UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN

Guiado de vehículo autónomo mediante tecnología LiDAR

Autor:

Dª Carlota Gómez Diego

Tutor:

Dr. D. Juan Carlos Aguado Manzano

D. Adrián Mazaira Hernández

VALLADOLID, JULIO 2022

TRABAJO FIN DE GRADO

TÍTULO:	**Guiado de vehículo autónomo mediante tecnología LiDAR**
AUTOR:	**Dª Carlota Gómez Diego**
TUTOR:	**Dr. D. Juan Carlos Aguado Manzano** **D. Adrián Mazaira Hernández**
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**Dr. D. Ignacio de Miguel Jiménez**
VOCAL:	**Dr. Dª. Noemí Merayo Álvarez**
SECRETARIO:	**Dr. D. Juan Carlos Aguado Manzano**
SUPLENTE:	**Dr. D. Ramón Durán Barroso**
SUPLENTE:	**Dr.Dª. Patricia Fernández Reguero**

FECHA:	**05/07/2022**
CALIFICACIÓN:	

Resumen

La investigación sobre el desarrollo de vehículos autónomos está evolucionando rápidamente y, con ello, las tecnologías involucradas. Los sistemas integrados en el vehículo son esenciales para percibir el entorno que rodea a la plataforma en movimiento, es decir, que el vehículo pueda percibir el entorno y actúe en consecuencia. En este Trabajo Fin de Grado se ha explorado las posibilidades que ofrece uno de estos sistemas, llamado *Light Detection and Ranging* (LiDAR). El principal propósito es integrar uno de estos dispositivos en un prototipo desarrollado en trabajos anteriores para obtener un guiado mediante el sensor LiDAR. Para ello, se ha estudiado las diferentes técnicas SLAM (*Simultaneous Localization And Mapping*) desarrolladas para hallar la localización y creación de mapa del entorno al mismo tiempo con la innovadora tecnología LiDAR. La técnica requerida proporciona una navegación fluida al vehículo.
El otro objetivo principal de este proyecto es modificar el diseño de una arquitectura eléctrica y electrónica del prototipo para su correcto funcionamiento, añadiendo recursos necesarios para cumplir las finalidades propuestas.

Abstract

Research into the development of autonomous vehicles is evolving fast and, with it, the technologies engaged. Integrated systems in vehicles are essential for sensing the environment surrounding the moving platform, it means the vehicle can perceive the environment and response accordingly. This Final Degree Project has explored the possibilities offered by one of these systems, called *Light Detection and Ranging* (LiDAR). The main purpose is to integrate one of these devices in a prototype developed in previous works in order to achieve the LiDAR sensor guided. For this purpose, the different SLAM (*Simultaneous Localization And Mapping*) techniques developed to find the location and create a map of the environment at the same time with the innovative LiDAR technology have been studied. The required technique provides a smooth navigation to the vehicle.
The other main objective of this project is to modify the design of an electrical and electronic architecture of the prototype for its correct operation, adding the necessary resources to fulfil the proposed purposes.

Palabras clave: Vehículo autónomo, LiDAR, posicionamiento *indoor*, ROS, SLAM.

Agradecimientos

En primer lugar, me gustaría agradecer este proyecto a mi tutor, Juan Carlos Aguado, por ofrecerme la oportunidad de trabajar en un ámbito totalmente desconocido para mí, por ayudarme a realizar este trabajo, dándome ánimos y echándome broncas cunando era necesario.

Por otro lado, quiero agradecer a mi cotutor, Adrián Mazaira, que ha estado ayudándome y enseñándome desde el primer día que empecé a trabajar en el proyecto, preguntando frecuentemente cómo iba evolucionando y dándome consejos.

Quiero dar las gracias también a mis compañeros y amigos por todas las horas pasadas en el laboratorio y por darnos ánimos mutuamente.

También agradecer a mi gran compañero de vida, Diego González, que me da fuerzas cuando más lo necesito y me aporta un gran apoyo desde el comienzo de la carrera.

Por último, a mi familia que me ha dado una buena educación desde pequeña, en especial, a mi abuela, mi tía, mi madre y mi hermana que son el pilar de mi vida, que me ayudan a superar todo tipo de situaciones. Gracias mamá por enseñarme a luchar y aportarme tu gran fortaleza.

Índice general

Índice de figuras

1 INTRODUCCIÓN

1.1 MOTIVACIÓN

Este Trabajo Fin de Grado es una continuación de la propuesta realizada por cuatro jóvenes estudiantes emprendedores para el concurso TwizyContest 2020 que llamaron TwizyLine. El proyecto planteaba crear un prototipo de vehículo autónomo de bajo coste que tuviera como aplicación concreta la creación de parkings automatizados con vehículos de conducción autónoma y conectado. El prototipo consistía en proporcionar una solución eficiente e innovadora siendo capaz de integrarlo en un vehículo eléctrico, que en este caso era el Renault Twizy por las condiciones de dicho concurso [1]. Las tareas estaban repartidas dentro del equipo, encargándose cada uno de una parte del proyecto. Los principales estudios e implementaciones llevados a cabo fueron los siguientes: un estudio de mercado sobre los parkings autónomos, una aplicación móvil, cuyo objetivo era permitir al usuario acceder al servicio; desarrollo de un servidor encargado del control del sistema y de cada aparcamiento autónomo; y diseño de un sistema para añadir capacidades de autonomía al coche [2] [3] [4] [5].

Figura 1. Logotipo y símbolo de TwizyLine.

Entre las razones que estos cuatro emprendedores adujeron para desarrollar de esta manera el vehículo era que la implementación de los vehículos autónomos tenía grandes dificultades no solo técnicas, sino también de aceptación por parte del público. Por lo tanto, en el concurso se planteó que demostrar un uso práctico de esta tecnología en un ambiente controlado como pueden ser los parkings automatizados era una buena manera de romper la barrera psicológica de los usuarios, a la vez que demostraron que se podía crear un modelo de negocio viable para dicha solución.

Por otro lado, las razones que llevan a la industria del automóvil a buscar este tipo de soluciones son varias. Primero, los vehículos han dejado de ser cuatro ruedas conectadas a un motor de combustión para convertirse en complejos sistemas que integran cada vez un mayor número de nuevas tecnologías, con un crecimiento que incluso podríamos calificar de exponencial. De hecho, la industria del automóvil ha sido históricamente de las más abiertas a integrar diferentes tecnologías emergentes.

Particularmente, una de las razones que más ha influido en el fuerte nivel de investigación, mejora e integración de nuevas tecnologías en los vehículos es el de los

accidentes de tráfico. La investigación de la industria de automoción relacionada con los accidentes de tráfico va desde desarrollar sistemas activos de control dentro de los vehículos para evitarlos hasta el desarrollo de sistemas pasivos y activos que minimicen los efectos de los mismos en caso de producirse. El continuo desarrollo e integración de medios, especialmente activos de control, lleva casi de manera natural a la idea del vehículo autónomo.

Aunque la estadística de accidentes ha mejorado mucho en las últimas décadas gracias a la integración de los sistemas de los que hemos hablado más arriba, a la mejora de las vías de circulación y a la concienciación de los conductores, que se han vuelto más prudentes, el número de accidentes que causan muertes y heridos graves sigue siendo bastante significativo. En España, en el año 2021 se produjeron 921 siniestros mortales, en los que fallecieron 1.004 personas y otras 3.728 resultaron heridas graves, lo que supone 97 fallecidos menos que el año 2019 (año de referencia), un 9% menos. Se trata del segundo mejor año de la serie histórica, solo por detrás de 2020, en el que fallecieron 870 personas, pero se trata de una estadística distorsionada debido a la pandemia [6]. Se puede ver gráficamente en la Figura 2 como el número de accidentes y de víctimas mortales se ha ido reduciendo a lo largo de la última década, gracias a los tres factores de los que hablábamos al principio de este párrafo: conductores más prudentes, mejora de las infraestructura y nuevas innovaciones tecnológicas.

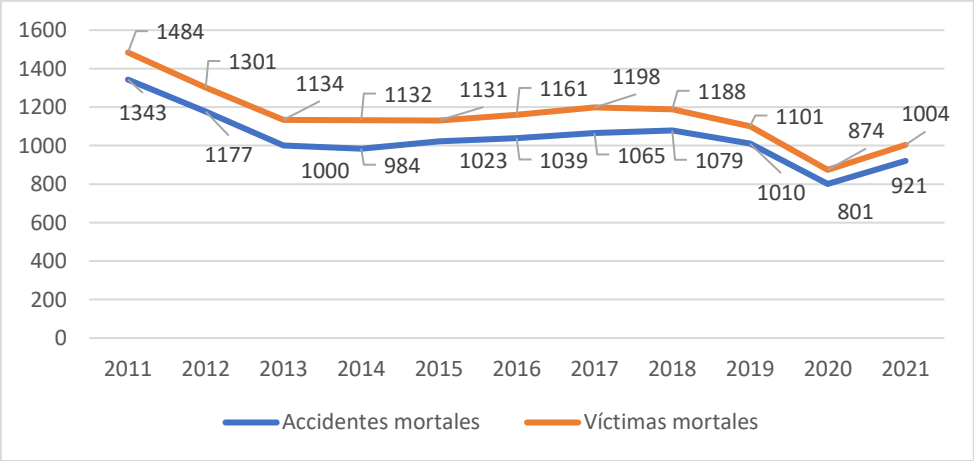

Figura 2. Siniestros mortales 2011-2021 [6].

A la vista de la evolución del número de accidentes de la que acabamos de hablar, está claro que merece la pena seguir invirtiendo en mejoras de la tecnología de los vehículos, pero la pregunta clave es en qué dirección.

Para determinar el sentido es importante saber cómo suceden dichos accidentes. En la mayoría de las ocasiones los accidentes en la carretera dependen del modo de conducción del conductor o de las condiciones meteorológicas adversas. La situación climática no se puede controlar pero, en cambio, sí que podemos conocer la meteorología con antelación y adecuar nuestro viaje y nuestro modo de conducción a esta. Asimismo, para reducir los riesgos de la conducción por carretera, es importante cumplir las normas viales y realizar una conducción segura. Una de las estrategias más sencillas para evitar los accidentes por

estos dos factores ya mencionados y que además conlleva otras ventajas es realizar lo que se denomina una conducción eficiente.

Una conducción eficiente consiste en una serie de reglas sencillas y eficaces con el objetivo de mejorar la seguridad vial y la comodidad del conductor, y reducir la contaminación acústica y atmosférica. Por lo tanto, una conducción eficiente ayudará a lograr un bajo consumo de energía, teniendo al mismo tiempo un mayor bienestar en la conducción lo que se ha demostrado que lleva a reducir los riesgos en la carretera [7]. A continuación, se enumeran una serie de ventajas de una conducción eficiente [8]:

- **Mejora el confort de conducción**: evitar frenazos y acelerones bruscos hace que se genere un estado de seguridad calmada en el conductor y al mismo tiempo evita el riesgo de accidentes.
- **Ahorro de energía**: el comportamiento del conductor afecta el consumo del vehículo. El vehículo automóvil consume un 15% de la energía total consumida en nuestro país [7]. Por tanto, teniendo una conducción eficiente obtendremos un menor consumo de energía a la vez que una mayor autonomía del vehículo.
- **Incremento de la seguridad vial**: cumpliendo una serie de pautas como puede ser mantener una buena distancia de seguridad para disponer de tiempo de reacción suficiente.

Para llegar a alcanzar esta conducción eficiente la industria automotriz ha desarrollado los sistemas avanzados de asistencia al conductor, ADAS (*Advanced Driver Assistance Systems)*. Hoy en día, las ayudas a la conducción dentro de un vehículo son esenciales, e incluso obligatorias para ciertas aplicaciones concretas, como puede ser el control de crucero adaptativo (*Adaptative Cruise Control*, ACC) o el asistente de mantenimiento del carril (*Lane Keep Assist*, LKA) [9]. Este tipo de ayudas son en realidad un paso previo para la introducción final del vehículo autónomo, que idealmente permitiría la conducción más eficiente posible y libre de accidentes.

Tanto en las ayudas a la conducción como en la investigación del vehículo autónomo requieren la inclusión de tecnología de última generación, tanto en el campo de la sensórica como en el campo de la computación. En el campo de la sensórica son necesarios dispositivos como, por ejemplo, cámaras, sensores radar, sensores de ultrasonidos o sensores LiDAR (*Light Detection and Ranging*). De entre este elenco de tecnologías, una de las más utilizadas y que parece más prometedora para su uso en el campo de la conducción autónoma es la tecnología LiDAR.

La tecnología LiDAR permite determinar la distancia desde un emisor láser a un objeto o superficie utilizando un haz láser pulsado, obteniendo una nube de puntos (2D o 3D) reflejados del escenario en el que se encuentre, con alta velocidad y precisión. Esta tecnología se emplea en diversas aplicaciones, algunas de las cuales se explicarán a continuación [10]:

- **Industria y energía**: determinación del estado de una planta industrial, detección de obstáculos y evitación de colisiones.
- **Ingeniería civil, infraestructuras y minería**: inspección periódica de diferentes infraestructuras para poder determinar el estado actual y las posibles variaciones existentes.
- **Forense**: ayuda a conocer con detalle y de forma precisa los escenarios de crímenes y accidentes de tráfico.
- **Arqueología y patrimonio**: para la representación tridimensional de forma rápida y precisa de elementos culturales.
- **Monitorización y estudios medioambientales**: análisis de densidad arbórea, medida de perfiles de nubes y aerosoles, ganadería profesional.

En la Figura 3 se puede ver un ejemplo de cómo un sensor LiDAR percibe el entorno de un vehículo, tratándose de una imagen compuesta por la medición de múltiples reflexiones captadas a partir de la emisión de pulsos láser, imagen que es denominada nube de puntos. Se puede observar que se trata de una tecnología de gran precisión, que permite observar en 3D el entorno cercano, y por lo tanto permite detectar obstáculos en la trayectoria del vehículo de manera relativamente sencilla.

Figura 3. Visión del LiDAR[1].

El proyecto TwizyLine, al que contribuye este Trabajo Fin de Grado, se basaba originalmente en sistemas de computación y sensores lo más baratos posibles. Así, por ejemplo, el guiado se realizaba construyendo un sigue-líneas basado en un sensor magnético, aunque posteriormente se añadió una cámara para seguir una línea pintada, y la computación necesaria se podía hacer con un simple miniordenador de tarjeta única. Esto está muy lejos de los tipos de sensores necesarios para vehículos realmente autónomos de los que hemos estado hablando en esta introducción, aunque esta elección

[1] Fuente de la imagen: https://japan.cnet.com/article/35130648/

del tipo de sensor es comprensible debido a la limitación presupuestaria de la que se partía. Además, se pudo comprobar que las soluciones propuestas no eran suficientemente sólidas, dado que el prototipo presentaba gran cantidad de limitaciones y de problemas técnicos a resolver.

Sin embargo, este primer prototipo permitió presentarse a diversos concursos que aportaron dinero suficiente como para plantear una mejora técnica del vehículo. Particularmente parecía especialmente atractiva la idea de incorporar tecnología LiDAR al vehículo con una capacidad de computación suficiente como para poder utilizar de manera eficiente dicha tecnología, dado que tal y como hemos dicho, parece ser una de las tecnologías más prometedoras dentro del desarrollo del vehículo autónomo. Por ello, en este Trabajo Fin de Grado nos centraremos en la aplicación de la tecnología LiDAR en el campo del vehículo autónomo y particularmente en su integración en el prototipo TwizyLine. La mayor ventaja que se espera obtener de incorporar un dispositivo LiDAR en el prototipo es mejorar la precisión en la percepción sobre el entorno del vehículo independientemente de la mayoría de variables ambientales, lo que se espera permita un mayor control del funcionamiento del prototipo en un ambiente controlado como es el de un parking cerrado.

1.2 OBJETIVOS

El principal objetivo de este proyecto es la integración de un LiDAR en un prototipo de vehículo autónomo tanto a nivel *hardware*, incluyéndolo en la estructura eléctrica del vehículo, como a nivel *software*, incorporando la información generada por el mismo al algoritmo de guiado del vehículo. En vista de este objetivo principal se definieron una serie de objetivos secundarios:

- Diseñar e implementar una nueva arquitectura eléctrica y electrónica del vehículo: El LiDAR requiere de una infraestructura eléctrica adecuada en el vehículo.
- Puesta en marcha del equipo LiDAR, configuración y lectura de sus datos en crudo o con las herramientas del fabricante: Uno de los primeros objetivos que se platearon fue desentrañar el funcionamiento del nuevo equipo utilizando las herramientas básicas del fabricante, para poder a posteriori integrarlo en el vehículo.
- Puesta en marcha del equipo *hardware* de tratamiento de los datos del LiDAR. El LiDAR es un dispositivo que genera una cantidad ingente de datos, y por lo tanto requiere de un sistema de tratamiento de estos datos muy diferente a los dispositivos que ya existían en el vehículo.
- Integración de *softwares* de gestión y procesamiento de los datos del LiDAR para su posterior utilización en el algoritmo de guiado del vehículo.
- Integración de otras tecnologías necesarias para el funcionamiento de la tecnología LiDAR.

1.3 ESTADOS Y MÉTODOS

Para realizar este proyecto se han tenido que seguir una serie de fases en cuanto a la integración del LiDAR y balizas de posicionamiento en interiores:

1. Primero, se hizo un **análisis sobre el desarrollo actual tecnológico** basado en el funcionamiento de los sensores LiDAR en un vehículo autónomo. Considerando esto, se planteó un estudio de las comunicaciones necesarias a utilizar en el vehículo.
2. Después, una vez se había finalizado el análisis, se hizo un **diseño de la reestructuración de la arquitectura eléctrica** del vehículo, basada en la que había anteriormente para conseguir una mayor eficiencia y seguridad. En este diseño también se tuvo en cuenta los requisitos de los dispositivos a utilizar en este proyecto, según se explica en el capítulo 3.
3. A continuación, se empezó con el **despliegue del *software*** que se va a procesar dentro del ordenador de fusión de datos del vehículo, en concreto, las técnicas utilizadas para la tecnología LiDAR.
4. Posteriormente, se realizó el **testeo del *software***, necesario para obtener un correcto funcionamiento en el vehículo.
5. Finalmente, se realizó una **implementación de la integración** de la arquitectura eléctrica diseñada anteriormente.

1.4 RECURSOS

Para llevar a cabo este proyecto, los materiales utilizados se describirán a continuación, los cuales han sido escogidos específicamente con este propósito:

- 1 ZOTAC ZBOX E Series Magnus es el ordenador donde se procesará todos los datos sobre los sensores, se ha denominado PC Fusión. Sus propiedades son las siguientes: 1 procesador Intel Core I7-10750H de 6 núcleos, 1 gráfica NVIDIA GeForce RTX 2080 SUPER, 1 disco duro 256GB SSD, 1 RAM DDR4 16GB, 6 puertos USB 3.0, 1 puerto USB tipo C, 2 puertos HDMI y 2 puertos Ethernet [11].
- 1 LiDAR RS-LIDAR-16, como su propio nombre indica tiene 16 haces láser, que ayudará a la determinación de la posición del vehículo [12].
- 1 *Starter Set HW v4.9, Indoor "GPS" Marvelmind ±2cm precision* para dar información redundante sobre la posición y navegación del vehículo en un entorno cerrado [13].

Por otro lado, también se ha sido utilizado material para la reestructuración de la arquitectura de protocolos de comunicación y la de alimentación, dichos recursos han sido los siguientes:

- 1 Hub con 5 canales CAN, 5 puertos DB9 hembra y 1 DB9 macho en cada respectivo canal. Este concentrador se ha denominado Spybox CAN.

- 1 caja de fusibles constituida por 6 vías con indicador LED para la protección del sistema eléctrico del coche [14].

1.5 ESTRUCTURA DE LA MEMORIA

Este documento fundamentalmente explica la integración de un dispositivo LiDAR en un prototipo de vehículo autónomo llamado Twizyline, proyecto comentado en la sección 1.1. Este trabajo está dividido en cinco capítulos, siendo el actual el primero. La estructura del resto del documento es la siguiente:

- **Capítulo 2: Estado del arte sobre la integración de la tecnología LiDAR en vehículo autónomo**. En este capítulo se presenta una definición sobre el vehículo autónomo, explicando los diferentes conceptos de dicho término. A continuación, se desarrollan diferentes puntos sobre la tecnología LiDAR: historia, funcionamiento y tipos. Para finalizar este apartado, se introducen las diferentes técnicas que se puede utilizar para hallar la localización del vehículo mediante la comunicación ROS.
- **Capítulo 3: Arquitectura eléctrica**. En esta sección se define la arquitectura eléctrica y electrónica del vehículo que está dividida en dos partes, por un lado, se presentan las comunicaciones que implementa el vehículo, por otro lado, se explica la estructura de la alimentación del vehículo.
- **Capítulo 4: Tecnologías utilizadas**. En este capítulo se muestra principalmente las tecnologías con las que se ha trabajado en este proyecto, detallando el funcionamiento de cada una de ellas.
- **Capítulo 5: *Software* desplegado.** En este apartado se detalla el *software* desplegado, así como lo inconvenientes previstos y hasta donde se ha conseguido llegar con la gestión y el procesamiento de los datos de las diferentes tecnologías utilizadas.
- **Capítulo 6: Conclusiones y líneas futuras**. Finalmente, se presenta una revisión del proyecto, además de una explicación sobre los resultados obtenidos, incidiendo tanto en aspectos positivos como negativos. En función de esta explicación se plantean diversas posibilidades de líneas de investigación futuras que podrían continuar con este trabajo**.**

2 ESTADO DEL ARTE

En este capítulo se describe una visión general sobre el vehículo autónomo, entrando en detalle sobre los principales conceptos y las tecnologías utilizadas para percibir el entorno. La mayoría de estas tecnologías ya están siendo utilizadas en el mercado gracias a las ayudas a la conducción que incluyen los vehículos hoy en día; más adelante se definen algunas de estas ayudas. Una de las tecnologías que se está integrando en vehículos autónomos para percibir el entorno es el dispositivo LiDAR, tecnología sobre la que se desarrolla este proyecto y que se explicará en profundidad en las siguientes secciones. El objetivo de este proyecto es conseguir guiar un vehículo autónomo de nivel 4 en un entorno de parking cerrado, para ello, se analiza tanto el sistema de comunicación para obtener los datos del LiDAR como las diferentes técnicas de SLAM para obtener una trayectoria desde que ha sido iniciada la ruta.

2.1 VEHÍCULO AUTÓNOMO

Un vehículo autónomo es un tipo de automóvil con variedad de tecnologías que conduce sin necesidad de intervención de un actor humano, siendo capaz de percibir el medio que le rodea [15]. Para empezar a hablar del vehículo autónomo hay que conocer los diferentes niveles conducción que existen y las cuestiones que debe resolver el vehículo en cada uno de ellos. Estos conceptos están recogidos en el estándar J3016 por el SAE (*Society of Automotive Engineers*), organización enfocada en la ingeniería aeroespacial, automoción y las industrias comerciales especializadas en la construcción del vehículo. El estándar define una taxonomía lógica para las funcionalidades de la clasificación de la conducción autónoma y, por otra parte, estandariza los conceptos, términos y usos relacionados con el fin de facilitar una comunicación clara [16].

Para empezar a diseñar un vehículo autónomo, hay que tener en cuenta tres principales finalidades tal y como se define en el documento de General Motors [17]: **percepción**, su tarea es incluir todas las tecnologías necesarias para que el vehículo perciba cualquier tipo de evento que ocurra en su entorno o para detectar cualquier tipo de objeto cerca de él; **planificación,** se encarga de procesar la información que genera la percepción para llegar a calcular la ruta óptima acorde con el destino, el potencial camino y los obstáculos presentes en este; y **control**, que es el encargado de mover el vehículo tanto longitudinal como transversalmente.

Uno de los conceptos principales de la conducción autónoma a tener en cuenta es la tarea de conducción dinámica (*Dynamic driving task*, DDT), que consiste en todas las tareas realizadas para conducir un vehículo. El DDT realiza dos de las tres tareas anteriormente descritas como fundamentales para diseñar un vehículo autónomo, tal y como se puede observar gráficamente en la Figura 4 [16]:

- **Control** del vehículo tanto transversal como longitudinalmente en tiempo real, en este caso, se tiene en cuenta las tareas básicas de la conducción.

- Una de las tareas de la **percepción** es llevar a cabo el OEDR (*Object and event detection and response*) que se encarga de la detección de obstáculos y eventos que se encuentren en el entorno y ejecutar una respuesta apropiada.

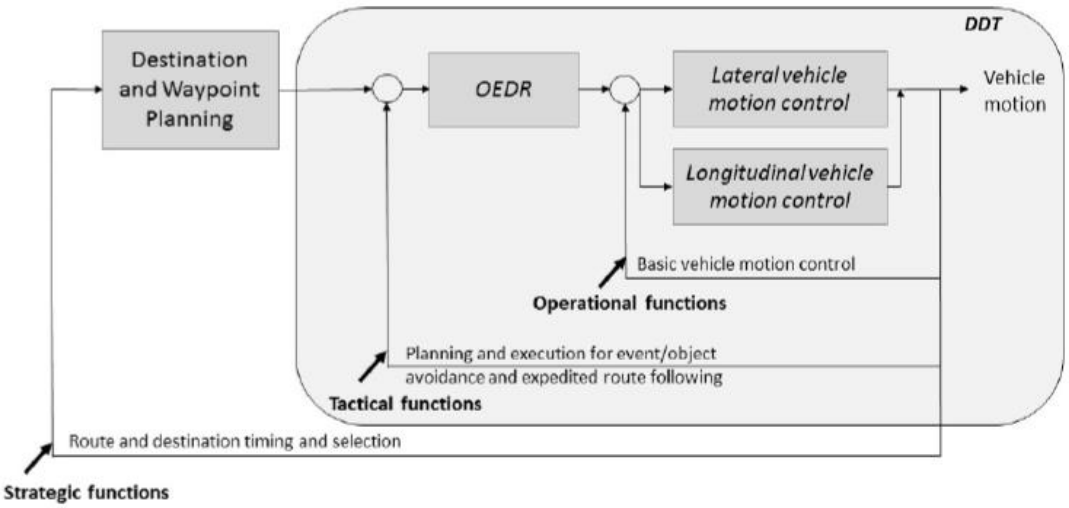

Figura 4. Vista esquemática de la tarea a la conducción (DDT) [16].

La primera finalidad fundamental para diseñar un vehículo autónomo (según General Motors [17]), y que también atañe al DDT, es la percepción. Esta se ha definido como la encargada de la detección de objetos y eventos, además de establecer la localización del vehículo. Esto requiere de una recopilación de datos de los sensores del vehículo y su respectivo procesamiento. La percepción comprende el entorno del vehículo gracias a los datos suministrados por los sensores. A continuación, se detallarán cuatro de las tecnologías más comúnmente usadas para la detección de obstáculos y eventos:

- **SONAR, sensores ultrasónicos**: por medio de la emisión y detección de ultrasonidos es capaz de medir la distancia del sensor al objeto. Son muy útiles para detectar objetos próximos, pero no para objetos lejanos. Estos dispositivos se pueden clasificar en función del alcance de detección y del campo de visión. La gran ventaja de estos sensores es que ofrecen buenos resultados incluso con mal tiempo y su funcionamiento no se ve afectado por la luz. Estos sensores se integran en el vehículo principalmente para funcionalidades de ayuda al aparcamiento.
- **RADAR (*Radio Detection and Ranging sensor*)**: emplea un método de detección de objetos por ondas de radio. Su principal uso es detectar obstáculos móviles y obtener su distancia y velocidad, aunque también funciona para obstáculos fijos. Este tipo de sensor no es el mejor candidato para identificar y clasificar obstáculos, pero sí destaca por su independencia de las condiciones meteorológicas o de luz. Otra de las ventajas de este sensor es su reducido coste en comparación con el resto de sistemas.
- **Cámara**: su objetivo es percibir el entorno junto con *software* de inteligencia artificial capaz de procesar las imágenes, esto hace posible reconocer y clasificar objetos, siendo esto la principal virtud de este sensor. La gran desventaja es que se requiere de alta carga computacional para ser capaz de procesar las imágenes

en tiempo real, además de que está tecnología no es fiable ni en condiciones meteorológicas adversas ni cuando se tiene una baja visibilidad, es decir, cuando es de noche.

- **LiDAR (*Light Detection and Ranging*)**: permite estimar la geometría de una escena en 3D a través de una luz láser y midiendo el reflejo con un sensor, teniendo en cuenta el retardo y la intensidad. Suelen incluir elementos giratorios con múltiples fuentes de luz para poder recoger más información. Los LIDAR son más precisos que las cámaras en la estimación de la profundidad, ya que no pueden ser engañados por las sombras, la luz del sol o los faros de otros vehículos. Estos dispositivos también requieren de menos computación que la cámara. Los LIDAR dan información absoluta e inmediata sobre la distancia a un punto, mientras que los sistemas basados en cámaras necesitan procesar e interpretar las imágenes antes de obtener un resultado fiable. Sin embargo, al igual que las cámaras, los LIDAR presentan dificultades en condiciones meteorológicas adversas, como la nieve o la lluvia. A diferencia de la tecnología RADAR, los LiDAR son más precisos y flexibles. En cualquier caso, la gran desventaja de los LiDAR frente a la cámara y el RADAR es su precio.

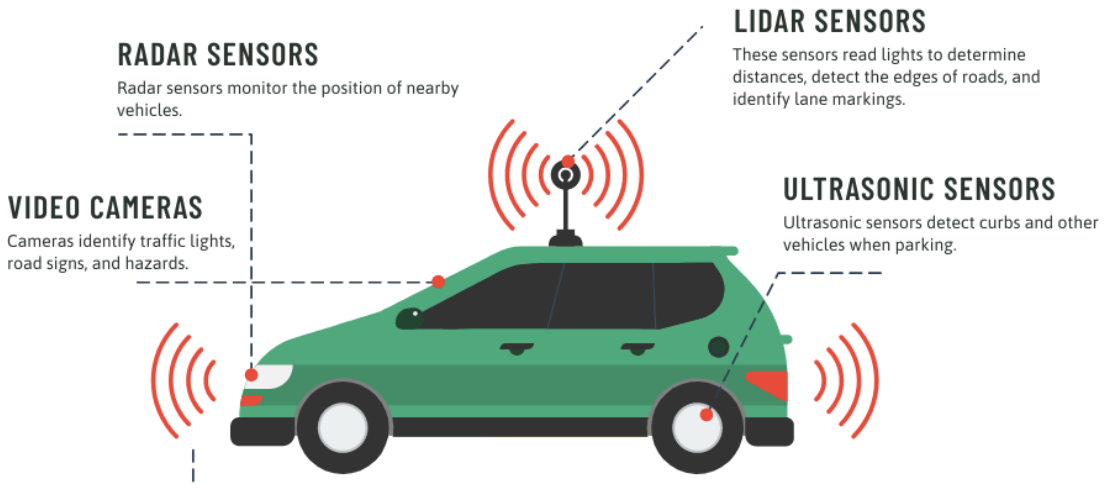

Figura 5. Clasificación de las diferentes tecnologías.

En la Figura 5 se representan las diferentes tecnologías descritas y sus usos más comunes dentro de un vehículo. Estos sensores tienen variedad de ventajas y desventajas las cuales afectan a la integración del vehículo autónomo, tal y como se ha descrito anteriormente. La principal desventaja común es su vulnerabilidad frente a los efectos meteorológicos.

El LiDAR es una tecnología de longitud de onda corta que consigue de forma precisa identificar objetos pequeños y proporcionar una imagen en 3D, sin embargo, se ve afectado gravemente por las gotas de lluvia ya que, si se encuentra cerca del emisor láser, existe una considerable posibilidad de detección errónea. Por otro lado, el sensor RADAR puede trabajar en condiciones meteorológicas variadas gracias a un amplio espectro de longitud de onda, pero este espectro puede ser afectado por la lluvia debido a que este fenómeno meteorológico puede atenuar la señal que emite el sensor, obteniendo una señal

débil y poco fiable. Los sensores de ultrasonidos y las cámaras son sistemas que les afectan también las condiciones meteorológicas ya que, si se encuentran cubiertos de nieve, hielo o suciedad su actuación no es óptima, es decir, el funcionamiento no es correcto [18]. En la Figura 6, se representa el espectro electromagnético utilizado en cada tipo de sensor, cabe destacar que cada tecnología utiliza un espectro distinto.

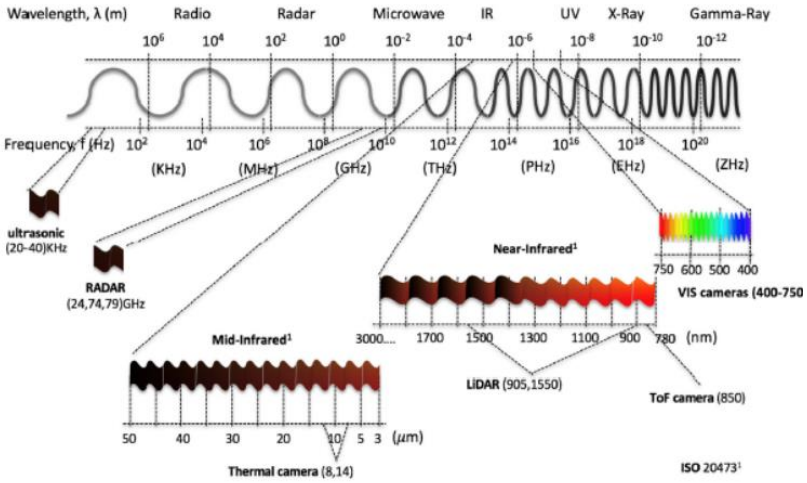

Figura 6. Espectro electromagnético utilizado en los sensores AV [18].

En la Figura 7, se resumen algunas de las principales características de los diferentes sensores utilizados en los vehículos autónomos.

Feature	LiDAR	RADAR	Camera	Ultrasonic
Primary Technology	Laser beam	Radio wave	Light	Sound wave
Range	~200 m	~250 m	~200 m	~5 m
Resolution	Good	Average	Very good	Poor
Affected by weather conditions	Yes	Yes	Yes	Yes
Affected by lighting conditions	No	No	Yes	No
Detects speed	Good	Very good	Poor	Poor
Detects distance	Good	Very good	Poor	Good
Interference susceptibility	Good	Poor	Very Good	Good
Size	Bulky	Small	Small	Small

Figura 7. Principales características de los sensores AV [18].

En general no se suele incluir un único tipo de tecnología en un vehículo autónomo, sino que se suele optar por integrar varios tipos de tecnologías. Integrando y fusionando diferentes tecnologías es posible cubrir las debilidades de unas gracias a otras, permitiendo así diseñar un sistema robusto.

Estas tecnologías son cada vez más utilizadas en vehículos disponibles en el mercado. Los coches que se fabrican hoy en día cuentan en su mayoría con un nivel de conducción autónoma de nivel 2, automatización parcial, y logran tal nivel gracias a los sensores incorporados en el vehículo. En la Figura 8, se muestra gráficamente un resumen sobre los diversos niveles de conducción autónoma. Cabe destacar que en los niveles 4 y 5, es prescindible la figura del conductor, es decir, se podría eliminar los elementos de control y manejo del vehículo ya que no es necesario que ninguna persona esté pendiente del DDT.

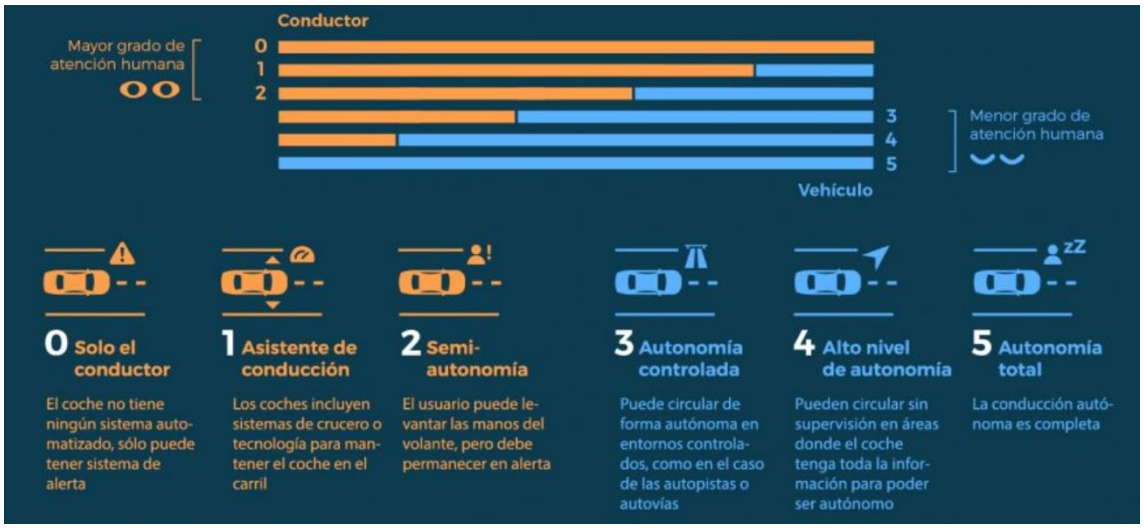

Figura 8. Niveles de conducción autónoma [19].

Como ya se ha mencionado anteriormente, los nuevos vehículos fabricados suelen ser de nivel dos de autonomía, aunque pueden tanto de nivel inferior como superior [20], dependiendo de las ayudas a la conducción que tengan integradas los vehículos. Estas ayudas, denominadas sistemas avanzados de asistencia al conductor (*Advanced Driver Assistance Systems*, ADAS) acercan a los vehículos convencionales a un vehículo autónomo, y son la demostración de la evolución de la tecnología en los vehículos y de la aplicación de la tecnología del vehículo autónomo a la calle. Algunos de estos sistemas ya se han mencionado en el apartado 1.1 y a continuación, se definen algunos de los principales sistemas [15]:

- **Sistema de aviso de colisión frontal** (*Forward Collision Warning, FCW)*: se avisa al conductor cuando se encuentra en una situación de proximidad con otro vehículo con una alta probabilidad de choque, dependiendo de la marca del vehículo los avisos varían entre el uso de las luces de intermitencia o señales hápticas [15].

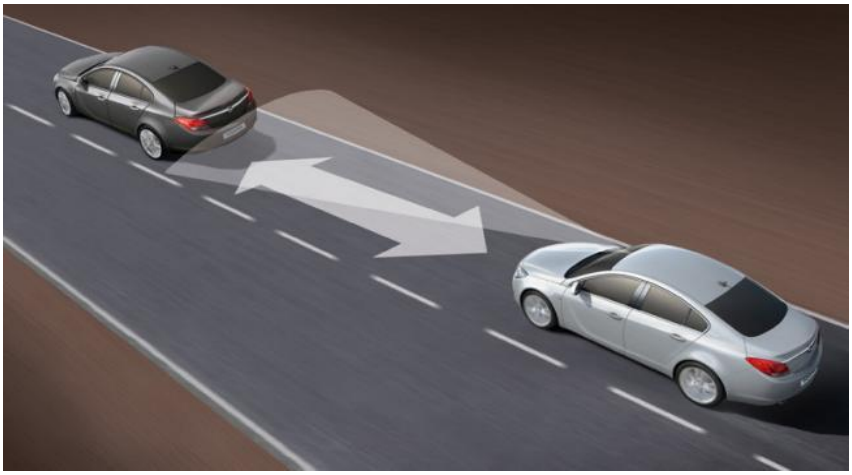

Figura 9. Sistema de aviso de colisión frontal [21].

- **Sistema de asistente de mantenimiento de carril** (*Lane Departure Warning*, LDW): se trata de avisar cuando el vehículo está fuera de su carril y no tiene la

intermitencia dada, es decir, avisa si cruza una línea blanca sin indicación, por lo que es especialmente importante en caso de somnolencia repentina durante la conducción [22].

Figura 10. Sistema de aviso de cambio de carril [22].

- **Control de crucero de adaptativo** (*Adaptative Cruise Control*, ACC): ajusta automáticamente la velocidad dependiendo del vehículo que encuentre delante para mantener una distancia de seguridad [23].

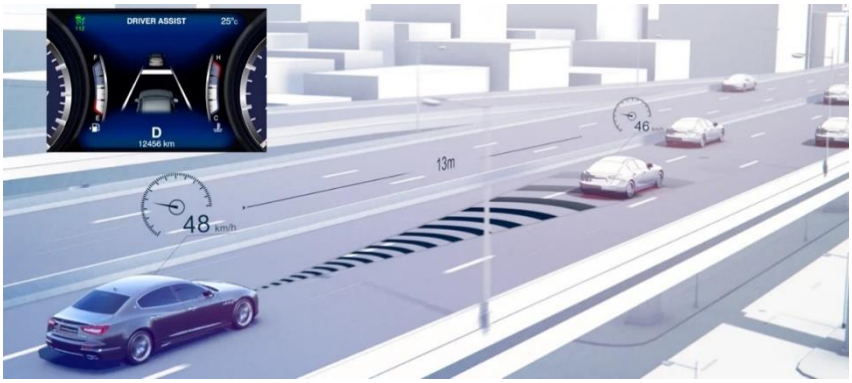

Figura 11. Control de crucero adaptativo [23].

- **Frenado de emergencia automático** (*Automatic Emergency Brake*, AEB): este sistema se ha diseñado para evitar una colisión sin necesidad de que el conductor intervenga, por tanto, es encargado de anticiparse a dicha colisión y efectuar la frenada [24].

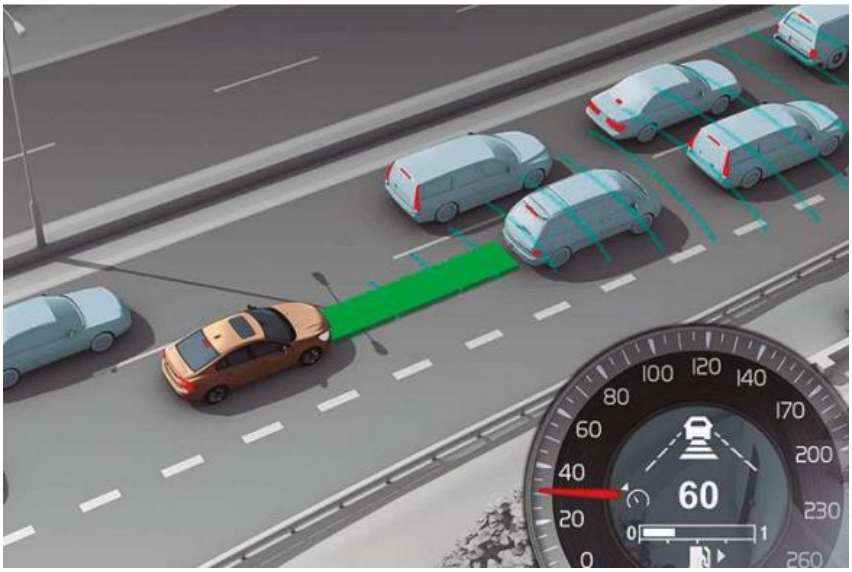

Figura 12. Frenado de emergencia autónomo [24].

- **Aparcamiento asistido** (*Parking Assistant*): este sistema se ha diseñado para facilitar el aparcamiento en zonas urbanas mediante la integración de herramientas tecnológicas en el vehículo.

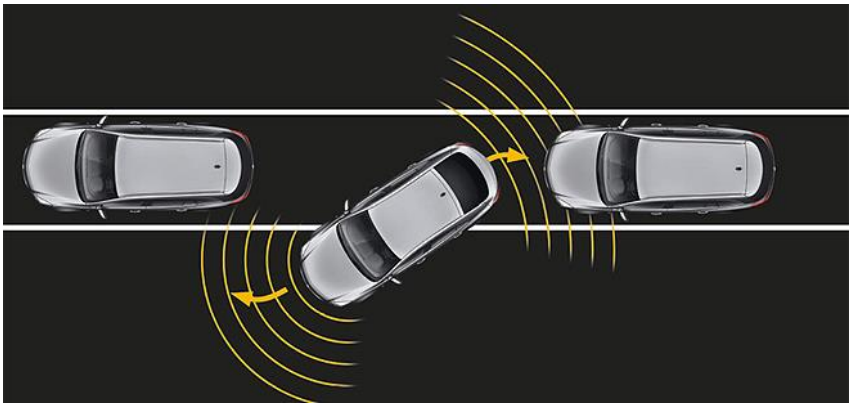
Figura 13. Aparcamiento asistido [25].

Todas estas ayudas a la conducción son un gran avance para la investigación del vehículo autónomo. Por este motivo, cada vez se añaden más herramientas tecnológicas al vehículo con el fin de percibir el contexto alrededor de la conducción. La última tecnología añadida en los sistemas ADAS es el dispositivo LiDAR, aunque estos se utilizan menos por su elevado coste. La tecnología LiDAR, en la que está basada este proyecto, es una de las tecnologías fundamentales en el sector de la conducción autónoma como se ha visto anteriormente. El LiDAR es utilizado en vehículos autónomos para detectar y evitar obstáculos, ya que gracias a la utilidad de los rayos láser facilita la navegación a través de una ruta a la vez que es posible la detección de obstáculos y eventos con una carga computacional mucho menor que la de una cámara.

2.2 LiDAR

Está sección describe en profundidad la tecnología LiDAR con el objetivo de entender su funcionamiento, lo cual es vital para este proyecto. Está estructurado en tres partes distintas: breve descripción de la historia de esta tecnología, explicación del funcionamiento, definición de los diferentes tipos de LiDAR que existen y, para terminar, especificación de las propiedades a tener en cuenta a la hora de elegir un sensor LiDAR para un vehículo autónomo.

2.2.1 Evolución del LiDAR

La medición de distancias mediante el uso de un haz de luz tiene ya un largo recorrido y ha ido evolucionando a lo largo de los años. Este tipo de mecanismo fue utilizado por primera vez en la década de 1930 para medir los perfiles de densidad del aire en la atmósfera por medio de la determinación de la intensidad de dispersión de los haces de luz. Sin embargo, no fue hasta pasada la invención del láser cuando también se desarrolló el LiDAR en la década de 1960. El primer sistema LiDAR fue diseñado por Malcolm Sticht en 1961 para ser utilizado en los aviones, en concreto en una empresa aeroespacial americana llamada *Hughes Aircraft Company*. Los primeros modelos se llamaban COLIDAR como acrónimo de *Coherent Light Detection and Ranging* - Detección y Alcance de Luz Coherente. El COLIDAR fue utilizado por primera vez en tierra en el año 1963 y se denominó COLIDAR Mark II, su principal aplicación era en el ámbito militar. Concretamente estos sistemas LiDAR se utilizaron para realizar el seguimiento de aviones. Esto se lograba gracias a la combinación de las capacidades que tenían de imagen láser y medición de distancias [26].

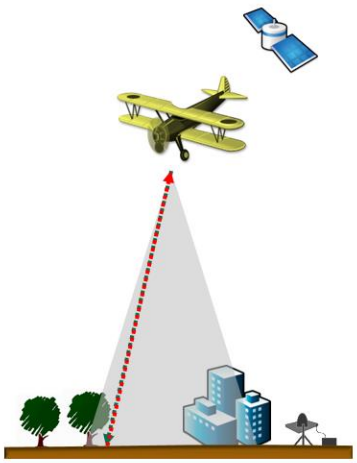

Figura 14. Funcionamiento LiDAR en aeronave [27].

En los años de la década los 70, la NASA comenzó a realizar teledetección usando láseres que se convertirían en el prototipo para el posterior despliegue de los sensores LiDAR [28]. En 1971, se utilizó por primera vez para crear mapas de la Luna durante la misión de Apollo 15. Estos mapas se utilizaban sobre todo en tareas de navegación. Desde entonces los usos del LiDAR se han multiplicado.

Aunque en los primeros tiempos los sensores LiDAR ayudaban a la navegación, no eran lo suficientemente precisos en la fotogrametría, esto es, no eran capaces de resolver la forma, dimensiones y posición de los objetos en un espacio utilizando las imágenes obtenidas. No obstante, la mejora de la localización y navegación usando LiDAR llegó cuando hubo disponibilidad comercial tanto del sistema global de posicionamiento (*Global Positioning System*, GPS) como de la unidad de medición inercial electrónica (*Inertial Measurement Unit*, IMU) a finales de la década de los 80 [28]. La posibilidad de realizar fusión de señales y aplicar técnicas como los filtros de Kalman permitió mejorar enormemente la fiabilidad de los resultados obtenidos. De hecho, veremos a lo largo de este Trabajo Fin de Grado que, aunque hablamos esencialmente del LiDAR como sistema de guiado, la mayoría de los paquetes informáticos que se pueden utilizar para la navegación incorporan la entrada de señales de otros sensores para mejorar la navegación del sistema.

2.2.2 FUNCIONAMIENTO LIDAR

La configuración básica de un sistema LiDAR se muestra en la Figura 15. Básicamente un LiDAR consta de un transmisor y un receptor. El LiDAR genera pulsos de luz cortos, del orden de nanosegundos con unas propiedades espectrales específicas que dependen del láser usado. El principio de funcionamiento que sigue es muy similar al del RADAR. El receptor recoge los fotones reflejados en distintas superficies. Para mejorar la calidad de la señal recibida se aplican diferentes tipos de filtros y análisis ópticos, desde filtrados por longitud de onda, por polarización o incluso procesamiento de la señal con métodos matemáticos. Esta información es utilizada para obtener principalmente dos variables. Por un lado, el tiempo desde el envío del pulso hasta su vuelta en forma de pulso reflejado. Por otro lado, la intensidad de reflexión. La primera variable permite el cálculo de las dimensiones, distancia, localización, etc., del objeto. La segunda variable puede ser utilizada para determinar el tipo de material del que se trata [26] [28].

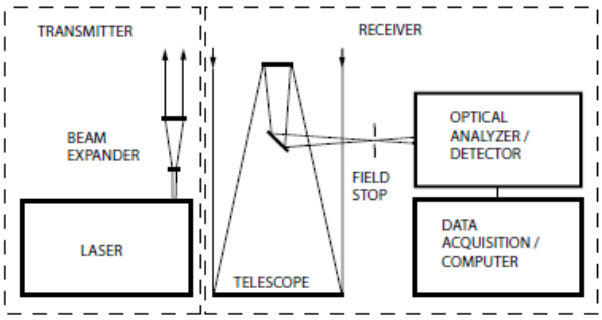

Figura 15. Bloques principales y funcionamiento básico de un sistema LiDAR [26].

El resultado final que se obtiene en la mayoría de aplicaciones, y particularmente en el sector de vehículo autónomo, es un mapeado del entorno del vehículo creando una nube de puntos, como se muestra en la Figura 16. Este mapeado se puede utilizar junto con mapas previos para localizar de forma exacta el vehículo, pero sobre todo se puede utilizar para detectar obstáculos en la ruta del vehículo o bien objetos móviles y su trayectoria y su probabilidad de colisión dentro de la ruta que tiene marcado el vehículo autónomo.

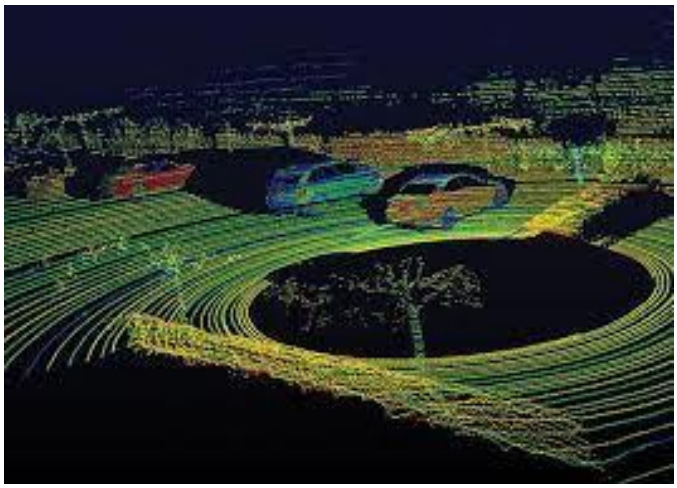

Figura 16. Visión sensor LiDAR.

2.2.3 TIPOS DE LIDAR

Lo sistemas LiDAR pueden clasificarse en función de diferentes criterios, tal y como se puede ver en la Figura 17. Si atendemos al ámbito de transporte en que se utiliza el LiDAR podemos distinguir entre sistemas aerotransportados y terrestres. Esta clasificación no es una mera separación por medio de transporte, sino que cada tipo de medio requiere un LiDAR con características adaptadas a dicho medio. Además, dentro de cada uno de ellos se distingue:

- **Aerotransportados**: el sistema es integrado es un avión o helicóptero. Su objetivo consiste en emitir el haz de láser hacia el suelo normalmente con un rango de 180 grados, con lo que su función es calcular la distancia hacia la tierra. Este tipo se divide en dos modelos de sensores:
    - **Topográficos**: son usados para examinar la superficie, los cuales pueden ser utilizados para las ciencias forestales, planos urbanos, etc.
    - **Batimétricos**: empleados para obtener la información de profundidad del agua y la elevación a la que se encuentra el transporte.
- **Terrestre**: el sistema puede ser integrado tanto en un vehículo en movimiento como en un dispositivo estacionario. Su objetivo consiste en recoger puntos con una gran precisión para ser capaz de identificar objetos, el haz de láser en este caso puede llegar a tener un rango de 360 grados en horizontal y, dependiendo de la aplicación concreta, desde una sola capa (LiDAR 2D) hasta múltiples capas (LiDAR 3D) hasta un máximo de unas 128. Este tipo también se divide en dos modelos:
    - **Móvil**: consiste en la obtención del conjunto de nubes de puntos desde una plataforma en movimiento. Este tipo de sistemas pueden estar integrados en diferentes transportes como: coches, trenes e incluso barcos.
    - **Estático**: consiste en la obtención del conjunto de nubes de puntos desde una ubicación estática, normalmente se encuentra elevado en un trípode. Este tipo de LiDAR es utilizado en aplicación como son la ingeniería, topografía, arqueología y minería.

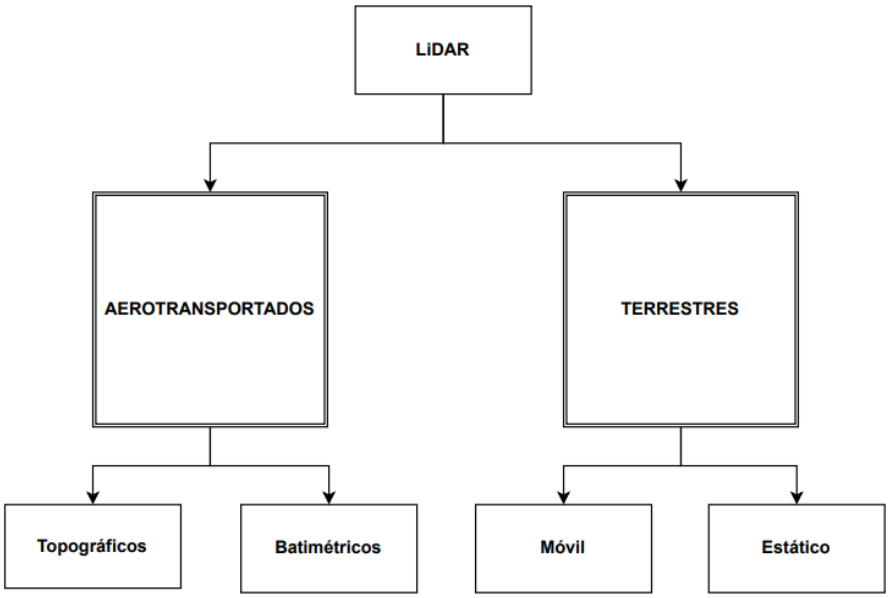

Figura 17. Clasificación de los sistemas LiDAR.

En este proyecto el tipo de LiDAR con el que se está trabajando es el terrestre móvil, ya que el objetivo es integrarlo dentro un prototipo de vehículo autónomo, es decir, una plataforma en movimiento.

Por otro lado, tal y como hemos adelantado ya, los dispositivos LiDAR también se pueden clasificar según el tipo de imagen que se obtenga, distinguiendo tres categorías distintas: 1D, 2D y 3D. Los tres tipos trabajan del mismo modo, la diferencia se encuentra en utilizar un láser fijo o un sistema de escaneo, y también la cantidad de haces láser utilizados en cada dispositivo.

- **1D**: se necesita únicamente un haz de láser cuyo objetivo es medir la distancia entre el dispositivo y el obstáculo, sólo un eje.

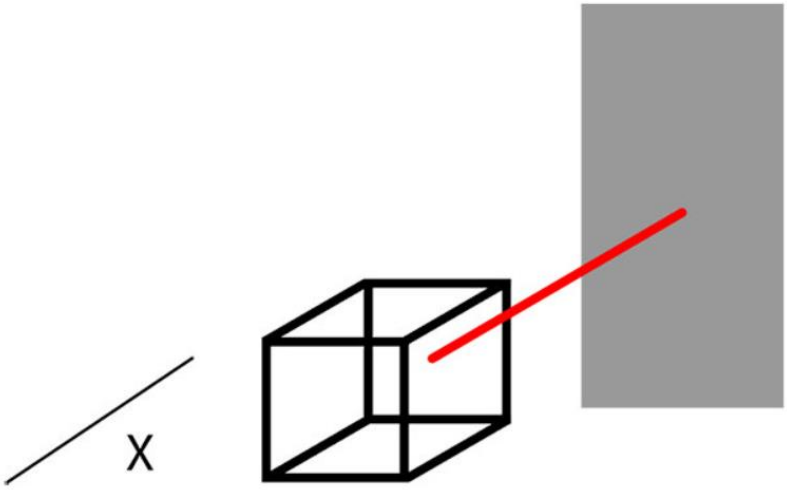

Figura 18. Esquema LiDAR 1D [29].

- **2D**: también es necesario únicamente un haz de láser, en cambio, este tiene un movimiento giratorio obteniendo, tanto la distancia como una posición de dos ejes.

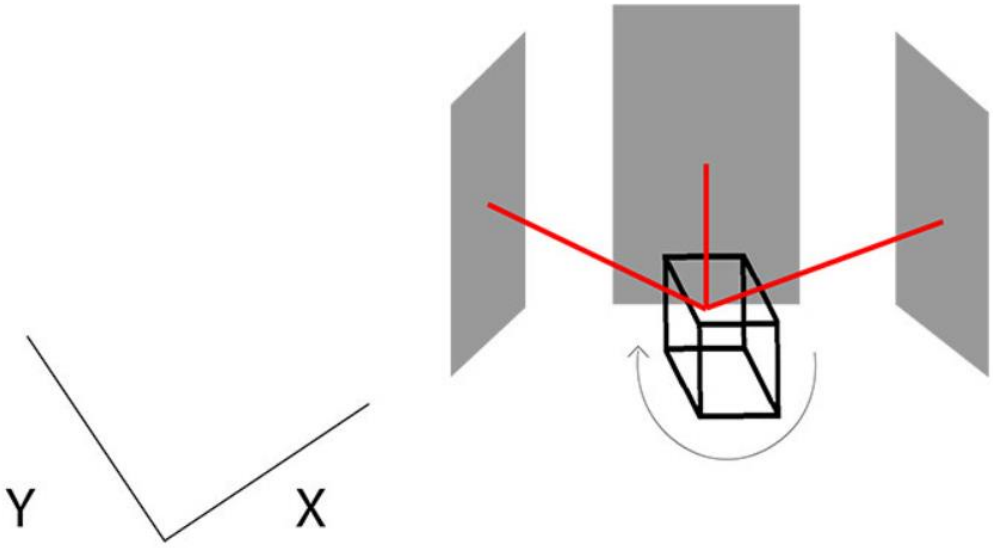

Figura 19. Esquema LiDAR 2D [29].

- **3D**: se necesitan varios haces de láser girando al mismo tiempo, así se obtienen tres ejes que forman una nube de puntos y que cada punto constituye una posición distinta. Cada haz tiene designado un ángulo con respecto a los otros haces.

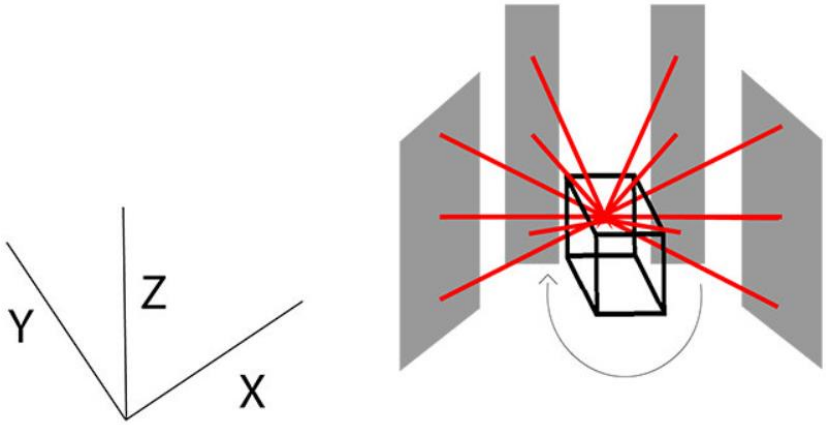

Figura 20. Esquema LiDAR 3D [29].

2.2.4 PROPIEDADES PRINCIPALES DE LIDARS PARA VEHÍCULOS AUTÓNOMOS

El tipo de LiDAR utilizado en este proyecto es el terrestre móvil ya que el objetivo es integrarlo en una plataforma en movimiento, como ya se ha mencionado anteriormente. A la hora de elegir un sensor LiDAR para vehículo autónomo, dependiendo del uso que

se le quiera dar, hay que tener en cuenta una serie de parámetros principales que se describen a continuación [30] [31]:

- 2D o 3D: un LiDAR 2D consigue detectar objetos, pero el LiDAR 3D proporciona una visión más detallada de la nube de puntos con el objetivo de determinar la forma y profundidad de los diferentes obstáculos. Por tanto, el LiDAR 3D es el que se ha decidido utilizar en este proyecto, ya que se percibe un entorno mucho más preciso y se puede conseguir tanto la localización del dispositivo como la creación de un mapa 3D.
- Campo de visión (FoV): se divide en dos tipos que son eje vertical y eje horizontal. El eje horizontal puede proporcionar una visión de hasta 360 grados con lo que se obtiene un plano de la superficie. El campo de visión vertical puede tener un rango de hasta 30 o 40 grados. Para un mismo número de capas, un menor rango ofrece una nube de puntos más compacta, lo que se traduce en una visión más precisa del entorno donde poder detectar el tipo de obstáculo. Por el contrario, si se aumenta el rango aumentará el campo de visión pudiendo percibir una imagen más amplia del entorno, aunque con menos resolución En nuestro caso, el LiDAR que utilizamos tiene un campo de visión vertical de 30 grados. Esta característica es exclusiva de los LiDAR 3D, ya que los LiDAR 2D tienen un campo de visión que consiste en un único plano, al contar con una única capa.
- Número de canales: consiste en el número de haces en el que se divide el campo de visión vertical. Un LiDAR con más canales ofrece una nube de puntos más detallada y precisa. La cantidad de canales depende de con qué exactitud necesites detectar un obstáculo y el presupuesto que se tenga. En este proyecto, se eligió un LiDAR de 16 canales ya que ofrece una visión precisa y el coste es asequible.
- Puntos por segundo: esta propiedad representa el número de mediciones láser que se obtienen por segundo, depende del número de canales, número de vueltas y de la frecuencia a la que funcione el dispositivo. Nuestro sensor LiDAR procesa 320000 puntos por segundo.
- Alcance: Esta propiedad indica las distancias a las que el LiDAR es capaz de detectar obstáculos. Por norma general, un LiDAR puede detectar la distancia a la que se encuentra un obstáculo cuando este se halla entre menos de un metro hasta 200 metros de distancia, sin embargo, el LiDAR tiene dificultades para detectar obstáculos en distancias cortas. Cuanto mayor alcance, más caro será el sensor. El LiDAR utilizado en este trabajo tiene un rango de alcance desde 20cm hasta 120m, valores adecuados para localizar el vehículo en un parking.
- Condiciones ambientales: Una de las problemáticas a las que se enfrentan los sensores que integran los vehículos autónomos es su susceptibilidad ante condiciones meteorológicas adversas. En el caso de las cámaras, por ejemplo, además de ver reducida su capacidad en situaciones de niebla, nieve o lluvia, se le añaden dificultades a la hora de trabajar en condiciones de poca luz, en cambios bruscos de iluminación, o situaciones con el sol de cara. Con respecto a otro tipo de sensores, el LiDAR funciona bastante bien en situaciones adversas, viéndose poco o nada afectado en situaciones de penumbra u oscuridad total. Sin embargo,

sí que se ve afectado por condiciones como lluvia, nieve o granizo. Una de las principales ventajas de este tipo de sensor es que funciona independientemente de si es de día o de noche, lo cual lo hace una tecnología atractiva para la conducción autónoma.

En cada propiedad se ha ido especificando las características que tiene el LiDAR con el que se trabajará en este proyecto. El sensor que hemos elegido deberá proporcionar la información que utilizaremos para el guiado del vehículo. Para convertir esta información bruta del entorno en datos útiles para el guiado es necesario aplicar un procesamiento a estos datos. Para ello, se han analizado una serie de técnicas para obtener una localización y mapeo simultáneo (SLAM) que se explicarán en el siguiente apartado. Aunque antes de explicar estas técnicas es necesario conocer las diferentes formas de comunicarse con el LiDAR para recibir esta información en bruto que posteriormente será procesada.

2.3 PAQUETES *SOFTWARE* DISPONIBLES

En la sección anterior se han explicado los conceptos principales del funcionamiento del vehículo autónomo, centrándonos especialmente en las tecnologías esenciales para la parte de percepción del vehículo, en concreto, la que principalmente se utiliza en este trabajo que es la tecnología *Light Detection and Ranging*, LiDAR. Pero todas las tecnologías de percepción para tener cierta utilidad necesitan algún tipo de procesamiento posterior. Este procesamiento realizado con *software* tiene dos partes diferenciadas. Por un lado, se tiene la comunicación del dispositivo con la aplicación final que realiza el cálculo, y por otro se tiene la aplicación en sí que realiza el cálculo en base a una serie de métodos suficientemente establecidos en la literatura.

Atendiendo a la comunicación entre el LiDAR y el *software* de procesamiento, podemos encontrar varias posibilidades a utilizar:

- **Matlab**: es una plataforma de programación diseñada específicamente para ingenieros y científicos para analizar y diseñar sistemas y productos. Este programa se utiliza principalmente para analizar datos, desarrollar algoritmos y crear modelos y aplicaciones [32]. En Matlab existen variedad de librerías, hay una en concreto llamada *Lidar Toolbox* para proporcionar algoritmos, funciones y aplicaciones para poder diseñar, analizar y testear el procesamiento del sistema LiDAR [33].
- ***Sockets***: este método permite conectar dos nodos de una red para facilitar su comunicación. Cada lenguaje de programación tiene su propia librería que implementa este método, siendo de los más utilizados las de C++ y Java. El principal problema de este método es que el programador se tiene que hacer cargo de definir la estructura de los datos y configurar los mensajes de envío. Cuando el sistema requiere la comunicación no ya con un solo dispositivo con un tipo de datos, sino con varios dispositivos con multitud de diferentes tipos de datos, esta aproximación suele volverse inmanejable por el aumento de la complejidad derivada de una mala escalabilidad.

- **ROS** (*Robot Operating System*): es un conjunto de librerías y herramientas *software* que colaboran para la construcción de aplicaciones robóticas [34]. Este sistema facilita la comunicación entre los actuadores, los sensores y los sistemas de control, esto es, evita el principal problema que se ha explicado en la solución basada en *sockets*. Además, una de las características principales del sistema ROS es que está orientado a dar comunicación en casi tiempo real, lo que, no siendo lo óptimo para este tipo de sistemas (que sería tiempo real) sí se aproxima lo suficiente. En cuanto a la comunicación con el LiDAR, no solo hay librerías que permiten comunicarse con el dispositivo utilizando las estructuras de datos nativas de los diferentes dispositivos y fabricantes, sino que además es posible encontrar una gran cantidad de paquetes *software* que se comunican con el bróker de ROS para capturar estos datos y procesarlos con diversidad de técnicas para la localización y mapeo del dispositivo que se describen más adelante.

En conclusión, el entorno más factible para trabajar con un dispositivo LiDAR es la comunicación mediante el sistema ROS y será el que analicemos en las siguientes subsecciones.

2.3.1 ROS

ROS (*Robot Operating Sytem*) es un entorno de trabajo de desarrollo de robótica que ofrece un conjunto de librerías y herramientas como se ha comentado anteriormente. Aunque en su nombre se defina como un sistema operativo, no se trata de tal, sino de un entorno de trabajo con una gran cantidad de repositorios que ofrecen paquetes de *software* para todo tipo de aplicaciones robóticas. Además, se basa en multilenguaje, es decir, admite diferentes lenguajes de programación como Python o C++ entre otros. ROS es *OpenSource* lo que quiere decir que su *software* es completamente abierto [34]. Para trabajar con este entorno hay que conocer los siguientes conceptos básicos:

- **Master**: ROS funciona como una red de nodos que se comunican entre sí usando tópicos, tal como se puede observar en la Figura 21 . En el núcleo de esta red se encuentra el ROSCore, quien en cierta manera centraliza las comunicaciones. ROSCore inicializa un nodo Máster necesario para establecer las comunicaciones entre los diferentes nodos que irán conformando la red de nuestra aplicación. Una vez la comunicación entre los nodos se ha establecido esta ya no pasará por el nodo maestro [35].
- **Nodos**: son procesos que realizan cálculos. Los nodos son ejecutables que se pueden comunicar con otros nodos una vez estén inicializados. Existen dos métodos para que los nodos se comuniquen entre sí: publicación/suscripción y solitud/repuesta.
- **Mensajes**: los nodos se comunican entre sí enviando mensajes. Los mensajes tienen una estructura de datos predefinida que facilita su transmisión. Esta estructura de datos puede estar estandarizada o se puede crear una nueva. La creación de una estructura nueva requiere que este definida tanto en el nodo publicador como en el nodo suscriptor.

- **Tópicos**: los mensajes se envían a través de un sistema de transporte con semántica de publicación/suscripción. Este método de publicación/ suscripción consiste en la transmisión de datos en un medio de difusión, es decir, multipunto. Un nodo envía un mensaje publicándolo en un tópico. Otro nodo necesita saber qué datos se están publicando en ese tópico, por tanto, se suscribe a él para obtener los datos correspondientes.
- **Servicios**: se trata de otro método de comunicación punto a punto para que los nodos puedan transmitir información mediante solicitud/respuesta. La comunicación *request/response* se basa en la transmisión de los datos entre dos nodos. Se distinguen dos partes, por un lado, se encuentra el nodo cliente que es el encargado de enviar la solicitud, *request*, por otro lado, el nodo servidor recibe la petición del nodo cliente, que le proporciona una respuesta, *response*. La ventaja de este método es que la comunicación es más eficiente, ya que no existen datos transmitiéndose continuamente, solo bajo petición [35].
- **Bags**: son un formato de ficheros para poder guardar y reproducir los datos de los mensajes. En un fichero bag puede recogerse toda la información enviada en diferentes tópicos durante un tiempo determinado. Esta información puede reproducirse posteriormente desde el fichero como si fuera el propio dispositivo ROS quien estuviera transmitiendo la información. Esto es especialmente útil con dispositivos LiDAR, pues es posible grabar en un fichero bag la percepción del LiDAR del entorno y reproducirlo posteriormente para trabajar con esos datos sin necesidad de repetir el trayecto.

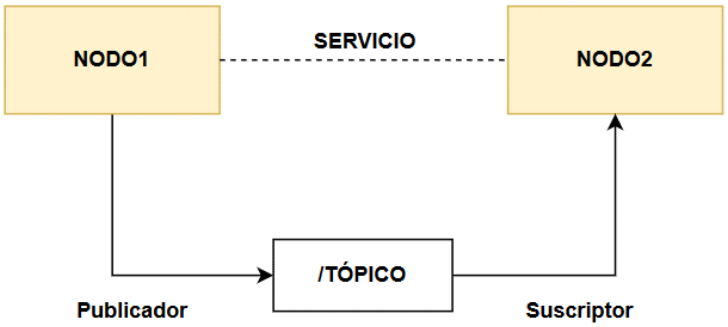

Figura 21. Comunicación ROS.

Los pasos necesarios para trabajar con ROS se explican a continuación [34]:

1. Creación de un espacio de trabajo: consiste en la ejecución de los comandos que se observan en la Figura 22. El procedimiento es crear las carpetas requeridas, cambiarnos de directorio desde la terminal e inicializar el espacio de trabajo.

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
```

Figura 22. Creación del espacio de trabajo [35].

2. Compilación del espacio de trabajo con el comando *catkin_make*, Figura 23.

```
$ cd ~/catkin_ws/
$ catkin_make
```

Figura 23. Comando de compilación del espacio de trabajo [35].

3. Los ficheros relevantes dentro de un espacio de trabajo son: package.xml, contiene la información del proyecto y sus dependencias y CMakeLists.txt, se encarga de describir cómo compilar el código fuente e instalar las dependencias necesarias. Dentro de un espacio de trabajo se puede trabajar con múltiples paquetes.
4. Para crear un paquete hay que situarse en el directorio ~/catkin_ws/src y ejecutar el comando que se muestra en la Figura 24.

```
$ catkin_create_pkg
```

Figura 24. Creación de paquete ROS [35].

5. Una vez se tiene creado el paquete, sólo hace falta crear el nodo dentro del código fuente y darle la funcionalidad requerida, es decir, si se va a encargar de publicar tópicos o suscribirse a ellos. Los nodos se pueden escribir en distintos lenguajes de programación, las librerías principales son: rospy, librería para programar en Python y roscpp, librería para programar en C++. Para saber cómo crear un nodo y publicar un tópico o suscribirse a él se puede seguir la información proporcionada por la Wiki ROS [35].

Una vez se ha descrito el funcionamiento básico para crear un espacio de trabajo donde poder compilar el código fuente y así crear un nodo, lo siguiente es compilar y lanzar ese paquete para que se pueda publicar la comunicación. Antes de lanzar el paquete hay que lanzar el ROScore para que este establezca la comunicación entre nodos.

Dependiendo de la información que se está transmitiendo podemos utilizar diferentes herramientas diseñadas en ROS. La herramienta fundamental para poder visualizar información gráfica es RVIZ (ROS *visualizer*). Se trata de una herramienta que permite una vista del modelo del robot o vehículo, se encarga de suscribirse a los tópicos de los sensores para obtener la información y reproducirla. RVIZ muestra los datos de sensores 3D, láseres y cámaras en forma de nube de puntos o imágenes de profundidad entre otras funcionalidades.

Actualmente existen dos versiones de ROS. La primera versión fue desarrollada dentro del ámbito académico para probar ciertos conceptos de robótica, y nunca se pensó como un sistema para productos comerciales. Sin embargo, debido al amplio éxito de esta primera versión, se decidió desarrollar una segunda versión que resolviera problemas no abordados en la primera y que por lo tanto fuera más apropiada para sistemas comerciales. A continuación, se describen ambas versiones:

- **ROS1**: se publicó por primera vez en 2007 para proporcionar herramientas de *software* para los usuarios como ya se ha mencionado anteriormente. Este sistema sólo puede ser instalado en Linux o MAC. Para obtener documentación sobre este sistema se creó un repositorio para subir información, llamado ROS Wiki. Desde su creación se han diseñado variedad de distribuciones, como podemos ver en [36]. La última distribución desarrollada en ROS1 se llama *noetic* y se ha asegurado su soporte hasta el año 2025. *Noetic* es la distribución instalada en este trabajo.
- **ROS2**: la primera distribución se publicó en 2017. Esta nueva versión de ROS se desarrolló por la necesidad de mejorar la primera versión de ROS. El desarrollo se realizó bajo los siguientes requisitos: seguridad robótica, comunicación distribuida y control en tiempo real.

En la Tabla 1 se ve una comparativa de las principales características de ambas versiones.

	ROS1	ROS2
Sistema Operativo	Linux/MAC	Linux/Windows/MAC
Lenguajes	C++ /Python 2	C++/Python 3.5
Procesos en un nodo	Un proceso	Varios procesos
Tipo de comunicación	Centralizada	Distribuida

Tabla 1. ROS1 vs ROS2.

ROS ha sido el sistema de comunicación elegido en este proyecto para comunicarnos con el dispositivo LiDAR. Para lograr el objetivo de este trabajo de guiar el vehículo mediante tecnología LiDAR, una vez se ha establecido la comunicación con ROS, el siguiente paso es el estudio de diversas técnicas para la localización y mapeo simultáneo que se describirán en la siguiente sección. Gracias a la técnica SLAM es posible localizar el vehículo dentro de un mapa en un sitio cerrado, lo cual es uno de los desafíos a los que se enfrenta el proyecto TwizyLine en el que se basa este trabajo.

2.3.2 PAQUETES ROS SLAM

La técnica de localización simultánea y mapeado (*Simultaneous Localization and Mapping,* SLAM) se basa en la construcción de mapas sobre un entorno desconocido por un robot y al mismo tiempo estima su trayectoria al desplazarse por ese entorno usando dicho mapa. El hecho de construir un mapa requiere una correcta localización del robot, al mismo tiempo, se requiere de un mapa para obtener una posición precisa [37]. En sitios cerrados existe una gran dificultad de localización, por tanto, con técnicas como esta se consigue dar una solución, es decir, se hace posible que un robot sea totalmente autónomo.

Figura 25. Principales disciplinas de un sistema autónomo [38].

En la Figura 25, se puede observar que las principales disciplinas de un sistema autónomo están divididas en: navegación, mapeado y localización. En este trabajo nos centramos tanto en el mapeo como en la localización para llegar a conseguir una conducción autónoma, por ello, se describirán tres técnicas de SLAM distintas, aunque existen muchas más. Las tres técnicas corresponden a tres paquetes implementados en ROS, por tanto, son de código abierto.

2.3.2.1 HECTOR SLAM

La técnica *Hector Simultaneous Localization and Mapping* está basado en la fusión de datos derivado del escaneo de LiDAR 2D con los datos que proporciona la IMU mediante un filtro de Kalman extendido (EFK). Este paquete fue desarrollado en 2013 por Stefan Kohlbrecher con el objetivo de conseguir una navegación y mapeado autónomo con robots de rescate. Para obtener una navegación fluida, se tiene que generar un mapa lo suficientemente preciso. Para ello, este algoritmo proporciona los varios paquetes con las siguientes funcionalidades [39] :

- **Hector_mapping**: se encarga de ejecutar el nodo SLAM.
- **Hector_geotiff**: su tarea es guardar el mapa y la trayectoria del robot.
- **Hector _trajectory_sever**: proporciona un nodo que se encarga de guardar los datos de las trayectorias con la ayuda de la librería tf (*transform*).

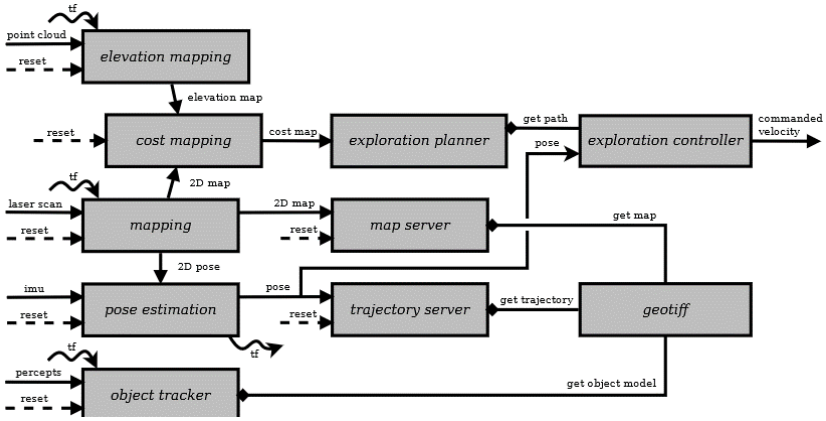

Figura 26. Esquema del sistema del paquete Hector SLAM.

Este algoritmo no requiere datos de odometría, ya que únicamente necesita la información que proporciona el LiDAR y las mediciones de la IMU. El sistema puede proporcionar mapas sobre los entornos incluso si el suelo no es llano. Se puede observar en la Figura 26 que un nodo del paquete fusiona el Filtro Extendido de Kalman (EKF) con la información proveniente de la IMU y del error de posición 2D del LiDAR. El filtro se basa en un modelo genérico del movimiento para vehículos terrestres, por tanto, tenemos un LiDAR tipo terrestre y móvil, y es mejorado gracias a la IMU, sin tener que depender de los datos de la odometría que normalmente pueden ser erróneos debido a las irregularidades del terreno, o inexistentes como en el caso de vehículos aéreos. Este paquete se encarga de generar un mapa con la información que recibe del LiDAR, una nube de puntos 2D. El mapa que se genera es un plano en 2D pero para obtener una representación gráfica que se pueda interpretar mejor se le asigna un determinado valor para elevar el mapa, y así obtener un mapa en 3D. Se puede observar un ejemplo del funcionamiento de esta técnica en la Figura 27.

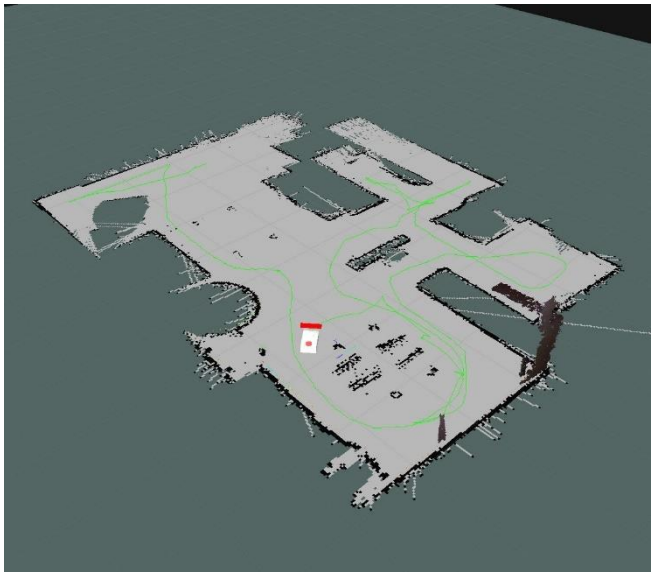

Figura 27. Visión de la técnica Hector SLAM [2].

En conclusión, este paquete ofrece múltiples opciones de trabajo. Para nuestro caso, este algoritmo no lo vamos a utilizar porque queremos tener la precisión que nos ofrece un LiDAR 3D, como el que tenemos, mientras que este paquete solo es capaz de trabajar con LiDAR 2D combinado con una IMU.

2.3.2.2 LOAM

El paquete LOAM fue desarrollado por J. Zhang en 2014. Esta técnica *Lidar Odometry and Mapping* requiere tanto la odometría necesaria para obtener la ubicación del vehículo como mapear el entorno desconocido en tiempo real. Por consiguiente, el método tiene como objetivo la estimación del movimiento y el mapeo utilizando la nube de puntos obtenida de un LiDAR 3D. Dado que la nube de puntos se recibe en instantes de tiempo

[2] Fuente de la imagen: https://joshvillbrandt.com/2016/01/16/mecanumbot-hector-slam/

diferentes, se produce una distorsión en la nube de puntos debido al movimiento del LiDAR. Esta técnica se divide en dos algoritmos [40]:

- Por un lado, el algoritmo de odometría estima la velocidad del vehículo y corrige la distorsión en la nube de puntos del LiDAR.
- Por otro lado, el algoritmo de mapeo se encarga de comparar y registrar la nube de puntos para la creación del mapa.

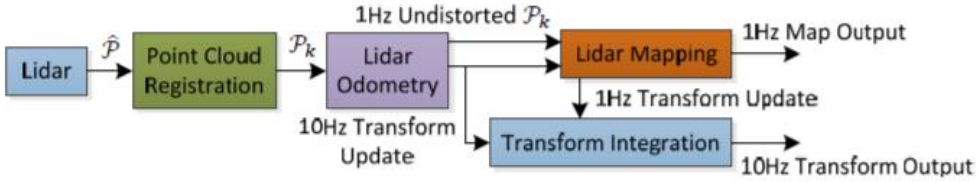

Figura 28. Diagrama de bloques del sistema software, LOAM [40].

La combinación de estos dos algoritmos garantiza la viabilidad de la técnica para ser resuelto en tiempo real. Es condición indispensable disponer de un LiDAR 3D para la creación de nube de puntos del entorno que rodea al vehículo. Los parámetros del algoritmo pueden ser ajustados según las condiciones que tenga el usuario para hacer funcionar su aplicación final.

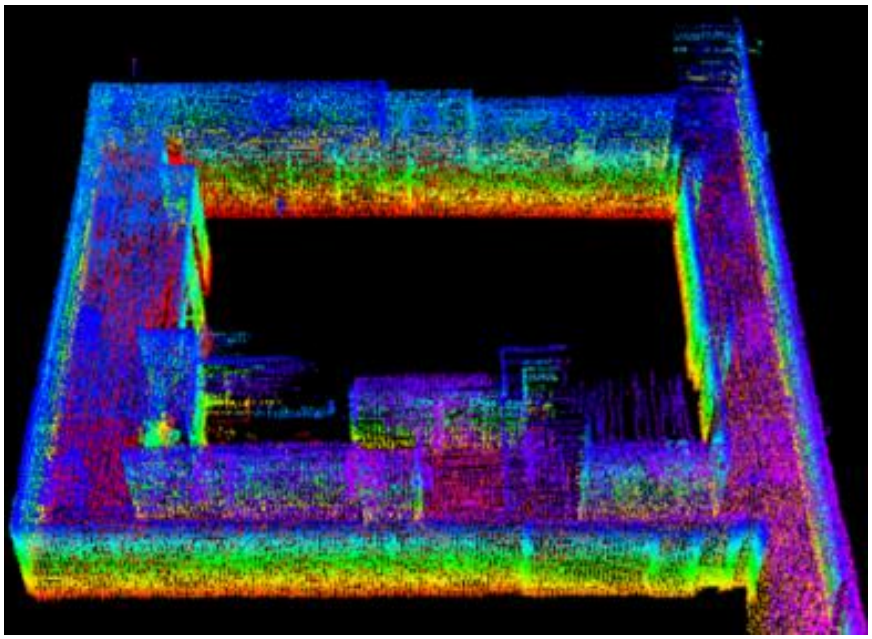

Figura 29. Mapa de puntos obtenido con el paquete LOAM [40].

Este paquete se describirá con más detenimiento en el apartado 5.4, ya que nos centraremos en él para desplegarlo y tratar de alcanzar los objetivos del trabajo. Una de las grandes ventajas de este paquete es que solamente necesita alimentarse de la nube de puntos para poder obtener sus resultados, lo cual lo convierte en una herramienta adecuada para nuestro propósito, ya que en el interior de un parking no se tiene cobertura GPS.

2.3.2.3 HDL_GRAPH_SLAM

HDL_Graph_SLAM es un paquete ROS de código abierto diseñado para implementarlo en tiempo real utilizando un LIDAR 3D. Este paquete fue creado por Kenji Koide. Se basa en la creación de un mapa 3D mediante el sistema SLAM con estimación de odometría cuyo objetivo es obtener la localización del vehículo. Se puede generar mapas tanto de entornos de exterior como de interior, con gran resolución de los elementos que lo rodean. Para entornos de exterior, se necesitaría datos GPS. En cambio, para entornos de interior, es necesario una IMU [41]. Este método consiste en cuatro nodos principales:

- *prefiltering_nodelet*: se encarga de filtrar la nube de puntos.
- *scan_matching_odometry_nodelet*: está basado en la estimación de la odometría.
- *floor_detection_nodelet*: detecta los planos del suelo.
- *hdl_graph_slam_nodelet*: se encarga de fusionar la odometría estimada y los datos de los planos del suelo.

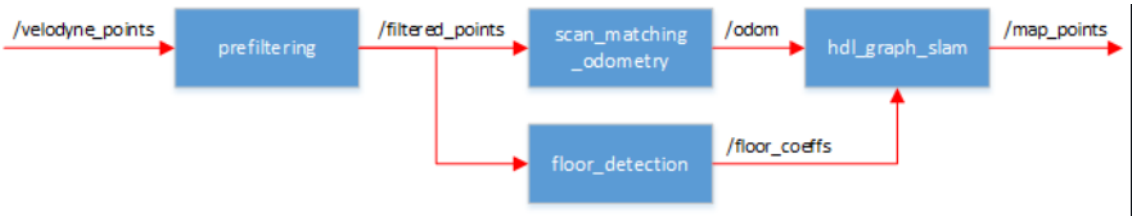

Figura 30. Esquema general del sistema HDL_Graph_SLAM[3].

En el paquete de ROS existen variedad de restricciones que se pueden habilitar o deshabilitar en función de los diferentes parámetros que hay en el fichero *launch* del paquete, los parámetros que pueden modificarse son: odometría, GPS, IMU, etc. En la Figura 31 se puede observar un ejemplo del funcionamiento de esta técnica.

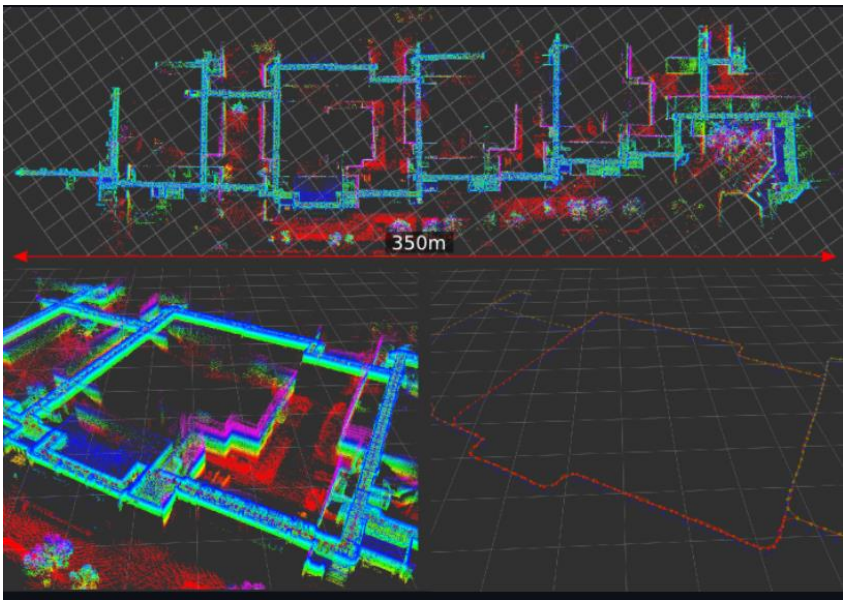

Figura 31. Mapa de puntos y mapa generado con el paquete HDL_graph_SLAM[4].

[3] Fuente de la imagen: https://github.com/koide3/hdl_graph_slam
[4] Fuente de la imagen: https://github.com/koide3/hdl_graph_slam

Sin embargo, esta técnica no puede ser utilizada ya que necesita más información como datos GPS o datos IMU para poder obtener la localización del vehículo, y en nuestro caso únicamente tenemos la información del LiDAR. A parte, esta técnica tiene la desventaja de que presenta elevadas desviaciones y tarda en corregirlas, por tanto, no es óptimo para un sistema en tiempo real [42].

3 Arquitectura Eléctrica y Electrónica

El prototipo desarrollado en el proyecto TwizyLine fue un esfuerzo realizado a caballo entre la finalización de los trabajos fin de grado de los miembros del equipo y la presentación de dicho prototipo en el concurso TwizyContest, casi seis meses después. Debido a ello, muchas de las estructuras de alimentación de dispositivos así como el cableado del que dispone el prototipo no están suficientemente documentadas.

Además, la estructura de alimentación eléctrica así como el cableado de comunicaciones (que llamaremos cableado electrónico) y los diferentes protocolos de comunicación están adaptados para un tipo de tecnología que podemos calificar de bajo consumo, dado que se puede alimentar directamente de la batería de 12V. El nuevo equipamiento que se quiere integrar en el vehículo requiere de protocolos de comunicación de alta velocidad además de suficiente potencia de alimentación, que hace inservible parte de la infraestructura eléctrica y electrónica originales.

Por ello en este capítulo se explican los cambios sobre la reestructuración de la arquitectura electrónica dividida en dos partes: alimentación de los nuevos dispositivos y protocolos de comunicación.

Por último, se explica la integración de un multímetro conectado a la conexión de alimentación de la controladora del motor de control lateral para monitorizar su funcionamiento adecuado.

3.1 Características eléctricas de los dispositivos

Antes de empezar a realizar una descripción de la nueva arquitectura de alimentación y comunicación, es interesante hacer un repaso a las características eléctricas y de comunicación de los dispositivos ya integrados así como de los que se quieren integrar para poder comprender las necesidades que se plantean.

3.1.1 Elementos de la arquitectura antigua

La arquitectura antigua del vehículo es explicada en detalle en [43]. En esta sección solo abordamos una descripción muy breve de los elementos que la integran.

Para empezar, los primero elementos de interés serían los sensores y actuadores, que serían los siguientes:

- Sensor magnético: Se trata de un dispositivo donado por la empresa ASTI [44] que funciona alimentado a 12 V. Además, su sistema de comunicación esta

basado en la variación de un nivel de continua, de tal manera que se requiere un conversor analógico digital al otro extremo del cable de comunicaciones.

- Sensores de ultrasonidos: Se tratan de tres sensores modelo EZ2 [45] conectados en serie y cuyo voltaje de alimentación es de 2.5V a 5.5V. También tiene su propio sistema de comunicaciones que consiste en transmitir la informacion de forma analógica. Sus principales objetivos son detectar y localizar objetos en un entorno variado.
- EPOS4 [46] y motor de Maxon [47] : Estos dos elementos forman el sistema de control lateral del vehículo. El motor está alimentado y controlado directamente por la EPOS4 y es a la EPOS4 a la que se hace llegar la alimentación y la comunicación. La EPOS4 puede trabajar entre los voltajes 8 y 75 VDC. Se comunica a través de USB con el modulo de control.
- Miniordenadores de Control y Comunicación: Estas unidades de procesamiento son las encargadas de realizar la tareas de planificación del control de vehículo (lateral y longitudinalmente) así como la comunicación del vehículo con el exterior. Se trata de dos Humming Board [48] industriales que se pueden alimentar con voltajes de 7V a 36V y cuentan con diferentes interfaces de comunicación.
- Otros sistemas: La descripción del resto de sistemas no es especialmente importante para los objetivos de este capítulo, dado que estarán alimentados directamente por los miniordenadores de Control y Comunicación y la comunicación se realiza directamente a través de sus interfaces estandar. La lista de estos dispositivos incluyen, tres arduinos: uno para el control de la acelaración, otro para el control de las marchas, otro para actuar como DAC entre el sensor magnético y el microordenador de control además de procesar y reenviar a este mismo PC la información de los sensores ultrasónicos; un sensor GPS, el modem de comunicación 4G, un dispositivo lector RFID y unos leds conectados al modulo de control para informar al usuario del modo en que se encuentra el vehículo

En la sección 3.2 se detalla la conexión original de estos equipos y las modificaciones introducidas.

3.1.2 NUEVOS ELEMENTOS A INTEGRAR

Los nuevos dispositivos a integrar están fundamentalmente relacionados con la tecnología LiDAR. Esta tecnología es mucho más exigente en capacidad de cálculo e impone por lo tanto una modificación sustancial de las conexiones de alimentación que teníamos hasta ahora. Además, la estructura original de conexiones que se tenía es bastante insegura, por lo que se tienen integrar nuevos elementos de seguridad en el sistema de alimentación global.

La lista de dispositivos más importantes a integrar y sus características de comunicación y alimentación son los siguientes:

- Sensor LiDAR: Se trata un sensor modelo RS-LIDAR-16 de la empresa RoboSense [49]. El sensor en sí mismo se alimenta a través de un transformador que ofrece 32 VDC. El transformador trabaja a voltaje estándar de 220 V y 50 Hz. Por otro lado, la conexión de comunicación, que se encuentra en la misma caja donde se encuentra la alimentación de continua, es una interfaz Ethernet.
- Ordenador de procesamiento: Se trata de un ordenador ZBOX Serie E, con tarjeta gráfica NVIDIA GeForce RTX 3080. La alimentación se realiza con 220 V a 50 Hz. En el aspecto de la alimentación, tiene importancia señalar que este dispositivo puede llegar a consumir hasta 330 watios. Tiene las interfaces típicas de comunicación de un ordenador de sobremesa, varios USB, conexión Ethernet, etc.
- *Inverter*: Se trata de un inversor de continua a alterna modelo *Pure Sine Wave Inverter* 2000W. Como hemos visto, se han incluido varios dispositivos que son alimentados a 220 V y que tienen un consumo relativamente grande. Por lo tanto, se debe incorporar un dispositivo que se encargue de transformar la salida de continua de la batería de tracción del Twizy a 220 V con suficiente capacidad de potencia de salida.
- Balizas de ultrasonidos en interiores: Se trata de unos sensores de posicionamiento modelo *Marvelmind Indoor "GPS"*, de la empresa Marvelmind [13]. Estas balizas no requieren conexión de alimentación y la comunicación se realiza de manera inalámbrica.
- Cámara de visión 3D: Se trata de la cámara modelo RealSense D400 [50]de Intel. En este proyecto no se aborda el funcionamiento de esta cámara, pero sí cómo quedaría situada y alimentada. La cámara se alimenta y comunica a través de un cable USB, con lo que su instalación no tiene dificultad.
- Caja de fusibles: Como ya hemos indicado al principio de esta sección, el sistema actual no cuenta con medidas de seguridad básicas. Por ello se debe añadir una caja de fusibles que permita proteger los elementos más importantes de nuestro sistema y además nos permita detectar posibles problemas en nuestro sistema de cableado.
- Seta de parada de emergencia: También estaría dentro de los elementos de seguridad que faltan. Debido a que es un prototipo que se mueve de forma autónoma, siempre hay posibilidad de fallo catastrófico. En este tipo de proyectos es esencial incorporar una seta de parada de emergencia que permita desconectar todos los sistemas y bloquear el vehículo.
- Spybox CAN: es un concentrador de CAN que está conectado a los módulos de control y de comunicación. Se añade para poder analizar las tramas CAN enviadas por estos módulos y asegurarnos de que todo se está enviando correctamente.

En la siguiente sección se analiza cada conexión eléctrica actual por separado y se explica cómo se ha modificado.

3.2 ALIMENTACIÓN

La anterior arquitectura de alimentación estaba constituida de tres partes distintas:

- Primero, los módulos de comunicación y control estaban alimentados mediante la toma OBD del coche, se puede ver en la Figura 32.

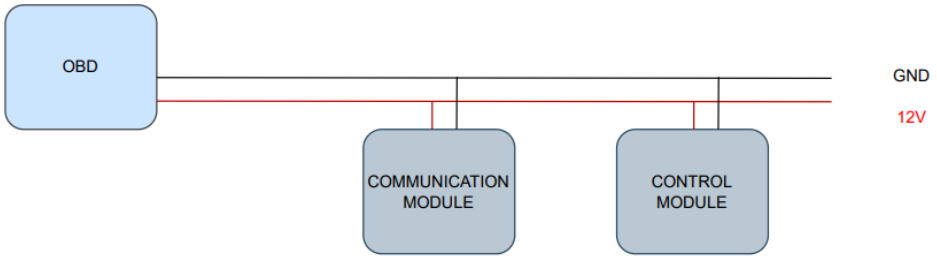

Figura 32. Esquema de alimentación módulos-OBD.

- Segundo, el sensor magnético, cuyo objetivo era el guiado del vehículo estaba alimentado por la toma del mechero del vehículo.

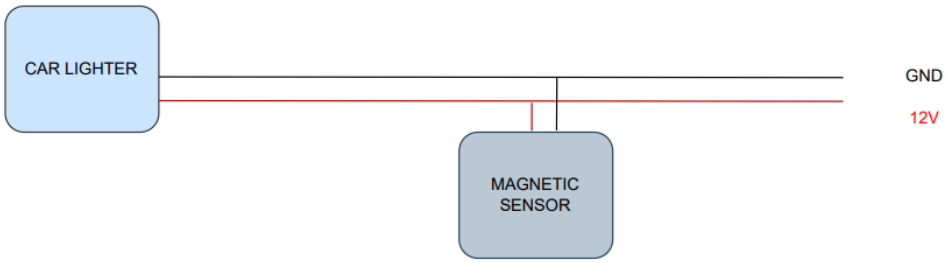

Figura 33. Esquema de alimentación mechero-sensor magnético.

- Tercero, la EPOS4, controladora de potencia del motor de dirección, como observamos en la Figura 34 iba conectada directamente a la batería de 12V.

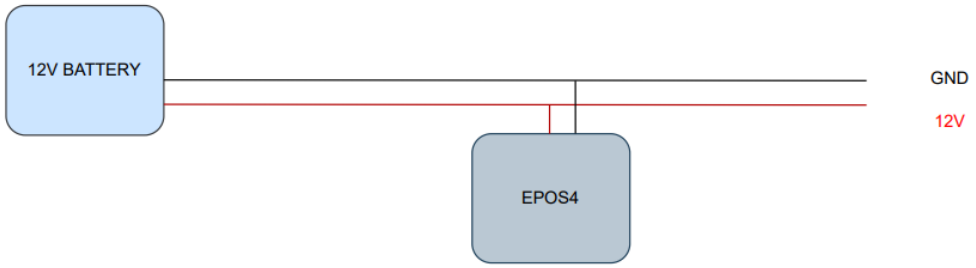

Figura 34. Esquema de alimentación batería-EPOS4.

A continuación, se explicará los cambios diseñados para la nueva arquitectura de alimentación. Está dividido en dos partes distintas:

- Primero, el OBD estaba conectado a los dos módulos como se ha visto anteriormente. En cambio, en la nueva arquitectura se ha añadido una caja de fusibles (Figura 35) cuya función es impedir que ocurra una sobrecarga de

electricidad o un cortocircuito en el vehículo obteniendo una seguridad en la arquitectura eléctrica del vehículo.

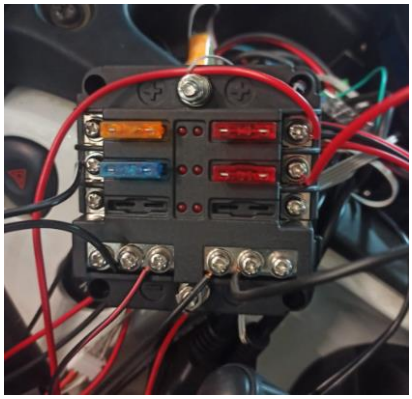

Figura 35. Caja de fusibles integrada en el vehículo.

La caja de fusibles que se ha integrado en el vehículo está formada por 6 vías con indicador LED. En nuestro caso, se han utilizado 4 vías como se puede observar en la Figura 36 y en la Figura 35. Cada vía ha utilizado un fusible de amperaje distinto, el valor correspondiente en cada caso son los siguientes:

- Módulo de comunicación y módulo de control: 2 fusibles de 2A.
- EPOS4: 1 fusible de 15A.
- Sensor Magnético: 1 fusible de 1A.

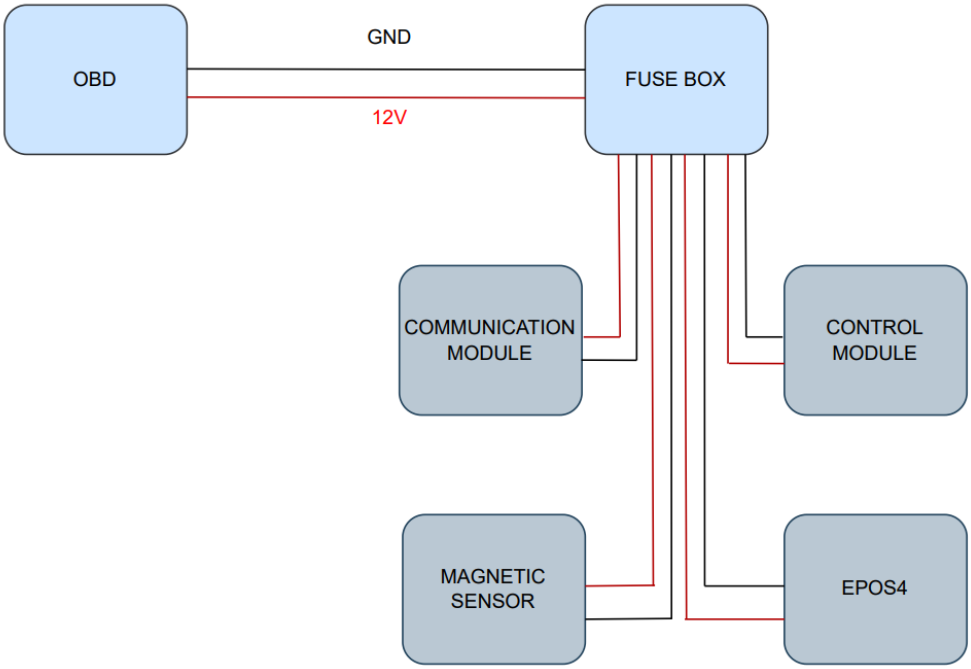

Figura 36. Esquema de alimentación añadiendo caja de fusibles.

- Segundo, se añaden dos tecnologías que son el PC Fusión y el LiDAR, como se puede observar en la Figura 37. Cabe destacar que para la toma de alimentación de ambas tecnologías es necesario un *inverter* conectado a la batería de tracción,

su función es transformar el voltaje para obtener una adecuada alimentación en los diferentes dispositivos.

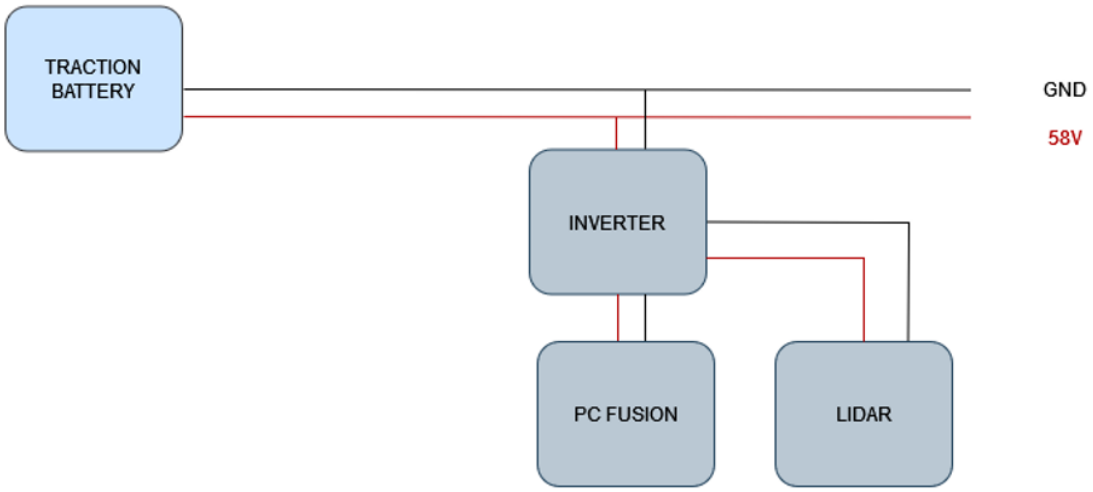

Figura 37. Esquema de alimentación de nuevas tecnologías con la batería de tracción.

Debido al tamaño de estos dispositivos, se ha incorporado un rack hecho a medida en el asiento del pasajero, de tal manera que todos los cables de alimentación y comunicación llegan de manera ordenada a este punto. En la foto de la se puede observar el resultado final, en el cual se muestra la estructura de dónde se encuentra situado el PC Fusion, Inversor y LiDAR en el rack.

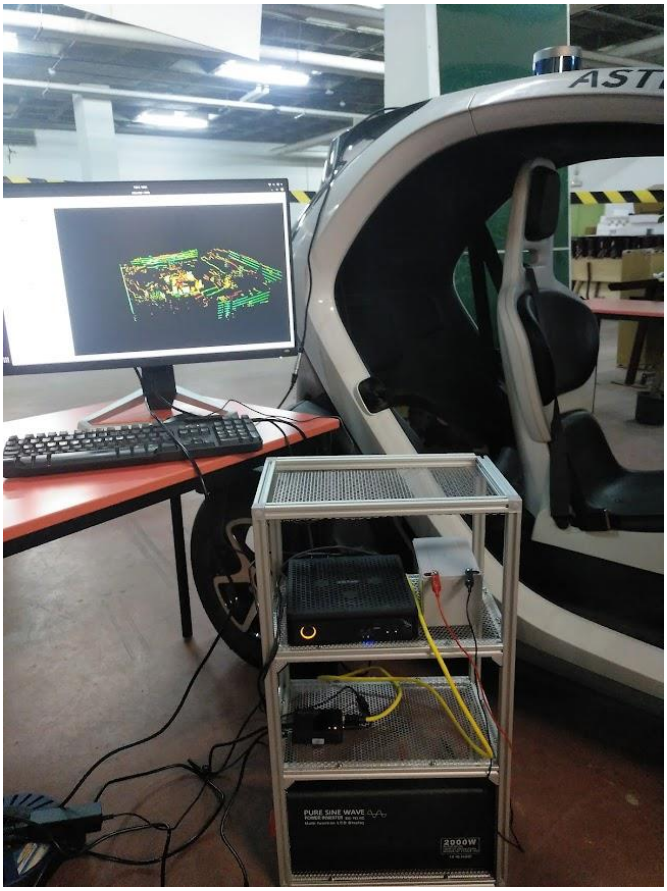

Figura 38. Rack con las nuevas tecnologías integradas.

Como se ha comentado anteriormente, en un proyecto de vehículo autónomo donde una máquina va a tomar el control de un vehículo dentro del cual es posible que vaya montada una persona, es necesario colocar elementos de seguridad que permitan a la persona actuar en caso de emergencia. En el caso de uso que plantea el proyecto TwizyLine el vehículo solo entra en modo autónomo una vez se ha asegurado que no existe ningún usuario dentro del vehículo ni en la zona de conducción autónoma. Sin embargo, al tratarse de un proyecto en desarrollo y de un prototipo en evolución, son comunes las pruebas en las que uno de los miembros del proyecto supervisa el funcionamiento del vehículo desde dentro de este, por lo que este elemento se vuelve fundamental para garantizar la seguridad. En cualquier caso, para todo proyecto en el que una máquina realice una acción cerca de una persona, aún en el caso de que la máquina no debiera actuar o tomar el control mientras esté la persona cerca, es recomendable añadir un elemento de seguridad de este tipo.

El objetivo de este elemento no es otro que el de interrumpir el funcionamiento del sistema dejando funcionales únicamente aquellas partes necesarias para que el usuario pueda volver a una situación fuera de peligro. En el caso de nuestro proyecto, se pretende con este elemento cortar la alimentación que aporta la batería de tracción del coche a los elementos del tren de potencia. De esta forma, al cortar la fuente de energía que alimenta el motor, el vehículo solo continuaría en movimiento por la inercia que hubiera adquirido antes de activar la seta de emergencia.

Al tratarse de un vehículo eléctrico, la batería de servicio de 12V también se alimenta a través de la batería de tracción del vehículo, en lugar de a través de un alternador como ocurre en los vehículos convencionales de tracción térmica. Por tanto, al cortar la alimentación directamente en la fuente, en la batería de tracción, es posible que se afecte a otros sistemas necesarios para evadir la situación de peligro. Por ello ha sido necesario un análisis del funcionamiento del vehículo antes de proceder a la instalación de este elemento de seguridad. Los sistemas críticos que deben mantenerse funcionales tras la activación del dispositivo de emergencia deben ser la dirección y el sistema de frenado.

Dado que el mecanismo de freno del Twizy no lleva ningún tipo de asistente electrónico, la frenada es puramente mecánica. Ni siquiera existe una bomba hidráulica eléctrica encargada de ejercer hacer circular el líquido de frenos. El usuario pisa el pedal de freno, que hace llegar el líquido de frenos hasta las pinzas de freno. La presión del líquido de frenos sobre los pistones hace que estos presionen las pastillas sobre el disco, lo cual genera una fricción que hace reducir la velocidad del vehículo hasta finalmente detenerlo. Debido a la simplicidad y ausencia de electrónica en este proceso, al cortar la alimentación del coche no se ve afectado este sistema.

Además del sistema convencional de freno, el Twizy también cuenta con un freno de mano que actúa sobre las ruedas traseras y cuya activación es completamente manual. Es decir, que el freno de mano se activa mediante el movimiento de una palanca situada bajo el volante, esta palanca tira de un cable que activa unos mecanismos de freno en las ruedas traseras inmovilizando el vehículo. En el Twizy, este sistema al completo también es

totalmente manual y no actúa en el ningún elemento electrónico (ni botón de freno de parking, ni ECU de freno, ni motores para presionar las pinzas traseras). Por lo que este sistema redundante también queda disponible para detener el vehículo, aunque se corte toda la electricidad del vehículo.

En cuanto al sistema de dirección, la dirección original del Twizy no cuenta con sistema de ayuda a la dirección o dirección asistida, por lo que este sistema también es puramente mecánico en el Twizy y no actúa ningún elemento electrónico. El volante gira solidario con la columna de dirección, la cual se conecta al eje delantero mediante la cremallera de dirección. Este movimiento circular se convierte por tanto en un movimiento transversal del eje, que mediante transformaciones a través de los trapecios y las rótulas de las ruedas se convierte en el giro angular de las ruedas. Sin embargo, debido a los cambios que se ejecutaron en el Twizy para robotizarlo y permitir su conducción a través de sistemas electrónicos, se introdujeron en el Twizy algunas modificaciones en la dirección.

En concreto, se engranó a la columna de dirección un motor Maxon de 100W de potencia. Este mecanismo, dificulta ligeramente la conducción del vehículo en modo manual al no existir un mecanismo de embrague entre el motor y la columna que permita separar ambos sistemas, pero no imposibilita la conducción. En cuanto al problema que nos atañe respecto al mecanismo de seguridad, cabe la posibilidad de que la dirección quede bloqueada por el motor engranado a la dirección en caso de algún bug en el *software* o problemas similares.

Por ello, se decide que la alimentación de la controladora del motor Maxon pueda ser interrumpida por nuestra seta de seguridad. De esta forma, al pulsar el mecanismo de emergencia, el motor quedará sin alimentación y el usuario podrá mover la dirección libremente, aunque con más esfuerzo que el necesario en un Twizy serie.

En la Figura 39 se puede observar un diagrama funcional de los sistemas que están conectados a la seta de seguridad. Y en la Figura 40 se detalla el conector que se ha construido para posibilitar la interrupción de la corriente entre la batería de tracción y el resto de sistemas del vehículo.

Para construir el bypass entre la batería y el vehículo que se muestra en la Figura 40, debe crearse un cable que una todos los pines de conector Yazaki macho y del conector Yazaki homólogo hembra. De los 12 pines de cada conector, 11 de ellos conformarán un cable de 15cm de longitud, correctamente recubierto de forma que el cable tenga cierta rigidez. Los otros 2 pines, 1 pin restante de cada conector Yazaki que se muestra en la Figura 41, formarán un único cable de 3m de longitud, que incluirá 2 hilos, 1 del conector macho y 1 del conector hembra. Este cable no es necesario terminarlo en ningún tipo de conector pues irán conectados directamente a la seta de seguridad. De nuevo es necesario que el cable vaya recubierto. En la Figura 41 y en la Figura 42 se muestra el pin del conector hembra y el del conector macho respectivamente que debe conectarse a la seta de seguridad

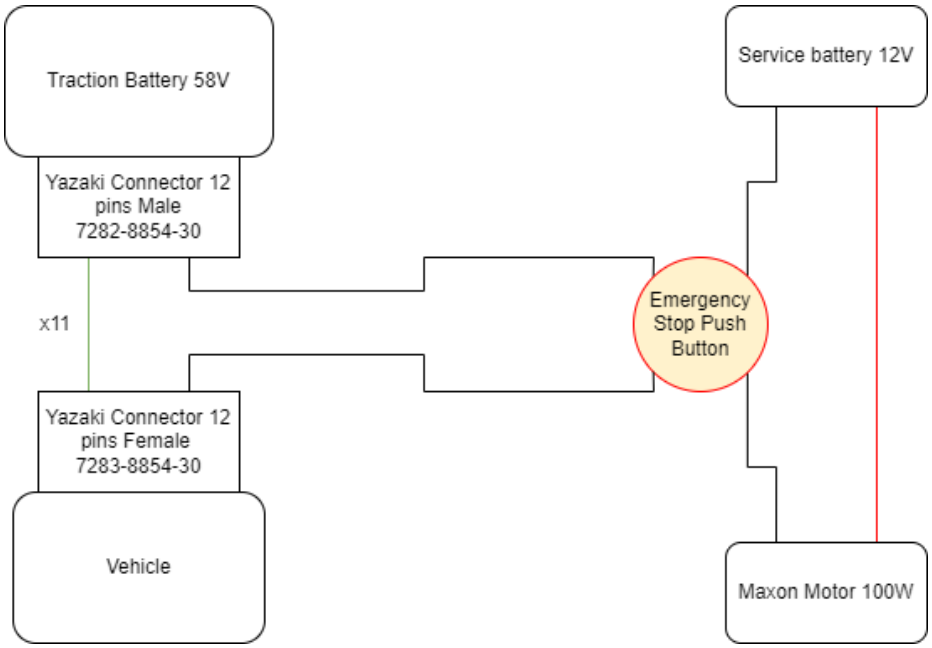

Figura 39. Diagrama de los sistemas conectados a la seta.

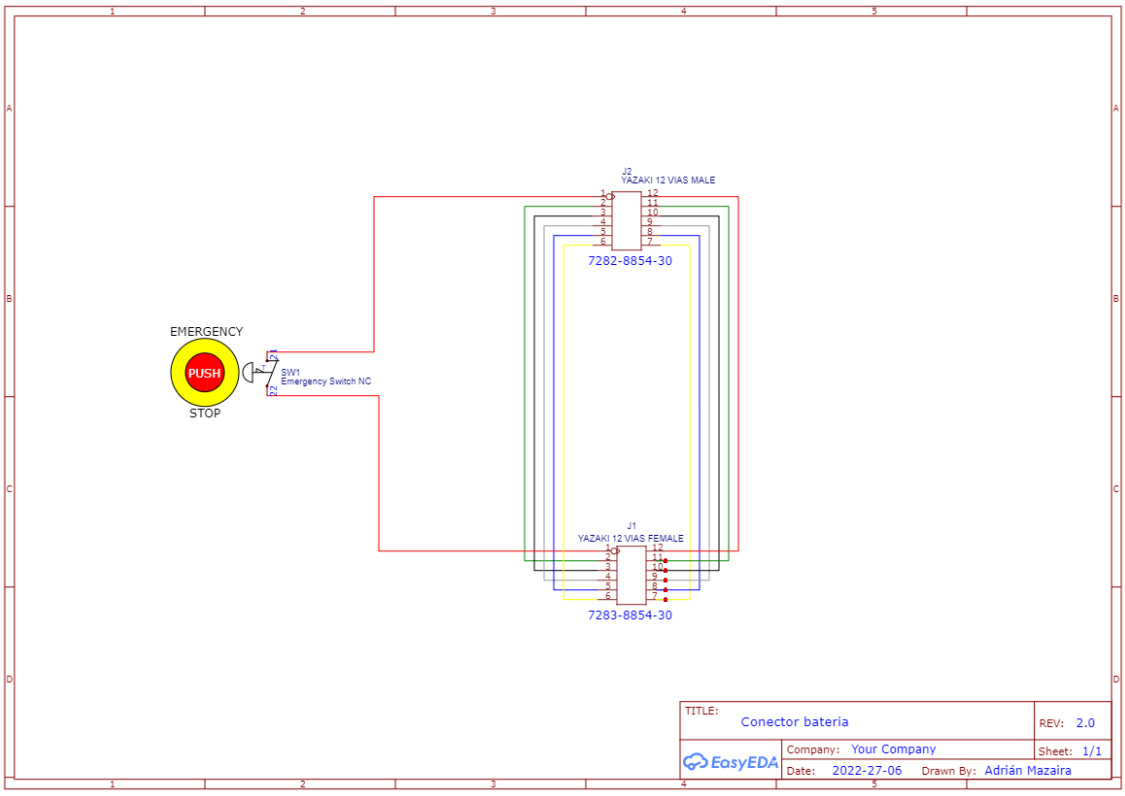

Figura 40. Esquema conexión seta de emergencia.

Figura 41. Conector hembra Yazaki, pin conectado a la seta de seguridad.

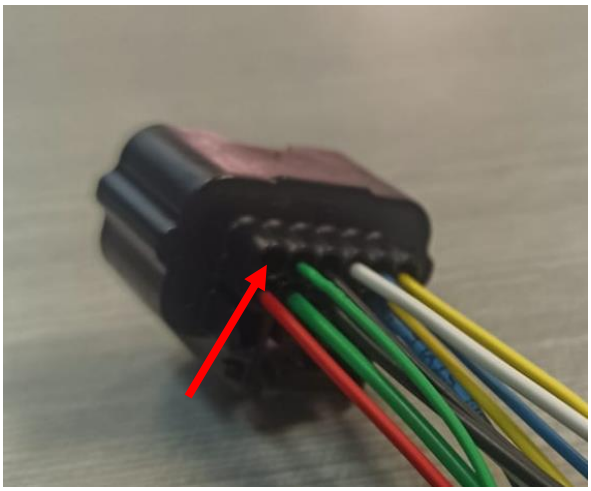

Figura 42. Conector macho Yazaki, pin conectado a la seta de seguridad.

Nótese que este mecanismo no corta directamente la alimentación de la batería al *inverter*, dado que entre estos dos elementos pueden circular corrientes muy elevadas, por tanto, este flujo de energía se realiza mediante un cableado específico para alta potencia. En la Figura 43, se muestra el conector de potencia en naranja a la izquierda, y a la derecha el conector Yazaki macho desde el cual se realizará el bypass. Mediante la interrupción de la corriente en la unión que se detalla en la Figura 40 es posible activar los relés de seguridad internos de la batería de forma que deje de circular corriente de la batería de tracción al resto de componentes del vehículo.

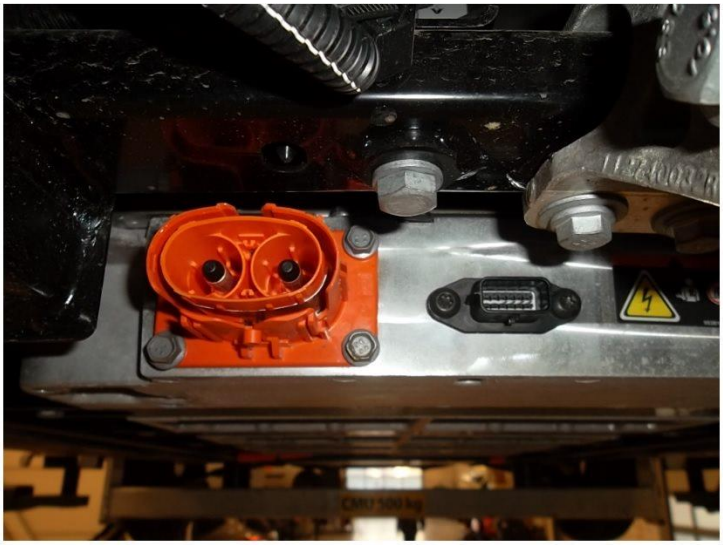

Figura 43. Conector naranja de potencia y conector negro Yazaki macho.

En cuanto al cableado para interrumpir la corriente al motor, se ha intervenido en el conector que une el cable que conecta las bornas de la batería de servicio con la entrada de alimentación de la controladora EPOS-4 del motor Maxon, Figura 44.

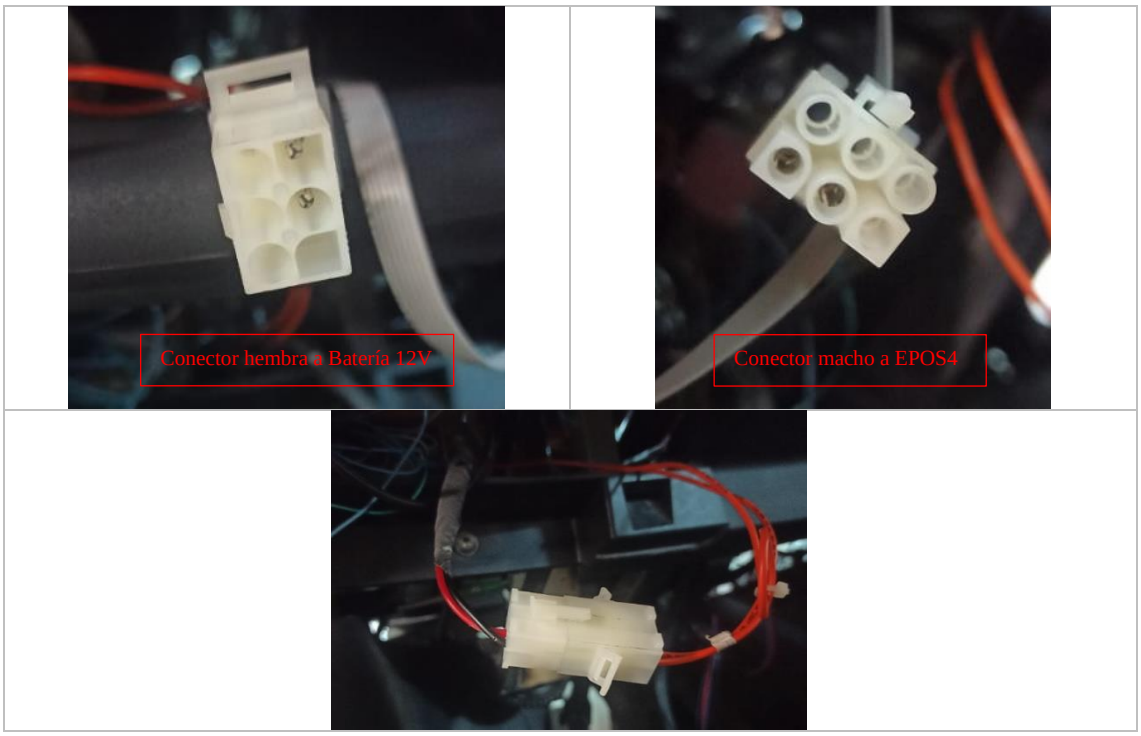

Figura 44. Conector EPOS4-Batería 12V.

Estos cables que parten directamente de las bornas de la batería de 12V se colocaron en la primera fase del proyecto. Este conector entre la controladora y la batería ya se modificó para colocar el multímetro INA 226 que se detalla en este trabajo en el punto 3.4, de modo que el resultado final de esta conexión es el que se detalla en el esquema de la Figura 45.

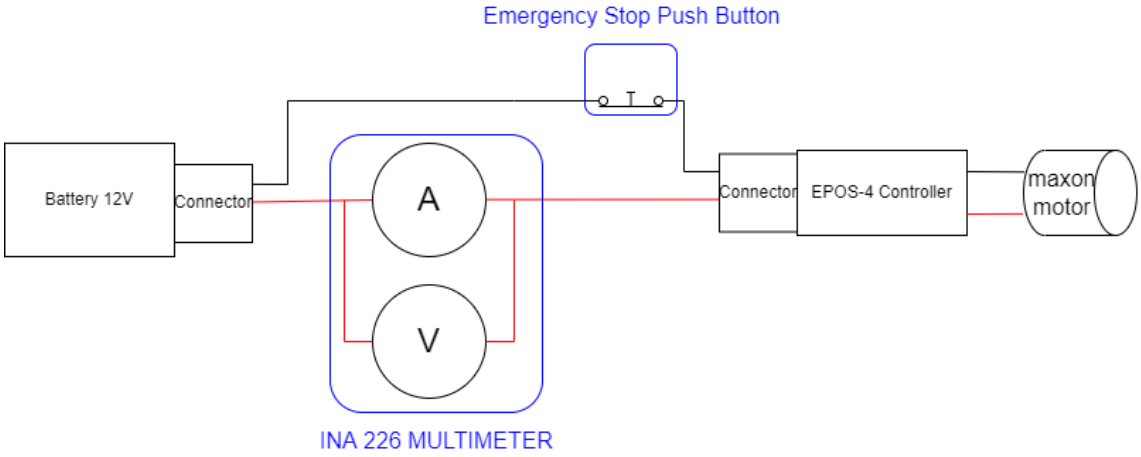

Figura 45. Resultado final del esquema de la seta de emergencia.

Este estudio se ha implementado en el vehículo, en la Figura 46 se puede ver el conector conectado a la EPOS4 uniendo el pin del cable rojo a la seta de emergencia.

Figura 46. Conexión EPOS4-Yazaki hembra-Yazaki macho-Yazaki hembra- Batería, cable rojo conectado a la seta de emergencia.

En la Figura 47, se muestra el resultado final de la integración de la seta de emergencia en el prototipo de vehículo autónomo, TwizyLine.

Figura 47. Resultado final seta integrada en el vehículo.

3.3 PROTOCOLOS DE COMUNICACIÓN

Antes de explicar la arquitectura actual de comunicación, se van a describir brevemente las principales interfaces físicas de comunicación utilizadas en la arquitectura:

- **CAN** (*Controller Area Network*): Se trata de un protocolo muy extendido en la industria de la automoción. Se trata de un sistema de comunicación distribuido (no hay un maestro de red) dónde los diferentes dispositivos se comunican a través de un medio compartido. La versión de bus CAN elegido nos permite trabajar a una tasa de 500 kbps.
- **USB** (*Universal Serial Bus*): Es un protocolo de comunicación punto a punto utilizando un bus serie. Su aplicación principal es la comunicación entre los periféricos y un ordenador.
- **Protocolo MAXON**: protocolo propietario de MAXON [46]. Este protocolo se utiliza en la comunicación entre la EPOS4 y el motor, es decir, la información que envía la controladora del motor para que para que este gire hasta la posición deseada con un perfil de velocidad y aceleración adecuado.
- **Ethernet**: las tres interfaces anteriores ya estaban en la arquitectura anterior. Esta es la nueva interfaz añadida, que es típica de los vehículos autónomos porque es la única con capacidad para transmitir las altas tasas binarias requeridas a un precio suficientemente asequible.

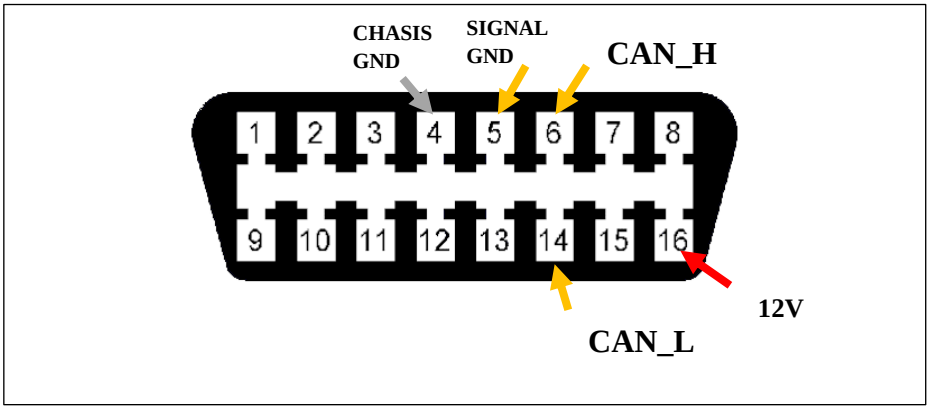

Figura 48. Interfaz estándar OBD. Pines CAN y de alimentación.

El esquema de conexión de comunicación que había en la primera versión del prototipo se puede observar en la Figura 49. Como hemos dicho, está compuesto por tres interfaces físicas de comunicación distintas: CAN, USB y MAXON. Los dos miniordenadores que existen en la arquitectura original, el de control y el de comunicación, se conectan al bus CAN preexistente en el Twizy. Esto quiere decir que a través de este medio compartido se pueden comunicar los dos miniordenadores entre ellos, permitiendo de esta manera realizar todas las tareas de planificación de la conducción autónoma, tal y como se puede leer en el Trabajo Fin de Grado de Ignacio Royuela [3]. Además, los dos miniordenadores pueden acceder a la información que proporcionan las Unidades de Control Electrónico (ECU) que posee el propio Twizy. Para realizar la conexión física se aprovecha la interfaz del OBD, tal y como se puede ver en la Figura 48, dónde las conexiones al bus CAN

están estandarizadas, en concreto, los pines 5 (GND), 6 (CAN_HIGH) y 14 (CAN_LOW).

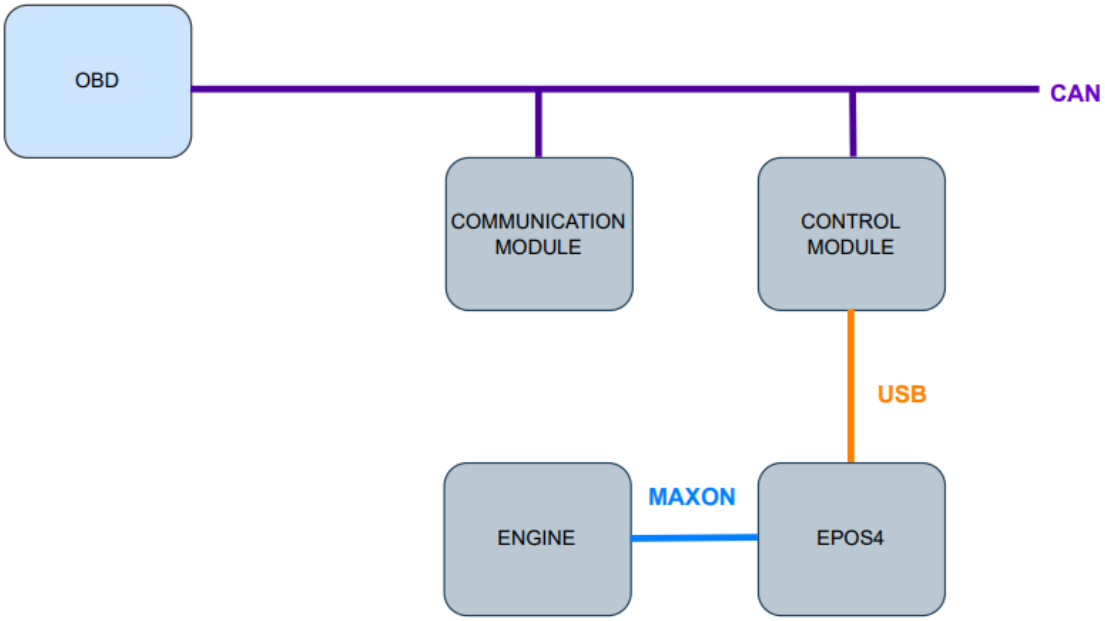

Figura 49. Arquitectura anterior de comunicación.

En la nueva arquitectura se ha añadido un concentrador de CAN (*Spybox* CAN), dispositivo que se puede ver en la Figura 50. Este concentrador permite conectar de manera flexible varios dispositivos al bus CAN a través de interfaces físicas sub-D9, siendo por lo tanto una solución mucho más eficiente. Además, en el caso de querer comprobar las comunicaciones por CAN, este sistema permite acceder al bus de manera muy sencilla.

Figura 50. Spybox CAN.

El concentrador consta de 5 canales CAN. Cada canal CAN consta de 4 puertos DB9 hembra y 1 puerto DB9 macho. La configuración de los pines de estos DB9 se puede ver en la Figura 51.

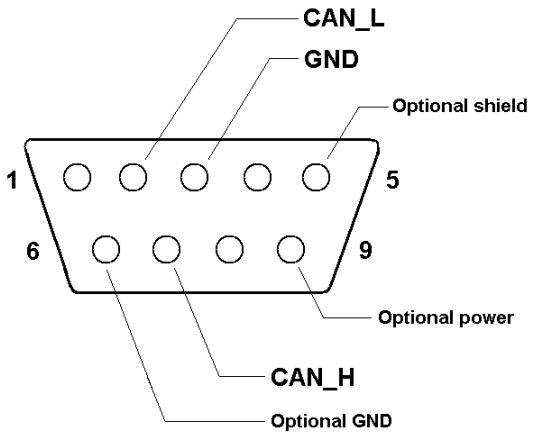

Figura 51. Pines conector DB9 [51].

Por otro lado, en la nueva arquitectura se han añadido dos dispositivos nuevos que requieren una interfaz física de comunicación: la cámara 3D y el LiDAR. El dispositivo LiDAR es el que se integra en este proyecto y como hemos visto anteriormente requiere una interfaz de comunicación de tipo Ethernet.

Gracias a la integración de las nuevas tecnologías incorporadas en el prototipo de vehículo autónomo, se obtiene la parte de percepción del mismo. Se observa en la Figura 52 que se han añadido tanto la cámara como el LiDAR y el PC Fusion. Es importante señalar que el PC Fusión se comunica con el módulo de control a través de una interfaz Ethernet. Teniendo en cuenta la cantidad de datos que va a transmitir al módulo de control sería factible utilizar también la interfaz física CAN, dado que una vez procesados todos los datos de los sensores de percepción, lo único que debe transmitir el PC de fusión es la localización del vehículo con respecto a la ruta que se quiere seguir. Sin embargo, el ordenador no tiene salida CAN, lo que dificulta sustancialmente su uso, y además en un entorno en el que se puede esperar añadir más dispositivos que requieran gran ancho de banda, parece más adecuado utilizar la interfaz Ethernet.

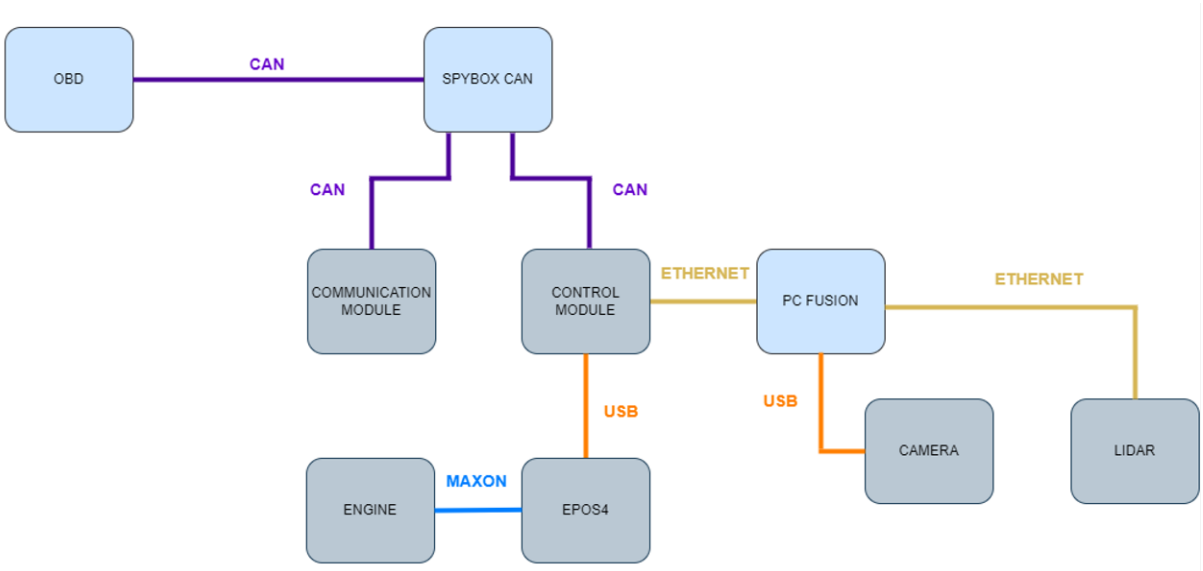

Figura 52. Esquema de la arquitectura actual de comunicación.

3.4 MULTÍMETRO BASADO EN INA 226

Una de las limitaciones del primer prototipo de TwizyLine consistía en la baja velocidad de movimiento del volante, que impedía que la dirección modificara su ángulo en el tiempo que requería el algoritmo de guiado, de tal manera que el vehículo perdía con mucha facilidad la trayectoria, tanto en curvas como incluso en rectas. Para llegar a la conclusión de que la pérdida de la trayectoria se debía a la baja velocidad a la que se movía el volante se realizaron diferentes simulaciones en MATLAB del algoritmo de guiado, cuyos resultados se pueden ver en el Trabajo Fin de Máster de Samuel Pilar [43].

Recordemos que en el prototipo, tal como se recoge en [5], la columna de dirección está engranada a un motor Maxon externo que se controla a través de una controladora de potencia EPOS 4 que a su vez recibe instrucciones del módulo de control, tal y como se ve en las Figura 53 y la Figura 52.

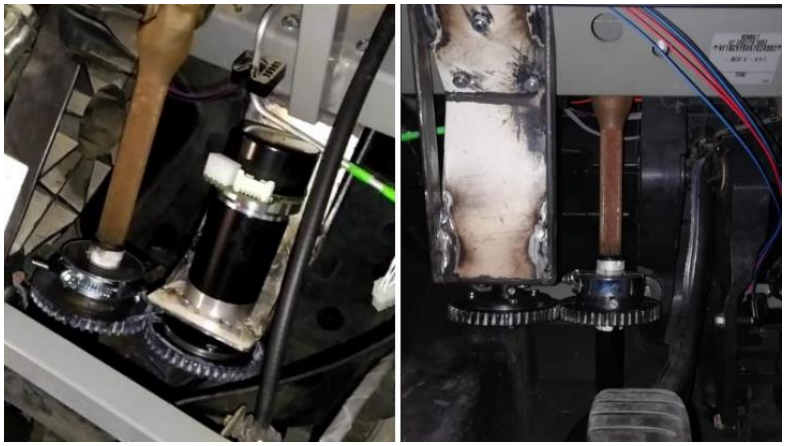

Figura 53. Engranaje de la columna de dirección y el motor de control [43].

Según las especificaciones del fabricante [47], este motor es capaz de desarrollar una potencia de 100W, de forma que alimentado a 12V sería capaz de alcanzar velocidades de hasta 3100 rpm. Según las simulaciones, con las especificaciones del motor se estimaba que el vehículo debía ser capaz de seguir la trayectoria a una velocidad de 5 Km/h. Sin embargo, durante la configuración de los parámetros de control de la EPOS4 se pudo comprobar que trabajando a elevadas rpm se producían frecuentes apagados del motor. Estos apagados ocurrían en numerosas ocasiones aun trabajando en rpms inferiores a las máximas marcadas por el fabricante, mientras que en otras situaciones era posible superar notablemente las rpm máximas que marcaba el fabricante. Esto indicaba que los apagados no se producían al superar un umbral concreto de rpms. Este apagado era más frecuente en las situaciones de máximo par, esto es, cuando la dirección se encontraba cerca del tramo final de la cremallera de dirección. La primera hipótesis que se barajó fue que esta limitación en la velocidad del motor venía dada a su vez por una limitación en la alimentación del motor. El motor, como se ha visto en la sección anterior, se alimenta a través de la EPOS 4, que a su vez está conectada directamente a la batería de servicio del Twizy. Existían dudas de que la batería de servicio fuera capaz de dar la corriente que se requería sin que hubiera una caída del voltaje de alimentación

Para comprobar esta hipótesis, se decidió monitorizar la potencia entregada por la batería a la controladora del motor a cada instante. En primer lugar, se realiza una búsqueda de mercado de dispositivos capaces de monitorizar y registrar voltaje y corriente al mismo tiempo. Los dispositivos más frecuentes para esta tarea suelen ser pinzas amperimétricas con función de registro. Sin embargo, finalmente se desecha esta idea pues los dispositivos dentro del rango de precios asequibles para el proyecto tienen una frecuencia de muestreo demasiado baja para la aplicación deseada.

Habiendo descartado la opción del dispositivo de mercado se decide desarrollar una herramienta propia para la tarea. Después de la pertinente búsqueda se decide que el mejor dispositivo para la tarea es el módulo INA 226, que soporta voltajes entre 0 y 36V DC y corrientes de hasta 20 A [52]. Este módulo, cuenta además con comunicación I2C. Con estas características, se desarrolla el prototipo final que se utilizará para la tarea de monitorización de la alimentación del motor. El prototipo se basa en el módulo INA 226 controlado desde un Arduino Nano, al cual se conecta una pantalla LCD (que también cuenta con comunicación I2C) para mostrar los datos al usuario.

Para el Arduino Nano se desarrolla un *script* que permite comunicar el Arduino con el módulo I2C y muestra al usuario a través de la pantalla LCD, los datos de voltaje, intensidad y potencia tanto en tiempo real como sus valores máximos. Además, se añade un botón conectado a uno de los pines digitales del Arduino para permitir al usuario resetear los valores máximos registrados por el dispositivo de cara a comenzar una nueva medida. El esquema eléctrico se muestra en la Figura 54.

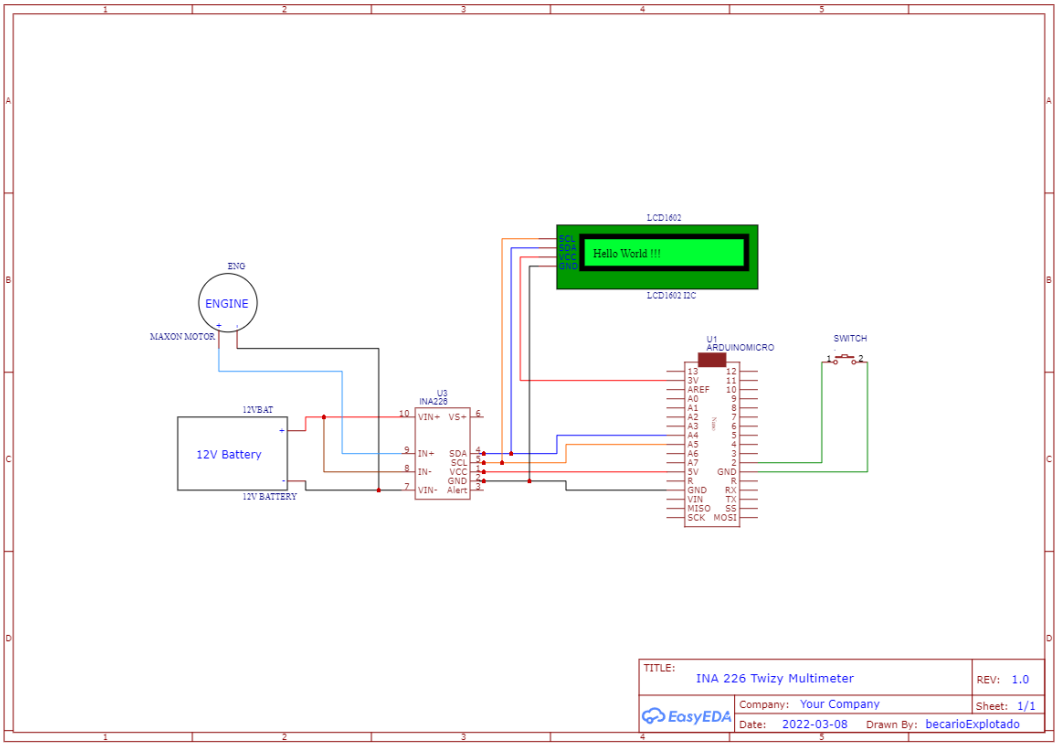

Figura 54. Esquema eléctrico de montaje del multímetro.

El pulsador se conecta al Arduino Nano con resistencia de *pull-up*. Nótese que no se muestra en el esquema ninguna resistencia entre el pin digital y el pulsador. Esto es

porque se utiliza la resistencia de *pull-up* interna con la que cuenta el Arduino Nano. A través de la orden

```
pinMode(2,INPUT_PULLUP)
```

se define el pin digital 2 del Arduino nano como entrada digital con resistencia interna de pull-up, no siendo necesaria añadir una resistencia externa. En la Figura 55 se muestra el esquema de una conexión de *pull-up*. De esta manera, cada vez que el pulsador sea activado, el pin 2 del Arduino se situará en baja induciendo el reseteo de los valores máximos registrados hasta ese momento.

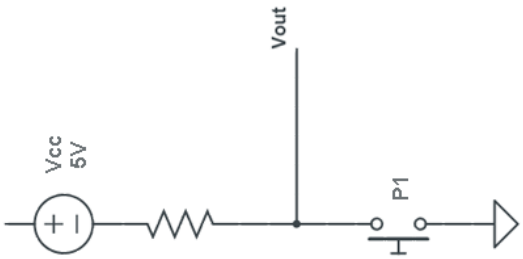

Figura 55. Esquema conexión pull-up.

El prototipo final utilizado y colocado en el vehículo se puede ver en la Figura 56. Se puede observar como la pantalla muestra al usuario los valores de voltaje, intensidad y potencia en tiempo real y también los valores máximos alcanzados en la medición actual. Se observa también en la imagen el botón integrado que permite resetear los valores máximos para iniciar una nueva medición, así como la cubierta robusta de todo el dispositivo que permite su correcta integración en el vehículo.

Figura 56. Multímetro integrado en el vehículo.

Gracias a este dispositivo, Adrián Mazaira, cotutor de este proyecto, fue capaz de diagnosticar que el problema no se encontraba en la alimentación del motor sino en la

configuración de este, es decir, sirvió para comprobar que no había caídas de alimentación y así deducir que el fallo venía de la configuración del motor. Modificándola se logró alcanzar los valores de velocidad de giro descritos por el fabricante en las especificaciones, manteniendo en todo momento la controladora del motor alimentada por la batería de 12V. Este diagnóstico y su posterior solución ayudaron enormemente a la evolución del proyecto.

Antes de finalizar esta sección, en la Figura 57 se puede observar el esquema completo de la arquitectura eléctrica y electrónica actual del prototipo de vehículo autónomo, TwizyLine.

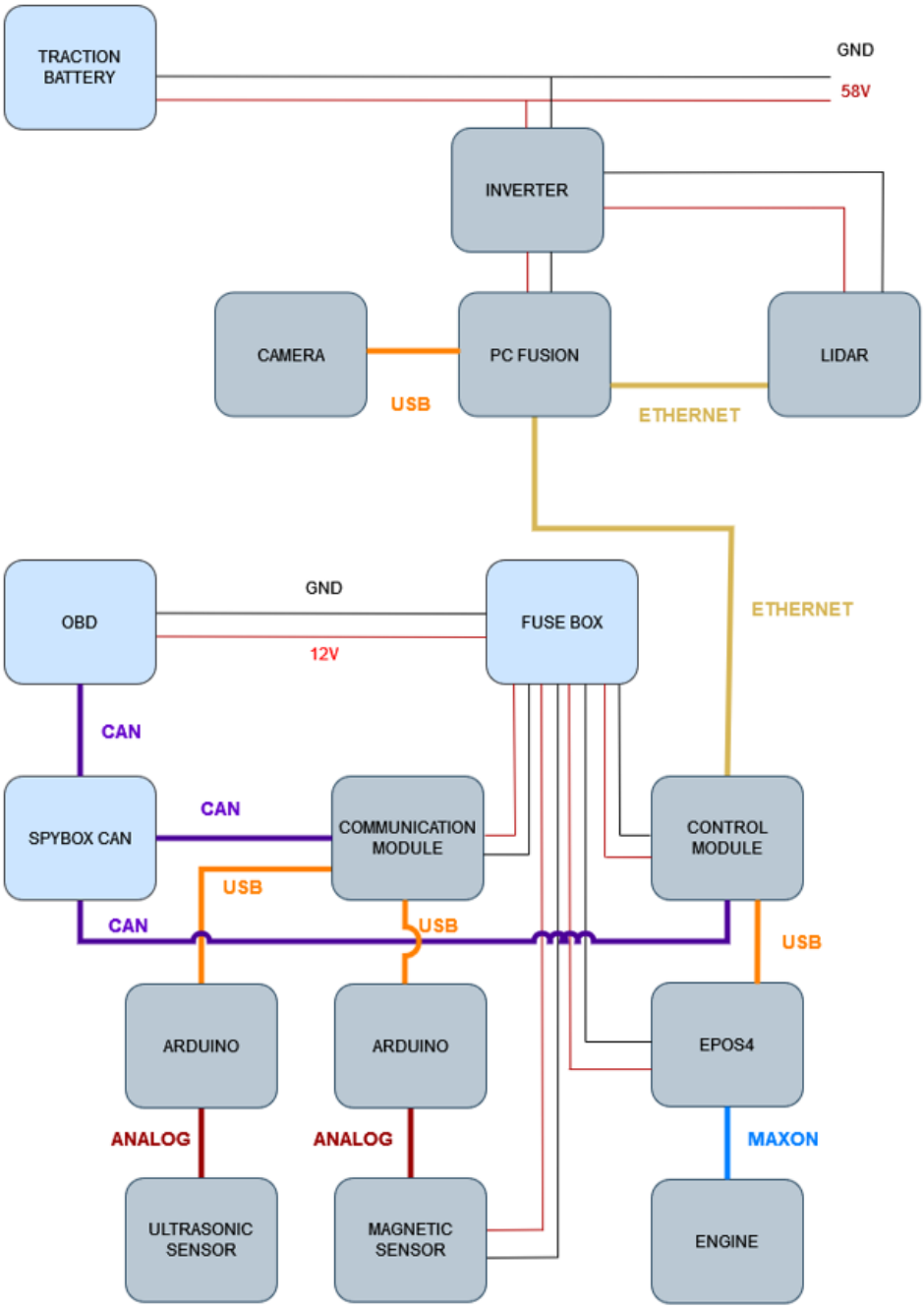

Figura 57. Arquitectura actual prototipo TwizyLine.

4 TECNOLOGÍAS UTILIZADAS

Como se ha dicho en el capítulo de introducción y motivación, el principal objetivo de este trabajo fin de grado es integrar nuevos sensores en el prototipo de Twizyline para hacerlo más funcional. En este capítulo, se explicará en profundidad estos sensores. Los dos sensores que se estudian son el RS-LíDAR-16 de Robosense y el sistema de localización *indoor* por ultrasonidos de Marvelmind. Se abordará no solo las características técnicas, sino también los datos que transmiten y cómo deben ser interpretados. El objetivo final es tener una localización y navegación del vehículo totalmente segura y precisa.

4.1 RS-LIDAR-16

En la sección 2.2, se ha descrito en profundidad la arquitectura y funcionamiento general de los sensores LiDAR. En este caso, se va a estudiar con más detalle el LiDAR empleado en este proyecto, el RS-LiDAR-16 de Robosense. En la Figura 58 se muestra una fotografía de dicho LiDAR.

Figura 58. RS-LIDAR-16 [12].

El RS-LiDAR-16 se trata de un LiDAR 3D de 16 haces. Ha sido diseñado para utilizarlo en aplicaciones de conducción autónoma, percepción del entorno por parte de robots y para el mapeo con vehículos aéreos no tripulados.

Las características más importantes para nuestra aplicación son las siguientes:

- Rango de medición de hasta 150 metros.
- Precisión de medición de hasta 2 centímetros.
- Velocidad de datos de hasta 320.000 puntos/segundo.
- Campo de visión horizontal (*Field Of View*, FOV) de 360°.
- Campo de visión vertical (*Field Of View*, FOV) de 30°.

Otras características importantes se pueden encontrar en las especificaciones mostradas en la Tabla 2.

Sensor	Medición de la distancia del tiempo de vuelo 16 canales Medición del alcance: 40cm a 150m Precisión: ±2 cm Campo de visión (Vertical): ±15º (30º en total) Resolución angular (Vertical): 2º Campo de visión (Horizontal): 360º Resolución angular (Horizontal/Acimut): 0.1º (5Hz) a 0.4º (20Hz) Tasa de rotación: 300/600/1200 rpm (5/10/20 Hz)
Láser	Clase 1 Longitud de onda: 905nm Haz de luz Horizontal: 7.4 mrad, Vertical: 1.4 mrad
Salidas	Tasa de datos: ~300.000 puntos/segundo 100Mbps Ethernet Datagrama UDP, incluye: Distancia Ángulo de rotación/Acimut Reflectividad calibrada Timestamp sincronizado (Resolución: 1 us)
Operación Mecánica/Eléctrica	Consumo de potencia: 12 W Voltaje: 9-32 VDC Peso: 0.87 kg (sin cable) Dimensiones: 109 mm diámetro x 80.7 mm altura Seguridad: IP67 Temperatura de operación: -30º a +60º Temperatura de almacenamiento: -40º a +85º

Tabla 2. Especificaciones RS-LIDAR-16.

El campo de visión vertical está dividido en 16 haces láser con ángulos de -15º a +15º con un total de 30º en intervalos de 2º entre cada haz láser. En la Figura 59 se puede observar los 16 haces láser junto con el ángulo con el que se emite cada haz y el identificador que se utiliza en los mensajes para identificarlos. Por lo tanto, el canal 1 corresponde al haz de ángulo 15º, el canal 2 al de -13º y así sucesivamente.

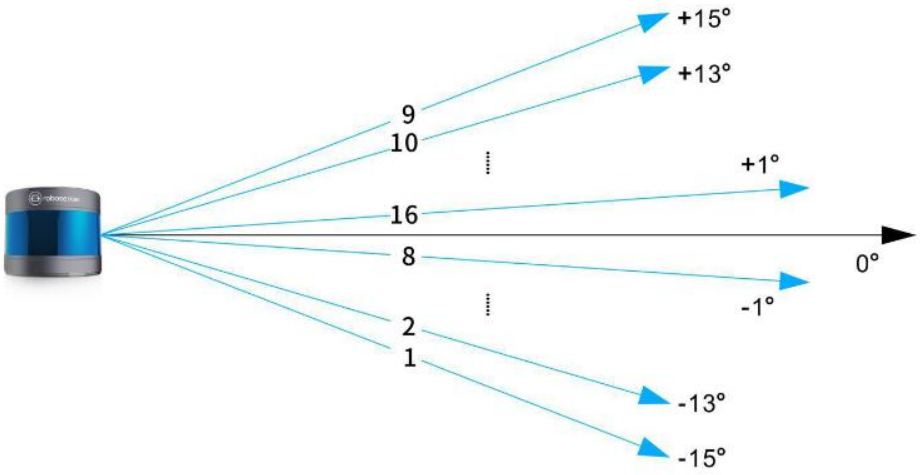

Figura 59. RS-LIDAR-16 ángulos verticales [49].

También es importante señalar el número de puntos capaz de capturar en horizontal por capa. Esto dependerá de la frecuencia de rotación, siendo de 3600 puntos para 5 Hz y 900 puntos para 20 Hz. Esto es crucial, dado que a 150 m la distancia entre dos puntos horizontales varia de 26 cm en el primer caso a 104 cm en el segundo caso, pudiendo pasar objetos de cierto tamaño desapercibidos a dicha distancia.

El RS-LIDAR-16 no solo calcula la distancia al objeto, sino que además realiza un procesamiento de señales que le permite calcular la reflectividad del objeto, permitiendo distinguir un tipo de material de otro.

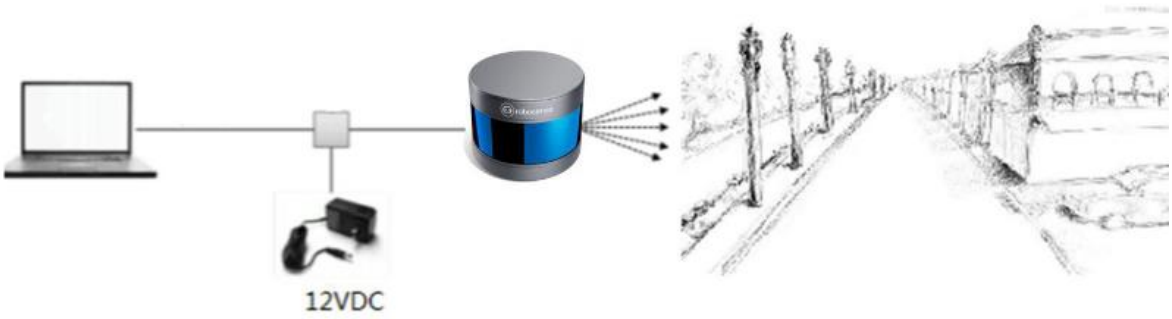

Figura 60. Sistema de imágenes RS-LiDAR [49].

El LiDAR no es capaz de trabajar de manera independiente, sino que debe estar conectado a una unidad de procesamiento. Una vez se conecta y se configura desde ella, el LiDAR envía paquetes de datos con información sobre el acimut, la distancia medida y la reflectividad. A partir de estos datos la unidad de procesamiento debe calcular las coordenadas X, Y y Z en cartesianas.

4.1.1 CÁLCULO DE LA NUBE DE PUNTOS

Como acabamos de comentar, el RS-LIDAR-16 exporta paquetes de datos que contienen diferentes valores: acimut y la distancia al punto. Sin embargo, para obtener una

representación en 3 dimensiones, efecto nube de puntos, es necesario transformar los valores de acimut y distancia a coordenadas X, Y y Z. Este cambio se puede realizar con una simple transformación de esféricas a cartesianas, según las siguientes operaciones:

$$x = r\cos(\omega)\sin(\alpha);$$

$$y = r\cos(\omega)\cos(\alpha);$$

$$z = r\sin(\omega);$$

El origen del sistema de coordenadas está definido en el centro del propio dispositivo. Cada parámetro corresponde con unos datos proporcionados para poder calcular el sistema de coordenadas cartesianas, en la Figura 61 se puede observar gráficamente:

- ***r*** es la distancia proporcionada.
- **ω** es el ángulo vertical del láser, este es fijo para cada capa, como ya se ha especificado anteriormente en la Figura 59.
- **α** es el ángulo horizontal, acimut.

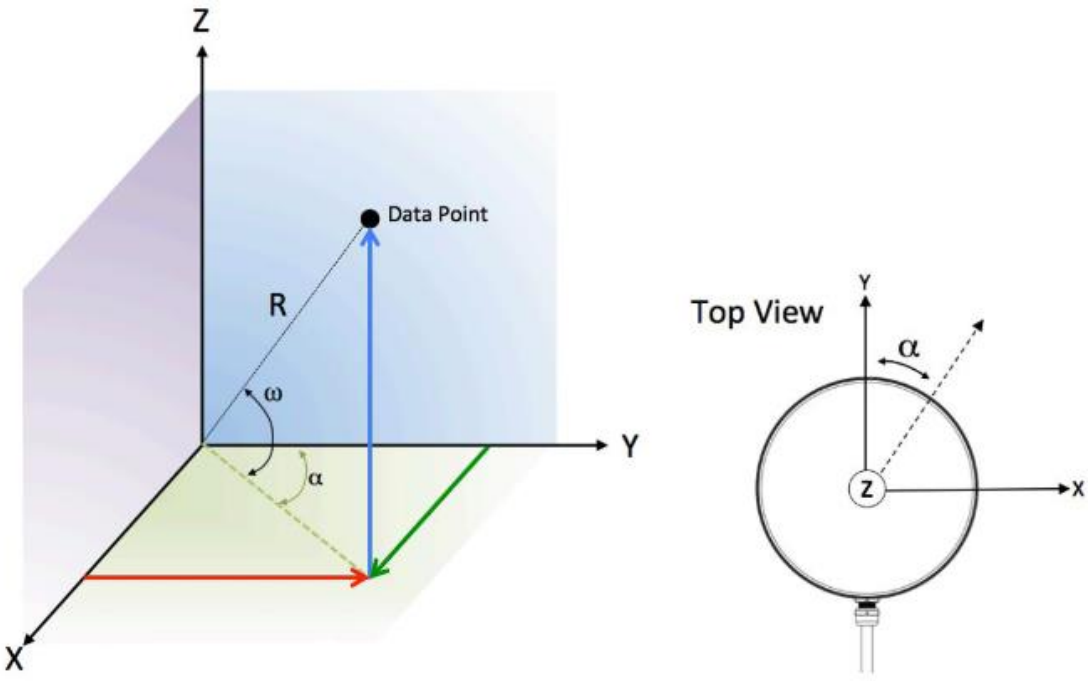

Figura 61. Mapa de coordenadas [49].

La empresa fabricante Robosense ofrece junto con el LiDAR diferentes paquetes *software* para comunicarse con el dispositivo. Entre ellos destaca el paquete ROS, llamado rslidar_sdk, desplegado en este proyecto. Este paquete proporciona la transformación de los datos, obteniendo una visualización de la nube de puntos en la herramienta RVIZ del sistema ROS, como se mostrará más adelante en el apartado 5.2.

4.1.2 PROTOCOLOS DE COMUNICACIÓN

El ordenador se comunica a través de Ethernet con el LiDAR a una velocidad de 100Mbps y el protocolo de comunicación utilizado es UDP. Además, existen dos tipos de protocolos cuya información va encapsulada en segmentos UDP: paquetes MSOP y paquetes DIFOP. Cada paquete UDP tiene una longitud de 1290 bytes siendo 42 bytes para la cabecera y 1248 bytes para el *payload*. Los identificadores correspondientes a la dirección IP y puerto vienen configurados de fábrica y se muestran en la Figura 62.

	IP Address	MSOP Port No.	DIFOP Port No.
RS-LiDAR-16	192.168.1.200	6699	7788
Computer	192.168.1.102		

Figura 62. Puerto y direcciones IP del LiDAR y PC [47].

La comunicación RS-LIDAR-16 está basada en tres tipos de protocolos de comunicaciones para establecer la conexión con el ordenador:

- **MSOP** (*Main Data Stream Output Protocol*): es el protocolo fundamental que envía los datos que se procesan en el LiDAR, es decir, distancia, acimut y reflectividad se envían al ordenador.
- **DIFOP** (*Device Information Output Protocol*): se encarga de monitorizar la configuración actual del sensor.
- **UCWP** (*User Configuration Write Protocol*): es el protocolo encargado de modificar algunos parámetros según sea necesario.

Protocol	Abbreviation	Function	Type	Size	Interval
Main Data Stream Output Protocol	MSOP	Scan Data Output	UDP	1248byte	~1.33 ms
Device Information Output Protocol	DIFOP	Device Information Output	UDP	1248byte	~100 ms
User Configuration Write Protocol	UCWP	Sensor Parameters Setting	UDP	1248byte	INF

Figura 63. Protocolos de RS-LIDAR-16 [47].

A continuación se describe con mayor profundidad estos protocolos y sus mensajes asociados.

4.1.2.1 MSOP

MSOP es el protocolo esencial en cuyos mensajes se encuentran los datos en bruto que transmite el dispositivo LiDAR al ordenador. Cada paquete MSOP tiene una longitud de 1248 bytes que se divide en tres bloques: 42 bytes de cabecera, 1200 bytes de *payload* y 6 bytes de cola, Figura 64.

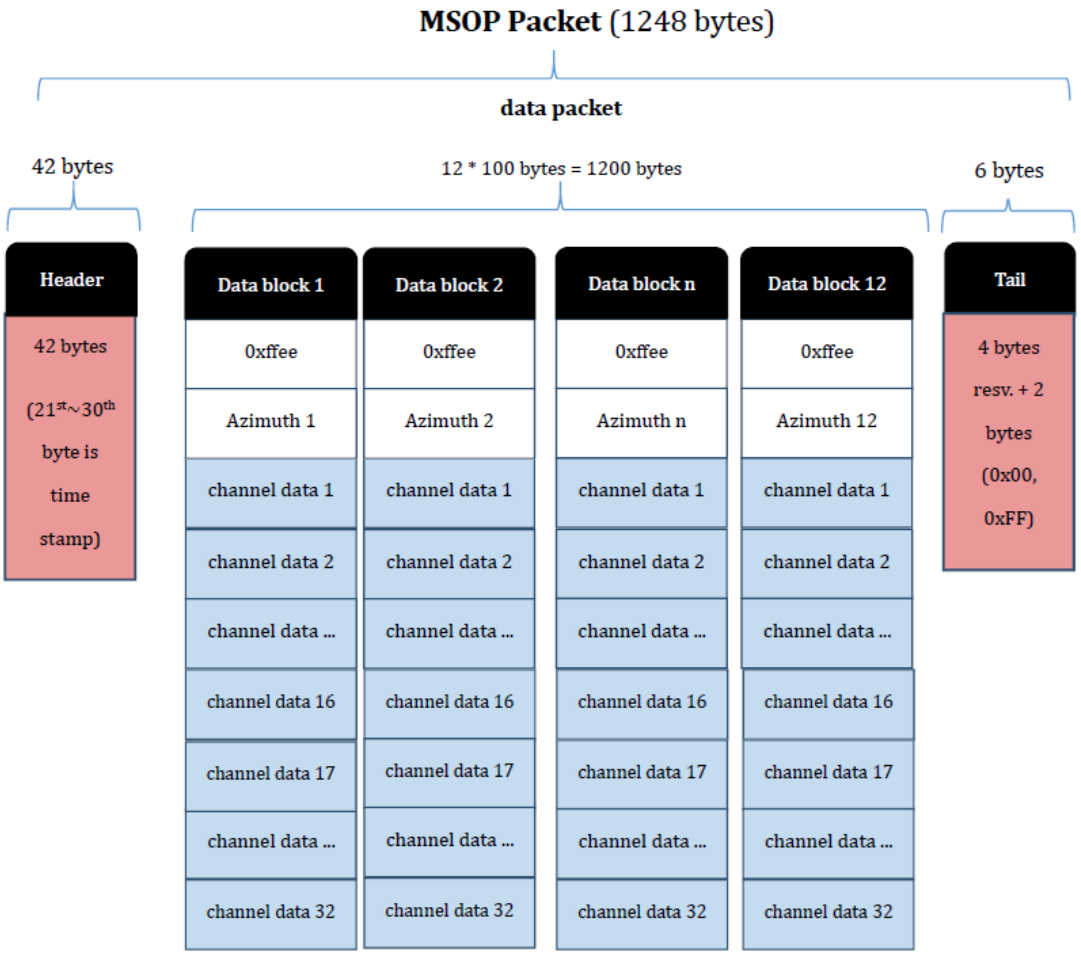

Figura 64. Paquete MSOP RS-LIDAR-16 [47].

Los 42 bytes de cabecera marcan el comienzo del paquete. Los primeros 8 bytes son la identificación del paquete y son siempre iguales

```
0x55 0xAA 0x05 0x0A 0x5A 0xA5 0x50 0xA0
```

En cambio, del byte 21 al 30 se define el *timestamp*, en el byte 31 se representa el modelo de LiDAR (Figura 65) y los bytes restantes están reservados para nuevas actualizaciones.

LiDAR Model (1 byte)	
0x01	RS-LiDAR-16
0x02	RS-LiDAR-32

Figura 65. Tabla de asignación de valor al byte 31 según modelo de LiDAR [47].

Los 1200 bytes de *payload* corresponde a los datos en bruto, es decir, se define la distancia, reflectividad y acimut. El *payload* está dividido en un total de 12 bloques de datos. Cada bloque tiene una longitud de 100 bytes. Respecto a los 12 bloques de datos, cada bloque de datos empieza con 2 bytes de identificación, 0xFFEE, después, 2 bytes para el acimut y 96 bytes que se dividide en 32 canales con 3 bytes cada canal donde se proporciona los datos de la distancia y reflectividad. Cada acimut graba 32 canales de

datos que corresponde con dos secuencias de 16 canales, como se muestra en Figura 66. El segundo conjunto de datos se corresponde con el siguiente acimut de cuyo ángulo no se informa. Por lo tanto si el acimut 1 indica el ángulo 0, y el acimut del bloque 2 indica el ángulo 0,4º, eso significa que los datos de los canales 17 a 32 del primer bloque de datos corresponden al acimut 0,2º.

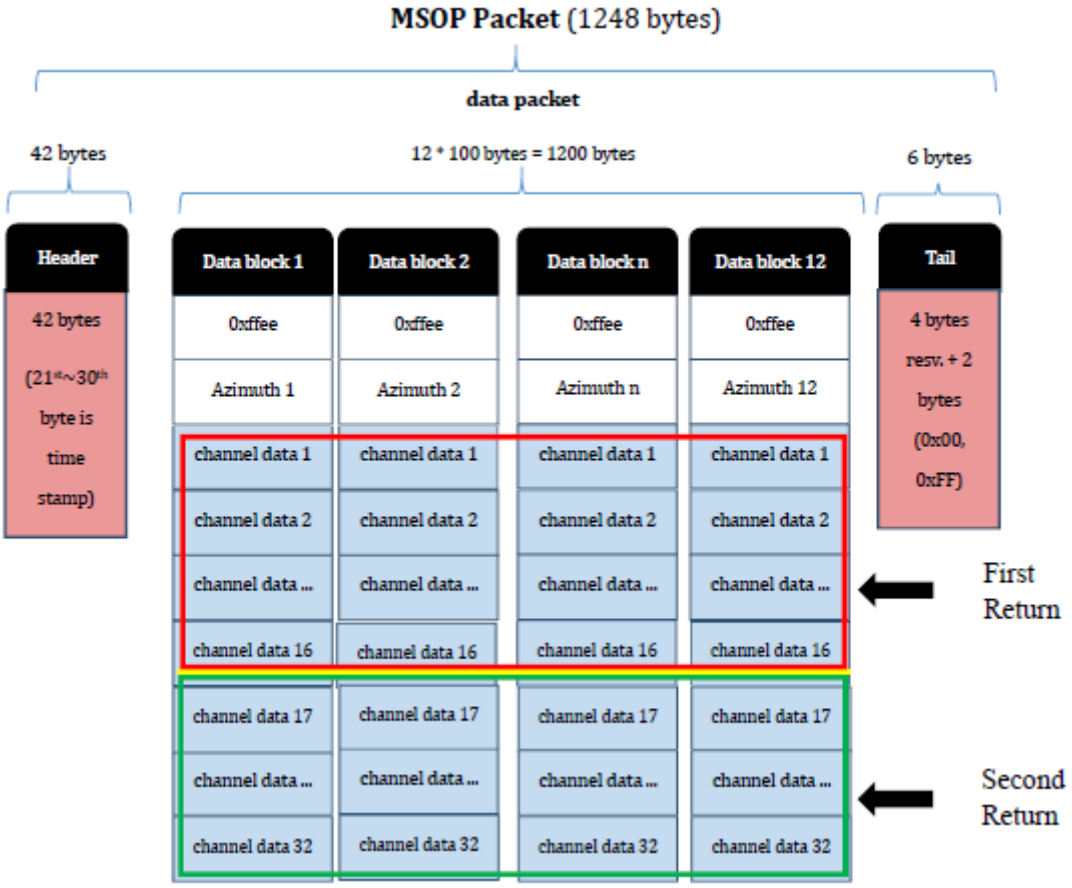

Figura 66. Dos secuencias de 16 canales MSOP RS-LIDAR-16 [47].

Cada canal de datos contiene 3 bytes, y está formado por 2 bytes para la distancia medida en centímetros y 1 byte para el valor de la reflectividad, tal y como se muestra en la Figura 67.

Channel Data N (3 bytes)		
2 bytes Distance		1 byte Reflectivity
Distance1 [15:8]	Distance2 [7:0]	Reflectivity

Figura 67. Canal de datos MSOP [47].

Por tanto, en cada bloque de 100 bytes hay dos secuencias de 16 datos. Estos datos corresponden a los 16 canales respectivamente. Los valores obtenidos en cada bloque pertenecen a un acimut en concreto. El acimut depende del ángulo de resolución fijado (0.1º o 0.4º), y este a su vez depende de la frecuencia de rotación del LiDAR, es decir, el rango del valor de acimut es de 0º a 360º pudiendo obtener la información del bloque de datos cada 0.1º o cada 0.4º, por ejemplo: si se tiene un ángulo de resolución con un valor

de 0.2º, el primer bloque de datos empezaría con un valor de acimut de 0º y el siguiente bloque con 0.4º (recordar que por cada bloque se envían 2 acimuts), como son 12 bloques, el bloque 12 del paquete MSOP le corresponderá un valor de 4.4º. En la Figura 68, se puede observar un segmento UDP capturado en *Wireshark*, donde se han hecho las comprobaciones necesarias para entender el funcionamiento de la transmisión de datos.

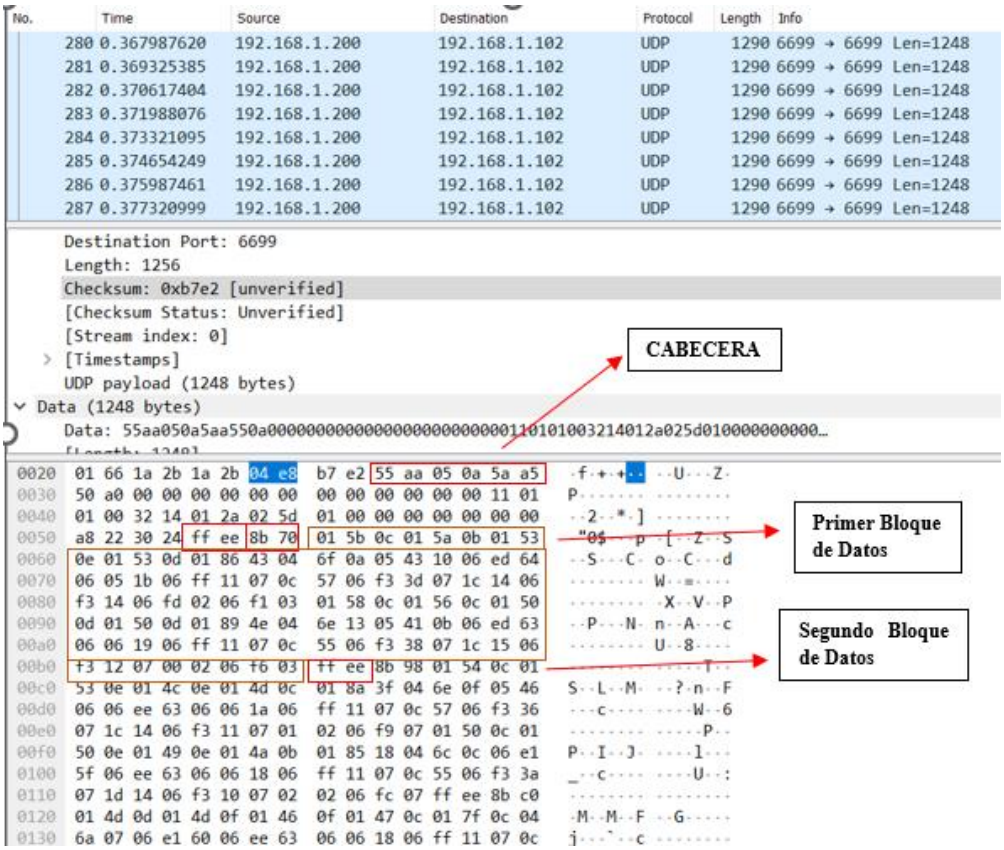

Figura 68. Representación paquete MSOP.

4.2 MARVELMIND INDOOR "GPS"

Marvelmind Indoor "GPS" es un sistema de navegación en interiores diseñado para proporcionar datos de localización con un margen de error de ±2 cm a robots y vehículos autónomos. También puede utilizarse para seguir objetos en movimiento mediante balizas móviles conectadas a ellos.

El sistema de navegación consiste en una red de balizas ultrasónicas fijas interconectadas mediante una interfaz de radio en una banda sin licencia, una o varias balizas móviles instaladas en objetos a seguir y un módem que sirve de puerta de enlace con el sistema desde un ordenador.

La localización de las balizas móviles se calcula en función del retardo de propagación de los impulsos ultrasónicos (*Time-Of-Flight* o TOF) entre las balizas fijas y las móviles.

La empresa Marvelmind ofrece este *hardware* junto con un *software* que puede construir el mapa de balizas estacionarias automáticamente. En casos sencillos, no es necesario introducir datos manuales adicionales ni realizar mediciones manuales de la distancia.

Figura 69. Kit Marvelmind Indoor "GPS".

En la Figura 69 se muestran las especificaciones técnicas más relevantes del kit de sensores de ultrasonidos. Entre las características más interesantes para nuestra aplicación podemos destacar el alcance de 50 metros. Este alcance junto con el hecho de que es posible crear una red de sensores hasta abarcar un área total de 1000 metros cuadrados lo hace ideal para nuestra aplicación, que debe estar pensada para interiores donde el GPS no puede penetrar. Además, frente a otras soluciones como las basadas en UWB (*Ultra-Wide Band*), tiene una tasa de refresco muy alta, permitiendo actualizar la localización de los vehículos autónomos con frecuencia suficiente para los algoritmos de guiado.

Distancia entre balizas	• Alcanza distancias hasta 50 metros • La distancia recomendada es de 30 metros
Área de cobertura	• Alcanza hasta 1000 m2 con las configuraciones de inicio del kit • Cobertura para territorios grandes es similar a las redes móviles
Precisión de localización	• Absoluta: 1-3 % de la distancia a las balizas • Precisión diferencial: ± 2 cm
Tasa de actualización de la localización	• 0.05 – 25 Hz • Puede ser configurado manualmente • Depende de la distancia entre las balizas móvil y estacionarias (distancias cortas – más alta la tasa de actualización) • Depende del número de balizas móviles: por ejemplo, 25 Hz de tasa de actualización para una baliza móvil, 25/2 Hz de tasa de actualización para dos balizas móviles, etc. • Depende ligeramente del número de balizas fijas

Tabla 3. Especificaciones técnicas más relevantes del kit de ultrasonidos de Marvelmind.

La configuración de *Marvelmind Indoor "GPS"* se basa en 4 balizas estáticas, 1 baliza móvil y un modem, pueden distinguirse dos tipos de arquitecturas: *Non Inverse*

Architecture (NIA) e *Inverse Architecture* (IA). En la arquitectura NIA la baliza móvil se encarga de emitir ultrasonidos y las balizas estáticas reciben esa información. En cambio, en la arquitectura IA funciona al revés, la baliza móvil recibe los ultrasonidos que están emitiendo las balizas estáticas. A continuación, se van a describir las diferentes balizas y el modem del kit de *Marvelmind Indoor "GPS":*

- **Baliza estacionaria**: emiten y reciben ultrasonidos durante la configuración del mapa. Una vez el mapa se congela, con la arquitectura no-inversa las balizas estacionarias solo funcionan como receptores. Normalmente se montan en las paredes o techos por encima del robot con los sensores ultrasónicos orientados hacia abajo, para proporcionar la cobertura de la señal ultrasónica más robusta y sin obstrucciones al robot.
- **Baliza móvil**: se encuentran situadas en el vehículo para obtener su ubicación en tiempo real. Se recomienda colocar la baliza móvil en posición horizontal para obtener mayor cobertura ultrasónica. No existe diferencia *hardware* entre la baliza estacionaria y la móvil.
- **Modem**: es el que se encarga del control central del sistema, el modem siempre tiene que estar alimentado cuando se están utilizando las balizas. El modem también se utiliza para la configuración de las balizas, gracias a la aplicación "Dashboard", desde dicha aplicación puedes monitorizar los datos y configurar la baliza móvil.

En nuestro caso la arquitectura óptima es el modelo NIA, ya que la arquitectura IA es más compleja y está diseñada para poder trabajar con más de una baliza móvil al mismo tiempo, esto requiere de más balizas estáticas emitiendo ultrasonidos. De momento, en nuestro proyecto sólo nos centraremos en tener una baliza móvil, con el objetivo de obtener el posicionamiento de un vehículo dentro de un parking cerrado. Estos sensores se deciden utilizar para obtener una posición más precisa, fusionando los datos obtenidos de la baliza móvil con los datos del dispositivo LiDAR. Para obtener está fusión de datos, la empresa ofrece un paquete desarrollado en ROS que consiste en publicar varios tópicos con la información de la posición y la distancia a la que se encuentran las balizas estáticas de la baliza móvil, se explica con más profundidad en el apartado 5.5.

5 *Software* Desplegado

En esta sección se va a describir el *software* desplegado, es decir, el objetivo es explicar la comunicación del LiDAR con el PC para poder analizar los datos que nos proporciona el LiDAR y la técnica utilizada para hallar la posición del dispositivo. Por otro lado, se encuentran los sensores de ultrasonidos los cuales proporcionan la posición y la distancia a la que está de cada uno de los sensores. El capítulo se ordena de la siguiente manera. Primero se hace un repaso a las técnicas de SLAM existentes, junto con las características de nuestra aplicación y de los sensores elegidos y se hace una selección de la técnica SLAM a utilizar. En las secciones siguientes se explican los pasos seguidos para poder obtener el posicionamiento del vehículo, teniendo que desplegar diferentes paquetes de ROS y explicando los inconvenientes encontrados.

5.1 Selección de Técnica SLAM

Antes de comenzar con la selección de la técnica SLAM merece la pena exponer brevemente las características principales de la aplicación que queremos desarrollar. Se trata de un vehículo que será autónomo de nivel 4 dentro de recintos cerrados que se utilizarán como aparcamientos automáticos. El vehículo debe recibir órdenes sobre en qué lugar aparcar exactamente, y ahora mismo se utiliza un sistema de tarjetas inalámbricas para que el vehículo sepa cuando está en un sitio del parking concreto. El guiado se realiza mediante una línea pintada en el suelo, o también por línea magnética.

El objetivo principal de este proyecto era integrar dos nuevos sensores a este prototipo, un LiDAR y, como apoyo al LiDAR, un localizador *indoor*. Estos dos nuevos dispositivos nos permiten aplicar la tecnología SLAM a nuestro problema, o bien una modificación de la misma. El LiDAR permite descubrir mapas automáticamente. Esto es muy conveniente para nuestra aplicación porque nos permite descubrir la disposición de los vehículos dentro del parking y crear un mapa en tres dimensiones. Por otra parte, el sensor de posicionamiento *indoor* nos da una gran precisión, muy superior a la que es necesaria para este tipo de aplicaciones (alrededor de dos centímetros cuando normalmente para vehículos autónomos se trabaja con precisiones de hasta 10 cm). Estas dos características permiten aplicar la técnica SLAM. El sistema funcionaría de la siguiente manera:

1. Primero deberíamos tener un mapa creado con el LiDAR de la zona por donde se va a mover el vehículo. Este mapa localizaría solo los elementos estáticos del recinto. Además, se generaría un sistema de coordenadas coordinado con el sistema de coordenadas del sistema de localización *indoor*, para que la localización calculada con ambos sistemas sea compatible.
2. Una vez se han integrado los sensores en el vehículo, que como hemos visto en la introducción del capítulo requiere varios pasos, la información es alimentada a la técnica SLAM y con ella se descubren los objetos que no son permanentes en el mapa y ayuda a describir una ruta a seguir dentro del mapa teniendo en cuenta dichos obstáculos.

Las decisiones que hay que tomar en este momento son las siguientes:

1. Forma de comunicación de los sensores con el resto del sistema.
2. Tipo de *software* SLAM a utilizar.

Como habíamos adelantado en el capítulo 2, en cuanto al sistema de comunicación de los sensores con el resto del sistema se seleccionó ROS. Este paradigma está orientado a sistema robóticos y es ampliamente utilizado también en vehículos autónomos. Además, nuestros sensores ya cuentan con *software* desarrollado por el propio fabricante que permite la comunicación en ROS.

Además, existían dos versiones de ROS. En este trabajo se ha utilizado ROS1. ROS1 tiene la ventaja de ser una versión mucho más estable y que tiene muchas más librerías desarrolladas. ROS2 por su parte ofrece mejoras en el campo de la comunicación, pero tiene, a fecha de hoy, un menor soporte. Efectivamente, el fabricante del LiDAR ofrece librerías para ROS1 y para ROS2, pero mientras para la primera librería fue posible extraer los datos en bruto e interpretarlos gracias a la wiki de ROS1 [53], con ROS2 no se fue capaz de lograrlo. Leer los datos en bruto resultó ser esencial para el trabajo fin de grado, dado que los paquetes requerían una transformación de los datos que sin la interpretación de los datos en bruto para crear el programa de transformación nos resultaban imposible de utilizar.

Es importante señalar que en cualquier caso sí se produjo algún avance con ROS2. Por ejemplo, fue posible transmitir los datos con un tópico, y por supuesto recibirlos suscribiéndose a dicho tópico. Lo que no resulto posible fue decodificar los datos recibidos. En cualquier caso, hay que tener en cuenta que ROS1 quedará obsoleto en algunos años y por lo tanto habrá que pensar en migrar de uno a otro.

El segundo punto que nos interesa es la técnica SLAM a utilizar. Por supuesto la técnica SLAM que queramos utilizar debe ser compatible con ROS1, razón por la que limitamos nuestro muestreo a las tres técnicas ya mencionadas. Por otro lado, ya se describieron algunas ventajas y desventajas de estas tres técnicas. Pero para nuestro caso nos interesa especialmente las siguientes características:

- Que no se necesiten datos GPS, es decir, que la técnica no se alimente con estos datos.
- Que no se requieran datos en 2D, es decir, que el método no requiera de un LiDAR 2D.
- Que la técnica sólo necesite un tópico con información del entorno en 3D.

La única técnica que cumple estas características es la técnica LOAM ya que sólo necesita suscribirse a un tópico que tenga información sobre una nube de puntos. El resultado de esta técnica es ofrecer la estimación de la posición del vehículo y el mapeado del entorno.

Elegida la técnica SLAM solo queda conectar todo el sistema para poder realizar una localización del vehículo en movimiento dentro de un mapa. Para ello se conectan los siguientes paquetes de ROS:

1. Paquete rslidar_sdk: su función es transformar los datos del LiDAR obteniendo una nube de puntos sobre el entorno. Se crea un nodo (/rslidar_sdk_node) y un tópico (/rslidar_points)
2. Paquete rs_to_velodyne: consiste en convertir el formato del fabricante Robosense a Velodyne. Se cambian el tópico de /rslidar_points a /velodyne_points.
3. Paquete LOAM: proporciona los datos sobre la odometría y el mapeado del entorno en tiempo real. Este paquete se suscribe al tópico /velodyne_points.
4. Paquete marvelmind: se despliega este paquete ROS para obtener el posicionamiento de la baliza móvil y a la distancia que se encuentra de las balizas estáticas.
5. Paquete AMCL3D: su tarea es obtener una localización más precisa dentro de un entorno conocido. Este paquete se suscribe a la información proporcionada por el paquete LOAM y el paquete marvelmind.

En la Figura 70, se puede observar un esquema gráfico sobre los diferentes paquetes descritos, para obtener una visión clara sobre el despliegue.

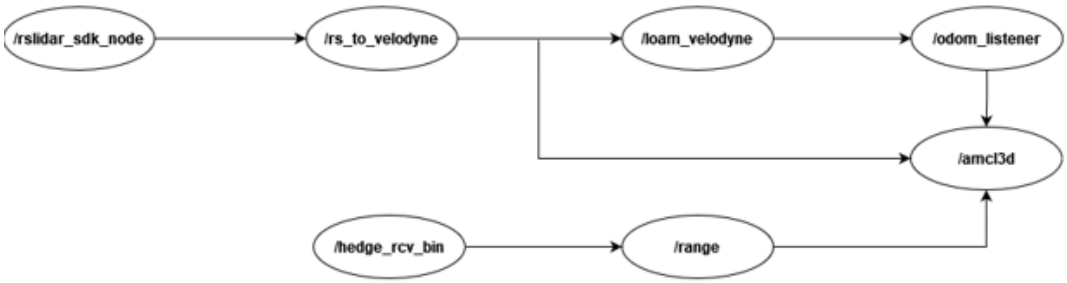

Figura 70. Paquetes desplegados.

Para aclarar el concepto que muestran las figuras donde están esquematizados los nodos y los tópicos de cada paquete, lo que está en forma de elipse corresponde a los nodos y en forma rectangular a los tópicos, Figura 71.

Figura 71. Leyenda tópico y nodo.

5.2 PAQUETE RS-LIDAR

El fabricante del RS-LiDAR-16 proporciona un paquete que consiste en la transformación de los datos en bruto que llegan al ordenador. El paquete se encarga de transformarlos a un sistema de coordenadas cartesianas como ya se ha comentado en el apartado 4.1.

Para instalar este paquete, llamado rslidar_sdk, primero se debe clonar desde su git [54]. Este paquete hay que compilarlo con el comando de ROS (*catkin_make*), aunque antes

hay que hacer una serie de modificaciones en el código fuente dependiendo del tipo de LiDAR o del mensaje que se quiere publicar:

- En el fichero de configuración /catkin_ws/src/rslidar_sdk/config/config.yaml, se tiene que modificar el parámetro *lidar_type* en función del tipo de LiDAR, en nuestro caso el parámetro corresponde con RS16 ya que se trabaja con RS-LIDAR-16. La modificación de ese parámetro se puede ver en la Figura 72.

```
lidar_type: RS16            #LiDAR type - RS16, RS32, RSBP, RS128
```

Figura 72. Definir tipo de LiDAR

- En el mismo fichero de configuración, el parámetro que define el mensaje fuente (*msg_source*) tiene que estar a uno ya que los datos vienen directamente desde el LiDAR y no desde otro paquete ROS o desde otro fichero.

```
msg_source: 1                #0: not use Lidar
                             #1: packet message comes from online Lidar
                             #2: packet message comes from ROS or ROS2
                             #3: packet message comes from Pcap file
```

Figura 73. Definir tipo de mensaje RS-LIDAR-16.

Después de haber compilado el paquete, este paquete se encarga de crear un nodo llamado /rslidar_sdk_node. Una vez creado el nodo que proporciona la transformación de los datos, se publica un tópico llamado /rslidar_points (Figura 74), la estructura de datos de este mensaje es definida como sensor_msgs/PointCloud2. La función de este mensaje es poser representar una nube de puntos, Figura 75.

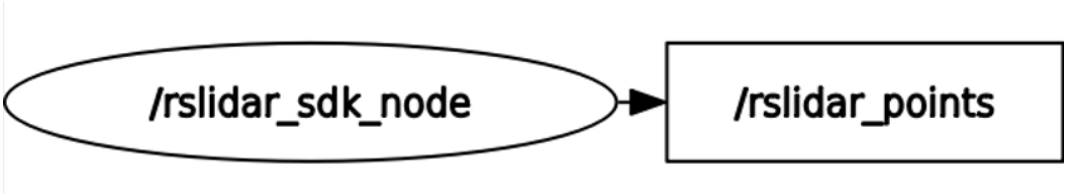

Figura 74. Nodo y tópico RS-LIDAR-16.

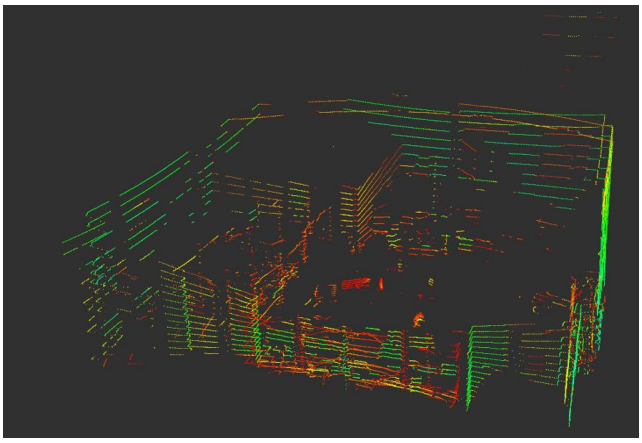

Figura 75. Nube de puntos RS-LIDAR-16.

El formato PointCloud2 consiste en proporcionar una colección de puntos con N-dimensiones, que pueden contener información adicional. La cabera almacena el instante en el que se reciben los datos y el identificador de las coordenadas. Después, se define la

altura y el ancho de la nube de puntos. En el PointField se describen los canales con su identificador respectivo. Se define cómo es el tipo de dato *big endian* o no. Luego, se fijan los valores sobre la longitud del punto en bytes y la longitud de la fila. Por último, se mandan todos los valores en bruto. En la Figura 76 se puede observar una estructura de datos sobre el formato PointCloud2.

```
std_msgs/Header header
uint32 height
uint32 width
sensor_msgs/PointField[] fields
bool is_bigendian
uint32 point_step
uint32 row_step
uint8[] data
bool is_dense
```

Figura 76. Estructura de datos sensor_msgs/PointCloud2.

Este es un formato estándar sobre el mensaje, en cambio, lo que varía es el formato del tópico que es un formato creado por el fabricante del LiDAR, Robosense y este no es un formato estándar. Por lo tanto, los paquetes SLAM no son capaces de consumir de manera directa estos datos y hay que hacer una transformación. Esta transformación la realiza el paquete que se explica en la siguiente sección y su salida podrá alimentar directamente el paquete LOAM.

5.3 CONVERSIÓN RS-LIDAR A VELODYNE

El funcionamiento de este paquete es convertir el tópico de la nube de puntos de Robosense (/rslidar_points) en formato de nube de puntos de Velodyne (/velodyne_points).

Velodyne es otro fabricante de dispositivos LiDAR. Estos tipos de LiDAR se han convertido en un estándar de facto y sus formatos son utilizados en muchos paquetes de código abierto. Por eso, hace falta cambiar el tópico para no tener que modificar el código fuente. La estructura de datos es igual para ambos formatos, excepto que el formato del tópico Velodyne añade una variable para especificar el número de canal que obtiene la información correspondiente al punto.

Para obtener la conversión, se clona el paquete en el mismo espacio de trabajo que rslidar_sdk. El paquete llamado rs_to_velodyne se suscribe al tópico /rslidar_points creando un nuevo nodo, rs_converter. El nodo se encarga de hacer la conversión y publicar el tópico /velodyne_points. Por lo tanto, se ha generado un sistema en tubería que realiza la transformación de los datos para que pueda ser consumido por el paquete LOAM, tal y como se puede ver en la Figura 77.

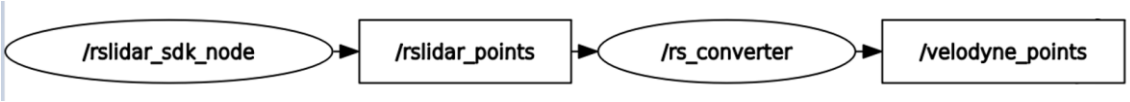

Figura 77. Conversión /rslidar_points a /velodyne_points.

El tópico /velodyne_points tiene que mostrar lo mismo que /rslidar_points. Las dos nubes de puntos se muestran en RVIZ, si se observa la Figura 78 se comprueba que se obtiene la misma visualización y que, por tanto, la conversión del formato ha sido correcta.

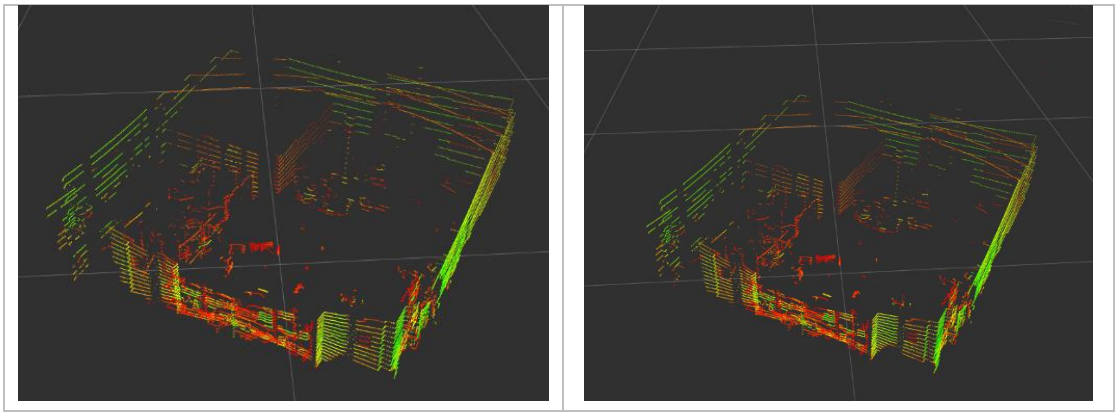

Figura 78. Resultado de la transformación de una nube de puntos de Robosense a Velodyne.

Esta conversión se ha llevado a cabo para poder alimentar el paquete LOAM como se ha comentado anteriormente, la mayoría de las técnicas de SLAM han sido desarrolladas con el formato Velodyne. Por tanto, en lugar de modificar el código fuente de estas técnicas se decidió convertir el formato para facilitar su despliegue.

5.4 Paquete LOAM

Existen variedad de técnicas para obtener localización y mapeado simultáneamente o técnicas SLAM. Sin embargo, gracias al estudio realizado sobre las diferentes técnicas que se explican en el apartado anterior se ha llegado a la conclusión de que la mejor técnica para desplegar en este caso es LOAM ya que este método utiliza exclusivamente la nube de puntos que calcula el LiDAR, por tanto, cumple las condiciones para utilizar en este proyecto.

La técnica LOAM se estructura en cuatro procesos distintos, Figura 79, que se describen a continuación:

- *PointCloudRegistration*: es el primer proceso que se ejecuta tras suscribirse a la nube de puntos. La función de este proceso es registrar la nube de puntos e identificar los bordes y las superficies planas del entorno.
- *LidarOdometry*: se encarga de calcular la odometría, es decir, realiza la estimación de la posición del vehículo con los datos obtenidos del proceso anterior, *PointCloudRegistration*. El resultado de este proceso se obtiene de comparar dos nubes de puntos en instantes distintos, obteniendo la posición y orientación del sensor.

- *LidarMapping*: su funcionamiento es el mismo que *LidarOdometry*, aunque en este la frecuencia es menor a cambio de obtener una precisión más elevada del cálculo de la odometría.
- *TransformIntegration*: su objetivo es fusionar tanto la información de *LidarOdometry* como *LidarMapping* para obtener una estimación de la posición precisa.

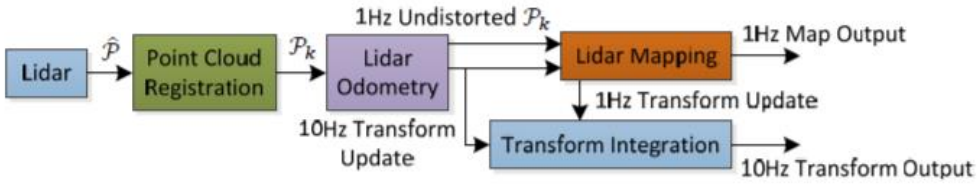

Figura 79. Diagrama de bloques del sistema software, LOAM [38].

Al realizar el despliegue de este paquete se encontraron varios inconvenientes. El primero fue que este paquete sólo había sido testeado con Velodyne LiDAR como ya se ha mencionado anteriormente y, por tanto, se tuvo que hacer la conversión de formato ya que en nuestro caso se utiliza el LiDAR de Robosense. Después, al hacer la compilación se mostraban errores por la versión de C++, lo que llevó a modificar el CMakeLists.txt del paquete. Para finalizar, una vez conseguida la compilación completa del espacio de trabajo, se lanzó el paquete LOAM, pero en la herramienta de visualización (RVIZ) no se mostraba nada. El motivo de esto era que el parámetro *Fixed frame* no estaba correctamente definido en el fichero de configuración de RVIZ (loam_velodyne.rviz). *Fixed frame* debe fijarse a /camera_init.

Tras resolver todos los inconvenientes se desplegó el paquete obteniendo dos procesos ejecutándose en paralelo. Por un lado, se obtiene la odometría del sensor, es decir, se publica en un mismo mensaje la posición y la orientación del LiDAR y se muestra en RVIZ mediante flechas, tal y como se ve en la Figura 80.

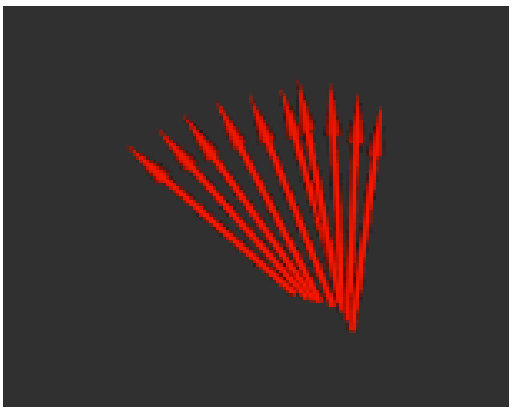

Figura 80. Visión del tópico odometría LOAM.

Por otro lado, se obtiene la nube de puntos que se va almacenando con el objetivo de crear un mapa del entorno por donde se va moviendo el LiDAR. En la Figura 81, se muestra el mapa que se ha obtenido del laboratorio de la universidad, donde se ha estado trabajando en este proyecto.

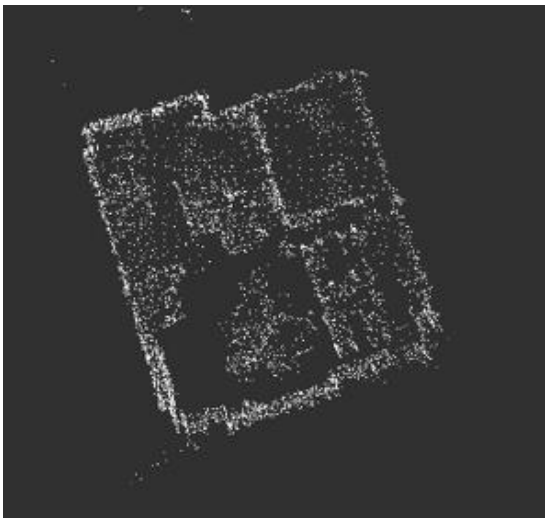

Figura 81. Mapa grabado del entorno.

En la Figura 82, se puede observar la nube de puntos y la odometría en tiempo real. La diferencia entre la Figura 81 y esta es que el mapa almacena todos los puntos que recorre el LiDAR, en cambio, la nube de puntos de la Figura 82 es la que se visualiza en tiempo real.

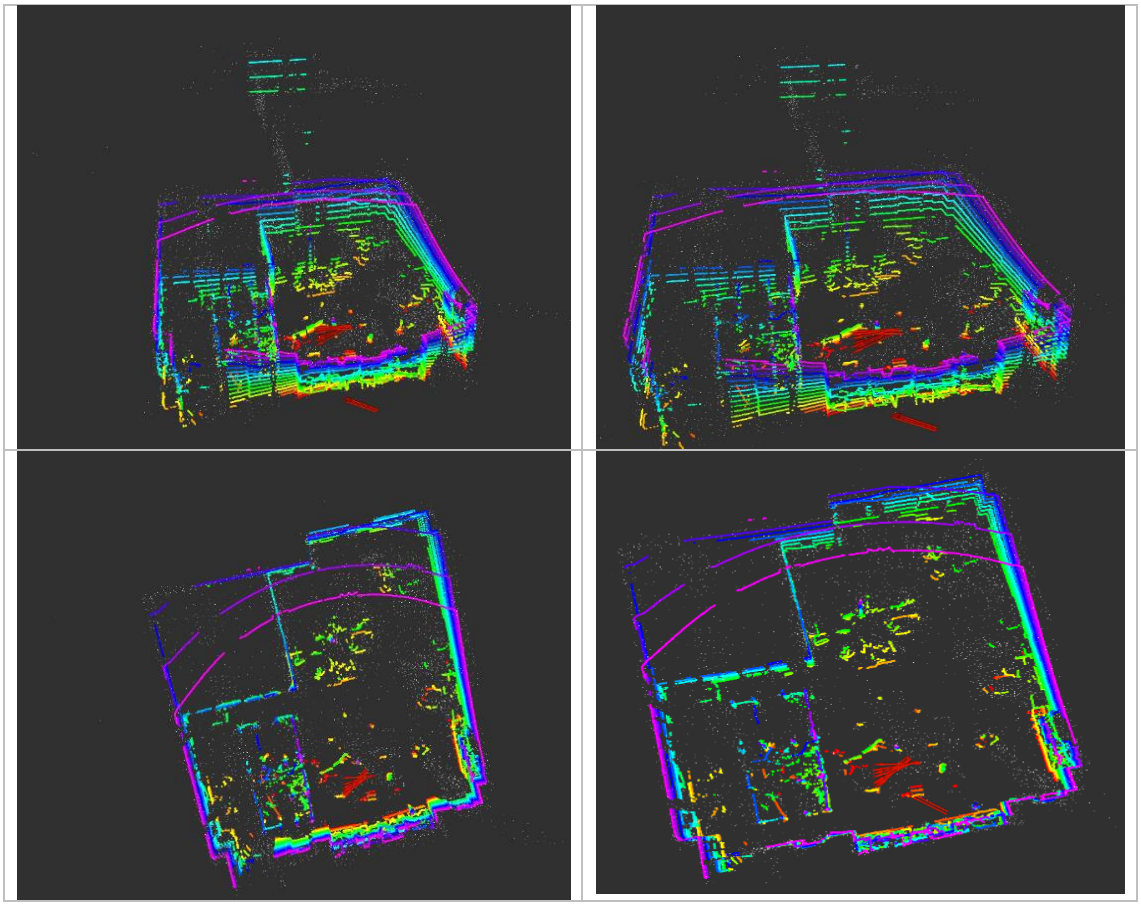

Figura 82. Diferentes perspectivas de visión de la técnica LOAM.

5.5 PAQUETE MARVELMIND

El motivo de trabajar con estos sensores al mismo tiempo que con el LiDAR es debido a la necesidad de una localización más precisa que la proporcionada por la técnica LOAM. En este caso, el objetivo es desplegar el paquete de ROS que proporciona el fabricante de los sensores *Marvelmind Indoor "GPS"* para obtener una posición redundante del vehículo y así alimentar al método amcl3d que se describirá en la siguiente sección, antes de clonar el paquete en el espacio de trabajo donde se encuentran los paquetes anteriores, hay que seguir unos pasos previos.

Primero, instalar la aplicación *Dashboard*, mostrada en Figura 83, en otro ordenador. Este ordenador debe conectarse al Modem, para que esté alimentado y conseguir mostrar los datos de las balizas móvil y estacionarias en la aplicación.

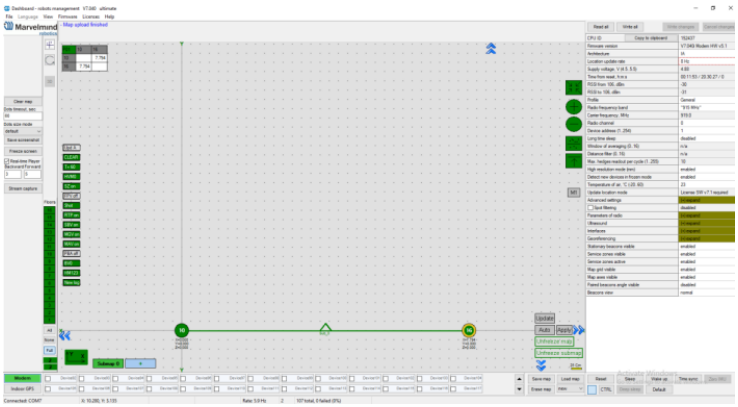

Figura 83. Aplicación Dashboard.

Segundo, se colocan cuatro balizas estacionarias (*stationary beacon*) formando aproximadamente un cuadrado. Desde la aplicación Dashboard, se observa el mapa creado y se configura la quinta baliza como baliza móvil. En la Figura 84, observamos dos imágenes. Por un lado, en la imagen de la izquierda se muestran las cuatro balizas estacionarias. Por otro lado, en la imagen de la derecha ya se ha activado la quinta baliza que corresponde con el número ocho, donde se puede observar mediante la línea rosa el recorrido que ha hecho en ese mismo instante.

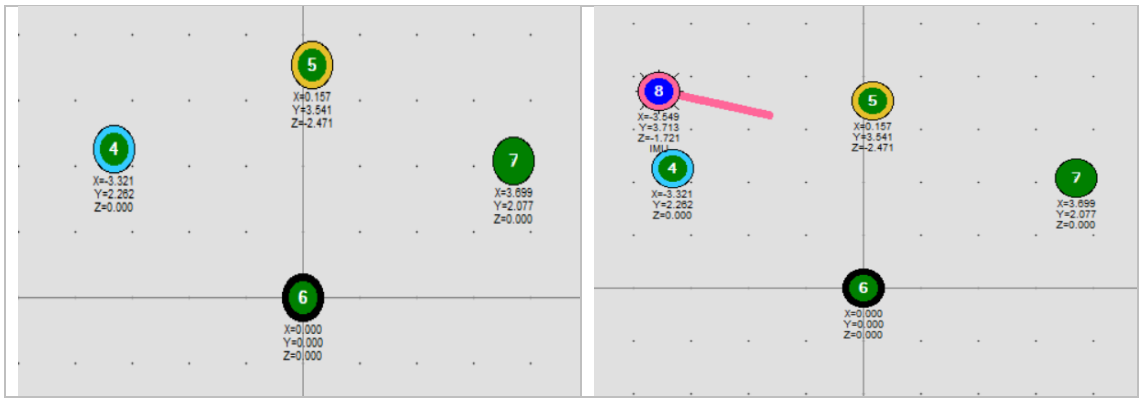

Figura 84. Representación de las balizas en el Dashboard.

Por último, se conecta la baliza móvil al ordenador donde se quieren obtener los datos de la posición mediante ROS, está baliza móvil tendrá que ir situada en el vehículo. El paquete ROS de marvelmind publica variedad de tópicos. Sin embargo, los tópicos fundamentales para nuestro caso son /beacon_pos_a y /beacon_distance. El paquete crea un nodo llamado /hedge_rcv_bin que publica diferentes tópicos, pero que en este caso nos quedamos solo con dos, como se puede observar en la Figura 85. En esa misma figura también se muestra otro nodo (/range) y otro tópico (/range_pose) que se han creado para alimentar al siguiente paquete y se describirá en el apartado 5.6.

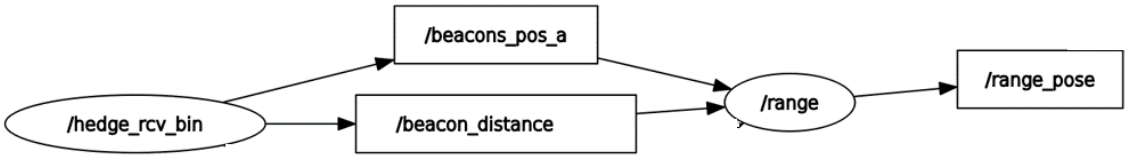

Figura 85. Nodos y tópicos balizas estacionarias y baliza móvil.

beacon_pos_a	address	uint8	Address of stationary beacon
	x_m	float64	X coordinate, meters
	y_m	float64	Y coordinate, meters
	z_m	float64	Z coordinate, meters
beacon_distance	address_hedge	uint8	Address of mobile beacon
	address_beacon	uint8	Address of stationary beacon
	distance_m	float64	Raw distance from mobile to stationary beacon, meters

Figura 86. Tópicos fundamentales Marvelmind.

Los tópicos vistos en la Figura 86 proporcionan información sobre la posición de cada baliza estacionaria (*/beacon_pos_a*) y la distancia correspondiente a la baliza móvil (*/beacon_distance*). Aunque también existe otro tópico que proporciona la información sobre la posición de la baliza móvil, el siguiente paquete, AMCL3D, no necesita este valor ya que con la información de los tópicos (*/beacon_pos_a* y */beacon_distance*) se encarga de calcular la posición de la baliza móvil. Se puede obtener una representación de las balizas en la herramienta de visualización, RVIZ. En la Figura 87, se puede observar que los marcadores en verde representan las balizas estáticas, en cambio, el marcador azul es la baliza móvil.

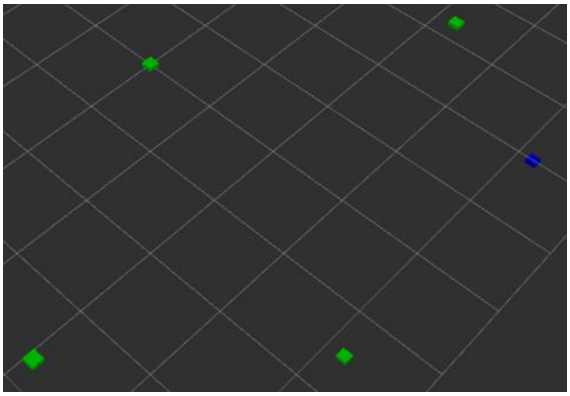

Figura 87. Representación balizas en RVIZ.

Debido a tener una localización del vehículo más segura y saber encontrarse en un mapa conocido, se decidió estudiar la técnica AMCL3D que se describe en el siguiente apartado.

5.6 PAQUETE AMCL3D

El método de localización de Monte Carlo adaptativo en 3D (*Adaptive Monte Carlo Localization*, AMCL3D) consiste en estimar la posición del vehículo o robot en un entorno 3D sin el uso de GPS. Este método utiliza sensores de distancia, LiDAR y un mapa conocido del entorno para calcular la posición. El algoritmo está alimentado por una odometría, una nube de puntos y la distancia entre los diferentes sensores que hay en el entorno. Se encarga de comparar el mapa conocido con la nube de puntos, teniendo en cuenta la distancia a los sensores se calcula una aproximación sobre la posición. La media de las diferentes partículas entre el cálculo hallado y la odometría dan el resultado de la posición y la orientación.

El objetivo de utilizar este paquete es obtener una localización dentro de un entorno conocido, lo que nos permite obtener una trayectoria sobre la ruta realizada. El funcionamiento está basado en el algoritmo de Monte Carlo general, el cual está divido en tres pasos: predicción, actualización y remuestreo, pero la versión de AMCL3D añade un paso más: inicialización. Esta inicialización consiste en fijar una posición del vehículo con respecto al mapa guardado para empezar a ejecutar el algoritmo [55].

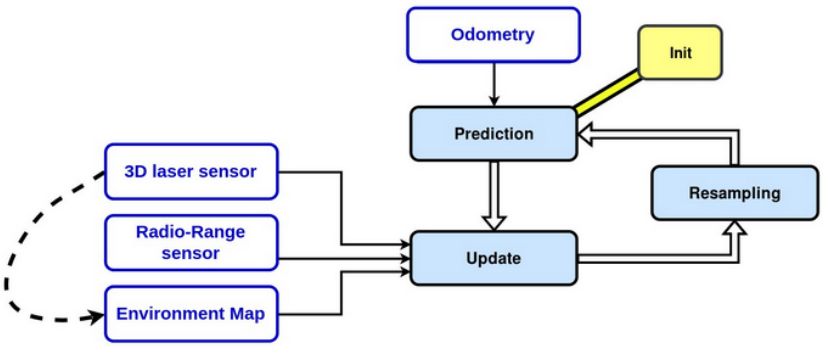

Figura 88. Funcionamiento del método AMCL3D.

La localización de Monte Carlo consiste en un filtro de partículas de localización, el cual representa una distribución probabilística de la posición del vehículo dotando de pesos a las partículas y, por tanto, estimando el punto exacto de la localización. En la **¡Error! No se encuentra el origen de la referencia.**, se observa que el algoritmo recibe la nube de puntos como un mapa del entorno, donde se va a mover el vehículo, y este algoritmo consigue localizarse gracias a la predicción del movimiento con la ayuda de los datos sobre la odometría que proporciona el paquete LOAM.

Para el despliegue de este paquete en nuestro sistema hay que conectarlo con los paquetes anteriores. En la Figura 89 se muestran los nodos fundamentales para llevar a cabo este

proyecto con sus respectivos tópicos. Aunque hay más nodos y tópicos los principales que dan el resultado requerido son los aquí representados.

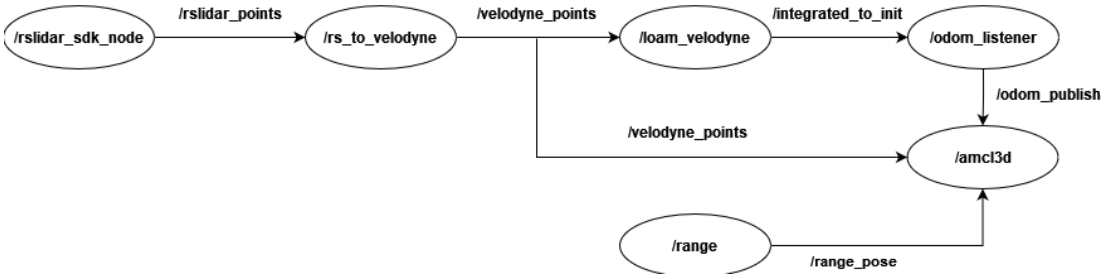

Figura 89. Principales nodos y tópicos de todos los paquetes desplegados.

Los tópicos que necesita el método AMCL3D para funcionar correctamente son los siguientes:

- **Nube de puntos (/velodyne_points):** el algoritmo se suscribe directamente al tópico de la nube de puntos donde le transmite la información sobre la visión del entorno en tiempo real.

```
<!-- To set the topic which has information about the point cloud from the vehicle view -->
<remap from="laser_sensor" to="/velodyne_points"/>
```

Figura 90.AMCL3D se suscribe al tópico /velodyne_points.

- **Odometría (/odom_publish):** en este caso, se ha tenido que crear un *script* ya que la estructura del mensaje de odometría necesaria para este paquete no es la misma estructura que el que nos proporciona la técnica LOAM. Se crea por lo tanto un nodo /odom_listener (Figura 89). Este nodo está suscrito al tópico de odometría de la técnica LOAM (/integrated_to_init), cuyo mensaje es con formato nav_msgs/Odometry.msg, y publica un tópico llamado /odom_publish que lleva la misma información que /integrated_to_init pero con formato geometry_msgs/TransformStamped.msg. La diferencia entre ambos mensajes es que el mensaje que proporciona la técnica de LOAM contiene información irrelevante para AMCL3D, en concreto no necesita ni el valor de covarianza ni la velocidad. Por ello, se tiene que hacer un *script* que tome los datos necesarios del mensaje nav_msgs/Odometry.msg y los vuelque en un mensaje con el formato de geometry_msgs/TransformStamped.msg.

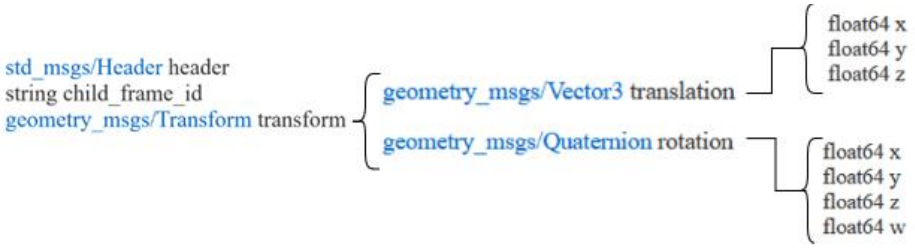

Figura 91. Estructura geometry_msgs/TransformStamped.msg [56].

En la Figura 92, se muestra la estructura del mensaje **nav_msgs/Odometry.msg**, dentro del cual la único información relevante para nosotros se encuentra contenida

a su vez en el mensaje **geometry_msgs/PoseWithCovariance,** el cual consta de todos los datos que necesita la estructura de **geometry_msgs/TransformStamped.msg**

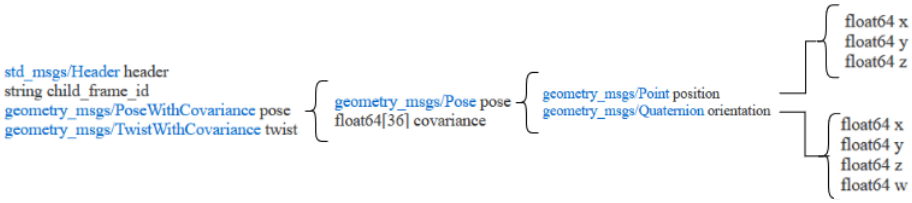

Figura 92. Estructura nav_msgs/Odometry.msg [57].

```
<!-- To set the topic which has information about the vehicle odometry -->
<remap from="odometry" to="/odom_publish"/>
```

Figura 93. AMCL3D se suscribe al tópico /odom_publish.

- **Distancia y posición de los sensores (/range_pose):** para obtener estos datos se ha generado un *script* donde se crea una estructura de mensaje nueva obteniendo información de los sensores de ultrasonidos (*Marvelmind Indoor "GPS"*). Esta estructura consta de la distancia a la que se encuentra la baliza móvil de cada baliza estática y también de la posición donde se encuentra cada baliza estática situada. Los dos primeros parámetros del tópico creado son los identificadores de la baliza móvil y la baliza estática, los otros dos siguientes parámetros corresponden a la posición de la baliza estática y a la distancia que se encuentran respectivamente la baliza estática y la baliza móvil, Figura 95.

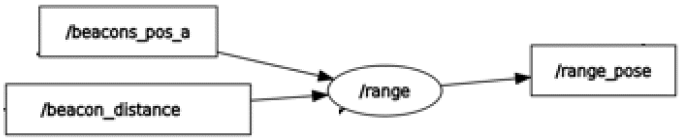

Figura 94. Tópicos y nodo distancia sensores.

La tarea del tópico /range_pose es publicar información que es imprescindible para que el método AMCL3D funcione.

```
uint64 source_id
uint64 destination_id
float64 range
geometry_msgs/Point position
```

Figura 95. Estructura datos /range_pose.

Una vez se tiene creado el nuevo tópico que alimenta al método AMCL3D, se modifica el fichero para que se suscriba a dicho tópico con el nombre correspondiente, Figura 96.

```
<!-- To set the topic which has information about the sensor-range sensors -->
<remap from="radiorange_sensor" to="/range_pose"/>
```

Figura 96. AMCL3D se suscribe al tópico /range_pose.

Se han descrito los tres tópicos fundamentales para alimentar al método de AMCL3D, pero falta por añadir el fichero del mapa. Recordemos que este mapa es necesario para conocer el entorno por donde se mueve el vehículo, y se hace una comparativa sobre la nube de puntos en tiempo real y el mapa con el objetivo de conocer en que parte del mapa nos encontramos.

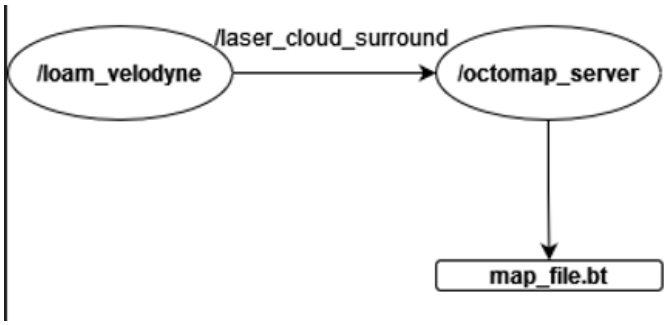

Figura 97. Estructura para crear el fichero que contiene el mapa.

El mapa se construye partiendo de la salida de la técnica LOAM, la cual consta de un tópico que genera una nube de puntos sobre el recorrido realizado con LOAM, el tópico se llama /laser_cloud_surround, se puede observar en la Figura 97. Para crear este mapa se ha utilizado otro paquete de ROS llamado octomap_server, cuyo cometido es suscribirse a un tópico que tenga información de una nube de puntos, y a través de esos datos crear un fichero con extensión .bt (*Binary Tree*) que es el mapa que alimenta al paquete AMCL3D. Se ha modificado el paquete octomap_server para suscribirse al tópico necesario, como se puede ver en la Figura 98.

```
<!-- data source to integrate (PointCloud2) -->
<remap from="cloud_in" to="/laser_cloud_surround" />
```

Figura 98. Octomap_server se suscribe a /laser_cloud_surround.

Para la comprobación y visualización del mapa, se utiliza la herramienta PCD viewer. Esta herramienta trabaja con un formato de fichero concreto denominado PointCloundData, con extensión .pcd. Por tanto, para poder visualizar nuestro mapa con esta herramienta ha sido necesario pasarlo de formato *Binary Tree* (map_file.bt) a PCD (*PointCloudData*) con el siguiente comando:

```
$ octree2pointcloud ~/catkin_ws/mapfile.bt cloud.pcd

$ pcd_viewer cloud.pcd
```

Al ejecutar el comando pcl_viewer, se abre una herramienta que muestra el mapa, se puede observar en la Figura 99.

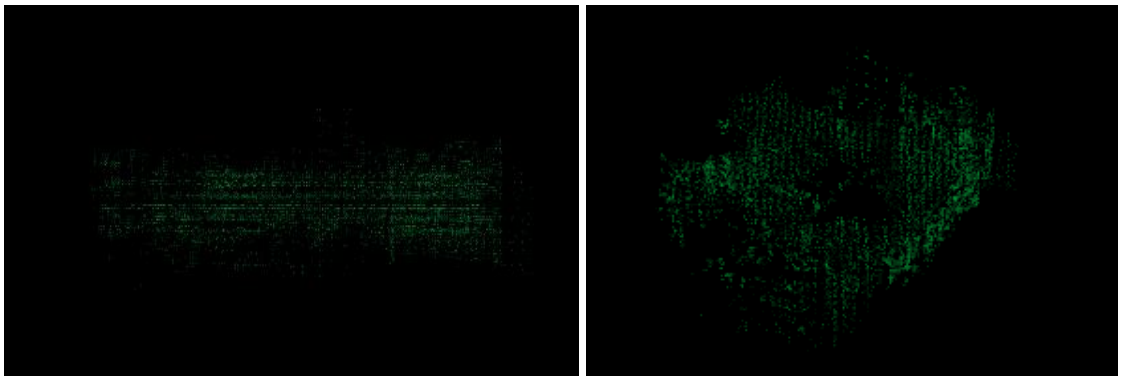

Figura 99. Comprobación del mapa, herramienta PCD viewer .

En conclusión, al punto que se ha conseguido llegar es el de alimentar al paquete AMCL3D con: el mapa, encargado de obtener un entorno conocido; nube de puntos en tiempo real, se hace una comparación con el mapa; odometría, estimación de la posición del dispositivo LiDAR y distancia de las balizas, se hace una comparación con la odometría, sin embargo, no se ha llegado a obtener el resultado deseado debido a que no se ha terminado de desplegar el paquete AMCL3D. Entre los problemas que plantea el despliegue actual, uno de los más importantes es que no se han trasladado las diferentes informaciones de localización a un mismo sistema de coordenadas, esencial para que el sistema AMCL3D trabaje de forma correcta.

6 Conclusión

A lo largo de este proyecto se han ido cumpliendo diferentes objetivos, uno de ellos descrito en la sección 1.2 es el diseño de una nueva arquitectura en el vehículo Twizy que se ha desarrollado en el apartado 3. Se han ido integrando diferentes tecnologías en la arquitectura electrónica del vehículo para obtener una mejor eficiencia en cuanto al funcionamiento del vehículo y seguir evolucionando para conseguir el objetivo de obtener un vehículo autónomo de nivel 4.

El objetivo principal de este Trabajo de Fin de Grado es el guiado del vehículo autónomo mediante tecnología LiDAR. Teniendo en cuenta el contexto del proyecto TwizyLine en el que se basa este trabajo, es decir, aparcamientos autónomos que pueden estar situados tanto en interior como en exterior, se decide utilizar para el guiado la técnica SLAM. Con este objetivo, se han analizado variedad de técnicas SLAM, aunque en este caso, nos hemos centrado en la técnica LOAM ya que no se requiere de datos GPS, y sólo son necesarios los datos de la nube de puntos que proporciona el LiDAR.

La técnica LOAM se encarga de la estimación de la posición del vehículo y de generar un mapeado del entorno del vehículo. Sin embargo, estas operaciones por si solas, no son suficientes para lograr realizar la tarea de guiado de un vehículo autónomo, ya que se requiere de una localización más precisa para un trabajo en tiempo real.

Para lograr nuestro objetivo se concluye que es necesario utilizar el método AMCL3D, dado que se obtiene una mayor precisión sobre la posición en un mapa conocido. Para ello, se hace un estudio sobre qué datos son necesarios en el algoritmo AMCL3D. Entre la información necesaria para este algoritmo se encuentra los datos y el mapa del entorno que ya se han obtenido de la técnica LOAM, aunque el mapa hay que crearlo en un fichero para conseguir un entorno concreto, y la información relativa a la posición del vehículo obtenida a través de las balizas de ultrasonidos mencionadas en este trabajo. También se necesita información sobre la nube de puntos en tiempo real para poder realizar una comparativa con el mapa proporcionado.

Aunque no se han obtenido los resultados esperados debido a que no se ha conseguido el despliegue total de la técnica AMCL3D, se ha llegado a proporcionar al algoritmo de todas las entradas requeridas: mapa, nube de puntos, odometría y distancia entre las balizas. En el uso de este paquete *software* pueden todavía aparecer varias cuestiones que se deban resolver, pero la principal de ellas es coordinar todos los sistemas de referencias de los diferentes sistemas de localización. Una vez se obtenga el despliegue del paquete, será relativamente sencillo utilizar los datos generados para calcular la posición del vehículo con respecto a una ruta a seguir y alimentar el miniordenador de control con los datos que necesita para seguir la ruta: ángulo y *offset* del vehículo con respecto a la ruta.

En este trabajo se han adquirido conocimientos que no se habían estudiado a lo largo de la carrera, como el uso de un dispositivo LiDAR, las técnicas de SLAM, o el funcionamiento del sistema ROS. La comunicación por ROS es un pilar muy importante

en este trabajo debido a que ha facilitado la comunicación entre múltiples máquinas, y seguirá siendo importante para seguir con el desarrollo del prototipo de vehículo autónomo. Al mismo tiempo, que he estado desarrollando este trabajo también he realizado prácticas en Renault y, entre las diferentes tareas asignadas una de ellas era tener conocimientos de ROS para trabajar con un prototipo de vehículo autónomo que tienen en la fábrica.

6.1 Líneas futuras

La principal línea futura es terminar el despliegue del paquete AMCL3D, con el que conseguimos una localización del vehículo en un entorno conocido. Gracias a esta localización, se puede obtener una trayectoria de la ruta realizada y conseguir una navegación fluida dentro del entorno. A continuación, se podría seguir una serie de objetivos que son desarrollados a continuación:

- El primer objetivo a cumplir después de conseguir el despliegue es realizar un estudio sobre donde va a ir integrado el dispositivo LiDAR en el vehículo.
- Implementar la comunicación con el módulo de control, es decir, enviarle los datos obtenidos gracias a las técnicas estudiadas en este trabajo y que el vehículo dependiendo de lo datos tome decisiones.
- A parte del LiDAR, se va a integrar una cámara como se ha definido en el apartado 3. El objetivo es que se lleguen a fusionar ambos datos para obtener una percepción completa del entorno.
- Detección de obstáculos dentro del entorno, ya sean obstáculos estáticos o dinámicos. Con la fusión de datos sobre los diferentes sensores podemos obtener un estudio sobre los diferentes métodos para la detección de obstáculos.

También cabe destacar que el prototipo que se quiere utilizar como sistema de demostración para diversas actividades de la Escuela. Por lo tanto, se propone como línea futura transmitir la información de los sensores a un servidor exterior que sea capaz de visualizar lo datos y por lo tanto crear demostraciones de cómo se utiliza esta tecnología en nuestro sector.

7 REFERENCIAS

[1] Groupe Renault & Segula Technologies, «Twizy Contest,» 2019. [En línea]. Available: https://www.twizycontest.com/.

[2] A. Mazaira Hernández, “Front-end implementation for an automatized car parking,” Valladolid, 2020.

[3] I. Royuela, «Four level autonomous vehicle for an automatized parking,» Valladolid, 2020.

[4] M. Martín, «Plan de Comunicación de TwizyLine plataforma de carshring con aparcamiento autónomo.,» Valladolid, 2021.

[5] S. Pilar Arnanz, “Back-end implementation for an automatized car parking,” Valladolid, 2020.

[6] Dirección General de Tráfico, «Siniestros 2021,» Enero 2022. [En línea]. Available: https://revista.dgt.es/es/noticias/nacional/2022/01ENERO/0107-Balance-prov-accidentalidad-2021.shtml. [Último acceso: Mayo 2022].

[7] Agencia Andaluza de la Energía, «Manual de conducción eficiente,» Instituto para la Diversificación y Ahorra de la Energía (IDAE), Madrid.

[8] Dirección General de Tráfico, «Conducción eficiente,» [En línea]. Available: https://www.dgt.es/muevete-con-seguridad/conviertete-en-un-buen-conductor/conduccion-eficiente/. [Último acceso: 03 Junio 2022].

[9] Dirección General de Tráfico, «Las ADAS obligatorias en 2022, una a una,» 18 Marzo 2022. [En línea]. Available: https://revista.dgt.es/es/motor/tecnologia-seguridad/2021/0518-Landing-ADAS.shtml. [Último acceso: 10 Junio 2022].

S. G. Matínez, «"Detección de objetos mediante LIDAR 3D para aplicaciones en robótica",» Alcalá, 2017.

[11] ZOTAC, «Serie ZBOX E,» [En línea]. Available: https://www.zotac.com/es/product/mini_pcs/magnus-en173080c-barebone-0. [Último acceso: 04 Junio 2022].

[12] Robosense, «RS-LIDAR-16,» [En línea]. Available: https://www.robosense.ai/en/rslidar/RS-LiDAR-16. [Último acceso: 2022 Junio 02].

[13] Marvelmind, «Marvelmind Indoor "GPS",» [En línea]. Available: https://marvelmind.com/. [Último acceso: 2022 Junio 03].

[14] ENDARK, «Porta Fusibles Coche,» [En línea]. Available: https://www.amazon.es/Fusibles-ENDARK-Indicador-Negativo-furgoneta/dp/B08BZFN8FV. [Último acceso: 03 Junio 2022].

[15] Jorge Gutierrez Moreno, «Implementación de vehículo autónomo en entorno simulado,» Universidad Carlos III , Madrid.

[16] SAE International, «Surface vehicle recommended practice,» 2021.

[17] General Motors Company, «Self-driving safety report,» 2018.

[18] J. Vargas, S. Alsweiss, O. Toker, R. R y J. Santos, «An overview of autonomous vehicles sensors and their vulnerability to weather conditions,» Sensors, 2021.

[19] 20MINUTOS, «Clasificación coches autónomos,» 2018. [En línea]. Available: https://www.20minutos.es/noticia/2825372/0/clasificiacion-coches-autonomos/. [Último acceso: 03 Junio 2022].

[20] NeoMotor, «Mercedes estrena la conducción de nivel 3,» 8 Mayo 2022. [En línea]. Available: 2022. [Último acceso: Junio 10 2022].

[21] Marcos Bureau, «Especial Seguridad Vial: sistemas de alerta de colisión,» [En línea]. Available: https://www.motorpasion.com.mx/seguridad/especial-seguridad-sistemas-de-alerta-de-colision.

[22] Diario Motor, «Alerta por cambio involuntario de carril: el ángel de la guarda que evita que te salgas de tu carril,» [En línea]. Available: https://www.diariomotor.com/2014/12/11/alerta-cambio-involuntario-carril/. [Último acceso: 03 Junio 2022].

[23] Maserati, «Control de crucero adaptativo con stop & go,» [En línea]. Available: https://www.maserati.com/es/es/servicios-posventa/seguridad/control-de-crucero-adaptativo. [Último acceso: 03 Junio 2022].

[24] Dirección General de Tráfico, «Sistemas de frenado de emergencia autónomos,» Abril 2016. [En línea]. Available: https://revista-org.dgt.es/es/motor/tecnologia-seguridad/2016/0419-Tecnologia-n235-sist-frenado-autonomo.shtml. [Último acceso: 04 Junio 2022].

[25] Autocity, «Coches con sistema de estacionamiento asistido,» 2016. [En línea]. Available: https://autocity.com/reportajes/los-coches-con-sistemas-de-estacionamiento-asistido-a-debate. [Último acceso: 2022 Junio 09].

[26] C. Weitkamp, Lidar, Range-Resolved Optical Remote Sensing of the Atmosphere, Springer, 2005.

[27] ArcGIS Desktop, «Was ist LiDAR?,» [En línea]. Available: https://desktop.arcgis.com/de/arcmap/latest/manage-data/las-dataset/what-is-lidar-data-.htm. [Último acceso: 03 Junio 2022].

[28] N. Mehendale, «Review on LiDAR technology,» Mumbai.

[29] Generationrobots, «What is LiDAR technology?,» [En línea]. Available: https://www.generationrobots.com/blog/en/what-is-lidar-technology/. [Último acceso: 14 Junio 2022].

[30] Clearpath Robotics, «How to choose the right LiDAR sensor for your project,» 2021. [En línea]. Available: https://www.wevolver.com/article/how-to-choose-the-right-lidar-sensor-for-your-project. [Último acceso: 08 Junio 2022].

[31] M. Khader y C. Cherian, «An Introduction to Automotive LIDAR,» Texas Instruments, 2020.

[32] MathWorks, «What is MATLAB?,» [En línea]. Available: https://www.mathworks.com/discovery/what-is-matlab.html. [Último acceso: 08 Junio 2022].

[33] MathWorks, «Lidar Toolbox,» [En línea]. Available: https://www.mathworks.com/help/lidar/index.html?s_tid=srchtitle_Lidar%20Toolbox_1. [Último acceso: 08 Junio 2022].

[34] ROS, «ROS - Robot Operating System,» [En línea]. Available: https://www.ros.org/. [Último acceso: 2022 Junio 09].

[35] ROS Tutorial, «Fundamentos de ROS,» [En línea]. Available: http://rostutorial.com/3-roscore-y-workspace/. [Último acceso: 22 Junio 2022].

[36] ROS, «Distributions,» [En línea]. Available: http://wiki.ros.org/Distributions. [Último acceso: 14 Junio 2022].

[37] H. Durrant y T. Bailey, «Simultaneous Localization and Mapping: Part I,» IEEE, 2006.

[38] A. Escalante, «Análisis del método de odometría lidar LOAM,» Sevilla, 2016.

[39] Departament of Computer Science and Mechanical Engineering, «Hector Open Source Modules for Autonomous Mapping and Navigation with Rescue Robots,» TU Darmstadt.

[40] J. Zhang y S. Singh, «LOAM: Lidar Odometry and Mappinf in Real-time,» Robotics: Science and Systems, 2014.

[41] Universidade Federal de Minas Gerais, «Investiga ̧c ̃ao de T ́ecnicas LiDAR SLAM para um Dispositivo Robótico de Inspeção de Ambientes Confinados,» Minas Gerais, 2020.

[42] R. Ribalda, «Localización y mapeado simultáneos para vehículos autónomos utilizando Lidar 3D,» Universidad de Alcalá, 2020.

[43] S. Pilar Arnanz, «Servicios de vehículo autónomo y conducción autónoma.,» Universidad de Valladolid, 2021.

[44] ASTI Mobile Robotics, «A30 ASEL.061 V2,» 2019.

[45] MaxBotix, «MB1020 LV-MaxSonar-EZ2 Datasheet,» 2021.

[46] Maxon Motor Ibérica S.A.U., «EPOS: Command Library Documentation,» 2021.

[47] MAXON, «MAXON flat motor,» [En línea]. Available: https://www.maxongroup.es/medias/sys_master/root/8831018893342/2018EN-270.pdf. [Último acceso: 14 Junio 2022].

[48] Solid Run, «Humming Board CBi,» [En línea]. Available: https://developer.solid-run.com/knowledge-base/hummingboard-cbi-getting-started/. [Último acceso: 20 Junio 2022].

[49] Robosense, «RS-LiDAR-16 User's Manual».

[50] Intel, «Intel RealSense Product Family D400 Series Datasheet,» 2022.

[51] Kvaser, «CAN Bus Connectors,» [En línea]. Available: https://www.kvaser.com/about-can/the-can-protocol/can-connectors/. [Último acceso: 16 Junio 2022].

[52] Texas Intruments, «INA226 Datasheet,» 2015.

[53] Robot Operating System, «Wiki ROS,» [En línea]. Available: http://wiki.ros.org/es. [Último acceso: 22 Junio 2022].

[54] Github, «rslidar_sdk,» [En línea]. Available: https://github.com/RoboSense-LiDAR/rslidar_sdk. [Último acceso: 22 Junio 2022].

[55] F. Perez-Grau, F. Caballero, A.Viguria, y A. Ollero, «Multi-sensor three-dimensional Monte Carlo localization for long-term aerial robot navigation,» International Journal of Advanced Robotic Systems, 2017.

[56] ROS, «geometry_msgs/TransformStamped.msg,» [En línea]. Available: http://docs.ros.org/en/api/geometry_msgs/html/msg/TransformStamped.html. [Último acceso: 22 Junio 2022].

[57] ROS, «nav_msgs/Odometry.msg,» [En línea]. Available: http://docs.ros.org/en/noetic/api/nav_msgs/html/msg/Odometry.html. [Último acceso: 22 Junio 2022].

ANEXO I

En este anexo se muestran todos los tópicos utilizados para desplegar los paquetes de ROS utilizados en este trabajo.

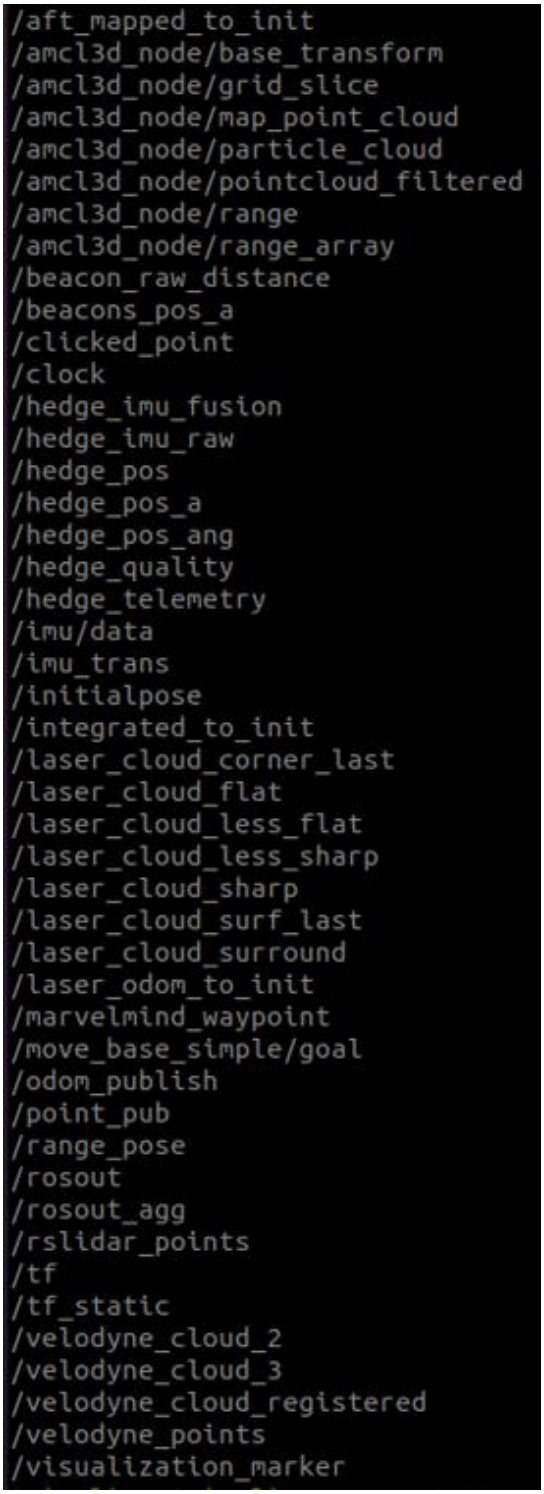

Figura 100. Lista de tópicos.

Diagrama de todos los paquetes desplegado, generado con la herramienta de ROS, rqt_graph:

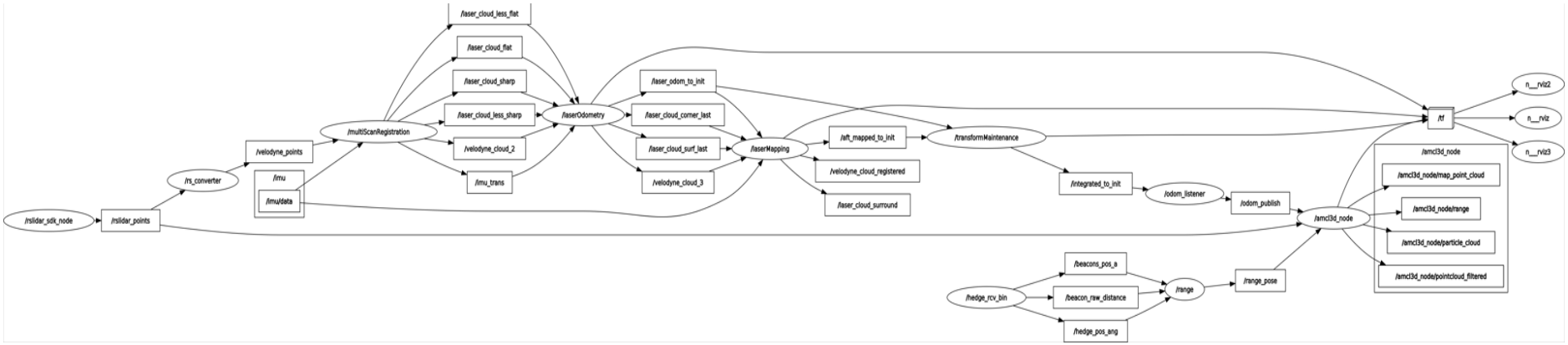

Figura 101. Diagrama de paquete desplegados.