



UNIVERSIDAD
DE MÁLAGA



ESCUELA DE INGENIERÍAS INDUSTRIALES

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

ÁREA DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

TRABAJO FIN DE GRADO

DESARROLLO DE UNA PLATAFORMA SENSORIAL
MULTIMODAL RGB - TÉRMICO - LIDAR 3D PARA
UN VEHÍCULO AUTÓNOMO

Grado en

INGENIERÍA ELECTRÓNICA, ROBÓTICA Y
MECATRÓNICA

AUTOR: MANUEL RICARDO DIOSA USMA

TUTOR: JESÚS MORALES RODRÍGUEZ

COTUTOR: JORGE MONTENEGRO NAVARRO

MÁLAGA, JUNIO DE 2025

DESARROLLO DE UNA PLATAFORMA SENSORIAL MULTIMODAL RGB - TÉRMICO - LiDAR 3D PARA UN VEHÍCULO AUTÓNOMO

Autor: Manuel Ricardo Diosa Usma.

Tutor: Jesús Morales Rodríguez.

Cotutor: Jorge Montenegro Navarro.

Departamento: Ingeniería de Sistemas y Automática.

Titulación: Grado en Ingeniería Electrónica, Robótica y Mecatrónica.

Palabras clave: Conducción autónoma, plataforma multisensorial, cámaras térmicas, cámaras estéreo, sensor LiDAR 3D, ROS2, SolidWorks, vehículo sensorizado.

Resumen

Este Trabajo de Fin de Grado presenta el diseño y desarrollo de una plataforma sensorial multimodal para vehículos autónomos, orientada a la mejora de la percepción del entorno. El sistema integra cuatro cámaras térmicas, útiles en condiciones de baja visibilidad; tres cámaras estéreo, que proporcionan visión en profundidad; y un sensor LiDAR 3D, que genera nubes de puntos precisas mediante pulsos láser. La integración software de los sensores se ha realizado utilizando ROS2, facilitando la adquisición, sincronización y gestión de los datos. Además, se ha diseñado mediante herramientas CAD la base de montaje para los sensores y equipos de computación, así como los soportes necesarios para su instalación en el vehículo.

**DEVELOPMENT OF A MULTIMODAL SENSORIAL PLATFORM
RGB - THERMAL - 3D LiDAR FOR AN AUTONOMOUS
VEHICLE**

Author: Manuel Ricardo Diosa Usma.

Supervisor: Jesús Morales Rodríguez.

Co-supervisor: Jorge Montenegro Navarro.

Department: *Systems and Automation Engineering.*

Degree: *Electronics, Robotics and Mechatronics Engineering.*

Keywords: *Autonomous driving, multisensor platform, thermal cameras, stereo cameras, 3D LiDAR sensor, ROS2, SolidWorks, sensor-equipped vehicle .*

Abstract

This Final Degree Project presents the design and development of a multimodal sensor platform for autonomous vehicles, aimed at enhancing environmental perception. The system integrates four thermal cameras, effective in low-visibility conditions; three stereo cameras, which provide depth perception; and a 3D LiDAR sensor that generates accurate point clouds using laser pulses. Sensor integration was carried out using ROS2, enabling efficient data acquisition, synchronization, and management. Additionally, the sensor mounting base and computing hardware supports were designed using CAD tools to ensure proper installation on the vehicle.

Agradecimientos

No quiero dejar pasar la oportunidad de agradecer a todas las personas que, con su apoyo, consejos o compañía, han contribuido a que este trabajo sea posible.

Quiero agradecer especialmente a mi tutor, Jesús Morales Rodríguez, y a mi cotutor, Jorge Montenegro Navarro, por su ayuda constante y, sobre todo, por la paciencia que han tenido conmigo a lo largo de todo este trabajo. Les agradezco profundamente su guía, disponibilidad y comprensión en cada una de las etapas del proyecto. También quiero extender mi agradecimiento a los trabajadores del Taller 27, por su acogida, su compañía y el buen trato que siempre me han brindado durante el desarrollo del proyecto, y en especial a David Padial Allue por su ayuda y consejos en la parte de diseño mecánico.

No puedo dejar de agradecer a mi familia, y en especial a mi padre, mi madre y mi hermano, por darme todo por mí y por estar siempre a mi lado, no solo en los buenos momentos, sino aún más en los difíciles. Gracias por vuestro apoyo incondicional, por animarme cuando más lo he necesitado y por ser siempre mi mayor fuente de fuerza y motivación.

También quiero agradecer de corazón a mis amigos más cercanos: Marta, Guille, Kris, Cristina, Sherina y Jaan. Gracias por darme ánimos, comprensión, compañía y fuerza en algún momento a lo largo de todos estos años. Vuestra presencia ha sido fundamental para seguir adelante y no rendirme, y valoro profundamente cada gesto de apoyo que habéis tenido conmigo.

Acrónimos

3D	Tridimensional
ABS	<i>Antilock Braking System</i>
CAD	<i>Computer Aided Design</i>
CUDA	<i>Compute Unified Device Architecture</i>
DDS	<i>Data Distribution Service</i>
FOV	<i>Field of View</i>
GPS	<i>Global Positioning System</i>
GPU	<i>Graphics Processing Unit</i>
IP	<i>Internet Protocol</i>
LiDAR	<i>Light Detection and Ranging</i>
LWIR	<i>Long Wave InfraRed</i>
PC	<i>Personal Computer</i>
PREMOVE	Predicción del Movimiento de los participantes del tráfico para la integración segura del vehículo autónomo en áreas urbanas
RGB	<i>Red, Green, Blue</i>
ROS2	<i>Robot Operating System 2</i>
RViz	<i>ROS Visualization</i>
SAE	<i>Society of Automotive Engineers</i>
SDK	<i>Software Development Kit</i>
SLAM	<i>Simultaneous Localization And Mapping</i>
TFG	Trabajo de Fin de Grado

UDP *User Datagram Protocol*

USB *Universal Serial Bus*

Índice

Resumen	V
Abstract	VII
Agradecimientos	IX
Acrónimos	XI
1 Introducción	1
1.1 Antecedentes.	1
1.2 Marco del trabajo.	3
1.3 Motivación y Objeto.	3
1.4 Planificación del trabajo.	4
1.5 Estructura del documento.	5
2 <i>Hardware</i> del sistema	7
2.1 Introducción.	7
2.2 Sensores del sistema.	8
2.2.1 LiDAR 3D.	8
2.2.2 Cámara estéreo.	10
2.2.3 Cámara térmica.	12
2.3 Diseño en CAD.	14
2.3.1 Herramienta SolidWorks.	14
2.3.2 Desarrollo de soportes y carcasa.	14
2.3.2.1 Modelado del soporte para el sensor LiDAR.	15
2.3.2.2 Modelado del soporte para la cámara estéreo.	16
2.3.2.3 Modelado del soporte para la cámara térmica.	18

2.3.3	Distribución en el vehículo.	20
3	Software del sistema	21
3.1	Introducción.	21
3.2	ROS2.	21
3.3	Paquetes de los sensores.	23
3.3.1	Sensor 3D LiDAR Robosense Helios 5515.	24
3.3.2	Cámaras estéreo ZED 2i.	26
3.3.3	Cámaras térmicas Seek Thermal Compact Pro.	27
3.3.4	Integración conjunta.	31
4	Resultados	33
4.1	Introducción.	33
4.2	Soportes y carcasas de los sensores.	33
4.2.1	Soporte para el sensor LiDAR.	33
4.2.2	Soporte para la cámara estéreo.	34
4.2.3	Soporte para la cámara térmica.	35
4.3	Nube de puntos e imágenes obtenidas a través de los sensores	38
5	Conclusiones y líneas futuras	41
5.1	Introducción.	41
5.2	Conclusiones.	41
5.3	Líneas futuras.	42
I	Apéndices	43
A	Manual de Instalación	45
A.1	Marco Introductorio.	45
A.2	ROS2 en Ubuntu 22.04.	45
A.3	Instalación de Paquetes.	47
A.3.1	Sensor LiDAR 3D Robosense Helios 5515.	47
A.3.2	Cámara estéreo ZED 2i.	48
A.3.3	Cámara térmica Seek Thermal Compact Pro.	49
B	Manual de Usuario.	53

B.1	Sensor LiDAR 3D Robosense Helios 5515.	53
B.2	Cámaras estéreo ZED 2i.	53
B.3	Cámaras térmicas Seek Thermal Compact Pro.	54
B.4	Sistema de sensores general.	54
C	Planos de diseño en CAD	55
C.1	Introducción.	55
C.2	Disposición a nivel general.	55
C.3	Campo de Visión.	56
C.3.1	Sensor LiDAR 3D.	56
C.3.2	Cámaras estéreo.	56
C.3.3	Cámaras térmicas.	56
C.4	Distribución en la plataforma.	57
C.5	Dimensiones de los diseños.	57
C.5.1	Soporte Sensor LiDAR 3D.	57
C.5.2	Soporte Cámaras estéreo.	57
C.5.3	Carcasa Cámaras térmicas.	57
	Bibliografía	61

Capítulo 1

Introducción

Contenido

1.1	Antecedentes.	1
1.2	Marco del trabajo.	3
1.3	Motivación y Objeto.	3
1.4	Planificación del trabajo.	4
1.5	Estructura del documento.	5

1.1. Antecedentes.

El avance de la industria y la tecnología en los últimos años ha hecho posible la aparición de sistemas cada vez más autónomos, los cuales son capaces de realizar tareas de mayor complejidad con una menor intervención por parte del ser humano. Debido a esto, el sector del transporte se ha visto impactado de forma muy notable, y es por ello que los vehículos autónomos han pasado a un primer plano de investigación en el que se buscan activamente aplicaciones reales.

Desde principios de los años 2000, muchos fabricantes y empresas tecnológicas han incorporado sistemas automáticos en vehículos con el objetivo de optimizar la seguridad, la experiencia de conducción y la eficiencia del transporte. Sistemas como el mantenimiento de carril, la frenada automática de emergencia o el control de cruce adaptativo se han vuelto comunes en vehículos comerciales. A su vez, han surgido proyectos centrados en alcanzar niveles más altos de automatización, incluyendo vehículos capaces de desplazarse sin intervención humana [1].

Estos vehículos dependen en gran medida de su capacidad para percibir el entorno, razón por la cual integran distintos sensores especializados, como las cámaras RGB (*Red Green Blue*), que captan imágenes en color útiles para la detección de señales de tráfico y marcas viales; los sensores LiDAR (*Light Detection and Ranging*), que proporcionan mediciones precisas de distancia y permiten generar representa-

ciones tridimensionales del entorno mediante pulsos láser; las cámaras estéreo, que ofrecen visión en profundidad al captar imágenes desde dos perspectivas, lo que facilita la localización espacial de objetos; y los sensores térmicos, capaces de detectar diferencias de temperatura y especialmente eficaces en condiciones de baja visibilidad, como durante la noche o con meteorología adversa [2].

De cada uno de estos sensores se capta un tipo distinto de información, pero también presentan limitaciones si se utilizan de forma aislada. Por ejemplo, las cámaras RGB pueden fallar en condiciones de baja visibilidad donde haya poca luz, y el LiDAR, aunque preciso en la distancia, no ofrece información visual detallada. Para reducir o superar estas limitaciones, se han desarrollado técnicas de fusión sensorial, donde se combinan los datos de varios dispositivos y se obtiene una percepción más completa y fiable del entorno del vehículo [3].

La Sociedad de Ingenieros de Automoción (*SAE, Society of Automotive Engineers*) propone una clasificación del grado de automatización en los vehículos. Esta clasificación, conocida como SAE J3016 [4], define seis niveles de automatización, que van desde el Nivel 0 (sin asistencia en la conducción) hasta el Nivel 5 (automatización completa). Cada nivel establece el grado de intervención que se requiere por parte del conductor y las funciones que asume el sistema automatizado del vehículo. En los niveles más bajos, el ser humano debe mantener el control en todo momento, mientras que, a medida que se asciende en la escala, el sistema asume funciones críticas como el control de velocidad, la dirección o incluso la toma de decisiones en condiciones normales o complejas. En la Figura 1.1 se muestra una representación gráfica de esta clasificación, que ayuda a comprender de forma visual cómo evoluciona el papel del conductor en función del nivel de autonomía alcanzado.

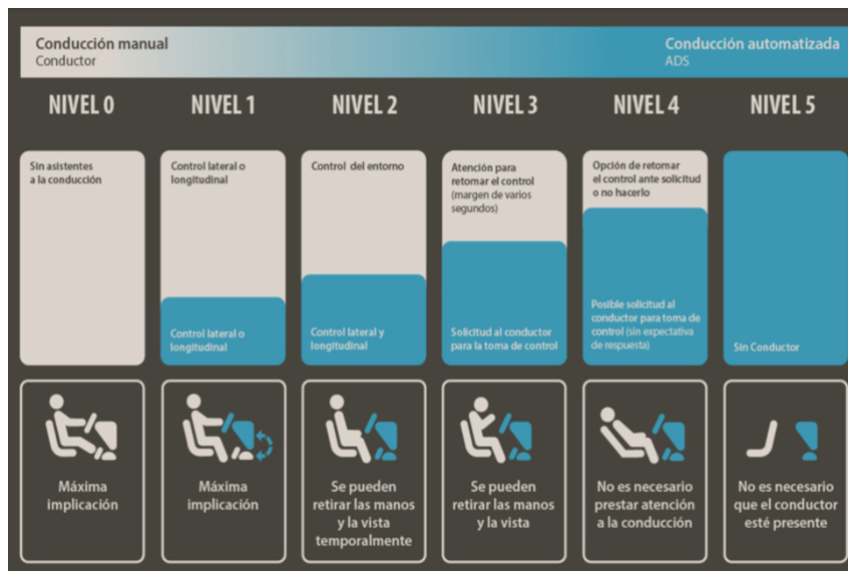


Figura 1.1: Implicación humana en la conducción según la autonomía [5].

A través de los distintos niveles de autonomía definidos por la SAE, se puede observar cómo varía el grado de intervención humana en la conducción. El Nivel

0 corresponde a vehículos sin ningún tipo de asistencia, mientras que el Nivel 1 introduce ayudas como el ABS (*Anti-Lock Brake System*), el control de crucero, el cambio automático de marchas o el aviso de salida de carril. En el Nivel 2, el vehículo puede controlar aceleración y frenado, aunque el conductor debe supervisar constantemente. A partir del Nivel 3, el sistema puede tomar decisiones de forma autónoma en determinadas condiciones y solicitar intervención en situaciones puntuales, como durante la conducción por autopista o en retenciones de tráfico. El Nivel 4, aún en fase de pruebas avanzadas, permite una conducción completa sin intervención humana en entornos controlados, y el Nivel 5 representa la autonomía total en cualquier entorno, sin necesidad de conductor. Aunque aún no están comercializados, y faltan un marco legal e infraestructuras adecuadas, ya existen prototipos funcionales [6].

1.2. Marco del trabajo.

El Departamento de Ingeniería de Sistemas y Automática de la Escuela de Ingenierías Industriales de la Universidad de Málaga está llevando a cabo un proyecto de investigación e innovación denominado PREMOVE (Predicción del Movimiento de los participantes del tráfico para la integración segura del vehículo autónomo en áreas urbanas) [7]. La Junta de Andalucía es la encargada de financiar esta iniciativa de excelencia bajo la referencia *ProyExcel_00684*.

El objetivo principal de PREMOVE es investigar y desarrollar nuevas estrategias que permitan mejorar la percepción del entorno por parte de los vehículos autónomos, centrándose especialmente en la detección, el análisis y la predicción del comportamiento de los participantes del tráfico, como peatones, ciclistas y vehículos motorizados. Estas estrategias están orientadas a incrementar la seguridad y fiabilidad de la conducción autónoma en entornos urbanos complejos con un nivel de incertidumbre, donde el sistema debe ser capaz de detectar obstáculos imprevistos que puedan aparecer de forma repentina en la trayectoria del vehículo.

1.3. Motivación y Objeto.

El objetivo del presente Trabajo de Fin de Grado es el diseño, construcción e implementación, en un entorno ROS2 (*Robot Operating System 2*), de una plataforma multisensorial que permita equipar un vehículo experimental con diversos sensores. Esta plataforma ha sido concebida para capturar datos del entorno urbano mediante cámaras térmicas, cámaras estéreo y sensores LiDAR, lo que permitirá generar conjuntos de datos (*datasets*) para el entrenamiento, validación y evaluación de algoritmos de percepción, fusión sensorial y predicción de trayectorias en futuras fases del proyecto.

1.4. Planificación del trabajo.

El desarrollo de este Trabajo de Fin de Grado requiere la combinación de distintas tareas, tanto de *software* como de *hardware*, y su evolución se basará en comprender cómo funciona y se complementa todo el sistema y sus componentes. A continuación, se detalla, de forma general, cómo se llevará a cabo dicho proceso.

En una primera fase, se preparará el entorno de trabajo mediante la instalación del sistema operativo Ubuntu 22.04, que servirá como base para el desarrollo. Se llevará a cabo un aprendizaje progresivo para familiarizarse con sus comandos y características. Posteriormente, se explorará el *middleware* ROS2, herramienta clave en el auge actual de los ecosistemas robóticos, realizando pruebas básicas de funcionamiento para entender su estructura y su potencial de comunicación entre nodos.

Una vez asentadas estas bases, se ha de estudiar el funcionamiento de los dispositivos que forman parte de la plataforma: cuatro cámaras térmicas Seek Thermal Compact Pro, tres cámaras estéreo ZED 2i de Stereolabs y un sensor LiDAR 3D Robosense Helios 5515. Se consultarán sus fichas técnicas, documentación oficial y recursos ofrecidos por los fabricantes, tanto a nivel de *hardware* como de *software*. Cada uno será probado de forma individual para comprender sus capacidades y limitaciones. Posteriormente, se adaptarán los paquetes de ROS2 y los *scripts* en Python necesarios para lograr su funcionamiento, ya que no están diseñados para funcionar en conjunto con todos los elementos que se quieren utilizar, sino de forma aislada.

Paralelamente, se desarrollará la fase de diseño físico del sistema. El objetivo será montar todos los sensores sobre una plataforma ubicada en la baka de un vehículo, junto a los ordenadores, el cableado y otros dispositivos como antenas GPS. Se estudiará la mejor forma de distribuirlos sobre la superficie para maximizar el campo de visión y minimizar interferencias o solapamientos entre ellos. Dentro de este montaje, será necesario diseñar carcasas personalizadas que garanticen la protección y estabilidad de cada componente. Dichas carcasas se modelarán en SolidWorks, tras realizar una primera toma de contacto con esta herramienta CAD (*Computer Aided Design*). A continuación, se avanzará en los diseños definitivos y, tras la obtención de varios prototipos mediante impresión 3D, se realizarán ajustes y validaciones para asegurar un buen encaje y robustez.

Finalmente, una vez completado todo el proceso técnico y las pruebas correspondientes, se abordará la redacción de la memoria, con el objetivo de documentar de forma clara y ordenada todos los pasos seguidos. Esta etapa permitirá consolidar los objetivos del proyecto, poner en valor la utilidad del sistema desarrollado y dejar constancia del trabajo realizado. Dicha documentación se establecerá como punto de partida para futuras ampliaciones, integraciones o mejoras dentro del marco del proyecto REMOVE.

1.5. Estructura del documento.

El presente documento se encuentra dividido en capítulos y anexos, cada uno de los cuales aborda una parte concreta del trabajo realizado. A continuación, se resume brevemente el contenido de cada sección:

- En el Capítulo 1 se introduce el trabajo a realizar, contextualizando el proyecto y explicando de forma general su finalidad, motivación y metodología.
- En el Capítulo 2 se desarrolla la parte relacionada con el *hardware*, donde se presentan los diferentes sensores utilizados y se detallan las fases de diseño de las carcasas y soportes mediante el *software* CAD SolidWorks.
- A continuación, en el Capítulo 3, se aborda la parte de *software*, justificando el uso del sistema operativo Ubuntu 22.04 y del *middleware* ROS2. Se explican los paquetes proporcionados por los fabricantes y las modificaciones realizadas para su puesta en marcha.
- En el Capítulo 4 se presentan los resultados obtenidos tras la implementación del sistema, incluyendo pruebas realizadas.
- Por último, en el Capítulo 5 se exponen las conclusiones generales, así como una reflexión sobre posibles líneas futuras del proyecto.
- Como apéndices, se incluye el Apéndice A, que actúa como guía técnica de instalación de los paquetes necesarios, tanto del entorno ROS2 en Ubuntu 22.04 como de cada uno de los sensores por separado. Además, se añade el Apéndice B a modo de manual de usuario del sistema y por último, el Apéndice C donde se presentan planos generales del conjunto y vistas más detalladas de los soportes y carcasas creadas.
- Finalmente, se incorpora la Bibliografía, donde se recogen las fuentes, recursos técnicos y documentación consultada para el desarrollo del trabajo.

Capítulo 2

Hardware del sistema

Contenido

2.1	Introducción.	7
2.2	Sensores del sistema.	8
2.2.1	LiDAR 3D.	8
2.2.2	Cámara estéreo.	10
2.2.3	Cámara térmica.	12
2.3	Diseño en CAD.	14
2.3.1	Herramienta SolidWorks.	14
2.3.2	Desarrollo de soportes y carcasa.	14
2.3.3	Distribución en el vehículo.	20

2.1. Introducción.

En este capítulo se describe el conjunto de componentes físicos que forman parte del sistema a desarrollar en el presente Trabajo Fin de Grado. El *hardware* empleado consta de un conjunto de sensores que percibirán datos del entorno que los rodea, haciendo uso de diferentes tecnologías. En primer lugar, se presentan los dispositivos de captación de información utilizados: un sensor LiDAR 3D, cámaras estéreo y cámaras térmicas, que se complementan, permitiendo obtener información del entorno.

Tras esto, se detalla el proceso de diseño mecánico realizado mediante *software* CAD, en particular con la herramienta SolidWorks. Esto ha sido necesario para desarrollar soportes personalizados y asegurar la correcta fijación y orientación de los sensores en el vehículo, garantizando tanto su funcionalidad como su protección física. Todo el *hardware* descrito en este capítulo se integra como base del sistema perceptivo sobre el cual se apoya el resto del desarrollo en ROS2 y las funcionalidades autónomas del sistema.

2.2. Sensores del sistema.

2.2.1. LiDAR 3D.

LiDAR es una tecnología de detección remota que mide distancias con alta precisión. Su funcionamiento es prácticamente igual al de un sonar, pero a diferencia de este (que usa ondas acústicas), el LiDAR usa pulsos de luz, midiendo el tiempo que tarda en reflejarse y en regresar al sensor. Así, es posible calcular distancias a objetos y combinar múltiples mediciones, que generan representaciones tridimensionales detalladas. En el ámbito de aplicación de este trabajo, es especialmente útil en la detección y evitación de obstáculos en la conducción autónoma [8].

Se ha decidido emplear el sensor LiDAR 3D modelo Helios 5515 de la empresa RoboSense que se muestra en la Figura 2.1. Está diseñado para aplicaciones en robótica, vehículos autónomos y mapeo, ofreciendo una alta densidad de puntos y un amplio campo de visión para una percepción precisa del entorno [9].



Figura 2.1: Sensor LiDAR 3D Robosense Helios 5515 [10].

Los detalles técnicos más destacables de este modelo se presentan en la Tabla 2.1, los cuales se han tomado a la hora de la creación de piezas en SolidWorks.

Parámetro	Valor
Número de haces	32
Campo de visión horizontal (FOV)	360°
Campo de visión vertical (FOV)	70° (-55° a +15°)
Resolución angular horizontal	0.2° / 0.4°
Resolución angular vertical	Hasta 1.33°
Precisión de medición	±2 cm (1–100 m) ±3 cm (resto)
Rango de detección	Hasta 150 m
Velocidad de rotación	600 / 1200 rpm (10 / 20 Hz)
Tasa de datos	576.000 pts/s (retorno único), 1.152.000 pts/s (doble retorno)
Longitud de onda del láser	905 nm
Clase de seguridad láser	Clase 1
Consumo de energía	12 W (típico)
Voltaje de operación	9 - 32 V
Temperatura de operación	-30°C a 60°C
Peso	1 kg
Dimensiones	100 mm (diámetro) × 100 mm (altura)
Grado de protección	IP67 (resistente al polvo y al agua)

Tabla 2.1: Especificaciones técnicas del sensor LiDAR Robosense Helios 5515 [9, 11].

En la Figura 2.2 que se presenta a continuación, se puede observar el funcionamiento del dispositivo y cómo son captados los datos y representados como nubes de puntos. Esto permite analizar el entorno con puntos situados a su respectiva distancia respecto del LiDAR. Herramientas como RSView, proporcionada por Robosense, muestran en tiempo real los datos, facilitando su integración en sistemas de navegación y percepción.

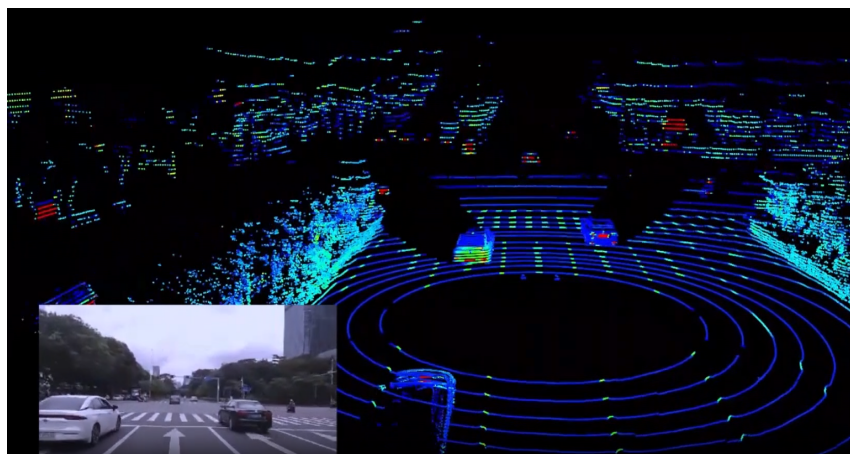


Figura 2.2: Visión del sensor LiDAR 3D Robosense Helios 5515 [12].

2.2.2. Cámara estéreo.

Al hablar de cámara estéreo, se hace referencia a un sistema de visión artificial con dos lentes (o sensores) que están separadas horizontalmente para capturar imágenes desde dos perspectivas diferentes, de forma que se simula la visión binocular humana. Teniendo en cuenta las diferencias entre las imágenes recogidas por cada lente, el sistema calcula la profundidad de cada píxel y genera una representación tridimensional del entorno.

Este tipo de cámaras tienen gran aplicación en robótica móvil y navegación autónoma, ya que facilitan estimar distancias sin necesidad de sensores activos como el LiDAR, siendo especialmente útiles en lugares bien iluminados o interiores. El modelo de cámara utilizado es el ZED 2i, perteneciente a la empresa StereoLabs. Este se muestra en la Figura 2.3.



Figura 2.3: Cámara estéreo ZED 2i [13].

Con este modelo se tiene una cámara estéreo RGB que incluye sensores inerciales integrados, ideal para mapeo 3D, navegación y SLAM (*Simultaneous Localization And Mapping*). También, como con el sensor anterior, se ha hecho uso de la información de sus características principales a la hora de diseñar los soportes necesarios para sujetar el dispositivo a la plataforma. Las principales características se muestran en las Tablas 2.2 y 2.3:

Parámetro	Valor
Resolución de captura	Hasta 2K a 15 fps, 1080p a 30 fps y 720p a 60 fps
Distancia de detección	Hasta 20 metros (modo 2K)
Campo de visión (FOV)	110° en diagonal
Distancia entre lentes	120 mm
Sensores adicionales	IMU, magnetómetro y barómetro integrados. Mejoran la estimación de movimiento y pose.

Tabla 2.2: Especificaciones funcionales de la cámara estéreo ZED 2i [13].

Parámetro	Valor
Interfaz	USB 3.0 con conector físico USB tipo C.
Montaje y conectividad	Carcasa de aluminio y frontal metálico intercambiable. Montaje roscas UNC 1/420 y M4.
Compatibilidad	SDK oficial, ROS, ROS2, Jetson, etc.

Tabla 2.3: Opciones de conectividad y compatibilidad de la cámara ZED 2i [14].

Se puede usar en combinación con el ZED SDK (*Software Development Kit*) para obtener nube de puntos 3D, seguimiento de objetos, SLAM, entre otros [13, 14, 15]. El funcionamiento está basado en el cálculo de profundidad usando la técnica de estéreo pasivo, donde no hay proyección de luz. Es por ello que no emite láser ni infrarrojo (lo que beneficia en exteriores, pese a la sensibilidad a las condiciones de iluminación). Se generan mapas de profundidad a partir del análisis de texturas y contrastes entre las imágenes captadas por cada una de las lentes de la cámara.

Como incluyen sensores de inercia, hay una mejora en la odometría visual-espacial (estimación de movimiento), lo que las hace útiles para fusión sensorial. Ejemplos de uso y visualización de este tipo de dispositivos son los que se muestran en la Figura 2.4. Se aprecian usos tales como: seguimiento corporal, detección de objetos, mapeo espacial, seguimiento geográfico, detección de planos, fusiones multicámara, etc.

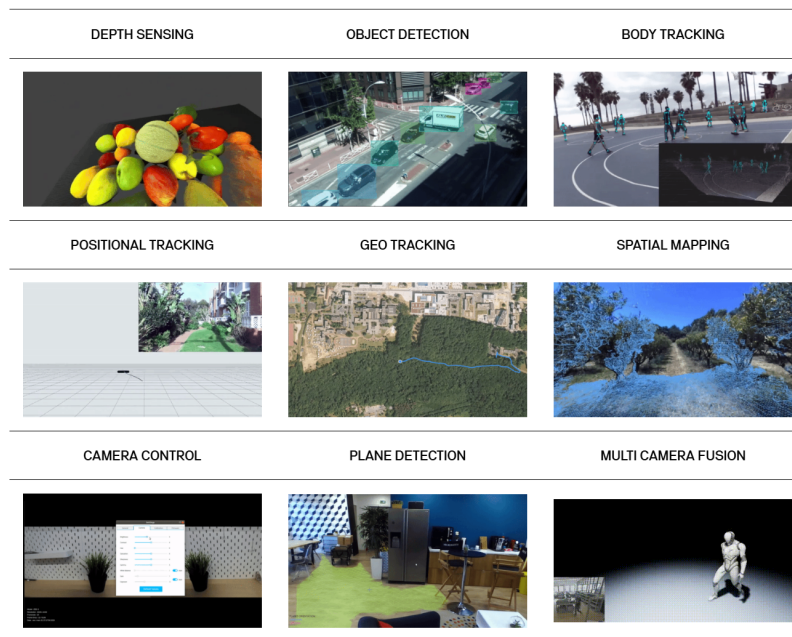


Figura 2.4: Ejemplos de utilización de una cámara ZED 2i de StereoLabs en diferentes ámbitos [15].

2.2.3. Cámara térmica.

El último tipo de sensor a utilizar es una cámara térmica. Se trata de un dispositivo que detecta radiación infrarroja emitida por los objetos en función de su temperatura y genera imágenes basadas en el calor emitido por los cuerpos. Es por ello que son útiles para vigilancia nocturna, detección en baja visibilidad, conducción autónoma en condiciones adversas, etc. Su funcionamiento se basa en sensores infrarrojos no refrigerados (frecuentemente microbolómetros), que miden la radiación en el rango de longitudes de onda 8–14 μm (LWIR, *Long-Wave InfraRed*). En este caso, se utiliza el modelo Compact PRO, fabricado por la compañía SeekThermal, el cual se muestra en la Figura 2.5, que se presenta a continuación.



Figura 2.5: SeekThermal Compact PRO [16].

La Seek Thermal Compact Pro es una cámara térmica pasiva:

- No emite ningún tipo de luz o láser.
- Detecta la radiación térmica natural.
- Ofrece una imagen 2D donde cada píxel representa una temperatura relativa del entorno.

Las características principales detalladas por el fabricante en el *datasheet* y en los distintos manuales de usuario son las que se presentan en la Tabla 2.4.

Parámetro	Valor
Resolución térmica	320 × 240 píxeles (QVGA)
Campo de visión (FOV)	32° (horizontal)
Rango de detección	Hasta 550 m (según condiciones)
Rango de temperatura	-40 °C a 330 °C
Interfaz	USB-C (o micro-USB en algunos modelos)
Compatibilidad	Android y iOS mediante <i>app</i> oficial
Frecuencia de actualización	15 Hz (en modelos Pro)

Tabla 2.4: Especificaciones técnicas de Seek Thermal Compact Pro [17, 18, 19].

Esta cámara está diseñada originalmente para *smartphones*, pero puede integrarse en otros sistemas con adaptaciones mediante *scripts* y visualización en PC (*Personal Computer*), como es el caso de este proyecto.

Las aplicaciones de estos dispositivos en el campo de la robótica son muy diversas, tales como: detección de obstáculos o personas en zonas de baja luminosidad, refuerzo de visión multimodal (fusión con cámaras RGB y LiDAR), tareas de búsqueda y rescate, navegación en entornos sin luz, etc. Las imágenes térmicas típicas, como la que se muestra en la Figura 2.6, se visualizan en escala de grises o *colormap* tipo *Iron*, donde el blanco o amarillo muestran las zonas calientes y el negro o azul oscuro las zonas frías.



Figura 2.6: Ejemplo de imagen térmica típica obtenida con esta cámara [20].

2.3. Diseño en CAD.

2.3.1. Herramienta SolidWorks.

La herramienta utilizada para la creación y desarrollo de las carcasas y soportes de los distintos elementos del proyecto ha sido SolidWorks, un CAD ampliamente empleado en entornos industriales y académicos para el modelado paramétrico en tres dimensiones.

La elección ha estado basada en la facilidad de diseño de forma precisa, la utilidad a la hora de montar ensamblajes complejos y la opción de crear planos técnicos compatibles con impresión en 3D. Además, esta herramienta ha permitido validar las dimensiones y el acoplamiento de las piezas gracias a sus funciones de simulación y ajuste de tolerancias, asegurando que los componentes pudieran montarse correctamente en el sistema final.

2.3.2. Desarrollo de soportes y carcasa.

A lo largo de este apartado se presentan los modelos desarrollados para cada uno de los sensores utilizados. Cada diseño ha sido realizado desde cero, considerando el tipo de sensor y sus condiciones específicas de montaje, su campo de visión y ensamblaje con la plataforma sensorial.

Asimismo, se ha considerado el espacio requerido para el paso de la tornillería normalizada y el cableado. Se ha partido de referencias oficiales proporcionadas por los fabricantes, cuando han estado disponibles, o a partir de mediciones propias

(con herramientas como el pie de rey) y se han generado los modelos con vista a su fabricación mediante impresión 3D en plástico o, en el caso del soporte del LiDAR, mediante corte, taladrado y pliegue de metal.

2.3.2.1. Modelado del soporte para el sensor LiDAR.

Prestando atención a la forma, tal y como se mostró en la Figura 2.1 y teniendo en cuenta características como el peso de 1 kilogramo y el campo de visión horizontal y vertical de 360° y 70° , respectivamente (Tabla 2.1), se optó por realizar este diseño en acero. Esta pieza ha de ser fácil de montar y desmontar en la plataforma sobre el vehículo, para su mantenimiento o posibles ajustes requeridos e ir fijada con una tornillería que le proporcione robustez y estabilidad. Se trata de una estructura con una altura superior a la del resto de soportes sobre la que se fija el sensor, tal y como se muestra en la Figura 2.7 de las vistas isométrica, frontal, lateral y superior obtenidas directamente de SolidWorks.

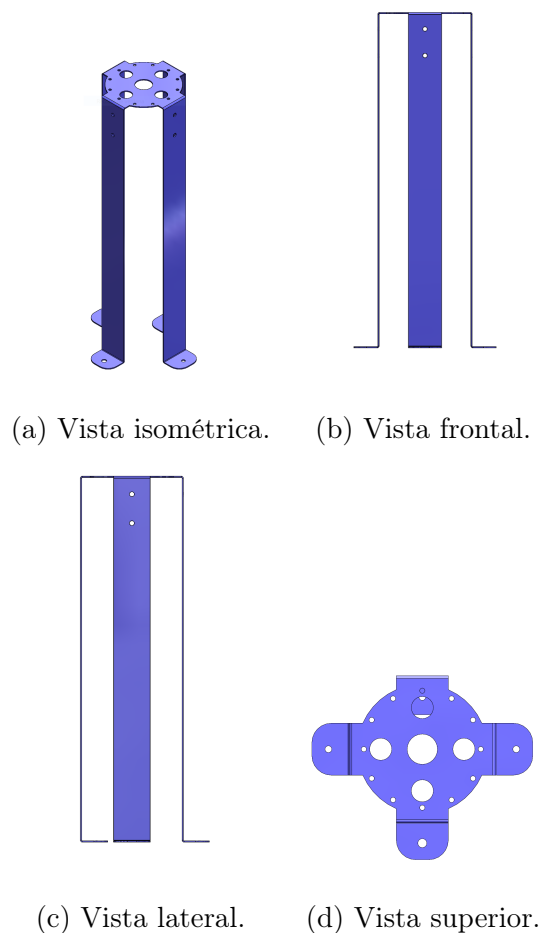


Figura 2.7: Distintas vistas del soporte para el sensor LiDAR.

En la figura 2.8 se presenta el LiDAR 3D montado sobre su soporte.

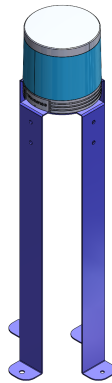
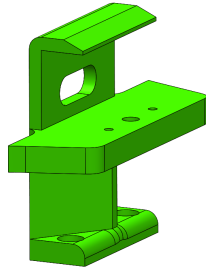


Figura 2.8: Sensor LiDAR 3D montado en el soporte.

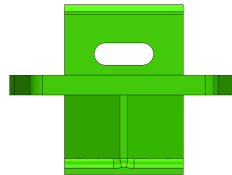
2.3.2.2. Modelado del soporte para la cámara estéreo.

En el caso de esta cámara, se ha fabricado un soporte que actúa como brazo para agarrarla, de forma que la única función que tiene dicho soporte es darle estabilidad y sujeción a la chapa metálica situada sobre el coche. No se ha cubierto el resto de la superficie de la cámara con ningún tipo de material o carcasa para evitar interferir en el campo de visión de ambas lentes. La pieza posee los orificios necesarios para que la cámara quede bien encajada según la geometría con la que viene de fábrica, ya sean orificios para introducir el cableado u orificios creados por el fabricante para el anclaje a alguna estructura.

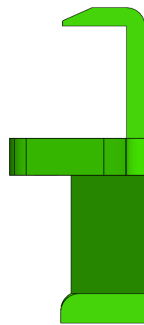
Se presenta, a continuación, la Figura 2.9 donde se muestran las diferentes perspectivas del soporte creado.



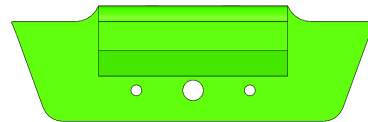
(a) Vista isométrica.



(b) Vista frontal.



(c) Vista lateral.



(d) Vista superior.

Figura 2.9: Distintas vistas de la pieza para la cámara estéreo ZED 2i.

Asimismo, la Figura 2.10 refleja el montaje de la cámara en la pieza creada.

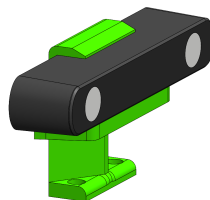


Figura 2.10: Cámara estéreo ZED 2i montada en el soporte.

2.3.2.3. Modelado del soporte para la cámara térmica.

En esta última cámara, a diferencia de las otras dos, se ha tenido que trabajar un poco más debido a que la base desde la que se ha partido ha sido única y exclusivamente la cámara, la cual no posee ningún orificio o enganche de fábrica. En un primer momento se consideraron diferentes alternativas de diseño: ya fuera mediante sujeción, como en el caso de la ZED 2i; encajada en una pieza que, a su vez, se dividía en dos mitades que se atornillaban; o mediante otras posibles soluciones.

Finalmente, se decidió crear una carcasa cuya funcionalidad fuera proteger la cámara, ya que por su tamaño y su poco peso es bastante susceptible a las adversidades del entorno y a desestabilizarse. La carcasa cuenta con los orificios necesarios para que la cámara quede bien ajustada. Además, posee un orificio trasero a través del cual se puede introducir alguna herramienta para extraer la cámara, ya que esta queda prácticamente a presión. La carcasa diseñada no cubre la totalidad del cuerpo de la cámara, sino que deja libre su lente regulable, cuya función es ajustar el enfoque en la imagen térmica proporcionada.

También se ha tenido en cuenta la necesidad de proteger el puerto de conexión de la cámara, así como el cable conectado a este, con el fin de evitar desconexiones u otros inconvenientes durante el funcionamiento. Tras la creación de la carcasa, se ha diseñado una tapa que la cubre casi por completo, dejando únicamente expuesta la lente de la cámara. Esta tapa se atornilla a través de cuatro orificios de sujeción, uno en cada esquina.

Todo el conjunto diseñado se atornilla, por la parte trasera de la carcasa, a la chapa metálica y queda orientado según lo necesario (y medido) para obtener el mayor rango de visión posible. A continuación, en la Figura 2.11, se muestran las distintas vistas principales de la carcasa diseñada para la cámara térmica. Estas vistas permiten observar la geometría general y la ubicación de los orificios necesarios para su fijación y funcionalidad. Cabe destacar que, en este caso, ha sido necesario el uso de un pie de rey para ajustar las medidas con precisión durante la construcción de la carcasa.



(a) Vista isométrica. (b) Vista frontal. (c) Vista lateral. (d) Vista superior.

Figura 2.11: Diferentes vistas de la carcasa diseñada para la cámara térmica.

La Figura 2.12 muestra la carcasa con la cámara térmica insertada en su interior. Esta configuración permite observar cómo queda ajustada a presión y cómo se respeta la visibilidad de la lente regulable.

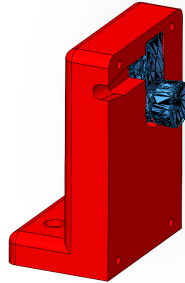
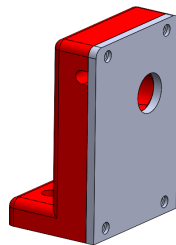
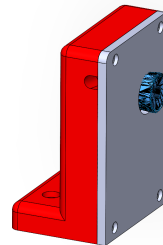


Figura 2.12: Carcasa con la cámara térmica insertada y sujeta a presión.

Finalmente, en la Figura 2.13, se presentan dos visiones del conjunto completo formado por la carcasa y su tapa: una sin cámara en el interior, que permite ver su configuración vacía, y otra con la cámara térmica ya colocada y protegida dentro del sistema.



(a) Conjunto vacío de carcasa y tapa.



(b) Conjunto carcasa y tapa con la cámara protegida.

Figura 2.13: Montaje final del conjunto carcasa y tapa, con y sin cámara en su interior.

2.3.3. Distribución en el vehículo.

En este punto se presenta la distribución completa del conjunto de sensores, compuesto por un LiDAR, tres cámaras estéreo y cuatro cámaras térmicas. Para reflejar esta disposición, se muestra la Figura 2.14, donde puede observarse cómo se han distribuido.

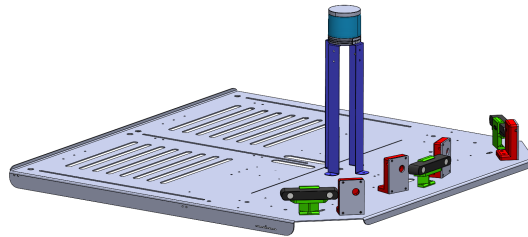


Figura 2.14: Distribución del conjunto de sensores sobre la plataforma sensorial.

Antes de que el diseño y la construcción de la plataforma sensorial estuvieran finalizados, fue necesario buscar una solución provisional. Para ello, se utilizó un tablón de madera sobre el que se colocaron los primeros prototipos, utilizándose en ese momento únicamente tres cámaras térmicas. Se realizó una serie de pruebas y se determinó que el FOV (*Field Of View*) alcanzado con tres térmicas no era suficiente, por lo que se tomó la decisión de añadir una cuarta cámara.

Posteriormente, el sistema completo fue fijado sobre una plataforma de aluminio específicamente diseñada con la herramienta SolidWorks (véase en la Figura 2.14). No obstante, dicho diseño queda fuera del alcance del presente Trabajo de Fin de Grado, ya que fue llevado a cabo por otro integrante del proyecto PREMOVE.

Las dimensiones exactas de los soportes, carcasas y chapa pueden consultarse en el Apéndice C, dedicado a los planos técnicos que acompañan esta memoria. En dicho anexo también se detallan las cotas de los diseños, así como las distancias y orientaciones seleccionadas para el montaje completo del sistema. En cuanto al proceso de fabricación, el resultado final se muestra en las figuras incluidas en el Capítulo 4.

Capítulo 3

Software del sistema

Contenido

3.1	Introducción.	21
3.2	ROS2.	21
3.3	Paquetes de los sensores.	23
3.3.1	Sensor 3D LiDAR Robosense Helios 5515.	24
3.3.2	Cámaras estéreo ZED 2i.	26
3.3.3	Cámaras térmicas Seek Thermal Compact Pro.	27
3.3.4	Integración conjunta.	31

3.1. Introducción.

En este capítulo se presenta la parte del Trabajo de Fin de Grado relacionada con la programación de todo el sistema de sensores. Se trata de una etapa en la que se hará uso de herramientas comunes en el ámbito de la robótica, como ROS2 y sistemas operativos en los que la comunicación entre dispositivos de este calibre es más sencilla, dinámica y versátil (Ubuntu 22.04).

Es en esta parte, donde serán necesarias modificaciones en los paquetes de funcionamiento de los elementos utilizados en el proyecto (a los paquetes o configuraciones proporcionadas por el fabricante). Se han adaptado y modificado dichos paquetes para adquirir datos del sistema sensorial.

3.2. ROS2.

Antes de entrar en detalle con lo comentado anteriormente, es necesario introducir y describir brevemente ROS2 como el núcleo de trabajo principal, siendo este un

middleware, es decir, un puente entre el *hardware* (sensores, en este caso las cámaras y el LiDAR) y el *software* (archivos, paquetes, nodos, etc.) que gestiona su funcionamiento, pudiéndose usar de forma clara y escalable. Esta arquitectura distribuida permite la comunicación entre estos sin depender de un nodo central, siendo muy útil a la hora del procesamiento en tiempo real y con una robustez crítica frente a fallos, perturbaciones o cuellos de botella.

El funcionamiento de ROS2 se basa en una serie de componentes clave, entre los que destacan los nodos, los tópicos, los servicios y los paquetes. Cada nodo es un proceso que realiza una tarea concreta. Algunos actúan como publicadores (*publishers*), enviando información a través de un canal llamado tópico. Otros nodos funcionan como suscriptores (*subscribers*), que reciben los datos de ese canal para procesarlos o actuar en consecuencia. En la Figura 3.1 se muestra un esquema de la arquitectura de ROS2.

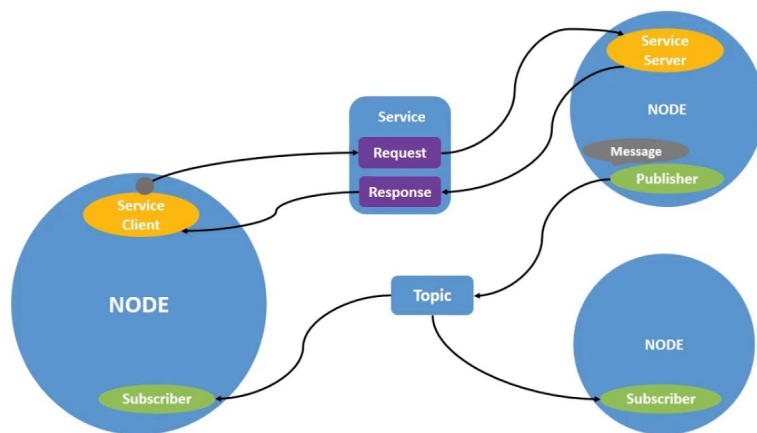


Figura 3.1: Esquema ejemplo de funcionamiento de ROS2 [21].

Por ejemplo, una cámara térmica puede publicar imágenes en un tópico específico (`/thermal/image_raw`) y un nodo suscriptor puede leer esa información para procesarla o visualizarla. Este sistema de comunicación se basa en una infraestructura DDS (*Data Distribution Service*) que garantiza el envío ordenado y en tiempo real de los mensajes entre nodos, lo cual es vital para sistemas autónomos.

También existen los servicios, que permiten que un nodo le pida una tarea concreta a otro (por ejemplo, solicitar la posición actual del vehículo o iniciar una grabación). Es una comunicación bidireccional y sincronizada, a diferencia de los tópicos, que son asíncronos y unidireccionales.

A nivel de organización, todo esto se agrupa en paquetes, que contienen el código, los scripts, las configuraciones y los lanzadores (`.launch.py`) necesarios para cada funcionalidad.

Además de la arquitectura, ROS2 incorpora herramientas que resultan muy útiles para visualizar y depurar el comportamiento del sistema. En este proyecto se ha

utilizado RViz2, un visor 3D que permite ver, en línea, datos como imágenes, nubes de puntos 3D o trayectorias.

En la plataforma multisensorial del proyecto, se ha configurado cada sensor (LiDAR, térmico y estéreo) para que funcione como un nodo que publica datos a un tópico específico. El LiDAR publica datos de nube de puntos en el tópico `/rslidar_points`. Las cámaras térmicas publican imágenes en `/thermal_cameras`. Las cámaras ZED 2i usan su propio *wrapper* y publican datos en `/zed2i/odom`, `/zed2i/rgb/image_rect_color` y `/zed2i/imu/data`.

Cada uno de estos tópicos puede ser escuchado para visualizar los datos (como RViz), grabarlos (como ROS2 Bag) o procesarlos mediante nodos. Este enfoque modular permite que, incluso si un sensor falla, los demás sigan funcionando y aportando información útil. Para el desarrollo del trabajo se ha instalado la distribución ROS2 Humble Hawksbill [22]. Esta versión es recomendada oficialmente para Ubuntu 22.04 y ha demostrado ser estable para trabajar con nodos, sensores y simuladores.

Para el desarrollo del sistema multisensorial se ha utilizado como entorno base el sistema operativo Ubuntu 22.04 LTS, ya que es la distribución más comúnmente utilizada en entornos de robótica, y es compatible de forma nativa con ROS2 [23]. Esta versión proporciona un sistema estable y actualizado, con soporte a largo plazo, lo cual es importante en proyectos que integran diferentes librerías, drivers y herramientas externas.

Una de las principales razones para utilizar Ubuntu es que muchas herramientas robóticas (como ROS2, RViz o ROS2 Bag) están pensadas y optimizadas para funcionar en este sistema. Además, la comunidad de usuarios y desarrolladores es muy activa, lo que facilita encontrar documentación y soporte ante problemas [22, 24].

3.3. Paquetes de los sensores.

En cuanto a los sensores utilizados, cada uno se ha integrado mediante su respectivo paquete ROS2. En el caso de las cámaras estéreo ZED 2i, estas han sido configuradas usando el *wrapper* oficial desarrollado por Stereolabs [25]. Respecto al sensor LiDAR Robosense, ha sido integrado con su SDK adaptado a ROS2 [26]. Finalmente, las cámaras térmicas Seek Thermal Compact Pro no cuentan con un paquete oficial para ROS2, por lo que para su integración, se ha utilizado una solución intermedia basada en una biblioteca en Python que permite acceder al flujo de imágenes térmicas, y posteriormente se ha creado un nodo personalizado en ROS2 que publica estas imágenes en un tópico específico. Ambos paquetes permiten la publicación de datos en tópicos ROS2 para ser suscritos o grabados mediante herramientas como RViz o ROS2 Bag. Todos estos paquetes tienen la opción de guardar el instante en el que se toman los datos (*timestamp*).

3.3.1. Sensor 3D LiDAR Robosense Helios 5515.

Para integrar este sensor en un sistema basado en ROS2, se emplea el paquete `rslidar_sdk`, desarrollado y mantenido por el propio fabricante Robosense. Este se encarga de funciones como:

- Recepción de datos del sensor en tiempo real (UDP, *User Datagram Protocol*) o desde archivos `.pcap` para pruebas.
- Decodificación, calibración y procesamiento según el modelo del sensor.
- Publicación de la nube de puntos en formato ROS2 (`PointCloud2`).
- Configuración sencilla a través de archivos `.yaml` que permiten definir parámetros como IP (*Internet Protocol*), puertos, tipo de entrada, entre otros.
- Visualización directa en RViz y compatibilidad con herramientas como ROS2 Bag para grabación de datos.

En cuanto al diseño de `rslidar_sdk`, este está basado en un nodo único centralizado (`rslidar_node`) que se encarga de todo el procesamiento del sensor. A diferencia de otros *drivers* que separan el flujo de trabajo en varios nodos, este paquete integra en una sola instancia la entrada de datos desde el sensor, la decodificación y corrección basada en calibraciones internas y la publicación de mensajes ROS2 con los datos procesados.

Esta arquitectura mejora el rendimiento, reduce la complejidad del sistema y facilita la integración con otros sensores (como las cámaras estéreo y térmicas utilizadas en el proyecto).

El paquete (`rslidar_sdk`) permite configurar el tipo de punto que se generará en las nubes de puntos publicadas por el nodo. Los formatos soportados son XYZI, que contiene únicamente la información espacial y la intensidad del retorno, y XYZIRT, que además añade el identificador del anillo (*ring*) y la marca temporal (*timestamp*) de cada punto. En este proyecto, el tipo de punto seleccionado es XYZIRT, ya que permite una mejor sincronización con las cámaras estéreo y térmicas, así como una segmentación vertical más precisa al conocer la capa de origen del haz láser.

El código del paquete se encuentra organizado en varias carpetas. Las más relevantes se muestran en la Tabla 3.1, que se presenta a continuación.

Carpeta	Descripción
<code>src/driver/</code>	Contiene el nodo principal (<code>rslidar_node</code>). Aquí se define cómo se inicializa el nodo, se gestionan los parámetros y se controla el flujo completo desde la recepción hasta la publicación de datos.
<code>src/input/</code>	Gestiona la entrada de datos desde el sensor (vía UDP) o desde archivos <code>.pcap</code> . Aquí se encuentra la lógica que recibe los paquetes crudos enviados por el LiDAR.
<code>src/decoder/</code>	Contiene los algoritmos específicos que interpretan los paquetes recibidos, transformándolos en puntos 3D válidos según el modelo del sensor.
<code>config/</code>	Incluye los archivos YAML para definir parámetros como tipo de punto, modelo de sensor, IP, puerto, tipo de entrada, etc.
<code>launch/</code>	Contiene los scripts para lanzar el nodo usando ROS2, facilitando su ejecución con parámetros predefinidos.

Tabla 3.1: Carpetas principales del paquete `rslidar_sdk`.

Estos módulos trabajan de forma conjunta dentro del nodo único del paquete, permitiendo una ejecución eficiente y modular, y además de los principales, el paquete `rslidar_sdk` incluye otras carpetas y archivos útiles para la visualización, configuración y depuración del sistema:

- `rviz/`: incluye configuraciones predefinidas para visualizar la nube de puntos en RViz2. Permite representar los datos por intensidad o por anillo (*ring*), facilitando la validación visual del sensor.
- `config/`: además de los archivos de parámetros generales, contiene ficheros de calibración angular específicos del modelo Helios. Estos datos permiten compensar desviaciones mecánicas o errores de fábrica.
- `launch/`: contiene scripts para lanzar fácilmente el nodo principal (`rslidar_node`) con diferentes configuraciones de red, tipo de entrada (UDP o `.pcap`), frecuencia, tipo de nube, etc.
- `rslidar_msgs/`: define mensajes personalizados utilizados internamente por el paquete, como `RslidarScan` o `RslidarPacket`, que representan datos crudos o escaneos intermedios antes de generar la nube de puntos final.

Estas funcionalidades adicionales complementan el núcleo del sistema y permiten una integración más robusta, especialmente durante fases de prueba, visualización y calibración. La Tabla 3.2 resume la estructura del paquete `rslidar_sdk` utilizado para la integración del sensor LiDAR.

Carpeta	Descripción y función principal
<code>src/driver/</code>	Implementa el nodo principal (<code>rslidar_node</code>). Gestiona la inicialización del sensor, lectura de parámetros y publicación de la nube de puntos.
<code>src/input/</code>	Módulo que recibe datos crudos del sensor en tiempo real (UDP) o desde archivos <code>.pcap</code> para pruebas y simulaciones.
<code>src/decoder/</code>	Contiene los algoritmos que interpretan los paquetes crudos, aplican correcciones y generan los puntos 3D válidos.
<code>rslidar_msgs/</code>	Define mensajes personalizados (<code>RslidarPacket</code> , <code>RslidarScan</code>) usados internamente antes de publicar la nube final.
<code>config/</code>	Archivos de configuración en formato YAML o CSV. Permiten definir parámetros del sensor, calibraciones angulares y tipo de nube generada.
<code>launch/</code>	Scripts para lanzar el nodo con diferentes configuraciones. Facilita el despliegue rápido en ROS2.
<code>rviz/</code>	Configuraciones prediseñadas para RViz2. Permiten visualizar la nube de puntos por intensidad, anillo (<i>ring</i>), tiempo(<i>timestamp</i>), etc.

Tabla 3.2: Estructura del paquete `rslidar_sdk` y función de sus carpetas principales.

3.3.2. Cámaras estéreo ZED 2i.

Para la integración de las cámaras ZED 2i en el sistema ROS2 no ha sido necesario modificar el código fuente del paquete, ya que el fabricante Stereolabs proporciona un wrapper oficial completo, compatible con ROS2, que se encuentra disponible en su repositorio de GitHub [27].

Este paquete permite lanzar una o varias cámaras utilizando archivos `launch` configurables, donde es posible definir distintos nombres de nodo, tópicos, *namespaces* y parámetros individuales para cada dispositivo.

Aunque el uso típico es con una sola cámara, el paquete está diseñado para admitir múltiples instancias, lo que resulta especialmente útil en este proyecto, donde se han integrado tres unidades. Cada una se ejecuta en su propio *namespace* (`/zed1`, `/zed2`, `/zed3`), utilizando el mismo nodo base (`zed_wrapper`) pero con parámetros únicos que se definen mediante argumentos de lanzamiento (*launch arguments*) y plantillas en los archivos de lanzamiento (`.launch.py`).

El sistema puede reutilizar el mismo paquete base para múltiples cámaras, evitando así la necesidad de mantener versiones duplicadas del driver, colisiones de tópicos y facilitando la sincronización. La Tabla 3.3 resume la estructura del paquete oficial `zed-ros2-wrapper`, desarrollado por Stereolabs para integrar sus cámaras

ZED en sistemas ROS2 [27].

Carpeta	Descripción y función principal
<code>zed_wrapper/</code>	Nodo principal que gestiona la conexión con la cámara ZED 2i, la captura de imágenes, la generación del mapa de profundidad y la publicación de topics.
<code>zed_launch/</code>	Archivos <code>.launch.py</code> que permiten lanzar una o varias cámaras con parámetros personalizados. Facilita la configuración de múltiples instancias mediante argumentos.
<code>zed_components/</code>	Contiene definiciones internas de los nodos componibles (<i>component nodes</i>) y la lógica modular usada en ROS2.
<code>config/</code>	Archivos YAML que permiten configurar parámetros de resolución, frecuencia, <i>frame IDs</i> , activación de profundidad, odometría, etc.
<code>rviz/</code>	Configuraciones prediseñadas para RViz2 que permiten visualizar la imagen en color, el mapa de profundidad y la reconstrucción espacial en 3D.
<code>msg/</code> y <code>srv/</code>	Archivos que definen mensajes y servicios personalizados que amplían la funcionalidad más allá de los mensajes estándar de ROS2.

Tabla 3.3: Estructura del paquete `zed-ros2-wrapper` y función de sus carpetas principales.

El archivo de lanzamiento `zed_camera.launch.py` está diseñado para lanzar una única instancia de cámara ZED, que se puede reutilizar múltiples veces en paralelo. Para integrar varias cámaras ZED 2i en el sistema ROS2 sin modificar el código fuente del *wrapper* oficial, se ha seguido el ejemplo proporcionado por el fabricante en el repositorio `zed-multi-camera` [28]. Este ejemplo muestra cómo lanzar múltiples instancias del archivo `zed_camera.launch.py`, asignando a cada cámara un *namespace*, un contenedor y una configuración independiente. Cada cámara se identifica por su número de serie y se lanza como una instancia independiente, permitiendo la ejecución concurrente en hilos separados.

En este proyecto, para integrar tres cámaras ZED 2i, se ha creado un archivo adicional llamado `zed_multi_camera.launch.py` que realiza tres llamadas independientes al `zed_camera.launch.py`, cambiando los argumentos *camera_name*, *serial_number* y *namespace* para cada cámara. De esta forma, no se ha modificado el paquete original, logrando una integración multicámara estable y escalable.

3.3.3. Cámaras térmicas Seek Thermal Compact Pro.

En el caso de estas cámaras, el fabricante no proporciona soporte oficial para sistemas basados en Linux (como es el caso de Ubuntu 22.04) ni integración directa con

ROS2. Por tanto, se recurre a soluciones desarrolladas previamente y compartidas a través de repositorios de código abierto como GitHub.

Se emplea como base el paquete `seek_thermal_ROS2` desarrollado por Titouan Malgouyres [29], el cual proporciona un nodo básico para la publicación de imágenes térmicas mediante el uso de la biblioteca `libseek`.

No obstante, el paquete original presentaba ciertas limitaciones, especialmente relacionadas con el uso simultáneo de múltiples cámaras y su integración con el resto del sistema multisensorial del proyecto. Por este motivo, fue necesario realizar una serie de adaptaciones, entre las que destacan:

- Modificación del código para permitir la selección manual del dispositivo de vídeo correspondiente a cada cámara.
- Implementación de archivos `launch` en ROS2 que permiten ejecutar múltiples instancias del nodo, cada una asociada a una cámara distinta.

Estas adaptaciones son fundamentales para permitir el funcionamiento simultáneo y coordinado de las cuatro cámaras térmicas utilizadas en el sistema, cada una con su propio nodo publicador, integradas dentro de la arquitectura global. A continuación, se compara el código fuente original y el adaptado en este proyecto con el fin de analizar en detalle las modificaciones en el paquete `seek_thermal_ROS2`. Dicho análisis se centra específicamente en el archivo `seekDevice.cpp` dentro del repositorio original.

Este archivo contiene la programación necesaria para la inicialización del dispositivo Seek Thermal, la captura de imágenes térmicas y su publicación como mensajes en ROS2.

1. Se adapta el método `open()`.

El código original del Listado 3.1.

Listado 3.1: Declaración de la función `open()` de la clase `SeekDevice`.

```
bool SeekDevice::open()
```

Modificado al código del Listado 3.2.

Listado 3.2: Declaración de la función `open(int camera)` de la clase `SeekDevice`.

```
bool SeekDevice::open(int camera)
```

Permitiendo así elegir qué cámara abrir si hay varias cámaras conectadas, como en este proyecto.

2. Se realizan también cambios en `open_device()`.

Se modifica el código original del Listado 3.3.

Listado 3.3: Declaración de la función `open_device()` de la clase `SeekDevice`.

```
bool SeekDevice::open_device()
```

Dando lugar al código del Listado 3.4.

Listado 3.4: Declaración de la función `open_device(int camera)` de la clase `SeekDevice`.

```
bool SeekDevice::open_device(int camera)
```

Dentro de este:

- Se añade un contador `camera_count` para contar cuántas cámaras hay conectadas.
- Se recorre la lista de dispositivos USB con `libusb_get_device_list`.
- Se compara el `idVendor` y `idProduct` (igual que en el original).
- Si se encuentra una cámara, se imprime su dirección *bus* y *device*, tal y como se muestra en el Listado 3.5.

Listado 3.5: Mensaje de error al detectar una cámara conectada.

```
error("Camera found: BUS %03d DEVICE %03d\n",  
      bus, device);
```

- Si el número de la cámara encontrada coincide con el solicitado, se abre esa cámara, como se muestra en el Listado 3.6.

Listado 3.6: Apertura del dispositivo USB si el contador coincide.

```
if (camera_count == camera) {  
    res = libusb_open(...);  
}
```

Con esto se crea la posibilidad de elegir específicamente la cámara 1, 2, 3, etc. El código que se muestra en el Listado 3.7 avisa si hay varias cámaras conectadas.

Listado 3.7: Mensaje ante la detección de múltiples cámaras.

```
if (camera_count > 1) {  
    printf("Multiple cameras detected: %d\n",  
          camera_count);  
}
```

La forma en que se abre, configura y usa cada cámara sigue siendo la misma, al igual que el control de errores y los mensajes por consola.

Gracias a estos cambios, ahora se puede elegir qué cámara abrir, usar varias cámaras conectadas al mismo tiempo y mostrar un aviso cuando se detectan varias.

3. Modificaciones en launch.

Es fundamental analizar también el archivo `launch`, ya que estos han sido modificados para permitir la ejecución simultánea de múltiples instancias del nodo, cada una asociada a una cámara térmica diferente. Se trata de modificaciones clave para que el sistema pueda lanzar automáticamente todas las cámaras térmicas y publicar sus datos sin conflictos, asegurando su integración fluida en el entorno ROS2. Se presenta en el Listado 3.8 el código original.

Listado 3.8: Archivo launch para ejecutar el nodo de la cámara Seek Thermal.

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='seekthermal_camera',
            executable='seekthermal_node',
            output='screen'),
    ])
```

Y, a continuación, en el Listado 3.9, su modificación para este proyecto.

Listado 3.9: Lanzamiento de múltiples nodos para cámaras térmicas Seek.

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription([
        Node(
            package="seekthermal_camera",
            executable="seekthermal_node",
            output='screen',
            name="thermal_camera_1",
            parameters=[
                {"camera": 1},
                {"topic": "/thermal_cameras/camera1"}
            ],
        ),
        Node(
            package="seekthermal_camera",
            executable="seekthermal_node",
            output='screen',
            name="thermal_camera_2",
            parameters=[
                {"camera": 2},
                {"topic": "/thermal_cameras/camera2"}
            ],
        ),
    ])
```

```

    ),
    Node(
        package="seekthermal_camera",
        executable="seekthermal_node",
        output='screen',
        name="thermal_camera_3",
        parameters=[
            {"camera": 3},
            {"topic": "/thermal_cameras/camera3"}
        ],
    ),
    Node(
        package="seekthermal_camera",
        executable="seekthermal_node",
        output='screen',
        name="thermal_camera_4",
        parameters=[
            {"camera": 4},
            {"topic": "/thermal_cameras/camera4"}
        ],
    ),
])

```

En el código original, se lanza una única instancia del nodo `seekthermal_node` sin parámetros adicionales, lo que limita el sistema al uso de una sola cámara. En cambio, en la versión modificada, se lanzan múltiples instancias del mismo nodo, una por cada cámara térmica.

3.3.4. Integración conjunta.

Partiendo de los paquetes individuales de cada sensor, se ha creado un paquete fusión que lanza simultáneamente dichos paquetes, permitiendo su funcionamiento en conjunto y la posibilidad de captar nubes de puntos (sensor LiDAR) e imágenes (estéreo y térmicas).

Para ello, se genera el archivo de configuración o `launch` dentro de la carpeta `premove_sensors`, consiguiendo así un entorno donde el sistema multisensorial funcione de forma coordinada. El código que se muestra en el Listado 3.10 es el encargado de dicha fusión.

Listado 3.10: Archivo `launch` que lanza todos los sensores.

```

from launch import LaunchDescription
from launch.actions import IncludeLaunchDescription
from launch.launch_description_sources import
    PythonLaunchDescriptionSource
from launch_ros.actions import Node
import os

```

```

def generate_launch_description():
    # Ruta base de los archivos instalados
    seekthermal_launch_file = os.path.join(
        '/home/premove/ros2_ws/install/seekthermal_camera/'
        'share/seekthermal_camera/launch',
        'seekthermal_camera.launch.py'
    )
    zed_multi_camera_launch_file = os.path.join(
        '/home/premove/ros2_ws/install/zed_multi_camera/share/'
        'zed_multi_camera/launch',
        'zed_multi_camera.launch.py'
    )
    rslidar_sdk_launch_file = os.path.join(
        '/home/premove/ros2_ws/install/rslidar_sdk/share/'
        'rslidar_sdk/launch',
        'start.py'
    )

    return LaunchDescription([
        IncludeLaunchDescription(
            PythonLaunchDescriptionSource(
                seekthermal_launch_file),
        ),
        IncludeLaunchDescription(
            PythonLaunchDescriptionSource(
                zed_multi_camera_launch_file),
            launch_arguments={
                'cam_names': '[stereo_1, stereo_2, stereo_3]',
                'cam_models': '[zed2i, zed2i, zed2i]',
                'cam_serials': '[38215876, 36559102, 31752731]'
            }.items(),
        ),
        IncludeLaunchDescription(
            PythonLaunchDescriptionSource(
                rslidar_sdk_launch_file),
        ),
    ])

```

Este archivo `launch` aúna los ficheros de cada uno de los paquetes: térmico, estéreo y LiDAR. Utiliza `IncludeLaunchDescription` para incorporar los archivos `launch` individuales ya instalados en sus respectivos paquetes (`seekthermal_camera`, `zed_multi_camera`, `rslidar_sdk`), a los que además se les pasan argumentos específicos para las cámaras estéreo, como nombres, modelos y números de serie, permitiendo su identificación individual.

Capítulo 4

Resultados

Contenido

4.1	Introducción.	33
4.2	Soportes y carcasas de los sensores.	33
4.2.1	Soporte para el sensor LiDAR.	33
4.2.2	Soporte para la cámara estéreo.	34
4.2.3	Soporte para la cámara térmica.	35
4.3	Nube de puntos e imágenes obtenidas a través de los sensores	38

4.1. Introducción.

Como resultado del trabajo realizado y en lo referente a la parte de diseño, en este capítulo se muestran las piezas ya impresas y listas para su integración en el sistema. Por otro lado, desde el punto de vista del software, se incluye una figura representativa que ilustra la visualización de las imágenes captadas por el sistema de sensores.

4.2. Soportes y carcasas de los sensores.

4.2.1. Soporte para el sensor LiDAR.

El resultado de la fabricación del soporte correspondiente al sensor LiDAR, el cual fue diseñado específicamente para garantizar su estabilidad y correcta fijación sobre la plataforma, se presenta en la Figura 4.1.

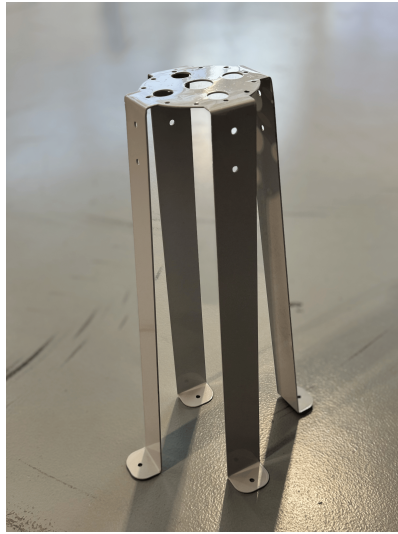


Figura 4.1: Soporte del sensor LiDAR fabricado en acero.

4.2.2. Soporte para la cámara estéreo.

Del mismo modo, el resultado para el soporte de la cámara estéreo ZED 2i, teniendo en cuenta sus dimensiones, puntos de sujeción y el campo visual necesario se muestra en las Figuras 4.2a y 4.2b.



(a) Vista frontal del soporte impreso para la ZED 2i.

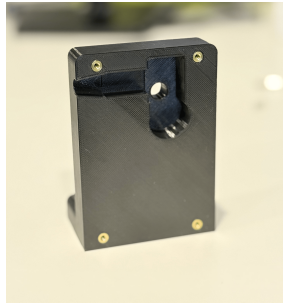


(b) Vista lateral del soporte con cámara incorporada.

Figura 4.2: Soporte de la cámara estéreo ZED 2i impreso en 3D.

4.2.3. Soporte para la cámara térmica.

En cuanto al soporte para la cámara térmica, la Figura 4.3 ilustra distintos ángulos de este una vez impreso en 3D.



(a) Carcasa.



(b) Carcasa con la cámara.



(c) Conjunto completo.

Figura 4.3: Soporte impreso para la cámara térmica *Seek Thermal Compact Pro*.

Una vez han sido fabricadas e impresas las piezas, y habiéndose montado y atornillado los sensores en cada una de ellas, se procede al ensamblaje de todo el conjunto en la plataforma metálica mencionada y mostrada anteriormente en la Figura 2.14 del Capítulo 2. Esto da como resultado las Figuras 4.4 y 4.5 .

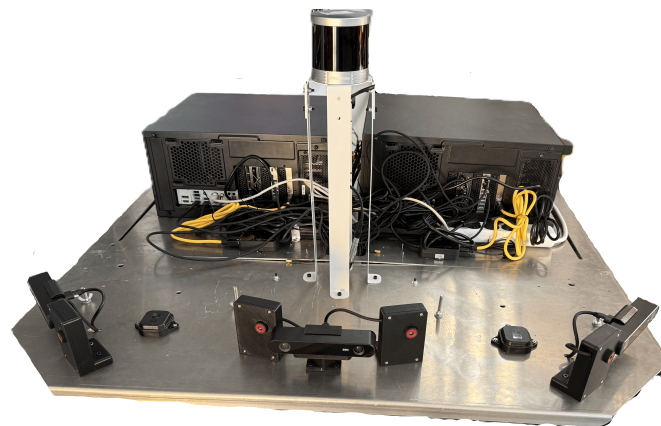


Figura 4.4: Vista frontal de la plataforma metálica con los sensores y sus soportes.



Figura 4.5: Vista lateral de la plataforma metálica con los sensores y sus soportes.

En ellas se puede apreciar el sistema multisensorial buscado en este proyecto. Además, cabe destacar que en dichas imágenes se observa cómo en la plataforma han de colocarse diversos dispositivos añadidos a estos sensores. Estos elementos se suman al LiDAR y a las cámaras con el objetivo de que el funcionamiento del sistema y la gestión de los datos se puedan llevar a cabo. Se trata de ordenadores, alimentación, cableado, etc.

En las Figuras 4.6 y 4.7 puede comprobarse que se diseñó una especie de chasis o cubierta para la protección de los dispositivos que se suman a los sensores. Sin embargo, este diseño también quedó fuera de este Trabajo Fin de Grado, siendo realizado, al igual que la plataforma de acero, por otro integrante del proyecto PREMOVE.

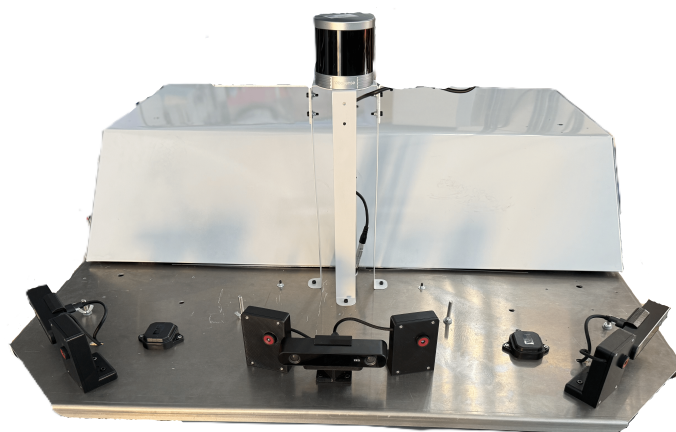


Figura 4.6: Vista frontal de la plataforma metálica con los sensores y sus soportes y la cubierta.

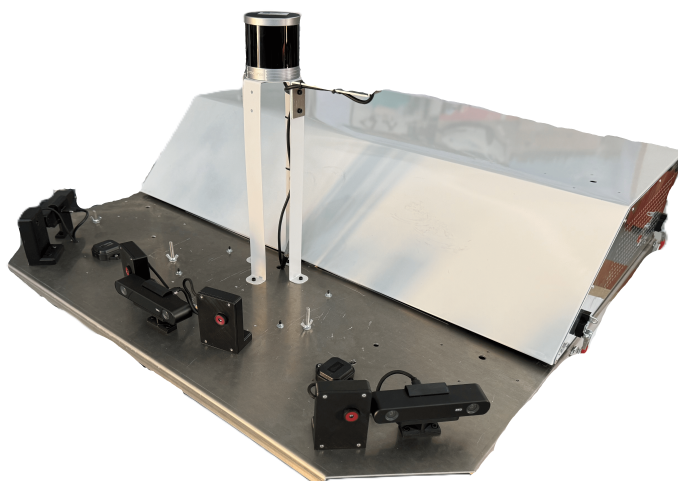


Figura 4.7: Vista lateral derecha de la plataforma metálica con los sensores y sus soportes y la cubierta.

A continuación, se comprueba el resultado de la disposición elegida y se procede a probar el funcionamiento interno (adquisición de imágenes y nube de puntos) del sistema.

4.3. Nube de puntos e imágenes obtenidas a través de los sensores

En lo relativo al apartado de *software*, se ha verificado la correcta adquisición y visualización de datos mediante ROS2. Para ello, se empleó la herramienta RViz, que muestra todos los sensores integrados en el sistema capturando información de forma simultánea. La Figura 4.8 muestra las imágenes a color proporcionadas por las cámaras estéreo ZED 2i, fundamentales para la percepción visual y la reconstrucción de la escena.



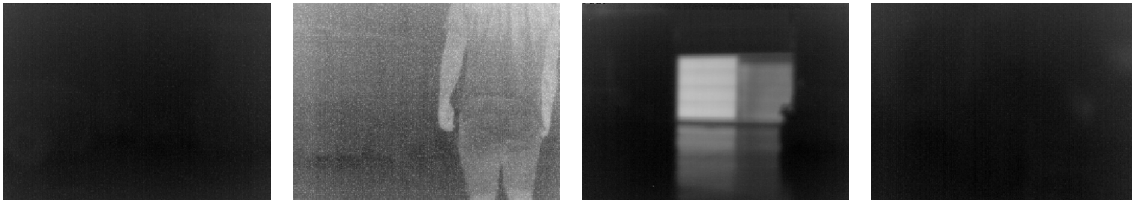
(a) ZED 2i Izquierda.

(b) ZED 2i Central.

(c) ZED 2i Derecha.

Figura 4.8: Ejemplos de visualización en RViz para las cámaras estéreo.

Además, se presenta la Figura 4.9, donde se observan las secuencias térmicas captadas por las cámaras Seek Thermal Compact Pro, que permiten detectar presencia y temperatura relativa.



(a) Cámara térmica Izquierda.

(b) Cámara térmica Central Izquierda.

(c) Cámara térmica Central Derecha.

(d) Cámara térmica Derecha.

Figura 4.9: Ejemplos de visualización en RViz para las cámaras térmicas.

Finalmente, se muestra la Figura 4.10, donde se representa la información tridimensional generada por el sensor LiDAR 3D Robosense Helios 5515, visualizada mediante una nube de puntos.

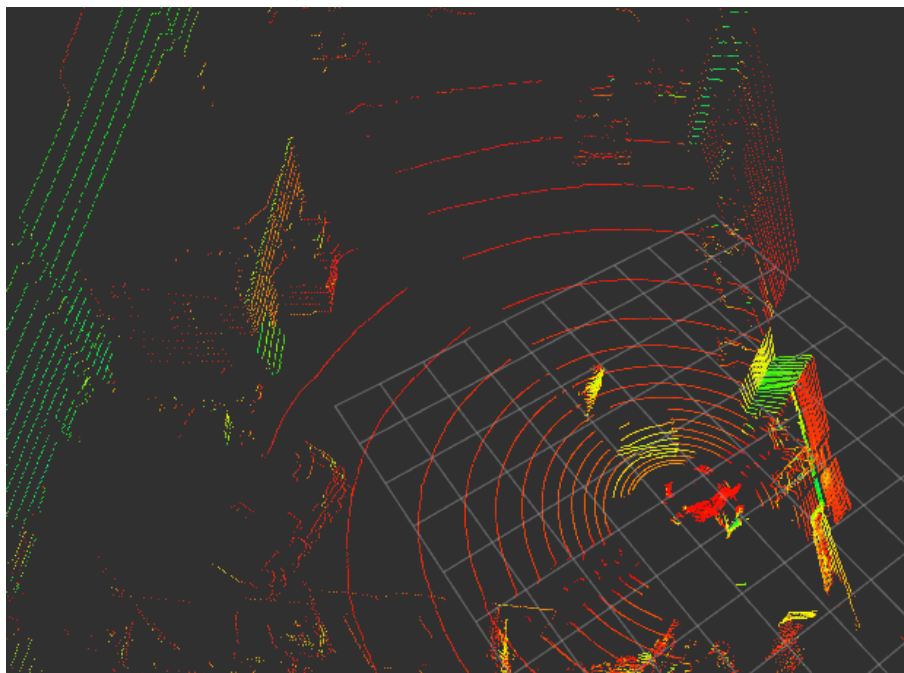


Figura 4.10: Ejemplos de visualización en RViz para sensor LiDAR 3D.

Esta visualización integrada confirma el correcto funcionamiento simultáneo de todos los sensores dentro del entorno ROS2.

Capítulo 5

Conclusiones y líneas futuras

Contenido

5.1	Introducción.	41
5.2	Conclusiones.	41
5.3	Líneas futuras.	42

5.1. Introducción.

En este último capítulo se presentan las conclusiones extraídas a lo largo del desarrollo del trabajo, así como una reflexión sobre los resultados obtenidos. Además, se proponen posibles líneas futuras que permitirían ampliar, mejorar o aplicar los conocimientos adquiridos en el presente proyecto a contextos más amplios dentro del ámbito de la robótica móvil y la conducción autónoma.

5.2. Conclusiones.

El objetivo principal de este Trabajo de Fin de Grado era desarrollar una plataforma sensorial robusta, capaz de captar imágenes y datos del entorno de un vehículo mediante varios sensores trabajando de forma coordinada. El propósito era garantizar que, ante la posible pérdida de visibilidad de alguno de los sensores, los demás continuaran aportando información útil. Este objetivo se ha alcanzado satisfactoriamente, y el sistema resultante ha demostrado un funcionamiento estable y fiable.

Durante el desarrollo del proyecto, se han diseñado y fabricado soluciones mecánicas específicas que aseguran una fijación adecuada y la protección de todos los componentes. Se han conseguido implementar estructuras funcionales mediante el

uso de SolidWorks, lo que ha permitido integrar correctamente todos los sensores sobre la plataforma del vehículo.

A nivel de software, se ha logrado configurar un entorno de trabajo completo en Ubuntu 22.04 con ROS2, integrando con éxito sensores comerciales térmicos, estéreo y LiDAR. Se han solucionado diversos problemas de compatibilidad entre ellos y se han adquirido competencias clave para el desarrollo de sistemas autónomos. Se ha logrado construir un sistema funcional, capaz de ser ampliado y adaptado a distintos escenarios reales.

Este proyecto ha permitido sentar las bases de un sistema multisensorial modular, sobre el cual ya se están realizando nuevos desarrollos. Todo lo aprendido durante el proceso no solo ha facilitado el cumplimiento de los objetivos planteados, sino que ha supuesto una experiencia formativa intensa y enriquecedora, con gran valor personal y profesional. Sin duda, estos conocimientos y capacidades serán fundamentales en futuros trabajos en el ámbito de la robótica móvil y los vehículos autónomos.

5.3. Líneas futuras.

A continuación, se muestran las posibles líneas futuras del trabajo.

- Calibración sensorial. Será necesario realizar la calibración intrínseca y extrínseca de los sensores involucrados, con el fin de mejorar la calidad de los datos y la coherencia espacial entre las distintas modalidades sensoriales, lo que resulta fundamental para una fusión de datos eficaz.
- Integración completa en vehículo y generación de un dataset multimodal. Montaje definitivo de la plataforma sensorial sobre un vehículo de pruebas, con el objetivo de captar y registrar datos reales del entorno. Se plantea la creación de un conjunto de datos multimodal que incluya imágenes RGB, imágenes térmicas y nubes de puntos obtenidas mediante LiDAR.
- Entrenamiento de redes neuronales con los datos adquiridos. Una vez generado el *dataset*, se propone su uso para entrenar modelos de inteligencia artificial, orientados a tareas de percepción como la detección y el seguimiento de participantes del tráfico (peatones, ciclistas, vehículos, etc.), contribuyendo así al desarrollo de sistemas avanzados de asistencia a la conducción.

Parte I

Apéndices

Apéndice A

Manual de Instalación

Contenido

A.1 Marco Introductorio.	45
A.2 ROS2 en Ubuntu 22.04.	45
A.3 Instalación de Paquetes.	47
A.3.1 Sensor LiDAR 3D Robosense Helios 5515.	47
A.3.2 Cámara estéreo ZED 2i.	48
A.3.3 Cámara térmica Seek Thermal Compact Pro.	49

A.1. Marco Introductorio.

A lo largo de este trabajo, ha sido necesaria la utilización de diversos paquetes. Algunos de ellos se han usado sin modificaciones, tal y como se presentaban en repositorios de código abierto, y otros han sido modificados para adaptarlos a los requisitos de funcionamiento de este proyecto.

Todos ellos están destinados a usarse con el *middleware* ROS2 bajo el sistema operativo Ubuntu 22.04. Es por ello que antes de proceder, ha de implementarse ROS2 Humble y trabajar en dicho entorno. En este apéndice se recogen los pasos a seguir para la instalación de los distintos paquetes necesarios para el TFG.

A.2. ROS2 en Ubuntu 22.04.

Para la gestión de entornos y paquetes existe una herramienta llamada Conda, empleada para : crear entornos con versiones específicas de Python y bibliotecas, evitando conflictos entre proyectos que requieren diferentes versiones de paquetes. Esta herramienta funciona en múltiples plataformas, siendo compatible con Linux, macOS y Windows.

En relación a esto, también existe Miniforge, una distribución ligera de Conda de gran utilidad para crear entornos aislados sin interferir con el sistema base. Y, en el mismo contexto, se tiene Mamba, que actúa como reemplazo de Conda, pero mucho más rápido resolviendo dependencias, compatible con Miniforge y con los mismos comandos que utiliza.

ROS2 tiene muchas dependencias específicas, por lo que usar las herramientas mencionadas anteriormente permite crear entornos reproducibles, limpios y separados para distintos proyectos. En este trabajo se propone instalar la distribución base de Miniforge y tras ello, instalar Mamba para acelerar la gestión de paquetes con el fin de utilizar ROS2 de manera eficiente.

Instalación de Miniforge.

Se ha de descargar el paquete del repositorio oficial de Github y proceder a su instalación en el sistema Linux [30]. Esto se realiza ejecutando los comandos del Listado A.1 que se muestran a continuación.

Listado A.1: Comando para descarga e instalación de Miniforge.

```
curl -L -O "https://github.com/conda-forge/miniforge/releases/
latest/download/Miniforge3-Linux-x86_64.sh"
bash Miniforge3-Linux-x86_64.sh
```

Instalación de Mamba.

Tras esto, se procede a instalar Mamba, empleando para ello los comandos mostrados en el Listado A.2.

Listado A.2: Comando para instalación de Mamba.

```
conda install mamba -c conda-forge
```

Paquete de ROS2.

Con los comandos mostrados en el Listado A.3, se genera un entorno llamado `premove_sensors`.

Listado A.3: Comandos para generar y activar un entorno para ROS2.

```
mamba create -n premove_sensors python=3.10
mamba init
mamba activate premove_sensors
conda config --env --add channels conda-forge
conda config --env --add channels robostack-staging
conda config --env --remove channels defaults
```

En el entorno `premove_sensors` se puede instalar ahora ROS2. Para esto, se hace uso del comando que se presenta en el Listado A.4.

Listado A.4: Comando para instalar ROS2.

```
mamba install ros-humble-desktop pocl
```

Con los comandos mostrados en el Listado A.5, se comprueba ahora la instalación ejecutando RViz2.

Listado A.5: Comandos para desactivar y reactivar el entorno.

```
mamba deactivate
mamba activate premove_sensors
rviz2
```

Para activar el entorno `premove_sensors` automáticamente al abrir un terminal, se añade el comando del Listado A.6 al final del fichero `/.bashrc`.

Listado A.6: Comando para añadir al fichero `.bashrc`.

```
mamba activate premove_sensors
```

A.3. Instalación de Paquetes.

En este apartado se describe la instalación de cada uno de los paquetes correspondientes a los diferentes sensores utilizados en este proyecto.

A.3.1. Sensor LiDAR 3D Robosense Helios 5515.

En primer lugar, para obtener datos del LiDAR 3D, es necesario descargar el paquete `rslidar_sdk` del repositorio oficial del fabricante en GitHub [26].

Es importante destacar que para integrar el driver del LiDAR en este proyecto, es necesario usar `rs_driver`. Este se puede descargar directamente de GitHub, ejecutando en la consola del sistema los comandos del Listado A.7 que se presentan a continuación.

Listado A.7: Comandos para clonar el SDK de Robosense y sus submódulos.

```
git clone https://github.com/RoboSense-LiDAR/rslidar_sdk.git
cd rslidar_sdk
git submodule init
git submodule update
```

Con el entorno de ROS2 ya descargado y preparado previamente (en Ubuntu 22.04), es necesario descargar las librerías `Yaml` y `libpcap`. Esto se realiza ejecutando los comandos que se muestran en el Listado A.8

Listado A.8: Comandos para actualizar el sistema e instalar las librerías `Yaml` y `libpcap`.

```
sudo apt-get update
sudo apt-get install -y libyaml-cpp-dev
sudo apt-get install -y libpcap-dev
```

Si se usa ROS2 se ha de modificar el archivo `package.xml` y reemplazar su contenido por los de `package_ros2.xml`. Para especificar el tipo de LiDAR a usar, se modifica el archivo `config/config.yaml` cambiando el tipo de sensor al Helios 5515 (RSHELIOS). Es importante mencionar que se ha de obtener la dirección del LiDAR entrando a 192.168.1.200 e incluirla en la IPv4 del equipo.

Para la compilación a través de Colcon se procede de la siguiente manera:

1. Se crea una carpeta `workspace` y a su vez una carpeta `src` en ella, y se guarda el proyecto `rslidar_sdk`.
2. Se descarga el proyecto de definición de paquetes en ROS2 a través del enlace encontrado en GitHub [31] , y se guarda `rslidar_msg` dentro de la carpeta `src` anteriormente creada.
3. En el `workspace` creado anteriormente, se ejecutan los comandos del Listado A.9 para compilar y ejecutar.

Listado A.9: Comandos para compilar y lanzar el nodo del sensor LiDAR Robosense.

```
colcon build
source install/setup.bash
ros2 launch rslidar_sdk start.py
```

A.3.2. Cámara estéreo ZED 2i.

Para el uso de las cámaras estéreo se necesita del paquete `zed-ros2-wrapper` proporcionado por Stereolabs en GitHub [27]. Los prerequisites para su funcionamiento son los siguientes:

1. Instalación de CUDA (*Compute Unified Device Architecture*), una plataforma de computación paralela creada por NVIDIA que permite usar la GPU (*Graphics Processing Unit*) (tarjeta gráfica) no solo para gráficos, sino también para realizar cálculos complejos y rápidos.[32].

En este caso, se selecciona como sistema operativo Linux, arquitectura x86_64, distribución Ubuntu, versión 22.04 e instalador tipo *runfile*. Se ejecutan los comando que se muestran en el Listado A.10.

Listado A.10: Comandos para descargar e instalar CUDA 12.9.1.

```
wget https://developer.download.nvidia.com/compute/cuda
/12.9.1/local_installers/cuda_12.9.1_575.57.08_linux.
run
sudo sh cuda_12.9.1_575.57.08_linux.run
```

2. Instalación de ZED SDK [33] [34]. Esto se realiza ejecutando los comandos del Listado A.11.

Listado A.11: Comandos para instalar ZED SDK v4.0.0 en Ubuntu 22.04 con CUDA 11.8

```
cd path/to/download/folder
sudo apt install zstd
chmod +x ZED_SDK_Ubuntu22_cuda12.9_v4.0.0.zstd.run
./ZED_SDK_Ubuntu22_cuda12.9_v4.0.0.zstd.run
```

3. Instalación de ZED ROS2 wrapper. Tras configurar correctamente todas estas dependencias, se siguen los pasos descritos en la página oficial de GitHub [27] y finalmente solo es necesario conectar las cámaras al puerto USB. Se han de ejecutar los comandos del Listado A.12.

Listado A.12: Comandos para clonar e instalar zed-ros2-wrapper en un workspace ROS2

```
cd ~/ros2_ws/src/
git clone https://github.com/stereolabs/zed-ros2-wrapper.git
cd ..
sudo apt update
rosdep update
rosdep install --from-paths src --ignore-src -r -y
colcon build --symlink-install --cmake-args=-
    DCMMAKE_BUILD_TYPE=Release --parallel-workers $(nproc)
echo source $(pwd)/install/local_setup.bash >> ~/.bashrc
source ~/.bashrc
```

4. Instalación del paquete zed-ros2-examples, para el launch multi cámara. Se trata de una colección de paquetes colcon. Se ha de clonar el repositorio, actualizar las dependencias y compilar el paquete. Esto se realiza ejecutando los comandos del Listado A.13.

Listado A.13: Clonado del repositorio de ejemplos de ZED ROS2 y compilación en el workspace.

```
cd ~/ros2_ws/src/
git clone https://github.com/stereolabs/zed-ros2-examples.git
cd ../
sudo apt update
rosdep update
rosdep install --from-paths src --ignore-src -r -y
colcon build --symlink-install --cmake-args=-DCMAKE_BUILD_TYPE=
    Release
source ~/.bashrc
```

A.3.3. Cámara térmica Seek Thermal Compact Pro.

En este caso, se utiliza el paquete disponible en GitHub titulado `seek_thermal_ROS2` que proporciona un nodo de ROS2 que permite acceder a los datos de la cámara

térmica a través de su interfaz USB, generando imágenes en línea [29].

1. Se instala Open CV3 ejecutando el comando del Listado A.14 [35].

Listado A.14: Comando para instalar OpenCV 3.2 con los módulos *contrib*.

```
wget -O - https://gist.githubusercontent.com/syneart/3e6bb68de8b6390d2eb18bff67767dcb/raw/OpenCV3.2withContrib.sh | bash
```

2. Se descarga la librería `libusb` con la línea de comando mostrada en Listado A.15.

Listado A.15: Comando para instalar la librería `libusb` en Ubuntu.

```
sudo apt install libusb-1.0-0-dev
```

3. Se añade la librería llamada `seek_thermal` y el paquete de ROS2 llamado `seekthermal_camera` a la carpeta `src` del `workspace` de ROS2. Se ejecutan para ello los comandos de Listado A.16.

Listado A.16: Clonado de un paquete dentro del espacio de trabajo de ROS2.

```
cd ~/ros2_ws/src
git clone https://github.com/Titou2222/seek_thermal_ROS2?tab=readme-ov-file
```

4. Se compila la librería con Cmake ejecutando los comandos del Listado A.17.

Listado A.17: Comandos para compilar e instalar la librería `seekthermal`.

```
cd ~/ros2_ws/src/seekthermal/build
cmake ..
make
sudo make install
```

5. Se compila ahora con Colcon usando el comando del Listado A.18.

Listado A.18: Compilación del paquete `seek_thermal` con `colcon`.

```
colcon build --packages-select seek_thermal
```

6. Se compila el paquete de ROS2 con los comando del Listado A.19.

Listado A.19: Fuente del entorno e instalación del paquete `seekthermal_camera`.

```
source install/setup.bash
colcon build --packages-select seekthermal_camera
```

7. Para poder ejecutar el driver es necesario establecer los permisos adecuados para el dispositivo de la cámara. Esto puede hacerse automáticamente copiando el archivo `config/99-seeckthermal.rules` de de este repositorio a la ruta `/etc/udev/rules.d` y ejecutando los comandos del Listado A.20.

Listado A.20: Recarga de reglas udev para permitir el uso de la cámara sin permisos.

```
sudo udevadm control --reload-rules
sudo udevadm trigger
```

El paquete encargado del funcionamiento de las cámaras térmicas fue originalmente creado para ROS1, por lo que se han de realizar las modificaciones necesarias para su portabilidad a ROS2. En el caso de que durante el proceso de instalación surjan errores relacionados con dependencias como OpenCV3, CUDA, libjasper-dev, el uso excesivo de hilos en la compilación (`make -j4`) o problemas específicos en archivos `CMakeLists.txt`, es recomendable consultar el repositorio original de este paquete, donde se presentan algunas soluciones posibles [29].

Apéndice B

Manual de Usuario.

Contenido

B.1	Sensor LiDAR 3D Robosense Helios 5515.	53
B.2	Cámaras estéreo ZED 2i.	53
B.3	Cámaras térmicas Seek Thermal Compact Pro.	54
B.4	Sistema de sensores general.	54

A continuación se detallan los comandos a ejecutar para el funcionamiento del sistema, una vez ya instalado el entorno y los diversos paquetes necesarios.

B.1. Sensor LiDAR 3D Robosense Helios 5515.

Para ejecutar el nodo del sensor LiDAR, se abre el terminal del sistema, se carga el entorno de ROS2 y se ejecutan los comandos del Listado B.1.

Listado B.1: Lanzamiento del nodo del sensor LiDAR en ROS 2.

```
source install/setup.bash
ros2 launch rslidar_sdk start.py
```

B.2. Cámaras estéreo ZED 2i.

Para ejecutar el nodo correspondiente a las cámaras estéreo, se abre el terminal del sistema, se carga el entorno de ROS2 y se ejecutan los comandos del Listado B.2.

Listado B.2: Lanzamiento del nodo de las cámaras estéreo ZED 2i en ROS 2.

```
source install/setup.bash
ros2 launch zed_wrapper zed2i.launch.py
```

B.3. Cámaras térmicas Seek Thermal Compact Pro.

Se abre el terminal, se configura el entorno ROS2 y se lanza el nodo específico para las cámaras térmicas con los comandos mostrados en Listado B.3.

Listado B.3: Lanzamiento del nodo de las cámaras térmicas Seek Thermal en ROS 2.

```
source install/setup.bash
ros2 launch seekthermal_camera seekthermalcamera.launch.py
```

B.4. Sistema de sensores general.

Se abre el terminal, se configura el entorno ROS2 y se lanza el nodo específico para todo el sistema. Se trata del *launch* que se encuentra en `premove_sensors` y que integra todos los sensores funcionando al mismo tiempo. Esto se realiza al ejecutar los comandos mostrados en Listado B.4.

Listado B.4: Lanzamiento del nodo general del sistema multisensorial.

```
source install/setup.bash
ros2 launch premove_sensors premove_sensors.launch
```

Para la visualización, tras la puesta en marcha de cada nodo, se puede ejecutar el comando del Listado B.5.

Listado B.5: Visualización de los datos con rviz2.

```
rviz2
```

Apéndice C

Planos de diseño en CAD

Contenido

C.1	Introducción.	55
C.2	Disposición a nivel general.	55
C.3	Campo de Visión.	56
C.3.1	Sensor LiDAR 3D.	56
C.3.2	Cámaras estéreo.	56
C.3.3	Cámaras térmicas.	56
C.4	Distribución en la plataforma.	57
C.5	Dimensiones de los diseños.	57
C.5.1	Soporte Sensor LiDAR 3D.	57
C.5.2	Soporte Cámaras estéreo.	57
C.5.3	Carcasa Cámaras térmicas.	57

C.1. Introducción.

En el apéndice que se presenta a continuación se recogen una serie de planos que explican, de forma visual, la distribución de los diferentes elementos que componen el diseño. La información se organiza de manera progresiva: desde una vista general de toda la plataforma, pasando por la distribución de los sensores sobre la estructura metálica, hasta llegar al detalle dimensional de cada uno de los soportes por separado.

C.2. Disposición a nivel general.

Se incluye un plano con la vista general de la distribución de los sensores, el cual permite visualizar de forma clara el aspecto global del diseño una vez montado sobre

la plataforma metálica y, a su vez, integrado en el vehículo. Esta vista resulta útil como referencia inicial para comprender la disposición relativa de cada uno de los elementos y su integración en el conjunto del sistema multisensorial. Se anexa a esta memoria bajo el título de `101_Ensamblaje_Coche_Vista_General`

C.3. Campo de Visión.

A continuación, se presenta de forma general el campo de visión asociado a cada tipo de sensor. Para ello, se incluyen una serie de planos individuales en los que se representa cada sensor por separado, considerando el número total de dispositivos utilizados para cada tecnología.

C.3.1. Sensor LiDAR 3D.

En el caso de este sensor, se adjunta el plano `102_LiDAR_3D_Campo_de_visión`, en el que se representa su campo de visión con una cobertura de 360° en el plano horizontal, lo que le permite generar una nube de puntos completa mediante el barrido láser.

C.3.2. Cámaras estéreo.

A continuación, se presenta la disposición de las tres cámaras estéreo junto con el soporte diseñado específicamente para su montaje. La representación gráfica incluye también el campo de visión estimado para cada unidad, modelado mediante conos proyectados desde las lentes en SolidWorks, lo que permite visualizar de forma aproximada su alcance y cobertura espacial al funcionar todos en conjunto. Dicho plano se titula `103_ZED_2i_Campo_de_visión`.

C.3.3. Cámaras térmicas.

Finalmente, se lleva a cabo la misma representación para el caso de las cámaras térmicas. En el documento correspondiente, `104_Seek_Thermal_Campo_de_visión`, se observa la configuración de las cuatro unidades utilizadas, cada una integrada en su respectiva carcasa protectora, que actúa simultáneamente como soporte estructural. Esta vista permite apreciar tanto su disposición espacial como el campo de visión proyectado desde cada dispositivo.

C.4. Distribución en la plataforma.

En el plano titulado `105_Sensores_Distribución_Plataforma` se recoge la distribución completa de todos los sensores sobre la estructura metálica (en sus respectivos soportes). El objetivo de este documento es mostrar, de forma precisa y acotada, la ubicación que ocupará cada dispositivo en el montaje final. Se detallan las posiciones relativas entre sensores, respetando criterios de simetría, cobertura y funcionalidad, lo que permite anticipar su disposición exacta durante la fase de integración física.

C.5. Dimensiones de los diseños.

En esta última parte del anexo se recogen las dimensiones correspondientes a los distintos diseños desarrollados para el sistema. Se incluyen los dibujos acotados de cada uno de ellos, con el objetivo de definir con precisión las medidas necesarias para su correcta fabricación y ensamblaje sobre la plataforma.

C.5.1. Soporte Sensor LiDAR 3D.

El plano `106_LiDAR_3D_Dimensiones` recoge las cotas detalladas del soporte diseñado para alojar el sensor LiDAR Robosense Helios 5515. Este dibujo permite visualizar con precisión las dimensiones clave del soporte, incluyendo longitudes, alturas y otras medidas necesarias para su integración estructural. El montaje del sensor exige una posición fija y elevada que garantice una cobertura completa de 360°, sin tener interferencias de los otros sensores.

C.5.2. Soporte Cámaras estéreo.

El plano `107_ZED_2i_Dimensiones` presenta las dimensiones del soporte diseñado para las cámaras estéreo ZED 2i. En él se recogen las medidas funcionales que permiten garantizar una sujeción precisa del dispositivo y asegurar su orientación correcta, en conjunto con las demás cámaras estéreo y el resto de sensores.

C.5.3. Carcasa Cámaras térmicas.

Las dimensiones de la carcasa diseñada para las cámaras térmicas Seek Thermal Compact Pro se recogen en el plano `108_Seek_Thermal_Dimensiones`, que incluye vistas generales del ensamblaje, con las cotas principales del cuerpo de la carcasa y de la tapa. Además, se presenta una vista de explosión a modo de referencia del modelo.

En la Tabla C.1, que se muestra a continuación, se recopilan, a modo de resumen, todos los planos técnicos incluidos en este anexo. Se indica en ellos el número de hoja, el título del plano, la escala utilizada y el formato del dibujo. Esta tabla sirve como índice gráfico para localizar, de manera rápida y clara, los diferentes documentos técnicos desarrollados a lo largo del proyecto.

Plano	Título	Escala	Formato
101	101_Ensamblaje_Coche_Vista_General	1:20	A3
102	102_LiDAR_3D_Campo_de_visión	1:20 y 1:50	A3
103	103_ZED_2i_Campo_de_visión	1:50	A3
104	104_Seek_Thermal_Campo_de_visión	1:50	A3
105	105_Sensores_Distribución_Plataforma	1:10	A3
106	106_LiDAR_3D_Dimensiones	1:2 y 1:5	A3
107	107_ZED_2i_Dimensiones	1:1 Y 1:2	A3
108	108_Seek_Thermal_Dimensiones	1:2 y 1:1	A3

Tabla C.1: Listado de planos técnicos incluidos en el Anexo B.

Bibliografía

- [1] SAE International. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles, April 2021. SAE Standard J3016_202104.
- [2] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. In *International Journal of Robotics Research (IJRR)*, 2013. Accessed: 2025-06-04.
- [3] Giorgio L. Foresti and Carlo S. Regazzoni. Multisensor data fusion for autonomous vehicle navigation in risky environments. *IEEE Transactions on Vehicular Technology*, 51(5):1165–1185, 2002.
- [4] SAE International. Sae j3016™: Levels of driving automation are defined in new update. <https://www.sae.org/blog/sae-j3016-update>, 2021. Blog post. Accessed: 2025-06-04.
- [5] ResearchGate. Figura 2: Niveles de automatización de un coche propuestos por la sae (2018), 2021. Imagen extraída de ResearchGate. Accessed: 2025-06-04.
- [6] Europeos Viajeros. Coches autónomos: tipos, niveles de autonomía y funcionamiento. <https://www.europeosviajeros.com/articulos/coches-autonomos/>, 2022. Artículo divulgativo. Accessed: 2025-06-04.
- [7] Universidad de Málaga - Grupo de Robótica y Mecatrónica. Proyecto premove. <https://www.uma.es/robotics-and-mechatronics/info/138109/premove/>, 2023. Accessed: 2025-06-04.
- [8] OpenTopography. Lidar basics. https://opentopography.org/lidar_basics, 2023. Accessed: 2025-06-04.
- [9] ROS Components. Robosense rs-helios 5515. <https://www.roscomponents.com/product/rs-helios-5515/>, 2024. Accessed: 2025-06-04.
- [10] RobotShop. Robosense rs-helios 5515 lidar (150 m). <https://eu.robotshop.com/de/products/robosense-rs-helios-5515-lidar-150-m>, 2024. Accessed: 2025-06-04.

-
- [11] Landscape Othmerding. Robosense lidar sensor rs-helios 5515. <https://landscape.othmerding.com/robosense-lidar-sensor-rs-helios-5515/19310>, 2023. Accessed: 2025-06-04.
 - [12] RoboSense. Rs-helios series lidar sensor. <https://www.robosense.ai/en/rslidar/Helios>, 2023. Accessed: 2025-06-04.
 - [13] Stereolabs. Zed 2i stereo camera. <https://www.stereolabs.com/en-es/store/products/zed-2i>, 2023. Accessed: 2025-06-04.
 - [14] Stereolabs. Zed 2i camera overview & datasheet. https://www.mouser.com/datasheet/2/1520/ZED_2i_Datasheet_v1_2-3498613.pdf, 2022. Accessed: 2025-06-04.
 - [15] Stereolabs. Stereolabs documentation hub. <https://www.stereolabs.com/docs>, 2023. Accessed: 2025-06-04.
 - [16] Seek Thermal. Compactpro ff - android (micro usb). <https://shop.thermal.com/compact-pro-ff-android-micro-usb>, 2023. Accessed: 2025-06-04.
 - [17] Seek Thermal. Compactpro - technical specifications. https://www.thermal.com/uploads/1/0/1/5/10156002/seek_compact_pro_product_specs.pdf, 2022. Accessed: 2025-06-04.
 - [18] Seek Thermal. Compact series user manual. https://www.thermal.com/uploads/1/0/1/3/101388544/compact_series_user_manual-web.pdf, 2022. Accessed: 2025-06-04.
 - [19] Seek Thermal. Compactpro thermal imaging camera. <https://www.thermal.com/compact-series.html>, 2023. Accessed: 2025-06-04.
 - [20] Element Product Development. Seek thermal – compact infrared imaging. <https://www.elementpd.com/portfolio-item/seek-thermal/>, 2022. Accessed: 2025-06-04.
 - [21] The Construct. Ros2 basics - introduction to ros2 and setup. <https://www.youtube.com/watch?v=ae0S9xqblrg>, 2022. Consultado en junio de 2025.
 - [22] Open Robotics. Ros 2 humble hawkbill documentation. <https://docs.ros.org/en/humble/index.html>, 2023. Consultado en junio de 2025.
 - [23] Canonical Ltd. Ubuntu 22.04 lts (jammy jellyfish). <https://ubuntu.com/>, 2022. Sistema base utilizado en el proyecto.
 - [24] Open Robotics. Rviz2 and ros 2 bag usage guide. <https://docs.ros.org/en/humble/Tutorials.html>, 2023. Visualización y grabación de datos en ROS2.
 - [25] Stereolabs. Zed ros2 wrapper - github repository. <https://github.com/stereolabs/zed-ros2-wrapper>, 2024. Consultado en junio de 2025.

-
- [26] RoboSense. rslidar_sdk: Robosense lidar driver for ros and ros2. https://github.com/RoboSense-LiDAR/rslidar_sdk, 2024. Acceso el 9 de junio de 2025.
 - [27] Stereolabs. zed-ros2-wrapper: Stereolabs zed ros2 integration. <https://github.com/stereolabs/zed-ros2-wrapper>, 2024. Acceso el 9 de junio de 2025.
 - [28] Stereolabs. zed-multi-camera: Sample launch for multiple zed cameras. <https://github.com/stereolabs/zed-multi-camera>, 2023. Acceso el 9 de junio de 2025.
 - [29] Titouan Malgouyres. seek_thermal_ros2: Ros2 package for seek thermal cameras. GitHub repository, 2023. https://github.com/Titou2222/seek_thermal_ROS2. Accessed June 9, 2025.
 - [30] conda forge. Miniforge: A conda-forge distribution. <https://github.com/conda-forge/miniforge>, 2025. Accedido el 10 de junio de 2025.
 - [31] RoboSense-LiDAR. rslidar_msg - custom ros2 message definitions for robosense lidars. https://github.com/RoboSense-LiDAR/rslidar_msg, 2024. Último acceso: junio de 2025.
 - [32] NVIDIA Corporation. Cuda toolkit - download for linux (ubuntu 22.04). https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&Distribution=Ubuntu&target_version=22.04&target_type=runfile_local, 2025. Último acceso: junio de 2025.
 - [33] Stereolabs. Stereolabs developers - release notes and downloads. <https://www.stereolabs.com/en-es/developers/release>, 2025. Último acceso: junio de 2025.
 - [34] Stereolabs Inc. *How to Install ZED SDK on Linux*. Stereolabs Inc., June 2025.
 - [35] syneart. Opencv 3.2 with contrib modules installation script. <https://gist.github.com/syneart/3e6bb68de8b6390d2eb18bff67767dcb/raw/OpenCV3.2withContrib.sh>, 2017. Script de instalación automatizada de OpenCV 3.2 en sistemas Linux. Consultado en junio de 2025.

