8º Congreso Nacional de Mecatrónica
Noviembre 26 y 27, 2009. Veracruz, Veracruz.

Evaluación de 4 Estructuras de Control Mediante un Simulador Basado en un Robot de 2 Grados de Libertad

José Guillermo Cebada Reyes, Jesús Tuxpan Meneses, Pablo Sánchez-Sánchez, Fernando Reyes-Cortés, Luz Carmen Gómez Pavón, Antonio Michua-Camarillo, Benjamín Calderón-Flores, Jaime Cid-Monjaraz, José Arturo Mancilla-Morales y Gabriela Morales-Timal
Universidad Autónoma de Puebla, F. C. E., Grupo de Robótica *oocelo*
lepable@ece.buap.mx y freyes@ece.buap.mx

Resumen **- El objetivo principal de este artículo es desarrollar una plataforma gráfica de simulación basada en un brazo robot de dos grados de libertad que evalúe el comportamiento de 4 estructuras de control pertenecientes a la familia de los controladores saturados; así como describir la metodología que se emplea en la elaboración del simulador.**

I. INTRODUCCIÓN

Una plataforma gráfica de simulación o simulador permite reproducir el comportamiento de un sistema mediante las ecuaciones matemáticas que lo describen, y brindar un ambiente gráfico de interpretación de datos dependientes de los estímulos internos y externos, para que el usuario final entienda dicho comportamiento. Por esta razón los simuladores son herramientas utilizadas en el diseño y creación de prototipos.

El diseño y desarrollo de sistemas mecatrónicos se puede dividir en tres etapas principales, el planteamiento y análisis matemático del sistema, la segunda etapa consiste en la comprobación de los cálculos mediante realización de un simulador; y finalmente la construcción del sistema. En este artículo se abordan las dos primeras etapas.

La evolución de los simuladores se ha dado en función a los avances en las computadoras y en los lenguajes de programación; los simuladores actualmente son una herramienta poderosa, exacta y amigable en la interacción con el usuario.

Es importante recalcar que un simulador es una herramienta que requiere del modelo dinámico del sistema y de una estructura de control que nos permita realizar correctamente la tarea asignada. En cambio, una animación son solamente fotogramas sucesivos que permiten brindar la sensación de movimiento.

A continuación se enumeran los pasos necesarios para desarrollar una plataforma gráfica de simulación:

- Análisis geométrico del sistema (cinemática directa e inversa).
- Propuesta y análisis de una estructura de control.
- Bosquejo del sistema utilizando software de diseño asistido por computadora (AutoCAD©).
- Programación del modelo dinámico y el control en una plataforma de programación (Visual C++©).
- Evaluación del simulador mediante el desarrollo de pruebas controladas.

Partiendo de estos pasos se analiza un robot de dos grados de libertad ROTRADI, figura 1.

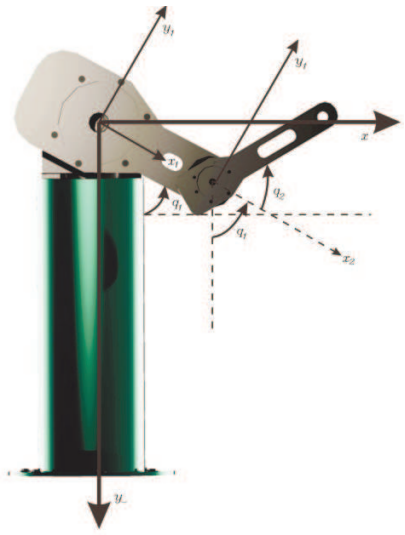

Figura 1. Robot ROTRADI

Usando la formulación de Euler-Lagrange, encontramos las ecuaciones de movimiento que describen al sistema considerando todos los parámetros involucrados en función al sistema de referencia.

El modelo dinámico de un robot manipulador describe el comportamiento de un sistema a un estímulo específico, ya sea este un estímulo interno o externo; esta representación matemática se obtiene aplicando leyes físicas para interpretar la dinámica del robot y está descrito como:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + f(\tau,\dot{q}) = \tau \qquad (1)$$

donde: $q,\dot{q},\ddot{q} \in \mathbb{R}^{n\times 1}$ son el vector de posición, velocidad y aceleración articular del robot, respectivamente; $M(q) \in \mathbb{R}^{n\times n}$ es la matriz de inercias. $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ es la matriz de Coriolis y fuerza centrípeta; $g(q) \in \mathbb{R}^{n\times 1}$ es el par gravitacional; $f(\tau,\dot{q}) \in \mathbb{R}^{n\times 1}$ es el vector de fricción; $\tau \in \mathbb{R}^{n\times 1}$ es el par aplicado.

El modelo dinámico requiere de los parámetros del fabricante para describir específicamente al robot manipulador deseado, en la siguiente tabla se listan los parámetros físicos del brazo robot de dos grados de libertad.

Tabla I.
PARAMETROS FÍSICOS

Descripción	Notación	Valor
Longitud del eslabón 1	l_1	0,4500 m
Longitud del eslabón 2	l_2	0,4500 m
Distancia al centro de masa 1	l_{c_1}	0,0910 m
Distancia al centro de masa 2	l_{c_2}	0,0480 m
Masa eslabón 1	m_1	23,902 kg
Masa eslabón 2	m_2	3,8800 kg
Momento de Inercia para eslabón 1	I_1	1,2660 kgm^2
Momento de Inercia para eslabón 2	I_2	0,0930 kgm^2
Aceleración de la gravedad	g	9,8100 $\frac{m}{s^2}$

I-A. Metodología de diseño de controladores

En esta sección se presenta la metodología utilizada para proponer estructuras de control y los resultados referentes al análisis de estabilidad de los controladores analizados.

I-A1. Moldeo de Energía

Típicamente para proponer controladores en espacio articular se utiliza el método de moldeo de energía. Para aplicar este método es necesario considerar el modelo dinámico con ausencia de fricción y otras perturbaciones cuyo esquema de control esta descrito como:

$$\tau = \nabla \mathcal{U}(k_p, \tilde{q}) - f_v(k_v, \dot{q}) + g(q) \qquad (2)$$

donde $\mathrm{U}(k_p, \tilde{q})$ es la energía potencial artificial que depende de kp y está definida por:

$$\mathcal{U}(k_p, \tilde{q}) = \frac{f(\tilde{q})^T K_p f(\tilde{q})}{2} \qquad (3)$$

El término $f_v(k_v, \dot{q})$ es una *función derivativa* que depende de k_v k_v (función de amortiguamiento) cuya expresión matemática es:

$$f_v(k_v, \dot{q}) = q^T f_v(k_v, \dot{q}) > 0 \qquad (4)$$

Además, se debe considerar una función de Lyapunov de la forma:

$$V(\dot{q}, \tilde{q}) = \frac{\dot{q}^T M(q) \dot{q}}{2} + \mathcal{U}(k_p, \tilde{q}) \qquad (5)$$

donde $\mathrm{U}(k_p, \tilde{q})$ es la *energía potencial artificial*.

El método de moldeo de energía consiste en encontrar una función $\mathrm{U}(k_p, \tilde{q})$ que cumpla con las condiciones de la función de Lyapunov:

$$\begin{array}{lll} V(0,0) = 0 & \forall & \dot{q}, \tilde{q} = 0 \\ V(\dot{q}, \tilde{q}) > 0 & \forall & \dot{q}, \tilde{q} \neq 0 \end{array} \qquad (6)$$

I-A2. Controladores evaluados

Las estructuras de control que cumplen con las características antes mencionadas son los siguientes:

- $\tau = K_p \tilde{q} - K_v \dot{q} + g(q)$
- $\tau = K_p \tanh(\tilde{q}) - K_v \tanh(\dot{q}) + g(q)$
- $\tau = K_p \arctan(\tilde{q}) - K_v \arctan(\dot{q}) + g(q)$
- $\tau = K_p \Psi_{\tilde{q}} - K_v \Psi_{\dot{q}} + g(q)$ donde: $\Psi_{\tilde{q}} = \dfrac{\operatorname{senh}(\tilde{q})}{\sqrt{1 + \operatorname{senh}^2(\tilde{q})}}$ y $\Psi_{\dot{q}} = \dfrac{\operatorname{senh}(\dot{q})}{\sqrt{1 + \operatorname{senh}^2(\dot{q})}}$

II. REPRESENTACIÓN GRÁFICA

Después de analizar el sistema y obtener el modelo dinámico y la estructura de control, el siguiente paso consiste en bosquejar el prototipo a través de sus vistas frontal, lateral y superior, las cuales nos sirven para tener una referencia del sistema en el plano. Entre más detallado este el dibujo obtendremos una mejor representación gráfica del sistema. El diseño asistido por computador (Computer Aided Design por sus siglas en inglés), es el uso de un amplio rango de herramientas computacionales que se utilizan para la creación de entidades geométricas. La herramienta CAD que se utiliza en el diseño del simulador es AutoCAD©, programa de diseño para dibujo en 2D y 3D desarrollado por la empresa Autodesk©. Los dibujos que definen al brazo robot de dos grados de libertad ROTRADI (vista frontal, lateral y superior) se encuentran en centímetros.

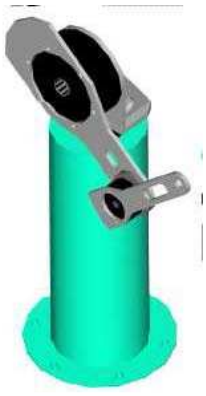

Figura 2. Representación gráfica en AutoCAD©

Una vez que se bosqueja el prototipo tridimensionalmente se procede a definir el simulador pieza por pieza, ya que el movimiento del prototipo está en referencia a los eslabones. Por tal motivo es necesario identificar piezas clave que funcionen como referencia para un movimiento por bloques. Definidas todas las piezas clave se procede a transformarlas en una estructura o fichero de datos que represente mediante una rejilla rectangular de pixeles o puntos de color un formato de imagen (BITMAP, PGM o 3DS).

Al realizar esta transformación, los componentes de la imagen (cabecera, identificador, tamaño y cuerpo de la imagen) se encuentran en un archivo encriptado en código ASCII o binario. En este caso ocupamos el formato 3DS debido que el programa AutoCAD© permite transformar imágenes DWG a 3DS.

III. PROGRAMACIÓN DEL SISTEMA

El siguiente paso consiste en relacionar mediante un lenguaje de programación las diferentes entidades tanto gráficas como matemáticas para poder describir el comportamiento del prototipo.

III-A. OpenGL

OpenGL es una especificación estándar que define una Interfaz de Programación de Aplicaciones API (Application Programming Interface) multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. Fue desarrollada por Silicon Graphics Inc. en 1992. Su nombre viene del inglés Open Graphics Library, cuya traducción es librería de gráficos libre.

A grandes rasgos, OpenGL es una especificación, es decir, un documento que describe un conjunto de funciones y su comportamiento exacto. A partir de estas especificaciones se crean implementaciones (bibliotecas de funciones creadas para enlazar con las funciones de la especificación OpenGL). La compatibilidad de OpenGL abarca desde Borland C, Visual C, Java, visual Fortran entre otros. El dibujo en OpenGL se basa en la composición de pequeños elementos, con los que vamos construyendo la escena deseada, estos elementos se llaman *primitivas*. Todas las primitivas de OpenGL son objetos de una o dos dimensiones, abarcando desde puntos simples a líneas o polígonos complejos. Para definir el área de trabajo en OpenGL hay que imaginar un volumen con perspectiva ortonormal, como se puede observar en la figura.

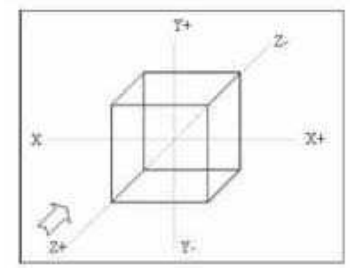

Figura 3. Primitivas de OpenGL

Es importante ubicar el sólido, una vez hecho esto, se verifica con 3D studio el origen del objeto y se procede a crear un nuevo proyecto en Visual C++, tomando en cuenta que debe estar incluida la librería glut.dll en la carpeta de *system* y *system32* de windows; posteriormente en el proyecto se incorporan las librerías de acuerdo al diagrama a bloque de la figura siguiente.

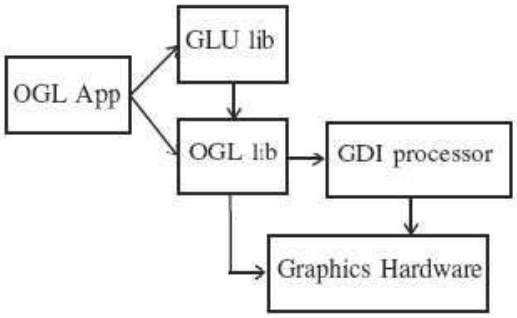

Figura 4. Procedimiento de cargado de librerías

III-B. Visual C++.

Visual C++ es un lenguaje de programación, está especialmente diseñado para el desarrollo y depuración de código escrito para las API's de Microsoft Windows, DirectX y la tecnología Microsoft .NET Framework.

Existen ciertas herramientas en Visual C++ para construir un simulador como la clase, la cual contiene atributos, métodos, mensajes y eventos. Para comprender lo antes expuesto debemos tener en claro algunas definiciones como:

- **Objetos**: Un programa orientado a objetos se compone solamente de objetos, donde este es una encapsulación genérica de datos y de los procedimientos para manipularlos.
- **Mensajes**: Cuando un programa orientado a objetos se ejecuta, los objetos están recibiendo, interpretando y respondiendo a mensajes de otros objetos, lo que origina cambios en el estado del objeto.
- **Métodos**: Un método se implementa en una clase y determina como tiene que actuar el objeto cuando recibe un mensaje.
- **Clases**: Una clase se puede considerar como una plantilla para crear objetos de esa clase o tipo; esta describe los métodos y los atributos que definen las características comunes a todos los objetos de esa clase.

III-C. Cargar el modelo dinámico

Para cargar el modelo dinámico en Visual C++ es necesario crear una nueva clase, la cual se denomina modelo dinámico, en esta clase

utilizamos el algoritmo de Runge-Kutta de Cuarto Orden. Finalmente el ambiente del simulador es el siguiente:

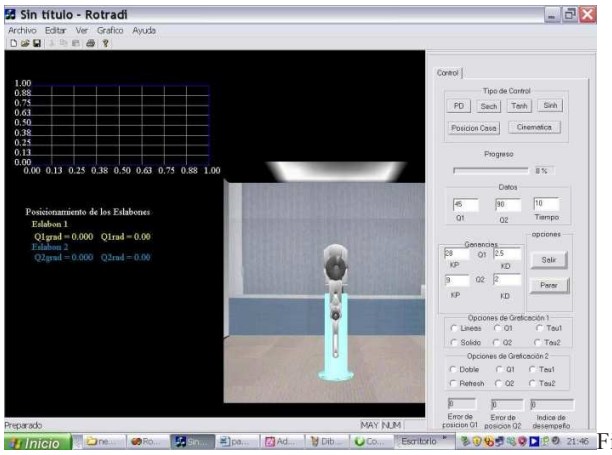

Figura 5. Ambiente principal

III-D. Pruebas del simulador

Para las prueba del simulador se utiliza las siguientes ganancias Kv1=2.5 y Kp1=28 para el eslabón 1 o bien el eslabón del hombro; y para el eslabón 2 o bien el eslabón del codo tiene una ganancia de Kv2=2 y Kv2=9.

III-D1. Prueba 1: Controlador PD

Utilizando la plataforma gráfica de simulación desarrollada se evalúa el comportamiento del controlador PD, en la figura 6 se observan las curvas de comportamiento y el movimiento realizado por el brazo robot de dos grados de libertad.

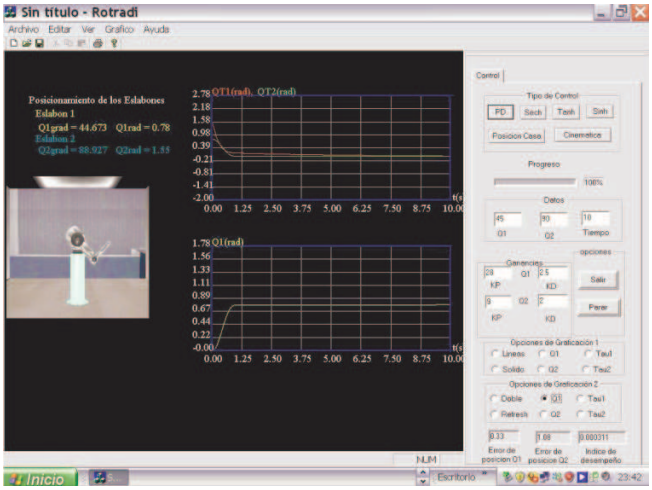

Figura 6. Control PD

El simulador desarrollado muestra tanto el índice de desempeño como las gráficas del error de posición, figura 7

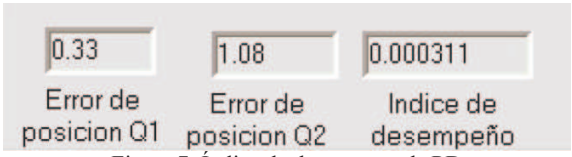

Figura 7. Índice de desempeño de PD

III-D2. Prueba 2: Controlador tanh

A continuación se evalúa el control tanh utilizando el simulador desarrollado.

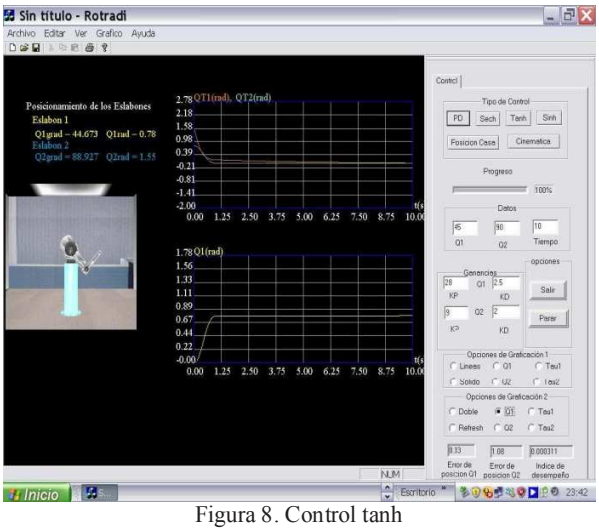

Figura 8. Control tanh

Cuyo índice de desempeño y error de posición es:

Figura 9. Índice de desempeño de tanh

III-D3. Prueba 3: Control arctan

La siguiente estructura de control que se evalúa es la arctan, el simulador muestra las gráficas de comportamiento y en la figura 10 se puede observar como alcanza la posición deseada. Es importante destacar que para este tipo de funciones se debe de considerar el error de sobre-flujo (overflow) que se pudiera generar al resolver el sistema no lineal, por lo cual es necesario acotar la función.

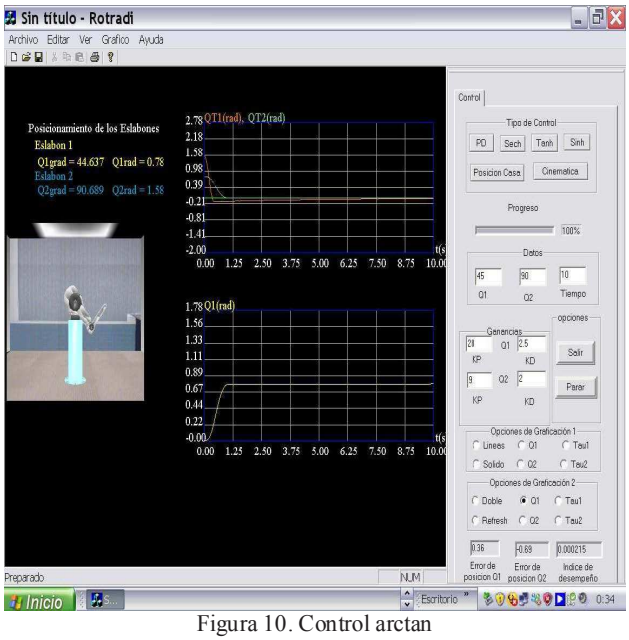

Figura 10. Control arctan

Cuyo índice de desempeño y error de posición es:

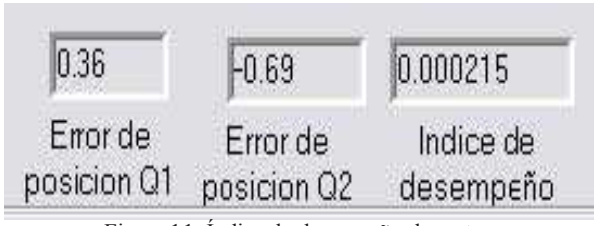

Figura 11. Índice de desempeño de arctan

III-D4. Prueba 4: Control sin

Finalmente podemos observar el comportamiento y desempeño del controlador sin.

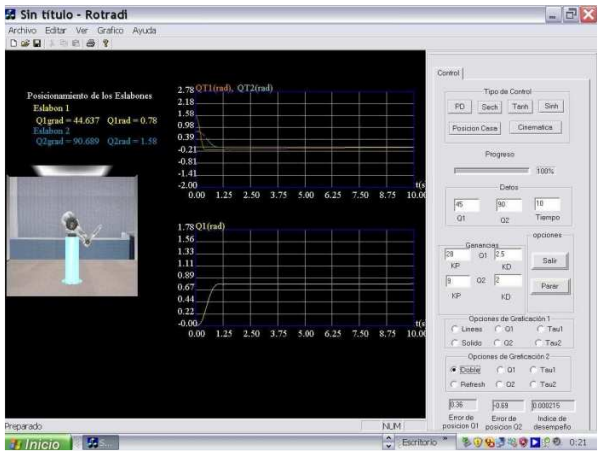

Figura 12. Control sin

El índice de desempeño y el error de posición se muestran a continuación:

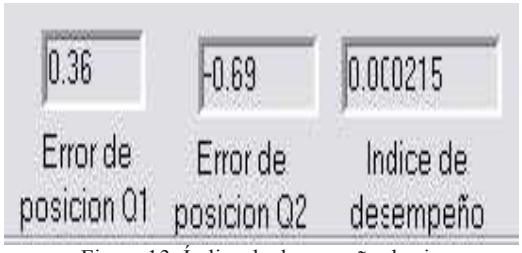

Figura 13. Índice de desempeño de sin

IV. CONCLUSIONES

Desarrollar una plataforma gráfica de simulación o simulador orientado específicamente a un sistema, nos permite evaluar rápidamente diversas estructuras de control, las cuales nos permiten comprender el comportamiento de las diversas funciones que nos permiten controlar el sistema. Un simulador es una herramienta idónea en la evaluación, diseño, desarrollo y construcción de sistemas mecatrónicos pues nos permite evaluar el modelo dinámico y las estructuras de control aun cuando no se cuente con la planta física.

REFERENCIAS

[1] J. J. Craig, Introduction to Robotics Mechanics and Control, (New York: Addison-Wesley, 1989).
[2] F. Reyes, J Cid and C. Campuzano, Development of an Experimental Platform with open architecture for Robots Manipulators, Universidad Autónoma de Puebla, Puebla Mexico.
[3] A. Lona, On Output Feedback Control of Euler-Lagrange, (1996).
[4] H. Goldstein, Classical Dynamics, (Reading, MA. Addison-Wesley, 1950).
[5] A. Barrientos, L. F. Peñin, C. Balaguer and R. Aracil, Fundamentos de Robótica, (Madrid, España: McGraw Hill, 1997).
[6] L. Sciavicco and B. Siciliano, Modeling and Control of Robot Manipulators, (Napoles, Italia: McGraw Hill, 1996).
[7] M. Takegaki and S. Arimoto, A NewFeedback Method for Dynamic Control of Manipulators, Journal of Dynamic System, Measurement and Control, 102(2), 1981, 119-125.
[8] R. Kelly and V. Santivañez, A class of global regulators with bounded control actions for robot manipulators, Proceedings of the 35th conference on decision and control, Kobe, Japan, 1996, 3382-3387.
[9] R. Kelly, R. Haber, R. Haber-Guerra and F. Reyes, Lyapunov Stable Control of Robot Manipulators: A Fuzzy Self-Tunning Procedure, Intelligent Automation and Soft Computing, 5(4), 1999, 313-326.
[10] F. Reyes and C. Campuzano, PD-Type Controller with Nonlinear Proporcional Gain for Robot Manipulators, XX Congreso Internacional Académico de Ingeniería Electrónica, Puebla, México, 1998, 357-360.
[11] Dorado de la Calle Julian, Una Aproximacion a OpenGL, Universidad de Cataluña España.
[12] Neider Jackie, OpenGL Programming Guide (Reed Book), Addison Wesley Publishing Company, Massachusetts Institute of Technology.
[13] Ceballos Sierra Fco. Javier, Programación Orientada a Objetos con C++, México DF: Alfaomega Ra-Ma, 1998, Capítulos 3,6,10,17.
[14] A. Viguria, A. Prieto, M. Fiacchini, R. Cano, F. R. Rubio, J. Aracil, C. Canudas-de-Wit, Desarrollo y Experimentación de un Vehículo Basado en Péndulo Invertido (PPCAR), Departameto de Ingeniería de Sistemas y Automática, Universidad de Sevilla, España, Laboratoire d' Autometique de Grenoble (CNRS-LAG), Francia.