CENTRO DE INVESTIGACIONES EN OPTICA, A.C.

POSGRADO INTERINSTITUCIONAL DE CIENCIA Y TECNOLOGÍA

OPTIMIZACIÓN DE TÉCNICAS DE MODELACIÓN Y CONTROL PARA MANIPULADORES ROBÓTICOS Y SU APLICACIÓN EN ROBOTS VIRTUALES

TESIS

QUE PARA OBTENER EL GRADO

ACADÉMICO DE

DOCTOR EN CIENCIA Y TECNOLOGÍA EN LA ESPECIALIDAD DE MECATRÓNICA Y DISEÑO MECÁNICO

PRESENTA

M.M.A. FILEMÓN ARENAS ROSALES

DIRECTOR

DR. FERNANDO MARTELL CHÁVEZ

POSGRADO INTERINSTITUCIONAL CIENCIA Y TECNOLOGIA

AGUASCALIENTES, AGS., NOVIEMBRE, 2023

AGRADECIMIENTOS

Agradezco a Dios y a la vida por darme la oportunidad de terminar una meta más de las que tengo por cumplir.

A mi esposa Josefina por su amor y su apoyo incondicional en todo momento, a mis hijos Lucas y Marijose quienes son parte de mi motivación e inspiración para llegar hacer un ejemplo para ellos.

Agradezco plenamente a mi asesor el Dr. Fernando Martell Chávez por confiar en mí y darme la oportunidad de seguir trabajando con él, gracias por su gran apoyo, motivación y paciencia, ya que sin su ayuda y conocimiento no hubiese sido posible en realizar este proyecto.

A mis padres Filemón Arenas Silva y Evangelina Rosales Esquivel por darme la vida, por su amor, apoyo y sus enseñanzas en todo momento.

A mis asesores de comité tutorial el Dr. Rigoberto López Padilla y el Dr. Luis Manuel Valentín Coronado por su invaluable tiempo y su acompañamiento en las asesorías de la revisión de mi trabajo.

A los miembros del jurado la Dra. Irma Yolanda Sánchez Chávez, el Dr. Carlos Alberto Paredes Orta, el Dr. Juan Manuel López Tello y el Dr. Raúl Arturo Ortiz Medina por su gran apoyo en la revisión de este trabajo.

A mis hermanos y hermanas por su apoyo incondicional en todo momento.

Al Centro de Investigaciones en Óptica A.C (CIO) Unidad Aguascalientes y al Posgrado Interinstitucional en Ciencia y Tecnología (PICYT) por darme la oportunidad de estudiar un doctorado y brindarme todo el apoyo para mi formación profesional, agradezco las enseñanzas y consejos brindados por mis maestros en este trayecto de 4 años en mi estancia en el doctorado.

Al Consejo Nacional de Humanidades Ciencia y Tecnología (CONAHCYT) por apoyar económicamente mis estudios en mi estadía dentro del doctorado.

A mis compañeros por ser parte de este logro, gracias por compartir sus conocimientos y experiencias.

A todas las personas que colaboraron en mi formación y en este proyecto, gracias.

RESUMEN

El trabajo a distancia se está volviendo común en escuelas y universidades, la formación a distancia implica el uso de tecnologías como las máquinas de realidad virtual que facilitan la enseñanza y el aprendizaje de temas de ingeniería como robótica y mecatrónica. Los robots virtuales y sus sistemas de control se pueden implementar usando software de código abierto y plataformas de hardware de bajo costo. Este documento presenta la metodología para el desarrollo de un robot virtual articulado UR5 diseñado para su uso en un laboratorio virtual para la realización de prácticas de robótica y control. El modelado y la animación 3D del robot virtual se realizan en Solidworks y Blender respectivamente. El robot virtual requiere interactuar con un controlador externo utilizando un esquema “hardware in the loop”. El controlador externo se programa en CODESYS y utiliza Modbus TCP como protocolo de comunicación. Como técnicas de modelación y control del robot se propone la estimación del par de carga a partir de un modelo de motor de corriente directa de orden reducido y convertido a tiempo discreto para simular los actuadores. Se utilizan algoritmos de control que combinan una máquina de estados finitos para el control de posición de las articulaciones con un control de velocidad que utiliza un controlador proporcional integral derivativo basado en el principio de modelación interna. El esquema de control propuesto permite la operación manual y de lazo cerrado de los ejes de movimiento. Las ecuaciones cinemáticas se implementaron también en el controlador externo para lograr el control cinemático inverso del robot UR virtual. El sistema desarrollado es útil para el aprendizaje práctico de la robótica, tanto del control de los ejes de movimiento como del control cinemático.

Palabras clave: Blender, CODESYS, Control Cinemático, Estimación de Par de Carga, Laboratorio Virtual, Maquina de Estados Finitos, Modbus TCP, Robot virtual, UR5.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

CAPÍTULO 1 INTRODUCCIÓN

La robótica es una tecnología mecatrónica que permite automatizar procesos de manufactura de una forma eficiente, por lo cual se tiene que estar continuamente mejorando a la par del avance tecnológico para alcanzar funciones más avanzadas de automatización. El empleo de sistemas robóticos es muy versátil e idóneo para muchas aplicaciones de la industria manufacturera, como es el caso de robots con forma de brazo articulado que realiza funciones como; ensamble de componentes, manipulación de máquinas y herramientas, tomar y colocar piezas, entre otras funciones. Las compañías que disponen de sistemas robóticos con configuración de brazo articulado suelen manejar solo una marca de equipo debido a los costos o al trabajo que este realice, limitándose a programar solo el equipo con el que cuentan, además de que en los sectores educativos no todos tienen un sistema de brazo robótico que les permita entrenar en un ambiente real o solo disponen de versiones de simuladores que carecen de contenido técnico para su aprendizaje.

Este trabajo presenta el diseño de un laboratorio virtual de bajo costo de un robot articulado UR5 para su uso como plataforma de capacitación y enseñanza en conceptos de robótica y su control cinemático. Mientras que otros trabajos y sistemas relacionados están más enfocados al aprendizaje de la programación de robots, el sistema desarrollado está orientado a la enseñanza y aprendizaje del movimiento del robot y el control cinemático. El software Blender V2.79 se utiliza como una herramienta de software para simular y animar el modelo 3D del robot UR5; por otra parte, el software CODESYS V3.5 se utiliza para desarrollar e implementar el controlador del sistema tipo hardware in the loop. La comunicación del simulador de robot y el controlador del robot se realiza a través del protocolo de red industrial Modbus TCP. Con el sistema desarrollado, los usuarios necesitan programar el control de movimiento y generar algoritmos de control cinemático inverso en el controlador externo, desarrollando así habilidades para el diseño y control de robots.

1.1 Antecedentes

Hoy en día, con los desarrollos de tecnologías de la información y la comunicación, implementan nuevas estrategias de enseñanza y aprendizaje en beneficio de la sociedad, aportando diferentes herramientas que permiten la interacción de las personas a través de videoconferencias, seguimiento de procesos por medio de accesos remotos o desarrollar pruebas mediante simulaciones y laboratorios virtuales (LV). Los LV se utilizan cada vez más en los campos de la educación, la investigación y el desarrollo, donde universidades y centros de investigación han estado trabajando muy activamente en el desarrollo de LV, desarrollando plataformas de formación equipadas con dispositivos de control para uso industrial y máquinas de realidad virtual para simulación de procesos. El uso de entornos virtuales facilita el aprendizaje y desarrollo de habilidades en ingeniería, particularmente en el uso de robots, sensores, actuadores, redes integradas. Los LV es creado por un sistema de software que simula el proceso real que se puede conectar a un equipo real o emulado.

Los LV también puede ayudar a introducir a los usuarios, con el concepto de sistemas ciber-físicos (CPS), que integra sistemas físicos con sus modelos computacionales, así como la interacción de personas a través de diferentes modalidades [1]. Los CPS son relevantes en el estudio ya que es una de las tecnologías fundamentales de la Industria 4.0. Para diseñar e implementar una LV enfocado a mecatrónica o robótica es necesario desarrollar e integrar modelos de los mecanismos, sensores y actuadores, para diseñar un sistema de control que pueda compensar la respuesta dinámica del sistema mecatrónico real. En cierto sentido, los LV puede parecerse a algunas funciones de un gemelo digital o virtual de un sistema físico. La función de los gemelos digitales es desarrollar modelos virtuales basados en equipos o procesos que incluyen la mayor parte de su información, lo que permite simular el comportamiento del sistema a través de una interfaz de realidad virtual donde se pueden realizar [2-4]. Las plataformas educativas que incorporan gemelos digitales se están convirtiendo en herramientas de formación que permiten el aprendizaje práctico de conceptos de ingeniería [5-7].

Actualmente existen diferentes laboratorios virtuales enfocados a las disciplinas de automatización, mecatrónica y robótica [8-10]. Institutos de investigación han desarrollado estaciones de trabajo robóticas integrando modelos CAD como el robot ABB IRB 120 y software de simulación para su ejecución [11]. Los colegios y universidades en China emplean sistemas de enseñanza con laboratorios virtuales, utilizando diferentes tecnologías para el aprendizaje, como el desarrollo de un sistema de laboratorio virtual robótico basado en Unity 3D [12]. Otros laboratorios virtuales involucran diferentes modelos de robots como una plataforma de simulación que integra un robot de seis grados de libertad y un robot SCARA de 4 grados de libertad permite enseñar la cinemática del robot [13]. Otro laboratorio virtual en sistemas mecatrónicos divide la plataforma virtual en dos módulos; un módulo para la enseñanza de la robótica y otro para la enseñanza de los componentes de los sistemas hidráulicos [14]. Otro ejemplo es un laboratorio virtual basado en un robot industrial, que está programado en MatLab, en una moderna línea de producción industrial como entorno experimental [15]. Los sistemas de control también se pueden enseñar en laboratorios virtuales utilizando modelos de sistemas mecatrónicos o robots [16]. Estas herramientas proporcionan acceso seguro y sin restricciones a los laboratorios y experimentos con los equipos, lo que reduce las consecuencias de los errores de los usuarios y mejora la experiencia al permitir que el experimento se repita desde diferentes ángulos [17, 18].

1.2 Definición del Problema

En la actualidad es necesario que los estudiantes, profesionistas y técnicos sean capacitados y entrenados en robótica, en particular para que desarrollen habilidades en control y en la programación de robots para ejecutar distintas tareas en procesos de manufactura. Un ambiente virtual presenta la ventaja de poder capacitar y entrenar en un robot manipulador como el UR5, sin necesidad de adquirir los equipos físicos, permitiendo la integración del laboratorio virtual a un entorno de industria 4.0, que puedan ser más competentes en la automatización y soporte de los procesos de manufactura.

1.3 Justificación

Los ingenieros que desarrollan tecnología y soportan sistemas de manufactura requieren desarrollar competencias en tecnologías de automatización y control aplicadas a robots manipuladores industriales. Así es como los laboratorios virtuales han tomado un mayor impulso debido al distanciamiento social derivado de la contingencia sanitaria del COVID19. El desarrollar laboratorios virtuales de robótica permite aprovechar las Tecnologías de Información y Comunicaciones (TICs) en favor de la capacitación en automatización y control.

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar un robot virtual en el contexto de la industria 4.0 para su aplicación en la enseñanza de la robótica industrial, enfocado al aprendizaje práctico de sistemas de control de robots.

1.4.2 Objetivos Específicos

- Desarrollar un robot virtual UR5 en el software Blender que trabaje de manera independiente en un sistema de cómputo, y que cuente con un sistema de control y de operación utilizando el software CODESYS en un esquema “hardware in the loop”, comunicando ambos sistemas mediante un protocolo de red industrial.
- Proponer técnicas de control de movimiento basadas en un control híbrido que combine autómatas de estados finitos para el control posición y el control de velocidad mediante control proporcional integral derivativo (PID).

1.5 Hipótesis

Un sistema de control híbrido que combina y optimiza técnicas de control de autómatas de estados finitos con control proporcional-integral-derivativo sintonizado con el principio de modelación interna e implementado con el estándar IEC61131 en una computadora personal, permite controlar la velocidad y posición de los ejes de movimiento, la posición y orientación cartesiana de un robot virtual articulado de 6 grados de libertad aplicable en capacitación y entrenamiento de sistemas de control de robots.

CAPÍTULO 2 MARCO TEÓRICO

2.1 Industria 4.0

La industria 4.0 es la transformación en el campo digital, representada por la digitalización de los procesos, automatización y el uso de las tecnologías de la información aplicada en la manufactura. Permite la flexibilidad en los procesos y la toma de decisiones en tiempo real. Algunas tecnologías que están impulsando esta transformación por las capacidades de interacción y el intercambio de información entre humanos y máquinas son; internet de las cosas (IOT), computación en la nube, gemelo digital, inteligencia artificial y machine learning [19], aunado a otras tecnologías como laboratorios virtuales, robots virtuales impresión 3D, ingeniería inversa [20]. La Industria 4.0 está transformando la forma en que las empresas desarrollan, fabrican, y mejoran sus productos.

2.2 Robots virtuales

La modelación y simulación es una fase importante en el desarrollo del diseño de un robot virtual, los prototipo 3D que incorpore un conjunto de características físicas de un robot además de incluir el entorno de trabajo que lo rodea posibilita verificar su funcionalidad real en un entorno computacional. Con el desarrollo de estos laboratorios virtuales se pueden realizar estudios específicos del robot, tales como, configuraciones singulares, análisis de movilidad; detalles del ensamblado de sus compontes, estudio del espacio de trabajo, localización de problemas de control y detección de posibles fallas. Ingenieros e investigadores que estudian a los robots manipuladores corroboran sus mecanismos, la cinemática inversa y directa, además de la dinámica a través de software de simulación como lo son Unity, 3ds Max, Cinema 4D y Blender [21-24]. Estos softwares permiten la modelación y simulación de diferentes robots virtuales (cartesiano, scara, brazo articulado o delta), permiten modificar los diseños 3D a través de iluminación, renderizado, animación y creación de gráficos tridimensionales, y además realizan cálculos de fricción, fuerzas, momentos, modelos de colisión, límites articulares, entre otras. A pesar de que el enfoque de estas plataformas es el modelado de aspectos

mecánicos, surgen inconvenientes al momento de desarrollar la aplicación de un manipulador bajo una acción de un controlador. Para integrar una técnica de control al modelado de un robot manipulador virtual es necesario utilizar un software especializado como CODESYS. Este tipo de software proporciona un entorno de programación adecuado para las técnicas de control de un robot UR5 [25], por ejemplo, y constituye una plataforma de bajo costo para la implementación de laboratorios de robótica y control [26], y otros tipos de laboratorio de modalidad en línea [27]. En la actualidad existen diferentes softwares para modelar y simular los robots virtuales, por ejemplo: SIMUROB, ROS, RoboDK, RokiSIM, entre otros. Este tipo de softwares están diseñados para modelar cualquier robot industrial, permiten el desarrollo de prototipos 3D para que el usuario pueda modificar o modelar sus propios diseños de robots virtuales. Además, estos softwares proporcionan herramientas para desarrollar animaciones 3D, en donde es posible simular un motor físico, incorporar la cinemática y dinámica de cuerpos rígidos y flexibles, aumentar el nivel de detalle del modelo del robot virtual, y mejorar el modelado incorporando instrumentación y control.

2.3 Niveles de control en los robots

En la actualidad existen niveles de control en los robots que permiten hacer más eficiente los programas de control en el controlador, permitiendo dar operabilidad y flexibilidad del robot dentro del diseño mecánico y de los sensores.

El objetivo del control del robot, a través de funciones en distintos niveles, es permitirle realizar una terea especifica.

El desarrollo del robot se ha realizado a nivel de modelación, control y lenguaje de programación. Dichos niveles permiten establecer una clasificación y jerarquía de funciones en el software del controlador.

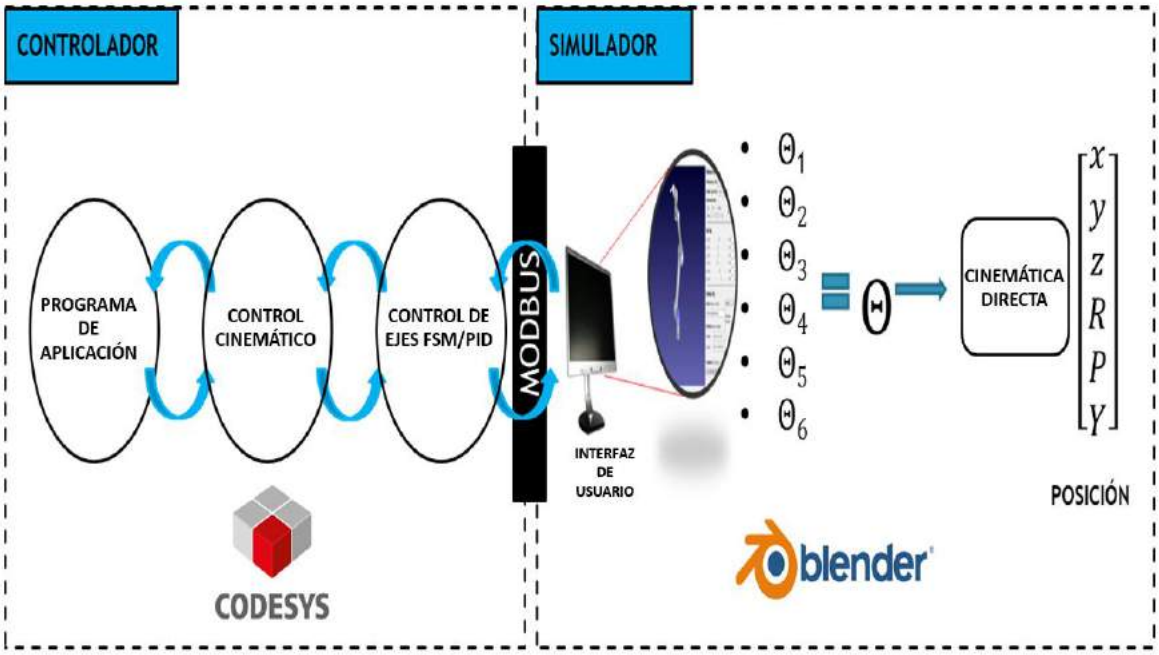


Figura 1. Esquema de controlador-simulador UR5.

Fuente: Elaboración propia.

Los programas en el controlador del robot pueden ser agrupados de acuerdo al nivel de control que realizan o en las formas para realizar su función; se consideran tres niveles de control: control de ejes de movimiento, control cinemático inverso y programa de aplicación como se muestra en la figura 1.

El uso de software nos permite programar los niveles de control, validando el código que se realiza para el modelado y control de posición y velocidad del motor CD. El software utilizado para la programación del controlador es el software CODESYS.

2.4 Control cinemático inverso

El modelo cinemático del robot UR5 es necesario para la simulación numérica y el control del robot virtual UR5. La cinemática directa del robot se puede obtener a través de la geometría y ubicación espacial de cada uno de los eslabones que componen el sistema, mientras que la cinemática inversa se puede lograr mediante la determinación de las variables articulares en función de la posición y orientación del efector final. Para asignar las coordenadas del robot, el sistema robótico se

encuentra en posición inicial y a cada eslabón del brazo robótico se le asignan los marcos que ayudan a calcular los parámetros de las articulaciones. Los parámetros de las uniones se obtienen mediante la convención de Denavit-Hartenberg (D-H) donde se obtienen las matrices de transformación homogéneas para cualquier robot manipulador [28].

La figura 2 muestra los marcos conjuntos y las indicaciones de parámetros para definir las matrices de transformación homogéneas.

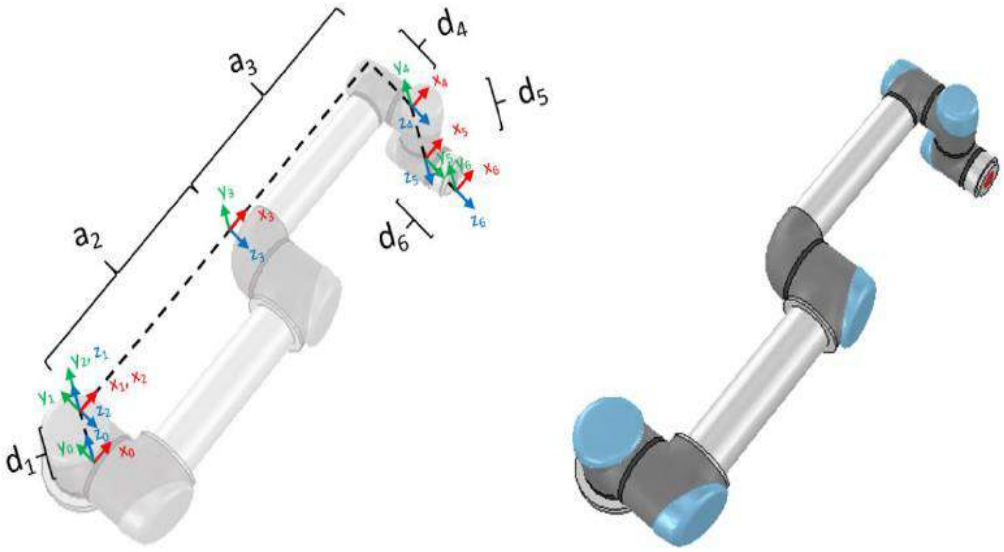


Figura 2. Representación de marcos del robot UR5.

Fuente: Villalobos [29].

El robot UR5 es un modelo de brazo articulado con 6 seis grados de libertad con articulaciones rotativa. Los parámetros calculados a través de los marcos de referencia están representados en la Tabla 5 mediante el algoritmo DH, donde i_n= representa los ejes que componen el robot, α_i= ángulo medido alrededor del eje x, a_i= distancia medida a lo largo del eje x, d_i= distancia medida a lo largo del eje z, θ_i= ángulo medido alrededor del eje z.

Tabla 1 Parámetros Denavit-Hartenberg

i_n	α_i	a_i	d_i	θ_i
1	$\frac{\pi}{2}$	0	L_1	θ_1
2	0	a_2	0	θ_2
3	0	a_3	0	θ_3
4	$\frac{\pi}{2}$	0	L_4	θ_4
5	$-\frac{\pi}{2}$	0	L_5	θ_5
6	0	0	L_6	θ_6

La matriz de transformación homogénea de la metodología DH viene dada por Ec. (1). Multiplicando las seis matrices de transformación homogénea representadas en Ec. (2), se obtiene la posición y rotación del efector final con respecto a la base del robot UR5.

$$ {}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cdot \cos a_i & \sin\theta_i \cdot \sin a_i & a_i \cdot \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cdot \cos a_i & \cos\theta_i \cdot \sin a_i & a_i \cdot \sin\theta_i \\ 0 & \sin a_i & \cos a_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

$$ {}^{0}T_6 = {}^{0}T_1 \cdot {}^{1}T_2 \cdot {}^{2}T_3 \cdot {}^{3}T_4 \cdot {}^{4}T_5 \cdot {}^{5}T_6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

Simplificando el cálculo de la matriz Ec. (2) es posible obtener las ecuaciones para la orientación Ec. (3) y la posición Ec. (4).

$$ r_{11} = c_1 c_{234} c_5 c_6 + c_6 s_1 s_5 - c_1 s_{234} s_6 \tag{3.1} $$

$$ r_{12} = -c_1 c_{234} c_5 c_6 - s_1 s_5 s_6 - c_1 c_6 s_{234} \tag{3.2} $$

$$ r_{13} = -c_1 c_{234} s_5 + c_5 s_1 \tag{3.3} $$

$$ r_{21} = c_{234} c_5 c_6 s_1 - c_1 c_6 s_5 - s_1 s_{234} s_6 \tag{3.4} $$

$$r_{22} = -c_{234}c_5s_1c_6 + c_5s_5s_6 - c_6s_1s_{234} \quad (3.5)$$

$$r_{23} = -c_{234}s_1s_5 - c_1c_5 \quad (3.6)$$

$$r_{31} = c_5c_6s_{234} + c_{234}s_6 \quad (3.7)$$

$$r_{32} = -c_5c_6s_{234} + c_{234}c_6 \quad (3.8)$$

$$r_{33} = -s_{234}s_5 \quad (3.9)$$

$$p_x = -c_1c_{234}s_5d_6 + c_5s_1d_6 + c_1s_{234}d_5 + s_1d_4 + c_1c_{23}a_3 + c_1c_2a_2 \quad (4.1)$$

$$p_y = -c_{234}s_1s_5d_6 - c_1c_5d_6 + s_1s_{234}d_5 - c_1d_4 + c_{23}s_1a_3 + c_2s_1a_2 \quad (4.2)$$

$$p_z = -s_{234}s_5d_6 - c_{234}d_5 + s_{23}a_3 + s_2a_2 + d_1 \quad (4.3)$$

La solución de la cinemática inversa del robot UR5 se obtiene mediante las ecuaciones de cinemática directa que se obtuvieron previamente, obteniendo las ecuaciones para la posición angular Ec. (9).

$$A_1 = p_x - d_6r_{13} \quad (5)$$

$$B_1 = d_6r_{23} - p_y \quad (6)$$

$$K_c = p_xc_1 + p_ys_1 + c_{234}s_5d_6 - s_{234}d_5 \quad (7)$$

$$K_s = p_z - d_1 + c_{234}d_5 + s_{234}s_5d_6 \quad (8)$$

$$\theta_1 = a\tan 2\,(A_1, B_1) \pm a\,tan\,2\left(\sqrt{A_1^2 + B_1^2 - d_4^2, d_4}\right) \quad (9.1)$$

$$\theta_2 = a\,tan\,2\,(K_s, K_c) - a\,tan\,2\,(a_3s_3, a_3c_3 + a_2) \quad (9.2)$$

$$\theta_3 = \pm a\tan 2\,(s_2, s_3) \quad (9.3)$$

$$\theta_4 = \theta_{234} - \theta_2 - \theta_3 \quad (9.4)$$

$$\theta_5 = \pm a\,tan\,2\,(s_5, c_5) \quad (9.5)$$

$$\theta_6 = a\,tan\,2\,(s_6, c_6) \quad (9.6)$$

2.4.1 Parámetros del robot UR5

La figura 2 muestra la referencia de los marcos que representan los eslabones que compone el robot UR5, mostrando los valores calculados en la Tabla 2.

Tabla 2 Valores calculados de Denavit-Hartenberg.

i_n	α_i	a_i	d_i	θ_i
1	$\frac{\pi}{2}$	0	0.089	θ_1
2	0	-0.425	0	θ_2
3	0	-0.329	0	θ_3
4	$\frac{\pi}{2}$	0	0.109	θ_4
5	$-\frac{\pi}{2}$	0	0.094	θ_5
6	0	0	0.082	θ_6

2.5 Control de eje independiente

Un enfoque de control implementado para la estabilización y el control de robots y otros sistemas mecánicos consiste en que el control de cada articulación se diseña de forma independiente [30,31]. Este método se fundamenta en que las entradas de control de cada articulación solo dependen de la medición del desplazamiento y la velocidad de la articulación correspondiente, por lo que el control empleado es el control de eje independiente. Debido a su estructura este tipo de esquemas de control ofrece algunas ventajas al usar el control de eje independiente como guardar la comunicación entre las diferentes articulaciones. Dado que la carga computacional de los controladores puede reducirse en las implementaciones reales solo se requiere hardware de bajo costo. Finalmente, el control de articulaciones independientes tiene la característica de la escalabilidad ya que los controladores en todas las articulaciones tienen la misma formulación. En este modelo del Robot UR5 se presenta el diseño de control de eje independiente basado en el modelo de articulación única.

2.6 Motor de corriente directa

Los actuadores utilizados para mover los ejes de movimiento del robot UR5 son motores de corriente directa (CD) que están formados por dos imanes pegados al estator del motor y un conjunto de bobinas de cobre en el rotor del motor. El funcionamiento se basa en la atracción o la repulsión entre el campo magnético que se genera en las bobinas por el paso de la electricidad y los imanes que se encuentran alrededor de ellas [32]. El motor de CD es el que más se usa en el accionamiento de los robots [33].

2.6.1 Modelación matemática de un motor de CD

Los motores de CD son dispositivos bien conocidos que se pueden usar como actuadores de muchos mecanismos en los que es necesario controlar la velocidad y la posición del eje de movimiento. El modelo matemático del motor de CD requiere dos ecuaciones: la primera para modelar la respuesta dinámica del circuito eléctrico y la segunda para modelar la respuesta dinámica del dispositivo mecánico. Estas ecuaciones se basan en las leyes de circuito de Kirchhoff y las leyes de movimiento de Newton [34]. La figura 3 muestra el modelo electromecánico de un motor de CD.

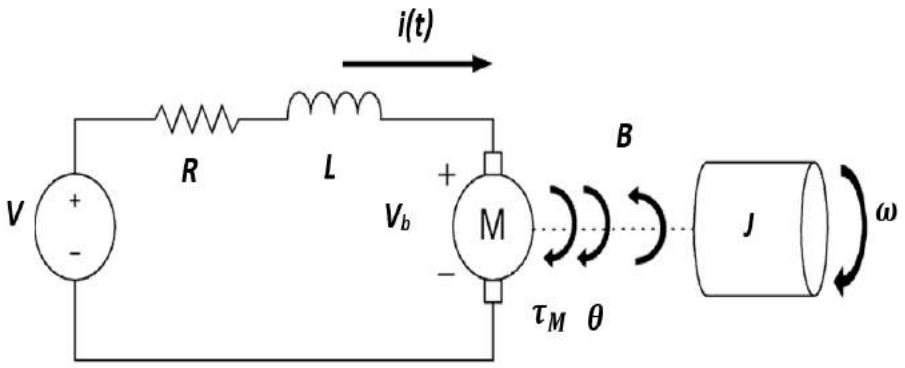


Figura 3. Circuito del motor CD.

Fuente: Elaboración propia.

El modelo ilustrado tiene los parámetros eléctricos y mecánicos que se muestran en la Tabla 3.

Tabla 3 Parámetros del motor CD

Símbolo	Parámetros
V	Voltaje
ω	Velocidad angular
i(t)	Corriente
L	Inductancia
R	Resistencia
J	Inercia del rotor
V_b	Fuerza electromotriz
B	Coeficiente de fricción
K_t	Constante del torque
K_b	Constante EMF

Aplicando las leyes de Kirchhoff una ecuación diferencial de primer orden Ec. (10), puede expresarse para el circuito eléctrico;

$$V(t) = R \cdot i(t) + L\frac{di(t)}{dt} + K_b \cdot \omega(t) \tag{10}$$

La fuerza electromotriz (EMF) está dada por:

$$V_b(t) = K_b \cdot \omega(t) \tag{11}$$

Ec. (12) se obtiene sustituyendo la Ec. (11) en la Ec. (10);

$$V(t) = R \cdot i(t) + L\frac{di(t)}{dt} + V_b(t) \tag{12}$$

Para la parte mecánica del motor de CD se considera otra ecuación diferencial de primer orden, la Ec. (13) donde $\tau_L =$ par de carga.

$$\tau_M - \tau_L = J\frac{d\omega(t)}{dt} + B \cdot \omega(t) \quad (13)$$

Considerando $\tau_L = 0$ y el torque del motor proporcional a la corriente:

$$\tau_M(t) = K_t \cdot i(t) \quad (14)$$

Sustituyendo la Ec. (14) en la Ec. (13);

$$K_t \cdot i(t) = J\frac{d\omega(t)}{dt} + B \cdot \omega(t) \quad (15)$$

Ec. (12) y Ec. (15) también se pueden representar en el dominio de la frecuencia aplicando la transformada de Laplace:

$$V\ (s) = R \cdot i(s) + Ls \cdot i(s) + V_b(s) \quad (16)$$

$$K_t \cdot i(s) = Js \cdot \omega(s) + B \cdot \omega(s) \quad (17)$$

Donde:

$$i(s) = \frac{V(s) - V_b(s)}{R + Ls} \quad (18)$$

Sustituyendo la Ec. (18) en la Ec. (17):

$$K_t \cdot \frac{V(s) - V_b(s)}{R + Ls} = Js \cdot \omega(s) + B \cdot \omega(s) \quad (19)$$

La figura 4 muestra un típico diagrama de bloques del modelo matemático de un motor de CD representando en el lado izquierdo el proceso electromagnético del estator y en el lado derecho el proceso mecánico del rotor, también se aprecia que la corriente del circuito eléctrico se convierte en par, se resta el par de carga y se integra el circuito mecánico y se tiene retroalimentación de la tensión inducida [35].

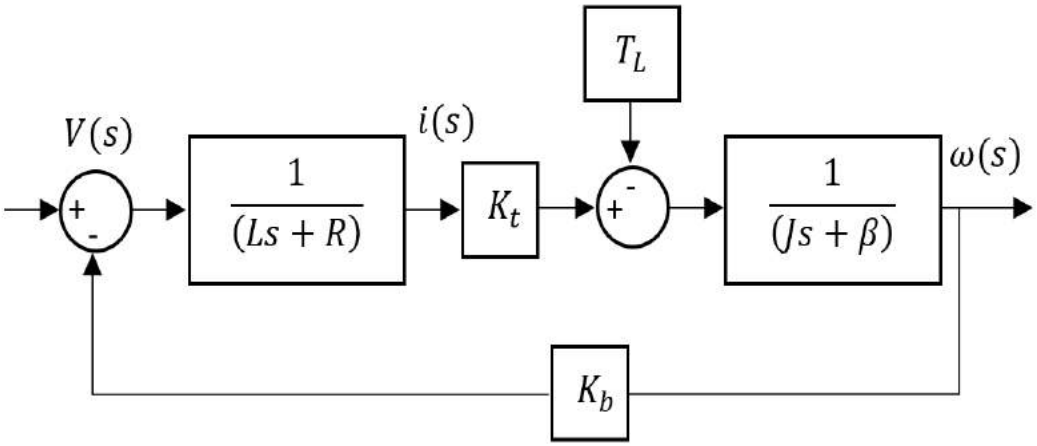


Figura 4. Diagrama de bloques del motor de CD.

Fuente: Elaboración propia.

La función de transferencia de la figura 4 es representada por:

$$G(s) = \frac{\omega(s)}{V(s)} = \frac{K_t}{JLs^2+JRs+LBs+BR+K_t \cdot K_b} \quad (20)$$

El sistema de segundo orden se puede reducir a un sistema de primer orden considerando el hecho de que la constante de tiempo mecánica tiene un orden de diferencia con respecto a la constante de tiempo eléctrica.

Mostrando el circuito eléctrico utilizando un bloque simplificado de ganancia constante (K_e) del motor de CD, el diagrama de la figura 5 corresponde a un sistema de primer orden en lazo cerrado.

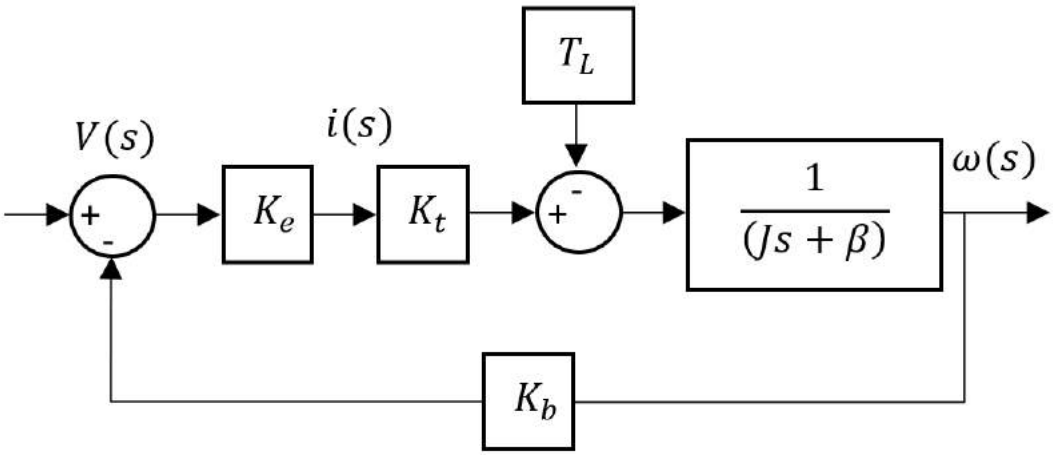


Figura 5. Diagrama de bloques simplificado del motor de CD.

Fuente: Elaboración propia.

De la figura 5, la relación entrada-salida esta por:

$$G(s) = \frac{\omega(s)}{V(s)} = \frac{K_e \cdot K_t}{Js + B + K_e \cdot K_t \cdot K_b} \tag{21}$$

Ec. (21) se puede manipular para que coincida con una función de transferencia de primer orden, donde $K =$ ganancia de primer orden:

$$G(s) = \frac{\omega(s)}{V(s)} = \frac{K}{\tau s + 1} \tag{22}$$

La ganancia y la constante de tiempo de la dinámica simplificada de primer orden está dada por las siguientes ecuaciones:

$$K = \frac{K_e \cdot K_t}{B + K_e \cdot K_t \cdot K_b} \tag{23}$$

$$\tau = \frac{J}{B + K_e \cdot K_t \cdot K_b} \tag{24}$$

2.7 Control proporcional, integral, derivativo (PID)

Los motores CD son típicamente controlados con controladores proporcional integral derivativo (PID). La estructura del controlador PID es un elemento que permite regular las señales de los sistemas, esto incluye la estabilidad de lazo cerrado, el ruido en las señales del sistema y por supuesto la capacidad del controlador para filtrar adecuadamente el ruido inherente a las señales en tiempo real [36]. En el caso de control de posición típicamente se usa un control proporcional, pero es posible utilizar otra técnica de control como la que se propone en este trabajo como las máquinas de estados finitos.

2.8 Máquina de estados finitos

Una máquina de estado finito (FSM) es una lógica matemática que es esencialmente un programa de computadora: representa una secuencia de instrucciones a ser ejecutadas donde cada instrucción depende del estado actual de la máquina y del actual estímulo [37]. Las técnicas de control tienen que implementarse en una arquitectura de hardware y software como procesadores, CODESYS y Blender.

2.9 Herramientas de software

Las herramientas de software para la creación de contenidos permiten desarrollar proyectos científicos y tecnológicos vanguardistas aplicando herramientas tecnológicas que impacten el entorno educativo y laboral. Los centros de investigación y entidades educativas han incorporado nuevas tecnologías de hardware y software que les permiten innovar y mejorar el desarrollo de sus proyectos de modo que logran diversificar la educación y capacitación para adquirir las habilidades y competencias a su medida de forma interactiva.

2.9.1 Blender

Para el desarrollo del robot UR5 se utilizan las herramientas de diseño asistido por computadora (CAD) que permitan optimizar los diseños de gráficos con representación tridimensional. Se utilizó la versión de Blender V2.79, que es un software de creación 3D gratuita y de código abierto, es una alternativa real al software de diseño 3D disponible comercialmente permite modelar y generar animación en 3D [38].

2.9.2 CODESYS

La plataforma utilizada para la programación del controlador es CODESYS V3.5 SP13, software de automatización líder independiente del hardware IEC 61131-3 para aplicaciones de controladores de desarrollo e ingeniería, uno de los sistemas más comunes para PLC y reguladores industriales que permite realizar la programación de sistemas autómatas de diferentes fabricantes. Este sistema embebido maneja la programación orientada a objetos en un entorno de tiempo real [39].

La aplicación desarrollada para este trabajo tiene programado el control de posición utilizando la metodología de Máquina de Estados Finitos y el control de velocidad empleando el algoritmo PID para controlar los actuadores de ejes de movimiento del robot UR5.

CAPÍTULO 3 MODELACIÓN Y CONTROL DE UN MOTOR CD

Los actuadores que conforman el robot pueden consistir en motores CD. Los modelos de estos motores se requieren discretizar para poderlos simular. El método de diferencias hacia atrás de Euler y el método de transformada bilineal se comparan considerando un tiempo de muestreo adecuado que no suele especificarse en los métodos de discretización. La discretización de diferencias hacia atrás de Euler de un modelo de motor CD de orden reducido se resolvió y se calculó el par de carga. El estudio del cálculo del par de carga del motor de CD se realizó utilizando una discretización efectiva de la dinámica del motor y la elección adecuada del tiempo de muestreo. La estimación en tiempo real del par de carga del motor se puede utilizar en una estrategia de control proporcional integral derivativo basada en un modelo interno de control.

3.1 Método de diferencias hacia atrás de Euler (EBD)

La discretización requiere obtener funciones de transferencia en los dominios “s” y “z”. Una forma alternativa de obtener las ecuaciones es aplicar directamente en el dominio del tiempo la diferenciación numérica definida por el método EBD.

La expresión de la primera derivada por EBD es:

$$\mathrm{f}'(\mathrm{X_N}) = \frac{\mathrm{f(X_N) - f(X_{N-1})}}{\Delta \mathrm{T}} \tag{25}$$

La ecuación diferencial de primer orden para el motor de CD considerando $y = \omega$ y $V = u$ por coherencia con la nomenclatura utilizada para salida y entrada, respectivamente, está dada por:

$$\tau \frac{dy(t)}{dt} + y(t) = K \cdot u(t) \tag{26}$$

Aplicando EBD a la Ec. (26):

$$\tau \cdot \left(\frac{y_n - y_{n-1}}{T}\right) + y_n = K \cdot u_n \tag{27}$$

Ec. (27) se puede escribir como:

$$y_n - y_{n-1} + y_n \cdot \left(\frac{T}{\tau}\right) = K \cdot u_n \cdot \left(\frac{T}{\tau}\right) \tag{28}$$

Simplificando Ec. (28):

$$y_n = y_{n-1} \cdot \left(\frac{\tau}{\tau+T}\right) + K \cdot u_n \cdot \left(\frac{T}{\tau+T}\right) \tag{29}$$

Las siguientes relaciones se definen como parametrizadas, la ecuación discretizada obtenida directamente con el método EBD puede tener los parámetros α, τ y T en tiempo discreto definiendo las siguientes relaciones:

$$\alpha = \frac{\tau}{\tau+T} \tag{30}$$

$$(1-\alpha) = \frac{T}{T+\tau} \tag{31}$$

El modelo discreto se puede expresar como:

$$y(n) = \alpha \cdot y_{k-1} + (1-\alpha) \cdot K \cdot u_n \tag{32}$$

3.1.1 Análisis de EBD en el dominio de la frecuencia

El análisis en el plano *w* se realiza para funciones de transferencia discretas. Puede obtenerse una función de transferencia en *z* a partir de la función de transferencia en *s* para la discretización de diferencias hacia atrás se define por la Ec. (33) y Ec. (34):

$$s = \frac{z-1}{zT} \tag{33}$$

$$s = \frac{1-z^{-1}}{T} \tag{34}$$

Sustituyendo la Ec. (34) en la Ec. (27):

$$G(z) = \frac{K}{\tau \cdot \left(\frac{1-z^{-1}}{T}\right)+1} = \frac{K \cdot z}{z \cdot (1+\gamma)-\gamma} \tag{35}$$

Donde:

$$\gamma = \frac{\tau}{T} \tag{36}$$

Considerando la ecuación de la transformación *w* en la Ec. (35):

$$G(z) = \frac{K \cdot \left(\frac{1+\frac{T}{2} \cdot w}{1-\frac{T}{2} \cdot w}\right)}{\left(\frac{\left(1+\frac{T}{2} \cdot w\right) \cdot \left(1+\frac{\tau}{T}\right)}{1-\frac{T}{2} \cdot w}\right) - \frac{\tau}{T}} \tag{37}$$

Simplificando la Ec. (37) se obtiene:

$$G(w) = \frac{K\cdot(Tw+2)}{w\cdot(2\tau+T)+2} \tag{38}$$

3.2 Método por transformada bilineal (BLT)

Para la simulación computacional diferentes métodos son utilizados, la transformación bilineal es un método de discretización que se usa convencionalmente para convertir una función de transferencia en el dominio de Laplace en una función de transferencia en el dominio z a partir de la cual se pueden obtener ecuaciones de diferencias.

El método BLT se utiliza en el diseño de filtros digitales para obtener las características de respuesta deseadas [40], realiza un mapeo entre el plano *s* y el plano *z* de acuerdo con la Ec. (39) donde T es el período de muestreo del sistema de control de tiempo discreto considerado:

$$s = \frac{2}{T}\frac{(z-1)}{(z+1)} \tag{39}$$

Aplicando la transformada bilineal a la Ec. (22):

$$G(z) = \frac{K}{\frac{2\cdot\tau}{T}\left(\frac{z-1}{z+1}\right)+1} \tag{40}$$

Las variables de entrada-salida se cambian a "u" y "y" para hacer coincidir la nomenclatura de las ecuaciones discretas con la terminología convencional para un sistema de entrada-salida.

La Ec. (41) se obtiene:

$$\frac{\omega(s)}{V(s)} = \frac{y(s)}{u(s)} = \frac{K\left(1+s^{-1}\right)}{(1+\sigma)+s^{-1}(1-\sigma)} \tag{41}$$

Donde:

$$\sigma = \frac{2\tau}{T} \tag{42}$$

Aplicando la transformada inversa de z se obtiene una expresión para la salida:

$$y_k = \frac{(\sigma-1)\cdot y_{k-1}+K(u_k+u_{k-1})}{1+\sigma} \tag{43}$$

3.2.1 Análisis BLT en el dominio de la frecuencia

La transformada *w* es una función de mapeo para el dominio *z*, el plano *w* se usa en lugar del plano *s*. La transformación *w* se usa para el análisis de sistemas de tiempo discreto [41]. La transformación *w* es una transformación bilineal dada por:

$$z = \frac{\left(1+\frac{T}{2}\cdot w\right)}{\left(1-\frac{T}{2}w\right)} \tag{44}$$

Sustituyendo la Ec. (44) en la Ec. (41) se obtiene:

$$G(w) = \frac{K\left(\frac{1+\frac{T}{2}\cdot w}{1-\frac{T}{2}\cdot w}+1\right)}{\left(\frac{1+\frac{T}{2}\cdot w}{1-\frac{T}{2}\cdot w}\right)\cdot(1+\sigma)+(1-\sigma)} \tag{45}$$

La simplificación de la Ec. (45) resulta en:

$$G(w) = \frac{K}{\tau\cdot w+1} \tag{46}$$

Dado que el mapeo de frecuencia de *s* a *z* y el mapeo de w se definen mediante transformadas bilineales se obtiene un sistema de primer orden en w, por lo tanto, la respuesta de frecuencia para el modelo original en el plano *s* es la misma que la respuesta de frecuencia en la frecuencia w (w también se llama frecuencia ficticia).

3.3 Comparación de métodos de discretización EBD y BLT

Este estudio compara dos métodos de discretización para el modelo de un motor de CD apto para ser utilizado en la implementación de mecanismos virtuales. La evaluación de las ecuaciones obtenidas previamente, Eq. (29) y Eq. (43), se realiza para comparar los dos métodos de discretización en el estudio, se selecciona un tiempo de muestreo de τ / 15 y esta selección se explica más adelante en la sección comparación en el dominio del tiempo.

3.3.1 Comparación en el dominio del tiempo

Para comparar tanto la discretización obtenida con EBD como la discretización obtenida con BLT se considera una entrada de paso unitario, la respuesta de intervalo de tiempo continuo es:

$$y(t) = 1 - e^{\frac{-T}{\tau}} \tag{47}$$

La solución exacta para la respuesta de paso de tiempo discreto es:

$$y(n) = 1 - e^{\frac{-nT}{\tau}} \tag{48}$$

Las simulaciones para los sistemas de tiempo discreto de EBD y BLT se llevan a cabo utilizando el período de muestreo T= τ/15, los valores de respuesta se muestran en la Tabla 4, las columnas se refieren a la solución exacta, la transformada bilineal (BLT) y la aproximación de diferencias hacia atrás (EBD) además de los errores entre BLT y EBD. El error de los métodos analizados y de la integral del error se muestran en la figura 6 y la figura 7, respectivamente.

Tabla 4 Valores numéricos y errores para el período de muestreo T= τ/15

n	*SOLUCIÓN EXACTA*	*TRANSFORMADA BILINEAL*		*DIFERENCIAS HACIA ATRÁS DE EULER*	
	y_n	$y(k_n)$	e_r	$y(k_n)$	e_r
1	0.0091	0.0045	0.0045	0.0088	0.0002
2	0.0176	0.0134	0.0042	0.0171	0.0005
3	0.0256	0.0216	0.0039	0.0249	0.0007
4	0.0331	0.0294	0.0037	0.0322	0.0009
5	0.0401	0.0366	0.0034	0.0390	0.0010
6	0.0466	0.0434	0.0032	0.0454	0.0012
7	0.0528	0.0497	0.0030	0.0514	0.0013
8	0.0585	0.0557	0.0028	0.0571	0.0014
9	0.0639	0.0612	0.0026	0.0624	0.0015
10	0.0689	0.0664	0.0024	0.0673	0.0015

11	0.0736	0.0712	0.0023	0.0719	0.0016
12	0.0779	0.0758	0.0021	0.0763	0.0016
13	0.0821	0.0800	0.0020	0.0804	0.0016
14	0.0859	0.0840	0.0019	0.0842	0.0016
15	0.0895	0.0877	0.0017	0.0878	0.0016
16	0.0928	0.0912	0.0016	0.0912	0.0016
17	0.0960	0.0944	0.0015	0.0943	0.0016
18	0.0989	0.0975	0.0014	0.0973	0.0016
19	0.1017	0.1003	0.0013	0.1000	0.0016
20	0.1043	0.1030	0.0012	0.1026	0.0016

La Tabla 4 muestra los valores obtenidos usando un tiempo de muestreo T= τ/15, para n=1 a n=20. A medida que aumenta el número de iteraciones las aproximaciones se acercan a la solución exacta. El error generado por los métodos BLT y EBD también se compara y analiza con respecto a la solución exacta mostrando que el método EBD tiene un menor error en comparación con el método BLT, se puede hacer una observación más detallada si se grafican los valores de error.

La figura 6 muestra los valores de tiempo t < τ, el método EBD tiene un error menor en comparación con el método BLT pero para los valores de tiempo t > τ el método BLT obtiene el error más bajo.

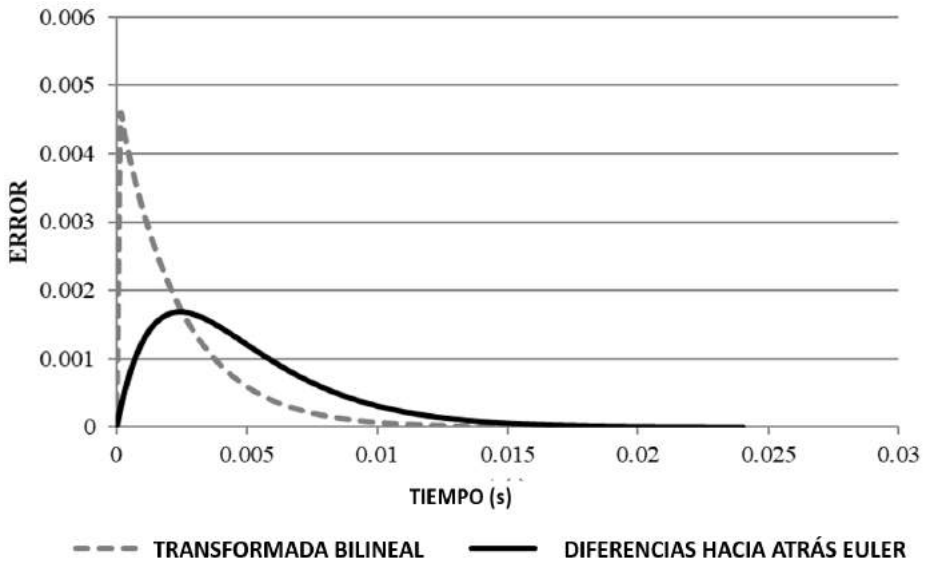


Figura 6. Error con T= τ/15.

Fuente: Elaboración propia.

La figura 7 muestra los valores de la integración del error (error acumulativo), se puede observar que en la respuesta transitoria el BLT acumula un valor mayor que EBD pero en el largo plazo los errores acumulativos tienden a converger.

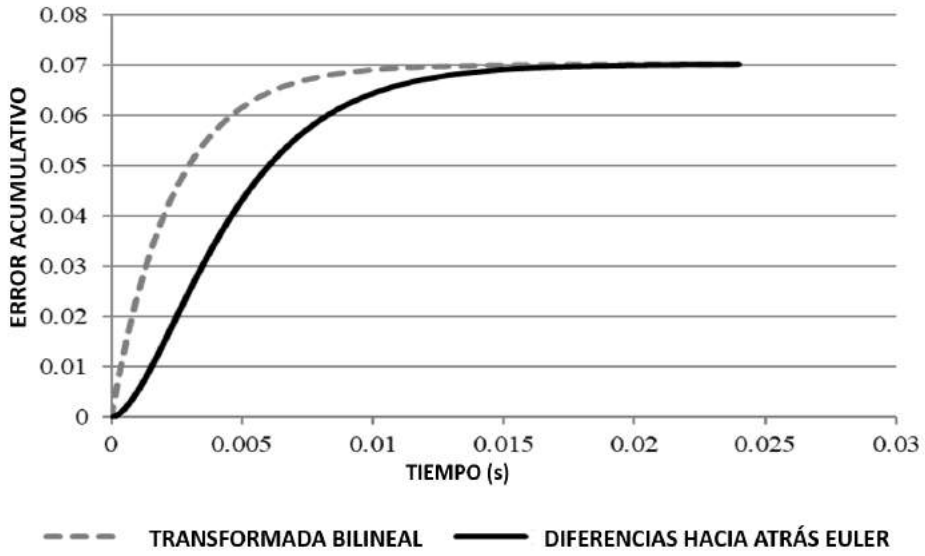


Figura 7. Integral del error con T = τ/15.

Fuente: Elaboración propia.

Otro análisis interesante se puede realizar seleccionando diferentes períodos de muestreo la Tabla 5 muestra los valores obtenidos para la evaluación numérica en la muestra n = τ/T para la solución exacta, los métodos EBD y BLT para cada uno de los siguientes periodos de muestreo T= τ/5, T= τ/10, T= τ/15, T= τ/20 y T=τ/25. Los valores de respuesta y los errores de EBD y BLT se grafican respectivamente en la figura 8 y la figura 9.

Tabla 5 Valores numéricos EBD y BLT para períodos de muestreo en n = τ/T.

$\frac{\tau}{n}$	***SOLUCIÓN EXACTA***	***TRANSFORMADA BILINEAL***		***DIFERENCIAS HACIA ATRÁS EULER´S***	
	y_n	$y(k_n)$	e_r	$y(k_n)$	e_r
$\tau/5$	0.0895	0.0839	0.0056	0.0847	0.0048
$\tau/10$	0.0895	0.0868	0.0027	0.0870	0.0025
$\tau/15$	0.0895	0.0877	0.0017	0.0878	0.0016
$\tau/20$	0.0895	0.0882	0.0013	0.0882	0.0012
$\tau/25$	0.0895	0.0884	0.0010	0.0885	0.0010

En la figura 8 se puede observar que para períodos de muestreo bajos (T= τ/5, T= τ/10) el método EBD tiene una diferencia menor con respecto a la solución exacta. Comparado con el método BLT, para períodos de muestreo altos (T= τ/15, T= τ/20 y T= τ/25) los valores de amplitud de EBD y BLT se acercan mucho y con la misma desviación que la solución exacta, por lo que se necesita una observación detallada adicional.

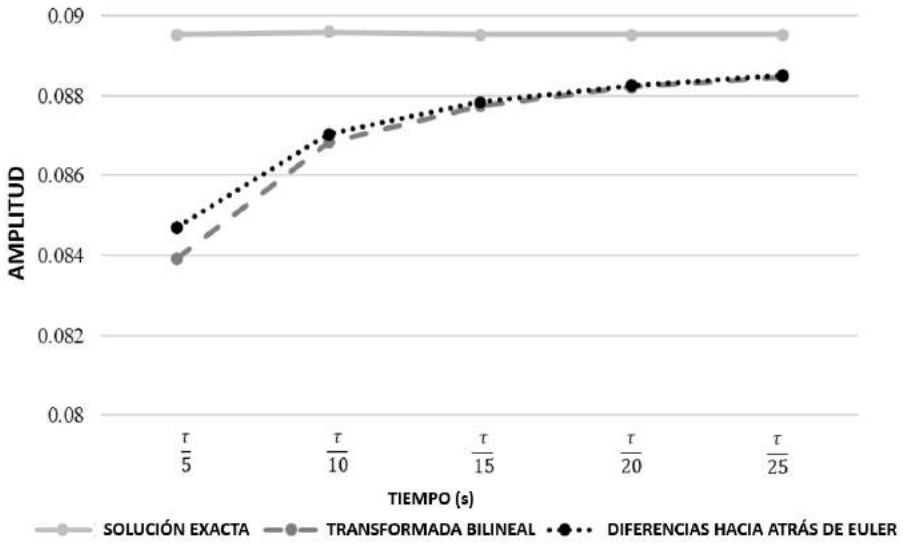


Figura 8. Períodos de muestreo evaluados en n= τ/T para EBD y BLT.

Fuente: Elaboración propia.

En la figura 9 se muestra una comparación de los errores de EBD y BLT con respecto a la solución exacta, para un tiempo de muestreo de τ/5 se obtiene el porcentaje de error para ambos métodos, el método BLT con un 6.25% en comparación con el método EBD que es solo 5.38%, esto indica que EBD es más preciso que BLT, considerando ahora un tiempo de muestreo de τ/15 el método EBD y BLT tienen un error ligeramente inferior al 1.89%. A medida que se reduce el tiempo de muestreo los valores de error se acercan y convergen al mismo valor.

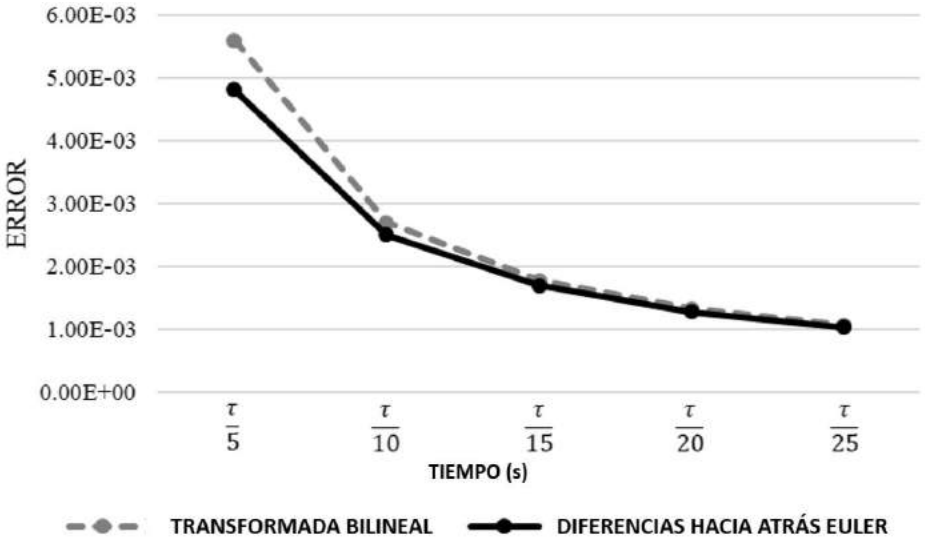


Figura 9. Valores de error para periodos de muestreo n= τ/T.

Fuente: Elaboración propia.

3.3.2 Comparación en el dominio de la frecuencia

La figura 10 muestra los diagramas de Bode con la respuesta de frecuencia de las funciones de transferencia de la Ec. (38) y Ec. (46), correspondiente a las respuestas de frecuencia de BLT y EBD respectivamente. Se realiza una comparación de la respuesta de frecuencia y muestra respuestas similares tanto de magnitud como de fase en el ancho de banda de interés, esto está por debajo de la frecuencia de corte del sistema bajo análisis. Se nota una magnitud y fase diferente, pero para frecuencias superiores a la frecuencia de corte, esto se espera debido a la existencia del cero. Este análisis de frecuencia valida la equivalencia de ambos métodos de discretización.

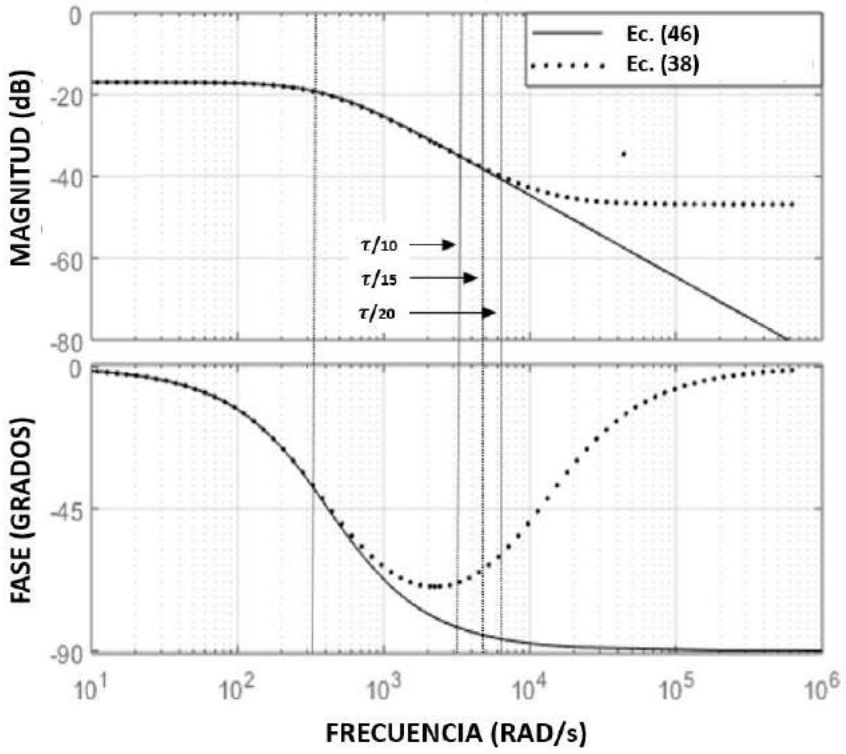


Figura 10. Comparación método BLT y EBD tanto en magnitud como en fase.

Fuente: Elaboración propia.

La frecuencia de corte del modelo en estudio es de 327 rad/s, considerando una frecuencia de muestreo dada por $\tau/10$ se obtiene un ancho de banda de 3,270 rad/s, por debajo de esta frecuencia de muestreo coincide la magnitud y fase de ambos sistemas BLT y EBD. Ahora considerando que un muestreo de $\tau/15$ genera un ancho de banda de 4,905 rad/s, se puede observar que la magnitud entre ambos sistemas aún es bastante cercana pero ahora tienen diferencias de 45° (π/4) en la fase. Finalmente, considerando una frecuencia de muestreo de $\tau/20$ que corresponde a un ancho de banda de 6.540 rad/s, la magnitud y fase comienza a diferenciarse para frecuencias más altas. De este análisis de frecuencia se puede decir que la discretización EBD tiene mejor margen de ganancia y margen de fase que la discretización BLT.

3.4 Estimación de carga dinámica del motor CD

El método EBD permite una discretización de un modelo de motor de CD de orden reducido con suficiente precisión. Un posible uso interesante del modelo discretizado es derivar una expresión para estimar el par de carga del motor. Las variaciones de carga se pueden simular considerando la posición variable del efector final y su proyección de carga en cada una de las articulaciones. Esta aplicación se presenta en esta sección.

3.4.1 Cálculo del par de carga del motor con el modelo de orden reducido

Una ecuación de diferencia para la velocidad del motor considerando el modelo de orden reducido, se puede expresar considerando T_L como una variable de perturbación:

$$\omega_n = \alpha \cdot \omega_{n-1} + (1-\alpha) \cdot [(V - K_b \cdot \omega_{n-1}) \cdot K_t \cdot K_e - T_L] \tag{49}$$

El par de carga, T_L, es otra entrada que se resta del par electromagnético generado por el circuito eléctrico del inducido. Si se puede medir la velocidad del motor, se puede utilizar la Ec. (49), para resolver el par de carga, ahora se puede considerar el par de carga estimado:

$$T_{LE} = (V - K_b \cdot \omega_{n-1}) \cdot K_t \cdot K_e - \frac{\omega_n - \alpha \cdot \omega_{n-1}}{1-\alpha} \tag{50}$$

3.4.2 Estimación de carga a velocidad variable y carga variable

Se realizaron pruebas de simulación para validar la respuesta de la ecuación de par de carga derivada de escenarios de velocidad variable y carga variable. En una prueba de velocidad variable con una carga constante, se aplica una rampa de voltaje (Va) para acelerar el motor de velocidad cero a una velocidad máxima y luego para desacelerar el motor de nuevo a velocidad cero. Para esta prueba, se aplica una señal de par de carga simulada con un valor medio de 8 y una variación aleatoria de +/- 2. La figura 11 muestra la estimación de la velocidad del motor (W) y el par de carga estimado (Tlest). Como se observa, la carga estimada se sitúa en torno a un valor medio con variaciones pequeñas y limitadas. Esta prueba muestra que es posible calcular la perturbación aplicada a partir de la ecuación propuesta incluso durante cambios continuos de velocidad.

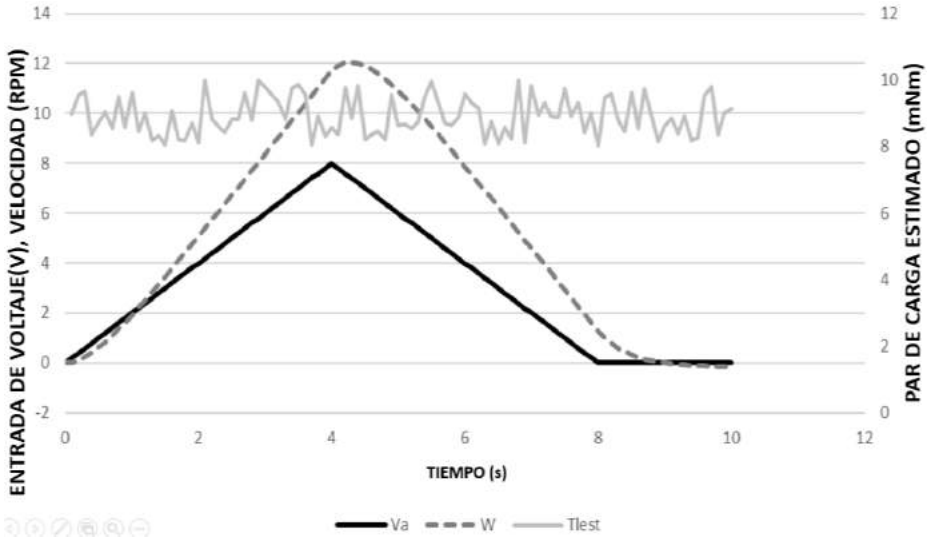


Figura 11. Estimación de carga con velocidad variable.

Fuente: Elaboración propia.

Se realizó una segunda prueba para verificar la correcta estimación de carga con la ecuación propuesta bajo una carga variable. Esta segunda prueba se realiza aplicando voltaje de entrada constante y variando el par de carga de entrada. La figura 12 muestra cómo la variación en la carga del motor afecta la velocidad del motor de desarrollo, como se esperaba para un motor real. El motor desacelera

cuando se aumenta la carga y el motor acelera cuando se reduce la carga aplicada. Esta prueba valida la ecuación propuesta; la respuesta simulada reproduce el comportamiento físico esperado de un motor de CD bajo carga variable.

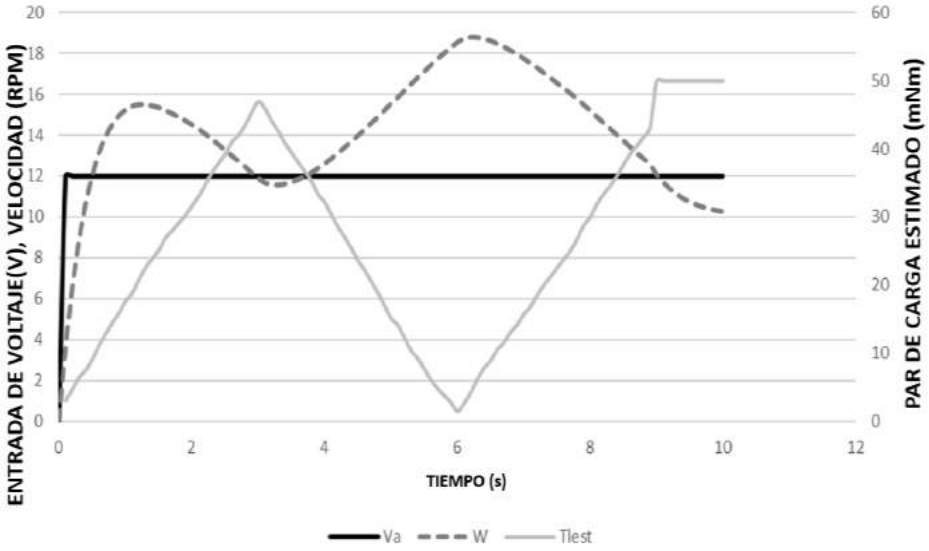


Figura 12. Perturbación de velocidad por variación de carga.

Fuente: Elaboración propia.

Las pruebas anteriores de velocidad variable y carga variable sirvieron para validar la ecuación de par de carga. Por esta razón, la ecuación de carga aquí propuesta se puede utilizar para estimar el par de carga del motor en tiempo real.

3.5 Control de velocidad con estimación de carga

Hay muchos métodos para ajustar los parámetros PID [42, 43]. El controlador PID se expresa en términos de la señal de error (e) y tiene una ganancia proporcional (K_P), una ganancia integral (K_I) y una ganancia derivada (K_D). El controlador PID continuo ideal tiene la forma;

$$u(t) = K_P \cdot e(t) + K_I \int e(t) \cdot dt + K_D \cdot \frac{de(t)}{dt} \tag{51}$$

Aplicando la transformada de Laplace a la Ec. (52), se obtiene una ley de control en el dominio de la frecuencia;

$$u(s) = e(s) \cdot \left[K_P + \frac{K_I}{s} + K_D \cdot s\right] \tag{52}$$

Para simplificar, Eq. (52) se puede convertir a Eq. (53) considerando una ganancia de controlador común única:

$$u(s) = e(s) \cdot K_C \left[1 + \frac{1}{T_i \cdot s} + T_d \cdot s\right] \tag{53}$$

Con los parámetros de ajuste: ganancia del controlador (K_C), constante de tiempo integral (T_i) y constante de tiempo derivada (T_d). Eq. (53) se puede reorganizar como Eq. (54) para facilitar la discretización de la ley de control:

$$T_i \cdot u(s) = [K_C(T_i \cdot T_d \cdot s^2 + T_i \cdot s + 1)] \cdot e(s) \tag{54}$$

Una representación en tiempo discreto de la ecuación anterior (54) aplicando la diferenciación numérica EBD conduce a:

$$T_i s \cdot \left(\frac{u_K - u_{K-1}}{T}\right) = K_C \cdot \left[T_i \cdot T_d \left(\frac{e_K - 2e_{K-1} + e_{K-2}}{T^2}\right) + T_i \cdot \left(\frac{e_K - e_{K-1}}{T}\right) + e_K\right]$$

(55)

Para la equivalencia de integración trapezoidal, el último término de error puede estimarse como un promedio de dos muestras de error dado Ec. (56), y expresarse como Ec. (57):

$$e_K = \frac{e_K + e_{K-1}}{2} \tag{56}$$

$$T_i \cdot \left(\frac{u_K - u_{K-1}}{T}\right) = K_C \cdot \left[T_i \cdot T_d \left(\frac{e_K - 2e_{K-1} + e_{K-2}}{T^2}\right) + T_i \cdot \left(\frac{e_K - e_{K-1}}{T}\right) + \frac{e_K + e_{K-1}}{2}\right] \tag{57}$$

El principio de control de modelo interno (IMC) se puede utilizar para ajustar los PID para sintetizar controladores robustos. Se pueden lograr respuestas rápidas y no oscilantes incluso con variaciones en los parámetros del motor [44-46]. La ley de control convencional de un controlador proporcional-integral-derivativo (PID) de tiempo discreto es:

$$u_K = u_{K-1} + K_C \cdot \left[(e_K - e_{K-1}) + \frac{T}{T_I} \cdot \left(\frac{e_K + e_{K-1}}{2}\right) + \frac{T_d}{T} \cdot (e_K - 2e_{K-1} + e_{K-2})\right] \tag{58}$$

Los términos proporcional e integral se calculan utilizando el principio IMC-PID, es decir, las ganancias de los términos proporcional e integral se pueden configurar

para obtener una respuesta críticamente amortiguada, lo que implica un rendimiento óptimo y estable del sistema.

$$K_c = K_d = \frac{1}{K_m} \cdot \frac{\tau}{\tau_{LC}} \quad (59)$$

$$T_i = \frac{\tau_m + \tau_{LC}}{2} \quad (60)$$

Donde:

τ_{LC}= Constante de tiempo de bucle cerrado deseada

K_m= ganancia del modelo mecánico

τ_m = Constante de tiempo del modelo mecánico

La estimación de par de carga derivada, se puede calcular en tiempo real y usarse en un término derivado del IMC-PID para compensar las perturbaciones causadas por la variación en la carga. Una ganancia derivada se puede definir por el índice de cambio de la variación de la estimación de carga, como se especifica por:

$$D = \frac{K_d \cdot (T_{LE_n} - T_{LE_{n-1}})}{T} \quad (61)$$

Donde:

D= Término derivativo con estimación de par de carga

K_d = Ganancia derivada en perturbaciones de carga

T_{LE_n} = Par de carga estimado de la muestra actual

$T_{LE_{n-1}}$= Par de carga estimado de la muestra anterior

Como puede verse en la figura 13, la velocidad angular de la articulación se controla bajo un cambio de referencia y con una pequeña variación continua no despreciable de la carga. En esta respuesta del sistema de control de circuito cerrado, cuando cambia el par de carga estimado, el término derivativo compensa el aumento o la disminución del par de carga mientras la velocidad del motor se mantiene constante en la velocidad deseada.

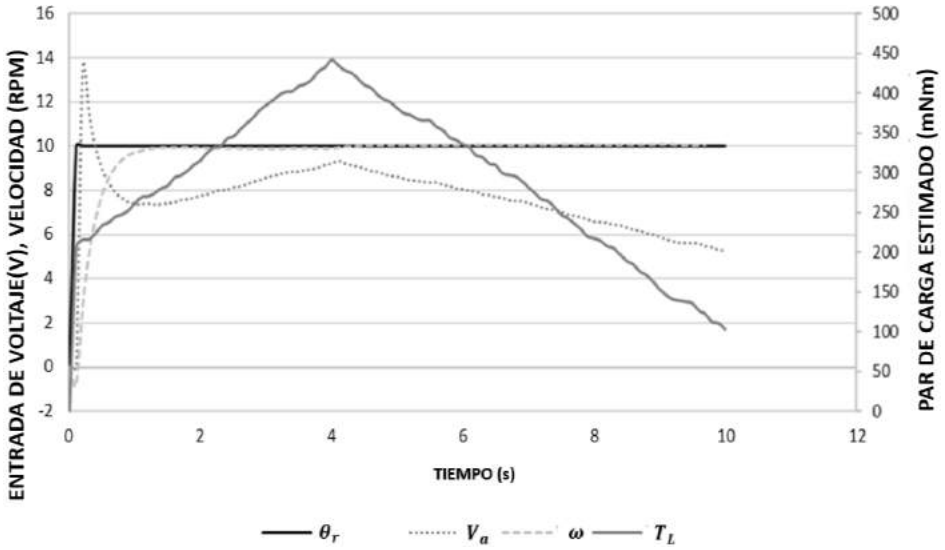


Figura 13. Control de velocidad IMC-PID.

Fuente: Elaboración propia.

Además, se puede utilizar un filtro de respuesta infinita al impulso (Infinite Impulse Response, IIR), con el parámetro β, como polo del filtro, para suavizar posibles variaciones repentinas debido al ruido del sensor de velocidad:

$$T_{LE_{ok}} = \beta \cdot T_{LE_{ok-1}} + (1-\beta) \cdot T_{LE_{ik}} \tag{62}$$

Donde:

$T_{LE_{ok}}$ = Par de carga estimado de salida del filtro

$T_{LE_{ik}}$= Par de carga estimado de entrada del filtro

El filtro IIR se puede especificar considerando una constante de tiempo que sea al menos la mitad de la constante de tiempo mecánica del motor de CD para asegurar que la frecuencia de corte de los sistemas sea mayor que la frecuencia de corte del sistema mecánico. Por lo tanto, el parámetro β en la Ec. (62):

$$\beta = \frac{\frac{\tau_m}{2}}{\frac{\tau_m}{2}+T} \tag{63}$$

La figura 14 muestra el esquema de control PID considerando las ganancias proporcionales e integrales a ser calculadas con el procedimiento IMC convencional

para un proceso de primer orden, y el término derivado calculado usando la ecuación propuesta para estimar el par de carga del motor. Se trata de un controlador de velocidad robusto para aplicaciones de control de motores de CD de carga variable.

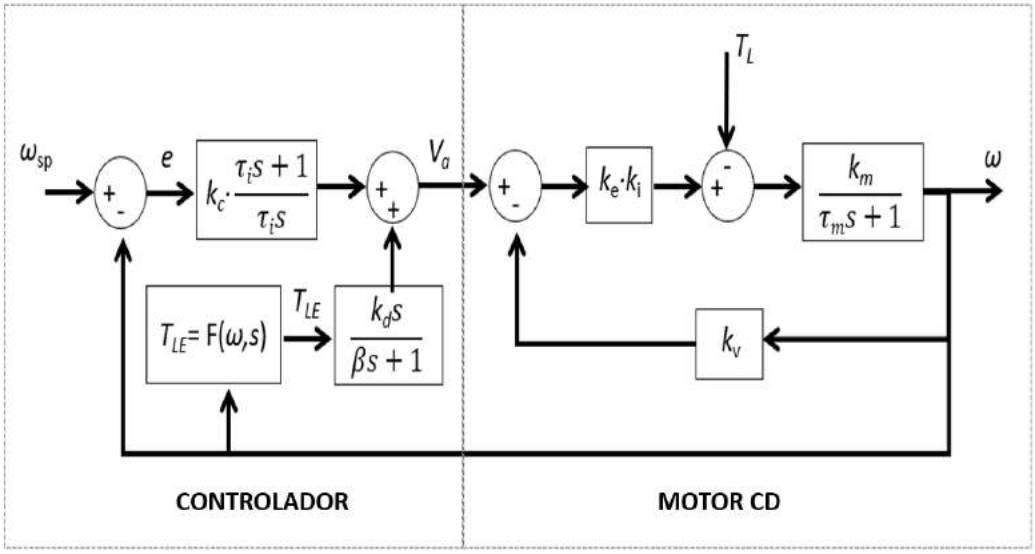


Figura 14. Controlador IMC-PID con la estimación del par de carga propuesta.

Fuente: Elaboración propia.

Se realiza una simulación computacional con la ecuación de control PID discreta, considerando los términos proporcional e integral utilizando el criterio de ajuste IMC y con el término derivativo propuesto. La respuesta del controlador de velocidad bajo variación de carga se muestra en la figura 15.

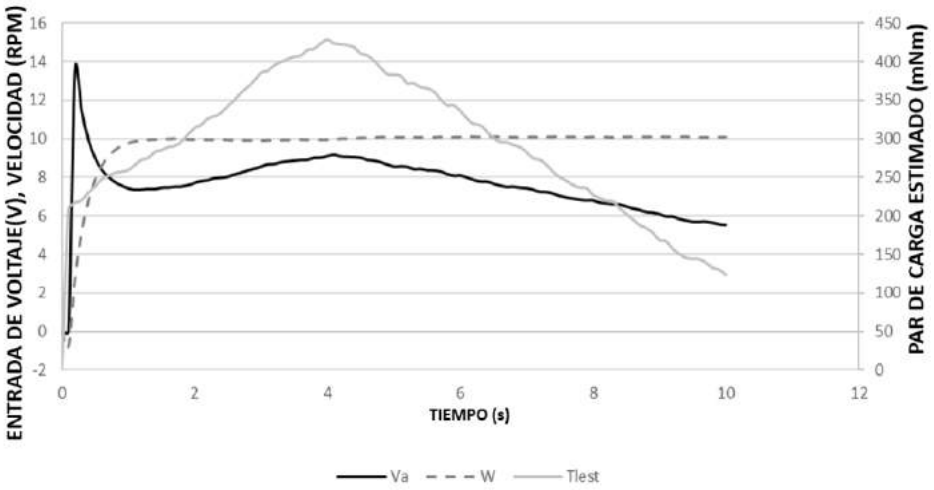


Figura 15. Control de velocidad PID-IMC bajo variación de carga continua.

Fuente: Elaboración propia.

Como se puede ver en la figura 15, la respuesta de velocidad del motor se controla bajo un cambio de referencia y con una variación continua de la carga. En esta respuesta del sistema de control de lazo cerrado, cuando el par de carga estimado cambia, el término derivativo compensa el aumento o disminución del par de carga mientras la velocidad del motor se mantiene estable a la velocidad deseada. El control PID-IMC considerando el término derivativo basado en la estimación en tiempo real del par de carga condujo a un buen desempeño del control de velocidad del motor ante variaciones del par de carga real.

CAPÍTULO 4 CONTROL CASCADA DE POSICIÓN DE UN MOTOR CD

Los motores de CD son actuadores adecuados para aplicaciones de control de movimiento ya que su respuesta dinámica es bien conocida y más fácil de controlar que otros motores eléctricos. Existen diversos métodos para controlar la posición y la velocidad de los motores de CD, uno de ellos es el control PID. Los motores de CD son más sencillos de modelar matemáticamente y, por lo tanto, los controladores PID se pueden ajustar con el IMC u otras técnicas de ajuste. Este capítulo presenta una estrategia de control en cascada para el control de posición de un motor de CD que se basa en un autómata de estado finito (FSM). La referencia de velocidad calculada por el lazo de posición externo de la máquina de estado finito se aplica al lazo de control de velocidad PID basado en el IMC. Este control combinado de FSM-IMC-PID se compara con un esquema convencional de control cascada IMC-P para el control de posición y velocidad. El esquema de control de posición y velocidad propuesto tiene el potencial de optimizar muchas aplicaciones diversas de control de movimientos.

4.1 Control cascada de velocidad y posición de un motor de CD

El motor de CD tiene un sistema de posicionamiento angular que comúnmente se controla a través de algoritmos de control PID para optimizar su funcionamiento [47]. Dado que la velocidad del motor está integrada en el proceso mecánico, solo se necesita un controlador proporcional para controlar la posición de un eje de motor de CD. La figura 16 muestra la estrategia de control P-PID en cascada completa. La velocidad angular del motor se puede controlar mediante un lazo interno con un controlador IMC-PID, mientras que la posición angular del motor se controla mediante un lazo externo con un controlador P. En este esquema de control, θ_{SP} y ω_{SP} son las señales de referencia para los controladores externos e internos, respectivamente, siendo $e_1 = \theta_{SP} - \theta$ y $e_2 = \omega_{SP} - \omega$. El error e_1 ingresa al bloque del controlador P, que controla la posición ajustando la velocidad del motor; por lo tanto, la salida del controlador P se pasa como velocidad de referencia ω_{SP}. El error

de velocidad e_2 ingresa al bloque del controlador PID; la salida del controlador PID aplica un voltaje al motor de CD para afectar la velocidad. El modelo de motores de CD requiere que la entrada de voltaje desarrolle una velocidad ω, que está integrada para dar una posición angular θ del eje de movimiento del motor de CD. Además, el par de carga actúa como una variable de perturbación que puede compensarse directamente mediante el término derivativo del controlador de velocidad PID.

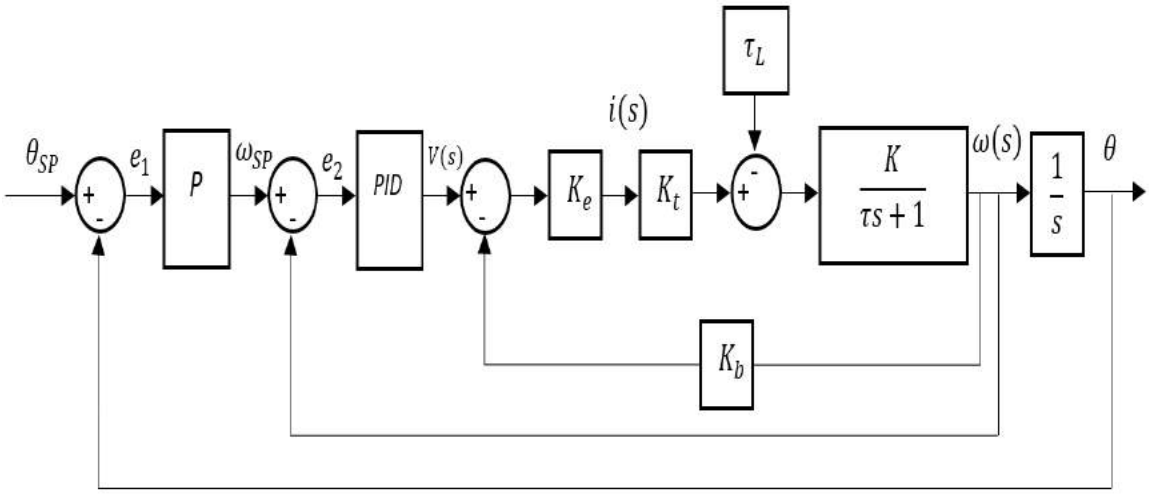


Figura 16. Esquema de control de posición en cascada (P) y en velocidad (PID).

Fuente: Elaboración propia.

La ley de control para el controlador de posición viene dada por la Ec. (64), considerando una ganancia proporcional multiplicada por la diferencia de error entre dos muestras de error consecutivas:

$$u_K = u_{K-1} + K_C(e_K - e_{K-1}) \tag{64}$$

Dado que la variable de control (o manipulación) está en forma integral, es necesario establecer los límites de velocidad de los límites superior e inferior:

$$u_K > \omega_{max}\ then\ u_K = \omega_{max} \tag{65}$$

$$u_K < \omega_{min}\ then\ u_K = \omega_{min} \tag{66}$$

La figura 17 muestra la señal de posición actual usando un controlador P, con K_C= 1, T_i= 0.7 y una señal de referencia de 25 (punto de ajuste de posición). Se observan los cambios de velocidad determinados por el controlador. La velocidad máxima es ω_{MAX}=10 rad/s para una dirección y $-\omega_{MAX}$= -10 rad/s para la otra dirección. Existe una banda muerta de tal manera que la velocidad mínima es ω_{MIN}= 0.1 rad/s para una dirección, y $-\omega_{MIN}$= -0.1 rad/s para la otra dirección. La señal de transición de velocidad cae gradualmente, alcanzando la banda de saturación. La velocidad está representada en el eje secundario del gráfico.

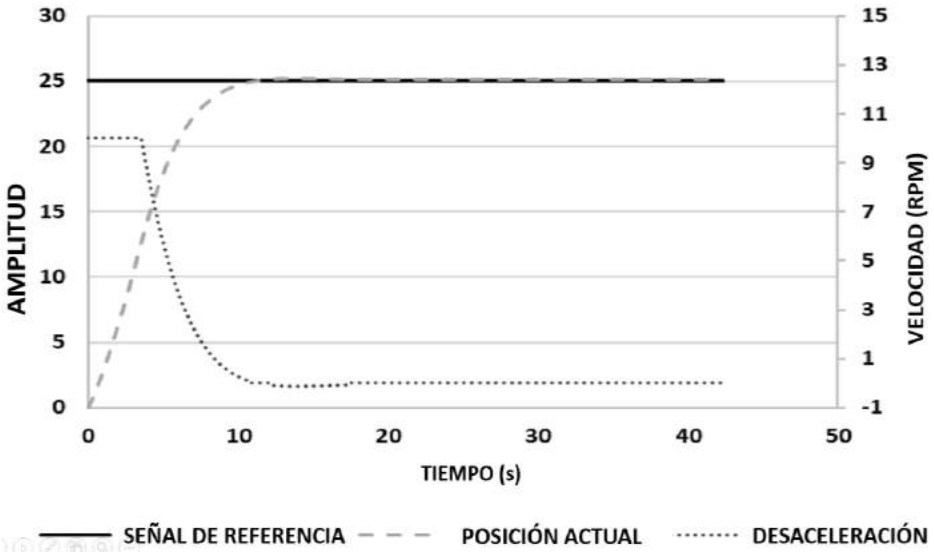


Figura 17. Control de posición P-PID.

Fuente: Elaboración propia.

4.2 Estrategia de control FSM-IMC-PID en cascada propuesta

La máquina de estados finitos o autómata finito se define matemáticamente con el quíntuplo: $G = (X, \Sigma, \delta, x_0, X_m)$, donde X es el conjunto de estados, Σ es el conjunto de símbolos de los eventos, δ representa los eventos o funciones lógicas para las transiciones entre estados, x_0 es el estado inicial y X_m es un subconjunto de estados. Las funciones de transición son condiciones lógicas para el cambio de estado (eventos de entrada o condiciones para detectar que las acciones de la etapa actual se completaron y el sistema puede pasar al siguiente estado). En el estado actual, se realizan acciones (se activan los comandos de salida). FSM describe el

comportamiento del sistema como una máquina con un número finito de estados, es decir, a través de estados, eventos y acciones. FSM es un modelo de computación que puede reaccionar a las entradas y el estado del sistema, para realizar transacciones de un estado a otro y generar una salida en un período de tiempo [48,49].

4.3 Control cascada FSM-IMC-PID para el control de posición y velocidad

La estrategia de control en la que se utiliza un FSM como controlador de posición se muestra en la figura 18. En el diagrama de bloques, se representan el modelo del motor de CD y los lazos de control interior y exterior. La señal de error de posición e_1 entra en el bloque del controlador FSM. La salida del controlador FSM genera una referencia de velocidad ω_{SP}. El error de velocidad e_2 se calcula y se envía como entrada al controlador PID. La salida del controlador PID es el voltaje del inducido del motor de CD. El proceso del motor de CD responde a la entrada de voltaje con una velocidad ω, que se integra para producir el desplazamiento θ del eje del motor de CD.

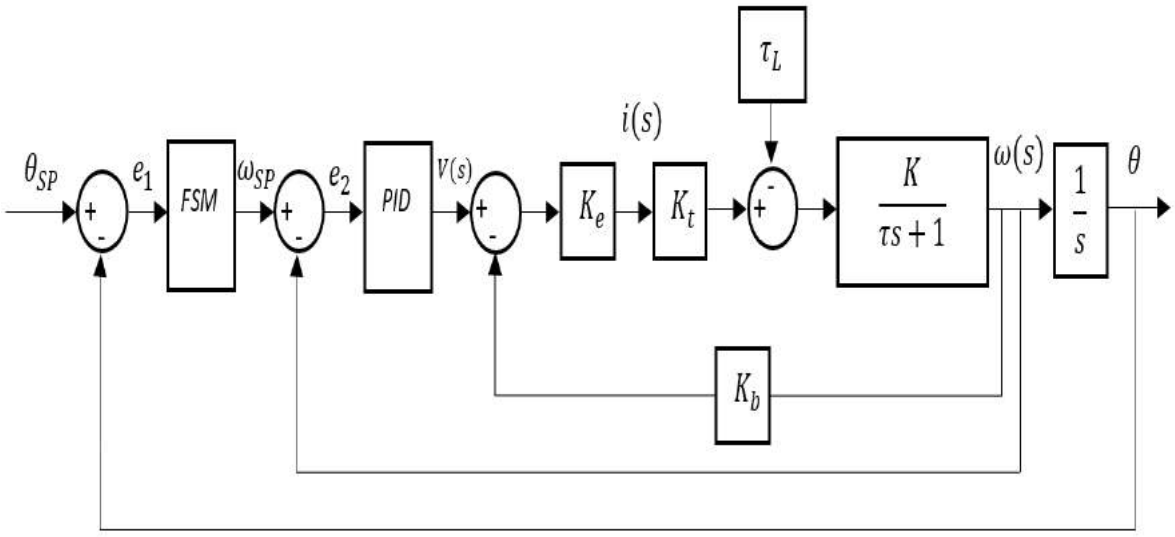


Figura 18. Esquema casca de control de posición (FSM) y velocidad (PID).

Fuente: Elaboración propia.

Para el control de posición del motor de CD, el autómata se define con los siguientes elementos: el estado inicial $x_0 = 0$ corresponde a la salida de velocidad cero: $X_m = \{x_0\} = \{0\}$, es decir, el estado inicial para asegurar que cuando el controlador de posición está activado, el sistema puede iniciar una secuencia de posicionamiento. Los estados se definen según la secuencia de control como $X = \{x_0, x_1, x_2, x_3, x_4, x_5\} = \{0, 1, 2, 3, 4,5\}$, los eventos son $\Sigma = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\}$, y las funciones de transición son $\delta = \{\delta_0,\ \delta_1,\ \delta_2, \delta_3, \delta_4, \delta_5\}$. La figura 19 muestra el diagrama de estados del control de posición del motor de CD, que describe el FSM. El diagrama tiene cinco estados que controlan la posición del motor, donde ω es la velocidad actual, e_1 es el error de posición, D_n es la ventana de desaceleración negativa, D_p es la ventana de desaceleración positiva, S_n es la ventana de parada negativa, S_p es la ventana de parada positiva. El estado inicial es 0 y aplica velocidad angular cero, $\omega = 0$. Si se cambia el punto de referencia de posición se calcula un error, si el error es positivo $e_1 > S_P$, la FSM pasa al estado 1 (ω_{MAX}), que genera una velocidad máxima constante hasta alcanzar una ventana de desaceleración, si el $e_1 < D_P$, la FSM cambia al estado 2 (ω_{MIN}), que aplica una velocidad mínima. Cuando la retroalimentación de posición se acerca a la posición deseada $e_1 < S_P$, el estado 5 se activa y habilita un controlador proporcional $\omega = k_c \times e_1$, y la transición al estado cero ocurre cuando $e_1 < \frac{S_p}{2}$. El estado cero aplica velocidad cero y, por lo tanto, detiene el motor. Si se cambia el punto de ajuste de posición y se calcula un error negativo, $e_1 < S_n$, el FSM pasa al estado 3, que aplica una velocidad negativa máxima constante, $-\omega_{MAX}$, hasta que el error alcanza una ventana de desaceleración, $e_1 > D_n$; luego, el FSM cambia al estado 4 que aplica una velocidad mínima, $-\omega_{MIN}$. Cuando la retroalimentación de posición se acerca a la posición deseada, $e_1 > S_n$, pasa al estado 5. Finalmente, el FSM pasa al estado cero para establecer la velocidad cero si $e_1 > \frac{S_n}{2}$.

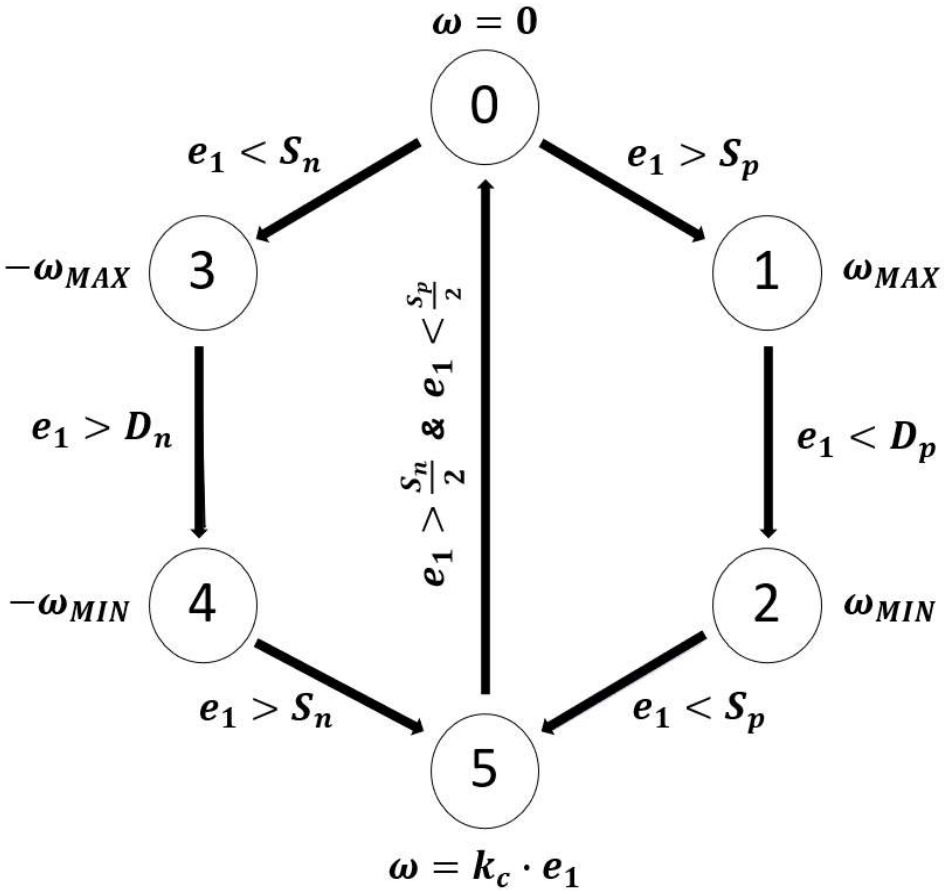


Figura 19. Diagrama de estados del controlador de posición.

Fuente: Elaboración propia.

4.4 Respuesta del controlador FSM-IMC-PID

La figura 20 muestra el uso de un controlador FSM para un cambio de posición controlado el eje del motor DC de una referencia de posición angular de 0 a 25, con los límites de velocidad antes mencionados para la salida del controlador de posición. En esta prueba, la referencia de velocidad máxima es mantenida por el FSM por más tiempo que el controlador de posición P, lo que permite una respuesta más rápida, donde el tiempo para alcanzar el valor de referencia es de 15 s cuando se alcanza $\boldsymbol{\theta}$ =25 y $\boldsymbol{\omega}$ =0. Luego, el FSM reduce abruptamente su salida y procede con cambios más pequeños en la velocidad deseada para alcanzar la posición deseada sin sobrepasarse. Los valores de velocidad se leen en el eje secundario

del gráfico. La implementación del FSM requiere una etapa adicional para activar el proporcional en una pequeña ventana de error para la desaceleración final.

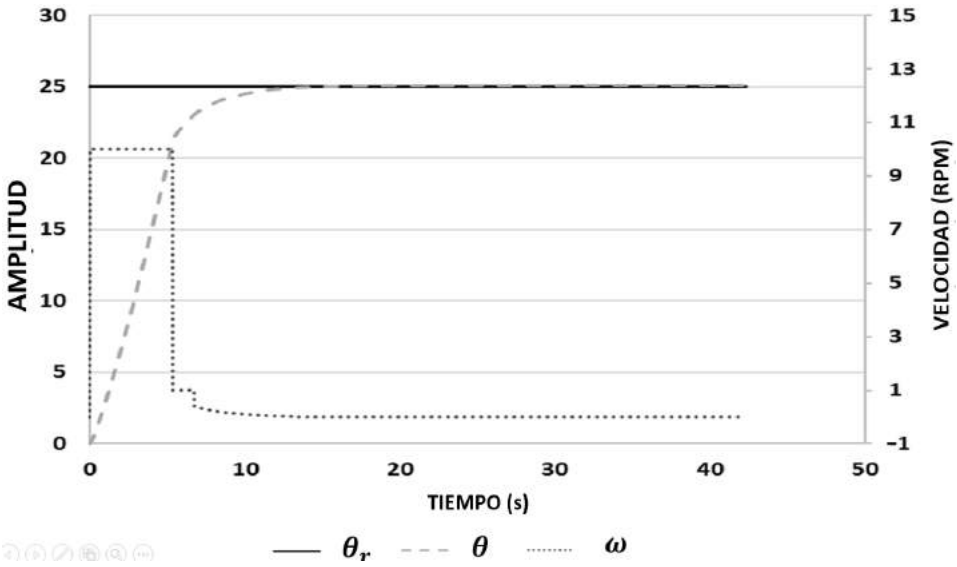


Figura 20. Control de posición FSM-IMC-PID.

Fuente: Elaboración propia.

4.5 Comparación de controlador FSM-IMC-PID

Los controladores de posición proporcional y FSM se evaluaron utilizando el controlador de velocidad IMC-PID. Se seleccionaron y aplicaron referencias de velocidad máxima a ambos controladores, y el control de posición de lazo cerrado se compara a continuación. En la figura 21 se muestra la respuesta de la variable posición utilizando los dos esquemas de control. La respuesta del controlador FSM fue no lineal; se acercó más rápidamente a la señal de referencia θ_r= 25 y alcanzó el valor deseado en 9.83 s. En comparación, el uso del controlador proporcional de posición resolvió la respuesta en 11.33 s. La estabilidad del controlador FSM está garantizada por las transiciones del controlador de un estado a otro.

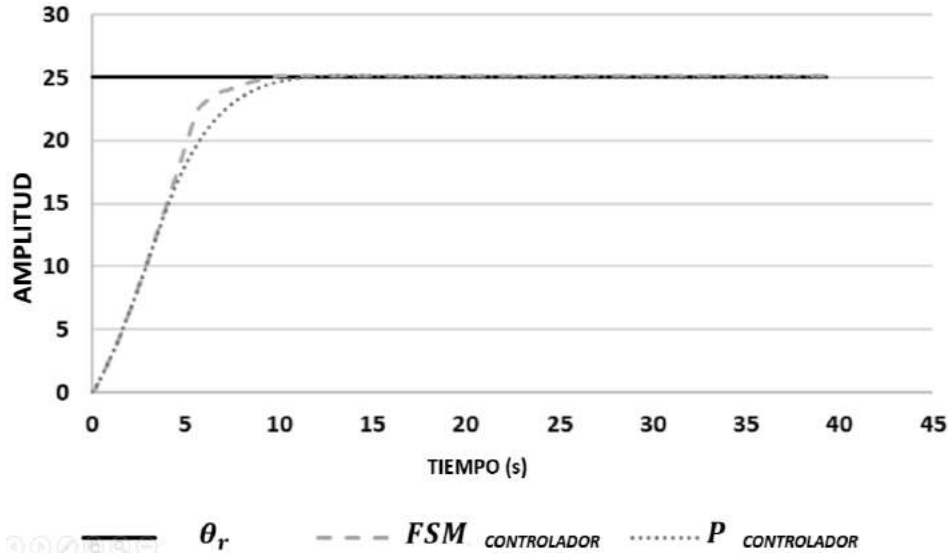


Figura 21. Comparación del control FSM y control P para la posición.

Fuente: Elaboración propia.

Las manipulaciones de velocidad que generan los controladores de posición para el controlador de velocidad IMC-PID interno se comparan en la figura 22. El punto de ajuste de velocidad que generó el controlador de posición proporcional fue el esperado, inicialmente limitado por la velocidad máxima y luego calculado para ser directamente proporcional al error de posición. El controlador FSM generó dos referencias de velocidad constante: la velocidad máxima y luego la velocidad de desaceleración constante. Finalmente se alcanzó una referencia de velocidad cero en el tiempo 10.46 s con el FSM y en el tiempo 10.8 s con el controlador P. La aproximación final se produjo con una velocidad directamente proporcional al pequeño error, que podría incorporarse con un estado de aproximación adicional dentro de la FSM.

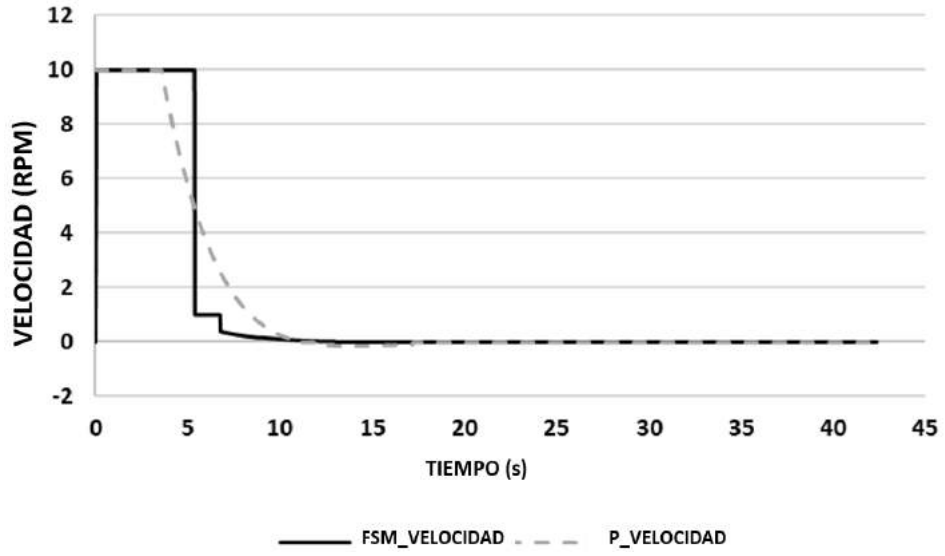


Figura 22. Manipulación de velocidad de los controladores de posición.

Fuente: Elaboración propia.

La figura 23 muestra la comparación de los errores de posición generados durante el uso de los controladores de posición FSM y P. Las curvas de error son similares excepto por la fase de transición final. El FSM conduce a errores menores por una aproximación acelerada al valor de consigna, y una desaceleración oportuna para alcanzar el valor final deseado con precisión, es decir, sin oscilación. La integral de tiempo del error de posición absoluta durante la prueba de simulación resulta un 6% menor para el caso del controlador FSM con respecto al uso del controlador P.

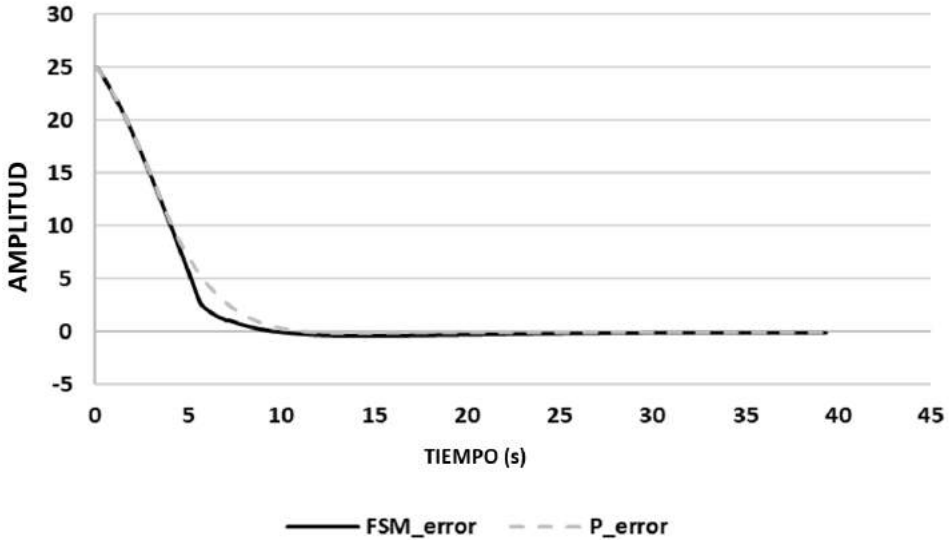


Figura 23. Comparación de los controladores FSM-IMC-PID y P-IMC-PID.

Fuente: Elaboración propia.

Se consideraron los siguientes índices de rendimiento: error cuadrático integral (ISE), error absoluto integral (IAE) y error absoluto integral multiplicado por el tiempo (ITAE). Estos fueron evaluados para los casos del controlador FSM y del controlador proporcional durante un tiempo de simulación de 20 s cubriendo la respuesta transitoria y asegurando la estabilización final del sistema. Los índices de desempeño se muestran en la Tabla 6, así como la mejora porcentual utilizando FSM con respecto a un P para el control de posición.

Tabla 6 Índice de rendimiento del control FSM y control proporcional.

ÍNDICE DE DESEMPEÑO	***FSM***	***P***	%
ISE	74,466.68	78,257.19	4.84
IAE	4498.84	4814.55	6.56
ITAE	11,561.20	13,548.74	14.67

CAPÍTULO 5 DESARROLLO DEL ROBOT VIRTUAL UR5

El robot virtual UR5 desarrollado en Blender se implementó en una estación de trabajo con conectividad Modbus a un SoftPLC externo corriendo en la misma estación de trabajo a la que se puede acceder de forma remota. El robot virtual debe controlarse en un esquema de simulación HiL mediante la programación del controlador CODESYS.

El esquema de la figura 24 muestra algunas posibles configuraciones de control que se pueden implementar. El primer selector de la izquierda es para habilitar el control de cinemática inversa para el posicionamiento cartesiano del robot virtual para la operación directa de los ángulos de articulación. El segundo selector es para habilitar el control de posición con el control de posición basado en FSM propuesto, aquí que requiere una respuesta dinámica servocontrolada en los actuadores del robot.

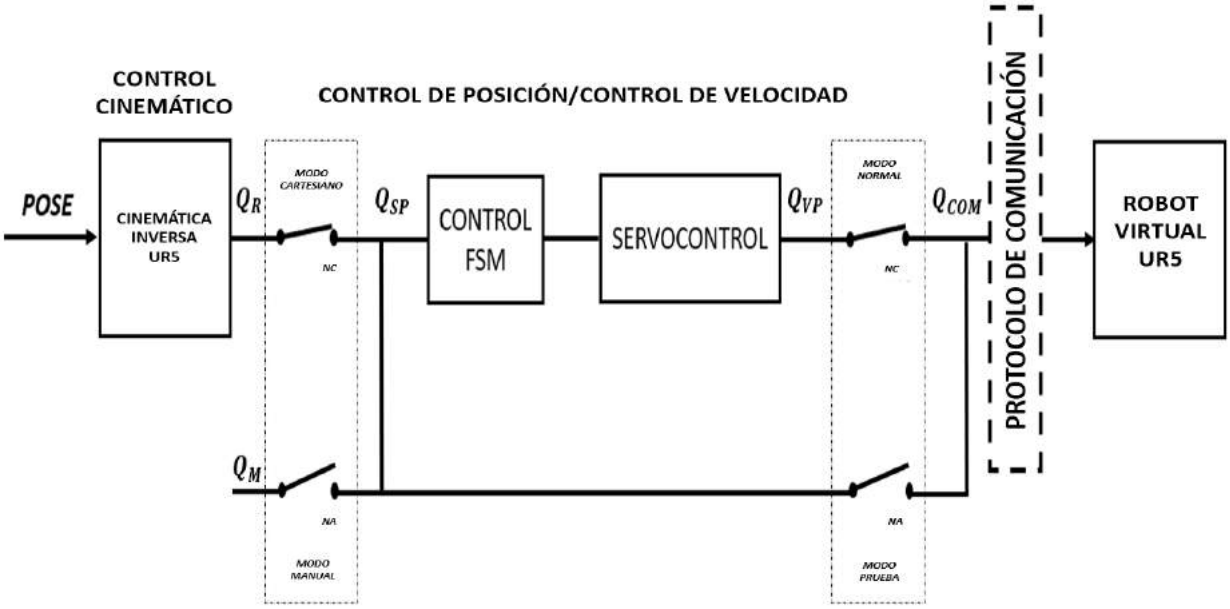


Figura 24. Esquema de configuración del modo cartesiano y modo prueba.

Fuente: Elaboración propia.

Los interruptores selectores pueden activar diferentes configuraciones para el modo cartesiano, modo manual, modo normal y modo de prueba. Las variables vectoriales

se utilizan para referirse a todos los motores conjuntos. Q_R es el valor requerido, Q_{SP} es el valor de referencia, Q_M es el valor manual, Q_{VP} es el valor de la variable de proceso y Q_{COM} es el valor de comunicación. Si el sistema está activo en modo cartesiano y normal, realiza la cinemática inversa para calcular las posiciones angulares requeridas de todas las articulaciones, copiando las posiciones requeridas a los valores de referencia de control de posición y velocidad, el modo normal se utiliza para validar la cinemática inversa. Si se cambia la configuración del sistema a modo manual y prueba, el modo manual está implícito para dar una referencia directa a cada articulación; los valores manuales de cada ángulo se copian a los valores de referencia del control de posición y velocidad, mientras que el modo de prueba ejecuta el control de todas las articulaciones y luego copia el valor de las variables de proceso a las variables de comunicación.

5.1 Modelación 3D del Robot UR5 en Blender

Para realizar el modelo 3D del robot UR5 se utilizó el software Blender como plataforma de animación y simulación del proyecto debido a que es de libre acceso y su entorno de trabajo permite la comunicación en red industrial.

En este apartado se muestra el robot virtual desarrollado en el software Blender, el cual parte con el diseño del Robot UR5 realizado en el software SOLIDWORKS, una vez realizado el modelo, los archivos se guardan en formato .STL y se exportan posteriormente al software Blender, véase figura 25.

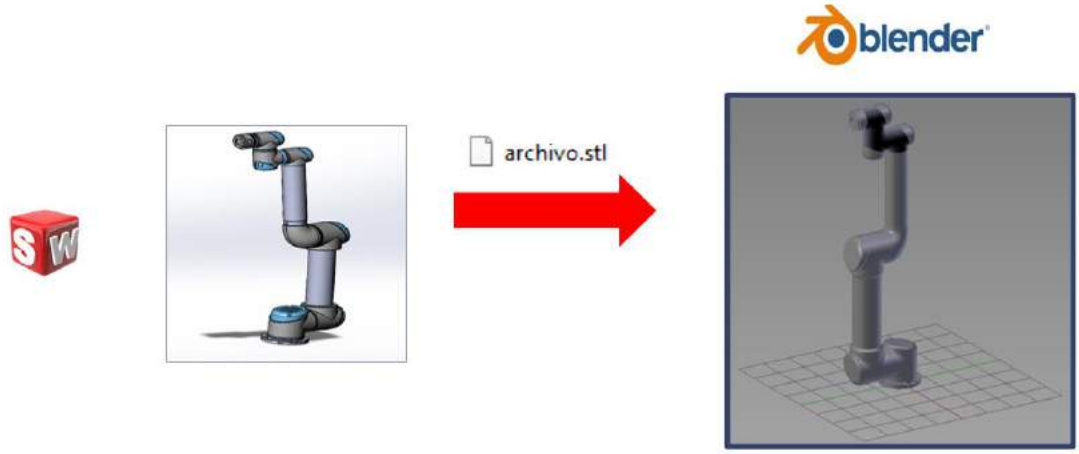


Figura 25. Esquema de importación de archivos.

Fuente: Elaboración propia.

5.1.1 Componentes principales del robot UR5

Las partes principales que conforman el robot virtual UR5 están constituidas por las siguientes articulaciones; articulación base, articulación hombro, articulación codo, articulación de la muñeca, como se muestra en la figura 26.

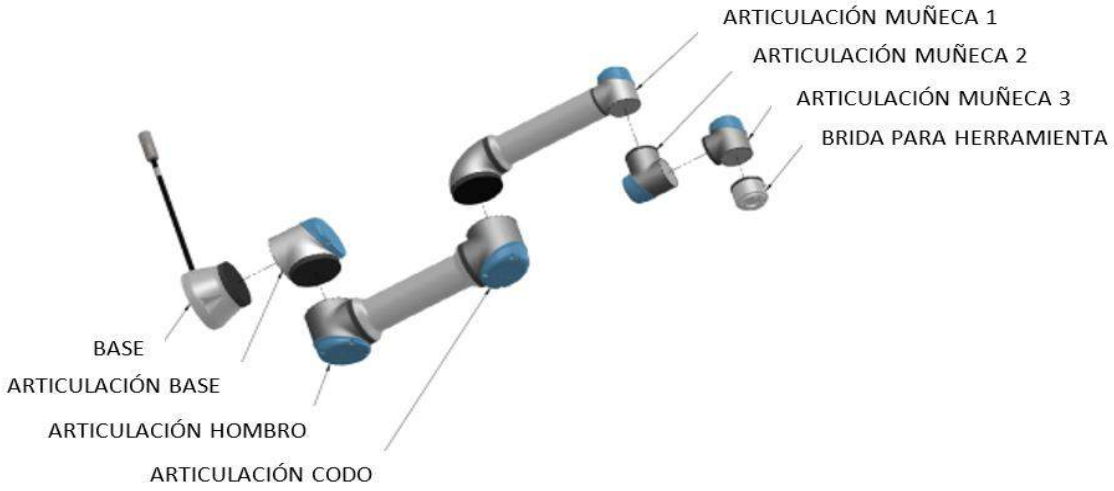


Figura 26. Partes del robot UR5.

Fuente: Universal Robots [50].

A continuación, se muestra los componentes que constituyen el robot UR5 de manera 3D en el software Blender, véase la figura 27.

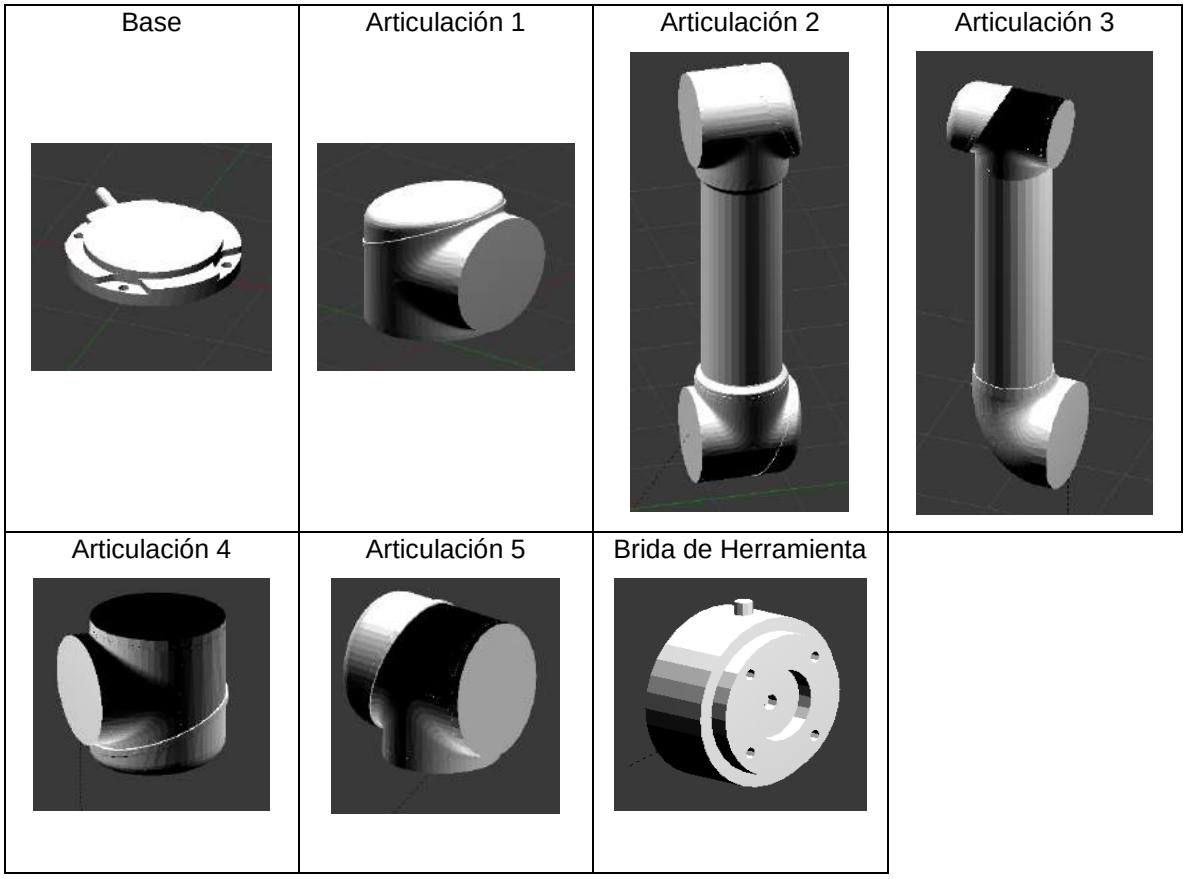


Figura 27. Base, articulaciones, brida de herramienta del robot UR5.

Fuente: Elaboración propia.

Una vez realizado el modelo del robot UR5 dentro del software Blender, se comienza a configurar los elementos del robot 3D, la importación de los archivos coloca los componentes de una forma no organizada, véase figura 28, en la cual se tiene que acomodar en la posición inicial del robot UR5 con las partes que conforman el modelo.

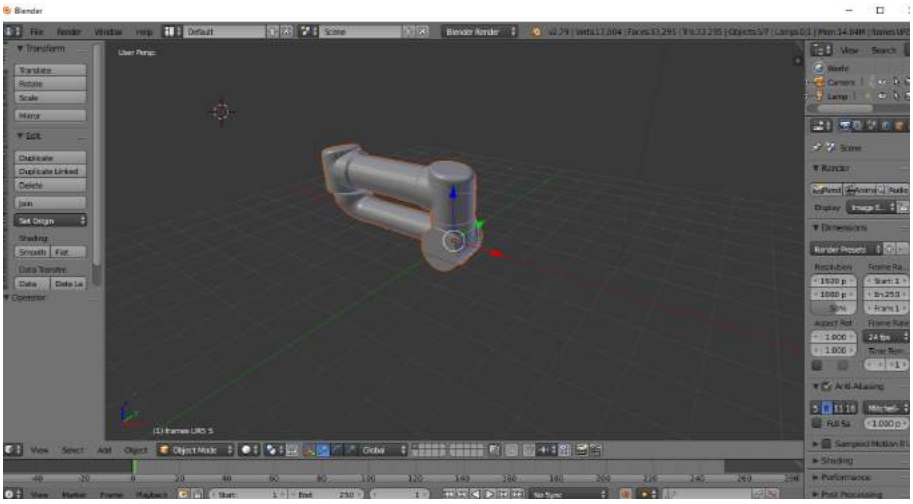


Figura 28. Posicionamiento aparente de los componentes del robot.

Fuente: Elaboración propia.

El ensamble de las partes que conforman el robot virtual UR5 es importante para las configuraciones que se realizan en cada uno de los eslabones, es por ello que las relaciones de posición que se generan en cada componente influyen en el movimiento rotatorio que realice cada articulación, a continuación, se describe la unión de cada componente;

- Base. Se empieza con la configuración para posicionar la base del robot, de tal manera que quede situado en la posición de origen, véase figura 29.

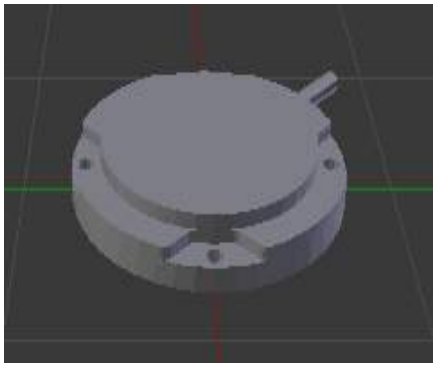


Figura 29. Base del robot.

Fuente: Elaboración propia.

- Articulación 1. Después se configura el eslabón 1, de tal manera que quede situado en la parte superior de la base que se colocó anteriormente, véase figura 30.

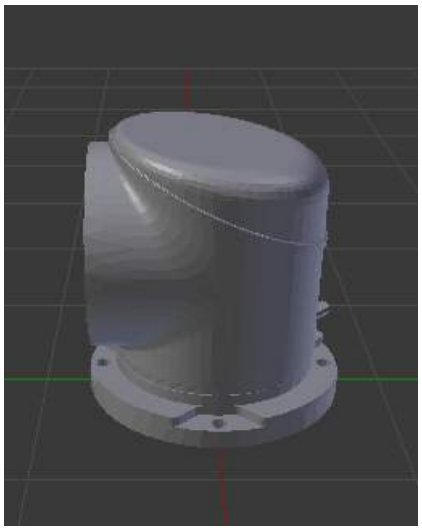


Figura 30. Ensamble de base con eslabón 1.

Fuente: Elaboración propia.

- Articulación 2. Seguidamente se configura eslabón 2, de tal manera que queda unido en la parte lateral del eslabón 1 que se colocó anteriormente, véase figura 31.

Figura 31. Ensamble eslabón 1 con eslabón 2.

Fuente: Elaboración propia.

- Articulación 3. Posteriormente se configura el eslabón 3 del robot, de tal manera que queda unido con el eslabón 2 que se colocó anteriormente, véase figura 32.

Figura 32. Ensamble eslabón 2 con eslabón 3.

Fuente: Elaboración propia.

- Articulación 4. Luego se configura la articulación del eslabón 4 del robot, de tal manera que queda unido con el eslabón 3 que se colocó anteriormente, véase figura 33.

Figura 33. Ensamble eslabón 3 con eslabón 4.

Fuente: Elaboración propia.

- Articulación 5. A continuación, se configura la articulación del eslabón 5 del robot, de tal manera que queda unido con el eslabón 4 que se colocó anteriormente, véase figura 34.

Figura 34. Ensamble eslabón 4 con eslabón 5.

Fuente: Elaboración propia.

- Articulación 6. Finalmente se configura la articulación de la brida para el efector del robot, de tal manera que queda unido con el eslabón 5 que se colocó anteriormente, véase figura 35.

Figura 35. Ensamble eslabón 5 con brida para herramienta.

Fuente: Elaboración propia.

5.1.2 Emparentado objetos

Al modelar un objeto complejo, todos los objetos deben de estar enlazados unos a otros. Se tiene que asignar como padre (superior) de todos los hijos (inferior) a los objetos que se va dar animación. El desplazamiento que realice la función de padre afectara el movimiento que realice la función de hijo.

Estas características son indispensables en los eslabones del robot en 3D para obtener la relación de padre e hijo entre las articulaciones y generar el movimiento autónomo de cada objeto en relación con los demás objetos, a esto se le denomina grados de libertad (GDL) y es el número de grados de libertad que conforma el robot UR5. Para emparentar objetos del modelo virtual UR5 dentro de Blender, se elige al menos dos objetos, primero objeto hijo y después el objeto padre, posteriormente se presiona las teclas ctrl-p para generar la relación. En la figura 36 se puede observar el Robot UR5 conformado por todas sus componentes, pero las articulaciones que lo integran no están vinculadas unas con otras, por ese motivo es necesario aplicar la relación padre-hijo.

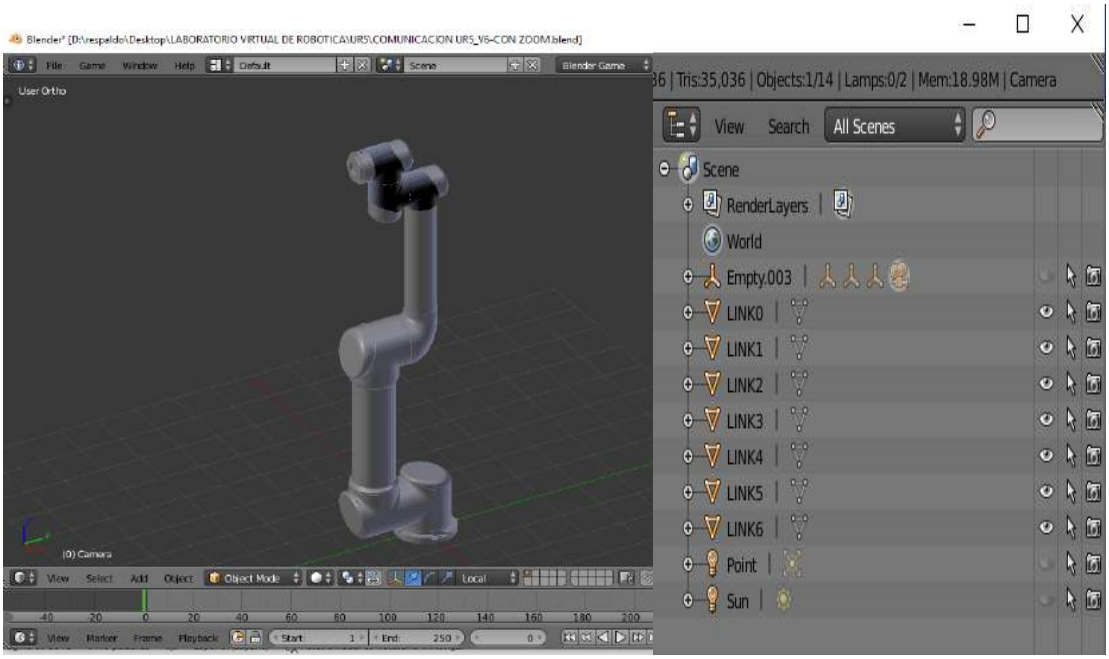


Figura 36. Unión de componentes Robot UR5.

Fuente: Elaboración propia.

Para emparentar los objetos que constituyen el proyecto, en la figura 37 se puede ver la relación del eslabón 2 como hijo y el eslabón 1 como padre.

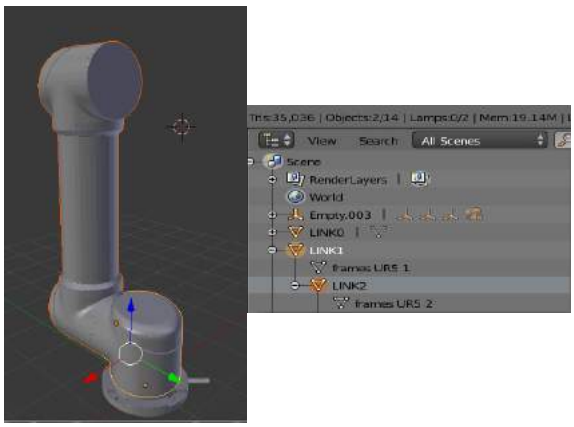


Figura 37. Emparentar eslabón 2 con eslabón 1.

Fuente: Elaboración propia.

En la figura 38 se puede ver la relación del eslabón 3 como hijo y el eslabón 2 como padre.

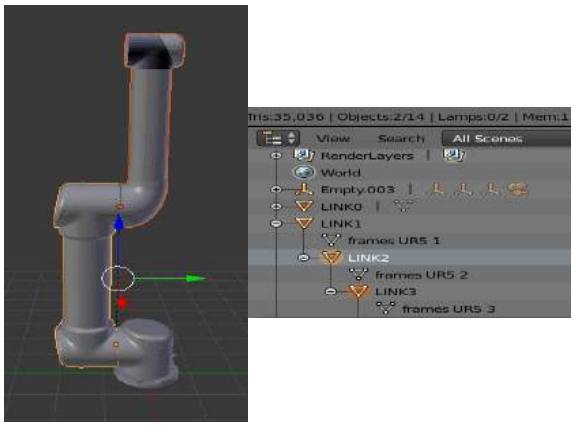


Figura 38. Emparentar eslabón 3 con eslabón 2.

Fuente: Elaboración propia.

En la figura 39 se puede ver la relación del eslabón 4 como hijo y el eslabón 3 como padre.

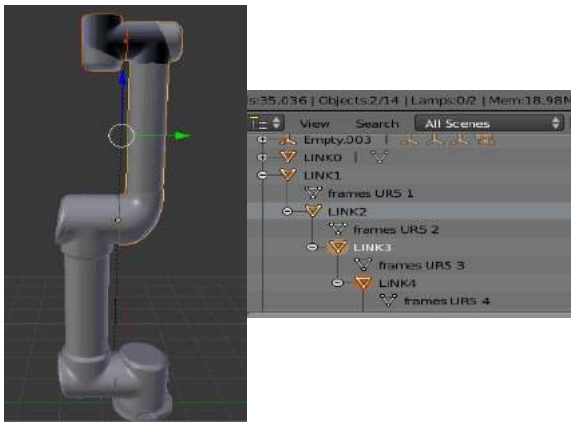


Figura 39. Emparentar eslabón 4 con eslabón 3.

Fuente: Elaboración propia.

La figura 40 se puede ver la relación del eslabón 5 como hijo y el eslabón 4 como padre.

Figura 40. Emparentar eslabón 5 con eslabón 4.

Fuente: Elaboración propia.

Finalmente se puede ver la relación del eslabón 6 como hijo y el eslabón 5 como padre, véase figura 41, siendo el eslabón 1 el principal objeto que permite el movimiento y rotación de la cadena cinemática del robot UR5.

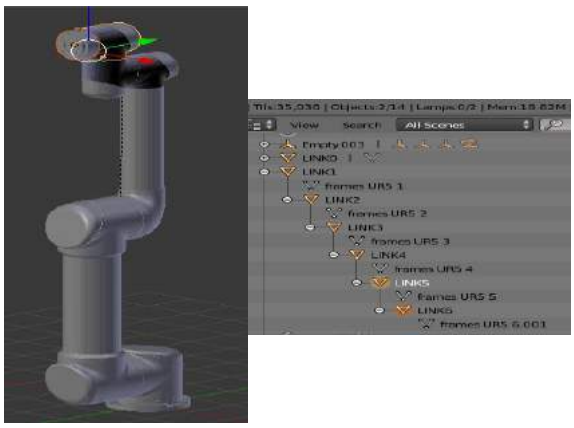


Figura 41. Emparentar eslabón 6 con eslabón 5.

Fuente: Elaboración propia.

5.1.3 Asignación de punto de origen en articulaciones.

La asignación de puntos de origen de un objeto en Blender es el punto en el espacio que representa a todo el objeto, estableciendo la posición en la cual se localiza el objeto. El origen de un objeto es importante porque define la posición del mismo en su entorno de trabajo y los cambios que se realicen en estas funciones en general se evalúen conforme al punto de origen.

La asignación del punto de origen de las articulaciones comienza estando en modo edición, posicionando el cursor en el punto centro de la cara de una articulación y después seleccionarla. Cuando la cara esta seleccionada se utiliza las teclas shift+s para abrir el menú de opciones, en este momento se selecciona mover el cursor 3D a la selección actual permitiendo que el cursor 3D se traslade al centro geométrico de la cara de la articulación. Después se mueve el origen del objeto a la posición del cursor 3D, entrando al modo objeto se da clic derecho al objeto y se selecciona la opción set origin, se elige la opción origen del cursor 3D haciendo que el origen del objeto se mueva a la posición actual.

En la figura 42, el punto de origen se posiciona en el centro de la articulación 1.

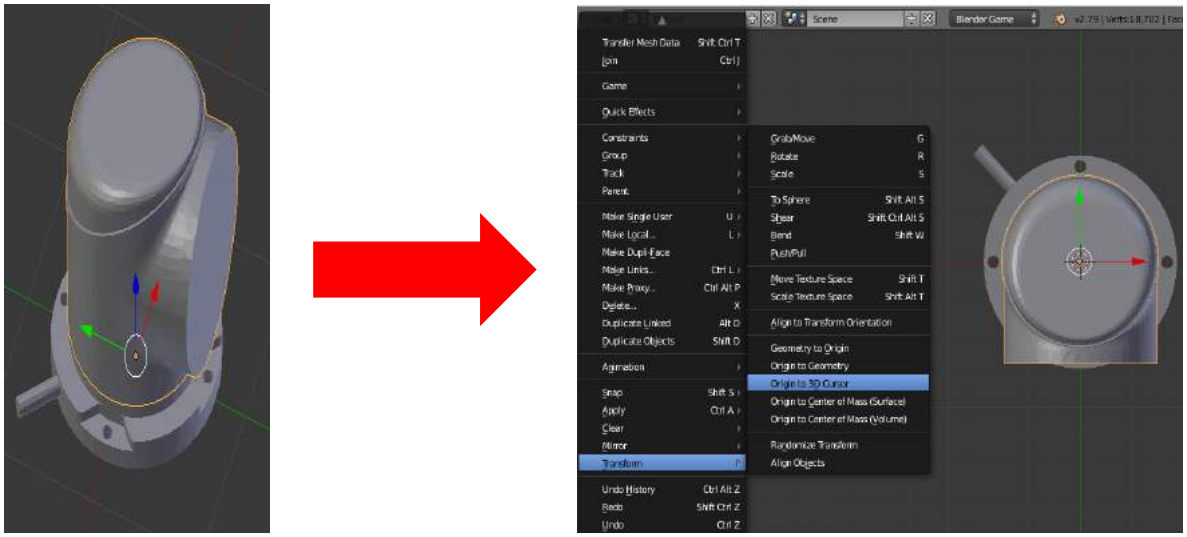


Figura 42. Punto de origen de la articulación 1.

Fuente: Elaboración propia.

En la figura 43 se puede observar el punto de origen que se posiciona en el eje de rotación de la articulación 2.

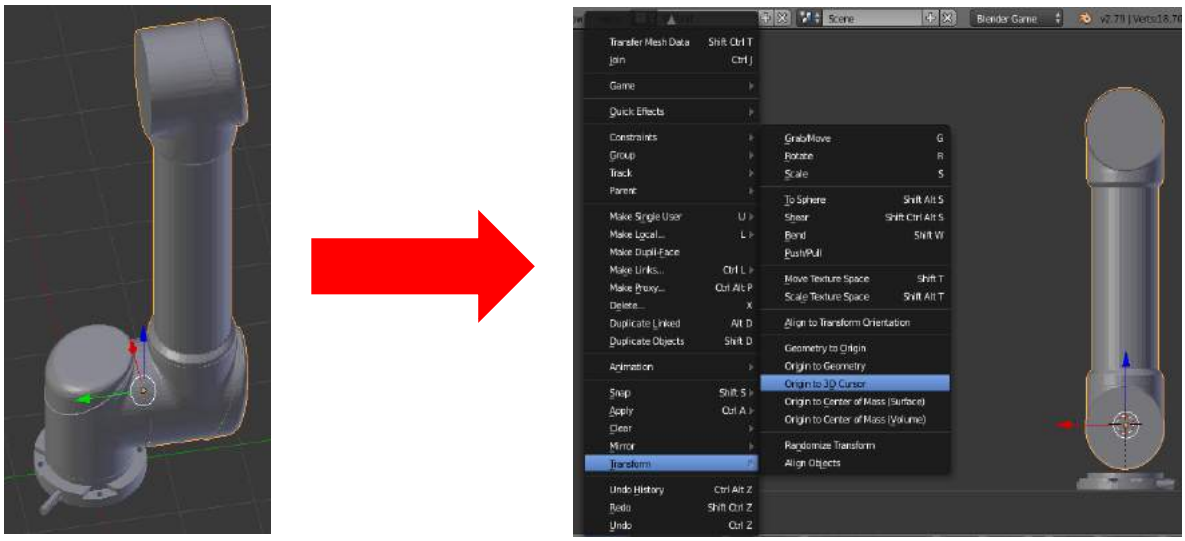


Figura 43. Punto de origen de la articulación 2.

Fuente: Elaboración propia.

En la figura 44 se puede ver el punto de origen que se posiciona en el eje de rotación de la articulación 3.

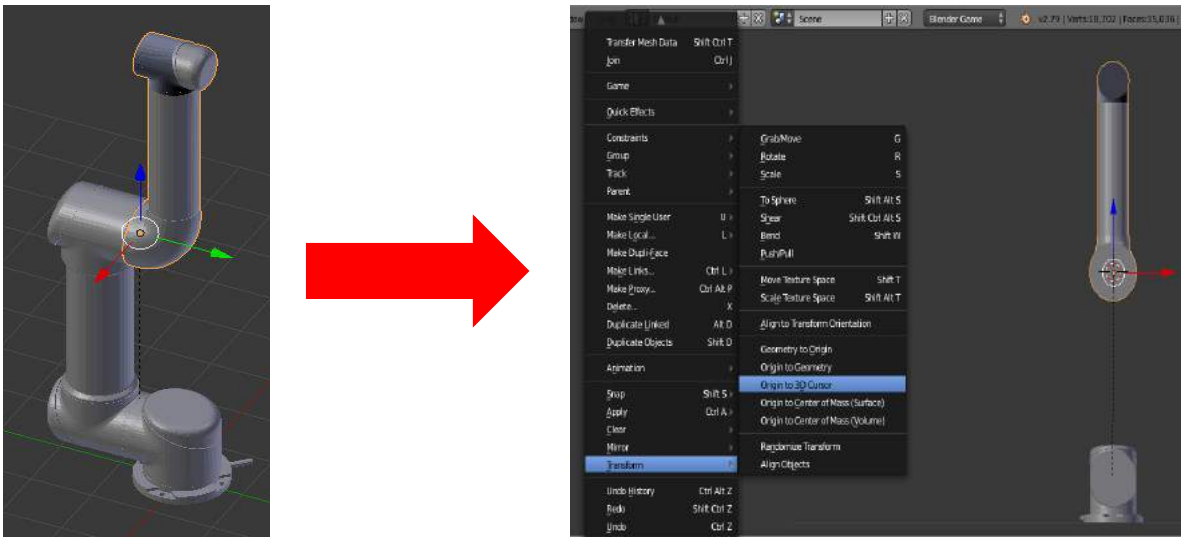


Figura 44. Punto de origen de la articulación 3.

Fuente: Elaboración propia.

El punto de origen que se posiciona en el eje de rotación de la articulación 4 se puede ver en la figura 45.

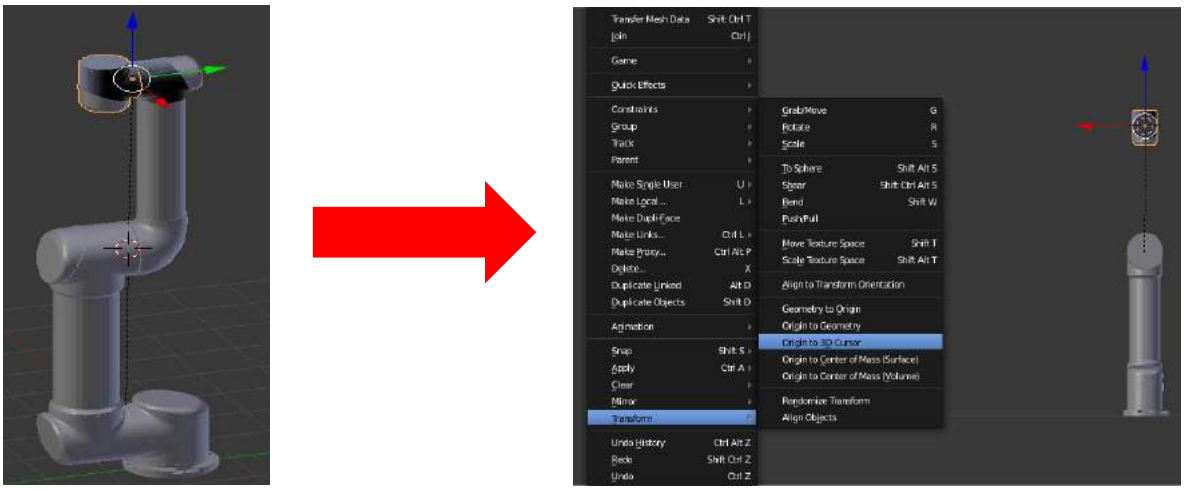


Figura 45. Punto de origen de la articulación 4.

Fuente: Elaboración propia.

En la figura 46 se puede ver el punto de origen que se posiciona en el eje de rotación de la articulación 5.

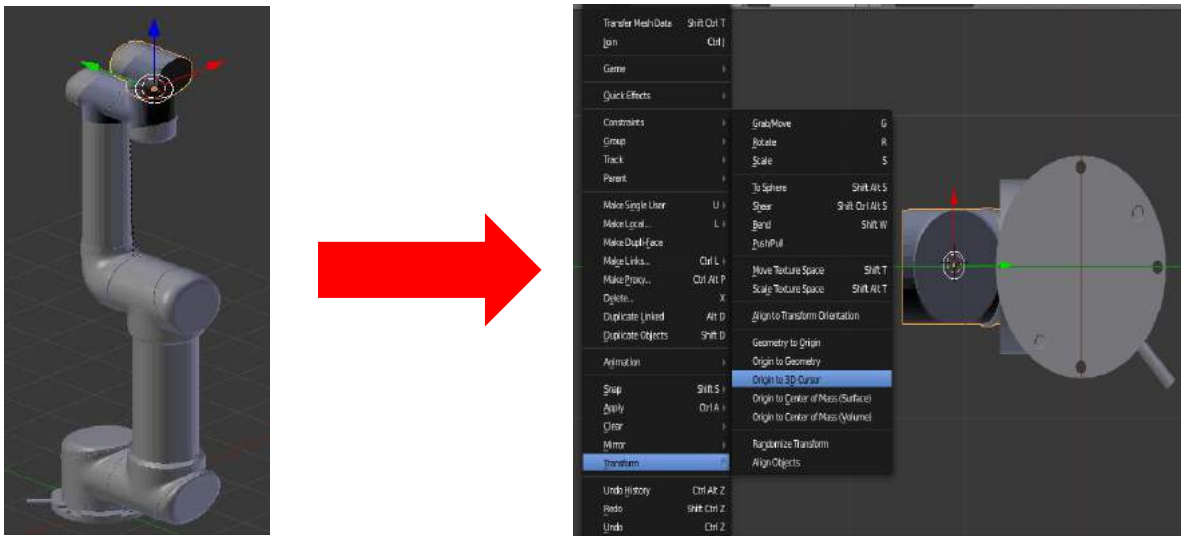


Figura 46. Punto de origen de la articulación 5.

Fuente: Elaboración propia.

En la figura 47 se puede ver el punto de origen que se posiciona en el eje de rotación de la articulación 6 (brida de herramienta).

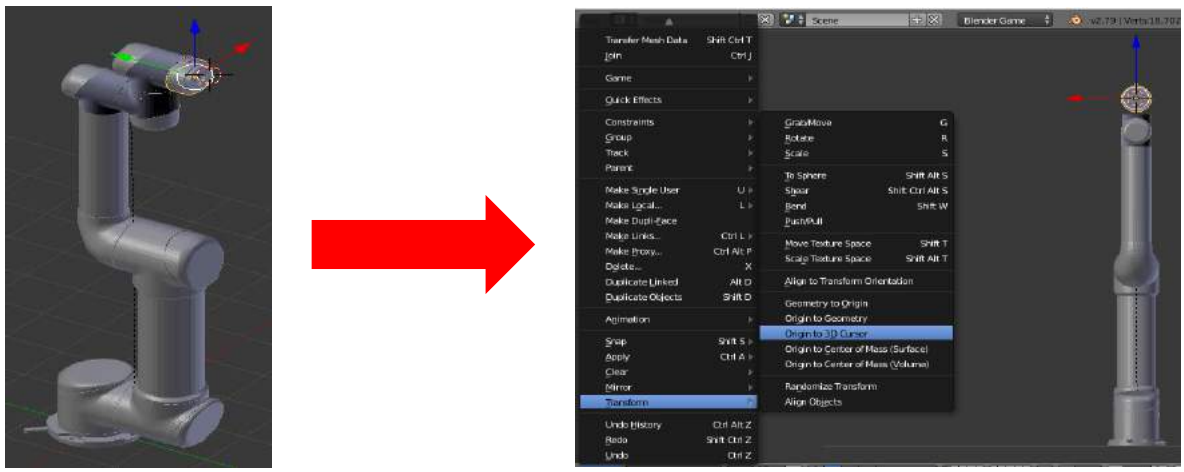


Figura 47. Punto de origen de la brida de herramienta.

Fuente: Elaboración propia.

Los cambios que se realicen en el modo objeto se modifican en relación al punto asignado. El punto de origen de la pieza es el vector que se representa con las coordenadas X, Y, Z, el cual se muestra al seleccionar la pieza, una vez ubicado el punto de origen de inicio, este punto de origen se cambia de posición para cada eje que conforma el robot UR5, con base a los ejes de rotación de cada eslabón.

Se puede editar el origen de un objeto 3D y que su posición sea la adecuada para los objetos, si se trata del Robot UR5 el origen del objeto debe estar en el eje de rotación de cada articulación de tal manera que al rotar la pieza el movimiento tendrá una rotación natural.

5.2 Animación del robot virtual UR5

En la siguiente sección se muestra la configuración de la animación del robot UR5 en entornos de realidad virtual mediante el uso del software Blender. El movimiento del robot es de 6 grados de libertad además se muestran los procesos necesarios para controlar el movimiento rotacional y visualización en un entorno de realidad virtual.

5.2.1 Configuración blender game engine

Blender Game Engine (BGE) es una herramienta de Blender para desarrollar proyectos en tiempo real como simulaciones, juegos y renderizados arquitectónicos. Esta aplicación utiliza un sistema de bloques lógicos gráficos (una combinación de sensores, controladores y actuadores) para controlar el movimiento y la visualización de objetos. La animación de los proyectos se mejora con herramientas de enlaces de Python.

En la figura 48, se muestra la configuración del módulo de BGE para la animación de los objetos del robot UR5 (a), utiliza un sistema de bloques lógicos gráficos; una combinación de sensores (b), controladores (c) y actuadores (d), para controlar el movimiento y la visualización de objetos.

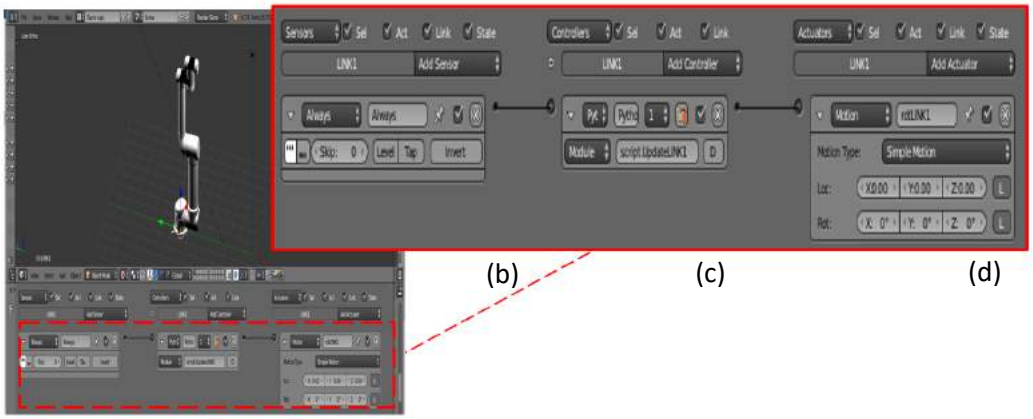


Figura 48. Configuración de bloques lógicos en Blender.

Fuente: Elaboración propia.

Para cada objeto (articulación) que conforma el robot UR5 es necesario configurar el módulo BGE, programando los bloques lógicos, en el bloque de control lleva un código script.UpdateLINK1 y en el bloque del actuador rotLINK1, donde se muestra la estructura para la articulación 1, véase la figura 49.

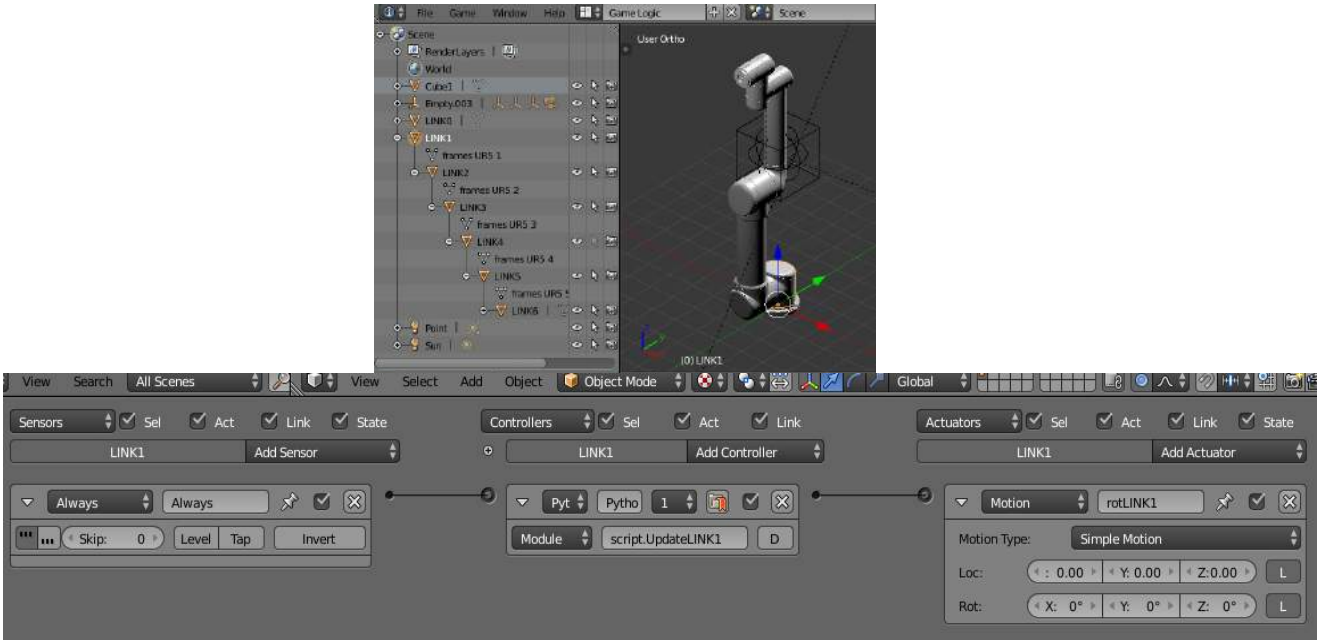


Figura 49. Configuración de bloques lógicos articulación 1.

Fuente: Elaboración propia.

En la figura 50 se puede observar la configuración de los bloques lógicos, en el bloque de control lleva un código script.UpdateLINK2 y en el bloque del actuador rotLINK2, para la articulación 2.

Figura 50. Configuración de bloques lógicos articulación 2.

Fuente: Elaboración propia.

Se puede observar en la figura 51 la configuración de los bloques lógicos para la articulación 3, en donde el bloque de control lleva un código script.UpdateLINK3 y en el bloque del actuador rotLINK3.

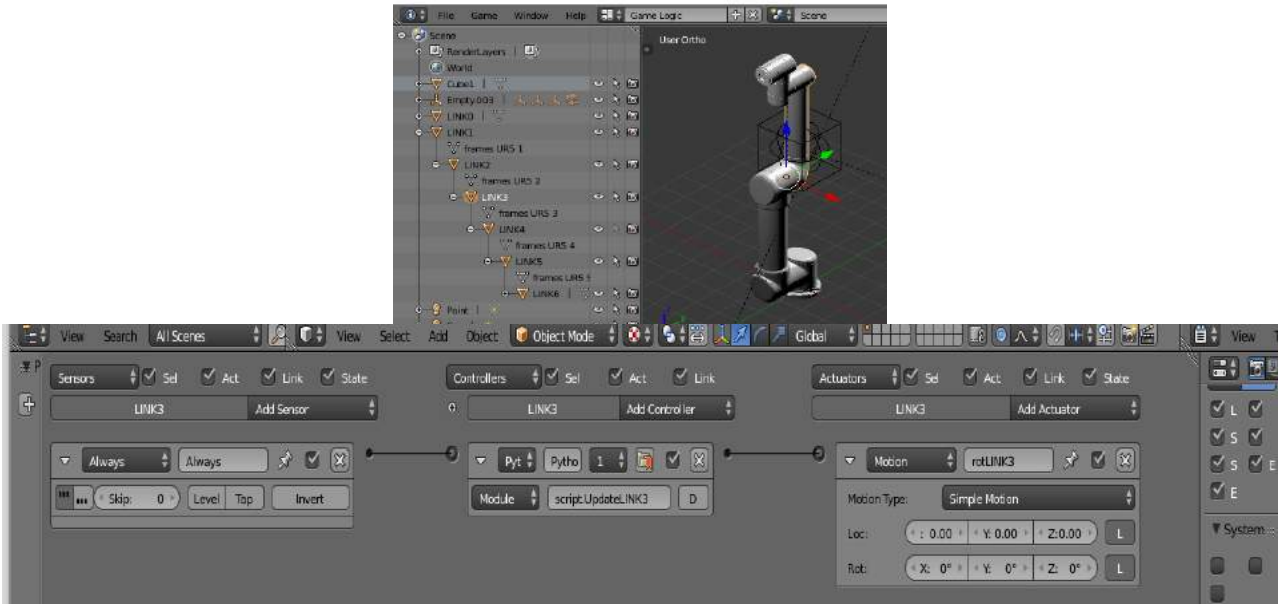


Figura 51. Configuración de bloques lógicos articulación 3.

Fuente: Elaboración propia.

En la figura 52 se puede observar la configuración de los bloques lógicos para la articulación 4, en el bloque de control lleva un código script.UpdateLINK4 y en el bloque del actuador rotLINK4.

Figura 52. Configuración de bloques lógicos articulación 4.

Fuente: Elaboración propia.

Se puede observar en la figura 53 la configuración de los bloques lógicos para la articulación 5, en donde el bloque de control lleva un código script.UpdateLINK5 y en el bloque del actuador rotLINK5.

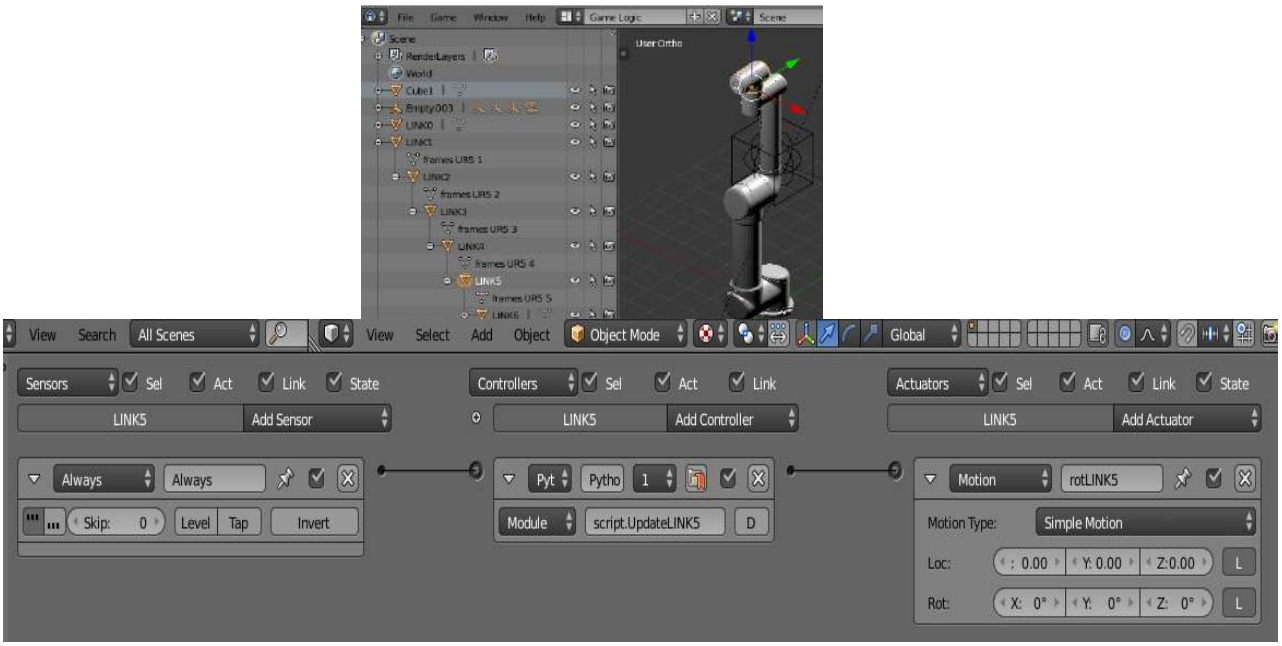


Figura 53. Configuración de bloques lógicos articulación 5.

Fuente: Elaboración propia.

En la figura 54 se puede observar la configuración de los bloques lógicos para la articulación 6 (brida de herramienta), en el bloque de control lleva un código script.UpdateLINK6 y en el bloque del actuador rotLINK6.

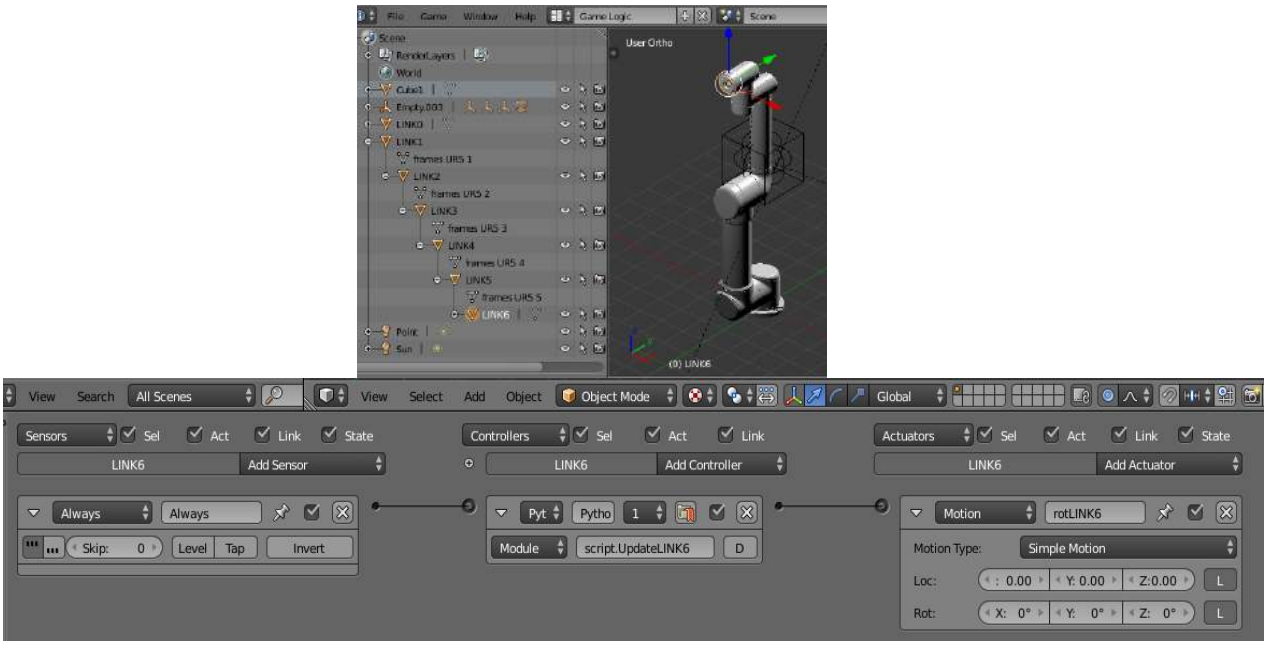


Figura 54. Configuración de bloques lógicos articulación 6.

Fuente: Elaboración propia.

5.2.2 Programación en Blender

Blender cuenta con una consola para programar con un lenguaje de programación en Python, en esta se puede insertar secuencias de comandos que se utiliza para manipular, personalizar y automatizar las instalaciones de un proyecto, y poder ser ejecutado en tiempo real.

En las siguientes imágenes se muestra el código de las secuencias programadas por cada objeto (eslabón) para generar su movimiento rotacional de cada articulación, y posteriormente estar comunicado con un protocolo de comunicación industrial MODBUS, que permitirá la comunicación en entre el software Blender y el software CODESYS.

- En la figura 55 se puede observar el código de programación que se utilizó en el software Blender para comunicar el eje 1 entre el software CODESYS - Blender y poder realizar el movimiento rotacional a través de la comunicación maestro-esclavo.

```
def UpdateLINK1(controller):

    try:
        c = ModbusClient(host="176.16.3.22", port=502)
    except ValueError:
        print("Error with host or port params")
    Q1=0

    if c.open():
        ReadRegs = c.read_input_registers(0, 7)
        c.close()
        timeA=time()
        print(ReadRegs)
        WachtDog=ReadRegs[0]

        Q1=((ReadRegs[1]-10000)/100)*0.15

    rot_L1=float(Q1)/1.0

    co=g.getCurrentController()

    rotLINK1=co.actuators["rotLINK1"]
    co.activate(rotLINK1)

    if rot_L1==0.0:
        KB=g.keyboard.events

        if KB [events.WKEY]:
         rot_L1=1.05

        if KB [events.SKEY]:
         rot_L1=-1.05

    rotLINK1.dRot=[0.0,0.0,rot_L1/100.0]

    return
```

Figura 55. Configuración de script en Blender articulación 1.

Fuente: Elaboración propia.

- En la figura 56 se puede observar el código de programación que se utilizó en el software Blender para comunicar el eje 2 entre el software CODESYS - Blender y poder realizar el movimiento rotacional a través de la comunicación maestro-esclavo.

```
def UpdateLINK2(controller):

    try:
        c = ModbusClient(host="176.16.3.22", port=502)
    except ValueError:
        print("Error with host or port params")
    #Q1=0
    Q2=0
    if c.open():
        ReadRegs = c.read_input_registers(0, 7)
        c.close()
        timeA=time()
        print(ReadRegs)
        WachtDog=ReadRegs[0]

        Q2=((ReadRegs[2]-10000)/100)*0.15

    rot_L2=float(Q2)/1.0

    co=g.getCurrentController()

    rotLINK2=co.actuators["rotLINK2"]
    co.activate(rotLINK2)

    if rot_L2==0.0:
        KB=g.keyboard.events

        if KB [events.EKEY]:
         rot_L2=2.05

        if KB [events.DKEY]:
         rot_L2=-2.05

    rotLINK2.dRot=[0.0,rot_L2/100.0,0.0]

    return
```

Figura 56. Configuración de script en Blender articulación 2.

Fuente: Elaboración propia.

- En la figura 57 se puede observar el código de programación que se utilizó en el software Blender para comunicar el eje 3 entre el software CODESYS - Blender y poder realizar el movimiento rotacional a través de la comunicación maestro-esclavo.

```
def UpdateLINK3(controller):

    try:
        c = ModbusClient(host="176.16.3.22", port=502)
    except ValueError:
        print("Error with host or port params")
    Q3=0
    #Q2=0
    if c.open():
        ReadRegs = c.read_input_registers(0, 7)
        c.close()
        timeA=time()
        print(ReadRegs)
        WachtDog=ReadRegs[0]
        Q3=((ReadRegs[3]-10000)/100)*0.15

    rot_L3=float(Q3)/1.0

    co=g.getCurrentController()

    rotLINK3=co.actuators["rotLINK3"]
    co.activate(rotLINK3)

    if rot_L3==0.0:
        KB=g.keyboard.events

        if KB [events.RKEY]:
         rot_L3=2.05

        if KB [events.FKEY]:
         rot_L3=-2.05

    rotLINK3.dRot=[0.0,rot_L3/100.0,0.0]

    return
```

Figura 57. Configuración de script en Blender articulación 3.

Fuente: Elaboración propia.

- En la figura 58 se puede observar el código de programación que se utilizó en el software Blender para comunicar el eje 4 entre el software CODESYS - Blender y poder realizar el movimiento rotacional a través de la comunicación maestro-esclavo.

```
def UpdateLINK4(controller):

    try:
        c = ModbusClient(host="176.16.3.22", port=502)
    except ValueError:
        print("Error with host or port params")
    Q4=0
    #Q2=0
    if c.open():
        ReadRegs = c.read_input_registers(0, 7)
        c.close()
        timeA=time()
        print(ReadRegs)
        WachtDog=ReadRegs[0]
        Q4=((ReadRegs[4]-10000)/100)*0.15


    rot_L4=float(Q4)/1.0

    co=g.getCurrentController()

    rotLINK4=co.actuators["rotLINK4"]
    co.activate(rotLINK4)

    if rot_L4==0.0:
        KB=g.keyboard.events

        if KB [events.TKEY]:
         rot_L4=2.05

        if KB [events.GKEY]:
         rot_L4=-2.05

    rotLINK4.dRot=[0.0,rot_L4/100.0,0.0]

    return
```

Figura 58. Configuración de script en Blender articulación 4.

Fuente: Elaboración propia.

- En la figura 59 se puede observar el código de programación que se utilizó en el software Blender para comunicar el eje 5 entre el software CODESYS - Blender y poder realizar el movimiento rotacional a través de la comunicación maestro-esclavo.

```
def UpdateLINK5(controller):

    try:
        c = ModbusClient(host="176.16.3.22", port=502)
    except ValueError:
        print("Error with host or port params")
    Q5=0
    if c.open():
        ReadRegs = c.read_input_registers(0, 7)
        c.close()
        timeA=time()
        print(ReadRegs)
        WachtDog=ReadRegs[0]
        Q5=((ReadRegs[5]-10000)/100)*0.15


    rot_L5=float(Q5)/1.0

    co=g.getCurrentController()

    rotLINK5=co.actuators["rotLINK5"]
    co.activate(rotLINK5)

    if rot_L5==0.0:
        KB=g.keyboard.events

        if KB [events.YKEY]:
         rot_L5=0.05

        if KB [events.HKEY]:
         rot_L5=-0.05

    rotLINK5.dRot=[0.0,0.0,rot_L5/100.0]

    return
```

Figura 59. Configuración de script en Blender articulación 5.

Fuente: Elaboración propia.

- En la figura 60 se puede observar el código de programación que se utilizó en el software Blender para comunicar el eje 5 entre el software CODESYS - Blender y poder realizar el movimiento rotacional a través de la comunicación maestro-esclavo.

```
def UpdateLINK6(controller):

    try:
        c = ModbusClient(host="176.16.3.22", port=502)
    except ValueError:
        print("Error with host or port params")
    Q6=0
    if c.open():
        ReadRegs = c.read_input_registers(0, 7)
        c.close()
        timeA=time()
        print(ReadRegs)
        WachtDog=ReadRegs[0]
        Q6=((ReadRegs[6]-10000)/100)*0.15

    rot_L6=float(Q6)/1.0

    co=g.getCurrentController()

    rotLINK6=co.actuators["rotLINK6"]
    co.activate(rotLINK6)

    if rot_L6==0.0:
        KB=g.keyboard.events

        if KB [events.UKEY]:
         rot_L6=3.05

        if KB [events.JKEY]:
         rot_L6=-3.05

    rotLINK6.dRot=[0.0,rot_L6/100.0,0.0]

    return
```

Figura 60. Configuración de script en Blender articulación 6.

Fuente: Elaboración propia.

5.3 Simulación hardware in the loop

Las pruebas hardware in the loop son una técnica en la que las señales reales de un controlador son conectadas a un sistema de pruebas que simula la realidad. En esta sección se muestra el protocolo de comunicación industrial y la programación utilizada.

5.3.1 Protocolo de comunicación industrial MODBUS

El protocolo de comunicación industrial que se utiliza para la comunicación del software CODESYS-Blender es MODBUS, empleado para transmitir información a

través de dispositivos electrónicos. El dispositivo que solicita información se llama maestro y los que suministran la información son los esclavos, en la figura 61 se representa el esquema de comunicación.

Figura 61. Esquema de comunicación.

Fuente: Elaboración Propia.

Su arquitectura puede ser utilizada en cualquier tipo de dispositivos, ya sea un PLC, una pantalla HMI, un panel de control, o cualquier otro tiene la capacidad de usar el protocolo Modbus para iniciar una operación. Puede realizarse la comunicación por serial o por Ethernet. La arquitectura del sistema MAESTRO-ESCLAVO está dada por una dirección IP y un puerto serial, que deben de tener ambos softwares CODESYS-Blender para la comunicación. Para comunicar ambos softwares es necesario configurar su IP y su puerto serial. En la figura 62 se puede observar dentro de CODESYS la configuración para la dirección IP y el valor del puerto serial como se observa en la figura 63.

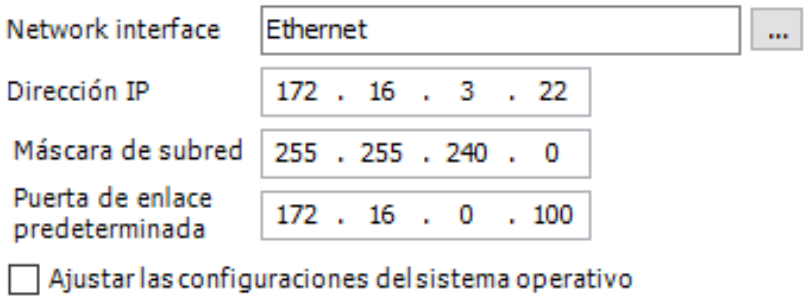


Figura 62. Dirección IP en CODESYS.

Fuente: Elaboración Propia.

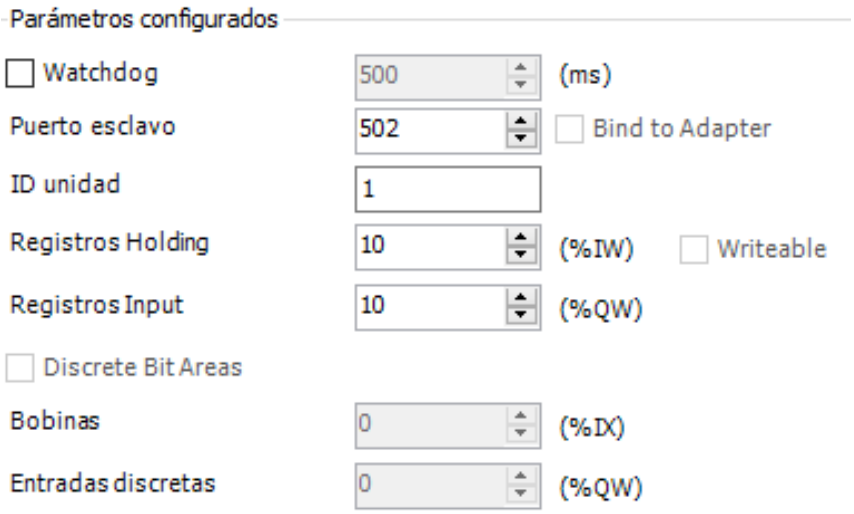


Figura 63. Puerto serial en CODESYS.

Fuente: Elaboración Propia.

En la figura 64 se visualiza la dirección IP que se tiene en Blender además del puerto serial.

```
try:
    c = ModbusClient(host="172.16.3.22", port=502)
except ValueError:
```

Figura 64. Dirección IP y puerto serial en Blender.

Fuente: Elaboración Propia.

5.3.2 Programación en CODESYS

Se generó un proyecto en CODESYS, en el cual incluye, el control de posición a través de la máquina de estados, el servocontrol de velocidad e integración de posición, cálculo para la cinemática inversa y directa, además de un programa como programa principal.

- En la figura 65 se muestra el código de programación del control de posición a través de la máquina de estados para el eje 1 que conforma el robot UR5.

```
CASE STATE1 OF
   0: GVL.W1SP:=0;
      IF ((GVL.Q1SP-GVL.Q1PV)>ErrPos) THEN STATE1:=1;
      END_IF
      IF ((GVL.Q1SP-GVL.Q1PV)<ErrNeg) THEN STATE1:=3;
      END_IF
   1: GVL.W1SP:=VelMaxPos;
      IF ((GVL.Q1SP-GVL.Q1PV)<DesPos) THEN STATE1:=2;
      END_IF
   2: GVL.W1SP:=VelMinPos;
      IF ((GVL.Q1SP-GVL.Q1PV)<ErrPos) THEN STATE1:=0;
      END_IF
   3: GVL.W1SP:=VelMaxNeg;
      IF ((GVL.Q1SP-GVL.Q1PV)>DesNeg) THEN STATE1:=4;
      END_IF
   4: GVL.W1SP:=VelMinNeg;
      IF ((GVL.Q1SP-GVL.Q1PV)>ErrNeg) THEN STATE1:=0;
      END_IF
 END_CASE
```

Figura 65. Programa de control de posición eje 1.

Fuente: Elaboración propia.

- El servocontrol de velocidad e integración de posición, programado para el eje 1, se muestra en la figura 66.

```
ALFA:=TAO/(TAO+TS);

GVL.W1ACT:=ALFA*W1last+(1-ALFA)*GVL.W1SP;  // Dinámica del servocontrol de velocidad
IF GVL.W1ACT>100.0 THEN
  GVL.W1ACT:=100.0;
END_IF
IF GVL.W1ACT<-100.0 THEN
  GVL.W1ACT:=-100.0;
END_IF
W1last:=GVL.W1ACT;

GVL.Q1PV:=Q1last+TS*K1*GVL.W1ACT;        // Integración de la velocidad en posición
IF GVL.Q1PV>180.0 THEN
  GVL.Q1PV:=180.0;
END_IF
IF GVL.Q1PV<-180.0 THEN
  GVL.Q1PV:=-180.0;
END_IF
Q1last:=GVL.Q1PV;
```

Figura 66. Programa control de velocidad eje 1.

Fuente: Elaboración propia.

- En la figura 67 se muestra el código de programación del control de posición a través de la máquina de estados para el eje 2 que conforma el robot UR5.

```
CASE STATE1 OF
   0: GVL.W2SP:=0;
      IF ((GVL.Q2SP-GVL.Q2PV)>ErrPos) THEN STATE1:=1;
      END_IF
      IF ((GVL.Q2SP-GVL.Q2PV)<ErrNeg) THEN STATE1:=3;
      END_IF
   1: GVL.W2SP:=VelMaxPos;
      IF ((GVL.Q2SP-GVL.Q2PV)<DesPos) THEN STATE1:=2;
      END_IF
   2: GVL.W2SP:=VelMinPos;
      IF ((GVL.Q2SP-GVL.Q2PV)<ErrPos) THEN STATE1:=0;
      END_IF
   3: GVL.W2SP:=VelMaxNeg;
      IF ((GVL.Q2SP-GVL.Q2PV)>DesNeg) THEN STATE1:=4;
      END_IF
   4: GVL.W2SP:=VelMinNeg;
      IF ((GVL.Q2SP-GVL.Q2PV)>ErrNeg) THEN STATE1:=0;
      END_IF
 END_CASE
```

Figura 67. Programa de control de posición eje 2.

Fuente: Elaboración propia.

- El servocontrol de velocidad e integración de posición, programado para el eje 2, se muestra en la figura 68.

```
ALFA:=TAO/(TAO+TS);

GVL.W2ACT:=ALFA*W2last+(1-ALFA)*GVL.W2SP;  // Dinámica del servocontrol de velocidad
IF GVL.W2ACT>100.0 THEN
  GVL.W2ACT:=100.0;
END_IF
IF GVL.W2ACT<-100.0 THEN
  GVL.W2ACT:=-100.0;
END_IF
W2last:=GVL.W2ACT;

GVL.Q2PV:=Q2last+TS*K1*GVL.W2ACT;        // Integración de la velocidad en posición
IF GVL.Q2PV>180.0 THEN
  GVL.Q2PV:=180.0;
END_IF
IF GVL.Q2PV<-180.0 THEN
  GVL.Q2PV:=-180.0;
END_IF
Q2last:=GVL.Q2PV;
```

Figura 68. Programa control de velocidad eje 2.

Fuente: Elaboración propia.

- En la figura 69 se muestra el código de programación del control de posición a través de la máquina de estados para el eje 3 que conforma el robot UR5.

```
CASE STATE1 OF
   0: GVL.W3SP:=0;
      IF ((GVL.Q3SP-GVL.Q3PV)>ErrPos) THEN STATE1:=1;
      END_IF
      IF ((GVL.Q3SP-GVL.Q3PV)<ErrNeg) THEN STATE1:=3;
      END_IF
   1: GVL.W3SP:=VelMaxPos;
      IF ((GVL.Q3SP-GVL.Q3PV)<DesPos) THEN STATE1:=2;
      END_IF
   2: GVL.W3SP:=VelMinPos;
      IF ((GVL.Q3SP-GVL.Q3PV)<ErrPos) THEN STATE1:=0;
      END_IF
   3: GVL.W3SP:=VelMaxNeg;
      IF ((GVL.Q3SP-GVL.Q3PV)>DesNeg) THEN STATE1:=4;
      END_IF
   4: GVL.W3SP:=VelMinNeg;
      IF ((GVL.Q3SP-GVL.Q3PV)>ErrNeg) THEN STATE1:=0;
      END_IF
  END_CASE
```

Figura 69. Programa de control de posición eje 3.

Fuente: Elaboración propia.

- El servocontrol de velocidad e integración de posición, programado para el eje 3, se muestra en la figura 70.

```
ALFA:=TAO/(TAO+TS);

GVL.W3ACT:=ALFA*W3last+(1-ALFA)*GVL.W3SP;  // Dinámica del servocontrol de velocidad
IF GVL.W3ACT>100.0 THEN
  GVL.W3ACT:=100.0;
END_IF
IF GVL.W3ACT<-100.0 THEN
  GVL.W3ACT:=-100.0;
END_IF
W3last:=GVL.W3ACT;

GVL.Q3PV:=Q3last+TS*K1*GVL.W3ACT;        // Integración de la velocidad en posición
IF GVL.Q3PV>180.0 THEN
  GVL.Q3PV:=180.0;
END_IF
IF GVL.Q3PV<-180.0 THEN
  GVL.Q3PV:=-180.0;
END_IF
Q3last:=GVL.Q3PV;
```

Figura 70. Programa control de velocidad eje 3.

Fuente: Elaboración propia.

- En la figura 71 se muestra el código de programación del control de posición a través de la máquina de estados para el eje 4 que conforma el robot UR5.

```
CASE STATE1 OF
   0: GVL.W4SP:=0;
      IF ((GVL.Q4SP-GVL.Q4PV)>ErrPos) THEN STATE1:=1;
      END_IF
      IF ((GVL.Q4SP-GVL.Q4PV)<ErrNeg) THEN STATE1:=3;
      END_IF
   1: GVL.W4SP:=VelMaxPos;
      IF ((GVL.Q4SP-GVL.Q4PV)<DesPos) THEN STATE1:=2;
      END_IF
   2: GVL.W4SP:=VelMinPos;
      IF ((GVL.Q4SP-GVL.Q4PV)<ErrPos) THEN STATE1:=0;
      END_IF
   3: GVL.W4SP:=VelMaxNeg;
      IF ((GVL.Q4SP-GVL.Q4PV)>DesNeg) THEN STATE1:=4;
      END_IF
   4: GVL.W4SP:=VelMinNeg;
      IF ((GVL.Q4SP-GVL.Q4PV)>ErrNeg) THEN STATE1:=0;
      END_IF
  END_CASE
```

Figura 71. Programa de control de posición eje 4.

Fuente: Elaboración propia.

- El servocontrol de velocidad e integración de posición, programado para el eje 4, se muestra en la figura 72.

```
ALFA:=TAO/(TAO+TS);

GVL.W4ACT:=ALFA*W4last+(1-ALFA)*GVL.W4SP;  // Dinámica del servocontrol de velocidad
IF GVL.W4ACT>100.0 THEN
  GVL.W4ACT:=100.0;
END_IF
IF GVL.W4ACT<-100.0 THEN
  GVL.W4ACT:=-100.0;
END_IF
W4last:=GVL.W4ACT;

GVL.Q4PV:=Q4last+TS*K1*GVL.W4ACT;        // Integración de la velocidad en posición
IF GVL.Q4PV>180.0 THEN
  GVL.Q4PV:=180.0;
END_IF
IF GVL.Q4PV<-180.0 THEN
  GVL.Q4PV:=-180.0;
END_IF
Q4last:=GVL.Q4PV;
```

Figura 72. Programa control de velocidad eje 4.

Fuente: Elaboración propia.

- En la figura 73 se muestra el código de programación del control de posición a través de la máquina de estados para el eje 5 que conforma el robot UR5.

```
CASE STATE1 OF
   0: GVL.W5SP:=0;
      IF ((GVL.Q5SP-GVL.Q5PV)>ErrPos) THEN STATE1:=1;
      END_IF
      IF ((GVL.Q5SP-GVL.Q5PV)<ErrNeg) THEN STATE1:=3;
      END_IF
   1: GVL.W5SP:=VelMaxPos;
      IF ((GVL.Q5SP-GVL.Q5PV)<DesPos) THEN STATE1:=2;
      END_IF
   2: GVL.W5SP:=VelMinPos;
      IF ((GVL.Q5SP-GVL.Q5PV)<ErrPos) THEN STATE1:=0;
      END_IF
   3: GVL.W5SP:=VelMaxNeg;
      IF ((GVL.Q5SP-GVL.Q5PV)>DesNeg) THEN STATE1:=4;
      END_IF
   4: GVL.W5SP:=VelMinNeg;
      IF ((GVL.Q5SP-GVL.Q5PV)>ErrNeg) THEN STATE1:=0;
      END_IF
END_CASE
```

Figura 73. Programa de control de posición eje 5.

Fuente: Elaboración propia.

- El servocontrol de velocidad e integración de posición, programado para el eje 5, se muestra en la figura 74.

```
ALFA:=TAO/(TAO+TS);

GVL.W5ACT:=ALFA*W5last+(1-ALFA)*GVL.W5SP;  // Dinámica del servocontrol de velocidad
IF GVL.W5ACT>100.0 THEN
  GVL.W5ACT:=100.0;
END_IF
IF GVL.W5ACT<-100.0 THEN
  GVL.W5ACT:=-100.0;
END_IF
W5last:=GVL.W5ACT;

GVL.Q5PV:=Q5last+TS*K1*GVL.W5ACT;        // Integración de la velocidad en posición
IF GVL.Q5PV>180.0 THEN
  GVL.Q5PV:=180.0;
END_IF
IF GVL.Q5PV<-180.0 THEN
  GVL.Q5PV:=-180.0;
END_IF
Q5last:=GVL.Q5PV;
```

Figura 74. Programa control de velocidad eje 5.

Fuente: Elaboración propia.

- En la figura 75 se muestra el código de programación del control de posición a través de la máquina de estados para el eje 6 que conforma el robot UR5.

```
CASE STATE1 OF
   0: GVL.W6SP:=0;
      IF ((GVL.Q6SP-GVL.Q6PV)>ErrPos) THEN STATE1:=1;
      END_IF
      IF ((GVL.Q6SP-GVL.Q6PV)<ErrNeg) THEN STATE1:=3;
      END_IF
   1: GVL.W6SP:=VelMaxPos;
      IF ((GVL.Q6SP-GVL.Q6PV)<DesPos) THEN STATE1:=2;
      END_IF
   2: GVL.W6SP:=VelMinPos;
      IF ((GVL.Q6SP-GVL.Q6PV)<ErrPos) THEN STATE1:=0;
      END_IF
   3: GVL.W6SP:=VelMaxNeg;
      IF ((GVL.Q6SP-GVL.Q6PV)>DesNeg) THEN STATE1:=4;
      END_IF
   4: GVL.W6SP:=VelMinNeg;
      IF ((GVL.Q6SP-GVL.Q6PV)>ErrNeg) THEN STATE1:=0;
      END_IF
  END_CASE
```

Figura 75. Programa de control de posición eje 6.

Fuente: Elaboración propia.

- El servocontrol de velocidad e integración de posición, programado para el eje 5, se muestra en la figura 76.

```
ALFA:=TAO/(TAO+TS);

GVL.W6ACT:=ALFA*W6last+(1-ALFA)*GVL.W6SP;  // Dinámica del servocontrol de velocidad
IF GVL.W6ACT>100.0 THEN
  GVL.W6ACT:=100.0;
END_IF
IF GVL.W6ACT<-100.0 THEN
  GVL.W6ACT:=-100.0;
END_IF
W6last:=GVL.W6ACT;

GVL.Q6PV:=Q6last+TS*K1*GVL.W6ACT;        // Integración de la velocidad en posición
IF GVL.Q6PV>180.0 THEN
  GVL.Q6PV:=180.0;
END_IF
IF GVL.Q6PV<-180.0 THEN
  GVL.Q6PV:=-180.0;
END_IF
Q6last:=GVL.Q6PV;
```

Figura 76. Programa control de velocidad eje 6.

Fuente: Elaboración propia.

En el siguiente código se muestran las ecuaciones de cinemática directa dentro del programa CODESYS. Estas ecuaciones se obtuvieron a partir de la matriz de transformación homogénea mostrada en la sección 2.4.

- En la figura 77 se muestra el código de las ecuaciones de orientación programadas en CODESYS.

```
R11:=(C1*C234*C5*C6)+(C6*S1*S5)-(C1*S234*S6);   //ORIENTACION nx=R11
R12:=(-C1*C234*C5*S6)-(S1*S5*S6)-(C1*C6*S234);  //ORIENTACION ox=R12
R13:=(-C1*C234*S5)-(C5*S1);                     //ORIENTACION ax=R13
R21:=(C234*C5*C6*S1)-(C1*C6*S5)-(S1*S234*S6);   //ORIENTACION ny=R21
R22:=(-C234*C5*S1*S6)+(C1*S5*S6)-(C6*S1*S234);  //ORIENTACION oy=R22
R23:=(-C234*S1*S5)-(C1*C5);                     //ORIENTACION ay=R23
R31:=(S234*C5*C6)+(C234*S6);                    //ORIENTACION nz=R31
R32:=-C5*S234*S6+C234*C6;                       //ORIENTACION oz=R32
R33:=(-S234*S5);                                //ORIENTACION az=R33
```

Figura 77. Código de ecuaciones de orientación.

Fuente: Elaboración propia.

- Las ecuaciones de posición “X”, “Y” y “Z” se muestran en la figura 78.

```
Px:= (-C1*C234*S5*GVL.d6)+(C5*S1*GVL.d6)+(C1*S234*GVL.d5)+(S1*GVL.d4)+(C1*C23*GVL.a3)+(C1*C2*GVL.a2);
Py:= (-C234*S1*S5*GVL.d6)-(C1*C5*GVL.d6)+(S1*S234*GVL.d5)-(C1*GVL.d4)+(C23*S1*GVL.a3)+(C2*S1*GVL.a2);
Pz:=-S234*S5*GVL.d6-C234*GVL.d5+S23*GVL.a3+S2*GVL.a2+GVL.d1;
```

Figura 78. Código de ecuaciones de posición.

Fuente: Elaboración propia.

Los códigos que se presenta a continuación muestran las ecuaciones de la cinemática inversa programadas en el software CODESYS; theta1 véase figura 79, theta2 véase figura 80, theta3 véase figura 81, theta4 véase figura 82, theta5 véase figura 83, theta6 véase figura 84.

```
H:=ATAN2(15.0,30.0);

A:=Px-(GVL.d6*R13);
B:=(GVL.d6*R23)-Py;

Q1:=ATAN2(A,B)+ATAN2(SQRT((A*A)+(B*B)-(d4*d4)),d4);  //ANGULO DE POSICION POSITIVO

Q1N:=ATAN2(A,B)-ATAN2(SQRT((A*A)+(B*B)-(d4*d4)),d4);  //ANGULO DE POSICION NEGATIVO

GVL.Q1REQ:=Q1*180/3.14;
```

Figura 79. Código de ecuaciones de ángulo θ_1

Fuente: Elaboración propia.

```
C2:=COS(GVL.aq2);
S2:=SIN(GVL.aq2);
C3:=COS(GVL.aq3);
S3:=SIN(GVL.aq3);

KC:=C2*(GVL.a2+C3*GVL.a3)-S2*(S3*GVL.a3);
KS:=S2*(GVL.a2+C3*GVL.a3)+C2*(S3*GVL.a3);

Q2:=ATAN2(KC,KS)-ATAN2((GVL.a3*C3)+GVL.a2,(GVL.a3*S3));
GVL.Q2REQ:=Q2*180/3.14;
```

Figura 80. Código de ecuaciones de ángulo θ_2

Fuente: Elaboración propia.

```
S234:=SIN(GVL.aq2+GVL.aq3+GVL.aq4);
C234:=COS(GVL.aq2+GVL.aq3+GVL.aq4);
S5:=SIN(GVL.aq5);

KC3:=(C1*Px)+(S1*Py)-(S234*GVL.d5)+(C234*S5*GVL.d6);
KS3:=Pz-GVL.d1+(C234*GVL.d5)+(S234*S5*GVL.d6);

IC3:=((KS3*KS3)+(KC3*KC3)-(GVL.a2*GVL.a2)-(GVL.a3*GVL.a3))/(2*GVL.a2*GVL.a3);
 IF IC3 >1.0 THEN IC3:= 1.0;
    END_IF
 IF IC3 <-1.0 THEN IC3:=-1.0;
    END_IF

IS3:=SQRT(1-(IC3*IC3));

Q3:=ATAN2(IS3,IC3);      //ANGULO DE POSICION POSITIVO
GVL.Q3REQ:=Q3*180/3.14;
```

Figura 81. Código de ecuaciones de ángulo θ_3

Fuente: Elaboración propia.

```
A234:=IC5*IC6;
B234:=IS6;
D234:=(C1*R11)+(S1*R21);
E234:=R31;

Q234:=ATAN2((A234*D234)+(B234*E234),(A234*E234)-(B234*D234));

Q4:=Q234-Q2-Q3;          //ANGULO DE POSICION POSITIVO

Q4N:= Q4+GVL.pi;         //ANGULO DE POSICIÓN NEGATIVO

GVL.Q4REQ:=Q4*180/3.14;
```

Figura 82. Código de ecuaciones de ángulo θ_4

Fuente: Elaboración propia.

```
S1:=SIN(GVL.aq1);
C1:=COS(GVL.aq1);
BC1:=(S1*S1)*(R11*R11)-(2*S1*R11*C1*R21)+(C1*C1)*(R21*R21);
BC2:=(S1*S1)*(R12*R12)-(2*S1*R12*C1*R22)+(C1*C1)*(R22*R22);


IC5:=(S1*R13)-(C1*R23);
IS5:=SQRT(BC1+BC2);

Q5:=ATAN2(IS5,IC5);       //ANGULO DE POSICION POSITIVO
GVL.Q5REQ:=Q5*180/3.14;
```

Figura 83. Código de ecuaciones de ángulo θ_5

Fuente: Elaboración propia.

```
IC6:= ((S1*R11)-(C1*R21))/IS5;

IS6:= ((C1*R22)-(S1*R12))/IS5;

Q6:=ATAN2(IS6,IC6);
GVL.Q6REQ:=Q6*180/3.14;
```

Figura 84. Código de ecuaciones de ángulo θ_6

Fuente: Elaboración propia.

5.3.2.1 Interfaz de operación del controlador

Los programas de control de posición por máquina de estados para cada eje, además del programa de servocontrol de velocidad e integración de posición de cada articulación, incluyendo las ecuaciones programadas para la cinemática inversa y directa, conforman el programa principal. A continuación, se muestra la línea principal del texto programado del proyecto.

- Generación del pulso de periodo. El código para la ejecución de tiempo de muestreo se muestra en la figura 85.

```
IF RegCnt>10 THEN
    PulsoPer:=TRUE;
    RegCnt:=0;
ELSE
    RegCnt:=RegCnt+1;
    PulsoPer:=FALSE;
END_IF

// Ciclo de Control
IF PulsoPer THEN
```

Figura 85. Código para la ejecución de tiempo de muestreo.

Fuente: Elaboración propia.

- En la figura 86, se puede observar el código para habilitar el control de cinemática inversa y el modo manual para cada uno de los ejes.

```
IF (Modo_Oper=TRUE) THEN

IKIN();

GVL.Q1SP:=GVL.Q1REQ;
GVL.Q2SP:=GVL.Q2REQ;
GVL.Q3SP:=GVL.Q3REQ;
GVL.Q4SP:=GVL.Q4REQ;
GVL.Q5SP:=GVL.Q5REQ;
GVL.Q6SP:=GVL.Q6REQ;

GVL.Q1MAN:=GVL.Q1REQ;
GVL.Q2MAN:=GVL.Q2REQ;
GVL.Q3MAN:=GVL.Q3REQ;
GVL.Q4MAN:=GVL.Q4REQ;
GVL.Q5MAN:=GVL.Q5REQ;
GVL.Q6MAN:=GVL.Q6REQ;

ELSE
    GVL.Q1SP:=GVL.Q1MAN;
    GVL.Q2SP:=GVL.Q2MAN;
    GVL.Q3SP:=GVL.Q3MAN;
    GVL.Q4SP:=GVL.Q4MAN;
    GVL.Q5SP:=GVL.Q5MAN;
    GVL.Q6SP:=GVL.Q6MAN;

END_IF
```

Figura 86. Código para habilitar el modo cinemático inverso o modo manual.

Fuente: Elaboración propia.

- El código para habilitar el control de posición y de velocidad se muestran en la figura 87.

```
CTRL_EJE1();
CTRL_EJE2();
CTRL_EJE3();
CTRL_EJE4();
CTRL_EJE5();
CTRL_EJE6();

GVL.Q1COM:=GVL.W1ACT;
GVL.Q2COM:=GVL.W2ACT;
GVL.Q3COM:=GVL.W3ACT;
GVL.Q4COM:=GVL.W4ACT;
GVL.Q5COM:=GVL.W5ACT;
GVL.Q6COM:=GVL.W6ACT;
```

Figura 87. Código para habilitar el control de posición y velocidad.

Fuente: Elaboración propia.

- En la figura 88, se puede observar el código de comunicación que permite escribir los valores.

```
GVL.Q1COMX:=REAL_TO_INT(GVL.Q1COM*100+10000);
GVL.Q2COMX:=REAL_TO_INT(GVL.Q2COM*100+10000);
GVL.Q3COMX:=REAL_TO_INT(GVL.Q3COM*100+10000);
GVL.Q4COMX:=REAL_TO_INT(GVL.Q4COM*100+10000);
GVL.Q5COMX:=REAL_TO_INT(GVL.Q5COM*100+10000);
GVL.Q6COMX:=REAL_TO_INT(GVL.Q6COM*100+10000);
```

Figura 88. Código para habilitar la escritura de comunicación.

Fuente: Elaboración propia

5.3.3 Interfaz de operación del sistema del robot UR5

La interfaz de usuario del robot virtual UR5, se compone con dos ventanas: la del controlador en CODESYS y el modelo del robot en Blender. Para ejecutar el sistema del robot virtual UR5, se tiene que tener abierto el controlador en CODESYS y el modelo del robot en Blender. Una vez abiertas estas ventanas, se selecciona la ventana del controlador para activar el sistema, en donde se escribe los valores de la posición angular que debe de ejecutar el robot en el eslabón seleccionado dentro de la interfaz de control. Para ello es necesario compilar el código e iniciar la sesión para para que empiece a mandar un registro de datos a la ventana de Blender donde se encuentra en modelo del robot virtual, véase figura 89.

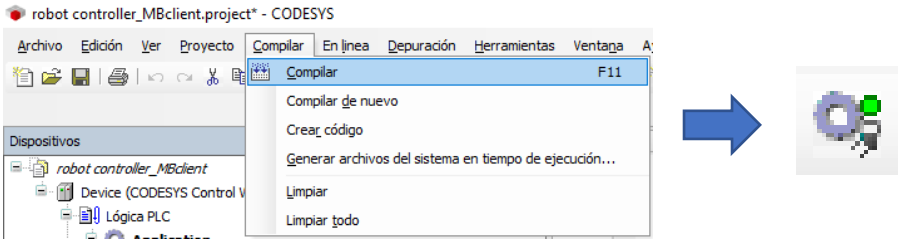


Figura 89. Activación del programa del controlador.

Fuente: Elaboración Propia.

Al mismo tiempo es necesario activar la herramienta "*toggle system console"* que se encuentra en el software Blender, la cual permite abrir la consola para monitorear los datos que se están enviando del software CODESYS hacia Blender, véase figura 90.

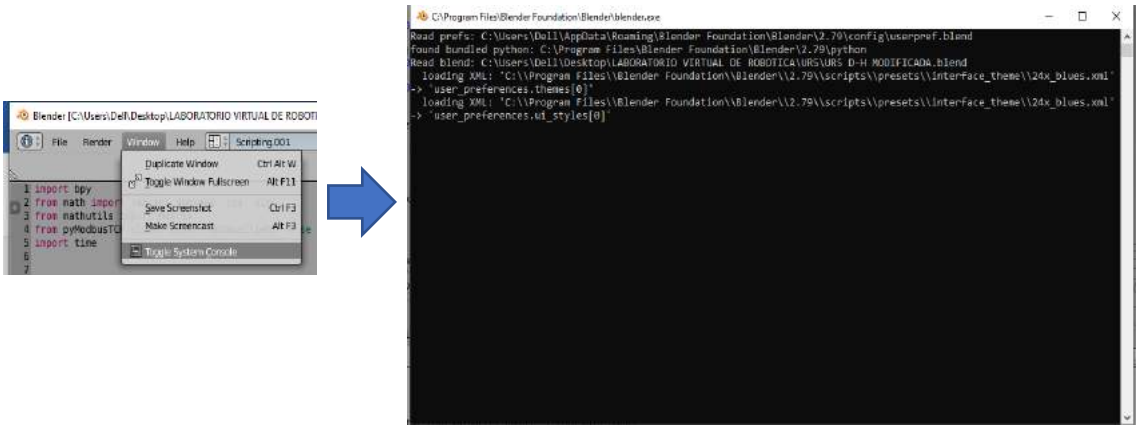


Figura 90. Activación de consola en Blender.

Fuente: Elaboración Propia.

La interfaz de usuario de CODESYS contiene el controlador del sistema. Se tiene el programa en modo manual desde el inicio de arranque del sistema, el cual se puede ajustar a modo cartesiano dando click en el botón de modo de operación. Se tiene la pantalla de movimiento articular, en la cual por medio de perillas para seis rotaciones se dan los valores deseados a las articulaciones; a partir de esos valores se obtiene la posición “X”, “Y” y “Z”, y además se muestran los ángulos y posición reales que el robot manda. En la figura 91 se puede observar el panel de control con las que se hicieron las pruebas entre ambas plataformas.

Figura 91. Interfaz de control CODESYS.

Fuente: Elaboración Propia.

La segunda ventana está dentro del software Blender, contiene el modelo del robot UR5 en 3D y parte del programa del protocolo de comunicación industrial modbus. Para validar que el programa está corriendo en tiempo real, se teclea la tecla “p”

para iniciar el programa en Blender y se visualiza la comunicación a través del protocolo modbus entre los softwares CODESYS-Blender, mostrando en la consola de Blender el monitoreo de datos que se escriben en la interfaz del controlador, véase figura 92.

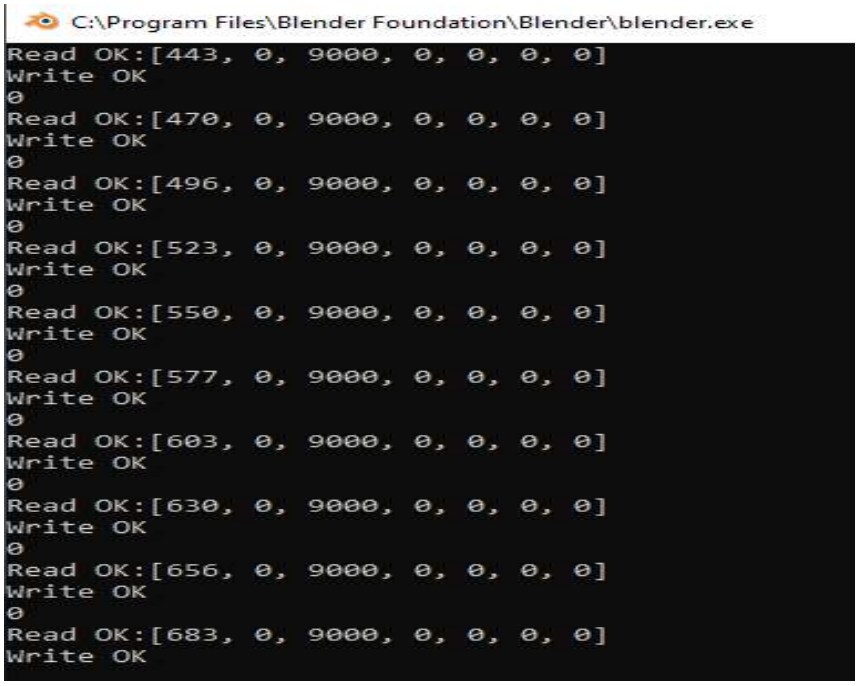


Figura 92. Monitoreo de comunicación de datos CODESYS-Blender.

Fuente: Elaboración Propia.

5.4 Pruebas de funcionamiento del robot UR5

A continuación, se muestran las pruebas de funcionamiento que se realizaron al robot UR5, en el modo de operación manual y cartesiano para los seis ejes de rotación que conforman el sistema. Se muestra la posición de origen en la figura 93.

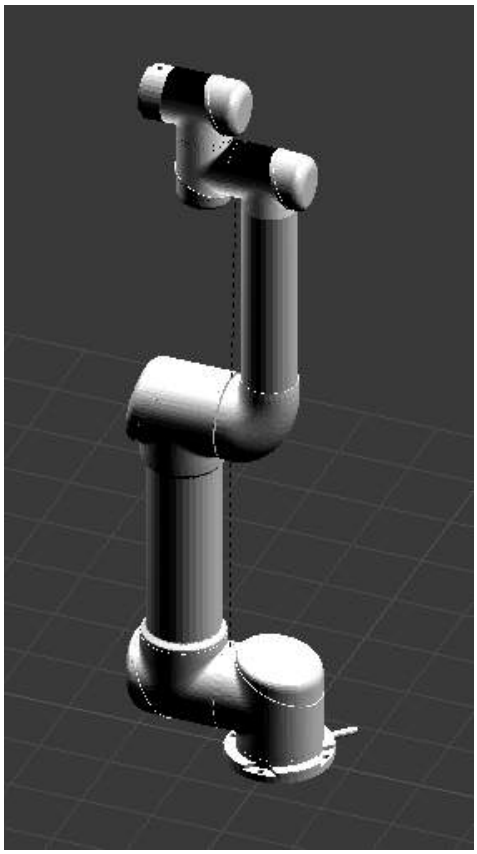


Figura 93. Posición de origen del Robot UR5.

Fuente: Elaboración Propia.

5.4.1 Pruebas en modo articulado

A continuación, se muestra en vista superior la posición de origen del robot UR5, teniendo los siguientes valores para las siguientes articulaciones; $\theta_1 = 0$, $\theta_2 = 0$, $\theta_3 = 0$, $\theta_4 = 0$, $\theta_5 = 0$, $\theta_6 = 0$, véase figura 94.

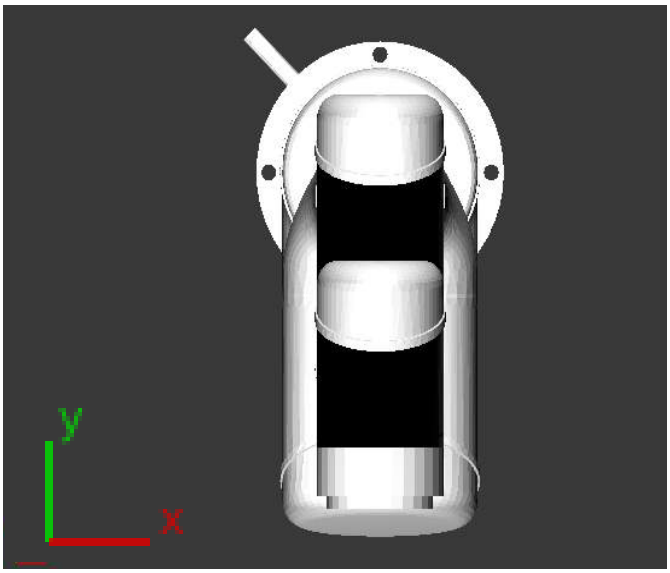


Figura 94. Vista superior, posición de origen.

Fuente: Elaboración Propia.

Para la prueba del primer eje el valor escrito en el controlador es igual $\theta_1 = 45°$, donde el eje realiza un movimiento rotación al valor deseado, véanse figura 95 y figura 96.

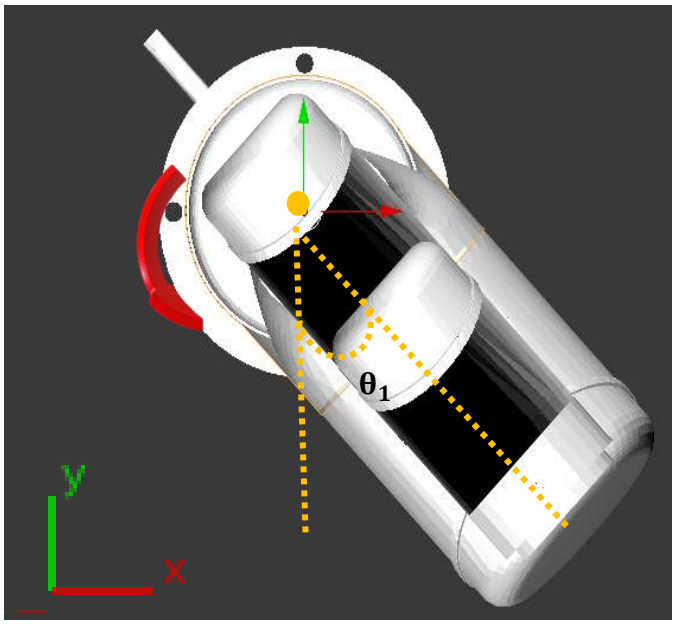


Figura 95. Vista superior θ_1.

Fuente: Elaboración Propia.

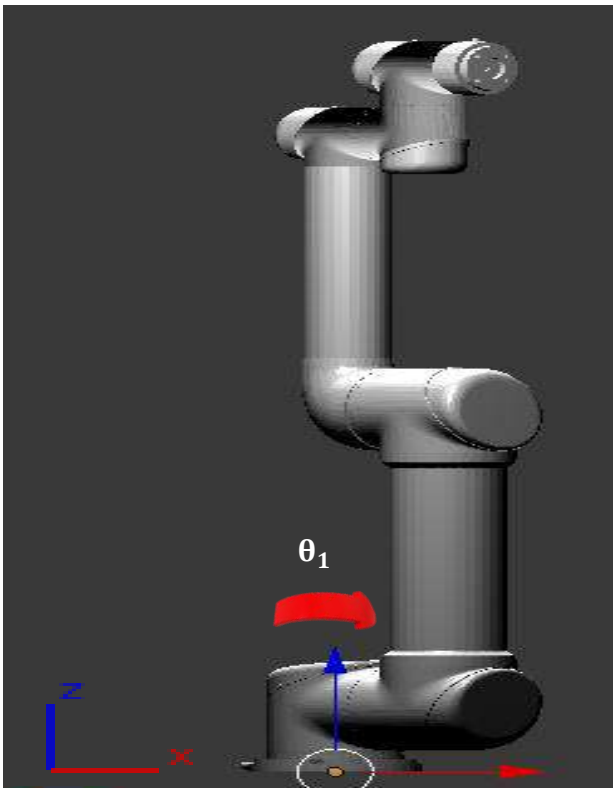


Figura 96. Vista isométrica θ_1.

Fuente: Elaboración Propia.

Para la prueba del segundo eje el valor escrito en el controlador es igual $\theta_2 = 45°$, donde el eje realiza un movimiento rotación al valor deseado, véanse figura 97 y figura 98.

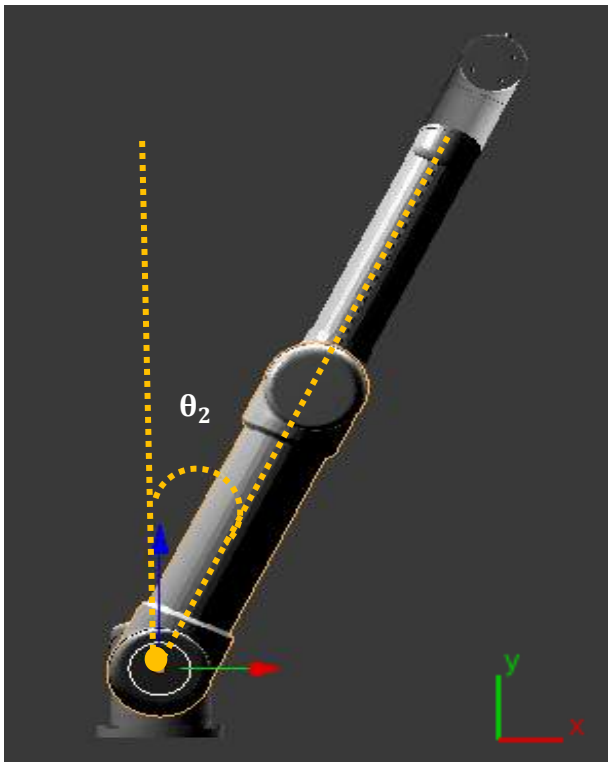


Figura 97. Vista frontal θ_2.

Fuente: Elaboración Propia.

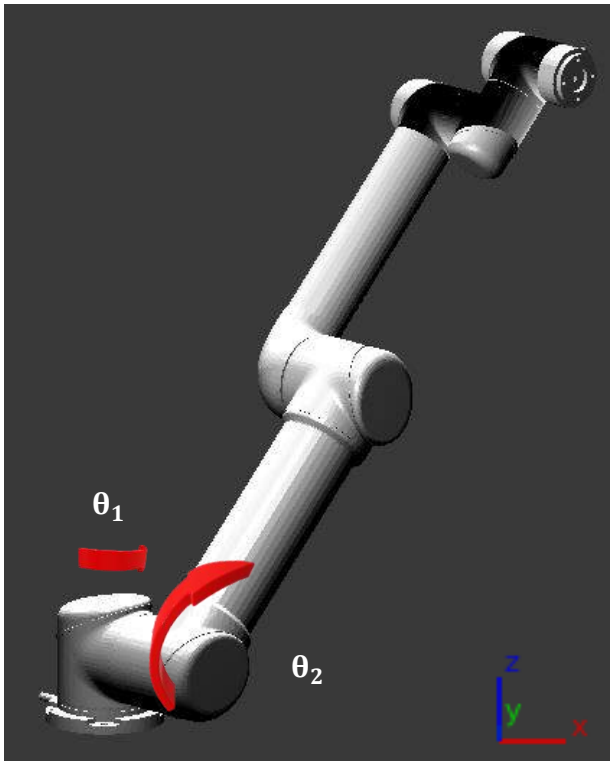


Figura 98. Vista isométrica θ_1, θ_2.

Fuente: Elaboración Propia.

Para la demostración del tercer eje, se considera la posición de la articulación θ_1, θ_2, como punto de inicio para el movimiento de $\theta_3 = -45°$, donde el eje realiza un movimiento rotación al valor deseado, véanse figura 99 y figura 100.

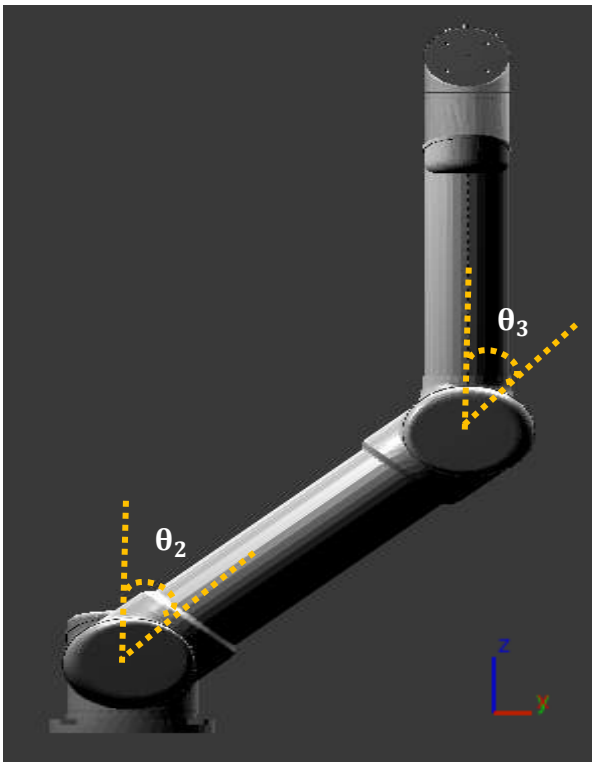


Figura 99. Vista frontal θ_3.

Fuente: Elaboración Propia.

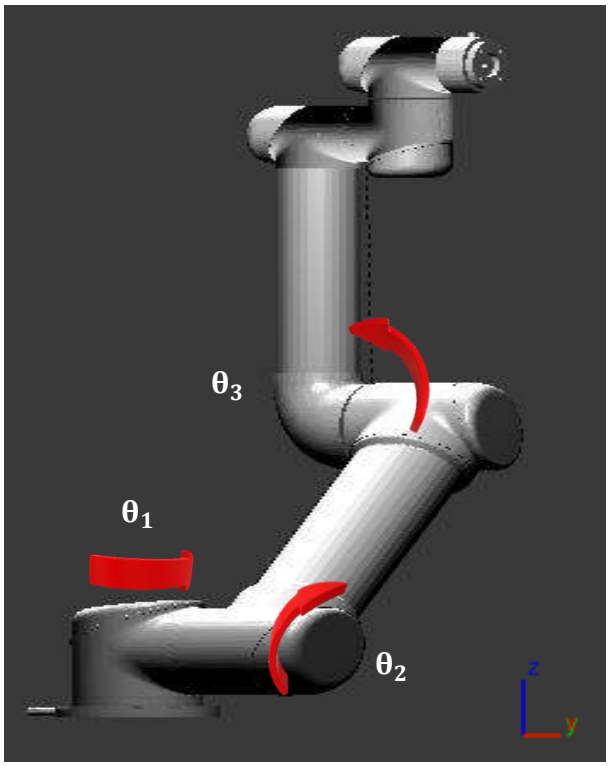


Figura 100. Vista isométrica $\theta_1, \theta_2, \theta_3$.

Fuente: Elaboración Propia.

Para la prueba del cuarto eje, se considera la posición de la articulación θ_2 y θ_3 como en el punto de inicio para el movimiento de $\theta_4 = 90°$, donde el eje realiza un movimiento rotación al valor deseado, véanse figura 101 y figura 102.

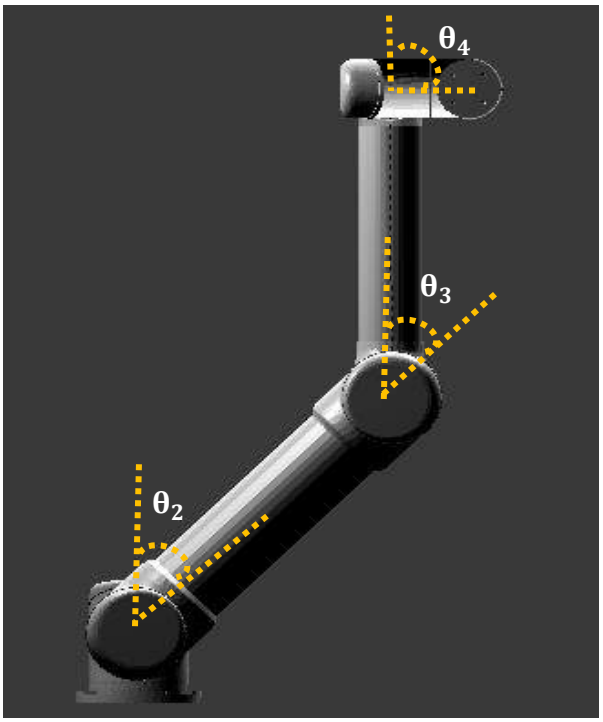


Figura 101. Vista frontal θ_4.

Fuente: Elaboración Propia.

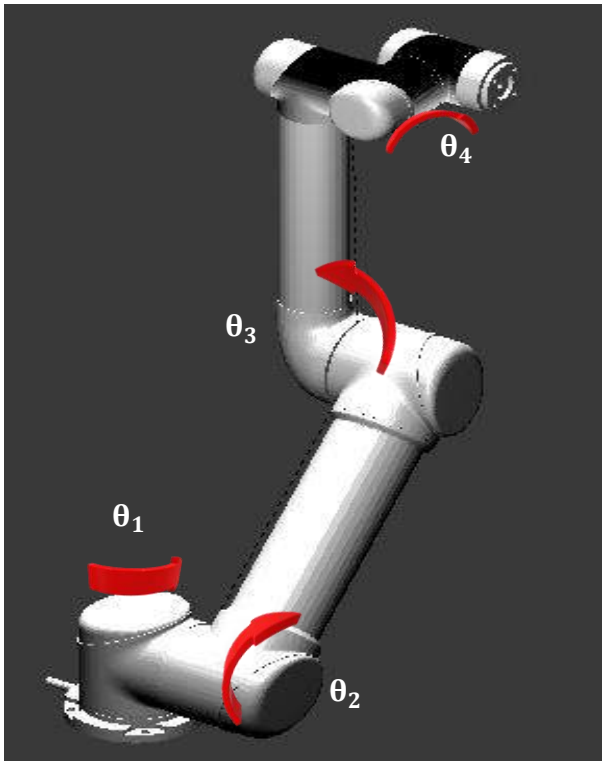


Figura 102. Vista isométrica $\theta_1, \theta_2, \theta_3, \theta_4$.

Fuente: Elaboración Propia.

Para la demostración del quinto eje, se considera la posición de la articulación θ_2, θ_2, θ_3, θ_4 como en el punto de inició para el movimiento de $\theta_5 = 20°$, donde el eje realiza el movimiento rotación al valor deseado, véanse figura 103 y figura 104.

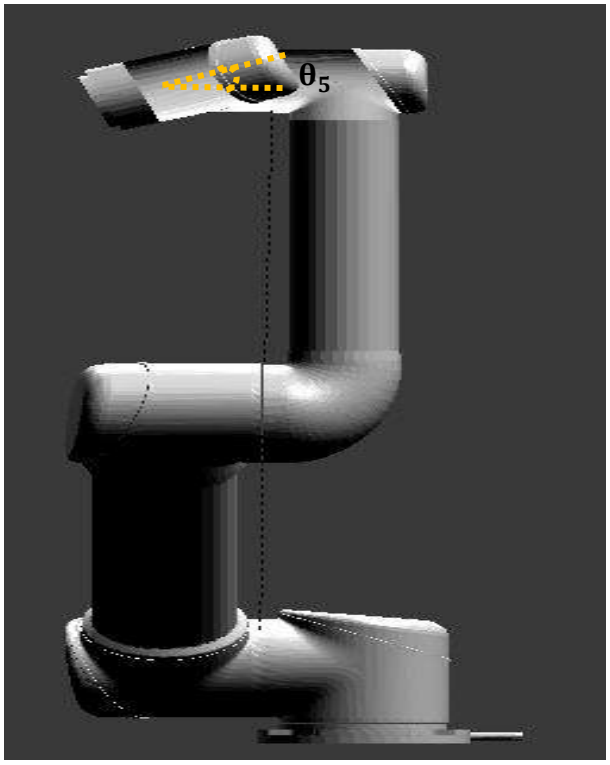


Figura 103. Vista frontal θ_5.

Fuente: Elaboración Propia.

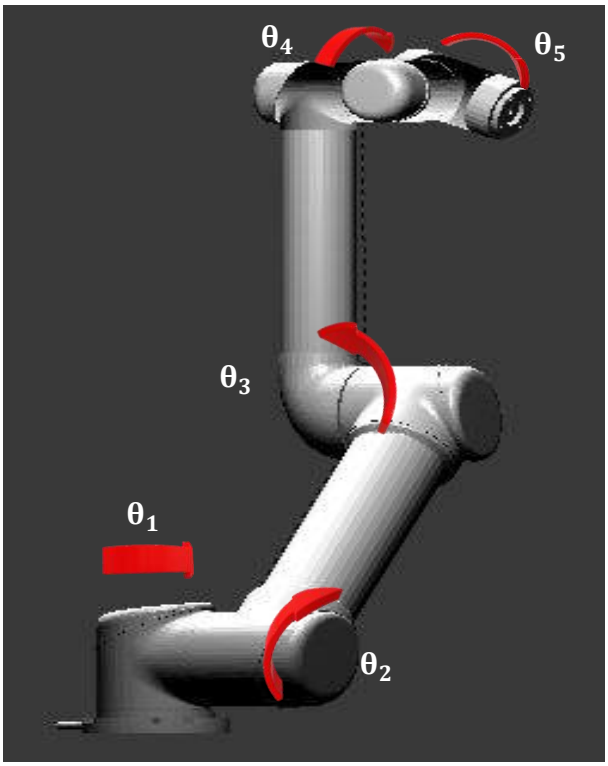
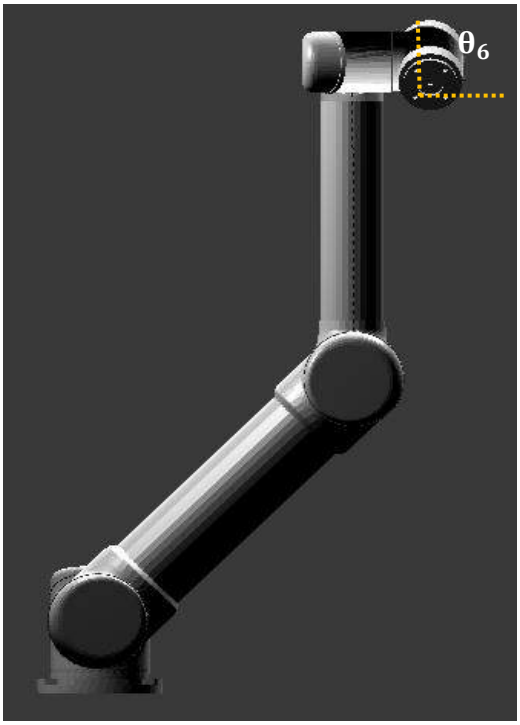


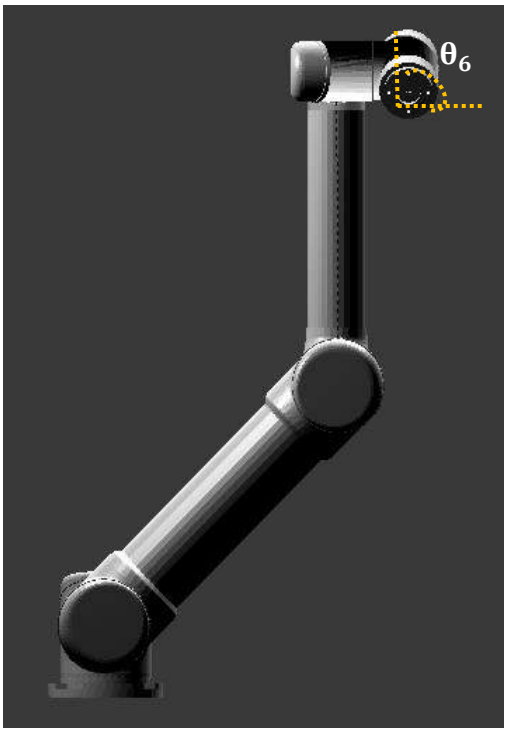
Figura 104. Vista isométrica $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$.

Fuente: Elaboración Propia.

Para la prueba del sexto eje de rotación, se considera la posición de la articulación θ_1 θ_2, θ_3, θ_4 θ_5, como en el punto de inició para el movimiento de $\theta_6 = 45°$, donde el eje realiza un movimiento rotación al valor deseado, véanse figura 105 y figura 106.



A) Vista frontal $\theta_6 = 0°$.



B) Vista frontal $\theta_6 = 45°$.

Figura 105. Vista frontal θ_6.

Fuente: Elaboración Propia.

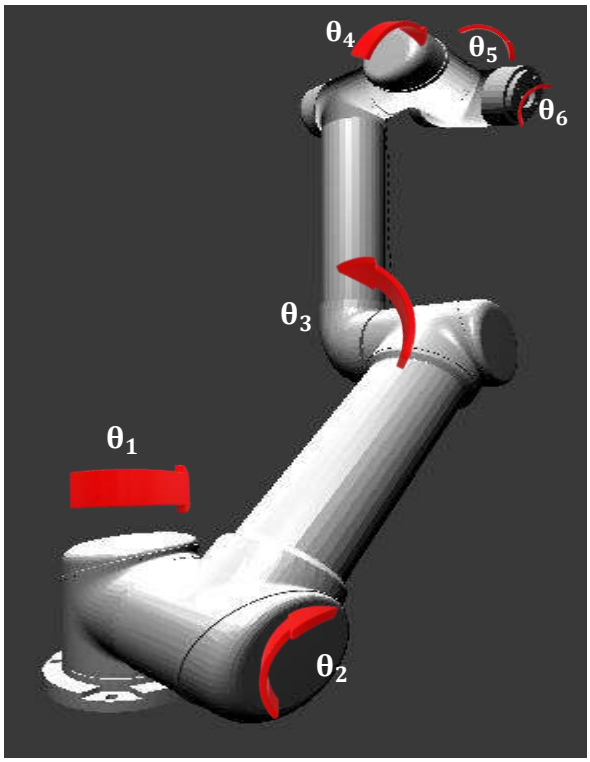


Figura 106. Vista isométrica $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$.

Fuente: Elaboración Propia.

Las señales de trayectoria de la posición de los ejes se grafican durante la simulación virtual en el espacio de las articulaciones, para cada uno de los seis ejes que conforman el robot UR5. Las coordenadas correspondientes a la posición de inicio para cada eje son θ_1= 0, θ_2= 0, θ_3= 0, θ_4= 0, θ_5= 0 y θ_6= 0, que posteriormente se ajusta la trayectoria de cada eje a θ_1= 73.27, θ_2= 36.05, θ_3= 31.15, θ_4= 76.45, θ_5= 133.78 y θ_6= 92.29 las cuales se muestran en la figura 107. Las coordenadas presentadas del robot corresponden al rendimiento real en las condiciones de funcionamiento supuestas.

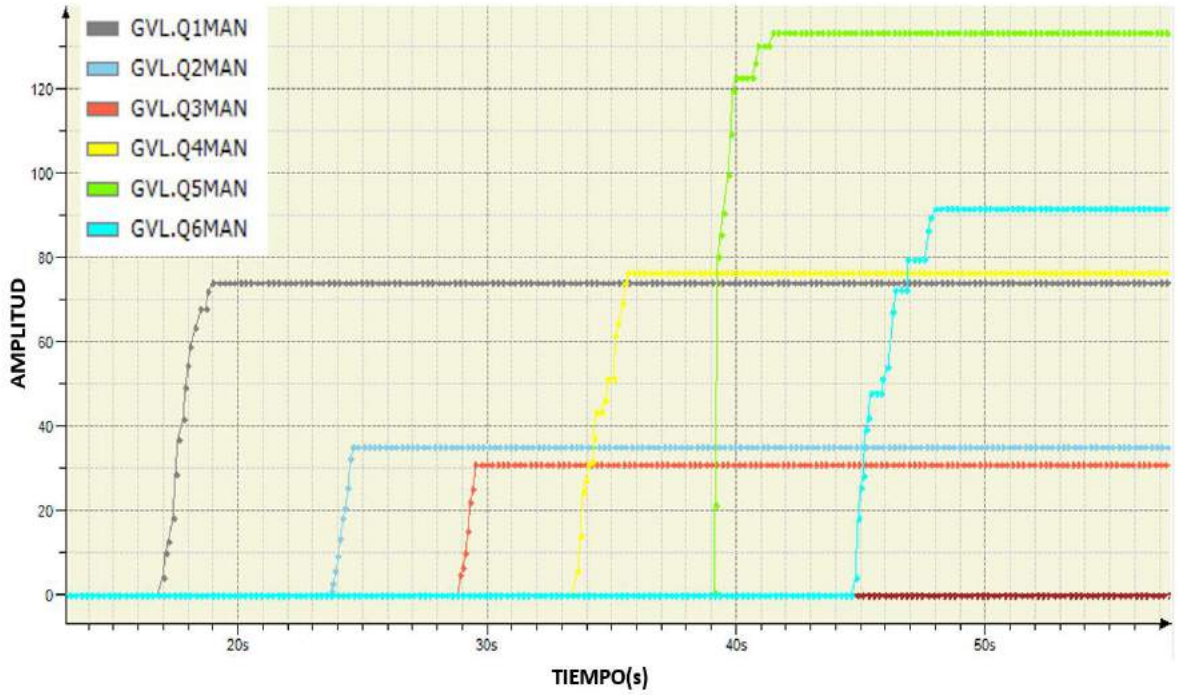


Figura 107. Coordenadas de los 6 ejes en el espacio articular.

Fuente: Elaboración Propia.

La figura 108 muestra el cambio de trayectoria a partir de la posición alcanzada que se muestra en la figura 107. Las coordenadas correspondientes al cambio de posición para cada eje son θ_1= 173.12, θ_2= 54.70, θ_3= 63.56, θ_4= 110.08, θ_5= 133.78 y θ_6= 37.86.

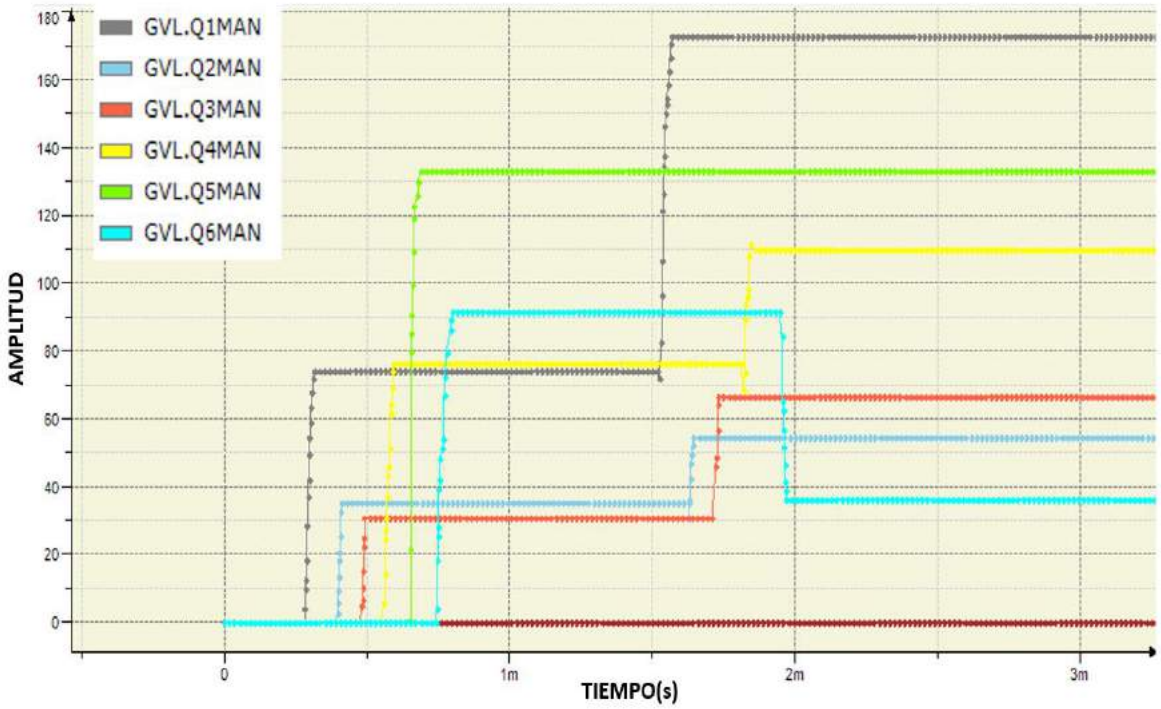


Figura 108. Cambio de coordenadas de los 6 ejes en el espacio articular.

Fuente: Elaboración Propia.

5.4.2 Pruebas en modo cartesiano

A continuación, se muestran las pruebas de funcionamiento que se realizaron al robot UR5, en el modo de operación cartesiano considerando la posición “X”, “Y” y “Z”.

Se trazan diferentes trayectorias para verificar los cálculos planteados en el algoritmo de cinemática inversa, considerando la posición “X”, “Y”, y “Z”, partiendo de la posición en donde los valores son X=0, Y=0 y Z=0 y posteriormente se alcanza la posición X=170, Y=150 y Z=90. Véanse la figura 109 los resultados obtenidos.

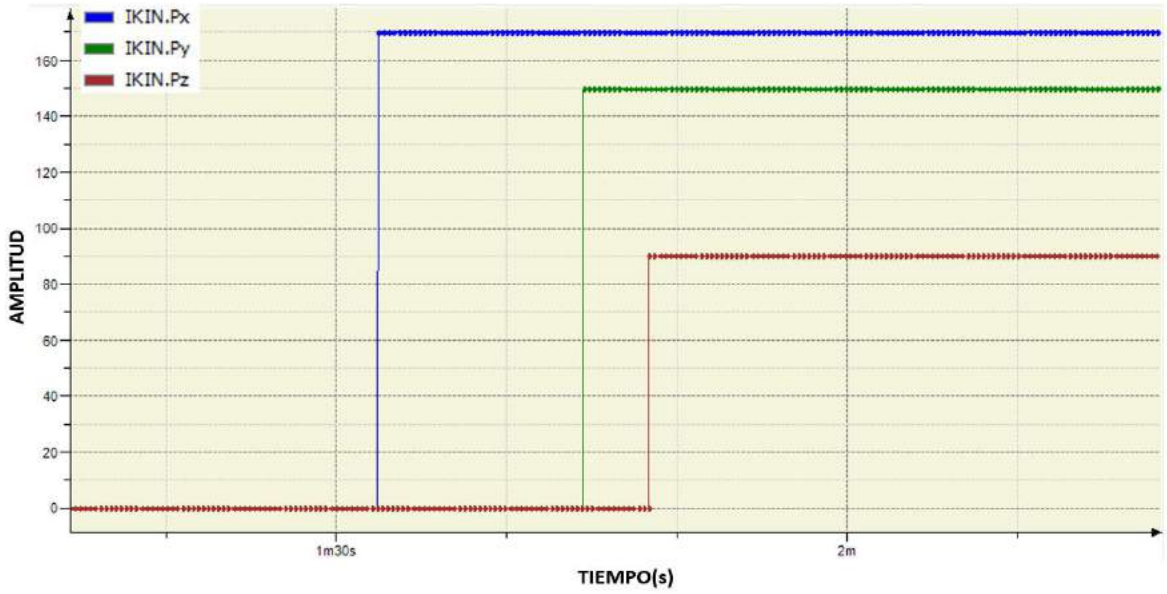


Figura 109. Coordenadas de posición X, Y y Z.

Fuente: Elaboración Propia.

La figura 110 muestra el cambio de trayectoria a partir de la posición alcanzada que se muestra en la figura 109. Las coordenadas correspondientes al cambio de posición son X= 173.12, Y= 54.70 y Z = 37.86.

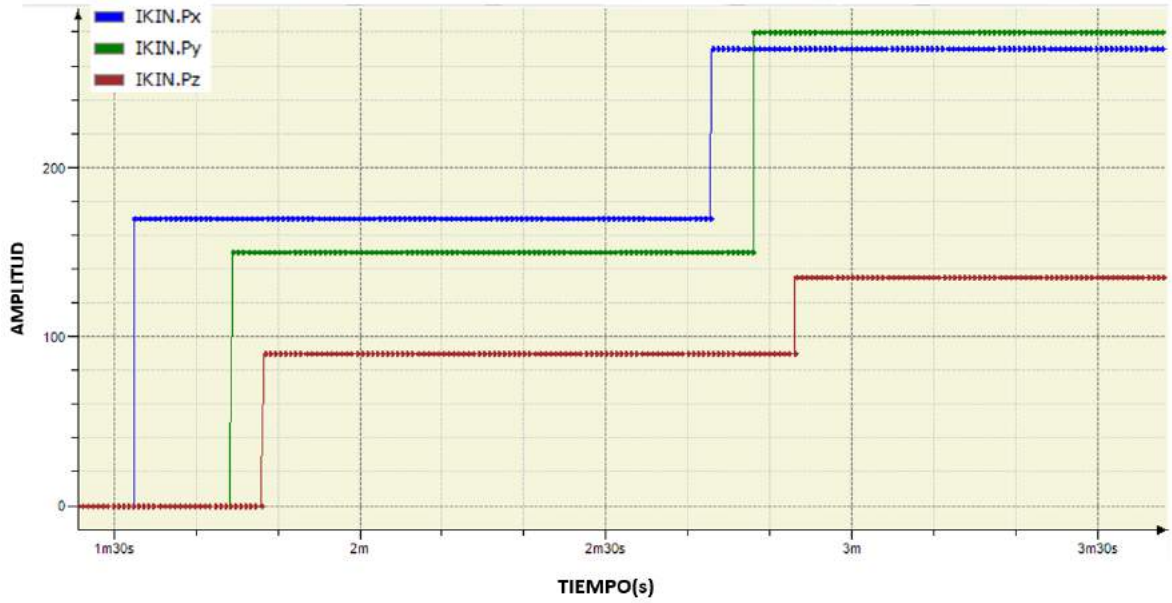


Figura 110. Cambio de coordenadas de posición X, Y y Z.

Fuente: Elaboración Propia.

CAPÍTULO 6 CONCLUSIONES

6.1 Discusión sobre el sistema desarrollado

El robot virtual UR5 desarrollado en Blender se implementó en una estación de trabajo con conectividad Modbus a un SoftPLC externo corriendo en la misma estación de trabajo a la que se puede acceder de forma remota. Este robot virtual debe controlarse en un esquema de simulación “hardware in the loop” mediante la programación del controlador CODESYS.

El sistema desarrollado es un robot UR5 virtual que necesita ser programado tanto para el nivel de control conjunto como para el control cartesiano. Las respuestas dinámicas de los actuadores y su control de posición se pueden modelar, programar, simular y verificar en el controlador externo. La solución cinemática inversa también debe programarse en el controlador externo para el control del robot virtual para lograr una posición y orientación cartesianas deseadas y finalmente para el seguimiento de la trayectoria.

El sistema se desarrolló como una herramienta educativa para la capacitación remota en línea sobre brazos robóticos, en particular el robot tipo UR. El robot virtual es una interfaz visual con conectividad Modbus a un controlador externo, donde el usuario enfrenta el desafío de desarrollar el código para modelar los actuadores, diseñar e implementar el control del actuador (posición y velocidad angular), y programar las ecuaciones de cinemática directa e inversa para poder controlar la posición y orientación del robot.

6.2 Discusión sobre las técnicas de modelación y control

La simulación realizada en Blender y el control cinemático inverso programado en CODESYS funcionan con la velocidad que se genere al aplicar un voltaje al motor CD. El actuador permitirá que se mueva, pero eventualmente se puede estabilizar con un control de velocidad y al normalizar el control de velocidad se estabiliza el control de posición esto implica colocar el control para las articulaciones.

La estrategia de control cascada para el control de posición del motor de CD se llevó a cabo con el método máquina de estados finitos. Este método de control se compara con otro esquema de control cascada IMC-P para el control de posición y velocidad. Esta comparación muestra que el control propuesto mantiene las variables controladas dentro de rangos aceptables.

El método EBD fue comparado con el método BLT considerando un modelo de motor de orden reducido, los resultados de esta comparación muestran mejor desempeño a favor del método EBD. La discretización del modelo se utilizó también para estimar el par de carga en tiempo real del motor, que se puede utilizar en estrategias de control.

Considerando que se tiene un reductor en el motor CD, si se tiene amplificación del par de carga se podría controlar la carga que se obtenga, pero si hay desviaciones en la carga en realidad se tiene perturbaciones en la velocidad y el estimador de par de carga debe ser diferente y se debe compensar. Se propone que al tener un servomotor con reductor en donde la carga es trasmitida al eje del motor pueda compensar las variaciones de la carga del robot considerando la estrategia de control de eje independiente.

La estrategia propuesta no implica utilizar el modelo dinámico del robot, si no únicamente el modelo dinámico de los actuadores y se controlan con las variaciones en el par de carga como perturbaciones.

El control propuesto compensará la dinámica del motor, es decir, si se tiene controlado el eje, se puede decir que se tiene controlada la dinámica del robot. Este control robusto permite monitorear la velocidad y se tiene un estimador de par de carga que está actuando como término derivativo, esto logra un control robusto. Esta estrategia de control articular independiente fue programada y verificada, además se combinó con el control cinemático inverso para mostrar la aplicabilidad del sistema desarrollado para el aprendizaje de temas de robótica y control.

6.3 Discusión sobre la aplicación del robot virtual

Los cursos de diseño robótico que incluyen cinemática y control son convencionalmente más teóricos ya que los controladores de robots comerciales se consideran cajas negras y normalmente solo se pueden usar para entrenar robótica a nivel de usuario (orientado a tareas).

La característica principal de esta implementación es tener entidades separadas: el robot virtual y el sistema de control del robot, como en los sistemas robóticos reales que se componen del robot físico y el sistema de control. Los sistemas desarrollados ayudan a los estudiantes a aprender conceptos prácticos de cinemática de robots y control de articulaciones independientes, que son dos niveles básicos de control que deben aprenderse en el área de robótica, particularmente en el diseño de brazos robóticos.

El usuario debe desarrollar algoritmos de control en tiempo real para los dos niveles críticos de control del robot: el control articular independiente y el control cinemático inverso para la posición y orientación cartesianas.

Se debe de considerar que el equipo o los equipos que se requieran para llevar a cabo el proyecto, estos deben tener la capacidad de instalación de un sistema operativo para instalar los softwares utilizados, además de poder comunicarse a través de un protocolo de red industrial para la comunicación entre los softwares. Los softwares que se utilizan para el desarrollo de estos proyectos se actualizan periódicamente a nuevas versiones que pueden limitar el uso de herramientas.

6.4 Principales conclusiones

Los laboratorios virtuales ya se han implementado con éxito en universidades e industrias, para aprendizaje práctico en cursos de posgrado, talleres para estudiantes o capacitación especializada en automatización, robótica, estaciones virtuales de pintura automotriz y tecnología en el contexto de la Industria 4.0. El sistema robótico virtual UR5 presentado se desarrolló en Blender y requiere un controlador externo con conectividad Modbus para controlar el robot virtual

articulado de 6 grados de libertad. Para validar el sistema desarrollado, se implementan estrategias de cinemática inversa y control de movimiento en CODESYS SoftPLC.

El robot virtual UR5 fue desarrollado siguiendo las tendencias educativas en tecnología basada en el uso de simulación por computadora y aplicaciones de realidad virtual aplicados en la industria 4.0. Utiliza el software Radmin VPN como software de control remoto, lo que nos permite conectarnos a nuestro robot virtual desde cualquier lugar y tener un control total sobre él. La característica principal de esta implementación es tener entidades separadas: el robot virtual y el sistema de control del robot, como en los sistemas robóticos reales que están compuestos por el robot físico y el sistema de control. La arquitectura hardware in the loop permite el desarrollo y verificación del sistema de control en tiempo real del robot UR5, permitiendo a los usuarios probar diferentes técnicas de control a través del protocolo de comunicación ModBus. En este estudio se propuso y verificó una técnica novedosa para el control de la posición de las articulaciones basada en un FSM con los robots virtuales UR5.

Se cumplió con los objetivos de diseñar un robot virtual a partir de un modelo 3D de un robot UR5 en el software Blender, en donde este robot virtual está conformado por servo-actuadores, su programa de control cinemático es externo y se encuentra dentro del programa CODESYS en el contexto de la industria 4.0, donde el reto del usuario es proponer técnicas para sistema de control de posición y velocidad para controlar los servo-actuadores.

El sistema propuesto se desarrolló como una plataforma educativa para la implementación de los algoritmos de control requeridos, es decir, para el control de cada una de las articulaciones del robot y para el control cinemático inverso del robot. Los estudiantes pueden generar su propio algoritmo de control y probarlo con el simulador de robot 3D. De esta forma, los usuarios pueden desarrollar y comparar diferentes estrategias de control en tiempo real. Por tanto, el sistema es una plataforma adecuada para enseñar y aprender la cinemática del robot UR5 y su control.

6.5 Trabajo a futuro

- Los robots colaborativos requieren sensores de fuerza, en el desarrollo del presente proyecto se definió la estimación dinámica del par de carga de cada eje de movimiento, esta estimación puede ser usada para incluirse en algoritmos de seguridad en robots colaborativos.
- Desarrollar el intérprete de lenguaje de programación UR script para poder interactuar con el control cinemático, el sistema desarrollado también sería útil para la capacitación y entrenamiento en programación de robots.
- Desarrollar diferentes modelos de robots virtuales para expandir el laboratorio virtual y aplicar diferentes metodologías de estrategias de control y cinemática inversa.

REFERENCIAS BIBLIOGRÁFICAS

[1] J. I. Olszewska. The virtual classroom: a new cyber physical system. SAMI, 2021; 187-192, p.

DOI:10.1109/SAMI50585.2021.9378678

[2] J. Leng, H. Zhang, D. Yan, Q. Liu, X. Chen, D. Zhang. Digital twin-driven manufacturing cyber-physical system for parallel controlling of smart workshop. J Ambient intell human comput, 2018; 1155–1166, p.

https://doi.org/10.1007/s12652-018-0881-5

[3] L. Perez, S.R. Jimenez, N. Rodriguez, R. Usamentiaga, D. F. García. Digital twin and virtual reality based methodology for multi-robot manufacturing cell commissioning. Applied sciences, 2020; 1-18, p.

DOI:10.3390/app10103633.

[4] H. D Wuttke, K. Henke, R. Hutschenreuter. Digital twins in remote labs. Cyber-physical systems and digital twins, 2019; 289-297, p.

DOI: 10.1007/978-3-030-23162-0_26

[5] I. Verner, D. Cuperman, S. Gamer, A. Polishuk. Training Robot Manipulation skills through practice with digital twin of baxter. International Association of Online Engineering, 2019; 58-70, p.

https://doi.org/10.3991/ijoe.v15i09.10493

[6] K. Droder, P. Bobka, T. Germann, F. Gabriel, F. Dietrich. A machine learning enhanced digital twin approach, for human robot collaboration. Conference on assembly technologies and systems, 2018; 187-192, p.

https://doi.org/10.1016/j.procir.2018.02.010

[7] S. Jeršovy, A. Tepljakov. Digital twins in extended reality for control system applications. International Conference on Telecommunications and Signal Processing, 2020; 274-279, p.

DOI: 10.1109/TSP49548.2020.9163557

[8] A. Rukangu, A. Tuttle and K. Johnsen. Virtual reality for remote controlled robotics in engineering education. IEEE Conference on virtual reality and 3D user interfaces abstracts and workshops, 2021; 751-752, p.

DOI: 10.1109/VRW52623.2021.00258.

[9] Tselegkaridis, S., Sapounidis, T. Simulators in educational robotics: A Review. Educ. Sci., 2021; 2-12, p.

https://doi.org/ 10.3390/educsci11010011

[10] Hao, C., Zheng, A.., Wang, Y., Jiang, B. Experiment information system based on an online virtual laboratory. Future Internet, 2021; 1-19, p.

https://doi.org/10.3390/fi13020027

[11] K. Velisek, R. Holubek, D. Delgado, R. Ruzarovsky, N. Vetrikova. Design of a robotized workstation making use of the integration of cad models and robotic simulation software as way of pairing and comparing real and virtual environment. MATEC Web of Conferences, 2017; 1-6, p.

DOI: 10.1051/matecconf/20179405008

[12] M. Lin, L. San, Y. Ding. Construction of robotic virtual laboratory system based on Unity3D. IOP Conf. Series: Materials Science and Engineering, 2020; 1-7, p.

DOI:10.1088/1757-899X/768/7/072084

[13] X. Xu, P. Guo, J. Zhai, X. Zeng. Robotic kinematics teaching system with virtual reality, remote control and an on–site laboratory. International Journal of Mechanical Engineering Education, 2018; 197-220, p.

DOI: 10.1177/0306419018807376

[14] V. Petrović, B. Nikolić, K. Jovanović, V. Potkonjak. Development of virtual laboratory for mechatronic systems. Advances in robot design and intelligent control, 2016; 622-630, p.

DOI 10.1007/978-3-319-49058-8_68

[15] C. Li, L. Fu, L. Wang. Innovate engineering education by using virtual laboratory platform based industrial robot. Chinese Control and Decision Conference, 2018; 3467-3472, p.

DOI: 10.1109/CCDC.2018.8407723

[16] Le, P.-N.; Kang, H.-J. A robotic calibration method using a model-based identification technique and an invasive weed optimization neural network compensator. Appl. Sci., 2020; 1-14, p.

https://doi.org/10.3390/app10207320

[17] D. Antonelli, A. Christopoulos, M-J. Laakso, V. Dagienė, A. Juškevičienė, V. Masiulionytė-Dagienė, M.Mądziel M, Stadnicka D, Stylios C. A Virtual Reality Laboratory for Blended Learning Education: Design, Implementation and Evaluation. Education Sciences. 2023; 13(5):528. https://doi.org/10.3390/educsci13050528

[18] Alam, A., Mohanty, A. (2023). Evaluation of Software Engineering Virtual Laboratory in Determining Undergraduate Students' Conceptual Understanding: A Blended Learning Model Using Collaborative-Creative Virtual Learning Environment Employing Critical Pedagogy. In: Shakya, S., Du, KL., Ntalianis, K. (eds) Sentiment Analysis and Deep Learning. Advances in Intelligent Systems and Computing, vol 1432. Springer, Singapore. https://doi.org/10.1007/978-981-19-5443-6_65

[19] IBM. Que es industria 4.0. https://www.ibm.com/mx-es/topics/industry-4-0#:~:text=La%20Industria%204.0%20es%20la,mejoran%20y%20distribuyen%20sus%20productos.

(18 de agosto del 2023)

[20] C. Cortés, J.M Landeta, J.G Bocarando, F. Aguilar, Martin Larios. El Entorno de la Industria 4.0: Implicaciones y Perspectivas Futuras. Conciencia Tecnológica, núm. 54, 2017 Instituto Tecnológico de Aguascalientes, México. 2017; 1-19, p.

[21] M. Lin, L.San, Y. Ding. Construction of Robotic Virtual Laboratory System Based on Unity3D. IOP Conf. Series: Materials Science and Engineering, 2020;1-7, p.

[22] W. Zhao. Application of virtual reality technology in motion simulation and control of industrial robot. Proc. SPIE 12610, Third International Conference on Artificial Intelligence and Computer Engineering, 2023.

[23] F. Arenas, F. Martell, I.Y. Sanchez, C.A. Paredes. Virtual laboratory for online learning of UR5 robotic arm inverse kinematic and joint motion control. In Proceedings of the 2021 International Conference on Electrical, Computer and Energy Technologies (ICECET), Cape Town, South Africa, 9–10 December 2021.

[24] N.P. Silva, S. Eloy, R. Resende. Robotic construction analysis: simulation with virtual reality. Heliyon 8, 2022; 1-13, p.

[25] F. Arenas, F. Martell, I.Y. Sanchez. Virtual UR5 Robot for Online Learning of Inverse Kinematics and Independent Joint Control Validated with FSM Position Control. Robotics, 2023; 1-16, p.

[26] S. Ay, M. Kutay, T. Ercan. A Low cost manipulator training set. 8 6th International Conference on Control Engineering & Information Technology (CEIT), 2018; 1-6, p.

[27] E. I. Merzlikina, N. S. Dilbikova and G. V. Farafonov. Use of CODESYS for Laboratory Classes in the Online Mode. VI International Conference on Information Technologies in Engineering Education (Inforino), 2022; 1-4.

[28] K. Jahnavi, P. Sivraj. Teaching and learning robotic arm model. In Proceedings of the 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT 2017), 2017; 1570–1575, p.

[29] J. Villalobos, I.Y. Sanchez, F. Martell. Singularity Analysis and Complete Methods to Compute the Inverse Kinematics for a 6-DOF UR/TM-Type Robot. Robotics 2022; 11, 137, p.

[30] P. Mesmer, M. Neubauer, A. Lechler, A. Verl. Robust design of independent joint control of industrial robots with secondary encoders, Robotics and Computer-Integrated Manufacturing, 2022; 1- 11, p.

[31] M. S. Tran, N. B. Lee, V. T. Nguyen, D. C. Dang, E. H. Choi, Y. B. Kim. Independent Joint Control System Design Method for Robot Motion Reconstruction. International Conference on Advanced Engineering Theory and Applications, 2018; 627-638, p.

[32] G. Zabala. Robótica; guía teórica y práctica. User express. ISBN 978-987-1347-56-8. 2007; 127, p.

[33] A. Barrientos, L. F. Peñin, C. Balaguer, R. Aracil. Fundamentos de robótica. España; Mc Graw Hill, 2007; 31, p.

[34] Devendra K. Chaturtvedi. Modeling and Simulation of Systems, Using MATLAB and Simulink. USA: CRC Press, 2010; 84-86, p.

[35] R. Hernández. Introducción a los sistemas de control; conceptos, aplicaciones y simulación con matlab. México: Pearson, 2010; 95, p.

[36] M. A. Pérez, E. V. Cuevas, D. Zaldívar. Fundamentos de robótica y mecatrónica con Matlab y simulink. México; Alfaomega, 2014; 597, p.

[37] M. Alavi, S. Aliaga, M. Murga. Máquinas de estado finitos. ISSN24115-2322, vol. 8, 2016; 42-43, p.

[38] M. Lidon. Blender; curso práctico. España: Ra-Ma, 2017; 13, p.

[39] M. A Ridao. Introducción a la programación de autómatas programables usando Codesys. España; Editorial Universidad de Sevilla, 2016; 8, p.

[40] Y. Jiang, X. Hu, S. Wu. Transformation Matrix for Time Discretization Based on Tustin's Method. Hindawi Publishing Corporation. Mathematical Problems in Engineering Volume 2014; 9, p.

[41] K. Ogata. Sistemas de Control en Tiempo Discreto. México: Prentice Hall Hispanoamericana, 1996; 228, p.

[42] G. Huang, S. Lee. Pc-based PID speed control in DC motor. International conference on audio, language and image processing: 2008; 400-407, p.

DOI: 10.1109/ICALIP.2008.4590052

[43] J. C. Basilio, S. R. Matos. Design of PI and PID controllers with transient performance specification. IEEE, transactions on education: 2002; 364-370, p.

DOI: 10.1109/TE.2002.804399

[44] I.S. Okoro and C.O. Enwerem, Performance assessment of a model-based DC motor scheme, Applications of Modelling and Simulation, 3(3), 2019; 145–153, p.

[45] K. Tripathi. IMC based tuning method of PID controller for DC motor speed control; 2016, https://www.semanticscholar.org/paper/IMC-based-tuning-methodof-PID-controller-for-DCTripathi/ef2f2f5deabfd3068f8311d735db8f52becf0e26

[46] A.U. Omeiza, P. Avong, and B. Kwembe, DC motor speed control using internal model controller: Industrial Transformation Strategy, IJEAT, 9(5), 2020; 300–306, p.

[47] Sailan, K., & Kuhnert, K.D. DC motor angular position control using PID controller for the porpuse of controlling the hydraulic pump. International conference on control, engineering & information technology. proceedings engineering & technology: 2018; 22-26, p.

[48] Tahtawi, A.R., Somantri, Y., & Haritman., E. Design and implementation of PID control-based FSM algorithm on line following robot. Jurnal teknologi rekayasa: 2017; 23-30 p.

[49] Hanzic, F., Jezernik, K., & Cehner., S. Mechatronic Control System on a Finite-State Machine. Automatika: 2013; 126-138, p.

DOI:10.7305/automatika.54-1.298

[50] Universal Robots. Universal robots e-series manual de usuario. https://cfzcobots.com/wp-content/uploads/2018/06/UR3e_User_Manual_es_Global.pdf.

(16 de marzo del 2023)

ANEXOS

A. Conference ICASAT, 2021 / Book chapter.

Title: Comparison of Euler's Backward Difference and Bilinear Transform Discretization Methods for Modeling and Simulation of a DC Motor.

Abstract: The modelling and simulation of mechatronics systems are becoming more important with the appearance of the concept of cyberphysical systems. Virtual axes of motion can be simulated with simple DC motor models which require discretization methods. Bilinear transform method is commonly used as a precise discretization method. This study compares the Euler's backwards differences method and the bilinear transform method in both time and frequency domains for the discretization of a model of a DC motor. For the backward differences method two design criteria are used: one is the computation of the discrete pole in terms of the time constant and the sampling time, and the second is choosing a sample time corresponding to one fifteenth of the time constant. The Euler's backwards differences method is found suitable and even better than bilinear transform method for this type of application. This study shows that Euler´s Backwards Discretization using a sampling time of τ/15 is good to reproduce a first order dynamic response of a conventional DC motor.

Reference: Filemon Arenas Rosales - Fernando Martell Chávez - Irma Yolanda Sánchez Chávez – Rigoberto Lopez Padilla – Luis Manuel Valentin Coronado. Comparison of Euler´s Backward Difference and Bilinear Transform Discretization Methods for Modeling and Simulation of a DC Motor. Recent trends in sustaible engineering. ICASAT 2021. Lecture note in networks and systems, vol 297, springer, cham. https://doi.org/10.1007/978-3-030-82064-0_5.

B. Journal Mechatronic Systems and Control, 2021.

Title: Discrete Time DC Motor Model for Load Torque Estimation for PID-IMC Speed Control.

Abstract: This study explores the direct current (DC) motor load torque calculation using an effective discretization of the motor dynamics and the proper choice of sample time. The Euler backward differences and the bilinear transform methods are compared considering an adequate sampling time that is not usually specified in discretization methods. The Euler backward differences discretization of a reduced order motor model was solved and computed for the load torque. The real-time estimation of the motor load can be used in an internal model control-based proportional integral derivative control strategy. Results show that the proposed control scheme has robust performance for variable load and variable speed for motor control applications.

Reference: Filemon Arenas Rosales - Fernando Martell Chávez – Irma Yolanda Sánchez Chávez. Discrete Time DC Motor Model for Load Torque Estimation for PID-IMC Speed Control. Journal mechatronics systems and control, acta press. from journal (201) mechatronics systems and control-2021. vol.49,2021. issn(online): 2561-178x. doi: 10.2316/j.2021.201-0265

C. Conference ICECET, 2021.

Title: Virtual laboratory for online learning of UR 5 robotic arm inverse kinematic and joint motion control.

Abstract: Nowadays, remote work is becoming common in schools and universities, distance online laboratories can apply technologies such as virtual reality machines that facilitate the teaching and learning of engineering topics such as robotics and mechatronics. Virtual robots and their control systems can be implemented in open-source software and low-cost hardware platforms. This article presents the methodology for the development of an UR articulated virtual robot, designed as a tool for the students to carry out remote online practices oriented to program and test control algorithms. The 3D modeling and animation of the virtual robot is done in blender in such a way that the robot needs to interact with an external controller in a hardware in the loop simulation scheme. The external controller can be programed in Codesys and uses Modbus TCP as a communication protocol. Control algorithms using a finite states machine methods for the joints positioning, combined with a speed control based on a proportional integral derivative control, were used to validate the jog manual operation of the robot. The kinematic equations were also programed in Codesys to implement the inverse kinematic control of the virtual robot. The virtual robot is found to be suitable for practical learning of control methods of the UR5 robotic manipulator.

Reference: F. Arenas-Rosales, F. Martell-Chavez, I. Y. Sanchez-Chavez and C. A. Paredes-Orta, "Virtual laboratory for online learning of UR5 robotic arm inverse kinematic and joint motion control," 2021 International Conference on Electrical, Computer and Energy Technologies (ICECET), Cape Town, South Africa, 2021, pp. 1-6, doi: 10.1109/ICECET52533.2021.9698490.

D. Journal Robotics MDPI, 2023.

Title: Virtual UR5 Robot for Online Learning of Inverse Kinematics and Independent Joint Control Validated with FSM Position Control.

Abstract: Virtual remote laboratories have already been successfully implemented in educational centers for practical learning of mechatronics and robotic systems. This article presents the development of a virtual articulated UR-type robot, designed as an educational tool that is suitable for programming and evaluating both the inverse kinematics control of the robot and the independent control of the robot joints. The 3D model of the virtual robot was developed in the Blender V2.79 software and uses the Modbus TCP industrial communication protocol for the communication to an external controller implemented in Codesys V3.5 software. The developed system allows the students to generate and test their own control algorithm for the robot joints with the visualization of the achieved performance in 3D and real time. Tailored control systems can be compared on the virtual robot. In this study, a novel technique for the joint position control based on an FSM is proposed and verified with the virtual UR5 robots to prove that the developed system is a suitable platform to teach and learn the inverse kinematics control and independent joint control of the UR5 robotic arm.

Reference: Arenas-Rosales, F.; Martell-Chavez, F.; Sanchez-Chavez, I.Y.; Paredes-Orta, C.A. Virtual UR5 Robot for Online Learning of Inverse Kinematics and Independent Joint Control Validated with FSM Position Control. Robotics 2023, 12, 23. https://doi.org/10.3390/robotics12010023.