Saber, Universidad de Oriente, Venezuela.Vol. 20 Nº 3: 334-342. (2008)

MODELO DINÁMICO DE UN ROBOT INDUSTRIAL UTILIZANDO ARQUITECTURAS DE PROCESADORES DISPERSOS

THE DYNAMIC MODEL OF AN INDUSTRIAL ROBOT USING ARCHITECTURES OF DISPERSED PROCESSORS

Alfonso S. Alfonsi, Cruz E. Rivero

Grupo de Arquitecturas de Sistemas de Control, Departamento de Computación y Sistemas, Escuela de Ingeniería y Ciencias Aplicadas, Universidad de Oriente, Núcleo de Anzoátegui, 6001, Barcelona, Venezuela. e-mail: asas@cantv.net

RESUMEN

Se obtiene el modelo dinámico de un robot industrial de N articulaciones utilizando arquitecturas de procesadores dispersos. Se resuelve tomando como base el procedimiento de Newton-Euler, considerando los aspectos de un planificador, cuyo tiempo de ejecución es el tiempo de cómputo. Enmarcado dentro de la metodología para el desarrollo de sistemas de tiempo real, se conecta un sistema multiprocesador a los eslabones del robot, y debido al acoplamiento dinámico entre los eslabones adyacentes, las relaciones precedentes se manifiestan por medio de las subtareas a ser ejecutados en el sistema. Es decir, que un adecuado desarrollo conduce a una formulación recursiva en la que se obtienen posición, velocidad y aceleración del eslabón i, y posteriormente las fuerzas y pares que actúan sobre el eslabón i referidos a la base del robot. Se ejemplifica con el desarrollo de un planificador estático que determina el modelo dinámico de un robot industrial, utilizando HOOD como metodología y el lenguaje Ada como soporte de ejecución, ambos ampliamente utilizados en entornos de tiempo real.

Palabras clave: Modelo dinámico, multiprocesador, planificador de tiempo real, robot industrial.

ABSTRACT

The dynamic model of an industrial robot of N joints is obtained using architectures of dispersed processors. Newton-Euler procedure was used to solve the equations, considering the aspects from a scheduler, whose run time is the computation time. A multiprocessor system is connected to the robot's links, for the development of a real-time system, and due to the dynamic joining among the adjacent links, the precedent relationships are manifested by means of the subtask to be run in the system. That is to say that an appropriate development drives to a recursive formulation in which position, speed and acceleration of the link i is obtained, and later on the forces and pairs that acts on the link i referred to the robot's base. It's exemplified with the development of a static scheduler that determines the dynamic pattern of an industrial robot, using HOOD like methodology and the Ada language as run support, both broadly used in real time environments.

Key words: Dynamic model, multiprocessor, scheduling of real time, industrial robot.

INTRODUCCIÓN

El modelo dinámico relaciona: la localización del robot definida por sus variables articulares o coordenadas de localización de su extremo, con sus derivadas velocidad y aceleración; las fuerzas y pares aplicadas en las articulaciones (o en el extremo del robot); y los parámetros dimensionales del robot, longitud, masas e inercias de sus elementos. Esto conduce al desarrollo de las ecuaciones de movimiento dinámico para las diversas articulaciones en términos de los parámetros geométricos e inerciales de los elementos, complicándose a medida que aumenta el número de grados de libertad, teniendo que resolverse de manera iterativa mediante la utilización de un procedimiento numérico (Barrientos *et al*. 1997).

Por otro lado, las arquitecturas de procesadores dispersos (paralelos y distribuidos), son campos amplios y definidos en la arquitectura de computadores, conceptualizados en la rama del conocimiento de la ciencia de la computación. Una arquitectura de procesadores dispersos es un conjunto de procesadores que son capaces de trabajar cooperativamente para solucionar un problema computacional. Esta definición incluye a supercomputadores paralelos que tengan cien

Recibido: febrero 2007. Aprobado: enero 2008.
Versión final: febrero 2008.

o mil procesadores, redes o estaciones de trabajo, estaciones de trabajos de multiprocesadores, entre otros. Son interesantes porque ellos ofrecen el potencial para concentrar recursos (procesadores, memoria, ancho de banda e I/O) sobre importantes problemas de computación.

A esta arquitectura se le pueden planificar la ejecución de las tareas, asignándole características temporales significativas, que permitan la obtención de resultados lógicos, como posición, velocidad, aceleración, fuerza y par, sumándole al planificador características de tiempo real.

La motivación del presente aporte se encuentra influenciada por el trabajo de Luh y Lin (1982), quienes entregan un algoritmo de planificación en ambiente multiprocesador para tratar el modelo dinámico de un robot, adjudicándole tareas específicas estáticas a cada procesador. También se fundamenta, en el trabajo realizado por Taewhan *et al*. (1994), quienes desarrollan un algoritmo de planificación para recursos condicionales compartidos. Stankovic *et al*. (1995), aporte que reúne enfoques de algoritmos de planificación de tiempo real, donde atacan la forma como integrar la coacción de precedencia y distribución de recursos. Sáez *et al*. (1997), autores que analizan la conducta y el desempeño de cuatro algoritmos dinámicos para la planificación en multiprocesadores. Cheng *et al*. (1997), con su propuesta del algoritmo *Least Space-Time First* (*LSTF*), que trata tareas complejas con un modelo general de multiprocesadores. Y Petit *et al*. (2000), quienes proponen nuevas arquitecturas de procesadores dispersos.

El objetivo es la formulación del modelo dinámico de un robot industrial utilizando para ello arquitecturas de procesadores dispersos, considerando los aspectos de planificación de tareas, cumpliendo los requisitos de recursos y de tiempo.

La obtención del modelo dinámico de un robot es importante si se quieren alcanzar los siguientes fines: simulación del movimiento, diseño y evaluación de la estructura mecánica, dimensión de los actuadores, diseño y evaluación de los controladores.

MATERIALES Y MÉTODOS

Los robots de eslabones seriales son mecanismos complicados donde la conducta dinámica de cada eslabón afecta los adyacentes, para darle el enfoque paralelo a la resolución del problema, se le asigna un sistema multiprocesador, y en el acoplamiento dinámico se crean relaciones precedentes a lo largo de las subtareas asignadas a los procesadores que crean unos intervalos inútiles de tiempo, que se reflejan en los procesadores que esperan por las subtareas predecesoras, para iniciar la ejecución de las suyas. Lo anterior amerita una planificación de la ejecución de cada tarea y subtarea, utilizando un algoritmo que cumpla con los límites de tiempo.

Se utiliza la metodología para el desarrollo de programas en paralelo PCAM (Partición, Comunicación, Aglomeración y Mapeo), y se toma el paralelismo de tarea, el cual se basa en el principio de cooperación entre dos o más procesos, donde diferentes procesos cooperan para tratar de resolver diferentes partes de un problema común, pasándose datos entre sí en búsqueda de la solución (Coller *et al*. 1998).

Por otro lado, la obtención del modelo dinámico a partir de la formulación de *Newton-Euler*, se parte del equilibrio de fuerzas y pares, permitiendo que el desarrollo de estas ecuaciones conduzca a una formulación recursiva en la que se obtiene la posición, la velocidad y la aceleración del eslabón *i* referidos a la base del robot a partir de los correspondientes del eslabón anterior (*i*-1) y también, del movimiento relativo de la articulación *i*. Por lo tanto, partiendo del eslabón 1 se llega al eslabón *n*. En base a estos datos se procede a determinar las fuerzas y pares actuantes sobre el eslabón i referida a la base del robot a partir de los correspondientes del eslabón superior (*i*+1), recorriéndose así, todos los eslabones, desde el eslabón *n* al eslabón 1.

De lo expresado anteriormente, se elige una estructura de cálculo con un esquema de arquitectura dispersa (multiprocesador) para el modelo dinámico de un robot de *n* articulaciones, ya que se puede controlar la ejecución de las tareas a realizar en las *n* articulaciones, particionándolas en *n* subtareas que serán asignadas por el planificador de tiempo real a los *n* procesadores que estén listos para responsabilizarse de ésta ejecución. Se fundamenta este proceso por el algoritmo de computación recursiva de las ecuaciones dinámicas (Luh *et al*. 1980; Luh y Lin 1982).

Sumando todo se obtiene un algoritmo de planificación que depende del número de procesadores del sistema y su homogeneidad o heterogeneidad, de las relaciones de precedencia entre las tareas de

aplicación, del método de sincronización de tareas, y de las restricciones de tiempo que lo convierten en un sistema de tiempo real esencial.

División de subtareas para calcular par y fuerza

Se utiliza el algoritmo de Luh-Walker-Paul de la formulación Newton-Euler (Luh *et al*. 1980), presentado en Barrientos *et al*. (1997) resumido a continuación:

1. Asignar a cada eslabón un sistema de referencia de acuerdo a las normas de *Denavit-Hartenberg* (*D-H*).

2. Establecer las matrices de rotación ${}^{i-1}R_i$ y sus inversas.

3. Establecer las condiciones iniciales para el Sistema de la base $\{S_0\}$: La velocidad angular, la aceleración angular y la velocidad lineal son típicamente nulas salvo que la base del robot esté en movimiento. Para el extremo del robot se conocerá la fuerza ${}^{i+1}F_{i+1}$ y el par ejercido externamente ${}^{i+1}T_{i+1}$.

La fuerza externa F_i y el par externo T_i de la articulación *i* (o eslabón *i*) se obtienen en dos procesos, el primero calculando velocidad, aceleración, velocidad angular y aceleración angular, del eslabón en un tiempo definido, desde la base a la mano, subdivididas en diez (10) subtareas de las tareas *i*, para *i* = 1,2,...*n*; y luego, la $(F_i)/(T_i)$ en una articulación se calcula en orden inverso, desde la mano a la base, cinco subtareas de *i* = *n*,...1. Las subtareas que se generan se muestran en la Figura 1.

$$T1/i: {}^{i}w_i = {}^{i}R_{i-1}\left({}^{i-1}w_{i-1} + Z_0\dot{q}_i\right) \quad \textit{rotación}$$

$$T1/i: {}^{i}w_i = {}^{i}R_{i-1}\,{}^{i-1}w_{i-1} \quad \textit{traslación}$$

$$T2/i: T2 = {}^{i-1}w_{i-1} \times Z_0\dot{q}_i \quad \textit{rotación}$$

$$T2/i: T2 = 0 \quad \textit{traslación}$$

$$T3/i: {}^{i}\dot{w}_i = {}^{i}R_{i-1}\left({}^{i-1}w_{i-1} + Z_0\ddot{q}_i\right) + T2 \quad \textit{rotación}$$

$$T3/i: {}^{i}\dot{w}_i = {}^{i}R_{i-1}\,{}^{i-1}\dot{w}_{i-1} + T2 \quad \textit{traslación}$$

$$T4/i: T4 = {}^{i}\dot{w}_i \times {}^{i}p_i \quad \textit{rotación}$$

$$T4/i: T4 = {}^{i}w_i \times {}^{i}p_i + 2{}^{i}w_i \times {}^{i}R_{i-1}Z_0\dot{q}_i \quad \textit{traslación}$$

$$T5/i: T5 = {}^{i}R_{i-1}\,{}^{i-1}\dot{v}_{i-1} \quad \textit{rotación}$$

$$T5/i: T5 = {}^{i}R_{i-1}\left(Z_0\ddot{q}_i + {}^{i-1}\dot{v}_{i-1}\right) \quad \textit{traslación}$$

$$T6/i: T6 = {}^{i}w_i \times {}^{i}p_i$$

$$T7/i: {}^{i}\dot{v}_i = {}^{i}w_i \times T6 + T4 + T5$$

$$T8/i: T8 = {}^{i}\dot{w}_1 \times {}^{i}S_i$$

$$T9/i: T9 = {}^{i}\dot{w}_1 \times \left({}^{i}w_i \times {}^{i}S_i\right)$$

$$T10/i: {}_i a^{i} = T8 + T9 + {}^{i}\dot{v}_i$$

Los cálculos para la Fuerza/Torque a aplicar Ti se obtiene utilizando cinco subtarea de $i = n, \cdots 1$

$$T11/i: {}^{i}f_i = {}^{i}R_i\,{}^{i+1}f_{i+1} + m_i\,{}^{i}a_i$$

$$T12/i: T12 = {}^{i}R_{i+1}\left[{}^{i+1}n_i + \left(\left({}^{i+1}R_i\,{}^{i}p_i\right) \times {}^{i+1}f_{i+1}\right)\right]$$

$$T13/i: T13 = {}^{i}I_i\,{}^{i}\dot{w}_i + {}^{i}w_i \times \left({}^{i}I_i\,{}^{i}w_i\right)$$

$$T14/i: {}^{i}n_i = T12 + \left({}^{i}p_i + {}^{i}S_i\right) \times m_i\,{}^{i}a_i + T13$$

$$T15/i: \mathrm{T}_i = {}^{i}n_i^{T}\,{}^{i}R_{i-1}Z_o \quad \textit{rotación}$$

$$T15/i: \mathrm{T}_i = {}^{i}f_i^{T}\,{}^{i}R_{i-1}Z_o \quad \textit{traslación}$$

Figura 1. División de subtareas para el sistema.

Tiempo de ejecución de las subtareas

Las operaciones aritméticas para cada subtarea del programa están analizadas individualmente. Se asume que las operaciones de carga de datos no consume una cantidad significativa de tiempo, en comparación con operaciones de punto flotante, multiplicaciones, sumas y restas. Así el número de operaciones aritméticas que están listadas en Luh y Lin (1982), se utilizarán para representar la base para el cálculo de tiempo de ejecución de la subtarea requerida. No obstante, dependiendo del computador utilizado se puede determinar el tiempo de ejecución consumido para cada subtarea (Intel 2000).

Para el cálculo del cálculo del tiempo de ejecución (t_e) de una tarea se utiliza (1).

$$t_e = \frac{CT^* t}{CD} \qquad (1)$$

Siendo *CT* el ciclo de tiempo, *t* la unidad de tiempo, y *CD* el ciclo de desempeño del procesador.

Este tiempo, sumado al manejo de instrucciones por medio de las conexiones *pipeline*, originan un desempeño del procesador excelente al ejecutar las tareas, debido a que tanto la memoria de procesamiento y caché refuerzan lo eficaz que puede trabajar un equipo a una frecuencia de 500 MHz (Intel 2000).

Entonces, tomando la tarea *T1/1* que se ejecuta en 2 *CT*, en un procesador de 500 MHz (*CD*), y la unidad de tiempo *t* es un segundo, se tiene que $t_e = 4ns$.

Algoritmo de planificación para el sistema multiprocesador

Una vez definidas las subtareas genéricas y sus respectivos tiempos de ejecución, las cuales forman parte de la base de datos, se crea una tabla de relación entre las subtareas específicas, tomando como insumo, número de articulaciones 'NA', el tipo de articulación 'TA' y el número de procesadores 'NPR'; luego se construyen las subtareas específicas a utilizar para formar la tabla de relación, tomando en consideración las subtareas predecesoras y sucesoras.

Ahora, se procede a planificar las subtareas obtenidas en los procesadores para cumplir el compromiso de tiempo de ejecución, y asegurarse que la obtención de los Torques/Fuerzas a aplicar en las articulaciones estén dentro de un plazo de finalización definido.

En el algoritmo son empleados los siguientes términos: Factibilidad de programación: un total de *n* secuencias de subtareas, en las cuales la secuencia *j* para la tarea $j = 1,2,3,... n$, indica el orden de ejecución de esas subtareas en los procesadores *i* que satisface el cálculo o preparación precedente. Puntero de tiempo *CP(i)*: es el instante de tiempo en el cual una subtarea (*Tj/i*) completa su ejecución. Tiempo de ejecución *TE*: el tiempo que le lleva al procesador ejecutar la subtarea asignada. Número de subtareas predecesoras *NPRED*: es la cantidad de tareas predecesoras planificadas referidas a cada subtarea.

Cada uno de los *NPR* procesadores debe tomar algún estado durante un intervalo de tiempo completo de computación y permite a su vez, determinar los tiempos ociosos. Estos estados son: Ejecución, el procesador está en el proceso de ejecución de una subtarea. Listo, el procesador ha completado la ejecución de una subtarea y está preparado para ejecutar otra. Alto, el procesador está ocioso y espera por una subtarea para poder entrar a la actividad, el algoritmo debe evitar éste estado.

Las subtareas deben tomar también durante un instante de tiempo específico algún estado, el cual depende de la tenencia de todos los requerimientos de datos hábiles para un cálculo. Esto se utiliza para satisfacer las restricciones de precedencia. Los estados son: listo, todos los predecesores de una subtarea de la tarea *j*, deben estar preparados en un cierto instante de tiempo, porque sus resultados serán tomados por un procesador *i* en un instante de tiempo después; espera, cuando una subtarea no está preparada para su ejecución en un instante de tiempo; y completada, cuando una subtarea ha sido completada en un instante de tiempo.

Desarrollo del planificador estático

Para el modelado y construcción del planificador de tiempo real se utilizó la metodología *HOOD* (*Hierarchical Object-Oriented Design*), (Burns and Wellings 1994; Burns and Wellings 1995), esta considera los pasos sugeridos por la ingeniería de software: análisis y especificación de los requerimientos, diseño, codificación, prueba e integración.

HOOD se desarrolla según un proceso iterativo, centrado en la etapa de diseño, es decir, validar todos los

aspectos que se puedan (lógicos) en esta etapa, y se presta especial atención a la validación del comportamiento temporal, que lo diferencian de otras metodologías, (Spiteri 2007).

Con respeto al soporte de ejecución se usa el lenguaje de programación Ada comúnmente utilizado en el desarrollo de sistemas de tiempo real, a parte de ser la primera herramienta de programación ISO y ANSI que sirve se apoyo a la programación orientada a objeto concurrente, (Crespo y Alonso 2006). También tiene la facilidad de intercambiarse con otros lenguajes, lo que constituye una selección ideal óptima para todos los sistemas multilenguajes actualizados, (ISO/IEC 1995; Smith 2001; ISO/IEC 2006).

Se presentan algunos detalles del diseño, con la descomposición jerárquica de primer nivel del Planificador y la descomposición jerárquica del objeto activo Modelo, Figura 2. En la Figura 3 se muestra el diagrama de paquetes, donde sus compartimientos (definiciones de tipo y subprogramas) fueron suprimidos para mejorar el aspecto visual del diagrama, y la especificación del paquete de variables del planificador.

El sistema planificador requiere un computador con las siguientes características mínimas de hardware: un procesador Pentium, Celeron, Duron o Athlon de 400 MHz, 32 Mb de RAM, unidad de disco compacto superior a 8X, unidad de disco duro 5 Gb y monitor SVGA de 15", y tener instalado el sistema operativo Windows 98 o superior.

RESULTADOS

Se ejemplifica la experiencia, determinando el modelo dinámico de un robot industrial de tipo cilíndrico, donde la articulación 1 presenta un movimiento independiente tipo prismático y la articulación 2 presenta un movimiento independiente tipo rotacional. Se empieza llamando al sistema planificador y el guiará en el procedimiento para la obtención del modelo. En las Figuras 4 y 5 se muestran las ventanas principales del software, donde se descarga la información y se visualiza la planificación.

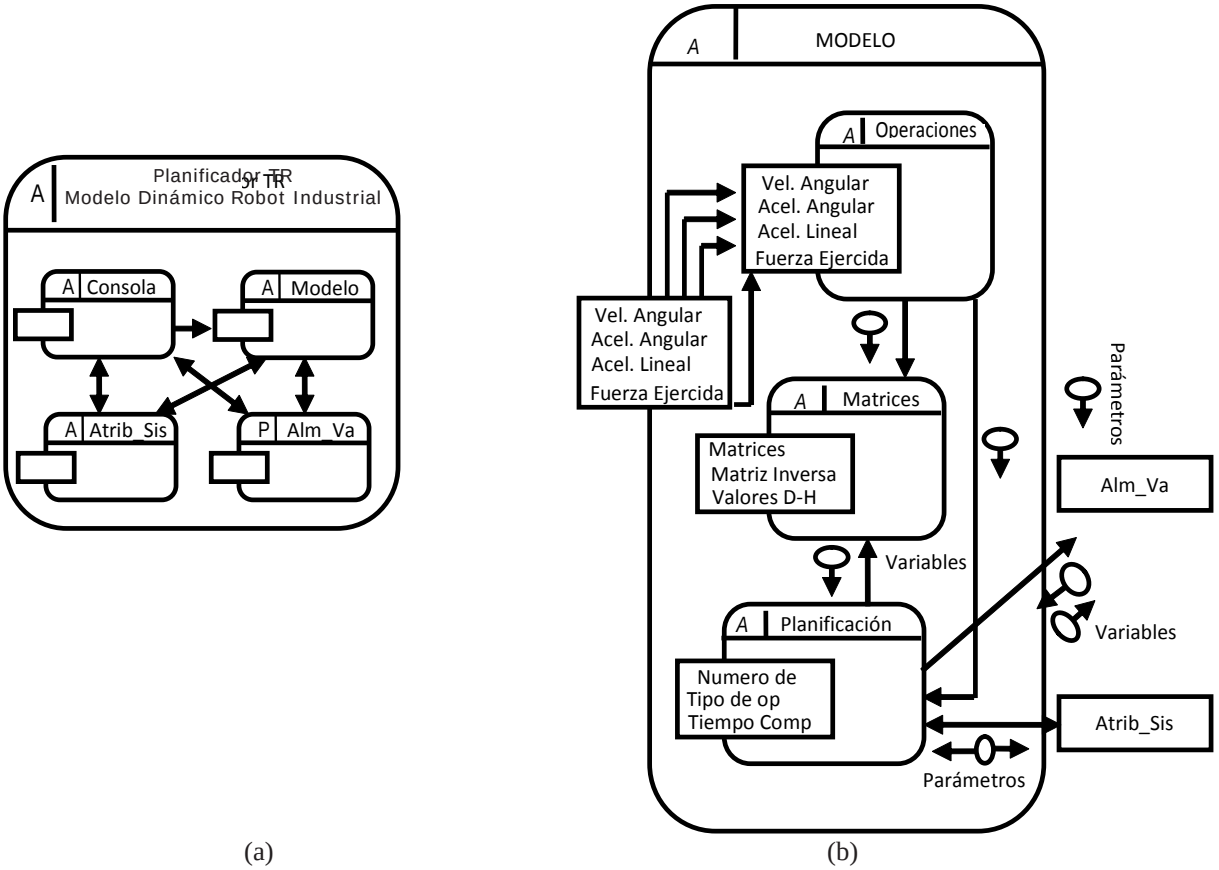


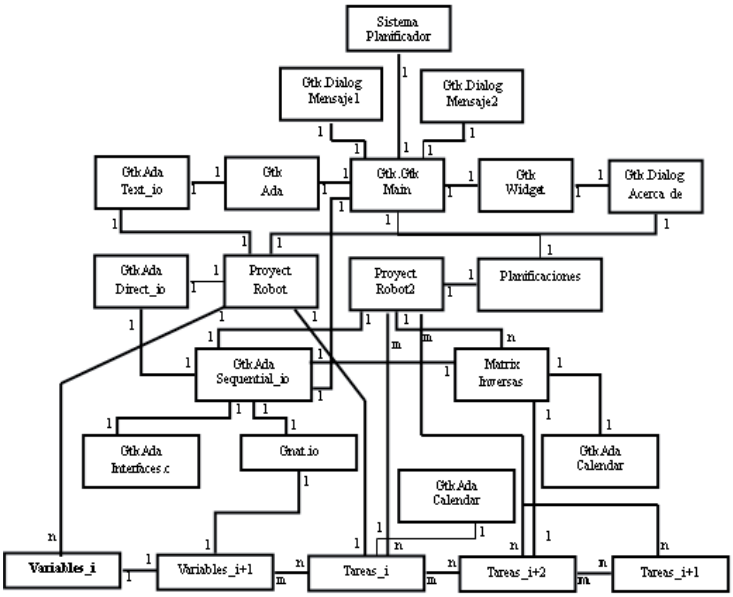
Figura 2. Detalles del diseño: (a) Descomposición jerárquica de primer nivel del sistema. (b) Descomposición jerárquica del paquete MODELO.

La Figura 6 ilustra la forma como se muestran las ecuaciones diferenciales en los objetos gráficos empleados para ello. A parte de esto, el sistema puede determinar los vectores que rigen la velocidad y la aceleración angular del modelo obtenido. Estos atributos son visualizados mediante la activación de los botones titulados como *Velocidad Angular* y *Aceleración Angular.*

El alcance del sistema no queda allí, se realiza el procesamiento simbólico de la interacción de cada una de las tareas mediante dos gráficos que representan a las articulaciones del robot (uno para cada eslabón), por medio de la cual es posible determinar los tiempos de ejecución de las tareas. Estos procesos se visualizan activando el botón correspondiente a *Diagramas de Planificación* en la ventana de planificaciones. Los tiempos de ejecución (*te*) para cada tarea planificada en las articulaciones se ilustra gráficamente en la Figura 7.

DISCUSIÓN

Lógicamente, el orden de complejidad computacional de la formulación recursiva *Newton-Euler* es O(*n*) lo que indica que depende directamente del número de grados de libertad. Para mayor información de los sistemas robotizados puede consultar Barrientos *et al.* (1997).

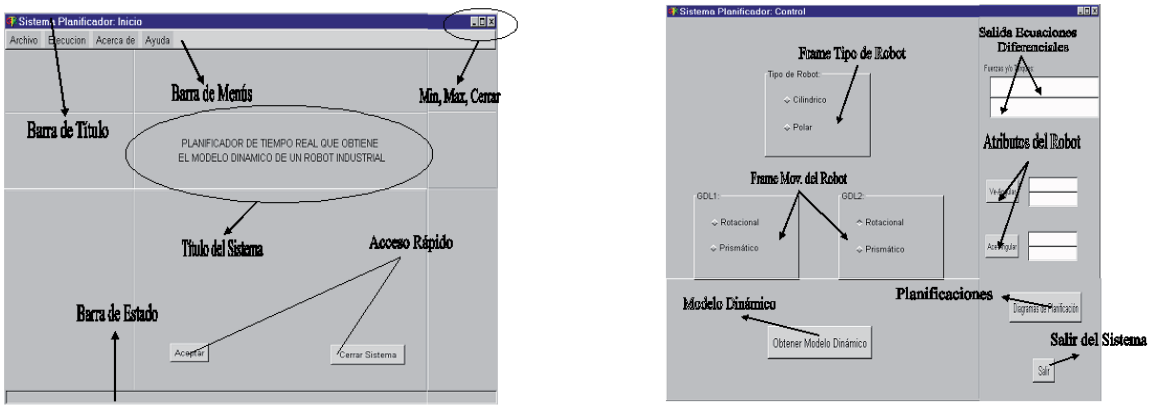


(a)

```
package variables is
   i,tipo_robot,tipo_movi:integer:=0;
   type matriz is array(1..10,1..2) of character;
   type cadena is array(1..2) of character;
   type tipo1 is array(1..10) of integer;
   type rotacion is array(1..3,1..6) of  character;
   s:String(1..10);
   indicar:array(1..4) of integer;
   tipo,tercer,alfa:tipo1;
   teta,distancia:matriz;
   letra:cadena;argu:float;
   m1,m2,m1inv,m2inv:rotacion;
   procedure inicio;
   procedure strcpy(destino:out
matriz;origen:in
   cadena;index:in integer);
   procedure print(nombre:in matriz;index:in
   integer);
```

(b)

Figura 3.- Detalles de la codificación: (a) Diagrama de paquetes. (b) Paquetes de variables.



(A) (B)

Figura 4. Ventanas principales: (A) Ventana Introductoria del Sistema Planificador. (B) Ventana de Control del Sistema.

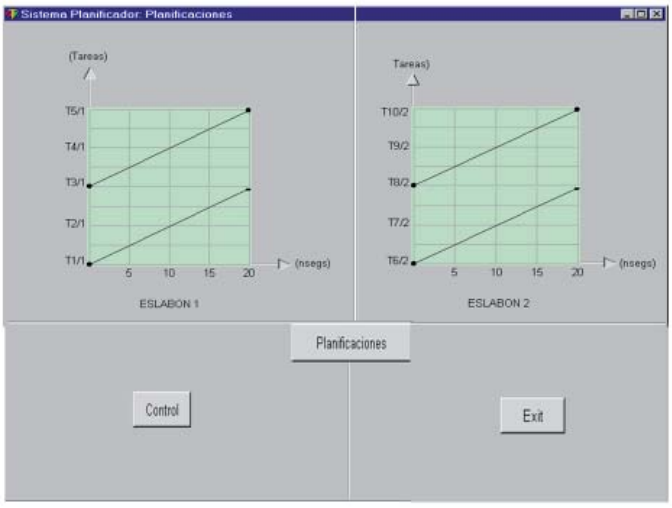


Figura 5.- Ventana de Planificaciones del Sistema Planificador.

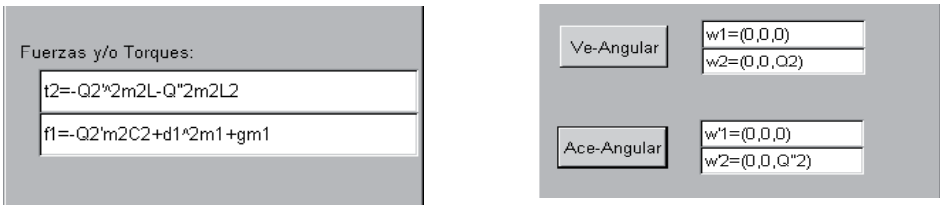


Figura 6.- Salida de ecuaciones diferenciales y atributos.

La propuesta se ambienta en arquitecturas de procesadores dispersos, en este caso paralelo, con la variante de aplicar un sistema multiprocesador conectado a las articulaciones de un robot industrial, y las tareas a ejecutarse para obtener el modelo dinámico, se planifican dependiendo del tiempo de ejecución de cada una de ellas y del procesador que esté disponible, almacenando los resultados en una memoria compartida. Haciendo énfasis en la utilización de esquemas de planificación de tiempo real, Luh y Lin (1982), habían planteado un esquema donde un procesador es asignado a una articulación y las tareas son específicas de esa articulación, teniendo su propia memoria para almacenar datos, y la planificación se lleva a cabo tal que el tiempo para completar el cálculo de las tareas sea el mínimo.

Por otro lado, tomando en consideración los aspectos de planificación, en el trabajo de Stankovic *et al*. (1995), se encuentran algunos enfoques de algoritmos de planificación de tiempo real que atacan el cómo integrar la coacción de precedencia y distribuir recursos en la planificación de tiempo real, que sería interesante atacar en futuras entregas.

Con respecto a la experiencia con el planificador, en la Figura 7 se aprecia que el tiempo de activación de la tarea T1/1 y T3/1 es el mismo (ambas se están ejecutando en paralelo y comienzan en 0 ns) en cambio, sus tiempos límites de ejecución son distintos (T1/1 finaliza en 4 ns y T3/1 finaliza en 12 ns). La tarea 2 planificada en el eslabón 1 (T2/1) está condiciona a que termine T1/1, por lo que su activación ocurre después de ésta, presenta un período de activación (T) entre (4 - 8 ns) y estipula un tiempo límite de ejecución de 5 ns. La tarea T3/1 no depende de T2/1, pero sí de T1/1, entones de ejecuta en paralelo con T1/1 pero termina luego. La tarea T4/1 se activa luego que finalicen tanto T3/1, T2/1 y T2/2 ya que depende de ellas. La tarea T3/2 depende de las tareas T1/1 y T1/2, lo que implica que su tiempo límite ocurre después que éstas finalicen. En la tabla 1 se especifican los parámetros de ejecución relacionados con las tareas del sistema.

CONCLUSIONES

El enfoque que se dio al tratamiento dinámico de un robot, al trasladar las ecuaciones dinámicas, que pueden ser ejecutadas bajo ambiente uniprocesador, a

un ambiente multiprocesador, pone de manifiesto, que una herramienta utilizada en buena forma puede ayudar a enfocar un problema, darle solución y abrir nuevos caminos para seguir el estudio. Se diseñó la propuesta utilizando un multiprocesador, apoyándose en los sistemas de tiempo real y los estudios de planificadores, siendo las tareas quienes definen el escenario que describe la carga, con un conjunto de restricciones temporales que especifican la respuesta del sistema. Se aplicó a un robot industrial de dos (2) articulaciones.

Se entrega un software que permite obtener el modelo dinámico basado en planificación estática en tiempo real. Clasifica los parámetros de tiempo relacionados con la ejecución de las tareas que intervienen en la obtención del modelo dinámico de un robot industrial. Los parámetros, valores y resultados obtenidos se presentan de forma variable de acuerdo a los argumentos o condiciones del robot industrial a procesar, lo que incide directamente en la planificación, ciclos de tiempo, desempeño y por su puesto, en los tiempos de ejecución de las tareas.

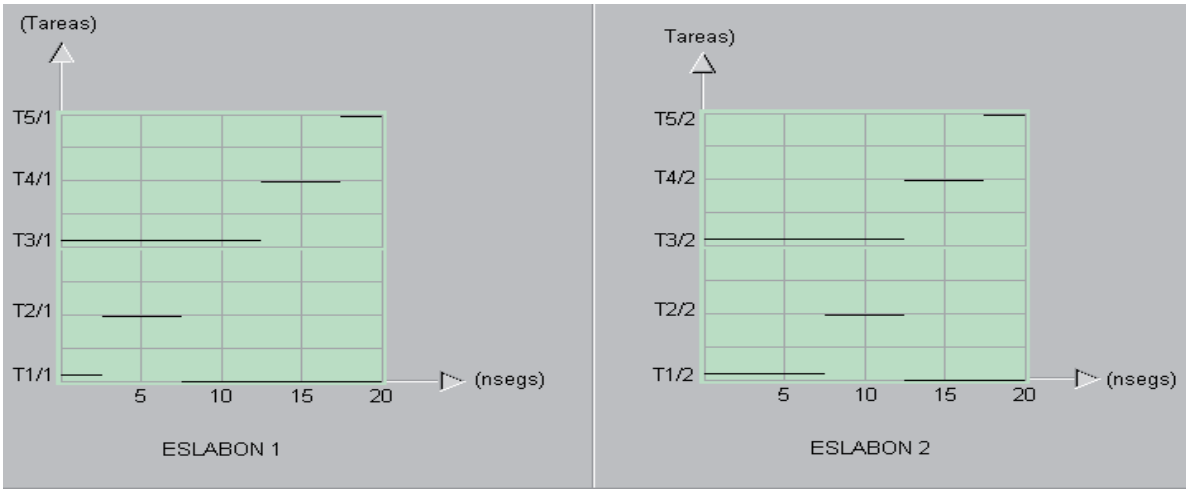


Figura 7.- Diagramas de planificación de las tareas.

Tabla 1. Relación de los (te) y (CT) de cada tarea.

Tareas	***Eslabón***	***Ciclos de Tiempo***	***Tiempo de Activación*** (ns)	***Período de Activación*** (ns)	***Plazo de finalización*** (ns)
T1/1	1	2	0	0 - 4	5
T2/1	1	2	4	4 - 7	4
T3/1	1	6	0	0 - 12	13
T4/1	1	2	13	13 - 17	5
T5/1	1	2	17	17 - 20	4
T1/2	2	2	0	0 - 4	5
T2/2	2	2	8	8 - 12	5
T3/2	2	6	0	0 - 12	13
T4/2	2	2	13	13 - 17	5
T5/2	2	2	17	17 - 20	4

REFERENCIAS BIBLIOGRÁFICAS

BARRIENTOS A., PEÑÍN L., BALAGUER C., ARACIL R. 1997. Fundamentos de Robótica. McGraw-Hill, España, 327p.

LUH J., LIN C. 1982. Sheduling of parallel computation for a computer-controlled mechanical manipulator. IEEE Trans. on Systems, Man, and Cybernetics. SMC-12(2):214-234.

TAEWHAN K., NORITAKE Y., LIU J., LUI C. 1994. A algorithm of the scheduling for conditionals shared recourses. IEEE Trans. 13(4).

STANKOVIC J., SPURI M., DI NATALE M., BUTTAZO G. 1995. Implications of classical scheduling results for real time systems. IEEE Computer. 28(6):16-25.

SAEZ S., VILA J., CRESPO A. 1997. Dynamics scheduling solutions for real-time multiprocessors systems. Control Eng. Practice. 5(-):1007-1013.

CHENG B., STOYENKO A., MARLOWE T., BARUAH S. 1997. LSTF: A new scheduling police for complex real-time task in multiple processor systems". Automática. 33(5):921-926.

PETIT S., SAHUQUILLO J., PONT A. 2000. Performance evaluation of consistency models using a new simulation enviroment for SVMS. In the 2th International Workshop of Software Distributed Shared Memory (ICS'2000), Santa Fé, USA.

COLLER D., SINGH J., GUPTA A. 1998. Parallel computer arquitecture: A hardware/software approach. Morgan Faufmann editors, USA, pp.1100.

LUH J., WALKER M., PAUL R. 1980. On-line computational scheme for mechanical manipulators. ASME Journal of Dynamic Systems, Measurement, and Control. 102(2):69-76.

INTEL CORPORATION. 2000. Soporte Técnico Intel Corporation". 16 de junio 2001.Disponible en http://www.intel.com/support/processor/pentiumiii/perfomance.

BURNS A., WELLINGS A. 1994. A structured design method for real-time systems. Real-Time Systems. Journal of Real Time Systems. 6(1):73-114.

BURNS A., WELLINGS A. 1995. HRT: A structured design method for hard real-time Ada systems. Real-Time Systems. Elseiver Ed., Amsterdan.

SPITERI A. 2007. A comparison of software analisys and desing methods for real tiem systems. In: Proc. of World Academy of Sciencia, Engeneering and Thechnology. 21(-):55-59. Available in: http://www.waset.org/pwase/v21/.

ISO/IEC. 2006. Ada reference manual. ISO/IEC 8652:200Y(E). 379p. http://www.adaic.com/standars/osrm/.

CRESPO A., ALONSO A. 2006. Una panorámica de los sistemas de tiempo real. Rev Iberoamericana de Automática e Informática Industrial. 3(2):7-18.

SMITH M. 2001. Object-Oriented Software in Ada 95. 2th edition. School of Computing. University of Brighton.

ISO/IEC. 1995. Ada reference manual. ISO/IEC/ANSI 8662:1995. 582p. Available in: http://www.adaic.com/standars/osrm/.