CENTRO DE INVESTIGACIONES EN OPTICA, A.C.

POSGRADO INTERINSTITUCIONAL DE CIENCIA Y TECNOLOGÍA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA BIN PICKING CON SU GEMELO VIRTUAL VISUALIZADO EN DISPOSITIVOS MÓVILES MEDIANTE REALIDAD AUMENTADA

TESIS

QUE PARA OBTENER EL GRADO

ACADÉMICO DE

MAESTRO EN CIENCIAS Y TECNOLOGÍA

EN LA ESPECIALIDAD DE

MECATRÓNICA

PRESENTA

INGENIERO, ALAN FERNANDO SANTACRUZ RODRÍGUEZ

DIRECTOR

DOCTOR, CARLOS ALBERTO PAREDES ORTA

_____

V.°B.° Dr. Carlos Alberto Paredes Orta

AGUASCALIENTES, AGUASCALIENTES, Febrero 2023

Agradecimientos

A mis grandiosos padres que siempre su apoyo ha estado presente y el cariño nunca ha faltado.

A mis hermanos en los que siempre he encontrado un gran apoyo.

Al Doctor Carlos Alberto Paredes Orta, por sus enseñanzas, apoyo y escucha.

Al Doctor Fernando Martell, por su rectitud, constancia, y apoyo para la elaboración de este proyecto.

Al Doctor Luis Valentín, y sus grandes desafíos que me hizo mirar más allá.

Al todos los que me apoyaron para la conclusión de este trabajo, muchas gracias, colaboradores y amigos.

Resumen

Este trabajo aborda la construcción de un gemelo digital para la capacitación en robótica abordando desde su modelado cinemático y dinámico hasta la generación del entorno virtual de una aplicación Pick and Place. Este entorno virtual, permite visualizar los efectos de diferentes técnicas de control como herramienta de capacitación por medio del esquema Hardware in the Loop, permitiendo interactuar al usuario en el diseño del controlador, y a su vez, con el fin de demostrar que el sistema es controlable, se presta el control resuelto mediante una interface de realidad aumentada.

Índice general

Agradecimientos I

Resumen II

1. Introducción 1

2. Marco teórico 4

2.1. Modelado matemático en 3D 4
2.2. Robot 6
2.3. Modelado cinemático de robots 7
2.4. Trayectoria de Jerk Mínimo 8
2.5. Modelado dinámicos de robots 10
2.6. Dinámica directa 14
2.7. Métodos númericos de integración 14
2.8. Sistemas de primer orden y el tiempo de estabilización 16
2.9. Control PID 18
2.10. Control por IMC 18
2.11. Robot SCARA 20
2.12. Espacio de configuraciones de un robot 21
2.13. Gemelo digital 21
2.14. Realidad aumentada 23
2.15. Modbus 24

3. Metodología 25

3.1. Modelado del Robot SCARA 26
3.1.1. Modelado 3D 26
3.1.2. Modelado de la cinemática directa 28
3.1.3. Modelado del espacio de configuraciones 29

3.1.4. Cinemática inversa del Robot SCARA por el método geométrico 33
3.1.5. Velocidades en el Robot SCARA 36
3.1.6. Cinemática inversa por un método numérico 37
3.1.7. Generación de una trayectoria punto a punto 39
3.2. Modelado Dinámico del robot SCARA 40
3.2.1. Modelo simplificado 42
3.2.2. Añadiendo los términos de fricción 44
3.2.3. Dinámica directa 45
3.2.4. Simulación en tiempo real 46
3.2.5. Ajustes en la dinámica para la simulación en tiempo real 48
3.3. Control dinámico del robot SCARA 49
3.4. Diseño del gemelo digital en Unity 3D 56
3.4.1. Cinemática en Unity 3D 57
3.4.2. Comportamiento dinámico del gemelo virtual 60
3.4.3. Interacción con las piezas para la aplicación Pick and Place . . . 63
3.4.4. Control para una aplicación Pick and Place 65
3.5. Simulación HIL 65
3.5.1. Comunicación usando el protocolo Modbus 66
3.5.2. Arquitectura para la simulación HIL 67
3.6. Control mediante realidad aumentada del gemelo digital 70
3.6.1. Conexión del robot 3D en Unity con Realidad Aumentada 71
3.6.2. Ajuste de la posición en base a los marcadores de realidad aumentada 73
3.6.3. Flujo de control del robot virtual para aplicaciones Pick and Place 74

4. Resultados **75**
4.1. Integración de Robot Virtual SCARA 75
4.2. Simulador para la educación 78
4.2.1. Simulación a nivel de cinemático con realidad aumentada 80
4.2.2. Simulación HIL con controladores externos 81

5. Conclusiones **83**

6. Trabajo futuro **85**

7. Anexos **86**

Índice de figuras

2.1. Representación de la Ley de la mano derecha 5
2.2. Ejemplos de articulaciones 7
2.3. Trayectoria de Jerk Mínimo 9
2.4. Analogía de la fuerza centrifuga 11
2.5. Analogía de la fuerza de Coriolis 11
2.6. Relación de τ y la estabilización del sistema 17
2.7. Diagrama de bloques del controlador PID 18
2.8. Diagrama del control IMC 18
2.9. Diagrama del control IMC, agrupamiento 19
2.10. Diagrama del control IMC, reducido 19
2.11. Diagrama general de un robot SCARA 21
2.12. Ejemplos de espacios de configuración 21
2.13. Representación de un gemelo digital 22
2.14. Esquema de simulación HIL para el sistema robótico 22
2.15. Ejemplo de una aplicación usando realidad aumentada 23
2.16. Interacciones de los clientes y los tipos de memorias en el protocolo Modbus 24

3.1. Diagrama en V del proyecto 25
3.2. Plano del robot SCARA, generado del diseño 3D 26
3.3. Modelo 3D del Robot SCARA Delta DRS60L 27
3.4. Modelo 3D del robot SCARA en la mesa polar 27
3.5. Diseño de la mesa polar 28
3.6. Plano del espacio del trabajo del Robot SCARA DRS60L 29
3.7. Primera aproximación del espacio de configuraciones 30
3.8. Segunda aproximación del espacio de trabajo 31
3.9. Plano para la aproximación del espacio de trabajo 31

3.10. Triángulo para la aproximación del espacio de trabajo 31
3.11. Tercera aproximación del espacio de trabajo 32
3.12. Planteamiento geométrico de la cinemática inversa 33
3.13. Cálculo de α 33
3.14. Solución de q_2 34
3.15. Solución de γ 34
3.16. Ubicación del ángulo β 35
3.17. Relación entre las dos soluciones geométricas 35
3.18. Optimización del error 37
3.19. Cinemática inversa de manera numérica 38
3.20. Trayectoria de Jerk mínimo con limitación de velocidad y aceleración . . 39
3.21. Diagrama dinámico del robot SCARA 43
3.22. Comparativa de distintos métodos de integración numérica 47
3.23. Pruebas de estabilidad de la simulación de la dinámica directa 47
3.24. Comparativa entre la dinámica directa y la inversa 48
3.25. Diagrama de la estrategia de control para el robot SCARA 53
3.26. Respuesta del controlador de la articulación 2, ante las variaciones . . 55
3.27. Respuesta del sistema robótico usando el control propuesto 55
3.28. Creación del proyecto dentro de Unity 3D 56
3.29. Robot importado en Unity 3D 56
3.30. Jerarquía del robot SCARA en Unity 3D 57
3.31. Ejemplo de movimiento del Robot SCARA en Unity 57
3.32. Ubicación de las articulaciones en Unity 3D 58
3.33. Robot con scripts en Unity 3D para el modelado cinemático 59
3.34. Pruebas de movimiento en Unity 3D 59
3.35. “Articulated Body en Unity” 60
3.36. Control del gemelo virtual en Unity 61
3.37. Idenficicación de la articulación 1, Entrada vs Velocidad 61
3.38. Respuesta del controlador de la articulaciòn 1 ante variaciones de fase y ganancia 62
3.39. Respuesta del controlador de la articulaciòn 2 ante variaciones de fase y ganancia 63
3.40. Respuesta del controlador de la articulaciòn 3 ante variaciones de fase y ganancia 63

3.41. Maquina de estado de las piezas del entorno virtual 64
3.42. Entorno virtual con el robot SCARA y su interacción con las piezas . . . 64
3.43. Diagrama de flujo del control Scara para aplicaciones pick and place . . 65
3.44. Diagrama de flujo para pruebas de velocidad en Modbus 66
3.45. Tiempos de respuesta del protocolo Modbus 67
3.46. Esquemas de simulación HIL 67
3.47. Prueba de comunicación Unity-Labview (Unity) 69
3.48. Prueba de comunicación Unity-Labview (Labview) 69
3.49. Creación de la base de datos de marcadores 70
3.50. Creación de un marcador RA, robot 70
3.51. Creación de un marcador RA, objetivo 71
3.52. Marcadores RA construidos 71
3.53. Descarga de la base de datos 71
3.54. Referencia del robot 3D con respecto al marcador de realidad aumentada 72
3.55. Interfaz de pruebas cinemáticas con realidad aumentada 72
3.56. Pruebas de detección y posicionamiento del robot con realidad aumentada 72
3.57. Posiciones detectadas con realidad aumentada 73
3.58. Diagrama de flujo de la aplicación Pick and Place usando realidad aumentada 74

4.1. Secuencia de diseño 75
4.2. Ejemplo de movimiento del Robot SCARA en Unity 75
4.3. Diferencias entre la dinámica directa e inversa en el modelo optimizado 76
4.4. Armado del gemelo virtual con la librería de física de Unity 3D 76
4.5. Control de las articulaciones en el robot virtual 77
4.6. Diagrama de control del robot virtual 78
4.7. Entorno virtual de la aplicación Pick and Place 78
4.8. Niveles de simulación en la aplicación creada 79
4.9. Control del robot virtual por medio de los marcadores de realidad aumentada 79
4.10. Enlace entre la simulación dinámica y la visualización de realidad aumentada 80
4.11. Simulación cinemática a travéz de la realidad aumentada 80
4.12. Estación de simulación HIL 81

Índice de anexos

A. Restricción del espacio de trabajo implementado en Python 85

B. Trayectoria de Jerk Mínimo con restricciones de velocidad y aceleración en MATLAB 86

C. Método de Newton-Rapshon para la solución de la cinemática inversa en MATLAB 88

D. Código de la simulación de la dinámica directa simplificada con el control propuesto en MATLAB 90

E. CinematicaDirecta.cs 94

F. QtoUnity.cs 96

G. ServoControlador.cs 98

H. ControlConTeaching.cs 101

I. ControlScaraDinamico.cs 112

J. TransformadaDeEspacio.m 118

1 Introducción

Este proyecto contribuye a la iniciativa de la red de laboratorios virtuales del Consejo Nacional de Ciencia y Tecnología, CONACYT, la cual fue presentada por los centros de Investigación, con el fin de maximizar el aprovechamiento de los recursos en cada centro para los avances de la ciencia en el país [1]. Propiciado por la emergencia sanitaria del año 2020, donde los estudiantes de distintos centros educativos dejaron de asistir de manera presencial, situación provocada por el COVID-19 [2, 3, 4]. Donde el rendimiento de los estudiantes disminuyo [5], dadas las limitaciones los espacios y expericneicas que ofrecían los centros educativos [6]. Particularmente, la enseñanza en temas de ingeniería robótica y de control viene acompañadas de un aprendizaje en base a experiencias [7], la cual se encuentra adaptándose a nuevos paradigmas por medio de diferentes herramientas, como lo son video conferencias, como simuladores, aplicaciones interactivas, y laboratorios virtuales [8, 9, 10]. Y es conocido que, para su desarrollo, presenta un interesante problema de control [11, 12, 13], requiriendo de cuidar la seguiridad del individuo y la del entorno.

Muchos de los robots virtuales son utilizados para enseñar la programación secuencial de los movimientos para cumplir una determinada tarea, sin embargo, los robots son un interesante sistema para ser controlado [11, 12, 13], que requieren de cuidar la seguiridad del individuo y el entorno, de una manera asequible.

Esto puede ser realizado con gemelos digitales en tiempo real en esquemas de simulación “Hardware in the Loop”, que ya ha sido utilizado en estaciones de enseñanza, como el control de motores [14].

El desarrollo de gemelos digitales requiere del modelado tanto matemático como tridimensional del sistema, en el caso de sistemas robóticos, por el modelado de las articulaciones y eslabones que conforman el robot, partiendo del modelado cinemático para la ubicación del robot en el espacio tridimensional, mientras, que el modelado dinámico desarrolla la relación de los movimientos del robot y las fuerzas que lo propician [15]. En la literatura, existen trabajos referentes al robot SCARA en tanto al

modelado cinemático [16], dinámico [17] y control del mismo [18, 19]. De manera que puede ser sugerido como modelo introductorio al diseño de controladores para sistemas robótico, sin embargo, es importante acotar la complejidad del sistema, siendo que la dinámica puede volverse bastante compleja [20] y al incorporarla, es deseable, un modelo simplificado que represente el estado dinámico suficiente para presentar un problema de control de sistemas robóticos y para fines didácticos.

Una manera novedosa de presentar estos gemelos digitales [21], ha sido por medio de la realidad aumentada, al permitir identificar una imagen o modelo 3D y posicionar un objeto virtual en dicha posiciòn, metodología que ya se ha incursionado en diferentes áreas de la enseñanza [22, 23, 24] y la educación.

A diferencia de otros trabajos existentes 1.1, este proyecto aborda el modelado cinemático, dinámico, para una simulación de una estación en pick and place que identifica las posiciones mediante marcadores de realidad aumentada.

Trabajo	Cinemática	Dinámica	HIL	3D	RA
Programación de robots industriales... [21]	x			x	x
Gemelo digital de un robot Fanuc... [25]	x			x	x
Concepción de la integración de ... [26]	x			x	x
Diseño, análisis y simulación de ... [27]	x	x			
Simuladores HIL para la educación ... [28]	x	x	x		
Este trabajo	x	x	x	x	x

Cuadro 1.1: Comparativa de trabajos similares

Siendo entonces los objetivos y alcances de este proyecto, el generar un gemelo digital de una estación pick and place, que sea controlable considerando el modelado cinemático y dinámico, basado en el robot DRS60L con el objetivo de incursionar a los usuarios en el estudio de la robótica, particularmente a los diferentes niveles de control que se encuentran en un sistema robotico, esto mediante el uso del esquema de simulación “Hardware In The Loop(HIL)”, la realidad aumentada, y el protocolo de comunicación.

Este trabajo pretende generar un entorno virtual de una estación pick and place para el apoyo de las capacitaciones en robótica, controlable, estable, y con un tiempo de respuesta aceptable, tanto computacionalmente como en visualización.

Este trabajo describe el desarrollo y modelado de un robot virtual SCARA, integra-

do en un entorno de “Pick and Place”, partiendo desde el modelado tridimensional, el modelado cinemático y dinámico (Considerando los efectos de la fricción), para incursionar un control simplificado, que permita interacturar con los marcadores de realidad aumentada para generar y controlar el gemelo virtual.

2 Marco teórico

Modelado matemático en 3D

Un objeto en un espacio tridimensional considera la coordenada $< x, y, z >$ y la rotación del objeto, expresadas mediante la matriz de transformación homogénea, denotada como H, una matrix cuadrada de 4 por 4, conformada de los siguientes elementos:

$$H = \begin{bmatrix} R & T \\ P & S \end{bmatrix} \tag{2.1}$$

- R, La matrix de rotación, una matrix cuadrada de 3 por 3.
- T, El vector de posición que corresponde $< x, y, z >^T$.
- P, El vector de perspectiva, para robótica siempre es $< 0, 0, 0 >$
- S, El factor de escala, para robótica siempre es 1.

La matriz de rotación R se compone de vectores unitarios en cada dirección que expresan la dirección del eje x, y, z, respectivamente, asociados mediante la regla de la mano derecha[29] 2.1.

$$R = \begin{bmatrix} x & y & z \end{bmatrix} = \begin{bmatrix} x_x & y_x & z_x \\ x_y & y_y & z_y \\ x_z & y_z & z_z \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} \tag{2.2}$$

De manera convencional en libros de robótica, se relaciona con los vectores $[\vec{n}, \vec{o}, \vec{a}$ [30]. Estas rotaciones, pueden ser calculadas mediante las siguientes expresiones [31]:

$$R_z(\alpha) = \begin{bmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$$R_y(\beta) = \begin{bmatrix} cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & cos(\beta) & 0 \end{bmatrix} \tag{2.4}$$

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\gamma) & sin(\gamma) \\ 0 & sin(\gamma) & cos(\gamma) \end{bmatrix} \tag{2.5}$$

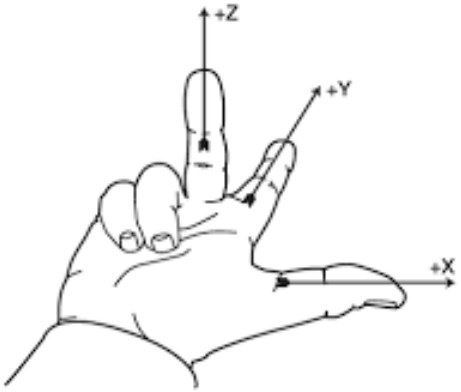

Figura 2.1: Representación de la Ley de la mano derecha

Para las orientaciones compuestas, simplemente basta con multiplicar las matrices de cada giro, por ejemplo, la rotación de Euler ZYZ [31]:

$$R_{zyz} = Rot_z(\theta)Rot_y(\theta)Rot_z(\theta) \tag{2.6}$$

Estas rotaciones se pueden expresar como:

$${}^0R_i = {}^0 R_1 *^1 R_2 * ... *^{i-1} R_i \tag{2.7}$$

La rotación R_i en base al sistema 0, es decir, como se vería la rotación R_i, si es medida con la orientación base.

Para formar la matriz homogénea desde una matriz de rotación, basta con sustituir R, por ejemplo, la rotación en el eje z, en forma de matriz homogénea:

$$T_{Rot_Z}(\theta) = \begin{bmatrix} Rot_Z(\theta) & T \\ P & S \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & 0 \\ sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

Mientras que la posición se define por medio de la matriz homogénea, acompañada de su rotación local:

$$Transl(x,y,z) = \begin{bmatrix} R_{11} & R_{12} & R_{13} & x \\ R_{21} & R_{22} & R_{23} & y \\ R_{31} & R_{32} & R_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & c_x \\ 0 & 1 & 0 & c_y \\ 0 & 0 & 1 & c_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

Donde esta posición se expresa en P, y si se busca crear un movimiento compuesto, también se puede aplicar la multiplicación de las matrices homogéneas en base al siguiente ejemplo:

$$T_2 = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

Se denota que:

$$T_1 * T_2 = \begin{bmatrix} 1 & 0 & 0 & c_x + d_x \\ 0 & 1 & 0 & c_y + d_y \\ 0 & 0 & 1 & c_z + d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

Cada matriz homogénea puede verse como la posición y orientación de un objeto, o bien, denotar un cambio relativo entre un sistema coordenado y otro, relación expresada en la robótica:

$${}^0T_i = {}^0\,T_1 *^1 T_2 * ... *^{i-1} T_i \tag{2.13}$$

Robot

La primera aparición del término Robot, esta relacionado a Karel Capek [32], en una obra conocida como “R.U.R’, que significa robota, entendido como trabajo, sin embargo, en el sentido forzado y obligatorio, siendo una máquina que realiza una serie de tareas por el mismo [33]. Estos sistemas desde su incursión en la industria en

General Motors [31], han ido creciendo ha diferentes cambios de aplicación que parten tanto en el sector industrial de la manufactura y la logística, así como en servicios, por ejemplo el sector médico. [34]. Visto desde una manera formal, la ISO 8373:2012 [29] define un robot industrial como un maquina controlada, y reprogramable multi-propósito, que puede encontrarse fijo o en un movimiento.

Modelado cinemático de robots

Los robots están conformados por eslabones y articulaciones, donde una articulación proporciona el movimiento relativo entre entre dos partes [35], como lo puede ser el codo de una persona, mientras que un eslabón, conecta las articulaciones, como puede ser el antebrazo a el codo de un ser humano como un codo, esto puede ser visto como un espaciador. Algunos ejemplos de articulación son 2.2:

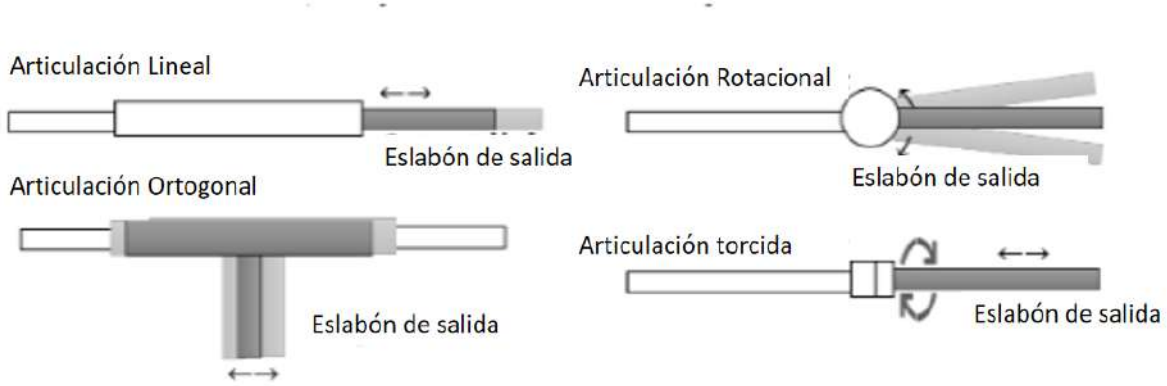

Figura 2.2: Ejemplos de articulaciones

Estas articulaciones 2.2 provocan el movimiento en el sistema robótico y a su vez determinan, los grados de libertad de un robot, en el momento que se le conecta un actuador, como puede ser un motor, o en su defecto, un pistón. Para ello se hace uso de la convención de Denavit-Haterberg, una forma estandarizada que resulta en una transformación homogénea que representa el tipo de articulación y el espaciado generado por el eslabón. Cabe mencionar que en este trabajo se usan los parámetros estándar para el análisis cinemático, y para el análisis dinámico, se usará la notación propuesta por Craig. Convención estándar D-H [31]:

$$^{n-1}T_n = T_{Rot_z}(\theta) * Transl(a, 0, d) * T_{Rot_x}(\alpha) \tag{2.14}$$

Donde se denotan 4 parámetros:

$$Dh_n(a, alpha, d, theta) = {}^{n-1}T_n = T_{Rot_z}(\theta_n) * Transl(a_n, 0, d_n) * T_{Rot_x}(\alpha_n) \quad (2.15)$$

Convención modificada D-H [15, 30]:

$${}^{j-1}\zeta_j = T_{Rot_x}(\alpha_{j-1}) * Transl(a_{j-1}, 0, d_j) * T_{Rot_z}(\theta_j) \quad (2.16)$$

El objeto del cambio propuesto por Craig brinda un acercamiento más simple al cálculode las aceleraciones y velocidades propagadas, en los algoritmos dinámicos, sin embargo, de manera usual se trabaja sobre la convención clásica para el análisis cinemático.

Según el parámetro que cambie respecto al tiempo representará el tipo de articulación que se modele bajo esta convención, en el caso de que ninguna varié, se considera un eslabón fijo, o de solo conexión.

- $\theta = \theta(t)$, es rotacional.
- $d = d(t)$, es prismático.

Estos parámetros usualmente se encuentran en una tabla para el modelado de los sistemas robóticos, y pueden ser abreviados por el tipo de articulación, por ejemplo “RP”, que sería un robot, con la primera articulación rotacional y la segunda prismática.

Trayectoria de Jerk Mínimo

Para el movimiento de sistemas robóticos, es necesario generar una trayectoria, que interpole el estado actual del robot, con el estado deseado, con el fin de respetar, condiciones de velocidad, aceleración deseados. La trayectoria de jerk mínimo, minimiza el efecto de la variación de la fuerza en los sistemas mecánicos [36], en su análogo con la expresión:

$$F = m * a \rightarrow F(t) = m \int j(t)dt \quad (2.17)$$

Estas trayectorias parten desde la integración a su paso por la aceleración, la velocidad y finalmente la posición. Sin embargo, su cálculo termina con una expresión polinomial de quinto grado[36]:

$$x = x_0 + (x_f - x_0) * (6 * \tau^5 - 15 * \tau^4 + 10\tau^3) \quad (2.18)$$

Donde τ se refiere al tiempo escalado [37], es decir:

$$\tau = \frac{t_a - t_0}{t_r} \tag{2.19}$$

Siendo t_a el tiempo actual, t_0 el tiempo inicial, y t_r el tiempo de ejecución del movimiento. Cabe mencionar que realmente se trata de una función a trozos. En caso de que $ta < t_0$ el valor de la trayectoria es x_0, y si $t_a - t_o > t_r$, entonces la trayectoria denota la posición x_f. Motivo de que las expresiones únicamente acotan ese espacio de trabajo, y cualquier consideración fuera de ese rango podría traer efectos extraños o no deseados. La trayectoria de Jerk Mínimo 2.3 se acota en un $t_r = 1$ y un $t_0 = 0$:

$$x(x_0, x_f, \tau) = x_0 + (x_f - x_0) * (6 * \tau^5 - 15 * \tau^4 + 10\tau^3) \tag{2.20a}$$

$$\dot{x}(x_0, x_f, \tau) = -30\,\tau^2\,(x_0 - x_f)\,(\tau - 1)^2 \tag{2.20b}$$

$$\ddot{x}(x_0, x_f, \tau) = -60\,\tau\,(x_0 - x_f)\left(2\,\tau^2 - 3\,\tau + 1\right) \tag{2.20c}$$

$$\dddot{x}(x_0, x_f, \tau) = -60\,(x_0 - x_f)\left(6\,\tau^2 - 6\,\tau + 1\right) \tag{2.20d}$$

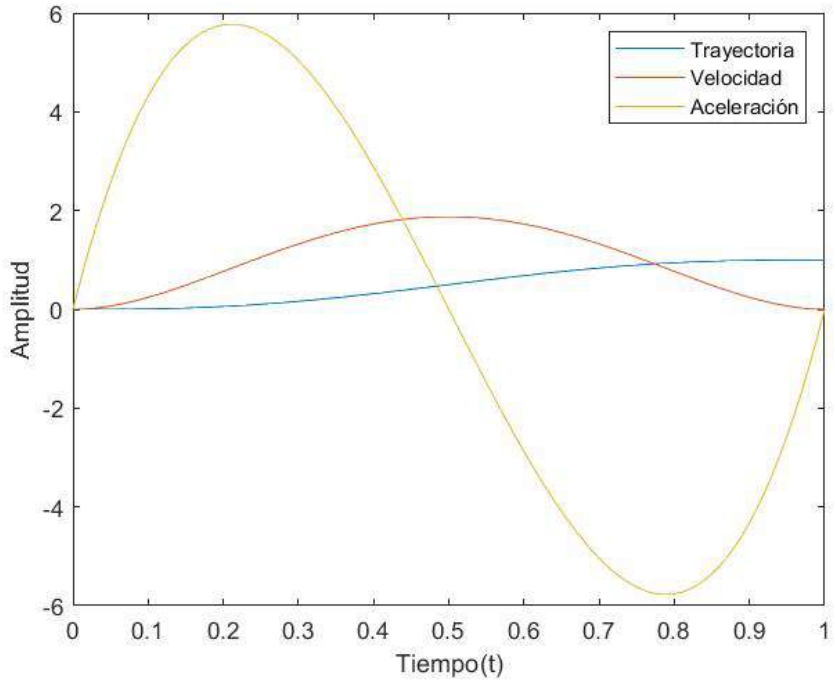

Figura 2.3: Trayectoria de Jerk Mínimo

Los máximos y mínimos de una función se pueden determinar cuando la derivada de la función de estudio es igualada a 0:

$$f'(x) == 0 \tag{2.21}$$

Al calcular las raíces de la función de Jerk, se denotan los puntos de máxima y mínima aceleración:

$$\dddot{x}(x_0, x_f, \tau) = 0 \rightarrow [0{,}2118, 0{,}7887] \tag{2.22}$$

Las raíces de la aceleración denotan los puntos de máxima y mínima velocidad:

$$\ddot{x}(x_0, x_f, \tau) = 0 \rightarrow [0, 0{,}5, 1] \tag{2.23}$$

De esta manera tanto la velocidad máxima es $1{,}8750$ mientras que la aceleración máxima es $5{,}7783$. Esto es en base a 1 segundo, y 1 unidad en la trayectoria.

Modelado dinámicos de robots

Los sistemas robotics además de conformarse de eslabones y articulaciones, los movimientos se realizan través de fuerzas o torques, según sea el tipo de articulación.

$$\tau = Robot(\ddot{q}, \dot{q}, dotq) \tag{2.24}$$

Si bien pueden intervenir más factores en los sistemas robóticos, en esencia parte, las condiciones del sistema robótico definen el torque que será necesario para realizar un movimiento especifico [15]. Formalmente, esto puede escribirse como [30, 31, 15]:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau \tag{2.25}$$

Donde se encuentra la matriz M, denominada la matriz de masa o de inercia, esta representa el análogo a la expresión:

$$F = m\ddot{q} \tag{2.26}$$

Mientras que para articulaciones rotacionales sería el análogo por medio de la ecuación de Euler, sin embargo, representa la inercia del sistema robótico en base al estado del robot. Si se despreciarán los efectos de $C(q, \dot{q})\dot{q}$ y de $G(q)$.

$$\tau = M(q)\ddot{q} \tag{2.27}$$

Cabe mencionar que τ se puede referir a fuerza si la articulación es prismática, o torque, si es rotacional. Mientras que el término de $C(q, \dot{q})\dot{q}$ se refiere a la carga debida a las velocidades de las articulaciones, compuesta de dos efectos, el efecto centrifugo y el de Coriolis [38]. El primero básicamente representa lo que le pasará a un objeto al

estar en la lavadora, que por efectos del movimiento rotacional 2.4, tiende a moverse la ropa a las paredes de la misma, debido a la fuerza centrifuga.

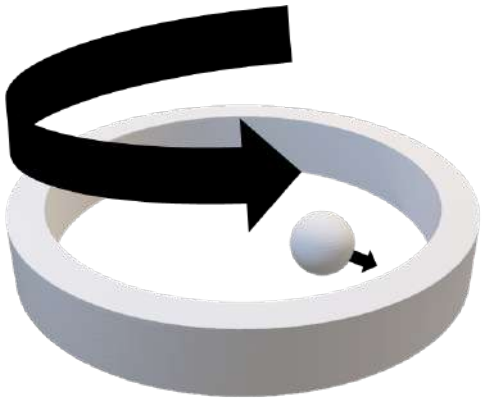

Figura 2.4: Analogía de la fuerza centrifuga

En el caso de la fuerza de Coriolis, es necesario pensar en que una pelota pequeña sobre una base para pasteles que se encuentra girando, si bien, la pelota, la ponemos en contacto con la base, y le damos un pequeño impulso, esta tomaría el camino recto, pero al estar girando la base, su movimiento no se describiría exactamente en forma de linea recta, si se observa desde una vista superior 2.5, a esa fuerza, que aparentemente hace que no se mueva en línea recta, se le conoce como la fuerza de Coriolis.

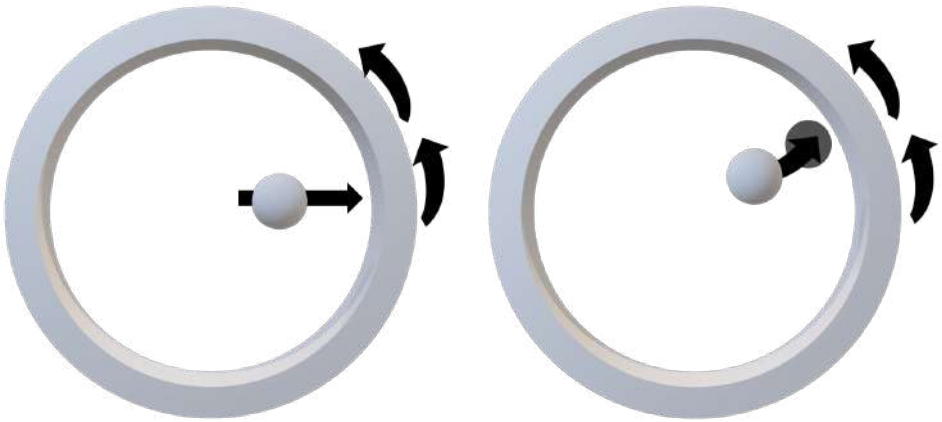

Figura 2.5: Analogía de la fuerza de Coriolis

Ahora bien el termino $G(q)$ es simplemente el efecto de la gravedad. Una manera para determinar la dinámica de un sistema es por medio de las ecuaciones de Euler-

Lagrange[39]:

$$L = K - V \tag{2.28}$$

$$\frac{d}{dt}\frac{dL}{d\dot{x}} - \frac{L}{dx} = \tau \tag{2.29}$$

Donde K, es la energía cinemática, es decir, la energía de cada movimiento que tiene cada movimiento:

$$K = \frac{1}{2}m\dot{\vec{x}}^2 \tag{2.30}$$

$$K = \frac{1}{2}I\dot{\vec{w}}^2 \tag{2.31}$$

Mientras que V, es la energía potencial, es decir, aquella por la posición meramente, la más usual es por la gravedad.

$$v = mgh \tag{2.32}$$

Es decir, que la energía potencial debido a la gravedad es, la masa por la velocidad y la altura desde el datum. Sin embargo, cada elemento que conforma el sistema robótico realiza aportaciones al total de la energía tanto potencial como cinemática. Esto puede plantear el Lagrangiano como [40]:

$$L(q, \dot{q}) = T(q, \dot{q}) - V(q) \tag{2.33}$$

En el espacio generalizado de un robot, se denota como:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i \tag{2.34}$$

Así mismo existe el planteamiento por medio de las ecuaciones de Newton-Euler, basado en el principio de causa reacción, resumido a la siguiente manera “Si en cada eslabón de un sistema robótico, tiene una velocidad y aceleración particular, debe de existir una fuerza o torque que lo permita”. El algoritmo de Newton-Euler, se compone de dos partes [15], en la primera únicamente se calcularán las velocidades y aceleraciones del sistema robótico actuales, y de manera recursiva, aquella fuerza de balance para propiciar dicho movimiento.

Para este caso ${}^i\boldsymbol{\omega}_i$ y ${}^i\dot{\boldsymbol{\omega}}_i$ representan la velocidad y aceleración angular i con respecto al sistema de referencia i-ésimo, mientras que $\dot{\boldsymbol{\theta}}_i$ and $\ddot{\boldsymbol{\theta}}_i$ representan la respectiva aceleración y velocidad de la articulación i. En el caso de las velocidades y aceleraciones lineales expresadas en base al sistema de referencia i-ésimo son representadas por ${}^i\dot{\boldsymbol{v}}_i$ and ${}^i\dot{\boldsymbol{v}}_{C_i}$, a su vez, la velocidad y aceleración lineal de la articulación se denotan por $\dot{\boldsymbol{d}}_i$ y $\ddot{\boldsymbol{d}}_i$. Mientras que la Pc_i representa el vector desde el inicio

del eslabón con dirección hacia el centro de masa, y I_i. Sin olvidar que ${}^{i}\boldsymbol{F}_i$ and ${}^{i}\boldsymbol{N}_i$ representan las fuerzas y torques de balance, mientras que ${}^{i}\boldsymbol{f}_i$ and ${}^{i}\boldsymbol{n}_i$ existentes en las articulaciones.

$$ {}^{i+1}\boldsymbol{\omega}_{i+1} = {}^{i+1}_{i}\boldsymbol{R}\,{}^{i}\boldsymbol{\omega}_i + \dot{\boldsymbol{\theta}}_{i+1}\,{}^{i+1}\hat{\boldsymbol{Z}}_{i+1} \tag{2.35} $$

$$ \begin{aligned} {}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} &= {}^{i+1}_{i}\boldsymbol{R}\,{}^{i}\dot{\boldsymbol{\omega}}_i + {}^{i+1}_{i}\boldsymbol{R}\,{}^{i}\boldsymbol{\omega}_i \times \dot{\boldsymbol{\theta}}_{i+1}\,{}^{i+1}\hat{\boldsymbol{Z}}_{i+1} \\ &\quad + \ddot{\boldsymbol{\theta}}_{i+1}\,{}^{i+1}\hat{\boldsymbol{Z}}_{i+1} \end{aligned} \tag{2.36} $$

$$ \begin{aligned} {}^{i+1}\dot{\boldsymbol{v}}_{i+1} &= {}^{i+1}_{i}\boldsymbol{R}({}^{i}\dot{\boldsymbol{\omega}}_i \times {}^{i}\boldsymbol{P}_{i+1} \\ &\quad + {}^{i}\boldsymbol{\omega}_i \times \left({}^{i}\boldsymbol{\omega}_i \times {}^{i}\boldsymbol{P}_{i+1}\right) + {}^{i}\dot{\boldsymbol{v}}_i) \end{aligned} \tag{2.37} $$

$$ \begin{aligned} {}^{i+1}\dot{\boldsymbol{v}}_{C_{i+1}} &= {}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} \times {}^{i+1}\boldsymbol{P}_{C_{i+1}} + {}^{i+1}\dot{\boldsymbol{v}}_{i+1} \\ &\quad + {}^{i+1}\boldsymbol{\omega}_{i+1} \times \left({}^{i+1}\boldsymbol{\omega}_{i+1} \times {}^{i+1}\boldsymbol{P}_{C_{i+1}}\right) \end{aligned} \tag{2.38} $$

$$ {}^{i+1}\boldsymbol{F}_{i+1} = m_{i+1}\,{}^{i+1}\dot{\boldsymbol{v}}_{C_{i+1}} \tag{2.39} $$

$$ \begin{aligned} {}^{i+1}\boldsymbol{N}_{i+1} &= {}^{C_{i+1}}\boldsymbol{I}_{i+1}\,{}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} \\ &\quad + {}^{i+1}\boldsymbol{\omega}_{i+1} \times {}^{C_{i+1}}\boldsymbol{I}_{i+1}\,{}^{i+1}\boldsymbol{\omega}_{i+1} \end{aligned} \tag{2.40} $$

$$ {}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} = {}^{i+1}_{i}\boldsymbol{R}\,{}^{i}\boldsymbol{\omega}_i \tag{2.41} $$

$$ \begin{aligned} {}^{i+1}\dot{\boldsymbol{v}}_{i+1} &= {}^{i+1}_{i}\boldsymbol{R}\Big({}^{i}\dot{\boldsymbol{\omega}}_i \times {}^{i}\boldsymbol{P}_{i+1} \\ &\quad + {}^{i}\boldsymbol{\omega}_i \times \left({}^{i}\boldsymbol{\omega}_i \times {}^{i}\boldsymbol{P}_{i+1}\right) + {}^{i}\dot{\boldsymbol{v}}_i\Big) \\ &\quad + 2{}^{i+1}\boldsymbol{\omega}_{i+1} \times \dot{\boldsymbol{d}}_{i+1}\,{}^{i+1}\hat{\boldsymbol{Z}}_{i+1} + \ddot{\boldsymbol{d}}_{i+1}\,{}^{i+1}\hat{\boldsymbol{Z}}_{i+1} \end{aligned} \tag{2.42} $$

Una vez que se han calculado las velocidades y aceleraciones propagadas, en cada una de las articulaciones, se permite calcular el torque, de equilibrio en la articulación denotado por las siguientes ecuaciones, cabe mencionar que una se refiere a las articulaciones prismaticas desde la ecuación (2.43) mediante (2.46) y la articulación rotacional como, desde (2.44) hasta (2.45)

$$ {}^{i}\boldsymbol{f}_i = {}^{i}_{i+1}\boldsymbol{R}\,{}^{i+1}\boldsymbol{f}_{i+1} + {}^{i}\boldsymbol{F}_i \tag{2.43} $$

$$
\begin{aligned}
{}^{i}\boldsymbol{n}_{i} &= {}^{i}\boldsymbol{N}_{i} + {}^{i}_{i+1}\boldsymbol{R}^{i+1}\boldsymbol{n}_{i+1} \\
&\quad + {}^{i}\boldsymbol{P}_{C_i} \times {}^{i}\boldsymbol{F}_{i} + {}^{i}\boldsymbol{P}_{i+1} \times {}^{i}_{i+1}\boldsymbol{R}^{i+1}\boldsymbol{f}_{i+1}
\end{aligned} \tag{2.44}
$$

$$
\tau_i = {}^{i}\boldsymbol{n}_{i}^{T}\,{}^{i}\hat{\boldsymbol{Z}}_{i} \tag{2.45}
$$

$$
\tau_i = {}^{i}\boldsymbol{f}_{i}^{T}\,{}^{i}\hat{\boldsymbol{Z}}_{i} \tag{2.46}
$$

Dinámica directa

En base a las ecuaciones anteriores y la descripción de las fuerzas que intervienen en un robot articulado, es posible plantear la pregunta qué movimiento desarrollaría el sistema robótico, si le aplico un torque en esta articulación, es decir, la entrada y salida del sistema, planteada de manera elegante por medio del algoritmo “Compositive Rigidbody Algorithm” o en su manera abreviada “CRA” o “CRBA” [41]. Al despejar la aceleración articular de la ecuación 2.25:

$$
\ddot{q} = M(q)^{-1}(\tau - C(q,\dot{q})\dot{q} - G(q)) \tag{2.47}
$$

Al proponer $\tau_0 = C(q,\dot{q})\dot{q} + G(q)$:

$$
\ddot{q} = M(q)^{-1}(\tau - \tau_0) \tag{2.48}
$$

Esto define la simulación de los sitemas robóticos, mediante el siguiente sistema [15]:

$$
\frac{d}{dt}\begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}(\tau - \tau_0) \end{bmatrix} \tag{2.49}
$$

Métodos númericos de integración

Existen métodos números enfocados a resolver el problema de integrar una ecuación diferencial:

$$
y' = f(t,y) \rightarrow y(t) \tag{2.50}
$$

Que toman el problema por medio de una ecuación recursiva, para computar la aproximación de la integral:

$$
y_n = y_{n-1} + f_i(t_0 + n * h, y_{n-1}) \tag{2.51}
$$

Donde f_i varia según el método que sea usado, mientras que h se refiere al paso de integración, a continuación se muestra una comparativa 2.1 de los diferentes métodos númericos[42]:

Método	Max. error
R-K quinto orden	9.67e-12
ODE45	9.46e-11
Fehlberg R-K45 de cuarto orden	9.48e-10
Fehlberg R-K45 de quinto orden	9.70e-10
Mersons	9.70e-9
Gills	9.87e-9
R-K34 de tercer orden	8.71e-9
R-K cuarto orden	8.71e-9
Euler	7.32e-3

Cuadro 2.1: Comparativa de métodos númericos

Para este trabajo se abordan únicamente 3, iniciando con el método de Euler, que define f_i como $f_i(t, y) = t_s * f'(t, y)$, dejando la expresión 2.51 como:

$$y_n = y_{n-1} + t_s * f(t_0 + n * t_s, y_{n-1}) \tag{2.52}$$

El método de Runge-Kutta, es un método basado en 4 aproximaciones, para el cálculo del siguiente valor, se basa en las siguientes expresiones [42]:

$$k_1 = hf(t_n, y_n) \tag{2.53}$$

$$k_2 = hf(t_n + t_s/2, y_n + k_1/2) \tag{2.54}$$

$$k_3 = hf(t_n + t_s/2, y_n + k_2/2) \tag{2.55}$$

$$k_4 = hf(t_n + t_s/2, y_n + k_3) \tag{2.56}$$

$$y_{n+1} = y_n + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6} \tag{2.57}$$

El método de Dormand-Prince, es un método adaptativo, el cual brinda un mejor desempeño que el algoritmo de Runge-Kutta, al posibilitar el cambio de h, según las características del sistema, se basa en las siguientes expresiones[43, 44]:

$$k_1 = hf\left(t_k, y_k\right) \tag{2.58}$$

$$k_2 = hf\left(t_k + \tfrac{1}{5}t_s, y_k + \tfrac{1}{5}k_1\right), \tag{2.59}$$

$$k_3 = hf\left(t_k + \tfrac{3}{10}t_s, y_k + \tfrac{3}{40}k_1 + \tfrac{9}{40}k_2\right), \tag{2.60}$$

$$k_4 = hf\left(t_k + \tfrac{4}{5}t_s, y_k + \tfrac{44}{45}k_1 - \tfrac{56}{15}k_2 + \tfrac{32}{9}k_3\right), \tag{2.61}$$

$$k_5 = hf\left(t_k + \tfrac{8}{9}t_s, y_k + \tfrac{19372}{6561}k_1 - \tfrac{25360}{2187}k_2 + \tfrac{64448}{6561}k_3 - \tfrac{212}{729}k_4\right), \tag{2.62}$$

$$k_6 = hf\left(t_k + t_s, y_k + \tfrac{9017}{3168}k_1 - \tfrac{355}{33}k_2 - \tfrac{46732}{5247}k_3 + \tfrac{49}{176}k_4 - \tfrac{5103}{18656}k_5\right), \tag{2.63}$$

$$k_7 = hf\left(t_k + t_s, y_k + \tfrac{35}{384}k_1 + \tfrac{500}{1113}k_3 + \tfrac{125}{192}k_4 - \tfrac{2187}{6784}k_5 + \tfrac{11}{84}k_6\right). \tag{2.64}$$

El siguiente paso y_n es calculado como:

$$y_{n+1} = y_n + \frac{35}{384}k_1 + \frac{500}{1113}k_3 + \frac{125}{192}k_4 - \frac{2187}{6784}k_5 + \frac{11}{84}k_6.$$

Así mismo, el paso de quinto orden es calculado como z_{k+1} :

$$z_{k+1} = y_k + \frac{5179}{57600}k_1 + \frac{7571}{16695}k_3 + \frac{393}{640}k_4 - \frac{92097}{339200}k_5 + \frac{187}{2100}k_6 + \frac{1}{40}k_7$$

Donde la diferencia $|z_{k+1} - y_{k+1}|$.

$$|z_{k+1} - y_{k+1}| = |\frac{71}{57600}k_1 - \frac{71}{16695}k_3 + \frac{71}{1920}k_4 - \frac{17253}{339200}k_5 + \frac{22}{525}k_6 - \frac{1}{40}k_7|$$

Será evaluada para el cálculo del tamaño del siguiente paso en base a:

$$s = \left(\frac{\epsilon h}{2\left|z_{k+1} - y_{k+1}\right|}\right)^{\frac{1}{5}} \tag{2.65}$$

$$h_{opt} = sh \tag{2.66}$$

Cabe mencionar que si el error $|z_{k+1} - y_{k+1}| > \epsilon$, donde ϵ es la tolerancia permitida, el paso se rechaza y se vuelve a realizar el cálculo del siguiente valor, considerando el ajuste $h = h_{opt}$

Sistemas de primer orden y el tiempo de estabilización

Un sistema con respuesta dinámica, es decir, ya sea un mecanismo, un circuito eléctrico, y otros fenomenos físicos pueden ser modelados por una ecuación de diferencial de primer orden [31]:

$$y' = ay - bx \tag{2.67}$$

Donde al realizar la tranformada de laplace:

$$sY(s) = aX(s) - bY(s) \tag{2.68}$$

Permite determinar una expresión, llamada función de transferencia, es decir, la relación que existe entre la señal de entrada y salida en el régimen de las frecuencias:

$$\frac{Y(s)}{X(s)} = \frac{b}{s+a} \tag{2.69}$$

Usualmente en el análisis de sistemas de control, se realiza el siguiente despeje:

$$\frac{Y(s)}{X(s)} = \frac{\frac{b}{a}}{\frac{1}{a}s+1} \tag{2.70}$$

Para definir τ como la constante de tiempo del sistema y k_p como la ganancia del sistema:

$$\frac{Y(s)}{X(s)} = \frac{k_p}{\tau s+1} \tag{2.71}$$

Esto quiere decir que al introducirle una señal escalón, tardará un tiempo en estabilizarse τ, y tendrá una ganancia k_p [45] como en la gráfica 2.6:

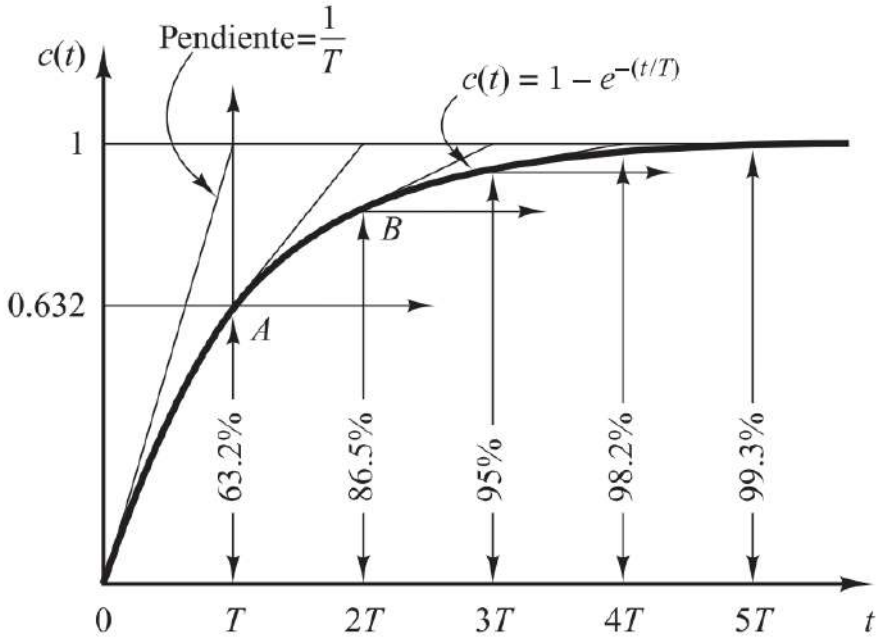

Figura 2.6: Relación de τ y la estabilización del sistema

Regularmente el tiempo de ascentamiento se denota por cuatro veces τ, aunque depende de la aplicación y la precisión buscada.

Control PID

El control PID, es una métodología de control basado en 3 acciónes [46], siendo P, una accion proporcial; I, una acción integral y D, es una acción derivativa [47] como se se muestra en 2.7.

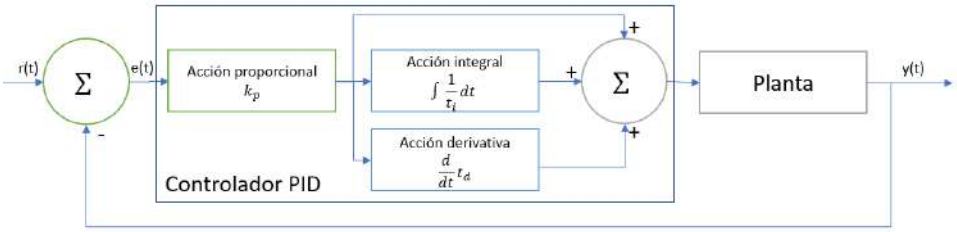

Figura 2.7: Diagrama de bloques del controlador PID

Siendo de interés para distitnos campos de la ingeniería [48], con el fin de llevar un sistema determinado, a un estado deseado, como puede ser la posición de un motor [49] o el control de sistemas robóticos articulados [50].

Control por IMC

Es un tipo de control cuyas siglas significan “Internal Model Control”, dependiente de un paramétro de sintonía λ que representa el tiempo deseado del sistema [51]. Recordando que el tiempo de un sistema regularmente se expresa por T o τ[45]. Parte del siguiente diagrama de bloques 2.8:

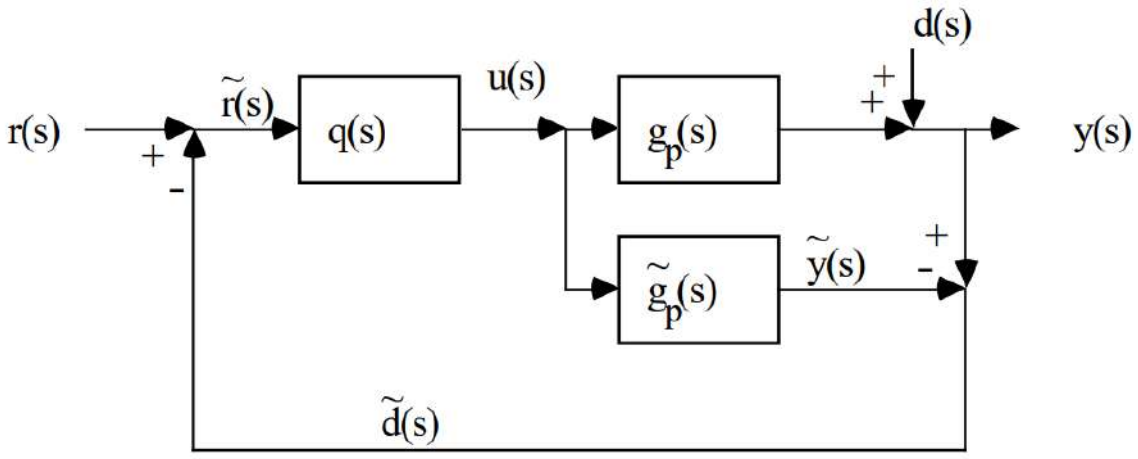

Figura 2.8: Diagrama del control IMC

El sistema que se desea controlar se denota por $g_p(s)$, mientras que $\tilde{g_p}(s)$ es el modelo teorizado, y $q(s)$ el control propuesto, si se agrupa el término $q(s)$, es decir 2.9:

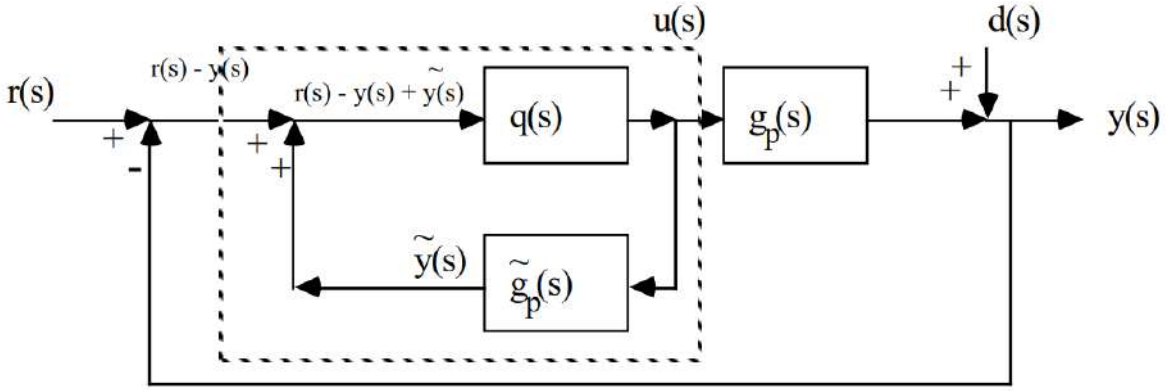

Figura 2.9: Diagrama del control IMC, agrupamiento

Donde se conoce que pueden ser agrupados usando la siguiente expresión, donde g_c es el control en forma IMC:

$$g_c(s) = \frac{q(s)}{1 - \tilde{g_p}(s)q(s)} \tag{2.72}$$

Dejando expresado el lazo de control de la siguiente manera 2.10:

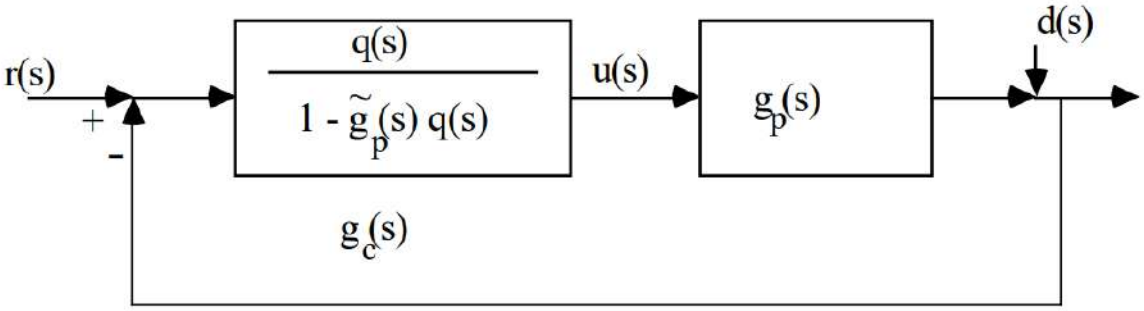

Figura 2.10: Diagrama del control IMC, reducido

Para encontrar los términos faltantes, se sustituye la formula 2.72:

$$g_c(s) = \frac{\frac{q(s)}{1-\tilde{g_p}(s)q(s)} * g(s)}{1 + \frac{q(s)}{1-\tilde{g_p}(s)q(s)} * g_p(s)} \tag{2.73}$$

$$g_c(s) = \frac{\frac{q(s)}{1-\tilde{g_p}(s)q(s)} * g_p(s)}{\frac{1-\tilde{g_p}(s)q(s)}{1+\tilde{g_p}(s)q(s)} - \frac{q(s)}{1-\tilde{g_p}(s)q(s)} * g_p(s)} \tag{2.74}$$

$$g_c(s) = \frac{q(s) * g_p(s)}{1 - \tilde{g_p}(s)q(s) + q(s) * g_p(s)} \tag{2.75}$$

Si $-\tilde{g_p}(s)q(s) + q(s) * g_p(s) = 0$, entonces el sistema fue modelado de manera precisa, lo que permite definir a g_c como:

$$g_c(s) = q(s) * g_p(s) \tag{2.76}$$

Si se busca que el sistema se comporte como un sistema de primer orden, con el paramétro λ de ajuste:

$$q_c(s) = \frac{1}{\lambda s + 1} \tag{2.77}$$

$$\frac{1}{\lambda s + 1} = q(s) * g_p(s) \tag{2.78}$$

Entonces:

$$q(s) = \frac{1}{(\lambda s + 1) * g_p(s)} \tag{2.79}$$

Esta metodología, permite conocer los valores de sintonía y requieridos de un controlador PID, en base a λ y el modelo de la planta [51], a continuación se muestran algunas 2.2:

g_p	g_c	k_p	t_i	t_d	t_f
$\frac{kp}{\tau_p s+1}$	$\frac{1}{\lambda s+1}$	$\frac{\tau_p}{k_p \lambda}$	t_p	–	–
$\frac{kp}{s}$	$\frac{1}{\lambda s+1}$	$\frac{1}{k_p \lambda}$	–	–	–
$\frac{kp}{s(\tau_p s+1)}$	$\frac{1}{\lambda s+1}$	$\frac{1}{k_p \lambda}$	–	t_p	–

Cuadro 2.2: Formulas para el Internal Model Control

Formando un controlador de la siguiente forma [47]:

$$g_c = k_p * (1 + \frac{1}{t_i s} + t_d s) \tag{2.80}$$

Cuando la tabla denota “-”, se refiere a que ese parámetro no es requerido, formando controladores con solo la parte proporcional derivativa, solo proporcional, entre otras combinaciones. Esta metodología, el término de lambda, permite establecer la robustez que sea requerida, el rechazo de perturbaciones, el esfuerzo de los actuadores, o la consideración particular del problema.

Robot SCARA

SCARA, es el acrónimo de “Selective Compliant Arm for Robotic Assembly” [52], este tipo de robots manipuladores son usados en aplicaciones de movimiento de objetos, o ensamblajes debido a su velocidad, sin embargo, carecen de potencia para

realizar trabajos pesados 2.11. Es común encontrarlos en dos versiones una “RRPR” y “RRP”.

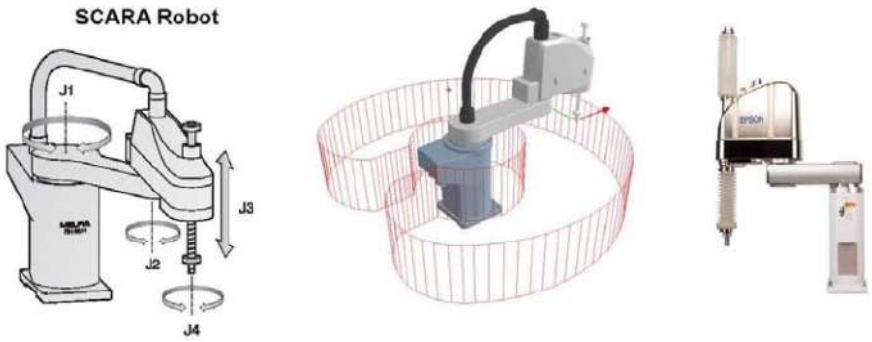

Figura 2.11: Diagrama general de un robot SCARA

Espacio de configuraciones de un robot

Cada tipo de robot crea un espacio llamado espacio de configuración 2.12, un espacio conformado de cada punto es alcanzable por el robot y aceptado por el entorno, en la mayoría de los casos, por no entrar en colisión [53].

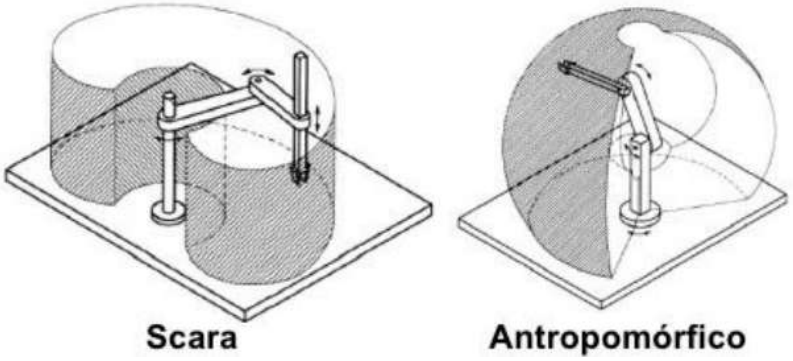

Figura 2.12: Ejemplos de espacios de configuración

Siendo una manera de acotar, las configuraciones alcanzables del sistema robótico por límites físicos, ya sea por la construcción, o bien, el espacio donde se encuentra.

Gemelo digital

Se refiere a la simulación de un sistema físico, que permite exponerlo a diferentes condiciones, y situaciones para observar su comportamiento ante ellos. El Dr. Michael Grieves y John Vickers los dividen en dos categorías [54]:

- Para prototipado, compuesto de toda la información del modelo físico, que incluyen los requerimientos, un modelo 3D, una lista de materiales, y las conexiones entre los diferentes elementos.
- Como instancia, de manera que este gemelo digital, se encuentre conectado al dispositivo físico durante su vida, ya sea para reportar, monitorear, diagnosticar, controlar y optimizar el sistema físico

Esta técnica 2.13 permite minimizar los costos de prototipado, y se ha utilizado en diferentes campos de la robótica Particularmente tiene el fin de evaluar la propuesta de un sistema en un determinado ambiente, o su desempeño ante condiciones adversas, minimizando los costos de prototipado, siendo llevados a la simulación en diferentes campos de la robótica [21, 55, 56].

Figura 2.13: Representación de un gemelo digital

Cabe mencionar que este término no debe de ser confundido, con el término de simulación “Hardware in the Loop”, siendo que mientras un gemelo digital, representa es una simulación computacional del sistema, una simulación HIL, es una manera de probar los algoritmos de control de un controlador, siguiendo el siguiente esquema 2.14

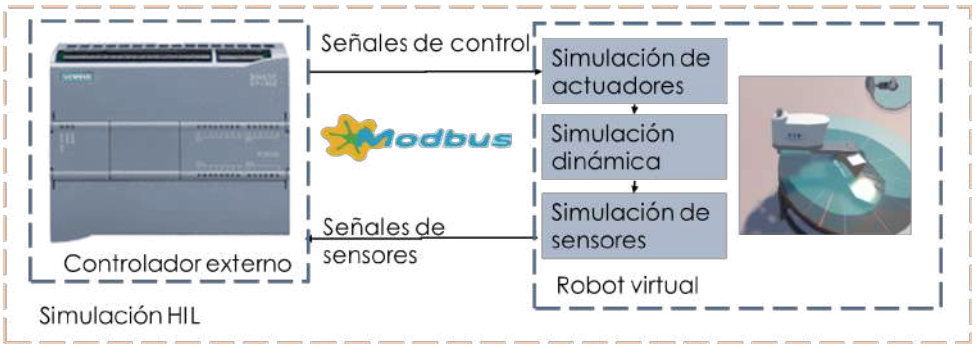

Figura 2.14: Esquema de simulación HIL para el sistema robótico

Realidad aumentada

La realidad aumentada ha sido un concepto que ha ganado popularidad a lo largo de los años, ha entrado cada vez en nuevos campos, como lo es en la robótica, a través de la presentación de estaciones de trabajo, y nuevas experiencias. Basándose en la identificación de un marcador 2D, o 3D, en el cual es proyectada información relevante 2.15.

Figura 2.15: Ejemplo de una aplicación usando realidad aumentada

Las implementaciones en realidad aumentada dependen de un dispositivo que tengan una camara para buscar los patrones de intéres, denotados, marcadores, estos pueden ser códigos QR, imagenes, y otras alternativas. Su clasificación se encuentra denotada como [57]:

- Nivel 0 (Vinculada al mundo real). Aplicaciones que vincular el mundo físico, como los códigos de barras.
- Nivel 1 (RA con marcadores). Aplicaciones que usan marcadores, ya sean a blanco y negro, dibujos esquematicos, para la identificación de patrones 2D y realizar la superposición.
- Nivel 2 (RA sin marcadores). Aplicaciones que remplazan el uso de marcadores con el GPS para determinar la la posición y orientación de los puntos de interes y realizar la superposición.
- Nivel 3 (Visión aumentada). Representado por dispositivos como los ”Google Glass“, lentes de contacto de alta tecnología, inmersivos, y de experiencias personalizadas.

Modbus

El protocolo de comunicación Modbus, es un protoloco de mensajería cumpliendo con la 7 del modelo OSI [58]. En este protocolo, se define un cliente y un servidor [59], definiendo servidor, al que tiene los registros físicamente, mientras que el cliente los solicita y modifica 2.16.

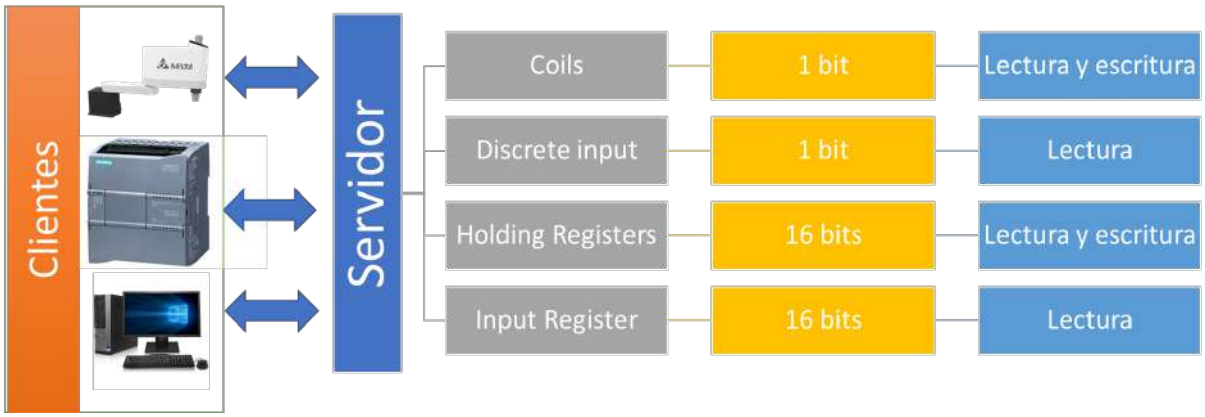

Figura 2.16: Interacciones de los clientes y los tipos de memorias en el protocolo Modbus

3 Metodología

La metodología de este trabajo se apoya en una modalidad en V **??**, se segmentado para este proyecto como 3.1.

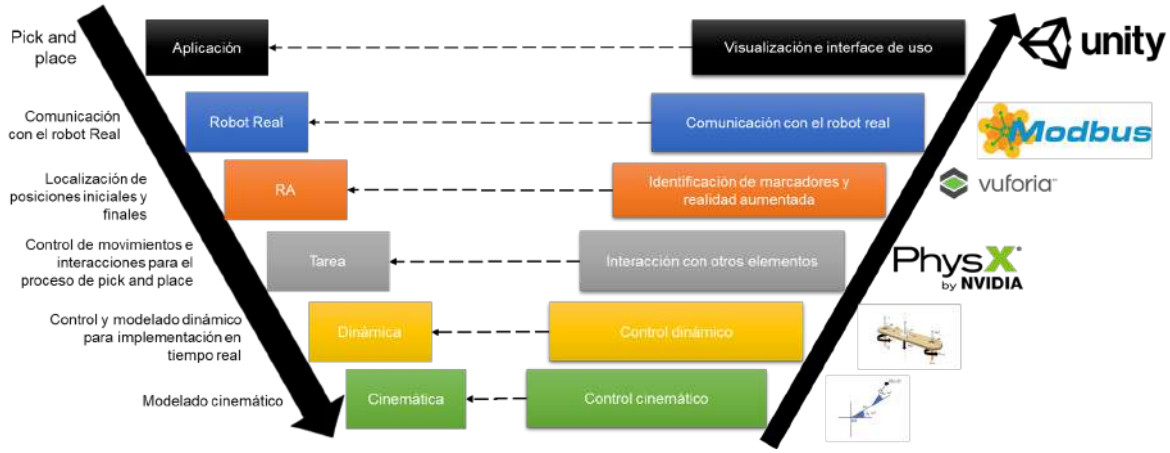

Figura 3.1: Diagrama en V del proyecto

Partiendo del modelado cinemático, y su correspondiente solución, que permite avanzar en el modelado dinámico, su simplificación, y las condiciones impuestas por el motor de fisica que se utilizará, en este caso PhysX de Nvidia. Esto con la finalidad de plantear la estrategia de control que pueda aplicarse al entorno generado en PhysX, y que ha su vez, permita integrar las condiciones existentes en una estación pick and place, es decir, el tomado de las piezas en el entorno simulado, y a su vez, permitir simular que pasa cuando no se realiza correctamente el flujo de control. El intercambio de datos es gobernado por medio del procolo Modbus para integrar la capacidad de transferir el control del robot virtual a un dispositivo externo como un PLC (Simulación HIL), o así mismo, permitir el control de un robot real, en base, a la secuencia generada por medio del teaching mediante el uso de realidad aumentada (Digital Tween).

Para la elaboración de este trabajo se utilizaron los siguientes programas:

- Delta Robot DS
- Unity 2020 LTS

- SolidWorks 2019
- Blender 2.83 LTS
- Matlab 2021
- Robotics Toolbox de Matlab
- Laptop Ryzen 5, 8GB RAM
- Labview y NI Modbus
- Tia portal 7

Modelado del Robot SCARA

Para el modelado del robot SCARA, es necesario abordar, de manera inicial el modelado CAD para posteriormente involucrar las restricciones impuestas por el equipo físico, y las impuestas por el motor de física de Unity 3D.

3.1.1. Modelado 3D

El diseño del Robot SCARA DRS60L 3.2, se realizó en base a las especificaciones encontradas en la hoja de datos, respetando las diferentes distancias entre cada eslabón e involucrando un elemento, que permitiría diseñar el modelo dinámico propuesto, y así mismo vincular las posiciones relativas de una manera más simple.

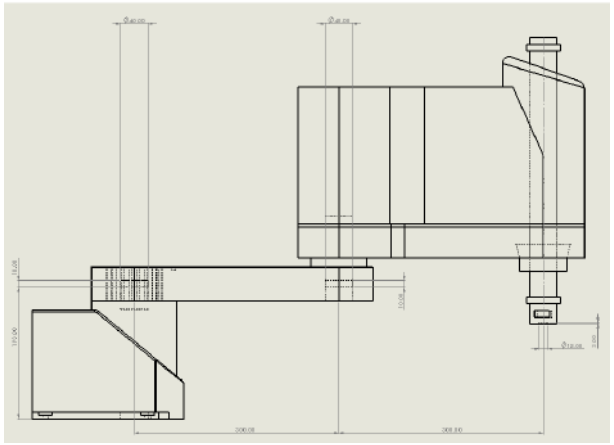

Figura 3.2: Plano del robot SCARA, generado del diseño 3D

Para establecer las posiciones de las masas del modelo simplificado, y así mismo denotar la ubicación de las articulaciones se hace en base a discos, que estarán ocultos, sin embargo, sirven de puntos de pivote para transferir el movimiento al eslabón correspondiente:

- N1, 40mm x 10 mm.
- N2, 40mm x 10 mm.
- N3, 12mm x 03 mm

Así mismo para las pruebas, se construyen 2 modelos, iniciando con el modelo que únicamente se compone del modelo 3D del robot, el cual se usará posteriormente para la conexión y vinculo con el marcador de realidad aumentada, mientras que otro se desarrollará para pruebas de control, tanto a nivel cinemático como a nivel dinámico.

Diseño del Robot SCARA DRS60L 3.3.

Figura 3.3: Modelo 3D del Robot SCARA Delta DRS60L

Diseño del Robot SCARA DRS60L junto con la mesa polar 3.4

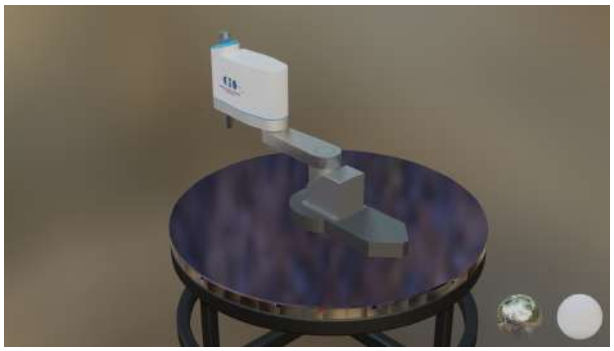

Figura 3.4: Modelo 3D del robot SCARA en la mesa polar

Denotando la vista superior de la mesa polar 3.5:

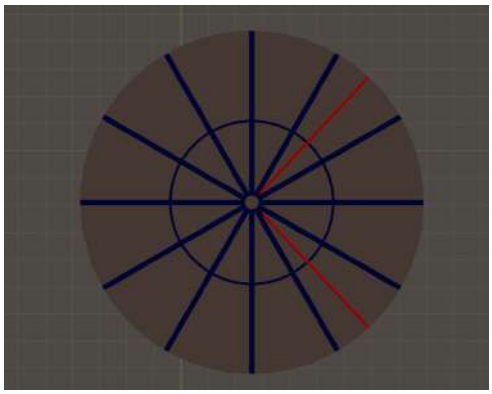

Figura 3.5: Diseño de la mesa polar

3.1.2. Modelado de la cinemática directa

El SCARA seleccionado para este trabajo parte de los siguientes parámetros DH, denotados en 3.1, parten del robot DRS60L 3.6.

Link	α_i	a_i	d_i	Variable
L1	0	0.3	0.139	Theta
L2	180	0.300	0	Theta
L3	0	0	d	d
L4	0	0	0	Theta

Cuadro 3.1: Tabla de parámetros DH

El resultado de la cinemática directa del robot SCARA, se puede denotar por la multiplicación de las matrices de transformación homogéneas (2.15)

$$^{0}T_4 = DH(0{,}3, 0, 0{,}139, \theta_1) * DH(180\,\deg, 0{,}3, 0, \theta_2) * DH(0, 0, \theta_3, 0) * DH(0, 0, 0, \theta_4) \quad (3.1)$$

Donde la posición final del robot SCARA puede ser denotada mediante las siguientes ecuaciones:

$$x = 0{,}300 * cos(\theta_1) + 0{,}300 * cos(\theta_1 + \theta_2) \quad (3.2a)$$

$$y = 0{,}300 * sin(\theta_1) + 0{,}300 * sin(\theta_1 + \theta_2) \quad (3.2b)$$

$$z = 0{,}139 - \theta_3 \quad (3.2c)$$

$$\theta_f = \theta_1 + \theta_2 - \theta_4 \quad (3.2d)$$

O de manera generalizada como:

$$x = l_2 * cos(\theta_1) + l_3 * cos(\theta_1 + \theta_2) \tag{3.3a}$$

$$y = l_2 * sin(\theta_1) + l_3 * sin(\theta_1 + \theta_2) \tag{3.3b}$$

$$z = l_1 - \theta_3 \tag{3.3c}$$

$$\theta_f = \theta_1 + \theta_2 - \theta_4 \tag{3.3d}$$

Los valores de distancia, están denotadas en metros, para evitar problemas con el planteamiento dinámico.

3.1.3. Modelado del espacio de configuraciones

El espacio de configuraciones, consiste en el conjunto de todas las posiciones y orientaciones que puede tomar el robot. En este caso se consideran los límites encontrados en la hoja de datos del Robot SCARA Delta DRS60L [60] 3.6.

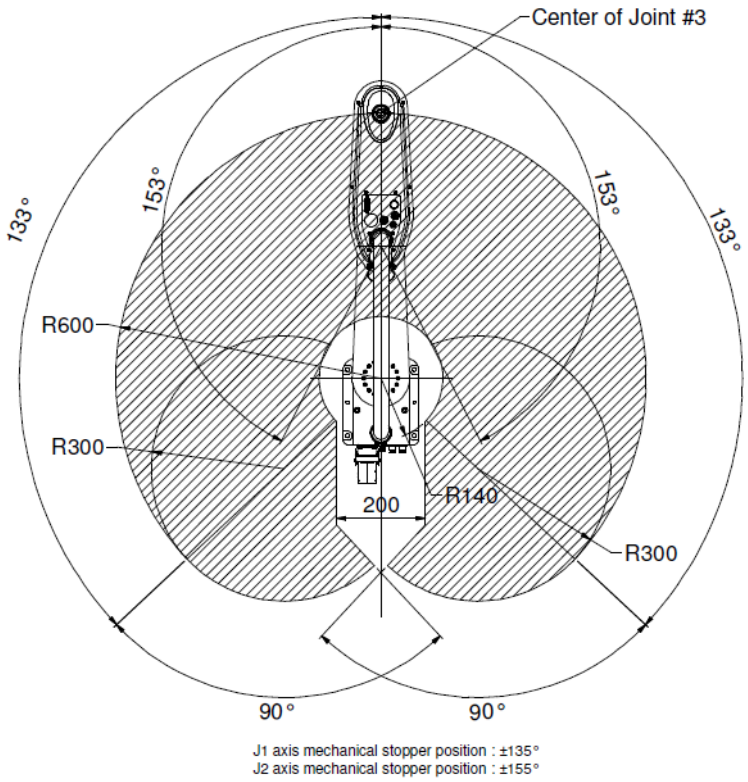

Figura 3.6: Plano del espacio del trabajo del Robot SCARA DRS60L

Siendo la primera restricción por los valores máximos y mínimos de los actuadores:

$$\vec{q_c} \in \begin{bmatrix} -133\,\text{deg}, 133\,\text{deg} \\ -153\,\text{deg}, 153\,\text{deg} \\ 0, 200 \\ -360\,\text{deg}, 360\,\text{deg} \end{bmatrix} \tag{3.4}$$

Mientras que una segunda restricción, parte del espacio cartesiano, este caso la relación x-y

$$(x_c, y_c) : 0{,}140 <= |x, y| <= 0{,}600 \tag{3.5}$$

En el caso de z, la restricción se define como:

$$z_c \in [-0{,}061, 0{,}139] \tag{3.6}$$

El valor de $-0{,}061$ parte de $0{,}139 - 0{,}2$. Permitendo, de manera actual, gráficar el espacio de configuraciones 3.7 en base a la cinemática directa (3.2).

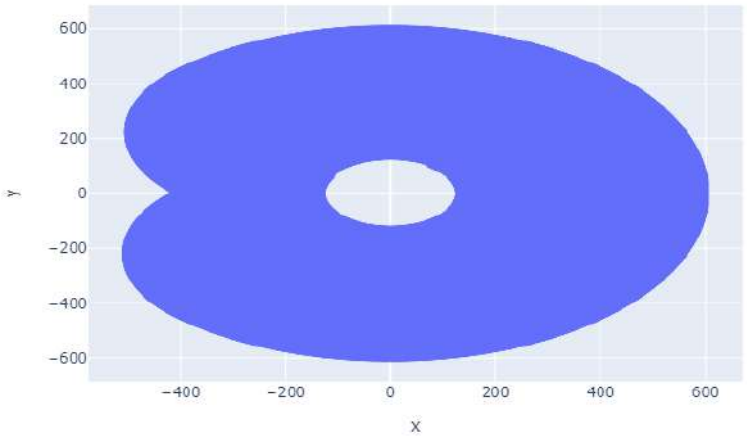

Figura 3.7: Primera aproximación del espacio de configuraciones

En el caso de la parte trasera, se pueden excluir aquellos puntos pertenecientes a un rectángulo, definido como:

$$Rect = (x, y) \in (x < 0, |y| > 100) \tag{3.7}$$

Es decir:

$$(x_c, y_c) : (0{,}140 <= |x, y| <= 0{,}600) - (x < 0, |y| > 100) \tag{3.8}$$

Quedando la segunda aproximación del espacio de trabajo 3.8

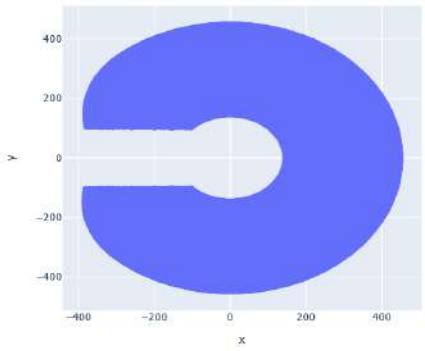

Figura 3.8: Segunda aproximación del espacio de trabajo

La parte trasera se denota por el plano 3.9.

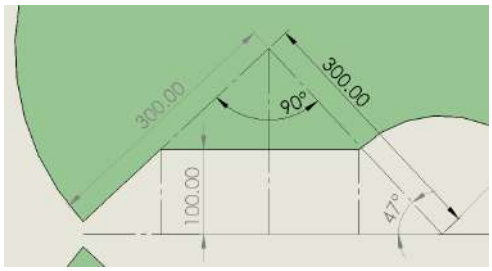

Figura 3.9: Plano para la aproximación del espacio de trabajo

Permitiendo plantear el triángulo 3.10

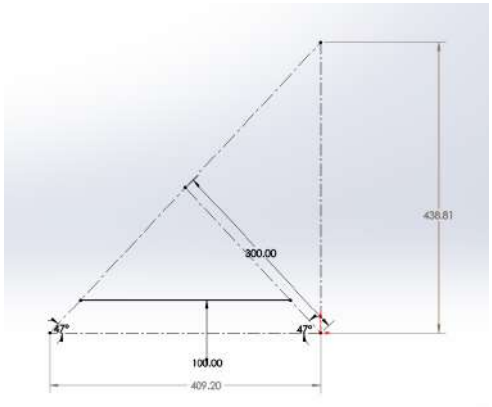

Figura 3.10: Triángulo para la aproximación del espacio de trabajo

La hipotenusa del triángulo, se puede plantear por una ecuación de la recta:

$$tan(\theta) = \frac{CO}{CA} \tag{3.9a}$$

$$x_1 = 100/tan(47\,\mathrm{deg}) \tag{3.9b}$$

$$cos(\theta) = \frac{CA}{h} x_2 = h * cos(47\,\mathrm{deg}) \tag{3.9c}$$

$$x_a = x_1 + x_2 \simeq 298 \tag{3.9d}$$

Donde x_1 se refiere al punto inicial de la hipotenusa del lado izquierdo, mientras que x_2 se refiere al punto final de la hipotenusa.

$$y = 438/409{,}2 * x + 438 \tag{3.10a}$$

Esto concluye que los puntos válidos del espacio de configuraciones del robot 3.11, serán válidos, si además de estar acotados por (3.4), en el calculo de la cinemática directa, se requiere que los valores de x,y generados esten dentro dentro de la restricción (3.8), y eliminando aquellos que se encuentran en la zona (3.7). Sin embargo, los últimos calculos, permiten incluir la zona trianguar, en la izquierda, si $x < 0$ y $y < 100$, donde aquellos puntos válidos en el espacio de trabajo serán aquellos que esten sobre la recta calculada (3.10), es decir, $(x, y) \in (-438 > 438/409{,}2 * x - y)$

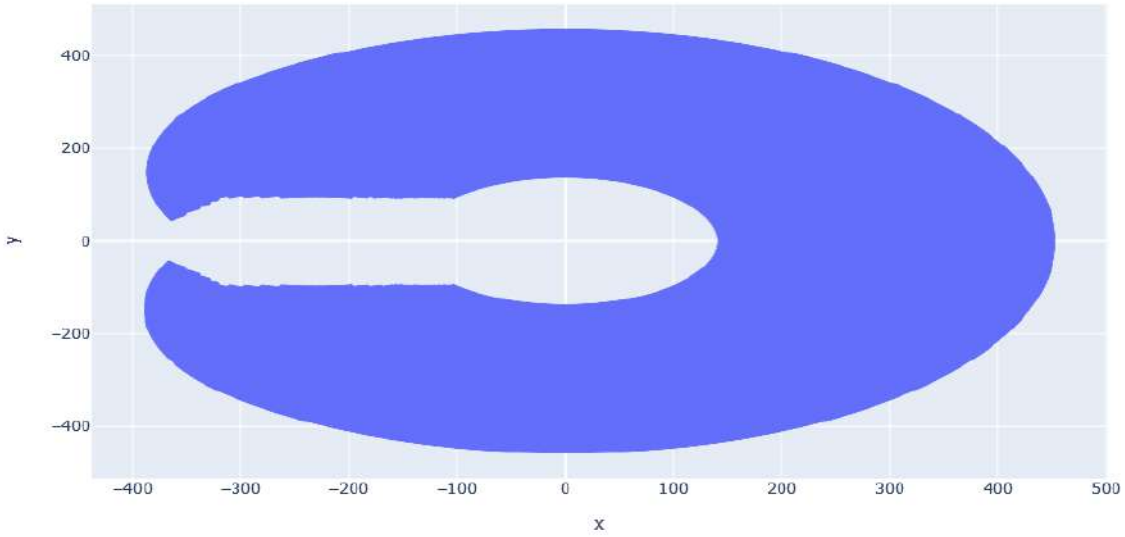

Figura 3.11: Tercera aproximación del espacio de trabajo

Un ejemplo de la implementación de estas restricciones del espacio de trabajo, puede encontrarse en el anexo A.

3.1.4. Cinemática inversa del Robot SCARA por el método geométrico

En los sistemas robóticos, el cálculo de la cinemática inversa, radica en encontrar los valores de posición de cada articulación, con el fin de llevar el efector final a una posición deseada. De manera inicial para resolver la cinemática inversa de un robot SCARA [61] parte del diagrama 3.12:

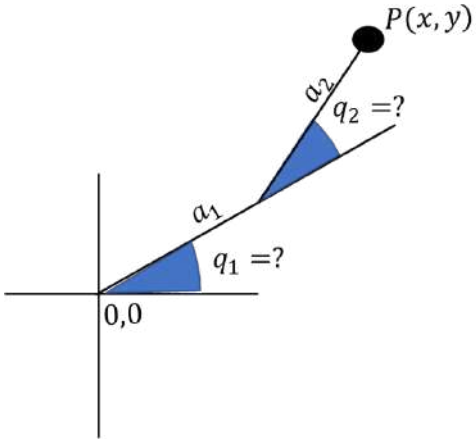

Figura 3.12: Planteamiento geométrico de la cinemática inversa

Siendo posible denotar el triángulo 3.13

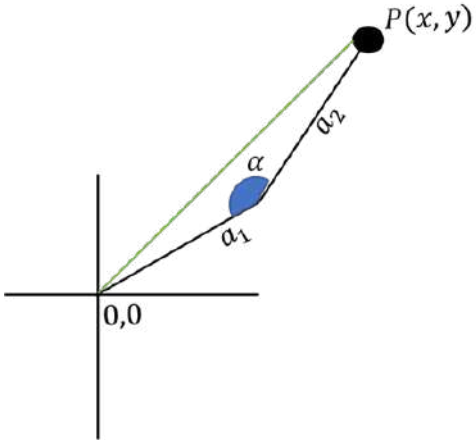

Figura 3.13: Cálculo de α

Usando la fórmula de cosenos:

$$r^2 = a_1^2 + a_2^2 - 2 * a_1 * a_2 * cos(\alpha) \tag{3.11}$$

Permite resolver el parámetro α:

$$\alpha = acos(\frac{a_1^2 + a_2^2 - r^2}{2 * a_1 * a_2}) \tag{3.12}$$

Con el valor de α se puede resolver el valor de q_2 3.14

$$q_2 = \pi - \alpha \tag{3.13}$$

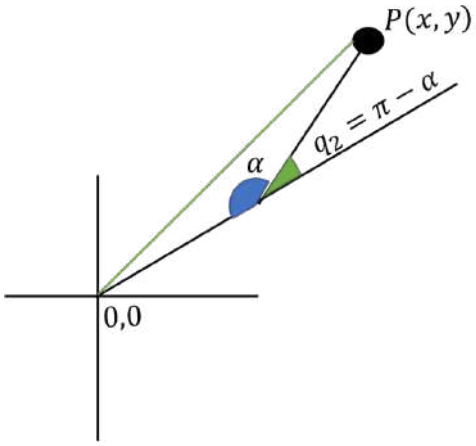

Figura 3.14: Solución de q_2

Mientras que para definir el valor de γ se plantea en base al diagrama 3.15:

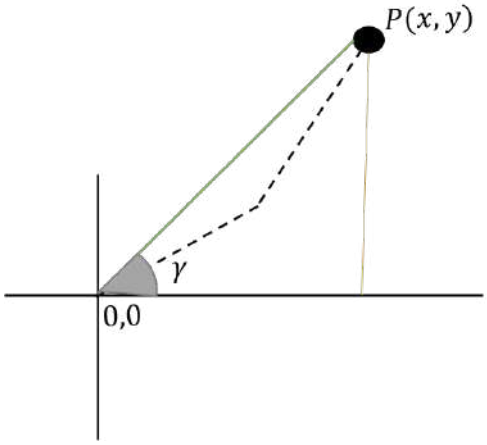

Figura 3.15: Solución de γ

$$\gamma = atan2(y, x) \tag{3.14a}$$

(3.14b)

De esta manera, se puede expresar la relación con los ángulos faltantes 3.16

$$a_2^2 = a_1^2 + r^2 - 2a_1 r * cos(\beta) \tag{3.15a}$$

$$\beta = acos((a_2^2 - a_1^2 - r^2)/(2a_1 r)) \tag{3.15b}$$

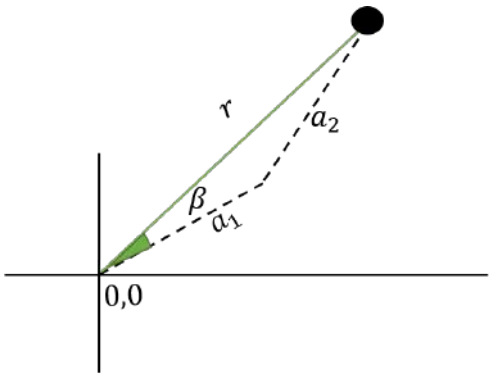

Figura 3.16: Ubicación del ángulo β

Dejando la solución de q_1 como:

$$q_1 = \gamma - \beta \tag{3.16}$$

Sin embargo, si se toma el vector entre el origen, y el punto final como una linea de simetría, los vectores, se puede denotar la segunda solución del robot SCARA 3.17

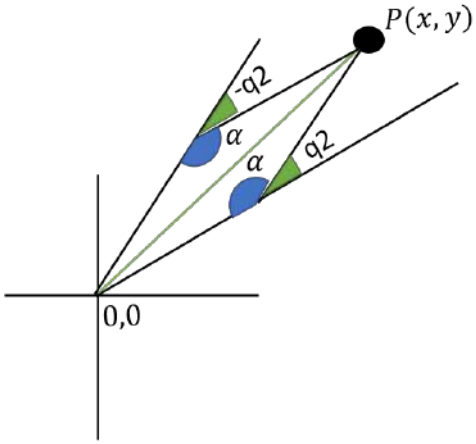

Figura 3.17: Relación entre las dos soluciones geométricas

$$q_2 = \alpha - \pi \tag{3.17a}$$

$$q_1 = \gamma + \beta \tag{3.17b}$$

De manera simple para trabajar el sector inferior de las soluciones cuando el punto solicitado $y < 0$, se puede trabajar con el valor absoluto:

$$y = |y_{requested}| \tag{3.18}$$

Por consiguiente los valores obtenidos por esta condición, serán multiplicados por -1. Mientras que la solución de la tercera y cuarta articulación parten únicamente del despeje de las ecuaciones de la cinemática directa 3.2:

$$\theta_3 = z - l3 \tag{3.19a}$$

$$\theta_4 = \theta_f - \theta_1 - \theta_2 \tag{3.19b}$$

3.1.5. Velocidades en el Robot SCARA

Las velocidades en los sistemas robóticos, pueden ser descritos de una forma compacta como[30]:

$$\dot{\vec{x}} = J(\vec{\theta})\dot{\vec{\theta}} \tag{3.20}$$

Donde $J(\theta)$ corresponde al Jacobiano, una matriz especial constituida de derivadas parciales de la siguiente manera:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial u_1}{\partial x_1} & \cdots & \frac{\partial u_1}{\partial x_j} \\ \cdots & \cdots & \cdots \\ \frac{\partial u_n}{\partial x_j} & \cdots & \frac{\partial u_j}{\partial x_j} \end{bmatrix} \tag{3.21}$$

Denotando que el Jacobiano de la posición final del robot SCARA puede ser calculado como:

$$\mathbf{J} = \nabla \begin{bmatrix} l_2 * cos(\theta_1) + l_3 * cos(\theta_1 + \theta_2) \\ l_2 * sin(\theta_1) + l_3 * sin(\theta_1 + \theta_2) \\ l_1 - \theta_3 \\ \theta_1 + \theta_2 - \theta_4 \end{bmatrix} \tag{3.22}$$

Resultando en:

$$J(\theta_1, \theta_2, \theta_3, \theta_4) = \begin{bmatrix} -l_3 \sin(\theta_1 + \theta_2) - l_2 \sin(\theta_1) & -l_3 \sin(\theta_1 + \theta_2) & 0 & 0 \\ l_3 \cos(\theta_1 + \theta_2) + l_2 \cos(\theta_1) & l_3 \cos(\theta_1 + \theta_2) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 1 & 0 & -1 \end{bmatrix} \tag{3.23}$$

Al unirlo con la expresión 3.20 para expresar la velocidad de la siguiente manera:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} -l_3 \sin(\theta_1 + \theta_2) - l_2 \sin(\theta_1) & -l_3 \sin(\theta_1 + \theta_2) & 0 & 0 \\ l_3 \cos(\theta_1 + \theta_2) + l_2 \cos(\theta_1) & l_3 \cos(\theta_1 + \theta_2) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \\ \dot{\theta_3} \\ \dot{\theta_4} \end{bmatrix} \tag{3.24}$$

La velocidad cartesiana resulta en:

$$Velocidad'_{cartesiana} = \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} \tag{3.25}$$

3.1.6. Cinemática inversa por un método numérico

Una vez que se tiene definido el Jacobiano del Robot SCARA, es posible determinar la posición como una función, denotada en $\mathbb{R}^4 \Rightarrow \mathbb{R}^4$:

$$Dk(\vec{\theta}) = Dk(\theta_1, \theta_2, \theta_3, \theta_4) = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} l_2 * cos(\theta_1) * l_3 * cos(\theta_1 + \theta_2) \\ l_2 * sin(\theta_1) + l_3 * sin(\theta_1 + \theta_2) \\ l_1 - \theta_3 \\ \theta_1 + \theta_2 - \theta_4 \end{bmatrix} \tag{3.26}$$

Representando la diferencia entre la posición actual del robot y la posición deseada como $f(\vec{\theta})$:

$$f(\vec{\theta}) = DK(\vec{\theta}) - \vec{p} \tag{3.27}$$

Donde $\vec{p}$ es la posición deseada del robot:

$$\vec{p} = \begin{bmatrix} Dx \\ Dy \\ Dz \\ Dw \end{bmatrix} \tag{3.28}$$

La función (3.27) que se puede optimizar con el método de Newton-Rapshon multivariable, basado en la siguiente expresión [**?**]:

$$\vec{\theta_{n+1}} = \vec{\theta_n} + J^{-1}(\vec{\theta_n}) f(\vec{\theta_n}) \tag{3.29}$$

La inversión de la matriz se lleva acabo de manera general por la pseudoinversa de Moore-Penrose en sistemas robóticos [62]. Esto permite optimizar el error a lo largo de las iteraciones 3.18 y alcanzar la configuración requierida en las articulaciones para alcazar la posición deseada.

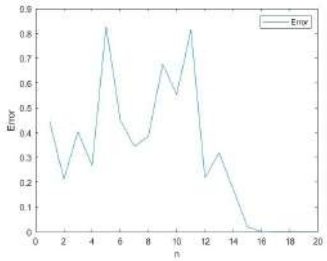

Figura 3.18: Optimización del error

Este método se puede expresar con el siguiente diagrama:

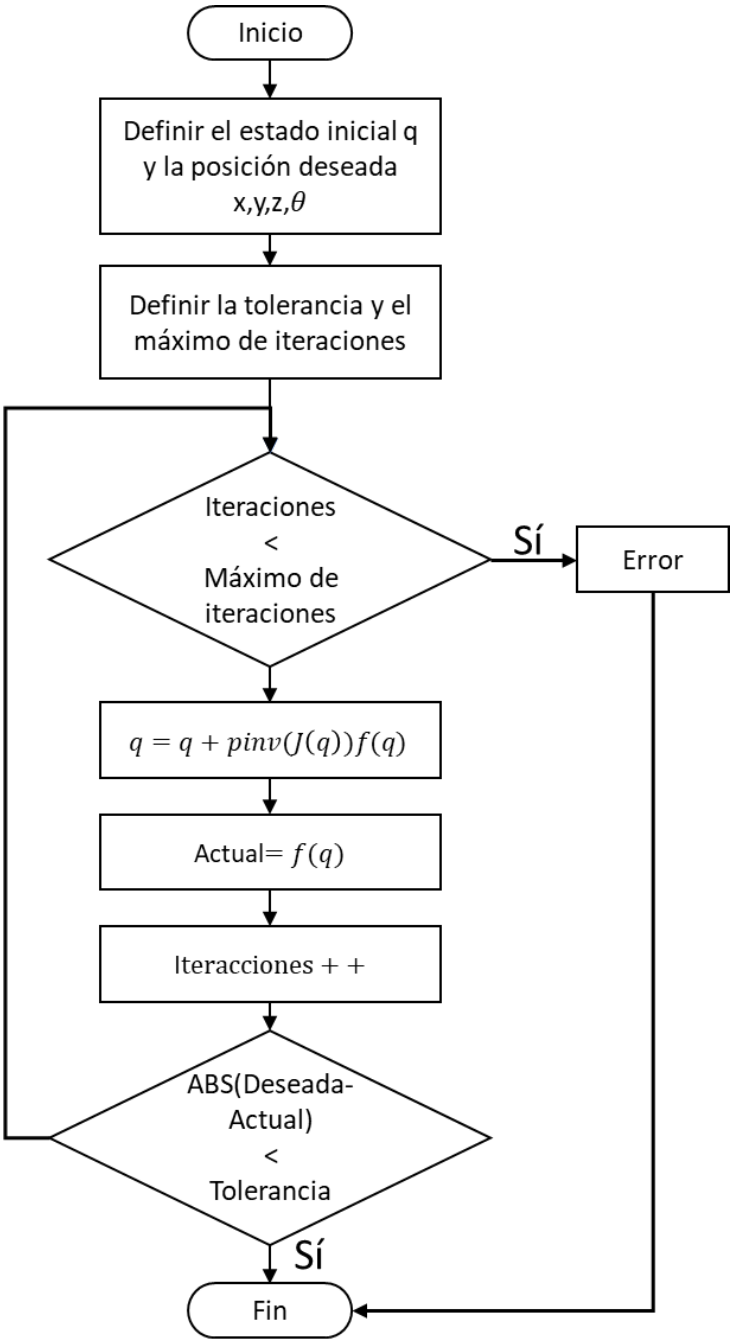

Figura 3.19: Cinemática inversa de manera numérica

Donde es posible utilizar el valor resultante para la planteación de trayectorias requeridas por el sistema robotico.

3.1.7. Generación de una trayectoria punto a punto

Para ello se define un estado inicial denotado como:

$$\vec{p_0} = Dk(\vec{q_0}) \tag{3.30}$$

Mientras que la posición deseada:

$$\vec{p_f} = [x, y, z, w]^T \tag{3.31}$$

Posteriormente se valida la posición deseada en el espacio cartesiano, y si se encuentra dentro del espacio de trabajo, se computa la cinemática inversa de la posición, es decir, que se traslada el punto al espacio de configuraciones.

$$\vec{q_f} = C_{inv}(\vec{p_f}) \tag{3.32}$$

Recordando el generador de trayectorias de Jerk mínimo, es posible expresar la ecuación 2.18 a manera de vectores como:

$$\vec{x} = \vec{x_0} + (\vec{x_f} - \vec{x_0}) * (6 * \tau^5 - 15 * \tau^4 + 10\tau^3) \tag{3.33}$$

Denotando la velocidad máxima como:

$$Vel_{max}(\vec{x_0}, \vec{x_f}, t_r) = 1{,}8750 * abs(\vec{x_f} - \vec{x_0}) * 1/t_r \tag{3.34}$$

Mientras que para la aceleración:

$$Ace_{max}(\vec{x_0}, \vec{x_f}, t_r) = 5{,}7783 * abs(\vec{x_f} - \vec{x_0}) * 1/t_r \tag{3.35}$$

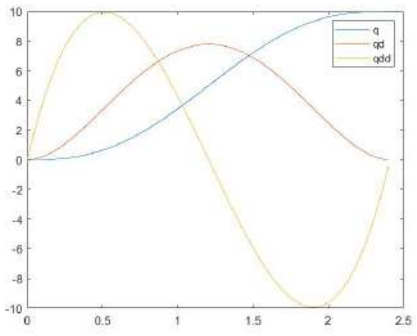

Figura 3.20: Trayectoria de Jerk mínimo con limitación de velocidad y aceleración

Al proponer un tiempo para el movimiento de ruta, se puede escalar para cumplir ya sea la restricción de velocidad o de aceleración, el código generado puede encontrarse en el anexo B. El código propuesto devuelve de la gráfica limitada 3.20 en términos de posición, velocidad y aceleración, de una manera simple y práctica de implementarse.

Modelado Dinámico del robot SCARA

El método de Newton-Euler es usado para estudiar la dinámica de sistemas robóticos [15]. Este método recursivo, permite en la primera iteración calcular la velocidad y aceleración de las masas que componen el eslabón del robot, y de manera recursiva las fuerzas que estarán en la articulación. La tabla 3.2 se usa en base a la notación 2.16. Lo cual genera la solución general dinámica del robot SCARA:

Cuadro 3.2: Parámetros DH modificados del robot SCARA

Link	$\alpha(^\circ)$	$a(m)$	$d(m)$	Variable
L1	0	0	0	$\theta_1 \in [-133, 133]^\circ$
L2	0	$l_2 = 0{,}300$	0	$\theta_2 \in [-153, 153]^\circ$
L3	180	$l_3 = 0{,}300$	d	$d \in [0, 0{,}200]m$
L4	0	0	0	$\theta_4 \in [-360, 360]^\circ$

Las iteraciones de la propagación de la velocidad y la aceleración [15] vas desde $i : 0 \rightarrow 3$, usando para las articulaciones rotacionales($i = 0, 1, 3$) las ecuaciones 2.36 y 2.37, mientras que mientras que para la articulación prismática ($i = 2$), son 2.41 y 2.42.

A continuación se detallan las matrices utilizadas para los cálculos del modelo dinámico:

$$ {}^0R_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} ; {}^1R_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.36} $$

$$ {}^2R_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} ; {}^3R_4 = \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 \\ \sin\theta_4 & \cos\theta_4 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.37} $$

$$ {}^4R_5 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.38} $$

$$ {}^0P_1 = \begin{bmatrix} 0 \\ 0 \\ L_1 \end{bmatrix} \tag{3.39} $$

$$^{1}P_{2} = \begin{bmatrix} L_2 \\ 0 \\ 0 \end{bmatrix} \tag{3.40}$$

$$^{2}P_{3} = \begin{bmatrix} L_3 \\ 0 \\ -d_3 \end{bmatrix} \tag{3.41}$$

$$^{3}P_{4} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.42}$$

$$^{4}P_{5} = \begin{bmatrix} x_{ef} \\ 0 \\ z_{ef} \end{bmatrix} \tag{3.43}$$

$$^{i}P_{C_i} = \begin{bmatrix} x_{C_i} \\ y_{C_i} \\ z_{C_i} \end{bmatrix} \tag{3.44}$$

$$^{C_i}I_i = \begin{bmatrix} I_{xx_i} & -I_{xy_i} & -I_{xz_i} \\ -I_{xy_i} & I_{yy_i} & -I_{yz_i} \\ -I_{xz_i} & -I_{yz_i} & I_{zz_i} \end{bmatrix} \tag{3.45}$$

Mientras que, para las iteraciones hacia atrás, las fuerzas actuando en el efector final serán consideradas como:

$$^{5}f_{5} = \begin{bmatrix} 0 \\ 0 \\ f_z \end{bmatrix} \tag{3.46}$$

$$^{5}n_{5} = \begin{bmatrix} 0 \\ 0 \\ n_z \end{bmatrix} \tag{3.47}$$

Las expresiones resultantes representan el torque y las fuerzas en cada articulación requeridas para el movimiento de cada una.

$$\begin{aligned}\tau_1 =& - n_z + \ddot{\theta}_1\Big(I_{zz1} + L_2^2 m_2 + m_1\Big(x_{C_1}^2 + y_{C_1}^2\Big) + m_2 L_2\Big(x_{C_2}c_2 - y_{C_2}s_2\Big) \\ &+ m_3 L_2\Big(L_2 + s_2 y_{C_3} + c_2 x_{C_3} + c_2 L_3\Big)\Big) + \ddot{\theta}_{12}\Big(I_{zz2} + I_{zz3} + I_{zz4} \\ &+ m_2\Big(L_2\Big(c_2 x_{C_2} - s_2 y_{C_2}\Big) + x_{C_2}^2 + y_{C_2}^2\Big) + m_3\Big(y_{C_3}\Big(s_2 L_2 + y_{C_3}\Big) \\ &+ \Big(x_{C_3} + L_3\Big)\Big(x_{C_3} + L_3 + c_2 L_2\Big)\Big)\Big) - \ddot{\theta}_4 I_{zz4} \\ &+ \Big(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2\Big)L_2\Big(-m_2\Big(s_2 x_{C_2} + c_2 y_{C_2}\Big) + m_3\Big(c_2 y_{C_3} - s_2 x_{C_3} - s_2 L_3\Big)\Big) \\ &+ {}^4F_{4x}\Big(y_{C_4} - \Big(L_3 + c_2 L_2\Big)s_4 + s_2 c_4 L_2\Big) - {}^4F_{4y}\Big(x_{C_4} + \Big(L_3 + c_2 L_2\Big)c_4 + s_2 s_4 L_2\Big)\end{aligned} \tag{3.48}$$

$$\begin{aligned}\tau_2 =& - n_z + \ddot{\theta}_{12}\Big(I_{zz2} + I_{zz3} + I_{zz4}\Big) - I_{zz4}\ddot{\theta}_4 + \Big(y_{C_4} - s_4 L_3\Big){}^4F_{4x} \\ &- \Big(x_{C_4} + c_4 L_3\Big){}^4F_{4y} + m_3\Big(\ddot{\theta}_{12}\Big(y_{C_3}^2 + \Big(x_{C_3} + L_3\Big)^2\Big) + \dot{\theta}_1^2 L_2\Big(s_2\Big(x_{C_3} + L_3\Big) - c_2 y_{C_3}\Big) \\ &+ \ddot{\theta}_1 L_2\Big(s_2 y_{C_3} + c_2\Big(x_{C_3} + L_3\Big)\Big)\Big) + m_2\Big(\ddot{\theta}_{12}\Big(x_{C_2}^2 + y_{C_3}^2\Big) + \dot{\theta}_1^2 L_2\Big(s_2 x_{C_2} + c_2 y_{C_2}\Big) \\ &+ \ddot{\theta}_1 L_2\Big(c_2 x_{C_2} - s_2 y_{C_2}\Big)\Big)\end{aligned} \tag{3.49}$$

$$\tau_3 = f_z + \Big(\ddot{d}_3 - g\Big)\left(m_3 + m_4\right) \tag{3.50}$$

$$\tau_4 = n_z - \Big(\ddot{\theta}_{12} - \ddot{\theta}_4\Big)\, I_{zz4} - y_{C_4}{}^4F_{4x} + x_{C_4}{}^4F_{4y} \tag{3.51}$$

$$\begin{aligned}{}^4F_{4x} =& m_4\Big(\Big(\ddot{\theta}_{12} - \ddot{\theta}_4\Big)y_{C_4} - \Big(\dot{\theta}_{12} - \dot{\theta}_4\Big)x_{C_4} - c_2 c_4 L_2\dot{\theta}_1^2 + s_2 c_4 L_2\ddot{\theta}_1 \\ &- c_4 L_3\dot{\theta}_{12}^2 - s_2 s_4 L_2\dot{\theta}_1^2 - c_2 s_4 L_2\ddot{\theta}_1 - s_4 L_3\ddot{\theta}_{12}\Big)\end{aligned} \tag{3.52}$$

$$\begin{aligned}{}^4F_{4y} =& m_4\Big(-\Big(\ddot{\theta}_{12} - \ddot{\theta}_4\Big)x_{C_4} - \Big(\dot{\theta}_{12} - \dot{\theta}_4\Big)y_{C_4} + c_2 s_4 L_2\dot{\theta}_1^2 - s_2 s_4 L_2\ddot{\theta}_1 \\ &+ s_4 L_3\dot{\theta}_{12}^2 - s_2 c_4 L_2\dot{\theta}_1^2 - c_2 c_4 L_2\ddot{\theta}_1 - c_4 L_3\ddot{\theta}_{12}\Big)\end{aligned} \tag{3.53}$$

3.2.1. Modelo simplificado

En orden para generar un modelo simplificado de la dinámica del robot SCARA que pueda ser utilizado para fines de capacitación se considera el modelo dinámico 3.21.

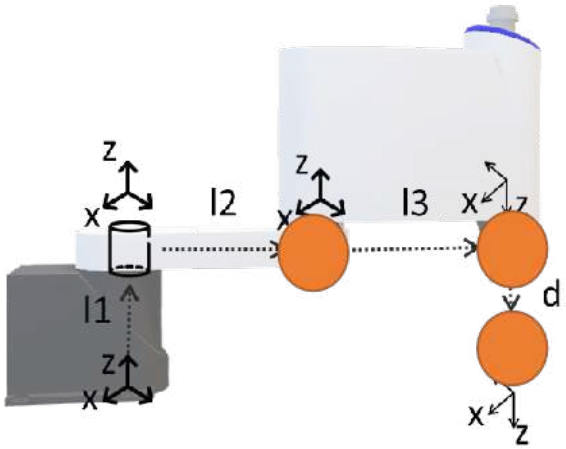

Figura 3.21: Diagrama dinámico del robot SCARA

Esto genera que las matrices de incercia puedan expresarse como:

$$ {}^{C_1}I_1 = {}^{C_2}I_2 = {}^{C_3}I_3 = {}^{C_4}I_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.54} $$

Así mismo propicia los cambios en la posición de los centros de masa de la siguiente manera:

$$ {}^1P_{C_1} = \begin{bmatrix} L_2 \\ 0 \\ 0 \end{bmatrix} ; {}^2P_{C_2} = \begin{bmatrix} L_3 \\ 0 \\ 0 \end{bmatrix} ; {}^3P_{C_3} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} ; {}^4P_{C_4} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.55} $$

Así mismo se asume que el robot, no se moverá (${}^0v_0$, ${}^0\omega_0 y {}^0\dot{\omega}_0$ son iguales a 0_{3x1}). Sin embargo, la gravedad será considerada como:

$$ {}^0\dot{v}_0 = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 9{,}81\,\mathrm{m\,s^{-2}} \end{bmatrix} \tag{3.56} $$

En el caso de que no se modelará el efector final ($m_4 = 0\,\mathrm{kg}$, $x_{ef} = 0$ and $y_{ef} = 0$) permite expresar el sistema de manera simplificada como se muestra a continuación.

$$ \begin{aligned} \tau_1 \quad = \quad & -n_z + \ddot{\theta}_1\Big(m_2L_2^2 + m_1L_2^2 + m_2c_2L_2L_3 + m_3L_2\Big(L_2 + c_2L_3\Big)\Big) \\ & +\ddot{\theta}_{12}\Big(m_2\Big(c_2L_2L_3 + L_3^2\Big) + m_3L_3\Big(L_3 + c_2L_2\Big)\Big) \\ & -s_2L_2L_3\Big(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2\Big)\Big(m_2 + m_3\Big) \end{aligned} \tag{3.57} $$

$$\tau_2 = -n_z + (m_2 + m_3)\left(L_3^2\ddot{\theta}_{12} + s_2 L_2 L_3 \dot{\theta}_1^2 + c_2 L_2 L_3 \ddot{\theta}_1\right) \tag{3.58}$$

$$\tau_3 = f_z + \left(\ddot{d}_3 - g\right) m_3 \tag{3.59}$$

$$\tau_4 = n_z \tag{3.60}$$

3.2.2. Añadiendo los términos de fricción

En orden para considerar los términos de fricción evitando que un sistema robótico permanezca en movimiento ante una perturbación, aun cuando esta sea una señal de control o debido a efectos numéricos de la simulación. Esto es posible al usar la función de Rayleigh, denotada en sistemas planteados mediante la ecuación de Euler-Lagrange como [63]:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} = 0 \tag{3.61}$$

En el caso de sistemas perturbados (Con entrada de τ o f):

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} = Q_i \tag{3.62}$$

D, es la función de disipación de Rayleigh [64]:

$$D = \frac{1}{2}(b_i\delta_1^2 + b_2\delta_2^2 + ... + b_r\delta_r^2) \tag{3.63}$$

En sistemas robóticos es planteada como:

$$D = \frac{1}{2}(b_1\dot{q_i}^2 + b_2\dot{q_2}^2 + ... + b_n\dot{q_n}^2) \tag{3.64}$$

Esto quiere decir que la fricción viscosa de una articulación, es simplemente:

$$\frac{\partial D}{\partial \dot{q_n}} = b_n\dot{q_n} \tag{3.65}$$

Transformando la dinámica de un robot de:

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) \tag{3.66}$$

a:

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F_v(\dot{q}) \tag{3.67}$$

Siendo F_v, un vector fila denotado por:

$$F_v(\dot{q}) = \begin{bmatrix} b_0\dot{q_0} \\ b_1\dot{q_1} \\ ... \\ b_n\dot{q_n} \end{bmatrix} \tag{3.68}$$

3.2.3. Dinámica directa

Usando el algoritmo conocido como “The Composite-Rigid-Body Method” o “The Composite-Rigid-Body Method Algorithm”, es el método más rápido disponible para computar la dinámica directa [65]. Mientras que la dinámica inversa calcula la fuerza o torque para mover una articulación en base a un movimiento [50]:

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) \tag{3.69}$$

Sin embargo, si se requiere conocer el movimiento generado por una fuerza o torque de entrada, se puede despejar $\ddot{q}$, que representa la aceleración de la articulación:

$$\ddot{q} = M(q)^{-1}(\tau - (C(q,\dot{q})\dot{q} + G(q))) \tag{3.70}$$

De manera que este método, abreviado como CRB, denota esta expresión como[65]:

$$\ddot{q} = M(q)^{-1}(\tau - \tau_0) \tag{3.71}$$

Donde:

$$\tau_0 = C(q,\dot{q})\dot{q} + G(q) \tag{3.72}$$

En este caso, si consideramos la fricción, como $fv(\dot{q})$ recordando (3.65):

$$\tau_0 = C(q,\dot{q})\dot{q} + G(q) + F_v(\dot{q}) \tag{3.73}$$

Integrando el cálculo con el método de Newton-Euler:

$$rne(q,\dot{q},\ddot{q},g,b) = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F_v(\dot{q}) \tag{3.74}$$

τ_0 puede ser encontrado como:

$$\tau_0 = rne(q,\dot{q},0,g,b) = C(q,\dot{q})\dot{q} + G(q) + F_v(\dot{q}) \tag{3.75}$$

Mientras que la matriz de inercia como:

$$M_{:,i}(q)\ddot{q} = rne(q, eye(n)_{:,i}, 0, 0, 0) \tag{3.76}$$

Tomando las ecuaciones computadas de la dinámica del SCARA, se denota como:

$$M_{:,0} = \begin{bmatrix} (m_1 + m_2 + m_3)\ {l_1}^2 + (m_2 + m_3)\left({l_3}^2 + l_2\ \cos(\theta_2)\ l_3\right) + l_2\, l_3\ \cos(\theta_2) \\ (m_2 + m_3)\left({l_3}^2 + l_2\ \cos(\theta_2)\ l_3\right) \\ 0 \end{bmatrix} \tag{3.77}$$

$$M_{:,1} = \begin{bmatrix} (m_2 + m_3) & (m_2 + m_3)\left({l_3}^2 + l_2\ \cos(\theta_2)\ l_3\right) \\ {l_3}^2(m_2 + m_3) & \\ 0 & \end{bmatrix} \tag{3.78}$$

$$M_{:,2} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.79}$$

Mientras que τ_0, se denota por:

$$\tau_0 = \begin{bmatrix} b_1\,\theta_{1\mathrm{d}} - l_2\,l_3\ \sin(\theta_2)\ (m_2 + m_3)\left({\theta_{2\mathrm{d}}}^2 + 2\,\theta_{1\mathrm{d}}\,\theta_{2\mathrm{d}}\right) \\ l_2\,l_3\ \sin(\theta_2)\ (m_2 + m_3)\,{\theta_{2\mathrm{d}}}^2 + b_2\,\theta_{2\mathrm{d}} \\ b_3\,\theta_{3\mathrm{d}} - \frac{981\,m_3}{100} \end{bmatrix} \tag{3.80}$$

3.2.4. Simulación en tiempo real

Si se considera el siguiente caso, donde se requiere un método de integración numérica para computar los valores de q y $\dot{q}$ 2.49:

$$\vec{\theta}'' = CRB(\vec{\theta}, \vec{\theta}', \vec{\tau}, \vec{b}) \tag{3.81}$$

En este trabajo se evalúan 3 métodos numéricos, siendo el de Euler el más simple, el de Runge-Kutta, se basa en 4 aproximaciones, mientras que el de Dormand-Prince, es de paso adaptativo. Evaluando la estabilidad y la carga computacional que demandaría el computo del mismo. Los parámetros utilizados para esta simulación consisten en dos entradas escalón a diferente tiempo y con diferente magnitud, para observar su comportamiento:

$$\tau_1 = 0{,}05(step(t-3) - step(t-5)); \tag{3.82}$$

$$\tau_2 = 0{,}1(step(t-2) - step(t-6)) \tag{3.83}$$

$$b_1 = 2; b_2 = 2; \tag{3.84}$$

$$m_1 = 1; m_2 = 1; \tag{3.85}$$

$$l_1 = 0{,}139; l_2 = 0{,}300; l3 = 0{,}300 \tag{3.86}$$

En la gráfica 3.22 se pueden comparar los 3 métodos, se observa que tanto el método de Runge-Kutta como el de Dormand-Prince se mantienen estables, mientras que el método de Euler, se muestra de manera inestable, debido a la oscilación y la diferencia que tiene respecto a los otros métodos.

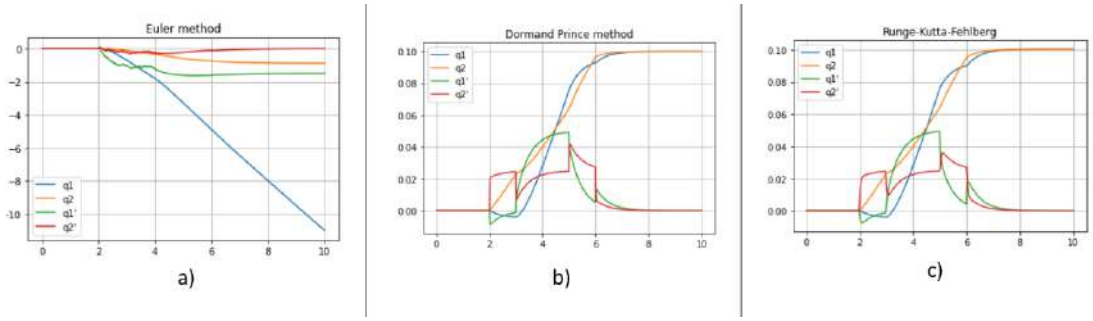

Figura 3.22: Comparativa de distintos métodos de integración numérica

Sin embargo, dada la complejidad del método de Dormand-Prince, es preferible, utilizar el método de Runge-Kutta, el cual es sometido ante señales de escalón y rampa [45] para comprobar su estabilidad, referido en las gráficas 3.23.

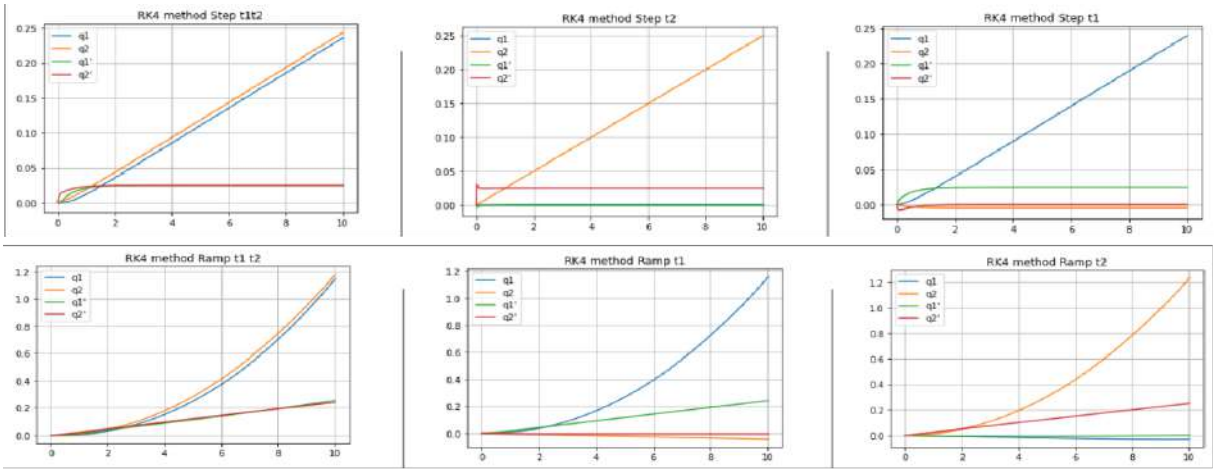

Figura 3.23: Pruebas de estabilidad de la simulación de la dinámica directa

En cada gráfica, se detalla el comportamiento de la articulación 1 y 2 del robot SCARA, para verificar que no existan oscilaciones, cambios bruscos o desbordamientos númericos, relativos a la simualción númerica del método seleccionado.

Para analizar la respuesta de la cinemática directa, se computa el torque requerido en la dinámica inversa, y se introduce como parámetro en la dinámica directa, para comparar el error que existe en la dinámica directa esto se observa en 3.24.

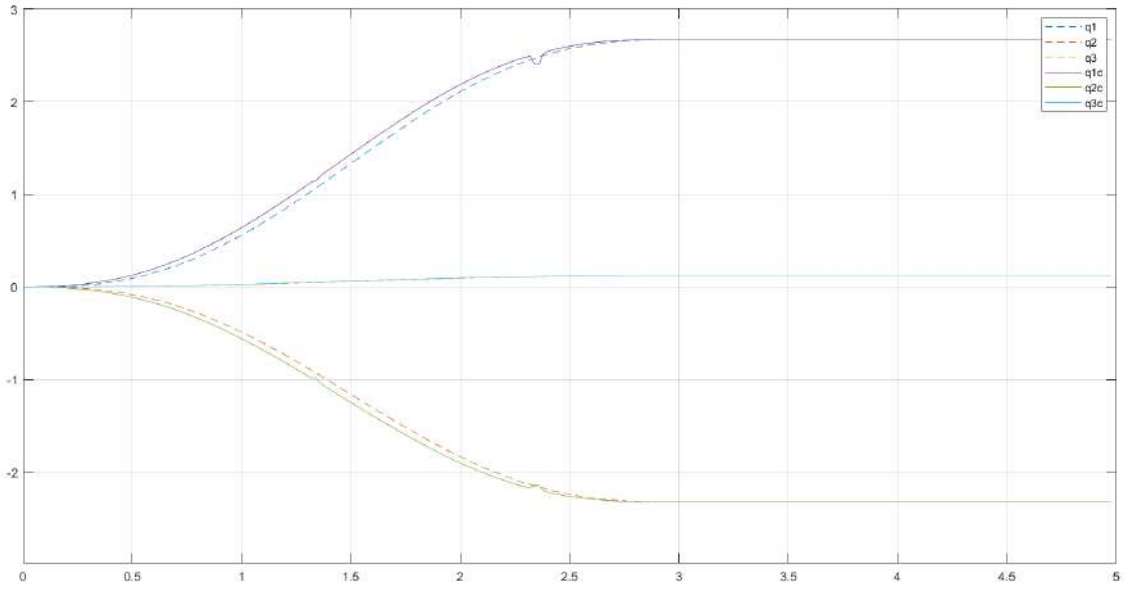

Figura 3.24: Comparativa entre la dinámica directa y la inversa

La diferencia en 3.24 entre la dinámica directa e inversa, se trabajará en base a solo considerar el modelado dinámico de los elementos que verdaderamente intervienen en el diseño.

3.2.5. Ajustes en la dinámica para la simulación en tiempo real

Para el modelo simplificado se omiten los efectos de la cuarta articulación, y del efector final, siendo que para esta aplicación pick and place, no se considera una aplicación donde importe la orientación de la pieza sujeta, y así mismo, el efector final al ser un electro-imán, se puede incluir en la altura del eslabón final.

Link	$\alpha(^\circ)$	$a(m)$	$d(m)$	Variable
L1	0	l_2	l_1	$\theta_1 \in [-133, 133]^\circ$
L2	0	$l_3 = 0{,}300$	0	$\theta_2 \in [-153, 153]^\circ$
L3	180	0	d	$d \in [0, 0{,}200]m$

Cuadro 3.3: Tabla de parámetros DH ajustada a tiempo real

$$ {}^1P_{C_1} = \begin{bmatrix} L_2 \\ 0 \\ L_1 \end{bmatrix} ; {}^2P_{C_2} = \begin{bmatrix} L_3 \\ 0 \\ 0 \end{bmatrix} ; {}^3P_{C_3} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} ; \tag{3.87} $$

Al aplicar el método “Composite-Rigid-Body Method”, es necesario volver a calcular los términos de τ_0 y de la matriz de inercia $M(q)$.

$$\tau_0 = \begin{pmatrix} -\sin(\theta_2)\ l_2\, l_3\, \theta_{\mathrm{d}2}\, (2\, m_2 + m_3)\, (2\, \theta_{\mathrm{d}1} + \theta_{\mathrm{d}2}) \\ \sin(\theta_2)\ l_2\, l_3\, (\theta_{\mathrm{d}1})^2\, (2\, m_2 + m_3) \\ -g\, m_3 \end{pmatrix} \tag{3.88}$$

$$\begin{aligned} M_{0,0} &= 4\,(l_2)^2\, m_1 + (l_2)^2\, m_2 + (l_2)^2\, m_3 + 4\,(l_3)^2\, m_2 \\ &\quad + (l_3)^2\, m_3 + 4\,\cos(\theta_2)\ l_2\, l_3\, m_2 + 2\,\cos(\theta_2)\ l_2\, l_3\, m_3 \end{aligned} \tag{3.89}$$

$$M_{1,0} = l_3\, m_3\, (l_3 + \sigma_1) + 2\, l_3\, m_2\, (2\, l_3 + \sigma_1) \tag{3.90}$$

$$\sigma_1 = cos(\theta_2) * l_2 \tag{3.91}$$

$$M_{2,0} = 0 \tag{3.92}$$

$$M_{:,2} = \begin{pmatrix} 4\,(l_3)^2\, m_2 + (l_3)^2\, m_3 + \cos(\theta_2)\ l_2\, (2\, l_3\, m_2 + l_3\, m_3) \\ (l_3)^2\, (4\, m_2 + m_3) \\ 0 \end{pmatrix} \tag{3.93}$$

$$M_{:,3} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.94}$$

Control dinámico del robot SCARA

El control dinámico consiste en proponer una ley de control para convertir el sistema robótico, desde la ecuación dinámica 3.69 en un sistema lineal, compensando los términos no lineales del sistema.

$$\ddot{q} = M(q)^{-1}(\tau - C(q,\dot{q}) - G(q) - F_v(\dot{q})) \tag{3.95}$$

Mientras que la ley de control por torque controlado se puede definir como:

$$\tau_c = M(q)(\ddot{q}_d + K_d\dot{e} + K_p e) + C(q,\dot{q})\dot{q} + G(q) + Fv(\dot{q}) \tag{3.96}$$

Donde si se sustituye la expresión (3.96) en 3.95.

$$\ddot{q} = M(q)^{-1}(M(q)(\ddot{q}+K_d\dot{e}+K_p q_d)+C(q,\dot{q})\dot{q}+G(q)+Fv(\dot{q})-C(q,\dot{q})-G(q)-F_v(\dot{q})) \tag{3.97}$$

Puede simplificarse a:

$$\ddot{q} = M(q)^{-1}(M(q)(\ddot{q} + K_d\dot{e} + K_p e)) \tag{3.98}$$

$$\ddot{q} = \ddot{q_d} + K_d\dot{e} + K_p e \tag{3.99}$$

Si se define que $e = q_d - q$, entonces:

$$\ddot{q} = \ddot{q_d} + K_d(\dot{q_d} - \dot{q}) + K_p(q_d - q) \tag{3.100}$$

Al aplicar la transformada de Laplace:

$$s^2Q(s) = s^2Q_d(s) + K_d s(Q_d(s) - Q(s)) + K_p(Q_d(s) - Q(s)) \tag{3.101}$$

De manera que al separar los términos se tiene que:

$$s^2Q(s) + K_d sQ(s) + K_pQ(s) = s^2Q_d(s) + sK_dQ_d(s) + K_pQ_d(s) \tag{3.102}$$

$$\frac{Q(s)}{Q_d(s)} = \frac{s^2 + K_d s + K_p}{s^2 + K_d s + K_p} = 1 \tag{3.103}$$

Cualquier valor que se escoja, resultaría en una identidad entre la señal de entrada y la señal de salida, sin embargo, carece de pautas para la selección correcta de los parámetros k_p y k_d.

Ahora bien, si la dinámica del sistema en base a 3.70 se puede rescribir como:

$$\ddot{q} = M(q)^{-1}(\tau - \tau_0) = M(q)^{-1}(\tau) - M(q)^{-1}(\tau_0) \tag{3.104}$$

Al sustituir el valor de τ_0 por sus componentes:

$$\ddot{q} = M(q)^{-1}(\tau) - M(q)^{-1}(C(q,\dot{q})\dot{q} + G(q) + F_v(\dot{q})) \tag{3.105}$$

Donde se define w, como los términos adicionales del modelo es decir, los efectos de Coriolis y de fricción:

$$w = M(q)^{-1}(C(q,\dot{q})\dot{q} + F_v(q)) \tag{3.106}$$

Permite escribir la dinámica de la siguiente manera:

$$\ddot{q} = M(q)^{-1}(\tau) - w - M(q)^{-1}G(q) \tag{3.107}$$

Se introduce τ_c junto con la compensación de la gravedad:

$$\ddot{q} = M(q)^{-1}(\tau_c + G_c(q)) - w - M(q)^{-1}G(q) \tag{3.108}$$

Permite cancelar la carga de la gravedad en el sistema robótico, si $M(q)^{-1}G_c(q) - M(q)^{-1}G(q) = 0$

$$\ddot{q} = M(q)^{-1}(\tau_c) - w \tag{3.109}$$

Siendo entonces w el cambio en el modelo debido a variaciones de carga, fricción, o los mismos efectos de Coriolis y las fuerzas centrifugas. Entonces los ajustes de la sintonía del controlador pueden efectuarse en base a $w = 0$, dicha sintonía considera que el robot se mueve a una velocidad, donde los efectos de la fricción y de Coriolis son despreciables.

$$\ddot{q} = M(q)^{-1}(\tau_c) \tag{3.110}$$

Sin olvidar que la forma completa del controlador con compensación de gravedad, será:

$$G_q(s) = \tau_c + G_c(q) \tag{3.111}$$

O bien, controlador sin compensación de gravedad:

$$G_q(s) = \tau_c \tag{3.112}$$

Permitiendo expresar el sistema en base a un modelo de espacio de estados, compuesto de la siguiente manera:

$$\dot{x} = Ax + Bu \tag{3.113}$$

$$y = Cx + Du \tag{3.114}$$

Para computar los valores de sintonía del controlador en base a tau_c, se pueden ajustar en base a un punto de operación del sistema robótico como $q = [q_1(0), q_2(0), ...q_n(0)]$, o bien, en tiempo real como: $q(t) = [q_1(t), q_2(t), ...q_n(t)$.

Revisando la expresión (3.110) es posible establecer una matriz de inercia instantánea conocida como M_s

$$M_s = M(q(t)) \tag{3.115}$$

Así mismo se define un vector de estados que representa la dinámica del sistema como:

$$x = \begin{bmatrix} q_0 \\ q_1 \\ ... \\ q_n \\ \dot{q_0} \\ \dot{q_1} \\ ... \\ \dot{q_n} \end{bmatrix} \tag{3.116}$$

Permitiendo expresar el sistema como:

$$\dot{x} = Ax + Bu \tag{3.117}$$

$$y = Cx + Du \tag{3.118}$$

Lo importante en este análisis recae en que A, representa la integración de la velocidad y será conformada por:

$$A = \begin{bmatrix} zeros(n,n) & eye(n) \\ zeros(n,n) & zeros(n,n) \end{bmatrix} \tag{3.119}$$

Mientras que B, será denotada por, representando el estado del sistema en base a la matriz de inercia:

$$B = \begin{bmatrix} zeros(n,n) & zeros(n,n) \\ zeros(n,n) & inv(M_s) \end{bmatrix} \tag{3.120}$$

Posteriormente C, puede ser denotada por una matriz identidad de ($C \in \mathbb{R}^{2n} * \mathbb{R}^{2n}$). , y siendo D, una matriz vacía ($D \in \mathbb{R}^{2n} * \mathbb{R}^{2n}$). n se refiere al número de articulaciones del robot bajo análisis:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.121}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2{,}64 & -4{,}233 & 0 \\ 0 & 0 & 0 & -4{,}233 & 8{,}995 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.122}$$

Es posible computar las funciones de transferecia en base a la fórmula [45]:

$$H(s) = C(sI - A)^{-1}B + D \tag{3.123}$$

Donde únicamente se consideran los valores, de las funciones de transferencia en base al actuador principal:

$$tf_{\tau_1 \to q_1} = tf_{mayorEfecto} \tag{3.124}$$

Estas funciones de transferencia cuando $q = [q_1(0), q_2(0), q_3(0)]$ son:

$$tf_{\tau_1 \to q_1} = 1/s * 2{,}46/s \tag{3.125}$$

$$tf_{\tau_2 \to q_2} = 1/s * 8{,}995/s \tag{3.126}$$

$$tf_{\tau_3 \to q_3} = 1/s * 1/s \tag{3.127}$$

Sistema controlado mediante el esquema de control 3.25

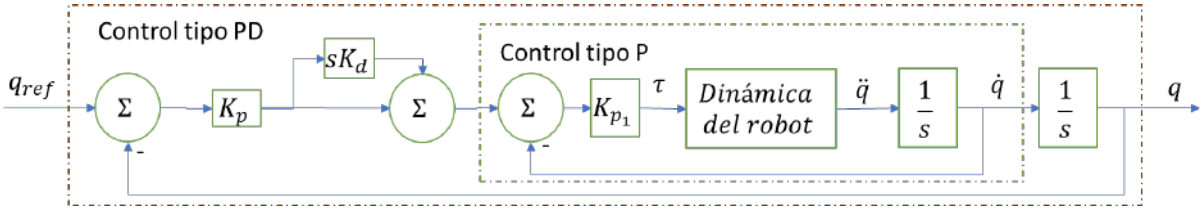

Figura 3.25: Diagrama de la estrategia de control para el robot SCARA

El modelado de la velocidad en las articulaciones corresponde de manera general a la forma:

$$tf_{vel} = k/s \tag{3.128}$$

Donde la técnica de Internal Model Controller [51] permite controlar el sistema, para obtener un tiempo de ascentamiento deseado λ, es decir, convertir la dinámica del sistema a 3.129.

$$Sys_{controlado} = \frac{1}{\lambda s + 1} \tag{3.129}$$

Es importante para la selección de λ, sea la misma para todas las articulaciones, evitando problemas de oscilaciones que se transmitan a lo largo del sistema robotico, para este caso,$\lambda_{vel} = 0{,}02 * 80$, que corresponde al control de la velocidad. De manera que las ganancias en base a la técnica de IMC son:

$$K_v el = \frac{1}{K_{sys} * \lambda_{vel}} = \begin{bmatrix} 0{,}2363 \\ 0{,}0695 \\ 0{,}6250 \end{bmatrix} \tag{3.130}$$

Al incorporar el control en la velocidad, las articulaciones se comportan de la siguiente manera:

$$sys = \frac{1}{s} \frac{1}{\lambda_{vel} s + 1} \tag{3.131}$$

Para la selección de la lamba del control de la posición, se necesita evaluar el rechazo ocasionado por los efectos de Coriolis $C(q, \dot{q})\dot{q}$. Esto puede evaluarse en la simulación de la dinámica directa, para este caso $\lambda_{pos} = \lambda_{vel}/8$ mostro un buen desempeño. El cálculo de las ganancias generan un controlador PD, usando IMC, denotadas por:

$$k_{pos} = \frac{1}{\lambda_{pos}} = \begin{bmatrix} 5 \\ 5 \\ 5 \end{bmatrix} \tag{3.132}$$

$$D_{pos} = \lambda_{pos} = \begin{bmatrix} 8 \\ 8 \\ 8 \end{bmatrix} \tag{3.133}$$

La salida del controlador PD-P es τ_c. Las ganancias obtenidas y el control generado, puede ser probado en cada articulación, utilizando un disco de incertidumbre, el cual, puede modificar la ganancia y la fase de la función de transferencia, para observar el comportamiento del controlador, ante una perturbación externa, o bien, una inexactitud del modelado.

En este caso para la segunda articulación, se utiliza un disco de incertidumbre para variar la ganancia hasta 1.5 veces, y la fase de $-30\,\mathrm{deg}$ hasta $30\,\mathrm{deg}$ En el caso de la articulación dos, se puede definir como:

$$controlV = feedback(K_{vel_2} * F * G2, 1) \tag{3.134}$$

$$controlPD = pid(k_{pos_2}, 0, D_{pos_2}) \tag{3.135}$$

$$PosControl = feedback(controlPD * VelControl * tf(1,[1,0]),1) \tag{3.136}$$

Permitiendo observar que el control del sistema es estable ante estas variaciones, ante una entrada de tipo escalón en la figura 3.26

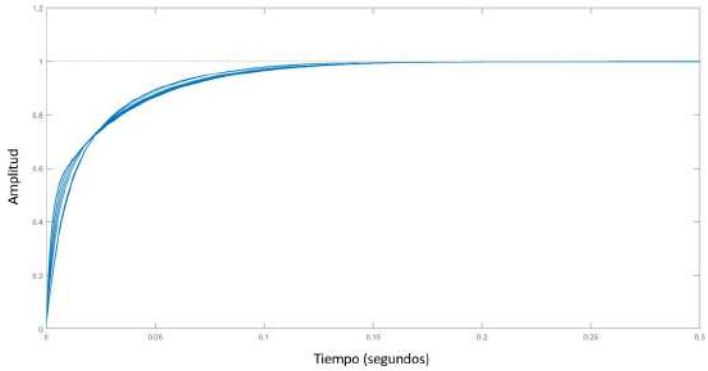

Figura 3.26: Respuesta del controlador de la articulación 2, ante las variaciones

Resultados similares se obtuvieron con las otras articulaciones, donde se observo la estabilidad ante la incertidumbre que pudiera tener el modelo. Estas ganancias al involucrarse en el modelo dinámico considera la compensación de la gravedad como:

$$G_c = rne(q,0,0,g,0) \tag{3.137}$$

De manera que la señal de control es:

$$u = \tau_c + rne(q,0,0,g,0) \tag{3.138}$$

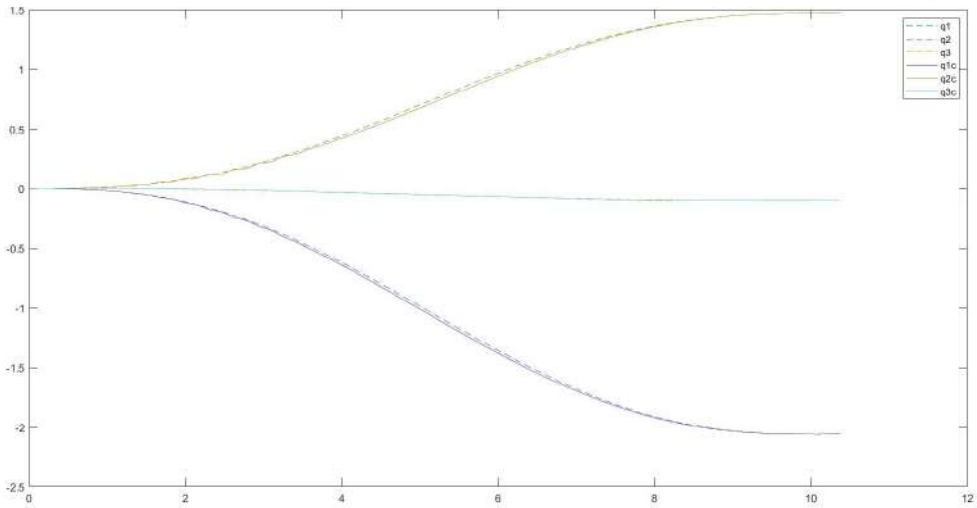

Figura 3.27: Respuesta del sistema robótico usando el control propuesto

El resultado de la simulación de la dinámica 3.27 directa utilizando el controlador sintonizado para el seguimiento de una trayectoria, muestra un seguimiento aceptable y estable. El código generado para la simulación del control de la dinámica directa, se puede encontrar en el anexo B.

Diseño del gemelo digital en Unity 3D

Para la optimización del proyecto para dispositivos móviles el proyecto utilizá el "Universal Render Pipeline" de Unity 3D, como se muestra en la figura 3.28

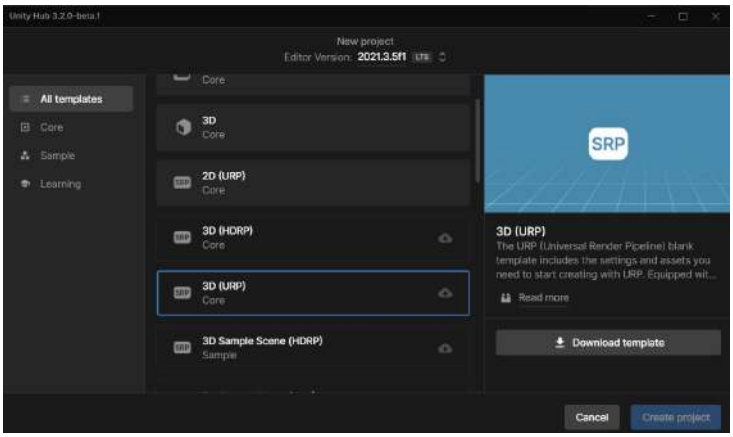

Figura 3.28: Creación del proyecto dentro de Unity 3D

Una vez creado el proyecto, se importarón los modelos desde Unity 3D como en la imagen 3.29

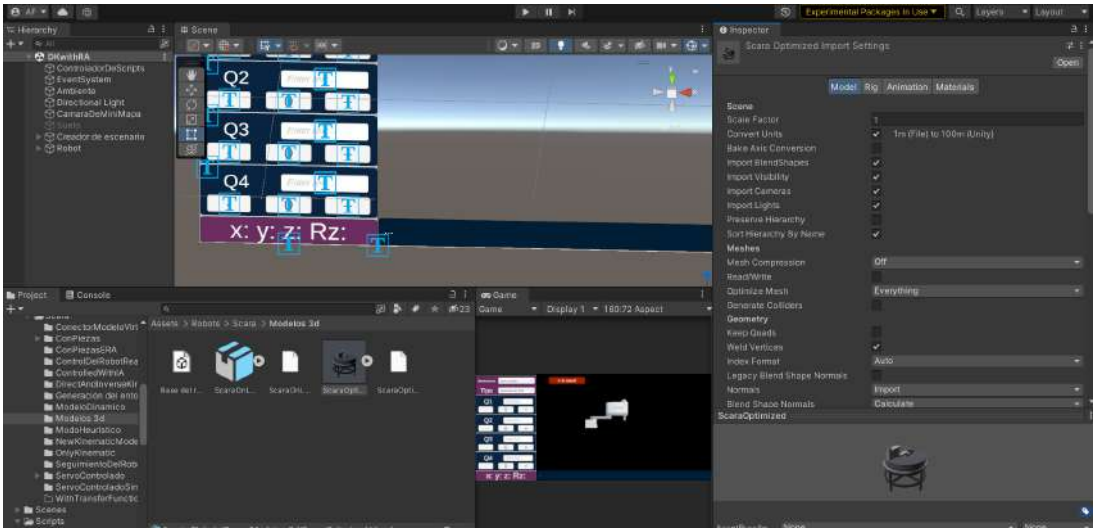

Figura 3.29: Robot importado en Unity 3D

El sistema robótico se construyó aprovachando las posiciones relativas que maneja Unity 3D, definida por su jerarquía de objetos en 3D 3.30.

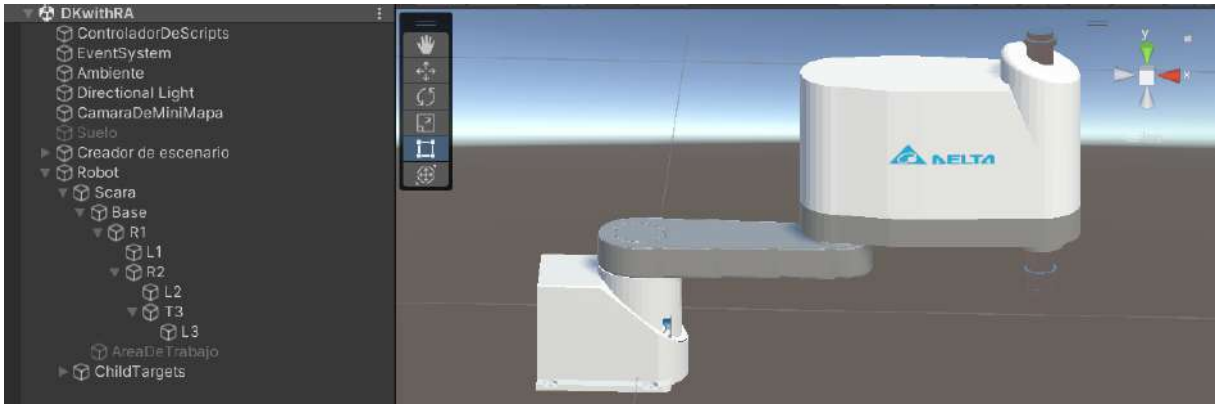

Figura 3.30: Jerarquía del robot SCARA en Unity 3D

La jerarquía 3.31 ayuda a generar los movimientos del sistema robótico, sin necesidad de incluir una programación adicional.

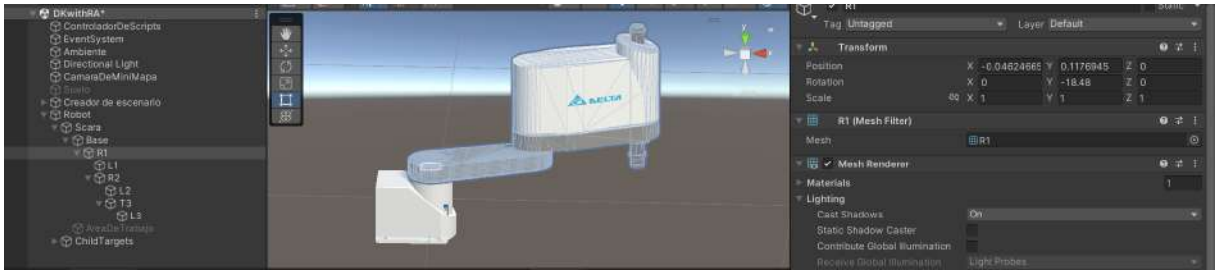

Figura 3.31: Ejemplo de movimiento del Robot SCARA en Unity

Al controlar la orientación y rotación de los “GameObject”.

3.4.1. Cinemática en Unity 3D

Para generar el movimiento suave entre el cambio de las posiciones del sistema robótico generado en Unity 3D, se modela en base a una ecuación diferencial de primer orden, que actua como un cambio suave entre un valor y el otro:

$$y' + ay = bx \tag{3.139}$$

Donde la ecuación de transferencia es[45]:

$$sY(s) + aY(s) = bX(s) \tag{3.140a}$$

$$Y(s)(s+a) = bX(s) \tag{3.140b}$$

$$\frac{Y(s)}{X(s)} = \frac{b}{s+a} \tag{3.140c}$$

$$\frac{Y(s)}{X(s)} = \frac{K}{\tau_s s + 1} \tag{3.140d}$$

$$\tag{3.140e}$$

Se denota que el valor de τ_s, es constante de tiempo del sistema, si definimos $K = 1$, esto significa, el tiempo que la salida se volverá la salida, si la ecuación diferencial (3.139), es discretizada, se puede llegar a la siguiente expresión:

$$y_n = y_{n-1} * \alpha + (1-\alpha)x_n \tag{3.141a}$$

$$\alpha = \frac{\tau}{Time_{sampling} + \tau} \tag{3.141b}$$

Los cálculos de la cinemática directa en base al modelo del Robot SCARA DRS60L 3.32 se realizan en el código “CinematicaDirecta.cs” (Anexo E).

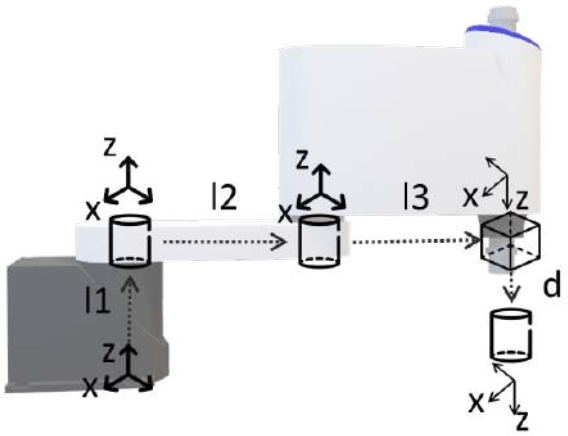

Figura 3.32: Ubicación de las articulaciones en Unity 3D

El vinculo entre la posición y rotación relativa de las articulaciones en Unity 3D respecto a los valores de las articulaciones son provistos por el código de “QtoUnity.cs”(Anexo F). con el fin de representar correctamente las posiciones en el espacio tridimensional de Unity 3D.

Posteriormente para animar el modelo, los cambios en los valores de las articulaciones son computados primero por el sistema de primer orden, y su salida, enviada

al script de “CinematicaDirecta.cs”, permitiendo que, ante un cambio en el valor deseado de las articulaciones, se genere una animación, este código tiene el nombre de “ServoControlador.cs” (Anexo G).

Estos scripts son agregados a la base del robot SCARA y se vinculan los componentes necesarios para la ejecución 3.33.

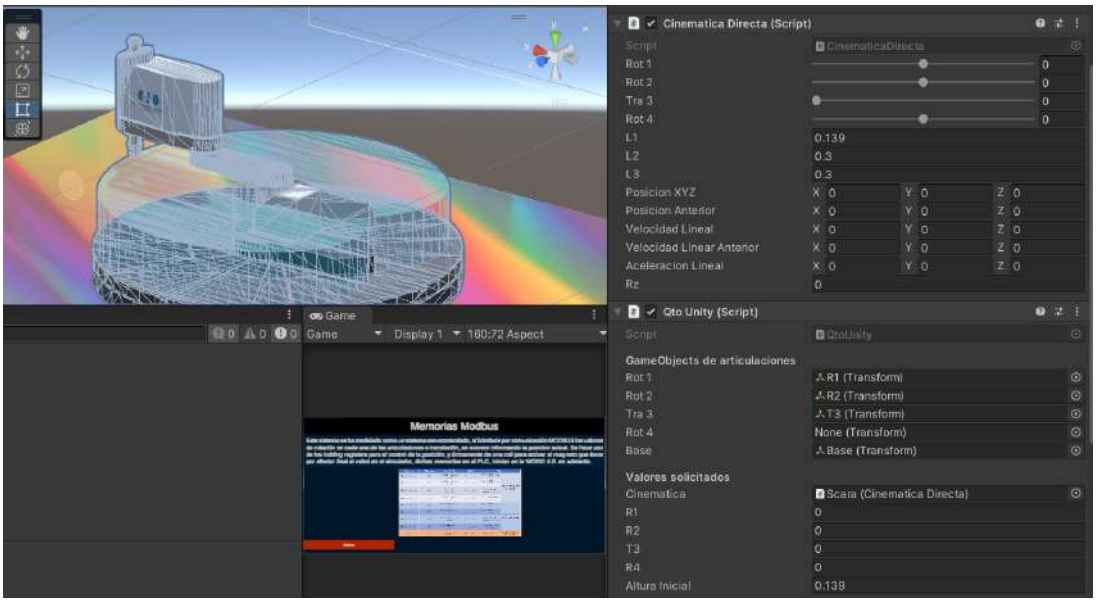

Figura 3.33: Robot con scripts en Unity 3D para el modelado cinemático

A continuación se muestra un ejemplo del movimiento del robot 3.34, para comprobar el correcto funcionamiento de los códigos generados.

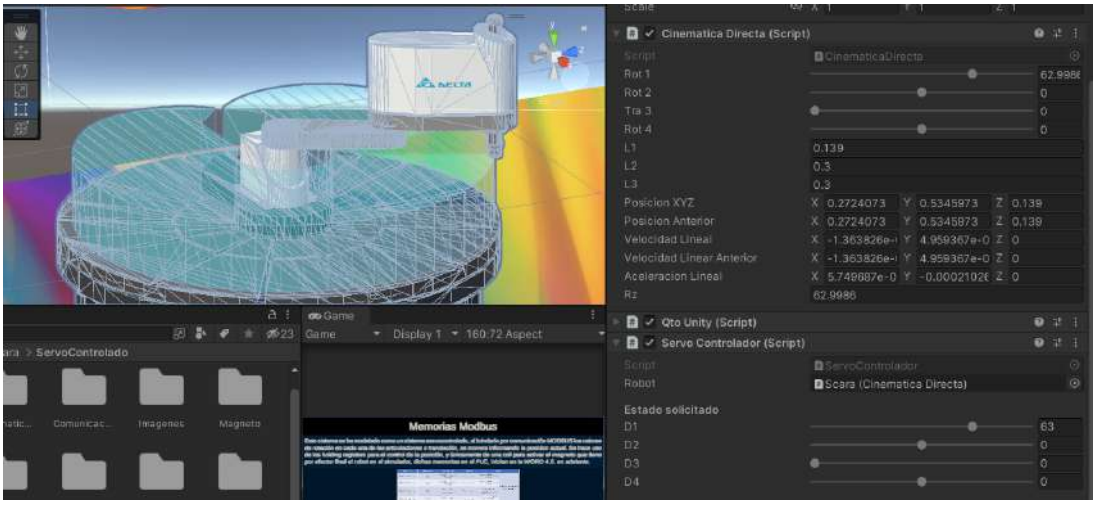

Figura 3.34: Pruebas de movimiento en Unity 3D

3.4.2. Comportamiento dinámico del gemelo virtual

Una vez que la cadena Cinemática del sistema robótico se ha construido en Unity 3.35, a cada eslabón, se introduce el script “Articulated Body” encontrado en el “Robotics Toolbox” de Unity 3D, que realiza la simulación dinámica en base al algoritmo de “Articulated Rigidbody ” de Featherstone.

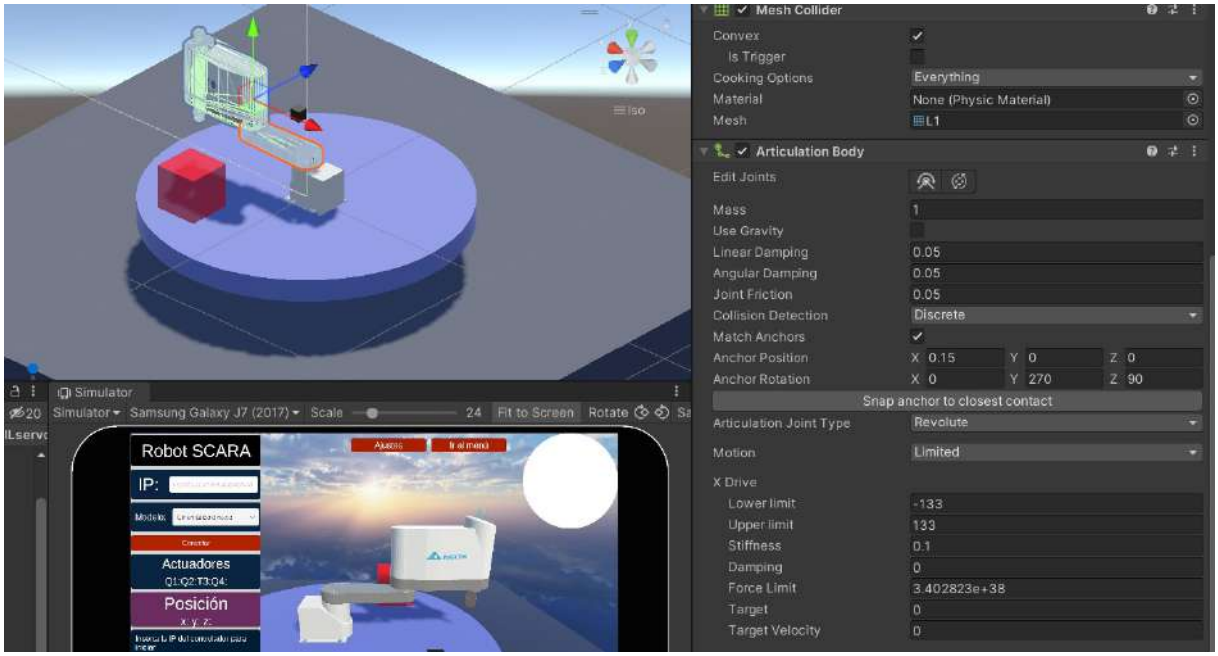

Figura 3.35: “Articulated Body en Unity”

Configurando las articulaciones en base a los siguientes paramétros:

- Base, “Fixed”
- L1, “Revolute”, mínimo -133, máximo 133, stifness 0.1
- L2, “Revolute”, mínimo -153, máximo 153, stifness 0.1
- J3, “Pismatic”, mínimo -0.2, máximo 0, stifness 0.1

Esto permite establecer la dinámica en el motor de físicas de Unity 3D usando el PhysisX, y además, la acota al espacio de configuraciones, planteado en las secciones anteriores.

Para el control del robot virtual es necesario seguir la estrategia presentada en base a (3.131). De manera general, el control, según el caso puede ser PID, PD, o cualquier otro esquema, por generalización, se ilustra con un PID en base al diagrama 3.36

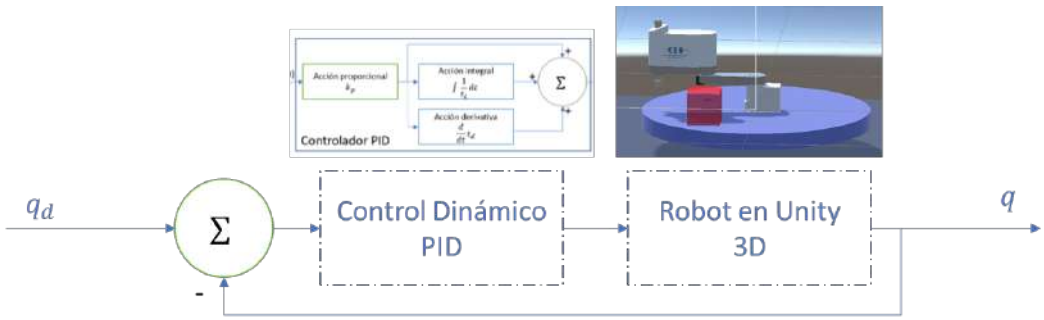

Figura 3.36: Control del gemelo virtual en Unity

Para evaluar la diferencia del modelo planteado y la simulación realizada en Unity 3D, se genera cada articulación como “Fixed” dejando al análisis una por una, introduciendo una señal escalón, para análizar la función de transferecia de cada una, para evitar la interferencia de las otras en el sistema robótico. Identificando los valores relativos a la matriz de inercia $M_{i,i}$, junto con el componente de la fricción, siendo la relación principal del τ_i y la aceleración $\ddot{q}_i$. Para ello se introduce una una entrada escalón, obteniendo el comportamiento 3.37.

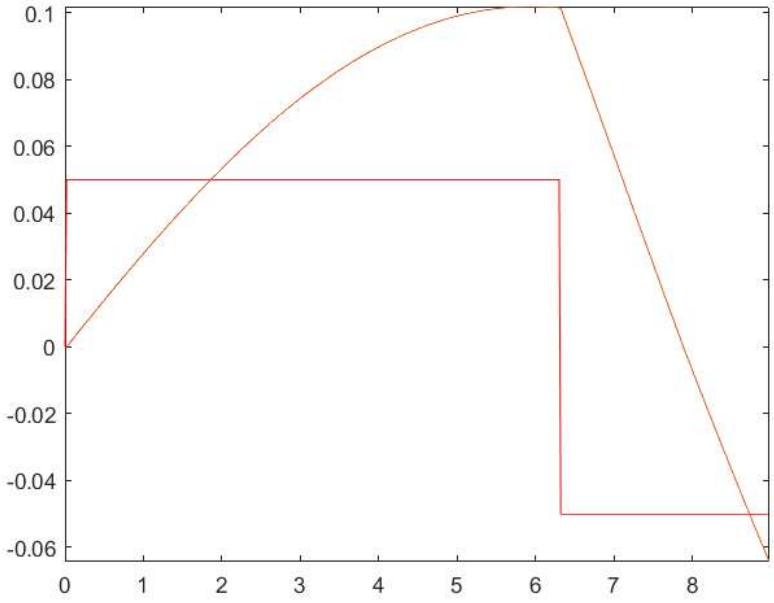

Figura 3.37: Idenficicación de la articulación 1, Entrada vs Velocidad

Recordando que al considerar la fricción se tendrá un comportamiento de segundo orden, de la siguiente manera:

$$\frac{sQ_i(s)}{\tau_i(s)} = \frac{k_p}{\tau s + 1} \tag{3.142}$$

Siendo las ecuaciones identificadas para cada articulaciôn.

$$\frac{sQ_1(s)}{\tau_1(s)} = \frac{2{,}25}{2{,}3419s + 1} \tag{3.143}$$

$$\frac{sQ_2(s)}{\tau_2(s)} = \frac{2{,}626}{1{,}439s + 1} \tag{3.144}$$

$$\frac{sQ_3(s)}{\tau_3(s)} = \frac{2{,}809}{1{,}234s + 1} \tag{3.145}$$

De manera que el factor de ajuste será el mismo para todas las articulaciones, para evitar la existencia de vibraciones ocasionadas por un distinto tiempo de respuesta, y acotadas, mediante la selección de los actuadores utilizados. El controlador resultante para este sistema consiste en un PD, denotando las ganancias, ajustadas a un valor de $\lambda = 10 * ts$.

Articulación	P	D
1	0.887	2.08
2	1.9	2.74
3	1.78	2.21

Cuadro 3.4: Valores de sintonía para las articulaciones del robot virtual

Cabe mencionar que este tipo de síntesis de controladores, permite generar controladores estables en base a variaciones de fase y cambios en la ganancia para J1 3.38, J2 3.39 y J3 3.40.

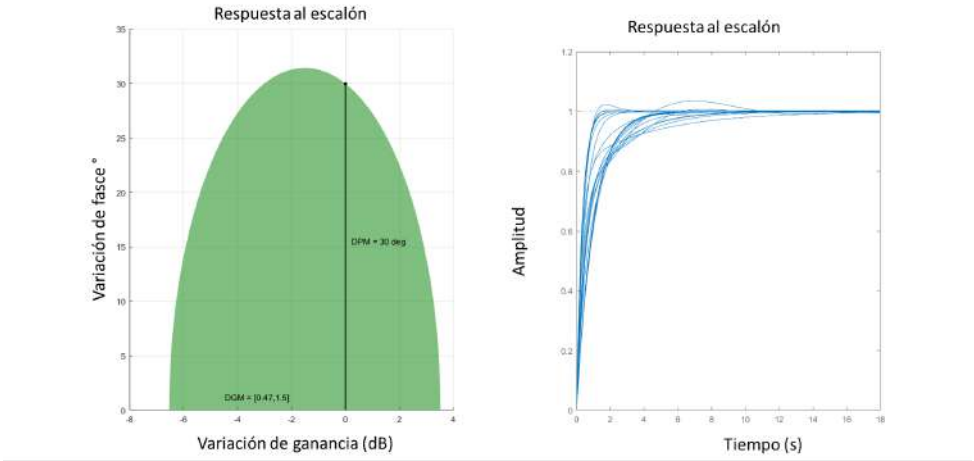

Figura 3.38: Respuesta del controlador de la articulaciòn 1 ante variaciones de fase y ganancia

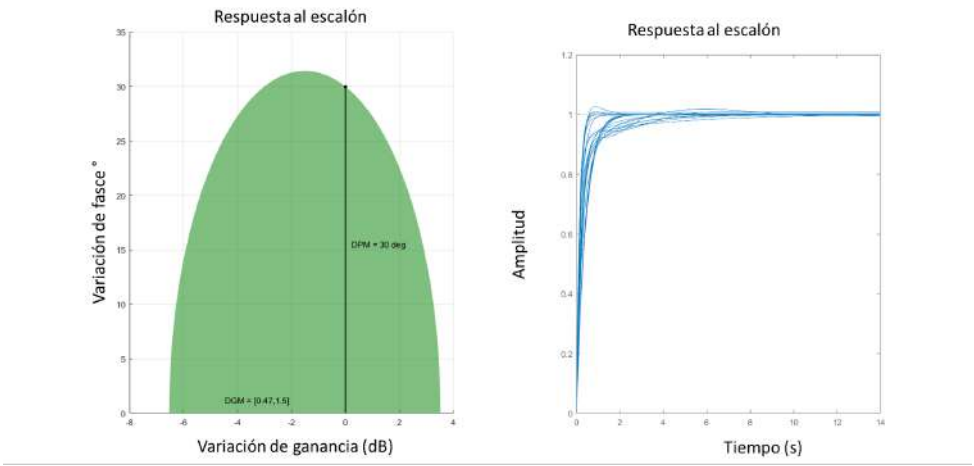

Figura 3.39: Respuesta del controlador de la articulaciòn 2 ante variaciones de fase y ganancia

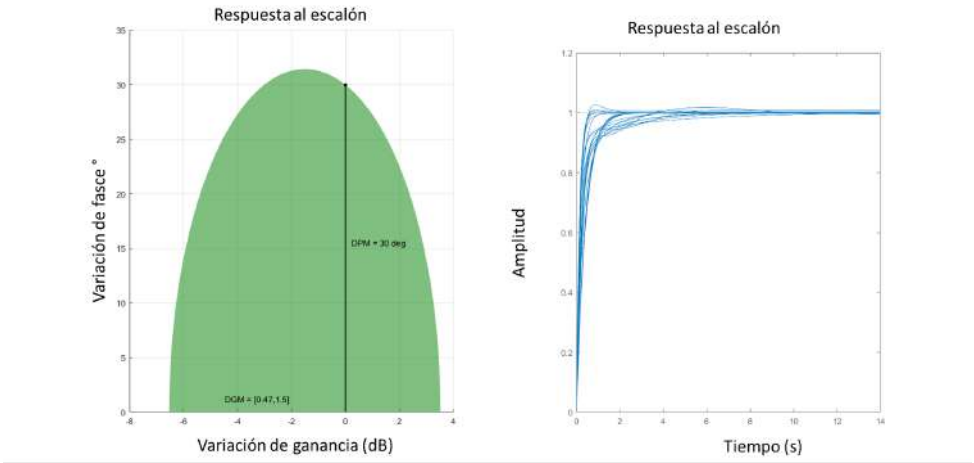

Figura 3.40: Respuesta del controlador de la articulaciòn 3 ante variaciones de fase y ganancia

Dicha implementación del control de las articulaciones en Unity 3D puede encontrarse en el anexo I. Sin olvidar, que este método precisa de la compensación de la gravedad, para funcionar correctamente, es decir que la señal de control sea $u = PD + Gravedad(q)$

3.4.3. Interacción con las piezas para la aplicación Pick and Place

Para generar el entorno virtual de la aplicación pick and place, es necesario, generar las diferentes piezas que se involucran en el proceso, atadas a una lógica que permita, así mismo, evitar posiciones inalcanzables para el robot virtual modelado. Las consideraciones más importantes rádican en selecionar de manera aleatoria una po-

sición del robot SCARA, lo cual se puede realizar calculando la cinemática directa en base a un estado de las articulaciones aleatorio, recordando. Este proceso se puede resumir en el diagrama de flujo 3.41.

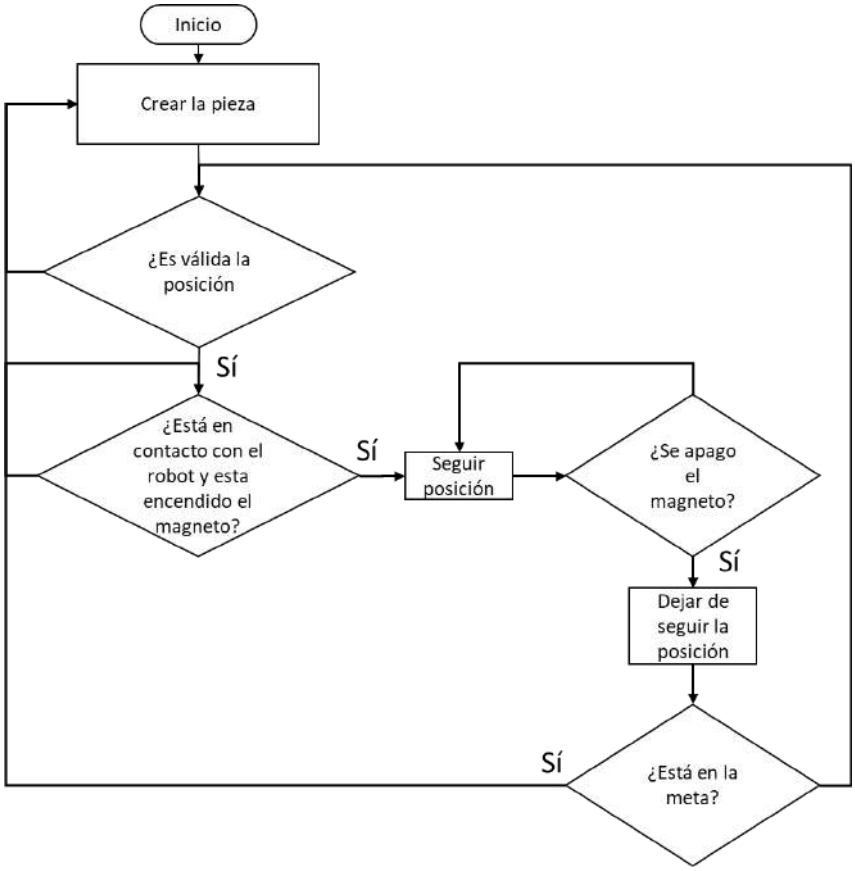

Figura 3.41: Maquina de estado de las piezas del entorno virtual

Esto permite generar el entorno dinámico 3.42 para la prueba del sistema completo, y así mismo, prevenir que las piezas sean arrojadas a una posición inalcanzable por el robot virtual.

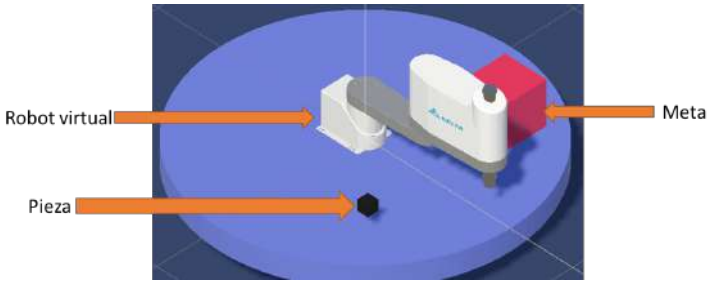

Figura 3.42: Entorno virtual con el robot SCARA y su interacción con las piezas

3.4.4. Control para una aplicación Pick and Place

El robot SCARA diseñado, requiere tener una secuencia que le permita generar la tarea de pick and place de manera continua, sin ninguna complicación, siempre y cuando, dicha pieza se encuentre dentro del espacio de configuraciones, expresada en el flujo 3.43.

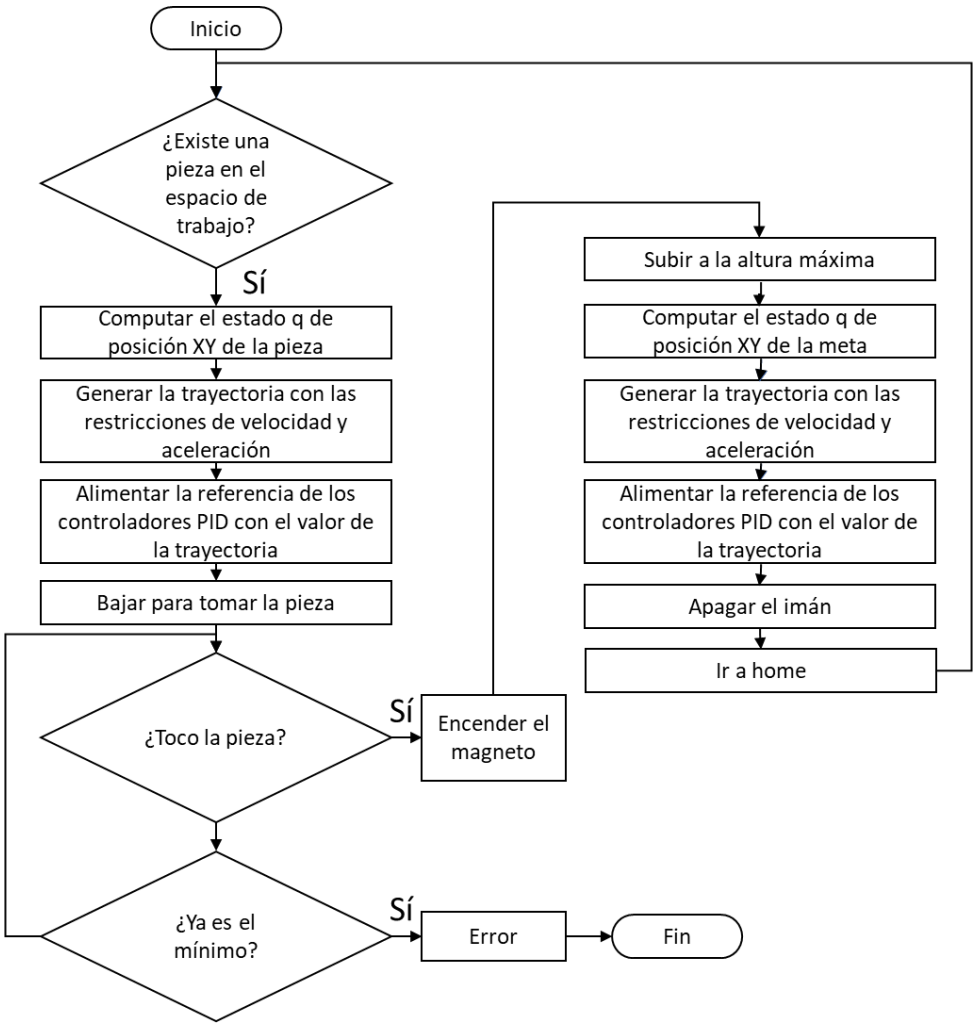

Figura 3.43: Diagrama de flujo del control Scara para aplicaciones pick and place

Simulación HIL

Para la simulación HIL, es necesario establecer un medio de comunicación entre el robot virtual y los controladores externos, en este proyecto se usa el protocolo de comunicación industrial Modbus en su modalidad TCP/IP.

3.5.1. Comunicación usando el protocolo Modbus

Usando el protocolo de comuncación Modbus en su versión TCP/IP, es necesario, evaluar el tiempo de respuesta del mismo, para evitar problemas en el entorno de la simulación [66]. Problema generado en la capa transferencia de datos según el modelo OSI [58]. Este análisis se realiza en base al flujo 3.44.

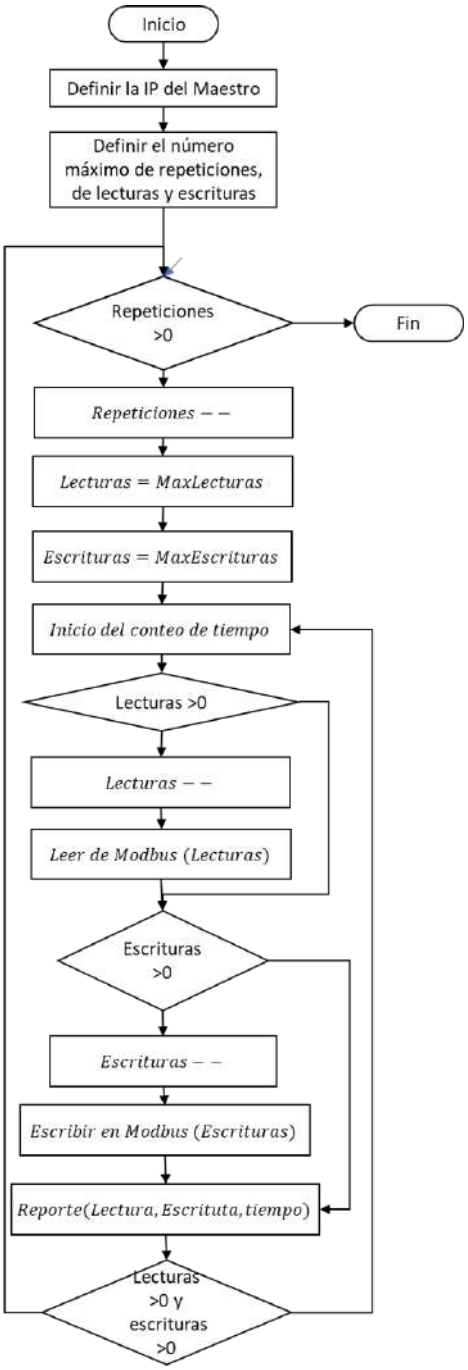

Figura 3.44: Diagrama de flujo para pruebas de velocidad en Modbus

Estas pruebas fueron realizadas sobre un PLC Siemens S7, por medio de solicitudes de lectura y escritura contra el tiempo de respuesta.

El comportamiento del protocolo Modbus 3.45, denota una especie de ruido causado por las condiciones de la red, la velocidad de procesamiento del PLC, y otros factores en el equipo de computo utilizados para la prueba:

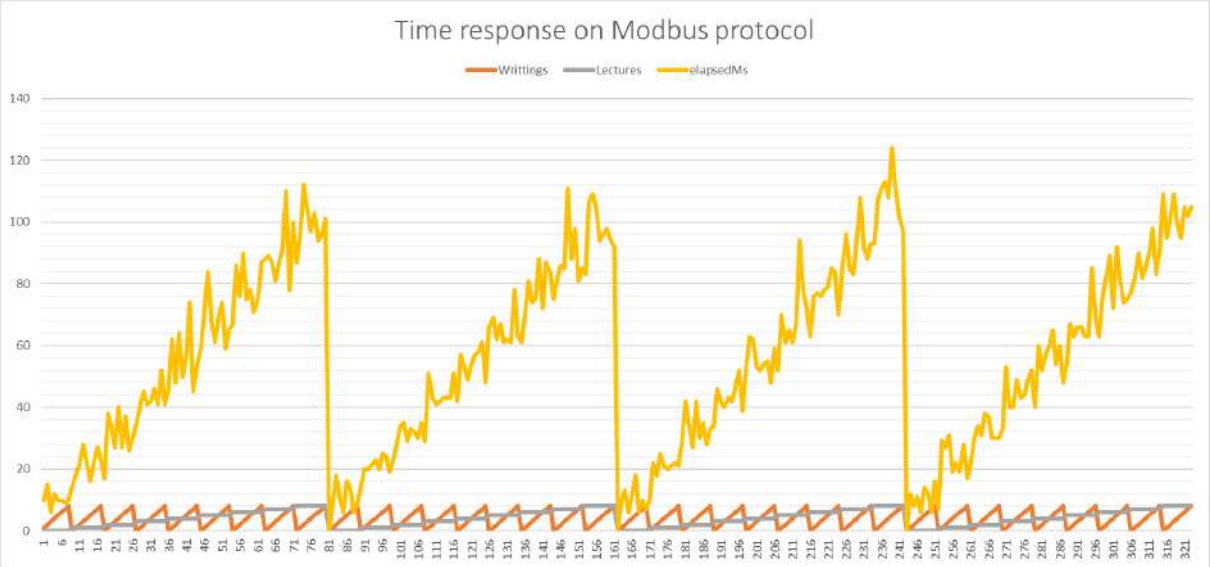

Figura 3.45: Tiempos de respuesta del protocolo Modbus

Es posible determinar una relación lineal entre el número de lecturas, y el número de lecturas:

$$tiempo_{ms} = 6{,}348 + 11{,}465 * Lecturas + 0{,}698 * Escrituras \tag{3.146}$$

Recordando que el tiempo de muestro debe de ser de al menos 2 veces la constante del tiempo del sistema.

3.5.2. Arquitectura para la simulación HIL

La simulación HIL de este proyecto se basa en la arquitectura 3.46:

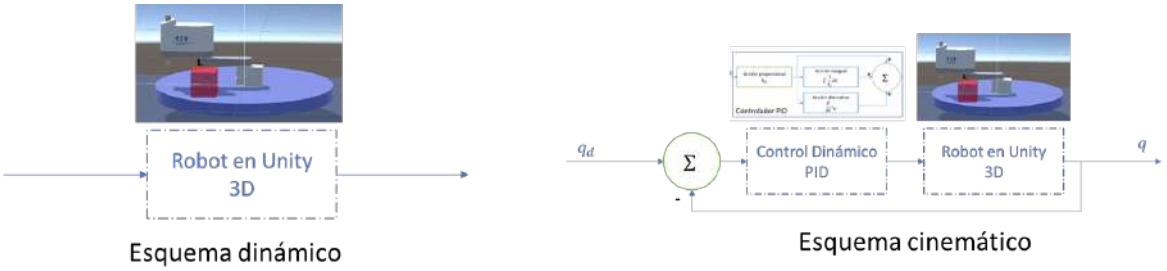

Figura 3.46: Esquemas de simulación HIL

Donde la comunicación por medio del protocolo Modbus, se ha organizado en base a la tabla 3.5.2, cabe mencionar que únicamente se han uso de los Holding Registers siendo que son de lectura y escritura por el maestro.

Dirección	Tipo	Descripción	Equivalencia	Escritura/Lectura
1	Holding register	Contador	valor	Escritura
2	Holding register	q1	valor*0.01 °	
3	Holding register	q2	valor*0.01 °	
4	Holding register	q3	valor*1mm	
5	Holding register	Pieza X	valor*1mm	
6	Holding register	Pieza Y	valor*1mm	
7	Holding register	Meta X	valor*1mm	
8	Holding register	Meta Y	valor*1mm	
9	Holding register	S1	valor*0.01	Lectura
10	Holding register	S2	valor*0.01	
11	Holding register	S3	valor*0.001	
12	Holding register	Magneto	valor!=0	

Cuadro 3.5: Registros usados en la simulación HIL

En el caso de S1, S2, S3, dependen de la modalidad que se requiera, si se usa como servocontrolado son:

$$q_{1d} = S1 * 0{,}01 \tag{3.147}$$

$$q_{2d} = S2 * 0{,}01 \tag{3.148}$$

$$q_{3d} = S3 * 0{,}001 \tag{3.149}$$

En el caso de que se opte por el modelo dinámico, se compone de la siguiente estructura:

$$\tau_1 = S1 * 0{,}01 \tag{3.150}$$

$$\tau_2 = S2 * 0{,}01 \tag{3.151}$$

$$\tau_3 = S3 * 0{,}001 \tag{3.152}$$

Un ejemplo de comunicación usando labview y Unity 3D puede ser apreciado en la figura 3.47

Figura 3.47: Prueba de comunicación Unity-Labview (Unity)

Siguiendo el flujo 3.48 en labview para la comunicación de la siguiente manera:

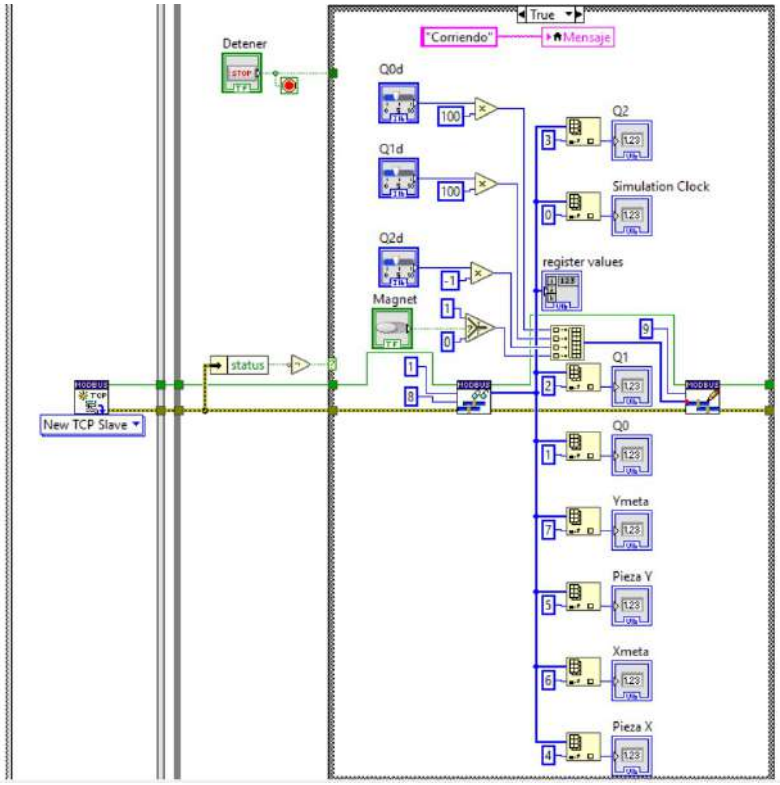

Figura 3.48: Prueba de comunicación Unity-Labview (Labview)

De manera similar puede implementarse en cualquier plataforma que permita el uso de Modbus.

Control mediante realidad aumentada del gemelo digital

Cambiando el control de un controlador externo por realidad aumentada, tiene la finalidad de volver al sistema controlable mediante marcadores de realidad aumentada, con el fin de proveer una simulación a nivel de aplicación del sistema. Al generar las posiciones iniciales y finales mediante la realidad aumentada, permite que el sistema sea adaptable a diferentes espacios, mientras que utilizar marcadores de tipo cúbico, permiten su visualización desde diferentes ángulos y una manipulación más cómoda que mover un marcador 2D. A través de la librería de Vuforia, es posible definir los marcadores utilizados en este trabajo. La creación de la base de datos se ilustra en 3.49

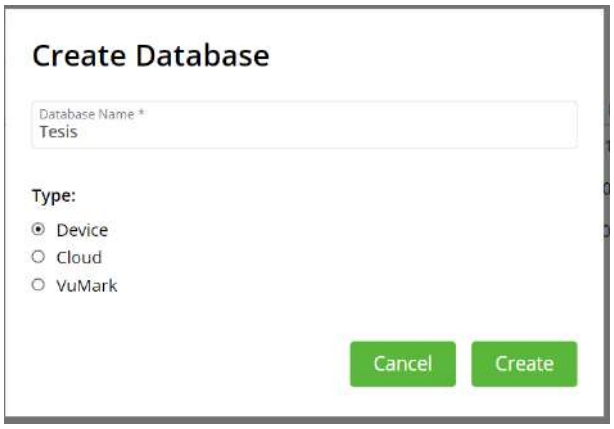

Figura 3.49: Creación de la base de datos de marcadores

Iniciando con la creación del marcador del robot 3.50, basado en un marcador multi objetivo de diez centímetros por lado. Donde se establecen las imágenes que conforman el objetivo, por lado y cada cara, cuidando el orden para evitar que se tengan problemas de detección posteriormente.

Figura 3.50: Creación de un marcador RA, robot

Mientras que la pieza objetivo 3.51, siendo de 5cm x 5cm, queda como:

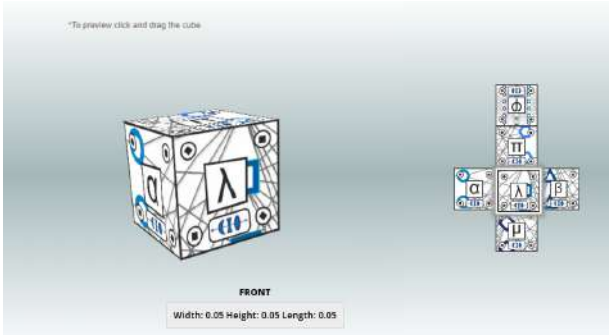

Figura 3.51: Creación de un marcador RA, objetivo

La construcción de los marcadores de manera física 3.52 se realizo a través de una impresión 3D del cubo, y posteriormente se cubrieron con impresiones de etiquetas en cada una de las caras respetando el orden definido en Vuforia.

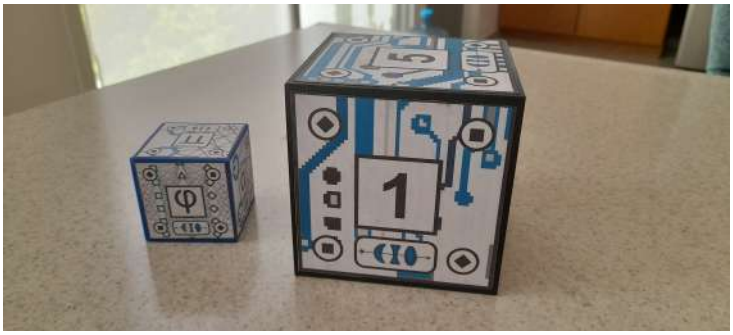

Figura 3.52: Marcadores RA construidos

3.6.1. Conexión del robot 3D en Unity con Realidad Aumentada

Usando la librería de Vuforia, luego de definir los marcadores, se descargo la base de datos 3.53.

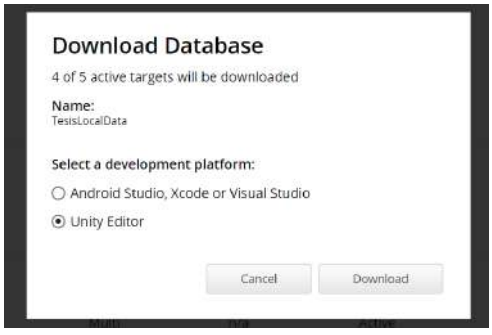

Figura 3.53: Descarga de la base de datos

Posteriormente se importa a Unity 3D, y el modelo 3D respecto a su correspondiente marcador de realidad aumentada 3.54.

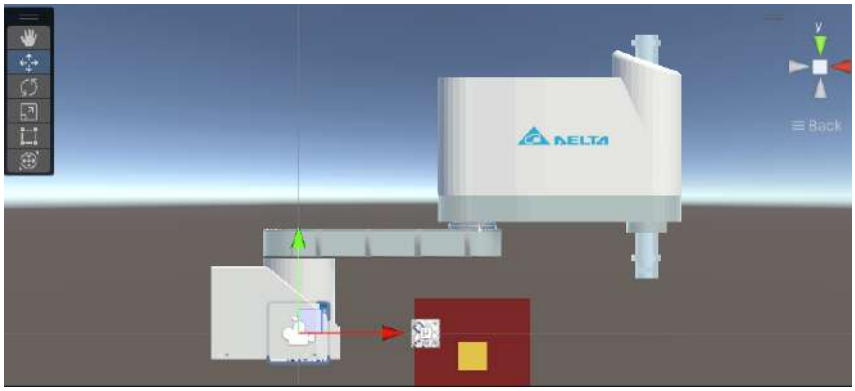

Figura 3.54: Referencia del robot 3D con respecto al marcador de realidad aumentada

Una vez establecida la relación entre el marcador y el modelo virtual del robot, es posible generar la siguiente interface como pruebas de movimiento en el espacio de configuraciones del robot 3.55:

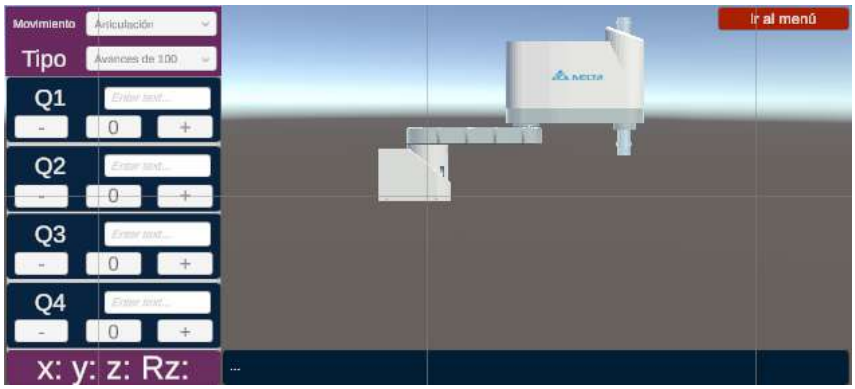

Figura 3.55: Interfaz de pruebas cinemáticas con realidad aumentada

A continuación, se muestra una prueba de la detección del marcador y el posicionamiento del modelo 3D sobre el mismo 3.56

Figura 3.56: Pruebas de detección y posicionamiento del robot con realidad aumentada

Mientras que para el control de la posición en el espacio cartesiano se incorpora el apartado de un selector intercambiar entre ambos modos de operación, esto con el fin de verificar el correcto funcionamiento de la cinemática inversa, y el movimiento en el espacio de trabajo. Este cambio se realiza por medio de una función desarrollada en el script de “ControlConTeaching.cs”, código que se encuentra en anexo H.

3.6.2. Ajuste de la posición en base a los marcadores de realidad aumentada

Sea una posición detectada por el marcador 3D, de manera que posee:

$$P = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \tag{3.153}$$

De manera que al detectar el robot y el objetivo de realidad aumentada se generan dos posiciones 3.57, una llamada ${}^{0}P_{robot}$ y otra llamada ${}^{0}P_{objetivo}$, sin embargo, es necesario que la base de ${}^{0}P_{objetivo}$ sea proyectada en base a P_{robot}.

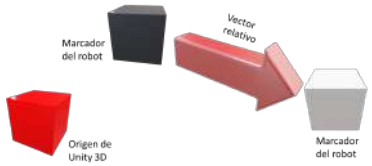

Figura 3.57: Posiciones detectadas con realidad aumentada

- Vector relativo entre el objetivo y el robot:

$$v_r = P_{objetivo}.T - P_{robot} \tag{3.154}$$

- Cambio de base:

$$Ajuste = P_{robot}.R^{-1} * v_r \tag{3.155}$$

- Definir la posición en base al robot:

$${}^{robot}P_{objetivo} = P = \begin{bmatrix} eye(3) & Ajuste \\ 0 & 1 \end{bmatrix} \tag{3.156}$$

Una implementación de esta transformación puede ser encontrada en los anexos, como “TransformadaDeEspacio.m”.

3.6.3. Flujo de control del robot virtual para aplicaciones Pick and Place

Cuando es posible llevar la pieza del espacio de Unity 3D detectada por medio de realidad aumentada al espacio de coordenadas del sistema base del robot, es posible expresar, el flujo de control requerido 3.58 para estas aplicaciones.

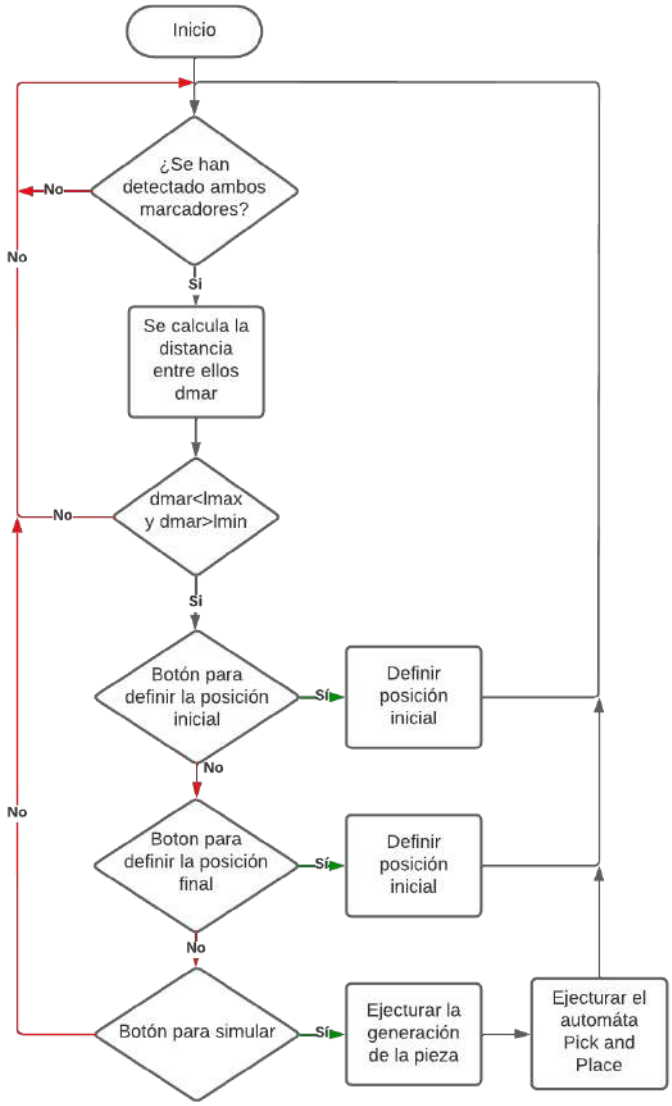

Figura 3.58: Diagrama de flujo de la aplicación Pick and Place usando realidad aumentada

4 Resultados

Integración de Robot Virtual SCARA

El modelado del robot SCARA para la aplicación, converge en un modelo digital siguiendo el flujo 4.1

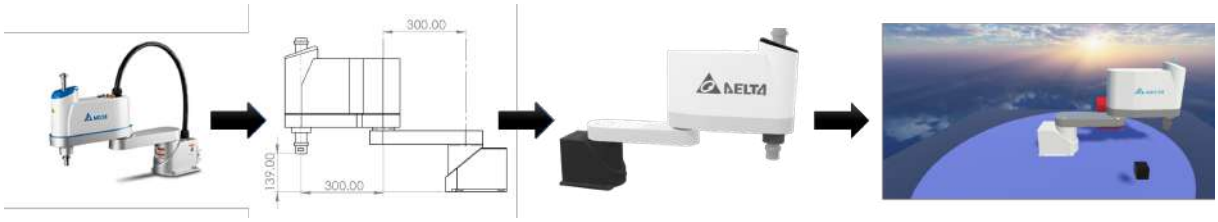

Figura 4.1: Secuencia de diseño

Donde se parte del modelado tridimensional del robot, ajustando las posiciones de movimiento mediante un elemento de pivote, así mismo, al encontrarse en Unity 3D, las ecuaciones de cinemática de cinemática directa 3.3 e inversa 3.12,3.15,3.14 permiten conocer con precisión la posición final del efector final, y así mismo, solicitar movimientos al sistema robotico 4.2.

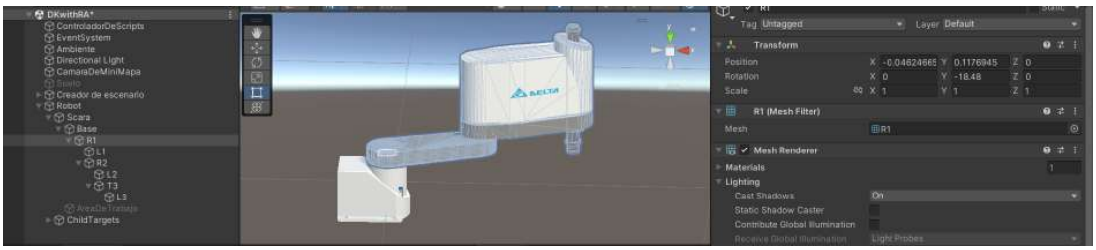

Figura 4.2: Ejemplo de movimiento del Robot SCARA en Unity

De manera estructurada, tambien se involucrá el modelado dinámico antes discutido mediante el algoritmo CRB, el cual, junto con el método de integración de Runge-Kutta permite generar una aproximación bastante cercana a la dinámica inversa. Esto se puede observar en la gráfica 4.3.

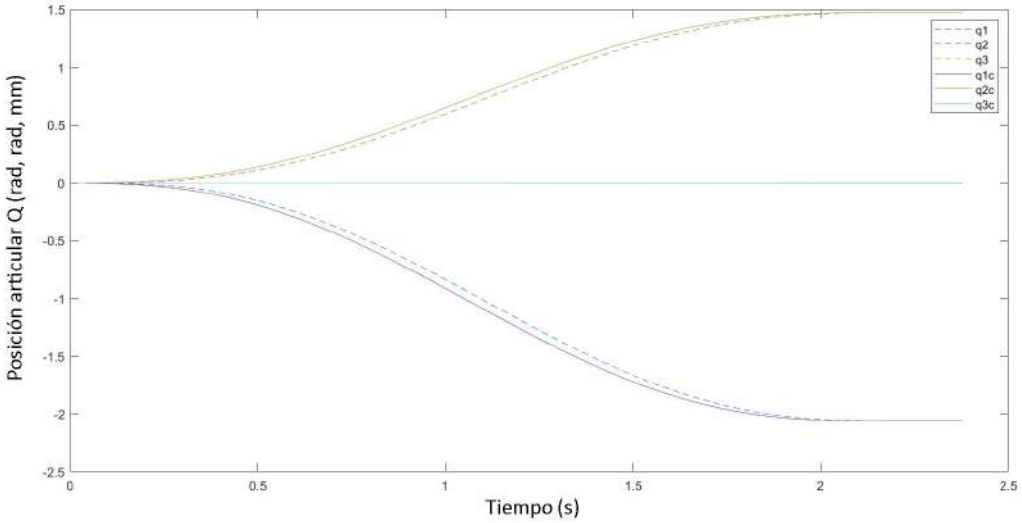

Figura 4.3: Diferencias entre la dinámica directa e inversa en el modelo optimizado

El modelo dinámico propuesto 3.21 se ha construido mediante el toolbox de Unity para robótica 4.4 , utilizando los componentes de física, los “Articulated body”:

Figura 4.4: Armado del gemelo virtual con la librería de física de Unity 3D

Esta construcción del sistema dinámico usando la librería de robótica de Unity, permite que el robot interactué con el motor de físicas interno de Unity, el NVIDIA PhysX, el cual, permite resolver las colisiones, y fuerzas en el entorno virtual. Que será usado para simular la interacción del robot, con las piezas objetivo, siendo que ahora aprovechando el modelado dinámico que se tiene y el control propuesto, cada articulación es sujeta a una pequeña rampa de torque o fuerza, para identificar el componente de la matriz de inercia principal, y así mismo, visualizar los efectos de la fricción, que involucra Unity 3D, resultando en las siguientes funciones de transferencia:

$$\frac{Q_1(s)}{\tau_1(s)} = \frac{2{,}25}{s(2{,}3419s+1)} \tag{4.1}$$

$$\frac{Q_2(s)}{\tau_2(s)} = \frac{2{,}626}{s(1{,}439s+1)} \tag{4.2}$$

$$\frac{Q_3(s)}{\tau_3(s)} = \frac{2{,}809}{s(1{,}234s+1)} \tag{4.3}$$

Este comportamiento incluye la fricción viscosa de cada articulación. Expresiones que permiten utilizar la métodología IMC para determinar las ganancias necesarias para cada articulación, sin embargo, todas las articulaciones deben de compartir, la misma λ, para asegurar el buen funcionamiento del sistema. En la tabla 4.1 se ilustran las ganancias calculadas para el control.

Articulación	P	D
1	0.887	2.08
2	1.9	2.74
3	1.78	2.21

Cuadro 4.1: Valores de sintonía para las articulaciones del robot virtual

Con la finalidad de que las articulaciones del robot, puedan ser controladas en base a la trayectoria y movimientos deseados, se evalúa el desempeño de los controladores mediante la variación de la ganancia de la función de transferencia, y la fase, a continuación, se muestran los resultados de dicho análisis 4.5.

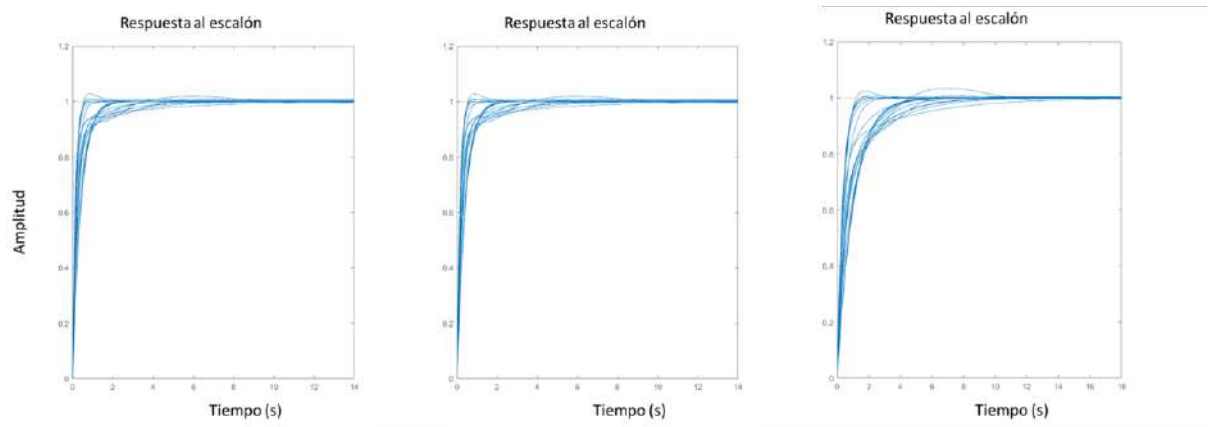

Figura 4.5: Control de las articulaciones en el robot virtual

La estrategia para el control del robot, se basa en el diagrama de control 4.6, el control de la secuencia permite definir las posiciones requeridas para la elaboración de la tarea, mientras que el controlador cinemático genera los valores necesarios en las articulaciones para llegar a dicho punto, mientras que el planeador de trayectorias, permite establecer la trayectoria para llegar al punto deseado en base a las restricciones de aceleración y velocidad máxima, dando la alimentación al control dinámico para realizar los movimientos que serán simulados por parte del robot y el entorno virtual diseñado.

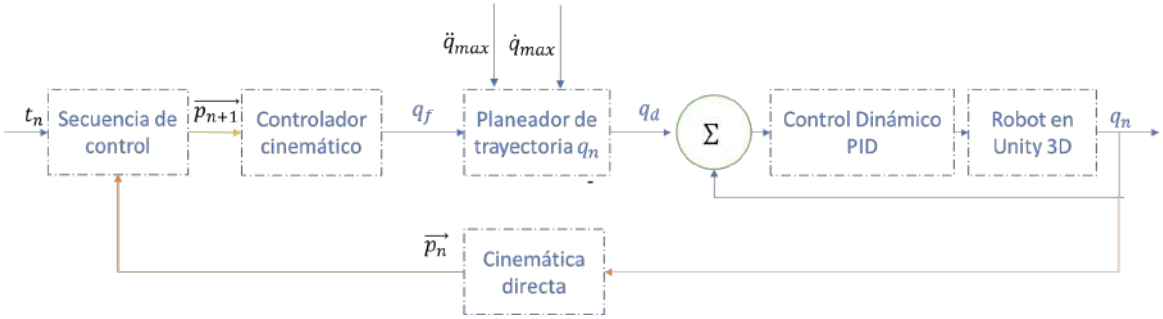

Figura 4.6: Diagrama de control del robot virtual

Permitiendo que el robot virtual pueda interactuar con el ambiente virtual generado para aplicaciones de pick and place 4.7.

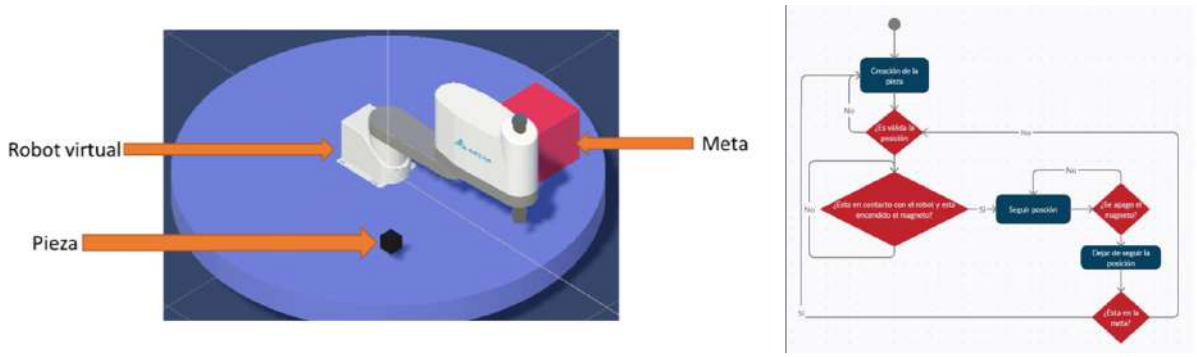

Figura 4.7: Entorno virtual de la aplicación Pick and Place

Simulador para la educación

El control del robot virtual, se define a distintos niveles para la interacción con los usuario 4.8, con el fin de observar los distintos niveles de control, que son involucrados en cada una de las etapas de los sistemas robóticos.

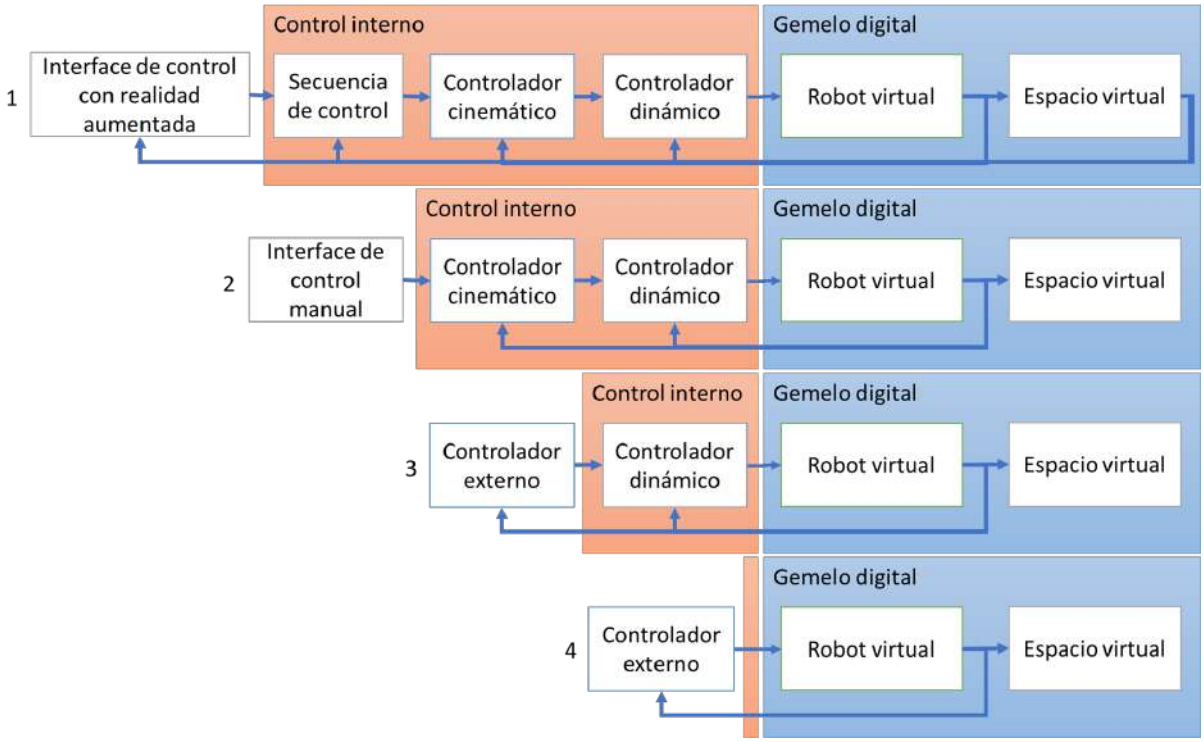

Figura 4.8: Niveles de simulación en la aplicación creada

Simulación a nivel de aplicación

El primer nivel representa la simulación de una aplicación pick and place 4.9, y su control 3.58 por medio de los marcadores de realidad aumentada 3.52.

Figura 4.9: Control del robot virtual por medio de los marcadores de realidad aumentada

En la interface 4.9 se pueden apreciar los botones correspondientes a la definición, de las posiciones iniciales y la posición de la meta, así como el botón inferior para simular el proceso en el robot virtual, según el flujo 3.58, para el control del robot

virtual.

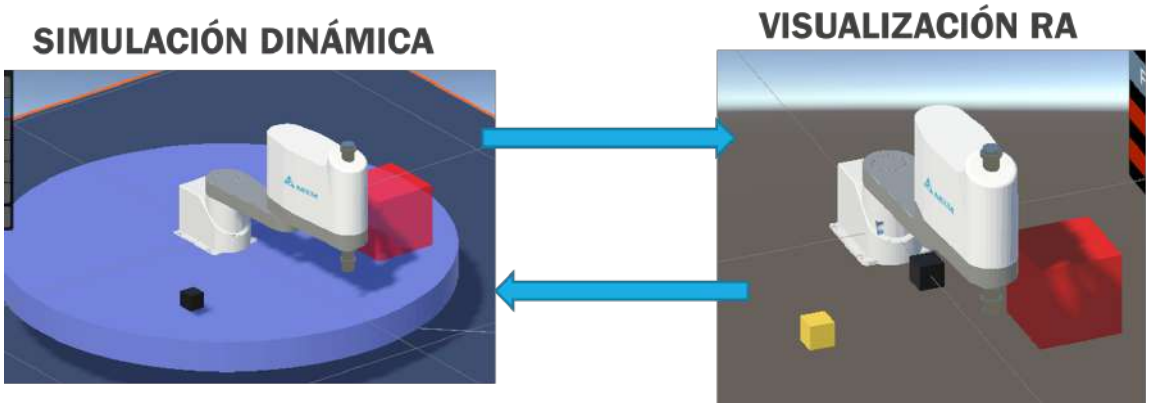

Figura 4.10: Enlace entre la simulación dinámica y la visualización de realidad aumentada

Para evitar que la varianza en la detección de los marcadores afecten la simulación dinámica al introducir ruido por medio de variaciones en la velocidad de las articulaciones, se divide en dos, permitiendo que el modelo de realidad aumentada sea únicamente un espejo de la simulación dinámica 4.10.

4.2.1. Simulación a nivel de cinemático con realidad aumentada

Este planteamiento permite simular la cadena cinemática y su solución inversa a través de la realidad aumentada, con el fin de brindar una visión del movimiento del robot SCARA, este únicamente 4.11.

Figura 4.11: Simulación cinemática a travéz de la realidad aumentada

4.2.2. Simulación HIL con controladores externos

En este nivel las señales de control del robot SCARA son obtenidas por medio de un controlador externo, con la finalidad de simular el funcionamiento que se obtiene al proponer una estrategia de control personalizada tanto a nivel cinemático como a nivel dinámico, permitiendo afrontar los dos problemas de control que se tienen en los sistemas robóticos, así mismo, considerando la posibilidad de simular las colisiones, efectos de un mal planteamiento de control y de secuenciamiento, en las estaciones de transferencia 4.12.

Figura 4.12: Estación de simulación HIL

Permitiendo que la plataforma de Modbus está abierta a una gran variedad, de sistemas tanto industriales como de simulación, es posible realizar, la comunicación del simulador con entornos como:

- Labview
- Matlab
- PLC
- Microcontroladores
- Tarjetas de adquisición de datos

El entorno virtual junto con el robot virtual, permite simular:

- La aplicación pick and place.
- Las Colisiciones
- El comportamiento dinámico en tiempo real

- El comportamiento cinemático
- El movimiento del modelo 3d

De manera que la conexión de entornos virtuales con esquemas de simulación como lo es “HIL”, permite depurar aplicaciones de robótica, que aparte de ser gemelos digitales, permiten evaluar, un componente crítico de los sistemas robóticos, el controlador.

5 Conclusiones

- La conexión del robot virtual con otras plataformas de control, permite evaluar las condiciones de control, y de desempeño del mismo, partiendo a diferentes niveles de complejidad, genera un flujo de trabajo para simualciones similares aprovechando el motor de físicas de Unity 3D.

- Este robot, permite la capacitación de manera inicial a temas de robotica, tanto automatizado el proceso de una aplicación pick and place, hasta considerar la conexión de un controlador externo para desarollar la lógica de control, esto con la finalidad de ser una herramienta de capacitación, que aprovecha el uso de las capacidades de gráficas de un motor de juegos.

- El uso de marcadores de realidad aumentada que sean cúbicos para el control de sistemas robóticos, presentan ventajas de manipulación y de rango de visión a comparartiva de los marcadores tradicionales. Esto se debe al tamaño y la facilidad con la que pueden ser cambiados de posición, y que pueden deterctarse, desde diferentes angulos.

- En el caso de la simulación HIL, es importante mencionar que el hilo de la comunicación debe de ser aislado del proceso de la simulación minimizando los efectos de las perturbaciones en los procesos de comunicación y evitando retrasos y errores en la simulación, donde únicamente se toman los ultimos datos adquiridos, cuando se realiza el cálculo de la dinámica actual.

- La generación de las piezas en base al método Monte-Carlo, permite generar una mayor confianza en la estrategia de control planteada mientra, mientras se aprovecha el entorno simulado.

- El planteamiento de la dinámica directa, una vez establecida la dinámica inversa, se puede utilizar para computar de manera rápida las ecuaciones de dinámica

directa por medio del algoritmo “CRBA”, y se le puede adicionar la fricción, utilizando el planteamiento de la ecuación de Rayleigh.

- El control por medio de la matriz principal de la matriz de inercia, se precisa como una manera de control simple y robusta, siempre y cuando los factores de la gravedad sean compensados. Esta compensación, puede ser planteada por medio del algoritmo de Newton-Euler.
- Este sistema fue implementado en la red de laboratorios de la CONACYT, para las capacitaciones en robótica y temas afines.

6 Trabajo futuro

Este sistema puede constituirse en base a la construcción de sistemas y simuladores para la educación de la robótica, así mismo, acelerar el desarrollo de sistemas robóticos mediante la incorporación de un entorno virtual para técnicas HIL que permitan dictaminar la eficciencia del control propuesto. Cabe mencionar que incluir la modelación de los actuadores permitiría un acercamiento más profundo y preciso en el modelado de sistemas robóticos, al evaluar el consumo energetico, requerimientos de los controladores, y así mismo, datos de utilidad para la industrial 4.0.

El control propuesto en este trabajo para el control de la dinámica del sistema no se escapa a que pueda ser implementada de manera adaptativa en base a los cambios de la matriz de inercia y usando la métodología IMC, para recalcular los valores de los controladores, sin embargo, la síntesis de los controladores en el punto de equilibrio seleccionado, ha permitido generar un controlador con un desempeño aceptable.

El modelado de otros sistemas robóticos utilizando Unity 3D puede ser interesante, por la conexión que se puede establecer con el motor de físicas interno, evaluando, posibles escenarios de colisión de aplicaciones más complejas.

Actualmente la literatura y los programas de diseño tienen un espacio de oportunidad entre el modelado dinámico inverso de los robots, y la dinámica directa, para el modelado de sistemas robóticos, que efectivamente, resultan en aplicaciones interesantes de la industria 4.0, como lo es, la previsión de seguridad en espacios industriales, el desempeño real de un sistema robótico en determinada situación, o bien, de manera interesante, para temas de mantenimiento predictivo.

7 Anexos

A. Restricción del espacio de trabajo implementado en Python

```
Distancia=math.sqrt(xT**2+yT**2)
if(Distancia>=0.1435 and Distancia<=0.6):
    if xT<0:
        if np.abs(yT)>100:
            return true
        else:
            altura=438.81
            distancia=409.2
            m=altura/distancia
            ecu=altura+xT*m
            if np.abs(yT)>ecu:
                return true
    else:
        return true
return false
```

B. Trayectoria de Jerk Mínimo con restricciones de velocidad y aceleración en MATLAB

```
maxAce=10
maxVel=100
initial=0
final=10
scalingTime=1;
mV=1.8750;
mA=5.7735;
delta=final-initial;
if(maxVel<mV*delta)
    scalingTime=mV*delta/maxVel
end
endTime=scalingTime;
time=0:0.01:scalingTime;
path=minJerk(time,endTime,initial,final);
sup=max(path(3,:))
while(maxAce<sup)
    scalingTime=1/100+scalingTime;
    endTime=scalingTime;
    time=0:0.01:scalingTime;
    path=minJerk(time,endTime,initial,final);
    sup=max(path(3,:));
end
endTime=scalingTime;
time=0:0.01:scalingTime;
path=minJerk(time,endTime,initial,final);
plot(time,path(1,:))
hold on
```

```
plot(time,path(2,:))
plot(time,path(3,:))
legend('q','qd','qdd')
max(path(3,:))
hold off
```

C. Método de Newton-Rapshon para la solución de la cinemática inversa en MATLAB

```
%DkSCARA(l1,l2,l3,theta1,theta2,theta3,theta4)
%JSCARA(l2,l3,theta1,theta2)
state=[0;0;0;0];
wanted=[0.2;0.3;0;0]
li1=0.139
li2=0.2
li3=0.3
tol=1e-3
maxN=20;
errorSec=zeros(1,maxN);
actual=D_SCARA(li1,li2,li3,state(1),state(2),state(3),state(4));
record=[];
for i =1:maxN
    actual=D_SCARA(li1,li2,li3,state(1),state(2),state(3),state(4));
    if(i==1)
        record=actual;
    else
        record=[record actual];
    end
    MiError=wanted-actual;
    state=state+pinv(J_SCARA(li2,li3,state(1),state(2)))*MiError;
    actualE=norm(MiError);
    errorSec(i)=actualE;
    if(i==maxN)
        'No resuelto'
    end
    if(actualE<tol)
```

```
        'Resuelto'
        break
    end
end
plot(errorSec)
plot(record')
actual
state
hold off
```

D. Código de la simulación de la dinámica directa simplificada con el control propuesto en MATLAB

```
%Valores de configuración
l=[0.139,0.3,0.3];
m=[1,1,1];
b=[0.1,0.1,0.1];
experience=1;
ts = 0.02;
rng(10)
%Variables de guardado
Data=[]
Real=[]
%Estado de inicio q0
initial=[0,0,0];
IndexOfTrim=[];
s1=[rand(1,1)*deg2rad(153)*sign(rand(1,1)-0.5),...
rand(1,1)*deg2rad(133)*sign(rand(1,1)-0.5),...
rand(1,1)*0.2];
tr=10+10*ts;
LastErrorPosicion=[0;0;0];
for i=1:experience
    final=[rand(1,1)*deg2rad(153)*sign(rand(1,1)-0.5),...
    rand(1,1)*deg2rad(133)*sign(rand(1,1)-0.5),...
    rand(1,1)*0.2];
    %Limpieza de variables para un nuevo movimiento
    timeR=tr;
    rep=floor((timeR+10*ts)/ts);
    actualTime=0;
    ace=zeros(rep,length(initial));
```

```
    acec=zeros(rep,length(initial));
    torque=zeros(rep,length(initial));
    pos=zeros(rep,length(initial));
    vel=zeros(rep,length(initial));
    ace=zeros(rep,length(initial));
    Pxyz=zeros(rep,3);
    Pxyzc=zeros(rep,3);
    states=zeros(rep,length(initial)*2);
    deltaState=zeros(rep,length(initial)*2);
    states(1,:)=[initial,zeros(1,3)]';
    for j=1:rep
%       Generación de trayectoria
        [p,b,c]=inter5(initial,final,timeR,actualTime);
        yw=[p,b]';
        pos(j,:)=p;
        vel(j,:)=b; %Delta position
        ace(j,:)=c; %Delta speed
        wantedP=[p(1),p(2),p(3),0];
        [x,py,z,~]=SCARAFK(wantedP,1);
        Pxyz(j,:)=[x,py,z];
%       Estado anterior
        if(j>1)
            LastState=states(j-1,:);
        else
            LastState=states(j,:);
        end
%       cálculode M(q) y la fricción
        M=massMatrixSCARA(l2,l3,m1,m2,m3,LastState(2));
%       Fricción
        Fv=LastState(4:6)'.*b';
        Des=pinv(M)*(-Fv);
%       Control de Posición
        errorEnPosicion=pos(j,:)'-LastState(1:3)';
        deltaPosicion=(pos(j,:)'-LastState(1:3)')/ts;
```

```
        deltaPosicion(3)=0;
        Uspeed=Pc.*errorEnPosicion+Pd.*deltaPosicion;
        LastErrorPosicion=errorEnPosicion;
%       Control de Velocidad
        u=VelP.*(Uspeed-deltaPosicion);
%       compensación de gravedad
        u(3)=9.81+u(3);
        inputTorque=u;
        torque(j,:)=inputTorque;
%       CRB Algorithm
        acec(j,:)=rneDirectSCARA(-9.81,l2,l3,m1,m2,m3,...,
        u(1),u(2),u(3),LastState(2),...,
        LastState(4),LastState(5))+Des;
%       Limite de aceleración máxima
        maxAcel=4*9.81;
        acec(j,:)=(abs(acec(j,:))>maxAcel).*sign(acec(j,:))*maxAcel+...
        (abs(acec(j,:))<maxAcel).*acec(j,:);
        %acec(j,:)=robot.forwardDynamics([0,0,0]',[0,0,0]',inputTorque)+Des;
        states(j,:)=LastState+[LastState(4:6),acec(j,:)]*ts;
%       Computo de la posición final
        state=[states(j,1),states(j,2),states(j,3),0];
        [x,py,z,w]=SCARAFK(state,l);
        Pxyzc(j,:)=[x,py,z];
        actualTime=actualTime+ts;
    end
    initial=final;
    if(i==1)
        Data=[states acec deltaState];
        Real=[torque,pos,vel,ace];
        IndexOfTrim=[rep];
    else
        Data=[Data;states,acec deltaState];
        Real=[Real;torque,pos,vel,ace];
        IndexOfTrim=[IndexOfTrim,rep];
```

```
    end
    re=pos;
    co=states(:,1:3);
    diff=sum(sum(abs(re-co)./flip(1:size(states,1))',1));
    %[kp, kd,diff+sum(max(abs(co-re)))]

end
acec=acec*ts;
sizeOfView=size(Real,1);
timeV=1:size(Real,1);
timeV=timeV*ts;

plot(timeV,Real(:,4:6),'--',timeV,Data(:,1:3))
legend('q1','q2','q3','q1c','q2c','q3c')
% plot(timeV,Real(:,7:9),'--',timeV,Data(:,4:6))
% legend('qd1','qd2','qd3','qd1c','qd2c','qd3c')
% plot(timeV,Real(:,4:6)-Data(:,1:3))
% legend('eq1','eq2','eq3')
% plot(timeV,Real(:,3),timeV,Data(:,1:3))
% legend('t3')
% plot(timeV,Real(:,1:3))
% legend('t1','t2','t3')
```

E. CinematicaDirecta.cs

```
public class CinematicaDirecta : MonoBehaviour
{
    //Estado del robot SCARA
    [Range(-133,133)]
    public float Rot1 = 0;
    [Range(-153, 153)]
    public float Rot2 = 0;
    [Range(0, 0.2f)]
    public float Tra3 = 0;
    [Range(-360,360f)]
    public float Rot4 = 0;
    //Valores para D.H
    public float L1 = 0.139f;
    public float L2 = 0.3f;
    public float L3 = 0.3f;
    //Resultados de la Cinemática directa
    public Vector3 posicionXYZ = new Vector3();
    public Vector3 posicionAnterior;
    public Vector3 velocidadLineal;
    public Vector3 velocidadLinearAnterior;
    public Vector3 aceleracionLineal;
    public float Rz;

    void Update()
    {
        var fk = CalculateFk(new Vector4(Rot1, Rot2, Tra3, Rot4));
        posicionXYZ = new Vector3(
            fk.x,
```

```
            fk.y,
            fk.z
        );
        Rz = fk.w;
        velocidadLineal = (posicionXYZ - posicionAnterior) / Time.deltaTime;
        posicionAnterior = posicionXYZ;
        aceleracionLineal = (velocidadLineal - velocidadLinearAnterior) / Time.delt
        velocidadLinearAnterior = velocidadLineal;
    }
    public Vector4 CalculateFk(Vector4 currentState)
    {
        return new Vector4()
        {
            x = L2 * Mathf.Cos(currentState.x * Mathf.Deg2Rad) +
                L3 * Mathf.Cos(currentState.x * Mathf.Deg2Rad +
                currentState.y * Mathf.Deg2Rad),
            y = L2 * Mathf.Sin(currentState.x * Mathf.Deg2Rad) +
                L3 * Mathf.Sin(currentState.x * Mathf.Deg2Rad +
                currentState.y * Mathf.Deg2Rad),
            z = L1 - currentState.z,
            w = currentState.x + currentState.y + currentState.w
        };
    }
}
```

F. QtoUnity.cs

```
public class QtoUnity : MonoBehaviour
{
    [Header("GameObjects de articulaciones")]
    public Transform rot1;
    public Transform rot2;
    public Transform tra3;
    public Transform rot4;
    public Transform @base;
    [Header("Valores solicitados")]
    public CinematicaDirecta cinematica;
    public float r1 = 0;
    public float r2 = 0;
    public float t3 = 0;
    public float r4 = 0;
    public float alturaInicial = 0.139f;

    private void Start()
    {
        alturaInicial = tra3.localPosition.y;
    }
    void Update()
    {
        r1 = cinematica.Rot1*-1;
        r2 = cinematica.Rot2*-1;
        t3 = cinematica.Tra3;
        r4 = cinematica.Rot4;

        //Valores de la rotación 1
```

```
        var rotacion1 = rot1.localRotation.eulerAngles;
        rotacion1.y = r1;
        rot1.localRotation = Quaternion.Euler(rotacion1);
        //Valores de la rotación
        var rotacion2 = rot2.localRotation.eulerAngles;
        rotacion2.y = r2;
        rot2.localRotation = Quaternion.Euler(rotacion2);
        //Altura del tornillo sin fin
        var translacion3 = tra3.localPosition;
        translacion3.y = alturaInicial - t3;
        tra3.localPosition = translacion3;
        //Rotación de la cuarta articulacion
        if (rot4 == null) return;
        var rotacion4 = rot4.localRotation.eulerAngles;
        rotacion4.y = r4;
        rot4.localRotation = Quaternion.Euler(rotacion4);
    }
}
```

G. ServoControlador.cs

```
public class ServoControlador : MonoBehaviour
{
    public CinematicaDirecta robot;
    [Header("Estado solicitado")] [Range(-133, 133)]
    public float d1 = 0;
    [Range(-153, 153)] public float d2 = 0;
    [Range(0, 0.2f)] public float d3 = 0;
    [Range(-360, 360f)] public float d4 = 0;
    [Header("Configuración del tiempo de asentamiento")]
    public float q1Tao = 1;
    public float q2Tao = 1;
    public float t3Tao = 1;
    public float q4Tao = 1;
    private float _alfa1, _alfa2, _alfa3, _alfa4;
    private void Start()
    {
        CalculateAlphas();
    }
    private static float GetAlpha(float delta, float tao)
    {
        return tao / (delta + tao);
    }
    private void CalculateAlphas()
    {
        _alfa1 = GetAlpha(Time.deltaTime, q1Tao);
        _alfa2 = GetAlpha(Time.deltaTime, q2Tao);
        _alfa3 = GetAlpha(Time.deltaTime, t3Tao);
        _alfa4 = GetAlpha(Time.deltaTime, q4Tao);
```

```
    }
    private void CheckInput()
    {
        d1 = d1 switch
        {
            < -133 => -133,
            > 133 => 133,
            _ => d1
        };

        d2 = d2 switch
        {
            < -155 => -155,
            > 155 => 155,
            _ => d2
        };

        d3 = d3 switch
        {
            < 0 => 0,
            > 0.2f => 0.2f,
            _ => d3
        };

        d4 = d4 switch
        {
            < -360 => -360,
            > 360 => 360,
            _ => d4
        };
    }
    private void Update()
    {
        CalculateAlphas();
```

```
        CheckInput();
        robot.Rot1=robot.Rot1*_alfa1+d1*(1-_alfa1);
        robot.Rot2=robot.Rot2*_alfa2+d2*(1-_alfa2);
        robot.Tra3=robot.Tra3*_alfa3+d3*(1-_alfa3);
        robot.Rot4=robot.Rot4*_alfa4+d4*(1-_alfa4);
    }
    public Vector4 GetResultingState()
    {
        return robot.CalculateFk(new Vector4(d1,d2,d3,d4));
    }
    public bool IsDone(float tolerance)
    {
        return (Math.Abs(robot.Rot1 - d1) < tolerance
                && Math.Abs(robot.Rot2 - d2) < tolerance
                && Math.Abs(robot.Tra3 - d3) < tolerance
                && Math.Abs(robot.Rot4 - d4) < tolerance);
    }
}
```

H. ControlConTeaching.cs

```
public class ControlConTeaching : MonoBehaviour
{
    [Header("Tipos de movimientos")]
    public TMP_Dropdown tipoDeMovimiento;
    public TMP_Dropdown velocidadMovimiento;
    [Header("Q1")]
    public TMP_InputField q1;
    public TextMeshProUGUI q1Label;
    public Button q1Plus;
    public Button q1Cero;
    public Button q1Minus;
    [Header("Q2")]
    public TMP_InputField q2;
    public TextMeshProUGUI q2Label;
    public Button q2Plus;
    public Button q2Cero;
    public Button q2Minus;
    [Header("Q3")]
    public TMP_InputField q3;
    public TextMeshProUGUI q3Label;
    public Button q3Plus;
    public Button q3Cero;
    public Button q3Minus;
    [Header("Q4")]
    public Image tercerFactor;
    public TMP_InputField q4;
    public TextMeshProUGUI q4Label;
    public Button q4Plus;
```

```
public Button q4Cero;
public Button q4Minus;
[Header("Conexiones con el modelo")]
public ServoControlador servoScara;
public CinematicaDirecta FKScara;
public InverseScara IKScara = new InverseScara();
[Header("Mostrar los datos del robot")]
public TextMeshProUGUI mostradorDePosicion;
public TextMeshProUGUI mostradorDeArticulaciones;
public void SetAsJoinMode()
{
    q1Label.text = "Q1";
    q2Label.text = "Q2";
    q3Label.text = "Q3";
    q4Label.text = "Q4";
    q1.text = $"{FKScara.Rot1:000.00}";
    q2.text = $"{FKScara.Rot2:000.00}";
    q3.text = $"{FKScara.Tra3:000.00}";
    q4.text = $"{FKScara.Rot4:000.00}";
    q1Cero.gameObject.SetActive(true);
    q2Cero.gameObject.SetActive(true);
    q3Cero.gameObject.SetActive(true);
    q4Cero.gameObject.SetActive(true);
}
public void SetAsCartesianMode()
{
    q1Label.text = "X";
    q2Label.text = "Y";
    q3Label.text = "Z";
    q4Label.text = "Rz";
    q1.text = $"{FKScara.posicionXYZ.x:000.00}";
    q2.text = $"{FKScara.posicionXYZ.y:000.00}";
    q3.text = $"{FKScara.posicionXYZ.z:000.00}";
    q4.text = $"{FKScara.Rz:000.00}";
```

```
        q1Cero.gameObject.SetActive(false);
        q2Cero.gameObject.SetActive(false);
        q3Cero.gameObject.SetActive(false);
        q4Cero.gameObject.SetActive(false);

    }
    // Start is called before the first frame update
    void Start()
    {
        tipoDeMovimiento.onValueChanged.RemoveAllListeners();
        tipoDeMovimiento.onValueChanged.AddListener(CambioDeTipoDeMovimiento);
        q1Cero.onClick.RemoveAllListeners();
        q2Cero.onClick.RemoveAllListeners();
        q3Cero.onClick.RemoveAllListeners();
        q4Cero.onClick.RemoveAllListeners();
        q1Cero.onClick.AddListener(SetZeroQ1);
        q2Cero.onClick.AddListener(SetZeroQ2);
        q3Cero.onClick.AddListener(SetZeroQ3);
        q4Cero.onClick.AddListener(SetZeroQ4);
        q1Plus.onClick.RemoveAllListeners();
        q2Plus.onClick.RemoveAllListeners();
        q3Plus.onClick.RemoveAllListeners();
        q4Plus.onClick.RemoveAllListeners();
        q1Plus.onClick.AddListener(AddQ1);
        q2Plus.onClick.AddListener(AddQ2);
        q3Plus.onClick.AddListener(AddQ3);
        q4Plus.onClick.AddListener(AddQ4);
        q1Minus.onClick.RemoveAllListeners();
        q2Minus.onClick.RemoveAllListeners();
        q3Minus.onClick.RemoveAllListeners();
        q4Minus.onClick.RemoveAllListeners();
        q1Minus.onClick.AddListener(LessQ1);
        q2Minus.onClick.AddListener(LessQ2);
        q3Minus.onClick.AddListener(LessQ3);
```

```
        q4Minus.onClick.AddListener(LessQ4);
        q1.gameObject.SetActive(false);
        q2.gameObject.SetActive(false);
        q3.gameObject.SetActive(false);
        q4.gameObject.SetActive(false);
        SetAsJoinMode();
    }

    private void Update()
    {
        if (mostradorDePosicion != null)
        {
            var actualRobot = servoScara.GetResultingState();
            mostradorDePosicion.text =
            $"x:{actualRobot.x*1000:000.00} " +
            $"y:{actualRobot.y*1000:000.00} " +
            $"z:{actualRobot.z*1000:000.00} " +
            $"Rz:{actualRobot.w:000.00}";
        }
        if (mostradorDeArticulaciones != null)
        {
            mostradorDeArticulaciones.text =
                $"Q1:{servoScara.d1:000.00} \t" +
                $"Q2:{servoScara.d2:000.00} \t" +
                $"Q3:{servoScara.d3:000.00} \t" +
                $"Q4:{servoScara.d4:000.00}";
        }
    }
    private void AddQ1()
    {
        ModifyQ(0, true);
    }
    private void AddQ2()
    {
```

```
        ModifyQ(1, true);
    }
    private void AddQ3()
    {
        ModifyQ(2, true);
    }
    private void AddQ4()
    {
        ModifyQ(3, true);
    }
    private void LessQ1()
    {
        ModifyQ(0, false);
    }
    private void LessQ2()
    {
        ModifyQ(1, false);
    }
    private void LessQ3()
    {
        ModifyQ(2, false);
    }
    private void LessQ4()
    {
        ModifyQ(3, false);
    }
    private void ModifyQ(int q, bool positiveOrNegative)
    {
        var speed = velocidadMovimiento.value switch
        {
            0 => 100,
            1 => 10,
            2 => 1f,
            _ => 0.0f
```

```
};
if (!positiveOrNegative)
{
    speed *= (-1.0f);
}
Debug.Log($"M:{tipoDeMovimiento.value}," +
            $"V:{velocidadMovimiento.value}," +
            $"Q:{q}," +
            $"Dir{positiveOrNegative}");
switch (tipoDeMovimiento.value)
{
    case 0:
        switch (q)
        {
            case 0:
                servoScara.d1 += speed;
                break;
            case 1:
                servoScara.d2 += speed;
                break;
            case 2:
                servoScara.d3 += speed/100f;
                break;
            case 3:
                servoScara.d4 += speed;
                break;
        }
        break;
    case 1:
        var newPosition = servoScara.GetResultingState();
        Debug.Log($"xa:{newPosition.x:0.0000}," +
                    $"ya:{newPosition.y:0.0000}," +
                    $"za:{newPosition.z:0.0000}," +
                    $"ra:{newPosition.w:000.0}");
```

```
switch (q)
{
    case 0:
        newPosition.x += speed/1000;
        break;
    case 1:
        newPosition.y += speed/1000;
        break;
    case 2:
        newPosition.z += speed/1000;
        break;
    case 3:
        newPosition.w += speed;
        break;
}
Debug.Log($"xn:{newPosition.x:0.0000}," +
            $"yn:{newPosition.y:0.0000}," +
            $"zn:{newPosition.z:0.0000}," +
            $"rn:{newPosition.w:000.0}");
if (!IsValidNewXYZ(newPosition))
{
    Debug.Log("Is invalid the new position");
    return;
}
var solutions=IKScara.Solve(newPosition);
Debug.Log($"S1:{solutions.Item1}\tS2:{solutions.Item2}");
var evaluationSol = CheckSolutions(solutions);
if (!evaluationSol.Item1)
{
    Debug.Log("Non reachable the solutions");
    return;
}
Debug.Log($"Selected solution {evaluationSol.Item2}");
servoScara.d1 = evaluationSol.Item2.x;
```

```
                servoScara.d2 = evaluationSol.Item2.y;
                servoScara.d3 = evaluationSol.Item2.z;
                servoScara.d4 = evaluationSol.Item2.w;
                break;
        }
    }
    private void SetZeroQ1()
    {
        servoScara.d1 = 0;
    }
    private void SetZeroQ2()
    {
        servoScara.d2 = 0;
    }
    private void SetZeroQ3()
    {
        servoScara.d3 = 0;
    }
    private void SetZeroQ4()
    {
        servoScara.d4 = 0;
    }
    private void CambioDeTipoDeMovimiento(int arg0)
    {
        switch (arg0)
        {
            case 0:
                SetAsJoinMode();
                break;
            case 1:
                SetAsCartesianMode();
                break;
        }
    }
```

```
private bool IsValidNewXYZ(Vector4 newPosition)
{
    var x = newPosition.x;
    var y = newPosition.y;
    var z = newPosition.z;
    var rz = newPosition.w;
    var distance = Mathf.Sqrt(x * x + y * y);
    if (distance > IKScara.L2 * 2 / 3f && distance<(IKScara.L2+IKScara.L3))
    {
        if (FKScara.L1-0.2f<=z && z<=FKScara.L1)
        {
            if (-180f < rz || rz < 180f)
            {
                //Debug.Log("Acepted xyz theta new position");
                return true;
            }
            Debug.Log("Punto con una rotación inalcanzable");
            return false;
        }
        Debug.Log("Punto fuera de z");
        return false;
    }
    Debug.Log("Punto muy lejano");
    return false;
}
private Tuple<bool,Vector4> CheckSolutions(Tuple<Vector4,Vector4> Solutions)
{
    var actualStateOfRobot = new Vector4(servoScara.d1, servoScara.d2, servoSca
    var myReturn= new Tuple<bool, Vector4>(false,new Vector4(0,0,0,0));
    var solutionsAcepted = new List<Vector4>();
    if (IsOnWorkArea(Solutions.Item1))
    {
        solutionsAcepted.Add(Solutions.Item1);
    }
```

```
        if (IsOnWorkArea(Solutions.Item2))
        {
            solutionsAcepted.Add(Solutions.Item2);
        }

        if (solutionsAcepted.Count == 0)
        {
            return myReturn;
        }

        var bestSolution = new Vector4();
        var costSolution = float.MaxValue;
        foreach (var sol in solutionsAcepted)
        {
            var actualCost = (sol-actualStateOfRobot).sqrMagnitude;
            if (!(actualCost < costSolution)) continue;
            bestSolution = sol;
            costSolution = actualCost;
        }
        return new Tuple<bool, Vector4>(true, bestSolution);
    }
    private static bool IsOnWorkArea(Vector4 Solution)
    {
        if (Mathf.Abs(Solution.x) > 133.0f)
        {
            Debug.Log("Rechazada por la primera articulación");
            return false;
        }
        if (Mathf.Abs(Solution.y) > 155.0f)
        {
            Debug.Log("Rechazada por la segunda articulación");
            return false;
        }
        if (Solution.z is < 0 or > 0.2f)
```

```
        {
            Debug.Log("Rechazada por la tercera articulación");
            return false;
        }

        if (Solution.w is >= -360 and <= 360) return true;
        Debug.Log("Rechazada por la cuarta articulación");
        return false;
    }
}
```

I. ControlScaraDinamico.cs

```
public class ControlScaraDinamico : MonoBehaviour
{
    [Header("Articulaciones")]
    public ArticulationBody j1;
    public ArticulationBody j2;
    public ArticulationBody j3;
    public ArticulationBody j4;
    [Header("Controladores de posición")]
    [Range(-133,133)]
    public float s1;
    public float v1;
    public float a1;
    [Range(-153,153)]
    public float s2;
    public float v2;
    public float a2;
    [Range(-0.2f,0f)]
    public float s3;
    public float v3;
    public float a3;
    [Range(-153,153)]
    public float s4;
    public float v4;
    public float a4;
    [Header("PID adjusts")]
    public bool pidControl=false;
    // J1 Num=2.973;  Den=[1,2.874,1.476] Lambda=0.02s DMargin (+-0.94, 58 Deg) Cri
    // public Pid ControlJ1= new Pid(48.3f, 16.7f,24.8f,15f); //Improved using IMC
```

```
public Pid ControlJ1= new Pid(2.22f, 0f,5.19f,15f); //Improved using IMC contro
//  J2 Num=1.825;  Den=[1,0.695] Lambda=0.02s DMargin (+-0.94, 58 Deg) Critical
public Pid ControlJ2= new Pid(1.9f, 0f,2.74f,4f);           //Improved
// Num=2.259; Den=[1,0.8042]; Ts=0.02
public Pid ControlJ3= new Pid(1.78f, 0f,2.21f,1f);      //Pending to improve us
public Pid ControlJ4= new Pid(0.3f, 0.1f,0.3f,0.1f);    //Non usable for this v
public Vector3 ForceDirection = new Vector3(0, 1, 0);
public float Signal1, Signal2, Signal3, Signal4;
[Header("Controlled Torque")]
public Vector<float> mases = new DenseVector(new float[]{1,1,1,});
public Vector<float> link=new DenseVector(new float[]{0.139f,0.3f,0.3f});
public Vector<float> damp=new DenseVector(new float[]{0.1f,0.1f,0.1f});

[Header("Valores deseados")]
public Vector4 actualQ = new Vector4(0, 0, 0, 0);
public Vector4 wantedQ = new Vector4(0, 0, 0, 0);
public Vector4 esperado=new Vector4();
public Vector4 actual=new Vector4();

public CinematicaDirecta calculador;

[Header("Reporte para análisis de control")]
public string NameReport = "identJ3_2";
public bool isOverIdentification = false;
public Vector4 Amplitude = new Vector4(.10f, 0.1f, 0.1f, 0.1f);
public float wangular = 0.01f;

public float seconds2report = 5f;
public int _maxReportSize = 100;
private int _counter = 0;
private StreamWriter reporteW = null;

// Set the TcpListener on port 13000.
private EasyModbus.ModbusServer servidorDeDepuracion;
```

```
public bool magnet = false;
public bool isMovingAPiece = false;

[ReadOnly]
public float actualError = 0;
private void Start()
{
    _maxReportSize =  (int)(seconds2report/ 0.02f);
}
string vector2CSV(Vector4 vec)
{
    return vec[0] + "," + vec[1] + "," + vec[2] + "," + vec[3];
}
void FixedUpdate()
{
    var rotDirection = new Vector3(0, 1, 0);

    if (isOverIdentification)
    {
        if (_counter == 0)
        {
            reporteW = new StreamWriter(NameReport + DateTime.Now.Hour.ToString
            var sEncabezado = "Time,J1,J2,J3,J4,T1,T2,F3,T4";
            reporteW.WriteLine(sEncabezado);
            const string fl = "0,0,0,0,0,0,0,0,0";
            reporteW.WriteLine(fl);
        }
        else
        {
            if (_counter < _maxReportSize)
            {
                var verIn = new Vector4(Signal1, Signal2, Signal3, Signal4);
                StringBuilder stringBuilder = new StringBuilder();
```

```
            stringBuilder.Append((_counter*Time.fixedDeltaTime)+","+vector2
            reporteW.WriteLine(stringBuilder.ToString());
        }
        else if (_counter == _maxReportSize)
        {
            reporteW.Flush();
            reporteW.Close();
            Debug.Log("Se ha generado el reporte para la identificación");
        }
    }
    if (j4 != null)
    {

        j4.AddTorque(rotDirection*Signal4);
        actualQ[3] = j4.jointPosition[0];
        wantedQ[3] = -s4 * Mathf.Deg2Rad;
    }
    else
    {
        Signal4 = 0;
        actualQ[3] = 0;
        wantedQ[3] = 0;
    }
    j1.AddTorque(rotDirection*Signal1);
    j2.AddTorque(rotDirection*Signal2);
    j3.AddForce(ForceDirection*Signal3);

    actualQ[0] = j1.jointPosition[0];
    actualQ[1] = j2.jointPosition[0];
    actualQ[2] = j3.jointPosition[0];

    wantedQ[0] = -s1 * Mathf.Deg2Rad;
    wantedQ[1] = -s2 * Mathf.Deg2Rad;
    wantedQ[2] = s3;
```

```
        _counter++;
        return;
    }
    //Control de articulación I

    if (pidControl)
    {
        Signal1 = ControlJ1.Step(-s1 * Mathf.Deg2Rad, j1.jointPosition[0], Time
        j1.AddTorque(rotDirection * Signal1);
        Signal2 = ControlJ2.Step(-s2 * Mathf.Deg2Rad, j2.jointPosition[0], Time
        j2.AddTorque(rotDirection * Signal2);
        //var qMass = J3.mass;
        Signal3 = ControlJ3.Step(s3, j3.jointPosition[0], Time.fixedDeltaTime);
        j3.AddForce(ForceDirection * Signal3);
        if (j4 != null)
        {
            Signal4 = (ControlJ4.Step(-s4 * Mathf.Deg2Rad, j4.jointPosition[0],
            j4.AddTorque(Signal4 * rotDirection);
            actualQ[3] = j4.jointPosition[0];
            wantedQ[3] = -s4 * Mathf.Deg2Rad;

        }
        else
        {
            s4 = 0;
            actualQ[3] = 0;
            wantedQ[3] = 0;
        }
    }
    else
    {
        j1.AddTorque(rotDirection * -s1);
        j2.AddTorque(rotDirection * -s2);
        j3.AddForce(ForceDirection * s3);
```

```
        }
        actualQ[0] = j1.jointPosition[0];
        actualQ[1] = j2.jointPosition[0];
        actualQ[2] = j3.jointPosition[0];
        wantedQ[0] = -s1 * Mathf.Deg2Rad;
        wantedQ[1] = -s2 * Mathf.Deg2Rad;
        wantedQ[2] = s3;
        actual = calculador.CalculateFk(new Vector4(-j1.jointPosition[0], -j2.joint
            actualQ[3]));
        esperado = calculador.CalculateFk(new Vector4(s1, s2, s3, s4));
    }

    public float ActualError()
    {
        wantedQ[0] = -s1 * Mathf.Deg2Rad;
        wantedQ[1] = -s2 * Mathf.Deg2Rad;
        wantedQ[2] = s3;
        wantedQ[3] = -s4 * Mathf.Deg2Rad;

        var vectorError = (wantedQ - actualQ);
            vectorError[2] *= 10;
            actualError=vectorError.sqrMagnitude;
            return actualError;
    }
}
```

J. TransformadaDeEspacio.m

```
% Robot
P1=[1,1,1]
R1=rotz(15, 'deg')
L1=rt2tr(R1, P1')
% Tarjet
P2=[3,2,6]
R2=roty(0, 'deg')
L2=rt2tr(R2, P2')
% Ubicación 1
trplot(L1)
trplot(L2)
%Diferencia relativa
dife=P2'-P1'
%Transformación de espació, cambio de base
rot=(R1'*dife)
L3=rt2tr(eye(3),rot)
newP=L1*L3
L2(1:3,4)-newP(1:3,4)
% Salida de matlab [0;0;0]
```

Bibliografía

[1] CONACYT, “Red de laboratorios virtuales,” 4 2021. [Online]. Available: https://laboratoriosvirtuales.mx/

[2] ASEE, *COVID-19 AND ENGINEERING EDUCATION*, 2020. [Online]. Available: www.asee.org.

[3] E. Yamamura and Y. Tsustsui, “The impact of closing schools on working from home during the covid-19 pandemic: evidence using panel data from japan,” *Review of Economics of the Household*, vol. 19, pp. 41–60, 3 2021.

[4] P. Matzinger and J. Skinner, “Strong impact of closing schools, closing bars and wearing masks during the covid-19 pandemic: results from a simple and revealing analysis.” *medRxiv : the preprint server for health sciences*, 9 2020. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/33024976http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC7536875

[5] V. Ramos, J. César, L. Claudia, G. Saldaña, R. M. Elena, L. Abel, B. Brugada, L. Elizabeth, G. Fragoso, and C. Margarita, “Academic performance during the covid-19 pandemic and its relationship with demographic factors and alcohol consumption in college students,” *International Journal of Environmental Research and Public Health*, vol. 19, 1 2022.

[6] UNESCO, UNICEF, and W. Bank, *The State of the Global Education Crisis A Path to Recovery.*

[7] O. Broadbent, T. Up, and E. Mccann, *Experience-led learning for engineers-A good practice guide*, 2016.

[8] S. Tselegkaridis and T. Sapounidis, “Simulators in educational robotics : A review,” 2021.

[9] J. Grodotzki, T. R. Ortelt, A. E. Tekkaya, J. Grodotzki, T. R. Ortelt, and A. E. Tekkaya, "Remote and virtual labs for engineering education 4.0," *Procedia Manufacturing*, vol. 26, pp. 1349–1360, 2018. [Online]. Available: https://doi.org/10.1016/j.promfg.2018.07.126

[10] V. B. Gómez-pablos and A. G. valcárcel Muñoz-repiso, "Project-based learning in the classroom : a case study at high school level," 2020.

[11] J. Broisin, R. Venant, and P. Vidal, "Lab4ce : a remote laboratory for computer education," 2015.

[12] W. H. B, C. Salzmann, and H. Jamkojian, "Enabling the automatic generation of user interfaces for remote laboratories," vol. 1, 2018.

[13] T. Bower, "Teaching introductory robotics programming: Learning to program with national instruments' labview," *IEEE Robotics and Automation Magazine*, vol. 23, pp. 67–73, 6 2016.

[14] C. M. Ionescu, E. Fabregas, S. M. Cristescu, S. Dormido, and R. D. Keyser, "A remote laboratory as an innovative educational tool for practicing control engineering concepts," vol. 56, pp. 436–442, 2013.

[15] J. J. Craig, *Introduction to robotics : mechanics and control*. Addison-Wesley, 1989.

[16] M. T. Das and L. C. Du, "Mathematical modelling , simulation and experimental verification of a scara robot mathematical modelling , simulation and experimental verification of a scara robot," 2005.

[17] M. Selvakumar, "Kinematic modeling and simulation of a scara robot by using solid dynamics and verification by matlab / simulink robot by using solid dynamics and verification by," 1999.

[18] B. Ibrahim and A. M. A. Zargoun., "Modelling and control of scara manipulator," *ScienceDirect*, vol. 42, pp. 106–113, 2014. [Online]. Available: file:///C:/Users/alanf/AppData/Local/Temp/modelling-and-control-of-scara-manipulator.pdf

[19] A. Ali, A. Moghadam, M. R. Gharib, M. Moavenian, and K. Torabi, "Modelling and control of a scara robot using quantitative feedback theory," 2009.

[20] B. Ibrahuim and A. M. Z. And, "Modelling and control of scara manipulator," *International COnference of Robot PRIDE 2013-2014*, vol. 42, pp. 106–113, 2014.

[21] A. Burghardt, D. Szybicki, P. Gierlak, K. Kurc, P. Pietruś, and R. Cygan, "Programming of industrial robots using virtual reality and digital twins," *Applied Sciences (Switzerland)*, vol. 10, 1 2020.

[22] V. S. Pantelidis, "Reasons to use virtual reality in education and training courses and a model to determine when to use virtual reality," *Themes in Science and Technology Education*, vol. 2, pp. 59–70, 2010.

[23] D. Vergara, M. P. Rubio, and M. Lorenzo, "On the design of virtual reality learning environments in engineering," 2017.

[24] J. Alvarez and M. Macías, "Programming logical controllers using remote labs and virtual reality."

[25] G. Garg, V. Kuts, and G. Anbarjafari, "Digital twin for fanuc robots: Industrial robot programming and simulation using virtual reality," *Sustainability (Switzerland)*, vol. 13, 9 2021.

[26] K. Herbus and P. Ociepka, "Conception of integration of the virtual robot model with the control system."

[27] Y. Ismail, A. Mashhadany, and Y. A. Mashhadany, "Design, analysis, simulation, and virtual-reality-verified intelligent controller for industrial-application scara robot." [Online]. Available: https://www.researchgate.net/publication/250928609

[28] W. Grega, "Hardware-in-the-loop simulation and its application in control education," *Proceedings - Frontiers in Education Conference*, vol. 2, 1999.

[29] ISO, "Industrial robots - definition and classification industrial robot as defined by iso 8373 : 2012 : An automatically controlled , reprogrammable , multipurpose," pp. 29–42, 2012. [Online]. Available: https://ifr.org/img/office/Industrial_Robots_2016_Chapter_1_2.pdf

[30] P. Corke, *Robotics, Vision and Control*. Springer International Publishing, 2017, vol. 118. [Online]. Available: http://link.springer.com/10.1007/978-3-319-54413-7

[31] E. y. Z. Daniel Pérez, Marco Antonio; Cuevas, *Fundamentos de robótica y mecatrónica con MATLAB(c) y Simulink(c)*. AlfaOmega, 2015.

[32] F. M. Sánchez-Martín, P. J. Schlegl, F. M. Rodríguez, J. S. Bayarri, V. M. Font, J. P. Redorta, and H. V. Mavrich, "Historia de la robótica: De arquitas de tarento al robot da vinci. (parte ii)," *Actas Urologicas Espanolas*, vol. 31, pp. 185–196, 2007.

[33] O. L. Dictionaries, "Robot definition," 2021. [Online]. Available: https://www.oxfordlearnersdictionaries.com/definition/english/robot?q=robot

[34] M. Ben-Ari and F. Mondada, *Elements of Robotics NN ML in robotics. Fundamentals*, 2017.

[35] J. P. Lewis, M. J. Keeling, P. Rohani, K. Features, F. Lecture, H. S. Paul, R. Lainerie, G. J. Ritz, J. George, D. From, D. From, D. From, G. J. Ritz, J. George, E. Elbeltagi, D. From, D. From, D. From, O. J. Klakegg, and G. Oberlender, *ww w . E asy En gi nee rin g ww w syE ngi rin g*, 2007, vol. 31. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/01446193.2013.867519%0Ahttp://www.sciencedirect.com/science/article/B98GG-4T7XCGF-9D/2/95c40489e0e5ee3c0670ee1cf3f0e65f%0Ahttp://books.google.com/books?id=zvswkI3hkvoC&pgis=1

[36] A.-N. Sharkawy, "Chapter 5 minimum jerk trajectory generation for straight and curved movements: Mathematical analysis."

[37] P. Sanz, *Robotics: Modeling, Planning, and Control*, 2009, vol. 16.

[38] S. Echeandia and P. M. Wensing, "Numerical methods to compute the coriolis matrix and christoffel symbols for rigid-body systems," *Journal of Computational and Nonlinear Dynamics*, vol. 16, 9 2021.

[39] A. J. Brizard, "An introduction to lagrangian mechanics," 2007.

[40] M. M. Miloššžefran and F. Bullo, "Lagrangian dynamics," 2004. [Online]. Available: http://motion.me.ucsb.edu/pdf/2004a-zb.pdf

[41] R. Featherstone, *ROBOT DYNAMICS ALGORITHMS*. Kluwer Academic Publishers, 1985.

[42] G. Arora, V. Joshi, and I. S. Garki, "Developments in runge–kutta method to solve ordinary differential equations," pp. 193–202, 3 2020.

[43] İsmail KOYUNCU and H. İbrahim ŞEKER, "Implementation of dormand-prince based chaotic oscillator designs in different iq-math number standards on fpga," *Sakarya University Journal of Science*, pp. 859–868, 10 2019.

[44] T. Kimura, "On dormand-prince method," 2009. [Online]. Available: http://documents.wolfram.com/v5/Built-inFunctions/AdvancedDocumentation/

[45] K. Ogata, *Ingeniería de control Moderno*, 2010. [Online]. Available: www.pearsoneducacion.com

[46] ——, "Sistemas de control tiempo discreto 2ed," 1996. [Online]. Available: http://books.google.com/books?hl=en&lr=&id=aYFUs17m0YQC&oi=fnd&dq=Sistemas+de+Control+en+tiempo+Discreto&ots=5HaM3ANz2B&sig=PRRSAw4Dx3sXmzGHf-ITHfVul_4

[47] V. V. Patel, "Ziegler-nichols tuning method: Understanding the pid controller," *Resonance*, vol. 25, pp. 1385–1397, 10 2020.

[48] R. P. Borase, D. K. Maghade, S. Y. Sondkar, and S. N. Pawar, "A review of pid control, tuning methods and applications," 2020.

[49] M. A. Ibrahim, A. K. Mahmood, and N. S. Sultan, "Optimal pid controller of a brushless dc motor using genetic algorithm," *International Journal of Power Electronics and Drive Systems*, vol. 10, pp. 822–830, 6 2019.

[50] R. Kelly, *Control of robot manipulators in joint space*. Springer, 2005.

[51] "The imc-based pid procedure," 1999.

[52] A. D. Luca, "Industrial robotics what is a robot ?" pp. 1–68, 2016.

[53] L. M. Fernández, "Analysis of industrial robots. study of a scara," 2016.

[54] M. Grieves and J. Vickers, "Digital twin : Mitigating unpredictable , undesirable emergent behavior in complex systems," 2017.

[55] D. Twins, "applied sciences programming of industrial robots using virtual reality and digital twins," 2020.

[56] N. Kousi, C. Gkournelos, C. Giannoulis, A. Santana, P. Afonso, A. Zanin, and R. Wernke, "Digital twin for adaptation of robots' behavior in flexible assembly lines," *Procedia Manufacturing*, vol. 28, pp. 121–126, 2019. [Online]. Available: https://doi.org/10.1016/j.promfg.2018.12.020

[57] A. B. Sevilla, "Realidad aumentada en educación." [Online]. Available: http://oa.upm.es/45985/1/Realidad_Aumentada__Educacion.pdf

[58] "El modelo osi."

[59] MODBUS, "Modbus application protocol spefication," pp. 1–50, 2012. [Online]. Available: https://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[60] C. World and C. World, "Automation for a changing world smart robot for upgraded industries."

[61] P. Corke, "Inverse kinematics for a 2-joint robot arm using geometry." [Online]. Available: https://robotacademy.net.au/lesson/inverse-kinematics-for-a-2-joint-robot-arm-using-geometry/

[62] P. A. G. T. Ag, "The moore-penrose pseudoinverse (math 33a: Laub)," vol. 1, pp. 1–5, 1972. [Online]. Available: http://www.math.ucla.edu/~laub/33a.2.12s/mppseudoinverse.pdf

[63] O. Paulsson, "The rayleigh dissipation function," pp. 1–17, 2014.

[64] E. Minguzzi, "Rayleigh ' s dissipation function at work," pp. 1–11.

[65] R. Fearestone, "Inverse dynamics," pp. 89–100, 2008.

[66] U. Technologies, "Order of execution for event functions," 7 2022. [Online]. Available: https://docs.unity3d.com/Manual/ExecutionOrder.html