



GRADO EN INGENIERÍA DE ROBÓTICA SOFTWARE

Escuela de Ingeniería de Fuenlabrada

Curso académico 2024-2025

Trabajo Fin de Grado

Diseño de controladores gamificados
para rotot de rehabilitación

Tutor: Julio Salvador Lora Millán

Cotutor: Juan Alejandro Castaño Peña

Autor: Sofía Perales Díez



Este trabajo se distribuye bajo los términos de la licencia internacional CC BY-SA International License (Creative Commons Attribution-ShareAlike 4.0). Usted es libre de *(a) compartir*: copiar y redistribuir el material en cualquier medio o formato para cualquier propósito, incluso comercialmente; y *(b) adaptar*: remezclar, transformar y construir a partir del material para cualquier propósito, incluso comercialmente. La licenciente no puede revocar estas libertades en tanto usted siga los términos de la licencia:

- *Atribución.* Usted debe dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo de la licenciente.
- *Compartir igual.* Si remezcla, transforma o crea a partir del material, debe distribuir su contribución bajo la misma licencia del original.

No hay restricciones adicionales — No puede aplicar términos legales ni medidas tecnológicas que restrinjan legalmente a otras a hacer cualquier uso permitido por la licencia.

Agradecimientos

En primer lugar, me gustaría agradecer a la Universidad Rey Juan Carlos por brindarme la oportunidad de formar parte de esta institución. A todo el equipo docente que me ha formado académicamente, y en especial, a mis tutores Julio Salvador Lora y Juan Alejandro Castaño por la confianza y el apoyo depositado en mí para llevar a cabo este proyecto.

Agradecer a mis padres por todo lo que me han dado y por estar siempre a mi lado apoyándome e impulsándome a superarme y convertirme en la mejor versión de mí misma. A mi hermana, de la que aprendo todos los días y que me motiva a ser alguien en quien pueda inspirarse y confiar. Y a mi abuela, que es un referente para mí y me ha defendido siempre.

También agradecer a mis amigos, aquellos que se alegran por mis logros como si fuesen los suyos propios. Por la paciencia que tienen conmigo y por estar en las buenas y en las malas. A Nuria de la Fuente, por ser mi alma gemela desde el primer momento que nos conocimos. A Lucía Álvarez, por convertirse en mi hermana, doy gracias por habernos encontrado en la otra punta del mundo. A mis amigas del cole, quienes me han visto crecer y con las que he compartido la mayor parte de mi vida. Y a mis amigos de la universidad, gracias por las risas, los desayunos en la cafetería y sobre todo por el apoyo en momentos de estrés. Porque los ratitos con vosotros han hecho que el camino sea más fácil y ameno.

Y por último, gracias a Hispamast y todos sus empleados por darme la oportunidad de hacer las prácticas de empresa y formarme laboralmente. En especial a mis jefes y amigos Javier Trabalón y Alejandro Caballero, por cuidarme y tomarme como vuestra aprendiz. Gracias a vosotros he disfrutado de cada minuto durante las prácticas y ojalá sigamos compartiendo momentos juntos.

Todos vosotros me habeis hecho ser quien soy, gracias desde el corazón.

*A mi yo de pequeña;
quien soñaba con ser ingeniera, enhorabuena lo hemos conseguido*

Madrid, 6 de junio de 2025

Sofía Perales Díez

Resumen

El ictus es una de las principales causas de discapacidad motora, afectando a la movilidad y calidad de vida de los pacientes. La rehabilitación temprana, durante los primeros meses tras el ictus, es crucial para optimizar la recuperación funcional.

En los últimos años, diversos estudios han evaluado la efectividad de la integración de dispositivos robóticos en los procesos de rehabilitación, mostrando resultados prometedores en cuanto a la mejora de la movilidad y participación del paciente.

El presente trabajo tiene como objetivo desarrollar un sistema de rehabilitación motora para la extremidad superior de pacientes post-ictus, basado en un entorno gamificado personalizado que potencie la motivación del paciente y mejore la efectividad del tratamiento. La plataforma consta de tres componentes principales: una interfaz de control que permite al terapeuta adaptar las sesiones de rehabilitación de acuerdo con las capacidades del paciente, una interfaz de visualización que facilita el monitoreo en tiempo real del progreso del paciente, y una interfaz de juego que permite hacer los ejercicios de rehabilitación de manera interactiva y dinámica. Además, el sistema almacena los datos más relevantes durante la terapia, lo que facilita realizar un análisis detallado y evaluar el progreso de cada paciente.

Se espera que este sistema no solo facilite la rehabilitación motora, sino que también mejore la motivación y adherencia al tratamiento, ofreciendo una solución más efectiva para la recuperación post-ictus, gracias a su adaptación personalizada a las necesidades de cada paciente, un mayor control sobre los ejercicios y la escalabilidad que permite ajustarse a diferentes modos de terapia.

Abstract

Stroke is one of the main causes of motor disability, affecting patients' mobility and quality of life. Early rehabilitation, during the first months after stroke, is crucial to optimize functional recovery.

In recent years, several studies have evaluated the effectiveness of integrating robotic devices in rehabilitation processes, showing promising results in terms of improved patient mobility and participation.

The present work aims to develop a motor rehabilitation system for the upper extremity of post-stroke patients, based on a personalized gamified environment that enhances patient motivation and improves treatment effectiveness. The platform consists of three main components: a control interface that allows the therapist to adapt the rehabilitation sessions according to the patient's capabilities, a visualization interface that facilitates real-time monitoring of the patient's progress, and a game interface that allows the rehabilitation exercises to be performed interactively and dynamically. In addition, the system stores the most relevant data during therapy, which facilitates detailed analysis and evaluation of each patient's progress.

It is expected that this system will not only facilitate motor rehabilitation, but also improve motivation and adherence to treatment, offering a more effective solution for post-stroke recovery, thanks to its personalized adaptation to the needs of each patient, greater control over the exercises and scalability that allows it to adjust to different modes of therapy.

Acrónimos

API *Interfaz de Programación de Aplicaciones*

ECV *Enfermedad Cerebro-vascular*

EPIK *Estimulación para Promover la Independencia mediante Kinect*

GUI *Interfaz Gráfica de Usuario*

MIDAS *Multi-sensorial Immersive Dynamic Autonomous System*

OMS *Organización Mundial de la Salud*

ROS *Sistema Operativo Robótico*

RV *Realidad Virtual*

SAFE *Alianza Europea contra el Ictus*

SEN *Sociedad Española de Neurología*

SERMEF *Sociedad Española de Rehabilitación y Medicina Física*

SNS *Sistema Nacional de Salud*

Índice general

1. Introducción	1
2. Objetivos	9
2.1. Descripción del problema	9
2.1.1. Objetivo general	9
2.1.2. Objetivos específicos	10
2.2. Requisitos	10
2.3. Competencias	11
2.4. Metodología	12
2.5. Plan de trabajo	12
3. Plataforma de desarrollo	14
3.1. Sistema operativo	14
3.2. Entorno de desarrollo	14
3.2.1. Python	14
3.2.2. ROS 2	15
3.3. Componentes físicos	16
4. Diseño	18
4.1. Esquema de nodos y topics	19
4.2. Interfaz de registro de un paciente	21
4.3. Interfaz de control	22
4.4. Interfaz de visualización	25
4.5. Juego flappy	27
5. Conclusiones	32
Bibliografía	34

Índice de figuras

1.1. Prevalencia registrada de enfermedad cerebro-vascular según sexo y edad en España, 2019	2
1.2. Incidencia del ictus en España, 2020	2
1.3. Terapia asistida por el robot MIT-MANUS	4
1.4. Prototipo comercial del robot rehabilitador Gentle/s	5
1.5. Sistema de rehabilitación EPIK	7
1.6. Videojuego Peter Jumper para la rehabilitación de manos y muñecas	7
3.1. Actuador T-Series	16
4.1. Esquema de nodos y topics	20
4.2. Interfaz de registro de un paciente	22
4.3. Interfaz de configuración de los parámetros terapéuticos	24
4.4. Interfaz de configuración de los límites y orientación del brazo robótico	25
4.5. Interfaz de visualización del terapeuta	27
4.6. Nivel 2 del juego sin perturbación	30
4.7. Nivel 1 del juego con perturbación	31

Listado de códigos

4.1. Encabezado del fichero de configuración	23
4.2. Cargar un sonido al juego	28
4.3. Movimiento vertical del jugador	29

Listado de ecuaciones

4.1. Cálculo del error de trayectoria en el tiempo	28
4.2. Cálculo del error de trayectoria por posición	28
4.3. Cálculo del nivel de asistencia según el nivel de dificultad	29

Capítulo 1

Introducción

El ictus es una enfermedad cerebro-vascular (ECV). El término ictus en latín significa golpe. Es un trastorno brusco de la circulación cerebral que altera la función de una determinada región del cerebro.

Según la Organización Mundial [de la Salud, 2022] (OMS) es la primera causa de discapacidad y la segunda de muerte en el mundo. Aproximadamente 15 millones de personas sufren un ictus cada año; entre ellas, 5 millones mueren y otros 5 millones quedan con alguna discapacidad permanente, lo que supone una carga significativa para las familias y comunidades, como indica la [OMS, 2025].

Las causas o factores de riesgo según la Federación Española [del Ictus, 2024] son las siguientes:

1. Sufrir un ictus recientemente.
2. Presión sanguínea elevada.
3. Fumar.
4. Padecer diabetes.
5. Sufrir una enfermedad cardíaca.
6. Estación del año y clima.
7. Contador de glóbulos rojos alto.
8. Consumo excesivo de alcohol y drogas.

Según el Sistema Nacional de Salud, [SNS, 2024], cada año, alrededor de 120,000 personas sufren un ictus en España. La edad es uno de los factores de riesgo principales de esta enfermedad, aunque ocurre en todos los grupos de edad. En los últimos años la

incidencia en adultos jóvenes ha aumentado un 40 %. Se estima que 1 de cada 4 personas en el mundo sufrirán un ictus a lo largo de su vida. Gracias a los avances científicos, tecnológicos y clínicos se han podido desarrollar tratamientos para minimizar los déficit que produce. En la Imagen 1.1, se muestra la prevalencia registrada de enfermedad cerebro-vascular por 1.000 habitantes, según sexo y edad en España en 2019. Y, en la Imagen 1.2, se pueden observar datos sobre la incidencia del ictus en España en el año 2020.

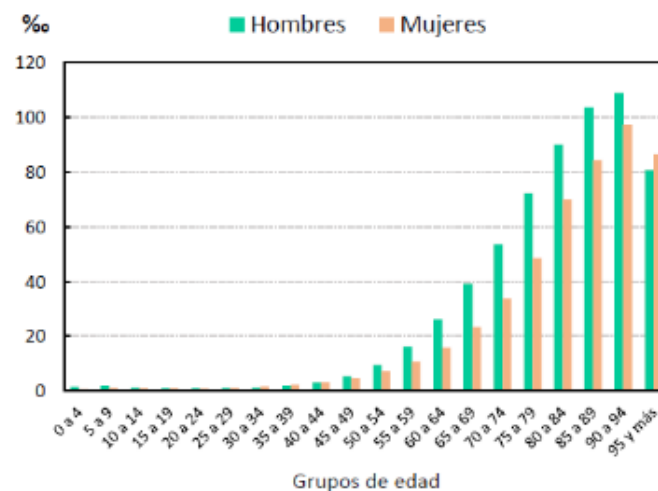


Figura 1.1: Fuente: Figura extraída del Informe Anual de Salud del Sistema Nacional de Salud 2020-2021. Fuente de datos: Ministerio de Sanidad. Base de Datos Clínicos de Atención Primaria (BDCAP)

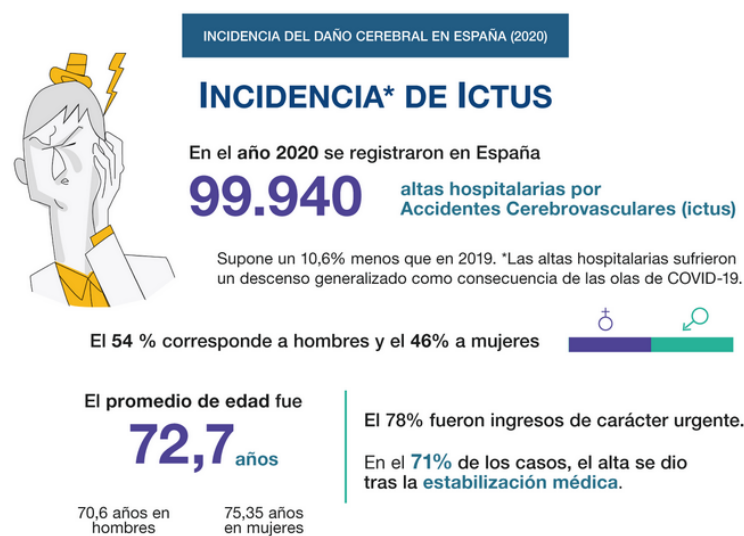


Figura 1.2: Fuente de datos: Encuesta de Movilidad Hospitalaria (INE, 2020). Ministerio de España. FEDACE. Observatorio Estatal Daño Cerebral

Según la OMS, la rehabilitación se define como un "conjunto de medidas que ayudan a las personas que tienen o probablemente tendrán una discapacidad, a conseguir y mantener el funcionamiento óptimo en interacción con su ambiente".

Asimismo el Grupo de Estudio de ECV de la Sociedad Española de Neurología, [SEN, 2007], considera que la rehabilitación es una de las partes más importantes para el tratamiento de la mayoría de los pacientes que han sufrido un ictus y han sobrevivido. Suele comenzarse en los primeros días de estancia en el hospital y se recomienda a cualquier paciente que, previo al ictus, fuese autosuficiente. Es importante entender que es difícil volver a una situación igual a la inicial. El objetivo fundamental es ayudar al paciente a adaptarse, ya que el ictus se recupera de forma espontánea o puede no recuperarse nunca, dependiendo de su gravedad, la cual determina la duración del programa de rehabilitación y su intensidad. Los seis primeros meses son los de mayor importancia para recuperar los movimientos voluntarios. Con un programa adecuado, 1/3 de los pacientes vuelven a su trabajo.

El plan de acción europeo para el ictus 2018-2030, propuesto por la Alianza Europea contra el ictus [SAFE, 2030], surge con el objetivo de mejorar el cuidado de los ictus y la vida después de estos. Entre los objetivos generales para el año 2030 se encuentran:

1. Garantizar que al menos el 90 % de la población tenga acceso a la rehabilitación precoz en la unidad de ictus.
2. Proporcionar el alta precoz con apoyo a por lo menos el 20 % de los pacientes de ictus en todos los países.
3. Ofrecer programas de acondicionamiento físico a todos los pacientes de ictus que viven en la comunidad.
4. Proporcionar un plan documentado para la rehabilitación en la comunidad y el apoyo al autocontrol de todos los pacientes de ictus con dificultades residuales al recibir el alta hospitalaria.
5. Garanticen que se revisan las necesidades de rehabilitación, y otras, de todos los pacientes de ictus entre los 3 y 6 meses después de sufrirlo y anualmente a partir de ese momento.

La Sociedad Española [de Rehabilitación y Medicina Física, 2024] (SERMEF), redacta que el 50 % de los supervivientes de ictus sufre secuelas y la rehabilitación

temprana reducirá su dependencia en un 20 %. Además, manifiesta que existen desigualdades en los accesos a estos tratamientos entre Comunidades Autónomas. Entre las principales secuelas se encuentra la pérdida de función motora, que afecta entre el 50 % y 85 % de los pacientes, trastornos del habla, disfunciones cognitivas, eplasticidad y debilidad muscular.

Una vez conocida la prevalencia del ictus y la importancia de la rehabilitación, se valoran las ventajas de introducir la robótica en este campo.

En el artículo de [Newport, 2007], se analizaban los beneficios de la rehabilitación asistida por robots frente a la fisioterapia tradicional. Esta consiste en la realización de ejercicios manuales por parte de un fisioterapeuta cualificado, aunque son muy exigentes en cuanto al tiempo de dedicación al paciente. El uso de terapias mediadas por robots podría mejorar sustancialmente la recuperación de la función tras un ictus y, a su vez, reducir las exigencias impuestas a terapeutas y hospitales. Por ello se comenzó una investigación para utilizar dispositivos mecánicos en el proceso de rehabilitación.

Varios autores han propuesto el uso de dispositivos robóticos para la rehabilitación del miembro superior. Existen diversos modelos como el robot MIT-MANUS o Gentle/s.

El primero, fue desarrollado por los investigadores [Krebs et al., 2004], Volpe y Hogan, del Massachusetts Institute of Technology y del Burke Medical Research Institute. Se ha demostrado que reduce de manera eficaz el tiempo de recuperación motriz del paciente al realizar los ejercicios apropiados para la rehabilitación del hombro y el codo. En la Imagen 1.3, se puede observar un paciente hospitalizado por ictus durante la terapia en el Hospital de Rehabilitación Burke (White Plains, NY), llevada a cabo con una versión comercial de MIT-MANUS.

El segundo, es un robot de asistencia robótica en rehabilitación neurológica y motora. Como se indica en el artículo [Coote et al., 2008], la tasa de recuperación fue mayor durante la fase de terapia mediada por el robot que en la fase base para la mayoría de los sujetos. En la Imagen 1.4, se muestra a un usuario utilizando el prototipo comercial Gentle/s.

Según [Volpe et al., 2000], la rehabilitación asistida por robot ofrece ventajas importantes respecto a la fuerza muscular, el aumento de las puntuaciones clínicas y un grado mayor de recuperación de la independencia funcional.



Figura 1.3: Fuente: Robótica de rehabilitación: prueba piloto de una extensión espacial para MIT-Manus



Figura 1.4: Fuente: Retos y oportunidades de la neurorrehabilitación asistida por robots. ResearchGate

En los últimos 5 años, se han realizado distintos análisis sobre la efectividad de la incorporación robótica en la rehabilitación.

- En el artículo [Lee et al., 2023] se realiza un metanálisis para medir el rendimiento de las extremidades superiores de 18 estudios. Concluyó que la rehabilitación con brazo robótico y una duración de entre 30 y 60 minutos por sesión, mejoraron significativamente la función de las extremidades superiores.
- En el artículo [Hernandez-Echarren and Sanchez-Cabeza, 2023] se indica que la terapia con robots es una terapia relativamente nueva, permite aumentar el número de repeticiones en la ejecución de movimientos específicos y concluyen que presenta beneficios en todas las fases de rehabilitación del ictus.
- En el artículo [Sue et al., 2024] se realiza una búsqueda para analizar el efecto de los robots de rehabilitación seleccionando 18 ensayos con 573 pacientes con

ictus y los resultados mostraron que los pacientes que reciben este tipo de terapia mejoraron significativamente las puntuaciones de evaluación motora. Concluyeron que el entrenamiento asistido por robot es superior al entrenamiento convencional, lo que respalda el uso de robots en la práctica clínica.

Las técnicas de gamificación no son solamente lúdicas, sino que deben conseguir mejorar el aprendizaje. Para ello, se incluyen diversos elementos como son: el objetivo del juego, distintos tipos de obstáculos y retos para incentivar un mayor esfuerzo por parte del paciente, promoviendo su superación, y recompensas para generar un refuerzo positivo en su progreso y fomentar su motivación. La gamificación mejora la salud de los pacientes mejorando los ámbitos de vida, fomentando el cumplimiento del tratamiento, facilitando el diagnóstico de la enfermedad, mejorando la salud psicológica y ralentizando el deterioro cognitivo, como se explica en [AlmirallMed, 2021].

Para que la rehabilitación del paciente no resulte monótona y desmotivadora, es interesante incluir entornos gamificados en las terapias como refleja [Mubin et al., 2022]. Con ello se consigue un mejor compromiso y adherencia, un incremento de la motivación del paciente y permite la personalización del ejercicio.

Como explica [Sánchez-Gil et al., 2025] los dispositivos gamificados se pueden clasificar en cuatro tipos: robótica monitorizada, no monitorizada, realidad virtual y estimulación eléctrica neuromuscular. La conclusión del estudio recoge que los pacientes tuvieron un impacto positivo significativo en los niveles emocional, social y personal, además de una mejora en su función motora y cognitiva.

Aplicado a la práctica, se realizó un estudio por [Rubino et al., 2024] que consistió en la rehabilitación del brazo gamificada para personas con ictus. El juego se personaliza según la capacidad de cada paciente y aumenta su dificultad en función del rendimiento del individuo. El objetivo del estudio fue evaluar si 10 días de entrenamiento gamificado mejora la velocidad de la recuperación motora del brazo. Los datos del estudio sugieren que las intervenciones de rehabilitación motora gamificadas pueden beneficiar el comportamiento motor después de un accidente cerebro-vascular.

Algunos juegos y plataformas utilizadas en la actualidad para rehabilitación motora post-ictus son los siguientes:

- Sistema de Estimulación para Promover la Independencia mediante Kinect

(EPIK), desarrollado por investigadores del Grupo de Telemática e Imagen (GTI) de la Universidad de Valladolid (UVa). Basado en el sensor Kinect con el objetivo de realizar ejercicios de equilibrio. La noticia fue publicada por [Casadomo, 2015]. En la Imagen 1.5 se muestra el funcionamiento del sistema de rehabilitación EPIK como un videojuego 3D basado en la imitación de siluetas.



Figura 1.5: Fuente: [Casadomo, 2015]

- Peter Jumper y Andrómeda son dos videojuegos desarrollados por la Universidad Carlos III en colaboración con a Escuela Politécnica del Ecuador y los hospitales ASEPEYO en Barcelona y Madrid. Se trata de un sistema de rehabilitación para personas con problemas de movilidad en mano y muñeca. Permite analizar la evolución del paciente mediante los datos biométricos generados. Publicado por [SER, 2025]. En la Imagen 1.6 se observa a un usuario con el controlador manejando el videojuego *Peter Jumper*.



Figura 1.6: Fuente: [SER, 2025]

- Proyecto Multi-sensorial Immersive Dynamic Autonomous System (MIDAS), compuesto por un exoesqueleto de mano, un subsistema de realidad virtual (RV) y un subsistema olfativo. En su conjunto involucra cuatro de los cinco sentidos durante la rehabilitación. Puede encontrarse publicado en [arXiv, 2022].

Una vez realizada una revisión sobre el ictus, los métodos de rehabilitación, la gamificación y la importancia de su aplicación en la recuperación del paciente, se plantea la necesidad de crear un proyecto para la rehabilitación motora de la extremidad superior mediante una plataforma robótica gamificada. Este proyecto se enfoca en el diseño de la arquitectura software, compuesta por interfaces gráficas para el control y configuración de los parámetros terapéuticos y visualización del rendimiento del paciente. Asimismo, se desarrolla un videojuego para fomentar la motivación y atención del paciente, que puede personalizar el terapeuta según las necesidades de cada usuario.

Capítulo 2

Objetivos

Un objetivo bien definido es el punto de partida de todo logro.

W. Clement Stone, *Success Through a Positive Mental Attitude*, 1959

Después de haber establecido el marco contextual de este proyecto, continuo con la descripción del problema, los requisitos, las competencias, la metodología y el plan de trabajo empleado para su desempeño.

2.1. Descripción del problema

La recuperación motora tras un ictus requiere de terapias personalizadas y motivadoras que promuevan la participación activa del paciente. Tal como señalan [Clark et al., 2019], la ausencia de interfaces gráficas intuitivas y retroalimentación visual puede provocar una menor implicación del paciente durante el tratamiento, afectando de forma negativa los resultados terapéuticos.

En este contexto, el presente proyecto aborda dicha limitación mediante el diseño y desarrollo de un sistema de interacción gráfica que combina, por un lado, una plataforma de gamificación centrada en el paciente, con el objetivo de fomentar su participación y motivación, y por otro, una interfaz de visualización clínica centrada en el médico, que registre las métricas de desempeño y facilite el seguimiento de la evolución terapéutica.

2.1.1. Objetivo general

Diseñar una interfaz gráfica intuitiva y funcional para que un terapeuta controle y personalice las sesiones terapéuticas post-ictus, así como desarrollar un entorno gamificado que motive y facilite la participación activa del paciente durante su rehabilitación motora unilateral. Con el fin de mejorar la eficacia de dichas sesiones

y aumentar la adherencia del paciente al tratamiento, se monitorizan tanto el progreso como las métricas de ejecución del paciente. De la misma manera, se almacenan los datos más relevantes durante la terapia para facilitar el análisis y la evaluación de los resultados.

2.1.2. Objetivos específicos

Con el fin de alcanzar esta meta, se han definido los siguientes objetivos específicos:

- *O1.* Desarrollar entornos gamificados que incluyan estímulos visuales y auditivos, niveles de dificultad y misiones particulares, adaptadas al perfil del paciente, con el objetivo mejorar su atención y motivación.
- *O2.* Diseñar una interfaz gráfica intuitiva para que el terapeuta controle el desarrollo de la terapia y el modo de rehabilitación, permitiéndole visualizar el progreso del paciente y ajustar los parámetros del juego según su rendimiento.
- *O3.* Establecer una arquitectura capaz de recibir datos de sensores y controlar actuadores para coordinar estímulos visuales y físicos que el paciente experimente durante la terapia, con el fin de crear un entorno inmersivo que potencie la eficacia del tratamiento.
- *O4.* Crear una interfaz gráfica que permita definir los datos personales de un paciente y registrar su progreso después de cada sesión, con el objetivo de personalizar y ajustar de forma automática los parámetros en futuras sesiones.
- *O5.* Integrar perturbaciones controladas en el entorno de juego terapéutico, con el fin de introducir obstáculos que desafíen el control motor y la capacidad de adaptación del paciente, promoviendo una mayor atención, planificación motora y activación neuromuscular.

2.2. Requisitos

A continuación, se enumeran los requisitos que se deben satisfacer:

- Compatibilidad con la arquitectura del sistema robótico y su software de control, garantizando una integración fluida sin afectar al rendimiento del sistema.
- Diseño de una interfaz funcional y adaptable, que se ajuste a las necesidades terapéuticas del usuario impuestas por el equipo rehabilitador.

- Integración de señales de tipo perturbación controladas, para introducir variabilidad y desafíos en a terapia.
- Incorporación de elementos de gamificación, como niveles de dificultad y retroalimentación visual, que aumente la atención del paciente.
- Capacidad para adaptar la asistencia robótica, en función del esfuerzo realizado por el paciente, de forma progresiva.
- Registro de los datos terapéuticos generados durante las sesiones, para permitir su uso y análisis posterior.
- Validación del sistema mediante la valoración de los resultados obtenidos por los usuarios, considerando métricas objetivas y subjetivas.

2.3. Competencias

Durante la realización de este proyecto, se han puesto en práctica diversas competencias generales, específicas y básicas del grado y del desarrollo de interfaces y sistemas dinámicos.

- *CG1*: Diseño software. Conocimiento de materias básicas y tecnologías, que le capacite para el aprendizaje de nuevos métodos y tecnologías, así como que le dote de una gran versatilidad para adaptarse a nuevas situaciones.
- *CB2*: Planificación y sistemas cognitivos. Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.
- *CB4*: Trabajo de Fin de Grado. Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.
- *CE1*. Diseño e implementación de interfaces gráficas de usuario (GUI) con herramientas como Tkinter.
- *CE2*. Desarrollo de aplicaciones interactivas con elementos de gamificación, aplicando principios de diseño centrado en el usuario.

- *CE3*. Integración de interfaces con dispositivos físicos mediante comunicación en tiempo real basada en el Sistema Operativo Robótico (ROS).
- *CE4*. Evaluación de la usabilidad y experiencia de usuario con métricas objetivas y percepciones.
- *CE15*: Arquitectura software para robots. Capacidad de diseñar y programar aplicaciones robóticas y sistemas inteligentes en red usando middlewares, mecanismos de comunicación y estándares propios del ámbito de la Ingeniería Robótica.

2.4. Metodología

Para llevar a cabo este proyecto, se ha optado por seguir el paradigma iterativo centrado en el usuario (User-Centered Design). Previamente, se ha realizado una fase de análisis e investigación sobre el estado del arte y necesidades específicas para comprobar la viabilidad de su desarrollo. Posteriormente, se procedió con el diseño del prototipo para validar el diseño gráfico y la navegación. Más adelante, la realización de pruebas continuas favoreció la incorporación de mejoras en el diseño visual, la lógica de juego y la retroalimentación. Una vez se desarrolló una primera versión fiable, las interfaces se conectaron con el sistema de control del robot permitiendo crear un entorno cambiante y adaptable. Por último, se llevó a cabo una evaluación de conformidad a los usuarios para comprobar la efectividad de la plataforma.

2.5. Plan de trabajo

El trabajo se ha organizado de forma progresiva y coordinada con el resto del equipo técnico. Para ello, el proyecto se ha dividido en las siguientes etapas:

1. *Investigación*. En esta fase inicial, se llevó a cabo un análisis detallado de las arquitecturas robóticas existentes en el ámbito de la rehabilitación, así como las propuestas lúdicas aplicadas en terapias interactivas. El fin de este estudio fue identificar funcionalidades relevantes y enfoques innovadores que sirvieran de inspiración para el diseño y desarrollo de una plataforma que combine control robótico, evaluación clínica y gamificación.
2. *Planteamiento del software*. Una vez se establecieron los objetivos y requisitos del proyecto, se procedió con el diseño conceptual de la plataforma, incluyendo

un esquema de nodos, topics y flujos de datos en ROS 2. Este diseño permitió establecer de forma estructurada la arquitectura del sistema.

3. *Desarrollo de la parte gráfica.* A continuación, se desarrollaron las interfaces gráficas, orientadas al terapeuta, para permitir configurar parámetros clínicos, controlar el sistema robótico y almacenar los datos terapéuticos más relevantes por paciente. Adaptándolo a un diseño intuitivo y claro. A través de continuas pruebas se optimizó tanto la aplicación como el uso de estas interfaces.
4. *Conexión entre el software y hardware.* Una vez consolidada una primera versión fiable del software, se continuó con la integración con el sistema robótico, incluyendo la sensorización del movimiento del paciente. Se validó la comunicación bidireccional entre los componentes software y hardware.
5. *Evaluación y conclusiones.* En esta etapa final, se realizaron pruebas de funcionamiento para evaluar la precisión del sistema, la generación de las señales de control y perturbación, y la usabilidad general de la plataforma. Los resultados obtenidos permitieron sacar conclusiones y recomendaciones para futuras mejoras.

Paralelamente, se ha ido completando la presente memoria a lo largo de este proceso. Asimismo, se han realizado reuniones periódicas con el equipo para compartir los progresos de cada área, proponer mejoras y resolver cuestiones, trabajando de manera coordinada hacia un objetivo común.

A continuación, se procede a tratar las plataformas de desarrollo empleadas en este proyecto.

Capítulo 3

Plataforma de desarrollo

A continuación, se explican las plataformas de desarrollo utilizadas a nivel de software y hardware, que han permitido alcanzar los objetivos descritos en el capítulo anterior.

3.1. Sistema operativo

Un sistema operativo puede definirse como el conjunto de programas que gestionan el hardware de un ordenador o dispositivo y permiten el funcionamiento de otros programas. Como describen [Soriano-Salvador and Guardiola Múzquiz, 2022], una distribución es una colección concreta de software de área de usuario y un núcleo del sistema operativo.

Para la realización de este proyecto se ha utilizado el sistema operativo de tipo GNU/Linux¹ y su distribución Ubuntu², en concreto la versión Ubuntu 22.04. La razón de usar esta versión de Ubuntu se debe a su compatibilidad con la distribución *Humble*³ de ROS 2, ya que el sistema ha sido diseñado específicamente para funcionar con dicha distribución.

3.2. Entorno de desarrollo

3.2.1. Python

Python⁴ es el lenguaje de programación utilizado para el desarrollo de este proyecto. Se trata de un lenguaje de alto nivel creado por Guido van Rossum y publicado por primera vez en 1991. Según la documentación de [Stack Overflow,] es un lenguaje ampliamente reconocido por su simplicidad y flexibilidad, lo que hace que sea ideal tanto para principiantes como para desarrolladores expertos. Cuenta con un sistema

¹<https://www.debian.org/releases/stable/i386/ch01s02.es.html>

²<https://ubuntu.com/>

³<https://docs.ros.org/en/humble/index.html>

⁴<https://www.python.org/>

de tipos dinámico y gestión automática de la memoria y soporta múltiples paradigmas de programación, incluyendo orientación a objetos, programación funcional y estilos procedimentales. Además, dispone de una amplia y completa biblioteca estándar, que facilita el desarrollo de aplicaciones de propósito general.

La versión de Python utilizada para el desarrollo de la GUI es Python 3.10.13. En particular, se emplearon las siguientes librerías:

- *matplotlib*: Matplotlib⁵ es una biblioteca completa para crear visualizaciones estáticas, animadas e interactivas en Python. Se ha empleado para la graficación de las señales de trayectoria y perturbación, la posición del paciente en los entornos gamificados y de visualización del médico, y los datos clínicos recogidos durante la terapia.
- *numpy*: NumPy⁶ es una biblioteca de Python que proporciona un objeto matriz multidimensional, varios objetos derivados como matrices y matrices enmascaradas y una serie de rutinas para realizar operaciones rápidas con matrices, incluidas operaciones matemáticas, lógicas, manipulación de formas, ordenación, selección, entrada/salida, transformadas discretas de Fourier, álgebra lineal básica, operaciones estadísticas simples, simulación aleatoria, entre otros. En especial se ha implementado dentro de los entornos gamificados para crear las señales límites, de trayectoria y perturbación para su posterior representación.
- *tkinter*: Tkinter⁷ es la interfaz por defecto de Python para el kit de herramientas de GUI Tk. En concreto, se han utilizado los módulos Tk y Ttk para el diseño de la interfaz de control del médico y la interfaz de registro de pacientes.
- *pygame*: Pygame⁸ es una biblioteca multiplataforma gratuita y de código abierto para el desarrollo de aplicaciones multimedia. Utilizada para incluir efectos de sonido dentro de un videojuego.

3.2.2. ROS 2

Como explica [Martín Rico, 2023], ROS⁹ es un middleware que aumenta las capacidades del sistema para crear aplicaciones robóticas. El número dos indica que se trata de la segunda generación de este middleware. Los paquetes de ROS 2

⁵<https://matplotlib.org/>

⁶<https://numpy.org/>

⁷<https://docs.python.org/es/3.13/library/tkinter.html>

⁸<https://pypi.org/project/pygame/>

⁹<https://www.ros.org/>

están organizados en distribuciones, las cuales están formadas por múltiples paquetes que aseguran su interoperabilidad. Esta arquitectura modular permite estructurar el sistema como un grafo de computación, compuesto por nodos de ROS 2 que se comunican entre sí, permitiendo así que un robot realice una serie de tareas.

Se destacan las siguientes librerías:

- *rclpy*: *rclpy*¹⁰ proporciona la Interfaz de Programación de Aplicaciones (API) canónica de Python para interactuar con ROS 2.
- *std_msgs*: *std_msgs*¹¹ define tipos de mensajes básicos y estandarizados para la publicación y suscripción entre nodos. Los tipos de mensajes utilizados en este proyecto son *Int32*, *Int32MultiArray*, *Float32* y *Float32MultiArray*.

3.3. Componentes físicos

Una vez definido el software, se especifica el hardware encargado de ejecutar las acciones en un entorno físico. A pesar de que este trabajo se centra en el software del sistema, es importante conocer los dispositivos a utilizar y el propósito de los mismos para desarrollar una arquitectura que se ajuste mejor a las necesidades y alcances del proyecto global.

Para el movimiento del brazo robótico se utiliza un actuador de HEBI Robotics¹², más concretamente el modelo T-Series, como puede observarse en la Imagen 3.1. Este actuador está diseñado para funcionar como un componente robótico con todas las funciones. La salida gira continuamente, no requiere calibración ni referencia alguna de arranque, lo que permite que funcione en brazos robóticos con múltiples grados de libertad.



Figura 3.1: Fuente: HEBI Robotics

¹⁰<https://docs.ros.org/en/iron/p/rclpy/>

¹¹https://docs.ros2.org/foxy/api/std_msgs/index-msg.html

¹²<https://www.hebirobotics.com/actuators>

Destaca por tener un torque preciso y controlado, fundamental en aplicaciones que requieren movimientos suaves y controlados, como es el caso. Gracias a su diseño, puede mantener un torque constante durante el movimiento, lo que asegura precisión y fluidez en las acciones del brazo. Además, permite simular niveles de resistencia y asistencia debido a su ajuste dinámico.

Una de las principales razones de la selección de este modelo es la integración de un sensor de posición, lo que permite registrar el movimiento del brazo del paciente y facilita la validación de los entornos gamificados, los cuales proponen una serie de actividades donde la posición del paciente se toma como referencia para el desarrollo de actividades interactivas y dinámicas dentro de un contexto terapéutico. De la misma manera, el actuador incorpora un control de fuerza, que permite diseñar señales de tipo perturbación que simulan un nivel de resistencia o asistencia durante el ejercicio, lo que favorece la adaptación de la terapia a las capacidades de cada paciente.

Capítulo 4

Diseño

*El diseño no es solo lo que se ve y lo que se siente.
El diseño es cómo funciona.*

Steve Jobs, *conferencia de lanzamiento del Apple iPod, 2001*

Después de definir la plataforma de desarrollo, se procede a explicar el diseño de la aplicación.

El sistema está diseñado para gestionar la rehabilitación motora de pacientes post-ictus mediante la interacción con un brazo robótico. Esto se lleva a cabo a través de una aplicación que controla diversos parámetros que afectan a la trayectoria, las perturbaciones y nivel de asistencia del brazo. A continuación, se definen los puntos clave del sistema:

- **Trayectoria:** representa el recorrido deseado que el paciente debe seguir durante la terapia. Es una señal definida por los parámetros de frecuencia, amplitud y tipo de señal (trayecto constante o dinámico). El objetivo es medir la precisión del paciente al seguir la trayectoria.
- **Perturbación:** es una señal que altera la trayectoria. Se define mediante los parámetros de amplitud, duración y tipo (perturbación escalón o sinusoidal). El objetivo es proporcionar un desafío adicional al paciente.
- **Límites:** definen las restricciones físicas de movilidad del brazo robótico. Se establecen un límite inferior (mínimo) y superior (máximo), así como un offset que determina la posición inicial del brazo. Se ajustan llevando el brazo a la posición deseada.
- **Asistencia:** se refiere al grado de apoyo que ofrece el brazo robótico al paciente. Este parámetro se ajusta tanto de forma dinámica, según el nivel de dificultad del juego, como manual, a través de la interfaz de control.

- Nivel de dificultad: hace referencia al nivel de dificultad del juego. El nivel de dificultad aumenta de forma progresiva, reduciendo el nivel de asistencia del robot. Puede ajustarse tanto dinámicamente como manualmente al igual que la asistencia.

El sistema combina una lógica de ajuste automática y manual para la asistencia y nivel de dificultad. El ajuste automático permite adaptar la terapia en tiempo real según el rendimiento del paciente, lo que favorece una progresión personalizada y reduce la necesidad de intervención del terapeuta. Por otro lado, el control manual ofrece flexibilidad al terapeuta para modificar los parámetros según criterios específicos o en situaciones particulares. Esta dualidad mejora la versatilidad del sistema y permite adaptarlo tanto a sesiones autónomas como supervisadas.

Durante el desarrollo surgieron distintos desafíos que condicionaron el diseño final de la plataforma. Entre ellos se destaca la diferencia en los rangos de movimiento de los diferentes perfiles terapéuticos. Para ello, se decidió representar las gráficas estableciendo un tamaño fijo de ventana en base a los valores de los límites mínimo y máximo definidos. De esta manera se crea un margen visual más amplio que permite al usuario centrarse en la dinámica del juego.

Más adelante, se describen los nodos y topics utilizados en la plataforma.

4.1. Esquema de nodos y topics

En la Imagen 4.1 se contempla un esquema de los nodos, topics y tipos de datos que se utilizan para la comunicación de la plataforma software con el actuador.

Seguidamente, se detalla la función de los nodo y la información transmitida por cada uno de los topics:

- **LimitNode**: este nodo gestiona la configuración de los límites del brazo robótico. Utiliza el topic `/RobotLimits`, como un booleano para indicar el límite que se ajusta.
- **ScrollPublisherNode**: este nodo publica los parámetros de la trayectoria y la perturbación en el topic `/SliderParameters`, y del juego (asistencia y nivel de dificultad) en `/GameParameters`. La actualización de los datos se realiza cada *50ms*.

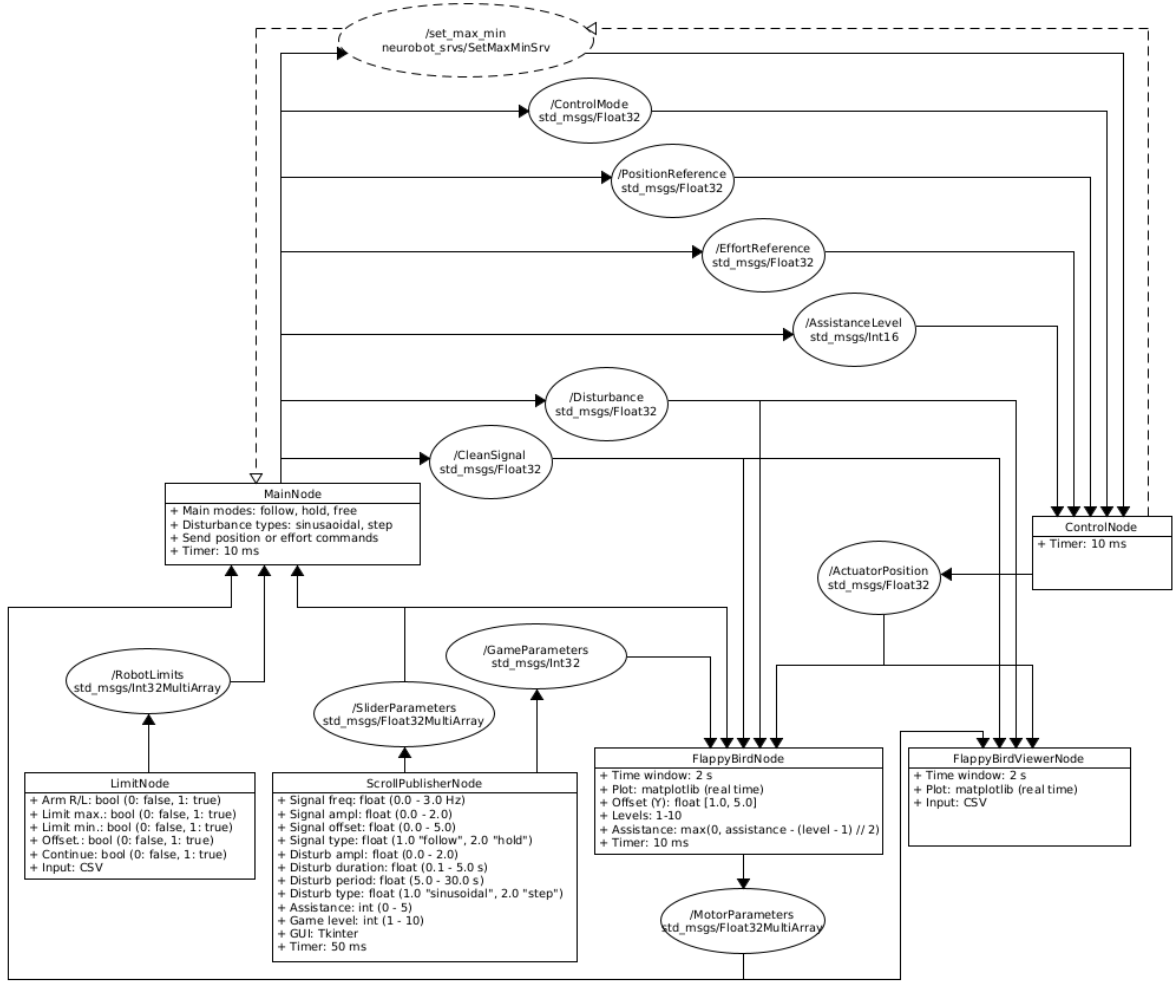


Figura 4.1: Esquema de nodos y topics

- **FlappyBirdNode**: este nodo implementa la lógica del juego basado en seguir una trayectoria o mantener una posición. Recibe la señal de referencia del topic `/CleanSignal`, la señal de perturbación de `/Disturbance`, la asistencia y nivel de dificultad de `/GameParameters`, los límites y el offset de `/SliderParameters` y la posición del jugador de `/ActuatorPosition`. Y publica los datos de referencia de las señales y tiempo, en el momento de la posición del jugador en el eje X, en el topic `/MotorParameters`. La actualización de los datos se realiza cada $10ms$.
- **FlappyBirdViewerNode**: este nodo grafica las señales de trayectoria, perturbación y la posición del paciente, que recibe de los topics `/CleanSignal`, `/Disturbance` y `/ActuatorPosition`, respectivamente, para visualizar el progreso del paciente durante la terapia. Además, calcula el error de posición a través de los datos publicados en el topic `/MotorParameters`, explicados en el nodo anterior.

La arquitectura modular de los nodos permite una separación clara entre la lógica del juego, la visualización y el control de la plataforma. Esta estructura

facilita el desarrollo, pero introduce restricciones en cuanto a la sincronización de los datos. Pueden producirse desfases leves en sesiones prolongadas o con múltiples perturbaciones simultáneas debido a la diferencia entre la frecuencia de publicación y la latencia de los callbacks. No obstante, la frecuencia fija de actualización es suficiente para mantener una ejecución estable.

El proyecto se divide en cuatro scripts, el primero se utiliza para crear o registrar un paciente, el segundo lanza la GUI y controla los parámetros del juego, el tercero permite visualizar el comportamiento del paciente durante la terapia, y el cuarto es el propio juego.

4.2. Interfaz de registro de un paciente

Este script permite gestionar un conjunto de datos, que registran a un paciente, mediante una GUI.

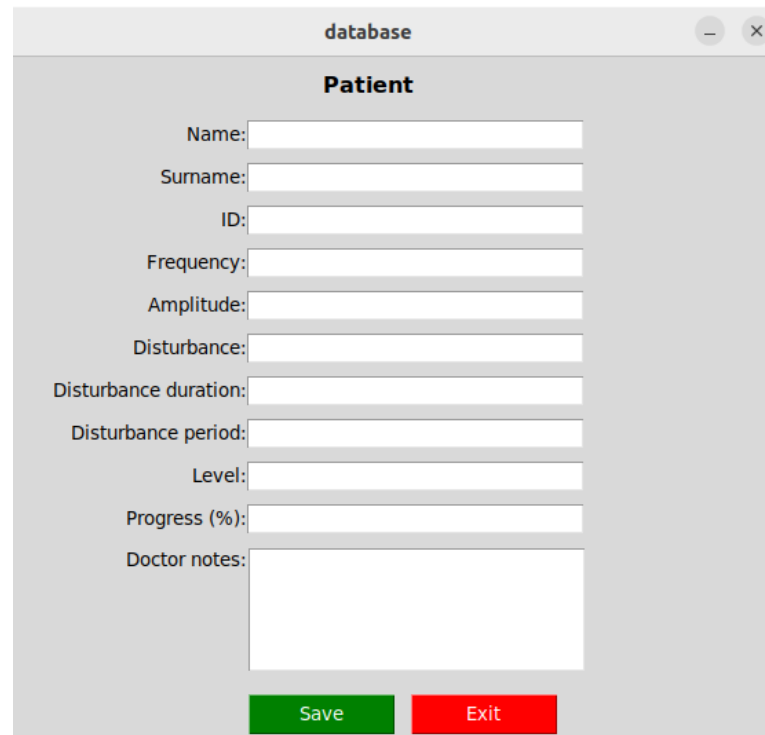
En primer lugar, se importan las bibliotecas estándar, mencionadas en el capítulo anterior, como `tkinter`, que se utiliza para crear la interfaz gráfica, `csv` para guardar los datos en un archivo CSV para su posterior uso, y `os` para interactuar con el sistema de archivos.

Se obtiene el directorio de inicio del usuario y se crea un directorio dentro de este, si no existe, bajo el nombre *database*, donde se almacenan los ficheros con los datos de registro y análisis terapéuticos de cada paciente.

Se definen tres funciones principales que gestionan las operaciones de la GUI. La función `exit()` cierra la ventana principal de Tkinter, `clear()` limpia los campos de entrada y texto, y `savedata()` guarda los valores de los campos, valida que el ID sea un número y crea un subdirectorio bajo el nombre del ID, si no existe, donde guarda los datos en un archivo CSV llamado `ID.csv`. Si el archivo no existe, se escribe el encabezado utilizando `writer.writeheader()`. Los datos se escriben con `csv.DictWriter` y son guardados como `strings`.

Se crea una ventana principal con un título y tamaño fijo de 600×500 píxeles. Los estilos visuales se configuran con `ttk.Style` y se definen dos frames distintas, `main_frame` se utiliza como contenedor de los campos de entrada y texto y `button_frame` agrupa la lógica de los botones. Cada campo es un `Entry` enlazado a una variable de tipo `StringVar` y hacen referencia al nombre, apellido e ID del paciente, frecuencia, amplitud y perturbación de la señal que generará el brazo robótico, nivel y progreso del juego y un espacio para que el doctor incluya observaciones. Se crean

dos botones, *Save* llama a `savedata()` para guardar los datos, que a su vez llama a `clear()` para limpiar los campos una vez que estos se han almacenado, y el botón *Exit* llama a la función `exit()` que cierra la aplicación. En la Imagen 4.2 puede observarse el estilo de la interfaz.



The image shows a Tkinter window titled "database" with a standard macOS-style title bar (minimize, maximize, close buttons). Inside the window, the title "Patient" is centered at the top. Below it, there is a vertical list of labels followed by input fields: "Name:", "Surname:", "ID:", "Frequency:", "Amplitude:", "Disturbance:", "Disturbance duration:", "Disturbance period:", "Level:", "Progress (%)", and "Doctor notes:". The "Doctor notes" field is a larger text area. At the bottom of the window, there are two buttons: a green "Save" button and a red "Exit" button.

Figura 4.2: Interfaz de registro de un paciente

`root.mainloop()` inicia el bucle de eventos de Tkinter y la interfaz está activa hasta que el usuario cierra la ventana.

4.3. Interfaz de control

Este script combina ROS 2 y Tkinter para crear una GUI que gestiona límites físicos (mínimo, máximo y offset) del robot, adaptándose al brazo específico del paciente que se va a rehabilitar, configura y publica parámetros para crear una señal de control y perturbación para un sistema robótico, parámetros de asistencia y nivel de dificultad para un videojuego terapéutico y almacena dichas configuraciones por paciente.

Al inicio del script se importan las librerías `rclpy` y `std_msgs.msg` para gestionar la comunicación entre Python y ROS 2, `tkinter` para crear la interfaz gráfica, `csv` para el manejo de archivos CSV, `os` para interactuar con el sistema y `datetime` para gestionar la fecha de creación de los archivos.

Se crea un directorio en el directorio de inicio del usuario, si no existe, bajo el nombre *database*, donde se almacenan los ficheros con los datos de configuración de

cada paciente.

Se definen dos clases, `ScrollPublisherNode` y `ScrollGUI`. La primera es una clase de ROS 2 que publica los parámetros de dos señales, de tipo trayectoria y perturbación en el topic `/SliderParameters` y los datos de asistencia y nivel de juego en `/GameParameters`. Los mensajes son de tipo `Float32MultiArray` y `Int32MultiArray`, y se publican cada `50ms` y únicamente cuando se actualiza el valor de los datos, respectivamente. La decisión de publicar ambas señales en el mismo topic a dicha velocidad se debe a que la actualización de los datos debe realizarse de forma casi instantánea para permitir ajustes inmediatos en los parámetros de la terapia y evaluar la rapidez de respuesta del paciente. Y, se ha comprobado que esta estimación es lo suficientemente rápida para satisfacer estos requisitos. Los datos necesarios para generar la señal de trayectoria son frecuencia, amplitud y tipo, mientras que para la perturbación son duración, amplitud y tipo. El tiempo de las señales se define de manera distinta para evitar redundancias, ya que la perturbación no es constante. En su lugar, se estima una duración que facilita el entendimiento del terapeuta. El tipo de señal hace referencia al modo de juego y se codifica como 1.0 (hold o mantener) y 2.0 (follow o seguir), y el tipo de perturbación como 1.0 (sinusoidal) y 2.0 (step o escalón). También se define un offset que determina la distancia entre la señal principal y los límites superior e inferior, ajustando así los márgenes del camino. La segunda clase crea la interfaz gráfica y permite ajustar los parámetros del juego en tiempo real y publicarlos a través de un nodo de ROS. Los datos se guardan en un archivo CSV bajo el nombre `ID-year-month-day-config_<index>.csv` en un subdirectorío llamado *config* dentro del directorío `home/user/database/ID/`. `index` hace referencia al número de archivo de la sesión diaria. Si el archivo no existe, se escribe un encabezado con los nombres de los campos a los que hacen referencia los datos como se observa en el Código 4.1.

```
header = ["frequency", "amplitude", "offset", "signal", "disturbance",
          "duration", "period", "mode"]
```

Código 4.1: Encabezado del fichero de configuración

Se crean deslizadores, utilizando `ttk.Scale`, para ajustar la frecuencia, amplitud, offset, duración y periodo de las señales, lo que facilita la configuración de los parámetros con mayor precisión y comodidad. Se utiliza `Combox` para permitir la selección entre los distintos tipos de señal y perturbación y niveles de asistencia (del 0 al 5, donde 0 es asistencia nula y 5 asistencia máxima) y de juego (del 1 al 10, de menor a mayor dificultad). Los botones *Update Signal* y *Update levels* se utilizan para

actualizar los datos en el topic correspondiente, y el botón *Exit* para salir. En la Imagen 4.3 se observa el aspecto de la ventana principal de control.

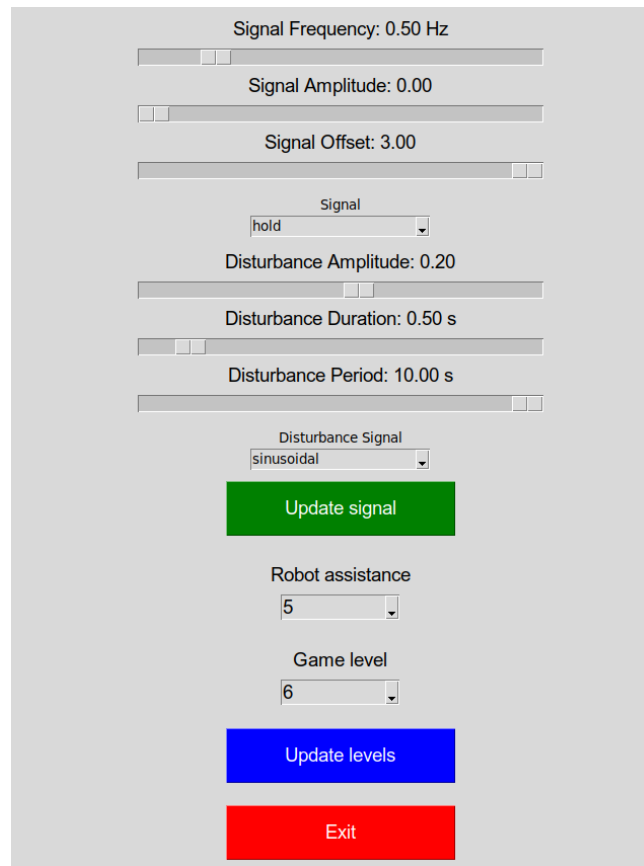


Figura 4.3: Interfaz de configuración de los parámetros terapéuticos

La función `saveconfig()` guarda la configuración actual de la GUI en el archivo CSV. Las funciones `update*()` convierten el valor del deslizador de `str` a `float`. Entre ellas se destacan `updatesignal()`, que actualiza los atributos del nodo y llama a `saveconfig()`, que añade al final del archivo la nueva configuración, y `updatelevel()`, que publica un mensaje con los datos de asistencia y nivel de dificultad del juego. La función `close()` finaliza el nodo y cierra la GUI y el método `run()` inicia el bucle de eventos de Tkinter junto con `spin_once()`. `loadcsv()` lee el último registro del archivo del paciente que se pasa como parámetro y devuelve los valores de frecuencia (Hz), amplitud (rad) de las señales de trayectoria y perturbación, duración (s), periodo (s) y nivel del juego. Todos son `float` excepto el último que es un `int`.

`main()` es la función principal y se encarga de extraer el ID del paciente desde la ruta al archivo de registro, cargar los datos de las señales y el juego desde el CSV, crear el nodo `ScrollPublisherNode` con dichos parámetros e iniciar la GUI. `startgui()` es la primera interfaz que se lanza y gestiona la configuración del brazo que se va a rehabilitar y de los límites físicos del robot a través de la publicación de un `Int32MultiArray`

que actúa como un booleano indicando el botón que se ha pulsado (resetear, mínimo, máximo u offset). El offset corresponde a la posición inicial desde la cual comienza la terapia. Además, comprueba que se ha seleccionado o bien brazo derecho o izquierdo y que los límites mínimo y máximo se han definido correctamente antes de permitir establecer el offset o continuar a la siguiente ventana. La decisión de ajustar los límites y el offset mediante el movimiento del brazo robótico a una posición específica, y luego guardar dicha posición a través de un botón, responde a un requisito explícito del cliente. En la Imagen 4.4 se muestra el formato de la ventana de inicio.

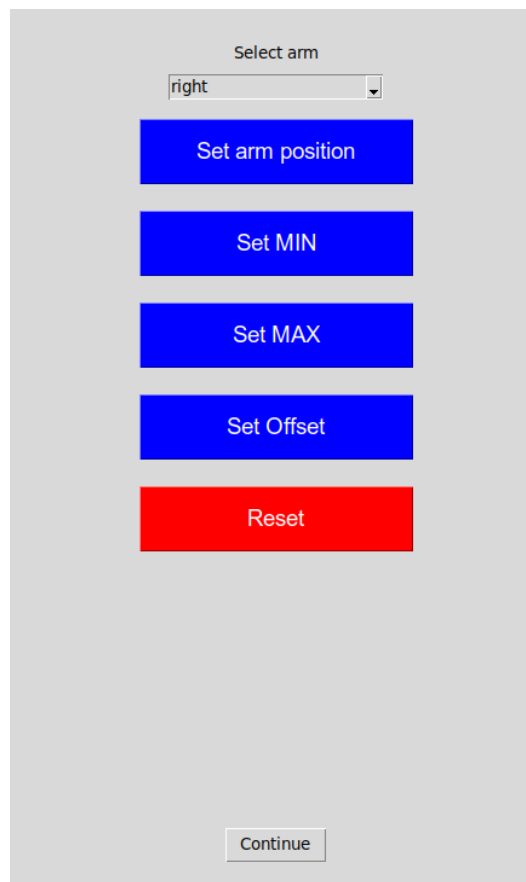


Figura 4.4: Interfaz de configuración de los límites y orientación del brazo robótico

4.4. Interfaz de visualización

Este script permite observar y registrar la ejecución de un paciente en una tarea de seguimiento de trayectoria como parte de una sesión de rehabilitación motora.

Además de incluir las bibliotecas mencionadas en la Sección 4.3, se utilizan `matplotlib` para visualizar los datos del rendimiento del jugador en tiempo real, `numpy` para manejar los datos de las señales que cambian con el tiempo y `sys` para gestionar la ruta al archivo CSV con los datos de registro del paciente.

Se utiliza el modo interactivo de `matplotlib`, `plt.ion()`, que permite actualizar dinámicamente los gráficos sin bloquear el hilo principal. Las ventanas son deslizantes respecto al eje X, no obstante debemos mantener un tamaño fijo para que la señal no se desplace hacia la izquierda, por ello se implementa `np.roll()`. El tamaño de la ventana en el eje Y se ajusta a 3 ya que es el máximo recorrido que puede hacer el brazo.

Se define la clase `FlappyBirdViewerNode` que actúa como nodo de ROS 2 y se suscribe a los topics `/CleanSignal`, donde se publica la señal de referencia que se toma como trayectoria deseada que el jugador debe seguir, `/Disturbance`, que contiene la señal de perturbación, `/ActuatorPosition`, que publica la posición del jugador en el eje Y, y `/MotorParameters`, para obtener el tiempo del juego, la posición del jugador en el eje X y el nivel de asistencia del robot. Los mensajes de los primeros tres topics son de tipo `Float32` y el último es un `Float32MultiArray`. `signalcallback()` se activa cada vez que se recibe un valor de la señal, si el jugador está activo se actualizan los datos (tiempo, límites, posición, error de trayectoria y detección de colisiones) y se grafican. Para la detección de colisiones se implementa una lógica simple basada en la comparación de la posición del jugador con los límites de la señal. `playercallback()` actualiza la posición del jugador, el offset que separa la señal de los límites y el tiempo total y marca que el jugador está activo.

En la Imagen 4.5, se muestran dos gráficos. El que está en la parte superior muestra la señal de trayectoria (línea azul), la perturbación de tipo escalón (línea verde), los límites superior e inferior (líneas discontinuas grises) y la posición actual del jugador (punto rojo), proporcionando una visión clara y completa de la terapia. Y, la parte inferior grafica el error de posición con respecto a la trayectoria deseada en función del tiempo, lo que permite detectar fallos de control, fatiga o pérdida de atención.

En un principio, el error se calculaba en función del tiempo como se muestra en la Ecuación 4.1, pero, más adelante, se optó por calcular la interpolación (o selección por índice más cercano) del valor de la trayectoria de referencia (x, y) evaluando la posición del jugador en el eje X, x_p , como se muestra en la Ecuación 4.2. El segundo cálculo ofrece una aproximación más precisa al error, ya que está directamente relacionado con la ubicación del jugador. A diferencia del primer cálculo, que depende del instante de tiempo y puede verse afectado por variaciones en la sincronización de los datos.

Al finalizar la ejecución, los datos más relevantes como el tiempo, la señal, los límites mínimo y máximo, la posición del jugador, el offset, el error, las colisiones y el nivel de asistencia se almacenan en un archivo CSV bajo el nombre `ID-year-month-day-metrics_<index>.csv` en un subdirectorio llamado *metrics*

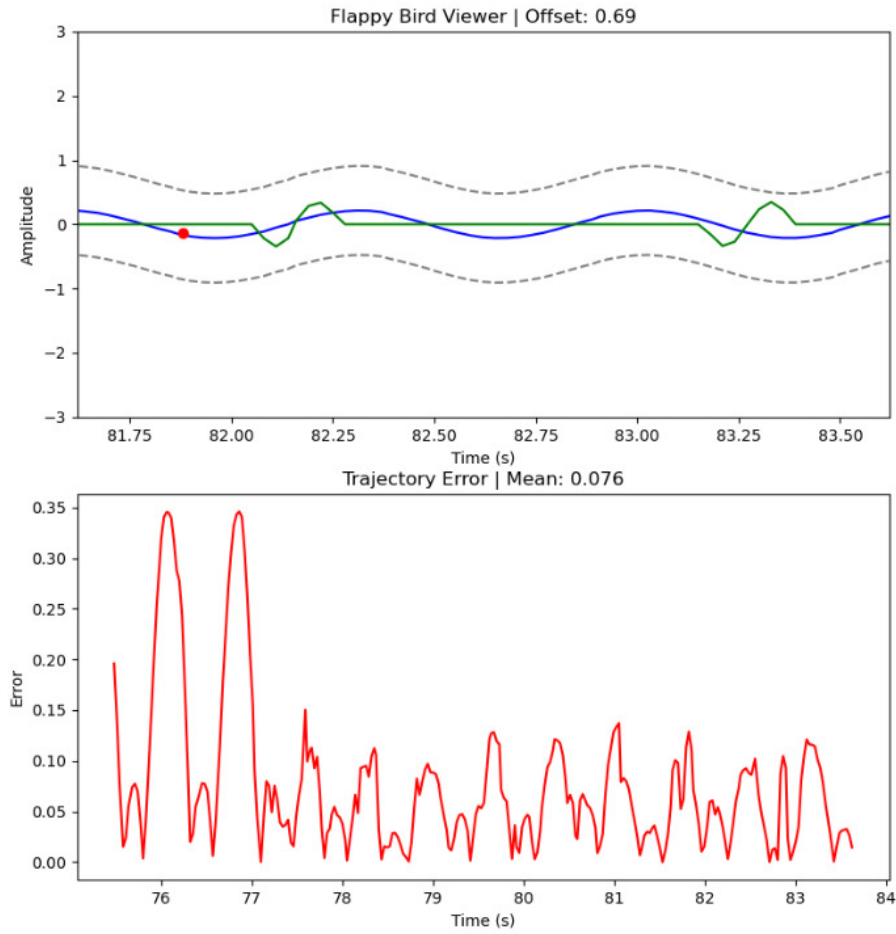


Figura 4.5: Interfaz de visualización del terapeuta

dentro del directorio `home/user/database/ID/`. `index` hace referencia al número de archivo de la sesión diaria. Esto facilita un análisis posterior de la terapia.

La función principal `main()` verifica que se ha pasado como único argumento un archivo CSV con los datos de registro del paciente, extrae el ID desde la ruta, crea el nodo `FlappyBirdViewerNode` y escucha de los topics.

4.5. Juego flappy

Este script implementa un juego basado en el Flappy Bird, adaptado a un entorno ROS 2 para interactuar con sensores y actuadores en tiempo real. Existen dos modos de juego distintos, *hold*, el jugador debe mantener la posición, y *follow*, el jugador debe seguir una trayectoria.

Se implementan las librerías `rclpy`, y `std_msgs` para permitir la comunicación con ROS 2, `matplotlib` y `numpy` para la visualización y el manejo de señales, y `pygame` para utilizar efectos de sonido.

Se emplean sonidos distintos cuando se sube de nivel, se consigue una recompensa

$$error_t = |y_{player}(t) - y_{reference}(t)| \quad (4.1)$$

Ecuación 4.1: Cálculo del error de trayectoria en el tiempo

$$error(x_p, y_p) = |y_p - f(x_p)| \quad (4.2)$$

Ecuación 4.2: Cálculo del error de trayectoria por posición

o se colisiona. Esto permite que el juego sea más interactivo. En el Código 4.2, escrito en *Python*, se observa como se carga un sonido desde un directorio, *sounds*, dentro del paquete del proyecto. Los formatos de audio más comunes soportados por `pygame.mixer` son *.wav*, *.ogg* y *.mp3*. La elección de utilizar la extensión *.wav* se debe a su simplicidad, alta calidad y compatibilidad con todos los sistemas.

```
pygame.mixer.init()
level_up_sound = pygame.mixer.Sound('../sounds/level_up.wav')
```

Código 4.2: Cargar un sonido al juego

La clase `FlappyBirdNode` engloba la lógica del programa. En `__init__` se crean los suscriptores `/CleanSignal`, de tipo `Float32`, es la señal de referencia, `/Disturbance`, también `Float32`, es la perturbación, `/GameParameters`, es un `Int32MultiArray`, contiene el nivel del juego y de asistencia del robot, `/SliderParameters`, `Float32MultiArray`, para obtener el offset entre la señal y los límites superior e inferior, y `/ActuatorPosition`, es un `Float32`, contiene la posición del jugador en el eje Y. Se crea un publicador `Float32MultiArray`, llamado `/MotorParameters`, que contiene la marca de tiempo de la ventana, las referencias de la señal y la perturbación en (x_p, y) , la posición del jugador en el eje X y la asistencia actualizada. Se crea una ventana gráfica de tamaño $2x3$, donde 3 es el recorrido máximo del brazo. En ella se representan la señal principal, los límites mínimo y máximo, las perturbaciones, la posición del jugador y objetos como estrellas y asteroides que forman parte de la lógica del juego.

La posición del jugador se representa como un punto rojo que cambia de color según la distancia a la que se encuentra entre la trayectoria deseada y los límites (negro significa que está cerca de la trayectoria, gris que está a una distancia media y rojo indica peligro de colisión con los límites). La posición se representa como una coordenada en los ejes XY, en el eje X permanece siempre a la misma distancia (a 0,25 unidades del origen), y en el eje Y se muestra la posición que se recibe del sensor de posición del motor. En un principio, para validar el movimiento del jugador, la posición en el eje Y se basaba en el manejo de las flechas arriba y abajo del teclado. Para ello

se utilizó la librería `pynput`. En el Código 4.3, escrito en Python, se muestra la lógica de las teclas.

```
def on_press(self, key):
    try:
        direction = -1 if self.inverted_gravity else 1
        if key == keyboard.Key.up:
            self.player_y += 0.1 * direction
        elif key == keyboard.Key.down:
            self.player_y -= 0.1 * direction
    except:
        pass
```

Código 4.3: Movimiento vertical del jugador

El método `updatelevel()` cambia los colores del fondo y de la línea según el nivel, reproduce el sonido de subida de nivel y disminuye la asistencia para que la dificultad sea progresiva. La asistencia del robot se disminuye utilizando la Fórmula 4.3. Se utiliza el máximo entre 0 y el valor calculado según el nivel para que la asistencia no sea un valor negativo.

$$assistance = \max(0, assistance - (level - 1)/2) \quad (4.3)$$

Ecuación 4.3: Cálculo del nivel de asistencia según el nivel de dificultad

Para crear un juego dinámico que capte la atención del paciente se crea una misión secundaria en cada nivel que se explica a continuación:

- Niveles 1, 4 y 7: tienen como objetivo permanecer dentro de los límites por un tiempo determinado. Esto permite evaluar la precisión del paciente. Además, cuando se sobrepasa uno de los límites el contador se para y se reanuda de nuevo cuando se vuelve al camino.
- Niveles 2 y 6: se enfocan en el ajuste de la posición para chocar con las estrellas que están dispuestas en la señal principal. De esta forma se consigue que el paciente permanezca y siga la trayectoria deseada.
- Niveles 3 y 8: pretenden ejercitar la rapidez de reacción mientras el paciente evita colisionar con los asteroides.
- Niveles 5 y 9: fomentan la flexibilidad cognitiva del paciente para adaptarse a una nueva lógica de control basada en el concepto de gravedad invertida (arriba es abajo y viceversa).

- Nivel 10: permite al médico evaluar distintos escenarios límites mediante ajustes en los parámetros de la señal, perturbación y el offset.

En las Imágenes 4.6 y 4.7 se comparan dos niveles diferentes del juego. La primera imagen presenta límites más estrictos que la segunda, que se han coloreado para que se vean claramente en el estudio, pero en el juego son dos líneas de color blanco punteadas para evitar distracciones del paciente. En la segunda imagen, se observan dos perturbaciones de tipo sinusoidal con sentidos opuestos, representadas por triángulos verdes cuya punta indica el signo de la perturbación. Si fuese una perturbación de tipo escalón, los triángulos se pintan de color naranja siguiendo la misma lógica de graficación. En contraste, la primera imagen no muestra ninguna perturbación.

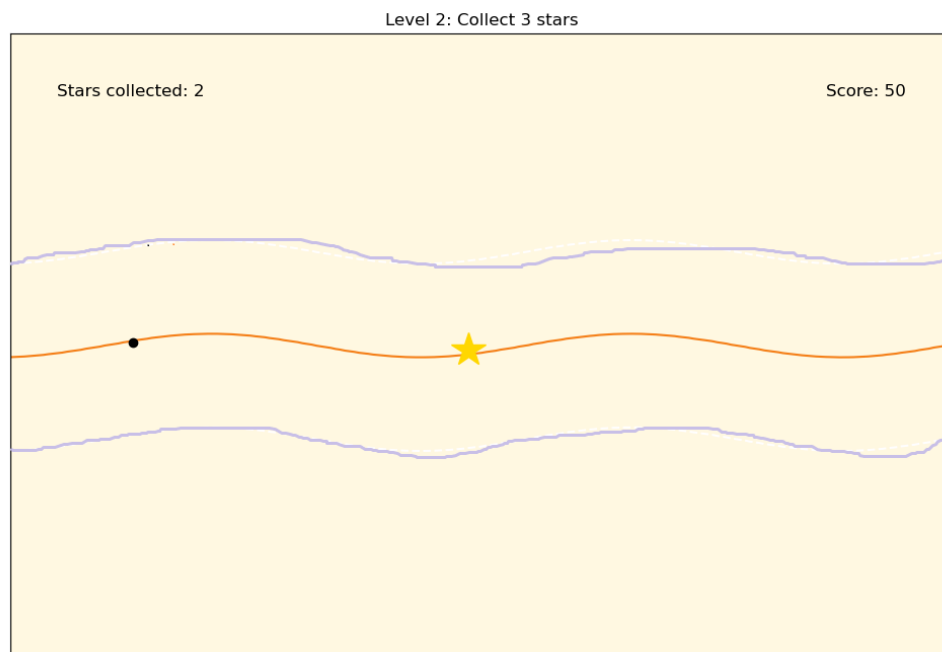


Figura 4.6: Nivel 2 del juego sin perturbación

Se introduce un sistema de recompensas para aumentar la motivación y destreza del paciente. Esto permite reforzar comportamientos y habilidades específicas para alcanzar los objetivos propuestos. Cada vez que se consigue una estrella y se sube de nivel se suman 10 y 30 puntos, respectivamente, a un contador. Y, cuando se colisiona con un asteroide, el contador decrementa en 5 puntos. Para ello se utilizan las funciones `incrementscore()` y `decrementscore()`.

Los objetos como las estrellas, los asteroides y los triángulos, que indican el sentido y comienzo de una perturbación, se crean con `ax.plot`. Las estrellas se generan de forma aleatoria en el eje X, y los asteroides en ambos ejes con límites definidos entre $(0, 3, offset)$ y $(-offset, -0, 3)$. El máximo de objetos visuales al mismo tiempo

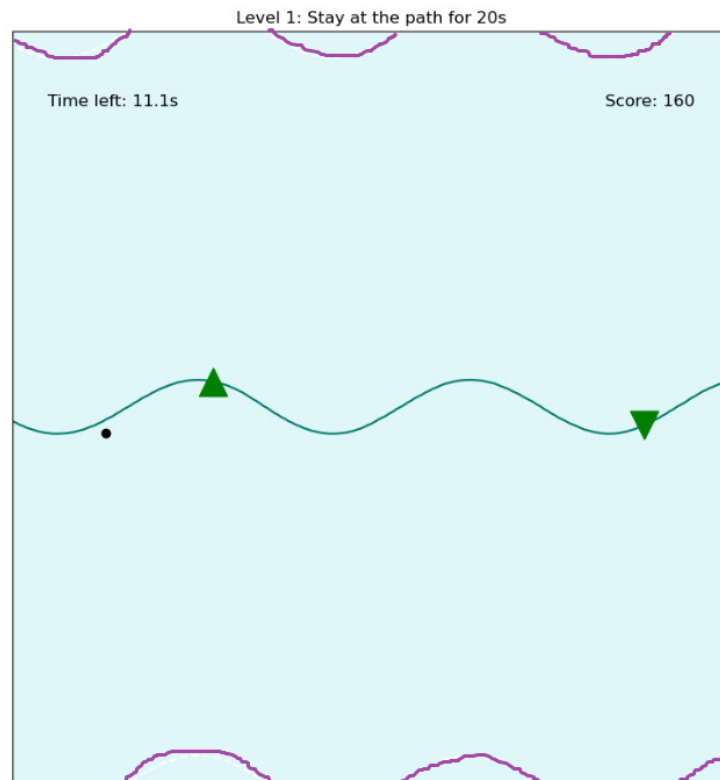


Figura 4.7: Nivel 1 del juego con perturbación

dentro de la ventana es 2. Las funciones `generatestar()`, `generateasteroid()` y `generatedisturb()` encapsulan la lógica de creación de objetos, y la función `clearobjects()` la eliminación de estos.

El desarrollo del juego está concentrado en `listenercallback()`, que es el bucle principal y se ejecuta cada vez que se recibe un nuevo dato en el topic `/CleanSignal`. Se utilizan las funciones `np.roll` y `plt.ion()` de las bibliotecas NumPy y Matplotlib, respectivamente. `np.roll` actualiza la señal y la perturbación. Como se explica en la Sección 4.4, los gráficos se actualizan en tiempo real con `plt.ion()`.

Los callbacks actualizan los datos que se reciben de los topics, estas funciones son `disturbcallback()`, `levelcallback()`, `offsetcallback()` y `positioncallback()`.

La función `main()` inicializa el nodo ROS, crea una instancia del juego y entra en un bucle hasta que el usuario detenga el programa.

El proyecto es fácilmente escalable en el caso que se quiera añadir una señal que sea combinación de otras. Simplemente se debe obtener la señal del topic `/CleanSignal` si es de tipo trayectoria o de `/Disturbance` si es de tipo perturbación. En cuanto a la selección de esta nueva señal, es suficiente con incluir una nueva opción dentro de la `Combox` a la que hace referencia.

Capítulo 5

Conclusiones

En este último capítulo se detallan los objetivos y requisitos cumplidos, se redactan las conclusiones que se han obtenido a lo largo de todo el proyecto y se describen las posibles líneas futuras que dan continuidad a este trabajo de fin de grado.

Se ha alcanzado el objetivo principal de este proyecto, esto es, diseñar una interfaz gráfica intuitiva y funcional para que un terapeuta controle y personalice las sesiones terapéuticas post-ictus, así como desarrollar un entorno gamificado que motive y facilite la participación activa del paciente durante su rehabilitación motora unilateral. A su vez, se han cumplido todos los subobjetivos definidos en la Sección 2.1.2 y los requisitos establecidos en la Sección 2.2:

1. Se ha desarrollado un entorno gamificado personalizado al perfil de cada paciente, que incluye diferentes estímulos visuales y auditivos, niveles de dificultad y asistencia, y misiones particulares.
2. Se han integrado perturbaciones controladas en el entorno de juego terapéutico, lo que hace que sea variable y presente un desafío en el control motor y la capacidad de adaptación del paciente.
3. Se ha diseñado una interfaz gráfica funcional y adaptable para que el terapeuta controle el desarrollo de la terapia y el modo de rehabilitación, permitiéndole visualizar el progreso del paciente y ajustar los parámetros del juego según su rendimiento.
4. La arquitectura software de la plataforma es capaz de recibir datos de sensores y controlar actuadores para coordinar los estímulos visuales y físicos que el paciente experimente durante la terapia.
5. Se ha creado una interfaz gráfica que permite definir los datos personales de un paciente y registrar su progreso después de cada sesión.

6. Se ha evaluado la usabilidad de la plataforma a través de una encuesta de satisfacción a los usuarios.

Se concluye que los entornos gamificados incrementan significativamente la motivación de los usuarios durante el proceso de rehabilitación, según las encuestas realizadas. Los usuarios disfrutaban más cuando la terapia se desarrolla en un entorno dinámico e interactivo. Además, la introducción de misiones secundarias dentro del juego ha demostrado una mejora del nivel de compromiso de los usuarios, ya que se sienten más involucrados en el proceso.

La personalización de las sesiones de rehabilitación, adaptadas a las necesidades de cada paciente, permite que el sistema se ajuste a su progreso y ofrece mejores resultados terapéuticos. Según las encuestas, los usuarios se muestran más satisfechos con esta adaptación, porque les ofrece una terapia diseñada específicamente para sus capacidades. Esto mejora la eficacia del tratamiento y deriva en una mayor adherencia.

Para finalizar, se proponen las siguientes mejoras que permiten la continuidad de este proyecto:

- Mostrar el comienzo de la señal sinusoidal de tipo perturbación, en vez de un dibujar un triángulo, para que el usuario identifique el movimiento de forma más clara.
- Integrar una nueva señal que sea combinación de otras para introducir variabilidad a las señales de trayectoria y perturbación.
- Crear un script que analice el rendimiento del paciente a partir de los datos terapéuticos almacenados.
- Diseñar un nuevo juego enfocado en la mejora de la coordinación motora, ofreciendo una alternativa de rehabilitación.
- Evaluar la plataforma en pacientes con ictus para determinar su eficacia en un entorno sanitario.

Bibliografía

[AlmirallMed, 2021] AlmirallMed (2021). La gamificación en el ámbito de la salud. <https://neurologia.almirallmed.es/blog/la-gamificacion-en-el-ambito-de-la-salud/> [Accedido: 30/05/2025].

[arXiv, 2022] arXiv (2022). Midas: Multi-sensorial immersive dynamic autonomous system improves motivation of stroke affected patients for hand rehabilitation. <https://arxiv.org/abs/2203.10536> [Accedido: 30/05/2025].

[Casadomo, 2015] Casadomo (2015). Un juego para la rehabilitación de pacientes que han sufrido ictus. <https://www.casadomo.com/2015/09/22/un-juego-para-la-rehabilitacion-de-pacientes-que-han-sufrido-ictus> [Accedido: 30/05/2025].

[Clark et al., 2019] Clark, W. E., Manoj, S., and O'Connor, R. J. (2019). Evaluating the use of robotic and virtual reality rehabilitation technologies to improve function in stroke survivors: A narrative review. *Rehabilitation and Assistive Technologies Engineering*, 6:3–5.

[Coote et al., 2008] Coote, S., Murphy, B., Harwin, W., and Stokes, E. (2008). The effect of the gentle/s robot-mediated therapy system on arm function after stroke. *Clin Rehabil.*, 22(5):395–405.

[de la Salud, 2022] de la Salud, O. M. (2022). World stroke day 2022. <https://www.who.int/srilanka/news/detail/29-10-2022-world-stroke-day-2022> [Accedido: 24/05/2025].

[de Rehabilitación y Medicina Física, 2024] de Rehabilitación y Medicina Física, S. E. (2024). El 50rehabilitación temprana reducirá se dependencia en un 20 <https://www.sermef.es/el-50-de-los-supervivientes-de-ictus-sufre-secuelas-y-la-rehabilitacion-temprana> [Accedido: 26/05/2025].

- [del Ictus, 2024] del Ictus, F. E. (2024). Día mundial del ictus: recursos para estudiar esta enfermedad. <https://mirial.es/blog/estudiante-medicina/82-dia-mundial-del-ictus-recursos-para-estudiar-esta-enfermedad> [Accedido: 30/05/2025].
- [Hernandez-Echarren and Sanchez-Cabeza, 2023] Hernandez-Echarren, A. and Sanchez-Cabeza, A. (2023). Hand robotic devices in neurorehabilitation: A systematic review on the feasibility and effectiveness of stroke rehabilitation. *Rehabilitacion (Madr.)*, 57(1):100758.
- [Krebs et al., 2004] Krebs, H., Ferraro, M., Buerger, S., Newbery, M., Makiyama, A., and Sandmann, M. (2004). Rehabilitation robotics: pilot trial of a spatial extension for mit-manus. *Neuroengineering Rehabil.*, 1:5.
- [Lee et al., 2023] Lee, B.-O., Saragih, I. D., and Batubara, S. O. (2023). Robotic arm use for upper limb rehabilitation after stroke: A systematic review and meta-analysis. *Kaohsiung J Med Sci*, 39:435–445.
- [Martín Rico, 2023] Martín Rico, F. (2023). A concise introduction to robot programming with ros2. pages 1–5. Este libro se ha preparado a partir de una copia camera-ready proporcionada por los autores.
- [Mubin et al., 2022] Mubin, O., Alnajjar, F., Al-Mahmud, A., Jishtu, N., and Alsinglawi, B. (2022). Exploring serious games for stroke rehabilitation: a scoping review. *Disabil Rehabil Assist Technol.*, 17(2):159–165.
- [Newport, 2007] Newport, R. (2007). Ventajas de la rehabilitación asistida mediante robot en la recuperación de las funciones motriz y visuoespacial en pacientes en fase de recuperación de un accidente cerebrovascular. *Revista Española de Geriatria y Gerontología*, 41:66–73.
- [OMS, 2025] OMS (2025). Stroke, cerebrovascular accident. <https://www.emro.who.int/health-topics/stroke-cerebrovascular-accident/index.html> [Accedido: 24/05/2025].
- [Rubino et al., 2024] Rubino, C., Lakhani, B., Larssen, B., Kraeutner, S., Andrushko, J., Borich, M., and Boyd, L. (2024). Gamified practice improves paretic arm motor behaviour in individuals with stroke. *Neurorehabilitation and Neural Repair*, 38:832–844.
- [SAFE, 2030] SAFE (2018-2030). Ictus: Plan de actuación en europa.

[SEN, 2007] SEN (2007). Guía de información al paciente con ictus.

[SER, 2025] SER, C. (2025). La universidad carlos iii crea dos videojuegos para favorecer la rehabilitación de manos y muñecas. <https://cadenaser.com/cmadrid/2025/03/04/la-universidad-carlos-iii-crea-dos-videojuegos-para-favorecer-la-rehabilitacio> [Accedido: 30/05/2025].

[SNS, 2024] SNS (2024). Estrategia en ictus del sistema nacional de salud. *Ministerio de Sanidad*.

[Soriano-Salvador and Guardiola Múzquiz, 2022] Soriano-Salvador, E. and Guardiola Múzquiz, G. (2022). Fundamentos de sistemas operativos: una aproximación práctica usando linux. pages 21–23. Para obtener la versión más reciente de este documento: <https://honecomp.github.io>.

[Stack Overflow,] Stack Overflow, D. Python notes for professionals. page 2. This is an unofficial free book created for educational purposes and is not affiliated with official Python® group(s) or company(s).

[Sue et al., 2024] Sue, T., Wang, M., Chen, Z., and Feng, L. (2024). Effect of Upper Robot-Assisted Training on Upper Limb Motor, Daily Life Activities, and Muscular Tone in Patients With Stroke: A systematic review and meta-analysis. *Rehabilitacion (Madr.)*, 14(11).

[Sánchez-Gil et al., 2025] Sánchez-Gil, J., Sáez-Manzano, A., López-Luque, R., Ochoa-Sepúlveda, J., and E, C.-C. (2025). Gamified devices for stroke rehabilitation: A systematic review. *Comput Methods Programs Biomed.*, 258:108476.

[Volpe et al., 2000] Volpe, B., Krebs, H., Hogan, N., Edelstein, L., Diels, C., and Aisen, M. (2000). A novel approach to stroke rehabilitation:: obot aided sensorymotor stimulation. *Neurology*, 54:1938–44.