“DISEÑO DE UN SISTEMA DE CONTROL EN TIEMPO REAL PARA EL KERNEL DEL SISTEMA OPERATIVO UTILIZANDO MATLAB-SIMULINK”

Marco Espinel Cangui, Ing. Germán Ñacato Caiza, Ing. Freddy Tapia León

ESPE, ECUADOR, mespinel81@hotmail.com
ESPE, ECUADOR, gnacato@espe.edu.ec
ESPE, ECUADOR, fmtapia@espe.edu.ec

RESUMEN

En este artículo se describe la estructura de un Kernel de Tiempo Real. Las ventajas de este sistema son principalmente su tamaño, modularidad, facilidad de poder comunicarse con otros Kernels ejecutándose en otras computadoras y su interfaz con un sistema gráfico de simulación de tareas de tiempo real.
El Kernel de Tiempo Real es pequeño (aproximadamente 20.5Kb) y además realiza funciones que todo sistema en tiempo real requiere, como son:

a) Manejo de interrupciones externas
b) Ejecución de tareas concurrentes
c) Comunicación y sincronización entre procesos mediante las primitivas de acceso a buzones y semáforos
d) Planificación de tareas periódicas y aperiódicas, mediante las políticas de planificación Rate Monotonic y Earliest Deadline First (EDF)
e) Manejo de tiempos.

El objetivo principal de este programa es controlar los procesos de tiempo real del Kernel del Sistema Operativo utilizando las herramientas del Matlab y Simulink de cada tarea o proceso que se está ejecutando en el sistema operativo en tiempo real. Las prioridades del Sistema son proporcionar un ambiente gráfico, confiable y rápido, el sistema permite el control y monitoreo de los procesos que se encuentran en el Kernel, con las que se podrá realizar el monitoreo de funciones que todo el sistema en un periodo de tiempo determinado.

Palabras Clave: Sistema Tiempo Real, Sistema de Control, Sistema Operativo

ABSTRACT

This article describes the structure of a kernel of Real Time. The advantages of this real-time system are mainly its size, its modularity, the ease of being able to communicate with other real-time Kernels running on other computers and its interface with a graphical system for simulation of real-time tasks. The real-time Kernel is small in size (20.5Kb) and in addition it performs functions that all real-time system requires, as are:

***a)** External interrupt handling*

***b)** Implementation of concurrent tasks*

***c)** Communication and synchronization between processes using the primitives of mailbox access and traffic lights*

***d)** Planning of periodic and aperiodic tasks, through planning policies Rate Monotonic and earliest deadline First (EDF)*

***e)** Handling time.*

The main objective of this program is to control the processes of real-time Operating System Kernel using Matlab and Simulink for each task or process is running in real-time operating system. System Priorities is to provide a graphical environment, reliable and fast, the system will enable control and monitoring of the processes that are in the kernel, with which it can perform monitoring functions that any system of period of time.

Key words: Real-time system, Control System, Operating System

1. INTRODUCCIÓN

Un sistema de tiempo real es un sistema de procesamiento de información que responde a estímulos de entrada generados externamente en un período de tiempo.

Las respuestas correctas dependen no sólo de los resultados lógicos sino también del tiempo en que son entregadas.

En la actualidad el avance de la tecnología ha permitido una minimización en los dispositivos electrónicos y un incremento en la capacidad de procesamiento. Estos dispositivos se encuentran en general empotrados (embebidos) dentro de aplicaciones tales como teléfonos celulares, PALM´s, computadoras industriales, robots, satélites, automóviles y también en distintos aparatos electrónicos. La mayoría de estos dispositivos contienen un procesador y un pequeño sistema

operativo embebido el cual es capaz de controlar todo el hardware de manera eficiente. Debido a la naturaleza de estos dispositivos en ocasiones se les demanda no solo un correcto y eficiente funcionamiento si no también un estricto cumplimiento de requerimientos.

Un sistema operativo de tiempo real por lo general trabaja sobre una plataforma embebida (que puede ser un micro-controlador, un DSP (Procesador Digital de Señal) o cualquier procesador convencional. Debe ser capaz de controlar todos los recursos de hardware de la plataforma en la que se encuentra y también de administrar todos los tiempos de ejecución de las tareas de tiempo real.

El Kernel es totalmente independiente, y se encarga de controlar el hardware y los procesos que se encuentren en la máquina, o en su caso, en el sistema embebidos én donde se implemente.

Durante muchos años, los principales consumidores de sistemas de tiempo real han sido militares. Sin embargo, hoy la significativa reducción del costos del hardware han hecho posible para la mayoría de las compañías proporcionar sistemas (y productos) de tiempo real para diversas aplicaciones, por ejemplo que incluyen control de procesos, automatización industrial, investigación médica y científica, gráficos de computadoras, comunicaciones locales y de largo alcance, sistemas aeroespaciales, prueba asistida por computadora y un vasto abanico de instrumentación industrial.

Los sistemas de tiempo real se utilizan principalmente en la industria y son sistemas diseñados para funcionar en entornos con limitaciones de tiempo. Un sistema de tiempo real debe tener la capacidad para operar en forma fiable según limitaciones de tiempo específicas; en otras palabras, debe tener capacidad para procesar adecuadamente la información recibida a intervalos definidos claramente (regulares o de otro tipo).

Estos son algunos ejemplos de sistemas operativos de tiempo real:

- OS-9
- RTLinux(RealTimeLinux)
- QNX
- VxWorks

2. METODOLOGÍA

El proyecto planteado requirió inicialmente una clasificación y recopilación de información referente a los conceptos de un Sistema en Tiempo Real y como se realiza el control de los distintos procesos o tareas que se están ejecutando en el Kernel del Sistema Operativo, para una mayor fiabilidad, manejo de errores e

implementación de un adecuado análisis de los procesos empotrados.

COMET es una metodología de Tiempo Real que da un enfoque más claro al manejo de los distintos procesos que se van ejecutando en el Kernel del Sistema Operativo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga la necesidad del usuario final dentro de un tiempo y presupuesto previsible. Es una metodología de desarrollo iterativo enfocada hacia "los casos de uso, manejo de riesgos y manejo de la arquitectura".

METODOLOGÍA UTILIZADA

La metodología COMET tiene como características relevantes:

- Facilidad de uso: define el esfuerzo realizado por las personas involucradas en el desarrollo del software, para seguir la metodología hasta lograr obtener el producto esperado.
- Flexibilidad: capacidad para adaptar la metodología al problema, tomando sólo aquello que se considere necesario.
- Claridad de los diagramas: define la facilidad para la elaboración de los diagramas de parte de los involucrados en el desarrollo.
- Usa Herramientas automatizadas de Soporte: indica el uso de herramientas automatizadas para asistir a los involucrados en la generación.
- Documentación de Soporte: disponibilidad de manuales, guías y cualquier tipo de documento para la orientación acerca del uso de la metodología.
- Propicia Características de Calidad: indica las características de calidad que propicia la metodología.
- Contempla Diseño Arquitectónico: indica si la metodología dentro de sus fases, concuerda con el desarrollo de la arquitectura del sistema.
- Contempla Evaluación Arquitectónica: indica si la metodología hace uso de algún método de validación de los requisitos no funcionales (características de calidad) en etapas tempranas del desarrollo.

3. DESARROLLO

3.1 Estructura del Kernel

El Kernel soporta una arquitectura con un sólo procesador, en el cual los procesos se ejecutan concurrentemente. Inicialmente el Kernel está configurado para trabajar con un máximo de 32 procesos. Sin embargo se puede aumentar el número de los procesos a controlar por el Kernel. En cuanto al manejo de procesos, el Kernel trabaja con procesos estáticos, esto es, que desde la inicialización del sistema, los procesos ya se encuentran definidos y no cambian durante la vida de éste. Dentro del Kernel, existen dos procesos importantes llamados: primero y último los cuales son activados al iniciar el Kernel.

En la Figura 1 se presenta la arquitectura general del Kernel en tiempo real. En el nivel más bajo se encuentra los distintos dispositivos que los controla.

En el siguiente nivel están las distintas funciones del Kernel, conocidas como *primitivas*. En la parte superior, se encuentran los procesos del usuario quienes interaccionan con el Kernel. Entre el Kernel y los procesos del usuario se encuentra una interface la cual permite desligar a ambos componentes. Esta interface permite, mediante interrupciones, conectar a ambos componentes aunque estos hayan sido compilados por separado. Esta interface permite que los procesos de aplicación puedan estar en un lenguaje distinto al del Kernel.

El Kernel soporta una arquitectura de un procesador, en el cual los procesos se ejecutan concurrentemente.

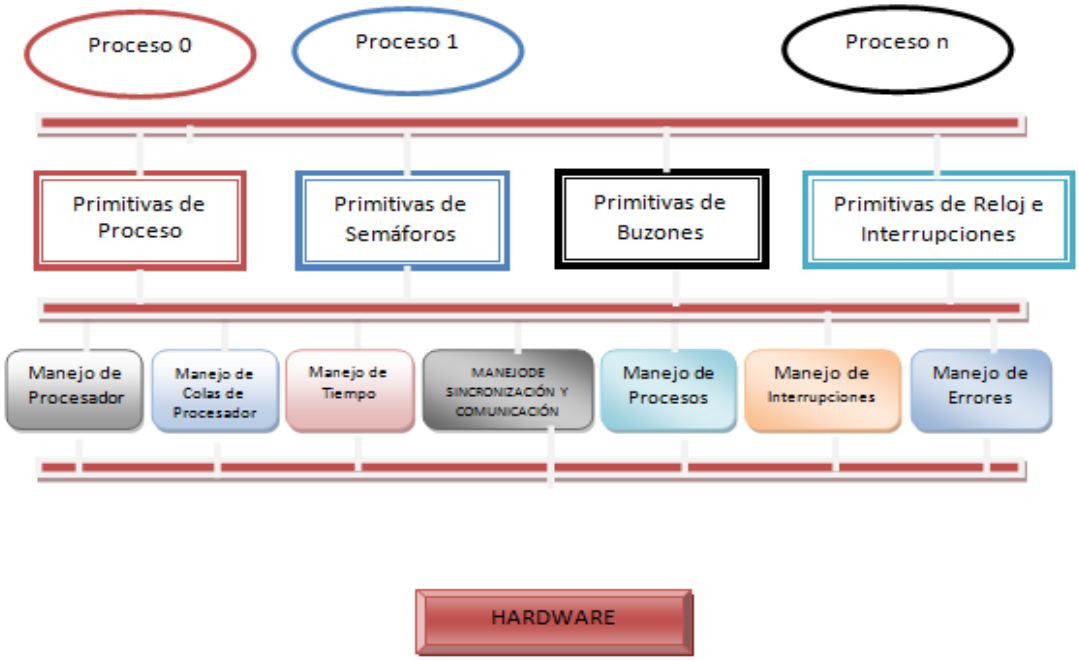

Figura 1 Arquitectura del Kernel

3.2 Estado de los Procesos

Modelo de procesos con dos estados:

- Los procesos que no estén ejecutándose se guardan en una cola de procesos, donde esperan su turno de ejecución en el procesador.
- Cada entrada de la cola es un puntero a un proceso en particular. Cuando un proceso se interrumpe, este pasa a la cola de procesos en espera. Si un proceso termina o se abandona, sale del sistema.

El modelo de procesos de cinco estados se muestra en la Figura 2

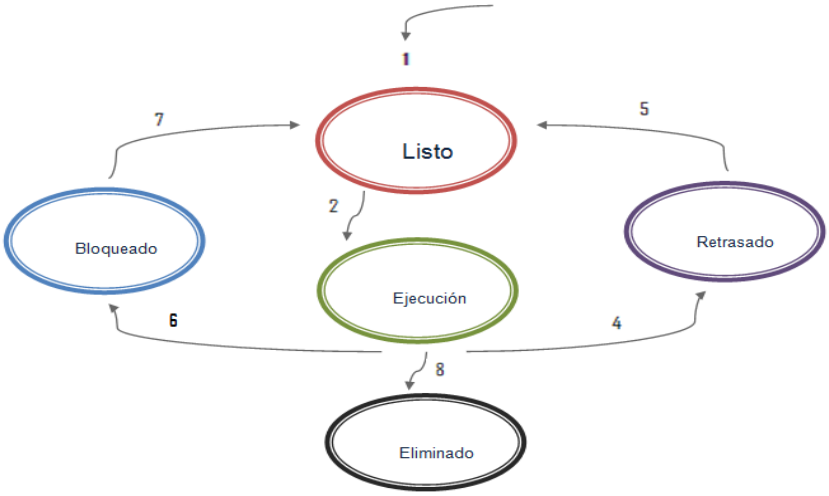

Figura 2 Estado de los Procesos

3.3 Planificación de Procesos

En las aplicaciones de Tiempo Real, lo importante es proporcionar un orden de ejecución a las tareas de tiempo real, de forma tal que todas estas cumplan con sus plazos de respuesta. Para esto, es necesario conocer con mayor precisión los tiempos de arribo, ejecución y respuesta, de cada una de las tareas del sistema.

Así mismo es necesario conocer con precisión los tiempos de los períodos de ejecución (inter-arribo de tareas) para el caso de las tareas periódicas. Las políticas de planificación permiten dar un orden de ejecución a las tareas y garantizar el correcto funcionamiento temporal de las mismas.

El Kernel desarrollado utiliza tres tipos de planificadores:

- Round Robin.
- Rate Monotonic.
- Earliest Deadline First (EDF).

El planificador Round Robin es el más simple y se dedica a atender las tareas en base a su prioridad. Las tareas en la cola de listos se encuentran ordenadas de acuerdo a su prioridad, la cual es asignada por el usuario. Las tareas con la misma prioridad son atendidas de forma Round-Robin.

La planificación Rate Monotonic asigna prioridades a las tareas en base a sus períodos. Las tareas con menor período (o mayor frecuencia de ejecución) reciben la mayor prioridad. La asignación de prioridades en la
Política Rate Monotic es estática, es decir, que se asigna una sola vez, al principio de la ejecución de las tareas, y esta no cambia durante su ejecución.

La planificación EDF asigna las prioridades de forma dinámica. La prioridad se asigna de acuerdo a la cercanía de las tareas (de su tiempo de arribo) con respecto a su plazo de respuesta. La tarea con plazo de respuesta más cercano (en un momento determinado) es la que recibe mayor prioridad. Este tipo de planificación permite que las prioridades de las tareas cambien todo el tiempo, dependiendo de su cercanía en ese momento con su plazo de respuesta.

4. DISEÑO DEL SISTEMAS DE TIEMPO REAL

El Kernel de tiempo real descrito en este artículo, es un componente de un ambiente de desarrollo de sistemas de tiempo real más amplio. El ambiente de desarrollo se muestra en la Figura 3 y consiste de los siguientes componentes:

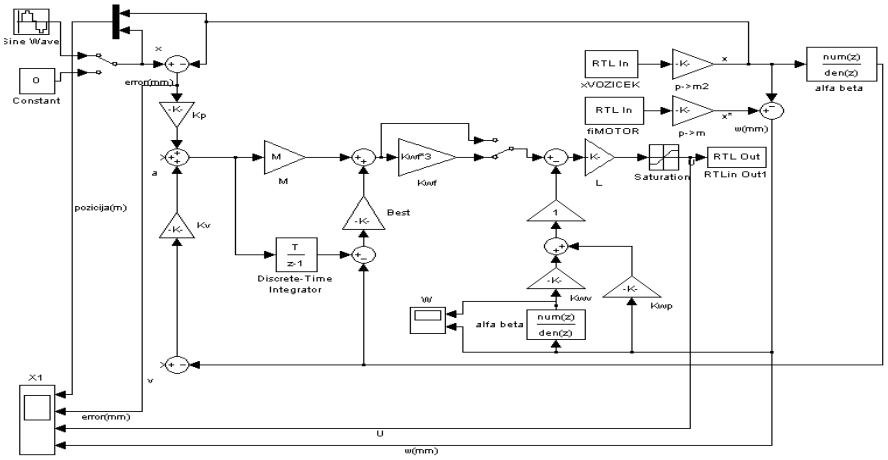

Figura 3 Sistema de Tiempo Real del Kernel del Sistema Operativo

4.1 Ambiente Visual del Sistemas de Tiempo Real

Este ambiente visual está diseñado para el monitoreo de los procesos en tiempo real en forma visual utilizando SimuLink (Herramienta para generar aplicaciones de control de procesos) como se visualiza en la figura 4

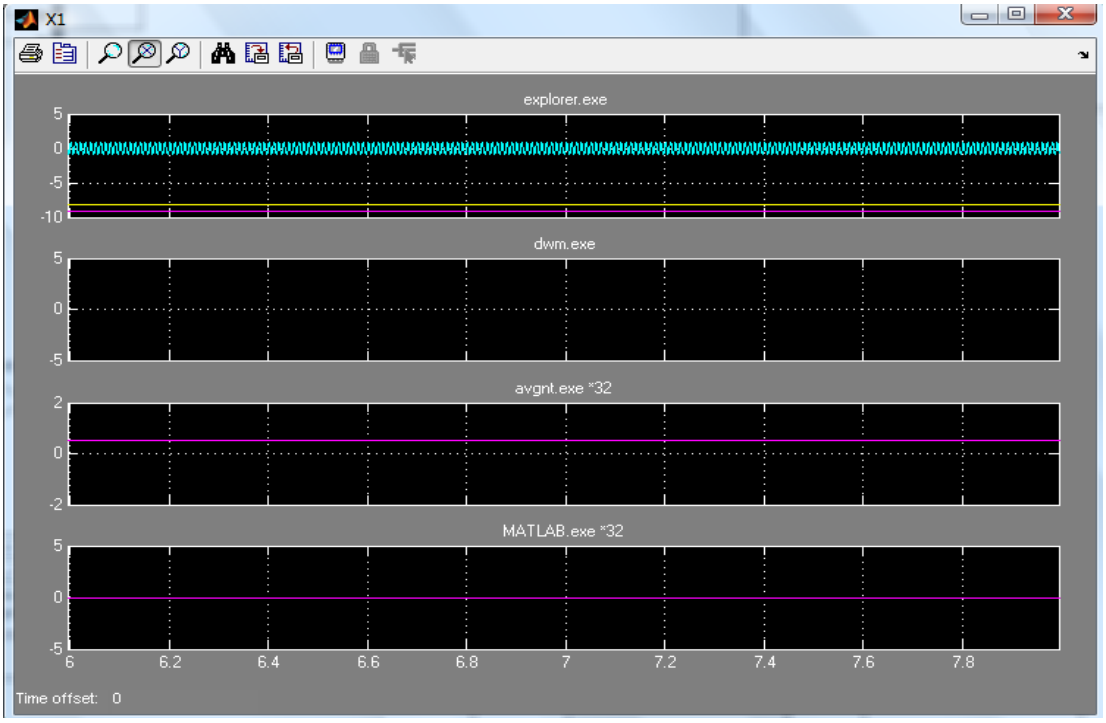

Figura 4 Procesos en Tiempo Real

La simulación de las tareas en tiempo real se ejecuta en forma estática y dinámica. En el modo estático, será posible definir un conjunto de tareas de tiempo real y graficar su comportamiento. En el modo dinámico, el Kernel se conectará en forma dinámica al ambiente de simulación a fin de presentar en forma gráfica la ejecución del Kernel.

5. CONCLUSIONES Y TRABAJO FUTURO

Una vez realizado el estudio de factibilidad del presente proyecto, el Sistema de Control en Tiempo Real para el Kernel del Sistema Operativo funciono adecuadamente.

El Sistema se basa en un registro de usuario, garantiza la confiabilidad y facilidad de uso del sistema.

Se construyó el sistema para verificar los procesos del sistema que permitirá ver procesos empotrados y los procesos que se están ejecutando en dicho CPU.

Con el fin de mejorar el rendimiento del CPU, el sistema realiza un análisis de todos los procesos del Kernel del Sistema Operativo, para un mejor rendimiento y confiabilidad, donde también se pueda

interactuar con el usuario y verificar los procesos que no se están utilizando.

6. REFERENCIAS BIBLIOGRÁFICAS

[1] http://itzamna.bnct.ipn.mx:8080/dspace/bitstream/123456789/7016/1/CONTROLTIEMPO.pdf

[2] http://dspace.espoch.edu.ec/bitstream/123456789/632/1/38T00250.pdf

[3] *mat21.etsii.upm.es/ayudainf/.../**Matlab**70/**matlab**70primero.pdf*

[4] M. Malinowski, M. P. Kazmierzowski, and A. Trzynadlowski, “Direct Power Control with virtual flux estimation for three-phase PWM rectifiers,” in *IEEE International Symposium on Industrial Electronics, ISIE-2000*, vol. 2, Puebla, Mexico, 4-8 december 2000, pp. 442–447.