DESARROLLO DE UN CONTROLADOR ABIERTO PARA UN ROBOT INDUSTRIAL TIPO SCARA

J.L González Sánchez, E. Baeyens Lázaro, F.Gayubo Rojo,
J. Pérez Turiel, J.C.Fraile Marinero, F.J. García González

Instituto de las Tecnologías Avanzadas de la Producción (ITAP)
Universidad de Valladolid (España)

Resumen: En la última década ha aumentado el interés en la utilización de plataformas hardware y software abiertas como soporte de aplicaciones de control industrial. En éste artículo se muestra el desarrollo de un sistema de control abierto para un robot industrial YAMAHA YK7000, de tipo SCARA, siguiendo la filosofía definida por la arquitectura OMAC (*Open Modular Architecture for Controllers*), 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, encargándose el otro de las funciones de interfaz con el usuario y otras aplicaciones no críticas, comunicándose ambos mediante un enlace TCP/IP.

Palabras Clave: Sistemas controlados por computador, control de robot, robots industriales, sistemas de fabricación, integración.

1. INTRODUCCIÓN

El sector de la robótica, al contrario que el de la máquina herramienta y otros segmentos de 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 (Weisel, 1999).

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 (Pristschow, *et al.*,1997).

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.
- 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, OSEC) 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.

Es importante resaltar que el desarrollo se ha realizado sobre un robot industrial de serie (YAMAHA YK7000, cedido por Fasa-Renault) 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.

2. ARQUITECTURAS ABIERTAS

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 (Pritschow, *et al.*, 1993). Entre ellas destacamos tres:

- OSEC *(Open System Environment for Controllers).* (OSE Consortium, 1995)
- OSACA (*Open System Architecture for Controls within Automation Systems*). (OSACA, 2003)
- OMAC *(Open Modular Architecture Controller).* (OMAC, 2003)

OSEC (OSE Consortium, 1995) es una iniciativa japonesa desarrollada para PC-Intel/Windows. Esta es la razón por la que ha sido inicialmente desechada, ya que limita el número de plataformas en las que se puede aplicar esta arquitectura además del interés de trabajar en tiempo real usando RT-LINUX.

OSACA (Amezaga, *et al.*, 1997) es una iniciativa europea para definir una arquitectura de control abierta, independiente de fabricante, con el objetivo de mejorar la competitividad y flexibilidad de los suministradores y usuarios de sistemas de control. No está limitada a PC-Intel, es soportada por diversos sistemas operativos, y está compuesta por tres áreas principales: el sistema de comunicación, una arquitectura de referencia y el sistema de configuración.

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 y evolucionada de las tres anteriores.

En 1994 los tres grandes fabricantes de coches en EE.UU. publicaron un documento (Chrysler, et al. 1994) 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.

OMAC no define una 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). Gracias a esta modularidad determinadas partes de la arquitectura pueden ser adaptadas a nuevas necesidades. Los módulos actúan como cajas negras pero siempre deben respetar las interfaces de aplicaciones (API´s) que permiten la cooperación con otros módulos. La modularidad nos garantiza las características propias de un controlador abierto como son la interoperatividad, escalabilidad y portabilidad.

3. DESCRIPCIÓN DEL ROBOT

El robot YAMAYA YK7000 es un robot de tipo SCARA (*Selective Compliance Arm Robot for Assembly*), dedicado a la manipulación de piezas. Posee cuatro grados de libertad, de los cuales los dos primeros (ejes X e Y) son de tipo rotacional, siendo el tercero (eje Z) de tipo prismático. Los tres primeros ejes permiten posicionar el efector final, que es este caso consiste en una pinza neumática, mientras que el cuarto (eje R) realiza la orientación del efector final, mediante un movimiento de rotación paralelo a los dos primeros. En la figura 1 se muestra el esquema cinemático de este robot.

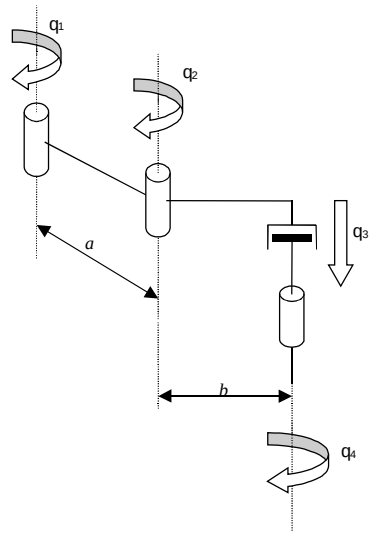

Figura 1. Esquema cinemático del robot SCARA

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 codificadores incrementales en cuadratura, de tipo óptico, situados sobre los ejes de los motores del robot.

4. ELEMENTOS HARDWARE

El robot YAMAHA disponía originalmente de un armario de control con estructura propietaria que incluía 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).

Dada la inoperatividad del controlador original fue necesario sustituir todo el armario de control por nuevos elementos que permitiesen desarrollar sus funcionalidades según una arquitectura abierta.

La solución desarrollada se basa en una tarjeta de control de ejes con un DSP. Esta tarjeta se encuentra conectada al bus del PC. Las características de la tarjeta ARCS Lightning DSP Board elegida para el proyecto son:

- 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 tarjeta de control de ejes recibe las instrucciones de la aplicación de control y genera los perfiles de velocidad para cada eje del robot, que se envía como señal analógica a la etapa de potencia del robot.

La etapa de potencia utiliza un PWM para generar la tensión modulada que alimenta a los motores. Esta nueva etapa consiste en dos tarjetas variadoras, LEAG DC Servodriver, modelos TFM060 y TFM080, alimentadas por un rectificador trifásico de puente de diodos que genera 75 V. de tensión continua a partir de 53 V de tensión alterna trifásica

Al carecer el robot de tacómetros para cerrar el lazo de velocidad, se ha usado el circuito de compensación IxR que determina la velocidad de giro del motor a través de la intensidad que circula por él.

Adicionalmente se ha implantado un sistema de seguridad, que incorpora, entre otros, un circuito *watch-dog*, que evita que el robot quede sin control ante fallos en los PCs o en la aplicación de control del robot. Esto mejora el problema de robustez que plantea el uso de PCs en aplicaciones industriales. Los nuevos elementos electrónicos de gestión de fallos graves realizan las comprobaciones pertinentes para el funcionamiento seguro del robot como son la detección de las señales de emergencia. Además se encarga de la adaptación de señales entre los diferentes elementos que constituyen el sistema. En la figura 2 se muestra un diagrama del sistema implantado.

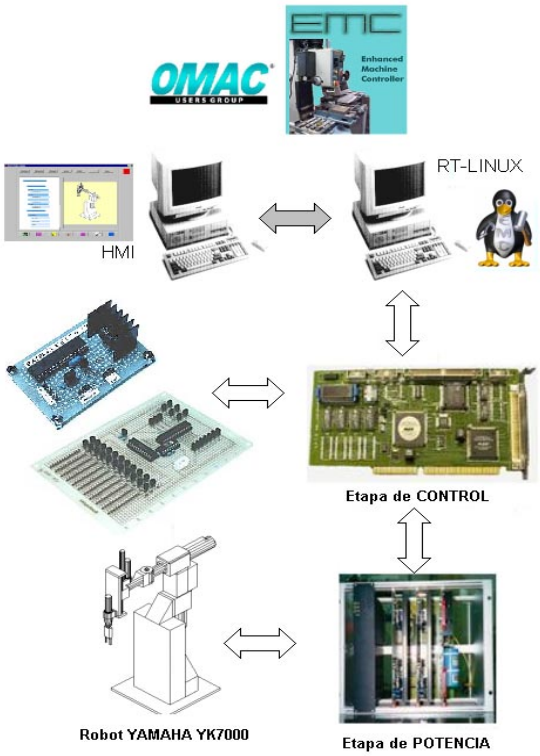

Figura 2. Arquitectura hardware implantada.

5. SOFTWARE DE CONTROL

Se ha optado por una plataforma dual para separar la parte de control directo del robot de la interfaz con el usuario. Este tipo de implantación ofrece dos ventajas. Por un lado, el software de control tiene mayores requisitos de tiempo y cálculo, evitando cargar la CPU con operaciones de tipo gráfico, manejo de ficheros, etc. Por otra parte, como el operario sólo tiene que manejar el PC que hace de interfaz, disminuye el riesgo de fallo en el PC de control. Los dos PCs se comunican a través de un enlace TCP/IP.

El PC encargado de la aplicación de control funciona bajo RT-LINUX, mientras que el que soporta el HMI usa como sistema operativo Windows NT, permitiendo que el operario pueda gobernar el sistema, además de monitorizar los datos necesarios para su control.

Se ha utilizado el software EMC (Enhanced Machine Controller) (LinuxCNC, 2001) como referencia para la aplicación. Este software está realizado de acuerdo a las especificaciones de OMAC. Se trata de un esfuerzo desarrollado por la agencia americana NIST (National Institute of Standards and Technology) para el desarrollo y validación de interfases 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, Real-Time Control System (NIST, 2003), y hace uso de las bibliotecas NIST. La biblioteca 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 biblioteca RCS implanta un modelo de comunicación el cual permite control de procesos y está desarrollado en C/C++.

Los componentes del software EMC son: un controlador de movimiento (EMCMOT), un controlador discreto de Entradas/Salidas (EMCIO), un ejecutor de tareas que realiza la coordinación de las mismas (EMCTASK) y una colección de interfaces gráficos.

Para la nueva arquitectura de control se han realizado dos aplicaciones de EMC: la primera utiliza la tarjeta de control de ejes como una tarjeta de E/S, y la segunda emplea el DSP para realizar funciones de control y gestión de entradas y salidas.

5.1 EMC y tarjeta de adquisición de datos.

La interfaz hombre máquina (HMI) se ejecuta en WINDOWS NT, mientras que las tareas de control lo hacen bajo RT-LINUX.

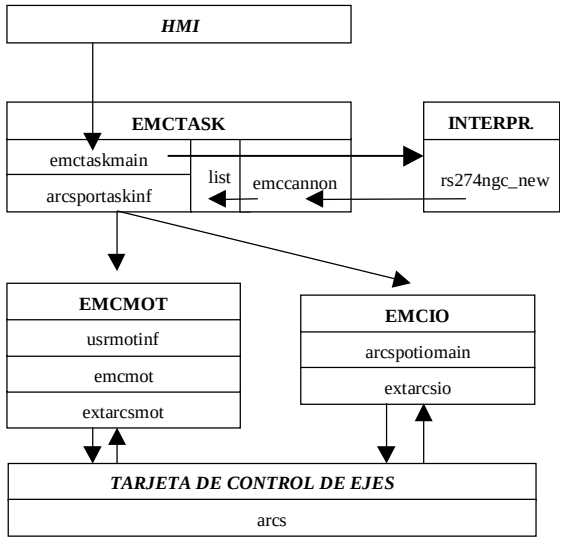

Figura 3. Esquema de la aplicación

La figura 3 muestra un esquema de la aplicación desarrollada. Básicamente, su funcionamiento es el siguiente:

- Comunicación (*EMCNML*)

La comunicación entre los distintos módulos de la aplicación EMC está soportada a partir de NML (*Neutral Message Language*) (Shackleford, et al. 2000). Los mensajes NML son enviados desde el HMI hacia EMCTASK, y desde este hacia EMCMOT y EMCIO, y son puestos por la interfaz en la lista de comandos intrepretados.

- Administrador de Tareas (*EMCTASK*)

Planifica las tareas necesarias en función de los comandos recibidos y manda las ordenes pertinentes a los módulos *EMCIO* y *EMCMOT*. Para ello, se encarga de interpretar los programas escritos en códigos G y M. Este manejador interpreta código basado en la norma RS274/NGC, un estándar de programación de maquinas herramientas.

El lazo de control general de EMCTASK consiste en una ejecución cíclica de la siguientes funciones:

- *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 E/S). En modo *auto*, se pasan los comandos al interprete RS274/NGC y, como resultado, se añaden a una lista de mensajes NML para su posterior ejecución.
- *emcTaskExecute*() obtiene el siguiente elemento de la lista y configura el estado de ejecución para la precondiciones que necesite. Estas precondiciones incluyen la espera por un movimiento, espera de E/S, etc. Una vez satisfechas, emite la orden y se configura el estado a las postcondiciones correspondientes. Una vez satisfechas las postcondiciones, se obtiene el siguiente elemento de la lista, y se hace lo mismo.

- Controlador de entradas/salidas discretas (EMCIO)

Se encarga de gestionar las entradas salidas de nuestro sistema, recogiendo todas las señales asociadas a nuestro distintos dispositivos. Los controladores de E/S discretos son altamente específicos para cada máquina y en general no se pueden adecuar a la técnica genérica usada en el controlador de movimientos. Por tanto, EMC usa EMCIO de puente entre el PC y la máquina sin alterar el código fuente del núcleo de EMC. En el nivel EMCIO se leen los mensajes NML y se 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 el driver.

- Controlador de movimiento (EMCMOT)

Es el módulo encargado del control de movimiento. La aplicación hace uso de un control de posición mediante un PID para el control de los motores CC.

El control de movimiento incluye el muestreo de la posición de los ejes a ser controlados, el cómputo del siguiente punto de la trayectoria, interpolación entre estos puntos de la trayectoria y cómputo de salida hacia los motores. Para servo sistemas, la salida está basada en una compensación PID.

El control de movimiento incluye límites de software programables, interfaces al límite de hardware, compensación del servo PID, error de seguimiento máximo, velocidad y aceleración seleccionables.

El controlador de movimiento está concebido para que sea genérico. Se facilita una aplicación programada en C de forma que pueda ser integrado dentro de EMC sin modificar ninguna parte central del código cuando se modifica el hardware específico.

- Interfaz gráfico (HMI)

Es usado por el operador y manda las órdenes pertinentes al módulo EMCTASK. Para ello se hace uso de la norma RS274NGC (Kramer, *et al.*, 2000). Se ha adaptado alguno de los códigos G y M para aplicaciones específicas del robot como apertura y cierre de pinza, etc. Más adelante se comentarán sus características.

Una vez que se dispone de un driver de control de las tarjetas de entradas-salidas e instalado EMC bajo la versión de RT-LINUX que la soporta (en nuestro caso la versión 2.3), se necesita configurar los ficheros de inicio con los archivos que se leen en el arranque ya que ningún controlador real usará los valores por defecto, por lo que hay que modificar los parámetros de inicio para adaptarlos a las particularidades del sistema bajo el que se instala EMC. Estos archivos de configuración son:

- *emc.ini* contiene todos los parámetros de la máquina tales como descripción de las señales, factores de posicionamiento, duraciones de ciclo, unidades, número de ejes y características de los ejes, etc. Necesitará ser corregido para ajustarlo al sistema.
- *emc.nml* contiene la configuración para la comunicación para la memoria compartida y los puertos de la red que se necesitan, es muy probable que este fichero no necesite ser modificado.
- *tool.tbl* solo contiene la información de las herramientas, longitud y diámetro de cada una y posición en el carrusel de intercambio de herramientas.
- *rs274ngc.var* contiene variables específicas del control numérico del lenguaje,

5.2 *Segunda fase: EMC y tarjeta de* control de ejes.

En esta segunda fase se ha usado el DSP que incorpora la tarjeta de control de ejes. Este DSP realiza ahora funciones propias del bloque EMCMOT (interpolación, cinemática, ...) y gestiona las E/S de nuestro sistema (figura 4). De esta forma los tiempos empleados en el control de los ejes han disminuido apreciablemente.

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 sino en el PC, de esta forma se pueden desarrollar nuevas estrategias para el planificador de trayectorias.

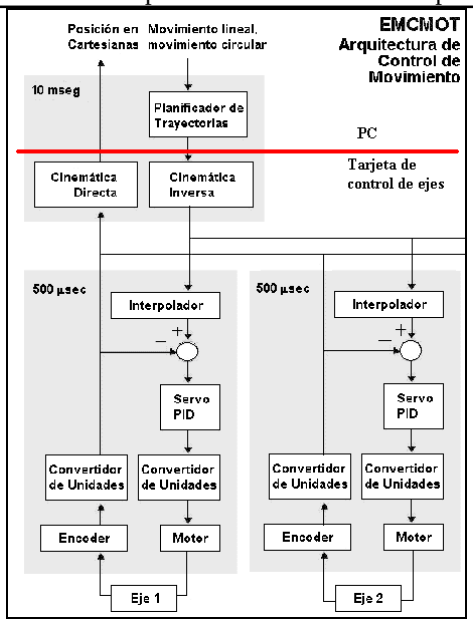

Figura 4 Control de movimiento (EMCMOT).

6. INTERFAZ HOMBRE-MÁQUINA (HMI)

La Interfaz Hombre-Máquina (HMI), se encarga de la relación del sistema de control con el usuario. El usuario puede iniciar el sistema, reconfigurarlo, entrar en modo manual, automático, etc. También permite la visualización de la posición del robot, así como el estado del sistema OSACA. No tiene interfaz de proceso.

Para iniciar la aplicación de control se han de seguir los siguientes pasos:

- Ejecutar el programa controlador *emc.run* en el PC de control.
- Ejecutar el programa controlador *Hmi.exe* en el PC HMI (Figura 5). Mediante la opción Iniciar comenzará la ejecución de los programas de control en el PC de control.

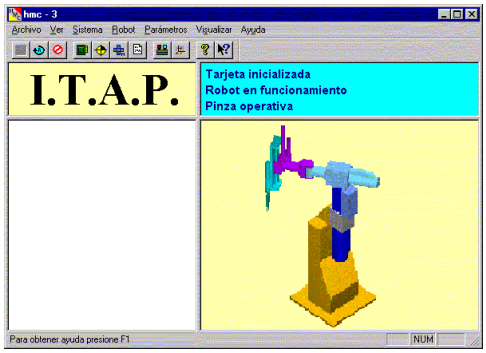

Figura 5: HMI - Interfaz de usuario

- A partir de ahora, el usuario no necesita utilizar el ordenador dedicado al control directo del robot, y podrá manejar el sistema desde el HMI.
- Una vez que haya terminado de utilizar el sistema, el usuario debe parar su ejecución y salir del programa HMI.

Las opciones de la aplicación HMI, agrupadas en los distintos menús disponibles, son las siguientes:

- **Menú Archivo :** Permite cambiar o editar los ficheros de configuración, de programa y de puntos del robot.
- **Menú Sistema :** Sirve para controlar el sistema de control. Permite iniciar el sistema, reconfigurarlo o pararlo.
- **Menú Robot :** En este menú se puede elegir entre modo control o modo simulación (para verificar un programa sin mover el robot). Permite inicializar la tarjeta de control de ejes, y verificar el funcionamiento de los motores del robot, así como buscar la posición de origen del robot. Es posible elegir entre control manual o automático (modo programa) (Figura 6). En esta caso, la ejecución puede ser paso a paso (instrucción a instrucción) o bien un programa completo. También permite ejecutar la parada del robot en cualquier momento.

Figura 6: Operación en Modo programa

- **Menú Parámetros :** Sirve para visualizar los ficheros en uso por el programa, las características de los Objetos de Arquitectura y los parámetros de la tarjeta y del robot.
- **Menú Visualizar :** Permite la monitorización del sistema de control, mostrando los estados de cada módulo. También ofrece la monitorización de las posiciones de los ejes del robot, tanto en coordenadas articulares como en cartesianas, así como el estado de la pinza.

7. 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.

Se ha partido de una arquitectura cerrada y propietaria, y hemos realizado la apertura de la arquitectura. Esto 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, reconfigurable y adaptable a nuevas necesidades.

REFERENCIAS

Amezaga J., Barg J., Brühl J., Lutz P., Nacsa J., Pohl M., Sozzi M., Wälde K., Zulauf P. (1997). OSACA Handbook, *OSACA Association*, Stuttgart, Germany.

Chrysler, Ford Motor and General Motors. (1994). Requirements of Open, Modular Architecture Controllers for Applications in the automotive Industry. White Paper. Ver.1.1.

Kramer T.R., Proctor F., Messina E.. (2000). The NIST RS274/NGC Interpreter - Version 3. *www.isd.mel.nist.gov/personnel/kramer/pubs/RS274NGC_3.pdf*

LinuxCNC. (2001) EMC Handbook *www.linuxcnc.org/handbook*

NIST (2003), Real-Time Control Systems Library. Software and documentation. *www.isd.mel.nist.gov/projects/rcslib*

OMAC (2003). Página web del OMAC Users Group. *www.omac.org*

OMAC Working Group (1999), *OMAC API v.0.2*3. www.isd.cme.nist.gov/projects/teamapi.

OSACA (2003). Página web del proyecto OSACA. *www.osaca.org*

OSE Consortium (1995), OSEC: Open System Environment for Controller. Architecture Draft, *osec.skk-inc.co.jp*

Pritschow G., Tran T. L., Hohenadel J.. (1997) Standalone PC-Controller on an open platform, *Proceedings of the 30th International Symposium on Automotive Technology & Automation*, Florence, Italy.

Pritschow, G.; Daniel, C.; Junghans, G.; Sperling, W. (1993) Open System Controllers - A Challenge for the Future of the Machine Tool Industry. *CIRP Annals*. **Vol. 42/1**, 449-452.

Shackleford W. P., Proctor F. M., Michaloski J. L. (2000). The Neutral Message Language: A Model and Method for Message Passing in Heterogeneous Environments. *Proceedings of the 2000 World Automation Conference*

Weisel W. (1999). The State of the Art in Robot Control Solutions. Taking Advantage of the PC Platform. *Robotics World Magazine*, Douglas Publications, **vol. 17**, 38-43