ESCUELA POLITÉCNICA DEL EJÉRCITO

DPTO. DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

TÍTULO DEL PROYECTO:

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

Previa a la obtención del Título de:

INGENIERO EN SISTEMAS E INFORMÁTICA

POR:

MARCO ANTONIO ESPINEL CANGUI

SANGOLQUÍ, Junio del 2012

CERTIFICACIÓN

Certifico que el presente trabajo fue realizado en su totalidad por el Sr. MARCO ANTONIO ESPINEL CANGUI como requerimiento parcial a la obtención del título de INGENIERO EN SISTEMAS E INFORMÁTICA.

Sangolquí, Junio del 2012

_____

ING. GERMAN ÑACATO

DIRECTOR

DEDICATORIA

- Dedico esta Tesis a la memoria de mi padre Germán Espinel y especialmente a mi madre Carmen Cangui, que día a día fue el impulso para poder elaborar el presente trabajo investigativo y de esta manera haber hecho realidad el sueño de culminar mi Carrera Profesional.

- A los Catedráticos de la Escuela Superior Politécnica del Ejército que sin egoísmo alguno, brindaron sus sabias enseñanzas pero de manera especial a los Ingenieros Germán Ñacato y Freddy Tapia que me guiaron en la elaboración de esta Tesis.

Marco Espinel

AGRADECIMIENTOS

- Quiero expresar mi agradecimiento a Dios, por darme la vida y permitir disfrutar de este momento donde alcanzo una nueva meta en mi vida.

- A mi familia que me apoyaron anímica, moral, material y económicamente durante todos estos años.

- A la Escuela Superior Politécnica del Ejército, que a través de sus catedráticos vertió todos sus conocimientos en mi preparación académica.

- Es por esto que dejo plasmado mi agradecimiento en tan preciado documento que me permitió concluir con mi título de Ingeniero en Sistemas.

Marco Espinel

ÍNDICE DE CONTENIDO

LISTADO DE TABLAS

LISTADO DE CUADROS

LISTADO DE FIGURAS

Figura 4.12 ADVs No Registrado. Registrar Usuario para ingresar al Sistema.

Elaboración Propia

Figura 4.13 ADVs Ayuda. Informa que se debe hacer para ingresar al sistema.

Elaboración Propia

Figura 4.14 CONTROL DE PROCESOS. Elaboración Propia.

CAPÍTULO 1

INTRODUCCIÓN

1.1 Introducción

Un sistema de tiempo real es un sistema de procesamiento de información que responderá estímulos de entrada generados externamente en un período de tiempo y funcionamiento.

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

Los sistemas de tiempo real generan alguna acción en respuesta a sucesos externos. Para realizar esta función, ejecutan una adquisición y control de datos a alta velocidad bajo varias ligaduras de tiempo y fiabilidad. Debido a que estas ligaduras son muy rigurosas, los sistemas de tiempo real están frecuentemente dedicados a una única aplicación.

Durante muchos años, los principales consumidores de sistemas de tiempo real eran militares. Sin embargo, hoy la significativa reducción del los costos del hardware han hecho posible para la mayoría de las compañías proporcionar sistemas (y productos) de tiempo real para diversas aplicaciones, 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.''[1]

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 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 (RealTime Linux);
* QNX;
* VxWorks.

1.2 Tema

Diseño de un Sistema de Control en Tiempo Real para el Kernel del Sistema Operativo utilizando MatLab-SimuLink.

[1] **Sitio en el Web: Asistencia de Tiempo Real**

1.3 Antecedentes

En los últimos años, el procesamiento en tiempo real ha pasado a ser indispensable para un sistema operativo y, en particular, el planificador, es quizás el componente más importante de un sistema en tiempo real. Los ejemplos de aplicaciones actuales incluyen experimentos de laboratorio, control del tráfico aéreo. Los sistemas de la próxima generación serán:

Los vehículos autónomos todo terreno: Para que un vehículo autónomo pueda tomar decisiones útiles en tiempo real, los sensores están obligados a obtener información de la regeneración del medio ambiente. Algún tipo de ordenador a bordo, a menudo un pequeño microprocesador, puede tomar estos datos y determinar el estado del vehículo. Esto puede ser tan simple como el seguimiento de la ubicación actual del vehículo, pero también puede incluir otros tipos de datos de medición.

Sistemas de fabricación inteligente: Permiten implementar algunos características y mecanismos de procesamiento de los sistemas. Entre los sistemas inteligentes destacan las Redes Neuronales (Redes de Neuronas Artificiales), la Lógica Difusa y la Computación Evolutiva.

Estaciones espaciales: Es una construcción artificial diseñada para hacer actividades en el espacio exterior, con muy diversos fines. Se distingue de otra nave espacial tripulada por su carencia de un sistema de propulsión principal (en lugar de eso, otros vehículos son utilizados como transporte desde y hacia la

estación), y de medios de aterrizaje. Las estaciones espaciales están destinadas a orbitar la Tierra.

También puede definirse como un tipo de procesamiento en el que la exactitud del sistema no depende solo del resultado lógico de un cálculo sino también del instante en que se produzca el resultado. El proceso se ejecuta durante un largo periodo de tiempo, y durante ese tiempo realiza alguna función repetitiva en respuesta a algún evento de tiempo real. Las tareas intentan controlar o reaccionar ante sucesos que tienen lugar en mundo exterior, es posible asociar un plazo a una tarea en particular, donde el plazo especifica tanto un instante de comienzo como de final. Dichas tareas pueden clasificarse en rígidas o flexibles.

1.- Una tarea rígida en tiempo real debe cumplir el plazo, en otro caso producirá daños no deseados o un error fatal en el sistema.

2.- Una tarea flexible en tiempo real tiene un plazo asociado que se convierte, pero no obligatoriamente, aunque haya vencido el plazo aun tiene sentido planificar y completar la tarea.

3.- Una tarea aperiódica debe comenzar o terminar en un plazo, o puede tener una restricción tanto para el comienzo como para la finalización. Una tarea periódica, el requisito se puede enunciar como una vez por cada periodo.

1.4 Planteamiento del Problema

Se elaborará un Sistema de Control en Tiempo Real para gestionar los distintos procesos que se generan en el Kernel del Sistema Operativo, en donde sea posible visualizar las tareas que se están ejecutando en el Kernel, el control de los procesos mediante este Sistema de Control en Tiempo Real proporcionando información acerca de la ejecución de los procesos, tales como:

Nivel 1. Gestión de Memoria: Proporcionando las facilidades de bajo nivel para la gestión de memoria secundaria necesaria para la ejecución de procesos.

Nivel 2. Procesador: Se encarga de activar los quantums para cada uno de los procesos, creando interrupciones de hardware cuando no son respetadas.

Nivel 3. Entrada/Salida: Proporciona las facilidades para poder utilizar los dispositivos de E/S requeridos por cada proceso.

Nivel 4. Información o Aplicación o Interprete de Lenguajes: Facilita la comunicación con los lenguajes y el sistema operativo para aceptar las ordenes en cada una de las aplicaciones. Cuando se ejecuta un programa, el software de este nivel crea el ambiente de trabajo e invoca a los procesos correspondientes.

Nivel 5. Control de Archivos: Proporciona la facilidad para el almacenamiento a largo plazo y manipulación de archivos con nombre, va asignando espacio y acceso de datos en memoria.

Lo que permitiría una toma de decisiones para reducir costos y evitar errores.

El **Kernel**, que representa las funciones básicas del sistema operativo, como por ejemplo, la gestión de la memoria, de los procesos, de los archivos, de las entradas/salidas principales y de las funciones de comunicación. En un sistema de tiempo real el principal componente lo constituye el sistema operativo. Los Sistemas operativos debe ser capaces de controlar los recursos de hardware de la plataforma que se encuentran y también el administrar todos los tiempos de ejecución de las tareas de tiempo real. [2]

Es necesario la elaboración de un sistema que cumpla con todas las necesidades para que los procesos que están con fallos tengan una mejor repuesta de ejecución, esto llevaría hacer un mantenimiento correctivo si es necesario al ordenador.

A través de este programa se controlarán los procesos de tiempo real del Kernel del Sistema Operativo, utilizando las herramientas MATLAB Y SIMULINK, que genere automáticamente el código de cada tarea o proceso que se está ejecutando en el sistema operativo en tiempo real.

1.5 Justificación

En la actualidad el avance de la tecnología ha permitido una miniaturización en los dispositivos electrónicos y un incremento en la capacidad de procesamiento de estos dispositivos. Estos dispositivos se encuentran en general empotrados

[2] Sistemas de Tiempo Real y Lenguajes de Programación (3ª Edición) Autores.- Alan Burns; Andy Wellings

(embebidos) dentro de aplicaciones tales como teléfonos celulares, PALM's, computadoras industriales, robots, satélites, automóviles y también en distintos aparatos electro-domésticos. La mayoría de estos dispositivos contienen un procesador y un pequeño sistema operativo empotrado el cual es capaz de controlar todo el hardware de manera eficiente.

Debido a la naturaleza de estos dispositivos embebidos, en ocasiones se les demanda no solo un correcto y eficiente funcionamiento sino también un estricto cumplimiento de requerimientos temporales. A estos sistemas se les conoce como sistemas de tiempo real embebidos.

Un sistema operativo de tiempo real se ejecuta por lo general sobre un plataforma embebida (que puede ser un micro controlador, un DSP o cualquier procesador convencional). Este sistema operativo debe ser capaz de controlar todos los recursos de hardware de la plataforma que se encuentra y también de administrar todos los tiempos de ejecución de las tareas de tiempo real. Normalmente el Kernel no maneja discos, memoria cache, DMA o complejos sistemas de redes de comunicaciones, debido a que su ejecución debe ser predecible (debe ser posible estimar con exactitud sus tiempos de ejecución).

Las tareas de tiempo real son procedimientos secuenciales que se ejecutan en forma *concurrente* con otras tareas del sistema. A una tarea de tiempo real se le asocia, un tiempo de arribo, un período de ejecución y un plazo de respuesta. Una tarea de tiempo real que controla procesos críticos (en los cuales no se permite la pérdida de su plazo de respuesta).

1.6 Objetivos

Objetivo General

Desarrollar un Sistema de Control en Tiempo Real para el Kernel del Sistema Operativo utilizando MatLab-SimuLink que permita visualizar los procesos del Kernel en Tiempo Real y las tareas que se ejecutan lo que permitirá mejorar la sincronización y funcionamiento del Sistema Operativo.

Objetivos Específicos

- Realizar un Sistema para el Monitoreo de los Procesos que se ejecutan en los Sistemas Operativos, para verificar si están o no empotrados, utilizando Matlab-Simulik.
- Analizar el tiempo de respuesta que se va dando con cada proceso del Sistema Operativo, los procesos van a ser controlados para que tengan mejor tiempo de repuesta, el control se hace con la utilización de este sistema que lo conformarán Matlab-Simulik.
- Crear un ambiente de Control, en donde los procesos de tiempo real sean Monitoreados de forma visual y generar automáticamente el código de cada tarea.

1.7 Alcance

El diseño de sistemas de tiempo real, requiere de herramientas que admitan verificar y visualizar el comportamiento temporal de cada una de las tareas del Kernel del Sistema Operativo.

Con este propósito, consiste en el desarrollo de un sistema de tiempo real con herramientas visuales (Matlab-Simulink) de simulación de sistemas. En esta línea se trabajará en la integración de un kernel en tiempo determinado, de forma que sea posible hacer un seguimiento de tareas en un periodo.

Las prioridades del Sistema es proporcionar un ambiente grafico, confiable y rápido, para esto se realizará una interfaz bajo restricciones de rendimiento muy rigurosas.

El Sistema permitirá el Control y Monitoreo del los Procesos que se encuentran en el Kernel, con las que se podrá realizar monitoreo de funciones que todo sistema de periodo de tiempo determinado 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), y e) manejo de tiempos.

El Sistema no podrá monitorear procesos distintos a los que se mencionó anteriormente, a su vez el Sistema proporcionará visores adecuados para llevar un control de los procesos y permitirá la mejora en la calidad del funcionamiento de los mismos.

CAPÍTULO 2

SISTEMA DE CONTROL Y SISTEMAS DE TIEMPO REAL

2.1 Introducción Sistema de Control

Desde el punto de vista de la teoría de control, un sistema o proceso está formado por un conjunto de elementos relacionados entre sí que ofrecen señales de salida en función de señales o datos de entrada.

Es importante resaltar el hecho de que no es necesario conocer el funcionamiento interno, o cómo actúan entre sí los diversos elementos como son: Mantener el sistema estable, independiente de perturbaciones y desajustes, conseguir las condiciones de operación objetivo de forma rápida y continua, trabajar correctamente bajo un amplio abanico de condiciones operativas, manejar la restricciones de equipo y proceso de forma precisa, para caracterizar el sistema. Para ello, sólo se precisa conocer la relación que existe entre la entrada y la salida del proceso que realiza el mismo (principio de caja negra). El aspecto más importante de un sistema es el conocimiento de su dinámica, es decir, cómo se comporta la señal de salida frente a una variación de la señal de entrada. Un conocimiento preciso de la relación entrada/salida permite predecir la respuesta del sistema y seleccionar la acción de control adecuada para mejorarla. De esta manera, el diseñador, conociendo cuál es la dinámica deseada, ajustará la acción

de control para conseguir el objetivo final frente a perturbaciones externas del sistema.[3]

En vista de todo lo expuesto, se puede definir un sistema de control como el conjunto de elementos que interactúan para conseguir que la salida de un proceso se comporte tal y como se desea, mediante una acción de control como se puede observar en la Figura 2.1.

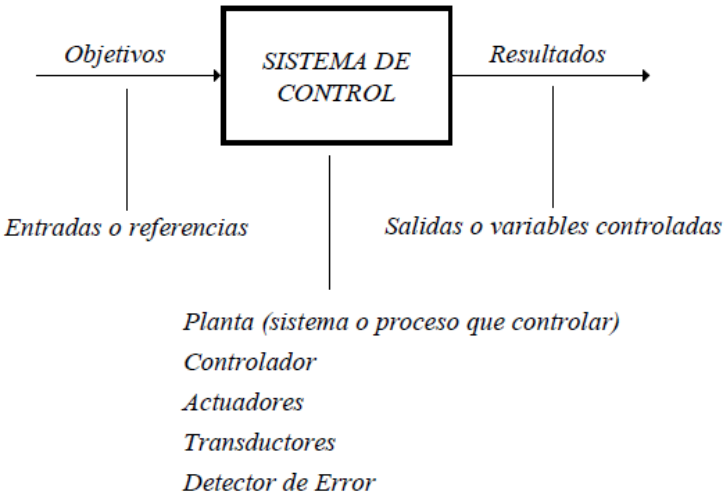


Figura. 2.1 Diagrama de un sistema de control

[3] **Sitio en la Web**

http://www.edicionsupc.es/ftppublic/pdfmostra/EE00301M.pdf

2.2 Introducción al Control

Esencialmente, controlar implica la medición y la corrección de las actividades de los procesos para asegurarse de que se están llevando a cabo los planes para alcanzar los objetivos fijados.

Esta definición simple de control muestra que es prácticamente el mismo proceso sin importar que actividad se considere. La esencia de la mayor parte del control es cierta clase de retroalimentación. Las principales técnicas de control deben ser clasificadas como tradicionales, en el sentido de que han sido utilizadas durante mucho tiempo por los Sistemas Operativos.

2.2.1 Sistemas de Control en Lazo Abierto

La acción de control se calcula conociendo la dinámica del sistema, las consignas y estimando las perturbaciones. Esta estrategia de control puede compensar los retrasos inherentes del sistema anticipándose a las necesidades del usuario. Sin embargo, el lazo abierto generalmente es insuficiente, debido a los errores del modelo y a los errores en la estimación de las perturbaciones. Por ello, es común la asociación de lazo cerrado, lazo abierto, de modo que el lazo cerrado permite compensar los errores generados por el lazo, abierto como se indica en la figura2.2.

2.2.2 Sistema de Control en Lazo Cerrado

La acción de control se calcula en función del error medido entre la variable controlada y la consigna deseada. Las perturbaciones, aunque sean desconocidas son consideradas indirectamente mediante sus efectos sobre las variables de salida. Este tipo de estrategia de control puede aplicarse sea cual sea la variable controlada. La gran mayoría de los sistemas de control que se desarrollan en la actualidad son en lazo cerrado, como se indica en la figura 2.2.

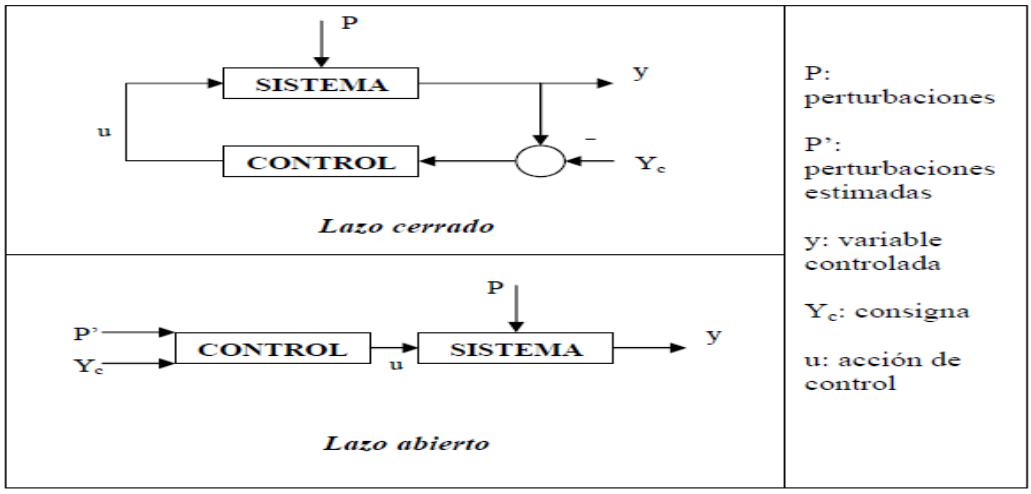


Figura 2.2 *Control lazo cerrado vs. Control lazo abierto*

2.3 Retroalimentación y sus Efectos

El uso de la retroalimentación tiene el propósito de reducir el error entre la entrada de referencia y la salida del sistema. Sin embargo, la importancia de los efectos de la retroalimentación en los sistemas de control es mucho más importante. La reducción del error del sistema es solamente uno de los diferentes efectos importantes que la retroalimentación tiene en un sistema. La retroalimentación también tiene efectos en las características de desempeño del sistema tales como estabilidad, ganancia total, sensibilidad y reducción de errores.

En general, se puede afirmar que, cuando las variables de un sistema exhiben una secuencia cerrada de *relaciones de causa y efecto*, el sistema cuenta con una retroalimentación, como se visualiza en la figura 2.3.

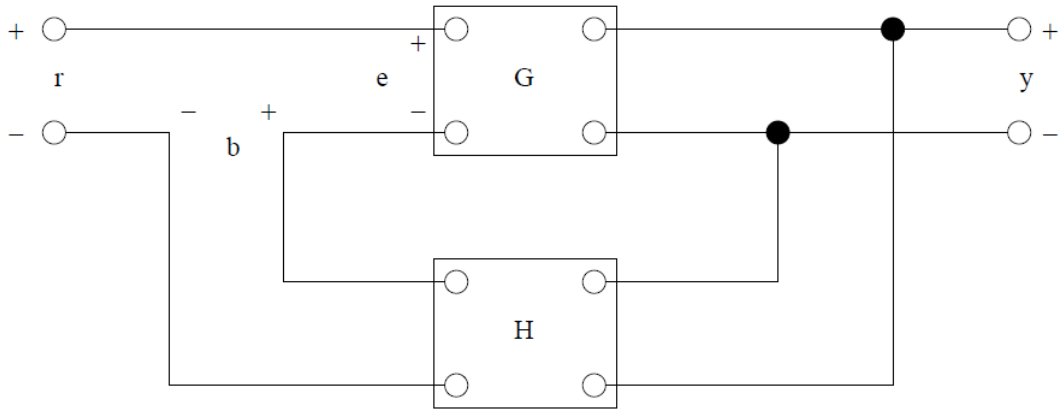


Figura 2.3 Retroalimentación y sus Efectos

Se puede investigar los efectos de la retroalimentación sobre diversos aspectos del desempeño de un sistema. En este punto, sin contar con los conocimientos generales necesarios y sin haber estudiado las bases matemáticas de la teoría de los sistemas lineales, solo se puede aplicar una notación estática simple para la discusión. Tomando como referencia la configuración del sistema de retroalimentación simple que se muestra en la figura 2.3, donde *r* es la señal de entrada, *y* es la señal de salida, *e* es el error y *b* es la señal de retroalimentación. Los parámetros *G* y *H* pueden considerarse como ganancias constantes. Mediante operaciones algebraicas simples, se puede demostrar que la relación entrada-salida de un sistema es:

$$M = \frac{y}{r} = \frac{G}{1 + GH} \tag{1}$$

Esta relación básica de la estructura del sistema retroalimentado permite conocer más a fondo los efectos importantes de la retroalimentación.

Como puede apreciarse en la ecuación 1, la retroalimentación afecta a la ganancia *G* de un sistema sin retroalimentación por un factor de 1+*GH*. La referencia de la retroalimentación en el sistema de la figura 2.3 es negativa, pues a la señal de retroalimentación se le asigna un signo menos. La cantidad *GH* puede incluir en sí misma un signo negativo, por lo que el efecto general de la retroalimentación es que puede incrementar o reducir la ganancia. En un sistema de control práctico, *G* y *H* son funciones de frecuencia, por lo que la magnitud de 1 + *GH* puede ser mayor que 1 en un intervalo de frecuencia pero inferior a 1 en otros.

Por consiguiente, la retroalimentación puede aumentar la ganancia del sistema en un intervalo de frecuencia y disminuirlo en otro.

2.4 Tipos de Retroalimentación

Los elementos esenciales que aparecen en un sistema de control por realimentación:

Primero.- Un elemento que mide las variables de estado ("output").

Segundo.- Un medio de comparar esa salida con el valor deseado para la misma.

Tercero.- Un método de realimentar esta información a la entrada (variables de control) de tal forma que se minimiza la desviación de la salida respecto al nivel deseado.

Los sistemas o procesos de control suelen ser representados de modo conveniente mediante diagramas funcionales en los que se visualiza el papel de cada uno de los órganos del sistema. Un ejemplo se puede visualizar en la figura2.4

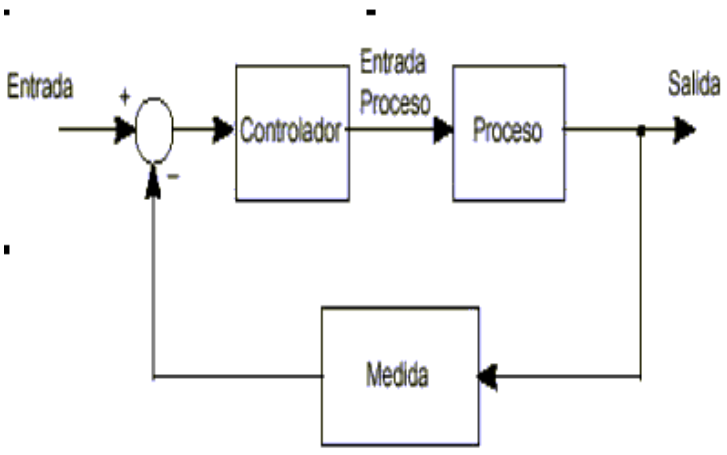


Figura 2.4 Diagramas Funcionales

Se representa en la figura 2.4 un proceso físico, mecánico, biológico, etc., con una entrada previsible dentro de ciertos límites, pero no exactamente, y una salida deseable qD. El valor real de la salida q0 es detectado por una unidad de medida que envía una señal a un elemento diferenciador. Este mide la diferencia o error qD-q0 y transmite una señal a la unidad controladora, la cual actúa sobre el proceso de forma adecuada a fin de anular dicho error. Obviamente, la estabilidad es una cualidad deseable de cualquier sistema de control. Es necesario que la perturbación que se efectúa en los controles a fin de corregir el error de desviación en la salida no cause una alteración excesiva en sentido contrario al de dicha desviación. De ser así, el error del proceso pasaría alternativamente de un sentido al otro, desvirtuándose el sistema de control en su propia finalidad. Un sistema de control inestable puede ejemplificarse en la marcha de un aprendiz de ciclista. Un pequeño error inicial de dirección y equilibrio es corregido con intensidad creciente, acabando inexorablemente el recorrido con una caída.

2.5 Resumen Sistema de Control

Un sistema puede definirse conceptualmente como un ente que recibe unas acciones externas o variables de entrada, y cuya respuesta a estas acciones externas son las denominadas variables de salida.

Las acciones externas al sistema se dividen en dos grupos, *variables de control*, que se pueden *manipular, y perturbaciones* sobre las que no es posible ningún tipo de control. Dentro de los sistemas se encuentra el concepto de sistema de control. Un sistema de control es un tipo de sistema que se caracteriza por la presencia de una serie de elementos que permiten influir en el funcionamiento del sistema. La finalidad de un sistema de control es conseguir, mediante la manipulación de las variables de control, un dominio sobre las variables de salida, de modo que estas alcancen unos valores prefijados (consigna).

Un sistema de control ideal debe ser capaz de conseguir su objetivo cumpliendo los siguientes requisitos:

1. Garantizar la estabilidad y, particularmente, ser robusto frente a perturbaciones y errores en los procesos del Sistema.
2. Ser tan eficiente como sea posible, según un criterio preestablecido para cada uno de los procesos del Kernel, normalmente este criterio consiste en que la acción de control sobre las variables de entrada sea realizable, evitando que los procesos se queden estancados o aislados.
3. Fácilmente implementar y cómodo de operar en tiempo real con ayuda de un ordenador.

2.6 Introducción Sistemas de Tiempo Real

La entidad de software más importante de cualquier sistema operativo, es *el proceso*. Un proceso o tarea es un cálculo que es ejecutado por el procesador o también podemos mencionar del Kernel que es el Núcleo del Sistema Operativo, de una forma secuencial. Cuando kernel tiene que ejecutar un conjunto de tareas, el procesador tiene que ser asignado a varias tareas de acuerdo a un criterio u orden de ejecución predefinido. El conjunto de reglas que determina el orden en el cual las tareas son ejecutadas es llamado algoritmo de planificación.

Una tarea que está lista para ejecutarse, estará en ejecución si está ha sido seleccionada por el planificador, o se encontrará en espera si alguna otra tarea de mayor prioridad se encuentra en ejecución. Una tarea que puede potencialmente ejecutarse en el kernel y se la llama tarea activa. Una tarea esperando en el procesador es llamada tarea lista, y todas las tareas esperando por el procesador son mantenidas en una cola de espera llamada cola de listos. El planificador elige el orden de ejecución de las tareas basándose en la política establecida por el algoritmo de planificación elegido.

Las tareas también pueden estar en estado de espera por tiempo o espera por un recurso físico. En estos estados las tareas solo estarán listas para ejecución hasta que se termine el tiempo de espera o se desocupe el recurso físico por el que están esperando. Las tareas en estado de espera por tiempo se mantienen en una cola que se denomina cola de procesos retrasados por tiempo mientras que las

tareas en estado de espera por recurso se mantienen en una cola de procesos esperando por el recurso.

En la mayoría de los sistemas operativos que permiten la activación dinámica de tareas, la tarea en ejecución puede ser interrumpida en cualquier parte de su código, con el fin de que una tarea de mayor prioridad pueda acceder. La operación de suspender la tarea en ejecución e insertarla dentro de la cola de listos es llamada desalojo.

Un sistema de tiempo real es un sistema informático que interacciona con su entorno, sobre el que realiza acciones de control que se producen dentro de intervalos de tiempo bien definidos, como se indica en la Figura 2.5

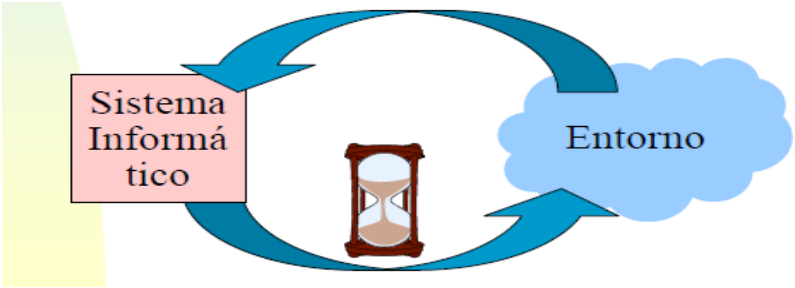


Figura 2.5 Sistema de Tiempo Real

"Un sistema operativo de tiempo real (SOTR o RTOS -*Real Time Operating System* en inglés), es un sistema operativo que ha sido desarrollado para aplicaciones de tiempo real. Como tal, se le exige corrección en sus respuestas bajo ciertas restricciones de tiempo. Si no las respeta, se dirá que el sistema ha fallado. Para garantizar el comportamiento correcto en el tiempo requerido se necesita que el sistema sea predecible."[4]

[4] Sitio Web

2.6.1 Introducción Tiempo Real

Se dice que un ordenador trabaja en tiempo real cuando realiza una transacción que le ha sido ordenada desde un terminal en ese mismo momento, sin espera alguna.

Para tener una definición exacta de lo que es tiempo real, tenemos que ver que todos los sistemas en tiempo real tienen restricciones temporales. La palabra tiempo significa que el correcto funcionamiento del sistema, depende no solo del resultado lógico sino también depende del tiempo en que se produce un cambio o un resultado. Y la palabra real dentro de los sistemas se refiere a la reacción que esta tiene a eventos externos que se realizan durante su funcionamiento. En otras palabras se puede decir tiempo real a recibir un dato, información, en el menor tiempo posible.

Por ejemplo:

Tiempo real en Internet: Los datos en tiempo real en Internet no necesariamente significan que la información está disponible en el momento en el que se recopila o en el momento en el que se envía, sino que todo dato y/o información que se actualiza en forma regular que cambia con frecuencia.

http://es.wikipedia.org/wiki/Sistema_operativo_de_tiempo_real

Por ejemplo las imágenes satelitales nuevas se actualizan cada hora se le conoce como “datos en tiempo real”.

Tiempo real en la TV: Este tipo de televisión se da a través de la Internet supuestamente en tiempo real pero siempre hay un retardo por más pequeño que sea.

Tiempo Real en el Sistema Operativo: El *kernel* o *núcleo* del sistema operativo es la porción más pequeña de sistema operativo que provee tres funciones.

Cada una de estas tres funciones del kernel multitarea está asociada a un problema diferente. Y dichos problemas tienen diferentes soluciones. Por ejemplo, si se dispone de tres procesos listos para ser ejecutados, ¿En qué orden deben ser ejecutados? , ¿Cómo se reparte el tiempo de CPU entre los tres?, ¿Se ejecuta primero el más urgente? Si se elige esta última opción , ¿cuál es el más urgente? .

2.6.2 Tipos de Restricciones

Las restricciones típicas que pueden ser especificadas sobre las tareas en tiempo real son las siguientes: restricciones de tiempo, restricciones de precedencia y restricciones de exclusión mutua sobre recursos compartidos.

Restricciones en tiempo: Los sistemas en tiempo real están caracterizados por actividades computacionales (tareas o procesos) con restricciones severas de tiempo que deben completarse en orden para alcanzar el comportamiento deseado.

Una restricción en tiempo que presentan las tareas es el plazo de respuesta, el cual representa el tiempo antes del cual una tarea debe completar su ejecución sin causar ningún daño del sistema. Dependiendo de las consecuencias que se presenten por la pérdida de los plazos, las tareas de tiempo real se distinguen normalmente en dos clases:

Duras: Se dice que una tarea es dura si la pérdida del plazo de respuesta puede causar consecuencias catastróficas dentro del sistema. En este caso, cualquier instancia de las tareas debe garantizar a priori el cumplimiento del plazo de las tareas. Esto se hace considerando el peor caso del tiempo de cómputo de las tareas.

Suaves: Se dice que una tarea es suave si la pérdida de un plazo de respuesta no pone en peligro a personas o causa pérdidas económicas. En los sistemas de tiempo real suaves, la pérdida de plazos produce únicamente una disminución en el desempeño del sistema.

En general, una tarea de tiempo real puede ser caracterizada por los siguientes parámetros:

Tiempo de llegada: Es el tiempo en el cual una tarea está lista para su ejecución.

Tiempo de cómputo en el peor caso: Es el tiempo de cómputo necesario para ejecutar la tarea sin interrupción.

Plazo de respuesta máxima: Es el tiempo límite antes del cual una tarea debe completarse para evitar daño al sistema.

Tiempo de comienzo: Es el tiempo en el cual una tarea inicia su ejecución.

Tiempo de finalización: Es el tiempo en el cual una tarea termina su ejecución.

Criticidad: Es un parámetro relacionado a la consecuencia de la pérdida de plazos de respuesta, o se relaciona también con la importancia de las tareas en el sistema.

Latencia: representa el retraso en la terminación de una tarea con respecto a su plazo de respuesta.

Los parámetros temporales son ilustrados en la figura 2.6. Otra característica que puede ser especificada dentro de las tareas en tiempo real concierne la regularidad de su activación.

En particular, las tareas pueden ser definidas como periódicas o aperiódicas. Las tareas periódicas consisten de una secuencia infinita de actividades de cómputo, llamadas instancias, que son activadas regularmente a una frecuencia fija. Las tareas aperiódicas se caracterizan por consistir de una sola instancia y tener tiempo de arribo desconocido. El tiempo de cómputo de una tarea aperiódica se conoce solo hasta que la tarea arriba.

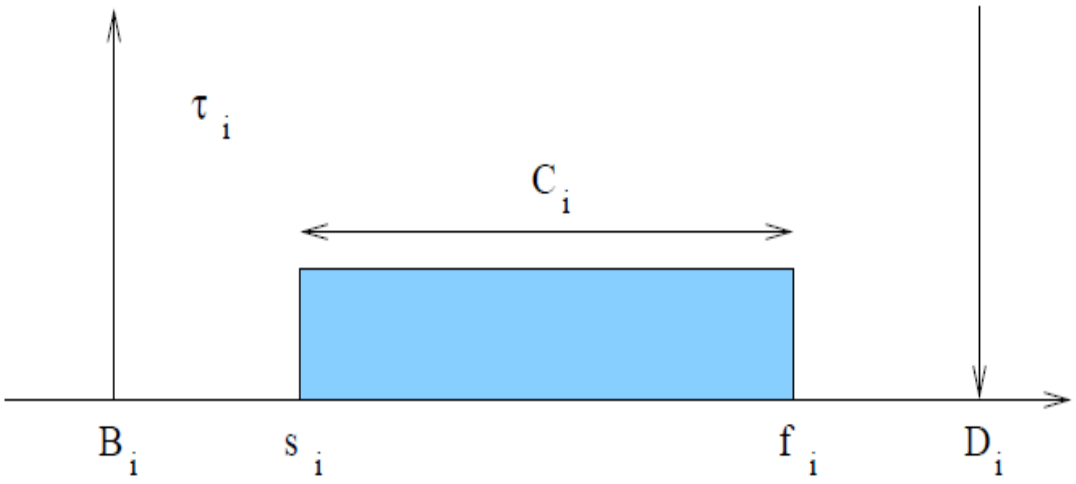


Figura 2.6: Parámetros típicos de una tarea en tiempo real

Restricciones de precedencia

En ciertas aplicaciones, las actividades computacionales no pueden ser ejecutadas en orden arbitrario, si no que tienen que respetar algunas relaciones de precedencia definidas en la etapa de diseño.

Restricciones sobre recursos

Desde el punto de vista de un proceso, un recurso es cualquier dispositivo de entrada/salida o estructura de software que puede ser utilizada por el proceso. Típicamente, un recurso puede ser una estructura de datos, un conjunto de variables, una área de memoria, un archivo, una porción de un programa o un dispositivo periférico. Un recurso que es dedicado exclusivamente a un proceso en particular se dice que es un recurso privado, mientras que un recurso que puede ser utilizado por dos o más tareas es llamado recurso compartido.

Para mantener la consistencia de datos en la mayoría de recursos compartidos, no se permite el acceso simultáneo por dos o más tareas a estos datos, sino que se requiere del cumplimiento de la condición de exclusión mutua entre las tareas que compiten por este recurso.

2.6.3 Definición del Problema de Planificación

Para definir un problema de planificación se necesita especificar los siguientes tres conjuntos:

Un conjunto de *n* tareas T = *T*1*; T*2*; : : : ; Tn*, un conjunto de *m* procesadores A = *p*1*; p*2*; : : : ; pm* y un conjunto recursos *R* = *r*1*; r*2*; : : : ; rs*. Además, es necesario especificar las restricciones de tiempo y de precedencia entre tareas. En este contexto, la planificación permite asignar los procesadores del conjunto A y los recursos del conjunto *R* a tareas del conjunto T de acuerdo a un orden que permita completar todas las tareas bajo las restricciones impuestas. Se ha demostrado que este problema es de tipo NP-completo, lo cual implica que su solución es muy difícil de obtener.

Para reducir la complejidad en la construcción de planificadores, podemos simplificar la arquitectura de la computadora, por ejemplo, restringiendo al sistema para que utilice un solo procesador, adoptar un modelo con desalojo, utilizar prioridades fijas, eliminar precedencias o restricciones de recursos, asumir una activación simultanea de tareas y suponer que en el sistema existen conjuntos de tareas homogéneos (únicamente periódicas o únicamente aperiódicas).

2.6.4 Clasificación de los Algoritmos de Planificación

De entre la gran variedad de algoritmos propuestos para la planificación de tareas en tiempo Real en el kernel, podemos identificar las siguientes clases:

Planificación con desalojo. La tarea en ejecución del Kernel puede ser interrumpida en cualquier momento para asignar al Kernel a otra tarea, de acuerdo a una política de planificación predefinida.

Planificación sin desalojo. Una tarea una vez iniciada, es ejecutada por el procesador hasta que termine su actividad. En este caso, todas las decisiones de planificación son tomadas cuando una tarea inicia o termina su ejecución.

Planificación estática. Los algoritmos estáticos son aquellos en los cuales las decisiones de planificación están basadas en parámetros fijos, los cuales son asignados a las tareas antes de su activación.

Planificación dinámica. Los algoritmos dinámicos son aquellos en los cuales las decisiones de planificación están basadas en parámetros dinámicos que pueden cambiar durante la ejecución del sistema.

Planificación fuera de línea. Un algoritmo de planificación es usado fuera de línea si es ejecutado sobre el conjunto completo de tareas antes de la ejecución

actual de las tareas. El planificador generado en esta forma, es almacenado en una tabla y después ejecutado por un despachador.

Planificación en línea. Un algoritmo de planificación es usado en línea si las decisiones de planificación son tomadas a tiempo de ejecución cada vez que una nueva tarea llega o cuando una tarea termina su ejecución.

Planificación óptima. Un algoritmo es óptimo si minimiza algunas funciones de costo definidas sobre el conjunto de tareas. Cuando ninguna función de costo es definida y lo único concerniente es alcanzar una planificación factible, entonces se dice que un algoritmo de planificación es óptimo si este encuentra una solución de planificación que no puede ser contra decida por ningún otro algoritmo de planificación.

La solución de planificación indica si un conjunto de tareas es factible o no factible cumple o no con sus plazos de respuesta.

Planificación heurística. Un algoritmo heurístico encuentra una solución aproximada que intenta estar cerca de la solución óptima.

2.6.5 Planificación de Tareas Periódicas

En la mayoría de las aplicaciones de control en tiempo real, las actividades periódicas representan la mayor demanda computacional del sistema. Las tareas periódicas típicamente realizan acciones como la adquisición de datos, lazos de control y monitoreo del sistema en este caso del Kernel.

Tales actividades necesitan ejecutarse en forma cíclica a frecuencias fijas, estas frecuencias pueden deducirse de los requerimientos del sistema específicamente del Kernel.

2.6.6 Factor de Utilización

Factor de Utilización (U)

Dado un conjunto *T* de *n* tarea periódicas, el factor de utilización del procesador *U* es la fracción de tiempo consumido por el procesador en la ejecución del conjunto de tareas. Dado que Ci / Pi es la fracción de tiempo que consume el procesador en ejecutar la tarea *Ti*, el factor de utilización para *n* tareas está dado por:

$$U_i = \frac{C_i}{T_i} \qquad (2)$$

De la ecuación (2) se tiene que el factor de utilización para un conjunto de tareas es:

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

(3)

Es uno de los algoritmos de planificación más antiguos, sencillo y usado, en este cada proceso tiene asignado un intervalo de tiempo de ejecución llamado *Quantum*. FIFO Round Robin es muy sencillo de implementar, todo lo que necesita el planificador es conocer la cola de los procesos listos y una vez que el proceso en ejecución consume su Quantum, se le quita el procesador y se le asigna al siguiente en la cola, colocando al proceso que salió al final de la cola de procesos listos.

El análisis de planificabilidad para este algoritmo consiste únicamente en verificar si su factor de utilización excede al 100 % (U ≤ 1).

2.6.7 Planificación Monótona de Frecuencias

El método con mayor perspectiva para la resolución de conflictos de la planificación multitarea con tarea periódica es la planificación monótona de frecuencia (RMS, Rate Monotonic Scheduling). El RMS lo que hace es asignar prioridades a las tareas en función de sus periodos.

En la figura 2.7 muestras los parámetros relevantes de las tareas el periodo ***T*** de las tareas es el tiempo que transcurre entré la llegada de una tarea y la siguiente llegada de la misma tarea. La frecuencia de una tarea (en hercios)es simplemente, la inversa de su periodo (en segundos).

Proceso	Instante de llegada	Tiempo de ejecución	Plazo de inicio
A	10	20	110
B	20	20	20
C	40	20	50
D	50	20	90
E	60	20	70

Tabla 2.1 Perfil de ejecución de cinco tareas aperiódicas

El tiempo ***C*** de ejecución (o computación) es el tiempo de procesamiento de cada acontecimiento de una tarea. Debe quedar claro que, en un sistema monoprocesador, el tiempo de ejecución no debe ser mayor que el periodo ($C \leq T$). Si una tarea periódica se ejecuta siempre hasta el final, es decir si nunca se le niega el servicio por insuficiencia de recursos la utilización del procesador por parte de la tarea es $U = C / T$ es decir si una tarea tiene un periodo de 80 minutos y un tiempo de ejecución de 55 minutos su utilización del procesador será de $55 / 80 = 0,6875$. Como se visualiza en la Tabla 2.1

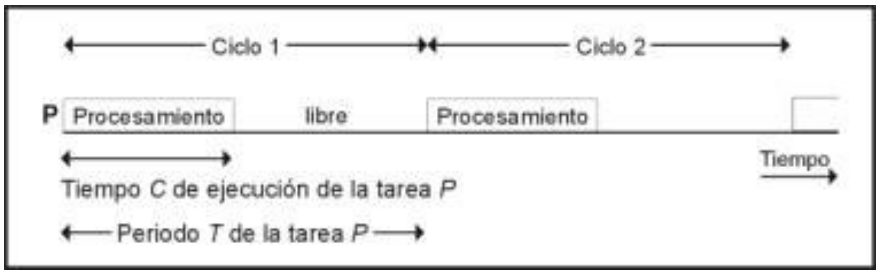


Figura 2.7 Diagrama de tiempos de tareas periódicas.

En RMS, la tarea de más alta prioridad es la del periodo más corto, la segunda de mayor alta prioridad es la del segundo tiempo más corto y así sucesivamente. Cuando una tarea se encuentra disponible para ser ejecutada, se da servicio primero a la que tenga el periodo más corto. Si se dibuja la prioridad de las tareas en función de sus frecuencias el resultado que se obtiene en una función monótona creciente de aquí es de donde proviene el nombre de planificación monótona de frecuencia.

Una medida para comprobar la efectividad de un algoritmo de planificación periódico esta en ver si garantiza o no que se cumpla todos los plazos rígidos de las tareas. Suponga que se dispone de ***n*** tareas cada una de las cuales tiene un periodo y un ***0*** tiempo de ejecución fijos. La figura 2.8 y la tabla 2.2 muestran las características de las prioridades y el valor límite del los RMS.

Entonces para que sea posible cumplir todos los plazos debe cumplirse que:

$$\frac{C1}{} + \frac{C2}{} + \frac{C3}{} + \ldots\ldots\ldots + \frac{Cn}{} \leq 1$$

T1 T2 T3 Tn

$$\frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_3} + \ldots\ldots + \frac{C_n}{T_n} \;?\; 1$$

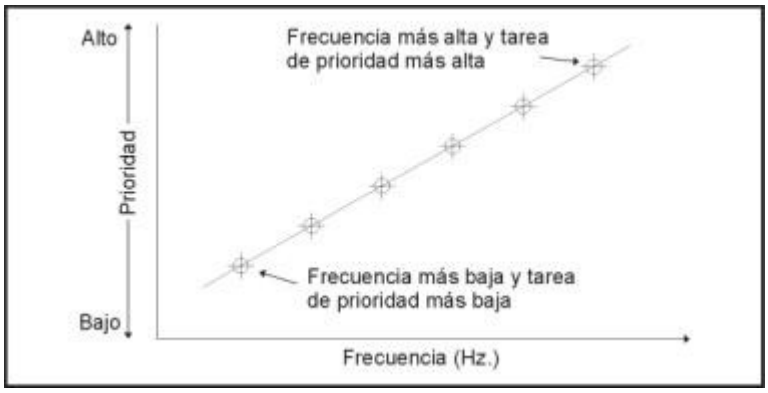


Figura 2.8 Una tarea con RMS.

Tabla 2.2 Valor del límite superior RMS.

n	$n(2^{1/n}-1)$
1	1,0
2	0,828
3	0,779
4	0,756
5	0,743
6	0,734
⋮	⋮
∞	ln 2 ≈ 0,693

La suma de las utilizaciones del procesador por parte de las tareas individuales no puede exceder un valor de 1, que corresponde a la utilización total del procesador esto lo que permite es proporcionar un límite para el número de tareas que puede planificar con éxito un algoritmo de planificación perfecto. Para un algoritmo en particular, el límite puede ser inferior.

Para RMS, se puede demostrar que se cumple la siguiente desigualdad:

$$\frac{C1}{} + \frac{C2}{} + \frac{C3}{} + \ldots\ldots\ldots + \frac{Cn}{} \leq n \quad (2 - 1)$$

T1 T2 T3 Tn

$$\frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_3} + \ldots\ldots\ldots + \frac{C_n}{T_n} \quad ? \quad n(2 - 1)$$

La principal ventaja que ofrece este tipo de planificación es que la estabilidad se consigue fácilmente. Cuando un sistema no puede cumplir todos los plazos, debido a la sobrecarga o a errores transitorios, es necesario poder garantizar que se cumplirán los plazos de las tareas fundamentales, dando por supuesto que este grupo de subtareas se puede planificar. En el método de asignación estática de prioridades, basta con asegurar que las tareas fundamentales reciban prioridades relativamente altas, esto se hace en el RMS estructurando las tareas fundamentales para que tengan periodos cortos o modificando las prioridades del RMS para que tengan en cuenta dichas tareas.

2.6.8 Planificación Earliest Deadline First

El algoritmo de planificación EDF (Earliest Deadline First) es otro algoritmo de planificación para sistemas en tiempo real el cual utiliza el plazo de respuesta de las tareas como la base para asignar prioridades.

En esta política de planificación, la tarea con plazo más cercano es quien recibe la mayor prioridad de ejecución. Bajo este algoritmo, la condición necesaria y suficiente para que un conjunto de tareas periódicas tenga una planificación viable es:

(U≤1)

(6)

EDF consigue un factor de utilización del 100% para los conjuntos de tarea que planifica, por lo que se puede decir que es óptimo globalmente, es decir, que si existe un algoritmo que proporcione una planificación factible con un determinado conjunto de tareas periódicas, entonces EDF también proporciona una planificación factible para dicho conjunto de tareas.

2.6.9 Ejemplo

“Como se aprecia en la figura 2.9 se ha creado una aplicación que consiste de 4 tareas de tiempo real.

La prioridad de las tareas es la misma. La tarea 1 representa un reloj en movimiento. Las tareas 2 y 3 ejecutan un juego en el cual un carácter rebota sobre

una ventana. Finalmente la tarea 4 consiste en un editor de caracteres que se ejecuta dentro de su ventana. La política de planificación utilizada fue FIFO Round-Robín.

Durante la ejecución de esta aplicación, el Quantum utilizado fue de 5 mili-segundos. Con esta resolución, fue posible observar que cada tarea se ejecuta como si estuviera en un procesador diferente."[5]

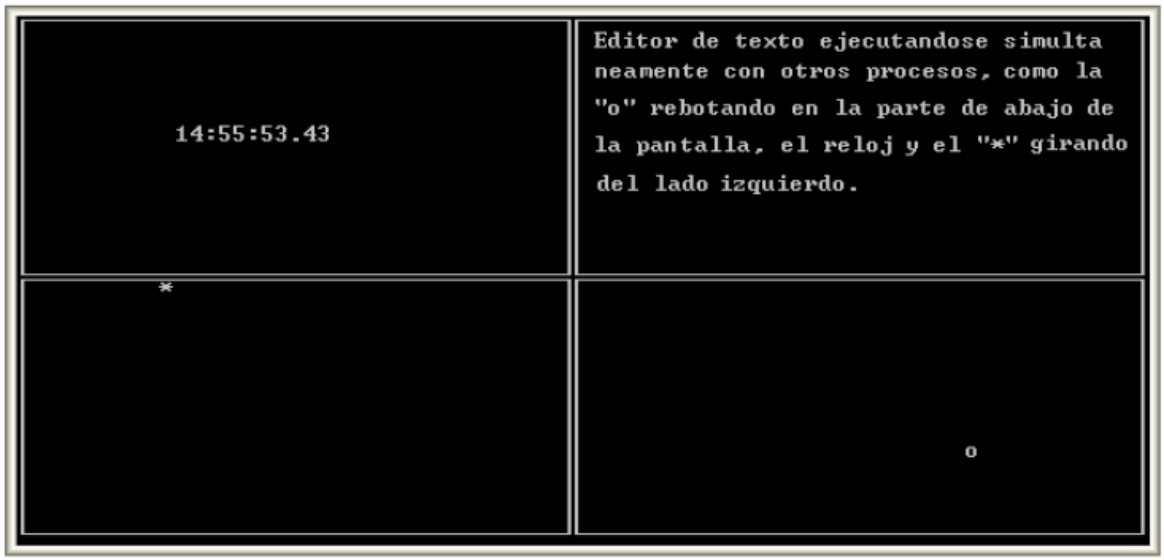


Figura 2.9: Muestra de 4 tareas ejecutándose sobre el Kernel

[5] http://delta.cs.cinvestav.mx/~pmalvarez/miranda-mejia.pdf

2.6.10 Manejo de Sobrecargas

EDF **(Earliest Deadline First)** puede degradar rápidamente su desempeño durante intervalos de sobrecarga. Esto es debido al hecho que EDF da la más alta prioridad a aquellas tareas que estén más cerca de perder sus plazos. Existen casos en los cuales la llegada de una nueva tarea puede causar que todas las tareas pierdan sus plazos, esté es un fenómeno indeseable llamado *efecto Dómino*.

Para evitar el efecto dómino, el sistema operativo y el algoritmo de planificación deben estar diseñados para manejar sobrecargas transitorias en una forma controlada, de manera que el daño debido a la pérdida de plazos pueda minimizarse.

En un ambiente de tiempo real duro, un sistema está sobrecargado cuando, basado en suposiciones del peor caso, no existe planificación factible para el conjunto total de tareas, por lo cual una o más tareas perderán sus plazos de respuesta.

2.6.11 Resumen Sistema de Tiempo Real

En este capítulo se describen los conceptos más importantes de los sistemas de tiempo real, los cuales son utilizados en el resto de la tesis. En este capítulo se enumeraron los diferentes tipo de restricciones que pueden especificarse sobre las

tareas de tiempo real, de las cuales la más importantes para este estudio son las restricciones de tiempo, esto es, si las tareas son duras o suaves.

Se definió también el problema de planificación, así como la clasificación de los algoritmos de planificación. Se describió el concepto de factor de utilización ampliamente utilizado en esta tesis, y se describieron los algoritmos de planificación más utilizados en los sistemas de tiempo real, el EDF (*Earliest Deadline First*) y el RM (*Rate Monotonic*), con un ejemplo donde se explica la planificación de estos dos algoritmos sobre un conjunto de dos tareas. El sistema desarrollado en esta tesis, contempla la implementación de las políticas de planificación EDF y RM. Mediante simulación observaremos el comportamiento del sistema propuesto con las dos políticas de planificación, implementadas, y bajo diferentes configuraciones.

CAPÍTULO 3

ANÁLISIS Y DISEÑO

RESUMEN

3. Metodología para el desarrollo e implementación

Para el desarrollo del presente proyecto de la tesis se utilizará la siguiente metodología COMET (Concurrent Object Modeling and architectural design mEThod), propuesta en el Plan de Tesis, ésta metodología es fácil de usar, y posee diagramas sencillos de elaborar, en comparación a otras metodologías que se usan para tiempo real, en la siguiente tabla se verá algunas comparaciones entre metodologías, tabla 3.1

Tabla 3.1 Comparación de las metodologías en Aspectos Generales

	METODOLOGÍA		
ASPECTOS GENERALES	COMET	OCTOPUS	ROPES
FACILIDAD DE USO	Alta	Media	Media
CLARIDAD DE LOS DIAGRAMAS	Alta	Media	Media
HERRAMIENTAS AUTOMATIZADAS DE SOPORTE	Si	Si	Si
DOCUMENTACIÓN DE SOPORTE	Alta	Alta	Media
DISEÑO DE LA ARQUITECTURA	Si	Si	Si
EVALUACIÓN DE LA ARQUITECTURA	No	No	No

3.1 Metodología COMET (Concurrent Object Modeling and architectural design mEThod):

Es una metodología que emplea notación UML, y está basada en un ciclo de desarrollo iterativo, con las siguientes fases: modelado de requisitos, análisis, diseño, construcción e integración incremental del software y validación del sistema, tal como se muestra en la figura 3.2.

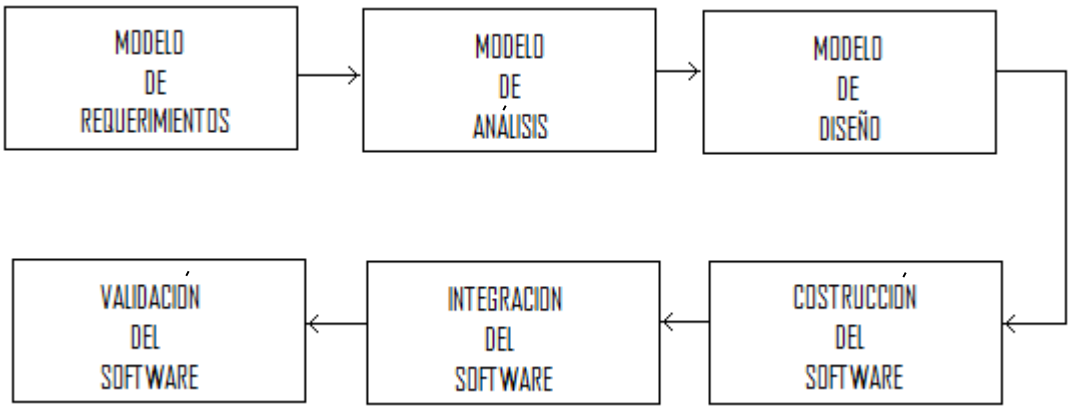


Figura 3.2 Fase de COMET

Define rasgos fundamentales que poseen las metodologías de desarrollo de software. Esta metodología permite:

- 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.

3.1.1.- Fase de Requerimientos

Los requisitos funcionales del sistema de Control de Tiempo Real se especifican mediante actores y casos de uso, los mismos que son fundamentales para determinar de esta forma sus requerimientos de información.

Este modelo permite proyectar las estrategias, procesos y flujos de datos del Sistema de Control de Tiempo Real al igual que las interrelaciones entre procesos y datos, con el fin de desarrollar un plan de sistema de información capaz de guiar

el desarrollo de un sistema que permita dar soporte al área en estudio en el cumplimiento de sus objetivos.

3.1.2.- Fase de Análisis

En esta etapa se refinan los requerimientos del Sistema de Control de Tiempo Real, para describir los objetos que intervienen y sus interacciones, a través de diagramas de clase (modelo estructural) y mediante colaboraciones y/o diagramas de estado (comportamiento dinámico), se determina el alcance del Sistema.

3.1.3.- Fase de diseño

Se desarrolla la arquitectura del software se selecciona la aproximación básica para resolver el problema. Durante el diseño del sistema, se decide la estructura y el estilo global. La arquitectura del sistema es la organización global del mismo en componentes llamados subsistemas. La arquitectura proporciona el contexto en el cual se toman decisiones más detalladas en una fase posterior del diseño.

3.1.4.- Fase de construcción

Se diseña el comportamiento estático y dinámico del Sistema de Control de Tiempo Real. Se desarrolla completamente el software y los documentos necesarios que componen el sistema. El resultado de esta fase es un producto listo para que los usuarios puedan operar y sea consiste del software integrado en

las plataformas adecuadas, los materiales para soporte del usuario y una descripción de la versión actual.

3.1.5.- Fase de integración

Se integran los módulos de software creados, el sistema de Control de Tiempo Real se encuentra listo para ser Aprobado.

3.1.6.- Validación temporal

Es el proceso de revisión que el sistema de software producido cumple con las especificaciones y que cumple su cometido. Es normalmente una parte del proceso de pruebas de software de un proyecto, que también utiliza técnicas tales como evaluaciones, inspecciones, y tutoriales. La validación es el proceso de comprobar lo que se ha especificado es lo que el usuario realmente quería.

3.2. Fase 1: Modelo de requerimientos

3.2.1 Introducción

Los métodos tradicionales en cualquier ingeniería requieren como primer paso la obtención de los requisitos en forma de especificaciones por parte del cliente. Este problema habitualmente tiene complicaciones debidas al paso entre el lenguaje natural y los lenguajes más formales en ingeniería. Por lo tanto la obtención de los requisitos es un paso complejo y que no tiene una solución sencilla. Se suelen

utilizar los métodos de pregunta-respuesta o los de cuestionarios plantilla para perfilar la versión inicial, pero se requieren sucesivas iteraciones (incluso con otras fases de desarrollo) antes de obtener unas especificaciones adecuadas.

3.2.2 Identificación de roles (actores) y tareas

En esta fase se analizan las necesidades de los usuarios finales del software para determinar qué objetivos debe cubrir. De esta fase surge una memoria llamada SRD (documento de especificación de requisitos), que contiene la especificación completa de lo que debe hacer el sistema sin entrar en detalles internos, tal como se especifica en la tabla 3.1

Es importante señalar que en esta etapa se debe **consensuar** todo lo que se requiere del sistema y será aquello lo que seguirá en las siguientes etapas, no pudiéndose requerir nuevos resultados a mitad del proceso de elaboración del software.

Cuadro 3.1 Descripción de rol. Elaboración propia

Rol	**Descripción**
Administrador	Usuario que tiene la capacidad de gestionar y administrar los procesos que se van dando en el Kernel del Sistema Operativo.
Usuario	Usuario que posee privilegios para ver los procesos del sistema.

3.2.3 Especificación de casos de uso

Se han identificado 2 roles para manejar el sistema como se observa en el cuadro 3.1. Después de definir los roles, se identifican las tareas del sistema para asegurarse que el conjunto de tareas se han representadas adecuadamente.

Cuadro 3.2 Definición de tareas del actor Administrador.

Identificación de tareas por roles		
Rol	**Módulos**	**Tarea**
Administrador	**Inicio**	1. Buscar información de los procesos del Sistema Operativo.
	Ingresar	2. Se ingresa al sistema ingresando datos del usuario y contraseña.
	Registrarse	
	Modificar Perfil	
	Secciones	3. New: cuando el proceso está siendo creado. 4. Running: cuando el proceso se está ejecutando. 5. Waiting: cuando el proceso está esperando que se cumpla algún otro evento. 6. Ready: cuando el proceso esta pronto

		para ejecutar, esperando por la CPU. 7. Terminated: cuando el proceso está terminado.

Cuadro 3.3 Definición de tareas del actor Usuario.

Identificación de tareas por roles		
Rol	**Módulos**	**Tarea**
Usuario	**Ingresar**	8. Ingresar al sistema utilizando un usuario y contraseña.
	Inicio	9. Buscar información de los procesos.
	Registrarse	10. Ingresar datos del perfil de usuario
	Modificar Perfil	
	Secciones	11. Buscar procesos empotrados en el Sistema Operativo. 12. Visualizar los procesos

3.2.4. Especificación de casos de uso

Cuadro 3.4 Caso de Uso 1. Buscar información de los procesos.

Elaboración propia.

Buscar información de los procesos	**No. 1**
Roles:	Administrador, Usuario

Tareas:	- 6
Precondición:	
Descripción:	1. Los usuarios del sistema: Administrador, Usuario ingresan al sistema. 2. El sistema sugiere la opción start simulink. 3. El sistema muestra una ventana de los procesos que se están ejecutando.
Postcondición:	Ninguna

Cuadro 3.5 Caso de Uso 2. Ingresar al sistema utilizando un usuario y contraseña. Elaboración propia.

Ingresar al sistema sin usar un usuario y contraseña	**No. 2**
Roles:	Administrador, Usuario
Tareas:	2 – 7

Precondición:	
Descripción:	1. El Administrador o Usuario ingresa al sistema.
Postcondición:	Ninguna

Cuadro 3.6 Caso de Uso 3. Buscar los procesos que se están ejecutando. Elaboración propia.

Buscar los procesos que se están ejecutando	**No. 3**
Roles:	Administrador, Usuario
Tareas:	4 – 8
Precondición:	

Descripción:	1. Los usuarios del sistema: Administrador, Usuario ingresan al sistema. 2. Los usuarios ingresan. 3. El sistema sugiere la opción start simulink. 4. El sistema muestra una ventana que se está ejecutando los procesos.
Postcondición:	Ninguna

Cuadro 3.7 Caso del Uso 4. Pre visualizar los procesos. Elaboración propia.

Pre visualizar los procesos del Sistema Operativo	**No. 4**
Roles:	Administrador, Usuario
Tareas:	9
Precondición:	El Administrador debe iniciar sesión en el sistema (Caso uso referente a la tarea 2).
Descripción:	1. El Administrador y Usuario ingresa al sistema. 2. El Administrador y Usuario escoge la opción de ver los

	procesos.
Postcondición:	Ninguna

3.2.5 Especificación de UIDs

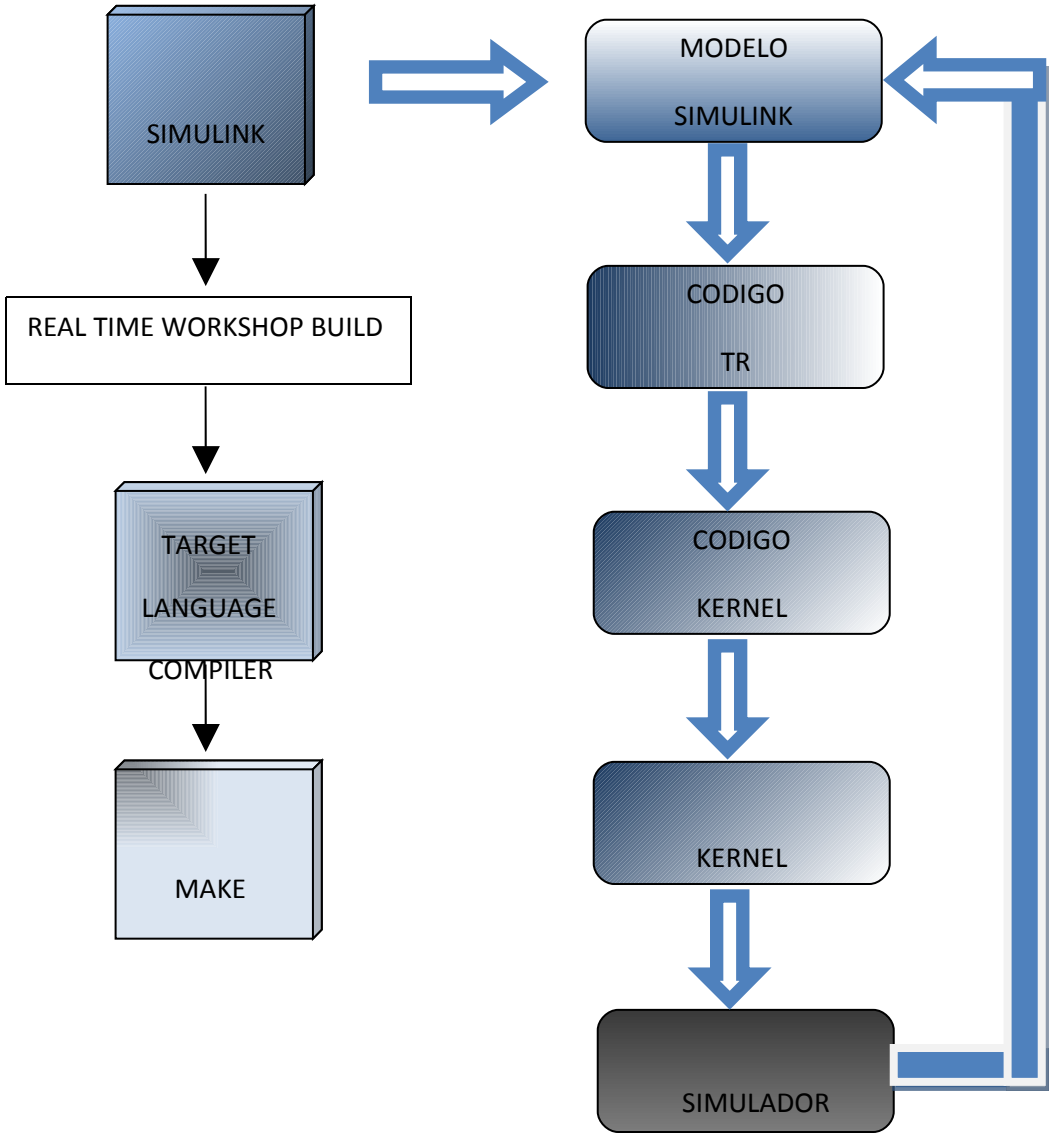
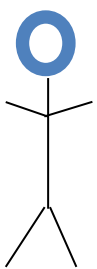


Figura 3.2 UID 1 correspondiente al Cuadro 3.4 Caso de Uso 1. Buscar información de los procesos. Elaboración propia.



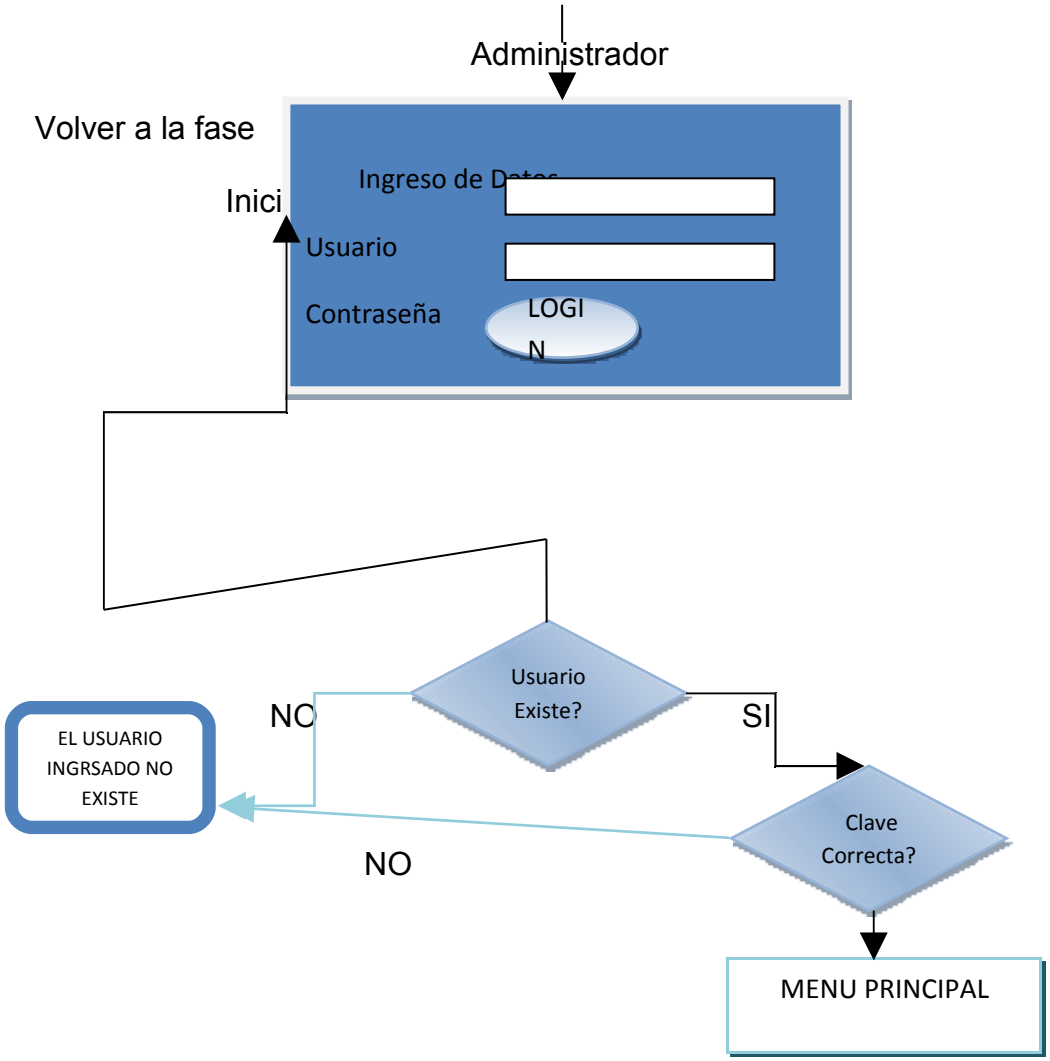
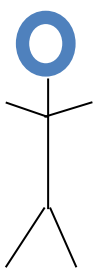


Figura 3.3 UID 2 correspondiente al Cuadro 3.5 Caso de Uso 2. Ingresar al sistema utilizando un usuario y contraseña. Elaboración propia



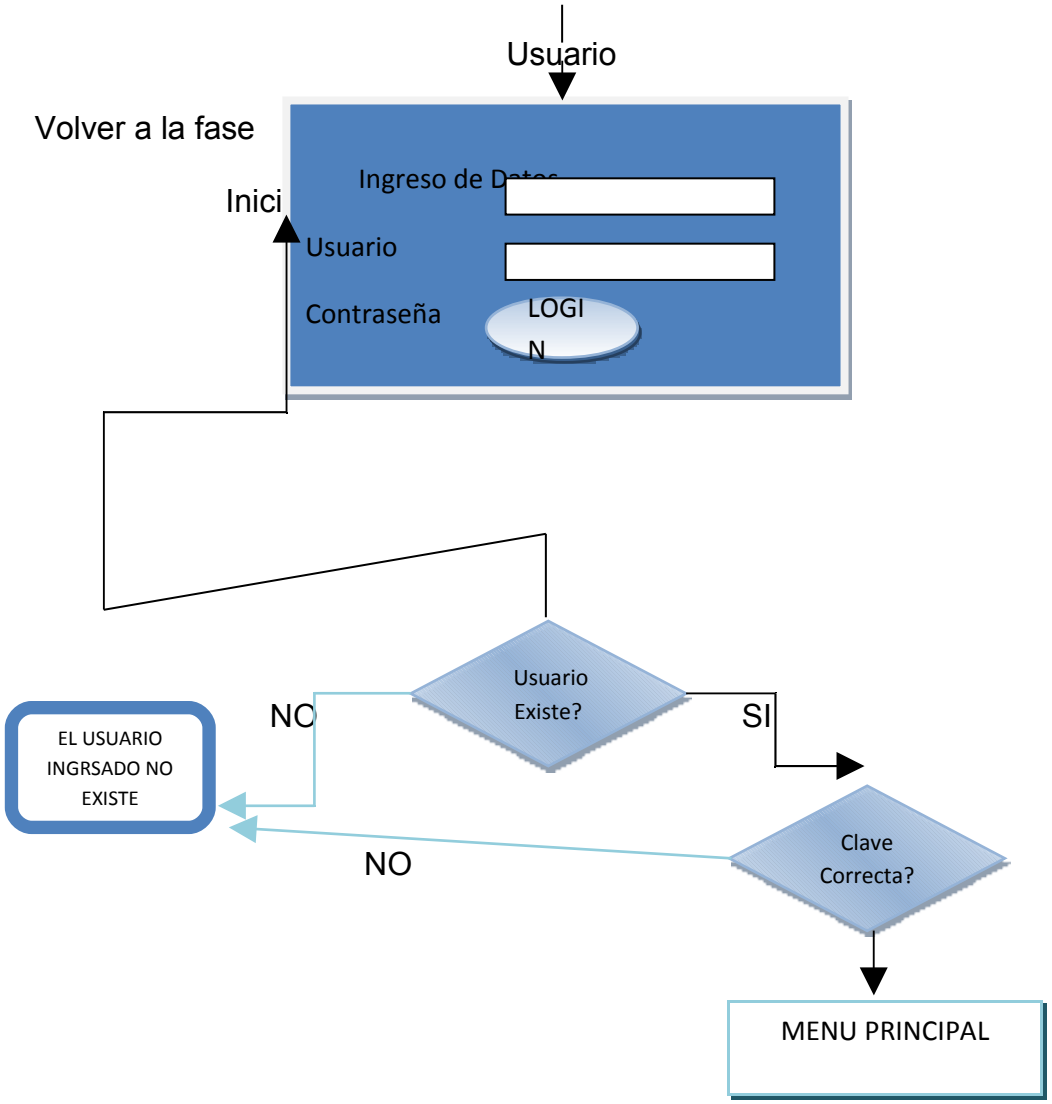
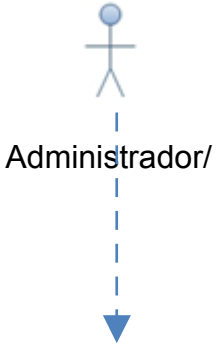


Figura 3.4 UID 2 correspondiente al Cuadro 3.5 Caso de Uso 2. Ingresar al sistema utilizando un usuario y contraseña. Elaboración propia.



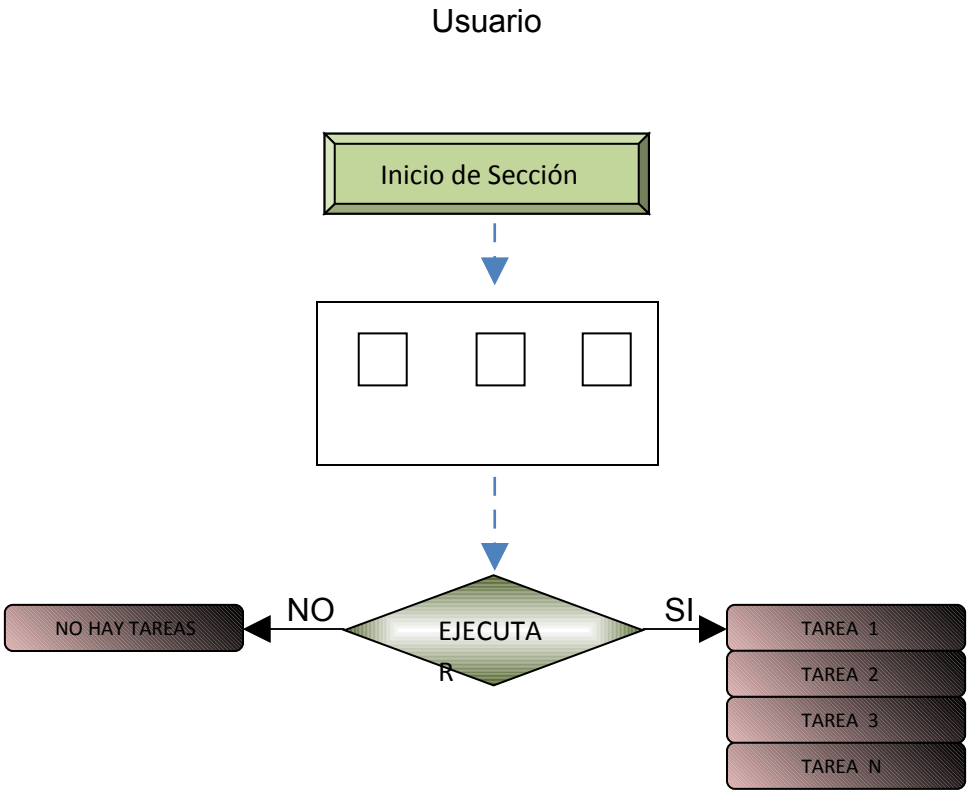
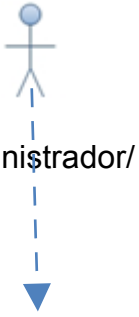


Figura 3.5 UID 3 correspondiente al Menú de Inicio



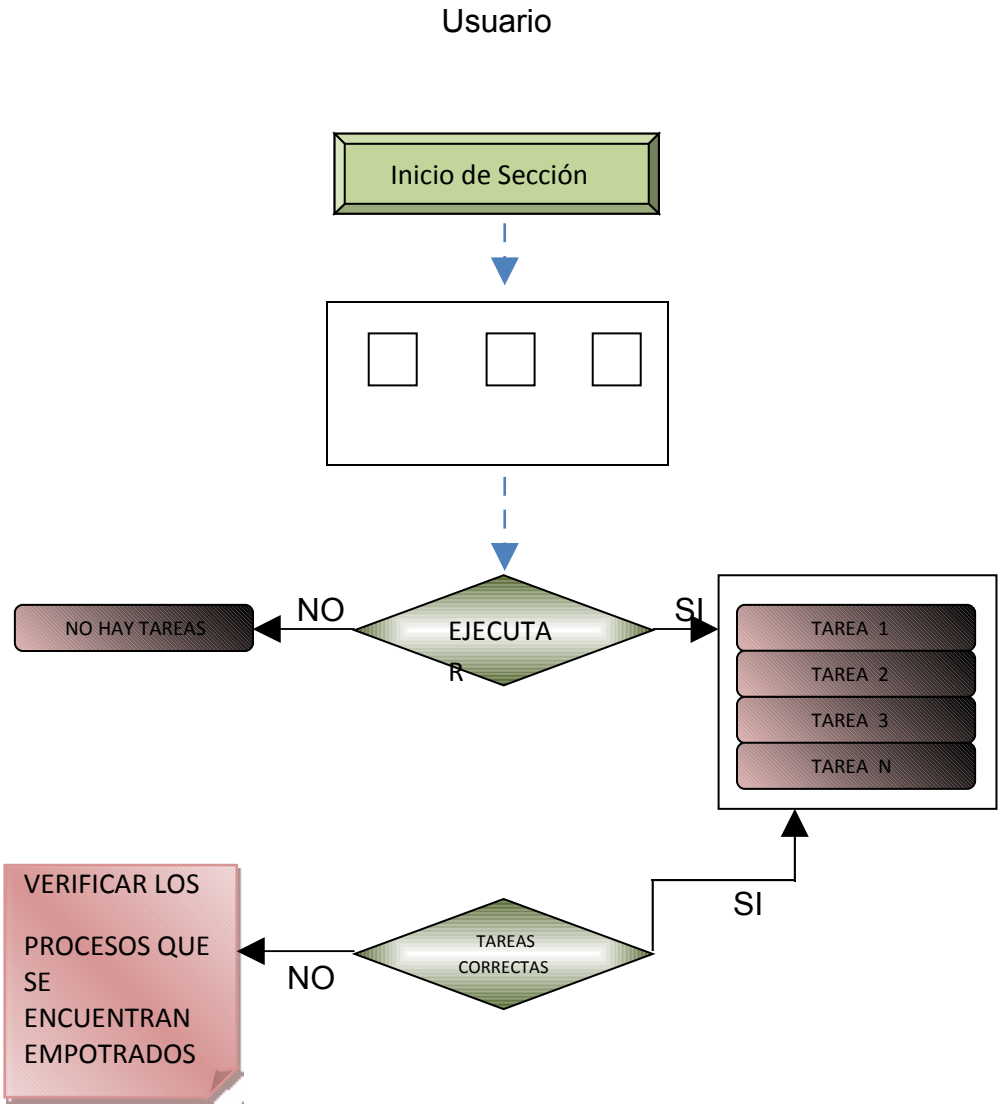
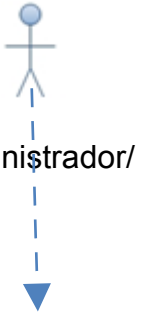


Figura 3.6 UID 4 correspondiente al Caso de Uso 4. Elaboración propia.



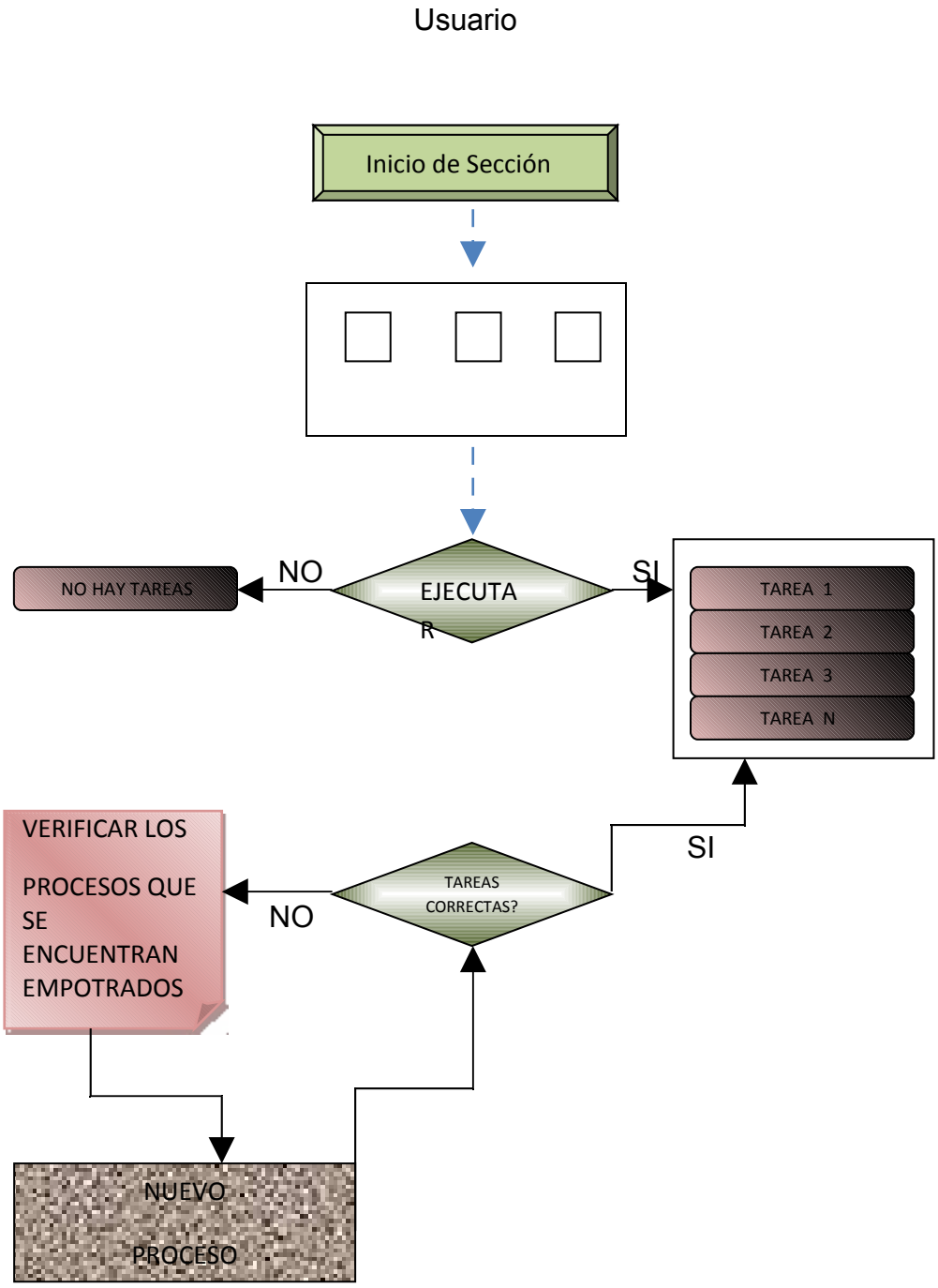


Figura 3.7 UID 4 correspondiente al Caso de Uso 4. Elaboración propia.

3.3 **Fase 2: Diseño Conceptual**

Cuando se trabaja bajo el análisis conceptual de una situación, se refiere a la abstracción de hechos reales de los cuales se emite un concepto o es posible

hacer una idea de ello. Para poder realizar la abstracción de un tema en un área específica, a nivel informático, es necesario tener los requerimientos formulados por los usuarios con respecto a este. Estos requerimientos contienen el conjunto de hechos y reglas que dan pauta a la creación del *esquema conceptual* donde por medio de este se podrá realizar una descripción de alto nivel de la futura base de datos. Para manipular este esquema se utiliza un *modelo conceptual* que proporciona un lenguaje que permite utilizar un conjunto de símbolos (estándares) para la creación de este.

Para modelar Conceptualmente es posible utilizar varios Modelos de Datos Un modelo práctico para ilustrar el diseño conceptual es el modelo entidad relación.

El software de tiempo real está muy acoplado con el mundo externo, esto es, el software de tiempo real debe responder al ámbito del problema en un tiempo dictado por el ámbito del problema. Debido a que el software de tiempo real debe operar bajo restricciones de rendimiento muy rigurosas, el diseño del software esta conducido frecuentemente, tanto por la arquitectura del hardware como por la del software, por las características del sistema operativo.

El sistema que se muestra en el grafico 3.1, se basa en el Sistema en Tiempo Real para el Kernel del Sistema Operativo.

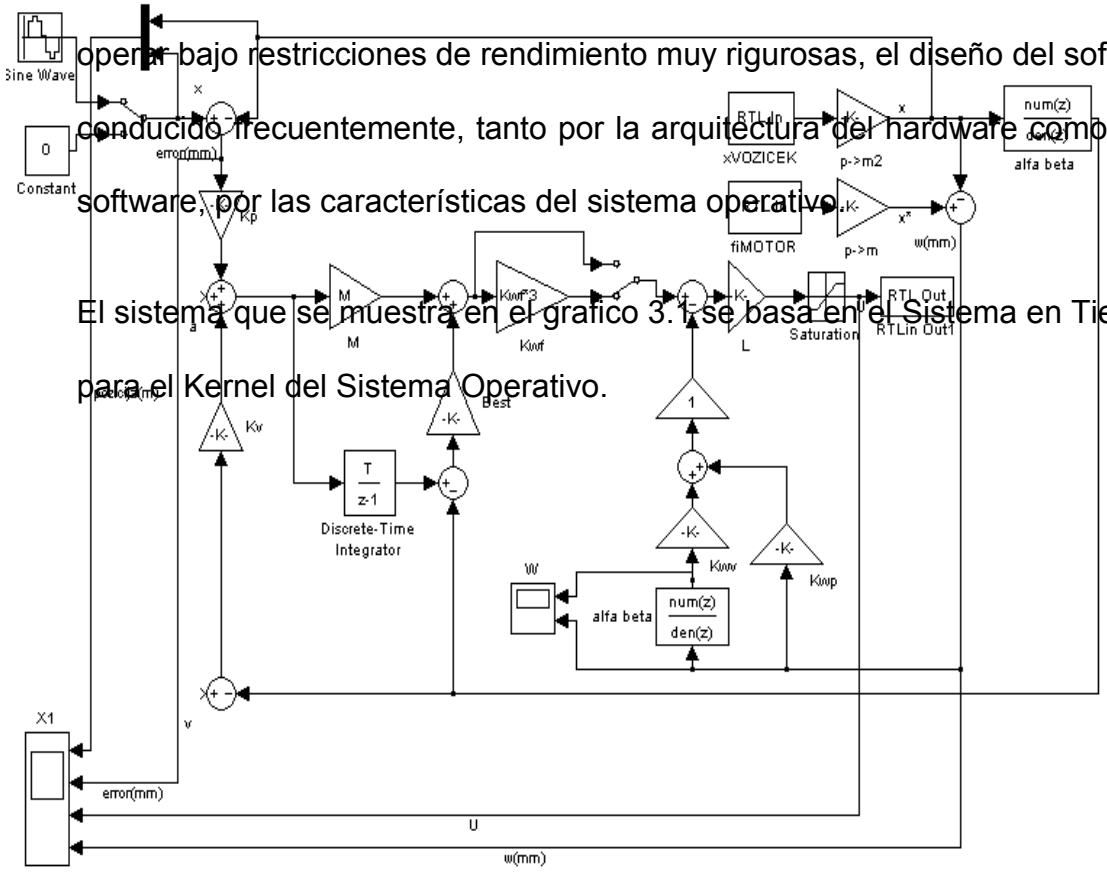


Figura 3.8 Sistema de Tiempo Real del Kernel del Sistema Operativo

3.3.1 Kernel

En la Figura 3.8 se presenta la arquitectura del Kernel en tiempo real. En el nivel más bajo se encuentra los distintos dispositivos que controla el Kernel. 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.

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

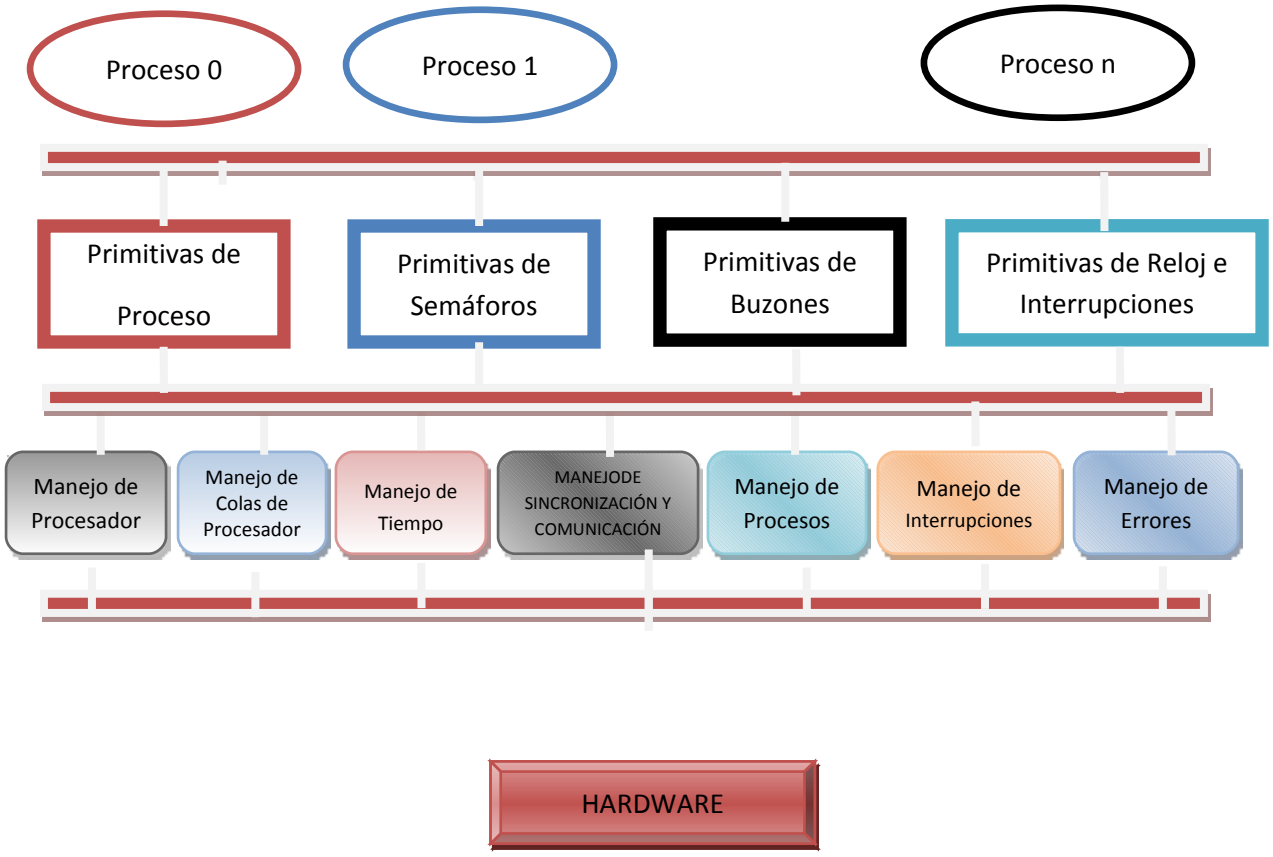


Figura 3.9 Arquitectura del Kernel

3.3.2 Procesos

Inicialmente el Kernel está configurado para trabajar con un máximo de 32 procesos. Sin embargo, el usuario, con base en sus necesidades, 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.

3.3.3Tareas.

Monotarea: Solamente puede ejecutar un proceso (aparte de los procesos del propio S.O.) en un momento dado. Una vez que empieza a ejecutar un proceso, continuará haciéndolo hasta su finalización y/o interrupción.

Multitarea: Es capaz de ejecutar varios procesos al mismo tiempo. Este tipo de S.O. normalmente asigna los recursos disponibles (CPU, memoria, periféricos) de forma alternada a los procesos que los solicitan, de manera que el usuario percibe que todos funcionan a la vez, de forma concurrente.

3.3.4 Comunicación entre Tareas

Las diferentes tareas de un sistema no pueden utilizar los mismos datos o componentes físicos al mismo tiempo. Hay dos métodos para tratar este problema.

Uno de los métodos utiliza semáforos. En general, el semáforo binario puede estar cerrado o abierto. Cuando está cerrado hay una cola de tareas esperando la apertura del semáforo.

Los problemas con los diseños de semáforos son bien conocidos: inversión de prioridades y puntos muertos (deadlocks).

En la inversión de prioridades, una tarea de mucha prioridad espera porque otra tarea de baja prioridad tiene un semáforo. Si una tarea de prioridad intermedia impide la ejecución de la tarea de menor prioridad, la de más alta prioridad nunca llega a ejecutarse. Una solución típica sería otorgar a la tarea que tiene el semáforo la prioridad de la tarea más prioritaria de las que están esperando dicho semáforo. Esto se denomina algoritmo de herencia básica de prioridad.

En un punto muerto, dos tareas (T1,T2) pretenden adquirir dos semáforos (semA,semB) en orden inverso. En este caso si T1 adquiere semA y T2 adquiere semB cuando intenten adquirir el segundo semáforo no podrán hacerlo ya que lo tiene la otra tarea. De esta forma entran en un punto muerto del que ninguna de las dos tareas puede salir sin intervención externa. Esto se resuelve normalmente

mediante un diseño por ej. Obligando a adquirir los semáforos en un orden concreto.

La otra solución es que las tareas se manden mensajes entre ellas. Esto tiene los mismos problemas: La inversión de prioridades tiene lugar cuando una tarea está tratando un mensaje de baja prioridad, e ignora un mensaje de más alta prioridad en su correo. Los puntos muertos ocurren cuando dos tareas realizan envíos bloqueantes (se quedan en la función de envío esperando a que el receptor reciba el mensaje). Si T1 manda un mensaje de forma bloqueante a T2 y T2 manda un mensaje de igual forma a T1 ninguna de las dos tareas saldrá de la función de envío quedando ambas bloqueadas ya que no podrán llegar a la función de recepción. Puede resolverse reordenando envíos y recepciones o empleando envíos no bloqueantes o temporizados.

Aunque su comportamiento en tiempo real es algo más difícil de analizar que los sistemas de semáforos, los sistemas basados en mensajes normalmente son más sencillos de desarrollar que los sistemas de semáforo.

3.3.5 BCP de los Procesos en el Kernel

Cada proceso tiene asociada una estructura llamada Bloque de Control de Proceso (BCP) la cual contiene la información necesaria para el control del proceso. La estructura del BCP en el Kernel está representada en la Figura 3.10,

en el cual se puede apreciar que la información está ordenada de la siguiente manera:

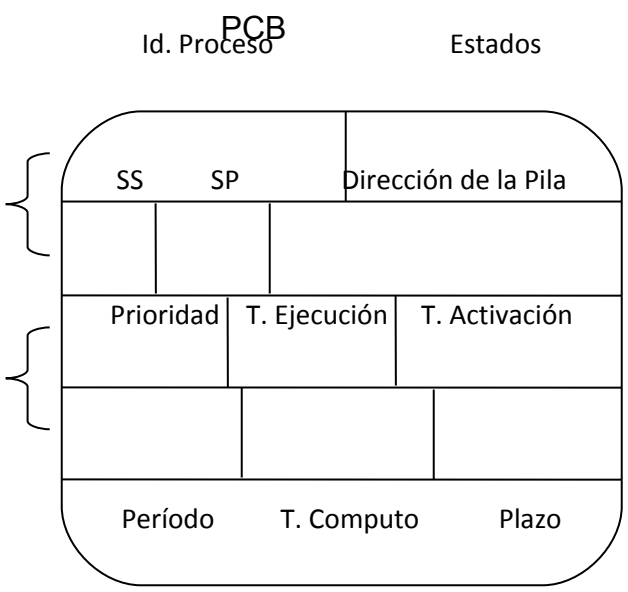


Figura 3.10 Bloque de Control de Proceso

1) Información del proceso.- Aquí se encuentra almacenado el identificador del proceso, el *estado* en que se encuentra durante la ejecución del Kernel (listo, suspendido, etc.) y la dirección de la pila que le corresponde al proceso. La pila almacena el contenido de sus registros (cs, ip, bp, di, si, ds, es, dx, cx, bx, ax y flags) así como las direcciones de las instrucciones que ejecuta el proceso. Los registros incluidos en el stack son indispensables para que el Kernel pueda llevar a cabo el *cambio de contexto*.

2) Información para el planificador.- Aquí se encuentran los datos necesarios para que el proceso pueda ser planificado. Contiene información como la prioridad, el tiempo de cómputo, el plazo, el tiempo de activación y el tiempo de ejecución. Estos datos son necesarios para que las políticas de planificación puedan calcular cuando le corresponde ejecutarse a cada proceso.

3) Información de memoria utilizada en buzones.- Sólo contiene un apuntador a una dirección de memoria reservada para almacenar un mensaje del buzón en caso de que el proceso sea introducido a la cola de procesos bloqueados por buzón (cpbb).

3.3.6 Estados de los Procesos en el Kernel

Cada proceso puede encontrarse en cualquiera de los siguientes estados [7], como se describe en la Figura 3.11:

Ejecución.- En este estado, el proceso tiene el control del procesador.

Listo.- En este estado, el proceso se encuentra listo para ejecutarse y espera a que se le asigne el procesador.

Bloqueado.- En este estado, el proceso se encuentra bloqueado en algún recurso, como puede ser un semáforo, o un buzón.

Retrasado.- En este estado el proceso se encuentra bloqueado, esperando por un tiempo determinado antes de regresar al estado de listo.

Eliminado.- En este estado se encuentran los procesos que no han sido activados o que ya han terminado su ejecución.

Los procesos cambian de estado por alguna de las siguientes razones:

1. El proceso es creado.

2. Se le asigna el procesador.

3. Se le quita el procesador debido a: un bloqueo, el alistamiento de un proceso con mayor prioridad o por el fin de su tiempo de cómputo.

4. El proceso se retrasa.

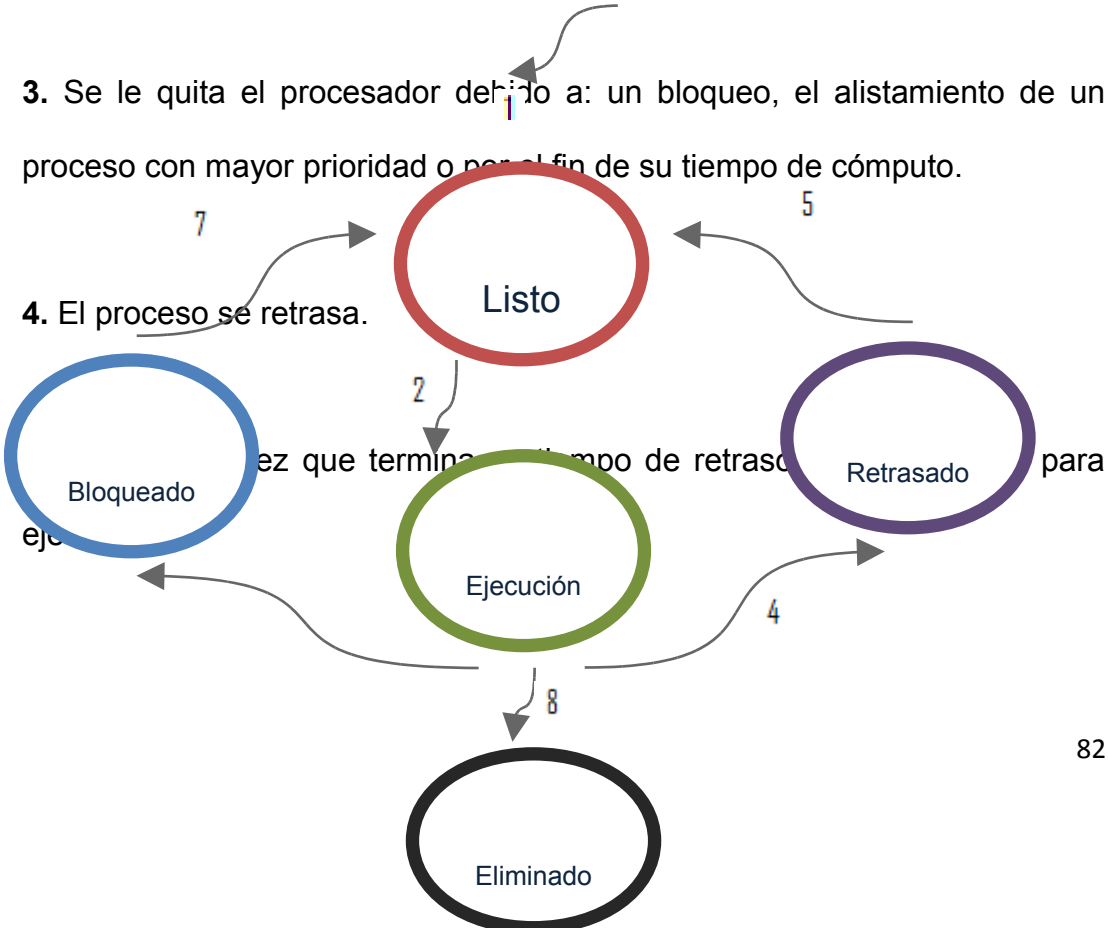


Figura 3.11 Estados de los Procesos

6. El proceso se bloquea por un recurso o interrupción.

7. Obtuvo el recurso que esperaba y queda listo para ejecutarse.

8. El proceso se elimina a sí mismo.

3.3.7 Transiciones entre Estados

Los procesos cambian de estado por alguna de las siguientes razones:

1. El proceso es creado y activado.

2. Se le asigna el procesador.

3. Se le quita el procesador debido a: un bloqueo, el alistamiento de un proceso con mayor prioridad o por el fin de su tiempo de cómputo.

4. El proceso se retrasa.

5. Ocurre una vez que termina su tiempo de retraso, quedando listo para ejecutarse.

6. El proceso se bloquea por un recurso o interrupción.

7. Obtuvo el recurso que esperaba y queda listo para ejecutarse.

8. El proceso se elimina a sí mismo.

3.3.8 Manejadores

3.3.8.1 Manejo de Registros

El manejador de registros contiene los procedimientos necesarios para controlar los registros y los datos en memoria interna. Las operaciones aquí realizadas sólo se ejecutan una vez (al iniciar el Kernel) y son fundamentales ya que preparan al sistema para que éste pueda trabajar en la máquina permitiéndole realizar correctamente los cambios de contexto entre los procesos. La Figura 4.1 nos muestra los procedimientos que contiene el manejador de registros.

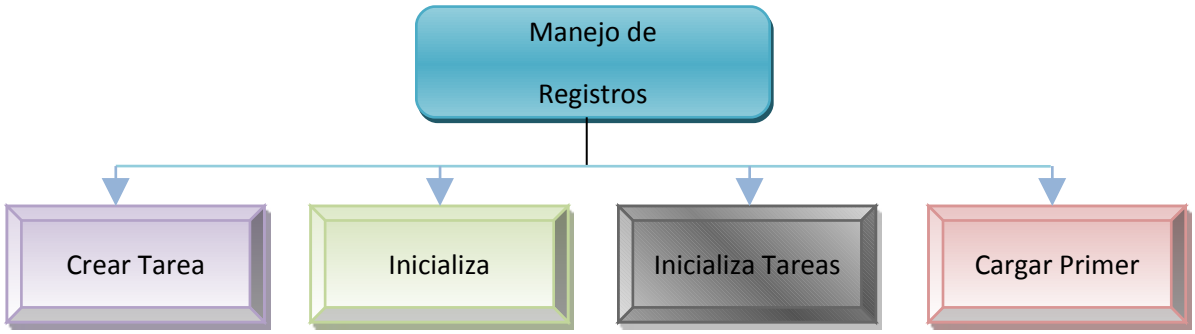


Figura 3.13 Procedimientos de Manejador de Registros

CreaTarea :

NOMBRE: void CreaTarea(void (*DirTarea)(),unsigned id,unsigned prio,unsigned long TC, unsigned long P).

FUNCIÓN: Inicia los registros de la tarea indicada, obtiene los valores del SS y SP, el CS y el IP del proceso, llena la pila con los registros de la máquina en el

siguiente orden: FLAGS, CS, IP, AX, BX, CX, DX, ES, DS, SI, DI, BP y además asigna la prioridad, el tiempo de computo y el período al proceso para que este pueda ser planificado.

ENTRADAS: La dirección de la tarea a crear, su identificador, prioridad, período y tiempo de cómputo.

SALIDAS: Una tarea con pila y registros listos para ser utilizados.

3.3.8.2 Manejo del Procesador

El manejador de procesos (tareas) es uno de los servicios básicos con los que cuenta el Kernel y ofrece varias funciones de soporte tales como la creación y eliminación de procesos, la planificación de tareas (EDF, Rate Monotonic, FIFO Round Robin) y el cambio de contexto.

Un *proceso* es un conjunto de código secuencial identificado con un nombre y una prioridad, el cual se ejecuta concurrentemente con otros procesos.

Cada proceso tiene asociada una estructura llamada PCB (Process Control Block) la cual contiene la información necesaria para el control del proceso.

3.3.8.3 Manejo de Colas de Procesos

Debido a que el Kernel se ejecuta sobre un solo procesador (aunque tiene también la capacidad de comunicarse con otros procesadores sobre los que se ejecute el

Kernel), solo un proceso puede estar ejecutándose en el procesador, mientras que varios procesos pueden encontrarse listos para ejecución y otros más pueden estar bloqueados por recursos.

Dependiendo del estado en que se encuentren las tareas, cada una de estas puede encontrarse en alguna de las siguientes colas:

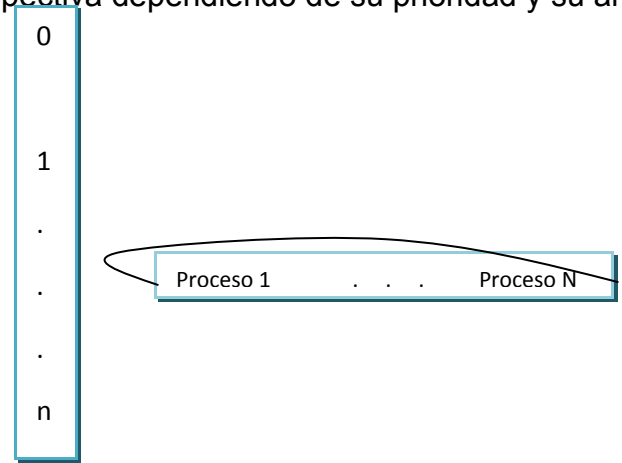
1. Cola de procesos listos.

2. Cola de procesos bloqueados por enviar o recibir un mensaje.

3. Cola de procesos bloqueados por espera en un semáforo.

4. Cola de procesos retrasados.

Las colas de procesos fueron diseñadas con el fin de manejar eficientemente la ejecución de las tareas de acuerdo a su estado. Las tres primeras colas tienen una estructura similar con múltiples niveles de prioridades, como se muestra en la Figura 3.13. La primera tarea en la cola es aquella que cuenta con la mayor prioridad en el sistema. En nuestro Kernel, asumimos que los procesos se insertan al final de la cola respectiva dependiendo de su prioridad y su arribo a la cola.



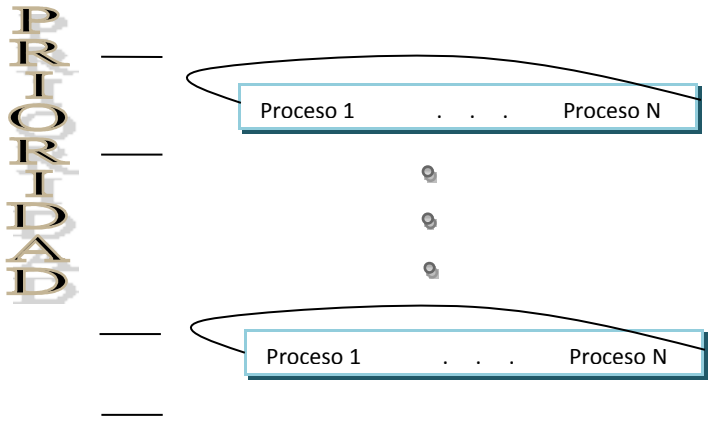


Figura 3.14 Cola de Procesos

3.3.8.4. Manejo de Planificadores

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 la mayor precisión posible los tiempos de arribo, los tiempos de ejecución y los tiempos de 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.

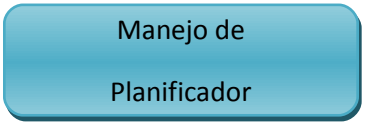
El Kernel desarrollado utiliza tres tipos de planificadores:

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

El planificador FIFO Round Robin es 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 FIFO 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. Ver la figura 3.14



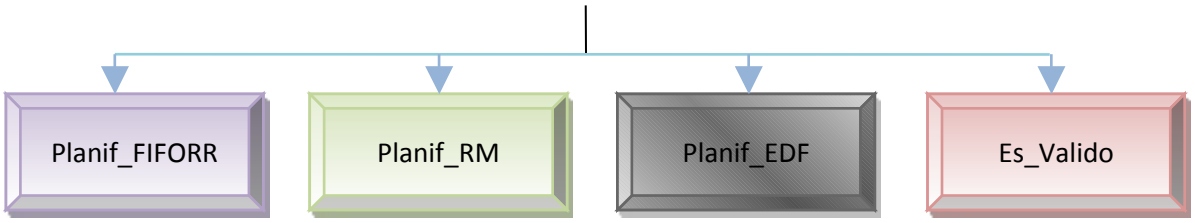


Figura 3.14 Procedimiento de manejador de Planificador

3.3.8.5 Manejo del Reloj

Mediante el manejador del tiempo, el Kernel puede manejar el tiempo en que una tarea se ejecuta en forma continua. A este tiempo se le conoce como *Quantum* y depende de la resolución del timer del sistema. Se modificó el timer del sistema para lograr una resolución aproximada de 1 milisegundo. Por esta razón, el Quantum solo puede darse en unidades de 1 milisegundo.

Dentro del manejo del tiempo también se permite que una tarea pueda dormirse durante un cierto tiempo mediante la primitiva *retrasa*. Con esta primitiva un proceso es incluido a una cola de procesos dormidos, y despertar ´a hasta que se consuma el tiempo que solicita. El parámetro que recibe la primitiva retrasa, permite al proceso que la ejecuta dormirse por un número dado de unidades de tiempo.

3.3.8.6 Manejo de Interrupciones

Actualmente el Kernel procesa solo las solicitudes de interrupción del reloj y del teclado. La interrupción 8 es capturada con el fin de manejar el reloj del Kernel. Esta interrupción fue re-programada a 1 milisegundo con la finalidad de aumentar la granularidad del sistema. De esta forma, el *Quantum*, o el tiempo máximo asignado a una tarea para su ejecución, cuenta con mayor precisión. Con esta interrupción el Kernel controla el cambio de contexto y el tiempo de bloqueo de la primitiva retrasa. La interrupción del teclado permite al Kernel interrumpir su ejecución, y es útil en la detección de caracteres del teclado.

3.3.8.7 Manejo de Errores

Cuando un proceso llama a una primitiva, antes de ejecutarla, el Kernel lleva a cabo una serie de validaciones para detectar posibles parámetros erróneos o eventos que no se esperaban. El Kernel tiene implementado un manejador de errores de forma que si se llega a detectar uno, el sistema manda un mensaje de error indicando qué fue lo que ocurrió y además, termina la ejecución del mismo debido a que es un sistema de tiempo real y los procesos que se manejan son muy delicados por lo que no debe continuar si ocurre un error.

Este manejador tiene un solo procedimiento encargado de manipular los errores.

Códigos de Error

ERROR 1: Ocurre cuando el procedimiento “BuscaMayor” intenta conocer el proceso de mayor prioridad pero resulta que la cola de procesos listos está vacía.

ERROR 2: Este error se genera cuando se intenta insertar un proceso en cualquiera de las colas circulares tipo FIFO (la de listos, buzones o semáforos) y no hay espacio disponible en la cola.

ERROR 3: Este error sale cuando se intenta sacar un proceso de cualquier cola tipo FIFO circular y no hay elementos en ella (se encuentra vacía).

ERROR 4: Error que ocurre cuando se intenta eliminar un proceso de una cola tipo FIFO circular y no hay elementos en ella.

ERROR 5: Este error se genera cuando el proceso que se desea eliminar no está en la cola indicada.

ERROR 6: Se genera cuando la primitiva “Retrasa” tiene un tiempo de retraso inválido.

ERROR 7: Ocurre cuando la primitiva “Activa” contiene un identificador de proceso mayor a los permitidos en el Kernel.

ERROR 8: Este error se genera al intentar activar un proceso con mayor prioridad que la del proceso que lo creó.

ERROR 9: Ocurre cuando se intenta utilizar un identificador de semáforo mayor al número de semáforos permitidos.

ERROR 10: Este error sale cuando se intenta dar un valor inicial diferente de 0 o 1 al semáforo binario.

ERROR 11: Error que se genera cuando se intenta crear un semáforo ya creado.

ERROR 12: Error que sale cuando se intenta utilizar un semáforo que todavía no ha sido creado.

ERROR 13: Este error ocurre cuando la cuenta del semáforo es inválida.

ERROR 14: Ocurre cuando se intenta utilizar un número de buzón mayor a los permitidos.

ERROR 15: Sale este error cuando se quiere crear un buzón que ya ha sido creado.

ERROR 16: Error que se genera cuando se intenta utilizar un buzón que todavía no ha sido creado.

ERROR 17: Este error sale cuando en el archivo de configuración se ha escrito un planificador no válido.

3.3.4 Configuración e Inicialización del Kernel

3.3.4.1. Configuración del Kernel

El Kernel cuenta con un archivo de configuración llamado CONFIG.H, en el cual el usuario puede modificar el contenido de las variables para que el Kernel se adapte a sus necesidades.

Entre las cosas que se pueden modificar se encuentran:

1. El mecanismo de planificación que el Kernel utilizará (PLANIFICADOR).Puede ser Rate Monotonic, Earliest Deadline First y FIFO Round Robin.

2. El número de tareas hechas por el usuario (NTAREAS). La cantidad máxima de tareas o procesos depende de esta variable ya que automáticamente se suman la primera y última tarea al valor total de procesos en el Kernel (MAXPRO=NTAREAS+2). Cada vez que se cree una nueva tarea o proceso se debe aumentar aquí su valor.

3. El número máximo de prioridades manejadas por el Kernel (MAXPRIO).

4. El número total de semáforos (MAXSEM).

5. El total de buzones a utilizar en el Kernel (MAXBUZ).

6. El número total de mensajes a almacenar en el buzón (MAXMJE).

7. La longitud máxima para el mensaje (MAXLONGMJE).

8. El tiempo de cómputo permitido en el mecanismo de planificación FIFO Round Robin (QUANTUM).

Por defecto el Kernel se encuentra configurado de la siguiente manera:

- Utiliza el planificador Rate Monotonic.
- Permite crear 10 semáforos.
- Existen 5 tareas hechas por el usuario.
- Maneja 15 niveles de prioridad.
- Soporta 15 Buzones.
- Almacena 10 mensajes por buzón con una longitud de 15bytes cada uno.
- El Quantum para la planificación FIFO Round Robin es de 5.

3.3.4.2 Señales de los procesos

El Kernel contiene mecanismos de sincronización y comunicación entre tareas, los cuales se manejan mediante semáforos y buzones.

3.3.4.2.1 Semáforos

Mediante los *semáforos* es posible sincronizar y establecer regiones de exclusión mutua entre las tareas del sistema. Un semáforo es una variable entera que se crea en el espacio de memoria del Kernel, y cuyo valor sólo se puede manipular con tres primitivas que hacen sus operaciones de forma atómica1.

Las tres primitivas de los semáforos son:

IniciaSemáforo.- En esta primitiva se le da un valor inicial (positivo) a la variable del semáforo, con lo cual se crean una cola asociada al semáforo. En esta cola, se incluir án los procesos bloqueados por este recurso.

Señal.- Esta primitiva permite incrementar en una unidad, a la variable del semáforo. Si la variable del semáforo es negativa indicará que existe algún proceso bloqueado (en la cola de este semáforo). Por lo tanto si al ejecutar la primitiva señal, el contador es negativo, algún proceso bloqueado en la cola de este semáforo (el que se encuentre al principio de la cola), se pasará a la cola de procesos listos. En este caso, el Kernel provocará un cambio de contexto para

verificar si el proceso desbloqueado, es de mayor prioridad que el proceso en ejecución. Si al ejecutar la primitiva señal no existen procesos bloqueados (el contador es cero o positivo), se incrementa el contador del semáforo y el proceso continuará su ejecución.

Espera.- Esta primitiva permite decremento en una unidad a la variable del semáforo. Si después de decremento esta variable, su valor es negativo, el proceso que ejecuta esta primitiva es enviado a la cola del semáforo y sacado de ejecución.

En este caso, el Kernel invocará a la rutina de cambio de contexto para ejecutar al siguiente proceso de la cola de listos. Por el contrario, si después de ejecutar la primitiva espera, la variable del semáforo contiene un valor cero o positivo, el proceso continuará su ejecución.

En la Figura 3.15 se muestran 2 tareas haciendo acceso de un semáforo para implementar regiones críticas. El uso de estas regiones críticas permite a cada proceso accesar una variable compartida en forma exclusiva.

3.4.2. Buzones

Un *buzón* es un área de memoria compartida capaz de contener un número limitado de mensajes. Estos mensajes son almacenados en el buzón mediante una cola circular tipo FIFO. Con los buzones es posible comunicar a dos o más procesos entre sí. Cada buzón tiene asociada una cola de procesos. En esta cola

de procesos se introducirán aquellos procesos que se encuentren bloqueados por este buzón como muestra la figura 3.15.

En el manejo de los buzones se utilizan tres operaciones básicas:

CrearBuzón.- Esta primitiva permite crear la estructura de datos que define al buzón y su cola de procesos asociada.

EnviarMensaje.- Mediante esta primitiva es posible enviar un mensaje en formato ASCII al buzón indicado (el cual previamente tuvo que haber sido creado). Si al enviar un mensaje alguna tarea se encontraba bloqueada esperando mensajes, esta tarea ser desbloqueada y enviada a la cola de procesos listos.

Si un proceso envía un mensaje al buzón y el buzón se encuentra lleno, provocará que el proceso se incluya en la cola del buzón. Este proceso permanecerá en esta cola hasta que otro proceso reciba mensajes y libere suficiente espacio del buzón.

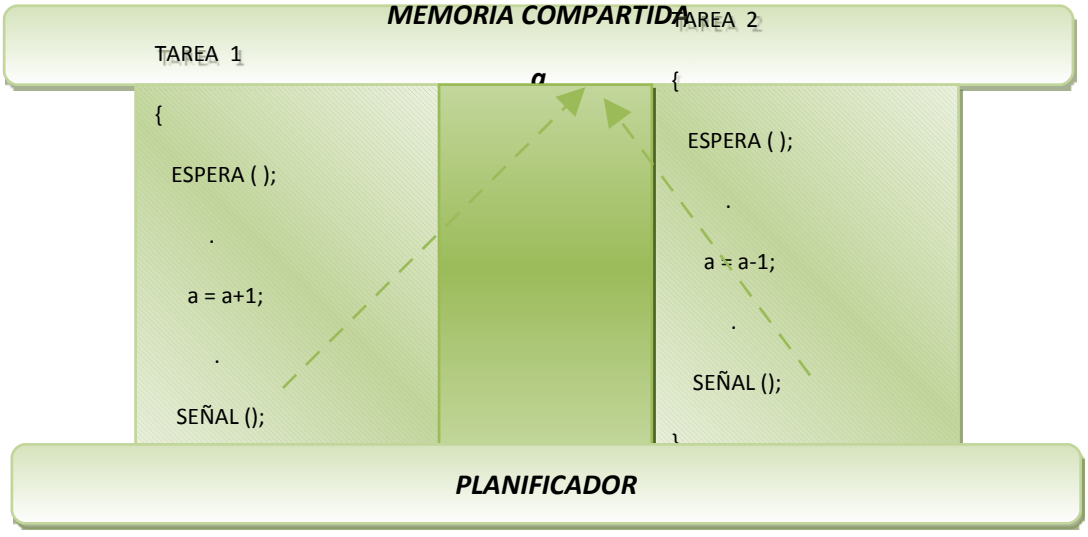


Figura 3.15 Utilización de los semáforos en el Kernel

RecibirMensaje.- Esta primitiva permite recibir mensajes del buzón indicado. Si el buzón se encuentra vacío, el proceso que invoca a esta primitiva será bloqueado en la cola respectiva.

Note que la cola del buzón contendrá solo a procesos bloqueados esperando recibir mensajes, o solo a procesos esperando enviar mensajes al buzón. Por su implementación, los dos tipos de procesos no pueden coexistir en una misma cola.

3.3.4.3 Inicialización del Kernel

Para poder ejecutar el Kernel es necesario compilar el archivo KERNEL.C, este archivo es el principal y es el que inicializa todo el sistema. Aquí se encuentran declaradas todas las librerías que contienen los manejadores y las primitivas del kernel y además, es aquí donde se deben indicar el nombre de los archivos que contienen las tareas o proceso a ejecutar. El proceso de inicialización del sistema es el siguiente:

1. Se cambia la resolución del "timer" para forzar que el temporizador de la máquina genere interrupciones periódicas cada milisegundo.

2. Se ejecuta la primitiva inicializa ( ) la cual se encarga de lo siguiente:
Inicializar cada una de las tareas poniendo su estado a "NUEVO".

Inicializar las colas de procesos utilizadas en el kernel (cola de procesos listos, retrasados, de semáforos y de procesos bloqueados por buzón), además de crear el semáforo 0, el cual está reservado para la región crítica.

3. Crea los procesos primero y último.

4. Se crean los procesos hechos por el usuario.

5. Activa los procesos primero y último.

6. Por ´ultimo manda a ejecutar la primera tarea.

Una vez ejecutada la primera tarea, quien se encargó de activar todos los procesos que se van a correr en el kernel, el control y la decisión de lo que se va a ejecutar en el procesador la toma el planificador y a partir de ahí, es este quien decide el comportamiento del kernel y la única manera de terminar con la ejecución del kernel es pulsando la tecla ESC o FIN.

3.3.5 Datos Técnicos

3.3.5.1 Tamaño

Debido a que el Kernel fue diseñado para su implementación en un sistema empotrado, el Kernel contiene sólo las funciones que son necesarias para el control de los sistemas de tiempo real en este tipo de dispositivos. Por lo tanto, el tamaño del Kernel es muy pequeño (20.5Kb). Esta característica hace posible que se pueda implementar en sistemas empotrados con altas restricciones de

memoria. En la Tabla 3.2 se muestra a detalle el tamaño de cada uno de los archivos que conforman el Kernel.

NOBRE DEL ARCHIVO	***TAMAÑO***
Kernel	1.05Kb
Config	231 bytes
Constant	1.55Kb
Librería	1.97Kb
Manreg	1.49Kb
Mancpu	491 bytes
Manplan	1.99Kb
Mancolas	4.24Kb
Manreloj	376 bytes
Manint	304 bytes
Manerror	1.85Kb
Pribuz	2.51Kb
Pripo	621 bytes
Prisem	1.67Kb
Pritim	255 bytes
TOTAL	***20.5Kb***

Tabla 3.2 Tamaño del Kernel

3.3.5.2. Tiempos de ejecución del Kernel

En cualquier aplicación de tiempo real es importante conocer el tiempo de ejecución de las primitivas diseñadas en el Kernel, debido a la predictibilidad con que debe contar el sistema. En la Tabla 3.3 se indica que el cambio de contexto tarda en promedio 32.44 microsegundos.

Debido a que la granularidad del Kernel es de 1 milisegundo (1,000 interrupciones por segundo), el tiempo que le resta al proceso para ejecutarse (después de la

ejecución de la rutina de cambio de contexto) sería de aproximadamente 967.55 microsegundos.

Los tiempos presentados en la Tabla 3.3 fueron obtenidos en una computadora Laptop Pentium Core Duo a 2GHz, con 4GB de RAM. Con la finalidad de obtener las mediciones más realistas posibles, el Kernel se ejecutó sólo bajo el ambiente de MS-DOS (fuera del ambiente de Windows).

PRIMITIVA	**Tiempo *minimo(us)***	**Tiempo máximo (us)**	**Tiempo Promedio(us)**
Inicialización del Sistema	11,43	13,23	12,34
Cambio de contexto	30,37	34,12	32,44
Activa	11,73	12,57	11,93
Elimina	11,73	14,24	12,08
Retrasa	11,73	13,23	12,23
Crea Semáforo	11,73	12,57	11,81
Señal	11,73	13,4	12,17
Espera	11,73	12,57	12,17

Crea Buzón	11,73	12,57	11,83
Envía Mensaje	12,57	14,24	12,8
Recibe Mensaje	13,4	15,92	14,23
Inserta a la cola	11,73	12,57	12,32
Saca de la cola	11,73	12,57	12,3
Busca el mayor	23,46	24,3	23,59
Earliest Deadline First	11,73	15,08	12,28
Rate Monotonic	11,73	13,4	12,25

Tabla 3.3 Tiempos de ejecución de las primitivas del Kernel

CAPÍTULO 4

DISEÑO DE NAVEGACIÓN

4.1.- Modelo General Nodos

El diseño del interfaz es uno de los elementos "clave" en la realización del programa.

Podemos definir el interfaz como:

"El conjunto de trabajos y pasos que seguirá el usuario, durante todo el tiempo que se relacione con el programa, detallando lo que verá y escuchará en cada momento, y las acciones que realizará, así como las respuestas que el sistema le dará"

Este interfaz requiere, en sí mismo, un esfuerzo mental independiente del contenido que nos muestre. El usuario además de entender el mensaje, ha de comprender la mecánica y la operativa que le oferta el interfaz. (Sintaxis, órdenes, códigos, abreviaciones, iconos...). Todo esto supone una carga de memoria sumada por el usuario. Un buen sistema, por tanto, ha de requerir menos esfuerzos mentales de manejo del interfaz y concentrar la atención en el contenido que quiere transmitir.

Con el fin de que esta carga de memoria sea minimizada, es muy importante establecer un sistema de ayudas adecuado. Estas ayudas han de ser diferentes de las que proporciona el personaje central (animación) que se centrará en el contenido. Las ayudas al interfaz, se basarán sobre todo en la operativa y la aclaración de funciones de los elementos visuales o acústicos.

De hecho el interfaz es en realidad un modelo mental permanente, es decir una representación cognitiva o conceptualización que el usuario hace del sistema. A fin de que este modelo se mantenga a lo largo del programa ha de tener una consistencia, es decir mantener su coherencia de principio a fin. Por ello se han de mantener las reglas, los criterios en la operatividad, la imagen parcial o total, etc... Una incoherencia de diseño puede aportar pérdidas de eficacia del propio contenido que se quiera dar.

Las características básicas que queremos conseguir con este interfaz, se podrían sintetizar:

- Facilidad de aprendizaje y uso.
- Representación permanente de un contexto de acción (fondo).
- El objeto de interés ha de ser de fácil identificación.
- Diseño ergonómico (barra de acciones o iconos, preferentemente a la derecha).
- Las interacciones se basarán en acciones físicas sobre elementos de código visual o auditivo (iconos, imágenes, mensajes...) antes que en selecciones de tipo menú con sintaxis y órdenes.
- Las operaciones serán rápidas, incrementales y reversibles, con efectos inmediatos.
- Tratamiento del error bien cuidado y adecuado al nivel de usuario y contenidos trabajados.

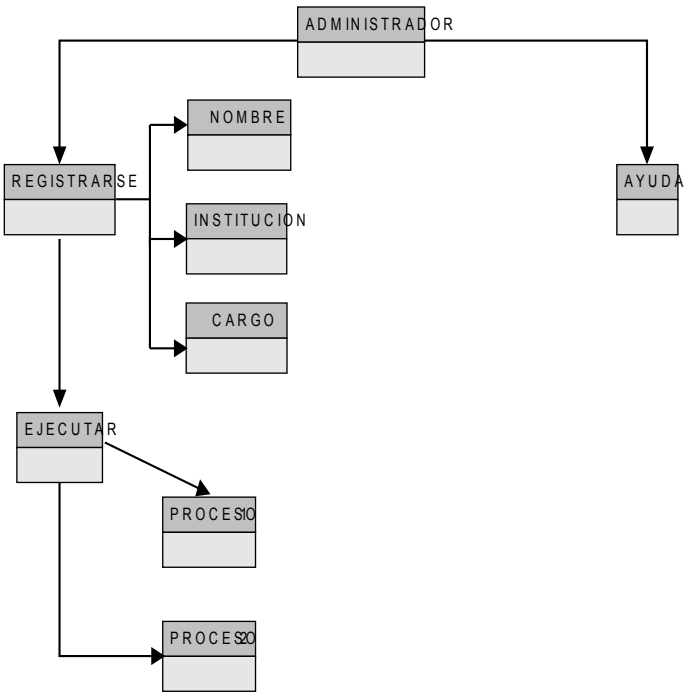


FIGURA 4.1 Modelo General de Nodos. Elaboración propia.

4.2 Esquema Navegacional

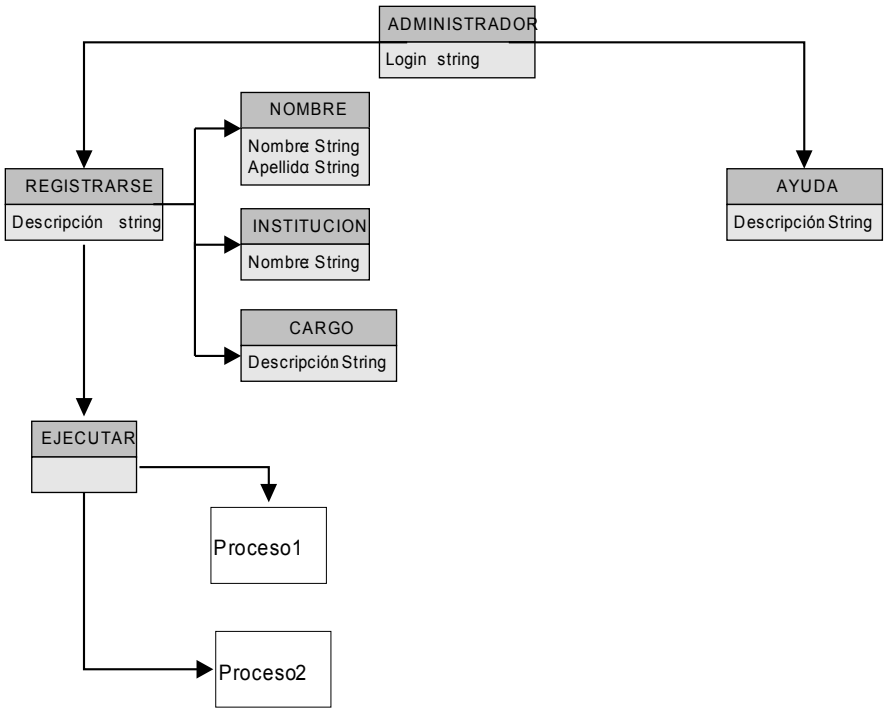


Figura 4.2 Esquema General del Contexto de Navegación. Elaboración propia.

4.3 Contextos Navegacionales

Para general el modelo navegacional se tomará en cuenta las tareas definidas en el ambiente de aplicación, las cuales se encuentran definidas en la especificación de los UIDs. A continuación se representa el contexto navegacional relacionadas con cada rol o actor identificado y posteriormente de forma general.

ROL ADMINISTRADOR

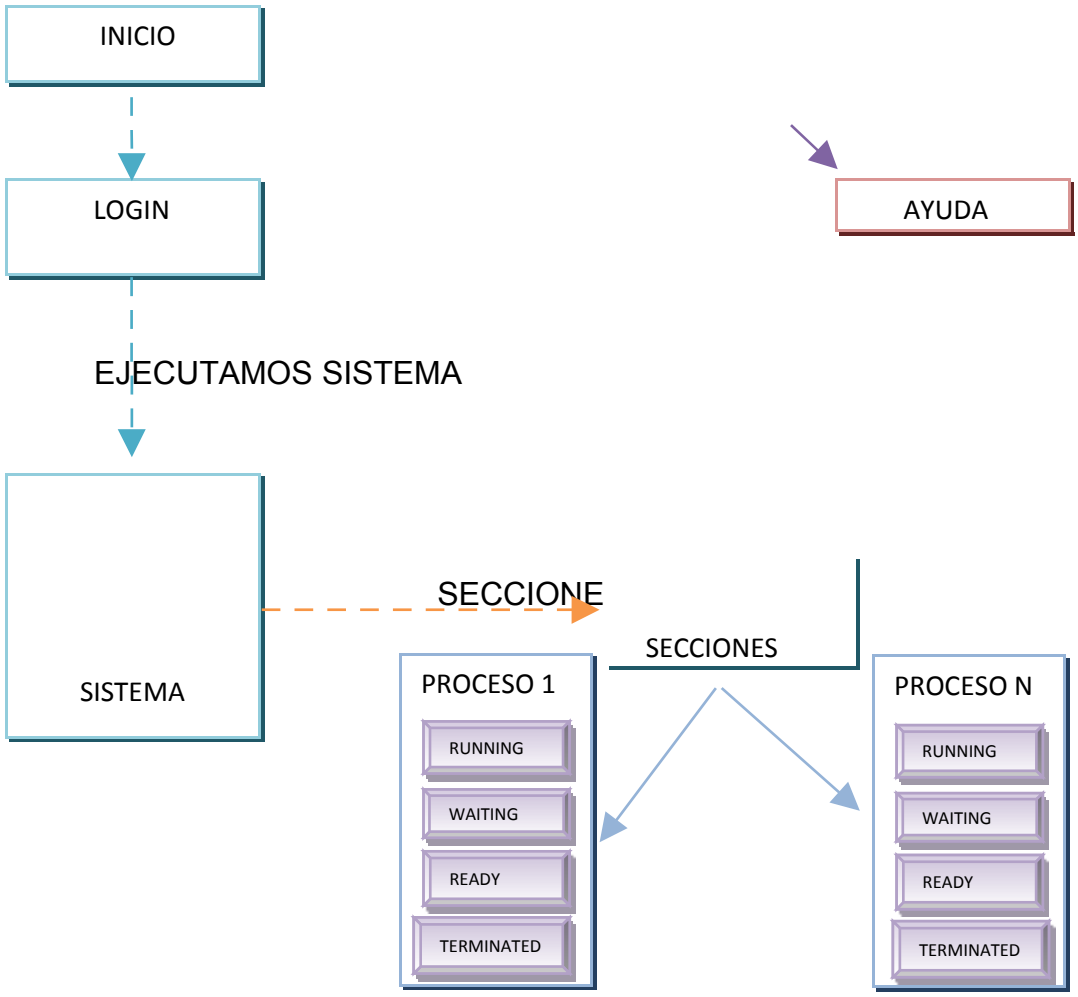


Figura 4.3 CN1 contexto Navegacional módulo Secciones. Rol Administrador. Elaboración propia.

Rol Usuario NO Registrado

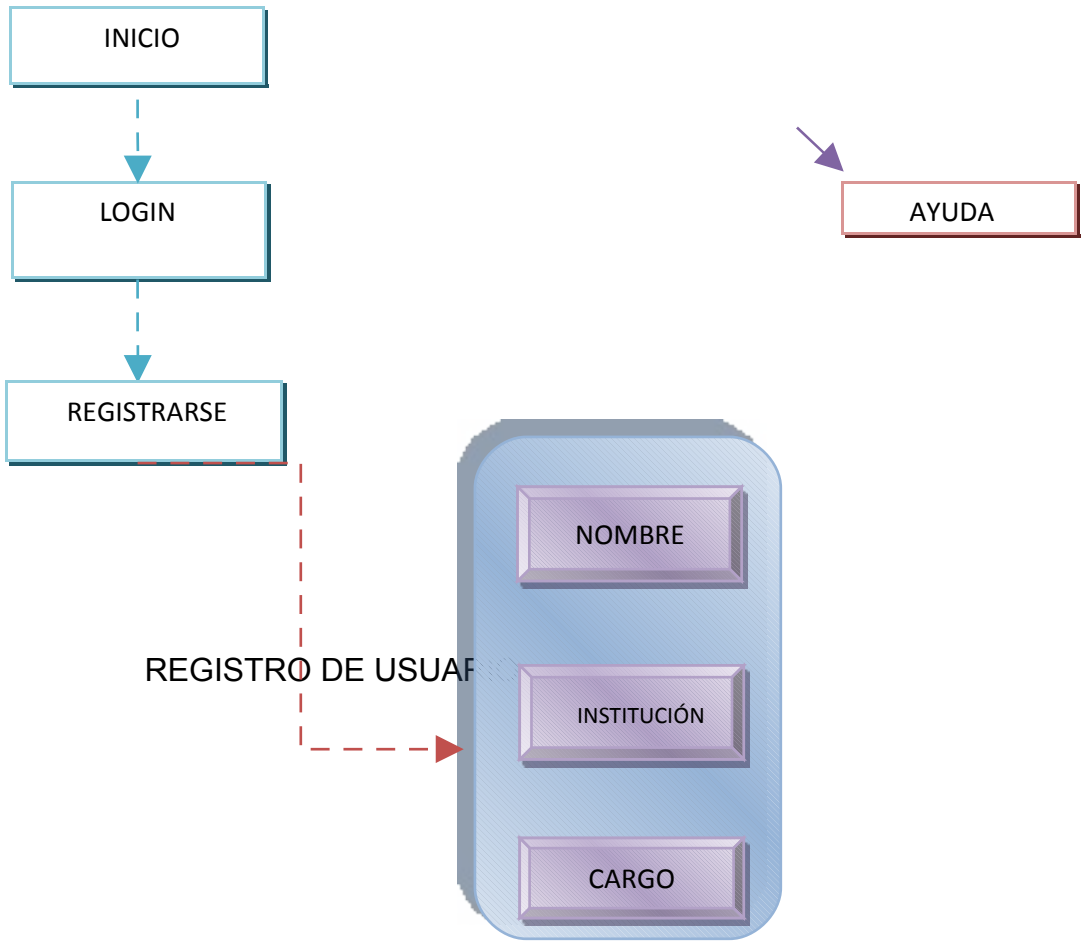


Figura 4.4 CN2 contexto Navegacional módulo Eventos. Rol Usuario no Registrado. Elaboración propia.

Rol Usuario Registrado

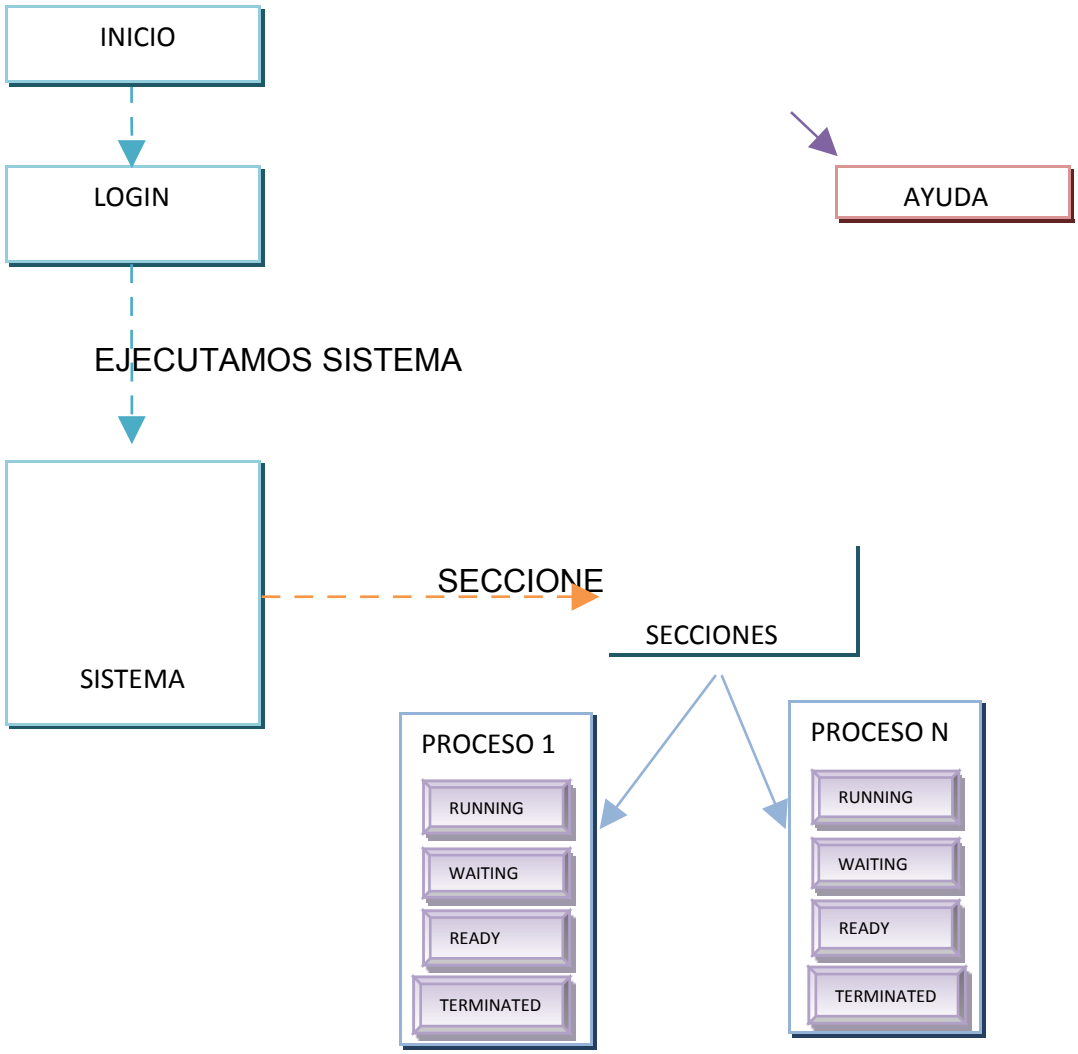


Figura 4.5 CN3 contexto Navegacional módulo Secciones. Rol Usuario Registrado. Elaboración propia.

4.4 Diseño de Interfaz Abstracta

Modelo General ADVs

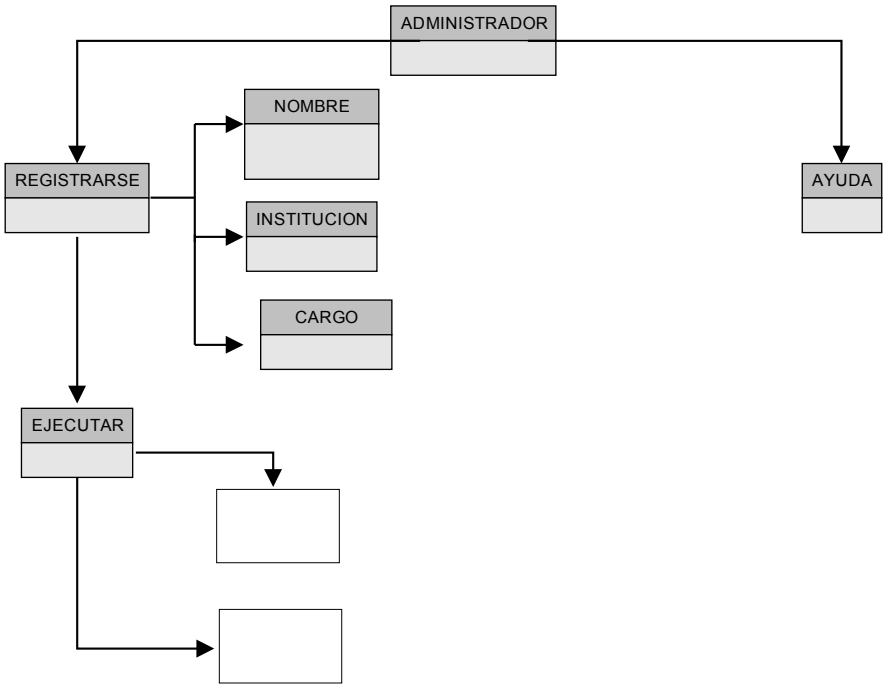


Figura 4.6 Modelo General de ADVs

ADVs

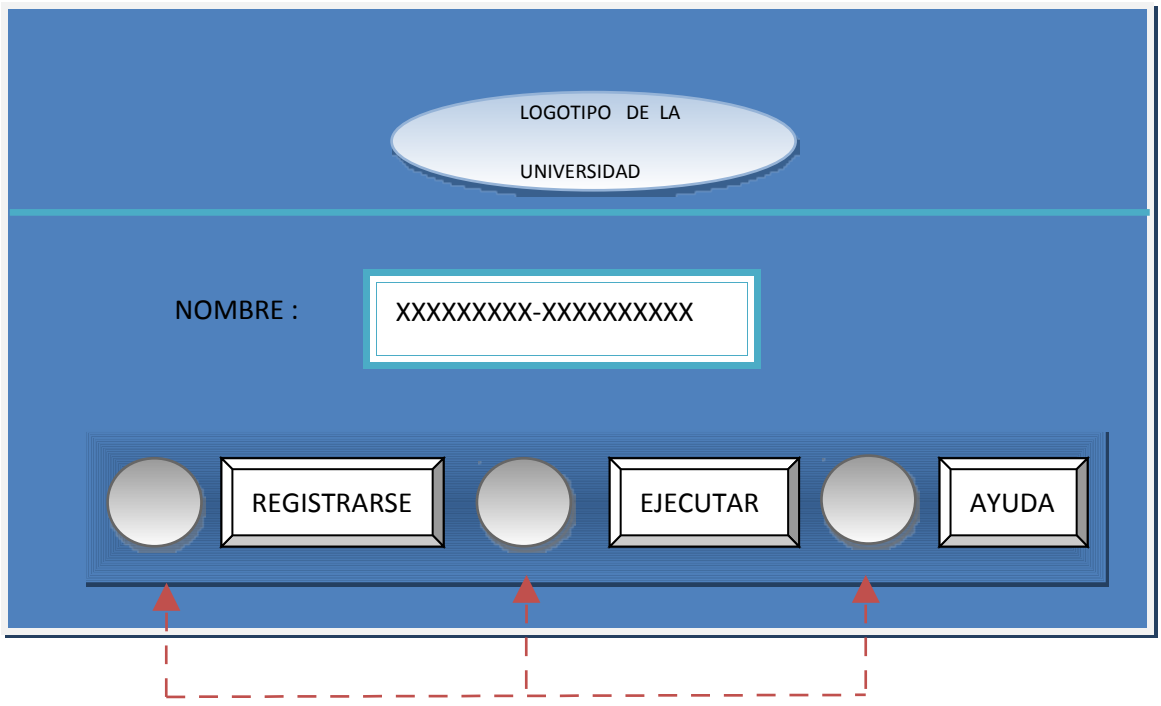


Figura 4.7 ADVs Inicio. Rol Visitante. Elaboración propia

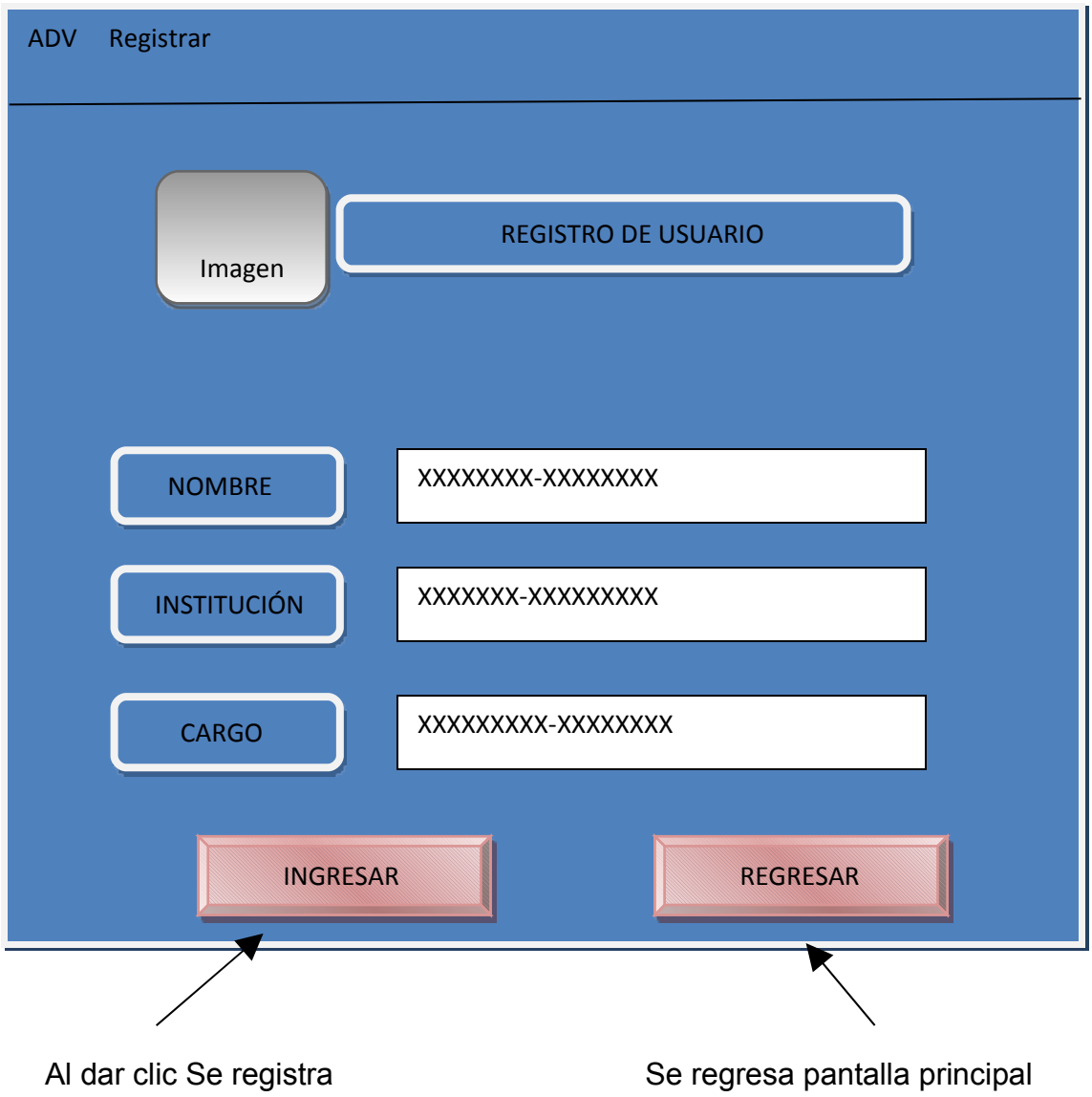


Figura 4.8 ADVs Registrarse. Elaboración propia.

ADvs EJECUTAR

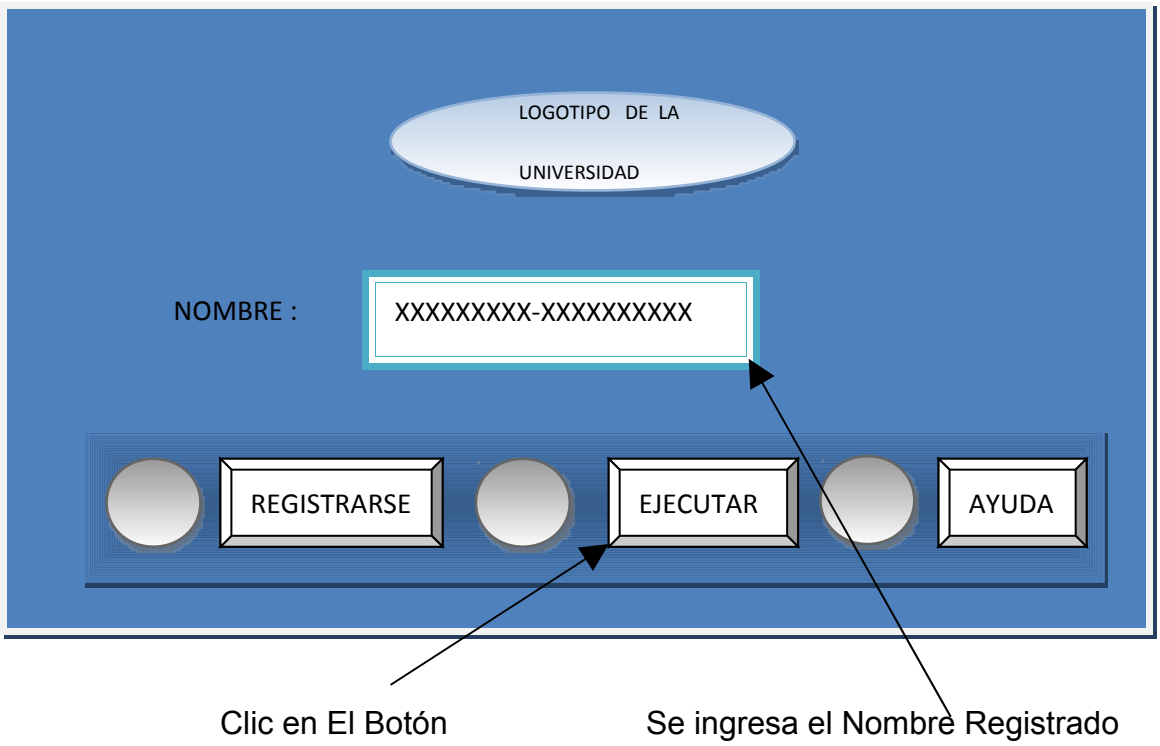


Figura 4.9 ADVs Inicio. Rol Administrador/Usuario. Elaboración propia

ADVs

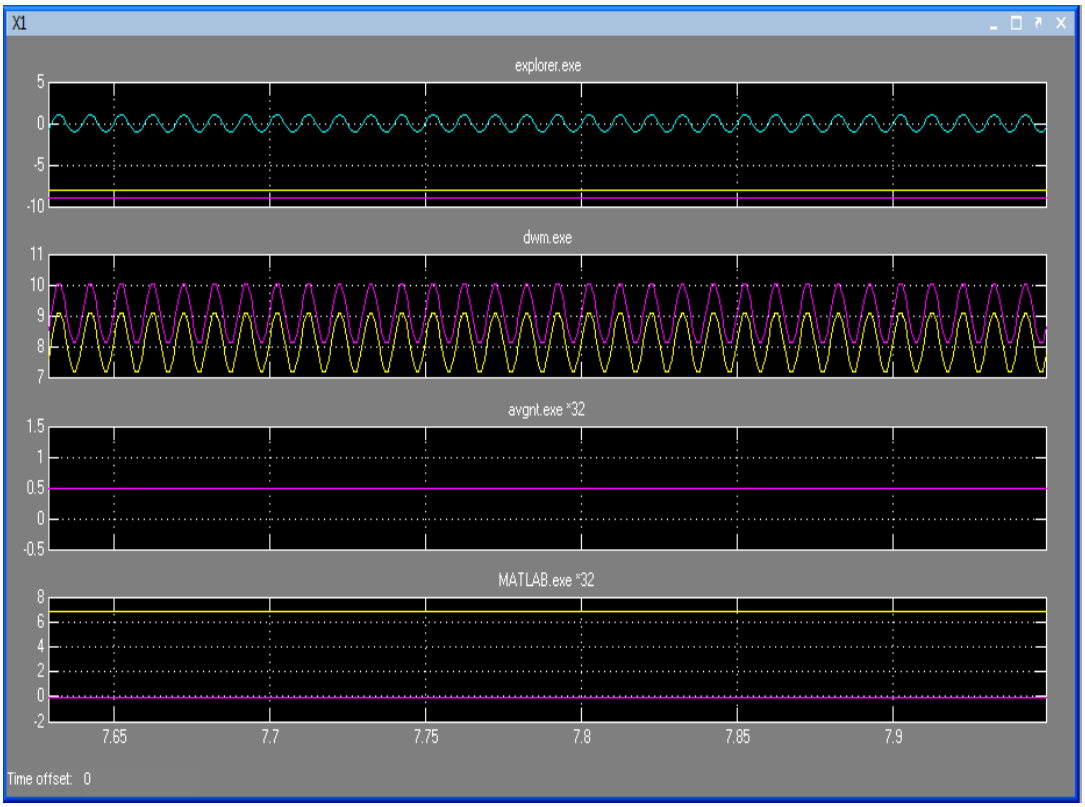


Figura 4.10 ADVs Muestra los Procesos que se están ejecutando. Elaboración Propia

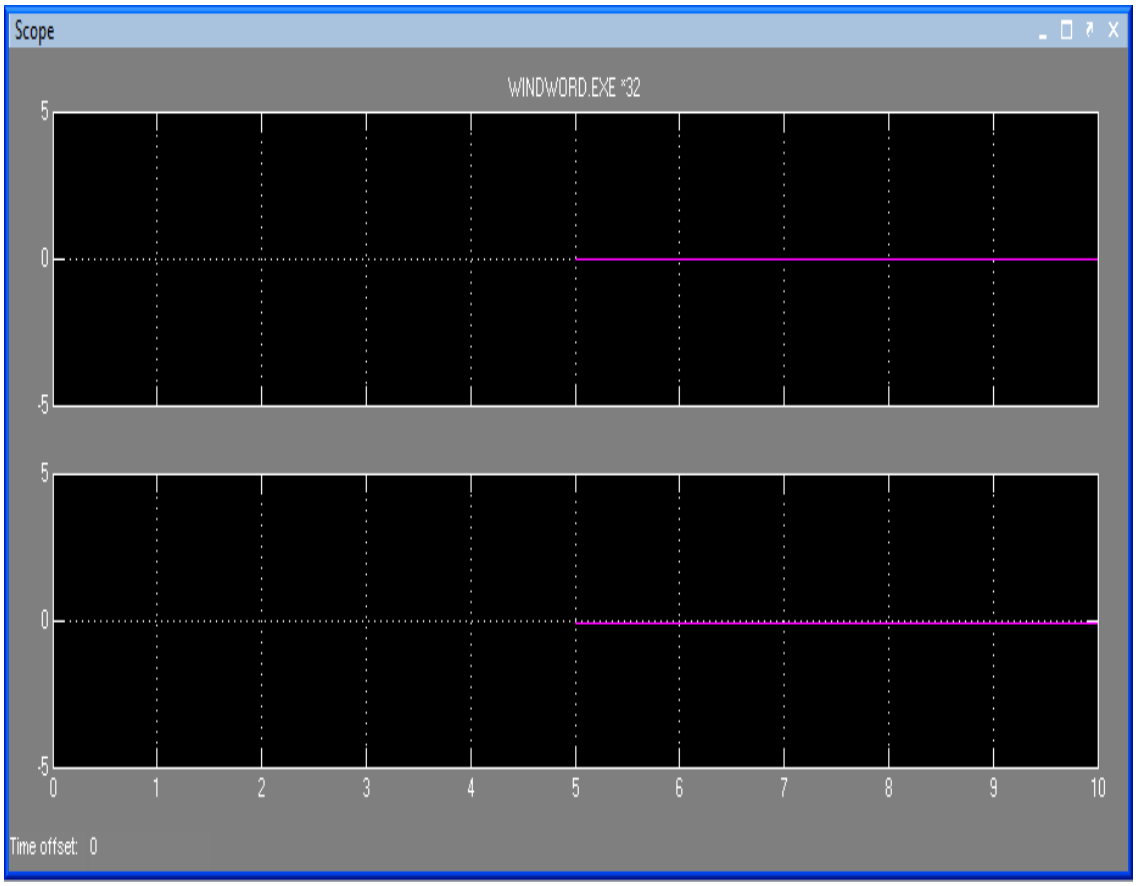


Figura 4.11 ADVs Muestra los Procesos que se están ejecutando. Elaboración Propia

ADVs USUARIO NO REGISTRADO

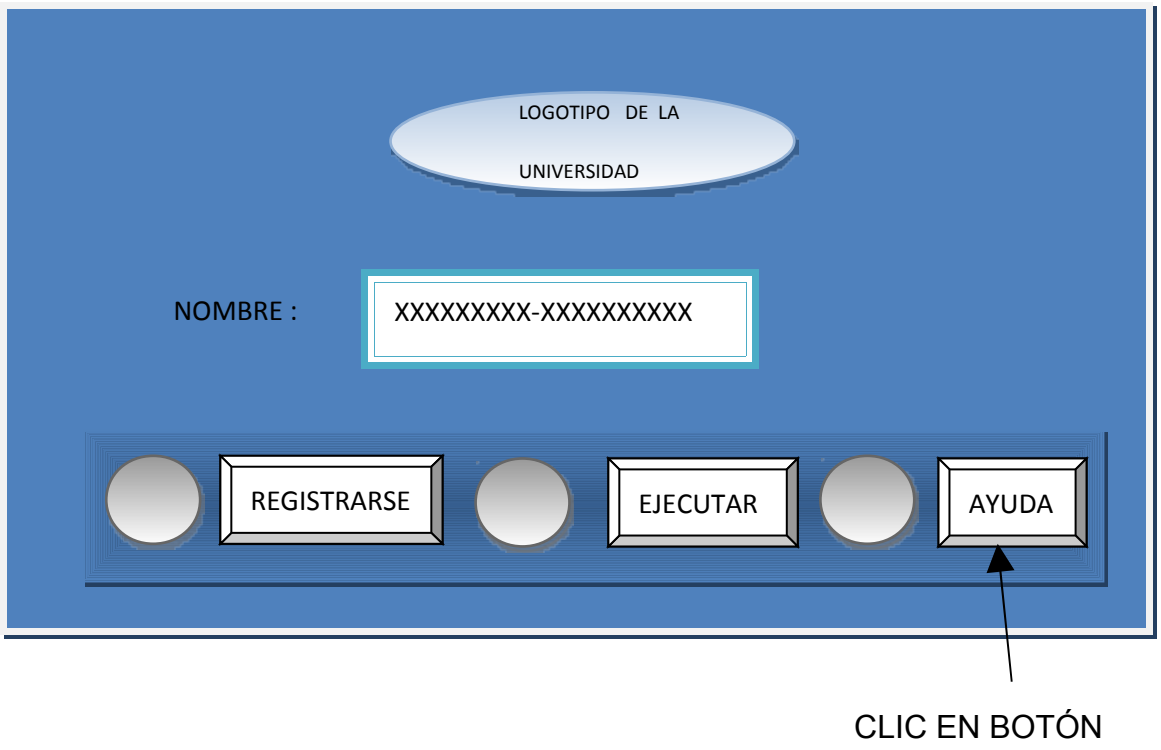


Figura 4.12 ADVs No Registrado. Registrar Usuario para ingresar al Sistema.

Elaboración Propia

ADVs Ayuda

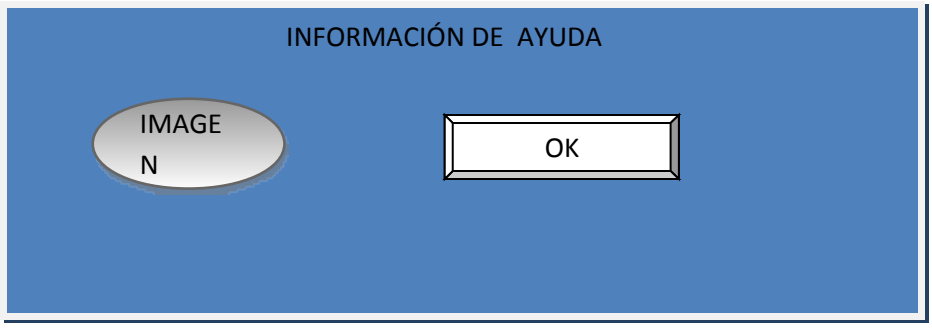


Figura 4.13 ADVs Ayuda. Informa que se debe hacer para ingresar al sistema.

Elaboración Propia

4.5 Implementación

Las etapas anteriores de la metodología han permitido obtener un claro conocimiento de la información que será mostrada en el Sistema, además de las funciones que ejecutan procesos en el sistema, además de una noción básica de las interfaces que tendrá el mismo.

4.5.1 Aplicación ejecutable

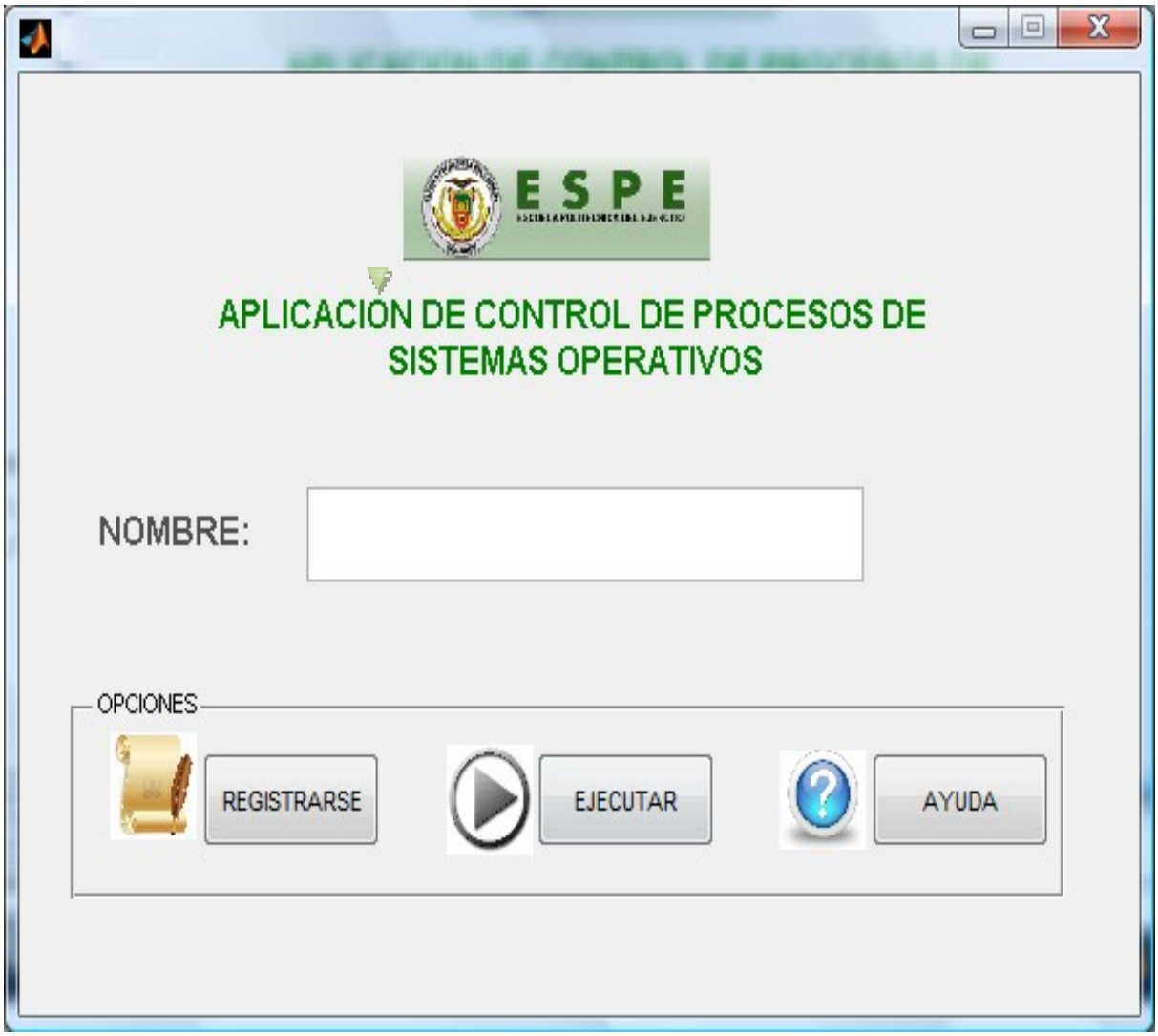


Figura 4.14 CONTROL DE PROCESOS. Elaboración Propia

Pruebas

Pruebas de Unidad

Para realizar las pruebas se ha tomado como unidad algunos equipos para verificar sus procesos.

A continuación se muestran las pruebas realizadas a las distintos procesos de distintos CPU, teniendo en cuenta los controles, condiciones y contenidos de las mismas.

Cuadro 4.1 Acceso normal al sistema

Página:	PRINCIPAL		
Contenido:	Despliegue correcto del contenido.		
Precondición:	Ninguna		
Poscondición:			
Rol:	Administrador		
Prueba	**Operación/Control**	**Respuesta esperada**	**Resultado Obtenido**
Ingresar usuario y contraseña correctamente.	Validar datos ingresados con la BDD.	Muestra página de Administración del Sistema	Aparece la pantalla principal del Sistema.
Ingresar usuario y contraseña erróneos.	Validar datos ingresados con la BDD.	Muestra mensaje de error.	Aparece mensaje de error del Sistema.

Cuadro 4.2 Ingresar Sistema

Página:	X1		
Contenido:	Despliegue correcto de la Simulación de los procesos		
Precondición:			
Poscondición:			
Rol:	Administrador		
Prueba	**Operación/Control**	**Respuesta esperada**	**Resultado Obtenido**
Ingresar Nueva Sección.	Registrar en la BDD.	Mostrar SistemaX1 Y SCOPE.	Aparece Sistema con los Resultados
No ingresar datos del artículo y presionar el botón guardar sección.	Validar datos ingresados.	Mostrar mensaje de error en la misma sistema.	Aparecen alertas de error en la mismo sistema.

Cuadro 4.3 Ingresar nuevo Administrador

Página:	REGUS		
Contenido:	Despliegue correcto del Sistema de Usuarios.		
Precondición:			
Poscondición:			
Rol:	Administrador		
Prueba	**Operación/Control**	**Respuesta esperada**	**Resultado Obtenido**
Ingresar Nuevo Administrador.	Validar datos ingresados y registrarlos en la	Mostrar página de Usuarios.	Aparece la página de Usuarios.

	BDD.		
Ingresar Nuevo Administrador.	Validar datos ingresados.	Mostrar mensaje de error en la misma página.	Aparece mensaje de error en la misma página.
No ingresar datos de Administrador el botón guardar Administrador.	Validar datos ingresados.	Mostrar mensaje de error en la misma página.	Aparecen alerta de error en la misma página.

Cuadro 4.4 Ingresar nuevo Usuario

Página:	REGUS		
Contenido:	Despliegue correcto de la página de Usuarios		
Precondición:			
Poscondición:			
Rol:	Administrador		
Prueba	**Operación/Control**	**Respuesta esperada**	**Resultado Obtenido**
Ingresar Nuevo Usuario.	Validar datos ingresados y registrarlos en la BDD.	Mostrar página de Usuarios.	Aparece la página de Usuarios.
Ingresar Nuevo Usuario.	Validar datos ingresados.	Mostrar mensaje de error en la misma página.	Aparece mensaje de error en la misma página.
No ingresar datos de Usuario el	Validar datos ingresados.	Mostrar mensaje de error en la	Aparecen alerta de error en la misma

botón guardar Usuario.		misma página.	página.

Conclusiones de las Pruebas de Unidad

Al realizar las pruebas de unidad se concluye lo siguiente:

El ingreso al sistema es realizo mediante Usuario y Contraseña.

Existen controles en el sistema al momento de ingresar contenido en el mismo lo cual le permite al usuario tener una guía en el manejo del sistema.

Se verifican los procesos que se están ejecutando en las maquinas y los que no se están ejecutando.

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5. Conclusiones

5.1 Conclusiones

Una vez realizado el estudio de factibilidad del presente proyecto, se tiene información necesaria y suficiente que permita llegar a las siguientes conclusiones:

- Se realizó un análisis la metodología COMET, la que permitirá construir el Sistema de Control en Tiempo Real para el Kernel del Sistema Operativo. Este modelo de desarrollo de software denominado COMET, (Concurrent Object Modeling and Architectural Desing Method), es una metodología que implemente UML y está basada en un ciclo de desarrollo iterativo, tiene las siguientes fases: Modelo de requerimientos, Modelo de Análisis, Modelo de diseño, Construcción del Software, Integración del Software, Validación del Software.

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

- De acuerdo a los requerimientos de usuario 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.

5.2 Recomendaciones

- Se recomienda utilizar la metodología COMET, ya que permite el diseño de Sistemas, con esta metodología se involucrar mayores costos de diseño al principio, pero a mediano y largo plazo reducen notablemente los tiempos de desarrollo al tener como objetivo principal la reusabilidad de diseño, y así simplificar la evolución y el mantenimiento.

- Se sugiere que en el CPU que se va a realizar dicho procedimiento tenga instalado MATLAB, para el manejo de Simulink ya que es un simulador y no se puede hacer ejecutables de un simulador.

- Se recomienda usar Matlab portable para el uso de este programa, por lo que este no es muy pesado.

- En el sistema deben estar registrados los usuarios para poder utilizar, se recomienda que se realice la vista de procesos cuando el computador se encuentre ejecutando varios procesos para ver si algún proceso no se encuentra empotrado o mal utilizado.

- Como desarrollador del software se aconseja asignar una mayor cantidad de tiempo a la fase de obtención de requerimientos ya que esta es la base para empezar el desarrollo de un sistema.

Glosario

Actor.- Es aquel que proporciona o recibe directa o indirectamente datos al sistema.

Administrador.- Súper usuario, el cual se encarga de la administración del sistema, debe estar registrado para hacer uso del sistema.

ADVs – Vista de Datos Abstracta.- Especifican la organización y el comportamiento de la interfaz.

Caso de Uso.- Es un conjunto de actividades que se ejecutan ordenadamente para entregar un resultado importante al actor.

Clase.- Es una descripción de un grupo de objetos.

Metodología: Se encarga de elaborar estrategias de desarrollo de software que promuevan prácticas adoptativas en vez de predictivas.

Rol.- Papel que desempeña alguien en un análisis. Un actor representa roles.

Usuario-. Sinónimo de Actor. Individuo que debe estar registrado para poder usar el sistema.

UML: Lenguaje de Modelamiento Unificado.

BIBLIOGRAFÍA:

- Teoría y simulación de sistemas asociativos
  Bresme Fernández, Fernando

- Herramienta para la implantación hardware de controladores sobre FPGAs

  Martínez Álvarez, José Javier - Guerrero González, Antonio - López Coronado

- *mat21.etsii.upm.es/ayudainf/.../**Matlab**70/**matlab**70primero.pdf*

- *mercur.utcluj.ro/.../**Simulink**%20Tutorial.**pdf***

- 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.

- Benchaib, S. Poullain, J.-L. Thomas, and J. Alacoque, "Discrete-Time Field-Oriented Control for SM-PMSM Including Voltage and Current Constraints," in *IEEE Electrical Machines and Drives Conference*, vol. 2, 1-4 june 2003, pp. 999–1005.

WEB Biografia

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

- http://dspace.espoch.edu.ec/bitstream/123456789/632/1/38T00250.pdf

- http://www.esi2.us.es/~jaar/Datos/RegAuto/Practica3.pdf

- http://users.isr.ist.utl.pt/~alex/micd0506/simulink.pdf