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

Diseño de una Plataforma Gráfica de Simulación Basada en un Péndulo en Configuración Furuta

José Arturo Mancilla-Morales, Pablo Sánchez-Sánchez, Fernando Reyes-Cortés, Antonio Michua-Camarillo,
Benjamín Calderón-Flores, Jaime Cid-Monjaraz, Enrique López-Pérez y Gabriel Romero-Rodríguez
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 evaluar el comportamiento de un péndulo en configuración Furuta usando una plataforma gráfica de simulación, la cual es desarrollada mediante la programación orientada a objetos y el motor gráfico OpenGL©. El simulador que se desarrolla está basado en el prototipo de la empresa Quanser©.**

I. INTRODUCCIÓN

El péndulo invertido es un sistema que constituye un banco de pruebas muy interesante en la teoría de control; cuyo objetivo consiste en situar el péndulo en la posición vertical invertida de forma autónoma y hacer que permanezca en esa posición. El problema de control, considerado local, reside en intentar estabilizar el sistema en una posición inestable en lazo abierto, lo que constituye un problema de control muy notable; este problema por su carácter local puede resolverse con métodos lineales; y así se ha hecho desde los años 60's [1, 2]. El interés por controlar este tipo de sistemas origina la aparición de diversas modificaciones, una de ella es el péndulo invertido sobre base móvil, cuyas limitaciones originan la aparición del péndulo en configuración Furuta cuyas características estructurales resuelven el problema de fin de carrera propio del péndulo invertido sobre base móvil. El péndulo en configuración Furuta es propuesto por Katsuhisa Furuta del Instituto de Robótica de Tokio en el año 1970, esta aportación consiste en un péndulo rotatorio que cuenta con un motor de eje vertical unido a un brazo de cuyo extremo cuelga el péndulo. Con este artificio se evitan los problemas de final de carrera aunque aparecen otros nuevos debido a que la dinámica del sistema, una de estas problemáticas es la aparición de la fuerza centrífuga y de Coriolis [3]. El problema que ofrece este nuevo sistema es más general y complejo, el cual consiste en llevar el péndulo desde cualquier posición hasta la posición invertida, problema también conocido como swing-up [4]. Este artículo está distribuido de la siguiente forma, en la sección 2 se presenta la descripción matemática del sistema; en la sección 3 se incluye la propuesta de control, la cual cuenta con la debida comprobación de estabilidad en términos de Lyapunov; en la sección 4 se describe la metodología utilizada para elaborar la plataforma gráfica de simulación, en la siguiente sección se evalúa el comportamiento del simulador; y finalmente en la sección 6 se presentan las conclusiones del trabajo.

II. MODELO DEL SISTEMA

Un método ampliamente utilizado para obtener el modelo dinámico de un sistema es el método de Euler-Lagrange [6], el cual utiliza la diferencia entre la energía cinética $\mathrm{K}(q,\dot{q})$ y la energía potencial $\mathrm{U}(q)$,

$$\mathcal{L}(q,\dot{q}) = \mathcal{K}(q,\dot{q}) - \mathcal{U}(q), \tag{1}$$

término que se conoce como Lagrangiano.

La energía cinética es la suma de la energía de traslación más la energía de rotación,

$$\mathcal{K}(q,\dot{q}) = \frac{1}{2}mv^2 + \frac{1}{2}I\dot{q}^2, \tag{2}$$

y la energía potencial como:

$$\mathcal{U}(q) = mgh. \tag{3}$$

Partiendo de estas consideraciones al resolver la ecuación de Euler-Lagrange para un sistema conservativo, un sistema conservativo es un sistema mecánico en que la energía mecánica se mantiene, la ecuación se define como [6]:

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}(q,\dot{q})}{\partial(\dot{q})}\right] - \frac{\partial \mathcal{L}(q,\dot{q})}{\partial q} = \tau - f(\tau,\dot{q}) \tag{4}$$

donde q es un vector de coordenadas generalizadas, τ es un vector de fuerzas y pares aplicados en las coordenadas generalizadas y el Lagrangiano $\mathrm{L}(q,\dot{q})$ es la diferencia entre la energía cinética y la potencial [6]

$$\mathcal{L}(q,\dot{q}) = \mathcal{K}(q,\dot{q}) - \mathcal{U}(q); \tag{5}$$

obtenemos el modelo dinámico del robot manipulador; en forma más general, obtenemos el modelo para un robot de n grados de libertad. La estructura del modelo dinámico es la siguiente [6]:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + f(\tau,\dot{q}) + \tau_d = \tau \tag{6}$$

donde $q,\dot{q},\ddot{q} \in R^{n\times 1}$ son el vector de posición, velocidad y aceleración articular del robot, respectivamente, $M(q) \in R^{n\times n}$ es la matriz de Inercias, $C(q,\dot{q}) \in R^{n\times n}$ es la matriz de Coriolis y fuerza Centrípeta, $g(q) \in R^{n\times 1}$ es el par gravitacional, $f(\tau,\dot{q}) \in \square^{n\times 1}$ es el vector de fricción, $\tau \in \square^{n\times 1}$ es el par aplicado y $\tau_d \in \square^{n\times 1}$ representa las perturbaciones externas [5,6]. En ausencia de la fricción y otras perturbaciones, el modelo dinámico para un robot rígido de n grados de libertad es el siguiente [3]:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau. \tag{7}$$

II-A. Modelo del péndulo Furuta

El péndulo de configuración Furuta se muestra en la figura 1:

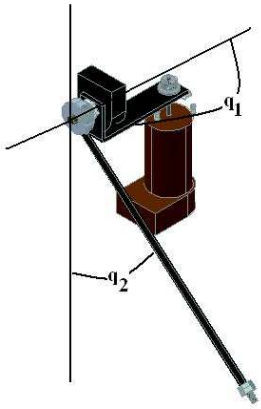

Figura 1. Péndulo en configuración Furuta

Se observa que el primer elemento (brazo) tiene un giro alrededor de un eje perpendicular a la base, mientras que el segundo elemento (péndulo) se encuentra colocado en un extremo del brazo y su eje de giro es co-lineal al eje axial del brazo, realizando un giro en un plano perpendicular al brazo. Por lo tanto la cinemática para el brazo es:

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} l_1 \cos(q_1) \\ l_1 \sin(q_1) \\ -l_2 \end{bmatrix} \tag{8}$$

Para el péndulo tenemos:

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} l_1 \cos(q_1) \\ l_1 \sin(q_1)\cos(q_2) + l_2 \sin(q_2) \\ l_1 \sin(q_1)\sin(q_2) - l_2 \cos(q_2) \end{bmatrix} \tag{9}$$

Después de obtener la velocidad, las energías cinética y potencial, el Lagrangiano y resolver la ecuación de Euler-Lagrange obtenemos el modelo dinámico del sistema:

$$M(q) = \begin{bmatrix} J_0 + m1(L_0^2 + l_1^2 \sin^2(q_2)) & m_1 L_0 l_1 \cos(q_2) \\ m_1 L_0 l_1 \cos(q_2) & J_1 + m_1 l_1^2 \end{bmatrix} \tag{10}$$

$$C(q,\dot{q}) = \begin{bmatrix} m_1 l_1^2 S(q_2) C(q_2)\dot{q}_2 + d_0 & m_1 l_1^2 S(q_2) C(q_2)\dot{q}_1 \\ -m_1 l_1^2 S(q_2) C(q_2)\dot{q}_1 & -m_1 L_0 l_1 S(q_2)\dot{q}_2 \\ & d_1 \end{bmatrix} \tag{11}$$

$$g(q) = \begin{bmatrix} 0 \\ -m_1 l_1 g \sin(q_2) \end{bmatrix} \tag{12}$$

donde q_1 es el ángulo que toma el brazo, q_2 es el ángulo que toma el péndulo, l_0 es la longitud del brazo, l_1 es la longitud desde el grado de libertad hasta el centro de masa del péndulo, m_1 es la masa del péndulo, J_0 es la inercia del brazo, J_1 es la inercia del péndulo, d_0 y d_1 son la viscosidad de cada articulación.

III. ESTRUCTURA DE CONTROL

Para garantizar que un sistema realice la tarea para el cual fue diseñado es necesario conocer las expresiones matemáticas que lo describen y la estructura de control que proporciona el torque o par aplicado necesario para realizar dicha tarea; por lo cual en esta sección se detalla la metodología empleada para proponer esquemas de control. El cálculo del vector τ involucra generalmente a una función vectorial no lineal de q, $\dot{q}$ y $\ddot{q}$. Esta función se denomina ley de control o simplemente controlador. Este artículo se basa en el control de posición, el cual consiste en llevar el efector final del robot desde una posición inicial q_0 a una posición deseada q_d, es decir, encontrar un par aplicado τ tal que la posición real tienda a la posición deseada cuando el tiempo evoluciona hacia el infinito.

III-A. Control PD

El estudio formal del controlador PD fue realizado en 1981 por Morikazu Takagaki y Suguru Arimoto, cuya estructura matemática es:

$$\tau = K_p\tilde{q} - K_v\dot{q} + g(q) \tag{13}$$

donde $K_p, K_v \in \mathbb{R}^{n\times n}$ son matrices simétricas definidas positivas, $\tilde{q} = q_d - q \in \mathbb{R}^{n\times 1}$ es el error de posición, $q_d \in \mathbb{R}^{n\times 1}$ es la posición deseada y $q \in \mathbb{R}^{n\times 1}$ la posición real. Para realizar la comprobación de estabilidad se relaciona la estructura de control con el modelo dinámico (6); así tenemos:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} = K_p\tilde{q} - K_v\dot{q} \tag{14}$$

El siguiente paso es obtener la ecuación del sistema en lazo cerrado:

$$\frac{d}{dt}\begin{bmatrix} \tilde{q} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -\dot{q} \\ M(q)^{-1}[K_p\tilde{q} - K_v\dot{q} - C(q,\dot{q})\dot{q}] \end{bmatrix} \tag{15}$$

Ahora la función candidata de Lyapunov, puede expresarse como:

$$V(\dot{q},\tilde{q}) = \frac{\dot{q}^T M(q)\dot{q}}{2} + \frac{\tilde{q}^T K_p \tilde{q}}{2} \tag{16}$$

derivando con respecto al tiempo $\dot{V}(\dot{q},\tilde{q})$, es:

$$\dot{V}(\dot{q},\tilde{q}) = \dot{q}^T M(q)\ddot{q} + \frac{\dot{q}^T \dot{M}(q)\dot{q}}{2} + \dot{q}^T K_p \dot{\tilde{q}} \qquad (17)$$

Al sustituir $M(q)\ddot{q}$ de la ecuación de lazo cerrado (15) se obtiene:

$$\dot{V}(\dot{q},\tilde{q}) = -\dot{q}^T K_v \dot{q} + \frac{1}{2}\dot{q}^T[\dot{M}(q) - 2C(q,\dot{q})]\dot{q} \qquad (18)$$

utilizando la propiedad $\frac{1}{2}\dot{q}^T[\dot{M}(q) - 2C(q,\dot{q})\dot{q}]\dot{q} \equiv 0$ se simplifica:

$$\dot{V}(\dot{q},\tilde{q}) = -\dot{q}^T K_v \dot{q} \leq 0 \qquad (19)$$

la función $V(\dot{q},\tilde{q})$ es semidefinida negativa $\dot{V}(\dot{q},\tilde{q}) \leq 0$. Por el teorema de LaSalle se demuestra la estabilidad asintótica del punto de equilibrio en el origen del controlador PD con compensación de gravedad. Se define un conjunto como:

$$\begin{aligned} \Omega &= \{\dot{q} \in \mathbb{R}^n, \tilde{q} \in \mathbb{R}^n : \dot{V}(\dot{q},\tilde{q}) = 0\} \\ \Omega &= \{\dot{q} \in \mathbb{R}^n, \tilde{q} = 0 \in \mathbb{R}^n\} \end{aligned} \qquad (20)$$

por lo tanto $V(\dot{q},\tilde{q})$ es una función definida positiva con un único y mínimo punto en $\begin{bmatrix} q^T & \dot{q}^T \end{bmatrix} = 0$, se concluye que el conjunto invariante más grande que está en Ω es el origen.

III-B. Control tanh

El controlador de posición con compensación de gravedad denominado *tanh* fue propuesto originalmente por Cai, L. y Song, G. y reformulado más tarde por Kelly, R., Santibáñez, V. y Reyes, F., este controlador pertenece a la familia de los controladores saturados cuya característica es que convergen a cierta cota a medida que el argumento crece indefinidamente, estas funciones son radialmente acotadas. Considérese la siguiente estructura de control:

$$\tau = K_p \tanh(\tilde{q}) - K_v \tanh(\dot{q}) + g(q) \qquad (21)$$

donde:

$$\tanh(\tilde{q}) = \begin{bmatrix} \tanh(\tilde{q}_1) \\ \tanh(\tilde{q}_2) \\ \vdots \\ \tanh(\tilde{q}_n) \end{bmatrix} \qquad (22)$$

Sustituimos la ecuación (21) en la ecuación de lazo cerrado:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} = K_p \tanh(\tilde{q}) - K_v \tanh(\dot{q}) \qquad (23)$$

obtenemos la ecuación del sistema en lazo cerrado:

$$\frac{d}{dt}\begin{bmatrix} \tilde{q} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -\dot{q} \\ M(q)^{-1}[K_p \tanh(\tilde{q}) - K_v \tanh(\dot{q}) - C(q,\dot{q})\dot{q}] \end{bmatrix} \qquad (24)$$

cuya función candidata de Lyapunov es:

$$V(\dot{q},\tilde{q}) = \frac{\dot{q}^T M(q)\dot{q}}{2} + \begin{bmatrix} \sqrt{\ln(\cosh(\tilde{q}_1))} \\ \sqrt{\ln(\cosh(\tilde{q}_2))} \\ \vdots \\ \sqrt{\ln(\cosh(\tilde{q}_n))} \end{bmatrix}^T K_p \begin{bmatrix} \sqrt{\ln(\cosh(\tilde{q}_1))} \\ \sqrt{\ln(\cosh(\tilde{q}_2))} \\ \vdots \\ \sqrt{\ln(\cosh(\tilde{q}_n))} \end{bmatrix} \qquad (25)$$

al derivarla con respecto al tiempo $\dot{V}(\dot{q},\tilde{q})$, queda:

$$\dot{V}(\dot{q},\tilde{q}) = \dot{q}^T M(q)\ddot{q} + \frac{\dot{q}^T \dot{M}(q)\dot{q}}{2} + \begin{bmatrix} \sqrt{\ln(\cosh(\tilde{q}_1))} \\ \sqrt{\ln(\cosh(\tilde{q}_2))} \\ \vdots \\ \sqrt{\ln(\cosh(\tilde{q}_n))} \end{bmatrix}^T K_p \begin{bmatrix} \frac{\tanh(\tilde{q})}{\sqrt{\ln(\cosh(\tilde{q}))}} \end{bmatrix} \dot{\tilde{q}} \qquad (26)$$

donde $\begin{bmatrix} \frac{\tanh(\tilde{q})}{\sqrt{\ln(\cosh(\tilde{q}))}} \end{bmatrix} \in \mathbb{R}^{n\times n}$ y $K_p \in \mathbb{R}^{n\times n}$ y tiene la forma:

$$\begin{bmatrix} \frac{\tanh(\tilde{q})}{\sqrt{\ln(\cosh(\tilde{q}))}} \end{bmatrix} = \begin{bmatrix} \frac{\tanh(\tilde{q}_1)}{\sqrt{\ln(\cosh(\tilde{q}_1))}} & 0 & 0 & 0 \\ 0 & \frac{\tanh(\tilde{q}_2)}{\sqrt{\ln(\cosh(\tilde{q}_2))}} & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \frac{\tanh(\tilde{q}_n)}{\sqrt{\ln(\cosh(\tilde{q}_n))}} \end{bmatrix} \qquad (27)$$

sustituimos $M(q)\ddot{q}$ de la ecuación de lazo cerrado (24) se obtiene:

$$\dot{V}(\dot{q},\tilde{q}) = -\dot{q}^T K_v \begin{bmatrix} \tanh(\dot{q}_1) \\ \tanh(\dot{q}_2) \\ \vdots \\ \tanh(\dot{q}_n) \end{bmatrix} + \frac{1}{2}\dot{q}^T[\dot{M}(q) - 2C(q,\dot{q})]\dot{q} \qquad (28)$$

aplicamos la propiedad $g(q) = \frac{\partial U(q)}{\partial q}$ se simplifica a:

$$\dot{V}(\dot{q},\tilde{q}) = -\dot{q}^T K_v \begin{bmatrix} \tanh(\dot{q}_1) \\ \tanh(\dot{q}_2) \\ \vdots \\ \tanh(\dot{q}_n) \end{bmatrix} \leq 0 \qquad (29)$$

la función $V(\dot{q},\tilde{q})$ es entonces una función de Lyapunov dado que:

$$\dot{V}(\dot{q},\tilde{q}) \leq 0 \qquad (30)$$

obteniendo estabilidad asintótica global con el empleo del teorema de LaSalle:

$$\dot{V}(\dot{q},\tilde{q}) < 0 \qquad (31)$$

IV. PLATAFORMA DE SIMULACIÓN

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 *(x, y, z)*. Entre más detallado este el dibujo obtendremos una mejor representación gráfica del sistema. Para realizar el bosquejo del sistema se utiliza el programa AutoCAD©.

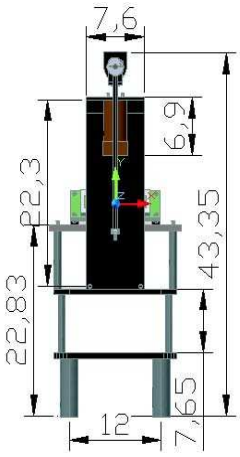

Figura 2. Vista frontal.

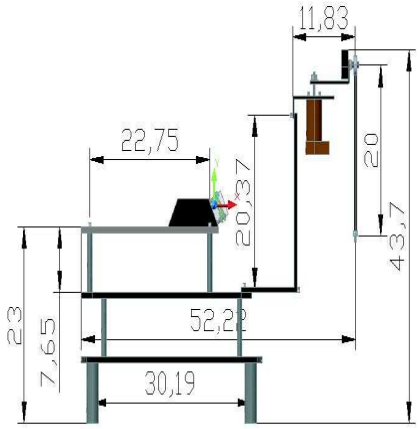

Figura 3. Vista lateral.

Una vez que se bosqueja el prototipo tridimensionalmente, es importante identificar las piezas clave que hacen referencia a un movimiento en el prototipo y transformarlos en objetos para más tarde migrarlo al simulador pieza por pieza, ya que cuando se programa la clase donde se descarga la imagen los cuerpos que generan un movimiento en el péndulo realizan un movimiento por bloques en el simulador.

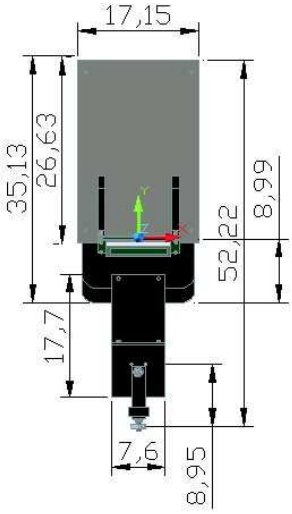

Figura 4. Vista superior.

Antes de emigrar el sólido del prototipo y una vez 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).

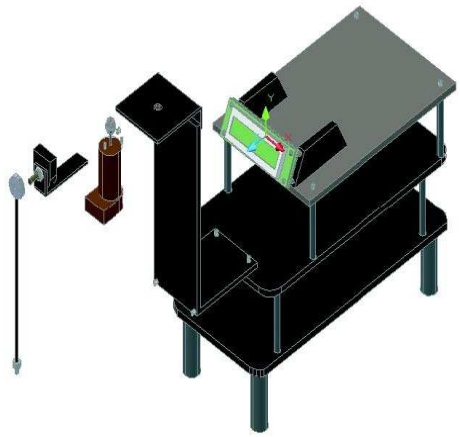

Figura5. Piezas clave en el movimiento.

Por lo que las piezas individuales son migradas a la plataforma 3DStudioMax, programa de diseño desarrollado por Autodesk. Una vez que se encuentran en dicha plataforma, son llevadas hacia el origen de su sistema de referencia en el plano coordenado, y se procede a transformarlas en una estructura o fichero de datos que represente mediante una rejilla rectangular de pixeles o puntos de color al formato de imagen 3DS.

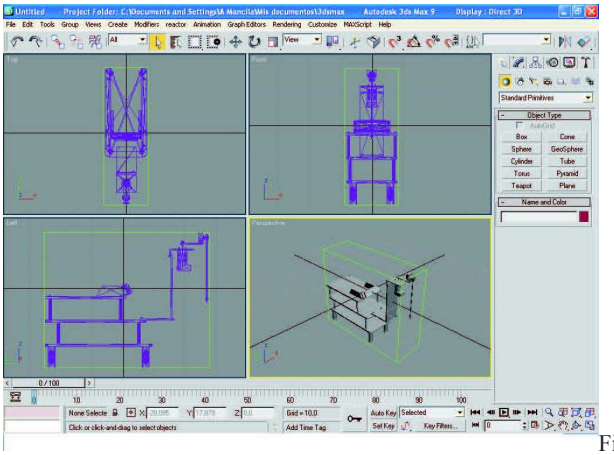

Figura 6. Ubicación de las referencias usando 3DStudio.

IV-A. OpenGL

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. OpenGL es un motor gráfico cuyas rutinas están integradas en la tarjeta gráfica, posee todas las características necesarias para la representación de escenas 3D modeladas con polígonos, desde el pintado más básico de triángulos, hasta el mapeado de texturas, transparencias, efectos de iluminación, etcétera. OpenGL fue desarrollada por Silicon Graphics Inc. en 1992. Su nombre viene del inglés Open Graphics Library, cuya traducción es librería de gráficos libre. 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 [11]. OpenGL es un conjunto de especificación estándar que definen una Interfaz de Programación de Aplicaciones (Application Programming Interface) multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. 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*.

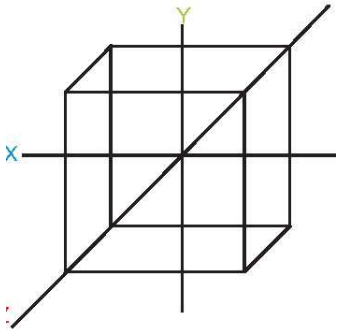

Figura7. Definición del espacio tridimencional en OpenGL.

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 7. Es importante ubicar el sólido, una vez hecho esto, se verifica con 3DStudio© 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 8.

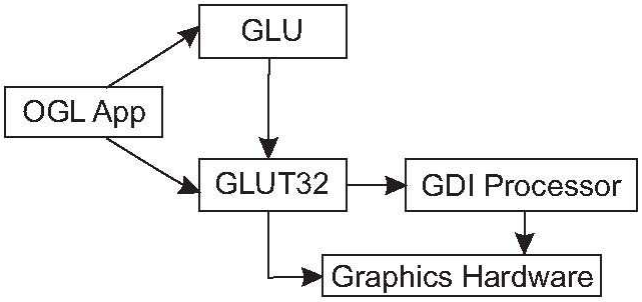

Figura 8. Procedimiento para cargar librerías OpenGL.

Las librerías que se agregan al proyecto son: la GLU que contiene funciones gráficas de más alto nivel, que permiten realizar operaciones complejas; y la GLUT que es un paquete auxiliar para construir aplicaciones de ventanas, además de incluir algunas primitivas geométricas auxiliares.

A continuación se enlistan algunas operaciones importantes:

1. Inicializar el modo de visualización.
2. Inicializar el tamaño de la ventana.
3. Inicializar la posición de la ventana.
4. Crear la ventana.
5. Definir el color de fondo.
6. Activar el test del buffer de profundidad.
7. Definir la función para redibujar la ventana.
8. Para generar nuevas ventanas en Open GL se utiliza la instrucción *glviewport(x, y, cx, cy)* donde *(x, y)* son la posición de la nueva ventana y *(cx, cy)* son el tamaño de esta.
9. Finalmente, tenemos que hacer que el bucle de eventos se ponga en marcha.

En OpenGL para manipular imágenes en tres dimensiones se necesitan definir las siguientes operaciones matriciales:

- **Rotación**: se define mediante la primitiva *glRotatef(tetha, x, y, z)*. Esta función multiplica la matriz actual por una matriz de rotación de θ grados respecto al eje *(x, y, z)*.
- **Traslación**: Se define mediante la primitiva *glTranslatef(x, y, z)*. Aplica una traslación en *(x, y, z)* sobre la matriz actual.
- **Escalado**: Se define mediante la primitiva *glScalef(sx, sy, sz)* Escalado de cada uno de los ejes.

La función *glPushMatrix* nos permite guardar la matriz actual en la pila, mientras que la función *glPopMatrix* toma la matriz que se encuentre en el top de la pila y la asigna a la matriz actual. Todas estas funciones nos sirven para el tratamiento da la imagen descargada, esta descarga se realiza en una clase donde contendrá variables de tipo puntero las cuales reservaran memoria para que estas sean tratadas. Esta clase debe ser declarada en la cabecera del proyecto principal, las cuales entregaran dos tipos: *.cpp* y *.h*; en los archivos *.cpp* estará todo lo referido a la descarga de imágenes, y en los archivos .h estarán todas las variables que se utilizan para la descarga del archivo 3DS.

IV-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. Los pasos para programar un simulador se describen en el diagrama de flujo de la figura 9. 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 [3,6].

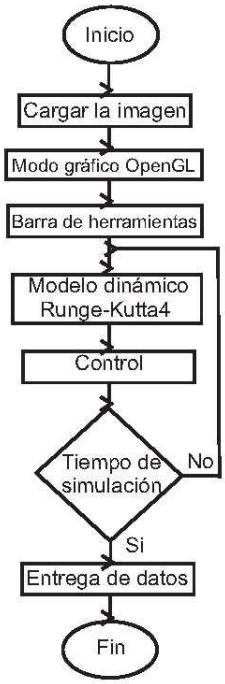

Figura 9. Diagrama de flujo general.

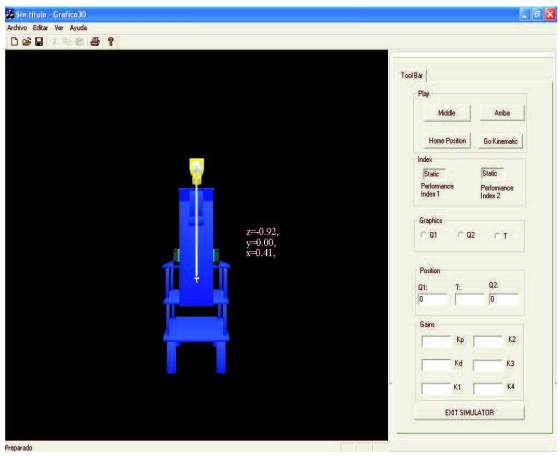

Figura10. Simulador.

Para el proyecto se utiliza el MFC ClassWizzard el cual es un asistente que despliega los recursos para poder diseñar el proyecto; generando una clase para la manipulación de los gráficos (clase vista o View); en esta clase incluimos las cabeceras necesarias para el procesado gráfico, y los mensajes OnCreate, OnDraw, OnPaint, OnInitialUpdate, OnSize, OnEraseBkgnd, OnLButtonDown, OnLButtonUp, OnMouseMove [6].

IV-C. Cargar el Modelo Dinámico

Para cargar el modelo dinámico en Visual C++ esnecesario crear una nueva clase, la cual se denomina modelo dinámico, en esta clase utilizamos el algoritmo de Runge Kutta de Cuarto Orden, el algoritmo se aplica a la dinámica del péndulo Furuta. La figura 10 muestra la pantalla del simulador.

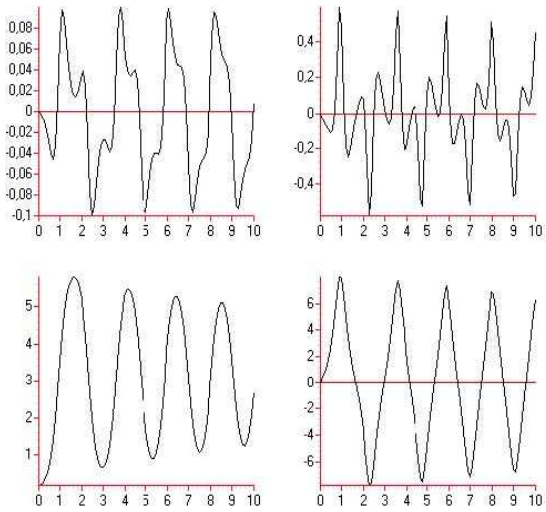

Figura 11. Resultados del simulador

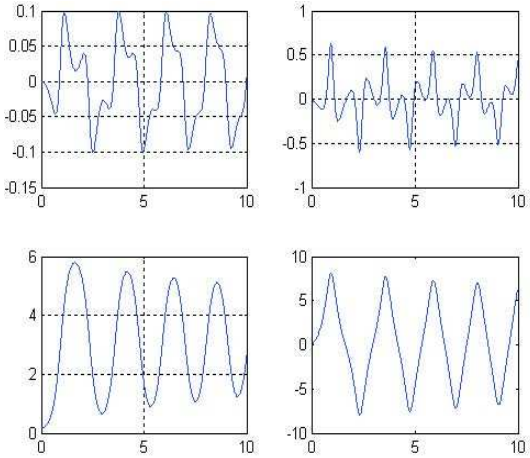

Figura 12. Simulación del péndulo Furuta en Matlab©

V. RESULTADOS

Los resultados obtenidos por el simulador se ilustran en la figura 11. En la parte superior de la figura se observa la posición y velocidad del brazo respectivamente, en la parte inferior se tiene la posición y velocidad del péndulo. Además, simulado el péndulo Furuta con un programa comercial como lo es Matlab©, se obtuvieron los resultados ilustrados en las figura 12.

VI. CONCLUSIONES

Un simulador es una herramienta computacional que utiliza un conjunto de elementos para describir el comportamiento de un sistema, estos componentes fueron: el modelo dinámico, el cual describe desde un punto de vista matemático el comportamiento del sistema a un estímulo; la identificación paramétrica ilustra los parámetros físicos del sistema; la estructura de control, la cual mediante la comparación entre la posición deseada $\tilde{q}$ y la posición real q originan que el error de posición $\tilde{q}$, que se busca que sea cero conforme el tiempo evoluciona haciendo que el efector final del robot manipulador alcance el objetivo; la representación tridimensional que brinda la perspectiva de profundidad y hace que el ambiente gráfico sea fácil de interpretar; y los algoritmos de programación que relacionan estos elementos para formar un todo.
El objetivo principal de este trabajo fue modelar y evaluar un péndulo invertido en configuración Furuta haciendo uso de una plataforma gráfica de simulación; los parámetros que definen la física del sistema fueron obtenidos del manual del fabricante; para la etapa de control se usaron estructuras bien conocidas como lo son el control PD y el control tanh, las cuales cuentan con la debida comprobación de estabilidad; la representación tridimensional del sistema se desarrollo usando diseño asistido por computadora AutoCAD©; y los algoritmos de programación se generaron en la plataforma Visual C©. El desempeño del simulador se evaluó mediante la comparación con la plataforma comercial Matlab©. Por lo cual se concluye que el trabajo realizado fue satisfactorio.

VII. REFERENCIAS

[1] S. Mori, H. Nishihara and K. Furuta, Control of unstable mechanical system control of pendulum", Int. J. Control, vol. 23 pp 673-692, 1976.

[2] K. Furuta, T. Ochiai, and N. Ono, Attitude control of a triple inverted pendulum". Int. J. Control, vol. 39 pp 1351-1365, 1984.

[3] Rosheim M. E., Robot Evolution, John Wiley ans Sons Inc., New York, 1994.

[4] J. Aracil y Gordillo F., El péndulo invertido: un desarrollo para el control no lineal, Escuela Superior de Ingenieros, Universidad de Sevilla, 2005.

[5] Moreno María C., Hernández Laura C., Camacho Johanna P., Control de un péndulo invertido empleando lógica Difusa y Algoritmos Genéticos, Ingeniería y Universidad, Bogotá, Colombia.

[6] A. Loría, On Output Feedback Control of Euler-Lagrange, (France: Université de Technologie de Compiégne, PhD Thesis, 1996).