SciVerse ScienceDirect
Disponible en www.sciencedirect.com

Revista Iberoamericana de Automática e Informática industrial 10 (2013) 96–103

www.elsevier.es/RIAI

# Implementación basada en el middleware OROCOS de controladores dinámicos pasivos para un robot paralelo

Marina Vallés[a,*], Jose I. Cazalilla[a], Ángel Valera[a], Vicente Mata[b], Álvaro Page[c]

[a] *Instituto de Automática e Informática Industrial, Universidad Politécnica de Valencia, Valencia, España*
[b] *Centro de Investigación de Tecnología de Vehículos, Universidad Politécnica de Valencia, Valencia, España*
[c] *Departamento de Física Aplicada, Universidad Politécnica de Valencia, Valencia, España*

**Resumen**

La complejidad actual de los sistemas robotizados y de las aplicaciones que éstos deben realizar requiere que los robots dispongan de un control automático que permita la ejecución de las distintas tareas que forman parte del algoritmo de control y que tenga en cuenta cuestiones relacionadas por ejemplo con la periodicidad, el modo de ejecución, el hardware que se utilizará, etc. Para el desarrollo de este tipo de aplicaciones de control en los últimos años se tiende a la programación basada en componentes puesto que ésta permite obtener código reusable. Así mismo también se está incrementando la utilización de middlewares que permiten la abstracción de los sistemas operativos, el soporte de tiempo real y la infraestructura de comunicaciones. En el presente artículo se propone la utilización de un middleware orientado especialmente a la robótica: OROCOS. Así se describe cómo haciendo uso de una de sus librerías, Orocos Toolchain, se han desarrollado una serie de componentes correspondientes a distintos algoritmos para el control dinámico de robots, aplicándose a un robot paralelo de 3 grados de libertad (DOF). 

*Palabras Clave:*
tiempo-real, middleware, implementación basada en componentes, manipuladores paralelos.

## 1. Introducción

El control automático de los sistemas robotizados actuales involucra cada vez más la implementación de distintas tareas a realizar por el robot con distinto grado de complejidad, de naturaleza periódica o aperiódica, ejecutadas de manera local o distribuidas a través de una red de comunicaciones y para lo cual es necesario manejar hardware de distinta naturaleza. Abordar la implementación de controladores para una nueva plataforma robotizada supondría volver a desarrollar código adecuado para el nuevo hardware o en el mejor de los casos adaptar el ya existente. En la última década ha habido un interés creciente en los sistemas software basados en componentes que faciliten el trabajo en situaciones como las anteriormente descritas a partir del desarrollo de componentes reusables y que puedan interoperar fácilmente entre ellos. Según (Clements and Shaw, 2009) plantear el desarrollo de código bajo esta filosofía es posible debido a la situación de evolución o maduración en la que se encuentra actualmente la arquitectura del software en donde la programación orientada a objetos se encuentra totalmente establecida y en la que un componente podría entenderse como un objeto, destacando su interoperatividad y reusabilidad. Así, en (Alonso et al., 2011) se comenta que el *desarrollo de software basado en componentes* (CBSD) es una alternativa de diseño que favorece la reutilización de elementos software y facilita el desarrollo de sistemas a partir de elementos preexistentes, siendo un componente software una unidad de composición con interfaces bien definidas y un contexto de uso explícito. Como complemento a esta definición, en (Tang et al., 2011) se expone que el CBSD se basa en la descomposición de un software en componentes lógicos o funcionales con interfaces bien definidas, con el fin de facilitar el tiempo de desarrollo y mejorar el mantenimiento del sistema. Analizar el coste de un CBSD, sin embargo, es un tema que ha evolucionado poco dentro de la ingeniería del software y en su estimación resulta importante considerar, no sólo en el coste de la creación, sino también la eficiencia y productividad. A la hora de desarrollar un CBSD es conveniente contar con un entorno de trabajo que dé soporte a este tipo de filosofía de programación del software. Dentro de la robótica existen algunos middleware que facilitan dicha labor. Los middleware funcionan como una capa de abstracción de software distribuida,

* Autor en correspondencia
*Correos electrónicos:* mvalles@isa.upv.es (Marina Vallés ), jocamo5@ei.upv.es (Jose I. Cazalilla), giuprog@isa.upv.es (Ángel Valera), vmata@mcm.upv.es (Vicente Mata), afpage@ibv.upv.es (Álvaro Page)


http://dx.doi.org/10.1016/j.riai.2012.11.009

que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red). Así el middleware abstrae de la complejidad y heterogeneidad de las redes de comunicaciones subyacentes, así como de los sistemas operativos y lenguajes de programación, proporcionando una API para la fácil programación y manejo de aplicaciones distribuidas (Campbell et al., 1999) . En la actualidad existen diversos middlewares, muchos de ellos pensados para aplicaciones robóticas como son Player (Gerkey et al., 2003) (Collett et al., 2005), Orca, Miro (Utz et al., 2002), OpenRDK, Marie (Cote et al., 2006), Smartsoft, Orocos (Bruyninckx, 2001) o ROS.

Para el presente trabajo se escogió el middleware OROCOS por estar especialmente pensado para robots industriales y por la librería de tiempo real que ofrece, la Orocos Toolchain, la cual, lo hace especialmente indicado para aplicaciones en las que se requiera manejo de tiempo real, como es el caso de la plataforma robótica escogida, un robot paralelo.

Un Manipulador Paralelo (MP) es un mecanismo en donde el movimiento del elemento final, generalmente una plataforma móvil, está conectado a la base fija por medio de, al menos dos cadenas cinemáticas (Theodor, 2003). Los MPs han supuesto un campo de investigación muy activo durante los últimos años. En la actualidad la aplicación de estos resultados de investigación se ha transferido a la industria, ver (Pierrot et al., 2009), principalmente debido a las ventajas que los MPs presentan frente a los robots serie (o de cadena cinemática abierta) ya que, al dividir la carga entre varias articulaciones, los MPs proporcionan alta velocidad y precisión de posicionado, capacidad de carga y rigidez. Debido a estas ventajas podemos encontrar diversas aplicaciones de los MPs, como por ejemplo el popular robot Delta (Merlet, 1962), que con sólo 3 DOF prismáticos se puede aplicar satisfactoriamente a tareas de pick-and-place (Clavel, 1988), aplicaciones médicas, como equipos de reanimación cardio-pulmonar (Li and Xu, 2007), máquinas-herramienta (Pierrot et al., 2009), simuladores de vuelo o conducción, máquinas de pruebas, etc. (Steward, 1965), (Gough and Whitehall, 1962), (Merlet, 2000) o (Tsai, 1999).

Además de los conocidos y habituales esquemas de control PD y PID, se pueden encontrar una gran variedad de esquemas de control utilizados en los MPs, como por ejemplo, control robusto no lineal (Kim et al., 1999), control robusto (Fu and Mills, 2007), control borroso (Stan et al., 2009) y control por modos deslizantes (Gou et al., 2009) entre otros.

Sin embargo, para controlar de forma efectiva un manipulador consiguiendo una respuesta rápida y precisa es necesario el diseño de esquemas de control basados en el modelo dinámico del robot (Abdellatif and Heimann, 2010). El modelo utilizado en este trabajo corresponde a un modelo reducido para el MP que se obtiene siguiendo la metodología propuesta previamente por (Díaz-Rodríguez et al., 2010), basada en considerar simplificaciones debido a las simetrías y geometrías de las partes del robot.

El control dinámico del MP se obtiene a partir de controladores basados en la pasividad y la parte novedosa de este artículo corresponde a la implementación que se ha llevado a cabo de dichos controladores basada en el concepto de componentes con el objetivo de permitir su reusabilidad en otras aplicaciones. Así, se describe la descomposición de éstos en componentes y su implementación haciendo uso del middleware para robots OROCOS.

Así, el resto del artículo se organiza de la siguiente manera. En la segunda sección, se describe el funcionamiento del middleware OROCOS utilizado para este trabajo. En la tercera sección se describe el robot paralelo empleado en el trabajo, así como las ecuaciones obtenidas para su modelado. En la sección cuarta se procede a describir los controladores diseñados así como su implementación. En la sección quinta se muestran los resultados reales conseguidos con la plataforma de control desarrollada sobre el robot real. Y se finaliza con la sección sexta en donde se describen las conclusiones del trabajo

## 2. El middleware OROCOS

OROCOS (Open RObot COntrol Software) proporciona las cuatro funciones principales de un middleware: abstracción del sistema operativo y soporte de tiempo real, servicios middleware, infraestructura de comunicación y un modelo basado en componentes (Orocos, 2011a).

El proyecto OROCOS se puso en marcha en el 2001 a propuesta de EURON (la Red Europea de Robótica) gracias a un proyecto subvencionado por la Comunidad Europea. Pese a que OROCOS es un middleware de reciente creación y en continuo desarrollo, poco a poco se está consolidando como una de las mejores opciones para realizar el control de robotS, estando presente en la actualidad en diversos proyectos de investigación de ámbito internacional (Orocos, 2011b).

El proyecto OROCOS soporta en la actualidad tres librerías: Kinematics and Dynamics (KDL), Bayesian Filtering (BFL) y Orocos Toolchain.

La librería KDL permite reducir el modelado y la especificación del movimiento a un problema meramente geométrico (aunque con varios sistemas de referencia) con cálculos matemáticos, pudiéndose resolver un movimiento en base a unas especificaciones en cada una de las articulaciones. Para ello desarrolla un marco de aplicación independiente para el modelado, y otro para el cálculo de cadenas cinemáticas en robots, modelos biomecánicos humanos, figuras animadas por ordenador, máquinas herramientas, etc. Además, proporciona librerías predefinidas de clases de objetos geométricos, cadenas cinemáticas de varias clases (serie, humanoide, paralelas o móviles) así como su especificación de movimiento e interpolación.

La librería BFL proporciona un marco de aplicación independiente para, por ejemplo, el procesamiento de la información recursiva y algoritmos de cálculo basado en la regla de Bayes, tales como filtros de Kalman o Filtros de Partículas. Estos algoritmos pueden ser ejecutados como una aplicación en tiempo real, o ser utilizados para la estimación de las aplicaciones de Cinemática y Dinámica.

Relacionado con esta librería, cabe destacar que en la actualidad existen aplicaciones similares a la que propone OROCOS, llegando a ser, incluso, mejores y más sencillas en su uso. Por ello, el desarrollo en los últimos años de la BFL es mínimo, ya que no es tan usado como otras librerías del proyecto OROCOS.

Orocos ToolChain es la librería de más reciente creación del proyecto OROCOS. Su primera versión apareció en julio de 2010 y, poco a poco, se está convirtiendo en la herramienta más usada. Dado que ha sido la librería utilizada en el trabajo del presente artículo a continuación se verá con más detalle.

### 2.1. *La librería Orocos Toolchain*

La particularidad de la ToolChain es que en esta herramienta vienen incluidas otras herramientas como:

(a) *AutoProj*, que es una herramienta para descargar y compilar las librerías necesarias de forma automática.

(b) *Real Time Toolkit* (RTT) permite a los desarrolladores la creación de componentes totalmente configurables (incluso en tiempo de ejecución) e interactivos basados en el control de aplicaciones en tiempo real, pudiéndose usar en aplicaciones con características tales como capturar y graficar el flujo de datos entre los componentes, modificar los algoritmos en tiempo de ejecución, configurar los componentes y la aplicación a partir de archivos XML, interactuar con otros dispositivos directamente desde una interfaz gráfica de usuario o mediante el *prompt*, ampliar las aplicaciones con estructuras de datos propias y ejecutar la aplicación tanto en sistemas operativos estándar como sistemas de tiempo-real. Además, el RTT, permite que los componentes se ejecuten en cualquier sistema operativo ofreciendo todas las funciones y comandos del tiempo real, como la comunicación de los componentes y la configuración de los mismos mediante un archivo XML.

(c) *Orocos Component Library* permite crear todos los tipos de componentes (apoyándose en RTT). Cada uno de los componentes está definido a partir de una primitiva llamada *TaskContext*, la cual define el entorno (contexto) básico que debe tener un componente en OROCOS (que posteriormente se modifica y configura según las preferencias del usuario). Este contexto está descrito por cinco primitivas: *Event*, *property*, *Command*, *Method* y *Data port*. En la siguiente subsección se explica más en detalle en qué consiste un componente OROCOS.

(d) *OroGen y TypeGen* son dos herramientas para generar código OROCOS a partir de unas cabeceras descritas, o de un fichero de descripción del componente (con una sintaxis ya predefinida).

De este modo, y en un único paquete, está todo lo necesario para comenzar a realizar componentes que tengan distintas funcionalidades. Además, una de las ventajas de Orocos ToolChain es su soporte multiplataforma ya que se puede trabajar tanto en Linux como en Windows o MAC (en algunos casos, sin utilizar la parte de tiempo real).

### 2.2. *Componentes en OROCOS*

Uno de los objetivos de OROCOS es que el software para sistemas complejos no se construya únicamente en tiempo de compilación sino también en tiempo de lanzamiento a ejecución o incluso en ejecución. Esto se permite mediante la generación y ejecución de componentes. Un componente OROCOS es una unidad básica de funcionalidad que puede ejecutar (en tiempo real) uno o más programas en un único hilo (o thread). De cara al usuario, el sistema está libre de prioridades y todas las operaciones son no bloqueantes, incluido el intercambio de datos y otras formas de comunicación, como por ejemplo los eventos y comandos. Otro aspecto importante es que los componentes de tiempo real pueden comunicarse de forma transparente con componentes que no sean de tiempo real.

Así, mediante el diseño modular, se permite un rápido seguimiento del flujo de ejecución del programa, facilitando la creación de nuevos componentes que, combinándolos con otros, pueden conseguir nuevas funcionalidades. Esta estructura modular, además, permite una ejecución distribuida del código correspondiente a módulos distintos, pudiéndose obtener tiempos de ejecución menores que los obtenidos mediante una ejecución serie. Otra de las ventajas del planteamiento basado en componentes para desarrollar software es que el código se puede reusar, pudiéndose parametrizar su ejecución en tiempo de lanzamiento.

Figura 1: Robot paralelo 3-PRS desarrollado

## 3. El Robot Paralelo Desarrollado

En el ámbito del proyecto de investigación Identificación de Parámetros Físicos en Sistemas Mecánicos Complejos (DPI2005-08732-C02-01/02) del Plan Nacional del Ministerio de Ciencia e Innovación se desarrolló un manipulador paralelo. Éste puede verse como una plataforma móvil conectada a la base fija mediante tres cadenas cinemáticas. Cada cadena consiste en un motor que mueve un actuador de husillo de bolas y una varilla de conexión. La figura 1 muestra el aspecto mecánico del robot desarrollado.

La elección de la arquitectura y movimiento del robot paralelo vino determinada por la necesidad de desarrollar un robot de bajo coste capaz de generar dos movimientos rotacionales (roll y pitch) y un movimiento lineal de elevación (z). Cuando se combinan movimientos prismáticos y de revolución se pueden encontrar arquitecturas 3-PRS (Chablat and Wenger, 2003) y 3-RPS (Lee and Arjunan, 1991), donde la notación R, P y S hacen referencia a las articulaciones de rotación, prismáticas y

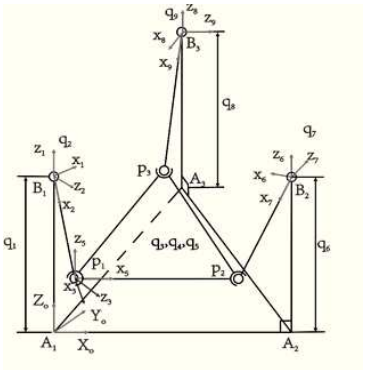


Figura 2: Localización de los sistemas de coordenadas

esféricas respectivamente. Para este robot se escogió la arquitectura 3-PRS tras comparar las ventajas e inconvenientes de cada una de ellas.

Para poder establecer el control del MP fue necesario desarrollar tanto el modelo cinemático como el dinámico del robot, los cuales se describen a continuación.

### 3.1. Modelo Cinemático Directo

La cinemática directa del MP consiste en, dados los movimientos lineales de los actuadores, encontrar los ángulos de balanceo ($\beta$) y alabeo ($\gamma$) y la elevación (z). Se puede utilizar la notación de Denavit-Hartenbert para establecer las coordenadas generalizadas del modelo cinemático. La tabla 1 muestra los parámetros D-H del robot considerado. A partir de la tabla 1 se puede ver como a partir de 9 coordenadas generalizadas, se puede definir la cinemática del robot. La localización de los sistemas de coordenadas para modelar la cinemática se muestra en la figura 2.

Tabla 1: Parámetros D-H para el PM de 3-DOF

| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| $d_i$ | $q_1$ | 0 | 0 | 0 | 0 | $q_6$ | 0 | $q_8$ | 0 |
| $\alpha_i$ | 0 | 0 | $l_a$ | 0 | 0 | 0 | 0 | 0 | 0 |
| $\theta_i$ | $\frac{\pi}{6}$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ | $\frac{5\pi}{2}$ | $q_7$ | $\frac{-\pi}{2}$ | $q_9$ |
| $\alpha_i$ | 0 | $\frac{\pi}{2}$ | 0 | $\frac{\pi}{2}$ | $\frac{\pi}{2}$ | 0 | $\frac{\pi}{2}$ | 0 | $\frac{\pi}{2}$ |

Las tres patas del robot paralelo se conectan a la plataforma móvil formando un triángulo equilátero. Por ello se puede ver que la longitud entre $p_i$ y $p_j$ es constante e igual a $l_m$. Así,

$$f_1(q_1,q_2,q_6,q_7) = \\ \|(\vec{r}_{A_1B_1}+\vec{r}_{B_1p_1})-(\vec{r}_{A_1A_2}+\vec{r}_{A_2B_2}+\vec{r}_{B_2p_2})\|-l_m=0 \quad (1)$$

$$f_2(q_1,q_2,q_8,q_9) = \\ \|(\vec{r}_{A_1B_1}+\vec{r}_{B_1p_1})-(\vec{r}_{A_1A_3}+\vec{r}_{A_3B_3}+\vec{r}_{B_3p_3})\|-l_m=0 \quad (2)$$

$$f_3(q_6,q_7,q_8,q_9) = \|(\vec{r}_{A_1A_3}+\vec{r}_{A_3B_3}+\vec{r}_{B_3p_3}) \\ -(\vec{r}_{A_1A_2}+\vec{r}_{A_2B_2}+\vec{r}_{B_2p_2})\|-l_m=0 \quad (3)$$

En la cinemática directa, la posición de los actuadores es conocida. Así, el sistema de ecuaciones (1) - (3) se puede ver como un sistema no-lineal con $q_2$, $q_7$ y $q_9$ como desconocidas. Para la resolución del sistema no lineal el problema cinemático directo, en este trabajo, se ha utilizado el método numérico de Newton-Rhapson ya que tiene una convergencia muy rápida (convergencia cuadrática) cuando la estimación inicial está cerca de la solución deseada (Jalón and Bayo, 1994). El método es iterativo y se puede escribir como:

$$\begin{bmatrix} q_2 \\ q_7 \\ q_9 \end{bmatrix}^{i+1} = \begin{bmatrix} q_2 \\ q_7 \\ q_9 \end{bmatrix}^{i} - J_i^{-1} \begin{bmatrix} f_1(q_2,q_7) \\ f_2(q_2,q_9) \\ f_3(q_7,q_9) \end{bmatrix}^{i} \quad (4)$$

En la ecuación $i$ significa que las variables y funciones se evalúan en la iteración $i$. La matriz $J$ es la matriz Jacobiana de $f_i$ con respecto a las variables [$q_2$,$q_7$,$q_9$]. El proceso iterativo finaliza cuando

$$\sqrt{(f_1(q_2,q_7)^i)^2+(f_1(q_2,q_9)^i)^2+(f_1(q_7,q_9)^i)^2} < \varepsilon \quad (5)$$

donde $\varepsilon$ es una cantidad positiva pequeña establecida por el usuario.

El método de Newton-Rhapson requiere una aproximación inicial tan cercana como sea posible al valor solución. En este caso esto no supone ningún problema ya que la posición inicial de la articulación que conecta la plataforma con el actuador es aproximadamente $\frac{2\pi}{5}$. La subsecuente aproximación inicial considera los valores de la posición previa del robot.

La localización de la plataforma móvil se define usando el sistema de coordenadas unido a él. Una vez encontradas las coordenadas generalizadas para las patas del robot, se puede encontrar la posición de los puntos $p_i$. Estos tres puntos comparten el plano de la plataforma. Basado en estos puntos se puede construir la matriz rotacional de la plataforma con respecto a la base. A partir de dicha matriz de rotación se pueden calcular directamente tanto el resto de coordenadas generalizadas ($q_3$, $q_4$ y $q_5$) del robot como la elevación (z) y los ángulos de balanceo ($\beta$) y alabeo ($\gamma$) de la plataforma.

### 3.2. Modelado Cinemático Inverso

La cinemática inversa consiste en, dados los ángulos de balanceo y alabeo de la plataforma ($\beta$ y $\gamma$) y la elevación (z), encontrar el movimiento lineal de los actuadores. Usando el sistema de ángulos fijo X-Y-Z; la matriz de rotación se puede definir como:

$${}^{O}R_p = \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma - s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma - c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{bmatrix} \quad (6)$$

donde c* y s* hacen referencia al cos(*) y sen(*) respectivamente. Dados $\gamma$ y $\beta$, el ángulo de cabeceo ($\alpha$) se puede encontrar como sigue:

$$\alpha = \arctan 2(s_\beta s_\gamma, (c_\gamma + c_\beta)) \quad (7)$$

Las posiciones de los actuadores ($q_1$, $q_6$ y $q_8$) se pueden calcular directamente a partir de dicha matriz de rotación.

### 3.3. Modelado Dinámico

Como es bien conocido, la ecuación dinámica de los robots manipuladores se puede expresar mediante la siguiente ecuación no lineal:

$$M(\vec{q},\vec{\Phi})\cdot\ddot{\vec{q}} + \vec{C}(\vec{q},\dot{\vec{q}},\vec{\Phi}) + \vec{G}(\vec{q},\vec{\Phi}) = \vec{\tau} \quad (8)$$

donde $M(\vec{q},\vec{\Phi})$, $\vec{C}(\vec{q},\dot{\vec{q}},\vec{\Phi})$ y $\vec{G}(\vec{q},\vec{\Phi})$ son la matriz de masas del sistema, el vector de fuerzas centrífugas y de Coriolis y los términos gravitacionales respectivamente, y como se puede apreciar, dependen de los parámetros dinámicos $\vec{\Phi}$ . $\vec{q}$ y $\vec{\tau}$ son los vectores de las coordenadas y pares generalizados. Para poder obtener los términos de la ecuación dinámica se debe construir el modelo para que éste esté en forma lineal a los parámetros dinámicos (Grotjahn et al., 2004), (Díaz-Rodríguez et al., 2010):

$$K(\vec{q},\dot{\vec{q}},\ddot{\vec{q}})\cdot\vec{\Phi} = \vec{\tau} \quad (9)$$

En la ecuación (9) solo se puede identificar un conjunto de parámetros porque algunos de ellos tienen una contribución pequeña o insignificante en el comportamiento dinámico del sistema. Además, éstos son propensos al ruido en las medidas y a dinámicas no modeladas (Díaz-Rodríguez et al., 2008). Para obtener la identificación del modelo dinámico se ha utilizado una metodología que aparece en (Díaz-Rodríguez et al., 2010).

Los términos de la ecuación (8) se pueden obtener una vez realizada la identificación del proceso. Así, se pueden calcular los términos del vector gravitacional haciendo cero las velocidades y aceleraciones generalizadas de la ecuación (9):

$$K(\vec{q},\dot{\vec{q}}=0,\ddot{\vec{q}}=0)\cdot\vec{\Phi} = \vec{G}(\vec{q}) \quad (10)$$

Los términos centrífugos y de Coriolis dependen de las coordenadas generalizadas y de las velocidades. Haciendo cero las aceleraciones generalizadas de la ecuación (9) otra vez se puede establecer que:

$$K(\vec{q},\dot{\vec{q}},\ddot{\vec{q}}=0)\cdot\vec{\Phi} - \vec{G}(\vec{q}) + M^d A_d^{-1}\vec{b} = \vec{C}(\vec{q},\dot{\vec{q}}) \quad (11)$$

Por último, la matriz de masa se puede determinar de la forma siguiente:

$$K^*(\vec{q},\dot{\vec{q}}=0,\vec{e_i})\cdot\vec{\Phi}^* - \vec{G}(\vec{q}) = M_i(\vec{q}) \quad (12)$$

Siendo $M_i(\vec{q})$ la columna i-ésima de la matriz de masas, y $\vec{e_i}$ = [ 0 ... 1 ... 0 ]$^T$ un vector columna con un 1 en la posición i-esima.

Por lo tanto, las ecuaciones diferenciales que describen la ecuación del movimiento (8) se puede construir utilizando las ecuaciones (9)-(12).

### 3.4. Desarrollo del Manipulador Paralelo 3-PRS

En el manipulador paralelo 3-PRS desarrollado se pueden distinguir dos partes: la unidad mecánica y la unidad de control.

En lo que se refiere a la unidad mecánica (ver figura 1), para actuar sobre las 3 articulaciones del robot se han utilizado 3 motores brushless BMS465, de AEROTECH, equipados con encoders que proporcionan 4.000 cuentas por revolución. Para actuar sobre dichos motores se utilizan las etapas de amplificación BA10, de AEROTECH.

De la misma forma que el robot paralelo, la especificación y el diseño del sistema de control del robot se realizó íntegramente en la Universidad Politécnica de Valencia. Así se optó por una arquitectura basada en un PC industrial de altas prestaciones 4U Rackmount con 7 slots PCI y 7 ISA. Tiene un procesador Intel® Core 2 Quad/Duo processor a 2,5GHz y una memoria de 4 Gb RAM.

El sistema se complementa con las tarjetas de adquisición de datos necesarias para poder acceder a las señales de robot. En concreto se tiene una tarjeta Advantech PCI-1720 para el suministro de las acciones de control a los 3 motores. Así mismo, se utiliza otra tarjeta de Advantech: la PCL-833 para la lectura de encoders.

La arquitectura propuesta proporciona 2 ventajas muy interesantes: su bajo precio y su flexibilidad. Al ser un entorno abierto basado en PC se puede utilizar cualquier entorno de programación lo que permite, por ejemplo, implementar diferentes algoritmos de generación de referencias y control no-lineal, utilización de sensorización avanzada, como por ejemplo sensores de fuerza/par, visión artificial, sistemas inerciales, etc.

## 4. Diseño e Implementación en OROCOS de los controladores

### 4.1. Algoritmos de Control para el Robot Paralelo

Los algoritmos de control implementados en este trabajo corresponden a controladores basados en la pasividad. Éstos resuelven el problema de control mediante la explotación de la estructura física del sistema robotizado, y en especial de su propiedad de pasividad. La filosofía de diseño de estos controladores es reconfigurar la energía natural del sistema de tal forma que se consiga el objetivo de seguimiento del control (Ortega and Spong, 1989).

En el problema de control punto a punto (regulación del sistema), los controladores basados en pasividad se pueden ver como casos particulares de la siguiente ley de control general:

$$\tau_e = -K_p e - K_d\dot{q} - u \quad (13)$$

donde $e = q - q_d$ y $u$ varía dependiendo de la clase de controlador tal como se muestra en la tabla 2.

Tabla 2: Controladores por pasividad punto a punto

| Controlador | u |
|---|---|
| PD+G | -G(q) |
| PD+G0 | $-G(q_d)$ |
| PID | $K_i\int_0^t e\,\mathrm{d}t$ |

Además, en este trabajo se han implementado y probado también controladores de trayectoria basados en la pasividad. Al igual que ocurría para los controladores punto a punto, podemos encontrar en la literatura varios controladores. En este

trabajo se ha optado por el propuesto por (Paden and Panja, 1988), en el que la energía cinética y potencial se modifica de forma adecuada para que el controlador sea global y asintóticamente estable. La expresión del controlador es la siguiente:

$$\tau_c = M(q)\ddot{q}_d + C(q,\dot{q})\dot{q}_d + G(q) - K_p e - K_d \dot{e} \qquad (14)$$

En la ecuación 14 se puede apreciar cómo el controlador tiene dos partes: una compensación de los términos dinámicos del robot y un controlador tipo proporcional-derivativo.

### 4.2. *Entorno de desarrollo e Implementación de los controladores*

Para establecer el control y el desarrollo de aplicaciones con el manipulador paralelo, en este trabajo se ha utilizado la librería Orocos Toolchain con el sistema Linux Ubuntu, optándose por Xenomai para dotarlo de la funcionalidad de un sistema operativo de tiempo real.

Para la implementación de los controladores diseñados se planteó su especificación como componentes (ver figura 3). En primer lugar se tiene un componente *SensorPos* que accede en este caso a la tarjeta PCL-833 y que se utiliza para realizar la lectura de los valores de los encoders de las tres articulaciones del robot paralelo.

El componente *Actuador* es el encargado de suministrar las acciones de control al robot, realizándose mediante la tarjeta de conversiones de Digital/Analógico PCI-1720 de Advantech. Para ello se tienen 3 instancias diferentes de un único componente, lo que permite en tiempo de ejecución seleccionar el canal de salida de la tarjeta y el puerto del componente por el que recibirá el valor de la acción de control a suministrar a la etapa de potencia del robot.

El componente *Gener_Refe* es el encargado de generar los valores de referencia de movimiento correspondientes a la posición, velocidad y aceleración, las cuales se usarán como entrada al resto de módulos que componen el controlador.

Además, debido a que el robot no está equipado con sensores de velocidad, se ha tenido que desarrollar un componente (*Estim_vel*) encargado de realizar la estimación de la velocidad a partir de la posición real de las 3 articulaciones del robot.

Para la implementación de los algoritmos de control se han desarrollado una serie de componentes que se pueden combinar y reusar, permitiendo conformar distintas estrategias de control. Así, se ha implementado el componente *PID* que puede funcionar como un controlador pasivo punto-a-punto o formar parte de un controlador dinámico de trayectoria en donde los términos inerciales, gravitacionales y centrífugos y de Coriolis se calculan mediante los componentes *Inercia*, *Gravity* y *Coriolis*. La selección de la estrategia de control punto-a-punto o trayectoria se realiza mediante el componente *Combinador*.

De forma adicional al controlador y los componentes encargados de la sensorización y actuación, se ha implementado un componente al cual se le ha dado el nombre de *Supervisor* que es el encargado de almacenar el valor de las temporizaciones de cada uno de los componentes y el tiempo total de ejecución, así como las referencias y las posiciones reales del robot. Además, puede detectar de manera automática errores en la lectura de la posición del robot y/o errores en el sistema de actuación, de manera que si detecta alguna situación anómala, además de generar un aviso al operario humano, realiza una parada de emergencia mediante la desactivación de la etapa de potencia del robot paralelo.

Es importante destacar que OROCOS permite varias opciones para la temporización e interconexión de los puertos de las componentes. Con este middleware se puede asignar un periodo fijo a una determinada tarea, aunque también es posible despertar a una tarea mediante un puerto de evento.

En el esquema de control desarrollado se ha asignado un periodo de muestreo fijo de 10mS al módulo de lectura de los encoders. El resto de módulos se despiertan en cascada. De esta forma los módulos de control se ejecutan sólo cuando le llega el flujo de datos del módulo de encoders, y los 3 módulos de convertidores sólo se ejecutarán cuando el módulo combinador les envíe los valores calculados de las acciones de control.

Por lo tanto, mediante este esquema en cascada, se está garantizando que un módulo sólo se ejecute cuando tenga todos los datos que necesita a través de sus puertos de entrada, estableciéndose el periodo de muestreo en el módulo que desencadena todo el proceso.

## 5. Resultados

A partir de la arquitectura hardware (robot/controlador) y software desarrollado, se han llevado a cabo diversas aplicaciones de control del manipulador paralelo. En las figuras siguientes se muestra el comportamiento del robot paralelo para distintas trayectorias y estrategias de control. Es necesario tener en cuenta que las referencias suministradas a la unidad de control del robot están en el espacio de la tarea, especificando así la orientación (*balanceo* y *alabeo*) y la altura z de la plataforma. Sin embargo, el control se realiza en el espacio de las articulaciones ($q_1$, $q_6$ y $q_8$). Por ello el controlador calcula para cada iteración del bucle de control en tiempo real el problema cinemático inverso.

La figura 4(a) muestra la respuesta de las 3 articulaciones del robot paralelo controlado por controladores punto a punto. En color azul se ha graficado las referencias de movimiento, en negro la respuesta de un PID y en rojo la respuesta del PD con la compensación de la gravedad. Se puede apreciar como en todos los casos la respuesta del robot es muy buena y sigue a la referencia de una forma muy precisa. La figura 4(b) muestra las acciones de control de dichos controladores, en negro la acción de control de un PID y en rojo la del PD con la compensación de la gravedad.

La figura 5 compara la respuesta obtenida con un controlador pasivo punto a punto (PD con compensación de la gravedad, en color rojo) y el controlador de trayectoria pasivo de Paden (color negro). En la columna (a) se muestran las posiciones de las 3 articulaciones. La columna (b) muestra los errores entre la referencia y la respuesta real del robot. Se puede comprobar fácilmente que los errores obtenidos con el controlador de trayectoria son mucho más pequeños que los obtenidos con el controlador punto a punto.

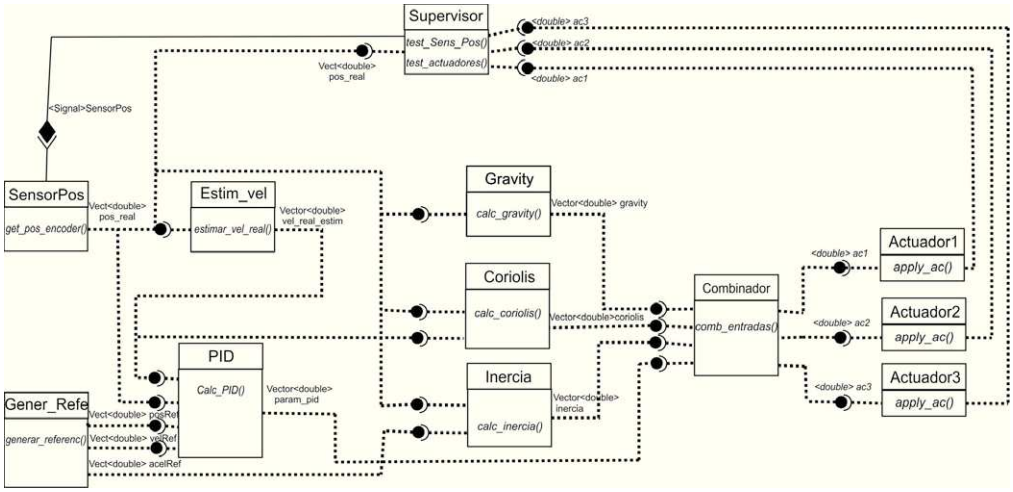
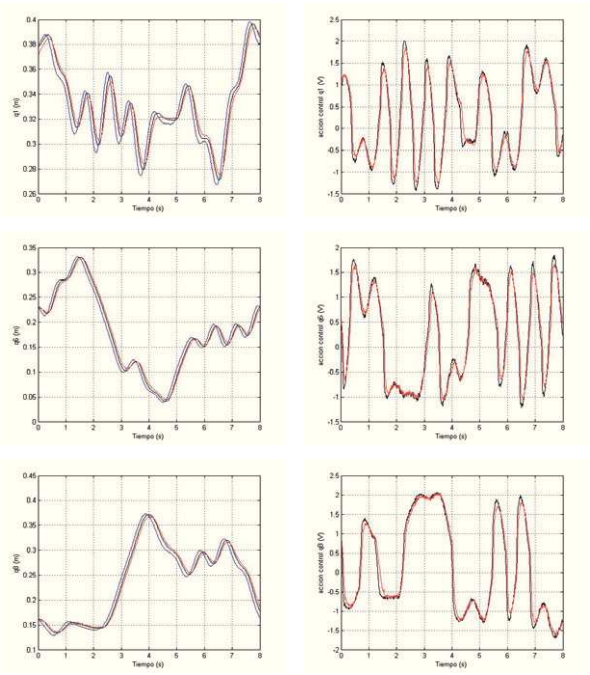
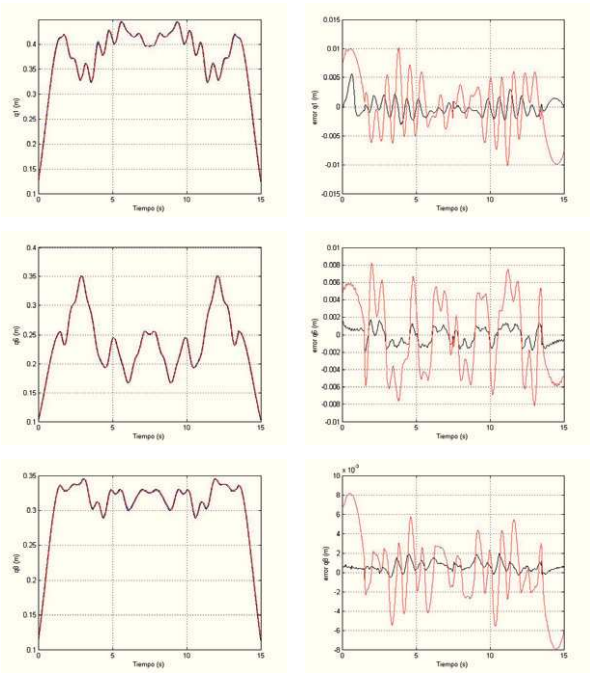



Figura 3: Arquitectura de componentes de control del manipulador paralelo



(a) posiciones $q_1$, $q_6$ y $q_8$ (b) acciones de control

Figura 4: Respuesta del robot paralelo con controladores punto a punto



(a) posición $q_1$, $q_6$ y $q_8$ (b) error $q_d$-q

Figura 5: Respuesta controlador punto a punto y trayectoria

## 6. Conclusiones

Este artículo se ha propuesto una forma novedosa de implementar controladores en tiempo-real haciendo uso del middleware OROCOS y de una programación basada en componentes. Esto permite que las implementaciones desarrolladas y el intercambio de información sean independientes del hardware disponible, su desarrollo más homogéneo y reusables. Esta forma de implementar los controladores permite también el que, dependiendo de la situación en que se encuentre el sistema, sea posible lanzar a ejecución un determinado controlador u otro, reduciendo así la complejidad del código y haciéndolo más modular. Estas características ofrecen muchas ventajas de cara a trabajos futuros relacionados con cambios de controlador dependiendo del estado del sistema y de su entorno o en la distribución del cómputo dependiendo de la disponibilidad de recursos.

Para poder validar el desarrollo realizado se ha establecido el control de un manipulador paralelo 3-PRS. En este trabajo se han implementado diferentes controladores dinámicos basados en la pasividad, abordándose tanto el problema de control

punto-a-punto como el control de trayectoria. En todos los casos se ha podido comprobar que la respuesta del robot ha sido muy buena.

## English Summary

### Dynamic controllers implementation based on the OROCOS middleware.


### Abstract

Automatic control of robotic systems nowadays deals more and more with the implementation of different tasks to be achieved by the robot with distinct complexity degree, periodic or aperiodic, with local execution or distributed along a communication network and needing to deal with different kinds of hardware. Deal with the controller implementation for a new robotic platform involved to develop suitable software again for the new hardware or, in the best case, to adapt the existing one. In the recent years it has been tending to a component-based programing that allows to develop reusable software and to use a middleware that make it possible to abstract the developed software from the used hardware and from the available communication protocols. At this paper one of these middleware has been used, specially oriented to robotics as is OROCOS. Using the Orocos Toolchain library the real-time components needed for the implementation of several dynamic controllers for a parallel manipulator have been developed. At this paper the robot and the designed controllers are described and the results over the actual robot are shown.



*Keywords:*

Real-time, middleware, based-on-components implementation, parallel manipulators.


## Agradecimientos


Los autores desean expresar su agradecimiento al Ministerio de Ciencia e Innovación de España por la financiación parcial de este trabajo bajo los proyectos (FEDER/CICYT) de investigación DPI2009-13830-C02-01 y DPI2010-20814-C02-(01, 02).


## Referencias


Abdellatif, H., Heimann, B., 2010. Advanced model-based control of a 6-dof hexapod robot: A case study. IEEE/ASME Transactions on Mechatronics 15, 269–279.

Alonso, D., J.A. Pastor, and P. Sánchez, a. B. l., Vicente-Chicote, C., 2011. Generación automática de software para sistemas de tiempo real: Un enfoque basado en componentes, modelos y frameworks. Revista Iberoamericana de Automática e Informática Industrial 9(2), 170–181.

Bruyninckx, H., 2001. Open robot control software: the orocos project. In: In IEEE International Conference on Robotics and Automation (ICRA'01), vol. 3, pp. 2523-2528.

Campbell, A., Coulson, G., Kounavis, M., 1999. Managing complexity: Middleware explained. IEEE Computer Society.

Chablat, D., Wenger, P., 2003. Architecture optimization of a 3-dof translational parallel mechanism for machining applications, the orthoglide. IEEE Transactions on Robotics and Automation 19, 403–410.

Clavel, R., 1988. Delta, a fast robot with parallel geometry. In: Proceedings of 18th International Symposium on Industrial Robot.

Clements, P., Shaw, M., july-aug. 2009. "the golden age of software architecturerevisited. Software, IEEE 26 (4), 70 –72.
DOI: 10.1109/MS.2009.83

Collett, I. T., MacDonald, B. A., Gerkey, B. P., 2005. Player 2.0: Toward a practical robot programming framework. In: In Australasian Conference on Robotics and Automation (ACRA'05).

Cote, C., Letourneau, D., Michaud, F., Brosseau, Y., 2006. Robotic software integration using marie. International Journal of Advanced Robotic Systems 3.

Díaz-Rodríguez, M., Mata, V., Farhat, N., Provenzano, S., 2008. Identifiability of the dynamic parameters of a class of parallel robots in the presence of measurement noise and modeling discrepancy. Mechanics Based Design of Structures and Machines 36, 478–498.

Díaz-Rodríguez, M., Mata, V., Valera, A., Page, A., 2010. A methodology for dynamic parameters identification of 3-dof parallel robots in terms of relevant parameters. Mechanism and Machine Theory 45, 1337–1356.

Fu, K., Mills, J. K., 2007. Robust control design for a planar parallel robot. International Journal of Robotics & Automation 22, 139–147.

Gerkey, B., Vaughan, R., Howard, A., 2003. The player/stage project: Tools for multi-robot and distributed sensor systems. In: In Proceedings of the 11th International Conference on Advanced Robotics.

Gou, H. B., Liu, Y. G., Liu, G. R., Li, H. R., 2009. Cascade control of a hydraulically driven 6-dof parallel robot manipulator based on a sliding mode. Control Engineering Practice 16, 105–168.

Gough, V., Whitehall, S., 1962. Universal tire test machine. In: Proceedings of 9th International Technical Congress FISITA.

Grotjahn, M., Heimann, B., Abdellatif, H., 2004. Identification of friction and rigid-body dynamics of parallel kinematic structures for model-based control. Multibody System Dynamics 11, 273–294.

Jalón, J.-G., Bayo, E., 1994. Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time challenge. Springer-Verlag, New-York.

Kim, D., Kang, J. Y., Lee, K. I., 1999. Nonlinear robust control design for a 6-dof parallel robot. KSME International Journal 13, 557–568.

Lee, K., Arjunan, S., 1991. A three-degrees-of-freedom micromotion in-parallel actuated manipulator. IEEE Transactions on Robotics and Automation 7, 634–641.

Li, Y., Xu, Q., 2007. Design and development of a medical parallel robot for cardiopulmonary resuscitation. IEEE/ASME Tr 12, 265–273.

Merlet, J.-P., 2000. Parallel Robots. Kluwer, London, U.K.

Ortega, R., Spong, M., 1989. Adaptive motion control of rigid robots: a tutorial. Automatica 25, 877–888.

Paden, B., Panja, R., 1988. Globally asymptotically stable pd+ controller for robot manipulators. Int. J. on Control 47, 1697–1712.

Pierrot, F., Nabat, V., Company, O., Krut, S., Poignet, P., 2009. Optimal design of a 4-dof parallel manipulator : From academia to industry. IEEE Transactions on Robotics 25, 213–224.

Stan, S. D., Balan, R., Maties, V., Rad, C., 2009. Kinematics and fuzzy control of isoglde3 medical parallel robot. Mechanika 1, 62–66.

Steward, 1965. A platform with 6 degree of freedom. In: Proceedings of the Institution of mechanical engineers.

Tang, J., Mu, L., Kwong, C., Luo, X., 2011. An optimization model for software component selection under multiple applications development. European Journal of Operational Research 212 (2), 301 – 311.
DOI: 10.1016/j.ejor.2011.01.045

Theodor, I., 2003. Standardization of terminology. Mechanism and Machine Theory 38, 597–1111.

Tsai, L. W., 1999. Robot Analysis: The Mechanics of Serial and Parallel Manipulator. Wiley Interscience, Canada.

Utz, H., Sablatnög, S., Enderle, S., Kraetzchmar, G., 2002. Miro - middleware for mobile robot applications. IEEE Transactions on Robotics an 18, 493–497.