Universidad Intercontinental de la Empresa

Robots Autónomos

Equipo: Yago Ramos Sánchez y Alan Ariel Salazar

Create3-autonomy-hub

reate3-autonomy-hub

Participación de los integrantes del equipo

	Integrantes	Participa (SI/NO/%)	
1	Yago Ramos Sánchez	SI	Responsable
2	Alan Ariel Salazar	SI	

Columna Participa

SI → 100% de la calificación.
NO → 0% de la calificación.
% → % de la calificación que se le asigna al participante.

"Cualquier hecho que enfrentes no es tan importante como tu actitud hacia él, ya que eso determina tu éxito o fracaso."
Norman Vincent Peale

Universidad Intercontinental de la Empresa

Contenido

Universidad Intercontinental de la Empresa

1 Resumen general

1.1 Descripción

En este proyecto desarrollamos un sistema completo de navegación autónoma para el robot iRobot Create 3, basado en campos de potencial atractivos y repulsivos. Partimos de las prácticas previas de la asignatura y extendimos la arquitectura para soportar navegación multi-objetivo mediante waypoints secuenciales y navegación nodo-a-nodo sobre un grafo topológico del entorno (GRAPHOS).

La aplicación se estructura en varios módulos: un núcleo de campos de potencial que combina fuerzas atractivas hacia el objetivo con fuerzas repulsivas derivadas de los sensores infrarrojos; un sistema de seguridad que ajusta dinámicamente la velocidad en función del clearance disponible; herramientas visuales para definir rutas y grafos topológicos; y un conjunto de scripts de logging y análisis que permiten evaluar cuantitativamente el comportamiento del robot. Todo el flujo, desde la definición del mapa hasta la ejecución en el robot y el análisis posterior, se ha diseñado para ser reutilizable y fácilmente configurable.

Para revisar configuracion de codigo y/o formulas se recomienda visitar el siguiente repositorio: Github

1.2 Objetivos específicos

En este proyecto nos planteamos los siguientes objetivos específicos:
• Implementar un sistema de navegación autónoma basado en campos de potencial atractivos y repulsivos sobre el iRobot Create 3.
• Incorporar navegación secuencial por waypoints, permitiendo definir rutas complejas a través de zonas de interés.
• Diseñar e integrar un grafo topológico del entorno (GRAPHOS) y calcular rutas óptimas nodo-a-nodo mediante Dijkstra.
• Desarrollar herramientas visuales para la configuración de puntos de navegación y para la edición del grafo topológico.
• Implementar un control de velocidad adaptativo basado en niveles de seguridad y clearance efectivo.
• Registrar y analizar datos de navegación mediante logs para comparar funciones de potencial y ajustar parámetros de forma objetiva.

1.3 Limitaciones

El alcance del proyecto se centra en la navegación en entornos interiores estructurados, con un mapa estático y obstáculos principalmente cuasi-estáticos (muebles, cajas, elementos del laboratorio). No abordamos el problema del mapeado simultáneo (SLAM) ni el seguimiento de obstáculos móviles rápidos. La localización del robot se basa en odometría, por lo que asumimos cierto nivel de deriva acumulada, mitigada con tolerancias adaptativas en la llegada a los objetivos. Además, el sistema trabaja sobre un único robot y no contempla coordinación multi-robot ni integración con mapas métricos externos.

1.4 Unidades de la asignatura

De las unidades que se estudiaron marque las que se involucran en el proyecto.

Unidades	Proyecto

Unidad I Robótica	
Unidad II Locomoción	X
Unidad III Sistemas de Percepción y Modelado del Entorno	X
Unidad IV Navegación de Robots Móviles.	X
Unidad V Robots Sociales	X

2 Entorno

2.1 Espacio físico

Realizamos todas las pruebas en un espacio interior de laboratorio, con forma aproximadamente rectangular y de varios metros de lado, que acondicionamos específicamente para el proyecto. El área de trabajo estaba delimitada por las propias paredes del aula y por el mobiliario disponible (mesas, sillas, estanterías…), de manera que se formaban de forma natural varios pasillos y algunas zonas más abiertas donde el robot podía maniobrar con más libertad. Dentro de ese espacio decidimos marcar claramente un punto de inicio, varias zonas intermedias y una zona de destino, de forma que tuviéramos un recorrido coherente para las misiones de navegación. Además, colocamos obstáculos adicionales en posiciones estratégicas para forzar al robot a desviarse de la trayectoria directa y obligarle a aprovechar realmente el sistema de evitación de obstáculos y la planificación por waypoints o por grafo.

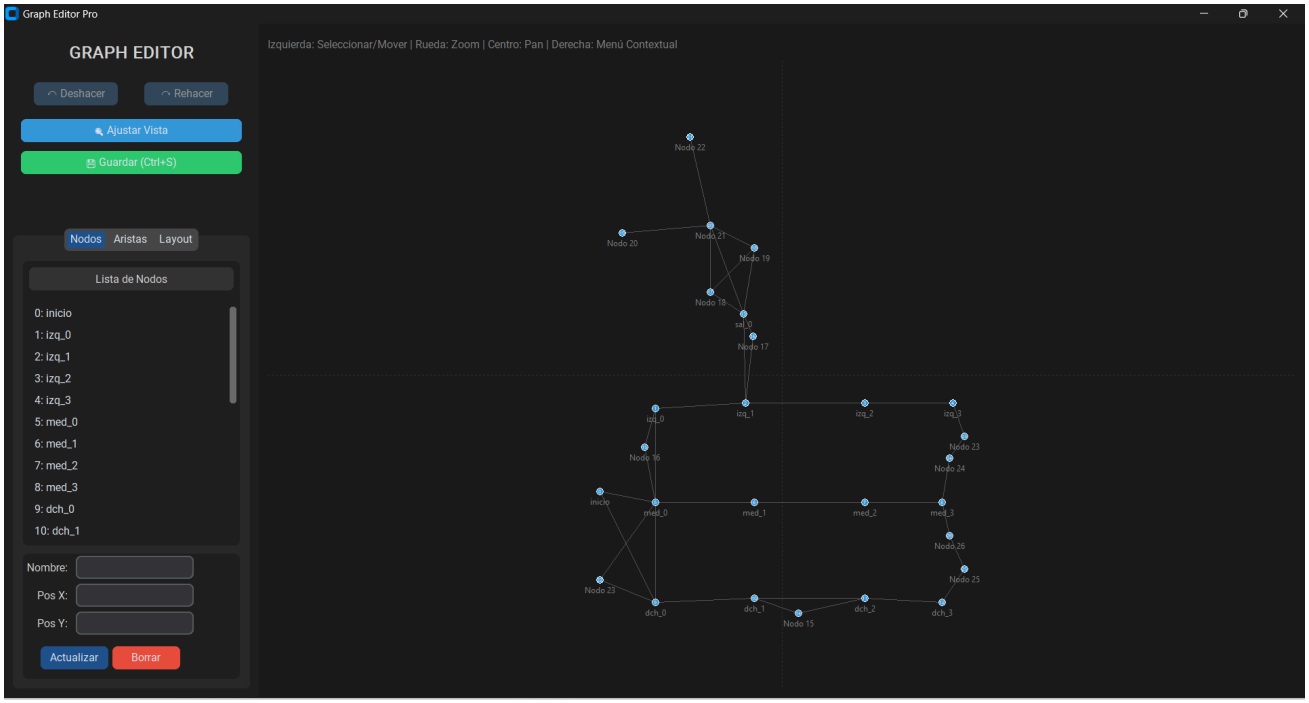
A partir de este entorno físico construimos un mapa aproximado en coordenadas métricas (trabajando en centímetros), que es el que utilizamos como referencia en todo el proyecto. En ese mapa situamos las paredes, las zonas transitables y los obstáculos más relevantes, y sobre él fuimos definiendo tanto los puntos de navegación (punto inicial, waypoints y punto final) como los nodos del grafo topológico que emplea **GRAPHOS**. De esta manera conseguimos que el modelo que maneja la aplicación estuviera alineado con la realidad del laboratorio: cuando hablamos de un nodo, de un waypoint o de una arista en el grafo, siempre sabemos a qué zona física corresponde dentro del espacio de pruebas.

Se

2.2 Modelado del entorno

Para modelar el entorno seguimos una aproximación híbrida métrica-topológica, porque queríamos aprovechar las ventajas de ambos enfoques. Por un lado, a nivel métrico trabajamos con un sistema de coordenadas plano donde describimos cada posición mediante x, y y un ángulo de orientación theta. Las distancias las expresamos en centímetros y los ángulos en grados. Este mismo sistema lo utilizamos tanto para definir los puntos de navegación en el fichero **points.json** como para interpretar la odometría del robot, de forma que todo esté coordinado. Gracias a esto podemos situar de manera coherente el punto inicial, los waypoints y el objetivo final sobre el mapa, y comparar directamente la posición estimada del robot con las posiciones de referencia que hemos definido.

Por otro lado, a nivel topológico describimos el entorno como un grafo dirigido y ponderado, en el que cada nodo representa una zona navegable del laboratorio y cada arista representa un posible paso de una zona a otra, con un coste asociado (normalmente relacionado con la distancia o la dificultad del trayecto). Esta parte la gestionamos con **GRAPHOS**, que nos ofrece un editor visual para crear y modificar el grafo, guardar toda la información en formato JSON y, a partir de un nodo de origen y otro de destino, calcular el camino óptimo entre ambos. Una vez tenemos ese camino en forma de secuencia de nodos, el propio sistema lo traduce automáticamente en una secuencia de punto inicial, waypoints intermedios y punto final que reutilizamos en el resto de la navegación basada en campos de potencial, sin tener que redefinir las rutas a mano cada vez.



3 Diseño de la aplicación

En nuestro caso organizamos la aplicación pensando desde el principio en que fuese fácil de entender, de mantener y de ampliar. Por eso optamos por una arquitectura modular, donde cada parte del sistema tiene una responsabilidad clara y bien definida. La capa más externa está formada por los programas que utilizamos para lanzar las distintas modalidades de navegación (solo potencial atractivo, potencial combinado con evitación de obstáculos y navegación apoyada en grafo topológico). Desde el punto de vista del usuario, basta con elegir qué modo quiere probar y con qué parámetros; a partir de ahí, el programa correspondiente se encarga de inicializar el robot, cargar la configuración y conectarse con el resto de módulos internos sin que haya que cambiar código cada vez que queremos hacer una prueba diferente.

Por debajo de esa capa de entrada concentramos la lógica común en varios bloques bien separados. Contamos con un bloque de configuración donde centralizamos todas las constantes y parámetros del sistema (ganancias de los campos de potencial, límites de velocidad, umbrales de sensores, anchura mínima de los gaps navegables, tolerancias de llegada al objetivo, etc.), de forma que cualquier ajuste fino se puede hacer desde un único sitio. Sobre esa base trabaja el bloque de campos de potencial, encargado de calcular las fuerzas atractivas hacia el objetivo, las fuerzas repulsivas debidas a los obstáculos detectados por los sensores infrarrojos, la combinación vectorial de ambas y la detección de gaps. Otro bloque se encarga específicamente de la seguridad y el control de velocidad: aplica las restricciones de aceleración y frenado, decide el nivel de seguridad en función del clearance disponible y limita la velocidad máxima permitida en cada instante. Finalmente, disponemos de bloques dedicados al registro de información, que guardan en tiempo real los datos de sensores, la posición estimada del robot, las velocidades calculadas y otras variables relevantes para poder analizarlas con calma después de cada ejecución.

Además de estos componentes internos, desarrollamos también herramientas de apoyo que nos facilitan mucho la preparación de las pruebas. Por un lado, tenemos una interfaz visual para configurar rutas mediante puntos: sobre un mapa del entorno podemos marcar el punto inicial, los waypoints intermedios y el punto final con simples clics, ver la trayectoria resultante y ajustar la orientación inicial de forma intuitiva. Por otro lado, disponemos de un editor gráfico del grafo topológico (GRAPHOS) que nos permite crear nodos, moverlos sobre un plano, conectar zonas mediante aristas, asignar pesos y guardar el resultado de forma que el resto del sistema pueda utilizarlo directamente para calcular rutas óptimas entre zonas. Toda la comunicación con el robot se realiza a través de la API oficial del iRobot Create 3, que integramos de manera que para nosotros el trabajo diario se reduce a pensar en términos de objetivos, fuerzas, velocidades lineales y angulares, mientras que los detalles de bajo nivel (envío de órdenes a las ruedas, lectura periódica de sensores, etc.) quedan encapsulados dentro de la propia aplicación.

4 Funcionalidades de la aplicación

Las principales funcionalidades de nuestra aplicación se pueden resumir en varios bloques, pero todos están conectados entre sí y pensados para que el robot pueda navegar de forma autónoma, segura y analizable. A continuación describimos cada una de ellas con más detalle:

- **Navegación hacia un único objetivo utilizando distintos campos de potencial atractivos (lineal, cuadrático, cónico y exponencial).**
  En la versión más básica del sistema podemos definir un único objetivo y dejar que el robot se dirija hacia él usando diferentes modelos de campo atractivo. Nosotros podemos elegir el tipo de potencial en función del comportamiento que queramos: por ejemplo, uno más suave cerca de la meta, otro que acelere más cuando el robot está lejos, o uno que se sature a cierta distancia. Esto nos permite comparar de manera directa cómo cambia la trayectoria y la velocidad simplemente cambiando el modelo matemático, sin tocar el resto de la lógica de navegación.

- **Navegación secuencial por waypoints, visitando en orden todos los puntos definidos entre el punto inicial y el punto final, con confirmaciones sonoras y mensajes de progreso.**
  Más allá del caso de un solo objetivo, el sistema nos permite definir una ruta completa formada por varios puntos intermedios. El robot los va visitando en orden, comprobando en cada momento si ya está dentro de la tolerancia de llegada para pasar al siguiente. Cada vez que alcanza un waypoint se emite un sonido característico y se muestra información de estado, de forma que podemos seguir fácilmente el progreso de la misión sin tener que adivinar en qué parte de la ruta se encuentra.

- **Navegación nodo-a-nodo sobre un grafo topológico, calculando automáticamente el camino óptimo entre nodos de origen y destino y generando la secuencia equivalente de puntos de navegación.**
  Cuando trabajamos con el grafo topológico, ya no pensamos solo en coordenadas sueltas, sino en zonas conectadas entre sí. Podemos seleccionar un nodo de origen y uno de destino y dejar que el sistema calcule el camino más conveniente teniendo en cuenta los pesos de las conexiones. A partir de ese camino se genera automáticamente la secuencia de puntos que el robot debe seguir, lo que

nos permite planificar recorridos más complejos de forma mucho más cómoda y estructurada.

- **Evitación de obstáculos basada en campos de potencial repulsivos, construidos a partir de las lecturas normalizadas de los sensores infrarrojos del robot.**
  Mientras el robot se dirige a sus objetivos, el sistema está continuamente leyendo los sensores y detectando obstáculos en su entorno cercano. En función de la distancia estimada y de la dirección de cada obstáculo, se genera una fuerza repulsiva que se suma a la fuerza atractiva hacia el objetivo. De esta forma, el robot no sigue una línea recta "ciega", sino que ajusta su trayectoria para rodear obstáculos y mantener una distancia de seguridad razonable sin necesidad de un mapa dinámico complejo.

- **Control de velocidad adaptativo con varios niveles de seguridad en función del espacio libre disponible y de la distancia de frenado estimada.**
  La velocidad del robot no es fija, sino que se ajusta continuamente en función de lo cerca que esté de los obstáculos y de la capacidad real de frenar a tiempo. El sistema calcula una distancia de frenado aproximada a partir de la velocidad actual y la compara con el espacio libre delante del robot. Según esa combinación, se activa un nivel de seguridad (emergencia, crítico, advertencia, precaución o libre) que limita la velocidad máxima permitida en cada momento. Así conseguimos que el robot vaya rápido cuando el entorno es seguro y reduzca velocidad de forma proactiva cuando la situación se vuelve más comprometida.

- **Detección de gaps navegables entre obstáculos y reducción selectiva de fuerzas repulsivas para permitir el paso por pasillos estrechos.**
  Una funcionalidad importante es que el sistema no solo ve obstáculos de forma aislada, sino que también es capaz de detectar "huecos" entre ellos. Cuando identifica un espacio suficientemente ancho entre dos obstáculos, lo marca como gap navegable y reduce de manera controlada la fuerza repulsiva de los bordes que lo forman. Gracias a esto, el robot no se queda bloqueado ante dos objetos cercanos, sino que se anima a pasar por el hueco si hay espacio suficiente, lo que le permite atravesar pasillos y zonas estrechas que serían difíciles de gestionar con una repulsión más simple.

- **Mecanismos de escape de trampas en C, ajustando temporalmente las ganancias para favorecer la exploración y la salida de mínimos locales.**
  En algunos escenarios, el robot puede quedar rodeado por obstáculos en forma de "C" y tender a quedarse oscilando o atrapado en un mínimo local del campo de potencial. Para estos casos incorporamos un modo especial de escape: cuando detectamos que hay muchos sensores ocupados y no se aprecia un gap claro, el sistema modifica temporalmente las ganancias de las fuerzas atractivas y repulsivas, aumenta la capacidad de giro y fuerza al robot a explorar alternativas. Con este mecanismo buscamos que el robot encuentre una salida razonable en situaciones que, de otro modo, podrían bloquear la navegación.

- **Registro de datos de navegación y herramientas de análisis que generan dashboards con métricas, trayectorias y fuerzas, facilitando la comparación entre distintos experimentos.**
  Por último, cada ejecución del sistema queda registrada en ficheros de log donde almacenamos posición, velocidades, lecturas de sensores, fuerzas calculadas y otros datos relevantes. A partir de esos registros hemos desarrollado herramientas de análisis que generan gráficas y paneles de control

con la trayectoria seguida, la evolución de la distancia al objetivo, los niveles de seguridad activados, la magnitud de las fuerzas repulsivas, etc. Esto nos permite comparar de forma objetiva diferentes configuraciones, tipos de potencial o parámetros, y tomar decisiones basadas en datos en lugar de quedarnos solo con una impresión visual de cómo se movía el robot.

5 Uso de la aplicación

El flujo de uso típico de nuestra aplicación empieza siempre por la definición de la ruta de navegación. Para ello tenemos varias opciones, según lo que queramos probar. Por un lado, podemos configurar la ruta de forma manual editando el fichero **data/points.json**, donde indicamos el punto inicial, los waypoints intermedios y el punto final. Por otro lado, cuando queremos algo más visual y cómodo utilizamos el configurador de puntos **visual_point_config.py**, que nos permite marcar directamente sobre un mapa el inicio, los puntos intermedios y el destino con el ratón y ver al momento cómo queda la trayectoria. En el modo de navegación topológica el flujo es algo distinto: ahí trabajamos con el editor de grafo **grafos/visualize.py**, donde definimos nodos, aristas y pesos, y después simplemente elegimos un nodo de origen y uno de destino para que el sistema calcule el camino óptimo. Una vez tenemos definidos los puntos o el grafo, lanzamos el programa que nos interese (por ejemplo, **PRM02_P01.py** para probar navegación solo con potencial atractivo, o **PRM02_P02_EQUIPO1.py** para navegación con potencial combinado y waypoints) y seleccionamos el tipo de potencial, las ganancias y el resto de parámetros a través de argumentos en la línea de comandos.

A partir de ese momento, durante la ejecución, la aplicación se encarga de casi todo de forma automática. El sistema va leyendo continuamente los sensores del robot, calculando las fuerzas atractivas hacia el siguiente objetivo y las fuerzas repulsivas debidas a los obstáculos, y combinando ambas para obtener una dirección de avance segura. En función del nivel de seguridad que corresponda en cada instante, ajusta la velocidad permitida y envía las velocidades de las ruedas al robot de forma transparente para el usuario. Mientras tanto, nosotros podemos seguir el progreso de la misión gracias a los mensajes que se muestran en la consola (por ejemplo, cuando se alcanza un waypoint o se pasa al siguiente) y a las señales acústicas que marcan hitos importantes, como la llegada a cada punto intermedio o la finalización completa de la ruta. Al mismo tiempo, el sistema va registrando en segundo plano datos detallados de la navegación: trayectoria, velocidades lineales y angulares, fuerzas calculadas, número de obstáculos detectados, niveles de seguridad activados, etc., de modo que cada prueba queda documentada para poder analizarla después con calma.

En cuanto a las dificultades, una de las que más hemos notado es la deriva de la odometría en recorridos largos. Esto nos obligó a introducir tolerancias adaptativas a la hora de decidir si el robot “ha llegado” a un waypoint o al objetivo final, de forma que el sistema fuese robusto frente a pequeños errores acumulados de posición. Otra parte delicada ha sido la calibración del modelo que convierte lecturas infrarrojas en distancias y el ajuste de las ganancias repulsivas: si las fuerzas son demasiado bajas, el robot se acerca demasiado a los obstáculos; si son demasiado altas, se vuelve excesivamente conservador y la trayectoria se alarga mucho. Como caso de uso representativo, tomamos una ejecución completa en la que el robot recorre una ruta con varios waypoints utilizando un potencial atractivo cuadrático combinado con el campo repulsivo. En esa prueba el robot consigue evitar obstáculos, atravesar gaps navegables y llegar al objetivo final con un error de posición de pocos centímetros y un error angular muy pequeño, lo que nos sirvió como validación global del comportamiento del sistema.

6 Conclusiones

A lo largo de este proyecto diseñamos, implementamos e integramos un sistema de navegación autónoma para el iRobot Create 3 basado en campos de potencial, capaz de operar tanto con un único objetivo como con rutas compuestas por múltiples waypoints e incluso con planificación sobre un grafo topológico del entorno. El sistema combina de forma efectiva fuerzas atractivas hacia el objetivo y fuerzas repulsivas derivadas de los sensores infrarrojos, apoyadas por un control de velocidad adaptativo que prioriza la seguridad sin renunciar a la eficiencia cuando el entorno lo permite.

Los experimentos realizados, analizados mediante los logs y los dashboards generados, muestran que el robot es capaz de evitar obstáculos, atravesar gaps navegables y alcanzar el objetivo final con errores de posición inferiores a las tolerancias fijadas y con una orientación final muy precisa. La arquitectura modular y la centralización de parámetros en `config.py` facilitan la reutilización del código y la realización de nuevas pruebas variando potenciales y ganancias.

Como posibles líneas de trabajo futuro, consideramos la integración con mapas métricos generados mediante SLAM, la adaptación online de parámetros en función del entorno y la comparación sistemática con otros enfoques de planificación y navegación, como planners globales clásicos o técnicas de aprendizaje por refuerzo.

7 Resumen de aportes individuales.

A solicitud del responsable del equipo cada integrante del equipo le enviará un email enumerando sus **aportes específicos** al proyecto.
El responsable del equipo copiará desde el email dichos aportes para cada integrante.

Integrante: Yago Ramos Sánchez (Responsable del Equipo)
Aportes:
• Diseño del mapa del entorno de trabajo y del sistema de mapeo en coordenadas métricas.
• Definición y ajuste del modelado topológico del entorno mediante nodos y aristas en GRAPHOS.
• Configuración y validación de rutas mediante waypoints sobre el mapa.
• Participación en la planificación de las pruebas y en el análisis de los resultados obtenidos a partir de los logs.

Integrante: Alan Ariel Salazar
Aportes:
• Implementación de la API de comunicación con el iRobot Create 3 desde los scripts principales.
• Desarrollo e integración de las funciones de potencial atractivo y repulsivo, así como del control de velocidad adaptativo.
• Implementación de los módulos de registro de datos y de las herramientas de análisis y visualización de logs.
• Participación en la ejecución de pruebas, ajuste de parámetros y validación del comportamiento global del sistema.

8 Entrega

Se entrega un fichero pdf con el informe final y un fichero comprimido en zip con los programas de la aplicación.

- Informe: El equipo deberá subir el informe al campus virtual en formato pdf,
- Fichero comprimido (zip): El equipo deberá subir un fichero comprimido con los programas desarrollados.

Fecha límite de entrega

08/12/2025 a las 19:00 horas

9 Calificación

Rubrica de calificación

INFORME					Aplicación	Dificultad	
Deficiente (2 pts.)	Suficiente (5 pts.)	Completo (8 pts.)	Muy Completo (10 pts.)	Calificación Informe (10%)	PRODUCTO (90%)	GRADO DIFICULTAD (0,7-1,2)	Calificación Final
				0			0,0

Nota: La no entrega de la participación de los integrantes y/o el resumen de aportes individuales tendrá una penalización del 50% sobre la calificación total de la entrega.

10 Auto calificación

El equipo deberá realizar un análisis autocrítico del proyecto y calificar el trabajo desde la visión de un gerente que ha contratado el producto desde un cargo de responsabilidad en una empresa.

Calificación Global del Proyecto
Creo que nuestro proyecto merece una auto calificación de 10/10, principalmente por dos aspectos clave: cómo hemos implementado la API de OpenAI y cómo hemos diseñado nuestro sistema de mapeo con puntos y aristas. En la parte de la API, no nos limitamos a hacer una llamada básica, sino que la integramos de forma coherente dentro del flujo de trabajo del proyecto, utilizándola como apoyo para documentar, analizar resultados y estructurar mejor el desarrollo. Esto nos ha permitido trabajar de una forma más cercana a un entorno profesional, apoyándonos en herramientas avanzadas para mejorar la calidad del informe, la claridad de las explicaciones y la organización general del proyecto. Por otro lado, el sistema de mapeo que construimos a base de puntos y aristas no es simplemente un “dibujo” del entorno, sino una representación topológica

pensada para que el robot pueda planificar y razonar rutas de forma estructurada. Definimos nodos, conexiones y recorridos posibles, y lo hicimos de manera que fuese editable, reutilizable y fácil de entender. Entre la integración cuidada de la API de OpenAI y el diseño de nuestro propio sistema de mapeo, considero que el trabajo tiene un nivel de acabado y de intención técnica que justifica una auto calificación de 10/10.