DISEÑO DE LIBRERÍA CON ENFOQUE DIDÁCTICO PARA CONTROL DE ROBOTS MANIPULADORES EN SISTEMAS EMBEBIDOS

LIBRARY DESIGN WITH DIDACTIC APPROACH FOR CONTROL OF ROBOT MANIPULATORS IN EMBEDDED SYSTEMS

Francesco García Luna
Universidad Autónoma de Ciudad Juárez, México
francesco.garcia@uacj.mx

Alma Rodríguez Ramírez
Universidad Autónoma de Ciudad Juárez, México
alma.rodriguez.ram@uacj.mx

David Luviano Cruz
Universidad Autónoma de Ciudad Juárez, México
david.luviano@uacj.mx

Recepción: 14/septiembre/2019 **Aceptación:** 29/octubre/2019

Resumen

En el presente artículo se muestra una herramienta computacional para el control de posición y/o orientación de un robot manipulador de n-grados de libertad. La herramienta cuenta con las funciones necesarias para calcular la cinemática directa del robot mediante matrices de transformación homogénea de rotación y traslación, la matriz Jacobiana de velocidades lineales y/o angulares, y el control de posición y/o orientación mediante un esquema de control PID. Se muestran resultados del uso de la librería en un experimento de control de posición de un robot de 3 grados de libertad de revoluta (RRR), donde la posición del efector final converge en la referencia en tiempo finito.

Palabra(s) Clave: Didáctico Librería, Manipulador, Robot, Sistema embebido.

Abstract

In the present paper, a computational tool for position/orientation control of a n-degrees of freedom manipulator robot is shown. The tool contains the necessary

functions in order to calculate the robot's forward kinematics by means of the rotation/translation homogeneous transformation matrices, the linear/angular velocities Jacobian matrices, and the position/orientation control by means of a PID scheme. The results using the library in a 3 rotational degrees of freedom robot manipulator set-point control is shown, where the final effector's pose converges at the reference in finite time.

Keywords**:** *Didactic, Embedded system, Library, Manipulator, Robot.*

1. Introducción

En cuanto a sistemas embebidos en contexto académico se refiere, es inevitable pensar en la plataforma Arduino. Arduino es un microcontrolador de una sola placa (SBM, por sus siglas en inglés), de acceso libre que cuenta con entradas/salidas digitales y analógicas, y distintos protocolos de comunicación (Serial, I2C, SPI). Esta plataforma es utilizada mayormente para el prototipado de sistemas mecatrónicos gracias a su bajo costo y fácil programación utilizando un lenguaje de programación exclusivo basado en Processing.

Debido a las características de Arduino, han existido numerosos trabajos dedicados a diseñar y programar librerías o scripts enfocados en resolver alguna situación en particular. Tal es el caso donde se diseñó una librería para controlar sistemas SCADA mediante el protocolo RS-232 y Modbus [Tipsuwanporn, Numsomran, Samaimak, & Harnnarong, 2014]; como resultados se mostró una comparativa con un sistema convencional, resultando su sistema más robusto. El mismo año, se utilizó la plataforma Arduino para establecer comunicación con un PLC mediante el protocolo Modbus [Kuang, 2014],. El sistema funcionaba estableciendo al PLC como el sistema maestro y al Arduino como el sistema esclavo. Del mismo modo, existen otros trabajos que se han enfocado en revisar las capacidades que tiene un SBM modular para adquisición de señales y sistemas de control [Cvjetkovic & Matijevic, 2016], demostrando que estos sistemas son adecuados para darle solución a numerosos problemas.

También, se han utilizado las SBM en conjunto con paquetes de software especializado para complementar sus funciones. En el año 2016 se propuso una

alternativa para la experimentación en el diseño de filtros digitales mediante un Arduino DUE y Matlab [Fernández et al., 2016]. Utilizando el sistema propuesto son capaces de visualizar en casi tiempo real el comportamiento de los filtros digitales.

Existen otros trabajos en donde se diseñó una interfaz gráfica de usuario (GUI, por sus siglas en inglés) multiplataforma para la gestión y control de diversas plataformas Arduino, ya sea mediante RS-232 o Bluetooth [Qian & Cheng, 2016]. La interfaz fue diseñada con la intención de facilitar la programación y ofrecer una alternativa a Arduino IDE (Ambiente de desarrollador integrado utilizado para programar las placas). Arduino IDE también ha sido objeto de atención para la comunidad científica. Se diseñó una herramienta para Arduino IDE para facilitar el aprendizaje del lenguaje de programación de las SBM Arduino [Torroja, Lopez, Portilla, & Riesgo, 2016]. Esta herramienta permite depurar el código ejecutándolo línea por línea. Esta característica sirve sobre todo a usuarios principiantes o aficionados del prototipado de sistemas mecatrónicos.

Por otro lado, se ha diseñado un sistema embebido en automóviles para tratar de reducir el riesgo de accidentes automovilísticos [Seelam & Lakshmi, 2017]. Entre sus características se encuentran el control de velocidad (en puentes, carreteras, zonas escolares y ciudades), control del claxon (en zonas en donde está prohibido su uso) y un detector de alcohol para la identificación de conductores en estado de ebriedad. Para validar sus resultados, diseñaron un robot móvil diferencial con distintos sensores y actuadores e integraron el sistema propuesto en él, emularon distintas situaciones y demostraron el funcionamiento de su sistema.

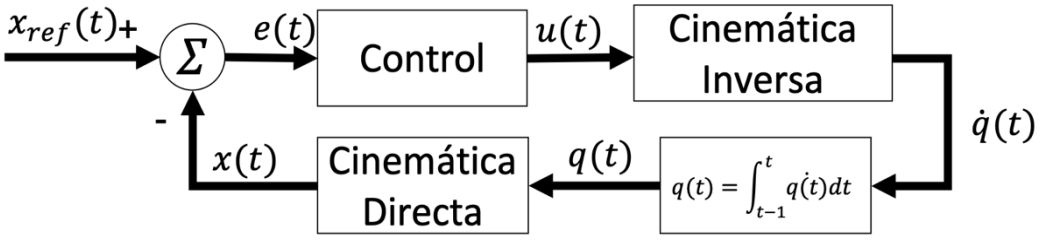
En el presente artículo se propone una alternativa de libre acceso para el control de posición y/o orientación de robots manipuladores de n-grados de libertad mediante el uso de sistemas embebidos para uso didáctico en clase o para aficionados, facilitando el aprendizaje de sistemas de control de manipuladores robóticos. En la sección 2 se muestra la metodología utilizada, (modelo cinemático del robot, descripción de las funciones a incluir en la librería y diseño de la clase y sus métodos). En la sección 3 se muestran los resultados experimentales de un control de posición de un robot manipulador de 3 grados de libertad de revoluta (RRR) utilizando un Arduino MEGA y la librería creada comprobando su eficacia en un

experimento de control de posición. Por último, en la sección 4, se presentan las discusiones y conclusiones.

La librería fue diseñada de forma modular, es decir, que puede ser modificada para adecuarse a cualquier tipo de robot manipulador de revoluta. Si se desea utilizar algún grado de libertad prismático, se deben hacer cambios en el cálculo del Jacobiano. Esta librería ayudará a los usuarios inexpertos a poder prototipar sistemas mecatrónicos con facilidad para resolver problemas específicos.

2. Métodos

Se diseñó y modeló un robot manipulador de 3 grados de libertad de revoluta (figura 1), el cual fue el objeto de estudio para determinar las funciones necesarias para el diseño de un esquema de control de posición y/o orientación de un robot manipulador de n-grados de libertad en n-dimensiones a lazo cerrado.



Fuente: Elaboración propia

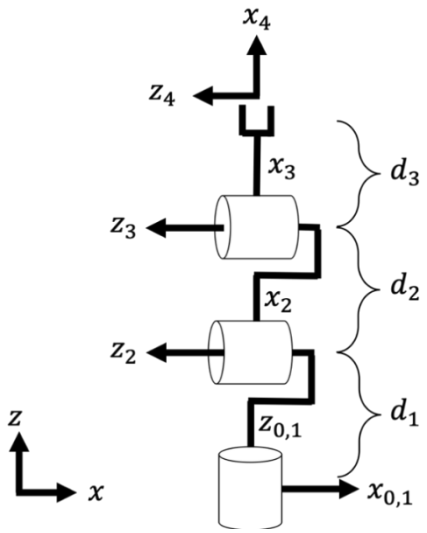
Figura 1 Esquema básico de control de un robot manipulador.

Se observa que los bloques necesarios para poder controlar un robot manipulador son: Cinemática directa, Cinemática inversa y Control.

La figura 2 muestra la representación de los marcos coordenados del robot de 3 grados de libertad, donde $d_i = 1$: Longitud del i-ésimo eslabón.

La cinemática directa del robot se obtuvo mediante una serie de transformaciones geométricas desde el marco coordenado inercial (base del manipulador) hasta el marco coordenado del efector final utilizando una convención propia:

- La i-esima coordenada generalizada se mueve sobre él z_i.
- El eje x_i está alineado con el eslabón que une dos articulaciones.



Fuente: Elaboración propia

Figura 2 Representación de los marcos coordenados del robot de 3 grados de libertad.

Utilizando esta convención, se procedió a calcular las transformaciones del marco coordenado i al marco coordenado $i+1$ hasta $i=n$, ecuaciones 1, 2, 3 y 4.

$$A_0^1 = Rz(q_0) \tag{1}$$

$$A_1^2 = Ry\left(-\frac{\pi}{2}\right) Tx(d_1) Rz(q_1) \tag{2}$$

$$A_2^3 = Tx(d_2) Rz(q_2) \tag{3}$$

$$A_3^4 = Tx(d_3) \tag{4}$$

Donde:

- A_i^{i+1}: Matriz de transformación homogénea entre el marco coordenado i y el marco coordenado $i+1$.
- $R_\alpha(\beta)$: Matriz de transformación homogénea de rotación de β-radianes sobre el eje α.
- $T_a(b)$: Matriz de transformación homogénea de traslación b-unidades sobre el eje a.

Utilizando las ecuaciones 1 a la 4, se describe la transformación del marco coordenado inercial hacia el marco coordenado del efector final, ecuación 5.

$$A_0^4 = A_0^1 A_1^2 A_2^3 A_3^4 \tag{5}$$

Dada la matriz bloques que se muestra en la ecuación 6, se puede obtener la pose o vector de estados del efector final con respecto al marco coordenado inercial considerando el vector de traslación final (t) y la matriz de rotación final (R).

$$A_{4x4} = \begin{bmatrix} R_{3x3} & t_{3x1} \\ 0_{1x3} & 1_{1x1} \end{bmatrix} \tag{6}$$

Para poder determinar la pose, es necesario convertir la matriz de rotación a una representación de orientación vectorial. Se eligió convertir la matriz de rotación a ángulos de Euler con la convención XYZ o roll-pitch-yaw, ecuación 7.

$$\left(R = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} \right) \leftrightarrow \left(\begin{bmatrix} roll \\ pitch \\ yaw \end{bmatrix} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \right)$$

Donde:

$$R = Rx(\phi)Ry(\theta)Rz(\psi) \tag{7.1}$$

$$r_{00} = \mathrm{c}_\theta c_\psi \tag{7.2}$$

$$r_{01} = -c_\theta s_\psi \tag{7.3}$$

$$r_{02} = s_\theta \tag{7.4}$$

$$r_{10} = c_\phi s_\psi + s_\phi s_\theta c_\psi \tag{7.5}$$

$$r_{11} = c_\phi c_\psi - s_\phi s_\theta s_\psi \tag{7.6}$$

$$r_{12} = -s_\phi c_\theta \tag{7.7}$$

$$r_{20} = s_\phi s_\psi - c_\phi s_\theta c_\psi \tag{7.8}$$

$$r_{21} = s_\phi c_\psi + c_\phi s_\theta s_\psi \tag{7.9}$$

$$r_{22} = c_\phi c_\theta \tag{7.10}$$

Donde $c_\alpha = \cos(\alpha)$, $s_\alpha = sen(\alpha)$ y r_{ij} es el elemento en el renglón i de la columna j de la matriz de rotación resultante de una rotación pura sobre el eje x, después sobre el eje y y por último sobre el eje z.

Se observa que, de la ecuación 7.4 se puede despejar θ; de la división de las ecuaciones 7.7 y 7.10 se puede despejar ϕ y, por último, de la división de las ecuaciones 7.3 y 7.2 despejar ψ.

La convención de ángulos de Euler es singular en $\theta = \pm\frac{\pi}{2}$. Considerando esto, los ángulos de Euler se definen en las ecuaciones 8, 9 y 10.

$$\phi = \begin{cases} 0 \; si \; r_{02} = 1 \\ 0 \; si \; r_{02} = -1 \\ atan2(-r_{12}, r_{22}) \; otro \end{cases} \tag{8}$$

$$\theta = \begin{cases} \frac{\pi}{2} \; si \; r_{02} = 1 \\ -\frac{\pi}{2} \; si \; r_{02} = -1 \\ \sin^{-1}(r_{02}) \; otro \end{cases} \tag{9}$$

$$\psi = \begin{cases} atan2(r_{10}, r_{11}) \; si \; r_{02} = 1 \\ atan2(r_{10}, r_{11}) \; si \; r_{02} = -1 \\ atan2(-r_{01}, r_{00}) \; otro \end{cases} \tag{10}$$

Utilizando las ecuaciones 6 y 7 se calcula la pose, ecuación 11.

$$pose = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} \tag{11}$$

Una vez determinado el modelo cinemático del robot, se determinó la estructura de la librería. Se utilizó el paradigma de orientación a objetos para facilitar la reusabilidad del código. La clase tiene los métodos descritos en la tabla 1, en la misma tabla se indican los tipos de datos de entrada y salida de cada función.

Los algoritmos mencionados en tabla 1, son:

- Algoritmo 1: Cálculo del producto cruz entre dos vectores:

  función producto cruz: v_3 ***= cross (vector*** v_1***, vector,*** v_2***)***

  vector $v_3 = \begin{bmatrix} v_{1_1}v_{2_2} - v_{1_2}v_{2_1} \\ v_{1_0}v_{2_2} - v_{1_2}v_{2_0} \\ v_{1_0}v_{2_1} - v_{1_1}v_{2_0} \end{bmatrix}$

  return v_3

Tabla 1 Funciones propuestas a incluir como métodos.

Función	Entradas	Salidas
Producto cruz (Algoritmo 1)	Dos vectores de tamaño 3x1	Vector de tamaño 3x1
Rotación en el eje x (Algoritmo 2)	Escalar	Matriz de tamaño 4x4
Rotación en el eje y (Algoritmo 3)		
Rotación en el eje z (Algoritmo 4)		
Traslación en el eje x (Algoritmo 5)		
Traslación en el eje y (Algoritmo 6)		
Traslación en el eje z (Algoritmo 7)		
Conversión de matriz de rotación a ángulos de Euler(Algoritmo 8)	Matriz de tamaño 4x4	Vector de tamaño 3x1
Cinemática directa (Algoritmo 9)	Vector de tamaño nx1	Estructura con: • Matriz de tamaño 4x4 • Matriz de tamaño 3x3 • Dos vectores de tamaño 3x1 • Vector de tamaño 6x1 • Matriz de tamaño 6xn • Dos matrices de tamaño 3xn
Pseudo-inversa de Jacobiano (Algoritmo 10)	Matriz de tamaño 6xn	Matriz de tamaño nx6
Pseudo-inversa de Jacobiano de velocidades lineales o angulares (Algoritmo 11)	Matriz de tamaño 3xn	Matriz de tamaño nx3
Control en 3 dimensiones (Algoritmo 12)	Tres vectores de tamaño 3x1 Tres matrices de tamaño 3x3	Vector de tamaño 3x1
Control en 6 dimensiones (Algoritmo 13)	Tres vectores de tamaño 6x1 Tres matrices de tamaño 6x3	Vector de tamaño 6x1

Fuente: Elaboración propia.

- Algoritmo 2: Cálculo de matriz de transformación homogénea de rotación en x un ángulo α:

  función matriz de rotación en el eje x: $R =$ ***Rx (escalar*** a***)***

  matriz $R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(a) & -\sin(a) & 0 \\ 0 & \sin(a) & \cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

  return R

- Algoritmo 3: Cálculo de matriz de transformación homogénea de rotación en y un ángulo α:

  función *matriz de rotación en el eje y:* $R =$ *Ry (****escalar*** a*)*

  matriz $R = \begin{bmatrix} \cos(a) & 0 & \sin(a) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(a) & 0 & \cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

return R

- Algoritmo 4: Cálculo de matriz de transformación homogénea de rotación en z un ángulo α:

función *matriz de rotación en el eje z:* $R = Rz$ (***escalar*** a)

matriz $R = \begin{bmatrix} \cos(a) & -\sin(a) & 0 & 0 \\ \sin(a) & \cos(a) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

return R

- Algoritmo 5: Cálculo de matriz de transformación homogénea de traslación en x una distancia α:

función *matriz de traslación en el eje x:* $T = Tx$ (***escalar*** a)

matriz $T = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

return T

- Algoritmo 6: Cálculo de matriz de transformación homogénea de traslación en y una distancia a:

función *matriz de traslación en el eje y:* $T = Ty$ (***escalar*** a)

matriz $T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & a \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

return T

- Algoritmo 7: Cálculo de matriz de transformación homogénea de traslación en z una distancia a:

función *matriz de traslación en el eje z:* $T = Tz$ (***escalar*** a)

matriz $T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & a \\ 0 & 0 & 0 & 1 \end{bmatrix}$

return T

- Algoritmo 8: Cálculo para convertir una matriz de rotación a ángulos de Euler usando la convención XYZ:

función *conversión de matriz de rotación a ángulos de Euler:* $euler =$ *rotm2euler (**matriz** R)*

vector $euler$*: vector de orientación con la convención XYZ (ver ecuaciones 8, 9 y 10)*

return $euler$

- Algoritmo 9: Cálculo de los elementos pertinentes a la cinemática directa:

función *Cinemática directa:* $fk =$ *forward_kinematics (**vector** q)*

matriz $A_0^i = A_0^1 \dots A_0^i \forall i$

vector o_i*: traslación de la matriz de transformación* A_0^i

vector z_i*: tercera columna de la rotación de la matriz de transformación* A_0^i

estructura fk

matriz $fk.A$*: cinemática directa*

matriz $fk.R$*: rotación final*

vector $fk.traslation$*: traslación final*

vector $fk.orientation$*: orientación final en ángulos de Euler*

vector $fk.pose$*: pose del efector final*

matriz $fk.J = \begin{bmatrix} (o_n - o_i) \times z_i \\ z_i \end{bmatrix} \forall i$*: Jacobiano para articulaciones de revoluta, donde* o_i *es el vector de traslación de la* i*-ésima articulación.*

matriz $fk.J_v = [(o_n - o_i) \times z_i] \forall i$*: Jacobiano de velocidades lineales articulaciones de revoluta*

matriz $fk.J_\omega = [z_i] \forall i$*: Jacobiano de velocidades angulares para articulaciones de revoluta.*

return fk

- Algoritmo 10: Cálculo la pseudo-inversa con regularización por la derecha de la matriz Jacobiana.

función *pseduo-inversa del Jacobiano:* $J^+ =$ *pinvJ (**matriz** J)*

matriz $kI = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$

matriz $J^+ = J^T (J\,J^T + kI)^{-1}$

return J^+

- Algoritmo 11: Cálculo pseudo-inversa con regularización por la izquierda de la matriz Jacobiana de velocidades lineales o angulares.

  función *pseduo-inversa del Jacobiano de velocidades lineales/angulares:* $J_{v/w}^{+} = pinvJv$ (***matriz*** J)

  matriz $kI = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$

  matriz $J^+ = (J^T J + kI)^{-1} J^T$

  return J^+

- Algoritmo 12: Control PID para un sistema en 3 dimensiones.

  función *control en 3 dimensiones:* $u = control3$ (***vector*** e, ***vector*** ie, ***vector*** de, ***matriz*** K_P, ***matriz*** K_i, ***matriz*** K_d)

  vector $u = K_p e + K_i ie + K_d de$

  return u

- Algoritmo 13: Control PID para un sistema en 6 dimensiones.

  función *control en 6 dimensiones:* $u = control6$ (***vector*** e, ***vector*** ie, ***vector*** de, ***matriz*** K_P, ***matriz*** K_i, ***matriz*** K_d)

  vector $u = K_p e + K_i ie + K_d de$

  return u

- Algoritmo 14: Control de posición de un robot manipulador.

  función *loop ()*

  $fk_t = forward\_kinematics(q)$

  $x_t = fk.traslation$

  $e_t = x_{ref} - x_t$

  $ie_t = (e_t + e_{t-1})h$

  $de_t = (e_t - e_{t-1})/h$

  $K_p = 5$

  $K_i = 0.01$

  $K_d = 0.01$

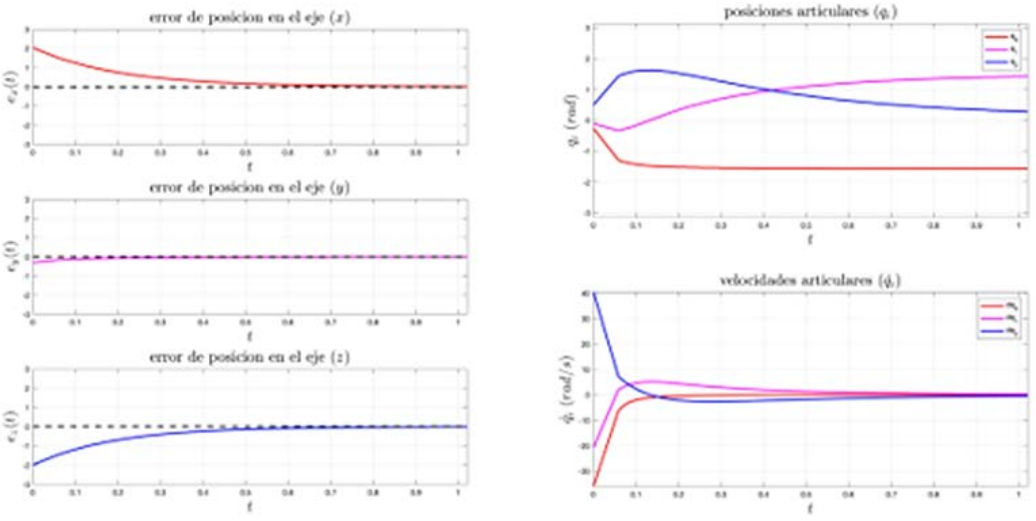
$$u_t = control3(e_t, ie_t, de_t, K_p, K_i, K_d)$$
$$J_v^+ = pinvJv(fx.J_v)$$
$$\dot{q}_t = J_v^+ u_t$$
$$q_{t+1} = q_t + \dot{q}_t h$$

3. Resultados

Utilizando la clase y métodos diseñados junto con el modelo cinemático del robot manipulador, se diseñó un experimento de control de posición siguiendo la figura 1 utilizando un Arduino MEGA debido a su practicidad, accesibilidad. La programación de Arduino se divide en dos partes: La función **setup ()** y la función **loop ()**.

Se definieron las variables globales para almacenar el paso de integración ($h = 1/100$), las coordenadas generalizadas ($q = [0,0,0]^T$), la referencia de posición ($x_{ref} = [2,0,1]^T$), las coordenadas generalizadas y error anterior para la integración numérica ($q_0 = [0,0,0]^T$ y $e_o = [0,0,0]^T$ respectivamente). En la función de **setup ()** solo se inicializó el puerto serial a 115,200 Baudios para el monitoreo del sistema.

Por último, la función **loop ()** quedo definida en el algoritmo 14. Se obtuvieron los resultados mostrados en la figura 3.



Fuente: Elaboración propia

Figura 3 Gráfica de la evolución del error de posición en los tres ejes con respecto al tiempo y evolución de la posición y velocidad articular con respecto al tiempo.

Se puede observar que el error de posición en los tres ejes converge en 0, indicando que la posición euclidiana del efector final ha llegado a la referencia, es decir $x_t \rightarrow x_{ref}$. Las posiciones articulares comienzan en la posición inicial y terminan en la configuración necesaria (mas no única) para converger en la referencia. De la misma forma, las velocidades articulares convergen a cero una vez terminado el experimento. Es importante observar que al utilizar un controlador PID para estabilizar la posición del efector final en la referencia, la velocidad inicial ($v_{t=0}$) es 0, mientras que en el primer instante ($v_{t>0}$) es la velocidad máxima permitida por el sistema, convergiendo de forma exponencialmente asintótica. Las ganancias del controlador fueron establecidas arbitrariamente y sintonizadas a prueba y error para conseguir la convergencia deseada, sin embargo, estas pueden ser modificadas sin inconveniente alguno.

4. Discusión

La librería puede ser utilizada para simular y controlar manipuladores robóticos. Aunque para lograr esto hace falta conectar los servomotores al sistema y mapear la posición articular de salida del algoritmo a la posición absoluta del servomotor.

Este tipo de librerías pueden ser utilizadas en la enseñanza de materias afines a robótica o mecatrónica, debido a la facilidad de su uso y su modularidad. Ofrece funciones con documentación para poder controlar tanto la orientación como la posición del efector final de cualquier robot manipulador. Aunque para esto, es necesario realizar algunos cambios en la librería. Por ejemplo, si el manipulador tiene articulaciones prismáticas, se debe de cambiar el Jacobiano acorde, si la cinemática es diferente, habrá que cambiar la cinemática en la función correspondiente, etc. Actualmente, la librería se utiliza en clases de Robótica, Diseño Mecatrónico y Tesis de la Universidad Autónoma de Ciudad Juárez en la carrera de Ingeniería Mecatrónica, y les sirve a los alumnos para llevar a la práctica lo aprendido en clase. Uno de los planes a futuro es subirla al catálogo de librerías de Arduino y portarla a distintas plataformas como Raspberry Pi, FPGA´s; y en distintos lenguajes como C++, Python, Simulink, Matlab, etc.

5. Referencias y bibliografía

[1] Cvjetkovic, V. M., & Matijevic, M. (2016). Overview of architectures with arduino boards as building blocks for data acquisition and control systems. International Journal of Online Engineering, 12(7), 10–17. https://doi.org/10.3991/ ijoe.v12i07.5818.

[2] Fernández, J., Gemin, W., Rivera, R., Revuelta, M., Kuzman, M., & Hidalgo, R. (2016). Digital filter design with Arduino DUE and Matlab. 2015 16th Workshop on Information Processing and Control, RPIC 2015. https://doi.org/10.1109 /RPIC.2015.7497060.

[3] Kuang, Y. (2014). Communication between PLC and arduino based on modbus protocol. Proceedings - 2014 4th International Conference on Instrumentation and Measurement, Computer, Communication and Control, IMCCC 2014, 370–373. https://doi.org/10.1109/IMCCC.2014.83.

[4] Qian, B., & Cheng, H. H. (2016). ChDuino: A real-time controller for Arduino. MESA 2016 - 12th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications - Conference Proceedings, 1–6. https://doi.org/10.1109/MESA.2016.7587191.

[5] Seelam, K., & Lakshmi, C. J. (2017). An Arduino based embedded system in passenger car for road safety. Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2017, (Icicct), 268–271. https://doi.org/10.1109/ICICCT.2017.7975201.

[6] Tipsuwanporn, V., Numsomran, A., Samaimak, S., & Harnnarong, S. (2014). Development of redundant bus library for arduino to apply in SCADA system. International Conference on Control, Automation and Systems, (lCCAS), 42–46. https://doi.org/10.1109/ICCAS.2014.6987955.

[7] Torroja, Y., Lopez, A., Portilla, J., & Riesgo, T. (2016). A Serial Port Based Debugging Tool To Improve Learning With Arduino. 2015 Conference on Design of Circuits and Integrated Systems, DCIS 2015, 0–3. https://doi.org/10.1109/DCIS.2015. 7388612.