La Mecatrónica en México, Vol. 3, No. 3, páginas 101 - 108, Septiembre 2014
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN en trámite, 2014 Derechos de autor y derechos conexos, Asociación Mexicana de Mecatrónica A.C

Desarrollo de un Simulador de un Robot Cartesiano para dispositivos móviles utilizado Solidworks y Open GL ES 2.0

Meléndez Campos Javier[1], Jiménez López Eusebio[2], Ruiz Domínguez Alejandra[1], Urbalejo Contreras Arturo[3], Reyes Ávila Luis[4], Luna Acosta Noé[3] y Vázquez Cuevas Ignacio[5]

[1] Universidad La Salle Noroeste, jmelendezcampos@hotmail.com, alejandra_ruiz92@hotmail.com
[2] ULSA Noroeste - CINNTRA de la Universidad Tecnológica del Sur de Sonora-IIMM, ejimenezl@msn.com
[3] CIAAM de la Universidad Tecnológica del Sur de Sonora, urbalejoa@hotmail.com, nluna@uts.edu.mx
[4] Instituto Mexicano del Transporte-IIMM, lreyesa@imt.mx
[5] Universidad Tecnológica de Nogales-ITN, ing.ignacio.javier@gmail.com

Resumen

En este artículo se presenta el desarrollo de un simulador cinemático de un robot cartesiano de dos grados de libertad (2GDL) usando el software Solidworks 2014 y gráficos en Open GL ES 2.0. El modelo cinemático del robot fue construido haciendo un análisis de cuerpo rígido y un análisis de trayectoria rectilínea conformada por lugares geométricos rectilíneos y perfiles de trayectoria trapezoidales suavizados con funciones polinomiales de grado 5. El simulador fue diseñado para ser visualizado y operado en dispositivos móviles (Tablets) bajo el sistema operativo iOS. El robot fue diseñado y construido virtualmente en Solidworks y posteriormente los archivos fueron transferidos a OpenGL ES 2.0. Los modelos cinemáticos del robot y la trayectoria fueron importados del software de cálculo simbólico MatLAB. El simulador puede ser fácilmente operado desde una Tablet.

Palabras clave: Robot cartesiano, SolidWorks, Open GL, Dispositivos Móviles.

1. Introducción

El diseño de cualquier diseño mecatrónico hoy en día implica necesariamente la incorporación de la simulación computacional en cualquiera de los sistemas que componen el desarrollo de un producto, un proceso o un sistema. En forma general, el término simular es sinónimo de imitar el comportamiento de un sistema con algún propósito específico [1]. Para realizar las simulaciones numéricas y gráficas de robots se han utilizado diferentes programas de uso específico, entre estos destacan los siguientes: a) aquellos que se basan en el diseño mecánico, por ejemplo Autocad®, SolidWorks®, Mechanical Desktop®, etc., y b) aquellos que se basan en el modelo dinámico, por ejemplo Matlab®-Simulink®, Simnon, etc [2]. El desarrollo de una simulación depende de las necesidades del cliente, de la selección del software y de las plataformas de programación. En este proceso no se aplica una metodología específica, más bien el integrador debe explorar las distintas posibilidades que se tienen en cuanto a tecnologías disponibles y recursos computacionales.

Existen diversos trabajos que reportan desarrollos de simuladores robóticos, por ejemplo en [3] se presenta el desarrollo de un robot hexápodo usando CATIA y un software para el cálculo y el modelado dinámico. En [4] se utiliza el software Solidworks para el análisis y modelado cinemático y cinético de un robot de rescate. En este artículo se presenta el desarrollo de un simulador de un robot cartesiano usando Solidworks 2014, Open GL ES 2.0, MatLAB y bibliotecas de Apple. El simulador se opera desde un dispositivo móvil. Se describen el modelo cinemático y de trayectoria relacionados con el robot y un método de interpolación polinomial.

2. Descripción del problema

Se busca desarrollar un robot cartesiano para que forme parte de un proceso productivo de una celda de manufactura didáctica. Para la concepción y fabricación del robot, se requiere de la formación de un grupo multidisciplinario conformados por alumnos y profesores de la Universidad la Salle Noroeste y la

Universidad Tecnológica del Sur de Sonora (UTS) bajo el convenio de la RED ALFA. Los requerimientos técnicos para el desarrollo del robot fueron los siguientes:

1) Diseño en CAD de las partes y ensamble del robot.
2) Desarrollo de un simulador del robot integrando trayectorias rectilíneas con perfiles de trayectoria trapezoidales.
3) Uso de técnicas de manufactura aditiva para la fabricación del robot.
4) Debe incluir motores paso a paso.
5) El control y manejo del robot debe realizarse por medio de un dispositivo móvil (Tablet o celular).
6) El robot formará parte de una celda de manufactura para propósitos didácticos del proyecto PROMEP de la UTS y para hacer pruebas de sincronización en celdas de manufactura de las universidades que integran la red ALFA.

3. Propuesta de solución para el desarrollo del simulador

El desarrollo de cualquier sistema mecatrónico requiere del diseño o la utilización de un simulador computacional. Para el caso del robot solicitado por la empresa el diseño del simulador debe ser de tal forma que se pueda operar desde un dispositivo móvil, puesto que el control del robot físico se hará desde dicho dispositivo. Después de haber discutido algunas ideas se propuso el siguiente desarrollo del simulador:

1) Dibujar el robot y sus componentes en un paquete CAD.
2) Construir el modelo cinemático del robot y la trayectoria en un paquete de cálculo simbólico.
3) Usar métodos gráficos y software para dispositivos móviles y migrar la información del dibujo CAD y los modelos del robot.
4) Probar el simulador en el dispositivo móvil elegido.

4. Propuesta del software

Una vez elegido el camino y las ideas principales para el desarrollo del simulador se tomó la decisión de utilizar los siguientes paquetes y plataformas computacionales:

1) Para el desarrollo del robot en CAD se usó Solidworks 2014.
2) Para modelar y obtener las funciones y/o datos numéricos del robot y de la trayectoria se usó MatLAB.
3) Sistema operativo iOS para dispositivos móviles y X-CODE para la programación.
4) Lenguaje de programación OBJETIVE-C.
5) Para la simulación gráfica se usó OPEN GL ES 2.0.

La Figura 1 muestra el software utilizado.

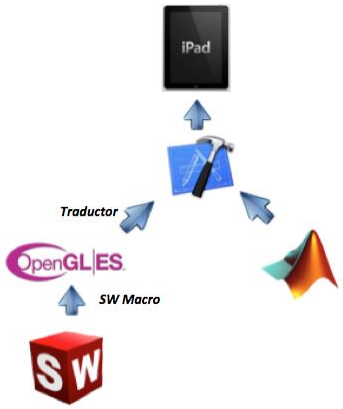

Fig. 1. Software utilizado para el desarrollo del robot cartesiano.

La Mecatrónica en México, Vol. 3, No. 3, páginas 101 - 108, Septiembre 2014
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN en trámite, 2014 Derechos de autor y derechos conexos, Asociación Mexicana de Mecatrónica A.C

5. Desarrollo del simulador

En esta sección se presenta el desarrollo del simulador motivo de estudio en este artículo.

5.1 Modelado cinemático del robot.

El modelado cinemático del robot se dividió en dos partes: 1) Modelo de cuerpo rígido para el robot y 2) Modelo de trayectoria. La Figura 2 muestra la configuración del robot [5].

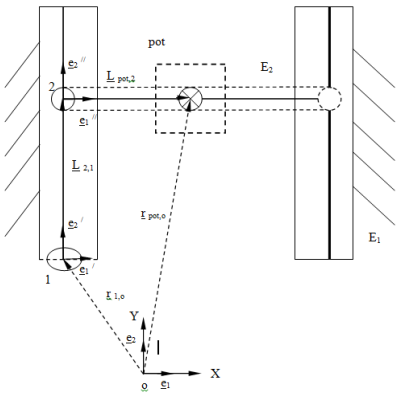

Fig. 2. Configuración del robot.

De acuerdo con la Figura 2, el robot está constituido por dos eslabones (E_1 y E_2). El objetivo del modelado cinemático es construir las ecuaciones de posición, desplazamiento, velocidad y aceleración relacionadas con el punto del órgano termina (pot) con referencia al origen del sistema cartesiano inercial (X,Y).

El modelo de posición es:

$$\underline{r_{pot,0}} = \underline{r_{1,0}} + l_{2,1} \bullet \underline{e_2^{'}} + l_{pot,2} \bullet \underline{e_1^{''}} \tag{1}$$

Aquí, $l_{2,1}$ y $l_{pot,2}$ son las longitudes de los eslabones y los sistemas móviles $\underline{e}_2^{/}$ y $\underline{e}_1^{//}$ se desplazan con los eslabones.

La expresión (1) se puede escribir en términos de coordenadas de la manera siguiente:

$$\underline{r_{pot,0}} = \underline{r_{1,0}} + (y_2 - y_1) \bullet \underline{e_2} + (x_{pot} - x_2) \bullet \underline{e_1} \tag{2}$$

Aquí, $\underline{e}_1$ y $\underline{e}_2$ son las bases canónicas localizadas en el origen del sistema de referencia mostrado en la Figura 1. Por otro lado, el modelo de desplazamiento es el siguiente:

$$\underline{r_{pot,0}^{'}}(t) = \underline{r_{1,0}} + \underline{L_{2,1}^{'}}(t) + \underline{L_{pot,2}^{'}}(t) \tag{3}$$

Los modelos de velocidad y aceleración son:

$$1) \quad \underline{\dot{r}'_{pot,0}}(t) = \underline{\dot{L}'_{2,1}}(t) + \underline{\dot{L}'_{pot,2}}(t) \tag{4}$$

$$2) \quad \underline{\ddot{r}'_{pot,0}}(t) = \underline{\ddot{L}'_{2,1}}(t) + \underline{\ddot{L}'_{pot,2}}(t) \tag{5}$$

O, en forma equivalente:

La Mecatrónica en México, Vol. 3, No. 3, páginas 101 - 108, Septiembre 2014
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN en trámite, 2014 Derechos de autor y derechos conexos, Asociación Mexicana de Mecatrónica A.C

$$1)\quad \dot{\underline{r}}_{pot,0}(t) = \dot{l}_{2,1}(t) \bullet \underline{e}_2 + \dot{l}_{pot,2}(t) \bullet \underline{e}_1$$

$$2)\quad \ddot{\underline{r}}_{pot,0}(t) = \ddot{l}_{2,1}(t) \bullet \underline{e}_2 + \ddot{l}_{pot,2}(t) \bullet \underline{e}_1$$

Por otro lado, el modelo de trayectoria del robot estudiado se compone de lugares geométricos rectilíneos y de perfiles de trayectoria trapezoidales. La Figura 3 muestra en líneas punteadas los perfiles trapezoidales de la trayectoria y en línea continua los perfiles suavizados con interpolación polinomial [6].

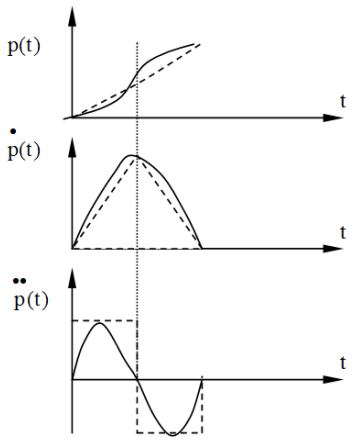

Fig. 3. Perfiles de trayectoria trapezoidales.

Para el caso del simulador motivo de estudio fue usada interpolación polinomial grado 5 para suavizar los perfiles. El modelo de trayectoria final del movimiento del robot del punto **a** al punto **b** en el espacio-tiempo (en este caso en el plano-tiempo) es el siguiente:

$$1)\ \underline{R}_{b,o}(t) = \underline{R}_{a,o} + p(t) \bullet \frac{1}{d_{b,a}} \begin{bmatrix} x_b - x_a \\ y_b - y_a \\ z_b - z_a \end{bmatrix} \tag{6}$$

$$2)\ \dot{\underline{R}}_{b,o}(t) = \dot{p}(t) \bullet \frac{1}{d_{b,a}} \begin{bmatrix} x_b - x_a \\ y_b - y_a \\ z_b - z_a \end{bmatrix}$$

$$3)\ \ddot{\underline{R}}_{b,o}(t) = \ddot{p}(t) \bullet \frac{1}{d_{b,a}} \begin{bmatrix} x_b - x_a \\ y_b - y_a \\ z_b - z_a \end{bmatrix}$$

Nótese que el vector $\underline{R}_{b,o}(t)$ localiza el punto **b** (punto extremo del lugar geométrico rectilíneo) desde un origen **o** en el espacio tiempo, $\underline{R}_{a,o}$ localiza el punto inicial del lugar geométrico desde el origen **o**, $d_{b,a}$ es la distancia entre dos puntos y p(t) es una función polinómica de grado 5, esto es:

$$p(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \alpha_3 t^3 + \alpha_4 t^4 + \alpha_5 t^5 \tag{7}$$

La Mecatrónica en México, Vol. 3, No. 3, páginas 101 - 108, Septiembre 2014
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN en trámite, 2014 Derechos de autor y derechos conexos, Asociación Mexicana de Mecatrónica A.C

El paso final para completar el modelado cinemático del robot es ensamblar las ecuaciones (3) y (5) con las ecuaciones (6).

5.2 Programación en MatLAB.

Con la finalidad de determinar la posición, velocidad y aceleración del efector final (pot) del robot motivo de estudio, se desarrolló un algoritmo el cual tomaba como argumentos: la posición inicial y final del robot, así como la velocidad a la cual el usuario quiere que se mueva el dispositivo. El polinomio de grado cinco consiste en la multiplicación de matrices en las cuales intervienen las ecuaciones de posición, velocidad y aceleración del objeto. Para facilitar la tarea al algoritmo desarrollado en Objetive-C, se realizaron las multiplicaciones de matrices en el software MATLAB, el cual dio como resultado una ecuación general del movimiento la cual se transfirió a código C para integrarse con el algoritmo.

De esta manera mediante controles en la interfaz, el usuario puede indicar la posición en el espacio a la que el robot se situará, así como la velocidad en la que se desplazará de un punto a otro. La Figura 4 muestra los perfiles de trayectoria suavizados en MatLAB.

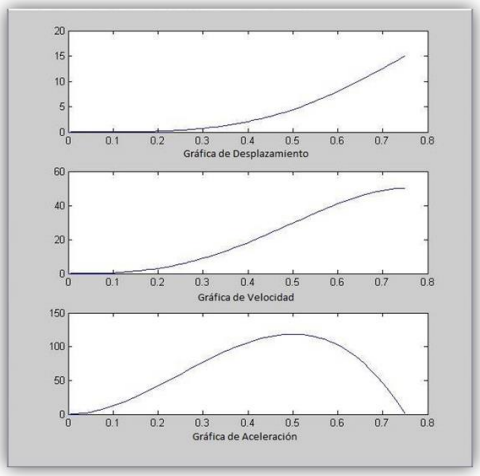

Fig. 4. Perfiles de trayectoria suavizados en MatLAB.

A) Dibujo en sólidos del robot.

El prototipo del robot fue dibujado en la plataforma Solidworks 2014. La figura siguiente muestra la configuración del robot:

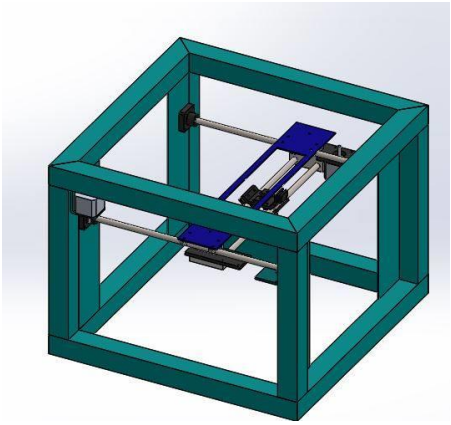

Fig. 5. Modelado en sólidos del robot cartesiano.

La Mecatrónica en México, Vol. 3, No. 3, páginas 101 - 108, Septiembre 2014
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN en trámite, 2014 Derechos de autor y derechos conexos, Asociación Mexicana de Mecatrónica A.C

El modelado del robot en Solidworks es clave para el desarrollo y la animación del simulador, ya que la información geométrica de los componentes del robot serán migrados a Open GL ES 2.0.

5.3

5.4 Desarrollo gráfico.

Para mejorar la interacción con el usuario se realizó una simulación gráfica en OpenGL la cual tendría que ser lo más representativamente posible al diseño real del robot. Apple, mediante el paquete de XCode, provee una serie de bibliotecas y ejemplos que encapsulan y facilitan el uso de muchas cosas de interés para los desarrolladores, de manera que estos puedan adaptarlos a su plataforma de manera más fácil. De esta forma, se pueden desarrollar aplicaciones más complejas reduciendo de manera significativa el tiempo de programación. Dentro de estas bibliotecas se encuentra una llamada GLKit, la cual encapsula, entre otras cosas, todo lo correspondiente a OpenGL ES 2.0. En dicha biblioteca se encuentran **Objetos** los cuales facilitan la generación y configuración de ventanas (en este entorno llamadas **Vistas**) para renderizado en 2D y 3D. Como se mencionó anteriormente, OpenGL ES 2.0 grafica figuras a partir de primitivas básicas como lo son el triángulo y la línea. Es por eso que el robot motivo de estudio se dibujó usando estas primitivas de manera recurrente, siendo el dibujado vértice por vértice un poco tedioso. Se buscó implementar una forma de integrar los gráficos desde otra plataforma.

Como recurso inicial se contaba con un diseño final del robot construido en Solidworks 2014, (ver Figura 5). Este software es conocido por ser altamente utilizado en la industria, siendo uno de los líderes a nivel mundial en lo que se refiere a software de diseño asistido por computadora. Sin embargo, Solidworks y los archivos generados por este, no permiten la transferencia directa de gráficos a OpenGL ES 2.0. Se tuvo que diseñar una metodología de migración de gráficos de las piezas y ensambles de Solidworks a información ordenada de vértices, la cual puede ser adoptada e interpretada por OpenGL ES 2.0, para posteriormente ser graficada triángulo por triángulo en pantalla.

Para realizar la migración se utilizó como intermediario el formato OBJ usado en algunos programas de Diseño 3D que a su vez son implementados con regularidad en la industria de los videojuegos. Este formato guarda de manera ordenada y de manera entendible información básica de las figuras, como los vértices, caras, colores y materiales. Solidworks tampoco es capaz de exportar sus figuras y ensambles en este formato por sí solo, sin embargo en internet se pueden encontrar múltiples programas llamados “Solidworks Macros” que al sumarse a Solidworks, entre otras tareas son capaces de realizar exportaciones en este formato.

Como ya se mencionó con anterioridad, el formato OBJ sólo se utilizó como intermediario entre Solidworks y OpenGL ES 2.0, es por eso que se tuvo que buscar y modificar un programa que tomara la información del archivo OBJ y lo estructurara en archivos de formato C (.h y .c) de tal manera que fuera posible acceder a esta información desde OBJECTIVE-C. Una vez generados estos archivos, son importados a XCode e instanciados desde el Objeto de la clase correspondiente a GLKit. Si se abren los archivos generados por el programa se encontrarán con una serie de arreglos simples de C en el que se enumeran de manera ordenada todos los vértices, incluso se le da un formato para que sea más fácil visualizar los grupos de tres vértices correspondientes a cada uno de los triángulos. De esta manera la figura del robot será mostrada en pantalla justo como se mostraba en Solidworks.

Cabe mencionar que está en manos del usuario aplicar técnicas de programación de OpenGL para realizar las transformaciones que él desee a la figura, es decir, escalar, desplazar o en su defecto rotar la figura.

Para el caso del robot motivo de estudio, se usó una rotación general para que el usuario fuera capaz de manipular la perspectiva en la que miraba la figura del robot. Para esto se usó un método que se volvió muy popular con el aumento de las pantallas táctiles en la vida cotidiana de las personas. Consiste en encerrar la figura en una esfera imaginaria, el usuario se encarga de tocar la pantalla y arrastrar la esfera, de manera que la figura dentro de esta también se rota. Se puede decir que es muy similar a la manera en que se interactúa con un globo terráqueo, mientras este gira. Tal vez sea muy común para todos realizar este tipo de gestos en pantalla ya que se utilizan en la mayoría de las aplicaciones para teléfonos inteligentes y tablets que se encuentran en el mercado en las cuales se utilizan gráficos.

La Mecatrónica en México, Vol. 3, No. 3, páginas 101 - 108, Septiembre 2014
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN en trámite, 2014 Derechos de autor y derechos conexos, Asociación Mexicana de Mecatrónica A.C

Además de las rotaciones se utilizaron desplazamientos, regularmente llamados traslaciones. Es de vital importancia realizar un seccionado de las partes móviles del ensamble de Solidworks, así como realizar la exportación separada de cada una de ellas, de tal manera que las partes del robot puedan ser manipuladas de manera independiente al momento de las transformaciones de OpenGL. Es en esta parte donde se determinó una posición (0,0) para el efector final del robot, y desde ahí se realizan los traslados correspondientes a cada eje de la máquina. La distancia de traslado es indicada por el algoritmo en el cual se realizó el polinomio de grado 5. En resumen, se pretende que se enlacen las variables de forma que cada vez que haya un cambio y se decida mover la posición del robot, se actualiza la variable de traslado gráfico para que el usuario se percate del comportamiento del robot de manera inmediata. La Figura 6 muestra el simulador final del robot cartesiano.

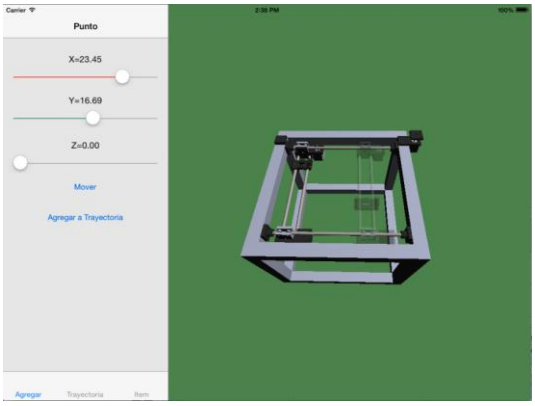

Fig. 6. Simulador final del robot cartesiano.

6. Conclusiones

En este artículo se ha descrito el desarrollo de un simulador cinemático de un robot cartesiano, el cual es operado desde un dispositivo móvil. Las principales conclusiones se resumen en los puntos siguientes:

-Los diseños mecatrónicos requieren necesariamente de la simulación computacional. Para el caso de robot motivo de estudio, la simulación cinemática fue hecha integrando diferentes softwares y plataformas de programación lo que resultó un sistema fácil de operar en un dispositivo móvil.
-La simulación cinemática requiere del desarrollo de modelos matemáticos operacionales y de paquetes de cálculo formal. Para el caso del robot estudiado en este artículo, el modelado se realizó en forma separada (robot y trayectoria) y, posteriormente, se integraron los modelos. Este proceso facilitó la programación en MatLAB.
-El entorno de desarrollo XCode y las bibliotecas que incluye, hicieron más fácil el diseño e implementación del simulador para el sistema operativo iOS, incluyendo cálculos avanzados y gráficos de alta calidad, teniendo como producto final una aplicación amigable y fácil de usar por el usuario.
-La metodología usada para la migración de gráficos redujo el tiempo de programación y dio como resultado un modelo detallado del diseño original hecho en Solidworks.

Agradecimientos

Los autores de este trabajo agradecen a las universidades y empresas que integran la RED Interinstitucional ALFA, a la RED de Manufactura y Mantenimiento, a la empresa Spin – OFF Innovación en Ingeniería de Manufactura y Mantenimiento S. de R.L. MI, al Cuerpo Académico CIAAM de la UTS, al Programa del Mejoramiento del Profesorado (PROMEP) bajo el proyecto Desarrollo de un Prototipo de Robot Industrial para Aplicaciones en Celdas de Producción Automatizadas Clave UTSSON-CA-4, por el apoyo brindado para la realización de este trabajo.

La Mecatrónica en México, Vol. 3, No. 3, páginas 101 - 108, Septiembre 2014
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN en trámite, 2014 Derechos de autor y derechos conexos, Asociación Mexicana de Mecatrónica A.C

Referencias

[1] Jiménez E., Navarro J., Reyes L, Luna N., Urbalejo A., Islas M., Castro J. Simulación de una celda de manufactura usando Quaterniones y Mathematica. *11º Congreso Nacional de Mecatrónica*. Villahermosa, Tabasco. 2012.

[2] Maldonado H., Silva R., Ramos E., Hernández V., Rivera J. Modelado y simulación de un robot rígido de dos grados de libertad. *Lat. Am. J. Phys. Educ.* Vol.5, No. 1. 2011.

[3] Mahapatra A., Shekhar S., Kumar D. Modeling and Simulation of Wave Gait of a Hexapod Walking Robot: A CAD/CAE Approach. *International Journal of Robotics and Automation (IJRA).* Vol. 2, No. 3, pp. 104~111. 2013.

[4] Hassanzadeh E., Shahmohammadi M., Khamseh N. Kinematic and Kinetic Study of Rescue Robot by SolidWorks Software . *Research Journal of Applied Sciences, Engineering and Technology* 5(21): 5070-5076. 2013.

[5] Sobarso J. Modelo y simulación de las trayectorias de un robot cartesiano usando la integracion LabVIEW-SolidWorks para aplicaciones industriales. Tesis de Maestría en Ciencias de la Ingeniería Mecatrónica. Instituto Tecnológico Superior de Cajeme. 2012.

[6] Jorge A. "Fundamentals of Robotic Mechanical System. Theory, Methods and Algorithms". Springer-Verlag. New York. 1997.