HERRAMIENTA COMPUTACIONAL PARA LA ENSEÑANZA DE LA ROBÓTICA

Braulio Cruz Jiménez
Universidad Autónoma de Yucatán
braulio.cruz@correo.uady.mx

Jannette Contreras Rivero
Universidad Autónoma de Yucatán
jannette.contreras@correo.uady.mx

Ricardo Peón Escalante
Universidad Autónoma de Yucatán
rpeon@correo.uady.mx

Luis Ricalde Castellanos
Universidad Autónoma de Yucatán
lricalde@correo.uady.mx

Resumen

En este artículo se presenta el diseño e implementación de una herramienta computacional para el apoyo de la enseñanza de la robótica, la cual permite simular y programar robots antropomórficos de tres grados de libertad de manera virtual; esto se logra a través de la definición de las posiciones de operación del robot por el usuario y escribiendo el código que utilice dichas posiciones. Posee también conexión inalámbrica por medio de Bluetooth, lo que permite la interconexión con dispositivos físicos robóticos de tres grados de libertad. Esta herramienta computacional permite la utilización de una interface visual, facilitando la simulación y programación de robots físicos mediante el empleo de los modelos cinemáticos directo e inverso del robot. Adicionalmente la interface visual permite al usuario desligarse de la complejidad de los lenguajes de programación, debido a que la programación se realiza en modo gráfico, lo que permite focalizarse en el

diseño de la aplicación robótica en lugar de privilegiar la sintaxis del lenguaje de programación del robot.

Palabra(s) Clave(s): Cinemática directa, cinemática inversa, interfaz virtual, robótica educativa.

1. Introducción

Un simulador robótico es un software para recrear un entorno robotizado sin depender físicamente en el robot real, reduciendo así los costos de desarrollo y mejora de la eficiencia mediante la simplificación del trabajo de desarrollo. Utilizando plataformas de programación modernas, los comportamientos de un robot se pueden simular con gran precisión en comparación con el robot real [1].

La mayoría de los simuladores robóticos se pueden aplicar a los robots actuales sin necesidad de modificaciones importantes. Por otra parte, los sistemas robóticos son cada vez más complejos hoy en día, y los requisitos de rendimiento son más altos, por lo que los simuladores robóticos son de vital importancia en la investigación y el desarrollo de los sistemas robóticos. Por otra parte, los simuladores aplicados en la investigación robótica son altamente eficientes en la verificación de la seguridad, la fiabilidad y la robustez de los algoritmos en desarrollo [2]. En el área de la educación, los investigadores también han demostrado que los estudiantes pueden aprender lenguajes de programación de manera más rápida por medio de un simulador robótico [2]. Existe una gran cantidad de simuladores de robótica hoy en día y se pueden dividir en dos categorías principales: comercial y de código abierto

En [3] se presenta un ejemplo de un simulador comercial llamado Webots, que se utiliza para la creación de prototipos rápidos de robots móviles. Los modelos simulados tienen propiedades personalizadas tales como la masa y la fricción proporcionadas por librerías externas utilizadas para la simulación de dinámica del cuerpo rígido.

Virtual Robot Experimentation Platform (V-REP) es otro tipo de software comercial, consistente en un simulador 3D que se utiliza para simular cualquier robot, especialmente los sistemas de control distribuido, y es una aplicación versátil

compatible con Linux, MacOS y Windows. V -REP también se utiliza para el desarrollo rápido de algoritmos, prototipado rápido y monitoreo remoto [4].

Easy-Rob es un software de simulación para plataformas robóticas en plantas de manufactura que permite al usuario visualizar procesos de operación como manipulación y recubrimiento. Easy-Rob también tiene funcionalidad de generar películas en formato AVI y provee bibliotecas para la importación y exportación de archivos de dibujo [5].

RobotStudio es un simulador que utiliza el lenguaje RAPID en Windows como plataforma de software para robots industriales desarrollados por ABB. Esta plataforma de software permite a los usuarios desarrollar entornos de simulación realistas basados en robots virtuales que son copias exactas del software utilizado en las líneas de producción [6].

Microsoft Robotics Developer Studio (MRDS) es un software en 3D basado en Windows desarrollado por Microsoft para apoyar una variedad de sensores y muchas otras plataformas. Este simulador es compatible con todas las versiones del sistema operativo Windows y también es capaz de simular robots de servicio [7].

Modular Open Robots Simulation Engine (MORSE) es un simulador genérico de código abierto aplicado principalmente en el ámbito académico para simular robots autónomos en ambientes interiores y exteriores. Los programas están escritos en su mayoría como simples scripts de Python [8].

Robot Operating System (ROS) es una plataforma de software distribuido de código abierto para robótica que permite a montar archivos ejecutables en tiempo de ejecución, lo que permite a los investigadores en el campo de la robótica reutilizar los códigos fuente de la comunidad ROS. ROS se basa en el sistema operativo Ubuntu Linux y soporta una variedad de idiomas a través de C ++, Python, Octave, y LISP por medio de un simple lenguaje de definición de interfaz [9]. ARS es otro ejemplo de simulador de código abierto genérico para robótica académica escrito en Python y se caracteriza por tener una documentación extensa [10]. ICUB es una plataforma de simulación cognitiva completa para un

robot humanoide que implementa la complejidad de la cinemática y provee una interface de control [11], [12].

En [13] se presenta el desarrollo de una aplicación móvil destinada al control de un brazo robot. La aplicación se elaboró mediante la herramienta de programación basada en bloques MIT App Inventor, la cual permite desarrollar aplicaciones para dispositivos móviles que funcionan bajo el sistema operativo Android. La interfaz gráfica de la aplicación es intuitiva y cuenta con tres modos de operación: manual, programación y automático. Tanto la aplicación móvil como el brazo robot tienen la finalidad de apoyar el aprendizaje de estudiantes que cursan la carrera de Ingeniería Mecatrónica.

Uno de los campos con buen desarrollo es la robótica educativa, la cual es un medio de aprendizaje para fortalecer áreas específicas del conocimiento y desarrollar competencias en el estudiante mediante la planeación, diseño, implementación, prueba y puesta en marcha de robots [14].

La robótica educativa es propicia para apoyar habilidades productivas, creativas, digitales y comunicativas; convirtiéndose en un motor para la innovación produciendo cambios en las personas, en las ideas y actitudes, modos de actuar y pensar de los estudiantes y educadores [14].

En este artículo se presenta el diseño e implementación de una interface virtual para la simulación y programación de robots de tres grados de libertad, buscando despertar el interés de los estudiantes y transformando las asignaturas tradicionales en más atractivas e integradoras mediante la creación de un entorno de aprendizaje que recrea los problemas del ambiente que los rodea.

2. Desarrollo

El objetivo de este proyecto es desarrollar una plataforma de simulación y programación de robots con una interface amigable y que permita al usuario aprender de manera natural conceptos sobre robótica. Cuando se ejecuta la aplicación aparece una pantalla donde se visualiza una imagen de un robot de cadena cinemática abierta de tres grados de libertad así mismo se observa la estructura del programa y la lista de posiciones en la figura 1.

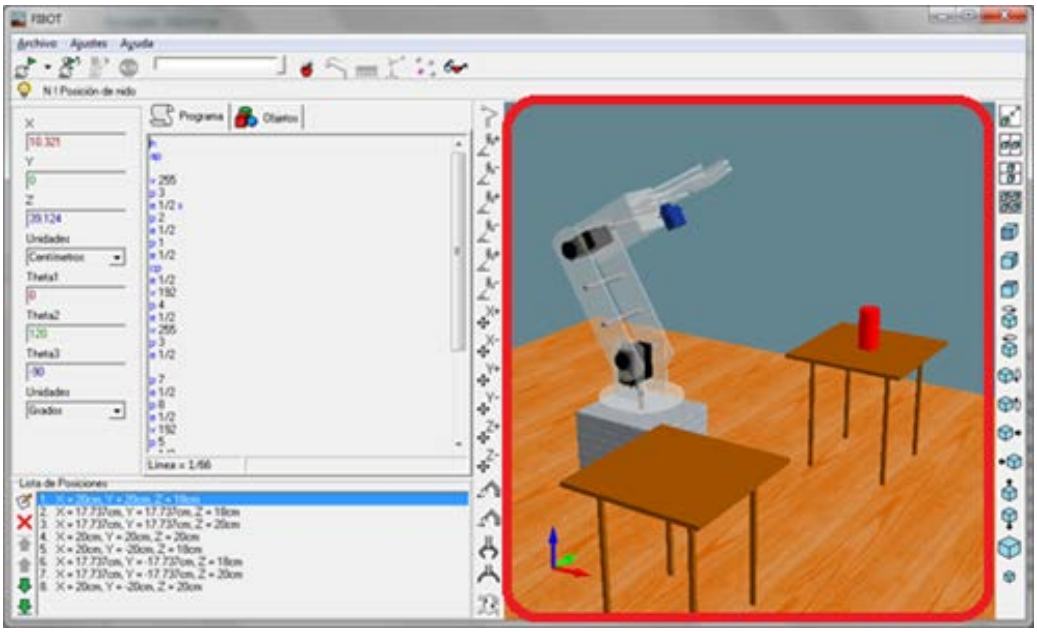

Figura 1 Interfaz principal del simulador.

La interface gráfica de usuario fue desarrollada en Borland C++ Builder 6, para tener acceso al panel del código seleccione la pestaña Programa, en este panel podrá redactar y editar el programa como en cualquier editor de texto plano. El programa está conformado por líneas de código, las cuales sólo puede contener un comando seguido de un comentario, el cual inicia con el carácter '!' y abarca hasta el final de la línea; es importante que la simulación no se encuentre en ejecución para poder editar el código. Se pueden agregar nuevos objetos seleccionando la pestaña Objetos, una vez realizado esto, en el panel de objetos se podrá elegir un objeto de la lista y su imagen aparecerá en el panel de vista previa, así como su orientación y propiedades de tamaño. Para ejecutar la simulación se presiona el botón de reproducir, en este modo se ejecutarán secuencialmente todos los comandos contenidos en el código del programa desde el primer comando hasta el último, posteriormente la ejecución del programa se detendrá automáticamente.

Cinemática del robot

La cinemática del robot estudia el movimiento del mismo con respecto a un sistema de referencia. Se interesa por la descripción analítica del movimiento

espacial del robot como una función del tiempo, y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares [16]. A continuación, se desarrollan los modelos cinemático directo e inverso de un robot de tres grados de libertad que se implementarán en la interface visual.

Modelo cinemático directo del robot

La configuración del robot que se va a utilizar en la herramienta computacional está compuesta por una cadena cinemática de tres eslabones rectos de longitudes r_1, r_2 y r_3, como se muestra en la figura 2 [17].

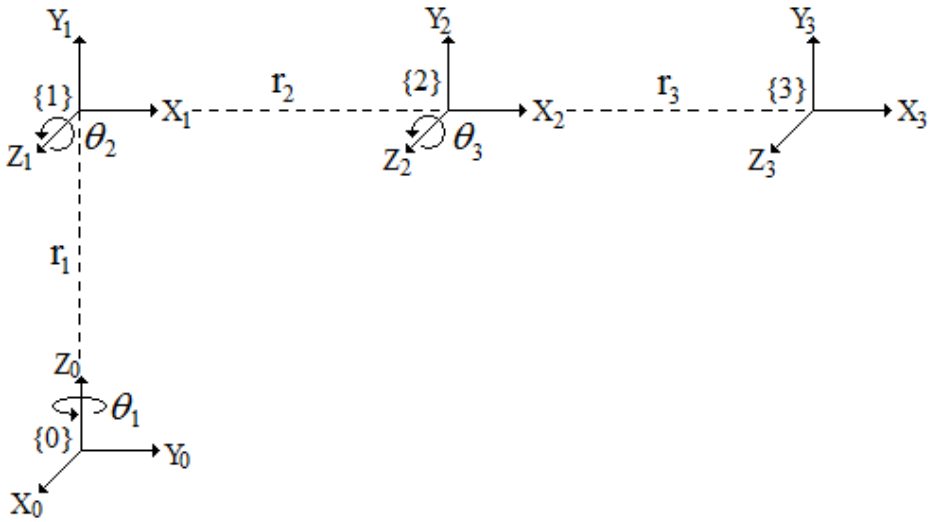

Figura 2 Cadena cinemática.

Los parámetros de Denavit-Hartenberg y rango de movimiento de cada unión se resumen en la tabla 1.

Tabla 1 Parámetros de Denavit-Hartenberg.

Eslabón	θ_i	α_i	d_i	a_i	Rango
r_1	θ_1	90°	r_1	0	-90 a 90
r_2	θ_2	0	0	r_2	0 a 180
r_3	θ_3	0	0	r_3	-90 a 90

Las matrices de transformación que se obtienen con los parámetros de tabla 1 son ecuaciones 1, 2 y 3.

$$ {}^{0}A_{1}=\begin{bmatrix} \mathrm{c}\theta_1 & 0 & \mathrm{s}\theta_1 & 0 \\ \mathrm{s}\theta_1 & 0 & -\mathrm{c}\theta_1 & 0 \\ 0 & 1 & 0 & r_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

$$ {}^{1}A_{2}=\begin{bmatrix} \mathrm{c}\theta_2 & -\mathrm{s}\theta_2 & 0 & r_2\,\mathrm{c}\theta_2 \\ \mathrm{s}\theta_2 & \mathrm{c}\theta_2 & 0 & r_2\,\mathrm{s}\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

$$ {}^{2}A_{3}=\begin{bmatrix} \mathrm{c}\theta_3 & -\mathrm{s}\theta_3 & 0 & r_3\,\mathrm{c}\theta_3 \\ \mathrm{s}\theta_3 & \mathrm{c}\theta_3 & 0 & r_3\,\mathrm{s}\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

La matriz de transformación global $^0A_3$ proveniente de la multiplicación de ecuaciones 1, 2 y 3, está dada por ecuación 4.

$$ {}^{0}A_{3}=\begin{bmatrix} c\theta_1c\theta_2c\theta_3-c\theta_1s\theta_2s\theta_3 & -c\theta_1c\theta_2s\theta_3-c\theta_1c\theta_3s\theta_2 & s\theta_1 & r_2c\theta_1c\theta_2+r_3c\theta_1c\theta_2c\theta_3-r_3c\theta_1s\theta_2s\theta_3 \\ c\theta_2c\theta_3s\theta_1-s\theta_1s\theta_2s\theta_3 & -c\theta_2s\theta_1s\theta_3-c\theta_3s\theta_1s\theta_2 & -c\theta_1 & r_2c\theta_2s\theta_1+r_3c\theta_2c\theta_3s\theta_1-r_3s\theta_1s\theta_2s\theta_3 \\ c\theta_2s\theta_3+c\theta_3s\theta_2 & c\theta_2c\theta_3-s\theta_2s\theta_3 & 0 & r_1+r_2s\theta_2+r_3c\theta_2s\theta_3+r_3c\theta_3s\theta_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4} $$

Para obtener las ecuaciones de la cinemática directa aplicamos la matriz de transformación a ecuación 4 al origen, ecuación 5.

$$ \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}={}^{0}A_{3}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}=\begin{bmatrix} r_2c\theta_1c\theta_2+r_3c\theta_1c\theta_2c\theta_3-r_3c\theta_1s\theta_2s\theta_3 \\ r_2c\theta_2s\theta_1+r_3c\theta_2c\theta_3s\theta_1-r_3s\theta_1s\theta_2s\theta_3 \\ r_1+r_2s\theta_2+r_3c\theta_2s\theta_3+r_3c\theta_3s\theta_2 \\ 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{5} $$

A partir de obtener las ecuaciones cinemáticas se procede a plantear el pseudocódigo para implementar la función que se utilizará en la herramienta computacional para calcular las posiciones del efector final del robot.

Implementación de la cinemática directa

Las ecuaciones obtenidas del modelo cinemático directo se codificaron utilizando el lenguaje de programación C++ para calcular la cinemática directa (ecuaciones simplificadas). Dicho programa se presenta en la figura 3.

```
void ForwardKinematics(double theta1, double theta2, double
theta3, double r0, double r1, double r2, double* x, double* y,
double* z)
{     double R = r2 * cos(theta2) + r3 * cos(theta2 + theta3);
                        *x = R * cos(theta1);
                        *y = R * sin(theta1);
        *z = r1 + r2 * sin(theta2) + r3 * sin(theta2 + theta3);
}
```

Figura 3 Función de implementación de la cinemática directa.

Modelo cinemático inverso

Debido a la simplicidad del robot, cuyo esquema se muestra en figura 4, la cinemática inversa se calculará de manera analítica. El modelo obtenido a partir de este análisis servirá para implementar la función que calcule el valor de las variables articulares en la interfaz de simulación.

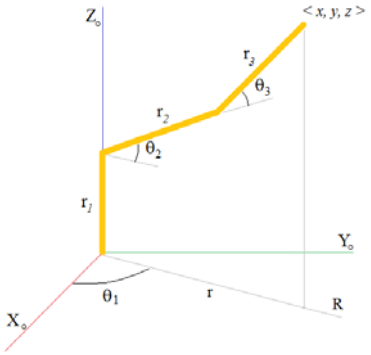

Figura 4 Esquema del robot de tres grados de libertad.

El ángulo θ_1 se obtiene mediante ecuación 6.

$$\theta_1 = \text{Atan}\,2(y, x) \tag{6}$$

Para el ángulo θ_3, observando la figura 5 se tiene que, por teorema de Pitágoras, ecuación 7.

$$r^2 + (z - r_1)^2 = h^2 \tag{7}$$

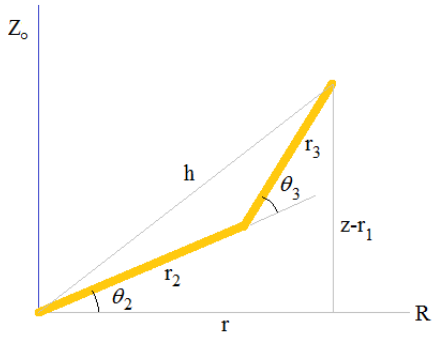

Figura 5 Esquema del robot seccionado.

Por otro lado, usando la ley de coseno se tiene ecuación 8.

$$r_2^2 + r_3^2 + 2r_2 r_3 \cos\theta_3 = h^2 \tag{8}$$

Combinando las ecuaciones 7 y 8, se tendrán ecuaciones 9.

$$\begin{aligned} r^2 + (z - r_1)^2 &= r_2^2 + r_3^2 + 2r_2 r_3 \cos\theta_3 \\ \cos\theta_3 &= \frac{r^2 + (z - r_1)^2 - r_2^2 - r_3^2}{2r_2 r_3} \\ \sin\theta_3 &= \sqrt{1 - \cos^2\theta_3} \end{aligned} \tag{9}$$

Simplificando ecuaciones 9, se obtiene el ángulo θ_3, ecuación 10.

$$\theta_3 = \text{Atan}\,2\left(\sqrt{1 - \cos^2\theta_3}, \cos\theta_3\right) \tag{10}$$

Para el cálculo de θ_2 se utiliza la figura 6.

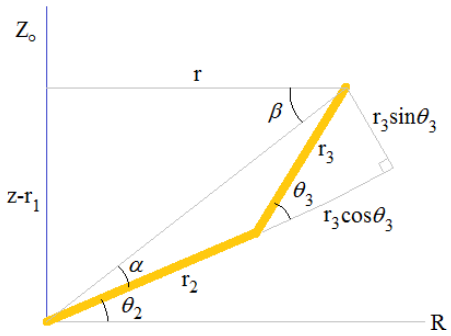

Figura 6 Esquema del robot seccionado con triángulos rectángulos.

Para calcular θ_2 se tiene:

$$\theta_2 = \beta - \alpha \tag{11}$$

En donde β se determina mediante ecuación 12.

$$\beta = \tan^{-1}\left(\frac{z-r_1}{r}\right) = A\tan 2\left(\frac{z-r_1}{\sqrt{x^2+y^2}}\right) \quad \text{y} \quad \alpha = A\tan 2\left(\frac{r_3\sin\theta_3}{r_2+r_3\cos\theta_3}\right) \tag{12}$$

Sustituyendo ecuación 12 en 13 obtenemos θ_2, ecuación 13.

$$\theta_2 = A\tan 2\left(\frac{z-r_1}{\sqrt{x^2+y^2}}\right) - A\tan 2\left(\frac{r_3\sin\theta_3}{r_2+r_3\cos\theta_3}\right) \tag{13}$$

Implementación de la cinemática inversa

La codificación en C++ de la cinemática inversa se presenta en la figura 7 solamente para la primera solución.

```
void InverseKinematics(double x, double y, double z, double r1,
double r2, double r3, double* theta1, double* theta2, double*
theta3)
{     *theta1 = atan2(y, x);
      double Q = (x*x + y*y + (z-r1)*(z-r1) - r2*r2 - r3*r3) / (2
      * r2 * r3);
      *theta3 = atan2(sqrt(1 - Q*Q), Q);
      *theta2 = atan2(z - r1, sqrt(x*x + y*y)) - atan2(r3 *
      sin(*theta3), r2 + r3 * cos(*theta3));
```

Figura 7 Función de implementación de la cinemática inversa.

Debido a la multiplicidad de soluciones para la cinemática inversa es necesario disponer de un criterio que nos permita obtener la mejor solución de entre las posibles, para ello se considerará como solución óptima la que requiera menor movimiento de los eslabones para alcanzar la posición deseada. Para llevar a cabo lo anterior se desarrolló el pseudocódigo mostrado en la figura 8.

```
Theta1, Theta2 y Theta3 son los ángulos actuales

Calcular Theta11, Theta12, Theta31, Theta32, Theta21, Theta22,
Theta23, Theta24

Si |Theta11 - Theta1| < |Theta12 - Theta1|
     Theta1 = Theta11
     Theta3 = Theta31;
     Si |Theta21 - Theta2| < |Theta22 - Theta2|
          Theta2 = Theta21
     En otro caso
          Theta2 = Theta22
     Fin Si
En otro caso
     Theta1 = Theta12
     Theta3 = Theta32;
     Si |Theta23 - Theta2| < |Theta24 - Theta2|
          Theta2 = Theta23
     En otro caso
          Theta2 = Theta24
     Fin Si
Fin Si
```

Figura 8 Pseudocódigo para encontrar la solución óptima para los ángulos del robot.

3. Resultados

La interfaz se diseñó y compiló utilizando el entorno de desarrollo integrado Borland C++ Builder 6 en forma de un software orientado a la simulación y programación mediante comunicación serial del brazo robot como se muestra en la figura 9.

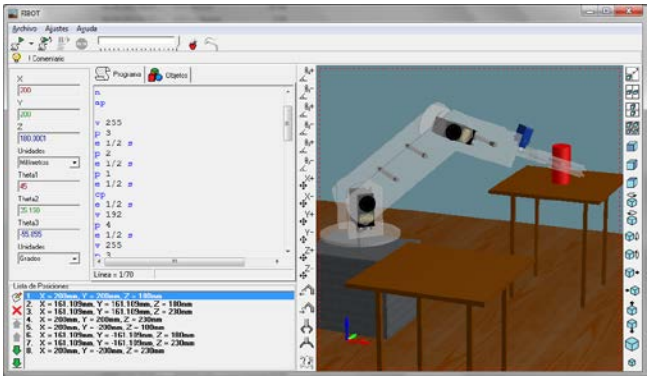

Figura 9 Pantalla principal de la interfaz de simulación y programación.

Dentro de la interfaz, el brazo se puede mover modificando de manera individual ya sean los ángulos de rotación de cada eslabón o las coordenadas de la pinza mediante unos botones que incrementan o decrementan en una cantidad predeterminada estos valores, de igual forma se puede modificar la velocidad que se realiza este movimiento y el estado y grado de apertura de la pinza.

La interfaz contiene unos cuadros con texto en donde pueden ingresarse los valores numéricos que se desea adquieran los ángulos de rotación o las coordenadas de la pinza para un control más preciso. Adicionalmente se pueden grabar puntos en una lista, los cuales se van almacenado en una lista para posteriormente ser referenciados en el programa o hacer que el brazo vaya a esa posición haciendo doble clic sobre el elemento en la lista.

Es posible dentro de la interfaz la programación mediante código de rutinas sencillas para que el brazo las ejecute, para ello se diseñó un lenguaje de programación del brazo orientado a tal efecto, en la figura 10 se presenta ejemplo de un programa con comentarios.

```
! Inicio de programa
     N                    ! Ir a posición de inicio
     AP                   ! Abrir pinza
     CI 10 20 25 CM ABS   ! Cinemática inversa a x=10 y=20 z=25
     E 1 S                ! Esperar 1 s
     CP                   ! Cerrar la pinza
     P 1                  ! Ir al punto 1 en la lista
     AP                   ! Abrir pinza
     N                    ! Ir a posición de inicio
! Fin de programa
```

Figura 10 Fragmento de código para el movimiento del robot.

El editor de programa dentro de la interface tiene coloreado de sintaxis, como en el programa de ejemplo, los comentarios se colorean automáticamente en verde y las palabras clave en azul, además, al escribir un comando se muestra una referencia rápida de la sintaxis del comando lo cual agiliza la programación.

Una vez que el programa está escrito, si es que no existen errores de sintaxis, puede ejecutarse y simularse mediante un modelo en 3D con las mismas proporciones que el brazo robot; adicionalmente, es posible introducir primitivas en 3D dentro de la simulación, con las cuales puede interactuar la pinza del brazo robot, como se muestra en la figura 11.

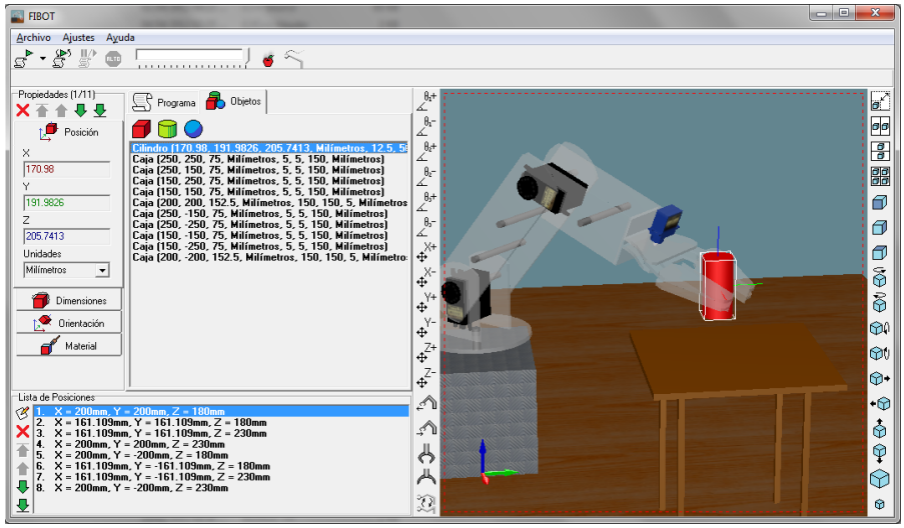

Figura 11 Vista previa del robot manipulando objetos.

Cabe mencionar que el programa, la lista de puntos y primitivas 3D, pueden guardarse en archivos que posteriormente pueden ser abiertos y editados en este mismo software; esto para no perder el trabajo cuando se cierre la aplicación, hay

que tener en consideración la medición de la distancia del objeto al robot y al momento de guardar los puntos de posición del brazo no mover los objetos o posicionarlos como estaban antes.

4. Conclusiones

Los programas comerciales de simulación robótica usualmente poseen una alta calidad y tecnologías avanzadas, pero al mismo tiempo incrementan de manera significativa el costo de desarrollo para los usuarios. Adicionalmente, muchos proveedores comerciales proporcionan solamente simuladores para sus propios productos, lo que resta versatilidad y flexibilidad en los desarrollos robóticos.

El software desarrollado permite aprender de manera fácil y rápida los principios y teorías de programación usadas comúnmente en los robots. Se diseñó e implementó un software para emular un ambiente robótico para educación que permite simular y programar un robot de manera virtual.

El software implementado permite el desarrollo de competencias básicas en el estudiante como trabajo en equipo, pensamiento sistemático, apertura, individualidad, identificación y solución de problemas, gestión de proyectos, y otras en la formación integral de un profesionista.

Como trabajo futuro es la adición de una biblioteca con robots de diferentes configuraciones, dispositivos de transferencia y objetos primitivos.

5. Bibliografía y Referencias

[1] K. Kimoto, S. Yuta, "Autonomous mobile robot simulator-a programming tool for sensor-based behaviour". Autonomous Robots. Vol. 1. Jun 1995. Pp. 131-148.

[2] A. Liu, J. Newsom, C. Schunn, R. Shoop, "Students learn programming faster through robotic simulation". Tech Directions. Vol. 72. Mar. 2013. pp. 16-19.

[3] O. Michel, "Cyberbotics ltd webots professional mobile robot simulation". International Journal of Advanced Robotic Systems. Vol. 1. 2004. Pp. 39-42.

[4] E. Rohmer, S. P. N. Singh, M. Freeze, "V-REP: a versatile and scalable robot simualtion framework". Proc. Int. Conf. Intelligent Robots and Systems (IROS 13). Mar. 2013. Pp. 1321-1326.

[5] Easy Rob 3D robot simulation tool. http://www.easy-rob.com/easy_rob.html. Agosto 2016.

[6] L. Xu, "Robotstudio: A modern IDE-based approach to reality computing". Proc. The 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 07). Mar. 2007. Pp. 440-444.

[7] J. Jackson, "Microsoft robotics studio: A technical introduction". IEEE Robotics and Automation Magazine. vol. 14. no. 4. Dec.2007. Pp. 82-87.

[8] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan, "Modular open robots simulation engine: MORSE". Proc. Int. Conf. Robot. And Automation (lCRA 11). 2011. Pp. 46-51.

[9] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Weeler, A. Y. Ng, "ROS: An open-source robot operating system," Proc. ICRA Workshop Open Source Software, 2009, Pp. 1-6.

[10] ARS: Phyton robotics simulation. https://ars-project.readthedocs.io/en/latest/index.html. Agosto 2016.

[11] N. G. Tsagarakis, B. Vanderborght, M. Laffranchi, D. G. Caldwell, "The mechnical design of the new lower body for the child humanoid robot ‘iCub’". Proc. Int. Conf. Intelligent Robots and Systems (IROS 09). Oct. 2009. Pp. 4962-4968.

[12] V. Tikhandoff, A. Cangelosi, P. Fitzpatrick, G. Metta, L. Natale, F. Nori, "An open-source simulator for cognitive robotics research: the prototype of the iCub humanoid robot simulator". Proc. 8th Workshop on Performance Metrics for Intelligent Systems (PerMIS '08). 2008. Pp. 57 -61.

[13] J. Medina Cervantes, N. Castro Gutiérrez, E. Mejía Sánchez, R. Villafuerte Díaz, “Aplicación móvil para el control de un brazo robot”. Revista Iberoamericana de Producción Académica y Gestión Educativa. Volumen 4. Núm. 1. Junio 2016. Pps.21-43.

[14] N. García Hurtado, L. Castillo García, A. Escobar Jiménez, “Plataforma educativa ROBI”. Revista Colombiana de Tecnologías de Avanzada. Volumen 1. Número 19, 2012

[15] F. Bravo Sánchez, A. Forero Guzmán, “La robótica como un recurso para facilitar el aprendizaje y desarrollo de competencias generales”. Teoría de la Educación, Educación y Cultura en la Sociedad de la Información. vol. 13. núm. 2. 2012. pp. 120-136.

[16] J. Craig, Introducción a la Robótica. 3a Edición. 2006. Ed. Pearson.

[17] M. Spong, S. Hutchinson, M. Vidyasagar, Robot Modelling and Control. 2005. Ed. John Wiley & Sons.

6. Autores

M.C. Braulio Cruz Jiménez obtuvo su grado de Maestría en Ciencias con especialidad en Automatización por el Tecnológico de Monterrey, su área de investigación son los sistemas de control y procesamiento de señales.

M.C. Jannette Contreras Rivero obtuvo su grado de Maestría en Ciencias con especialidad en Sistemas Ambientales por el Tecnológico de Monterrey, su área de investigación es matemáticas aplicadas.

M.I. Ricardo Peón Escalante obtuvo su grado de Maestro en Ingeniería Mecánica especialidad en diseño mecánico por la Universidad Nacional Autónoma de México y su área de investigación es síntesis óptima de sistemas mecánicos.

Dr. Luis Ricalde Catellanos obtuvo su grado de Doctor en Ingeniería Eléctrica por el Centro de Investigación y de Estudios Avanzados del IPN Unidad Guadalajara, su área de investigación son los sistemas de control utilizando redes neuronales.