DESARROLLO DE UN CONTROLADOR ABIERTO PARA UN ROBOT INDUSTRIAL TIPO SCARA

González Sánchez,J.L.
Baeyens Lázaro, E.
Gayubo Rojo, F.
{jossan, enrbae, fergay}@eis.uva.es

Instituto de las Tecnologías Avanzadas de la Producción (ITAP)
ETS de Ingenieros Industriales Universidad de Valladolid
Paseo del Cauce, s/n. 47011 Valladolid

Resumen

A lo largo de la última década ha ido en aumento el interés hacia la utilización de plataformas hardware y software abiertas como soporte de aplicaciones de control industrial. Prueba de ello es la aparición de diferentes propuestas (OSACA, OSEC, OMAC) para la definición de una arquitectura abierta para el control de equipos de producción. La utilización de un PC (ordenador personal) como plataforma de control facilita la conectividad, estandarización y disponibilidad de componentes, y la reducción de costes. Existe, sin embargo, un aspecto que ha sido poco estudiado: la aplicación de estos conceptos sobre equipos diseñados con una estructura de control propietaria.

*Como una primera etapa para estudiar su viabilidad y evaluar dificultades y limitaciones, se ha desarrollado un sistema de control abierto para un robot industrial YAMAHA YK7000, de tipo SCARA. El controlador implantado sigue la filosofía definida por la arquitectura OMAC (*Open Modular Architecture for Controllers*). Las funciones de control se han implantado sobre una plataforma dual (dos PCs): uno de ellos soporta las funciones críticas, ejecutando las tareas de cálculo de trayectorias y control de ejes a través de una tarjeta de control de ejes comercial. El segundo PC soporta las funciones de interfaz con el usuario y otras aplicaciones no críticas. La comunicación entre ambos se realiza mediante TCP/IP sobre ethernet.*

Palabras Clave: Arquitecturas abiertas de control, robots industriales, sistemas de fabricación.

1 INTRODUCCIÓN

El sector de la robótica, al contrario que el de la máquina herramienta y otros segmentos del equipamiento automático industrial, no ha adoptado ningún estándar significativo en el área del control, por lo que cada fabricante ofrece su propia tecnología propietaria, dificultando la integración de los robots en los entornos de producción y la incorporación de avances tecnológicos en equipos ya instalados[1].

Un robot industrial está diseñado para soportar una larga vida de trabajo. Sin embargo, los controladores quedan obsoletos más rápidamente que los robots que controlan. Cuando un fabricante lanza al mercado una nueva generación de controladores, las mejoras e innovaciones del último modelo no suelen estar disponibles para las versiones previas. Incluso es frecuente que las mejoras introducidas en el lenguaje de programación no puedan ser utilizadas en sistemas ya en funcionamiento. Incluso, la disponibilidad de recambios en tarjetas y componentes electrónicos por parte del fabricante suele limitarse a un periodo de tiempo mucho menor que la vida útil del equipo. Estos factores dificultan enormemente incorporar los avances tecnológicos en equipos ya instalados y que han supuesto, habitualmente, una fuerte inversión económica.

El uso de un ordenador (PC) como plataforma permite implantar una arquitectura abierta de control. Las principales ventajas son la posibilidad de utilizar un sistema operativo de amplia difusión (MS Windows, UNIX) con acceso a una enorme base de aplicaciones software, la fácil integración del PC en una red de comunicaciones (con acceso a recursos locales o remotos) y la disponibilidad de gran número de entornos de desarrollo de aplicaciones. Uno de los argumentos más importantes para utilizar un control basado en PC es el bajo coste del hardware y la amplia oferta de todo tipo de productos por parte de diferentes proveedores. Finalmente, con el soporte adecuado, un PC puede ser utilizado en aplicaciones de tiempo real[2].

En este trabajo se presenta la implantación de una estructura de control abierta sobre plataforma PC, basada en la propuesta OMAC, para el control de un

robot industrial de tipo SCARA. Nuestros principales objetivos, al abordar este desarrollo, han sido los siguientes:

- Profundizar en el conocimiento de una de las propuestas más prometedoras en el campo de los sistemas abiertos para control de equipos industriales, como es OMAC[3].
- Evaluar las dificultades que surgen al sustituir un controlador industrial propietario y cerrado por una plataforma abierta basada en PC.
- Contrastar la viabilidad de realizar el control de un robot industrial mediante un controlador abierto basado en OMAC. Al igual que ocurre con otras iniciativas similares (OSACA[4], OSEC[5]) las implantaciones existentes se han realizado, sobre todo, en el campo de la máquina herramienta. En el caso concreto de OMAC, no tenemos conocimiento de su aplicación en el campo de la robótica.

Queremos resaltar que el desarrollo que se presenta se ha realizado sobre un robot industrial de serie (YAMAHA YK7000 [6][7]) y no sobre un equipo didáctico o de laboratorio, con la pretensión de que los problemas abordados y los resultados obtenidos sean un reflejo, lo más fiel posible, de situaciones posibles en un entorno industrial.

Por otra parte, se ha mantenido la funcionalidad completa del controlador original al existir la posibilidad de su uso con la simple conexión de la manguera del robot al armario original. Esto nos permite realizar estudios comparativos entre ambas configuraciones.

2 OMAC

2.1 INTRODUCCIÓN

Debido a los problemas existentes por la abundancia de sistemas propietarios en control de robots y máquinas herramientas, se están llevando a cabo diversas iniciativas internacionales para el estudio y desarrollo de arquitecturas de control abiertas. Entre ellas destacamos tres:

- OSEC *(Open System Environment for Controllers).*
- OSACA (*Open System Architecture for Controls within Automation Systems*).
- OMAC *(Open Modular Architecture Controller)..*

OSEC es una iniciativa japonesa desarrollada para PC-Intel/Windows. Esta es la razón por la cual no hemos trabajado sobre ella ya que limita el número de plataformas en las que se puede aplicar esta arquitectura además de nuestro interés de trabajar en tiempo real usando RT-LINUX.

Sobre OSACA se ha trabajado en proyectos anteriores, y ha sido desechada pues el código no es todo lo abierto que se deseaba.

Las razones principales de la elección de OMAC son:

- Se dispone de una aplicación EMC con el código abierto.
- Es aplicable en distintos sistemas operativos y plataformas entre ellos RT-LINUX.
- Es la más actual de las tres anteriores.

En 1994 los tres grandes fabricantes de coches en EE.UU. publicaron un escrito sobre los requisitos de los futuros controladores abiertos y modulares. OMAC recogió estos requisitos. Actualmente muchas más empresas, instituciones y universidades están involucradas en este proyecto y no sólo de los EE.UU.

2.2 CONCEPTO DE MODULARIDAD DE OMAC

OMAC no ha definido un arquitectura de referencia fija, sino un conjunto de módulos para desarrollar con ellos diferentes tipos de controladores. Estos módulos están especificados en IDL (Interface Definition Language).

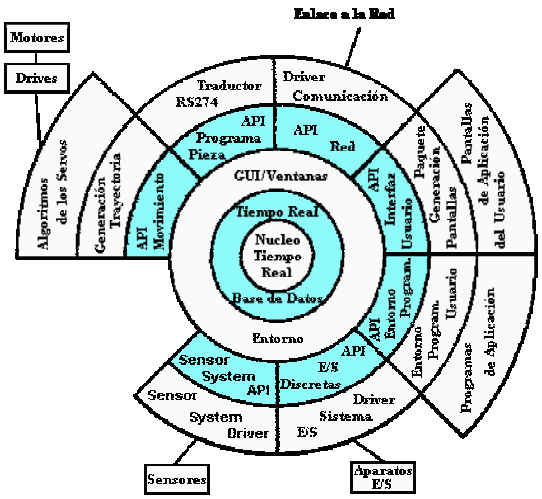

Figura 1: El concepto de modularidad de OMAC

Podemos definir la modularidad de OMAC (Figura 1) como la división de las funcionalidades del controlador. Gracias a esta propiedad determinadas partes de la arquitectura pueden ser adaptadas a nuevas necesidades. Estos módulos actúan como cajas negras pero siempre deben respetar las API´s que permiten la cooperación con los demás.

La modularidad nos garantiza las características propias de un controlador abierto como son la interoperatividad, escalabilidad y portabilidad.

3 DESCRIPCIÓN DEL SISTEMA

En este apartado se va a describir físicamente el robot que se pretende controlar. El robot YAMAYA YK7000 es un robot de tipo SCARA (*Selective Compliance Arm Robot for Assembly*), dedicado a la manipulación de piezas.

Este robot posee cuatro grados de libertad (Figura 2), de los cuales los dos primeros (ejes X e Y) son de tipo rotacional, siendo el tercero (eje Z) de tipo prismático, estos tres primeros ejes del robot permiten posicionar el efector final, que es este caso consiste en una pinza neumática, mientras que el cuarto (eje R) permite orientar dicho efector final, mediante un movimiento de rotación paralelo a los dos primeros. En la figura se muestra el esquema cinemático de este robot.

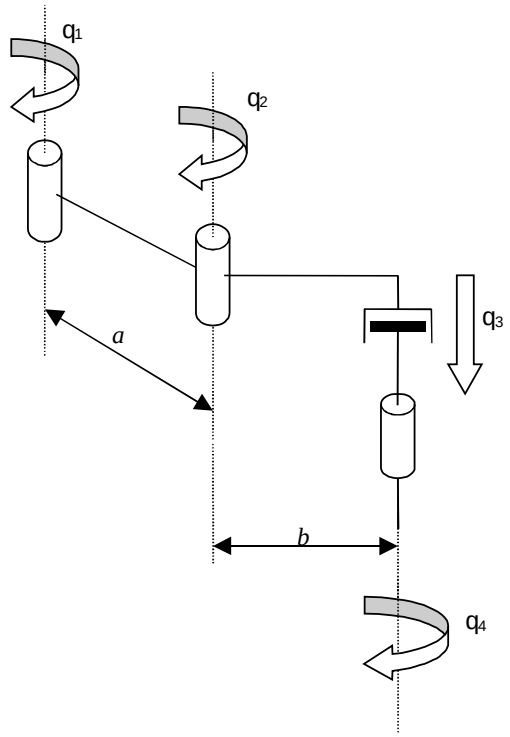

Figura 2: Esquema cinemático del robot SCARA.

El robot es de accionamiento eléctrico, a través de cuatro servomotores de corriente continua, dos de ellos de 350 W y otros dos de 80 W. El accionamiento de la pinza es de tipo neumático. La señal de posición está realimentada a través de encoders incrementales en cuadratura, de tipo óptico, situados sobre los ejes de los motores del robot.

4 SISTEMA IMPLANTADO

Originalmente partíamos de una estructura propietaria como era el robot YAMAHA y su armario de control inicial donde estaban incluidos los elementos de control (tarjeta de control de ejes), de potencia (alimentación y tarjetas variadoras) y de seguridad (lógica hardware para la gestión de fallos graves del sistema).

Para realizar nuestra nueva arquitectura abierta debimos eliminar ese armario original y sustituir cada una de sus funcionalidades por nuevos elementos consiguiendo así la apertura de nuestro sistema. En un principio se intentó aprovechar algunos elementos originales del robot como son las tarjetas variadoras, pero la falta de información sobre estas llevo a abandonar este intento y plantear la sustitución de todo el armario de control y potencia original.

Como se ha indicado el objeto del presente trabajo es el control mediante PC dentro de una arquitectura abierta de control de un robot, para lo que se ha utilizado una tarjeta de control de ejes, que recibe las instrucciones de la aplicación de control y genera los perfiles de velocidad para cada eje del robot como una señal, que será enviada a la etapa de potencia del robot. Se ha diseñado una nueva etapa de potencia (tarjetas variadoras) para la alimentación de los motores y una serie de elementos auxiliares para la coordinación de los diferentes elementos.

La solución desarrollada se basa en una tarjeta de control de ejes con un DSP

La tarjeta de control de ejes se encuentra conectada al bus del PC. Se adquirió la tarjeta ARCS Lightning DSP Board. Cabe destacar las siguientes características de la tarjeta:

- Control de cuatro ejes independientes.
- Uso de encoders incrementales en cuadratura
- Señales de referencia de velocidad analógicas entre ±10 V.
- Ocho canales analógicos-digitales.
- Cuatro canales digitales-analógicos.
- 32 entradas/salidas digitales.
- Funciona con un DSP TMS320C31 con un frecuencia de reloj de 40 MHz.

La etapa de potencia envía a los motores la tensión modulada a través de un PWM. Esta nueva etapa consistirá en dos tarjetas variadoras, concretamente los modelos LEAG DC Servodriver TFM060 y TFM080, alimentadas por un rectificador trifásico de puente de diodos que generará 75 V. de CC a partir de 53 V de corriente alterna trifásica.
Al carecer el robot de tacómetros que nos impedía el cierre del lazo de velocidad se hizo uso del circuito

de compensación IxR que determina la velocidad de giro del motor a través de la intensidad que circula por él. Este circuito ya venía integrado en las variadoras.

El sistema de control implantado se ha realizado sobre dos PCs que se comunican a través de tarjetas Ethernet, conectadas mediante un cable de par trenzado. El PC encargado de la aplicación de control funciona bajo RT-LINUX y el que soporta el HMI usa como sistema operativo Windows NT.

Adicionalmente se ha implantado un sistema de seguridad, que incorpora entre otros un circuito Watch-dog, a fin de evitar que el robot quede sin control ante fallos en los PCs o en la aplicación de control del robot, mejorando el problema de la robustez que plantea el uso de PCs en aplicaciones industriales. Estos nuevos elementos electrónicos de gestión de fallos graves realizarán las comprobaciones pertinentes para el funcionamiento seguro del robot como son la detección de las señales de emergencia (setas, finales de carrera físicos, detección de cuelgue del PC de control, etc.); además realizarán una etapa de adaptación de las tensiones de las señales entre los diferentes elementos que constituyen nuestro sistema. En la siguiente figura 3 se muestra un diagrama del sistema implantado.

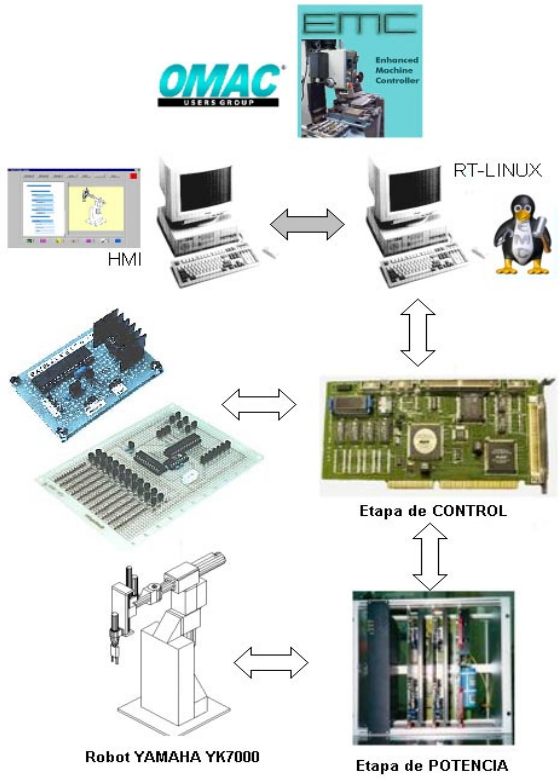

Figura 3: Sistema implantado.

5 APLICACIÓN IMPLANTADA

Se ha utilizado el software EMC[8] (Enhanced Machine Controller) como referencia para nuestra aplicación. Este software está realizado de acuerdo a las especificaciones de OMAC. Se trata de un esfuerzo desarrollado por el NIS[9] (National Institute of Standars and Technology) para el desarrollo y validación de interfaces para la arquitectura abierta de control propuesta por OMAC. Es un código libre y abierto.

El software EMC está basado en sistemas de tiempo real (RCS[10]), y hace uso de las librerías NIST. La librería RCS facilita la portabilidad del código a gran variedad de plataformas desde UNIX hasta Microsoft Windows, de tal forma que nos facilita una interfaz de aplicación (API) para operar en sistemas con recursos tales como memoria compartida, semáforos y temporizadores. La librería RCS implanta un modelo de comunicación el cual permite control de procesos y está desarrollado en C/C++.

La aplicación basada en EMC consta de los siguientes módulos:

- *Interface*: es usado por el operador y manda las órdenes pertinentes al módulo EMCTASK. Para ello se hace uso de la norma RS274NGC[11]. Hemos adaptado alguno de los códigos G y M para aplicaciones específicas del robot como apertura y cierre de pinza, etc... .
- *EMCTASK*: planifica las tareas necesarias en función de los comandos recibidos y manda las ordenes pertinentes a EMCIO y EMCMOT.
- *EMCMOT*: encargado del control de movimiento. En nuestra aplicación se hace uso de un control de posición mediante un PID y los motores controlados son de corriente continua.
- *EMCIO*: se encarga de gestionar las entradas salidas de nuestro sistema, recogiendo todas las señales asociadas a nuestro distintos dispositivos.
- *EMCNML*: encargado de gestionar la comunicación entre los distintos módulos.
- *Intérprete RS274NGC*: encargado de interpretar los códigos basados en esta norma.

Para esta nueva arquitectura de control se realizaron dos aplicaciones de EMC : usando completamente el software EMC y utilizando la tarjeta de control de ejes como una tarjeta de E/S, y en una segunda fase empleando el DSP de la tarjeta para realizar ciertas funciones de control y gestión de entradas y salidas.

5.1 PRIMERA FASE. EMC Y TARJETA DE ADQUISICIÓN DE DATOS.

En nuestro sistema disponemos de dos ordenadores, en uno se encuentra la aplicación de control y en otro

el interface hombre máquina. El interface es usado por el operador y manda las órdenes pertinentes (gestión entradas salidas, mover eje manualmente, etc.) al módulo EMCTASK. Para ello se hace uso de la norma RS274NGC. Hemos adaptado alguno de los códigos G y M (G0 movimiento lineal rápido, G1 movimiento lineal a velocidad programada, M3 apertura pinza, M4 cierre pinza, etc.) para aplicaciones específicas del robot como apertura y cierre de pinza, etc... . EMCTASK se encuentra en el PC donde está la aplicación de control. Planifica las tareas necesarias en función de los comandos recibidos y manda las ordenes pertinentes a EMCIO y EMCMOT. EMCMOT se encarga del control de movimiento. En la aplicación original se hace uso de un control de posición mediante un PID y los motores controlados son de corriente continua, este módulo incorpora la adquisición de datos de los encoders. (ver Figura 4)

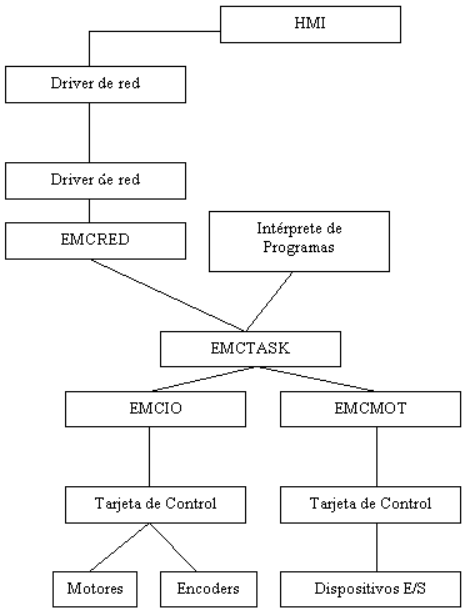

Figura 4: Esquema de la aplicación basada en EMC.

Físicamente la tarjeta de control de ejes adquiere los datos de los encoders y estos son recogidos por EMCMOT a través de las API´s pertinentes. EMCIO se encarga de gestionar las entradas salidas de nuestro sistema, recogiendo todas las señales asociadas a nuestros distintos dispositivos. Nosotros hemos usado la tarjeta de control de ejes como interface de entrada/salida. Nos comunicamos con la tarjeta de control de ejes adaptando las API´s propuestas por OMAC. EMCNML se encarga de gestionar la comunicación entre los distintos módulos. Mediante la función emcserver comunicamos una HMI ubicada en un ordenador remoto con la aplicación de control. La aplicación de control trabaja bajo RT-LINUX, mientras que el HMI se lanza en WINDOWS NT.

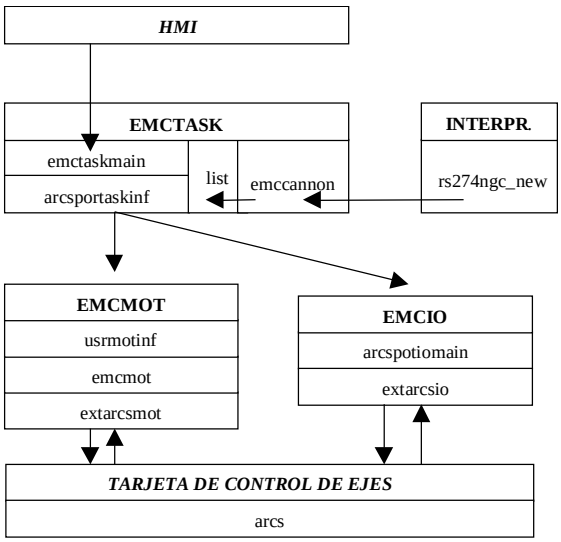

Figura 5: Esquema de la aplicación.

Una idea genérica del funcionamiento es la siguiente:

El lazo de control general de EMCTASK se encuentra en *emctaskmain.cc*; este programa principal llama ciclicamente a *emcTaskPlan*() y *emcTaskExecute*(). *emcTaskPlan*() lee el nuevo comando proveniente del HMI, y decide si esta basado en el modo (manual, auto, mdi) o en el estado (estop, on) de la maquina. Muchos comandos salen inmediatamente a los subsistemas (movimiento y IO). En modo auto, el interprete es llamado y como resultado se añaden a una lista comandos NML. El interprete de comandos llama a las funciones implantadas en emccannon según los comandos G,M .. recibidos que ponen los mensajes NML pertinentes en interp_list. *emcTaskExecute*() ejecuta un switch de *execState*. Si se realiza, obtendremos el siguiente elemento de la lista y configura *execState* para la precondiciones que necesite. Estas precondiciones incluyen la espera para un movimiento, espera de IO, etc... Una vez satisfechas, se edita el comando, y se configura *execState* a las postcondiciones. Una vez finalizado, se obtiene el siguiente elemento de la lista, y se hace lo mismo. En *arcsporttaskinf.cc* se encuentran la implantación de las funciones EMCNML que envían los comandos pertinentes desde el nivel de tareas EMCTASK hacia el nivel de movimiento EMCMOT y de entradas salidas EMCIO en función de las órdenes recibidas.

En el nivel EMCIO se encuentra *arcspotiomain.cc* que lee los mensajes NML y llama a los controladores específicos asociados para cada dispositivo entrada/salida como puede ser el controlador de la pinza del robot. Estos controladores

a su vez llaman a las funciones de la tarjeta de control de ejes para realizar las acciones pertinentes a través de las funciones implantadas en *extarcsio*.

En el modulo EMC MOT se encuentraN *emcmot.c* y *emcmot.h* que contienen las definiciones y declaraciones para el controlador de movimiento. Los comandos y las estructuras de datos son puestas en un buffer de memoria compartida, esto no se basa en NML. Los comandos emcmot son enviados desde el nivel de tareas. usrmotinf manipula la interfaz entre EMCMOT y su supervisor EMCTASK. Existen dos partes para los comandos de movimiento: por un lado la planificación de trayectorias y por otro el control de ejes. Estas dos partes funcionan, en una primera versión, como un único ejecutable donde el ciclo de planificación de trayectorias es llamado cada n ciclos del nivel de control de ejes. Existe un parámetro que permite variar esta relación. El ciclo de movimiento se inicia capturando la poisición de los ejes mediante la función *extEncoderReadAll*(). Se realiza la planificación y control de ejes y las tareas de movimiento permanecen inactivas hasta que una interrupción las vuelve a poner en movimiento. Existen dos modos de movimiento: un modo libre ,en el que cada eje opera de un modo independiente de los otros, correspondiente al modo manual del nivel de tareas y un modo coordinado correspondiente al modo auto, MDI del nivel de tareas. A través de las funciones implementadas en extarcsmot se produce la comunicación con la tarjeta.

La comunicación está soportada a partir de NML. Los mensajes NML son enviados desde la GUI hacia EMCTASK, y desde este hacia EMCMOT y EMCIO, y son puestos en *Interp_list* por la interface. En *canon.hh*, dentro del módulo EMCNML se encuentran las funciones llamadas por el intérprete de código, incluyendo funciones como *straight_feed*() que el intérprete llama cuando ve un G1. En *ec.hh* se encuentra la declaración para todas las clases y comandos NML y en *emc.cc* se encuentra la definición para las clases declaradas en *emc.hh.*

5.1 SEGUNDA FASE. EMC Y TARJETA DE CONTROL DE EJES.

Una vez desarrollada esta aplicación la modificamos. Usamos el DSP que incorpora la tarjeta de control de ejes. Este DSP ahora realiza funciones propias del bloque EMCMOT y gestiona las E/S de nuestro sistema (figura 6).

Este es el bloque EMCMOT usado en nuestro primer paso. Ahora realizamos las operaciones de interpolación, cinemática,... dentro de la tarjeta de control de ejes mediante una aplicación que se carga en su DSP.

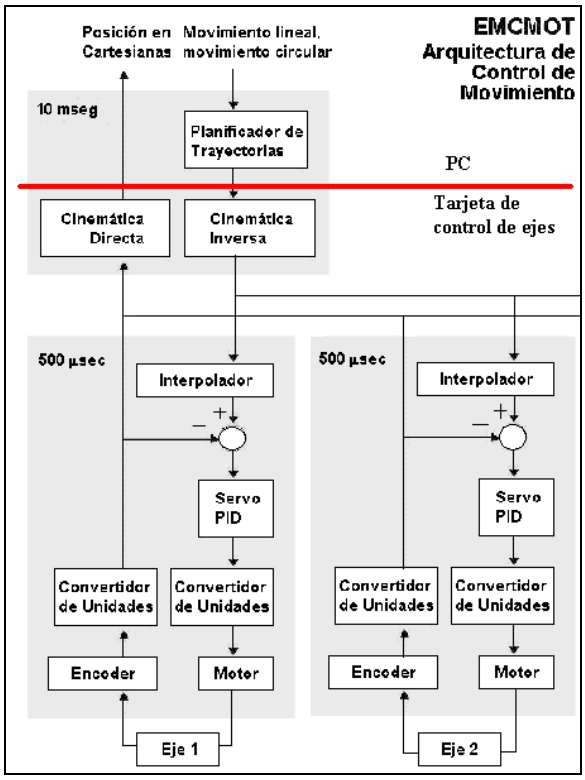

Figura 6: Control de movimiento (EMCMOT).

Las E/S son realizadas haciendo uso de las interrupciones de nuestra tarjeta de control de ejes. El software EMC es más flexible que el de la tarjeta de control de ejes, por ello el planificador de tareas no se realiza en el DSP de la tarjeta de control de ejes sino en el PC, de esta forma podemos desarrollar nuevas estrategias para el planificador de trayectorias.

Esta segunda forma de trabajar nos ha permitido quitar carga computacional en el PC, al pasar funciones del control de movimiento al DSP de la tarjeta de control de ejes. De esta forma los tiempos empleados en el control de nuestros ejes han disminuido apreciablemente.

6 CONCLUSIONES

En el presente trabajo se describe la implantación de un controlador abierto para un robot industrial de tipo SCARA basado en PC, mediante la utilización de una tarjeta de control de ejes y el desarrollo de un conjunto de aplicaciones software basadas en la metodología OMAC/EMC.

El control de un robot industrial mediante PC, basado en una arquitectura abierta, reduce de una forma significativa los costes del equipo. Permite actualizar el controlador, sin tener que sustituir el robot existente. Permite modificar el hardware de control

(por ejemplo, una tarjeta de control de ejes), con cambios mínimos en el software de control. Además, debido a la modularidad del sistema, si se desea modificar el programa de control, sólo habrá que cambiar los módulos que se vean afectados, y se podrán seguir utilizando los demás.

Por contra, un controlador basado en PC es menos robusto y tolerante a fallos que un controlador comercial. Esto obliga a incorporar toda una serie de medidas adicionales de supervisión y seguridad (hardware y software) para garantizar un funcionamiento que sea, como mínimo, tan fiable como el que proporciona el equipo original.

Hemos partido de una arquitectura cerrada y propietaria, y hemos realizado la apertura de la arquitectura. Esto nos ha permitido reutilizar un sistema robótico que poseía un controlador obsoleto. Además la apertura de la arquitectura nos va a permitir en el futuro realizar nuevas modificaciones en la arquitectura, actualizaciones del controlador, de los elementos que componen el sistema, etc.. con un mínimo numero de cambios en la estructura actual del controlador. La arquitectura actual es flexible y adaptable a nuevas necesidades.

Referencias

[1] W. Weisel, "*The State of the Art in Robot Control Solutions. Taking Advantage of the PC Platform*", Robotics World, Mayo-Junio 1999, pp. 38-43

[2] G. Pristschow, T. L. Tran, J. Hohenadel. "Standalone PC-Controller on an open platform", Institute of Control Technology for Machine Tools and Manufacturing Units, University of Stuttgart

[3] www.omac.org, Página web de la organización OMAC

[4] www.osaca.org, Página web de la organización OSACA

[5] www.sml.co.jp/osec, Página web del consorcio OSEC

[6] Yamaha Motor Co.Ltd. Yamaha Robot, YK7000 series. User's Manual.

[7] www.yamaharobotics.com/ Robots Yamaha

[8] www.linuxcnc.org, Página web de EMC

[9] www.nist.org, Página web del NIST

[10] www.isd.mel.nist.gov/projects/rcslib Real-Time Control Systems Library -- Software y documentación.

[11] www.isd.mel.nist.gov/personnel/kramer/pubs/RS274NGC_3.web/RS274NGC_3TOC.html