UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Grado en Ingeniería del Software

Aplicación de Modelos de Lenguaje a Gran Escala para la Identificación y Planificación de Tareas en un Robot Móvil

Application of Large Language Models for Task Identification and Planning in a Mobile Robot

Realizado por
Antonio Blas Moral Sánchez

Tutorizado por
José Raúl Ruiz Sarmiento
Francisco Ángel Moreno Dueñas

Departamento
Ingeniería de Sistemas y Automática

MÁLAGA, julio 2025

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN INGENIERÍA DEL SOFTWARE

Diseño de Sistemas Funcionales y Reactivos

Functional and Reactive Systems design

Realizado por
Santiago Sánchez Fernández

Tutorizado por
José Enrique Gallardo Ruíz

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2020

Fecha defensa: julio de 2020

Abstract

In these days, meanings like *social robotics* [1] get more significance, due to the interaction of humans with robots is more frequent in distinct fields of society, such as a support call from a company or receiving food from a restaurant. Besides social robotics, another field that has evolved exponentially is the *Large Language Models* (LLM), which in a few years, were capable of generating answers that make sense, but fail sometimes, and now they are capable of generate complex answers, reasoning like a human, and generating realistic pictures, videos and music.

This project wants to contribute to social robotics by combining this field with LLMs, obtaining a more human robot interaction, adding a better reasoning and better experience for the user. Specifically, the project will be applied in a mobile robot, focusing firstly in the identification and planification of tasks, which will be identified by one or more orders given by an human agent in oral or textual way. The planification of tasks execution, will be scheduled using the *behaviour trees* [3] logic.

Keywords: Social Robotics, Artificial Intelligence, LLM, Mobile Robot, Behaviour Tree

Resumen

A día de hoy, conceptos como la *robótica social* [1] van tomando más importancia, debido a que la interacción de los humanos con los robots es más frecuente en distintos ámbitos de la sociedad, ya sea una llamada de soporte de cualquier empresa o para recibir comida de un restaurante. Aparte de la robótica social, otro campo que también ha evolucionado, aunque este ha sido a nivel exponencial, son los *Modelos de Lenguaje a Gran Escala* (LLM), que han pasado en apenas unos años de generar respuestas con sentido, pero con algunos fallos de razonamiento, a poder generar respuestas complejas, tal y como las razonaría un ser humano, y no tan solo respuestas textuales, sino que tienen la capacidad de realizar imágenes realistas, vídeos y música.

Este Trabajo de Fin de Grado pretende contribuir a la robótica social, mediante la combinación de dicho campo con las LLM, obteniendo una interacción, por parte del robot, más humana y con un razonamiento que mejore la experiencia. En concreto, el trabajo se aplicará sobre un robot móvil, enfocándose principalmente en la identificación y planificación de tareas, las cuales se identificarán mediante una o varias ordenes dadas por un agente humano de forma oral o textual. En cuanto a la planificación de la ejecución de las tareas, se planificará utilizando la lógica de los *árboles de comportamiento* [3].

Palabras clave: Robótica Social, Inteligencia Artificial, LLM, Robot Móvil, Árboles de comportamiento

Índice

1 Introducción

1.1. Motivación

La motivación principal para realizar este proyecto es el potencial que ofrece mezclar dos campos que últimamente están creciendo exponencialmente: la **robótica social** y la **inteligencia artificial**. Gracias a juntar ambos conceptos, conseguimos que el robot sea capaz de interactuar de forma más humana, respondiendo frases más allá de un conjunto de respuestas programadas, y comprendiendo de manera más inteligente qué es lo que le está queriendo transmitir la persona. Además de lo mencionado, utilizaré también una estructura que se llama **árboles de comportamiento**, utilizado principalmente en robots o en PNJ (Personaje No Jugador) en los videojuegos, para planificar de forma inteligente las acciones en el trascurso de su realización.

1.2. Objetivos

El objetivo principal de este TFG es el desarrollo de un sistema capaz de transformar una orden dada en lenguaje natural por el usuario, a un conjunto de tareas que el robot pueda realizar. La ejecución y planificación de estas tareas se realizará mediante un árbol de comportamiento, el cual decide el orden de las tareas y se encarga del control de errores de las mismas. A continuación, listaré de forma breve otros objetivos más específicos.

- Aprendizaje de características del LLM y obtención de resultados para utilizar el LLM óptimo y más potente
- Aprendizaje de las estructuras de información compatibles con los árboles de comportamiento (Behavior Tree)
- Implementación del puente de comunicación entre el LLM y el *framework* ROS2

- Desarrollo de integración de mapas semánticos para ampliar el conocimiento del modelo sobre el entorno actual en el que se encuentra el robot
- Desarrollo de una interfaz de monitorización de las tareas, desde la cual podamos enviar la sentencia y seguir el estado de las tareas.

1.3. Estructura del documento

A continuación, se exponen las distintas secciones en las cuales está dividido este documento:

1. **Introducción**: En esta sección se describe el contexto y conceptos necesarios, seguido de las motivaciones de realización de este TFG y los objetivos y resultados de este trabajo. Finalmente, se describen de manera breve las tecnologías utilizadas en este TFG, tanto conceptuales como técnicas.
2. **Estado del arte**: En esta sección se abordan el estado del arte de los campos de estudio más relevantes para este trabajo, que en este caso se tratan de la **robótica social**, **modelos de lenguaje a gran escala** y **árboles de comportamiento**. Para cada campo existe una descripción del mismo, y además una explicación del estado actual del mismo.
3. **Diseño del sistema**: En esta sección se describe el diseño del sistema y se muestran los diagramas realizados durante el proceso. Además, comentaremos los distintos patrones de diseño utilizados en el desarrollo del sistema y se explicará la metodología de trabajo utilizada en el trabajo.
4. **Desarrollo del sistema**: En esta sección se muestra y se describe todo el desarrollo del resultado final, profundizando sobre cómo está hecha la herramienta y sobre algunas interfaces gráficas que se han desarrollado con el objetivo de visualizar los procesos.
5. **Análisis y pruebas**: En la sección de análisis y pruebas se exponen los análisis experimentales que se ha realizado para la toma de decisiones del proyecto.
6. **Conclusiones y líneas futuras**: Para finalizar, en esta sección se incluye la conclusión personal tras realizar el trabajo y las posibles líneas futuras con las que se podría continuar el proyecto.

1.4. Tecnologías utilizadas

En esta sección, se describirán las tencologías utilizadas para la realización del proyecto, incluyendo tanto tecnologías específicas como técnicas o estándares:

1.4.1. ROS2

ROS2 es un *framework* diseñado para el desarrollo de software para robots, el cual proporciona servicios como control de hardware, abstracción, gestión de dispositivos y comunicación entre procesos (o nodos) y paquetes del sistema. Aunque pudiera parecerlo, ROS2 no es un sistema operativo, sino que es un conjunto de herramientas y bibliotecas cuya finalidad principal consiste en facilitar el desarrollo de aplicaciones robóticas complejas, y su escalabilidad.

1.4.2. Python

Python es un lenguaje de programación multiparadigma y multiplataforma. Este lenguaje es muy simple y fácil de aprender, ya que su sintaxis es simple y similar al inglés, lo cual facilita su aprendizaje, y además tiene un tipado dinámico que facilita la asignación de variables. Además de lo mencionado, Python cuenta con un extenso repositorio de librerías, lo cual hace que utilizandolo puedas realizar aplicaciones de distintos tipos (análisis de datos, inteligencia artificial o el manejo de ROS2).

1.4.3. Qt

Qt es un *framework* de desarollo de interfaces gráficas multiplataforma el cúal se puede utilizar tanto con el lenguage de programación **Python** (mencionado anteriormente) o con el lenguaje de programación **C++**.

1.4.4. Computación en la nube

Debido a la gran carga computacional que supone un LLM, el propio robot no podría cargar con un servidor lo suficientemente potente para poder tener un modelo local (debido a limitaciones de potencia eléctrica y peso), por lo que se utiliza un computador externo potente (Ultra Edge) que recibe la información del robot mediante los ya mencionados Servicios de ROS2, y tras realizar el cómputo necesario, devuelve la respuesta al robot.

Gracias a esta tecnología, conseguimos generar y transmitir la información necesaria para el robot sin necesidad de afectar al rendimiento del equipo móvil del robot.

1.4.5. Py Trees

Py Trees es una librería para el lenguaje de programación Python la cual es una implementación de los árboles de comportamiento en *Python*. Esta librería incluye tanto clases para definir comportamientos dentro de un árbol, como la implementación de la lógica de un árbol de comportamiento y su ejecución.

2

Estado del arte

2.1. Robótica social

La **robótica social** es una rama de la robótica que se centra en el diseño y desarrollo de robots que pueden interactuar con los humanos de manera efectiva y natural (véase la figura 1). Estos robots están diseñados para trabajar en entornos sociales, como hogares, hospitales, escuelas y espacios públicos, y su objetivo es facilitar la interacción humano-robot y mejorar la calidad de vida de las personas. Actualmente, la robótica social está en auge debido a los avances en inteligencia artificial, aprendizaje automático y tecnologías de percepción, lo que ha permitido el desarrollo de robots más sofisticados y capaces de comprender y responder a las necesidades humanas.

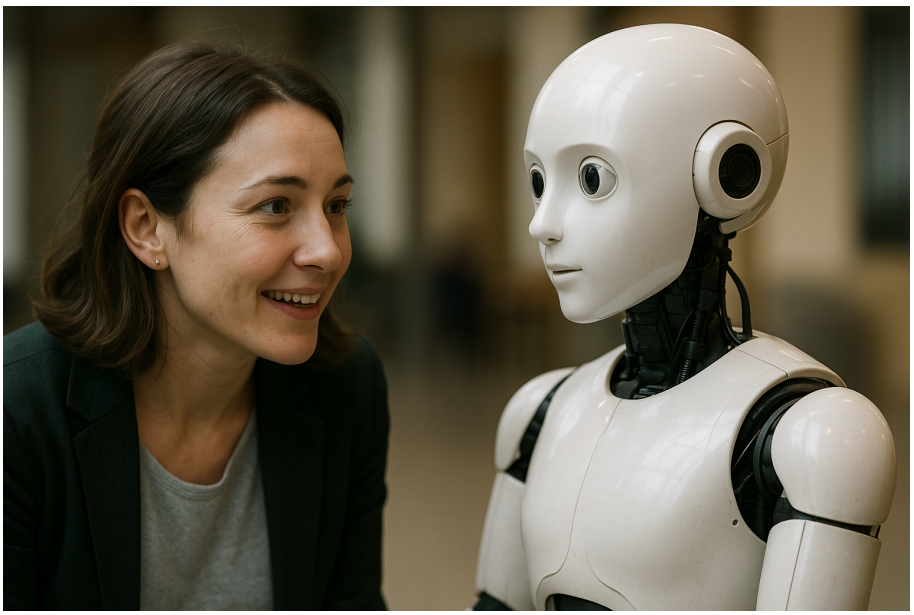

Figura 1: Ejemplo de un robot social interactuando con un humano

Algunos ejemplos de aplicaciones de la robótica social incluyen:

- **Asistentes personales**: Robots diseñados para ayudar a las personas en tareas diarias, como recordatorios, gestión de calendarios y asistencia en el hogar.
- **Educación**: Robots que pueden interactuar con estudiantes y profesores en entornos educativos, facilitando el aprendizaje y la enseñanza.
- **Salud**: Robots que pueden asistir a profesionales de la salud en hospitales y clínicas, proporcionando apoyo a pacientes y personal médico.
- **Entretenimiento**: Robots diseñados para interactuar con personas en entornos de ocio, como museos, parques temáticos y eventos públicos.
- **Compañía**: Robots que pueden proporcionar compañía y apoyo emocional a personas mayores[2] o solas, ayudando a combatir la soledad y el aislamiento social.

En el caso de este trabajo, el robot se podría categorizar como un asistente personal, ya que su objetivo consiste en ayudar a los usuarios mediante la realización de distintas tareas.

2.2. Modelos de Lenguaje a Gran Escala (LLM)

Los **modelos de lenguaje a gran escala** son una tecnología que se desarrolló y se popularizó hace pocos años, ganando mucha relevancia casi de forma inmediata por su gran potencial. Los LLMs comenzaron teniendo la funcionalidad principal de, a partir de unas instrucciones o un *prompt* en lenguaje natural, generar una respuesta textual razonada y con sentido, pudiendo desde obtener información de un tema concreto, hasta mantener una conversación relacionada con un tema en concreto.

Actualmente, estos modelos se han ido desarrollando y mejorando de manera exponencial, pudiendo no sólo generar texto, sino video e imágenes realistas, y música de cualquier género, todo siguiendo el *prompt* introducido.

Hoy en día, muchas empresas han decidido entrenar y desarrollar sus propios modelos, existiendo una gran variedad de opciones a elegir. Cada empresa crea modelos más robustos o más rápidos, los cuales están diseñados para dar una mejor respuesta, ser rápidos o poder ejecutarse en una máquina de menor gama. Algunos modelos son *open source* y puedes ejecutarlos en tu propia máquina, en cambio, otros modelos son accesibles mediantes APIs (gratis o de pago).

- **ChatGPT**: Este modelo fue el primer modelo de LLM y el más famoso hasta la fecha, es un modelo de pago para el uso desde APIs desarrollado por la empresa OpenAI.
- **Gemini**: Este modelo tiene versiones gratuitas y de pago para su acceso desde APIs, su desarrollador es Google.
- **Llama**: Este modelo se puede ejecutar desde una máquina propia con un permiso que se le pide a la empresa, en este caso Meta.
- **DeepSeek**: Este modelo es *open source*, es decir, puedes ejecutarla desde un sistema propio sin necesidad de licencia o permiso alguno, y además tiene una API para su acceso que también es gratuita.

Cada modelo ha sido entrenado con un número de parámetros, que usualmente define su robustez y rapidez a la hora de la ejecución, pero además de esos parámetros, se pueden configurar otros parámetros como la *temperatura* o el *top_k* que definen la aleatoriedad (o creatividad) con la que el modelo va a responder a la instrucción. Más creatividad implica una respuesta menos determinista, por tanto más creativa y similar a una respuesta humana, aunque si el nivel es demasiado alto, puede ser demasiado aleatorio y comenzar a responder mensajes sin sentido o muy alejados de las instrucciones dadas.

2.3. Árboles de comportamiento

Los **árboles de comportamiento** son una técnica diseñada para tomar decisiones en tiempo real sobre un actor y un escenario concreto. Esta técnica permite que un actor con ciertas acciones pueda tener cierta lógica e inteligencia sobre las acciones que realiza, sobretodo teniendo en cuenta que las decisiones que se toman varían dependiendo de las circunstancias que puedan ocurrir durante la ejecución. Esta técnica se utiliza en un gran número de ámbitos, algunos más destacables pueden ser los sistemas de control del robot[3] o la toma de decisiones de un Personaje No Jugador dentro del ámbito de los videojuegos. En ambos casos, consigue añadir una inteligencia y un razonamiento sobre las decisiones que se toman, simulando los robots un comportamiento más humano.

A continuación explicaremos el funcionamiento de esta técnica. Los árboles de comportamiento se componen de comportamientos, refiriéndose a los nodos del árbol de decisión.

Estos comportamientos pueden ser una acción a ejectuar, o un comportamiento compuesto, los cuales comentaremos más adelante. Estos comportamientos se ejecutan cada vez que se produce un *tick*, que podemos definir como una iteración por parte del nodo raíz sobre las acciones dentro del árbol.

Cada comportamiento debe devolver un *feedback* en cada tick, destacando entre ellos los estados de **succesful**, **failure** y **running**. Su propio nombre indica el estado del comportamiento, por ejemplo, **succesful** indica que el comportamiento ha terminado correctamente, **failure** que ha terminado con fallos y **running** que aún no ha terminado de ejecutarse. Gracias a estos estados, podemos llevar un control del árbol en tiempo de ejecución. Una vez explicado el contexto de los estados de los comportamientos, expondremos los distintos tipos de comportamientos y algunos ejemplos:

- **Comportamiento compuesto**: Se refiere a un comportamiento donde se ejecutan varias acciones. Estos comportamientos siguen una lógica para ejecutar las acciones hijas, por ejemplo, un comportamiento que debe ejecutar todas sus hijos, y si falla algún hijo falla también el comportamiento (*Comportamiento en secuencia*).
- **Comportamiento simple**: Se refiere a un comportamiento que ejecuta una acción o decisión. Este comportamiento se puede personalizar para las decisiones o acciones que se deseen tomar, proporcionando además el estado mencionado anteriormente, que sirve de feedback del sistema. Un ejemplo de este comportamiento puede ser mover el robot a algún sitio.

Investigando acerca de los recursos que existen sobre los árboles de comportamiento, se hallan distintas librerías que permiten utilizar esta técnica de forma simple, por ejemplo **Py Trees** para implementar estos árboles en el lenguaje *Python* y **BehaviorTree.CPP** para implementarlos en el lenguaje *C++*.

3

Diseño del sistema

En esta sección abordaremos el diseño que se ha llevado a cabo para la implementación del sistema, visualizando el sistema y sus interacciones con diagramas de distintas clases, como diagramas de clases, de arquitectura y de secuencia.

3.1. Metodología de trabajo

3.2. Arquitectura del sistema

A continuación, describiremos la arquitectura del sistema, mostrando los distintos componentes que lo componen y cómo se comunican entre sí. Para ello, utilizaremos distintos diagramas que representarán la organización de los componentes y como se comunican entre ellos.

Comenzaremos con un diagrama de la arquitectura del sistema (véase la figura 2), mostrando los distintos componentes y a en qué contexto se ejecutan, ya que en este caso, podemos diferenciar dos dispositivos: el robot físico y el ordenador dedicado al cómputo de los modelos.

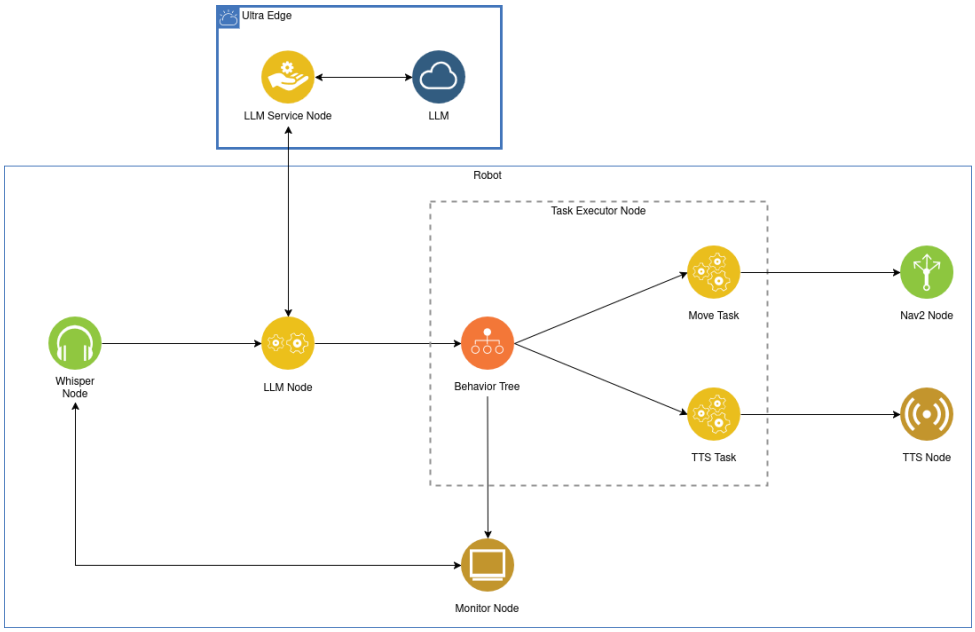

Figura 2: Diagrama de arquitectura del sistema.

Más adelante, en la sección de desarrollo del sistema, se explicará la implementación y la utilización de cada uno de los componentes mostrado más en profundidad.

3.3. Componentes del sistema

En esta sección describiremos las distintas clases y componentes del sistema.

3.3.1. Patrones de diseño

En esta subsección describiremos principlamente los patrones de diseño que se han utilizado en el desarrollo del sistema, basándonos en los componentes que aparecen en los diagramas expuestos anteriormente.

- **Fábrica (o *Factory*)**: Este patrón se utiliza para crear objetos sin especificar la clase exacta del objeto que se va a crear. En el sistema, se ha utilizado para varios componentes, como por ejemplo, la clase `TaskFactory`, que permite crear tareas en base a un conjunto dado de implementaciones de tareas, o en el caso de la clase `LLMFactory`, que permite crear instancias de modelos de la misma forma que `TaskFactory`.

- **Estrategia**: Este patrón permite definir una familia de algoritmos, encapsularlos y hacerlos intercambiables. En el sistema, se ha utilizado para la clase `Task`, que define un

conjunto de tareas que pueden ser ejecutadas por el sistema, y la clase `LLM`, que define un conjunto de modelos de lenguaje que pueden ser utilizados por el sistema.

- **Observador**: Este patrón define una relación de dependencia uno a muchos entre objetos, de tal forma que cuando un objeto cambia de estado, todos sus dependientes son notificados y actualizados automáticamente. Este patrón se ha utilizado en el sistema bajo la implementación dada del framework **ROS2**, donde cada nodo puede publicar y suscribirse a *topics*, los cuales permiten enviar y recibir información entre muchos nodos de forma sencilla.

3.4. Comunicación del sistema

Para finalizar la sección de diseño, describiremos las comunicaciones que se realizan en el sistema, tanto entre los distintos nodos de ROS2 como entre el sistema y el LLM. Para ello, utilizaremos diagramas de secuencia que representarán las interacciones entre los distintos componentes del sistema.

3.4.1. Generación de tareas desde el LLM

La comunicación para realizar la generación de tareas consta de tres componentes: el nodo del LLM, el nodo del servicio y el modelo de lenguaje. En este caso, el nodo recibe información desde el *topic* de la entrada de voz a texto, el cual envía al servicio. El servicio recibe la orden, y junto al prompt del sistema, la envía al LLM, y cuando este termina de generar la respuesta, el servicio devuelve el listado de tareas al nodo, y este lo reenvia al ejecutor de tareas.

Una exepción que puede ocurrir, es que el JSON devuelto no esté bien en el formato correcto, en cuyo caso el nodo volvería a mandar el mensaje al servicio para regenerar la respuesta. A continucación, se muestra el diagrama de secuencia que representa este proceso (véase la figura 3).

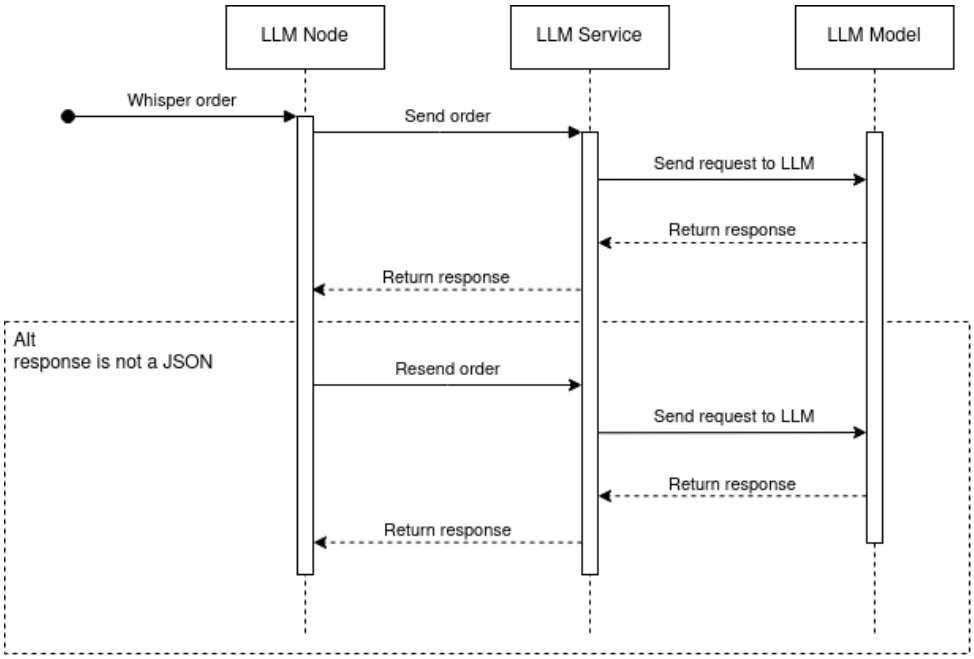

Figura 3: Diagrama de generación de tareas desde el LLM.

3.4.2. Planificación y ejecución de tareas

3.4.3. Monitorización de las tareas

La interfaz de monitorización de las tareas se comunica principalmente recibiendo información acerca del estado de las tareas, aunque también envía información al planificador de tareas, simulando el whisper del robot.

En el diagrama que se muestra a continuación (véase la figura 4), se puede observar cómo se comunican los distintos componentes del sistema para llevar a cabo esta monitorización. se puede observar el proceso desde que el usuario envía una orden al robot, hasta que las tareas del robot han finalizado. En este caso, podemos observar una ejecución completa del sistema, simplificando los procesos anteriormente explicados.

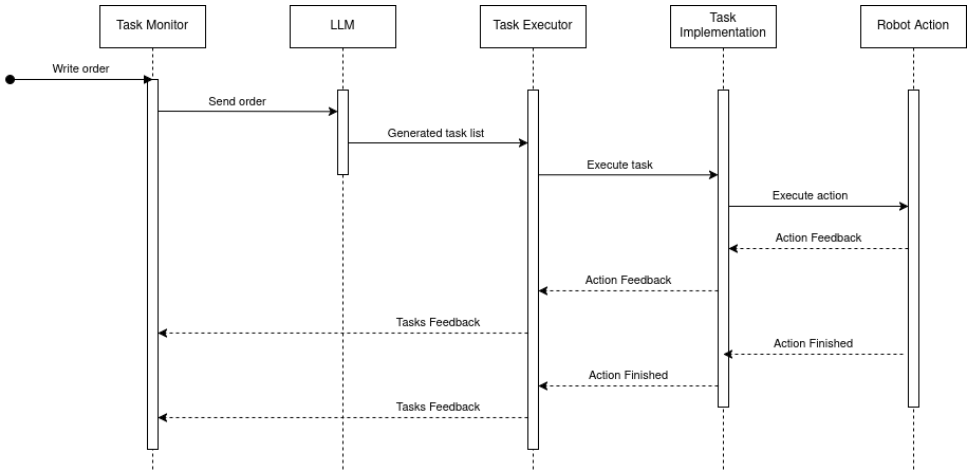

Figura 4: Diagrama de monitorización de tareas.

4

Desarrollo del sistema

4.1. Estructura del proyecto

Comenzaremos la sección de desarrollo dando contexto y mostrando la estructura del proyecto, para que el lector pueda entender cómo se ha organizado el código y qué componentes lo componen.

4.2. Sistemas de comunicaciones

En este apartado del desarrollo abordaremos los sistemas de comunicaciones que se han implementado tanto para comunicarse los nodos de ROS2 entre sí como la integración de un LLM como un servicio de ROS2

4.3. Modelo de lenguaje

A continuación, describiremos la implementación realizada para la comunicación de los distintos modelos de lenguaje que se han utilizado en el proyecto

4.4. Ejecutor y planificador de tareas

4.5. Integración con el robot físico

4.5.1. Navegación del robot

4.5.2. Integración texto a voz

4.6. Interfaces gráficas para monitorización

5

Análisis y pruebas

En esta sección abordaremos los análisis y pruebas realizadas durante el transcurso de investigación de este TFG.

5.1. Análisis

5.2. Pruebas

6

Conclusiones y Líneas Futuras

6.1. Conclusiones

6.2. Líneas Futuras

Referencias

[1] Antonio Cañete y col. “Sistema multimodal para la orientación de robots móviles hacia su interlocutor”. En: *Jornadas de Automática* 45 (2024).

[2] Ana Iglesias y col. “The town crier: a use-case design and implementation for a socially assistive robot in retirement homes”. En: *Robotics* 13.4 (2024), pág. 61.

[3] Petter Ögren y Christopher I Sprague. “Behavior trees in robot control systems”. En: *Annual Review of Control, Robotics, and Autonomous Systems* 5.1 (2022), págs. 81-107.

Apéndice A
Manual de Instalación

Para instalarlo el sistema haz lo siguiente:

- Ins

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA