Integración de equipos industriales para implementar un eje externo en una celda robotizada.

Carlos J. Cartelli and Mauricio Anigstein

Laboratorio de Robótica.
Facultad de Ingeniería, Universidad de Buenos Aires (FIUBA).
{cjcartelli@yahoo.com,manigst@fi.uba.ar}
http://www.fi.uba.ar

Resumen: El artículo presenta la integración de equipamiento industrial en el desarrollo e implementación de un eje externo de tipo mesa rotativa para una celda robotizada que incluye un manipulador típico de seis ejes. Se utilizan las capacidades de comunicación industrial disponibles en los componentes empleados para coordinar movimientos entre el robot y la mesa rotativa. Mediante un módulo de software específicamente programado, el usuario/programador del robot comanda las nuevas prestaciones con nuevos comandos compatibles con el entorno de programación nativo. De esta manera se obtiene un producto simple y robusto que expande la capacidad original de la celda robotizada.

Palabras Clave: Robot, eje externo, programación de robot industrial, comunicación industrial.

1 Introducción

Si bien los robots manipuladores de seis ejes pueden posicionar y orientar la herramienta manipulada dentro de su espacio alcanzable, la capacidad de movimientos está normalmente limitada por varios factores relacionados con:

- Su estructura cinemática: límites mecánicos de articulaciones, impedimentos de ciertos tipos de movimientos en -o cerca- de singularidades, limitaciones respecto a la carga manipulada.
- La celda que el robot integra: objetos con los que el manipulador podría colisionar, zonas de trabajo inseguras de operación, accesorios de las herramientas que limitan movimientos.

Estas cuestiones originan que la tarea demande simulaciones durante el diseño, necesidad de recurrir a movimientos adicionales, posiciones de trabajo complicadas e incluso herramientas y sujeciones sofisticadas. Además cuando se trabaja sobre celdas parcialmente estructuradas, empleando por ejemplo sensado y algoritmos de corrección de movimientos en línea [1], estas limitaciones no pueden resolverse sino hasta la ejecución del programa y la realización misma de los

movimientos. Cualquiera de estas circunstancias complica la solución, aumenta el tiempo de diseño de la tarea y por consiguiente incrementa el costo del desarrollo.

Una alternativa para simplificar estas dificultades consiste en agregar a la celda ejes externos, por ejemplo una mesa rotativa. Esta incorporación permite:

1. Que el robot pueda trabajar sobre un punto de la pieza a procesar y que simultáneamente se realicen -en ella o fuera de ella- otras operaciones: por ejemplo alimentando otra pieza a la mesa para optimizar un proceso de producción seriada y continua,
2. Lograr movimientos relativos entre la herramienta del robot y la pieza a trabajar que serían inviables usando sólo el manipulador: por ejemplo realizar un cordón de soldadura continuo alrededor de la circunferencia de un caño.

Esta ampliación en la capacidad de posicionamiento se justifica en la redundancia que este nuevo eje aporta a la tarea. Sin embargo, la versatilidad de posicionamiento también implica que el programador tenga que aplicar algún criterio para elegir la estrategia de resolución de redundancia. Estos criterios y estrategias definen y calculan las trayectorias de posicionamiento para cada movimiento, y requieren algoritmos demandantes y complicados corriendo en los controladores. En [2] se planteó esta problemática y se desarrolló una plataforma que permite ejecutar y evaluar diferentes criterios y estrategias de resolución de redundancia.

1.1 Objetivos y organización del artículo

El Laboratorio de Robótica de la FIUBA cuenta con una celda robotizada que incluye un manipulador industrial ABB sin ejes externos, un servomotor con controlador inteligente Siemens, comunicación industrial y Ethernet, sistemas de sensado de variados parámetros en línea, entre otros. En artículos previos [3],[4] se plantearon las etapas iniciales del desarrollo e implementación de un eje externo -tipo mesa rotativa- que se integra a la celda del manipulador. Este artículo presenta la plataforma final y los resultados obtenidos con la misma.

La integración no pretende competir con ejes externos comerciales, sino exponer las ventajas de comunicación industrial genérica para vincular componentes de diferentes fabricantes y tecnologías con un manipulador industrial. Esta necesidad se manifiesta de manera habitual cuando se diseñan celdas de trabajo robotizadas flexibles[5].

El artículo se organiza como sigue. La siguiente sección describe los componentes que integran el sistema de manera básica y plantea los requerimientos para la estructura de control y de comunicación empleadas. La sección 3 presenta la implementación y la funcionalidad provista por cada uno de los componentes de la solución. La sección 4 analiza los tipos de movimientos coordinados entre el eje externo y el manipulador, y las dificultades que surgen en su implementación. Por último en la sección 5 se exponen los resultados obtenidos con la plataforma.

2 Estructura del sistema

2.1 Componentes

- Robot + Controlador: El **robot** es un manipulador antropomorfo ABB IRB140 M2000 y el **controlador** un ABB SC4plus con placa de comunicación PROFIBUS ABB DSQC325A de tipo esclavo. En el controlador se cargan los programas de usuario que se quieren ejecutar en el robot. El lenguaje de programación empleado es nativo de robots ABB y se denomina RAPID [6]. Se trata de un lenguaje interpretado e imperativo de alto nivel, con programación estructurada y comandos para (i) movimiento del robot, (ii) comunicación con equipamiento externo y (iii) cómputo en línea. Se permite la inclusión de código auxiliar a través de módulos que pueden cargarse en línea.
- Motor + Drive : El **motor** es un servomotor sincrónico Siemens 1FK7042-5AF71 con velocidad nominal $3000rpm$ y sensor de posicionamiento resolver acoplado al eje. El **drive** o unidad de control es un Siemens Simovert Masterdrives MC 6SE7011-5EP50 que controla al motor mediante control vectorial de corriente. Posee puertos de comunicación general y programación: I/O análogicos y digitales, serie RS-485 y PROFIBUS tipo esclavo. Su programación se realiza mediante una secuencia paramétrica de comandos que configuran la interconexión de diversos bloques internos, funcionalidades y opciones de control. Si bien se trata de una unidad ya discontinuada, es posible programar parámetros de muy bajo nivel y obtener un diseño flexible de control interno.
- PLC: el **PLC** empleado es un Siemens Simatic S7-300 CPU317T-2DP. Es un PLC modular standard con módulo de comunicación PROFIBUS mapeable y configurable como maestro o esclavo[7]. El código programado corre de manera secuencial y se programa mediante lenguajes gráficos o textuales convencionales para PLCs.
- **Mesa** giratoria horizontal, es una estructura mecánica articulada, desarrollada y construida por alumnos de la materia Robótica como parte de prácticas de diseño mecánico. La velocidad de rotación máxima se fijó en $30rpm$. El acoplamiento mecánico mesa-motor es directo, con una unión sin juego muerto. Esto permite que en las tareas coordinadas se optimicen los valores de repetibilidad de posicionamiento.

La figura 1 muestra la nueva celda de trabajo que incluye el manipulador, el eje externo mesa giratoria construido y el resto del equipamiento descripto en los párrafos precedentes.

2.2 Diseño de control del eje externo

El objetivo del sistema de control para la nueva plataforma es proveer al programador del robot comandos para ejecutar y monitorear movimientos coordinados entre la mesa rotativa y el manipulador. Esos nuevos comandos se incorporan al

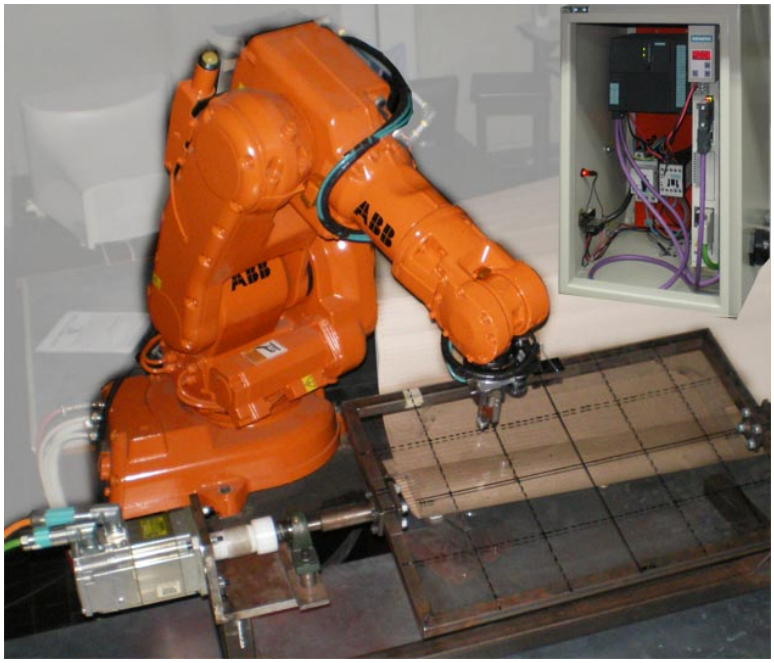

Figura 1. Celda de trabajo con el nuevo eje externo. Arriba der.: el drive y el PLC.

lenguaje nativo RAPID del manipulador a través de módulos y el programador los llama desde el programa del robot. Por lo tanto queda en evidencia que se define como maestro del nuevo sistema al controlador del robot. Pero además de comunicarse y comandar el eje externo, el controlador tiene que continuar con capacidad de responder a requerimientos del resto de la programación y con recursos para cómputo de movimientos del manipulador que no tengan necesariamente relación con el nuevo eje.

Estos requerimientos fundamentan las siguientes decisiones base de diseño:

- Los nuevos comandos son los únicos que el usuario necesita ejecutar para el control del eje externo y dejan ocultos temas particulares de la implementación del control del eje.
- Se programa el sistema interno de control de movimientos del eje externo dentro del drive, y el controlador del robot sólo tiene a cargo la fijación de los objetivos y monitoreo de los resultados. Esto implica que funcionalidades tales como: generación de trayectoria, control de velocidad y posición, cómputo de corriente/torque y monitoreo de estado del eje externo, se resuelven y residen en el drive a disposición para ser comandadas y monitoreadas desde el controlador del robot.
- Los comunicación de comandos de control y monitoreo se concreta en tiempo real para lograr la sincronización de ambos controladores.

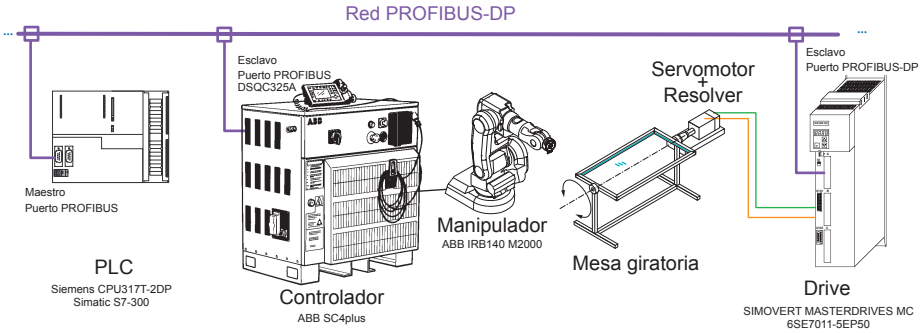

Figura 2. Componentes de la solución dentro de la red PROFIBUS.

3 Implementación

3.1 Comunicación

Para la comunicación entre robot y eje externo se empleó un bus industrial de campo PROFIBUS-DP con medio físico par trenzado apantallado RS-485. Las velocidades de comunicación de nivel de campo de PROFIBUS-DP[8] resultan suficientes en la aplicación planteada. La Fig. 2 muestra la arquitectura de red PROFIBUS implementada.

Incluso cuando la interfaz de red PROFIBUS del controlador del robot sólo acepta funcionalidad esclavo, el robot debe comandar los movimientos coordinados actuando como maestro de control. Para lograr esa capacidad se incorpora el PLC antes descripto configurado como maestro de la red. Se lo programa para que iterativamente

- lea desde el controlador del robot comandos que definen objetivos para el eje externo (SP) y los retrasmita hacia el drive que los interpreta y asigna a sus módulos de control interno.
- comunique en sentido opuesto las variables que monitorean el estado del eje externo hacia el robot (PV).

Estos datos son encapsulados en telegramas PROFIBUS y transferidos en línea a través de la arquitectura bus planteada. De manera simplificada, la intervención del PLC sólo cumple la función ilustrada en en Fig. 3.

3.2 Programación del drive - Unidad de control del motor

Las funcionalidades más importantes respecto al control del eje están contenidas en el drive. El drive permite programar un control a medida de variadas aplicaciones[9]. La *parametrización* o programación de la unidad de control consiste en

- configuración de potencia: define una secuencia de parámetros que informan al drive el tipo de motor empleado y sus características nominales.

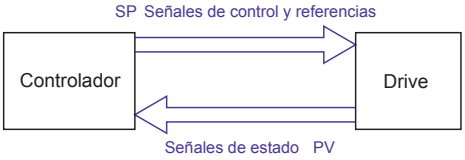

Figura 3. La comunicación lógica del sistema.

- *cableado* o conexión del circuito de control: tal como si se tratara de conexiones eléctricas se arma una estructura de bloques para obtener el circuito de control del motor deseado.
- activación y prioridad de cómputo de bloques: dado que los bloques son implementados por software, y los recursos de procesamiento son finitos, es necesario establecer estas prioridades para asegurar que la interconexión se compute de manera adecuada para la aplicación.
- seteo de señales de control general y acceso a señales de estado de funcionamiento.

La figura 4 muestra de manera simplificada la estructura de control que se parametrizó dentro del drive. La mayoría de los bloques representados son unidades libres que el programador puede disponer a su conveniencia dentro del 'circuito' de control. Esto evidencia la flexibilidad en la solución del control que este tipo de drives entregan al programador. Pero, para asegurar mantenibilidad de la estructura creada, es fundamental generar una documentación de apoyo que la detalle.

La estructura planteada puede dividirse en: interfaz de comunicación, generador de objetivos de control, y control propiamente dicho, que se describen a continuación.

Interfaz de comunicación. Contiene la configuración de la conexión a la red PROFIBUS, y las interfaces de entrada y salida de datos. La *interfaz de entrada* recibe a partir de las variables que el usuario ingresa en un comando RAPID (ver ejemplo en Sec. 3.3) el tipo de movimiento objetivo: modo absoluto o relativo; datos propios del movimiento objetivo: posición, velocidad y aceleración (angulares) y datos referidos al control del motor: encender-apagar, detener, freno de emergencia, etc. La *interfaz de salida* envía al controlador datos de estado de funcionamiento generales (por ejemplo errores de funcionamiento, objetivos cumplidos, etc.) y la posición actual del eje.

Generador de objetivos de control o posicionador básico. Es el encargado de traducir los objetivos que el usuario define para el eje en objetivos válidos de control del motor. El bloque de *setpoint transfer* verifica que los valores del movimiento objetivo estén dentro de rangos válidos y transfiere de manera simultánea esos objetivos al bloque de generación de rampa. El bloque

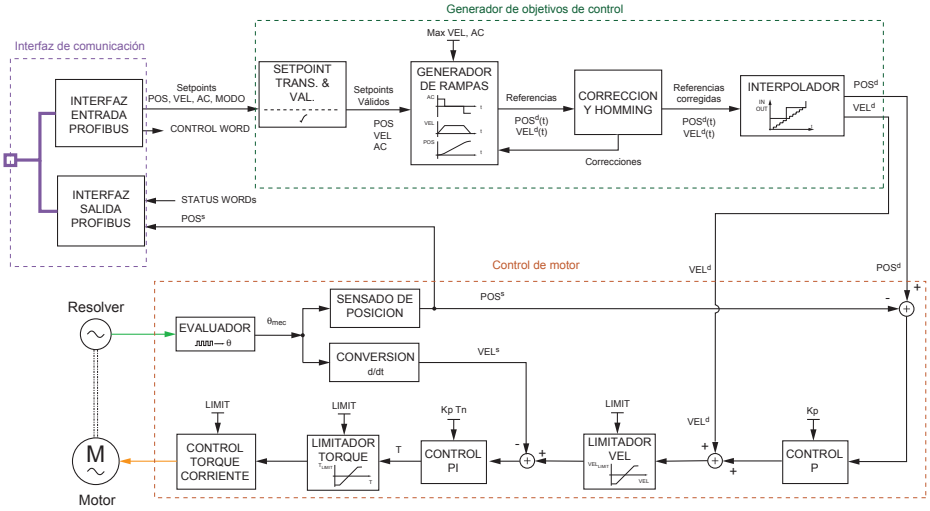

Figura 4. Estructura interna de control programada en el drive.

de *generación de rampa* genera las curvas de variación adecuada de posición y velocidad en el tiempo para llegar a los objetivos requeridos. El bloque de *corrección y homing* aporta corrección adicional para referenciamiento y *homing* (búsqueda de posición inicial de referencia) cuando se incorpora un sensor para este propósito. Las señales de referencia POS^d y VEL^d son ambas procesadas por un bloque *interpolador* que adapta los tiempos de variación de estas señales a los tiempos de procesamiento de los bloques siguientes del lazo de control. Esto es importante porque las variaciones podrían ser demasiado lentas (o rápidas) respecto al tiempo de procesamiento de los bloques de control y producirse saltos (o retardos) indeseados en la actuación. Con esto se 'suavizan' las variaciones deseadas a pesar de los tiempos de procesamiento diferentes.

Control del motor. Se implementa un control en cascada corriente(torque) - velocidad - posición, realimentando valores sensados y con *feed-forward* de la información generada en el posicionador básico. Esta estructura es ampliamente usada en la industria y conveniente por su flexibilidad [10]. Aun para la máxima velocidad de la mesa giratoria ($30rpm$) y debido al acoplamiento directo, el motor opera ampliamente apartado de su velocidad nominal ($3000rpm$). Esto hace que el control del motor sea exigente y su sintonización crítica. Un bloque *evaluador* de datos recibe las señales crudas del resolver y entrega un número θ_{mec} relacionado con el ángulo del eje del motor. El bloque de *sensado de posición* convierte ese valor a ángulo de la mesa POS^s y ajusta la resolución numérica del sensado, definiendo así la resolución máxima de posicionamiento del eje externo. El bloque *conversor* obtiene la estimación de velocidad del eje VEL^s a partir del cómputo de la derivada del ángulo del eje. El *control de posición* es un control proporcional con ganancia K_p ajustable, que realimenta la posición

sensada POS^s. La salida del controlador de posición se suma a la referencia de velocidad para acoplarse en cascada con el controlador de velocidad. El *control de velocidad* es un control proporcional-integral con ganancias ajustables (K_{vp} y T_n tiempo de integración) realimentado por la velocidad sensada VEL^s. Entre los controladores de posición y velocidad se interpuso un *limitador de velocidad* ajustable, fundamental por cuestiones de seguridad. Por tratarse de un lazo con sintonización crítica, ocurre que durante esa etapa pueden producirse efectos de inestabilidad que inducen velocidades objetivo elevadas (y riesgosas). Si el bloque detecta un objetivo excesivo de velocidad genera un mensaje de alerta y limita ese objetivo. La salida del controlador de velocidad es la referencia de torque que previamente es limitada por el *limitador de torque* ajustable. El bloque *controlador de torque/corriente* integra un controlador de flujo, un control vectorial de corriente y un etapa de potencia PWM que finalmente entrega energía al motor. Este bloque permite ajustar parámetros de muy bajo nivel, adaptando el drive a cada uno de los diferentes motores soportados.

3.3 Software en el controlador del manipulador

Dentro del controlador, y sobre el entorno RAPID, se programó un módulo que puede ser cargado en línea denominado `ExtAxisCtrl.mod`, que contiene un conjunto de comandos mínimo para operar toda la funcionalidad del eje externo. Este módulo encapsula también la comunicación bidireccional robot-servo, brindando al usuario comandos de programación de alto nivel[3]. Los datos/comandos resultantes y su funcionalidad son:

- Datos asociados a la mesa: El módulo contiene dos datos de programa con nombres predefinidos que tienen que ser considerados palabras reservadas al cargar el módulo. Estos datos son: `WorkObjectMesa` y `Axis7Angle`.
  - `WorkObjectMesa` contiene las ternas de referencia asociadas a la mesa rotativa y al objeto sobre el que se trabaje montado sobre la mesa (es decir que gira con ella). Para coordinar los movimientos entre el manipulador y la posición actual de la mesa sólo es necesario usar esta referencia en los comandos de movimiento. `WorkObjectMesa` es actualizada automáticamente mediante interrupciones temporizadas a intervalos temporales fijos ($\approx 0{,}25seg$).
  - `Axis7Angle` contiene el ángulo que está rotado el eje respecto a la posición 0° definida durante la calibración. Su valor se actualiza conjuntamente con el de `WorkObjectMesa`.
- Comandos de movimiento y control: Organiza una serie de comandos que permiten al programador ordenar movimientos de la mesa y monitorear los resultados de esas órdenes
  - `EncenderMesa` y `ApagarMesa` provocan respectivamente la activación o desactivación del control de posición del eje externo mesa rotativa.
  - `EstadoMesa` permite al programador verificar si el control de posición de la mesa se encuentra activo o si se presenta alguna falla en el drive.

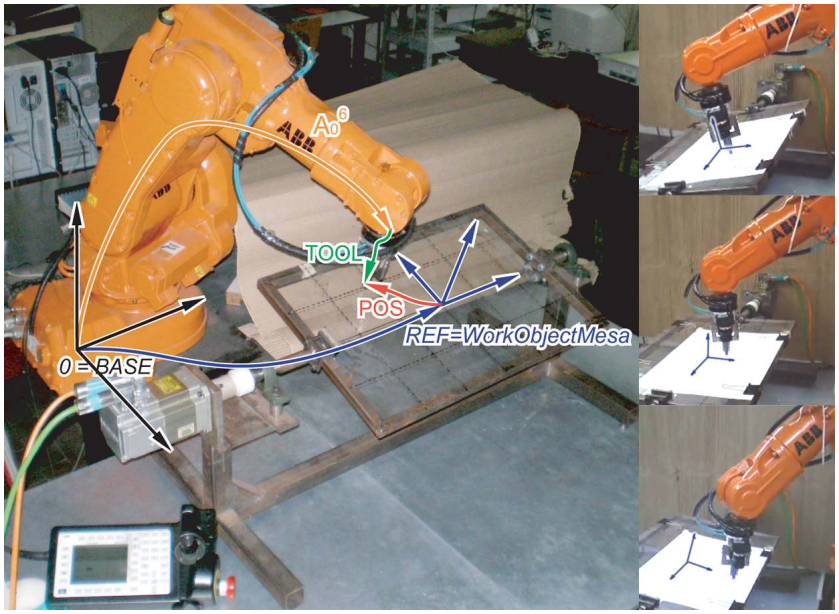

Figura 5. Izquierda: Referencia asociada a la mesa giratoria en relación con el resto de las ternas del robot. A la derecha: Se muestra como la referencia asociada a la mesa gira al girar la mesa en una tarea de movimientos coordinados.

- `MoverMesaAbs` y `MoverMesaRel` son los comandos para ordenar que la mesa se mueva a una posición angular objetivo definida de forma absoluta o relativa respectivamente. Ambos comandos permiten configurar la velocidad o el tiempo, la aceleración y el sentido con que el eje es llevado a la posición objetivo. Además en cada comando puede definirse si el movimiento del eje se realiza de manera simultánea -o no- al movimiento del robot.
- `DetenerMesa` detiene la mesa rotativa utilizando el valor de desaceleración del último movimiento programado.

– Comandos para calibración de mesa, de forma guiada o manual. La calibración es un procedimiento automático que, mediante tres puntos obtenidos por revolución de la mesa, calcula el eje de rotación respecto al sistema de coordenadas base del manipulador. Ese eje define de forma automática la ubicación de `WorkObjectMesa` y el ángulo 0° de la mesa. La calibración permite que el eje externo pueda ser montado en diferentes ubicaciones de la celda y sólo sea necesario calibrarlo en esa nueva ubicación antes de utilizarlo en movimientos coordinados. El la figura 5 se muestra la asociación de `WorkObjectMesa` con la mesa rotativa, resultante de la calibración. Los comandos para la calibración de la mesa son:
  - `WizDefWObjMesa` Inicia la *calibración guiada* de la mesa. La calibración guiada es un método interactivo que asiste al usuario para mostrar al

robot (mediante jogging) los tres puntos de revolución generados por un punto de la mesa definido como punto de calibración.
- `DefWObjMesa` realiza la calibración manual de la mesa. A diferencia de `WizDefWObjMesa`, los tres puntos de revolución son pasados por parámetro y no de manera interactiva. De esta forma es posible una calibración por datos obtenidos desde alguna otra fuente, como por ejemplo del conocimiento de montaje del eje o de datos obtenidos por sensado.

Programa de ejemplo.

```
EncenderMesa; !(a) Enciende y lleva el eje a 0 grados
MoverMesaAbs rax_7, V; !(b)
MoveL rt1,v50,fine,bola \WObj:=WorkObjectMesa; (c)
...
ApagarMesa; (d)
```

En (a) se inicia el control de la mesa y se la lleva a la posición inicial (0°). Posteriormente en (b) se la rota hasta el ángulo `rax_7` a velocidad `V`. Cuando se llega a ese objetivo, en (c) el manipulador mueve la herramienta `bola` a la posición `rt1`, definida en relación a la mesa `WorkObjectMesa`. Finalmente en (d) se apaga el control del eje. Como se observa, los comandos tienen una estructura de parámetros similar a los nativos del lenguaje RAPID (en el ejemplo, `MoveL` es nativo [6]) y permiten al usuario desconocer detalles respecto a implementación de la comunicación y el control del eje.

4 Coordinación de movimientos

Para realizar tareas sobre la mesa giratoria los movimientos del manipulador y del eje externo tienen que estar coordinados. El propósito es poder desarrollar esas tareas sin necesidad de repetir la enseñanza de puntos objetivo cuando la mesa cambia de orientación. Para lograrlo, se asocia la referencia `WorkObjectMesa` fija a la mesa y los movimientos del manipulador se refieren a esa terna. Cuando la mesa gira, para conservar la asociación válida, la referencia se actualiza utilizando el ángulo de rotación del eje. Los movimientos coordinados pueden clasificarse en no simultáneos o simultáneos.

4.1 Movimiento no simultáneo

Cuando el manipulador tiene que realizar una tarea sobre la mesa en reposo, no hay movimiento simultáneo del manipulador y del eje externo. Basta con que cada vez que el eje externo gira, se realice una secuencia de control que consiste en (i) enviar un comando para dar inicio del movimiento del eje con los setpoints (ii) esperar que el drive anuncie la concreción del movimiento, (iii) con la información final de la orientación del eje actualizar la referencia asociada a la mesa (`WorkObjectMesa`). Los errores de posicionamiento robot-mesa se originan principalmente por (i) la resolución del encoder -dado que se emplea una unión directa sin reducción- y a la (ii) calibración.

4.2 Movimiento simultáneo

Cuando es necesario que el manipulador siga una trayectoria respecto a la mesa en movimiento, la solución anterior no alcanza. Se requiere un movimiento simultáneo sincronizado en tiempo real entre la mesa giratoria y el manipulador. La sincronización se logra corrigiendo la trayectoria original del manipulador a medida que la mesa -y su referencia- rotan. Con ese propósito puede emplearse una técnica movimiento del manipulador con corrección de la referencia en línea, tal como la analizada en [1]. Pero el corrector de trayectoria provisto por el fabricante en el módulo de movimientos avanzados para el controlador ABB SCplus no es válido para esta función. Ocurre que, entre otras limitaciones, no puede modificar la orientación de la herramienta durante un movimiento en el que se emplea la corrección de trayectoria. Una alternativa a este corrector es coordinar el movimiento a partir de un método de segmentación y corrección de trayectoria deseada, como por ejemplo el planteado en [11]. Cualquiera sea el método de corrección empleado, siempre existirá un error de trayectoria adicional a los mencionados en 4.1 provocado por (i) el tiempo entre correcciones de trayectoria y (ii) el tiempo de actualización de la terna asociada a la mesa. Por lo tanto existirá un compromiso entre el error de trayectoria y la velocidad del movimiento simultáneo.

4.3 Resolución de configuración

Dado que cada movimiento de la mesa implica una actualización de la referencia asociada, se plantea un tema a resolver: la supervisión de la configuración del manipulador. El lenguaje RAPID no es robusto en este tema de importancia central para la programación. Se exige que el usuario indique la solución aproximada (cuadrante) de tres de sus ejes, limitando la versatilidad de la programación off-line. Este requerimiento se repite cada vez que se cambia la referencia, la herramienta o la posición objetivo. Para solucionar este problema se programó un módulo de resolución de cuadrantes [12], que se emplea de manera conjunta al módulo de control del eje externo en cada movimiento coordinado.

4.4 Resultados obtenidos

Para cada tipo de movimiento coordinado se obtuvieron los resultados presentados en la Tabla 1. En el movimiento no simultáneo, los valores de repetibilidad y exactitud son prácticamente coincidentes, dado que con la mesa detenida los errores de posicionamiento atribuibles al manipulador son despreciables respecto a los de la mesa rotativa. En movimiento simultáneo los valores se obtuvieron para movimientos en los que la velocidad del manipulador está por debajo de los $5mm/seg$.

5 Conclusiones y Trabajos Futuros

Se plantearon las bases del diseño de la integración, con el requerimiento de obtener un sistema funcional y amigable para el usuario/programador. Se des-

Tabla 1. Resultados de cada tipo de movimiento coordinado.

Alternativa	Repetibilidad	Exactitud
Movimiento no simultáneo	$\approx 1{,}2mm \pm 0,35°$	$\approx 1{,}3mm \pm 0,35°$
Movimiento simultáneo	$\approx 1{,}2mm \pm 0,35°$	$\approx 2{,}5mm \pm 0,45°$

cribió la solución implementada a nivel de comunicación, división del control y sincronización. Se detalló la funcionalidad de los comandos del nuevo módulo de usuario para control del eje externo. Se presentaron los problemas particulares que surgen para coordinar movimientos del manipulador con una referencia de movimientos móvil y externa. Se expusieron los resultados en cuanto a la repetibilidad y exactitud en movimientos coordinados.

La plataforma resultante muestra la flexibilidad de integración de manipuladores y otros componentes industriales, utilizando redes de comunicación diseñadas para este propósito. También evidencia que esa integración no es inmediata y, en general, requiere tareas de desarrollo previo a su implementación.

La cinemática redundante de la nueva plataforma (robot + eje externo = 7 ejes) se utilizará para continuar el análisis y ensayo de algoritmos y criterios de resolución de redundancia iniciado en trabajos previos [2].

Referencias

1. C. Cartelli and M. Anigstein, "Sobre la incorporación de nuevas estrategias de control a un robot con control de posición," *JAR 2012*, 2012.
2. C. Cartelli and M. Anigstein, "Implementing redundancy-resolution algorithms in a typical industrial robot," *IEEE Latin America Transactions*, vol. 12, no. 7, pp. 1228–1233, 2014.
3. N. Santillán, A. Lempel, G. Galache, and D. Malanij, "Integración entre un manipulador abb y un eje externo siemens," *AST 2008*, 2008.
4. C. Cartelli and M. Anigstein, "Implementación de un eje externo para un manipulador industrial," *RPIC 2011*, 2011.
5. D. Reclik, G. Kost, and J. Świder, "The signal connections in robot integrated manufacturing systems," *Juornal of Achievements in Materials and Manufacturing Engineering*, vol. 23, no. 1, 2007.
6. ABB, *RAPID Overview - V 4.0*. Vasteras, Sweden: ABB, 2000.
7. SiemensAG, *S7-300 CPU Data: CPU 317T-2 DP - Manual*. Nurnberg, Germany: Siemens AG, 2005.
8. P. nutzerorganisation e.V., *PROFIBUS System Description - Technology and Application*. Karlsruhe, Germany: PROFIBUS nutzerorganisation e.V., 2010.
9. SiemensAG, *SIMOVERT MASTERDRIVES Motion Control Compendium V1.66*. Nurnberg, Germany: Siemens AG, 2002.
10. N. Mohan, *Electric Drives an integrative approach - Ch. 8.* Minnesota, USA: Mnpere, 2000.
11. A. F. Brumovsky, V. M. Liste, and M. Anigstein, "Implementación de control de fuerzas en robots industriales: un caso," *JAR 2006*, 2006.
12. C. Cartelli and M. Anigstein, "Sobre la programación off-line de robots industriales y las múltiples soluciones para una posición objetivo," *JAR 2010*, no. 1, 2010.