UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Diseño e implementación de un sistema de tracción basado en ROS para la plataforma robótica de rehabilitación SWalker

ÁLVARO SALA AYALA

2022

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Diseño e implementación de un sistema de tracción basado en ROS para la plataforma robótica de rehabilitación SWalker

Autor
ÁLVARO SALA AYALA

Tutores

ÁLVARO GUTIÉRREZ MARTÍN
GABRIEL ALFONSO DELGADO OLEAS

2022

Resumen

La fractura de cadera es una lesión que afecta a más de 50000 personas en España cada año, generando un gran impacto económico y en la calidad de vida de los pacientes. La reabilitación de estas lesiones se basa en el ejercicio físico y el movimiento en sesiones de terapia, por lo que en los últimos años se han popularizado los exoesqueletos de rehabilitación en este ámbito.

El Grupo de Ingeniería Neural y Cognitiva (gNec) del CAR-CSIC, ha desarrollado un robot de rehabilitación para personas de avanzada edad que han sufrido una fractura de cadera, el SWalker. El SWalker introduce una rehabilitación orientada a la marcha con numerosos subsistemas que permiten acelerar la rehabilitación de maner segura y eficaz.

En este Trabajo Fin de Grado se pretende diseñar, desarrollar e implementar un nuevo sistema de tracción para el SWalker usando motores sin escobillas y un sistema de control escalable basado en la arquitectura ROS. Adicionalmente, el sistema deberá poder usarse remotamente y seguir rutas basadas en coordenadas para replicar el patrón de movimiento de una sesión de terapia.

Palabras clave: Rehabilitación asistida por robots, Sistema de tracción, Exoesqueleto, Fractura de cadera.

Abstract

Hip fracture is an injure that affects more than 50000 people every year in Spain alone, generating an economical impact and worsening the patient's life. Because rehabilitation of these injuries is based on exercise and movement in therapy sessions, exoskeletons have emerged over the last years to aid therapists in that task.

The "Grupo de Ingeniería Neural y Cognitiva (gNec)" of the "CAR-CSIC" has developed the SWalker, a rehabilitation robot for senior patients who have suffered a hip fracture. The SWalker introduces a gait-oriented rehabilitation with numerous subsystems that aid in achieving a faster recovery while being safe and effective.

In this Final Year's Proyect, a new traction system will be designed, developed and implemented in the SWalker, using brushless motors and a scalable control system based on ROS. Additionally, the system must be usable remotely and must be able to follow a coordinate-based path to simulate the movement pattern in a therapy sesion.

Keywords: Robot assisted rehabilitation, Traction System, Exoskeleton, Hip fracture.

Agradecimientos

Me gustaría agradecer en primer lugar a mi tutor y profesor Álvaro Gutiérrez por darme la oportunidad, los recursos y el apoyo necesarios para hacer el TFG en el campo que me apasiona. Gracias por confiar en mí, por ofrecerme tantas oportunidades únicas y ayudarme a crecer como ingeniero.

En segundo lugar, agradecer a todo el equipo del CAR-CSIC y en especial a mi tutor Gabriel Delgado, por ayudarme siempre que lo necesitaba e invitarme al CSIC cuando estaba atascado. Nunca olvidaré vuestra ayuda.

También quiero agradecer a mis amigos, tanto en Tres Cantos como en la Escuela, que me han proporcionado una red de apoyo sin la que no podría haber conseguido llegar hasta aquí. En especial a Okuyasu por echarme siempre una mano.

Por último me gustaría agradecer a mi familia, mis padres José Ramón y Lola y mis hermanas Laura y Elena que siempre han confiado en mí y me han apoyado en todas mis decisiones. Os llevo siempre en el corazón.

Índice general

Índice de figuras

Índice de tablas

Lista de Acrónimos

CAR: Centro de Automática y Robótica.

CSIC: Centro Superior de Investigaciones Científicas

ROS: Robotic Operating System

gNec: Grupo de Ingeniería Neural y Cognitiva

SSH: Secure Shell

DC: Direct Current

MIT: Massachusetts Institute of Technology

ACK: Acknowledgment

SQL: Structured Query Language

Capítulo 1

Introducción

En esta Sección se analizará el problema de la fractura de cadera y el impacto de la rehabilitación basada en robots. Se introducirán diversos exoesqueletos de rehabilitación antes y se explicará la plataforma en la que se va a basar el trabajo, el SWalker.

1.1. Fractura de cadera en personas de avanzada edad y su rehabilitación

Debido al aumento de la esperanza de vida de la poblacioń [1], las lesiones debidas a fracturas osteoporóticas se han visto incrementadas especialmente en personas ancianas. En especial, las fracturas de cadera (fractura del tercio proximal del fémur) se ven acentuadas. Solo en España se registran cada año más de 50.000 casos de fractura de cadera al año [2]. Este tipo de lesiones tiene un gran impacto sanitario, social y económico.

Las causas pueden ser diferentes pero la mayoría se deben a caídas que pueden ser causadas directamente por patologías anteriores o indirectamente por la consumición de medicamentos psicotrópicos para tratar otras condiciones [3][1]. Los agravantes de las fracturas de cadera son varios, pero destacan el estado de ánimo del paciente (pacientes con síntomas de depresión tras la fractura ven una mejora reducida en la rehabilitación), el estado nutricional, la falla cardíaca y el miedo a caer post fractura [1].

En cuanto a la rehabilitación, se centra en la recuperación de la movilidad del paciente mediante ejercicio físico. El ejercicio físico es vital para la recuperación total o parcial de la movilidad de este, así como para reducir la probabilidad de volver a sufrir una caída [4][5]. El problema es que los ejercicios físicos son muy difíciles para personas de avanzada edad que han sufrido una fractura de cadera. Es por esto que se han desarrollado múltiples plataformas robóticas y exoesqueletos para asistir al paciente durante las sesiones de rehabilitación.

1.1.1. Rehabilitación asistida

Estas plataformas robóticas pueden tener diversas funcionalidades y complejidad, pero generalmente tienen tres subsistemas comunes:

- Soporte del peso: para sostener al paciente; suele ser regulable para poder ajustar la carga y altura de la persona.

- Articulaciones: pueden ser activas o pasivas. Las articulaciones activas disponen de motores y realmente ayudan a la marcha, mientras que las pasivas suelen ser rodamientos para acompañar a la marcha. A parte de las diferencias físicas, la percepción del paciente es completamente diferente para cada una. Las activas restringen más el movimiento, pero ayudan a la marcha, mientras que las pasivas no aportan ayuda pero permiten mover la articulación con más libertad.
  No todas las articulaciones del robot tienen que ser necesariamente activas, lo más normal es encontrar una mezcla. La cadera será siempre una articulación activa ya que es el punto más crítico, pero la rodilla no tiene por qué serlo. Los tobillos casi siempre son articulaciones pasivas para proporcionar más movilidad al paciente.
- Sistema de tracción: para desplazar al paciente. No siempre será un sistema de empuje, también existen algunos basados en cintas ergométricas (comúnmente cintas de correr).
- Rotación de cadera: es común olvidarse de la rotación de cadera cuando se habla de reproducción de la marcha, pero es una parte esencial si se desea replicar de manera fiel la manera de andar de una persona. En los robots de rehabilitación no se suele encontrar este subsistema ya que se centran en el movimiento de la pierna como método de rehabilitación. La falta de este subsitema genera la necesidad de usar algún tipo de apoyo (muletas o andador) para ayudar a la marcha del paciente.

1.2. Métodos de rehabilitación asistida existentes

Existen en el mercado varios modelos de robots para asistencia en terapia. Estos varían en precio según la cantidad de funcionalidades que tengan.

1.2.1. EksoNR

EksoNR [6] es un exoesqueleto desarrollado por *ekso bionics* para ayudar a la neuro rehabilitación. No dispone de sistema de tracción, pero cuenta con articulaciones activas en la cadera y rodillas. Las articulaciones de los tobillos son pasivas.

Tampoco tiene un sistema para soportar el peso del paciente, pero la disposición del robot consigue que este solo tenga que aguantar su propio peso.

Figura 1.1: EksoNR

1.2.2. Gait Trainer GT II

Gait Trainer GT II [7] es la segunda generación de plataformas de rehabilitación de la marcha desarrolladas por Reha-Stim Medtec (ver Figura 1.2). Se centra en rehabilitación de lesiones ortopédicas y parapléjicas. Como sistema de soporte de peso tiene un arnés con un elevador eléctrico para levantar al paciente en caso de que se encuentre sentado en una silla de ruedas; lo que facilita la labor del terapeuta.

No dispone de articulaciones, si no que usa un sistema parecido a una bicicleta elíptica. Con todo el peso soportado por el arnés, el paciente solo tiene que concentrarse en andar hacia delante de la manera que le indique el terapeuta.

Figura 1.2: GaitTrainer GTII

1.2.3. Lokomat

Desarrollado por Hocoma, Locomat [8] es un exoesqueleto que unifica todos los subsitemas mencionados en la sección 1.1.1 (ver Figura 1.3). Al igual que el Gait

Trainer GT II, usa un arnés para el soporte de peso, además de incluir articulaciones activas en la cadera y rodillas y una cinta ergométrica como sistema de tracción. Además cuenta con un eje que permite la rotación de cadera, por lo que dispone de todos los subsistemas que permiten replicar la marcha normal.

Figura 1.3: Lokomat Pro de Hocoma

1.3. Plataforma SWalker

La plataforma SWalker [9] es un proyecto del Grupo de Rehabilitación del Grupo de Ingeniería Neural y Cognitiva (gNec), CAR-CSIC que prentende crear una estructura para la rehabilitación segura de fractura de cadera. Cumple tres objetivos principales para ser un sistema de rehabilitación válido: ser seguro y útil para los pacientes, facilitar el soporte del paciente y su movilidad, y monitorizar variables para determinar la cantidad de rehabilitación necesaria y el progreso.

Figura 1.4: SWalker

1.4. Motivación y objetivos

Ya que la rehabilitación asistida con robots puede ayudar a recuperar la movilidad del paciente, este Trabajo Fin de Grado prentende realizar mejoras en la plataforma SWalker de una manera robusta y escalable. La principal motivación de estas mejoras es que el sistema de tracción actual del SWalker no está automatizado y es ruidoso debido a la caja de engranajes de los motores. Para ello, se analizarán primero los sistemas a implementar y después se realizará la remodelación en tres niveles:

- Estructura: reconstruir el bastidor, introduciendo nuevos motores y encoders de manera resistente y duradera para sustituir a los motores anteriores que introducen ruido.
- Hardware: implementar un controlador de motores programable y realizar la calibración del sistema, usando como centro de control una Raspberry Pi 4.
- Software: implementar un sistema escalable basado en ROS y un sistema de conectividad basado en SSH para que sea posible conectarse desde un dispositivo externo.

1.5. Estructura del documento

El documento se organiza en los siguientes capítulos.

En el Capítulo 1 se resume el problema de fractura y los robots de asistencia a la rehabilitación existentes. En el Capítulo 2 se realiza un análisis de los motores y tarjetas de control existentes, así como una comparación y selección del hardware a utilizar. El Capítulo 3 describe los elementos hardware y software del sistema y su esquema de conexión. En el Capítulo 4 se desarrolla el montaje de la estructura hardware y software, introducida en el Capítulo 3. El Capítulo 5 muestra un análisis de los resultados de movimiento de la plataforma. Por último el Capítulo 6 contiene un resumen del proyecto y el futuro de la plataforma SWalker.

Capítulo 2

Estado del arte

2.1. Estado actual del SWalker

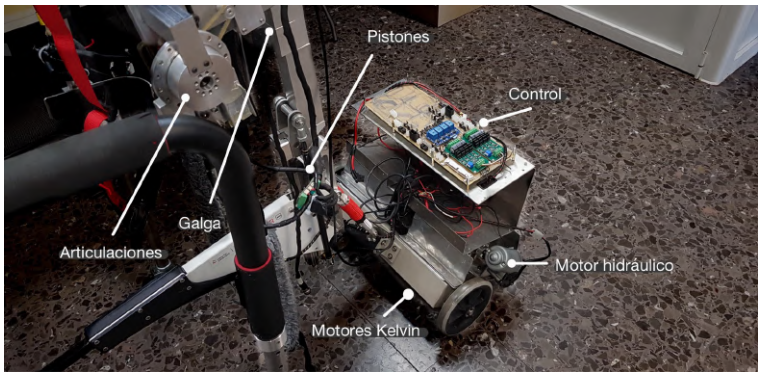


Figura 2.1: Subsistemas del SWalker

El sistema de tracción actual del SWalker se basa en dos motores Kelvin [10] con encoders HEDS-5540 [11] (ver Figura 2.2). Estos motores incluyen su propia reductora y disponen de un par máximo de 8Nm. El problema con estos motores es su y que generan un alto nivel de ruido debido a la presencia de una caja de engranajes.

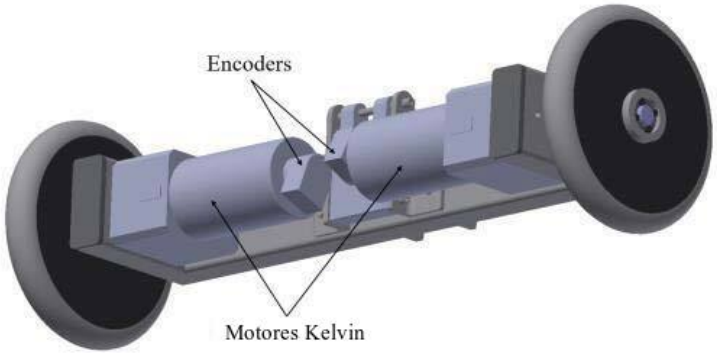


Figura 2.2: Sistema de tracción actual [9]

La medición de peso del paciente sujeto a la plataforma SWalker se realiza mediante una galga SB-FT1 [12] (ver Figura 2.1) que soporta hasta 350 Kg con una sensibilidad nominal de $0{,}1\,\% = 0{,}35kg$. La elevación del paciente se realiza mediante dos actuadores lineales E21BX300-U-001 [13] (ver Figura 2.1). Estos son pistones neumáticos alimentados a 24V con una fuerza máxima de 2200N. Encima de los motores kelvin se encuentra el motor que comprime el aire

Las articulaciones de la cadera van sujetas mediante dos barras de aluminio. Tienen un grado de libertad en el plano sagital y permiten un movimiento de $90^{\underline{o}}$. La posición se mide con un potenciometro (ver Figura 2.3).

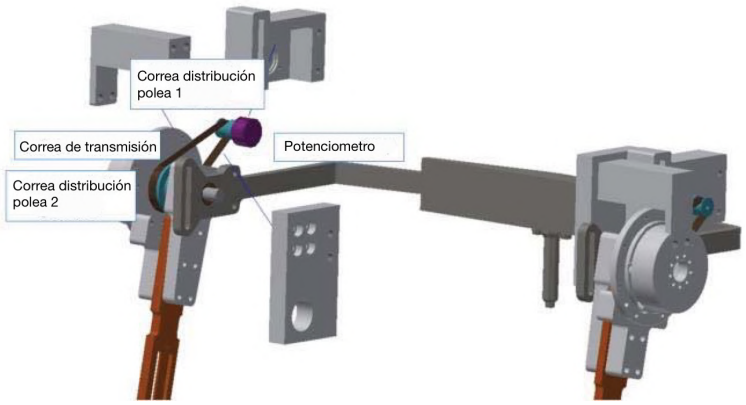


Figura 2.3: Esquema exoesqueleto de cadera [9]

2.2. Motores para plataformas robóticas

Los motores DC que podemos encontrar en la robótica actual se dividen en dos grupos; con escobillas y sin escobillas. De estos derivan una serie de motores con aplicaciones concretas.

- *Con escobillas:* se basan en un rotor con electroimanes que gira contenido en un estator formado por imanes permanentes. Las escobillas conmutan cuando el rotor gira e invierten la corriente para que siga girando en la misma dirección (ver figura 2.4). Este funcionamiento tiene la ventaja de que no necesita un controlador externo, pero también tiene desventajas importantes.

  El principio de funcionamiento tiene como inconveniente la presencia de un punto de torque nulo. Cuando el plano de la bobina es paralelo al campo magnético, el par motor se anula. Gracias a la inercia que lleva el rotor es posible escapar de este punto, pero la eficiencia del motor y el par efectivo se ven reducidos. Otro inconveniente es que en este punto las escobillas generan un cortocircuito que no es lo suficientemente relevante como para dañar el circuito pero que consume potencia y reduce su eficiencia.

  Por estas razones, la principal desventaja de los motores con escobillas es un par muy bajo. Además la fricción de las escobillas genera un desgaste y acorta el tiempo de vida útil del motor.

  Para aplicaciones robóticas existen combinaciones basadas en el motor con escobillas, fundamentalmente:

  - *Motor DC con reductora:* la solución inmediata es usar una caja de engranajes para reducir la velocidad y aumentar el par según la necesidad. Gracias a la variedad de reductoras exsistentes esta solución está muy extendida, pero tiene el problema de que la caja de engranajes introduce mucho ruido acústico.
  - *Servos:* para aplicaciones que requieren poco par pero más precisión existen los servos, que se componen de un motor con reductora y un lazo de control para regular la posición mediante un encoder.

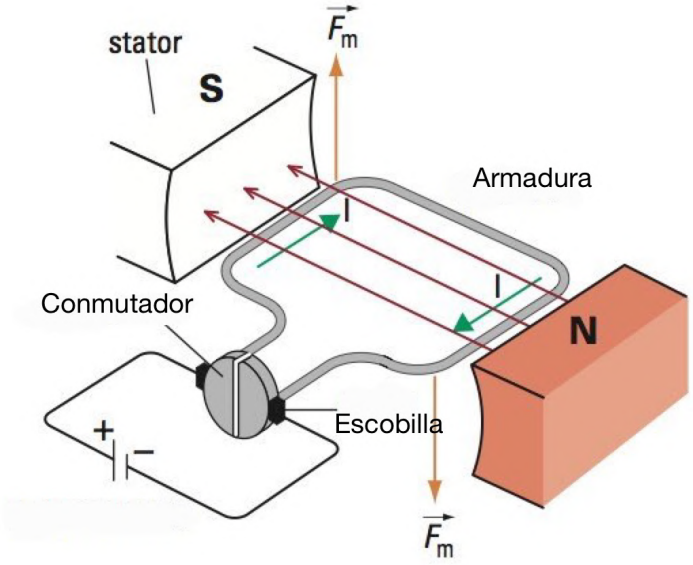


Figura 2.4: Funcionamiento de un motor con escobillas [14]

- *Sin escobillas:* su principio de operación sigue siendo electroimanes en el rotor e imanes permanentes en el estator (ver Figura 2.5), pero usa un sistema de control para mantener el campo magnético de ambos desbalanceado. Este proceso asegura que no hay puntos con par nulo, aunque el par disminuirá con la velocidad. Tampoco tiene fricciones a parte del rodamiento, por lo que es más duradero que el motor con escobillas. El inconveniente de estos motores es su alto precio debido a la necesidad de incorporar un sistema de control externo.

    - *Motor paso a paso:* al igual que con los motores con escobillas, una variante muy usada son os motores paso a paso. Estos siguen el mismo principio que su predecesor pero cambiando los electroimanes al estator y los imanes permanentes en el rotor. De este modo se puede girar el rotor en pasos de ángulo constante, α, siguiendo la fórmula

$$\alpha = \frac{2\pi}{N}[Rad] \tag{2.1}$$

    dónde N es el número de electroimanes en el estator.

Figura 2.5: Ejemplo de motor sin escobillas [15]

Capítulo 3

Diseño conceptual

3.1. Arquitectura hardware

La arquitectura hardware se basa en la conexión de los encoders, motores, tarjeta de control y Raspberry Pi. Un diagrama simplificado de conexión se puede ver en la Figura 3.1.

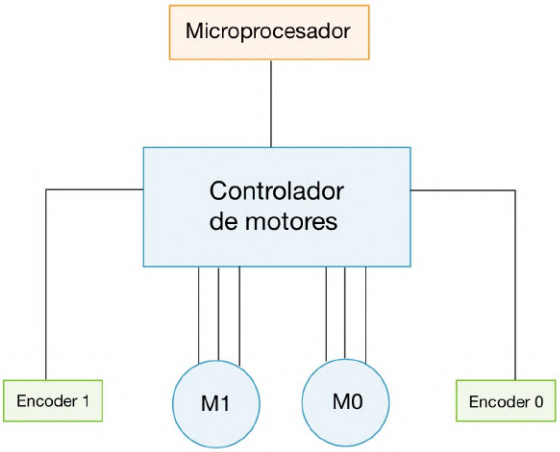


Figura 3.1: Esquema hardware simplificado

3.2. Comparación de motores sin escobillas

Tras analizar los motores, la mejor solución para mover el SWalker serán los motores sin escobillas ya que necesitamos un gran torque a bajes velocidades y un control fino.

A continuación compararemos tres motores diferentes comunes en aplicaciones robóticas: *Turnigy Multistar* 100kv [16], *ODrive dual shaft* 150kv [17] y un motor de hoverboard [18] (ver Tabla 3.1).

Nuestra aplicación se basa en mover el peso de un paciente de 80 kg como máximo y el peso extra de la plataforma, por lo que nos interesan motores con mucho par y que sean, en la medida de lo posible, económicos. Otro parámetro usado en la comparación

es la constante de velocidad, Kv, que se define como la velocidad media que alcanza el motor sin carga por cada voltio y se mide en RPM/V. Se comparan los tres motores teniendo en cuenta los parámetros anteriores en la Tabla 3.1.

Motor	Kv [RPM/V]	Par [Nm]	Potencia [W]	Precio [€]	Peso [kg]
Turnigy Multistar	149	3.77	2261	102.81	0.674
ODrive dual shaft	150	3.86	2328	109	1
Hoverboard	16	12.92	831	40	3.146

Tabla 3.1: Comparación de motores sin escobillas [19]

El motor Turnigy y el motor de ODrive son parecidos en sus características; alta constante de velocidad y potencia, pero bajo par y alto precio. Sin embargo el motor de hoverboard ofrece mucho más par por menos de la mitad del precio, con el inconveniente de que pesa mucho más. Su baja constante de velocidad no supone un gran problema ya que el ámbito de la rehabilitación no requiere altas velocidades por lo que no necesitaremos altos voltajes. El peso del motor a pesar de ser grande, es similar al de los motores Kelvin anteriores, por lo que no supondrá un problema con una estructura robusta y estable.

Por estas razones se usarán dos motores de hoverboard como sustitución de los motores Kelvin. Para cerrar el lazo de control se utilizará un encoder que se explicará en la Sección 3.2.2.

3.2.1. Motor hoverboard

El motor hoverboard [18] (ver Figura 3.2) consta de 30 imanes permanentes y tiene incorporado un sensor de efecto hall para detectar la posición ya que es un motor sin escobillas. Sin embargo, este sensor tiene una resolución muy baja, de apenas 90 cpr (ciclos por revolución, explicado más adelante) por lo que será necesario utilizar un encoder para detectar la posición.

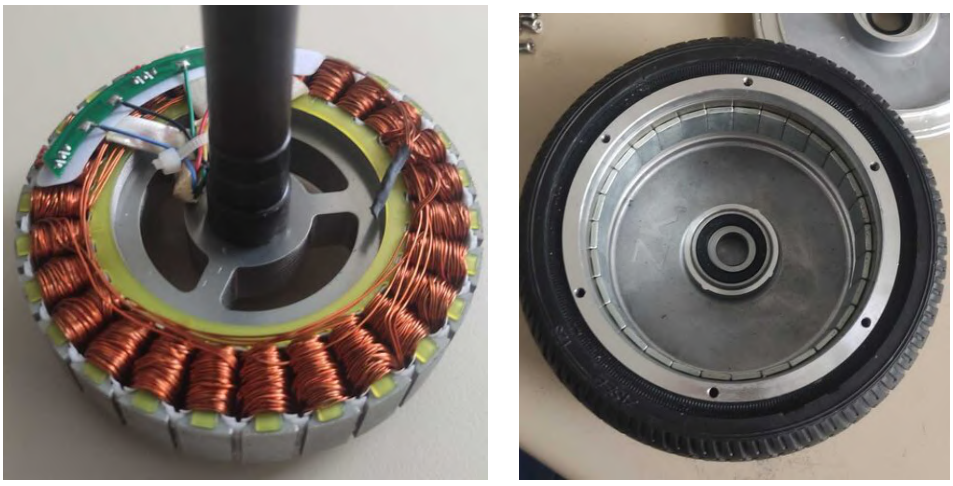


Figura 3.2: Motor hoverboard

3.2.2. Encoder

Los encoders son sensores que permiten determinar la posición de un motor. Su funcionamiento se basa en un disco unido al eje del motor (ver Figura 3.3) que está codificado con bandas transparentes u opacas. Estas dejan pasar o no la luz que emiten unos diodos infrarojos hasta unos fotorreceptores en el otro lado del disco. Con cada rotación del eje se generan una serie de pulsos. De esta manera, se mide la resolución de un encoder en pulsos por revolución (*ppr*), es decir, el número de pulsos generado cuando el eje da una vuelta completa. Un ejemplo de los pulsos generados en un encoder de dos fases en cuadratura se puede ver en la Figura 3.4. La resolución de un encoder se representa también en *counts per revolution*, cpr, que define el número de cambios de estado de la señal generada, en una revolución. En un encoder en cuadratura, como es este caso, se calcula multiplicando por cuatro el valor en ppr.

Para el sistema usaremos un encoder genérico de 600 pulsos por revolución [20] (ver Figura 3.5) ya que tiene resolución mas que suficiente para las velocidades que alcanzará el robot y es económico. Para acoplarlo a las ruedas se usarán engranajes con una relación de 2:1, por lo que los pulsos por revolución efectivos serán $600 * 2 = 1200 ppr$.

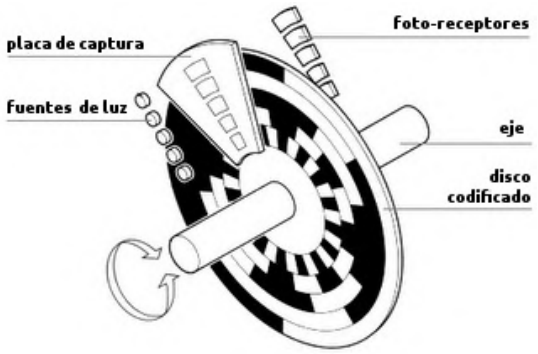


Figura 3.3: Funcionamiento de un encoder incremental [21]

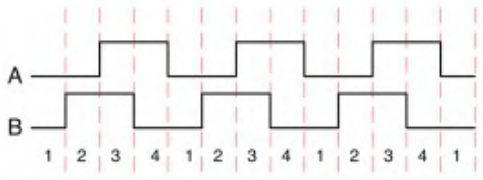


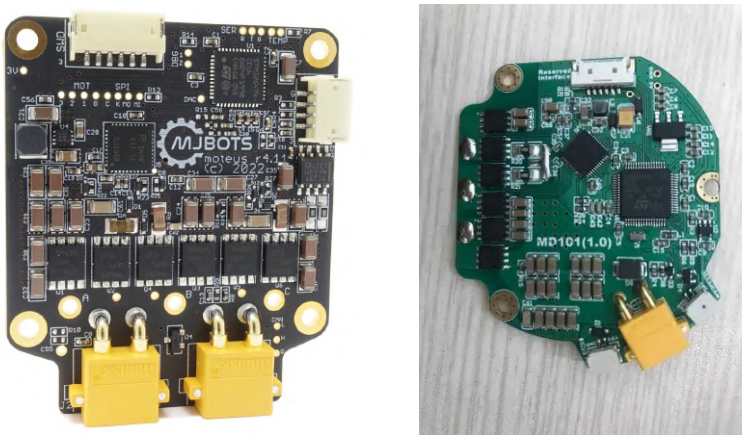
Figura 3.4: Fases de un encoder en cuadratura [22]

Figura 3.5: Encoder de 600 ppr

3.2.3. Tarjetas de control

Al usar motores sin escobillas, como hemos visto en la sección anterior, es imperativo un sistema electrónico de control de lazo cerrado. En el mercado existen varias opciones, algunas de las más populares son la tarjeta *Moteus r4.11* (ver Figura 3.6*(a)*) y la *Cheetah mini* del MIT (ver Figura 3.6*(b)*). Ambas proporcionan voltajes máximos de 44V, corriente máxima de 100A por fase, con una frecuencia de control de 40kHz.

Para este TFG se ha elegido la controladora de motores ODrive v3.6 [23] (ver Figura 3.7). Esta tarjeta de control fue desarrollada por *ODrive Robotics*, que también diseñan motores (uno de ellos se analizó en la Sección 3.2). Esta tarjeta es la opción elegida ya que dispone de dos drivers que soportan un motor de tres fases y un encoder cada uno. De este modo, con una sola tarjeta se puede implementar el sistema de tracción entero. Sus características eléctricas permiten un voltaje de entrada máximo de 56V y 120A por motor. Con respecto al sistema de control, es completamente programable por USB o SPI. Gracias a esta fácil comunicación se puede implementar dentro de un sistema de alto nivel como ROS, que se desarrollará e la Sección 3.3.1.



(a) *(b)*

Figura 3.6: Controladores Moteus r4.11 *(a)* [24] y MIT Cheetah mini *(b)* [25]

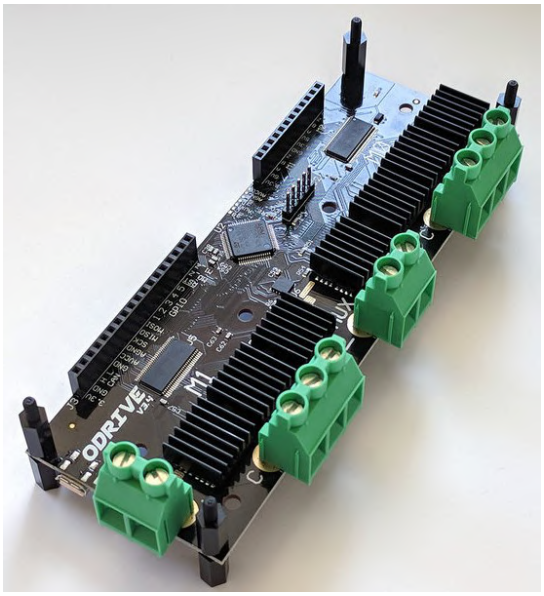


Figura 3.7: Controlador ODrive v3.6 [23].

A la hora de usar el controlador ODrive v3.6 se deben tener dos cosas en cuenta:

- Alimentación: existen dos modelos de ODrive; uno de 24V y otro de 56V. Estos admiten voltajes de entrada de 12-24V y 12-56V, respectivamente. Se elige el modelo de 56V ya que se pretenden usar tres baterías de 12V y 7 Ah en serie. De este modo se podrá entregar la corriente máxima que soporta el ODrive, 25 A.
- Resistencia de carga: al trabajar con motores con mucha potencia hay que tener en cuenta la corriente que puede regresar a la fuente en la deceleración. Para esto se usa una resistencia de carga que disipe la potencia. Para este trabajo se usa una resistencia de 50W y 2Ω.

Para comunicarse con el ODrive e implementar un sistema escalable basado en ROS, es necesario usar una Raspberry Pi. La razón es que se necesita soporte para un sistema operativo Linux. Los criterios de selección del modelo son que disponga de conectividad WiFi y que soporte el sistema operativo seleccionado: Ubuntu 20.04. Por estas razones se escoge una Raspberry Pi 4.

Finalmente, se recoge el diseño completo en la Figura 3.9.

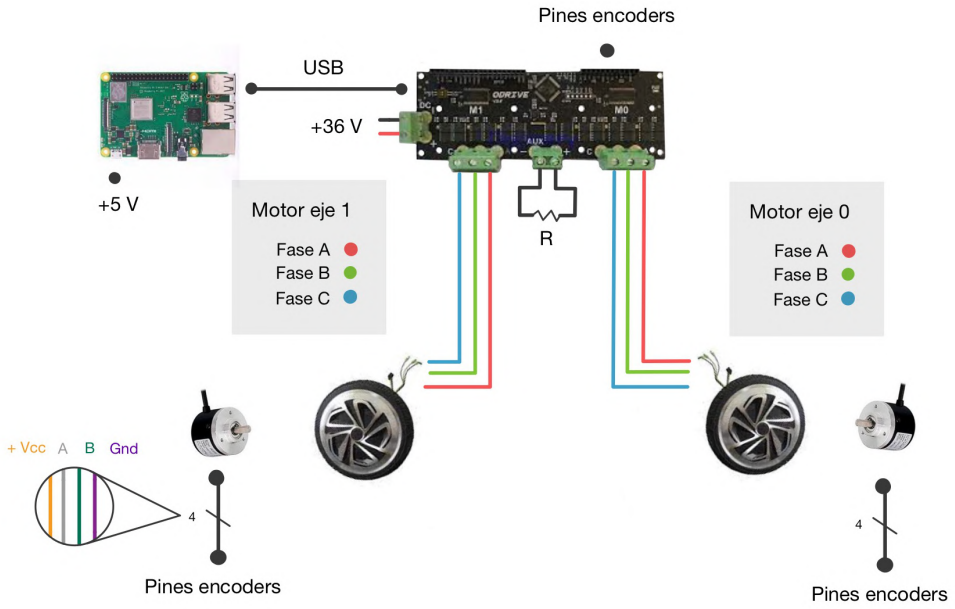


Figura 3.8: Diseño hardware del controlador de motores. La resistencia R hace referencia a la resistencia de carga de 50W

3.3. Arquitectura software

3.3.1. Robotic Operating System

Se pretende que el sistema sea escalable para que en un futuro se puedan introducir los demás subsistemas del SWalker. Para esto se usará *Robotic Operating System* (ROS) [26]. A pesar de su nombre, ROS no es un sistema operativo, si no un conjunto de librerías y herramientas para facilitar la construcción de robots. La idea principal es encapsular los datos de los sensores, actuadores y controladores en unos componentes llamados nodos. La ventaja, es que los nodos se comunican sin importar el código fuente de cada elemento. Estos nodos se comunican de dos formas diferentes:

- *Topics* [27]: conectan uno o más nodos subscriptores con un nodo publicador. El nodo publicador envía información periódicamente en *broadcast* a los nodos subscriptores. Es la forma más básica de comunicación en ROS.

- Servicios [28]: conectan uno o varios nodos cliente con un nodo servidor. Los clientes envían peticiones y el servidor responde.

De esta manera podemos crear un nodo publicador que envíe, por ejemplo, la velocidad de los motores desde el ODrive. La Raspberry Pi actuaría como nodo subscriptor leyendo los datos y representándolos. Este procedimiento se desarrollará en la Sección 4.3.

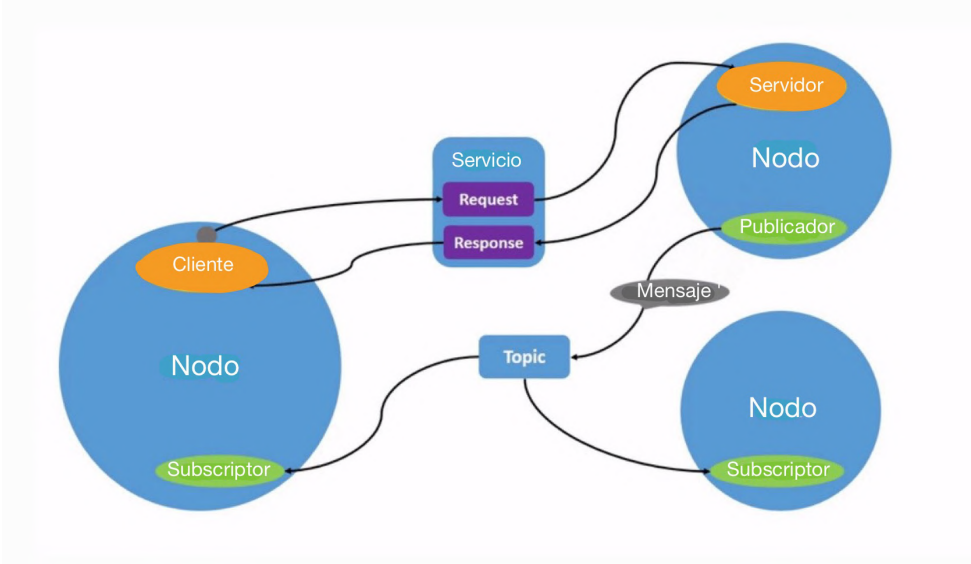


Figura 3.9: Grafo ROS [29]

3.3.2. Raspberry Pi

Para hacer el sistema más portable se controlará la Raspberry Pi desde el móvil mediante SSH. Se usa la aplicación Termux [30], un emulador de terminal Linux para el móvil. A la vez, el dispositivo móvil actúa de *hotspot* para crear la red WiFi que soporta la conexión SSH.

Capítulo 4

Implementación del sistema

4.1. Estructura física

La estructura inicial del SWalker se muestra en la Figura 4.1. Lo que se observa es el bastidor inicial que se descubre al retirar la pieza de aluminio que soporta la electrónica, las baterías y el motor que controla la bomba hidráulica. Los motores Kelvin van sujetos al bastidor a presión por lo que es necesario rehacer todo el armazón.

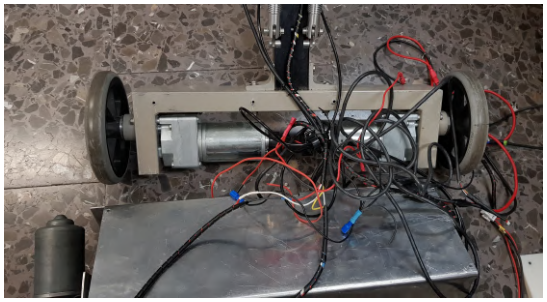


Figura 4.1: Bastidor antiguo

Para construir la estructura se usarán perfiles de Bosch de 40x40 mm [31]. Estos son listones de aluminio con railes para atornillarlos y montar estructuras. Se planea montar los motores siguiendo el esquema de la Figura 4.2; donde la pieza *a* es el perfil, la pieza *b* es el engranaje unido al motor hoverboard (se puede ver en la Figura 4.3), la pieza *c* es el engranaje que une el encoder, la pieza *d* es el encoder y la pieza *e* sirve para sujetar el encoder al perfil.

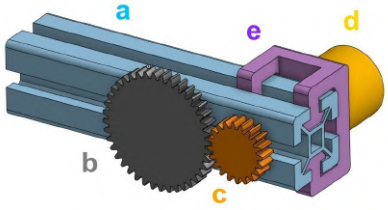


Figura 4.2: Esquema del motor completo

El primer paso es cortar los perfiles a medida con una sierra de mano y taladrar los agujeros para el motor y el engranaje del encoder en el perfil. Se usa un taladro de columna y se lima hasta que el motor entra perpendicularmente, como se ve en la Figura 4.3.

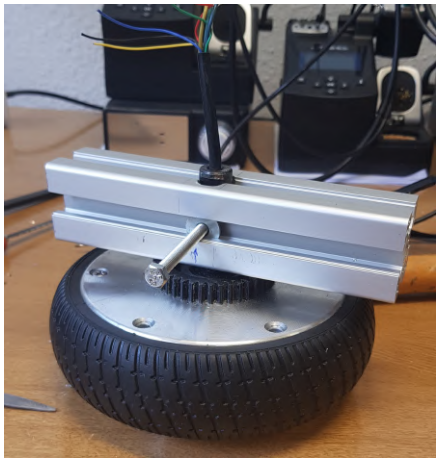
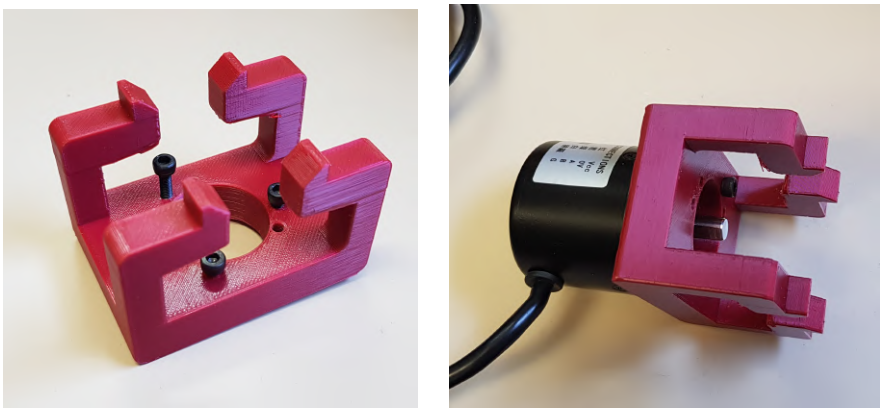


Figura 4.3: Motor unido al perfil

El siguiente paso es diseñar en 3D la sujección del encoder (pieza *e* en la Figura 4.2) y el engranaje del encoder (pieza *c* en la Figura 4.2). La pieza que sujeta el encoder se desliza en el perfil para poder ajustarse a la posición deseada (ver Figura 4.4). Por último se montan los componentes en el perfil. El orden que se sigue al



(a) *(b)*

Figura 4.4: Sujección encoder *(a)* y encoder completo *(b)*

montarlo es importante para que todos los elementos queden alineados. Primero se atornilla el encoder a la sujección de encoder como en la Figura 4.4 *(b)*. Después se desliza hasta la posición deseada y se acopla el engranaje del encoder. Por último se introduce el motor y se fija con un tornillo y una tuerca. El montaje completado se puede ver en la Figura 4.5.

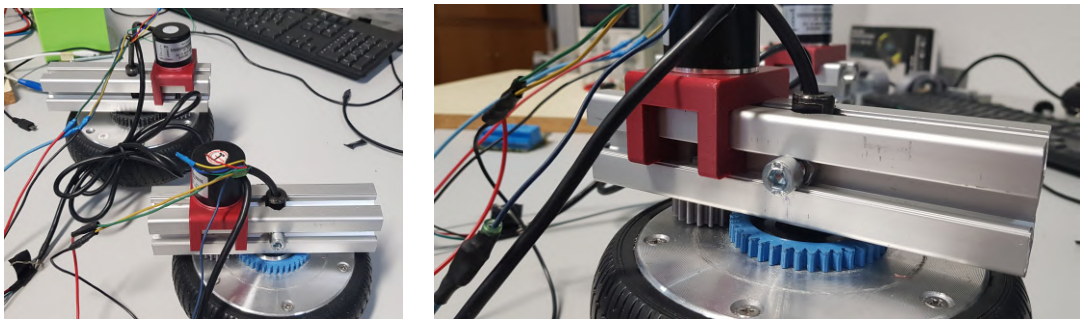


Figura 4.5: Motores completos

Al montar el nuevo bastidor (ver Figura 4.6), se identifica un problema; los ejes de los motores están desalineados. Tienen una separación de 1.25cm. Este desalineamiento no tiene ningún efecto en el movimiento hacia adelante o hacia atrás, pero si lo tiene en los giros, ya que se deberá girar un ángulo mayor o menor dependiendo del sentido de giro.

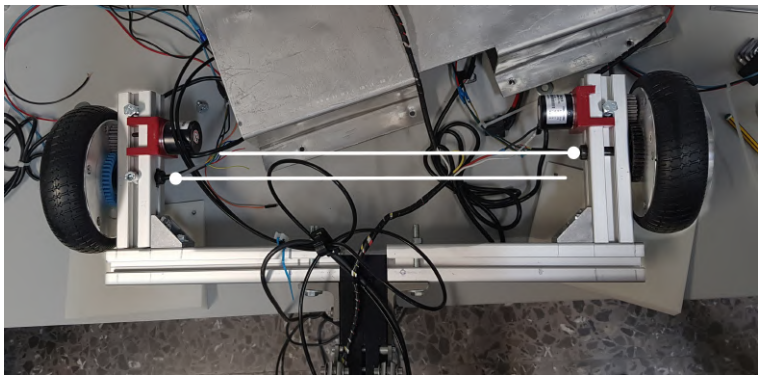


Figura 4.6: Ejes de los motores desalineados

4.1.1. Compensación de los giros

El desalineamiento observado se puede compensar por software, pero se deben tener las siguientes consideraciones:

- Los giros se harán tomando como eje de giro una de las ruedas. Es posible girar la estructura actuando un motor en cada sentido, pero sería incómodo para el paciente.
- Se debe poder introducir un ángulo deseado (α) y que la plataforma gire en consecuencia.
- El ODrive expresa los movimientos de los motores en revoluciones o *turns*.
- El diámetro de la rueda es de 6,5" = 16,51cm.
- Se deben poder hacer giros en sentido horario y antihorario usando como eje de giro tanto la rueda derecha como la izquierda.

Analizamos primero el problema de giro sin tener en cuenta el desalineamiento (ver Figura 4.7), para definir el numero de vueltas que debe dar la rueda (V) para girar un ángulo determinado (α).

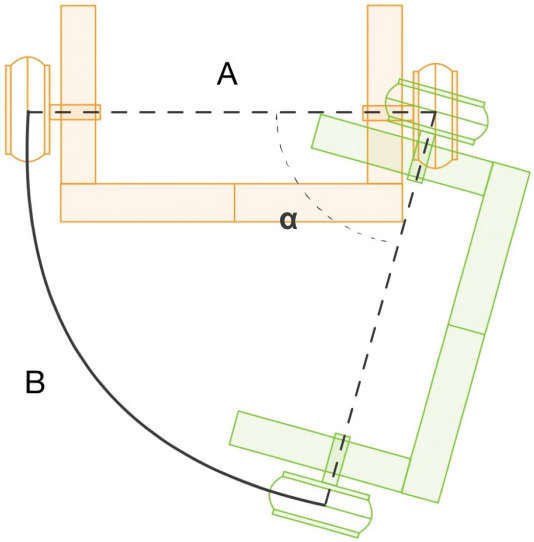


Figura 4.7: Esquema de giro

Se calcula la cuerda del arco con la separación entre ejes (A) en la Ecuación 4.1 y el número de vueltas usando el diámetro de la rueda (D) en la Ecuación 4.2.

$$B = \frac{2\pi A\alpha}{360}[cm] \tag{4.1}$$

$$V = \frac{B}{perimetro} = \frac{2\pi A\alpha}{360\pi D} = \frac{A\alpha}{180D}[vueltas] \tag{4.2}$$

A continuación introducimos el desalineamiento siguiendo la Figura 4.8. Las consecuencias de este desalineamiento son que, dependiendo del sentido de giro (horario o antihorario), el ángulo efectivo que se introducirá en la Ecuación 4.2 se verá incrementado o decrementado.

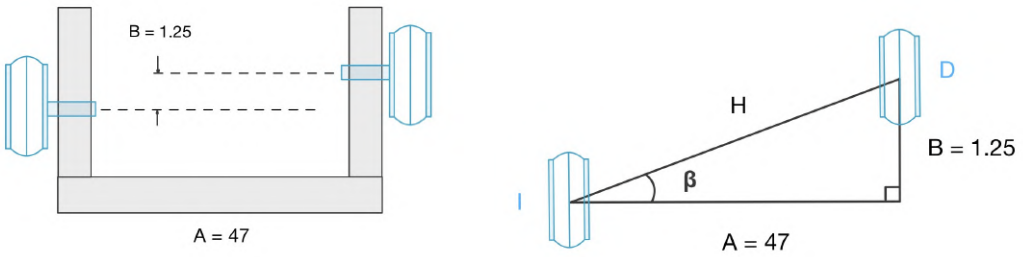


Figura 4.8: Esquema del ajuste

Obtenemos el ángulo de desajuste (β) con la Ecuación 4.3 y la nueva distancia entre ejes con la Ecuación 4.4. Esta será la hipotenusa del triángulo de desajuste de la Figura 4.8. En la Figura 4.9 se representan los cuatro posibles giros. Se puede ver que en los dos giros en sentido antihorario, el ángulo que debe girar la plataforma es $\gamma = \alpha - \beta$, donde α es el ángulo que se desea girar y β es el ángulo de desajuste. En el caso de los giros en sentido horario el ángulo es $\gamma = \alpha + \beta$. Distinguiendo los dos casos de giro antihorario y horario obtenemos, con el procedimiento anterior, las Ecuaciones 4.5 y 4.6, respectivamente.

$$\beta = arctg(\frac{B}{A}) \tag{4.3}$$

$$H = \sqrt{A^2 + B^2} \tag{4.4}$$

$$V_{CCW} = \frac{H(\alpha - \beta)}{180D}[vueltas] \tag{4.5}$$

$$V_{CW} = \frac{H(\alpha + \beta)}{180D}[vueltas] \tag{4.6}$$

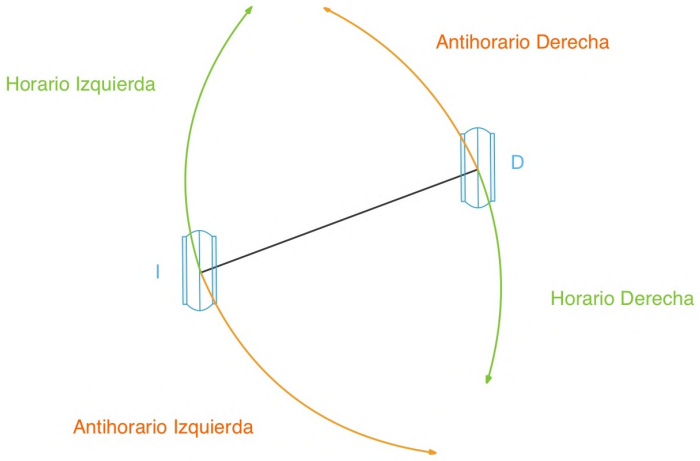


Figura 4.9: Esquema de giro horario y antihorario para la rueda izquierda (I) y derecha (D)

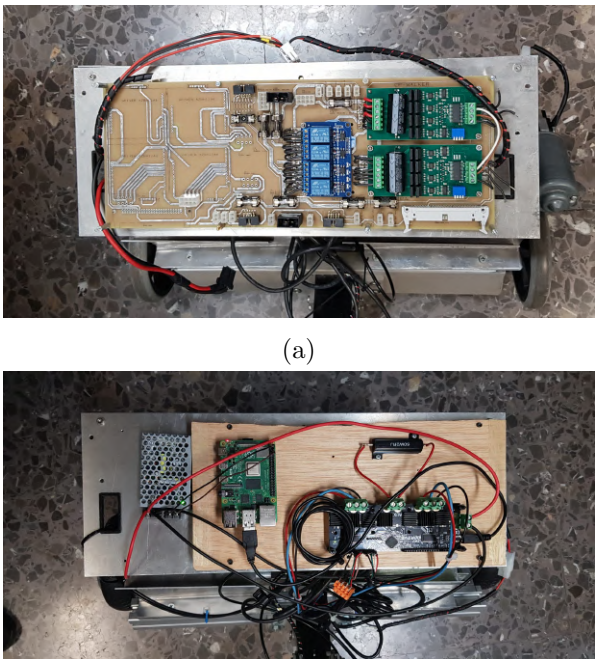
4.2. Estructura hardware

Anteriormente, el control lo llevaba a cabo la tarjeta de la Figura 4.10a. Esta cuenta con una STM32 como centro de control, con una etapa de potencia para cada motor y relés para el resto de sistemas. Este diseño no estaba automatizado y dependía de pulsadores e interruptores, por lo que se introduce el nuevo sistema de control de la Figura 3.9.

En dicha Figura se destacan los voltajes a los que se debe alimentar cada elemento. Los motores hoverboard funcionan a 36V asi que alimentaremos el ODrive con ese voltaje usando tres baterías en serie de 12V [32]. La Raspberry Pi debe alimentarse a

5V. No tendría sentido introducir otra fuente de alimentación teniendo las baterías, así que se usa un *buck converter* de 36V a 5V [33] .

Tras atornillar todos los componentes a una superficie de madera, se añade un interruptor a las baterías para encender y apagar el sistema y se obtiene el resultado de la Figura 4.10b.



(a)

(b)

Figura 4.10: Sistemas de control antiguo *(a)* y nuevo *(b)*

4.2.1. Calibración del sistema

Con todos los componentes montados se puede configurar el ODrive y calibrar los motores. Primero se debe configurar el ODrive con los parámetros específicos del motor hoverboard:

- **Pole pairs**: número de pares de polos magnéticos. Este valor es equivalente a la mitad de los imanes permanentes. El motor que se va a usar tiene 30 imanes (ver Figura 3.2) por lo que este valor valdrá 15.

- **Torque constant**: la constante de par también se debe incluir en la configuración. Se calcula con la constante de velocidad (K_v) en la Ecuación 4.7. Para determinar la constante de velocidad es necesario hacer un *drill test* [34], que consiste en rotar el motor con un taladro y medir el voltaje entre fases. Con un osciloscopio se observa una señal de voltaje sinusoidal. Podemos obtener la velocidad mecánica del motor en vueltas por segundo dividiendo la frecuencia de la señal sinusoidal por los pares de polos magnéticos (15). Multiplicando este valor por 60 obtenemos la velocidad en RPM (revoluciones por minuto). Por último dividiendo el voltaje de pico (en voltios) entre la velocidad del motor (en RPM) obtenemos la constante de velocidad.

Este desarrollo se resume en la Ecuación 4.8, donde f es la frecuencia de la señal sinusoidal y V_p es el voltaje de pico.

$$K_T = \frac{60}{2\pi K_v} \tag{4.7}$$

$$K_v = \frac{f * 60}{15 * Vp} = \frac{4f}{Vp} \tag{4.8}$$

Tras esta configuración se hace una calibración del motor. Así se obtiene el valor de la resistencia e inductancia de cada fase. A continuación se configuran los parámetros del encoder:

- **CPR**: los ciclos por revolución son el número de cambios de nivel en una revolución. Se calcula multiplicando por cuatro el valor de pulsos por revolución (ver Figura 3.4). En este caso debemos tener en cuenta también los engranajes de acople entre el motor y el encoder, con relación 2:1, $CPR = 2*600*4 = 4800$.

- **Encoder mode**: este encoder es de tipo incremental por el comportamiento visto en la Sección 3.2.2.

Por último se configura el lazo de control del ODrive. Observando la Figura 4.11 [35] se ven tres bloques de control: de posición, velocidad y corriente. Cada uno de estos es una variación de un controlador de tipo PID, es decir, que responde al error de manera proporcional, así como a la integral del error y a la derivada del error. Los tres controladores funcionan de la siguiente manera:

- **Controlador de posición**: es de tipo P. Responde proporcionalmente al error según el parámetro *pos_gain*. El error se calcula como la resta entre la posición deseada (la referencia) y la posición obtenida con el encoder.

- **Controlador de velocidad**: de tipo PI, responde tanto al error (parámetro *vel_gain*), como a la integral del error (parámetro *vel_integrator_gain*). El error se calcula también con el encoder.

- **Controlador de corriente**: a pesar de que es un lazo de control de corriente, se usa para controlar el par del motor usando la constante de par (K_T) y la resistencia de fase.

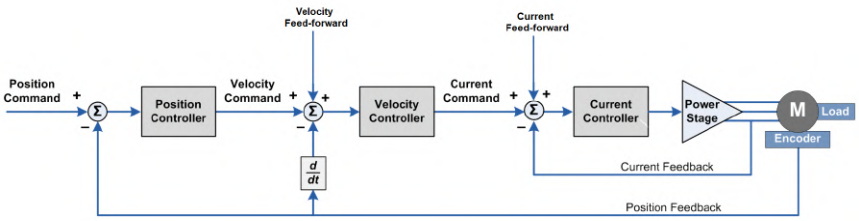


Figura 4.11: Lazo de control del ODrive

Para configurar el lazo de control [35] hace falta modificar los parámetros *pos_gain*, *vel_gain* y *vel_integrator_gain*. Estos dos primeros valores son las ganancias de los controladores de posición y velocidad.

La ganancia de posición se calibra con el motor sin carga. Se le introduce una posición y se aumenta la ganancia hasta que se observe sobreelongación en la respuesta (el motor se pasa de la posición de referencia y tiene que volver en sentido contrario, también llamado *overshoot*) y cuando se observa se fija el valor de *pos_gain* a la mitad de ese valor.

La ganancia en velocidad se calcula también con el motor sin carga. Se aumenta la ganancia en velocidad *vel_gain* hasta que se observan vibraciones en el motor y se escoge la mitad de ese valor.

Para la calibración del parámetro *vel_integrator_gain* [35] se necesita el ancho de banda, o tiempo de respuesta del motor, según la Ecuación 4.9. Se puede calcular con una representación gráfica a una respuesta de tipo escalón. En la Figura 4.12 se introduce una posición de 0 cuando la posición actual es 1 y se observa un tiempo de respuesta de 40ms y un ancho de banda de $BW = \frac{1}{40ms} = 25Hz$.

$$vel\_integrator\_gain = \frac{1}{2} * BW * vel\_gain \tag{4.9}$$

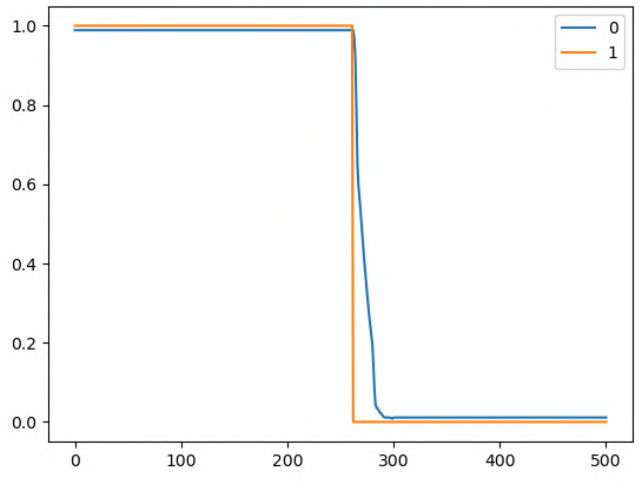


Figura 4.12: Calibración del parámetro *vel_integrator_gain*

4.3. Estructura software

La capa de ROS (ver Figura 4.13) consta de tres nodos:

- **Raspberry Pi**: será el nodo que recibe los datos para representarlos y genera *requests* para definir movimientos del motor.
- **OD_control**: recibe los *requests* del nodo anterior y administra el movimiento

del motor. Responde a las solicitudes con *ACK*.

- **OD_velocidad**: publica periodicamente los datos de velocidad y par del motor.

Para la comunicación entre nodos se crean paquetes personalizados de ROS llamados interfaces. Las interfaces permiten enviar múltiples datos de múltiples tipos en cada transacción. De este modo podemos enviar todos los datos necesarios en cada momento a la vez. La interfaz de comunicación entre la Raspberry y OD_control tiene los siguientes datos:

- **Axis**: el eje del ODrive que se desea configurar. Toma los valores 0 (motor izquierdo) o 1 (motor derecho).
- **Idle**: cuando su valor 1, el motor puede girar libremente. Si vale 0 se comprueban los siguientes parámetros.
- **Calibration**: cuando su valor es 1 se realiza la calibración completa del motor del eje seleccionado. En el caso contrario se evalúan los siguientes parámetros.
- **Giro**: indica si se desea realizar un giro. Puede tomar los valores 0 (no se desea girar), 1 (giro con la rueda izquierda) o 2 (giro con la rueda derecha). Cuando tiene un valor distinto de 0, se configura el motor en modo control de velocidad y se lee el siguiente parámetro. En caso contrario se evalúa *control_mode*.
- **Ang**: el ángulo en grados que se desea girar. Se realizan los cálculos de la Sección 4.1.1 y se realiza el giro. Para realizar el giro, se obtiene la cuenta del encoder actual y se le suma el número de vueltas calculado (V) multiplicado por el CPR. De este modo obtenemos el valor que tomará la cuenta del encoder cuando se haya completado el giro. A continuación se entra en un bucle en el que se mueve el motor hasta que la cuenta actual sea igual a la calculada. El parámetro *ang* puede ser negativo (giro horario) o positivo (giro antihorario) (ver Figura 4.9).
- **Control mode**: tiene tres valores posibles, para tres modos de control diferentes: 1, control de par; 2, control de velocidad; o 3, control de posición. Configura el motor correspondiente para ser controlado con par, velocidad o posición acorde con los siguientes parámetros.
- **Pos, Vel, Torque**: configuran el motor con la posición velocidad o torque indicados. Solo se configura el parámetro correspondiente al modo de control seleccionado en *control_mode*, el resto se ignoran.
- **ACK**: respuesta del servidor con el valor de los parámetros que se han modificado.

La interfaz entre la Raspberry y el nodo OD_velocidad envía los datos calculados de velocidad y par de cada rueda en cuatro parámetros que toman valores reales. El esquema del grafo ROS implementad se encuentra en la Figura 4.13.

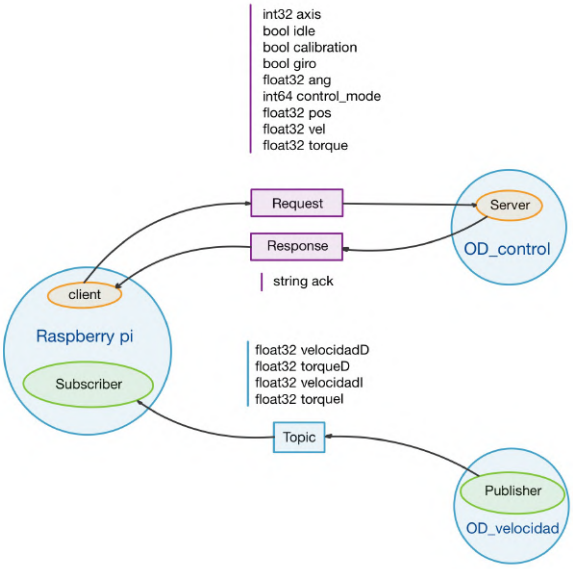


Figura 4.13: Esquema de conexiones ROS

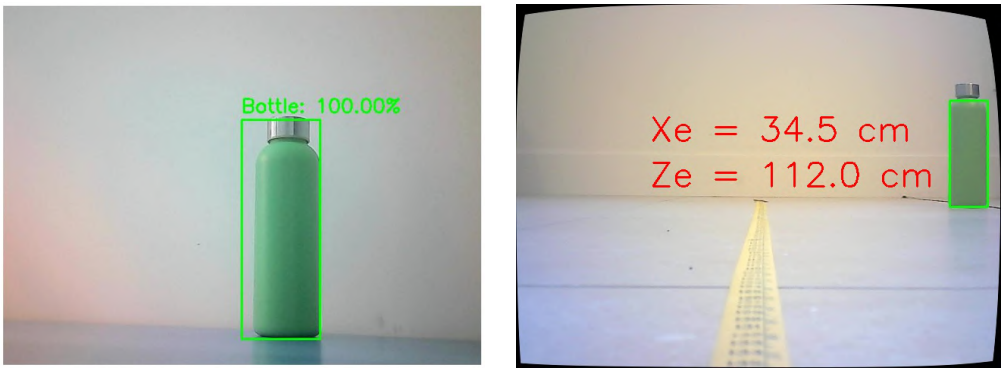
Por último se generan cinco códigos en *bash* para automatizar las peticiones al servidor. El primer código es *service.sh*, que inicia el nodo servidor y realiza una calibración completa de amba ruedas. El segundo código es *avanza05.sh*, que realiza una petición para mover la plataforma hacia delante a una velocidad de 0.5 vueltas por segundo, aproximadamente 0.52 m/s. El tercero es *retrocede05.sh* que hace la misma función que el código anterior pero en sentido contrario. El cuarto es *parar.sh* que detiene ambos motores y mantiene la posición de las ruedas. Por último, *giroH90.sh* realiza un giro de 90^{o} en sentido horario sobre la rueda izquierda.

La principal ventaja de estos códigos es que realizan las peticiones en paralelo usando *forking*. Esto significa que se inician los procesos que envían las peticiones al servidor a la vez en vez de esperar a que acabe el primero para enviar el siguiente. Cada proceso que se ejecuta en paralelo es una petición para mover una rueda por lo que, de este modo, se empiezan a mover las dos ruedas a la vez.

4.4. Seguimento de rutas con coordenadas

Un aspecto importante de las sesiones de terapia asistida por robots es que el paciente pueda seguir una ruta marcada por el terapeuta. Para evitar tener que hacer un control en tiempo real de la dirección del SWalker, se implementa un sistema de seguimiento de ruta con la ayuda del estudiante de Máster en Ingeniería Biomédica, Santiago Ros.

En su Trabajo Fin de Máster [36], Santiago utiliza una *Jetson Nano* y una cámara USB para entrenar un modelo de inteligencia artificial. De esta forma consigue reconocer objetos en el entorno (ver Figura 4.14*(a)*) y calcular su posición en el espacio (ver Figura 4.14*(b)*) midiendo su tamaño.



(a) *(b)*

Figura 4.14: Reconocimiento de imágen *(a)* y obtención de coordenadas *(b)*

Uniendo ambos trabajos se consigue un sistema que puede guiar al SWalker a una coordenada concreta (ver Figura 4.15). Para esto se calcula el ángulo de giro con la Ecuación 4.10 y la distancia a recorrer con la Ecuación 4.11. Así podemos realizar dos peticiones al servidor descrito en la Sección 4.3. La primera petición será un giro de ángulo alfa (α) sobre la rueda izquierda o derecha dependiendo de si la coordenada X_e es positiva o negativa. La segunda petición es avanzar a una velocidad de 0.5 vueltas por segundo durante la distancia P. Para medir la distancia se usa la cuenta de los encoders de las ruedas. Como el encoder tiene 4800 pulsos por vuelta en el sistema actual, se obtiene la cuenta que tendrá el encoder al llegar a las coordenadas con la Ecuación 4.12. Así podemos iniciar el movimiento ejecutando el archivo *avanza05.sh* y esperar en bucle hasta que cambien las coordenadas o se llegue a la cuenta calculada. Si se llega a la cuenta y no ha llegado una nueva coordenada, se ejecuta el archivo *para.sh*, que configura ambas ruedas con el parámetro *idle* a 1 para que se detenga el movimiento pero no se bloqueen las ruedas y el paciente pueda moverse.

$$\alpha = arctan(\frac{X_e}{Z_e})[grados] \tag{4.10}$$

$$P = \sqrt{X_e^2 + Z_e^2}[cm] \tag{4.11}$$

$$C = CPR_{actual} + \frac{P}{\pi D} * 4800 = CPR_{actual} + \frac{\sqrt{X_e^2 + Z_e^2}}{\pi D} * 4800[vueltas] \tag{4.12}$$

Para el intercambio de coordenadas se usa un *topic* que publica periódicamente las coordenadas siguientes de la ruta al nodo Raspberry Pi. Así, el grafo ROS introducido en la Sección 4.3 se convierte en el de la Figura 4.16.

El código completo usado en el proyecto, así como vídeos del funcionamiento del robot están disponibles en la dirección: `https://github.com/Alvarr333/TFG2022_code.git`.

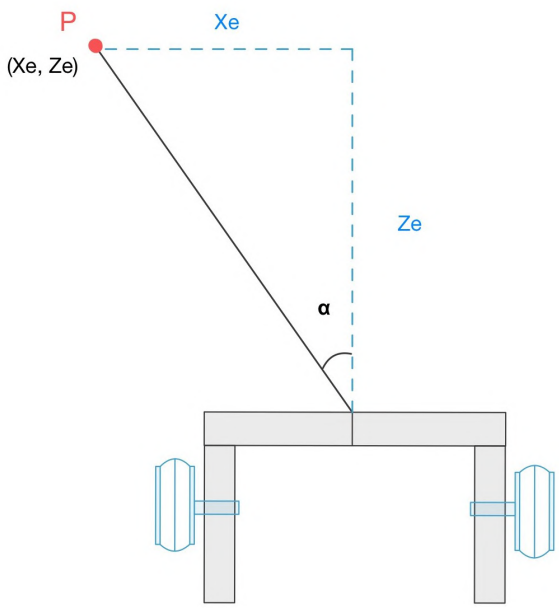


Figura 4.15: Esquema de movimiento del SWalker con coordenadas

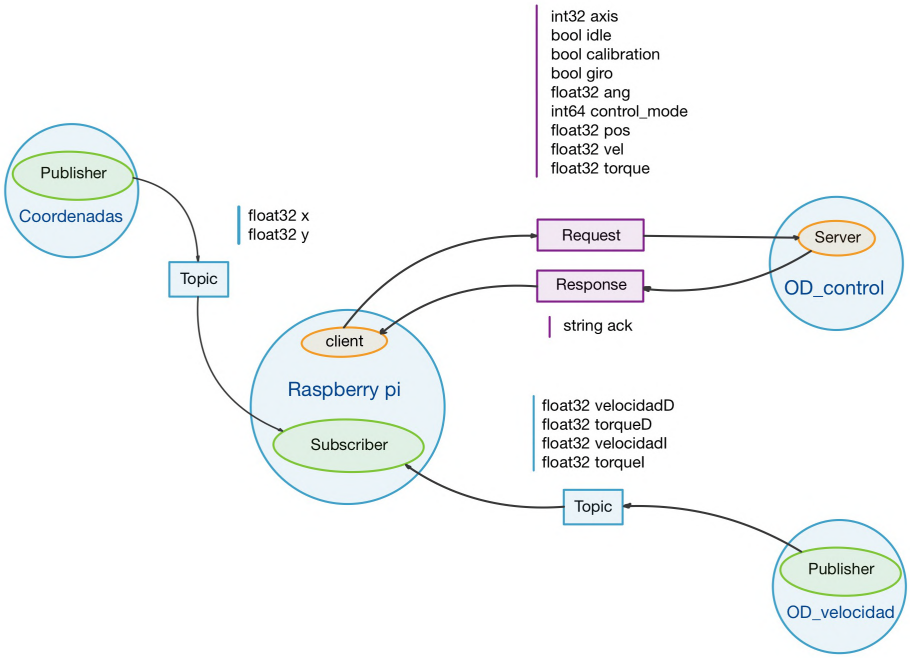


Figura 4.16: Grafo ROS aumentado con el nodo de coordenadas

Capítulo 5

Resultados y validación del sistema

Se realizarán dos tipos de pruebas para comprobar si el sistema es válido: pruebas con carga y sin carga. Las pruebas sin carga se realizan con los motores en el aire y las pruebas con carga se realizan con las rueda en el suelo del laboratorio. Al ser una superficie plana de interior es asemejable al suelo de una sala de rehabilitación.

5.1. Datos de movimiento

La obtención de los datos se realiza mediante *bags*, que son funciones de ROS que recogen los datos enviados en un *topic* y los almacenan en una base de datos SQL. Para la representación de estos datos se usa la herramienta PlotJuggler [37].

Primero hacemos la prueba de mover el sistema a una velocidad de 0.5 vueltas/s sin carga, obteniendo los datos de velocidad de la Figura 5.1. Se observa que la velocidad no es constante, si no que varía entre 0.52 y 0.35 vueltas/s para la rueda derecha y entre -0.61 y -0.25 en la rueda izquierda. El valor medio de las curvas es de aproximadamente 0.45 vueltas/s, similar al de referencia. La rueda izquierda tiene velocidad negativa debido a que las ruedas estan dispuestas en espejo, por lo que deben girar en sentidos contrarios para que la plataforma avance hacia una dirección.

Las rápidas variaciónes de la velocidad al rededor de la velocidad media observada se debe a que el motor está calibrado para funcionar con carga; las ganancias son elevadas y sin resistencia el motor sufre una sobreelongación (*overshoot*) mayor.

En cuanto al par en las ruedas, observando la Figura 5.2, se observa un incremento inicial de par para que la velocidad alcance la referencia, seguido de correcciones que intentan mantener la velocidad para mantenerse constante al final del movimiento. Este valor constante es distinto de cero para mantener las ruedas en el mismo sitio (a velocidad nula) ya que se trata de un control de lazo cerrado.

A continuación se añade al carga con los mismos valores de entrada, obteniendo los datos de velocidad de la Figura 5.3. Se observa variaciones menos significativas respeto al valor medio de velocidad (entre 0.25 y 0.32 vueltas/s en la rueda derecha y entre -0.34 y -0.28 vueltas/s en la rueda izquierda), pero este valor medio es mucho menor que en el caso del sistema sin carga (0.3 vueltas/s frente a 0.45 vueltas/s). Este fenómeno ocurre por el incremento de fricción causado por el peso del sistema completo. Este aumento de peso también genera un aumento del tiempo de subida,

es decir, el tiempo que tarda el motor en alcanzar el valor medio de velocidad por primera vez.

Se puede apreciar el par de los motores con carga en la Figura 5.4. Se identifican valores máximos menores con el motor con carga pero con menores variaciones, lo que concuerda con el análisis de la gráfica de velocidad con carga.

Se ha comprobado que el sistema en presencia de carga es más estable pero también más atenuado. Aún así la reducción de velocidad media no es un problema en el ámbito de la rehabilitación ya que lo importante es el movimiento continuado del paciente.

Por último se recogen los resultados experimentales en la Tabla 5.1.

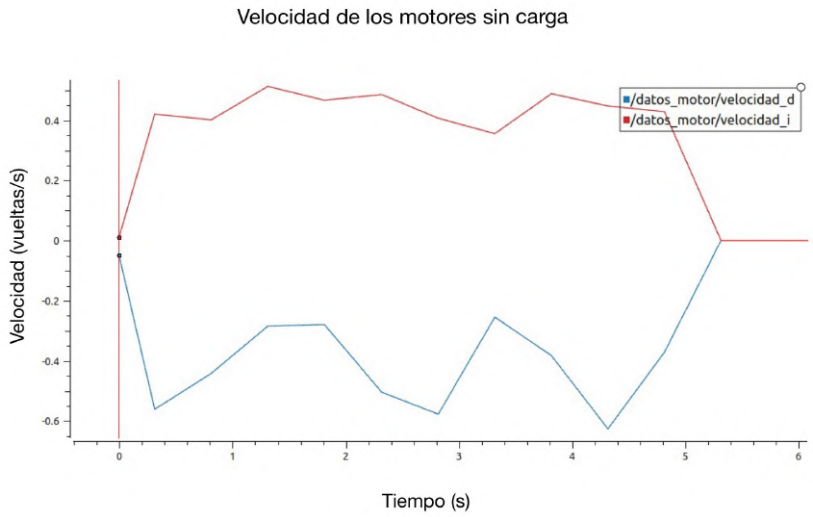


Figura 5.1: Datos de velocidad de la rueda derecha (curva azul) e izquierda (curva roja) con el motor sin carga

Experimento	Velocidad media [vueltas/s]	Par medio [Nm]	Tiempo de subida [s]
Sin carga Izquierda	0.42	0.11	0.28
Sin carga Derecha	0.49	0.21	0.25
Con carga Izquierda	0.28	0.45	1.2
Con carga Derecha	0.3	0.55	1.4

Tabla 5.1: Resumen de los datos obtenidos

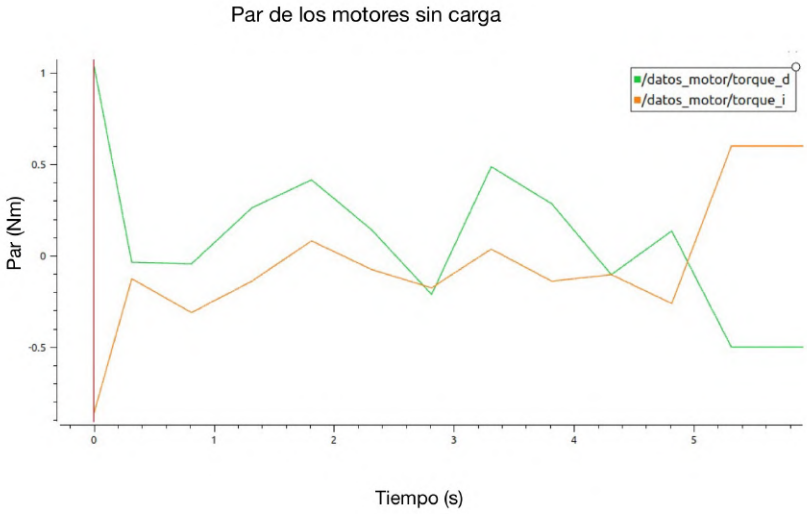


Figura 5.2: Datos de par de la rueda derecha (curva verde) e izquierda (curva naranja) con el motor sin carga

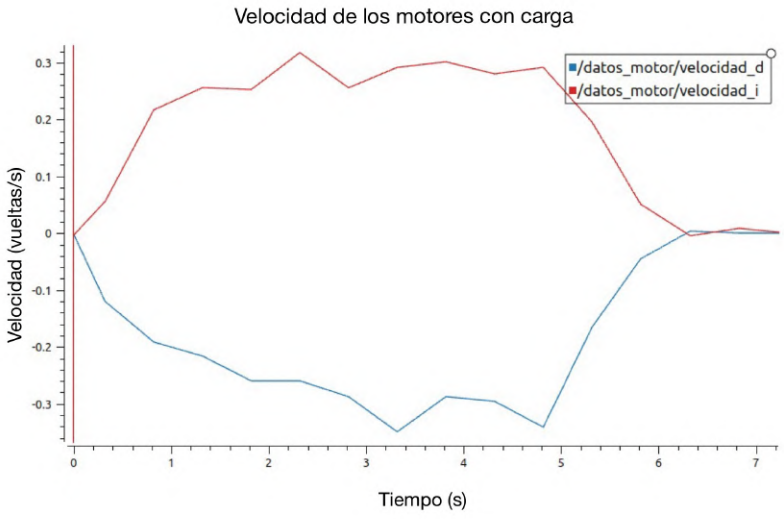


Figura 5.3: Datos de velocidad de la rueda derecha (curva azul) e izquierda (curva roja) con el motor con carga

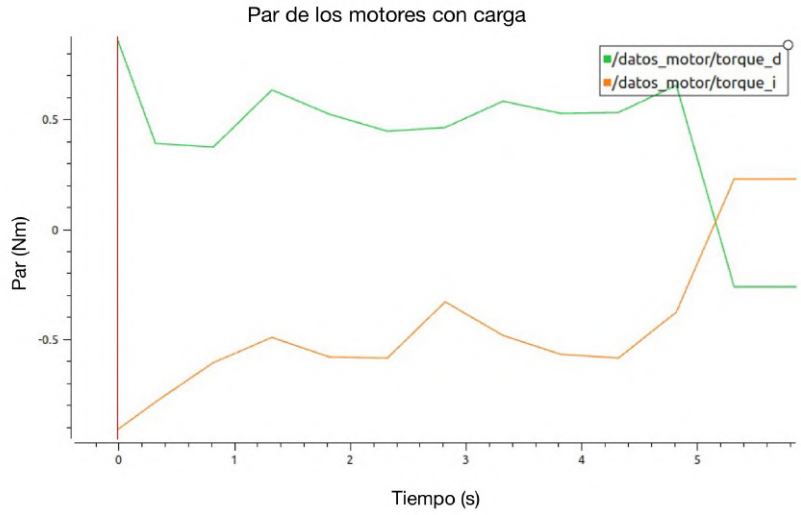


Figura 5.4: Datos de par de la rueda derecha (curva verde) e izquierda (curva naranja) con el motor con carga

Capítulo 6

Conclusiones

6.1. Objetivos cumplidos

El principal objetivo de este trabajo es el diseño, implementación y validación de un sistema de tracción mejorado para la plataforma robótica SWalker de manera durardera y escalable. Se ha implementado este sistema de manera modular y segura, respetando las limitaciones de movimiento de un paciente que ha sufrido fractura de cadera. Tras el análisis realizado en la validación del sistema, se concluye que se han cumplido los objetivos propuestos en la Sección 1.4:

- Estructura: se ha construido un nuevo bastidor con perfiles de aluminio Bosch y una estructura que soporta los motores y encoders de manera fiable y duradera.
- Hardware: se ha imlplementado un sistema de control configurable basado en la tarjeta de control ODrive y se ha calibrado el sistema para ser usado en superficies planas de interior.
- Software: el sistema software implementado es escalable gracias a los nodos ROS, portable, ya que puede ser accedido remotamente por SSH, y sencillo de utilizar gracias a los códigos bash creados.

6.2. Líneas futuras

Este trabajo fin de grado marca el inicio de la remodelación del SWalker, pero deja las siguientes líneas futuras a implementar:

- Implementar el resto de subsistemas del SWalker como nodos en la estructura ROS creada en este trabajo. Estos subsistemas se implementan como servicios (si se trata de actuadores como los pistones o los motores de la cadera) o topics (si se trata de sensores como la galga de detección de peso).
- Introducir una interfaz gráfica para hacer más sencillo el control del sistema y que pueda ser usado por personal de rehabilitación.
- Probar el sistema completo en una clínica de rehabilitación con pacientes reales.

Bibliografía

[1] R. Bachiller Caño, Y. Soler de Paz, P. A. Jiménez Méndez, and D. Díaz Hernández. Fractura de cadera en ancianos. *European Journal of Health Research*, (6(1)):5–15, 2020.

[2] Antonio Herrera, Angel Antonio Martínez, Luis Ferrandez, Enrique Gil, and Alonso Moreno. Epidemiology of osteoporotic hip fractures in spain. *International orthopaedics*, 30(1):11–14, 2006.

[3] Marcela Villalobos Ulate. Factores que influyen en la recuperación funcional del paciente adulto mayor con fractura de cadera. *Ciencia & Salud*, 4(5), 2020.

[4] Catherine Sherrington, Nicola J Fairhall, Geraldine K Wallbank, Anne Tiedemann, Zoe A Michaleff, Kirsten Howard, Lindy Clemson, Sally Hopewell, and Sarah E Lamb. Exercise for preventing falls in older people living in the community. *Cochrane database of systematic reviews*, 1(1):CD012424–CD012424, 2019.

[5] J Halbert, M Crotty, C Whitehead, I Cameron, S Kurrle, S Graham, H Handoll, T Finnegan, T Jones, A Foley, and M Shanahan. Multi-disciplinary rehabilitation after hip fracture is associated with improved outcome: A systematic review. *Acta dermato-venereologica*, 39(7):507–512, 2007.

[6] Ekso Bionics. Ekso nr. `https://eksobionics.com/eksonr/`. [Online, Accedido por última vez: 29/05/2022].

[7] Reha-Stim Medtec. Gait trainer gt ii. `https://reha-stim.com/gt-ii/`. [Online, Accedido por última vez: 29/05/2022].

[8] Hocoma. Locomat. `https://www.hocoma.com/solutions/lokomat/`. [Online, Accedido por última vez: 29/05/2022].

[9] V. Costa, O. Ramírez, J.S. Lora-Millan, E. Urendes, E. Rocon, L. Perea, and R. Raya. Design of a robotic platform for hip fracture rehabilitation in elderly people. In *2020 8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)*, pages 599–604, 2020.

[10] Kelvin Gear. Motores kelvin k80. `https://www.elmeq.es/ftp/productsFiles/368/KELVIN-K80-K90-ES.pdf`. [Online, accedido por última vez: 8/6/2022].

[11] Infineon. Encoder heds 5540. `https://docs.rs-online.com/b3e7/0900766b8130f67e.pdf`. [Online, accedido por última vez: 8/6/2022].

[12] PCM-UK. Galga sb-ft1. https://pcm-uk.com/wp-content/uploads/2020/08/SB-FT1-Data-Sheet.pdf. [Online, accedido por última vez: 8/6/2022].

[13] Bansbach easylift. Actuador lineal neumático. https://www.bansbach.com/index.php/en/products/easye-line/easye. [Online, accedido por última vez: 8/6/2022].

[14] Itectec. Esquema de funcionamiento de un motor con escobillas. https://itectec.com/electrical/electrical-self-inductance-in-steady-state-equivalent-model-of-\brushed-dc-machine/. [Online, accedido por última vez: 9/6/2022].

[15] Rotor magazin. Ejemplo de motor sin escobillas. https://www.rotor-magazin.com/basiswissen-aufbau-und-funktion-von-brushless-motoren/. [Online, accedido por última vez: 9/6/2022].

[16] Hobby King. 9235-100kv turnigy multistar brushless multi-rotor motor. https://hobbyking.com/en_us/9235-100kv-turnigy-multistar-brushless-multi-rotor-motor.html?___store=en_us. [Online, accedido por última vez: 9/6/2022].

[17] ODrive. Dual shaft motor - d6374 150kv. https://odriverobotics.com/shop/odrive-custom-motor-d6374-150kv. [Online, accedido por última vez: 9/6/2022].

[18] eBay. Motor hoverboard sin escobillas. https://www.ebay.com/itm/353548829494?hash=item5251270336:g:YS8AAOSwwylg1n6s. [Online, accedido por última vez: 9/6/2022].

[19] ODrive Robotics. Comparación de motores sin escobillas. https://docs.google.com/spreadsheets/d/12vzz7XVEK6YNIOqH0jAz51F5VUpc-lJEs3mmkWP1H4Y/edit#gid=0. [Online, accedido por última vez: 9/6/2022].

[20] AliExpress. Encoder 600 ppr incremental de dos fases. https://id.aliexpress.com/item/32756209569.html?gatewayAdapt=glo2idn. [Online, accedido por última vez: 12/6/2022].

[21] Encoder - ¿qué es? 4 tipos de encoder y su funcionamiento., -03-07T10:10:52+00:00 2020.

[22] Carmine Noviello. *Mastering STM32 : a step-by-step guide to the most complete ARM Cortex-M platform, using a free and powerful development environment based on Eclipse and GCC*. C. Noviello, S.l.], 2018.

[23] ODrive. Odrive v3.6. https://odriverobotics.com/shop/odrive-v36. [Online, accedido por última vez: 9/6/2022].

[24] MJbots. Moteus r4.11 controller. https://mjbots.com/products/moteus-r4-11. [Online, accedido por última vez: 9/6/2022].

[25] MIT. Cheetah mini. https://es.aliexpress.com/item/1005001809102045.html. [Online, accedido por última vez: 9/6/2022].

[26] Open Robotics. Robotic operating system. https://www.ros.org/. [Online, accedido por última vez: 12/6/2022].

[27] ROS. Ros topics. https://docs.ros.org/en/foxy/Tutorials/Topics/Understanding-ROS2-Topics.html. [Online, accedido por última vez: 12/6/2022].

[28] ROS. Ros services. https://docs.ros.org/en/foxy/Tutorials/Services/Understanding-ROS2-Services.html. [Online, accedido por última vez: 12/6/2022].

[29] ROS. Ros nodes. https://docs.ros.org/en/foxy/Tutorials/Understanding-ROS2-Nodes.html. [Online, accedido por última vez: 12/6/2022].

[30] Termux. Emulador de terminal linux para móviles. https://termux.com/. [Online, accedido por última vez: 12/6/2022].

[31] Bosch. Perfil bosch 40x40 mm. https://perfilesbosch.com.mx/app/producto/perfil-40x40/. [Online, accedido por última vez: 12/6/2022].

[32] Leroy Merlin. Batería solar agm u-power 12v 7ah. https://www.leroymerlin.es/fp/82461739/bateria-solar-agm-u-power-12v-7. [Online, accedido por última vez: 12/6/2022].

[33] Olfer. Sd-15b-5 buck converter. https://www.olfer.com/sd-15b-5.html?gclid=Cj0KCQjwhqaVBhCxARIsAHK1tiNaUJzh-p8eM81qcYagnPPedfRvgH3olQl\anbeJqZdcXmmpqUVCLLEaAqA9EALw_wcB. [Online, accedido por última vez: 12/6/2022].

[34] Oskar Weigl. Drill test of a hoverboard motor. https://discourse.odriverobotics.com/t/project-hoverarm/441/2?u=madcowswe. [Online, accedido por última vez: 12/6/2022].

[35] ODrive. Lazo de control del controlador odrive. https://docs.odriverobotics.com/v/latest/control.html. [Online, accedido por última vez: 12/6/2022].

[36] Santiago Ros. Development of a computer vision based robotic guidance system for the neurorehabilitation of brain injury patients. 2022.

[37] Davide Faconti. Programa de representación de datos plotjuggler. https://www.plotjuggler.io/. [Online, accedido por última vez: 13/6/2022].

Apéndice A

Aspectos éticos, económicos, sociales y ambientales

A.1. Impacto social y económico

Como se mencionó en la Sección 1.1, la fractura de cadera es una lesión que conlleva un alto coste social y económico. Estas fracturas afectan a la capacidad de desplazamiento de la persona, lo que puede ramificar en problemas en muchas otras áreas de la vida diaria.

Dentro del ámbito de la mejora en la rehabilitación que suponen las terapias asistidas por robots, este trabajo ha contribuido modernizando un exoesqueleto para facilitar su uso y escalabilidad. De esta forma se ayuda a la mejora de la calidad de vida de pacientes que sufren esta lesión acelerando la recuperación tras la fractura. También se mejora la calidad de trabajo de terapeutas que tienen que realizar un esfuerzo adicional para asistir a los pacientes cuando realizan movimientos de rehabilitación. Con plataformas como el SWalker, los terapeutas se pueden centrar en dar instrucciones y guiar las sesiones de rehabilitación, sin preocuparse del esfuerzo que supone mover al paciente de forma segura. De este modo se reduce el tiempo y esfuerzo dedicado a cada sesión de terapia.

A.2. Impacto ético

La introducción progresiva de robots en el ámbito biomédico trae consigo inquietudes respecto a la ética de introducir máquinas en un ámbito tan delicado como la medicina y el tratamiento de pacientes. Estas dudas se basan en la desconfianza, normalmente por parte del paciente, hacia el robot por razones de seguridad y eficacia. Como se ha explicado en la Sección 1.1, la eficacia de estas terapias está demostrada. Este Trabajo Fin de Grado pretende ayudar a la seguridad y fiabilidad del sistema haciéndolo más robusto. Otro aspecto ético importante es el reemplazo de los trabajadores por robots que se ha visto en los últimos años. En este aspecto, el SWalker pretende trabajar con los terapeutas, en vez de reemplazarlos, sirviendo a modo de apoyo y asistiendo al profesional.

Apéndice B

Presupuesto económico

Este trabajo se ha realizado sobre la plataforma SWalker, propiedad del CAR-CSIC y con materiales proporcionados por la Universidad Politécnica de Madrid y la Escuela Técnica Superior de Ingenieros de Telecomunicación.

- **Personal**: Para determinar el coste horario se usa el dato del sueldo medio anual de un ingeniero de telecomunicaciones *junior* del COIT (Colegio Oficial de Ingenieros de Telecomunicación): 25000€. Teniendo en cuenta los gastos de IRPF que supone a la empresa (33 %) y el número medio de horas laborables al año (1800) se obtiene el resultado de la Tabla B.1). Se fija tambien un tiempo aproximado de dedicación de 400 horas.

	Coste horario (€)	**Horas**	**Total (€)**
Estudiante de ingeniería	14	400	6500

Tabla B.1: Costes de personal.

En cuanto a los recursos materiales, se calcula el precio teniendo en cuenta la amortización en €/mes, calculada como el precio del producto dividido entre los meses de vida útil según lo indica la Agencia Tributaria (ver Tabla B.2).

	Tiempo de vida (años)	Uds.	Coste (€)	Amortización (€/mes)	Uso (meses)	Total (€)
Taladro	18	1	140	0.65	2	1.3
Sierra	18	1	10	0.05	2	0.1
Perfil Bosch	8	2	35	0.36	2	1.44
Motor hoverboard	8	2	40	0.42	4	3.34
Encoder 600ppr	5	2	23	0.38	4	3.06
Tarjeta ODrive	10	1	220	1.83	4	7.3
Raspberry Pi 4	10	1	31	0.26	3	0.78
Cable eléctrico	8	1	5	0.05	1	0.05
Resistencia 50W	8	1	5	0.05	4	0.2
Tornillos	8	1	8	0.08	1	0.08
Madera	8	1	20	0.21	1	0.21
Baterías de 12V	8	3	14	0.15	1	0.45
Buck Converter	5	1	13	0.22	1	0.22
Fuente de 30V	8	1	60	0.63	4	2.5
MATLAB	1	1	800	66.67	1	66.67
TOTAL						**87.7**

Tabla B.2: Costes de recursos materiales.

Juntando todos los costes anteriores y teniendo en cuenta el 21 % de IVA se obtienen los costes totales de la Tabla B.3.

	Coste
Costes de personal	6500€
Costes de material	87.7€
Subtotal	6587.7€
IVA	1383.42€
Total	7971.12€

Tabla B.3: Costes totales.