Model of SCORBOT-ER 4PC in Simulink

Guillermo Flores Cuevas, Abraham Mejía Medina, Alberto Salinas Chávez
150951, 142406, 149038
gflores10.1926@gmail.com, manuelmejiam13@gmail.com, alberto260994@gmail.com

***Abstract*—Obtener una representación del brazo robótico, *SCORBOT-ER 4PC*, con el propósito de facilitar la modificación de su comportamiento sin la necesidad de interactuar con él fisícamente para evitar el riesgo en cuanto a descomponer un componente o bien una parte del mismo mientras se realizan pruebas. Se utilizó *Matlab R2015b* para la implementación del modelo así como también se realizó el modelado de las piezas en *NX 11* para una mejor visualización del brazo robótico; de la misma forma con el uso del framework que nos ofrece *Simulink*, *Simscape*, se llegó a un modelo funcional con un control para manipular la posición del robot desde una interfaz gráfica introduciendo los ángulos de giro para cada uno de sus ejes, 4 en total.**

I. Introduction

En la actualidad persiste el problema de modificar el comportamiento de un brazo robótico con el riesgo de descomponer o dañar de alguna manera el robot físico, por eso es que hoy en día las simulaciones juegan un papel muy importante dentro del mundo de la robótica. La ingeniería asistida por computadora juega un rol esencial en la fase de pruebas con maquinaria costosa como lo son los brazos robóticos en el caso de una empresa manufacturera de productos. Si contamos con un modelo lo suficientemente fiable de un robot físico es posible hacer pruebas en la simulación para analizar la respuesta del mismo y determinar si es viable llevar la prueba a la etapa física. La simulación permite reproducir virtualmente las respuestas del robot y estudiar su comportamiento para analizar el impacto de posibles cambios o comparar diferentes alternativas de diseño sin el riesgo de los experimemtos a escala real. Por medio de la simulación podemos analizar información con el fin de obtener el diseño más eficiente con diferentes objetivos.

- Optimización de recursos.
- Validación de la inversión a realizar.
- Identificación de restricciones del proyecto.
- Evaluación de alternativas para el proyecto.
- Simulación de condiciones externas.

II. Objetivo y Propósito

Hacer un modelo funcional del brazo robótico *SCORBOT-ER 4PC* en Simulink con ayuda del framework Simscape para que después pueda ser utilizado para realizar pruebas antes de llevar a cabo otros proyectos de mayor complejidad que impliquen el uso del robot físico o bien como material didáctico para los ingenieros durante la carrera.

III. Requerimientos y específicaciones

El modelo debe ser capaz de replicar el movimiento del brazo robótico, se debe implementar el uso de un motor DC para que su salida proporcione el torque necesario para el movimiento utilizando un control con el propósito de fijar el movimiento de cada parte del robot de acuerdo a los valores, ángulos, ingresados por el usuario. Se requiere de una versión de Matlab en donde Simulink cuente con el framework Simscape y Nx 11 para el modelado 3D de las distintas piezas que componen el *SCORBOT-ER 4PC*. Las especificaciones del modelo del motor DC implementado en simulink están dadas por su datasheet. 2

Fig. 1. Motor dc marca Pittman.

Assembly Data	Symbol	Units	Value	
Reference Voltage	E	V	12	
No-Load Speed	S_{NL}	rpm (rad/s)	71	(7.4)
Continuous Torque (Max.)[1]	T_C	oz-in (N-m)	480	(3.4E+00)
Peak Torque (Stall)[2]	T_{PK}	oz-in (N-m)	2585	(1.8E+01)
Weight	W_M	oz (g)	23.7	(671)
Motor Data				
Torque Constant	K_T	oz-in/A (N-m/A)	3.25	(2.29E-02)
Back-EMF Constant	K_E	V/krpm (V/rad/s)	2.40	(2.29E-02)
Resistance	R_T	Ω	0.71	
Inductance	L	mH	0.66	
No-Load Current	I_{NL}	A	0.33	
Peak Current (Stall)[2]	I_P	A	16.9	
Motor Constant	K_M	oz-in/√W (N-m/√W)	4.11	(2.90E-02)
Friction Torque	T_F	oz-in (N-m)	0.80	(5.6E-03)
Rotor Inertia	J_M	oz-in-s^2 (kg-m^2)	1.0E-03	(7.1E-06)
Electrical Time Constant	τ_E	ms	1.06	
Mechanical Time Constant	τ_M	ms	8.5	
Viscous Damping	D	oz-in/krpm (N-m-s)	0.053	(3.5E-06)
Damping Constant	K_D	oz-in/krpm (N-m-s)	12.5	(8.5E-04)
Maximum Winding Temperature	θ_{MAX}	ºF (ºC)	311	(155)
Thermal Impedance	R_{TH}	ºF/watt (ºC/watt)	56.3	(13.5)

Fig. 2. Datasheet motor dc.

Asimismo, para correr la interfaz gráfica se necesita de *GUIDE*, el ambiente para el desarrollo de interfaces gráficas proporcionado por *MATLAB*.

IV. Diseño de solución

En primer lugar es necesario caracterizar el motor a utilizar para que produzca el torque y la velocidad necesarios para mover cada uno de los grados de libertad del brazo, esta parte del diseño de la solución no fue necesario llevarla a cabo

pues ya se contaba con la información de los motores usados en el robot dado que el modelo se encuentra físicamente en el laboratorio. Una vez que se haya caracterizado el motor ideal es necesario buscar su respectivo datasheet que nos servirá para implementarlo en *Simulink*.

Una vez que se realizan los respectivos cálculos para implementar dentro de *MATLAB* el motor, es necesario definir cuales serán las variables de entrada del mismo y cual es la respuesta que nosotros buscamos para conseguir el movimiento de los grados de libertad; la velocidad a la que gira el modelo del motor implementado depende únicamente de una entrada, el voltaje inducido al mismo, de igual manera, la salida está dada en términos del torque generado por la inyección de dicho voltaje.

Pasando a la parte del Diseño Asistido por Computadora, haciendo uso del software *NX 11*, es necesario modelar los distintos componentes básicos del manipulador, por ello las piezas modeladas fueron las siguientes:

- Base del robot: No proporciona ningún grado de libertad pues está fija. 3
- Base superior del robot: Proporciona el primer grado de libertad girando sobre su eje Z 4
- Parte anterior del Brazo: Análogamente, es la parte del hombro al codo proporciona un segundo grado de libertad sobre el eje X. 5
- Parte posterior del Brazo: Análogamene, es la parte del codo a la muñeca brindándonos un tercer grado de libertad sobre el eje X. 6
- Garra: Pieza de sujetamiento, dándonos el cuarto grado de libertad sobre el eje X. 7

Nota: Se toma un sistema de referencia en donde el eje X está posicionado horizontalmente, Y es perpendicular a él y Z apunta hacia arriba en el mundo físico. Cuando se describe el grado de libertad es respecto al sistema de referencia de la pieza más no del mundo.

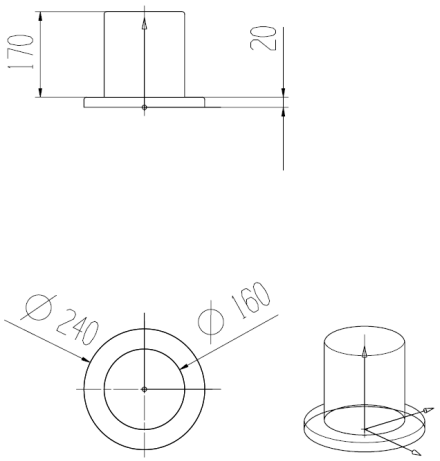

Fig. 3. Draft de la base.

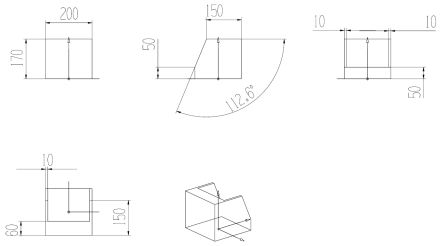

Fig. 4. Draft de la base superior.

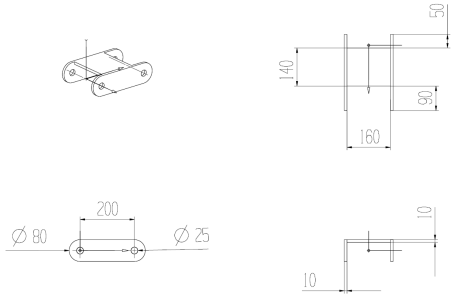

Fig. 5. Draft de la parte anterior del brazo.

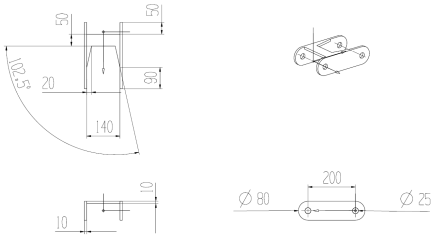

Fig. 6. Draft de la parte posterior del brazo.

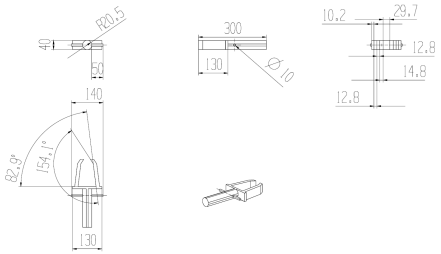

Fig. 7. Draft de la garra.

Para llevar a cabo una simulación más real es necesario contar con las medidas lo más aproximadas a la realidad de cada una de las piezas por ello mismo se determinaron con el uso de un Vernier y posteriormente se pasó al modelado de las mismas en el software. Se crearon los planos correspondientes a las piezas indicando las medidas significativas de cada una de ellas, a su vez, se exportaron archivos ".stl" de cada una de ellas para poder importarlas desde *Simulink* como sólidos.

Una vez ya en *Simulink* con todas las piezas importadas como sólidos es necesario ubicar cada una de ellas en posición para obtener la estructura física del *SCORBOT-ER 4PC*, esto se logra implementando las traslaciones y rotaciones correspondientes a cada una de las piezas respecto al sistema de referencia del mundo establecido ya por *Simulink*, se usó el bloque de "rigid transform" para llegar a las transformaciones correspondientes. Tras haber realizado este proceso con todas las piezas se llego al modelo de la figura 8.

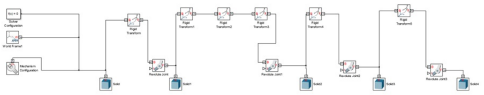

Fig. 8. Modelo del brazo robot en simulink.

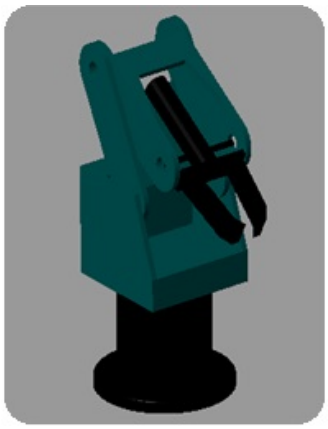

Fig. 9. Vista del brazo robot en simulink.

Ahora bien, se necesitan definir las articulaciones del robot para determinar el eje sobre el cual gira cada pieza, para ello se usaron "spherical joint" pues posteriormente dicho bloque nos servirá para determinar el ángulo de giro actual de la pieza (por ángulo de giro se entiende que es de acuerdo al eje sobre el que rota la pieza en específico). Cabe mencionar que dentro de las dificultades del problema se encuentran los sistemas de referencia que toma Simulink pues éstos están dados por el sistema de referencia sobre el cual se modeló la pieza en específico; en caso de tener errores de este tipo, dentro de la simulación pueden dar lugar a varias situaciones de entre ellas:

- Que el eje especificado sobre el cual se requiere el giro en la pieza de simulink no sea el correcto aún y que dentro del programa se haya especificado correctamente
- Que el centro definido en la pieza a partir del cual va a girar no sea el deseado, llegando a resultar en una pieza que rote sobre su centro de gravedad, lo cual en este caso no es lo que se requiere pues por ejemplo las partes del brazo giran a partir de su base más no de su centro de gravedad.

Un aspecto técnico importante es definir adecuadamente a partir de donde se va a modelar la pieza dentro del sketch en *Nx* para evitar estos problemas.

Ahora bien, hay que inducirle un torque a la articulación para lo cual tenemos que introducir nuestro modelo del motor DC al proyecto principal. A continuación se presenta una explicación más detallada sobre las ecuaciones obtenidas a partir del motor y como tal el modelo implementado del mismo:

$$Vs = RiL(t) + L\frac{di(t)}{dt} + VB(EMF) \tag{1}$$

$$VB = KB * w(t) \tag{2}$$

$$T = IL * \frac{dw(t)}{dt} \tag{3}$$

$$T = KT * i(t) \tag{4}$$

$$KT * i(t) = IL * \frac{dw(t)}{dt} \tag{5}$$

$$Vs = RiL(t) + L\frac{di(t)}{dt} + KB * w(t) \tag{6}$$

$$\frac{dw(t)}{dt} = \frac{KT}{IL} * i(t) \tag{7}$$

$$\frac{di(t)}{dt} = \frac{1}{L} * [Vs - Ri(t) - KB * w(t)] \tag{8}$$

Al llegar a las ecuaciones (7) y (8) lo que se necesita es integrarlas para obtener por un lado la velocidad y por otro la corriente en el motor. LLegando así al modelo de la figura número 10.

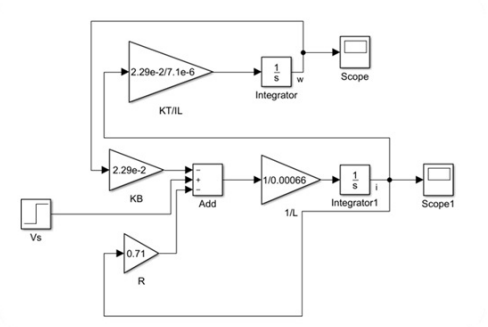

Fig. 10. Modelo del motor dc en simulink.

Para facilitarnos la visualización del proyecto en conjunto, el modelo del motor pasó a ser un solo bloque con una entrada y una salida ya descritas con anterioridad. Ya teniendo un torque que inyectar y las articulaciones bien definidas para rotar cada pieza ahora es necesario implementar un control que nos permita girar cada una de ellas ”n” grados especificados por un usuario.

Para implementar un control adecuado primero es necesario conocer la posición actual de la pieza que deseamos llegue a ”n” grados de giro, los ”spherical joint” cuentan con un atributo, position, que nos regresa un vector de 1x4 representando la posición actual de la pieza en cuaterniones. Por ello mismo, es necesario traducir esos cuaterniones a ángulos de coordenadas polares. Entonces, lo que se hace es extraer ese vector de 1x4 y traducirlo a ángulos de rotación (Quaternions to Rotation Angles, bloque de *Simulink*) 11 y finalmente obtener de los 3 ángulos de rotación el que nos importa a nosotros dependiendo de la pieza de que se esté hablando.

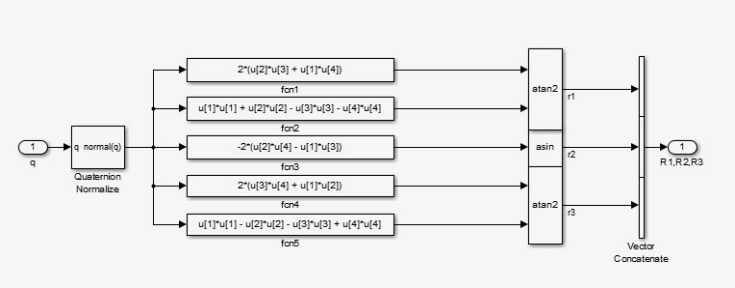

Fig. 11. Función que convierte quaterniones a ángulos de rotación.

```
function y = fcn(position)
%#codegen
y = abs(((180)/(pi))*position(3));
end
```

Fig. 12. Función que convierte radianes a grados.

El control implementado fue a lazo cerrado y consta de dos partes principales:

- Determinar cuándo debe parar de rotar cierta articulación de acuerdo a los datos especificados por el usuario.
- Determinar si la entrada en ángulos fue un ángulo positivo o bien un ángulo negativo

Se logró el control obteniendo la posición angular actual del grado de libertad, comparándola en todo momento con un switch que determinaria el voltaje a inducir al motor, en el momento en que la articulación llegara al grado deseable se le inducia un voltaje de cero al motor DC; por otro lado para controlar el sentido del giro (ángulos negativos) se implementó un segundo switch que verificara si el ángulo ingresado por el usuario era positivo o bien negativo, en caso de ser negavito se multiplica por un ”-1” el voltaje para así cambiar el sentido del giro de la articulación.

Hemos acabado con el proceso en su totalidad, únicamente falta describir la GUI implementada. 13

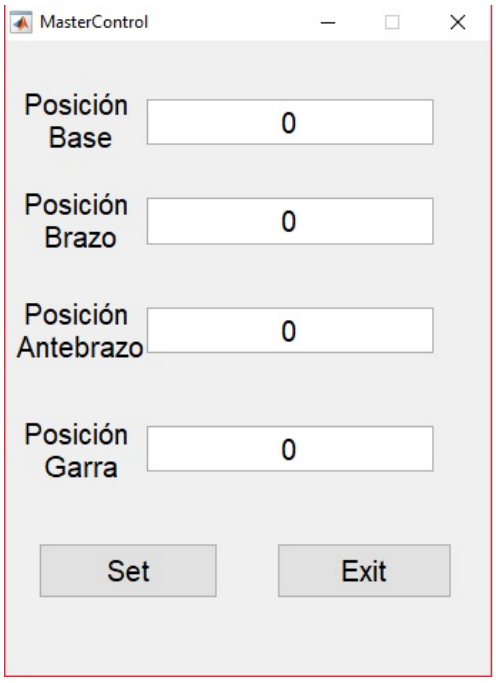

Fig. 13. Interfaz gráfica de usuario.

Como se logra observar tenemos únicamente 2 botones: uno para correr la simulación con los ángulos deseados por el usuario y otro para salir de la simulación; a su vez, se tienen los distintos campos en que se modifican las especificaciones en cuanto a ángulos deseados para cada articulación. Una vez que se especifican todos los ángulos se oprime el boton ”set” y correrá la simulación. Lo importante dentro de la GUI es la implementación de dicho botón, pues en él se cambian los valores de los umbrales ya descritos dentro del proyecto de simulink en la parte del control permitiendo así llegar a la posición deseada al *SCORBOT-ER 4PC* a cada articulación.

```
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
set_param('SimRobot/Switch1','Threshold',num2str(...
    abs(str2double(get(handles.voltTxtbox,'string')))));
set_param('SimRobot/Switch2','Threshold',num2str(...
    abs(str2double(get(handles.edit2,'string')))));
set_param('SimRobot/Switch3','Threshold',num2str(...
    abs(str2double(get(handles.edit3,'string')))));
set_param('SimRobot/Switch4','Threshold',num2str(...
    abs(str2double(get(handles.edit4,'string')))));


set_param('SimRobot/baseAngle','value',get(handles.voltTxtbox, 'string'));
set_param('SimRobot/arm1Angle','value',get(handles.edit2, 'string'));
set_param('SimRobot/arm2Angle','value',get(handles.edit3, 'string'));
set_param('SimRobot/clawAngle','value',get(handles.edit4, 'string'));

set_param('SimRobot','SimulationCommand','start');
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Fig. 14. Código de la interfaz de usuario.

V. Verificación y Pruebas / Problemas enfrentados

Para la verificación del correcto funcionamiento del motor DC construido se obtuvieron distintas respuestas en el tiempo variando el voltaje inyectado. De esta manera podemos observar gráficamente con un ”scope” la velocidad 15 y la corriente 16 alcanzada a partir de los distintos voltajes inducidos.

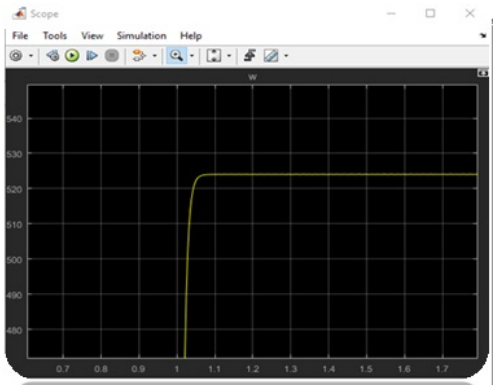

Fig. 15. Velocidad angular.

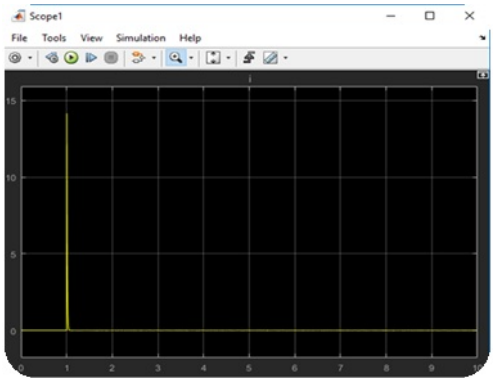

Fig. 16. Corriente.

Por su parte, la verificación y las pruebas del correcto funcionamiento de los sólidos ya estructurados correctamente para darle forma al brazo robótico, se obtuvieron inyectándole torque a las distintas articulaciones y observando la respuesta de las mismas. 17 y 18

Los principales problemas que enfrentamos fueron la falta de frameworks dentro de la versión más reciente con que se cuenta de *MATLAB*, por otro lado, inicialmente las piezas en *Nx* se encontraban erróneamente modeladas resultando en los problemas ya descritos con anterioridad respecto a los sistemas de referencia por lo que tuvieron que ser modeladas nuevamente desde cero; finalmente, el uso de ”spherical joints” no se encontraba contemplado en un principio pues al inicio contabamos con ”revolute joints”, sin embargo este tipo de articulaciones únicamente permite rotación en la pieza sobre su eje Z cuando nosotros realmente necesitamos para tres de los cuatro grados de libertad rotación sobre el eje X.

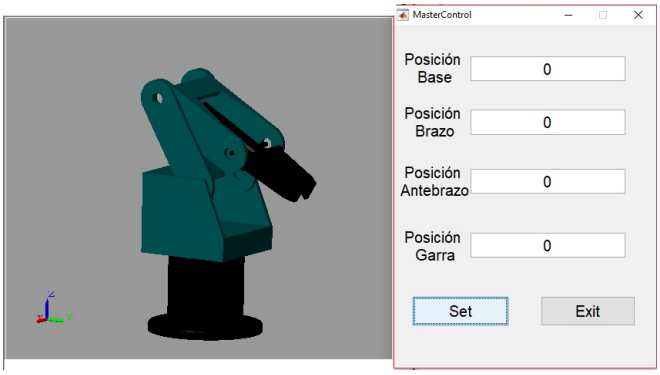

Fig. 17. Posición inicial del robot.

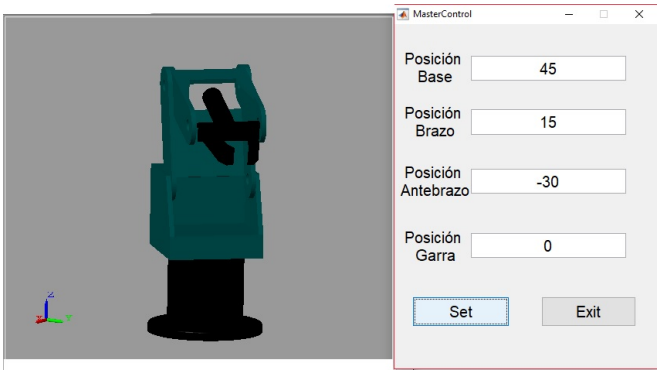

Fig. 18. Posición final del robot.

VI. Validación de requerimientos / Problemas enfrentados, recomendaciones

Se llegó a implementar un modelo cercano a la realidad en cuanto al peso y tamaño de las piezas, así como también se consiguió una solución en cuanto a la posibilidad de observar el comportamiento del brazo robótico con el uso de una interfaz amigable y sencilla para el usuario, en definitiva se cumplió con el objetivo de implementar la simulación como material didáctico para estudiantes de ingeniería. Ahora bien, pasando a la solución implementada para simular el comportamiento real del robot, nuestro modelo no considera una parte muy importante dentro del mismo que es la gravedad, el problema reside en la falta de conocimiento sobre teoría del control de sistemas no lineales pues nuestro problema necesariamente debe ser modelado dinámicamente tomando a consideración los centros de gravedad de las piezas, sus inercias y demas; que posteriormente necesitarían cada una de ellas un control PID para su correcto funcionamiento en el mundo real lo cual lleva evidentemente a una complejidad mucho mayor a la establecida en este reporte pero un mejor acercamiento a la realidad.

VII. Conclusion

Se llegó a una solución viable puesto que la simulación es funcional y no tiene problema alguno con la recepción de los distintos valores que el usuario pueda ingresar, de

igual manera representa una solución práctica y eficiente para disminuir el riesgo a la hora de realizar pruebas físicas con el brazo robótico. Por otro lado, el proyecto puede ser mejorado constantemente, en un inicio se podrían crear reestricciones para evitar el traslape de las partes sólidas del brazo, de igual manera con una versión más reciente de *MATLAB* se podría conseguir por medio de un ”Dashboard” la corrida de la simulación en tiempo real, pues cada vez que se ingresan nuevos valores angulares la simulación corre desde la posición inicial más no se mantiene en la última posición tras la última corrida. Por útlimo, se podrían implementar controladores PID por cada articulación para poder considerar la gravedad como un factor existente en el proyecto.

El objetivo principal se cumplió pues la simulación puede ser utilizada con fines didácticos para estudiantes de ingeniería y más adelante puede servir para proyectos de mayor complejidad que se vean en la necesidad de simular el *SCORBOT-ER 4PC* tomando a consideración factores más realistas como lo son las inercias de cada una de las masas, sus centros de gravedad, etc. En cuanto a la distribución del trabajo en el equipo, cada uno de nosotros realizó en algún momento cada parte que compone el proyecto en su totalidad, todos estuvimos conscientes en todo momento del avance del mismo así como las dificultades que se presentaban al tratar de solucionar el problema.