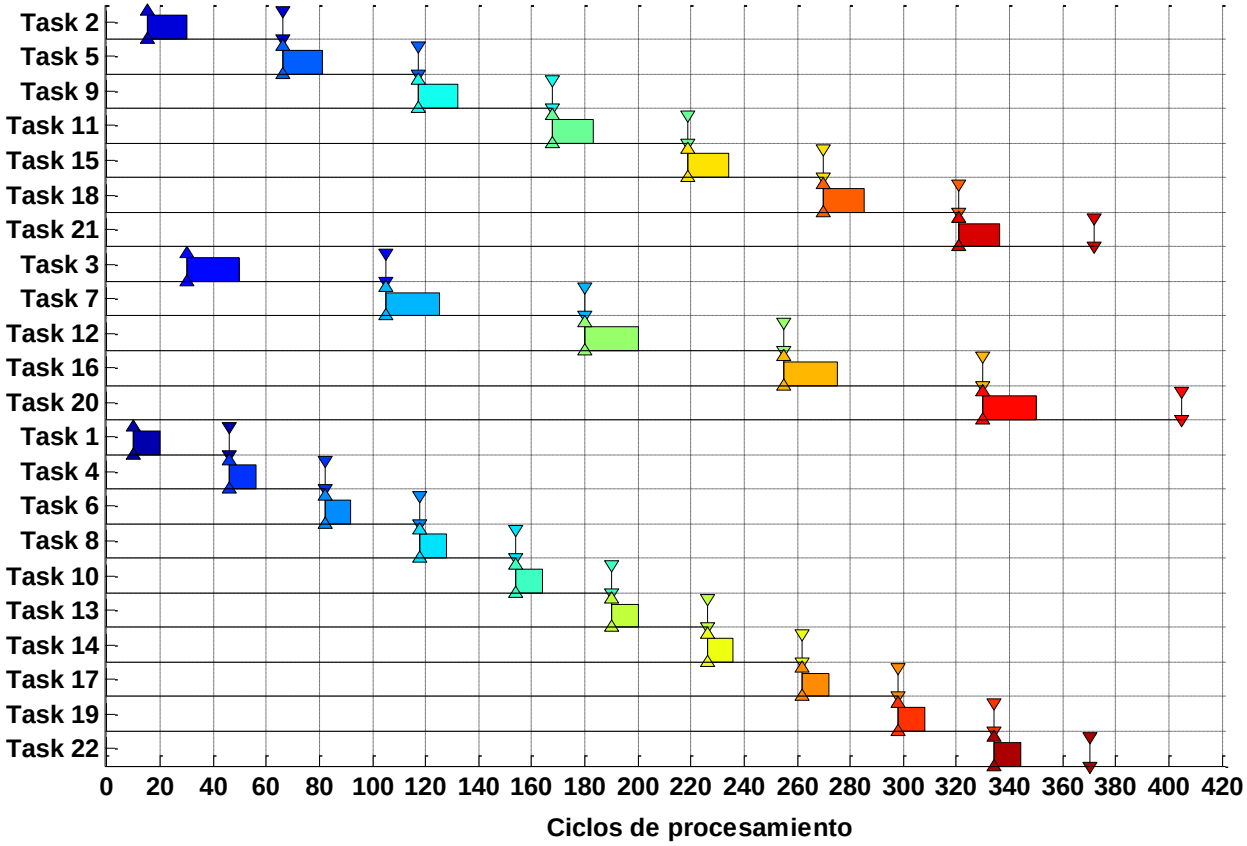


Figura A.1 Conjunto de tareas correspondiente a las características de la tabla A.1.

Apéndice A.2. Resultado completo de planificación de A.1.

A.2.1 Método de planificación por turno rotatorio

Tabla A.2.1 Resultado de la planificación por turno rotatorio para el conjunto de prueba de 22 tareas especificado en A.1.

N°	Nombre	T.Proc	T.Inicio	T.Entrega
1	Tarea CC	2	0	0
2	Tarea Idle	8	2	0
3	Tarea CC	2	10	0
4	Tarea 1	10	12	46
5	Tarea CC	2	22	0
6	Tarea 2	10	24	66
7	Tarea CC	2	34	0
8	Tarea 3	10	36	105
9	Tarea CC	2	46	0
10	Tarea 2	5	48	66
11	Tarea CC	2	53	0
12	Tarea 4	10	55	82
13	Tarea CC	2	65	0
14	Tarea 3	10	67	105
15	Tarea CC	2	77	0
16	Tarea 5	10	79	117
17	Tarea CC	2	89	0
18	Tarea 6	10	91	118
19	Tarea CC	2	101	0
20	Tarea 5	5	103	117
21	Tarea CC	2	108	0
22	Tarea 7	10	110	180
23	Tarea CC	2	120	0
24	Tarea 9	10	122	168
25	Tarea CC	2	132	0
26	Tarea 7	10	134	180
27	Tarea CC	2	144	0
28	Tarea 8	10	146	154
29	Tarea CC	2	156	0
30	Tarea 10	10	158	190
31	Tarea CC	2	168	0
32	Tarea 9	5	170	168
33	Tarea CC	2	175	0
34	Tarea 11	10	177	219
35	Tarea CC	2	187	0
36	Tarea 12	10	189	255
37	Tarea CC	2	199	0
38	Tarea 11	5	201	219
39	Tarea CC	2	206	0
40	Tarea 13	10	208	226
41	Tarea CC	2	218	0
42	Tarea 12	10	220	255
43	Tarea CC	2	230	0
44	Tarea 15	10	232	270
45	Tarea CC	2	242	0
46	Tarea 14	10	244	262
47	Tarea CC	2	254	0
48	Tarea 16	10	256	330
49	Tarea CC	2	266	0
50	Tarea 15	5	268	270
51	Tarea CC	2	273	0
52	Tarea 17	10	275	298
53	Tarea CC	2	285	0
54	Tarea 16	10	287	330
55	Tarea CC	2	297	0
56	Tarea 18	10	299	321
57	Tarea CC	2	309	0
58	Tarea 19	10	311	334
59	Tarea CC	2	321	0
60	Tarea 18	5	323	321
61	Tarea CC	2	328	0
62	Tarea 21	10	330	372
63	Tarea CC	2	340	0
64	Tarea 20	10	342	405
65	Tarea CC	2	352	0
66	Tarea 21	5	354	372
67	Tarea CC	2	359	0
68	Tarea 22	10	361	370
69	Tarea CC	2	371	0
70	Tarea 20	10	373	405

A.2.2 Método de planificación por prioridad fija

Tabla A.2.2 Resultado de la planificación por prioridad fija para el conjunto de prueba de 22 tareas especificado en A.1**.**

N°	Nombre	T.Proc	T.Inicio	T.Entrega
1	Tarea CC	2	0	0
2	Tarea Idle	8	2	0
3	Tarea CC	2	10	0
4	Tarea 1	10	12	46
5	Tarea CC	2	22	0
6	Tarea 2	8	24	66
7	Tarea CC	2	32	0
8	Tarea 3	14	34	105
9	Tarea CC	2	48	0
10	Tarea 4	10	50	82
11	Tarea CC	2	60	0
12	Tarea 3	6	62	105
13	Tarea CC	2	68	0
14	Tarea 2	7	70	66
15	Tarea CC	2	77	0
16	Tarea 5	5	79	117
17	Tarea CC	2	84	0
18	Tarea 6	10	86	118
19	Tarea CC	2	96	0
20	Tarea 5	9	98	117
21	Tarea CC	2	107	0
22	Tarea 7	11	109	180
23	Tarea CC	2	120	0
24	Tarea 8	10	122	154
25	Tarea CC	2	132	0
26	Tarea 7	9	134	180
27	Tarea CC	2	143	0
28	Tarea 5	1	145	117
29	Tarea CC	2	146	0
30	Tarea 9	8	148	168
31	Tarea CC	2	156	0
32	Tarea 10	10	158	190
33	Tarea CC	2	168	0
34	Tarea 9	7	170	168
35	Tarea CC	2	177	0
36	Tarea 11	3	179	219
37	Tarea CC	2	182	0
38	Tarea 12	8	184	255
39	Tarea CC	2	192	0
40	Tarea 13	10	194	226
41	Tarea CC	2	204	0
42	Tarea 12	12	206	255
43	Tarea CC	2	218	0
44	Tarea 11	8	220	219
45	Tarea CC	2	228	0
46	Tarea 14	10	230	262
47	Tarea CC	2	240	0
48	Tarea 15	15	242	270
49	Tarea CC	2	257	0
50	Tarea 16	5	259	330
51	Tarea CC	2	264	0
52	Tarea 17	10	266	298
53	Tarea CC	2	276	0
54	Tarea 16	15	278	330
55	Tarea CC	2	293	0
56	Tarea 11	4	295	219
57	Tarea CC	2	299	0
58	Tarea 15	0	301	270
59	Tarea CC	2	301	0
60	Tarea 19	10	303	334
61	Tarea CC	2	313	0
62	Tarea 18	15	315	321
63	Tarea CC	2	330	0
64	Tarea 20	4	332	405
65	Tarea CC	2	336	0
66	Tarea 22	10	338	370
67	Tarea CC	2	348	0
68	Tarea 20	16	350	405
69	Tarea CC	2	366	0
70	Tarea 21	15	368	372

A.2.3 Método de planificación de menor holgura primero

Tabla A.2.3 Resultado de la planificación de menor holgura primero para el conjunto de prueba de 22 tareas especificado en A.1**.**

N°	Nombre	T.Proc	T.Inicio	T.Entrega
1	Tarea CC	2	0	0
2	Tarea Idle	8	2	0
3	Tarea CC	2	10	0
4	Tarea 1	10	12	46
5	Tarea CC	2	22	0
6	Tarea 2	15	24	66
7	Tarea CC	2	39	0
8	Tarea 3	7	41	105
9	Tarea CC	2	48	0
10	Tarea 4	10	50	82
11	Tarea CC	2	60	0
12	Tarea 3	13	62	105
13	Tarea CC	2	75	0
14	Tarea 5	15	77	117
15	Tarea CC	2	92	0
16	Tarea 6	10	94	118
17	Tarea CC	2	104	0
18	Tarea 7	12	106	180
19	Tarea CC	2	118	0
20	Tarea 9	1	120	168
21	Tarea CC	2	121	0
22	Tarea 8	10	123	154
23	Tarea CC	2	133	0
24	Tarea 7	1	135	180
25	Tarea CC	2	136	0
26	Tarea 9	14	138	168
27	Tarea CC	2	152	0
28	Tarea 7	7	154	180
29	Tarea CC	2	161	0
30	Tarea 10	10	163	190
31	Tarea CC	2	173	0
32	Tarea 11	15	175	219
33	Tarea CC	2	190	0
34	Tarea 12	1	192	255
35	Tarea CC	2	193	0
36	Tarea 13	10	195	226
37	Tarea CC	2	205	0
38	Tarea 12	19	207	255
39	Tarea CC	2	226	0
40	Tarea 15	1	228	270
41	Tarea CC	2	229	0
42	Tarea 14	10	231	262
43	Tarea CC	2	241	0
44	Tarea 15	14	243	270
45	Tarea CC	2	257	0
46	Tarea 16	5	259	330
47	Tarea CC	2	264	0
48	Tarea 17	10	266	298
49	Tarea CC	2	276	0
50	Tarea 16	1	278	330
51	Tarea CC	2	279	0
52	Tarea 18	15	281	321
53	Tarea CC	2	296	0
54	Tarea 16	14	298	330
55	Tarea CC	2	312	0
56	Tarea 19	10	314	334
57	Tarea CC	2	324	0
58	Tarea 21	15	326	372
59	Tarea CC	2	341	0
60	Tarea 20	1	343	405
61	Tarea CC	2	344	0
62	Tarea 22	10	346	370
63	Tarea CC	2	356	0
64	Tarea 20	19	358	405

A.2.4 Método de planificación de tiempo de entrega más próximo primero

Tabla A.2.4 Resultado de la planificación de tiempo de entrega más próximo primero para el conjunto de prueba de 22 tareas especificado en A.1**.**

N°	Nombre	T.Proc	T.Inicio	T.Entrega
1	Tarea CC	2	0	0
2	Tarea Idle	8	2	0
3	Tarea CC	2	10	0
4	Tarea 1	10	12	46
5	Tarea CC	2	22	0
6	Tarea 2	15	24	66
7	Tarea CC	2	39	0
8	Tarea 3	7	41	105
9	Tarea CC	2	48	0
10	Tarea 4	10	50	82
11	Tarea CC	2	60	0
12	Tarea 3	13	62	105
13	Tarea CC	2	75	0
14	Tarea 5	15	77	117
15	Tarea CC	2	92	0
16	Tarea 6	10	94	118
17	Tarea CC	2	104	0
18	Tarea 7	12	106	180
19	Tarea CC	2	118	0
20	Tarea 8	10	120	154
21	Tarea CC	2	130	0
22	Tarea 9	15	132	168
23	Tarea CC	2	147	0
24	Tarea 7	8	149	180
25	Tarea CC	2	157	0
26	Tarea 10	10	159	190
27	Tarea CC	2	169	0
28	Tarea 11	15	171	219
29	Tarea CC	2	186	0
30	Tarea 12	4	188	255
31	Tarea CC	2	192	0
32	Tarea 13	10	194	226
33	Tarea CC	2	204	0
34	Tarea 12	16	206	255
35	Tarea CC	2	222	0
36	Tarea 15	4	224	270
37	Tarea CC	2	228	0
38	Tarea 14	10	230	262
39	Tarea CC	2	240	0
40	Tarea 15	11	242	270
41	Tarea CC	2	253	0
42	Tarea 16	7	255	330
43	Tarea CC	2	262	0
44	Tarea 17	10	264	298
45	Tarea CC	2	274	0
46	Tarea 18	15	276	321
47	Tarea CC	2	291	0
48	Tarea 16	13	293	330
49	Tarea CC	2	306	0
50	Tarea 19	10	308	334
51	Tarea CC	2	318	0
52	Tarea Idle	1	320	0
53	Tarea CC	2	321	0
54	Tarea 21	13	323	372
55	Tarea CC	2	336	0
56	Tarea 22	10	338	370
57	Tarea CC	2	348	0
58	Tarea 21	2	350	372
59	Tarea CC	2	352	0
60	Tarea 20	20	354	405

Apéndice A.3. Glosario y abreviaturas

CPU: *Central Processing Unit*. En español "Unidad Central de Proceso". Es la parte de un computador o microcontrolador que interpreta las instrucciones, y procesa los datos que se encuentran en la memoria de programa del computador.

Deadline: Tiempo límite de entrega asociado a una tarea. La violación de este parámetro provoca problemas en sistemas de tiempo de real.

DFS: *Dynamic Frequency Scaling*. En español: "Escalamiento dinámico de frecuencia". Técnica de ahorro de potencia que posibilita cambiar la frecuencia de operación del CPU de forma dinámica según las características del conjunto de tareas en ejecución.

EDFS: *Earliest Deadline First Scheduling*. En español: "planificación por tiempo de entrega más próximo primero". Esquema de planificación de procesos que busca el cumplimiento de los deadline asignados al conjunto de tareas.

FPS: *Fixed Priority Scheduling*. En español: "planificación mediante prioridades fijas". Esquema de planificación que asigna una prioridad específica a cada tarea y según la cual se establece el orden de ejecución de las mismas.

Inanición: En programas concurrentes es posible que un proceso de prioridad baja nunca llegue a ejecutarse si el planificador o el control de los recursos compartidos respectivamente no permiten que el proceso pueda cumplir con sus objetivos, en presencia de procesos de prioridad mayor. Es decir, el proceso está sometido a una espera infinita.

mAh: *"miliAmperio por hora (mA/h)"*. Es la medida estándar para la capacidad de una batería. Indica la cantidad de carga eléctrica que fluye por las terminales de una batería. También señala la cantidad máxima de carga eléctrica que es capaz de almacenar una batería. Así, 1mAh es aproximadamente 3,6C (Coulomb).

MCU: *Microcontroller Unit*. Es la forma abreviada de referirse al microcontrolador que se trata de un computador completo en un solo circuito integrado, que incluye en su interior las tres unidades funcionales de una computadora: CPU, Memoria y Unidades de E/S.

PCB: *Process Control Block*: En Español: “Bloque de control de Proceso”. Es un registro especial donde el SO agrupa toda la información que necesita conocer respecto a un proceso particular. Cada vez que se crea un proceso el sistema operativo crea el PCB correspondiente para que sirva como descripción en tiempo de ejecución durante toda la vida del proceso.

Quantum: En español “Cuanto”. Cantidad de tiempo de CPU que puede hacer uso una tarea cada vez que se le asignan los recursos requeridos.

RRS: *Round Robin Scheduling*: En español: “planificación por turno rotatorio”. Esquema de planificación de procesos que busca una repartición equitativa del tiempo de procesamiento disponible.

Anexo B.1. Diagrama de osciladores de la familia PIC32MX

Figure 6-1: PIC32MX Family Clock Diagram

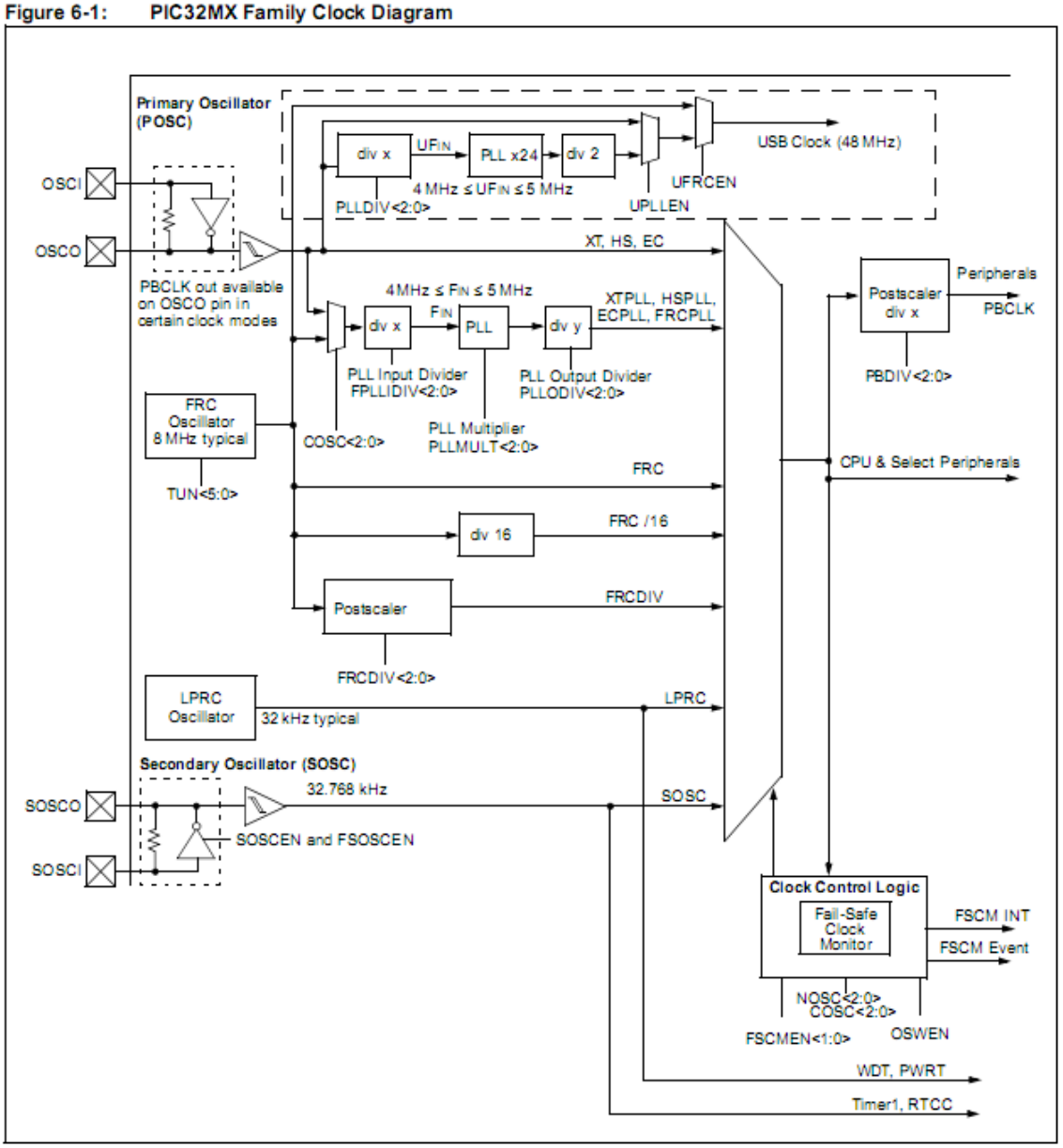


Figura B.1 Diagrama de osciladores de la familia PIC32MX, tomado de [17].

Anexo B.2. Hoja de información del proyecto

Información del estudiante:

Nombre: Alexander Fco. Valverde Serrano

Cédula: 2-0636-0538. **Carné ITCR:** 200517469.

Email: joskar11@gmail.com

Información del proyecto:

Nombre del Proyecto: Optimización del módulo de planificación de procesos del sistema operativo de tiempo real SIWA-RTOS utilizado en los nodos CRTecMote

Área del Proyecto: Estructura de Microprocesadores, Diseño de Sistemas Digitales, Análisis y diseño de algoritmos, Estructura de Sistemas Operativos en Tiempo Real, Sistemas Embebidos.

Información de la Empresa:

Nombre: Instituto Tecnológico de Costa Rica **Zona:** Los Ángeles, Cartago.

Dirección: 300 metros al este de la Escuela Sony.

Actividad Principal: Formación Académica.

Información del encargado en la Institución:

Nombre: William Marín Moreno.

Puesto que ocupa: Profesor Ing. Electrónica **Departamento:** Escuela Electrónica

Profesión: Ingeniero Electrónico **Grado académico:** Licenciatura

Teléfono: 2550-9239 **Email:** wmarin@itcr.ac.cr

Información del asesor en la empresa:

Nombre: Johan Carvajal Godínez

Puesto que ocupa: Profesor Ing. Electrónica **Departamento:** Escuela Electrónica

Profesión: Ingeniero Electrónico **Grado académico:** Licenciatura

Teléfono: 2550-9171 **Email:** johcarvajal@itcr.ac.cr