Revista Iberoamericana de Automática e Informática industrial

ISSN: 1697-7912. Vol. 8, Núm. 1, Enero 2011, pp. 64-79

www. revista-riai.org

Núcleo de Control para Sistemas Empotrados de Control: Una propuesta de Arquitectura.

Adel Fernández Prieto, Orestes Llanes-Santiago

Dpto de Automática y Computación Ave: 114, No:11901, e/119 y 127, CUJAE, Marianao Ciudad de La Habana, Cuba
(e-mail:adel,orestes@electrica.cujae.edu.cu)

Resumen
En este trabajo se define la arquitectura de un *núcleo de control* (**NC**) que permite ejecutar aplicaciones de control empotradas, así como los principales problemas que soportan su desarrollo. Se propone la estructura de los datos y los servicios que debe proveer este núcleo para aplicar varias estrategias de control que pueden ser utilizadas para garantizar seguridad, fiabilidad y economía en sus operaciones. En el trabajo se presentan dos ejemplos que ilustran el uso del *núcleo de control* . *Copyright © 2011 CEA*.

Palabras Clave:Sistemas Empotrados de Control, Sistemas de Tiempo Real, Núcleo de control, Escasez de recursos.

1. INTRODUCCIÓN

Los sistemas de control empotrados (SEC) operan con un grupo de restricciones donde los principales recursos tales como, potencia de procesamiento, memoria o facilidades de comunicación están limitados. Además, trabajan en tiempo real y necesitan ser capaces de enviar acciones dentro de un estricto plazo de entrega o *deadline* que garantice una operación fiable y segura del sistema.

Los sistemas de control empotrados aparecen en una amplia variedad de aplicaciones entre las que se puede mencionar a la aeronáutica, la mecatrónica e industria automovilística entre otros. La mayoría de las veces, las restricciones son de tiempo real crítico (hard real time). Esto significa que una acción de control debe ser enviada considerando la disponibilidad de recursos. Por lo tanto, un SEC debe ser robusto, tolerante a fallos, autónomo, reconfigurable, compacto, y de consumo reducido. (Albertos *et al.*, 2005*a*)

Los principales problemas que las aplicaciones de control deben tener en cuenta cuando se diseña un SEC son la existencia de incertidumbres, objetivos y requerimientos imprecisos, cambios en las condiciones de operación y la disponibilidad de recursos. Esto significa que a partir del diseño debe resultar un sistema capaz de enviar acciones de control que garanticen el mejor desempeño bajo una variedad de situaciones.

Desde un punto de vista de control, un SEC puede ser diseñado utilizando diferentes metodologías como por ejemplo: control robusto, control supervisor o control óptimo (consumo, calidad en los servicios). Sin embargo, el reto principal es diseñar e implementar un sistema de control provisto del mejor desempeño bajo diferentes escenarios y que garantice una operación segura. Esto tiene mucho que ver con la manera en que la solución de control es implementada bajo un entorno de reducida disponibilidad de recursos.

Las fases de diseño del control e implementación deben ser abordadas de conjunto (Årzén *et al.*, 2000). Para lograrlo, las operaciones de un sistema de control basado en computadoras debe ser analizado, garantizando que se cumplan las actividades críticas no importa las condiciones en que esté operando el sistema.

Actualmente se asumen un conjunto de elementos acerca de la implementación del control en la etapa de diseño(Albertos *et al.*, 2007); (ARTIST2, 2005); (Årzén and Cervin, 2005):

- Un sistema de adquisición se encargará de proveer los datos del proceso.
- Los actuadores envían las señales de control en el tiempo adecuado.
- El procesador calcula en tiempo las acciones de control.
- Los datos requeridos para realizar los cálculos son almacenados en memoria
- El patrón de muestreo es regular (constante, síncrono y uniforme para cualquier tarea de control).
- La energía y potencia consumida está garantizada.

Sin embargo, cuando se trata de los SEC es difícil poder garantizar el cumplimiento de todas estas premisas. El objetivo de este trabajo es el diseño de una arquitectura de software, cuyo eje fundamental está basado en una capa de software denominada núcleo de control (NC), que permita ejecutar uno o varios lazos de control, brindando a cada uno ellos un desempeño adecuado cuando no es posible cumplir con las premisas anteriores, debido a la escasez de recursos.

La estructura de este artículo es la siguiente: En la sección 2 se presentan los principales problemas que existen en la implementación de los SEC. En la sección 3 se propone una solución basada en el concepto de núcleo de control. En la sección 4 se presenta la arquitectura del núcleo de control, la estructura de los datos que la soporta y los principales servicios que la componen. En la sección 5 se desarrollan

Publicado electrónicamente: 04/01/2011
DOI:10.4995/RIAI.2011.01.09

algunos ejemplos que ilustran el uso del núcleo de control. Finalmente se presentan las conclusiones del trabajo.

2. PROBLEMAS EN EL DESARROLLO DE SEC

En esta sección se establecen los principales problemas que afectan el diseño e implementación de un SEC. El análisis se divide en dos partes, la primera contiene los llamados *problemas de tiempo* que básicamente afectan el cumplimiento del período de muestreo de un controlador. Además, se analizan las principales causas que les dan origen. En la segunda parte se analiza de manera resumida los problemas de energía que deben ser tenidos en cuenta en el diseño de un SEC.

2.1 Problemas de tiempo

Una de las razones fundamentales por la que los SEC presentan *problemas de tiempo*, se debe a la interacción de las tareas que se ejecutan bajo una determinada política de planificación. Por interacción de tareas se entiende el grado de interferencia que afecta el tiempo de activación y respuesta de una tarea debido a que otras, con mayor prioridad, están activas o listas para su ejecución.

Desde hace algunos años han comenzado a aparecer en la literatura científica numerosos artículos y trabajos relacionados con esta temática. Algunos ejemplos de los mismos son (Balbastre, 2002), (Cervin, 2003), (Albertos *et al.*, 2007), (Crespo *et al.*, 2006), (Crespo A., 1999).

La figura 1 muestra los principales parámetros temporales que intervienen en la ejecución de una tarea de control de tiempo real y cuales son los *problemas de tiempo* que se presentan. El análisis se ha divido en dos intervalos típicos de muestreo k y $k+1$ pertenecientes a una tarea con prioridad menor que la máxima.

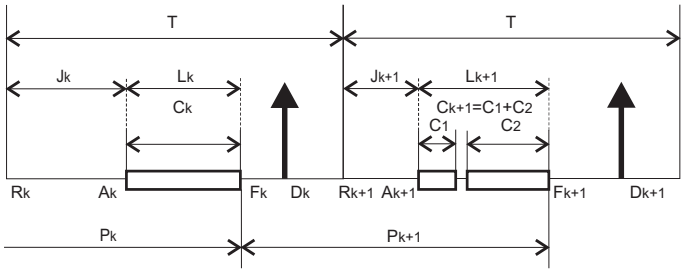

Figura 1. Problemas de tiempo en sistemas de control

T es el período nominal de la tarea periódica que se activa en los instantes R_k, R_{k+1} y R_{k+2}. Sin embargo, la activación de la tarea no implica que sea atendida inmediatamente. A_k, A_{k+1} son los instantes donde se atiende la tarea de control y en general se asume que $A_{k+1} - A_k \neq T$.

Cada tarea de control consume un tiempo de ejecución C_k variable, que dependerá de la ley de control seleccionada y los medios de comunicación utilizados. Además, si otras tareas de mayor prioridad se deben atender, entonces este tiempo puede verse afectado. Por lo tanto se asume que $C_k \neq C_{k+1}$.

F_k representa el instante en el cual finaliza la activación de la tarea para el período actual. También se cumple que $F_{k+1} - F_k \neq T$ debido a que la activación y el tiempo de cómputo son variables.

P_k es el período real al cual se está ejecutando la tarea. D_k es el plazo de entrega o *deadline*, es decir el instante de tiempo en el cual debe ser entregada la acción de control para cumplir la restricción temporal del sistema controlado.

A continuación se definen los parámetros temporales más importantes que se tienen en cuenta para analizar los sistemas de control en tiempo real.

Definition 1. La latencia $L_k = A_k - F_k$ es el retardo de control y se refiere a la variación entre el comienzo A_k y finalización F_k de una tarea de control en un intervalo de muestreo.

Definition 2. El *jitter* $J_k = A_k - R_k = F_k - L_k$ es el tiempo transcurrido entre la activación de la tarea R_k y su atención A_k en un intervalo de muestreo.

Para lograr una buena implementación de una aplicación de control digital, es necesario que el período de ejecución real del controlador P_k, sea igual o muy similar al período nominal T para el cual fueron obtenidos los parámetros de diseño del controlador.

Proposición 1. Para que el período real de ejecución P_k sea igual al período nominal del controlador T es necesario que se cumpla una de las siguientes condiciones:

- Que $\Delta L_k = L_k - L_{k-1}$ y $\Delta J_k = J_k - J_{k-1}$ sean cero respectivamente, lo cual significa que entre dos intervalos de muestreo consecutivos tanto el *jitter* ($J_k = J_{k-1}$) como la latencia ($L_k = L_{k-1}$) permanecieron constantes.
- Que $\Delta L_k = -\Delta J_k$. Lo cual significa que el jitter y la latencia son iguales en magnitud pero de signo contrario.

Demostración:

A partir de la figura 1 se cumplen las siguientes relaciones:

$$F_k = L_k + J_k + R_k$$
$$P_k = F_k - F_{k-1}$$
$$P_k = (L_k + J_k + R_k) - (L_{k-1} - J_{k-1} - R_{k-1})$$
$$P_k = (R_k - R_{k-1}) + (L_k - L_{k-1}) + (J_k - J_{k-1})$$

Si:

$$\Delta L_k = L_k - L_{k-1}$$
$$\Delta J_k = J_k - J_{k-1}$$

Entonces se obtiene la siguiente expresión:

$$P_k = T + \Delta L_k + \Delta J_k$$

$\diamond$

ΔL_k y ΔJ_k son los valores que influyen en el incumplimiento del período nominal para el cual fue diseñado el controlador.

La variación en el período nominal causa variados efectos en los SEC. En (Albertos *et al.*, 2000) se concluye que mientras mayor esfuerzo de control se realiza para desplazar los polos de lazo abierto hacia los polos deseados en lazo cerrado, mayor incidencia tendrá la variación de los parámetros temporales en el desempeño del sistema.

En (Kao and Lincoln, 2004) se presenta un teorema de estabilidad para lazos de control con latencia variable. Basado en este teorema se define el concepto de *margen de jitter* para tareas de control (Cervin *et al.*, 2004), inspirado en el *margen de fase*, *margen de ganancia* y el *margen de demora*, que están definidos en la teoría de control clásica. Al calcular el *margen de jitter* se puede determinar cual es su valor máximo antes de que el sistema comience a disminuir su desempeño.

Otros efectos derivados de la variación del período nominal se abordan en (Årzén *et al.*, 1999), (Årzén and Cervin, 2005),(Crespo A., 1999), (Cervin *et al.*, 2003).

Causas de los problemas de tiempo Tal y como se mencionó anteriormente, la causa más importante de los *problemas de tiempo* se debe a la ejecución de un conjunto de tareas de tiempo real sobre un sistema digital basado en un microprocesador por las interacciones que pueden existir entre ellas. Estas interacciones pueden tener varias causas. Algunos ejemplos de las mismas son: el bloqueo de recursos, la habilitación de secciones críticas y semáforos, el desplazamiento de su ejecución por otra tarea de mayor prioridad o los ciclos de espera activos. En general es deseable que este tiempo de espera no exista o que sea despreciable en relación con su período de ejecución. Sin embargo, esto muchas veces no es posible.

Una segunda causa que interviene en los *problemas de tiempo* tiene que ver con la planificación de los sistemas de tiempo real. Para hacer este análisis se utilizan los tiempos de ejecución del peor caso ($WCET$), definido aquí por sus siglas en inglés. Generalmente los $WCET$ se consideran fijos, sin embargo, cuando se trata de tareas de control es importante considerar los tiempos de respuesta que están sujetos a incertidumbres motivadas por diferentes razones. Entre algunas de ellas se pueden mencionar los retardos inherentes a una red de comunicación donde estén situados los elementos de acción final y los sensores ya que estos retardos estarán en función de la tecnología, los nodos de la red y la topología escogida. Además, si el algoritmo de control escogido tiene un tiempo de ejecución variable, debido a una ley de control heurística, la incertidumbre es mayor.

2.2 Problemas de energía

Una gran cantidad de aplicaciones de los SEC soportan el funcionamiento de los mismos a partir de una alimentación por baterías por lo que otro elemento a tener en cuenta en el diseño e implementación de un SEC es la necesidad de ahorrar energía. En estos casos eso se traduce como la necesidad de ahorrar tiempo de procesamiento, ya sea en los elementos sensores, actuadores o a nivel del procesador utilizado, brindando con ello una mayor autonomía al sistema de control y tiempo de vida a las baterías.

3. UNA PROPUESTA DE SOLUCIÓN: EL NÚCLEO DE CONTROL

La solución a la incertidumbre que presentan los SEC debido a los problemas de tiempo, puede enfocarse de dos formas diferentes: elaborando una solución a la medida, donde se tengan en cuenta los requerimientos particulares del sistema a controlar, o pensando en una solución más general que permita dar soporte a un conjunto más amplio de aplicaciones. En el primero de los casos, se podrán tener en cuenta aspectos específicos de hardware y software, mientras que en el segundo, la solución debería estar enfocada en mayor medida hacia el software debido a su propiedad de adaptación y flexibilidad.

Algunos aportes muy interesantes están basados en la definición de un nuevo modelo de planificador para SOTR. Tal es el caso del denominado (Servidor de Control) que es encargado de distribuir las tareas de control (Cervin *et al.*, 2002). Su inconveniente fundamental está en que es definido sobre un planificador del tipo EDF (el primero que cumple su plazo de entrega), por lo que su implementación práctica es bastante compleja. En ese mismo artículo, también se propone un planificador por realimentación de los recursos disponibles basado en un algoritmo PID. Esta propuesta está bien fundamentada de forma teórica pero de igual manera es difícil de implementar. En la misma línea está también la propuesta realizada por (Xia *et al.*, 2008), esta vez utilizando redes neuronales para lograr la realimentación del estado del planificador. Esta técnica, aunque novedosa, no es posible de implementar sobre los SOTR que están disponibles tanto en el mercado comercial como en el ámbito científico.

La propuesta que se presenta en este trabajo se soporta en el desarrollo de una capa de software denominada *núcleo de control* y está inspirada en la arquitectura de los sistemas operativos actuales. Para cumplir con las características básicas de una aplicación común de software, un sistema operativo provee una capa de abstracción que oculta a las aplicaciones los detalles del hardware sobre el cual se ejecutarán. Esto permite el desarrollo de aplicaciones sin considerar la plataforma sobre la cual serán utilizadas. Además, permite que en cada momento se ejecuten las actividades que mayor prioridad tengan.

El elemento novedoso en este caso, es la asignación particular de prioridades que se da a los diferentes subsistemas que conforman el núcleo de control. Su elección se justifica debido a la sencillez y fácil implementación en sistemas empotrados que soporten la ejecución de un SOTR comercial, que usualmente presentan un mecanismo de planificación por prioridades fijas.

Por tanto, para la ejecución de aplicaciones de control en un SEC se propone la existencia del *núcleo de control*(NC) (Albertos *et al.*, 2006) cuya definición es la siguiente:

Una capa intermedia de software que provee un grupo de servicios básicos que permitirán aislar el algoritmo de control, de detalles de ejecución tales como: planificación de las tareas de tiempo real, interacción con otros lazos de control, demoras en la adquisición o envío de acciones de control y disponibilidad de recursos energéticos. A su vez, esta capa de software estará soportada por un sistema operativo de tiempo real (SOTR) que proveerá los servicios de planificación y ejecución de tareas, entrada y salida (E/S) y temporizadores.

Según este nuevo modelo, la implementación de una aplicación de control utilizando el NC estará dividida en dos partes: aquella que se ejecutará como parte del NC y aquella que se ejecutará a nivel de aplicación. En la primera estarán las actividades consideradas más prioritarias, mientras que en la segunda se situarán las restantes. A continuación se realizará un análisis que permitirá posteriormente definir ambos grupos.

3.1 Características generales de una aplicación de control digital en un SEC

Para implementar una aplicación de control digital no se debe tener en cuenta solamente el código del algoritmo de control. Hay muchas otras actividades que son cruciales para lograr que el controlador cumpla su objetivo:

Definition 3. El conjunto de actividades necesarias para implementar una aplicación de control digital será:

- **A1**: Adquisición de los datos esenciales.
- **A2**: Detección de fallos y/o retardos en la medición.
- **A3**: Selección de la ley de control.
- **A4**: Cálculo de la ley de control.
- **A5**: Actualización del estado Interno del control.
- **A6**: Enviar la acción de Control.
- **A7**: Completar la adquisición de todos los datos del proceso
- **A8**: Evaluar el desempeño del lazo de control.

No todas estas actividades tienen la misma relevancia. Por ejemplo, hay muchos procesos que cuando están en su punto de equilibrio, el cálculo de la ley de control **A4** no es una actividad importante. Como consecuencia, la actualización del estado interno **A5** y el completar la adquisición de datos **A7** tampoco lo son, ya que solo tributan a **A4**. Sin embargo, para poder cumplir con el tiempo de muestreo del sistema la acción de control **A6** debe enviarse continuamente, no importa que esta tenga o no variación. Así mismo, adquirir aquellos datos marcados como esenciales **A1**, es muy importante debido a que mediante ellos se conocerá el estado del proceso y por tanto se realizará la detección de los fallos o retardos que existen **A2**. Las actividades de menos relevancia serán la evaluación del desempeño del sistema **A8** y la selección de la ley de control **A3**, ya que la primera se realizará solo cuando haya suficiente capacidad de cómputo y en el caso de **A3** solo cuando existan condiciones adecuadas.

De acuerdo al análisis realizado anteriormente, el siguiente orden de relevancia puede ser establecido:

$$R_{A6} > R_{A1} > R_{A2} > R_{A4} > R_{A7} > R_{A5} > R_{A3} > R_{A8}$$

Donde R_{Ai} es un valor que indica la relevancia de la actividad Ai según el orden lógico de ejecución listado anteriormente.

3.2 Organización y definición de las actividades del núcleo de control

Teniendo en cuenta el análisis del epígrafe anterior, en primer lugar debe decidirse cuales de las actividades necesarias para ejecutar una aplicación de control digital deben pertenecer al NC

En este sentido, se propone que el grupo de actividades compuesto por $[A6, A1, \langle A4\rangle, A2, A7]$ sea parte de las responsabilidades que debe asumir el NC, debido a que son actividades básicas y generales que siempre deben ser llevadas a cabo de una manera similar en cualquier aplicación, independientemente del proceso controlado. El resto de actividades pueden ser colocadas a nivel de aplicación debido al grado de dependencia que tienen con respecto al proceso en cuestión.

La notación $\langle Ai\rangle$ significa que esta actividad puede incluirse o no dentro del NC. Las leyes de control que puedan ser generalizadas, por ejemplo: control por realimentación del estado, o control P,PI, PID, se podrán incluir, mientras que otras más especificas serán excluidas del mismo.

Además, el NC debe garantizar un conjunto de actividades que se definen a continuación:

Definition 4. Ejecución de una aplicación de control en modo seguro: Se define como la capacidad que tendrá el NC para mantener la respuesta del sistema en valores que sean seguros para los elementos del entorno del proceso controlado, incluyendo a los seres humanos, ante la limitación de recursos.

Definition 5. Envío de acciones de control de respaldo: Se define como una acción de control alternativa que es enviada a la planta por el NC cuando no se dispone de los recursos necesarios para obtenerla mediante la ley de control establecida por diseño.

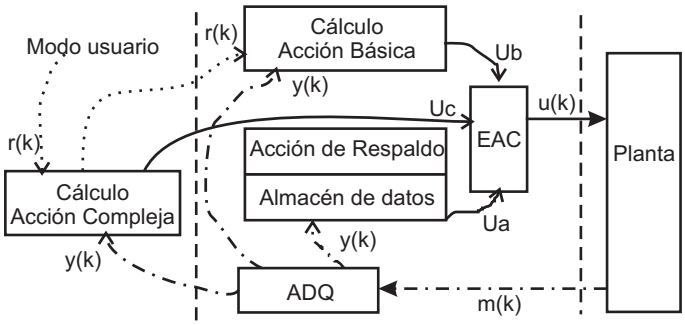

Figura 2. Actividades de Control

La Figura 2 muestra gráficamente las principales actividades del núcleo de control:

- **EAC**: Módulo de envío de la acción de control (EAC) que garantiza:
  - Acción de control básica (**Ub**) .
  - Acción de control compleja obtenida desde el modo de usuario (**Uc**).
  - Acción de respaldo basada en datos almacenados (**Ua**).
- **ADQ**: Adquisición de los datos (ADQ).
- **Almacén de datos/Acción de respaldo** : Almacenamiento de datos del proceso pasados y calculo de acciones de respaldo.

Donde $r(k)$ es la referencia, $m(k)$ las mediciones de la planta, $y(k)$ las mediciones filtradas, y finalmente $u(k)$ será el valor seleccionado como acción de control. Para garantizar la obtención de acciones de control de respaldo algunos datos deben ser almacenados y actualizados. Debe haber una tarea básica que se ocupe de organizar las acciones de control de respaldo utilizando una pila, ya sea por pre-asignación o como resultado de un conjunto de cálculos. Por ejemplo, en un control predictivo el algoritmo calcula una secuencia de acciones de control que pueden ser aplicables en los próximos intervalos de muestreo, si estas no son actualizadas en tiempo. Al ser almacenada la secuencia, esta puede ser usada en caso de que no haya una mejor opción. Otra opción será mediante algoritmos de estimación de variables, ya sea por extrapolación, u otro método conocido (Fernández *et al.*, 2009). En caso de emergencia esta acción podrá ser simplemente la anterior, o una acción segura que garantice la parada del proceso controlado, por ejemplo: cerrar una valvula o apagar un motor.

Es un objetivo fundamental del NC que los controladores se ejecuten en modo seguro. Esto significa que en aquellos instantes donde existe limitación de recursos, se debe aplicar alguna estrategia que permita mantener un correcto desempeño del sistema.

Teniendo en cuenta la bibliografía consultada (Albertos *et al.*, 2007), (Albertos *et al.*, 2005*b*), (Wittenmark *et al.*, 1995) se proponen como parte del NC dos tipos estrategias:

- El envío de una señal de control de respaldo, sobre todo en caso de pérdida o retardos en las mediciones.
- El intercambio de controladores o cambio de los períodos de muestreo, con el objetivo de reducir el consumo de energía, el tiempo de cómputo o la cantidad de variables de medición.

Basados en estas premisas se decidió que el NC podrá disponer de las siguientes leyes de control para ser aplicadas al proceso en caso necesario:

- Ley de Control Básica: Definida por algoritmos que presentan un tiempo de cómputo fijo y pequeño, ejemplo: controladores P, PI, PID, o por realimentación del estado. Además, deben cumplir la característica de ser parametrizables. Pueden pertenecer al NC
- Ley de Control Compleja: Definido por algoritmos que ocupan un mayor tiempo de cómputo como son los algoritmos adaptativos, predictivos, de optimización en línea o inteligentes entre otros. Nunca podrán pertenecer al NC
- Ley de Control Reducida: Definido por algoritmos basados en una ley de control compleja y que son reducidos por algún método conocido con el objetivo de que consuman un tiempo de cómputo menor que el original.
- Ley de Control Retardada: Definido por algoritmos basados en una ley de control compleja o básica que pueden estar sujetos a un cambio en los períodos de muestreo de controlador.

3.3 Supervisión dentro del núcleo de control

Como parte esencial del núcleo de control, además, debe existir una *actividad de supervisión* que permita tener en cuenta el comportamiento general del sistema de acuerdo al desempeño de las aplicaciones de control que se ejecutan, y se apliquen aquellas estrategias que permitan, en función de los eventos ocurridos y los recursos disponibles, mantener un adecuado desempeño del SEC. En este caso existen dos niveles que deben tomarse en cuenta:

Nivel de supervisión general Este nivel se encargará de mantener el estado del sistema como un todo, tomando en cuenta los recursos disponibles tales como la memoria, tiempo de procesamiento utilizado y energía consumida. De acuerdo a los índices de desempeño calculados para cada lazo de control se podrá tomar determinadas decisiones en dependencia de los recursos con que se cuenta en cada instante de tiempo.

Nivel de supervisión de planta Cada lazo de control ejecutado en el núcleo de control estará supervisado de forma individual, tomando en cuenta las ordenes recibidas del *nivel de supervisión general* y el estado interno del control.

La supervisión se ejecutará como una máquina de estados. Al registrar determinada configuración para un lazo de control quedarán definidos los diferentes estados por donde podrá transitar el control. Los posibles estados que pueden ser registrados para un control, en función del calculo de la ley de control, son los siguientes:

- Control Complejo: Definido por la ejecución de ley de control compleja.
- Control Básico: Definido por la ejecución de ley de control básica.
- Control reducido: Definido por la ejecución de una ley de control reducida.
- Control con retardo: Definido por la ejecución de una ley de control con un período de muestreo alterado.
- Acción de respaldo: definido por una acción de control de respaldo basada en datos almacenados.
- Acción segura: definido por una acción de control que se enviará en caso de emergencia.

Cada una de estas configuraciones debe ser registrada con un índice respecto del máximo tiempo de cómputo de la ley de control. Por ejemplo, la ley compleja normalmente representa un 100 %, que es el máximo tiempo de cómputo posible para calcular la señal de control. Luego, una acción de control reducida puede representar un 50 % de la misma. La figura 3 representa algunas de las posibles configuraciones que se pueden registrar en el núcleo de control.

Dentro del núcleo de control se pueden dar diferentes tipos de eventos, cada uno de los cuales será tratado de forma particular. La definición de los eventos está dada por la siguiente lista:

- Energía (E): Determinado por los niveles de batería medidos a través del sistema operativo. Es un evento del nivel general.
- Tiempo de Cómputo (TC): Determinado por un sobreconsumo de tiempo de procesamiento en el sistema. Es un evento del nivel general.
- Mediciones (M): Determinado por un retraso en la llegada de un dato a través de un canal de medición. Es un evento del nivel de planta.
- Plazos de entrega o *deadlines* (P): Determinado cuando algunas de las tareas de control pierde su plazo de entrega. Es un evento del nivel de planta.
- Desempeño (D): Determinado por la violación en valor de determinado índice de desempeño del sistema. Es un evento del nivel de planta.

La generación de los eventos va acompañada de parámetros que informan del estado en el cual se generó el mismo. Esto indicará el nivel de emergencia requerido y por consiguiente hacia cual estado deberá transitar el control de planta. Veamos un escenario donde se pongan de manifiesto algunas posibles transiciones:

Suponga un SEC con dos lazos de control cuyas configuraciones se muestran en la Tabla 1:

Tabla 1. Ejemplo de configuración de dos lazos de control

SEC			
Lazo 1		**Lazo 2**	
Estado	**Índice**	**Estado**	**Índice**
Control Complejo	100 %	Control Complejo	100 %
Control con Retardo	50 %	Control Reducido	60 %
Control Básico	20 %	Acción de Respaldo	5 %
Acción de Respaldo	5 %	-	-

Al lanzarse un evento del tipo E, el supervisor hará que el **Lazo 1** pase del estado *control complejo* al estado *control con*

retardo, lo cual supone una reducción del $50\,\%$ del tiempo de cómputo asignado para calcular el algoritmo de control de este lazo. El caso del **Lazo 2** es parecido, pero en este caso el nuevo estado será *control reducido* con un $40\,\%$ de reducción de tiempo de cómputo. Este nuevo estado de los lazos introducirá más disponibilidad de procesador o tiempo de ocio, el cual podrá ser utilizado para disminuir la tensión aplicada al microprocesador y por tanto el consumo energético.

Si el evento del tipo M ocurre para cualquiera de los lazos, el supervisor de nivel de planta pasará al estado *Acción de Respaldo* y enviará una acción de la pila de almacenamiento debido a que no se podrá calcular una acción de control basada en las mediciones actuales.

Al lanzarse un evento del tipo D para el **Lazo 1** se emite una situación de fallo que alcanza el nivel de supervisión general. En este nivel se determina cual es el lazo del sistema que, de acuerdo a su desempeño y configuración, será el afectado para eliminar esta condición anómala. Si se decidiera que es el propio **Lazo 1** entonces se pueden lanzar dos tipos de eventos posibles Dm o DM. La condición a verificar tiene que ver con la cantidad máxima de acciones de respaldo que se podrán enviar; Dm se lanza siempre que existan acciones de respaldo y DM en caso contrario. La figura 3 describe de forma gráfica la transición entre los estados. De acuerdo a la configuración registrada para el lazo, existen dos niveles posibles.

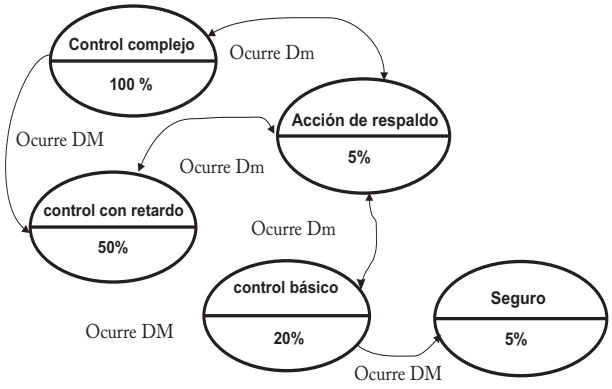

Figura 3. Supervisión por máquina de estados configurados

El estado *Acción de Respaldo* tiene un tiempo de vida igual a un período del controlador, inmediatamente se regresa al estado anterior, o sea nunca será un estado final del controlador, como si lo son los restantes. Esto es así, ya que este control es a lazo abierto y no debe permanecer activo por mucho tiempo.

4. ARQUITECTURA DEL NÚCLEO DE CONTROL

Según los conceptos e ideas desarrolladas en la sección anterior la arquitectura que se propone está basada en niveles. Para cumplir con sus responsabilidades cada nivel utilizará los servicios que proveen aquellos que están situados a nivel inferior. El NC está compuesto por un grupo de servicios que garantizarán la ejecución segura de varios lazos de control. La Figura 4 muestra estos niveles y sus interacciones:

donde:

- Medición de variables (ADQ): Se obtiene a través del sistema operativo y es utilizado tanto por el NC como por el nivel de aplicación.

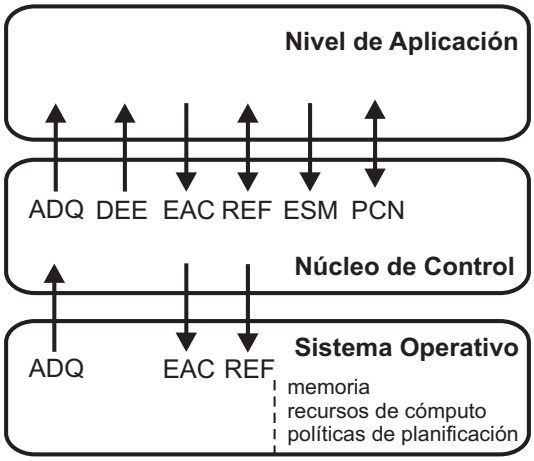

Figura 4. Capas e Interacciones del núcleo de control

- Referencias de control (REF): Son establecidas por el nivel de aplicación y en caso de emergencia también pueden ser manipuladas por el NC.
- Estimación de Salidas, entradas y referencias (ESM): de acuerdo a su nivel de complejidad pueden ser enviados desde el nivel de aplicación o calculados internamente por el NC.
- Protocolo (PCN): Conjunto de comandos que forman el protocolo de comunicación entre los niveles de aplicación y NC.
- Envío de las acciones de control (EAC): Se obtienen del nivel de aplicación y/o del NC y se envían a través del sistema operativo.
- Diagnóstico y estado de las entradas (DEE): Se analizan en el NC y se trasmiten hacia el nivel de aplicación.

La ordenes de mando se generan a partir de la *actividad de supervisión* que permitirá aplicar las estrategias de control, en el momento adecuado, a través de las siguientes acciones:

1. Chequeo de los recursos disponibles.
2. Chequeo del estado de las variables medidas del control.
3. Chequeo del desempeño de cada lazo de control.
4. Lanzamiento de eventos para intercambiar controladores o cambio de sus parámetros de ejecución.
5. Lanzamiento de eventos para el envío de acciones de control de respaldo en caso necesario.

El diseño del núcleo de la arquitectura del NC se basa en tres elementos fundamentales:

- La definición de las estructuras de datos de cada controlador en el núcleo de control.
- La definición de una plantilla de aplicación de usuario que garantice el cumplimiento del protocolo de intercambio entre el nivel de usuario y el núcleo de control.
- La definición de un grupo de servicios que garantizan el cumplimiento de sus objetivos y funciones.

4.1 Estructura de los controladores en el núcleo de control

Cada uno de estos servicios que se definen en las secciones siguientes está soportado por una estructura de datos que permite mantener la información de cada controlador. D_i es una tupla que define la estructura del lazo de control i que se desea ejecutar en el núcleo de control:

$$D_i = (F_i\ , CB_i\ ,\ I_i\ ,\ O_i\ ,\ R_i\ ,\ P_i\ ,\ S_i)$$

Donde:

- F_i: Función de usuario asociada con el lazo de control i.
- CB_i: Estructura que mantiene los parámetros de un controlador básico para el lazo de control i.
- I_i: Estructura que mantiene los parámetros asociados con la adquisición de datos pata el lazo de control i
- R_i: Estructura que mantiene los parámetros asociados con las referencias para el lazo de control i
- O_i:Estructura que mantiene los parámetros asociados con las salidas para el lazo de control i
- P_i: Parámetros Temporales relacionados con el lazo de control l_i
- S_i: Estado del lazo de Control i.

Cada lazo de control posee uno o más sensores, referencias o salidas. Su estructura es la siguiente:

I_i, R_i y O_i son vectores de datos de k, m y q dimensiones respectivamente ($I_i \in \mathbb{R}^k$, $R_i \in \mathbb{R}^m$, $O_i \in \mathbb{R}^q$ donde:

$$\begin{aligned} I_i &= \left\{i_i^1\ ,\ i_i^2\ ,\ \ldots, i_i^k\right\} \qquad \forall\, k \in \mathbb{N} \\ R_i &= \left\{r_i^1\ ,\ r_i^2\ ,\ \ldots, r_i^m\right\} \qquad \forall\, m \in \mathbb{N} \\ O_i &= \left\{o_i^1\ ,\ o_i^2\ ,\ \ldots, o_i^q\right\} \qquad \forall\, q \in \mathbb{N} \end{aligned}$$

y las estructuras de datos asociadas a i_i^k, r_i^m y o_i^q tienen la misma forma

$$i_i^k = r_i^m = o_i^q = \{e, \mathbf{B}\}$$

donde

e representa los dos posibles estados del dispositivo (sensor o actuador): fallo o no fallo.
$\mathbf{B} \in \mathbb{R}^{p+f}$ representa los p últimos valores y f valores futuros correspondientes al sensor o actuador.

Los valores futuros de las salidas son calculados en el algoritmo de control de nivel de usuario, siendo enviados al núcleo de control si existe la política de cálculo de acciones de respaldo configurada para el lazo de control i. En caso de no existir, entonces con un algoritmo de extrapolación interno del núcleo de control, es posible estimar los valores. En el caso de las entradas, esta variante no está implementada actualmente, aunque el espacio está reservado para usos futuros.

Es necesario además, almacenar los parámetros temporales correspondientes a cada lazo de control i:

$$P_i = (T_i\ ,\ d_i\ ,\ off_i\ ,\ w_i,\ pu_i)$$

donde

- T_i: Tiempo de muestreo
- d_i: Plazo de entrega
- off_i: Desplazamiento inicial entre la adquisición y el envío de la señal de control. Máxima latencia de entrada salida permitida.
- w_i: Tiempo de ejecución del peor caso (worst case executing time WCET) del algoritmo de control de nivel de usuario
- pu_i: prioridad de algoritmo de control de nivel de usuario del lazo de control i con respecto a los restantes lazos.

4.2 Aplicación de modo de usuario

El desarrollo de la aplicación de usuario está fundamentada en el cumplimiento del protocolo (PCN) que se presentó en la sección anterior. Este conjunto de comandos permitirá intercambiar información entre los niveles de usuario y el núcleo de control. Los comandos están formados por cadenas de caracteres que indican el tipo de configuración (*config*) que se desea colocar en el núcleo y un parámetro *extra* que será interpretado de acuerdo al comando.

La variable *config* puede tomar cuatro valores únicamente, siendo omitido en la declaración de función aquellos que no se deseen tener en cuenta para la supervisión:

- $config =' Ret'$: Significa una petición de cambio de período de muestreo, cuyo nuevo período se especificará en el parámetro *extra*.
- $config =' Red'$: Significa una petición de cambio de controlador. Si son más de dos el nuevo controlador se escoge de acuerdo al parámetro *extra*.
- $config =' Res'$: Significa una petición de cálculo de acción de respaldo de alto nivel, si es necesario algún parámetro debe realizarse en *extra*.
- $config =' Des'$: Significa una petición del índice actual de desempeño del sistema, utilizado por el supervisor para determinar el comportamiento del lazo de control. Esta llamada no devuelve una acción de control. El usuario debe garantizar que en cada período de muestreo se actualice el índice.

El formato general de la función que debe construir el usuario del núcleo es la siguiente:

```
Output:u usercontroller(Input: datain, config, extra)
.      Select (config)
.         case 'Ret':
.            configure_new_period(datain, extra);
.            u = compute_accion(datain) ;
.         case 'Red':
.            configure_reduced(datain, extra);
.            u = compute_action(datain) ;
.         case 'Res':
.            u = get_stored_action(datain) ;
.         case 'Des':
.            u = get_performance_index() ;
.         default:
.            u = compute_action(datain) ;
.      End-Select
.   Return u;
End-Function
```

Aquí las funciones *configure_new_period*, *configure_reduced*, son las que deben realizar el cambio de controlador y que no forman parte del núcleo de control, por lo que deben ser escritas por el usuario. Así mismo ocurre con las dos restantes *compute_accion*, *get_stored_action*, encargadas de calcular la acción de control y obtener una acción de respaldo apropiada. Así mismo, *get_performance_index()* es la encargada de devolver el valor de desempeño actual del sistema;

La función de usuario es registrada en el núcleo de control a través del parámetro F_i en cada lazo de control i.

4.3 Servicios básicos del Núcleo de Control

Este grupo de servicios está enfocado a dar soporte a las tareas de tiempo real que se están ejecutando a nivel de NC y a los errores y/o fallos que se producen durante su tiempo de vida. El diseño de estos servicios está basado en la utilización de un sistema operativo de tiempo real (SOTR) que soporte la planificación de tareas con prioridades fijas, que está bastante generalizado tanto en SOTR comerciales (FreeRTOS), como en SOTR con propósitos de investigación (Partikle, Shark). A continuación se hace la descripción de cada uno de ellos.

Servicio de administración de fallos: está relacionado con la detección y manejo de situaciones de fallos. Este servicio cuenta con varios métodos de detección de acuerdo a los tipos de señal manipuladas y el proceso controlado. Se consideran dos tipos de fallos:

1. Fallos intermitentes: Comienzan en un instante determinado, permanecen activos por un período de tiempo y luego desaparecen para aparecer posteriormente de forma aleatoria.
2. Fallos permanentes: Permanecen activos hasta tanto no se solucionen.

El servicio está enfocado a la determinación de los diversos fallos que pueden presentarse, algunos determinados por ventanas temporales de observación y métodos estadísticos de detección de fallos. Estos servicios podrán o no ser incluidos en una aplicación final de acuerdo a la disponibilidad de cómputo y recursos existentes.

Este servicio refleja sus resultados fundamentalmente en las estructuras I_i e R_i, actualizando específicamente el estado e de cada variable de entrada, si el fallo es del tipo intermitente y su presencia no afecta sensiblemente el funcionamiento del sistema. Si el fallo es de tipo permanente o es intermitente y afecta sensiblemente el comportamiento del sistema entonces sube hasta el nivel de supervisión de planta para ser tomado en cuenta en el estado general del sistema.

Servicio de detección de errores. La detección de errores se relaciona a las aplicaciones que conocen en detalle como deben ser manipulados los mismos. Dos tipos de errores deben ser considerados:

1. *Errores de tiempo*: Se presentan cuando no es posible cumplir con los *deadline* o períodos nominales de un controlador.
2. *Errores de valor*: Presentes por la evaluación de expresiones tales como la división por cero, o el incumplimiento de restricciones tales como los valores limites de la acción de control. El objetivo será que la aplicación no caiga en un estado de inconsistencia.
3. *Errores de perdidas de datos*: Está relacionado con la detección y manejo de situaciones anormales tales como la pérdida de datos.

Al presentarse reiteradamente errores de tiempo, el supervisor lanzará un evento de transición que permitirá aplicar algunas de las estrategias configuradas previamente que permitan reducir los tiempos de cómputo actuales y cumplir con los *deadline*. En el caso de los errores de perdidas de datos el tratamiento que se le da es similar al de fallos intermitentes que no afectan significativamente el comportamiento del sistema.

4.4 Servicios específicos del Núcleo de Control

Además de los servicios generales, otros más específicos, que tienen que ver con la ejecución de aplicaciones de control se proponen en esta sección.

La figura 5 muestra, de forma general, la interacción de estos servicios. En el caso del supervisor, el mismo mantiene intercambio con todos, de manera tal que permita cumplir con sus objetivos.

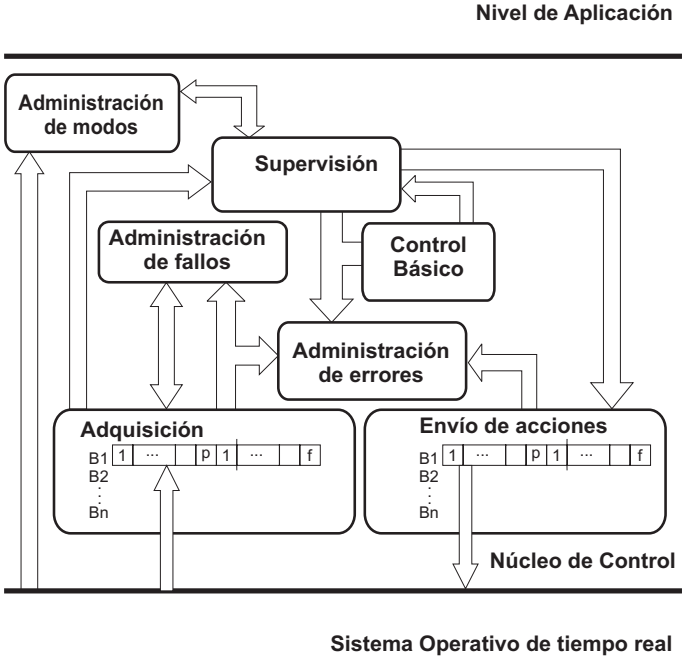

Figura 5. Servicios específicos propuestos para el núcleo de control

Servicios de adquisición y envío de la señal de control En la sección 3.2 se definieron las actividades **A1** y **A6** como propias del núcleo de control debido a que estas son comunes a cualquier lazo de control que se quiera implementar.

La manera en que estas actividades se ejecutarán en el núcleo de control está muy relacionada con una organización que garantice un período de muestreo real aproximado al período nominal $P_k \approx T$, según lo que se plantea en la proposición 1. En (Balbastre *et al.*, 2000), (Crespo A., 1999), (Crespo *et al.*, 2006) se desarrolla y demuestra una solución a los *problemas de tiempo*. Se plantea una política de planificación de tareas que divide el algoritmo de control digital en subtareas, donde cada lazo de control implementado en un SEC se debe dividir en:

- $\tau 1$:Tarea de Envío de la acción de control.
- $\tau 2$:Tarea de adquisición de datos.
- $\tau 3$:Tarea de cálculo de la acción de control.

El orden de prioridades es como se ha presentado siendo $\tau 1$ la más prioritaria . La secuencia lógica de ejecución de la aplicación de control indica que la primera tarea que debe realizarse es $\tau 2$. Sin embargo, la tarea de más prioridad es $\tau 1$. Por lo tanto, $\tau 1$ debe planificarse con un desplazamiento inicial con respecto a $\tau 2$ igual a la maxima latencia permitida por el proceso controlado. La tarea $\tau 3$ será ejecutada inmediatamente después que termine $\tau 2$. Por lo general las tareas $\tau 1$ y $\tau 2$ consumen muy poco tiempo de procesamiento, siendo $\tau 3$ la que

mayor potencia de cálculo demanda, sobre todo si es una ley de control compleja. Figura 6.

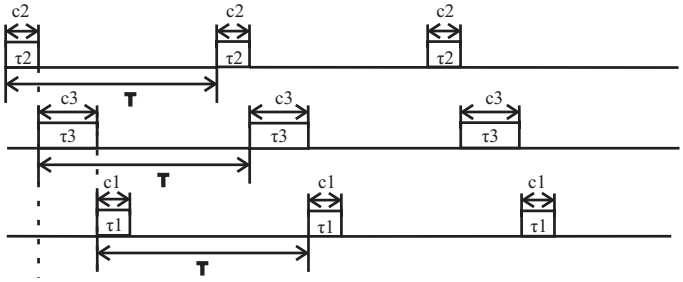

Figura 6. Subtareas de un lazo de control

La implementación de esta organización introduce un gran número de tareas, debido a que por cada lazo de control se deben planificar 3 tareas. Esto aumenta en gran medida la sobrecarga del SOTR, el uso de la memoria y el tiempo de ejecución de las tareas debido a los cambios de contexto que serán necesarios.

Es por ello que en la arquitectura del núcleo se proponen dos servicios independientes encargados de agrupar estas tareas:

- *Servicio de envío de la acción de control:* Agrupa en una tarea las actividades de envío de la acción de control correspondiente a todos los lazos de control dentro de un SEC. Este servicio mantiene un *buffer* con las n mediciones anteriores y realiza, en caso de retraso del canal de medición, una estimación basado en los valores pasados de f valores futuros. Este servicio actualiza los vectores B de las variables de entrada I_i y R_i en caso de ser referencias externas.
- *Servicio de adquisición de datos:* Agrupa en una tarea las actividades de adquisición correspondiente a todos los lazos de control dentro de un SEC. Este servicio mantiene un *buffer* con las n acciones anteriores y realiza, en caso de retraso del cálculo de la acción, una estimación basado en los valores pasados de f valores futuros. Este servicio actualiza los vectores B de las variables de salida O_i.

De forma básica la actividad de cada servicio es bastante parecida y viene dada por el siguiente algoritmo:

Datos y funciones
V_T: vector de períodos de cada lazo
V_A: vector de tiempo absoluto de activación
ti: tiempo absoluto inicial
$min\_v()$: devuelve el mínimo valor de un vector y los indices V_Indexs a los cuales pertenece:
$lenght\_v()$: cantidad de elementos de un vector:
$time\_to\_wakeup$: tiempo absoluto al cual debe realizarse la adquisición más próxima

```
// condiciones iniciales
.For i=0 to lenght_v(V_T)
.  V_A[i] = ti+V_T[i]
.End For
. While(1)
.   (time_to_wakeup, V_Indexs)=min_v(V_A)
.   For j=0 to lenght_v(V_Indexs)
.     V_A[V_Indexs[j]] = V_A[V_Indexs[j]+V_T[V_Indexs[j]]
.    End For
.   sleep_until(time_to_wakeup)
.   For j=0 to lenght_v(V_Indexs)
.     Get or Send data for each channel
.     Check status for each channel
.   End For
.End While
```

De acuerdo a la política de planificación por subtareas, el servicio de envío de la acción de control será el de mayor prioridad en el sistema, así como el de adquisición de datos será el que le sigue en prioridad.

Luego de cada adquisición de datos se efectúa un chequeo del estado de cada variable medida utilizando los servicios del administrador de fallos y actualizando la variable e correspondiente al canal de medición leído.

Servicios de supervisión y control Como se planteó en la sección 3.3 existen dos niveles de supervisión. El nivel general debe planificarse como una tarea independiente. En algunos modelos, dependiendo del SOTR, puede ser tomada como una tarea que se ejecuta cuando el procesador está desocupado. Este servicio es bastante dependiente del tipo de SOTR utilizado y en algunos casos también del hardware utilizado, pues es quién determinará como se están utilizando los recursos particulares del sistema.

El supervisor trabaja directamente sobre la estructura S_i que se define de la siguiente manera:

$$S_i = (\ L\ , Cu\ ,\ f\ )$$

donde L es el conjunto de estados que puede tomar el lazo de control i, debe contener al menos dos estados $\{normal\ y\ seguro\}$, y de acuerdo a la configuración particular podrán incluir $\{respaldo, reducido, periodo\}$. Cu contiene el estado actual, mientras que f contiene el valor de desempeño actual que posee el lazo control i.

De acuerdo a los recursos disponibles y el desempeño f de cada lazo de control este nivel solicita al supervisor de planta que se mantenga el estado actual o se pasa a otro.

Algunas variables de configuración deben ser establecidas para su funcionamiento, entre ellas los límites permisibles de desempeño de cada lazo.

En el caso del nivel de supervisión de planta estará conformado por una tarea por cada lazo de control que se activará inmediatamente después de la adquisición de los datos correspondientes y de acuerdo al estado en que se encuentre el control, decidirá que ley de control será utilizada. Dependiendo de la configuración que se haya definido se calculará la aplicación de usuario en su versión completa o reducida, el cálculo de leyes básicas, o de respaldo.

Este nivel es el encargado de ejecutar la máquina de estados que se definió en la sección 3.3, los cambios de estados están determinados por el estado del lazo de control, el estado de las variables de entrada, más el desempeño que tiene el sistema en un instante determinado.

4.5 Mecanismos de concurrencia utilizados en el núcleo de control

El mecanismo de concurrencia utilizado es por asignación de prioridades fijas. De acuerdo a los servicios que se han definido anteriormente ya ha quedado establecido lo siguiente:

- Las actividades de envío de la acción de control se agrupan para todos los sistemas o lazos de control ejecutados sobre el SEC en una tarea con máxima prioridad (1)
- Las actividades de adquisición se agrupan para todos los sistemas o lazos de control ejecutados sobre el SEC en una tarea con prioridad (2)

Esta propuesta no implica ningún análisis adicional con respecto al mecanismo de subtareas debido a que no tiene ninguna implicación en la planificación del sistema, es decir, los resultados son idénticos. También, garantiza que la utilización de los recursos de E/S sean controlados única y exclusivamente por dos tareas con lo cual no es necesario compartir recursos de hardware entre más tareas. La manipulación de la información que se pasan entre las tareas es llevada a cabo utilizando secciones críticas para evitar la inconsistencia de los datos asociados a la lectura de los canales de comunicación y el envío de la acción de control.

Las restantes actividades esenciales del núcleo quedan de la siguiente forma:

- La supervisión a nivel de sistema se ejecutan en tiempo ocioso del procesador (Idle).
- Las supervisión a nivel de lazo con prioridad (3). Siempre activada luego de realizar la adquisición de datos correspondiente al lazo de control actual.
- La ejecución de los algoritmos de control a nivel de usuario con prioridad mayor o igual que (4).

La elección de prioridades para los algoritmos de control debe ser propuesta por el diseñador siendo recomendable asignar las mismas según el período de muestreo de los controladores, tal y como se plantea en (Liu and Layland, 1973). Este parámetro es almacenado en la variable pr_i al registrar la función de nivel de usuario.

Debido a retardos, la actividad de envío de la acción de control se podría ejecutar sin que el algoritmo de control de nivel de usuario finalice; entonces, una acción de respaldo debe ser enviada a la planta. Si para este lazo no se ha registrado esta política, entonces se enviará la anterior y si el desempeño supera los límites establecidos, se enviará la acción segura.

5. EJEMPLOS

En los ejemplos que se presentan se usa una ley de control que proporciona acciones de control respaldo. Por tanto, los estados posibles serán: Control Complejo y Acción de respaldo.

Es válido señalar que estos ejemplos son solo dos entre tantos otros posibles que pudieran ser empleados. No es el objetivo de este trabajo abordar todas las opciones posibles, sino plantear algunas soluciones que ilustren la manera de implementar una aplicación de control para un SEC que se ejecute utilizando el núcleo de control.

El núcleo de control que se ha utilizado en estos ejemplos esta dado en dos versiones

1. Una simulación basada en el sistema operativo Truetime (Henriksson *et al.*, 2002). Compuesto por un conjunto de funciones que hacen uso del API de Truetime para realizar la gestión de tareas y la comunicación con los dispositivos externos.
2. El *Firmware* de *RobotC* (*ROBOTC*, n.d.), que es un SOTR que se ejecuta sobre los robots LEGO (*LEGO Mindstorms*, n.d.).

El planificador utilizado, en ambos caso, es el de prioridades fijas (FP), que nos permite hacer la distribución de tareas que se ha descrito en la sección 4. Entre las funciones básicas de simulación que deben ser realizadas están:

- Registro de canales de entrada y salida: se especifican aquellos que serán referencia o mediciones de proceso.
- Registro de los controladores: se incorporan la función de usuario y los controladores básicos y la configuración del nivel de supervision de planta.
- Condiciones de simulación: se especifican los parámetros temporales de ejecución tales como tiempos de cómputo, latencia y *jitter*; así como los períodos de muestreo de los controladores y su prioridad respecto a otros que se registren.

5.1 Ejemplo 1. Control por bloque multifrecuencia

En el trabajo(Albertos, 1990) se presenta un nuevo modelo para el diseño de sistemas de control muestreado. Su principal contribución es la obtención de un control que es capaz de muestrear y enviar acciones de control a diferentes períodos de muestreo.

Utilizando el modelo del sistema se pueden calcular n acciones de control con un intervalo de $To = nT$ a partir de datos tomados del proceso a un período T. A To se le denomina: período envolvente.

Para diseñar un controlador multifrecuencia representado en espacio de estados es necesario aplicar el teorema 1 desarrollado en (Albertos, 1990).

Teorema 1. Considere el sistema:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{1}$$

Donde A,B y C son las matrices de la ecuación estado, $x(t)$ es el vector de estado, $y(t)$ la salida y $u(t)$ el vector del control. Si el sistema es controlable, observable y sin perturbaciones, y se escoge un período de muestreo T tal que el sistema discreto que resulta es también controlable y observable, entonces existe un controlador multifrecuencia discreto con período $T_o = NT$ ($N \geq n$, siendo n el orden del sistema), tal que el modelo discretizado a T_o tiene cualquier estructura deseada $\bar{A}$ y el controlador discreto por realimentación de estado está dado por la ecuación 2. En el caso de que $N = n$, el controlador es único.

$$U_k = W^{\#}(\bar{A}^N - A^N)x_k \tag{2}$$

$W^{\#}$ es la seudoinversa de la matriz de controlabilidad.

Además, si ningún estado es medible, el controlador puede ser implementado como:

$$U_k = MR + M_1 Y_{k-1} + M_2 U_{k-1} \tag{3}$$

Donde R es la referencia que se desea alcanzar y M una matriz de ajuste. Las matrices M_1 y M_2 se obtienen de:

$$M_1 = W^{\#}(\bar{A} - A^N)A^N O^{\#} \quad (4)$$
$$M_2 = W^{\#}(\bar{A} - A^N)(W - A^N O^{\#} H)$$

Donde $O \triangleq [A^T c (A^2)^T c, \ldots, (A^N)^T c] \in \mathbb{R}^{N \times n}$ es la matriz de observabilidad multiplicada por A. En las ecuaciones 2 y 3 U_k e Y_k son vectores que se definen según las ecuaciones 5.

$$Y_k \triangleq Y(kT_o) = [y(kT_o + T), \ldots, y(kT_o + NT)]^T \in \mathbb{R}^N$$
$$U_k \triangleq U(kT_o) = [u(kT_o), u(kT_o + T), \ldots, \quad (5)$$
$$u(kT_o + NT - T)]^T \in \mathbb{R}^N$$

La estrategia que se plantea para formar parte del núcleo de control, difiere en alguna medida de la propuesta dada en (Albertos, 1990). Utilizando las ecuaciones 2 o 3, se podrá construir una version de la función de nivel de usuario donde el período de muestreo será T y en cada intervalo se obtendrá un vector de n acciones futuras, similar al horizonte deslizante del control predictivo. Si se utilizan apropiadamente las ecuaciones del controlador, el cálculo se puede dividir en dos partes. La primera parte se encargará de obtener el primer valor del vector de acciones $u(kT_o)$ cuyo resultado será aplicado de inmediato a la planta. Mientras que la segunda parte se encargará de obtener los restantes elementos del vector que serán utilizados como acciones de respaldo ante situaciones de emergencia o restricción de recursos.

Esta estrategia incrementa en alguna medida el tiempo de cómputo de la acción de control, debido a que esta debe obtener el vector de acciones futuras. Sin embargo, debe señalarse que la segunda parte del algoritmo no tiene porque ser calculada continuamente, lo cual convierte al controlador en uno convencional por realimentación del estado. Además, el algoritmo es más robusto debido a que si no existen los recursos suficientes, entonces se puede enviar a la planta una señal de respaldo que está basada en la dinámica del proceso. El ejemplo que se muestra en la siguiente sección brinda algunos resultados al respecto.

Diseño del controlador El proceso a controlar es un motor de corriente directa cuya función transferencial es:

$$G(s) = \frac{1000}{s(s+1)} \quad (6)$$

En espacio de estados discreto el sistema está representado por las siguientes matrices, el período de muestreo es de $T = 0{,}005$ segundos:

$$A = \begin{bmatrix} 0{,}9950 & 0 \\ 0{,}0050 & 1 \end{bmatrix} \quad (7)$$
$$B = \begin{bmatrix} 0{,}0050 \\ 0 \end{bmatrix}$$
$$C = [\,0 \;\; 1\,]$$

El intervalo de tiempo o período envolvente es $T_o = 10T$, lo cual implicará un horizonte de acciones de control de $N = 10$. La respuesta dinámica del sistema que se quiere lograr está dada por un coeficiente de amortiguamiento $\xi = 0{,}7$ y un tiempo de establecimiento de $ts = 0{,}05s$. A partir de este criterio de diseño se obtiene fácilmente la estructura deseada en forma de modelo en espacio de estado:

$$\bar{A} = \begin{bmatrix} 0{,}4600 & -29{,}4146 \\ 0{,}0036 & 0{,}9264 \end{bmatrix} \quad (8)$$

que luego se utilizará en las expresiones 4 para obtener los valores de $M_1, M_2 \in \mathbb{R}^{10 \times 10}$. La matriz M es una matriz diagonal que se calcula a partir de una formula recursiva según el método de ajuste de ganancia. Para ello debemos obtener el primer valor de la diagonal utilizando el teorema del valor final:

(9)

Dónde f_1 es el valor de ajuste y $R(k)$ es la referencia. Al sustituir los valores del ejemplo se obtiene $f_1 = 5{,}8976$.

$$M = \begin{bmatrix} f_1 & & \\ & \ddots & \\ & & f_{10} \end{bmatrix} \quad (10)$$

La formula recursiva está dada por 11, siendo k_1 el valor de la ganancia de realimentación a partir de la cual se obtiene la estructura deseada $\bar{A}$. Las variables f_i, k_i se inicializan con f_1, k_1 respectivamente:

$$f_i = f_{i-1} - k_{i-1} B f_1, \; 2 \leq i \leq 10 \in \mathbb{N} \quad (11)$$
$$k_i = k_{i-1} A - k_{i-1} B k_1, \; 2 \leq i \leq 10 \in \mathbb{N}$$

.

Condiciones de simulación y resultados Para estudiar el comportamiento del sistema se proponen la ejecución del controlador por bloques multifrecuencias (CBMF) obtenido en la sección anterior contra el siguiente:

$$u(k) = -k_1 [x_1(k) x_2(k)]^T + f_1 r(k); \quad (12)$$

Que es un controlador por realimentación del estado (CRE) equivalente al controlador obtenido si tomáramos un de horizonte $N = 1$, bajo la estrategia de bloques multifrecuencia.

Para el segundo caso se han programado dos tareas, una donde se realizan la adquisición, cálculo de la ley de control y envío de la acción de control con la menor prioridad y una tarea de mayor prioridad que es ejecutada también cada $0{,}005$ segundos pero con un tiempo de cómputo aleatorio entre 0.001 y 0.008, lo cual permite ver la influencia en el desempeño del control tanto de la variación de la latencia como del incumplimiento de los períodos de muestreo. Los resultados se muestran en la figura 7

El siguiente escenario muestra la ejecución de la estrategia de CBMF utilizando el núcleo de control. Además, se ha incorporado la misma tarea anterior con una prioridad mayor que la de la tarea de control y menor que las de adquisición y envío de la acción de control. En este caso la respuesta está dada según la figura 8.

El algoritmo de control situado a nivel de aplicación, por encima del núcleo de control, se puede descomponer en dos subtareas independientes:

- *Cálculo de la acción actual o control complejo*: Se obtiene de la expresión $m * f1 + m_1 Y_{k-1} + m_2 U_{k-1}$, donde m, m_1, m_2, son la primera fila de M, M_1, M_2 respectivamente.

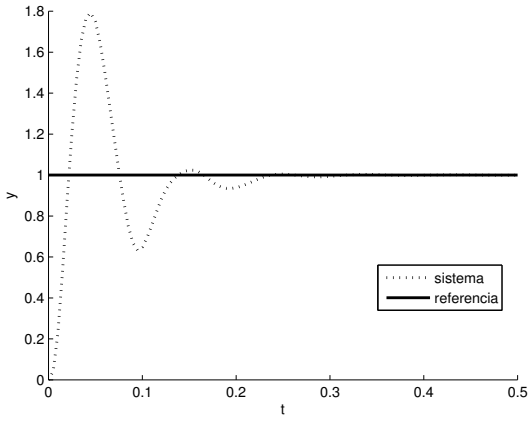

Figura 7. Respuesta del sistema con latencia variable y sin acción del núcleo de control

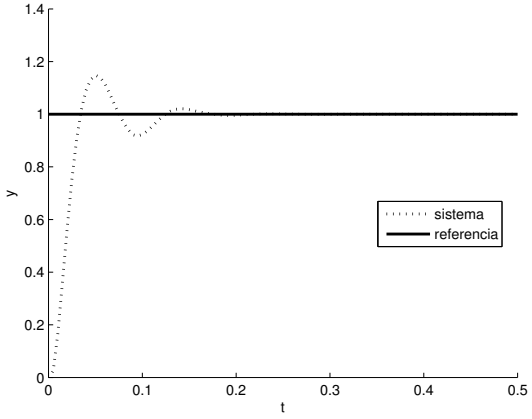

Figura 8. Respuesta del sistema con latencia variable y acción del núcleo de control

- *Actualización del vector de acciones futuras o calculo de acciones de respaldo*: A partir del cálculo del resto de filas correspondientes a M, M_1, M_2 que proporcionan el vector de acciones de control futuras.

Como se puede apreciar el utilizar los servicios del núcleo de control permite obtener mucho mejores resultados en la respuesta transitoria del sistema en cuanto al máximo sobreimpulso y el tiempo de establecimiento.

5.2 Ejemplo 2. Control Predictivo de Trayectoria

El desarrollo de este ejemplo se realiza, tanto sobre TrueTime, como sobre un robot LEGO. Las características de este robot son muy limitadas, tanto en su capacidad de memoria como en los recursos computacionales, teniendo un procesador cuya velocidad es de $48Mhz$.

Diseño del controlador El esquema de control propuesto será similar al que se brinda en (Klancar and Skrjanc, 2007) pero situado dentro del LEGO. La figura 9 muestra un control en cascada, donde existen tres lazos. Dos de ellos utilizan controles PID clásicos para mantener las velocidades de las ruedas izquierda y derecha, mientras que el tercero proporciona la referencias de dichas velocidades a través del algoritmo de control predictivo de trayectoria.

Figura 9. Esquema de control de trayectoria

El Bloque definido como control Predictivo se compone de dos partes denominadas control en adelanto U_f y control de realimentación U_b. La figura 10 muestra los detalles. El control en adelanto tiene en cuenta solamente la referencia, mientras que el control predictivo se encargará de *predecir* el comportamiento del error de seguimiento de la trayectoria de referencia. Para ello se ha linealizado el modelo cinemático del robot con respecto a la referencia.

Figura 10. Control predictivo

La implementación práctica de un control con estas características debe tener en cuenta que el cálculo de la señal de control se obtiene:

$$U = U_b + U_f$$
$$U_f = [\, V_r \;\; W_r \,]^T$$
$$U_b = (G^T \overline{Q} G + \overline{R})^{-1} G^T \overline{Q} (F_r - F) e(k)$$

dónde:

- U es un vector que contendrá las velocidades R_i, R_d de referencia para los controles PID de velocidad de cada motor.
- V_r es la velocidad lineal de referencia.
- W_r es la velocidad angular de referencia.
- G, F y F_r son la matrices de predicción obtenidas del modelo.
- $\overline{Q}$ y $\overline{R}$ son las matrices ampliadas de pesos.

El cálculo de U_f es relativamente simple y además puede realizarse de manera adelantada según se plantea en (Astrom and Wittenmark, 1995),(Cervin, 2003), ya que no incluyen valores obtenidos de la medición en tiempo real. Asimismo, a la operación $M = (G^T \overline{Q} G + \overline{R})^{-1} G^T \overline{Q} (F_r - Fr)$ también le ocurre otro tanto, siendo esta la que ocupa la mayor cantidad de tiempo de procesamiento debido al cálculo de la matriz inversa y dependiendo del horizonte de predicción H que se utilice.

En esta aplicación existen 3 lazos de control cuyos datos representan los siguientes sistemas a controlar:

1. D_1: Control PID de la velocidad de la rueda derecha V_d.
2. D_2: Control PID de la velocidad de la rueda izquierda V_i.
3. D_3: Control predictivo de la trayectoria

Los lazos D_1 y D_2 se ejecutarán siempre a bajo nivel, debido a la simplicidad de sus controladores.

Para el caso D_3 es necesario ejecutar una función de usuario cuyas políticas registradas serán las siguientes:

1. Control Normal: Obtiene el resultado $U(k)$ para el instante de tiempo discreto actual k
2. Acciones de respaldo: Obtiene el resultado M a partir del cual se puede obtener los valores $U_f(k+l)$ y $U_b(k+l)$ para cualquier valor futuro l que pertenezca al horizonte de predicción.

Condiciones de simulación y Resultados El experimento que se ha simulado en TrueTime utiliza como trayectoria las siguientes ecuaciones que definen los pares $[x_r, y_r]$ en el plano.

$$x_r = 1{,}1 + 0{,}7 * sin(2 * \pi * \frac{t}{30})$$
$$y_r = 0{,}9 + 0{,}7 * sin(4 * \pi * \frac{t}{30})$$

Además, se ha tenido en cuenta los límites de velocidad y aceleración, angular y tangencial permitidas por el robot, tomando las misma restricciones que se plantean en (Klancar and Skrjanc, 2007). Para el calculo de la matriz inversa se ha utilizado el algoritmo de Cholesky (Montenegro and Díaz, 1987), que garantiza el menor tiempo de cómputo.

El tiempo de muestreo T_3 del control predictivo se ha subido hasta $0{,}1$ segundos, mientras que Q ahora es:

$$Q = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0{,}1 \end{bmatrix}$$

Además, se ha simulado una perturbación que ocurre para $t = 26$ segundos. La perturbación es un paso escalón que afecta la posición del robot en $y = 0{,}05$ metros, similar a que el robot reciba un impacto que afecte la trayectoria.

La figura 11 representa la trayectoria seguida por el robot utilizando el algoritmo de alto nivel indicado anteriormente. Las coordenadas iniciales del robot son $x_i = 1{,}1$ e $y_i = 0{,}7$, que indica un error inicial entre la trayectoria de referencia y la posición. En este resultado no se ha introducido ninguna tarea adicional que afecte el período de muestreo T_3 del algoritmo predictivo.

Si se introduce una tarea adicional que desplaza el período de muestreo, entonces habrá un deterioro del comportamiento del sistema, que se traduce en un mal seguimiento de la trayectoria. En este caso el segmento es periódico con un tiempo de cómputo de $0{,}2$ segundos y período $0{,}3$ seg. La figura 12 representa la trayectoria seguida por el robot bajo estas nuevas condiciones sin aplicar las alternativas del **NC**.

Al utilizar como alternativa de trabajo del *núcleo de control* la estimación de valores utilizando el cálculo del algoritmo de predicción para instantes futuros, se pudo comprobar una mejoría en el desempeño del control. Los resultados se muestran en la figura 13.

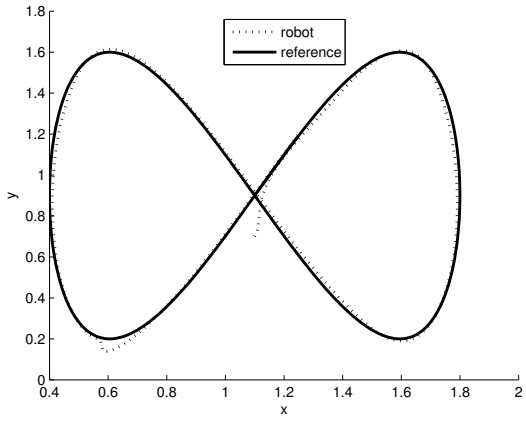

Figura 11. Trayectoria sin afectaciones de tiempo

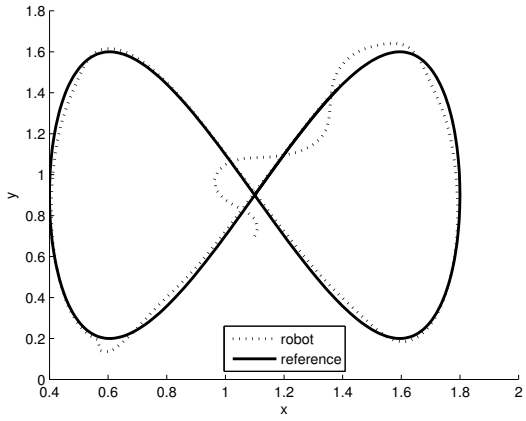

Figura 12. Trayectoria con afectaciones de tiempo

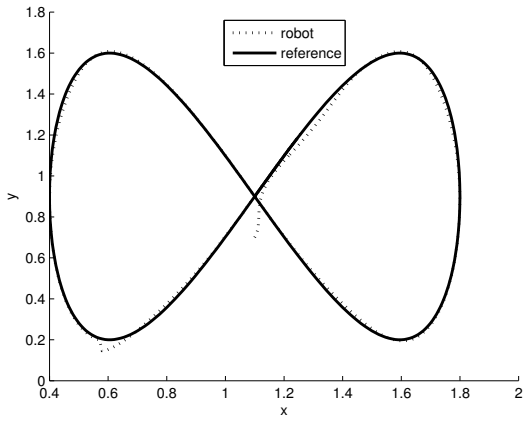

Figura 13. Trayectoria con afectaciones de tiempo y acción del *núcleo de control*

Esta situación también puede darse si es necesario reducir el tiempo de cómputo, y se decide prescindir del segmento de cálculo de la ley de control en dos de cada tres intervalos. De esta forma se reduce el tiempo de cómputo total tc en $2 * t_s - t_m$, donde:

- t_s: tiempo de cómputo del segmento $\tilde{s_3^{ud}}$

- t_m: tiempo de cómputo del segmento m^{ev} del *núcleo de control* encargada de ejecutar la alternativa trazada.

Las condiciones iniciales del sistema son muy importantes, ya que para obtener las matrices G y F, se hace una linealización del modelo cinématico del robot alrededor de la trayectoria de referencia (Klancar and Skrjanc, 2007). Ello implica que para condiciones iniciales alejadas de la trayectoria habrá un peor desempeño del control, ya que el error del modelo utilizado se incrementa. Sin embargo, bajo estas condiciones también el *núcleo de control* aporta una considerable mejoría en el seguimiento de la trayectoria. Las figuras 14, 15, y 16 muestran las trayectorias bajo condiciones de trabajo sin retardo, con retardo y utilizando el *núcleo de control* respectivamente para condiciones iniciales de $x_i = 1{,}1$ e $y_i = 0{,}5$.

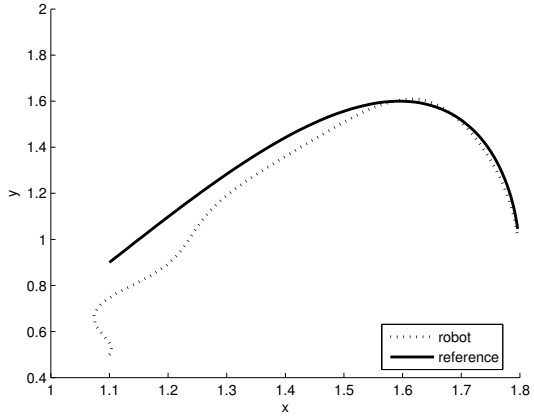

Figura 14. Trayectoria sin afectaciones de tiempo

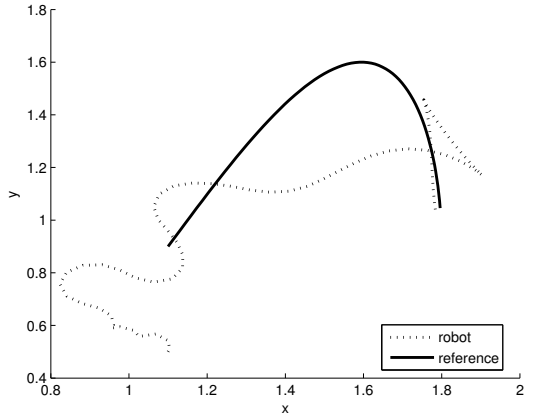

Figura 15. Trayectoria con afectaciones de tiempo

Como se aprecia en la figura 16 la presencia del núcleo logra mantener el desempeño del sistema de manera satisfactoria.

Comportamiento real utilizando el LEGO El experimento utiliza como trayectoria las siguientes ecuaciones que definen los pares $[x_r, y_r]$ en el plano.

$$x_r = 0{,}2 + 0{,}2 * cos(2 * \pi * \frac{t}{30})$$
$$y_r = 0{,}2 + 0{,}2 * sin(2 * \pi * \frac{t}{30})$$

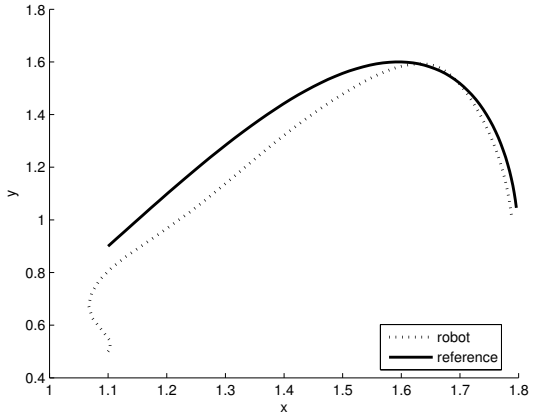

Figura 16. Trayectoria con afectaciones de tiempo y acción del núcleo de control

El motivo del cambio de trayectoria es debido a la poca precisión de los sensores de posición con que se cuenta. En este caso tomados a partir de las velocidades de las ruedas izquierda y derecha del LEGO.

Los lazos de control que se han configurado son similares a los propuestos en la simulación, sin embargo, el tiempo de muestreo T_3 del control predictivo se ha establecido en $0{,}15$ segundos, mientras que Q ahora es:

$$Q = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 60 & 0 \\ 0 & 0 & 0{,}13 \end{bmatrix}$$

Las coordenadas iniciales son $x_i = 0{,}5$ e $y_i = 0{,}2$ que igualmente indican un error inicial entre la posición del robot y la trayectoria a seguir. La figura 17 muestra la trayectoria real del robot sin que existan afectaciones de tiempo en la ejecución de los lazos de control.

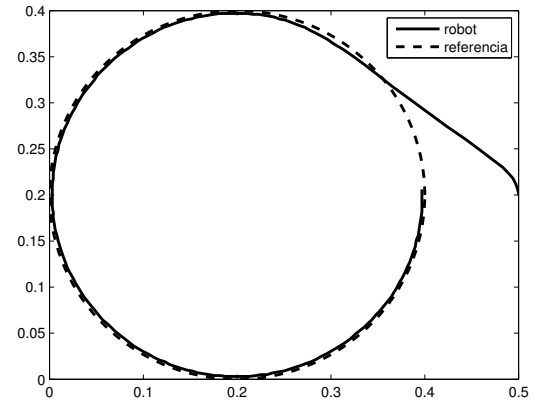

Figura 17. Trayectoria sin afectaciones de tiempo

Si se introduce una tarea que desplaza el periodo de muestreo habría un deterioro del comportamiento del sistema, que se traduce en un mal seguimiento de la trayectoria. En este caso, la tarea es periódica con un tiempo de cómputo de 0,2 segundos y periodo 0,3 segundos. La figura 18 representa la trayectoria seguida por el robot bajo estas nuevas condiciones sin aplicar las alternativas del NC.

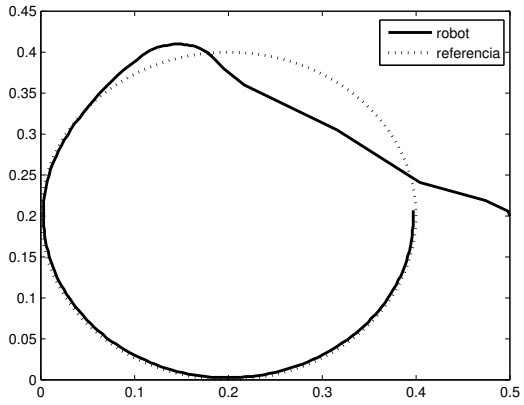

Figura 18. Trayectoria con afectaciones de tiempo

Al utilizar el núcleo de control bajo las mismas condiciones de afectaciones de tiempo debido a la planificación de otras tareas adicionales, se puede comprobar una mejoría en el seguimiento de la trayectoria. En este caso también se han enviado acciones de control de respaldo utilizando los cálculos del algoritmo predictivo. La figura 19 muestra los resultados utilizando esta estrategia.

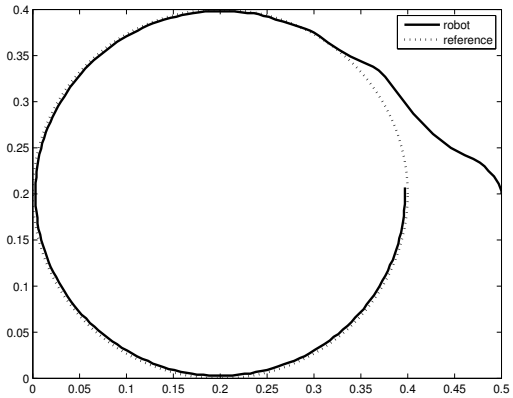

Figura 19. Trayectoria con afectaciones de tiempo utilizando el núcleo de control

6. CONCLUSIONES Y RECOMENDACIONES

El presente trabajo abordó el problema del diseño de sistemas de control empotrados bajo condiciones de escasos recursos. La solución que se propone está dada en función de la implementación de controladores digitales, su planificación en tiempo real y está basada en el concepto de núcleo de control. Los resultados principales son los siguientes:

- Definición del concepto de núcleo de control (NC), sus características y objetivos para un sistema monoprocesador.
- Desarrollo de una arquitectura de software basada en el NC que permite la ejecución y supervisión de uno o varios lazos de control incluyendo:
  - La definición de las capas que componen un sistema de control empotrado.
  - La definición de varios servicios encargados de ejecutar las actividades del control: adquisición, calculo de la ley de control, envío de la acción de control
- El estudio de un grupo de estrategias de control de bajo coste computacional que se ponen en práctica en situaciones de recursos limitados.

La flexibilidad de la arquitectura permite escoger las estrategias de control en dependencia del tipo de sistema controlado y las limitaciones que se presenten. Entre las propuestas se encuentra el intercambio de controladores en línea ya sea por reducción del controlador o por cambio de períodos de muestreo. También se abordan técnicas de ahorro basadas en el control multifrecuencia. Aunque algunas de ellas requieren procesamiento de cálculo adicional, este es realizado cuando hay suficiente disponibilidad, y los valores obtenidos, son reutilizados en situaciones de escasos recursos, donde probablemente la respuesta del sistema se vería afectada. En el trabajo fueron presentados dos ejemplos que ilustran el uso de una de las estrategias propuestas.

Como continuidad de estos trabajos se propone:

- Simular y probar otras estrategias soportadas por la arquitectura propuesta.
- Desarrollar e implementar un grupo de funciones capaces de detectar fallos en la adquisición de los datos. Estas deben consumir el menor tiempo de calculo posible.
- Estudiar y comprobar varios métodos para calcular los índices de desempeño del sistema que sean apropiados para el desarrollo tanto de funciones de nivel de usuario como internas del núcleo de control.

Finalmente es necesario señalar que todo el análisis realizado en este trabajo se se ha centrado en el caso de un sistema monoprocesador. El caso de aplicaciones distribuidas será considerado en próximos desarrollos basados en la arquitectura propuesta, donde se incluyan varios nodos de procesamiento conectados por un canal de comunicación.

AGRADECIMIENTOS

Este trabajo fue soportado en parte por el proyecto CYTED *Diseño y desarrollo de aplicaciones basadas en redes de sensores* D2ARS . Los autores además desean expresar su más profundo reconocimiento al Dr. Pedro Albertos Pérez y al Dr. Alfons Crespo Llorente de la UPV por todas las recomendaciones y apoyo brindados durante el desarrollo de este trabajo.

REFERENCIAS

Albertos, P., A. Crespo and J. Simó (2006). Control kernel: a key concept in embedded control systems. In: *IFAC Conf. on Mechatronics*. Heilderberg.

Albertos, P., A. Crespo, M. Vallés and I. Ripoll (2005*a*). Embedded control systems: some issues and solutions. In: *16th IFAC World Congress*. Praga, República Checa.

Albertos, P., M. Vallés, A. Cuenca and A. Valera (2007). Essential control in embedded control systems. In: *IFAC Symp. On Cost Oriented Automation*. La Habana, Cuba.

Albertos, P., M. Vallés and A. Valera (2005*b*). Control performances under logic changes in the operational conditions. *Journal of Computer and Systems Sciences International* **44**(11), 587–593.

Albertos, Pedro. (1990). Block multirate input-output model for sample-data control systems. *IEEE Transactions Automatic Control* **35**(9), 1085–1088.

Albertos, Pedro, Manuel Olivares and Mario E. Salgado (2000). Trade-off between time delays and control effort. In: *IFAC Workshop in Advances in Control Education*. Gold Coast, Australia.

ARTIST2 (2005). *Roadmap on Control of Real-Time Computing Systems*.

Årzén, Karl-Erik and Anton Cervin (2005). Control and embedded computing: Survey of research directions. In: *Proc. 16th IFAC World Congress*. Prague, Czech Republic.

Årzén, Karl-Erik, Anton Cervin, Johan Eker and Lui Sha (2000). An introduction to control and scheduling co-design. In: *Proceedings of the 39th IEEE Conference on Decision and Control*. Sydney, Australia.

Årzén, Karl-Erik, Bo Bernhardsson, Johan Eker, Anton Cervin, Klas Nilsson, Patrik Persson and Lui Sha (1999). Integrated control and scheduling. Technical Report ISRN LUTFD2/TFRT--7586--SE. Department of Automatic Control, Lund Institute of Technology, Sweden.

Astrom, K. J. and B. Wittenmark (1995). *Adaptive Control*. Addison Wesley Longman.

Balbastre, P., I. Ripoll and A. Crespo (2000). Control tasks delay reduction under static and dynamic scheduling policies. *Real-Time Computing Systems and Applications, International Workshop on* **0**, 522.

Balbastre, Patricia (2002). Modelo de tareas para la integración del control y la planificación en sistemas de tiempo real. PhD thesis. Departamento de Informática de Sistemas y Computadores, Universidad Politécnica de Valencia. Valencia, España.

Cervin, A. (2003). Integrated Control and Real-Time Scheduling. PhD thesis. Departament of Automatic Control, Lund Institute of Technology. Lund, Sweden.

Cervin, A., J. Ecker, B. Bernhardsson and K.E. Arzén (2002). Feedback feedforward scheduling of control tasks. *Real Time Systems* **23**(6), 25–53.

Cervin, Anton, Bo Lincoln, Johan Eker, Karl-Erik Årzén and Giorgio Buttazzo (2004). The jitter margin and its application in the design of real-time control systems. In: *Proceedings of the 10th International Conference on Real-Time and Embedded Computing Systems and Applications*. Göteborg, Sweden. Best paper award.

Cervin, Anton, Dan Henriksson, Bo Lincoln, Johan Eker and Karl-Erik Årzén (2003). How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime. *IEEE Control Systems Magazine* **23**(3), 16–30.

Crespo, A., P. Albertos, M. Vallés, M. Luesma and J. Simó (2006). Schedulability issues in complex embedded control systems. In: *Computer-Aided Control Systems Designand 2006 IEEE International Symposium*. Munich, Germany.

Crespo A., Ripoll Ismael, Albertos P. (1999). Reducing delays in rt control: the control action interval. In: *14th IFAC World Congress on Automatic Contro*. Elsevier Science. p. 0.

Fernández, A., P. Albertos, M. Vallés and O. LLanes (2009). Estructuras de control en sistemas empotrados. In: *IX Simposio Internacional de Automatización, 9th International Symposium on Automation, Informática 2009*. Cuidad Habana, Cuba.

Henriksson, Dan, Anton Cervin and Karl-Erik Årzén (2002). TrueTime: Simulation of control loops under shared computer resources. In: *Proceedings of the 15th IFAC World Congress on Automatic Control*. Barcelona, Spain.

Kao, Chung-Yao and Bo Lincoln (2004). Simple stability criteria for systems with time-varying delays. *Automatica* **40**(8), 1429–1434.

Klancar, G. and I. Skrjanc (2007). Tracking-error model-based predictive control for mobile robots in real time. *Robotics and Autonomous Systems* **55**, 460–469.

LEGO Mindstorms

LEGO Mindstorms *(n.d.). Disponible en: http://mindstorms.lego.com/.*

Liu, C. L. and James W. Layland (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. JACM **20***(1), 46–61.*

Montenegro, Arnaldo Gómez and Lilliam Alvarez Díaz (1987). Métodos numéricos del algebra lineal.

ROBOTC

ROBOTC (n.d.). Disponible en www.robotc.net.

Wittenmark, Björn, Johan Nilsson and Martin Törngren (1995). Timing problems in real-time control systems. In: *Proceedings of the 1995 American Control Conference*. Seattle, Washington.

Xia, Feng, Yu-Chu Tian, Youxian Sun and Jinxiang Dong (2008). Neural feedback scheduling of real-time control tasks. *CoRR*.