Actualización integral basada en Arduino y ROS de un manipulador robótico Scorbot ER-VII

León Herrera
Laboratorio de Robótica y Sistemas Embebidos
Departamento de Computación
Universidad de Buenos Aires, CABA
Email: lherrera@dc.uba.ar

Facundo Pessag
Laboratorio de Robótica y Sistemas Embebidos
Departamento de Computación
Universidad de Buenos Aires, CABA
Email: fpessacg@dc.uba.ar

Pablo De Cristóforis
Laboratorio de Robótica y Sistemas Embebidos
Departamento de Computación
Universidad de Buenos Aires, CABA
Email: pdecris@dc.uba.ar

***Resumen*—En este trabajo presentamos el conjunto de acciones llevadas a cabo en el manipulador robótico Scorbot ER-VII para reemplazar tanto el hardware como el software original del mismo. Esta actualización integral está basada en microcontroladores Arduino y ROS (Robot Operating System). También se presenta un modelo cinemático para el control del brazo y un modelo en formato URDF obtenido a partir de Xacro que permiten la visualización en Rviz y la simulación en Gazebo. El código del proyecto fue publicado en: https://github.com/lrse/scorbot-vii para beneficio de la comunidad científica.**

I. Introducción

Los manipuladores robóticos han ganado gran aceptación en la industria gracias a su capacidad de llevar a cabo un número variado de tareas de manera repetitiva con rapidez y precisión, y la posibilidad de adaptarse a diversos entornos de trabajo. En todo manipulador se pueden distinguir cuatro componentes principales que lo conforman: partes electromecánicas, hardware, firmware de bajo nivel y software de alto nivel. Es la suma de ellos, lo que hace que los manipuladores puedan operar como un sistema autónomo, versátil, y robusto para desarrollar múltiples propósitos.

Los manipuladores robóticos Scorbot fueron lanzados al mercado durante la década '90 por la empresa Eshed Robotec, llamada ahora Intelinek Inc. [6]. Estos robots fueron adquiridos en la Argentina por varias universidades para tareas de docencia e investigación. Hoy en día tanto el hardware como el software de control de estos manipuladores se encuentran completamente desactualizados respecto de las prestaciones que ofrecen los sistemas actuales. Sin embargo, las partes electromecánicas siguen funcionando correctamente en la mayoría de los casos, con lo cual tiene sentido plantearse la reingeniería de los mismos para volver a ponerlos operativos a bajo costo.

En este trabajo presentamos el conjunto de acciones que fueron llevadas a cabo para actualizar de manera integral el hardware y software de un Scorbot ER-VII, a partir del uso de placas Arduino [9] y ROS (Robot Operating System) [4]. La adopción de estas tecnologías permitió la incorporación de un entorno de simulación del manipulador en Gazebo [7] y visualización del estado en Rviz, a partir de un modelo hecho en Xacro [12] que luego se pasa a formato URDF.

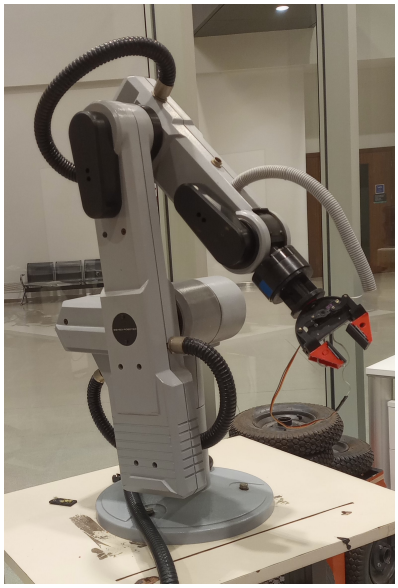


Figura 1. Foto del Scorbot ER-VII con el efector final original reemplazado por una pinza paralela actuada por un servo-motor.

El Scorbot ER-VII tiene 4 articulaciones aparentes (base, hombro, codo y muñeca) y 1 no aparente (la rotación en la muñeca) que le confieren sus grados de libertad. Como efector final posee originalmente una pinza neumática de apertura paralela. Cada articulación es actuada por un motor de corriente continua. La transmisión mecánica entre el giro del motor y la rotación de cada articulación se realiza mediante un sistema de polea con correa dentada y una caja reductora. Cada uno de los motores tiene acoplado a su eje un encoder óptico incremental. Sobre cada articulación se encuentran los micro-switches que definen los finales de carrera y un home-switch para cada articulación. En la Figura 1 podemos ver una foto del Scorbot ER-VII.

A continuación presentamos una explicación detallada de las modificaciones realizadas en cada uno de los componentes del manipulador: Hardware, Firmware, Software. Luego presentamos el modelo cinemático y por último el entorno de visualización y simulación en RViz y Gazebo respectivamente.

II. HARDWARE

El hardware que controla el movimiento del brazo fue reemplazado por un conjunto de microcontroladores Arduino [9] conectados en una placa de prototipado (ver Figura 2). Se siguen utilizando las fuentes de energía originales para alimentar tanto a las placas Arduinos como a los motores. Por otro lado, el efector final original fue reemplazado por una pinza servo-motor a la cual se le incorporaron dos sensores de tacto capacitivos.

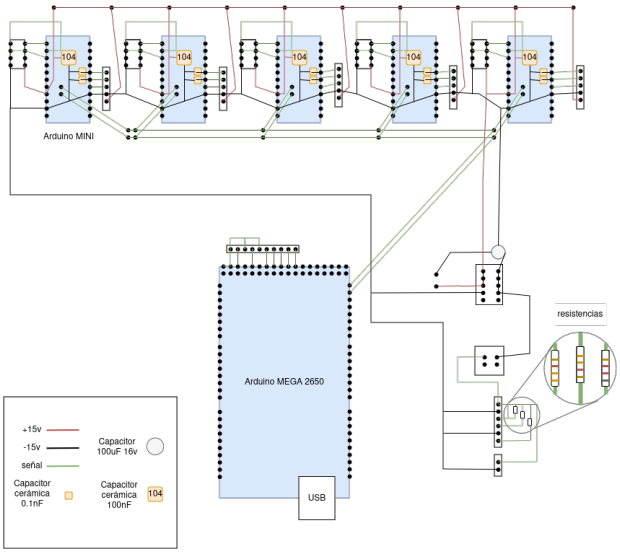


Figura 2. Diagrama del circuito de control del SCORBOT ER-VII renovado. Solo se incluyen las conexiones en la placa de prototipado.

Específicamente, se utilizaron 5 Arduino Pro MINIs, 1 Arduino MEGA 2560, 1 Arduino UNO, 2 sensores de tacto capacitivos TTP223B y 5 puentes H Arduino IBT-2 para reemplazar el hardware de control original del manipulador.

En la Figura 2 podemos ver un diagrama de la placa de prototipado con los Arduinos Pro MINIs y el Arduino MEGA 2560. Cada Arduino Pro MINI lee la señal del encoder correspondiente a cada una de las 5 articulaciones del manipulador a través de un conector de 5 pines y envía una señal PWM al puente H que controla el movimiento del motor de corriente continua mediante un conector de 8 pines. Como se puede observar en la Figura 2, la comunicación entre el Arduino MEGA y los MINIs se hace por un canal de dos pines y el protocolo de comunicación I^2C. En la Figura 3 también podemos ver como está hecha la conexión entre los Arduinos MINIs y sus respectivos puente-H y encoders de la articulación correspondiente.

La pinza neumática original fue reemplazada por una que se acciona mediante un servo-motor, a la cual se le incorporaron sensores de tacto capacitivos para la detección de objetos en el agarre. Para esto, diseñamos dos piezas en TinkerCad [10] y Ultimaker Cura [11] de manera de poder acomodar los sensores de tacto por detrás de una lamina de goma que sirve como material sensible al entrar en contacto con un objeto en el momento del agarre. El modelo 3D de las piezas puede

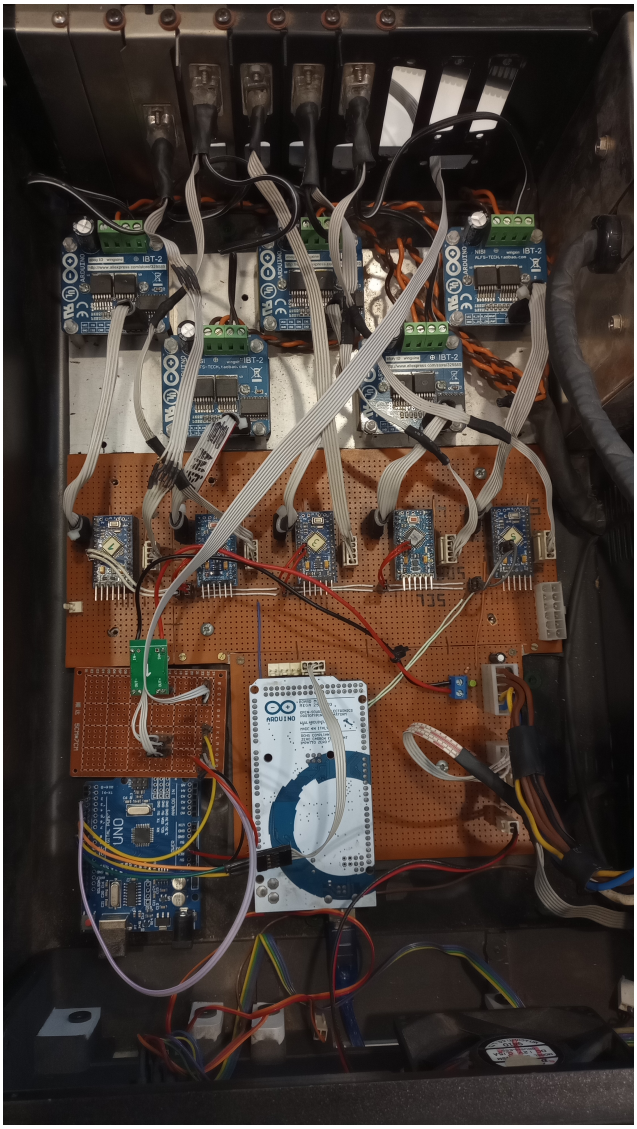
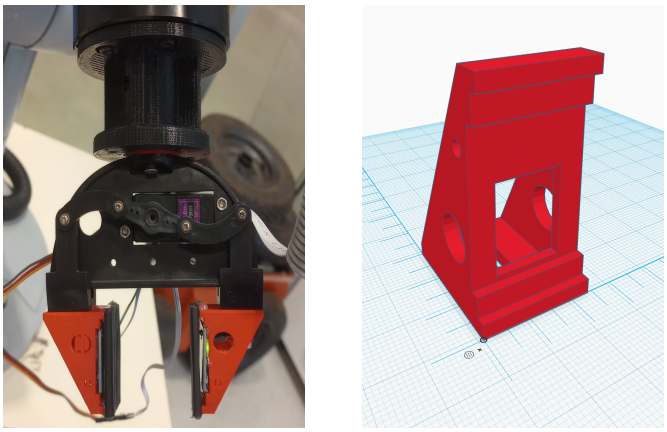


Figura 3. Foto de las placas Arduinos y puentes H que reemplazan al hardware original de control del Scorbot ER-VII. Pueden observarse las conexiones entre los Arduino Pro MINIs, los puentes H, las fuentes de energía y los conectores que van al manipulador.

observarse en la Figura 4b, donde se ven los peldaños que ayudan a separar el sensor de la lamina de goma y el agujero por donde salen los pines del sensor a la sección de atrás para conectarse a los cables de alimentación y señal. En la Figura 4a se puede ver la nueva pinza montada en el Scorbot ER-VII como efector final.

Los sensores de tacto y el servo-motor que actúa la pinza están conectados a la fuente de baja tensión y al Arduino UNO, que lee la señal de los sensores y envía la señal al servo-motor para controlar el ángulo de apertura de la pinza (ver Figura 3). El Arduino UNO se conecta a 4 pines al Arduino MEGA quien a su vez le indica cuando cerrar y abrir la pinza y reporta cuando sensa que un objeto fue agarrado. El Arduino UNO utiliza uno de sus pines de interrupción para cada sensor y otro para pasarle un ángulo de apertura al servo-motor. El Arduino



(a) (b)

Figura 4. A la izquierda (a): la nueva pinza montada al SCORBOT ER-VII con los dos sensores capacitivos. A la derecha (b): el modelo 3D de la pieza diseñada para montar los sensores y unas laminas de goma que ayudan a sensar

MEGA se conecta a través de un cable USB con una PC que corre ROS y un nodo ROSserial para poder recibir comandos y reportar el estado de de cada articulación del manipulador.

III. Firmware

En el Arduino UNO se ejecuta una maquina de estados simple que mantiene el estado de la pinza y dependiendo del mismo comanda el cierre o apertura, al llegarle la orden correspondientes del Arduino MEGA, a quien a su vez le reporta su estado cada vez que ocurre una variación. Los estados de la pinza son: abierta, cerrada, abriendo, cerrando, esperando señal de sensor de tacto izquierdo, esperando señal del sensor de tacto derecho, agarrando objeto.

Cuando se desea agarrar un objeto con la pinza, el Arduino UNO espera recibir la señal de alguno de los dos sensores de tacto y cuando esto ocurre, cambia a otro estado donde sigue cerrando mientras espera la señal contacto del otro sensor. Cuando ésta llega, asume que terminó de agarrar un objeto completamente y la pinza se cierra 5 grados más para luego parar y terminal el comportamiento de cierre.

En cada Arduino Pro MINI se ejecuta el código que se encarga de crear una señal PWM a partir de un control PID a lazo cerrado con las lecturas de los encoders de la articulación correspondiente. Además, cada Arduino Pro MINI debe comunicarse con el Arduino MEGA para informar el ángulo de la articulación que controla.

El Arduino MEGA ejecuta una biblioteca de ROS para Arduino que le permite recibir y publicar mensajes por su puerto serial estableciendo una conexión con el nodo *ROSSerial* de la computadora que controla el manipulador. El código que se ejecuta en el Arduino MEGA informa cada determinado tiempo los ángulos de cada articulación del brazo y se comunica con el Arduino MINI correspondiente al recibir un mensaje para mover alguna articulación o con el Arduino UNO para cerrar o abrir la pinza. En caso de interactuar con el Arduino UNO, se usan 4 pines para la comunicación: uno para saber si la pinza está abierta o cerrada, otro para saber si está agarrando un objeto o no, un tercero para setear si se quiere abrir o cerrar la pinza, y el último para ordenar cuando tiene que cerrarse o abrirse la pinza.

Para recibir y enviar mensajes entre el Arduino MEGA y los Arduinos MINIs, se usa una comunicación de dos pines y el protocolo I^2C. Se utilizan varias funciones para cada interacción posible entre el Arduino MEGA y cada Arduino MINI, entre las que vale la pena mencionar:

- Definir una velocidad para una articulación durante un determinado lapso de tiempo.
- Definir un ángulo deseado para una articulación.
- Leer el valor del encoder de la articulación.
- Setear los parámetros del Ardunio MINI (entre ellos los coeficientes del controlador PID)

.

Para la comunicación entre el Arduino MEGA y la computadora que se realiza por el puerto serie se definen los siguientes mensajes:

- Velocidad y dirección de movimiento para las articulaciones.
- Parada total de todos movimientos del brazo.
- Abrir o cerrar la pinza.
- Mover cada articulación a un ángulo especificado.
- Mover cada articulación a una posición de inicio (*home*).

.

IV. Software de alto nivel

En la computadora que controla el Scorbot ER-VII se ejecuta un sistema que implementa un conjunto de nodos de ROS con los que se controla el brazo real y/o se realiza la simulación en Gazebo. En la figura 5 se pueden observar los nodos de ROS activos durante la ejecución del sistema.

Como puede observarse, para leer los comandos de entrada se utilizan los nodos *ros_joy* [14] y *ros_keyboard* [15] para el joystick y el teclado respectivamente. Luego se utiliza el nodo *universal_teleop* [16] que traduce los comandos a *eventos* y *controles* para el nodo *scorbot*, el cual a su vez comunica estos eventos al brazo a través de nodo *Scorbot_serial*.

Los eventos del nodo *universal_teleop* se usan para comunicar el input de los botones del joystick y las teclas del teclado, y se implementan mediante un *string* detallando la acción a realizar y un valor booleano que nos indica si hay que finalizar o iniciar la acción. Las acciones que están definidas hasta ahora son: activar el modo seguro del manipulador (que setea una velocidad baja para todas las articulaciones), ir a una posición inicial (*home*), cerrar o abrir la pinza, mover hacía un lado u otro el codo, muñeca o rotación del brazo o parar en seco todos los movimientos. Un control es como un evento pero en lugar de un valor booleano tiene un valor entero asociado, esto se usa para comunicar el input de los controles analógicos del joystick y en nuestra configuración del joystick lo usamos para mover las articulaciones de la base y el hombro del brazo. El nodo *scorbot* toma estos eventos y controles y los interpreta para publicar en varios *topics* de

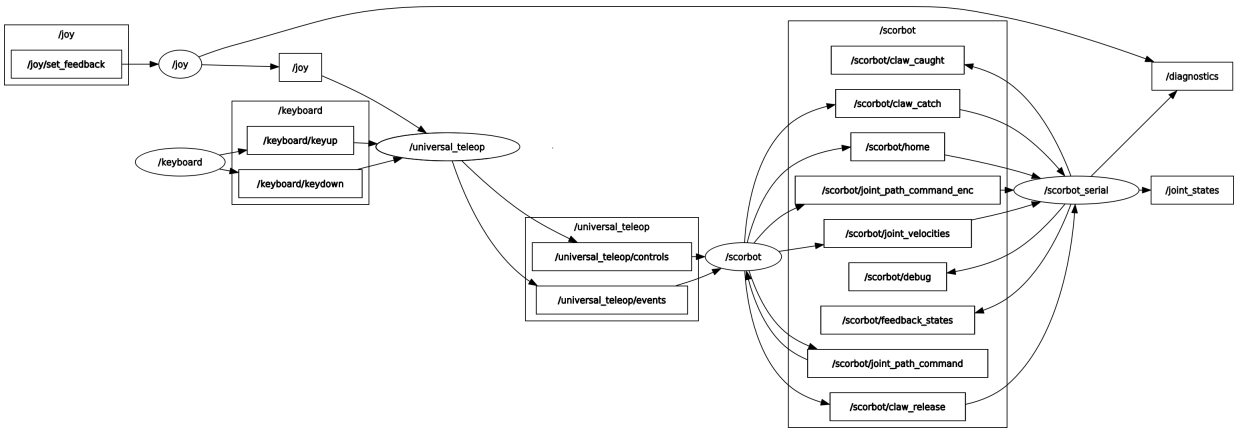


Figura 5. Esquema de los nodos de ROS activos al ejecutar el sistema de control del SCORBOT ER-VII en una PC.

manera que el nodo *ROSserial* pueda leerlos, entre los que podemos mencionar:

- */claw_catch* y */claw_release* se usan para cerrar y abrir la pinza respectivamente,
- */home* se usan para ir a la posición donde se inició el SCORBOT,
- *joint_path_command_enc* se usa para comunicar una posición deseada para cada articulación del brazo,
- *joint_path_command* es el conjunto de posiciones que se le comunican al nodo *scorbot* para pasarle al Arduino MEGA,
- */joint_velocities* es un arreglo de velocidades para cada motor del brazo,
- */claw_caught* se usa para comunicar si la pinza agarró un objeto,
- */debug* se usa para mensajes de control,
- *feedback_states* comunica cuando se llega a la posición deseada anterior a la que se debía mover el brazo,
- *joint_states* reporta el ángulo actual en radianes de las articulaciones.

V. Modelo cinemático

El modelo cinemático que desarrollamos para el SCORBOT ER-VII está basado en los modelos presentados por Predescu y Predescu [1], y por Predescu y Stroe [2]. A este modelo se le hicieron algunas modificaciones para que funcionaran mejor con la simulación de Gazebo [7]:

- Usamos como origen de coordenadas la base del Scorbot en vez de su primer articulación.
- Usamos rotaciones en sentido horario en vez de anti-horario.
- No utilizamos un sistema paralelo para la rotación de la 5ta articulación (en la Figura 6 la rotación θ_5, el $roll$ del efector), si no que esa rotación se agrega a la última matriz de transformaciones del sistema.

Parametrizamos el brazo de la forma vista en la Figura 6, donde θ_i es la rotación de la articulación i, y a_i y d_i son la medida de la sección i del robot. De esta forma conseguimos el modelo que vemos en la figura 7 y la siguiente tabla de

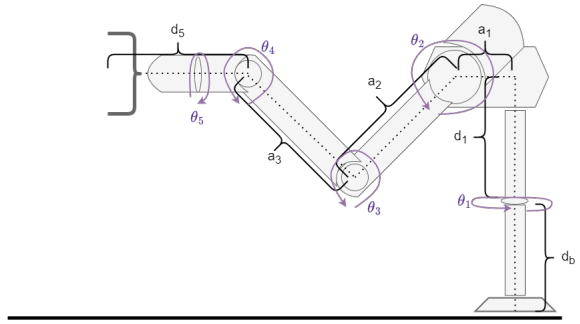


Figura 6. Parámetros y medidas del Scorbot ER-VII usadas para calcular la cinemática directa.

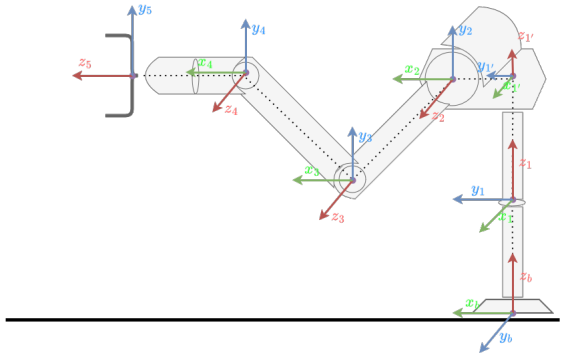


Figura 7. Modelo cinemático para el Scorbot ER-VII usado en este proyecto

parámetros Denavit Hartenberg (D-H), donde θ es la rotación en el eje z, β es la rotación del eje y y γ la rotación del eje x.

Luego, definimos las siguientes abreviaciones:

- $c_k = cos(\sum_{i \in k} \theta_i)$ donde $k \subseteq \{1, 2, 3, 4, 5\}$
- $s_k = sen(\sum_{i \in k} \theta_i)$ donde $k \subseteq \{1, 2, 3, 4, 5\}$
- $c_{\frac{\pi}{2}} = cos(\frac{\pi}{2}) = 0$
- $s_{\frac{\pi}{2}} = sen(\frac{\pi}{2}) = 1$

A partir de este modelo podemos calcular la cinemática directa con las matrices de rotación y traslación correspondientes:

Parámetros de D-H					
Sección i	$d_i(mm)$	$a_i(mm)$	$\theta_i(rad)$	$\beta_i(rad)$	$\gamma_i(rad)$
b	200	0	0	0	0
1	17	0	θ_1	0	0
1'	0	43	0	$\frac{\pi}{2}$	$\frac{\pi}{2}$
2	0	300	θ_2	0	0
3	0	250	θ_3	0	0
4	0	0	θ_4	$\frac{\pi}{2}$	0
5	190	0	θ_5	0	0

Cuadro I
TABLA DE PARÁMETROS DENAVIT HARTENBERG (D-H) DEL MANIPULADOR SCORBOT ER-VII.

T_1^b : rotación $\frac{\pi}{2}$ en z + negamos x para invertirlo + traslación d_b en z

$$T_1^b = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_b \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{T}_1^b = \begin{bmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & db \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$T_{1'}^1$: rotación θ_1 en z + traslación d_1 en z

$$\mathrm{T}_{1'}^1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{T}_{1'}^1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$T_2^{1'}$: rotación $\frac{\pi}{2}$ en x + rotación $\frac{\pi}{2}$ en y + traslación a_1 en x

$$\mathrm{T}_2^{1'} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{T}_2^{1'} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T_3^2 : rotación θ_2 en z + traslación a_2 en x

$$\mathrm{T}_3^2 = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{T}_3^2 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T_4^3 : rotación θ_3 en z + traslación a_3 en x

$$\mathrm{T}_4^3 = \begin{bmatrix} c_3 & -s_3 & 0 & 0 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{T}_4^3 = \begin{bmatrix} c_3 & -s_3 & 0 & a_3c_3 \\ s_3 & c_3 & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T_5^4 : rotación θ_4 en z + rotación $\frac{\pi}{2}$ en y

$$\mathrm{T}_5^4 = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_{\frac{\pi}{2}} & 0 & s_{\frac{\pi}{2}} & 0 \\ 0 & 1 & 0 & 0 \\ -s_{\frac{\pi}{2}} & 0 & c_{\frac{\pi}{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{T}_5^4 = \begin{bmatrix} 0 & -s_4 & c_4 & 0 \\ 0 & c_4 & s_4 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T_e^5 : rotación θ_5 en z + traslación d_5 en z

$$\mathrm{T}_e^5 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{T}_e^5 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Con éstas matrices podemos calcular la cinemática directa del manipulador:

$$T_e^b = T_1^b \times T_{1'}^1 \times T_2^{1'} \times T_3^2 * T_4^3 \times T_5^4 \times T_e^5$$

$$T_e^b = \begin{bmatrix} s_{234}c_1s_5 + s_1c_5 & s_{234}c_1c_5 + s_1s_5 & -c_{234}c_1 & x \\ -s_{234}s_1s_5 - c_1c_5 & -s_{234}s_1c_5 + c_1s_5 & c_{234}s_1 & y \\ c_{234}s_5 & c_{234}c_5 & s_{234} & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

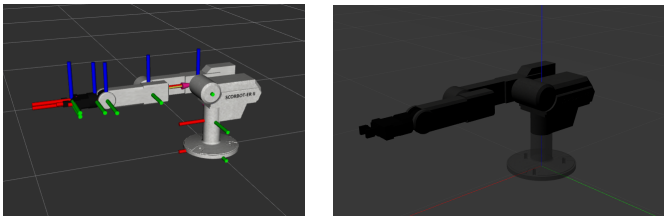
de la cual podemos derivar las siguientes ecuaciones para el punto $(x, y, z)_e$, donde se ubica el efector con los ángulos de articulaciones $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$ con el origen del sistema de coordenadas en la base del robot:

- $x = -cos(\theta_1) * (a_1 + a_2 * cos(\theta_2) + a_3 * cos(\theta_2 + \theta_3) + d_5 * cos(\theta_2 + \theta_3 + \theta_4)$
- $y = sen(\theta_1) * (a_1 + a_2 * cos(\theta_2) + a_3 * cos(\theta_2 + \theta_3) + d_5 * cos(\theta_2 + \theta_3 + \theta_4)$
- $z = d_b + d_1 + a_2 * sen(\theta_2) + a_3 * sen(\theta_2 + \theta_3) + d_5 * sen(\theta_2 + \theta_3 + \theta_4)$

Todos estos calculos fueron realizados en MatLab Online [5].

VI. Simulación en Gazebo y Visualización en Rviz

Uno de los primeros objetivos de la actualizacion integral del Scorbot ER-II fue poder contar con un entorno de simulación y visualización. Gracias a un trabajo del laboratorio de robótica en la Universidad de Sevilla [3], en el que presentan un modelo y una simulación de un Scorbot ER-VII pudimos desarrollar nuestra simulación en Gazebo y visualización en RViz como puede verse en la Figura 8, con algunas modificaciones menores para nuestra conveniencia, como el cambio de la longitud de la pinza para coincidir con la nuestra. Para esto se genera un modelo en formato URDF a partir de un modelo en Xacro que pudimos obtener del mencionado trabajo.



(a) (b)

Figura 8. A la izquierda (a): visualización del Scorbot ER-VII en Rviz y a la derecha (b) simulación en Gazebo.

VII. Conclusiones y Trabajo Futuro

En este trabajo se presenta una re-ingeniería del manipulador Scorbot ER-VII para volver a ponerlo operativo. Para esto se realizó una actualización integral del hardware y software basada en Arduino y ROS. Podemos afirmar que el resultado de este trabajo fue positivo ya que volvimos a poder controlar el manipulador de manera completa. Como trabajo futuro nos proponemos desarrollar un planificador de trayectorias que se ejecute en la computadora del manipulador y comanden los movimientos que permitan que el efector final recorra varios puntos en el espacio de trabajo del robot.

Referencias

[1] L. Predescu, M. Predescu, *PLANNING THE TRAJECTORY OF THE SCORBOT-ER VII ROBOT*, 5th International Conference Çomputational Mechanics and Virtual EngineeringÇOMEC 2013 24-25 October 2013, Braşov, Romania.

[2] Laurentiu Predescu, Ion Stroe, *ON THE KINEMATIC OF THE SCORBOT ER-VII ROBOT*, U.P.B Sci. Bull., Series D, Vol. 77, Iss. 2, 2015.

[3] RTC research group, *Event Based Scorbot ER-VII*.

[4] Stanford Artificial Intelligence Laboratory et al., 2018. *Robotic Operating System*, Available at: https://www.ros.org.

[5] The MathWorks, Inc. (2022). MATLAB version: 9.13.0 (R2022b). Accessed: January 01, 2023. Available: https://www.mathworks.com.

[6] Intelitek incorporated. https://intelitek.com/

[7] gazebosim.org

[8] http://wiki.ros.org/rviz

[9] www.arduino.cc

[10] www.tinkercad.com

[11] https://ultimaker.com/es/software/ultimaker-cura/

[12] http://wiki.ros.org/xacro

[13] http://wiki.ros.org/rosserial

[14] http://wiki.ros.org/joy

[15] http://wiki.ros.org/keyboard

[16] http://wiki.ros.org/universal_teleop