universidad de león

Escuela de Ingenierías I. I.

Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo de Fin de Grado

MAHURI (Mobile Application for Human-Robot Interaction)

Autor: Iván Fernández González

Tutora: Lidia Sánchez González

Co-tutor: Miguel Ángel González Santamarta

UNIVERSIDAD DE LEÓN
Escuela de Ingenierías I.I.
GRADO EN INGENIERÍA INFORMÁTICA
Trabajo de Fin de Grado

ALUMNO: Iván Fernández González

TUTORA: Lidia Sánchez González

TÍTULO: MAHURI (Mobile Application for Human-Robot Interaction)

CONVOCATORIA: Julio, 2022

RESUMEN:

Este trabajo de fin de grado, describe el diseño y desarrollo de una aplicación para dispositivos móviles Android que permite interactuar con robots de múltiples maneras.

Las funcionalidades que ofrece la aplicación son la tele operación del robot, tanto de forma gráfica como a través de comandos de voz, la navegación de forma autónoma de igual manera a través de la interfaz y de comandos de voz, y por último el acceso a la imagen proporcionada por la cámara del robot y reconocimiento de objetos en las imágenes adquiridas.

La aplicación ha sido desarrollada con Kotlin y Java y, para su correcto funcionamiento, el dispositivo móvil donde se ejecute debe estar conectado a la misma red que el ordenador que esté conectado al robot. En este ordenador se han desarrollado dos nodos escritos en C++ y Python, los cuales utilizando ROS2 envían las órdenes solicitadas por el usuario al robot para que este las ejecute.

Las pruebas realizadas tanto en entornos simulados como en un entorno real, como es el apartamento sito en el módulo de Investigación Cibernética, validan la aplicación desarrollada. Por último, señalar que esta aplicación ocupó la segunda posición en el II Concurso de Prototipos (modalidad software) de la Escuela de Ingenierías Industrial, Informática y Aeroespacial.

ABSTRACT:

This final degree Project describes the design and development of a mobile application for Android devices that allows users to interact with robots in multiple ways.

The functionalities offered by the application are remote operation of the robot, both graphically and through voice commands, autonomous navigation in the same way through the interface and voice commands, and lastly, access to the image provided by the robot´s camera and object recognition in the acquired images.

The application has been developed with Kotlin and Java and, in order to work properly, the mobile device where it runs must be connected to the same network as the computer that is connected to the robot. Two nodes written in C++ and Python have been developed on this computer, which, using ROS2, send the orders requested by the user to the robot so that it can execute them.

The test carried out both in simulated and real environment, such as the apartment located in the Cybernetic Investigation module, validate the developed app. Finally, it should be noted that this application took second place in the II Prototype Contest (software modality) of the Industrial, Computer and Aerospace Engineering School.

Palabras clave: Aplicación móvil, Android, Robot de servicios, ROS2, Kotlin.

Firma del alumno:	**VºBº Tutor:**

Índice de contenidos

Índice de figuras

Índice de tablas

Glosario de términos

- CU: Caso de uso.
- GPS: Global Positioning System.
- HU: Historia de usuario.
- IDE: Integrated Devolpment Environment.
- IP: Internet Protocol.
- MIC: Módulo de Investigación Cibernética.
- ROS: Robot Operating System.
- SSH: Secure Shell.
- TCP: Transmission Control Protocol.
- ULE: Universidad de León.
- WiFi: Wireless Fidelity.

Introducción

Este Trabajo de Fin de Grado trata sobre el desarrollo de una aplicación para dispositivos móviles que utilicen el sistema Operativo Android que permita el control de ciertas acciones que puedan realizar diferentes robots de forma inalámbrica. De esta forma, un usuario sin conocimientos sobre robótica y sistemas operativos sería capaz de tele operar un robot, enviar al robot órdenes y supervisar su funcionamiento.

Planteamiento del problema

En los últimos años ha existido un gran auge en la demanda e interés por temas relacionados con la robótica, lo que ha llevado incluso a que muchos de los robots hoy en día comercializados, sean destinados tanto para fines educativos, domóticos, investigación o incluso simplemente para fines lúdicos. Dentro del ámbito en el que se plantea el problema se excluyen aquellos robots que están destinados para propósitos industriales. Este aumento en el desarrollo y demanda en la robótica ha supuesto que cada vez más personas puedan adquirir o estén en contacto con algún tipo de robot destinado a alguno de los fines indicados anteriormente.

Uno de los problemas que han surgido, y que trata de resolver este proyecto, es la dificultad para interactuar, en algunas ocasiones, con este tipo de robots, sobre todo para gente más inexperta en el ámbito informático o robótico. En muchos de los casos, estos robots incluyen un mando a control remoto que permite la interacción y el dominio de ciertas acciones que el robot incluye. En otras ocasiones, las empresas ofrecen aplicaciones tanto móviles como web nativas que permiten al usuario realizar el control del robot adquirido.

En cualquiera de las dos situaciones mencionadas, la interacción no suele ser del todo eficiente e intuitiva. Además, en el caso de que se trate de un mando a control remoto existe la posibilidad de perderlo o dañarlo y que por tanto se pierda la utilidad del robot por completo. En el caso de la aplicación móvil o web se requiere de la instalación de la misma para poder hacer de las diferentes opciones que te proporcione el robot, y, si se diera la opción de tener dos o más robots de marcas o empresas distintas se requeriría la utilización de diferentes aplicaciones por separado sin poder unificarlas en una.

También, aunque este caso es menos probable, el problema que surge si una persona decide crear su propio robot. Para posteriormente poder hacer uso del mismo debe desarrollar una manera de poder controlarlo, ya sea mediante un mando, una aplicación o un programa independiente.

Objetivos

El principal objetivo de este proyecto es el desarrollo de una aplicación para dispositivos móviles que permita la interacción y control diferentes tipos de robots de forma sencilla, rápida e intuitiva. Para ello, los principales objetivos a cubrir en el proyecto son:

- Realizar un análisis de las diferentes opciones que existen a día de hoy en el mercado que permitan realizar acciones similares a las ofrecidas por la aplicación a desarrollar.
- Desarrollo de una aplicación móvil bien diseñada y con una curva de aprendizaje baja que permita hacer un correcto y agradable uso de la misma a cualquier usuario independientemente de su nivel de conocimientos informáticos.
- Desarrollo del programa encargado de recibir las acciones que el usuario indique en la aplicación móvil y trasladarlas al robot.
- Conexión estable y rápida entre la aplicación móvil y el programa que se envía las órdenes al robot.
- Respuesta rápida y en tiempo real del robot a las diferentes acciones que el usuario desee.
- Opción de poder controlar el robot mediante comandos de voz y no solo desde la interfaz de la aplicación móvil.
- Posibilidad de enviar al robot a los lugares indicados en entornos mapeados y controlados a través de la aplicación.
- Posibilidad de observar la cámara del robot desde la aplicación móvil pudiendo identificare diferentes objetos que el robot visualice.

Metodología

En esta sección se van a detallar las diferentes metodologías utilizadas a la hora de desarrollar el proyecto. En este caso se ha seguido la metodología de modelo en cascada para los proyectos de desarrollo de software. Los pasos de este modelo en cascada están definidos en la siguiente imagen.

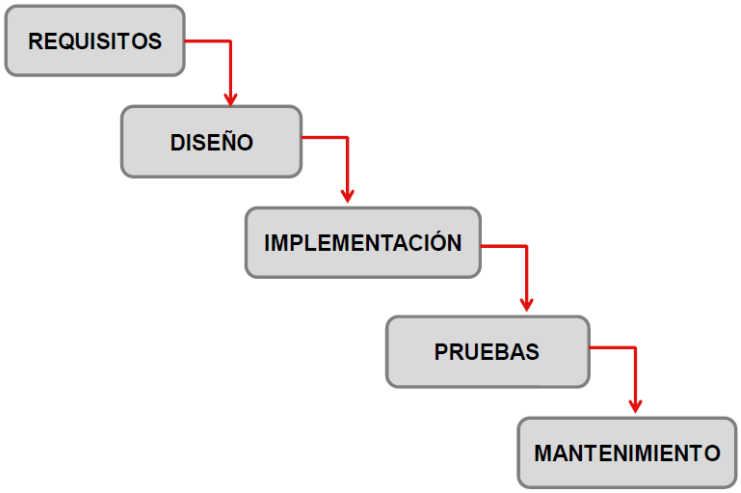


Figura 0.1 Modelo en cascada [26]

A mayores de usar el modelo en cascada para el desarrollo del proyecto, se ha seguido la metodología Scrum para la organización del mismo. Scrum es un tipo de metodología ágil muy utilizado hoy en día en los equipos de desarrollo de software, que incluye diferentes herramientas que hacen que sea muy flexible, lo que permite que se puedan realizar gran cantidad de cambios a lo largo del proceso de desarrollo del proyecto sin causar un gran impacto en el resultado final del mismo ni en el tiempo de desarrollo.

En Scrum existen, generalmente tres tipos de roles, los cuales son:

- Product Owner: En este, el Product Owner sería Iván Fernández González, ya que, en los equipos que trabajan con una metodología Scrum, el Product Owner es el cliente final al que va a ser entregado el proyecto, y que, por tanto, es el encargado de definir los requisitos y funcionalidades necesarias del producto final.
- Scrum Master: El Scrum Master es el rol que se encarga de proporcionar formación al equipo de desarrollo, ayudar a organizar los plazos y resolver problemas que le puedan surgir al equipo de desarrollo a la hora de realizar el proyecto. En este caso habría dos Scrum Master, que serían Lidia Sánchez González y Miguel Ángel González Santamarta.
- Equipo de desarrollo: Se trata del grupo de trabajadores encargados del desarrollo del proyecto. En este caso el equipo de desarrollo estaría formado única y exclusivamente por Iván Fernández González.

Scrum cuenta principalmente con dos pilas donde se recogen las diferentes historias de usuario:

- Product backlog (Pila del producto): En esta pila se recogen todas las historias de usuario propuestas por el Product Owner y consensuadas con el equipo de desarrollo.
- Sprint backlog (Pila del sprint): En esta pila se encuentran todas las historias de usuario que van a ser desarrolladas en el sprint en el que se encuentren para, de esta manera, poder, en cada sprint, incrementar o mejorar la funcionalidad del producto final.

Esta metodología se caracteriza por el uso de sprints a la hora de organizar el trabajo y funcionalidades a desarrollar. Un sprint es un periodo de tiempo con una duración acordada entre todos los miembros del equipo, en el que se llevará a cabo el desarrollo de diferentes historias de usuario, de tal manera que al final de cada sprint se haya incrementado el producto añadiendo al mismo el desarrollo de las historias de usuario seleccionadas para ese sprint.

Cada uno de los sprints cuenta con unas fases definidas:

1. Planificación del sprint: En esta reunión, el equipo de desarrollo planifica las historias de usuario y el trabajo que va a realizar durante ese sprint. En esta reunión también está presente el Scrum Master. Se debe dejar claro,

al final de esta reunión cuál es el trabajo a desarrollar durante este sprint y el resultado que se desea obtener cuando finalice el mismo.
2. Scrum diario: Se trata de una reunión diaria con un periodo de duración muy corto en el que cada uno de los miembros del equipo de desarrollo expone lo que ha hecho el día anterior, las tareas que tiene para el día en el que se realice la reunión y los posibles problemas que le hayan surgido.
3. Revisión del sprint: Consiste en una pequeña reunión informal al final del sprint en la que el equipo de desarrollo se reúne con el Product Owner para observar la demostración del nuevo incremento que se ha hecho en el proyecto durante ese sprint. Esta reunión también se utiliza para que el Product Owner repase la pila del producto basándose en las historias de usuario cumplidas durante el desarrollo del sprint.
4. Retrospectiva del sprint: El equipo de desarrollo se reúne para documentar y analizar tanto los éxitos como los problemas o fracasos que hayan surgido durante el sprint y poner en común diferentes soluciones que les permitan mejorar los próximos sprints.

Ahora que se ha definido la metodología Scrum, en este proyecto se ha tomado como duración de cada uno de los sprints el periodo de dos semanas, una vez finalizado cada uno de los sprints se realizaba una reunión entre el los Scrum Masters y el equipo de desarrollo para comprobar cómo había funcionado el anterior sprint y detallar qué es lo que se iba a realizar en los siguientes. Se tomaron como historias de usuario para la pila del producto aquellas que han sido definidas en el apartado anterior de objetivos del proyecto. Cada uno de los sprints se detalla con más profundidad en el apartado 2.2 de este mismo documento.

Estructura del trabajo

En este apartado, se permite obtener una visión general del contenido que se encontrará en cada una de las secciones del documento a través de una pequeña descripción de los mismos:

- **Introducción:**
  En esta sección se presenta al lector la finalidad, objetivos y metodología utilizada en el proyecto.

- **1. Estudio del problema:**
  En esta sección del documento se describe el problema encontrado y que, por tanto, el proyecto desarrollado trata de solucionar. Además, se hace un estudio sobre otras soluciones que sean similares o aborden el problema desde otro punto de vista.

- **2. Gestión del proyecto software:**
  En este apartado se muestra cómo se ha llevado a cabo la planificación a la hora de desarrollar la solución propuesta. Se detallan temas como el alcance del proyecto, la planificación, la gestión de los diferentes recursos

y el ámbito legal que se ha tenido en cuenta a la hora de desarrollar el proyecto.

- **3. Solución:**
  Esta sección describe con mayor exactitud cómo es la solución desarrollada incluyendo una descripción sobre cómo ha sido el proceso de desarrollo.

- **4. Evaluación:**
  Este apartado demuestra la validez de la solución desarrollada y las diferentes formas y técnicas utilizadas para asegurar el correcto funcionamiento de la misma.

- **5. Conclusión:**
  En este apartado se detallan los conceptos y aprendizaje obtenidos una vez la solución ha sido desarrollada. Se realiza una reflexión sobre los problemas encontrados durante el desarrollo del mismo y una opinión personal del proyecto.

- **Anexo A. Control de versiones:**
  En este anexo se describe el sistema de control de versiones utilizado y la forma en la que fue usado para el desarrollo del proyecto.

- **Anexo B: Seguimiento de proyecto de fin de carrera:**
  Se representa el diagrama de planificación inicial y final del proyecto.

- **Anexo C: Manual de usuario:**
  Se indican las acciones que se deben realizar para poner en funcionamiento el sistema y hacer un correcto uso del mismo.

1 Estudio del problema

En esta sección se destacan los principales conceptos que se deben conocer antes de indagar en la solución propuesta en el proyecto. Se incluyen, además, algunas opciones similares a la desarrollada que ya existen a día de hoy en el mercado.

1.1 El contexto del problema

Hoy en día, el auge por el uso de robots en la vida cotidiana está aumentando de forma muy considerable y cada vez son más las personas que hacen uso de los mismos o han tenido contacto con alguno de ellos. Pero existe un problema, ya que, en muchas ocasiones, la forma de controlarlos no es del todo intuitiva o sencilla independientemente del tipo de robot que sea o la función para la que esté destinado.

Es común encontrar robots que solo pueden ser dirigidos a través de mandos a control remoto, lo cual, aparte de limitar en cierta medida las acciones que el robot puede realizar, también puede suponer un problema si el mismo es extraviado o estropeado ya que el robot quedaría inutilizable. Además, en varias ocasiones, estos mandos suelen ser poco ergonómicos y portables, lo que puede dificultar de manera significativa el control del robot. Mientras que hoy en día un móvil lo usa gente de todas las edades y características, los mandos están más limitados al mundo de los videojuegos o para ciertas actividades tele operadas como manejo de grúas u otros equipos. Otro punto negativo a destacar de este tipo de forma de controlar los robots, es que es muy difícil poder reunir todas las acciones que puede realizar el robot dentro de un solo mando y en la mayoría de ocasiones solo se utiliza para mover el robot a los lugares que se desee.

Otra de las formas de controlar los robots más comunes es a través de una aplicación nativa proporcionada por la empresa fabricante del robot. Esta suele ser una mejor opción, pero solo te permite realizar ciertas acciones que, en muchas ocasiones, no son todas las que se podrían realizar con el robot. Además, otro punto negativo, es que, en caso de tener más de un robot se debería tener instaladas y configuradas más de una aplicación, lo que en muchas ocasiones puede suponer un problema para algunas personas que no estén muy familiarizadas con la tecnología o que cuenten con dispositivos con bajos recursos.

1.2 El estado de la cuestión

Actualmente no existen gran cantidad de soluciones que traten de solucionar el problema encontrado. A continuación, se describen y comparan algunas de las existentes:

RemoteBot [1] **:**

Se trata de una aplicación cliente-servidor que permite controlar los robots mediante dispositivos móviles Android. Fue desarrollada por un grupo de estudiantes pertenecientes a la Universidad de La Plata en Argentina y su desarrollo fue iniciado en el año 2009 utilizando los lenguajes de programación Python y Java. Cuenta con una interfaz que muestra los controles para manejar el robot a través de la que se permite controlar parámetros como la velocidad a la que se va a desplazar el robot (de 0 a 100), también se puede configurar el modo avance de manera que el robot avance sin chocar, muestra los valores del sensor de obstáculos del robot, una opción para que el robot gire a la mitad de la velocidad indicada y diferentes flechas y botones que permiten mover al robot tanto para adelante, atrás, derecha, izquierda o detenerlo por completo.

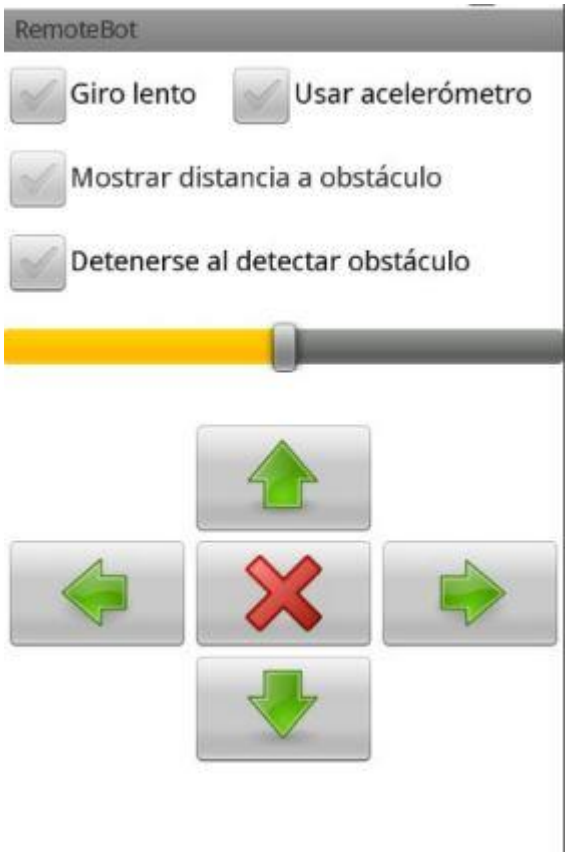


Figura 1.1 Captura de pantalla RemoteBot

La aplicación RemoteBot tiene algunas similitudes con la solución propuesta en este proyecto como por ejemplo la posibilidad de controlar de forma remota la dirección y movimientos del robot. Sin embargo, la solución descrita en este documento aparte de tratarse de una aplicación más accesible, también incluye otras funcionalidades a mayores como la posibilidad de realizar estas mismas

acciones a través de comandos de voz, la posibilidad, en entornos controlados, de poder enviar al robot al lugar donde se desee, y también se permite observar la cámara del robot a través de la aplicación móvil, ventajas con las que RemoteBot no cuenta.

Robot Control [2] **:**

Robot Control es una aplicación para dispositivos móviles Android desarrollada en 2016 por la compañía Dalvik Apps que permite controlar robots a través del Bluetooth del dispositivo móvil. Entre las funciones que incluye se pueden destacar la opción de dirigir el robot hacia adelante, atrás, izquierda, derecha o incluso detenerlo a través de su interfaz gráfica. Otras funciones a desatacar, es que en los robots que cuenten con esa funcionalidad se permite también controlar la altura de la pala, como podría ser el caso de una excavadora, o la altura de las muñecas del robot.

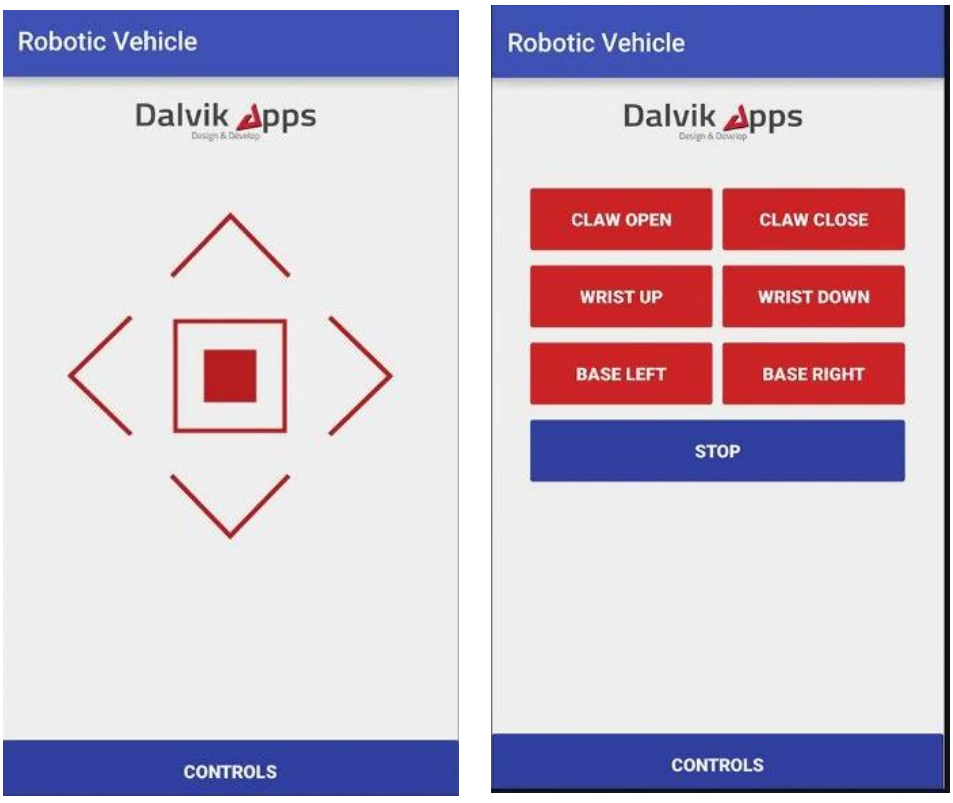


Figura 1.2 Capturas de pantalla Robot Control

Al igual que la aplicación RemoteBot mencionada anteriormente, esta aplicación comparte la funcionalidad con la descrita en este documento de poder controlar de forma remota el robot. Pero en este caso no permite las otras acciones mencionadas anteriormente como hacerlo a través de la voz, poder enviar al robot a diferentes lugares de forma automática o poder observar, en caso de que el robot disponga de una, la cámara del mismo. Además, solo permite controlar

robots que cuenten con tecnología Bluetooth, lo cual puede ser una desventaja, ya que algunos de ellos hoy en día no cuentan con esta tecnología.

Formant [3] **:**

Formant es una aplicación web que permite realizar funciones como operar, observar y analizar diferentes aspectos de los robots con los que sea vinculada.

Entre sus funciones más destacadas está la de poder tele operar con el robot a través de su interfaz de manera totalmente remota, obtener datos en tiempo real del robot, tanto del gps, velocidad de movimiento, orientación, etc. Además, cuenta con la funcionalidad de poder observar la cámara del robot en tiempo real o incluso poder acceder al robot a través del comando ssh de manera remota por si se necesitara realizar alguna configuración adicional.

Cuenta con un diseño bastante moderno y accesible que permite, en una sola pantalla poder realizar diferentes acciones y, a su vez, poder acceder a diferentes datos recopilados del robot que pueden llegar a ser mostrados incluso en forma de gráfica de manera que sea mucho más legible y accesible. La empresa que desarrolla esta herramienta fue fundada por ingenieros de software especialistas en robótica y gerentes de productos de empresas líderes en tecnología como pueden ser Google, Amazon Web Services, Microsoft o IBM.

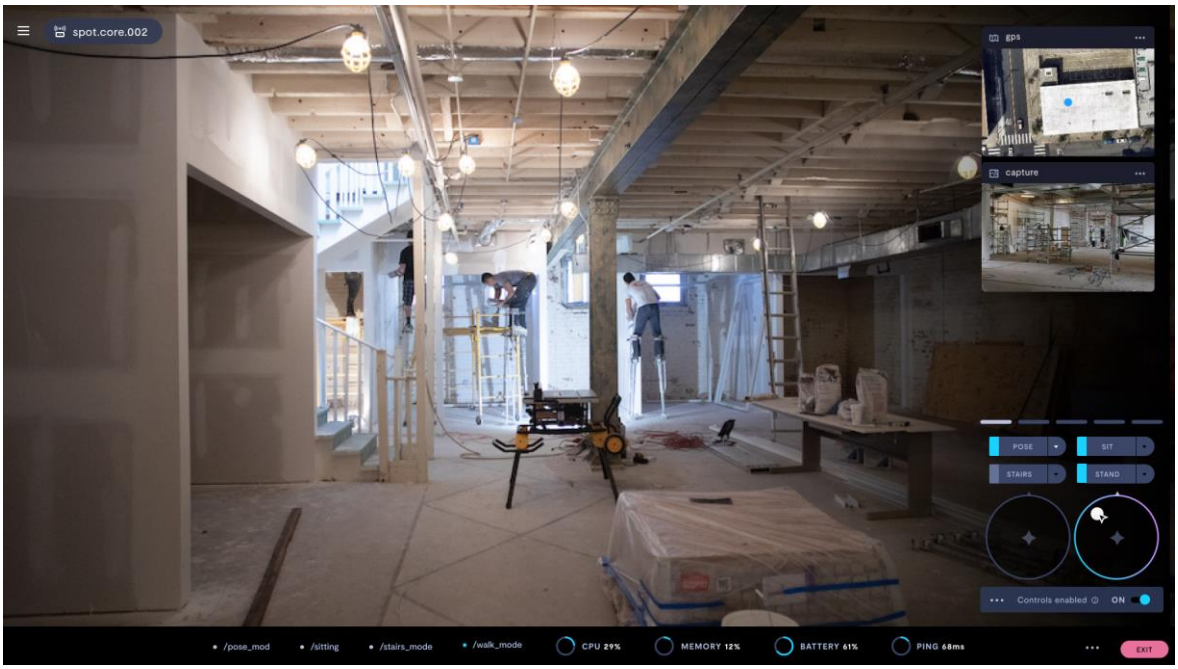


Figura 1.3 Captura de pantalla Formant

Como se puede observar, esta aplicación tiene un desarrollo y apariencia mucho más profesional que las otras dos soluciones mencionadas anteriormente. En comparación con el proyecto descrito en este documento cuenta con todas las funcionalidades ofrecidas por el mismo excepto la del control de las acciones

mediante comandos de voz. Asimismo, cabe destacar que esta aplicación está limitada para ser usada solo por algunos robots, no podría ser aplicable a cualquiera que se deseara. Además, se trata de una aplicación web destinada a un entorno más industrial, mientras que este proyecto pretende que sea utilizable de manera sencilla por cualquier tipo de usuario y robot. Otro punto a destacar, es que en este caso Formant se trata de una aplicación web, mientras que la solución desarrollada es una aplicación móvil.

1.3 La definición del problema

Como se ha comentado en los apartados anteriores se ha detectado una ausencia de una aplicación para dispositivos móviles que permita reunir diferentes funcionalidades como pueden ser la posibilidad de tele operar los robots tanto a través de la interfaz de la aplicación como a través de comandos de voz, la posibilidad de enviar el robot a un lugar en concreto siempre y cuando el entorno esté mapeado, y por último la posibilidad de observar desde la propia aplicación la cámara del robot. Además, como se ha podido ver, no existe una aplicación que pueda realzar todas estas acciones para la mayoría de los robots, ya que muchas de ellas están limitadas a los robots fabricados por la propia empresa o aquellos que cuenten con una tecnología específica como el bluetooth.

El problema que se intenta resolver en este proyecto es precisamente esta falta de aplicaciones en el ámbito robótico que permita poder interactuar con los robots de una forma más sencilla para cualquier tipo de usuario. Además, cada vez es más frecuente el uso de robots para realizar cualquier tipo de tareas de nuestra vida cotidiana, y esta solución podría servir como introducción a la robótica a personas que no están para nada familiarizadas con este tema.

A su vez, también podría resultar muy útil para personas más familiarizadas con el tema y que por ejemplo se aventuraran a construir sus propios robots, ya que en estos casos siempre suele ser necesario el uso de un mando externo como el de una consola con su correspondiente configuración asociada al robot para poder controlarlo. En este caso, simplemente habría que cambiar un par de parámetros para poder adaptarlos al robot que se haya creado y la aplicación sería totalmente funcional, además, podría permitir al usuario realizar una mayor cantidad de acciones que si usara un mando para su control, como puede ser la posibilidad de observar la cámara del robot o el control por voz del mismo.

Otra de las ventajas que tiene la solución propuesta es que puede ser ejecutada en dispositivos móviles, lo que lo hace una aplicación mucho más accesible para cualquier usuario ya que hoy en día prácticamente todas las personas poseen uno y hacen uso del mismo en su vida cotidiana.

2 Gestión de proyecto software

En esta sección se reflejan todos los elementos que se han tenido en cuenta a la hora de planificar y definir el proyecto. De esta manera podemos hacer una estimación sobre cuáles serían los recursos necesarios divididos en diferentes apartados para llevar a cabo el desarrollo de la solución propuesta.

2.1 Alcance del proyecto

Este apartado hace referencia a los diferentes costes, productos y servicios necesarios para llevar a cabo el desarrollo del proyecto.

2.1.1 Definición del proyecto

El proyecto que se desea llevar a cabo es el desarrollo de una aplicación para dispositivos móviles con sistema operativo Android que permita realizar las funciones de control de un robot a través de su interfaz gráfica o mediante comandos de voz, este control será un tele operador que permitirá mover al robot hacia adelante, hacia atrás, a la derecha, a la izquierda o detenerlo por completo. Otras de las funcionalidades que implementa la aplicación es de poder desplazar al robot, en entornos controlados y mapeados a diferentes lugares o habitaciones del espacio en el que se encuentre, esto también puede ser realizado tanto a través de la interfaz gráfica de la aplicación como a través de comandos de voz. La última funcionalidad que se la aplicación debe implementar es la de mostrar la imagen que se vea a través de la cámara del robot previamente habiendo sido procesada por una red neuronal que permita detectar algunos objetos que el robot esté visualizando en ese momento.

A mayores de la aplicación móvil también se debe desarrollar un programa que se ejecute en el ordenador que esté conectado al robot para que sea este el que reciba las acciones solicitadas por el usuario en la aplicación y que las traslade al robot para que este mismo las ejecute.

2.1.2 Estimación de tareas y recursos

En este apartado se describen las tareas y recursos que se prevé que serán necesarios para llevar a cabo el desarrollo completo del proyecto.

2.1.2.1 Tareas

Como se mencionó anteriormente el proyecto se va a desarrollar utilizando la metodología en cascada, lo que indica que, en primer lugar, será necesario un periodo de tiempo dedicado a que el equipo de desarrollo investigue las tecnologías y herramientas a utilizar para, de esta manera, poder realizar una mejor estimación sobre los tiempos que puede llevar el desempeño de cada tarea.

Una vez finalizado este periodo de investigación se procede a comenzar la fase de desarrollo en la que se utiliza la metodología ágil Scrum, mencionada en el apartado referente a la metodología en la sección de introducción del documento.

2.1.2.2 Recursos

Los recursos que se requieren para el desarrollo del proyecto se pueden dividir en las siguientes categorías:

- Físicos: Dentro de este tipo de recursos se encuentran algunos como la oficina, necesaria para la planificación y realización del desarrollo por parte de todo el equipo de trabajo y diferentes gastos que esta oficina conlleva como el gasto de la luz o internet. También se encuentra dentro de esta categoría el equipo necesario para el desarrollo del proyecto, es decir, un ordenador para cada uno de los integrantes del equipo y al menos un robot destinado a la investigación, como por ejemplo TIAGo [4] para realizar las diferentes pruebas necesarias con la aplicación móvil. En lo referente al software no existe ningún coste adicional en las tecnologías a utilizar, ya que todas ellas son de código abierto.
- Humanos: En este tipo de recursos se definen las personas que conformarán el equipo de trabajo encargado de llevar a cabo el proyecto. En el caso de que el proyecto fuera desarrollado a nivel comercial, el equipo debería estar formado por un Scrum Master que será el encargado de guiar en el uso de la metodología Scrum al equipo de desarrollo, que estaría formado por un desarrollador software junior, un diseñador gráfico y un tester de software. En el caso del proyecto realizado al que se hace referencia en este documento solo existiría un Scrum Master que sería la tutora del trabajo de fin de grado y el resto de puestos serían ocupados por el autor del documento.

2.1.3 Presupuesto

El presupuesto del proyecto tiene como funcionalidad realizar una estimación inicial del coste total del mismo desde el inicio hasta el fin de su desarrollo. En esta sección se mostrará el coste de cada uno de los recursos necesarios para la realización del proyecto.

En este caso, para llevar a cabo el desarrollo de nuestro proyecto el único hardware específico que debería adquirirse sería el robot indicado anteriormente que sería el utilizado para realizar las pruebas necesarias para garantizar el correcto. Se ha elegido este tipo de robot ya que es de los más comunes y de los más comercializados dentro del ámbito de la investigación. Además, dentro de estos gastos deben incluirse también los gastos referentes al alquiler de la oficina y los gastos que puede conllevar la misma. Estos precios deben ser calculados en función de la estimación inicial que debería prolongarse el desarrollo del proyecto, que en este caso se estima en un periodo de 7 meses.

El presupuesto necesario para adquirir los recursos físicos necesarios para desarrollar el proyecto sería:

- Robot TIAGo: el coste estimado de este tipo de robot destinado a la investigación se encuentra alrededor de los 30.000€ [5], ya que la configuración que necesitaríamos del mismo no sería la más compleja.

Objeto	Cantidad	Precio	Total
Robot TIAGo	1	30.000 €	30.000 €
Total		30.000 €	

Tabla 2.1 Presupuesto recursos físicos

En estos gastos no se incluye el alquiler de la oficina ni los propios gastos que esta conlleva como la luz, el internet o el equipo informático de la misma ya que se incluye como costes indirectos a la hora de calcular el presupuesto final.

A continuación, se realizará un estudio similar referente a los recursos humanos necesarios para el desarrollo del proyecto. Como se comentó anteriormente serían necesarios un Scrum Master, un desarrollador de software junior, un diseñador gráfico y un tester de software:

- Scrum Master: El sueldo medio en España ronda los 24€/h [6].
- Desarrollador de software junior: Según [7] el sueldo medio en España de un desarrollador de software junior es aproximadamente de 12€/h.
- Diseñador gráfico: En este caso, el sueldo promedio de un diseñador gráfico en España se sitúa entre los 18€/h [8].
- Tester de software: Por último, el sueldo medio de un tester de software ronda los 14 €/h según la fuente [9].

Estimando cuántas horas de trabajo serían necesarias por parte de cada uno de los trabajadores del equipo se puede calcular el presupuesto necesario:

	Salario (€/h)	Horas	Total
Scrum Master	24 €	600	14.400 €
Desarrollador de software	12 €	1100	13.200 €
Tester de software	18 €	600	10.800 €
Diseñador gráfico	14 €	600	8.400 €
Total		46.800 €	

Tabla 2.2 Presupuesto recursos humanos

De esta manera podemos calcular una estimación de la cantidad de presupuesto necesaria para la realización del proyecto:

Tipo de presupuesto	Porcentaje	Coste
Físicos		30.000 €
Humanos		46.800 €
Total costes directos		76.800 €
Costes indirectos	15%	11.520 €
Total costes		88.320 €
Benefico industrial	11%	9.715,20 €
Subtotal		98.035,20 €
IVA aplicable	21%	20.587,40
Total		118.622,60 €

Tabla 2.3 Presupuesto total

De esta manera, tras haber realizado el cálculo con todos los costes indirectos, el beneficio industrial y el IVA aplicable, se determina que el presupuesto final del proyecto rondará los 118.622,60 €.

2.2 Plan de trabajo

En esta sección se exponen las diferentes tareas que formarán parte del desarrollo del proyecto.

2.2.1 Identificación de tareas

A continuación, se muestra una lista que muestra la organización de las tareas en las que se ha dividido el desarrollo del proyecto.

- Investigación
- Diseño de la arquitectura del sistema
- Diseño de la interfaz del sistema
- Planificación
- Implementación
    - Creación del proyecto (Sprint 1):
        - HU01: Creación del proyecto en Android Studio
        - HU02: Creación del repositorio en GitHub
    - Interfaz de la aplicación (Sprint 2):
        - HU03: Desarrollo de la interfaz de la aplicación
    - Nodo tele operación (Sprint 3):
        - HU04: Desarrollo del nodo tele operador servidor
        - HU05: Desarrollo del nodo tele operador cliente
        - HU06: Conexión nodo tele operador - aplicación
    - Funcionalidad tele operación y navegación (Sprint 4):
        - HU07: Funcionalidad tele operador

        - HU08: Funcionalidad navegación
    - Reconocimiento de voz (Sprint 5):
        - HU09: Reconocimiento de voz
        - HU10: Funcionalidad reconocimiento de voz
    - Nodo cámara (Sprint 6):
        - HU11: Desarrollo del nodo cámara
    - Funcionalidad cámara (Sprint 7):
        - HU12: Conexión nodo cámara - aplicación
        - HU13: Funcionalidad cámara
    - Testeo del sistema (Sprint 8).
        - HU14: Pruebas del sistema
- Documentación del proyecto
- Revisión de la documentación

2.2.2 Estimación de tareas

En este apartado se detallan las diferentes tareas listadas en el apartado anterior. Cabe destacar, antes de comenzar con el desarrollo de este apartado, que las tareas de investigación, diseño y planificación no forman parte de la metodología Scrum, se trata de unas tareas previas a esta metodología pero que son necesarias para el correcto desarrollo del proyecto.

2.2.2.1 Investigación

Esta primera fase del proyecto se utiliza para obtener la información necesaria sobre cuáles son las tecnologías que se van a usar para el desarrollo del mismo de manera que se satisfagan de forma correcta los requisitos propuestos. Tras realizar esta investigación se llegó a la conclusión de que la mejor tecnología para el desarrollo de la aplicación móvil era el lenguaje de programación Kotlin.

Se debía realizar un estudio también, sobre la tecnología que se iba a utilizar para establecer la conexión con los robots y poder enviar las órdenes necesarias y que este fuera capaz de ejecutarlas correctamente. En este caso se decidió que la mejor tecnología para desempeñar esta tarea era ROS2, y que para publicar la información y establecer la comunicación con la aplicación móvil se desarrollarían los nodos en los lenguajes C++ y Python.

2.2.2.2 Diseño de la arquitectura del sistema

Una vez la fase de investigación ha sido completada se debe proceder a la fase de diseño de la arquitectura del sistema. Finalmente se llegó a la conclusión de que el diseño de la arquitectura del sistema debía ser como el representado en la Figura 3.10.

2.2.2.3 Diseño de la interfaz del sistema

En esta fase se trata de desarrollar un prototipo sobre cómo va a lucir, en este caso, la interfaz de la aplicación móvil desarrollada. Este diseño viene especificado en el apartado 3.2.2.2 de este mismo documento.

2.2.2.4 Planificación

La fase de planificación sirve para establecer cuáles serán los materiales necesarios para llevar a cabo el desarrollo del proyecto, detallando en la misma la gestión de los requisitos y estimación de los recursos necesarios para el proyecto. Además, se establece una estimación sobre la duración de cada una de las tareas y la duración total de todo el proyecto.

2.2.2.5 Implementación

Esta es la fase en la que se comienza a implementar la metodología Scrum. Para ello es necesario definir, en primer lugar, cuáles serán las historias de usuario que formarán parte de la pila del producto. Las historias de usuario están formadas por los siguientes campos:

- ID: Se trata de un identificador único asignado a cada una de las historias de usuario, el formato es HUXX.
- Nombre: Nombre con el que se identifica la historia de usuario.
- Estimación: Indica el nivel de esfuerzo en un rango de 0 a 10 que requiere la realización de la tarea descrita.
- Valor: Preferencia respecto al resto de tareas. Se representa con un rango de 0 a 100.
- Descripción: Texto que describe en lo que consiste la historia de usuario.
- Condiciones de satisfacción: Condiciones que deben cumplirse para que la historia de usuario pueda darse por completada.

En el caso de este proyecto las historias de usuario que conforman la pila del producto serán las siguientes:

HU01	**Creación del proyecto en Android Studio**	
Creación y configuración inicial del proyecto en la plataforma Android Studio seleccionando las tecnologías que van a ser utilizadas para el desarrollo del mismo.		
Estimación: 2	**Valor:** 100	**Dependencias:**
Condiciones de satisfacción: • Comprobar que el proyecto ha sido creado y compila de forma correcta. • Ejecutar de forma correcta el proyecto inicial.		

Tabla 2.4 Historia de usuario HU01

HU02	**Creación del repositorio en GitHub**	
Crear el repositorio privado en la plataforma GitHub y enlazarlo con el proyecto anteriormente creado en Android Studio.		
Estimación: 3	**Valor:** 80	**Dependencias:** HU01
Condiciones de satisfacción: • Realizar el primer push al repositorio y comprobar que se realiza de forma correcta.		

Tabla 2.5 Historia de usuario HU02

HU03	**Desarrollo de la interfaz de la aplicación**	
Creación de la interfaz dentro del proyecto según haya sido detallada en la fase de diseño de la interfaz del sistema.		
Estimación: 9	**Valor:** 70	**Dependencias:**
Condiciones de satisfacción: • Comprobar que la interfaz luce y funciona de forma correcta.		

Tabla 2.6 Historia de usuario HU03

HU04	**Desarrollo del nodo tele operador servidor**	
Realizar el programa en C++ que contenga el socket que se conecte con la aplicación móvil y reciba las órdenes que esta envíe.		
Estimación: 4	**Valor:** 60	**Dependencias:**
Condiciones de satisfacción: • Comprobar que recibe correctamente los mensajes procedentes de la aplicación móvil. • Envía correctamente las órdenes al nodo tele operador cliente.		

Tabla 2.7 Historia de usuario HU04

HU05	**Desarrollo del nodo tele operador cliente**	
Desarrollar el programa en C++ que reciba las órdenes del nodo tele operador servidor y que envía las órdenes al robot en función del mensaje recibido.		
Estimación: 6	**Valor:** 60	Dependencias: HU04
Condiciones de satisfacción: • Comprobar que la conexión con el nodo servidor se realiza de forma correcta y se reciben los mensajes enviados por el mismo. • El robot ejecuta correctamente las acciones enviadas por este nodo.		

Tabla 2.8 Historia de usuario HU05

HU06	**Conexión nodo tele operador - aplicación**	
Desarrollar la parte del proyecto dentro de la aplicación encargada de establecer la comunicación entre el nodo tele operador y la aplicación móvil.		
Estimación: 5	**Valor:** 80	**Dependencias:** HU04
Condiciones de satisfacción: • El nodo tele operador servidor recibe de forma correcta los mensajes enviados por la aplicación a través del socket.		

Tabla 2.9 Historia de usuario HU06

HU07	**Funcionalidad tele operador**	
Dar funcionalidad a los botones correspondientes a la parte de tele operación dentro de la interfaz y configurar los eventos que lanzarán.		
Estimación: 4	**Valor:** 60	**Dependencias:** HU03, HU06
Condiciones de satisfacción: • Los eventos y mensajes desencadenados por cada uno de los botones al ser pulsados se ejecutan de manera correcta.		

Tabla 2.10 Historia de usuario HU07

HU08	**Funcionalidad navegación**	
Dar funcionalidad a los botones correspondientes a la navegación a diferentes ligares dentro del entorno mapeados y configurar los eventos que lanzan.		
Estimación: 4	**Valor:** 60	**Dependencias:** HU03, HU06
Condiciones de satisfacción: • Los eventos y mensajes desencadenados por cada uno de los botones al ser pulsados se ejecutan de manera correcta.		

Tabla 2.11 Historia de usuario HU08

HU09	**Reconocimiento de voz**	
Desarrollar el sistema encargado de captar y reconocer los comandos de voz enviados a través del micrófono del dispositivo móvil.		
Estimación: 7	**Valor:** 50	**Dependencias:**
Condiciones de satisfacción: • Comprobar que el sistema reconoce de forma correcta los mensajes enviados a través del micrófono del dispositivo móvil.		

Tabla 2.12 Historia de usuario HU09

HU10	**Funcionalidad reconocimiento de voz**	
Enviar los comandos reconocidos a través del reconocimiento de voz al nodo tele operador.		
Estimación: 3	**Valor:** 65	**Dependencias:** HU09
Condiciones de satisfacción: • Comprobar que el nodo tele operador recibe de forma correcta los mensajes captados a través del reconocimiento de voz de la aplicación.		

Tabla 2.13 Historia de usuario HU10

HU11	**Desarrollo del nodo cámara**	
Desarrollar el programa en Python que capte las imágenes de la cámara del robot, las pase por la red neuronal elegida para detectar objetos y las envíe a la aplicación móvil.		
Estimación: 8	**Valor:** 80	**Dependencias:**
Condiciones de satisfacción: • Comprobar que el nodo capta correctamente las imágenes de la cámara del robot. • Comprobar que la red neuronal capta de forma correcta los objetos visualizados por el robot.		

Tabla 2.14 Historia de usuario HU11

HU12	**Conexión nodo cámara - aplicación**	
Desarrollar la parte de la aplicación móvil que realiza la conexión entre la misma y el nodo de la cámara.		
Estimación: 5	**Valor:** 60	**Dependencias:** HU11
Condiciones de satisfacción: • La aplicación recibe de forma correcta la imagen enviada por el nodo correspondiente a la cámara.		

Tabla 2.15 Historia de usuario HU12

HU13	**Funcionalidad cámara**	
Desarrollar la parte de la aplicación móvil que permita mostrar la imagen recibida por parte del nodo de la cámara.		
Estimación: 4	**Valor:** 45	**Dependencias:** HU12
Condiciones de satisfacción: • La aplicación muestra de forma correcta la imagen recibida.		

Tabla 2.16 Historia de usuario HU13

HU14	Pruebas del sistema	
Realizar pruebas para verificar que el sistema funciona de forma correcta y cumple con los requisitos acordados.		
Estimación: 4	**Valor:** 45	**Dependencias:** HU07, HU8, HU10, HU13
Condiciones de satisfacción: • La aplicación cumple los requisitos establecidos		

Tabla 2.17 Historia de usuario HU14

2.2.2.6 Documentación y revisión de la documentación

Estas dos fases de documentación y revisión de la misma sirven para, tras haber finalizado el desarrollo del proyecto, describirlo en este documento con el fin exponerlo al público que desee obtener información sobre el mismo.

2.2.3 Planificación de tareas

Las diferentes tareas mencionadas anteriormente en las que se descompone el proyecto están organizadas de manera independiente una de otra exceptuando las tareas dedicadas al desarrollo, que son las que se realizan utilizando la metodología Scrum y se dividen en sprints. Cada uno de estos sprints estaría formado por una reunión de planificación, el propio sprint y, finalmente, una reunión de revisión del mismo. De esta forma, la organización del proyecto ha sido desarrollada de la siguiente manera:

- **Investigación:** 01/12/2021 – 19/12/2021

  Participantes: Scrum Master y Equipo de desarrollo.
- **Diseño de la arquitectura del sistema:** 20/12/2021 – 09/01/2022

  Participantes: Scrum Master y Equipo de desarrollo.
- **Diseño de la interfaz del sistema:** 10/01/2022 – 23/01/2022

  Participantes: Scrum Master y Equipo de desarrollo.
- **Planificación:** 24/01/2022 – 06/02/2022

  Participantes: Scrum Master y Equipo de desarrollo.
- **Sprint 1:** 07/02/2022 – 20/02/2022

  Participantes: Scrum Master y Equipo de desarrollo (excepto Tester).
  - Reunión de planificación: Durante esta reunión se deciden cuáles serán las historias de usuario que van a ser desarrolladas en el sprint. En este caso se decidió que se tratará de la creación del proyecto.
  - Sprint: Las historias de usuario que forman parte de la pila del sprint son: HU01 y HU02.
  - Reunión de revisión: Todas las historias de usuario de la pila del sprint se completaron con bastante margen de tiempo.

- **Sprint 2:** 21/02/2022 – 06/03/2022

  Participantes: Scrum Master y Equipo de desarrollo (excepto Tester).

  - Reunión de planificación: Durante esta reunión se deciden cuáles serán las historias de usuario que van a ser desarrolladas en el sprint. En este caso se decidió que se tratará de interfaz de la aplicación.
  - Sprint: Las historias de usuario que forman parte de la pila del sprint son: HU03.
  - Reunión de revisión: Todas las tareas se completaron con éxito, pero posteriormente hubo que retocar algunas partes de la interfaz para que la aplicación luciera mejor y fuese más intuitiva.

- **Sprint 3:** 07/03/2022 – 20/03/2022

  Participantes: Scrum Master y Equipo de desarrollo (excepto Tester).

  - Reunión de planificación: Durante esta reunión se deciden cuáles serán las historias de usuario que van a ser desarrolladas en el sprint. En este caso se decidió que se tratará del nodo tele operador.
  - Sprint: Las historias de usuario que forman parte de la pila del sprint son: HU04, HU05 y HU06.
  - Reunión de revisión: Las tareas de la pila del sprint se completaron de manera exitosa sin ningún contratiempo destacable.

- **Sprint 4:** 21/03/2022 – 03/04/2022

  Participantes: Scrum Master y Equipo de desarrollo (excepto Tester).

  - Reunión de planificación: Durante esta reunión se deciden cuáles serán las historias de usuario que van a ser desarrolladas en el sprint. En este caso se decidió que se tratará de la funcionalidad referente a la tele operación y navegación.
  - Sprint: Las historias de usuario que forman parte de la pila del sprint son: HU07 y HU08.
  - Reunión de revisión: Las historias de usuario seleccionadas para este sprint se completaron de manera satisfactoria solventando un problema que bloqueaba las acciones que se enviaban al robot.

- **Sprint 5:** 04/04/2022 – 17/04/2022

  Participantes: Scrum Master y Equipo de desarrollo (excepto Tester).

  - Reunión de planificación: Durante esta reunión se deciden cuáles serán las historias de usuario que van a ser desarrolladas en el sprint. En este caso se decidió que se tratará del reconocimiento de voz.
  - Sprint: Las historias de usuario que forman parte de la pila del sprint son: HU09 y HU10.
  - Reunión de revisión: Todas las tareas se completaron de manera exitosa sin ningún contratiempo destacable.

- **Sprint 6:** 18/04/2022 – 01/05/2022

Participantes: Scrum Master y Equipo de desarrollo (excepto Tester).

- Reunión de planificación: Durante esta reunión se deciden cuáles serán las historias de usuario que van a ser desarrolladas en el sprint. En este caso se decidió que se tratará del nodo de la cámara.
- Sprint: Las historias de usuario que forman parte de la pila del sprint son: HU11.
- Reunión de revisión: En un primer momento resultó complicado establecer la conexión con la cámara del robot, pero finalmente se solucionó el problema y se completaron las historias de usuario de la pila del sprint.

- **Sprint 7:** 02/05/2022 – 15/05/2022

  Participantes: Scrum Master y Equipo de desarrollo (excepto Tester).

  - Reunión de planificación: Durante esta reunión se deciden cuáles serán las historias de usuario que van a ser desarrolladas en el sprint. En este caso se decidió que se tratará de la funcionalidad de la cámara.
  - Sprint: Las historias de usuario que forman parte de la pila del sprint son: HU12 y HU13.
  - Reunión de revisión: La conexión entre el nodo de la cámara y la aplicación en un primer momento no se realizaba de forma exitosa. Posteriormente, cuando este problema fue solucionado, la imagen no se mostraba en la aplicación de forma correcta, pero finalmente pudieron completarse las historias de usuario contenidas en la pila del sprint de forma exitosa.

- **Sprint 8:** 16/05/2022 – 29/05/2022

  Participantes: Scrum Master y Equipo de desarrollo.

  - Reunión de planificación: Durante esta reunión se deciden cuáles serán las historias de usuario que van a ser desarrolladas en el sprint. En este caso se decidió que se tratará de testeo del sistema.
  - Sprint: Las historias de usuario que forman parte de la pila del sprint son: HU14.
  - Reunión de revisión: En este sprint se retocaron sobre todo apartados relacionados con una mejora en la interfaz de la aplicación y eliminación y limpieza de código sobrante.

- **Documentación del proyecto:** 30/05/2022 – 30/06/2022
- **Revisión de la documentación:** 01/07/2022 – 08/07/2022

2.3 Gestión de recursos

Dentro de este apartado se van a listar y asignar los diferentes recursos necesarios para llevar a cabo el correcto desarrollo del proyecto.

2.3.1 Especificación de recursos

Como bien se había mencionado, los diferentes recursos a utilizar en este proyecto son:

- Humanos:
  - Scrum Master
  - Equipo de desarrollo: Formado por un desarrollador de software junior, un diseñador gráfico y un tester.
- Físicos:
  - Robot TIAGo
  - Un ordenador para cada integrante del equipo
  - Oficina:
    - Luz
    - Internet
    - Alquiler

El gasto económico total, como se indicó en el apartado del presupuesto rondaría los 118.622,60 €.

2.3.2 Asignación de recursos

En primer lugar, a cada miembro del equipo se le asignará un ordenador y un espacio en la oficina para que puedan desempeñar su trabajo de forma correcta. El robot, por otro lado, será de uso compartido entre todos los miembros del equipo de desarrollo, pero principalmente será usado por el desarrollador de software y el tester ya que será utilizado mayoritariamente para realizar pruebas y comprobar que la aplicación funciona de forma correcta.

2.4 Gestión de riesgos

En este apartado del documento se resaltan los diferentes riesgos que pueden darse durante el desarrollo del proyecto para, de esta manera, saber cómo actuar ante ellos en caso de que acontezcan.

2.4.1 Identificación de riesgos

Los riesgos pueden clasificarse en tres tipos: externos, de implementación o de planificación:

- Riesgos externos
  - Problemas con el robot TIAGo
  - Cambios en las versiones de los entornos utilizados
  - Fallo del controlador de versiones GitHub
- Riesgos de implementación
  - Incompatibilidad de librerías
  - Fallos en la aplicación móvil

- Riesgos de planificación
  - Incumplimiento de objetivos
  - Desfase temporal respecto a la planificación
  - Pérdida de datos o información

2.4.2 Análisis de riesgos

En este punto se van a analizar los diferentes riesgos listados en el apartado anterior a través de la siguiente tabla:

ID	Riesgo	Probabilidad	Impacto	Causa	Solución
R1	Problemas con el robot TIAGo	Baja	Alto	Defectos de fábrica, golpes, mal funcionamiento o deterioro	Tratar de contactar con el servicio técnico y, en caso necesario valorar una posible reparación
R2	Cambios en las versiones de los entornos utilizados	Media	Bajo	Cambios de versiones durante el desarrollo del proyecto	Revisar la documentación, valorar en qué afecta al proyecto y decidir si proceder con la actualización o no
R3	Fallo del controlador de versiones GitHub	Baja	Alto	Fallo en los servidores	Contactar con la empresa y esperar a la solución del problema
R4	Incompatibilidad de librerías	Media	Medio	Modificación o eliminación de las librerías utilizadas por terceros	Buscar y proponer una nueva librería
R5	Fallos en la aplicación móvil	Media	Alto	Confusiones programando o problemas externos	Tratar de encontrar dónde se ha producido el error y trabajar para solucionarlo

R6	Incumplimiento de objetivos	Media	Alto	Falta de organización o desconocimiento de los mismos	Evaluación diaria del trabajo realizado y objetivos cumplidos
R7	Desfase temporal respecto a la planificación	Media	Alto	Problemas de organización del equipo o problemas externos	Replanificar y adaptarse a los nuevos plazos establecidos
R8	Pérdida de datos o información	Baja	Bajo	Pérdida de documentación o datos sobre el proyecto	Rehacer los documentos perdidos e intentar recuperar los datos

Tabla 2.18 Análisis de riesgos

2.5 Legislación y normativa

Dentro de este apartado se mencionan todas las leyes y normas que se deben aplicar y respetar a la hora de desarrollar el proyecto. En este caso, el proyecto se basa en el desarrollo de una aplicación móvil, por tanto, debe cumplir, en primer lugar, con el *Real Decreto 1112/2018, de 7 de septiembre, sobre accesibilidad de los sitios web y aplicaciones para dispositivos móviles del sector público* [10], este Real Decreto, tiene como objeto mejorar el funcionamiento del mercado interior, aproximar las disposiciones legales, reglamentarias y administrativas de los Estados miembros relativas a los requisitos de accesibilidad como conjunto de principios y técnicas que se deben respetar a la hora de diseñar, construir, mantener u actualizar los sitios web y las aplicaciones para dispositivos móviles. En este caso, este Real Decreto está más dirigido a aplicaciones para dispositivos móviles que sean desarrolladas para entidades del sector público, pero trata temas como los tiempos de cargas o la presentación de informes que pueden ser aplicables al desarrollo de este proyecto.

Por otro lado, ya que la aplicación móvil desarrollada pueda tratar en algún momento con datos personales de los usuarios que la utilicen, deberá respetar también la *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos y Garantía de los Derechos Digitales (LOPDGDD)* [11], en esta ley se explica cómo deben tratarse y hacerse uso de los datos personales de los usuarios recopilados a través de la aplicación móvil respetando así el derecho fundamental de las personas físicas a la protección de datos personales, amparado por el artículo 18.4 de la Constitución española [12], que dice lo siguiente:

Artículo 18.4. La ley limitará el uso de la informática para garantizar el honor y la intimidad personal y familiar de los ciudadanos y el pleno ejercicio de sus derechos.

3 Solución

En este apartado, se abordan los temas referentes al producto final desarrollado tratando temas como la funcionalidad final, las tecnologías utilizadas o cómo ha sido el proceso de desarrollo.

3.1 Descripción de la solución

El proyecto desarrollado consta de una aplicación para dispositivos móviles con sistema operativo Android que permita controlar un robot a través de la misma de forma totalmente inalámbrica, sencilla e intuitiva. Las principales funcionalidades de la aplicación móvil es la de permitir la tele operación con el robot, tanto a través de su interfaz gráfica como mediante comandos de voz, en entornos controlados y mapeados. La aplicación también permite enviar al robot a cualquiera de los puntos o habitaciones configuradas con anterioridad también haciendo uso tanto de la interfaz gráfica de la aplicación como de los comandos de voz. Por último, otra funcionalidad con la que cuenta la aplicación móvil desarrollada es la posibilidad de monitorizar lo que el robot está visualizando a través de su cámara dentro de la propia aplicación. Además, la imagen visualizada por el robot previamente a ser enviada a la aplicación móvil para poder ser visualizada desde la misma, es tratada por una red neuronal que permite detectar algunos de los objetos que se encuentren en su campo de visión, añadiendo así una mayor información al usuario que utilice la aplicación.

La aplicación móvil cuenta con cuatro pantallas diferentes que pueden ser accedidas a través de un menú colocado en la parte inferior:

- La primera de las pantallas es la que ofrece la posibilidad de tele operar con el robot: muestra un botón que permite que el robot avance hacia adelante, otro hacia atrás, otro para que avance a la derecha y otro para la izquierda, además de contar con un botón de stop que hace que el robot se detenga por completo.
- La segunda pantalla es la encargada de la navegación, en ella simplemente se muestran los diferentes puntos configurados a los que el robot puede ser enviado, pulsando en cada uno de ellos el robot viajará de forma totalmente autónoma hacia el punto seleccionado.
- La tercera pantalla es la perteneciente a los comandos de voz, en ella aparece un micrófono que, al ser pulsado, comienza a escuchar y reconocer los comandos que el usuario le indique.
- Por último, la cuarta pantalla es la referente a la cámara, en ella simplemente se permite visualizar las imágenes obtenidas por la cámara del robot previamente procesadas por la red neuronal mencionada anteriormente.

Para el correcto funcionamiento de la solución propuesta, a parte de la aplicación móvil se han desarrollado también dos nodos que corren sobre el ordenador que

esté conectado al robot. El primero de estos nodos es el encargado de recibir y ejecutar las acciones propuestas por el usuario de la aplicación móvil referentes a la navegación y tele operación. Por otro lado, el segundo nodo es el encargado de capturar las imágenes de la cámara del robot, pasarla por la red neuronal y enviar la imagen ya tratada a la aplicación móvil para que pueda ser mostrada en la misma.

3.2 El proceso de desarrollo

A mayores de la metodología ágil Scrum explicada con anterioridad, se ha seguido un proceso de desarrollo clásico, el cual está dividido en diferentes fases entre las que se encuentra el análisis de requisitos, el diseño del sistema y las pruebas.

3.2.1 Análisis

Dentro de esta sección se enumeran de forma detallada, todos los requisitos que fueron definidos y que, por tanto, debe cumplir el proyecto finalmente desarrollado. De esta forma, cualquier solución desarrollada que no cumpliera cualquiera de estos requisitos no podría darse por válida. En este caso, la especificación de los requisitos está basada en el estándar IEEE 830-1998 [13].

3.2.1.1 Definición de requisitos

En este apartado se van a listar todos los requisitos que la aplicación móvil desarrollada debe cumplir. Se encuentran divididos en tres tipos: de usuario, funcionales y no funcionales.

Requisitos de usuario:

- **RU1: Tele operación.**
  La aplicación debe permitir poder tele operar con el robot a través de la misma.

- **RU2: Navegación.**
  El sistema debe permitir enviar al robot de forma autónoma a los diferentes puntos previamente configurados dentro del entorno mapeado.

- **RU3: Comandos de voz.**
  La aplicación móvil debe permitir reconocer diferentes comandos de voz a través de los cuales se pueda tener el control de la navegación y la tele operación del robot.

- **RU4: Visualización de la cámara.**
  El sistema desarrollado debe poder mostrar lo que el robot esté visualizando en cada momento a través de su cámara.

- **RU5: Interfaz de usuario.**
  El proyecto contará con una interfaz de usuario que muestre las diferentes opciones que la aplicación permite realizar, como la tele operación, la navegación, el reconocimiento de los comandos de voz y la visualización de la cámara. Además, debe tratarse de una interfaz intuitiva y accesible para todo tipo de usuarios.

Requisitos funcionales:

- **RF1: Sistema modular.**
  El proyecto debe de tratarse de un sistema modular que permita, en un futuro, poder ampliar las funcionalidades y resolver los problemas detectados de una manera sencilla.

- **RF2: Comunicación inalámbrica.**
  La aplicación debe poder conectarse con los nodos que se ejecutan en el ordenador, y, por ende, con el robot, siempre y cuando, tanto el robot, en caso de que contara con la posibilidad de conectarse a una, como los nodos y la aplicación se encuentren dentro de la misma red.

- **RF3: Tecnología ROS2.**
  El sistema debe ser poder capaz de interactuar con cualquier tipo de robot que utilice como sistema operativo ROS2. También debería poder interactuar con robots que utilicen el sistema operativo ROS mediante el uso de un bridge que se ejecute en el ordenador en el que estén corriendo los nodos y transforme las instrucciones de ROS2 a ROS para que el robot pueda entenderlas y, por tanto, ejecutarlas de manera correcta.

- **RF4: Una pantalla para cada funcionalidad.**
  La aplicación deberá estar dividida en diferentes pantallas que puedan ser accedidas a través de un menú situado en la parte inferior de la pantalla de manera que, cada una de las pantallas, muestre la información necesaria y la interacción con el usuario sea más fluida e intuitiva.

- **RF5: Remote control.**
  Cuando el usuario acceda a esta opción en el menú inferior de la aplicación se le mostrará una pantalla en la que aparecerá el mando a control remoto que le permita tele operar con el robot.

- **RF6: Navigation.**
  Opción que se mostrará en el menú de la parte inferior de la aplicación que, tras ser pulsado, se acceda a una pantalla en la que aparezcan los diferentes lugares configurados con anterioridad hacia los que se pueda enviar al robot de manera autónoma.

- **RF7: Mic.**
  Será una de las opciones que aparezcan en el menú de la parte inferior de la aplicación y que permita al usuario acceder a la pantalla que le permita interactuar con el robot a través de comandos de voz.

- **RF8: Camera.**
  Última de las opciones que aparecerán en el menú inferior de la aplicación a través de la cual se podrá acceder a la pantalla en la que se pueda visualizar la imagen que visualiza el robot a través de su cámara.

- **RF9: Flecha adelante.**
  Botón que se aparecerá en la pantalla Remote Control y que, al ser pulsado, hará que el robot se desplace hacia adelante, en caso de que el robot ya se estuviera desplazando en esa dirección incrementará su velocidad.

- **RF10: Flecha atrás.**
  Botón que se aparecerá en la pantalla Remote Control y que, al ser pulsado, hará que el robot se desplace hacia atrás, en caso de que el robot ya se estuviera desplazando en esa dirección incrementará su velocidad.

- **RF11: Flecha derecha.**
  Botón que se aparecerá en la pantalla Remote Control y que, al ser pulsado, hará que el robot gire hacia la derecha, en caso de que el robot ya estuviera girando en esa dirección incrementará su velocidad.

- **RF12: Flecha izquierda.**
  Botón que se aparecerá en la pantalla Remote Control y que, al ser pulsado, hará que el robot gire hacia la izquierda, en caso de que el robot ya estuviera girando en esa dirección incrementará su velocidad.

- **RF13: Botón STOP.**
  El usuario presionará este botón que se encontrará dentro de la pantalla de Remote Control si desea que el robot se detenga por completo.

- **RF14: Pictogramas.**
  Dibujos que aparecerán en la pantalla Navigation y que serán representativos de los diferentes lugares configurados anteriormente hacia los que el robot pueda dirigirse dentro del entorno mapeado. Una vez sean pulsados, el robot se dirigirá hacia la zona que representen.

- **RF15: Botón LISTEN.**
  Se trata de un botón que se encuentra en la pantalla Mic que, tras ser pulsado, el dispositivo móvil en el que se esté ejecutando la aplicación, comenzará a escuchar a través de su micrófono los diferentes comandos que el usuario quiera ordenar al robot para que este ejecute.

- **RF16: Visualizar cámara.**
  Dentro de la pantalla Camera aparecerá un recuadro este recuadro en el que se mostrará la imagen, tras haber sido analizada por la red neuronal, que el robot está visualizando a través de su cámara en ese momento.

- **RF17: Topic nodo cámara.**
  La imagen tras haber sido tratada por la red neuronal en el nodo referente a la cámara se publicará en un topic de ROS2 de manera que pueda ser visualizada como cualquier otro topic y no únicamente desde la aplicación móvil.

- **RF18: Especificaciones del hardware.**
  El sistema está formado por diferentes dispositivos hardware que deben estar conectados entre sí y que deben contar con unas especificaciones concretas cada uno de ellos para el correcto funcionamiento del mismo. Estas especificaciones se muestran en la siguiente tabla.

Dispositivo	Requisitos
Ordenador	• Sistema operativo Ubuntu 20.04 • Al menos 8 GB de RAM
Dispositivo móvil	• Sistema operativo Android
Robot	Dos opciones: • Sistema operativo ROS2 • Sistema operativo ROS y utilización de un bridge en el ordenador

Tabla 3.1 Requisitos Hardware

- **RF19: Herramientas del software.**
  A continuación, en la siguiente tabla, se muestran las diferentes herramientas software necesarias para el correcto funcionamiento del sistema.

Herramienta	Versión
Android	6.0 o superior
ROS	ROS2 foxy
Python	3.8.10

Tabla 3.2 Requisitos software

Requisitos no funcionales:

- **RN1: Seguridad.**
  La conexión entre la aplicación y los nodos que se ejecutan en el ordenador se realizará mediante sockets de manera que los mensajes no puedan ser capturados por terceros.
- **RN2: Fiabilidad.**

El tiempo de respuesta entre que el usuario ejecute una acción en la aplicación móvil y el robot comience a realizarla debe ser inferior o igual a 5 segundos.

- **RN3: Disponibilidad.**
  La aplicación móvil debe poder ser completamente funcional siempre y cuando el robot y la misma aplicación se encuentren dentro de la misma red.
- **RN4: Mantenibilidad.**
  El equipo de desarrollo, una vez el proyecto haya sido finalizado, realizarán un mantenimiento de la aplicación desarrollada para, de esta manera, poder añadir más funcionalidades en un futuro o mejorar las ya existentes.
- **RN5: Usabilidad.**
  La aplicación desarrollada debe ser usable y accesible por todo tipo de usuarios independientemente de sus conocimientos sobre robótica o informática previos.

3.2.1.2 Especificación de requisitos

En esta sección se van a describir los casos de uso que satisfacen los requisitos expuestos en el anterior apartado, en este caso son los que aparecen el Figura 3.1 representada a continuación. También se mostrarán los diagramas de secuencia referentes a cada uno de ellos para mostrar cómo es la interacción del usuario con cada uno de ellos.

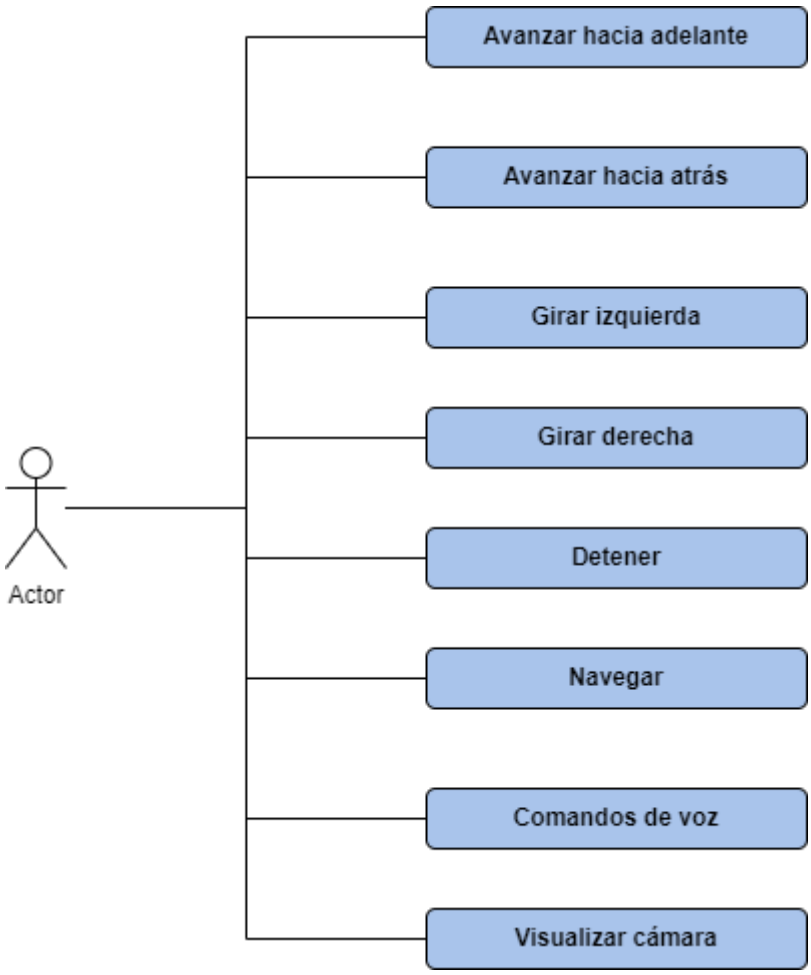


Figura 3.1 Diagrama casos de uso

CU1: Avanzar hacia adelante

Actor principal: Sistema

Personal involucrado e intereses:

El usuario quiere que el robot se desplace hacia adelante, y que, en caso de que ya lo esté haciendo, que aumente la velocidad con la que está realizando el movimiento.

El sistema debe enviarle la señal al robot para que realice la acción requerida por el usuario.

Precondiciones:

El nodo tele operador debe estar ejecutándose en un ordenador que esté conectado al robot y que se encuentre conectado, a su vez, a la misma red que el dispositivo móvil desde el que se está ejecutando la aplicación.

Postcondiciones:

El robot comienza a desplazarse hacia adelante y, en caso de que ya se encontrara desplazándose en esa dirección aumentará la velocidad con la que realiza el movimiento.

Flujo básico:

Actor	**Sistema**
1. El usuario abre la aplicación móvil.	
2. El usuario accede al menú “Remote control".	
3. El usuario pulsa la flecha que apunta hacia arriba.	
	4. El sistema captura la pulsación.
	5. A través del socket abierto con el nodo tele operador el sistema envía el mensaje “UP” al mismo.
	6. El nodo recibe el mensaje
	7. El nodo envía los parámetros necesarios al topic responsable de la tele operación del robot para que ejecute la acción de desplazarse hacia adelante.

Flujo alternativo:

5a. El socket con el nodo no está abierto.

1. La aplicación abre un socket con el nodo tele operador.

5b. El nodo tele operador no está ejecutándose.

1. La aplicación no podrá abrir el socket con el nodo tele operador.
2. La acción solicitada no podrá ejecutarse.

5c. El ordenador no está conectado a la misma red que el dispositivo móvil.

1. El nodo no podrá recibir el mensaje.
2. La acción solicitada no podrá ejecutarse.

7a. El ordenador no se encuentra conectado al robot.

1. El nodo no podrá enviar el mensaje al topic del robot.
2. La ejecución del nodo se detendrá.

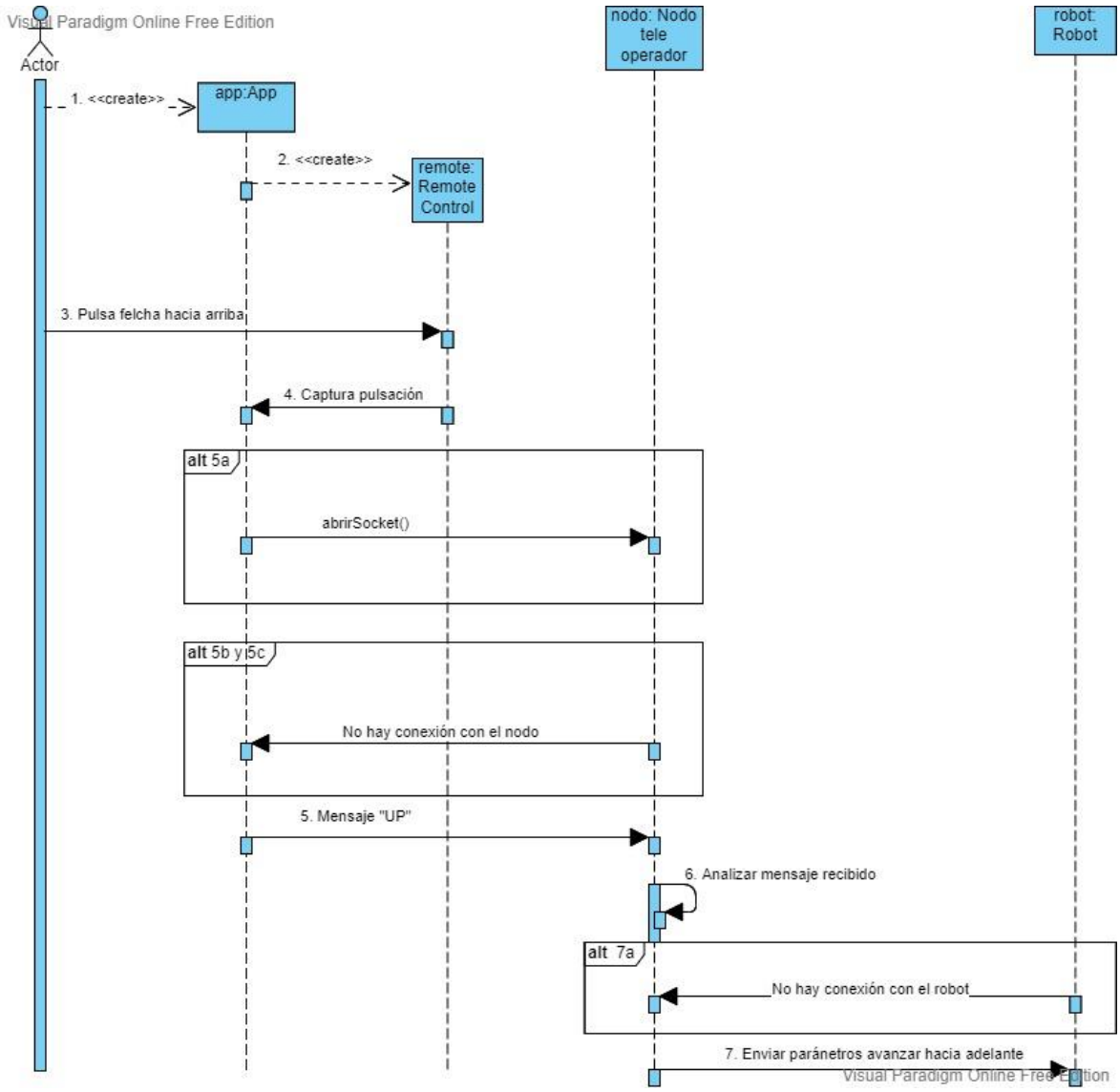


Figura 3.2 Diagrama secuencia CU1: Avanzar hacia adelante

CU2: Avanzar hacia atrás

Actor principal: Sistema

Personal involucrado e intereses:

El usuario quiere que el robot se desplace hacia atrás, y que, en caso de que ya lo esté haciendo, que aumente la velocidad con la que está realizando el movimiento.

El sistema debe enviarle la señal al robot para que realice la acción requerida por el usuario.

Precondiciones:

El nodo tele operador debe estar ejecutándose en un ordenador que esté conectado al robot y que se encuentre conectado, a su vez, a la misma red que el dispositivo móvil desde el que se está ejecutando la aplicación.

Postcondiciones:

El robot comienza a desplazarse hacia atrás y, en caso de que ya se encontrara desplazándose en esa dirección aumentará la velocidad con la que realiza el movimiento.

Flujo básico:

Actor	**Sistema**
1. El usuario abre la aplicación móvil.	
2. El usuario accede al menú “Remote control".	
3. El usuario pulsa la flecha que apunta hacia abajo.	
	4. El sistema captura la pulsación.
	5. A través del socket abierto con el nodo tele operador el sistema envía el mensaje “DOWN” al mismo.
	6. El nodo recibe el mensaje
	7. El nodo envía los parámetros necesarios al topic responsable de la tele operación del robot para que ejecute la acción de desplazarse hacia atrás.

Flujo alternativo:

5a. El socket con el nodo no está abierto.

1. La aplicación abre un socket con el nodo tele operador.

5b. El nodo tele operador no está ejecutándose.

1. La aplicación no podrá abrir el socket con el nodo tele operador.
2. La acción solicitada no podrá ejecutarse.

5c. El ordenador no está conectado a la misma red que el dispositivo móvil.

1. El nodo no podrá recibir el mensaje.
2. La acción solicitada no podrá ejecutarse.

7a. El ordenador no se encuentra conectado al robot.

1. El nodo no podrá enviar el mensaje al topic del robot.
2. La ejecución del nodo se detendrá.

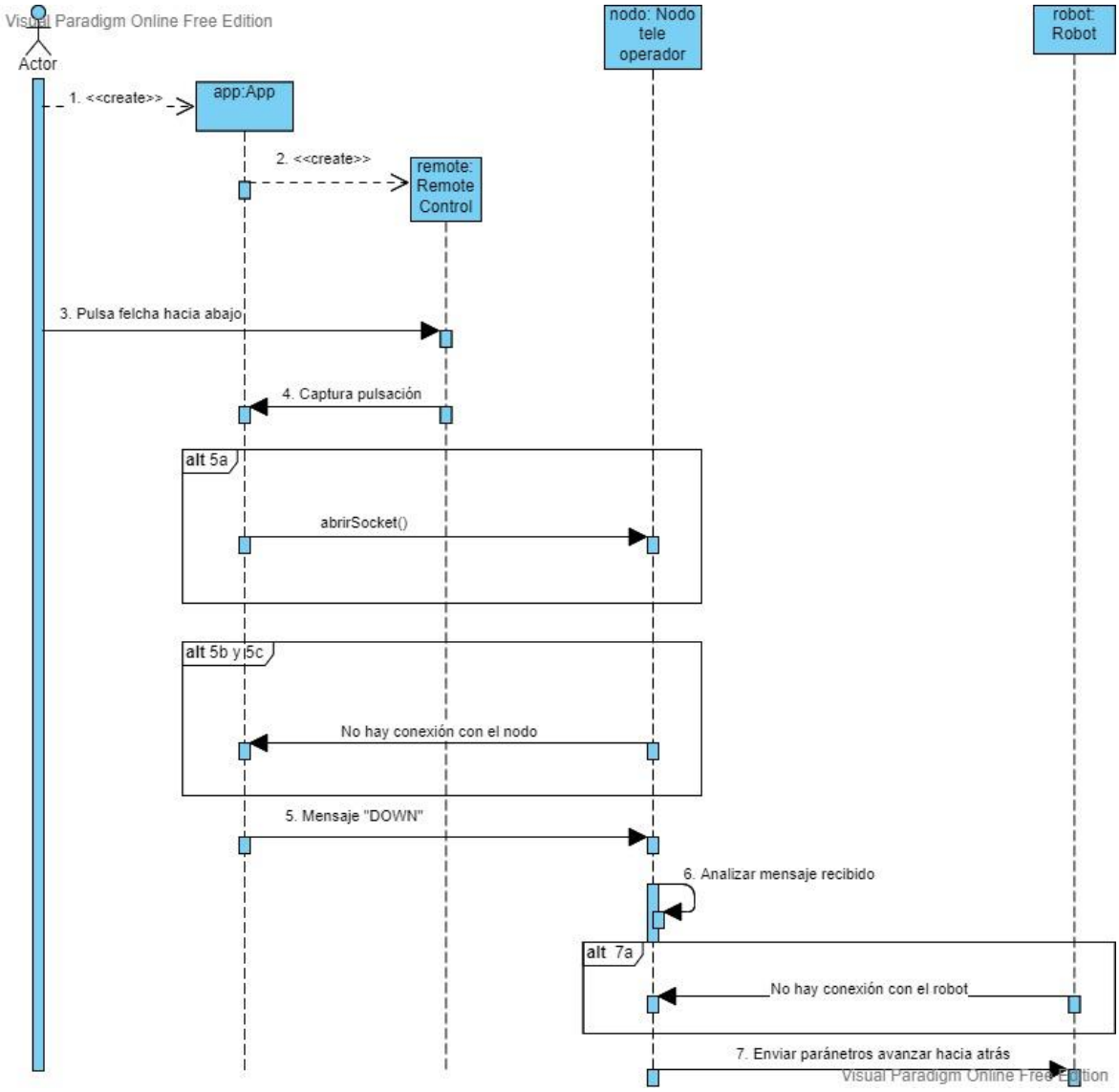


Figura 3.3 Diagrama secuencia CU2: Avanzar hacia atrás

CU3: Girar izquierda

Actor principal: Sistema

Personal involucrado e intereses:

El usuario quiere que el robot gire a la izquierda, y que, en caso de que ya lo esté haciendo, que aumente la velocidad con la que está realizando el movimiento.

El sistema debe enviarle la señal al robot para que realice la acción requerida por el usuario.

Precondiciones:

El nodo tele operador debe estar ejecutándose en un ordenador que esté conectado al robot y que se encuentre conectado, a su vez, a la misma red que el dispositivo móvil desde el que se está ejecutando la aplicación.

Postcondiciones:

El robot comienza a girar a la izquierda y, en caso de que ya se encontrara girando en esa dirección aumentará la velocidad con la que realiza el movimiento.

Flujo básico:

Actor	**Sistema**
1. El usuario abre la aplicación móvil.	
2. El usuario accede al menú “Remote control".	
3. El usuario pulsa la flecha que apunta hacia la izquierda.	
	4. El sistema captura la pulsación.
	5. A través del socket abierto con el nodo tele operador el sistema envía el mensaje “LEFT” al mismo.
	6. El nodo recibe el mensaje
	7. El nodo envía los parámetros necesarios al topic responsable de la tele operación del robot para que ejecute la acción de girar a la izquierda.

Flujo alternativo:

5a. El socket con el nodo no está abierto.

1. La aplicación abre un socket con el nodo tele operador.

5b. El nodo tele operador no está ejecutándose.

1. La aplicación no podrá abrir el socket con el nodo tele operador.
2. La acción solicitada no podrá ejecutarse.

5c. El ordenador no está conectado a la misma red que el dispositivo móvil.

1. El nodo no podrá recibir el mensaje.
2. La acción solicitada no podrá ejecutarse.

7a. El ordenador no se encuentra conectado al robot.

1. El nodo no podrá enviar el mensaje al topic del robot.
2. La ejecución del nodo se detendrá.

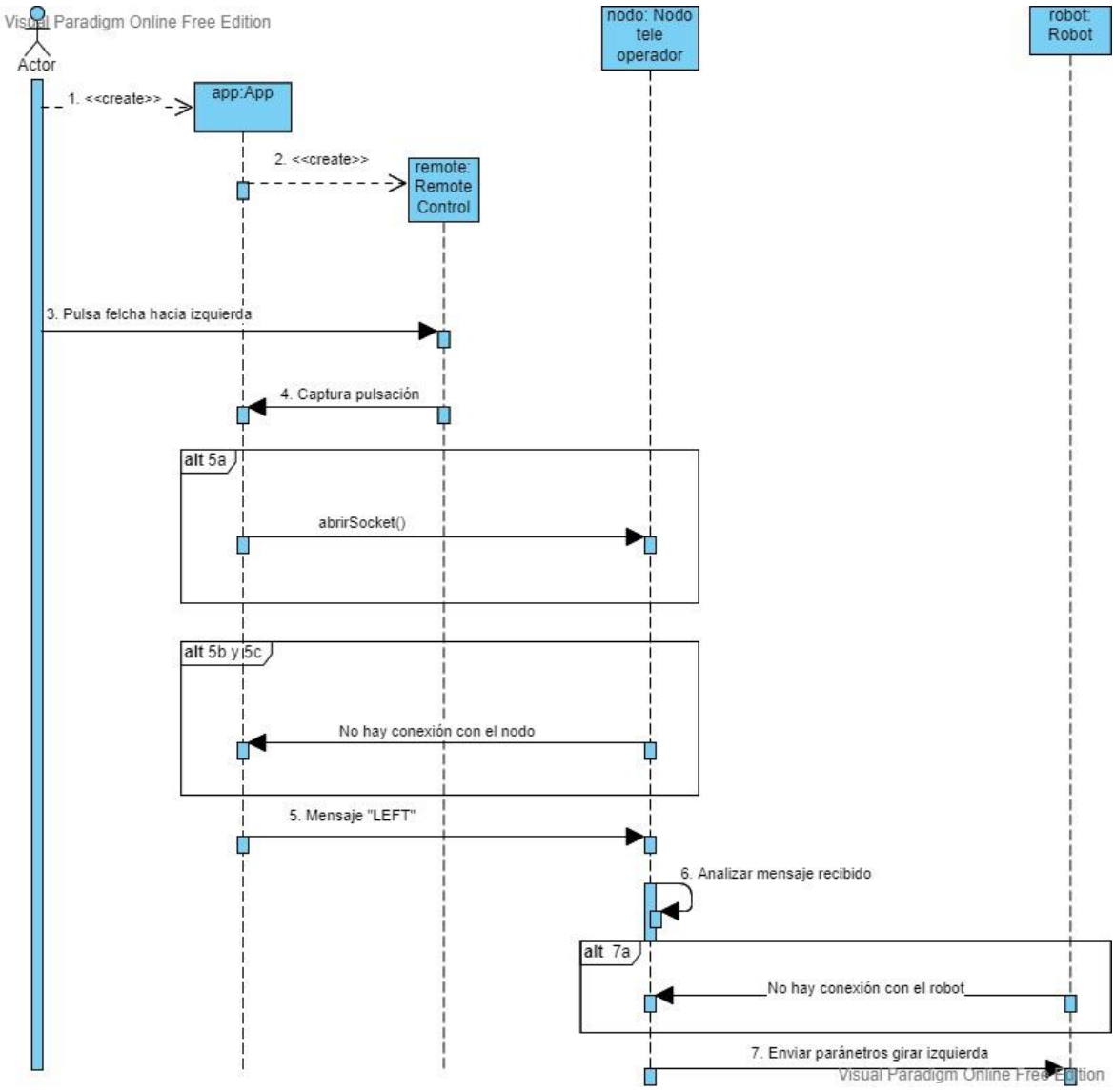


Figura 3.4 Diagrama secuencia CU3: Girar izquierda

CU4: Girar derecha

Actor principal: Sistema

Personal involucrado e intereses:

El usuario quiere que el robot gire a la derecha, y que, en caso de que ya lo esté haciendo, que aumente la velocidad con la que está realizando el movimiento.

El sistema debe enviarle la señal al robot para que realice la acción requerida por el usuario.

Precondiciones:

El nodo tele operador debe estar ejecutándose en un ordenador que esté conectado al robot y que se encuentre conectado, a su vez, a la misma red que el dispositivo móvil desde el que se está ejecutando la aplicación.

Postcondiciones:

El robot comienza a girar a la derecha y, en caso de que ya se encontrara girando en esa dirección aumentará la velocidad con la que realiza el movimiento.

Flujo básico:

Actor	**Sistema**
1. El usuario abre la aplicación móvil.	
2. El usuario accede al menú “Remote control".	
3. El usuario pulsa la flecha que apunta hacia la derecha.	
	4. El sistema captura la pulsación.
	5. A través del socket abierto con el nodo tele operador el sistema envía el mensaje “RIGHT” al mismo.
	6. El nodo recibe el mensaje
	7. El nodo envía los parámetros necesarios al topic responsable de la tele operación del robot para que ejecute la acción de girar a la derecha.

Flujo alternativo:

5a. El socket con el nodo no está abierto.

1. La aplicación abre un socket con el nodo tele operador.

5b. El nodo tele operador no está ejecutándose.

1. La aplicación no podrá abrir el socket con el nodo tele operador.
2. La acción solicitada no podrá ejecutarse.

5c. El ordenador no está conectado a la misma red que el dispositivo móvil.

1. El nodo no podrá recibir el mensaje.
2. La acción solicitada no podrá ejecutarse.

7a. El ordenador no se encuentra conectado al robot.

1. El nodo no podrá enviar el mensaje al topic del robot.
2. La ejecución del nodo se detendrá.

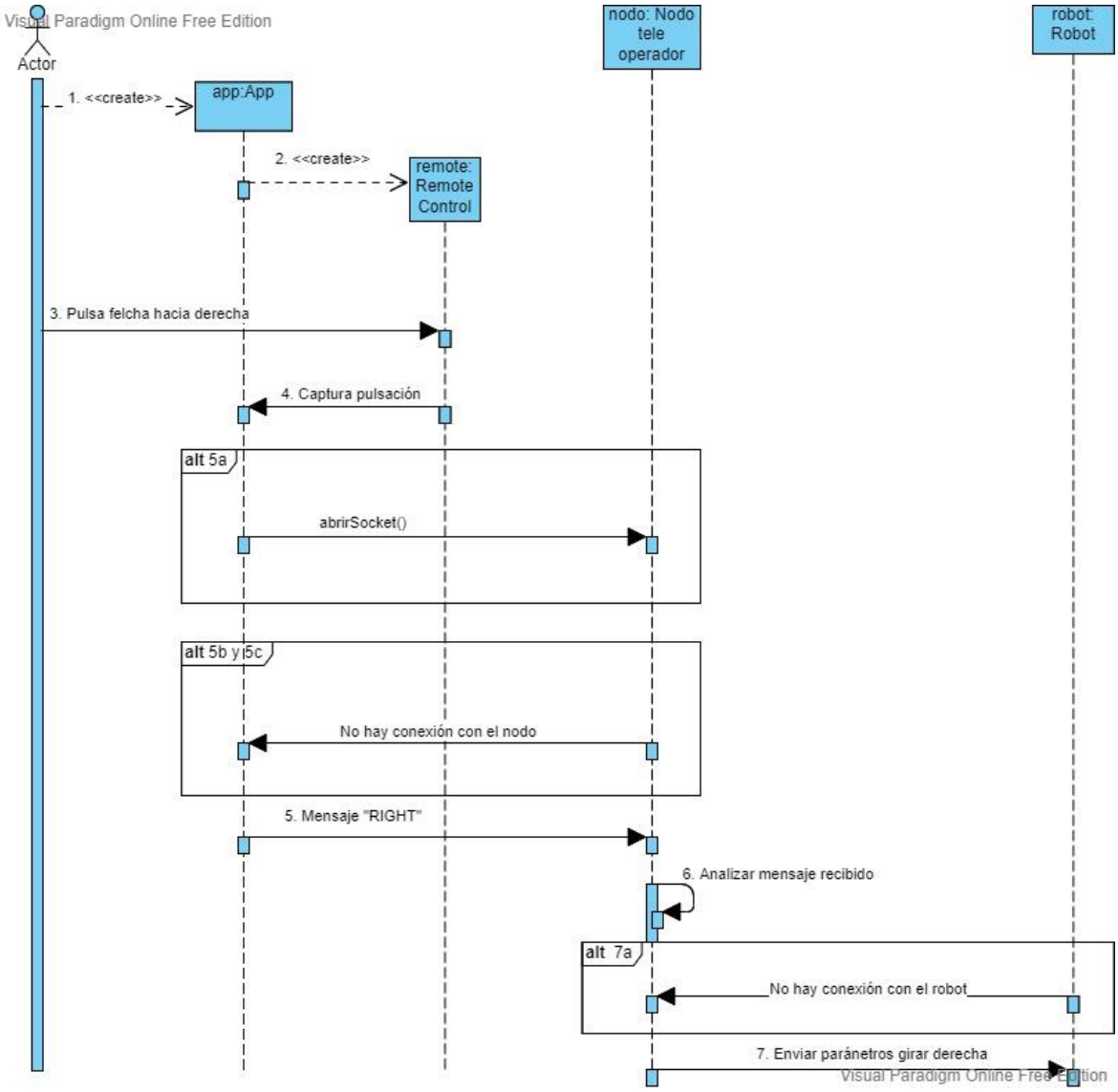


Figura 3.5 Diagrama secuencia CU4: Girar derecha

CU5: Detener

Actor principal: Sistema

Personal involucrado e intereses:

El usuario quiere que el robot se detenga.

El sistema debe enviarle la señal al robot para que realice la acción requerida por el usuario.

Precondiciones:

El nodo tele operador debe estar ejecutándose en un ordenador que esté conectado al robot y que se encuentre conectado, a su vez, a la misma red que el dispositivo móvil desde el que se está ejecutando la aplicación.

Postcondiciones:

El robot se detiene por completo en caso de que estuviera en movimiento, y si ya estaba detenido sigue en ese estado.

Flujo básico:

Actor	**Sistema**
1. El usuario abre la aplicación móvil.	
2. El usuario accede al menú “Remote control".	
3. El usuario pulsa el botón “STOP”.	
	4. El sistema captura la pulsación.
	5. A través del socket abierto con el nodo tele operador el sistema envía el mensaje “STOP” al mismo.
	6. El nodo recibe el mensaje
	7. El nodo envía los parámetros necesarios al topic responsable de la tele operación del robot para que ejecute la acción de detenerse.

Flujo alternativo:

5a. El socket con el nodo no está abierto.

1. La aplicación abre un socket con el nodo tele operador.

5b. El nodo tele operador no está ejecutándose.

1. La aplicación no podrá abrir el socket con el nodo tele operador.
2. La acción solicitada no podrá ejecutarse.

5c. El ordenador no está conectado a la misma red que el dispositivo móvil.

1. El nodo no podrá recibir el mensaje.
2. La acción solicitada no podrá ejecutarse.

7a. El ordenador no se encuentra conectado al robot.

1. El nodo no podrá enviar el mensaje al topic del robot.
2. La ejecución del nodo se detendrá.

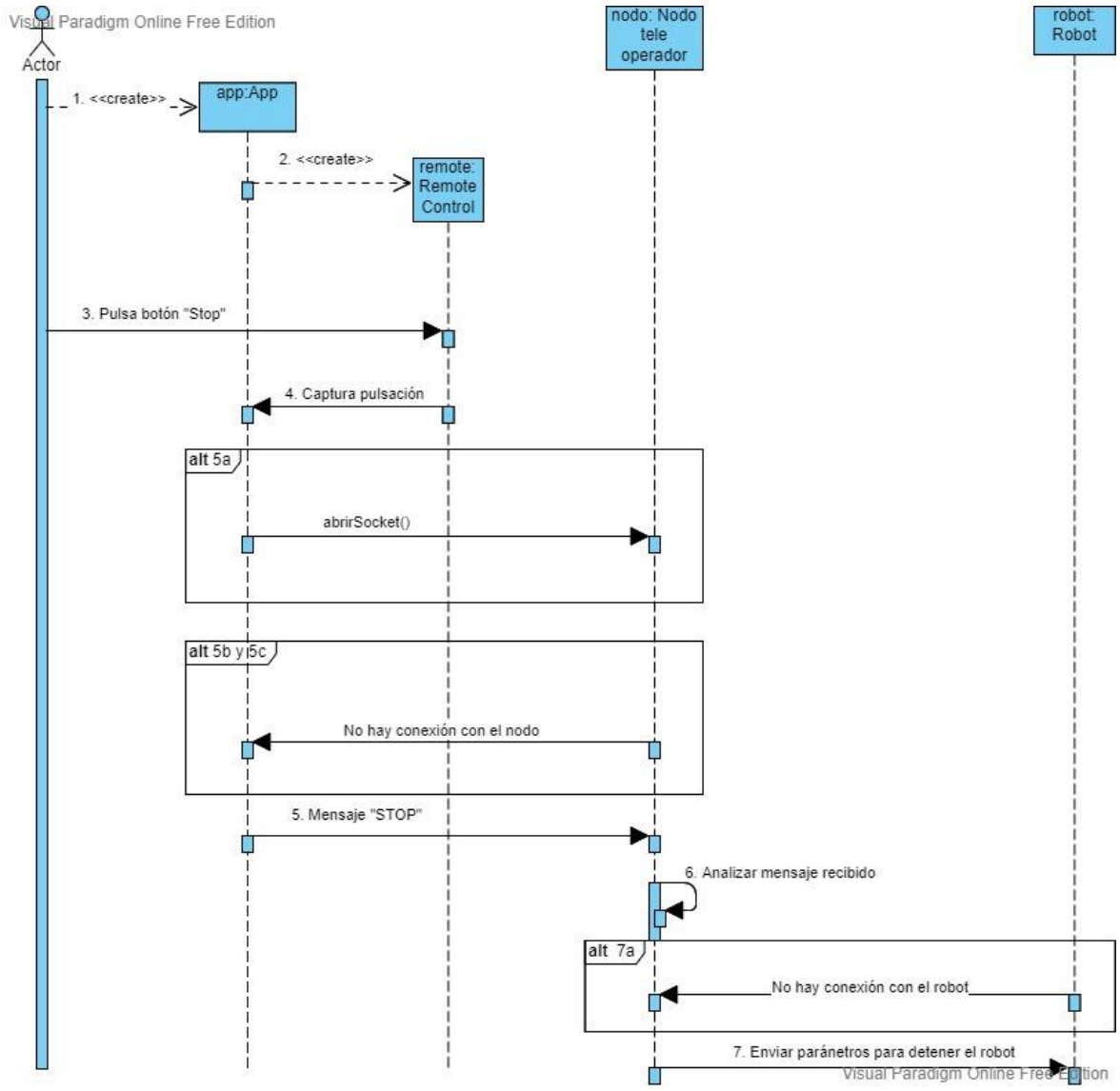


Figura 3.6 Diagrama secuencia CU5: Detener

CU6: Navegar

Actor principal: Sistema

Personal involucrado e intereses:

El usuario quiere que el robot navegue hacia uno de los lugares preconfigurados dentro del entorno mapeado.

El sistema debe enviarle la señal al robot para que realice la acción requerida por el usuario.

Precondiciones:

El nodo tele operador debe estar ejecutándose en un ordenador que esté conectado al robot y que se encuentre conectado, a su vez, a la misma red que el dispositivo móvil desde el que se está ejecutando la aplicación.

Postcondiciones:

El robot se desplaza de forma autónoma hasta el punto o la habitación seleccionada por el usuario en la aplicación hasta detenerse una vez alcance el punto.

Flujo básico:

Actor	**Sistema**
1. El usuario abre la aplicación móvil.	
2. El usuario accede al menú “Navigation".	
3. El usuario selecciona el pictograma correspondiente al punto al que desea que navegue el robot.	
	4. El sistema captura la pulsación.
	5. A través del socket abierto con el nodo tele operador el sistema envía el mensaje configurado para el punto seleccionado al mismo.
	6. El nodo recibe el mensaje
	7. El nodo envía las coordenadas del punto seleccionado al topic encargado de la navegación del robot.

Flujo alternativo:

5a. El socket con el nodo no está abierto.

1. La aplicación abre un socket con el nodo tele operador.

5b. El nodo tele operador no está ejecutándose.

1. La aplicación no podrá abrir el socket con el nodo tele operador.
2. La acción solicitada no podrá ejecutarse.

5c. El ordenador no está conectado a la misma red que el dispositivo móvil.

1. El nodo no podrá recibir el mensaje.
2. La acción solicitada no podrá ejecutarse.

7a. El ordenador no se encuentra conectado al robot.

1. El nodo no podrá enviar el mensaje al topic del robot.

2. La ejecución del nodo se detendrá.

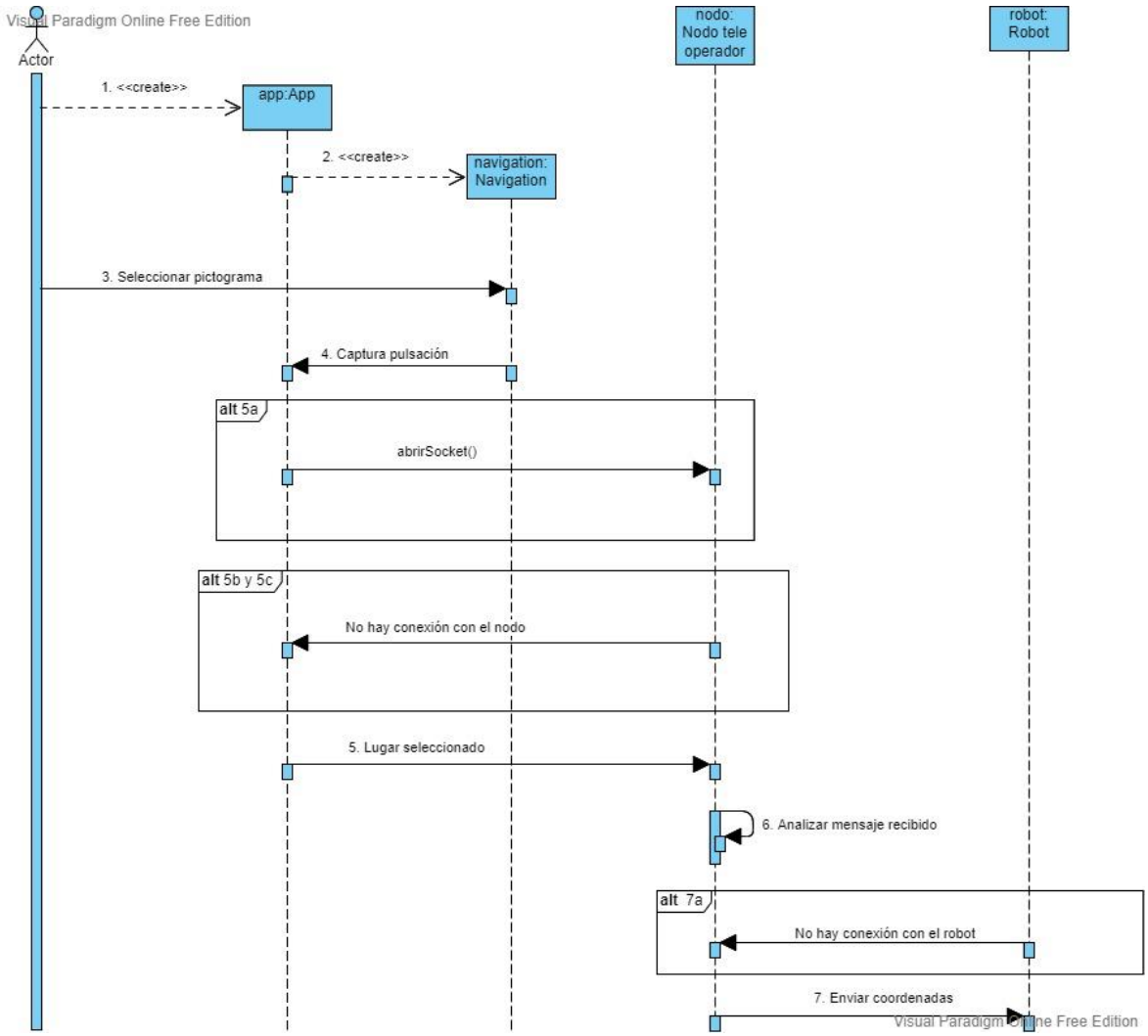


Figura 3.7 Diagrama secuencia CU6: Navegar

CU7: Comandos de voz

Actor principal: Sistema

Personal involucrado e intereses:

El usuario quiere poder controlar el robot a través de alguno de los comandos de voz que la aplicación móvil ofrece.

El sistema debe enviarle la señal al robot para que realice la acción requerida por el usuario.

Precondiciones:

El nodo tele operador debe estar ejecutándose en un ordenador que esté conectado al robot y que se encuentre conectado, a su vez, a la misma red que el dispositivo móvil desde el que se está ejecutando la aplicación.

Postcondiciones:

El robot ejecuta la acción solicitada por el usuario a través del comando de voz, bien sea navegar hacia algún lugar indicado, desplazarse en la dirección indicada o detenerse por completo.

Flujo básico:

Actor	**Sistema**
1. El usuario abre la aplicación móvil.	
2. El usuario accede al menú “Mic".	
3. El usuario pulsa el botón Listen.	
	4. La aplicación muestra un micrófono en la pantalla indicando que el usuario ya puede reproducir el comando deseado.
5. El usuario, a través de su voz, indica el comando que desee.	
	6. El sistema captura y traduce a texto el comando reproducido por el usuario.
	7. A través del socket abierto con el nodo tele operador el sistema envía el mensaje configurado para el punto seleccionado al mismo.
	8. El nodo recibe el mensaje
	9. El nodo envía la información o al topic encargado de la tele operación del robot o al encargado de la navegación en función del comando introducido por el usuario.

Flujo alternativo:

6a. El comando no puede ser reconocido

1. La aplicación no reconoce el comando y no realiza ninguna acción.

7a. El socket con el nodo no está abierto.

1. La aplicación abre un socket con el nodo tele operador.

7b. El nodo tele operador no está ejecutándose.

1. La aplicación no podrá abrir el socket con el nodo tele operador.

2. La acción solicitada no podrá ejecutarse.

7c. El ordenador no está conectado a la misma red que el dispositivo móvil.

1. El nodo no podrá recibir el mensaje.
2. La acción solicitada no podrá ejecutarse.

9a. El ordenador no se encuentra conectado al robot.

1. El nodo no podrá enviar el mensaje al topic del robot.
2. La ejecución del nodo se detendrá.

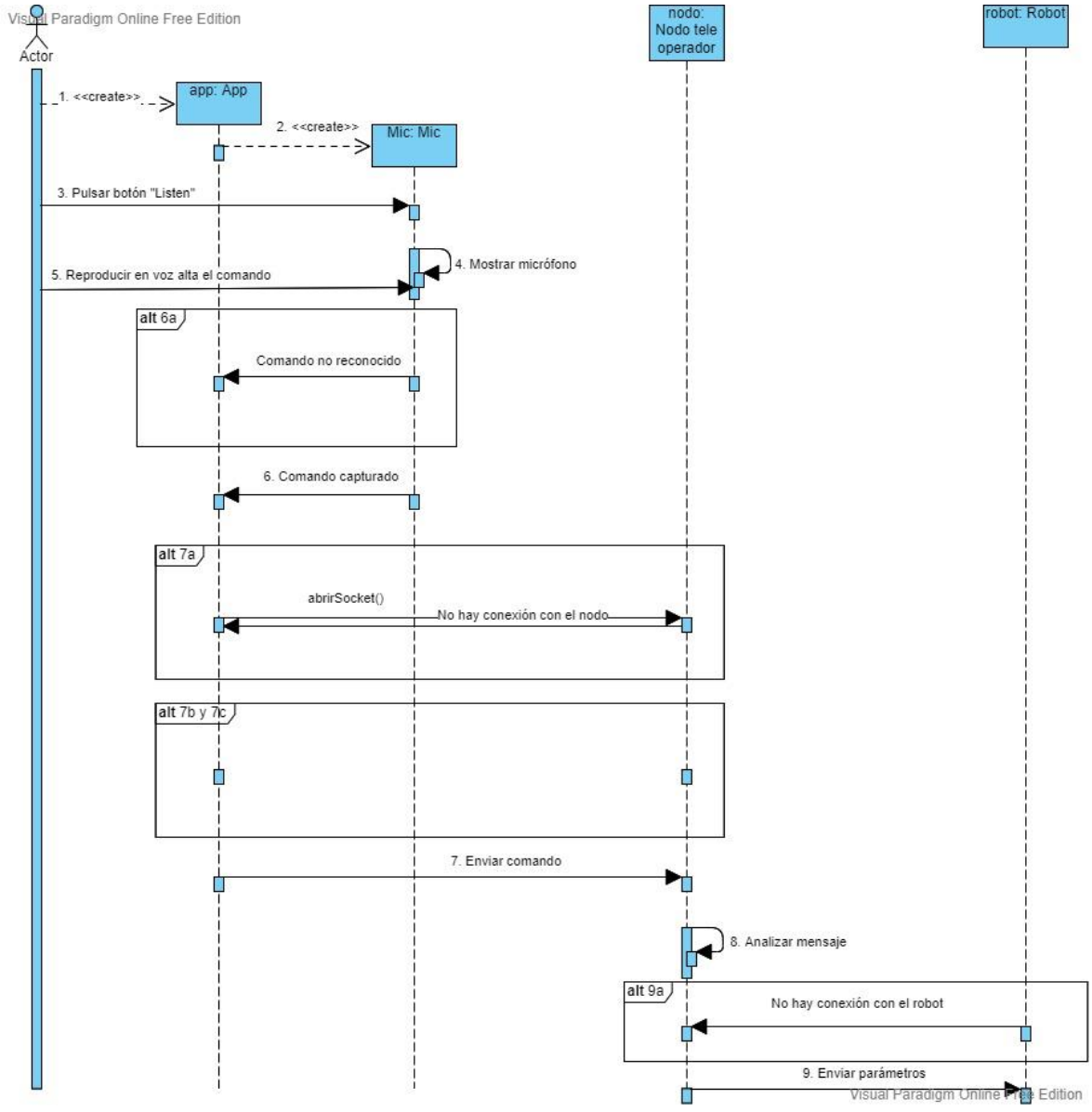


Figura 3.8 Diagrama secuencia CU7: Comandos de voz

CU8: Visualizar cámara

Actor principal: Sistema

Personal involucrado e intereses:

El usuario quiere que el robot se detenga.

El sistema debe enviarle la señal al robot para que realice la acción requerida por el usuario.

Precondiciones:

El nodo tele operador debe estar ejecutándose en un ordenador que esté conectado al robot y que se encuentre conectado, a su vez, a la misma red que el dispositivo móvil desde el que se está ejecutando la aplicación.

Postcondiciones:

El robot se detiene por completo en caso de que estuviera en movimiento, y si ya estaba detenido sigue en ese estado.

Flujo básico:

Actor	**Sistema**
1. El usuario abre la aplicación móvil.	
2. El usuario accede al menú “Camera".	
	3. El sistema captura la pulsación.
	4. A través del socket abierto con el nodo de la cámara el sistema solicita la imagen de la cámara.
	5. El nodo recibe el mensaje.
	6. El nodo solicita la imagen de la cámara al robot a través del topic encargado de esta acción.
	7. El nodo trata la imagen recibida a través de una red neuronal para reconocer algunos de los objetos que aparezcan en la misma.
	8. El nodo envía la imagen ya tratada a la aplicación a través del socket mencionado anteriormente.
	9. La aplicación recibe la imagen y la muestra por pantalla.

Flujo alternativo:

4a. El socket con el nodo no está abierto.

1. La aplicación abre un socket con el nodo tele operador.

4b. El nodo tele operador no está ejecutándose.

1. La aplicación no podrá abrir el socket con el nodo tele operador.
2. La acción solicitada no podrá ejecutarse.

4c. El ordenador no está conectado a la misma red que el dispositivo móvil.

1. El nodo no podrá recibir el mensaje.
2. La acción solicitada no podrá ejecutarse.

6a. El ordenador no se encuentra conectado al robot.

1. El nodo no podrá enviar el mensaje al topic del robot.
2. La ejecución del nodo se detendrá.

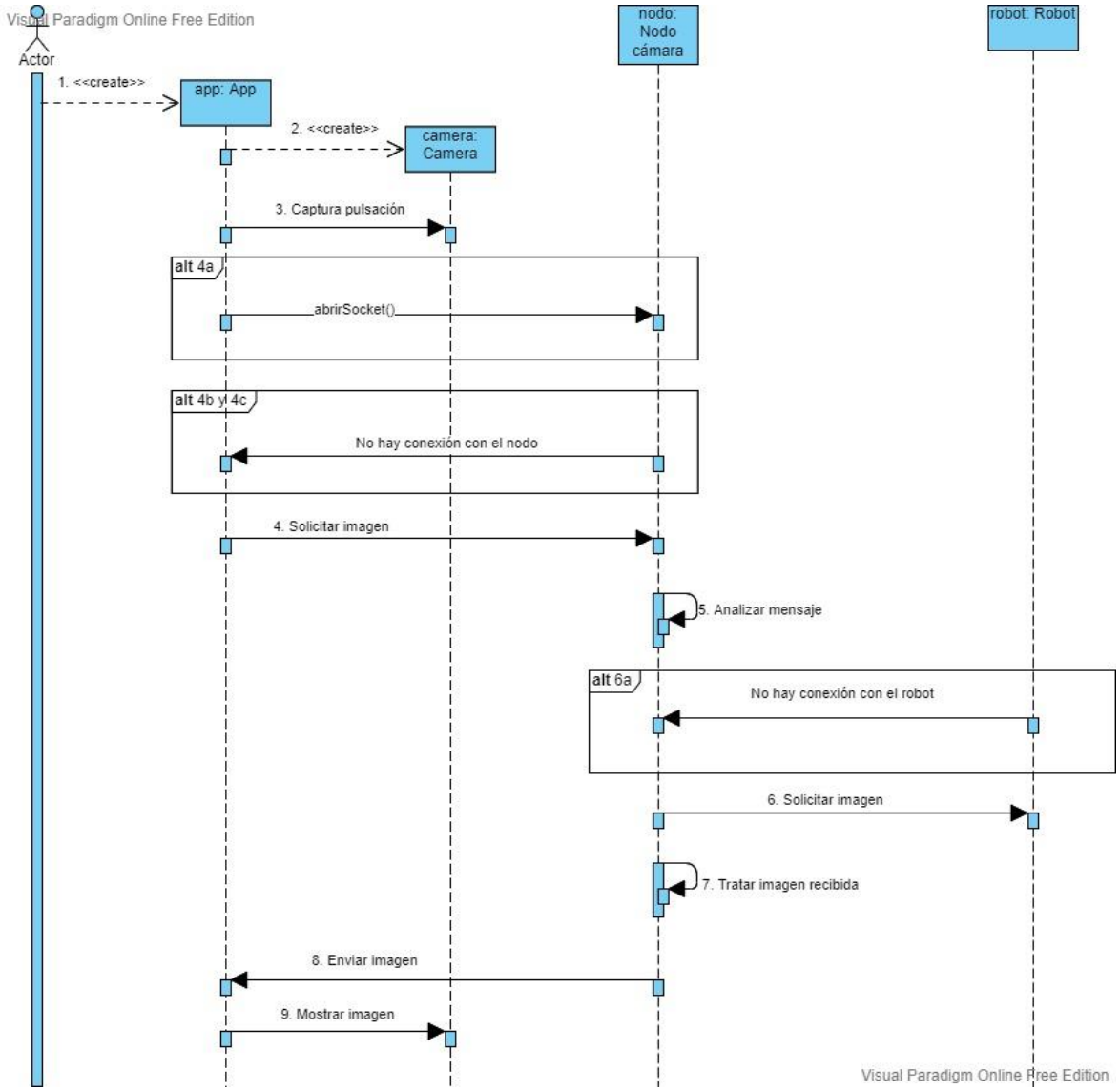


Figura 3.9 Diagrama secuencia CU8: Visualizar cámara

3.2.2 Diseño

En este apartado se definirá cómo está diseñado el sistema, su implementación y las pruebas que se han realizado para validar su correcto funcionamiento.

3.2.2.1 Diseño de sistema

En esta sección, se va a mostrar de forma gráfica, a través de la Figura 3.10 cómo está diseñado el sistema, más específicamente, cuáles son las relaciones que existen entre los componentes hardware y software que constituyen el proyecto.

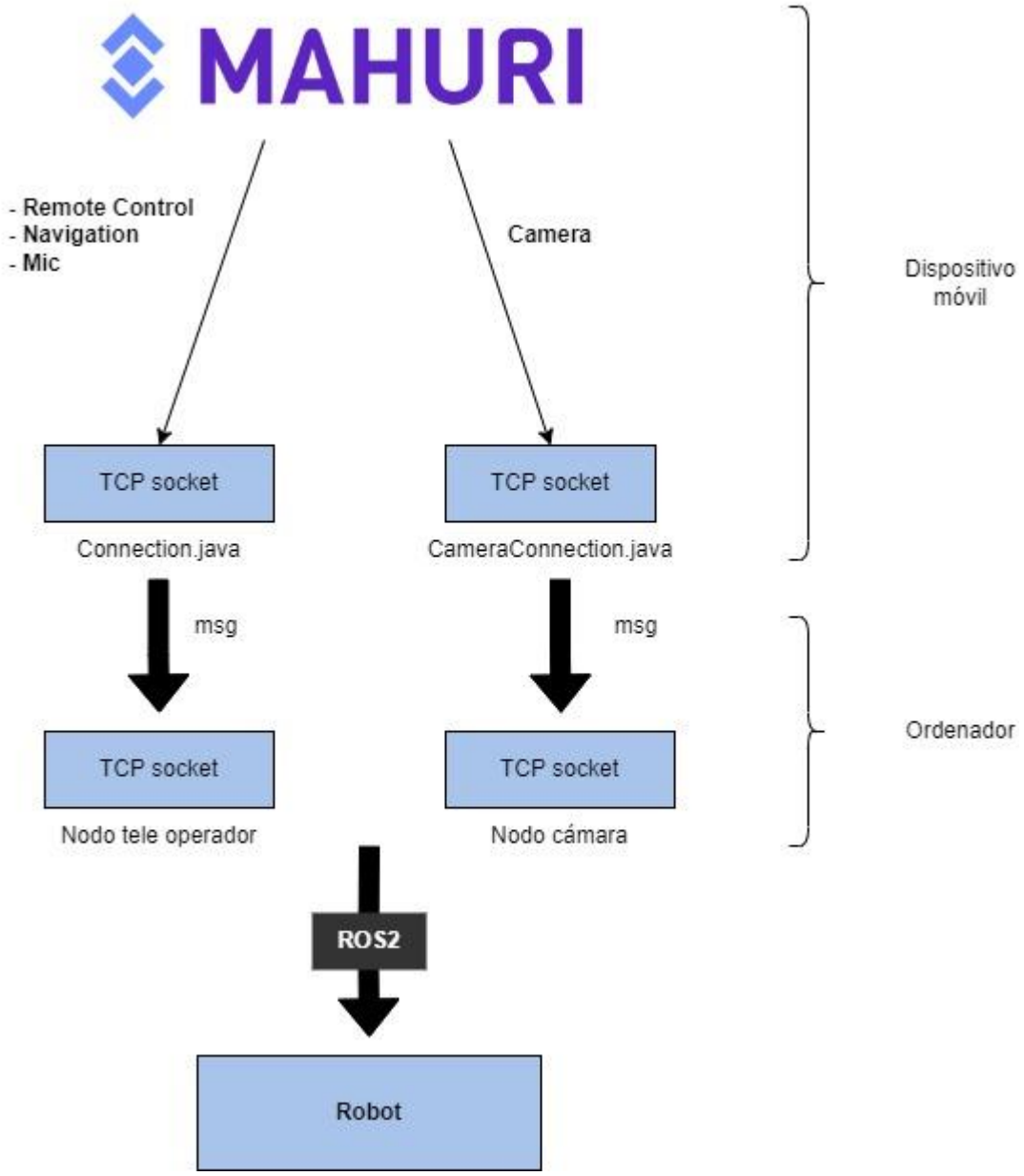


Figura 3.10 Arquitectura del sistema

3.2.2.2 Diseño detallado

La arquitectura, en este caso, como se puede observar en la Figura 3.10, está formada principalmente por tres elementos principales: el dispositivo móvil, el ordenador y el robot. A continuación, se va a describir cada uno de ellos:

- **Dispositivo móvil:** Se trata del dispositivo con sistema operativo Android 6.0 o superior desde el que se ejecuta la aplicación móvil desarrollada, de nombre MAHURI en este caso. Este dispositivo es el encargado de ejecutar la aplicación y enviar las peticiones que el usuario de la misma desee al ordenador a través de los distintos sockets TCP abiertos. Si el usuario se encuentra en las pantallas "Remote control", "Navigation" o "Mic", las peticiones se envían al nodo tele operador a través de la clase Connection.java que se encuentra ejecutándose en el ordenado, mientras que, si el usuario se encuentra en la pantalla "Camera", las peticiones serán enviadas al nodo de la cámara, que también se encuentra ejecutándose en el ordenador, a través de la clase CameraConnection.java.

- **Ordenador:** El ordenador, por otro lado, es el que actúa de intermediario entre el dispositivo móvil en el que se ejecuta la aplicación móvil desarrollada y el robot. Para el correcto funcionamiento del sistema en el ordenador deben estar ejecutándose los dos nodos mencionados anteriormente, el nodo responsable de la tele operación y el responsable de la cámara. Estos nodos son los que establecen la conexión con la aplicación móvil a través de los sockets TCP y, los que posteriormente, envían las peticiones al robot para que realice las acciones requeridas por el usuario. A continuación, se va a explicar brevemente la función de cada uno de estos nodos:
  - Nodo tele operador: Este nodo está compuesto por dos programas escritos en lenguaje C++, uno de ellos simplemente es el encargado de recibir los mensajes enviados por la aplicación a través del socket TCP y enviárselos al segundo programa. Este segundo programa, una vez recibe los mensajes enviados por el primero, se encarga de identificar cada uno de los mensajes recibidos y, en función de qué mensaje sea, enviarle al robot los diferentes parámetros necesarios a través del topic de ROS2 indicado para que este realice las acciones que el usuario haya requerido. La función principal de este nodo es la de tele operar el robot a través de la interfaz gráfica de la aplicación móvil que se encuentra en la pantalla "Remote control" de la misma, o la de hacer que el robot navegue hasta el lugar indicado por el usuario en la pantalla "Navigation". Estas dos acciones, tanto la de tele operar, como la de navegar, también pueden ser realizadas a través de comandos de voz, es por eso que también se llama a este nodo desde la pantalla "Mic".
  - Nodo de la cámara: Este nodo, escrito en lenguaje Python, entra en funcionamiento cuando el usuario accede a la pantalla "Camera" de la aplicación móvil. Para ello, debe estar ejecutándose en el ordenador esperando a que el usuario realice esta petición desde la aplicación móvil. Una vez el usuario accede a esta pantalla, la

aplicación móvil, a través del socket TCP abierto con el nodo, realiza una petición para poder mostrar en la aplicación la imagen captada por la cámara del robot. En ese momento, una vez recibe esta petición, el nodo realiza una petición al robot, utilizando ROS2 e indicando el topic correspondiente a la imagen de la cámara, para, de esta forma, poder obtener la imagen que está capturando la cámara del robot en ese momento. Una vez recibe la imagen, el mismo nodo se encarga de analizar la imagen utilizando una red neuronal que permite identificar algunos de los objetos que aparecen en la imagen, creando así una nueva en la que aparezcan los diferentes recuadros señalando cuáles han sido los elementos reconocidos por la red y su nombre. Esta nueva imagen es la que posteriormente, este nodo envía a la aplicación móvil a través del socket TCP para que pueda ser mostrada en la misma y pueda ser vista por el usuario.

- **Robot:** El robot, en este caso, simplemente sirve para comprobar el correcto funcionamiento del sistema desarrollado. Debe estar conectado al ordenador en el que se están ejecutando los nodos comentados anteriormente, bien sea a través de una red inalámbrica, si el robot dispusiera de esta tecnología, o, a través de una conexión de tipo Ethernet. Por supuesto, para el correcto funcionamiento de la aplicación, el robot debe estar encendido y se debe asegurar que todos los topics responsables de la tele operación, navegación y cámara del robot se están ejecutando de manera correcta, ya que van a ser con los que se comunique el resto del sistema.

En lo referente a la interfaz de la aplicación móvil ha sido desarrollada utilizando el lenguaje de programación Kotlin. Se ha distribuido de manera que solo cuente con un Activity, que en Kotlin es como se llama a cada una de las pantallas que conforman las aplicaciones desarrolladas con este lenguaje, en adelante, esta Actitvity va a ser denominada como "Pantalla principal". Sobre esta pantalla principal se van mostrando los diferentes Fragments, que en Kotlin se podrían tratar como vistas, en función de la acción solicitada por el usuario. De esta manera, la interfaz de la aplicación MAHURI cuenta con:

- Pantalla principal:
  - Es la pantalla sobre la que se muestran todas las vistas de la aplicación.
  - En la parte superior de la misma aparece el nombre la aplicación.
  - La parte central es la zona en la que van a ser mostradas las diferentes vistas que compone la aplicación.
  - En la parte inferior aparece un menú con las cuatro opciones que presenta la aplicación, las cuales están representadas cada una de ellas por un símbolo de manera que sean reconocibles. Estas opciones son:
    - Remote control: representada por un joystick.

        - Navigation: representada por un mapa.
        - Mic: Representada por un micrófono.
        - Camera: representada por una cámara.
- Vista Remote control:
    - Encargada de la tele operación con el robot.
    - Contiene cuatro flechas dispuestas en forma de joystick de manera que cada una indica una dirección, una hacia adelante, hacia atrás, hacia la derecha y hacia la izquierda. Estas flechas sirven para dirigir al robot cada una en la dirección que indican.
    - En el medio de las cuatro flechas aparece un botón con el texto "STOP", que servirá para detener el robot.
- Vista Navigation:
    - Encargada de la navegación autónoma del robot.
    - Contiene los pictogramas de los lugares configurados con anterioridad.
    - Estos pictogramas son imágenes que, al ser pulsadas, el robot se dirigirá al lugar que represente cada una de ellas.
- Vista Mic:
    - Esta vista es la encargada de captar a través del micrófono del dispositivo móvil los comandos de voz que el usuario realice.
    - Contiene un único botón, que, al ser pulsado, aparecerá un micrófono de manera que indique que el dispositivo ha comenzado a escuchar.
- Vista Camera:
    - Se encarga de mostrar al usuario la imagen que está siendo captada por la cámara del robot.
    - Únicamente incluye un recuadro en el que aparece la imagen de la cámara del robot.

3.2.3 Implementación

En este apartado se van a presentar los distintos lenguajes, librerías y herramientas que han sido utilizadas para llevar a cabo todo el desarrollo del proyecto, desde su implementación hasta el desarrollo de la memoria.

Lenguajes:

- Kotlin: Se trata de un lenguaje de programación desarrollado por Jetbrains y que permite crear multitud de tipos de aplicaciones, su uso más común es para el desarrollo de aplicación móviles para dispositivos con sistema operativo Android, pero también puede ser usado para desarrollar aplicaciones multiplataforma, aplicaciones web o de escritorio. Una ventaja que tiene es que es interoperable con Java y otros lenguajes [14]. En el caso de este proyecto, Kotlin ha sido utiliza para desarrollar la aplicación móvil, tanto la parte front-end como la parte back-end.
- Java: Java es uno de los lenguajes de programación más utilizados dentro del desarrollo de software. Fue comercializado por primera vez en 1995 por

Sun Microsystems. Puede ser utilizado para desarrollar todo tipo de software, desde aplicaciones móviles, juegos, software de empresa, etc [15]. En el caso de este proyecto, Java ha sido utilizado en la parte de la aplicación móvil para crear las dos clases que permitían realizar la conexión a través de los sockets TCP con los dos controladores. Esto es posible, ya que, como se comentó anteriormente, Kotlin es interoperable con Java.

- XML: XML es un lenguaje de marcado de propósito general. Su principal función es la de compartir datos a través de diferentes sistemas, como Internet [16]. En este sistema, XML ha sido utilizado para la creación de la parte gráfica de la aplicación móvil, ya que Kotlin utiliza este lenguaje para ese propósito. También ha sido utilizado en el nodo tele operador para crear el archivo de configuración del paquete que debe ser compilado para que pueda ser utilizado con ROS2.
- C++: Se trata de un lenguaje de programación compilado, multiparadigma y orientado a objetos. Una particularidad que posee es que un código fuente escrito en C puede ser compilado como C++. Algunos de los programas más populares escritos en C++ son los sistemas operativos Windows, Mac OS y Linux, al igual que algunos navegadores como Google Chrome o la página web de Amazon [17]. En este proyecto este lenguaje ha sido utilizado para desarrollar el nodo encargado de la tele operación y navegación del robot.
- Python: Python es un lenguaje multiparadigma, que permite que permite que sea flexible y fácil de aprender. Además, las aplicaciones que tiene no están limitadas a un área específica, las áreas en las que más es utilizado son el diseño de aplicaciones web y la inteligencia artificial. Tiene la filosofía de ser fácil de entender y leer para personas con pocos conocimientos de programación [18]. En el caso de este proyecto, Python ha sido utilizado para desarrollar el nodo encargado de obtener la imagen de la cámara del robot y enviársela a aplicación móvil para que el usuario pueda visualizarla.

Librerías:

- ROS2: Es un conjunto de librerías y herramientas de código abierto destinado a construir aplicaciones destinadas a interactuar con robots. La principal finalidad de ROS es ofrecer una plataforma estándar a los desarrolladores que ofrezca un mismo entorno para las aplicaciones en robótica [19]. En este caso, ROS2 ha sido utilizado en los dos nodos, tanto en el encargado de la tele operación y navegación, como en el encargado de la cámara para enviar órdenes y hacer peticiones al robot que se pretendía controlar a través de la aplicación móvil.
- Cv_bridge: Se trata de una librería que convierte las imágenes ROS a imágenes OpenCV [20]. En el proyecto esta librería fue utilizada en el nodo que obtenía las imágenes de la cámara del robot para poder transformarlas a imágenes reconocibles y poder visualizarlas de manera correcta en la aplicación móvil.

- RecognizerIntent: Se trata de una librería Android que permite realizar un reconocimiento de voz a través del micrófono del dispositivo [21]. Fue utilizada en este proyecto en la vista "Mic" para poder reconocer los comandos de voz que el usuario introdujera.

Herramientas:

- Android Studio: Es un entorno de desarrollo integrado (IDE) especialmente dedicado al desarrollo de aplicaciones Android. Entre sus características más destacadas está la de poder configurar emuladores de dispositivos móviles Android donde poder ejecutar las aplicaciones que se estén desarrollando [22]. En esta plataforma he desarrollado la aplicación móvil.
- Visual Studio Code: Se trata de un editor de código fuente ligero, pero con una gran cantidad de funcionalidades y con soporte integrado para lenguajes como C++, Python o JavaScript [23]. En este caso Visual Studio Code ha sido utilizado para desarrollar los dos nodos mencionados anteriormente, tanto el encargado de la tele operación y navegación como el encargado de la cámara.
- Gazebo: Es una herramienta de simulación que en el caso de este proyecto ha sido utilizada para ejecutar diferentes simuladores que permitían comprobar el correcto funcionamiento de la aplicación sin necesidad de utilizar un robot físico.
- Rviz: Es una herramienta de visualización 3D que ofrece ROS [24]. En este proyecto se utiliza en complemento con Gazebo para poder visualizar distintos parámetros del robot y algunos de sus topics.
- GitHub: Se trata del controlador de versiones utilizado para este proyecto. Permite almacenar los proyectos en remoto.
- Draw.io: Se trata de un editor online utilizado para crear el diagrama de casos de uso y el de la arquitectura del sistema.
- Visual paradigm: Es un editor de diagramas online con el que se han creado los diagramas de secuencia que aparecen en el documento.
- Microsft Project: Herramienta online utilizada para realizar los diagramas de Gantt del Anexo B.
- Google Drive: Es un servicio ofrecido por Google que permite el almacenamiento y acceso de archivos de forma online. En este caso ha sido utilizado para almacenar y gestionar toda la información referente a la documentación del proyecto.

3.2.4 Pruebas

Para testear este proyecto no podían utilizarse las pruebas unitarias normalmente utilizadas en los proyectos de desarrollo de software. En el caso de este proyecto, la mayoría de las pruebas realizadas han sido utilizando el simulador ofrecido por el co-tutor Miguel Ángel González Santamarta que se encuentra en el siguiente repositorio de GitHub [25]. Este simulador era ejecutado en las plataformas Gazebo y Rviz, mencionadas en el apartado 3.2.3 de este documento. Este simulador permitía poder realizar todo tipo de pruebas de igual manera que si se contara con un robot físico sobre el que poder ser realizadas. En la Figura 3.11 se

puede observar una captura de pantalla de la herramienta Gazebo en la que se muestra el robot utilizado en la simulación y que, por tanto, era el sobre el que se realizaban las diferentes pruebas para comprobar el correcto funcionamiento de la aplicación móvil.

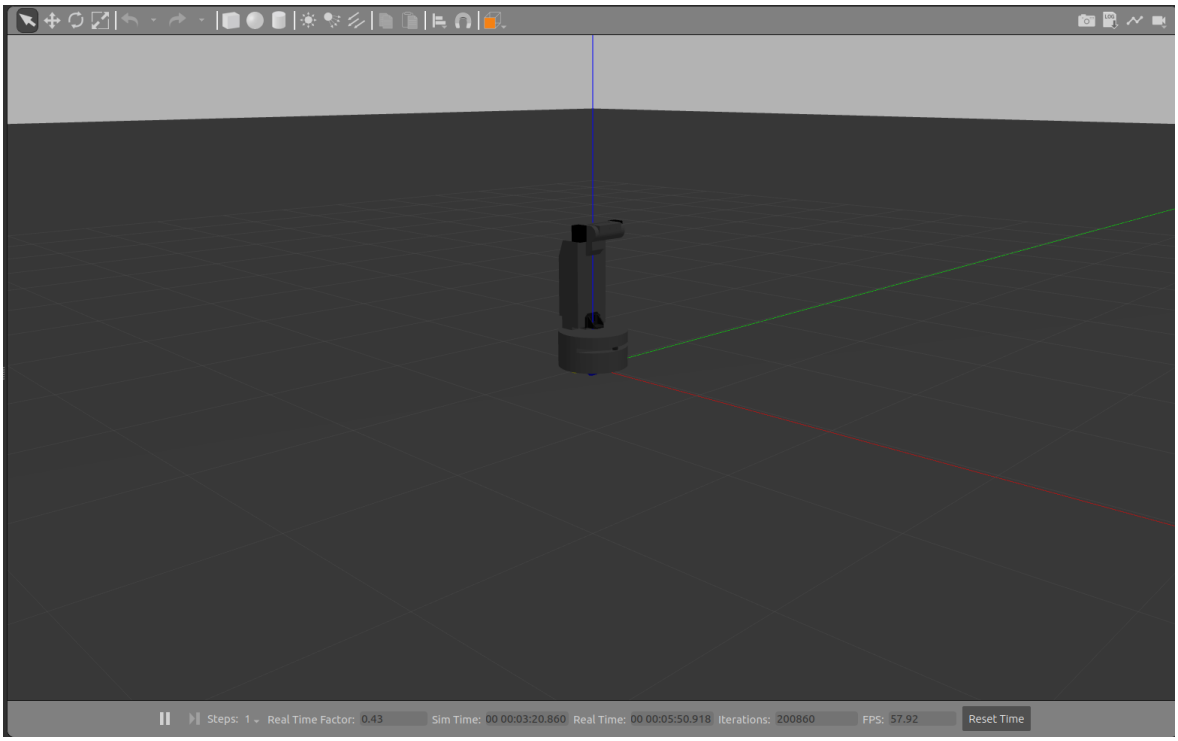


Figura 3.11 Simulador Gazebo

Un punto a destacar importante que tenía el uso de este simulador es que, tanto el robot, como el entorno en el que se realizaba la simulación eran los mismos que existían en el laboratorio del grupo de Robótica de la Universidad de León, esto tenía principalmente dos ventajas. En primer lugar, al tratarse del mismo robot, los topics que necesitaba conocer del mismo para poder configurarlos en los nodos eran los mismos que tendría el robot físico.

Por otro lado, como se ha comentado, el entorno que ofrecía el simulador era el mismo entorno que existía en el edificio MIC (Módulo de Investigación Cibernética) de la Universidad de León utilizado para realizar las pruebas con los robots, el cual está reflejado en la Figura 3.13. Este entorno, el cual se ve reflejado en la captura de pantalla obtenida de la herramienta RViz en la Figura 3.12, había sido mapeado e incluido en el simulador por miembros del grupo de Robótica, de manera que se conservaban las mismas coordenadas, y, por tanto, solo tendría que configurarlas una vez en el nodo para realizar la funcionalidad de navegación desde la aplicación móvil. Otras de las ventajas que ofrecía el simulador es que permitía también poder obtener la imagen capturada por la cámara del robot simulado, haciendo uso también de la herramienta RViz, esto era muy útil para, posteriormente, poder desarrollar la parte de la aplicación en la que mostraba en la pantalla del dispositivo móvil la imagen visualizada por la cámara del robot.

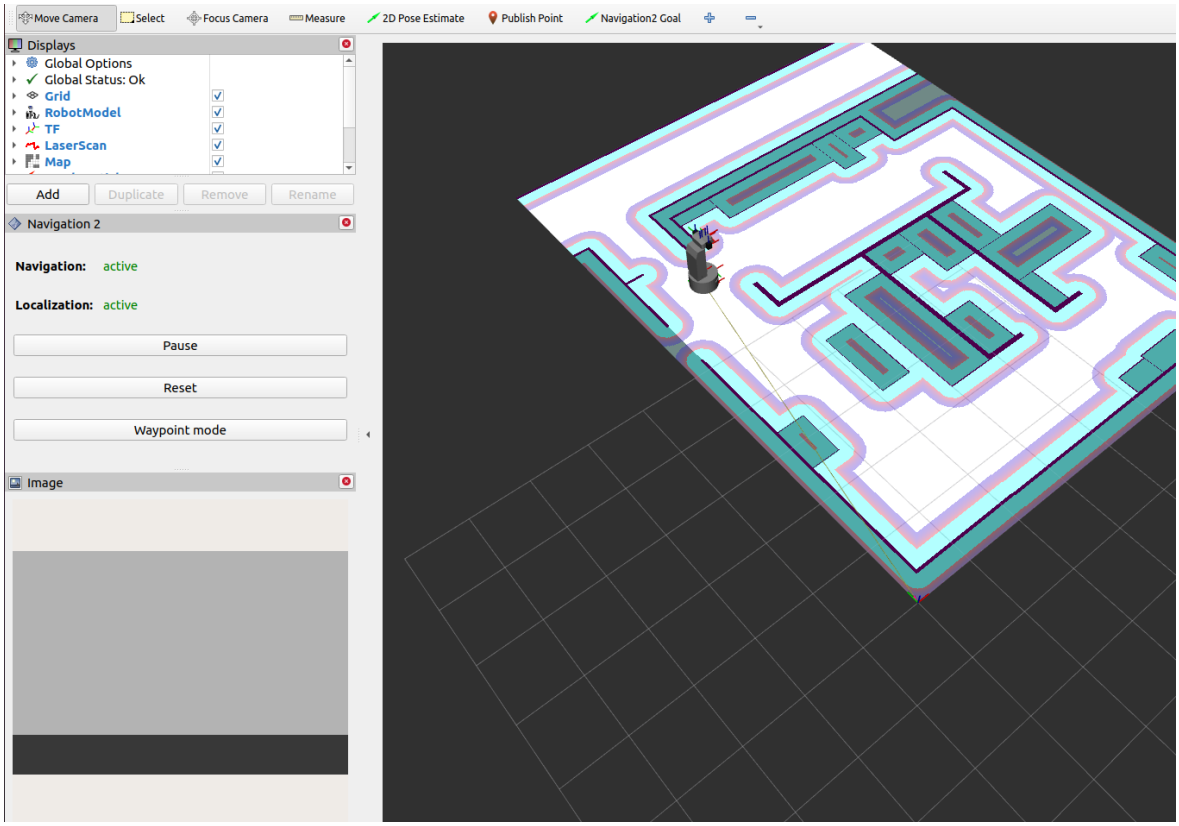


Figura 3.12 Simulador RViz

Una vez se podía asegurar que la aplicación funcionaba de forma correcta en el simulador mencionado con anterioridad, se acudía al edificio MIC de la Universidad de León, en el cual se encontraban los robots y el entorno físico, reflejados en la Figura 3.13. De esta manera, ya podían realizarse una mayor cantidad de pruebas y comprobar que la aplicación móvil desarrollada funcionaba correctamente en un entorno real.

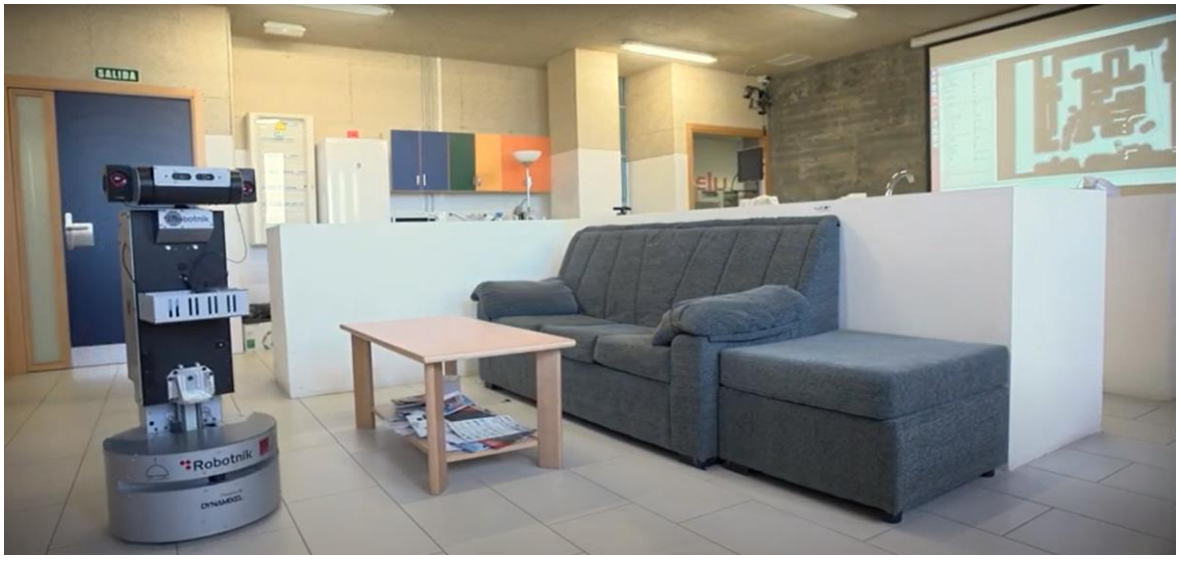


Figura 3.13 Entorno de pruebas edificio MIC

Los dos robots utilizados para realizar las pruebas ofrecidos por el grupo de Robótica de la Universidad de León son, el robot TIAGo, el cual aparece reflejado en la Figura 3.15, y el robot RB-1 representado en la Figura 3.14.

Figura 3.14 Robot RB-1 [28]

Figura 3.15 Robot TIAGo [27]

3.3 El producto del desarrollo

Como se ha especificado en los anteriores apartados de este mismo documento, la aplicación móvil desarrollada cuenta con cuatro pantallas diferentes, en esta sección se va a mostrar cada una de las pantallas, explicando su funcionalidad. Cabe destacar que la aplicación desarrollada cuenta con modo oscuro y modo claro que se habilita en función del modo en el que esté configurado el dispositivo sobre el que se está ejecutando, es por eso que, para cada una de las vistas se va a añadir un pantallazo de su modo oscuro y modo claro.

En primer lugar, se va a presentar la pantalla “Remote control”, la cual aparece representada en las Figuras 3.16 y 3.17. Como se puede ver en esas dos figuras, la vista presenta una especia de mando a control remoto en el cual aparecen representadas cuatro flechas, cada una apuntando a la dirección en la que se desee desplazar el robot, y un botón de STOP situado en el centro de las flechas de manera que, al ser pulsado, el robot se detendría por completo. Se trata de una vista sencilla y fácil de entender para cualquier tipo de usuario que desee hacer uso de la aplicación. Un punto a destacar para el resto de vistas que van a ser presentadas, es que, una vez se accede a cada una de ellas a través del menú

inferior de la aplicación aparece resaltada la vista en la que te encuentras y el nombre de la misma, de manera que sea claramente identificativo.

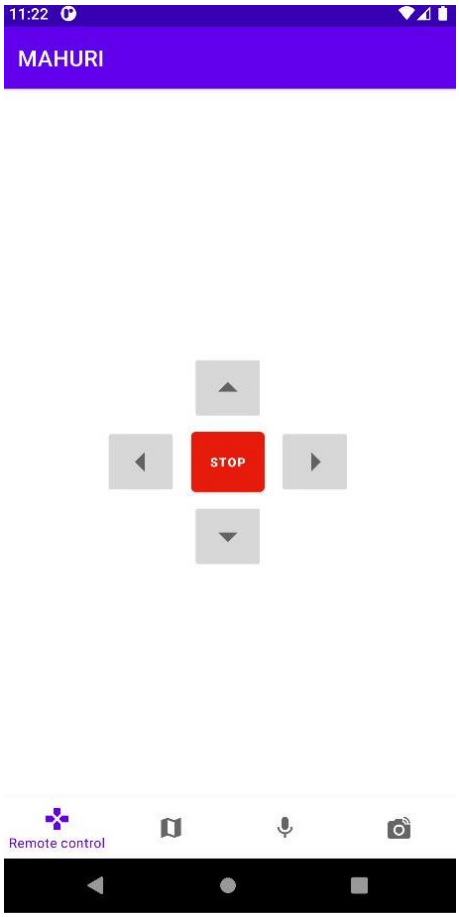


Figura 3.16 Remote control claro

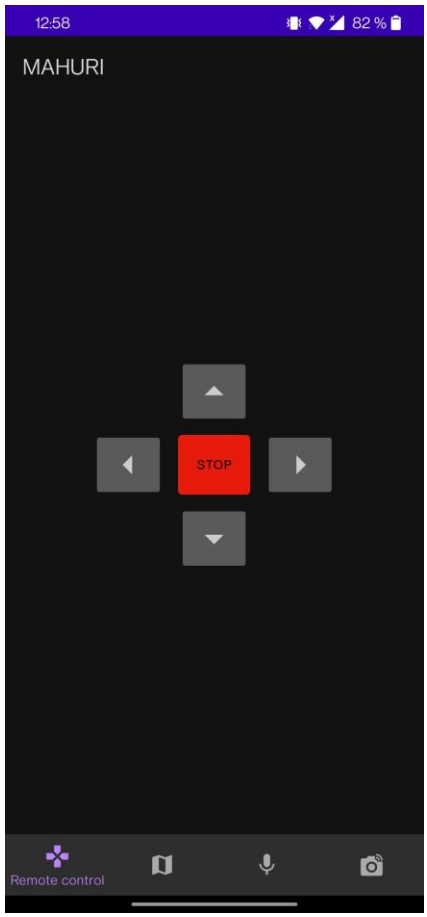


Figura 3.17 Remote control oscuro

La segunda pantalla siguiendo el orden del menú de la parte inferior de la aplicación es la llamada "Navigation", en esta pantalla aparecen diferentes pictogramas de los lugares preconfigurados con anterioridad en el sistema acompañados, debajo de los mismos, del nombre del lugar que representan. Los pictogramas se muestran en forma de lista vertical de manera que aparezcan en un tamaño grande que permita ser identificado por cualquier tipo de usuario que haga uso de la aplicación.

La funcionalidad que ofrece esta pantalla consiste en, como ya ha sido explicado anteriormente en este documento, en el momento en el que el usuario pulse uno de los pictogramas el robot se dirigirá de forma autónoma al mismo en el entorno en el que haya sido configurado. Esta parte de la aplicación aparece representada tanto con el tema oscuro como con el tema claro en las Figuras 3.18 y 3.19.

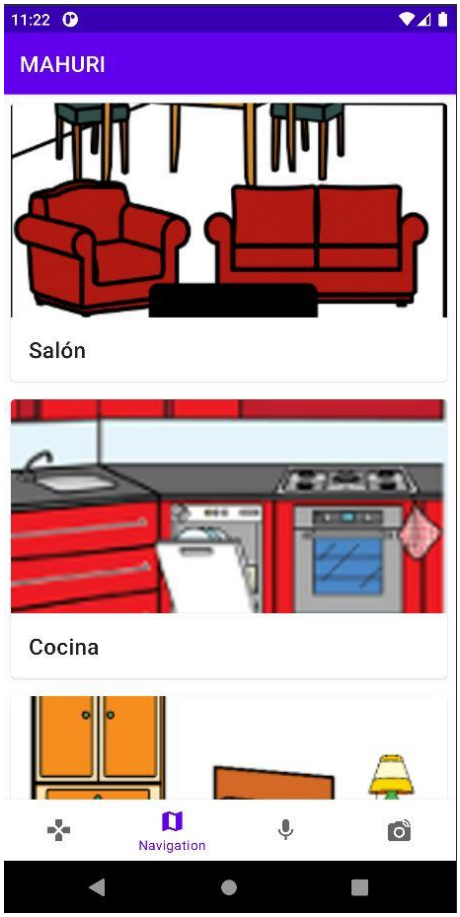


Figura 3.18 Navigation claro

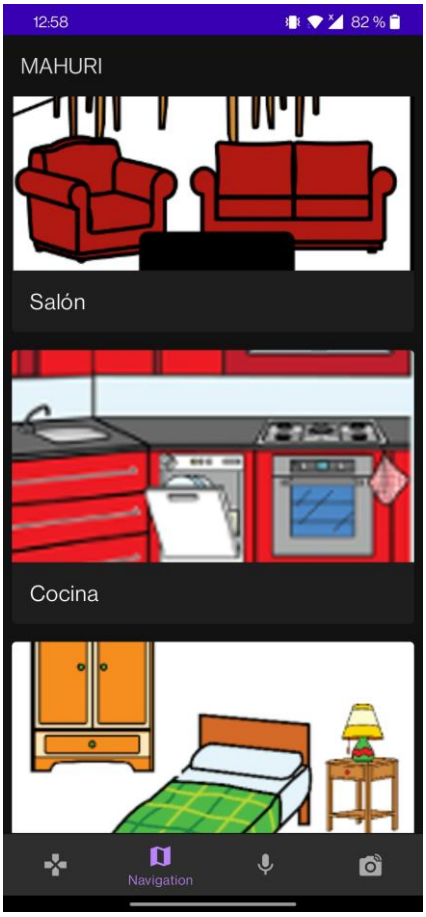


Figura 3.19 Navigation oscuro

La tercera pantalla que aparece en el orden que muestra el menú de la aplicación es la llamada "Mic". Esta pantalla es la encargada de realizar el reconocimiento de los comandos de voz que permitan al usuario de la aplicación realizar ordenar al robot realizar las mismas acciones que las mencionadas en la pantalla "Remote control" y "Navigation" pero todo a través de comandos de voz. En esta vista solo aparece un botón en la parte central con un icono de un micrófono, como se puede observar en las Figuras 3.21 y 3.22, que, tras ser pulsado, aparecerá un recuadro como el mostrado en la Figura 3.20 que indicará que el dispositivo está escuchando y listo para reconocer el comando de voz que interprete.

Figura 3.20 Reconocimiento de voz

En esta pantalla se decidió mostrar solo el botón en grande mencionado anteriormente con el icono del micrófono de manera que fuera indicativo, visible y accesible para todo tipo de usuarios que pudieran utilizar la aplicación, y más concreto esta funcionalidad en específico. A través de esta pantalla el usuario puede introducir diferentes comandos de voz como “Adelante”, “Atrás”, “Stop” y dirigir, de esta manera el robot, o incluso algunos como “Cocina” o “Salón”, para, de esta manera enviar al robot de forma autónoma a esos puntos. Esta funcionalidad mejora de forma notable la accesibilidad de la aplicación, ya que puede ser muy útil para personas mayores o que no estén tan acostumbradas al uso de este tipo de aplicaciones.

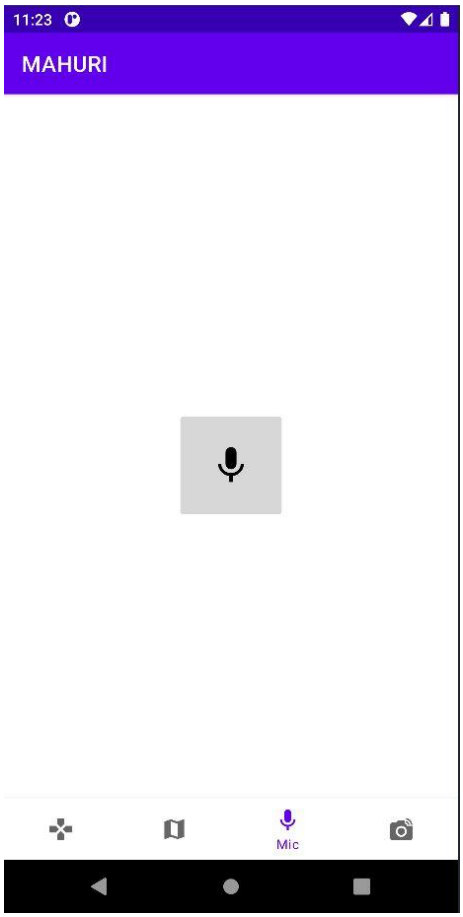


Figura 3.21 Mic claro

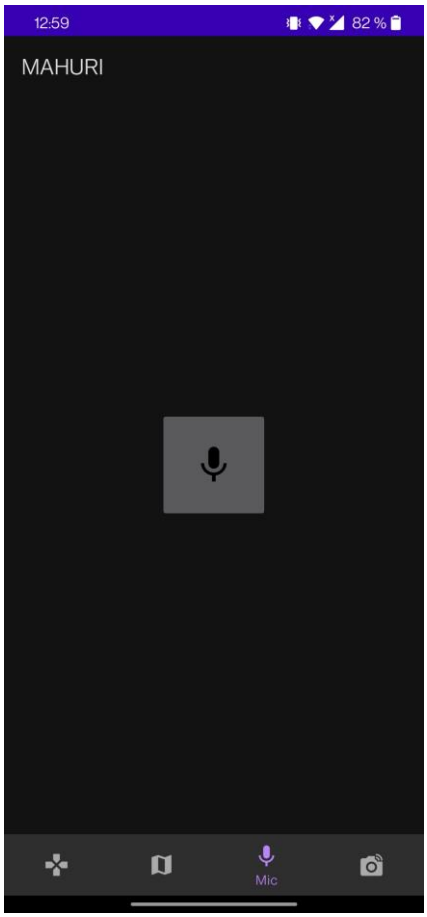


Figura 3.22 Mic oscuro

En la última pantalla, la que toma por nombre “Camera”, simplemente se muestra, en la parte central de la pantalla, la imagen obtenida de la cámara del robot a tiempo real y tras haber sido analizada por la red neuronal incluida dentro del nodo de la cámara que le permitía detectar algunos de los objetos que aparecieran en la misma. De esta manera, el usuario en todo momento podría acceder a la cámara del robot y visualizar el lugar en el que se encuentra, si existe algún obstáculo entorpeciendo su camino, etc. La detección de objetos se realiza a través de una red ya entrenada la cual es llamada desde el nodo de la cámara,

desde este mismo nodo se pintan en la imagen los diferentes recuadros que permiten identificar dónde se encuentran los objetos detectados por la red junto con su nombre gracias a la librería OpenCV. Algunas de las imágenes obtenidas y procesadas por este nodo son las siguientes:

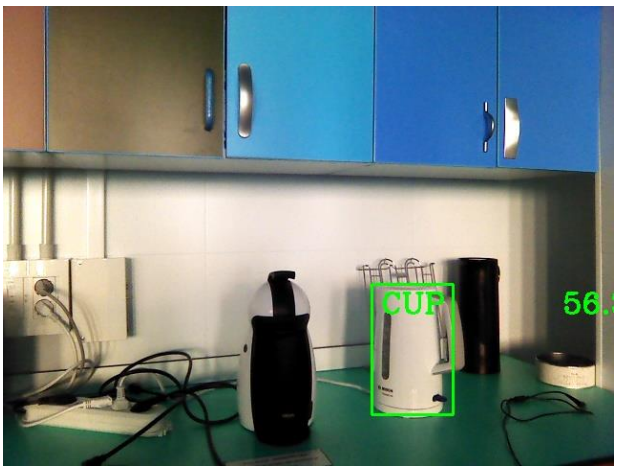


Figura 3.23 Imagen nodo cámara 1

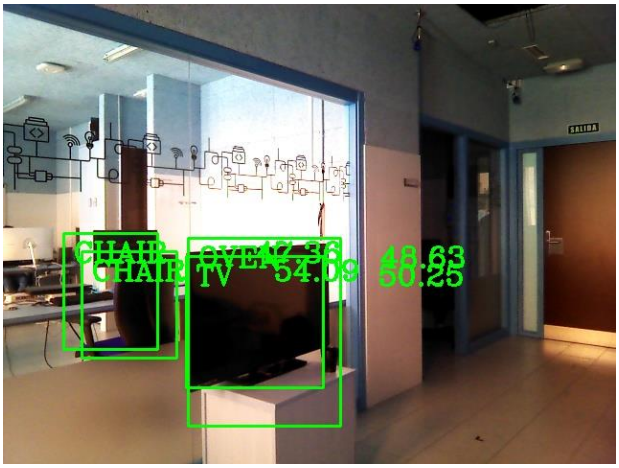


Figura 3.24 Imagen nodo cámara 2

Figura 3.25 Imagen nodo cámara 3

4 Evaluación

En esta sección se va a explicar las pruebas que se han llevado a cabo para demostrar la validez del sistema desarrollado y comprobar que satisface los objetivos definidos anteriormente en este documento.

4.1 Proceso de evaluación

En este apartado se va a explicar la forma en la que se ha evaluado el sistema y los casos de prueba utilizados para comprobar su correcto funcionamiento.

4.1.1 Forma de evaluación

La forma de evaluar el software ha sido a través de simuladores de robots, y con los propios robots físicos proporcionados por el grupo de Robótica de la Universidad de León, como se explica en el apartado 3.2.4 de este mismo documento. Además, con el entorno de pruebas situado en el edificio MIC se permitía desplegar el sistema en un entorno real, y poder así, comprobar su correcto funcionamiento en un entorno similar al que está diseñado para ser utilizado.

4.1.2 Casos de prueba

Los dos casos de prueba a tener en cuenta en este proyecto son en el entorno simulado y en el entorno real:

- **Prueba en entorno simulado:**
  Para esta primera prueba se utilizó el simulador reflejado en las Figuras 3.11 y 3.13. En este caso, los simuladores proporcionaban la oportunidad de realizar multitud de pruebas sin la necesidad de contar con un robot físico y un entorno real. De esta manera, resulta mucho más sencillo poder realizar pruebas con una frecuencia de tiempo muy corta y poder detectar los fallos con mayor rapidez.
- **Prueba en entorno real:**
  Estas pruebas fueron realizadas en el entorno representado en la Figura 3.12 de este documento y con los robots que aparecen en las Figuras 3.14 y 3.15. Tanto el entorno como los robots fueron proporcionado por el grupo de Robótica de la ULE. En estas pruebas se podía comprobar, de forma real, cómo era el funcionamiento de la aplicación y si cumplía con los requisitos y objetivos establecidos. Además, permitía probar el sistema en diferentes situaciones para comprobar si se adaptaba y funcionaba de forma correcta, ya que el robot TIAGo (Figura 3.15), contaba con el sistema operativo ROS2 y la conexión al ordenador donde se estaban ejecutando los nodos se realizaba mediante cable Ethernet. Por otro lado, el robot RB-1, el representado en la Figura 3.14, contaba con sistema operativo ROS, por lo que era necesario utilizar un bridge que convirtiera las órdenes de ROS2

que emitía el sistema a ROS para que pudieran ser comprendidas y ejecutadas por el robot, y, además, la conexión entre el ordenador y el robot se realizaba de manera inalámbrica a través de la red WiFi que el mismo ofrecía, a la cual debía estar conectado también el dispositivo móvil desde el que se ejecutaba la aplicación móvil desarrollada.

4.2 Análisis de resultados

Como se ha mencionado en la sección anterior, las pruebas se realizaron en dos entornos diferentes, tanto en un entorno simulado, a través del simulador proporcionado por el co-tutor Miguel Ángel González Santamarta, como en un entorno físico real. Todas las pruebas eran realizadas, en primer lugar, en el entorno simulado, ya que permitía poder realizar gran cantidad de cambios sin necesidad de utilizar una gran cantidad de recursos, posteriormente, una vez se comprobaba que las acciones se realizaban de forma correcta en el simulador, se comprobaba en el entorno físico.

En el entorno físico podían realizarse una mayor cantidad de pruebas con variables que en el simulador no se podían controlar, como, por ejemplo, la posibilidad de obstaculizar el paso del robot con algunos objetos una vez el usuario había mandado la orden desde la aplicación de que navegara hasta un cierto punto de forma autónoma y comprobar así, cómo respondía el robot. Otro ejemplo de las pruebas que podían ser realizadas en el entorno físico era la de comprobar si la red neuronal que analizaba las imágenes provenientes de la cámara del robot reconocía de forma exitosa los objetos que aparecían en las mismas. Esta prueba no podía ser realizada en el simulador ya que contaba con muy poca cantidad de objetos para identificar y no eran del todo realistas. Todas estas pruebas fueran superadas de manera exitosa siguiendo los objetivos fijados al comienzo del documento con los cuales debía cumplir el sistema finalmente desarrollado.

Por último, gracias a los robots y el entorno proporcionados por el área de robótica de la Universidad de León, se pudo comprobar que la aplicación funcionaba de forma correcta tanto con robots que no disponían de red inalámbrica propia, como es el caso del robot TIAGo, representado en la Figura 3.14, y que, por tanto, el ordenador debía ser conectado a través de un cable Ethernet al propio robot. Por otro lado, se comprobó que el sistema también funcionaba de forma correcta con robots que contaran con su propia red inalámbrica, como es el caso de robot RB-1 (Figura 3.15), a la cual debían estar conectados tanto el dispositivo móvil en el que se ejecutara la aplicación, como el ordenador en el que se estuvieran ejecutando los nodos que recibieran las órdenes de la misma. Un punto añadido, además, era que el robot RB-1 contaba con sistema operativo ROS, en lugar de ROS2, por lo que se pudo asegurar también que, mediante el uso de un bridge ejecutado en el ordenador que transforma las órdenes de ROS2 a ROS, el sistema era compatible con robots que solo dispusieran de sistema operativo ROS.

5 Conclusión

En esta sección se va a explicar los puntos a destacar desde un punto de vista más subjetivo del desarrollo del proyecto, así como las líneas futuras en las que se podría trabajar para mejorar el mismo.

5.1 Aportaciones realizadas

Una vez el trabajo ha sido finalizado, se pueden obtener varias conclusiones en relación al proyecto. Se ha conseguido alcanzar el objeto principal, que consistía en el desarrollo de una aplicación móvil que permitiera interactuar con robots. Se ha desarrollado un sistema que permite la tele operación del robot, tanto de forma gráfica como mediante la voz, la navegación de forma autónoma del mismo al lugar indicado por el usuario dentro de la aplicación, y, por último, la visualización de la imagen capturada por la cámara del robot. De esta forma, se permite al usuario disfrutar una interacción rápida e intuitiva con el robot que se encuentre conectado a la aplicación. Puede ser de gran ayuda para personas que estén desarrollando sus propios robots y deseen probar su funcionamiento sin necesidad de utilizar un mando alternativo, o para personas que deseen mejorar la interacción con sus robots ofreciendo una mejor usabilidad y mayor cantidad de funcionalidades.

5.2 Trabajos futuros

En este apartado se van a listar las posibles mejoras o ampliaciones que podrían llevarse a cabo en un futuro en el sistema:

- Posibilidad de introducir manualmente la dirección IP del ordenador que se encuentre conectado al robot y en el que se estén ejecutando los nodos.
- Mejorar la navegación autónoma en el nodo encargado de la tele operación de manera que se realice de forma síncrona.
- Capacidad de poder introducir las coordenadas de un punto de manera manual para que el robot pueda navegar a el mismo sin estar previamente configurado.
- Poder introducir, a través de línea de comandos, el nombre de los topics de cada robot para, en caso de que se posea más de un robot, poder adaptar la aplicación de forma más rápida a cada uno de ellos.
- Incluir una funcionalidad en la que se pueda ordenar más acciones como controlar los brazos del robot, en caso de que cuente con ellos, de manera que se pueda coger algún objeto.

5.3 Problemas encontrados

ROS2 en Windows

En un primer momento, uno de los problemas que me encontré surgió a la hora de realizar la instalación y configuración de ROS2 en Windows. Esto era debido a que

muchas de las funcionalidades del mismo no están pensadas para ser ejecutadas sobre este sistema operativo. Uno de los errores que me encontré fue la incompatibilidad a la hora de ejecutar los diferentes simuladores. Otro problema a destacar fue la dificultad en comparación con un sistema operativo Linux, para la creación de diferentes sockets que permitieran a mi aplicación móvil conectarse con los diferentes entornos desarrollados con código ROS2.

Tras encontrarme con esta gran cantidad de dificultades que suponía la realización de este proyecto sobre el sistema operativo Windows 10, decidí realizarlo sobre el sistema operativo Ubuntu 20.04. Para ello, la opción más óptima a la hora de trabajar con este nuevo sistema operativo fue la instalación de forma nativa en mi ordenador personal junto con el sistema operativo Windows 10.

Suscripción al nodo dentro del topic_callback

Al momento de iniciarme en el uso de ROS2 y desarrollar mi primer programa con esta tecnología, uno de los errores que me surgieron se encontraba en el programa que enviaba las órdenes al simulador. En este caso, el problema surgió debido a que tanto la inicialización del publicador como las diferentes operaciones que debía realizar el mismo, estaban incluidas dentro de la función suscriptora del nodo que escuchaba los mensajes provenientes de mi aplicación móvil. Es decir, que el nodo que se encargaba de publicar los mensajes para que fueran escuchados por el simulador, estaba declarado dentro de la función que escuchaba los mensajes provenientes de la aplicación, que en este caso era la función topic_callback.

Este error provocaba que los mensajes que la aplicación móvil emitía eran escuchados correctamente por el nodo, pero, sin embargo, no llegaban nunca a ser publicados para que pudieran ser escuchados por el nodo del simulador, ya que se formaba un bucle infinito dentro de la función suscriptora. De esta manera era imposible enviar ninguna orden al simulador, aunque estuviera formulada de forma correcta.

Socket nodo de la cámara

En un primer momento, al crear el programa en Python que se suscribía al topic de la cámara del robot para obtener la imagen de la misma y enviársela a la aplicación, una vez creaba el socket para enviar la imagen a la misma no realizaba la llamada a la función accept() del socket para que esperara a recibir la conexión por parte del socket creado en la aplicación, lo que provocaba que el programa se detuviera por completo al momento de ser iniciado lanzando una excepción del tipo ConnectionRefusedError. Una vez añadida la función accept() en el socket, el nodo no intentaba enviar ninguna imagen hasta que conseguía establecer la conexión de manera correcta con el socket creado en la aplicación móvil.

5.4 Opiniones personales

El desarrollo de este proyecto me ha permitido ampliar mi conocimiento en una gran variedad de ramas relacionadas con la informática y la robótica. En primer lugar, me ha permitido obtener un mayor conocimiento del área relacionada con el desarrollo de aplicaciones para dispositivos móviles, en especial me ha servido para conocer e iniciarme en el lenguaje Kotlin. Por otro lado, me ha proporcionado una gran cantidad de conocimientos relacionados con la robótica, que era un tema el cual no había tratado con antelación, me ha resultado muy interesante conocer cómo funciona ROS y la forma en la que se mandan y reciben parámetros de los robots.

Además, por otra parte, he comprendido cómo planificar y documentar de forma correcta un proyecto de estas características y la importancia que tiene. A su vez, también me ha servido para conocer, aunque no sea de manera totalmente real, cómo funciona un equipo de trabajo de desarrollo de software en el día a día, lo cual creo que me va a ser de gran utilidad en mi próximo paso de introducirme al mundo laboral.

6 Premios

En este apartado se hace mención al premio Accésit recibido en el II Concurso de prototipos (modalidad software) de la Escuela de Ingenierías Industrial, Informática y Aeroespacial de la Universidad de León el día 3 de junio de 2022 tras presentar en el mismo el proyecto descrito en este documento.

Figura 6.1 Premio Accésit Concurso de prototipos

Lista de referencias

[1] C. Queiruga, C. B. Tzancoff, and F. López, “RemoteBot: una Aplicación que Combina Robots y Dispositivos Móviles”.

[2] “Robot Control - Aplicaciones en Google Play.” https://play.google.com/store/apps/details?id=com.dalvikapps.robotcontrol&hl=es&gl=US (accessed Jun. 20, 2022).

[3] “Robot Data and Operations Platform for Heterogenous Fleets - Formant.” https://formant.io/ (accessed Jun. 20, 2022).

[4] “TIAGo - El manipulador móvil para tu investigación | PAL Robotics.” https://pal-robotics.com/es/robots/tiago/ (accessed Jun. 21, 2022).

[5] “El robot TIAGo - GrupoADD.” https://grupoadd.es/el-robot-tiago (accessed Jun. 22, 2022).

[6] “Sueldo: Scrum Master (Junio, 2022) | Glassdoor.” https://www.glassdoor.es/Sueldos/scrum-master-sueldo-SRCH_KO0,12.htm (accessed Jun. 22, 2022).

[7] “Sueldo: Desarrollador De Software Junior (Junio, 2022) | Glassdoor.” https://www.glassdoor.es/Sueldos/desarrollador-de-software-junior-sueldo-SRCH_KO0,32.htm?clickSource=searchBtn (accessed Jun. 22, 2022).

[8] “Sueldo: Disenador Grafico (Junio, 2022) | Glassdoor.” https://www.glassdoor.es/Sueldos/dise%C3%B1ador-gr%C3%A1fico-sueldo-SRCH_KO0,17.htm?clickSource=searchBtn (accessed Jun. 22, 2022).

[9] “Sueldo: Tester (Junio, 2022) | Glassdoor.” https://www.glassdoor.es/Sueldos/tester-sueldo-SRCH_KO0,6.htm?clickSource=searchBtn (accessed Jun. 22, 2022).

[10] M. de la Presidencia and R. E. Con Las Cortes Igualdad, “I. DISPOSICIONES GENERALES MINISTERIO DE LA PRESIDENCIA, RELACIONES CON LAS CORTES E IGUALDAD,” 2018, Accessed: Jun. 25, 2022. [Online]. Available: http://www.boe.es

[11] J. del Estado, “Disposición 16673 del BOE núm. 294 de 2018,” 2018, Accessed: Jun. 25, 2022. [Online]. Available: http://www.boe.es

[12] “Título I. De los derechos y deberes fundamentales - Constitución Española.” https://app.congreso.es/consti/constitucion/indice/titulos/articulos.jsp?ini=18&tipo=2 (accessed Jun. 25, 2022).

[13] “(PDF) Especificación de Requisitos según el estándar de IEEE 830 | Christian Lazcano - Academia.edu.” https://www.academia.edu/8970934/Especificaci%C3%B3n_de_Requisitos_seg%C3%BAn_el_est%C3%A1ndar_de_IEEE_830 (accessed Jun. 26, 2022).

[14] “Get started with Kotlin | Kotlin.” https://kotlinlang.org/docs/getting-started.html (accessed Jun. 28, 2022).

[15] "¿Qué es Java y para qué es necesario?" https://www.java.com/es/download/help/whatis_java.html (accessed Jun. 28, 2022).

[16] "Introducción a XML - XML: Extensible Markup Language | MDN." https://developer.mozilla.org/es/docs/Web/XML/XML_introduction (accessed Jun. 28, 2022).

[17] "El lenguaje C++ — Fundamentos de Programación en C++." https://www2.eii.uva.es/fund_inf/cpp/temas/1_introduccion/introduccion.html (accessed Jun. 28, 2022).

[18] "¿Para qué sirve Python? Razones para utilizarlo | ESIC." https://www.esic.edu/rethink/tecnologia/para-que-sirve-python (accessed Jun. 28, 2022).

[19] "ROS 2 Documentation — ROS 2 Documentation: Foxy documentation." https://docs.ros.org/en/foxy/index.html (accessed Jun. 28, 2022).

[20] "cv_bridge - ROS Wiki." http://wiki.ros.org/cv_bridge (accessed Jun. 28, 2022).

[21] "RecognizerIntent | Android Developers." https://developer.android.com/reference/android/speech/RecognizerIntent (accessed Jun. 28, 2022).

[22] "Introducción a Android Studio | Desarrolladores de Android | Android Developers." https://developer.android.com/studio/intro/?gclid=CjwKCAjwzeqVBhAoEiwAOrEmzW8yqor1qd5k3lDCNA3qHsjHKfIxdVMGtd7-2AQtdEW_RjWwB60VEhoCKdoQAvD_BwE&gclsrc=aw.ds (accessed Jun. 28, 2022).

[23] "Documentation for Visual Studio Code." https://code.visualstudio.com/docs (accessed Jun. 29, 2022).

[24] "rviz - ROS Wiki." http://wiki.ros.org/rviz (accessed Jun. 29, 2022).

[25] "mgonzs13/ros2_rb1." https://github.com/mgonzs13/ros2_rb1 (accessed Jun. 29, 2022).

[26] "file.png (771×513)." https://static.wixstatic.com/media/fd2cc9_128a79ff8412496d9ee11d4a22d5e286~mv2.png/v1/fit/w_771%2Ch_513%2Cal_c/file.png (accessed Jun. 20, 2022).

[27] "Grupo Robotica Universidad de León en Twitter: 'TIAGo es uno de los robots que utilizaremos en nuestro máster en Robótica y Sistemas Inteligentes. ¿Quieres saber más? Visita nuestra web: https://t.co/0ujD8bKwwi 🟢Segundo plazo de inscripción para el curso 21/22 ABIERTO 🟢 https://t.co/sVcbYOofRw' / Twitter." https://twitter.com/RoboticaUnileon/status/1429863984530956289 (accessed Jun. 29, 2022).

[28] “La ULE y la Universidad Rey Juan Carlos (URJC) investigan un sistema de localización para robots sociales móviles | Universidad de León.” https://www.unileon.es/noticias/la-ule-y-la-universidad-rey-juan-carlos-urjc-investigan-un-sistema-de-localizacion-para-robots-socia (accessed Jun. 29, 2022).

ANEXO A: Control de Versiones

Para realizar el control de versiones del desarrollo del sistema se ha utilizado la plataforma GitHub, de manera que se pudieran almacenar todas las modificaciones realizadas en el proyecto en un servidor remoto y poder recuperar, en caso de que fuera necesario, alguna versión anterior. Además, se guardaba también una copia de las versiones más recientes de la documentación en una carpeta de Google Drive compartida con la tutora del Trabajo de Fin de Grado, de esta manera se permitía también un mejor seguimiento del trabajo avanzado por su parte. A continuación, se van a listar los diferentes enlaces en los que se encuentran el código necesario para poner en marcha el sistema:

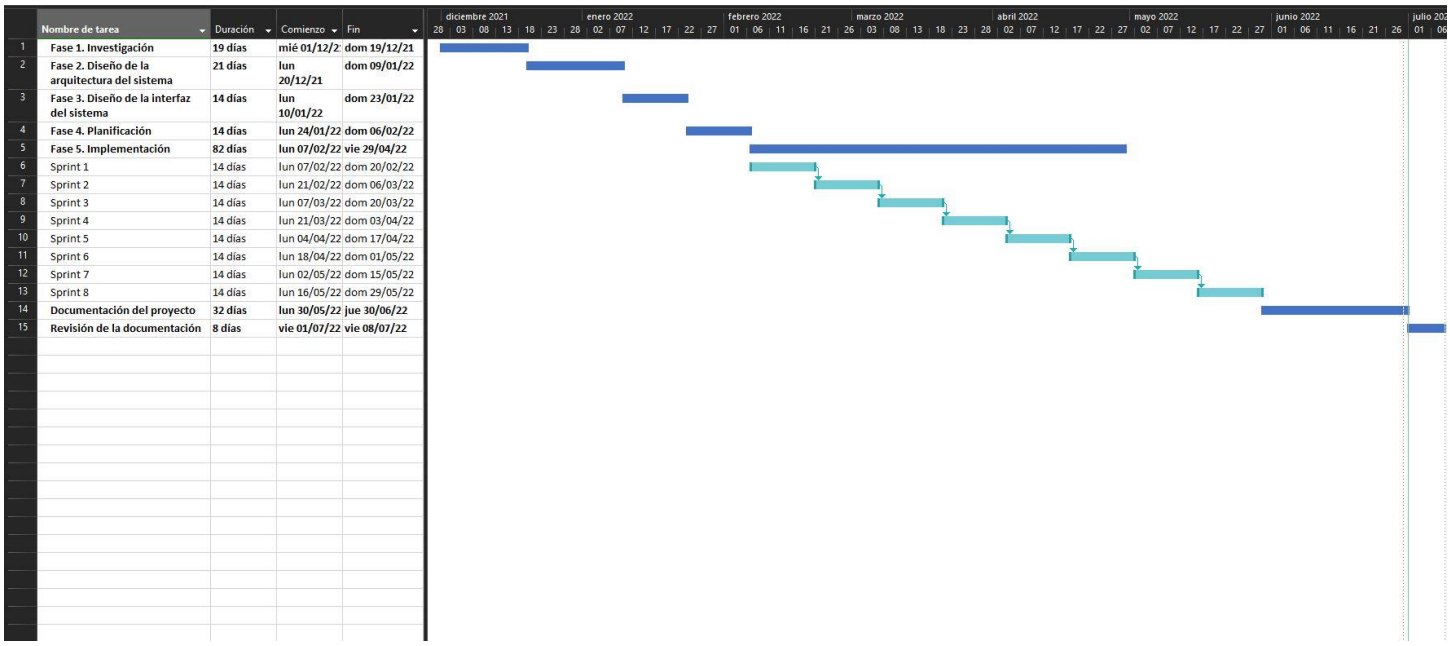
- **Código de la aplicación móvil:**
  https://github.com/iferng09/MAHURI

- **Código del nodo tele operador:**
  https://github.com/iferng09/MAHURI_teleop

- **Código del nodo de la cámara:**
  https://github.com/iferng09/MAHURI_camera

ANEXO B: Seguimiento de proyecto fin de carrera

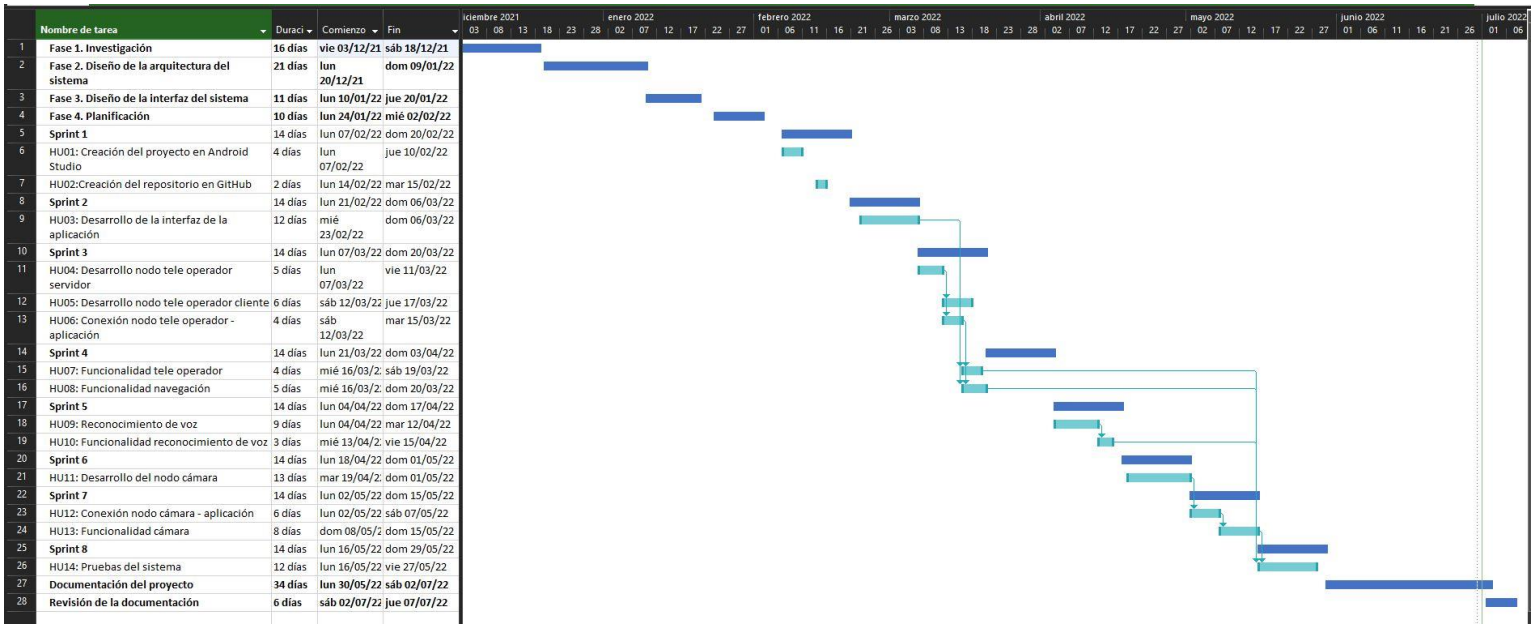
Anexo B.1 Planificación inicial



	Nombre de tarea	Duración	Comienzo	Fin
1	Fase 1. Investigación	19 días	mié 01/12/21	dom 19/12/21
2	Fase 2. Diseño de la arquitectura del sistema	21 días	lun 20/12/21	dom 09/01/22
3	Fase 3. Diseño de la interfaz del sistema	14 días	lun 10/01/22	dom 23/01/22
4	Fase 4. Planificación	14 días	lun 24/01/22	dom 06/02/22
5	Fase 5. Implementación	82 días	lun 07/02/22	vie 29/04/22
6	Sprint 1	14 días	lun 07/02/22	dom 20/02/22
7	Sprint 2	14 días	lun 21/02/22	dom 06/03/22
8	Sprint 3	14 días	lun 07/03/22	dom 20/03/22
9	Sprint 4	14 días	lun 21/03/22	dom 03/04/22
10	Sprint 5	14 días	lun 04/04/22	dom 17/04/22
11	Sprint 6	14 días	lun 18/04/22	dom 01/05/22
12	Sprint 7	14 días	lun 02/05/22	dom 15/05/22
13	Sprint 8	14 días	lun 16/05/22	dom 29/05/22
14	Documentación del proyecto	32 días	lun 30/05/22	jue 30/06/22
15	Revisión de la documentación	8 días	vie 01/07/22	vie 08/07/22

Figura B.1 Planificación inicial

Anexo B.2 Planificación final



	Nombre de tarea	Duraci	Comienzo	Fin
1	**Fase 1. Investigación**	**16 días**	**vie 03/12/21**	**sáb 18/12/21**
2	**Fase 2. Diseño de la arquitectura del sistema**	**21 días**	**lun 20/12/21**	**dom 09/01/22**
3	**Fase 3. Diseño de la interfaz del sistema**	**11 días**	**lun 10/01/22**	**jue 20/01/22**
4	**Fase 4. Planificación**	**10 días**	**lun 24/01/22**	**mié 02/02/22**
5	**Sprint 1**	14 días	lun 07/02/22	dom 20/02/22
6	HU01: Creación del proyecto en Android Studio	4 días	lun 07/02/22	jue 10/02/22
7	HU02:Creación del repositorio en GitHub	2 días	lun 14/02/22	mar 15/02/22
8	**Sprint 2**	14 días	lun 21/02/22	dom 06/03/22
9	HU03: Desarrollo de la interfaz de la aplicación	12 días	mié 23/02/22	dom 06/03/22
10	**Sprint 3**	14 días	lun 07/03/22	dom 20/03/22
11	HU04: Desarrollo nodo tele operador servidor	5 días	lun 07/03/22	vie 11/03/22
12	HU05: Desarrollo nodo tele operador cliente	6 días	sáb 12/03/22	jue 17/03/22
13	HU06: Conexión nodo tele operador - aplicación	4 días	sáb 12/03/22	mar 15/03/22
14	**Sprint 4**	14 días	lun 21/03/22	dom 03/04/22
15	HU07: Funcionalidad tele operador	4 días	mié 16/03/2	sáb 19/03/22
16	HU08: Funcionalidad navegación	5 días	mié 16/03/2	dom 20/03/22
17	**Sprint 5**	14 días	lun 04/04/22	dom 17/04/22
18	HU09: Reconocimiento de voz	9 días	lun 04/04/22	mar 12/04/22
19	HU10: Funcionalidad reconocimiento de voz	3 días	mié 13/04/2	vie 15/04/22
20	**Sprint 6**	14 días	lun 18/04/22	dom 01/05/22
21	HU11: Desarrollo del nodo cámara	13 días	mar 19/04/2	dom 01/05/22
22	**Sprint 7**	14 días	lun 02/05/22	dom 15/05/22
23	HU12: Conexión nodo cámara - aplicación	6 días	lun 02/05/22	sáb 07/05/22
24	HU13: Funcionalidad cámara	8 días	dom 08/05/2	dom 15/05/22
25	**Sprint 8**	14 días	lun 16/05/22	dom 29/05/22
26	HU14: Pruebas del sistema	12 días	lun 16/05/22	vie 27/05/22
27	**Documentación del proyecto**	**34 días**	**lun 30/05/22**	**sáb 02/07/22**
28	**Revisión de la documentación**	**6 días**	**sáb 02/07/22**	**jue 07/07/22**

Figura B.2 Planificación final

ANEXO C: Manual de usuario

Anexo C.1 Introducción

MAHURI es un sistema que permite el control de ciertas acciones de robots a través de una aplicación móvil para dispositivos móviles Android. El sistema está conformado por la propia aplicación móvil, y dos nodos que se ejecutan en un ordenador que debe estar conectado al robot y a la misma red que la aplicación móvil. A continuación, se detallan, en los siguientes apartados, los pasos a seguir para poner en funcionamiento el sistema:

Figura C.1 Logo MAHURI

Figura C.2 Icono Aplicación móvil MAHURI

Anexo C.2 Hardware

Los elementos hardware necesarios para el correcto funcionamiento del sistema son:

- **Dispositivo móvil:**
  - Sistema operativo Android 6.0 o superior.
  - Aplicación MAHURI instalada.

- **Ordenador:**
  - Sistema operativo Ubuntu.
  - ROS2 instalado.
  - Nodo tele operador descargado y compilado.
  - Nodo cámara descargado.
- **Robot:**
  - Sistema operativo ROS2 o sistema operativo ROS ejecutando un bridge en el ordenador que permita transformar las órdenes de ROS2 a ROS.

Como se ha comentado con anterioridad, los tres dispositivos deben estar conectados a la misma red para el correcto funcionamiento del sistema. En el caso de que el robot cuente con su propia red inalámbrica, el ordenador y el dispositivo móvil deben estar conectados a la misma. Por otro lado, si el robot no cuenta con su propia red, este deberá estar conectado al ordenador a través del cable de red Ethernet, y la aplicación móvil y el ordenador deben estar conectados a la misma red inalámbrica.

Anexo C.3 Puesta en funcionamiento

El primer paso para poner en funcionamiento el sistema, consiste en instalar la aplicación móvil en el dispositivo en el que va a ser ejecutada. Para ello, es necesario, en primer lugar, descargar el proyecto del correspondiente repositorio de GitHub mencionado en el anexo A de este mismo documento. Una vez haya sido descargado se debe importar en Android Studio y cambiar las direcciones IP de las clases Connection (Figura C.3) y CameraConnection (Figura C.4) introduciendo la dirección IP del ordenador en el que se vayan a ejecutar los nodos. Una vez se hayan realizado estos cambios se debe conectar el dispositivo móvil con las opciones de desarrollador activadas al ordenador e instalar la aplicación a través de la opción que ofrece Android Studio. Una vez haya sido instalada, la aplicación aparecerá representada con el icono que aparece representado en la Figura C.2 de este mismo documento.

```
if (s == null) {
    //change it to your IP
    s = new Socket( host: "192.168.1.85",  port: 6000);
    writer = new PrintWriter(s.getOutputStream());
    Log.d( tag: "javaClass",  msg: "CONNECTED");
}
```

Figura C.3 Cambio IP Connection

```
if (s == null) {
    //change it to your IP
    s = new Socket( host: "192.168.1.85",  port: 6001);
}
```

Figura C.4 Cambio IP CameraConnection

A continuación, se deben descargar los dos nodos, tanto el tele operador como el de la cámara, de los repositorios indicados en el Anexo A de este documento. Antes de ponerlos en funcionamiento es necesario realizar algunos cambios en los nodos para que se adecúen al robot que se vaya a manejar y el entorno en el que se encuentre. En primer lugar, en el archivo subscriber.cpp se debe cambiar el nombre de los topics encargados de la tele operación y navegación adaptándolos a los del robot que se desee conectar al sistema (Figura C.5). En este mismo archivo se deben cambiar también las coordenadas de cada uno de los lugares configurados en el nodo para que coincida con las coordenadas del entorno en el que vaya a ser ejecutado (Figura C.6)

```
twist_pub_ = this->create_publisher<geometry_msgs::msg::Twist>("/cmd_vel", 10);

action_client_ = rclcpp_action::create_client<nav2_msgs::action::NavigateToPose>(this, "navigate_to_pose");
```

Figura C.5 Cambio topics nodo tele operador

```
if(mensj == "COCINA"){
  goal.pose.pose.position.x = 3.79;
  goal.pose.pose.position.y = 6.77;
  //goal.pose.pose.position.z = -1.9821582495221923e-06;
  goal.pose.pose.orientation.x = 0.0;
  goal.pose.pose.orientation.y = 0.00;
  goal.pose.pose.orientation.z = 0.99;
  goal.pose.pose.orientation.w = 0.12;
```

Figura C.6 Cambio coordenadas nodo tele operador

En cuanto al nodo de la cámara, simplemente se debe modificar en el archivo camera_node.py el nombre del topic en la parte indicada en la Figura C.7, de manera que coincida con el nombre del topic del robot que se vaya a conectar al sistema.

```
self.subscription = self.create_subscription(Image, '/camera/image_raw', self.listener_callback, 10)
self.subscription
```

Figura C.7 Cambio topic nodo cámara

Una vez se hayan realizado estos cambios, se deben seguir los pasos indicados en cada uno de los archivos README.md de los repositorios para compilarlos y ponerlos en funcionamiento una vez el robot haya sido conectado al ordenador. De esta manera, ya se podría ejecutar la aplicación en el dispositivo móvil y comenzar a hacer uso del sistema.