INNOVA Research Journal, ISSN 2477-9024
(Octubre, 2018). Vol. 3, No.10.1 pp. 184-194
DOI: https://doi.org/10.33890/innova.v3.n10.1.2018.754
URL: http://revistas.uide.edu.ec/index.php/innova/index
Correo: innova@uide.edu.ec

Utilización del middleware orocos para la implementación de controladores en tiempo real

Use of the orocos middleware to implement real-time controllers

Fabricio Manuel Tipantocta Pillajo.
SUCRE Instituto Superior Tecnológico, Ecuador

Ana Verónica Rodas Benalcázar
Oscar Iván Zambrano Orejuela
Escuela Politécnica Nacional, Ecuador

Autor para correspondencia: ftipantocta@tecnologicosucre.edu.ec
Fecha de recepción: 02 de agosto de 2018 - Fecha de aceptación: 15 octubre de 2018

Resumen: El presente artículo muestra la implementación de controladores en tiempo real mediante el middleware OROCOS. Como aplicación, en este trabajo se desarrolla el algoritmo de un controlador PID para el control automático de velocidad en lazo cerrado de un motor DC. Se evalúa el diseño de la planta experimentando con los tres controladores: el proporcional, proporcional-integral y el proporcional-integral-derivativo diseñados por software y se toma muestras de la variable controlada cada 10 milisegundos. Se determina la efectividad del controlador en tiempo real al momento que la variable controlada sigue inmediatamente al valor de la consigna en varias pruebas administradas, observando como resultado su media y su desviación estándar con respecto a los tres controladores por separado y en conjunto. El algoritmo en tiempo real que mejor se adaptó al control de velocidad fue un PID, el cual evidenció resultados óptimos de regulación con cada cambio experimental de la variable de consigna.

Palabras Claves: sistema de control; middleware; motor dc; pid

Abstract: The article shows how to use the OROCOS middleware as an instrument to perform controllers in real time and applying it to the model of a plant composed of a motor-generator set. The design of the plant is evaluated by experimenting with three types of controllers such as the proportional, proportional-integral and the proportional-integral-derivative designed by software and samples are taken every 10 milliseconds of the controlled variable, which in this case is the speed and so see their behavior. The effectiveness of the controller is looked at in real time at the moment that the controlled variable immediately follows the value of the set-point in several administered tests, observing as a result its mean and its standard deviation with respect to the three controllers, to know which one is better. The real-time algorithm that best adapted to the speed control was a PID, reacting adequately with each experimental change of the set-point variable.

Key Words: control systema; middleware; motor dc; orocos; pid

Introducción

La necesidad de sistemas de control con altas características impone una complejidad creciente en hardware, software y algoritmos. El conocimiento de software libre permite diseñar algoritmos de control confiables a la hora de integrar hardware y software. Por lo general, muchos controladores de robots comerciales así como otro tipo de sistemas, se basan en arquitecturas propietarias, lo que impide su integración en sistemas más complejos o incluso en la reutilización de software. (Ioris, Lages, & Santini, 2012)(Soetens & Bruyninckx, 2010)

El orocos (Open Robot Control Software) es un framework para la construcción de aplicaciones basadas en componentes y en tiempo real y para aplicaciones en sistemas de control, automatización y robótica. Desde el principio, proporcionó un lenguaje de scripting y una implementación de máquina de estado que permite implementar el comportamiento de un componente sin la necesidad de recompilarlo. Más importante aún, este lenguaje es seguros en tiempo real, lo que significa que la ejecución se lleva a cabo de manera temporalmente determinista. (Soetens & Bruyninckx, 2010) (Hernández Millán, Ríos Gonzales, & Bueno López, 2016)

La aplicación de técnicas avanzadas de control permite la manipulación de sistemas mecánicos complejos. Por ello existen diferentes métodos de control entre las que se pueden mencionar el PID, sistemas inteligentes, redes neuronales, algoritmos genéticos o lógica difusa, y herramientas computacionales que sirven para el desarrollo del cálculo matemático y simulación de las diferentes plantas a controlar. (Vallés, Cazalilla, Valera, Mata, & Page, 2013) (Ogata, 2010)

El objetivo de este proyecto es utilizar el middleware Orocos como instrumento para realizar controladores en tiempo real y aplicarlo en el modelo de una planta compuesto por un conjunto motor-generador.

Metodología

El diseño del proyecto integra el software necesario para aplicaciones de tiempo real, el modelo de la planta de un conjunto motor-generador, el algoritmo de control en el middleware Orocos, y la aplicación de hardware y software explicados en los siguientes apartados.

Sistema Operativo de Robot (ROS)

El Sistema Operativo de Robot (ROS) es un framework para la escritura de software de robots y funciona en Linux. Es una colección de herramientas, bibliotecas y convenciones que tienen como objetivo simplificar la tarea de crear comportamientos para robots complejos y robustos a través de una amplia variedad de plataformas robóticas. (Martinez & Fernández, 2013) En este proyecto se usa el sistema operativo de robots ROS para la instalación y el funcionamiento de la herramienta “Real Time Toolkit (RTT)” del software OROCOS, la cual permite crear componentes de software en tiempo real, sripts interactivos y genera los códigos del controlador en tiempo real (Acosta, Gallardo, & Pérez, 2016) (Sanz, n.d.).

Orocos Real Time Toolkit

Middleware Orocos (Open Robot Control Software), es un framework de software en código abierto y desarrollado en C++, para la implementación de algoritmos basados en componentes para aplicaciones en sistemas de automatización y robótica. (Martinez & Fernández, 2013) Orocos está estructurado en tres partes: 1) Librería de Cinemática y Dinámica (KDL); 2) Librería de Filtrado Bayesiano (BFL); 3) Orocos Toolchain RTT(Buys et al., n.d.) (Santini & Lages, 2010).

Orocos Toolchain es la principal herramienta para crear aplicaciones de robótica en tiempo real utilizando componentes de software configurables, el cual proporciona extensiones a otros frameworks robóticos, generadores de código para transferir datos definidos por el usuario entre componentes distribuidos, componentes programables y configurables en tiempo de ejecución y en tiempo real. (Vallés et al., 2013) (Comité Español de Automática CEA, 2011)

El controlador para la planta se realizó instalando previamente la herramienta Orocos RTT sobre el sistema operativo de robots ROS, en un ambiente de Ubuntu Linux.

Modelo de la planta

Para diseñar el controlador en tiempo real es primordial empezar con el modelo matemático de la planta a controlar. La planta escogida es un motor de corriente continua. El motor tiene parámetros tanto eléctricos como mecánicos y están descritas por las siguientes ecuaciones eléctricas y mecánicas respectivamente:

$$L\frac{di}{dt} + Ri = V - emf \quad (1)$$

$$. J_M \frac{d^2\theta}{d_t{}^2} + B\frac{d\theta}{d_t} + K\theta = T_L - J_L \frac{d^2\theta}{d_t{}^2} \quad (2)$$

El motor seleccionado tiene los siguientes parámetros: R = 6.4Ω, L=3 mH, J = $1.54x10^{-6}kgxm^2$, Kt = 0.0207 (Nxm)/A, Ke = 0.0206 (volt)/ (rad/s). En función de las características eléctricas y mecánicas del motor, se define la función de transferencia resultante del modelo de la planta como:

$$G(s) = \frac{53.906}{s(s+1.16)} \quad (3)$$

A este modelo de planta se le diseña el controlador para el control de la velocidad.

Diseño del controlador

Los sistemas de control en lazo cerrado se denominan también sistemas de control realimentados, como lo muestra la figura 3. En un sistema de control en lazo cerrado, se alimenta al controlador la señal de error de actuación, que es la diferencia entre la señal de entrada y la señal de realimentación, con el fin de reducir el error y llevar la salida del sistema a un valor deseado. El término control en lazo cerrado siempre implica el uso de una acción de control realimentado para reducir el error del sistema.

Según (Ogata, 2010) una unidad de control puede reaccionar de diversas maneras ante una señal de error y proporcionar determinadas señales de salida para que actúen los elementos correctores para lo cual se puede diseñar diferentes modos de control como:

- El modo proporcional
- El modo derivativo
- El modo integral.

Se puede combinar las acciones proporcionales y derivativas, proporcionales integrales y proporcionales, integrales y derivativas. A este último se le conoce como controlador de tres términos o PID. (Siegwart & Nourbakhsh, 2004) (Quigley, Gerkey, & Smart, 2015)

Controlador PID

Al combinar los tres modos de control (proporcional, derivativo e integral) se obtiene un controlador que no tiene desviación en el error y disminuye la tendencia a que se produzcan oscilaciones. La ecuación que describe su comportamiento es:

$$\text{PID(s)} = Kp\left[1 + \frac{1}{Tis} + Tds\right] \tag{4}$$

Discretizando la acción proporcional:

$$P(t_k) = k_p[r(t_k) - y(t_k)] \tag{5}$$

Acción integradora:

$$I(t_{k+1}) = I(t_k) + \frac{k_p h}{T_i} e(t_k) \tag{6}$$

Acción derivativa:

$$D(t_k) = \frac{T_d}{T_d + Nh} D(t_{k-1}) - \frac{k_p T_d N}{T_d + Nh}[y(t_k) - y(t_{k-1})] \tag{7}$$

El controlador discreto PID será la suma de los controladores anteriores:

$$u(t_k) = P(t_k) + I(t_k) + D(t_k) \tag{8}$$

El resultado de discretizar un sistema continuo, es para procesar la información adquirida por medio de una tarjeta de adquisición DAQ, y por medio de orocos se puede desarrollar el o los controladores necesarios de aplicación en tiempo real. El PID, como el algoritmo ya discretizado es:

Error=Setpoint-Realimentación;
P=Error*Kp;
I = I_0 + (Kp/Ti)*error_0*T;

D=(Kp*Td)*(error-error_0)/T;
PID=P+I+D;

Para el diseño del controlador se consideró un controlador PID discreto que se desarrolló con el método de aproximación rectangular, transformando las ecuaciones de la planta de tiempo continuo a tiempo discreto en el dominio Z. En la siguiente ecuación se expresa el algoritmo del PID con su fórmula general:

$$u(t) = K_p e(t) + K_i \int dt + K_d \frac{de}{dt} \quad (9)$$

Utilizando el método de aproximaciones la fórmula general del controlador PID queda de la siguiente manera:

$$u(n) = u(n-2) + K1e(n) + K2e(n-1) + K3e(n-2) \quad (10)$$

Donde las constantes están definidas como:

- K1 = Kp + Kd/T + Ki/T
- K2 =KiT – 2Kd/T
- K3 =Kd/T – Kp
- u(n) = señal de control en el intervalo de tiempo n
- u(n-2) = n-2 muestras
- e(n) = señal de error en el tiempo n
- e(n-1) = señal de error en el tiempo n-1
- T = Intervalo de muestreo

Utilizando el software Matlab, se discretiza la planta y con los parámetros de ganancia, derivación, e integración sumadas se coloca una perturbación randómica para así observar el modelo resultante controlado, el cual se muestra en la figura 1.

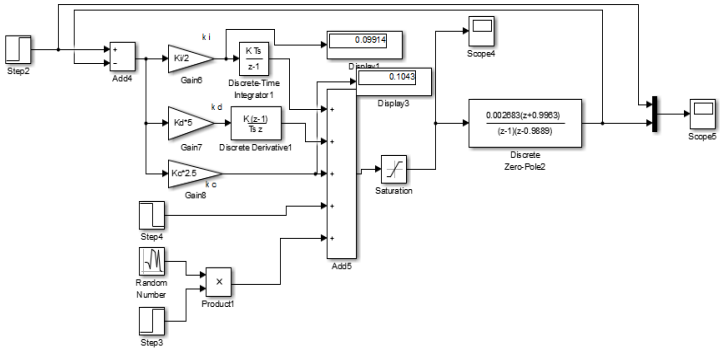

Figura 1. Controlador PID discreto.
Fuente: (Elaboración propia)

Se simula el control utilizando Simulink de Matlab, se analiza la respuesta a una señal escalón y se extrae las constantes necesarias con el auto sintonización del controlador PID, observando así el tiempo mejor de respuesta como muestra la figura 2.

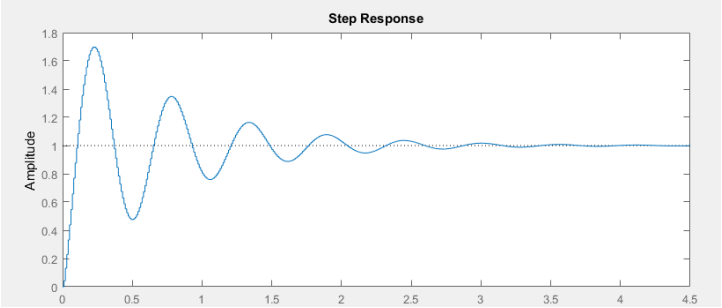

Figura 2. Respuesta de un controlador PID a una señal escalón.
Fuente: (Elaboración propia)

Una vez comprobado el algoritmo diseñado se procede a la implementación. Orocos efectúa un sistema en tiempo real determinista. No basta que las acciones del sistema sean correctas, sino que, además tienen que ejecutarse dentro de un intervalo de tiempo determinado. Debe tener fiabilidad y seguridad ya que un fallo en el sistema de control puede comportarse de forma peligrosa y sobre todo concurrencia en la ejecución de comandos, pues un sistema en tiempo real generalmente tiene interacción con dispositivos físicos (Villarroel, 2013).

Cuando cumple estos parámetros se puede llamar sistemas en tiempo real; Orocos está diseñado para cumplir con cada uno de las características mencionadas en el párrafo anterior, y el controlador diseñado para el proyecto se encuentra desarrollado con este framework. Se puede mencionar que Orocos asegura el trabajo en tiempo real.

El hardware del proyecto está compuesto por: 1) Tarjeta de adquisición PCI-1711; 2) Circuito conversor ac-dc; 3) Conjunto motor-generador, indicados en la figura 3.

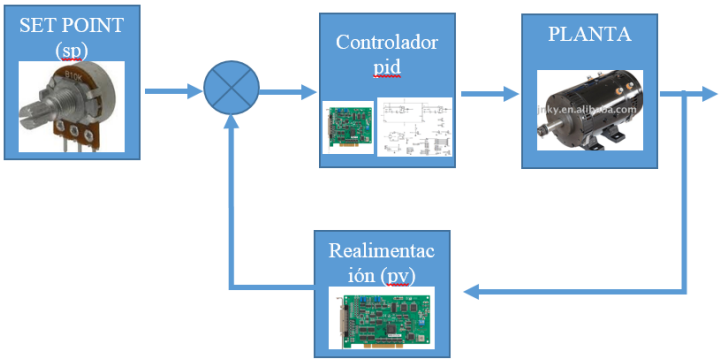

Figura 3. Sistema en lazo cerrado de control de proceso.
Fuente: (Elaboración propia)

Diseño del conversor ac-dc como regulador de voltaje

Como parte del proyecto se diseñó un conversor electrónico de potencia AC-DC, para la regulación de la velocidad del motor DC. (Rashid, 2002) Este conversor se conecta a una tarjeta de adquisición de datos de la marca ADVANTECH PCI-1711U, la cual sirve de enlace entre hardware y software.

El método escogido para manejar la velocidad del motor es control de fase con variación del ángulo alpha (α) entre 0 y 180°. El voltaje regulado aplicado al motor depende del ángulo de disparo y se relaciona con él mediante la siguiente ecuación:

$$V_{DC} = \frac{A}{T}(\cos(\alpha) + 1) \quad (11)$$

Resultados

Para las pruebas se utilizó la señal del Potenciómetro que representa la variable de consigna o set point que ingresa a un canal de la tarjeta de adquisición. Para adquirir la variable controlada se optó por el valor de voltaje que entrega el generador acoplado directamente al motor principal, este elemento entrega un valor de 0 a 1 voltio conectado a otro canal de la tarjeta de adquisición y escalada con la velocidad de 0 a 1000RPM.

Arranque del deployer

El deployer es el entorno de ejecución de los programas en tiempo real, se escribe en un terminal la instrucción rosrun rtt_ros deployer para iniciarlo, este es el entorno en el cual funciona el programa. Para las pruebas de los controladores en tiempo real como son el Control Proporcional (P), Proporcional-Integral (PI) y Proporcional-Integral-Derivativo (PID), se ha modificado las constantes Kp, Ti y Td. A continuación se presenta las pruebas realizadas con cada control seleccionado y el resultado obtenido.

Proporcional (P)

Cabe mencionar que, gracias al algoritmo se puede tomar muestras de datos de velocidad con periodos de 10ms. Para los experimentos se establece un valor determinado de consigna (sp). Por ejemplo 800 r.p.m, y al momento de energizar el sistema, la variable controlada (pv) debe seguir a la consigna, pero en el controlador proporcional intenta alcanzarlo sin lograrlo, porque el error estacionario en este tipo de controlador no le permite llegar al valor deseado. Se presenta en la figura 4 el resultado de la toma de muestras.

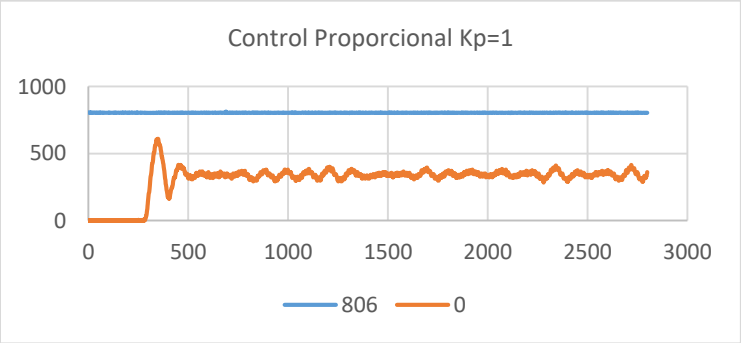

Figura 4. Pruebas con Control Proporcional con Kp=1.
Fuente: (Elaboración propia)

Proporcional - integral (PI)

El efecto de combinar las acciones proporcional e integral, es un cambio en la salida del controlador sin desviación en el error, pero mejora el resultado al momento de seguir la variable controlada a la consigna, se puede evidenciar con la figura 5.

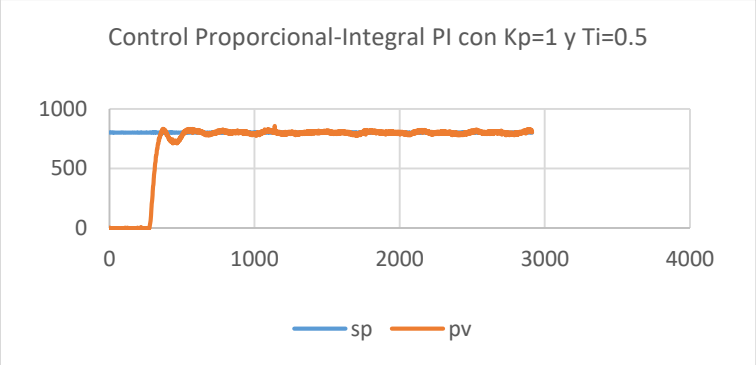

Figura 5. Pruebas con Control Proporcional-Integral con Kp=1 y Ti=0.5.
Fuente: (Elaboración propia)

Proporcional-integral-derivativo (pid)

Al combinar los tres modos de control (proporcional, derivativo e integral) se obtiene un controlador que no tiene desviación en el error y disminuye la tendencia a que se produzcan oscilaciones, se puede mostrar el resultado en la figura 6.

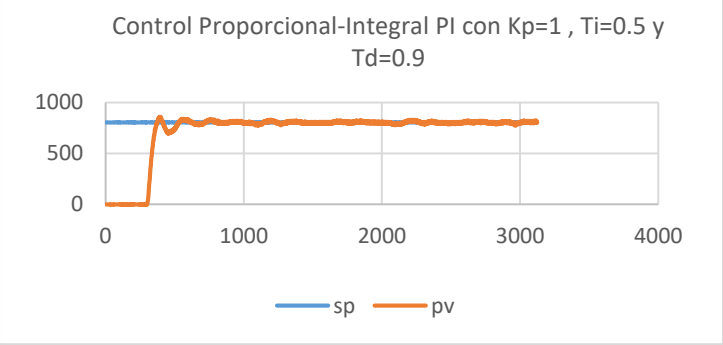

Figura 6. Pruebas con Control Prop-Int-Der con Kp=1,Ti=0.5 y Td=0.9.
Fuente: (Elaboración propia)

Se puede decir que las gráficas son similares con el anterior control, pero analizando los datos obtenidos entre los tres controladores se puede apreciar dos parámetros fundamentales, el valor medio y la desviación estándar de 2500 datos adquiridos y mostrados en la Tabla 1.

Tabla 1. Valor medio, desviación estándar y error de los tres controladores

Variable	Obs	Mean	Std. Dev.	Std. Err.
P	2500	338.873	61.811	2.223
PI	2500	758.078	155.72	3.082
PID	2500	786.649	134.18	1.655

Si se toma en cuenta el valor medio de los tres, el controlador PID con un valor de 786.6 r.p.m se aproxima al valor de consigna de 800 r.p.m. Este resultado muestra que el controlador PID maneja de mejor manera la planta ya que el error es 1.65 y mantiene la variable de entrada (velocidad) controlada.

Conclusiones

El proyecto desarrollado, es el primero de su clase en el Ecuador, la importancia se basa en la aplicación para sistemas de automatización y robótica en tiempo real, aventajando a sistemas comerciales, por el hecho de ser realizado en software libre y accesible a código reutilizable, utilizando como plataforma el sistema de software ROS y aplicando el framework Orocos para su implementación.

El controlador PID diseñado con algoritmos en el framework Orocos, convierte al proyecto en un sistema a tiempo real, conectando sin dificultad al hardware con software, y gracias al muestreo de datos, alrededor de los 10 milisegundos, se obtiene una gran cantidad de información que permitieron observar la eficiencia del controlador utilizado.

El sistema de control de velocidad implementado, responde a los controladores proporcionales, proporcional-integral, proporcional-integral-derivativo; sin embargo, la mejor respuesta se obtiene con el sistema de control PID.

Los sistemas en control en tiempo real no son muy conocidos en nuestro ámbito, ya que llamamos tiempo real a cualquier dato enviado por comunicación, y en realidad es un sistema determinista que observa las variables antes, dentro y después del proceso, siendo así un sistema eficiente que puede ayudar al momento de trabajar y asegurar sistemas de automatización, control y robótica.

Bibliografía

Acosta, G., Gallardo, J., & Pérez, R. (2016). Arquitectura de control reactiva para la navegación autónoma de robots móviles Reactive control architecture for autonomous mobile robot navigation, *24*(1), 173–181. https://doi.org/10.4067/S0718-33052016000

Buys, K., Bellens, S., Vanthienen, N., Decre, W., Klotzb, M., Laet, T. De, & Smits, R. (n.d.). Haptic coupling with the PR2 as a demo of the OROCOS - ROS - Blender integration.

Comité Español de Automática CEA. (2011). *Libro blanco de la robótica en España*. *Ministero de Ciencia e Innovación, Gobierno de España*. Retrieved from http://www.ceautomatica.es/sites/default/files/upload/10/files/Libro Blanco De La Robotica 2_v2.pdf

Hernández Millán, G., Ríos Gonzales, L. H., & Bueno López, M. (2016). Implementación de un controlador de posición y movimiento de un robot móvil diferencial. *Revista Tecnura*, *20*(48), 123–136. https://doi.org/10.14483/udistrital.jour.tecnura.2016.2.a09

Ioris, D., Lages, W., & Santini, D. (2012). Integrating the Orocos Framework and the Barrett Wam Robot. *Ece.Ufrgs.Br*, 1–8. Retrieved from http://www.ece.ufrgs.br/~fetter/robocontrol2012_98855_1.pdf

Martinez, A., & Fernández, E. (2013). *Learning ROS for Robotics Programming*. *Book*. https://doi.org/10.1017/CBO9781107415324.004

Ogata, K. (2010). *Ingeniería de control moderna* (5ta ed.). Perason. Prentice Hall.

Pal_robotic, Lauven, & FMTC. (2011). Recuperado el 6 de 11 de 2016, de The orocos project: http://www.orocos.org/rtt

Quigley, M., Gerkey, B., & Smart, W. D. (2015). *Programming Robots with ROS A Practical Introduction to the Robot Operating System*. *Journal of Chemical Information and Modeling* (Vol. 53). https://doi.org/10.1017/CBO9781107415324.004

Rashid, M. (2002). *Electrónica de potencia: Circuitos, Dispositivos y Aplicaciones* (Tercera). Florida, USA: Prentice Hall.

Santini, D. C., & Lages, W. F. (2010). An Architecture for Robot Control Based on the Orocos Framework. *Workshop De Robotica Aplicada E Automacao*, 1–10.

Sanz, J. L. C. O. E. P. R. (n.d.). Simulación de vehículos autónomos usando V-REP bajo ROS, 806–813.

Siegwart, R., & Nourbakhsh, I. R. (2004). *Introduction to Autonomous Mobile Robots*. *Robotica* (Vol. 23). https://doi.org/10.1109/ROBOT.2010.5509725

Soetens, P., & Bruyninckx, H. (2010). OROCOS RTT-Lua: an Execution Environment for building Real-time Robotic Domain Specific Languages. *Autonomous Robots*, 284–289.

Vallés, M., Cazalilla, J. I., Valera, Á., Mata, V., & Page, Á. (2013). Implementación basada en el middleware OROCOS de controladores dinámicos pasivos para un robot paralelo. *RIAI - Revista Iberoamericana de Automatica e Informatica Industrial*, *10*(1), 96–103. https://doi.org/10.1016/j.riai.2012.11.009

Villarroel, J. L. (2013). Sistemas de Tiempo Real. *Departamento de Informática e Ingeniería de Sistemas*. Zaragoza: Universidad de Zaragoza.