Herramienta para el Control y la Manipulación de un robot Puma real

Hernán Darío Trullo Muñoz

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Ingeniería en Automática Industrial

Popayán, diciembre 2023

Herramienta para el Control y la Manipulación de de un robot Puma real

Monografía presentada como requisito parcial para optar por el título de Ingeniero en Automática Industrial

Hernán Darío Trullo Muñoz

Director. PhD. Óscar Andrés Vivas Albán

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Ingeniería en Automática Industrial

Popayán, diciembre 2023

Nota de aceptación.

Firma del presidente del jurado

Firma del jurado

Popayán, diciembre de 2023

Agradecimientos

En primer lugar, quiero agradecer a mi madre Maria Elcy Muñoz y a mi padre Hernán Marino Trullo, a quienes les debo la vida y todo lo que hasta el momento soy, sin ellos el sueño de educarme en una universidad no hubiera sido 'cimentado'.

En segundo lugar, agradecer al Ingeniero Juan Miguel Villa De Latorre por haberme instruido a la práctica de mis conocimientos dentro de su empresa y por darme consejos de vida valiosos que siempre voy a tener presentes, además de que fue él quien hizo posible el inicio de este proyecto con el préstamo de uno de sus robots de su empresa.

También agradecer a mi director de trabajo de grado, PhD. Óscar Andrés Vivas Albán, por el acompañamiento durante este último año de mi carrera universitaria y por los consejos de vida, que sin duda servirán como referencia para continuar mi proceso como profesional.

Finalmente y no menos importante, agradecer a mis compañeros y futuros colegas: Juan Sebastián Montenegro y Juan David Ruiz; sin temor a equivocarme, son el mejor equipo de trabajo. Fueron claves durante este proceso, pues nuestras discusiones académicas siempre tuvieron un nivel alto de debate gracias a los excelentes estudiantes que son.

Resumen

El presente proyecto plantea el diseño y construcción de una herramienta software para la manipulación y control de un robot PUMA *Programmable Universal Manipulation Arm* desde la plataforma Unity 3D.

Inicialmente, se realiza una descripción del robot PUMA, proporcionado por la empresa Ready Packers SAS de la ciudad de Popayán. Aquí se describen aspectos relacionados con la mecánica del robot, los sensores y la tarjeta controladora.

A continuación, se procede a la identificación de los requisitos del proyecto. Estos requisitos incluyen la persistencia de la información, la generación de trayectorias de grado 5, la controlabilidad, la interacción hombre-computadora y la comunicación electrónica. Se ilustran estos requisitos con diagramas de uso detallados que representan de manera concisa la interacción entre el operario y la aplicación, lo que proporciona una comprensión técnica de la dinámica de información en esta interacción.

En la fase de análisis técnico, se detallan las herramientas de software que se emplearon en la construcción de la aplicación. Además, se explican en profundidad los modelos matemáticos necesarios para la generación de trayectorias y el control del robot, así como la arquitectura de control utilizada.

El documento continúa con el diseño del software, donde se presenta el modelo de diseño que describe los aspectos visuales de la aplicación. Luego, se detallan los diagramas de clases y componentes que proporcionan una visión detallada y general, respectivamente, de la organización de la información y cómo cada componente accede a ella en la aplicación. Este es el primer paso

en la programación de la herramienta en software.

También, dentro de este proyecto se explica la construcción de los algoritmos utilizados. Cada algoritmo recoge los requerimientos en cuanto a interacción humano-computador, generación de trayectorias, comunicación y base de datos.

Finalmente, se llevan a cabo pruebas con el robot PUMA real, que involucran la creación de trayectorias articulares y cartesianas. Los resultados indican un error cuadrático medio de $\pm 1 cm^2$ en algunas trayectorias a altas velocidades y $\pm 0{,}5 cm$ a velocidades medias. Además, los errores angulares de $\pm 0{,}2^\circ$ en las articulaciones 1, 2 y 3 (articulaciones de traslación del robot).

Índice

Lista de figuras

Índice de figuras

Lista de tablas

Índice de cuadros

1. Introducción

Una herramienta de manipulación y control para prototipos de robots reales, es una interfaz que provee las facilidades a nivel de software para que se puedan asignar trayectorias, tanto en el espacio articular como en el espacio cartesiano, a diversos robots reales. Este tipo de herramienta deberá contar también con algoritmos que realicen una compensación a los desvíos en las trayectorias deseadas del robot. Algunos aportes en la academia utilizan el control CTC (control por par calculado) o también el control PD (proporcional-derivativo) [7], mientras otros utilizan controles híbridos, que son la combinación del CTC o el PID con diferentes conceptos de inteligencia artificial [8] [9]. Otra característica importante en estas interfaces es que poseen módulos de comunicación que se encargan de realizar el envío y recepción de los datos al acceder directamente a la tarjeta o placa conectada al hardware (sensores, motores, drivers).

Algunos estudios de investigación hablan acerca de la utilización de estas interfaces como herramienta de control y manipulación en aplicaciones de control, adquisición de datos, seguimiento de trayectorias y análisis de las desviaciones. Todo lo anterior es utilizado en las etapas iniciales de la creación de robot reales, por lo que estas interfaces son de suma importancia casi en cualquier aplicación donde se involucren robots.

Por lo tanto, el enfoque de este proyecto es la creación de una herramienta de manipulación y control para un robot real que cumpla con las características de control, adquisición de datos, asignación de trayectorias, análisis de desviaciones y visualización de gemelos digitales, características de una herramienta de alto nivel.

Se dispone de un robot Puma de 6 grados de libertad proporcionado por la empresa Ready Pac-

kers S.A.S. de la ciudad de Popayán. Este robot opera en modo de lazo abierto y emplea motores paso a paso como actuadores, así como sensores de posición para el monitoreo en tiempo real. La comunicación con el microcontrolador Raspberry Pi se establece mediante UDP. La herramienta de software diseñada desempeña un papel fundamental al proporcionar el par necesario para alcanzar posiciones específicas y, al mismo tiempo, recopila los valores angulares en cada instante de muestreo.

Durante las pruebas de simulación realizadas con la Raspberry Pi, se registraron errores de seguimiento en las articulaciones del robot. En las primeras articulaciones de posición, el error promedio fue de aproximadamente $\pm 0{,}4^{\circ}$, mientras que en las articulaciones finales, el error de seguimiento fue de $\pm 0{,}5^{\circ}$. El error cuadrático medio obtenido fue de aproximadamente $\pm 1cm$. Asimismo, se llevaron a cabo pruebas con los sensores J3 y J5 reales del robot, revelando errores de seguimiento de $\pm 0{,}4^{\circ}$ en dichas articulaciones. Pruebas con el robot real arrojaron errores de seguimiento de $\pm 0{,}45^{\circ}$ en las articulaciones 1,2,3; Lo cual implicó un error cuadrático medio de $\pm 0{,}9cm$.

2. Estado del Arte

2.1. Conceptos preliminares

La robótica es un campo multidisciplinario que involucra aspectos de la ingeniería, como la programación o computación, electrónica, física, mecánica, entre otros. Por otro lado, la automatización es el conjunto de técnicas que se utilizan para generar sistemas o procesos que funcionen sin la intervención humana o parcialmente.
La unión de estas dos disciplinas produce sistemas robóticos autónomos o semi-autónomos, que realizan tareas de manera repetitiva y con cierto grado de precisión.

Los robots se definen como máquinas autónomas capaces de realizar tareas de manera automática. Estas máquinas se utilizan para trabajos como la soldadura, el ensamblaje de partes y autopartes de otras máquinas, la cirugía, la exploración espacial, la agricultura. Los robots se dividen en dos grandes categorías, **robots industriales** y **robots móviles**. Los robots industriales se utilizan para procesos de manufactura en la realización de trabajos repetitivos a lo largo de una línea de producción. Los robots móviles se utilizan para la exploración espacial y la agricultura de alta precisión.

La robótica moderna se basa en gran medida en la computación y, últimamente, en la inteligencia artificial. Posee un sistema de sensores y algoritmos que le permiten percibir cierta parte del mundo que los rodea para tomar decisiones basadas en esta información. Esto les permite aprender y adaptarse a nuevas situaciones dentro del entorno de trabajo.

2.2. Interfaces de control y manipulación para robots

Recientemente, algunos estudios de investigación apuntan a la realización de este tipo de herramientas para prototipos de robots reales y virtuales. Es el caso de [1], el cual implementa una interfaz para el control y manipulación de un robot SCARA, estableciendo comunicación entre varias plataformas como SolidWorks, Arduino y LabView: SolidWorks para observar el gemelo digital, Arduino como interfaz de adquisición de datos y accionamiento de actuadores, y finalmente LabView como plataforma para generar las trayectorias y realizar el control en tiempo real del robot. Este autor concluye que gracias a la plataforma desarrollada se puede manipular el robot sin la necesidad de tener sistemas complejos como aquellos suministrados por los grandes fabricantes que construyen este tipo de software.

Figura 1: ”Front”panel para controlar y visualizar los parámetros del manipulador SCARA [1]

La 1 muestra cómo la interfaz provee disposiciones para asignar posiciones en el espacio cartesiano (x,y,z) al robot SCARA, del mismo modo, muestra los valores que se obtienen de cada articulación en tiempo real, así como el espacio para la comunicación con la tarjeta con la que se manipula directamente el robot. Algo a considerar en esta interfaz es que no proporciona una disposición para guardar trayectorias ya asignadas. Sin embargo, una de sus características a resaltar es que se evidencian los valores tanto en el sistema de coordenadas espacial como cartesiano,

fundamental para realzar análisis más completos acerca de los movimientos complejos del robot.

Otro caso es el de [10], pues muestra el diseño de un software diseñado en Matlab, el cual es probado inicialmente en un robot virtual en CoppeliaSim para finalmente accionar un robot real. Las trayectorias y el control son realizados en el entorno Matlab-Simulink y la comunicación con CoppeliaSim y con el robot real se hace mediante ROS, obteniéndose movimientos del robot real iguales a los programados en Matlab.

Del mismo modo que el caso anterior, [2] diseña una interfaz para manipular un robot SCARA de 6DOF en el espacio articular , el cual está basado en una GUI Matlab/Simulink para la visualización de la interfaz y control del robot. Este software cuenta con varias características que permiten, entre otras: llevar al robot a la posición inicial por defecto, asignar trayectorias a cada articulación por separado, y parar la ejecución en cualquier momento. Concluye que el diseño de la interfaz proporcionó flexibilidad para asignar trayectorias múltiples, aunque se presentó una gran complejidad al realizar su diseño y elaboración.

La 2 muestra una parte de la interfaz desarrollada. En ella se puede evidenciar un caso de asignación de trayectorias, en este caso hasta 10 posiciones articulares, además cuenta con un módulo para bloquear o abrir y cerrar la gripper en el caso en que se requiera. Como se aprecia, esta es una GUI realizada en Matlab/Simulink por lo que el motor que se requiere para compilarla es el propio Matlab.

En estas interfaces es común validar algunos modelos y errores de seguimiento. [3] lo imple-

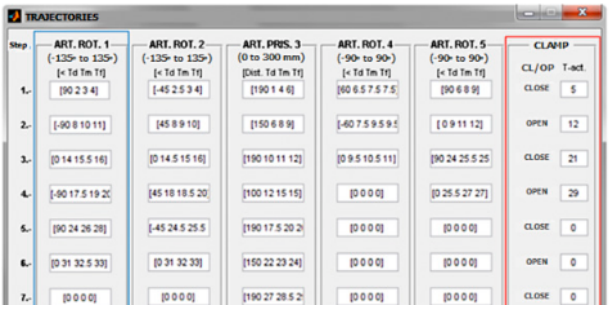


Figura 2: Interfaz de programación de trayectorias[2]

menta, pues en su artículo presenta la descripción del software y hardware utilizado para el manejo de un modelo de robot industrial de 4 DOF. El software permite la asignación de trayectorias en el espacio articular y cartesiano en el modo automático o manual, el control de los motores que accionan el robot es un algoritmo basado en el controlador PID. El artículo concluye que la velocidad del efector final influye de manera inversa en la precisión del robot, afectando la velocidad de seguimiento y la posición final de las trayectorias. Finalmente, menciona que en trabajos futuros se plantea la creación de una biblioteca y un kit de ejecución que permita la conexión rápida del robot a la interfaz y la mejora del error de seguimiento.

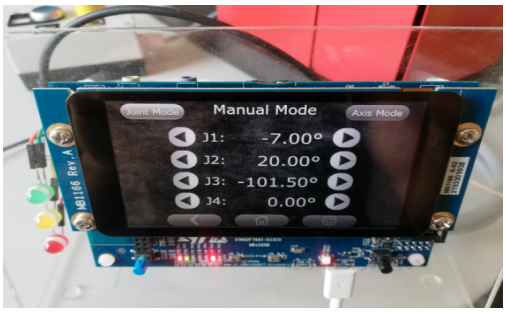


Figura 3: Modo manual- control articular [3]

La figura 3 muestra la interfaz vista desde una ”PanelView”, en donde se pueden asignar las posiciones en cada una de las articulaciones en modo manual o automático. El diseño de la interfaz es sencillo y práctico al mismo tiempo, pues cumple con la finalidad de asignar este tipo de trayec-

torias tanto en el eje de coordenadas cartesianas (axis mode) como en el modo articular (join mode).

Otros autores se han aventurado a realizar interfaces que validan modelos más complejos como el dinámico, y del mismo modo que lo mencionado anteriormente, validan el comportamiento de las trayectorias, solo que esta vez lo hacen sin un prototipo real. Es el caso de [11], que diseña una plataforma para el cálculo de la cinemática inversa y la dinámica directa de distintos tipos de robots: SCARA, Cartesiano, y Manipulador Robótico (ARM); teniendo en cuenta distintos parámetros como los centros de masa y los frotamientos, entre otros. La plataforma presenta gráficas acerca del comportamiento de cada tipo de robot relacionado con el seguimiento de las trayectorias como: error, velocidades y torques. La plataforma permite asignar las trayectorias a cada robot y en él se pueden analizar los torques, velocidades y las configuraciones singulares; concluye que este tipo de software es útil para comprender el funcionamiento de cada robot y el dimensionamiento de los mismos, por lo que es una herramienta útil para el diseño y prueba de diversas configuraciones.

Algunos autores también han creado interfaces para robots cartesianos (robots cuyas articulaciones son prismáticas). [4] por ejemplo, construye una interfaz para manipular un robot cartesiano de 2DOF para el embalaje mediante la técnica de “motion control” a través de la plataforma LabView. Utiliza una comunicación DCMNET para enviar los comandos de control desde un computador industrial por medio de la tarjeta denominada como “motion control driver”. En esta plataforma el usuario puede mover el robot en el sistema de coordenadas cartesianas, cambiar velocidades de movimientos, evidenciar la posición actual del robot, así como observar indicadores que alertan de la finalización de la tarea de *pick and place* del robot. Concluye que el robot tiene la capacidad de llegar a cualquier punto dentro del sistema de coordenadas (x,y), además de que proporciona un soporte de referencia para la asignación de trayectorias más complejas.

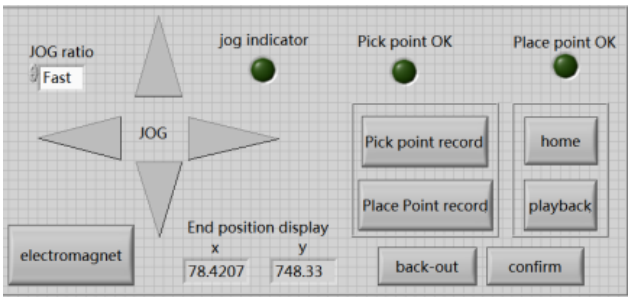


Figura 4: Interfaz de enseñanza del manipulador [4]

La interfaz presentada en 4, muestra una serie de comandos y disposiciones para que el usuario manipule el robot. Debido a que el robot es un cartesiano de 2DOF, los botones en forma de triángulo cumplen la función de moverlo en el plano (x,y). También esta interfaz muestra las posiciones del robot tanto en el eje x como y, que son equivalentes a las de cada articulación prismática.

Dentro de la investigación se ha encontrado que existen autores que han diseñado *frameworks* para diseñar aplicaciones para manipular robots. Es el caso de [12], que desarrolla un *framework* para diseñar aplicaciones enfocadas a aplicaciones robóticas. Este *framework* se encuentra dividido en varios módulos o capas que se encargan de los aspectos que integran soluciones de robótica, tales como: sensores, actuadores, interfaz de usuario, trayectorias, control y comunicación en tiempo real. El sistema desarrollado es ideal para la construcción de aplicaciones independiente del proveedor; además, permite flexibilidad para aplicar conceptos ya establecidos en el campo, así como la integración de una variedad de dispositivos de diferentes fabricantes como KUKA, Stäubli, Universal Robots, etc.

En este *framework* se han realizado una variedad de aplicaciones, una de estas aplicaciones es la que presenta [13], pues desarrolla una herramienta intuitiva para tele-operar un dúo de brazos robóticos de 6DOF, utilizando el marco de programación orientado a objetos conocido como SoftRobot y como plataforma de interfaz de usuario Microsoft Software. Este software permite al

tele-usuario ubicar los robots en una posición determinada, realizar operaciones de *pick and place* así como trayectorias más complejas. El artículo muestra que la interconexión de estos dos robots se hizo de manera sencilla, esto gracias a la API SoftRobot. Del mismo modo, la potencia de la solución se vio reflejada en la calidad de los movimientos y facilidad con que el usuario realiza las pruebas, sin embargo, se presentan algunos inconvenientes cuando el robot sobrepasa una configuración singular, uno de ellos es el reinicio obligatorio que este debe hacer.

Algunas aplicaciones requieren que el robot tenga un gemelo digital, esto es debido a que proporciona una herramienta de simulación en tiempo real con el fin de evidenciar el correcto seguimiento de las trayectorias. Este es el caso de [14], que presenta el diseño de una interfaz natural para controlar y manipular un robot PA10 de 7 DOF. El control y la asignación de trayectorias se realiza con una interfaz de usuario diseñada en Matlab y la visualización del gemelo digital se hace en la herramienta Unity 3D. Las pruebas muestran un error de seguimiento de cerca del 10 % entre el robot y el movimiento de la mano del usuario; finalmente, trabajos futuros proponen construir un entorno donde se simula un abdomen humano, para probar la viabilidad del mecanismo en operaciones quirúrgicas.

Existen casos en donde se han utilizado otras herramientas de software libre o código abierto, es el caso de [15] el cual plantea una arquitectura de software en tiempo real basada en Linux para controlar y manipular un robot SCARA de 4DOF vía internet. Los autores mencionan que la interfaz se diseña con la tecnología HTML por lo que casi cualquier dispositivo inteligente con acceso a internet puede acceder a dar órdenes al robot. Concluyen que Linux en tiempo real es un software bastante flexible debido a que proporciona facilidad de realizar software robusto a bajo costo, además, cuenta con la facilidad de adaptación a las distintas situaciones.

En el Departamento de Electrónica, Instrumentación y Control de la Universidad del Cauca, se encuentran una serie de robots que se han construido a lo largo de los años, los cuales se manejan actualmente con interfaces propias de Arduino. Estos robots son, junto con sus grados de libertad: Robot PUMA (6DOF), robot quirúrgico Latbot (3DOF), robot portaendoscopio Hibou (3DOF), robot PA10 (7DOF), además de 4 pequeños robots comerciales de (4DOF) y (5DOF). Y gracias a la empresa Ready Packers, se tiene un robot industrial tipo PUMA (6DOF). El objetivo es construir una interfaz de alto nivel para el control de este robot industrial, que pueda luego ser escalable a los demás robots del Departamento.

3. Proceso de construcción del robot

3.1. Robot utilizado

El robot PUMA (*Programmable Universal Machine for Assembly*) es una familia de robots industriales de brazo articulado, que incluye al PUMA560, PUMA560LX, PUMA 760, PUMA 762 y otros modelos. Estos robots fueron desarrollados en la década de 1970 por la compañía Unimation, siendo uno de los primeros robots industriales ampliamente utilizados en la fabricación y producción en masa.

3.1.1. Descripción robot PUMA real

El robot PUMA mostrado en la figura 5, muestra el robot Puma objeto de estudio, el cual es un robot proporcionado por la empresa Ready Packers de la ciudad de Popayán. Este robot tiene una estructura que pesa alrededor de $100Kg$. Comienza con una base fija la cual está a $1{,}20m$ del

Figura 5: Robot PUMA: Empresa Ready Packers

suelo. Los cuerpos $D3$ y $R4$ (ver tabla 2), miden $0{,}63m$ y $0{,}51m$ respectivamente. Estos valores son los mismos valores que se utilizaron en los modelos que se obtienen más adelante.

- Accionadores.

  Inicialmente, está el motor asociado a la primera articulación 1, este motor es un motor paso a paso dirigido por un 'AC Servo Driver'. Este motor es un potente actuador el cual tiene a la salida un máximo de 2.5HP de potencia con un torque 6.5Nm. Además, cuenta con una serie de opciones para programar el tipo de control, tales como: torque/velocidad, torque/posición, posición, velocidad, torque [16], ver figura 6(b). Después está el motor asociado a la articulación 2; este motor posee las mismas características del motor anterior. Las articulaciones 3, 4, 5 y 6 están asociadas a motores con un menor torque y potencia. Se utilizaron los motores Nema 17 (motores 4 y 5), Nema 34 (Motor 3) y Nema 11 (Motor 6) (ver 1). Adicionalmente, estos motores paso a paso se pueden controlar, cuando la frecuencia de los pulsos de arranque está entre un rango especificado por la curva par/frecuencia que tiene cada motor.

Motor	Referencia	Par (Nm)	Velocidad (rpm)
1	AC Stepper	6.5	3000
2	AC Stepper	6.5	3000
3	Nema 34	3	1000
4	Nema 17	1	1000
5	Nema 17	1	1000
6	Nema 11	0.12	1000

Cuadro 1: Características de los motores del robot

Cada motor tiene asociado un driver, estos son los encargados de convertir la acción de control en torque/posición hacia los motores. El motor 1 y 2 conectados al 'AC Servo Driver' (ver figura 6(b)); los motores 3, 4, 5, 6 conectados al driver del modelo CY556 (ver figura 6(a)). El accionamiento de estos motores se traduce en la cantidad de pulsos (LOW-HIGH) enviados al driver y el tiempo entre estos pulsos. Por lo anterior, desde la aplicación se deben enviar los torques que tienen una relación con el tiempo entre pulsos (frecuencia). Cabe resaltar que para realizar el movimiento de estos actuadores se requiere del envío de la posición convertida a pulsos. Por lo tanto, el controlador debe calcular tanto la frecuencia entre pulsos como los pulsos necesarios para llegar a una posición determinada.

El diagrama eléctrico de la conexión entre la Raspberry y los drivers se muestra en la figura

(a) Driver motores (3, 4, 5, 6) (b) AC Driver motores (1,2)

Figura 6: Drivers motores robot

7 y 8.

Los transistores son necesarios, pues la salida de voltaje de la Raspberry es 3.3V y los drivers tienen como entrada 5V en la señal de control. Para el caso se han utilizado transistores **2n2222**.

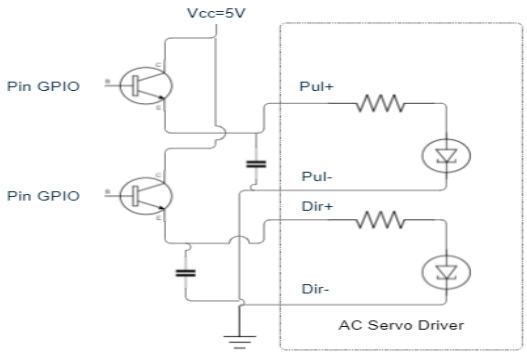


Figura 7: Diagrama eléctrico entre la Raspberry y los drivers 1 y 2 sin desacople

- Sensores

  En cuanto a los sensores utilizados, los motores 1y 2 los traen incorporados, es decir, se debe acceder a estos valores posicionales directamente desde el driver que lo controla. Para

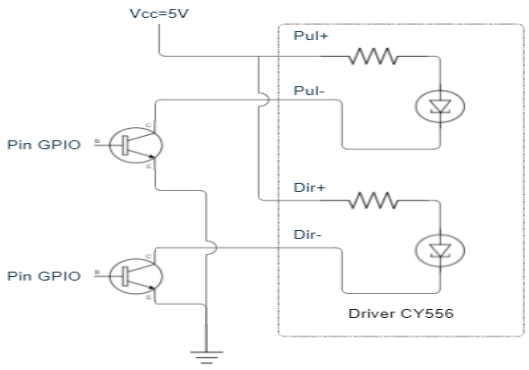


Figura 8: Diagrama eléctrico entre la Raspberry y los drivers 3,4,5 y 6 sin desacople

las siguientes articulaciones se han utilizado 'Rotary Encoders', el cual se muestra en la figura 10, de la marca Taiss. Estos sensores tienen un voltaje de referencia de 5 a 24 V DC, una velocidad de detección de pulsos de hasta 3000 rpm/min y 1000 pulsos/rev. Además poseen 2 canales de datos los cuales envían un tren de pulsos cuando están en movimiento (uno desfasado del otro, 90 grados). Con esto se garantiza una lectura en la posición de la articulación y su dirección mediante un algoritmo de detección de cuadratura (ver figura 9) [17]. Para obtener esta lectura se ha desarrollado dicho algoritmo en una tarjeta Arruino, la cual se comunica serialmente con la Raspberry a través de los puertos /dev/ttyAMA- (puertos USB de la Raspberry). Este algoritmo se muestra en el anexo (72).

- Tarjeta Controladora

  La tarjeta controladora del robot es una Raspberry PI3 B la cual posee características tales como: 2Gb RAM, 64Gb de espacio en disco (variable), E/S seriales USB y USB2, adaptadores de red Wifi y Ethernet, y un sistema operativo basado en Linux conocido como Rasbian.

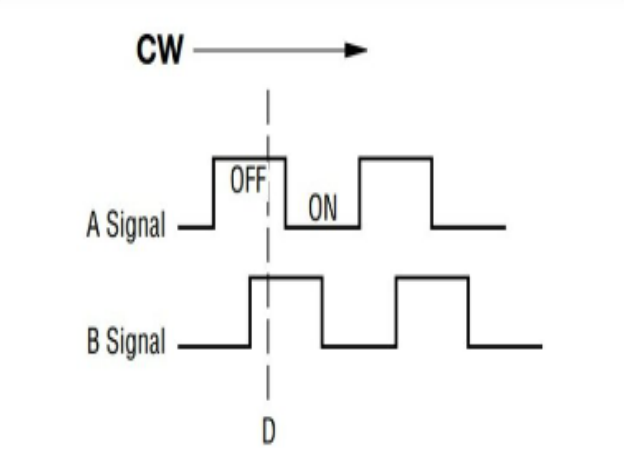


Figura 9: Representación de una señal de cuadratura (tomado de [5])

Figura 10: Sensor encoder rotativo (tomado de la web)

Figura 11: Raspberry PI 3B

3.1.2. Descripción matemática general del robot

El robot utilizado es un modelo PUMA que presenta articulaciones rotoides en su arquitectura. Este robot está compuesto por seis grados de libertad, que se organizan de la siguiente manera: la

primera articulación (R1), seguida por la segunda articulación (R2) que está desfasada 90 grados con relación a la anterior, luego un eslabón con una dimensión (D3), que corresponde al brazo del robot. A continuación se encuentra la tercera articulación (R3) seguida de otro eslabón con una dimensión (r4), correspondiente al antebrazo. Finalmente, el robot cuenta con una 'muñeca' compuesta por tres articulaciones rotoides (R4, R5 y R6).

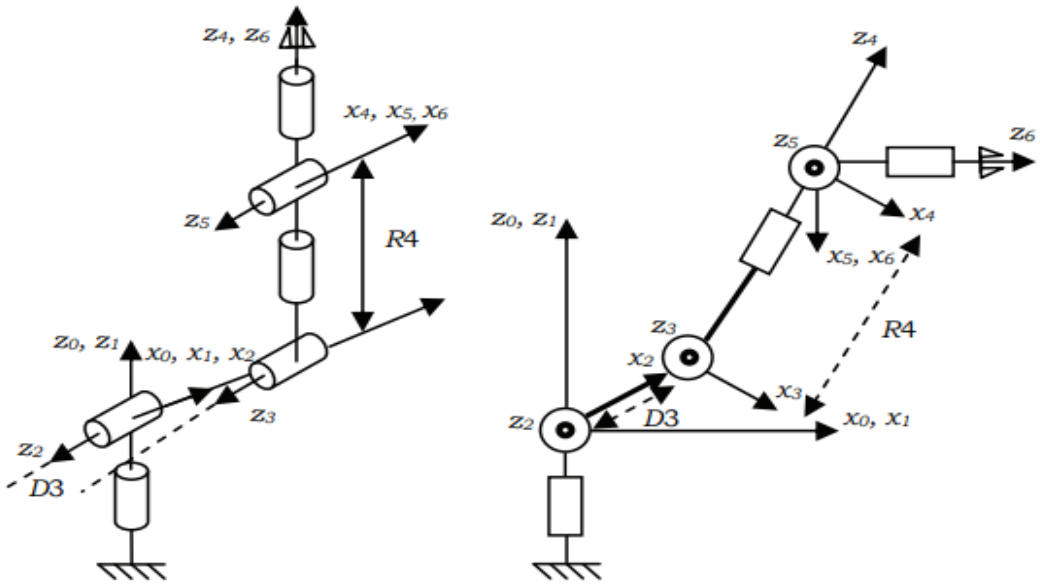


Figura 12: Arquitectura robot Puma

Ahora bien, los valores de θ_j, α_j, r_j y d_j correspondientes a los valores geométricos del robot PUMA son:

j	γ_j	α_j	r_j	θ_j	d_j
1	0	0	0	θ_1	0
2	0	90	0	θ_2	0
3	0	0	D3	θ_3	0
4	0	-90	0	θ_4	R4
5	0	90	0	θ_5	0
6	0	-90	0	θ_6	0

Cuadro 2: Parámetros geométricos Robot Puma 560

3.1.3. Modelos matemáticos

Un robot manipulador serial es un mecanismo compuesto de una cadena de eslabones unidos por articulaciones que poseen su propio sistema de referencia, es decir, posición y rotación determinadas. Un robot manipulador serial puede tener *n* grados de libertad, los cuales dependen de las características y disposición geométrica de cada articulación.

El modelo geométrico que describe esta cadena de eslabones está basado en el método de **Khalil-Kleinfinger** implementado en [7], que consiste en una matriz de transformación homogénea que describe las posiciones y orientaciones de 'ese' sistema de referencia respecto al anterior.

$$ {}^{\text{n-1}}T_n = \begin{bmatrix} C\theta_j & -S\theta_j & 0 & d_j \\ C\alpha_j S\theta_j & C\alpha_j C\theta_j & -S\alpha_j & -r_j S\alpha_j \\ S\alpha_j S\theta_j & S\alpha_j C\theta_j & C\alpha_j & r_j S\alpha_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

Por lo anterior, las matrices de transformación homogénea ${}^0T_1\ {}^1T_2\ {}^2T_3\ {}^3T_4\ {}^4T_5\ {}^5T_6$ del robot Puma se calculan reemplazando los valores de la tabla 2. Por simplicidad se tiene que C_n, S_n, S_{nm}, C_{nm} son el seno o coseno de los ángulos θ_j

$$ {}^0T_1 = \begin{bmatrix} C1 & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

$$
{}^{1}T_{2} = \begin{bmatrix} C2 & -S2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S2 & C2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}
$$

$$
{}^{2}T_{3} = \begin{bmatrix} C3 & -S3 & 0 & D3 \\ S3 & C3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}
$$

$$
{}^{3}T_{4} = \begin{bmatrix} C4 & -S4 & 0 & 0 \\ 0 & 0 & 1 & R4 \\ -S4 & -C4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}
$$

$$
{}^{4}T_{5} = \begin{bmatrix} C5 & -S5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S5 & C5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}
$$

$$
{}^{5}T_{6} = \begin{bmatrix} C6 & -S6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ S6 & C6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}
$$

- **Modelo geométrico**

El modelo geométrico del robot es un modelo que describe las posiciones cartesianas en fun-

ción de las posiciones articulares y viceversa. Este modelo es hallado con las disposiciones mencionadas anteriormente mostradas en la ecuación 1.

- Modelo geométrico directo

  El MGD expresa las posiciones cartesianas (x,y,z, rotx, roty, rotz) en función de las posiciones articulares, es decir, al conocerse las posiciones articulares, se puede calcular la posición cartesiana en que se encuentra el efector final.

  Este modelo se obtiene al realizar una multiplicación matricial ordenada entre: 2 3 4 5 6 7.

  S_x = C1(C23(C4C5C6-S4S6)-S23S5C6)-S1(S4C5C6-C4S6)

  S_y = S1(C23(C4C5C6-S4S6)-S23S5C6)+C1(S4C5C6-C4S6)

  S_y = S23(C4C5C6-S4S6)+C23S5C6

  n_x = C1(-C23(C4C5C6-S4S6)-S23S5C6)+S1(S4C5C6-C4S6)

  n_y = S1(-C23(C4C5C6-S4S6)-S23S5C6)-C1(S4C5C6-C4S6)

  n_z = -S23(C4C5S6+S4C6)-C23S5S6

  a_X = -C1(C23C4S5 +S23C5)+S1S4S5

  a_X = -S1(C23C4S5 +S23C5)-C1S4S5

  a_X = -S23C4S5+C23C5

  P_x = -C1(S23R4-C2D3)

  P_y = -S1(S23R4-C2D3)

  P_z = C23R4+S2D3

- Modelo geométrico inverso

El MGI es el modelo que obtiene las posiciones angulares a partir de posiciones cartesianas determinadas, es decir, qué posiciones angulares se necesitan para llegar a una posición cartesiana determinada.

En [7] se utiliza el método de Paul para calcular este modelo. Los valores s_x, s_y, s_z, n_x, n_y, n_z, a_x, a_y, a_z son los valores correspondientes a la matriz de orientación y p_x, p_y, p_z el vector de traslación, en el espacio tridimensional.

Para calcular los valores de la matriz de orientación, hay varios métodos, uno de ellos es el cálculo por ángulos de Euler, que se basa en una rotación secuencial en cada uno de los ejes de referencia. En MATLAB por ejemplo, se encuentra como *eul2rotm(orix, oriy,oriz)*. En Unity se conoce como *Matrix4x4.TRS(Vector3.zero, rotation, Vector3.one)* donde 'rotation' es *Quaternion.Euler(orix, oriy, oriz)*.

Para el ángulo 1,

$$\theta_1 = atan(P_y, Px)$$

Para el ángulo 2,

$$\theta_2 = atan(S2, C2)$$

$$C2 = \frac{YZ - \epsilon X\sqrt{X^2 + y^2 - Z^2}}{X^2 + Y^2}$$

$$S2 = \frac{YZ + \epsilon X\sqrt{X^2 + y^2 - Z^2}}{X^2 + Y^2}; \epsilon = \pm 1$$

$$X = 2P_z D3$$

$$Y = -2B1D3$$

$$Z = (R4)^2 - (D3)^2 - (PZ)^2 - (B1)^2$$

$$B1 = C1P_x - S1P_y$$

Para el ángulo 3,

$$\theta_3 = atan(S3, C3)$$

$$S3 = \frac{-P_z S2 - B1C2 + D3}{R4}$$

$$C3 = \frac{-B1S2 + P_z C2}{R4}$$

Para el ángulo 4,

$$\theta_4 = atan(H_z, -H_x)$$

$$H_x = C23(C1a_x + S1a_y) + S23a_z$$

$$H_y = -S23(C1a_x + S1a_y) + C23a_z$$

Para el ángulo 5,

$$\theta_5 = atan(S5, C5)$$

$$C5 = H_y$$

$$S5 = -C4H_x + S4H_z$$

$$H_z = S1a_x - C1a_x$$

Para el ángulo 6,

$$\theta_6 = atan(S6, C6)$$

$$S6 = -C4F_z - S4F_x$$

$$C6 = -C4G_z - S4G_x$$

$$F_x = C23(C1s_x + S1s_y) + S23s_z$$

$$F_z = S1s_x - C1s_y$$

$$G_x = C23(C1n_x + S1n_y) + S23n_z$$

$$G_z = S1n_x - C1n_y$$

Al enviar al robot las posiciones angulares calculadas, se puede llevar el órgano terminal del robot a cualquier posición (x, y, z) y orientación (rotx, roty, rotz) deseada. De esta manera se logra un control preciso del movimiento del robot y se pueden realizar tareas específicas de manera eficiente y confiable.

- **Dinámica del robot**

El modelo dinámico de un robot industrial es una representación matemática que describe

cómo el robot se mueve y se comporta en respuesta a las fuerzas que actúan sobre él. Este modelo se utiliza para diseñar y controlar el movimiento del robot, así como para predecir su comportamiento en diferentes situaciones.

El modelo dinámico de un robot industrial se compone de varias partes. Una de las partes principales es la cinemática, que describe la relación entre el movimiento del robot y sus parámetros físicos, como la longitud de sus brazos y la posición de sus articulaciones. Otra parte importante es la dinámica, que describe cómo el robot responde a las fuerzas que actúan sobre él, como la gravedad, la fricción, y la resistencia del aire.

El modelo dinámico del robot se utiliza para diseñar controladores de movimiento que permiten al robot realizar tareas específicas. Estos controladores utilizan la información del modelo dinámico para ajustar la velocidad, la posición y la aceleración del robot para realizar la tarea de forma precisa y eficiente.

En esta ocasión se obtiene el **modelo dinámico inverso** del robot, que es el encargado de obtener los torques necesarios para llegar a las posiciones, velocidades y aceleraciones deseadas, el cual se representa de la siguiente manera.

$$\Gamma = f(q, \dot{q}, \ddot{q}) \tag{8}$$

- **Trayectorias Grado 5**

El polinomio de 5^{to} grado, es un modelo matemático utilizado en robótica y otras áreas para describir trayectorias de movimiento suaves y continuas. Este polinomio es especialmente útil porque garantiza una continuidad en la aceleración a lo largo de la trayectoria y tiene una aceleración inicial y final nulas, lo que lo convierte en una herramienta poderosa para la

planificación de movimiento en robots.

$$q(t) = 10\left[\frac{t}{tf}\right]^3 - 15\left[\frac{t}{tf}\right]^4 + 6\left[\frac{t}{tf}\right]^6 \tag{9}$$

La figura 13 muestra la trayectoria generada por esta función, en ella se aprecia que tanto el inicio como el final de la trayectoria son suaves, lo cual implica aceleraciones nulas al comienzo y al final, lo mismo que para las velocidades.

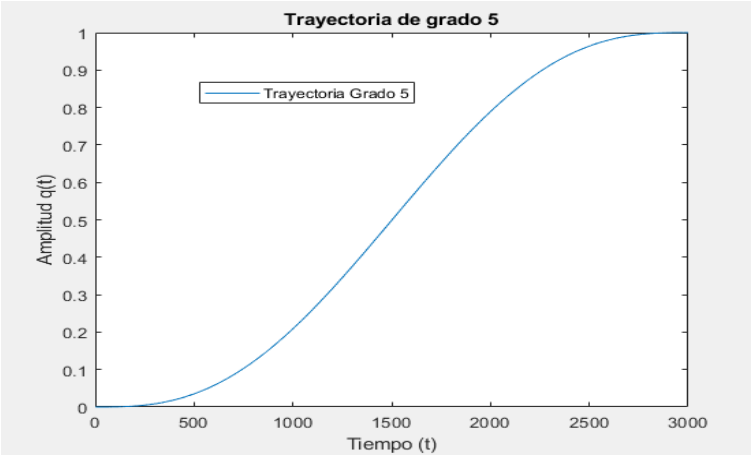


Figura 13: Trayectoria grado 5 de (0-1 rad)

3.1.4. Situación de los sensores del robot

Las pruebas en los sensores muestran resultados **NO** esperados en la posición angular de cada articulación. Inconvenientes como ruido eléctrico en las fuentes de voltaje, interferencias electromagnéticas cuando el robot está en movimiento, bloqueo de la Raspberry al momento de conectar las referencias **(GND)** a un mismo punto, fueron algunos de los sucesos no esperados que se tuvieron al momento de integrar los sensores y drivers-motores al sistema. Es de notar que el robot se recibió en ese estado, venía ya construido y el propósito del proyecto era realizar su interfaz de

control y manipulación, mas no implicaba cambiar su hardware o su diseño inicial.

Para solucionar parte de estos inconvenientes se optó por desacoplar el circuito de la Raspberry y los drivers con el **opto-acoplador 4N33** con el fin de lograr su accionamiento, ver figura 14. Cabe resaltar que se ha cambiado el transistor por el opto-acoplador en todos los circuitos, la figura solo muestra lo realizado para el driver 1 y 2.

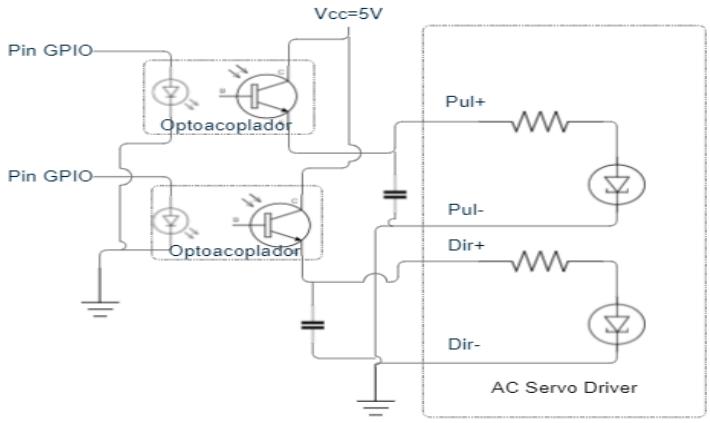


Figura 14: Diagrama electrónico del driver 1 y 2 desacoplado

4. Proceso de construcción del software

4.1. Captura de requisitos del software

4.1.1. Visión del producto

La interfaz será diseñada de tal manera que permita la comunicación bidireccional entre el robot y el computador. Esto significa que la interfaz debe ser capaz de recibir datos del robot, como su posición actual, y enviar datos al robot, como los valores de las posiciones y torques. Además, la interfaz deberá ser fácil de usar para que cualquier persona pueda operar el robot sin la

necesidad de tener conocimientos especializados.

La visión del producto también incluye la capacidad de proporcionar trayectorias articulares y cartesianas, así como el control en lazo cerrado. Esto permitirá al robot realizar tareas específicas dentro de un entorno industrial. La trayectoria cartesiana se refiere al movimiento del robot en el espacio tridimensional y la trayectoria articular se refiere al movimiento de las articulaciones del robot. El control en lazo cerrado asegurará que el robot se mueva con precisión y se ajuste automáticamente a cualquier cambio en las condiciones del entorno.

Además de las funcionalidades mencionadas anteriormente, la interfaz incluye la integración con una base de datos. Esta base de datos permitirá al usuario guardar las trayectorias que usa con mayor frecuencia y acceder a ellas de manera rápida y sencilla. La base de datos también puede incluir información sobre las tareas que el robot ha realizado en el pasado, lo que puede ser útil para analizar y mejorar el rendimiento del robot en el futuro. La integración de una base de datos en la interfaz del robot no solo proporciona una funcionalidad adicional para el usuario, sino que también es una herramienta valiosa para el análisis y mejora continua del proceso en el que el robot está involucrado.

Otra característica importante de la interfaz del robot PUMA de 6 grados de libertad es la visualización de un gemelo digital del robot. Un gemelo digital es una réplica virtual de un objeto físico y en este caso, del robot. La visualización del gemelo digital permitirá al usuario corroborar la validez de las trayectorias cartesianas antes de enviarlas al robot. El gemelo digital también puede ser utilizado para simular y optimizar las trayectorias antes de que el robot las ejecute físicamente en la planta industrial. Esta funcionalidad de visualización es crucial para garantizar la seguridad del operador y la eficiencia del proceso industrial. Con la visualización del gemelo digital, el usuario puede hacer ajustes a las trayectorias antes de enviarlas al robot, evitando así posibles colisiones o

movimientos no deseados. En general, la visualización de un gemelo digital proporciona una herramienta valiosa para mejorar el rendimiento y la seguridad del robot, y por lo tanto, del proceso en el que se utiliza.

4.1.2. Diagrama de casos de uso

El diagrama de casos de uso de la aplicación (ver figura 15) para manipular el robot PUMA 560 se compone de cinco elementos. El usuario es el actor que interactúa con la aplicación. El primer caso de uso permite al usuario conectarse al robot. A continuación, el usuario puede elegir entre dos opciones: acceder a la base de datos de trayectorias previamente guardadas o crear nuevas trayectorias utilizando la aplicación. En ambos casos, el usuario puede simular la trayectoria en el gemelo digital del robot para verificar su validez. Finalmente, el usuario puede enviar la trayectoria validada al robot para su ejecución en el proceso. En resumen, el diagrama de casos de uso permite al usuario interactuar con el robot PUMA 560 de manera eficiente y segura, asegurando la validez de las trayectorias antes de ser ejecutadas en el entorno.

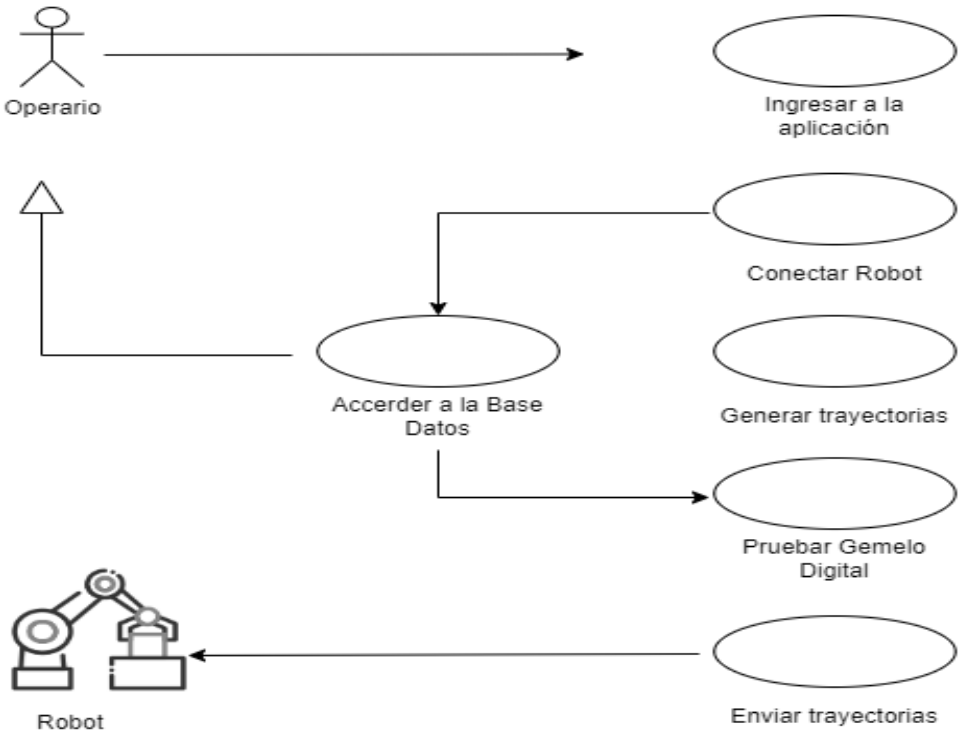


Figura 15: Ejemplo diagrama de casos de uso

4.1.3. Casos de uso detallado

Caso de uso	Ingresar a la aplicación
Actores	Operario
Prioridad	Media
Código	R1
Referencias cruzadas	R2

Descripción	El operario ingresa a la aplicación, donde se muestra un panel de bienvenida. El usuario puede navegar por las diferentes pestañas de la interfaz, pestaña de espacio cartesiano, articular, base de datos y trayectorias libres; sin embargo, no puede accionar el robot hasta que haya comunicación con el controlador.

Cuadro 3: Caso de uso detallado (ingresar aplicación)

Caso de uso	Conectar robot
Actores	Operario
Prioridad	Alta
Código	R2
Referencias cruzadas	R1
Descripción	Para que el operario pueda controlar el robot, es necesario establecer una conexión con el controlador encargado de accionarlo y recibir los datos de los sensores. La aplicación requerirá que exista una comunicación activa entre la misma y el controlador para poder cargar las trayectorias en el robot. Sin esta comunicación en un estado activo, la transferencia de las instrucciones no podrá realizarse de manera efectiva. Es fundamental asegurar una conexión estable y operativa entre la aplicación y el controlador para garantizar un control adecuado del robot y el intercambio fluido de información entre ambos.

Cuadro 4: Caso de uso detallado (Conectar robot)

Caso de uso	Generar trayectorias
Actores	Operario
Prioridad	Media
Código	R3
Referencias cruzadas	R1
Descripción	El operario cuenta con dos opciones para generar trayectorias: crearlas en el espacio articular, es decir, estableciendo movimientos directos de las articulaciones; o generarlas en el espacio cartesiano, lo que implica movimientos en un espacio tridimensional. Para facilitar estas tareas, el usuario dispone de varias herramientas gráficas que le permiten crear las trayectorias mencionadas anteriormente de manera intuitiva y visual. De esta manera, se promueve una experiencia más fluida y accesible para el usuario.

Cuadro 5: Caso de uso detallado (Generar trayectorias)

Caso de uso	Acceder a la base de datos
Actores	Operario
Prioridad	Media
Código	R4

Referencias cruzadas	R1
Descripción	Al acceder a la pestaña de la base de datos, el operario dispone de varias opciones relacionadas con las trayectorias. Estas opciones incluyen guardar, modificar, eliminar y cargar trayectorias. Cuando se crea una nueva trayectoria, el operario tiene la posibilidad de guardarla en la base de datos para su uso posterior. Además, se le proporciona la capacidad de modificar trayectorias existentes o aquellas que están guardadas en la base de datos. Asimismo, se brinda una interfaz para eliminar trayectorias almacenadas que ya no son necesarias. Por último, el operario puede cargar trayectorias previamente guardadas con el fin de probarlas en el gemelo digital. Estas funcionalidades proporcionan al operario un mayor control y flexibilidad en la gestión de las trayectorias almacenadas en la base de datos.

Cuadro 6: Caso de uso detallado (Acceder a la base de datos)

Caso de uso	Probar gemelo digital
Actores	Operario
Prioridad	Alta
Código	R5
Referencias cruzadas	R1

Descripción	En esta situación, el operario realiza pruebas en las trayectorias una vez que han sido creadas o modificadas. El gemelo digital desempeña un papel crucial en la validación de estas trayectorias. Una vez que se han verificado los movimientos esperados en el gemelo digital, se procede a enviar las trayectorias al controlador responsable de gestionar el robot. Esta validación en el gemelo digital garantiza que las trayectorias sean correctas y seguras antes de ser ejecutadas por el robot real. De esta manera, se minimizan los riesgos y se asegura un funcionamiento preciso y confiable del robot con base en las trayectorias diseñadas por el operario.

Cuadro 7: Caso de uso detallado (Probar gemelo digital)

Caso de uso	Enviar trayectorias
Actores	Operario
Prioridad	Alta
Código	R6
Referencias cruzadas	R1

Descripción	En este caso de uso, el operario tiene la capacidad de enviar la trayectoria que ha sido creada o cargada desde la base de datos. La aplicación también ofrece la función de visualización en tiempo real, lo que permite al operario observar las posiciones articulares y cartesianas del robot mientras se ejecuta la trayectoria. Esta característica proporciona una retroalimentación inmediata y visual al operario, permitiéndole monitorear y verificar el comportamiento del robot en tiempo real.

Cuadro 8: Caso de uso detallado (Enviar trayectorias)

4.2. Análisis

4.2.1. Descripción de las herramientas software

- **Unity 3D** es un motor de videojuegos que utiliza el lenguaje de programación C# basado en .NET, además de otros lenguajes de programación como JavaScript. C# es un lenguaje de programación orientado a objetos, lo que significa que es capaz de representar objetos del mundo real en el código y manipularlos a través de métodos y propiedades. En cuanto a la creación de representaciones virtuales de un robot PUMA en Unity 3D, es posible modelar y animar el robot utilizando el editor de Unity, que incluye herramientas para la creación de modelos 3D, animaciones, efectos visuales y física.

- **El entorno .NET** es una plataforma de código abierto desarrollado por Microsoft, que permite el desarrollo y ejecución de aplicaciones en diversos lenguajes de programación, incluyendo C#. El entorno .NET proporciona una serie de herramientas y librerías que permiten

una mayor productividad y facilidad de desarrollo para los programadores.

- **C#** es uno de los lenguajes de programación más populares para el desarrollo de aplicaciones en el entorno .NET. C# es un lenguaje de programación moderno y elegante, diseñado para la creación de aplicaciones de alta calidad y eficiencia en el uso de recursos. Es un lenguaje de programación orientado a objetos que incluye características avanzadas, como la gestión de memoria automática, la herencia de clases y la sobrecarga de operadores. Una de las ventajas de C# es su capacidad para compilar y ejecutar aplicaciones en múltiples plataformas, incluyendo Windows, Linux y macOS, gracias a la implementación del entorno .NET. Además, C# es un lenguaje de programación muy popular en la industria de los videojuegos, especialmente en el desarrollo de juegos para plataformas como PC, Xbox y PlayStation.

- **Visual Studio Code** es un editor de código de uso común para programadores, es el editor de código por defecto de Unity. Ofrece una gran variedad de características útiles para la programación, como resaltado de sintaxis, autocompletado de código, depuración, y control de versiones integradas. Es compatible con varios lenguajes de programación, incluyendo C# y JavaScript, por lo que es una buena opción para programadores que trabajan en proyectos de Unity.

4.2.2. Algoritmos utilizados para la construcción de las mecánicas de la simulación

La virtualización de un robot PUMA utilizando el *framework* de Unity 3D y los diseños CAD determina el grado de fidelidad del gemelo. Para lograr esto, se requiere importar los diseños CAD del robot PUMA a Unity y luego ensamblar las diferentes piezas para crear el modelo completo

del robot.

Una vez que se han importado los diseños CAD y ensamblado las piezas en Unity, se podrán aplicar algoritmos de simulación para controlar la rotación de las articulaciones del robot. Estos algoritmos de simulación deben tener acceso a la rotación sobre el propio eje de referencia de cada articulación.

En Unity, se puede acceder a la rotación de las articulaciones del robot utilizando las transformaciones. Cada articulación del robot PUMA se puede representar como un objeto en Unity con su propio Transform. Se puede acceder y modificar la rotación de estas articulaciones utilizando las propiedades del Transform.

Una vez que se ha obtenido la rotación de cada articulación del robot, se puede relacionarla directamente con la posición angular de las articulaciones del robot PUMA. Esto permitirá controlar y simular los movimientos del robot virtual de acuerdo con las rotaciones definidas.

4.3. Controlador CTC

El control por par calculado constituye un tipo de control no lineal empleado en la gestión de robots. Este método se basa en el modelo dinámico del robot para determinar los pares necesarios que satisfacen una trayectoria específica. Se destaca por su robustez y precisión, lo que lo hace aplicable a robots de diversas dimensiones y complejidades. [18]

Este enfoque de control presenta varias ventajas en comparación con otros métodos. En primer lugar, demuestra ser robusto, mostrando eficacia en entornos con ruido y perturbaciones. Además, destaca por su precisión al seguir trayectorias complejas con exactitud. Por último, su versatilidad permite su aplicación en robots de diversas dimensiones y complejidades.

El control por par calculado encuentra aplicaciones en una amplia variedad de contextos robóticos, incluyendo robots industriales, de servicio, médicos e investigativos. Se consolida como una técnica esencial para la gestión efectiva de robots en diversas áreas de aplicación.[19]

4.3.1. Arquitectura de control

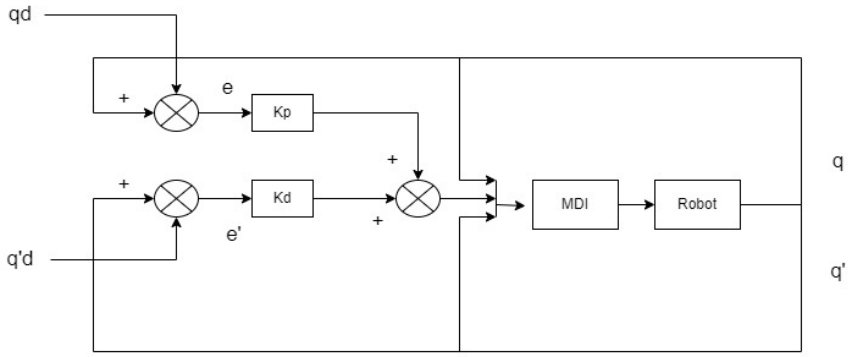


Figura 16: Arquitectura de control propuesta (fuente: Propia)

La Figura 16 ilustra la arquitectura de control CTC empleada en el robot. Los sensores del robot captan las posiciones y velocidades que se utilizan para calcular las aceleraciones correspondientes en el intervalo de tiempo de muestreo. Estas aceleraciones se envían al MDI (ecuación 8), que se encarga de calcular los torques necesarios para alcanzar las posiciones, velocidades y aceleraciones deseadas. En resumen, el sistema utiliza la retroalimentación de los sensores del robot para calcular los torques requeridos y así lograr el movimiento deseado.

4.3.2. Asignación del esfuerzo de control a los motores

El control de los motores paso a paso regularmente se realiza en lazo abierto; lo cual implica el posicionamiento angular del eje mediante la asignación de pulsos (LOW-HIGH) desde el microcontrolador, sin embargo, uno de los problemas comunes es la pérdida de los pasos debido a la

frecuencia mínima necesaria para que el motor arranque o continúe con el movimiento [20] [6].

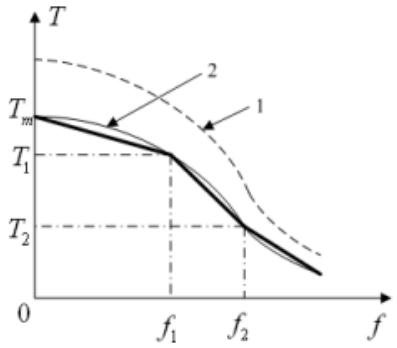


Figura 17: Curva característica par/frecuencia [6].

La figura anterior 17, muestra la zona de trabajo o también denominada campo de giro, delimitada por las curvas 1 y 2. La curva 1 representa la relación par/velocidad del motor en movimiento, mientras que la curva 2 representa la frecuencia de funcionamiento del motor en arranque. Es importante mantener a los motores del robot en esta zona de trabajo para garantizar que funcionen de forma eficiente. El valor de la frecuencia se calcula en función del valor del par calculado por el algoritmo de control CTC del software. Con este valor se garantiza la asignación consecuente del esfuerzo de control a los motores.

Una aproximación de la función, se podría expresar mediante una función lineal. Esta representa la relación entre la frecuencia y el par asignado:

$$T(tm) = T_m + K * f(tm) \tag{10}$$

- T_m = punto de corte con el eje y (T)
- K = Constante de par/frecuencia

- $T(tm)$ = par en el tiempo de muestreo
- tm = tiempo de muestreo
- $f(tm)$ = frecuencia
- Cabe aclarar que la pendiente K es negativa, a mayor frecuencia, el par es menor.

Al tomar los valores nominales de los torques de cada motor se establece que los motores van a trabajar al 80 $\%$ de la capacidad.

Motor	Referencia	Par (Nm)
1	AC Stepper	5.2
2	AC Stepper	5.2
3	Nema 34	2.4
4	Nema 17	0.8
5	Nema 17	0.8
6	Nema 11	0.1

Cuadro 9: Características de los motores del robot: torques nominales de operación

Los anteriores valores corresponden al valor T_m mostrado en la figura 17.

Las pruebas experimentales proporcionaron que la frecuencia óptima de trabajo está entre $13Khz$ y $15Khz$ y que el par de movimiento está entre T_m y 0. Con lo anterior, se pueden calcular los valores de la pendiente K de la ecuación 10.

Por ejemplo, para el motor 1 la ecuación queda representada de la siguiente manera:

$$T(t) = 18{,}72 - 0{,}001 * f(t) \tag{11}$$

La anterior ecuación es utilizada para valores de frecuencia entre $15Khz$ y $13Khz$.

4.3.3. Cálculo de los valores en el software

Del lado del software se obtiene el par necesario para que cada actuador llegue a la posición angular deseada. Es por esto que los valores obtenidos se escalizan a valores de frecuencia antes calculados. Por lo tanto, la función lineal que representa el par asignado a cada motor viene dado por:

$$T(t) = 18000 - 961{,}53 * f(t) \tag{12}$$

Con las ecuaciones 11 y 12 se garantiza la asignación del par a cada motor, para que este llegue a la posición asignada por la trayectoria.

4.4. Diseño de la herramienta

4.4.1. Modelo de diseño

En esta primera parte se realizó el diseño de la plataforma en el software de libre acceso Unity 3D junto al lenguaje de programación C#. En este entorno se programan los modelos de cada tipo de robot, estos modelos son el MGI y el MDI [7], además, se construyen los algoritmos para obtener las trayectorias grado 5 las cuales son las responsables de permitir el movimiento de arranque suave que garantiza un esfuerzo mínimo y menor riesgo de daño en los actuadores. Por otra parte, en el "*front*", se realiza el diseño del gemelo digital del robot objetivo de este proyecto y se crean

las disposiciones para que se permita guardar las trayectorias en una base de datos, las cuales serán realizadas por el robot.

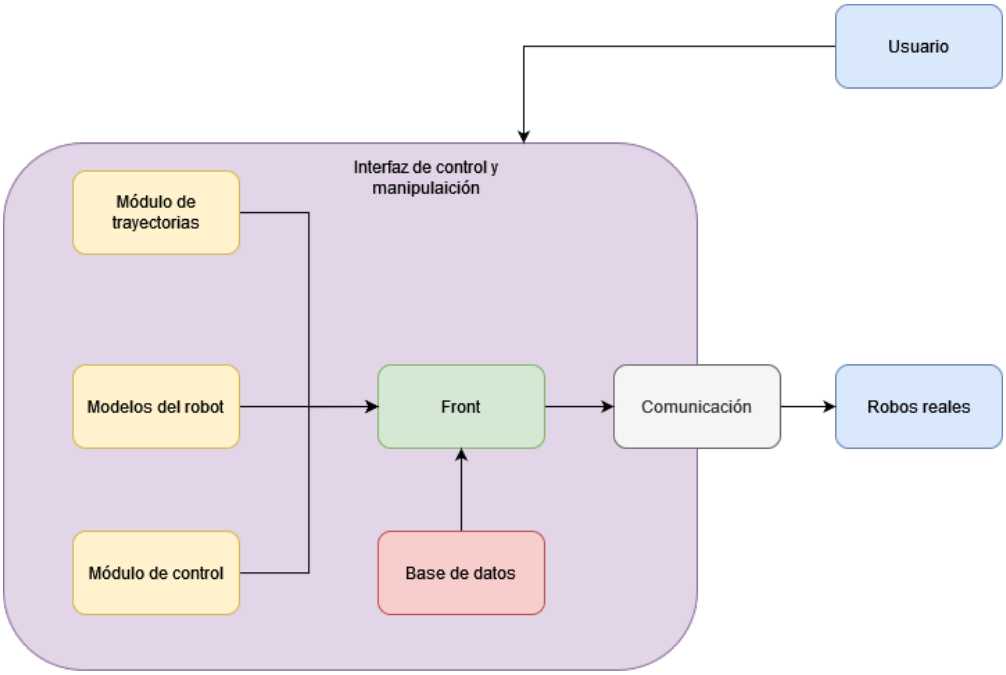


Figura 18: Arquitectura de la interfaz propuesta (fuente: Propia)

- **Front**

  El *front* es la parte visible de la aplicación, ahí se encuentran las pestañas con las que el usuario va a interactuar. Estas pestañas son *espacio articular*, *espacio cartesiano*, *base de datos*, *trayectorias libres*, *conexión*. Adicionalmente, el operario puede observar el gemelo digital dentro de un entorno virtual. En el *espacio articular*, el operario puede generar las trayectorias articulares con *sliders*. En el *espacio cartesiano*, el operario puede generar las trayectorias tridimensionales, para esto cuenta con un '*EditText*' para cada valor (x,y,z,rotx, roty, rotz). En la *base de datos*, el operario tiene las opciones de cargar, guardar, eliminar y modificar las trayectorias. En *trayectorias libres*, el operario tiene trayectorias predetermi-

nadas con una complejidad mayor que se le pueden cargar al robot. Finalmente, en *conexión*, el operario se conecta o desconecta al controlador que gestiona el robot.

- **Base de datos**

  Los datos de la base de datos se almacenan en un archivo local. La estructura general de la base de datos consta de cuatro campos: trayectoria, nombre, descripción y código. El campo de **trayectoria** guarda las filas de valores que aparecen en el "*listbox*"temporal cuya función es mantener las trayectorias deseadas por el usuario en tiempo de ejecución. Cada trayectoria tiene un **nombre** y una **descripción**, que proporciona información al usuario cuando se cargan las trayectorias en la interfaz principal. Además, hay un código generado internamente por la aplicación.

  La base de datos ofrece funcionalidades como cargar, guardar, eliminar y modificar los campos mencionados anteriormente, con excepción del código, que es administrado internamente. Estas funciones permiten al usuario acceder a una base de datos que le brinda la capacidad de manipular los datos de las trayectorias de manera conveniente.

  En resumen, la base de datos se distingue por su capacidad para organizar, almacenar y gestionar eficientemente las trayectorias de un robot. Esto proporciona al usuario accesibilidad, flexibilidad y seguridad en la manipulación de los datos, optimizando así el rendimiento y la eficacia del sistema en general.

- **Comunicación**

  Este módulo contiene los algoritmos necesarios para comunicar el robot con la herramienta software. La comunicación se realiza a través del protocolo UDP, protocolo de datagramas de Usuario (UDP, por sus siglas en inglés) que es un protocolo de comunicación en la capa

de transporte del modelo OSI. A diferencia del Protocolo (TCP), UDP se caracteriza por ser un protocolo de entrega de datos no orientado a la conexión y sin confirmación de recepción. Es el encargado de enviar los datos de las trayectorias al controlador del robot y recibir los valores de las posiciones en este instante de muestreo. Además, codifica y decodifica las cadenas de datos que se transmiten entre los dispositivos conectados.

Para establecer comunicación entre la Raspberry Pi3 y el ordenador encargado de ejecutar la aplicación en Unity 3D, se ha diseñado una arquitectura que usa el protocolo UDP dentro de su modelo de implementación.

Para el caso de la aplicación se han seguido las disposiciones que provee la pila TCP/IP (ver 19), con el cambio de TCP a UDP en la capa de *transporte*.

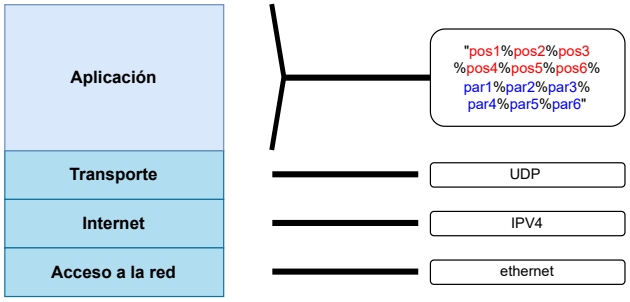


Figura 19: Modelo de la pila TCP/IP

La arquitectura de comunicación (ver figura 20) ha dejado la Raspberry como el servidor UDP y el ordenador como el cliente UDP. Este servidor se encuentra a la espera de la llegada de los valores angulares enviados desde el ordenador, a los cuales debe llegar el robot en cada instante de tiempo. Una vez esto ocurre, la Raspberry envía la acción de control, para luego enviar los valores de las posiciones angulares de cada articulación de vuelta al ordenador.

Esta comunicación bidireccional ocurre en una fracción de cada instante de tiempo.

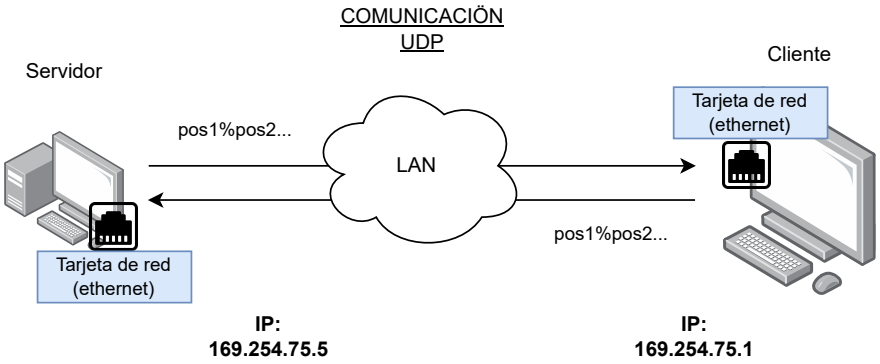


Figura 20: Arquitectura de comunicación

- **Módulo de trayectorias**

  El módulo de trayectorias transforma los valores ingresados o asignados por el usuario en trayectorias de quinto grado. Estas funciones generan variables que contienen las seis trayectorias ('th1 th2 th3 th4 th5 th6'), ya sea en coordenadas articulares o cartesianas, con base en las posiciones iniciales y finales proporcionadas.

- **Modelos del robot**

  Los modelos matemáticos como el MGI, MGD y MDI se implementan como funciones que calculan las relaciones entre las variables articulares y cartesianas de un robot. Estas funciones permiten obtener las posiciones articulares a partir de las posiciones y rotaciones cartesianas (en el caso del MGI), o viceversa (en el caso del MGD). Por ejemplo, para el MGI, la función toma como entrada las posiciones y rotaciones cartesianas deseadas y calcula las configuraciones articulares correspondientes. Es decir, dada una posición y rotación en el espacio de trabajo, la función MGI determinará los ángulos o valores articulares requeridos para que el robot alcance esa posición y orientación. De otra parte, la codificación

del modelo dinámico inverso (MDI) es un aporte importante de este trabajo, ya que todo el modelo matemático del robot se define al interior del computador, con lo cual se puede implementar un controlador basado en el modelo, en este caso un CTC.

En cuanto a los algoritmos asociados al gemelo digital, estos son algoritmos utilizados para simular el comportamiento de un robot en un espacio virtual antes de implementarlo en el robot real. Estos algoritmos permiten probar y optimizar el comportamiento del robot en un entorno seguro y controlado, evitando posibles daños o accidentes en el entorno real.

- **Módulo de control**

  El módulo de control desempeña una función crucial al recibir los valores de posición en cada instante de tiempo. Realiza cálculos precisos para determinar el esfuerzo de control necesario con el fin de guiar al robot hacia la posición deseada a lo largo de la trayectoria. Este proceso se lleva a cabo de manera eficiente y precisa, asegurando un control óptimo y fluido del robot en todo momento. Gracias a este módulo, se logra una coordinación precisa y suave del robot a medida que se desplaza siguiendo la trayectoria definida, brindando resultados confiables y de alta calidad en la ejecución de tareas.

4.4.2. Diagrama de componentes de la aplicación

Este diagrama representa como cada instancia de clase accede a los recursos proporcionados por otros módulos y/o paquetes. Por ejemplo, la capa de presentación contiene componentes de los módulos de base de datos, gemelo digital, panel cartesiano, panel articular. Este a su vez tiene accede a un módulo externo denominado como 'Comunicación', el cual funciona como una interfaz, pues solo se implementan sus métodos.

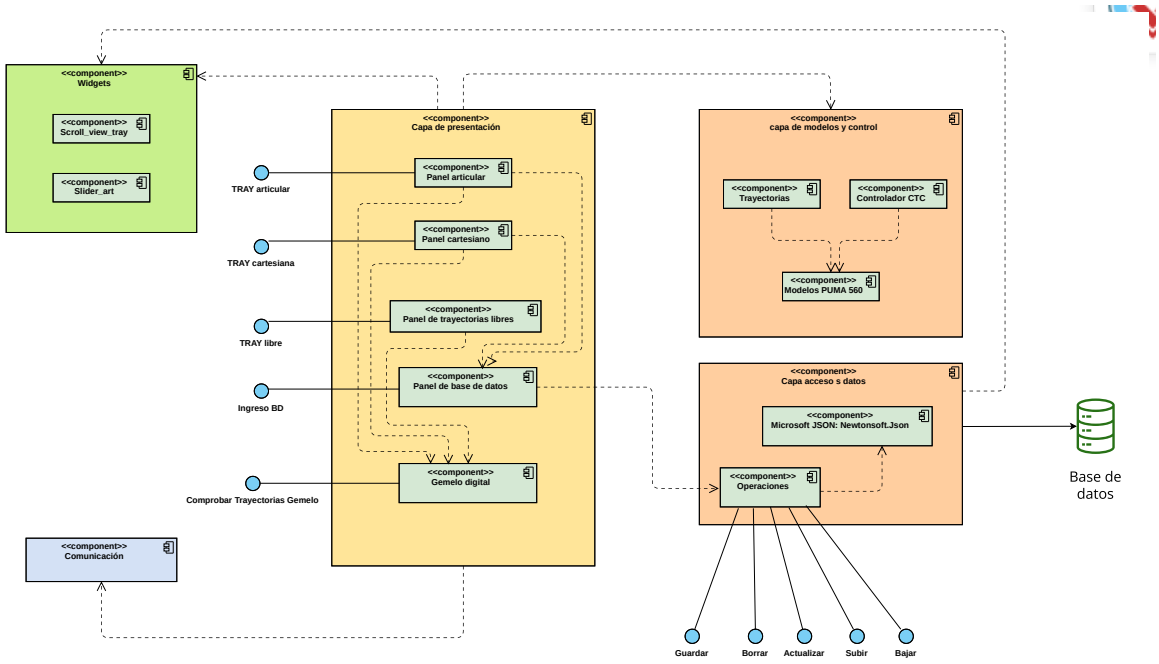


Figura 21: Diagrama de componentes de la aplicación

4.4.3. Diagrama de clases

Las figuras 22 23 y 24, presentan los diagramas de clases de los módulos generales de la aplicación. En estos diagramas se puede comprender el flujo que sigue la información dentro de la aplicación, desde la generación hasta el guardado de la misma.

Por ejemplo, en la figura 22 se muestra el diseño del panel articular y cartesiano, pues tienen los mismos patrones de diseño salvo en la disposición del '*front*'. En la figura 23 se muestra el diseño del panel de las trayectorias predefinidas, el cual describe el acceso de este módulo a los distintos paquetes y clases. En la figura 24 se muestra el diseño del panel de la base de datos, el cual describe el acceso de este módulo a los módulos que utilizan el entorno .Net para acceder a la lectura y escritura de los archivos.

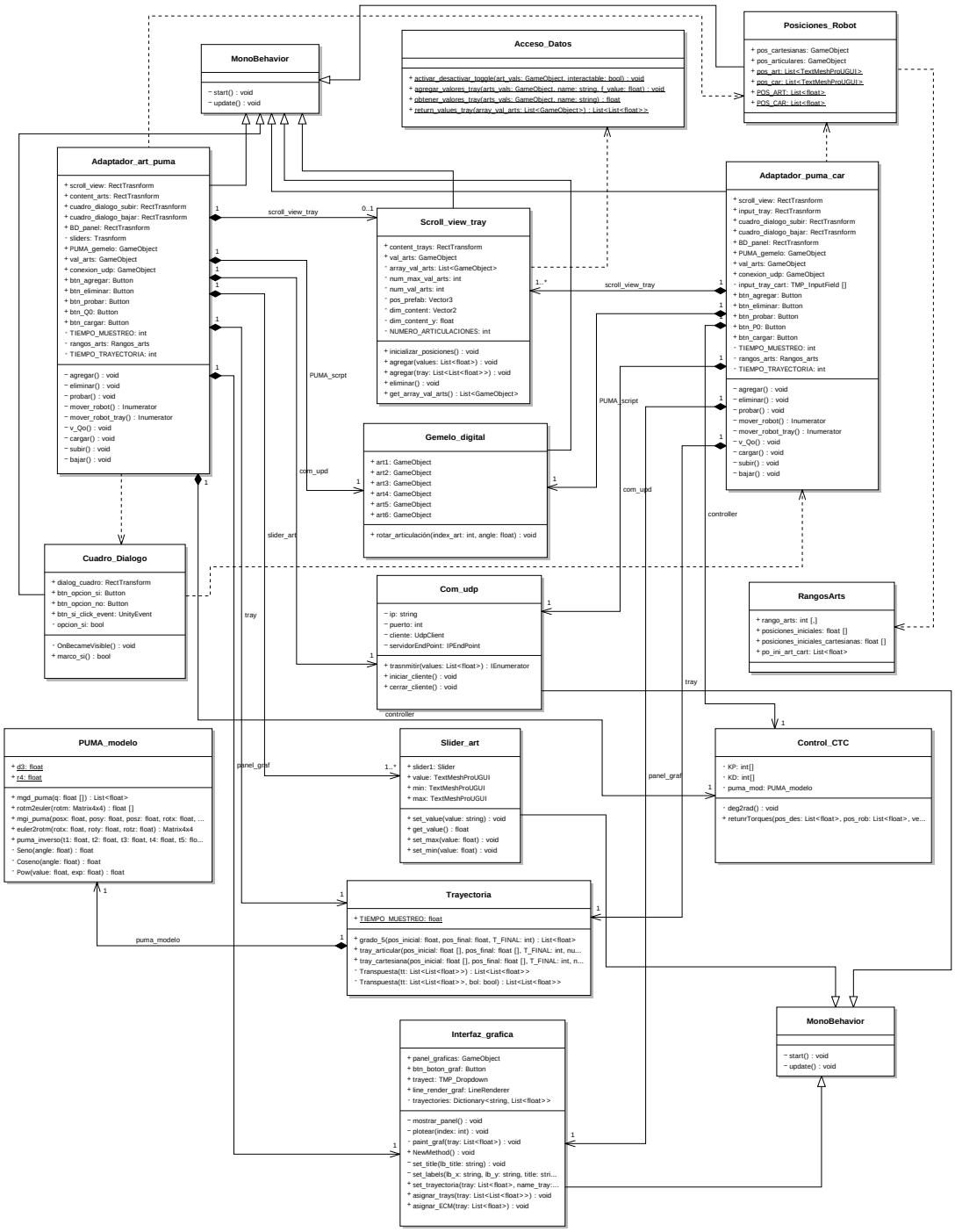


Figura 22: Diagrama de clases: panel articular y cartesiano

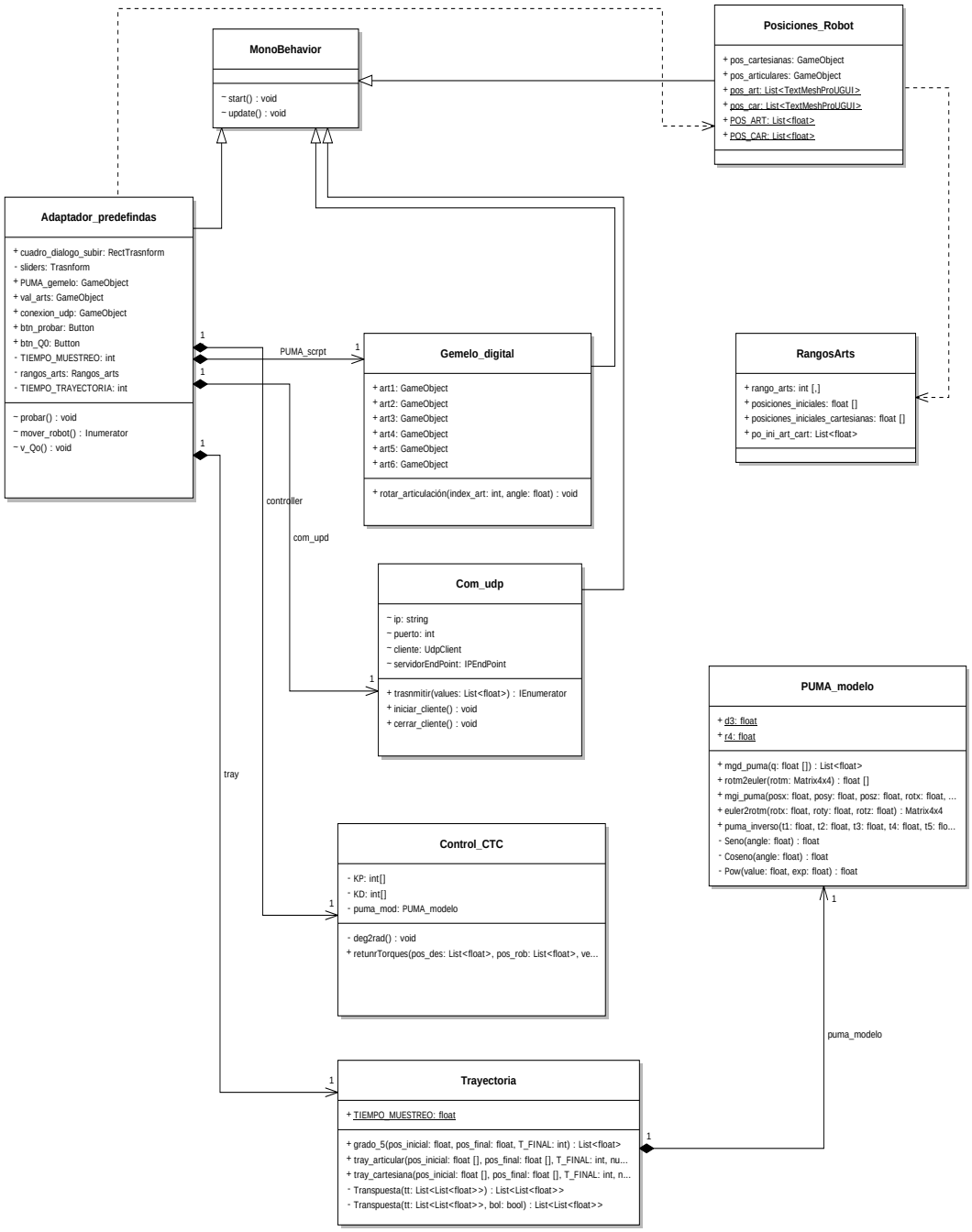


Figura 23: Diagrama de clases: panel de trayectorias predefinidas

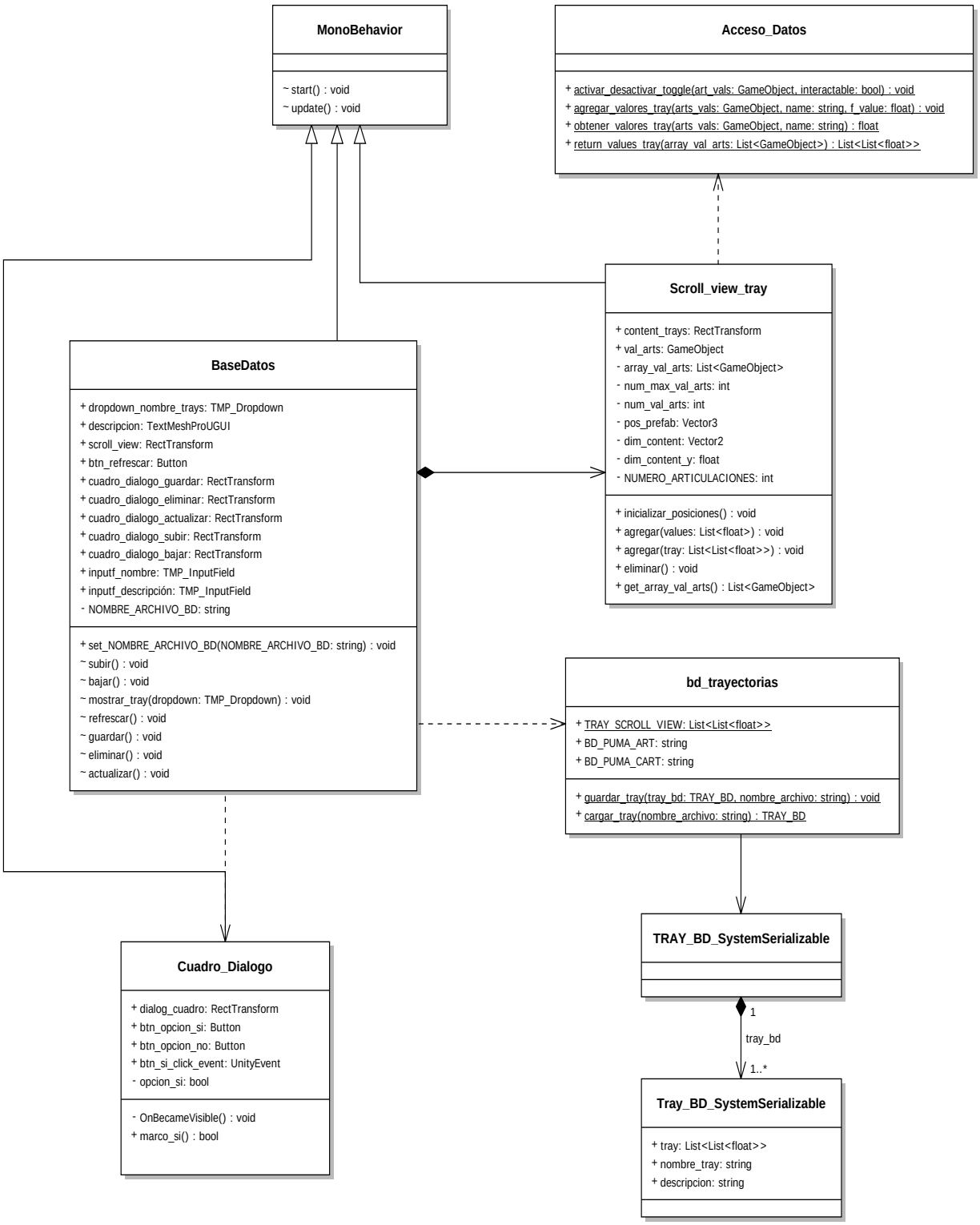


Figura 24: Diagrama de clases: panel de base de datos

4.5. Construcción de la herramienta

La solución software construida para la manipulación y control del robot PUMA real, cuenta con las características descritas en los diseños y los requerimientos mostrados anteriormente.

Inicialmente, se tiene que cuenta con 4 '*frames*' o paneles: articular, cartesiano, base de datos, y trayectorias preestablecidas.

- **Articular**

  Este panel (ver figura 25) cuenta con 6 '*sliders*' que permiten mover cada articulación de forma independiente. El usuario tiene la opción de mover los deslizadores individualmente o en conjunto, dentro de un rango específico, y luego activar el movimiento del robot mediante un botón designado. Además, hay un botón que devuelve el robot a su posición inicial.

  Adicionalmente, se incluyen dos botones que permiten guardar y/o modificar trayectorias en la base de datos, así como cargar una trayectoria completa en el robot.

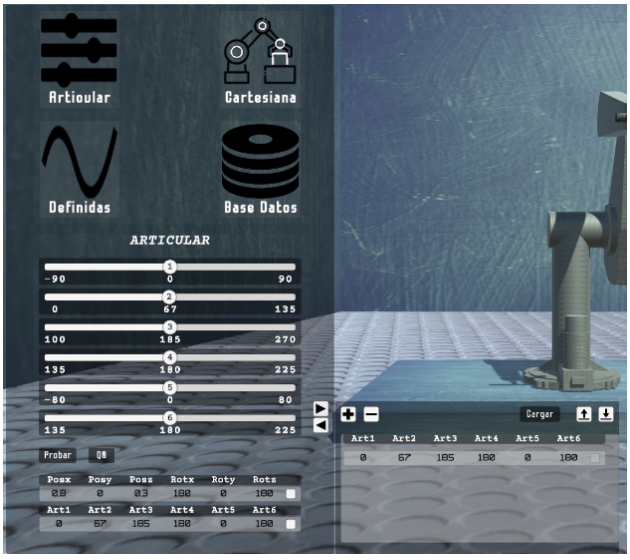


Figura 25: Panel articular

- **Cartesiano**

  En el panel cartesiano (ver figura 26) se encuentran 6 '*editText*' o editores de texto, los cuales representan las posiciones en cada uno de los ejes (x, y, z) y las rotaciones (rotx, roty, rotz). Al igual que en el panel anterior, se dispone de un área para cargar trayectorias en el espacio cartesiano y enviarlas al pánel encargado de gestionar la base de datos.

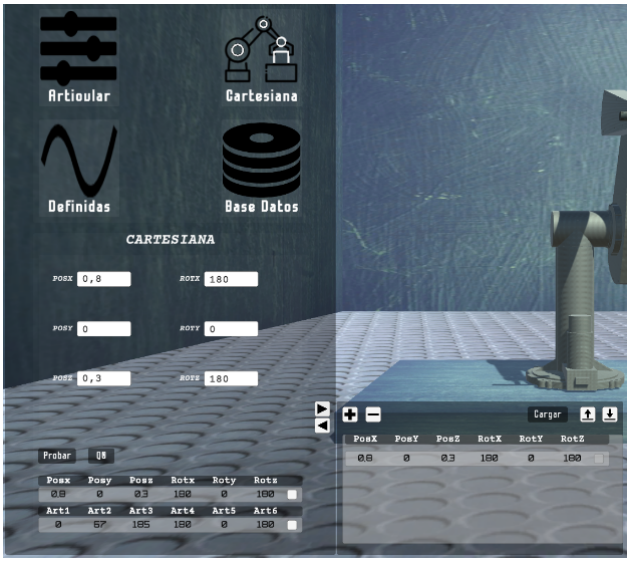


Figura 26: Panel cartesiano

- **Trayectorias preestablecidas**

  En este panel se han definido dos tipos de trayectorias predefinidas: circulares y sinusoidales.

  En la primera interfaz (como se muestra en la figura), se puede generar un círculo con un centro en el punto $p(x_1, y_1, z_1)$ y un radio r_1. El programa utiliza estos valores para ejecutar un algoritmo que crea una trayectoria alrededor del eje x, es decir, el círculo se forma en un plano definido por los ejes z e y.

  En la segunda interfaz, se puede generar una trayectoria sinusoidal. Para ello, el usuario debe proporcionar un punto cartesiano inicial, un punto cartesiano final y una frecuencia que debe

estar dentro de ciertos rangos predefinidos. Similar al caso anterior, esta trayectoria se traza a lo largo del eje x, pero varía a lo largo de los ejes x e y.

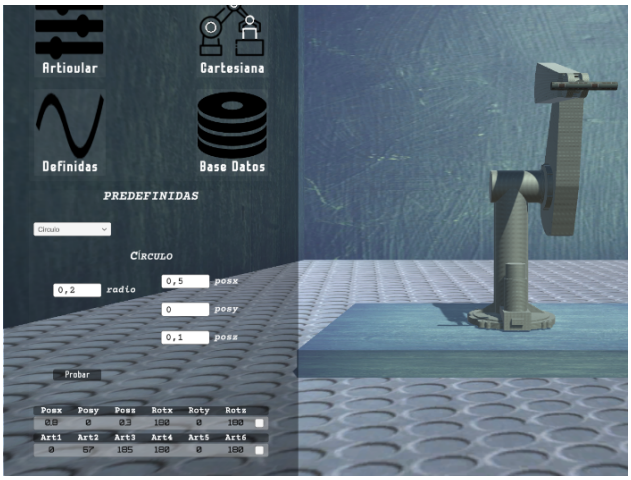


Figura 27: Panel trayectorias predefinidas

- **Base de datos**

  En este panel se pueden gestionar las trayectorias almacenadas en los archivos locales del programa con formato *.json*. Estos archivos se denominan *trayectorias articulares y trayectorias cartesianas*.

  El panel se divide en dos secciones: **base de datos articular**, figura 29 y **base de datos cartesiana**, figura 28. En cada una de estas secciones se guardan las trayectorias generadas desde los paneles articulares y cartesianos, respectivamente. Además, es posible actualizar y eliminar las trayectorias almacenadas.

- **Comunicación**

  Los algoritmos para la implementación de la comunicación UDP entre la Raspberry PI3 y el ordenador se muestran en el (anexo: 69). Estas líneas de código describen el proceso mediante el cual se envían y se reciben paquetes de datos desde el lado del ordenador al

Figura 28: Panel base de datos, trayectorias cartesianas

Figura 29: Panel base de datos trayectorias articulares

dejar este como cliente UDP.

Cabe resaltar que a excepción del panel de la base de datos, en todos los paneles se pueden

evidenciar las posiciones del robot en tiempo real.

Además, cada panel tiene acceso a las clases que contienen las funciones encargadas de calcular las trayectorias de quinto grado. Estas funciones a su vez, utilizan los modelos del robot PUMA para generar dichas trayectorias. Por otro lado, la clase de 'control' utiliza el modelo dinámico inverso del robot PUMA para determinar el torque requerido en cada una de sus articulaciones.

Los algoritmos elaborados en los módulos: interfaz, control, modelos, trayectorias, y comunicación, se presentan en la sección de anexos al final del documento (ver 67 68 70 71). La carpeta que contiene todo el proyecto se pueden encontrar en el siguiente enlace de GitHub.

4.5.1. Instalación

El software cuenta con la capacidad de establecer una conexión con el robot a través del protocolo UDP. En términos simples, la aplicación envía el torque o par calculado y las posiciones necesarias para que cada articulación del robot alcance las trayectorias deseadas o creadas por el usuario, y recibe el valor angular de cada articulación en un instante de tiempo.

Debido a la arquitectura de comunicación física entre el ordenador y la Raspberry PI, en necesario asignar una dirección IP estática tanto en la Raspberry PI como en el ordenador. Con lo anterior, los adaptadores de red de cada dispositivo se podrán comunicar mediante el protocolo antes mencionado.

Por otra parte, el software viene en una carpeta comprimida en zip, que el usuario debe descomprimir. Dentro de este hay un archivo con extensión .exe llamado *PumaSoft.exe*.
Finalmente, tiene que ejecutar o abrir este archivo, el cual abre la interfaz principal, una vez ahí el usuario puede navegar por todas las secciones que tiene la interfaz para el manejo y control del robot PUMA real.

Por último, los requisitos recomendados que debe tener el ordenador para correr el ejecutable '*archivo.exe*', se muestran a continuación:

- Sistema operativo: Windows 7 SP1+, macOS 10.12+, o una distribución de Linux compatible.
- Procesador: Intel Core i3 o equivalente (se recomienda un procesador más potente para proyectos más grandes o exigentes).
- Memoria RAM: Al menos 4 GB de RAM.
- Espacio en disco: 10 GB de espacio libre en el disco duro para la instalación del programa y los archivos asociados.
- Tarjeta gráfica (GPU): Se recomienda una tarjeta gráfica compatible con DirectX 11 o posterior. Algunas funciones de Unity pueden requerir una tarjeta gráfica más potente.
- Resolución de pantalla: Se recomienda una resolución de pantalla de al menos 1280x720 píxeles.

4.5.2. Instrucciones de uso

La manipulación del robot PUMA mediante la aplicación se especifica en detalle en el respectivo manual, ver páginas: 101 - en adelante.

En este manual se detalla el uso de las pantallas como 'panel articular', 'panel cartesiano' y 'panel trayectorias predefinidas'. En estas pantallas se pueden generar varios tipos de trayectorias

como lo son 'trayectorias articulares', 'trayectorias cartesianas' y 'trayectorias predefinidas'. También detalla el uso de una base de datos asociada a una interfaz que proporciona las acciones para su accesibilidad.

5. Resultados

5.1. Resultados de simulación

5.1.1. Trayectorias articulares

Para esta prueba se ha realizado un movimiento de las 6 articulaciones como muestra la figura 30.

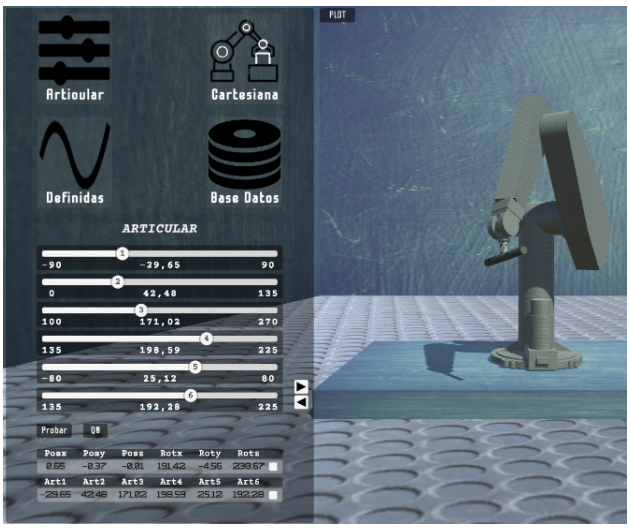


Figura 30: Prueba trayectorias articulares

Ahora los errores que se obtuvieron con ayuda de la interfaz se muestran en las figuras 31 32 33 34 34 35 36.

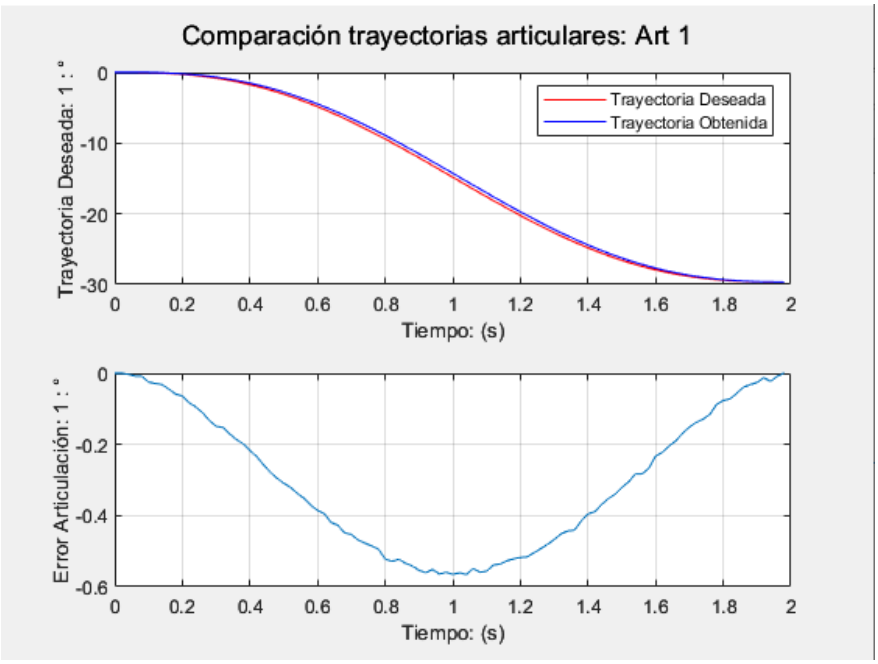


Figura 31: Gráficas comparativas articulación 1

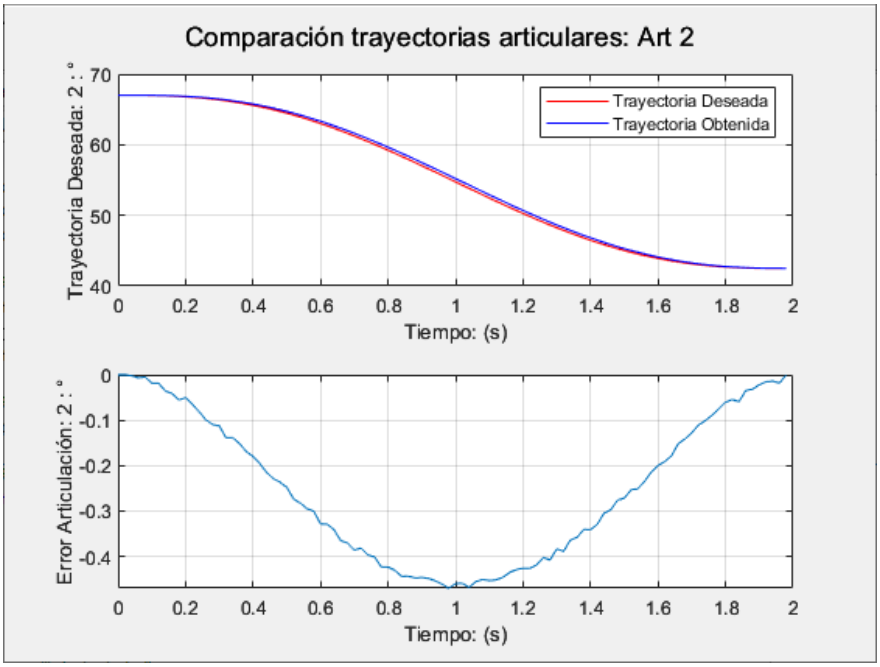


Figura 32: Gráficas comparativas articulación 2

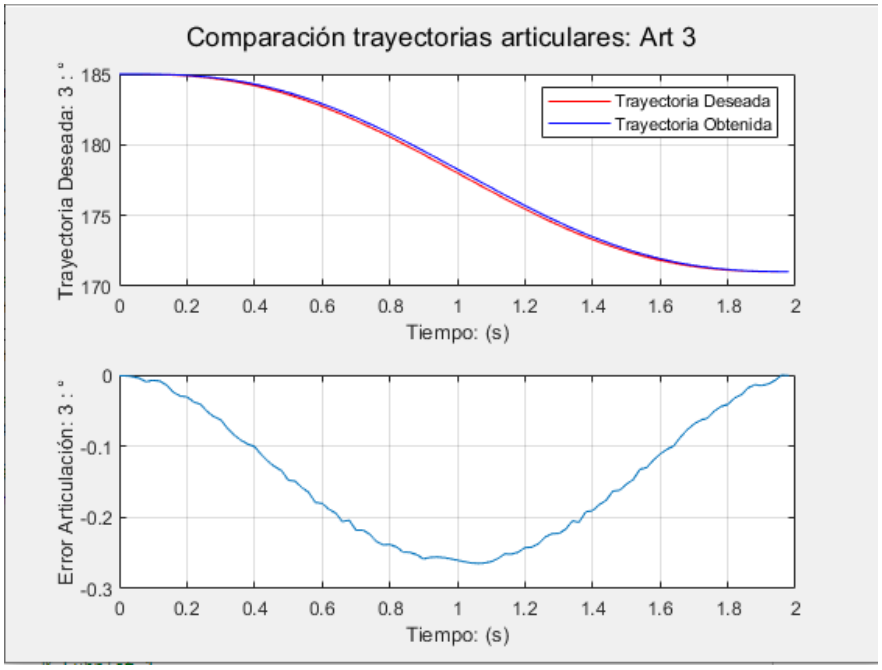


Figura 33: Gráficas comparativas articulación 3

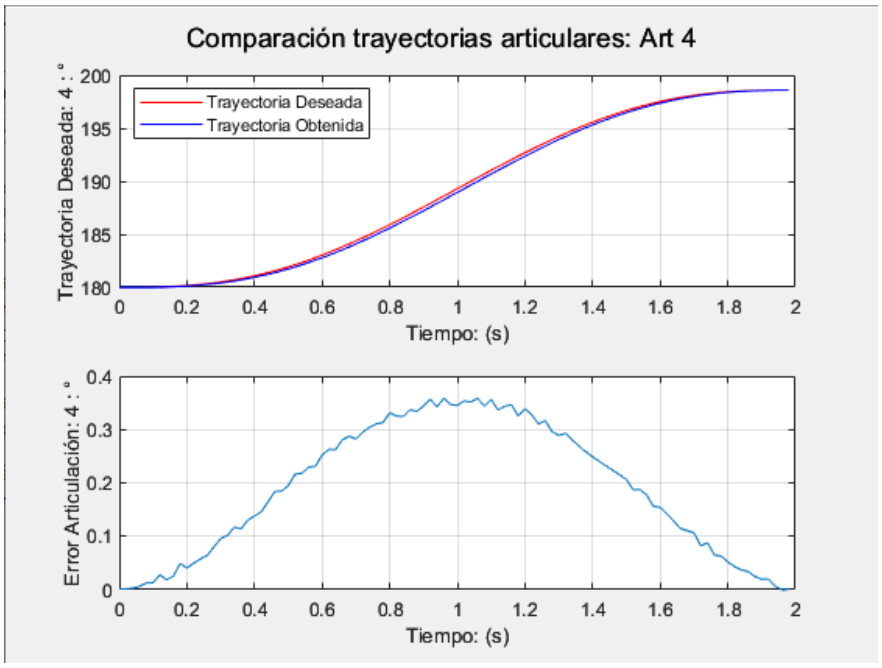


Figura 34: Gráficas comparativas articulación 4

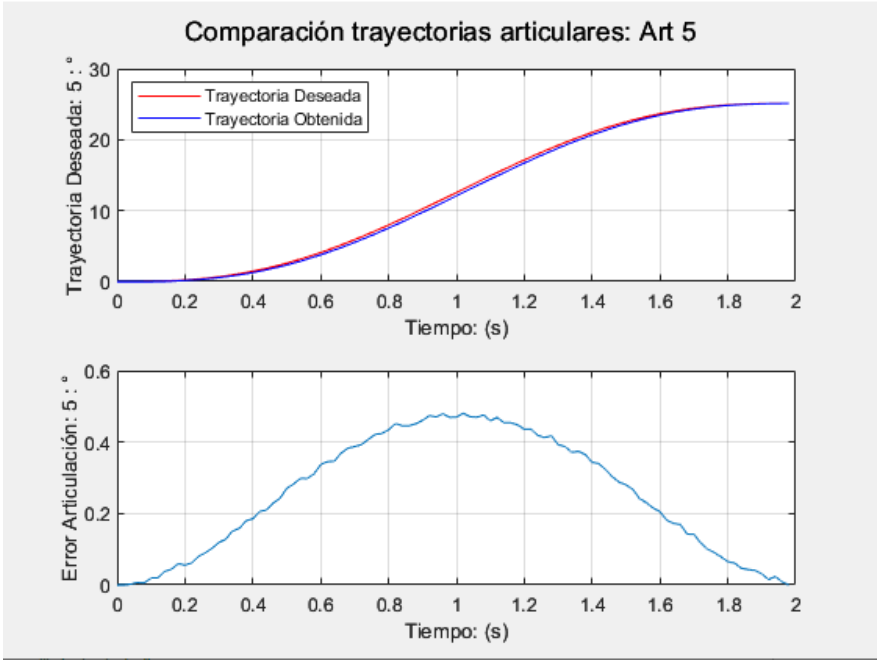


Figura 35: Gráficas comparativas articulación 5

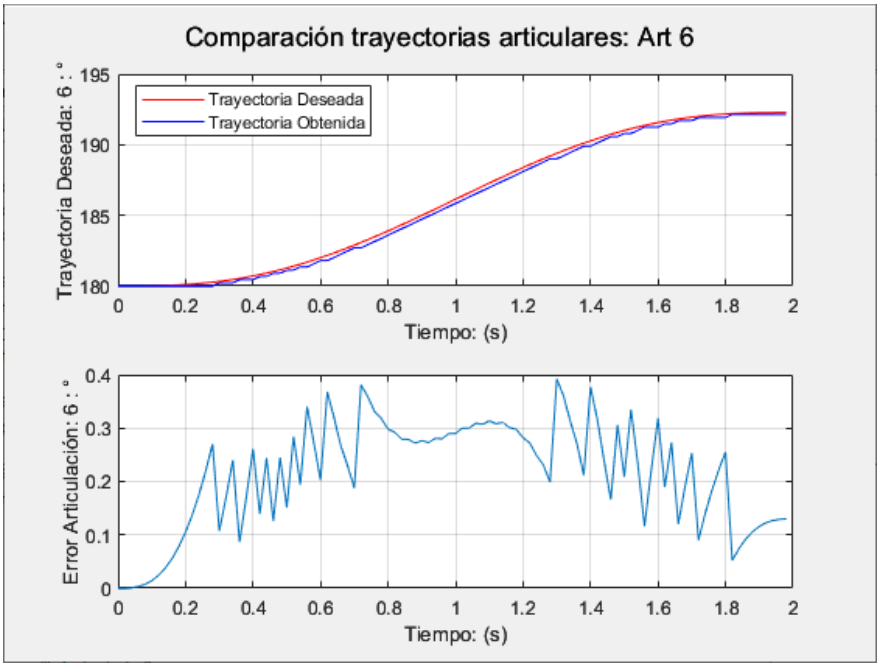
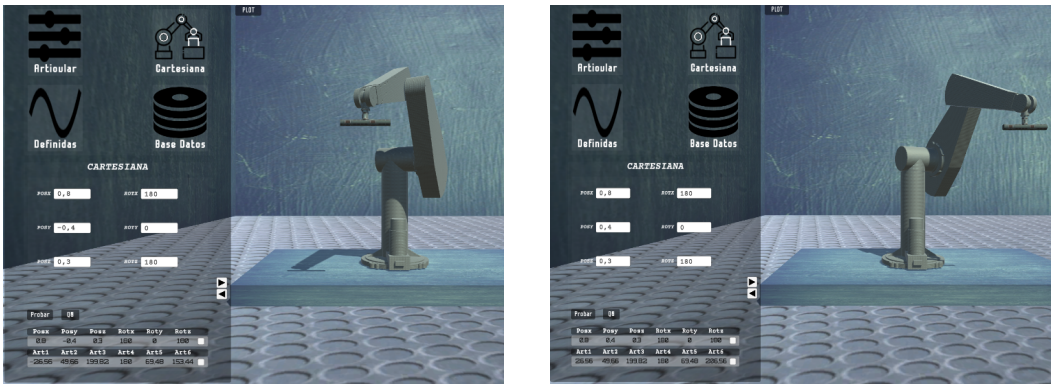


Figura 36: Gráficas comparativas articulación 6

5.1.2. Trayectorias cartesianas

Para esta prueba se ha realizado una trayectoria punto a punto manteniendo el efector final sobre una línea recta como muestra la figura 37.



(a) Gemelo digital en la posición inicial (b) Gemelo digital en la posición final

Figura 37: Posiciones iniciales y finales de la trayectoria lineal

Ahora los errores que se obtuvieron con ayuda de la interfaz se muestran en las figuras 38, 39, 40, 41, 41, 42, 43, 45, 44.

Lo importante en este caso es el comportamiento cartesiano deseado. En la siguiente gráfica se muestra la línea deseada versus la línea obtenida. Esta última parece que varía mucho en el eje z; sin embargo, la escala en este eje es muy pequeña, obteniéndose al final un error máximo de un centímetro en la parte media de la trayectoria.

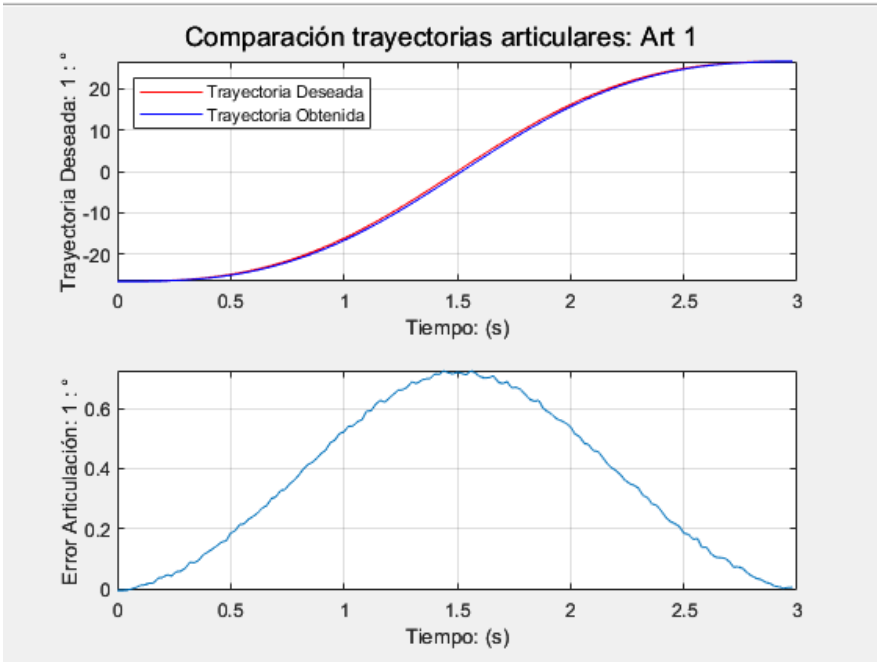


Figura 38: Gráficas comparativas articulación 1

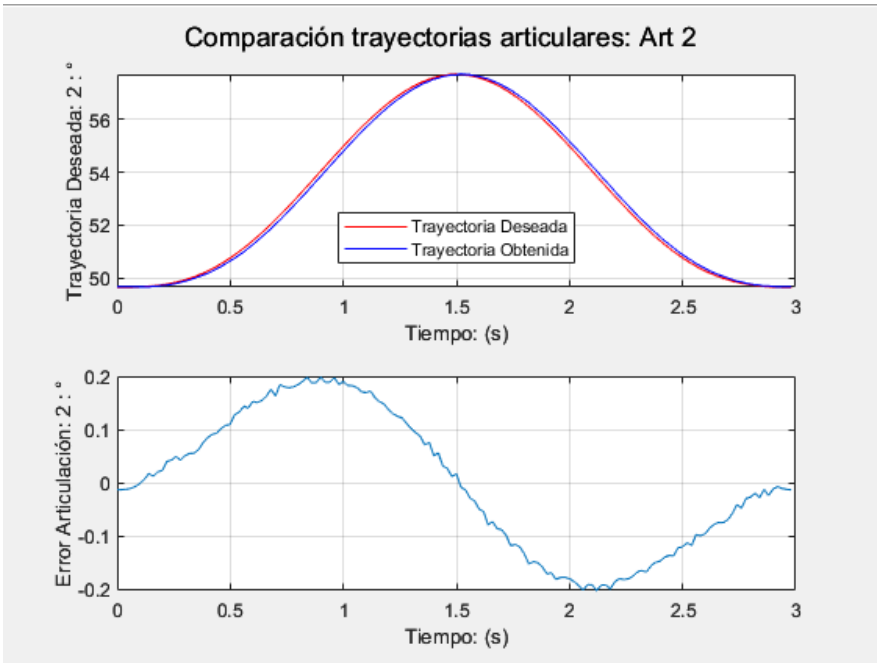


Figura 39: Gráficas comparativas articulación 2

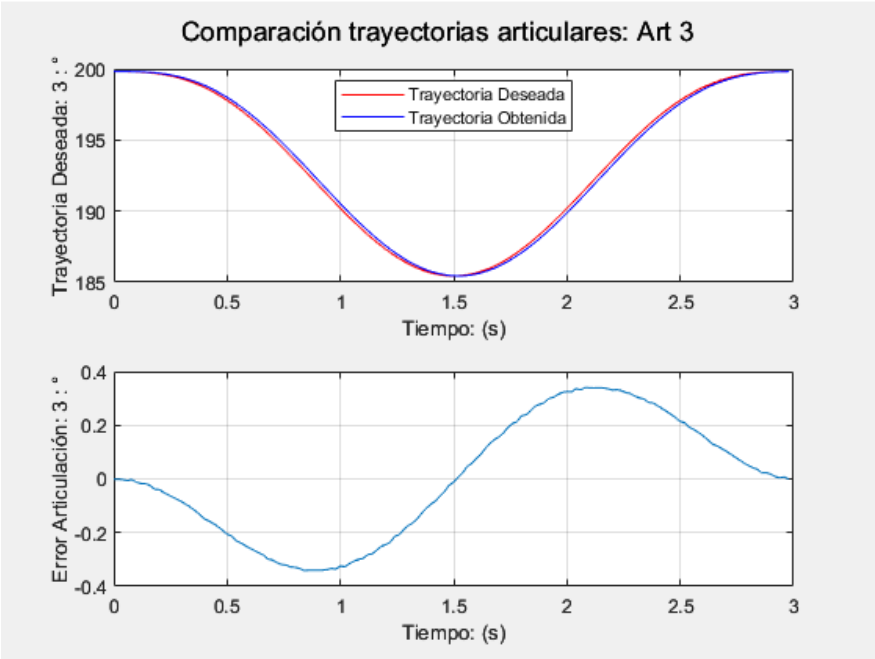


Figura 40: Gráficas comparativas articulación 3

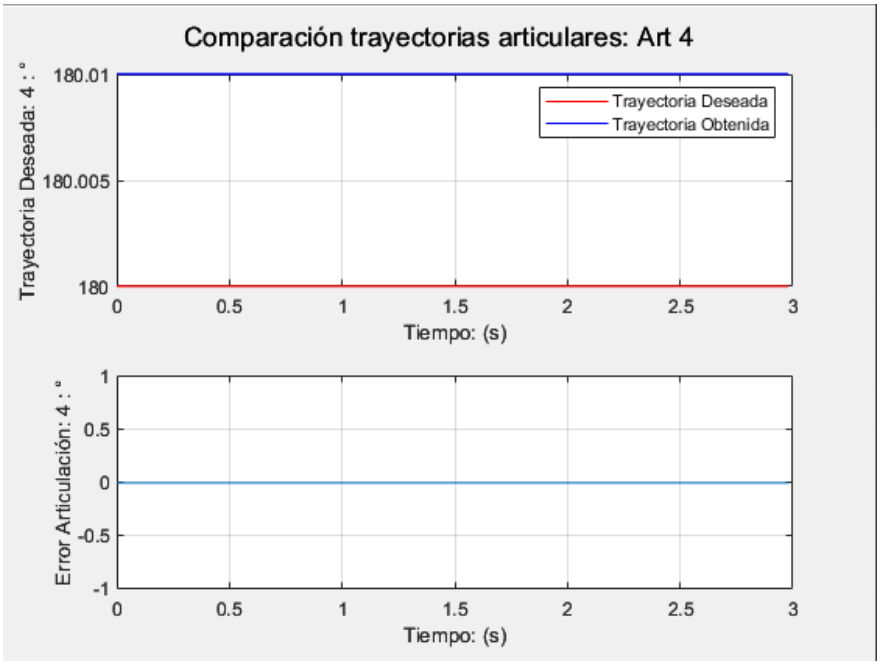


Figura 41: Gráficas comparativas articulación 4

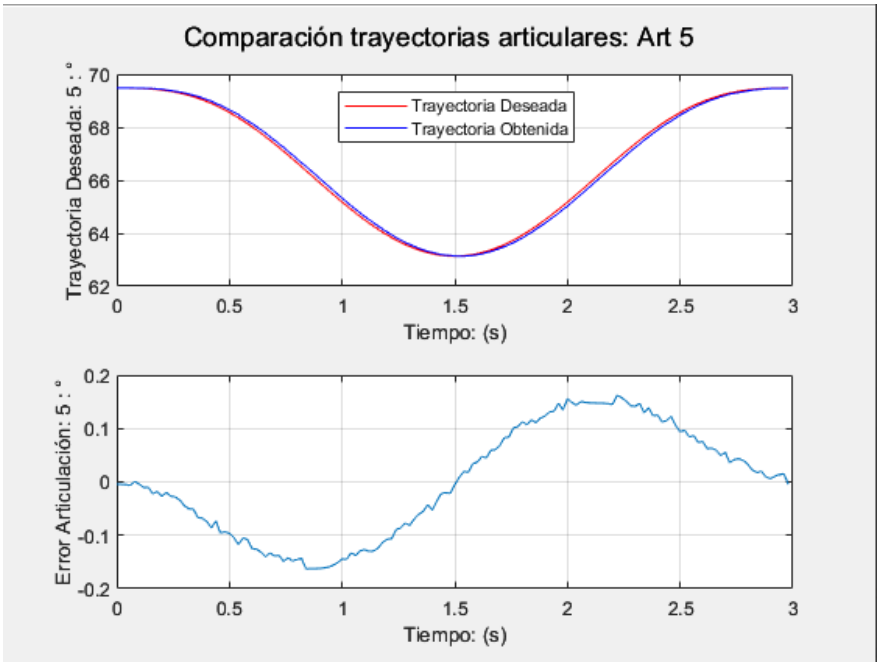


Figura 42: Gráficas comparativas articulación 5

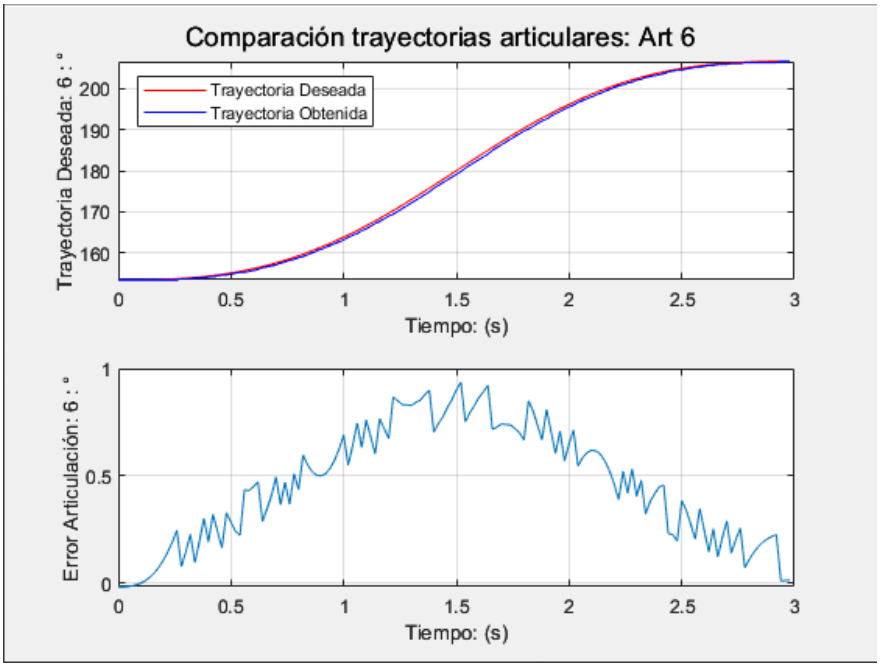


Figura 43: Gráficas comparativas articulación 6

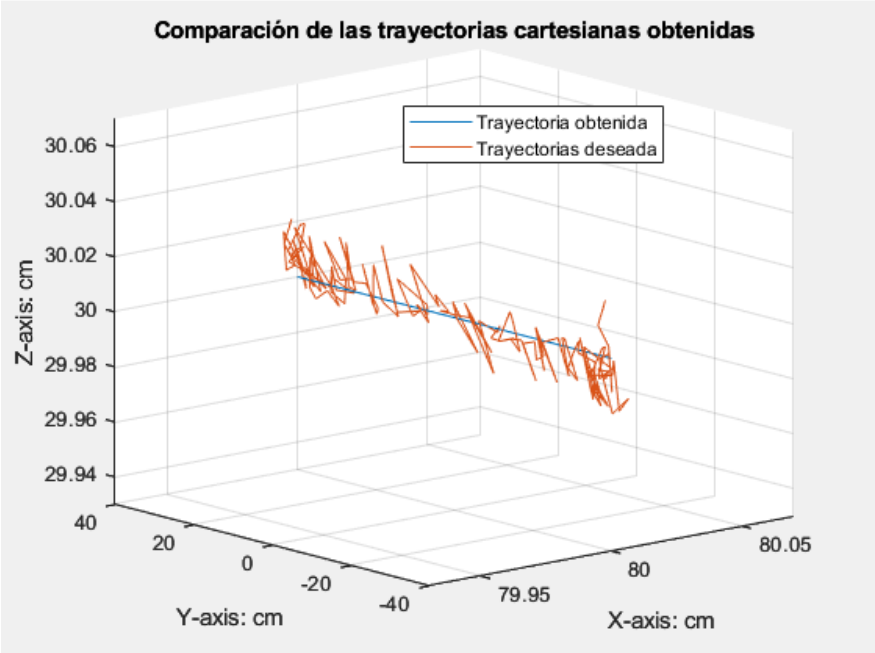


Figura 44: Comparativa entre la trayectoria obtenida y deseada

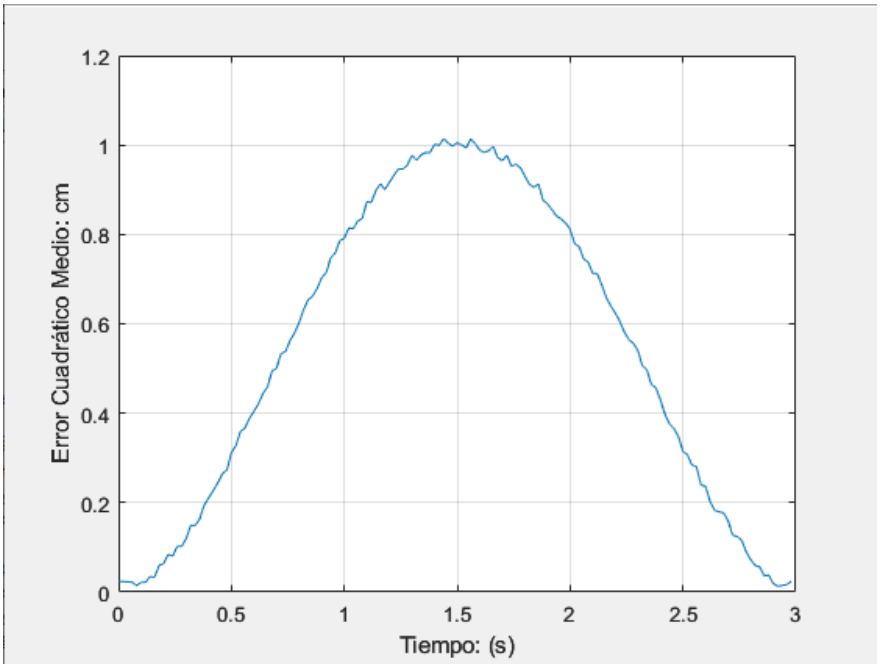


Figura 45: Error euclidiano obtenido

5.1.3. Trayectorias predefinidas

Para esta prueba se ha realizado una trayectoria circular sobre el eje x del robot, manteniendo una rotación fija en las articulaciones de la muñeca.

Ahora los errores que se obtuvieron con ayuda de la interfaz se muestran en las figuras 46, 47, 48, 49, 49, 42, 51, 53, 52.

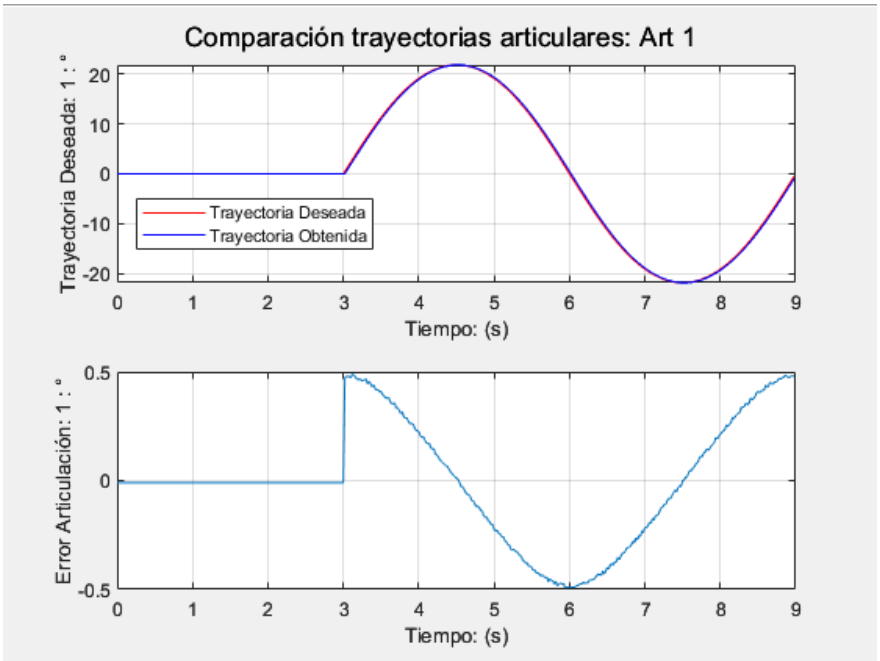


Figura 46: Gráficas comparativas articulación 1

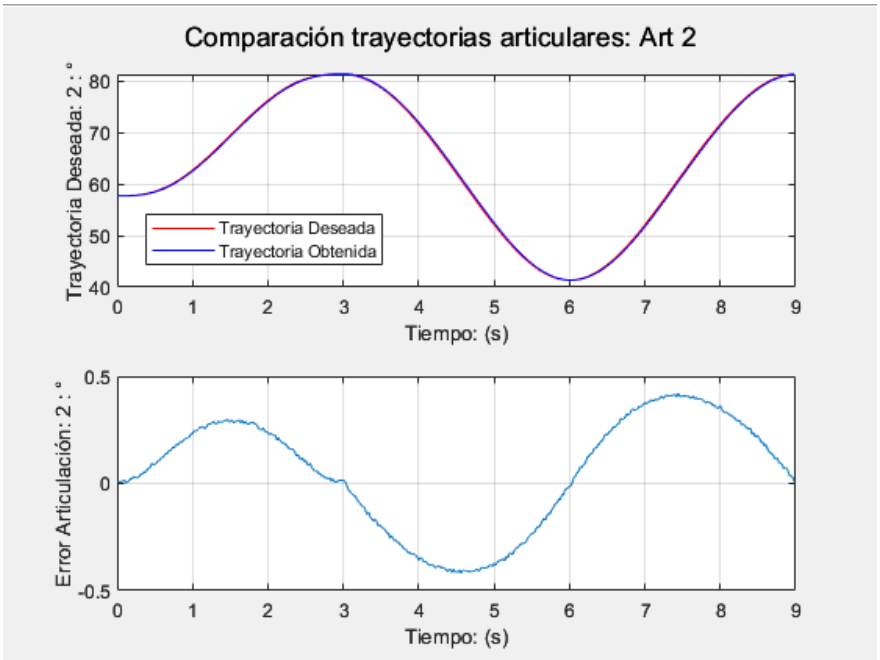


Figura 47: Gráficas comparativas articulación 2

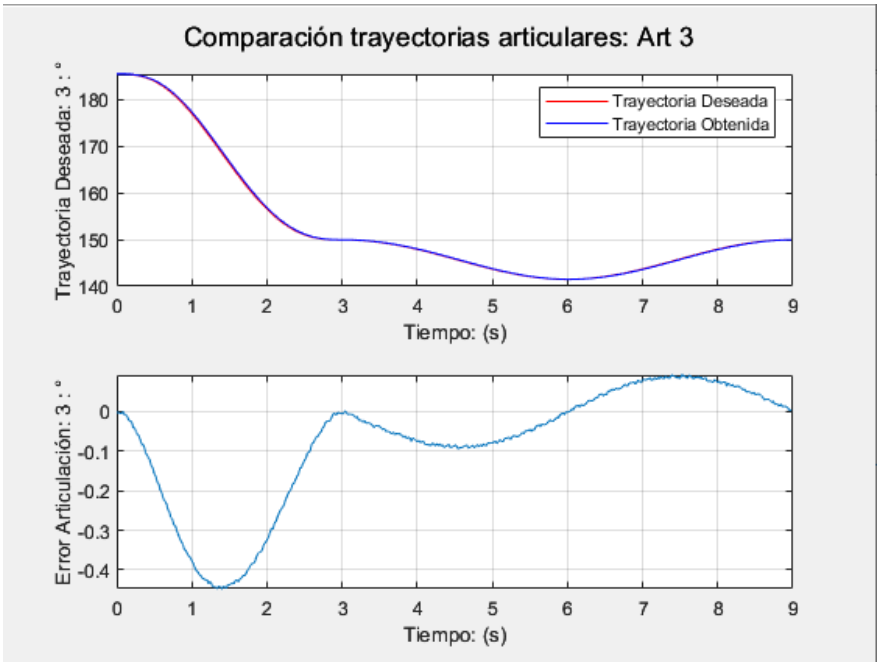


Figura 48: Gráficas comparativas articulación 3

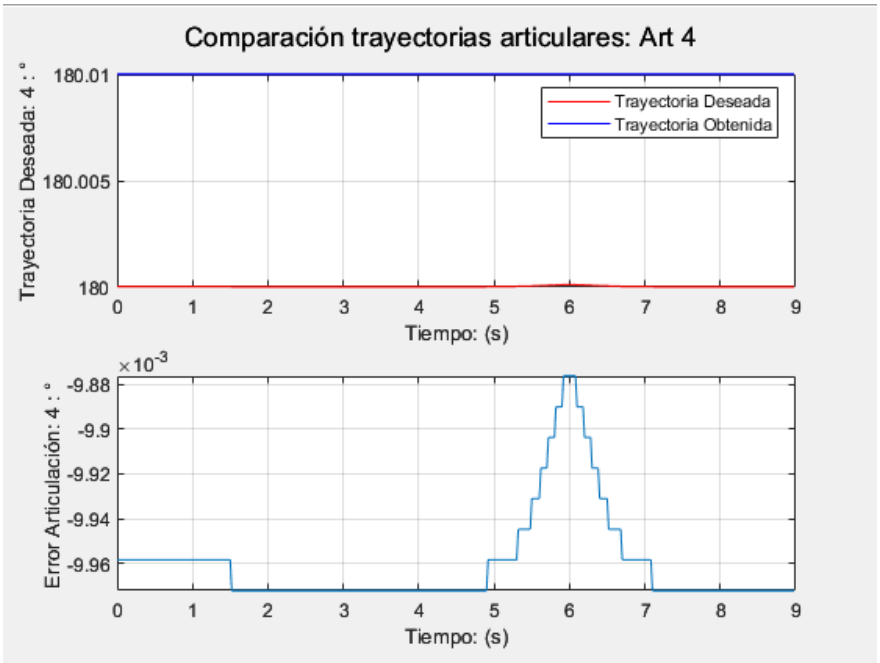


Figura 49: Gráficas comparativas articulación 4

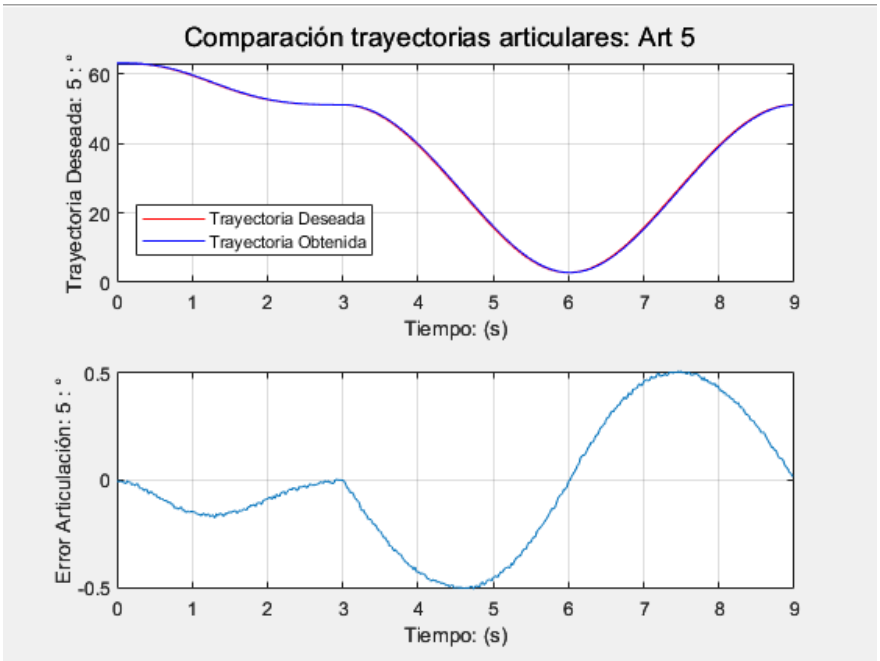


Figura 50: Gráficas comparativas articulación 5

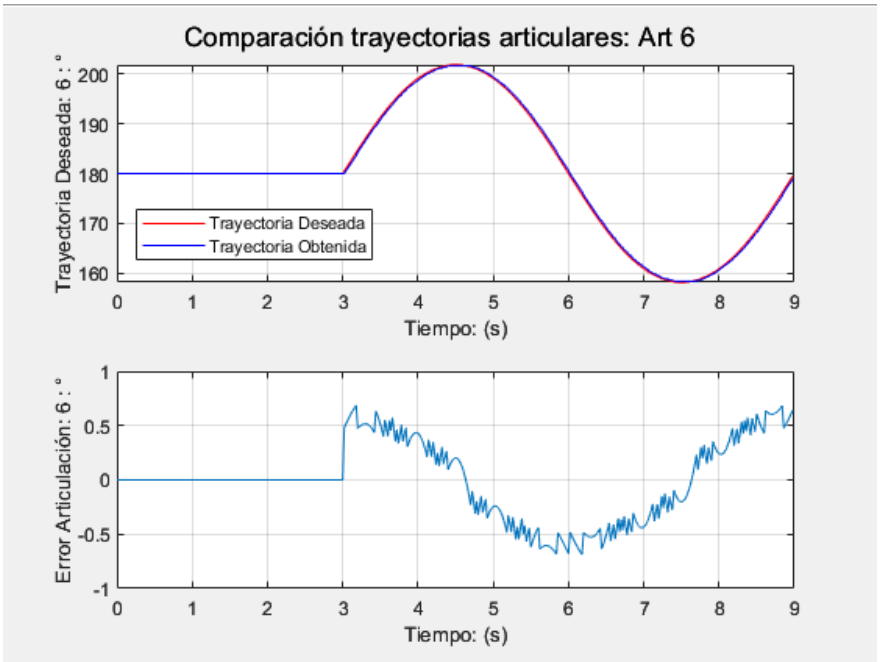


Figura 51: Gráficas comparativas articulación 6

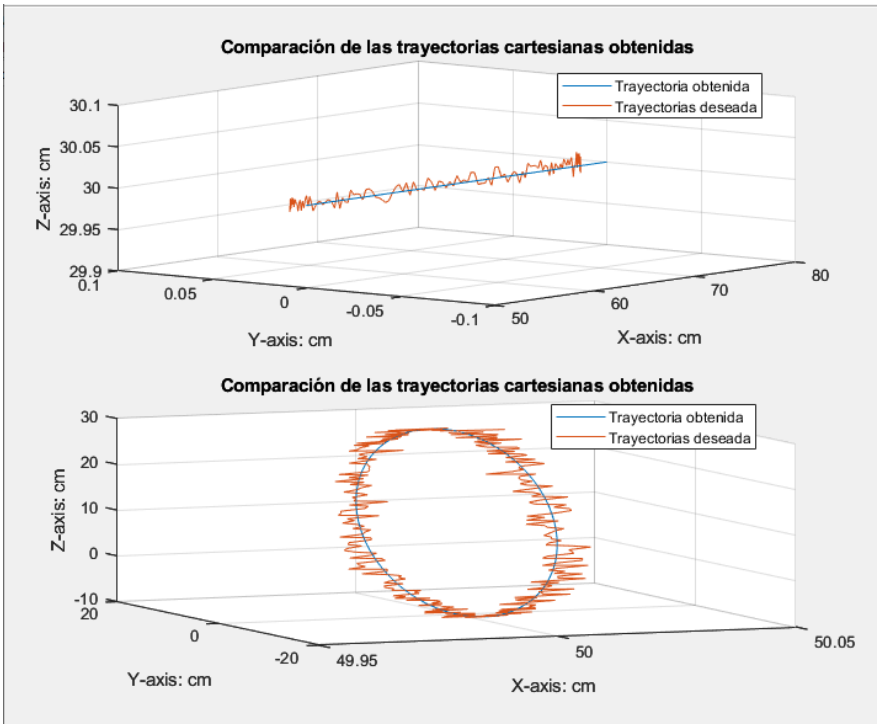


Figura 52: Comparación entre las trayectorias cartesiana deseada y obtenida

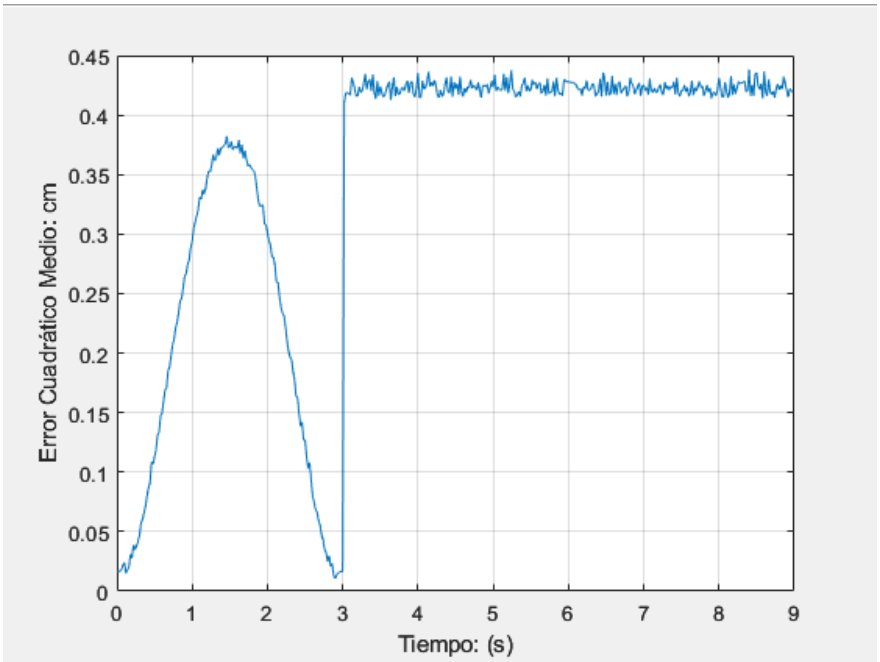


Figura 53: Error euclidiano obtenido trayectoria total

5.2. Resultados con el robot real

Para estas pruebas se han realizado una serie de trayectorias en las cuales se analizan las articulaciones $(J1, J2$ y $J3)$, ya que la rotación de la muñeca no cambia, por simplicidad, y por características del hardware (el sensor de la articulación 5 no funciona). Las gráficas muestran una similitud con las simulaciones que se hicieron desde la misma interfaz.

5.2.1. Trayectorias articulares

Para la prueba en el espacio articular se han movido tres articulaciones $(J1, J2$ y $J3)$. La articulación J1 se ha movido desde una posición de $(14$ a $0)°$, la J2 de $(54$ a $67)°$ y la J3 de $(199{,}5$ a $185)°$. Las siguientes gráficas (54, 55, 56) muestran las comparativas entre la trayectoria deseada y la trayectoria obtenida, incluida la gráfica del error de seguimiento articular.

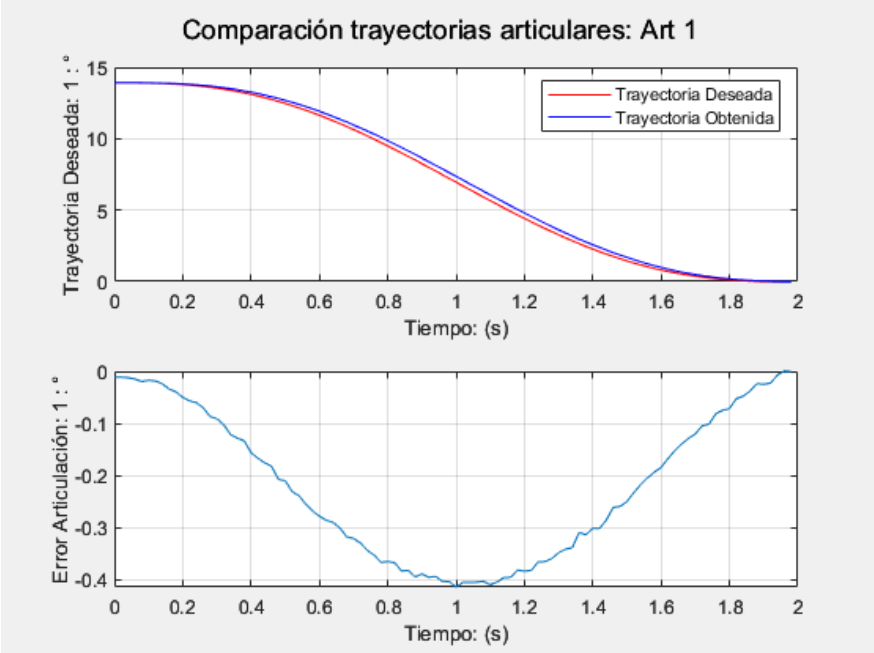


Figura 54: Gráficas comparativas articulación 1

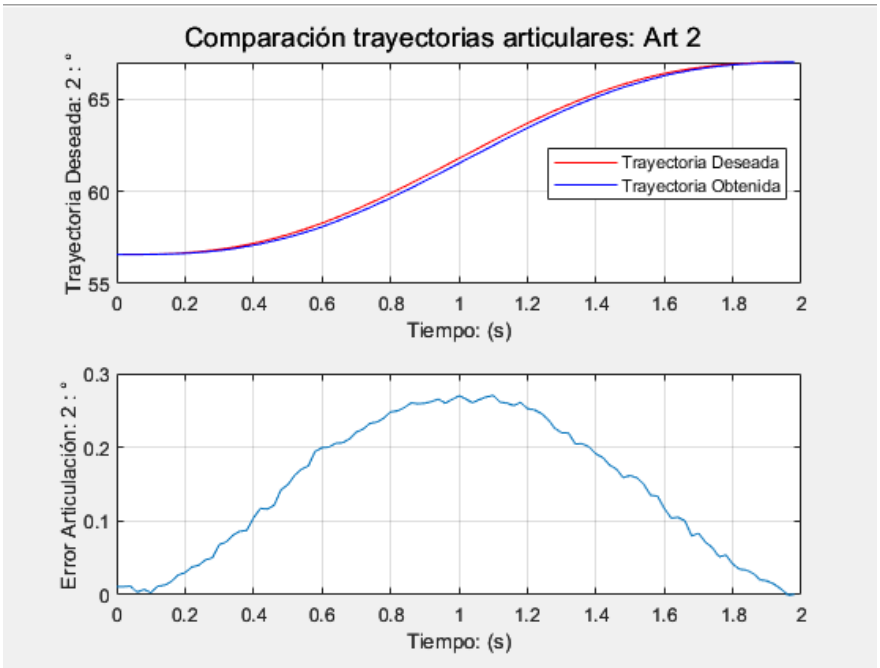


Figura 55: Gráficas comparativas articulación 2

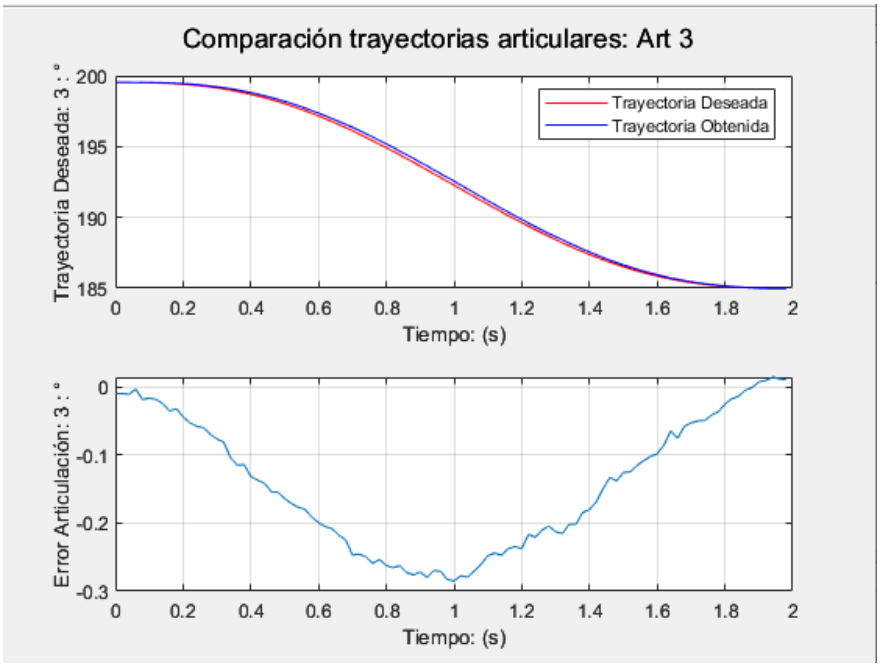


Figura 56: Gráficas comparativas articulación 3

5.2.2. Trayectorias cartesianas

Para la prueba en el espacio cartesiano se ha dispuesto de la siguiente trayectoria:

P1 = $x1 : 0{,}8m$, $y1 : -0{,}4m$, $z1 : -0{,}3m$

P2 = $x2 : 0{,}8m$, $y1 : 0{,}4m$, $z2 : -0{,}3m$

Las siguientes gráficas (57, 58, 59, 61, 60) muestran las comparativas entre la trayectoria deseada y la trayectoria obtenida, incluida la gráfica del error euclidiano.

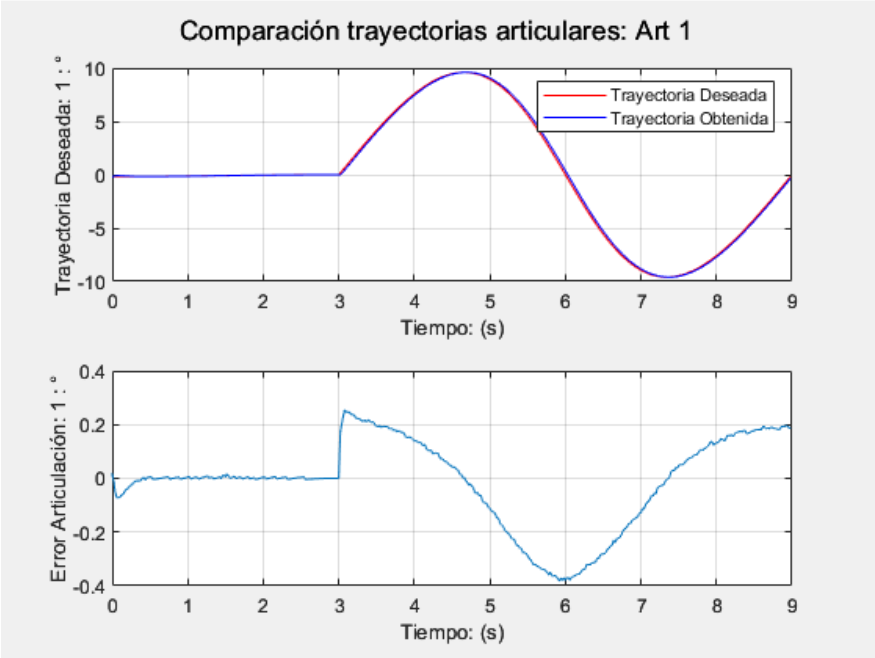


Figura 57: Gráficas comparativas articulación 1

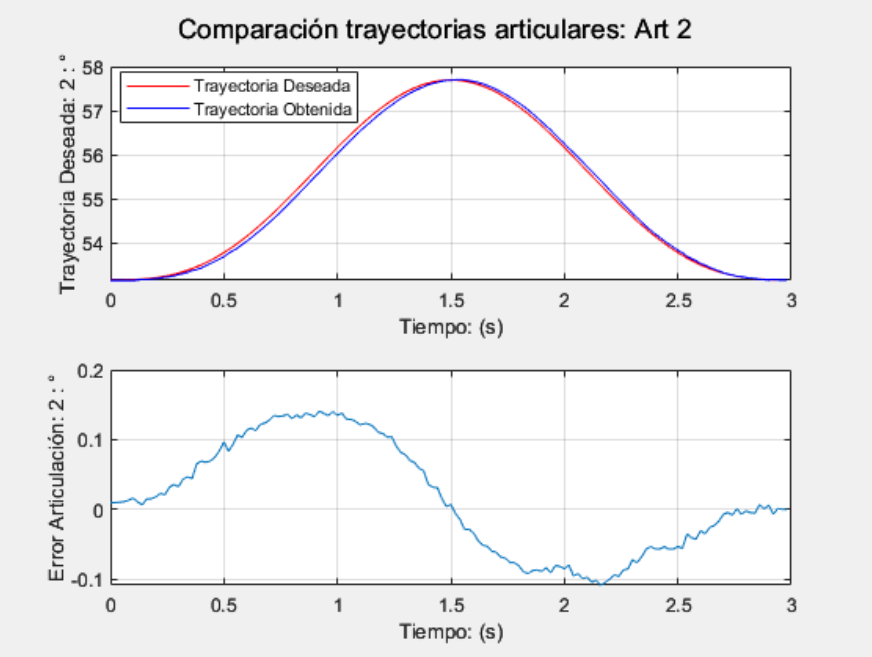


Figura 58: Gráficas comparativas articulación 2

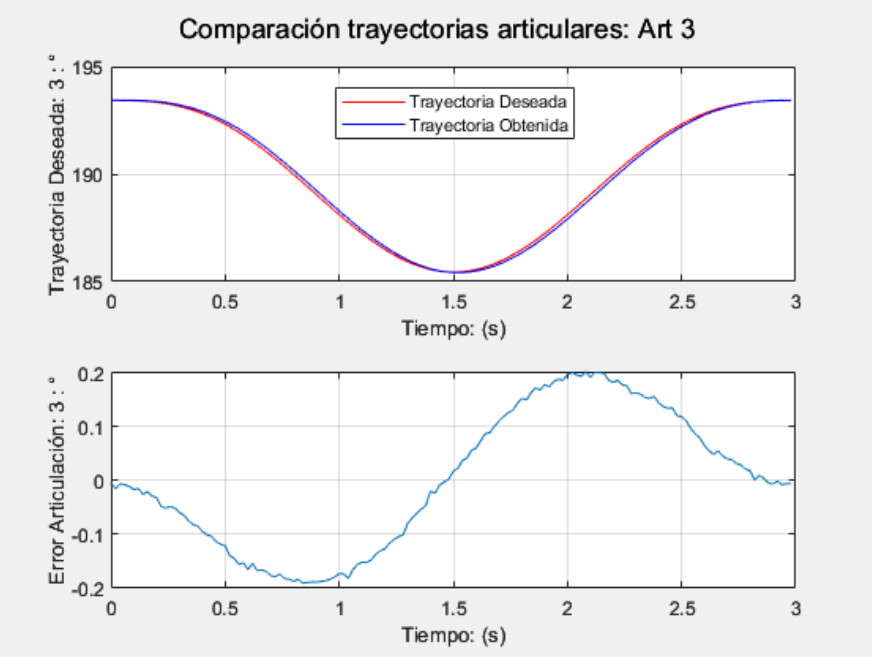


Figura 59: Gráficas comparativas articulación 3

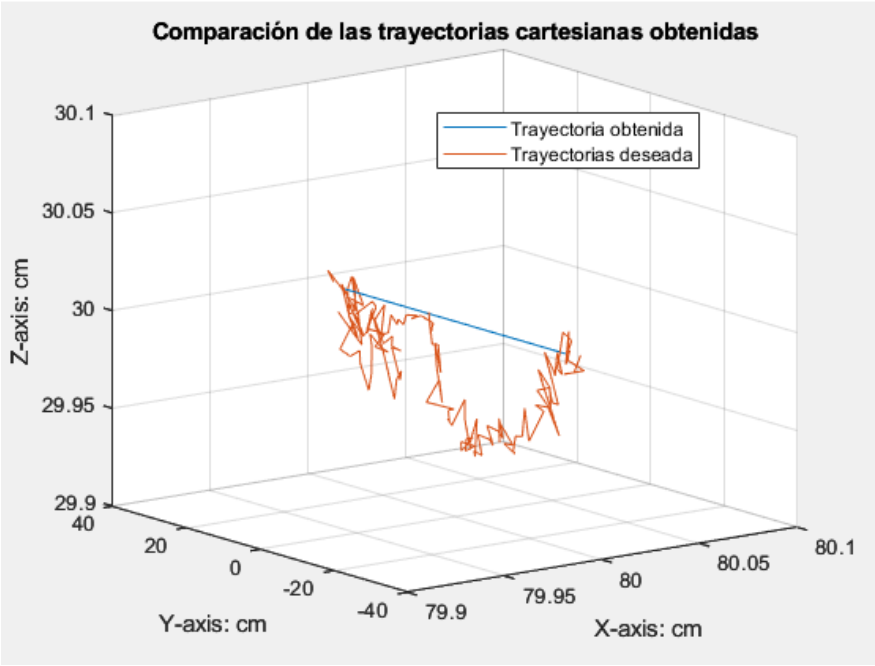


Figura 60: Comparativa entre la trayectoria cartesiana obtenida y la deseada

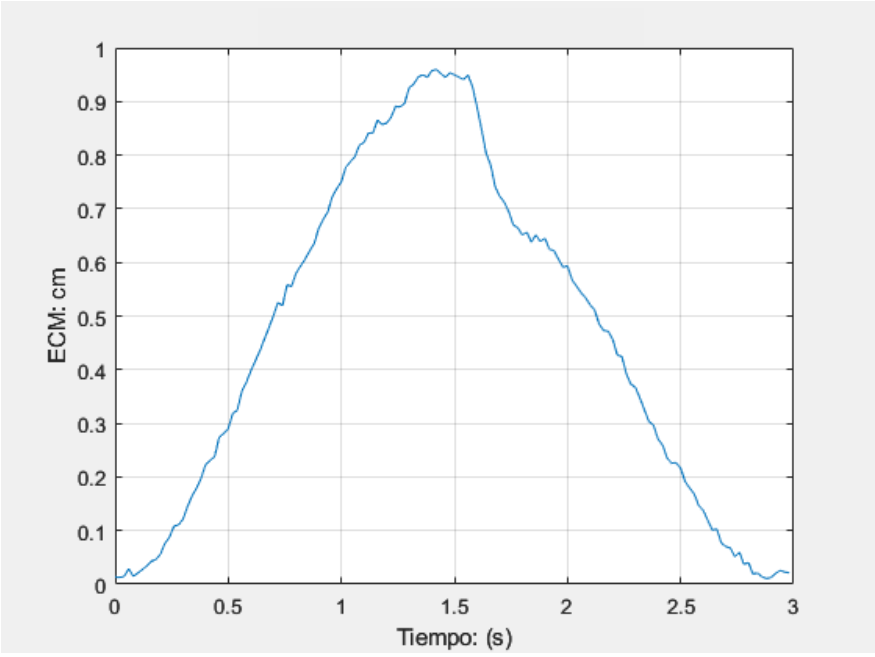


Figura 61: Error euclidiano obtenido

5.2.3. Trayectorias predefinidas

En esta última prueba se ha dispuesto de una trayectoria compleja, la cual forma un círculo entre los ejes x y y, y reposa sobre el eje z. Tiene un radio de 0.1m y las coordenadas del centro son:

P1 = $x1 : 0{,}6m$, $y1 : 0m$, $z1 : 0{,}4m$

Las siguientes gráficas (62, 63, 64, 66, 65) muestran las comparativas entre la trayectoria deseada y la trayectoria obtenida, incluida la gráfica del error euclidiano.

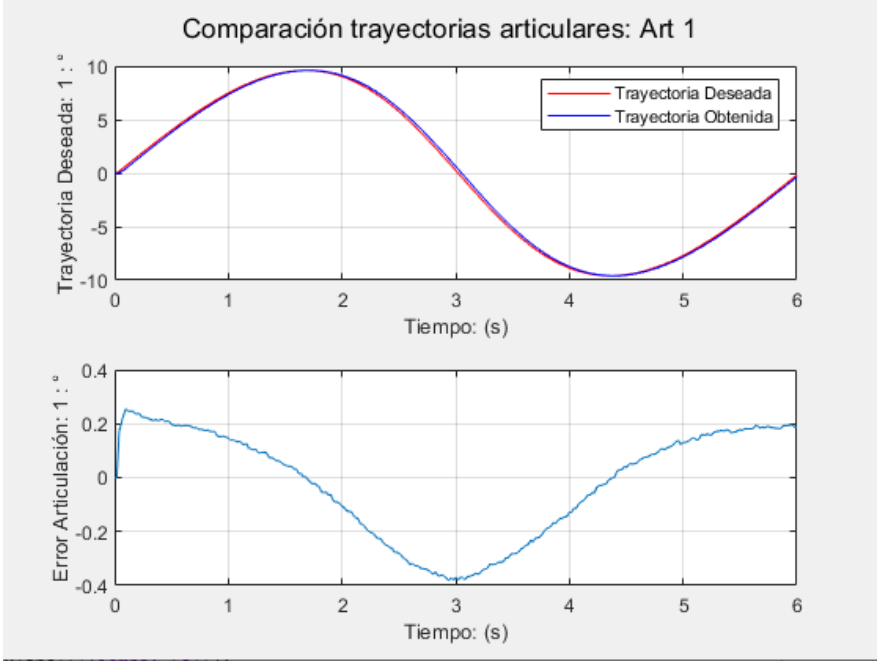


Figura 62: Gráficas comparativas articulación 1

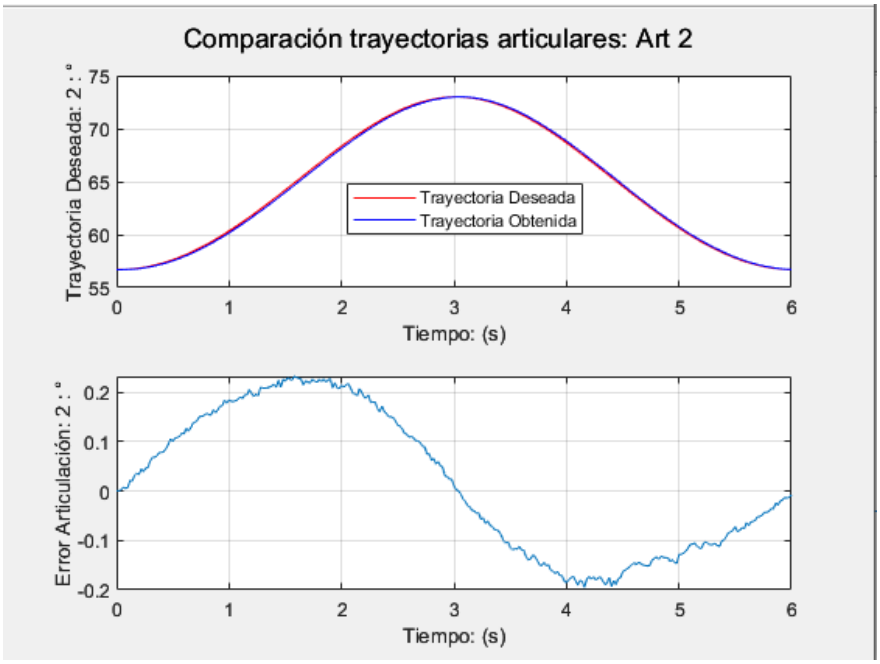


Figura 63: Gráficas comparativas articulación 2

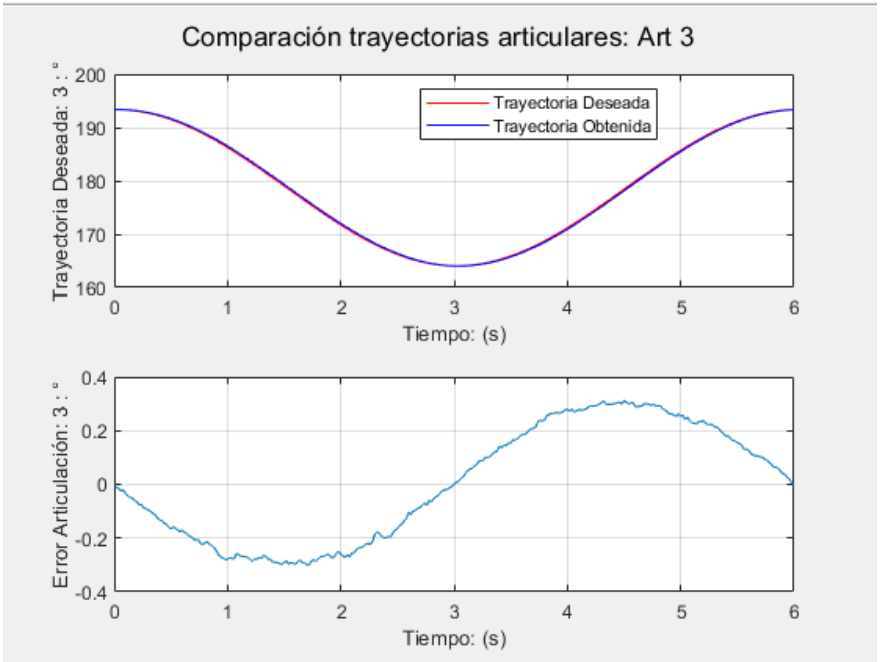


Figura 64: Gráficas comparativas articulación 3

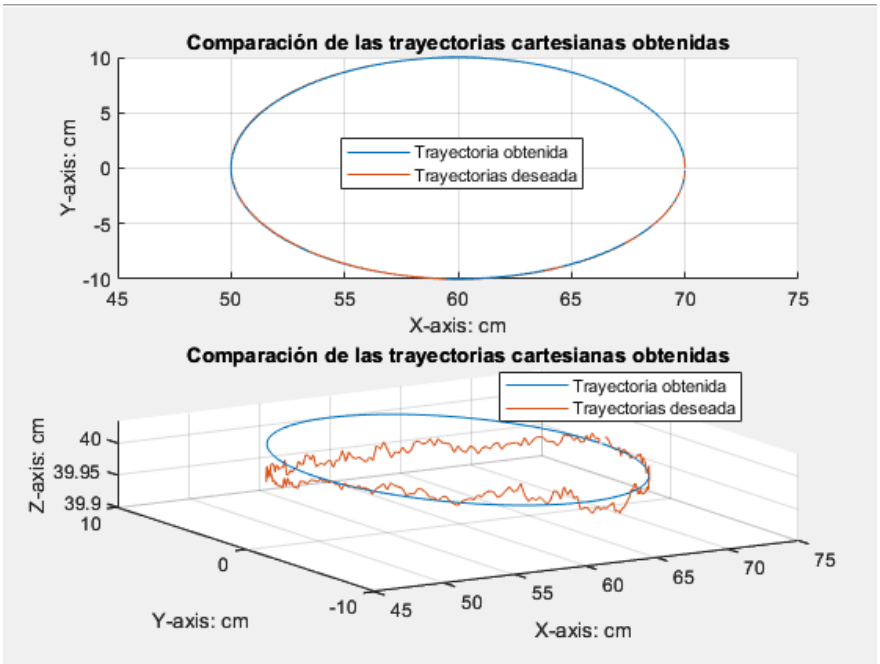


Figura 65: Comparativa entre la trayectoria cartesiana obtenida y la deseada

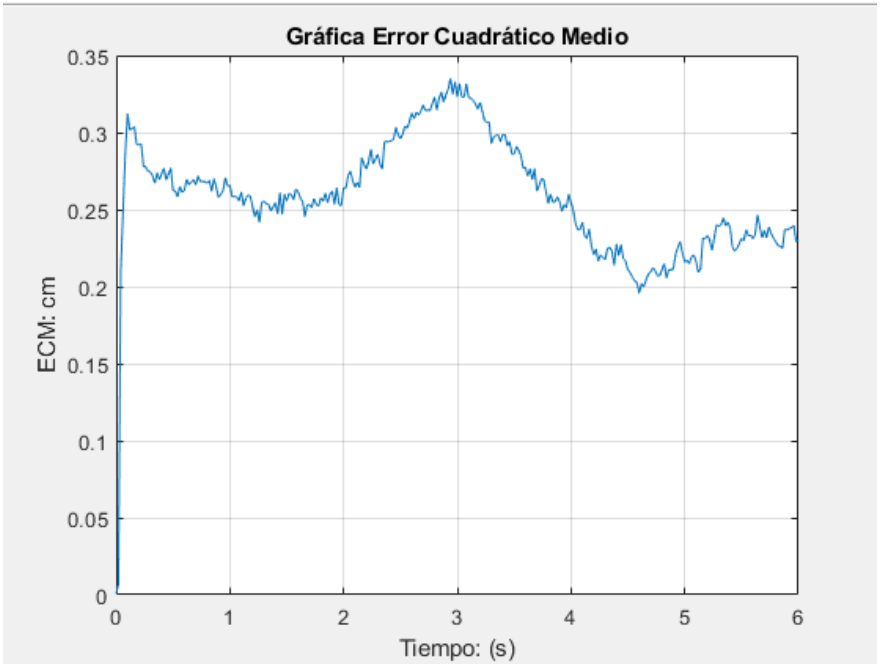


Figura 66: Error euclidiano obtenido

6. Conclusiones

Este proyecto ha logrado el diseño y desarrollo exitoso de una herramienta de software destinada a la manipulación y control de un robot PUMA, proporcionado por la empresa Ready Packers S.A.S en la plataforma Unity 3D. El robot utilizado consta de 6 articulaciones, las cuales están accionadas por motores paso a paso y sensores de posición conocidos como rotary encoder. El robot tiene una estructura de 100 kg de peso, con una base fija situada a 1.20 m del suelo y con medida de brazo y antebrazo de 0.63 m y 0.51 m, respectivamente.

El diseño del software se ha abordado con atención a los aspectos visuales y la organización de la información, lo que sienta las bases para la programación efectiva de la herramienta. El desarrollo de algoritmos específicos ha permitido cumplir con los requisitos de interacción humano-computadora, generación de trayectorias, comunicación y gestión de la base de datos. Esta aplicación se desarrolló en el entorno '.NET' y el lenguaje de programación C#, los cuales son implementados en la plataforma de Unity 3D. Los modelos matemáticos que se han obtenido: MGI, MGD, MDI, y polinomio de grado 5to; fueron codificados en el lenguaje de C#. La aplicación tiene la capacidad de calcular trayectorias cartesianas en un espacio tridimensional, manteniendo una rotación específica del efector final, y también puede determinar el esfuerzo de control requerido para que cada actuador alcance la consigna deseada. Adicionalmente, para la comunicación entre la herramienta software y la Raspberry PI (*tarjeta controladora*) se utilizó el protocolo UDP, lo cual garantiza un traspaso de información bidireccional de baja latencia, aunque se pierdan algunos paquetes en el proceso.

Las pruebas de simulación en el espacio articular, cartesiano y con trayectorias predefinidas arrojaron errores de seguimiento articular de $\pm 0{,}5^{\circ}$ y error euclidiano medio de $\pm 0{,}4cm$. Las

pruebas con el robot PUMA real tuvieron una serie de inconvenientes, los cuales están relacionados con la lectura de los sensores '*explicados en detalle en la sección 3.1.4*'.

Finalmente, se realizaron pruebas articulares, cartesianas y predefinidas del controlador CTC en el robot PUMA real con las articulaciones 1, 2 y 3 , manteniendo la muñeca fija, las cuales arrojaron errores articulares de seguimiento de hasta $\pm 0{,}35^{\circ}$ en las pruebas articulares. Para las pruebas en el espacio cartesiano se obtuvieron errores de seguimiento de $\pm 0{,}1cm$ en trayectorias lineales y de $\pm 0{,}35cm$ para trayectorias complejas como las circulares.

Se pudo comprobar la funcionalidad de la aplicación mediante las pruebas articulares, cartesianas y trayectorias predefinidas, con errores pequeños en el espacio articular y cartesiano, pese a las dificultades que implica el trabajar con un robot tan grande y que ya viene diseñado, sin la posibilidad de cambiar aspectos del hardware. La plataforma desarrollada es un punto de partida para continuar con otros robots, incluyendo sus respectivos modelos matemáticos en el código, y así de esta forma poderlos manipular de una forma cómoda y precisa. Futuros trabajos podrían recomendar al fabricante un diseño más óptimo del hardware con el fin de mejorar los resultados obtenidos.

7. Bibliografía

Referencias

[1] J. Cesar, C. Muñoz, H. Acosta, R. B. Álvarez, and P. Mendoza, "Design of control interface for a scara manipulator with subactuated final effector," vol. 25, 2021.

[2] U. Claudio, C. Juan, and P. José, "Design, construction and control of a scara manipulator with 6 degrees of freedom," *Journal of Applied Research and Technology*, vol. 14, no. 6, pp. 396–404, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1665642316300931

[3] K. Borowski and A. Wojtulewicz, "Implementation of robotic kinematics algorithm for industrial robot model using microcontrollers," vol. 55, no. 4, 2022, Conference paper, p. 248 – 253, cited by: 0; All Open Access, Bronze Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85137170013&doi=10.1016/2fj.ifacol.2022.06.041&partnerID=40&md5=5a6573a46fea7066d0ddb21da7792406

[4] P. Wang, L. Yu, L. Miao, and L. Zhang, "Research on motion control of the 2-dof packing robot," in *Journal of Physics: Conference Series*, vol. 1550, no. 6. IOP Publishing, 2020, p. 062011.

[5] "Conectar un codificador rotativo de cuadratura con arduino." [Online]. Available: https://www.makerguides.com/es/quadrature-rotary-encoder-with-arduino/

[6] S. Ricci and V. Meacci, "Simple torque control method for hybrid stepper motors implemented in fpga," *Electronics*, vol. 7, no. 10, p. 242, 2018.

[7] A. Vivas, *Diseño y control de robots industriales: teoría y práctica*, 1st ed. Elaleph, Argentina, 2010, vol. 1.

[8] W. Ziling, Z. Lai, S. Xiaojie, L. Guoyue, L. Rui, and H. Yun, "Hybrid force/position control in workspace of robotic manipulator in uncertain environments based on adaptive fuzzy control," *Robotics and Autonomous Systems*, vol. 145, p. 103870, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092188902100155X

[9] W. Guanglei, Z. Xuping, Z. Lina, L. Zirong, and L. Jinguo, "Fuzzy sliding mode variable structure control of a high-speed parallel pnp robot," *Mechanism and Machine Theory*, vol. 162, p. 104349, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X21001075

[10] J. M. Vivas, Andrés y Sabater, "Manipulación de robots ur5 usando matlab/simulink y ros," in *2021 Conferencia Internacional IEEE sobre Mecatrónica y Automatización (ICMA)*, 2021.

[11] A. Olaru, T. Dobrescu, S. Olaru, and I. Mihai, "Proper labview platform for the assisted research of the kinematics and dynamics in robotics," 2021.

[12] A. Angerer, A. Hoffmann, A. Schierl, M. Vistein, and W. Reif, "Robotics api: Object-oriented software development for industrial robots," *Journal of Software Engineering for Robotics*, vol. 4, no. 1, pp. 1–22, 2013.

[13] A. Angerer, A. Bareth, A. Hoffmann, A. Schierl, M. Vistein, and W. Reif, "Two-arm robot teleoperation using a multi-touch tangible user interface," 2012.

[14] D. Muñoz, E. Núñez, and O. Vivas, "Pa10 robot's movement through natural interface," *Journal of Autonomous Intelligence*, vol. 5, no. 1, p. 53 – 61,

2022, cited by: 0; All Open Access, Gold Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85139012966&doi=10.326292/fjai.v5i1.506&partnerID=40&md5=0f533e4dd4d572b9b02b38d1e7c2c563

[15] R. Aroca, D. Tavares, and G. Caurin, "Controlador de robot scara usando linux en tiempo real," in *2007 IEEE/ASME conferencia internacional sobre mecatrónica inteligente avanzada*, 2007.

[16] "Shenzhen guanhong automatización co.,ltd." [Online]. Available: https://www.szghauto.com/

[17] O. I. Hernández, "Lector de señales en cuadratura aquadb con interfaz serial utilizando fpga," 2012. [Online]. Available: https://www.cenam.mx/Dimensional/pdfs/Art_04_Lect_Cuadratura_con_FPGA.pdf

[18] S. M. Orozco-Soto and J. C. R. Fernández, "Control par calculado difuso basado en pasividad para seguimiento de trayectorias de robots manipuladores." *Res. Comput. Sci.*, vol. 91, pp. 131–141, 2015.

[19] E. Yime Rodríguez, J. Roldán Mckinley, and J. Villa Ramírez, "Control por par calculado de un robot paralelo planar 2-rr," *Prospectiva*, vol. 15, no. 2, pp. 85–95, 2017.

[20] W. Lin and Z. Zheng, "Simulation and experiment of sensorless direct torque control of hybrid stepping motor based on dsp," in *2006 International Conference on Mechatronics and Automation*. IEEE, 2006, pp. 2133–2138.

8. Anexos

COM_upd.cs M | Adaptador_art_puma.cs X | DriverRobotInterfaz.cs | bd_trayectorias.cs M

Assets > Scripts > Interface > Articular > Adaptador_art_puma.cs > Adaptador_art_puma > Start()

```
9
10  public class Adaptador_art_puma : MonoBehaviour
11  {
12      public RectTransform scroll_view;
13      public RectTransform content_arts;       // El ambiente content del scrol_arts view
14      public GameObject val_arts;               // El prebaf de las trayectorias articulares
15
16
17      public Button btn_agregar;          // Boton agregar
18      public Button btn_eliminar;         // Boton para eliminar una trayectoria
19      public Button btn_probar;           // Boton para probar una taryectoria en los sliders
20      public Button btn_Q0;               // Boton para llevar el robot a la posicion inicial
21      public Button btn_cargar;           // Btoton para cargar trayectorias del scroll view
22
23
24      // Cuadros de dialogo
25      public RectTransform cuadro_dialogo_subir;
26      public RectTransform cuadro_dialogo_bajar;
27
28      // Base de datos
29      public RectTransform BD_panel;
30
31      public GameObject gemelo_digital;
32      // driver _robot_ graficas y demas (clase estatica)
33      public GameObject obj_driver_rob_inter;
34
35      // Script que maneja el scroll view de las trayectorias
36      private Scroll_view_tray scrol_view_tray;
37
38      // Numero de articulaciones
39      private int NUMERO_ARTICULACIONES = 6;
```

Figura 67: Algoritmo ventana articular, cartesiana y predefinidas

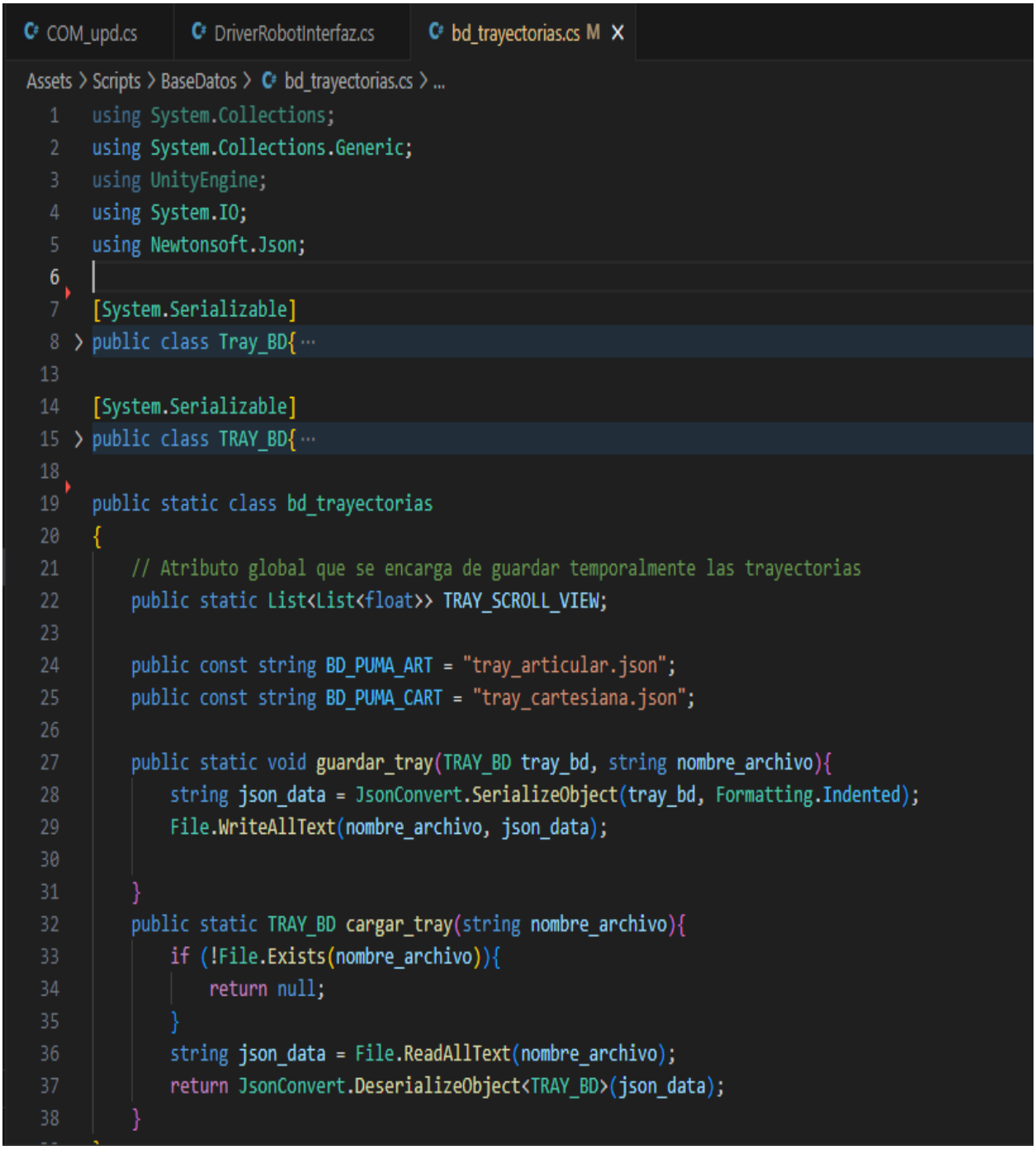


Figura 68: Algoritmo base de datos

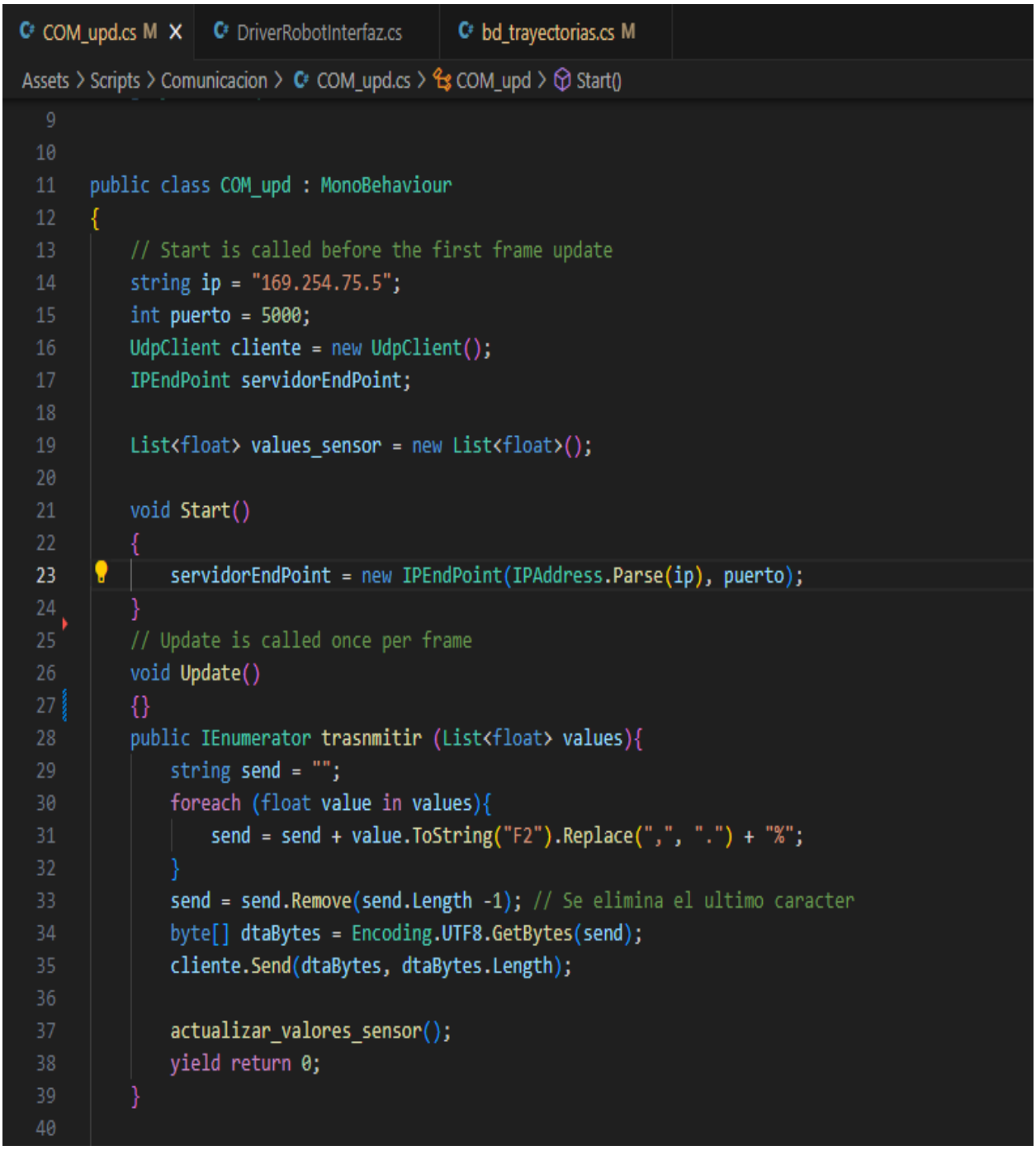


Figura 69: Algoritmo de comunicación

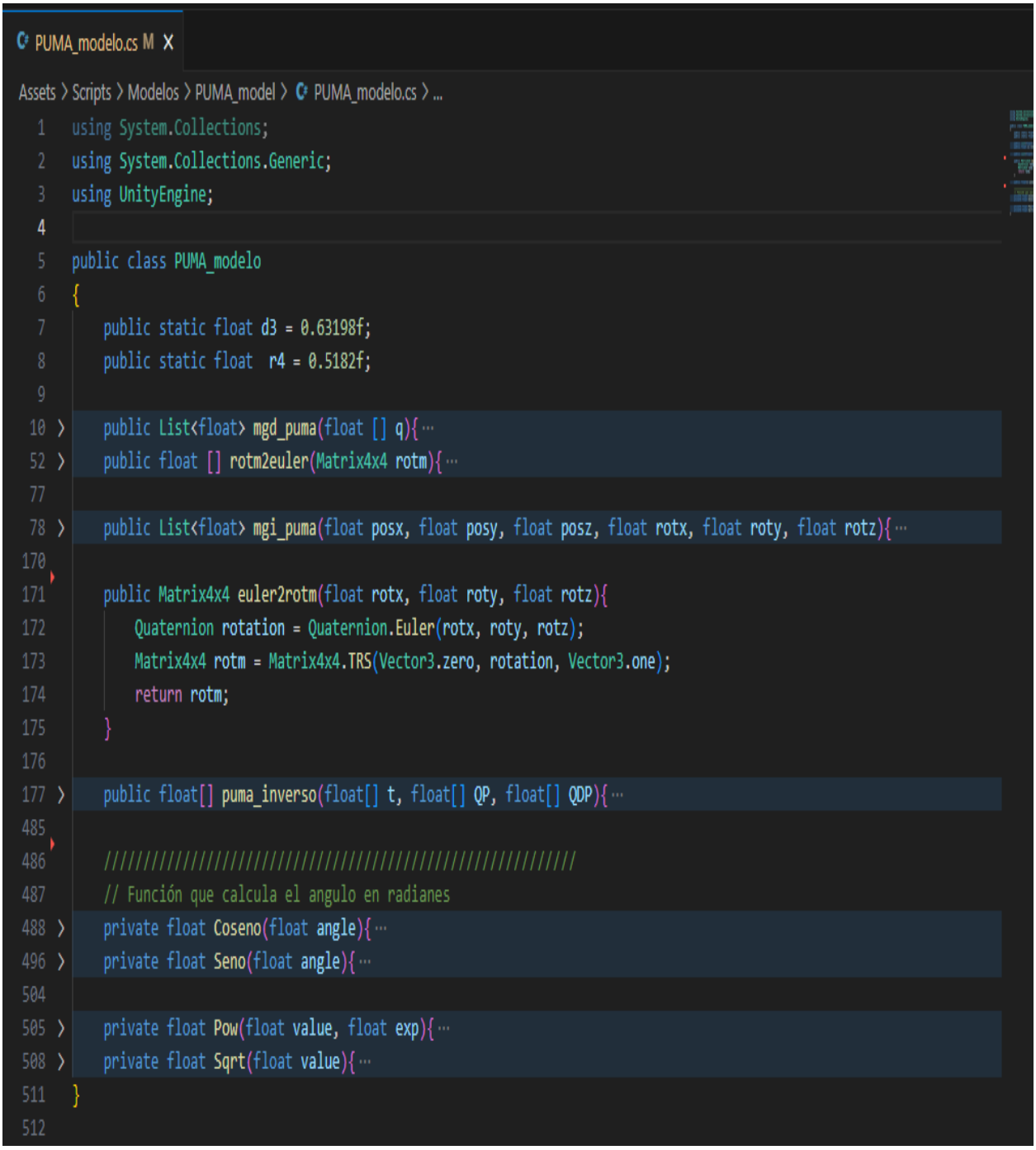


Figura 70: Algoritmo de los modelos del robot

COM_upd.cs M | Trayectoria.cs X | DriverRobotInterfaz.cs | bd_trayectorias.cs M

Assets > Scripts > Trayectorias > Trayectoria.cs > Trayectoria > Transpuesta(List<List<float>> tt, bool bol)

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using RobSof.Assets.Scripts.Interface.Articular;
5
6   public class Trayectoria
7   {
8       public static float TIEMPO_MUESTREO = 0.02f;
9       private PUMA_modelo puma_modelo = new PUMA_modelo();
10      Rangos_arts rangos_arts = new Rangos_arts();
11
12      public List<float> grado_5(float pos_inicial, float pos_final, int T_FINAL){
13          List<float> tray = new List<float>();
14          int num_steps = Mathf.RoundToInt(T_FINAL/TIEMPO_MUESTREO);
15          float delta_pos = pos_final-pos_inicial;
16          float temps = 0;
17          for (int j= 0; j< num_steps; j++ ){
18              tray.Add(pos_inicial + delta_pos*(10*Mathf.Pow(temps/T_FINAL, 3) - 15*Mathf.Pow(temps/T_FINAL,4)
19              + 6*Mathf.Pow(temps/T_FINAL,5)));
20              temps += TIEMPO_MUESTREO;
21          }
22          return tray;
23      }
24
```

Figura 71: Algoritmo de trayectorias

Figura 72: Algoritmo de cuadratura

PUMA-SOFT – Guía de Usuario (App)

v. 1.1

Hernán Dario Trullo Muñoz

January 16, 2024

Introduction

PUMA-SOFT es una plataforma software que permite la programación de trayectorias en el espacio cartesiano y el espacio articular al robot Puma de 6 grados de libertad. Esta plataforma se comunica a través de UDP con una placa raspberry PI3 + y realiza el control CTC de robot en tiempo real. Adicionalmente, brinda y provee las herramientas software que permiten el buen manejo del robot Puma, tale como: programación de trayectorias articulares y cartesianas, validación de trayectorias predefinidas, control CTC en tiempo real y la gestión de una base de datos para guardar las trayectorias definidas por el usuario.

Contents

1 Inicio

Esta guía te proporcionará los conocimientos necesarios para comenzar a utilizar la aplicación PUMA-SOFT, que te permitirá controlar y manipular en tiempo real un robot PUMA 560. A lo largo de esta guía, encontrarás descripciones detalladas y pasos prácticos que te permitirán utilizar el robot de manera intuitiva, incluso si tienes un nivel básico de experiencia en el campo de la robótica. Estamos aquí para ayudarte a aprovechar al máximo esta emocionante herramienta, ¡así que comencemos!

2 Instalación

2.1 Software

El software viene en una carpeta comprimida en zip. El usuario debe descomprimir dicha carpeta. Dentro de este hay un archivo con extensión .exe llamado *PumaSoft.exe*. Finalmente tiene que ejecutar o abrir este archivo, el cual abre la interfaz principal, una vez ahí el usuario puede navegar por todas las secciones que tiene la interfaz para el manejo y control del robot Puma real.

2.2 Hardware

Debido a la arquitectura del controlador que utiliza la Raspberry Pi 4 (consultar anexos), es necesario que esta esté conectada a un enrutador que se encargue de gestionar el envío de paquetes en ambas direcciones, tanto desde la Raspberry como desde el computador que compila la aplicación.

En consecuencia, es necesario conocer la dirección IP asignada por el enrutador a la Raspberry Pi 4. Con esta dirección IP, el computador en el cual se ejecuta el software puede establecer una comunicación con el robot de forma efectiva.

3 Comunicación

El robot debe estar conectado a la misma red que el computador que compila la aplicación. Además se debe conocer la dirección ip que el router asigna a la raspberry. Esa dirección ip es la que el usuario debe colocar en la aplicación para que pueda existir una comunicación segura y estable entre el computador y la raspberry. Al omitir alguno de estos detalles, no se podrá establecer comunicación con la raspberry.

4 Creación de trayectorias

En esta sección muestran el paso a paso con que el usuario usuario final puede utilizar para crear trayectorias articulares y cartesianas.

4.1 Articulares

Para ingresar a esta sección el usuario debe presionar el botón del panel principal con el nombre *'Articular'*, como se muestra en la figura 1.

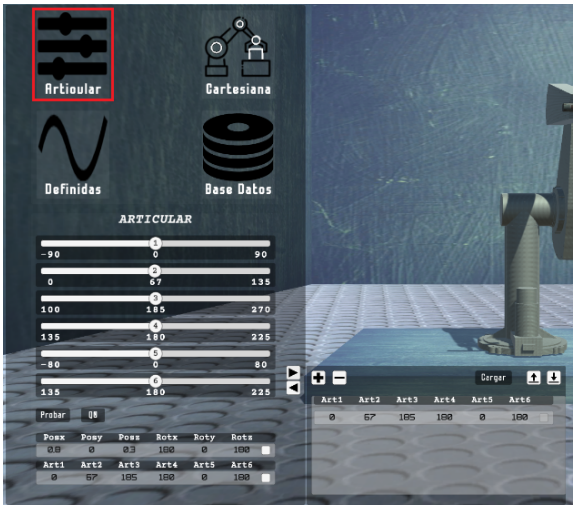


Figure 1: Panel trayectorias articulares.

4.1.1 Trayectorias simples

Una vez en el panel de las 'trayectorias articulares', podrá mover a disposición los 6 sliders que hay disponibles. Estos sliders se asocian a cada articulación física del robot PUMA real.

Para controlar cada articulación del robot, el usuario puede utilizar los sliders asociados a cada una de ellas. Por ejemplo, en la Figura 2 se muestra una configuración particular de los sliders. Luego, al presionar el botón **Probar**, el robot se moverá a la posición indicada por los sliders. Esta posición también se puede observar en el **panel de posiciones**, donde se muestra tanto la posición articular como la posición cartesiana. En este caso, nos interesa la posición articular.

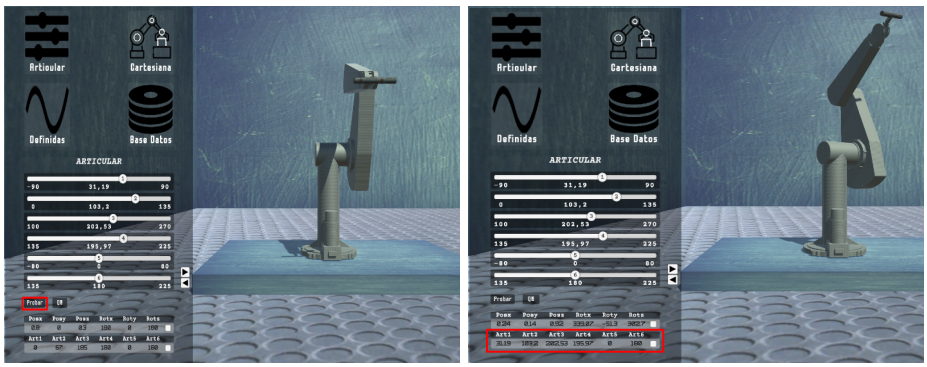


Figure 2: Movimiento del gemelo digital con los sliders.

4.1.2 Trayectorias compuestas

La interfaz cuenta con una herramienta que despliega una subsección, la cual puede ser utilizada para la creación de trayectorias compuestas de dos o mas puntos articulares. La figura 3, muestra los botones utilizados para el despliegue de esta subsección. Adicionalmente, con el botón que tiene el signo **+**, el usuario puede, por ejemplo, agregar el punto articular antes asignado, con lo cual, se van agregando en un listbox ,una debajo de otra.

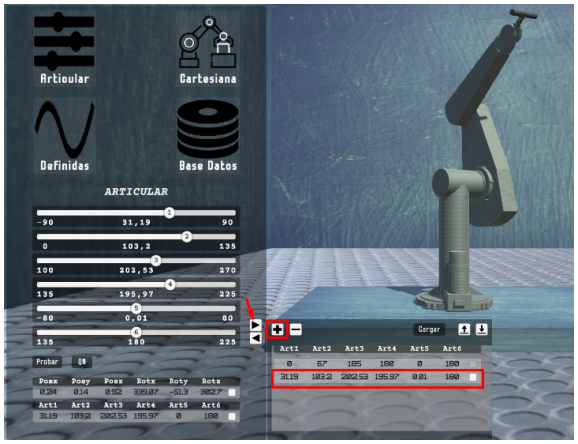


Figure 3: despliegue de la subsección de trayectorias compuestas.

En esta subsección el usuario puede cargar la trayectoria articular que ya ha creado simplemente presionando el botón **Cargar**. En otro caso, puede eliminar algunas de las casillas agregadas al 'marcar' las que desea eliminar, esto al presionar el botón con el signo '**–**'. Ver figura 4.

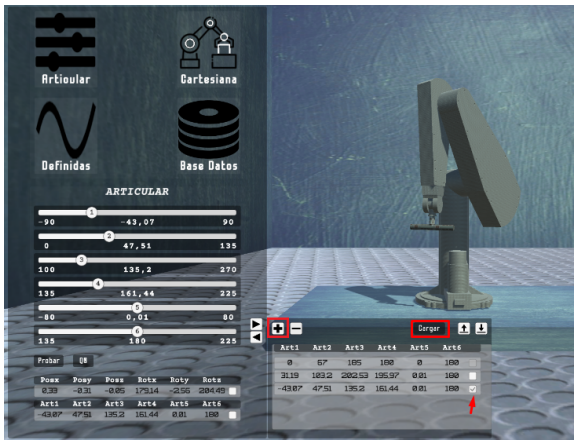


Figure 4: Eliminar y cargar trayectorias.

4.2 Cartesianas

Para ingresar a esta sección el usuario debe presionar el botón del panel principal con el nombre *'Cartesianas'*, como se muestra en la figura 1.

4.2.1 Trayectorias simples

En esta sección el usuario puede programar trayectorias punto a punto *lineales* en el espacio cartesiano; es decir que el usuario puede mover el robot en una posición *x,y,z* con su rotación *rotX, rotY, rotZ*.

Por ejemplo, el robot inicia en una posición cartesiana que ya está predefinida, a continuación el usuario puede escribir la posición *x,y,z* y la rotación *rotX, rotY, rotZ* en las cajas de texto dispuestas para ello, ver figura 5.

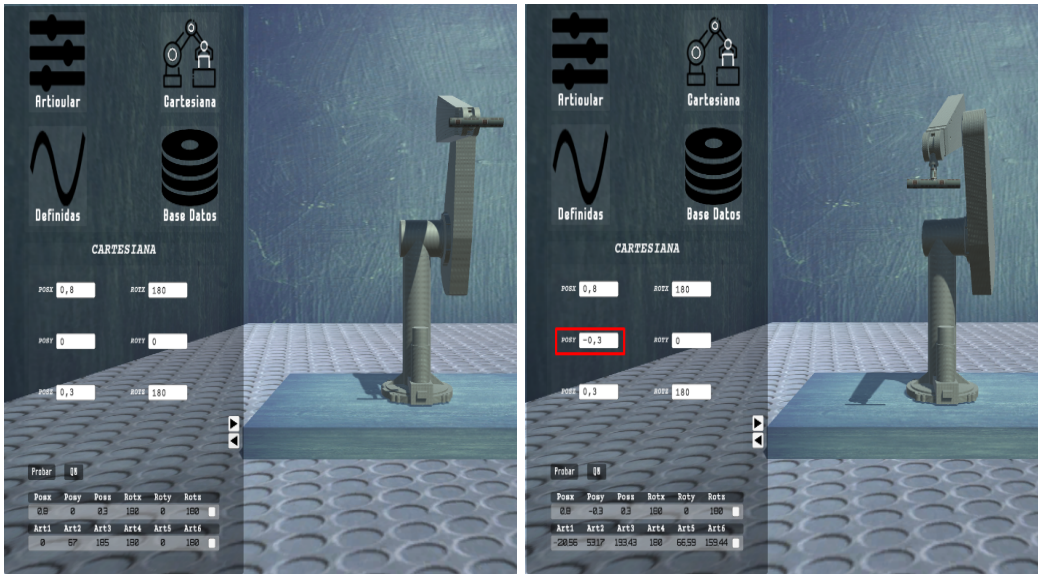


Figure 5: Movimiento del gemelo digital en el espacio cartesiano.

4.2.2 Trayectorias compuestas

En el mismo caso que el anterior para las trayectorias articulares, este panel cuenta con una herramienta para programar trayectorias compuestas formadas por mas de un punto en el espacio cartesiano.

El funcionamiento de la acción **cargar**, consiste en que el robot vuelve a la posición inicial y luego recorre cada uno de los puntos programados en el 'listbox', ver figura 6.

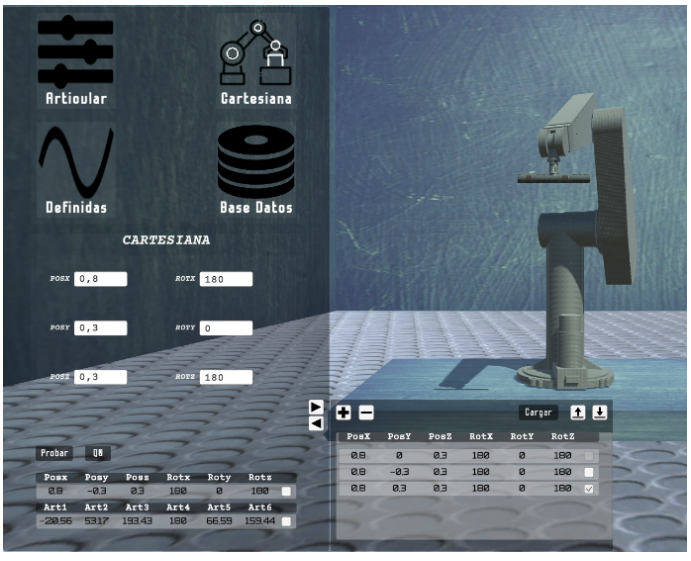


Figure 6: Trayectorias cartesianas compuestas.

4.3 Predefinidas

En este panel se han definido dos tipos de trayectorias predefinidas: circulares y sinusoidales.

En la primera interfaz (como se muestra en la figura), se puede generar un círculo con un centro en el punto $p(x_1, y_1, z_1)$ y un radio r_1. El programa utiliza estos valores para ejecutar un algoritmo que crea una trayectoria alrededor del eje x, es decir, el círculo se forma en un plano definido por los ejes z e y.

En la segunda interfaz, se puede generar una trayectoria sinusoidal. Para ello, el usuario debe proporcionar un punto cartesiano inicial, un punto cartesiano final y una frecuencia que debe estar dentro de ciertos rangos predefinidos. Similar al caso anterior, esta trayectoria se traza a lo largo del eje x, pero varía a lo largo de los ejes x e y.

De la misma manera que el panel predefinidas, este tiene, botones para proba la trayectoria y devolver al robot a la posición inicial preestablecida por la aplicación.

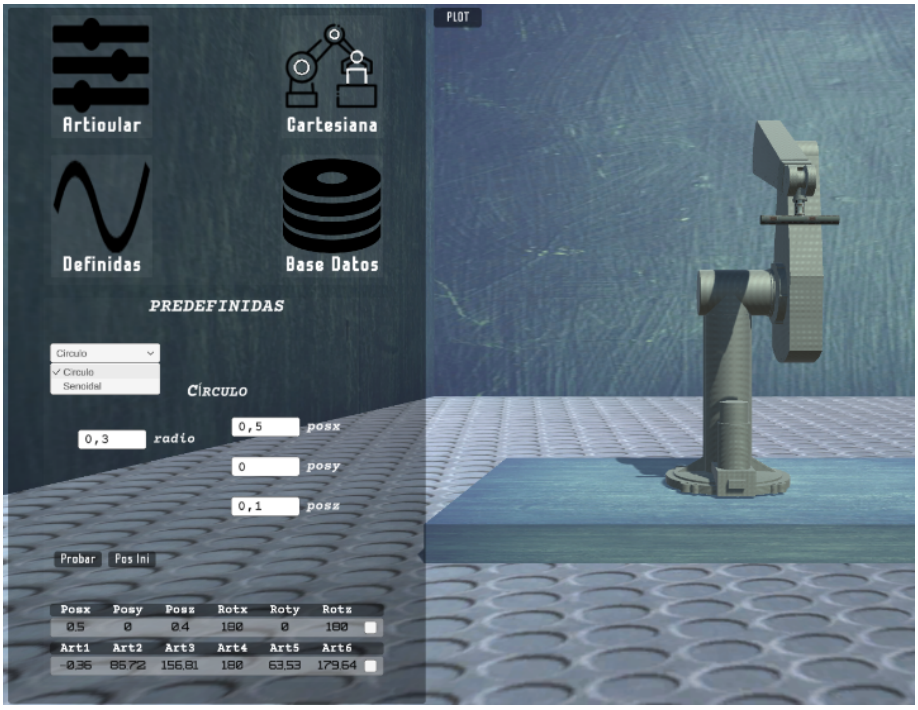


Figure 7: Panel de trayectorias predefinidas.

4.4 Panel de visualización de gráficas

El panel de visualización de errores es una herramienta que permite visualizar la precisión con la que el robot efectúa una trayectoria articular o cartesiana.

El usuario accede a esta disposición mediante el botón 'plot' mostrado en el recuadro rojo de la figura 8.

Una vez ahí, el usuario puede evidenciar cada uno de los errores recopilados durante la trayectoria. Además el usuario también puede evidenciar el error cuadrático medio 'ECM' cuando se efectúe una trayectoria cartesiana ver figura 9.

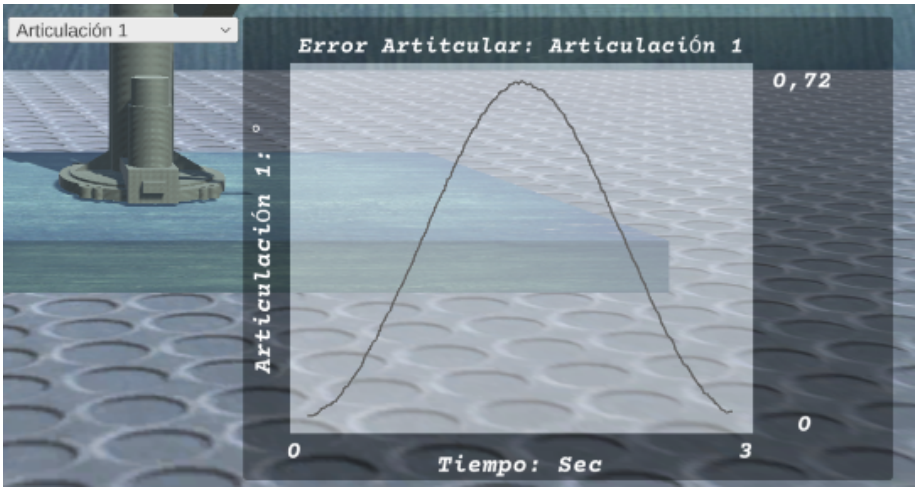


Figure 8: Panel visualización de gráficas: error articular 1

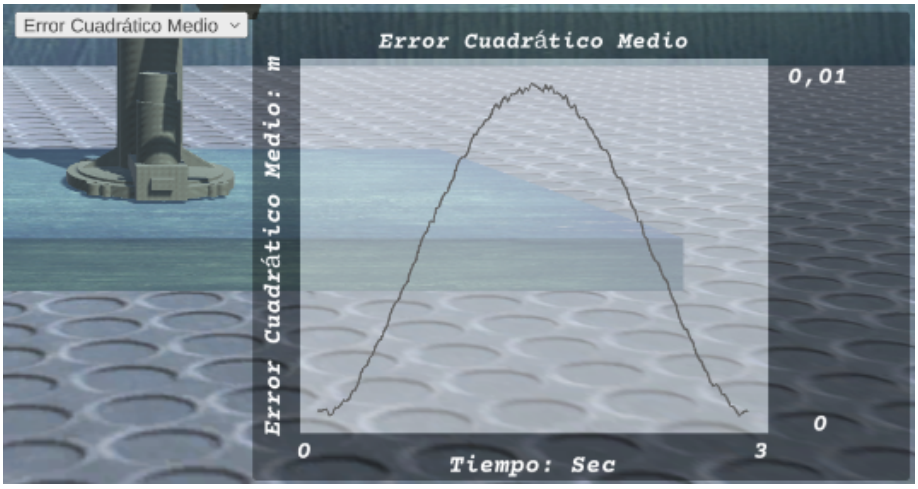


Figure 9: Panel visualización de gráficas: error cuadrático medio

5 Acceso a la base de datos

Para el acceso a la base de datos, se ha dispuesto de un panel con el mismo nombre. Para acceder a este panel el usuario debe dar 'click' en el botón con este nombre.

El usuario tiene herramientas que le permiten modificar o actualizar, eliminar y guardar las trayectorias previamente cargadas. Para hacer el cargue de las trayectorias al panel de la base de datos, éste debe previamente haberlas cargado desde alguno de los paneles 'articular', 'cartesiano', 'predefinidas'; como se muestra en la figura 10; con los botones resaltados: - ↑ - para enviar la trayectoria hacia la base de datos y - ↓ - recibir la trayectoria al 'listbox' local.

Con la trayectoria cargadas hacia el panel 'base de datos', el usuario puede bajarlas, adicionalmente, debe asignar un nombre y una descripción para guardarlas por primera vez (ver figura 11).

Una vez el usuario tenga listos: el nombre y la descripción, puede actualizar o modificar y guardar los cambios en la base de datos (ver figura 12).

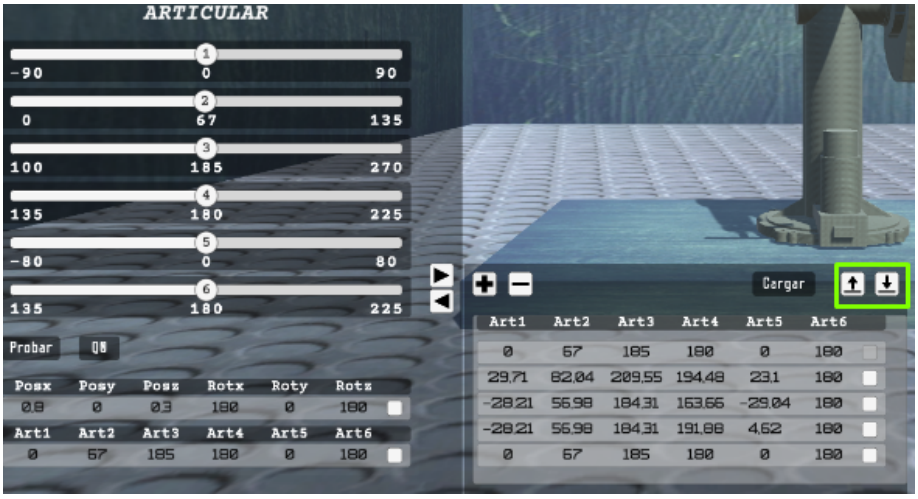


Figure 10: Trayectorias cargadas desde el panel 'articular'.

Figure 11: Panel base de datos: modificación de la trayectoria.

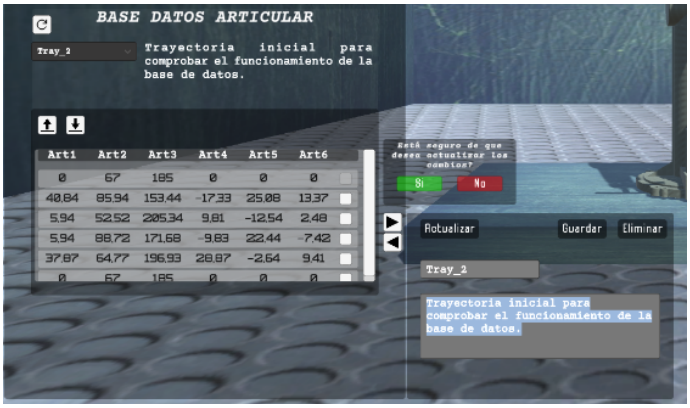


Figure 12: Panel base de datos: actualización de la trayectoria.

Finalmente el usuario puede cargar las trayectorias desde la base de datos, al seleccionar una de los 'items' en la lista desplegable que se actualiza cada que se presiona el botón resaltado en el recuadro verde de la figura 11.