Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN TECNOLOGÍA INDUSTRIAL

TRABAJO FIN DE GRADO

PUESTA EN MARCHA Y ACTUALIZACIÓN DE PLATAFORMA ROBÓTICA MÓVIL BASADA EN ROS KINETIC

Alumno/Alumna: <Blázquez Muguerza, Eneko>

Director/Directora (1): <Irigoyen Gordo, Eloy>

Director/Directora (2): <Larrea Sukia, Mikel>

Curso: <2018-2019>

Fecha: <Bilbao, 14 de julio, 2019>

Índice

Resumen

La plataforma software ROS (Robot Operating System) es una herramienta para ayudar a la programación de robots y sensores, tanto simulados como reales. Mantener una correcta actualización del código posibilita seguir avanzando importantemente en el estudio de diferentes áreas. Las plataformas robóticas móviles, ofrecen además, una gran variedad de ventajas con la correcta implementación de dichos codigos.

El objetivo de este trabajo es la puesta en marcha y actualización de una plataforma móvil utilizando el software ROS en la versión Kinetic. Dicha plataforma se encontraba sin la actualización correspondiente de su software, por tanto, en el proyecto se ha llevado a cabo la actualización mencionada. Así mismo, en este trabajo se ha utilizado la plataforma robótica móvil para la navegación manual y simulada. Lo que ha ofrecido la posibilidad de generar un mapa del entorno en el que se ubica nuestro robot. Para esta tarea en concreto ha sido fundamental la implementación de los sensores adecuados.

Palabras clave: ROS (Robot Operating System), Summit, robot, control manual, navegación.

Abstract

The software platform ROS (Robot Operating System) is a tool to help the programming of robots and sensors, simulated or real. Maintaining a correct code update makes it possible to continue making important progresses in the study of different fields. Mobile robotic platforms also offers a wide variety of advantages with the correct implementation of these codes.

The objetive of this work is the start-up and update of a mobile platform using the software ROS in the Kinetic version. This platform was without the corresponding update of its software, therefore, the previously mentioned update has been carried out in the proyect. In the same way, in this work the mobile robotic platform has been

used for manual simulated navigation. What has offered the possibbility of generating a map of the environment in which our robot is located. For this particular task, the implementation of the appropriate sensors has been fundamental.

Key words: ROS (Robot Operating System), Summit, robot, manual control, navigation.

Laburpena

ROS (Robot Operating System) software plataforma bai simulatutako, bai benetako errobot eta sentsoreen programazioan laguntzen duen tresna da. Kodearen eguneratze egoki bat izateak alor ezberdinetan aurrerakuntza garrantzitsuak egitea ahalbidetzen du. Errobot plataforma mugikorrek, gainera, abantaila aukera zabal eskaintzen dituzte kode hauen inplementazioarekin.

Lan honen helburura ROS softwarearen Kinetic bertsioa erabiliz plataforma mugikor baten abiaraztea eta eguneratzea lortzea da. Plataforma hau bere softwarearen eguneratzea egin gabe zegoen, beraz, proiektuan zehar eraman da aurrera. Halaber, lan honen eskuzko zein simulatutako nabigaziorako errobot plataforma mugikorra erabili da. Honi esker, gure errobota aurkitzen den inguruko mapa bat sortzea lortu da. Zeregin honetarako zehazki beharrezkoa izan da sentsore egokien inplementazioa.

Gako-hitzak: ROS (Robot Operating System), Summit, errobota, eskuzko kontrola, nabigazioa.

Ilustraciones

1. RESUMEN EJECUTIVO

- Alumno: Eneko Blázquez Muguerza.
- Director (1): Eloy Irigoyen Gordo.
- Director (2): Mikel Larrea Sukia.
- Título: Puesta en marcha y actualización de plataforma robótica móvil basada en ROS Kinetic.
- Resumen: Se ha realizado una puesta en marcha y actualización de una plataforma móvil terrestre modelo Summit, de Robotnik, en ROS Kinetic. Para la conducción manual del modelo simulado, se le ha añadido un sistema de sensorización, así como otros parámetros para la generación de mapas del entorno.
- Palabras clave o etiquetas: ROS (Robot Operating System), Summit, robot, control manual, navegación.

2. INTRODUCCIÓN

La robótica es un campo de la ingeniería en el que la investigación nunca llegará a su fin. La rápida evolución de las tecnologías que integran los robots hace que haya un sinfín de nuevas líneas de desarrollo que ofrecen diferentes alternativas de futuro tanto en el campo de la medicina, como en la tecnología aeroespacial, y un largo etcétera.

Hoy en día existen diversas herramientas que ayudan a que distintos robots y plataformas ganen autonomía durante su ejecución. Este es el campo en el que se basa este proyecto: el control inteligente. Casualmente, uno de los campos más beneficiados por el avance tecnológico. Este ámbito de la robótica trata de dotar a las máquinas de inteligencia artificial, para que sean capaces de realizar determinadas tareas sin necesidad de control manual.

Un mayor o menor rendimiento de un sistema inteligente, depende directamente de la calidad del software utilizado. Para ese concreto sistema, son las plataformas software los que permiten una gestión precisa y concreta de cada elemento. Tanto el control inteligente como la puesta en marcha de los sistemas robotizados, requieren de actualizaciones en dicho software. La puesta en marcha de los diferentes sistemas inteligentes suponen de cambios significativos ajustados a las necesidades que van surgiendo a lo largo del desarrollo. Además, es conveniente que estas modificaciones se realicen con el fin de mejorar la comprensión del código para facilitar así futuras líneas de trabajo.

El objetivo principal de este Trabajo Fín de Grado es optimizar el desarrollo e implementación del software ROS (Robot Operating System) para conseguir el aprovechamiento máximo de todos los recursos de los que se dispone.

3. CONTEXTO

En lo que a este proyecto se refiere, se ha trabajado a fondo con el software ROS (Robot Operating System), principal método utilizado en la programación de robots. Este es un software libre que proporciona las herramientas necesarias para el control de todo tipo de robots y sensores. Además, los propios creadores del software, ofrecen un fácil acceso al estudio y comprensión del mismo, aportando una gran capacidad de crear programas nuevos y de innovar, el cual le dota de gran versatilidad.

Con este software, se ha realizado (como más adelante detallaremos) un análisis para la comprensión y estudio de las herramientas de programación, y un posterior desarrollo de determinados paquetes para la navegación de un robot modelo Summit, de Robotnik.

Este proyecto se ha basado principalmente en el desarrollo y acondicionamiento del software. El eficaz y rápido análisis y estudio de las herramientas de programación, se ha conseguido gracias a la gran labor realizada por parte de los desarrolladores del software, como se verá a lo largo del proyecto. Por tanto, hacen más sencillo el desarrollo y comprensión de los paquetes, tanto los creados para el proyecto como los ya existentes e implementados en él.

La actualización del software es un paso necesario que sirve como base de futuros proyectos. La puesta en marcha que se ha hecho para esta plataforma móvil generará una línea de futuras investigaciones a seguir en este ámbito. Al tratarse de tecnología en constante desarrollo, es importante la elección de la versión de software en la que se desea trabajar. Se decidió, por tanto, utilizar la versión ROS Kinetic, pues esta se encuentra en una fase terminada, la cual no requerirá de incorporaciones nuevas. Si se hubiera querido trabajar en una versión más actual, como puede ser ROS Melodic, podría ocurrir que los creadores incorporasen nuevos parámetros. Lo cual supondría tener que volver a empezar o modificar constantemente el trabajo.

En este proyecto en concreto, se ha decidido trabajar con una simulación de robot modelo Summit de Robotnik. Dado que en el laboratorio del departamento de ingeniería de sistemas y automática tienen dicho modelo de robot, será de gran utilidad este estudio, el cual dará lugar a la creación de una base para futuras investigaciones.

Ilustración 1: robot móvil Summit, de Robotnik

El mencionado robot móvil, como se aprecia en la ilustración, es un coche todoterreno de tamaño medio, el cual lleva un ordenador incorporado. Dicho coche es de gran utilidad ya que se le puede acoplar en su chasis un sistema de sensorización. Además, el gran tamaño de sus ruedas supone una ventaja a la hora de navegar por terrenos irregulares y de difícil acceso, es por esto que se decidió utilizar este robot y no otros modelos existentes en el laboratorio.

Al ser un trabajo basado en la simulación, todo el proyecto se ha realizado en un ordenador remoto, el cual tiene un teclado “qwerty”. Este tipo de teclado es un elemento fácilmente accesible hoy en día desde cualquier ordenador, tablet, teléfono móvil, etc. Es por eso que se ha creído oportuno realizar el control remoto a través de él, pues al ser un elemento tan común, es fácil y sencillo y lo dota de una versatilidad a la hora de dirigir el robot.

Además, se decidió utilizar la simulación de la cámara kinect concretamente por distintos motivos. Esta cámara proporciona no solo la capacidad de visión, sino que además dota al proyecto de los recursos necesarios para generar mapas del entorno simulado. Otra de las razones principales, fue el hecho de que ya se hubiese trabajado anteriormente con este sensor. Y como se ha mencionado anteriormente en la elección del modelo del robot, servirá como base para futuras investigaciones.

4. OBJETIVOS

Para concretar los objetivos del proyecto se ha decidido un objetivo principal, y posteriormente los objetivos parciales que han hecho posible llegar a ese objetivo principal.

4.1. OBJETIVO PRINCIPAL

La meta que se puso para el final del proyecto era tener un ordenador remoto con una versión de ROS Kinetic. La simulación del robot Summit además tendría integrados unos sensores con los que poder analizar el entorno, con lo que podríamos enviar directrices para que el robot navegara autónomamente.

Para llegar a este objetivo final se han fijado varios objetivos parciales.

4.2. OBJETIVOS PARCIALES

Estos son los objetivos que marcamos al inicio del trabajo. No necesariamente implica tener que realizarlos en el orden mostrado, aunque sí es conveniente.

Objetivo parcial 1: Instalación y análisis de ROS Kinetic

El primer objetivo parcial es adquirir conocimientos sobre Ubuntu y ROS, ya que son las principales herramientas de trabajo a lo largo de todo el proyecto. Con ellas se han desarrollado los programas necesarios para poder realizar todas las tareas.

Para instalar ROS se seguirán las instrucciones y tutoriales que aparecen en su página web. [1]

No se ha mencionado anteriormente, pero se realizará una instalación limpia, formateando primero el ordenador e instalando Ubuntu y ROS. Sería muy interesante poder hacer esta instalación anteriormente señalada en un robot real que tenga un ordenador incorporado en el. El procedimiento sería el mismo, y los beneficios que aportaría serían mayores.

Cabe mencionar que la correcta y completa instalación de todo el sistema operativo y el software son un paso clave para que todo funcione en condiciones óptimas.

Objetivo parcial 2: Simulación del robot Summit

El siguiente objetivo parcial es hacer una simulación con el ordenador remoto para seguir familiarizándonos con el software ROS Kinetic. Para ello clonamos de la pagina de GitHub los repositorios que subió la empresa que se dedica a fabricar los robots Summit, Robotnik. [2]

Vale la pena mencionar que estos repositorios son gratuitos y están a la disposición de cualquier persona. Una vez descargados todos los archivos necesarios se procede a la simulación del robot Summit. [3]

Objetivo parcial 3: Integración de un sistema de sensorización

El tercer objetivo parcial se basa en la implementación y programación de distintos sensores en el modelo simulado del robot Summit.

Existen muchos tipos de sensores, hay sensores que únicamente constituyen un láser para poder percibir el entorno, hay otros que tienen cámaras integradas para poder tener visión de lo que el modelo de robot tenga enfrente, etc. La finalidad que tiene este objetivo es poder hacer que el robot reciba una serie de estímulos que le permitan navegar en un entorno simulado.

La elección de estos sensores se detallará más adelante cuando se expliquen los parámetros necesarios para las diferentes actividades que se desea que haga el robot, ya sea mapear un entorno, navegar, explorar... Entre estos podrían incluirse la cámara kinect, el sensor hokuyo, el sensor kobuki, el sensor imu, ... entre otros.

5. BENEFICIOS

Para analizar los diferentes beneficios que ha supuesto la realización de este proyecto, se han organizado tres grupos principales, que son los siguientes: beneficios técnicos, beneficios económicos y beneficios sociales.

Cabe mencionar que los beneficios no abarcan únicamente lo concreto del proyecto, por si en un futuro se desea trasladar toda la base de desarrollo que contiene este proyecto para conectar el ordenador remoto en el que hemos trabajado con una plataforma real del robot Summit. Abriéndonos a este campo es donde trabajaremos todo lo relacionado a los beneficios, pues al ser una actividad más práctica ofrece una amplia variedad de beneficios añadidos.

Ilustración 2: robot móvil Summit, de Robotnik, navegando por un terreno irregular

5.1 BENEFICIOS TÉCNICOS

Los beneficios técnicos a su vez, se pueden dividir en dos apartados dependiendo del enfoque: por una parte, los beneficios técnicos que ofrece el resultado final del proyecto, es decir, el modelo del robot simulado con las implementaciones realizadas durante todo el trabajo; y por otra parte, el valor de toda la documentación generada a partir de la investigación realizada sobre control inteligente con ROS.

Como ya se ha comentado en anteriores apartados, la integración de ROS en el Summit ofrece una amplia variedad de posibilidades en cuanto a aplicaciones.

El Summit es un coche que puede moverse por terrenos difíciles. No tiene dificultades para moverse por zonas no asfaltadas, gracias a sus grandes ruedas y a la gran flexibilidad de su amortiguación. Gracias a esto se pueden idear aplicaciones para llevar a cabo tareas en lugares naturales o de difícil acceso para la mayoría de robots, sin mencionar ya para personas humanas.

Gracias a ROS podemos crear e instalar en nuestros equipos los diferentes programas y paquetes que sean necesarios para las tareas que se quieran realizar con el robot. Vamos a citar unos ejemplos de estas tareas: exploración, ya que como hemos dicho anteriormente, gracias al control remoto y a los sensores incorporados, podemos ver el entorno que va encontrando a su paso; seguimiento, al igual que en el caso anterior, utilizando la cámara podemos realizar un seguimiento a una persona o al objetivo que fuera preciso; vigilancia, ya que se le puede ordenar al robot que navegue una y otra vez por una ruta concreta de un mapa y así grabar el entorno de toda la ruta. Habrá muchas más posibilidades abiertas gracias a esta plataforma.

En cuanto al otro enfoque de los beneficios técnicos, hay que comentar que, al integrar el Summit con el software de ROS, se han producido una serie de modificaciones en el uso de dicho software para el control y navegación de un robot. Todo lo que se ha

aprendido durante la realización de este proyecto, puede ser información de gran utilidad para trabajos similares, aunque se realicen con robots y elementos diferentes.

Con las anotaciones que se han ido tomando a lo largo del proyecto, se ha creado un manual de usuario. Con él, podemos ver la forma de utilizar las aplicaciones específicas creadas a lo largo de este trabajo. Este manual se adjuntará como documento adicional.

5.2 BENEFICIOS ECONÓMICOS

Como ya se ha explicado en anteriores apartados, el objetivo de este proyecto ha sido crear una plataforma inteligente que ofrece una amplia versatilidad en lo que a sus diferentes usos se refiere.

Empresas de muy diferentes ámbitos (como por ejemplo sanidad, seguridad, etc.) podrían beneficiarse de las ventajas y las posibilidades de este trabajo. Esto hace que los beneficios potenciales del trabajo puedan resultar muy interesantes.

Para empezar, el proyecto tiene mucho interés desde la perspectiva de la investigación. Los beneficios sociales (que se expondrán en el próximo apartado) que puede ofrecer el trabajo final hacen que sea presumible una inversión de dinero público para la adquisición de ejemplares para diferentes usos en administraciones y servicio públicos.

Hay que recalcar que el proyecto tiene un fuerte componente de investigación. Tras la realización de este proyecto, hay varias líneas de investigación a seguir, relacionadas también con el Summit y ROS. Por lo tanto, presumiblemente el trabajo seguirá adelante, y por el momento, los beneficios económicos pueden considerarse secundarios. Aunque, como se ha explicado, está claro que tenemos una plataforma inteligente y de gran versatilidad, la cual puede dar lugar a ideas y líneas de desarrollo que sean económicamente fructíferas.

5.3 BENEFICIOS SOCIALES

En primer lugar, hay que decir que la investigación que se ha llevado a cabo con este proyecto puede ser de gran ayuda para futuros trabajos que se realicen en este ámbito, y por tanto, puede servir como base para la realización de otros trabajos que, debido a su similitud con este, proporcionarán similares beneficios sociales. Es decir, esta investigación proporcionará indirectamente beneficios sociales, ayudando a realizar otros trabajos que serán útiles para casos más específicos.

Ahora se explicarán los beneficios del trabajo en sí mismo. El resultado final es la simulación de un modelo de robot remotamente manejable, y también con un cierto grado de autonomía. Además, tiene gran versatilidad en cuanto a sus movimientos y lugares por los que se pueda desplazar, si se llevara a cabo una conexión con un robot real. Teniendo estos puntos en cuenta, está claro que hay tres campos en los que resultaría de mucha utilidad.

El primero es la exploración. En terrenos de difícil acceso, o ambientes contaminados por radiación (por ejemplo, pongamos que tras un accidente), el robot es ideal para navegar por estos terrenos en la búsqueda de personas perdidas, atrapadas o que necesiten algún tipo de ayuda. Con el debido desarrollo, el robot podría tener las características necesarias para realizar esta tarea, y si su rendimiento es bueno, cabe esperar que diferentes instituciones privadas como públicas, encargadas de exploración y/o salvamento de personas en circunstancias como las mencionadas se interesen.

El segundo es la vigilancia, el cual tiene cierta relación con el primero. Empresas o instituciones públicas que requieran vigilancia constante, podrán interesarse por el robot, no como sustitución de las cámaras de vigilancia, las cuales son claramente muy útiles, sino como complemento para dichas cámaras o los diferentes sistemas de vigilancia. La ventaja del robot es que puede moverse y vigilar diferentes lugares, sitios donde resulte complicado instalar una cámara o lo que fuera necesario. Sería una cámara de vigilancia móvil.

Por último, tenemos el cuidado y atención de personas. Teniendo en cuenta su autonomía, el robot es una herramienta excelente para este campo. Puede servir para asistir y guiar a personas ciegas o con visión limitada, puede servir también de ayuda a personas con movilidad reducida, y un largo etcétera.

El aprovechamiento y exploración de todas estas posibles aplicaciones será un trabajo posterior a la realización de este proyecto.

6. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Con el fin de solucionar el problema que se nos plantea, la navegación simulada y autónoma de una plataforma móvil terrestre, al inicio del trabajo se visualizará el camino a seguir de la manera más eficiente posible. Para ello se organizará una serie de tareas a seguir, las cuales expondremos en este capítulo.

6.1 CONFIGURACIÓN DE LA PLATAFORMA ROS

Previo al desarrollo del trabajo, se realizará una toma de contacto con la plataforma ROS, realizando los diversos tutoriales que nos proporciona la misma página web, como hemos comentado previamente. También se ha mencionado que los objetivos parciales no tendrían por qué ser resueltos en el orden indicado. No obstante, al ser esta solución de carácter básico a la hora de realizar el trabajo, es imperativo que el trabajo se lleve a cabo por el primer objetivo parcial.

6.2 CONFIGURACIÓN DE LOS PAQUETES PARA LA SIMULACIÓN DEL ROBOT

La simulación del robot es una tarea necesaria para la solución del problema final. Es un pilar fundamental del estudio final, el cual podría, pero no debería compaginarse con el resto de las soluciones. Por esto, y como más adelante resaltaremos a través de un esquema, será esta la segunda tarea a seguir.

6.3 GENERACIÓN DE LOS PARÁMETROS PARA EL MODELO DEL ROBOT

Lo que no sabíamos que ocurriría antes de empezar el trabajo era que encontraríamos tantas dificultades a la hora de generar los parámetros del robot. Esta tarea se ha ido solapando con las siguientes dado el desconocimiento de su magnitud.

Como se expondrá en el siguiente punto, al incorporar los distintos sensores, estos generan una serie de dificultades que van de la mano con la solución. Un ejemplo puede ser, al principio solo se contempló la opción de incorporar la cámara kinect, hasta que se detectó que no era suficientemente grande el rango del láser y se tuvo que implementar con otro láser más potente, el hokuyo, y así añadir mayor alcance a la hora de “mapear” el entorno.

6.4 INCORPORACIÓN DE SENSORES

Al inicio del trabajo se decidió utilizar la cámara kinect, la cual era idónea para la visión, el mapeo, la navegación y la navegación autónoma. Con un solo sensor se obtienen múltiples añadidos, los cuales generan unos contratiempos a solucionar tanto en este

apartado como en el anterior, y cuya solución fue también incorporar el láser hokuyo por su eficiencia a la hora de generar el mapa del entorno.

Por otro lado, tenemos el problema de programación, que incluye la colocación del modelo simulado del sensor en el robot simulado. Este recibe y genera unos parámetros que necesitan del punto anterior para su solución. Por ejemplo, el sensor kinect genera visión 2D, lo cual es una dificultad a la hora de querer mapear y navegar, puesto que ambas funciones necesitan de un láser óptico.

6.5 SOLUCIÓN FINAL

Para terminar expondremos aquí la solución final del desarrollo del trabajo de fin de grado. El objetivo final sería que el robot simulado respondiera a todos los comandos que se le ordenen para la correcta navegación remota tanto en un terreno conocido como por conocer; para esto es imprescindible la buena ejecución de todas las soluciones propuestas anteriormente. Es verdad que algunas de ellas pueden llevarse a cabo solapando el trabajo, pues es imposible finalizar completamente la una sin la otra, pero como punto final tanto las directrices para la navegación como la capacidad de visión del sensor del robot son obligatorias para el planteamiento de trabajo que hicimos al principio de este.

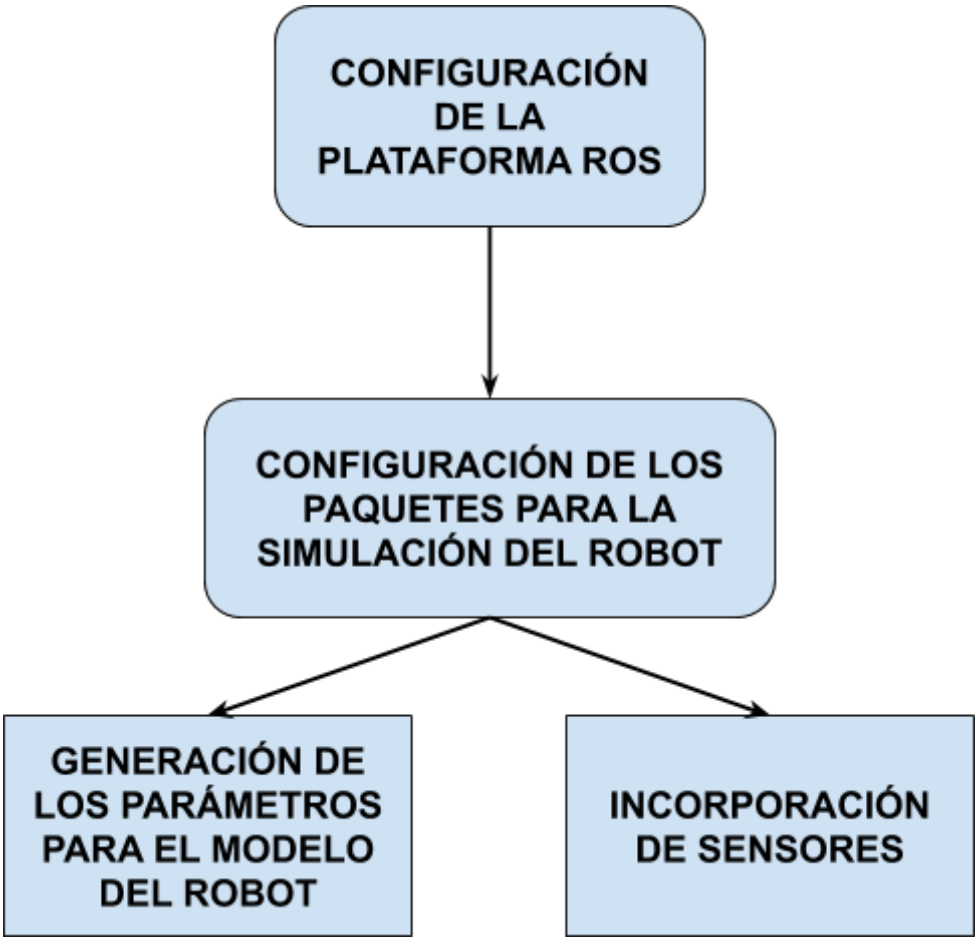


Ilustración 3: esquema orientativo de la descripción de la solución propuesta

7. RESUMEN DEL DISEÑO

El diseño de nuestro robot inteligente se ha basado principalmente en el desarrollo de software mediante ROS. En este apartado se va a resumir el proceso de desarrollo de dicho software, explicando las razones y necesidades que han empujado cada paso del proyecto.

Este apartado se divide en cuatro puntos, los correspondientes a los objetivos parciales y como consecuencia al resultado final. Algunos puntos a su vez están divididos en varias partes, dependiendo de los pasos que se hayan dado para conseguir cada objetivo.

7.1 INSTALACIÓN Y ANÁLISIS DE ROS KINETIC

La herramienta de trabajo principal de este proyecto ha sido ROS. Se trata de un software diseñado exclusivamente para control de robots. Es un software libre, y se puede descargar e instalar muy fácilmente desde su página web.

Ya que anteriormente se trabajó con el robot Summit con el que vamos a trabajar y se abandonó en una versión de ROS más antigua (ROS fuerte), nos ha parecido idóneo actualizar este programa a la versión nombrada. Puesto que al no ser la última versión, ya está completa y no van a "parchearla".

Al ser la principal herramienta de trabajo, se decidió un primer paso muy importante que sería analizar su funcionamiento y características de trabajo. La página web ofrece una serie de tutoriales de nivel básico, los cuales resultan muy útiles para el aprendizaje del sistema.

Además de eso, en este primer objetivo parcial se han instalado Ubuntu y ROS en el ordenador remoto. De esta forma tenemos el software necesario para seguir trabajando en las siguientes tareas.

Instalación de Ubuntu en el ordenador remoto

Para poder empezar a trabajar con ROS Kinetic, primero ha habido que instalarlo en el ordenador remoto. ROS Kinetic está diseñado para trabajar sobre el sistema operativo

Ubuntu en la versión 16.04, por lo tanto el primer paso ha sido instalar Ubuntu en la versión correspondiente. Este es un punto a favor desde el punto de vista económico, puesto que tanto Ubuntu como ROS son software libre, y se puede descargar de forma rápida y simple de internet.

Una vez instalado el sistema operativo en el ordenador remoto, se instalan todas las actualizaciones para que el sistema funcione correctamente. Tras esto ya tendríamos el sistema operativo totalmente listo para utilizar.

No se ha mencionado anteriormente, pero se realizará una instalación limpia, formateando primero el ordenador e instalando Ubuntu y ROS. Sería muy interesante poder hacer esta instalación anteriormente señalada en un robot real que tenga un ordenador incorporado en el. El procedimiento sería el mismo, y los beneficios que aportaría serían mayores.

Instalación de ROS en el ordenador remoto

Habiendo realizado la instalación del sistema operativo, el siguiente paso es la instalación del software ROS Kinetic. La instalación es sencilla y podemos encontrar todos los pasos a seguir en la página web de ROS. [4]

Siguiendo esa guía se ha instalado ROS Kinetic en el ordenador remoto.

Análisis de ROS

Este ha sido un punto vital para el trabajo. Como se ha expuesto anteriormente, ROS es la principal herramienta del proyecto. Por lo tanto, desde el principio se recalcó el análisis de ROS como algo totalmente necesario para poder trabajar de una manera más eficiente.

La página web de ROS ofrece una serie de tutoriales, como hemos comentado antes, de carácter básico para aprender a funcionar con ROS desde cero. Los tutoriales han resultado de gran utilidad a la hora de completar este punto, contienen tanto explicaciones como ejemplos y ejercicios que realizar para ir entendiendo mejor el funcionamiento del software. [5]

Cabe mencionar que tras realizar los tutoriales, se formó una idea general del funcionamiento de ROS, pero cuando realmente se ha aprendido a utilizarlo ha sido al empezar a trabajar, ya que se han encontrado muchos obstáculos concretos de nuestro trabajo en el camino que ha habido que sortear. Al no ser obstáculos tan genéricos como la ayuda que ofrecen los tutoriales, han sido de gran ayuda a la hora de comprender realmente todo el carácter de ROS.

7.2 SIMULACIÓN DEL ROBOT SUMMIT

Dentro de la simulación del robot Summit, se ha creído oportuno dividir el objetivo en dos fases. La primera de ellas es el estudio y análisis de los paquetes de simulación del robot, pues ya advertimos en el anterior punto que llegados aquí es donde más aprendizaje se realiza puesto que es cuando se empieza verdaderamente a trabajar con el software ROS y no antes durante los tutoriales. La segunda fase sería la incorporación de un control remoto para nuestro robot.

Estudio y análisis de los paquetes de simulación del robot Summit

Como venimos haciendo antes de los tutoriales, y siendo también el código del robot Summit un código libre y gratuito, se procede a la clonación de los paquetes de mano de la empresa encargada de fabricar dicho robot, Robotnik. Estos paquetes se encuentran entre sus repositorios en la página de GitHub, además la última versión accesible está escrita en una versión anticuada de ROS llamada fuerte.

Esto último sugiere que para el correcto funcionamiento en nuestra versión, ROS Kinetic, es necesario hacer unos ajustes. Al clonar estos paquetes en nuestro espacio de trabajo llamado "catkin_ws", es necesaria la adición de otros paquetes que se pueden encontrar también de manera libre en la página de GitHub. Para ello, es necesario comprender al completo todos y cada uno de los paquetes que se están instalando en el ordenador remoto, para qué sirve cada archivo ".launch", cómo se genera cada nodo, la necesidad de cambiar determinados parámetros...

Para el lanzamiento de este paquete, vale con aplicar lo siguiente en el terminal:

roslaunch summit_gazebo summit_office.launch

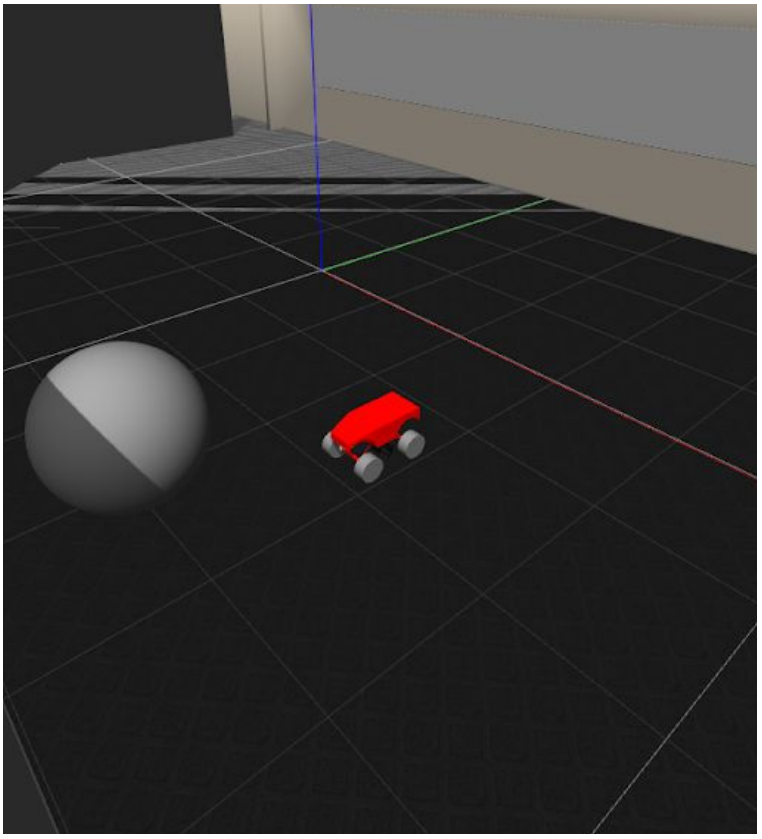


Ilustración 4: modelo simulado visto en el programa Gazebo de ROS

Esto abrirá el programa gazebo, con el modelo del robot Summit simulado en el entorno de la oficina de Robotnik de Valencia. Añadieron otro lanzador el cual abría el simulador gazebo sin ningún tipo de entorno.

Toda la información detallada se redactará en el manual de usuario que se adjuntará junto con el proyecto.

Implementación de los paquetes para el control remoto de la simulación

A parte de los paquetes básicos para la simulación del modelo del robot, son necesarios también otros tantos para conseguir lo que se propuso como objetivo final. Uno de estos paquetes es para el control remoto del robot, al igual que otros pueden ser para la incorporación de los modelos de los sensores que se vayan a utilizar. Es por esto que en el anterior capítulo hemos mencionado que la solución al objetivo parcial no es algo que pueda comenzar y acabar sin solaparse con otro tipo de objetivos como los que acabamos de mencionar.

El problema al que nos enfrentamos al clonar el código fue que habían cambiado la manera que tiene el robot para navegar, pues utiliza un mecanismo de dirección Ackermann, el cual se utiliza para que “El cuadrilátero de Ackermann o de Jeantaud se aplica en la actualidad y universalmente en todos los coches. Es el sencillo mecanismo que realiza la unión entre los ejes de las ruedas directrices del vehículo. A fin de que pueda producirse un cambio de dirección sin que exista deslizamiento de las ruedas sobre el suelo, es necesario que los ejes de todas las ruedas pasen por un mismo punto” [6]. Actualmente la mayoría de plataformas terrestres móviles utilizan una rotación en su base o una rotación de los neumáticos que los exime de utilizar este tipo de mecanismo.

Es por esto que se decidió únicamente incorporar un tipo de control remoto y que esté fuera con el teclado del ordenador remoto. Es cierto que se pueden implementar otros

tantos como el del propio mando del robot Summit o incluso un Wiimote. Esto supondría un esfuerzo adicional que no era lo que buscamos, pues nuestro trabajo final es la navegación autónoma del robot en un entorno simulado.

Para la implementación del control remoto, se hizo una búsqueda en la página web de GitHub la cual nos llevó a un repositorio de código libre que servía para implementar un control remoto vía teclado de ordenador [7]. Como se ha hecho en anteriores casos, se pasó a la clonación de los paquetes a nuestro espacio de trabajo. Se han tenido que realizar una serie de cambios manuales en el paquete, en concreto el "topic" publicado por el nodo del teclado no coincide con el recibido por el control del robot. Para arreglarlo se a tenido que editar el "topic" publicado por el teclado llamado "cmd_vel" y ha sido modificado por "/summit_robot_control/command". En concreto este "topic" es el encargado de transmitir la información del movimiento del teclado al modelo de simulación.

Lanzando ese archivo, se consigue controlar el robot utilizando el teclado del ordenador remoto.

Cabe mencionar que se hicieron unos cambios en el comportamiento del control del robot, pues la velocidad máxima que admitía el modelo era muy pequeña comparado con el entorno que se requería recorriera el robot Summit. Estos cambios aparecerán en el manual de usuario adjunto al documento del trabajo.

7.3 INTEGRACIÓN DEL SISTEMA DE SENSORIZACIÓN Y ANÁLISIS DEL ENTORNO

Para el análisis del entorno simulado, se creyó oportuno al principio del trabajo, la instalación en el modelo del robot Summit un sensor que pudiera actual como láser para mapear, y a la vez como cámara para poder ver y detectar objetos utilizando lo que se denomina nube de puntos.

Instalación de la cámara kinect en el modelo del robot

El primer paso, como hemos venido siguiendo, es el de la clonación del paquete del sensor, también código libre y suministrado por la empresa Robotnik, como los anteriores. En este punto tuvimos una gran ayuda pues Robotnik no ofrece únicamente el código de la cámara kinect, sino que en su paquete llamado /robotnik_sensors, el cual se puede encontrar entre sus repositorios en GitHub, el cual nos proporciona el código de muchos sensores más.

Primero de todo hay que modificar los archivos correspondientes para la creación del modelo simulado e implementar ahí el código de la cámara kinect conseguido por Robotnik y casando la cámara con los nodos correspondientes. Además, es necesario elegir la correcta colocación de la cámara en el modelo del robot.

Esta cámara al ser lanzada por el simulador de ROS, generará una serie de "topics" que se utilizaran para visualizar y recibir información del entorno simulado. Todas las modificaciones detalladas serán expuestas en el manual de usuario que se entregará junto con el trabajo del proyecto.

Entre los "topics" que genera esta cámara no se encuentra ninguno que sea del tipo láser, pues todos ellos son para la creación de una nube de puntos o para visualizar la imagen de la cámara. Es por esto que se necesita de un paquete que transforme esta nube de puntos generada en 2D a un laser óptico, el cual nos va a permitir generar un mapa. Esto puede ser muy interesante a la hora de implementar un sistema de navegación autónoma en un futuro.

Implementación del paquete para la optimización del láser

Como hemos marcado en el anterior apartado, es preciso de un paquete que modifique la nube de puntos que genera la cámara kinect para obtener un laser óptico con el que mapear el entorno. Para ello teníamos distintas opciones, utilizar el paquete llamado /depthimage_to_laserscan, otro llamado /pointcloud_to_laserscan y por último /kinect_laserscan. Finalmente decidimos utilizar el primer mencionado, pues fue el recomendado por el profesor. Como se podrá ver en el manual de usuarios expuesto, son necesarios unos cambios en el paquete /summit_sim que es el encargado de cargar el modelo del robot con la cámara kinect, pues este crea unos "frames" los cuales tienen que recogerse en el paquete /depthimage_to_laserscan. Una vez hechos todos los cambios se ha implementado el lanzador del conversor a láser, en el lanzador principal de la simulación, con lo cual al usar el archivo ".launch" mencionado en el primer apartado, este además de cargar el modelo del robot Summit y el entorno de la oficina de Robotnik, cargaría en la parte superior del robot la cámara kinect.

A continuación se adjunta el código del archivo ".launch" que se ha creado para la conversión de la nube de puntos a laser óptico:

```
<launch>
 <node pkg="depthimage_to_laserscan" type="depthimage_to_laserscan"
name="depthimage_to_laserscan" args="load
depthimage_to_laserscan/DepthImageToLaserScanNodelet">
   <remap from="camera_info" to="/kinect/depth/camera_info" />
   <remap from="image" to="/kinect/depth/image_raw" />
   <remap from="scan" to="/depth_scan" />
   <param name="scan_height" value="10" />
   <param name="range_max" value="30" />
   <param name="output_frame_id" value="camera_depth_frame" />
 </node>
</launch>
```

Cabe mencionar que no todo el paquete fue creado a la hora de implementar esta función. Se copió la mayor parte de la información de un repositorio de código libre de la página GitHub. [8]

Análisis y estudio para la generación del mapa del entorno simulado

Una vez se tiene un láser integrado al modelo del robot, se puede proceder a la generación del mapa del entorno simulado. Para ello, y con ayuda de los tutoriales gratuitos ofrecidos vía YouTube por el canal The Construct gracias a la plataforma dedicada a la enseñanza del software ROS, se genera un paquete capaz de usar este láser convertido para mapear el entorno. [9][10]

Este paquete que se creó, se subió a la plataforma GitHub con código libre bajo el nombre de “summit_mapping”, de manera que todo aquel que quiera utilizarlo pueda. Dentro del paquete se generaron dos carpetas, una para almacenaje de los mapas generados por el láser, y otro para almacenar el lanzador del programa.

Además, gracias a The Construct también creamos un paquete con un lanzador que tuviera la capacidad de darle al robot un punto de referencia a la hora de localizarlo. Por ejemplo, pongamos que el robot se pierde en el mapa, nosotros, gracias a este paquete creado podríamos sugerirle al robot una zona en la cual creemos que esta para que el se organice después de haberse perdido. Este paquete utiliza el mapa generado por el paquete anterior y los datos obtenidos gracias a la nube de puntos generada por la cámara kinect.

Como el paquete anterior, también se subió a la plataforma GitHub con código libre bajo el nombre de “summit_localizer”, para el libre acceso de todos los usuarios.

A continuación se detallaran los comandos a ejecutar para el uso de estos archivos, empezando por el referido al “mapeo” del entorno:

```
roslaunch summit_gazebo summit_kinect.launch
Ctrl+Shift+T
rosrun rviz rviz
Ctrl+Shift+T
roslaunch summit_mapping mapping.launch
```

Una vez conseguido el mapa que deseábamos, habría que ejecutar el guardado de mapa con el siguiente comando en un terminal nuevo:

```
roscd summit_mapping/config
rosrun map_server map_saver -f <nombre del mapa>
```

Para ahora ejecutar el archivo de localización del robot deben seguirse estos comandos:

```
roslaunch summit_gazebo summit_kinect.launch
Ctrl+Shift+T
rosrun rviz rviz
Ctrl+Shift+T
roscd summit_mapping/config
rosrun map_server map_server <nombre del mapa>.yaml
roslaunch summit_localizer localizer.launch
```

Un inconveniente que tuvimos en este proceso fue el observar que la cámara kinect no tenía suficiente rango de visión, pues el láser únicamente puede abarcar un abanico de unos 30º al igual que la nube de puntos. Esto genera un gran problema a la hora de generar el mapa, puesto que el robot no encontraba el final de las habitaciones y reseteaba su localización. Es por esto que se creyó de vital importancia la integración de otro sensor que tuviera mas rango de visión. Desgraciadamente, este otro sensor

no tendría la capacidad de visión ni de crear nube de puntos como tiene la cámara kinect, pero no sería necesario, pues al ser una simulación, podrían integrarse ambos sensores.

Cabe mencionar que se modificó el rango de alcance del laser convertido modificando el archivo del paquete /depthimage_to_laserscan en el archivo ".launch" pasando de 3 a 30 su rango máximo.

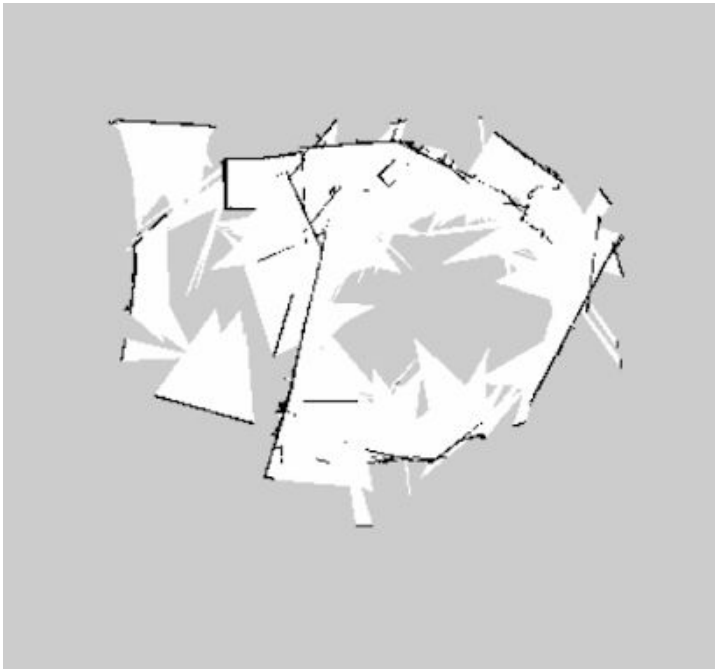


Ilustración 5: mapa generado por la cámara kinect en el programa RViz de ROS

Instalación del sensor hokuyo

El sensor hokuyo, a diferencia de la cámara kinect, únicamente sirve para generar un laser óptico, que abarca unos 330º y con un alcance mayor al de la cámara kinect (el cual luego también se modifico). El único objetivo de este sensor es generar un mapa del entorno que luego sirva para poder navegar autónomamente, es decir, el objetivo que tenía anteriormente la cámara kinect.

Al principio de este capítulo he mencionado que Robotnik proporciona no solo el código libre de la cámara kinect en su paquete /robotnik_sensors, sino que ofrece muchos más, entre otros, el sensor hokuyo. Esta vez se repite todo el procedimiento anterior: añadir el código en el paquete encargado de generar el modelo de la simulación, implementar el modelo del sensor en el modelo del robot simulado y casar los parámetros de los sensores para la generación del mapa y para la percepción del laser.

Una vez modificados todos los parámetros, obtenemos un mapa perfecto del entorno simulado, el cual no habríamos podido tener sin el paso a paso de todas y cada una de las tareas aquí mencionadas.

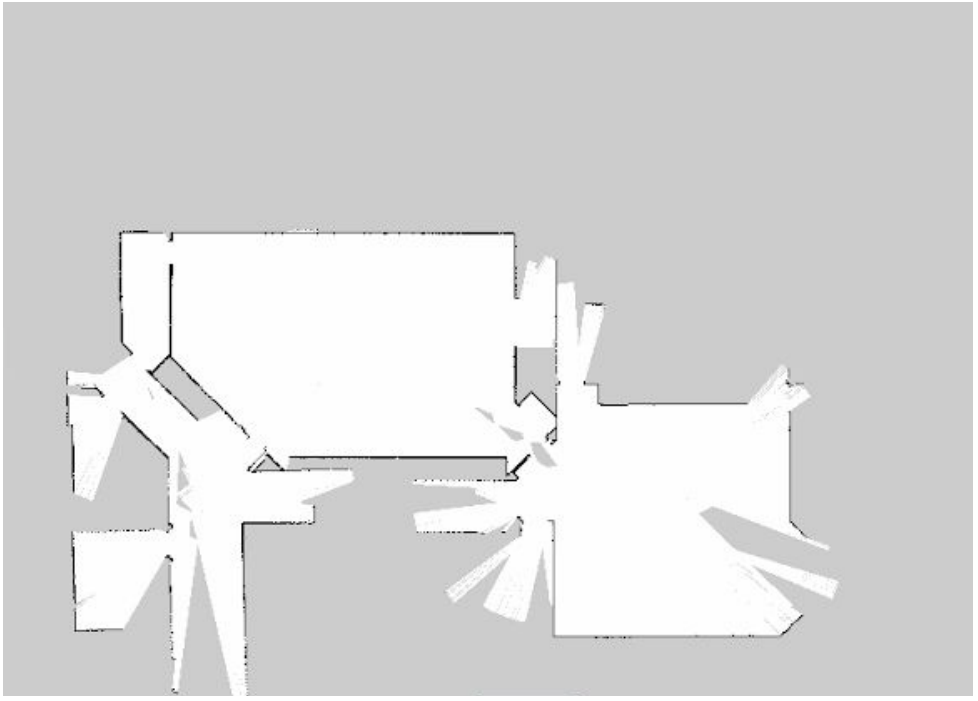


Ilustración 6: mapa generado por el sensor hokuyo en el programa RViz

7.4 RESULTADO FINAL

En los tres apartados anteriores se ha explicado detalladamente cada uno de los objetivos parciales que se han realizado a lo largo del proyecto. En este, se explica el resultado final desde un punto de vista general y más breve.

Como resultado de lo realizado durante el trabajo, disponemos de una simulación de robot capaz de ser controlado mediante el teclado del ordenador. Además esta simulación de robot tiene la capacidad de utilizar distintos sistemas de sensorización.

La configuración del control remoto le ofrece al usuario mover a gusto el modelo del robot en un concreto mapa cargado. Bien en la oficina de Valencia de los creadores, Robotnik, como en otro requerido.

También los sistemas de sensorización además de conseguir generar el mapa para que el usuario disponga de él, ofrecen la capacidad de tener imagen de lo que se encuentra delante de la simulación gracias a la cámara kinect.

Todo ello le brinda al modelo simulado del Summit gran versatilidad, pudiendo ser utilizado para muchas aplicaciones diferentes, como sería el control real, del que más adelante hablaremos.

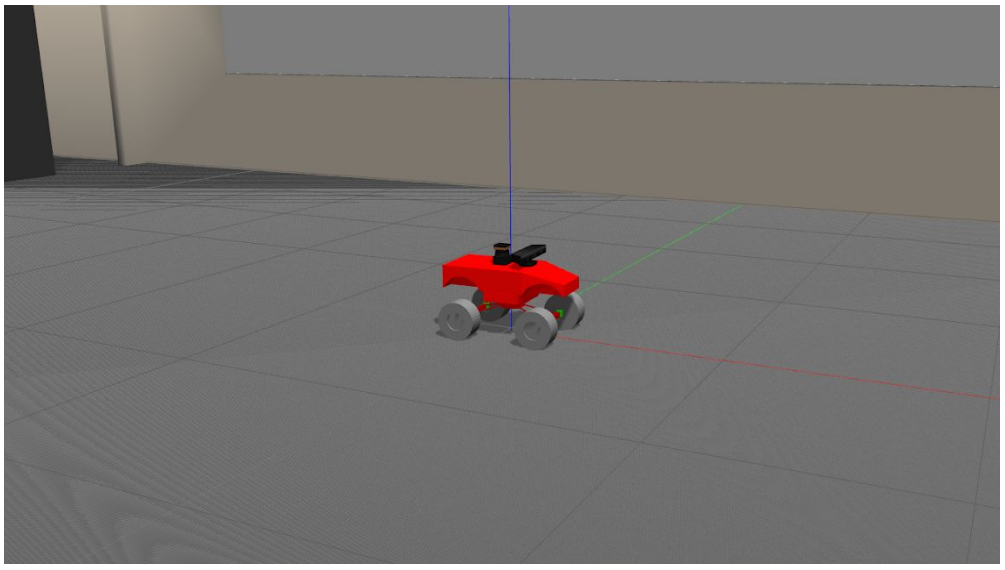


Ilustración 7: vista del robot simulado del programa Gazebo de ROS

8. PLAN DE PROYECTO Y PLANIFICACIÓN

En este apartado se detallan los paquetes de trabajo, las tareas e hitos del proyecto y los plazos, así como los participantes y la función de cada uno de ellos. Los plazos fijados al principio del trabajo no coinciden con los expuestos, pero en general el trabajo ha tenido una duración prácticamente igual o superior a la estimada, por lo tanto no ha supuesto un gran problema.

8.1. GRUPO DE TRABAJO

Esta es la lista de los integrantes del grupo de trabajo en este proyecto:

- Eloy Irigoyen: director del proyecto.
- Mikel Larrea: director del proyecto.

- Eneko Blázquez: ingeniero júnior, realizador del proyecto. Ha tenido una carga de trabajo de 4 horas semanales, en las que se incluyen las horas que ha estado con ambos directores del proyecto.

8.2. DEFINICIÓN DE LOS PAQUETES DE TRABAJO

Se ha dividido el trabajo a realizar a lo largo de todo el proyecto en paquetes de trabajo más pequeños. Cada uno de estos paquetes está subdividido en tareas que ir logrando a lo largo del trabajo. A continuación se definirán los paquetes, y más adelante se hará un apartado para cada paquete definiendo cada tarea e hito que forman dicho paquete.

- Estudio de las necesidades del proyecto y elaboración de las especificaciones técnicas: un primer paquete en el que se ha buscado información necesaria para el comienzo de la realización del trabajo.
- Objetivo parcial 1: en los objetivos parciales se ha realizado lo expuesto en apartados anteriores.
- Objetivo parcial 2
- Objetivo parcial 3
- Objetivo parcial 4
- Fase de gestión del proyecto: en este último paquete se ha concluido la documentación del trabajo, se ha redactado tanto la memoria como el manual de usuario. También se ha realizado una valoración de los resultados en comparación con el planteamiento que hubo al inicio.

Estudio de las necesidades del proyecto y elaboración de las especificaciones técnicas

1. Recopilación de la información de las distintas alternativas.
2. Realización de un estudio de las alternativas y elección de la utilizada en el proyecto.

3. Fijación de las especificación y objetivos que seguir.

Objetivo parcial 1

1. Instalación de la versión requerida de Ubuntu en el ordenador remoto.
2. Instalación de ROS en la versión kinect siguiendo los pasos de su página web en el ordenador remoto.
3. Familiarización con la herramienta ROS kinect en base a seguir los tutoriales de su página web.

Objetivo parcial 2

1. Puesta en marcha del código existente para la simulación de la plataforma Summit de la casa Robotnik.
2. Implementación de los cambios requeridos para la correcta ejecución.
3. Control remoto de la plataforma mediante teclado del ordenador.
4. Navegación manual en mapa simulado
5. Comprensión de los paquetes involucrados en la simulación del Summit.

Objetivo parcial 3

1. Modificación necesaria para integrar el sistema de sensorización.
2. Creación de los programas oportunos para la generación del mapa del entorno.
3. Creación de los programas oportunos para la correcta localización del modelo simulado en el entorno.
4. Comprensión de los paquetes involucrados en la simulación.

Fase de gestión del proyecto

1. Valoración los resultados obtenidos en función de lo esperado al inicio.
2. Redacción de la memoria final del proyecto, así como el manual de usuario.

8.3. PLAZOS

En este apartado se definirá la carga de trabajo de cada uno de los paquetes. Se ha decidido no dividirlo por las tareas, ya que sería complicado definir exactamente el orden y los momentos en los que se han realizado cada una sin pasar a la siguiente. Los plazos fijados al inicio del trabajo difieren de los que realmente se han logrado. En cualquier caso, y a pesar de las variaciones, se ha llegado al resultado final en el tiempo esperado, cumpliendo con todas las tareas.

Cabe recordar que el proyecto comenzó a realizarse a partir de Junio de 2018 hasta Julio de 2019, contando Agosto como mes de vacaciones a la hora de contar las horas de trabajo. Aunque el ingeniero junior no pudo tener acceso completo a las instalaciones puesto que se encontraba realizando un programa de estudios en el extranjero desde Septiembre de 2018 hasta Enero de 2019, la duración ha sido de 12 meses de trabajo a media jornada para el ingeniero júnior. Debido al tiempo pasado realizando el programa Erasmus, se han ajustado los plazos contando todo ese tiempo como 27 semanas.

En la tabla observamos la previsión de duración de los paquetes que se hizo antes del inicio del proyecto:

PAQUETE DE TRABAJO	TIEMPO (SEMANAS)

Estudio de las necesidades del proyecto y elaboración de las especificaciones técnicas	6
Objetivo parcial 1	5
Objetivo parcial 2	4
Objetivo parcial 3	4
Fase de gestión del proyecto	8

En esta segunda tabla, se definen las duraciones que han tenido finalmente cada paquete:

PAQUETE DE TRABAJO	**TIEMPO (SEMANAS)**
Estudio de las necesidades del proyecto y elaboración de las especificaciones técnicas	5
Objetivo parcial 1	4
Objetivo parcial 2	3
Objetivo parcial 3	7
Fase de gestión del proyecto	9

Las variaciones, como se puede ver, no han sido demasiado grande exceptuando el paquete relacionado con el objetivo parcial 3. Como se ha expuesto previamente, la realización de este objetivo tuvo una serie de problemas añadidos que no supimos valorar a la hora de iniciar el proyecto. En cuanto al último paquete, que se refiere principalmente a la redacción de la memoria, empezó a redactarse en paralelo desde

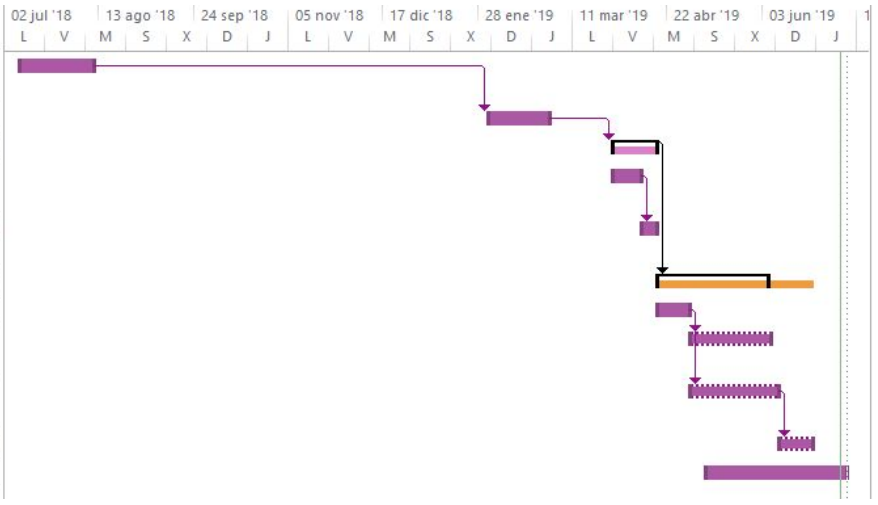
abril dando poco valor al principio dado que el resto de objetivos parciales consumían la mayoría del tiempo.

La documentación se ha escrito a lo largo de todo el tiempo de duración del proyecto. Dado el programa de estudios realizado en el extranjero, no ha sido posible trabajar en el laboratorio de ingeniería de sistemas y automática, de ahí el parón que tuvo el proyecto entre los meses de Septiembre de 2018 a Enero de 2019. No obstante, sin sonar repetitivo, los plazos de el proyecto se han realizado exactamente en el tiempo previsto.

9. DIAGRAMA DE GANTT

Aquí se muestra la distribución del trabajo a lo largo del tiempo que ha durado el proyecto.

Nombre de tarea	Duración	Comienzo	Fin
Estudio de las necesidades del proyecto y elaboración de las especificaciones tecnicas	5 sem.	lun 09/07/18	vie 10/08/18
Objetivo parcial 1	4 sem.	sáb 02/02/19	jue 28/02/19
Objetivo parcial 2	**2,8 sem.**	**vie 29/03/19**	**mié 17/04/19**
Estudio y analisis de los paquetes para la simulacion del robot	9 días	vie 29/03/19	mié 10/04/19
Implementación de los paquetes para el control remoto	5 días	jue 11/04/19	mié 17/04/19
Objetivo parcial 3	**7 sem.**	**jue 18/04/19**	**mié 05/06/19**
Instalación de la cámara kinect	2 sem.	jue 18/04/19	mié 01/05/19
Implementación de los paquetes para la optimización del láser	26 días	jue 02/05/19	jue 06/06/19
Analisis y estudio para la generación del mapa del entorno simulado	28 días	jue 02/05/19	lun 10/06/19
Instalación del sensor hokuyo	11 días	mar 11/06/19	mar 25/06/19
Fase de gestión del proyecto	9 sem.	jue 09/05/19	mié 10/07/19



10. RECURSOS BÁSICOS

En este capítulo del proyecto se listaran los distintos elementos y recursos utilizados para la realización final.

Se ha dividido este en dos apartados, uno para hardware y otro para software.

10.1. HARDWARE

- Un ordenador remoto.

10.2. SOFTWARE

- Sistema operativo Ubuntu versión 16.04, libre.
- ROS (Robot Operating System) versión Kinetic, libre.
- Programación en lenguajes C++ y/o Python, para los programas de ROS.

11. RESUMEN ECONÓMICO

En este apartado se van a detallar los gastos que se han asumido para este proyecto, relacionados tanto con los recursos materiales, como con los humanos. Los gastos previstos a inicios del proyecto coinciden con los expuestos.

11.1. CUADRO DE PRECIOS UNITARIOS

En este cuadro que se expondrá a continuación se van a resumir los precios unitarios de los recursos materiales, de los recursos humanos y de los servicios utilizados.

CONCEPTO	CANTIDAD	PRECIO	COSTE
RECURSOS MATERIALES			
Ordenador	1	350€	350€
Monitor	1	150€	150€
Teclado	1	39,95€	39,95€
Ratón	1	9,95€	9,95€
RECURSOS HUMANOS			
Director del proyecto (1)	62 horas	50€/hora	3.100€
Director del proyecto (2)	62 horas	50€/hora	3.100€
Ingeniero júnior	250 horas	40€/hora	10.000€
SERVICIOS			
Transporte en metro*	1 año	267,50€/año	267,50€
Conexión a internet**	1 año	0€/año	0€

*Solamente se incluye el transporte del ingeniero júnior, ya que los otros dos integrantes deben acudir a la universidad debido al resto de sus tareas, con lo cual su transporte no se incluye dentro de los gastos del proyecto.

**Se ha utilizado la conexión a internet proporcionada por la universidad, por lo tanto no ha habido coste alguno al respecto.

11.2. CUADRO DE MEDIDAS Y CÁLCULOS DEL PRESUPUESTO

Aquí se calcularán los costes totales, desglosados por los objetivos parciales definidos en los anteriores apartados.

CONCEPTO	CANTIDAD	PRECIO	COSTE
Estudio de las necesidades del proyecto y elaboración de las especificaciones técnicas			*4.717€*
Director del proyecto (1)	15 horas	50€/hora	750€
Director del proyecto (2)	15 horas	50€/hora	750€
Ingeniero júnior	60 horas	40€/hora	2.400€
Ordenador	1	350,00€	350,00€
Monitor	1	150,00€	150,00€
Teclado	1	39,95€	39,95€
Ratón	1	9,95€	9,95€
Transporte en metro	1 año	267,50€	267,50€
Conexión a internet	1 año	0€/año	0€/año
Objetivo parcial 1			*3.200€*
Director del proyecto (1)	12 horas	50€/hora	600€
Director del proyecto (2)	12 horas	50€/hora	600€
Ingeniero júnior	50 horas	40€/hora	2.000€
Objetivo parcial 2			*2.600€*
Director del proyecto (1)	10 horas	50€/hora	500€
Director del proyecto (2)	10 horas	50€/hora	500€
Ingeniero júnior	40 horas	40€/hora	1.600€
Objetivo parcial 3			*3.900€*

Director del proyecto (1)	15 horas	50€/hora	750€
Director del proyecto (2)	15 horas	50€/hora	750€
Ingeniero júnior	60 horas	40€/hora	2.400€
Fase de gestión del proyecto			*2.600€*
Director del proyecto (1)	10 horas	50€/hora	500€
Director del proyecto (2)	10 horas	50€/hora	500€
Ingeniero júnior	40 horas	40€/hora	1.600€
Suma			17.017€
Imprevistos (5%)			851€
Total sin IVA			17.868€
Impuestos (IVA, 21%)			3.752€
TOTAL			**21.621€**

11.3. CUADRO RESUMEN DEL PRESUPUESTO

Por último, en este cuadro se sintetizan los costes de los diferentes ámbitos, y se muestran los costes totales.

CONCEPTO	COSTE
Recursos materiales	550€
Recursos humanos	16.200€
Servicios	267,50€
Suma	17.017€
Imprevistos (5%)	851€
Total sin IVA	17.868€

Impuestos (IVA, 21%)	3.752€
TOTAL	**21.621€**

12. CONCLUSIONES

El trabajo realizado con el software ROS ha sido una experiencia muy enriquecedora. Ya que durante los años dedicados a la carrera como estudiante nunca he cursado una asignatura ni siquiera parecida a la programación con este software. En términos generales, ROS tiene su parte positiva y también su parte negativa, pues al ser una plataforma de código libre, diferentes personas pueden crearlo independientemente.

El aspecto positivo al que me refiero es su gran versatilidad. Permite crear paquetes personalizados para integrar infinitas aplicaciones que se deseen, de forma relativamente sencilla. Esta es una de las razones fundamentales por las que se decidió utilizar este software y no otro, y después del proyecto, se ha confirmado el potencial del mismo.

El aspecto negativo como he comentado en un aspecto más general es el fallo de la organización. Al utilizar códigos y archivos creados por distintas personas, los paquetes utilizados no siguen ningún tipo de pauta o criterio. Prácticamente hemos resumido los paquetes necesarios en el manual de usuario que se proporcionará junto a este documento, pero es un tanto complicado reducirlo más. La organización de los paquetes y la total comprensión de ellos sería un aspecto muy positivo que podría hacerse en un futuro.

Además, como aspecto negativo también se ha observado a lo largo del proyecto es la falta de información en relación a algunos aspectos concretos de nuestro robot. Como dijimos al principio del trabajo, este modelo Summit tiene un mecanismo de dirección

Ackermann, el cual dificulta un tanto las tareas a la hora de implementar el control remoto y otras tantas.

Indirectamente también resulta un tanto negativo el hecho de tener que estudiar el software ROS de manera independiente. Durante el curso no hay posibilidad de recibir clases, y el único método de aprendizaje son unas cuantas plataformas, así como los tutoriales ofrecidos en la propia página de ROS.

La desventaja de todos estos aspectos negativos es que nos ha llevado más tiempo del que creíamos necesario la realización de todo el trabajo. No obstante, y obviando la parte negativa del software, cabe mencionar que se ha aprendido muchísimo utilizando ROS. No solo en lo que al software se refiere, sino también en el ámbito de informática y a la hora de utilizar la plataforma Ubuntu. Es por esto que debemos quedarnos con esa idea positiva, y esperar que todo este trabajo sirva de ayuda para futuras áreas de investigación.

Por otra parte, habría que mencionar que el trabajo realizado podría ser de interés para la empresa Robotnik, la empresa fabricante del Summit. Es cierto que la empresa ha desarrollado otro tipo de robots más actualizados y además que no tienen el problema de la dirección Ackermann ya mencionada. No obstante, les seria de gran ayuda ofreciendo lo trabajado en este proyecto a gente que aun tenga una versión anticuada de la plataforma móvil terrestre Summit.

13. FUTURAS LÍNEAS A SEGUIR

A lo largo de la creación de este proyecto, han surgido nuevas ideas para seguir trabajando en ROS con el robot Summit. En concreto hay una idea que sería muy interesante de desarrollar pero que por falta de tiempo y recursos no ha sido posible la implementación de esta en el proyecto, se ha hablado en otros apartados de ella, y es la referida al control real de la plataforma Summit. Otra de las ideas es la implementación de otro tipo de controles remotos, el cual podría ser de mayor utilidad

y brindaría una libertad extra a la hora de navegar remotamente por un entorno. Y por último, una idea extra sería la implementación de los paquetes necesarios para la navegación autónoma del robot.

Este apartado se divide en tres puntos, cada uno de ellos para explicar una de las dos ideas mencionadas.

13.1. IMPLEMENTACIÓN DE CONTROLES REMOTOS

Desde el momento en que se integraron los paquetes para el control remoto mediante teclado del modelo simulado del Summit se tanteó la posibilidad de integrar otro tipo de controles. Finalmente se descartaron pues no eran necesarios para lograr el objetivo final y dado que era un proyecto largo de realizar con un apartado dedicado a líneas futuras, desde el primer momento se eliminó la idea de añadirlos.

La implementación de estos controles, como se ha explicado en el apartado de la definición del proyecto, requeriría modificar el código, pues nuestro robot contiene el mecanismo de dirección Ackermann, el cual necesita recibir un tipo de información distinta a la que envían la mayoría de controles remotos.

Este apartado podría dividirse en distintas partes, pues la cantidad de controles remotos que se pueden añadir es enorme. No obstante, no queriendo engordar demasiado el trabajo se sugieren dos tipos de mandos con los cuales se ha trabajado en anteriores versiones del robot Summit. Estos mandos son el mando propio del Summit y el mando de la videoconsola Wii, o Wiimote.

13.2. NAVEGACIÓN AUTÓNOMA

Este concepto es muy interesante a la hora de trabajar con robots. En nuestro proyecto se han seguido los pasos requeridos hasta este preciso punto. Pues desde un principio se dejó clara la idea de dejar en líneas futuras a seguir, este apartado.

Con la implementación de la navegación autónoma del modelo simulado, brindaremos a nuestro robot la capacidad de explorar cómo navegar por sí mismo el entorno en el que se encuentra.

El concepto de navegación significa proveer al robot del mapa simulado y generado a partir de los paquetes de navegación para que al mandarle una serie de directrices el robot llegará al punto deseado de la manera más rápida y eficaz posible.

Si se desarrolla un poco más la idea de navegación, se llega al concepto de exploración. Este concepto sería parecido al de navegación con la diferencia de que el mapa sería totalmente desconocido tanto para el robot como para el usuario. Esto ofrecería una amplia cantidad de beneficios, mencionados anteriormente. Sería un paso muy interesante que seguir para la persona que coja el proyecto a continuación.

13.3. NAVEGACIÓN REAL

La navegación real es un punto con el que se fantaseo a principios del proyecto. Como último objetivo que implementar en un futuro estaría la comunicación entre el PC y el Summit. Para ello se necesitaría aplicar unas actualización en el modelo real, puesto que se encuentra actualmente en la versión antigua tanto de la plataforma Ubuntu como del software ROS.

Además la capacidad de conectar el robot real al ordenador, brindará grandes beneficios, pues implementada la navegación autónoma, podríamos conseguir que el robot navega por sí solo por un área contaminada al cual las personas no puedes llegar, por ejemplo.

14. BIBLIOGRAFÍA

[1]- «ROS.org | Powering the world's robots». http://www.ros.org/.

[2]- «GitHub».. https://github.com/.

[3]- *Packages for the simulation of the Summit robot. Contribute to RobotnikAutomation/summit_sim development by creating an account on GitHub*. C++. 2014. Reprint, Robotnik Automation, 2015. https://github.com/RobotnikAutomation/summit_sim.

[4]- «kinetic/Installation/Ubuntu - ROS Wiki». http://wiki.ros.org/kinetic/Installation/Ubuntu.

[5]- «ROS/Tutorials - ROS Wiki». http://wiki.ros.org/ROS/Tutorials.

[6]- «ACKERMANN (Dirección) - Definición - Significado». https://diccionario.motorgiga.com/diccionario/ackermann-direccion-definicion-significado/gmx-niv15-con43.htm.

[7]- Kouros, George. *ROS keyboard and joystick teleoperation scripts for robots with ackermann steering: gkouros/ackermann-drive-teleop*. Python, 2019. https://github.com/gkouros/ackermann-drive-teleop.

[8]- *Converts a depth image to a laser scan for use with navigation and localization.: ros-perception/depthimage_to_laserscan*. C++. 2012. Reprint, ROS Perception, 2019. https://github.com/ros-perception/depthimage_to_laserscan.

[9]- «The Construct - YouTube». https://www.youtube.com/channel/UCt6Lag-vv25fTX3e11mVY1Q.

[10]- «ROS Development Studio». https://rds.theconstructsim.com/r/af35a9e2f869/.