ESCUELA NAVAL DE SUBOFICIALES A.R.C. "BARRANQUILLA"

SEXTANTE

Desarrollo de un sistema de control posicional e interfaz gráfica para el SCORBOT–ER 5PLUS

Brayan Rueda Mayorga, Henry Armando Pinzon Gonzalez, Jorge Luis Zarate Arroyo, Saul Antonio Perez Perez & Javier Jimenez-Cabas

[a] *Ingeniería mecatrónica, Universidad Autónoma del Caribe, Barranquilla, Colombia.*
brued97@hotmail.com, hapg96@gmail.com, zaratej_96@hotmail.com, saul.perez@uac.edu.co
[b] *Centro de Investigaciones, Universidad del Costa, Barranquilla, Colombia.*
javier.jimenez@uac.edu.co

Recibido: Diciembre 12, 2018.
Recibido en su versión corregida: Febrero 26, 2019.
Aceptación: Marzo 22, 2019

Cómo citar: Mayorga Rueda, B., Pinzón Gonzalez, H.A., Zarate Arroyo, J.L., Perez Perez, S.A. & Jimenez-Cabas, J. (2019). Desarrollo de un sistema de control posicional e interfaz gráfica para el Scorbot-ER 5plus. Revista Sextante, 20, pp. 33- 43, 2019.

Resumen
Este articulo muestra cómo se desarrolla un Sistema de control y una interfaz gráfica para la manipulación de un robot SCORBOT-ER 5plus ubicado en el laboratorio de robótica de la Universidad Autónoma del Caribe con el fin de agregar una herramienta adicional a través de la cual los estudiantes puedan aprender de manera más practica el campo de la robótica. Esto será posible gracias al uso de técnicas y conceptos de electrónica, electricidad, control, mecánica y robótica entre otros, que ayudará a desarrollar el sistema de control posicional necesario, así como la creación e implementación de una interfaz gráfica para poner el robot de nuevo en funcionamiento en los laboratorios de la institución.

Palabras claves: *Cinemática directa; Cinemática inversa; Control; Posicionamiento; Procesamiento de datos; Robótica; Scorbot-ER 5Plus.*

Development of a positional control system and graphic interface for Scorbor-ER 5plus

Abstract
This article shows how to develop a control system and a graphical interface to the manipulator robot SCORBOT-ER 5plus in the Robotic Laboratory of the Universidad Autónoma del Caribe in order to add an extra tool through which all students can feed in a more practical way in the field of robotics. This will be possible thanks to the use of techniques and concepts of electronics, electricity, control, mechanics and robotics among others, which will help to develop the necessary positional control system, as well as the creation and implementation of a graphical interface to put the robot back into operation in the laboratories of the institution.

Keywords: *Control; Data processing; Direct kinematics; Positioning; Reverse kinematics; Robotics; Scorbot-ER 5Plus.*

1. Introducción

Los robots industriales son, actualmente, herramientas de gran aceptación y utilización por partes de empresas en diferentes sectores del país, debido a que con la ayuda de estas máquinas automatizadas sus productos aumentan en distintas facetas como: variedad, cantidad, calidad e incrementan significativamente la velocidad de producción sin descuidar su durabilidad, por lo que los ingresos de las empresas empiezan a crecer. La utilización de estos robots en la industria lleva a que el sector académico se vea obligado a tener un nuevo propósito específico, el de generar un personal (estudiantes) capacitado para la utilización, manipulación, creación; y que además posean las aptitudes para realizarle mantenimiento a dichas máquinas (Constaín, 2015).

Sin embargo, el principal obstáculo para las academias es el gran costo de estas máquinas, causando que no todas las instituciones tengan la facilidad de exhibir y enseñar en físico a sus estudiantes dichas máquinas. Con lo importante que se ha convertido este tema para la industria, las instituciones se ven obligadas a realizar un gran esfuerzo para no estancarse en el constante avance tecnológico y adquirir estos dispositivos. Infortunadamente, en la mayoría de los casos solo logran comprar máquinas de segunda mano, que en repetidas ocasiones se encuentran en un estado deteriorado o con muy poca vida útil restante, imposibilitando el uso adecuado de los equipos.

El presente trabajo busca desarrollar un sistema de control y una interfaz gráfica al robot manipulador SCORBOT-ER 5plus de 5 grados de libertad (ver Figura 1) disponible en el Laboratorio de Robótica de la Universidad Autónoma del Caribe (Intelitek, 1996) debido a que el sistema que controla al robot se encuentra fuera de servicio a causa de su mal funcionamiento. Esto obligará a la utilización de técnicas de electrónica, electricidad, control, mecánica y robótica para desarrollar el sistema de control posicional necesario, además, la creación e implementación de una interfaz gráfica para poner nuevamente en marcha el robot en los laboratorios de la institución.

2. Metodología

Para el diseño e implementación de un sistema de control posicional de un brazo robótico se requiere tener los dispositivos necesarios que permitan captar la información de los sensores y procesarla; y, por último, hacer las decisiones adecuadas, según lo que se haya pedido.

El desarrollo de este sistema de control no solo implica tener los dispositivos adecuados, también es necesario encapsularlos y aislarlos de toda perturbación externa como la humedad, el polvo, la suciedad, contacto físico, entre otras. Esto permitirá un mayor rendimiento del sistema y una mayor durabilidad de los dispositivos electrónicos. El recubrimiento con el cual se aislará el sistema será el de un cubo de acrílico como el que se muestra en la Figura 2.

Dentro de esta caja estarán todos los elementos que conformarán el sistema de control. Se tomó la decisión del acrílico por tres razones principales; precio, estética y facilidad a la hora de armar o desarmar en el caso de que se necesite hacer alguna modificación o un mantenimiento.

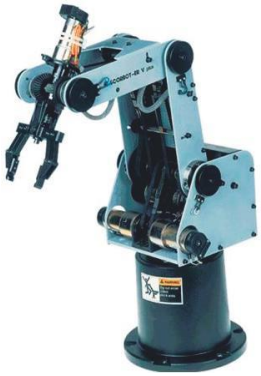

Figura 1. Imagen del robot Scorbot-er Vplus.
Fuente: (Intelitek, 1996).

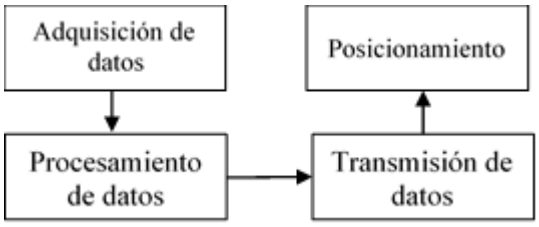

Figura 2. Ciclo del proceso.
Fuente: Los autores.

La Figura 2 muestra las 4 etapas en las que se divide el proyecto. Se inicia con la adquisición de datos. Aquí, por medio de la interfaz gráfica, el usuario ingresa la posición deseada para el robot. Estos datos son emitidos hacia el software Matlab para su procesamiento. En esta etapa Matlab decide si los valores ingresados son compatibles o no (solo son compatibles números que se encuentren dentro del límite de movimiento del robot), en caso tal de que sean incompatibles con el sistema, saldrá un error, normalmente no es compatible cuando el dato ingresado supera el ángulo de alcance del motor y/o robot. En cambio, si son compatibles se prosigue a transmitir los datos al Arduino, para que este ponga en movimiento los motores y empiece la lectura de los sensores ópticos, y de esta manera completar el posicionamiento deseado del robot.

El sistema recorre todas estas etapas para poder llegar, ya sea por cálculo de cinemática directa o de cinemática inversa, a la posición que se desea al ingresar los datos. En la adquisición de datos es en donde el sistema espera que el usuario ingrese los datos en la interfaz gráfica. Estos datos son los necesarios para que el sistema funcione. Además, estos son los datos que posicionarán al robot donde el usuario desee. Al recibir la información la interfaz gráfica almacena y transmite cada dato, uno por uno, a Matlab para empezar a procesarlos. En esta etapa el software se encarga de primero definir si la información recibida es posible de procesar, es decir, si los datos adquiridos en la etapa anterior se pueden utilizar en su totalidad para mover el robot.

Las incompatibilidades de mayor repetición son las de colocar un ángulo mayor al del límite o colocar letras en vez de números. Una vez resuelto el inconveniente de compatibilidad, Matlab empieza a realizar todos los cálculos necesarios (ya sea por cinemática directa o cinemática inversa) para ofrecer a la siguiente etapa la cantidad de recorrido que tiene que trasladarse cada motor, para así llegar a la posición deseada. Al finalizar el procesamiento de datos esta información es transmitida de Matlab hacia Arduino, para que este último mande a los motores a que se muevan en la dirección establecida y hasta el nivel que se haya calculado en la etapa anterior. Finalmente, en la etapa de posicionamiento el Arduino empieza a leer los pulsos que mandan los sensores ópticos. Con esta lectura se almacenan y se cuentan la cantidad de pulsos hasta llegar a la cantidad calculada en la etapa de procesamiento de datos, una vez que los pulsos contados sean iguales a los pulsos calculados el Arduino da la orden de parar los motores y se finaliza el programa debido a que ya se llegó a la posición deseada.

Sistema de control

El sistema de control para el robot está implementado en un Arduino Mega 2560, que es el 'cerebro' que lee a los sensores y toma las decisiones. Los sensores encargados de dar la información de la posición de las articulaciones del robot son 2: los encoders ópticos H22L01 que nos indican hacia dónde y cuánto se está moviendo el robot; y los de finales de carrera, que nos confirman la posición inicial del robot. Para que la tarjeta del Arduino pueda manipular a los actuadores, necesita la ayuda del driver L298N, que proporcionará una salida de 12v a 24v y hasta 2A por canal, permitiendo controlar por completo el motor sin generar pérdidas de potencia en el sistema. Además, al sistema de control posee unos pilotos para fallas del sistema, con esto el operador podrá ver, analizar y actuar según el piloto que se active.

Los cálculos de la cinemática son dos: el primero es el de cinemática directa, que nos ayudará a determinar cuál es la posición y orientación del extremo final del robot con respecto a un sistema de coordenadas que se toma como referencia; mientras que el segundo, el de cinemática inversa, se utiliza cuando se desea encontrar los valores que deben adoptar las coordenadas articulares del robot, para que su extremo se oriente y se posicione según una determinada localización espacial. Con estos dos cálculos podremos mover los motores y, por consiguiente, los eslabones del robot de la manera deseada y a las posiciones que se quiera, siempre y cuando estén dentro del límite del área de trabajo del robot.

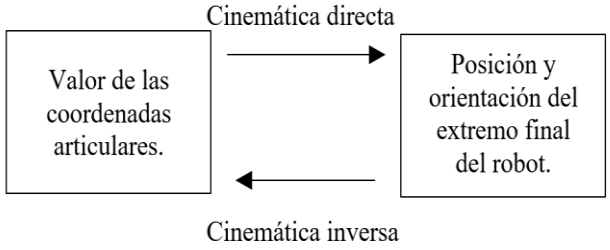

Figura 3. Relación entre cinemática directa e inversa.
Fuente: Los autores.

En la Figura 3 se ejemplifica la relación entre cinemáticas, que serán descritas a continuación.

Cinemática directa

En la cinemática de un robot se encuentran dos problemas fundamentales a resolver: el primero de ellos se conoce como el problema de cinemática directa, y consiste en determinar cuál es la posición y orientación del extremo final de robot, con respecto a un sistema de coordenadas que se toma como referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot; del segundo, denominado cinemática inversa, se hablará más adelante.

Como ya se indicó anteriormente, para realizar los cálculos de la cinemática directa son necesarios los parámetros de Denavit Hartenberg, dichos parámetros son 4 (θ_i,d_i,a_i,∝_i) y están condicionados bajo las condiciones geométricas del robot, es decir, por cada eslabón y por cada articulación. Estos parámetros se definen como:

- θ_i: Este es el ángulo que se forma entre los ejes X_(i-1) y X_i, medido con respecto al eje Z_(i-1). Este parámetro es variable si la articulación es giratoria.

- d_i: Es la distancia que existe entre el sistema de coordenadas i – 1 hasta la intercepción del eje Z_(i-1) con X_i a lo largo del eje Z_(i-1). Este parámetro es variable si la articulación es prismática.

- a_i: Es la distancia a lo largo del eje X_i que va desde la intersección del eje Z_(i-1) con el eje X_i hasta el origen del sistema i-ésimo en el caso de las articulaciones giratorias.

- ∝_i: Es el ángulo de separación del eje Z_(i-1) y el eje Z_i, medido en un plano perpendicular al eje X_i, utilizando la regla de la mano derecha.

Para obtener los parámetros mencionados anteriormente es necesario agregar primero las coordenadas del robot como se puede ver en la Figura 4 donde se pueden ver las coordenadas escogidas para el robot. La amarilla representa el eje Y, la azul Z, y la roja X.

Después de obtener los parámetros y las coordenadas el siguiente paso es calcular los parámetros Denavit Hartenberg (D-H) que se pueden ver en la Tabla 1. En el parámetro θ_i todos están sin especificación debido a que todas las articulaciones del ScorBot son rotativas y por ende este valor siempre está en constante variación.

Matlab nos ofrece la posibilidad de simplificar los cálculos complejos, es por esto que con el siguiente código Matlab calculará la cinemática directa para luego integrarla a la interfaz gráfica y poder mover el robot con este método, es decir, que los resultados obtenidos serán las coordenadas que representan los ángulos ingresados en el entorno gráfico.

Código de Matlab para el cálculo de cinemática directa

%Medidas entre cada origen de las articulaciones del robot. Están en centímetros.

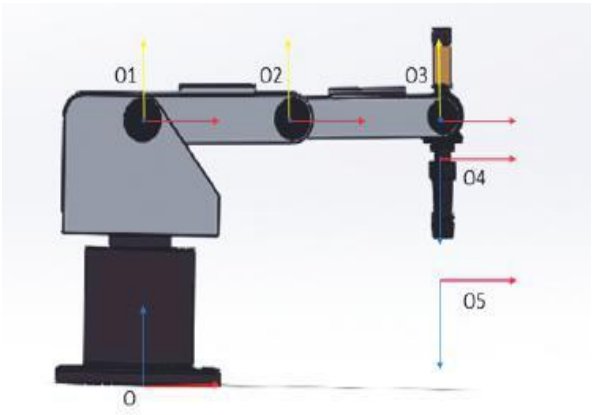

Figura 4. Coordenadas del robot.
Fuente: Los autores.

Tabla 1. Parámetro D-H

Eslabón	θ_i	d	a	$\propto_i$
0	q_1	L	a	$-\frac{\pi}{2}$
1	q_1	0	a	0
2	q_1	0	a	0
3	q_1	0	0	$-\frac{\pi}{2}$
4	q_1	L	0	0

Fuente: Los autores.

```
L1=35; L2=14; al=2.5; a2=22.5; a3=a2;
% Parámetros Denavit-Hartenberg del robot
Teta = [q (1) q (2) q (3) q (4) q (5)];
d = [Ll 0 0 0 L2];
Alfa = [-pi/2 O O -pi/2 0];
a = [al a2 a3 0 0]; 1
```

```
%Cálculo de matrices de transformación homogénea

A01 = DenavitHartenberg (teta (1), d (1), a (1), alfa (1));
Al2 = DenavitHartenberg (teta (2), d (2), a (2), a1fa (2));
A23 = DenavitHartenberg (teta (3), d (3), a (3), alfa (3));
A34 = DenavitHartenberg (teta (4), d (4), a (4), alfa (4));
A45 = DenavitHartenberg (teta (5), d (5), a (5), alfa (5));

%Cálculo de la matriz T

A02 = A01 * Al2;
A03 = A02 * A23;
A04 = A03 * A34;
A05 = A04 * A45;
T = A05;

% Vector de posición (x, y, z) de cada sistema de coordenadas

X0 = 0; y0 = 0; z0  = 0;
Xl = A01 (1,4); yl =A01 (2,4);  zl= A01 (3,4);
X2 = A02 (1,4); y2 =A02 (2,4);  z2= A02 (3,4);
X3 = A03 (1,4); y3 =A03 (2,4);  z3= A03 (3,4);
X4 = A04 (1,4); y4 =A04 (2,4);  z4= A04 (3,4);
X5 = A05 (1,4); y5 =A05 (2,4);  z5= A05 (3,4);
```

Un ejemplo de los resultados que otorgará este código se podrá ver en el conjunto de Ecuación 1-4

$$\theta = (2 * \pi * \theta)/360 \tag{1}$$

$$\propto = (2 * \pi * \propto)/360 \tag{2}$$

$$A_i = \begin{pmatrix} \cos\theta & -\sin\theta * \cos\alpha & \sin\theta * \sin\alpha & L * \cos\theta \\ \sin\theta & \cos\theta * \cos\alpha & -\cos\theta * \sin\alpha & L * \sin\theta \\ 0 & \sin\alpha & \cos\alpha & d \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

$$A = A_0 * A_1 * A_2 * A_3 * A_4 \tag{4}$$

Con la utilización del conjunto de Ecuación 1-4 se podrán encontrar las coordenadas necesarias para llegar a los ángulos deseados que se le ingresarán por medio de la interfaz.

Con los parámetros de la Ecuación 5-6 del robot son con los que se harán todos los cálculos relacionados a los movimientos de este mismo. Para llegar a las coordenadas deseadas primero se tienen que ingresar unos ángulos (en este ejemplo serán todos 0 para la posición inicial) y luego calcular la matriz homogénea de cada eslabón hasta que al final se puedan multiplicar todas y así obtener la matriz final.

En la Figura 5 se pueden observar los valores de las variables mencionadas para los cálculos de cinemática directa y la resolución de las coordenadas que se obtienen para los ángulos de posición inicial, en Ecuación 7-14.

$$L1 = 17;\ L2 = 27;\ L3 = 18;\ L4 = 12.5; \tag{5}$$

$$d1 = 21.5;\ d2 = 0;\ d3 = 0;\ d4 = 0; d5 = L3 + L4; \tag{6}$$

$$A_0 = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 21.5 \\ 0 & 0 & 0 & 1 \end{matrix} \tag{7}$$

$$A_1 = \begin{matrix} 1 & 0 & 0 & 17 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \tag{8}$$

$$A_2 = \begin{matrix} 1 & 0 & 0 & 27 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \tag{9}$$

$$A_3 = \begin{matrix} 1 & 0 & 0 & 30.5 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \tag{10}$$

$$A_4 = \begin{matrix} 1 & 0 & 0 & 30.5 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \tag{11}$$

$$A = A_0 * A_1 * A_2 * A_3 * A_4 \tag{12}$$

$$A = \begin{bmatrix} 1 & 0 & 0 & 105 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 21.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

$$\begin{matrix} X = 105 \\ Y = 0 \\ Z = 21.5 \end{matrix} \tag{14}$$

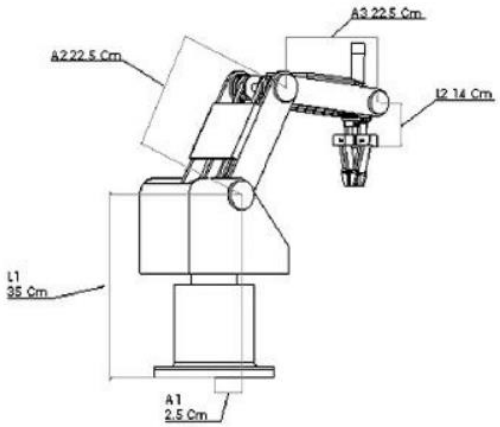

Figura 5. Medidas de los parámetros D-H.
Fuente: Los autores.

Cinemática inversa

De igual manera que los cálculos de la cinemática directa, para hacer los cálculos de cinemática inversa también se usó el software Matlab. Con este se pudieron calcular todas las matrices de manera más fácil, además, se pudo comunicar Matlab con el Arduino paralelamente, para transmitir los datos a la interfaz y esta pudiera mover el robot a la posición deseada.

Para estos cálculos se tuvo que hacer un análisis geométrico de cómo está compuesto el robot para poder obtener las fórmulas con las que se obtendrán las coordenadas del robot. Además, se utilizan los parámetros relacionados con los eslabones del robot, es decir, los mismos parámetros que se utilizaron en los cálculos de la cinemática directa.

Se puede observar parte del código que se tiene implementado en el robot para que este pueda realizar los cálculos de cinemática inversa.

Código para cálculo de cinemática inversa

```
Px = str2double (get (handles.X, String•)):
Py • str2double (get (handles.T, 'Stringrn:
Pz = atz2double (get (hand1ea.2, IString,));
Balance • str2double (oetlhandles.balance,
'Untos));
Cabeceo • atr2double (get (handles.cabeceo,
IStrino1));
L2 = 18: L3 = 11.5; LI = 3.2; LS = 13.51; dl =
17.17; d2 = 0; d3 = O; d4 = O;
d6 = L4+L5;
Rxy = agft ((Px"2)+ (Py"2));
Alpha • atand (Py/Px);
Beta = atand ((d2+d3+d1)/ (wart ((kxy^2)-
((d2+d3+d4) ^2))));
Theta = Alpha + Beta;
Lf (Px==0) a (Py==0) 1
   Theta=0;
End
Xprials=Rxy:
Yptlma=Pz;
Afx=coad (Cabeceo) • (d6);
Afy=alnd (Cabeceo) • (d6):
LadoA•ptima-Aly-d1;
LadoB=Xprima-Afx;
Coup.- ((Led0A^2) + (Lad013^2)-(1.2"2)-(L3-2))/
(2•L2•L3):
Theta3= atand ((sgrt (1-(cosq3^2)))/coaq3):
Beta2= atand (LadoA / LadoB): Alpha2=
atand11L•sind (Theta3))/ (1.2+ (L3•cosd
(Theta3))));
Theta2• Beca2 - Alpha2;
Theta4=Cabeceo-Theta2-Theta3;
Theta5= Balance;
If (Theta<0.3 16 Theta >-0.3)
   Theta = 0;
If (Theta<0.3 && Theta >-0.3)
   Theta • O;
End
```

En este algoritmo parcial de Matlab, el software extrae la información digitada en los cuadros de texto X, Y, Z, balance y cabeceo. Luego, a partir de la información obtenida, se calculan los ángulos a los que se debe mover cada motor para alcanzar la posición deseada, para ello se emplean formulas cinemáticas y los parámetros D-H (Tabla 1). Al finalizar los cálculos, los ángulos obtenidos como resultados son mostrados en sus respectivos cuadros, y en el caso de que se esté dentro de la simulación Matlab actualiza los datos en Simulink para que el robot se mueva y el usuario pueda observar previamente cómo quedaría posicionado el robot con los datos ingresados. Para la realización de los cálculos es necesario contar con los datos mencionados en Ecuación 5-6.

$$\begin{matrix} Px = 105 \\ Py = 0 \\ Pz = 21.5 \end{matrix} \qquad (15)$$

Lo primero para calcular la cinemática inversa es tener unas coordenadas en las cuales queremos que el robot este posicionado. Por facilidad y para comprobar de que estos cálculos funcionan se

utilizarán las coordenadas que se obtuvieron como resultado en la cinemática directa. Al final los ángulos que se obtengan acá deben ser los mismos que se ingresaron en el cálculo anterior (ver Ecuación 15).

$$Rxy = \sqrt{Px^2 + Py^2} \quad (16)$$

$$\propto = \tan^{-1}(Py/Px) \quad (17)$$

$$\beta = \tan^{-1}(d2 + d3 + d4)/\sqrt{(Rxy^2) - (d2 + d3 + d4)^2} \quad (18)$$

$$\theta = \alpha + \beta \quad (19)$$

Luego de tener las coordenadas deseadas, se pasa a encontrar por medio del método geográfico las ecuaciones necesarias para calcular los ángulos (θ) de cada eslabón (Ecuación 16-19).

De esta manera se obtiene el primer ángulo de la cinemática inversa.

Para calcular los siguientes ángulos se continúa utilizando el análisis geométrico y, además, se integran dos conceptos nuevos relacionados con el actuador final. Balance y cabeceo, estos son unos ángulos de navegación derivados de los ángulos de Euler, que ayudan a representar la orientación de un objeto (el actuador final en este caso) en tres dimensiones (Antonio Barrientos). Debido a que la posición a la cual va a llegar el robot es la inicial, es decir, todos sus ángulos son 0, el balance y cabeceo toman estos valores de igual manera. A continuación, en el conjunto de Ecuación 20-28, se prosigue con las ecuaciones para el cálculo de los ángulos restantes.

De esta manera se obtienen todos los ángulos a los cuales se tiene que mover el robot para llegar a las coordenadas ingresadas al inicio. En el código de Matlab como resultado se obtuvo la Ecuación 29:

$$Afx = \cos(Cabeceo) * (d5) \quad (20)$$

$$Afy = \sin(Cabeceo) * (d5) \quad (21)$$

$$q_3 = ((Pz - Afy - d1)^2 + (Rxy - Afx)^2 - (L1^2) - (L2^2))/(2 * L1 * L2) \quad (22)$$

$$\theta_3 = \tan^{-1}\sqrt{\frac{1 - (q_3^2)}{q_3}} \quad (23)$$

$$\beta_2 = \tan^{-1}(Pz - Afy - d1)^2/(Rxy - Afx)^2 \quad (24)$$

$$\propto_2 = \tan^{-1}(L2 * \sin(\theta_3))/(L1 + (L2 * \cos(\theta_3)) \quad (25)$$

$$\theta_2 = \beta_2 - \propto_2 \quad (26)$$

$$\theta_4 = Cabeceo - \theta_2 - \theta_3 \quad (27)$$

$$\theta_5 = Balance \quad (28)$$

$$Angulos = [0]\ [-0.2]\ [0]\ [0.2]\ [0] \quad (29)$$

Es decir, todos los ángulos dieron 0 aproximadamente como debería ser.

3. Resultados

Al finalizar este documento y oficializar su entrega física, se espera haber cumplido con los objetivos definidos con anterioridad, es decir, recuperar el robot, dejándolo 100% funcional. Además, con manuales/guías para que de esta manera toda la comunidad educativa de la universidad pueda hacer prácticas y aprender de una manera más práctica y didáctica.

Interfaz gráfica

Para que el estudiante o quien quiera manipular el robot pueda hacerlo de una manera didáctica y directa, se ha diseñado una interfaz gráfica con el software de Matlab que permitirá controlar el robot, sin embargo, debido a problemas de deterioro en la estructura mecánica del robot se tuvo que cambiar la interfaz. Por tal motivo se mostrará la interfaz deseada y la interfaz que se usa actualmente. El diseño de dicha interfaz deseada se puede ver en la (Figura 6).

Cinemática directa e inversa: En este apartado se podrán colocar los valores de los ángulos a los cuales el que este manejando el robot quiera que este llegue, el software se encargará de realizar el cálculo para hallar los valores de la cinemática, ya sea directa o inversa.

Cabeceo y balance: Estos son unos ángulos que se encuentran al mover el actuador final, dichos ángulos serán calculados conjunto a los de la cinemática.

El único indicador que posee la interfaz es el de estado, dicho indicador, como su nombre señala, es el que nos informa si el robot está conectado y con comunicación (Verde) o si se encuentra sin comunicación (Rojo).

El botón con nombre Cerrar Gripper tiene como función cerrar o abrir el Gripper según sea el caso. El Gripper tiene como posición predeterminada

abierta. El nombre del botón cambiará con respecto a la posición actual del Gripper.

Por último, en los apartados de simulación y enviar; se realizan una simulación y se envían las coordenadas al robot para empezar a moverse.

En el caso de la interfaz actual se pueden evidenciar los cambios que se hicieron; se eliminaron las casillas de motor 4 y motor 5, puesto que el Gripper se mantendrá en todo momento (para evitar que los problemas mecánicos afecten todo el funcionamiento del robot), y se eliminaron también las casillas de cabeceo y balanceo, pues estas afectan también el movimiento del Gripper. Los cambios realizados permiten trabajar al robot, con la única diferencia de que el Gripper se mantendrá siempre situado en su posición inicial (donde se activa el final de carrera). El diseño de dicha interfaz que se usará al finalizar el proyecto se puede ver en la (Figura 7).

Simulación

En este apartado la interfaz genera una simulación de los movimientos que hará el robot dependiendo de las coordenadas que el usuario ingresó en el software. Para la simulación se usará una imagen del robot diseñada en el software CAD SolidWorks (Figura_8).

Los que se encargan de mover el robot en la simulación son unos bloques en Simulink, que procesan, determinan, calculan y ejecutan las coordenadas propuestas. Dichos bloques se pueden ver en la (Figura 9).

Controlador

Para el arreglo del robot se pensó en utilizar diferentes tipos de controladores. Se empezó por un controlador general que tuviera la capacidad de gobernar los movimientos del robot, pero al implementar dicho controlador este funcionaba de manera incorrecta debido a la gran cantidad de datos que debía procesar, esto causaba que se presentara un porcentaje de error considerable en el estado estacionario del sistema, por lo que nunca llegaba a estar cerca al valor que debía estabilizarse. Al ver estos problemas y ver que la cantidad de datos a procesar era imposible de disminuir se optó por

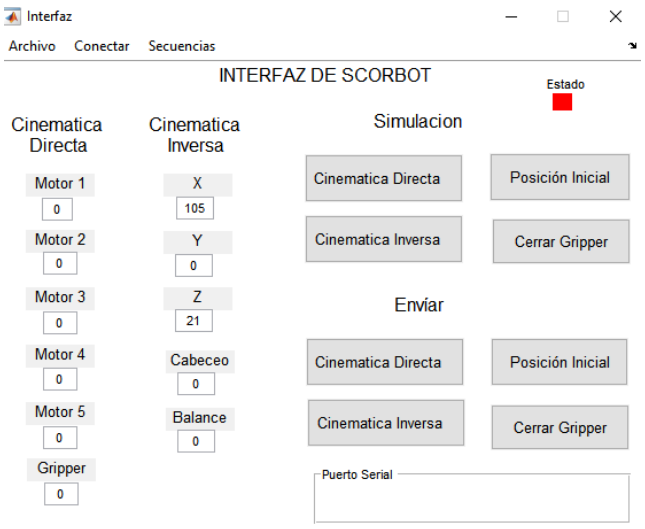

Figura 6. Interfaz gráfica deseada.
Fuente: Los autores.

Figura 7. Interfaz gráfica usada.
Fuente: Los autores.

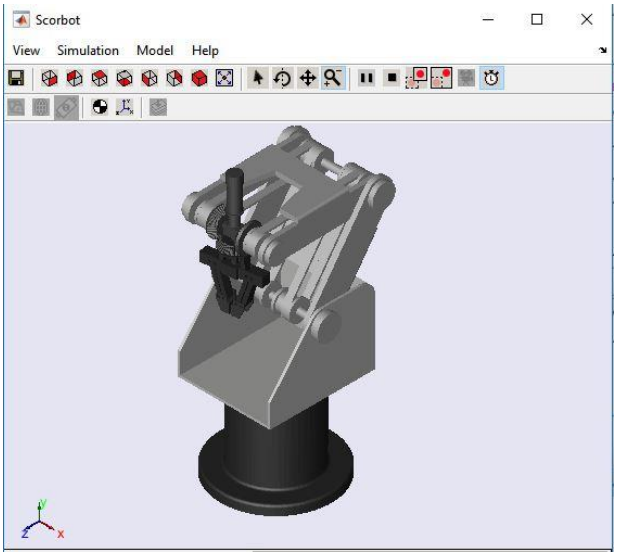

Figura 8. Imagen de simulación.
Fuente: Los autores.

dividir las cargas, es decir, crear un controlador individual que se encargara del movimiento de un solo motor, es decir, en vez de controlar todos los motores a la vez se enfocara simplemente en controlar un solo motor para luego unir el resultado de cada uno de ellos y obtener un resultado global deseado. Se utiliza una función de transferencia que es la que controla a la ecuación del motor (Figura 10) y una saturación para determinar el rango de operación del sistema. En la siguiente figura se puede observar la ecuación de los motores (todos los motores tienen la misma ecuación).

En la Figura 11 se puede apreciar la ecuación que representa los movimientos de cada motor del robot. Todas las ganancias se sintonizaron a prueba y error, en ninguna se utilizó algún método de sintonización. Para saber si los controladores funcionaron se graficó sin la utilización del controlador y con la utilización de este. Como se puede observar en la Figura 12 la señal que no tiene controlador queda con la onda lejos del setpoint deseado (10, en este caso) sin embargo, en la que tiene el controlador si alcanza y se estabiliza en el valor deseado.

4. Recomendaciones

Para que el robot quede con una mayor optimización se proseguirá a realizar una serie de recomendaciones que se dividirán en 3 grupos, pero que están todos enlazados entre sí.

- Hardware.
- Programación.
- Generales.

Hardware

El cambio de hardware es lo principal para que el robot pueda funcionar de la mejor manera. La recomendación más importante es la de realizar un cambio de tarjeta de control. Actualmente funciona con un Arduino Mega, el cual tiene grandes ventajas, sin embargo, es limitado en cuanto a velocidad y procesamiento de datos, por lo que con un cambio hacia un FPGA se podría tener una lectura más veloz y precisión de los sensores, lo que permitirá un control más eficiente de los motores, aumentando la suavidad y la exactitud de los movimientos. En la Figura 13 se puede observar el dispositivo que se recomienda. El núcleo de esta placa se basa en el Intel FCyclone V SoC FPGA. Este chip de Intel incorpora los últimos núcleos integrados Cortex-AP de doble núcleo con una lógica programable de primera clase para ofrecer diseños flexibles y una estructura de hardware que los usuarios pueden definir, además, esta placa también incluye un USB-Blaster II integrado, SDRAM de 1 GB, cabeceras de expansión de 2x40 pines, red Gigabit Ethernet y un ADC de 12 bits de resolución (Krambeck, 2018) que no solo mejorará el rendimiento del robot, sino también su interfaz gráfica.

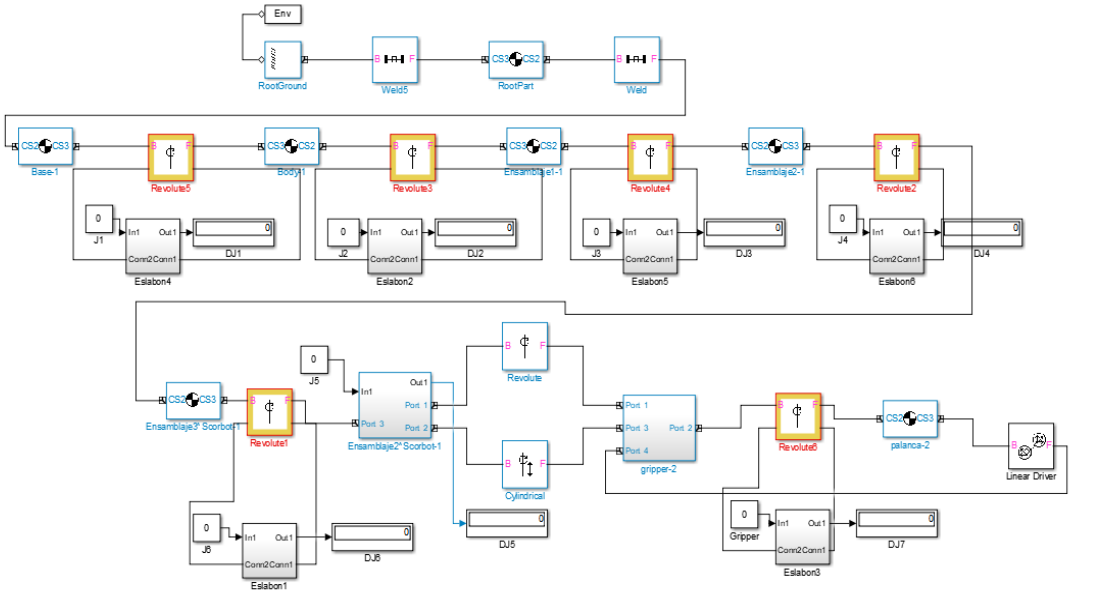

Figura 9. Bloques de control de la simulación.
Fuente: Los autores.

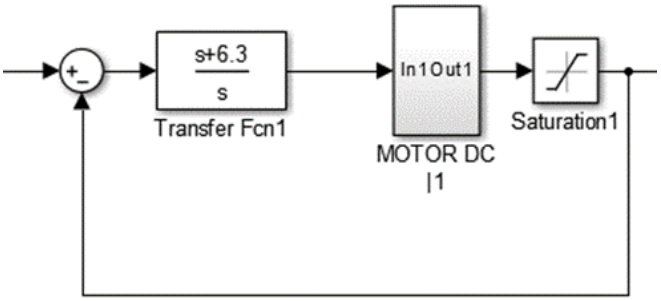

Figura 10. Bloques del controlador.
Fuente: Los autores.

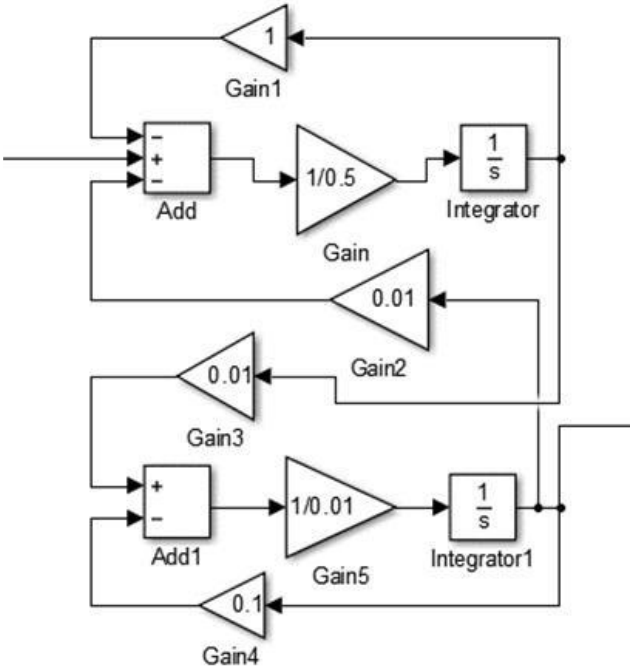

Figura 11. Ecuación de los motores.
Fuente: Los autores.

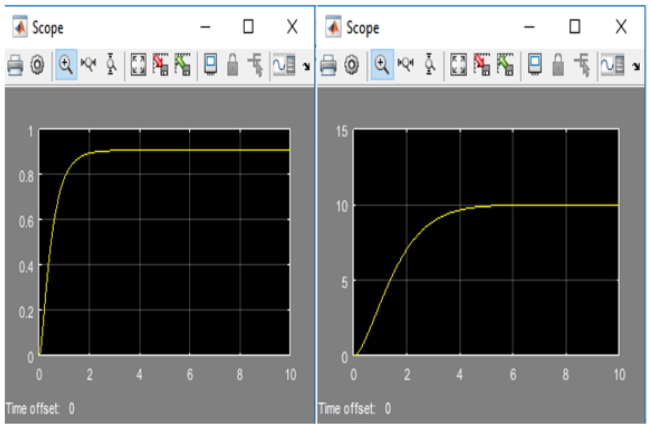

Figura 12. Grafica sin controlador (izquierda) y con controlador (derecha).
Fuente: Los autores.

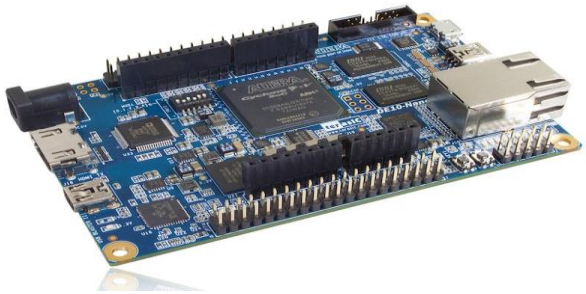

Figura 13. Terasic DE10-Nano (Electronics, 2017).
Fuente: Los autores.

Programación

En este apartado se recomienda optimizar la lectura de datos y la lógica de control utilizada para poder optimizar el rendimiento del robot. No obstante, la programación debe ir ligada al cambio de hardware recomendado, debido a que sin ese cambio no podrá ser optimizado de forma notable el robot, porque lo más importante ahora mismo para el robot es tener el hardware más apto para funcionar de la mejor manera.

Generales

Para una mayor duración con el robot funcionando de óptima forma se recomendarán algunos aspectos generales para el mantenimiento y cuidado del ScorBot.

- Realizarles mantenimiento a las correas cada 6 meses.

- Aplicarle lubricante para prevenir daño en los engranajes y disminuir desgaste.

- Aplicarle antioxidante a la estructura metálica del robot.

- Mantener en un ambiente con temperaturas entre 20° y 25°.

- Mantener en un ambiento con baja humedad.

- Hacerle limpieza cada año a la caja del controlador.

- No cargar objetos de mayor peso que el límite establecido en los datos técnicos.

Al seguir estas recomendaciones se podrá conservar el robot con mayor vida de uso.

5. Conclusiones

Se pudo observar un correcto funcionamiento del sistema de control posicional diseñado tanto como en el software de simulación Matlab, como en la implementación al robot. Todos los motores respondieron adecuadamente al controlador llegando siempre a la posición deseada, con excepción al motor de Gripper, debido a un problema con el sensor óptico de posicionamiento que el actuador final posee.

Infortunadamente no se contó con los fondos monetarios necesarios para un reemplazo del mismo (lo cual sería ideal) y se intentó hacerle mantenimiento, sin embargo, solo se pudo tener apto para su uso por un corto lapso de tiempo, obstruyendo de esta manera el completo funcionamiento del robot, no obstante, el ScorBot quedó funcionando en un 80% de su totalidad sin haber realizado ningún cambio a su estructura, es decir, se tienen 4 de sus 5 grados de libertad, con excepción de movimiento rotatorio del actuador final debido al problema mencionado anteriormente. Si siguen las recomendaciones otorgadas se podrá dejar funcionando al 100% el sistema de control del robot.

En tanto a la interfaz gráfica se puede observar cómo esta sí quedó totalmente funcional y didáctica, además, con un apartado de simulación en el que se puede ver un correcto funcionamiento del controlador y del robot si se tuvieran a la mano todas las piezas de este en buen estado.

Se contará con unos manuales de usuarios para que el estudiante o quien vaya a realizar prácticas en el robot, pueda guiarse antes de utilizarlo.

Por último, se dejaron listas unas guías de laboratorio didácticas con todas las actividades que se pueden realizar con el estado actual del robot y lo principal que se puede aprender a la hora de realizar prácticas con el ScorBot. Dichos manuales y guías están en el laboratorio de robótica de la Universidad Autónoma del Caribe.

Referencias

Antonio Barrientos, L. P. (s.f.). Fundamentos de robotica. Madrid: Concepcion Fernandez Madrid.

Constaín, A. &. (30 de Enero de 2015). ResearchGate. Recuperado el 27 de Agosto de 2017, de https://www.researchgate.net

Electronics, D.-K. (14 de 04 de 2017). digikey. Obtenido de https://www.digikey.com.mx/es/product-highlight/t/terasic-tech/de10-nano

Intelitek. (1996). SCORBOT-ER 5Plus. Manchester.

Krambeck, D. (Febrero de 2018). Obtenido de All About Circuits: https://www.allaboutcircuits.com/news/top-three-FPGA-development-boards-new-designers/