

Análisis, desarrollo y resultados de la
integración de sensores
ultrasónicos para la mejora de
navegación de un robot RFID

Álvaro Jiménez Barreno



Universitat
Pompeu Fabra
Barcelona

Análisis, desarrollo y resultados de la integración de sensores ultrasónicos para la mejora de navegación de un robot RFID

TREBALL FI DE GRAU DE
Álvaro Jiménez Barreno

Director: Rafael Pous, Lluís Hernández

Grau en Enginyeria en Sistemes Audiovisuals

Curs 2023-2024



Universitat
Pompeu Fabra
Barcelona

Escola
d'Enginyeria

Dedicado a mi yo del futuro para que recuerde
que fue aquí donde comenzó todo.

Agradecimientos

Quiero agradecer a todo el equipo de Keonn Technologies por el soporte que me han dado para que este trabajo sea posible. Especialmente, a Lluís Hernández, mi tutor, por guiarme cuando lo he necesitado, darme apoyo con sus conocimientos y ser un gran pilar durante estos meses. A Rafael Pous, director del trabajo, por darme la oportunidad de realizar este proyecto y aprender de él. Al equipo de producción, por la ayuda y el soporte que me han brindado siempre que lo he necesitado. A la Universidad Pompeu Fabra, por facilitar sus recursos y herramientas a los estudiantes.

Quiero agradecer también a mi familia escogida. A Estel, a Jordina y a Joan, por vuestro apoyo incondicional y vuestro acompañamiento durante estos meses. Y finalmente a mis padres y a mi hermana, por creer en mí y hacer que todo esto haya sido posible.

Muchas gracias.

Resumen

Este trabajo estudia la mejora de la navegación de un robot RFID en entornos de retail, con el objetivo de minimizar los puntos ciegos laterales y garantizar una navegación segura. Se busca una solución económica y óptima mediante la integración de cuatro sensores de proximidad ultrasónicos. Se analizan y testean varios modelos de sensores, eligiendo el mejor para la integración en la base robótica. Se realizan modificaciones en los archivos de descripción del robot, se optimiza la detección de obstáculos, el filtrado de datos y la comunicación con ROS. Además, se diseña una carcasa 3D especializada para los sensores. Finalmente, se llevan a cabo pruebas en un entorno simulado de tienda, evaluando el rendimiento del robot antes y después de la integración, destacando las ventajas obtenidas y extrayendo conclusiones que respaldan las hipótesis iniciales expuestas.

Resum

Aquest treball estudia la millora de la navegació d'un robot RFID en entorns de retail, amb l'objectiu de minimitzar els punts cecs laterals i garantir una navegació segura. Es busca una solució econòmica i òptima mitjançant la integració de quatre sensors de proximitat ultrasònics. S'analitzen i testen diversos models de sensors, triant el millor per a la integració en la base robòtica. Es realitzen modificacions en els arxius de descripció del robot, s'optimitza la detecció d'obstacles, el filtratge de dades i la comunicació amb ROS. A més, es dissenya una carcasa 3D especialitzada per als sensors. Finalment, es duen a terme proves en un entorn simulat de botiga,avaluant el rendiment del robot abans i després de la integració, destacant els avantatges obtinguts i extraient conclusions que recolzen les hipòtesis inicials exposades.

Abstract

This work studies the improvement of navigation for an RFID robot in retail environments, aiming to minimize lateral blind spots and ensure safe navigation. An economical and optimal solution is sought by integrating four ultrasonic proximity sensors. Various sensor models are analyzed and tested, choosing the best one for integration into the robotic base. Modifications are made to the robot's description files, obstacle detection, data filtering, and communication with ROS are optimized. Additionally, a specialized 3D housing for the sensors is designed. Finally, tests are conducted in a simulated store environment, evaluating the robot's performance before and after integration, highlighting the advantages obtained and drawing conclusions that support the initial hypotheses presented.

Índice

Introducción.....	1
1.1 Motivación.....	1
1.2 Hipótesis.....	1
1.3 Contexto del trabajo.....	3
Análisis comparativo de sensores de proximidad.....	6
2.1 Introducción.....	6
2.2 Selección del tipo de sensor de proximidad.....	6
2.3 Precio.....	7
2.4 Configuración y cantidad de sensores.....	7
2.4.1 Distribución estratégica.....	7
2.4.2 Distribución estratégica.....	7
2.5 Modelo de los sensores ultrasónicos.....	9
2.5.1 Descripción de modelos.....	9
2.5.2 Conectividad de los modelos.....	10
2.6 Testeo de los modelos elegidos.....	11
2.7 Conclusiones.....	13
Desarrollo de la integración de sensores.....	14
3.1 Introducción.....	14
3.2 Implementación del software.....	14
3.2.1 Desarrollo del paquete de ROS ultrasonic_sensor.....	14
3.2.2 Automatización de la detección de datos vía serial.....	15
3.2.3 Modificaciones del archivo de descripción URDF.....	15
3.2.4 Integración de los datos del sensor al paquete de navegación move_base.....	15
3.2.5 Implementación del filtro de datos Kalman.....	17
3.3 Diseño del modelo 3D para la integración del sensor en el robot.....	19
3.3.1 Prototipos iniciales de diseños 3D.....	19
3.3.2 Iteraciones de mejora del modelo.....	20
3.3.3 Modelo 3D final.....	21
Pruebas de navegación y testing.....	25
4.1 Introducción.....	25
4.2 Escenario 1.....	26
4.3 Escenario 2.....	28
4.4 Escenario 3.....	29
4.5 Resultados y conclusiones.....	30
Conclusiones y trabajo futuro.....	31
Apéndice.....	35

Listado de figuras

Fig. 1: Esquema de un sistema RFID básico.....	3
Fig. 2: AdvanReader 160.....	3
Fig. 3: Antena SP11.....	3
Fig. 4: Robin-200.....	4
Fig. 5: Esquema del Robin-200. Medidas de la torre.....	4
Fig. 6: Esquema del Robin-200. Medidas de la base.....	4
Fig. 7: Funcionamiento de ROS Master sobre los diferentes nodos.....	4
Fig. 8: Esquema de obtención de datos mediante topics y servicios.....	4
Fig. 9: Navegación del Robin-200 sobre un mapa en RViz.....	5
Fig. 10: Esquema de un sensor IR.....	6
Fig. 11: Esquema de un sensor ultrasónico.....	6
Fig. 12: Cobertura inicial del Robin-200.....	7
Fig. 13: Cobertura final implementando el Caso 2 de sensorización del Robin-200.....	8
Fig. 14: Modelo MaxBotix LV-MaxSonar EZ0.....	9
Fig. 15: Modelo MaxBotix HRLV-ShortRange EZ1.....	9
Fig. 16: Modelo DFROBOT SEN URM37.....	9
Fig. 17: Esquema de comunicación PC/Microcontrolador con UART y USB-RS232.....	10
Fig. 18: Esquema de pines del cable USB-RS232.....	11
Fig. 19: Funcionamiento del nodo ultrasonic_sensor.....	15
Fig. 20: Listado de puertos ttyUSB del Robin-200.....	15
Fig. 21 Mensaje de tipo Range.....	16
Fig. 22: Visualización de mensajes Range del sensor ultrasónico en RViz.....	16
Fig. 23: Sensores ultrasónicos al lado de un palé.....	17
Fig. 24: Sensores ultrasónicos al lado de una estantería con huecos.....	17
Fig. 25: Visualización en RViz. Detección 1 de los sensores.....	17
Fig. 26: Visualización en RViz. Detección 2 de los sensores.....	17
Fig. 27: Gráfico de resultados del KF sin modificar.....	19
Fig. 28: Gráfico de resultados del KF modificado.....	19
Fig. 29: Visualización en RViz del Robin-200 con detecciones constantes.....	19
Fig. 30: Modelo 1 en 3D.....	20
Fig. 31: Carcasa impresa.....	20
Fig. 32: Sensor en el robot con el modelo 1.....	20
Fig. 33: Modelo 2 en 3D.....	20
Fig. 34: Carcasa impresa.....	20
Fig. 35: Sensor en el robot con el modelo 2.....	20
Fig. 36: Modelo 3D de la segunda iteración.....	21
Fig. 37: Modelo impreso con tapa puesta.....	21
Fig. 38: Modelo impreso defectuoso.....	21
Fig. 39: Modelo 3D de la tapa.....	21
Fig. 40: Modelo 3D final de la carcasa.....	21
Fig. 41: Esquema de conexión entre sensores y PC del Robin-200.....	22
Fig. 42: Carcasa con sensor atornillada. Vista frontal.....	22

Fig. 43: Carcasa con sensor atornillada. Vista trasera.....	22
Fig. 44: Pieza superior con los cuatro sensores integrados.....	22
Fig. 45: Sensores conectados al USB HUB.....	22
Fig. 46: Sensores integrados en el lateral izquierdo.....	22
Fig. 47: Sensores integrados en el lateral derecho.....	22
Fig. 48: Visualización de RViz del Robin-200 generando obstáculos al rotar.....	23
Fig. 49: Visualización de RViz del Robin-200 generando obstáculos al girar.....	23
Fig. 50: Visualización de RViz del Robin-200 generando obstáculos en un área libre del pasillo.....	23
Fig. 51: Escenario inicial del sensor en el robot.....	23
Fig. 52: Escenario modificado en la tapa del sensor.....	23
Fig. 53: Visualización en RViz del Robin-200 rotando sin detectar ruido.....	24
Fig. 54: Visualización en RViz del Robin-200 en un pasillo sin generar ruido en espacios libres.....	24
Fig. 55: Modelo 3D final de la tapa, con reducción de ángulo de apertura.....	24
Fig. 56: Sensores del lateral izquierdo del robot con la nueva tapa.....	24
Fig. 57: Sensores del lateral derecho del robot con la nueva tapa.....	24
Fig. 58: Escenario inicial de testeо. Pasillo con dos cajas que bloquean el paso del Robin-200.....	25
Fig. 59: Captura de RViz mostrando el escenario inicial.....	25
Fig. 60: Visualización del escenario 2 en RViz.....	25
Fig. 61: Visualización del escenario 3 en RViz.....	25
Fig. 62: Visualización en RViz de replaneo lateral.....	26
Fig. 63: Visualización en RViz de replaneo por las cajas una vez rotas.....	26
Fig. 64: Visualización de RViz del Robin-200 actualizando su planner rodeando el pasillo.....	26
Fig. 65: Visualización de RViz del Robin-200 moviéndose por el espacio actualizando el planner.....	26

Listado de tablas

Tabla 1: Comparativa de casos con los modelos MaxBotix.....	12
Tabla 2: Resultados del escenario 1.....	27
Tabla 3: Resultados del escenario 2.....	28
Tabla 4: Conclusiones numéricas del escenario 2.....	28
Tabla 5: Resultados del escenario 3.....	29
Tabla 6: Conclusiones numéricas del escenario 3.....	29

Capítulo 1

Introducción

El capítulo de *Introducción* describe el escenario en el que se trabajará este proyecto, las hipótesis y objetivos propuestos, destacando soluciones para cumplirlos y el contexto del trabajo. Se ha creado un repositorio en [GitHub](#) para acceder al contenido que da soporte a este trabajo, entre los cuales se encuentran archivos Python, planos, modelos 3D y contenido multimedia.

1.1 Motivación

La motivación del proyecto reside en mejorar el producto Robin-200, un robot de inventario autónomo basado en tecnología RFID. El objetivo de esta mejora es minimizar los puntos débiles con los que cuenta actualmente la navegación del robot y conseguir un producto más completo, diferenciándolo de los demás robots autónomos en el campo del retail.

Se analiza la navegación del robot teniendo en cuenta los sensores actuales y la cobertura de detección de obstáculos, para maximizarla y eliminar posibles ángulos muertos. Se tienen en cuenta las necesidades del producto para satisfacerlas, encontrando un balance entre eficiencia, precisión y cohesión con el producto. También se busca una solución económica. Se pretende obtener buenos resultados sin que la solución suponga un aumento excesivo del coste del Robin-200. Con esto se pretende conseguir un robot con una navegación más eficiente, que proporcione mejores resultados en tienda y sea un producto más atractivo respecto a sus competidores.

Mejorar la navegación del Robin-200 mediante sensorización proporcionaría mayor conocimiento del entorno para el robot. Por tanto, este sería capaz de analizar el espacio de forma más precisa, sería más resiliente a cambios de entorno y se conseguiría un mayor rendimiento en los inventarios, con unas lecturas RFID más estables.

El objetivo principal es que, después de este trabajo, se apliquen los cambios a nivel comercial e integrar la solución propuesta al producto en caso de que las hipótesis se cumplan y se obtenga una evidente mejora para la funcionalidad del producto.

1.2 Hipótesis

El trabajo se construye a partir de las siguientes hipótesis:

1. La base de Robin-200 es un cuboide rectangular y hace que el robot tenga puntos ciegos en las áreas laterales. En tiendas de *retail*, es posible que ocurra el siguiente escenario:
 - El robot al navegar por pasillos estrechos o bloqueados por objetos, enfrenta dificultades para encontrar una ruta efectiva hasta la salida. Debido a la falta de sensores laterales, el robot interpreta erróneamente los pasillos laterales como espacios transitables, resultando en intentos fallidos de navegación y posibles colisiones.

Se hipotetiza que la integración de sensores en los laterales del robot mejorará significativamente su capacidad de detección de obstáculos y evitará colisiones durante la navegación.

Propuesta: Se llevará a cabo un análisis y pruebas comparativas entre el rendimiento de navegación del Robin-200 antes y después de la integración de los sensores. Se registrarán y analizarán datos sobre la detección y evasión de obstáculos en entornos reales y preparados para estas pruebas. Entre otros, se estudiará:

- a. El comportamiento del robot en pasillos estrechos y pasillos sin salida.

- b. El tiempo de recorrido de la tienda leyendo los artículos RFID.
 - c. El número de replanificaciones de ruta que el robot realiza para llegar a una determinada posición.
 - d. La estabilidad del sistema de navegación del robot con los nuevos sensores basándose en la precisión de la detección de obstáculos.
 - e. La fiabilidad de los sensores en diferentes condiciones ambientales y situaciones recurrentes en entornos de retail.
2. El tipo, número, distribución estratégica y selección del modelo de sensor será determinante para detectar obstáculos desde diferentes ángulos y maximizar la cobertura de detección del entorno.
- Propuesta:** Se propone realizar un estudio completo de sensores basándose en las necesidades del Robin-200. Se seguirán los siguientes puntos:
- a. **Selección del tipo de sensor:** Se contemplarán sensores de infrarrojos, inductivos, láser, ópticos y de ultrasonido, entre otros.
 - b. **Determinación del número de sensores:** Se determinará la cantidad óptima de sensores necesarios para proporcionar una cobertura lateral adecuada para el Robin-200. Se tendrá en cuenta las dimensiones de este y el rango y apertura de visión de los sensores.
 - c. **Distribución estratégica de los sensores:** Se determinará la posición más eficiente, junto al número de sensores para maximizar la cobertura lateral del Robin-200.
 - d. **Selección del modelo de sensor:** Una vez valorado el tipo, el número de sensores y su distribución en el robot, se realizarán pruebas comparativas con varios modelos de sensores para determinar cuál ofrece la mejor combinación de precisión, alcance, fiabilidad e integración en el robot. Se considerarán aspectos como:
 - La eficacia del uso de datos en el framework.
 - La facilidad de integración en el robot.
 - La velocidad de respuesta.
 - El precio.
 - e. **Testeo en un entorno real:** Finalmente, se integrarán los sensores en el robot y se realizarán pruebas exhaustivas siguiendo los aspectos mencionados en la hipótesis anterior.
3. La implementación de estos dispositivos en el producto será completa y coherente con el robot, tanto a nivel de hardware como de software.

Propuesta: Para asegurar una integración completa de los sensores al producto, se asegurará una:

- a. **integración con el framework ROS y el software de navegación** para incorporar los sensores en el proceso de planificación de rutas y evasión de obstáculos. Se ajustarán los algoritmos de navegación para tener en cuenta los datos de los sensores y garantizar un desplazamiento seguro y eficiente del robot.
- b. **integración física en el hardware del robot**, en la que se diseñará una estructura de soporte para situar los sensores de manera segura y estable. Se realizarán las modificaciones necesarias en el chasis del robot para garantizar que los sensores queden correctamente posicionados y protegidos contra posibles daños.
- c. **conexión eléctrica y electrónica adecuada** entre los sensores y el Robin-200. Se diseñará e implementará una conexión lo menos intrusiva posible a la actual que permita integrar los sensores de forma segura.

Se pretende llevar a cabo estas propuestas y exponer los resultados obtenidos en las conclusiones finales para determinar si las hipótesis se han cumplido.

1.3 Contexto del trabajo

En este apartado se contextualiza el trabajo, explicando los conceptos teóricos en los que está basado. De esta forma, se pretende dar conocimiento al lector de los campos trabajados para facilitar la comprensión de las decisiones tomadas durante el proyecto.

1. Radio-Frequency Identification (RFID)

La tecnología Radio-Frequency Identification (RFID) emplea acoplamiento electromagnético en la banda de radiofrecuencia para la identificación única de objetos [1]. Un sistema RFID básico (Fig. 1) consta de tres elementos principales [2] [3]:

- **Tag RFID:** Contiene un identificador único, almacenado en su chip integrado, que puede ser leído a distancia.
- La **antena** absorbe ondas de radiofrecuencia del reader y permite la comunicación bidireccional, enviando y recibiendo datos.
- El **reader** emite señales de radio que activan el tag RFID. Al recibir estas señales, el tag responde con una señal de retorno que contiene su identificador único, entre otros. Esta señal es capturada por la antena del lector y convertida en datos digitales.

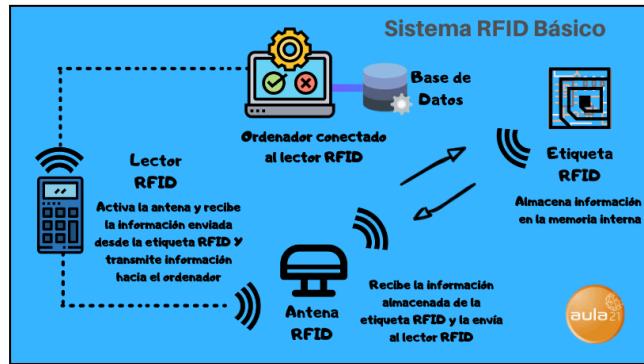


Fig. 1: Esquema de un sistema RFID básico

El sistema que utiliza el Robin-200 es el de dos readers AdvanReader 160 (Fig. 2) [4] y cuatro antenas SP11 (Fig. 3) [5], ambos productos de Keonn Technologies. Estos elementos, junto con la circulación autónoma del robot, permiten realizar una lectura de los ítems con tags RFID de una tienda. Este proceso se conoce como una misión de inventario del Robin-200. En esta, el robot navega autónomamente mientras tiene sus readers activos y va leyendo los tags de su alrededor.

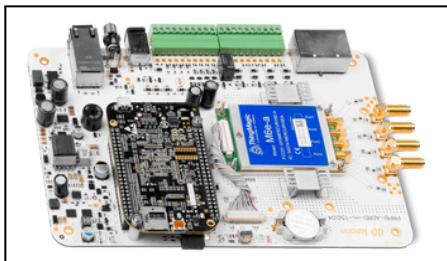


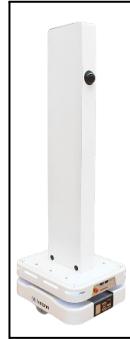
Fig. 2: AdvanReader 160



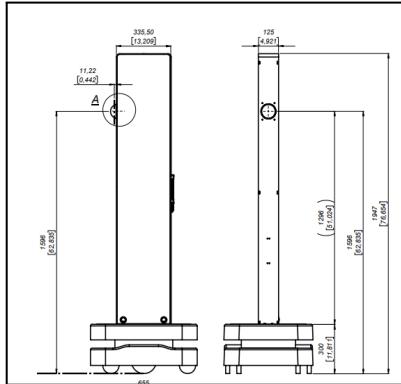
Fig. 3: Antena SP11

2. Robin-200: Producto de Keonn Technologies

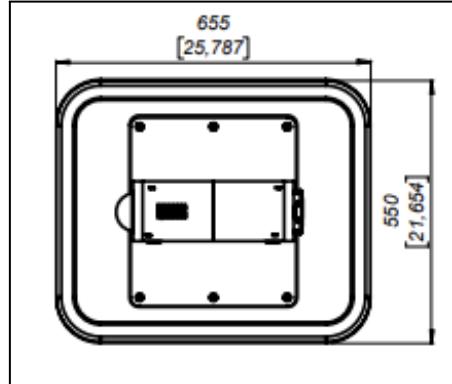
Robin-200 (Fig. 4) es un robot autónomo creado por Keonn Technologies, así como el robot con el que se trabaja para este proyecto [6]. Este producto se trata de un sistema móvil y autónomo de RFID que realiza inventarios automáticos en diferentes espacios, sobre todo en tiendas de retail, proporcionando una mayor precisión en el inventario RFID en comparación con los lectores de mano tradicionales.



Cuenta con un láser 3D con ángulo de detección de 180° en la parte frontal de la torre (Fig. 5) y dos cámaras RGB-D con ángulo de detección de 78°, una en la parte frontal de la base y la otra en la parte trasera (Fig. 6) [7].



*Fig. 5: Esquema del Robin-200.
Medidas de la torre*



*Fig. 6: Esquema del Robin-200.
Medidas de la base*

Fig. 4: Robin-200

El principal desafío del Robin-200 es navegar por el entorno de manera óptima y realizar la gestión de inventario RFID. Para ello, uno de los métodos utilizados es la dinamización de obstáculos en el entorno. El mapa por donde navega se procesa para convertirlo en escala de grises, lo que permite al robot navegar a través de obstáculos en el mapa en caso de que los sensores no los detecten. Se explica con más detalle en el punto 5: *Navegación del robot con move_base* de este mismo apartado.

3. Robot Operating System (ROS)

Robot Operating System (ROS) es un conjunto de frameworks para el desarrollo de software de robots, que proporciona herramientas, bibliotecas y convenciones para simplificar la creación de aplicaciones robóticas complejas [8] [9] [10] [11]. Facilita la comunicación entre componentes de un sistema robótico a través de una arquitectura basada en la publicación y suscripción de **mensajes**, permitiendo la integración de sensores, actuadores y algoritmos de procesamiento de datos.

Una característica clave de ROS es su estructura modular, que permite crear nodos individuales para tareas específicas. Estos nodos, gestionados por el ROS Master (Fig. 7), se comunican mediante mensajes de datos estructurados a través de canales llamados topics. Además, ROS soporta servicios y acciones, que permiten la comunicación sincrónica y asíncrona entre nodos (Fig. 8).

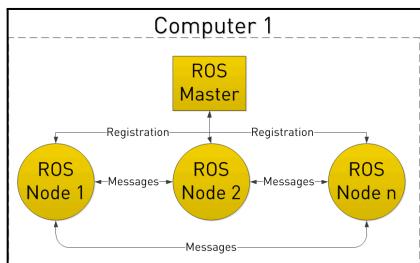


Fig. 7: Funcionamiento de ROS Master sobre los diferentes nodos

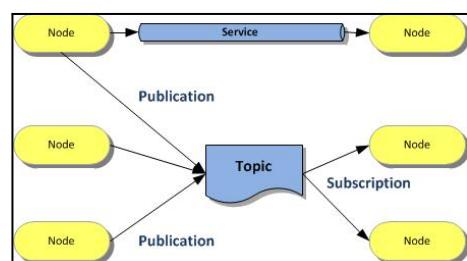


Fig. 8: Esquema de obtención de datos mediante topics y servicios

Una herramienta esencial dentro de ROS es **RViz**. Tiene la característica de visualización 3D que permite representar la información de los topics de manera visual.

RViz facilita la depuración, el análisis y el desarrollo de algoritmos y sistemas robóticos complejos al proporcionar una representación visual clara y detallada de los datos en tiempo real. Además, puede mostrar datos de sensores como cámaras y LiDAR, visualizar modelos 3D, y mostrar mapas y trayectorias, haciendo que la interpretación de los sistemas robóticos sea más intuitiva [12] [13].

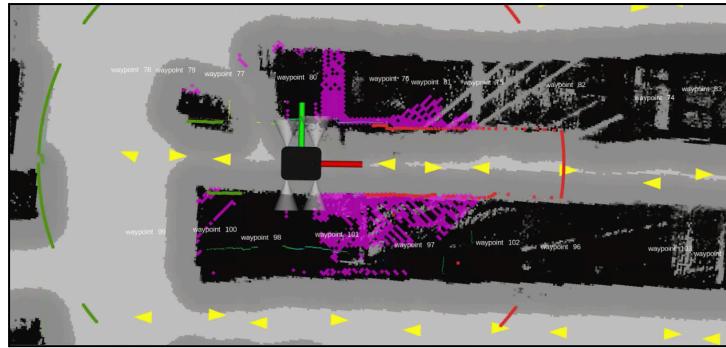


Fig. 9: Navegación del Robin-200 sobre un mapa en RViz

4. Unified Robot Description Format (URDF)

El Unified Robot Description Format (URDF) es un archivo en formato XML utilizado en el sistema ROS para definir la estructura física y cinemática de un robot. Este archivo describe detalladamente los componentes del robot, como los links, que unen las partes rígidas del robot, y joints, que indican cómo se conectan y se mueven esos enlaces entre sí [14]. Incluyen propiedades físicas como la masa, el centro de masa y la geometría de colisión, y otras propiedades visuales para la representación gráfica del robot en simuladores como RViz.

El archivo XML Macros es una extensión del formato URDF para facilitar y optimizar la creación de descripciones de robots complejas. Se pueden definir bloques reutilizables de código XML y parametrizar componentes, lo que permite una mayor flexibilidad y eficiencia en la configuración del robot. Los archivos Xacro son procesados para generar el URDF final, lo que simplifica la gestión y el mantenimiento de descripciones de robots grandes y complejos. Esto resulta en un flujo de trabajo más modular y manejable.

5. Navegación del robot con *move_base*

El paquete *move_base* es una herramienta esencial en la navegación robótica, diseñada para guiar una base móvil hacia un objetivo especificado en el entorno. Este paquete combina un planificador global y uno local para realizar tareas de navegación complejas. Además, *move_base* gestiona dos costmaps, uno para cada planificador ([véase Apéndice-1](#)).

En la Fig. 9 se observa al Robin-200 navegando por un entorno definido por un mapa. Respecto a la navegación, se observan los siguientes elementos:

- La inflación de obstáculos (gris oscuro) es generada por dos tipos de costmaps:
 - a. El global costmap del mapa estático (figuras en negro).
 - b. El local costmap (rosa) para evitar colisiones.
- El local costmap se construye a partir de los datos captados por varios sensores:
 - a. La cámara frontal (puntos rojos).
 - b. La cámara trasera (puntos verdes).
 - c. Los sensores ultrasónicos laterales (conos blancos).
- Los goals (flechas amarillas) son los objetivos a los que debe llegar el robot y permiten moverse por el espacio de forma autónoma.

Capítulo 2

Análisis comparativo de sensores de proximidad

Este capítulo abarca todo el estudio realizado para implementar los sensores al Robin-200. Se explica la motivación del por qué se usan sensores de proximidad así como una introducción al análisis a realizar, basado en las propuestas para cumplir las hipótesis. Se realizan conclusiones de cada uno de los apartados, especificando los resultados más relevantes. Finalmente, se realizan unas conclusiones generales del capítulo, que contienen la elección final de sensores.

2.1 Introducción

Los sensores de proximidad detectan la presencia o ausencia de objetos sin contacto físico, transmitiendo esta información como una señal eléctrica. Esto asegura su durabilidad y minimiza el desgaste [15]. Funcionan convirtiendo datos en señales eléctricas, permitiendo la fácil interpretación por sistemas electrónicos. Su capacidad para detectar objetos y movimientos de manera confiable y sin contacto los hace esenciales en sistemas automatizados.

2.2 Selección del tipo de sensor de proximidad

En este apartado se comentan los principales sensores de proximidad más usados en el campo de la robótica [16] [17]:

- **Infrarrojos - IR:** Emite luz infrarroja y mide la reflexión o absorción de la luz para detectar un objeto (Fig. 10) [18].
- **Ultrasónico:** Emite ondas de sonido de alta frecuencia y mide el tiempo que tardan las ondas en rebotar en un objeto y regresar al sensor (Fig. 11) [19].
- **Óptico:** Combinación de luz y receptor para detectar objetos (p. ej.: cámaras)
- **Láser:** Utiliza un haz láser enfocado para medir la reflexión o absorción de la luz láser para detectar objetos.

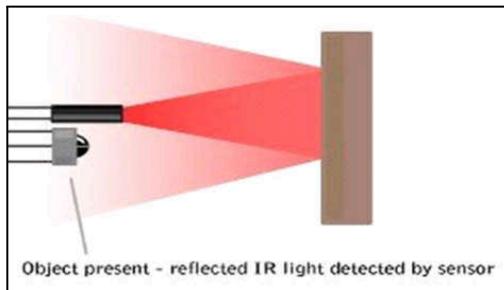


Fig. 10: Esquema de un sensor IR

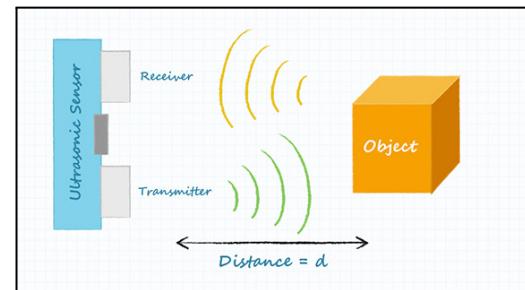


Fig. 11: Esquema de un sensor ultrasónico

En el apéndice ([véase Apéndice-2](#)) se puede encontrar un análisis comparativo de sensores. Durante el análisis, se observa como los sensores ultrasónicos son una solución sólida. Aunque no son los sensores con mayor precisión, ofrecen una buena exactitud para la detección de obstáculos y una rápida respuesta crucial para la navegación en tiempo real y la evasión de objetos. No se ven afectados por condiciones ambientales y son eficaces en la detección de objetos difíciles como aquellos de formas irregulares.

2.3 Precio

El costo es un elemento crítico en la selección de componentes para el testeo y la posterior implementación. Por ello, es necesario considerar el costo para garantizar que la solución sea efectiva y asequible. Con una solución económica, se consigue:

1. Reducción de costos de producción con las que se obtiene un margen de ganancias más amplio a nivel comercial.
2. Mayor margen para experimentar y prototipar con ellos.
3. Mayor flexibilidad de diseño. Esto permite poder analizar e implementar configuraciones con múltiples unidades, mejorando la robustez del sistema.

Es por eso que este estudio se ha realizado con la intención de encontrar una opción que ofrezca buenos resultados sin aumentar significativamente el precio del producto.

Se observa ([véase Apéndice-3](#)) que los sensores de ultrasonido son significativamente más económicos en comparación con los sensores láser industriales utilizados por la competencia. Este bajo precio aporta las cuatro características destacadas previamente. Otros sensores de precios elevados como las cámaras pueden ser una buena solución para maximizar la cobertura lateral del Robin-200, pero no se estarían cumpliendo las hipótesis, ya que se busca una solución económica. La solución propuesta debe ser fiable, con mejoras notables y con un buen balance entre eficiencia y precio

2.4 Configuración y cantidad de sensores

2.4.1 Distribución estratégica

Tanto la ubicación como el número de sensores a incorporar en el robot, son dos factores importantes a tener en cuenta en la decisión final. Para ello, se considera:

- El diseño del robot y sus dimensiones.
- La facilidad de integración de los sensores en el robot.
- Las capacidades del sensor teniendo en cuenta la necesidad de cobertura lateral del robot.

El robot se divide en una torre y una base:

- En el caso de la **torre**, esta tiene dos tapas laterales que se pueden abrir. En la base están todas las conexiones con el láser, los readers y antenas, el dispositivo móvil, entre otras. Incluir los sensores en la torre es inviable por una limitación de espacio, dificultad de acceso a los sensores, la poca estabilidad de la torre y la poca optimización del espacio.
- En el caso de la **base**, los sensores estarán en los extremos del robot, irían integrados en una parte más estable y espaciosa. Por lo que se decide situarlos en los laterales de esta.

2.4.2 Distribución estratégica

Para el estudio de esta sección, se ha usado el software AutoCAD para realizar varios [sketches](#) teniendo en cuenta las dimensiones del robot y las distancias necesarias para cubrir los laterales, teniendo en cuenta los posibles ángulos de apertura de los sensores.

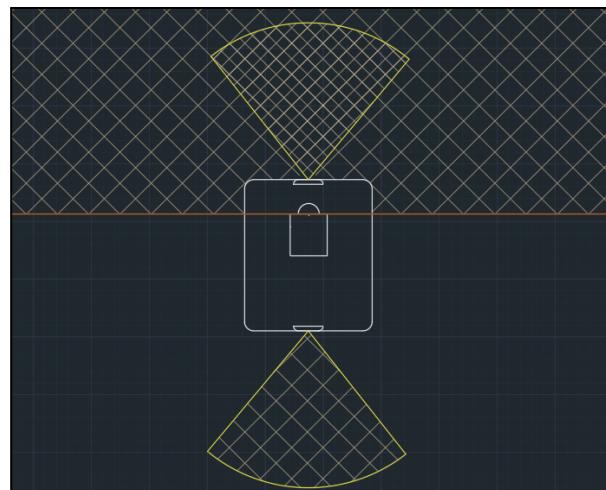


Fig. 12: Cobertura inicial del Robin-200

Se parte de la cobertura de sensores inicial del Robin-200 dada por el láser 3D y las cámaras RGB-D frontal y trasera (Fig. 12).

Se realizan tres escenarios para analizar diferentes configuraciones de cobertura lateral ([véase Apéndice-4](#)).

1. Un sensor lateral en el centro, en cada uno de los lados:
 - Los sensores pueden ver el suelo antes de llegar a detectar un obstáculo debido al amplio rango de detección, por lo que las lecturas no serán fiables. Se determina que no es una buena opción.
2. Dos sensores laterales situados a 115 mm de cada esquina de la base:
 - Con ángulos y distancias razonables, se puede cubrir toda la parte lateral del robot eficientemente. Aunque duplicar el número de sensores aumenta el costo, también proporciona datos más precisos y el doble de información.
3. Un sensor situado en el centro del lateral (a 212.5 mm de cada sensor en la esquina) y dos sensores situados a 115 mm de cada esquina de la base.
 - A 300 mm, el robot logra una cobertura total, pero los ángulos y distancias no mejoran significativamente las capacidades de los sensores y encarecen la solución. La superposición excesiva en el área de detección entre sensores hace que esta opción no sea óptima.

Después de estudiar estos tres casos, la distribución que se utiliza para implementar los sensores al robot será la segunda opción: **Dos sensores laterales situados a 115 mm de cada esquina de la base**. Esta configuración proporciona una cobertura lateral completa, eliminando puntos ciegos y garantizando una detección de obstáculos con ángulos de apertura y distancias razonables. Aunque no es la solución más económica, se obtiene un buen balance entre información y resolución de los datos. Se ha realizado un esquema de cobertura total (Fig. 13) con una distancia límite de 750 mm y un ángulo de apertura de 31,64° para los sensores, asegurando una detección eficaz.

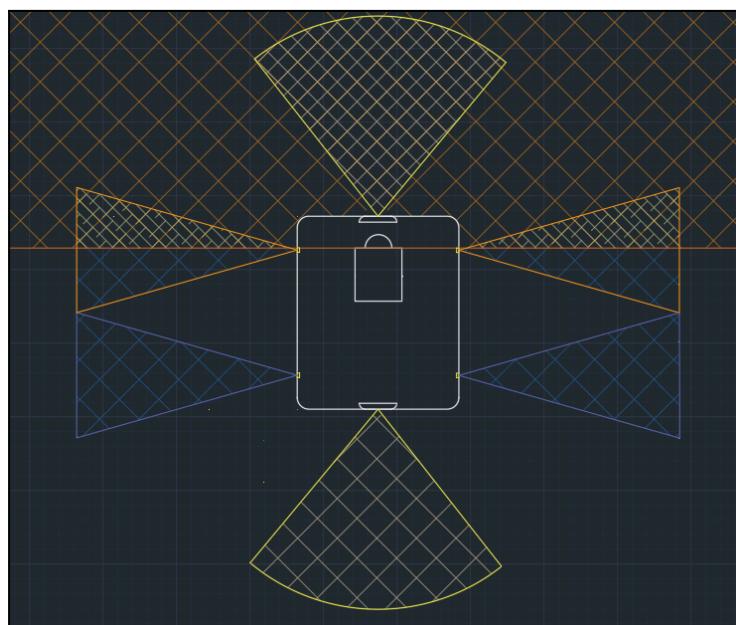


Fig. 13: Cobertura final implementando el Caso 2 de sensorización del Robin-200

2.5 Modelo de los sensores ultrasónicos

2.5.1 Descripción de modelos

Una vez elegido el tipo de sensor, distribución en el robot y cantidad de sensores, se realiza un estudio entre los sensores más comunes que hay disponibles en el mercado. Basándose en la [hipótesis 2](#), los sensores deben ser fiables y obtener el mejor balance entre precisión de datos, facilidad de integración y precio. Se realiza un análisis y pruebas con varios sensores del mercado como el HC-SR04 [28], Parallax PING [28] [29], XXS18P1PM12 [30]. No se han incluido en el trabajo debido a la falta de compatibilidad con una conexión directa con el PC, necesitando implementar un Arduino en la solución ([véase 2.5.2](#)). Los sensores ultrasónicos propuestos son siguientes:

- MaxBotix LV-MaxSonar

Las principales especificaciones de la serie MaxBotix LV-MaxSonar (Fig. 14) son las siguientes:



Fig. 14: Modelo
MaxBotix
LV-MaxSonar EZ0

Voltaje de entrada: 2.5V - 5V	Rango de medición: 15 cm - 640 cm
Consumo de corriente: 2 mA	Dimensiones: 16.4 mm x 16.4 mm x 15.5 mm
Ángulo de detección: 20° - 30°	Precio: 28.95€

Estas especificaciones muestran un sensor eficiente, con una alimentación de bajo voltaje y amperaje, lo que permite su uso en sistemas con restricciones energéticas. Pueden conectarse directamente a un microcontrolador, lo que facilita su integración en proyectos de electrónica y automatización [20].

Estos sensores proporcionan una salida analógica que emite un voltaje proporcional a la distancia medida, una salida serie para datos en formato serial, y una señal PWM (modulación por ancho de pulso) cuya duración es proporcional a la distancia medida ([véase Apéndice-5.1](#)) [21].

- MaxBotix HRLV-ShortRange

Las principales características de la serie MaxBotix HRLV-ShortRange (Fig. 15) son las siguientes:

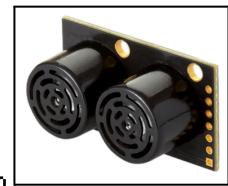


Fig. 15: Modelo
MaxBotix
HRLV-ShortRange
EZ1

Voltaje de entrada: 2.5V - 5V	Rango de medición: 2 cm - 500 cm
Consumo de corriente: 2.5 mA - 3.5 mA	Dimensiones: 38.0 mm x 22.2 mm x 13.9 mm
Ángulo de detección: 30° - 40°	Precio: 32.95€

Se observa un sensor adecuado para aplicaciones que requieren alta precisión en un rango corto a medio, con bajo consumo de energía. Su alimentación de bajo voltaje y amperaje permite su uso en sistemas con restricciones energéticas ([véase Apéndice-5.2](#)).

- DFROBOT SEN URM37

Las principales características del modelo DFROBOT SEN URM37 de la marca DFROBOT (Fig. 16) son las siguientes:



Fig. 16: Modelo
DFROBOT SEN URM37

Voltaje de entrada: 3.3V - 5.5V	Rango de medición: 2 cm - 800 cm
Consumo de corriente: 20 mA	Dimensiones: 51 mm x 22 mm x 14 mm
Ángulo de detección: Hasta 60°	Precio: 16€

Este sensor es una opción robusta para aplicaciones que requieren una amplia gama de medición de distancias, desde muy cortas hasta largas ([véase Apéndice-5.3](#)). No obstante, se necesita un sensor más centrado en distancias cortas. El fabricante nos asegura precisión en todo su rango, pero su ángulo de apertura tan amplio no es ideal para nuestro escenario, porque puede llegar a detectar el suelo muy rápidamente [22] [23] [24].

2.5.2 Conectividad de los modelos

Comunicación UART

La Universal Asynchronous Receptor-Transmisor (UART) se utiliza en sensores ultrasónicos porque permite una integración sencilla y económica con microcontroladores y PCs, facilitando la transmisión de datos con solo dos cables (Fig. 17). Aunque existen protocolos más rápidos, la UART ofrece una solución eficiente y fiable para recibir datos de sensores, manteniendo un equilibrio entre simplicidad y funcionalidad sin perder frecuencia de lectura. Además permite la conexión directa de un sensor a un PC mediante un adaptador USB-SERIAL, facilitando la comunicación y transferencia de datos [25]. Para más información sobre la comunicación UART, [véase Apéndice-6](#).

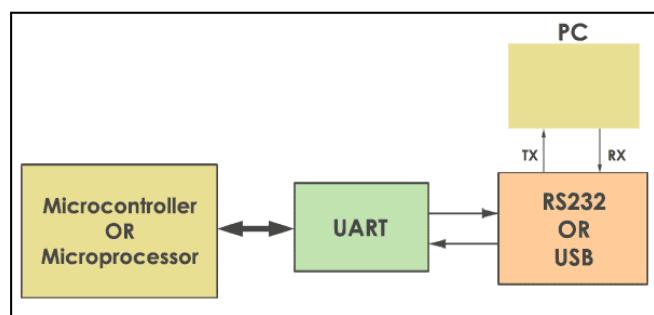


Fig. 17: Esquema de comunicación PC/Microcontrolador con UART y USB-RS232

Tipos de comunicación digital

A continuación se especifican los tipos de comunicación digital más comunes usados en transmisión de datos en sensores ultrasónicos:

Transistor-Transistor Logic (TTL): Comunicación serial de corto alcance con niveles de voltaje bajos, fácil de integrar con microcontroladores pero susceptible a interferencias electromagnéticas.

Recommended Standard 232 (RS-232): Comunicación serial de alcance medio con niveles de voltaje altos, mejor inmunidad a interferencias pero tiene menor velocidad que otros tipos de comunicaciones (RS-485).

RS-485: Comunicación serial de largo alcance y multidrop con alta inmunidad a interferencias, pero más compleja de implementar y requiere hardware adicional para conversión de señales [26].

Se decide que los sensores deben tener la posibilidad de transmitir datos directamente al PC mediante comunicación RS232 debido a su facilidad de integración, robustez ante cambios de entorno, fiabilidad y eficiencia para respuestas a distancias medias.

Compatibilidad de comunicaciones en los sensores

Una vez elegido el tipo de comunicación, se observan las compatibilidades de los sensores estudiados y cuáles se adaptan mejor al caso estudiado:

1. DFROBOT SEN URM37

- Utiliza pines de señal para *trigger* y *echo*, y opciones de comunicación serial (TX/RX) para transmitir datos.
- Ofrece comunicación TTL y RS232, proporcionando flexibilidad en la integración.
- Puede conectarse directamente a un PC usando un adaptador USB-TTL o USB-RS232. La configuración del adaptador y el software adecuado permite recibir datos seriales directamente.
- Proporciona una salida serial donde los datos de distancia se pueden recibir directamente, simplificando la integración.

2. MaxBotix HRLV-ShortRange y MaxBotix LV-MaxSonar

- Proporciona múltiples formas de comunicación: salida analógica, PWM, y serial.
- Utiliza comunicación TTL o RS232 para la salida serial.
- Los datos pueden ser leídos directamente en el PC en formato serial mediante un adaptador USB-TTL o USB-RS232.

Para realizar pruebas y garantizar una fácil integración con el Robin-200, se escogen los sensores MaxBotix HRLV-ShortRange, MaxBotix LV-MaxSonar y DFROBOT SEN URM37. Estos cuentan con las características necesarias para cumplir con la [hipótesis 3](#). En el caso de los modelos MaxBotix, que no son de gama única, se utilizan los de la gama EZ0 y EZ1 ya que tienen un ángulo de apertura más amplio y es más adecuado para nuestro caso. Para el conversor de comunicación, se utilizará el cable USB-RS232-WE-1800-BT_5.0

2.6 Testeo de los modelos elegidos

Para el testeo en los sensores se ha seguido las indicaciones de la Fig. 18. En todas las conexiones el pin RX del sensor debe ir al pin TX del cable y viceversa. El VCC y GND del sensor también deben ir conectados con los pines respectivos del cable.

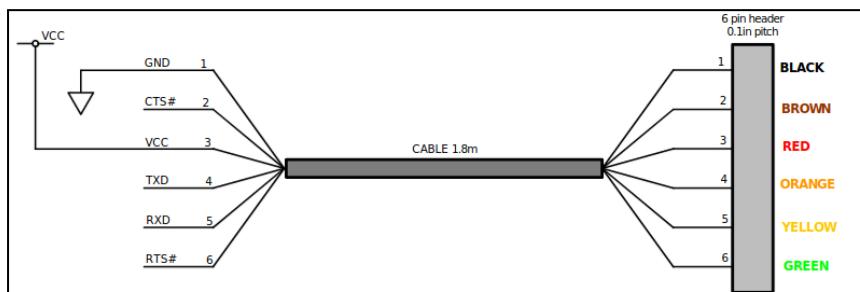


Fig. 18: Esquema de pines del cable USB-RS232

Testeo de los modelos MaxBotix

Se conectan los pines del sensores ([véase Apéndice-5.1, Fig. 5.1.1 / Apéndice-5.2, Fig. 5.2.1](#)) a los del cable USB-RS232. Para el testeo se han realizado diversos escenarios con el fin de observar la precisión de los sensores en diferentes distancias y alturas ([véase Apéndice-7](#)). En la Tabla 1 se muestran los resultados obtenidos con ambos sensores.

Tabla 1: Comparativa de casos con los modelos MaxBotix

Casos (sensor situado en una superficie)	Detecciones [mm]	
	LV-MaxSonar EZ0	HRLV-ShortRange EZ0/EZ1
Caso 1: Objeto a 135 mm	300	136
Caso 2: Objeto a 295 mm	305	297
Caso 3: Objeto a 1040 mm	1025	1043
Caso 4: Sin objeto	3437 con fluctuaciones a 767	5000 con fluctuaciones a 3791
Casos (sensor a diferentes distancias del suelo)	Detecciones [mm]	
Caso 5: Sensor a 100 mm del suelo. Objeto a 750 mm del sensor	412	379
Caso 6: Sensor a 200 mm del suelo. Objeto a 1000 mm del sensor	863	692
Caso 7: Sensor a 100 mm del suelo. Objeto a 1800 mm del sensor	Oscilación entre 1300 y 1800	1327

Se observa que el MaxBotix LV-MaxSonar EZ0 muestra buen rendimiento en distancias medias, pero presenta problemas significativos de precisión en distancias largas. Se concluye que, con el incremento de la distancia, la eficacia del sensor disminuye. Esto resulta en lecturas más variables al detectar tanto el suelo como objetos. Además, se ha identificado un error acumulativo en distancia ([véase Apéndice-7.1](#)).

Los modelos MaxBotix HRLV-ShortRange EZ0 y EZ1, se comportan de forma más precisa en distancias cortas y medias. Empiezan a detectar a partir de los 20 mm y no de los 300 mm como el anterior modelo. Este rango de medidas es crucial para que el robot detecte obstáculos en pasillos estrechos y zonas donde existe una posible colisión. Con las mismas pruebas, este sensor proporciona más fiabilidad, precisión y datos más robustos ([véase Apéndice-7.2](#)).

Testeo del modelo DFROBOT

En el caso del sensor DFROBOT SEN URM37 no se han realizado pruebas significativas, ya que resultó ser un modelo difícil de implementar y poco eficiente. Para recibir una lectura, es necesario enviar un comando de lectura en hexadecimal, transmitirlo desde el TX del USB al RX del sensor, interpretarlo en binario para devolver la lectura y convertirla en hexadecimal.

Debido a que PuTTY no permite enviar secuencias de bits, no se lograba leer datos de forma continua. Se consiguió realizar con la librería stty, pero el proceso de enviar una orden de lectura para cada dato ralentiza el sistema, convirtiéndose en una opción menos adecuada para el caso de uso actual a comparación de los modelos MaxBotix.

2.7 Conclusiones

Dado el testeo de estos tres sensores, se concluye que la gama de sensores MaxBotix HRLV-ShortRange. Concretamente los testeados MB1603 y MB1613 son la mejor opción para implementarlos en el Robin-200 por las siguientes razones:

- Los modelos ShortRange detectan distancias mínimas de 2 cm, mientras que los modelos MaxSonar distancias mínimas de 15 cm. Se observa que están diseñados para aplicaciones de menor alcance, por lo que suponen una mejor opción.
- Estos sensores han sido los más precisos y los que proporcionan menor error de detección.
- Sus detecciones son más fiables y estables en diversas alturas y ángulos.
- La configuración es más sencilla y hay mejor soporte para librerías en Python (pyserial) [31].
- Los datos se reciben constantemente sin necesidad de enviar órdenes de lectura.
- El precio de integración de los sensores es económico, por lo que se consigue el equilibrio entre eficiencia y bajo coste.

El modelo que se escoge finalmente es el MB1613 (HRLV-ShortRange EZ1) por su mejor balance en aspectos como la detección de personas, la amplitud del haz y la sensibilidad a los datos ([Apéndice-5.2, Fig. 5.2.5](#)).

Capítulo 3

Desarrollo de la integración de sensores

3.1 Introducción

Este capítulo describe el proceso de implementación del software y hardware para integrar los sensores en el producto y utilizar sus datos para la navegación.

En el aspecto del software, se detalla cómo se obtuvieron los datos de los sensores, la automatización de recepción de datos, la modificación de los archivos URDF del robot y la modificación de archivos del paquete de navegación ROS *move_base*. Finalmente, se añadió un filtro de datos, optando por el filtro Kalman tras analizar y probar varios filtros. Este procedimiento se hizo inicialmente en un sensor, y al llegar a una implementación estable, se replicó para los cuatro sensores.

En cuanto al hardware, se explica la creación de carcasa para los sensores, destacando las características necesarias para su integración eficiente. Se describe el proceso iterativo para obtener el modelo 3D final de las cuatro carcasa con tapa para el robot, destacando los problemas obtenidos en cada iteración y cómo se resuelven.

3.2 Implementación del software

En este apartado se desarrolla el software para incluir los sensores ultrasónicos en la navegación del robot. Se explica cómo se crea el paquete de ROS y cuál es el flujo de trabajo que sigue. Se comenta como se ha automatizado la obtención de datos para la posterior modificación del paquete de navegación *move_base*. Finalmente, se incluye en este paquete el script del filtro de Kalman ([véase 3.2.5](#)) para realizar un filtrado de los datos de los sensores.

3.2.1 Desarrollo del paquete de ROS *ultrasonic_sensor*

El primer paso para implementar los sensores al robot es crear un paquete en el workspace del Robin-200. En el directorio source se construye una carpeta con un archivo *CMakeLists.txt* y un *package.xml*, necesarios para compilar el paquete e incluir dependencias, entre otros. A partir de aquí se crea la carpeta *scripts* donde se encuentran todos los archivos de código Python y la carpeta *launch*, donde se crea el archivo de lanzamiento de nodos. También se pueden organizar archivos de ROS como servicios, acciones, plug-ins... pero no se ha necesitado en este desarrollo.

Al implementar este paquete al resto existentes en el sistema del Robin-200, se aprovecha la modularidad de ROS. Se facilita la organización del código y su reutilización en diferentes partes del proyecto. El código principal se desarrolla en el script [*ultrasonic_sensor.py*](#). Como se puede ver representado en la Fig. 19, el nodo se encarga de:

1. Recibir datos de los cuatro sensores conectados por un USB HUB.
2. Filtrar los datos usando el Kalman Filter.
3. Construir los mensajes Range para cada sensor [32] [33], que serán útiles para la posterior modificación del paquete *move_base*.
4. Publicar los mensajes en cuatro topics diferentes.

De esta forma los datos permiten ser recibidos por nodos mediante la suscripción a estos topics.

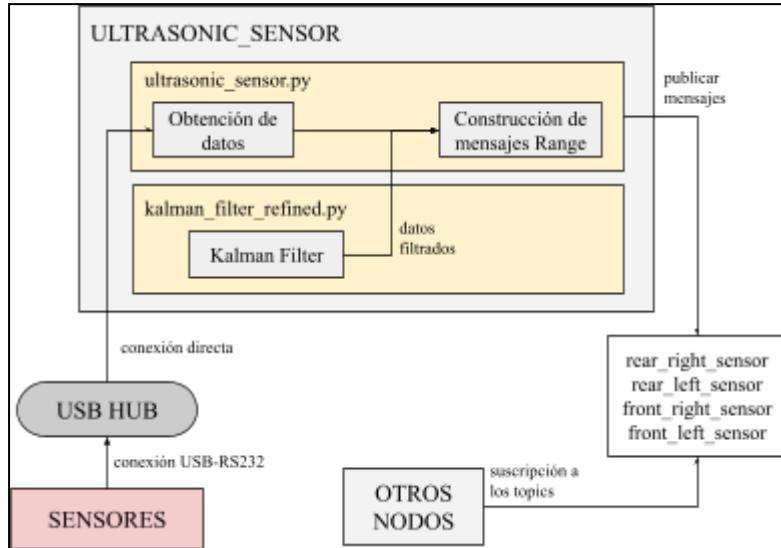


Fig. 19: Funcionamiento del nodo `ultrasonic_sensor`

3.2.2 Automatización de la detección de datos vía serial

Para gestionar datos mediante conexiones USB de manera consistente en Linux de forma automatizada, se recomienda fijar los puertos usando enlaces permanentes, independientemente del orden de conexión. Esto se logra implementando `udev rules`, el gestor de dispositivos en espacio de usuario del kernel de Linux. Permiten administrar y personalizar el comportamiento de los dispositivos conectados, asegurando que los puertos `tty` se mantengan asociados al mismo dispositivo.

Para automatizar la detección de los sensores mediante USB y fijar cuatro puertos, se crean cuatro archivos de configuración de reglas en el directorio `/etc/udev/rules` ([véase Apéndice-8](#)). Al actualizar las reglas, se observa que aparecen los `ttyUSB` especificados (Fig. 20).

```
robot@ther0-221207aa:~$ ls /dev/tty*USB*
/dev/ttyUSB0  /dev/ttyUSB3  /dev/ttyUSB_BMS      /dev/ttyUSB_IMU      /dev/ttyUSB_REARRIGHT_DS
/dev/ttyUSB1  /dev/ttyUSB4  /dev/ttyUSB_FRONTLEFT_DS  /dev/ttyUSB_LEDS
/dev/ttyUSB2  /dev/ttyUSB5  /dev/ttyUSB_FRONTRIGHT_DS /dev/ttyUSB_REARLEFT_DS
robot@ther0-221207aa:~$
```

Fig. 20: Listado de puertos `ttyUSB` del Robin-200

De esta forma, como los enlaces simbólicos están relacionados con el serial del USB, cada sensor apuntará al dispositivo serie USB al que se conecte desde el mismo enlace, por lo que la obtención de datos se realiza siempre por el mismo puerto `ttyUSB`.

3.2.3 Modificaciones del archivo de descripción URDF

Una vez se ha creado el paquete en ROS, se modifican los archivos de descripción del robot e implementan la descripción de los sensores. En el contexto de este trabajo, el archivo de descripción genérico del Robin-200 ya está creado. Se incluyen en él los archivos de descripción de sensores ([véase Apéndice-9](#)), que se pueden encontrar [en el siguiente link del repositorio del trabajo](#).

3.2.4 Integración de los datos del sensor al paquete de navegación `move_base`

Para integrar los datos de los sensores al stack de navegación como capas de obstáculos, se realizan los siguientes pasos:

1. Se publican mensajes de ROS de tipo Range (Fig. 21) en los topics creados en *ultrasonic_sensor.py*. Se asegura que la variable *float32 range* corresponde a los datos proporcionados por el sensor en tiempo real. Este tipo de mensajes vienen definidos en el paquete *range-sensor-layer* [34], que es el tipo de capa utilizado. En RViz (Fig. 22) se muestran en forma de cono, situados en el frame especificado en este caso.

```
uint8 ULTRASOUND=0
uint8 INFRARED=1
std_msgs/Header header
uint8 radiation_type
float32 field_of_view
float32 min_range
float32 max_range
float32 range
```

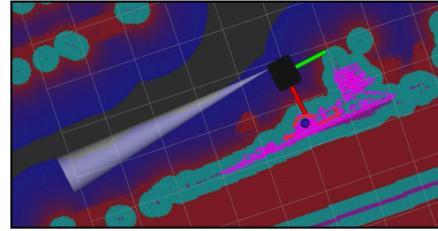


Fig. 21 Mensaje de tipo Range

Fig. 22: Visualización de mensajes Range del sensor ultrasónico en RViz

2. Se modifican los archivos del paquete *move_base* que contienen las definiciones de los costmaps a incluir durante la navegación. En este caso: *common_costmap.yaml*, *global_costmap.yaml*, *local_costmap.yaml* y *move_base.yaml*. Al añadir los sensores en estos archivos, se definen las capas de obstáculos y se incluyen en la detección de obstáculos, capa de inflación y proceso de *recovery_behaviors* ([véase 1.3](#), punto 5)
 - Archivo **common_costmap.yaml**: Se añaden cuatro capas de obstáculos donde se especifican los parámetros correspondientes según la documentación de *range-sensor-layer* [35].

```
Unset
#Se modifica el nombre para cada capa de obstáculos
ultrasonic_sensor_rear_right:
#Se modifica el topic para cada capa de obstáculos
topics: ['/keonn/distance_sensor/rear_right_sensor']
no_readings_timeout: 2.0
clear_threshold: 0.05
mark_threshold: 0.9
clear_on_max_reading: true
```

- Archivos **global_costmap.yaml** y **local_costmap.yaml**: Se añaden cuatro capas como plugins para asegurarse de que las lecturas de los sensores ultrasónicos se integren correctamente en el sistema de navegación.

```
Unset
#Se modifica el nombre para cada capa de obstáculos
- name: ultrasonic_sensor_rear_right
  type: "range_sensor_layer::RangeSensorLayer"
```

- Archivo **move_base.yaml**: Se añaden las cuatro capas de obstáculos tanto en *conservative_reset* como *aggressive_reset* para asegurar los reinicios de datos de esas capas correctamente.

Unset

layer_names: [ultrasonic_sensor_rear_right, ultrasonic_sensor_front_right, ultrasonic_sensor_rear_left, ultrasonic_sensor_rear_right, [resto de capas del Robin-200]]

Con estos cambios, el robot tiene implementados los cuatro sensores recibiendo datos y siendo procesados para su navegación. Sin embargo, durante la puesta en práctica, se identifica el siguiente problema que afecta la navegación del robot:

- En objetos con agujeros (Fig. 23 y Fig. 24) el sensor obtiene detecciones intermitentes. Esto ocurre porque detecta obstáculos cercanos pero a la vez es capaz de detectar valores más lejanos según en qué posición se encuentre el robot. Como los sensores de ultrasonidos sólo proporcionan un valor de detección, esta va oscilando entre valores, causando inestabilidad a la navegación del robot (Fig. 25 y Fig. 26).



Fig. 23: Sensores ultrasónicos al lado de un palé



Fig. 24: Sensores ultrasónicos al lado de una estantería con huecos

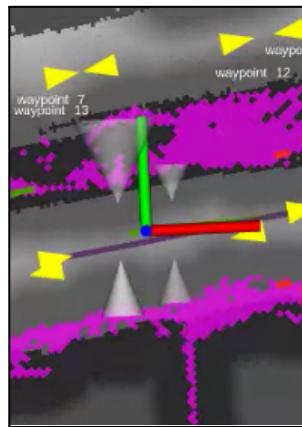


Fig. 25: Visualización en RViz. Detección 1 de los sensores

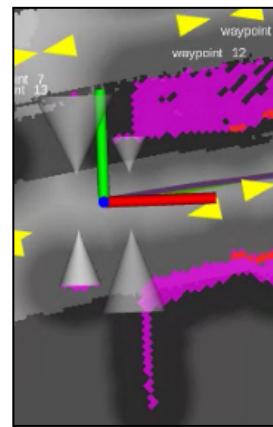


Fig. 26: Visualización en RViz. Detección 2 de los sensores

3.2.5 Implementación del filtro de datos Kalman

Introducción

Dada la necesidad de eliminar fluctuaciones en los datos de los sensores, se propone un suavizado de estos mediante un algoritmo de filtrado. Esta decisión ha sido respaldada por el estudio realizado en el artículo [36]. Se han estudiado, implementado y testeado el Moving Average Filter [37] [38], Median

Filter [39] [40] [41] y Kalman Filter [42] [43], determinando que este último es el que da mejor resultado y, por tanto, el que se aplicará en este proyecto.

Kalman Filter (KF): Estima el estado de un sistema dinámico a partir de una serie de mediciones ruidosas, siendo especialmente útil para la navegación de robots [44]. No sólo suaviza las lecturas de los sensores, sino que también predice y corrige el estado futuro del sistema ([véase Apéndice-10](#)). No obstante, es más complejo de implementar y requiere conocimiento de los modelos del sistema y del ruido.

El filtro de Kalman trabaja con diversas fuentes de datos y esto proporciona las siguientes ventajas:

1. Optimización recursiva.
2. Eficacia para manejar tanto el ruido del proceso como el de la medición.
3. Filtrado eficiente de múltiples sensores.
4. Predicción del estado futuro del sistema .
5. Corrección de las predicciones basadas en las nuevas mediciones.

Implementación del Kalman Filter en el código

Para implementar el Kalman Filter, se ha utilizado la librería de Python `filterpy.kalman`. Para definir la matriz de ruido, se ha usado la función `Q_discrete_white_noise` de la librería `kalman.common`. Los pasos para implementar el filtro se encuentran en el código `kalman_filter_refined.py` y son los siguientes:

1. Inicialización de los parámetros

Inicialización del filtro: El estado del sistema tiene dos dimensiones: posición y velocidad. Las observaciones (distancias detectadas por el sensor) tienen una dimensión: distancia.

Estimación inicial del estado: Se asume que la posición inicial es 2 metros y la velocidad inicial es 0. Se ha observado que es óptimo empezar con un valor inicial elevado, debido a que el Robin-200 al lanzar la navegación requiere de tiempo para localizarse y empezar la misión. Por lo que el algoritmo se estabiliza mucho antes. De esta forma, se evitan detecciones falsas muy cortas, generando costmap al lado del robot.

Matriz de transición de estado: Describe cómo el estado cambia de un paso de tiempo al siguiente.

Covarianza inicial del error del estado: Mapea el estado del sistema al espacio de observación. En este caso, solo la posición se observa directamente desde el sensor de ultrasonido, no la velocidad.

Covarianza inicial del error: Se observa que un valor de 100 proporciona un equilibrio entre la confianza en las mediciones iniciales y el modelo del sistema. Estabiliza las estimaciones iniciales y ajusta el filtrado a medida que se recopilan más datos. Un valor grande (1000) hará que el filtro confíe demasiado en las mediciones iniciales. Un valor pequeño (5) hará que el filtro confíe demasiado en el modelo inicial.

Covarianza del ruido de medición: Esta es la covarianza del ruido de medición. Este valor debe ser ajustado según la precisión real del sensor. Se ha observado que un valor de 0.05 es adecuado si la precisión del sensor es aproximadamente ± 0.1 metros.

Matriz de covarianza del ruido: Se inicializa el ruido bidimensional, con intervalos de updates de 0.1 s y varianza del ruido del proceso de 0.13.

Valor inicial: Se decide un valor inicial de 0.3, valor medio de detección de los sensores.

2. Predicción y filtrado

Predicción del estado: Se realiza una predicción del siguiente estado.

Detección de outliers: Se añaden condiciones para detectar outliers y filtrar posibles variaciones bruscas de lecturas de los sensores.

Actualización del filtro: Si la muestra no es un outlier, la nueva medición se actualiza.

Ajuste del estado: Se verifica la diferencia entre la muestra y el valor anterior. Si supera un threshold se ajusta directamente a la muestra actual. Si no, se ajusta como el promedio entre los dos valores.

Actualización del valor previo: Se actualiza el valor previo a la muestra detectada.

Se observa como los valores de los sensores no oscilan de forma brusca (Fig. 27 y Fig. 28).

- En la Fig. 26, los **valores reales** han sido generados mediante una distribución uniforme de 0 a 5 ($U \sim (0,5)$). Los **valores filtrados** han sido estimados usando el KF sin añadir modificaciones al algoritmo.
- En la Fig. 27, los **valores reales** han sido declarados específicamente para detectar outliers y valores no comunes. Los **valores filtrados** han sido estimados usando el modelo detallado anteriormente del KF añadiendo modificaciones y condiciones de medidas.

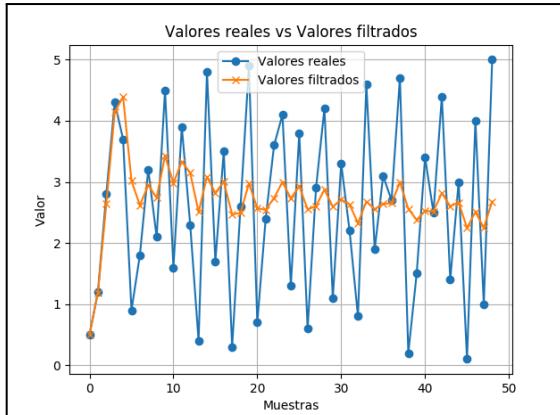


Fig. 27: Gráfico de resultados del KF sin modificar

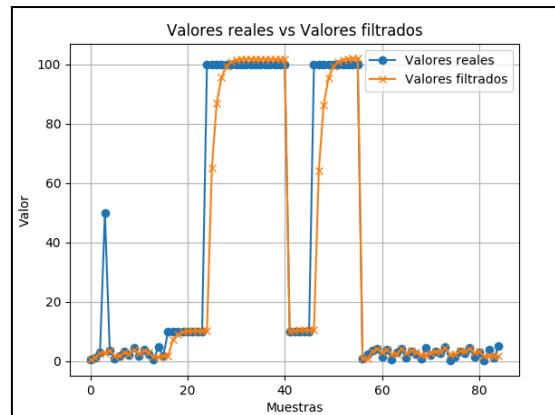


Fig. 28: Gráfico de resultados del KF modificado

También se consigue eliminar el problema de fluctuaciones de valores al implementar este filtro (Fig. 29).

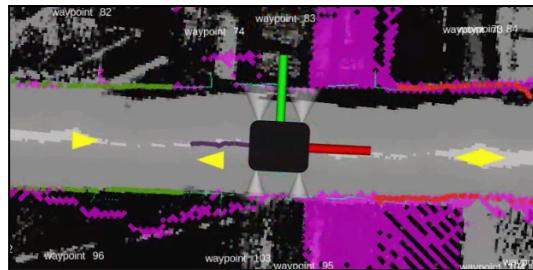


Fig. 29: Visualización en RViz del Robin-200 con detecciones constantes

3.3 Diseño del modelo 3D para la integración del sensor en el robot

En este apartado se describe la incorporación de los cuatro sensores en la estructura del robot. Se explican los prototipos iniciales y las iteraciones de mejora para llegar a un modelo de carcasa y tapa finales. Estos aseguran una integración orgánica en el diseño global del Robin-200.

3.3.1 Prototipos iniciales de diseños 3D

Para diseñar las carcasas, se utilizó el software TinkerCAD, basándose en las dimensiones de los sensores y las especificaciones del producto. Se buscó un diseño sencillo, robusto y fácil de integrar, creando dos prototipos: uno que se adhiere a la pieza superior y otro a la pieza inferior. Ambos diseños incluyen una tapa a medida que se cierra mediante clic. Se pudieron realizar varias iteraciones de impresión de los modelos hasta obtener un modelo final adecuado para las carcasas.

- **Modelo 1 (Fig. 30):** Este modelo se adhiere a la pieza inferior. Cuenta con una apertura lateral para que los cables conectados al USB-RS232 del sensor puedan salir perfectamente. También cuenta con un orificio superior central tanto en la tapa como en la carcasa para añadir tornillos de métrica 2 y que la tapa no se desprenda (Fig. 32). Este modelo va atornillado a las piezas laterales del robot con dos tornillos de métrica 3 ubicados en las extensiones traseras en forma de L de la pieza (Fig. 33).

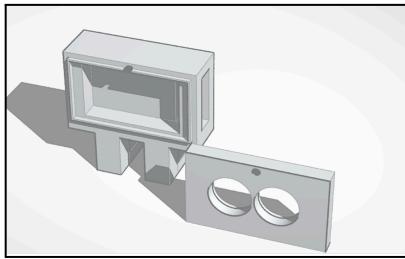


Fig. 30: Modelo 1 en 3D

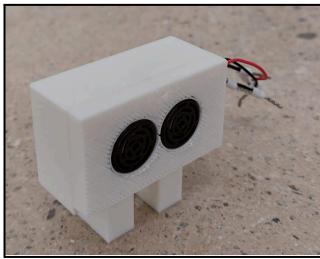


Fig. 31: Carcasa impresa

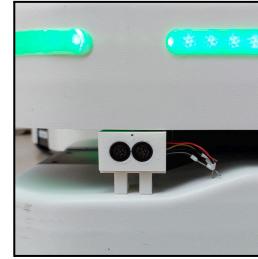


Fig. 32: Sensor en el robot con el modelo 1

- **Modelo 2 (Fig. 33):** Este modelo se adhiere a la pieza superior. También cuenta con la apertura lateral y el orificio superior central tanto en la tapa como en la carcasa para añadir tornillos de métrica 2 (Fig. 34). Este modelo va atornillado a la pieza donde se ubican los leds del robot mediante dos tornillos de métrica 3 ubicados en las extensiones superiores en forma de I de la pieza (Fig. 35).

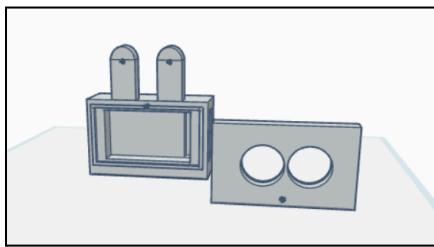


Fig. 33: Modelo 2 en 3D

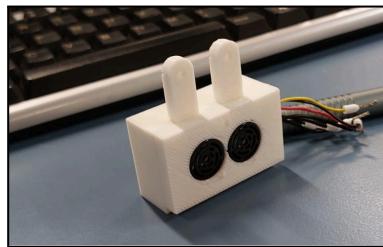


Fig. 34: Carcasa impresa



Fig. 35: Sensor en el robot con el modelo 2

Tras imprimir los dos modelos de carcasa, se determinó que ambos consiguen integrar el sensor en los laterales del robot. No obstante, la pieza inferior de la base está sujetada por cuatro tornillos. Se trata de una pieza móvil, que proporciona inestabilidad de detección al usar el primer modelo 3D. Sin embargo, el segundo modelo es más estable, ya que está atornillado a la pieza superior, que es más rígida y se adhiere mejor a la base. Se reduce la posibilidad de colisión con otros objetos y se obtiene más seguridad para los sensores. También queda mejor integrado y sobresalta menos. Por ello, se eligió el segundo modelo para el diseño final de las carcasa.

3.3.2 Iteraciones de mejora del modelo

Una vez elegido el modelo, se detectan las siguientes imperfecciones a mejorar:

- El sensor sobresale de la parte lateral del robot.
- Las pestañas superiores quedan muy altas.
- El acabado de la parte inferior no había quedado pulido. La zona inferior no es plana y la distancia de los extremos con la ranura superior y la inferior eran diferentes.
- Se recomienda realizar dos agujeros en el interior de la pieza para fijar mejor el sensor.

Estos fallos se corregen en el siguiente modelo mediante los siguientes cambios:

- Se incrementa una pestaña superior en la carcasa para avanzar las pestañas y al atornillar la carcasa el sensor no sobresalga de la línea lateral del robot.
- Se acortan las pestañas superiores.
- Se realizan dos agujeros extra en el interior de la carcasa para atornillar el sensor.
- Se cambian las dimensiones de la tapa para poder encargar con la carcasa.
- Se reubica el tornillo tapa/carcasa a la zona inferior.
- Se crean dos diseños diferentes de la tapa con encajes diferentes.
 - El modelo 1 tiene el encaje por todos lados.

- El modelo 2 tiene el encaje solo a los laterales.

Se mejora el diseño (Fig. 36) pero empeora en la forma de cómo cierran las tapas debido al material de impresión y a las medidas tan pequeñas entre los laterales de la carcasa y el encaje. Se observa lo siguiente:

- Los acabados laterales cuentan con muchas imperfecciones (Fig. 37).
- Los encajes se acaban rompiendo.
- El tornillo deforma la unión entre la tapa y la carcasa (Fig. 38).
- Al atornillar la tapa, se puede levantar la parte superior de la tapa debido a la fragilidad del material.

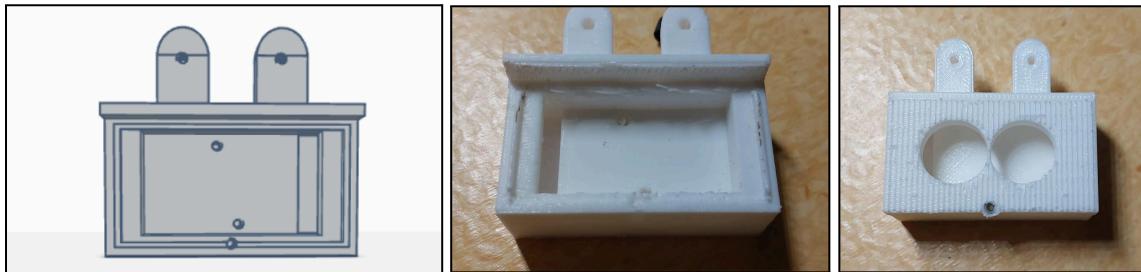


Fig. 36: Modelo 3D de la segunda iteración

Fig. 37: Modelo impreso con tapa puesta

Fig. 38: Modelo impreso defectuoso

Después de reiterar sobre este modelo, se realizan los siguientes ajustes:

- El encaje solo se sitúa en los costados tanto de la carcasa como de la tapa. Estos no son unificados, sino que en medio hay un agujero para atornillar la tapa de la pieza (Fig. 39).
- Los dos tornillos que unen la tapa de la carcasa proporcionan más robustez al modelo.
- Se realizan dos modelos la apertura lateral izquierda y derecha para conectar los pines de los sensores con los USB de forma más accesible. Se imprimen dos unidades de cada modelo.

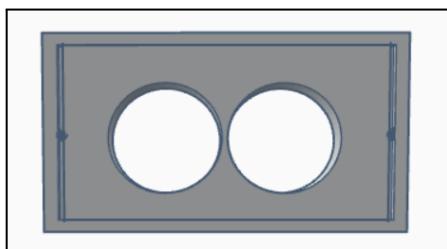


Fig. 39: Modelo 3D de la tapa

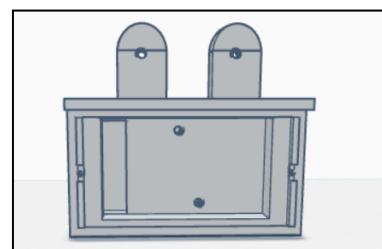


Fig. 40: Modelo 3D final de la carcasa

Al resolver estos ajustes, se obtiene el modelo 3D de la carcasa final (Fig. 40). En el caso de la tapa, se observa en el siguiente apartado cómo se modifica una vez se ha testeado en la navegación del Robin-20

3.3.3 Modelo 3D final

Con el modelo finalizado, se imprimen los cuatro modelos de carcasa y tapa en 3D y se montan en la pieza superior del Robin-200 (Fig. 41). Se añaden los tornillos necesarios para fijar el sensor a la pieza (Fig. 42, Fig. 43 y Fig. 44). Se conectan con los cuatro cables USB-RS232 a su vez conectados a un USB HUB comunicándose con el puerto USB del PC del robot (Fig. 45). Finalmente, se vuelve a atornillar la pieza superior al robot (Fig. 46 y Fig 47).

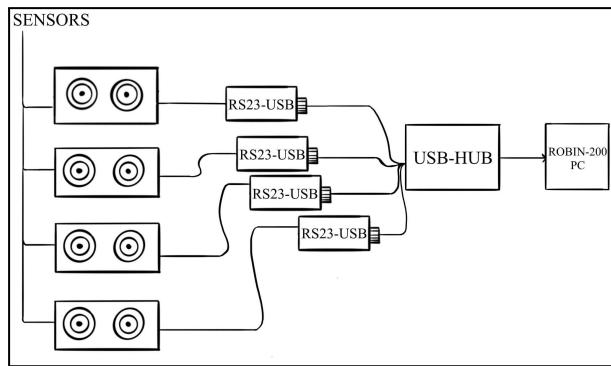


Fig. 41: Esquema de conexión entre sensores y PC del Robin-200

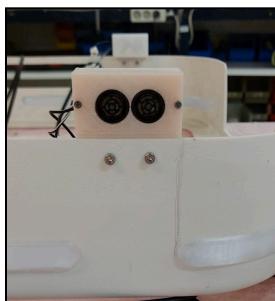


Fig. 42: Carcasa con sensor atornillada. Vista frontal

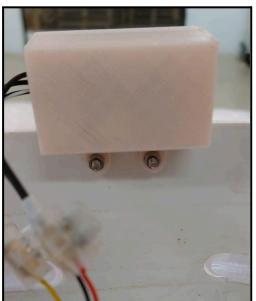


Fig. 43: Carcasa con sensor atornillada. Vista trasera



Fig. 44: Pieza superior con los cuatro sensores integrados



Fig. 45: Sensores conectados al USB HUB



Fig. 46: Sensores integrados en el lateral izquierdo



Fig. 47: Sensores integrados en el lateral derecho

Modificación final del diseño de la tapa

Al añadir los sensores con las tapas diseñadas, se llevan a cabo unas pruebas básicas y se observan los siguientes errores:

- El sensor detecta mucho ruido al rotar (Fig. 48) y al girar (Fig. 49), provocando un barrido y generando un círculo o un arco de obstáculos.

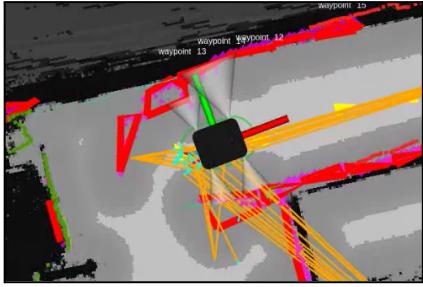


Fig. 48: Visualización de RViz del Robin-200 generando obstáculos al rotar

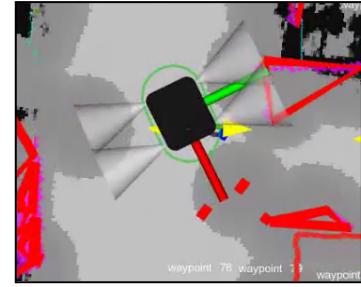


Fig. 49: Visualización de RViz del Robin-200 generando obstáculos al girar

- En pasillos con diversos objetos, el sensor no corrige bien las medidas cuando detecta un área con obstáculos seguida de un área libre (Fig. 50). Esto provoca un barrido lineal de obstáculos cuando realmente no los hay.



Fig. 50: Visualización de RViz del Robin-200 generando obstáculos en un área libre del pasillo

Considerando los errores observados, el equipo de soporte de MaxBotix recomendó reducir un poco el ángulo de apertura del sensor. Es por eso que se decide realizar una reducción del diámetro de los agujeros en la tapa de la carcasa.

En la Fig. 51 se muestra el método usado para obtener el ángulo exacto de los sensores en el robot. Se calcula que $a = 190$ mm (distancia del sensor al suelo); $b = 750$ mm en estático, con ángulo de apertura $2\alpha = 2 \cdot \text{arctg}(192/723) = 2 \cdot 15.76 = 31.53^\circ$.

A partir de aquí, se añaden unos conos de reducción de diámetro en las tapas de las carcasa para disminuir el área de propagación de las ondas emitidas (Fig. 52) [45]. Se propone que $a = 2$ mm; $b = 8$ mm, obteniendo un ángulo de apertura de $\alpha = 20.55^\circ$ y consiguiendo una reducción de ángulo del 34.8% .

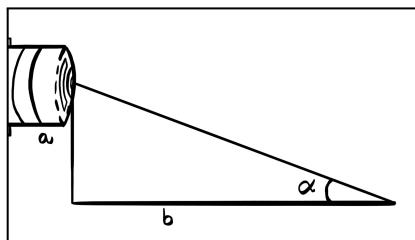


Fig. 51: Escenario inicial del sensor en el robot

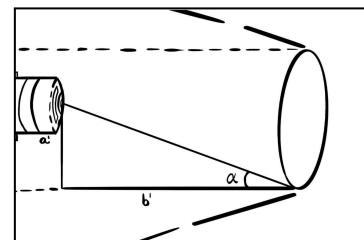


Fig. 52: Escenario modificado en la tapa del sensor

Con la modificación, se observa que se resuelven los errores detectados:

- Los sensores no detectan ruido al rotar o al girar bruscamente (Fig. 53).
- Los sensores no realizan un barrido de datos al pasar por pasillos que no están completamente delimitados por obstáculos (Fig. 54).

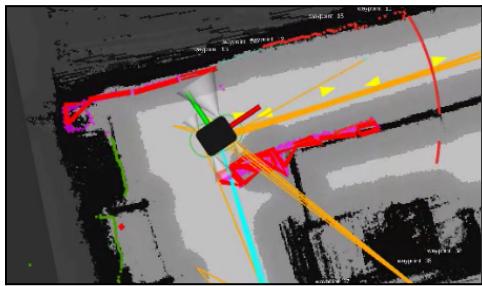


Fig. 53: Visualización en RViz del Robin-200 rotando sin detectar ruido

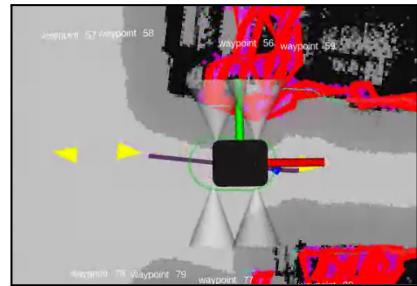


Fig. 54: Visualización en RViz del Robin-200 en un pasillo sin generar ruido en espacios libres

Finalmente, se integra el nuevo modelo de tapa al robot (Fig. 55, Fig. 56 y Fig. 57):

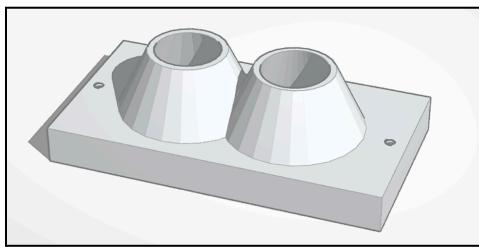


Fig. 55: Modelo 3D final de la tapa, con reducción de ángulo de apertura



Fig. 56: Sensores del lateral izquierdo del robot con la nueva tapa



Fig. 57: Sensores del lateral derecho del robot con la nueva tapa

Se realizan planos en 2D para mostrar las medidas de las piezas creadas ([véase Apéndice-11](#)). Los modelos se pueden descargar [en esta sección del repositorio](#).

Capítulo 4

Pruebas de navegación y testing

4.1 Introducción

En este capítulo se presentan las pruebas de navegación del Robin-200 en el estado inicial y, posteriormente, con la configuración final de sensores implementada. Se considera la configuración final aquella que incluye el filtro de Kalman y el filtro físico de la tapa.

Se presentan tres escenarios distintos, en los que el robot navegará, y en los que obtienen conclusiones numéricas respecto a cada escenario para determinar si ha habido una mejora de navegación y, por tanto, si la [hipótesis 1](#) se ha cumplido.

- La primera prueba consiste en lanzar diez goals en pasillos bloqueados por objetos (Fig. 58 y Fig. 59).



Fig. 58: Escenario inicial de testeo. Pasillo con dos cajas que bloquean el paso del Robin-200

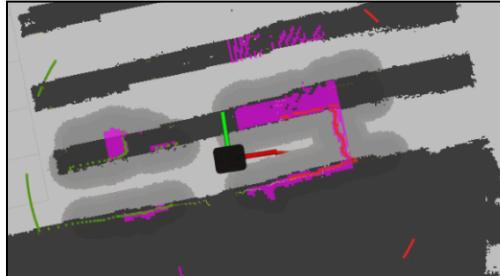


Fig. 59: Captura de RViz mostrando el escenario inicial

- La segunda prueba consiste en lanzar cinco misiones de inventario sin pasillos bloqueados. Se busca observar si en un escenario favorable el robot tiene dificultades para navegar por un mapa con un total de 66 goals (Fig. 60) y si con sensores laterales mejora su navegación.

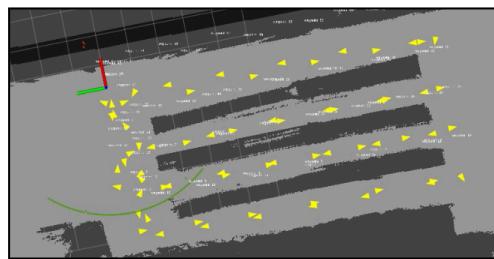


Fig. 60: Visualización del escenario 2 en RViz

- La última prueba consiste en realizar cinco misiones de inventario con dos pasillos bloqueados (Fig. 61). Se observa cómo se gestiona la navegación del Robin-200 sin sensores ultrasónicos en un entorno desfavorable y si se observan mejoras al añadirlos.

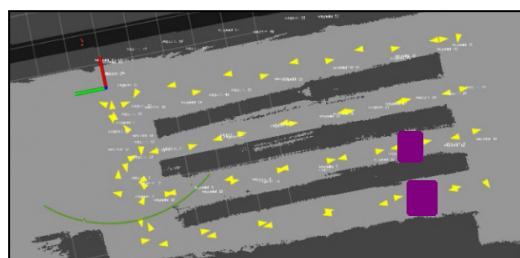


Fig. 61: Visualización del escenario 3 en RViz

Con esta información se obtienen los resultados necesarios para determinar si la implementación de los sensores ha mejorado la navegación del Robin-200.

4.2 Escenario 1

Por un lado, se observa el siguiente comportamiento en el Robin-200 sin sensores ultrasónicos:

1. Avanza hacia el goal hasta que las cajas obstruyen completamente el paso.
2. Genera un global plan por los laterales de él mismo, debido a que el entorno cargado está en escala de grises. No se detectan obstáculos a sus laterales (Fig. 62). Es decir, realiza un **replanning lateral**.
3. Rota hasta detectar los pasillos por los sensores frontales y traseros, mientras que el local planner va replaneando la ruta por sus laterales.
4. Al encararse a los pasillos, se dejan de detectar las cajas, porque están a los laterales del robot. Por tanto, se replanea por encima de ellas (Fig. 63).
5. Se genera un bucle entre los pasos 3-4, impidiendo que el robot llegue nunca a su objetivo.

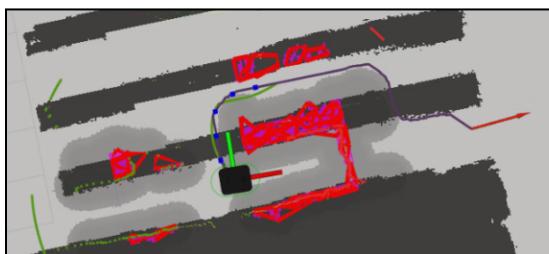


Fig. 62: Visualización en RViz de replaneo lateral



Fig. 63: Visualización en RViz de replaneo por las cajas una vez rotada

Por otro lado, se observa el siguiente comportamiento en el Robin-200 con sensores ultrasónicos:

1. Avanza hacia el goal hasta que las cajas obstruyen completamente el paso.
2. Replanea rodeando el pasillo, o por una zona donde ni el láser frontal, ni las cámaras frontal y trasera, ni los sensores laterales son capaces de detectar (Fig. 64).
3. Se desplaza por el entorno a la vez que va actualizando el global y local planners (Fig. 65).
4. Se repite el paso 3 hasta llegar a su objetivo.

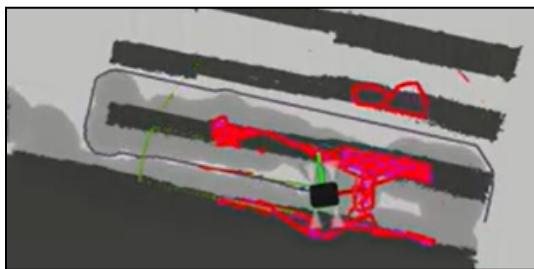


Fig. 64: Visualización de RViz del Robin-200 actualizando su planner rodeando el pasillo

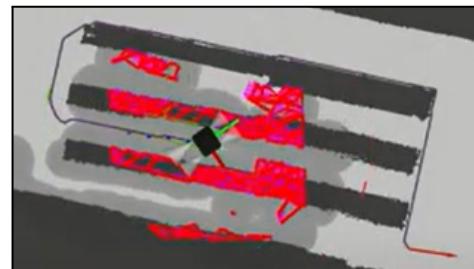


Fig. 65: Visualización de RViz del Robin-200 moviéndose por el espacio actualizando el planner

Se muestran los resultados obtenidos en cada iteración en ambos casos (Tabla 2). Se calcula si el robot ha llegado al objetivo, el tiempo de llegada en caso satisfactorio, el número de replannings y los valores medios de estas medidas.

Tabla 2: Resultados del escenario 1

Iters.	Llegada al GOAL	Tiempo de llegada	# replannings	Tiempo medio de llegada	% de efectividad	# medio replannings
Resultados en el caso del Robin-200 sin sensores ultrasónicos						
Iter. 1	No	-	-	-	0	-
Iter. 2	No	-	-			
...						
Iter. 9	No	-	-	-	0	-
Iter. 10	No	-	-			
Resultados en el caso del Robin-200 con sensores ultrasónicos						
Iter. 1	Sí	4:23	6	3:10 min	80	2,25
Iter. 2	Sí	2:34	3			
Iter. 3	Sí	2:56	0			
Iter. 4	Sí	1:58	0			
Iter. 5	No	-	-			
Iter. 6	Sí	4:31	2			
Iter. 7	Sí	3:01	2			
Iter. 8	Sí	2:43	4			
Iter. 9	Sí	3:17	1			
Iter. 10	No	-	-			

En el caso de las pruebas sin sensores, se observa un escenario muy desfavorable para el correcto rendimiento del robot. En ninguno de los casos llega al goal y en todos ellos se observa indefinidamente el bucle de replaneos laterales.

En el caso de las pruebas con sensores, se observa una mejora evidente (Tabla 3). En los casos que no ha conseguido llegar al goal, no se ha podido contabilizar el número de replannings laterales debido a que corrige la ruta constantemente mientras rota.

En el primer caso, todos los tests han acabado en un bucle de replannings laterales. En el segundo caso, se observa cómo el robot consigue llegar al 80% de goals.

4.3 Escenario 2

En este escenario, se ha observado el tiempo de inicio y final de misión, total de goals y goals consecutivos cancelados, tags leídos y número de replannings laterales realizados en cada uno de los cinco inventarios. Se realizan pruebas del robot sin y con sensores y se extraen resultados (Tabla 3).

Tabla 3: Resultados del escenario 2

Misión	Inicio de misión [min]	Final de misión [min]	Tiempo transcurrido [min]	GOALS cancelados de 66	GOALS consecutivos cancelados	TAGs leídos	# replannings laterales
Resultados en el caso del Robin-200 sin sensores ultrasónicos							
Misión 1	07:00:02	07:18:03	18:01	1	1	3416	3
Misión 2	08:00:01	08:16:43	16:42	0	0	3493	1
Misión 3	09:00:01	09:19:32	19:31	2	1	3375	4
Misión 4	11:15:01	11:31:22	16:21	1	1	3181	5
Misión 5	11:45:01	12:04:34	19:33	2	1	3394	5
Resultados en el caso del Robin-200 con sensores ultrasónicos							
Misión 1	10:30:01	10:46:11	16:10 min	0	0	3512	0
Misión 2	18:47:51	19:04:13	16:22 min	0	0	3075	0
Misión 3	19:34:28	19:53:22	18:54 min	1	1	3379	0
Misión 4	18:22:07	18:40:08	18:01 min	1	1	3273	0
Misión 5	19:07:27	19:26:49	19:22 min	1	1	3316	0

Se observa que la mejora más relevante es la reducción total de replannings laterales. Lo cual significa que el robot tiene un mejor conocimiento de por donde puede circular, aumentando la precisión y fiabilidad de su planning. En un entorno sin pasillos bloqueados, el robot inicialmente ya proporciona buenos resultados. No obstante, se consigue mejorar ligeramente los resultados.

Tabla 4: Conclusiones numéricas del escenario 2

	Tiempo promedio [min]	Promedio goals cancelados	Promedio goals consecutivos cancelados	Promedio TAGs	Promedio replannings
Caso 1: Sin sensores	18:01	1.2	0.8	3372	3.6
Caso 2: Con sensores	17:46	0.6	0.6	3312	0

A partir de las pruebas realizadas extraen los siguientes resultados (Tabla 4):

- Se reduce en un 1,46% el tiempo de inventarios.
- Se reduce un 50% la cancelación de goals.
- Se reduce un 25% la cancelación consecutiva de goals.
- Se reduce un 1.57% en la cantidad de lecturas de tags RFID.
- Se reduce un 100% los replannings laterales.

4.4 Escenario 3

En este escenario, también se ha observado el tiempo de inicio y final de misión, total de goals y goals consecutivos cancelados, tags leídos y número de replannings laterales realizados en cada uno de los cinco inventarios (Tabla 5). No obstante, la trayectoria que realiza el Robin-200 está bloqueada por cajas en dos de cuatro pasillos, que impiden la navegación libre del robot.

Tabla 5: Resultados del escenario 3

Misión	Inicio de misión [min]	Final de misión [min]	Tiempo transcurrido [min]	Goals cancelados de 66	Goals consecutivos cancelados	TAGs leídos	# replannings laterales
Resultados en el caso del Robin-200 sin sensores ultrasónicos							
Misión 1	19:20:27	19:50:16	29:49	22	13	3785	+50
Misión 2	20:00:01	20:35:10	35:09	23	23	3288	+50
Misión 3	21:00:01	21:31:27	31:26	13	7	3500	+50
Misión 4	15:00:01	15:21:27	21:26	6	3	3288	+50
Misión 5	08:00:01	08:34:37	34:37	17	7	3334	+50
Resultados en el caso del Robin-200 con sensores ultrasónicos							
Misión 1	14:20:01	14:44:08	24:07 min	7	3	3330	13
Misión 2	15:00:01	15:21:27	21:26 min	5	3	3309	7
Misión 3	15:30:01	15:55:42	25:41 min	9	4	3276	11
Misión 4	17:00:01	17:25:01	25 min	14	9	3316	23
Misión 5	17:47:37	18:09:28	21:51 min	4	2	3329	14

En este escenario se observa una mejora general al implementar los sensores. Esto supone disminuir el tiempo transcurrido, aumentar el número de goals conseguidos y disminuir los replannings laterales. Se observa una leve disminución de lecturas de tags RFID. El robot sin sensores tarda más en hacer una misión de inventario, por lo que está más tiempo con el sistema RFID activo y, por ende, obtiene más lecturas. No obstante, las lecturas proporcionadas por el robot con sensores son más estables y no fluctúan tanto.

Tabla 6: Conclusiones numéricas del escenario 3

	Tiempo promedio [min]	Promedio goals cancelados	Promedio goals consecutivos cancelados	Promedio TAGs	Promedio replannings
Caso 1	30:29	16.2	10.6	3439	+50
Caso 2	23:37	7.8	4.2	3312	13.6

A partir de las pruebas, se extraen los siguientes resultados (Tabla 6):

- Se reduce el tiempo de inventario en un 25,43%.
- Se reduce un 51,81% la cancelación de goals.

- Se reduce un 60.37% la cancelación consecutiva de goals.
- Se reduce en un 3.57% la cantidad de lecturas de tags RFID
- Se reduce más de un 72.8% el promedio de replannings laterales.

4.5 Resultados y conclusiones

Teniendo en cuenta los resultados obtenidos en los apartados 4.1, 4.2 y 4.3 se obtienen las siguientes conclusiones:

- a. Los sensores pueden ayudar a optimizar ligeramente la ruta del robot incluso en entornos sin obstrucciones, pero son especialmente útiles en entornos bloqueados, permitiendo encontrar rutas alternativas eficientemente.
- b. La reducción de cancelación de goals demuestra que los sensores son útiles para mantener al robot en curso en situaciones complicadas.
- c. Los sensores permiten una navegación más estable y predecible, reflejado tanto en entornos bloqueados como libres.
- d. Las lecturas de RFID son más estables pero ligeramente menores debido a una ruta optimizada a pesar de los bloqueos. Aún así, la diferencia no es significativa y se mantiene dentro de un rango eficiente para la operación del robot.

Capítulo 5

Conclusiones y trabajo futuro

El objetivo del proyecto, la mejora de navegación del Robin-200, se ha cumplido. Se consigue reducir significativamente el número de replannings laterales por falta de detección en objetos. También se consigue mejorar el producto y hacerlo más eficiente en entornos difíciles para la navegación, y por ello, más autónomo. Se consigue implementar una solución eficiente y económica, evitando incrementar excesivamente el coste del Robin-200.

Durante el desarrollo de este proyecto, se ha realizado un análisis completo de sensorización del robot para la posterior integración con el producto. Todo este trabajo respalda los resultados positivos obtenidos en las pruebas de navegación. Al desarrollar un paquete en el entorno de ROS y automatizar el proceso de obtención de datos, se consigue incluir la sensorización a la navegación del Robin-200. No obstante, esto no es suficiente para llegar a la solución más eficiente. Para ello, implementar un filtrado de datos con el Kalman Filter ha aportado una mejora notable de detección de obstáculos, suavizando los valores de lectura y proporcionando más robustez al sistema.

Adicionalmente, la implementación de una carcasa física con tapa, atornillada directamente a la base del Robin-200 también ha resultado en un desarrollo favorable para la mejora. Al fijar los dispositivos se consigue aportar más estabilidad y seguridad. Se reduce la posibilidad de que factores externos condicionen las lecturas de los sensores ultrasónicos. Con ello, se consigue un robot autónomo más adaptable a entornos cambiantes y proporciona mejores resultados de inventario.

Se concluye que el rendimiento de la navegación del Robin-200 mejora notablemente en comparación con el estado previo a este trabajo. Se han trabajado las hipótesis propuestas y se han afirmado, resultando en un robot más autónomo y eficiente para adaptarse a entornos de retail.

Respecto al trabajo futuro, se proponen los siguientes aspectos a analizar y desarrollar: En primer lugar, se propone implementar el desarrollo al producto de manera estable, de forma que las futuras unidades del Robin-200 ya cuenten con un sistema de cobertura lateral con sensores ultrasónicos perfectamente integrado. Se propone seguir trabajando en el desarrollo de este proyecto de manera activa.

Como futuras mejoras a estudiar, se sugiere realizar un análisis más completo de filtrado con diferentes sensores. Se han obtenido resultados favorables aplicando un filtro de Kalman a sensores ultrasónicos, pero no se han extraído datos al combinar diferentes filtros con diferentes sensores. Estas pruebas podrían ser más favorables y mejorar aún más el rendimiento de la navegación Robin-200.

Finalmente, queda pendiente realizar pruebas continuadas en un entorno de retail real, para observar la navegación del robot y cómo responde a cambios constantes del entorno dado un mismo mapa estático. Se propone estudiar las variables tenidas en cuenta en las pruebas de navegación para extraer conclusiones directamente relacionadas con un caso real.

Bibliografía

- [1] Wikipedia. (2024, 23 enero). *RFID*. Wikipedia, la Enciclopedia Libre. <https://es.wikipedia.org/wiki/RFID>
- [2] ¿Qué es RFID cómo funciona y en qué se utiliza? Ventajas Principales. (2023, 12 abril). Nephos IT. <https://www.nephost.com/que-es-rfid-como-funciona-y-en-que-se-utiliza/>
- [3] Aula. (2023, 15 mayo). RFID: todo lo que necesitas saber | Aula21. *aula21 | Formación para la Industria*. <https://www.cursosaula21.com/que-es-el-rfid/>
- [4] Keonn Technologies. (2024, 14 mayo). *AdvanReader-160 : High power UHF reader with an on-board computer*. Keonn. <https://keonn.com/components-product/advanreader-160/>
- [5] Keonn Technologies. (2024, abril 10). *Advantenna-SP11 : High gain wide beam antenna*. Keonn. <https://keonn.com/components-product/advantenna-sp11/>
- [6] Keonn Technologies. (2024, mayo 27). *Robin : RFID robot for inventory and location*. Keonn. <https://keonn.com/systems-product/robin-200/>
- [7] *Robin-200 RFID robot for automatic inventory*. (s. f.). Keonn Technologies. <https://keonn.com/wp-content/uploads/Keonn-Robin-200-Data-sheet.pdf>
- [8] Isaac. (2021, 17 diciembre). ROS: el sistema operativo para robótica. *Hardware Libre*. <https://www.hwlibre.com/ros/>
- [9] Mazzari, V. (2023, 27 abril). *ROS - Robot Operating System*. Génération Robots - Blog. <https://www.generationrobots.com/blog/en/ros-robot-operating-system-2/>
- [10] Robotics, C. (s. f.). *ROS 101: Intro to the Robot Operating System - Robohub*. <https://robohub.org/ros-101-intro-to-the-robot-operating-system/>
- [11] Delgado, D. O. (2017, septiembre 21). Qué es ROS (Robot Operating System). *OpenWebinars.net*. <https://openwebinars.net/blog/que-es-ros/>
- [12] *rviz - ROS Wiki*. (s. f.-b). <http://wiki.ros.org/rviz>
- [13] *Rviz, descripción de las herramientas. Desarrollo de un robot para la caracterización y tr.* (s. f.). <https://1library.co/article/rviz-descripci%C3%B3n-herramientas-desarrollo-robot-caracterizaci%C3%B3n-tr.q76rw5ny>
- [14] *XML Robot Description Format (URDF) - Modelado de la silla*. (s. f.). <https://1library.co/article/xml-robot-description-format-urdf-modelado-de-silla.8ydd1rey>
- [15] *Tipos de sensores de proximidad*. (s. f.). TECNOLOGIA ELECTRONICA. <https://tecnologiaelectron.blogspot.com/2014/04/tipos-de-sensores-de-proximidad.html>
- [16] *Sensors - ROS Wiki*. (s. f.). <http://wiki.ros.org/Sensors>
- [17] Sme, T. C. (2024a, mayo 10). Mastering Proximity Sensor Types: A Comprehensive guide. *Techie Science*. <https://es.lambdageeks.com/proximity-sensor-types/>
- [18] *Using IR Sensors in Navigation - ROS Answers: Open Source Q&A Forum*. (s. f.). <https://answers.ros.org/question/129581/using-ir-sensors-in-navigation/>

- [19] WatElectronics. (2023, 23 octubre). *Ultrasonic Sensor : working, Specifications, benefits & its applications.* WatElectronics.com. <https://www.watelectronics.com/ultrasonic-sensor/#:~:text=Knowing%20the%20specifications%20of%20an%20ultrasonic%20sensor%20helps,milliseconds.%20The%20Beam%20angle%20is%20around%205%200>.
- [20] MaxBotix. (s. f.). *LV-MaxSonar-EZ Datasheet.* <https://maxbotix.com/pages/lv-maxsonar-ez-datasheet>
- [21] MaxBotix. (s. f.-a.). *HRLV-ShortRange-EZ Datasheet.* <https://maxbotix.com/pages/hrlv-shortrange-ez-datasheet>
- [22] DFRobot. (2020, 7 abril). DFRobot URM Ultrasonic Distance Sensor Selection Guide. *dfrobot.com.* <https://www.dfrobot.com/blog-1491.html>
- [23] dfrobot.com. (s. f.). *Fermion: URM37 Ultrasonic Distance Sensor Breakout (2~800cm, RS232 / UART).* <https://www.dfrobot.com/product-53.html>
- [24] *URM37_V5.0_Ultrasonic_Sensor_SKU_SEN0001_DFRobot.* (s. f.). https://wiki.dfrobot.com/URM37_V5.0_Ultrasonic_Sensor_SKU_SEN0001_
- [25] Admin. (2020, 21 noviembre). *Protocolo de comunicación UART – ¿Cómo funciona?* IWOFR. <https://iwofr.org/es/protocolo-de-comunicaci%C3%B3n-uart-c%C3%B3mo-funciona/>
- [26] Tritek. (2023, 30 octubre). *¿Necesita comunicación RS485 en baterías de litio?* Ebike Battery Manufacturer. <https://tritekbattery.com/es/rs485-communication-in-lithium-batteries/>
- [27] Workshop, D. (2023, 12 abril). *HC-SR04 Ultrasonic Distance Sensor with Arduino | DroneBot Workshop.* DroneBot Workshop. <https://dronebotworkshop.com/hc-sr04-ultrasonic-distance-sensor-arduino/>
- [28] Parallax. (2024, 13 junio). *PING))) Ultrasonic Distance Sensor - Parallax.* <https://www.parallax.com/product/ping-ultrasonic-distance-sensor/>
- [29] *Parallax PING))) specifications.* (s. f.). Mouser. <https://www.mouser.com/datasheet/2/321/28015-PING-Sensor-Product-Guide-v2.0-461050.pdf>
- [30] *XXS18P1PM12 Ultrasonic Sensor Data Sheet.* (s. f.). Distributor Data Solutions. <https://media.distributordatasolutions.com/schneider2/2021q1/documents/97edee3a8bd65ff557f4a70101475c7a6c4933b9.pdf>
- [31] *Welcome to pySerial's documentation — pySerial 3.4 documentation.* (s. f.). <https://pyserial.readthedocs.io/en/latest/>
- [32] *sensor_msgs/Range Documentation.* (s. f.). https://docs.ros.org/en/melodic/api/sensor_msgs/html/msg/Range.html
- [33] *range_sensor_layer: range_sensor_layer::RangeSensorLayer Class Reference.* (s. f.). https://docs.ros.org/en/jade/api/range_sensor_layer/html/classrange_sensor_layer_1_1RangeSensorLayer.html
- [34] DLu. (s. f.). *GitHub - DLu/navigation_layers.* GitHub. https://github.com/DLu/navigation_layers

[35] *range_sensor_layer - ROS Wiki*. (s. f.). https://wiki.ros.org/range_sensor_layer

[36] *Implementation of Kalman Filter for Stabilizing Ultrasonic Sensor Reading on Distilled Ethanol Levels Measurement*. (2023, 15 noviembre). IEEE Conference Publication | IEEE Xplore. <https://ieeexplore.ieee.org/document/10366700>

[37] Wikipedia. (2024, 20 marzo). *Moving average*. Wikipedia. https://en.wikipedia.org/wiki/Moving_average

[38] *Moving average filters*. (s. f.). <https://www.dspguide.com/ch15.htm>

[39] Wikipedia. (2024b, mayo 11). *Median filter*. Wikipedia. https://en.wikipedia.org/wiki/Median_filter

[40] Gonzalez, S. (2023, 5 agosto). *Understanding the Median Filter: A Powerful Tool for Noise Reduction in Image Processing*. Salvatore Labs. <https://blog.salvatorelabs.com/understanding-the-median-filter-a-powerful-tool-for-noise-reduction-in-image-processing/>

[41] *1-D median filtering - MATLAB medfilt1 - MathWorks India*. (s. f.). <https://in.mathworks.com/help/signal/ref/medfilt1.html>

[42] Wikipedia. (2024c, junio 13). *Kalman filter*. Wikipedia. https://en.wikipedia.org/wiki/Kalman_filter

[43] *Kalman filter example | Lulu's blog*. (s. f.). <https://lucidar.me/en/kalman-filters/example-of-kalman-filter/>

[44] Michel van Biezen. (2015, 12 septiembre). *Special Topics - The Kalman Filter (1 of 55) What is a Kalman Filter?* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=CaCcOwJPytQ>

[45] *Reducción de razones trigonométricas al primer cuadrante. Circunferencia unitaria de 16 puntos. Ejemplos resueltos*. (s. f.). https://calculo.cc/temas/temas_trigonometria/trigonometria-teoria/trigo_reduccion.html

Apéndice

Apéndice 1: Stack de navegación.....	36
Apéndice 2: Tablas de características de sensores.....	38
Apéndice 3: Comparativa de precios de sensores en el mercado.....	40
Apéndice 4: Análisis del número de sensores.....	41
Apéndice 5: Especificaciones de los modelos estudiados.....	44
Apéndice 6: Formato de protocolo UART.....	48
Apéndice 7: Testeo de sensores.....	49
Apéndice 8: Archivos .rules de udev.....	52
Apéndice 9: Modificaciones del Xacro del Robin-200.....	53
Apéndice 10: Algoritmo de Kalman Filter.....	54
Apéndice 11: Planos 2D de los modelos 3D finales.....	55

Apéndice 1: Stack de navegación

En este apéndice se profundiza en el stack de navegación de *move_base* que usa el Robin-200, especificando los nodos y su funcionamiento, el flujo de trabajo del paquete [1] [2] y qué cambios se realizan en el Robin-200 respecto a la interpretación de mapas.

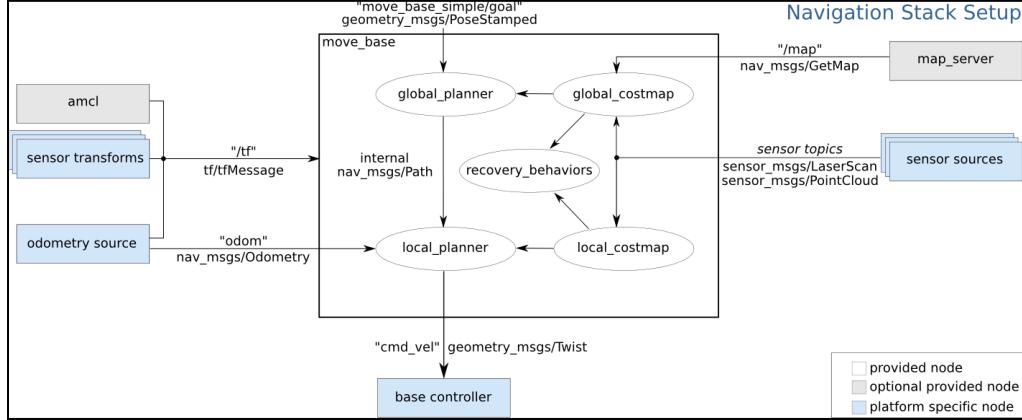


Fig. 1.1: Esquema de funcionamiento del paquete *move_base*.

En la Fig. 1.1, se observa el principal funcionamiento del paquete de *move_base*. Los nodos azules varían según la plataforma del robot, los nodos grises son opcionales pero están disponibles para todos los sistemas, y los nodos blancos son necesarios y también están disponibles para todos los sistemas. Centrándonos en los nodos blancos:

- **global_planner**: Crea una ruta desde la ubicación actual del robot hasta el objetivo final, considerando el mapa completo del entorno.
- **local_planner**: Se encarga de la navegación en tiempo real, ajustando la ruta planificada globalmente según las condiciones actuales del entorno.
- **global_costmap**: Proporciona una representación del entorno a gran escala, indicando las áreas navegables y los obstáculos.
- **local_costmap**: Proporciona una representación dinámica y detallada del entorno inmediato alrededor del robot.
- **recovery_behaviors**: Se utilizan estrategias para salir de situaciones difíciles o errores en la navegación. Existen dos métodos:
 - a. **conservative_reset**: Realiza un reseteo del costmap en el área cercana al robot, actualizando solo las zonas más cercanas.
 - b. **aggressive_reset**: Realiza un reseteo del costmap en un área más amplia alrededor del robot, abarcando una mayor porción del entorno.

La navegación del robot sigue el siguiente flujo:

1. **map_server** proporciona el mapa estático del entorno, utilizado por el **global_costmap**.
2. **sensor sources** suministran datos de los sensores, utilizados por el **local_costmap** para detectar obstáculos en tiempo real y actualizarse.
3. **sensor transforms (TF)** gestionan las transformaciones entre diferentes marcos de referencia, como entre la base del robot y los sensores. En este punto, también se gestionarán las TF implementadas en nuestros sensores.
4. **odometry source** proporciona la posición y velocidad del robot, esencial para el seguimiento preciso de la ruta planificada.
5. **global_costmap** utiliza el mapa del **map_server** para generar un mapa de costos global.
6. **local_costmap** usa los datos de los sensores para crear un mapa de costos local con información sobre obstáculos cercanos.
7. **global_planner** toma el **global_costmap** y planifica una ruta desde la posición actual del robot hasta el objetivo final.

8. **local_planner** utiliza el **local_costmap** para ajustar la ruta en tiempo real, evitando obstáculos inmediatos.
9. **recovery_behaviors** se activan si el robot encuentra un problema durante la navegación.
10. **base_controller** recibe los comandos de movimiento del **local_planner** y los ejecuta, controlando los motores del robot para moverlo según la ruta planificada.

La conversión de los mapas originales (Fig. 1.2) en escala de grises (Fig. 1.3) provoca las siguientes diferencias:

- Las áreas en gris no son ni completamente libres (valor de 255) o completamente ocupadas (valor de 0), sino que son áreas con probabilidad parcial de estar ocupadas, dependiendo de su valor.
- El parámetro mode: scale asegura que la imagen en escala de grises se interprete correctamente, y el sistema de navegación del robot utilizará esta información para planificar rutas y evitar obstáculos.
- El robot con esta configuración puede navegar por obstáculos que en el mapa aparecen pintados pero no esté detectando con la información de los sensores.

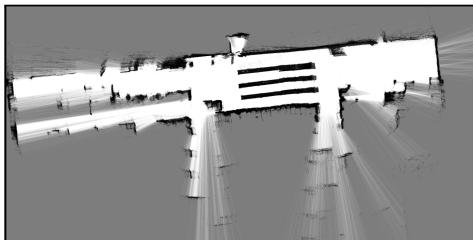


Fig. 1.2: Mapa generado directamente por el robot.

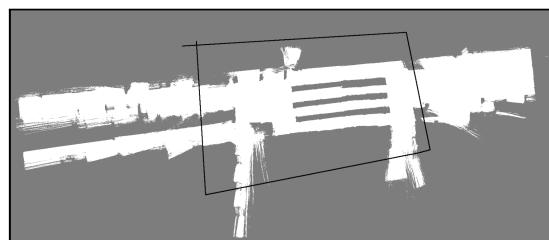


Fig. 1.3: Mapa editado en escala de grises con delimitadores en negro que actúan como obstáculos críticos. Las líneas alrededor del mapa en color negro actúan como obstáculos críticos.

Apéndice 2: Tablas de características de sensores

A partir de la información del apartado 2.2, se realiza una comparación de sensores más comunes en el campo de la robótica, obteniendo la Tabla 2.1 y Tabla 2.2.

Tabla 2.1: Comparativa de sensores respecto al funcionamiento, el alcance y el precio [3] [4] [5].

Tipo de sensor de proximidad	Funcionamiento	Alcance [cm]	Precio [€]
Infrarrojos	Luz infrarroja	5 - 100 (Corto alcance)	5 - 50
Ultrasónico	Ondas de sonido	2 - 500 (Largo alcance)	20 - 200
Inductivo	Electromagnético	0,1 - 1,5 (Corto alcance)	10 - 100
Capacitivo	Campo eléctrico	0,1 - 2 (Corto alcance)	15 - 150
Fotoeléctrico	Haces de luz	3 - 3000 (Largo alcance)	25 - 250
Láser	Haces de luz láser	0.1 - +10.000	100 - +25.000

Tabla 2.2: Características principales de los sensores inductivos, láser, magnéticos, ópticos y ultrasónicos [6].

Tipo de sensor				
Inductivo	Láser	Magnético	Óptico	Ultrasónico
Detección de objetos metálicos	Detección de objetos con alta precisión	Detección de objetos metálicos	Haces de luz para detección de objetos	Ondas de sonido para la detección de objetos
Rango amplio de detección	Rango muy amplio de detección	Rango medio de detección	Rango de detección limitado	Rango amplio de detección
Alta precisión	Alta precisión y tiempo de respuesta	Alta precisión y durabilidad	Alta precisión y tiempo de respuesta	Alta precisión y tiempo de respuesta
No se ve afectado por factores ambientales.	Sí se ve afectado por factores ambientales.	Sí se ve afectado por factores ambientales.	Sí se ve afectado por factores ambientales.	No se ve muy afectado por factores ambientales.
Utilizado en aplicaciones industriales.	Permite el modelado detallado del entorno en aplicaciones de mapeo y escaneo tridimensional	Detecta objetos magnéticos sin contacto	Detecta objetos con precisión	Detecta objetos difíciles de detectar con otros sensores

A partir del estudio que realizó *Wolfgang Ewald* [7], donde incluye modelos representativos sobre algunos tipos de sensores de proximidad, se ha extraído la Tabla 2.3.

Tabla 2.3: Tabla comparativa de sensores de proximidad basada en detección de movimiento, proximidad, luz ambiental, gestos y color.

	Tipo de sensor			
	IR LED	IR Laser	Ultrasonido	Fotodiodo
Sensor de movimiento	No	Sí	Sí	No
Sensor de proximidad	Sí	Sí	Sí	No
- medida absoluta (m/mm)	No	Sí	Sí	-
Luz ambiental	Sí	Sí	No	Sí
- IR	Sí	Sí	No	Sí
Gestos	No	No	No	No
Color	No	No	No	No

Apéndice 3: Comparativa de precios de sensores en el mercado

En este apéndice se muestra la Tabla 3.1 obtenida para extraer conclusiones de diferentes sensores en el mercado en base al precio. Todos ellos han sido diseñados para ser implementados en el ámbito profesional e industrial de la robótica:

Tabla 3.1: Comparativa de dos modelos de sensores láser 3D, láser 2D, ópticos, cámaras RGB-D, ultrasónicos e inductivos basado en su precio unitario [8] [9] [10] [11] [12] [13] [14] [15] [16]

Sensores	Precio unitario [€]	Sensores	Precio unitario [€]
Láser 3D: Ouster OS1	17.000	Cámara RGB-D: Microsoft Azure Kinect DK	400
Láser 3D: Robosense BPearl	4.000	Cámara RGB-D: Intel RealSense D455	400
Láser 2D: SICK LMS111	1.941	Sensor ultrasónico: MaxBotix HRLV-ShortRange	32
Láser 2D: LG10A65PUQ	1.878	Sensor ultrasónico: DFROBOT URM37	16
Sensor óptico: CM3-U3-13Y3C-CS	350	Sensor inductivo: Pepperl+Fuchs NBN4-12GM40-Z0	30
Sensor óptico: Keyence LR-ZB250	450	Sensor inductivo: Turck Bi5-M18-AP6X-H1141	110

Apéndice 4: Análisis del número de sensores

En este apéndice se amplía la información simulando diferentes escenarios de posición de sensores en el Robin-200.

1. Se muestra un boceto (Fig. 4.1) de la configuración: Un sensor lateral en el centro, en cada uno de los lados. En la Tabla 4.1 se recoge la información obtenida simulando diferentes casos de configuración.

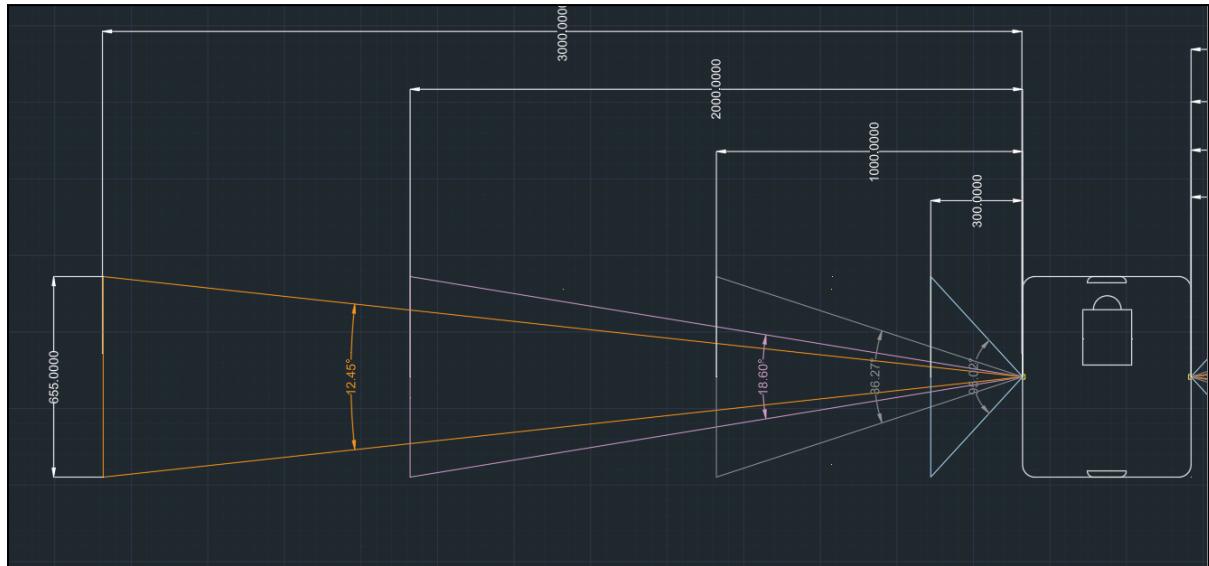


Fig. 4.1: Escenario con un sensor lateral y diferentes coberturas teniendo en cuenta distancia y ángulo de apertura

Tabla 4.1: Distancia y ángulo necesarias para cubrir todo el lateral del robot con un sensor

Distancia [mm]	300	1000	2000	3000
Apertura [°]	95.02	36.27	18.60	12.45

2. Se realiza el mismo estudio con la configuración: Dos sensores laterales situados a 115 mm de cada esquina de la base. (Fig 4.2 y Tabla 4.2).

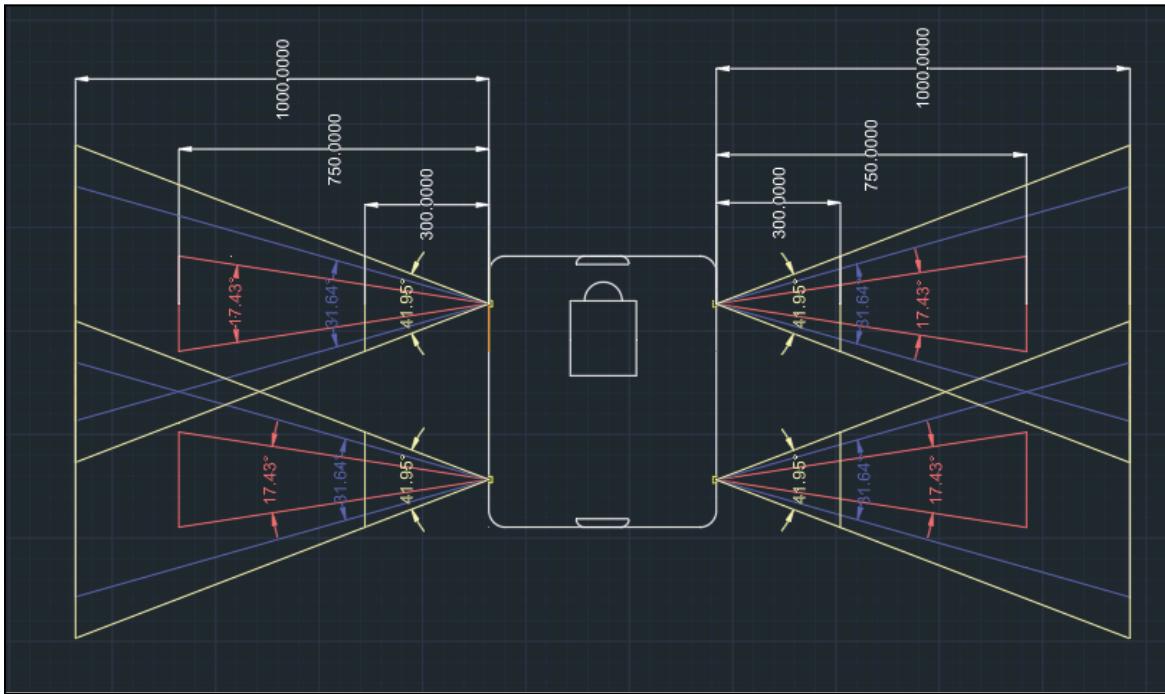


Fig. 4.2: Escenario con dos sensores laterales y diferentes coberturas teniendo en cuenta distancia y ángulo de apertura

Tabla 4.2: Distancia y ángulo necesarias para cubrir todo el lateral del robot con dos sensores

Distancia [mm]	300	750	1000
Apertura [°]	41.95	31.64	17.43

3. Se realiza el mismo estudio con la configuración: Un sensor situado en el centro del lateral (a 212.5 mm de cada sensor en la esquina) y dos sensores situados a 115 mm de cada esquina de la base (Fig.4.3 y Tabla 4.3).

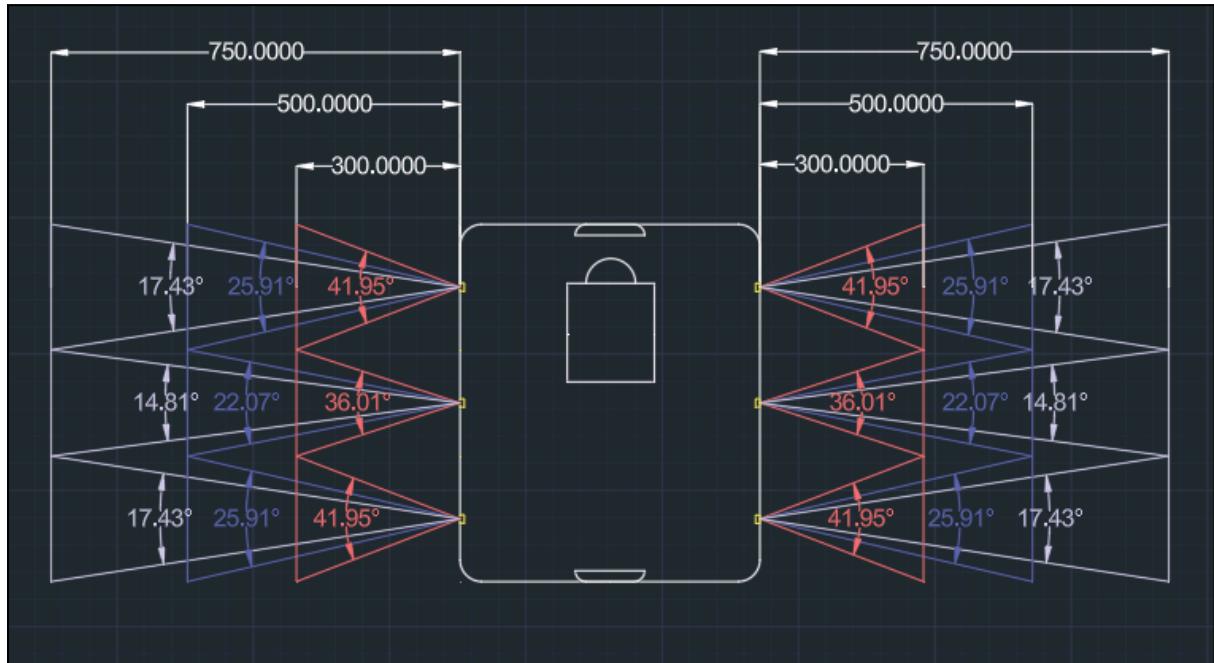


Fig. 4.3: Escenario con tres sensores laterales y diferentes coberturas teniendo en cuenta distancia y ángulo de apertura

Tabla 4.3: Distancia y ángulo necesarias para cubrir todo el lateral del robot con tres sensores

Distancia [mm]	300	500	750
Apertura [°]	41.95 / 36.01	25.91 / 22.07	17.43 / 14.81

Apéndice 5: Especificaciones de los modelos estudiados

En este apéndice se muestran las especificaciones más técnicas de los tres modelos principales estudiados y posteriormente sometidos a testing.

Apéndice 5.1: MaxBotix LV-MaxSonar EZ

Se muestra la tabla de pines (Fig. 5.1.1), necesaria para la posterior conexión de los sensores. Se muestran pruebas de apertura del sensor realizadas por MaxBotix (Fig. 5.1.2 y Fig. 5.1.3), información que da soporte a la elección del sensor. Se incluyen las dimensiones del sensor, importantes para la posterior integración de los sensores (Fig. 5.1.4).

Pin Out Description	
Pin 1-BW- *	Leave open or hold low for serial output on the TX output. When BW pin is held high the TX output sends a pulse (instead of serial data), suitable for low noise chaining.
Pin 2-PW-	This pin outputs a pulse width representation of range. The distance can be calculated using the scale factor of 147μS per inch.
Pin 3-AN-	Outputs analog voltage with a scaling factor of (Vcc/512) per inch. A supply of 5V yields ~9.8mV/in. and 3.3V yields ~6.4mV/in. The output is buffered and corresponds to the most recent range data.
Pin 4-RX-	This pin is internally pulled high. The LV-MaxSonar-EZ will continually measure range and output if RX data is left unconnected or held high. If held low the sensor will stop ranging. Bring high for 20μS or more to command a range reading.
Pin 5-TX-	When the *BW is open or held low, the TX output delivers asynchronous serial with an RS232 format, except voltages are 0-Vcc. The output is an ASCII capital ‘R’, followed by three ASCII character digits representing the range in inches up to a maximum of 255, followed by a carriage return (ASCII 13). The baud rate is 9600, 8 bits, no parity, with one stop bit. Although the voltage of 0-Vcc is outside the RS232 standard, most RS232 devices have sufficient margin to read 0-Vcc serial data. If standard voltage level RS232 is desired, invert, and connect an RS232 converter such as a MAX232. When BW pin is held high the TX output sends a single pulse, suitable for low noise chaining. (no serial data)
Pin 6+5V-	Vcc – Operates on 2.5V - 5.5V. Recommended current capability of 3mA for 5V, and 2mA for 3V. Please reference page 4 for minimum operating voltage versus temperature information.
Pin 7-GND-	Return for the DC power supply. GND (& Vcc) must be ripple and noise free for best operation.

Fig. 5.1.1: Guía de pines del MaxBotix LV-MaxSonar-EZ

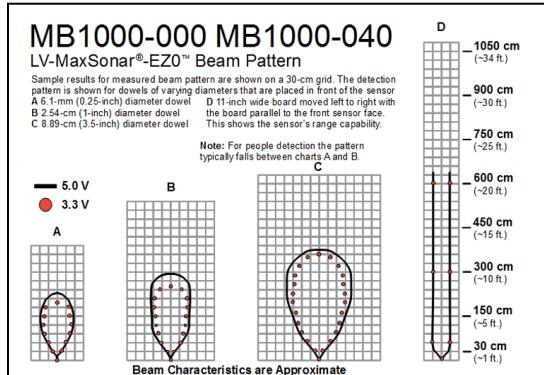


Fig 5.1.2: Esquema de rangos del modelo MaxBotix LV-MaxSonar EZ0.

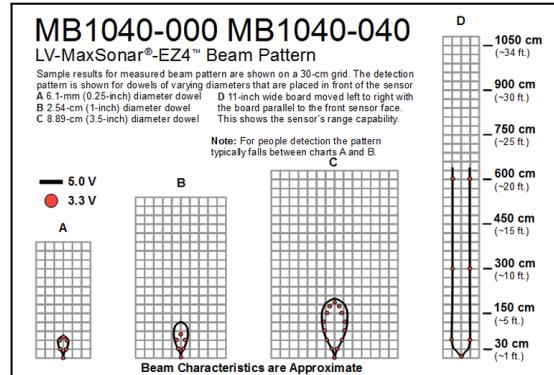


Fig 5.2.3: Esquema de rangos del modelo MaxBotix LV-MaxSonar EZ4.

LV-MaxSonar-EZ Mechanical Dimensions					
	A	0.785"	19.9 mm	H	0.100" 2.54 mm
	B	0.870"	22.1 mm	J	0.610" 15.5 mm
	C	0.100"	2.54 mm	K	0.645" 16.4 mm
	D	0.100"	2.54 mm	L	0.735" 18.7 mm
	E	0.670"	17.0 mm	M	0.065" 1.7 mm
	F	0.510"	12.8 mm	N	0.038" dia 1.0 mm dia
	G	0.124" dia	3.1 mm dia	weight, 4.3 grams	
	Part Number	MB1000	MB1010	MB1020	MB1030
	Paint Dot Color	Black	Brown	Red	Orange
	Part Number	MB1040			Yellow

Fig. 5.1.4: Tabla de dimensiones del MaxBotix LV-MaxSonar-EZ

Apéndice 5.2: MaxBotix HRLV-ShortRange EZ

Se muestra la tabla de pines (Fig. 5.2.1), necesaria para la posterior conexión de los sensores. Se muestran pruebas de apertura del sensor realizadas por MaxBotix (Fig. 5.2.2 y Fig. 5.2.3), información que da soporte a la elección del sensor. Se incluyen las dimensiones del sensor, importantes para la posterior integración de los sensores (Fig. 5.2.4). También se añade una tabla comparativa de gamas dentro del modelo HRLV-ShortRange (Fig. 5.2.5), importante para la selección de modelo final.

Pin Out Description

Pin 1- Temperature Sensor Connection: Leave this pin unconnected if an external temperature sensor is not used. For best accuracy, this pin is optionally connected to the HR-MaxTemp temperature sensor. Look up the HR-MaxTemp temperature sensor for additional information.

Pin 2- Pulse Width Output: This pin outputs a pulse width representation of the distance with a scale factor of 1uS per mm. Output range is 20uS for 20-mm to 5000uS for 5000-mm. Pulse width output is +/- 1% of the serial data sent.

Pin 3- Analog Voltage Output: On power-up the voltage on this pin, V_{observed} , is set to 0V. After which, the voltage on this pin has the voltage corresponding to the latest measured distance. This distance can be calculated with the following equation: $\text{distance} = [V_{\text{observed}} / ((V_{\text{cc}}/1024) * 6)] - 300$. (This output voltage is referenced to GND, Pin 7.) The analog voltage output is typically within ± 10 -mm of the serial output.

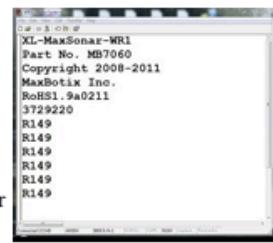
Using a 10bit analog to digital convertor, one can read the analog voltage bits (i.e. 0 to 1023) directly and multiply the number of bits in the value by 6 and subtract 300 to yield the range in mm. For example, 54 bits corresponds to 24-mm where $(54 * 6) - 300 = 24$ -mm.

Pin 4- Ranging Start/Stop: This pin is internally pulled high. If this pin is left unconnected or held high, the sensor will continually measure and output the range data. If held low, the HRLV-ShortRange-EZ will stop ranging. Bring high for 20uS or longer to command a range reading.

Real-time Range Data: When pin 4 is low and then brought high, the sensor will operate in real time and the first reading output will be the range measured from this first commanded range reading. When the sensor tracks that the RX pin is low after each range reading, and then the RX pin is brought high, unfiltered real time range information can be obtained as quickly as every 100mS.

Filtered Range Data: When pin 4 is left high, the sensor will continue to range every 100mS, but the output will pass through a 2Hz filter, where the sensor will output the range based on recent range information.

Pin 5-Serial Output: By default, the serial output is RS232 format (0 to Vcc) with a 1-mm resolution. The output is an ASCII capital "R", followed by four ASCII character digits representing the range in millimeters, followed by a carriage return (ASCII 13). The maximum distance reported is 5000. The serial output is the most accurate of the range outputs. Serial data sent is 9600 baud, with 8 data bits, no parity, and one stop bit. On power up the sensor will send serial data about the sensor. [View an example here](#).



V+ Pin 6 - Positive Power, Vcc: The sensor operates on voltages from 2.5V - 5.5V DC. For best operation, the sensor requires that the DC power be free from electrical noise. (For installations with known dirty electrical power, a 100uF capacitor placed at the sensor pins between V+ and GND will typically correct the electrical noise.) Please reference page 5 for minimum operating voltage verses temperature information.

GND Pin 7 – Sensor ground pin: DC return, and circuit common ground.

Fig. 5.2.1: Guía de pines del MaxBotix HRLV-ShortRange-EZ

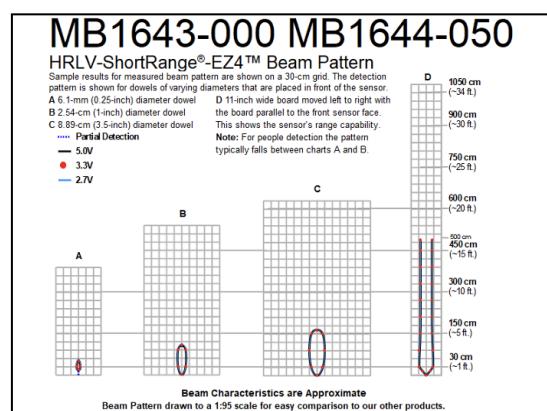
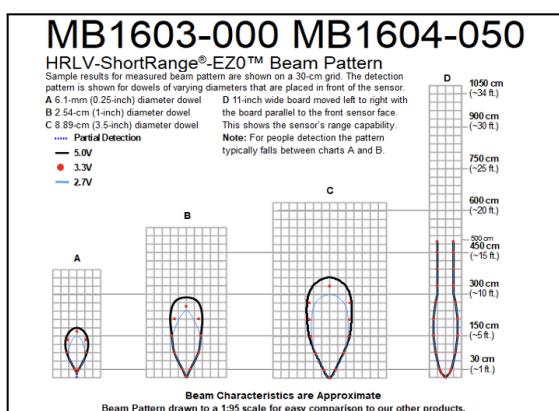


Fig 5.2.2: Esquema de rangos del modelo MaxBotix HRLV-ShortRange EZ0.

Fig 5.2.3: Esquema de rangos del modelo MaxBotix HRLV-ShortRange EZ4.

Mechanical Dimensions

The technical drawing illustrates the physical dimensions of the MaxBotix HRLV-ShortRange-EZ sensor module from three perspectives: top, side, and bottom. Key dimensions include:

- Top View (Left):** Dimensions A (height), C (width), D (depth), F, G, H, I, J, K.
- Side View (Middle):** Dimensions B (height), L (width), M (depth), N, O.
- Bottom View (Right):** Dimensions P (height), Q, R, S, T, U, V, W, X, Y, Z, ØK, ØL.

Table of Dimensions:

A	1.498"	38.05-mm	D	0.104"	2.64-mm	G	0.700"	17.78-mm	K	0.125 dia.	3.18-mm	K	0.125 dia.	3.18-mm	O	0.580"	13.97-mm
B	0.875"	22.23-mm	E	0.670"	17.02-mm	H	0.510"	12.95-mm	L	0.040" dia.	1.02-mm	L	0.040" dia.	1.02-mm	P	0.700"	17.78-mm
C	0.102"	2.59-mm	F	1.210"	30.73-mm	I	0.069"	1.75-mm	M	0.053"	1.35-mm	M	0.053"	1.35-mm	Q	0.343"	8.71-mm
Weight 8 Grams				J	0.169"	4.29-mm	N	0.625"	15.88-mm	N	0.625"	15.88-mm	R	0.427"	10.85-mm		

Fig. 5.2.4: Tabla de dimensiones del MaxBotix HRLV-ShortRange-EZ

Part Number	Serial Interface	People Detection	Wide Beam	High Sensitivity	Best Balance	Large Targets	Narrow Beam	Noise Tolerance	5 Meter Range
MB1603	RS232	Yes	Yes	Yes					Yes
MB1613	RS232	Yes	Yes	Yes	Yes				Yes
MB1623	RS232				Yes				Yes
MB1633	RS232				Yes	Yes	Yes	Yes	Yes
MB1643	RS232					Yes	Yes	Yes	Yes
MB1604	TTL	Yes	Yes	Yes					Yes
MB1614	TTL	Yes	Yes	Yes	Yes				Yes
MB1624	TTL				Yes				Yes
MB1634	TTL				Yes	Yes	Yes	Yes	Yes
MB1644	TTL					Yes	Yes	Yes	Yes

Fig. 5.2.5: Tabla comparativa de modelos de MaxBotix HRLV-ShortRange

Apéndice 5.3: DFROBOT URM37

Para este sensor se ha tenido en cuenta el esquema de pines para su conexión (Fig. 5.3.1) y pruebas de apertura del sensor (Tabla 5.3.1), información proporcionada por el fabricante.

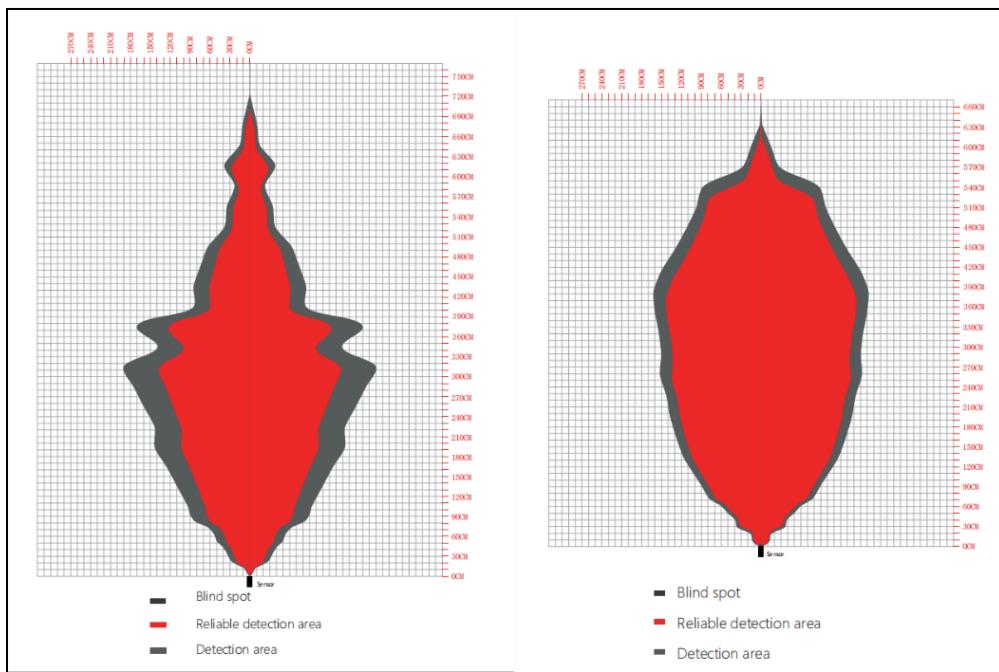


Fig. 5.3.1: Ángulo de haz del sensor de distancia ultrasónico SEN URM37

Tabla 5.3.1: Esquema de conexiones del SEN URM37

1	VCC	Voltage (3.3V - 5.5V)
2	GND	Ground
3	NRST	Reset
4	ECHO	Distancia medida presentada por la salida de datos 0-25000US por ancho de pulso PWM, representante de 1 CM/50US
5	SERVO	Pin de control del servo
6	COMP/TRIG	COMP: Modo encendido/apagado, cuando la distancia detectada es menor que un valor preestablecido, este pin baja. TRIG: entrada de disparo del modo PWM
7	DAC_OUT	Salida de voltaje analógico. El voltaje es proporcional a la distancia.
8	RXD	Pin de recepción de datos del módulo de comunicación asíncrona: nivel RS232/TTL
9	TXD	Pin de recepción de datos del módulo de comunicación asíncrona: nivel RS232/TTL

Apéndice 6: Formato de protocolo UART

La UART (Universal Asynchronous Receiver-Transmitter) es un dispositivo hardware esencial que facilita la comunicación en serie entre dispositivos electrónicos de manera asíncrona [17]. Esta tecnología es fundamental en sistemas embebidos y aplicaciones de comunicaciones, permitiendo la transmisión de datos bit a bit con un protocolo barato, sencillo y eficiente [18].

Se inicia la comunicación con un bit de inicio '0'. Este bit de inicio marca el comienzo de la transferencia de datos en serie, mientras que un bit de parada, que puede ser par o impar, finaliza la transacción. El bit de paridad par se representa con '0' (cuando hay un número par de '1') y el bit de paridad impar se representa con '1' (cuando hay un número impar de '1').

La transmisión de datos (Fig. 6.1) se realiza a través de una única línea de transmisión (TxD). En este contexto, '0' se considera espacio y '1' se conoce como estado de marca.

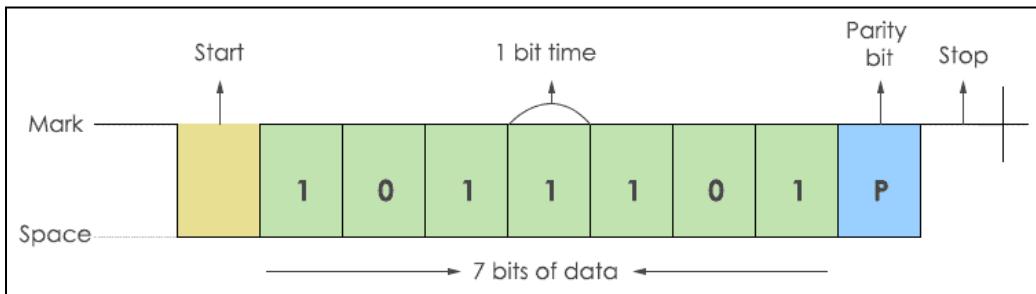


Fig. 6.1: Esquema de transmisión de datos (1 byte) en protocolo UART

Para la recepción de datos (Fig. 6.2), se utiliza la línea de recepción (RxD). Los datos se reciben bit a bit, siguiendo el mismo formato y sincronización establecidos para la transmisión.

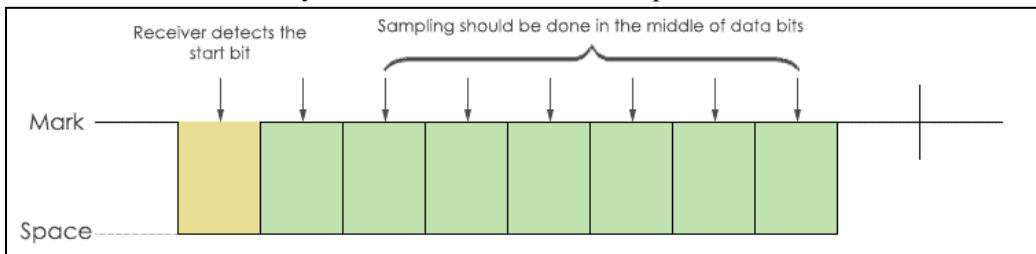


Fig. 6.2: Esquema de recepción de datos (1 byte) en protocolo UART

Apéndice 7: Testeo de sensores

En este apéndice se profundiza sobre las pruebas iniciales realizadas con los modelos MaxBotix. Los escenarios propuestos se reproducen para ambos sensores.

- En los escenarios 1 a 4 (Fig. 7.1 - Fig. 7.4) se miden diferentes distancias con el sensor a 0 cm del suelo.
- En los escenarios 5 a 7 (Fig. 7.5 - Fig. 7.10) se busca la distancia en la que el sensor detecta el suelo a partir de diferentes alturas.

En cada subapartado se muestran los resultados extraídos.



Fig 7.1: Caso 1: Distancia 135 mm

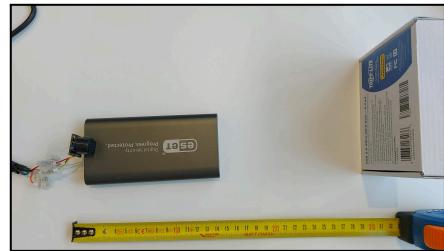


Fig 7.2: Caso 2: Distancia 295 mm



Fig 7.3: Caso 3: Distancia 1040 mm



Fig 7.4: Caso 4: Sin obstáculos delante



Fig. 7.5: Caso 5: Configuración del escenario con el sensor a 100 mm del suelo.



Fig 7.6: Caso 5: Objeto a 750 mm del sensor.



Fig 7.7: Caso 6: Configuración del escenario con el sensor a 200 mm del suelo.



Fig 7.8: Caso 6: Objeto a 1000 mm del sensor.



Fig. 7.9: Caso 7: Configuración del escenario con el sensor a 500 mm del suelo.

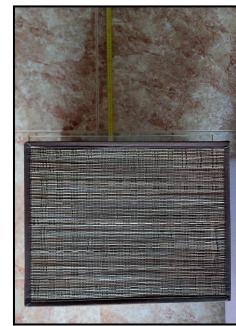


Fig 7.10: Caso 7: Objeto a 180 mm del sensor.

Apéndice 7.1: Testeo del MaxBotix LV-MaxSonar EZ0

A partir de los escenarios explicados en el inicio del Apéndice 7, se extraen la Tablas 7.1.1 y Tabla 7.1.2, donde se estudian los errores de las distancias reales y detectadas del MaxBotix LV-MaxSonar EZ0.

Tabla 7.1.1: Comparativa de medidas reales y detectadas. Estudio por casos

Casos	Distancia del objeto (d) [mm]	Distancia detectada (d_0) [mm]	Error absoluto [mm] $EA = d - d_0 $	Error relativo $ER = \frac{ d - d_0 }{d_0}$	Error relativo (%) $er\% = ER \cdot 100$
Caso 1	135	300	165	1.2	120
Caso 2	295	305	10	0.0338	3.38
Caso 3	1040	1025	15	0.0144	1.44
Caso 4	6400	2557 (media)	3843	0.6004	60.04

Tabla 7.1.2: Comparativa de la detección del suelo por el sensor. Estudio por distancias

Distancias del sensor respecto al suelo	Distancia de la detección de suelo
100 mm	412 mm
200 mm	863 mm
500 mm	1307 mm (detección no fiable)

Apéndice 7.2: Testeo del MaxBotix HRLV-ShortRange EZ0 / EZ1

A partir de los escenarios explicados en el inicio del Apéndice 7, se extraen la Tablas 7.2.1 y Tabla 7.2.2, donde se estudian los errores entre las distancias reales y detectadas del MaxBotix HRLV-ShortRange EZ0.

Tabla 7.2.1: Comparativa de medidas reales y detectadas. Estudio por casos

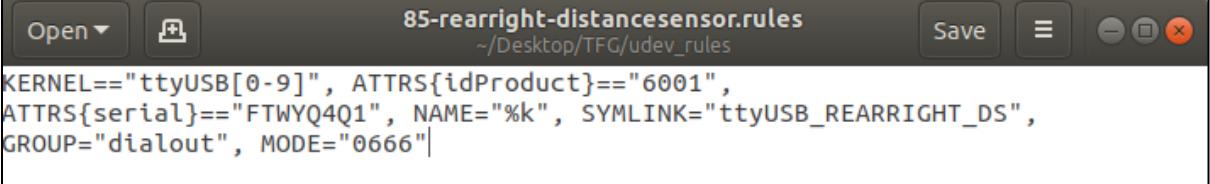
Casos	Distancia del objeto (d) [mm]	Distancia detectada (d_0) [mm]	Error absoluto [mm] $EA = d - d_0 $	Error relativo $ER = \frac{ d-d_0 }{d_0}$	Error relativo (%) $er\% = ER \cdot 100$
Caso 1	135	136	1	0.0074	0.74
Caso 2	295	297	2	0.0067	0.67
Caso 3	1040	1043	3	0.0028	0.28
Caso 4	5000	4395.5 (media)	604.5	0.1209	12.09

Tabla 7.2.2: Comparativa de la detección del suelo por el sensor. Estudio por distancias

Distancias del sensor respecto al suelo	Distancia de la detección de suelo
100 mm	379 mm
200 mm	692 mm
500 mm	1327 mm

Apéndice 8: Archivos .rules de udev

En este apéndice se amplia información sobre el proceso de creación de los cuatro archivos .rules, uno para cada sensor. Se sigue el modelo de la Fig. 8.1 para todos los archivos.



```
85-rearright-distancesensor.rules
~/Desktop/TFG/udev_rules
KERNEL=="ttyUSB[0-9]", ATTRS{idProduct}=="6001",
ATTRS{serial}=="FTWYQ4Q1", NAME="%k", SYMLINK="ttyUSB_REARRIGHT_DS",
GROUP="dialout", MODE="0666"
```

Fig. 8.1: Archivo de reglas udev del sensor derecho trasero.

Cada udev rule, contiene:

- **KERNEL=="ttyUSB[0-9]"**: Esta condición se cumple para dispositivos cuyo nombre en el kernel coincide con **ttyUSB** seguido de un dígito.
- **ATTRS{idProduct}=="6001"**: Se especifica el identificador de producto USB.
- **ATTRS{serial}=="XXXXXX"**: Se especifica el número de serie del dispositivo es **XXXXXX**.
- **NAME="%k"**: Especifica que el nombre del dispositivo debe ser el mismo que el nombre del kernel (**%k** es un comodín que se sustituye por el nombre del dispositivo en el kernel).
- **SYMLINK="ttyUSB_POSITION_DS"**: Crea un enlace simbólico con el nombre especificado que apunta al dispositivo que cumple con las condiciones anteriores. Estos nombres corresponden a **ttyUSB_REARRIGHT_DS**, **ttyUSB_REARLEFT_DS**, **ttyUSB_FRONTRIGHT_DS** y **ttyUSB_FRONTLEFT_DS**.
- **GROUP="dialout"**: Asigna el dispositivo al grupo **dialout**.
- **MODE="0666"**: Establece los permisos del dispositivo a **0666**, lo que significa que cualquier usuario puede leer y escribir en el dispositivo.

Apéndice 9: Modificaciones del Xacro del Robin-200

En este apéndice se muestra cómo se han implementado los archivos de descripción de los sensores en el archivo Xacro del Robin-200 [19].

- En la Fig. 9.1 se muestra cómo se importan los cuatro archivos ubicando su directorio.
- En la Fig 9.2 se muestra cómo se inicializa el nombre del frame del sensor, del frame el cual proviene y los valores de origen y roll, pitch y yaw que serán diferentes para cada sensor, dependiendo de cómo esté orientado el sensor en el robot.

```
<!-- Import distance sensor-->
<xacro:include filename="$(find robin_bringup)/description/distance_sensors/
ultrasonic_sensor_rear_right.urdf.xacro" />
<xacro:include filename="$(find robin_bringup)/description/distance_sensors/
ultrasonic_sensor_rear_left.urdf.xacro" />
<xacro:include filename="$(find robin_bringup)/description/distance_sensors/
ultrasonic_sensor_front_right.urdf.xacro" />
<xacro:include filename="$(find robin_bringup)/description/distance_sensors/
ultrasonic_sensor_front_left.urdf.xacro" />
```

Fig. 9.1: Importación de los cuatro archivos Xacro de descripción de los sensores de ultrasonido.

```
<xacro:ultrasonic_sensor_front_right prefix="$(arg prefix)_ultrasonic_sensor_front_right"
parent="$(arg prefix)base_link">
  <origin xyz="0.235 -0.2465 0.1194" rpy="0 0 4.7123"/>
</xacro:ultrasonic_sensor_front_right>
```

Fig 9.2: Implementación del frame del sensor delantero derecho en el archivo Xacro del Robin-200. Se sigue el mismo procedimiento para el resto.

Al realizar la implementación se incluyen en el modelo 3D del robot (Fig. 9.3 y Fig. 9.4).

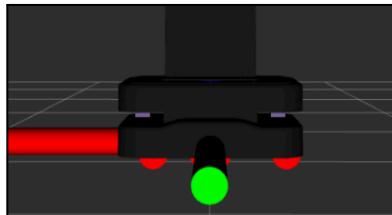


Fig 9.3: Visualización del lateral izquierdo del modelo del Robin-200 con los frames de los sensores.

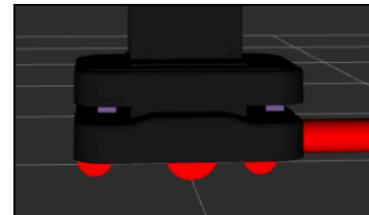


Fig. 9.4: Visualización del lateral derecho del modelo del Robin-200 con los frames de los sensores.

Apéndice 10: Algoritmo de Kalman Filter

En este apéndice se profundiza sobre el algoritmo Kalman Filter (Fig 10.1) y su funcionamiento [20] [21].

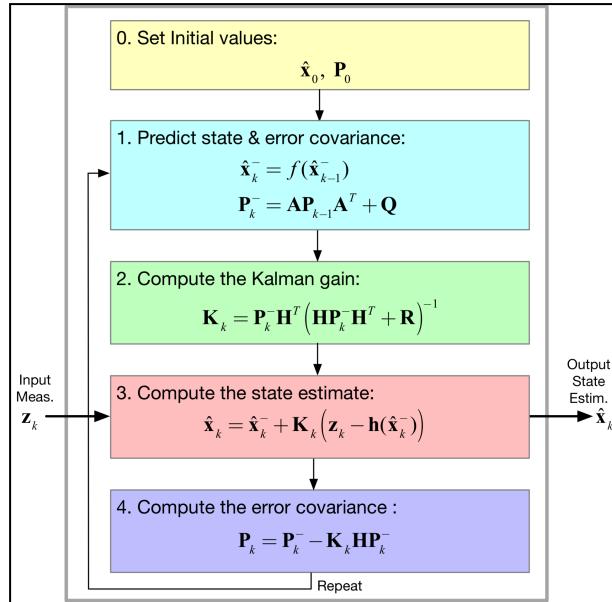


Fig 10.1: Workflow del Kalman Filter

0. Establecer Valores Iniciales

- $\hat{x}_0 \rightarrow$ Estimación inicial del estado.
- $P_0 \rightarrow$ Covarianza inicial del error.

1. Predecir el estado y la covarianza del error

- Estado Predicho $\hat{x}_{\bar{k}} = f(\hat{x}_{\bar{k}-1}) \rightarrow$ Se predice el estado actual basado en el estado previo.
- Covarianza del Error Predicha $P_{\bar{k}} = AP_{k-1}A^T + Q \rightarrow$ Se predice la covarianza del error utilizando la matriz de transición A a la covarianza del error previa, y la covarianza del ruido del proceso Q

2. Calcular la Ganancia de Kalman

- Kalman Gain $K_k = P_{\bar{k}} H^T (H P_{\bar{k}} H^T + R)^{-1} \rightarrow$ Se obtiene el Kalman Gain utilizando la covarianza del error predicha, la matriz de observación H , y la covarianza del ruido de medición R . Esta ganancia determina cómo se deben ajustar las predicciones basadas en las nuevas mediciones.

3. Calcular la estimación del estado

- Estado Estimado $\hat{x}_k = \hat{x}_{\bar{k}} + K_k (z_k - h(\hat{x}_{\bar{k}})) \rightarrow$ La estimación del estado se actualiza ajustando el estado predicho con la ganancia de Kalman y la diferencia entre la medición actual z_k y la medición predicha $h(\hat{x}_{\bar{k}})$.

4. Calcular la Covarianza del Error

- Covarianza del Error Actualizada $P_k = (I - K_k H) P_{\bar{k}} \rightarrow$ Se actualiza utilizando el Kalman Gain y la matriz de observación.

Apéndice 11: Planos 2D de los modelos 3D finales

En este apéndice se proporcionan los planos en 2D obtenidos con Autodesk Fusion de los modelos finales de la carcasa y las tapas [58] [59] [60] [61] [62] [63]. Se pueden encontrar a gran escala [en esta carpeta del repositorio](#).

- Carcasa MaxBotix HRLV-ShortRange EZ1 (MB1613)

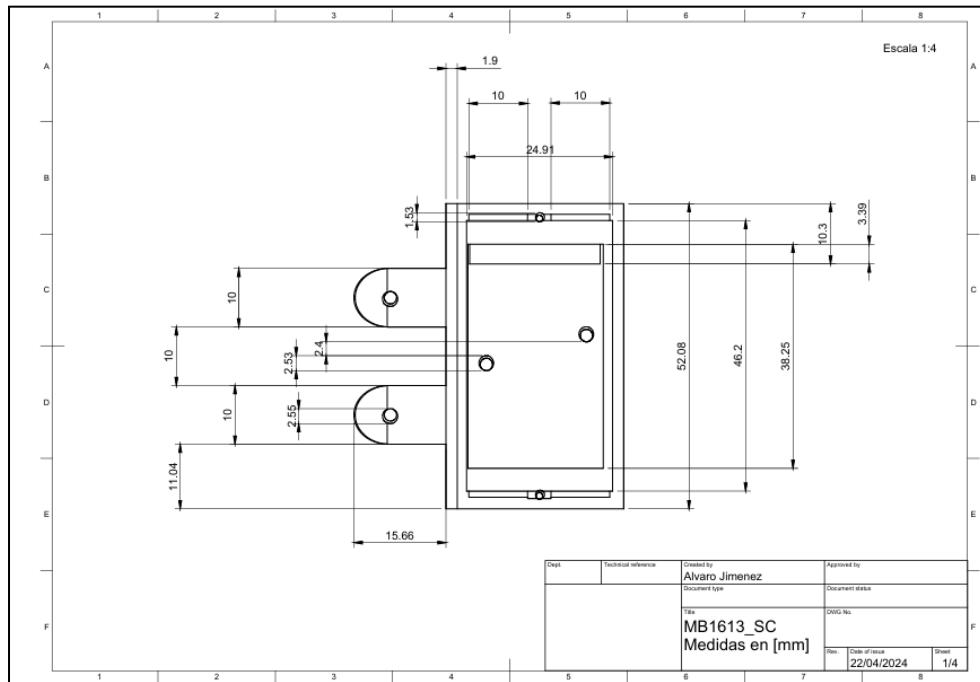


Fig 10.1: Plano 2D de la carcasa MB1613. Vista superior

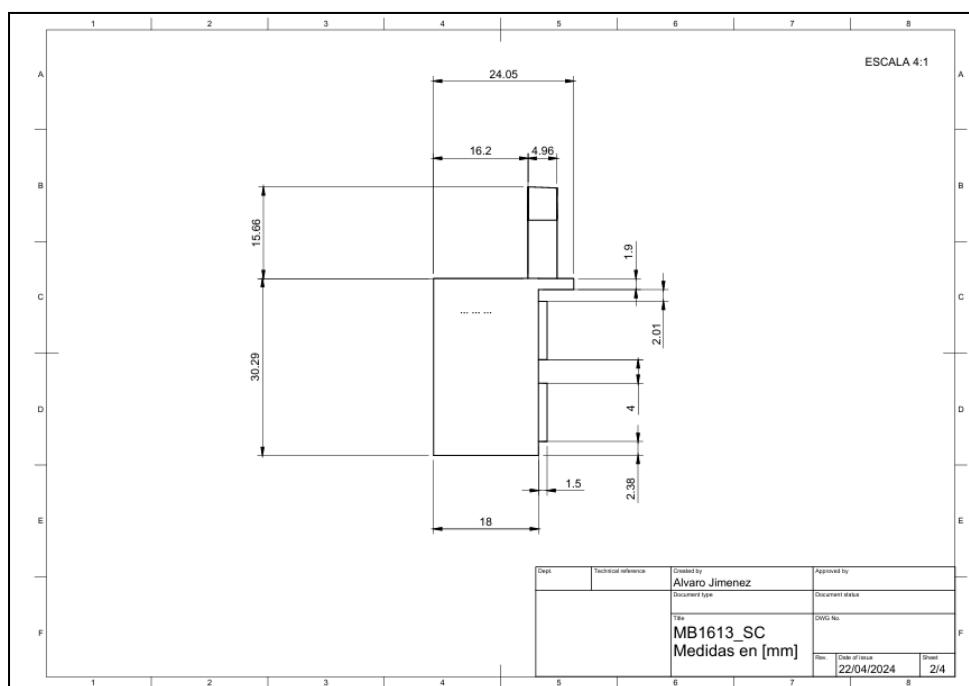


Fig 10.2: Plano 2D de la carcasa MB1613. Vista lateral sin apertura

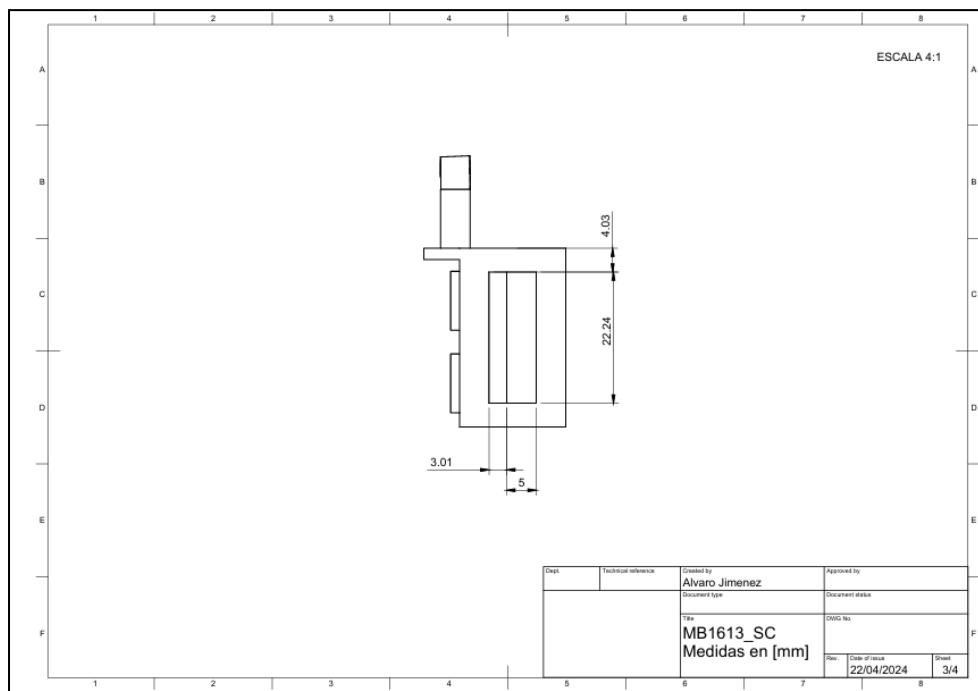


Fig 10.3: Plano 2D de la carcasa MB1613. Vista lateral con apertura

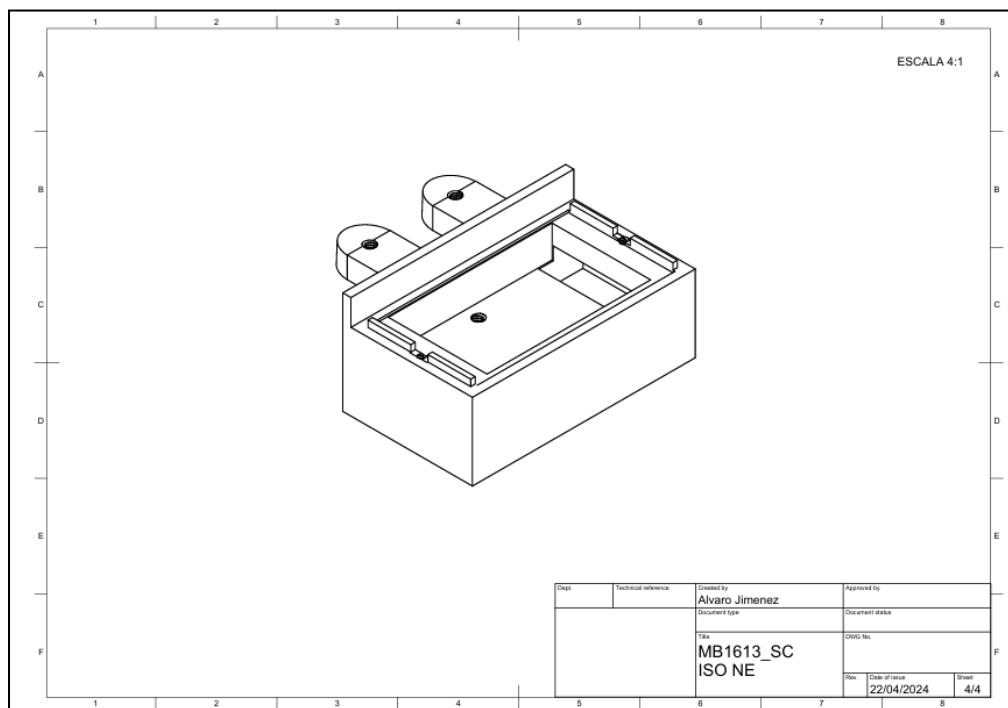


Fig 10.4: Plano 2D de la carcasa MB1613. ISO NO

- Tapa de la carcasa sin reducción de ángulo

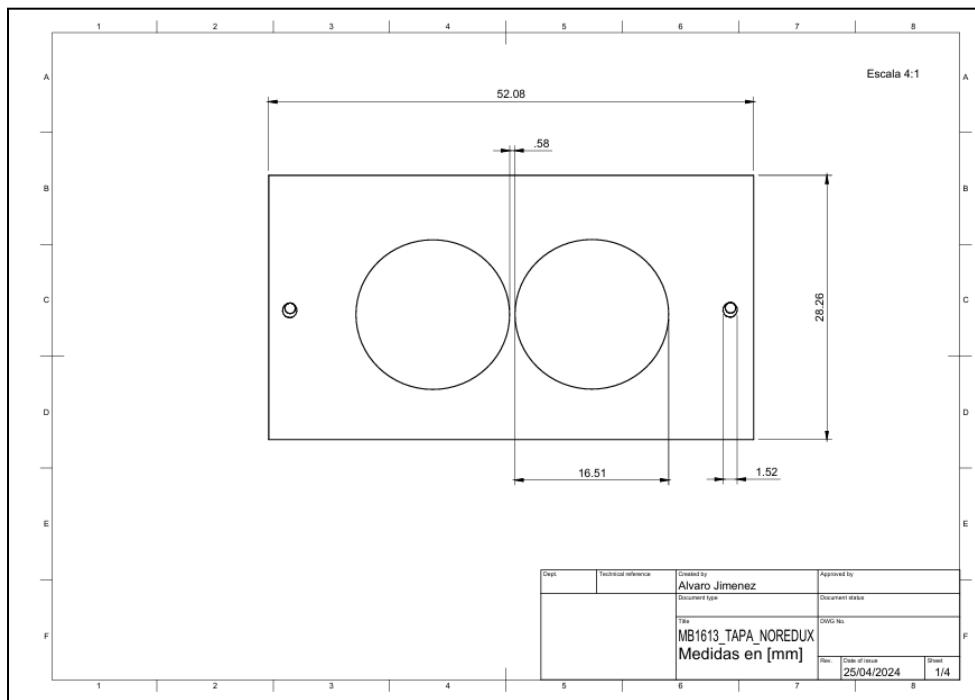


Fig 10.5: Plano 2D de la tapa sin reducción. Vista superior

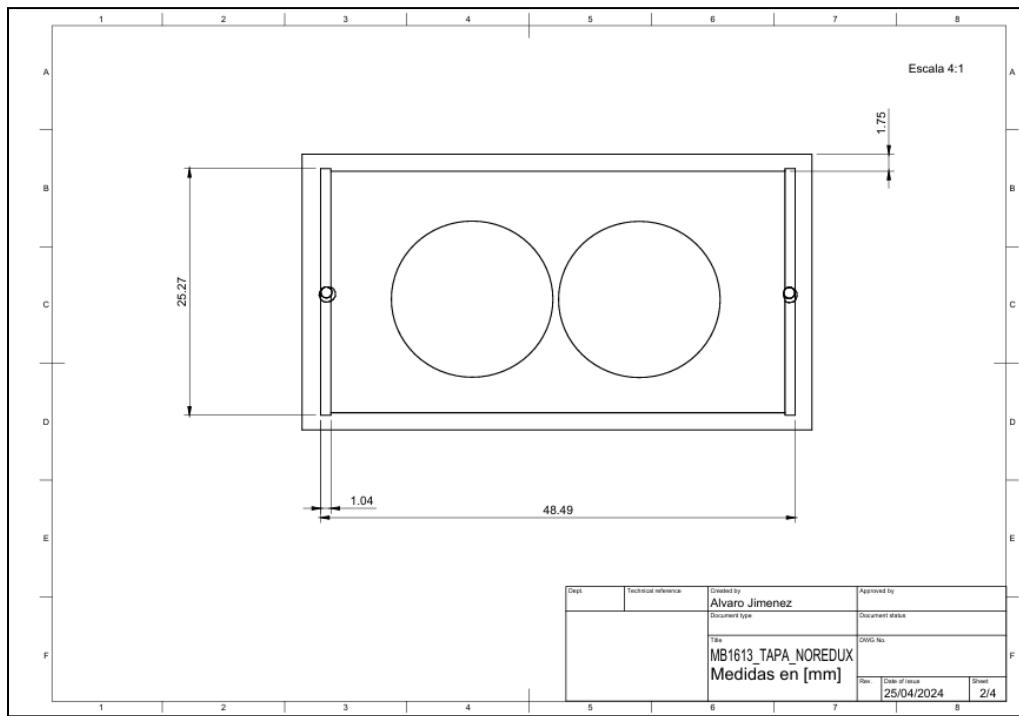


Fig 10.6: Plano 2D de la tapa sin reducción. Vista inferior

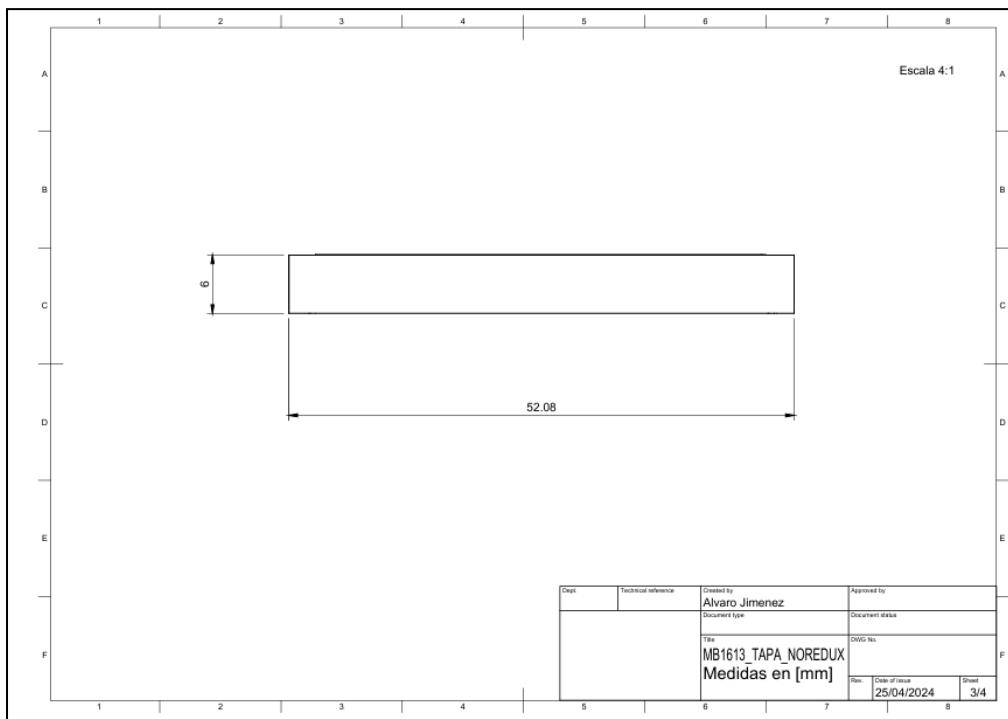


Fig 10.7: Plano 2D de la tapa sin reducción. Vista lateral

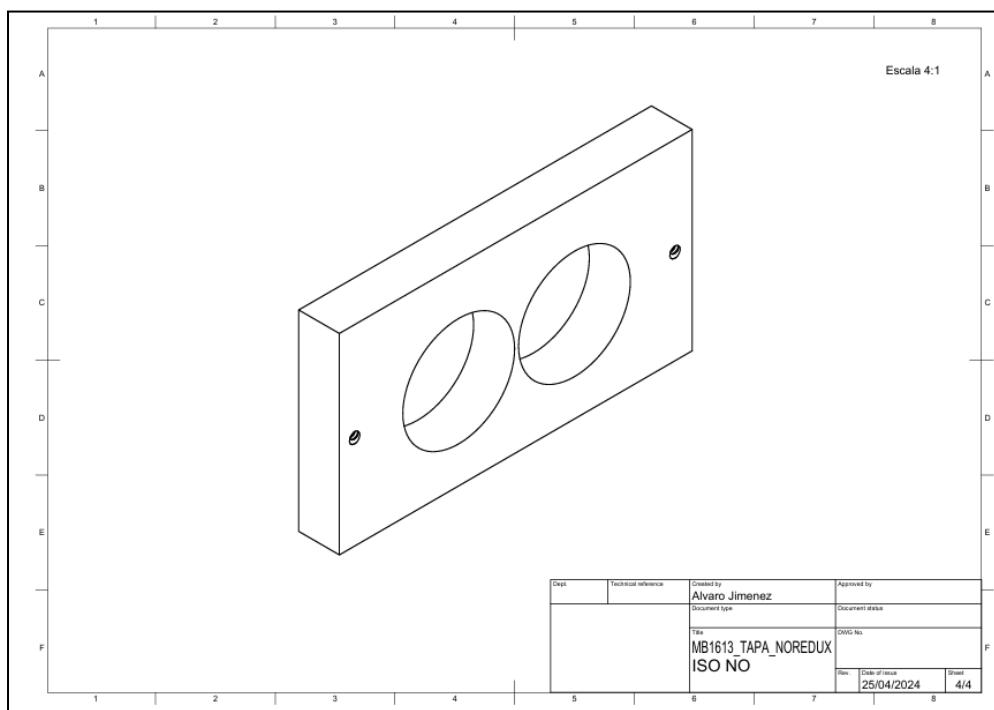


Fig 10.8: Plano 2D de la tapa sin reducción. ISO NO

- Tapa de la carcasa con reducción de ángulo

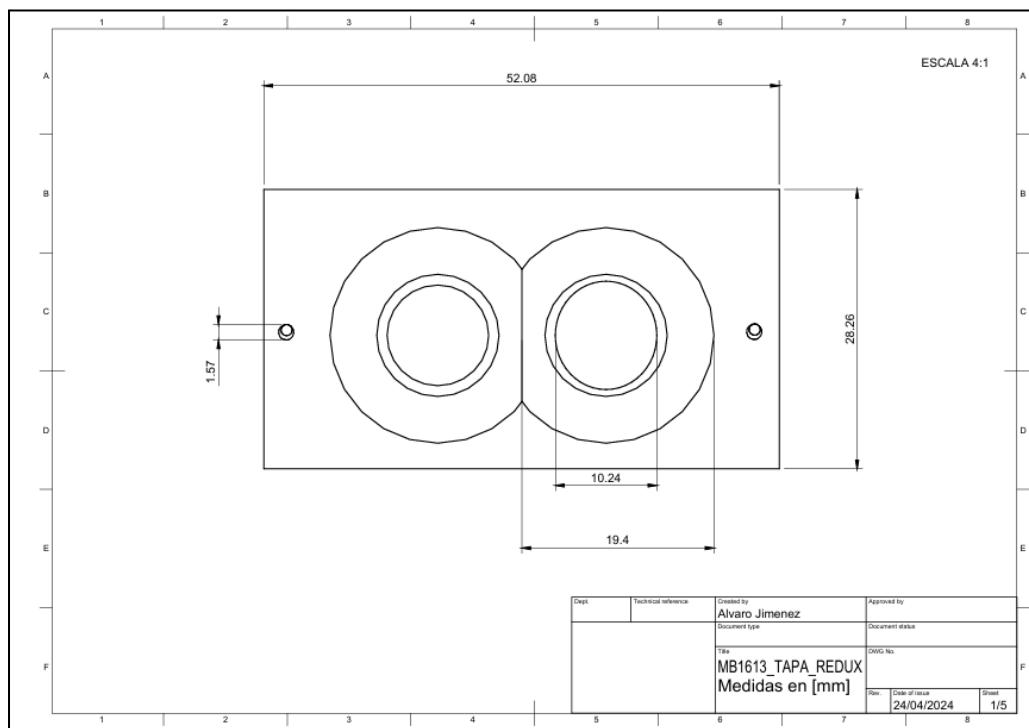


Fig 10.9: Plano 2D de la tapa con reducción. Vista superior

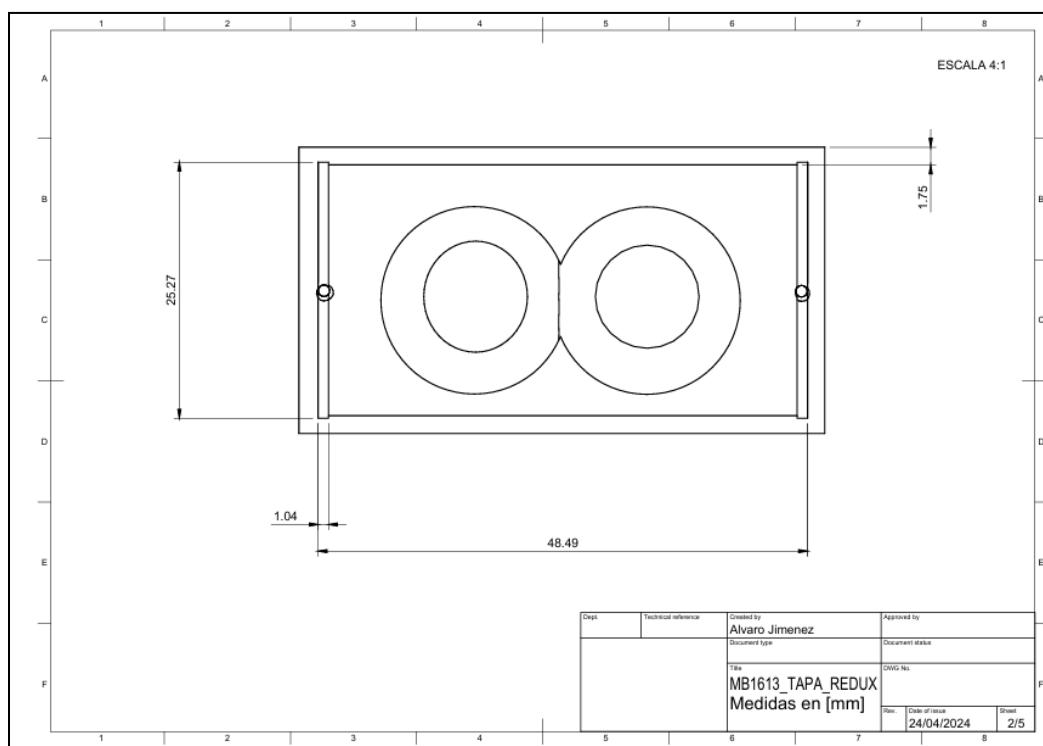


Fig 10.10: Plano 2D de la tapa con reducción. Vista inferior

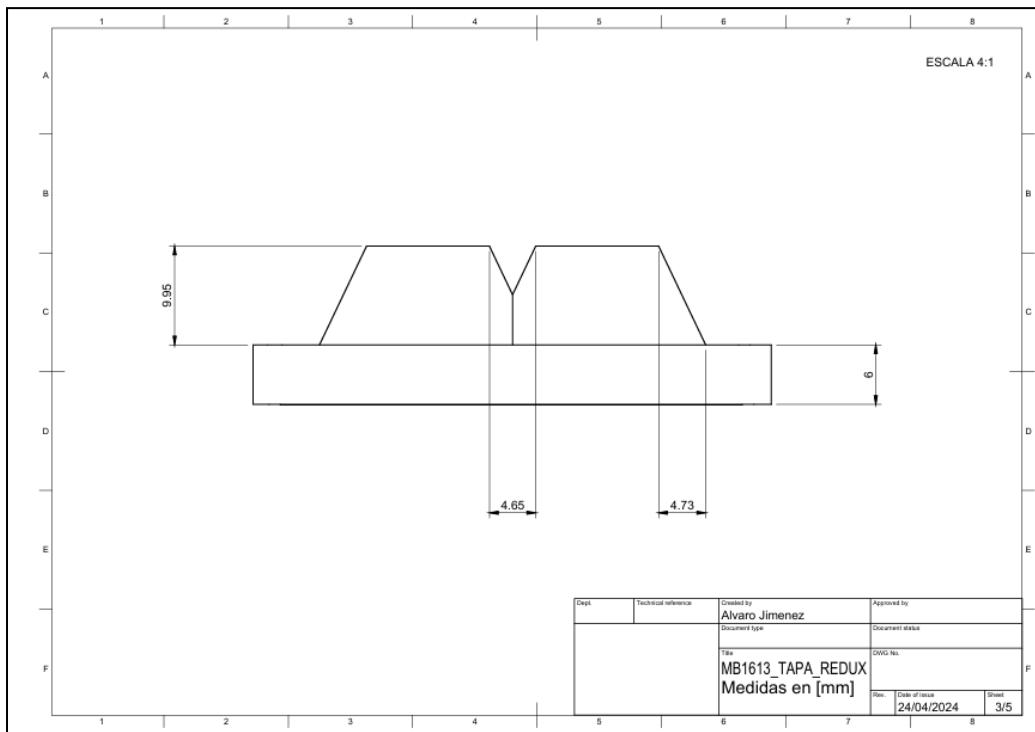


Fig 10.11: Plano 2D de la tapa con reducción. Vista lateral 1

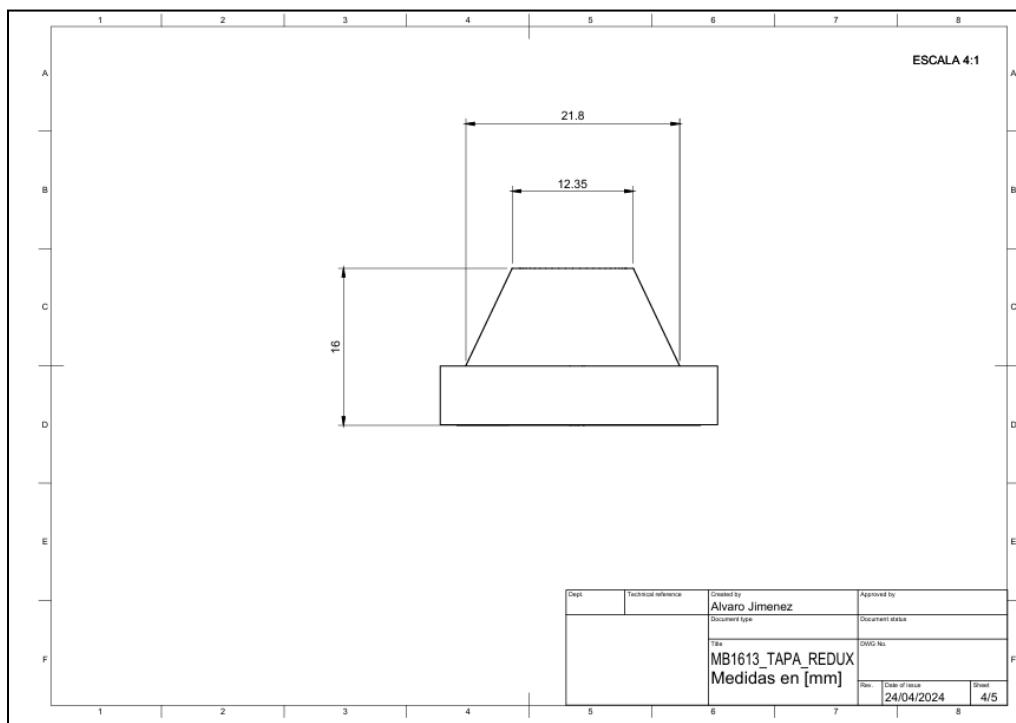


Fig 10.12: Plano 2D de la tapa con reducción. Vista lateral 2

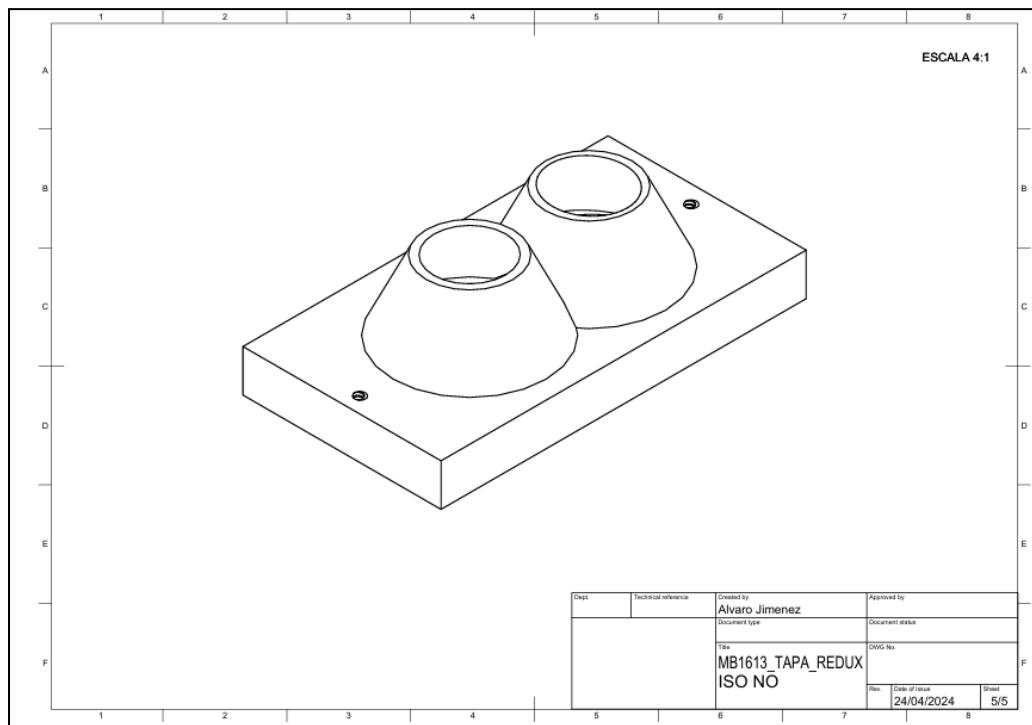


Fig 10.13: Plano 2D de la tapa con reducción. ISO NO

Bibliografía

- [1] *navigation/Tutorials - ROS Wiki*. (s. f.). <http://wiki.ros.org/navigation/Tutorials>
- [2] *move_base - ROS Wiki*. (s. f.). http://wiki.ros.org/move_base
- [3] Joe. (2023, 1 diciembre). *Aplicaciones de sensores de proximidad: usos esenciales revelados OMCH*. OMCH. <https://www.omchsmmps.com/es/proximity-sensor-applications-explained/>
- [4] Inductive & Capacitive Proximity Sensor Working - Your Electrical Guide. (2023, 3 septiembre). *Your Electrical Guide*. <https://www.yourelectricalguide.com/2017/12/proximity-sensor-working.html>
- [5] Sme, T. C. (2024, 10 mayo). Mastering the Range of IR Proximity Sensors: A Comprehensive Guide. *Techies Science*. <https://techiescience.com/es/ir-proximity-sensor-range/>
- [6] *Inductive Sensor Explained | Different Types and Applications - RealPars*. (s. f.). <https://www.realpars.com/blog/inductive-sensor>
- [7] Ewald, W. (2020, 2 diciembre). *Sensor comparison – ambient light, proximity, motion*. Wolles Elektronikkiste. <https://wolles-elektronikkiste.de/en/sensor-comparison-ambient-light-proximity-motion>
- [8] *RS-Bpearl-RoboSense - RoboSense | World Leader in LiDAR and Perception Solutions*. (s. f.). <https://www.robosense.ai/en/rslidar/RS-Bpearl>
- [9] RobotShop Europe. (s. f.). *Ouster OS1 64 Rev 7 Mid Range LiDAR Sensor*. <https://eu.robotshop.com/products/ouster-os1-64-rev-6-mid-range-lidar-sensor#:~:text=Price,%3A%20%E2%82%AC17.424%2C00>
- [10] dfrobot.com. (s. f.-b). *Fermion: URM37 Ultrasonic Distance Sensor Breakout (2~800cm, RS232 / UART)*. <https://www.dfrobot.com/product-53.html>
- [11] Camara 82635DSD455MP. (s. f.). Digikey. <https://www.digikey.es/es/products/detail/intel-realsense/82635DSD455MP/12429609>
- [12] Laser 2D. (s. f.). Digikey LG10A65PUQ. <https://www.digikey.es/es/products/detail/banner-engineering-corporation/LG10A65PUQ/12503974>
- [13] Laser 2D OD1-B035C15I25. (s. f.). Digikey. <https://www.digikey.es/es/products/detail/sick-inc/OD1-B035C15I25/8130714>
- [14] Sensor óptico CM3-U3-13Y3C-CS. (s. f.). Digikey. <https://www.digikey.es/es/products/detail/flir-integrated-imaging-solutions-inc/CM3-U3-13Y3C-CS/16528278>
- [15] Cámara RGB-D Azure Kinect. (s. f.). Microsoft. <https://www.microsoft.com/en-us/d/azure-kinect-dk/8pp5vxmd9nhq?activetab=pivot:overviewtab>
- [16] *What Filter to use for Ultrasonic Sensor? [X post from /r/Arduino]*. (s. f.). Reddit. https://www.reddit.com/r/robotics/comments/1fjt8b/what_filter_to_use_for_ultrasonic_sensor_x_post/
- [17] SwellFox. (2024, 30 mayo). *Communicating with a PC using UART - Part 11 Microcontroller Basics (PIC10F200)*. CircuitBread. <https://www.circuitbread.com/tutorials/communicating-with-a-pc-using-uart---part-11-microcontroller-basics-pic10f200>

- [18] *What is UART Protocol? UART Communication Explained.* (s. f.). Arrow. <https://www.arrow.com/en/research-and-events/articles/what-is-uart-protocol-uart-communication-explained>
- [19] Automaticaddison, A. (2021, 12 diciembre). *URDF vs. SDF – Link Pose, Joint Pose, Visual & Collision* — *Automatic Addison.* <https://automaticaddison.com/urdf-vs-sdf-link-pose-joint-pose-visual-collision/>
- [20] *Kalman Filter Variables* — *gps-helper 1.1.4 documentation.* (s. f.). https://gps-helper.readthedocs.io/en/latest/nb_examples/Kalman_GPS_practice.html
- [21] Franklin, W. (2024, 2 febrero). *Kalman filter Explained simply - the Kalman filter.* The Kalman Filter. <https://thekalmanfilter.com/kalman-filter-explained-simply/>
- [22] CADistas. (2021, 19 abril). *GENERACIÓN DE PLANOS a PARTIR DE UNA PIEZA 3D EN AUTOCAD TUTORIAL* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=4yyDXNmE6E0>
- [23] dorcuCom - Maker for Makers. (2022, 27 junio).  *Cómo EDITAR ARCHIVOS STL con Fusion 3D de forma FÁCIL y SENCILLA* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=Gy91I4SezXY>
- [24] *Extract a 2D section from 3D part.* (2016, 26 febrero). Autodesk Community. <https://forums.autodesk.com/t5/fusion-design-validate-document/extract-a-2d-section-from-3d-part/tid-p/6056295>
- [25] CAD CAM TUTORIAL BY MAHTABALAM. (2019, 5 septiembre). *How to make 2D drawing in Autodesk Fusion 360* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=e42kK2VxXa8>
- [26] The Mechanical Man. (2021, 19 julio). *Convert STL mesh to a Solid Body In Just 3 Minute | Fusion 360 #fusion360* [Vídeo]. YouTube. https://www.youtube.com/watch?v=_9N-oKdTeLk
- [27] Micrographics. (2020, 23 enero). *Create a 2D drawing in Fusion 360* [Vídeo]. YouTube. https://www.youtube.com/watch?v=69J_PN3vaYk