



UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

FACULTAD DE INGENIERÍA

Desarrollo y análisis de robot serial plano R||R.

**TESIS COLECTIVA**

Que como parte de los requisitos para obtener  
el Grado de Ingeniero en Automatización

**Presentan:**

Oswaldo Lara Hurtado  
Diego Andrés Ramírez Malagón

**Dirigida por:**

Dr. Gerardo Israel Pérez Soto

**Codirigida por:**

Dra. Karla Anhel Camarillo Gómez

Dr. Gerardo Israel Pérez Soto

Presidente

Dr. Miguel Ángel Martínez Prado


Secretario


Dra. Karla Anhel Camarillo Gómez


Vocal


Dr. J. Jesús de Santiago Pérez

Súplente

  
\_\_\_\_\_  
Firma

  
\_\_\_\_\_  
Firma

  
\_\_\_\_\_  
Firma

  
\_\_\_\_\_  
Firma

Centro Universitario, Querétaro, Qro. Noviembre 2020

*A mis padres y hermanos, porque sin ustedes yo no sería nada.*  
*Oswaldo Lara*

*Para Ximena, Leonardo, Emilia y Luciana.*  
*Diego Ramírez*

# Agradecimientos

Este trabajo jamás hubiera sido posible sin la contribución de muchas personas, compañeros y profesores. Quiero agradecer particularmente a Ana Monroy, Ruth Suárez, Héctor Rodríguez, Josué Martínez, Oscar Huelga y Rodrigo González, quienes nos apoyaron con amabilidad cuando acudimos a ellos, infinitas gracias.

Quiero agradecer en especial a nuestro asesor, el Dr. Gerardo Israel Pérez Soto, quien estuvo siempre en la mejor disposición de apoyarnos, brindándonos su respaldo y conocimiento, de manera que pudieran cumplirse adecuadamente los objetivos planteados para el desarrollo de este trabajo.

También me gustaría agradecer al M.C. José Luis Avendaño Juárez, quien fue mi profesor por varios años y quien me brindó su amistad y apoyo siempre que lo necesité, gracias.

Agradezco a la Universidad Autónoma de Querétaro, la cual fue mi hogar durante 5 años, me llenó de conocimientos y donde encontré amigos para toda la vida, ayudándome así con mi desarrollo personal y profesional, muchas gracias.

Finalmente, quiero agradecer a mi familia, a mi madre Gabriela, a mi padre Lazaro, a mi hermana Fernanda y a mis hermanos Jorge Luis, Omar y Leonardo, quienes siempre están conmigo y me apoyan sin condiciones, jamás terminaría de agradecerles, los amo.

*Oswaldo Lara Hurtado*

Agradezco a nuestro asesor de tesis, Dr. Gerardo Israel Pérez Soto, quien con su liderazgo y paciencia fue clave para ayudarnos a tener un objetivo claro y a unificar nuestras ideas.

A nuestros compañeros Ana Monroy, Héctor Rodríguez, Josué Martínez, Rodrigo González, Ruth Suárez y Oscar Huelga, quienes aportaron la base de este proyecto y nos brindaron soporte cada que lo necesitamos.

También agradecer a la Universidad Autónoma de Querétaro por otorgarme las herramientas necesarias para superar cualquier obstáculo que se me presente en mi vida profesional y personal. Me siento orgulloso de pertenecer a esta institución y siempre estaré agradecido con mis profesores por impulsarnos a dar nuestro mejor esfuerzo.

A mi familia por su apoyo y amor incondicional.

*Diego Andrés Ramírez Malagón*

# Resumen

En el presente trabajo, se muestra el desarrollo y análisis cinemático de un robot serial plano de dos grados de libertad, el cual es de un tamaño accesible, permitiendo su fácil transporte y resguardo. Se realiza el diseño mecánico del robot, para su posterior construcción e integración con los motores y la electrónica requerida para la transmisión de datos con la tarjeta de control; en este caso, una Raspberry Pi 3 Model B+<sup>®</sup>. En esta misma tarjeta, se implementa un controlador cinemático a partir de la solución cerrada del análisis cinemático inverso de posición, además, se desarrolla una interfaz gráfica para poder controlar el movimiento del robot, de manera que sea intuitiva para el usuario. El resultado final es un robot completamente funcional, capaz de realizar control de posición, programación punto a punto y planificación de trayectorias a través de la interfaz de usuario.

Palabras clave: *Robot serial plano, Análisis cinemático, Control de posición, Planificación de trayectorias.*

# Abstract

In the present work, the development and kinematic analysis of a two degrees-of-freedom serial plane robot is shown, which is sized for easy transportation and storage. The mechanical design of the robot is done, to continue with its construction and integration with the motors and the electronic circuit required to the data transmission with the control system; in this case, a Raspberry Pi 3 Model B+<sup>®</sup>. A kinematic controller is implemented in this system from the closed solution of the position inverse kinematics analysis, besides, a graphic interface is developed to control the robot movement, so that it is intuitive for the user. The final result is a completely functional robot, which is able to perform position control, point-to-point programming and planning path through the user interface.

Keywords: *Serial plane robot, Kinematic analysis, Position control, Planning path.*

# Índice general

|  |          |
|--|----------|
| Agradecimientos  | II       |
| Resumen  | III      |
| Abstract   | IV       |
| Índice general   | V        |
| Índice de figuras  | VII      |
| Índice de tablas   | IX       |
| <b>1. Introducción</b>   | <b>1</b> |
| 1.1. Antecedentes  | 1        |
| 1.1.1. Manipuladores seriales  | 2        |
| 1.2. Justificación   | 2        |
| 1.3. Descripción del problema  | 2        |
| 1.4. Fundamentos teóricos  | 3        |
| 1.4.1. Parámetros Denavit-Hartenberg                                   | 3        |
| 1.4.2. Matrices de transformación homogénea                            | 4        |
| 1.4.3. Análisis cinemático de posición                                 | 4        |
| 1.4.4. Matriz Jacobiana  | 6        |
| 1.4.5. Selección de tolerancias de maquinado                           | 7        |
| 1.5. Objetivos e hipótesis   | 7        |
| 1.5.1. Objetivo general  | 7        |
| 1.5.2. Objetivos particulares  | 7        |
| 1.5.3. Hipótesis   | 8        |
| 1.6. Equipo, materiales y software                                     | 8        |
| <b>2. Desarrollo del robot serial plano R  R</b>                       | <b>9</b> |
| 2.1. Descripción de hardware del robot serial plano R  R               | 9        |
| 2.1.1. Características servomotor Dynamixel MX-28                      | 9        |
| 2.1.2. Raspberry Pi 3 Model B+®  | 10       |
| 2.2. Análisis estático para la determinación de parámetros geométricos | 10       |
| 2.3. Diseño de los elementos estructurales del robot serial plano R  R | 13       |
| 2.4. Manufactura de los elementos estructurales                        | 14       |
| 2.5. Diseño electrónico del robot serial plano R  R                    | 16       |

|  |           |
|--|-----------|
| <b>3. Análisis cinemático del robot serial plano R  R</b>  | <b>18</b> |
| 3.1. Análisis cinemático de posición del robot serial plano R  R . . . . .   | 18        |
| 3.1.1. Análisis cinemático de posición directo del robot serial plano R  R . . . . .   | 18        |
| 3.1.2. Análisis cinemático de posición inverso del robot serial plano R  R . . . . .   | 19        |
| 3.2. Análisis cinemático de velocidad del robot serial plano R  R . . . . .  | 22        |
| 3.2.1. Análisis cinemático de velocidad directo del robot serial plano R  R . . . . .  | 22        |
| 3.2.2. Análisis cinemático de velocidad inverso del robot serial plano R  R . . . . .  | 23        |
| <b>4. Implementación del modelo cinemático y desarrollo de interfaz de usuario en el sistema embebido Raspberry Pi 3 Model B+(R)</b> | <b>24</b> |
| 4.1. Descripción de software para implementación del modelo cinemático . . . . .   | 24        |
| 4.2. Descripción de software para desarrollo de la interfaz de usuario . . . . .   | 30        |
| <b>5. Descripción de la interfaz de usuario</b>  | <b>35</b> |
| 5.1. Arranque de la aplicación y descripción de la ventana principal . . . . .   | 35        |
| 5.2. Descripción de la pestaña “Inversa” . . . . .   | 39        |
| 5.2.1. Descripción de la ventana “Trayectorias” . . . . .  | 41        |
| 5.3. Descripción de la pestaña “Directa” . . . . .   | 48        |
| 5.4. Descripción de la pestaña “Manual” . . . . .  | 49        |
| 5.5. Descripción de la pestaña “Grabar” . . . . .  | 50        |
| <b>6. Conclusiones</b>   | <b>52</b> |
| <b>A. Anexos</b>   | <b>53</b> |
| A.1. Planos del diseño final . . . . .   | 53        |
| A.2. Códigos generados en el desarrollo del robot serial plano R  R . . . . .  | 54        |
| A.3. Solución cerrada de velocidad usando Matlab . . . . .   | 55        |
| <b>Bibliografía</b>  | <b>57</b> |

# Índice de figuras

|  |    |
|--|----|
| 1.1. Ejemplo de un manipulador serial: Robot Pelicano, imagen obtenida de [6]. . . . . | 2  |
| 1.2. Parámetros Denavit-Hartenberg, imagen obtenida de [3]. . . . .                    | 3  |
| 1.3. Relación entre cinemática directa e inversa, imagen obtenida de [3]. . . . .      | 4  |
| 2.1. Motor Dynamixel MX-28. . . . .  | 9  |
| 2.2. Raspberry Pi 3 Model B+ <sup>®</sup> . . . . .                                    | 10 |
| 2.3. Subensamble de eslabones y servomotores. . . . .                                  | 11 |
| 2.4. Modelo estático del robot serial plano R  R. . . . .                              | 11 |
| 2.5. Prototipos preliminares. . . . .  | 13 |
| 2.6. Prototipo final. . . . .  | 14 |
| 2.7. Barrenado de la base del perfil Bosch. . . . .                                    | 15 |
| 2.8. Acabado final de la base principal. . . . .                                       | 15 |
| 2.9. Escuadrado de la base del perfil Bosch. . . . .                                   | 15 |
| 2.10. Barrenado del soporte del motor 1. . . . .                                       | 15 |
| 2.11. Evidencia de maquinado. . . . .  | 16 |
| 2.12. Evidencia de maquinado. . . . .  | 16 |
| 2.13. Diagrama eléctrico. . . . .  | 17 |
| 2.14. Placa de circuito impreso. . . . .   | 17 |
| 3.1. Configuración de referencia. . . . .  | 18 |
| 4.1. Interfaz software QT Creator. . . . .   | 31 |
| 4.2. Funciones generales de la interfaz. . . . .                                       | 31 |
| 4.3. Función “ <i>onExeInvClicked</i> ”. . . . .                                       | 32 |
| 4.4. Función “ <i>onMakeRecord2Clicked</i> ”. . . . .                                  | 33 |
| 4.5. Función “ <i>onExeDirClicked</i> ”. . . . .                                       | 33 |
| 4.6. Función “ <i>onMakeCircleClicked</i> ”. . . . .                                   | 34 |
| 4.7. Función “ <i>onPlotCClicked</i> ”. . . . .  | 34 |
| 5.1. Escritorio de la Raspberry Pi. . . . .  | 35 |
| 5.2. Ventana de ejecución de la interfaz de usuario. . . . .                           | 36 |
| 5.3. Ventana principal. . . . .  | 36 |
| 5.4. Ventana principal. . . . .  | 36 |
| 5.5. Posición inicial de robot. . . . .  | 37 |
| 5.6. Posición de apagado de robot. . . . .   | 37 |
| 5.7. Ventana principal. . . . .  | 38 |
| 5.8. Comparación orientación de codo en coordenadas [10,-15]. . . . .                  | 38 |
| 5.9. Comparación orientación de codo en coordenadas [10,-15]. . . . .                  | 39 |



|   |    |
|---|----|
| 5.10. Captura de coordenadas. . . . .                           | 39 |
| 5.11. Ejecución del análisis. . . . .                           | 39 |
| 5.12. Valores resultantes de las variables articulares. . . . . | 40 |
| 5.13. Opción “Trayectorias” del método inverso. . . . .         | 40 |
| 5.14. Ejemplo correcto. . . . .                                 | 41 |
| 5.15. Ejemplo incorrecto. . . . .                               | 41 |
| 5.16. Ventana “Trayectorias”. . . . .                           | 42 |
| 5.17. Captura de datos. . . . .                                 | 42 |
| 5.18. Ejecución de la trayectoria. . . . .                      | 42 |
| 5.19. Ejemplo correcto. . . . .                                 | 42 |
| 5.20. Botón “Graficar”. . . . .                                 | 43 |
| 5.21. Gráfica de la trayectoria. . . . .                        | 43 |
| 5.22. Pestaña “Elipse”. . . . .                                 | 43 |
| 5.23. Captura de datos. . . . .                                 | 43 |
| 5.24. Ejecución de la trayectoria. . . . .                      | 44 |
| 5.25. Ejemplo correcto. . . . .                                 | 44 |
| 5.26. Botón “Graficar”. . . . .                                 | 45 |
| 5.27. Gráfica de la trayectoria. . . . .                        | 45 |
| 5.28. Pestaña “Otras”. . . . .                                  | 46 |
| 5.29. Captura de datos. . . . .                                 | 46 |
| 5.30. Ejecución de la trayectoria. . . . .                      | 46 |
| 5.31. Ejemplo correcto. . . . .                                 | 46 |
| 5.32. Botón “Graficar”. . . . .                                 | 47 |
| 5.33. Gráfica de la trayectoria. . . . .                        | 47 |
| 5.34. Trayectoria robot para $k=3$ . . . . .                    | 47 |
| 5.35. Gráfica de la trayectoria con valor $k = 4$ . . . . .     | 48 |
| 5.36. Gráfica de la trayectoria con valor $k = 5$ . . . . .     | 48 |
| 5.37. Gráfica de la trayectoria con valor $k = 6$ . . . . .     | 48 |
| 5.38. Captura de datos. . . . .                                 | 49 |
| 5.39. Ejecución del análisis. . . . .                           | 49 |
| 5.40. Respuesta del robot. . . . .                              | 49 |
| 5.41. Ventana principal. . . . .                                | 50 |
| 5.42. Captura de datos. . . . .                                 | 50 |
| 5.43. Ejecución del análisis. . . . .                           | 50 |
| 5.44. Captura de datos. . . . .                                 | 51 |
| 5.45. Ejecución del análisis. . . . .                           | 51 |
| A.1. Código QR planos robot plano R  R. . . . .                 | 53 |
| A.2. Código QR códigos robot plano R  R. . . . .                | 54 |

# Índice de tablas

|  |    |
|--|----|
| 2.1. Especificaciones del servomotor Dynamixel MX-28. . . . .                            | 10 |
| 2.2. Parámetros geométricos y físicos de los eslabones generados en SolidWorks®. . . . . | 12 |
| 3.1. Parámetros Denavit-Hartenberg del robot de dos grados de libertad. . . . .          | 19 |

Dirección General de Bibliotecas UAQ

# Capítulo 1

## Introducción

### 1.1. Antecedentes

La robótica es una rama de la tecnología que se ocupa del estudio, el diseño y la construcción de los robots, al igual que la interacción con el medio que lo rodea, resulta de la combinación de ciencias orientadas a satisfacer un objetivo común, el cual es diseñar y construir máquinas con la capacidad de desempeñar tareas realizadas por el hombre; entre las ciencias que se ocupan para la robótica se encuentra a la física, la mecánica, las matemáticas, la informática, la electrónica, la computación, entre otras [1].

El término “robot” proviene de la palabra checa *robot* y significa *trabajo*; fue introducido en nuestro vocabulario por el dramaturgo Karel Čapek en 1921 en su novela satírica *Rossum's Universal Robots*, donde describe al robot como una máquina que sustituye a los seres humanos para ejecutar tareas sin descanso [1].

El robot se define, de manera formal en la Organización Internacional para la Estandarización (ISO), como un manipulador multifuncional reprogramable, capaz de mover materiales, piezas, herramientas o dispositivos especiales, a través de movimientos variables programados, para el desempeño de tareas diversas [2].

Mecánicamente, un robot está formado por una serie de elementos o eslabones unidos mediante articulaciones que permiten un movimiento relativo entre cada dos eslabones consecutivos. El movimiento de cada articulación puede ser de desplazamiento, de giro, o de una combinación de ambos. Cada uno de los movimientos independientes que puede realizar cada articulación con respecto a la anterior, se denomina grado de libertad (GDL). El número de grados de libertad del robot viene dado por la suma de los grados de libertad de las articulaciones que lo componen [3].

Una definición de GDL presentada por Kutzbach-Grübler, es el número de parámetros independientes que se requieren para definir de forma única la posición del mecanismo en el espacio en cualquier momento. Existen diferentes criterios para calcularlos, el más usado es el Criterio de Kutzbach-Grübler [4]:

$$M = 3(L - 1) - 2J_1 - J_2 \quad (1.1)$$

donde  $M$  es el número de grados de libertad,  $L$  es el número de eslabones,  $J_1$  es el número de uniones de 1 GDL y  $J_2$  es el número de uniones de 2 GDL.

Los robots manipuladores se pueden dividir en categorías, siendo dos las más comunes los robots seriales y los robots paralelos, de los cuales sólo nos enfocaremos en los primeros, debido a que es el tipo de robot sobre el que se hablará en este trabajo.

### 1.1.1. Manipuladores seriales

Un robot serial (manipulador de cadena cinemática abierta) está formado por eslabones rígidos conectados en serie, básicamente por uniones prismáticas (P) o de revolución (R), un extremo se encuentra anclado a tierra y el otro se mueve libremente. Entre las diferentes clasificaciones de la literatura, se encuentran la clasificación acorde a su estructura cinemática, la clasificación por grados de libertad y por el área de trabajo [5]. La Figura 1.1, muestra un ejemplo de robot plano R||R, el cual se desarrolló en CICESE [6].



Figura 1.1: Ejemplo de un manipulador serial: Robot Pelicano, imagen obtenida de [6].

## 1.2. Justificación

En los últimos años, la mayoría de los estudios de control de robot seriales se desarrollaron considerando motores de corriente directa, CD; sin embargo, actualmente, se emplean servomotores para el desarrollo de proyectos. En particular, el equipo de competencia de robótica humanoide de la FI-UAQ, emplea servomotores Dynamixel como los utilizados en el presente proyecto. En este trabajo de tesis, se desarrolla un prototipo didáctico, con el cual los alumnos puedan implementar parte de los conocimientos desarrollados en la materia de Robótica. Además, la plataforma obtenida servirá como trabajo futuro en el planteamiento de leyes de control para este tipo de actuadores.

## 1.3. Descripción del problema

En la carrera de ingeniería en automatización, específicamente en la línea terminal de mecatrónica, se imparte la clase de Robótica, enfocada al cálculo cinemático de robots y en el control dinámico de los mismos. La profundidad con la que se tratan los temas de clase no permite la comprensión de los mismos, ya que no existen herramientas prácticas que permitan aplicar los cálculos que se estiman en el aula. Por lo que, contar con un robot serial básico para aplicar la teoría vista en clase tanto de

análisis cinemático directo e inverso de posición, como de teoría de control de robots manipuladores al replicar las leyes de control en un controlador afín a otras materias vistas durante la carrera.

## 1.4. Fundamentos teóricos

### 1.4.1. Parámetros Denavit-Hartenberg

Un robot manipulador puede describirse en forma cinemática proporcionando los valores de cuatro cantidades para cada articulación. Dos describen la articulación en sí misma, y los otros dos describen la conexión de la articulación con una articulación adyacente. En el caso de una articulación angular, la variable de articulación será  $\theta_i$  y las otras tres cantidades son parámetros de articulación fijos. Para las articulaciones prismáticas,  $d_i$  será la variable de articulación y las otras tres cantidades son parámetros de articulación fijos. La definición del robot por medio de estas cantidades es una convención que generalmente se le conoce como notación Denavit-Hartenberg.

A las cuatro cantidades antes mencionadas se les conoce como los parámetros Denavit-Hartenberg y existen distintas convenciones para la generación de estos, para este trabajo de tesis se utilizará la convención distal a dextrógiro.

Se colocan  $n + 1$  ejes coordenados en el robot, el eje coordenado  $(X_0, Y_0, Z_0)$  no se moverá y estará localizado en la base del robot; el resto de los ejes serán colocados en las articulaciones hasta llegar al actuador final.

Se generarán los cuatro parámetros Denavit-Hartenberg para cada eje coordenado:

- $\theta_i$  es el ángulo entre los ejes  $X_{i-1}$  y  $X_i$ , medido en un plano perpendicular a  $Z_{i-1}$ .
- $d_i$  es la distancia a lo largo del eje  $Z_{i-1}$  hasta la intersección de  $Z_{i-1}$  con  $X_i$ .
- $a_i$  es la distancia a lo largo de  $X_i$  hasta el origen del sistema  $i$ -ésimo.
- $\alpha_i$  es el ángulo entre el eje  $Z_{i-1}$  y el eje  $Z_i$ , medido en un plano perpendicular al eje  $X_i$ .

La Figura 1.2 muestra la asignación de los ejes coordenados para estimar los parámetros Denavit-Hartenberg.

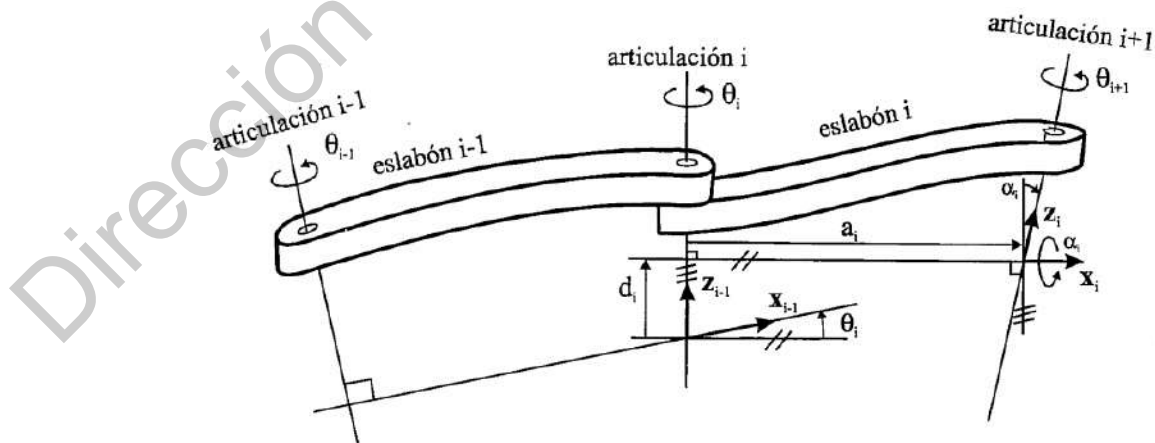


Figura 1.2: Parámetros Denavit-Hartenberg, imagen obtenida de [3].

### 1.4.2. Matrices de transformación homogénea

Tras obtenerse los parámetros de las articulaciones, se crea una matriz de transformación homogénea elemental para cada articulación con la siguiente matriz:

$${}^{i-1}A_i = \begin{bmatrix} c_i & -s_i c_{\alpha_i} & s_i s_{\alpha_i} & a_i c_i \\ s_i & c_i c_{\alpha_i} & -c_i s_{\alpha_i} & a_i s_i \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

donde  $c_i \equiv \cos(\theta_i)$ ,  $s_i \equiv \sin(\theta_i)$ ,  $c_{\alpha_i} \equiv \cos(\alpha_i)$  y  $s_{\alpha_i} \equiv \sin(\alpha_i)$ . Los parámetros  $a_i$  y  $d_i$  son las distancias descritas anteriormente.

La multiplicación de las matrices genera la posición y orientación relativas desde el primer eslabón hasta el  $i$ -ésimo, es decir,

$${}^0A_i = {}^0A_1 {}^1A_2 {}^2A_3 \dots {}^{i-1}A_i \tag{1.3}$$

La matriz de transformación homogénea  ${}^0A_n$  es de especial utilidad, pues describe la posición y orientación del actuador final. También se le conoce como matriz  ${}^0T_n$ .

Se define como matriz de transformación homogénea  $T$  a una matriz de dimensión  $4 \times 4$  que representa la transformación de un vector de coordenadas homogéneas de un sistema de coordenadas a otro [3].

$$T = {}^{i-1}A_i = \begin{bmatrix} R_{3 \times 3} & P_{3 \times 1} \\ f_{1 \times 3} & w_{1 \times 1} \end{bmatrix} = \begin{bmatrix} \text{Rotación} & \text{Traslación} \\ \text{Perspectiva} & \text{Escalado} \end{bmatrix} \tag{1.4}$$

### 1.4.3. Análisis cinemático de posición

La cinemática del robot estudia el movimiento del mismo con respecto a un sistema de referencia. Así, la cinemática se interesa por la descripción analítica del movimiento espacial del robot como una función del tiempo, y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares [3].

Existen dos problemas fundamentales a resolver en la cinemática del robot; el primero de ellos se conoce como el problema cinemático directo de posición, y consiste en determinar cuál es la posición y orientación del actuador final del robot, con respecto a un sistema de coordenadas que se toma como referencia, en donde se suponen conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot; el segundo, denominado problema cinemático inverso de posición, resuelve la configuración que debe adoptar el robot para una posición y orientación del extremo conocidas [3].

La Figura 1.3 muestra un diagrama de la relación entre ambos problemas.

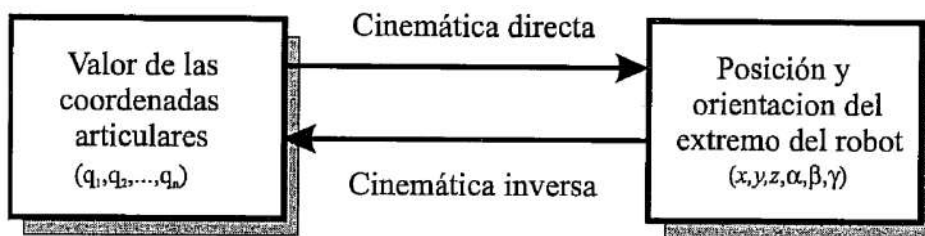


Figura 1.3: Relación entre cinemática directa e inversa, imagen obtenida de [3].

### 1.4.3.1. Análisis cinemático directo de posición

El problema cinemático directo se reduce a encontrar una matriz homogénea de transformación  $T$  que relaciona la posición y orientación del extremo del robot respecto del sistema de referencia fijo situado en la base del mismo. Esta matriz  $T$  será la función de las coordenadas articulares [3].

Para un robot serial de  $n$  GDL, la matriz de transformación homogénea que describirá la posición y orientación del actuador final con respecto al sistema de referencia se obtiene mediante:

$${}^0T_n = {}^0A_1 {}^1A_2 \dots {}^{n-2}A_{n-1} {}^{n-1}A_n \quad (1.5)$$

En este trabajo,  $n = 2$ , por lo que (1.5) se reduce a:

$${}^0T_2 = {}^0A_1 {}^1A_2 = \begin{bmatrix} & & x \\ R & & y \\ & & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.6)$$

Por lo que, la orientación del actuador final para el robot de este trabajo de tesis carece de sentido debido a su sencillez, y las coordenadas cartesianas  $x$  y  $y$  denotan la posición del extremo final del segundo eslabón con respecto al marco de referencia fijo. Dicho esto, se puede concluir que las coordenadas de posición dependen de las coordenadas articulares  $q_1$  y  $q_2$ . La relación entre ellas define el modelo cinemático directo de posición para este robot.

$$x = f_x(q_1, q_2) \quad (1.7)$$

$$y = f_y(q_1, q_2) \quad (1.8)$$

### 1.4.3.2. Análisis cinemático inverso de posición

El objetivo del problema cinemático inverso consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot para que su extremo se posicione y oriente según una determinada localización espacial [3].

Para el robot de este trabajo se define el modelo cinemático inverso de la siguiente manera:

$$q_1 = f_1(x, y) \quad (1.9)$$

$$q_2 = f_2(x, y) \quad (1.10)$$

El interés práctico del modelo cinemático inverso consiste en su empleo para obtener las especificaciones de posiciones articulares deseadas  $q_{d1}$  y  $q_{d2}$  a partir de especificaciones de posición deseada  $X_d$  y  $Y_d$  del actuador final del robot. A partir de esta información, pueden obtenerse las posiciones, velocidades y aceleraciones articulares deseadas. Cabe resaltar también que la cinemática inversa es de gran utilidad en la planificación de trayectorias.

### 1.4.4. Matriz Jacobiana

Es posible conocer las velocidades involucradas de un robot, es decir, las velocidades articulares y la velocidad lineal y angular del actuador final. Así, el sistema de control del robot debe establecer qué velocidades debe imprimir a cada articulación (a través de sus respectivos actuadores) para conseguir que el extremo desarrolle una trayectoria temporal concreta, por ejemplo, una línea recta a velocidad constante [3].

Para este y otros fines, es de gran utilidad disponer de la relación entre las velocidades de las coordenadas articulares y las de la posición y orientación del extremo del robot. La relación entre ambos vectores de velocidad se obtiene a través de la denominada matriz Jacobiana [3].

La matriz Jacobiana directa permite conocer las velocidades del extremo del robot a partir de los valores de las velocidades de cada articulación. Por su parte, la matriz Jacobiana inversa permitirá conocer las velocidades articulares necesarias para obtener unas velocidades determinadas en el actuador final del robot [3].

#### 1.4.4.1. Construcción de la matriz Jacobiana

Se requieren parámetros de la matriz  $T$  obtenida en el análisis cinemático directo de posición para generar la matriz Jacobiana, la cual se obtiene a partir de la siguiente ecuación:

$$\begin{bmatrix} J_{P_i} \\ J_{\theta_i} \end{bmatrix} = \begin{cases} \begin{bmatrix} Z_{i-1} \\ 0 \end{bmatrix} & \text{articulación prismática} \\ \begin{bmatrix} Z_{i-1} \times (P_e - P_{i-1}) \\ Z_{i-1} \end{bmatrix} & \text{articulación de revolución} \end{cases} \quad (1.11)$$

de la cual se utilizará para este trabajo el caso para articulaciones de revolución.

La matriz  $T$  de transformación se suele escribir de la siguiente forma:

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n & o & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.12)$$

donde  $n, o, a$  es una terna ortogonal que representa la orientación y  $p$  es un vector que representa la posición [3].

Por lo tanto, en la ecuación (1.11),  $Z_i$  representa el vector  $a$  de la matriz de transformación homogénea  ${}^0A_i$ ;  $P_i$  el vector de pose de  ${}^0A_i$ ; y  $P_e$  es la posición del actuador final, encontrado en  ${}^0T_n$ . Todos estos vectores de tamaño  $3 \times 1$ . El tamaño del Jacobiano es de  $6 \times n$ .

Para el cálculo de las velocidades se utiliza

$$\bar{V}_e = J(\bar{q})\dot{\bar{q}} = \begin{bmatrix} \dot{P}_e \\ \bar{\omega}_e \end{bmatrix} \quad (1.13)$$

donde  $\dot{P}_e$  representa la velocidad lineal y  $\bar{\omega}_e$  la velocidad angular del actuador final, ambos de tamaño  $3 \times 1$ .

Con la inversa del Jacobiano, se calculan las velocidades articulares a partir del vector de velocidad del actuador final, es decir, se puede obtener una solución cerrada de velocidad

$$\dot{\bar{q}} = [J(\bar{q})]^{-1}\bar{V}_e \quad (1.14)$$



Es necesario que el Jacobiano sea invertible, es decir, que en primera instancia, sea cuadrado. De no ser así, se recurre a la pseudo inversa izquierda (para robots cuyos GDL sean menores a 6) y pseudo inversa derecha (para más de 6 GDL).

Por otro lado, se requiere que no exista dependencia lineal en sus ecuaciones para que sea invertible. De ser así, físicamente, el robot se encontraría en una singularidad, lo que se traduce en velocidades muy altas, que a su vez genera exigencia de potencia, peligrosa para el entorno y el robot mismo.

Para comprobar si el Jacobiano es invertible, se calcula el determinante de la matriz y se encuentran las combinaciones de ángulos que resulten en cero

$$|J(\bar{q})| = 0 \quad (1.15)$$

### 1.4.5. Selección de tolerancias de maquinado

Los diferentes procesos de fabricación tienen grados de exactitud mínimos posibles, dependiendo del tamaño del trabajo, del estado del equipo y la habilidad del operador. En general, se recomiendan las siguientes tolerancias para cotas que no tengan efecto sobre la función de la pieza [7]:

- Dimensiones de 0 a 150mm:  $\pm 0.4mm$ .
- Dimensiones de 150 a 460mm:  $\pm 0.8mm$ .
- Dimensiones de 460mm en adelante:  $\pm 1.6mm$  o más.

## 1.5. Objetivos e hipótesis

### 1.5.1. Objetivo general

Desarrollar, analizar e instrumentar un robot plano serial R||R, mediante una Raspberry Pi® para controlar el movimiento de sus motores tipo Dynamixel, que será utilizado con fines didácticos.

### 1.5.2. Objetivos particulares

Los objetivos particulares son los siguientes:

- Realizar un análisis estático del sistema para el dimensionamiento adecuado de los eslabones.
- Maquinar y ensamblar las piezas que conformarán al robot serial plano R||R.
- Realizar análisis cinemático directo e inverso de posición y velocidad.
- Implementar un controlador cinemático a partir de la solución cerrada del análisis cinemático inverso de posición.
- Desarrollar una interfaz de usuario para la manipulación del robot.

### 1.5.3. Hipótesis

Se implementa un control de posición y planificación de trayectorias para un robot serial planar de 2 GDL con tan sólo los conocimientos adquiridos durante la materia de Robótica impartida en la Facultad de Ingeniería.

## 1.6. Equipo, materiales y software

Los materiales y software utilizados para el desarrollo del robot serial plano R||R son los siguientes:

- 2 motores Dynamixel MX-28.
- Raspberry Pi 3 Model B+®.
- PCB con circuito mínimo para integrado 74LS241.
- Elementos estructurales.
- Tornillería.
- SolidWorks®.
- MATLAB®.

## Capítulo 2

# Desarrollo del robot serial plano R||R

En este capítulo se presenta la descripción de cada una de las etapas que se emplearon para el desarrollo del robot serial plano R||R.

### 2.1. Descripción de hardware del robot serial plano R||R

En esta sección se presenta la descripción del hardware empleado para el desarrollo del robot serial plano R||R.

#### 2.1.1. Características servomotor Dynamixel MX-28

Para este proyecto de tesis fueron asignados servomotores Dynamixel MX-28 [8]. Este modelo se muestra en la Figura 2.1.



Figura 2.1: Motor Dynamixel MX-28.

De la hoja de especificación se resaltan las características de la Tabla 2.1:

Tabla 2.1: Especificaciones del servomotor Dynamixel MX-28.

| Característica     | Especificación        |
|--------------------|-----------------------|
| Resolución         | 4096 pul/rev          |
| Modo de operación  | Modo Joint (0 - 360°) |
| Peso               | 72g                   |
| Dimensiones        | 35.6×50.6 × 35.5mm    |
| Par                | 2.5 Nm (a 12V, 1.4A)  |
| Voltaje de entrada | 10.0 - 14.8V          |

El valor de torque a 12V indicado en la Tabla 2.1 es importante para determinar, junto al análisis estático, si de acuerdo a las dimensiones y peso de los componentes mecánicos del robot, no se sobrepasan las capacidades de los servomotores.

### 2.1.2. Raspberry Pi 3 Model B+®

Raspberry Pi® es un mini ordenador compuesto por puertos USB, HDMI, Ethernet y pines GPIO que lo convierten en una plataforma ideal para proyectos de robótica por su versatilidad. En este caso, se utiliza el modelo 3 B+ [9], mostrado en la Figura 2.2. Este modelo cuenta con 1GB de memoria RAM, 40 pines GPIO, ranura SD, entre otras características que permitirán ampliar el alcance inicial de este proyecto con el acondicionamiento de nuevos componentes, tales como sensores, cámaras o motores adicionales.

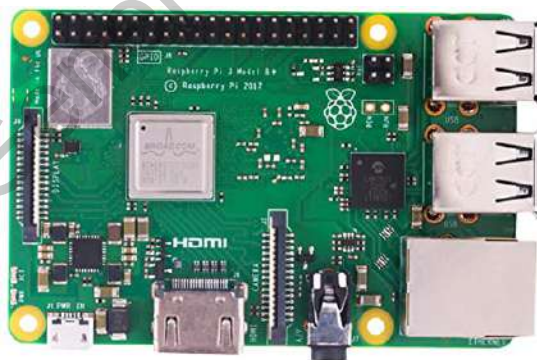


Figura 2.2: Raspberry Pi 3 Model B+®.

## 2.2. Análisis estático para la determinación de parámetros geométricos

Previo al análisis estático, se propusieron dimensiones de los eslabones que formarían parte del robot serial plano R||R. En la Figura 2.3 se muestra el modelo sólido propuesto del robot, para la obtención de los parámetros físicos y geométricos empleados en el análisis estático. El tipo de material seleccionado es aluminio 6061 debido a sus propiedades mecánicas y a su accesibilidad.

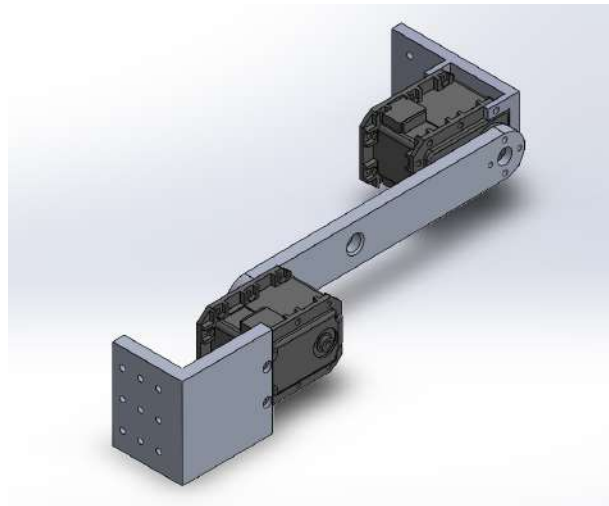


Figura 2.3: Subensamble de eslabones y servomotores.

En la Figura 2.4 se muestra el diagrama de cuerpo rígido empleado para el análisis estático, donde  $M_1$  y  $M_2$  son los pares de los servomotores Dynamixel MX-28;  $l_1$  y  $l_2$  son las longitudes de los eslabones 1 y 2, respectivamente;  $l_{c1}$  y  $l_{c2}$  son las distancias del centro de los motores al centro de su respectivo eslabón;  $W_{E1}$  y  $W_{E2}$  son los pesos de los eslabones 1 y 2, respectivamente, y  $W_{M2}$  es el peso del motor 2.

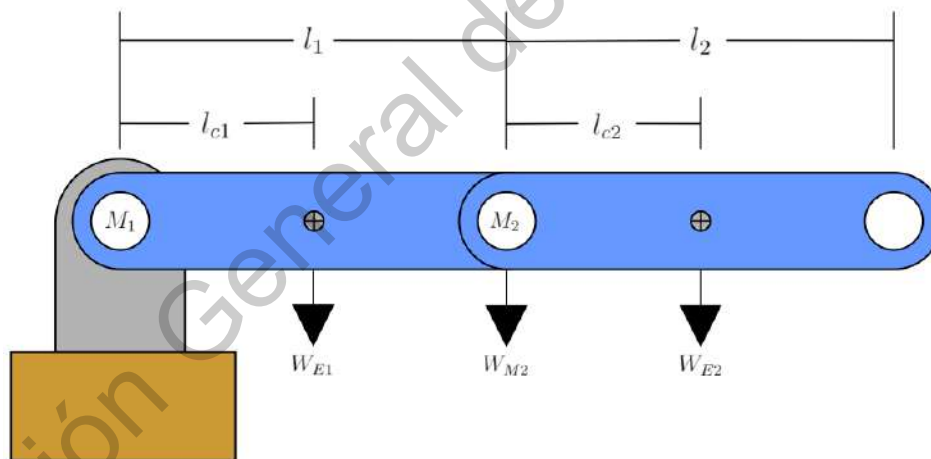


Figura 2.4: Modelo estático del robot serial plano R||R.

Los valores obtenidos del modelo en SolidWorks® del robot serial plano R||R se muestran en la Tabla 2.2.

Tabla 2.2: Parámetros geométricos y físicos de los eslabones generados en SolidWorks®.

| Parámetro | Valor      |
|-----------|------------|
| $l_1$     | 0.173m     |
| $l_2$     | 0.0537m    |
| $l_{c1}$  | 0.0865m    |
| $l_{c2}$  | 0.02685m   |
| $W_{E1}$  | 0.4994271N |
| $W_{E2}$  | 0.3187269N |
| $W_{M2}$  | 0.75537N   |

A partir de la sumatoria de momentos con respecto a  $M_1$ , considerando el sentido positivo en contra de las manecillas del reloj, se obtiene

$$\sum M_1 = 0 \quad (2.1)$$

$$M_1 - W_{E1}l_{c1} - W_{M2}l_1 - W_{E2}(l_1 + l_{c2}) = 0 \quad (2.2)$$

$$M_1 = W_{E1}l_{c1} + W_{M2}l_1 + W_{E2}(l_1 + l_{c2}) \quad (2.3)$$

Sustituyendo los valores de la Tabla 2.2, y efectuando las operaciones necesarias para cada uno de los sumandos de la ecuación (2.3), se obtiene

$$W_{E1}l_{c1} = (0.05091\text{kg})(9.81\text{m/s}^2)(0.0865\text{m}) = 0.0432\text{Nm} \quad (2.4)$$

$$W_{M2}l_1 = (0.077\text{kg})(9.81\text{m/s}^2)(0.173\text{m}) = 0.1306\text{Nm} \quad (2.5)$$

$$W_{E2}(l_1 + l_{c2}) = (0.03249\text{kg})(9.81\text{m/s}^2)(0.173\text{m} + 0.02685\text{m}) = 0.06369\text{Nm} \quad (2.6)$$

Al sustituir los valores anteriores en la ecuación (2.3), resulta

$$M_1 = 0.2375\text{Nm} \quad (2.7)$$

Se repite el procedimiento descrito previamente, ahora con respecto a  $M_2$ , considerando igualmente el sentido positivo en contra de las manecillas del reloj, y se obtiene

$$\sum M_2 = 0 \tag{2.8}$$

$$M_2 - W_{E2}l_{c2} = 0 \tag{2.9}$$

$$M_2 = W_{E2}l_{c2} \tag{2.10}$$

Sustituyendo los valores de la Tabla 2.2 del modelo en SolidWorks® del robot serial plano R||R en la ecuación (2.10), se obtiene

$$M_2 = W_{E2}l_{c2} = (0.03249\text{kg})(9.81\text{m/s}^2)(0.02685\text{m}) = 0.008557\text{Nm} \tag{2.11}$$

Con los resultados obtenidos en cada motor se puede afirmar que

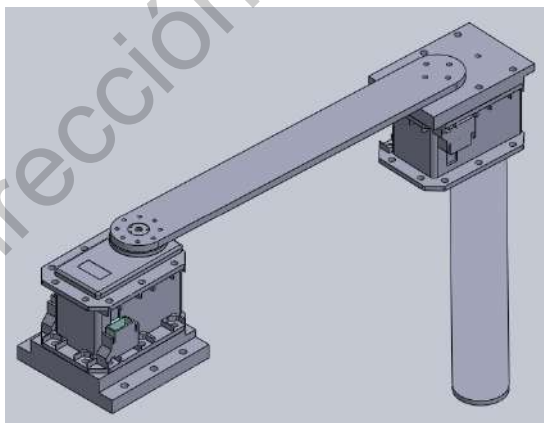
$$M_1 < 2.5\text{Nm} \tag{2.12}$$

$$M_2 < 2.5\text{Nm} \tag{2.13}$$

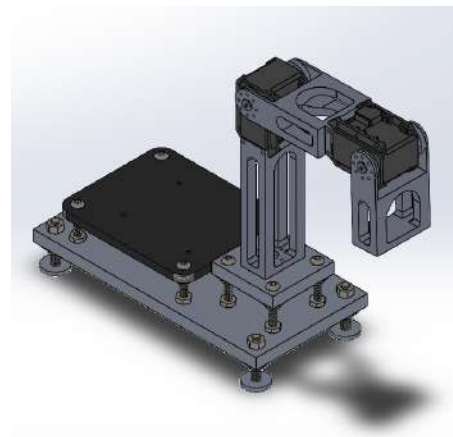
Por ende, y con base en el modelo en 3D propuesto, se puede concluir que ambos motores son aptos mecánicamente para su aplicación en este proyecto de tesis.

### 2.3. Diseño de los elementos estructurales del robot serial plano R||R

En la Figura 2.5 se muestran dos prototipos que fungieron como base del diseño final.



(a) Prototipo preliminar 1.



(b) Prototipo preliminar 2.

Figura 2.5: Prototipos preliminares.

En la Figura 2.5a se muestra un modelo horizontal donde ambos motores son capaces de dar giros de 360°, lo que expande la movilidad de los eslabones.

En la Figura 2.5b se muestra un modelo vertical donde ambos motores giran aproximadamente 270°. Además, posee eslabones con dimensiones similares a las de los servomotores.

Haciendo un análisis global, se optó por tomar características de ambos definiendo así un prototipo final, mostrado en la Figura 2.6.

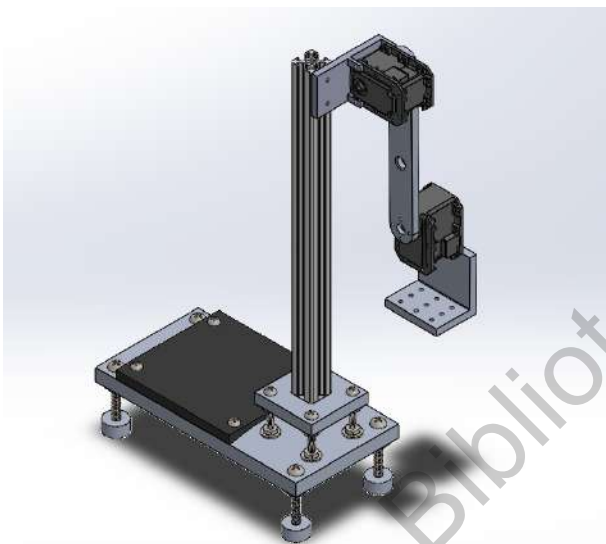


Figura 2.6: Prototipo final.

Del Prototipo 1 se tomó el diseño del primer eslabón y su capacidad de girar los 360°, además, se modificó para que el punto de ensamble fuera el cople del motor 2 y evitar así la manufactura de una pieza adicional. Del Prototipo 2, se tomó la base y orientación de los motores. Para el prototipo final, se contempló el uso de un perfil Bosch como soporte para darle altura y una base extendida para evitar colisiones cuando el mecanismo esté en movimiento.

Para mayor detalle, los planos de las piezas del diseño final se encuentran en el Apéndice A.1.

## 2.4. Manufactura de los elementos estructurales

Para la construcción del robot, se maquinaron las piezas utilizando una fresadora manual de 3 ejes y manteniendo una tolerancia  $\pm 0.4mm$ . Se muestran evidencias de este proceso.

En la Figura 2.7 se observa el barrenado de la base del perfil Bosch. En la Figura 2.8 se observa la realización del acabado final de la base principal, de manera que la pieza presente una mejor estética.





Figura 2.7: Barrenado de la base del perfil Bosch.



Figura 2.8: Acabado final de la base principal.

En la Figura 2.9 se observa el escuadrado de la base del perfil Bosch. En la Figura 2.10 se observa el barrenado del soporte del motor 1.



Figura 2.9: Escuadrado de la base del perfil Bosch.

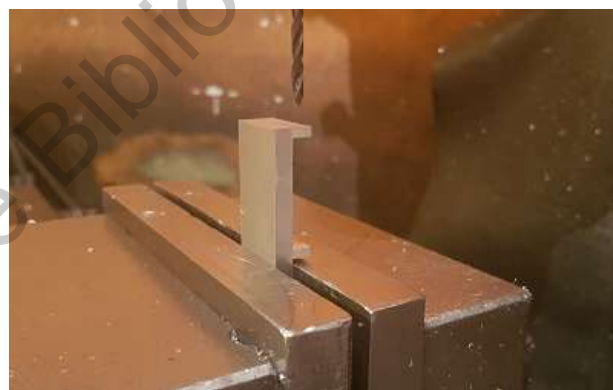


Figura 2.10: Barrenado del soporte del motor 1.

Tanto en la Figura 2.11 como en la Figura 2.12 se observa la operación de la fresadora manual.



Figura 2.11: Evidencia de maquinado.



Figura 2.12: Evidencia de maquinado.

## 2.5. Diseño electrónico del robot serial plano RllR

Debido a que la Raspberry Pi 3 Model B+<sup>®</sup> y los motores no comparten el mismo protocolo de comunicación, no es posible conectarlos directamente. Para lograr esta interacción, se diseñó una placa de circuito impreso (PCB, por sus siglas en inglés) en torno al circuito integrado 74LS241. Este componente es un buffer triestado, que sirve de intermediario entre las terminales de recepción y transmisión de la Raspberry Pi 3 Model B+<sup>®</sup> y el pin data de los servomotores conectados en serie. A continuación, se comparte los componentes utilizados:

- 1 74LS241
- 1 Conector 12V
- 1 Interruptor
- 1 Resistencia 10k $\Omega$
- 1 Resistencia 2.2k $\Omega$
- 1 Resistencia 330 $\Omega$
- 1 Tira de pines troquelados
- 1 Conector Molex de 3 pines
- 1 Conector Molex de 6 pines
- 2 LEDs

El diagrama eléctrico y el diseño de la PCB se muestran la Figura 2.13 y en la Figura 2.14, respectivamente. Para la validación de este diseño, se montaron los componentes en una protoboard y se desarrolló un código de prueba en lenguaje C++ por medio del software Geany en la Raspberry Pi 3 Model B+<sup>®</sup>. Una vez que se comprobó su correcto funcionamiento, se fabricó la PCB y se soldaron los componentes respectivos. La tarjeta electrónica final se muestra en la Figura 2.14b.

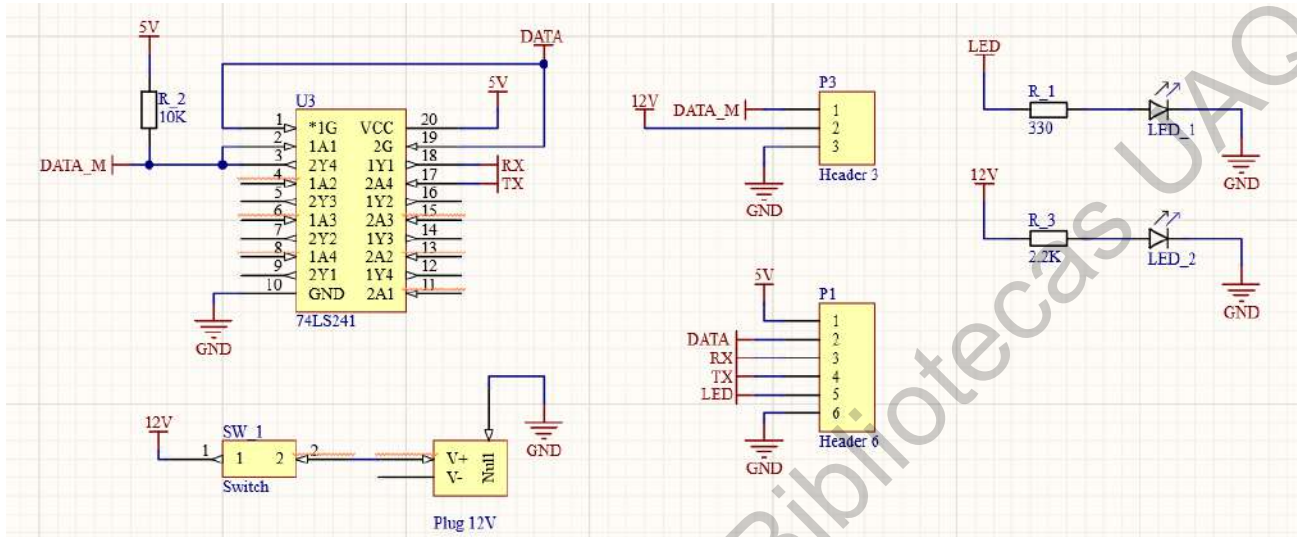
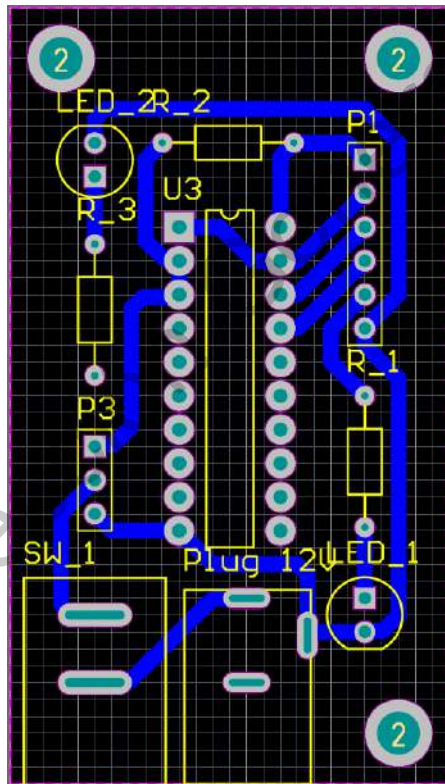


Figura 2.13: Diagrama eléctrico.



(a) Diseño PCB.



(b) PCB final.

Figura 2.14: Placa de circuito impreso.

# Capítulo 3

## Análisis cinemático del robot serial plano R||R

En este capítulo se presenta el análisis cinemático del robot plano R||R, abarcando el análisis de posición y el análisis de velocidad.

### 3.1. Análisis cinemático de posición del robot serial plano R||R

En esta sección se presenta el análisis cinemático de posición del robot serial plano R||R, abarcando el análisis directo y el análisis inverso, así como la obtención de la solución cerrada del sistema.

#### 3.1.1. Análisis cinemático de posición directo del robot serial plano R||R

La obtención de un conjunto de ecuaciones que relacionen la posición final debe partir de un análisis de parámetros Denavit-Hartenberg. Se establecen los ejes coordenados de referencia como se muestra en la Figura 3.1.

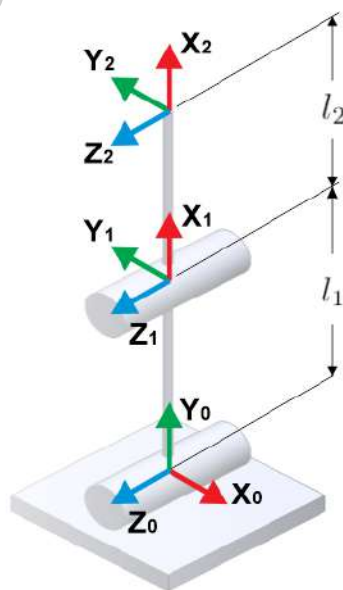


Figura 3.1: Configuración de referencia.

Los parámetros Denavit-Hartenberg obtenidos se encuentran en la Tabla 3.1.

Tabla 3.1: Parámetros Denavit-Hartenberg del robot de dos grados de libertad.

| $A_i$ | $\theta_i$ | $d_i$ | $a_i$ | $\alpha_i$ |
|-------|------------|-------|-------|------------|
| 1     | $\theta_1$ | 0     | $l_1$ | 0          |
| 2     | $\theta_2$ | 0     | $l_2$ | 0          |

Con base en los parámetros Denavit-Hartenberg, las matrices de transformación homogénea elementales, en función de las variables articulares, están dadas por

$${}^0A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & l_1 c_1 \\ s_1 & c_1 & 0 & l_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.1)$$

$${}^1A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & l_2 c_2 \\ s_2 & c_2 & 0 & l_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2)$$

donde  $s_i = \sin \theta_i$  y  $c_i = \cos \theta_i$ , para  $i = 1, 2$ .

Al multiplicar las ecuaciones (3.1) y (3.2) se obtiene la matriz  $T$ , que representa la transformación del marco 2, situado en el actuador final, respecto al marco 0, que es el sistema de referencia fijo.

$$T = {}^0A_2 = \begin{bmatrix} c_1 c_2 - s_1 s_2 & -c_1 s_2 - s_1 c_2 & 0 & l_1 c_1 + l_2 c_1 c_2 - l_2 s_1 s_2 \\ c_1 s_2 + s_1 c_2 & c_1 c_2 - s_1 s_2 & 0 & l_1 s_1 + l_2 c_1 s_2 + l_2 s_1 c_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3)$$

### 3.1.2. Análisis cinemático de posición inverso del robot serial plano R1R2

A partir de la definición de la matriz  $T$  y de la ecuación (1.4), se toma el vector  $\bar{P}$  de la ecuación (3.3), el cual proporciona la posición del actuador final del robot en términos de las variables articulares:

$$\bar{P} = \begin{bmatrix} l_1 c_1 + l_2 c_1 c_2 - l_2 s_1 s_2 \\ l_1 s_1 + l_2 c_1 s_2 + l_2 s_1 c_2 \\ 0 \end{bmatrix} \quad (3.4)$$

La posición del actuador final del robot será exclusivamente dentro del plano  $X$ - $Y$ , ya que  $P_z = 0$ . La solución cerrada del robot se presenta a continuación:

- Variable  $\theta_1$

A partir de la ecuación (3.4) se puede definir que

$$l_1 c_1 + l_2 c_1 c_2 - l_2 s_1 s_2 = P_x \quad (3.5)$$

$$l_1 s_1 + l_2 c_1 s_2 + l_2 s_1 c_2 = P_y \quad (3.6)$$

dejando del lado izquierdo de la igualdad todos los términos que incluyan la variable  $\theta_2$  resulta

$$l_2 c_1 c_2 - l_2 s_1 s_2 = P_x - l_1 c_1 \quad (3.7)$$

$$l_2 c_1 s_2 + l_2 s_1 c_2 = P_y - l_1 s_1 \quad (3.8)$$

factorizando  $l_2$  de (3.7) y (3.8)

$$l_2 [c_1 c_2 - s_1 s_2] = P_x - l_1 c_1 \quad (3.9)$$

$$l_2 [c_1 s_2 + s_1 c_2] = P_y - l_1 s_1 \quad (3.10)$$

aplicando identidades trigonométricas se tiene que

$$\cos \theta_1 \cos \theta_2 - \sin \theta_1 \sin \theta_2 = \cos(\theta_1 + \theta_2) \quad (3.11)$$

$$\cos \theta_1 \sin \theta_2 + \sin \theta_1 \cos \theta_2 = \sin(\theta_1 + \theta_2) \quad (3.12)$$

por lo tanto, sustituyendo (3.11) y (3.12) en (3.9) y (3.10), respectivamente, se tiene que

$$l_2 c_{12} = P_x - l_1 c_1 \quad (3.13)$$

$$l_2 s_{12} = P_y - l_1 s_1 \quad (3.14)$$

donde  $c_{12} = \cos(\theta_1 + \theta_2)$  y  $s_{12} = \sin(\theta_1 + \theta_2)$ .

Al elevar al cuadrado (3.13) y (3.14) se obtiene

$$l_2^2 c_{12}^2 = P_x^2 - 2P_x l_1 c_1 + l_1^2 c_1^2 \quad (3.15)$$

$$l_2^2 s_{12}^2 = P_y^2 - 2P_y l_1 s_1 + l_1^2 s_1^2 \quad (3.16)$$

al sumar las ecuaciones (3.15) y (3.16), y factorizando resulta

$$l_2^2 (s_{12}^2 + c_{12}^2) = P_x^2 + P_y^2 - 2P_x l_1 c_1 - 2P_y l_1 s_1 + l_1^2 (s_1^2 + c_1^2) \quad (3.17)$$

aplicando identidades pitagóricas en (3.17) se obtiene

$$l_2^2 = P_x^2 + P_y^2 - 2P_x l_1 c_1 - 2P_y l_1 s_1 + l_1^2 \quad (3.18)$$

al reordenar y agrupar se tiene que

$$(-2P_x l_1)c_1 + (-2P_y l_1)s_1 + (P_x^2 + P_y^2 + l_1^2 - l_2^2) = 0 \quad (3.19)$$

por ende, se puede definir que

$$a = -2P_x l_1 \quad (3.20)$$

$$b = -2P_y l_1 \quad (3.21)$$

$$c = P_x^2 - P_y^2 + l_1^2 - l_2^2 \quad (3.22)$$

por consecuencia, al sustituir resulta

$$ac_1 + bs_1 + c = 0 \quad (3.23)$$

aplicando identidades trigonométricas se tiene que

$$\cos_i = \frac{1 - \tan^2\left(\frac{\theta_i}{2}\right)}{1 + \tan^2\left(\frac{\theta_i}{2}\right)} \quad (3.24)$$

$$\sin_i = \frac{2 \tan\left(\frac{\theta_i}{2}\right)}{1 + \tan^2\left(\frac{\theta_i}{2}\right)} \quad (3.25)$$

por lo tanto, sustituyendo (3.24) y (3.25) en (3.23) se tiene que

$$a \left[ \frac{1 - \tan^2\left(\frac{\theta_1}{2}\right)}{1 + \tan^2\left(\frac{\theta_1}{2}\right)} \right] + b \left[ \frac{2 \tan\left(\frac{\theta_1}{2}\right)}{1 + \tan^2\left(\frac{\theta_1}{2}\right)} \right] + c = 0 \quad (3.26)$$

multiplicando la ecuación (3.26) por  $[1 + \tan^2\left(\frac{\theta_1}{2}\right)]$  se llega a

$$a \left[ 1 - \tan^2\left(\frac{\theta_1}{2}\right) \right] + b \left[ 2 \tan\left(\frac{\theta_1}{2}\right) \right] + c \left[ 1 + \tan^2\left(\frac{\theta_1}{2}\right) \right] = 0 \quad (3.27)$$

expandiendo

$$a - a \left[ \tan^2\left(\frac{\theta_1}{2}\right) \right] + 2b \left[ \tan\left(\frac{\theta_1}{2}\right) \right] + c + c \left[ \tan^2\left(\frac{\theta_1}{2}\right) \right] = 0 \quad (3.28)$$

reordenando para generar una ecuación cuadrática general de la forma  $ax^2 + bx + c = 0$

$$(c - a) \left[ \tan^2\left(\frac{\theta_1}{2}\right) \right] + 2b \left[ \tan\left(\frac{\theta_1}{2}\right) \right] + (a + c) = 0 \quad (3.29)$$

de esta manera se puede solucionar la ecuación para  $\tan\left(\frac{\theta_1}{2}\right)$ , a través de la fórmula general para ecuaciones cuadráticas

$$\tan\left(\frac{\theta_1}{2}\right) = \frac{-2b \pm \sqrt{4b^2 - 4(c^2 - a^2)}}{4(c - a)} \quad (3.30)$$

tras despejar, se obtiene la solución de  $\theta_1$

$$\theta_1 = 2 \tan^{-1} \left[ \frac{-2b \pm \sqrt{4b^2 - 4(c^2 - a^2)}}{4(c - a)} \right] \quad (3.31)$$

■ Variable  $\theta_2$

Dividiendo (3.14) entre (3.13) se obtiene

$$\tan(\theta_1 + \theta_2) = \frac{P_y - l_1 s_1}{P_x - l_1 c_1} \quad (3.32)$$

despejando de la ecuación anterior se obtiene la solución de  $\theta_2$ , considerando que se obtuvo el resultado de  $\theta_1$  previamente en la ecuación (3.31)

$$\theta_2 = \tan^{-1} \left[ \frac{P_y - l_1 s_1}{P_x - l_1 c_1} \right] - \theta_1 \quad (3.33)$$

## 3.2. Análisis cinemático de velocidad del robot serial plano R||R

En esta sección se presenta el análisis cinemático de velocidad del robot serial plano R||R, abarcando el análisis directo y el análisis inverso, así como la obtención de la solución cerrada del sistema.

### 3.2.1. Análisis cinemático de velocidad directo del robot serial plano R||R

Como se puede observar en las ecuaciones (1.13) y (1.14), para realizar un análisis cinemático de velocidad es necesario calcular la matriz Jacobiana, ya que es requerida tanto para el método directo como para el inverso. Para obtener dicha matriz se requieren las matrices  ${}^0A_1$  y  ${}^0A_2$ , las cuales corresponden a las ecuaciones (3.1) y (3.3), respectivamente.

Extrayendo la información necesaria de (3.1) y (3.3) para sustituir en (1.11), y tomando en cuenta que ambas variables corresponden a juntas de revolución, se obtiene

$$J_1 = \begin{bmatrix} {}^0\bar{Z}_0 \times ({}^0\bar{P}_2 - {}^0\bar{P}_0) \\ {}^0\bar{Z}_0 \end{bmatrix} = \begin{bmatrix} (0, 0, 1)^T \times (P_x, P_y, P_z)^T \\ (0, 0, 1)^T \end{bmatrix} = \begin{bmatrix} -P_y \\ P_x \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (3.34)$$

$$J_2 = \begin{bmatrix} {}^0\bar{Z}_1 \times ({}^0\bar{P}_2 - {}^0\bar{P}_1) \\ {}^0\bar{Z}_1 \end{bmatrix} = \begin{bmatrix} (0, 0, 1)^T \times (P_x - l_1 c_1, P_y - l_1 s_1, P_z)^T \\ (0, 0, 1)^T \end{bmatrix} = \begin{bmatrix} -(P_y - l_1 s_1) \\ P_x - l_1 c_1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (3.35)$$



$J_1$  y  $J_2$  representan cada una de las columnas que conforman la matriz Jacobiana del robot, por lo tanto, al sustituir las ecuaciones (3.5) y (3.6) en las ecuaciones (3.34) y (3.35), la matriz Jacobiana está dada por

$$J = \begin{bmatrix} -l_1 s_1 - l_2 c_1 s_2 - l_2 s_1 c_2 & -l_2 c_1 s_2 - l_2 s_1 c_2 \\ l_1 c_1 + l_2 c_1 c_2 - l_2 s_1 s_2 & l_2 c_1 c_2 - l_2 s_1 s_2 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \quad (3.36)$$

Por consiguiente, se puede realizar el análisis de velocidad directo, el cual está dado por

$$\bar{V}_e = J(\bar{q})\dot{\bar{q}} = \begin{bmatrix} \bar{P}_e \\ \bar{\omega}_e \end{bmatrix} \quad (3.37)$$

Lo cual permite obtener las velocidades del actuador final en términos de las velocidades articulares, donde  $\bar{P}_e$  representa la velocidad lineal y  $\bar{\omega}_e$  la velocidad angular.

### 3.2.2. Análisis cinemático de velocidad inverso del robot serial plano R1R2

Considerando que el robot es de dos grados de libertad y el actuador final sólo puede ubicarse libremente en el plano perpendicular a las direcciones de los ejes de las articulaciones cinemáticas de revolución, es posible realizar el análisis de velocidad con la submatriz correspondiente a la velocidad traslacional, la cual está dada por una submatriz  $J$ , la cual es

$$J_{22} = \begin{bmatrix} -l_1 s_1 - l_2 c_1 s_2 - l_2 s_1 c_2 & -l_2 c_1 s_2 - l_2 s_1 c_2 \\ l_1 c_1 + l_2 c_1 c_2 - l_2 s_1 s_2 & l_2 c_1 c_2 - l_2 s_1 s_2 \end{bmatrix} \quad (3.38)$$

Debido a la longitud de las ecuaciones solución, el análisis inverso de velocidad se realizó mediante software. Se adjunta el archivo *SolCerrVel.m* perteneciente a Matlab, cuyo código también se encuentra disponible en el Apéndice A.3.

# Capítulo 4

## Implementación del modelo cinemático y desarrollo de interfaz de usuario en el sistema embebido Raspberry Pi 3 Model B+<sup>®</sup>

En el presente capítulo, se presenta la implementación del modelo cinemático del robot plano R||R, así como el desarrollo de la interfaz de usuario en el sistema embebido Raspberry Pi 3 Model B+<sup>®</sup>.

### 4.1. Descripción de software para implementación del modelo cinemático

Se utilizó el software Geany para desarrollar un código que ejecutara el análisis cinemático del robot serial plano R||R, explicado y detallado en el Capítulo 3. A partir de la librería de motores Dynamixel, que se encuentra disponible en múltiples foros de programación, se escribieron las funciones que unen los cálculos con el movimiento de la plataforma.

Primeramente, para la transmisión de datos se utilizó el diseño electrónico mostrado en la Sección 2.5. Este circuito permite la comunicación entre los motores y la tarjeta Raspberry Pi 3 Model B+<sup>®</sup> con ayuda de la programación que se describirá en este capítulo. Como se puede observar en el circuito, existe un pin de datos en los motores, sobre el cual se transmite la información requerida para su control. Este pin debe habilitarse en la librería “*DynamixelSerial.cpp*” como se muestra en la función siguiente, la cual requiere de la dirección del pin, “*directionPin*”, que será usado para transmitir datos de la Raspberry Pi 3 Model B+<sup>®</sup> a los motores.

---

```
void DynamixelClass::begin(uint8_t directionPin)
{
    wiringPiSetup();
    Direction_Pin = directionPin;
    pinMode(Direction_Pin ,OUTPUT);
    fd=open("/dev/ttyS0",O_RDWR|O_NOCTTY|O_NDELAY);
    struct termios tty;
    s=tcgetattr(fd,&tty);
    a=cfsetispeed(&tty, B1000000);
```

```

        b=cfsetospeed(&tty, B1000000);
        tty.c_oflag = 0;
        c=tcsetattr(fd,TCSANOW,&tty);
        if(fd!=-1)
        {
            perror("Cant open port");
            return;
        }
    }
}

```

Varias funciones fueron creadas para cumplir con todas las tareas que se propusieron para la plataforma, a continuación, se explicarán las más importantes.

La función “*beginMovement*” declara e inicializa todas las variables que se utilizan en todo el código.

```

void calculateMovement::beginMovement(){
    l1=14.0;
    l2=8.3;
    aux1=0;
    aux2=0;
    x_n=0;
    y_n=0;
    a=0, b=0 , c=0;
    d=0, e=0 , f=0, g=0, h=0;
    t = 0, t1 = -20;
    count = 0;
    px=0, py=0;
    theta_1=0; //angle of first motor
    theta_2=0; //angle of second motor
    theta_1_2=0; //theta_1 + theta_2
    theta_1_hex=0x00; //theta_1 in hex
    theta_2_hex=0x00; //theta_2 in hex
    theta_1_motor=0; //angle for Dynamixel1
    theta_2_motor=0; //angle for Dynamixel2
    for (int j=0; j<200; j++){
        theta_1_hex_vec[j]=0;
        theta_2_hex_vec[j]=0;
    }
    speed=1023;
    flag=0;
    return;
}

```

La siguiente función calcula el valor del radio a partir de las coordenadas *px* y *py* ingresadas por el usuario para determinar si se puede trabajar a partir de los datos proporcionados.

```

float calculateMovement::calculateRadius(double PX, double PY){
    px = PX;
    py = PY;
    float r = sqrt(pow(px,2) + pow(py,2));
    return r;
}

```

Las siguientes funciones se utilizan para el posicionamiento del robot según se requiera, las cuales se describen a continuación:

1. Función “*positionHome*”: lleva los eslabones del robot a la posición de inicio, la cual representa valores de  $\theta_1 = 0^\circ$  y  $\theta_2 = 0^\circ$ .
2. Función “*positionTurnOff*”: lleva los eslabones a una posición segura para el robot antes de apagarlo.

---

```

void calculateMovement::positionHome(void){
    Dynamixel.moveSpeed(1,0,150);
    usleep(50000); // 1 //
    Dynamixel.moveSpeed(2,0,150);
    return;
}

void calculateMovement::positionTurnOff(void){
    Dynamixel.moveSpeed(1,30,150);
    usleep(90000);
    Dynamixel.moveSpeed(1,3071,120); // 2 //
    usleep(50000);
    Dynamixel.moveSpeed(2,0,120);
    return;
}

```

---

Para el método directo de posición se realizaron 3 funciones más, las cuales se observan y se describen a continuación:

1. Función “*moveDirect*”: lleva los motores a la posición introducida por el usuario con valores dentro del rango de 0-360°.
2. Función “*moveDirectTh1*”: lleva al motor 1, encargado de la variable  $\theta_1$ , a la posición introducida por el usuario con valores dentro del rango de 0-360°.
3. Función “*moveDirectTh2*”: lleva al motor 2, encargado de la variable  $\theta_2$ , a la posición introducida por el usuario con valores dentro del rango de 0-360°.

---

```

void calculateMovement::moveDirect(double theta_1, double theta_2){
    theta_1_hex = (theta_1 * 4095)/360;
    theta_2_hex = (theta_2 * 4095)/360;
    Dynamixel.moveSpeed(1,30,150); // 1 //
    usleep(90000);
    Dynamixel.moveSpeed(1,theta_1_hex,100);
    usleep(50000);
    Dynamixel.moveSpeed(2,theta_2_hex,100);
    return;
}

void calculateMovement::moveDirectTh1(double theta_1){
    theta_1_hex = (theta_1 * 4095)/360; // 2 //
    Dynamixel.moveSpeed(1,theta_1_hex,100);
    return;
}

```

```

    }

    void calculateMovement::moveDirectTh2(double theta_2){
        theta_2_hex = (theta_2 * 4095)/360;           // 3 //
        Dynamixel.moveSpeed(2,theta_2_hex,100);
        return;
    }

```

Es importante mencionar que estas funciones y las que se explican a continuación se complementan con el desarrollo de la interfaz que se explicará más adelante, ya que ambas partes van de la mano para la correcta implementación del modelo cinemático en la plataforma.

A continuación, se muestra la función “*moveInverse*”, que utiliza las ecuaciones (3.31) y (3.33) obtenidas del análisis cinemático de posición inverso para ejecutar el movimiento de los motores. Esta función se divide en los siguientes pasos:

1. Calcula  $\theta_1$  y  $\theta_2$ , con base en las coordenadas  $px$  y  $py$  ingresadas por el usuario.
2. Calcula ángulo conjugado para mantener rango de 0-360°, solo si es necesario.
3. Escala  $\theta_1$  y  $\theta_2$  al rango de resolución de los motores de 0-4095 unidades.
4. Llama al método “*moveSpeed*” incluyendo los valores obtenidos de  $\theta_1$  y  $\theta_2$ .

```

void calculateMovement::moveInverse(double px, double py, double& theta_1, double& theta_2,int sign){
    a = -2.0 * px * l1;
    b = -2.0 * py * l1;
    c = pow(px, 2.0) + pow(py, 2.0) + pow(l1, 2.0) - pow(l2, 2.0);
    d = pow(c, 2.0) - pow(a, 2.0);
    e = (-2.0 * b) + sign * sqrt(4.0 * pow(b, 2.0) - 4.0 * d);           // 1 //
    f = 2.0 * (c - a);
    theta_1 = 2.0 * atan2( e, f);
    g = l1 * sin(theta_1);
    h = l1 * cos(theta_1);
    theta_2 = atan2((py - g),(px - h)) - theta_1;
    theta_1 = double(degrees(theta_1));
    theta_2 = double(degrees(theta_2));

    if (theta_1 < 0){
        aux1 = theta_1;
        theta_1 = 360 + aux1;
    } // 2 //
    if (theta_2 < 0){
        aux2 = theta_2;
        theta_2 = 360 + aux2;
    }

    theta_1_motor = theta_1;
    theta_1_hex = (theta_1_motor * 4096)/360;
    theta_1_hex = int(theta_1_hex);

    theta_2_motor = theta_2;
    theta_2_hex = (theta_2_motor * 4096)/360; // 3 //
    theta_2_hex = int(theta_2_hex);
}

```

```

Dynamixel.moveSpeed(1,theta_1_hex,100);
usleep(50000); // 4 //
Dynamixel.moveSpeed(2,theta_2_hex,100);
return;
}

```

Es importante mencionar que la función anterior recibe además de los datos ingresados por el usuario, el dato *sign*, el cual será de gran ayuda para la orientación del codo en el robot, según lo desee el usuario. Esto será explicado con mayor detalle en la siguiente sección 4.2.

La función “*moveInverse*” es de gran importancia, ya que es la implementación de la solución cerrada de posición que se obtuvo en el capítulo previo, estos cálculos se repiten en varias funciones más del código, ya que se requieren para realizar todas las tareas de la plataforma. Por ejemplo, se adaptó en la función “*moveRecord*” para emular la programación punto a punto.

A continuación, se describe la función “*makingCircle*”, la cual se encarga de calcular y ejecutar la trayectoria de un círculo en 100 pasos, a partir de las coordenadas del centro del círculo ( $px, py$ ) y el radio  $r$  ingresados por el usuario. La primera sección se divide de la siguiente manera:

1. Inicializa ciclo *for* para ejecutar la secuencia 100 veces. Dentro de esta secuencia se calcula un par de coordenadas ( $x_y, y_n$ ) a partir de los datos ingresados por el usuario. Estos valores son almacenados en vectores.
2. Calcula el radio y analiza si el robot es físicamente capaz de alcanzar esas posiciones.
3. Calcula  $\theta_1$  y  $\theta_2$ , con base en las coordenadas ( $x_y, y_n$ ).

```

int calculateMovement::makingCircle(double px, double py, double r, int sign){
for(int i=0;i<=99;i++)
{
    aux1=cos(t+PI);        aux2=sin(t+PI);        // 1 //
    x_n=px+r*aux1;        y_n=py+r*aux2;
    x_n_vec[i]=x_n;        y_n_vec[i]=y_n;

    float rad = Movement.calculateRadius(x_n,y_n);
    if(rad > 22.3 || rad<5.7)
    {
        flag=1;
        i = 99;        // 2 //
    }
    if(rad <= 22.3 && rad>=5.7)
    {
        flag=2;

a=pow(x_n,2.0)+pow(y_n,2.0)+pow(l1,2.0)-pow(l2,2.0)+(2.0*x_n*l1);
b=-4 * y_n * l1;
c=pow(x_n,2.0)+pow(y_n,2.0)+pow(l1,2.0)-pow(l2,2.0)-(2.0*x_n*l1); // 3 //

theta_1 = 2 * atan2((-b + sign * sqrt(pow(b,2)-(4*a*c))), (2*a));
theta_1_2 = atan2((y_n - (l1*sin(theta_1))), (x_n-(l1 * cos(theta_1))));
theta_2 = theta_1_2 - theta_1;
theta_1 = int(degrees(theta_1));
theta_2 = int(degrees(theta_2));
}
}
}

```

La segunda sección de esta función se muestra y se explica a continuación:

4. Utiliza una condición para evitar que  $\theta_1$  y  $\theta_2$  sean negativos; si lo son, se calcula el ángulo conjugado de los valores obtenidos para hacerlos positivos.
5. Convierte el valor de las variables a cuentas por revolución, ya que los cálculos previos arrojan valores en grados. Los valores resultantes se almacenan en vectores. Finalmente se observa el incremento en  $t$  para la siguiente iteración.

---

```

if (theta_1 < 0){
    aux1 = theta_1;
    theta_1 = 360 + aux1;
}
if (theta_2 < 0){           // 4 //
    aux2 = theta_2;
    theta_2 = 360 + aux2;
}

theta_1_motor = theta_1;
theta_2_motor = theta_2;
theta_1_hex = (theta_1_motor * 4096)/360;           // 5 //
theta_2_hex = (theta_2_motor * 4096)/360;
theta_1_hex_vec[i] = int(theta_1_hex);
theta_2_hex_vec[i] = int(theta_2_hex);
}
t=t+0.063466;
}

```

---

La última sección de la función “*makingCircle*” se muestra y se explica a continuación:

6. Si los parámetros ingresados por el usuario se encuentran dentro del rango de movimiento del robot, los motores recorren la trayectoria en 100 pasos.
7. Para la ejecución del primer paso, detiene el robot una fracción de segundo. El resto de los pasos son realizados con un tiempo menor entre ellos. Si el robot tiene que avanzar a una posición relativamente lejana respecto a la posición anterior, tendrá que esperar aproximadamente 4 segundos para permitir que los motores lleguen a ese punto.
8. Una vez concluida la trayectoria, el robot regresa a su posición inicial.

---

```

if (flag == 2){
    Dynamixel.moveSpeed(1,30,150);
    usleep(90000);

    for(int j=0;j<=99;j++){
        Dynamixel.moveSpeed(1,theta_1_hex_vec[j],200);
        usleep(50000);           // 6 //
        Dynamixel.moveSpeed(2,theta_2_hex_vec[j],200);
        usleep(100000); //velocidad de los pasos
    }
}

```

---

```
    if(j==0)
        { usleep(4000000); }
    else{
        if (theta_1_hex_vec[j] < theta_1_hex_vec[j-1] &&
            theta_1_hex_vec[j-2] < theta_1_hex_vec[j-1])    // 7 //
            {usleep(3000000);}
        else
            usleep(500);
    }

    }
    usleep(2000000);
    Dynamixel.moveSpeed(1,0,150);    // 8 //
    usleep(50000);
    Dynamixel.moveSpeed(2,0,150);
    t=0;
}
return flag;
}
```

---

Además de la implementación de una trayectoria circular, se agregó también una para trayectoria elíptica, así como una para curvas especiales y una más para funciones algebraicas. Debido a la similitud y extensión de todas estas funciones, no se explicarán aquí, pero se pueden observar en el Anexo A.2, donde se encuentra el código completo de la plataforma.

## 4.2. Descripción de software para desarrollo de la interfaz de usuario

El software QT Creator se utilizó para desarrollar la interfaz humano-máquina que permite el control del robot. QT Creator es una plataforma libre con soporte para lenguaje C++ empleada principalmente para diseñar aplicaciones.

En la Figura 4.1, se muestra el software QT Creator. A la izquierda se pueden observar los tipos de herramientas disponibles para el diseño de una interfaz; espaciadores, botones, listas y cajas de texto son algunos ejemplos. En el centro se encuentra el visualizador del proyecto en desarrollo. A la derecha están las herramientas utilizadas y las propiedades de cada una.



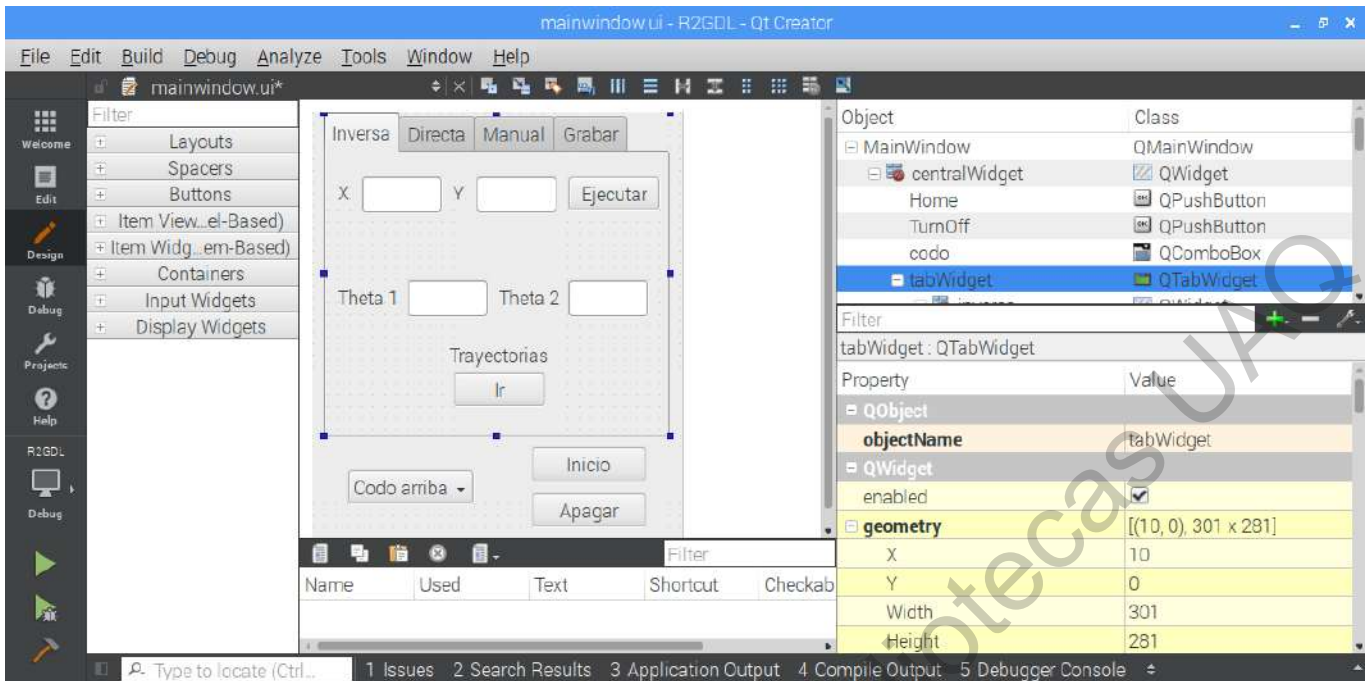


Figura 4.1: Interfaz software QT Creator.

En la Figura 4.2, se muestran 3 funciones principales ligadas a botones de la interfaz. La función “closeEvent”, ligada al botón de cierre de pestaña, apaga el LED indicador de envío de información y comanda la posición “TurnOff”. La función “onHomeClicked”, ligada al botón “Inicio”, enciende el LED indicador y lleva los motores a la posición “Home”. Y la función “onTurnOffClicked”, ligada al botón “Apagar”, apaga el LED indicador y los motores se mueven a la posición “TurnOff”.



Figura 4.2: Funciones generales de la interfaz.

En la Figura 4.3, se muestra la función “onExeInvClicked”, ligada al botón “Ejecutar”. Al inicio, captura las coordenadas ingresadas por el usuario y calcula el radio. En caso de que las dimensiones no

estén dentro del rango del movimiento del robot, se presenta la leyenda “Valores fuera de alcance”. En caso de ser capaz de alcanzar la posición deseada, envía la posición seleccionada del codo y las coordenadas a la función “*moveInverse*”, para obtener la solución cerrada de posición y a su vez, ejecutar el movimiento de los motores. Finalmente, muestra en la interfaz el mensaje “Valores correctos” junto con las expresiones resultantes y los valores de  $\theta_1$  y  $\theta_2$ .

```

void MainWindow::on_Exe_Inv_clicked()
{
    Dynamixel.turnOn(LED);
    usleep(50);
    Dynamixel.begin(MX);
    Movement.beginMovement();
    int signo;
    QString x = ui->coord_x->text();
    double PX = x.toDouble();
    QString y = ui->coord_y->text();
    double PY = y.toDouble();

    if(x!=""&&y!=""){
        float r = Movement.calculateRadius(PX,PY);
        double Theta1, Theta2;

        if(r > 22.3 || r<5.7)
        {
            ui->status->setText("Valores fuera de alcance");
            ui->lineEdit_theta1->setText("");
            ui->lineEdit_theta2->setText("");
        }

        if(r <= 22.3 && r>=5.7)
        {
            ui->status->setText("Valores correctos");
            if (ui->codo->currentIndex() ==0)
                signo = 1;
            else
                signo = -1;
            Movement.moveInverse(PX,PY,Theta1,Theta2,signo);
            QString theta1 = QString::number(Theta1);
            QString theta2 = QString::number(Theta2);
            ui->lineEdit_theta1->setText(theta1);
            ui->lineEdit_theta2->setText(theta2);
        }
    }
    else
        ui->status->setText("Ingresar valores");
}
}

```



Figura 4.3: Función “*onExeInvClicked*”.

En la Figura 4.4 se muestra una variante de la función “*Inversa*”, mencionada previamente. En este caso se utiliza la función “*onMakeRecord2Clicked*” que está ligada al botón “*Grabar*”. Al presionar este botón, la función almacena la posición capturada en dos vectores; uno para  $\theta_1$  y otro para  $\theta_2$ . Una vez que se hayan capturado todas las posiciones deseadas, se presiona el botón “*Ejecutar*” y el robot realiza la secuencia ingresada mediante la función “*onMakeRecordExe2clicked*”. La función “*onMakeRecordDelete2Clicked*”, borra los datos almacenados en los vectores e inicializa de nueva cuenta todas las variables en cero.



Figura 4.4: Función “onMakeRecord2Clicked”.

En la Figura 4.5, se observa la función “onExeDirClicked”, la cual está ligada a la pestaña “Directa” y a la acción de presionar el botón “Ejecutar”. Al presionarlo, la interfaz captura los datos proporcionados por el usuario y verifica que estos se encuentren dentro del alcance del robot. De no ser correctos aparecerá la leyenda “Valores fuera de alcance”, de lo contrario mostrará la leyenda “Valores correctos” y por ende se mandará llamar la función para realizar el movimiento de la plataforma a través del método directo. En este método también se considera la orientación del codo, por lo que es importante considerarlo al momento de hacer alguna prueba.



Figura 4.5: Función “onExeDirClicked”.

Este método se utilizó también en la pestaña “Manual”, para que el usuario pudiera manipular

el robot con la ayuda de las flechas de incremento y decremento. Para mayor detalle de esta función se puede ver el Anexo A.2. La aplicación de esta función se observa en el capítulo siguiente.

En la Figura 4.5 se observa la interfaz en la pestaña “Inversa”, ahí mismo se muestra el botón “Trayectorias”, el cual abre una ventana dedicada exclusivamente a diferentes tipos de trayectorias que se pueden implementar con el robot. Todas estas funciones son similares, por lo que sólo se explicará la correspondiente a la trayectoria circular, el resto se podrá observar con mayor detenimiento en el Anexo A.2. La Figura 4.7 muestra la función “onMakeCircleClicked”, la cual está ligada al botón “Ejecutar” y permite implementar una trayectoria circular a partir de los datos solicitados al usuario.



Figura 4.6: Función “onMakeCircleClicked”.

En la Figura 4.7 se observa de igual manera el botón “Graficar”, el cual está ligado a la función “onPlotCClicked”, la cual hace emerger una nueva ventana donde se plasmará la gráfica de la trayectoria implementada.

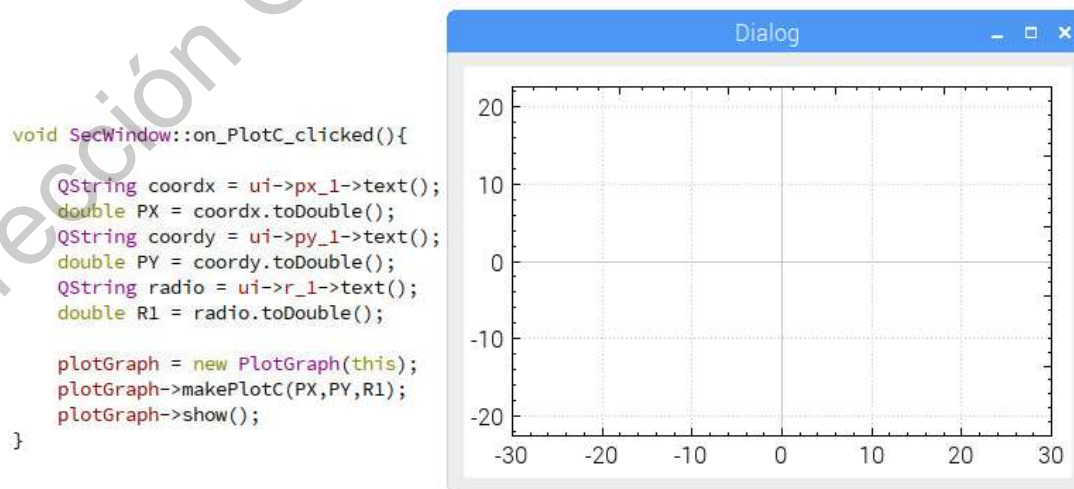


Figura 4.7: Función “onPlotCClicked”.

# Capítulo 5

## Descripción de la interfaz de usuario

En este capítulo se presenta el manejo adecuado de la interfaz de usuario para controlar el robot serial plano R||R.

### 5.1. Arranque de la aplicación y descripción de la ventana principal

El desarrollo de esta aplicación se realizó en la Raspberry Pi 3 Model B+<sup>®</sup> y a continuación se muestra su funcionamiento. En la Figura 5.1 se observa el escritorio de este mini ordenador, donde se encuentra la aplicación para el control de la plataforma.



Figura 5.1: Escritorio de la Raspberry Pi.

Al oprimir dos veces el ícono antes mencionado, se abre la ventana de la Figura 5.2 y se debe oprimir la opción marcada para poder ejecutar la aplicación.

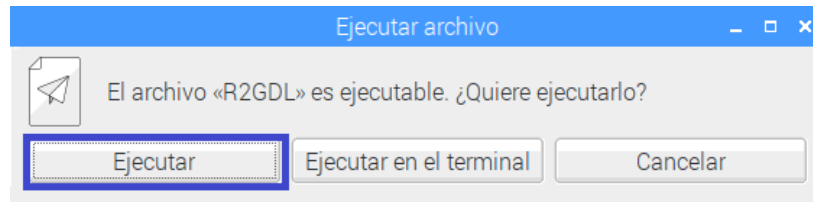


Figura 5.2: Ventana de ejecución de la interfaz de usuario.

Una vez hecho lo anterior, aparece la ventana de la Figura 5.3, la cual muestra la ventana principal de la aplicación, donde se observan las diferentes funciones que puede realizar el robot. En la Figura 5.4, se indican 3 botones, que son accesibles desde todas las pestañas. Estos botones son: Inicio, Apagar y Orientación de codo.

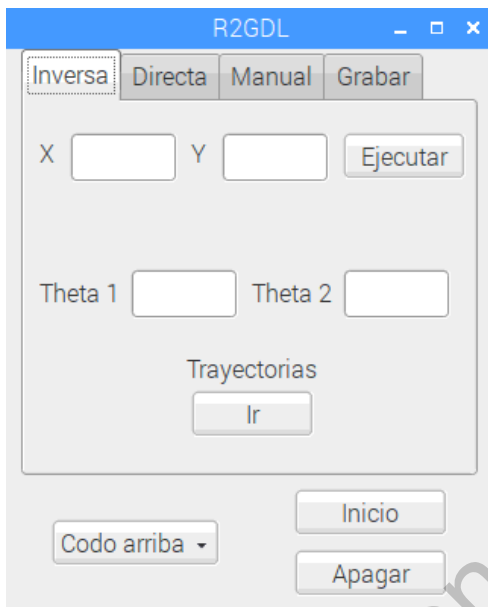
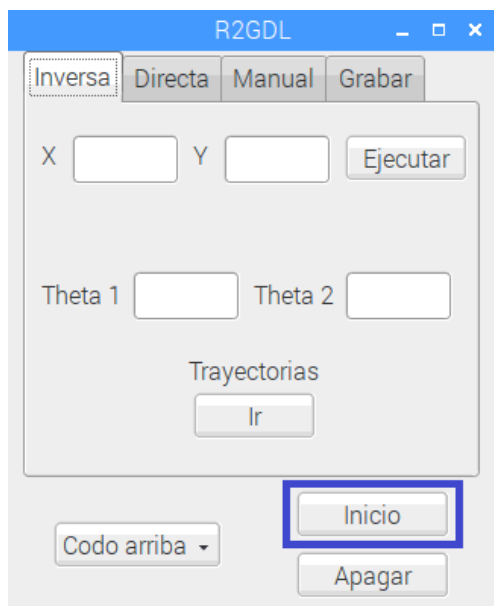


Figura 5.3: Ventana principal.

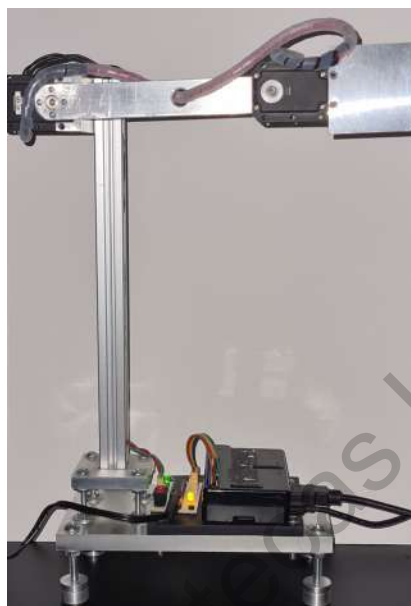


Figura 5.4: Ventana principal.

Al presionar el botón “Inicio”, el robot se posicionará de tal manera que los eslabones estén paralelos respecto al suelo, como se observa en la Figura 5.5. Representando esta posición en un plano, los motores se encontrarían en el punto inicial de 0°.



(a) Botón “Inicio”.



(b) Posición “Inicio”.

Figura 5.5: Posición inicial de robot.

Al presionar el botón “Apagar”, el robot se posicionará de tal manera que los eslabones estén perpendiculares respecto al suelo, como se observa en la Figura 5.6. Esto evita que al desconectar o apagar la plataforma, se enreden los cables.



(a) Botón “Apagar”.



(b) Posición “Apagar”.

Figura 5.6: Posición de apagado de robot.

En la Figura 5.7, se indica el selector “Orientación de codo”, que permite al usuario definir el sentido en el que se posicionará el codo, teniendo como opciones “Codo arriba” y “Codo abajo”. Esto representa las dos soluciones existentes para cada análisis.

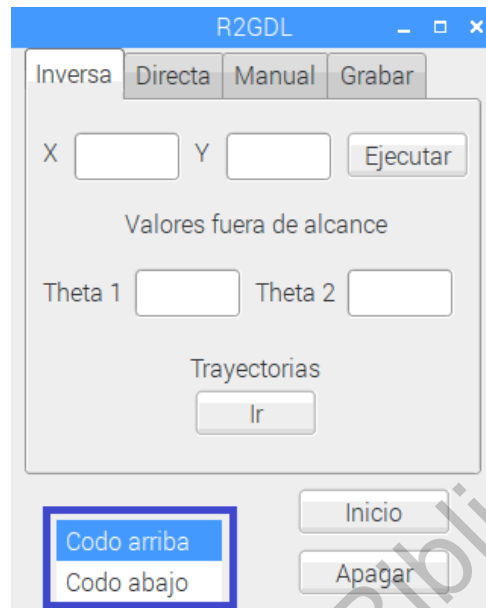


Figura 5.7: Ventana principal.

En la Figura 5.8, se observa la comparación entre los valores de  $\theta_1$  y  $\theta_2$  obtenidos en ambas respuestas para la coordenada [10,-15].



(a) Codo arriba.

(b) Codo abajo.

Figura 5.8: Comparación orientación de codo en coordenadas [10,-15].

En la Figura 5.9, se observa el robot ejecutando las dos respuestas para la coordenada [10,-15].





(a) Codo arriba.



(b) Codo abajo.

Figura 5.9: Comparación orientación de codo en coordenadas [10,-15].

## 5.2. Descripción de la pestaña “Inversa”

Inicialmente se muestra la pestaña “Inversa”, la cual se encarga de ejecutar el método cinemático inverso de posición en el robot. Para este método se necesitan los valores de posición deseados, en este caso las coordenadas  $X$ - $Y$ , las cuales son proporcionadas por el usuario y se colocan en los recuadros mostrados en la Figura 5.10. En la Figura 5.11 se observa el botón “Ejecutar”, el cual permite realizar el cálculo según las coordenadas deseadas, para después implementar los resultados obtenidos en la plataforma.



Figura 5.10: Captura de coordenadas.

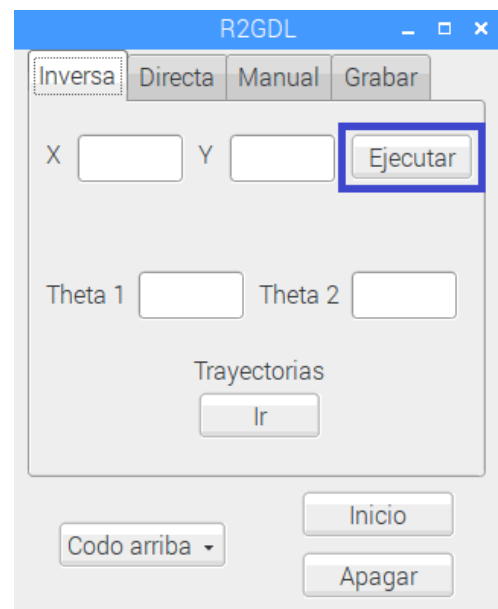


Figura 5.11: Ejecución del análisis.

Una vez realizado el análisis y su implementación en el robot, los valores finales de las variables articulares,  $\theta_1$  y  $\theta_2$ , se muestran en la interfaz donde remarca la Figura 5.12. La Figura 5.13 muestra la opción de planificación de trayectorias, la cual abrirá una ventana emergente que se explicará más adelante.

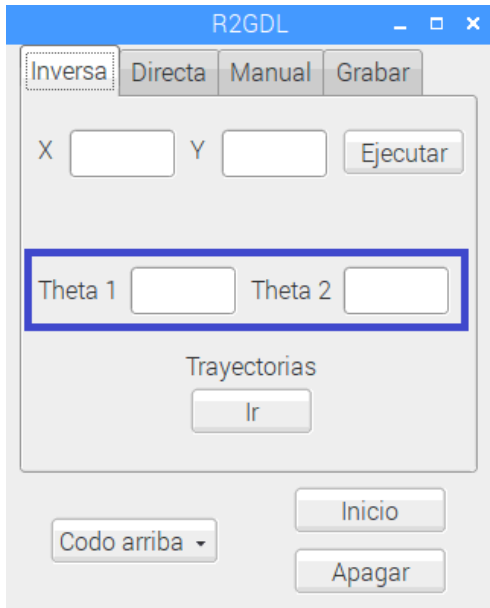


Figura 5.12: Valores resultantes de las variables articulares.



Figura 5.13: Opción “Trayectorias” del método inverso.

En la Figura 5.14 se observa un ejemplo, donde se ingresaron los valores de [10,-10]. Al ejecutar el programa la interfaz arroja un mensaje que declara que el cálculo se realizó correctamente, así mismo, los valores finales de las variables articulares se imprimen en la interfaz. Por otro lado, en la Figura 5.15 se muestra un caso erróneo, donde las coordenadas fueron [20,20], lo cual arroja un mensaje mencionando que los valores ingresados se encuentran fuera de alcance y, por ende, el análisis no se pudo realizar adecuadamente. En este caso será necesario ingresar nuevos valores y verificar que sean correctos.

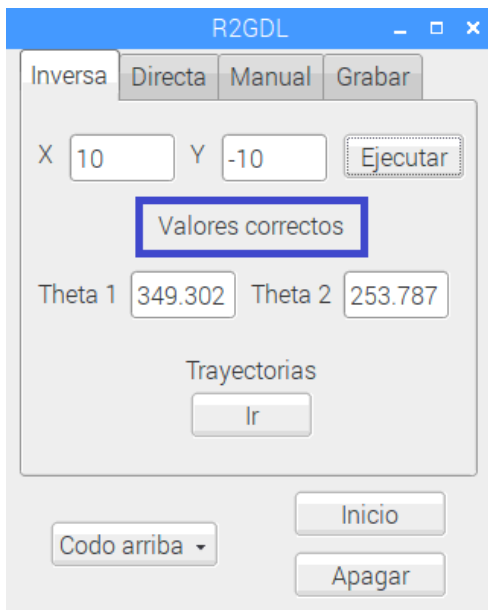


Figura 5.14: Ejemplo correcto.

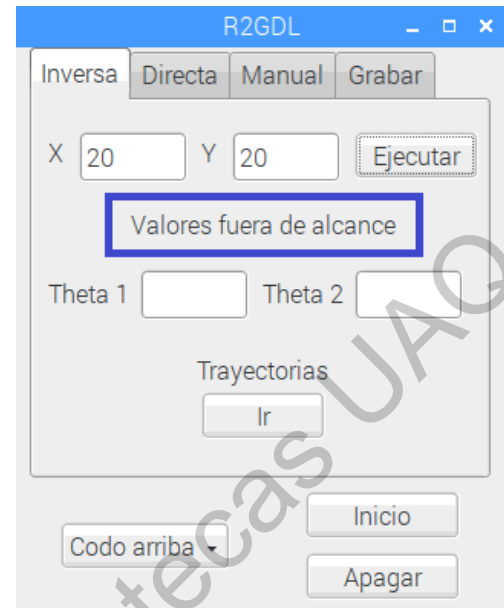


Figura 5.15: Ejemplo incorrecto.

### 5.2.1. Descripción de la ventana “Trayectorias”

La planificación de trayectorias busca llevar el actuador final del robot desde un punto a otro a través de una trayectoria definida. En la Figura 5.13 se indica el botón “Trayectorias”, el cual abre una ventana adicional con la que se puede implementar en la plataforma la planificación de trayectorias antes mencionada. Dicha ventana se observa en la Figura 5.16 y se puede ver que tiene varias pestañas con distintas funciones que a continuación se explicarán.

Primeramente, se encuentra la pestaña “Círculo”, la cual realiza una trayectoria circular a partir de las coordenadas del centro y el radio del círculo a trazar. Estos datos se introducen por el usuario en los recuadros mostrados en la Figura 5.17.

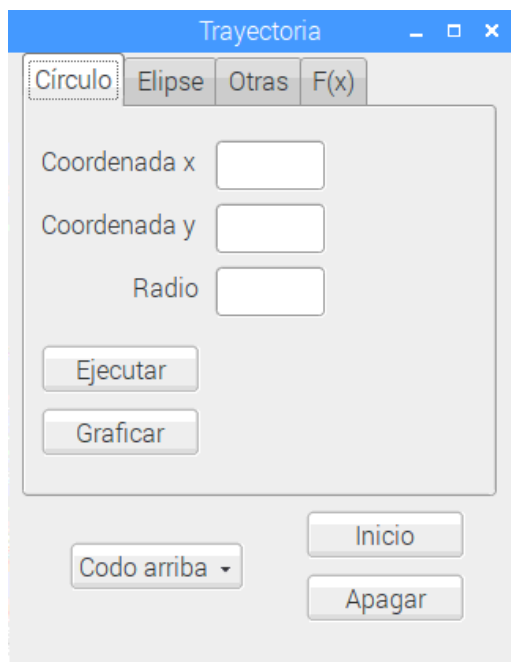


Figura 5.16: Ventana “Trayectorias”.



Figura 5.17: Captura de datos.

En la Figura 5.18 se observa el botón “Ejecutar”, el cual permite que se procesen los datos introducidos y se realice el cálculo de la trayectoria. Si los datos son correctos la plataforma comenzará la implementación de dicha trayectoria, a su vez se desplegará la información capturada en forma de las ecuaciones paramétricas del círculo a trazar, como se muestra en la Figura 5.19. Si los datos son incorrectos se desplegará la leyenda “Valores fuera de alcance”, en este caso el usuario deberá introducir nuevos valores y verificar que sean correctos.

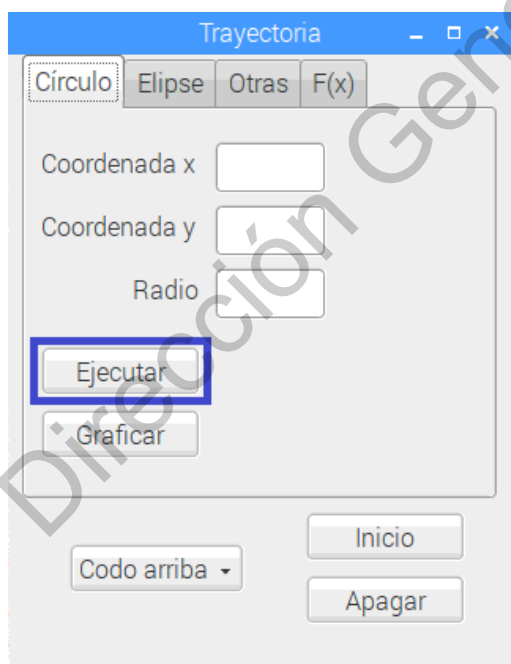


Figura 5.18: Ejecución de la trayectoria.

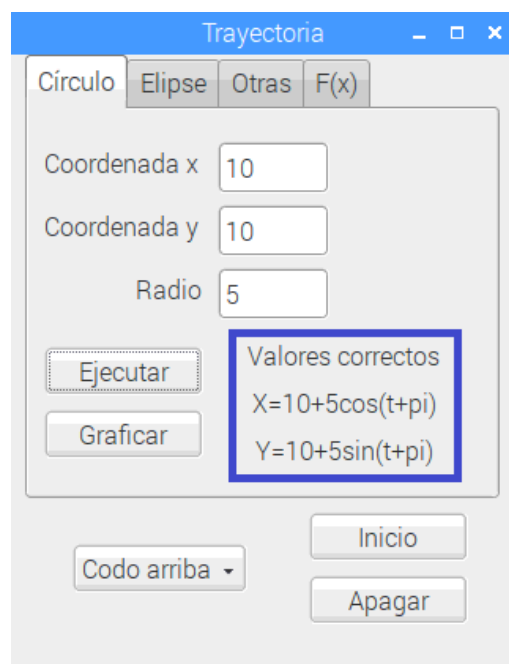


Figura 5.19: Ejemplo correcto.

Una vez realizado el cálculo y su implementación adecuados, la aplicación puede graficar la trayectoria que se obtuvo a partir de los datos introducidos por el usuario. En la Figura 5.20 se observa el botón “Graficar”, el cual despliega la ventana de la Figura 5.21, donde se observa la gráfica de la trayectoria circular.

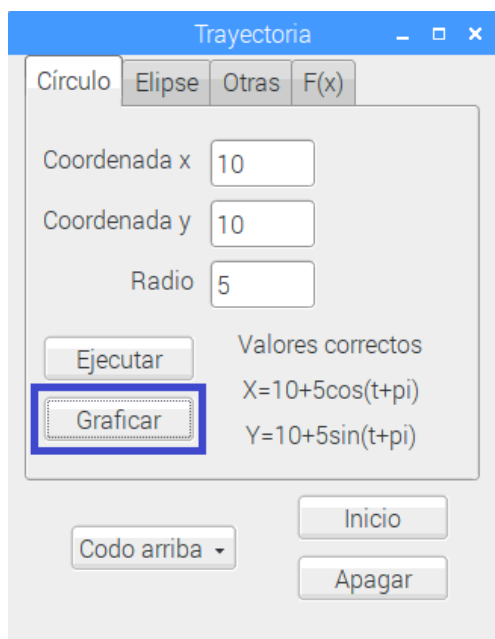


Figura 5.20: Botón “Graficar”.

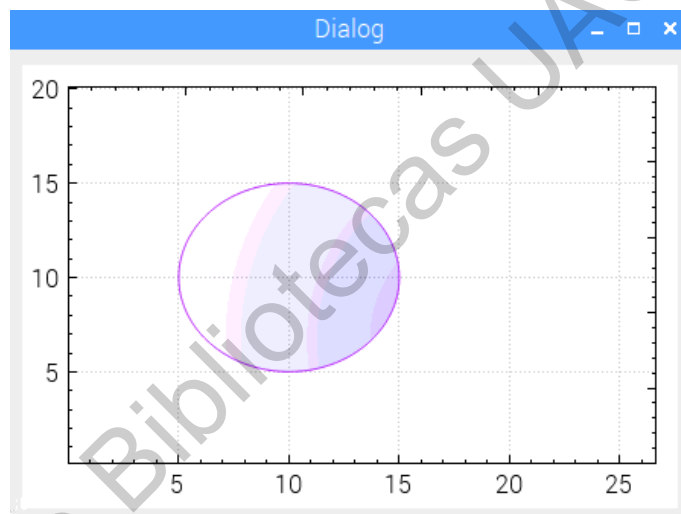


Figura 5.21: Gráfica de la trayectoria.

Después se encuentra la pestaña “Elipse”, Figura 5.22, la cual realiza una trayectoria elíptica a partir de las coordenadas del centro y los semiejes, los cuales deben ser introducidos por el usuario en los recuadros mostrados en la Figura 5.23.



Figura 5.22: Pestaña “Elipse”.

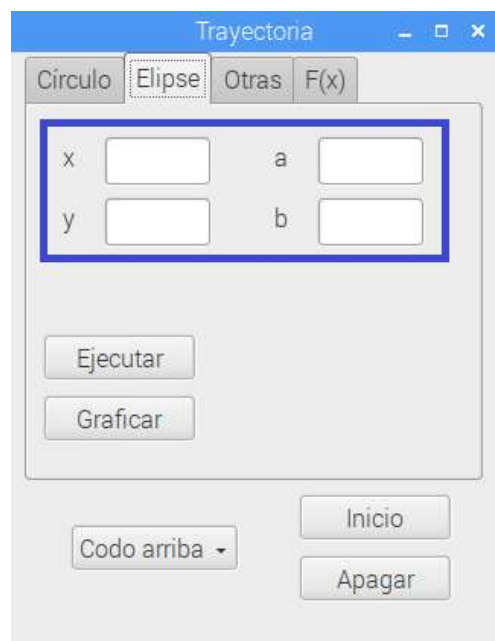


Figura 5.23: Captura de datos.

En la Figura 5.24 se observa el botón “Ejecutar”, el cual permite que se procesen los datos introducidos y se realice el cálculo de la trayectoria. Si los datos son correctos la plataforma comenzará la implementación de dicha trayectoria, a su vez se desplegará la información capturada en forma de las ecuaciones paramétricas de la elipse a trazar, como se muestra en la Figura 5.25. Si los datos son incorrectos se desplegará la leyenda “Valores fuera de alcance”, en este caso el usuario deberá introducir nuevos valores y verificar que sean correctos.



Figura 5.24: Ejecución de la trayectoria.



Figura 5.25: Ejemplo correcto.

Una vez realizado el cálculo y su implementación adecuados, la aplicación puede graficar la trayectoria que se obtuvo a partir de los datos introducidos por el usuario. En la Figura 5.26 se observa el botón “Graficar”, el cual despliega la ventana de la Figura 5.27, donde se observa la gráfica de la trayectoria elíptica.

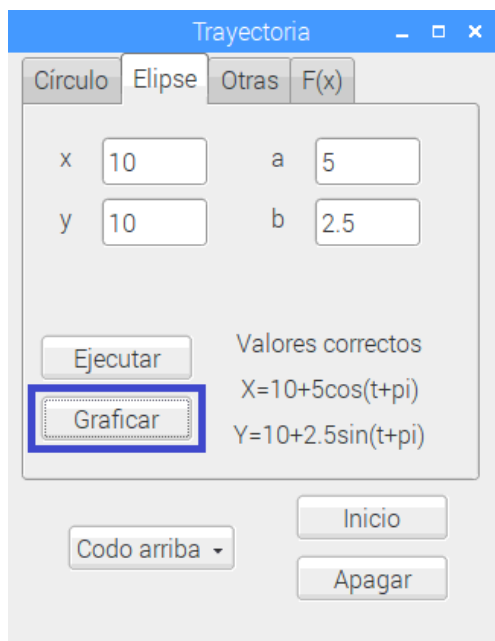


Figura 5.26: Botón “Graficar”.

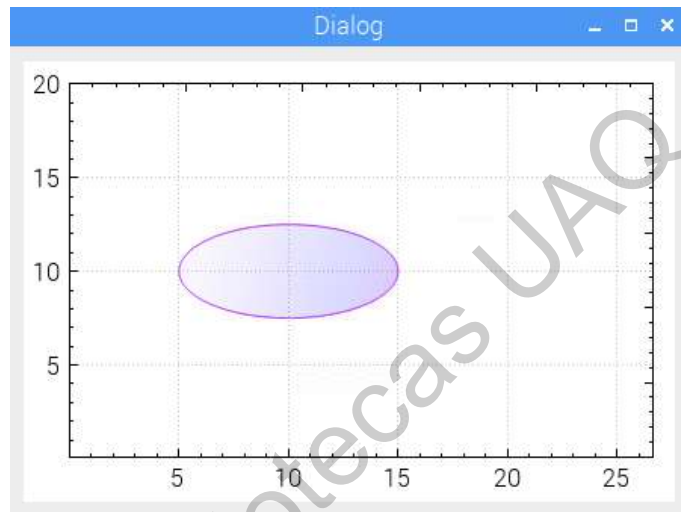


Figura 5.27: Gráfica de la trayectoria.

Después se encuentra la pestaña “Otras”, Figura 5.28, la cual permite realizar otras curvas a partir de las ecuaciones paramétricas. En este caso las ecuaciones paramétricas son de la forma

$$x = (a - b)\cos(t) + b\cos(t(k - 1)) \tag{5.1}$$

$$y = (a - b)\sin(t) - b\sin(t(k - 1)) \tag{5.2}$$

siendo  $k = \frac{a}{b}$ , por lo tanto, los valores de  $a$  y  $b$  deben elegirse con cuidado para obtener un resultado correcto. Estos valores deben ser introducidos por el usuario en los recuadros marcados de la Figura 5.29.

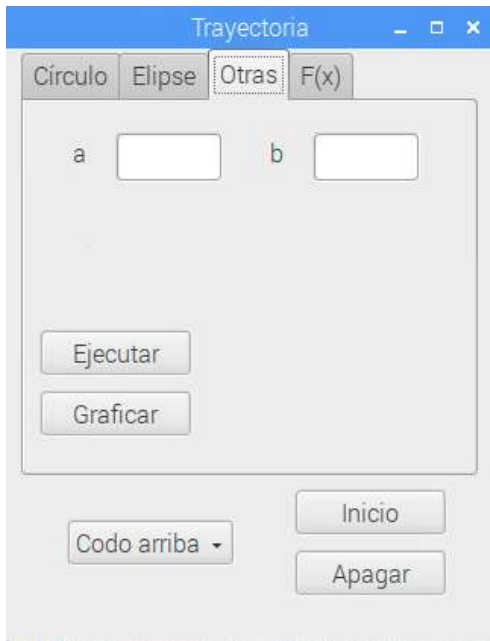


Figura 5.28: Pestaña “Otras”.

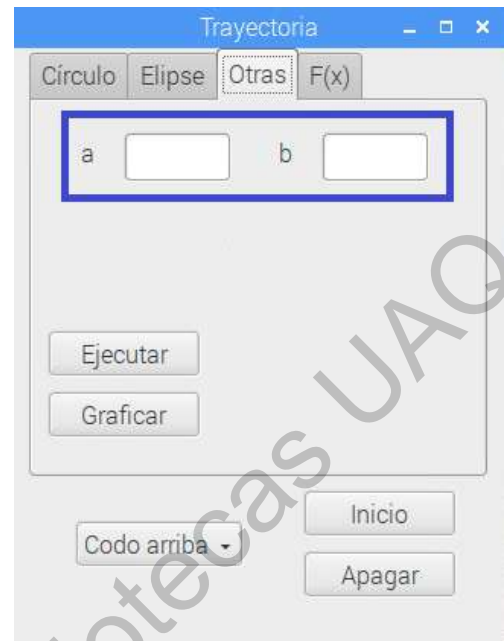


Figura 5.29: Captura de datos.

Como ejemplo se tomaron los valores de  $a = 18$  y  $b = 6$ , lo que resulta en  $k = 3$ , ya que estos valores permiten calcular una trayectoria que se encuentra dentro del alcance de la plataforma. En la Figura 5.30 se observa el botón “Ejecutar”, el cual permite que se procesen los datos introducidos y se realice el cálculo de la trayectoria. Si los datos son correctos la plataforma comenzará la implementación de dicha trayectoria, a su vez se desplegará la información capturada en forma de las ecuaciones paramétricas de la función a trazar, como se muestra en la Figura 5.31. Si los datos son incorrectos se desplegará la leyenda “Valores fuera de alcance”, en este caso el usuario deberá introducir nuevos valores y verificar que sean correctos.

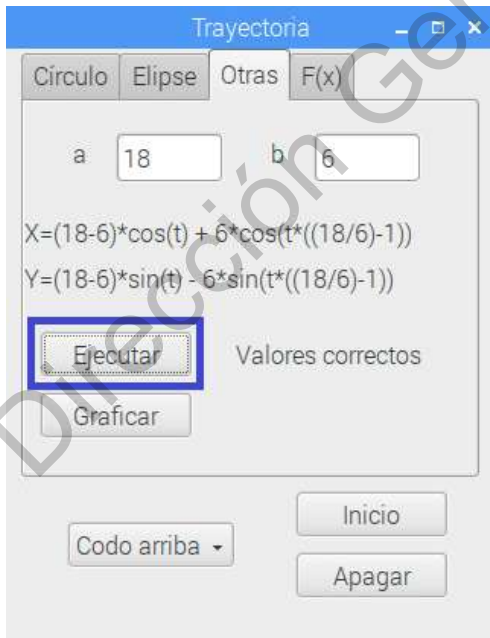


Figura 5.30: Ejecución de la trayectoria.

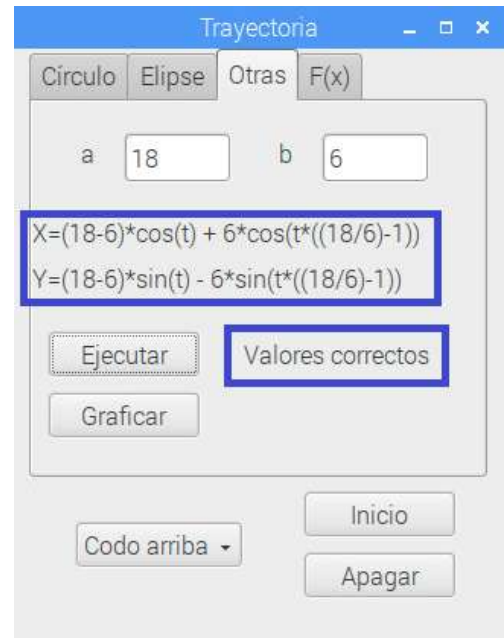


Figura 5.31: Ejemplo correcto.



Una vez realizado el cálculo y su implementación adecuados, la aplicación puede graficar la trayectoria que se obtuvo a partir de los datos introducidos por el usuario. En la Figura 5.32 se observa el botón “Graficar”, el cual despliega la ventana de la Figura 5.33, donde se observa la gráfica de la trayectoria implementada.

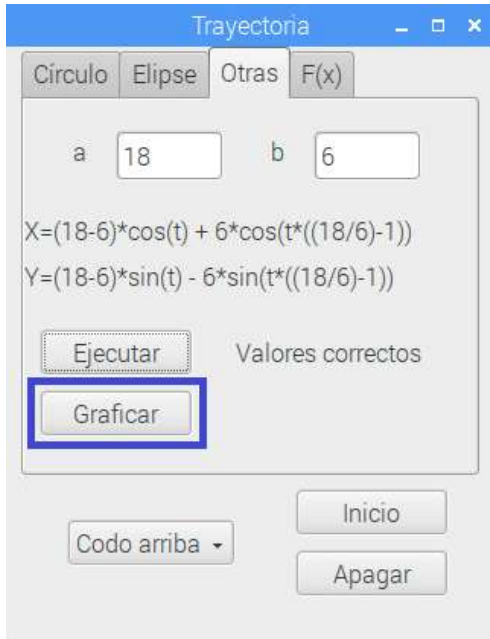


Figura 5.32: Botón “Graficar”.

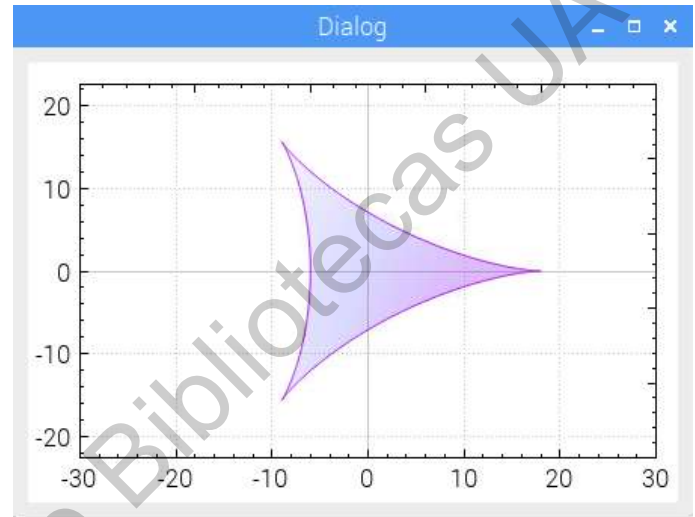


Figura 5.33: Gráfica de la trayectoria.

Se colocó un LED en el borde del segundo eslabón y se tomó una fotografía de larga exposición, que se muestra en la Figura 5.34, para visualizar parte de la trayectoria que ejecuta el robot.

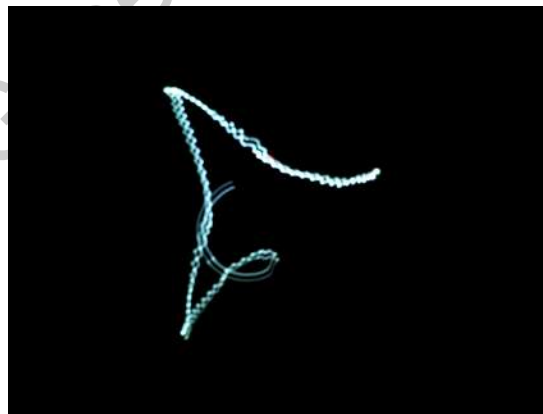


Figura 5.34: Trayectoria robot para k=3.

A continuación, se muestran ejemplos donde cambia la trayectoria según se varía el valor  $k$ . Como ya se observó en el ejemplo anterior, la Figura 5.33 representa la trayectoria para  $k = 3$ , la Figura 5.35 para  $k = 4$ , la Figura 5.36 para  $k = 5$  y la Figura 5.37 para  $k = 6$ .

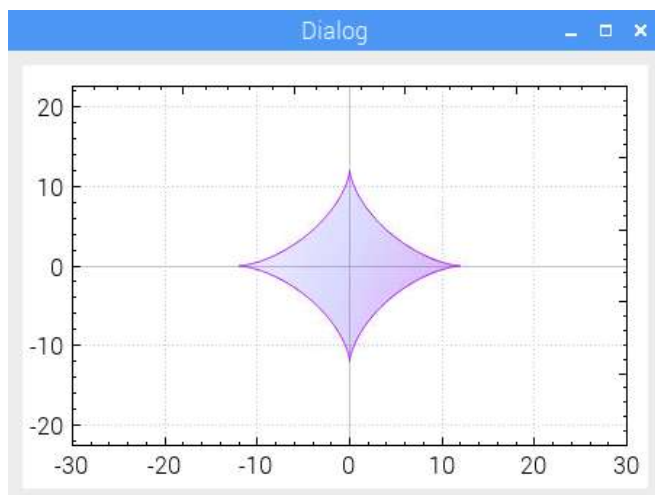


Figura 5.35: Gráfica de la trayectoria con valor  $k = 4$ .

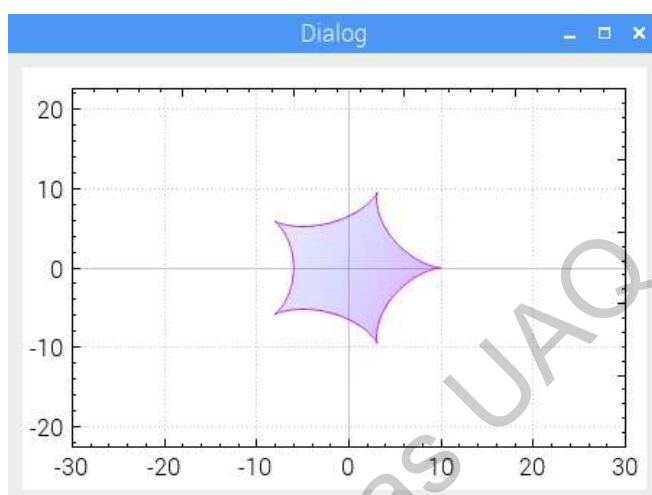


Figura 5.36: Gráfica de la trayectoria con valor  $k = 5$ .

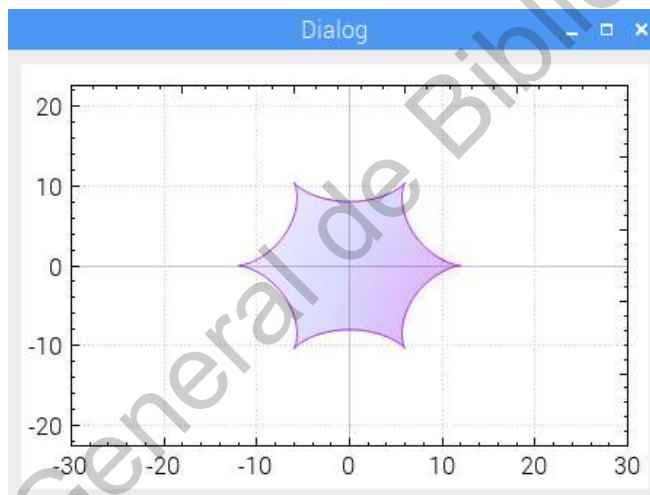


Figura 5.37: Gráfica de la trayectoria con valor  $k = 6$ .

### 5.3. Descripción de la pestaña “Directa”

Seleccionando la pestaña “Directa” en la interfaz, se desplegará la función del método cinemático directo de posición. En este caso la interfaz solicita los valores de las variables articulares,  $\theta_1$  y  $\theta_2$ , los cuales debe ingresar el usuario como se indica en la Figura 5.38. En la Figura 5.39 se observa el botón “Ejecutar”, el cual implementa en la plataforma los valores ingresados anteriormente.

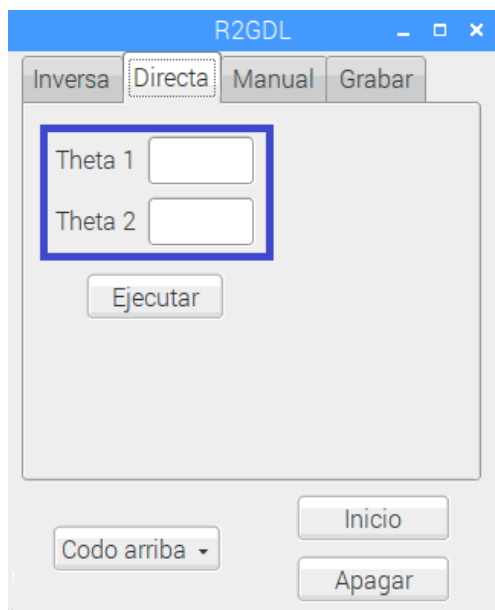


Figura 5.38: Captura de datos.



Figura 5.39: Ejecución del análisis.

En la Figura 5.40 se muestra la captura de los datos y su implementación en el robot.



(a) Valores ingresados.



(b) Posición  $\theta_1 = 225^\circ$  y  $\theta_2 = 90^\circ$ .

Figura 5.40: Respuesta del robot.

### 5.4. Descripción de la pestaña “Manual”

La pestaña “Manual” presenta una función similar a la pestaña “Directa”, con la diferencia que se incrementan o se decrementan los valores de las variables articulares,  $\theta_1$  y  $\theta_2$ , con las flechas indicadas en la Figura 5.41.



Figura 5.41: Ventana principal.

### 5.5. Descripción de la pestaña “Grabar”

La pestaña “Grabar” de la aplicación permite grabar diferentes puntos en el plano para el actuador final del robot, los cuales se insertan como se indica en la Figura 5.42. Una vez colocado el primer par de coordenadas, se debe presionar el botón “Grabar” como se observa en la Figura 5.43, con esto los valores del primer punto quedarán almacenados por el programa. Es importante mencionar que esta función toma en cuenta la selección en la orientación del codo, la cual se explica en la Sección 5.1, por lo que el usuario debe considerar esto para la implementación que desea realizar.



Figura 5.42: Captura de datos.

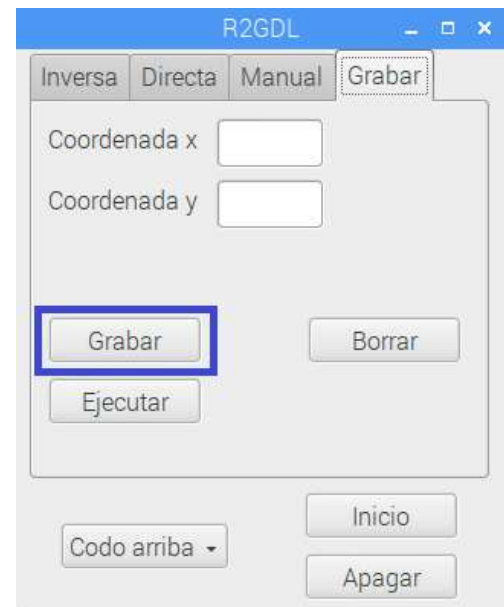


Figura 5.43: Ejecución del análisis.

Ya que se hayan capturado todos los puntos que se deseen, se presiona el botón “Ejecutar” como indica la Figura 5.44, con lo que la plataforma comenzará su movimiento hacia el primer punto almacenado, después al segundo y así continuará hasta llegar al último punto. A este procedimiento también se le conoce como programación punto a punto dentro del área de la robótica. en la Figura 5.45 se observa el botón “Borrar”, el cual permite borrar todos los puntos que se hayan capturado anteriormente, en caso de que haya habido alguna equivocación y sea requerido comenzar de nuevo.



Figura 5.44: Captura de datos.



Figura 5.45: Ejecución del análisis.

Dirección General de Bibliotecas UNO

# Capítulo 6

## Conclusiones

Se realizó un diseño mecánico preliminar que nos permitió evaluar las mejores opciones para la creación de un diseño final adecuado y eficiente, considerando el hecho de que el robot será utilizado para fines didácticos. Después se construyeron las piezas mecánicas con ayuda de una fresadora manual, lo que permitió aprender sobre los procesos de maquinado.

Una vez obtenida una plataforma mecánicamente funcional, se realizó el diseño electrónico de una tarjeta PCB para la transmisión de datos entre los motores y la Raspberry Pi 3 Model B+<sup>®</sup>, para su posterior integración al robot.

Finalmente, se desarrolló un algoritmo de control, mediante la Raspberry Pi 3 Model B+<sup>®</sup>, a partir de la solución cerrada del análisis inverso de posición, lo que nos permitió implementar control de posición, así como programación punto a punto y planificación de trayectorias. A su vez, se desarrolló una interfaz gráfica muy intuitiva, de manera que al usuario le fuera sencillo realizar las diferentes funciones que el robot puede realizar.

El diseño de esta plataforma tiene flexibilidad suficiente para añadir más grados de libertad y permitir un constante crecimiento en su alcance, de manera que se pueda seguir trabajando con ella en futuras aplicaciones.

Un ejemplo de esto se presentó al utilizarse el robot como herramienta de soporte para la implementación de perfiles de velocidad, un proyecto que fue desarrollado por alumnos de maestría de la Facultad con la finalidad de publicar un artículo sobre este tema. Ya sea Licenciatura, Maestría o Doctorado, se espera que el robot siga siendo de ayuda para los profesores y estudiantes de la Facultad.

Este proyecto resultó muy enriquecedor tanto a nivel académico como a nivel personal porque nos dio la oportunidad de desarrollarlo casi en su totalidad, partiendo de las bases esenciales de mecánica, programación y electrónica, para concluir con un robot completamente funcional que engloba nuestras ideas y aspiraciones, lo que nos llena de orgullo ya que estamos muy satisfechos con los resultados obtenidos.

Se obtuvo como resultado final una plataforma funcional que cumple con las tareas que se fijaron en los objetivos, lo que empata con nuestra hipótesis y nos permite concluirla como acertada, ya que haciendo uso de una Raspberry Pi 3 Model B+<sup>®</sup> para el desarrollo de un controlador cinemático y de una interfaz de usuario, fue posible implementar control de posición y planificación de trayectorias en el robot desarrollado.

# Apéndice A

## Anexos

### A.1. Planos del diseño final

En el presente anexo se muestra el código QR que redirige a la carpeta que contiene los planos de construcción del robot plano R||R.



Figura A.1: Código QR planos robot plano R||R.

## A.2. Códigos generados en el desarrollo del robot serial plano R||R

En el presente anexo se muestra el código QR que dirige a la carpeta que contiene todos y cada uno de los códigos desarrollados para la implementación del modelo cinemático y la creación de la interfaz gráfica.



Figura A.2: Código QR códigos robot plano R||R.



### A.3. Solución cerrada de velocidad usando Matlab

En el presente anexo se muestra el código fuente para la obtención de la solución cerrada de velocidad con ayuda de Matlab.

```
%Solución cerrada de velocidad

%Elementos de la matriz Jacobiana
syms J11 J21;
syms J12 J22;

%Variables de distancia
syms l1 l2;

%%%%%%%%%%%%%%
%Descomentar esta sección si se requieren los valores de los eslabones
%Unidades en metros
%l1=14;
%l2=8.3;
%%%%%%%%%%%%%%

%Variables angulares
syms Th1 Th2

%%%%%%%%%%%%%%
%Descomentar esta sección para una solución numérica
%En radianes
%Th1= pi/4;
%Th2= pi/2;
%%%%%%%%%%%%%%

%Jacobiano
J11= -l1*sin(Th1)-l2*cos(Th1)*sin(Th2)-l2*sin(Th1)*cos(Th2);
J21= l1*cos(Th1)+l2*cos(Th1)*cos(Th2)-l2*sin(Th1)*sin(Th2);

J12= -l2*cos(Th1)*sin(Th2)-l2*sin(Th1)*cos(Th2);
J22= l2*cos(Th1)*cos(Th2)-l2*sin(Th1)*sin(Th2);

J= [J11 J12;
J21 J22];

%Jacobiana inversa
Jinv= inv(J);
Jinv= simplify(Jinv);

%Pe variables de velocidad lineal de las coordenadas XY
```

```

%Ve= [xdot, ydot]
syms xdot ydot;

%%%%%%%%%%
%Aquí se pueden ingresar los datos de un problema:
%xdot=0;
%ydot=0;
%%%%%%%%%%

Ve= [xdot ydot]';

%El vector de velocidades individuales se obtiene de
%qdot=J(q)-1 * Ve

syms resultado;
resultado= Jinv*Ve;

%Valores de velocidad angular individual
qdot1= resultado(1);
qdot2= resultado(2);

qdot= [qdot1 qdot2]';

%%%%%%%%%%

```

Dirección General de Bibliotecas UAQ

# Bibliografía

- [1] Cortes, F. R. (2011). *Robotica: Control de robots manipuladores*. Alfaomega Grupo Editorial.
- [2] S. K. Saha. 2010. *Introducción a la robótica*, McGraw-Hill Educación: México.
- [3] Barrientos, A. (1997). *Fundamentos de robótica*. McGraw-Hill, Interamericana de España.
- [4] Norton, R. L. (2008). *Diseño de maquinaria. Síntesis y análisis de máquinas y mecanismos*. McGraw-Hill Higher Education.
- [5] Martínez, P. (2015). *Manipulador Paralelo Plano (3-RRR) con actuación virtual indirecta*. Tesis de Doctorado. Universidad Nacional Autónoma de México, México, D.F.
- [6] Kelly, R., Santibañez, V., Loría, A. (2005). *Control of Robot Manipulators in Joint Space*. Springer.
- [7] French, T. E., Vierck, C. J., Alba, R. F. A. (1981). *Dibujo de ingeniería*. McGraw-Hill Education.
- [8] ROBOTIS, INC. (2020). *DYNAMIXEL MX-28T*. ROBOTIS. Recuperado de <https://www.robotis.us/dynamixel-mx-28t/>
- [9] Raspberry Pi. (2017). *Raspberry Pi 3 Model B+*. Recuperado de <https://static.raspberrypi.org/files/product-briefs/200206+Raspberry+Pi+3+Model+B+plus+Product+Brief+PRINTDIGITAL.pdf>