Proyecto de Fin de Grado (Ingeniería de Computadores, Tecnologías de la Sociedad de la Información)

Diseño y Construcción de Robocar

KENTO REINOSO HAYASHI
RUBÉN HIGUERA CASTILLO
2023/2024

Contenido

Introducción

En la actualidad, la robótica y la inteligencia artificial están revolucionando la manera en que interactuamos con el mundo que nos rodea. Una de las aplicaciones más prometedoras y desafiantes de estas tecnologías es el desarrollo de vehículos autónomos, capaces de operar sin intervención humana, mejorando la seguridad y eficiencia en el transporte. Este proyecto de fin de grado se centra en el diseño y construcción de un robot coche autónomo, utilizando componentes comerciales de coste contenido, con el objetivo de alcanzar la máxima autonomía posible.

El documento que se presenta a continuación detalla el proceso de desarrollo del proyecto, desde la concepción inicial hasta la implementación final. Se abordan temas como la selección de sensores adecuados, la arquitectura funcional y tecnológica del sistema, y la metodología de trabajo adoptada para llevar a cabo el desarrollo de manera incremental y controlada.

Se hace énfasis en la importancia de una planificación detallada y una metodología de trabajo estructurada para enfrentar los retos que implica la construcción de un sistema tan complejo. Además, se discuten las problemáticas encontradas durante el proceso y cómo se superan, proporcionando una visión realista del proceso que implica llevar a cabo un proyecto de estas características.

Este trabajo no solo representa un desafío técnico, sino también una oportunidad para explorar las posibilidades que la robótica ofrece actualmente. Se espera contribuir al avance del conocimiento en el campo de la robótica de consumo y abrir camino para futuras investigaciones y desarrollos en el área de la conducción autónoma, especialmente enfocado a proyectos a nivel de usuario.

En la siguiente página se encuentra un resumen genérico de los apartados principales de este proyecto.

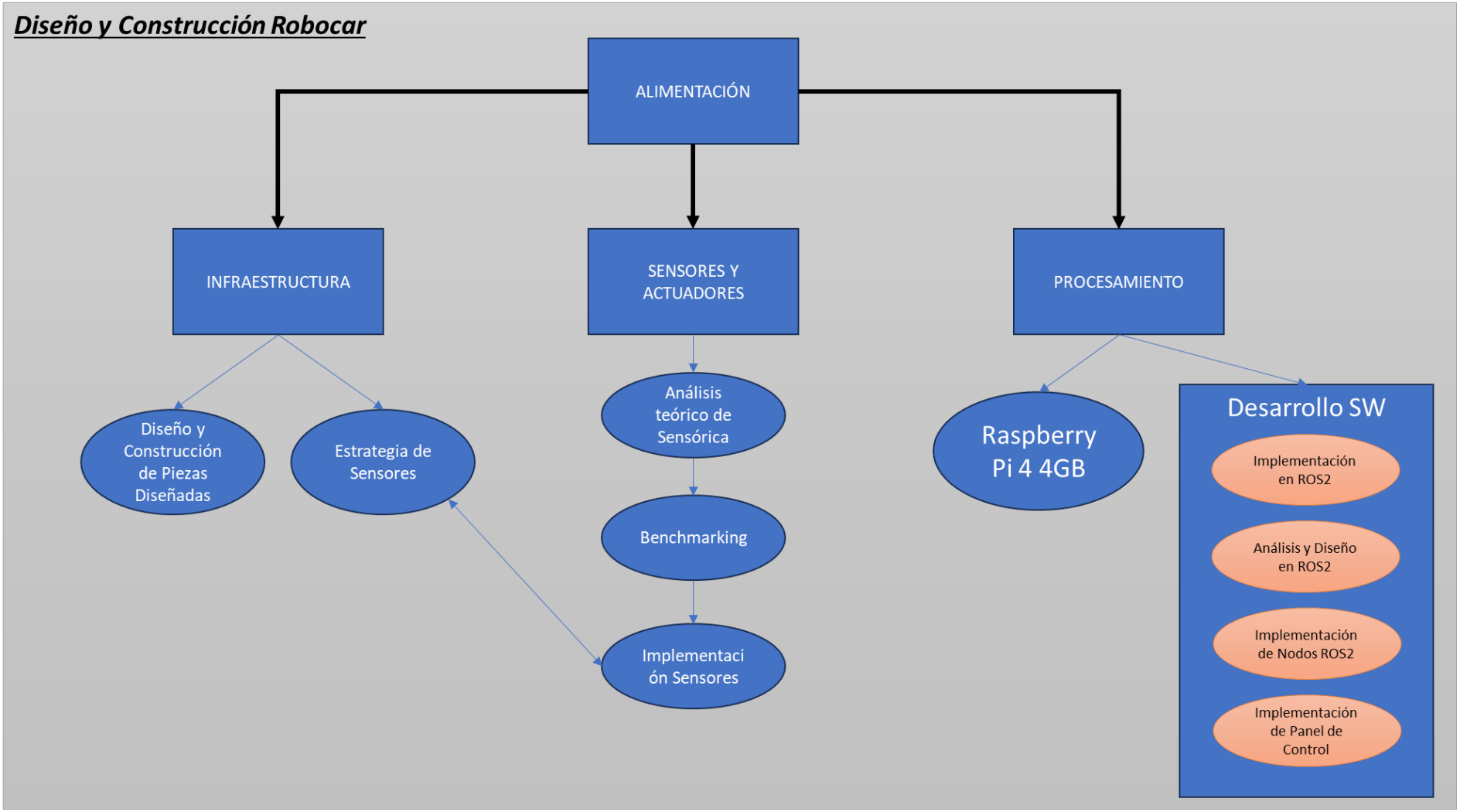


Figura 1: Esquema general del Proyecto

Marco Teórico

Definición de conducción autónoma

Una definición de conducción autónoma podría ser[1]: *"Un automóvil autónomo es un vehículo capaz de detectar su entorno y operar sin la participación humana. No se requiere que un pasajero humano tome el control del vehículo en ningún momento, ni se requiere que un pasajero humano esté presente en el vehículo en absoluto. Un automóvil autónomo puede ir a cualquier lugar al que vaya un automóvil tradicional y hacer todo lo que hace un conductor humano experimentado."*

La Society of Automotive Engineers (SAE) actualmente define 6 niveles de automatización de conducción que van desde el Nivel 0 (totalmente manual) hasta el Nivel 5 (totalmente autónomo). La SAE utiliza el término automatizado en lugar de autónomo. Una razón es que la palabra autonomía tiene implicaciones más allá de lo electromecánico. Un automóvil totalmente autónomo sería consciente de sí mismo y capaz de tomar sus propias decisiones. Por ejemplo, dices "llévame al trabajo", pero el vehículo decide llevarte a la playa. Sin embargo, un automóvil completamente automatizado seguiría las órdenes y luego se conduciría solo.

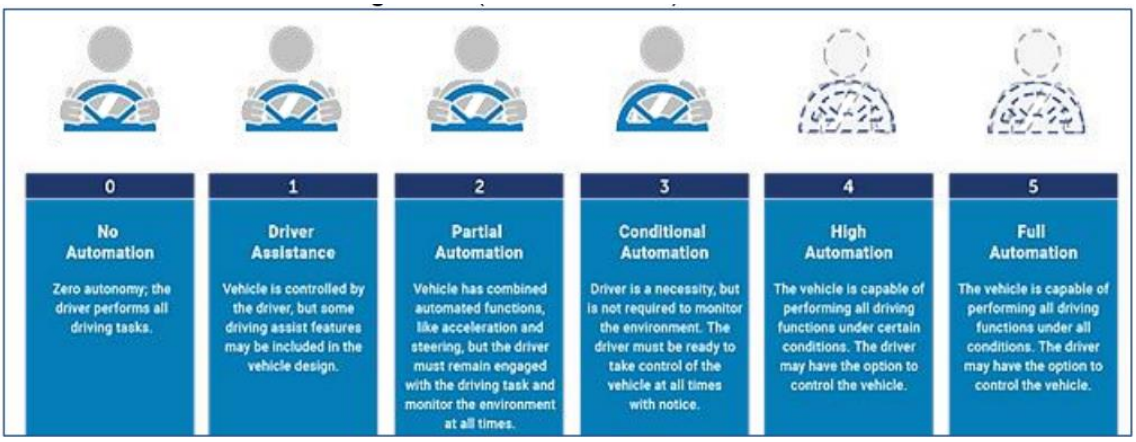


Figura 2 Niveles de automatización de la conducción

- Nivel 0: No hay automatización en la conducción
- Nivel 1: Asistencia a la conducción
- Nivel 2: Conducción automática parcial
- Nivel 3: Conducción automática condicional
- Nivel 4: Alta automatización de la conducción
- Nivel 5: Automatización total de la conducción

[1] https://www.synopsys.com/

Funcionamiento de los coches autónomos

Los automóviles autónomos se basan en sensores, actuadores, algoritmos complejos, sistemas de aprendizaje automático y procesadores potentes para ejecutar el software.

Los vehículos autónomos crean y mantienen un mapa de su entorno basado en una variedad de sensores ubicados en diferentes partes del vehículo. Los sensores de RADAR monitorean la posición de los vehículos cercanos. Las cámaras de vídeo detectan los semáforos, leen las señales de tráfico, rastrean otros vehículos y buscan peatones. Los sensores LIDAR (detección y rango de luz) rebotan pulsos de luz en los alrededores del automóvil para medir distancias, detectar bordes de caminos e identificar marcas de carril. Los sensores ultrasónicos en las ruedas detectan bordillos y otros vehículos al estacionar.

Luego, un software sofisticado procesa toda esta información sensorial, traza un camino y envía instrucciones a los actuadores del automóvil, que controlan la aceleración, el frenado y la dirección. Las reglas codificadas, los algoritmos para evitar obstáculos, el modelado predictivo y el reconocimiento de objetos ayudan al software a seguir las reglas de tráfico y superar los obstáculos.

Sensores utilizados en Vehículos Autónomos

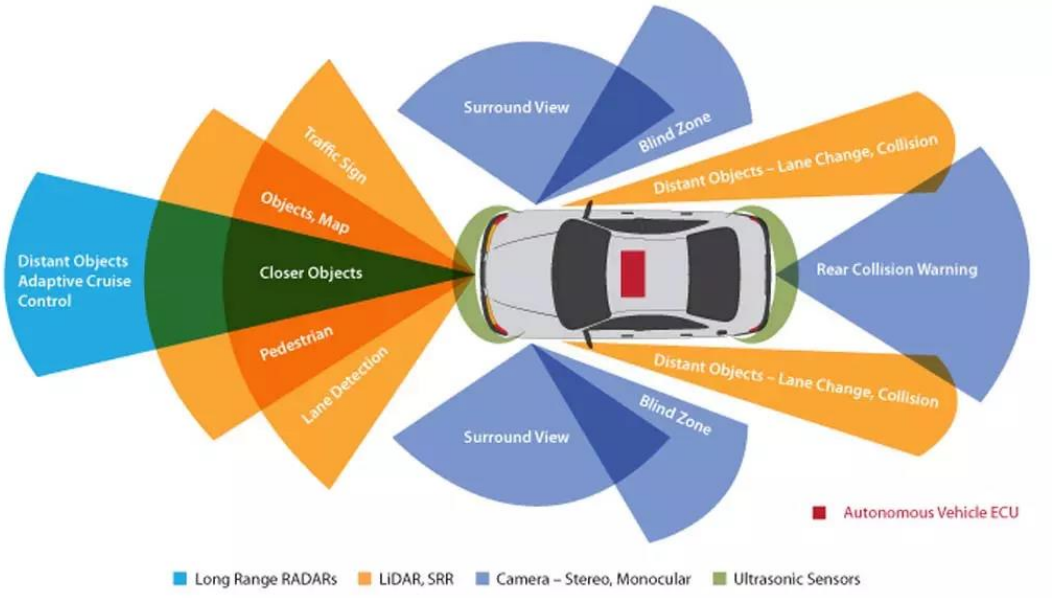


Figura 3 Esquema de sensorización de un vehículo autónomo

Se requiere una variedad de sensores y sistemas informáticos para que un vehículo replique el proceso humano tradicional de comprender su ubicación actual, la ubicación deseada y cómo navegar de manera segura por las rutas y carreteras. Entre los sensores disponibles actualmente, ninguna unidad individual es capaz de descifrar su entorno de manera que proporcione información suficiente para la conducción autónoma. En cambio, los ingenieros diseñan un sistema que incorpora una combinación de sensores, aprovechando estratégicamente las capacidades específicas de cada uno y cubriendo las deficiencias de otros. A continuación, describimos los sensores más comunes utilizados en las ayudas al conductor y los sistemas autónomos:

- **RADAR**. Estos sensores de radio, utilizados en barcos y aviones, emiten ondas electromagnéticas que, cuando son reflejadas, revelan la posición exacta de un obstáculo y lo rápido que se aproxima al vehículo. Relativamente similar a lo que los sensores de ultrasonidos logran, pero con un rango de alcance mucho mayor. ¿El problema? Que estos sensores solo captan información en 2D. Cuando la altura se convierte en un factor clave, estos radares no ofrecen una imagen completa de la situación. Afortunadamente, ya se trabajan en radares 3D que capten una mayor cantidad de información.
    - *PROS*: Son especialmente útiles en condiciones climáticas adversas, donde otros sensores como cámaras y lidar pueden fallar. No necesita una línea de visión directa; funciona bien en niebla densa, lluvia y nieve. Eficaz para medir velocidades relativas.
    - *CONS*: El campo de visión estrecho (estacionario) requiere varias unidades para una cobertura de 360 grados. Baja resolución. Sin color, contraste o reconocimiento óptico de caracteres.
- **LIDAR**. Es como los ojos y los oídos para un ser humano. Tiene capacidad de generar millones de haces, ofrece una visión de 360 grados y un alcance de aproximadamente la longitud de dos campos de fútbol alrededor del coche. Basados en láser, este sensor detecta formas y genera un mapa 3D del entorno del vehículo en tiempo real. Para ello se apoya inevitablemente en la información captada por las cámaras del vehículo, que permite a la unidad central de procesamiento diferenciar entre una persona, un ciclista, un motorista o, simplemente, un objeto situado en el arcén.
    - *PROS*: Crea un mapa 3D preciso de los alrededores de un vehículo. Funciona bien con poca luz.
    - *CONS*: Problemas al operar en niebla densa, lluvia y nieve. El más caro de la matriz de sensores. Sin color, contraste o reconocimiento óptico de caracteres. Necesita línea de visión directa.
- **Cámaras de Video**. Graba varias tomas fijas (fotogramas) para capturar una imagen en movimiento bidimensional.
    - *PROS*: Proporciona color, contraste y reconocimiento óptico de caracteres. Muy accesible.
    - *CONS*: Campo de visión limitado. Problemas con el cambio de luces y sombras, niebla densa, lluvia, resplandor solar, condiciones de poca luz. Muy intensivo en recursos de CPU.
- **Sensores de ultrasonidos**. Imitando a los murciélagos y los submarinos, estos sensores emiten ondas sonoras. Cuando estas ondas impactan con objetos, producen ecos, que son captados por los sensores de ultrasonido. Midiendo las diferencias entre la onda emitida y la captada, el vehículo es capaz de medir distancias y detectar obstáculos próximos. La eficacia de estos sensores es máxima en bajas velocidades y con obstáculos relativamente cercanos. Una de sus principales aplicaciones es el aparcamiento automático o la detección de obstáculos a baja velocidad.
    - *PROS*: Muy preciso en distancias cortas. Funciona bien en niebla densa, lluvia y todas las condiciones de luz. Pequeño y económico.
    - *CONS*: Rango limitado. Sin color, contraste o reconocimiento óptico de caracteres. No es útil para medir la velocidad.

- **Unidad de Medida Inercial (IMU)** - Combinación de acelerómetros y giroscopios que determinan el movimiento lineal y angular de un vehículo.
    - *PROS*: Las IMU actuales son compactas y económicas. Proporciona retroalimentación del movimiento real del vehículo. Funciona en todas las condiciones (instalado cerca del centro de gravedad del vehículo en el interior).
    - *CONS*: Se necesita mayor precisión; Los giroscopios de fibra óptica (FOG) son caros. Las señales tienen deriva debido a la integración matemática de la aceleración para determinar la velocidad y la posición (integración doble).
- **Global Navigation Satellite System (GNSS)** - Utiliza satélites para proporcionar posicionamiento geoespacial autónomo. El sistema de posicionamiento global (GPS) es un subconjunto de GNSS.
    - *PROS*: Cobertura mundial. Operación para todo clima. Proporciona posicionamiento entre vehículos que no pueden verse entre sí. Proporciona posicionamiento cuando no hay señales o marcas viales visibles.
    - *CONS*: La precisión depende del número de satélites en el campo de visión. La precisión general es inferior a la de otros sensores (+/- 1 m en uso público)

El arte de unir los datos de múltiples sensores en tiempo real permite que una CPU determine el estado del vehículo y su entorno. Este conocimiento permitirá tomar las acciones correctas en cada instante a través de los actuadores del vehículo. Las deficiencias individuales de cada tipo de sensor no se pueden superar utilizando más del mismo sensor, por lo que los datos de diferentes tipos de sensores deben combinarse para enmascarar las deficiencias de los sensores individuales. En resumen, el todo es mayor que la suma de sus partes. Al fusionar los sensores en un sistema integrado, los ingenieros permiten que el vehículo identifique su entorno y las amenazas con mayor precisión.

Toda la información se analiza y procesa a través del procesador del coche, que se encarga de manejar o enviar las acciones a realizar por los actuadores que controlan el volante, los frenos, el acelerador, etc. Todos los datos captados por los diferentes sensores y sistemas son procesados mediante algoritmos e inteligencia artificial para que el coche sepa reaccionar y aprenda cómo reaccionar antes posibles situaciones.

Arquitectura Funcional

Podemos dividir las funciones de un vehículo autónomo en 3 capas diferenciadas que colaboran con el objetivo de cumplir con el objetivo final que no es más que la conducción desde un punto origen hasta uno final de forma segura respetando las reglas de tráfico.

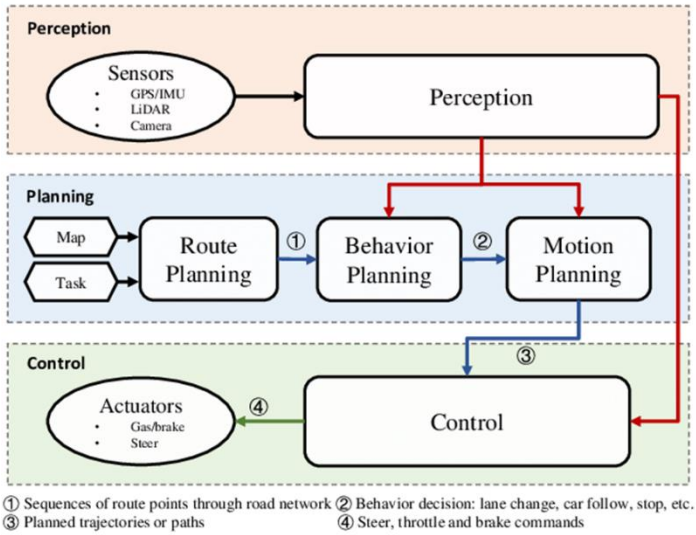


Figura 4: Diagrama Arquitectura Funcional

- **Percepción**. Gracias a los sensores proporciona información acerca del entorno, la posición, las señales, los carriles, los obstáculos, etc del vehículo. Los datos de los sensores son procesados en esta capa para poder ser proporcionados a las capas de planning y control de forma fiable y en tiempo real. Funciones que se encuadran en esta capa son por ejemplo:
    - Localización
    - Fusión de sensores
- **Planificación y decisión**. Genera la ruta hacia el destino y gestiona los eventos que se producen en el camino respetando las reglas de tráfico
- **Control**. Son los actuadores del vehículo: dirección, freno, aceleración, etc. Recibe órdenes de los niveles de percepción en casos de eventos sobrevenidos y de planificación para seguir la ruta.

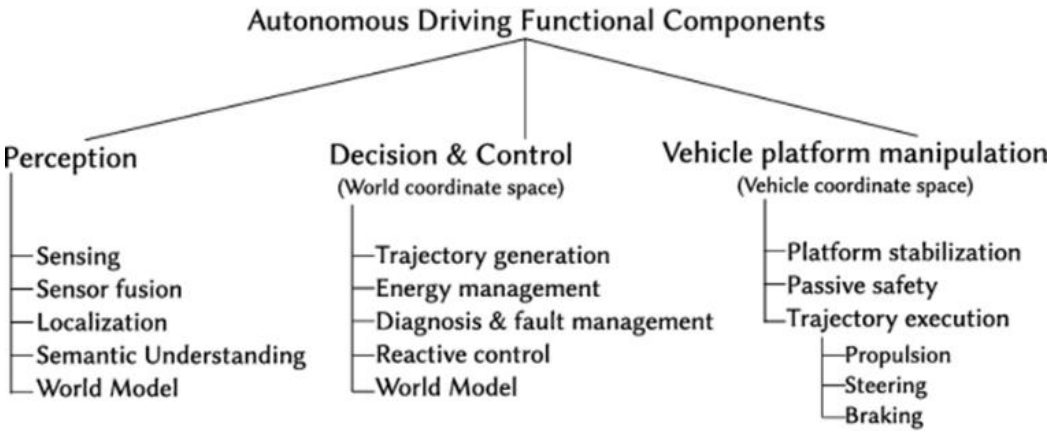


Figura 5: Funciones de Coche autónomo

Objetivos

Teniendo en cuenta todo lo anterior y la complejidad de cada apartado, se han planteado los siguientes objetivos:

- *Implementar un sistema de baterías capaz de soportar la carga del coche*
- *Demostrar las virtudes de sensores concretos en relación con aspectos específicos de la conducción autónoma mediante un benchmarking*
- *Incorporar un entorno ROS2 sobre el que programar la recolección y procesamiento de datos de los sensores*
- ***Objetivo final nº1**: Montaje del coche*
- ***Objetivo final nº2**: Implementar panel de control con datos en tiempo real de los sensores*

Metodología de trabajo general

Desarrollo en espiral incremental

El procedimiento de desarrollo en espiral incremental es una metodología estructurada y cíclica que se utiliza para desarrollar y probar sistemas complejos de manera iterativa. Este enfoque combina elementos del modelo en espiral y del desarrollo incremental, permitiendo una integración y prueba continuas de los componentes del sistema. A continuación, se describe el procedimiento en detalle:

Pasos del procedimiento de desarrollo espiral incremental

Definición de Objetivos y Planificación:

- Objetivos: Establecer claramente los objetivos del ciclo de desarrollo y prueba. Esto incluye identificar los requisitos específicos que se abordarán en la iteración actual.
- Planificación: Planificar las actividades necesarias para lograr los objetivos. Esto incluye la selección de los componentes a desarrollar y probar, así como la asignación de recursos y la programación de tareas. Para cada iteración se realiza un diagrama de GANTT, el cual establece plazos laxos para tareas estrictas asignadas a cada miembro del equipo.

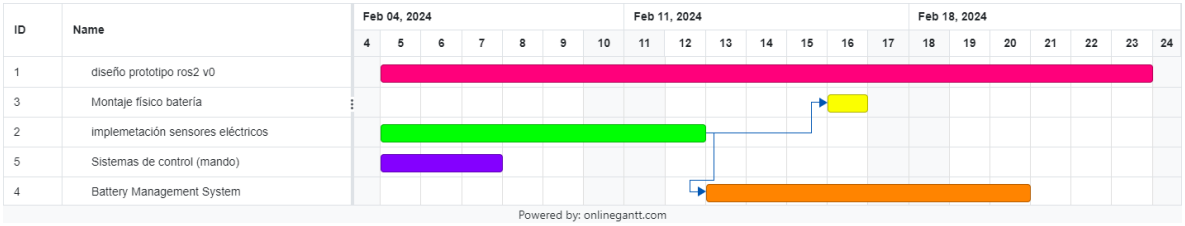


Figura 6 Diagrama de GANTT de una de las primeras iteraciones

Análisis de Riesgos y Prototipado:

- Identificación de Riesgos: Identificar y analizar los riesgos asociados con el desarrollo y la prueba de los componentes seleccionados. Incluye riesgos técnicos, de integración y de cumplimiento de requisitos.
- Prototipado: Crear prototipos o modelos iniciales para mitigar los riesgos identificados. Estos prototipos pueden ayudar a validar conceptos y a descubrir problemas tempranos en el ciclo de desarrollo.

Desarrollo Incremental:

- Implementación: Desarrollar los componentes seleccionados de manera incremental. Esto implica construir, codificar, integrar y compilar partes del sistema.
- Pruebas Unitarias: Realizar pruebas unitarias en cada componente desarrollado para asegurar que funcionen correctamente de manera aislada.

Integración y Pruebas:

- Integración Incremental: Integrar los componentes desarrollados con los componentes existentes del sistema. Este proceso se realiza de manera incremental, añadiendo nuevos componentes en cada iteración.
- Pruebas de Integración: Realizar pruebas de integración para asegurar que los componentes funcionen juntos de manera coherente y sin conflictos.

Evaluación y Revisión:

- Evaluación de Resultados: Evaluar los resultados de las pruebas y el desempeño del sistema integrado. Identificar cualquier problema o defecto que haya surgido durante las pruebas.
- Revisión de Objetivos: Revisar los objetivos y el plan de la siguiente iteración en base a los resultados de la evaluación. Ajustar los objetivos y el plan según sea necesario para la siguiente fase del ciclo.

Repetición del Ciclo:

- Iteración: Repetir el ciclo de desarrollo y prueba para la siguiente serie de componentes o requisitos. Cada ciclo incrementa la funcionalidad del sistema y mejora su estabilidad y rendimiento.
- Periodo del ciclo: Se establece un periodo de ciclo flexible con una duración aproximada de dos semanas. Depende de los objetivos establecidos en cada iteración y del estado de desarrollo del proyecto en cada momento.

Ventajas

El procedimiento de desarrollo en espiral incremental ofrece varias ventajas significativas: permite la identificación y mitigación temprana de riesgos, reduciendo la probabilidad de fallos graves en etapas posteriores; proporciona flexibilidad para realizar ajustes y mejoras continuas basadas en la retroalimentación; cada iteración asegura que cada componente funcione correctamente antes de añadir más complejidad; y permite la detección y corrección temprana de errores.

Conclusiones

El procedimiento de pruebas espiral incremental es una estrategia efectiva para desarrollar y probar sistemas complejos de manera controlada. Al combinar desarrollo incremental con un enfoque en la gestión de riesgos y la mejora continua se permite la creación de sistemas robustos y fiables, adaptándose a cambios y nuevas necesidades a lo largo del ciclo de desarrollo.

El ajuste de este procedimiento a proyectos que combinan hardware y software en conjunto es excelente, mayormente debido a la detección temprana de errores.

Planificación

La planificación se realiza durante el inicio de cada iteración del desarrollo, tal y como se explica en el Planificación. Esto, es estableciendo reuniones de proyecto con una frecuencia mínima de cada dos semanas.

Entorno de trabajo

Se cuenta con un laboratorio de hardware con diversidad de herramientas para el desarrollo de este proyecto. La mayor parte de la construcción física de componentes, así como el trabajo manual de electrónica se realizan en este espacio.

Esto es, elaboración de placas y de cables, integración de componentes físicos, pruebas físicas, impresión de piezas, etc.

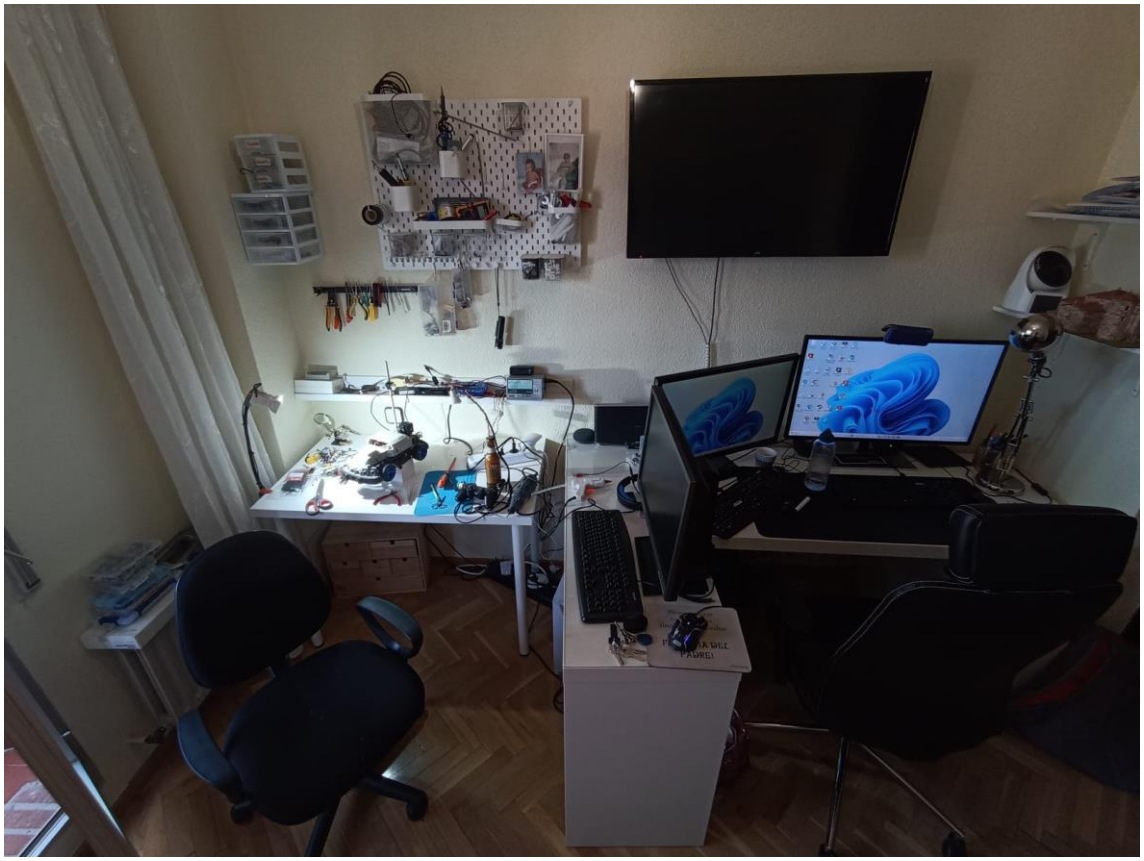


Figura 7 Entorno de Trabajo

Este espacio sirve a su vez como sala de reuniones en caso de ser presenciales.

Inventario

Para realizar el inventario, se han tenido en cuenta los componentes y herramientas principales para la construcción del robot. Adicionalmente se han tenido que usar herramientas como destornilladores, peladores de cable, pinzas, etc... que no están presentes en la tabla.

Tabla 1: Inventario

Componente	Precio
Chasis Robot DFRobot GPX: Asurada	110€
Raspberry Pi 4 4GB	80€
Sensor VL53L0X	1€
Sensor VL53L1X	4€
Sensor VL6180X	10€
Sensor HC-SR04	<1€
Sensor GP2YE03	3€
Sensor KY-032	<1€
RPi Camera Rev 1.3	10€
MPU6050	12€
PCA9685	8€
Display Batería 3S	1€
x6 celdas de iones de litio	Reciclado
PCBs	10€
Cables	2€
Resistencias	10€
Soldador	30€
Fuente de alimentación auxiliar	Reciclado
Total	**293€**

Hay que tener en cuenta que dentro de “Chasis Robot DFRobot GPX: Asurada” están los dos motores ESC para controlar la velocidad del robot, un servomotor para controlar la dirección y un mástil para sujetar la cámara, además de todo lo necesario para la construcción del robot (chasis, ruedas, etc...).

Actuadores

BLHeli_S (Motor ESC sin escobillas)

BLHeli_S es una variante del firmware BLHeli diseñado específicamente para controladores de velocidad electrónicos (ESC) que utilizan microcontroladores basados en silicio. BLHeli_S se ha hecho popular, especialmente en el mundo de los drones de carreras y coches radiocontrol, por su capacidad para ofrecer un control suave y responsivo de los motores sin escobillas.

Este controlador recibe señales PWM para controlar los motores del robot de manera precisa. Según las especificaciones, para inicializar los motores hay que seguir la siguiente secuencia:

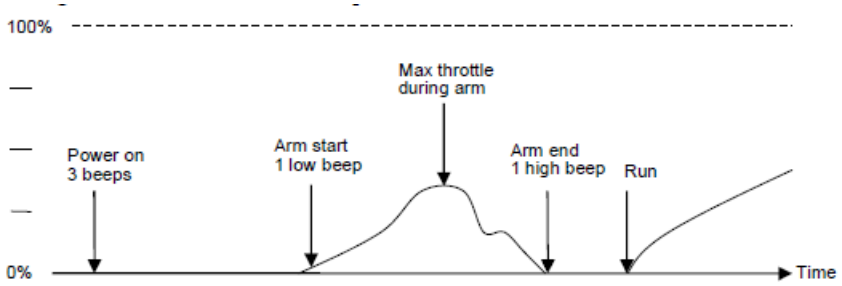


Figura 8: Secuencia de armado BLHeli_S

También proporciona una secuencia de armado para poder controlar los motores. Esta secuencia de armado emite una serie de sonidos que representan el estado del controlador:

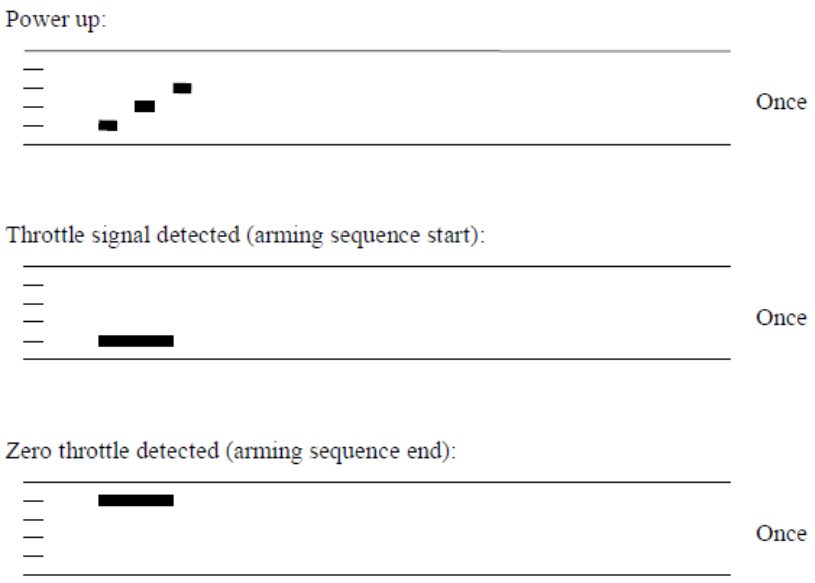


Figura 9: Estados BLHeli_S

Servomotor MG996R

Modelo popular y robusto utilizado frecuentemente en proyectos de robótica y otras aplicaciones que requieren un control preciso del movimiento.

Este servomotor es el encargado de manejar la dirección angular del robot, y al igual que los motores ESC, se controlan a través de una señal PWM.

Figura 10: Servomotor MG99R

Sensores

Los sensores proporcionan los datos necesarios para que un sistema pueda tomar decisiones informadas, adaptarse a cambios en su entorno, y realizar tareas con precisión, es decir, son esenciales para la autonomía y la funcionalidad de los robots.

En general los sensores considerados para el sistema se pueden clasificar en las siguientes categorías:

- **Sensores de distancia**: Se encuentra gran variedad implementados con diferentes tecnologías.
- **Sensores de visión**: Cámara.
- **Sensores de movimiento**: IMU (Unidad de medición inercial).
- **Sensores eléctricos**: Para la monitorización de parámetros eléctricos relativos a la alimentación propia del sistema.

El objetivo es desarrollar una estrategia de sensores suficientemente sofisticada para que los datos obtenidos por los mismos sean adecuados para una conducción autónoma sencilla en todos sus escenarios posibles.

A continuación, se describen sensores contemplados para cada una de las categorías.

Sensores de distancia

Se pretende contemplar un conjunto razonablemente amplio de sensores como posibles implementaciones en el robot, para considerar sus características y seleccionar los mejores ajustados a las cualidades requeridas. Inicialmente se realiza una comparativa puramente teórica a partir de la información disponible.

A continuación, se puede ver una tabla con una lista de los sensores considerados. En esta se especifican las cualidades más relevantes y una breve lista de pros y contras de cada uno.

Sensor	Voltaje de Alimentación	Interfaz de Datos	Rango de Medición	Pros	Contras
VL53L0X	2.6V - 3.5V	I2C	30 mm - 2000 mm	- Precisión superior en comparación con sensores ultrasónicos e infrarrojos. - Funciona bien incluso en condiciones de alta luz infrarroja ambiental. - Compensación de	- Ángulo de medición relativamente estrecho. - Precio algo más alto que otros sensores de distancia.

Sensor	Voltaje de Alimentación	Interfaz de Datos	Rango de Medición	Pros	Contras
				medición para operar detrás de un cristal protector.	
VL53L1X	2.6V - 3.5V	I2C	40 mm - 4000 mm	- Rango de medición ampliado en comparación con el VL53L0X. - Alta precisión en distancias más largas. - Funciona bien en presencia de luz infrarroja ambiental.	- Mayor costo en comparación con otros sensores. - No es adecuado para mediciones de distancias extremadamente cortas.
VL6180X	2.6V - 3.6V	I2C	5 mm - 200 mm	- Rango de medición de 5 mm a 200 mm. - Funciona en presencia de luz infrarroja ambiental. - Precisión aceptable para distancias más cercanas.	- No es tan preciso como el VL53L0X o VL53L1X. - Menor rango de medición en comparación con otros sensores.
HC-SR04	5V	Analógico (Voltaje)	2 cm - 400 cm	- Bajo costo y amplia disponibilidad. - Fácil de usar con Arduino y otros microcontroladores. - Rango de medición de 2 cm a 400 cm.	- Precisión limitada debido a ecos y reflexiones. - No funciona bien en entornos con obstáculos pequeños o superficies irregulares. - No es adecuado para mediciones precisas en

Sensor	Voltaje de Alimentación	Interfaz de Datos	Rango de Medición	Pros	Contras
					distancias cortas.
GP2Y0E03	4.5V - 5.5V	Analógico (Voltaje)	4 cm - 50 cm	- Rango de medición de 4 cm a 50 cm. - Buen rendimiento en detección de objetos cercanos. - Fácil de integrar con Arduino.	- Precisión limitada en comparación con sensores láser. - No funciona bien en distancias más allá de 50 cm. - Sensible a la reflectancia del objetivo. - No es adecuado para mediciones de larga distancia.
KY-032	3.3V - 5V	Analógico (Voltaje)	2 cm - 450 cm	- Bajo costo y fácil de usar. - Rango de medición de 2 cm a 450 cm. - Adecuado para proyectos de robótica y automatización.	- Precisión limitada. - Sensible a la reflectancia del objetivo. - No es adecuado para mediciones de alta precisión.

Tabla 2 Comparativa teórica de sensores de distancia

Cámara

Para la cámara se considera la Raspberry Pi Camera Rev 1.3. Es una elección económica y versátil para proyectos de robótica y vigilancia. Algunas de sus son:

- Costo asequible: Resulta ideal para proyectos con presupuestos limitados.
- Resolución y calidad de imagen: Tiene una resolución de 5 megapíxeles. Captura imágenes nítidas y videos en alta definición de hasta 1080p a 30 fps.
- Compatibilidad: Es compatible con Raspberry Pi 1, 2, 3 y 4, lo que facilita su integración con todos sus modelos. Esto es debido principalmente a un conector específico conocido como CSI (Camera Serial Interface). Es un pequeño conector

ubicado en la parte superior de la Raspberry. Es el punto de conexión para la cámara y permite la transmisión de datos entre la cámara y la placa. La interfaz CSI garantiza una conexión estable y de alta velocidad, lo que es esencial para capturar imágenes y videos de alta calidad.

En resumen, la cámara Raspberry Pi Camera Rev 1.3 ofrece una combinación de características asequibles, calidad de imagen y versatilidad que la convierte en una excelente opción para este proyecto.

Sensores de movimiento

Se considera el MPU6050, un sensor inercial de 6 grados de libertad (DoF) que combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes. Resulta económico y presenta un consumo energético muy contenido. Es ideal para medir la aceleración y la velocidad angular, lo que te facilita implementar funciones de navegación de forma efectiva. Además, su comunicación por I2C facilita su integración con el sistema.

Sensores eléctricos

El objetivo es la monitorización de parámetros eléctricos relativos a la alimentación propia del sistema, para consultar los sensores empleados consultar el apartado Baterías.

Benchmarking de los sensores

Motivación

La motivación para realizar un benchmarking exhaustivo de los sensores de distancia radica en la necesidad de identificar aquellos que ofrezcan el mejor equilibrio entre precisión, coste y fiabilidad para las aplicaciones específicas del coche robot autónomo. Una selección adecuada de sensores no solo optimiza el rendimiento del sistema, sino que también puede reducir los costos y mejorar la seguridad del vehículo.

Precisión y Fiabilidad:

- Precisión: En la navegación autónoma, la precisión de los sensores de distancia es esencial para evitar colisiones y para la correcta interpretación del entorno. Un sensor impreciso puede conducir a errores en la toma de decisiones y comprometer la seguridad del vehículo.
- Fiabilidad: Los sensores deben proporcionar lecturas consistentes y fiables en diversas condiciones ambientales. Un sensor que no puede adaptarse a cambios en la iluminación o a diferentes distancias puede generar datos erróneos y afectar negativamente al rendimiento global del sistema.

Condiciones Variables:

- Distancia: El coche robot debe ser capaz de detectar obstáculos a diferentes distancias, desde objetos cercanos hasta barreras lejanas. Por lo tanto, es crucial evaluar cómo se comportan los sensores a distintas distancias.
- Iluminación: Las condiciones de iluminación pueden variar significativamente, desde baja luz hasta ambientes muy iluminados. Evaluar el rendimiento de los sensores en estas condiciones ayuda a garantizar que el vehículo funcione de manera efectiva en cualquier entorno.

Toma de Decisiones Informada:

La comparación detallada y el análisis de diferentes sensores permiten tomar decisiones informadas sobre cuáles son los más adecuados para las necesidades específicas del proyecto. Esto incluye considerar no solo el rendimiento, sino también factores como el costo, la facilidad de integración y la robustez en condiciones reales.

Coste:

Considerar el coste de los sensores es una variable importante porque se dispone de al menos una unidad de testing de cada uno de los sensores descritos, pero se valora la posibilidad de adquirir más unidades del que sea necesario para la implementación final de la estrategia de sensores. Esto permite gestionar el presupuesto de manera eficiente y asegurar que se pueden escalar las soluciones sin incurrir en gastos excesivos.

Objetivos

El objetivo de este benchmarking es evaluar y comparar el rendimiento de múltiples sensores de distancia en condiciones variables de distancia e iluminación. Esto se logrará mediante un plan de pruebas sistemático que incluye la medición de precisión, tiempo de respuesta y fiabilidad de cada sensor. Al final del proceso, se espera identificar

las fortalezas y debilidades de cada sensor, proporcionando una base sólida para seleccionar los más adecuados para el coche robot autónomo.

Este enfoque garantiza que se seleccionen los sensores más adecuados, optimizando así el diseño y funcionamiento del vehículo, y asegurando su capacidad para operar de manera eficiente y segura en una variedad de entornos.

Plan de pruebas teórico

A continuación, se describe un plan de pruebas teórico para un benchmarking ideal que considera la distancia y la iluminación como variables clave:

Sensores:

Concretar exactamente qué sensores que se van a considerar en la realización del Benchmarking.

Variables:

- Distancia: Se evaluarán 2-3 distancias dentro del rango de15cm a 3m.
- Iluminación: Se considerarán dos o tres condiciones de iluminación para evaluar el rendimiento de los sensores bajo diferentes niveles de luz.

Distancias:

- Distancia Larga (3m): Medir la precisión y la respuesta de cada sensor a una distancia de 3 metros en condiciones de iluminación normal.
- Distancia Media (50 cm): Evaluar la precisión y la respuesta de cada sensor a una distancia de 50 cm en condiciones de baja iluminación.
- Distancia Corta (15 cm): Probar la precisión y la respuesta de cada sensor a una distancia de 15 cm bajo condiciones de alta iluminación.

Iluminación:

- Condiciones de Iluminación Normal: Mantener una iluminación estándar y evaluar la precisión de los sensores a una distancia media (por ejemplo, 50 cm).
- Baja Iluminación: Reducir la iluminación ambiente y medir la precisión de los sensores a una distancia larga (por ejemplo, 3 metros).
- Alta Iluminación: Aumentar la iluminación ambiente y evaluar la precisión de los sensores a una distancia corta (por ejemplo, 15 cm).

Métricas de Evaluación:

- Precisión: Comparar las lecturas del sensor con las distancias reales establecidas.
- Tiempo de Respuesta: Medir el tiempo que tarda cada sensor en detectar cambios en la distancia.
- Fiabilidad: Evaluar la consistencia de las lecturas del sensor bajo diferentes condiciones.

Procedimiento:

- Configurar cada sensor según las especificaciones del fabricante.
- Establecer las condiciones de prueba según el plan descrito anteriormente.
- Registrar las lecturas del sensor y las distancias reales para cada prueba.

- Analizar los datos recopilados y comparar el rendimiento de cada sensor bajo diferentes condiciones.
- Identificar fortalezas y debilidades de cada sensor en función de los resultados obtenidos.

Consideraciones Adicionales:

- Repetir las pruebas múltiples veces para obtener resultados más precisos.
- Utilizar un entorno controlado para minimizar variables externas que puedan afectar los resultados.
- Documentar cualquier anomalía o problema encontrado durante las pruebas para informar sobre la fiabilidad y la calidad de cada sensor

Este plan de benchmarking proporciona un marco sólido para comparar y evaluar el rendimiento de varios sensores de distancia en condiciones variables, permitiendo tomar decisiones informadas sobre qué sensor es más adecuado para diferentes aplicaciones.

Plan de ejecución de las pruebas

Implementación de los Sensores

Conectar y configurar todos y cada uno de los sensores según las especificaciones del fabricante. Requiere implementación de drivers específicos y circuitería analógica complementaria.

Realización de las Pruebas

Ejecutar las pruebas seleccionadas del plan de pruebas.

Recopilación y Presentación de Datos

Registrar todas las mediciones obtenidas. Mostrar los resultados en una tabla estructurada para facilitar el análisis comparativo.

Implementación de los sensores

La lista de sensores disponibles a la fecha de ejecución de las pruebas del benchmarking es: VL53L0X-v2, VL6180X, HC-SR04, GP2Y0E03, KY-032.

En general, el proceso a seguir para la implementación de los sensores es el siguiente:

- Se conecta a la Raspberry Pi asegurando una conexión firme y correcta de los pines de alimentación y datos, siguiendo las especificaciones del fabricante.
- En el software, se instalan las librerías necesarias, se configuran las direcciones de comunicación (I2C, GPIO, etc.) y se calibran los sensores para su correcto funcionamiento.

A continuación, se describe a grandes rasgos el proceso de implementación de cada uno de ellos.

VL53L0X

El sensor se conecta a la Raspberry Pi a través de la interfaz I2C.
Se sueldan los pines en la placa del sensor.
Se revisa el datasheet y las herramientas disponibles del fabricante en detalle.

Resulta destacable la API ofrecida por el fabricante para este sensor.
La API del VL53L0X es un conjunto de funciones en C que permiten el desarrollo de aplicaciones finales, controlando el sensor VL53L0X para realizar tareas como la inicialización y la medición de distancias. Existe un manual con el que se explica en detalle el funcionamiento (vl53l0x).

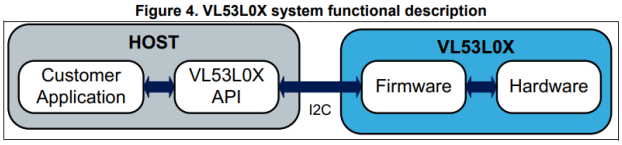


Figura 11 Descripción del sistema a nivel funcional

- **Código Fuente**: La API del VL53L0X está compuesta por código fuente en lenguaje C.
- **Control de Funciones**: Permite el control completo de las funciones de medición de distancia.
- **Portabilidad**: La API está diseñada para ser fácilmente portada y compilada en cualquier plataforma de microcontrolador, así como una máquina de estados con los modos de medición y funciones para conmutar entre ellas.
- **Ejemplos Incluidos**: Se incluyen varios ejemplos que demuestran cómo usar la API para realizar mediciones de distancia con las placas de expansión Nucleo F401 y VL53L0X.

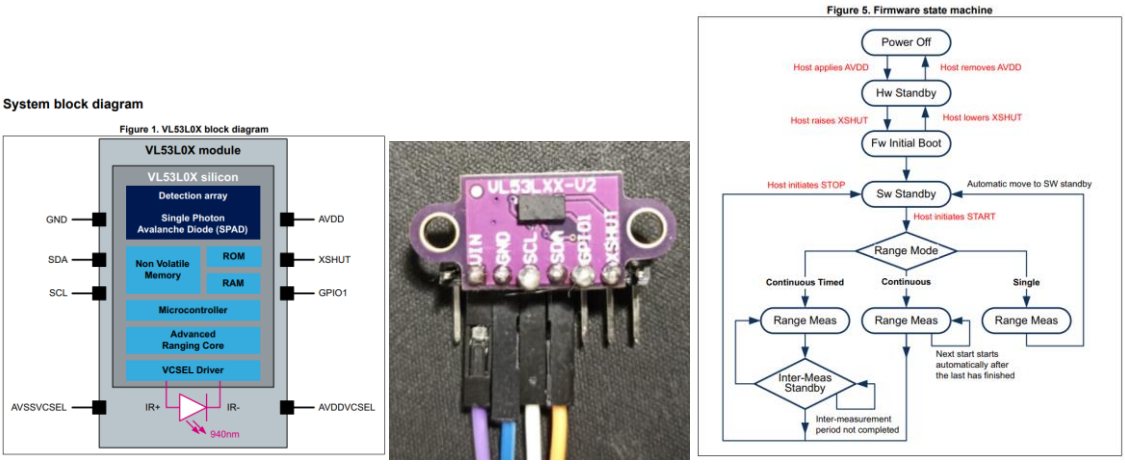


Figura 12 Diagrama de bloques (izq), sensor conectado (centro), máquina de estados de la API (derecha)

En el software, se instala la librería específica para VL53L0X, se configura la dirección I2C y se calibra el sensor según las especificaciones.
Se conecta el sensor y se obtienen mediciones individuales correctas, así como información adicional sobre su estado.

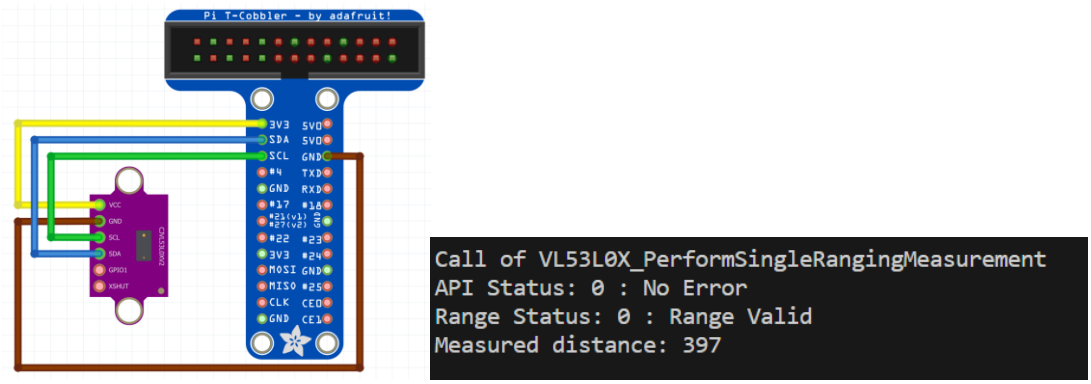


Figura 13 Conexión del sensor y log de salida del script de prueba

Una vez implementado y probado, el sensor se confirma como candidato del benchmarking.

VL6180X

Se conecta a la Raspberry a través de I2C y se implementa un script de prueba en Python empleando la librería adafruit_vl6180x.
Una vez implementado y probado, el sensor se confirma como candidato del benchmarking.

HC-SR04

Se consulta el datasheet y se establece una interfaz física concreta entre el sensor y la Raspberry mediante un código de colores. Es necesario incluir un divisor de tensión entre la salida del sensor y la entrada de la Raspberry para reducir su tensión de entrada.
Referencia al sitio web: (HCSR04)

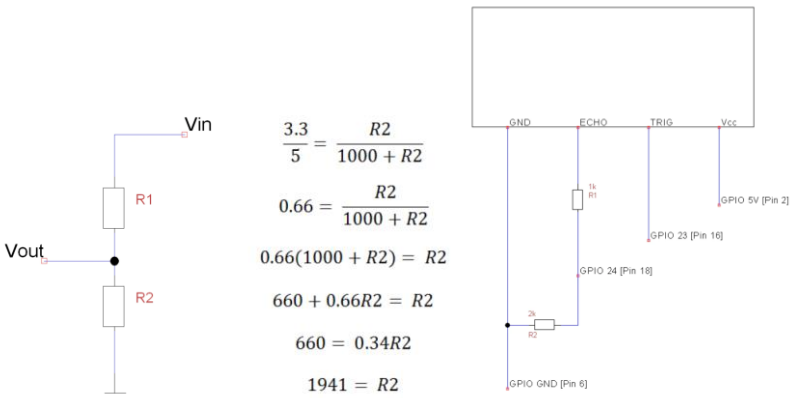


Figura 14 Divisor de tensión R1=1kOhm R2=2Kohm y conexionado

Una vez conectado a través del divisor de tensión, se realizan medidas de pruebas satisfactorias, quedando confirmado como candidato del benchmarking.

GP2Y0E03

Se consulta el datasheet y se establece una interfaz físico concreta entre el sensor y la Raspberry mediante un código de colores, descrita en la tabla de debajo.

Cable del sensor		
Vin IO	4	naranja
Vdd	1	Rojo
Gnd	3	Negro
SCL	6	Verde
SDA	7	Amarillo
Vout (A)	2	Azul

Tabla 3 Interfaz física con código de colores

El sensor tiene interfaz I2C así como analógica.
Referencia al sitio web: (gp2y)
Se realizan pruebas mediante ambas interfaces. Para ello se implementan pruebas en Python con la librería SMBus y en C empleando la librería WiringPiI2C.h y en sketches de Arduino empleando Wire.h. Se incluyen las pruebas con Arduino Nano debido a la disponibilidad de código público.
Se realizan pruebas en las que se concibe comportamiento errático, dando medidas correctas e incorrectas de forma impredecible.

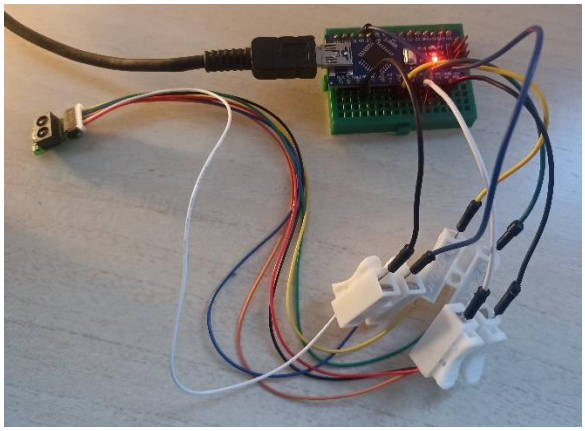


Figura 15 Pruebas del sensor con Arduino

Debido al comportamiento poco fiable del sensor se descarta como candidato para el benchmarking.

KY032

El sensor de distancia KY032 decide omitirse del Benchmarking debido a sus características específica. El output de este sensor es un valor digital booleano, y su rango de medición es de muy corto alcance.
Por este motivo no resulta comparable con los demás sensores en el benchmarking, quedando descartado del mismo.

Realización de las pruebas

Se realizan mediciones de distancia considerando principalmente los rangos de corta y media distancia. Para cada medida, se realiza una medida física y una medición con el sensor. En ambos casos se prueba en el escenario de iluminación abundante así como iluminación escasa. Toda medición se realiza al menos 3 veces para asegurar la fiabilidad

del dato. El objeto al que se mide la distancia es un ortoedro blanco de cartón. No refleja la luz y su color es similar al fondo.
Las condiciones de luz se definen como luz abundante para un valor aproximado de 150 lux en el objeto target y como luz escasa para un valor aproximado de 2 lux en el mismo.
Se realizan medidas para la precisión en la medición de distancias y para el ángulo de detección de los sensores.

Pruebas de precisión en medidas de distancia

Se determinan valores concretos para la distancia física entre el sensor y el objeto.
Se realizan la prueba para cada valor, sensor y condición de luz.
Las pruebas de los sensores VL5 y VL6 se realizan de forma simultánea debido a la facilidad de conectar ambos al bus I2C e integrarlos en una misma protoboard.
A continuación se pueden apreciar imágenes del setup de pruebas.

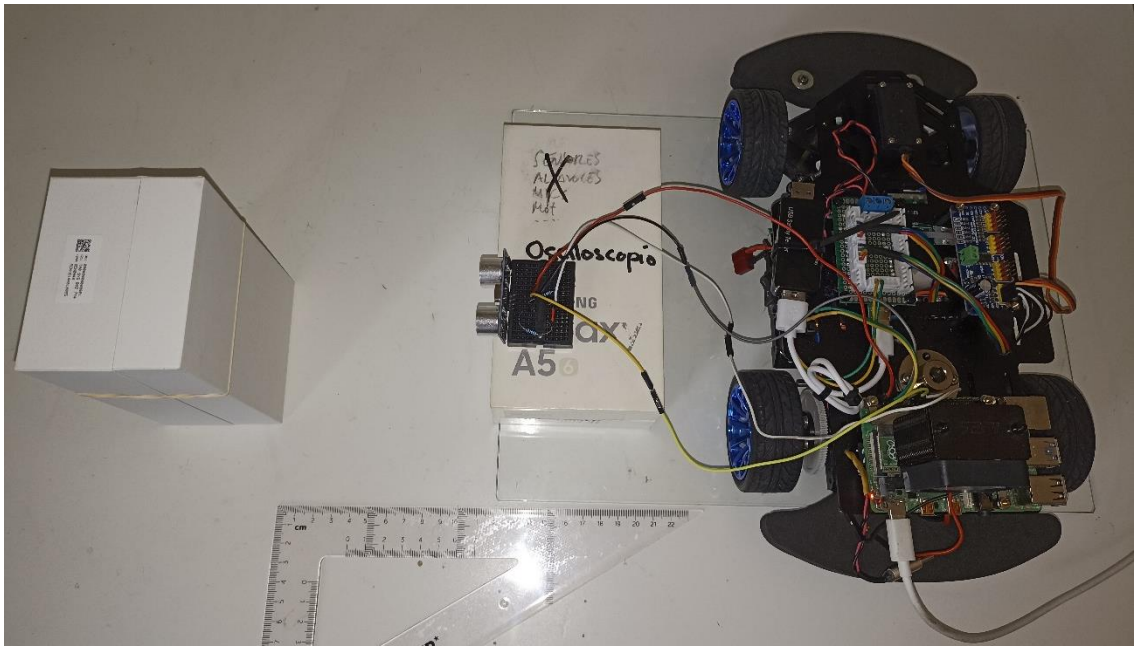


Figura 16 Prueba de distancia con el sensor HC-SR04

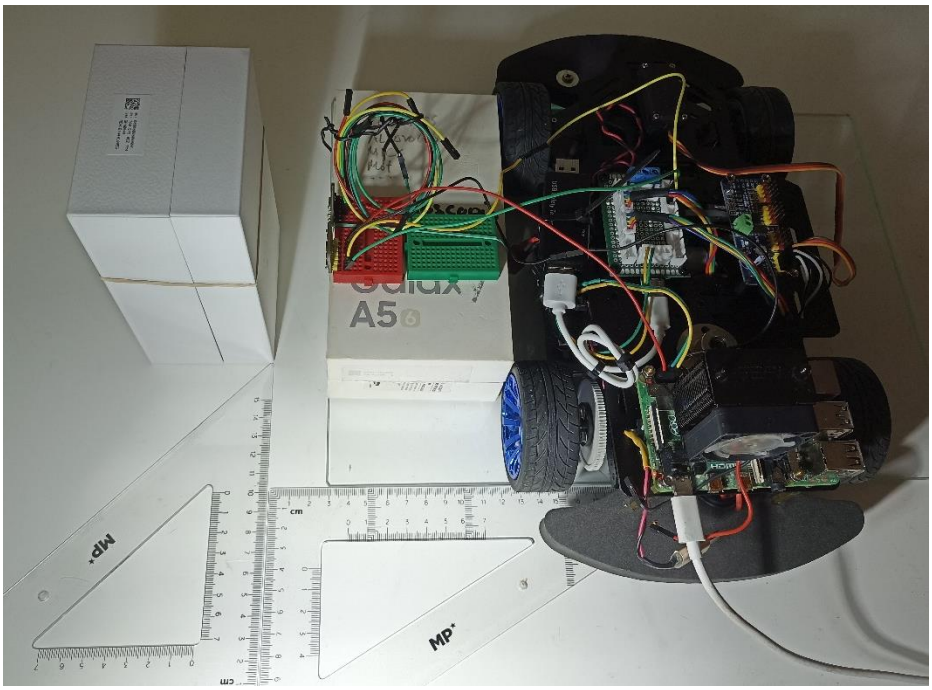


Figura 17 Prueba de distancia con el sensor VL5 y VL6

En las imágenes se puede ver el Setup de pruebas para mediciones determinadas. Previa la realización de la medida del sensor, se adecua la luz a las condiciones determinadas.

Pruebas de ángulo de detección

El objetivo es cubrir desde 0 a 45 grados de ángulo. Para ello, se realizan las medidas siguiendo la Figura de debajo. La distancia a se fija a 30 cm y la distancia b se modifica de forma que se comprueben ángulos entre 0 y 45 grados. Se fijan valores para los ángulos de 0, 15, 20, 30, 45. Se obtienen los valores de distancia mediante la sencilla formula trigonométrica X=a*cos(90-α). Los resultados obtenidos son 0, 7.76, 10.26, 15 y 21.21.

Las pruebas se hacen en condiciones óptimas de luz.

En el caso del sensor VL5, debido a ser de corta distancia se ajustan todas las medidas a la mitad. Obteniendo a = 15 cm y b = [0, 3.38, 5.13 y 10.6]

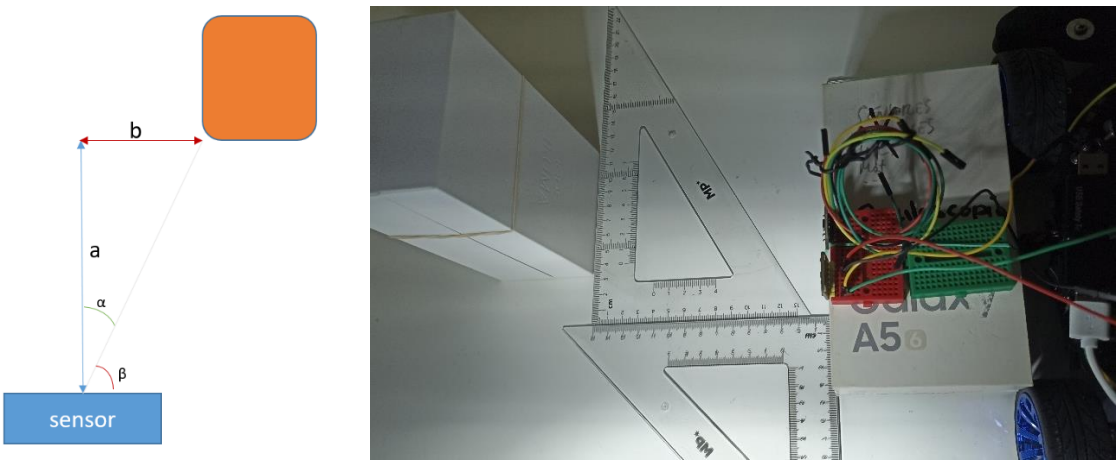


Figura 18 Distancias para medidas de ángulo (izq) y setup de pruebas para medida de ángulos

Resultados de las pruebas

Pruebas de distancia

Sensores		**Corta distancia**			**Media distancia**		
Medición física ->	**Luz**	**3**	**8**	**15**	**30**	**60**	**100**
HC	+	4.76	6.57	13.5	28.27	58.95	100.24
	-	4.5	6.82	14.4	29.8	59.93	100.5
VL6	+	1.6	7.0	13.8	25.5	25.5	25.5
	-	1.6	6.8	13.9	25.5	25.5	25.5
VL5	+	4.8	9.4	17	34.4	69.7	119.2
	-	4.6	9.3	16.9	33.7	71.4	119.6

Tabla 4 Resultado benchmarking prueba de distancia

Todos los valores se indican en cm.

Pruebas de ángulo

Sensor	**0º**	**15º**	**20º**	**30º**	**45º**
HC	1 (28.65)	1(33.07)	0(59)	0(61)	0 (60)
VL6	1 (14.7)	0(25.5)	0(25.5)	0(25.5)	0(25.5)
VL5	1 (43.4)	0 (72.6)	0 (73.1)	0 (73.7)	0 (72.4)

Tabla 5 Resultado benchmarking prueba de ángulo

Los resultados se corresponden con 1 equivalente a una detección correcta (una medida consistente con la distancia del objeto target) y 0 con una detección distinta o errónea.

Conclusiones

Se resumen las conclusiones del benchmarking en los siguientes puntos:

- En general los resultados son muy buenos. Todos los sensores muestran capacidad de medición con errors aceptables (<20%) en rangos y condiciones diversas.
- La cantidad de luz ambiente no afecta de forma notable a ninguno de los sensores.
- El HC demuestra ser considerablemente más preciso de lo esperado.
- Mediciones de distancias muy cortas (<3cm) introducen un error considerablemente significativo.
- El ángulo de medición de todos los sensores es muy estrecho.

Finalmente, se decide incluir el sensor KY-032 en la estrategia de sensores como forma de lidiar con las distancias muy cortas. Se emplea como una última medida de seguridad, para detectar presencia de obstáculos a muy corta distancia.

Sensor KY032

Se prueba empleando un script de prueba disponible en el datasheet del sensor, referencia: (ky032). Se establece una interfaz física con la raspberry y se procede a hacer pruebas y calibración. Por defecto detecta obstáculos a cualquier distancia <10.5cm. La calibración se realiza de forma manual para que se active a los 4.5cm de distancia. Girando el potenciómetro R6 (arriba en la imagen, como de puede apreciar en la Figura de debajo) se consigue ajustar el rango de detección al deseado.

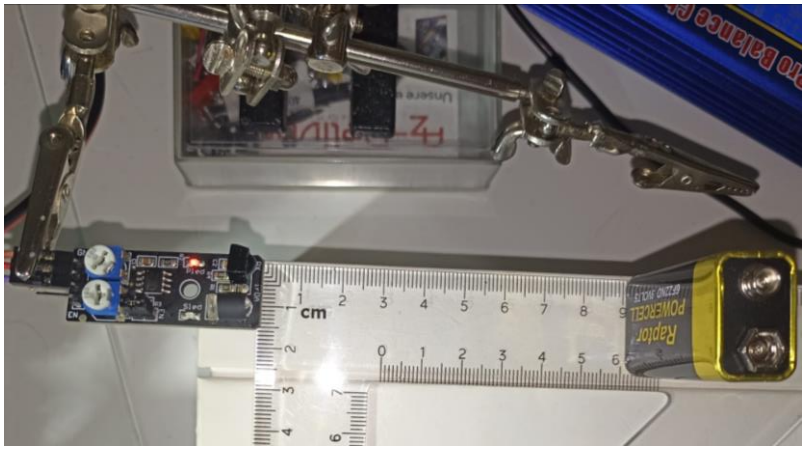


Figura 19 Calibración manual de la distancia de detección del sensor KY032

Estrategia de sensores

Se emplean los resultados finales del benchmarking, para plantear una estrategia final a implementar.

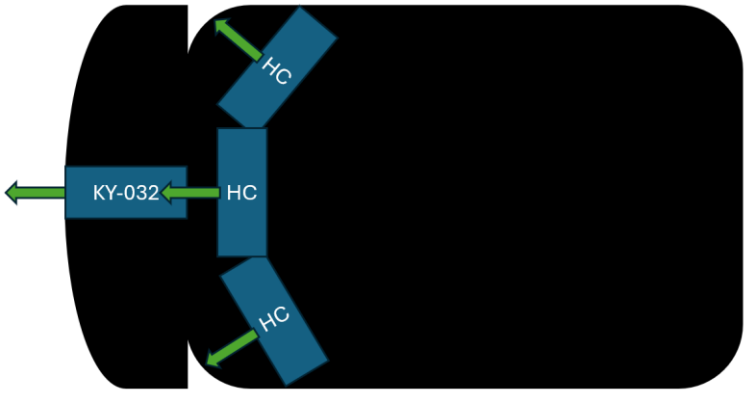


Figura 20 Distribución final de sensores en el robot

La estrategia final implica un sensor KY-032 integrado en la parte frontal del parachoques. Además, se decide incluir 3 sensores HC-SR04 para la medición de distancias en la dirección frontal. Se decide incluir este sensor debido a sus buenos resultados demostrados en el benchmarking. Se distribuyen en la parte frontal separados por un ángulo de 30º, evitando de esta manera puntos ciegos en la medición de distancias frontales.
El planteamiento es que el conjunto de los 3 sensores HC cubra 90º de visión en la dirección frontal con un rango de detección elevado, para evitar y prevenir colisiones de forma anticipada. El KY se integra como última línea de detección, la detección de un obstáculo por parte de este sensor debe tratarse con máxima prioridad por parte del SW debido a la proximidad que supone.
La sensorización de la sección trasera se pospone para líneas futuras.

Implementación de la estrategia de sensores a nivel Hardware

La implementación de la estrategia de sensores a nivel físico abarca tanto la conectividad física de los sensores como su colocación en el robot. Este proceso incluye la asignación de pines, la construcción de cables específicos y el montaje de los sensores en la estructura del robot para garantizar su correcto funcionamiento y optimización de la detección.

Conectividad Física

La conectividad física de los sensores implica:

- **Asignación de Pines:**
  Determinar y asignar los pines definitivos en la Raspberry Pi para la alimentación y los datos de cada sensor. Durante todo el proceso de desarrollo del robot se mantienen actualizadas una tabla de direcciones I2C y otra de pines ocupados de la Raspberry.

- **Interfaces de datos**
  - I2C: La topología de enrutado en bus ofrece una interfaz I2C válida para todos los periféricos que la implementan. La Raspberry tiene 2 pines por defecto dedicados a esta tarea para SDA y SCL.
  - Digital: Cualquier pin GPIO genérico de la Raspberry sirve para esta tarea. Se requiere de dos pines digitales para la implementación de cada uno de los sensoresHC-SR04.
- **Interfaces de alimentación**
  - Placa de distribución de 5V: Se adquiere directamente del BUCK y distribuye directamente a todos los periféricos que lo requieren.
  - 3.3V disponibles directamente de la Raspberry

Funcionalidad	**Pin**
I2C (SDA, SCL)	Gpio i2c (GPIO 2, pin 3; GPIO 3, pin 5)
Trigger HC derecha	GPIO 23, pin 16
Echo HC derecha	GPIO 24, pin 18
Trigger HC centro	GPIO 5, pin 29
Echo HC centro	GPIO 6, pin 31
Trigger HC izquierda	GPIO 17, pin 11
Echo HC izquierda	GPIO 27, pin 13

Tabla 6 Pines ocupados de la Raspberry en un momento puntual del desarrollo.

- **Construcción de Cables Específicos**:
  El proceso de creación de cables personalizados implica seleccionar un cable que se ajuste a las necesidades de conexión, como un cable Dupont, ajustar su longitud según los requerimientos específicos y finalmente conectar los terminales a los pines correspondientes.

  En el caso de los sensores HC-SR04, se requiere la implementación de un divisor de tensión directamente en el propio cable. Esto es necesario debido a la limitación de espacio físico dentro del robot, lo que impide la adición de una placa adicional para gestionar las conversiones de tensión. El divisor de tensión se construye utilizando las resistencias previamente especificadas. Este enfoque permite una integración eficiente del sensor sin comprometer el espacio disponible.
  El proceso de construcción de cada cable resulta delicado y tedioso, pero da un resultado óptimo a nivel de integración de los componentes. El proceso se realiza adquiriendo los componentes (conectores Dupont, cable y resistencias), cortando los cables a la longitud adecuada, trenzando los cables y añadiendo puntos de soldadura a las uniones. Se finaliza añadiendo tubo termo retráctil a las partes más delicadas.

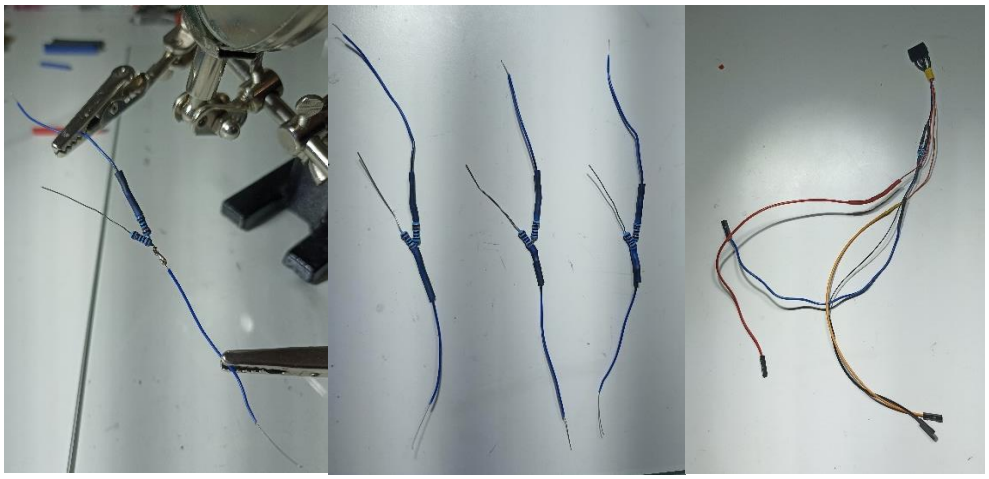


Figura 21 Proceso de construcción de cables para el HC-SR04

Finalmente, se prueban todos y cada uno de los cables construidos antes de implementarlos de forma definitiva en el robot.

Colocación Física de los Sensores

La colocación de los sensores en el robot se realiza según un desarrollo teórico establecido en la estrategia de sensores. Para ello se acatan los pasos descritos a continuación:

- Establecer requisitos para el posicionamiento de los sensores.
- Diseñar y fabricar soportes específicos.
- Montar los sensores en los soportes y estos en la estructura del robot.

Para ello se diseñan las piezas haciendo uso de software de modelado 3D, tal y como se describe en el apartado ‘Diseño y fabricación de piezas 3D a medida’.

Alimentación (Baterías)

A continuación, se explican los aspectos a considerar durante el diseño, construcción e implementación del componente de la batería. El procedimiento seguido para el desarrollo de este componente parte de la definición de una serie de requisitos para el componente, el diseño del componente, la construcción física del mismo y finalmente la integración de este en el sistema.

Se plantea la construcción del componente como una unidad que, a ser posible, debe brindar la posibilidad de reutilizar componentes sueltos disponibles, así como cumplir los requisitos de alimentación y monitorización del sistema.

Se propone un diseño inicial del componente con una configuración 2p3s. En este término se indica que hay un conjunto de 2 celdas en paralelo y 3 conjuntos de celdas en serie. Se requiere a su vez un componente para la protección y gestión de la carga, conocido como Battery Management System (BMS) compatible con la configuración planteada. Finalmente, es necesario un sensor de corriente para la monitorización del componente. A continuación se puede ver un esquema de una batería 2p3s conectada con su BMS (Figura de debajo).

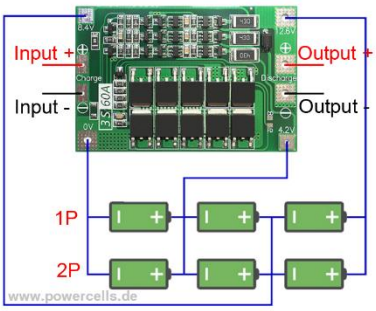


Figura 22 BMS 2p3s

La monitorización se plantea de forma que se pueda medir las tensiones entre las celdas en serie, de forma que se pueda verificar el funcionamiento del BMS encargado de balancear la carga entre las celdas. Asimismo, se debe poder medir la corriente consumida por el sistema completo.

Requisitos

El componente debe cumplir el conjunto de requisitos descritos a continuación.

1. **Alimentación de componentes**. Para ello, se requiere de ciertos valores de tensión y corriente concretos.
    a. Tensión. Valores de tensión requeridos para los diferentes componentes del sistema.
        i. 12V: Motores
        ii. 5V: Raspberry Pi, Sensores

iii. 3.3V: Sensores

b. Corriente. El componente debe cumplir unos requisitos mínimos de potencia. Se realiza una prueba sobre la batería para asegurar que puede proveer dicho mínimo, con un procedimiento descrito más adelante

2. **Protección: BMS**

   Un BMS debe proveer protección contra picos de tensión para preservar la integridad de las celdas de la batería y evitar daños en los componentes electrónicos conectados, garantizando así la seguridad y la fiabilidad del sistema.

   A su vez, el BMS debe asegurar un equilibrado de la carga entre las celdas del componente, tanto en el proceso de carga como el de descarga.

3. **Monitorizar: Arduino y Sensor Amperaje**

   Capacidad de medir tensiones en 4 puntos independientes del BMS y corriente en 1. De esta forma, se debe poder monitorizar el consumo total del sistema, así como los procesos de equilibrado de carga realizados por el BMS.

Diseño

Componentes

Una vez planteado un boceto a grandes rasgos del sistema, se procede a concretar los componentes exactos a implementar. A grandes rasgos, se requiere de los siguientes componentes:

- Celdas de batería
- BMS
- Sensores de tensión
- Sensores de corriente
- Transformador de nivel

Para la mayoría de estos componentes se cuenta con una serie de opciones existentes que solventan la funcionalidad requerida.

- **Batería**: se plantea reutilizar unidades libres provenientes de otros sistemas. Se dispone de 6 unidades, posibilitando la configuración deseada.
- **BMS**: se cuenta con el *BMS 3S40A 12.6V Rev2.3*. A pesar de no disponerse de información excesivamente detallada de este componente, presenta un ajuste perfecto con los requisitos planteados, soportando 3S entre 12.6 y 0V.
- **Sensores**: Se plantean inicialmente varios componentes posibles:
  - **Sensor ACS712**: Es un sensor de corriente de efecto Hall. Presenta características muy positivas como ser económico, tener un tiempo de respuesta de pocos microsegundos y un error acotado del 1.5%. La salida es una señal analógica en tensión, directamente proporcional a la corriente.
  - **Sensor INA3221**: Es un monitor de corriente y voltaje de 3 canales con interfaz I2C y SMBus. Su rango de tensiones es de 0 a 26V y tiene un error acotado del 0.25%.

- **Sensor INA226**: Es un monitor de corriente y potencia con una interfaz compatible con I2C o SMBus. Ofrece una precisión de 16 bits y puede medir corriente, tensión y potencia en aplicaciones de hasta 36 V.
- **Arduino Nano**: Es una placa microcontroladora compacta y versátil, ideal para proyectos electrónicos que requieren un tamaño reducido y bajo consumo de energía. Utiliza el microcontrolador ATmega328, el cual tiene un convertidor analógico a digital (ADC) de 10 bits con 8 canales. Esto significa que puedes leer hasta 8 señales analógicas, lo cual puede ser muy conveniente en el desarrollo de este componente.

- **Transformador de nivel**: Este componente debe transformar los niveles de voltaje provistos por la batería a los que sean convenientes de usar en el sistema. En este caso, la batería provee 12.6V que son útiles únicamente para los motores. Para la mayoría de componentes es necesario contar con una alimentación a 5V de tensión. Este componente se conoce comúnmente como "BUCK" o step down converter. Se emplea el modelo LM2596 DC-DC, que permite ajustar de forma manual la tensión de salida.

Diseño lógico

El diseño lógico del componente de la batería pasa por diferentes iteraciones, en función de la efectividad de los componentes en realizar cada tarea. El diseño a grandes rasgos es siempre el mismo. Consiste en una batería de 6 celdas 2p3s conectada con un BMS entre cada par de celdas. En la salida del BMS se encuentra el resto del sistema, incluyendo conexiones para la carga de la batería y la alimentación a 12.6V. A esta tensión se alimentan los motores y el transformador de nivel, que a su vez alimenta a 5V al resto de componentes del sistema. A esta configuración se le añaden sensores para medir la tensión entre celdas y el consumo de corriente total.

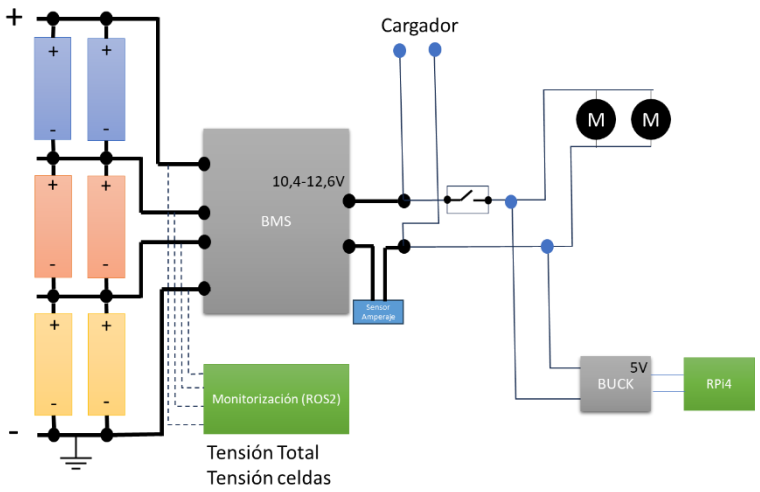


Figura 23 Esquema lógico general

Las diferentes iteraciones del diseño vienen dadas por diferentes estrategias a tomar a la hora de implementar los sensores de monitorización concretos. Cualquier combinación con los sensores disponibles es inicialmente considerada. Los componentes disponibles son los sensores INA3221, INA226 y Arduino Nano.

- **Arduino**: Implementar el componente con Arduino presenta la ventaja de que cuenta con suficientes canales ADC como para cubrir todas las mediciones necesarias para el componente de la batería. Se puede combinar con un sensor INA para la medición de corriente y medir toda las parametrizaciones requeridas. Presenta una interfaz física muy contenida (I2C, Power) y un diseño sencillo. Sin embargo, tienen el gran inconveniente de requerir un protocolo de comunicación con el controlador principal (Raspberry) que debe ser implementado. Además, los ADC son del rango 0-5V, por lo que se requiere de un circuito adicional que cuente con dos divisores de tensión de factor ½ y 1/3 y su posterior calibración.

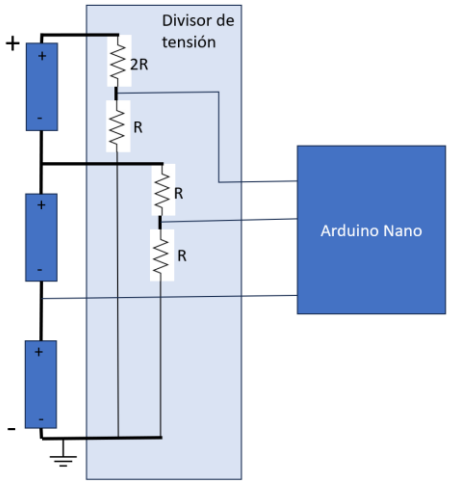


Figura 24 Esquema de divisores de tensión para monitorización con Arduino Nano

- **2 INAs**: Se puede emplear el INA3221 para medir 3 canales de tensión y el INA226 para medir un canal de corriente. Así, se cumplen los requisitos del sistema manteniendo una interfaz física sencilla y requiriendo implementar únicamente los drivers de los sensores sobre I2C. En la figura de debajo se puede apreciar el diseño empleando 2 INA3221 en esta configuración.

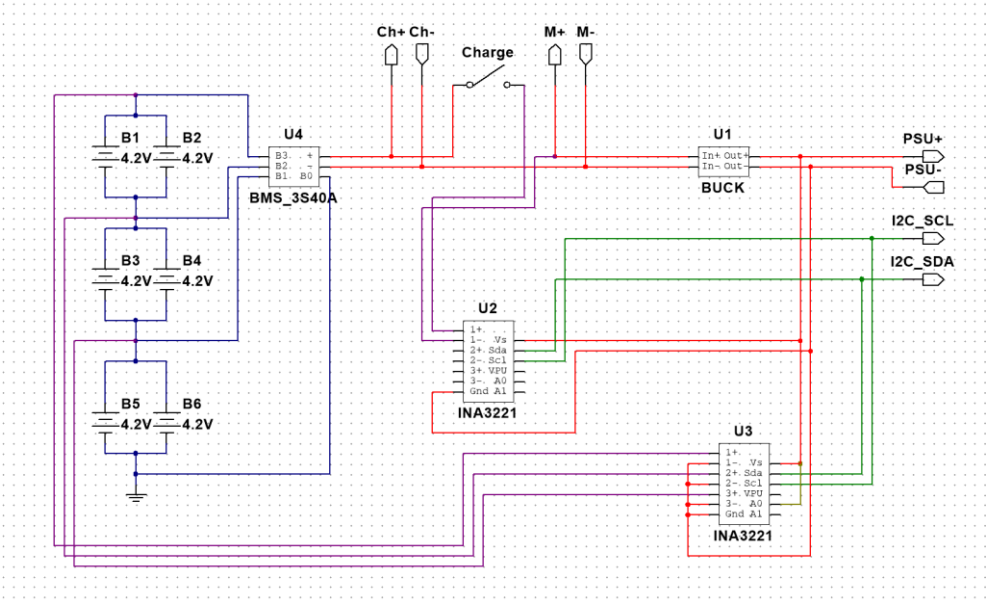


Figura 25 Batería sensorizada con 2 INAs

- **1 INA**: Se puede emplear el INA3221 con sus 3 canales para monitorizarlos de forma simultánea, uno de ellos midiendo tanto tensión como corriente. De esta forma se obtiene una configuración sencilla con un único sensor. Con un montaje adecuado es teóricamente correcto, sin embargo presenta el inconveniente de poder sufrir interferencia mutua y bajada de precisión en este diseño. Se descarta debido a dificultades en el correcto funcionamiento del canal de doble funcionalidad del INA3221.

De esta forma se concluye el diseño de la batería con una configuración de sensores de 2 INAs empleando el INA226 para medir corriente y el INA3221 para medir las tensiones.

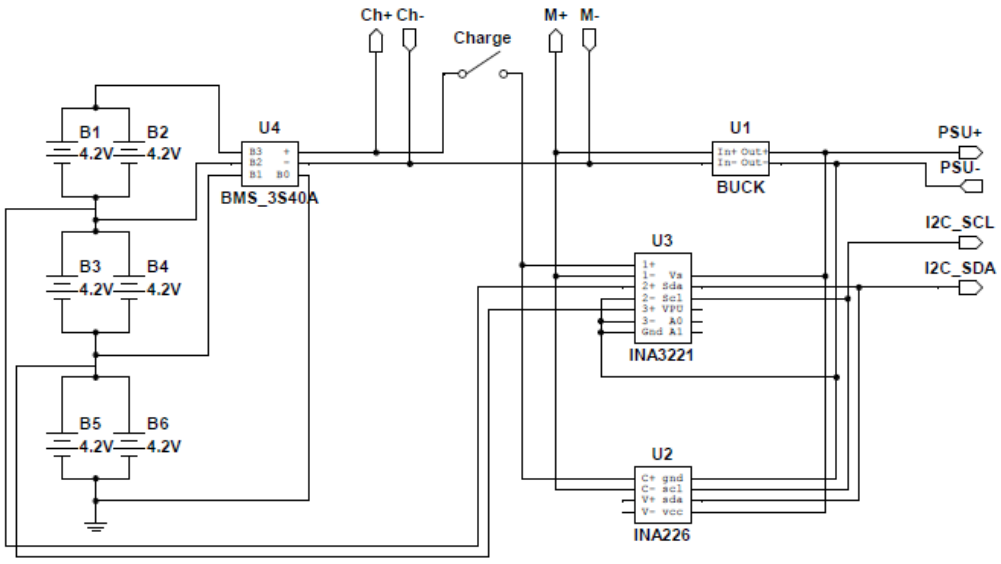


Figura 26 Esquemático definitivo de sensorización de batería

Diseño físico

El objetivo del diseño físico es la agrupación de componentes lógicos de la forma que se van a instalar en la implementación física. De esta forma, se pueden deducir las interfaces requeridas para la construcción del componente de forma rápida.

Para el diseño lógico escogido finalmente el diseño físico es trivial. Únicamente se agrupan las celdas de batería y el BMS en un único componente. En la figura de debajo se muestra la agrupación de los componentes, indicando de forma simplificada las interfaces de alimentación.

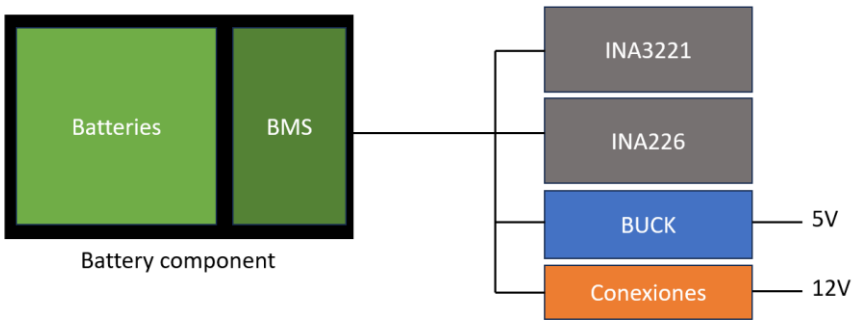


Figura 27 Diseño físico general del componente batería

Se agrega un componente para instalar la placa del INA3221. Se trata de otra placa que recibe la mayor parte de conexiones directas de la batería y las distribuye. Permite la monitorización de la batería así como la distribución de la alimentación. Contiene clemas, conectores Dupont y pistas. Conecta con el transformador de nivel y una placa de alimentación de 12.6V.

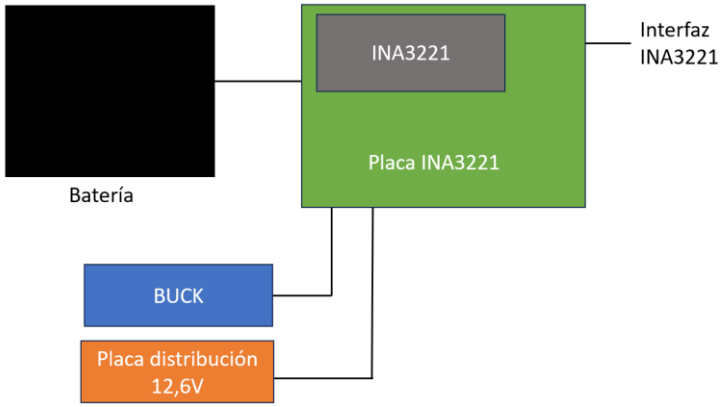


Figura 28 Placa del INA3221

De esta forma se deduce la interfaz física exacta requerida entre cada par de componentes físicos. Esta lista de conexiones resulta muy conveniente en la posterior etapa de construcción.

Construcción e implementación

Construcción física

Los componentes disponibles para la construcción de la batería se aprecian en la figura de debajo, con una imagen de fuente propia a la izquierda mostrando celdas, BMS, INA321, Arduino, ACS712, y BUCK; y el INA226 a la derecha, completando la lista de componentes previamente indicada.

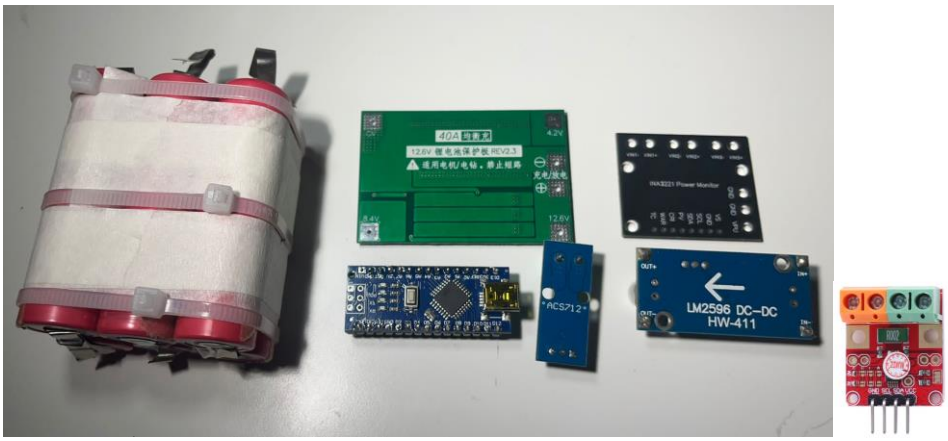


Figura 29 Componentes de la batería

En primer lugar, se construye el componente que junta las celdas de batería con el BMS. Para ello se juntan las celdas en una distribución 2*3 y se realizan las conexiones en la configuración 2p3s. Se añade el BMS y se conecta adecuadamente. Se utilizan cables gruesos de potencia, soldador, bridas, cinta y pegamento termofusible para construirlo. Se afina el acabado empleando malla cubre cables.

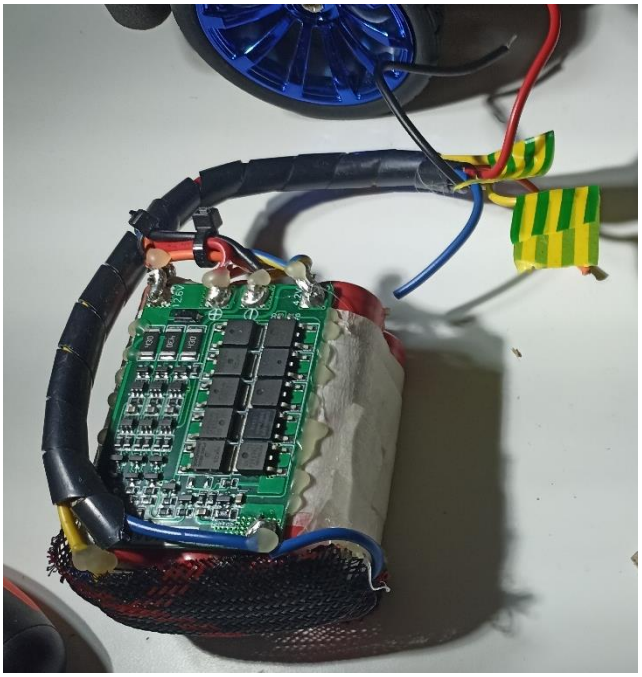


Figura 30 Foto de la batería: Celdas + BMS + conexiones

El resultado es un componente físico con los cables de todas las interfaces que implementa saliendo del mismo. En la figura superior se puede ver el manojo de cables tanto para distribución de alimentación como de input para los sensores de la batería.

A continuación, se establece en el chasis del robot la posición que ocupa la batería. Se decide ubicarla lo más centrada posible debido a que es el componente a instalar con más masa, y su posición puede tener repercusiones a la hora de maniobrar el coche a en movimiento. Con una distribución de peso adecuada se consigue mayor estabilidad, garantizando una mejora en la tracción y rendimiento en las curvas debido a menor inercia rotacional en el sistema.

Como procedimiento adicional de la fase de construcción se establece el soldado de una serie de pines que configuran la dirección I2C o el modo de funcionamiento de los sensores. Para ello se suelda o se elimina un punto de soldadura entre puntos habilitados en la placa. En el caso del INA226 se debe configurar la dirección del I2C a 0x44 eliminando un punto de soldadura en pin A1, así como configurarlo para medir en corriente agregando un punto entre el pin U y Vcc.

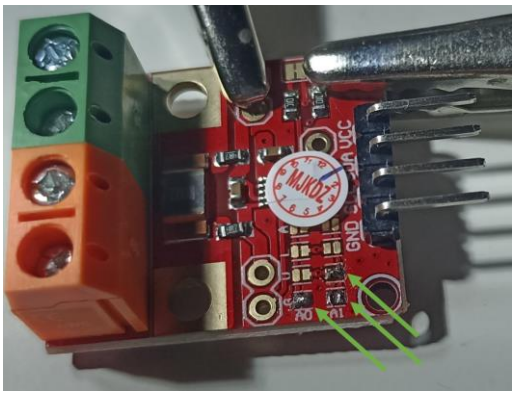


Figura 31 Puntos de soldadura sensor INA226

Para el INA3221 resulta ser suficiente con conectar el pin A0 a Vs del sensor sin necesidad de soldar, obteniendo la dirección 0x41.

Pruebas

Se realizan pruebas durante el transcurso del diseño de este componente.

- **Prueba preliminar de potencia**:
  El objetivo es asegurarse en una fase temprana del desarrollo de que la potencia provista por el componente cumple los requisitos. Para ello se calcula un valor aproximado de consumo del sistema completo. Este valor se obtiene de la suma de los componentes de mayor consumo del sistema en condiciones nominales. Se acota a P ~= 230W. Debido a que el consumo máximo aproximado de los motores es de 100W, la Raspberry 10W y el resto de los componentes presentan un consumo notablemente menor.
  La prueba se lleva a cabo empleando 4 resistencias de 50W conectadas en paralelo directamente a la batería y midiendo el consumo de potencia con un polímetro. De esta forma se asegura que el componente puede proveer al menos 200W de forma estable.

Figura 32 Resistencias de 50W

- **Prueba de los INA's**:
  Los sensores de tensión y corriente deben ser probados de forma aislada para comprobar que implementan la funcionalidad correspondiente de forma efectiva. Para ello se realizan pruebas de medición en tensión y en corriente con los sensores INA3221 e INA226. Las pruebas resultan correctas para mediciones en tensión en los 3 canales del INA3221 y corriente del INA226. Para la medición de corriente con el INA3221 no se obtienen resultados satisfactorios.

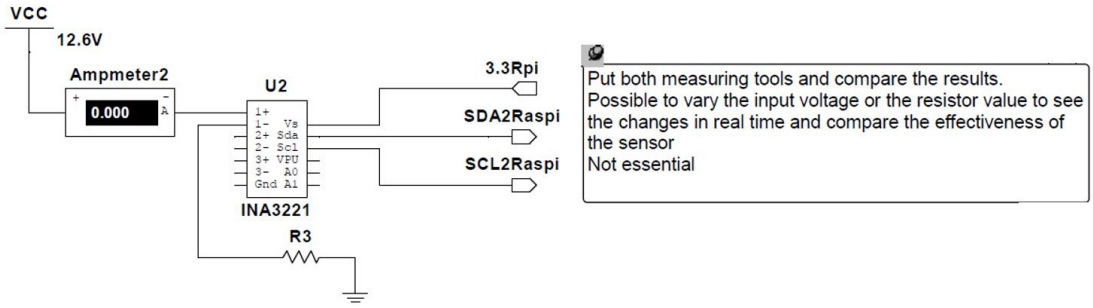


Figura 33 Prueba de medición de corriente con el INA3221

Instalación física del componente

El componente principal se instala centrado en el piso de abajo del chasis tal y como se especifica anteriormente, el resto de componentes se instalan de la forma más conveniente posible según el conexionado de los mismos. Dicha distribución contempla la batería situada en el centro y los sensores y placas distribuidos alrededor en la longitud del chasis.

Se agregan 2 componentes físicos.

- **Placa de alimentación 12.6V**: Consiste en una pequeña placa que conecta directamente con el componente de la batería y alimenta los componentes a esta tensión. Para ello cuenta con unas clemas que facilitan la conexión con los motores.

- **Medidor de carga**: Se añade un medidor de carga conectado directamente a la batería. Este soporta las tensiones de la batería y permite una lectura del estado de la batería rápida per imprecisa simplemente mirando el robot.

La distribución de alimentación a 5V se realiza en otra placa de distribución en el piso superior del chasis. Esta segunda placa distribuye 5V a todos los componentes del robot que lo requieren excepto la Raspberry, que se alimenta directamente del BUCK con un conector USB-C.

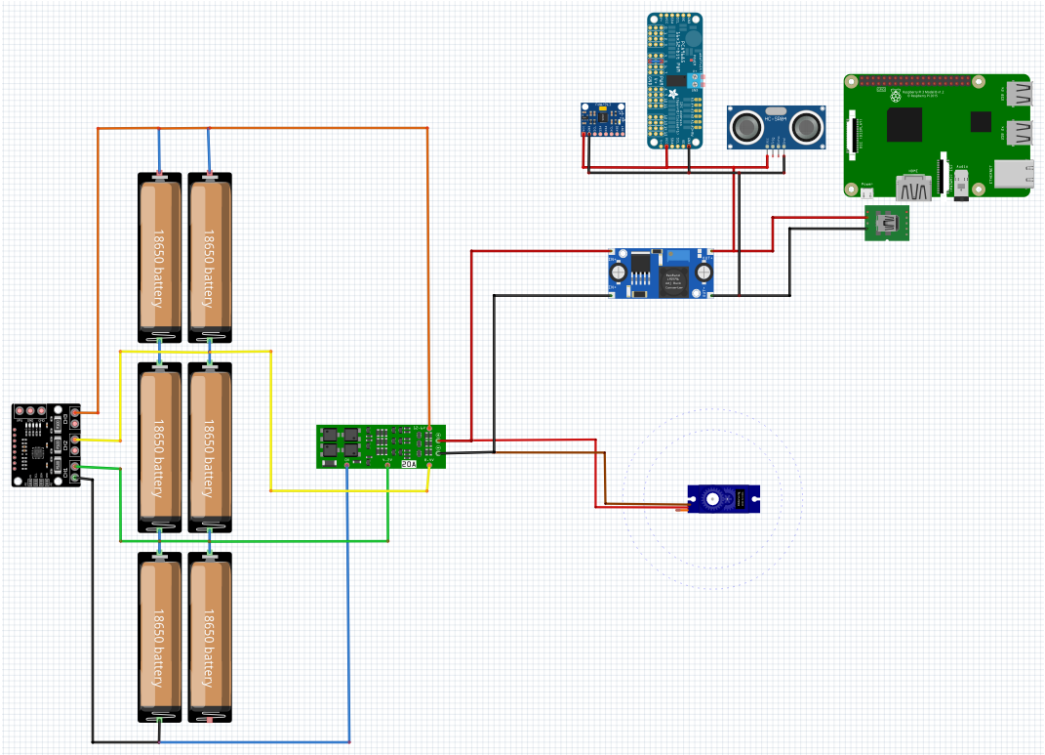


Figura 34 Componente batería y conexionado con el resto de componentes

Finalmente se concluye la construcción de este componente con la instalación de una clema de potencia para la carga de la batería y un interruptor de encendido.

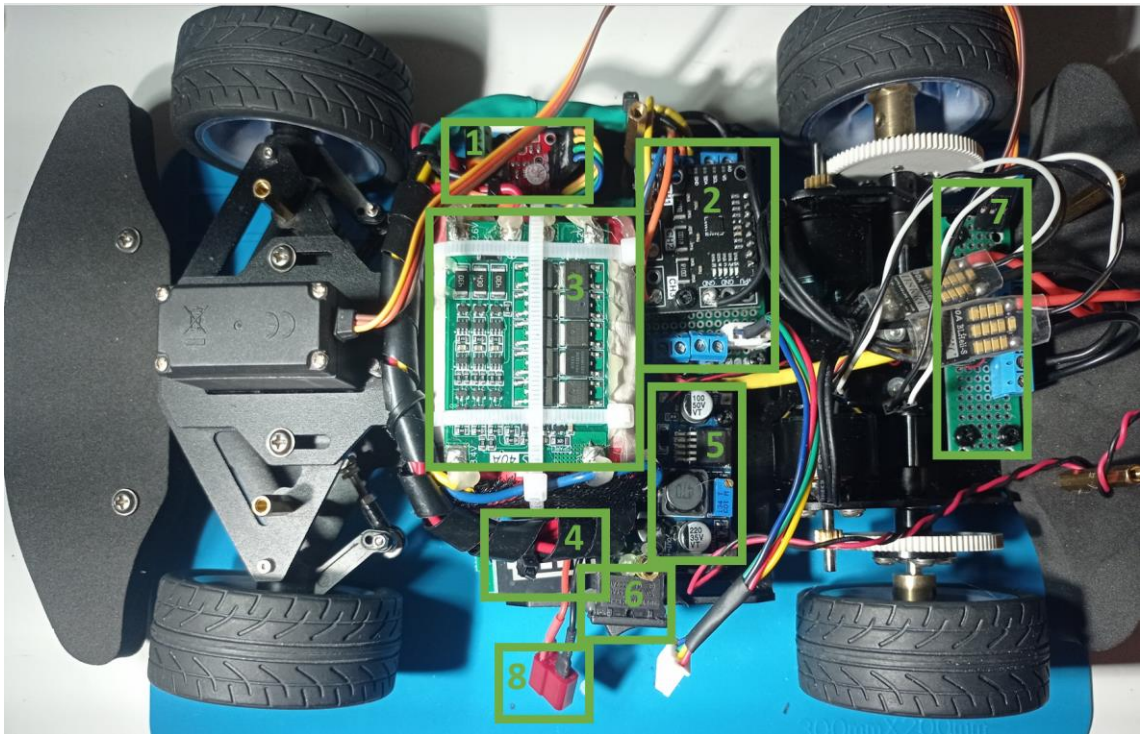


Figura 35: Componente batería instalado en el piso inferior del chasis

Id. Componente según Fig. superior	Nombre componente
1	INA226
2	INA 3221
3	BMS
4	Medidor de carga
5	BUCK o transformador de nivel
6	Interruptor de encendido
7	Placa distribuidora 12.6V
8	Clema de carga

Tabla 7 Componente batería instalados

Diseño y fabricación de piezas a medida

Diseño y Fabricación de Piezas a Medida

Este apartado describe el proceso de diseño y fabricación de piezas personalizadas para el coche robot, detallando los requisitos, el procedimiento de diseño, las herramientas utilizadas y los problemas encontrados. Las piezas se diseñan específicamente para optimizar la funcionalidad y estética del coche robot, pretendiendo asegurar una integración óptima entre los componentes electrónicos y mecánicos.

Piezas de Sensores

Definición de Requisitos

Se establecieron los siguientes requisitos para las piezas que alojan los sensores de distancia:

- Placeholder para Sensores: Debe haber espacio para albergar tres sensores HC-SR04 y un sensor KY032.
- Separación de Sensores: Los tres sensores HCSR04 deben estar separados por un ángulo de 30º para garantizar cobertura completa del frontal.
- Incorporación de Parachoques: La estructura debe incluir un parachoques para proteger el coche de colisiones.
- Protección de Sensores: Los sensores deben estar adecuadamente protegidos contra impactos y elementos ambientales.
- Ubicación del KY032: El sensor KY032 debe estar ubicado cerca del borde del parachoques pero sin sobresalir, permitiendo una detección precisa de objetos a muy cortas distancias.

También se establecen una serie de requisitos para las piezas que ejercen la función de chasis:

- Interferencia Mínima: No debe interferir con los demás componentes del robot, permitiendo una instalación y mantenimiento sencillos.
- Protección de Componentes: Debe proteger los componentes electrónicos y mecánicos contra daños físicos y condiciones ambientales.
- Estética: El diseño debe mantener una línea estética que contribuya a un acabado lo más profesional posible del coche robot.

Procedimiento del Diseño

El diseño de las piezas sigue un procedimiento que pretende maximizar el cumplimiento de los requisitos.

- Medición de Componentes:
  Se toman medidas precisas de todos los componentes relevantes para garantizar que las piezas diseñadas se ajustan tanto a los sensores como al chasis del coche.
- Importancia de los esbozos y bocetos a mano alzada:

Se realizan bocetos a mano alzada para visualizar y refinar las ideas de diseño antes de pasar al modelado 3D. Estos bocetos sirven como base para los modelos, permitiendo un proceso de diseño más fluido.

- Cumplimiento de Requisitos:
  Durante todo el proceso de diseño, se prioriza el cumplimiento de los requisitos establecidos, verificando iterativamente las dimensiones y funcionalidades necesarias.
- Diseño por piezas:
  Se optó por diseñar las piezas en partes distintas:
  - Pieza de los sensores.
    - Pieza Inferior: Funciona como soporte para el sensor KY032 y los faros, además de actuar como parachoques. Proporciona una base sólida y protege los componentes delanteros del coche.
    - Pieza Superior: Soporta los tres sensores HCSR04 en la posición adecuada y se integra con la pieza inferior.

  Esta decisión se tomó para facilitar el diseño y la impresión, así como para permitir un desarrollo incremental modular, donde la modificación de un componente de la pieza no afecte a la otra.

Procedimientos aplicables para Diseño 3D

Debido a la escasa experiencia del equipo en este tipo de tareas se realiza un breve estudio de los procedimientos más comunes::

- Modelado Paramétrico: Utiliza parámetros para definir las dimensiones y las relaciones entre diferentes partes del modelo. Este enfoque es útil para diseños que requieren ajustes precisos y repetidos.
- Modelado Directo: Permite modificar directamente la geometría del modelo sin necesidad de parámetros predefinidos. Es ideal para diseños más orgánicos y menos estructurados.
- Diseño Generativo: Utiliza algoritmos para generar múltiples iteraciones de un diseño basado en parámetros y restricciones definidos por el usuario. Este método es útil para optimizar el diseño según criterios específicos como peso, resistencia y materiales.

El procedimiento seguido en el proyecto es similar al **modelado paramétrico**, que utilizando herramientas como Fusion 360 permite definir y ajustar parámetros específicos para cumplir con los requisitos del diseño. Esta metodología asegura precisión y facilita la realización de cambios durante el proceso de diseño.

Piezas Diseñadas

Pieza Sensor:

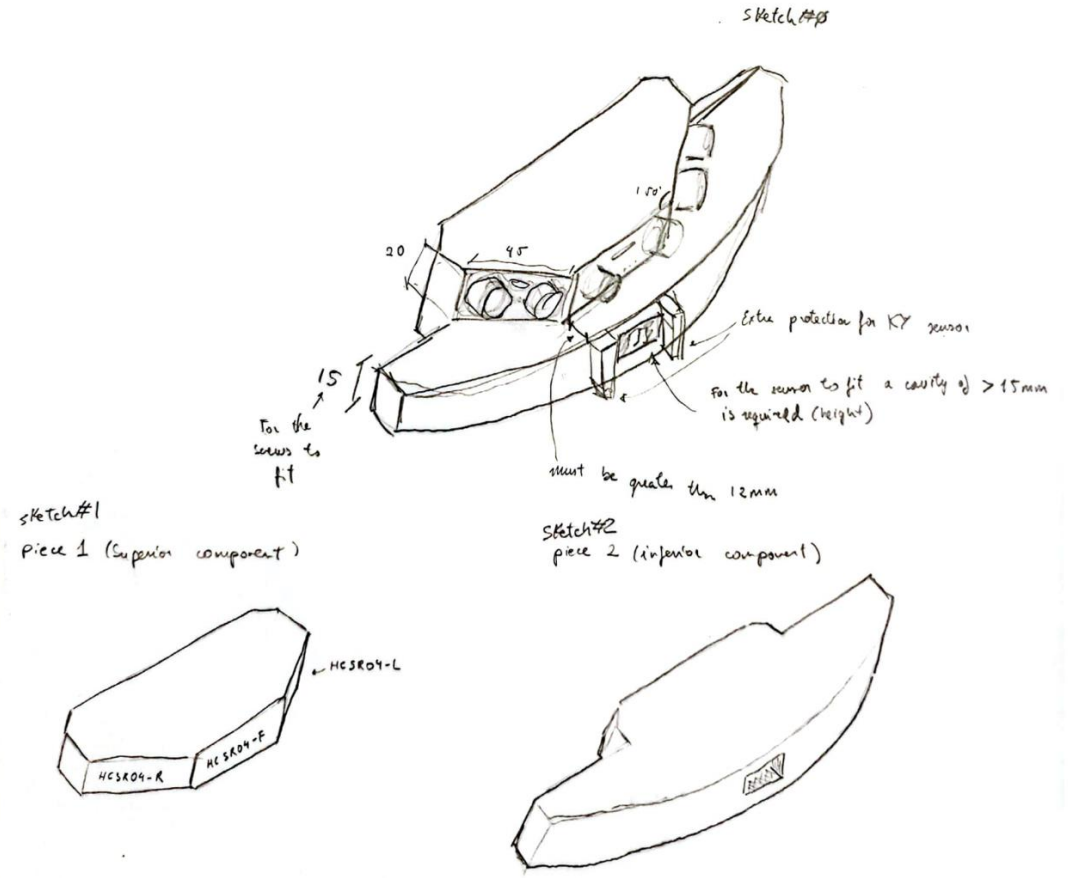


Figura 36 Boceto borrador pieza sensor

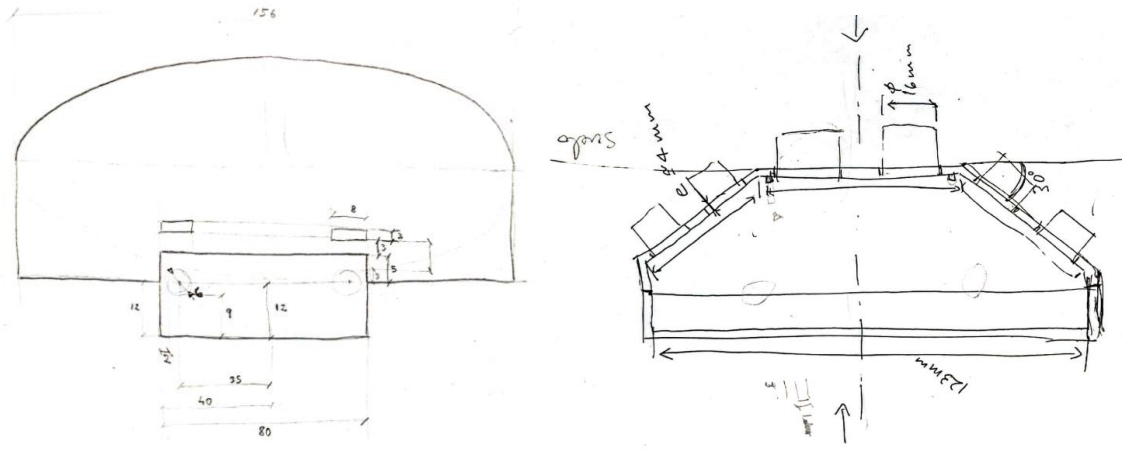


Figura 37 Bocetos componentes inferior (izq) y superior (der) de la pieza sensor

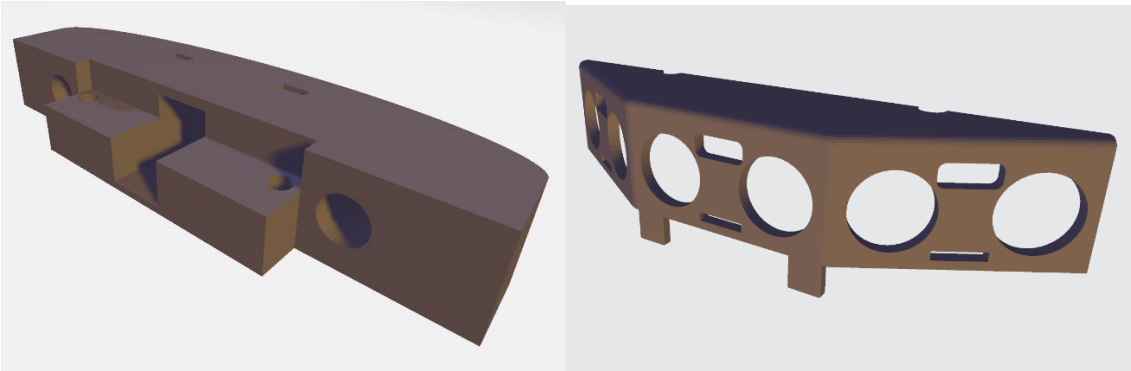


Figura 38 Render pieza sensor componente inferior (izq) y superior (der)

Pieza del Chasis

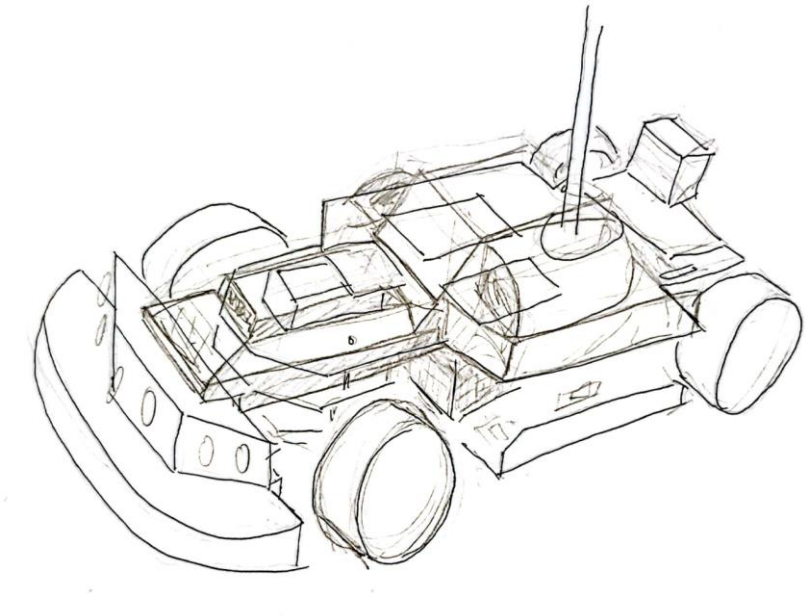


Figura 39 Boceto conceptual pieza chasis

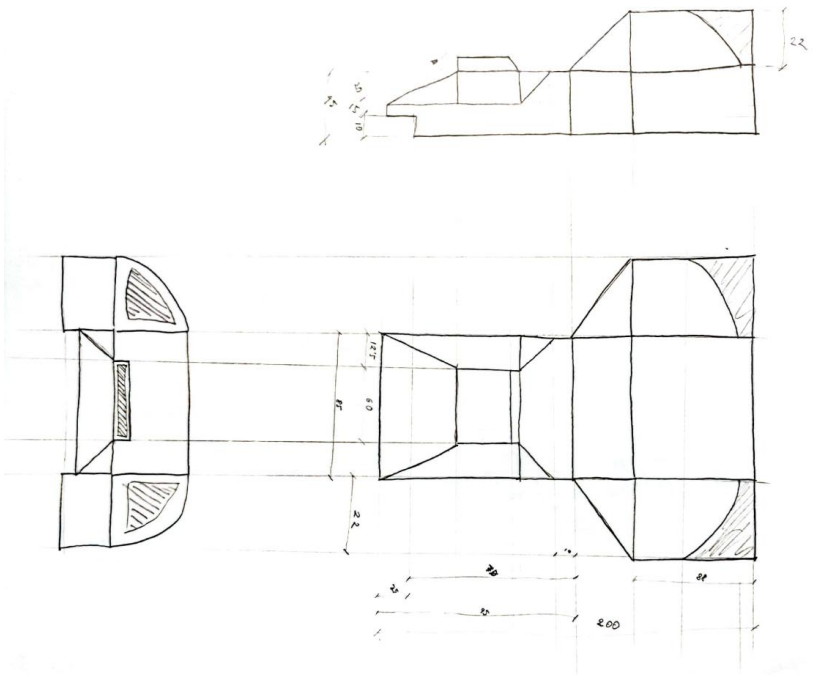


Figura 40 Vistas ortográficas boceto pieza chasis

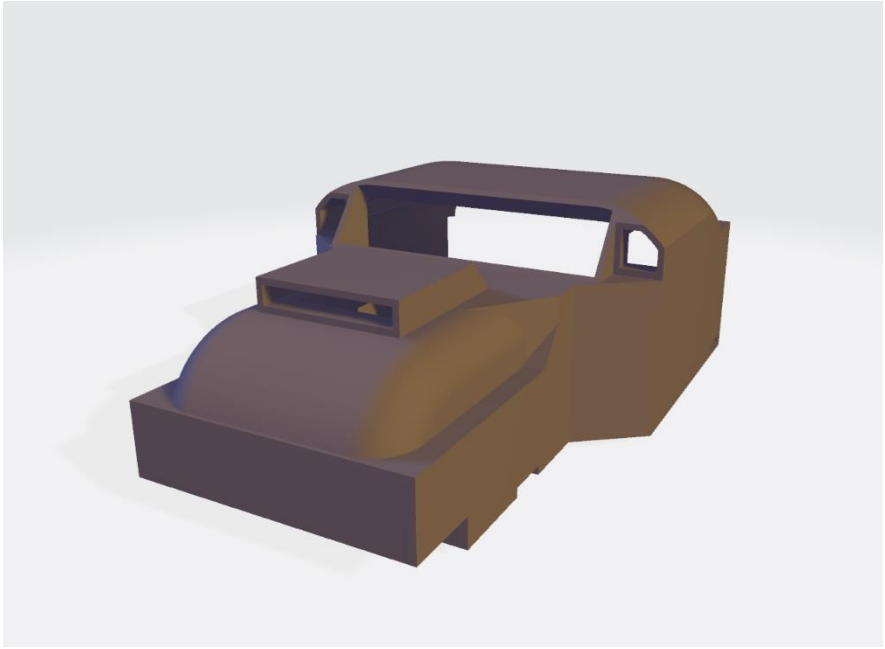


Figura 41 Render pieza chasis

Herramientas Utilizadas

El proceso de diseño y fabricación de las piezas se realiza utilizando las siguientes herramientas:

Fusion 360

Software de diseño CAD para modelado 3D paramétrico, permitiendo precisión y flexibilidad en el diseño. Cuenta con licencia académica para estudiantes de la UPM.

Software de Corte (Slicer)

'Cura' Utilizado para preparar los modelos 3D para la impresión, ajustando parámetros como la densidad del relleno, las temperaturas de impresión o grosor de las capas.

Impresora 3D

Impresora 3D con filamento PLA.

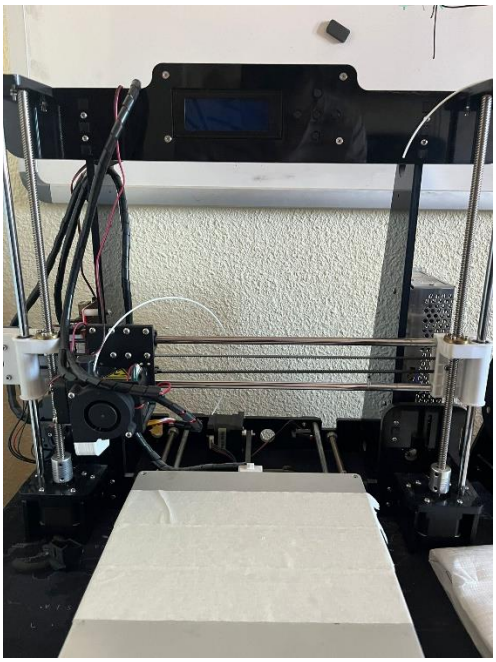


Figura 42: Impresora 3D Anet A8

Construcción

De modelo a pieza

Una vez diseñada la pieza se obtiene un modelo correspondiente con el SW de diseño empleado. En el caso de Fusion 360, se obtienen ficheros (.f3d), que se exportan a modelos 3d genéricos (.stl). Estos modelos se pasan al slicer, y una vez configurado se procede a imprimir. Se debe prestar atención durante el proceso de impresión debido a que puede haber errores que requieran intervención humana.

Procesado de la pieza

Una vez se obtiene la pieza física se extrae de la impresora y se añaden los retoques necesarios:

- Lijado de superficies imperfectas.
- Eliminado de soportes, a veces necesarios para la impresión.
- Retocado manual de las discrepancias entre el modelo y la pieza que puedan surgir.

Integración en el robot

Las piezas se incorporan en el robot mediante la interfaz diseñada para ello. En el proceso, se puede requerir instalar componentes adicionales en la pieza antes de instalarla en el chasis del coche.

- **Pieza sensor superior**: Se instalan los sensores HC antes de incorporarla.
- **Pieza sensor inferior:** Se instala el sensor KY y los faros, además de una espuma amortiguadora para absorber la energía cinética de posibles impactos frontales.
- **Pieza chasis**: Se puede instalar directamente sobre el chasis una vez se ordenan los componentes y cables previamente presentes en el mismo.

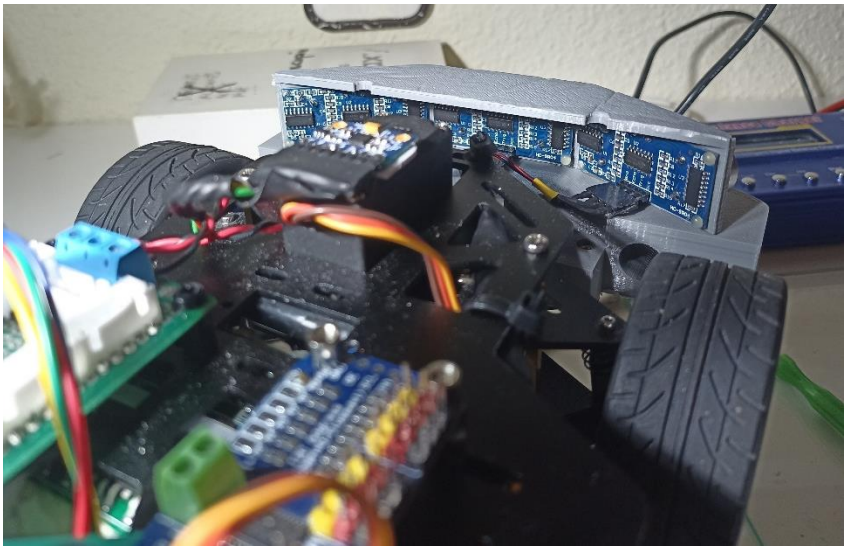


Figura 43 Integración de la pieza para sensores en el robot

Figura 44: Pieza Chasis 1

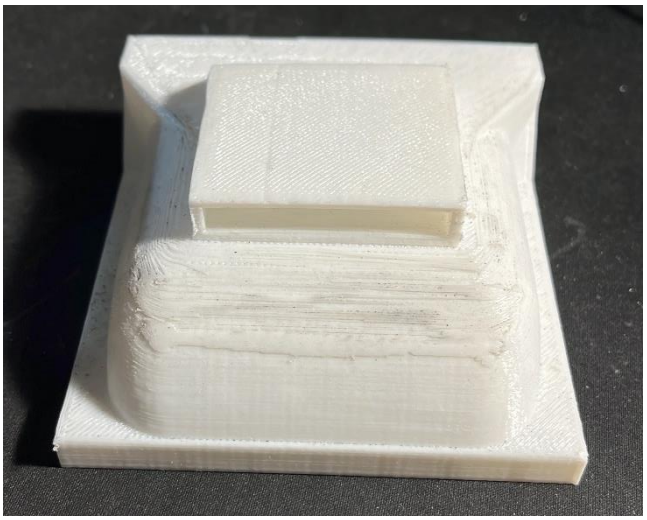


Figura 45: Pieza Chasis 2

Problemáticas Encontradas

Durante el proceso de diseño y fabricación, se identificaron y abordaron varias problemáticas:

Parametrización Incorrecta

La falta de una correcta parametrización de los diseños iniciales dificultó su edición, debido a una comprensión incompleta de las mejores prácticas de diseño paramétrico.

Errores en la Impresión

Se presentaron errores durante la impresión que no siempre pudieron ser identificados, afectando la calidad y funcionalidad de algunas de las piezas impresas.

Otros componentes

MPU6050

Sensor que combina un giróscopo de tres ejes y un acelerómetro con interfaz de comunicación I2C.

Este sensor es muy útil para controlar la posición del coche y poder detectar inclinaciones, golpes y movimiento en general, lo que podría llegar a ser crucial a la hora de implementar un algoritmo de conducción autónoma.

Figura 46: MPU6050

PCA9685

Esta placa es un controlador de PWM que ofrece 16 canales de salida independientes para PWM que son controlables a través de una interfaz de comunicación I2C.

En esta placa se conectan los dos motores ESC y el servomotor, de forma que a través de la comunicación I2C es posible controlar estos tres motores.

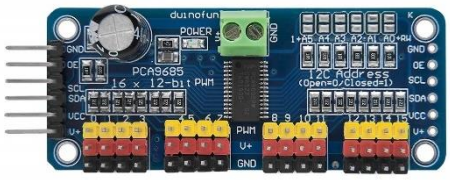


Figura 47: PCA9685

Joystick PS3

Para conectar el mando de la Play Station 3 a la Raspberry Pi 4 por bluetooth seguiremos este tutorial:

https://wiki.gentoo.org/wiki/Sony_DualShock

Antes de conectar el mando hay que configurar el bluetooth de la siguiente manera, editando el fichero de /etc/bluetooth/input.conf:

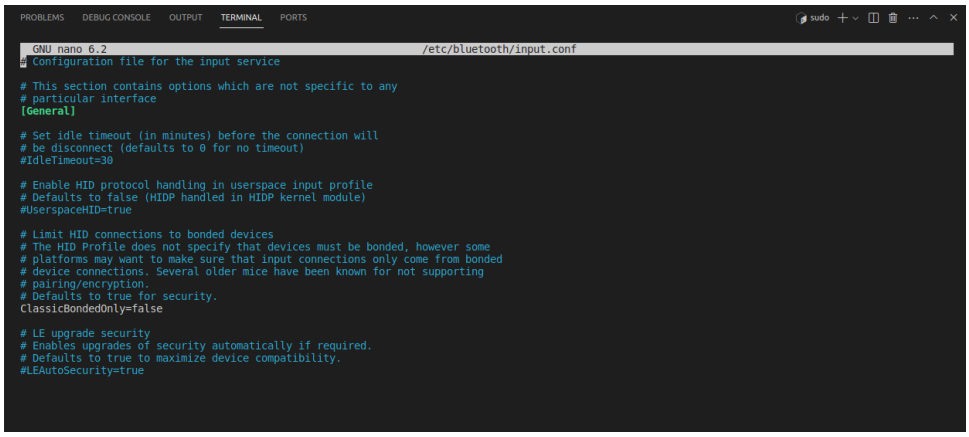
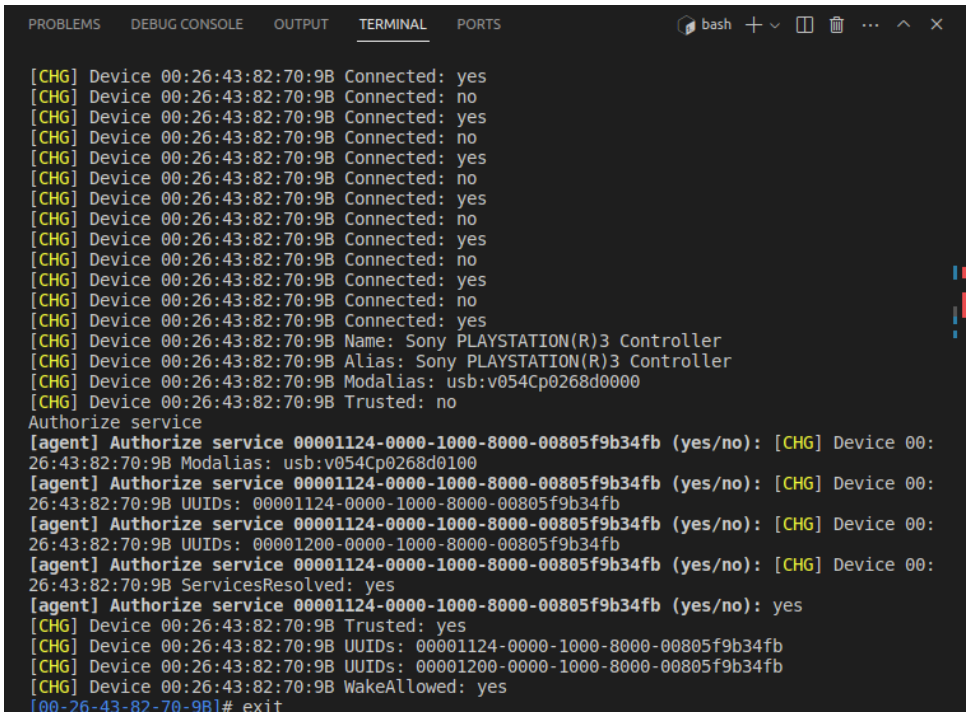


Figura 48: Configuración Bluetooth Raspberry

En resumen, para que funcione tiene que salirnos este mensaje de autorización. Después de muchos intentos, la forma de conseguirlo es simplemente instalando los paquetes de bluez (sudo apt install bluez-*) y con USB conectado te hace esa pregunta.



```
[CHG] Device 00:26:43:82:70:9B Connected: yes
[CHG] Device 00:26:43:82:70:9B Connected: no
[CHG] Device 00:26:43:82:70:9B Connected: yes
[CHG] Device 00:26:43:82:70:9B Connected: no
[CHG] Device 00:26:43:82:70:9B Connected: yes
[CHG] Device 00:26:43:82:70:9B Connected: no
[CHG] Device 00:26:43:82:70:9B Connected: yes
[CHG] Device 00:26:43:82:70:9B Connected: no
[CHG] Device 00:26:43:82:70:9B Connected: yes
[CHG] Device 00:26:43:82:70:9B Connected: no
[CHG] Device 00:26:43:82:70:9B Connected: yes
[CHG] Device 00:26:43:82:70:9B Connected: no
[CHG] Device 00:26:43:82:70:9B Connected: yes
[CHG] Device 00:26:43:82:70:9B Name: Sony PLAYSTATION(R)3 Controller
[CHG] Device 00:26:43:82:70:9B Alias: Sony PLAYSTATION(R)3 Controller
[CHG] Device 00:26:43:82:70:9B Modalias: usb:v054Cp0268d0000
[CHG] Device 00:26:43:82:70:9B Trusted: no
Authorize service
[agent] Authorize service 00001124-0000-1000-8000-00805f9b34fb (yes/no): [CHG] Device 00:
26:43:82:70:9B Modalias: usb:v054Cp0268d0100
[agent] Authorize service 00001124-0000-1000-8000-00805f9b34fb (yes/no): [CHG] Device 00:
26:43:82:70:9B UUIDs: 00001124-0000-1000-8000-00805f9b34fb
[agent] Authorize service 00001124-0000-1000-8000-00805f9b34fb (yes/no): [CHG] Device 00:
26:43:82:70:9B UUIDs: 00001200-0000-1000-8000-00805f9b34fb
[agent] Authorize service 00001124-0000-1000-8000-00805f9b34fb (yes/no): [CHG] Device 00:
26:43:82:70:9B ServicesResolved: yes
[agent] Authorize service 00001124-0000-1000-8000-00805f9b34fb (yes/no): yes
[CHG] Device 00:26:43:82:70:9B Trusted: yes
[CHG] Device 00:26:43:82:70:9B UUIDs: 00001124-0000-1000-8000-00805f9b34fb
[CHG] Device 00:26:43:82:70:9B UUIDs: 00001200-0000-1000-8000-00805f9b34fb
[CHG] Device 00:26:43:82:70:9B WakeAllowed: yes
[00-26-43-82-70-9B]# exit
```

Figura 49: Interfaz Bluetoothctl

Display de Batería 3S

Este display se conecta directamente a la salida del BMS y está configurado para medir el voltaje de la batería y poder ver a simple vista si la batería está cargada o descargada.

Figura 50: Display 3S

Software

En este apartado se explica todo lo relacionado con el Software del robot, desde el entorno de desarrollo hasta la creación del panel de control (objetivo final nº2 del proyecto) donde se visualizarán todos los datos obtenidos de los sensores.

Para los siguientes apartados, hay que tener en cuenta las siguientes consideraciones:

1. Todo el código se ejecutará sobre la Raspberry Pi 4
2. El lenguaje de programación principal es Python 3
3. Se utiliza Git para el control de versiones
4. El código se implementa sobre ROS2

Análisis y Diseño en ROS2

ROS2 (Robot Operating System 2) es un framework Open-Source que proporciona una serie de herramientas software y librerías que permite realizar aplicaciones robóticas desde simples robots móviles hasta sistemas robóticos complejos.

ROS2 tiene una arquitectura distribuida y modular, permitiendo que diferentes partes de un sistema robótico se comuniquen de manera eficiente. Utiliza el estándar DDS (Data Distribution Service) para la comunicación entre nodos, lo que garantiza una comunicación fiable y segura en sistemas distribuidos a gran escala.

Es importante destacar que ROS2 no es adecuado para sistemas de tiempo real, pero para este proyecto no es contemplado como un requisito obligatorio.

Conceptos básicos

Para realizar un diagrama que aplique a nuestro coche autónomo, primero debemos revisar los conceptos básicos de ROS2, que son los que aplican a este proyecto:

1. **Nodes**: Un nodo debe encargarse de una unidad funcional (p. Ej, un nodo para controlar los motores de las ruedas, para controlar un sensor láser...) Cada nodo envía y recibe datos a otros nodos a través de Topics, Services, Actions o Parameters.
2. **Topics**: Actúan como bus de comunicación entre nodos para intercambiar mensajes. Un nodo puede publicar datos a cualquier número de topics y puede estar suscrito a cualquier numero de topics.
3. **Services**: Mientras que los topics envían información de forma continuada, los servicios envían datos cuando un cliente se lo solicita.
4. **Parameters**: es la configuración de un nodo. Se pueden guardar enteros, comas flotantes, etc...
5. **Messages**: Los mensajes en ROS2 son estructuras de datos que se utilizan para el intercambio de información entre nodos. Pueden incluir tipos primitivos como enteros, flotantes y cadenas, además de estructuras más complejas como arreglos y anidaciones de otros mensajes. Cada mensaje en ROS2 está definido por un tipo que determina la estructura de los datos que contiene, lo que asegura que los nodos puedan interpretar los datos correctamente al recibirlos.

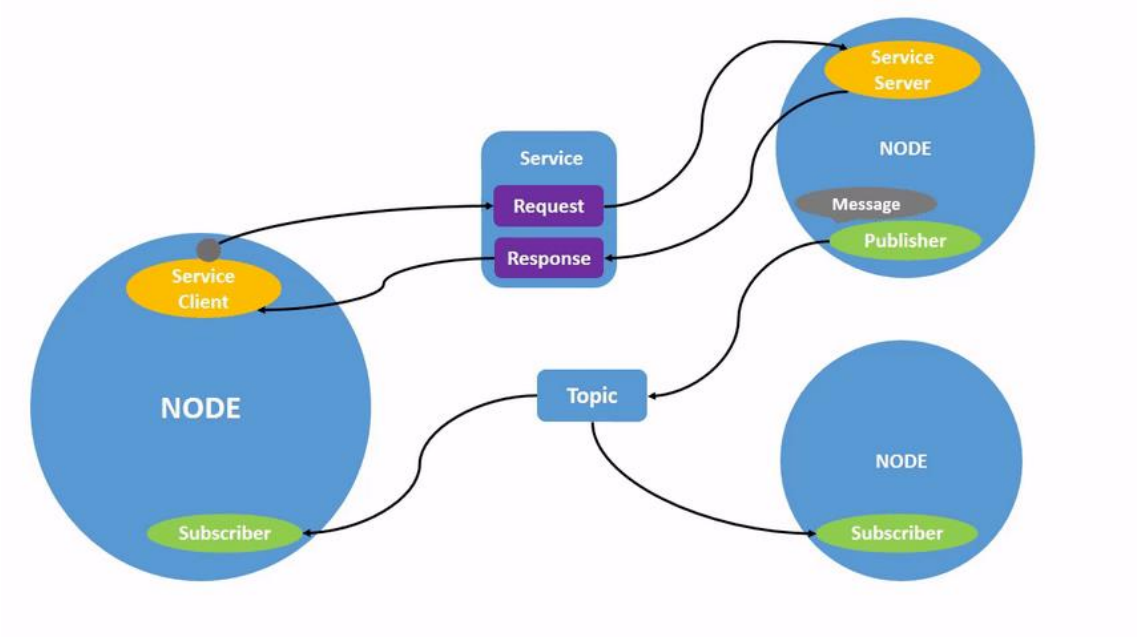


Figura 51: Esquema general ROS2

Diseño del Diagrama de Nodos

Para intercomunicar los distintos nodos y proporcionar un panel de control con todos los datos de los distintos sensores del coche se necesita definir un diagrama de nodos que soporte todos los datos que vengan de los sensores.

Es importante destacar que cada nodo se debe dedicar a una única tarea, por lo que en función de esto se definen los siguientes nodos con sus características:

Tabla 8: Características de los nodos ROS2

Nodo	Publica	Tipo Mensaje	Se suscribe
Cámara (camera_node)	/camera_publisher	Sensor_msgs.Image	X
Energy_node	/energy	Energy (personalizado)	X
Imu_node	/imu	Imu	X
Teleop_twist_joy	/joy	Joy	X
Distance_node	/ultrasound_data	Distance (personalizado)	X
Car_control_node	X	X	/joy

A continuación, se hace una breve descripción del funcionamiento de los nodos:

- **Camera_node**: captura imágenes de una cámara usando OpenCV y las publica en 'camera_image' a una tasa de 5 Hz (cada 0.2 segundos). Utiliza CvBridge para convertir las imágenes de OpenCV a un formato compatible con ROS2 (Image).

- **Energy_node**: publica información sobre la energía, incluyendo el voltaje de tres baterías y la corriente, en un tópico llamado 'energy' cada segundo. Utiliza dos bibliotecas, SDL_Pi_INA3221 y INA226Lib, para leer los valores de voltaje y corriente de los sensores correspondientes. Los valores se redondean a dos decimales antes de publicarlos.

- **Imu_node**: publica datos de aceleración y velocidad angular obtenidos de un sensor MPU6050 en un tópico llamado 'imu' a una tasa de 2 Hz (cada 0.5 segundos). Utiliza la biblioteca MPU6050 para interactuar con el sensor y publica los datos redondeados a dos decimales.

- **Teleop_Twist_Joy**: Se conecta al mando de PS3 y publica los datos de todos los botones y joysticks.

- **Distance_Node**: utiliza los tres sensores ultrasónicos HC-SR04 para medir distancias en las direcciones izquierda, derecha y centro, y un sensor de parada de emergencia (KY032). Publica los datos de distancia y el estado del sensor de emergencia en un tópico llamado 'ultrasound_data' a una frecuencia de 10 Hz.

- **Car_Control_Node**: Se encarga de contolar el robot en función de los datos interpretar los datos del mando PS3 publicados por el topic teleop_twist_joy.

En la página siguiente se encuentra el diagrama de nodos del proyecto.

Implementación de ROS2

Entorno de Desarrollo

se instala Ubuntu 22.04 LTS de 64 bits en la Raspberry Pi 4, utilizando el instalador oficial “Raspberry Pi Imager”. Se instala esta versión ya que a hasta la fecha de realización del proyecto esta versión es la última compatible con la versión Iron de ROS2.

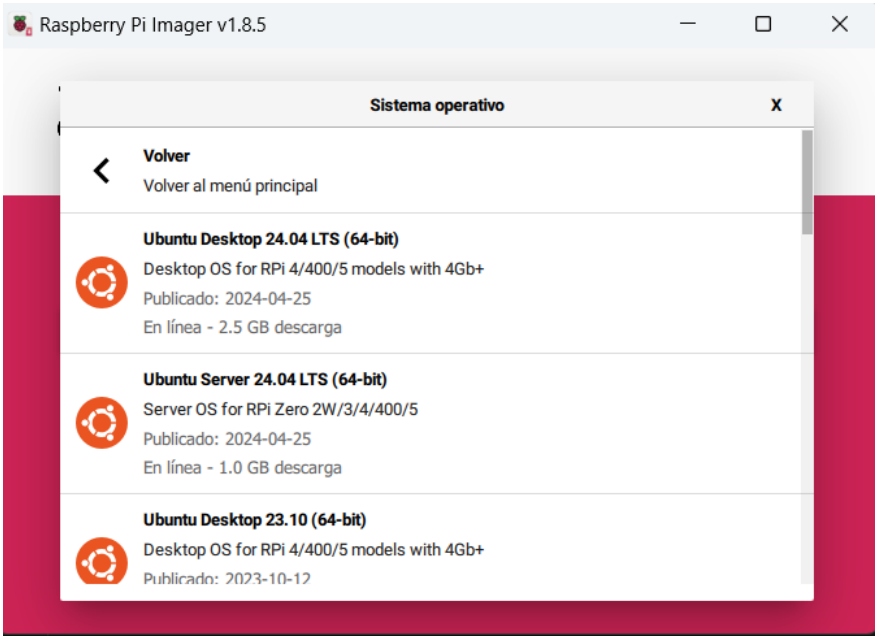


Figura 52: RPi Imager

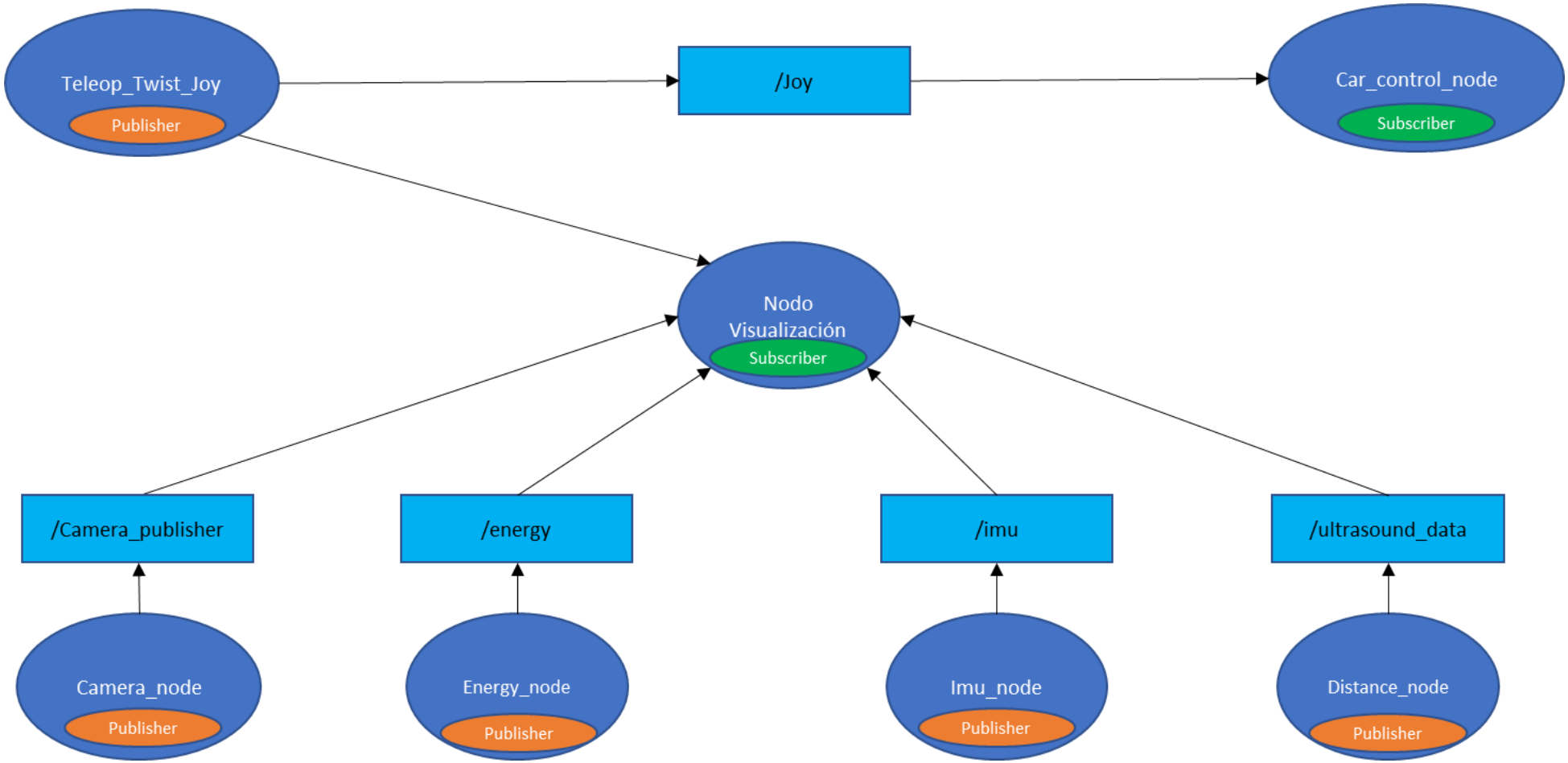


Figura 53: Diagrama de Nodos ROS2

Se debe configurar la red y habilitar el servicio SSH para que se pueda acceder posteriormente una vez quede instalado el sistema operativo. Para configurar estas opciones hay que pulsar la siguiente combinación: Ctrl + Shift + X.

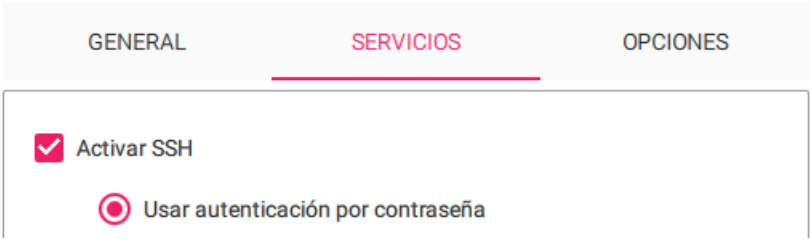


Figura 54: Configuración RPi Imager

Para desarrollar el código, se utiliza el IDE Visual Studio Code (VSCode), debido a la flexibilidad y a todas las extensiones que nos ofrece.

VSCode ofrece una extensión (Remote – SSH) para conectarse de forma remota por SSH a otros disposivos. En nuestro caso esto es imprescindible, ya que todo se desarrolla sobre la Raspberry Pi 4.

Según la configuración del sistema operativo, se crea el usuario “lab” con contraseña “lab”. Bastaría con insertar el comando “ssh lab@robocar.local”, a lo que nos pedirá la contraseña. Una vez introducida, ya estaremos dentro.

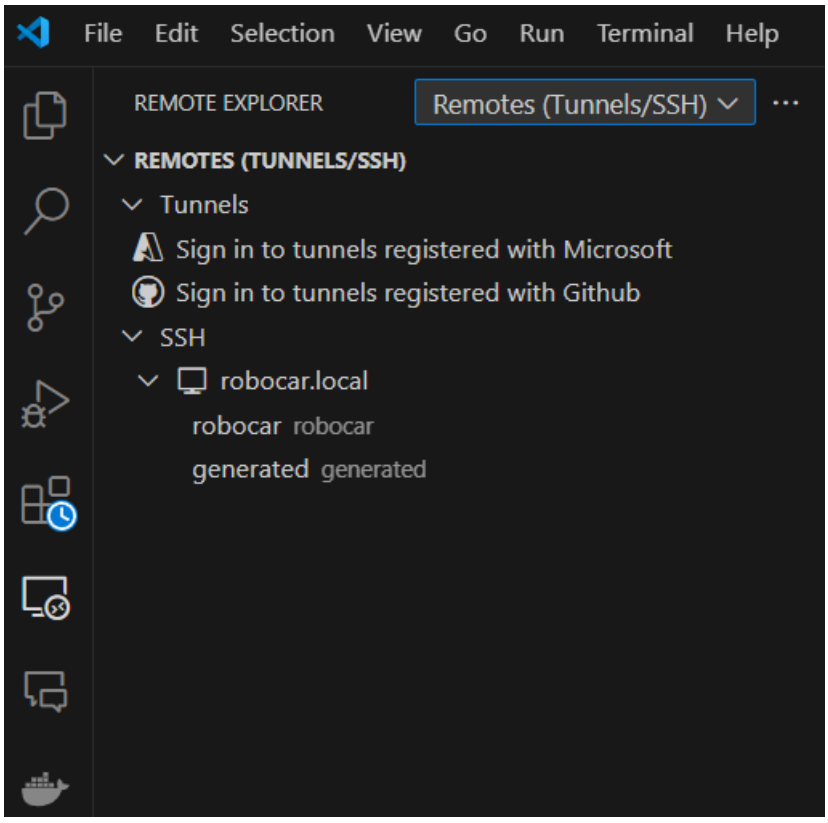


Figura 55: Conexión SSH en VSCode

Algunas extensiones útiles para nuestro proyecto y que se han instalado en VSCode son:

- ROS → Nos ayudará a la hora de programar en ROS2
- Pylance → Nos ayudará a programar en Python
- Github Copilot → Muy útil para ser más eficiente a la hora de programar

Todas estas extensiones se instalan en el servidor (RPi 4), para poder tenerlas en cualquier máquina cuando accedamos por SSH a la misma.

Adicionalmente se crea un entorno virtual de Python para que todas las dependencias de librerías usadas en este proyecto queden dentro del directorio .venv de la Raspberry.

Con esto ya está preparado el entorno de desarrollo, donde tendremos acceso al árbol de directorios y a la terminal del sistema.

Repositorio de Código (GitHub)

Para tener un control de versiones adecuado y la posibilidad de programar concurrentemente, todo el código está disponible en GitHub.

El código está subido en el repositorio de Git https://github.com/rubenhigorg/robocar, ubicado en el directorio /home/lab/robocar de la Raspberry.

La estructura general del repositorio es la siguiente:

Tabla 9: Estructura de Directorios de Robocar

Fichero/Directorio	Contenido
.venv	Entorno virtual de Python3 con las dependencias del proyecto
sixpair	Script para configuración del mando PS3
pruebas	Scripts para el benchmarking de los sensores
src	Contiene todo el código de los nodos de ROS2, estructurado en paquetes. Se describe más adelante
Launch.sh	Script para lanzar todos los nodos ROS2 del proyecto
Setup.sh	Script para inicializar todas las variables de entorno y comandos

Instalación de ROS2

Para instalar ROS2 en nuestra Raspberry Pi 4 se han seguido los pasos de la documentación oficial: https://docs.ros.org/en/iron/Installation/Ubuntu-Install-Debians.html. En nuestro caso hemos optado por instalar la versión *Iron*, que es la última versión estable en el momento en el que se empieza a usar para este proyecto.

En este punto tendríamos los comandos “ros2” y “colcon” disponibles, con lo que ya tendríamos la interfaz básica para ejecutar las funcionalidades principales de ROS2 y las herramientas fundamentales para el desarrollo de los nodos.

Desarrollo de nodos en ROS2

La estructura de ROS2 se divide en paquetes, que son las unidades básicas de organización. Los paquetes pueden contener nodos, bibliotecas, archivos de configuración, mensajes, servicios y acciones.

En este proyecto se desarrolla sobre tres paquetes ubicados en /robocar/src/:

1. **Messages_pkg**: Contiene mensajes personalizados para intercambiar mensajes entre los distintos nodos.

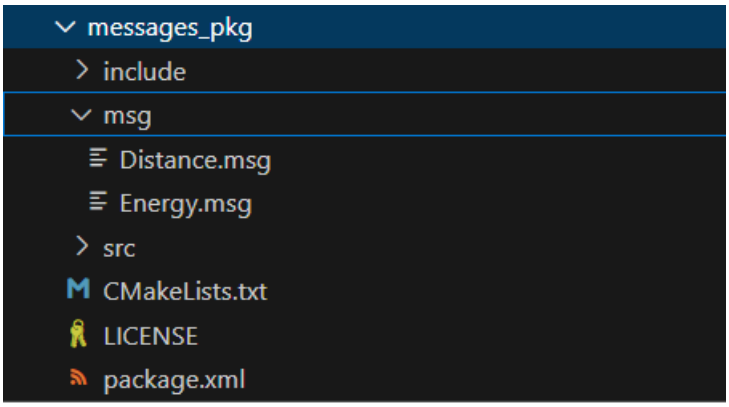


Figura 56: Paquete messages_pkg

2. **Robocar_pkg**: Contiene todos los nodos del proyecto
    - **Lib**: Directorio donde se almacenan las librerías de los distintos componentes usados por los nodos
    - **Robocar_pkg**: Contiene los ficheros Python con los nodos.
    - **Setup.py**: Aquí se indica de forma explícita los nodos que se pueden lanzar desde este paquete

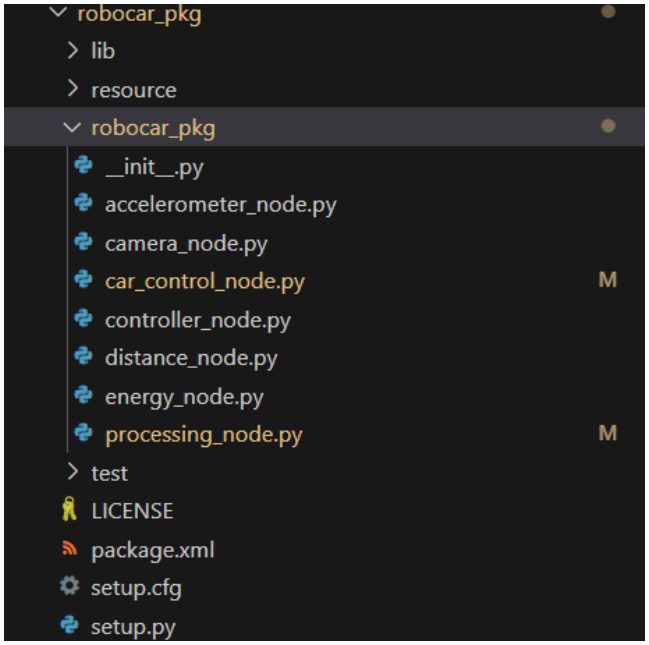


Figura 57: Paquete robocar_pkg

3. **Teleop_twist_joy**: Este paquete se utiliza para lanzar el nodo que publica los datos del mando de PS3. En el fichero .gitmodules se especifica la URL del repositorio del nodo (https://github.com/ros2/teleop_twist_joy.git), donde también se detalla la configuración del nodo.

Construcción y lanzamiento de nodos

Una vez implementado todo el código de los nodos hay que construirlos. Para ello hay que ubicarse dentro del directorio src del repositorio y lanzar el comando colcon build:

```
(.venv) lab@robocar:~/robocar$ cd src
(.venv) lab@robocar:~/robocar/src$ colcon build
Starting >>> messages_pkg
Starting >>> robocar_pkg
Starting >>> teleop_twist_joy
Finished <<< robocar_pkg [9.63s]
Finished <<< messages_pkg [21.1s]
[Processing: teleop_twist_joy]
Finished <<< teleop_twist_joy [1min 19s]

Summary: 3 packages finished [1min 20s]
```

Figura 58: Ejecución colcon build

El robot no siempre será accesible a través de una terminal, por lo que es imprescindible crear un servicio que se encargue al encender el sistema de lanzar todos los nodos automáticamente.

Para ello se crea un servicio que al arrancar el sistema operativo ejecute el script launch.sh del repositorio:

```
(.venv) lab@robocar:~/robocar/src$ cat /etc/systemd/system/robocar.service
[Unit]
Description=Robocar Nodes
After=network.target

[Service]
Type=simple
User=lab
ExecStart=/home/lab/robocar/launch.sh

[Install]
WantedBy=multi-user.target
```

Figura 59: Servicio robocar.service

Tras esto, basta con recargar los demonios de systemd con el comando “sudo systemctl daemon-reload” y posteriormente habilitarlo con el comando “sudo systemctl enable robocar.service”.

El contenido del script launch.sh es el siguiente:

launch.sh

```
1  #!/bin/bash
2  source /home/lab/robocar/.venv/bin/activate
3  source /home/lab/robocar/src/install/setup.sh
4  ros2 launch teleop_twist_joy teleop-launch.py &
5  ros2 run robocar_pkg car_control_node &
6  ros2 run robocar_pkg energy_node &
7  ros2 run robocar_pkg camera_node
8
```

Figura 60: launch.sh

Implementación del Panel de Control

El panel de control se debe encargar de mostrar todos los datos de los sensores de manera clara y entendible. Tras un análisis de todas las posibilidades, se opta por realizar el panel de control en Node-RED. Node-RED es una herramienta de programación visual que se ejecuta sobre Node.js, diseñada para la creación de flujos de datos interactivos y fácilmente configurables, orientados principalmente a aplicaciones de Internet de las Cosas (IoT). Es una aplicación construida sobre Node.js, detalle que posteriormente hay que tener en cuenta para interconectarlo con ROS2.

El uso de Node-RED tiene una serie de ventajas muy significativas, entre las que se destacan una interfaz visual intuitiva y sencilla de manejar y la escalabilidad, puesto que una vez interconectada con ROS2, nos ofrece una amplia gama de posibilidades de utilizar los datos para fines más complejos, como la optimización del rendimiento del robot y la detección anticipada de fallas mediante el monitoreo continuo y la evaluación de tendencias de los datos sensoriales.

Instalación de Node-RED

Para instalar Node-RED en la Raspberry hay que tener instalado previamente Nodejs. El método más sencillo es hacerlo a través del repositorio oficial de Ubuntu. También es necesario instalar npm, el gestor de paquetes de Nodejs.

Una vez instalado hay que establecer una interfaz de conexión entre estas dos herramientas de forma que Node-RED sea capaz de visualizar los nodos que estén ejecutándose en ROS2 y poder recibir los datos de los topics.

Interconexión Node-RED – ROS2

Para intercontectar estas dos herramientas hay que revisar la arquitectura de ROS2, mostrada en el diagrama de componentes de la figura X. De este diagrama los componentes más interesantes son:

- **ROS Client Library**: Biblioteca de bajo nivel que ofrece una interfaz genérica para la capa de middleware de ROS 2. Funciona como una capa de abstracción que se comunica directamente con el middleware, permitiendo que las implementaciones en diferentes lenguajes de programación accedan a las funcionalidades comunes de ROS 2, tales como la creación de nodos, publicación y suscripción a temas, y el manejo de servicios y acciones.
    - **rclPy**: Proporciona una interfaz para que los desarrolladores de Python puedan crear nodos de ROS 2 y gestionar comunicaciones de temas, servicios y acciones. Es la API que se usa en este proyecto.
    - **rclCPP**: Implementación de la ROS Client Library para C++. Ofrece funcionalidades similares a `rclPy`, pero aprovecha las capacidades de rendimiento y tiempo real de C++.

Estas dos interfaces vienen por defecto con la instalación de ROS2, pero para interconectar ROS2 con Node-RED hay que añadir otra interfaz para el lenguaje de

programación JavaScript. En el mismo nivel que estas dos librerías está también rclNodejs, fundamental para la comunicación entre las dos herramientas.

En primera instancia, se intenta instalar este plugin a través del proyecto oficial (Integration Service) de eProsima. Sobre la Raspberry se intenta compilar el paquete con Colcon sin éxito, debido a la insuficiencia de recursos, concretamente de RAM (4GB), para construir el paquete.

Debido a estas limitaciones, se ha optado por buscar alternativas más adecuadas para la capacidad de la Raspberry Pi. En este caso, se decide seguir con la instalación del plugin Eduart-Node-RED para ROS 2, disponible en el repositorio de GitHub: EduArt-Robotik/edu_nodered_ros2_plugin. Este plugin está diseñado específicamente para integrar Node-RED con ROS 2, proporcionando una solución más ligera y manejable que se ajusta mejor a las capacidades de la Raspberry Pi.

Esta alternativa levanta un contenedor Docker donde dentro tiene ROS2 y Node-RED con el plugin de interconexión de ambas herramientas. En el caso de este proyecto, se ejecutan todos los comandos del Dockerfile directamente sobre la Raspberry.

Una vez instalado, se deben ver los nodos del plugin de ROS2 en Node-RED, tal y como se muestra en la figura X.

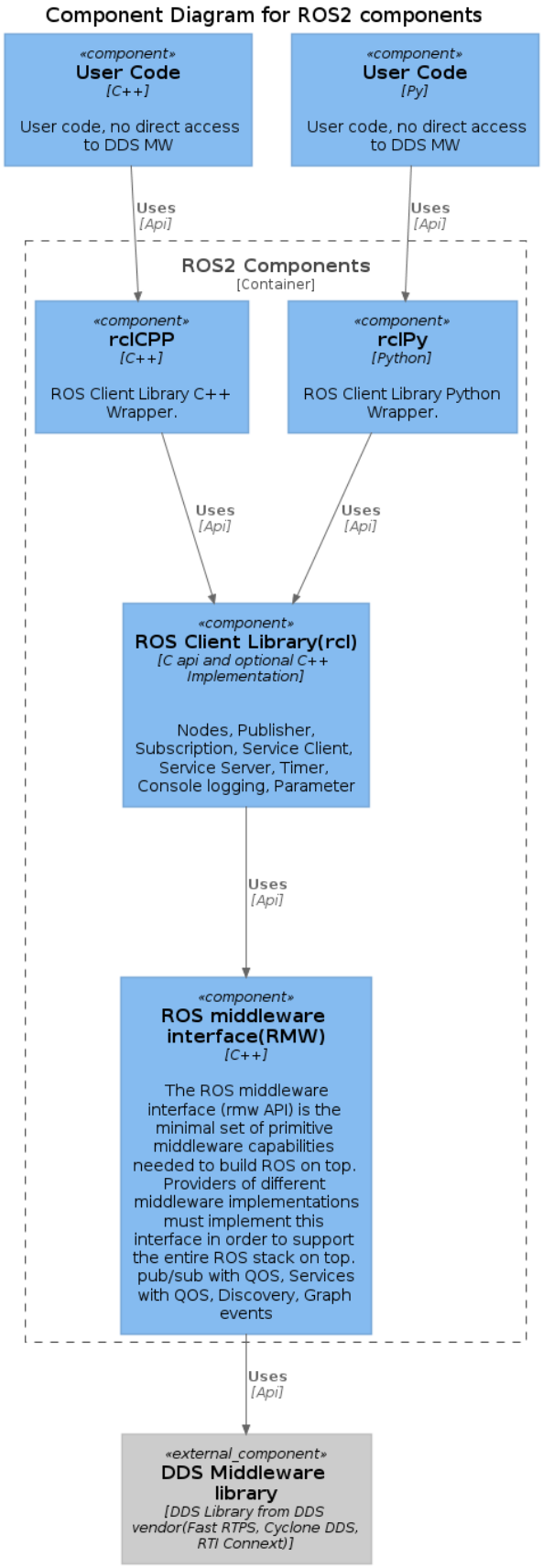


Figura 61: Arquitectura de ROS2

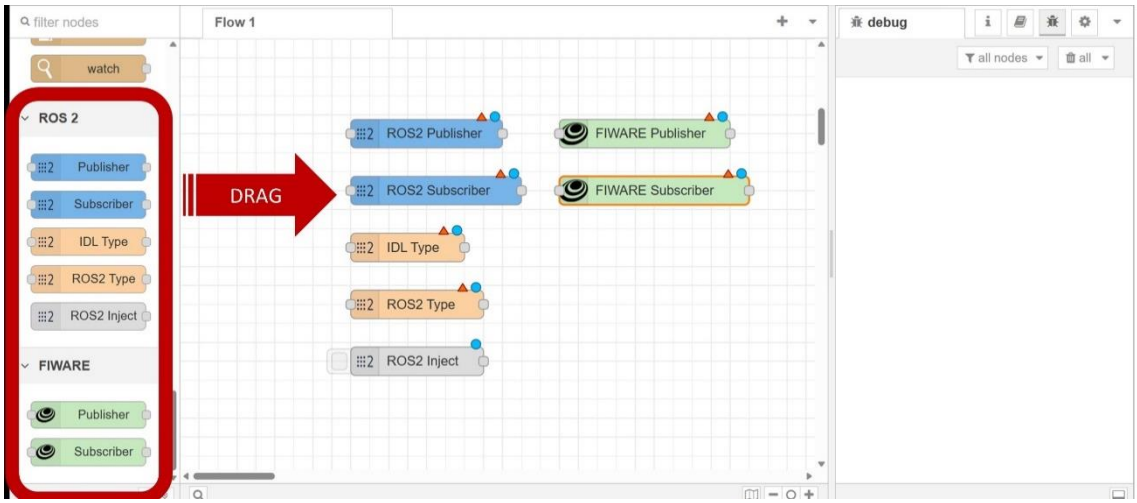


Figura 62: Nodos ROS2 en Node-RED

Creación del Dashboard en Node-RED

Una vez interconectados ROS2 y Node-RED, ya se pueden recoger los datos publicados por los nodos de ROS2. En el repositorio se encuentra Para ello, se sigue la siguiente estructura de nodos de Node-RED:

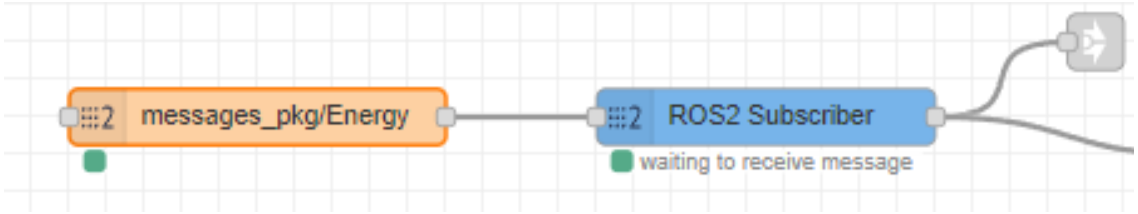


Figura 63: Ejemplo de uso de nodos en Node-RED

- **messages_pkg/Energy**: En primera instancia se indica el tipo de mensaje que se va a recibir

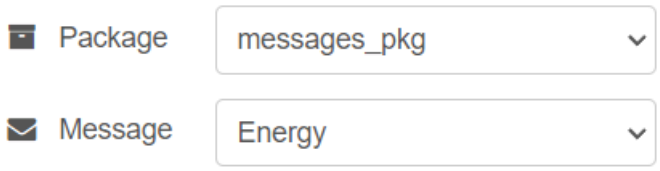


Figura 64: Propiedades ROS2 Type

- **ROS2 Suscriber**: Aquí se indica el nombre del Topic de ROS2 (/energy). El Domain ID el que usa ROS2 por defecto (0).

Figura 65: Propiedades ROS2 Subscriber

Además de estos nodos, se usan los nativos de Node-RED para la creación del dashboard, ubicados en la paleta y nombrados como “dashboard” y “dashboard 2”.

Diagrama de Batería

Con los nodos del plugin de ROS2, se recogen los datos del topic /energy y se usan para obtener la información de:

- Carga de Baterías
- Energía Consumida
- Equilibrio de Celdas (Salud de la batería)

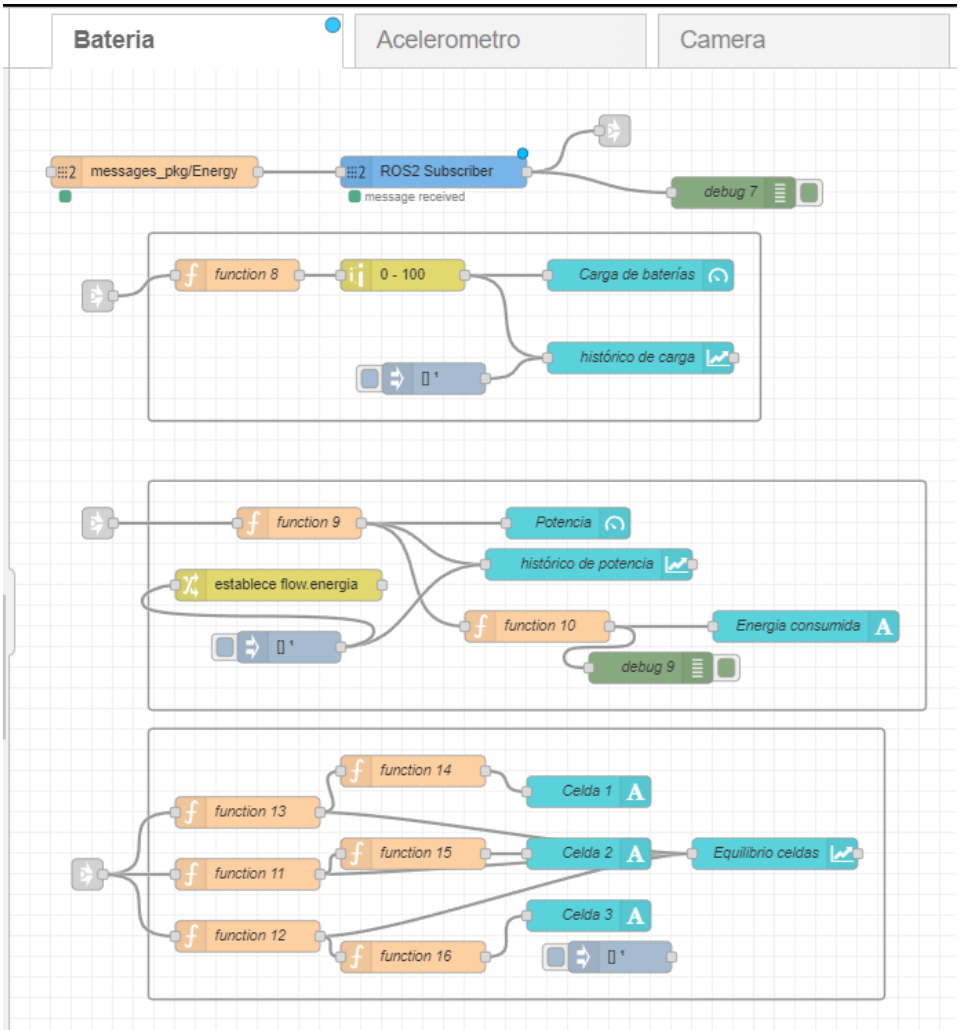


Figura 66: Diagrama de Batería

Diagrama de Cámara

Con el uso de otros nodos de Node_RED, se puede visualizar en tiempo real (con un retardo de algunos segundos) los frames de la cámara.

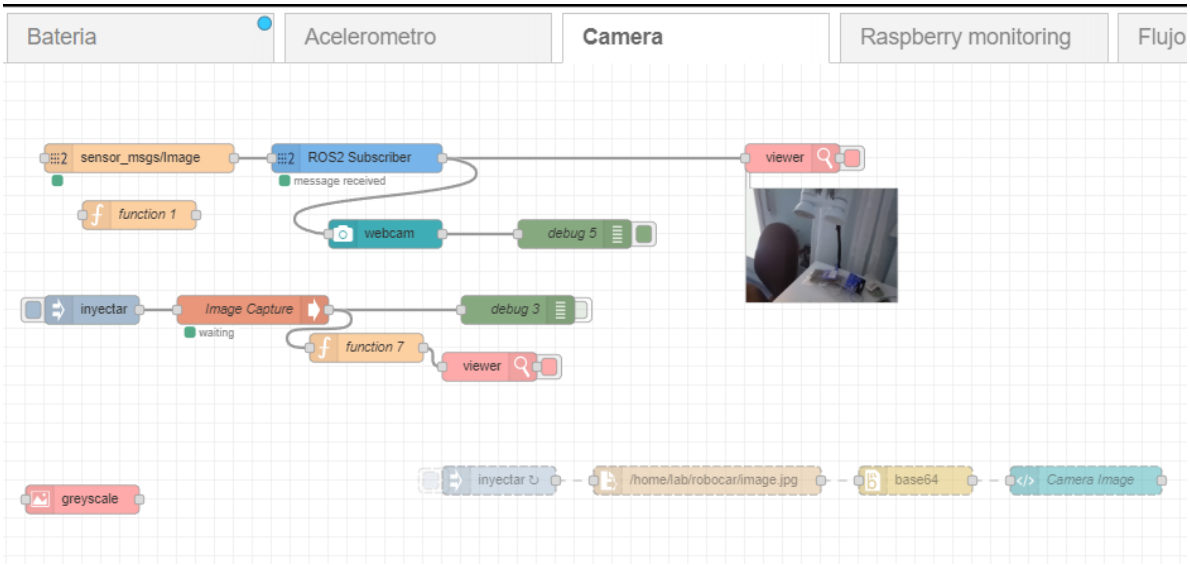


Figura 67: Diagrama de Cámara

Diagrama de monitoreo de la Raspberry

Con este diagrama se consigue tener un monitoreo a tiempo real del estado de la Raspberry:

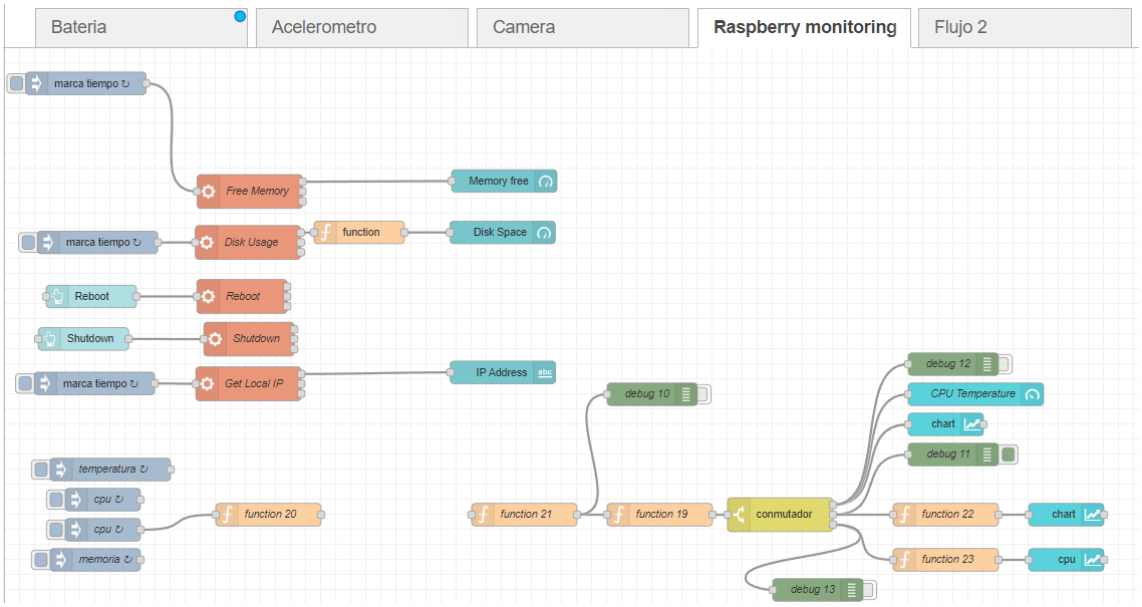


Figura 68: Diagrama de Monitoreo de Raspberry

Panel de Control: Energía

- **Salud Baterías**: Mide la diferencia de potencial entre las baterías leyendo del topic /energy.
- **Potencia**: Aquí se mide en W la potencia consumida por el coche. En este caso se puede ver que la potencia es negativa. Esto es debido a que está conectado con una fuente de alimentación externa, por lo que no circula corriente desde las baterías hasta la carga.
- **Carga Baterías**: Aquí se calcula un porcentaje de batería a partir de su voltaje.

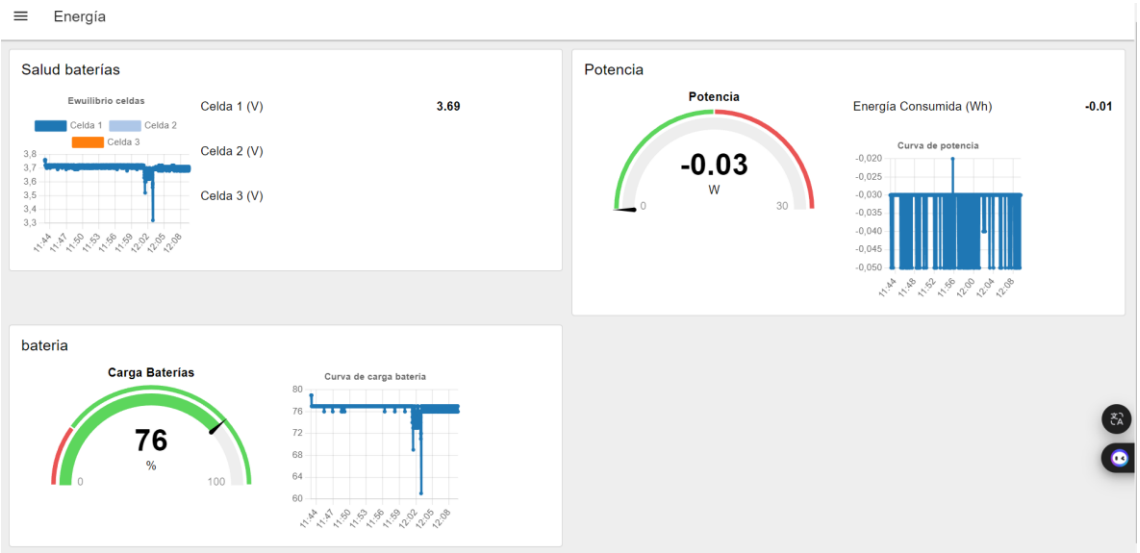


Figura 69: Panel de Control de Energía

Panel de Control: Cámara

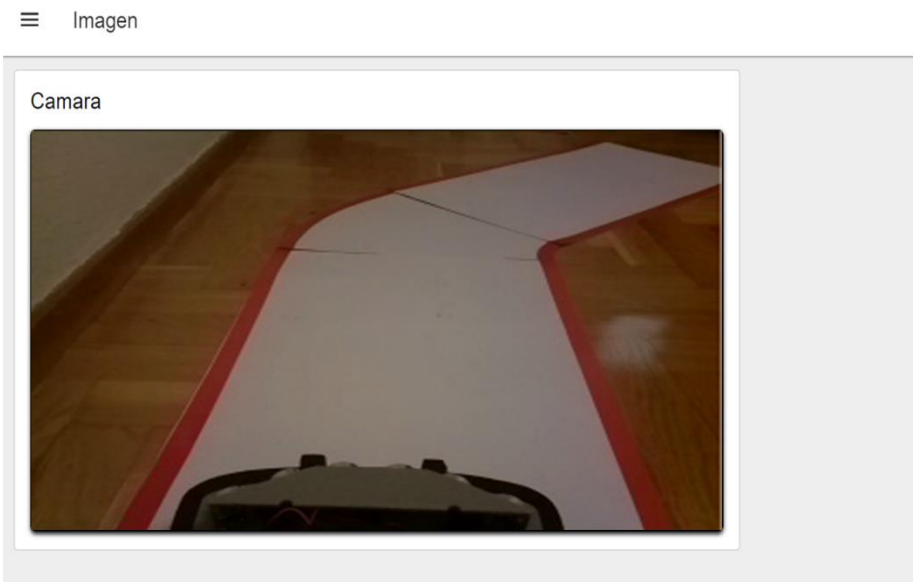


Figura 70: Panel de Control: Cámara

Panel de Control: Raspberry Pi

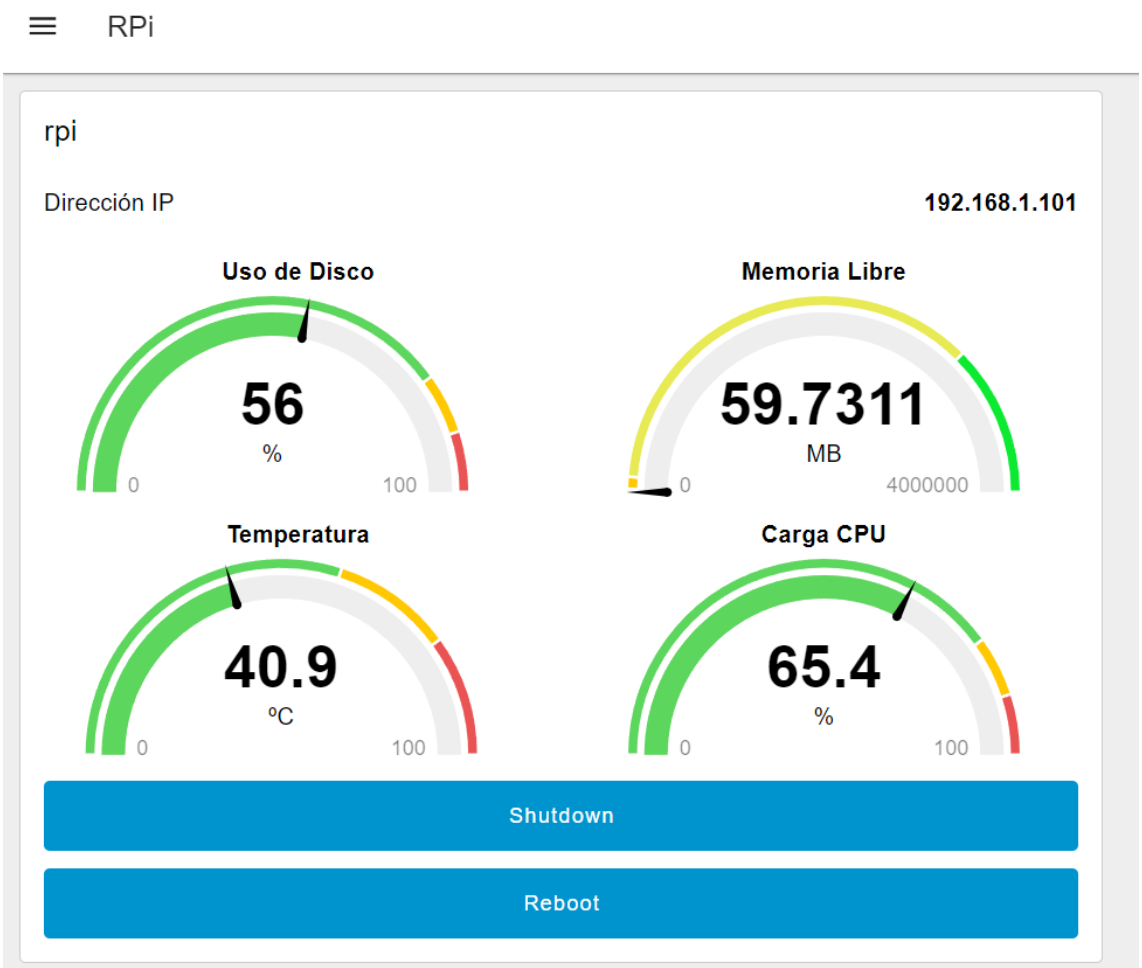


Figura 71: Panel de Control: RPi

Conclusiones

Durante el desarrollo del proyecto de construcción de un robot coche autónomo, se concluye que es posible alcanzar un producto hardware con un nivel de desarrollo maduro y potencial grado de autonomía considerable utilizando componentes comerciales y con un presupuesto contenido.

Metodología

Se determina que la metodología de trabajo en espiral incremental es efectiva para la implementación de sistemas de estas características, permitiendo ajustes rápidos basados en la retroalimentación y la detección temprana de errores.

Viabilidad Técnica y Económica

Se determina que es posible desarrollar un sistema autónomo funcional utilizando componentes comerciales de bajo costo. La correcta selección y calibración de sensores económicos permiten lograr resultados precisos y fiables, demostrando que un presupuesto limitado no necesariamente compromete la calidad del proyecto.

Desempeño de sensores

El benchmarking determina que el rendimiento de los sensores es adecuado bajo diferentes condiciones de luz y distancias. Se destaca que los errores de medición se mantienen dentro de márgenes aceptables (<20%), y que la luz ambiental no afecta significativamente el desempeño de los sensores. Además, resulta sorprendente que el sensor que demuestra mejores características generales es uno de los más baratos y aparentemente 'básico' de todos los probados.

Integración de Conocimientos

Se integran conocimientos de diversas áreas, como la electrónica, programación y diseño mecánico, demostrando la interdisciplinariedad necesaria en proyectos de ingeniería moderna.

Aplicación Práctica

Se evidencia la aplicabilidad práctica de los conceptos teóricos aprendidos en la universidad, reforzando la importancia de una educación basada en aplicación práctica del conocimiento adquirido.

Complicaciones del mundo físico

Destacar las complejidades añadidas que ocurren durante el desarrollo de un proyecto con componente física. En general, las predicciones de tiempo para todos los procesos que implican construcción de componentes son errados a la baja, y con frecuencia es necesario repetir, reconstruir y testear repetidas veces un mismo componente. Pese a todo, la satisfacción de conseguir un dispositivo hardware funcional es destacable.

Aplicaciones y Futuras Mejoras

Se determina que proyectos similares pueden beneficiarse de la documentación detallada y las estrategias de implementación utilizadas en este proyecto. Se sugiere que futuras iteraciones podrían explorar la incorporación de sensores adicionales o más avanzados, así como el uso de algoritmos de inteligencia artificial para mejorar la toma de decisiones y la autonomía del robot. También puede resultar un activo de valor el añadido de hardware de procesamiento más potente, como la Nvidia Jetson Nano, para correr algoritmos de procesamiento de imagen más potente a tiempo real.

En conclusión, se establece que la realización de este proyecto no solo es técnicamente posible con recursos limitados, sino que también proporciona una base sólida para futuros desarrollos y aplicaciones en el ámbito de la robótica autónoma a bajo costo.

Resultado Final

Tras la implementación de todos los bloques, así queda el robot:

Figura 72: Robocar

Tabla de Figuras

Tabla de tablas

Referencias

gp2y. (s.f.). Obtenido de https://www.luisllamas.es/medir-distancia-con-arduino-y-el-sensor-gp2y0e03/#esquema-de-montaje

HCSR04. (s.f.). Obtenido de https://thepihut.com/blogs/raspberry-pi-tutorials/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi

ky032. (s.f.). Obtenido de https://www.alldatasheet.es/datasheet-pdf/pdf/1284503/JOY-IT/KY-032.html

vl53l0x. (s.f.). Obtenido de https://www.st.com/resource/en/datasheet/vl53l0x.pdf