Universidad Miguel Hernández de Elche

MÁSTER UNIVERSITARIO EN ROBÓTICA

“Construcción y programación de un prototipo de robot trepador bípedo”

Trabajo de Fin de Máster

2021/2022

Autor: Marc Fabregat Jaén
Tutor: Adrián Peidró Vidal

Índice

1. Introducción

1.1. Antecedentes

El presente Trabajo de Fin de Máster (en adelante, TFM) se encuentra en el ámbito de los robots trepadores. Se trata de un campo de trabajo en constante y extenso estudio durante las últimas tres décadas. El objetivo principal de los robots escaladores es el de realizar tareas en lugares que supongan un peligro para un operario humano, ya sea por un difícil acceso o por el riesgo de caída desde una altura considerable. Es por ello que los escenarios donde los robots trepadores son útiles son vastos y diversos. Una de las tareas más apropiadas para este tipo de robots es la inspección y mantenimiento de estructuras verticales, tales como puentes, torres de distribución eléctrica o esqueletos de edificios.

Existe una gran variedad de robots trepadores propuestos por diferentes investigadores para realizar tareas en altura, por lo que es conveniente realizar una clasificación de los mismos. En (Schmidt and Berns, 2013) se presenta un análisis de los criterios de diseño y aplicaciones de este tipo de robots, así como una categorización de los mismos en función del tipo de adhesión y del método de locomoción que utilizan para moverse.

Atendiendo al tipo de adhesión, las principales tecnologías de las que se valen los robots trepadores para fijarse a las superficies por las que se trasladan son la sujeción mecánica, la adhesión magnética y la fijación neumática.

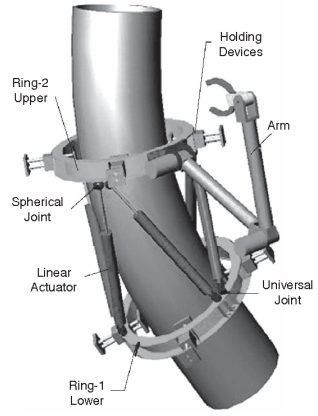


Figura 1: Ejemplo de robot paralelo con adhesión mecánica: TREPA (UMH)

La fijación mecánica se basa en la fijación por medio de extremos punzantes clavados en la superficie de adhesión o mediante pinzas para asir la estructura por dos caras opuestas. Ejemplos de uso de tecnologías de adhesión mecánica donde se realiza una prensión por lados opuestos de la estructura escalada se exponen en (Tavakoli et al., 2011), (Balaguer et al., 2000), (Yoon and Rus, 2007) y (Aracil et al., 2006) (figura 1). Las principales ventajas de este tipo de sujeción son el reducido consumo de energía y su robustez en superficies irregulares. No obstante, suelen ser métodos más lentos, necesitan acceder a caras opuestas de la estructura

y generalmente soportan menos carga que otras tecnologías.

Por otro lado, la adhesión por medios magnéticos es ideal para superficies metálicas. Consiste en la utilización de imanes o electroimanes para pegarse a la superficie (Shvalb et al., 2013). Esta tecnología de fijación ofrece una relación de fuerza de adhesión obtenida por área ocupada muy elevada. Sin embargo, debido a la naturaleza de la tecnología, las superficies para las que es válida están limitadas por las propiedades ferromagnéticas de las mismas. Ejemplos de superficies donde se utiliza son torres de distribución eléctrica, cascos de barcos o tanques de almacenamiento. En (Peidró et al., 2019) se presenta un diseño de garras para el robot HyReCRo (figura 2) que utilizan imanes permanentes conmutables.

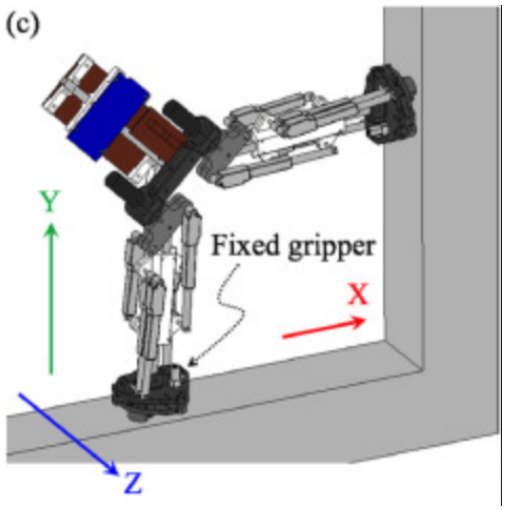


Figura 2: Ejemplo de robot con adhesión magnética: HyReCRo (UMH)

Otra tecnología utilizada es la adhesión neumática. Se sirve de ventosas pasivas o cámaras de presión negativa para la adhesión. La mayor ventaja que ofrece es la versatilidad en cuanto a superficies en las que funciona. No obstante, precisa de un área grande de pegado, por lo que no es apropiada para estruturas de superficie limitada. Un ejemplo de uso de esta tecnología se presenta en (Hernando et al., 2019) (figura 3).

En cuanto al método de locomoción, en (Tavakoli et al., 2011) se hace una diferenciación para robots trepadores de estructuras.

Por un lado, se encuentran los robots de movimiento continuo, los cuales utilizan ruedas para desplazarse. Suelen utilizar magnetismo o fricción para adherirse a las superficies. Sus principales ventajas son su simplicidad en cuanto al control y la rapidez que ofrecen (pueden llegar a la pose deseada antes). No obstante, presentan más problemas para evadir obstáculos y existe peligro de deslizamiento de las ruedas. Ejemplos de uso de este método de locomoción se pueden encontrar en (Tavakoli et al., 2013) y (Tâche et al., 2009), donde se emplean ruedas magnéticas, y en (Baghani et al., 2005), donde se usan 3 pares de ruedas que logran sujetarse a la superficie mediante fricción, tal como se muestra en la figura 4.

Por otro lado, los robots trepadores de movimiento paso a paso están compuestos

Figura 3: Ejemplo de robot con adhesión neumática: ROMERIN (UPM)

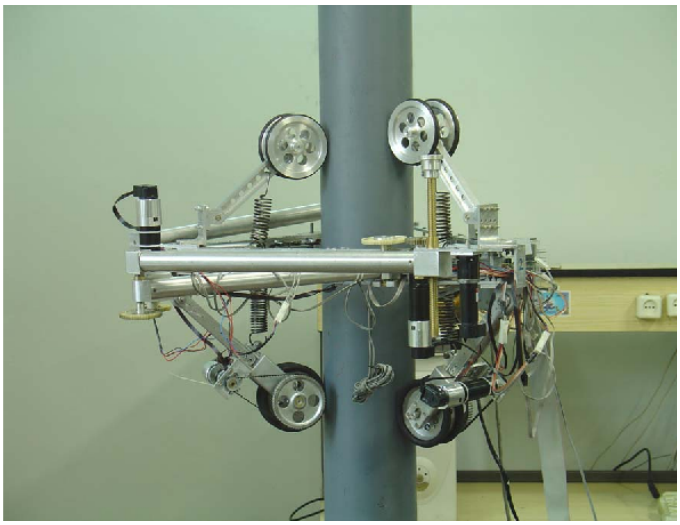


Figura 4: Ejemplo de robot trepador de movimiento continuo: UT-PCR (UT)

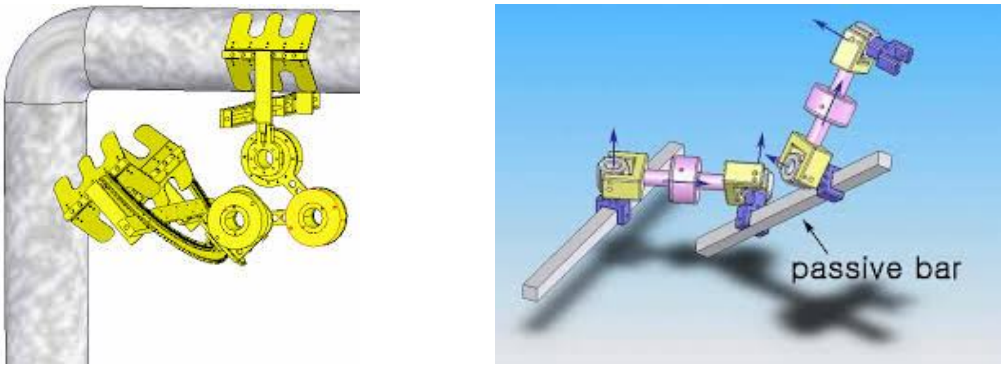
de 2 garras, las cuales utilizan para adherirse a la superficie, unidas a través de una cadena cinemática. Para desplazarse, el robot fija una de sus garras, que cumple el rol de base, para mover la otra garra, que actúa como efector final, hasta la nueva pose de pegado deseada. Tras ello, el efector final se adhiere a la superficie y se intercambian los roles de base y efector final para empezar otro ciclo. Durante cada paso, el robot se comporta como un robot manipulador, por lo que tienen una mayor movilidad y maniobrabilidad que los de movimiento continuo a costa de mayor complejidad en el control, mayores incercias y velocidad menor.

También existen métodos de locomoción híbridos como el presentado en (Viegas and Tavakoli, 2014). Combina el movimiento continuo, haciendo uso de ruedas para el desplazamiento planar, y el movimiento paso a paso al incorporar un brazo que se utiliza para llevar a cabo cambios de planos en las estructuras.

Dentro de los robots escaladores de estructuras reticulares tridimensionales con movimiento paso a paso, se puede hacer otra clasificación en función de la arqui-

tectura de la cadena cinemática que une sus garras.

En primer lugar, se encuentran las cadenas cinemáticas abiertas o de arquitectura serie. Generalmente, ofrecen un espacio de trabajo mayor y una mayor maniobrabilidad que las arquitecturas paralelas a costa de una rigidez menor y cuentan con una capacidad de carga limitada. Es el tipo de arquitectura más investigada y existen multitud de robots propuestos en la literatura. En (Balaguer et al., 2000) y (Tavakoli et al., 2011) (figura 5a) se proponen ejemplos de robots trepadores tipo serie con 6 y 4 grados de libertad (en adelante, GDL) respectivamente. Muchos robots propuestos están inspirados en el movimiento de las orugas al desplazarse. Por ejemplo, en (Guan et al., 2011) se propone un robot de 5 GDL y en (Shvalb et al., 2013) un robot redundante de 8 GDL. Siguiendo esta línea, (Mampel et al., 2009) presenta un robot modular cuya configuración puede ser modificada conectando módulos en serie para conseguir un determinado número de GDL. Por último, en (Yoon and Rus, 2007) se muestran robots de 3 GDL (figura 5b) los cuales pueden ser utilizados de forma individual o conectarlos en serie para formar cadenas cinemáticas de mayor complejidad.



(a) 3DCLIMBER (UC) (b) Shady3D (MIT)

Figura 5: Ejemplos de robots trepadores paso a paso de arquitectura serie.

Al igual que los manipuladores, también existen robots trepadores de arquitectura paralela, aunque son considerablemente menos comunes. En ellos, sus garras están unidas a través de un mecanismo de cadena cerrada formado por al menos 2 cadenas cinemáticas independientes. Sus principales ventajas son el elevado ratio de carga por peso del robot y su rigidez. Sin embargo, su espacio de trabajo es limitado y no cuentan con tanta maniobrabilidad como los robots tipo serie. Un ejemplo de robot escalador con arquitectura paralela se presenta en (Aracil et al., 2006) (figura 1). El robot propuesto utiliza una plataforma de Stewart para trepar por estructuras cilíndricas, como palmeras o tuberías.

Por último, existen robots de arquitectura híbrida, los cuales se basan en cadenas cinemáticas formadas mecanismos paralelos conectados en serie. La naturaleza híbrida de esta arquitectura permite aunar la alta maniobrabilidad y espacio de trabajo de los robots tipo serie, lo cual es útil para explorar estructuras tridimensionales, y la rigidez y capacidad de carga de la arquitectura paralela. En (Tavakoli et al., 2005) se propone un robot trepador híbrido compuesto por un mecanismo 3-R$\underline{P}$R paralelo unido en serie a un módulo rotacional. Otro ejemplo de arquitectura

híbrida se presenta en (Figliolini et al., 2010) en forma de robot bípedo donde cada una de sus patas está compuesta por la combinación en serie de 2 mecanismos 3-RPS paralelos. Finalmente, en (Peidró et al., 2015a) se muestra el robot HyReCRo (figura 6), sobre el que versará la totalidad de este TFM.

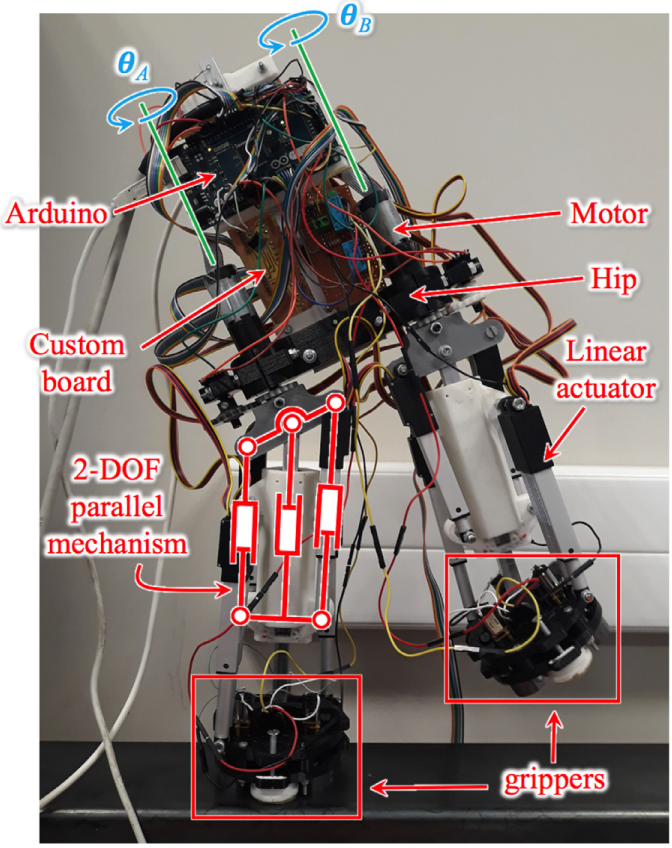


Figura 6: Prototipo del robot HyReCRo presentado en (Peidró et al., 2019).

El robot HyReCRo, desarrollado por el Grupo de Automatización, Robótica y Visión por Computador de la UMH, es un buen ejemplo para ilustrar la gran diversidad existente en cuanto a robots trepadores. Está diseñado para la inspección y mantenimiento de estructuras tridimensionales reticulares metálicas, como puentes, torres de distribución eléctrica o esqueletos de construcciones en forma de entramado. El robot HyReCRo cae dentro de la categorización de robot trepador de estructuras de movimiento paso a paso, ya que está constituido por 2 garras unidas a través de una cadena cinemática. Se trata de un robot bípedo, formado por dos patas unidas a una cadera, y en los pies de dichas patas dispone de sendas garras magnéticas con las que pegarse a la superficie de trabajo.

La cadena cinemática que une ambas patas está formada por 2 patas unidas a la cadera mediante articulaciones de rotación (θ_A y θ_B). A su vez, cada pata está constituida por la unión en serie de 2 mecanismos paralelos (se puede ver uno de ellos destacado en rojo en la figura 6). Por lo tanto, la arquitectura de la cadena cinemática del robot es de tipo híbrida serie-paralela.

Cada mecanismo paralelo (llamado módulo paralelo a lo largo del TFM) consta

de 2 GDL. El robot completo está formado por 4 módulos paralelos que, junto a las 2 articulaciones rotacionales de la cadera, resultan en un total de 10 GDL. Por ende, se trata de un robot cinemáticamente redundante, pues dispone de más de 6 GDL, que es el mínimo número de grados de libertad necesarios para posicionar y orientar una garra de forma totalmente arbitraria respecto a la otra.

La tecnología de adhesión que utiliza para fijarse a la superficie por la que se está desplazando es magnética al estar diseñado para realizar tareas en estructuras metálicas. En (Peidró et al., 2019) se presenta un diseño de las garras utlizando imanes permanentes conmutables. En la sección 2.3 se mostrarán de forma detallada. Además, en el artículo se presenta un prototipo del robot HyReCRo con el que se realizan los experimentos expuestos.

Sin embargo, la estructura del prototipo mostrado no era todo lo rígida que se deseaba y era débil a torsiones y esfuerzos. Además, debido a la electrónica por la que estaba compuesto y su disposición, el peso y centro de gravedad del robot eran elevados, produciéndose así pares elevados cuando el robot se inclinaba. El control del prototipo se realizaba mediante un microcontrolador Arduino, por lo que se hacía difícil la adición de algoritmos más complejos (por ejemplo de inteligencia artificial o visión por computador) al no disponer de un computador al uso.

Por ello, en este TFM se persigue diseñar y construir de forma íntegra un nuevo prototipo del robot HyReCRo, buscando remediar las cuestiones mencionadas, renovando toda la electrónica empleada en el robot. También se añadirá un computador al robot y se programará un nuevo sistema de control del prototipo.

1.2. Objetivos

El objetivo de este TFM es el diseño, la construcción y la programación de un nuevo prototipo del robot trepador de estructuras bípedo HyReCRo. Para acometer el objetivo, se llevarán a cabo los siguiente puntos:

- Nuevo diseño mecánico y estructural del prototipo con el fin de aumentar su rigidez y firmeza.
- Renovación de la eléctronica de la que se compone el robot, tanto de los actuadores como de los componentes que hacen posible el control del mismo.
- Programación de un nuevo sistema de control basado en ROS2.
- Creación de una interfaz gráfica que permita comandar acciones y consultar el estado del robot.

En primer lugar, con el nuevo diseño del prototipo se busca dotarlo de la rigidez y resistencia a torsiones y esfuerzos de las que carecía el prototipo original. Debido al desgaste inevitable de las piezas del prototipo original, existían holguras que

producían un juego no deseado entre los eslabones y las articulaciones del robot. Por otro lado, las articulaciones rotacionales de la cadera del prototipo original eran débiles ante esfuerzos radiales debido a que las articulaciones únicamente estaban formadas por los ejes de los motores que accionaban el movimiento. Además, las patas presentaban poca resistencia a torsiones en el eje vertical de las mismas. Con el nuevo diseño mecánico y estructural del robot se persigue corregir los problemas descritos.

En cuanto al segundo objetivo, mediante la renovación completa de la electrónica de robot se busca simplificar el control del mismo al sustituir los actuadores del prototipo original por nuevos actuadores con control integrado. Además, se pretende instalar un ordenador a bordo que permita la futura implantación de sistemas de control y algoritmos más complejos. También, con la sustitución de los componentes originales y la nueva disposición de los mismos se busca reducir el peso del nuevo prototipo y descender su centro de gravedad, lo cual es beneficioso para las posturas del robot en las que su centro de gravedad se aleje de su punto de apoyo.

El siguiente objetivo mencionado consiste en programar desde cero el sistema de control en su totalidad en ROS2. Utilizando ROS2 como *framework* para la programación del sistema, se consigue que el *software* del robot sea completamente compartible y *open source*. Además, existen multitud de métodos y herramientas ya creadas por la comunidad que facilitan la adición de futuros nuevos sistemas al robot. Compartir los avances conseguidos y poder hacer uso de los ya conseguidos por otros es clave para el buen desarrollo de la comunidad científica.

El último objetivo perseguido en este TFM es el de la programación de una interfaz gráfica de usuario. Su función es emular a una *teach pendant* utilizada para controlar robots industriales. Se busca que la interfaz permita comandar completamente el robot y permita ver en todo momento el estado de las señales del robot en ese instante.

1.3. Estructura de la memoria

El resto de la memoria de este TFM está organizado como sigue.

En el capítulo 2 se expone el robot HyReCRo. En primer lugar, se muestra el modelo cinemático de los módulos paralelos para después exponer el equivalente serie del robot y su cinemática. A continuación, se muestra el prototipo original del robot HyReCRo, el cual se tomará como referencia para el diseño del nuevo prototipo. Finalmente, se muestra el diseño de las garras que permiten la adhesión del robot, las cuales iban montadas en el prototipo original y se reutilizarán para el nuevo.

El capítulo 3 versa sobre el diseño y construcción del nuevo prototipo propuesto. Primeramente, se expone la estructura mecánica del nuevo prototipo, para luego mostrar la electrónica y su nueva disposición. Por último, se muestran imágenes del nuevo prototipo construido.

El cuarto capítulo corresponde a la programación del nuevo sistema de control del prototipo. La primera sección del capítulo introduce el sistema operativo ROS2 al lector, realizando un breve resumen de su historia y su funcionamiento. Seguidamente, se muestra el sistema de control programado, junto con la interfaz gráfica creada.

Finalmente, en el capítulo 5 se formulan las conclusiones a las que se ha llegado en este trabajo y se comentan posibles líneas de trabajo derivadas de los resultados conseguidos en este TFM.

2. Robot HyReCRo

En este TFM se presenta un prototipo del robot trepador bípedo HyReCRo. Su nombre es un acrónimo formado a partir de sus principales carácteristicas cinemáticas y su función: ***Hy****brid* ***Re****dundant* ***C****limbing* ***Ro****bot*.

Se trata de un robot trepador de estructruras metálicas diseñado por el Grupo de Automatización, Robótica y Visión por Computador de la Universidad Miguel Hernández de Elche, y su modelo cinemático, tanto directo como inverso, parámetros geométricos, espacio de trabajo, singularidades y garras, entre otros aspectos, han sido extensamente discutidos a lo largo de multitud de artículos publicados por el grupo de investgación, algunos de los cuales se citarán a continuación.

Con el fin de entender las ecuaciones y desarrollos mostrados a lo largo del TFM, en este capítulo se expondrá el modelo cinemático del robot HyReCRo, tanto de los módulos paralelos que forman las patas (Peidró et al., 2015b), como del equivalente serie del robot completo (Peidró et al., 2015a). Luego, se mostrará el prototipo original del robot, el cual se tomará como referencia para la construcción del nuevo prototipo presentado en este trabajo. Finalmente, se expondrá el diseño de las garras del robot original (Peidró et al., 2019), las cuales permiten la adhesión del robot durante su escalada y se utilizarán en el nuevo prototipo propuesto.

2.1. Modelo cinemático

El robot HyReCRo es un robot bípedo trepador de estructuras con movimiento paso a paso con 2 garras en sus extremos que permiten su pegado. Es decir, se trata de un robot cuyo método de locomoción consiste en mover una garra respecto a la otra haciendo uso de la cadena cinemática que las une. La cadena cinemática está formada por sus 2 patas unidas a la cadera mediante articulaciones de rotación (θ_A y θ_B). A su vez, cada pata está compuesta por la unión en serie de 2 módulos paralelos, los cuales se describirán a continuación, cada uno de ellos compuesto por 2 actuadores lineales.

La configuración descrita resulta en un robot de 10 GDL. Se ha demostrado que es posible explorar estructuras tridimensionales con robots de 4 GDL (Tavakoli et al., 2005, 2011). Es por ello que el robot HyReCRo cuenta con 6 GDL redundantes para esta tarea. En el caso de que fuera necesario posicionar y orientar de forma completamente arbitraria una garra respecto a la otra, el robot HyReCRo contaría con 4 GDL redundantes, puesto que la posición y la orientación relativas tienen 6 GDL. En cualquier caso, se trata de un robot redundante, por lo que el problema de cinemática inversa tendrá infinitas soluciones posibles. Los grados de libertad redundantes pueden ser empleados para realizar tareas secundarias, como evitar configuraciones articulares que resulten en singularidades, minimizar los pares de los actuadores, maximizar la maniobrabilidad del robot, evadir obstáculos o permitir la calibración del robot sin necesitar medios externos (Bennett and Hollerbach, 1991).

2.1.1. Módulos paralelos

Cada una de las patas del robot está formada por 2 módulos paralelos conectados en serie. Tal como se muestra en la figura 7, cada módulo paralelo consta de una base y una plataforma, cuyo movimiento relativo está restringido por una guía lineal, la cual se conecta a la plataforma mediante una articulación rotacional. La base y la plataforma están unidas mediante 2 actuadores lineales con rotación libre en su unión. Los 2 módulos paralelos que forman cada pata están unidos en serie al compartir la misma base, la cual se llamará cuerpo central de la pata. La plataforma del módulo paralelo inferior (número 1), actuará como garra y permitirá la adhesión magnética del robot. Por otro lado, la plataforma del módulo paralelo superior (número 2), actuará como articulación entre su pata y la cadera del robot.

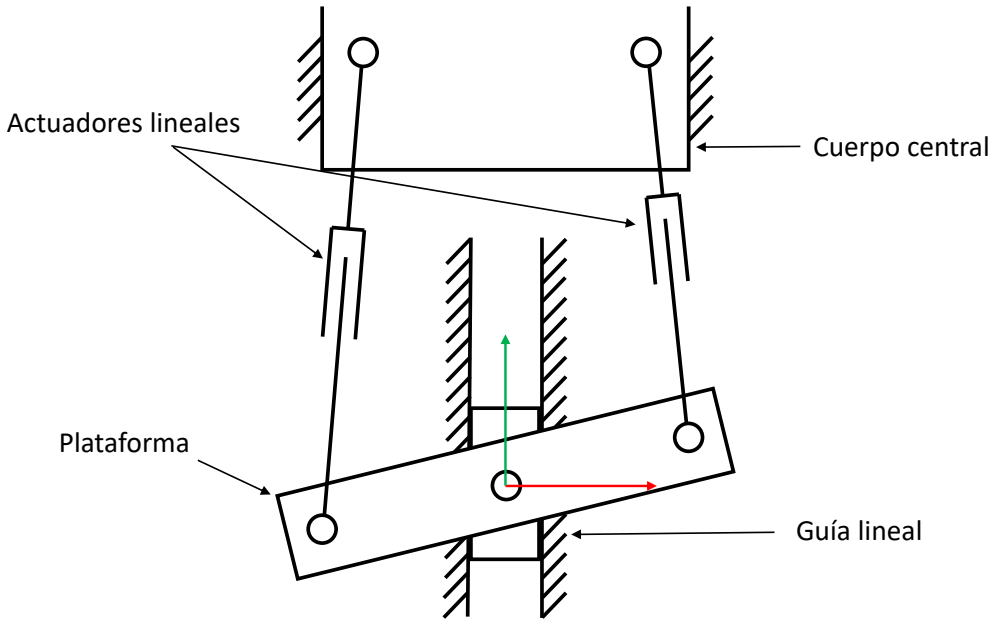


Figura 7: Representación del mecanismo de 2 GDL de los módulos paralelos.

El mecanismo de los módulos paralelos es del tipo 2RPR-PR. Modificando la longitud de los actuadores prismáticos, se puede variar la posición y orientación de la plataforma. Conocer la posición y orientación de la plataforma a partir de los valores de longitud corresponde al problema de cinemática directa.

Cinemática directa

En la figura 8 se muestran las variables utilizadas en el modelo cinemático para el módulo paralelo número i ($i \in \{1, 2\}$) de la pata j ($j \in \{A, B\}$).

La base de cada módulo paralelo tiene una longitud b desde el eje $\mathbf{y}$ del sistema de referencia hasta cada unión rotacional del actuador con la base. La plataforma

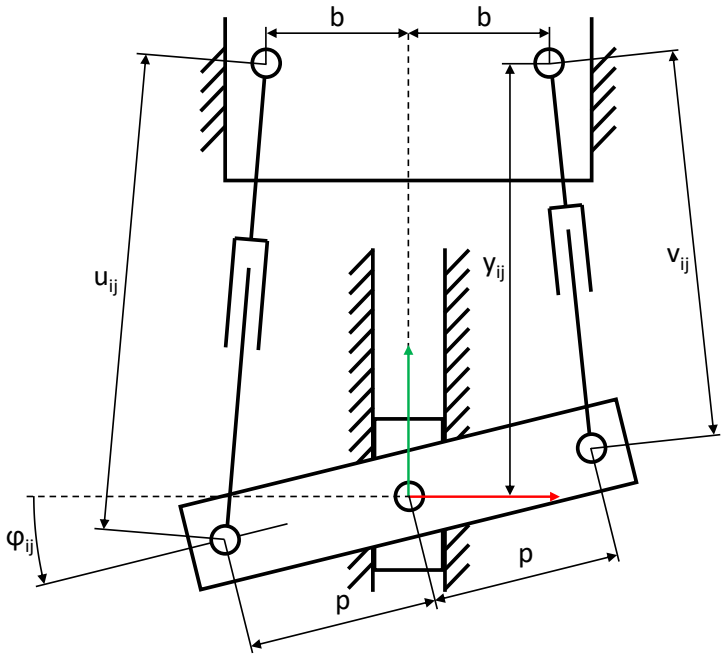


Figura 8: Cinemática de los módulos paralelos.

móvil de cada módulo tiene una longitud p desde desde su unión con la guía hasta cada unión con los actuadores.

Resolver la cinemática directa del módulo paralelo consiste en obtener la posición en el eje $\mathbf{y}$ (y_{ij}) y giro respecto al eje $\mathbf{z}$ (φ_{ij}) de la plataforma en función de las longitudes de los actuadores lineales (u_{ij} y v_{ij}). A partir de la figura 8, se extraen de forma directa las relaciones:

$$(p\,\cos\varphi_{ij} - b)^2 + (y_{ij} - p\,\sin\varphi_{ij})^2 = u_{ij}^2 \tag{1}$$

$$(p\,\cos\varphi_{ij} - b)^2 + (y_{ij} + p\,\sin\varphi_{ij})^2 = v_{ij}^2 \tag{2}$$

La combinación de estas ecuaciones genera un nuevo sistema equivalente. Sumar las ecuaciones (1) y (2) resulta en la ecuación (3), mientras que restando la ecuación (2) a la (1) se obitene la ecuación (4):

$$4\,b\,p\,\cos\varphi_{ij} = 2\,y_{ij}^2 + 2\,b^2 + 2\,p^2 - u_{ij}^2 - v_{ij}^2 \tag{3}$$

$$4\,y_{ij}\,p\,\sin\varphi_{ij} = v_{ij}^2 - u_{ij}^2 \tag{4}$$

$\cos\varphi_{ij}$ se puede despejar directamente de la ecuación (3):

$$\cos\varphi_{ij} = \frac{2\,y_{ij}^2 + 2\,b^2 + 2\,p^2 - u_{ij}^2 - v_{ij}^2}{4\,b\,p} \tag{5}$$

La ecuación (4) se puede elevar al cuadrado para obtener:

$$16\,y^2\,p^2\,(1 - \cos^2\varphi_{ij}) = (v_{ij}^2 - u_{ij}^2)^2 \tag{6}$$

Finalmente, se sustituye (5) en (6):

$$\Upsilon_{ij}^3 + k_2^{ij}\Upsilon_{ij}^2 + k_1^{ij}\Upsilon_{ij} + k_0^{ij} = 0 \tag{7}$$

donde:

$$\Upsilon_{ij} = y_{ij}^2 \tag{8}$$

$$k_2^{ij} = 2\,b^2 + 2\,p^2 - u_{ij}^2 - v_{ij}^2 \tag{9}$$

$$k_1^{ij} = \left[(b+p)^2 - \frac{u_{ij}^2 + v_{ij}^2}{2}\right]\left[(b-p)^2 - \frac{u_{ij}^2 + v_{ij}^2}{2}\right] \tag{10}$$

$$k_0^{ij} = \frac{b^2(u_{ij} + v_{ij})^2(u_{ij} - v_{ij})^2}{4} \tag{11}$$

La ecuación (7) es una función de Υ_{ij} que siempre tiene 3 raíces, de las cuales 2 pueden ser complejas. Para la raíz real positiva de mayor valor se calcula el valor de y_{ij} mediante:

$$y_{ij} = \pm\sqrt{\Upsilon_{ij}} \tag{12}$$

De las 2 soluciones obtenidas se escoge la positiva y utilizando la ecuación (5) se calcula $\cos\varphi_{ij}$. Dependiendo de la raíz del polinomio de la ecuación (7) utilizada y del signo escogido en (12) habrán 4 soluciones posibles, correspondientes a las 4 regiones diferentes expuestas en (Peidró et al., 2015a). Los valores escogidos corresponden a la región R_1 del citado artículo.

A partir de la ecuación (4) se despeja $\sin\varphi_{ij}$:

$$\sin\varphi_{ij} = \frac{v_{ij}^2 - u_{ij}^2}{4\,y_{ij}\,p} \tag{13}$$

Una vez conocidos $\sin\varphi_{ij}$ y $\cos\varphi_{ij}$, se puede calcular el valor de φ_{ij} de forma inequívoca con la función atan2:

$$\varphi_{ij} = \text{atan2}(\sin\varphi_{ij}, \cos\varphi_{ij}) \tag{14}$$

La función atan2, a diferencia de la función trigonométrica arctan convencional, devuelve el ángulo a partir de 2 parámetros de entrada. La función arctan es incapaz de determinar el cuadrante del ángulo únicamente a partir de su valor tangente. Para conocer el cuadrante concreto del ángulo, se utiliza la función atan2, que puede definirse de la siguiente manera:

$$\text{atan2}(y,x) = \begin{cases} 2\arctan\left(\dfrac{y}{\sqrt{x^2+y^2}+x}\right) & \text{si } x > 0 \text{ o } y \neq 0, \\ \pi & \text{si } x < 0 \text{ o } y = 0, \\ \textit{indefinido} & \text{si } x = 0 \text{ o } y. \end{cases} \tag{15}$$

De esta forma, se resuelve el problema de la cinemática directa del módulo paralelo al quedar completamente definidos los valores de y_{ij} y φ_{ij} a partir de los valores de las longitudes u_{ij} y v_{ij}.

Cinemática inversa

Por el contrario, la cinemática inversa consiste en expresar los valores de los actuadores lineales u_{ij} y v_{ij} en función de la posición y orientación de la plataforma y_{ij} y φ_{ij}. En este caso, se puede obtener la relación de forma sencilla a partir de las ecuaciones (1) y (2) extraídas directamente de la figura 8:

$$u_{ij} = \sqrt{(p\,\cos\varphi_{ij} - b)^2 + (y_{ij} - p\,\sin\varphi_{ij})^2} \tag{16}$$

$$v_{ij} = \sqrt{(p\,\cos\varphi_{ij} - b)^2 + (y_{ij} + p\,\sin\varphi_{ij})^2} \tag{17}$$

2.1.2. Equivalente serie del robot completo

Como ya se ha comentado, el robot completo está formado por 2 patas, cada una unida a la cadera a través de articulaciones rotacionales (θ_A y θ_B). Cada pata está compuesta por 2 módulos paralelos, cuya cinemática, tanto directa como inversa, se acaba de exponer. Sin embargo, esta arquitectura híbrida serie-paralela hace complicado su análisis cinemático de forma directa. Es por ello que, para analizarlo cinemáticamente, se modela el robot de forma simplificada como un robot serie equivalente de 8 GDL como el mostrado en la figura 9 mostrada a continuación.

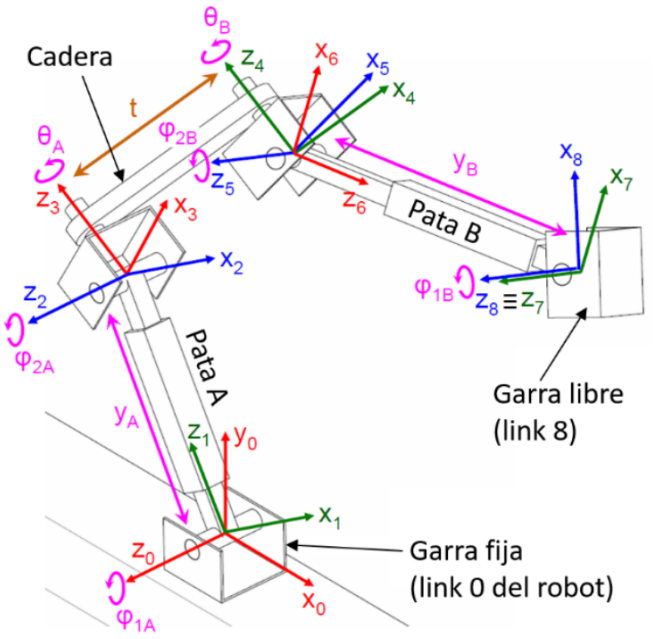


Figura 9: Equivalente serie del robot HyReCRo.

Tabla 1: Parámetros DH del equivalente serie del robot HyReCRo.

Link $i-1 \to i$	θ_i rot. z_{i-1}	d_i trans. z_{i-1}	a_i trans. x_i	α_i rot. x_i
$0 \to 1$	φ_{1A}	0	0	$-90°$
$1 \to 2$	0	y_A	0	$+90°$
$2 \to 3$	φ_{2A}	0	0	$-90°$
$3 \to 4$	θ_A	0	t	0
$4 \to 5$	θ_B	0	0	$+90°$
$5 \to 6$	φ_{2B}	0	0	$+90°$
$6 \to 7$	0	y_B	0	$-90°$
$7 \to 8$	φ_{1B}	0	0	0

El equivalente serie está formado por 6 articulaciones rotacionales $\{\varphi_{ij}, \theta_i\}$ y 2 articulaciones prismáticas $\{y_j\}$ donde i ($i \in \{1, 2\}$) indica si se trata del módulo paralelo inferior o superior y j ($j \in \{A, B\}$) indica la pata. El resto del trabajo utilizará el modelo equivalente serie, cuyos parámetros Denavit-Hartenberg (DH) se recogen en la tabla 1 y cuya cinemática, tanto directa como inversa, se analizará en la presente sección.

La cadera está dimensionada en función del parámetro t, que corresponde a la distancia entre ejes de las articulaciones rotacionales de la cadera, tal como se muestra en la figura 9. Los cuerpos centrales de los módulos paralelos también tienen sus dimensiones determinadas por el valor de la variable geométrica h, que indica la longitud de separación entre las bases de los módulos paralelos de cada pata. Se puede ver representada en la figura 11.

Cinemática directa

El problema de la cinemática directa en el equivalente serie del robot HyReCRo consiste en calcular la posición y orientación del efector final relativas a la base en función de los valores articulares de la cadera $\{\theta_j\}$ y de los módulos paralelos $\{y_{ij}, \varphi_{ij}\}$ (referidos a lo largo del trabajo como variables articulares intermedias). En la solución expuesta se considera el pie A como base (fijo), mientras que el pie B cumple el rol de efector final (libre). Los sistemas de referencia que se utilizarán para analizar la cinemática del robot se encuentran dibujados sobre el prototipo original en la figura 10 mostrada a continuación.

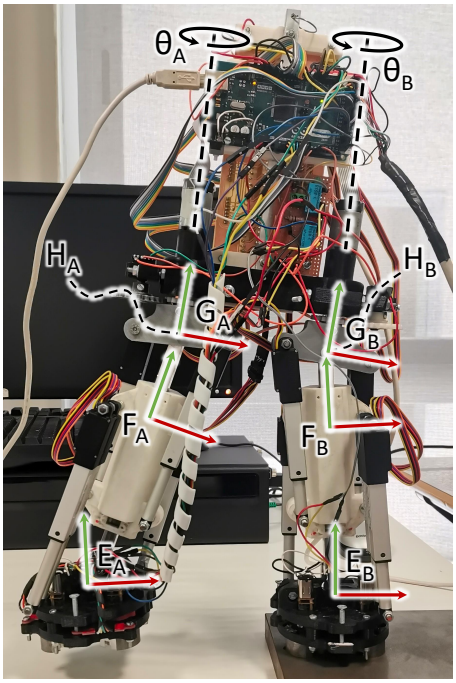


Figura 10: Sistemas de referencia del robot HyReCRo.

La posición y la orientación del efector final respecto a la base se puede expresar de forma implícita con la matriz de transformación ${}^{E_A}\mathbf{T}_{E_B}$. Una matriz de transformación es una matriz homogénea de dimensiones 4×4 que indica la posición y la orientación de un sistema de referencia respecto a otro. La matriz de transformación ${}^{E_A}\mathbf{T}_{E_B}$ tendrá la siguiente forma:

$$ {}^{E_A}\mathbf{T}_{E_B} = \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{p}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{18} $$

donde $\mathbf{p} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$ es un vector columna que contiene la traslación del sistema de refencia de la garra libre respecto a la fija y $\mathbf{R} = \begin{bmatrix} \mathbf{n} & \mathbf{o} & \mathbf{a} \end{bmatrix}$ es una submatriz de rotación 3×3 formada por los 3 vectores columna que definen la orientación de los ejes del sistema de la garra libre en la base formada por los ejes del sistema de la garra fija.

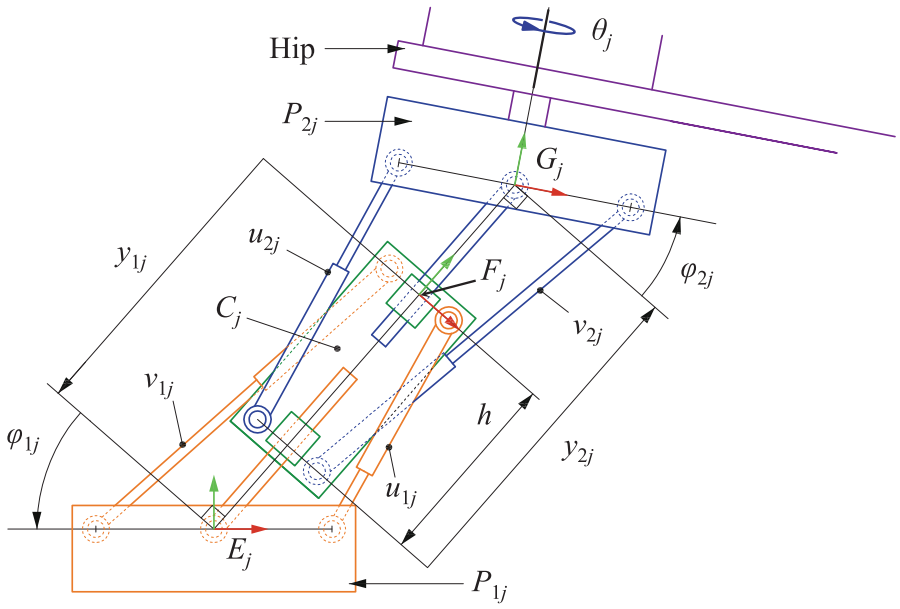


Figura 11: Cinemática de la pata j. Figura tomada de (Peidró et al., 2015a).

En la figura 11 se muestra la cinemática de la pata j del equivalente serie del robot HyReCRo. A partir de ella se puede obtener la matriz de transformación ${}^{E_j}\mathbf{T}_{H_j}$ que relaciona la pose del sistema de referencia de la articulación j de la cadera (H_j) respecto al pie de la pata j (E_j). Se obtiene mediante la multiplicación de matrices de transformación más sencillas, las cuales se pueden obtener de forma directa. A continuación se muestra la operación realizada a partir de las relaciones entre los sistemas de referencia E_j, F_j, G_j y H_j:

$$
\begin{aligned}
{}^{E_j}\mathbf{T}_{H_j} &= {}^{E_j}\mathbf{T}_{F_j}\,{}^{F_j}\mathbf{T}_{G_j}\,{}^{G_j}\mathbf{T}_{H_j} \\
&= \begin{bmatrix} \cos\varphi_{1j} & \sin\varphi_{1j} & 0 & y_{1j}\sin\varphi_{1j} \\ -\sin\varphi_{1j} & \cos\varphi_{1j} & 0 & y_{1j}\cos\varphi_{1j} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\varphi_{2j} & -\sin\varphi_{2j} & 0 & 0 \\ \sin\varphi_{2j} & \cos\varphi_{2j} & 0 & y_{2j}-h \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_j & 0 & \sin\theta_j & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_j & 0 & \cos\theta_j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta_j\cos\Phi_j & \sin\Phi_j & \sin\theta_j\cos\Phi_j & y_j\sin\varphi_{1j} \\ -\cos\theta_j\sin\Phi_j & \cos\Phi_j & -\sin\theta_j\sin\Phi_j & y_j\cos\varphi_{1j} \\ -\sin\theta_j & 0 & \cos\theta_j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{19}
$$

donde $y_j = y_{1j} + y_{2j} - h$ y $\Phi_j = \varphi_{1j} - \varphi_{2j}$.

La relación entre los sistemas de referencia de las articuaciones entre la cadera y cada pata corresponde a una traslación igual a t a lo largo de su eje $\mathbf{x}$:

$$
{}^{H_A}\mathbf{T}_{H_B} = \begin{bmatrix} \mathbf{I}_{3\times3} & [t,0,0]^T \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{20}
$$

La matriz de transformación del pie respecto a la articulación de la cadera se obtiene invirtiendo (19):

$$ {}^{H_j}\mathbf{T}_{E_j} = {}^{E_j}\mathbf{T}_{H_j}^{-1} \tag{21} $$

Realizando el producto de las 3 matrices de transformación calculadas se obtiene la matriz de transformación que expresa la posición y orientación del efector final respecto a la base:

$$ {}^{E_A}\mathbf{T}_{E_B} = {}^{E_A}\mathbf{T}_{H_A}\,{}^{H_A}\mathbf{T}_{H_B}\,{}^{H_B}\mathbf{T}_{E_B} \tag{22} $$

La matriz resultante del producto contiene el siguiente vector de posición $\mathbf{p}$:

$$ \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = y_A \begin{bmatrix} \sin\varphi_{1A} \\ \cos\varphi_{1A} \\ 0 \end{bmatrix} + y_B \begin{bmatrix} -\cos\Theta\cos\Phi_A\sin\varphi_{2B} - \sin\Phi_A\cos\varphi_{2B} \\ \cos\Theta\sin\Phi_A\sin\varphi_{2B} - \cos\Phi_A\cos\varphi_{2B} \\ \sin\Theta\sin\varphi_{2B} \end{bmatrix} + t \begin{bmatrix} \cos\theta_A\cos\Phi_A \\ -\cos\theta_A\sin\Phi_A \\ -\sin\theta_A \end{bmatrix} \tag{23} $$

donde $\Theta = \theta_A - \theta_B$.

Así mismo, la submatriz de rotación contenida en ${}^{E_A}\mathbf{T}_{E_B}$ equivale a:

$$ \mathbf{R} = \begin{bmatrix} \sin\Phi_A\sin\Phi_B + \cos\Theta\cos\Phi_A\cos\Phi_B & \sin\Phi_A\cos\Phi_B - \cos\Theta\cos\Phi_A\sin\Phi_B & \sin\Theta\cos\Phi_A \\ \cos\Phi_A\sin\Phi_B - \cos\Theta\sin\Phi_A\cos\Phi_B & \cos\Phi_A\cos\Phi_B + \cos\Theta\sin\Phi_A\sin\Phi_B & -\sin\Theta\sin\Phi_A \\ -\sin\Theta\cos\Phi_B & \sin\Theta\sin\Phi_B & \cos\Theta \end{bmatrix} \tag{24} $$

De esta forma queda resuelta la cinemática directa del equivalente serie del robot HyReCRo al definir la submatriz de rotación y el vector de posicón expresados en función de sus variables articulares.

Cinemática inversa

La cinemática inversa del robot consiste en calcular las coordenadas articulares que consiguen una posición y orientación concretas especificadas a partir de una matriz de transformación. En el caso de robots no redundantes, existe un número finito de soluciones a la cinemática inversa. Para el robot HyReCRo, en cambio, al tener múltiples grados de redundancia, existirán infinitas configuraciones articulares que consigan una misma pose.

El problema de la cinemática inversa parte conociendo la matriz de transformación ${}^{E_A}\mathbf{T}_{E_B}$, la cual está formada por la submatriz de rotación y el vector de posición de las ecuaciones (24) y (23) respectivamente.

A partir de la matriz de rotación $\mathbf{R}$, y suponiendo que los ejes $\mathbf{z}$ de los pies no son paralelos ni antiparalelos ($\mathbf{R}_{33} \neq \pm 1$), se puede calcular directamente el valor de Θ a partir de $\mathbf{R}_{33}$:

$$\Theta = \pm \arccos \mathbf{R}_{33} \tag{25}$$

De las 2 soluciones posibles soluciones se utilizará únicamente la positiva.

A partir de los elementos restantes de la tercera columna de la matriz de rotación y conociendo el valor de Θ se pueden extraer los valores del coseno y el seno de Φ_A:

$$\cos \Phi_A = \frac{\mathbf{R}_{13}}{\sin \Theta} \tag{26}$$

$$\sin \Phi_A = -\frac{\mathbf{R}_{23}}{\sin \Theta} \tag{27}$$

Conociendo $\sin \Phi_A$ y $\cos \Phi_A$ es posible calcular inequívocamente Φ_A mediante la función atan2, definida previamente en (15):

$$\Phi_A = \text{atan2}(\sin \Phi_A, \cos \Phi_A) \tag{28}$$

De forma análoga, a partir de los elementos restantes de la tercera fila de la matriz $\mathbf{R}$, conociendo el valor de Θ y mediante la función atan2, se puede extraer el valor de Φ_B de la forma:

$$\cos \Phi_B = -\frac{\mathbf{R}_{31}}{\sin \Theta} \tag{29}$$

$$\sin \Phi_B = \frac{\mathbf{R}_{32}}{\sin \Theta} \tag{30}$$

$$\Phi_B = \text{atan2}(\sin \Phi_B, \cos \Phi_B) \tag{31}$$

Al ser un robot con varios grados de redundancia, existirán infinitas configuraciones articulares que resulten en una misma posición y orientación del efector final, y por lo tanto, infinitas soluciones de la cinemática inversa. Por ende, si se quiere obtener una solución, será necesario dar valores a determinadas coordenadas articulares. Concretamente, el equivalente serie del robot HyReCRo expuesto cuenta con 8 GDL, por lo que habrá que dar valores a 2 variables de forma previa para obtener una solución única para una tarea que requiere 6 GDL, como es la de posicionar el efector final de forma totalmente arbitraria en el espacio tridimensional. En esta solución, se le darán valores a las variables articulares intermedias φ_{2B} e y_B. De esta forma, se extrae directamente el valor de θ_A a partir del elemento p_z del vector de posición:

$$\theta_A = \begin{cases} \arcsin\left(\dfrac{y_B \sin\Theta \sin\varphi_{2B} - p_z}{t}\right) \\ \pi - \arcsin\left(\dfrac{y_B \sin\Theta \sin\varphi_{2B} - p_z}{t}\right) \end{cases} \tag{32}$$

Una vez más, se obtienen 2 valores de θ_A en diferentes cuadrantes. Se utilizará el valor correspondiente al primer o cuarto cuadrante. De esta forma, se está siguiendo 1 de las 4 posibles ramas de soluciones que se bifurcan en las ecuaciones (25) y (32).

Conociendo θ_A y Θ se obteniene θ_B de forma directa:

$$\theta_B = \theta_A - \Theta \tag{33}$$

A continuación, a partir de los elementos p_x y p_y del vector de posición se genera un sistema de ecuaciones despejando el primer sumando de la ecuación (23):

$$\left.\begin{aligned} y_A \sin\varphi_{1A} &= p_x + y_B(\cos\Theta\cos\Phi_A \sin\varphi_{2B} + \sin\Phi_A \cos\varphi_{2B}) - t\cos\theta_A \cos\Phi_A \\ y_A \cos\varphi_{1A} &= p_y + y_B(\cos\Phi_A \cos\varphi_{2B} - \cos\Theta \sin\Phi_A \sin\varphi_{2B}) + t\cos\theta_A \sin\Phi_A \end{aligned}\right\} \tag{34}$$

Para facilitar la legibilidad, se sustituirá el lado derecho de las ecuaciones (34) por las variables K_1 y K_2:

$$\left.\begin{aligned} y_A \sin\varphi_{1A} &= K_1 \\ y_A \cos\varphi_{1A} &= K_2 \end{aligned}\right\} \tag{35}$$

De donde se puede extraer el valor de φ_{1A} mediante la función atan2:

$$\varphi_{1A} = \text{atan2}(K_1, K_2) \tag{36}$$

Y de donde también se puede calcular y_A:

$$y_A = \sqrt{K_1^2 + K_2^2} \tag{37}$$

Finalmente, se obtienen los ángulos φ_{ij} restantes:

$$\varphi_{2A} = \varphi_{1A} - \Phi_A \tag{38}$$

$$\varphi_{1B} = \Phi_B + \varphi_{2B} \tag{39}$$

Esta solución a la cinemática inversa, no obstante, únicamente es válida cuando los ejes **z** de los pies no son paralelos ni antiparalelos; es decir, cuando $\mathbf{R}_{33} \neq \pm 1$.

En el caso de que los ejes **z** de los pies sean paralelos o antiparalelos ($\mathbf{R}_{33} = \pm 1$), los elementos restantes de la tercera fila y columna de la matriz $\mathbf{R}$ serán nulos, por lo que el método expuesto no será válido. A continuación, se propone otra solución a la cinemática inversa cuando $\mathbf{R}_{33} = \pm 1$.

El valor de Θ se puede obtener de la misma forma. Sin embargo, la ecuación únicamente tendrá una única solución, puesto que $\mathbf{R}_{33} = \pm 1$:

$$\Theta = \arccos \mathbf{R}_{33} \tag{40}$$

De aquí en adelante los 2 métodos difieren. En este caso, además de dar valores a φ_{2B} e y_B, será preciso dar valor a la variable φ_{1B} para obtener una solución única. De esta forma, se puede calcular de forma directa el valor de Φ_B:

$$\Phi_B = \varphi_{1B} - \varphi_{2B} \tag{41}$$

Al ser los elementos restantes de la tercera fila y columna de la matriz $\mathbf{R}$ nulos, se deberán usar los elementos de las 2 primeras filas y columnas para obtener el valor de Φ_A. Concretamente, a partir de los elementos $\mathbf{R}_{12}$ y $\mathbf{R}_{22}$ y mediante las razones trigonométricas de la suma y diferencia de ángulos se obtiene:

$$\sin(\Phi_A - \mathbf{R}_{33}\Phi_B) = \sin\Phi_A \cos\Phi_B - \mathbf{R}_{33} \cos\Phi_A \sin\Phi_B = \mathbf{R}_{12} \tag{42}$$

$$\cos(\Phi_A - \mathbf{R}_{33}\Phi_B) = \cos\Phi_A \cos\Phi_B + \mathbf{R}_{33} \sin\Phi_A \sin\Phi_B = \mathbf{R}_{22} \tag{43}$$

Cabe recordar que, al ser los ejes **z** de los pies paralelos o antiparalelos, $\mathbf{R}_{33}$ siempre será 1 o -1, por lo que solo alterará en el signo a las ecuaciones anteriores y siempre se cumplirá la propiedad.

Una vez más, se puede usar la función atan2 definida en (15) y despejar Φ_A:

$$\Phi_A = \text{atan2}(\mathbf{R}_{12}, \mathbf{R}_{22}) + \mathbf{R}_{33}\, \Phi_B \tag{44}$$

El resto de variables articulares se calculan de la misma forma que en el método expuesto con anterioridad, cuando los pies no son paralelos ni antiparalelos. θ_A se calcula de igual forma que en la ecuación (32). θ_B se obtiene de (33). Se genera el sistema expuesto en la ecuación (34) y se extrae φ_{1A} de (36) y y_A de (37). Por último, se obtiene φ_{2A} de la ecuación (38).

Así, queda definida una configuración articular que satisfacerá la posición y orientación descritas en la matriz de transformación introducida, y, por ende, la cinemática inversa del robot. No obstante, cabe recordar la alta redundancia que presenta el robot. Es por ello que existen multitud de alternativas para resolver la cinemática inversa y ésta es solo una de ellas. Los grados de libertad redundantes pueden emplearse para conseguir tareas secundarias, resultando en soluciones a la cinemática inversa donde no sea necesario dar valores a algunas articulaciones.

2.2. Prototipo original

En esta sección se mostrará el prototipo original del robot HyReCRo, propuesto en (Peidró et al., 2019), el cual se ha empleado como referencia para la creación del nuevo prototipo presentado en los capítulos siguientes.

Tal como se expone en (Peidro et al., 2016), los parámetros que definen la geometría del robot $\{t, h, b, p\}$ afectan a su espacio de trabajo, permitiendo o no realizar ciertos movimientos como los cambios de plano convexos. En (Peidró et al., 2015a) se presenta una herramienta que permite simular el espacio de trabajo del robot en función de los parámetros geométricos del mismo. Utilizando los recursos mencionados, el prototipo se diseñó con las siguientes variables geométricas: $t = 110$, $h = 70$, $b = 25$ y $p = 31.5$ (valores en milímetros).

La mayoría de las piezas que componen el prototipo están modeladas utilizando *software* CAD e impresas con impresora 3D. El resto de las partes del robot están fabricadas en aluminio.

Las patas del robot, cada una de las cuales está formada por 2 módulos paralelos montados en serie, están unidas a la cadera mediante articulaciones de rotación actuadas por motores DC Maxon A-Max 22. Además, cada articulación lleva incorporada una reductora con un factor de reducción 590:1 con el fin de generar el par necesario y un potenciómetro para conocer su posición angular. En los módulos paralelos que componen el robot se emplean actuadores prismáticos DC Actuonix L12-50-210-12-P, los cuales incorporan potenciómetros embebidos que dan información sobre su longitud en cualquier instante.

Las garras montadas en el extremo de cada pata se expondrán en la siguiente sección.

El control del robot se realiza con un microcontrolador Arduino Mega 2560. El microcontrolador permite leer señales analógicas y digitales y enviar señales a los actuadores, las cuales pasan por una placa diseñada para amplificarlas y filtrarlas.

Las acciones se comandan por medio de un mando con una cruceta y 6 botones. La cruceta controla la traslación del efector final contenida en el plano formado por los ejes $\mathbf{x}$ e $\mathbf{y}$ del sistema de referencia de la base del robot. Mediante 2 botones se controla el giro de la garra alrededor del eje $\mathbf{z}$. Los 3 movimientos de traslación y rotación descritos se calculan siguiendo la relación $\Delta\xi = \mathbf{J}\Delta\mathbf{q}$, donde $\Delta\xi$ es un vec-

tor columna que contiene los incrementos de la posición y orienación comandados, $\mathbf{J}$ es la matriz Jacobiana expresada respecto al sistema de la base y $\Delta\mathbf{q}$ es un vector columna formado por los incrementos articulares que satisfacerán los incrementos comandados.

El control de los giros de la cadera (θ_A y θ_B) se realizan de forma independiente utilizando los 4 botones restantes.

Las señales para comandar los motores que accionan los imanes de las garras se producen cuando se pulsan botones montados directamente en la cadera del robot. Junto a ellos existe otro botón que cambia el sistema de referencia del robot en función de la garra que está fija y actúa como base del mismo.

2.3. Garras

El propósito original del robot HyReCRo es la escalada de estructruras metálicas tridimensionales para llevar a cabo tareas de mantenimiento e inspección. Es por ello que precisa de una forma de adherirse a la superficie mientras se desplaza por ella. En (Peidró et al., 2019) se presenta el diseño de unas garras magnéticas instaladas en cada pie del robot con la capacidad de aguantar los grandes pares generados por el peso del robot durante la escalada.

En esta sección se expone el diseño de las garras que aloja el prototipo original del robot HyReCRo y que se utilizarán en el nuevo prototipo que se presentará en el capítulo 3.

Las garras propuestas consisten en 3 imanes permanentes conmutables de forma mecánica distribuidos cada 120° en el plano de la garra. En el diseño de la garra, se persiguen 2 objetivos fundamentales: evitar el desprendimiento y el deslizamiento de la garra cuando está pegada. Para ello, se consideran 3 posturas principales adoptadas durante la exploración de estructuras reticulares, las cuales se muestra en la figura 12. A su vez, en cada postura se consideran 6 casos diferentes en función del eje y sentido en el que actúa la gravedad. Esto resulta en un total de 18 escenarios diferentes considerados durante el análisis realizado para el diseño de las garras.

El desprendimiento de los imanes de la garra ocurre cuando uno el Punto de Momento Cero sale de la envoltura convexa del área de contacto de la garra con la superficie a la que está pegada. El desarrollo de las ecuaciones que consideran los peores casos de los 18 escenarios contemplados y permiten dimensionar el diseño de la garra se discuten en los capítulo 4 y 5 de (Peidró et al., 2019). En el citado artículo se propone el diseño mostrado en la figura 13, donde se puede observar una vista explosionada de la garra.

El accionamiento de los imanes de la garra se realiza por medio del circuito mostrado en la figura 14. Cuando el circuito recibe un pulso de entrada, se activan los motores DC que rotan 180° su respectivo imán permanente hasta que se active

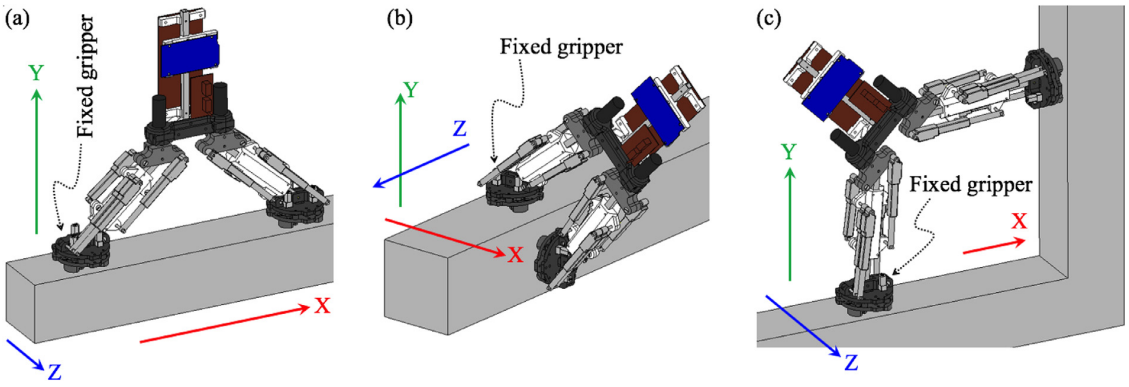


Figura 12: Posturas adoptadas durante la exploración cuando se realiza un desplazamiento longitudinal (a), una transición exterior entre superficies (b) y una transición interior entre superficies (c). Figura tomada de (Peidró et al., 2019).

su final de carrera de forma independiente, redirigiendo así el campo magnético de cada imán. En caso de que se esté realizando el pegado, el campo magnético se redirigirá para cerrarse a través de la superficie trepada. En caso contrario, cuando se quiera despegar la garra, la redirección del campo magnético se cerrará a través de un circuito interno.

Esta solución consigue una fuerza de pegado que supera la necesaria para evitar el desprendimiento de la garra. Sin embargo, cuando la fuerza de gravedad actúa en dirección de los ejes **x** o **z** del robot, la fuerza generada no es suficiente para evitar el deslizamiento de la garra. Para solucionarlo, se diseñaron unos accesorios de fricción que solventan el problema.

En el capítulo 6 de (Peidró et al., 2019) se discute y calcula el coeficiente de fricción necesario para evitar el deslizamiento. Se determinó que para los 18 escenarios propuestos, la fuerza de la gravedad causa el mayor deslizamiento en la garra cuando actúa en el sentido negativo del eje **z** del sistema de referencia del robot. Concretamente, cuando el robot está realizando un movimiento de desplazamiento longitudinal (figura 12a) y la gravedad actúa en el sentido descrito, el coeficiente de fricción necesario para que no ocurra deslizamiento es de 0.3392.

El accesorio de fricción diseñado consiste en 3 almohadillas circulares de goma Vytaflex® montados en una pieza de forma similar a un *fidget spinner*, donde las almohadillas se encuentran distribuidas de la misma forma que los imanes, con un desfase de 120° entre ellas. El accesorio puede ser montado o desmontado de la garra con facilidad y su altura puede ser regulada mediante 3 tornillos para evitar la separación entre la superficie y las gomas o los imanes, tal como se muestra en la figura 15.

Finalmente, en la figura 16 se muestra un renderizado de la garra completa, tal y como se diseñó en (Peidró et al., 2019).

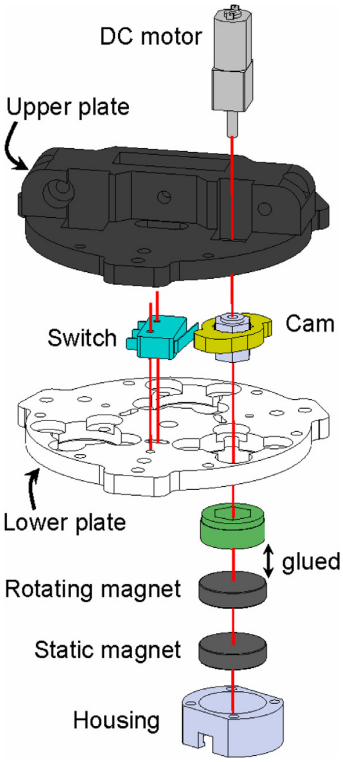


Figura 13: Vista explosionada de la garra. Figura tomada de (Peidró et al., 2019).

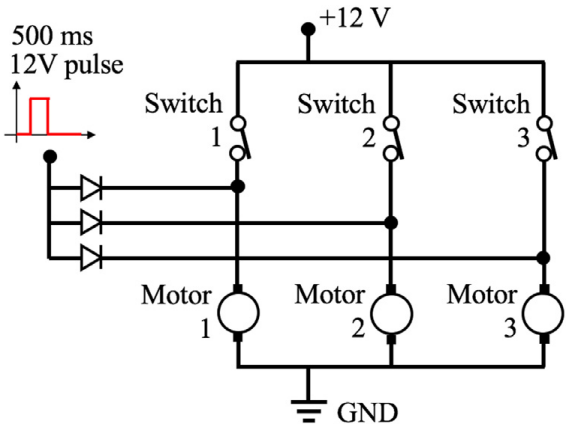


Figura 14: Circuito de accionamiento de los imanes permanentes conmutables. Figura tomada de (Peidró et al., 2019).

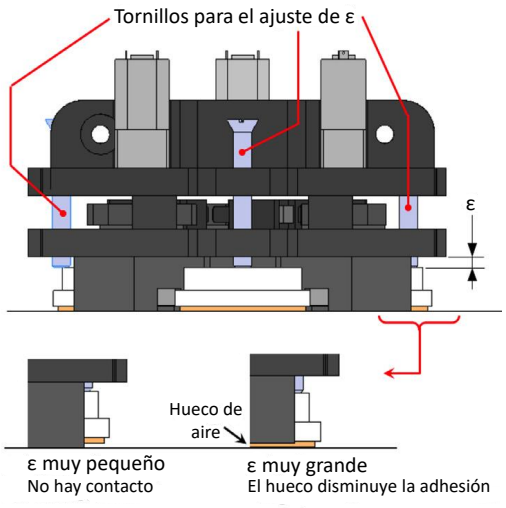


Figura 15: Ajuste de la altura del accesorio de fricción. Figura tomada de (Peidró et al., 2019).

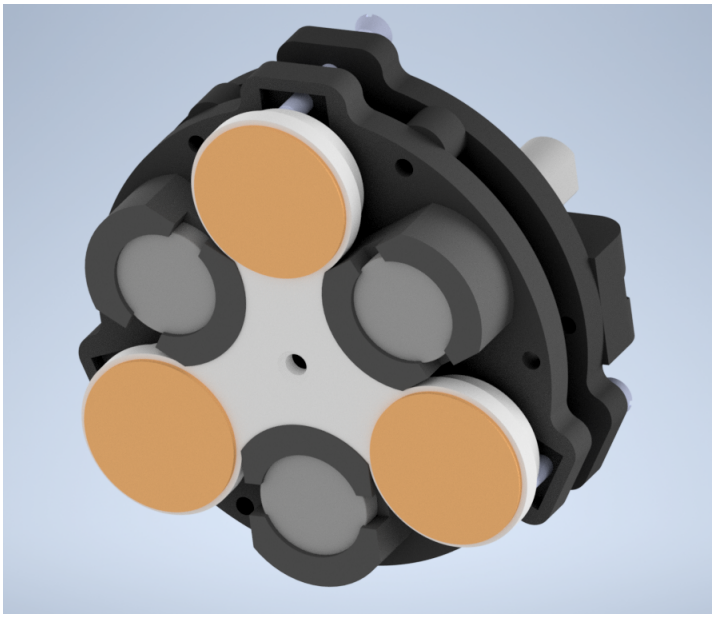


Figura 16: Garra completa, tal y como se diseñó en (Peidró et al., 2019).

En el artículo donde se diseñaron originalmente, el acercamiento y pegado de la garra se realizaba ajustando su pose de forma manual, lo cual es complicado de hacer con precisión debido a los lugares de difícil visión o acceso donde suele operar el robot. Es por ello que en el Trabajo de Fin de Grado (Fabregat, 2021) se incorporaron 3 sensores ópticos modelo VL6180 para estimar el plano de la superficie y aproximar automáticamente mediante control cinemático la garra libre a la superficie de pegado. La garra propuesta se puede observar en la figura 17.

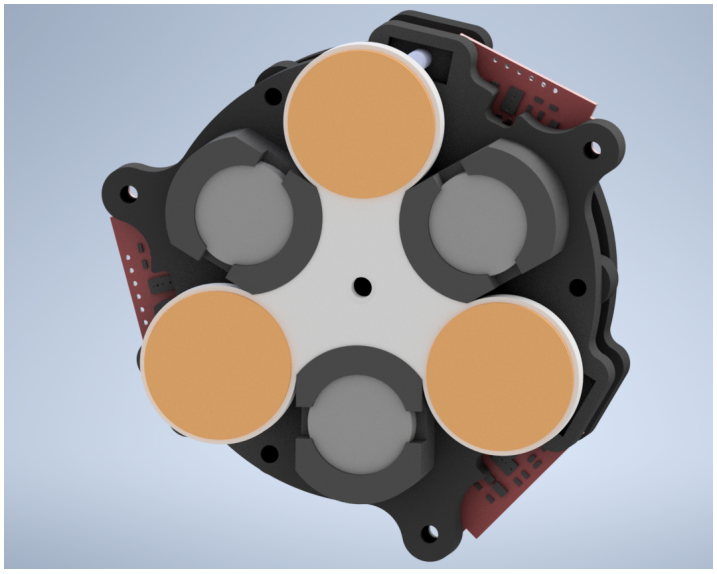


Figura 17: Garra completa con los sensores ópticos montados.

3. Diseño y construcción de un prototipo del robot HyReCRo

El objetivo principal abarcado en el TFM es el de la construcción de un nuevo prototipo del robot HyReCRo. En este capítulo se expondrá el nuevo diseño propuesto.

La solución presentada busca remediar algunas características del prototipo anterior. Los objetivos perseguidos con el nuevo diseño, y las secciones donde se abordan, son:

- Eliminar holguras provenientes del desgaste de las piezas del prototipo anterior.
- Mejorar la resistencia de las patas a torsiones. (Sección 3.1.1)
- Mejorar la resistencia de la articulación que une las patas a la cadera a esfuerzos radiales. (Sección 3.1.2)
- Descender el centro de gravedad y disminuir el peso del robot con el fin de reducir los pares producidos cuando el centro de gravedad se aleja del punto de apoyo del robot. (Secciones 3.1.2 y 3.2)
- Simplificar la electrónica y el control del robot con el propósito de reducir el tamaño y peso del mismo, así como los componentes y circuitos de los que precisa. (Sección 3.2)

3.1. Diseño mecánico

Todo el diseño mecánico de las piezas del prototipo se ha hecho utilizando Autodesk Inventor. A excepción de las garras, las cuales mantienen el diseño original presentado en (Peidró et al., 2019), se propone un nuevo diseño íntegro del *hardware* del robot, tomando como referencia el prototipo original.

En este apartado se empezará hablando del nuevo diseño de las patas, cada una formada por 2 módulos paralelos, para luego mostrar la cadera que los une formando el robot completo.

3.1.1. Patas

Como se ha mostrado en la sección 2.1.1, cada pata que compone al robot está formada por 2 módulos paralelos que comparten una misma base. A su vez, los 2 módulos paralelos de cada pata comparten una guía lineal que restringe el movimiento de las plataformas (garra del robot y pieza que sirve como articulación con la cadera) respecto a un eje.

Previamente, cada módulo paralelo disponía de una guía pasiva de aluminio de perfil cuadrado con lados de 10 mm (figura 18). Cada guía se deslizaba longitudinalmente a lo largo del cuerpo central de la pata, que actúaba como base común, restringiendo el movimiento de sendas plataformas.

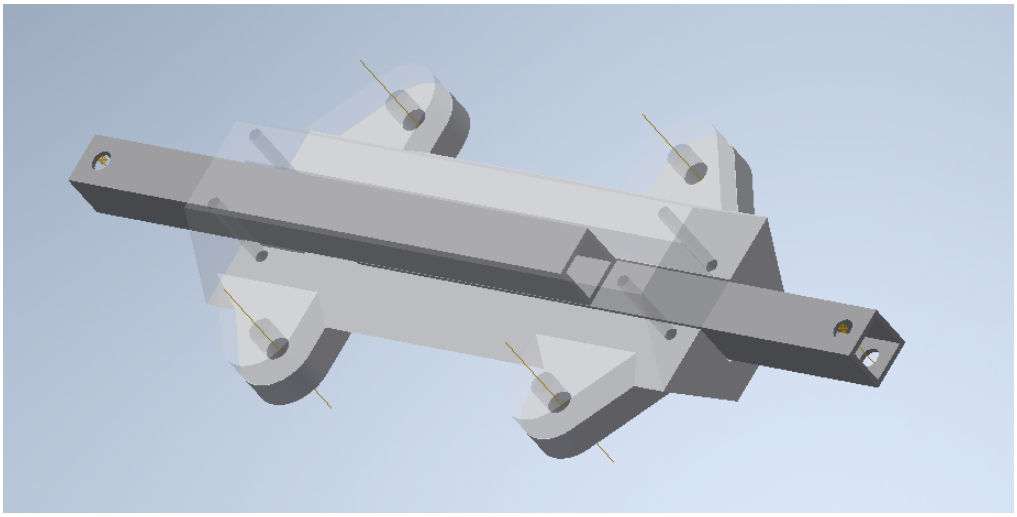


Figura 18: Mecanismo de guías del prototipo original.

Debido a los reducidos perfiles de las guías, la pata era débil ante torsiones sobre el eje de deslizamiento de las guías. En el nuevo diseño de las patas, se busca dotarlas de una mayor rigidez, eliminando la susceptibilidad de las patas a las torsiones descritas.

La solución adoptada ha sido la de aumentar el perfil de las guías notablemente. Para conseguir ubicar 2 guías considerablemente más grandes sin aumentar el volumen ocupado por cada pata, se ha optado por variar la forma en la que estaban dispuestas las guías. En lugar de tratarse de 2 guías de ejes paralelos, se han trasladado los ejes para hacerlos colineales, de manera que ambas guías comparten el mismo eje de desplazamiento. De esta forma, se ha conseguido utilizar una guía 3 veces más grande que las originales, de 30 mm de lado, junto a otra guía de 24 mm de lado. El diseño final, con el cual se consigue dotar a la pata con una mayor resistencia a torsiones, se muestra en la figura 19 a continuación.

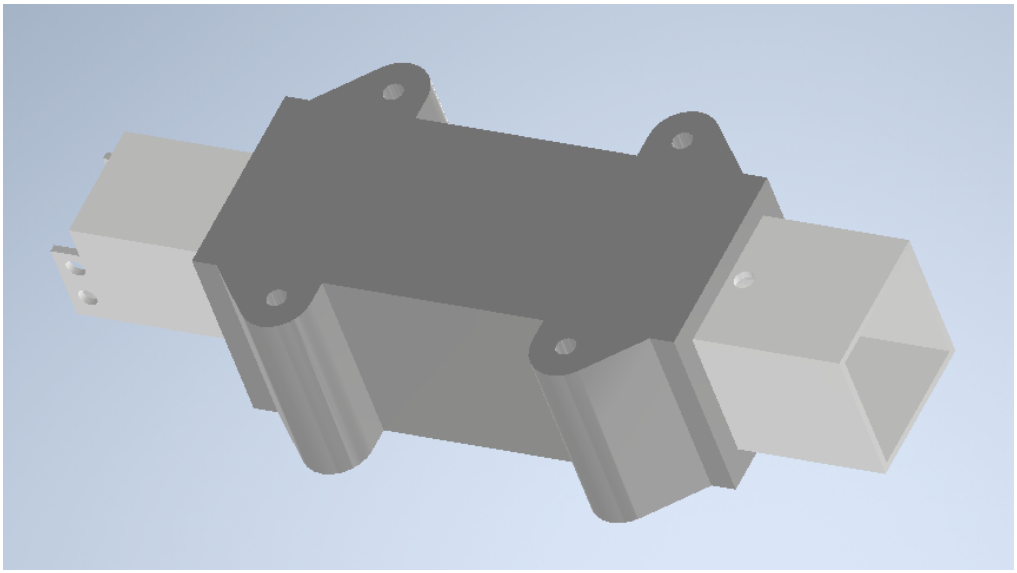


Figura 19: Mecanismo de guías del nuevo prototipo.

Aprovechando el rediseño del cuerpo central de las patas, se ha optado por reducir la longitud del parámetro geométrico h. En (Peidro et al., 2016) se estudia como afecta cada parámetro geométrico al espacio de trabajo del robot. Se muestra como, al reducir el valor de h, aumenta el espacio de trabajo a orientación constante del robot.

El objetivo perseguido al aumentar el espacio de trabajo es el de realizar transiciones entre planos perpendiculares convexos. Es por ello que, utilizando la herramienta de simulación del espacio de trabajo presentada en (Peidró et al., 2015a), se ha decidido realizar el nuevo diseño utilizando un valor geométrico $h = 60$ mm.

Originalmente, la articulación rotacional entre la guía y la plataforma del módulo paralelo inferior (garra de la pata) se realizaba directamente mediante una perforación en la guía, la cual se introducía en un espacio de la garra que permitía la unión rotacional. Sin embargo, la nueva guía es demasiado grande para introducirla en el espacio destinado para ello en la garra original. Al querer mantener la garra original, se ha diseñado una pieza, la cual se ubica al final de la guía a modo de tapón, que permite emular la unión original entre deslizadera y garra. En la figura 20, mostrada a continuación, se puede aprecriar la unión entre la garra y la deslizadera inferior de cada pata.

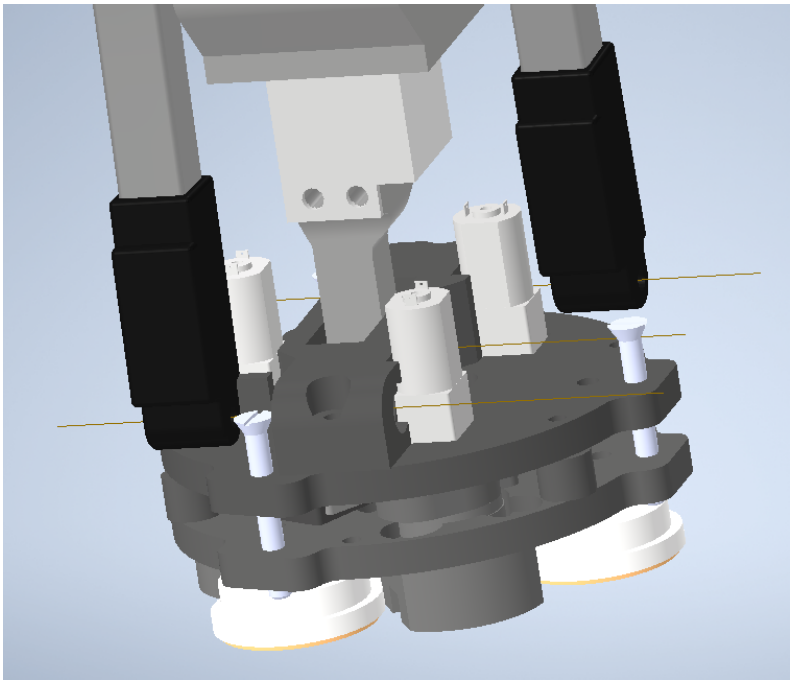


Figura 20: Articulación entre la guía inferior y la garra.

Para la unión rotacional entre la guía y la plataforma del módulo superior se ha realizado una pieza, semejante a la original, que, a su vez, sirve como articulación rotacional entre la pata y la cadera (figura 21). El sistema de articulación entre la pata y la cadera se explicará en la sección 3.1.2.

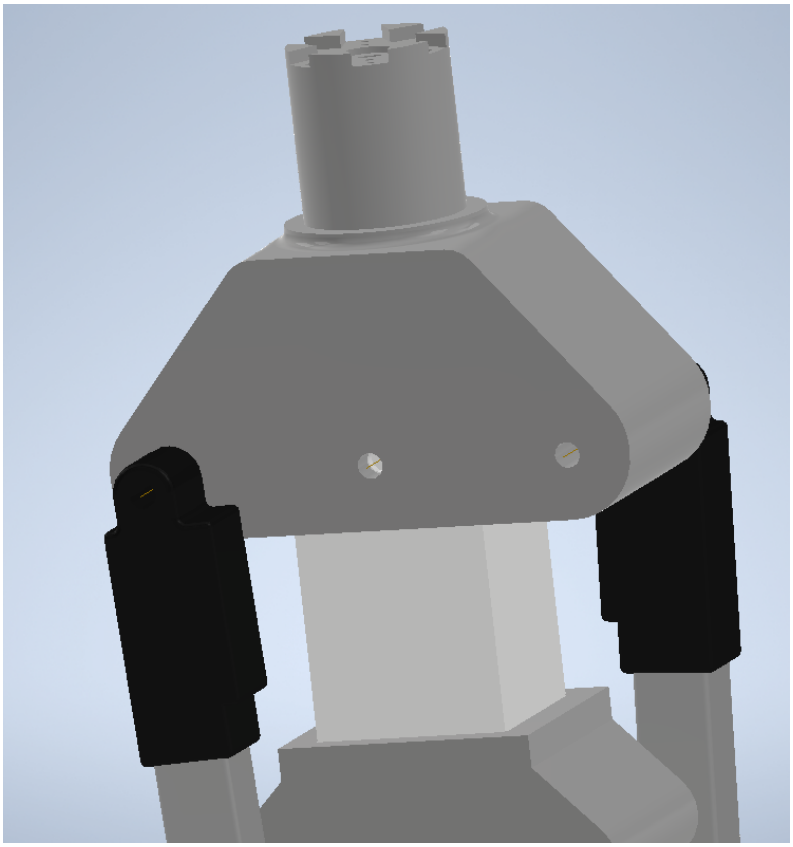


Figura 21: Articulación entre la guía superior y su plataforma.

Finalmente, se muestra la pata completa con una configuración de actuadores arbitraria (figura 22). En la imagen se puede apreciar la unión de todos los elementos descritos. Con el nuevo diseño de cada pata, se consigue una importante rigidez estructural, dotándola de resistencia a torsiones.

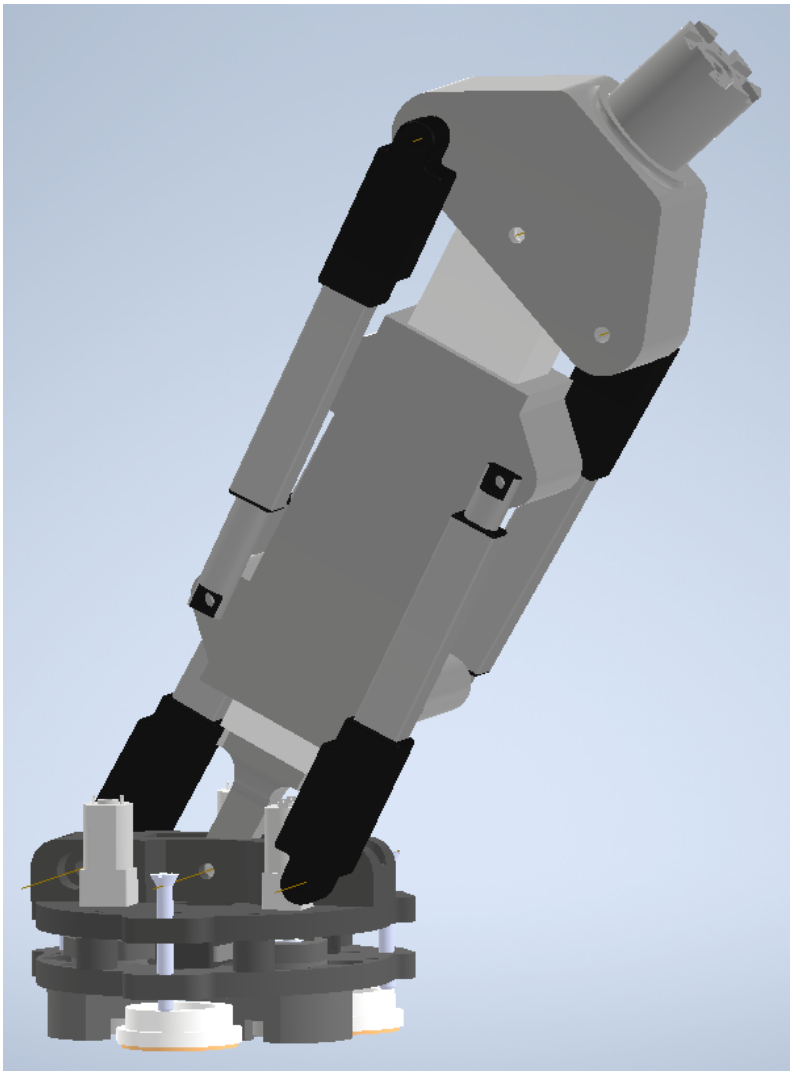


Figura 22: Pata completa.

Si se observa la vista seccionada de la pata completa (figura 23), se pueden apreciar varios detalles que merecen ser destacados:

- En la plataforma del módulo parelelo superior existe un espacio para que la guía se mueva libremente sin restringir el rango total de rotación del mecanismo.
- Las guías inferior y superior deslizan entre ellas gracias a unos patines que dotan de rigidez y permiten el movimiento lineal de ambas a lo largo del mismo eje.
- La pieza que permite la unión entre la guía inferior y la garra está dimensionada, al igual que la plataforma superior, para que no interfiera con el rango completo de rotación del módulo paralelo.

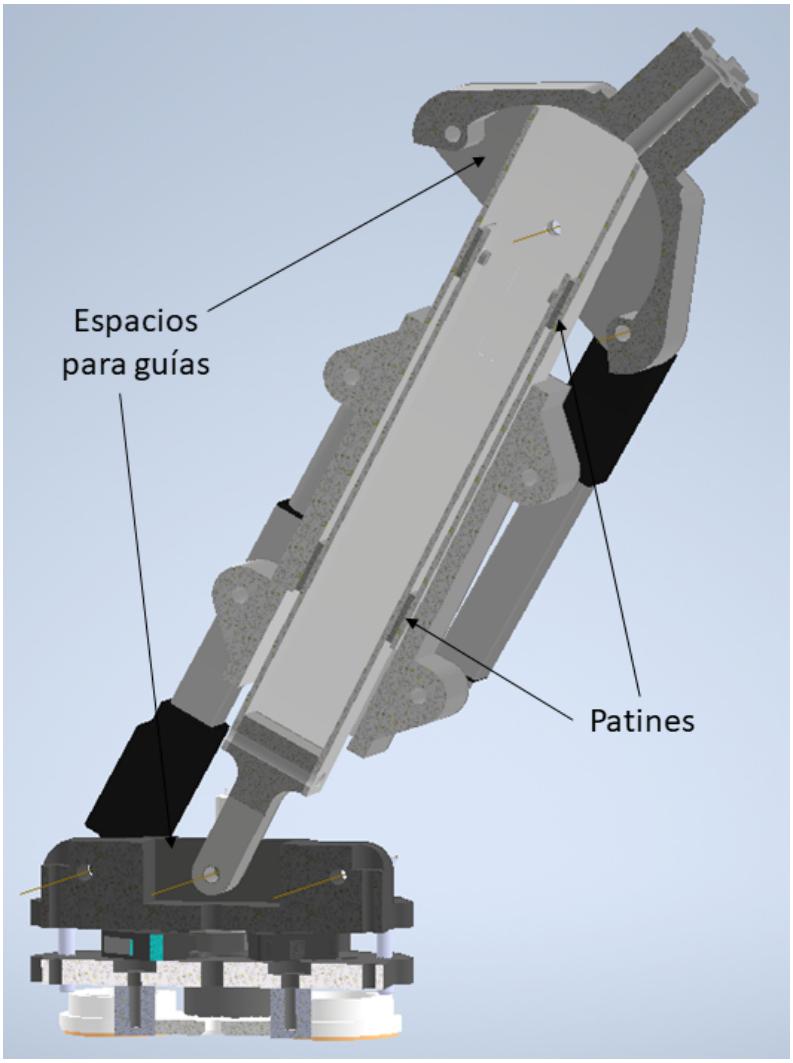


Figura 23: Pata seccionada.

3.1.2. Robot completo

El robot completo está formado por 2 patas idénticas unidas mediante articulaciones rotacionales a una cadera. En esta sección, se describirá el diseño mecánico la cadera, así como la unión de ésta a ambas patas, formando el robot completo.

En el prototipo original, a la cadera iba adherida una especie de mástil donde se ubicaba toda la electrónica (figura 24). Esto elevaba el centro de gravedad del robot, lo cual es perjudicial, puesto que al inclinarse es más usual que el centro de gravedad caiga fuera de la base, produciéndose momentos que lo desestabilicen.

Por otro lado, la articulación entre la cadera y cada pata estaba constituida únicamente por el eje de los actuadores (figura 24), lo cual conllevaba que la articulación fuese vulnerable a esfuerzos radiales en el eje de los motores.

Para reducir el centro de gravedad del robot, en el nuevo diseño se ha optado por disponer la electrónica y los componentes directamente en la cadera, reduciendo el

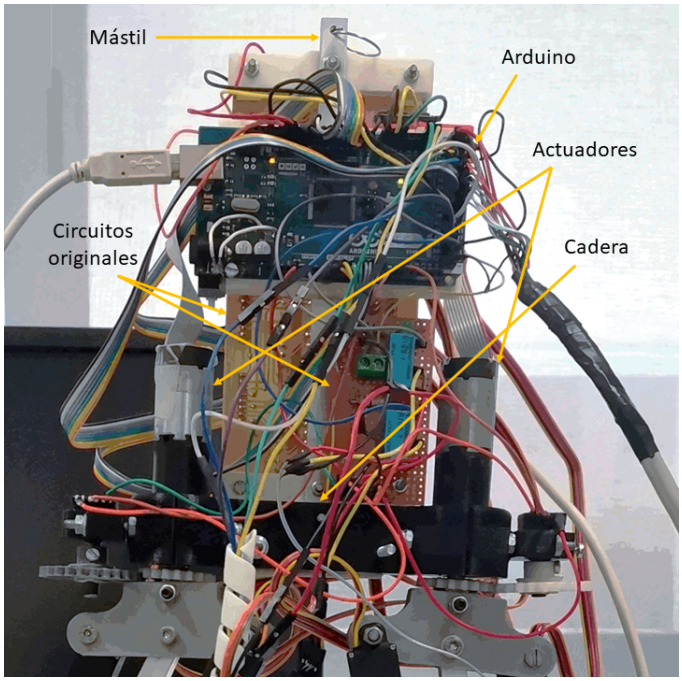


Figura 24: Cadera del prototipo original.

tamaño vertical del robot y descendiendo el centro de gravedad, tal como se puede observar en la figura 25. La electrónica empleada se expondrá en detalle en la sección 3.2.

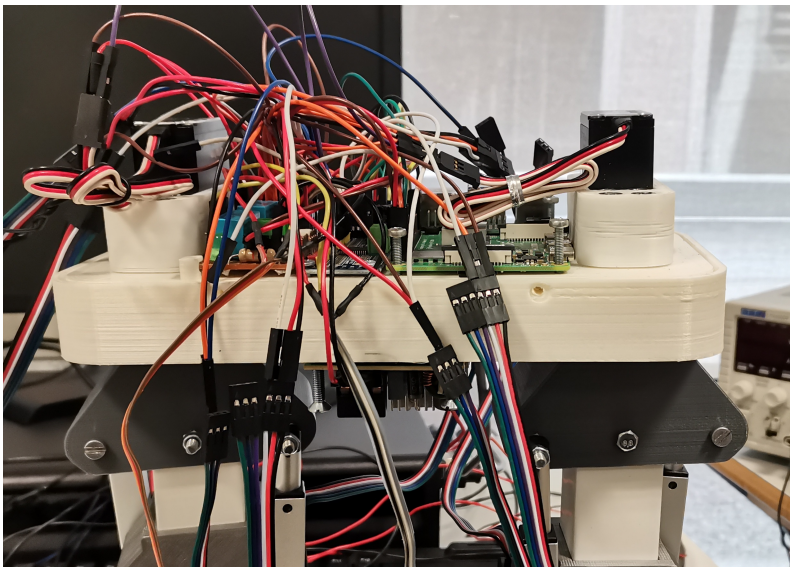


Figura 25: Cadera del nuevo prototipo.

Utilizando los mismos recursos empleados para calcular el nuevo valor de h en la sección 3.1.1, se ha aumentado el valor de t con el fin de aumentar el espacio de trabajo al realizar transiciones entre planos convexos. En el nuevo prototipo, la cadera se ha dimensionado para conseguir un valor de $t = 150$ mm.

En cuanto a la articulación entre las patas y la cadera, se ha adoptado una

solución para dotar al robot de rigidez ante fuerzas perpendiculares al eje de las articulaciones. Consiste en realizar la articulación y sujeción mecánica utilizando 2 rodamientos y liberar a los ejes de los motores de esas tareas, los cuales únicamente se encargan de actuar el movimiento rotacional. En la figura 26, se muestra una vista seccionada de las nuevas articulaciones.

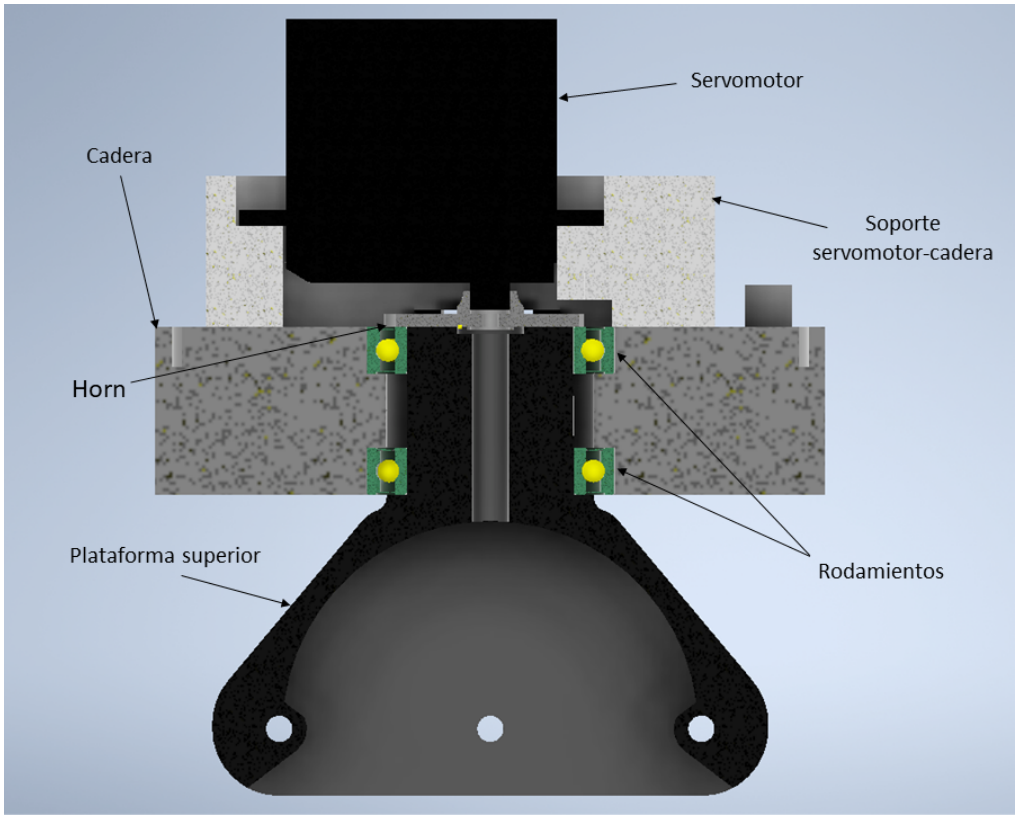


Figura 26: Articulación entre la cadera y la pata.

Se puede apreciar que los 2 rodamientos articulan la cadera y la pieza correspondiente a la plataforma del módulo paralelo superior de la pata. De esta forma, se logra dar una rigidez considerable a la unión rotacional.

Es el mismo *horn* del servo el que actúa como límite superior entre el rodamiento y la pieza. Se encuentra alojado en una muesca dimensionada en el extremo del cilindro de la plataforma superior y fijado respecto a ésta con tornillos.

El servomotor, que actúa el movimiento de la articulación, se encuentra fijo respecto a la cadera mediante un soporte que permite la unión. Cabe mencionar que para fijar el *horn* al servomotor ha sido necesario dejar un orificio dentro de la parte cilíndrica de la plataforma superior para poder introducir el tornillo que los fija.

3.2. Electrónica

En esta sección se expondrá toda la electrónica empleada en el nuevo prototipo. En el nuevo diseño se ha renovado completamente la electrónica de la que está compuesta el robot con el fin de simplificarlo, reduciendo el peso y tamaño del mismo.

Se empezará hablando de los actuadores empleados, para luego hablar del resto de componentes del robot.

3.2.1. Actuadores

El sistema de control del robot original consistía en un control desacoplado individual de cada articulación utilizando controladores PID. El lazo del sistema se cerraba con la señal de posición del actuador obtenida a partir del voltaje medido en un potenciómetro. En el caso de los actuadores lineales de los módulos paralelos, el potenciómetro se encontraba embebido en el mismo actuador. Sin embargo, en el caso de los actuadores rotacionales de las caderas, el potenciómetro iba montado en la misma articulación, lo que conllevaba un incremento en el volumen del robot.

Además del mayor tamaño y peso debido a los potenciómetros, este tipo de control conllevaba una complejidad que con el nuevo diseño se ha buscado eliminar. Es por ello que se han reemplazado todos los actuadores del prototipo original por actuadores que llevan el control integrado.

Para los 4 módulos paralelos que forman el robot, los cuales emplean en total 8 actuadores lineales, se han utilizado actuadores L12-50-210-12-I de Actuonix. Entre todas las posibles configuraciones ofrecidas por el fabricante, se ha optado por la versión de 12 V de alimentación, carrera de 50 mm y reductora de 210:1. Las especificaciones técnicas más importantes del actuador se recogen en la tabla 2 que se muestra a continuación:

Punto de potencia máxima	62 N a 3.2 mm/s
Punto de eficiencia máxima	36 N a 4.5 mm/s
Velocidad máxima	6.5 mm/s
Fuerza máxima	80 N
Fuerza estática máxima	200 N
Tensión máxima de alimentación	13.5 V
Corriente máxima	246 mA
Masa	40 g

Tabla 2: Especificaciones técnicas Actuonix L12-50-210-12-I.

La elección de los actuadores viene dada por la alta potencia que ofrecen en un formato compacto y ligero.

En cuanto al control, la versión escogida ofrece multitud de posibilidades: pro-

porcional a tensión de 0 a 5 V, proporcional a corriente de 4 a 20 mA (compatible con PLCs industriales), proporcional a la amplitud de pulsos y control con PWM (duty cycle). El control de los nuevos actuadores se va a realizar mediante longitud de pulsos, ya que los actuadores rotacionales de la cadera, los cuales se expondrán a continuación, precisan de este tipo de control.

Las articulaciones rotacionales que unen ambas patas a la cadera estaban actuadas por 2 motores controlados por sendos potenciómetros en el prototipo original. En el nuevo diseño se utilizan servomotores HPS-A700 de Futaba, los cuales llevan el lazo de control integrado. Las especificaciones del servomotor se pueden consultar en la tabla 3:

Velocidad	8.06 rad/s a 6.6 V
Par	6.669 Nm a 6.6 V
Tensión de alimentación	4.8 V - 8.4 V
Masa	82 g

Tabla 3: Especificaciones técnicas Futaba HPS-A700.

El control de los servomotores, así como de los actuadores lineales, se realiza modificando la amplitud de los pulsos de una señal de 5 V. En la figura 27 se muestra el funcionamiento del control de actuadores mediante amplitud de pulsos y una representación del estado resultante tanto en los actuadores lineales como en los servomotores.

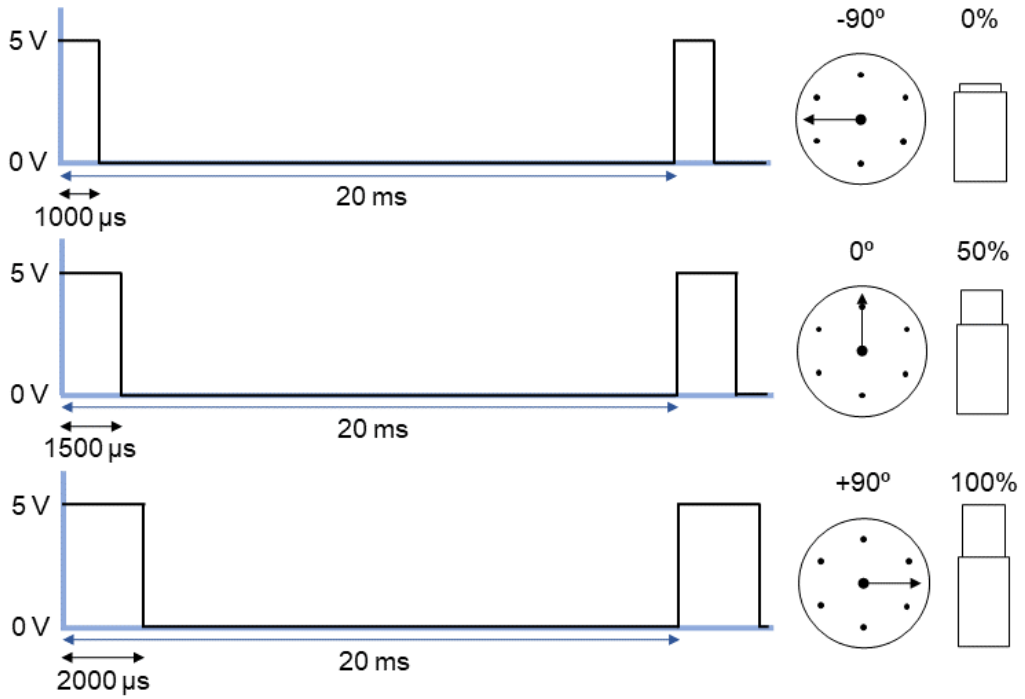


Figura 27: Control de los actuadores mediante amplitud de pulsos.

Los actuadores reciben un pulso cada 20 ms. La posición a la que se quiera

comandar el actuador vendrá dada por la amplitud del pulso que reciban. Cuando el pulso sea mínimo, normalmente correspondiente a 1000 μs, los actuadores estarán totalmente recogidos o en su ángulo mínimo. Cuando el pulso sea máximo, en general 2000 μs, los actuadores estarán totalmente estirados o en su posición angular máxima. Cuando el pulso que reciban esté entre el rango máximo y mínimo, la longitud o posición angular resultante será proporcional a la amplitud del pulso recibido.

En definitiva, esta nueva configuración de actuadores resulta en una disminución del peso y tamaño del prototipo, así como en una simplificación del control del robot debido a que los nuevos actuadores cuentan con un control integrado.

3.2.2. Componentes

En este apartado se presentarán los componentes utilizados para alimentar, controlar y comandar el robot.

Todo el sistema del robot se procesa en su computador. Anteriormente, el robot estaba controlado usando un microcontrolador como Arduino, el cual ofrece sencillez y un muy buen rendimiento y es muy recomendable para realizar el control de los actuadores, pero carece de las posibilidades que ofrece un ordenador al uso. En el nuevo prototipo, se ha optado por utilizar un ordenador en placa o SBC (*Single Board Computer*). Estos ordenadores ofrecen las mismas funcionalidades que un ordenador de sobremesa o portátil en un solo circuito. Generalmente, los diseños de los SBC se centran en un solo microprocesador junto con la memoria RAM integrada. Tanto el microprocesador, como la memoria y los puertos de entrada y salida se encuentran embebidos en la placa base, que a su vez sirve de soporte físico del computador.

En el prototipo presentado en este TFM se ha empleado una Raspberry Pi 4 como SBC. Las especificaciones más destacadas para el control del robot se recogen en la tabla 4, la cual se muestra a continuación:

Microprocesador	Broadcom BCM2711 (Quad core @ 1.5 GHz)
Memoria RAM	8 GB LPDDR4-3200
Conectividad	WiFi 802.11ac, Bluetooth 5.0
Puertos	40 pines GPIO, HDMI, USB 3.0, USB 2.0, Gigabit Ethernet
Tensión de alimentación	5 V

Tabla 4: Especificaciones técnicas Raspberry Pi 4.

Como sistema operativo se ha instalado Ubuntu Server 20.04, sobre el cual se ha programado el sistema de control usando ROS 2, el cual se explicará en detalle en el capítulo 4.

Para enviar las señales de control a los actuadores, se utiliza un controlador PWM PCA9685. El controlador se encarga de generar los pulsos descritos en el

apartado anterior necesarios para comandar a los acutadores a las posiciones deseadas. La información de los pulsos que ha de generar se envía utilizando el protocolo de comunicación I2C a través de los puertos GPIO de la Raspberry Pi hasta los puertos de entrada del controlador.

Sin embargo, el controlador funciona con señales de 3.3 V, mientras que los puertos GPIO de la Raspberry Pi 4 son de 5 V. Por lo tanto, es necesario emplear un convertidor bidireccional que convierta los 3.3 V a 5 V y viceversa.

Por último, es necesario un convertidor de tensión que convierta los 12 V con los que se alimentarán a los actuadores a los 5 V que alimentará a la Raspberry Pi 4. Esto se consigue mediante el convertidor DC-DC PSD-15ZG-R7VAI de Mean Well.

3.3. Prototipo construido

En esta sección se expone el prototipo construido siguiendo el diseño mostrado a lo largo del capítulo.

En primer lugar, en la figura 28 se muestra el robot HyReCRo en su posición estándar, inicio o *home*, donde todos sus actuadores se encuentran en su amplitud media, adoptando una configuración articular $[r_{1A}, l_{1B}, r_{2A}, l_{2B}, l_{1A}, r_{1B}, l_{2A}, r_{2B}, \theta_A, \theta_B] = [125, 125, 125, 125, 125, 125, 125, 125, 0, 0]$ (valores en milímetros y radianes).

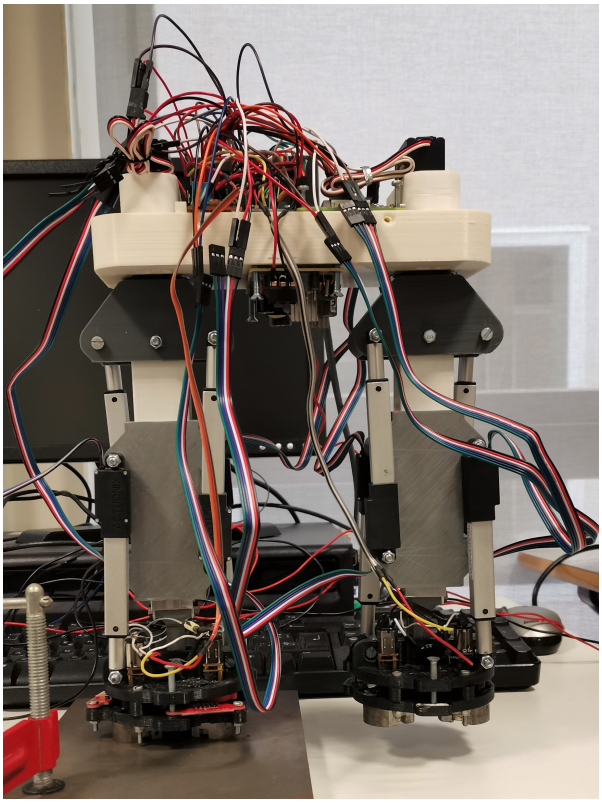


Figura 28: Nuevo prototipo del robot HyReCRo.

Tal como se muestra en la figura 12 de la sección 2.3, el robot precisa de 3 movimientos básicos para explorar estrucutras tridimensionales. A continuación se mostrará el prototipo construido adoptando cada una de las posturas de exploración.

En la figura 29 se puede observar al robot adoptando una configuración articular $[r_{1A}, l_{1B}, r_{2A}, l_{2B}, l_{1A}, r_{1B}, l_{2A}, r_{2B}, \theta_A, \theta_B] = [112, 115, 141, 145, 140, 117, 111, 0, 0]$, la cual le permite realizar un desplazamiento longitudinal a través de una superficie.

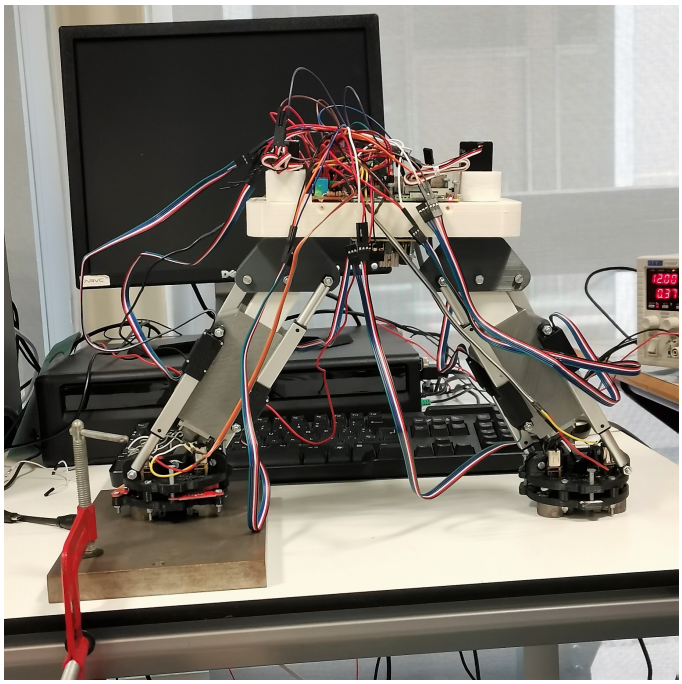


Figura 29: Desplazamiento longitudinal a través de un plano.

Para realizar una transición exterior entre superficies, el robot ha de adoptar una posición articular semejante a $[r_{1A}, l_{1B}, r_{2A}, l_{2B}, l_{1A}, r_{1B}, l_{2A}, r_{2B}, \theta_A, \theta_B] = [146, 152, 126, 123, 115, 103, 122, 128, 0, 0]$. En la figura 30 se muestra el prototipo en la configuración descrita.

El último movimiento necesario para la exploración de estructuras tridimensionales es la transición interior entre planos. En la figura 31, el robot adopta una posición que permite realizar la transición: $[r_{1A}, l_{1B}, r_{2A}, l_{2B}, l_{1A}, r_{1B}, l_{2A}, r_{2B}, \theta_A, \theta_B] = [144, 143, 146, 144, 111, 113, 110, 111, 0, 0]$.

Finalmente, cabe mencionar que el control de los imanes de las garras descrito en la sección 2.3 no se ha podido llevar a cabo utilizando los pines digitales de salida de la Raspberry Pi. Esto se debe a que la corriente proporcionada por los pines no es lo suficientemente elevada como para activar los relés del circuito. Como solución provisional, se ha optado por realizar el control de los imanes utilizando un controlador Arduino, tal como se hacía en el prototipo original.

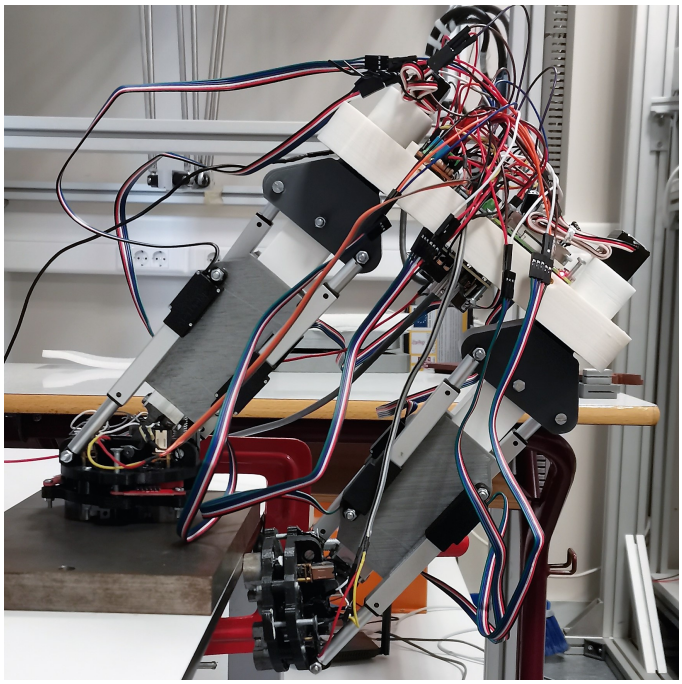


Figura 30: Transición exterior entre planos.

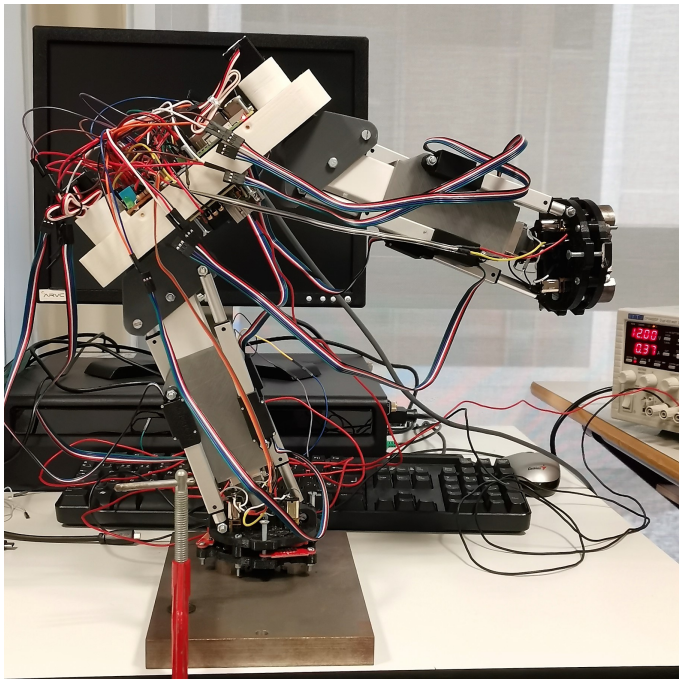


Figura 31: Transición interior entre planos.

4. Programación y control del prototipo

Este capítulo versará sobre la programación del sistema de control del nuevo prototipo del robot HyReCRo. Además, se mostrará una interfaz que permite comandar acciones y controlar el robot.

Todo el *software* del robot se ha desarrollado utilizando el *framework* ROS2, por lo que a continuación se realizará una pequeña introdución del mismo.

4.1. ROS2

ROS2 (*Robot Operating System 2*), es un "sistema operativo" para robots que deriva de su predecesor ROS.

ROS nace en 2007, con el nombre de *switchyard*, para controlar el robot STAIR (Quigley et al., 2007), desarrollado por el Laboratorio de Inteligencia Artificial de la Universdidad de Stanford. Posteriormente, en 2008, los creadores de *switchyard*, comienzan a trabajar en Willow Garage, desarrollando el robot PR2, cuyo sistema de control ya llevaría el nombre de ROS.

La filosofía de ROS consiste en hacer un "Linux para robots", citando a los desarrolladores originales de *switchyard*. Al igual que Linux, ROS fue diseñado para ser *open source*, siendo su código totalmente público. Los usuarios son capaces de elegir qué herramientas y librerías interaccionan con el núcleo de ROS para adaptarse completamente a las necesidades de una aplicación y robot concretos. A efectos prácticos, el núcleo de ROS es únicamente la estructura general de nodos y protocolo de mensajes sobre los que se construye el *software*.

A pesar de su nombre, ROS no es un sistema operativo al uso, sino una colección de herramientas para el desarrollo de *software* de un robot. ROS debe ejecutarse encima de otro sistema operativo, concretamente Linux de forma original.

Los procesos de ROS están basados en nodos, representados como una estructura de grafos, los cuales están conectados entre si a través de enlaces llamados *topics*. Los nodos se comunican con mensajes a través de los *topics* y todos se encuentran controlados por un nodo maestro (ROS Master). A pesar de tratarse de un sistema de maestro-esclavo, se considera que tiene una arquitectura descentralizada puesto que el maestro únicamente establece, registra y controla la comunicación entre nodos esclavos, no actúa como intermediario de mensajes.

Los nodos de ROS se pueden programar tanto en C++ como en Python. En sistemas con necesidad de comunicar varios computadores, ROS cuenta con formatos de serialización, protocolos de transporte y mecanismos de descubrimiento propios.

En 2017 aparece la primera distribución oficial de ROS2. Brian Gerkey, creador de ROS, declaró que el motivo del desarrollo de una nueva versión de ROS es

satisfacer necesidades que no existían al principio del proyecto. Algunas de las necesidades que ROS2 cubre y sus razones son:

- Sistemas de varios robots: ROS no dispone de un enfoque estándar para este tipo de sistemas.
- Sistemas embebidos: los microcontroladores o computadores embebidos dependen de un controlador con ROS.
- Sistemas en tiempo real: originalmente, ROS tenía requisitos de control en tiempo real.
- Redes no ideales: se busca que los sistemas se comporten de forma correcta a pesar de una mala conectividad.

A diferencia de ROS, solamente disponible en Linux (con versiones desarrolladas por la comunidad para Mac OS X), ROS2 es compatible de forma oficial con Linux, Windows y Mac OS X. Además, actualiza las versiones de C++ y Python a las que se dirige: de C++03 a C++11 y de Python 2 a Python 3.

En ROS2 se abandona la arquitectura de maestro-esclavo y ya no es necesario crear el nodo maestro (ROS Master) en un sistema. También se permite crear más de un nodo por proceso, lo cual no era posible con ROS. En el caso de control en tiempo real, ROS2 permite escribir nodos que se ejecuten en un RTOS (*Real Time Operating System*) de forma nativa.

Finalmente, se remodelan las interfaces y estructuras de comunicación entre nodos. En ROS2 se añaden 2 tipos de comunicación entre nodos aparte del original. A continuación se explicarán conceptos básicos sobre los 3 tipos de comunicación entre nodos disponibles en ROS2.

En primer lugar se encuentra la comunicación a través de *topics*. Éste es el método original de comunicación entre nodos de ROS. Consiste en un sistema de publicador y suscriptor. Cuando el publicador publica un mensaje en el *topic*, los suscriptores de ese *topic* reciben el mensaje. En la figura 32 se puede ver el funcionamiento de este tipo de comunicación. Un *topic* puede tener múltiples nodos que publiquen y que estén suscritos.

El uso típico de los *topics* es el de enviar información que está en constante actualización. Ejemplos apropiados son las medidas tomadas por un sensor, las imágenes tomadas por una cámara o los comandos de control dados por un *joystick*.

El segundo método de comunicación presentado es a través de servicios. En la figura 33 se muestra el comportamiento de los nodos durante la comunicación. El nodo cliente envía un mensaje de petición a través del servicio. La petición llega al servidor, el cual lo procesa y devuelve un mensaje de respuesta al cliente a través del servicio. Un servicio puede tener varios clientes pero un solo servidor.

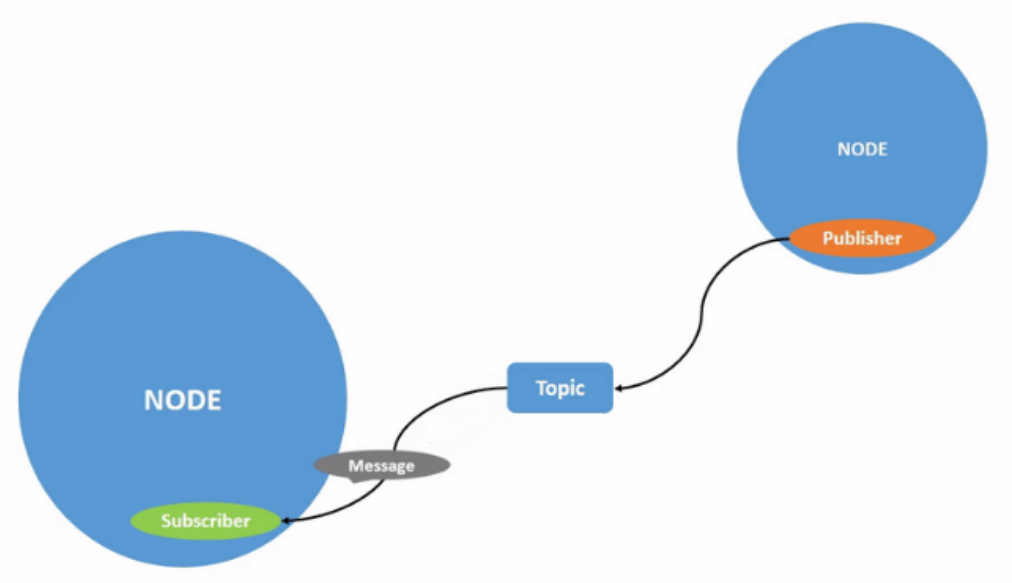


Figura 32: Comuncación a través de topics. Fuente: docs.ros.org

Los servicios se suelen utilizar para tareas que se realizan de forma ocasional con terminación rápida, como es el caso del cálculo de la cinemática inversa.

Por último, se presenta el modelo de comunicación por acciones. Este modelo de comunicación es una combinación de un *topic* y 2 servicios. Al igual que con los servicios, las acciones tienen un nodo cliente y un nodo servidor. Primero, el cliente envía un mensaje de petición a través del servicio de objetivo, informando de la acción a llevar a cabo por el servidor. El servidor responde a través del mismo servicio dando a entender que ha recibido el objetivo y ha empezado a ejecutarlo. Cuando se ha empezado a procesar la acción, el cliente inicializa el servicio de resultado, por el que luego el servidor enviará el resultado de la acción. Mientras se ejecuta la acción, el servidor está constantemente publicando mensajes de información retroalimentativa a través de un *topic* dedicado para ello. Cuando la acción se ha completado, el servidor envía un mensaje con el resultado de la acción a través del servicio de resultado. En la figura 34 se representa el modelo de comunicación a través de acciones.

Su funcionalidad es similar a la de los servicios, a excepción de que las acciones pueden ser canceladas. Su uso está pensado para tareas esporádicas con un tiempo de ejecución alto y que necesiten información retroalimentativa.

Una vez introducidos los conceptos más importantes de ROS2, se explicará el sistema desarrollado para el control del prototipo del robot HyReCRo.

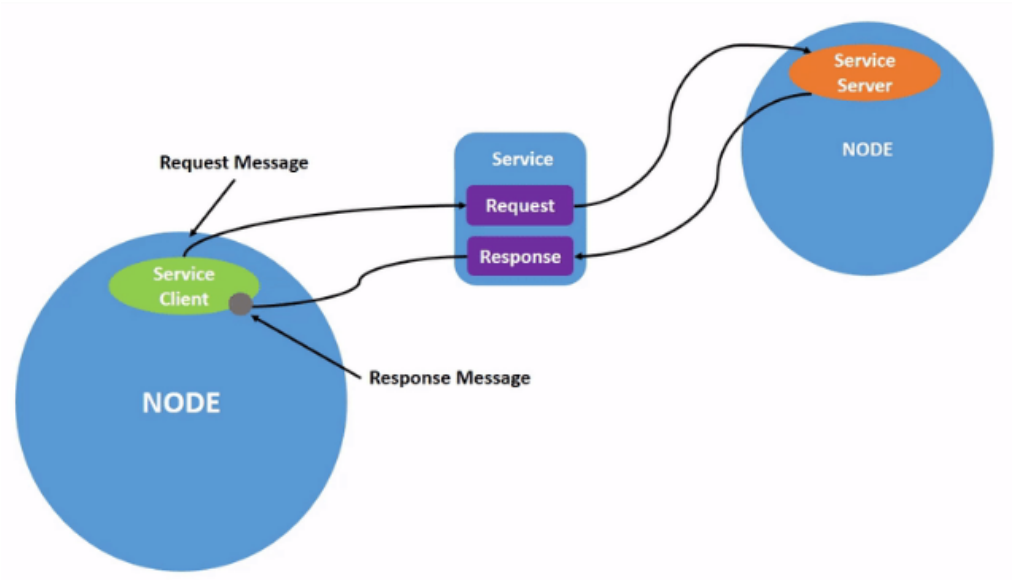


Figura 33: Comuncación a través de servicios. Fuente: docs.ros.org

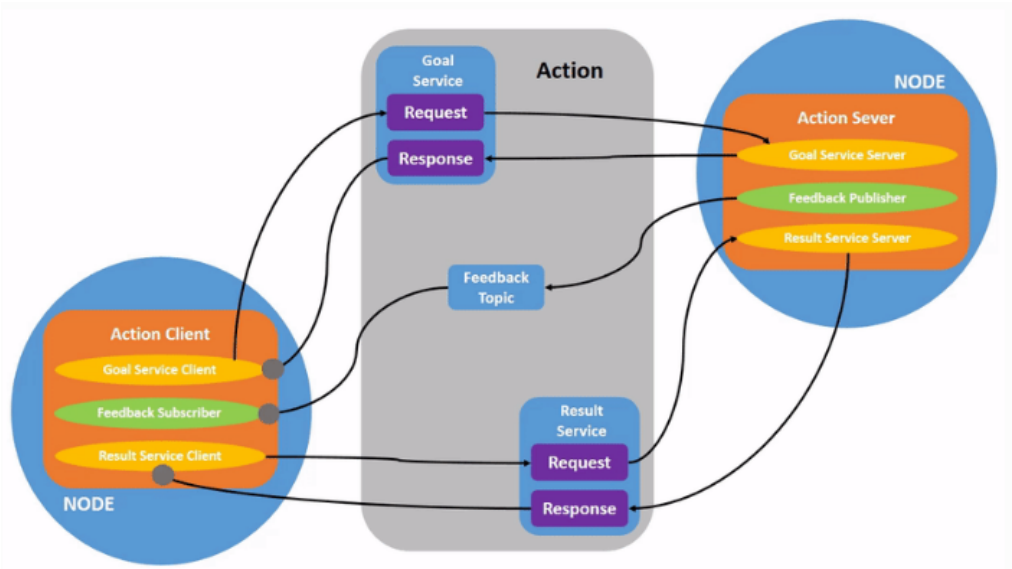


Figura 34: Comuncación a través de acciones. Fuente: docs.ros.org

4.2. Sistema de control

El sistema desarrollado en ROS2 está compuesto de 3 nodos, cada uno con una función específica. ROS permite libertad completa a la hora de organizar el sistema, pudiendo dividir una misma tarea en varios nodos. Sin embargo, para facilitar la comprensión a costa de una mayor complejidad de programación, se ha optado por limitar el sistema a 3 nodos, separados en acciones de entrada, procesamiento y salida. En la figura 35, mostrada a continuación, se puede observar una representacion gráfica en forma de grafo del sistema desarrollado.

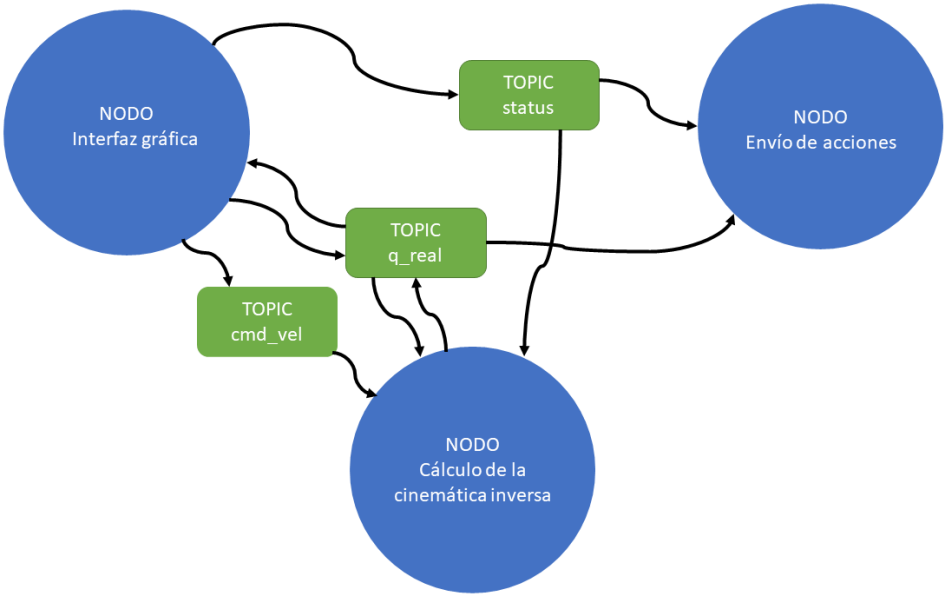


Figura 35: Representación gráfica del sistema desarrollado.

El método de comunicación entre los 3 nodos es a través de *topics*, a los cuales se suscriben y publican. En el sistema se utilizan 3 *topics* diferentes.

- `q_real`: A través de este *topic* se comparte la configuración articular del robot. Se trata de un vector formado por los valores de los 10 actuadores de los que se compone el robot, de la forma $[r_{1A}, l_{1B}, r_{2A}, l_{2B}, l_{1A}, r_{1B}, l_{2A}, r_{2B}, \theta_A, \theta_B]$.
- `status`: Se trata de un vector de 3 variables booleanas que indica el estado del robot. Está formado por 2 variables que indican si cada una de las garras está pegada o no y otra que indica qué pie está actuando como sistema de referencia de la base del robot.
- `cmd_vel`: Recoge los incrementos de posición y orientación del efector final comandados por el usuario a través de la interfaz gráfica.

A lo largo de esta sección se explicará el funcionamiento de los 3 nodos de los que está compuesto el sistema.

4.2.1. Interfaz gráfica de control

En primer lugar, se presenta el nodo que sirve como interfaz visual del usuario.

La idea original consistía en realizar una especie de *teach pendant*, similar al cuadro de mando del que se valen los operarios de robots industriales, que proporcione toda la información instantánea del estado del robot y desde donde se puedan comandar sus movimientos. La solución propuesta se puede observar en las figuras 36 y 37.

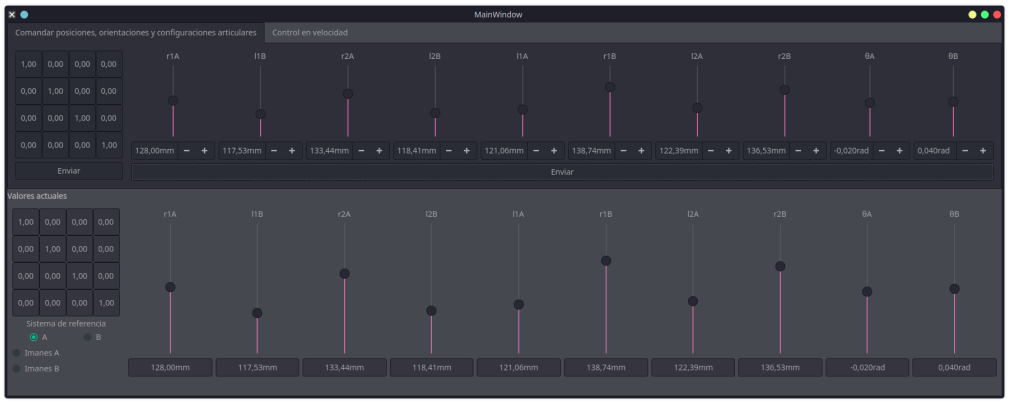


Figura 36: Interfaz gráfica de control (control articular).

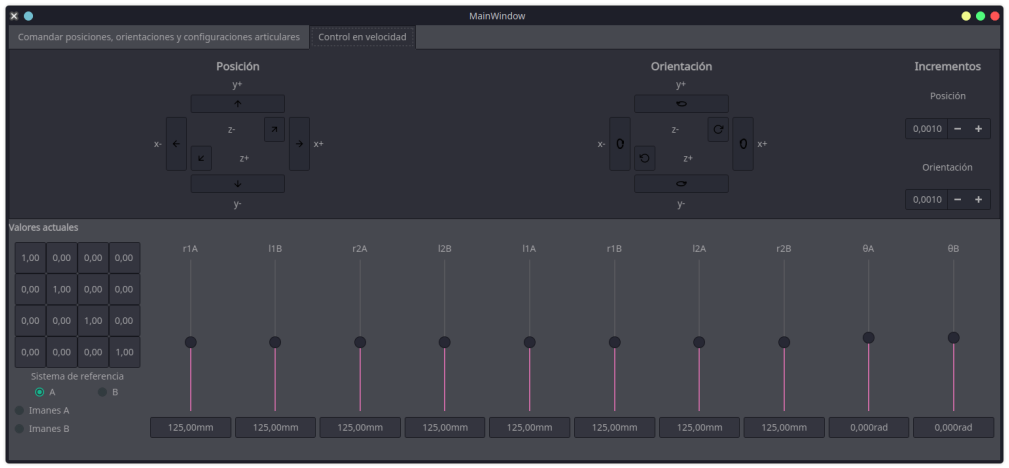


Figura 37: Interfaz gráfica de control (control por incrementos).

La interfaz programada permite realizar un control del robot mediante 2 modos, seleccionados a partir de las pestañas que modifican la mitad superior de la intefaz. En primer lugar, se puede realizar un control independiente de la posición de cada actuador (figura 36), correspondiente al modo de control en cinemática directa, donde se especifican las coordenadas articulares actuadas. La otra opción es controlar la traslación y el giro del efector final mediante un control por pequeños incrementos de posición (figura 37), correspondiente al modo de control en cinemática inversa, donde se especifica la pose de la garra libre. Los incrementos comandados en posición y orientación pueden ser modificados por el usuario. Además, en la parte inferior de la ventana, se muestra en todo momento los valores de

los actuadores del robot. Finalmente, en la parte inferior izquierda de la interfaz, se realiza el control del estado del robot, pudiendo modificar el estado de pegado de cada garra y el sistema de referencia que se utiliza como base del robot para los cálculos.

El código de la interfaz, programada en Python como nodo del sistema, se puede consultar en el anexo A.1.

La interfaz gráfica ha sido creada utilizando la librería Qt5. Se trata de un *framework* muy utilizado, siendo la base de importantes aplicaciones como VLC, Oracle VM VirtualBox o KDE entre muchas otras. Para el diseño de la disposición de los elementos en la ventana se ha utilizado el programa Qt Creator.

Qt Creator genera un archivo, el cual se cargará en el programa principal (nodo de ROS2), que funciona a modo de módulo de Python llamado `MainWindow.py`. El módulo sirve como plantilla de la interfaz, pudiendo programar sobre él las funcionalidades que requiera la aplicación, y se encuentra disponible para el lector en A.1.1. Sobre el código original generado, se han añadido las líneas 16-67, correspondientes a 2 nuevas clases de objetos derivados de clases propias de la librería, creados para satisfacer la necesidad de trabajar con decimales, lo cual no era posible con ciertos objetos nativos de Qt. Además, se han agregado las líneas 1525-1629, que sirven para realizar conexiones de *slots* y señales internas entre variables y objetos de la interfaz.

El programa principal de la interfaz, que actúa como nodo de ROS2, está compuesto por 3 clases que hacen posible su funcionamiento. La función `main` del programa inicia todos los procesos necesarios y crea un objeto de la clase `MainWindow`, la cual se encuentra definida a lo largo de las líneas 53-110.

La clase hereda las características de las clases `QMainWindow`, nativa de Qt, y `Ui_MainWindow`, importada desde el módulo `MainWindow.py` generado por Qt Creator. El método `__init__` es llamado cuando se crea una nueva instancia de la clase. En él, se crea el nodo de ROS2 como un objeto de la clase `Gui` y se ejecuta un proceso multihilo como instancia de la clase `ROS_spinner`. Ambas clases se explicarán a continuación.

Seguidamente, se definen varios métodos de la clase `MainWindow`, los cuales se ejecutan cuando ocurre cierto evento en la interfaz o en el sistema. Los métodos `new_status` y `new_qdes` se ejecutan cuando el usuario pulsa sobre algún botón que cambia el estado del robot (pie fijo y sistema de referencia) o comanda una nueva configuración articular, respectivamente. Así mismo, el método `new_vel` es activado cuando se pulsa un botón en la pestaña de control mediante incrementos de posición. Cuando cualquiera de los 3 métodos comentados es activado, publica el mensaje con los nuevos valores en su respectivo *topic*. Por otro lado, el método `timer_callback` se activa cuando el temporizador de la clase `ROS_spinner` envía una señal, y se encarga de actualizar los valores articulares del robot en ese instante.

Cuando se crea una instancia de la clase `Gui`, definida en las líneas 16-28, se

inicializan los publicadores y suscriptores de los que está compuesto el nodo. La clase tiene únicamente un método, el cual se ejecuta cada vez que el suscriptor del *topic* `q_real` recibe un nuevo mensaje. El método pasa el contenido del mensaje a una variable interna y cambia el valor de una variable booleana a `True`, que se utiliza para indicar que un nuevo mensaje se ha recibido.

Finalmente, la clase `ROS_spinner` sirve para mantener el nodo de ROS2 activo y permitir la comunicación entre clases. La instancia de la clase se ejecuta en un hilo separado del procesador y, en su inicialización, crea un temporizador de 100 ms el cual ejecuta el método `ROS_spin` en cada ciclo. Los nodos de ROS necesitan "girar" (en inglés, *spin*) para mantenerse activos y comprobar nuevos eventos en los *topics* a los que están suscritos. Es por ello que el método `ROS_spin` "gira" el nodo cada 100 ms. Además, cuando se recibe un nuevo mensaje del *topic* `q_real`, la función se encarga de transmitir la información desde la clase `Gui` hasta la clase `MainWindow`.

4.2.2. Cálculo de la cinemática inversa

Cuando el sistema recibe un comando de control por incrementos de posición a través de la interfaz gráfica, es necesario calcular los incrementos en las coordenadas articulares que consigan la traslación y rotación comandadas.

Para ello, se ha creado un nodo que se encarga de realizar el cálculo de los incrementos articulares. Debido a la complejidad proveniente de la necesidad de llevar a cabo operaciones algebraicas matriciales, se ha optado por programar el nodo en C++, debido al menor tiempo de cómputo que ofrece este lenguaje frente a Python. Además, se ha utilizado la librería de álgebra lineal y operaciones matriciales Eigen3 para C++.

En el `main` del programa (anexo A.2) únicamente se inicializa el nodo de ROS2. En el constructor del nodo se crean los suscriptores y publicadores del mismo. Tal como se aprecia en la figura 35, el nodo está compuesto de los suscriptores de los *topics* `status`, `cmd_vel` y `q_real`; y el publicador de `q_real`.

La clase del nodo únicamente tiene 3 funciones privadas, una para cada suscriptor, que se llamarán cuando su respectivo *topic* reciba un nuevo mensaje. En el caso de los *topics* `status` y `q_real`, la función asociada a cada suscriptor (líneas 108-111 y 112-124 respectivamente) únicamente se encarga de leer los mensajes recibidos y guardar sus valores en variables.

La función asociada al suscriptor de `cmd_vel` (líneas 54-107) se encarga de realizar el cálculo de la cinemática inversa cuando recibe un nuevo vector de incrementos de posición y orientación comandado. La función, y las funciones a las que llama internamente, se explicarán a lo largo de los siguientes párrafos.

En primer lugar, se llama a la función `real2int` (definida en las líneas 177-196) para obtener los valores de las variables articulares intermedias del equivalente serie

del robot a partir de las coordenadas articulares reales. Ésta, calcula la cinemática directa de cada módulo paralelo mediante la función `fk_parallel_module` (definida a lo largo de las líneas 141-170), que la resueve haciendo uso del método de Newton. A partir de una solución inicial cercana a la final, se calculan incrementos de forma iterativa a partir de las ecuaciones que definen la cinemática directa de los módulos paralelos (ecuaciones (16) y (17) de la sección 2.1.1).

A continuación, se calcula la matriz Jacobiana en coordenadas globales de la base del robot utilizando la función `jacob`, definida entre las líneas 221-312. La matriz Jacobiana expresa la relación entre las velocidades articulares del robot ($\dot{\mathbf{q}}$) y las velocidades de su efector final ($\dot{\mathbf{X}}$) de la forma:

$$\dot{\mathbf{X}} = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}} \tag{45}$$

Cuando variaciones en la posición articular son muy pequeñas, esta relación también se cumple:

$$\Delta\mathbf{X} = \mathbf{J}(\mathbf{q})\,\Delta\mathbf{q} \tag{46}$$

Seguidamente, utilizando la librería Eigen3, se realiza la descomposición en valores singulares (SVD) de la matriz Jacobiana. Se comprueba si algún valor singular es nulo, en cuyo caso el robot se encontrará en una singularidad. Para salir de ella, se utilizará una versión amortiguada de la Jacobiana, sumando una matriz identidad multiplicada por un valor muy pequeño a la Jacobiana original:

$$\mathbf{J}_d = \mathbf{J} + 0.01\,\mathbf{I} \tag{47}$$

Cabe mencionar que esta matriz amortiguada está introduciendo un error en la cinemática del robot. Sin embargo, este error introducido es despreciable puesto que únicamente se utiliza de forma puntual cuando el robot está dentro de la singularidad.

Los incrementos articulares que producirán los incrementos de la pose comandados se calculan despejando el término $\Delta\mathbf{q}$ de la ecuación (46):

$$\Delta\mathbf{q} = \mathbf{J}(\mathbf{q})^{-1}\,\Delta\mathbf{X} \tag{48}$$

No obstante, cuando la matriz Jacobiana no es cuadrada, como es el caso del equivalente serie del robot HyReCRo, no es invertible. Esto se produce debido a la redundancia del robot, resultando en una matriz Jacobiana de dimensiones 6×8. Para poder utilizar la relación de la ecuación (48), es necesario obtener la inversa de la Jacobiana utilizando una generalización. La pseudoinversa de Moore-Penrose permite obtener una solución al sistema lineal de ecuaciones con norma mínima. La operación a utilizar para el cálculo de la pseudoinversa depende de

las dimensiones de la matriz. Para la matriz Jacobiana del robot, al tener filas linealmente dependientes, se utiliza la siguente ecuación:

$$\mathbf{A}^{\dagger} = \mathbf{A}^{T} \left(\mathbf{A}\, \mathbf{A}^{T}\right)^{-1} \tag{49}$$

En el código se implementa en la función `pinv` a lo largo de las líneas 314-319.

A continuación, se calculan los incrementos articulares utilizando la ecuación (48) y se suma su resultado a los valores articulares iniciales. Las variables articulares resultantes, que son las intermedias del equivalente serie del robot, se vuelven a transformar en coordenadas articulares reales por medio de la función `int2real`. La función, definida en las líneas 198-219, llama internamente a la función `ik_parallel_module` (líneas 172-175), que resuelve la cinemática inversa de cada módulo paralelo siguiendo las ecuaciones (16) y (17) de la sección 2.1.1.

Finalmente, se comprueban los límites articulares del robot mediante la función `comprobar_limites` y se publica en el *topic* `q_real` el nuevo vector de posiciones articulares que conseguirán cumplir el incremento de posición u orientación comandado por el usuario.

4.2.3. Envío de las acciones

Finalmente, se expone el nodo encargado de enviar las acciones a los actuadores y circuitos. El código está disponible en el anexo A.3.

Para el control de los actuadores, se hace uso de los módulos de Adafruit para Python ServoKit y PCA9685. Para utilizar los pines GPIO de la Raspberry Pi de forma individual, se utiliza el módulo RPi.GPIO para Python.

En primer lugar, entre las líneas 19-50, se declaran las variables globales de inicialización de los módulos de control de la electrónica. Seguidamente, se define la clase del nodo, de la que se creará una instancia en el `main` del programa. En la inicialización de la clase, `__init__`, se crean los 2 suscriptores asociados a los *topics* `q_real` y `status`, que leen los valores articulares del robot y su estado, respectivamente.

A partir de la línea 71, se define el método que se ejecutará cuando se recibe un mensaje con una nueva configuración articular. A partir de las curvas de calibración obtenidas previamente para cada actuador, se calcula, a partir del valor de la distancia recibido, el porcentaje de la longitud de pulso (figura 27 de la sección 3.2.1) que conseguirá la longitud deseada. Seguidamente, se envían los porcentajes calculados.

Entre las líneas 96-107 se define la función ejecutada cada vez que se recibe un nuevo estado del robot. La función comprueba si ha ocurrido un cambio en el valor del pegado de cada garra. Si se ha comandado un cambio, se envía un pulso de 500

ms a través de los pines GPIO.

5. Conclusiones y trabajos futuros

En este capítulo se extraerán las conclusiones derivadas del trabajo desarrollado a lo largo del TFM y se propondrán futuras líneas de trabajo, tanto a largo como a corto plazo.

5.1. Conclusiones

El principal objetivo abarcado en este TFM ha sido la construcción y programación de un prototipo del robot trepador bípedo HyReCRo. Se ha abordado el objetivo dividiéndolo en 2 tareas, las cuales se desarrollan a lo largo de los capítulos 3 y 4.

En el capítulo 3 se ha presentado un nuevo prototipo del robot HyReCRo. En primer lugar, ha presentado el diseño mecánico de la estructura del nuevo prototipo. Seguidamente, se ha expuesto la nueva electrónica de la que está compuesto el robot. Finalmente, se muestra el prototipo construido. Se ha conseguido construir satisfactoriamente un prototipo más rígido (tanto a nivel torsional alrededor del eje de la pata, como en la unión entre cada pata y la cadera), con un centro de gravedad más bajo al renovar y modificar la disposición de la electrónica y un control más simples al emplear actuadores con control integrado.

En el cuarto capítulo se ha mostrado el sistema de control del robot, creado usando ROS2. Primeramente, se ha realizado una pequeña introducción a ROS2, exponiendo su historia y conceptos básicos, para posteriormente mostrar el sistema de control programado. El sistema se compone de 3 nodos, dedicados a la interfaz de control de usuario, al cálculo de la cinemática inversa y al envío de las acciones a los actuadores.

5.2. Trabajos futuros

En esta sección final, se proponen líneas de trabajo derivadas del desarrollo realizado a lo largo de este TFM.

En primer lugar, es necesario solucionar los problemas directamente relacionados con el trabajo realizado en el TFM. Por un lado, se debe solucionar el problema del control de los imanes descrito en la sección 3.3. Para ello, se debe aumentar la corriente proporcionada por los pines de la Raspberry Pi. Una posible solución consiste en crear un circuito que aumente la corriente empleando MOSFETs o amplificadores operacionales. Por otro lado, es necesario realizar una precisa calibración de los actuadores que componen el robot. Aunque la calibración actual de los actuadores ya permite realizar un control bastante preciso, es posible hacerlo con más precisión. De esa forma, las longitudes y ángulos comandados a los actuadores corresponderán con los reales.

También a corto plazo, se deben añadir nuevas funcionalidades a la interfaz de control expuesta en la sección 4.2.1. Nuevas características interesantes para su incorporación son la habilidad de guardar configuraciones articulares y poses concretas, representar visualmente el estado del robot y generar trayectorias.

Otra línea de trabajo, relacionada con el control por incrementos de posición explicado en la sección 4.2.2, consiste realizar un estudio y un mapa de las barreras internas que existen al resolver la cinemática inversa del robot (Peidró et al., 2018). Con ello, se podría optimizar el control explicado y futuras aplicaciones de *path planning*, evitando acercarse a las barreras internas estudiadas.

A largo plazo, es conveniente automatizar completamente los movimientos del robot, evitando así depender de un operador que realice el control en emplazamientos peligrosos o de difícil acceso o visibilidad. Con la implementación de los desarrollos realizados en el Trabajo de Fin de Grado (Fabregat, 2021) en el nuevo prototipo, es posible automatizar el pegado de las garras del robot mediante un control cinemático utilizando 3 sensores de distancia montados en las garras.

Finalmente, se propone añadir un segundo computador (SBC) a bordo del robot con el fin de procesar tareas complementarias al control directo del robot. De esta forma, se podrían delegar y repartir entre computadores tareas de visión por computador, localización y *mapping*, entre otras.

Referencias

Aracil, R., Saltaren, R., and Reinoso, O. (2006). A climbing parallel robot: a robot to climb along tubular and metallic structures. *IEEE Robotics Automation Magazine*, 13(1):16–22.

Baghani, A., Ahmadabadi, M., and Harati, A. (2005). Kinematics modeling of a wheel-based pole climbing robot (UT-PCR). In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation.* IEEE.

Balaguer, C., Giménez, A., Pastor, J. M., Padrón, V. M., and Abderrahim, M. (2000). A climbing autonomous robot for inspection applications in 3d complex environments. *Robotica*, 18(3):287–297.

Bennett, D. and Hollerbach, J. (1991). Autonomous calibration of single-loop closed kinematic chains formed by manipulators with passive endpoint constraints. *IEEE Transactions on Robotics and Automation*, 7(5):597–606.

Fabregat, M. (2021). Calibración y control cinemático de la adhesión de un robot trepador. Trabajo de Fin de Grado. Universidad Miguel Hernández de Elche.

Figliolini, G., Rea, P., and Conte, M. (2010). Mechanical design of a novel biped climbing and walking robot. In Parenti Castelli, V. and Schiehlen, W., editors, *ROMANSY 18 Robot Design, Dynamics and Control*, pages 199–206, Vienna. Springer Vienna.

Guan, Y., Jiang, L., Zhu, H., Zhou, X., Cai, C., Wu, W., Li, Z., Zhang, H., and Zhang, X. (2011). Climbot: A modular bio-inspired biped climbing robot. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1473–1478.

Hernando, M., Brunete, A., and Gambao, E. (2019). Romerin: A modular climber robot for infrastructure inspection. *IFAC-PapersOnLine*, 52(15):424–429. 8th IFAC Symposium on Mechatronic Systems MECHATRONICS 2019.

Mampel, J., Gerlach, K., Schilling, C., and Witte, H. (2009). A modular robot climbing on pipe-like structures. In *2009 4th International Conference on Autonomous Robots and Agents*, pages 87–91.

Peidró, A., Gil, A., Marín, J. M., Berenguer, Y., Payá, L., and Reinoso, O. (2015a). Monte-carlo workspace calculation of a serial-parallel biped robot. In *Robot 2015: Second Iberian Robotics Conference*, pages 157–169. Springer.

Peidró, A., Gil, A., Marín, J. M., Berenguer, Y., and Reinoso, O. (2015b). Kinematic analysis and simulation of a hybrid biped climbing robot. In *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 2, pages 24–34. IEEE.

Peidro, A., Gil, A., Marín, J., Berenguer, Y., and Reinoso, O. (2016). *Kinematics, Simulation, and Analysis of the Planar and Symmetric Postures of a Serial-Parallel Climbing Robot*, pages 115–135.

Peidró, A., Tavakoli, M., Marín, J. M., and Reinoso, Ó. (2019). Design of compact switchable magnetic grippers for the HyReCRo structure-climbing robot. *Mechatronics*, 59:199–212.

Peidró, A., Óscar Reinoso, Gil, A., Marín, J. M., and Payá, L. (2018). A method based on the vanishing of self-motion manifolds to determine the collision-free workspace of redundant robots. *Mechanism and Machine Theory*, 128:84–109.

Quigley, M., Berger, E., and Ng, A. (2007). Stair: Hardware and software architecture.

Schmidt, D. and Berns, K. (2013). Climbing robots for maintenance and inspections of vertical structures—a survey of design aspects and technologies. *Robotics and Autonomous Systems*, 61(12):1288–1305.

Shvalb, N., Moshe, B. B., and Medina, O. (2013). A real-time motion planning algorithm for a hyper-redundant set of mechanisms. *Robotica*, 31(8):1327–1335.

Tâche, F., Fischer, W., Caprari, G., Siegwart, R., Moser, R., and Mondada, F. (2009). Magnebike: A magnetic wheeled robot with high mobility for inspecting complex-shaped structures. *Journal of Field Robotics*, 26(5):453–476.

Tavakoli, M., Marques, L., and de Almeida, A. T. (2011). 3DCLIMBER: Climbing and manipulation over 3D structures. *Mechatronics*, 21(1):48–62.

Tavakoli, M., Viegas, C., Marques, L., Pires, J. N., and de Almeida, A. (2013). Omniclimber-ii: An omnidirectional climbing robot with high maneuverability and flexibility to adapt to non-flat surfaces. pages 1349–1354.

Tavakoli, M., Zakerzadeh, M., Vossoughi, G., and Bagheri, S. (2005). A hybrid pole climbing and manipulating robot with minimum dofs for construction and service applications. *Industrial Robot-an International Journal - IND ROBOT*, 32:171–178.

Viegas, C. and Tavakoli, M. (2014). A single DOF arm for transition of climbing robots between perpendicular planes. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE.

Yoon, Y. and Rus, D. (2007). Shady3d: A robot that climbs 3d trusses. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4071–4076.

A. Códigos

A.1. Nodo de la interfaz gráfica de control

```
import rclpy
from rclpy.node import Node
import hyrecro_interfaces.msg
from std_msgs.msg import String
from geometry_msgs.msg import Twist

import numpy as np
import math

# PyQt5
import sys
from PyQt5 import QtWidgets, uic, QtCore
from gui_hyrecro.MainWindow import Ui_MainWindow


class Gui(Node):

    def __init__(self):
        super().__init__('GUI_HyReCRo')
        self.status_pub = self.create_publisher(hyrecro_interfaces.msg.Status, 'status', 10)
        self.qreal_pub = self.create_publisher(hyrecro_interfaces.msg.QReal, 'q_real', 10)
        self.vel_pub = self.create_publisher(Twist, 'cmd_vel', 10)
        self.qreal_sub = self.create_subscription(hyrecro_interfaces.msg.QReal, 'q_real', self.qreal_callback, 10)

    def qreal_callback(self, msg):
        self.new_qreal = msg
        self.new_qreal_flag = True
        # print("nueva q_real")


class ROS_spinner(QtCore.QThread):
    def ROS_spin(self):
        # print("Spinning node once")
        rclpy.spin_once(self.node, timeout_sec=0)
        if self.node.new_qreal_flag:
            self.gui.timer_callback(self.node.new_qreal)
            self.node.new_qreal_flag = False

    def __init__(self, node, gui):
        QtCore.QThread.__init__(self)
        self.spinTimer = QtCore.QTimer()
        self.spinTimer.moveToThread(self)
        self.node = node
        self.gui = gui
        self.spinTimer.timeout.connect(self.ROS_spin)

    def run(self):
```

```
        self.spinTimer.start(100)
        loop = QtCore.QEventLoop()
        loop.exec_()


class MainWindow(QtWidgets.QMainWindow, Ui_MainWindow):
    def __init__(self, *args, obj=None, **kwargs):
        super(MainWindow, self).__init__(*args, **kwargs)
        self.setupUi(self)

        # Crea el nodo de ROS2
        self.gui_node = Gui()
        self.new_status()
        self.new_qdes()

        self.spinnerThread = ROS_spinner(self.gui_node, self)
        self.spinnerThread.start()

    def timer_callback(self, new_qreal):
        self.boxR1A.setValue(new_qreal.r_1a*1000)
        self.boxL1B.setValue(new_qreal.l_1b*1000)
        self.boxR2A.setValue(new_qreal.r_2a*1000)
        self.boxL2B.setValue(new_qreal.l_2b*1000)
        self.boxL1A.setValue(new_qreal.l_1a*1000)
        self.boxR1B.setValue(new_qreal.r_1b*1000)
        self.boxL2A.setValue(new_qreal.l_2a*1000)
        self.boxR2B.setValue(new_qreal.r_2b*1000)
        self.boxThetaA.setValue(new_qreal.theta_a)
        self.boxThetaB.setValue(new_qreal.theta_b)


    def new_status(self):
        status = hyrecro_interfaces.msg.Status()
        status.fijo_a = self.radioFijoA.isChecked()
        status.iman_a = self.radioImanesA.isChecked()
        status.iman_b = self.radioImanesB.isChecked()
        self.gui_node.status_pub.publish(status)

    def new_qdes(self):
        qreal = hyrecro_interfaces.msg.QReal()
        qreal.r_1a = self.boxR1ADes.value()/1000
        qreal.l_1b = self.boxL1BDes.value()/1000
        qreal.r_2a = self.boxR2ADes.value()/1000
        qreal.l_2b = self.boxL2BDes.value()/1000
        qreal.l_1a = self.boxL1ADes.value()/1000
        qreal.r_1b = self.boxR1BDes.value()/1000
        qreal.l_2a = self.boxL2ADes.value()/1000
        qreal.r_2b = self.boxR2BDes.value()/1000
        qreal.theta_a = self.boxThetaADes.value()
        qreal.theta_b = self.boxThetaBDes.value()
        self.gui_node.qreal_pub.publish(qreal)

    def new_vel(self):
        vels = Twist()
        vels.linear.x = (float(self.pushXp.isDown()) -  float(self.
    pushXm.isDown())) * self.boxDeltaPos.value()
```

```
103         vels.linear.y = (float(self.pushYp.isDown()) -  float(self.
    pushYm.isDown())) * self.boxDeltaPos.value()
104         vels.linear.z = (float(self.pushZp.isDown()) -  float(self.
    pushZm.isDown())) * self.boxDeltaPos.value()
105         vels.angular.x = (float(self.pushGXp.isDown()) -  float(
    self.pushGXm.isDown())) * self.boxDeltaOr.value()
106         vels.angular.y = (float(self.pushGYp.isDown()) -  float(
    self.pushGYm.isDown())) * self.boxDeltaOr.value()
107         vels.angular.z = (float(self.pushGZp.isDown()) -  float(
    self.pushGZm.isDown())) * self.boxDeltaOr.value()
108         self.gui_node.vel_pub.publish(vels)
109
110
111 def main(args=None):
112     rclpy.init(args=args)
113
114     app = QtWidgets.QApplication(sys.argv)
115
116     window = MainWindow()
117     window.show()
118
119     app.exec()
120
121     window.gui_node.destroy_node()
122     rclpy.shutdown()
123
124
125 if __name__ == '__main__':
126     main()
```

A.1.1. MainWindow

```
1  # -*- coding: utf-8 -*-
2
3  # Form implementation generated from reading ui file 'mainwindow.ui
       '
4  #
5  # Created by: PyQt5 UI code generator 5.15.6
6  #
7  # WARNING: Any manual changes made to this file will be lost when
       pyuic5 is
8  # run again.  Do not edit this file unless you know what you are
       doing.
9
10 import rclpy
11 from PyQt5 import QtCore, QtGui, QtWidgets
12
13
14 PUSH_DELAY = 200
15
16 class DoubleSlider(QtWidgets.QSlider):
17
18     # create our our signal that we can connect to if necessary
19     doubleValueChanged = QtCore.pyqtSignal(float)
20
```

```
    def __init__(self, *args, **kargs):
        super(DoubleSlider, self).__init__( *args, **kargs)
        self._multi = 10.0 ** 3
        self.setTracking(False)

        self.valueChanged.connect(self.emitDoubleValueChanged)

    def emitDoubleValueChanged(self):
        value = float(super(DoubleSlider, self).value())/self._multi
        self.doubleValueChanged.emit(value)

    def value(self):
        return float(super(DoubleSlider, self).value()) / self._multi

    def setMinimum(self, value):
        return super(DoubleSlider, self).setMinimum(value * self._multi)

    def setMaximum(self, value):
        return super(DoubleSlider, self).setMaximum(value * self._multi)

    def setSingleStep(self, value):
        return super(DoubleSlider, self).setSingleStep(value * self._multi)

    def singleStep(self):
        return float(super(DoubleSlider, self).singleStep()) / self._multi

    def setValue(self, value):
        super(DoubleSlider, self).setValue(int(value * self._multi))

    #unclickable
    def mousePressEvent(self, event):
        opt = QtWidgets.QStyleOptionSlider()
        self.initStyleOption(opt)
        pressedControl = self.style().hitTestComplexControl(QtWidgets.QStyle.CC_Slider, opt, event.pos(), self)
        if pressedControl != QtWidgets.QStyle.SC_SliderGroove:
            super(DoubleSlider, self).mousePressEvent(event)

class ReadOnlyDoubleSlider(DoubleSlider):

    def __init__(self, *args, **kargs):
        super(ReadOnlyDoubleSlider, self).__init__( *args, **kargs)

    def mousePressEvent(self, event):
        return None

    def wheelEvent(self, event):
        return None

class Ui_MainWindow(object):
```

```
70      def setupUi(self, MainWindow):
71          MainWindow.setObjectName("MainWindow")
72          MainWindow.resize(1170, 611)
73          self.centralwidget = QtWidgets.QWidget(MainWindow)
74          self.centralwidget.setObjectName("centralwidget")
75          self.verticalLayout = QtWidgets.QVBoxLayout(self.
    centralwidget)
76          self.verticalLayout.setContentsMargins(0, 0, 0, 0)
77          self.verticalLayout.setSpacing(0)
78          self.verticalLayout.setObjectName("verticalLayout")
79          self.grupo_enviar = QtWidgets.QTabWidget(self.centralwidget
    )
80          self.grupo_enviar.setObjectName("grupo_enviar")
81          self.tabWidget_2Page1 = QtWidgets.QWidget()
82          self.tabWidget_2Page1.setObjectName("tabWidget_2Page1")
83          self.horizontalLayout_6 = QtWidgets.QHBoxLayout(self.
    tabWidget_2Page1)
84          self.horizontalLayout_6.setObjectName("horizontalLayout_6")
85          self.verticalLayout_13 = QtWidgets.QVBoxLayout()
86          self.verticalLayout_13.setObjectName("verticalLayout_13")
87          self.gridLayout_7 = QtWidgets.QGridLayout()
88          self.gridLayout_7.setSpacing(0)
89          self.gridLayout_7.setObjectName("gridLayout_7")
90          self.des11 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
91          sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
92          sizePolicy.setHorizontalStretch(0)
93          sizePolicy.setVerticalStretch(0)
94          sizePolicy.setHeightForWidth(self.des11.sizePolicy().
    hasHeightForWidth())
95          self.des11.setSizePolicy(sizePolicy)
96          self.des11.setMinimumSize(QtCore.QSize(48, 48))
97          self.des11.setMaximumSize(QtCore.QSize(48, 48))
98          self.des11.setSizeIncrement(QtCore.QSize(0, 0))
99          self.des11.setWrapping(False)
100         self.des11.setAlignment(QtCore.Qt.AlignCenter)
101         self.des11.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
102         self.des11.setKeyboardTracking(True)
103         self.des11.setDecimals(2)
104         self.des11.setMaximum(1.0)
105         self.des11.setProperty("value", 1.0)
106         self.des11.setObjectName("des11")
107         self.gridLayout_7.addWidget(self.des11, 0, 0, 1, 1)
108         self.des12 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
109         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
110         sizePolicy.setHorizontalStretch(0)
111         sizePolicy.setVerticalStretch(0)
112         sizePolicy.setHeightForWidth(self.des12.sizePolicy().
    hasHeightForWidth())
113         self.des12.setSizePolicy(sizePolicy)
114         self.des12.setMinimumSize(QtCore.QSize(48, 48))
115         self.des12.setMaximumSize(QtCore.QSize(48, 48))
116         self.des12.setSizeIncrement(QtCore.QSize(0, 0))
```

```
117         self.des12.setWrapping(False)
118         self.des12.setAlignment(QtCore.Qt.AlignCenter)
119         self.des12.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
120         self.des12.setKeyboardTracking(True)
121         self.des12.setDecimals(2)
122         self.des12.setMaximum(1.0)
123         self.des12.setObjectName("des12")
124         self.gridLayout_7.addWidget(self.des12, 0, 1, 1, 1)
125         self.des13 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
126         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
127         sizePolicy.setHorizontalStretch(0)
128         sizePolicy.setVerticalStretch(0)
129         sizePolicy.setHeightForWidth(self.des13.sizePolicy().
    hasHeightForWidth())
130         self.des13.setSizePolicy(sizePolicy)
131         self.des13.setMinimumSize(QtCore.QSize(48, 48))
132         self.des13.setMaximumSize(QtCore.QSize(48, 48))
133         self.des13.setSizeIncrement(QtCore.QSize(0, 0))
134         self.des13.setWrapping(False)
135         self.des13.setAlignment(QtCore.Qt.AlignCenter)
136         self.des13.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
137         self.des13.setKeyboardTracking(True)
138         self.des13.setDecimals(2)
139         self.des13.setMaximum(1.0)
140         self.des13.setObjectName("des13")
141         self.gridLayout_7.addWidget(self.des13, 0, 2, 1, 1)
142         self.des14 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
143         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
144         sizePolicy.setHorizontalStretch(0)
145         sizePolicy.setVerticalStretch(0)
146         sizePolicy.setHeightForWidth(self.des14.sizePolicy().
    hasHeightForWidth())
147         self.des14.setSizePolicy(sizePolicy)
148         self.des14.setMinimumSize(QtCore.QSize(48, 48))
149         self.des14.setMaximumSize(QtCore.QSize(48, 48))
150         self.des14.setSizeIncrement(QtCore.QSize(0, 0))
151         self.des14.setWrapping(False)
152         self.des14.setAlignment(QtCore.Qt.AlignCenter)
153         self.des14.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
154         self.des14.setKeyboardTracking(True)
155         self.des14.setDecimals(2)
156         self.des14.setMaximum(1000.0)
157         self.des14.setObjectName("des14")
158         self.gridLayout_7.addWidget(self.des14, 0, 3, 1, 1)
159         self.des21 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
160         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
161         sizePolicy.setHorizontalStretch(0)
162         sizePolicy.setVerticalStretch(0)
```

```
163         sizePolicy.setHeightForWidth(self.des21.sizePolicy().
    hasHeightForWidth())
164         self.des21.setSizePolicy(sizePolicy)
165         self.des21.setMinimumSize(QtCore.QSize(48, 48))
166         self.des21.setMaximumSize(QtCore.QSize(48, 48))
167         self.des21.setSizeIncrement(QtCore.QSize(0, 0))
168         self.des21.setWrapping(False)
169         self.des21.setAlignment(QtCore.Qt.AlignCenter)
170         self.des21.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
171         self.des21.setKeyboardTracking(True)
172         self.des21.setDecimals(2)
173         self.des21.setMaximum(1.0)
174         self.des21.setObjectName("des21")
175         self.gridLayout_7.addWidget(self.des21, 1, 0, 1, 1)
176         self.des22 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
177         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
178         sizePolicy.setHorizontalStretch(0)
179         sizePolicy.setVerticalStretch(0)
180         sizePolicy.setHeightForWidth(self.des22.sizePolicy().
    hasHeightForWidth())
181         self.des22.setSizePolicy(sizePolicy)
182         self.des22.setMinimumSize(QtCore.QSize(48, 48))
183         self.des22.setMaximumSize(QtCore.QSize(48, 48))
184         self.des22.setSizeIncrement(QtCore.QSize(0, 0))
185         self.des22.setWrapping(False)
186         self.des22.setAlignment(QtCore.Qt.AlignCenter)
187         self.des22.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
188         self.des22.setKeyboardTracking(True)
189         self.des22.setDecimals(2)
190         self.des22.setMaximum(1.0)
191         self.des22.setProperty("value", 1.0)
192         self.des22.setObjectName("des22")
193         self.gridLayout_7.addWidget(self.des22, 1, 1, 1, 1)
194         self.des23 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
195         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
196         sizePolicy.setHorizontalStretch(0)
197         sizePolicy.setVerticalStretch(0)
198         sizePolicy.setHeightForWidth(self.des23.sizePolicy().
    hasHeightForWidth())
199         self.des23.setSizePolicy(sizePolicy)
200         self.des23.setMinimumSize(QtCore.QSize(48, 48))
201         self.des23.setMaximumSize(QtCore.QSize(48, 48))
202         self.des23.setSizeIncrement(QtCore.QSize(0, 0))
203         self.des23.setWrapping(False)
204         self.des23.setAlignment(QtCore.Qt.AlignCenter)
205         self.des23.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
206         self.des23.setKeyboardTracking(True)
207         self.des23.setDecimals(2)
208         self.des23.setMaximum(1.0)
209         self.des23.setObjectName("des23")
```

```
        self.gridLayout_7.addWidget(self.des23, 1, 2, 1, 1)
        self.des24 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.des24.sizePolicy().
    hasHeightForWidth())
        self.des24.setSizePolicy(sizePolicy)
        self.des24.setMinimumSize(QtCore.QSize(48, 48))
        self.des24.setMaximumSize(QtCore.QSize(48, 48))
        self.des24.setSizeIncrement(QtCore.QSize(0, 0))
        self.des24.setWrapping(False)
        self.des24.setAlignment(QtCore.Qt.AlignCenter)
        self.des24.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
        self.des24.setKeyboardTracking(True)
        self.des24.setDecimals(2)
        self.des24.setMaximum(1000.0)
        self.des24.setObjectName("des24")
        self.gridLayout_7.addWidget(self.des24, 1, 3, 1, 1)
        self.des31 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.des31.sizePolicy().
    hasHeightForWidth())
        self.des31.setSizePolicy(sizePolicy)
        self.des31.setMinimumSize(QtCore.QSize(48, 48))
        self.des31.setMaximumSize(QtCore.QSize(48, 48))
        self.des31.setSizeIncrement(QtCore.QSize(0, 0))
        self.des31.setWrapping(False)
        self.des31.setAlignment(QtCore.Qt.AlignCenter)
        self.des31.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
        self.des31.setKeyboardTracking(True)
        self.des31.setDecimals(2)
        self.des31.setMaximum(1.0)
        self.des31.setObjectName("des31")
        self.gridLayout_7.addWidget(self.des31, 2, 0, 1, 1)
        self.des32 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.des32.sizePolicy().
    hasHeightForWidth())
        self.des32.setSizePolicy(sizePolicy)
        self.des32.setMinimumSize(QtCore.QSize(48, 48))
        self.des32.setMaximumSize(QtCore.QSize(48, 48))
        self.des32.setSizeIncrement(QtCore.QSize(0, 0))
        self.des32.setWrapping(False)
        self.des32.setAlignment(QtCore.Qt.AlignCenter)
```

```
256         self.des32.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
257         self.des32.setKeyboardTracking(True)
258         self.des32.setDecimals(2)
259         self.des32.setMaximum(1.0)
260         self.des32.setObjectName("des32")
261         self.gridLayout_7.addWidget(self.des32, 2, 1, 1, 1)
262         self.des33 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
263         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
264         sizePolicy.setHorizontalStretch(0)
265         sizePolicy.setVerticalStretch(0)
266         sizePolicy.setHeightForWidth(self.des33.sizePolicy().
    hasHeightForWidth())
267         self.des33.setSizePolicy(sizePolicy)
268         self.des33.setMinimumSize(QtCore.QSize(48, 48))
269         self.des33.setMaximumSize(QtCore.QSize(48, 48))
270         self.des33.setSizeIncrement(QtCore.QSize(0, 0))
271         self.des33.setWrapping(False)
272         self.des33.setAlignment(QtCore.Qt.AlignCenter)
273         self.des33.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
274         self.des33.setKeyboardTracking(True)
275         self.des33.setDecimals(2)
276         self.des33.setMaximum(1.0)
277         self.des33.setProperty("value", 1.0)
278         self.des33.setObjectName("des33")
279         self.gridLayout_7.addWidget(self.des33, 2, 2, 1, 1)
280         self.des34 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
281         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
282         sizePolicy.setHorizontalStretch(0)
283         sizePolicy.setVerticalStretch(0)
284         sizePolicy.setHeightForWidth(self.des34.sizePolicy().
    hasHeightForWidth())
285         self.des34.setSizePolicy(sizePolicy)
286         self.des34.setMinimumSize(QtCore.QSize(48, 48))
287         self.des34.setMaximumSize(QtCore.QSize(48, 48))
288         self.des34.setSizeIncrement(QtCore.QSize(0, 0))
289         self.des34.setWrapping(False)
290         self.des34.setAlignment(QtCore.Qt.AlignCenter)
291         self.des34.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
292         self.des34.setKeyboardTracking(True)
293         self.des34.setDecimals(2)
294         self.des34.setMaximum(1000.0)
295         self.des34.setObjectName("des34")
296         self.gridLayout_7.addWidget(self.des34, 2, 3, 1, 1)
297         self.des41 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
298         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
299         sizePolicy.setHorizontalStretch(0)
300         sizePolicy.setVerticalStretch(0)
```

```
301         sizePolicy.setHeightForWidth(self.des41.sizePolicy().
    hasHeightForWidth())
302         self.des41.setSizePolicy(sizePolicy)
303         self.des41.setMinimumSize(QtCore.QSize(48, 48))
304         self.des41.setMaximumSize(QtCore.QSize(48, 48))
305         self.des41.setSizeIncrement(QtCore.QSize(0, 0))
306         self.des41.setWrapping(False)
307         self.des41.setAlignment(QtCore.Qt.AlignCenter)
308         self.des41.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
309         self.des41.setKeyboardTracking(True)
310         self.des41.setDecimals(2)
311         self.des41.setMaximum(0.0)
312         self.des41.setObjectName("des41")
313         self.gridLayout_7.addWidget(self.des41, 3, 0, 1, 1)
314         self.des42 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
315         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
316         sizePolicy.setHorizontalStretch(0)
317         sizePolicy.setVerticalStretch(0)
318         sizePolicy.setHeightForWidth(self.des42.sizePolicy().
    hasHeightForWidth())
319         self.des42.setSizePolicy(sizePolicy)
320         self.des42.setMinimumSize(QtCore.QSize(48, 48))
321         self.des42.setMaximumSize(QtCore.QSize(48, 48))
322         self.des42.setSizeIncrement(QtCore.QSize(0, 0))
323         self.des42.setWrapping(False)
324         self.des42.setAlignment(QtCore.Qt.AlignCenter)
325         self.des42.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
326         self.des42.setKeyboardTracking(True)
327         self.des42.setDecimals(2)
328         self.des42.setMaximum(0.0)
329         self.des42.setObjectName("des42")
330         self.gridLayout_7.addWidget(self.des42, 3, 1, 1, 1)
331         self.des43 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
332         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
333         sizePolicy.setHorizontalStretch(0)
334         sizePolicy.setVerticalStretch(0)
335         sizePolicy.setHeightForWidth(self.des43.sizePolicy().
    hasHeightForWidth())
336         self.des43.setSizePolicy(sizePolicy)
337         self.des43.setMinimumSize(QtCore.QSize(48, 48))
338         self.des43.setMaximumSize(QtCore.QSize(48, 48))
339         self.des43.setSizeIncrement(QtCore.QSize(0, 0))
340         self.des43.setWrapping(False)
341         self.des43.setAlignment(QtCore.Qt.AlignCenter)
342         self.des43.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
343         self.des43.setKeyboardTracking(True)
344         self.des43.setDecimals(2)
345         self.des43.setMaximum(0.0)
346         self.des43.setObjectName("des43")
347         self.gridLayout_7.addWidget(self.des43, 3, 2, 1, 1)
```

```
348         self.des44 = QtWidgets.QDoubleSpinBox(self.tabWidget_2Page1
    )
349         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
350         sizePolicy.setHorizontalStretch(0)
351         sizePolicy.setVerticalStretch(0)
352         sizePolicy.setHeightForWidth(self.des44.sizePolicy().
    hasHeightForWidth())
353         self.des44.setSizePolicy(sizePolicy)
354         self.des44.setMinimumSize(QtCore.QSize(48, 48))
355         self.des44.setMaximumSize(QtCore.QSize(48, 48))
356         self.des44.setSizeIncrement(QtCore.QSize(0, 0))
357         self.des44.setWrapping(False)
358         self.des44.setAlignment(QtCore.Qt.AlignCenter)
359         self.des44.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
360         self.des44.setKeyboardTracking(True)
361         self.des44.setDecimals(2)
362         self.des44.setMinimum(1.0)
363         self.des44.setMaximum(1.0)
364         self.des44.setObjectName("des44")
365         self.gridLayout_7.addWidget(self.des44, 3, 3, 1, 1)
366         self.verticalLayout_13.addLayout(self.gridLayout_7)
367         self.pushPose = QtWidgets.QPushButton(self.tabWidget_2Page1
    )
368         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Minimum, QtWidgets.QSizePolicy.Fixed)
369         sizePolicy.setHorizontalStretch(0)
370         sizePolicy.setVerticalStretch(0)
371         sizePolicy.setHeightForWidth(self.pushPose.sizePolicy().
    hasHeightForWidth())
372         self.pushPose.setSizePolicy(sizePolicy)
373         self.pushPose.setObjectName("pushPose")
374         self.verticalLayout_13.addWidget(self.pushPose)
375         self.horizontalLayout_6.addLayout(self.verticalLayout_13)
376         self.line = QtWidgets.QFrame(self.tabWidget_2Page1)
377         self.line.setFrameShape(QtWidgets.QFrame.VLine)
378         self.line.setFrameShadow(QtWidgets.QFrame.Sunken)
379         self.line.setObjectName("line")
380         self.horizontalLayout_6.addWidget(self.line)
381         self.verticalLayout_12 = QtWidgets.QVBoxLayout()
382         self.verticalLayout_12.setObjectName("verticalLayout_12")
383         self.horizontalLayout = QtWidgets.QHBoxLayout()
384         self.horizontalLayout.setObjectName("horizontalLayout")
385         self.verticalLayout_2 = QtWidgets.QVBoxLayout()
386         self.verticalLayout_2.setObjectName("verticalLayout_2")
387         self.label = QtWidgets.QLabel(self.tabWidget_2Page1)
388         self.label.setObjectName("label")
389         self.verticalLayout_2.addWidget(self.label, 0, QtCore.Qt.
    AlignHCenter)
390         self.sliderR1ADes = DoubleSlider(self.tabWidget_2Page1)
391         self.sliderR1ADes.setEnabled(True)
392         self.sliderR1ADes.setMinimum(102.5)
393         self.sliderR1ADes.setMaximum(152)
394         self.sliderR1ADes.setSingleStep(10)
395         self.sliderR1ADes.setOrientation(QtCore.Qt.Vertical)
396         self.sliderR1ADes.setProperty("value", 125)
```

```
397         self.sliderR1ADes.setObjectName("sliderR1ADes")
398         self.verticalLayout_2.addWidget(self.sliderR1ADes, 0,
    QtCore.Qt.AlignHCenter)
399         self.boxR1ADes = QtWidgets.QDoubleSpinBox(self.
    tabWidget_2Page1)
400         self.boxR1ADes.setMinimum(102.5)
401         self.boxR1ADes.setMaximum(152.0)
402         self.boxR1ADes.setSingleStep(0.5)
403         self.boxR1ADes.setProperty("value", 125)
404         self.boxR1ADes.setObjectName("boxR1ADes")
405         self.verticalLayout_2.addWidget(self.boxR1ADes)
406         self.horizontalLayout.addLayout(self.verticalLayout_2)
407         self.verticalLayout_3 = QtWidgets.QVBoxLayout()
408         self.verticalLayout_3.setObjectName("verticalLayout_3")
409         self.label_2 = QtWidgets.QLabel(self.tabWidget_2Page1)
410         self.label_2.setObjectName("label_2")
411         self.verticalLayout_3.addWidget(self.label_2, 0, QtCore.Qt.
    AlignHCenter)
412         self.sliderL1BDes = DoubleSlider(self.tabWidget_2Page1)
413         self.sliderL1BDes.setMinimum(102.5)
414         self.sliderL1BDes.setMaximum(152)
415         self.sliderL1BDes.setSingleStep(10)
416         self.sliderL1BDes.setProperty("value", 125)
417         self.sliderL1BDes.setOrientation(QtCore.Qt.Vertical)
418         self.sliderL1BDes.setObjectName("sliderL1BDes")
419         self.verticalLayout_3.addWidget(self.sliderL1BDes, 0,
    QtCore.Qt.AlignHCenter)
420         self.boxL1BDes = QtWidgets.QDoubleSpinBox(self.
    tabWidget_2Page1)
421         self.boxL1BDes.setMinimum(102.5)
422         self.boxL1BDes.setMaximum(152.0)
423         self.boxL1BDes.setProperty("value", 125)
424         self.boxL1BDes.setObjectName("boxL1BDes")
425         self.verticalLayout_3.addWidget(self.boxL1BDes)
426         self.horizontalLayout.addLayout(self.verticalLayout_3)
427         self.verticalLayout_4 = QtWidgets.QVBoxLayout()
428         self.verticalLayout_4.setObjectName("verticalLayout_4")
429         self.label_3 = QtWidgets.QLabel(self.tabWidget_2Page1)
430         self.label_3.setObjectName("label_3")
431         self.verticalLayout_4.addWidget(self.label_3, 0, QtCore.Qt.
    AlignHCenter)
432         self.sliderR2ADes = DoubleSlider(self.tabWidget_2Page1)
433         self.sliderR2ADes.setMinimum(102.5)
434         self.sliderR2ADes.setMaximum(152)
435         self.sliderR2ADes.setSingleStep(10)
436         self.sliderR2ADes.setProperty("value", 125)
437         self.sliderR2ADes.setOrientation(QtCore.Qt.Vertical)
438         self.sliderR2ADes.setObjectName("sliderR2ADes")
439         self.verticalLayout_4.addWidget(self.sliderR2ADes, 0,
    QtCore.Qt.AlignHCenter)
440         self.boxR2ADes = QtWidgets.QDoubleSpinBox(self.
    tabWidget_2Page1)
441         self.boxR2ADes.setMinimum(102.5)
442         self.boxR2ADes.setMaximum(152.0)
443         self.boxR2ADes.setProperty("value", 125)
444         self.boxR2ADes.setObjectName("boxR2ADes")
445         self.verticalLayout_4.addWidget(self.boxR2ADes)
```

```
446         self.horizontalLayout.addLayout(self.verticalLayout_4)
447         self.verticalLayout_6 = QtWidgets.QVBoxLayout()
448         self.verticalLayout_6.setObjectName("verticalLayout_6")
449         self.label_5 = QtWidgets.QLabel(self.tabWidget_2Page1)
450         self.label_5.setObjectName("label_5")
451         self.verticalLayout_6.addWidget(self.label_5, 0, QtCore.Qt.
    AlignHCenter)
452         self.sliderL2BDes = DoubleSlider(self.tabWidget_2Page1)
453         self.sliderL2BDes.setMinimum(102.5)
454         self.sliderL2BDes.setMaximum(152)
455         self.sliderL2BDes.setProperty("value", 125)
456         self.sliderL2BDes.setSingleStep(10)
457         self.sliderL2BDes.setOrientation(QtCore.Qt.Vertical)
458         self.sliderL2BDes.setObjectName("sliderL2BDes")
459         self.verticalLayout_6.addWidget(self.sliderL2BDes, 0,
    QtCore.Qt.AlignHCenter)
460         self.boxL2BDes = QtWidgets.QDoubleSpinBox(self.
    tabWidget_2Page1)
461         self.boxL2BDes.setMinimum(102.5)
462         self.boxL2BDes.setMaximum(152.0)
463         self.boxL2BDes.setProperty("value", 125)
464         self.boxL2BDes.setObjectName("boxL2BDes")
465         self.verticalLayout_6.addWidget(self.boxL2BDes)
466         self.horizontalLayout.addLayout(self.verticalLayout_6)
467         self.verticalLayout_9 = QtWidgets.QVBoxLayout()
468         self.verticalLayout_9.setObjectName("verticalLayout_9")
469         self.label_8 = QtWidgets.QLabel(self.tabWidget_2Page1)
470         self.label_8.setObjectName("label_8")
471         self.verticalLayout_9.addWidget(self.label_8, 0, QtCore.Qt.
    AlignHCenter)
472         self.sliderL1ADes = DoubleSlider(self.tabWidget_2Page1)
473         self.sliderL1ADes.setMinimum(102.5)
474         self.sliderL1ADes.setMaximum(152)
475         self.sliderL1ADes.setProperty("value", 125)
476         self.sliderL1ADes.setSingleStep(10)
477         self.sliderL1ADes.setOrientation(QtCore.Qt.Vertical)
478         self.sliderL1ADes.setObjectName("sliderL1ADes")
479         self.verticalLayout_9.addWidget(self.sliderL1ADes, 0,
    QtCore.Qt.AlignHCenter)
480         self.boxL1ADes = QtWidgets.QDoubleSpinBox(self.
    tabWidget_2Page1)
481         self.boxL1ADes.setMinimum(102.5)
482         self.boxL1ADes.setMaximum(152.0)
483         self.boxL1ADes.setProperty("value", 125)
484         self.boxL1ADes.setObjectName("boxL1ADes")
485         self.verticalLayout_9.addWidget(self.boxL1ADes)
486         self.horizontalLayout.addLayout(self.verticalLayout_9)
487         self.verticalLayout_5 = QtWidgets.QVBoxLayout()
488         self.verticalLayout_5.setObjectName("verticalLayout_5")
489         self.label_4 = QtWidgets.QLabel(self.tabWidget_2Page1)
490         self.label_4.setObjectName("label_4")
491         self.verticalLayout_5.addWidget(self.label_4, 0, QtCore.Qt.
    AlignHCenter)
492         self.sliderR1BDes = DoubleSlider(self.tabWidget_2Page1)
493         self.sliderR1BDes.setMinimum(102.5)
494         self.sliderR1BDes.setMaximum(152)
495         self.sliderR1BDes.setProperty("value", 125)
```

```
496         self.sliderR1BDes.setSingleStep(10)
497         self.sliderR1BDes.setOrientation(QtCore.Qt.Vertical)
498         self.sliderR1BDes.setObjectName("sliderR1BDes")
499         self.verticalLayout_5.addWidget(self.sliderR1BDes, 0,
    QtCore.Qt.AlignHCenter)
500         self.boxR1BDes = QtWidgets.QDoubleSpinBox(self.
    tabWidget_2Page1)
501         self.boxR1BDes.setMinimum(102.5)
502         self.boxR1BDes.setMaximum(152.0)
503         self.boxR1BDes.setProperty("value", 125)
504         self.boxR1BDes.setObjectName("boxR1BDes")
505         self.verticalLayout_5.addWidget(self.boxR1BDes)
506         self.horizontalLayout.addLayout(self.verticalLayout_5)
507         self.verticalLayout_8 = QtWidgets.QVBoxLayout()
508         self.verticalLayout_8.setObjectName("verticalLayout_8")
509         self.label_7 = QtWidgets.QLabel(self.tabWidget_2Page1)
510         self.label_7.setObjectName("label_7")
511         self.verticalLayout_8.addWidget(self.label_7, 0, QtCore.Qt.
    AlignHCenter)
512         self.sliderL2ADes = DoubleSlider(self.tabWidget_2Page1)
513         self.sliderL2ADes.setMinimum(102.5)
514         self.sliderL2ADes.setMaximum(152)
515         self.sliderL2ADes.setSingleStep(10)
516         self.sliderL2ADes.setProperty("value", 125)
517         self.sliderL2ADes.setOrientation(QtCore.Qt.Vertical)
518         self.sliderL2ADes.setObjectName("sliderL2ADes")
519         self.verticalLayout_8.addWidget(self.sliderL2ADes, 0,
    QtCore.Qt.AlignHCenter)
520         self.boxL2ADes = QtWidgets.QDoubleSpinBox(self.
    tabWidget_2Page1)
521         self.boxL2ADes.setMinimum(102.5)
522         self.boxL2ADes.setMaximum(152.0)
523         self.boxL2ADes.setProperty("value", 125)
524         self.boxL2ADes.setObjectName("boxL2ADes")
525         self.verticalLayout_8.addWidget(self.boxL2ADes)
526         self.horizontalLayout.addLayout(self.verticalLayout_8)
527         self.verticalLayout_7 = QtWidgets.QVBoxLayout()
528         self.verticalLayout_7.setObjectName("verticalLayout_7")
529         self.label_6 = QtWidgets.QLabel(self.tabWidget_2Page1)
530         self.label_6.setObjectName("label_6")
531         self.verticalLayout_7.addWidget(self.label_6, 0, QtCore.Qt.
    AlignHCenter)
532         self.sliderR2BDes = DoubleSlider(self.tabWidget_2Page1)
533         self.sliderR2BDes.setMinimum(102.5)
534         self.sliderR2BDes.setMaximum(152)
535         self.sliderR2BDes.setSingleStep(10)
536         self.sliderR2BDes.setProperty("value", 125)
537         self.sliderR2BDes.setOrientation(QtCore.Qt.Vertical)
538         self.sliderR2BDes.setObjectName("sliderR2BDes")
539         self.verticalLayout_7.addWidget(self.sliderR2BDes, 0,
    QtCore.Qt.AlignHCenter)
540         self.boxR2BDes = QtWidgets.QDoubleSpinBox(self.
    tabWidget_2Page1)
541         self.boxR2BDes.setMinimum(102.5)
542         self.boxR2BDes.setMaximum(152.0)
543         self.boxR2BDes.setProperty("value", 125)
544         self.boxR2BDes.setObjectName("boxR2BDes")
```

```
545             self.verticalLayout_7.addWidget(self.boxR2BDes)
546             self.horizontalLayout.addLayout(self.verticalLayout_7)
547             self.verticalLayout_10 = QtWidgets.QVBoxLayout()
548             self.verticalLayout_10.setObjectName("verticalLayout_10")
549             self.label_9 = QtWidgets.QLabel(self.tabWidget_2Page1)
550             self.label_9.setObjectName("label_9")
551             self.verticalLayout_10.addWidget(self.label_9, 0, QtCore.Qt
    .AlignHCenter)
552             self.sliderThetaADes = DoubleSlider(self.tabWidget_2Page1)
553             self.sliderThetaADes.setMinimum(-1.185)
554             self.sliderThetaADes.setMaximum(1.245)
555             self.sliderThetaADes.setSingleStep(10)
556             self.sliderThetaADes.setProperty("value", 0)
557             self.sliderThetaADes.setOrientation(QtCore.Qt.Vertical)
558             self.sliderThetaADes.setObjectName("sliderThetaADes")
559             self.verticalLayout_10.addWidget(self.sliderThetaADes, 0,
    QtCore.Qt.AlignHCenter)
560             self.boxThetaADes = QtWidgets.QDoubleSpinBox(self.
    tabWidget_2Page1)
561             self.boxThetaADes.setDecimals(3)
562             self.boxThetaADes.setMinimum(-1.185)
563             self.boxThetaADes.setMaximum(1.245)
564             self.boxThetaADes.setSingleStep(0.01)
565             self.boxThetaADes.setProperty("value", 0.0)
566             self.boxThetaADes.setObjectName("boxThetaADes")
567             self.verticalLayout_10.addWidget(self.boxThetaADes)
568             self.horizontalLayout.addLayout(self.verticalLayout_10)
569             self.verticalLayout_11 = QtWidgets.QVBoxLayout()
570             self.verticalLayout_11.setObjectName("verticalLayout_11")
571             self.label_10 = QtWidgets.QLabel(self.tabWidget_2Page1)
572             self.label_10.setObjectName("label_10")
573             self.verticalLayout_11.addWidget(self.label_10, 0, QtCore.
    Qt.AlignHCenter)
574             self.sliderThetaBDes = DoubleSlider(self.tabWidget_2Page1)
575             self.sliderThetaBDes.setMinimum(-1.185)
576             self.sliderThetaBDes.setMaximum(1.245)
577             self.sliderThetaBDes.setSingleStep(10)
578             self.sliderThetaBDes.setProperty("value", 0)
579             self.sliderThetaBDes.setOrientation(QtCore.Qt.Vertical)
580             self.sliderThetaBDes.setObjectName("sliderThetaBDes")
581             self.verticalLayout_11.addWidget(self.sliderThetaBDes, 0,
    QtCore.Qt.AlignHCenter)
582             self.boxThetaBDes = QtWidgets.QDoubleSpinBox(self.
    tabWidget_2Page1)
583             self.boxThetaBDes.setDecimals(3)
584             self.boxThetaBDes.setMinimum(-1.185)
585             self.boxThetaBDes.setMaximum(1.245)
586             self.boxThetaBDes.setSingleStep(0.01)
587             self.boxThetaBDes.setProperty("value", 0.0)
588             self.boxThetaBDes.setObjectName("boxThetaBDes")
589             self.verticalLayout_11.addWidget(self.boxThetaBDes)
590             self.horizontalLayout.addLayout(self.verticalLayout_11)
591             self.verticalLayout_12.addLayout(self.horizontalLayout)
592             self.pushArt = QtWidgets.QPushButton(self.tabWidget_2Page1)
593             self.pushArt.setObjectName("pushArt")
594             self.verticalLayout_12.addWidget(self.pushArt)
595             self.horizontalLayout_6.addLayout(self.verticalLayout_12)
```

```
596         self.horizontalLayout_6.setStretch(2, 1)
597         self.grupo_enviar.addTab(self.tabWidget_2Page1, "")
598         self.tab = QtWidgets.QWidget()
599         self.tab.setObjectName("tab")
600         self.horizontalLayout_2 = QtWidgets.QHBoxLayout(self.tab)
601         self.horizontalLayout_2.setObjectName("horizontalLayout_2")
602         self.verticalLayout_26 = QtWidgets.QVBoxLayout()
603         self.verticalLayout_26.setObjectName("verticalLayout_26")
604         self.label_28 = QtWidgets.QLabel(self.tab)
605         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Preferred, QtWidgets.QSizePolicy.Maximum)
606         sizePolicy.setHorizontalStretch(0)
607         sizePolicy.setVerticalStretch(0)
608         sizePolicy.setHeightForWidth(self.label_28.sizePolicy().
    hasHeightForWidth())
609         self.label_28.setSizePolicy(sizePolicy)
610         self.label_28.setObjectName("label_28")
611         self.verticalLayout_26.addWidget(self.label_28)
612         self.gridLayout_3 = QtWidgets.QGridLayout()
613         self.gridLayout_3.setObjectName("gridLayout_3")
614         self.label_24 = QtWidgets.QLabel(self.tab)
615         self.label_24.setObjectName("label_24")
616         self.gridLayout_3.addWidget(self.label_24, 0, 1, 1, 1,
    QtCore.Qt.AlignHCenter|QtCore.Qt.AlignBottom)
617         self.label_27 = QtWidgets.QLabel(self.tab)
618         self.label_27.setObjectName("label_27")
619         self.gridLayout_3.addWidget(self.label_27, 1, 0, 1, 1,
    QtCore.Qt.AlignRight)
620         self.gridLayout_2 = QtWidgets.QGridLayout()
621         self.gridLayout_2.setObjectName("gridLayout_2")
622         self.pushYp = QtWidgets.QPushButton(self.tab)
623         self.pushYp.setText("")
624         icon = QtGui.QIcon()
625         icon.addPixmap(QtGui.QPixmap("/home/marc/hyrecro_ws/src/
    gui_hyrecro/gui_hyrecro/iconos/arrow-up.svg"), QtGui.QIcon.
    Normal, QtGui.QIcon.Off)
626         self.pushYp.setIcon(icon)
627         self.pushYp.setCheckable(False)
628         self.pushYp.setChecked(False)
629         self.pushYp.setObjectName("pushYp")
630         self.pushYp.setAutoRepeat(True)
631         self.pushYp.setAutoRepeatDelay(PUSH_DELAY)
632         self.pushYp.setAutoRepeatInterval(PUSH_DELAY)
633         self.gridLayout_2.addWidget(self.pushYp, 0, 1, 1, 1)
634         self.pushXm = QtWidgets.QPushButton(self.tab)
635         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Fixed, QtWidgets.QSizePolicy.Minimum)
636         sizePolicy.setHorizontalStretch(0)
637         sizePolicy.setVerticalStretch(0)
638         sizePolicy.setHeightForWidth(self.pushXm.sizePolicy().
    hasHeightForWidth())
639         self.pushXm.setSizePolicy(sizePolicy)
640         self.pushXm.setText("")
641         icon1 = QtGui.QIcon()
642         icon1.addPixmap(QtGui.QPixmap("/home/marc/hyrecro_ws/src/
    gui_hyrecro/gui_hyrecro/iconos/arrow-left.svg"), QtGui.QIcon.
    Normal, QtGui.QIcon.Off)
```

```
643         self.pushXm.setIcon(icon1)
644         self.pushXm.setCheckable(False)
645         self.pushXm.setChecked(False)
646         self.pushXm.setObjectName("pushXm")
647         self.pushXm.setAutoRepeat(True)
648         self.pushXm.setAutoRepeatDelay(PUSH_DELAY)
649         self.pushXm.setAutoRepeatInterval(PUSH_DELAY)
650         self.gridLayout_2.addWidget(self.pushXm, 1, 0, 1, 1)
651         self.pushXp = QtWidgets.QPushButton(self.tab)
652         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Fixed, QtWidgets.QSizePolicy.Minimum)
653         sizePolicy.setHorizontalStretch(0)
654         sizePolicy.setVerticalStretch(0)
655         sizePolicy.setHeightForWidth(self.pushXp.sizePolicy().
    hasHeightForWidth())
656         self.pushXp.setSizePolicy(sizePolicy)
657         self.pushXp.setText("")
658         icon2 = QtGui.QIcon()
659         icon2.addPixmap(QtGui.QPixmap("/home/marc/hyrecro_ws/src/
    gui_hyrecro/gui_hyrecro/iconos/arrow-right.svg"), QtGui.QIcon.
    Normal, QtGui.QIcon.Off)
660         self.pushXp.setIcon(icon2)
661         self.pushXp.setCheckable(False)
662         self.pushXp.setChecked(False)
663         self.pushXp.setObjectName("pushXp")
664         self.pushXp.setAutoRepeat(True)
665         self.pushXp.setAutoRepeatDelay(PUSH_DELAY)
666         self.pushXp.setAutoRepeatInterval(PUSH_DELAY)
667         self.gridLayout_2.addWidget(self.pushXp, 1, 2, 1, 1)
668         self.pushYm = QtWidgets.QPushButton(self.tab)
669         self.pushYm.setText("")
670         icon3 = QtGui.QIcon()
671         icon3.addPixmap(QtGui.QPixmap("/home/marc/hyrecro_ws/src/
    gui_hyrecro/gui_hyrecro/iconos/arrow-down.svg"), QtGui.QIcon.
    Normal, QtGui.QIcon.Off)
672         self.pushYm.setIcon(icon3)
673         self.pushYm.setCheckable(False)
674         self.pushYm.setChecked(False)
675         self.pushYm.setObjectName("pushYm")
676         self.pushYm.setAutoRepeat(True)
677         self.pushYm.setAutoRepeatDelay(PUSH_DELAY)
678         self.pushYm.setAutoRepeatInterval(PUSH_DELAY)
679         self.gridLayout_2.addWidget(self.pushYm, 2, 1, 1, 1)
680         self.gridLayout = QtWidgets.QGridLayout()
681         self.gridLayout.setObjectName("gridLayout")
682         self.label_23 = QtWidgets.QLabel(self.tab)
683         self.label_23.setObjectName("label_23")
684         self.gridLayout.addWidget(self.label_23, 0, 0, 1, 1, QtCore
    .Qt.AlignRight)
685         self.pushZm = QtWidgets.QPushButton(self.tab)
686         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Preferred, QtWidgets.QSizePolicy.Preferred)
687         sizePolicy.setHorizontalStretch(0)
688         sizePolicy.setVerticalStretch(0)
689         sizePolicy.setHeightForWidth(self.pushZm.sizePolicy().
    hasHeightForWidth())
690         self.pushZm.setSizePolicy(sizePolicy)
```

```
691         self.pushZm.setText("")
692         icon4 = QtGui.QIcon()
693         icon4.addPixmap(QtGui.QPixmap("/home/marc/hyrecro_ws/src/
    gui_hyrecro/gui_hyrecro/iconos/arrow-up-right.svg"), QtGui.
    QIcon.Normal, QtGui.QIcon.Off)
694         self.pushZm.setIcon(icon4)
695         self.pushZm.setCheckable(False)
696         self.pushZm.setChecked(False)
697         self.pushZm.setObjectName("pushZm")
698         self.pushZm.setAutoRepeat(True)
699         self.pushZm.setAutoRepeatDelay(PUSH_DELAY)
700         self.pushZm.setAutoRepeatInterval(PUSH_DELAY)
701         self.gridLayout.addWidget(self.pushZm, 0, 1, 1, 1, QtCore.
    Qt.AlignRight)
702         self.pushZp = QtWidgets.QPushButton(self.tab)
703         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Minimum, QtWidgets.QSizePolicy.Minimum)
704         sizePolicy.setHorizontalStretch(0)
705         sizePolicy.setVerticalStretch(0)
706         sizePolicy.setHeightForWidth(self.pushZp.sizePolicy().
    hasHeightForWidth())
707         self.pushZp.setSizePolicy(sizePolicy)
708         self.pushZp.setText("")
709         icon5 = QtGui.QIcon()
710         icon5.addPixmap(QtGui.QPixmap("/home/marc/hyrecro_ws/src/
    gui_hyrecro/gui_hyrecro/iconos/arrow-down-left.svg"), QtGui.
    QIcon.Normal, QtGui.QIcon.Off)
711         self.pushZp.setIcon(icon5)
712         self.pushZp.setCheckable(False)
713         self.pushZp.setChecked(False)
714         self.pushZp.setObjectName("pushZp")
715         self.pushZp.setAutoRepeat(True)
716         self.pushZp.setAutoRepeatDelay(PUSH_DELAY)
717         self.pushZp.setAutoRepeatInterval(PUSH_DELAY)
718         self.gridLayout.addWidget(self.pushZp, 1, 0, 1, 1, QtCore.
    Qt.AlignLeft)
719         self.label_22 = QtWidgets.QLabel(self.tab)
720         self.label_22.setObjectName("label_22")
721         self.gridLayout.addWidget(self.label_22, 1, 1, 1, 1, QtCore
    .Qt.AlignLeft)
722         self.gridLayout_2.addLayout(self.gridLayout, 1, 1, 1, 1)
723         self.gridLayout_3.addLayout(self.gridLayout_2, 1, 1, 1, 1)
724         self.label_26 = QtWidgets.QLabel(self.tab)
725         self.label_26.setObjectName("label_26")
726         self.gridLayout_3.addWidget(self.label_26, 1, 2, 1, 1)
727         self.label_25 = QtWidgets.QLabel(self.tab)
728         self.label_25.setObjectName("label_25")
729         self.gridLayout_3.addWidget(self.label_25, 2, 1, 1, 1,
    QtCore.Qt.AlignHCenter|QtCore.Qt.AlignTop)
730         self.gridLayout_3.setRowStretch(1, 1)
731         self.verticalLayout_26.addLayout(self.gridLayout_3)
732         self.horizontalLayout_2.addLayout(self.verticalLayout_26)
733         self.verticalLayout_27 = QtWidgets.QVBoxLayout()
734         self.verticalLayout_27.setObjectName("verticalLayout_27")
735         self.label_29 = QtWidgets.QLabel(self.tab)
736         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Preferred, QtWidgets.QSizePolicy.Maximum)
```

```
737         sizePolicy.setHorizontalStretch(0)
738         sizePolicy.setVerticalStretch(0)
739         sizePolicy.setHeightForWidth(self.label_29.sizePolicy().
    hasHeightForWidth())
740         self.label_29.setSizePolicy(sizePolicy)
741         self.label_29.setObjectName("label_29")
742         self.verticalLayout_27.addWidget(self.label_29)
743         self.gridLayout_4 = QtWidgets.QGridLayout()
744         self.gridLayout_4.setObjectName("gridLayout_4")
745         self.label_30 = QtWidgets.QLabel(self.tab)
746         self.label_30.setObjectName("label_30")
747         self.gridLayout_4.addWidget(self.label_30, 0, 1, 1, 1,
    QtCore.Qt.AlignHCenter|QtCore.Qt.AlignBottom)
748         self.label_31 = QtWidgets.QLabel(self.tab)
749         self.label_31.setObjectName("label_31")
750         self.gridLayout_4.addWidget(self.label_31, 1, 0, 1, 1,
    QtCore.Qt.AlignRight)
751         self.gridLayout_5 = QtWidgets.QGridLayout()
752         self.gridLayout_5.setObjectName("gridLayout_5")
753         self.pushGYp = QtWidgets.QPushButton(self.tab)
754         self.pushGYp.setText("")
755         icon6 = QtGui.QIcon()
756         icon6.addPixmap(QtGui.QPixmap("/home/marc/hyrecro_ws/src/
    gui_hyrecro/gui_hyrecro/iconos/rotate-ccw_tumbado.svg"), QtGui.
    QIcon.Normal, QtGui.QIcon.Off)
757         self.pushGYp.setIcon(icon6)
758         self.pushGYp.setCheckable(False)
759         self.pushGYp.setChecked(False)
760         self.pushGYp.setObjectName("pushGYp")
761         self.pushGYp.setAutoRepeat(True)
762         self.pushGYp.setAutoRepeatDelay(PUSH_DELAY)
763         self.pushGYp.setAutoRepeatInterval(PUSH_DELAY)
764         self.gridLayout_5.addWidget(self.pushGYp, 0, 1, 1, 1)
765         self.pushGXm = QtWidgets.QPushButton(self.tab)
766         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Fixed, QtWidgets.QSizePolicy.Minimum)
767         sizePolicy.setHorizontalStretch(0)
768         sizePolicy.setVerticalStretch(0)
769         sizePolicy.setHeightForWidth(self.pushGXm.sizePolicy().
    hasHeightForWidth())
770         self.pushGXm.setSizePolicy(sizePolicy)
771         self.pushGXm.setText("")
772         icon7 = QtGui.QIcon()
773         icon7.addPixmap(QtGui.QPixmap("/home/marc/hyrecro_ws/src/
    gui_hyrecro/gui_hyrecro/iconos/rotate-cw_lado.svg"), QtGui.
    QIcon.Normal, QtGui.QIcon.Off)
774         self.pushGXm.setIcon(icon7)
775         self.pushGXm.setCheckable(False)
776         self.pushGXm.setChecked(False)
777         self.pushGXm.setObjectName("pushGXm")
778         self.pushGXm.setAutoRepeat(True)
779         self.pushGXm.setAutoRepeatDelay(PUSH_DELAY)
780         self.pushGXm.setAutoRepeatInterval(PUSH_DELAY)
781         self.gridLayout_5.addWidget(self.pushGXm, 1, 0, 1, 1)
782         self.pushGXp = QtWidgets.QPushButton(self.tab)
783         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Fixed, QtWidgets.QSizePolicy.Minimum)
```

```
784         sizePolicy.setHorizontalStretch(0)
785         sizePolicy.setVerticalStretch(0)
786         sizePolicy.setHeightForWidth(self.pushGXp.sizePolicy().
    hasHeightForWidth())
787         self.pushGXp.setSizePolicy(sizePolicy)
788         self.pushGXp.setText("")
789         icon8 = QtGui.QIcon()
790         icon8.addPixmap(QtGui.QPixmap("/home/marc/hyrecro_ws/src/
    gui_hyrecro/gui_hyrecro/iconos/rotate-ccw_ladp.svg"), QtGui.
    QIcon.Normal, QtGui.QIcon.Off)
791         self.pushGXp.setIcon(icon8)
792         self.pushGXp.setCheckable(False)
793         self.pushGXp.setChecked(False)
794         self.pushGXp.setObjectName("pushGXp")
795         self.pushGXp.setAutoRepeat(True)
796         self.pushGXp.setAutoRepeatDelay(PUSH_DELAY)
797         self.pushGXp.setAutoRepeatInterval(PUSH_DELAY)
798         self.gridLayout_5.addWidget(self.pushGXp, 1, 2, 1, 1)
799         self.pushGYm = QtWidgets.QPushButton(self.tab)
800         self.pushGYm.setText("")
801         icon9 = QtGui.QIcon()
802         icon9.addPixmap(QtGui.QPixmap("/home/marc/hyrecro_ws/src/
    gui_hyrecro/gui_hyrecro/iconos/rotate-cw_tumbado.svg"), QtGui.
    QIcon.Normal, QtGui.QIcon.Off)
803         self.pushGYm.setIcon(icon9)
804         self.pushGYm.setCheckable(False)
805         self.pushGYm.setChecked(False)
806         self.pushGYm.setObjectName("pushGYm")
807         self.pushGYm.setAutoRepeat(True)
808         self.pushGYm.setAutoRepeatDelay(PUSH_DELAY)
809         self.pushGYm.setAutoRepeatInterval(PUSH_DELAY)
810         self.gridLayout_5.addWidget(self.pushGYm, 2, 1, 1, 1)
811         self.gridLayout_6 = QtWidgets.QGridLayout()
812         self.gridLayout_6.setObjectName("gridLayout_6")
813         self.label_32 = QtWidgets.QLabel(self.tab)
814         self.label_32.setObjectName("label_32")
815         self.gridLayout_6.addWidget(self.label_32, 0, 0, 1, 1,
    QtCore.Qt.AlignRight)
816         self.pushGZm = QtWidgets.QPushButton(self.tab)
817         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Fixed, QtWidgets.QSizePolicy.Minimum)
818         sizePolicy.setHorizontalStretch(0)
819         sizePolicy.setVerticalStretch(0)
820         sizePolicy.setHeightForWidth(self.pushGZm.sizePolicy().
    hasHeightForWidth())
821         self.pushGZm.setSizePolicy(sizePolicy)
822         self.pushGZm.setText("")
823         icon10 = QtGui.QIcon()
824         icon10.addPixmap(QtGui.QPixmap("/home/marc/hyrecro_ws/src/
    gui_hyrecro/gui_hyrecro/iconos/rotate-cw.svg"), QtGui.QIcon.
    Normal, QtGui.QIcon.Off)
825         self.pushGZm.setIcon(icon10)
826         self.pushGZm.setCheckable(False)
827         self.pushGZm.setChecked(False)
828         self.pushGZm.setObjectName("pushGZm")
829         self.pushGZm.setAutoRepeat(True)
830         self.pushGZm.setAutoRepeatDelay(PUSH_DELAY)
```

```
831         self.pushGZm.setAutoRepeatInterval(PUSH_DELAY)
832         self.gridLayout_6.addWidget(self.pushGZm, 0, 1, 1, 1,
    QtCore.Qt.AlignRight)
833         self.pushGZp = QtWidgets.QPushButton(self.tab)
834         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Fixed, QtWidgets.QSizePolicy.Minimum)
835         sizePolicy.setHorizontalStretch(0)
836         sizePolicy.setVerticalStretch(0)
837         sizePolicy.setHeightForWidth(self.pushGZp.sizePolicy().
    hasHeightForWidth())
838         self.pushGZp.setSizePolicy(sizePolicy)
839         self.pushGZp.setText("")
840         icon11 = QtGui.QIcon()
841         icon11.addPixmap(QtGui.QPixmap("/home/marc/hyrecro_ws/src/
    gui_hyrecro/gui_hyrecro/iconos/rotate-ccw.svg"), QtGui.QIcon.
    Normal, QtGui.QIcon.Off)
842         self.pushGZp.setIcon(icon11)
843         self.pushGZp.setCheckable(False)
844         self.pushGZp.setChecked(False)
845         self.pushGZp.setObjectName("pushGZp")
846         self.pushGZp.setAutoRepeat(True)
847         self.pushGZp.setAutoRepeatDelay(PUSH_DELAY)
848         self.pushGZp.setAutoRepeatInterval(PUSH_DELAY)
849         self.gridLayout_6.addWidget(self.pushGZp, 1, 0, 1, 1,
    QtCore.Qt.AlignLeft)
850         self.label_33 = QtWidgets.QLabel(self.tab)
851         self.label_33.setObjectName("label_33")
852         self.gridLayout_6.addWidget(self.label_33, 1, 1, 1, 1,
    QtCore.Qt.AlignLeft)
853         self.gridLayout_5.addLayout(self.gridLayout_6, 1, 1, 1, 1)
854         self.gridLayout_4.addLayout(self.gridLayout_5, 1, 1, 1, 1)
855         self.label_34 = QtWidgets.QLabel(self.tab)
856         self.label_34.setObjectName("label_34")
857         self.gridLayout_4.addWidget(self.label_34, 1, 2, 1, 1)
858         self.label_35 = QtWidgets.QLabel(self.tab)
859         self.label_35.setObjectName("label_35")
860         self.gridLayout_4.addWidget(self.label_35, 2, 1, 1, 1,
    QtCore.Qt.AlignHCenter|QtCore.Qt.AlignTop)
861         self.gridLayout_4.setRowStretch(1, 1)
862         self.verticalLayout_27.addLayout(self.gridLayout_4)
863         self.horizontalLayout_2.addLayout(self.verticalLayout_27)
864         self.layout_incrementos = QtWidgets.QVBoxLayout()
865         self.layout_incrementos.setObjectName("layout_incrementos")
866         self.label_incrementos_1 = QtWidgets.QLabel(self.tab)
867         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Preferred, QtWidgets.QSizePolicy.Maximum)
868         sizePolicy.setHorizontalStretch(0)
869         sizePolicy.setVerticalStretch(0)
870         sizePolicy.setHeightForWidth(self.label_incrementos_1.
    sizePolicy().hasHeightForWidth())
871         self.label_incrementos_1.setSizePolicy(sizePolicy)
872         self.label_incrementos_1.setObjectName("label_incrementos_1
    ")
873         self.layout_incrementos.addWidget(self.label_incrementos_1,
     0, QtCore.Qt.AlignHCenter)
874         self.label_incrementos_2 = QtWidgets.QLabel(self.tab)
```

```
875         self.label_incrementos_2.setObjectName("label_incrementos_2
    ")
876         self.layout_incrementos.addWidget(self.label_incrementos_2,
     0, QtCore.Qt.AlignHCenter)
877         self.boxDeltaPos = QtWidgets.QDoubleSpinBox(self.tab)
878         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Minimum, QtWidgets.QSizePolicy.Fixed)
879         sizePolicy.setHorizontalStretch(0)
880         sizePolicy.setVerticalStretch(0)
881         sizePolicy.setHeightForWidth(self.boxDeltaPos.sizePolicy().
    hasHeightForWidth())
882         self.boxDeltaPos.setSizePolicy(sizePolicy)
883         self.boxDeltaPos.setMinimumSize(QtCore.QSize(0, 0))
884         self.boxDeltaPos.setAlignment(QtCore.Qt.AlignCenter)
885         self.boxDeltaPos.setDecimals(4)
886         self.boxDeltaPos.setSingleStep(0.0001)
887         self.boxDeltaPos.setProperty("value", 0.001)
888         self.boxDeltaPos.setObjectName("boxDeltaPos")
889         self.layout_incrementos.addWidget(self.boxDeltaPos)
890         self.line_incrementos = QtWidgets.QFrame(self.tab)
891         self.line_incrementos.setFrameShape(QtWidgets.QFrame.HLine)
892         self.line_incrementos.setFrameShadow(QtWidgets.QFrame.
    Sunken)
893         self.line_incrementos.setObjectName("line_incrementos")
894         self.layout_incrementos.addWidget(self.line_incrementos)
895         self.label_incrementos_3 = QtWidgets.QLabel(self.tab)
896         self.label_incrementos_3.setObjectName("label_incrementos_3
    ")
897         self.layout_incrementos.addWidget(self.label_incrementos_3,
     0, QtCore.Qt.AlignHCenter)
898         self.boxDeltaOr = QtWidgets.QDoubleSpinBox(self.tab)
899         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Minimum, QtWidgets.QSizePolicy.Fixed)
900         sizePolicy.setHorizontalStretch(0)
901         sizePolicy.setVerticalStretch(0)
902         sizePolicy.setHeightForWidth(self.boxDeltaOr.sizePolicy().
    hasHeightForWidth())
903         self.boxDeltaOr.setSizePolicy(sizePolicy)
904         self.boxDeltaOr.setMinimumSize(QtCore.QSize(0, 0))
905         self.boxDeltaOr.setAlignment(QtCore.Qt.AlignCenter)
906         self.boxDeltaOr.setDecimals(4)
907         self.boxDeltaOr.setSingleStep(0.0001)
908         self.boxDeltaOr.setProperty("value", 0.001)
909         self.boxDeltaOr.setObjectName("boxDeltaOr")
910         self.layout_incrementos.addWidget(self.boxDeltaOr)
911         self.horizontalLayout_2.addLayout(self.layout_incrementos)
912         self.horizontalLayout_2.setStretch(0, 1)
913         self.horizontalLayout_2.setStretch(1, 1)
914         self.grupo_enviar.addTab(self.tab, "")
915         self.verticalLayout.addWidget(self.grupo_enviar)
916         self.line_5 = QtWidgets.QFrame(self.centralwidget)
917         self.line_5.setFrameShape(QtWidgets.QFrame.HLine)
918         self.line_5.setFrameShadow(QtWidgets.QFrame.Sunken)
919         self.line_5.setObjectName("line_5")
920         self.verticalLayout.addWidget(self.line_5)
921         self.groupBox = QtWidgets.QGroupBox(self.centralwidget)
```

```
922         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Minimum, QtWidgets.QSizePolicy.Preferred)
923         sizePolicy.setHorizontalStretch(0)
924         sizePolicy.setVerticalStretch(0)
925         sizePolicy.setHeightForWidth(self.groupBox.sizePolicy().
    hasHeightForWidth())
926         self.groupBox.setSizePolicy(sizePolicy)
927         self.groupBox.setObjectName("groupBox")
928         self.horizontalLayout_5 = QtWidgets.QHBoxLayout(self.
    groupBox)
929         self.horizontalLayout_5.setObjectName("horizontalLayout_5")
930         self.verticalLayout_28 = QtWidgets.QVBoxLayout()
931         self.verticalLayout_28.setSpacing(0)
932         self.verticalLayout_28.setObjectName("verticalLayout_28")
933         self.gridLayout_9 = QtWidgets.QGridLayout()
934         self.gridLayout_9.setSpacing(0)
935         self.gridLayout_9.setObjectName("gridLayout_9")
936         self.act11 = QtWidgets.QDoubleSpinBox(self.groupBox)
937         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
938         sizePolicy.setHorizontalStretch(0)
939         sizePolicy.setVerticalStretch(0)
940         sizePolicy.setHeightForWidth(self.act11.sizePolicy().
    hasHeightForWidth())
941         self.act11.setSizePolicy(sizePolicy)
942         self.act11.setMinimumSize(QtCore.QSize(48, 48))
943         self.act11.setMaximumSize(QtCore.QSize(48, 48))
944         self.act11.setSizeIncrement(QtCore.QSize(0, 0))
945         self.act11.setWrapping(False)
946         self.act11.setAlignment(QtCore.Qt.AlignCenter)
947         self.act11.setReadOnly(True)
948         self.act11.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
949         self.act11.setKeyboardTracking(True)
950         self.act11.setDecimals(2)
951         self.act11.setMaximum(1.0)
952         self.act11.setProperty("value", 1.0)
953         self.act11.setObjectName("act11")
954         self.gridLayout_9.addWidget(self.act11, 0, 0, 1, 1)
955         self.act12 = QtWidgets.QDoubleSpinBox(self.groupBox)
956         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
957         sizePolicy.setHorizontalStretch(0)
958         sizePolicy.setVerticalStretch(0)
959         sizePolicy.setHeightForWidth(self.act12.sizePolicy().
    hasHeightForWidth())
960         self.act12.setSizePolicy(sizePolicy)
961         self.act12.setMinimumSize(QtCore.QSize(48, 48))
962         self.act12.setMaximumSize(QtCore.QSize(48, 48))
963         self.act12.setSizeIncrement(QtCore.QSize(0, 0))
964         self.act12.setWrapping(False)
965         self.act12.setAlignment(QtCore.Qt.AlignCenter)
966         self.act12.setReadOnly(True)
967         self.act12.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
968         self.act12.setKeyboardTracking(True)
969         self.act12.setDecimals(2)
```

```
970         self.act12.setMaximum(1.0)
971         self.act12.setObjectName("act12")
972         self.gridLayout_9.addWidget(self.act12, 0, 1, 1, 1)
973         self.act13 = QtWidgets.QDoubleSpinBox(self.groupBox)
974         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
975         sizePolicy.setHorizontalStretch(0)
976         sizePolicy.setVerticalStretch(0)
977         sizePolicy.setHeightForWidth(self.act13.sizePolicy().
    hasHeightForWidth())
978         self.act13.setSizePolicy(sizePolicy)
979         self.act13.setMinimumSize(QtCore.QSize(48, 48))
980         self.act13.setMaximumSize(QtCore.QSize(48, 48))
981         self.act13.setSizeIncrement(QtCore.QSize(0, 0))
982         self.act13.setWrapping(False)
983         self.act13.setAlignment(QtCore.Qt.AlignCenter)
984         self.act13.setReadOnly(True)
985         self.act13.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
986         self.act13.setKeyboardTracking(True)
987         self.act13.setDecimals(2)
988         self.act13.setMaximum(1.0)
989         self.act13.setObjectName("act13")
990         self.gridLayout_9.addWidget(self.act13, 0, 2, 1, 1)
991         self.act14 = QtWidgets.QDoubleSpinBox(self.groupBox)
992         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
993         sizePolicy.setHorizontalStretch(0)
994         sizePolicy.setVerticalStretch(0)
995         sizePolicy.setHeightForWidth(self.act14.sizePolicy().
    hasHeightForWidth())
996         self.act14.setSizePolicy(sizePolicy)
997         self.act14.setMinimumSize(QtCore.QSize(48, 48))
998         self.act14.setMaximumSize(QtCore.QSize(48, 48))
999         self.act14.setSizeIncrement(QtCore.QSize(0, 0))
1000        self.act14.setWrapping(False)
1001        self.act14.setAlignment(QtCore.Qt.AlignCenter)
1002        self.act14.setReadOnly(True)
1003        self.act14.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1004        self.act14.setKeyboardTracking(True)
1005        self.act14.setDecimals(2)
1006        self.act14.setMaximum(1000.0)
1007        self.act14.setObjectName("act14")
1008        self.gridLayout_9.addWidget(self.act14, 0, 3, 1, 1)
1009        self.act21 = QtWidgets.QDoubleSpinBox(self.groupBox)
1010        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
1011        sizePolicy.setHorizontalStretch(0)
1012        sizePolicy.setVerticalStretch(0)
1013        sizePolicy.setHeightForWidth(self.act21.sizePolicy().
    hasHeightForWidth())
1014        self.act21.setSizePolicy(sizePolicy)
1015        self.act21.setMinimumSize(QtCore.QSize(48, 48))
1016        self.act21.setMaximumSize(QtCore.QSize(48, 48))
1017        self.act21.setSizeIncrement(QtCore.QSize(0, 0))
1018        self.act21.setWrapping(False)
```

```
1019            self.act21.setAlignment(QtCore.Qt.AlignCenter)
1020            self.act21.setReadOnly(True)
1021            self.act21.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1022            self.act21.setKeyboardTracking(True)
1023            self.act21.setDecimals(2)
1024            self.act21.setMaximum(1.0)
1025            self.act21.setObjectName("act21")
1026            self.gridLayout_9.addWidget(self.act21, 1, 0, 1, 1)
1027            self.act22 = QtWidgets.QDoubleSpinBox(self.groupBox)
1028            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
1029            sizePolicy.setHorizontalStretch(0)
1030            sizePolicy.setVerticalStretch(0)
1031            sizePolicy.setHeightForWidth(self.act22.sizePolicy().
    hasHeightForWidth())
1032            self.act22.setSizePolicy(sizePolicy)
1033            self.act22.setMinimumSize(QtCore.QSize(48, 48))
1034            self.act22.setMaximumSize(QtCore.QSize(48, 48))
1035            self.act22.setSizeIncrement(QtCore.QSize(0, 0))
1036            self.act22.setWrapping(False)
1037            self.act22.setAlignment(QtCore.Qt.AlignCenter)
1038            self.act22.setReadOnly(True)
1039            self.act22.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1040            self.act22.setKeyboardTracking(True)
1041            self.act22.setDecimals(2)
1042            self.act22.setMaximum(1.0)
1043            self.act22.setProperty("value", 1.0)
1044            self.act22.setObjectName("act22")
1045            self.gridLayout_9.addWidget(self.act22, 1, 1, 1, 1)
1046            self.act23 = QtWidgets.QDoubleSpinBox(self.groupBox)
1047            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
1048            sizePolicy.setHorizontalStretch(0)
1049            sizePolicy.setVerticalStretch(0)
1050            sizePolicy.setHeightForWidth(self.act23.sizePolicy().
    hasHeightForWidth())
1051            self.act23.setSizePolicy(sizePolicy)
1052            self.act23.setMinimumSize(QtCore.QSize(48, 48))
1053            self.act23.setMaximumSize(QtCore.QSize(48, 48))
1054            self.act23.setSizeIncrement(QtCore.QSize(0, 0))
1055            self.act23.setWrapping(False)
1056            self.act23.setAlignment(QtCore.Qt.AlignCenter)
1057            self.act23.setReadOnly(True)
1058            self.act23.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1059            self.act23.setKeyboardTracking(True)
1060            self.act23.setDecimals(2)
1061            self.act23.setMaximum(1.0)
1062            self.act23.setObjectName("act23")
1063            self.gridLayout_9.addWidget(self.act23, 1, 2, 1, 1)
1064            self.act24 = QtWidgets.QDoubleSpinBox(self.groupBox)
1065            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
1066            sizePolicy.setHorizontalStretch(0)
1067            sizePolicy.setVerticalStretch(0)
```

```
1068            sizePolicy.setHeightForWidth(self.act24.sizePolicy().
    hasHeightForWidth())
1069            self.act24.setSizePolicy(sizePolicy)
1070            self.act24.setMinimumSize(QtCore.QSize(48, 48))
1071            self.act24.setMaximumSize(QtCore.QSize(48, 48))
1072            self.act24.setSizeIncrement(QtCore.QSize(0, 0))
1073            self.act24.setWrapping(False)
1074            self.act24.setAlignment(QtCore.Qt.AlignCenter)
1075            self.act24.setReadOnly(True)
1076            self.act24.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1077            self.act24.setKeyboardTracking(True)
1078            self.act24.setDecimals(2)
1079            self.act24.setMaximum(1000.0)
1080            self.act24.setObjectName("act24")
1081            self.gridLayout_9.addWidget(self.act24, 1, 3, 1, 1)
1082            self.act31 = QtWidgets.QDoubleSpinBox(self.groupBox)
1083            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
1084            sizePolicy.setHorizontalStretch(0)
1085            sizePolicy.setVerticalStretch(0)
1086            sizePolicy.setHeightForWidth(self.act31.sizePolicy().
    hasHeightForWidth())
1087            self.act31.setSizePolicy(sizePolicy)
1088            self.act31.setMinimumSize(QtCore.QSize(48, 48))
1089            self.act31.setMaximumSize(QtCore.QSize(48, 48))
1090            self.act31.setSizeIncrement(QtCore.QSize(0, 0))
1091            self.act31.setWrapping(False)
1092            self.act31.setAlignment(QtCore.Qt.AlignCenter)
1093            self.act31.setReadOnly(True)
1094            self.act31.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1095            self.act31.setKeyboardTracking(True)
1096            self.act31.setDecimals(2)
1097            self.act31.setMaximum(1.0)
1098            self.act31.setObjectName("act31")
1099            self.gridLayout_9.addWidget(self.act31, 2, 0, 1, 1)
1100            self.act32 = QtWidgets.QDoubleSpinBox(self.groupBox)
1101            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
1102            sizePolicy.setHorizontalStretch(0)
1103            sizePolicy.setVerticalStretch(0)
1104            sizePolicy.setHeightForWidth(self.act32.sizePolicy().
    hasHeightForWidth())
1105            self.act32.setSizePolicy(sizePolicy)
1106            self.act32.setMinimumSize(QtCore.QSize(48, 48))
1107            self.act32.setMaximumSize(QtCore.QSize(48, 48))
1108            self.act32.setSizeIncrement(QtCore.QSize(0, 0))
1109            self.act32.setWrapping(False)
1110            self.act32.setAlignment(QtCore.Qt.AlignCenter)
1111            self.act32.setReadOnly(True)
1112            self.act32.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1113            self.act32.setKeyboardTracking(True)
1114            self.act32.setDecimals(2)
1115            self.act32.setMaximum(1.0)
1116            self.act32.setObjectName("act32")
```

```
1117            self.gridLayout_9.addWidget(self.act32, 2, 1, 1, 1)
1118            self.act33 = QtWidgets.QDoubleSpinBox(self.groupBox)
1119            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
1120            sizePolicy.setHorizontalStretch(0)
1121            sizePolicy.setVerticalStretch(0)
1122            sizePolicy.setHeightForWidth(self.act33.sizePolicy().
    hasHeightForWidth())
1123            self.act33.setSizePolicy(sizePolicy)
1124            self.act33.setMinimumSize(QtCore.QSize(48, 48))
1125            self.act33.setMaximumSize(QtCore.QSize(48, 48))
1126            self.act33.setSizeIncrement(QtCore.QSize(0, 0))
1127            self.act33.setWrapping(False)
1128            self.act33.setAlignment(QtCore.Qt.AlignCenter)
1129            self.act33.setReadOnly(True)
1130            self.act33.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1131            self.act33.setKeyboardTracking(True)
1132            self.act33.setDecimals(2)
1133            self.act33.setMaximum(1.0)
1134            self.act33.setProperty("value", 1.0)
1135            self.act33.setObjectName("act33")
1136            self.gridLayout_9.addWidget(self.act33, 2, 2, 1, 1)
1137            self.act34 = QtWidgets.QDoubleSpinBox(self.groupBox)
1138            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
1139            sizePolicy.setHorizontalStretch(0)
1140            sizePolicy.setVerticalStretch(0)
1141            sizePolicy.setHeightForWidth(self.act34.sizePolicy().
    hasHeightForWidth())
1142            self.act34.setSizePolicy(sizePolicy)
1143            self.act34.setMinimumSize(QtCore.QSize(48, 48))
1144            self.act34.setMaximumSize(QtCore.QSize(48, 48))
1145            self.act34.setSizeIncrement(QtCore.QSize(0, 0))
1146            self.act34.setWrapping(False)
1147            self.act34.setAlignment(QtCore.Qt.AlignCenter)
1148            self.act34.setReadOnly(True)
1149            self.act34.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1150            self.act34.setKeyboardTracking(True)
1151            self.act34.setDecimals(2)
1152            self.act34.setMaximum(1000.0)
1153            self.act34.setObjectName("act34")
1154            self.gridLayout_9.addWidget(self.act34, 2, 3, 1, 1)
1155            self.act41 = QtWidgets.QDoubleSpinBox(self.groupBox)
1156            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
1157            sizePolicy.setHorizontalStretch(0)
1158            sizePolicy.setVerticalStretch(0)
1159            sizePolicy.setHeightForWidth(self.act41.sizePolicy().
    hasHeightForWidth())
1160            self.act41.setSizePolicy(sizePolicy)
1161            self.act41.setMinimumSize(QtCore.QSize(48, 48))
1162            self.act41.setMaximumSize(QtCore.QSize(48, 48))
1163            self.act41.setSizeIncrement(QtCore.QSize(0, 0))
1164            self.act41.setWrapping(False)
1165            self.act41.setAlignment(QtCore.Qt.AlignCenter)
```

```
1166            self.act41.setReadOnly(True)
1167            self.act41.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1168            self.act41.setKeyboardTracking(True)
1169            self.act41.setDecimals(2)
1170            self.act41.setMaximum(0.0)
1171            self.act41.setObjectName("act41")
1172            self.gridLayout_9.addWidget(self.act41, 3, 0, 1, 1)
1173            self.act42 = QtWidgets.QDoubleSpinBox(self.groupBox)
1174            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
1175            sizePolicy.setHorizontalStretch(0)
1176            sizePolicy.setVerticalStretch(0)
1177            sizePolicy.setHeightForWidth(self.act42.sizePolicy().
    hasHeightForWidth())
1178            self.act42.setSizePolicy(sizePolicy)
1179            self.act42.setMinimumSize(QtCore.QSize(48, 48))
1180            self.act42.setMaximumSize(QtCore.QSize(48, 48))
1181            self.act42.setSizeIncrement(QtCore.QSize(0, 0))
1182            self.act42.setWrapping(False)
1183            self.act42.setAlignment(QtCore.Qt.AlignCenter)
1184            self.act42.setReadOnly(True)
1185            self.act42.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1186            self.act42.setKeyboardTracking(True)
1187            self.act42.setDecimals(2)
1188            self.act42.setMaximum(0.0)
1189            self.act42.setObjectName("act42")
1190            self.gridLayout_9.addWidget(self.act42, 3, 1, 1, 1)
1191            self.act43 = QtWidgets.QDoubleSpinBox(self.groupBox)
1192            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
1193            sizePolicy.setHorizontalStretch(0)
1194            sizePolicy.setVerticalStretch(0)
1195            sizePolicy.setHeightForWidth(self.act43.sizePolicy().
    hasHeightForWidth())
1196            self.act43.setSizePolicy(sizePolicy)
1197            self.act43.setMinimumSize(QtCore.QSize(48, 48))
1198            self.act43.setMaximumSize(QtCore.QSize(48, 48))
1199            self.act43.setSizeIncrement(QtCore.QSize(0, 0))
1200            self.act43.setWrapping(False)
1201            self.act43.setAlignment(QtCore.Qt.AlignCenter)
1202            self.act43.setReadOnly(True)
1203            self.act43.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1204            self.act43.setKeyboardTracking(True)
1205            self.act43.setDecimals(2)
1206            self.act43.setMaximum(0.0)
1207            self.act43.setObjectName("act43")
1208            self.gridLayout_9.addWidget(self.act43, 3, 2, 1, 1)
1209            self.act44 = QtWidgets.QDoubleSpinBox(self.groupBox)
1210            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
1211            sizePolicy.setHorizontalStretch(0)
1212            sizePolicy.setVerticalStretch(0)
1213            sizePolicy.setHeightForWidth(self.act44.sizePolicy().
    hasHeightForWidth())
```

```
1214            self.act44.setSizePolicy(sizePolicy)
1215            self.act44.setMinimumSize(QtCore.QSize(48, 48))
1216            self.act44.setMaximumSize(QtCore.QSize(48, 48))
1217            self.act44.setSizeIncrement(QtCore.QSize(0, 0))
1218            self.act44.setWrapping(False)
1219            self.act44.setAlignment(QtCore.Qt.AlignCenter)
1220            self.act44.setReadOnly(True)
1221            self.act44.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1222            self.act44.setKeyboardTracking(True)
1223            self.act44.setDecimals(2)
1224            self.act44.setMinimum(1.0)
1225            self.act44.setMaximum(1.0)
1226            self.act44.setObjectName("act44")
1227            self.gridLayout_9.addWidget(self.act44, 3, 3, 1, 1)
1228            self.verticalLayout_28.addLayout(self.gridLayout_9)
1229            self.line_3 = QtWidgets.QFrame(self.groupBox)
1230            self.line_3.setFrameShape(QtWidgets.QFrame.HLine)
1231            self.line_3.setFrameShadow(QtWidgets.QFrame.Sunken)
1232            self.line_3.setObjectName("line_3")
1233            self.verticalLayout_28.addWidget(self.line_3)
1234            self.label_21 = QtWidgets.QLabel(self.groupBox)
1235            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Minimum, QtWidgets.QSizePolicy.Fixed)
1236            sizePolicy.setHorizontalStretch(0)
1237            sizePolicy.setVerticalStretch(0)
1238            sizePolicy.setHeightForWidth(self.label_21.sizePolicy().
    hasHeightForWidth())
1239            self.label_21.setSizePolicy(sizePolicy)
1240            self.label_21.setObjectName("label_21")
1241            self.verticalLayout_28.addWidget(self.label_21)
1242            self.horizontalLayout_4 = QtWidgets.QHBoxLayout()
1243            self.horizontalLayout_4.setSizeConstraint(QtWidgets.QLayout
    .SetDefaultConstraint)
1244            self.horizontalLayout_4.setSpacing(0)
1245            self.horizontalLayout_4.setObjectName("horizontalLayout_4")
1246            self.radioFijoA = QtWidgets.QRadioButton(self.groupBox)
1247            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Fixed)
1248            sizePolicy.setHorizontalStretch(0)
1249            sizePolicy.setVerticalStretch(0)
1250            sizePolicy.setHeightForWidth(self.radioFijoA.sizePolicy().
    hasHeightForWidth())
1251            self.radioFijoA.setSizePolicy(sizePolicy)
1252            self.radioFijoA.setChecked(True)
1253            self.radioFijoA.setObjectName("radioFijoA")
1254            self.horizontalLayout_4.addWidget(self.radioFijoA, 0,
    QtCore.Qt.AlignHCenter)
1255            self.radiofijoB = QtWidgets.QRadioButton(self.groupBox)
1256            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
1257            sizePolicy.setHorizontalStretch(0)
1258            sizePolicy.setVerticalStretch(0)
1259            sizePolicy.setHeightForWidth(self.radiofijoB.sizePolicy().
    hasHeightForWidth())
1260            self.radiofijoB.setSizePolicy(sizePolicy)
1261            self.radiofijoB.setObjectName("radiofijoB")
```

```
1262            self.horizontalLayout_4.addWidget(self.radiofijoB, 0,
    QtCore.Qt.AlignHCenter)
1263            self.verticalLayout_28.addLayout(self.horizontalLayout_4)
1264            self.radioImanesA = QtWidgets.QRadioButton(self.groupBox)
1265            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
1266            sizePolicy.setHorizontalStretch(0)
1267            sizePolicy.setVerticalStretch(0)
1268            sizePolicy.setHeightForWidth(self.radioImanesA.sizePolicy()
    .hasHeightForWidth())
1269            self.radioImanesA.setSizePolicy(sizePolicy)
1270            self.radioImanesA.setObjectName("radioImanesA")
1271            self.verticalLayout_28.addWidget(self.radioImanesA)
1272            self.radioImanesB = QtWidgets.QRadioButton(self.groupBox)
1273            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Maximum, QtWidgets.QSizePolicy.Maximum)
1274            sizePolicy.setHorizontalStretch(0)
1275            sizePolicy.setVerticalStretch(0)
1276            sizePolicy.setHeightForWidth(self.radioImanesB.sizePolicy()
    .hasHeightForWidth())
1277            self.radioImanesB.setSizePolicy(sizePolicy)
1278            self.radioImanesB.setObjectName("radioImanesB")
1279            self.verticalLayout_28.addWidget(self.radioImanesB)
1280            self.horizontalLayout_5.addLayout(self.verticalLayout_28)
1281            self.line_2 = QtWidgets.QFrame(self.groupBox)
1282            self.line_2.setFrameShape(QtWidgets.QFrame.VLine)
1283            self.line_2.setFrameShadow(QtWidgets.QFrame.Sunken)
1284            self.line_2.setObjectName("line_2")
1285            self.horizontalLayout_5.addWidget(self.line_2)
1286            self.verticalLayout_15 = QtWidgets.QVBoxLayout()
1287            self.verticalLayout_15.setObjectName("verticalLayout_15")
1288            self.horizontalLayout_3 = QtWidgets.QHBoxLayout()
1289            self.horizontalLayout_3.setObjectName("horizontalLayout_3")
1290            self.verticalLayout_16 = QtWidgets.QVBoxLayout()
1291            self.verticalLayout_16.setObjectName("verticalLayout_16")
1292            self.label_11 = QtWidgets.QLabel(self.groupBox)
1293            self.label_11.setObjectName("label_11")
1294            self.verticalLayout_16.addWidget(self.label_11, 0, QtCore.
    Qt.AlignHCenter)
1295            self.sliderR1A = ReadOnlyDoubleSlider(self.groupBox)
1296            self.sliderR1A.setMinimum(102.5)
1297            self.sliderR1A.setMaximum(152)
1298            self.sliderR1A.setOrientation(QtCore.Qt.Vertical)
1299            self.sliderR1A.setObjectName("sliderR1A")
1300            self.verticalLayout_16.addWidget(self.sliderR1A, 0, QtCore.
    Qt.AlignHCenter)
1301            self.boxR1A = QtWidgets.QDoubleSpinBox(self.groupBox)
1302            self.boxR1A.setAlignment(QtCore.Qt.AlignCenter)
1303            self.boxR1A.setReadOnly(True)
1304            self.boxR1A.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1305            self.boxR1A.setMinimum(102.5)
1306            self.boxR1A.setMaximum(152.0)
1307            self.boxR1A.setProperty("value", 102.5)
1308            self.boxR1A.setObjectName("boxR1A")
1309            self.verticalLayout_16.addWidget(self.boxR1A)
1310            self.horizontalLayout_3.addLayout(self.verticalLayout_16)
```

```
1311          self.verticalLayout_17 = QtWidgets.QVBoxLayout()
1312          self.verticalLayout_17.setObjectName("verticalLayout_17")
1313          self.label_12 = QtWidgets.QLabel(self.groupBox)
1314          self.label_12.setObjectName("label_12")
1315          self.verticalLayout_17.addWidget(self.label_12, 0, QtCore.
      Qt.AlignHCenter)
1316          self.sliderL1B = ReadOnlyDoubleSlider(self.groupBox)
1317          self.sliderL1B.setMinimum(102.5)
1318          self.sliderL1B.setMaximum(152)
1319          self.sliderL1B.setOrientation(QtCore.Qt.Vertical)
1320          self.sliderL1B.setObjectName("sliderL1B")
1321          self.verticalLayout_17.addWidget(self.sliderL1B, 0, QtCore.
      Qt.AlignHCenter)
1322          self.boxL1B = QtWidgets.QDoubleSpinBox(self.groupBox)
1323          self.boxL1B.setAlignment(QtCore.Qt.AlignCenter)
1324          self.boxL1B.setReadOnly(True)
1325          self.boxL1B.setButtonSymbols(QtWidgets.QAbstractSpinBox.
      NoButtons)
1326          self.boxL1B.setMinimum(102.5)
1327          self.boxL1B.setMaximum(152.0)
1328          self.boxL1B.setProperty("value", 102.5)
1329          self.boxL1B.setObjectName("boxL1B")
1330          self.verticalLayout_17.addWidget(self.boxL1B)
1331          self.horizontalLayout_3.addLayout(self.verticalLayout_17)
1332          self.verticalLayout_18 = QtWidgets.QVBoxLayout()
1333          self.verticalLayout_18.setObjectName("verticalLayout_18")
1334          self.label_13 = QtWidgets.QLabel(self.groupBox)
1335          self.label_13.setObjectName("label_13")
1336          self.verticalLayout_18.addWidget(self.label_13, 0, QtCore.
      Qt.AlignHCenter)
1337          self.sliderR2A = ReadOnlyDoubleSlider(self.groupBox)
1338          self.sliderR2A.setMinimum(102.5)
1339          self.sliderR2A.setMaximum(152)
1340          self.sliderR2A.setOrientation(QtCore.Qt.Vertical)
1341          self.sliderR2A.setObjectName("sliderR2A")
1342          self.verticalLayout_18.addWidget(self.sliderR2A, 0, QtCore.
      Qt.AlignHCenter)
1343          self.boxR2A = QtWidgets.QDoubleSpinBox(self.groupBox)
1344          self.boxR2A.setAlignment(QtCore.Qt.AlignCenter)
1345          self.boxR2A.setReadOnly(True)
1346          self.boxR2A.setButtonSymbols(QtWidgets.QAbstractSpinBox.
      NoButtons)
1347          self.boxR2A.setMinimum(102.5)
1348          self.boxR2A.setMaximum(152.0)
1349          self.boxR2A.setProperty("value", 102.5)
1350          self.boxR2A.setObjectName("boxR2A")
1351          self.verticalLayout_18.addWidget(self.boxR2A)
1352          self.horizontalLayout_3.addLayout(self.verticalLayout_18)
1353          self.verticalLayout_19 = QtWidgets.QVBoxLayout()
1354          self.verticalLayout_19.setObjectName("verticalLayout_19")
1355          self.label_14 = QtWidgets.QLabel(self.groupBox)
1356          self.label_14.setObjectName("label_14")
1357          self.verticalLayout_19.addWidget(self.label_14, 0, QtCore.
      Qt.AlignHCenter)
1358          self.sliderL2B = ReadOnlyDoubleSlider(self.groupBox)
1359          self.sliderL2B.setMinimum(102.5)
1360          self.sliderL2B.setMaximum(152)
```

```
1361            self.sliderL2B.setOrientation(QtCore.Qt.Vertical)
1362            self.sliderL2B.setObjectName("sliderL2B")
1363            self.verticalLayout_19.addWidget(self.sliderL2B, 0, QtCore.
    Qt.AlignHCenter)
1364            self.boxL2B = QtWidgets.QDoubleSpinBox(self.groupBox)
1365            self.boxL2B.setAlignment(QtCore.Qt.AlignCenter)
1366            self.boxL2B.setReadOnly(True)
1367            self.boxL2B.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1368            self.boxL2B.setMinimum(102.5)
1369            self.boxL2B.setMaximum(152.0)
1370            self.boxL2B.setProperty("value", 102.5)
1371            self.boxL2B.setObjectName("boxL2B")
1372            self.verticalLayout_19.addWidget(self.boxL2B)
1373            self.horizontalLayout_3.addLayout(self.verticalLayout_19)
1374            self.verticalLayout_20 = QtWidgets.QVBoxLayout()
1375            self.verticalLayout_20.setObjectName("verticalLayout_20")
1376            self.label_15 = QtWidgets.QLabel(self.groupBox)
1377            self.label_15.setObjectName("label_15")
1378            self.verticalLayout_20.addWidget(self.label_15, 0, QtCore.
    Qt.AlignHCenter)
1379            self.sliderL1A = ReadOnlyDoubleSlider(self.groupBox)
1380            self.sliderL1A.setMinimum(102.5)
1381            self.sliderL1A.setMaximum(152)
1382            self.sliderL1A.setOrientation(QtCore.Qt.Vertical)
1383            self.sliderL1A.setObjectName("sliderL1A")
1384            self.verticalLayout_20.addWidget(self.sliderL1A, 0, QtCore.
    Qt.AlignHCenter)
1385            self.boxL1A = QtWidgets.QDoubleSpinBox(self.groupBox)
1386            self.boxL1A.setAlignment(QtCore.Qt.AlignCenter)
1387            self.boxL1A.setReadOnly(True)
1388            self.boxL1A.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1389            self.boxL1A.setMinimum(102.5)
1390            self.boxL1A.setMaximum(152.0)
1391            self.boxL1A.setProperty("value", 102.5)
1392            self.boxL1A.setObjectName("boxL1A")
1393            self.verticalLayout_20.addWidget(self.boxL1A)
1394            self.horizontalLayout_3.addLayout(self.verticalLayout_20)
1395            self.verticalLayout_21 = QtWidgets.QVBoxLayout()
1396            self.verticalLayout_21.setObjectName("verticalLayout_21")
1397            self.label_16 = QtWidgets.QLabel(self.groupBox)
1398            self.label_16.setObjectName("label_16")
1399            self.verticalLayout_21.addWidget(self.label_16, 0, QtCore.
    Qt.AlignHCenter)
1400            self.sliderR1B = ReadOnlyDoubleSlider(self.groupBox)
1401            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Fixed, QtWidgets.QSizePolicy.Expanding)
1402            sizePolicy.setHorizontalStretch(0)
1403            sizePolicy.setVerticalStretch(0)
1404            sizePolicy.setHeightForWidth(self.sliderR1B.sizePolicy().
    hasHeightForWidth())
1405            self.sliderR1B.setSizePolicy(sizePolicy)
1406            self.sliderR1B.setMinimum(102.5)
1407            self.sliderR1B.setMaximum(152)
1408            self.sliderR1B.setOrientation(QtCore.Qt.Vertical)
1409            self.sliderR1B.setObjectName("sliderR1B")
```

```
1410            self.verticalLayout_21.addWidget(self.sliderR1B, 0, QtCore.
    Qt.AlignHCenter)
1411            self.boxR1B = QtWidgets.QDoubleSpinBox(self.groupBox)
1412            sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Minimum, QtWidgets.QSizePolicy.Fixed)
1413            sizePolicy.setHorizontalStretch(0)
1414            sizePolicy.setVerticalStretch(0)
1415            sizePolicy.setHeightForWidth(self.boxR1B.sizePolicy().
    hasHeightForWidth())
1416            self.boxR1B.setSizePolicy(sizePolicy)
1417            self.boxR1B.setAlignment(QtCore.Qt.AlignCenter)
1418            self.boxR1B.setReadOnly(True)
1419            self.boxR1B.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1420            self.boxR1B.setMinimum(102.5)
1421            self.boxR1B.setMaximum(152.0)
1422            self.boxR1B.setProperty("value", 102.5)
1423            self.boxR1B.setObjectName("boxR1B")
1424            self.verticalLayout_21.addWidget(self.boxR1B)
1425            self.horizontalLayout_3.addLayout(self.verticalLayout_21)
1426            self.verticalLayout_22 = QtWidgets.QVBoxLayout()
1427            self.verticalLayout_22.setObjectName("verticalLayout_22")
1428            self.label_17 = QtWidgets.QLabel(self.groupBox)
1429            self.label_17.setObjectName("label_17")
1430            self.verticalLayout_22.addWidget(self.label_17, 0, QtCore.
    Qt.AlignHCenter)
1431            self.sliderL2A = ReadOnlyDoubleSlider(self.groupBox)
1432            self.sliderL2A.setMinimum(102.5)
1433            self.sliderL2A.setMaximum(152)
1434            self.sliderL2A.setOrientation(QtCore.Qt.Vertical)
1435            self.sliderL2A.setObjectName("sliderL2A")
1436            self.verticalLayout_22.addWidget(self.sliderL2A, 0, QtCore.
    Qt.AlignHCenter)
1437            self.boxL2A = QtWidgets.QDoubleSpinBox(self.groupBox)
1438            self.boxL2A.setAlignment(QtCore.Qt.AlignCenter)
1439            self.boxL2A.setReadOnly(True)
1440            self.boxL2A.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1441            self.boxL2A.setMinimum(102.5)
1442            self.boxL2A.setMaximum(152.0)
1443            self.boxL2A.setProperty("value", 102.5)
1444            self.boxL2A.setObjectName("boxL2A")
1445            self.verticalLayout_22.addWidget(self.boxL2A)
1446            self.horizontalLayout_3.addLayout(self.verticalLayout_22)
1447            self.verticalLayout_23 = QtWidgets.QVBoxLayout()
1448            self.verticalLayout_23.setObjectName("verticalLayout_23")
1449            self.label_18 = QtWidgets.QLabel(self.groupBox)
1450            self.label_18.setObjectName("label_18")
1451            self.verticalLayout_23.addWidget(self.label_18, 0, QtCore.
    Qt.AlignHCenter)
1452            self.sliderR2B = ReadOnlyDoubleSlider(self.groupBox)
1453            self.sliderR2B.setMinimum(102.5)
1454            self.sliderR2B.setMaximum(152)
1455            self.sliderR2B.setOrientation(QtCore.Qt.Vertical)
1456            self.sliderR2B.setObjectName("sliderR2B")
1457            self.verticalLayout_23.addWidget(self.sliderR2B, 0, QtCore.
    Qt.AlignHCenter)
```

```
1458            self.boxR2B = QtWidgets.QDoubleSpinBox(self.groupBox)
1459            self.boxR2B.setAlignment(QtCore.Qt.AlignCenter)
1460            self.boxR2B.setReadOnly(True)
1461            self.boxR2B.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1462            self.boxR2B.setMinimum(102.5)
1463            self.boxR2B.setMaximum(152.0)
1464            self.boxR2B.setProperty("value", 102.5)
1465            self.boxR2B.setObjectName("boxR2B")
1466            self.verticalLayout_23.addWidget(self.boxR2B)
1467            self.horizontalLayout_3.addLayout(self.verticalLayout_23)
1468            self.verticalLayout_24 = QtWidgets.QVBoxLayout()
1469            self.verticalLayout_24.setObjectName("verticalLayout_24")
1470            self.label_19 = QtWidgets.QLabel(self.groupBox)
1471            self.label_19.setObjectName("label_19")
1472            self.verticalLayout_24.addWidget(self.label_19, 0, QtCore.
    Qt.AlignHCenter)
1473            self.sliderThetaA = ReadOnlyDoubleSlider(self.groupBox)
1474            self.sliderThetaA.setMinimum(-1.185)
1475            self.sliderThetaA.setMaximum(1.245)
1476            self.sliderThetaA.setOrientation(QtCore.Qt.Vertical)
1477            self.sliderThetaA.setObjectName("sliderThetaA")
1478            self.verticalLayout_24.addWidget(self.sliderThetaA, 0,
    QtCore.Qt.AlignHCenter)
1479            self.boxThetaA = QtWidgets.QDoubleSpinBox(self.groupBox)
1480            self.boxThetaA.setAlignment(QtCore.Qt.AlignCenter)
1481            self.boxThetaA.setReadOnly(True)
1482            self.boxThetaA.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1483            self.boxThetaA.setDecimals(3)
1484            self.boxThetaA.setMinimum(-1.185)
1485            self.boxThetaA.setMaximum(1.245)
1486            self.boxThetaA.setObjectName("boxThetaA")
1487            self.verticalLayout_24.addWidget(self.boxThetaA)
1488            self.horizontalLayout_3.addLayout(self.verticalLayout_24)
1489            self.verticalLayout_25 = QtWidgets.QVBoxLayout()
1490            self.verticalLayout_25.setObjectName("verticalLayout_25")
1491            self.label_20 = QtWidgets.QLabel(self.groupBox)
1492            self.label_20.setObjectName("label_20")
1493            self.verticalLayout_25.addWidget(self.label_20, 0, QtCore.
    Qt.AlignHCenter)
1494            self.sliderThetaB = ReadOnlyDoubleSlider(self.groupBox)
1495            self.sliderThetaB.setMinimum(-1.185)
1496            self.sliderThetaB.setMaximum(1.245)
1497            self.sliderThetaB.setOrientation(QtCore.Qt.Vertical)
1498            self.sliderThetaB.setObjectName("sliderThetaB")
1499            self.verticalLayout_25.addWidget(self.sliderThetaB, 0,
    QtCore.Qt.AlignHCenter)
1500            self.boxThetaB = QtWidgets.QDoubleSpinBox(self.groupBox)
1501            self.boxThetaB.setEnabled(True)
1502            self.boxThetaB.setAlignment(QtCore.Qt.AlignCenter)
1503            self.boxThetaB.setReadOnly(True)
1504            self.boxThetaB.setButtonSymbols(QtWidgets.QAbstractSpinBox.
    NoButtons)
1505            self.boxThetaB.setDecimals(3)
1506            self.boxThetaB.setMinimum(-1.185)
1507            self.boxThetaB.setMaximum(1.245)
```

```
1508            self.boxThetaB.setObjectName("boxThetaB")
1509            self.verticalLayout_25.addWidget(self.boxThetaB)
1510            self.horizontalLayout_3.addLayout(self.verticalLayout_25)
1511            self.verticalLayout_15.addLayout(self.horizontalLayout_3)
1512            self.horizontalLayout_5.addLayout(self.verticalLayout_15)
1513            self.horizontalLayout_5.setStretch(2, 1)
1514            self.verticalLayout.addWidget(self.groupBox)
1515            self.verticalLayout.setStretch(0, 1)
1516            MainWindow.setCentralWidget(self.centralwidget)
1517            self.statusbar = QtWidgets.QStatusBar(MainWindow)
1518            self.statusbar.setObjectName("statusbar")
1519            MainWindow.setStatusBar(self.statusbar)
1520
1521            self.retranslateUi(MainWindow)
1522            self.grupo_enviar.setCurrentIndex(1)
1523            QtCore.QMetaObject.connectSlotsByName(MainWindow)
1524
1525            # Grupo botones
1526
1527            self.grupo = QtWidgets.QButtonGroup()
1528            self.grupo.addButton(self.radioFijoA)
1529            self.grupo.addButton(self.radiofijoB)
1530
1531            self.grupo2 = QtWidgets.QButtonGroup()
1532            self.grupo2.setExclusive(False)
1533            self.grupo2.addButton(self.radioImanesA)
1534            self.grupo2.addButton(self.radioImanesB)
1535
1536            # Conexiones senales y slots
1537
1538            # R1ADes
1539            self.sliderR1ADes.doubleValueChanged.connect(self.boxR1ADes
    .setValue)
1540            self.boxR1ADes.valueChanged.connect(self.sliderR1ADes.
    setValue)
1541
1542            #  L1BDes
1543            self.sliderL1BDes.doubleValueChanged.connect(self.boxL1BDes
    .setValue)
1544            self.boxL1BDes.valueChanged.connect(self.sliderL1BDes.
    setValue)
1545
1546            # R2ADes
1547            self.sliderR2ADes.doubleValueChanged.connect(self.boxR2ADes
    .setValue)
1548            self.boxR2ADes.valueChanged.connect(self.sliderR2ADes.
    setValue)
1549
1550            # L2BDes
1551            self.sliderL2BDes.doubleValueChanged.connect(self.boxL2BDes
    .setValue)
1552            self.boxL2BDes.valueChanged.connect(self.sliderL2BDes.
    setValue)
1553
1554            # L1ADes
1555            self.sliderL1ADes.doubleValueChanged.connect(self.boxL1ADes
    .setValue)
```

```
1556            self.boxL1ADes.valueChanged.connect(self.sliderL1ADes.
    setValue)
1557
1558            # R1BDes
1559            self.sliderR1BDes.doubleValueChanged.connect(self.boxR1BDes
    .setValue)
1560            self.boxR1BDes.valueChanged.connect(self.sliderR1BDes.
    setValue)
1561
1562            # L2ADes
1563            self.sliderL2ADes.doubleValueChanged.connect(self.boxL2ADes
    .setValue)
1564            self.boxL2ADes.valueChanged.connect(self.sliderL2ADes.
    setValue)
1565
1566            # R2BDes
1567            self.sliderR2BDes.doubleValueChanged.connect(self.boxR2BDes
    .setValue)
1568            self.boxR2BDes.valueChanged.connect(self.sliderR2BDes.
    setValue)
1569
1570            # ThetaADes
1571            self.sliderThetaADes.doubleValueChanged.connect(self.
    boxThetaADes.setValue)
1572            self.boxThetaADes.valueChanged.connect(self.sliderThetaADes
    .setValue)
1573
1574            # ThetaBDes
1575            self.sliderThetaBDes.doubleValueChanged.connect(self.
    boxThetaBDes.setValue)
1576            self.boxThetaBDes.valueChanged.connect(self.sliderThetaBDes
    .setValue)
1577
1578            # R1A
1579            self.boxR1A.valueChanged.connect(self.sliderR1A.setValue)
1580
1581            # L1B
1582            self.boxL1B.valueChanged.connect(self.sliderL1B.setValue)
1583
1584            # R2A
1585            self.boxR2A.valueChanged.connect(self.sliderR2A.setValue)
1586
1587            # L2B
1588            self.boxL2B.valueChanged.connect(self.sliderL2B.setValue)
1589
1590            # L1A
1591            self.boxL1A.valueChanged.connect(self.sliderL1A.setValue)
1592
1593            # R1B
1594            self.boxR1B.valueChanged.connect(self.sliderR1B.setValue)
1595
1596            # L2A
1597            self.boxL2A.valueChanged.connect(self.sliderL2A.setValue)
1598
1599            # R2B
1600            self.boxR2B.valueChanged.connect(self.sliderR2B.setValue)
1601
```

```
1602         # ThetaA
1603         self.boxThetaA.valueChanged.connect(self.sliderThetaA.
     setValue)
1604 
1605         # ThetaB
1606         self.boxThetaB.valueChanged.connect(self.sliderThetaB.
     setValue)
1607 
1608         #
     ################################################################
1609         # ROS
     ############################################################
1610 
1611         self.radioFijoA.clicked.connect(self.new_status)
1612         self.radiofijoB.clicked.connect(self.new_status)
1613         self.radioImanesA.clicked.connect(self.new_status)
1614         self.radioImanesB.clicked.connect(self.new_status)
1615 
1616         self.pushArt.clicked.connect(self.new_qdes)
1617 
1618         self.pushXp.pressed.connect(self.new_vel)
1619         self.pushXm.pressed.connect(self.new_vel)
1620         self.pushYp.pressed.connect(self.new_vel)
1621         self.pushYm.pressed.connect(self.new_vel)
1622         self.pushZp.pressed.connect(self.new_vel)
1623         self.pushZm.pressed.connect(self.new_vel)
1624         self.pushGXp.pressed.connect(self.new_vel)
1625         self.pushGXm.pressed.connect(self.new_vel)
1626         self.pushGYp.pressed.connect(self.new_vel)
1627         self.pushGYm.pressed.connect(self.new_vel)
1628         self.pushGZp.pressed.connect(self.new_vel)
1629         self.pushGZm.pressed.connect(self.new_vel)
1630 
1631 
1632     def retranslateUi(self, MainWindow):
1633         _translate = QtCore.QCoreApplication.translate
1634         MainWindow.setWindowTitle(_translate("MainWindow", "
     MainWindow"))
1635         self.pushPose.setText(_translate("MainWindow", "Enviar"))
1636         self.label.setText(_translate("MainWindow", "r1A"))
1637         self.boxR1ADes.setSuffix(_translate("MainWindow", "mm"))
1638         self.label_2.setText(_translate("MainWindow", "l1B"))
1639         self.boxL1BDes.setSuffix(_translate("MainWindow", "mm"))
1640         self.label_3.setText(_translate("MainWindow", "r2A"))
1641         self.boxR2ADes.setSuffix(_translate("MainWindow", "mm"))
1642         self.label_5.setText(_translate("MainWindow", "l2B"))
1643         self.boxL2BDes.setSuffix(_translate("MainWindow", "mm"))
1644         self.label_8.setText(_translate("MainWindow", "l1A"))
1645         self.boxL1ADes.setSuffix(_translate("MainWindow", "mm"))
1646         self.label_4.setText(_translate("MainWindow", "r1B"))
1647         self.boxR1BDes.setSuffix(_translate("MainWindow", "mm"))
1648         self.label_7.setText(_translate("MainWindow", "l2A"))
1649         self.boxL2ADes.setSuffix(_translate("MainWindow", "mm"))
1650         self.label_6.setText(_translate("MainWindow", "r2B"))
1651         self.boxR2BDes.setSuffix(_translate("MainWindow", "mm"))
1652         self.label_9.setText(_translate("MainWindow", "<html><head
     /><body><p>thetaA</p></body></html>"))
```

```
1653            self.boxThetaADes.setSuffix(_translate("MainWindow", "rad")
    )
1654            self.label_10.setText(_translate("MainWindow", "thetaB"))
1655            self.boxThetaBDes.setSuffix(_translate("MainWindow", "rad")
    )
1656            self.pushArt.setText(_translate("MainWindow", "Enviar"))
1657            self.grupo_enviar.setTabText(self.grupo_enviar.indexOf(self
    .tabWidget_2Page1), _translate("MainWindow", "Comandar
    posiciones, orientaciones y configuraciones articulares"))
1658            self.label_28.setText(_translate("MainWindow", "<html><head
    /><body><p align=\"center\"><span style=\" font-size:12pt; font
    -weight:700;\">Posicion</span></p></body></html>"))
1659            self.label_24.setText(_translate("MainWindow", "y+"))
1660            self.label_27.setText(_translate("MainWindow", "x-"))
1661            self.label_23.setText(_translate("MainWindow", "z-"))
1662            self.label_22.setText(_translate("MainWindow", "z+"))
1663            self.label_26.setText(_translate("MainWindow", "x+"))
1664            self.label_25.setText(_translate("MainWindow", "y-"))
1665            self.label_29.setText(_translate("MainWindow", "<html><head
    /><body><p align=\"center\"><span style=\" font-size:12pt; font
    -weight:700;\">Orientacion</span></p></body></html>"))
1666            self.label_30.setText(_translate("MainWindow", "y+"))
1667            self.label_31.setText(_translate("MainWindow", "x-"))
1668            self.label_32.setText(_translate("MainWindow", "z-"))
1669            self.label_33.setText(_translate("MainWindow", "z+"))
1670            self.label_34.setText(_translate("MainWindow", "x+"))
1671            self.label_35.setText(_translate("MainWindow", "y-"))
1672            self.label_incrementos_1.setText(_translate("MainWindow", "
    <html><head/><body><p align=\"center\"><span style=\" font-size
    :12pt; font-weight:700;\">Incrementos</span></p></body></html>"
    ))
1673            self.label_incrementos_2.setText(_translate("MainWindow", "
    Posicion"))
1674            self.label_incrementos_3.setText(_translate("MainWindow", "
    Orientacion"))
1675            self.grupo_enviar.setTabText(self.grupo_enviar.indexOf(self
    .tab), _translate("MainWindow", "Control en velocidad"))
1676            self.groupBox.setTitle(_translate("MainWindow", "Valores
    actuales"))
1677            self.label_21.setText(_translate("MainWindow", "<html><head
    /><body><p align=\"center\">Sistema de referencia</p></body></
    html>"))
1678            self.radioFijoA.setText(_translate("MainWindow", "A"))
1679            self.radiofijoB.setText(_translate("MainWindow", "B"))
1680            self.radioImanesA.setText(_translate("MainWindow", "Imanes
    A"))
1681            self.radioImanesB.setText(_translate("MainWindow", "Imanes
    B"))
1682            self.label_11.setText(_translate("MainWindow", "r1A"))
1683            self.boxR1A.setSuffix(_translate("MainWindow", "mm"))
1684            self.label_12.setText(_translate("MainWindow", "l1B"))
1685            self.boxL1B.setSuffix(_translate("MainWindow", "mm"))
1686            self.label_13.setText(_translate("MainWindow", "r2A"))
1687            self.boxR2A.setSuffix(_translate("MainWindow", "mm"))
1688            self.label_14.setText(_translate("MainWindow", "l2B"))
1689            self.boxL2B.setSuffix(_translate("MainWindow", "mm"))
1690            self.label_15.setText(_translate("MainWindow", "l1A"))
```

```
1691            self.boxL1A.setSuffix(_translate("MainWindow", "mm"))
1692            self.label_16.setText(_translate("MainWindow", "r1B"))
1693            self.boxR1B.setSuffix(_translate("MainWindow", "mm"))
1694            self.label_17.setText(_translate("MainWindow", "l2A"))
1695            self.boxL2A.setSuffix(_translate("MainWindow", "mm"))
1696            self.label_18.setText(_translate("MainWindow", "r2B"))
1697            self.boxR2B.setSuffix(_translate("MainWindow", "mm"))
1698            self.label_19.setText(_translate("MainWindow", "<html><head
    /><body><p>thetaA</p></body></html>"))
1699            self.boxThetaA.setSuffix(_translate("MainWindow", "rad"))
1700            self.label_20.setText(_translate("MainWindow", "thetaB"))
1701            self.boxThetaB.setSuffix(_translate("MainWindow", "rad"))
1702
1703
1704 if __name__ == "__main__":
1705     import sys
1706     app = QtWidgets.QApplication(sys.argv)
1707     MainWindow = QtWidgets.QMainWindow()
1708     ui = Ui_MainWindow()
1709     ui.setupUi(MainWindow)
1710     MainWindow.show()
1711     sys.exit(app.exec_())
```

A.2. Nodo de cálculo de la cinemática inversa

```
#include <memory>
#include <chrono>

#include "rclcpp/rclcpp.hpp"
#include "hyrecro_interfaces/msg/q_real.hpp"
// #include "hyrecro_interfaces/msg/q_int.hpp"
// #include "hyrecro_interfaces/msg/xy_phi.hpp"
#include "hyrecro_interfaces/msg/status.hpp"
#include "geometry_msgs/msg/twist.hpp"


#include <math.h>
#include <eigen3/Eigen/Dense>

#define PI 3.1415926535897932384
#define B 0.025
#define P 0.0315
#define H 0.07
#define T 0.11

using namespace Eigen;
using std::placeholders::_1;
using namespace std::chrono_literals;

void fk_parallel_module(double l, double r, double &phi, double &y)
    ;
void ik_parallel_module(double phi, double y, double &l, double &r)
    ;
void real2int(double q_real[10], Ref<VectorXd> q_int);
void int2real(const Ref<VectorXd>& q_int, double q_real[10]);
MatrixXd jacob(const Ref<VectorXd>& q, bool fijo_A);
MatrixXd pinv(const Ref<MatrixXd>& J);
bool comprobar_limites(double q_real[10]);

bool g_fijo_A = true;
bool g_limites_corregidos;
VectorXd g_v(6);
VectorXd g_q_int(8), g_delta_q_int(8);
MatrixXd g_J(6,8);
double g_q_real[10];

class compute_vels : public rclcpp::Node
{
  public:
    compute_vels()
    : Node("compute_vels")
    {
      status_sub_ = this->create_subscription<hyrecro_interfaces::
    msg::Status>("status", 10, std::bind(&compute_vels::
    status_callback, this, _1));
      vel_sub_ = this->create_subscription<geometry_msgs::msg::
    Twist>("cmd_vel", 10, std::bind(&compute_vels::vel_callback,
    this, _1));
     q_real_sub_ = this->create_subscription<hyrecro_interfaces::msg
    ::QReal>("q_real", 10, std::bind(&compute_vels::q_real_callback
```

```
    , this, _1));
49      q_real_pub_ = this->create_publisher<hyrecro_interfaces::msg
    ::QReal>("q_real", 10);
50     //   q_int_pub_ = this->create_publisher<hyrecro_interfaces::
    msg::QInt>("q_int", 10);
51     }
52
53   private:
54   void vel_callback(const geometry_msgs::msg::Twist::SharedPtr msg)
       const{
55     g_v << msg->linear.x, msg->linear.y, msg->linear.z, msg->
     angular.x, msg->angular.y, msg->angular.z;
56     // std::cout << g_v << std::endl << std::endl;
57
58     // for (int i = 0; i<10; i++){
59     //  std::cout << g_q_real[i] << std::endl;
60     // }
61     real2int(g_q_real, g_q_int);
62     // std::cout << g_q_int << std::endl;
63     g_J = jacob(g_q_int, g_fijo_A);
64     JacobiSVD<MatrixXd> svd(g_J);
65     VectorXd sing_val = svd.singularValues();
66     if (sing_val(sing_val.size()-1) == 0) {
67       std::cout << "Singularidad. Se calculara jacobiana
     amortiguada (damped)." << std::endl;
68       g_J += 0.01*MatrixXd::Identity(6, 8);
69       std::cout << g_J << std::endl;
70     }
71     // std::cout << g_J << std::endl;
72     g_delta_q_int = pinv(g_J) * g_v;
73     // std::cout << pinv(g_J) << std::endl;
74     std::cout << g_delta_q_int << std::endl;
75     g_q_int += g_delta_q_int;
76
77     int2real(g_q_int, g_q_real);
78
79     // comprobar limites q_real
80     g_limites_corregidos = comprobar_limites(g_q_real);
81
82     // publicar q_real y q_int
83     auto Q = hyrecro_interfaces::msg::QReal();
84     Q.r_1a = g_q_real[0];
85     Q.l_1b = g_q_real[1];
86     Q.r_2a = g_q_real[2];
87     Q.l_2b = g_q_real[3];
88     Q.l_1a = g_q_real[4];
89     Q.r_1b = g_q_real[5];
90     Q.l_2a = g_q_real[6];
91     Q.r_2b = g_q_real[7];
92     Q.theta_a = g_q_real[8];
93     Q.theta_b = g_q_real[9];
94     q_real_pub_->publish(Q);
95
96     // auto Q_Int = hyrecro_interfaces::msg::QInt();
97     // Q_Int.phi_1a = q_int[0];
98     // Q_Int.y_a = q_int[1];
99     // Q_Int.phi_2a = q_int[2];
```

```
100     // Q_Int.theta_a = q_int[3];
101     // Q_Int.theta_b = q_int[4];
102     // Q_Int.phi_2b = q_int[5];
103     // Q_Int.y_b = q_int[6];
104     // Q_Int.phi_1b = q_int[7];
105     // q_int_pub_->publish(Q_Int);
106 
107       }
108   void status_callback(const hyrecro_interfaces::msg::Status::
      SharedPtr msg) const{
109     g_fijo_A = msg->fijo_a;
110     RCLCPP_INFO(this->get_logger(), "status_callback");
111   }
112   void q_real_callback(const hyrecro_interfaces::msg::QReal::
      SharedPtr msg) const{
113     g_q_real[0] = msg->r_1a;
114     g_q_real[1] = msg->l_1b;
115     g_q_real[2] = msg->r_2a;
116     g_q_real[3] = msg->l_2b;
117     g_q_real[4] = msg->l_1a;
118     g_q_real[5] = msg->r_1b;
119     g_q_real[6] = msg->l_2a;
120     g_q_real[7] = msg->r_2b;
121     g_q_real[8] = msg->theta_a;
122     g_q_real[9] = msg->theta_b;
123     RCLCPP_INFO(this->get_logger(), "q_real_callback");
124   }
125   rclcpp::Subscription<geometry_msgs::msg::Twist>::SharedPtr
      vel_sub_;
126   rclcpp::Subscription<hyrecro_interfaces::msg::Status>::SharedPtr
      status_sub_;
127   rclcpp::Publisher<hyrecro_interfaces::msg::QReal>::SharedPtr
      q_real_pub_;
128   rclcpp::Subscription<hyrecro_interfaces::msg::QReal>::SharedPtr
      q_real_sub_;
129 };
130 
131 int main(int argc, char * argv[]){
132   // real2int(q_real, q_int);
133 
134   rclcpp::init(argc, argv);
135   rclcpp::spin(std::make_shared<compute_vels>());
136   rclcpp::shutdown();
137 
138   return 0;
139 }
140 
141 void fk_parallel_module(double l, double r, double &phi, double &y)
      {
142 
143         double r0, r1;
144         double epsilon = 0.00001;
145         double Delta_phi, Delta_y;
146         double phi_previo = 0, y_previo = 0;
147         // Solucion inicial (del capitulo de InTech que escribieron
      Ubeda et al).
```

```
148         // Esta solucion inicial parte de que los actuadores
    lineales estaran normalmente casi paralelos, dada la geometria
    del robot.
149         phi = atan2(r-l,2*B);
150         y = (l+r)/2;
151         // Primeros residuos
152         r0 = (P*cos(phi)-B)*(P*cos(phi)-B) + (y+P*sin(phi))*(y+P*
    sin(phi)) - r*r;
153         r1 = (P*cos(phi)-B)*(P*cos(phi)-B) + (y-P*sin(phi))*(y-P*
    sin(phi)) - l*l;
154         // Como estaremos siempre lejos de singularidades paralelas
     (es decir, soluciones dobles), la convergencia de Newton sera
    cuadratica.
155         // Parece que, para detener el algoritmo de Newton, es mas
    conveniente ver cuando cambia poco la solucion (mejor que
    comprobar los residuos).
156         // Notese que y original NUNCA puede ser cero. Por tanto,
    nunca se detendra a la primera iteracion.
157         double eps;
158         eps = sqrt((phi-phi_previo)*(phi-phi_previo)+(y-y_previo)*(
    y-y_previo));
159         while(eps > epsilon){
160                 Delta_phi = (P*(r0 + r1)*sin(phi) + y*(r1 - r0))
    /(4*P*(y*y*cos(phi) - B*P*sin(phi)*sin(phi)));
161                 Delta_y = (y*(r0 + r1)*cos(phi) + B*(r1 - r0)*sin(
    phi))/(4*(B*P*sin(phi)*sin(phi) - y*y*cos(phi)));
162                 phi_previo = phi;
163                 y_previo = y;
164         phi = phi + Delta_phi;
165                 y = y + Delta_y;
166                 r0 = (P*cos(phi)-B)*(P*cos(phi)-B) + (y+P*sin(phi))
    *(y+P*sin(phi)) - r*r;
167                 r1 = (P*cos(phi)-B)*(P*cos(phi)-B) + (y-P*sin(phi))
    *(y-P*sin(phi)) - l*l;
168                 eps = sqrt((phi-phi_previo)*(phi-phi_previo)+(y-
    y_previo)*(y-y_previo));
169         }
170 }
171
172 void ik_parallel_module(double phi, double y, double &l, double &r)
    {
173         l = sqrt(pow(P*cos(phi) - B, 2) + pow(y - P*sin(phi), 2));
174         r = sqrt(pow(P*cos(phi) - B, 2) + pow(y + P*sin(phi), 2));
175 }
176
177 void real2int(double q_real[10], Ref<VectorXd> q_int){
178         // Calcula las coordenadas articulares intermedias a partir
     de los valores de las coordenadas articulares reales (de los
    actuadores)
179         // q_real: (r_1A, l_1B, r_2A, l_2B, l_1A, r_1B, l_2A, r_2B,
     theta_A, theta_B)
180         // q_int: (phi_1A, y_A, phi_2A, theta_A, theta_B, phi_2B,
    y_B, phi_1B)
181         double y_1A, y_2A, y_1B, y_2B;
182         // thetas
183         q_int(3) = q_real[8];
184         q_int(4) = q_real[9];
```

```
185         // modulo 1A
186         fk_parallel_module(q_real[4], q_real[0], q_int(0), y_1A);
187         // modulo 2A
188         fk_parallel_module(q_real[6], q_real[2], q_int(2), y_2A);
189         // modulo 2B
190         fk_parallel_module(q_real[3], q_real[7], q_int(5), y_2B);
191         // modulo 1B
192         fk_parallel_module(q_real[1], q_real[5], q_int(7), y_1B);
193         // cuerpos centrales de los modulos paralelos
194         q_int(1) = y_1A + y_2A - H;
195         q_int(6) = y_1B + y_2B - H;
196 }
197
198 void int2real(const Ref<VectorXd>& q_int, double q_real[10]){
199         // Calcula las coordenadas articulares reales (de los
    actuadores) a partir de los valores de las coordenadas
    articulares intermedias
200         // q_int: (phi_1A, y_A, phi_2A, theta_A, theta_B, phi_2B,
    y_B, phi_1B)
201         // q_real: (r_1A, l_1B, r_2A, l_2B, l_1A, r_1B, l_2A, r_2B,
     theta_A, theta_B)
202         double y_1A, y_2A, y_1B, y_2B;
203         // cuerpos centrales de los modulos paralelos
204         y_1A = (q_int(1) + H)/2;
205         y_2A = (q_int(1) + H)/2;
206         y_1B = (q_int(6) + H)/2;
207         y_2B = (q_int(6) + H)/2;
208         // modulo 1A
209         ik_parallel_module(q_int(0), y_1A, q_real[4], q_real[0]);
210         // modulo 2A
211         ik_parallel_module(q_int(2), y_2A, q_real[6], q_real[2]);
212         // modulo 2B
213         ik_parallel_module(q_int(5), y_2B, q_real[3], q_real[7]);
214         // modulo 1B
215         ik_parallel_module(q_int(7), y_1B, q_real[1], q_real[5]);
216         // thetas
217         q_real[8] = q_int(3);
218         q_real[9] = q_int(4);
219 }
220
221 MatrixXd jacob(const Ref<VectorXd>& q, bool fijo_A){
222   MatrixXd J = MatrixXd::Zero(6,8);
223   if (fijo_A){
224     // primera fila
225     J(0,0) = q(1)*cos(q(0)) - q(6)*(cos(q(0) - q(2))*cos(q(5)) -
    cos(q(3) - q(4))*sin(q(0) - q(2))*sin(q(5))) - T*sin(q(0) - q
    (2))*cos(q(3));
226     J(0,1) = sin(q(0));
227     J(0,2) = q(6)*(cos(q(0) - q(2))*cos(q(5)) - cos(q(3) - q(4))*
    sin(q(0) - q(2))*sin(q(5))) + T*sin(q(0) - q(2))*cos(q(3));
228     J(0,3) = q(6)*cos(q(0) - q(2))*sin(q(3) - q(4))*sin(q(5)) - T*
    cos(q(0) - q(2))*sin(q(3));
229     J(0,4) = -q(6)*cos(q(0) - q(2))*sin(q(3) - q(4))*sin(q(5));
230     J(0,5) = q(6)*(sin(q(0) - q(2))*sin(q(5)) - cos(q(0) - q(2))*
    cos(q(3) - q(4))*cos(q(5)));
231     J(0,6) = -sin(q(0) - q(2))*cos(q(5)) - cos(q(0) - q(2))*cos(q
    (3) - q(4))*sin(q(5));
```

```
232
233     // segunda fila
234     J(1,0) = q(6)*(sin(q(0) - q(2))*cos(q(5)) + cos(q(0) - q(2))*
        cos(q(3) - q(4))*sin(q(5))) - q(1)*sin(q(0)) + T*sin(q(0) - q
        (2))*cos(q(3));
235     J(1,1) = cos(q(0));
236     J(1,2) = -q(6)*(sin(q(0) - q(2))*cos(q(5)) + cos(q(0) - q(2))*
        cos(q(3) - q(4))*sin(q(5))) - T*sin(q(0) - q(2))*cos(q(3));
237     J(1,3) = T*cos(q(0) - q(2))*sin(q(3)) - q(6)*sin(q(0) - q(2))*
        sin(q(3) - q(4))*sin(q(5));
238     J(1,4) = q(6)*sin(q(0) - q(2))*sin(q(3) - q(4))*sin(q(5));
239     J(1,5) = q(6)*(cos(q(0) - q(2))*sin(q(5)) + cos(q(3) - q(4))*
        sin(q(0) - q(2))*cos(q(5)));
240     J(1,6) = cos(q(3) - q(4))*sin(q(0) - q(2))*sin(q(5)) - cos(q(0)
        - q(2))*cos(q(5));
241
242     // tercera fila
243     J(2,3) = q(6)*cos(q(3) - q(4))*sin(q(5)) - T*cos(q(3));
244     J(2,4) = -q(6)*cos(q(3) - q(4))*sin(q(5));
245     J(2,5) = q(6)*sin(q(3) - q(4))*cos(q(5));
246     J(2,6) = sin(q(3) - q(4))*sin(q(5));
247
248     // cuarta fila
249     J(3,3) = sin(q(0) - q(2));
250     J(3,4) = -sin(q(0) - q(2));
251     J(3,5) = -cos(q(0) - q(2))*sin(q(3) - q(4));
252     J(3,7) = cos(q(0) - q(2))*sin(q(3) - q(4));
253
254     // quinta fila
255     J(4,3) = cos(q(0) - q(2));
256     J(4,4) = -cos(q(0) - q(2));
257     J(4,5) = sin(q(0) - q(2))*sin(q(3) - q(4));
258     J(4,7) = -sin(q(0) - q(2))*sin(q(3) - q(4));
259
260     // sexta fila
261     J(5,0) = -1;
262     J(5,2) = 1;
263     J(5,5) = -cos(q(3) - q(4));
264     J(5,7) = cos(q(3) - q(4));
265   } else {
266     // primera fila
267     J(0,0) = T*cos(q(3))*(sin(q(7) - q(5)) + cos(q(7) - q(5))*cos(q
        (3) - q(4)) - 2*pow(cos(q(0) - q(2)),2)*sin(q(7) - q(5)) - 2*
        cos(q(0) - q(2))*sin(q(0) - q(2))*sin(q(7) - q(5)) - 2*pow(cos(
        q(0) - q(2)),2)*cos(q(7) - q(5))*cos(q(3) - q(4)) + 2*cos(q(0)
        - q(2))*cos(q(7) - q(5))*cos(q(3) - q(4))*sin(q(0) - q(2)));
268     J(0,1) = - sin(q(7) - q(5))*cos(q(2)) - cos(q(7) - q(5))*cos(q
        (3) - q(4))*sin(q(2));
269     J(0,2) = T*pow(cos(q(0) - q(2)),2)*sin(q(7) - q(5))*cos(q(3)) -
         T*pow(sin(q(0) - q(2)),2)*sin(q(7) - q(5))*cos(q(3)) + q(1)*
        cos(q(0) - q(2))*sin(q(7) - q(5))*sin(q(0)) - q(1)*sin(q(0) - q
        (2))*sin(q(7) - q(5))*cos(q(0)) + T*pow(cos(q(0) - q(2)),2)*cos
        (q(7) - q(5))*cos(q(3) - q(4))*cos(q(3)) - T*cos(q(7) - q(5))*
        cos(q(3) - q(4))*pow(sin(q(0) - q(2)),2)*cos(q(3)) - q(1)*cos(q
        (0) - q(2))*cos(q(7) - q(5))*cos(q(3) - q(4))*cos(q(0)) + 2*T*
        cos(q(0) - q(2))*sin(q(0) - q(2))*sin(q(7) - q(5))*cos(q(3)) -
        q(1)*cos(q(7) - q(5))*cos(q(3) - q(4))*sin(q(0) - q(2))*sin(q
```

```
    (0)) - 2*T*cos(q(0) - q(2))*cos(q(7) - q(5))*cos(q(3) - q(4))*
    sin(q(0) - q(2))*cos(q(3));
270  J(0,3) = (sin(q(0) - q(2))*sin(q(7) - q(5)) + cos(q(0) - q(2))*
    cos(q(7) - q(5))*cos(q(3) - q(4)))*(T*cos(q(0) - q(2))*sin(q(3)
    ) - q(6)*cos(q(0) - q(2))*sin(q(3) - q(4))*sin(q(5))) - (T*cos(
    q(0) - q(2))*sin(q(3)) - q(6)*sin(q(0) - q(2))*sin(q(3) - q(4))
    *sin(q(5)))*(cos(q(0) - q(2))*sin(q(7) - q(5)) - cos(q(7) - q
    (5))*cos(q(3) - q(4))*sin(q(0) - q(2))) - cos(q(7) - q(5))*sin(
    q(3) - q(4))*(T*cos(q(3)) - q(6)*cos(q(3) - q(4))*sin(q(5))) -
    cos(q(7) - q(5))*cos(q(3) - q(4))*(T*sin(q(3)) - q(6)*sin(q(3)
    - q(4))*sin(q(5))) + cos(q(0) - q(2))*cos(q(7) - q(5))*sin(q(3)
     - q(4))*(q(1)*sin(q(0)) - q(6)*(sin(q(0) - q(2))*cos(q(5)) +
    cos(q(0) - q(2))*cos(q(3) - q(4))*sin(q(5))) + T*cos(q(0) - q
    (2))*cos(q(3))) + cos(q(7) - q(5))*sin(q(0) - q(2))*sin(q(3) -
    q(4))*(q(6)*(cos(q(0) - q(2))*cos(q(5)) - cos(q(3) - q(4))*sin(
    q(0) - q(2))*sin(q(5))) - q(1)*cos(q(0)) + T*cos(q(0) - q(2))*
    cos(q(3)));
271  J(0,4) = cos(q(7) - q(5))*(T*cos(q(3) - q(4))*sin(q(3)) - T*pow
    (cos(q(0) - q(2)),2)*sin(q(3) - q(4))*cos(q(3)) - q(1)*cos(q(0)
     - q(2))*sin(q(3) - q(4))*sin(q(0)) + q(1)*sin(q(0) - q(2))*sin
    (q(3) - q(4))*cos(q(0)) - 2*q(6)*cos(q(3) - q(4))*sin(q(3) - q
    (4))*sin(q(5)) + 2*q(6)*pow(cos(q(0) - q(2)),2)*cos(q(3) - q(4)
    )*sin(q(3) - q(4))*sin(q(5)) + 2*q(6)*cos(q(3) - q(4))*pow(sin(
    q(0) - q(2)),2)*sin(q(3) - q(4))*sin(q(5)) - T*cos(q(0) - q(2))
    *sin(q(0) - q(2))*sin(q(3) - q(4))*cos(q(3)));
272  J(0,5) = (cos(q(7) - q(5))*sin(q(0) - q(2)) - cos(q(0) - q(2))*
    cos(q(3) - q(4))*sin(q(7) - q(5)))*(q(1)*sin(q(0)) - q(6)*(sin(
    q(0) - q(2))*cos(q(5)) + cos(q(0) - q(2))*cos(q(3) - q(4))*sin(
    q(5))) + T*cos(q(0) - q(2))*cos(q(3))) - (cos(q(0) - q(2))*cos(
    q(7) - q(5)) + cos(q(3) - q(4))*sin(q(0) - q(2))*sin(q(7) - q
    (5)))*(q(6)*(cos(q(0) - q(2))*cos(q(5)) - cos(q(3) - q(4))*sin(
    q(0) - q(2))*sin(q(5))) - q(1)*cos(q(0)) + T*cos(q(0) - q(2))*
    cos(q(3))) - q(6)*(sin(q(0) - q(2))*sin(q(7) - q(5)) + cos(q(0)
     - q(2))*cos(q(7) - q(5))*cos(q(3) - q(4)))*(sin(q(0) - q(2))*
    sin(q(5)) - cos(q(0) - q(2))*cos(q(3) - q(4))*cos(q(5))) - q(6)
    *(cos(q(0) - q(2))*sin(q(7) - q(5)) - cos(q(7) - q(5))*cos(q(3)
     - q(4))*sin(q(0) - q(2)))*(cos(q(0) - q(2))*sin(q(5)) + cos(q
    (3) - q(4))*sin(q(0) - q(2))*cos(q(5))) - sin(q(7) - q(5))*sin(
    q(3) - q(4))*(T*sin(q(3)) - q(6)*sin(q(3) - q(4))*sin(q(5))) +
    q(6)*cos(q(7) - q(5))*pow(sin(q(3) - q(4)),2)*cos(q(5));
273  J(0,6) = sin(q(7));
274  J(0,7) = (cos(q(0) - q(2))*cos(q(7) - q(5)) + cos(q(3) - q(4))*
    sin(q(0) - q(2))*sin(q(7) - q(5)))*(q(6)*(cos(q(0) - q(2))*cos(
    q(5)) - cos(q(3) - q(4))*sin(q(0) - q(2))*sin(q(5))) - q(1)*cos
    (q(0)) + T*cos(q(0) - q(2))*cos(q(3))) - (cos(q(7) - q(5))*sin(
    q(0) - q(2)) - cos(q(0) - q(2))*cos(q(3) - q(4))*sin(q(7) - q
    (5)))*(q(1)*sin(q(0)) - q(6)*(sin(q(0) - q(2))*cos(q(5)) + cos(
    q(0) - q(2))*cos(q(3) - q(4))*sin(q(5))) + T*cos(q(0) - q(2))*
    cos(q(3))) + sin(q(7) - q(5))*sin(q(3) - q(4))*(T*sin(q(3)) - q
    (6)*sin(q(3) - q(4))*sin(q(5)));
275
276  // segunda fila
277  J(1,0) = -T*cos(q(3))*(cos(q(3) - q(4))*sin(q(7) - q(5)) - cos(
    q(7) - q(5)) + 2*pow(cos(q(0) - q(2)),2)*cos(q(7) - q(5)) + 2*
    cos(q(0) - q(2))*cos(q(7) - q(5))*sin(q(0) - q(2)) - 2*pow(cos(
    q(0) - q(2)),2)*cos(q(3) - q(4))*sin(q(7) - q(5)) + 2*cos(q(0)
    - q(2))*cos(q(3) - q(4))*sin(q(0) - q(2))*sin(q(7) - q(5)));
```

```
278     J(1,1) = cos(q(3) - q(4))*sin(q(7) - q(5))*sin(q(2)) - cos(q(7)
        - q(5))*cos(q(2));
279     J(1,2) = T*pow(cos(q(0) - q(2)),2)*cos(q(7) - q(5))*cos(q(3)) -
        T*cos(q(7) - q(5))*pow(sin(q(0) - q(2)),2)*cos(q(3)) + q(1)*
    cos(q(0) - q(2))*cos(q(7) - q(5))*sin(q(0)) - q(1)*cos(q(7) - q
    (5))*sin(q(0) - q(2))*cos(q(0)) - T*pow(cos(q(0) - q(2)),2)*cos
    (q(3) - q(4))*sin(q(7) - q(5))*cos(q(3)) + T*cos(q(3) - q(4))*
    pow(sin(q(0) - q(2)),2)*sin(q(7) - q(5))*cos(q(3)) + 2*T*cos(q
    (0) - q(2))*cos(q(7) - q(5))*sin(q(0) - q(2))*cos(q(3)) + q(1)*
    cos(q(0) - q(2))*cos(q(3) - q(4))*sin(q(7) - q(5))*cos(q(0)) +
    q(1)*cos(q(3) - q(4))*sin(q(0) - q(2))*sin(q(7) - q(5))*sin(q
    (0)) + 2*T*cos(q(0) - q(2))*cos(q(3) - q(4))*sin(q(0) - q(2))*
    sin(q(7) - q(5))*cos(q(3));
280     J(1,3) = sin(q(7) - q(5))*sin(q(3) - q(4))*(T*cos(q(3)) - q(6)*
    cos(q(3) - q(4))*sin(q(5))) - (T*cos(q(0) - q(2))*sin(q(3)) - q
    (6)*sin(q(0) - q(2))*sin(q(3) - q(4))*sin(q(5)))*(cos(q(0) - q
    (2))*cos(q(7) - q(5)) + cos(q(3) - q(4))*sin(q(0) - q(2))*sin(q
    (7) - q(5))) + cos(q(3) - q(4))*sin(q(7) - q(5))*(T*sin(q(3)) -
     q(6)*sin(q(3) - q(4))*sin(q(5))) + cos(q(0) - q(2))*(cos(q(7)
    - q(5))*sin(q(0) - q(2)) - cos(q(0) - q(2))*cos(q(3) - q(4))*
    sin(q(7) - q(5)))*(T*sin(q(3)) - q(6)*sin(q(3) - q(4))*sin(q(5)
    )) - cos(q(0) - q(2))*sin(q(7) - q(5))*sin(q(3) - q(4))*(q(1)*
    sin(q(0)) + T*cos(q(0) - q(2))*cos(q(3)) - q(6)*sin(q(0) - q(2)
    )*cos(q(5)) - q(6)*cos(q(0) - q(2))*cos(q(3) - q(4))*sin(q(5)))
     + sin(q(0) - q(2))*sin(q(7) - q(5))*sin(q(3) - q(4))*(q(1)*cos
    (q(0)) - T*cos(q(0) - q(2))*cos(q(3)) - q(6)*cos(q(0) - q(2))*
    cos(q(5)) + q(6)*cos(q(3) - q(4))*sin(q(0) - q(2))*sin(q(5)));
281     J(1,4) = -sin(q(7) - q(5))*(T*cos(q(3) - q(4))*sin(q(3)) - T*
    pow(cos(q(0) - q(2)),2)*sin(q(3) - q(4))*cos(q(3)) - q(1)*cos(q
    (0) - q(2))*sin(q(3) - q(4))*sin(q(0)) + q(1)*sin(q(0) - q(2))*
    sin(q(3) - q(4))*cos(q(0)) - 2*q(6)*cos(q(3) - q(4))*sin(q(3) -
     q(4))*sin(q(5)) + 2*q(6)*pow(cos(q(0) - q(2)),2)*cos(q(3) - q
    (4))*sin(q(3) - q(4))*sin(q(5)) + 2*q(6)*cos(q(3) - q(4))*pow(
    sin(q(0) - q(2)),2)*sin(q(3) - q(4))*sin(q(5)) - T*cos(q(0) - q
    (2))*sin(q(0) - q(2))*sin(q(3) - q(4))*cos(q(3)));
282     J(1,5) = (cos(q(0) - q(2))*sin(q(7) - q(5)) - cos(q(7) - q(5))*
    cos(q(3) - q(4))*sin(q(0) - q(2)))*(q(6)*(cos(q(0) - q(2))*cos(
    q(5)) - cos(q(3) - q(4))*sin(q(0) - q(2))*sin(q(5))) - q(1)*cos
    (q(0)) + T*cos(q(0) - q(2))*cos(q(3))) - (sin(q(0) - q(2))*sin(
    q(7) - q(5)) + cos(q(0) - q(2))*cos(q(7) - q(5))*cos(q(3) - q
    (4)))*(q(1)*sin(q(0)) - q(6)*(sin(q(0) - q(2))*cos(q(5)) + cos(
    q(0) - q(2))*cos(q(3) - q(4))*sin(q(5))) + T*cos(q(0) - q(2))*
    cos(q(3))) - q(6)*(cos(q(0) - q(2))*cos(q(7) - q(5)) + cos(q(3)
     - q(4))*sin(q(0) - q(2))*sin(q(7) - q(5)))*(cos(q(0) - q(2))*
    sin(q(5)) + cos(q(3) - q(4))*sin(q(0) - q(2))*cos(q(5))) - q(6)
    *(cos(q(7) - q(5))*sin(q(0) - q(2)) - cos(q(0) - q(2))*cos(q(3)
     - q(4))*sin(q(7) - q(5)))*(sin(q(0) - q(2))*sin(q(5)) - cos(q
    (0) - q(2))*cos(q(3) - q(4))*cos(q(5))) - cos(q(7) - q(5))*sin(
    q(3) - q(4))*(T*sin(q(3)) - q(6)*sin(q(3) - q(4))*sin(q(5))) -
    q(6)*sin(q(7) - q(5))*pow(sin(q(3) - q(4)),2)*cos(q(5));
283     J(1,6) = cos(q(7));
284     J(1,7) = (sin(q(0) - q(2))*sin(q(7) - q(5)) + cos(q(0) - q(2))*
    cos(q(7) - q(5))*cos(q(3) - q(4)))*(q(1)*sin(q(0)) - q(6)*(sin(
    q(0) - q(2))*cos(q(5)) + cos(q(0) - q(2))*cos(q(3) - q(4))*sin(
    q(5))) + T*cos(q(0) - q(2))*cos(q(3))) - (cos(q(0) - q(2))*sin(
    q(7) - q(5)) - cos(q(7) - q(5))*cos(q(3) - q(4))*sin(q(0) - q
    (2)))*(q(6)*(cos(q(0) - q(2))*cos(q(5)) - cos(q(3) - q(4))*sin(
```

```
    q(0) - q(2))*sin(q(5))) - q(1)*cos(q(0)) + T*cos(q(0) - q(2))*
    cos(q(3))) + cos(q(7) - q(5))*sin(q(3) - q(4))*(T*sin(q(3)) - q
    (6)*sin(q(3) - q(4))*sin(q(5)));
285
286     // tercera fila
287     J(2,0) = -(pow(2,1/2)*T*(cos(2*q(2) - 2*q(0) - 2*q(3) + q(4) +
    PI/4) + cos(2*q(2) - 2*q(0) + q(4) + PI/4) - sin(2*q(0) - 2*q
    (2) + q(4) + PI/4) - cos(2*q(2) - 2*q(0) + 2*q(3) - q(4) + PI
    /4)))/4;
288     J(2,1) = -sin(q(3) - q(4))*sin(q(2));
289     J(2,2) = -sin(q(3) - q(4))*(T*pow(sin(q(0) - q(2)),2)*cos(q(3))
     - T*pow(cos(q(0) - q(2)),2)*cos(q(3)) + q(1)*cos(q(0) - q(2))*
    cos(q(0)) + q(1)*sin(q(0) - q(2))*sin(q(0)) + 2*T*cos(q(0) - q
    (2))*sin(q(0) - q(2))*cos(q(3)));
290     J(2,3) = (T*cos(2*q(3) - q(4)))/2 - (q(1)*sin(q(2) + q(3) - q
    (4)))/2 - (q(1)*sin(q(2) - q(3) + q(4)))/2 - (T*cos(2*q(0) - 2*
    q(2) - 2*q(3) + q(4)))/4 - (T*sin(2*q(0) - 2*q(2) - 2*q(3) + q
    (4)))/4 - (T*cos(2*q(0) - 2*q(2) + 2*q(3) - q(4)))/4 - (T*sin
    (2*q(0) - 2*q(2) + 2*q(3) - q(4)))/4;
291     J(2,4) = - q(6)*sin(q(5))*pow(cos(q(0) - q(2)),2)*pow(cos(q(3)
    - q(4)),2) + T*cos(q(3))*pow(cos(q(0) - q(2)),2)*cos(q(3) - q
    (4)) + q(6)*sin(q(5))*pow(cos(q(0) - q(2)),2)*pow(sin(q(3) - q
    (4)),2) + T*cos(q(3))*cos(q(0) - q(2))*cos(q(3) - q(4))*sin(q
    (0) - q(2)) + q(1)*sin(q(0))*cos(q(0) - q(2))*cos(q(3) - q(4))
    - q(6)*sin(q(5))*pow(cos(q(3) - q(4)),2)*pow(sin(q(0) - q(2))
    ,2) + q(6)*sin(q(5))*pow(cos(q(3) - q(4)),2) - q(1)*cos(q(0))*
    cos(q(3) - q(4))*sin(q(0) - q(2)) + q(6)*sin(q(5))*pow(sin(q(0)
     - q(2)),2)*pow(sin(q(3) - q(4)),2) - q(6)*sin(q(5))*pow(sin(q
    (3) - q(4)),2) + T*sin(q(3))*sin(q(3) - q(4));
292
293     // cuarta fila
294     J(3,0) = -cos(q(7) - q(5))*sin(q(3) - q(4));
295     J(3,2) = cos(q(7) - q(5))*sin(q(3) - q(4));
296     J(3,3) = -sin(q(7) - q(5));
297     J(3,4) = sin(q(7) - q(5));
298
299     // quinta fila
300     J(4,0) = sin(q(7) - q(5))*sin(q(3) - q(4));
301     J(4,2) = -sin(q(7) - q(5))*sin(q(3) - q(4));
302     J(4,3) = -cos(q(7) - q(5));
303     J(4,4) = cos(q(7) - q(5));
304
305     // sexta fila
306     J(5,0) = cos(q(3) - q(4));
307     J(5,2) = -cos(q(3) - q(4));
308     J(5,5) = 1;
309     J(5,7) = -1;
310   }
311   return J;
312 }
313
314 MatrixXd pinv(const Ref<MatrixXd>& J){
315   // pinv(a) = a' * inv(a*a')
316   MatrixXd A(6,6);
317   A = J * J.transpose();
318   return J.transpose() * A.inverse();
319 }
```

```
320
321 bool comprobar_limites(double q_real[10]){
322   // float k_min = 1.05;
323   // float k_max = 0.95;
324   bool corregido = false;
325   double minimos[2] = {0.1025, -1.185};
326   double maximos[2] = {0.152, 1.245};
327   for (int i=0; i<8; i++){
328     if (q_real[i] < minimos[0]){
329       q_real[i] = minimos[0];
330       corregido = true;
331     }
332     else if (q_real[i] > maximos[0]){
333       q_real[i] = maximos[0];
334       corregido = true;
335     }
336   }
337   for (int i=8; i<10; i++){
338     if (q_real[i] < minimos[1]){
339       q_real[i] = minimos[1];
340       corregido = true;
341     }
342     else if (q_real[i] > maximos[1]){
343       q_real[i] = maximos[1];
344       corregido = true;
345     }
346   }
347   return corregido;
348 }
```

A.3. Nodo de envío de las acciones

```
import sys

import hyrecro_interfaces.msg
import rclpy

import termios
import tty
from rclpy.node import Node

import time
import adafruit_pca9685
from adafruit_servokit import ServoKit
import serial
import board
import busio

import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

GPIO.setup(20, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(21, GPIO.OUT, initial=GPIO.LOW)


kit = ServoKit(channels=16, address=0x40)

i2c = busio.I2C(board.SCL, board.SDA)
pca = adafruit_pca9685.PCA9685(i2c)

pca.frequency = 50
# Actuador 1 - r2b
kit.servo[0].set_pulse_width_range(1020,2000)
# Actuador 2 - r1b
kit.servo[1].set_pulse_width_range(1020,2000)
# Actuador 3 - l2b
kit.servo[2].set_pulse_width_range(1030,1990)
# Actuador 4 - l1b
kit.servo[3].set_pulse_width_range(1020,1990)
# Actuador 5 - l1a
kit.servo[4].set_pulse_width_range(1020,2000)
# Actuador 6 - l2a
kit.servo[5].set_pulse_width_range(1020,2000)
# Actuador 7 - r1a
kit.servo[6].set_pulse_width_range(1030,1990)
# Actuador 8 - r2a
kit.servo[7].set_pulse_width_range(1020,1990)
# Servo 1 - theta_a
kit.servo[8].set_pulse_width_range(820, 2220)
# Servo 2 - theta_b
kit.servo[9].set_pulse_width_range(820, 2220)

class enviar_acciones(Node):

    def __init__(self):
        super().__init__('enviar_acciones')
```

```
        self.qreal_sub = self.create_subscription(
            hyrecro_interfaces.msg.QReal,
            'q_real',
            self.qreal_callback,
            10)
        self.qreal_sub  # prevent unused variable warning
        self.status_sub = self.create_subscription(
            hyrecro_interfaces.msg.Status,
            'status',
            self.status_callback,
            10)
        self.status_sub
        self.iman_a_prev = False
        self.iman_b_prev = False

    def qreal_callback(self, msg):
        frac0 = 19.74485532*msg.r_2b - 2.017023263
        frac1 = 19.79316766*msg.r_1b - 2.012744736
        frac2 = 19.97526575*msg.l_2b - 2.030714879
        frac3 = 20.08538141*msg.l_1b - 2.056976419
        frac4 = 19.97563427*msg.l_1a - 2.036362079
        frac5 = 19.91650562*msg.l_2a - 2.032913254
        frac6 = 19.71144891*msg.r_1a - 2.015328765
        frac7 = 19.86222255*msg.r_2a - 2.026223286
        frac8 = 0.409255568*msg.theta_a + 0.4904761905
        frac9 = 0.4058451049*msg.theta_b + 0.480952381

        print(frac0, frac1, frac2, frac3, frac4, frac5, frac6,
    frac7, frac8, frac9)

        kit.servo[0].fraction = frac0
        kit.servo[1].fraction = frac1
        kit.servo[2].fraction = frac2
        kit.servo[3].fraction = frac3
        kit.servo[4].fraction = frac4
        kit.servo[5].fraction = frac5
        kit.servo[6].fraction = frac6
        kit.servo[7].fraction = frac7
        kit.servo[8].fraction = frac8
        kit.servo[9].fraction = frac9

    def status_callback(self, msg):
        if msg.iman_a != self.iman_a_prev:
            GPIO.output(20, GPIO.HIGH)
            time.sleep(0.5)
            GPIO.output(20, GPIO.LOW)
        self.iman_a_prev = msg.iman_a

        if msg.iman_b != self.iman_b_prev:
            GPIO.output(21, GPIO.HIGH)
            time.sleep(0.5)
            GPIO.output(21, GPIO.LOW)
        self.iman_b_prev = msg.iman_b


def main(args = None):
    rclpy.init(args=args)
```

```
112
113     minimal_subscriber = enviar_acciones()
114
115     rclpy.spin(minimal_subscriber)
116
117     minimal_subscriber.destroy_node()
118     rclpy.shutdown()
119     GPIO.cleanup()
120
121 if __name__ == '__main__':
122     main()
```