Conceptos de Tiempo Real Aplicados a la Informática Industrial

Leonardo Moreno [1,2], Christian Geymonat [2], José M. Urriza [1]

[1]Universidad Nacional de la Patagonia San Juan Bosco, [2]ALUAR S.A.I.C.
e-mail: oelamon@hotmail.com, c_geymonat@hotmail.com, josemurriza@unp.edu.ar

Resumen. Los procesos industriales están gobernados en su gran mayoría, por controladores lógicos programables. En este trabajo se presenta un análisis entre la arquitectura típica de procesamiento en los controladores industriales y las soluciones de control utilizando herramientas de tiempo real. Posteriormente, tomando de referencia una plataforma basada en un *PLC* Siemens, se propone una solución para un problema real de control industrial. Se demostrará que si en la primeras etapas de diseño, se aplican herramientas actuales de tiempo real se simplifican las soluciones y a su vez se aporta robustez y eficiencia al sistema.

Palabras Claves: *RTS, PLC, RTOS, HMI, SCADA*

1 Introducción

Los Controladores Lógicos Programables (*PLC*), en las últimas décadas, se han vuelto primordiales en los ambientes industriales. Sin embargo, las exigencias de las industrias y sus aplicaciones, año a año requieren de mayores características como: capacidad de memoria, nuevas herramientas de software, interfaces gráficas más avanzadas, posibilidades de comunicación y publicación de datos, etc.

Por otro lado, son necesarias herramientas de programación que permitan a los usuarios del *PLC*, poder inspeccionar y controlar los procesos que poseen constricciones temporales y poder planificar las concurrencias de las tareas de manera que cumplan esas constricciones.

Además, en los últimos años se observa que ha evolucionado la conexión del entorno de automatización con los niveles superiores de administración. Esto se realiza a través de una creciente tendencia tecnológica a la unificación de redes en la planta industrial. De esta manera, se proporciona información en "*tiempo real*" en los niveles correctos de decisión.

1.1 Planificación de Tareas en un *PLC*

Los *PLC* que gobiernan los procesos industriales, poseen una determinada cantidad de tareas que administran cada uno de los subprocesos definidos por los usuarios.

La forma normal de organizar estas tareas es mediante un ejecutivo cíclico. Este es simplemente una lista fija de tareas a ejecutar. Cuando la lista termina, comienza nuevamente desde el principio. A continuación se presentan las fases del ejecutivo cíclico de un *PLC* Siemens ([1, 2]) en un ciclo de ejecución:

1. El sistema operativo (*SO*) del *PLC* inicia el monitoreo de tiempo de ciclo.
2. El *SO* escribe los valores de la tabla imagen del proceso de las salidas en los módulos de salida.
3. El *SO* lee el estado de las entradas desde los módulos y actualiza la tabla imagen del proceso de las entradas.
4. El *SO* ejecuta solo la lista de tareas que han sido programadas.
5. Al final del ciclo el *SO* ejecuta cualquier tarea especial pendiente.
6. El *SO* retorna al inicio de ciclo.

Se hace notar que, por lo general, durante la ejecución del programa el *SO* no accede directamente a las señales de campo (Entradas/Salidas), sino que interactuar con las tablas de imagen del proceso de Entrada/Salida. Los datos de las entradas, solo están disponible sin pasar por las tablas, cuando se utilizan módulos de entrada-salida especiales.

1.1.1 Bloques de Organización de Interrupción

Dentro de los *PLC* Siemens, existen los denominados *Bloques de Organización de interrupción* (*OB*). Los mismos son las tareas del ejecutivo cíclico. Dependiendo del modelo del microprocesador utilizado, encontramos distinta cantidad de estos *OB*. En las versiones de microprocesadores más simples, los tiempos de ejecución y las prioridades pueden ser fijos y hasta pude solo tener un *OB*.

En un *PLC* Siemens ([2]), el *SO* ejecuta como tarea estándar el *Bloque de Organización* 1 (*OB1*). El *SO* del *PLC*, posee una limitación y es que no puede determinar de antemano cuál es el tiempo de ejecución de los *OB*. Es claro que para muchos procesos de *TR*, esto puede ser una importante restricción a la hora de planificarlos. Sin embargo, existen herramientas de programación que subsanan esta limitación, las cuales se presentará en el siguiente punto.

En los *PLC* Siemens, es posible programar un OB con una determinada prioridad de ejecución y un periodo de ejecución. Consecuentemente, una instancia de un *OB* programado con una determinada prioridad, interrumpirá cada un estipulado período.

1.2 Tiempo de Ejecución de Ciclo

El tiempo de ejecución de ciclo, es el tiempo requerido por el *SO* para ejecutar: la lista de tareas cíclicas, todas las interrupciones del programa (Ejemplo: Los *OB* de interrupción) y las actividades propias del sistema.

El tiempo de ciclo, es monitoreado por el *SO,* ya que el tiempo de ciclo de ejecución (T_C) no es el mismo en cada ciclo. Por tal razón, el *SO* provee de algunas variables de control sobre el mismo. A continuación se detallan:

- Tiempo de ejecución del ciclo (T_C): Es el tiempo que tarda la actualización de las tablas, más la ejecución del *OB1*. Puede ocurrir que este tiempo se incremente, debido a la instanciación de una tarea con mayor prioridad.
- Tiempo de ejecución mínimo (T_{MIN}): Es el tiempo mínimo que debe durar el ciclo. Este tiempo es útil para asegurar un intervalo de ejecución del *OB1* o para evitar la actualización innecesaria de las tablas imagen de las entradas/salidas si el tiempo de ejecución un muy corto.
- Tiempo de ejecución máximo (T_{MAX}): Máxima duración de ciclo permitida. Si el tiempo de ejecución se excede del T_{MAX}, se podrá programar un bloque *OB* para capturar este error.
- Tiempo de espera (T_{WAIT}): Es el tiempo entre T_C y el T_{MIN} en caso de que el T_C sea menor del T_{MIN}. Dentro de este tiempo pueden aparecer tareas por interrupción o simplemente estar en reposo la *CPU* a la espera de que se cumpla el T_{MIN}.

En la Figura 1 se presenta un ejemplo de ciclo de ejecución, en donde se puede observar los distintos tiempos definidos.

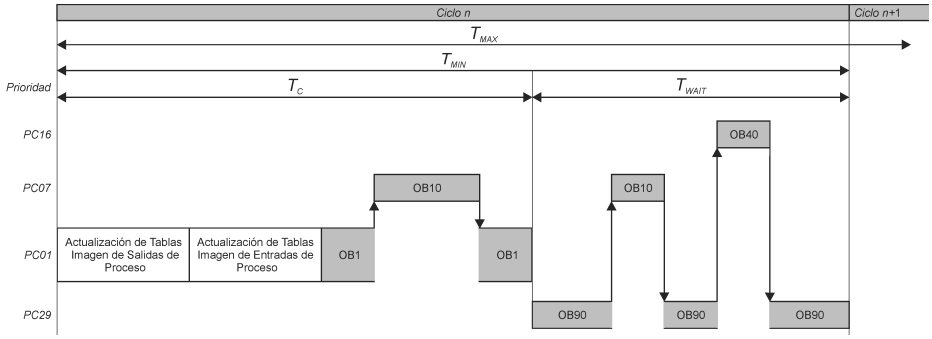

Fig. 1. Ejemplo de ciclo de ejecución en un *PLC Siemens*

2 Conceptos de Tiempo Real

2.1 Definición de Sistema de Tiempo Real

Es ampliamente aceptada en la literatura actual de tiempo real (*TR*) la definición realizada por Stankovic ([3]), la cual dice que: *"En los Sistemas de Tiempo Real* (*STR*) *los resultados no sólo deben ser correctos aritmética y lógicamente sino que, además, deben producirse antes de un determinado tiempo, denominado vencimiento"*.

Este vencimiento es el plazo que tiene la tarea para que su resultado sea aceptable. Después de este plazo, puede ocurrir que la solución sea inaceptable, que conserve algún grado de aceptabilidad o que, incluso, resulte contrario a la esperada.

2.2 Control en Tiempo Real

Por lo general, un *STR* en una planta industrial, interactúa en un entorno con una dinámica temporal predefinida. Se conocen la relación existente entre las entradas, las salidas y las constricciones temporales de los procesos.

Cuando se presenta en pantalla alguna variable controlada en función del tiempo, de un sensor de planta, se dice que "es en *TR"*. Es decir, en la pantalla se presenta lo que sucede.

Este concepto de *"tiempo real"* es el que comúnmente es tomado como válido. El mismo es utilizado en sistemas de monitoreo, Interfaz de Operación Hombre-Máquina (*HMI*) y *SCADA* (Supervisory Control & Data Acquisition – Sistema de Supervisión & Adquisición de Datos). Se dice comúnmente que el sistema funciona en *TR*.

Sin embargo, cuando se refiere a Sistemas de Control de *TR*, el concepto no es correcto. Una definición válida es ([4]): *La respuesta en TR es la capacidad de un sistema de responder a un evento o ejecutar una acción de manera determinística, confiable, y garantizada dentro de un período de tiempo determinado*.

Esta definición, sumada a la definición de *STR,* claramente expone que la respuesta del sistema debe realizarse dentro de un determinado período de tiempo.

2.3 Tiempo del Lazo de Control

Los *STR* utilizados en control, por lo general no difieren a grandes rasgos, de cómo se controlan un sistema clásico. Estos comparan el estado actual con el estado que se requiere alcanzar. Posteriormente, corrigen las variables de entrada de manera que el sistema se comporte de la manera que se desea. Esto lo logran a través de la lógica de control del sistema y la misma, puede ser muy simple o muy complejo, dependiendo del sistema a controlar. El tiempo que toma realizar esto, es denominado *Tiempo del Lazo de Control* (*TLC* - Figura 2).

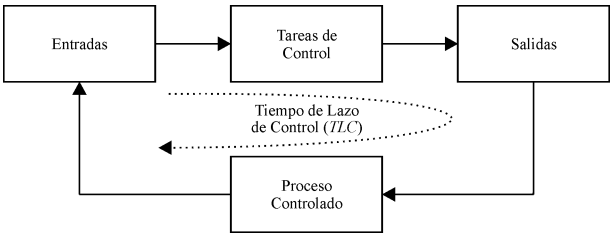

Fig. 2. Lazo de Control Típico.

El *TLC* es de suma importancia cuando las constricciones de tiempo del sistema controlado, imponen que el mismo se realice como máximo cada determinado tiempo. También, por lo general, el *TLC* posee un tiempo mínimo.

2.4 Sistemas de Tiempo Real Duro y Blando

Un *STR* duro, por definición es aquel en el cual las constricciones de tiempo son respetadas, es decir se cumple en ejecutar todas las tareas antes de sus vencimientos. El no cumplimiento, puede ocasionar que el sistema deje de funcionar o no funcione correctamente.

Por otro lado, un *STR* blando puede eventualmente perder vencimientos de sus tareas, sin provocar que el sistema funcione incorrectamente. Además, existen sistemas mixtos en los cuales conviven tareas duras con tareas blandas.

2.5 Planificación Mediante un Ejecutivo Cíclico o Prioridades

La planificación mediante un ejecutivo cíclico de *TR,* se rige mediante un ordenamiento de las tareas fijas de antemano. Un ejemplo de esto, es el ordenamiento que posee el *PLC* como el descripto en el punto 1.1.

Este tipo de planificación solo puede ser utilizado, cuando sumados todos los peores tiempos de ejecución de todas las tareas involucradas, el resultado, es menor o igual que el vencimiento de cada tarea dura. De otra manera, las tareas duras del *STR* podrían perder sus vencimientos.

No obstante, existen otros tipos de planificaciones. La planificación por prioridades apropiativa, es una de las alternativas más utilizadas. En particular se puede planificar por prioridades fijas o dinámicas. Entre las más conocidas en prioridades fijas, se encuentran *Periodos Monotónicos Crecientes* (*Rate Monotonic – RM*) ([5]) y en prioridades dinámicas, *Menor Tiempo al Vencimiento* (*Earliest Deadline First - EDF*) ([5]). Por otro lado, se dice que un planificador es apropiativo, si puede desalojar en cualquier instante a una tarea del recurso compartido, para cederlo a otra.

En *RM* las prioridades se asignan de manera que la tarea de menor periodo posee la mayor prioridad y así sucesivamente. En *EDF* la tarea que para un instante *t* se encuentra más próxima a vencerse posee la mayor prioridad.

3 Aplicación Real de un *STR*

A continuación se describe una aplicación real de un *STR*. La misma fue desarrollada sin tener en cuenta que pudiera ser extendida y consecuentemente no se aplicaron herramientas y conceptos de *TR*. Se presentaran sus debilidades y alternativas utilizando las herramientas de *TR* provistos por el *SO* del *PLC* utilizado para su fabricación.

3.1 Sierra de Corte de Barrotes de Aluminio

La Sierra de Corte de Barrotes de Aluminio está basada en un *PLC Siemens S7-300* ([2]). El mismo se encarga de controlar el corte de los barrotes de aluminio y de la mesa transportadora (Figura 6).

Se puede describir a grandes rasgos dos tareas principales. La primera es el manejo del sistema de corte de barrotes y la segunda es una *HMI*. Ambas compiten por el recurso computacional (*PLC*).

La sierra tiene requisitos bien definidos para cumplir con en el corte. La *Tarea 1* se encarga del transporte y corte con la precisión deseada y es procesada por *PLC*. La *Tarea 2* (*HMI*), intercambia información con un sistema *SCADA*. La información intercambiada requiere de cierto procesamiento intensivo por parte del *PLC* previo a su envío.

3.1.1 Principio de Funcionamiento de la Sierra

El principio de funcionamiento es el que se describe a continuación. Los barrotes ingresan a la sierra por una mesa de rodillos, que se mueve a una velocidad constante (ver Figura 6). Esta mesa, está equipada con un codificador (*encoder*) que determina la longitud actual de corte. Cuanto mayor es el tiempo que el sistema activa la mesa de rodillos, mayor es la longitud del barrote que entra a la sierra y consecuentemente, mayor es la longitud que acusa el codificador.

Esta longitud de corte es controlada por la *Tarea 1* del *PLC*. Cuando la longitud de corte solicitada, coincide con la longitud indicada por el codificador, más una tolerancia mínima, se detiene la mesa de rodillos, se enciende la sierra y se comienza el proceso de corte.

Sin la *Tarea 2* (*HMI*), la programación del *PLC* cumple con los requisitos de precisión en el corte.

3.1.2 Problemas en la Programación del *PLC* de la Sierra de Corte

La sierra actualmente posee un error de concepto en su programación. El mismo consiste, en que si se agregan tareas al ciclo de ejecución, el tiempo de consulta del *PLC* al codificador se incrementa y consecuentemente la precisión del corte disminuye.

Cuando se incluyen la *Tarea 2*, al ciclo de ejecución del *PLC*, se extiende a punto tal, que lleva la sierra a ejecutar cortes por fuera de la tolerancia máxima admitida.

La planificación de estas tareas, es un problema típico de *TR*, donde la concurrencia de las mismas sobre el recurso compartido, requiere ser atendidas sin que se pierdan las constricciones temporales de la *Tarea 1*. Además, es deseable que tampoco se pierdan las constricciones temporales de las restantes tareas.

3.2 Análisis del *STR* de la Sierra de Corte

En la Figura 3 se presenta la evolución temporal de la *Tarea 1* y las otras tareas (*O. Tareas*) en el *PLC*, sin la inclusión de la *Tarea 2*. Los peores tiempos de ejecución son de 50ms para la *Tarea 1* y *O. Tareas*.

Se hace notar que a fines prácticos, en un ejecutivo cíclico y sin definir un período, cada tarea posee el mismo periodo de ejecución, que la sumatoria de todos los peores tiempos de ejecución de cada tarea.

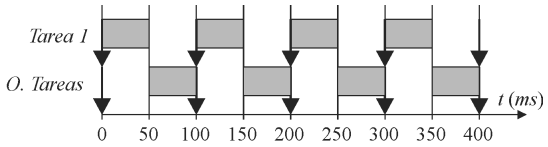

Fig. 3. Diagrama temporal con solo la *Tarea 1*.

Como se puede apreciar en esta solución, el peor *TLC* es de 100ms y consecuentemente la *Tarea 1* (*corte*) se ejecuta cada 100 ms como máximo. Según la especificación del equipo de corte de sierra de barrotes, la misma se debería ejecutar con un *TLC* no mayor a 150 ms, para cumplir con la precisión necesaria.

Se presenta a continuación en la Figura 4, como resulta el diagrama de tiempo incluyendo la *Tarea 2* (*HMI*) que consume como máximo 100 ms, en la ejecución cíclica del *PLC*.

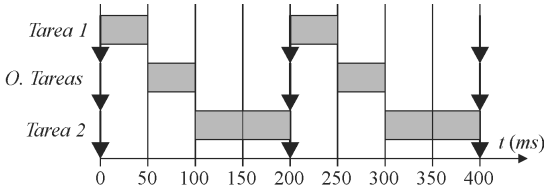

Fig. 4. Diagrama temporal incluyendo todas las tareas.

Como se puede apreciar, el *TLC* es de 200ms y no permite cumplir con la ejecución de la *Tarea 1* cada 150 ms. Con la planificación de tareas con un ejecutivo cíclico, este problema no tiene solución, salvo que se sacrifiquen algunos de los requerimientos o tareas.

4 Soluciones

4.1 Solución Teórica del *STR*

Para encontrar una solución es necesario analizar otros tipos de planificación. La planificación por prioridades fijas apropiativa por *RM,* puede brindar una solución eficiente en este caso.

Es claro que el *TLC* no puede ser menor a 100 ms y mayor a 150 ms. Tampoco es posible dejarlo en 100 ms, porque podría no dejar tiempo del microprocesador para ejecutar la *Tarea 2*. Consecuentemente, es necesario reevaluar los requerimientos del sistema.

Una actualización del sistema *SCADA* menor a 1 segundo, no es requerida para la supervisión. De esta manera y adoptando un *TLC* de 150 ms, con una planificación que permita la apropiación del recurso, es posible lograr planificar las 3 tareas sin que se venzan sus constricciones temporales.

Por otro lado, se debe ordenar a las tareas dando la mayor prioridad a la *Tarea 1*, la segunda prioridad a *O. Tareas* y finalmente la tercera prioridad a la *Tarea 2*. En la Figura 5 se presenta el diagrama temporal resultante.

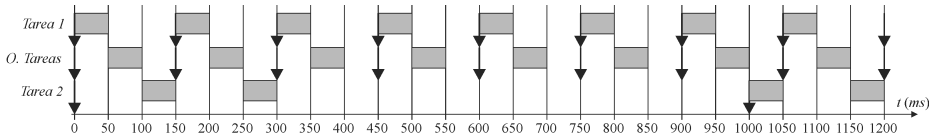

Fig. 5. Diagrama temporal final.

Se puede apreciar que el *STR* de esta forma planificado, cumple con los requisitos de precisión de la sierra y ninguna tarea pierde su vencimiento. Además, aparece tiempos ociosos a partir de los 400ms cada 150ms, hasta que una nueva instancia de la *Tarea 2* se presente. Este tiempo podría ser utilizado para la programación de futuras tareas con constricciones temporales que estén en el orden del segundo.

Fig. 6. Mesa de rodillos y sierra de barrotes.

Este caso es de simple resolución y puede ser realizado gráficamente. En sistemas más complejos, es necesario validar que el sistema es planificable por *RM* mediante un test de planificabilidad como los presentados en [6, 7].

4.2 Solución Implementada por el Fabricante

La solución implementada se resolvió realizando una división de la *Tarea 2* (*HMI*). Esta división fue realizada por el programador de la sierra. Sin embargo, las desventajas de resolver este tipo de problemas de esta forma, es que no es el *SO* del *PLC,* quien apropia a la tarea que se encuentra en el *CPU* y cede el recurso a otra tarea de mayor prioridad. Esta forma, no garantiza que si la tarea se extiende por

encima de su tiempo de ejecución, las tareas duras del sistema cumplan con sus vencimientos.

4.3 Solución Deseada

Por limitaciones propias en la construcción del *SO* del *PLC,* no es posible implementar la solución teórica de manera directa. Esto se debe, a que no se puede realizar un tiempo de ciclo máximo (T_{MAX}) de 150ms ya que la Tarea 2 se vería interrumpida y no terminaría nunca de ejecutarse. El problema es que, si bien el *SO* puede apropiar una tarea, no lo realiza entre ciclos de ejecución, sino que solamente dentro del ciclo de ejecución y reiniciaría la *Tarea 2* en vez de continuarla.

No obstante, es posible definir para la *Tarea 1*, un tiempo de ejecución periódico de 150 ms y ejecutarla en un *OB* con una prioridad mayor que el resto. De esta forma, se garantizan las constricciones temporales del fabricante en la especificación de la precisión del corte. Para esto, es necesario que la *Tarea 1*, lea directamente el valor del codificador y no a través de la tabla de imagen. Como la sierra cuenta con un modulo que lo permite (*Siemens SIMATIC FM-350* ([8])) esto es posible de realizarse.

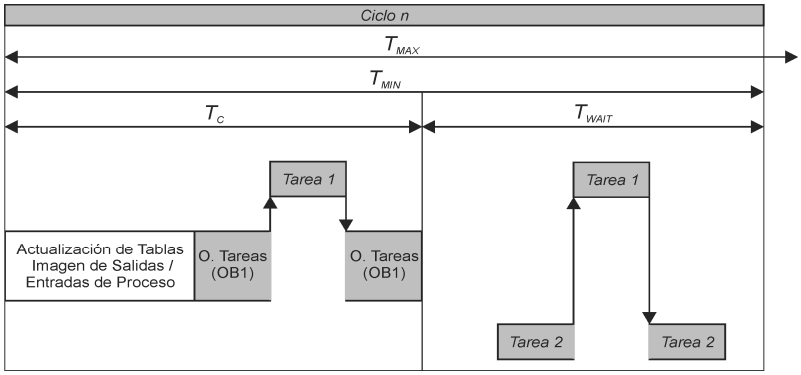

Fig. 7. Posible ejecución de la solución deseada.

La *Tarea 2,* debe ser programada con un tiempo de ejecución periódico de 1 segundo, dentro de un *OB* con menor prioridad que el de la *Tarea 1* y que *O. Tareas*. Las *O. Tareas*, deben ser programadas dentro del *OB1*. Los valores de T_{MAX} se define en 300ms, el T_{MIN} en 100ms, garantizando de esta forma la ejecución por ciclo mínima de una vez la *Tarea 1* y como máximo dos veces. Esto se debe, porque si se instancia la *Tarea 2* después de finalizar la ejecución del *OB1*, la misma comienza a ejecutarse. Si durante la ejecución de esta, se instancia la *Tarea 1* el *SO* del *PLC* apropiara a la *Tarea 2* para darle el *CPU* a la *Tarea 1*. Lo mismo ocurre con *O. Tareas* y la *Tarea 1* (Figura 7).

Esta solución resuelve el problema de forma óptima y sin necesidad de apelar a la creatividad y pericia del programador. Además, garantiza la ejecución de la tarea de corte con la precisión deseada. En la Figura 7, se presenta el comportamiento del

sistema de esta manera planteado, durante un ciclo de ejecución, en donde podría darse que la *Tarea 1* se ejecute 2 veces en un mismo ciclo.

5 Conclusiones

El análisis aquí realizado, demuestra que planteos con planificaciones de tareas por prioridades, resuelven de manera más eficiente problemas de *TR*, que la planificación estática por ejecutivo cíclico. Sin embargo, esto conlleva a que los programadores de este tipo de equipos, donde las características de funcionamiento son de *TR*, replanten los problemas con conceptos de *TR*.

La disciplina *STR* ha aportado en las últimas tres décadas nuevas herramientas de análisis, diseño y programación de sistemas, que poco a poco se introducen en el diseño de los *PLC*. Además, en la actualidad está dejando de ser una materia de posgrado, para ser una materia de grado en varias carreras como: Ing. Electrónica, Ing. Sistemas, Lic. Sistema, Lic. Informática, etc.

No obstante, para los problemas como el planteado, se encontraran soluciones que dependerán de la creatividad de los programadores para ajustar el modelo a un programa de ejecución cíclica, cuando en realidad lo que se debería realizar es modelar el sistema aplicando la mayor cantidad de herramientas de *TR* que el *SO* del *PLC* provea. También es común que este tipo de problemas se resuelvan aumentando el poder computacional, que por lo general, conlleva en un costo económico superior.

La inclusión de las nuevas herramientas de *TR* para resolver problemas de control en procesos industriales, sin dudas podría brindar el aporte necesario de claridad, eficiencia y determinismo que tales aplicaciones requieren.

Referencias

[1] Siemens. Available: http://www.automation.siemens.com/mcms/automation/en/industrial-controls/Pages/default.aspx

[2] Siemens, *SIMATIC S7-300 Automation System Module data*: Siemens, 2006.

[3] J. A. Stankovic, "Misconceptions About Real-Time Computing: A Serius Problem for Next-Generations Systems," *IEEE Computer,* vol. Octubre, pp. 10-19, 1988.

[4] R. Segarra, "Control de Tiempo Real," *Instrumentación y Control,* vol. 2, 2004.

[5] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of the ACM,* vol. 20, pp. 46-61, 1973.

[6] M. Joseph and P. Pandya, "Finding Response Times in Real-Time System," *The Computer Journal (British Computer Society),* vol. 29, pp. 390-395, 1986.

[7] J. M. Urriza*, et al.*, "Cálculo del Peor Tiempo de Respuesta en Sistemas de Tiempo Real con Limitados Recursos Computacionales," in *XXXIV Conferencia Latinoamericana de Informática. CLEI 2008*, ed. Santa Fe, Argentina, 2008, pp. 1415-1424.

[8] Siemens, *FM 350-2 counter module Installation and Parameter Assignment*, 2008.