Trabajo fin de Máster

Universidad de Valladolid

Máster en Ingeniería de Telecomunicación

Puesta a punto y optimización de un software de control lateral para un prototipo de vehículo autónomo

Autor:
David Manso Fernández

Tutores:
Juan Carlos Aguado Manzano
Adrián Mazaira Hernández

Septiembre 2024

Título:	**Puesta a punto y optimización de un software de control lateral para un prototipo de vehículo autónomo**
Autor:	**David Manso Fernández**
Tutor:	**Juan Carlos Aguado Manzano**
Departamento:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

Tribunal

Presidente:	**Ignacio de Miguel Jiménez**
Secretario:	**Ramón de la Rosa Steinz**
Vocal:	**Jesús María Hernández Mangas**

Fecha:	26 de septiembre de 2024
Calificación:	

Agradecimientos

Me gustaría dedicar unas palabras de agradecimiento a todas aquellas personas que han sido fundamentales en este proceso.

En primer lugar, quiero agradecer profundamente a mis padres, Ramiro y Lola, por su apoyo incondicional, su paciencia y por haber estado siempre a mi lado, brindándome la fuerza necesaria para seguir adelante. Su amor y dedicación han sido el pilar sobre el que he construido cada logro.

A mis hermanos, Jaime y María, por ser siempre una fuente de inspiración y por acompañarme en cada paso de este camino. Vuestro cariño y ánimo constante han sido fundamentales para mantenerme motivado en todo momento.

Quiero expresar mi gratitud también a mis tutores, Juan Carlos Aguado y Adrián Mazaira, por su guía y sus valiosas enseñanzas a lo largo de este proyecto. Sus consejos han sido esenciales para que este trabajo pudiera alcanzar el nivel que esperaba.

A Ignacio Royuela y Óscar Pérez, por su apoyo en los momentos más complejos del proyecto, por compartir sus conocimientos y por estar siempre dispuestos a ayudar. Sin vosotros, este proyecto no habría sido posible.

También quiero agradecer de corazón a toda la gente que de una manera u otra ha contribuido a que este proyecto se haya materializado. A todos los que me habéis brindado vuestro tiempo, apoyo y comprensión durante este proceso, os estoy eternamente agradecido.

Por último, este trabajo está dedicado con todo mi amor a Lucas y Carmela. Aunque todavía sois pequeños para entenderlo, sois una fuente de alegría inmensa en mi vida. Que este logro sea un ejemplo para vosotros de que con esfuerzo y dedicación todo es posible.

Resumen

Este Trabajo Fin de Máster ha consistido en la mejora, reestructuración y simplificación de la arquitectura y sistemas de un prototipo de vehículo autónomo. Al comienzo del proyecto el prototipo vehículo autónomo estaba inoperativo, faltaban varios módulos electrónicos por incorporar a la arquitectura y el esquema eléctrico era deficiente. Las tareas realizadas han incluido la modificación completa del cableado auxiliar del vehículo para convertirlo en autónomo, el rediseño de un controlador electrónico para el acelerador, la incorporación a la arquitectura de varios controladores electrónicos todavía no incluidos, entre los que destacamos el control de marchas y el control de acelerador, y la mejora del control lateral a través de la modificación de la aplicación de control.

Palabras Clave: Renault Twizy, TwizyContest, Vehículo Autónomo, Control Lateral, Acelerador, Caja de cambios, Python, Arduino.

Abstract

This Master Thesis consisted of improving, restructuring, and simplifying the architecture and systems of an autonomous vehicle prototype. At the beginning of the project, the autonomous vehicle prototype was inoperative, several electronic modules were missing to be incorporated into the architecture and the electrical schematic was deficient. The tasks performed included the complete modification of the vehicle's auxiliary wiring to make it autonomous, the redesign of an electronic controller for the accelerator, and the incorporation into the architecture of several electronic controllers not yet included, among which we highlight the gear control and the accelerator control, and the improvement of the lateral control through the modification of the control application.

Keywords: Renault Twizy, TwizyContest, Autonomous Vehicle, Lateral Control, Throttle, Gearbox, Python, Arduino.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

Este Trabajo de Fin de Máster es la continuación de la propuesta presentada por un grupo de jóvenes emprendedores para el concurso **TwizyContest 2020**, denominado **TwizyLine**. Estos jóvenes de la Universidad de Valladolid son: **Adrián Mazaira Hernández**, **Ignacio Royuela González**, **Mario Martín Fernández** y **Samuel Pilar Arnanz**[1]. Su proyecto proponía la creación de estacionamientos automatizados para vehículos autónomos, donde los automóviles equipados con la tecnología adecuada no solo podrían estacionarse por sí mismos, sino también recargarse automáticamente.

Figura 1.1: Logo del concurso *TwizyContest* 2020

Este proyecto fue distinguido con varios premios, incluido el primer premio a nivel nacional en el concurso organizado por Renault Group y Segula Technologies, y posteriormente el segundo premio a nivel internacional en el mismo certamen[2]. También fue ganador de los premios Prometeo y del concurso Campus Emprendedor[3]

La idea principal del proyecto consistía en desarrollar un prototipo de vehículo autónomo asequible con la finalidad de crear estacionamientos automatizados utilizando vehículos de conducción autónoma y conectados. El prototipo se centraba en proporcionar una solución innovadora y eficiente que pudiera integrarse en un vehículo eléctrico, en este caso el **Renault Twizy**, de acuerdo con los requisitos del concurso. La implementación de vehículos autónomos planteaba desafíos técnicos y de aceptación por parte del público. Por lo tanto, en el proyecto presentado al concurso se planteó la idea de demostrar la utilidad práctica de esta tecnología en un entorno controlado, como los estacionamientos automa-

tizados, como una manera efectiva de superar las barreras psicológicas de los usuarios y demostrar la viabilidad de un modelo de negocio para esta solución. [4]

Figura 1.2: Logo del concurso *TwizyLine*

La industria automotriz tiene diversas razones para buscar soluciones como esta. En primer lugar, los vehículos han evolucionado de simples máquinas de combustión de cuatro ruedas a complejos sistemas que incorporan cada vez más tecnologías emergentes. La seguridad vial ha sido un motor importante para la investigación, mejora e integración de nuevas tecnologías en los vehículos, ya que una conducción eficiente es una estrategia efectiva para prevenir accidentes. La conducción eficiente implica seguir una serie de pautas para mejorar la seguridad vial, el confort del conductor y reducir la contaminación. Algunas de las ventajas de este tipo de conducción son [5]:

- Mejora del confort de conducción: evita frenazos y acelerones bruscos, creando un entorno de conducción tranquilo y seguro, reduciendo así el riesgo de accidentes.
- Ahorro de energía: el comportamiento del conductor influye en el consumo del vehículo, y un manejo eficiente conlleva un menor consumo de energía y una mayor autonomía del vehículo.
- Incremento de la seguridad vial: mantener una distancia de seguridad adecuada es fundamental para contar con un tiempo de reacción suficiente.

A pesar de que el proyecto TwizyLine no se implementó en su totalidad, se lograron avances significativos que respaldan la posibilidad de su desarrollo en el futuro [6]. Las partes ya implementadas y documentadas de los miembros del equipo inicial incluyen el diseño y funcionamiento del estacionamiento, el desarrollo de software para una aplicación TwizyLine, la implementación de dos módulos de comunicación y control similares a unidades de control electrónicas (ECU), la creación de un servidor para gestionar los estacionamientos, y un estudio de un plan de negocios adecuado para el proyecto. Esto fue posible gracias a la donación de un Renault Twizy (Figura 1.3) por parte de la Fundación Renault, a la Universidad de Valladolid y la colaboración de Renault Group para asesorar en las diferentes modificaciones del vehículo.[7][4]

Desde la primera versión del prototipo se han ido implementando diversas mejoras con desigual resultado. Por ejemplo, se ha intentado mejorar la forma de localizar y guiar el vehículo mediante el uso de tecnología LiDAR, pero no se llegó a implementar en su totalidad por diversas razones.[5] También se propuso un cambio en la configuración del esquema eléctrico y de comunicaciones del vehículo, que se implementó parcialmente. Finalmente, el sistema de control lateral del vehículo tendría que ser mejorado sustancialmente para conseguir un mejor control y una velocidad adecuada del vehículo. Existen otras muchas mejoras que se podrían implementar en el vehículo, pero estas son las fundamentales para una mejora sustancial de su comportamiento. [6][5][8]

Figura 1.3: Renault Twizy

Vistas las carencias que presenta el prototipo actual, este Trabajo Fin de Máster tiene como objetivo fundamental resolver las carencias en cuanto al control lateral del mismo y todo lo que este involucra. Por lo tanto, el objetivo principal de este Trabajo de Fin de Máster es llevar a cabo las modificaciones, tanto de *hardware* como de *software*, necesarias en el vehículo para optimizar y perfeccionar el sistema de control lateral que implicaría desde una revisión de todos los sistemas electrónicos y sus comunicaciones hasta una mejor implementación del programa que ejecuta el control lateral de tal manera que al finalizar el proyecto se tenga un prototipo que utilizando el sistema de guía que ya tenía sea capaz de moverse con mayor precisión y a mayor velocidad.

1.2. Objetivos

El objetivo fundamental de este proyecto es la optimización del sistema de control lateral actual en el prototipo con el objetivo de ser capaces de girar las ruedas a una mayor velocidad, que nos permita aumentar la velocidad de paso en las curvas sin desviarnos de la trayectoria marcada por la línea azul que sigue el coche. Dado que el vehículo, al comienzo de este Trabajo Final de Máster, estaba inoperativo, llegar a este objetivo principal requería conseguir otros objetivos secundarios:

- Revisión y reestructuración integral de la alimentación y cableado de los componentes adicionales a este vehículo, incluyendo el diseño y fabricación de una PowerBox que permitiera una mayor modularidad a la hora de controlar el encendido y apagado de los equipos.
- Revisión del estado de otras unidades de control que no se pudieron probar en proyectos anteriores y en su caso rediseño de las mismas para conseguir un prototipo funcional en los términos expresados anteriormente.
- Modificación del programa de control lateral para mejorar la velocidad de respuesta del volante.

1.3. Fases y métodos

Para llevar a cabo el proyecto, se han seguido las siguientes fases:

1. **Revisión del estado del vehículo y de la documentación asociada:** La primera fase ha consistido en la familiarización con todos los aspectos de los proyectos anteriores relacionados con el presente, con especial atención a las posibles mejoras que ya se habían detectado antes de comenzarlo.

2. **Reestructuración de la arquitectura y cableado:** Una vez realizada la revisión del estado del vehículo que, como ya se dijo anteriormente, estaba complemente desmontado, se ha realizado una reestructuración de la alimentación de los componentes y su cableado de comunicaciones con el objetivo de obtener un vehículo más ordenado y simplificado, con los cables bien fijados y robustos evitando posibles fallos en el futuro. Además, se han implementado diversas medidas de seguridad y de ahorro de energía a través de una PowerBox.

3. **Análisis del funcionamiento de los componentes del vehículo:** Una vez se había resuelto la reestructuración eléctrica y de comunicaciones del vehículo, se pasó a comprobar que todos los componentes necesarios para el funcionamiento del prototipo estaban operacionales. Estos componentes se fueron probando individualmente con el objetivo de ver que no hubiera ningún fallo en ellos y se pudieran integrar *a posteriori*.

4. **Desarrollo y puesta a punto de los controladores del acelerador y marchas:** Después de la fase de análisis de funcionamiento de los componentes del vehículo, se pasó a rediseñar aquellos que fallaban y a diseñar y/o implementar los que faltaban. En concreto, se detectó que no existía un módulo software integrado en el programa principal que actuara sobre el controlador de marchas. A su vez, el controlador del acelerador del que se partía no funcionaba correctamente y carecíamos de un programa adecuado o una explicación adecuada sobre cómo debería haber funcionado. Por lo tanto, se decidió rediseñar volviendo a una versión anterior de la que teníamos conocimiento que funcionaba correctamente y que fue desechada por la calidad de su fabricación.

5. **Desarrollo del Nuevo Sistema de Control Lateral:** Una vez se tuvo un vehículo operativo, se procedió al desarrollo del nuevo sistema de control lateral. El objetivo principal es permitir que el vehículo pueda circular a una mayor velocidad sin desviarse de la ruta marcada. A la vez, una vez comprobada la nueva velocidad de giro del volante, se realizaron simulaciones sobre la nueva velocidad soportada por el vehículo en el circuito de estacionamiento diseñado en proyectos anteriores.

6. **Fase de pruebas:** Para finalizar se realizaron una serie de pruebas para comprobar que el vehículo estaba operativo y preparado para las siguientes mejoras a incorporar.

1.4. Recursos

El desarrollo de este proyecto ha sido posible gracias a la donación del ***Renault Twizy*** por parte de la empresa Renault Group y a la colaboración de la **Escuela Técnica Superior de Ingenieros de Telecomunicación** (ETSIT), que proporcionó un espacio para estacionar el vehículo y llevar a cabo el trabajo.

La mayoría del material empleado se destina a la integración en el vehículo. A continuación, se presenta una lista de los productos utilizados:

- **Renault Twizy:** Vehículo base para el proyecto (Figura 1.3).

- **Humming Board CBi (x2):** Se utilizarán dos Humming Board CBi (ver figura 1.4) como Unidades de Control Electrónico (ECU) en el vehículo. Estos dispositivos ejecutarán el sistema operativo Linux Debian 10.0, el cual está preconfigurado con un conjunto de repositorios que facilitan la instalación de software disponible para Linux. Cada uno de estos dispositivos tendrá funciones específicas, uno será el encargado de las comunicaciones y el otro del control del vehículo y los sensores externos.

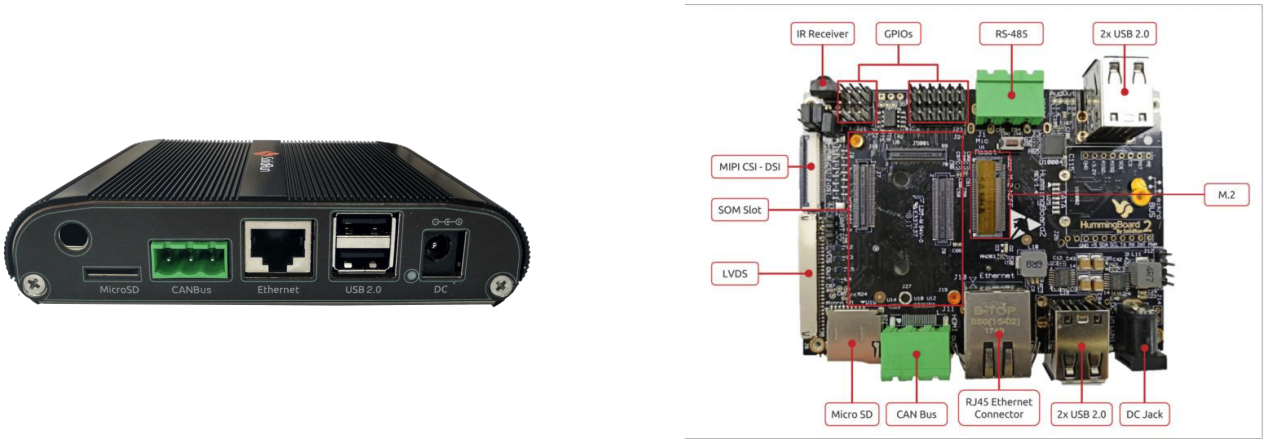


Figura 1.4: Humming Board CBi [7]

- **PowerBox:** Se ha diseñado una caja que alberga 6 fusibles en su interior junto con sus respectivos interruptores, permitiendo la conexión de diversos equipos, siendo capaces de controlar alimentación de manera modular, como se detallará más adelante.
- **Sensores de Ultrasonidos de Proximidad (x3):** Utilizados para la detección de objetos cercanos.
- **Sensor de RFID:** Utilizado para la detección de etiquetas RFID ubicadas en el suelo.
- **Hub USB de 6 Puertos:** Proporciona conectividad USB adicional.
- **Receptor GPS Garmin GPS18x USB:** Para la obtención de datos de posición y navegación (ver figura 1.5).

Figura 1.5: Receptor GPS Garmin GPS18x [7]

- **Maxon EPOS4:** Controladora del motor lateral mostrada en la figura 1.6.
- **Motor Maxon:** Incluye un motor de 100 W con su *encoder* MILE y una reductora con relación $1:81$.
- **SpyBox CAN:** Es un *hub* de buses CAN, el cual dispone de 5 puertos DB9 hembra y 1 puerto DB9 macho en cada canal.
- **Router:** El *router* se situará en la parte superior del *rack* y creará una red local en la que podremos comunicarnos con las Humming a través del protocolo SSH. Así como nos permitirá tener el servidor y cliente *mosquitto* dentro de la misma red.

Figura 1.6: Controladora EPOS4 Maxon Motor[7]

1.5. Estructura de la memoria

La memoria se organiza en tres partes principales, que abarcan desde el nivel más técnico de hardware hasta la implementación del software final. Primero, se describe todo lo relacionado con el cableado, las conexiones y los componentes electrónicos, estableciendo la arquitectura necesaria para el funcionamiento del proyecto. A continuación, se aborda el estudio de los diferentes controladores del Twizy, que gestionan las marchas, el acelerador y el giro de las ruedas. Se explicará la forma que tenemos de comunicarnos con ellos para poder controlar el vehículo. Finalmente, se hablará del programa principal junto con el del control lateral, que integra y coordina todos los elementos anteriores, asegurando un funcionamiento completo del sistema.

Capítulo 2

Revisión del proyecto TwizyLine

2.1. Introducción

La revolución de los vehículos autónomos ha capturado la atención del mundo, prometiendo transformar nuestra forma de transporte de una manera nunca antes vista y emergiendo como un área de investigación y desarrollo que promete transformar radicalmente la industria. Entre los diversos modelos de vehículos que se han utilizado como plataformas de experimentación, el Renault Twizy, un automóvil eléctrico compacto diseñado para la movilidad urbana, ha surgido como un candidato interesante que gracias a su simplicidad lo convierte en un lienzo ideal para explorar las posibilidades de la conducción autónoma.

En este capítulo revisamos el estado del proyecto TwizyLine, que será nuestro punto de partida, explicando su idea general y funciones implementadas hasta el momento.

2.2. El Renault Twizy como concepto de movilidad sostenible

El Renault Twizy (ver figura 2.1) es un vehículo eléctrico fabricado por el fabricante francés de automóviles Renault. Se lanzó al mercado en 2012 como un modelo de movilidad urbana, diseñado para ofrecer una alternativa compacta y eficiente para desplazamientos cortos en entornos urbanos congestionados. Con su diseño peculiar y su naturaleza eléctrica, el Twizy capturó la atención de los consumidores y los entusiastas de la movilidad eléctrica en todo el mundo. [9] [10]

Figura 2.1: Renault Twizy

El Twizy tiene capacidad para dos personas, con asientos dispuestos en tándem, lo que significa

que uno se sienta detrás del otro. Esta disposición compacta le permite al Twizy ocupar menos espacio en la carretera y facilita su manejo en entornos urbanos estrechos y concurridos. Además, su diseño abierto y sin ventanas laterales lo hace perfecto para climas cálidos y soleados. (ver Figura 2.2) [11]

Figura 2.2: Diseño del Renault Twizy

En términos de rendimiento, el Twizy está equipado con un motor eléctrico que ofrece una potencia modesta pero adecuada para la conducción en la ciudad. Se puso a la venta con dos motorizaciones [12] [13]:

- **Twizy 45**: Tiene un motor de 7.6 kW o 10 CV que alcanza una velocidad máxima de 45 km/h.
- **Twizy 80**: Tiene un motor de 12.6 kW o 17 CV que alcanza una velocidad máxima de 80 km/h.

Figura 2.3: Interior del Renault Twizy

Su batería eléctrica proporciona una autonomía que varía según las condiciones de conducción y la motorización, llegando hasta los 90 km y 100 km respectivamente en función de la versión. De todos modos, el Twizy está pensado para recorridos cortos y desplazamientos urbanos.[12] [9]

Una de las características más destacadas del Twizy es su enfoque en la sostenibilidad y la eficiencia energética. Al ser completamente eléctrico, el Twizy no produce emisiones de escape, lo que

lo convierte en una opción respetuosa con el medio ambiente para la movilidad urbana. Además, su diseño ligero y compacto, ya que tiene unas dimensiones de 2.3 m de largo y de tan solo 1.2 m de ancho, permitiendo un radio de giro desde 3.4 metros como se muestra en la figura 2.4. [13][12]

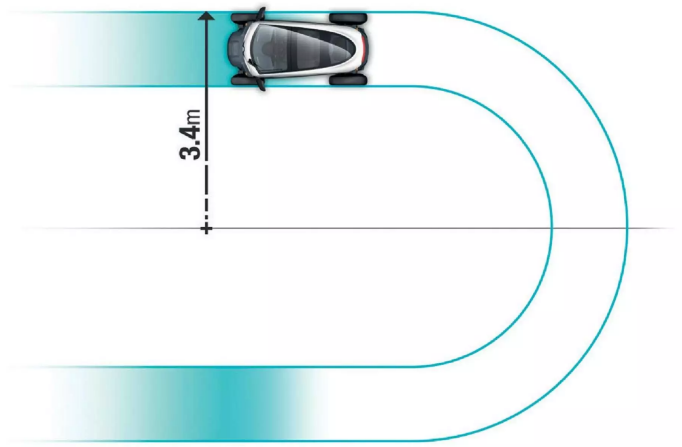


Figura 2.4: Radio de giro de Renault Twizy

A lo largo de los años, el Twizy ha ganado popularidad en el mercado de vehículos eléctricos y se ha convertido en una opción atractiva para aquellos que buscan una alternativa económica y sostenible para desplazamientos urbanos. Comenzó fabricándose en Valladolid, aunque desde finales de 2018, su producción se trasladó a Corea. En total, desde su lanzamiento hasta junio de 2023, la marca gala ha comercializado **33.340 unidades** del Twizy en 55 países, con un especial éxito en Francia, Alemania, Corea del Sur e Italia. [12]

2.3. Funcionamiento y evolución de los vehículos autónomos

El funcionamiento de los vehículos autónomos se basa en una compleja combinación de tecnologías, incluyendo sensores, sistemas de procesamiento de datos y algoritmos de control. A lo largo de los años, hemos sido testigos de una rápida evolución en estas tecnologías, desde los primeros prototipos de vehículos autónomos hasta los sistemas altamente sofisticados que vemos hoy en día en desarrollo. Cada nuevo prototipo trae consigo mejoras significativas en la capacidad de percepción y toma de decisiones de los vehículos autónomos, acercándonos cada vez más a un futuro de transporte completamente autónomo.

El funcionamiento de un vehículo autónomo se puede simplificar en tres partes:

- **Percepción del entorno**: Gracias a la variedad de tecnologías de sensores, entre las que destacan cámaras (Figura 2.6), escáneres LiDAR (Ver Figura 2.5) y radares de diferente tipo, nos permiten percibir el entorno y poder detectar obstáculos, peatones, señales de tráfico y otros vehículos en la carretera. Además, los vehículos autónomos suelen contar con alguna tecnología de geolocalización, que ayuda a completar la propiocepción.
- **Procesamiento de los datos obtenidos**: Los datos recopilados por estos sensores se procesan a través de sistemas de inteligencia artificial y algoritmos de aprendizaje automático, que analizan la información y toman decisiones en tiempo real sobre la navegación y el control del vehículo. Estos algoritmos son capaces de reconocer patrones en los datos sensoriales y anticipar posibles escenarios de conducción, lo que permite que el vehículo tome decisiones seguras y eficientes en la carretera.

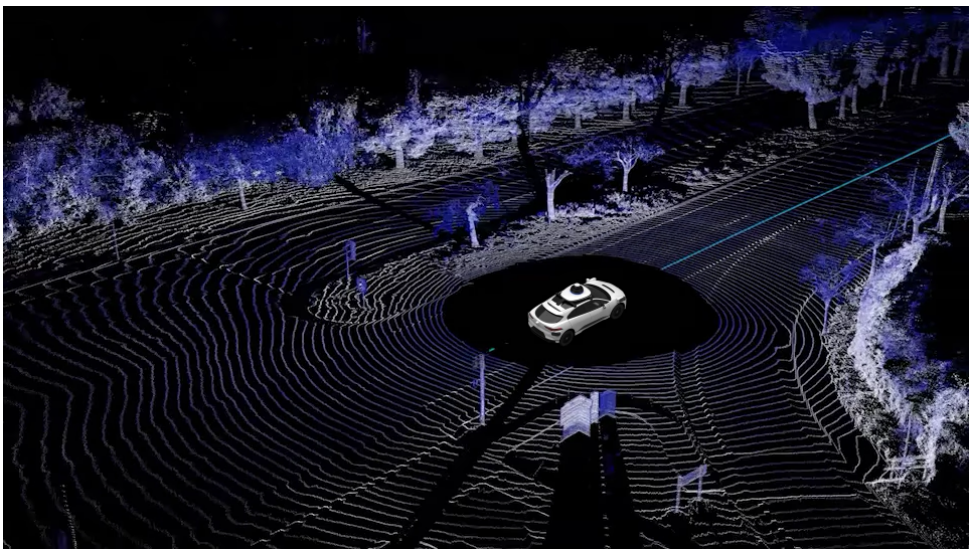


Figura 2.5: Entorno visto por el LiDAR [14]

- **Actuadores:** El último pilar de los vehículos autónomos son los actuadores que controlarán el movimiento del vehículo. Fundamentalmente, son los actuadores relacionados con el movimiento lateral y longitudinal, esto es, volante, acelerador y cambio de marchas (si el vehículo no es automático, aunque la mayoría de vehículos autónomos ya incorporan el cambio automático).

Figura 2.6: Entorno visto por las cámaras [14]

La evolución de los vehículos autónomos también ha sido impulsada por la creciente demanda de soluciones de movilidad más seguras, eficientes y accesibles. Con la promesa de reducir los accidentes de tráfico, mejorar la eficiencia del transporte y proporcionar una mayor autonomía a personas con movilidad reducida, los vehículos autónomos representan un área de investigación y desarrollo de gran importancia para la industria automotriz y la sociedad en su conjunto. [15] [5]

A lo largo de las últimas décadas, hemos sido testigos de una rápida evolución en estas áreas, lo que ha permitido el desarrollo de sistemas de conducción autónoma cada vez más sofisticados y capaces. En las primeras etapas de desarrollo, los avances se centraban principalmente en funciones básicas de asistencia al conductor, como el control de crucero adaptativo y el estacionamiento asistido. Sin embargo, en los últimos años, hemos visto un avance significativo hacia vehículos completamente autónomos, capaces de operar sin intervención humana en una amplia variedad de entornos y condiciones de conducción.[16] [15]

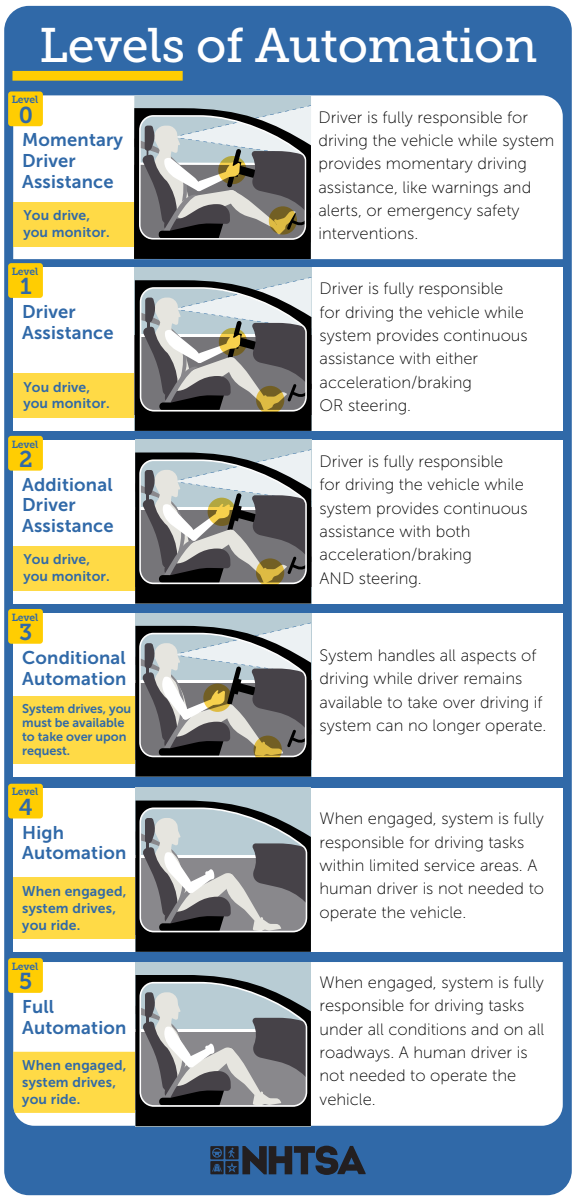


Figura 2.7: Niveles de autonomía [15]

Dada la diversidad de soluciones a las que se puede llamar vehículo autónomo, se ha llegado a un consenso sobre cómo categorizar los vehículos autónomos en niveles dependiendo del grado de autonomía que realmente es capaz de dar. Los niveles de autonomía de los vehículos son (Ver Figura 2.7) [15]:

- **Nivel 0**: Sin automatización. El conductor es responsable de todas las tareas de conducción.
- **Nivel 1**: El vehículo puede realizar algunas tareas. Estas tareas pueden ser el control lateral (mantenimiento de carril) o longitudinal (Control de Velocidad) pero las realiza de manera independiente y nunca de manera simultánea.
- **Nivel 2**: El vehículo puede controlar ambas tareas de manera simultánea. Un ejemplo puede ser el asistente de aparcamiento en el que el conductor no tenga que tocar ni el volante ni los pedales.
- **Nivel 3**: El vehículo puede realizar la mayoría de las tareas de conducción. Percibe el entorno y puede actuar en función de él. Este modo requiere supervisión constante del conductor. Algunos

vehículos actuales ya incorporan el nivel 3, como los Mercedes Clase S, que con su sistema Drive Pilot pueden moverse de manera autónoma, aunque de momento está aprobado por las autoridades a ir a un máximo de 60 km/h.

- **Nivel 4**: El coche puede realizar todas las tareas de la conducción, pero solo en dominios operacionales previamente definidos. Cuando el dominio operacional no se cumple, o bien el vehículo entra en modo seguro y se detiene o bien solicita la toma de control por parte del conductor. El proyecto TwizyLine se enmarcaría dentro de este nivel.
- **Nivel 5**: El coche ya no tiene ni pedales ni volante, ya que disponen de la automatización total y no depende del ser humano.

2.4. TwizyLine: estado del proyecto

2.4.1. Proyecto TwizyLine

La cuarta edición del concurso TwizyContest, con el que se comenzó el proyecto TwizyLine, puso el foco en la movilidad de los Juegos Olímpicos de París 2024. La idea no solo era ayudar al transporte de los atletas y periodistas de los Juegos de una manera limpia y eficiente, sino que además se pretendía buscar una revolución en el ámbito de los coches compartidos. Planteando una visión futurista de las ciudades inteligentes en la que no haya problemas de contaminación y de aparcamiento. [2]

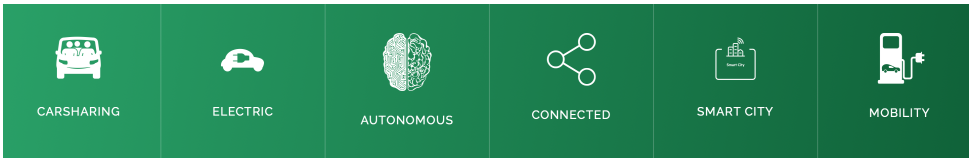


Figura 2.8: Pilares del proyecto TwizyLine

La idea principal presentada en TwizyLine consistía en el diseño de un *parking* inteligente en el que los coches se aparcaran de manera autónoma dentro de él sin necesidad de ningún conductor. El vehículo entra en modo autónomo en el momento que el conductor abandona el vehículo a la entrada del *parking* (*Leaving Zone* 2.9) y este sigue por sí mismo una banda magnética en el suelo hasta su plaza de aparcamiento, en ella se cargará el coche mediante un sistema de carga inalámbrica por inducción. De igual manera, cuándo se requiera los servicios de este coche, se dirigirá de manera autónoma a la salida (*Pick up Zone* 2.9) siguiendo las líneas donde estará esperando el conductor. [17]

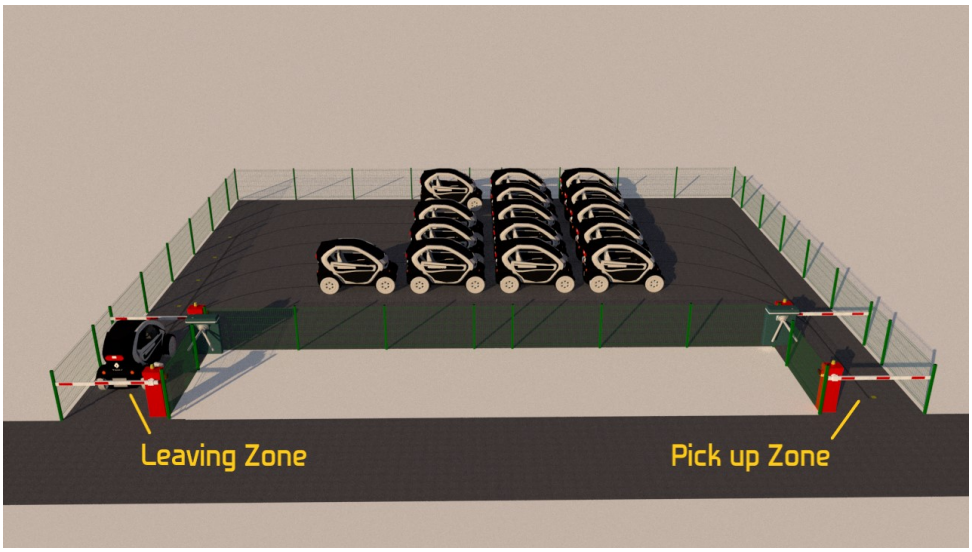


Figura 2.9: Implementación del TwizyLine Parking

Los coches para poder moverse de manera eficiente dentro del *parking* cuenta con un sistema de guiado en un trazado de líneas (representadas en la figura 2.10 como líneas de colores) en las que decide por cuál ir en el punto de decisión:

- **Líneas representadas en verde**: Es la línea de espera de los vehículos que no tienen ningún problema y pueden ser utilizados inmediatamente.
- **Líneas representadas en amarillo**: Tiene como destino la zona de cargadores dónde los vehículos se recargarán por inducción y la zona de reparación.

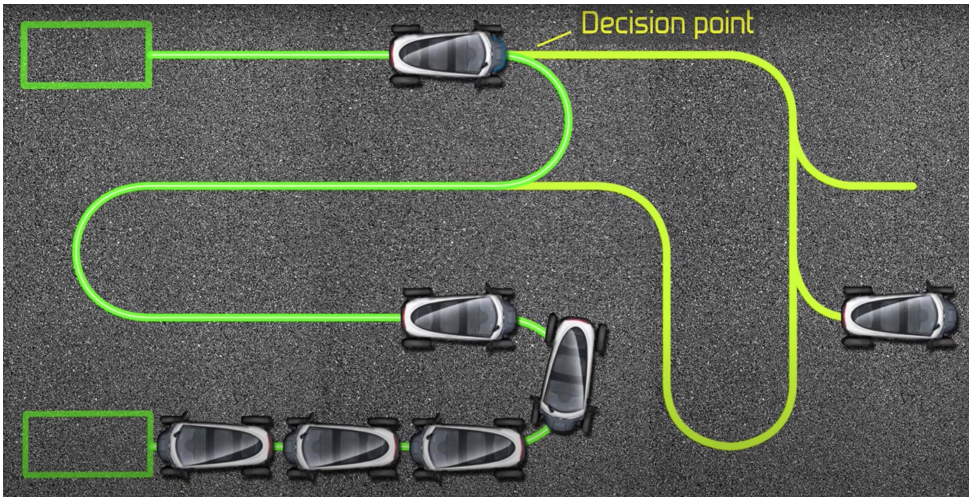


Figura 2.10: Sistema de guiado del vehículo dentro del *parking*

2.4.2. Arquitectura y Software

Para que el Twizy sea autónomo 100 % necesita poder moverse por sí solo, esto incluye tanto el control del movimiento longitudinal como el lateral. Además, requiere de un sistema de guiado o similar que le permita al vehículo saber su posición actual o poder ver hacia dónde tiene que ir. Las soluciones a cada una de estas necesidades han evolucionado con el tiempo. En las siguientes subsecciones se realiza una breve revisión a dicha evolución hasta el estado del vehículo justo antes del inicio de este proyecto.

Control Longitudinal

El control longitudinal solo se ha aplicado sobre el acelerador del vehículo. Hasta el momento no se ha intervenido el freno, debido a que se trata de un sistema mecánico que requeriría añadir un motor de control. Para ser capaces de acelerar el coche, ha sido necesario interceptar el circuito eléctrico que va desde el pedal del acelerador hasta el motor del vehículo. Desde el origen del proyecto se instaló un circuito electrónico controlado por un Arduino que era capaz de habilitar dos estados posibles: control manual del acelerador, dejando el circuito electrónico original del vehículo en su configuración original, y control electrónico del acelerador, en el que el Arduino realiza un bypass del pedal mecánico y controlando dos potenciómetros en el circuito electrónico diseñado simula la presión del pedal mecánico. A fecha de inicio del este proyecto existían dos versiones del circuito controlador del acelerador. Este punto se verá con más detenimiento en el apartado 4.3. Además, dentro del control longitudinal también se ha desarrollado un dispositivo para el control de las marchas automático del vehículo. Este dispositivo, aunque estaba originalmente previsto, fue una adición posterior al prototipo original de TwizyLine y no ha sido convenientemente probado en el vehículo. Este punto se verá con más detenimiento en el apartado 4.2.

Control Lateral

Uno de los principales problemas de vehículo Twizy es que no cuenta con dirección asistida. Esto significa que no era posible desarrollar una solución software que utilizará dicha asistencia para el control lateral. Por ello, desde el primer prototipo se optó por una solución que se basa en actuar sobre la columna de dirección del coche [6] con un motor de la compañía Maxon (ver Figura 2.11, sacada de [6]), en concreto el **Maxon Motor EC60**, junto con la controladora **EPOS4**, que será con la que nos comuniquemos para hacer trabajar el motor de Maxon y así poder girar el volante sin necesidad de intervención humana.

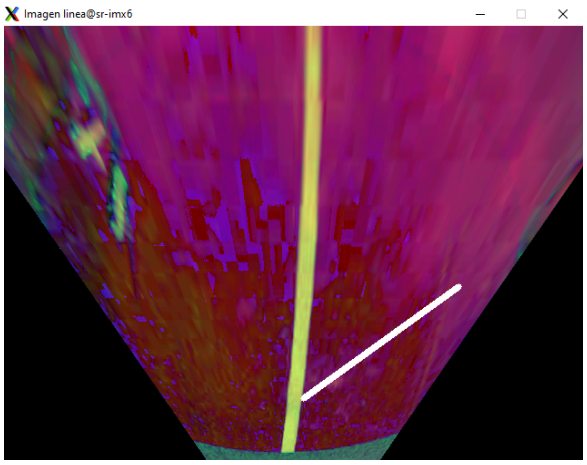
Figura 2.11: Implementación del conjunto del motor, los engranajes y el soporte en el Twizy.

Para acoplar el motor Maxon a la columna de dirección del vehículo se añadió una rueda dentada sujeta a dicha columna que transmiten directamente la potencia del giro del motor Maxon al eje de la dirección y así poder girar las ruedas. La controladora EPOS4 es la que recibe las órdenes del módulo de control y hace trabajar al motor de control lateral en consecuencia.

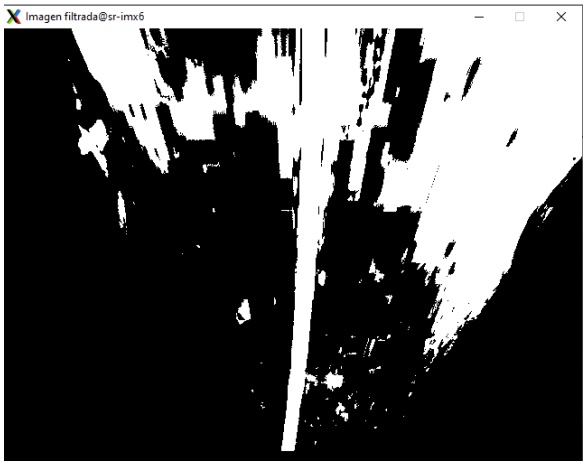
Guiado

El guiado original del vehículo estaba previsto realizarlo mediante unas bandas con propiedades magnéticas que, junto con el sensor magnético en la parte delantera del vehículo, permitían que se pudiera seguir dicha línea. Este sensor era capaz de detectar la distancia respecto al centro de la banda y así poder determinar hacia dónde y cuánto girar el volante. Los resultados de control no fueron lo suficientemente estables, dado que los requerimientos de control eran excesivamente estrictos (el vehículo no podía desplazarse más de 7 cm a izquierda o derecha de la línea magnética) por lo que finalmente se optó por buscar una alternativa al sensor magnético.

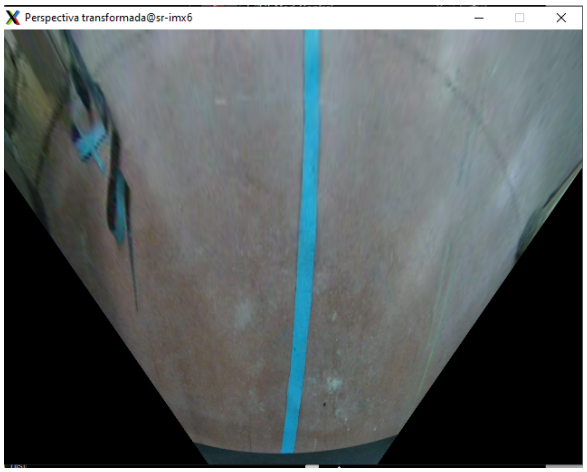
Finalmente, se decidió implementar un sistema de reconocimiento de imágenes que es capaz de reconocer una línea a seguir gracias a una cámara que se ha acoplado en la parte delantera del vehículo, junto a la tapa de carga del coche. Esta cámara apunta al suelo y, junto con un procesamiento de las imágenes capturadas por esta, es capaz de reconocer la línea y determinar la distancia respecto al centro de la misma. Las bandas magnéticas se sustituyeron por unas bandas de goma pintadas en color azul para que sean fácilmente reconocidas por la cámara. En la figura 2.12 se muestra el proceso que tiene una imagen capturada por la cámara, este proceso consiste en una transformación de la



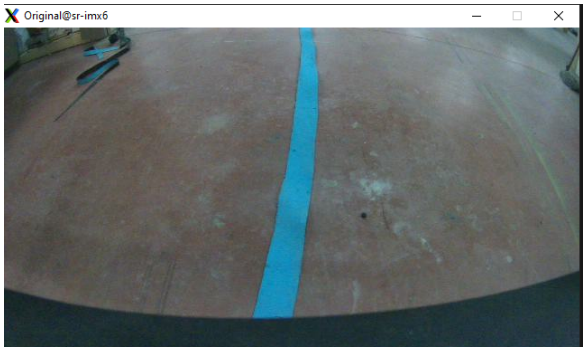
(a) Imagen procesada que muestra la línea a seguir



(b) Imagen filtrada



(c) Imagen transformada



(d) Imagen real capturada por la cámara

Figura 2.12: Conjunto de imágenes del proceso de guiado

perspectiva y un filtro que resalte el color azul de la banda del suelo.

Para que el vehículo conociera mejor el entorno, se añadieron un *array* de sensores de ultrasonidos, tres en la parte delantera, concretamente: uno en el centro y los otros dos en los extremos laterales. Estos sensores permiten al coche detectar obstáculos que se encuentren delante del vehículo y así poder evitar posibles atropellos cuando este vaya sin conductor.

El trabajo realizado por Carlota Gómez consistía también en el uso del LiDAR como sistema de guiado. La idea principal era disponer de un mapa 3D capturado previamente con el LiDAR con el objetivo de ser capaz de ubicar el vehículo en ese mapa a partir de la nube de puntos capturada en tiempo real. Este método no se llegó a incorporar debido a la dificultad de alimentar la librería de partida con las señales que solicitaba y a la dificultad de la localización del vehículo a partir de un mapa pregrabado, cuestión que el *software* con el que se trabajaba no resolvía.[5]

Procesado

Para hacer posible todo lo mencionado en este apartado, se necesitaba de un ordenador que recibiera todos los *inputs* capturados por los sensores y tomara decisiones al respecto. Para esto, se optó por el uso de la microcontroladora HummingBoard, que es similar a una Raspberry Pi. Esta es capaz de procesar la información recibida por los sensores, determinar si estamos siguiendo correctamente la línea o, por el contrario, si nos estamos desviando, y corregir el fallo. Además, puede detectar si hay

un obstáculo cerca gracias a los sensores de ultrasonidos y detener el coche si es necesario.

Para ejecutar las órdenes, se comunica con los Arduinos, tanto para recibir la información de los sensores como para actuar sobre el acelerador. También se comunica directamente con la controladora EPOS4 mediante USB para girar el coche.

Control Remoto

Una de las ideas de este proyecto era que el cliente pudiese solicitar el servicio del Twizy mediante su aplicación, por lo que se necesitaba que el coche estuviera conectado a Internet para que pudiera recibir las órdenes del servidor que se comunicaba con el *smartphone* del cliente. Por esto, se necesitaba de un módem 4G y de un servidor MQTT para la comunicación Vehículo-Smartphone. Para la optimización de recursos y un mejor aprovechamiento, se optó por incluir una nueva HummingBoard (Módulo de Comunicaciones) que se encargara de todo lo relacionado con las comunicaciones externas y actuara como intermediario entre el módulo de control e Internet.

A este módulo se le conectarán el módem 4G y el receptor GPS vía USB, mientras que, por otro lado, el módulo estará conectado a un servidor MQTT por el que recibirá las órdenes externas para actuar en consecuencia. El servidor MQTT estaba alojado en una Raspberry Pi situada fuera del vehículo, pero actualmente el servidor se ejecuta en un PC externo dentro de la misma subred del coche, como se detallará más adelante.

Cableado

El conexionado del vehículo en el momento de partida de este proyecto se tratará con una mayor profundidad en el capítulo siguiente.

Capítulo 3

Conexionado eléctrico y de comunicaciones

3.1. Introducción

Uno de los objetivos de este proyecto es realizar una reestructuración del cableado del vehículo. Esta reestructuración es necesaria debido a la escasa eficacia que la disposición actual de bastantes de sus componentes tenían. En general, aunque se encontraban dispuestos en huecos y lugares cercanos a donde se tenía que transmitir su señal, resultaba difícil manipularlos cuando se tenía que acceder a ellos para, por ejemplo, cambiar su configuración o para realizar cualquier tipo de comprobación. Además, debido a la gran cantidad de dispositivos que se han ido añadiendo, era especialmente complicado saber qué cables eran de alimentación, cuáles de envío de señales e incluso saber a qué dispositivos correspondían. Por ello, se ha optado por agrupar todos los componentes posibles en la parte trasera del vehículo, en concreto se ubican en un *rack* hecho a medida dispuesto en el asiento del copiloto, salvo algún componente como la EPOS4, que se encuentra en la parte delantera porque para obtener un rendimiento óptimo necesita estar cerca del motor de control lateral. Además, se ha utilizado un código de colores para identificar la función de cada cable y se han ordenado para que no estén a la vista y sea más difícil provocar fallo de manera no intencionada.

3.2. Estado Inicial

En las siguientes subsecciones se explica el estado del vehículo justo antes de hacer las modificaciones para que se entienda mejor el alcance de las mismas.

3.2.1. Parte delantera

En la parte delantera nos encontramos con los sensores de ultrasonidos, magnético y RFID. Estos sensores estaban conectados junto con el controlador del acelerador a un *hub* USB que agrupaba estos dispositivos y se conectaba al módulo de control mediante un cable USB. Este *hub* (figura 3.1) tenía botones para encender o apagar cada puerto de manera independiente, ya que el programa principal requería de una secuencia determinada de encendido de estos puertos para su correcto funcionamiento. Su ubicación era la guantera derecha junto con la controladora EPOS4 y la Humming Board del módulo de control, concentrando una gran cantidad de cables que puedan provocar problemas de conexionado en esa zona ante la más mínima manipulación.

En la guantera izquierda se encontraba el puerto OBD del coche del que se obtenían los 12 V de alimentación y la conexión al bus CAN del vehículo. Este puerto estaba conectado a un duplicador que nos permitía sacar alimentación y el bus CAN para cada Humming de manera independiente, como se muestra en la figura 3.5. Además, la controladora EPOS4 estaba alimentada directamente por la batería de servicio del Twizy con unos conectores que permitían la conexión y desconexión de la controladora. [5]

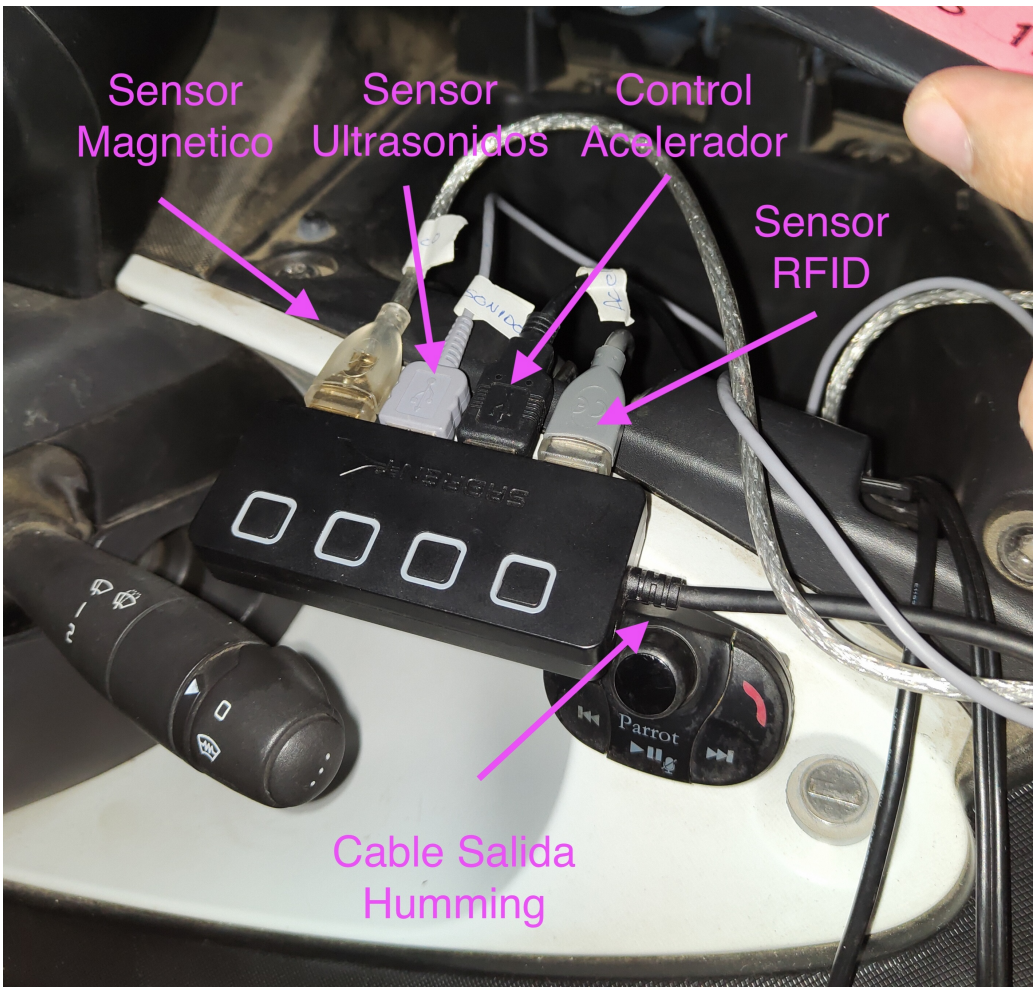


Figura 3.1: *hub* USB inicial

3.2.2. Parte trasera

Por otro lado, en la parte trasera se encontraba todo lo relacionado con las comunicaciones. Había un *rack* de aluminio con 4 niveles en el que estaban situados: el módulo de comunicación, al que se le conectaba un módem 4G y el GPS. También se encontraba el PC Máster que se encarga de realizar todas las operaciones con los datos obtenidos por el LiDAR y por último, había un inversor que transforma la corriente continua de la batería del vehículo en corriente alterna a 220 V para alimentar el PC Máster.

Tanto la parte delantera como la trasera estaban conectadas y alimentadas mediante cables que recorrían el coche por el lateral izquierdo del suelo del vehículo. Además, debido al estado precario de las conexiones y la disposición de los cables por el suelo del vehículo ocurrían problemas de conexionado y resultaba difícil de detectar dónde se producía el error. Por estos motivos, es por lo que se decidió hacer un nuevo planteamiento del conexionado eléctrico y de la ubicación de los componentes.

3.2.3. Comunicaciones

La forma de comunicarse con los módulos de control y de comunicaciones desde otro ordenador era mediante una comunicación RS232 a través de los pines que salían de la Humming Board. Estos pines, además de necesitar de cables y alargadores que suponen más incomodidad y puntos de fallo posibles, requerían de un adaptador RS232-USB para poder conectarlo al ordenador y mediante un terminal serie poder interactuar con él.

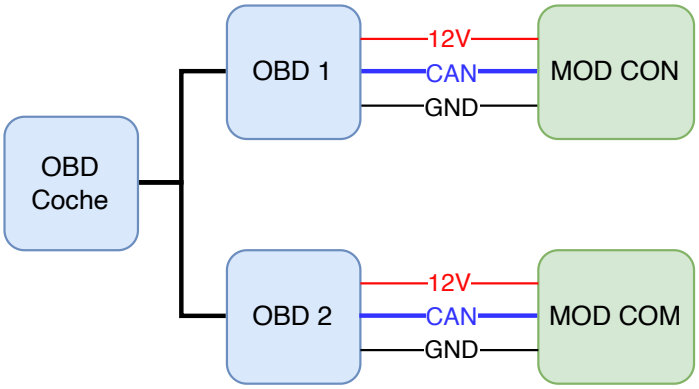


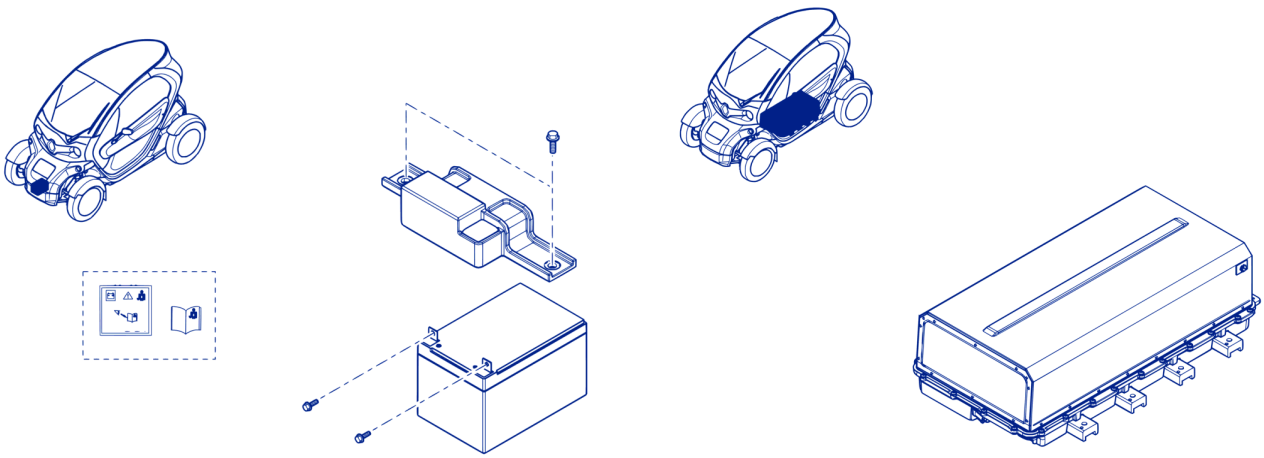
Figura 3.2: Esquema de la disposición original del OBD

Además, para el funcionamiento del vehículo en modo autónomo se necesita un *router* y un servidor *mosquitto* por dónde se reciben las órdenes del coche que se envían desde un móvil mediante el uso de tópicos en mensajes MQTT [7]. Este *router* se encontraba fuera del vehículo, así como el servidor, que estaba alojado en una Raspberry Pi en un laboratorio de la Universidad de Valladolid. El módulo de comunicaciones se conectaba al servidor mediante el módem 4G que le permitía tener acceso a internet.

3.2.4. Alimentación

En cuanto a la alimentación del vehículo tenemos que saber que los vehículos eléctricos disponen de dos baterías.

La primera de ellas es la batería de servicio, que es la que se incluye en todos los coches convencionales, sean eléctricos o de combustión. Esta batería es la que se encarga de suministrar electricidad a todos los sistemas del vehículo, como puede ser el motor de arranque o el sistema de alarma del vehículo. Este tipo de baterías suelen situarse en la parte delantera del vehículo justo debajo del capó, ver figura 3.3a, además de tener una tensión alrededor de 12 V y se pueden recargar mediante el alternador o la inercia de las ruedas. [18]



(a) Batería de servicio

(b) Batería de tracción

Figura 3.3: Baterías en Renault Twizy

Por otro lado, tenemos la batería de tracción del motor eléctrico del coche, esta suministra la energía necesaria al motor eléctrico para que este mueva el coche. En los vehículos modernos, son baterías mucho más grandes y pesadas, llegando a pesar hasta 720 Kg para una autonomía de 500 Km. Además de tener un voltaje que varía entre los 200 V y los 400 V dependiendo de la autonomía y la potencia del motor. Se sitúan en la parte inferior del coche y se pueden recargar mediante la frenada regenerativa o la propia inercia del vehículo. En concreto, el Renault Twizy tiene una batería de 46 Kg de peso y 58 V, además, tiene una capacidad de 6.1 kWh y se sitúa en la parte inferior como se puede ver en la figura 3.3b. [19]

3.3. Modificaciones

3.3.1. Parte delantera

En la parte delantera se ha prescindido del sensor magnético cuyo funcionamiento no cumplía con los requisitos establecidos. En cambio, se han mantenido los sistemas de ultrasonidos y RFID que estaban en la parte delantera para captar la información del entorno. Además, la EPOS4 también se ha mantenido en la parte delantera porque necesita estar en una posición relativamente cercana al motor de control lateral para evitar posibles retardos en las comunicaciones y problemas relacionados con la calidad de la señal que afecten al rendimiento.

Primera instancia

Como se ha visto anteriormente, estos sensores están conectados a un *hub* USB junto con el controlador del acelerador. A este *hub* se le ha añadido el controlador de marchas, que se va a ubicar junto a la rodilla izquierda (figura 3.4) y se fijará con velcro al guarnecido del vehículo. La salida del *hub* se encontraba en la guantera derecha junto al módulo de control, pero debido a que este último se ha movido a la parte trasera necesitamos un alargador USB que cruza el frontal del coche y llega hasta la parte trasera por el lateral izquierdo, ya que por ahí es donde se van a aglutinar todos los cables que vayan desde la parte delantera a la trasera del vehículo.

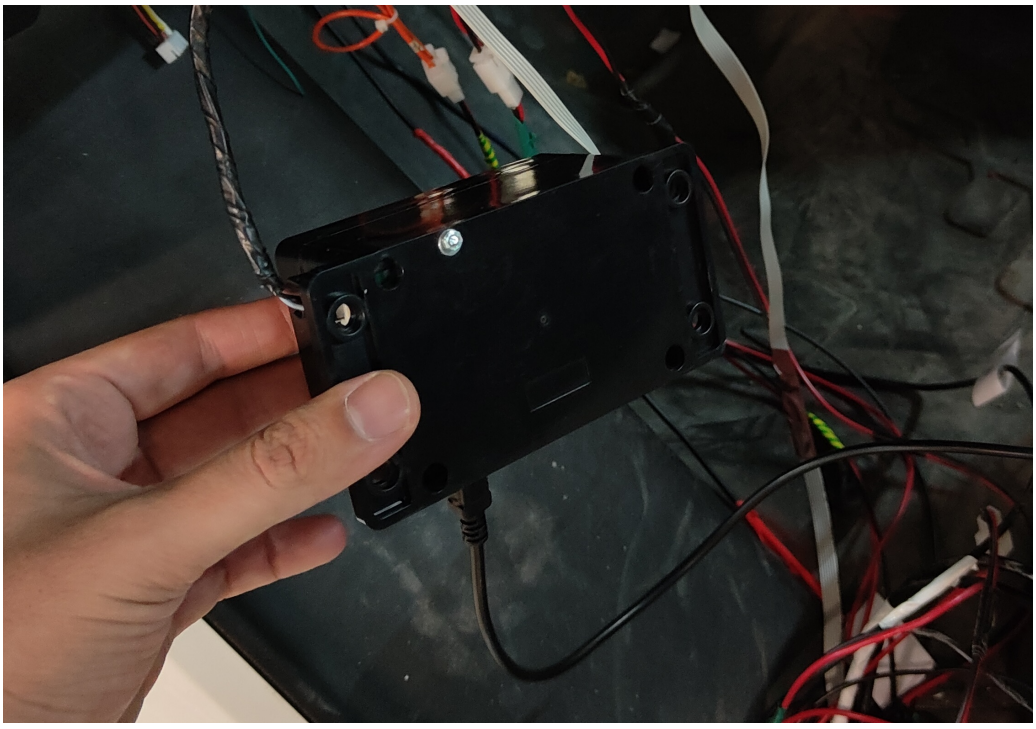


Figura 3.4: Controlador de marchas

Por último, el puerto OBD del coche, a diferencia de lo que se hacía antes, únicamente se va a utilizar para conectarse al bus CAN, ya que los 12 V se van a obtener directamente de la batería de servicio. Por lo que se ha fabricado un nuevo cable que va desde el OBD hasta la parte trasera donde se encuentra la SpyBox CAN. Este cable tiene un conector OBD del que obtenemos los pines necesarios para el bus CAN como son CANL, CANH y GND [5]. En el otro extremo se han soldado estos buses a un conector DB9 que se conectará al *hub* CAN.

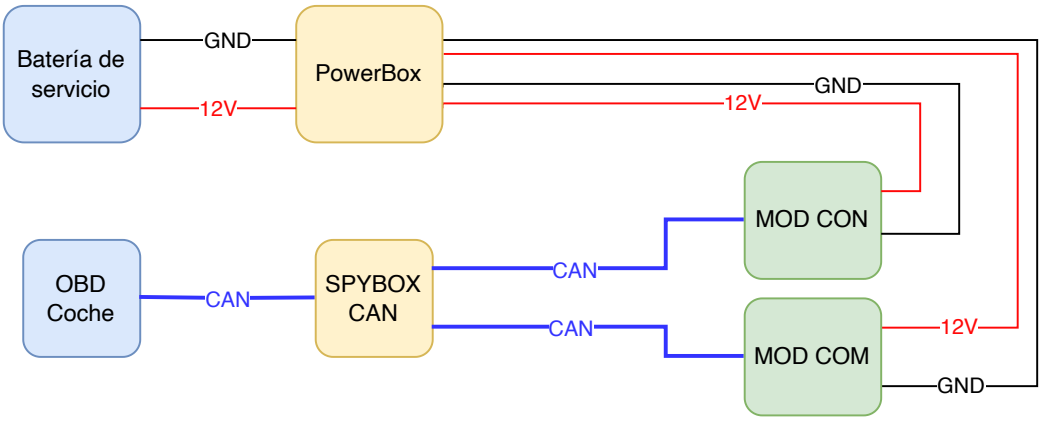


Figura 3.5: Esquema final de la disposición del OBD

El SpyBox CAN es un concentrador de buses CAN, como se ha hablado antes en el trabajo de Carlota Gómez [5], que sirve para facilitar las conexiones entre diferentes dispositivos a un mismo bus, así como para conectar un osciloscopio para visualizar las comunicaciones del bus. En nuestro proyecto conectaremos a esta SpyBox los módulos de comunicación y de control. Además, se ha añadido una resistencia de terminación de 120Ω entre los buses CANH y CANL para evitar que se produzcan interferencias en el bus.

Figura 3.6: Imagen frontal

Una vez aplicados estos cambios en la parte delantera del vehículo, se ha podido colocar las tapas de las guanteras que antes no cerraban, y ha permitido colocar la seta de emergencia encima de una de ellas, en concreto, la que se encuentra a la derecha del conductor.

3.3.2. Parte trasera

Debido a la necesidad de poner una caja de fusibles para proteger algunos de los equipos, se ha decidido diseñar una caja (**PowerBox**) en la que integre esta caja de fusibles junto con interruptores en la tapa, así como en los laterales los diferentes conectores a los que se enchufarán los equipos. Además, con el objetivo de ser capaces de medir el consumo total de todo el sistema, se optó por añadir el amperímetro descrito en [5] cuya entrada es la entrada de alimentación del vehículo y su salida es la que alimenta la caja de fusibles, como se ve en la figura 3.8. Por último, para evitar que este amperímetro descargue la batería continuamente, se ha decidido incluir un interruptor a mayores que actúe cómo **interruptor general** de toda la alimentación del sistema.

Figura 3.7: PowerBox

Cabe destacar que por razones de seguridad se han evitado poner conectores macho en los cables que tengan corriente para evitar posibles descargas que dañen los equipos o a las personas. En la figura 3.7 se muestra el resultado final de la construcción de la PowerBox con los botones y las tomas disponibles para conectar los diferentes dispositivos.

De la batería de servicio vamos a sacar los 12 V en continua que irán directamente a la entrada de la PowerBox que, entre la batería y la PowerBox colocaremos el amperímetro que nos permitirá conocer en tiempo real el consumo total de los equipos, así como llevar un registro de consumo. En la PowerBox tenemos la posibilidad de conectar hasta 6 dispositivos, pero actualmente solo se han conectado los módulos de control y comunicaciones, así como el *router* y la controladora EPOS4. Esta última es una de las adiciones más importantes a la PowerBox, dado que anteriormente se alimentaba con un cable directamente desde la batería de servicio y, dado que no tiene botón de encendido o apagado, la descargaba excepto si se desenganchaban los cables, lo que siempre era una acción susceptible a provocar fallos.

Respecto al módulo de control, se comunicará mediante un cable USB con la controladora EPOS4, así como con otro cable con el resto de los dispositivos que se encuentran conectados al *hub* USB de la guantera derecha. En este *hub* estarán conectados los sensores de RFID y Ultrasonidos, los controladores de marchas y del acelerador y por último, la cámara que es la que detecta la línea azul del

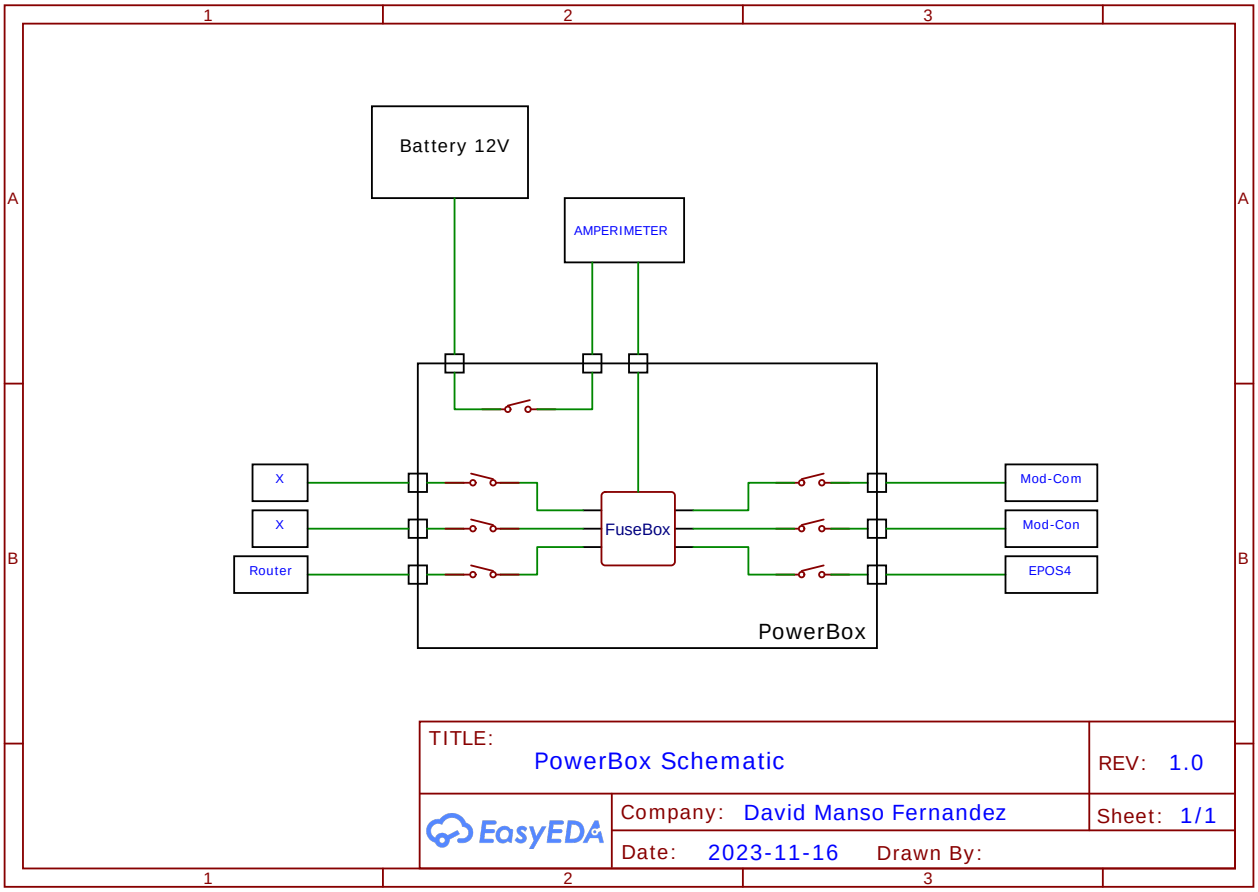


Figura 3.8: Esquema eléctrico PowerBox

suelo que ha de seguir.

3.3.3. Comunicaciones

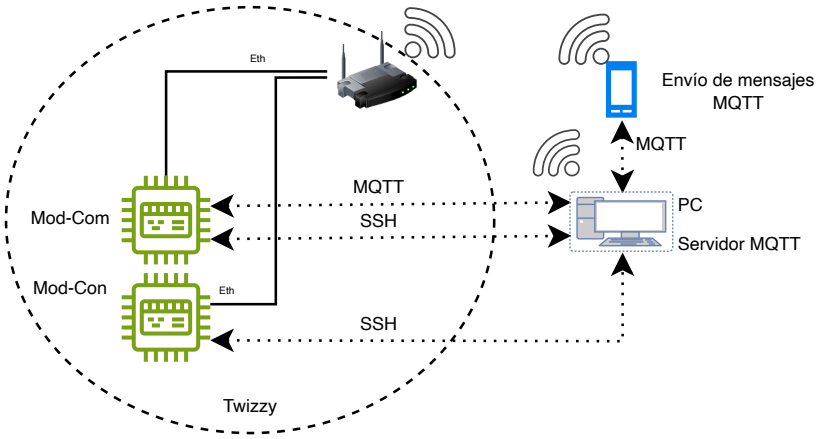


Figura 3.9: Esquema de la red local

En cuánto a las comunicaciones, con el objetivo de evitar la comunicación RS232 con los módulos de control, se ha decidido realizar esta conexión mediante el protocolo SSH. Este protocolo nos ofrece

la posibilidad de conectarnos a los módulos sin ningún cable, solamente es necesario estar en la misma red local. Para que esto sea posible se ha incluido en el *rack* trasero del vehículo el *router*, que se alimentará de los 220 V que nos da el inversor de corriente DC/AC. Además, ambos módulos están conectados mediante cables Ethernet al *router* que, junto con mi ordenador y un smartphone conectado vía Wifi, disponemos de todos los elementos necesarios en la misma red local para poder comunicarnos con el vehículo y enviarle los mensajes MQTT. En la figura 3.9 se representa esta red local.

3.4. Estado Final

El estado final del vehículo se detallará a continuación y su esquema se representa en la figura 3.11.

3.4.1. Parte delantera

Finalmente, debido a diferentes contratiempos (descritos a continuación), se ha optado por conectar los componentes de la siguiente manera:

- Al *hub* USB se conectarán los sensores de ultrasonidos y RFID, así cómo la cámara y el controlador de marchas.
- El controlador del acelerador se conectará directamente al módulo de control debido a que al pasar por el *hub* se produce una caída de alimentación, provocando que seamos incapaces de poder activar los relés que están en el interior del controlador. Este problema no nos permitía hacer el *bypass* entre el pedal y la señal de salida del Arduino del controlador. En la figura 3.10 se muestra la disposición final del controlador en el vehículo.

Figura 3.10: Disposición final del controlador del acelerador

- El cable USB de comunicación de la EPOS4 va directamente conectado al módulo de control como ya hacía anteriormente.
- La EPOS4 se alimentará finalmente desde la batería de servicio, como se hacía anteriormente. Esto es debido a que la controladora necesita una determinada alimentación y al querer introducir un interruptor entre medias para poder controlar el encendido y apagado, este introducía una caída de tensión que provocaba que la EPOS4 entrase en modo de fallo por error en las condiciones de alimentación. Este tema se explicará más adelante en el apartado 4.4.3.

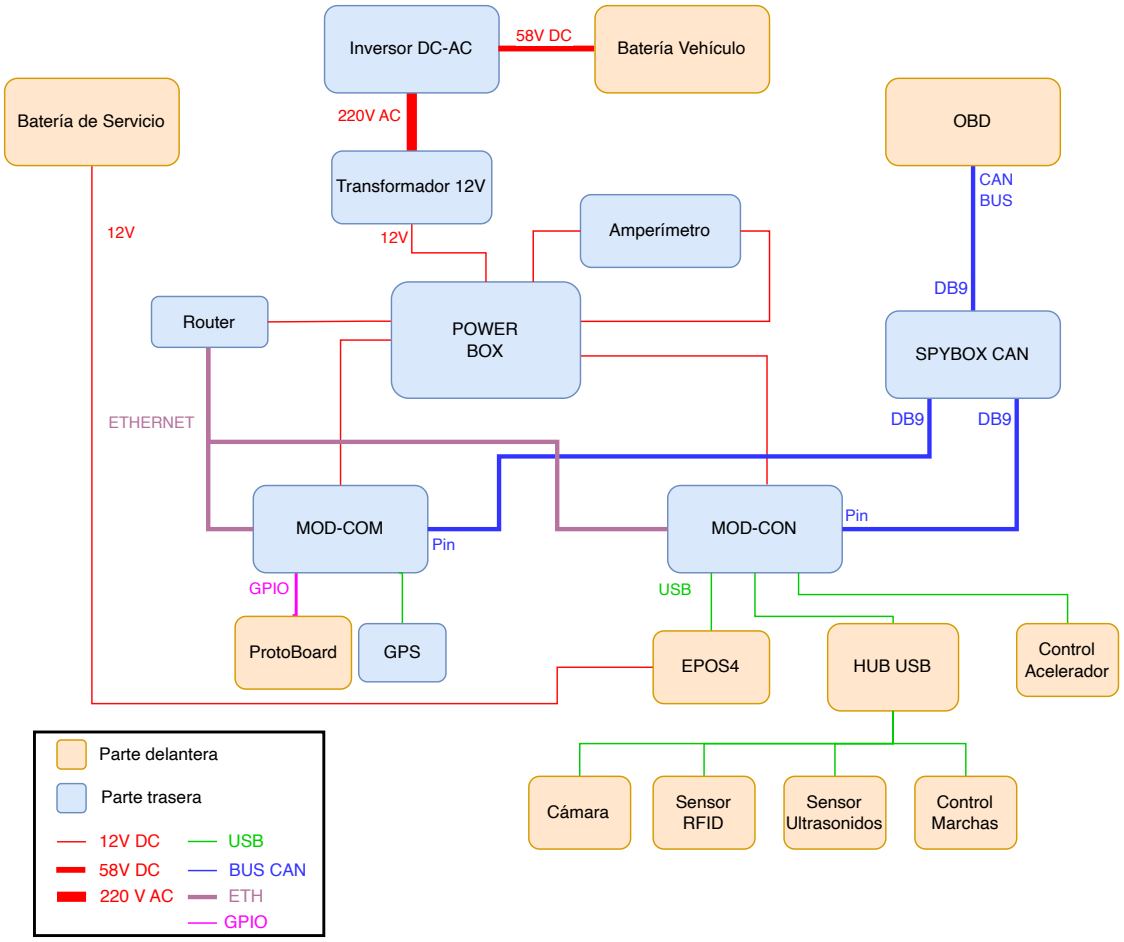


Figura 3.11: Esquema implementado del cableado del vehículo

3.4.2. Parte trasera

Debido al problema con la alimentación de la EPOS4 que veremos en la sección 4.4.3, los 12 V procedentes de la batería de servicio se van a destinar a la EPOS4, mientras que para alimentar el resto de componentes conectados a la PowerBox se va a utilizar el inversor de corriente alterna que estaba en el *rack* del coche. Inicialmente, este inversor está pensado para alimentar el PC Fusión en implementaciones futuras, pero de momento se va utilizar para alimentar la caja de conexiones. La forma de hacerlo es conectando un transformador de 12 V y 1 A a la salida del inversor de 220 V para obtener corriente continua y alimentar la PowerBox.

En la figura 3.12 se muestra la disposición de las Humming junto con la PowerBox en el nivel inferior del *rack*. A esta disposición habría que añadirle el *hub* como veremos en la figura 3.13.

Inicialmente, el *router* se alimentaba directamente del inversor con el transformador, como si se enchufase a la red eléctrica. Pero ya que había tomas de conexión restantes en la PowerBox se decidió conectarlo directamente el *router* a esta PowerBox para tener todos los dispositivos electrónicos en una misma caja de conexiones y de paso, obtener la medición completa del consumo de todos ellos.

En cuanto a la batería de tracción del vehículo, se le conectará el inversor de DC-AC para convertir los 400 V de corriente continua de la batería a 220 V en corriente alterna. Este inversor alimentará al PC máster en una futura implementación. Del *router* saldrán dos cables Ethernet que se conectan con las Humming Board, el módulo de control y el de comunicaciones, creando una red local cableada.

Figura 3.12: Disposición final de los elementos en el *rack*

El módulo de comunicaciones, mediante los pines GPIO que tiene en su interior, enciende y cambia de color los led que se encuentran encima del volante, con el objetivo de tener una indicación visual del modo en el que está el vehículo. [7] Además se le conectará el cable USB procedente del GPS que se situara en el techo del vehículo.

Figura 3.13: Estado final del *rack* junto con el *router* y el inversor

Por último, el bus CAN parte del puerto OBD del vehículo que se sitúa en la guantera izquierda. De aquí sacamos 3 cables que contienen los buses CANL, CANH y GND necesarios para la comunicación CAN. Estos cables llegan a un *hub* CAN, que se situará justo detrás del asiento del piloto de manera vertical, mediante el conector DB9 y de este *hub*, saldrán dos buses que conectarán con ambas Humming Board. Teniendo una topología de bus al que están conectados los módulos de control y comunicaciones junto con todas las comunicaciones CAN del vehículo. Para tener una visión más completa de la nueva implementación podemos ver en la figura 3.13 como están colocados los componentes en el *rack*, así como el *router* en la balda superior y el espacio libre destinado al PC fusión.

A modo de resumen cabe destacar que se ha pasado de un sistema cuyo cableado era muy frágil,

propenso a fallos y desordenado a un cableado que resulta fácil de trazar, mucho mejor desplegado evitando fallos y además se tiene una caja de conexiones con la que poder controlar los dispositivos que están encendidos, aportando un mejor control del consumo en el vehículo y un mejor control de qué dispositivos queremos que funcionen dependiendo de la prueba a realizar. El esquema eléctrico objetivo que se quiere conseguir en futuras implementaciones es el mostrado a continuación en la figura 3.14. Para lograr este esquema se necesitarán interruptores adecuados para la PowerBox como se explicará en capítulos siguientes, en concreto en el apartado 4.4.3.

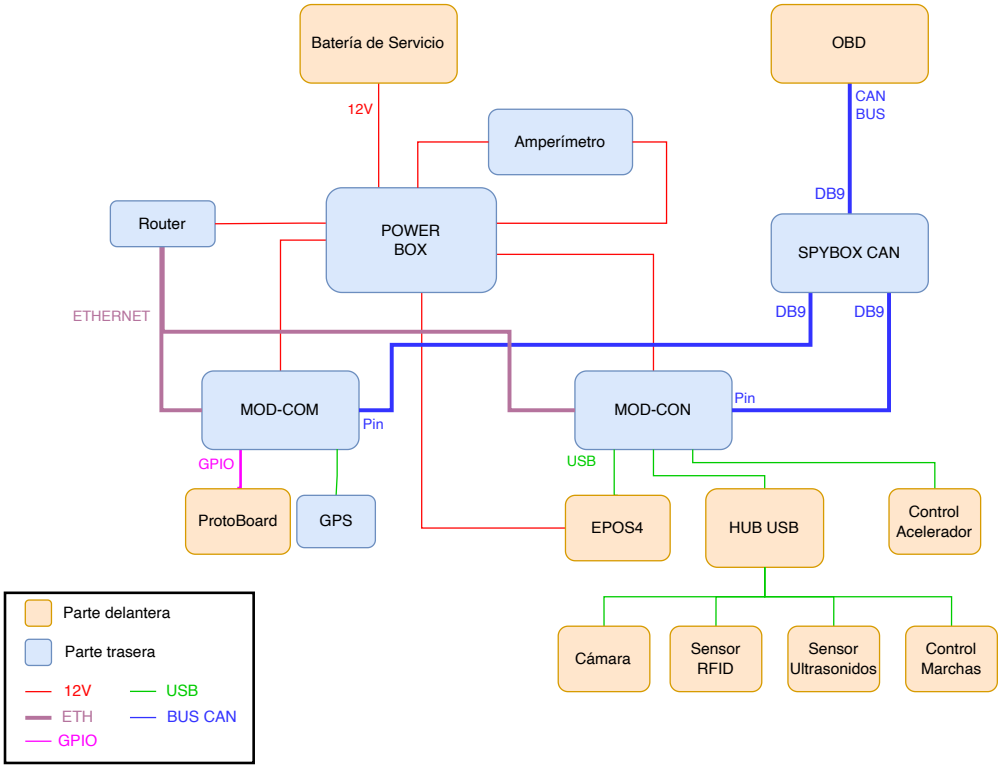


Figura 3.14: Esquema final objetivo del cableado del vehículo

Capítulo 4

Controladores

4.1. Introducción

En este capítulo vamos a ver los diferentes controladores que se han tenido que modificar para poder cumplir los requisitos del proyecto. Para poder cambiar la marcha del coche de manera autónoma se ha continuado con el proyecto de Mohammad Kanaan [8] que implementó un circuito controlador de marchas y se ha desarrollado el software de control e instalado en el vehículo. Para el control longitudinal se esperaba partir del circuito desarrollado también en el proyecto de Mohammad Kanaan, pero debido a que no había dejado un programa de control que explicará el funcionamiento del circuito y que, además, no parecía funcionar adecuadamente, se ha tenido que desarrollar uno nuevo desde cero.

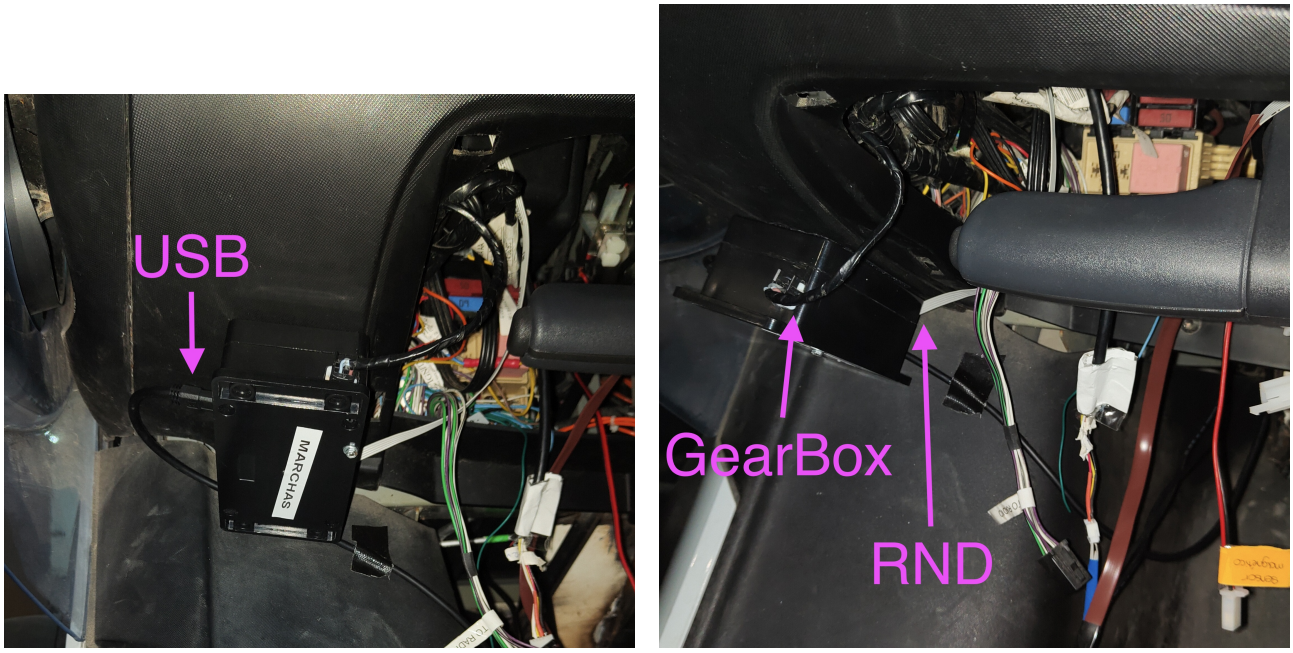
Por último, para el control lateral se ha trabajado con la controladora EPOS4 de la compañía Maxon Motor, que como se ha mencionado anteriormente, gracias a un motor y unos engranajes, permite girar la columna de dirección y controlar el giro de las ruedas. En este caso se decidió cambiar el modo en el que la controladora realizaba el giro de las ruedas, cambiando del modo de posición al modo de velocidad, como veremos más adelante, con el objetivo de poder girar las ruedas a una mayor velocidad lo que nos permita aumentar la velocidad del vehículo y seguir la línea con una mayor facilidad.

4.2. Control de la Caja de Cambios

Uno de los objetivos de este proyecto consiste en la integración de un controlador de marchas automático, el cual ha sido diseñado y fabricado en el proyecto de Mohammad Kanaan [8]. Este controlador, al igual que ocurrirá después con el acelerador, permite trabajar en dos modos. Por un lado, el modo manual, en el que el vehículo funciona como si no se hubiera incorporado un nuevo controlador y, por otro lado, el modo automático, en el que el controlador de marchas instalado realiza un *bypass* del control manual anulándolo y pasando a ser controlado por el nuevo controlador, tal y como se explica en la figura **??**. Este coche tiene una caja de cambios **RND** que significa que las marchas disponibles son únicamente **Drive**, **Reverse** y **Neutral**. El objetivo de este controlador aparte de los mencionados anteriormente es que sea capaz de permitir la conducción autónoma, introduciendo la marcha requerida mediante el módulo de control, así cómo la conducción manual por parte de un conductor, habilitando el cambio de marcha mediante los botones RND que el coche trae de serie, ver figura 4.1.

Figura 4.1: Botones RND del Renault Twizy

En cuanto a la instalación del controlador, se ha optado por ubicarlo junto a la guantera izquierda, mirando hacia abajo y sujetado por velcro, como se muestra en la figura 4.2. Por su parte, la botonera está conectada al controlador del motor de tracción del vehículo a través de un conector situado detrás de ella. El controlador se va a colocar al medio de este conector a través de sus dos salidas, una que se conecta con el controlador del motor de tracción y la otra que conecta con la botonera RND. Además, se tiene un cable USB que conecta con la Humming.



(a) Instalación frontal del controlador de marchas (b) Instalación lateral del controlador de marchas

Figura 4.2: Instalación del controlador de marchas

Por ejemplo, la idea principal en los inicios del proyecto es que llegue el vehículo conducido por un conductor a la entrada de un *parking*, lugar en el que el conductor abandona el vehículo y este se pondría en modo autónomo para dirigirse a su lugar de aparcamiento realizando los cambios de marchas necesarios electrónicamente.

Otra utilidad del control de marchas que se ha planteado es que a la hora de frenar el vehículo,

dado que el vehículo no tiene un sistema de freno controlable electrónicamente, se introduzca la marcha **R** y acelerar para reducir la velocidad. Esto requiere que la caja de cambios del coche cambie de manera autónoma.

Para lograr esto se utilizará el controlador de marchas diseñado por un Mohammad Kanaan [8] que integra un *Arduino Nano*, microcontrolador encargado de habilitar las señales pertinentes para realizar el control del cambio de marchas. Kanaan no dejó ninguna instrucción o programa que indicara cómo funcionaba el controlador, así que hubo que analizar el mismo y se programó un nuevo código para pasar de modo manual a autónomo y para cambiar las marchas en función de las necesidades que requiera el programa principal del vehículo.

El controlador está conectado con el módulo de control mediante USB y este es el encargado de decidir cómo tiene que actuar enviándole comandos por el bus serie. En función de los comandos recibidos, el controlador puede actuar de la siguiente forma:

Comando	**Función**
Manual	Control Manual
Automatic	Control Automático
Go	Marcha: D
Reverse	Marcha: R
Neutral	Marcha: N

Tabla 4.1: Tabla de órdenes y funcionamiento.

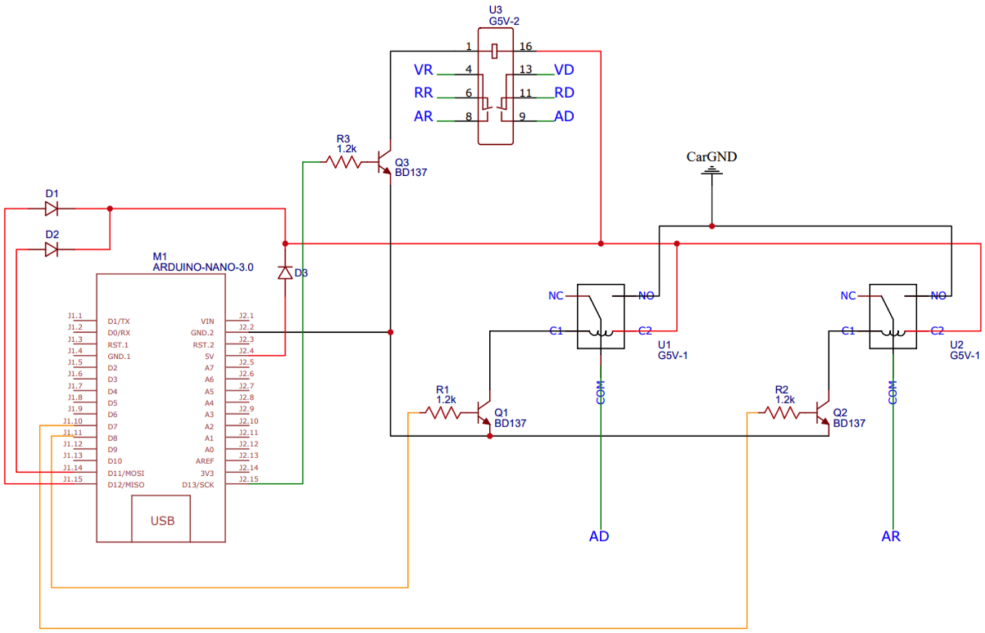


Figura 4.3: Circuito eléctrico del controlador de marchas [8]

La circuitería interna del controlador de marchas se basa en un *Arduino Nano* que controlará los estados y las acciones a través de los pines **D13**, **D7** y **D8**. En concreto, el pin D13 actúa sobre el relé U3, mostrado en la figura 4.3, que en función del voltaje que reciba por el pin 1 provoca que el controlador actúe como pasarela o tenga el control sobre las marchas del vehículo. Los pines D7 y D8

son los que actúan sobre el resto del circuito que representaría el actuador, permitiendo el *bypass* de la botonera para poner el vehículo en modo autónomo y el control de las señales eléctricas que recibe la caja de cambios para cambiar la marcha. La tabla lógica de estos pines es la siguiente:

Pin 13	Pin 7	Pin 8	Modo
0	X	X	Manual
1	0	0	Autónomo / Mantiene marcha
1	0	1	D
1	1	0	R
1	1	1	N

Tabla 4.2: Tabla lógica de modos de funcionamiento.

La idea es que si el pin D13 está en baja, el controlador actúe como pasarela entre lo que entra por los botones del vehículo y la caja de cambios. Mientras que si está a '1' la señal que entra al vehículo es la que se envía por el *Arduino* en función de lo que reciba del módulo de control.

4.3. Control Longitudinal

El sistema del pedal del acelerador en un vehículo, tanto de combustión como eléctrico, controla la cantidad de combustible o energía eléctrica que se envía al motor para aumentar la velocidad. Cuando el conductor presiona el pedal del acelerador, se envía una señal al sistema de gestión del motor, que interpreta esta entrada y ajusta la cantidad de combustible o energía eléctrica suministrada al motor, aumentando así la velocidad del vehículo. En un vehículo de combustión interna, esta señal se traduce en la apertura de la mariposa del acelerador, permitiendo que entre más aire y combustible al motor. En un vehículo eléctrico, la señal del pedal del acelerador controla la cantidad de energía eléctrica enviada al motor eléctrico.

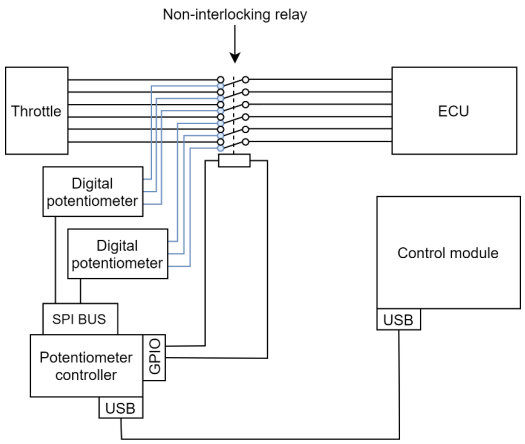


Figura 4.4: Implementación del control del acelerador

En cuanto a nuestro sistema, tenemos en la figura 4.4 la implementación del *bypass* que nos permite actuar sobre los valores que enviaría el pedal del acelerador en un funcionamiento "normal". El sistema cuenta con dos potenciómetros cuya salida se conecta directamente a la ECU que controla la cantidad de energía suministrada por el motor. Estos potenciómetros se controlan mediante un Arduino a través de comunicación SPI, que a su vez este se encuentra a la espera de las peticiones del módulo de control para actuar sobre el sistema de aceleración del vehículo. [7]

Durante el proceso de poner en marcha un vehículo de manera autónoma, se detectó que el controlador del acelerador diseñado por Kanaan no funcionaba, probablemente porque uno de los potenciómetros había fallado, impidiendo el control longitudinal del vehículo. Dado que no se contaba con documentación adecuada, fue necesario investigar desde cero el funcionamiento del sistema para comprender el sistema del acelerador y diagnosticar el problema, así como diseñar de nuevo un controlador que nos permitiera actuar sobre la aceleración del vehículo.

4.3.1. Investigación de un nuevo sistema

Para poner en contexto como funciona el acelerador, en la figura 4.5 se muestra el diagrama de bloques del acelerador. Tenemos como entrada la presión que se ejerce sobre el pedal, esta actúa sobre dos circuitos diferentes que contienen un potenciómetro y resistencias cada uno, cada circuito entrega un voltaje a su salida conocido como V_{out} que llega a la ECU que controla la energía que debe desplegar el motor.

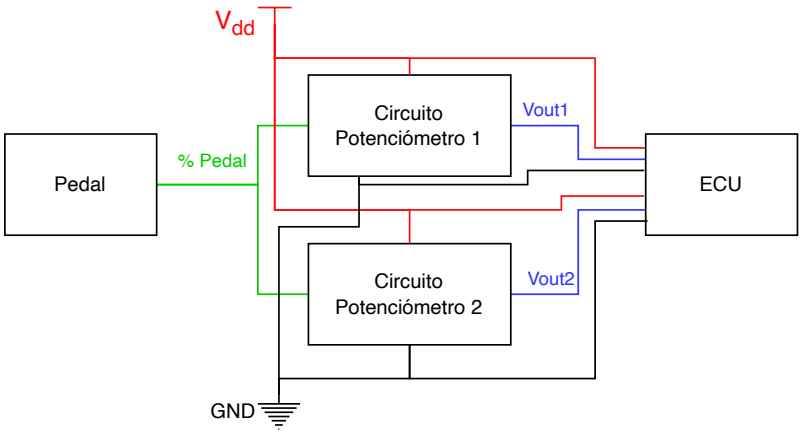


Figura 4.5: Diagrama de bloques del acelerador

Durante la investigación, se descubrió que, aparte de haber un microcontrolador defectuoso, la configuración de pines de salida V_{out1} y V_{out2} diseñada por Kanaan era incorrecta, ya que los pines de salida V_{out1} y V_{out2} (mostrados en la figura 4.6, sacada de [8]) estaban intercambiados, ya que la disposición correcta es el pin 1 para V_{out2} y el pin 4 para V_{out1}.

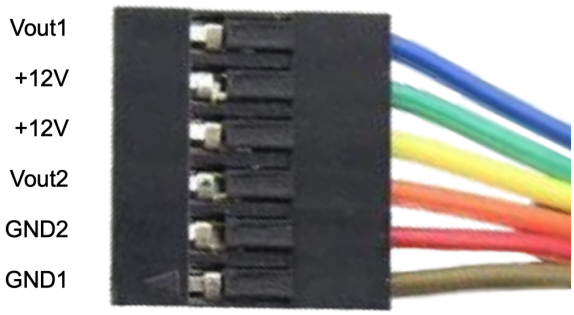


Figura 4.6: Conector de 6 pines del acelerador

Por lo tanto, se decidió rehacer por entero el controlador de acelerador, investigando desde cero el funcionamiento de acelerador en el vehículo. Una vía de investigación era medir las resistencias que

hay en los circuitos de los potenciómetros. A través de mediciones con un multímetro se ha llegado a la conclusión de que el circuito eléctrico del pedal del acelerador es el que se ve en la figura 4.7.

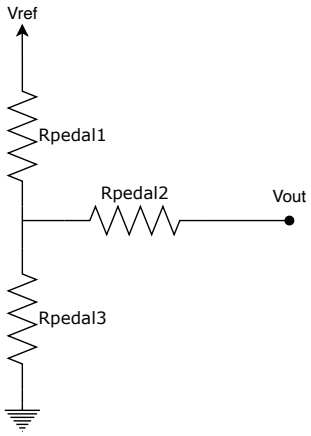


Figura 4.7: Circuito del pedal del acelerador

Por otra parte, se llevaron a cabo mediciones de voltaje en el cableado del vehículo, revelando varias cuestiones interesantes:

- El voltaje de referencia V_{ref} no coincide, como se esperaba, con V_{DD} (12.6 V), posiblemente debido a la existencia de una resistencia entre la batería y el punto V_{ref}.
- El voltaje de referencia no es el mismo para cada potenciómetro, la resistencia entre V_{DD} y V_{ref} tiene diferente valor para cada potenciómetro.
- Se planteó la existencia de una resistencia de salida R_{out} que podría afectar a los cálculos de los potenciómetros, por lo que había que verificar su existencia o despreciar su efecto.

Con el conocimiento del circuito eléctrico del pedal del acelerador, mostrado en la figura 4.8, se pudo diseñar un posible circuito eléctrico para todo el sistema de aceleración del vehículo.

Los voltajes de referencia V_{ref} se mantienen constantes, para el potenciómetro 1 V_{ref1} y para el potenciómetro 2 V_{ref2}, al pisar el pedal del acelerador, lo que asegura que la resistencia que forma el potenciómetro visto desde el punto V_{ref} sea constante. En este caso, se midió con un multímetro que la suma de las resistencias, $R_p = R_{pedal1} + R_{pedal3}$, es de $R_{p2} = 3{,}6K\Omega$ para el potenciómetro 2 y $R_{p1} = 1{,}82K\Omega$ para el potenciómetro 1. Además, la resistencia R_{out} se puede despreciar debido a su efecto insignificante en la salida, es decir, la podemos considerar un circuito abierto.

Con esta información, el esquema simplificado del sistema de aceleración del vehículo se muestra en la figura 4.9.

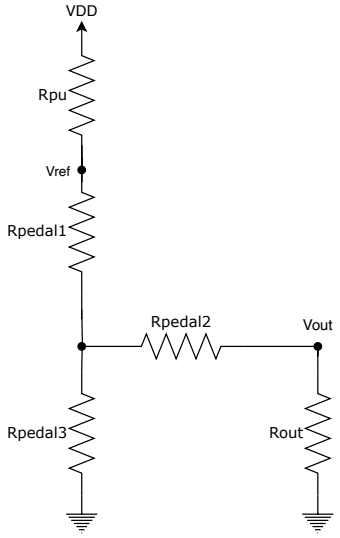


Figura 4.8: Circuito eléctrico completo del sistema del acelerador

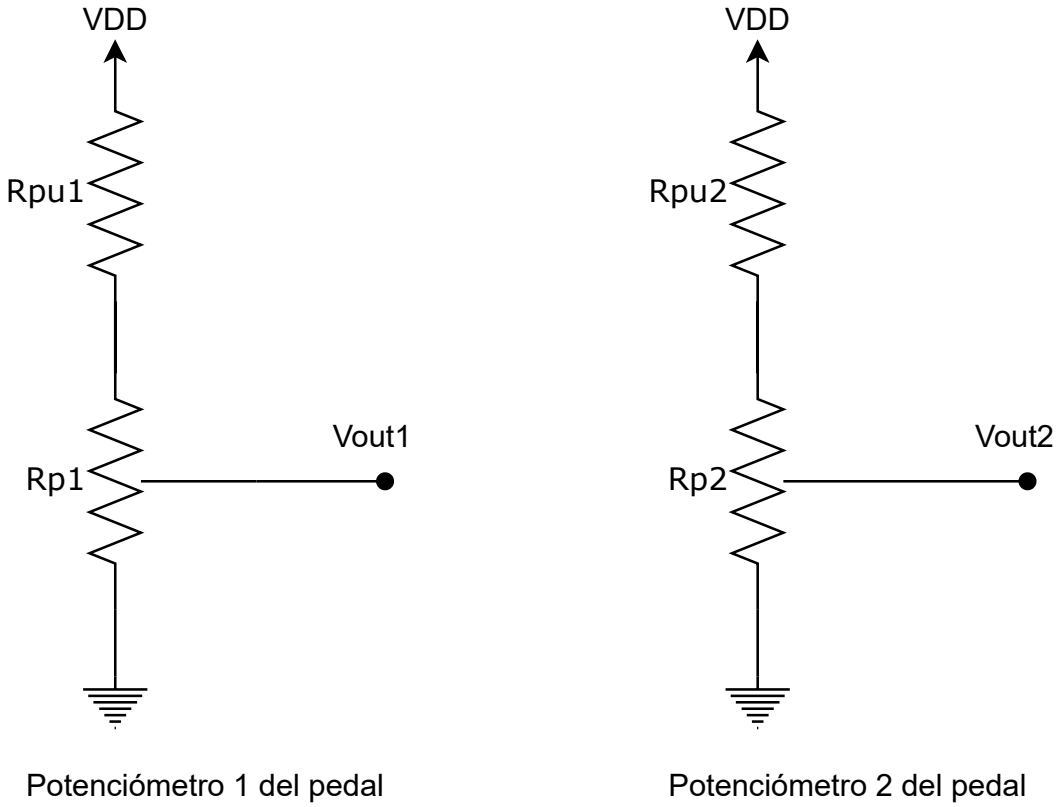


Figura 4.9: Sistema del acelerador simplificado

Posteriormente, a partir de los valores de las resistencias de los potenciómetros y de los voltajes de referencia medidos, se calcularon teóricamente los valores de la resistencia de *pull up* R_{pu} para

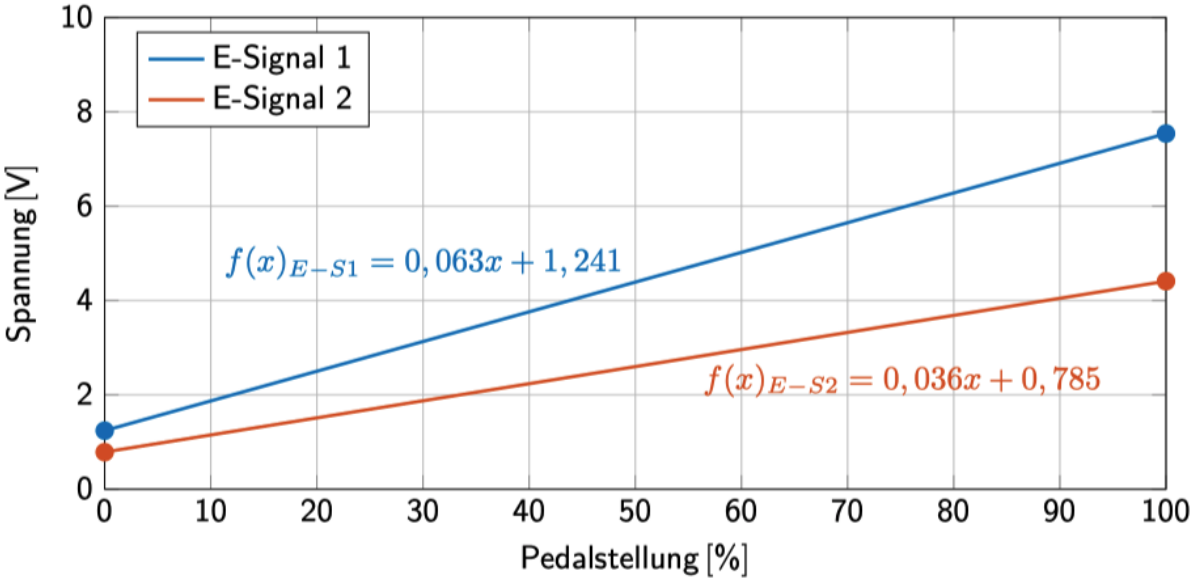


Figura 4.10: Rango de voltajes teórico

cada potenciómetro (R_{pu1} y R_{pu2}). Para el potenciómetro 1, con una resistencia R_{p1} de 1,82$K\Omega$ y un voltaje de referencia V_{ref1} de 8.81 V, y para el potenciómetro 2, con una resistencia R_{p2} de 3,6$K\Omega$ y V_{ref2} de 10.31 V, se obtuvieron los valores de resistencia R_1 de 4,23$K\Omega$ y 16,1$K\Omega$ respectivamente.

Potenciómetro 1:

$$\frac{V_{DD}}{R_{pu1} + R_{p1}} = I = \frac{V_{DD} - V_{ref1}}{R_{p1}}$$

$$R_{pu1} = V_{DD}\frac{R_{p1}}{V_{DD} - V_{ref1}} - R_{p1} = 4{,}23K\Omega$$

Potenciómetro 2:

$$\frac{V_{DD}}{R_{pu2} + R_{p2}} = I = \frac{V_{DD} - V_{ref2}}{R_{p2}}$$

$$R_{pu2} = V_{DD}\frac{R_{p2}}{V_{DD} - V_{ref2}} - R_{p2} = 16{,}1K\Omega$$

Los voltajes de salida están dentro de un rango de operación, que tiene un voltaje mínimo y un voltaje máximo. Para este vehículo los voltajes máximos teóricos son de 7,43 V para el potenciómetro 1 y 4,83 para el potenciómetro 2, así como los mínimos son de 1,37 V y 0,76 V respectivamente. En la gráfica 4.10, sacada de [20], se muestra la evolución de los voltajes de salida en función de la presión ejercida sobre el pedal.

En nuestro coche hemos tomado las medidas para comprobar el estado del sistema y los resultados son bastante similares, como se puede comprobar en la tabla 4.3 y en la gráfica 4.11. En ella se aprecia que el voltaje aumenta a medida que aplicamos una mayor presión sobre el pedal de una manera lineal. El único cambio más destacable se produce cuando aplicamos una fuerza del 100 % que el voltaje sufre una subida más pronunciada en comparación con la salida que nos da si aplicamos el 99 %. Esto se debe a que los coches tienen un sensor de final de recorrido en el pedal del acelerador que se activa

cuando pisamos el acelerador hasta el fondo y provoca una fuerte aceleración en comparación con el resto de recorrido del pedal.

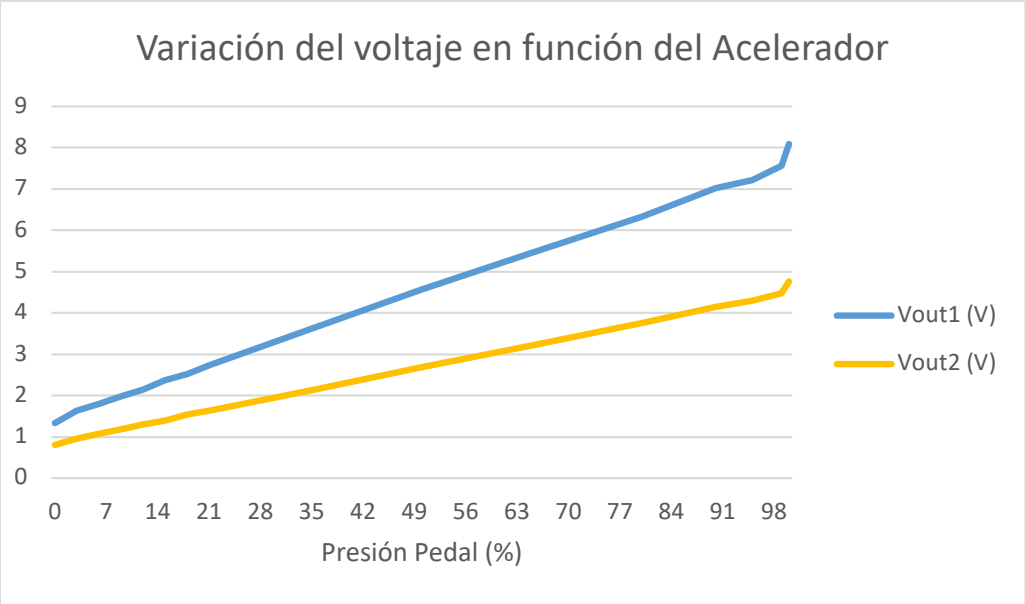


Figura 4.11: Gráfica de la variación de V_{out1} y V_{out2}

Pedal (%)	V_{out1} **(V)**	V_{out2} **(V)**
0	1.34	0.8
3	1.63	0.96
6	1.8	1.07
9	1.98	1.18
12	2.15	1.31
15	2.37	1.39
18	2.52	1.54
21	2.73	1.63
35	3.62	2.13
50	4.57	2.69
65	5.45	3.22
80	6.33	3.76
90	7.02	4.15
95	7.22	4.3
99	7.56	4.48
100	8.09	4.77

Tabla 4.3: Tabla de valores de V_{out1} y V_{out2} en función del pedal (%).

El objetivo es que los voltajes de salida V_{out} sean similares a los que suministraría el circuito del pedal del acelerador, por lo que primeramente es necesario comprobar si somos capaces de estar en el mismo rango de voltaje para ambos potenciómetros. En nuestro sistema se van a integrar unos potenciómetros que mantienen una resistencia total de $20K\Omega$. Esto implica que las V_{ref} de los dos potenciómetros cambiarán, pero esto no es especialmente importante porque no hay ningún circuito que lo detecte y nos aseguramos que la corriente del sistema va a ser menor que en el caso original. Por lo tanto, nuestra única preocupación es ver si con la variación de las resistencias que controlan el potenciómetro es posible mantener un rango de V_{out} igual al que tenemos en el pedal original, y es fácil comprobar que con los valores de resistencia calculados antes se pueden conseguir los voltajes máximos y mínimos medidos.

El rango del voltaje que los nuevos potenciómetros nos pueden dar a la salida estará entre tierra y la nueva V_{ref}. El valor de V_{ref} toma los siguientes valores:

Para el potenciómetro 1:

$$V_{ref1} = V_{DD} \frac{R_{p1}}{R_{p1} + R_1} = 10{,}4V$$

Para el potenciómetro 2:

$$V_{ref2} = V_{DD} \frac{R_{p2}}{R_{p2} + R_2} = 6{,}98V$$

Por lo tanto, con estos potenciómetros podemos seguir cumpliendo el rango de voltajes de salida que requiere el sistema del acelerador.

Por último, los potenciómetros son una resistencia variable en los que cada variación (paso) de esta significa, también, una variación en la resistencia. Por lo tanto, ahora que ya conocemos el rango de voltaje en el que trabajaremos, podemos averiguar el rango de pasos con los que trabajaremos en el potenciómetro para igualar los voltajes a los medidos en el sistema convencional del vehículo. Para esto necesitamos calcular primero la resistencia máxima y mínima necesaria para cumplir nuestro objetivo:

Para el potenciómetro 1:

$$R_{pedal3\_1max} = V_{out1max} \frac{20K\Omega + R_1}{V_{DD}} = 15{,}52K\Omega$$

$$R_{pedal3\_1min} = V_{out1min} \frac{20K\Omega + R_1}{V_{DD}} = 2{,}57K\Omega$$

Para el potenciómetro 2:

$$R_{pedal3\_2max} = V_{out2max} \frac{20K\Omega + R_2}{V_{DD}} = 13{,}66K\Omega$$

$$R_{pedal3\_2min} = V_{out2min} \frac{20K\Omega + R_2}{V_{DD}} = 2{,}29K\Omega$$

Los voltajes utilizados son los que se han medido del sistema convencional que se detallan en la tabla 4.3. Esta resistencia se entrega en un máximo de 1024 pasos, por lo que el rango de pasos en el que trabajaremos es el siguiente:

Para el potenciómetro 1:

$$n_{max1} = R_{pedal3\_1max} \frac{1024}{20K\Omega} = 797$$

$$n_{min1} = R_{Pmin1} \frac{1024}{20K\Omega} = 132$$

Para el potenciómetro 2:

$$n_{max2} = R_{Pmax2} \frac{1024}{20K\Omega} = 700$$

$$n_{min2} = R_{Pmin2}\frac{1024}{20K\Omega} = 117$$

4.3.2. Desarrollo del sistema

El circuito eléctrico del nuevo controlador del acelerador es el que se muestra en la figura 4.12 y consta de un Arduino Nano, dos relés (que tienen como objetivo controlar si estamos en modo autónomo y, por lo tanto, se hace un *bypass* del pedal o si estamos en modo manual y es el pedal el que controla los voltajes) y dos potenciómetros que son los encargados de simular la presión del pedal y que son controlados mediante el Arduino.

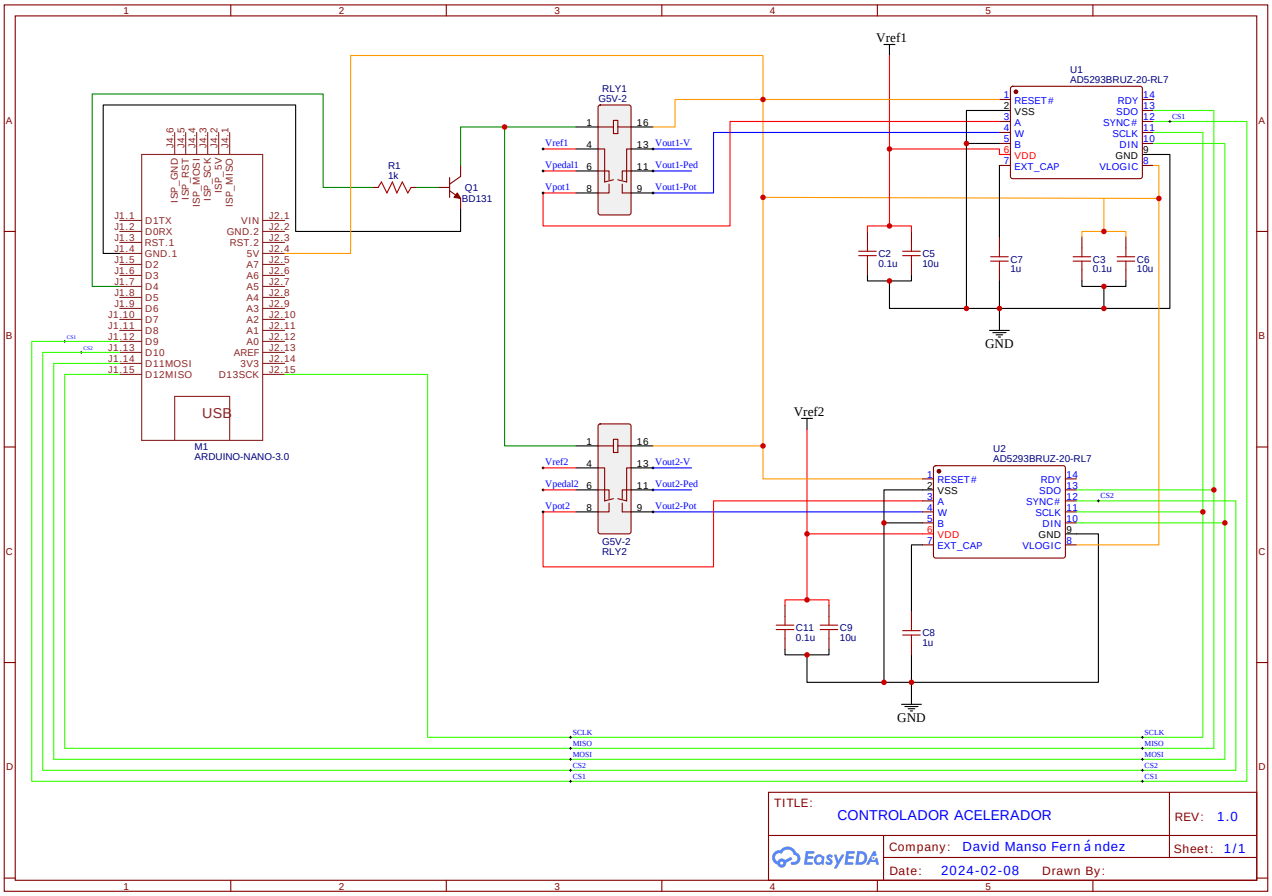


Figura 4.12: Esquema eléctrico del controlador del acelerador

Esta dualidad de funcionamiento, figura 4.13, es posible gracias a los relés que nos permiten conmutar la salida que llega de vuelta a la ECU del vehículo. Los relés en su posición por defecto actúan como pasarela comunicando la alimentación del vehículo V_{ref} con V_{pedal} para que al pedal le llegue la alimentación necesaria. Así mismo, el pedal nos devuelve un voltaje que se corresponde con la presión con la que se esté pisando el pedal $V_{out-Ped}$ y se la enviamos a la ECU para que actúe en consecuencia.

Por otro lado, el relé conmuta si la señal que le llegue por el pin 1 es igual que la que tiene en el pin 16, el cual recibe una señal constante de 5 V procedente del Arduino. Si ambas señales coinciden, el relé conmuta y, por lo tanto, las salidas también. Teniendo ahora desconectado completamente el pedal físico del acelerador y únicamente el voltaje que recibiría la ECU es el voltaje correspondiente con la salida del pin W del potenciómetro.

Mediante la comunicación SPI somos capaces de actuar sobre los potenciómetros con el objetivo

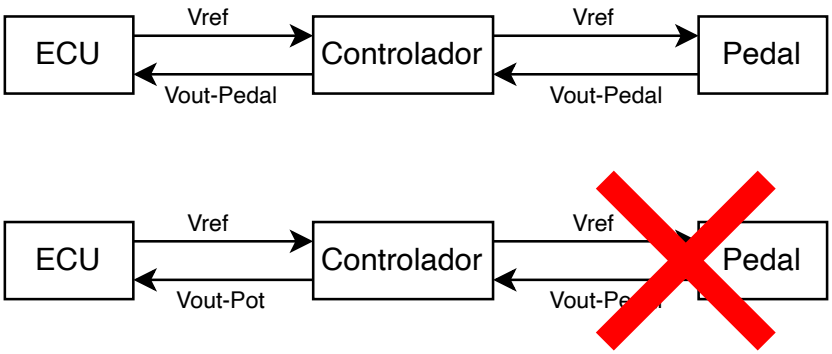


Figura 4.13: Dualidad del controlador del acelerador

de modificar los valores de resistencia del divisor de voltaje y tener el voltaje de salida $V_{out-Pot}$ que queramos. Este voltaje de salida oscila entre el voltaje V_a y V_b, siendo $V_a = V_{ref}$ y V_b será tierra.

4.3.3. Implementación del sistema

Por último solo queda implementar nuestro diseño y montarlo en una placa de prototipado, el resultado final es el que se muestra en la figura 4.14. Para comprobar su correcto funcionamiento antes de instalarlo en el coche se ha de comparar el voltaje de salida de ambos potenciómetros con el que produce el circuito interno del pedal (figura 4.7).

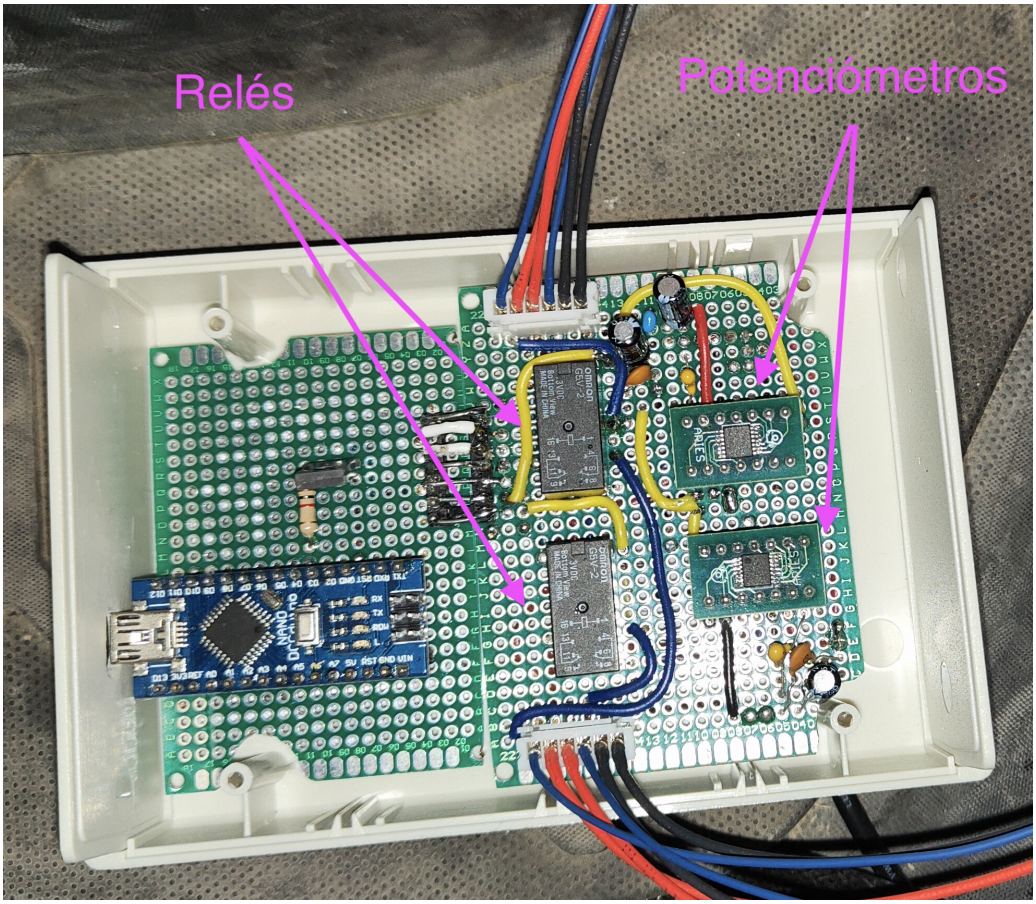


Figura 4.14: Diseño final de las placas de prototipado

Los valores teóricos de voltajes que entregaría nuestro controlador son los mostrados en la tabla 4.4. Se puede ver que son valores similares, aunque se aprecia que hay un pequeño *offset* en los valores con respecto a las mediciones. Este *offset* se puede deber a diferentes motivos cuya afectación en el funcionamiento del sistema es residual, por lo que se pueden considerar despreciables.

Pedal (%)	V_{out1} **(V)**	V_{out2} **(V)**
0	1.65	1.00
3	1.87	1.14
6	2.09	1.27
9	2.32	1.40
12	2.54	1.54
15	2.77	1.67
18	2.99	1.80
21	3.21	1.94
35	4.26	2.56
50	5.38	3.23
65	6.50	3.90
80	7.62	4.56
90	8.36	5.01
95	8.74	5.23
99	9.04	5.41
100	9.11	5.45

Tabla 4.4: Tabla de valores teóricos de nuestro sistema.

A la hora de la inicialización del controlador, nos encontramos con un problema y es que la primera vez que se activa el *bypass* fija una tensión de salida del controlador comparable a como si se ejerciera una presión de alrededor del 37 % sobre el pedal que, a pesar de enviar las órdenes para fijar la tensión de salida al mínimo posible, el sistema no actúa en consecuencia. Este problema desaparece cuando volvemos a desactivar el *bypass* y, a partir de aquí, cada vez que volvamos a activar el controlador, el sistema va a reaccionar a nuestras órdenes y funcionar correctamente. Por último, cabe destacar que cuando activamos el *bypass* provoca que en el cuadro de mandos del coche, se encienda el testigo de color naranja únicamente cuando el acelerador está siendo controlador por nuestro sistema.

4.4. Control Lateral

Para el control lateral, como se ha mencionado en ocasiones anteriores, se necesita de la combinación de diferentes componentes para poder hacerlo posible. Estos componentes son en nuestro caso la controladora **EPOS4 70/15** de la compañía Maxon Motor, así como su motor **EC 60 flat**, ver figura 4.16 sacada de [6]. Para poder transferir la potencia de giro desde este motor a la columna de dirección del Twizy se hace a través de un sistema de engranajes.

Figura 4.15: Esquema de la comunicación Módulo de Control - Motor

Figura 4.16: Controladora EPOS4 70/15 y motor EC 60 flat

El control del motor lo realiza la controladora EPOS4, a la que se le envía órdenes desde el módulo de control, como se muestra en la figura 4.15. El módulo de control enviará órdenes de alto nivel, como por ejemplo establecer un número de vueltas objetivo, velocidad, aceleración, etc. Además, la controladora interpretará estos datos para controlar el motor a través de pulsos, intentando cumplir los objetivos marcados por la orden de alto nivel.

4.4.1. Características Principales

La controladora EPOS4 (*Easy Positioning System*) de Maxon Group está diseñada para controlar motores de corriente continua (DC) con y sin escobillas (*brushless* DC), motores de corriente continua con codificador y motores lineales [21]. En particular, el modelo 70/15 permite un voltaje de operación de 10-70 V y un máximo de 15 A de corriente continua. Además, permite la comunicación a través de diferentes protocolos, como pueden ser CANopen, EtherCAT, USB, RS232, y Ethernet, lo que la convierte en un dispositivo de gran utilidad para muchos sistemas. [22]. Este dispositivo permite controlar el giro del motor con varios modos de funcionamiento, ya sea el modo de posición, velocidad y *homming* entre otros.[23]

También dispone de una serie de entradas y salidas configurables, tanto digitales como analógicas que permiten al sistema actuar en consecuencia de estas señales que, junto con su capacidad de ser programada mediante software externo, como puede ser EPOS Studio o mediante el uso de *scripts* en C++, otorgan a este dispositivo un amplio abanico de posibilidades.[24]

4.4.2. Modos de funcionamiento

La EPOS4 ofrece varios modos de funcionamiento para adaptarse a distintas aplicaciones y necesidades de control. A continuación se hace una breve descripción de los más importantes.

Modo de Posición

En el modo de posición, la controladora EPOS4 se encarga de mover el motor a una posición específica con gran precisión. El control de posición se realiza mediante retroalimentación del sensor, generalmente un codificador (*encoder*) que mide la posición actual del eje del motor y la compara con la posición objetivo.

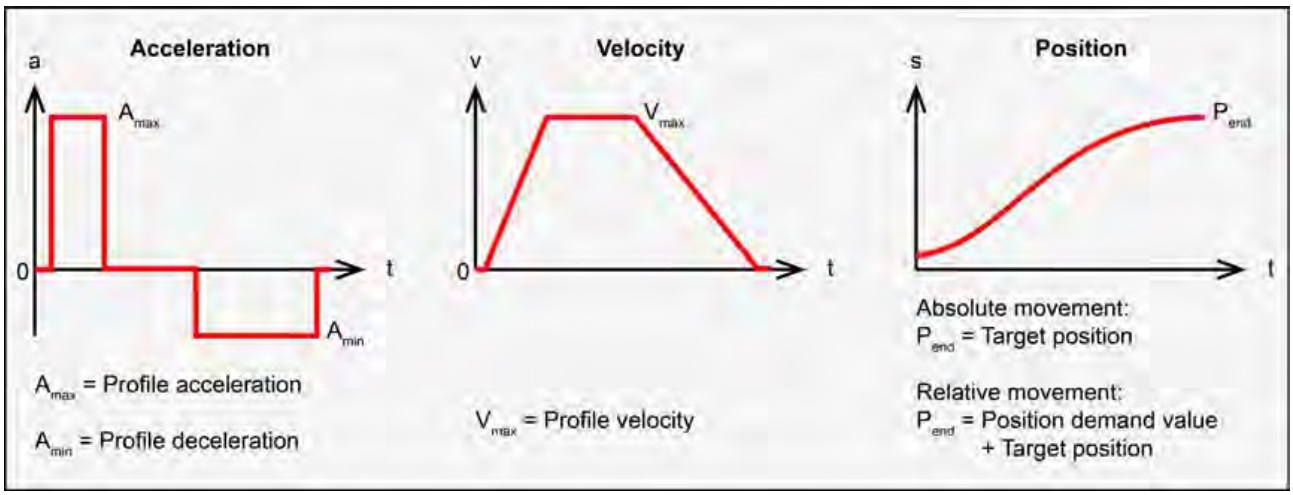


Figura 4.17: Gráficas de los parámetros del modo de posición

En cuánto a los modos de posicionamiento, tenemos dos maneras de trabajar con ellos, ver figura 4.17:

- **Posicionamiento absoluto**: El motor se mueve a una posición absoluta definida en función de una referencia previamente fijada, como puede ser la posición 0.
- **Posicionamiento relativo**: Permite mover el motor la cantidad de pasos indicada desde su posición actual.

La velocidad con la que se puede mover el motor viene definida por el usuario, incluyendo una velocidad máxima, así como su aceleración y deceleración. Además, el perfil de velocidad se controla mediante un perfil de velocidad que puede ser **trapezoidal** (ver figura 4.17) o **sinusoidal**, lo que garantiza los movimientos suaves y precisos.

Por último, este modo tiene dos perfiles de movimiento, derivados de los perfiles de velocidad previamente mencionados, como son el trapezoidal y sinusoidal. Vamos a destacar el primero, ya que comienza con una aceleración constante hasta que alcanza la velocidad deseada y se mantiene hasta que comienza a decelerar hasta llegar a la posición objetivo.

Modo de velocidad

El modo de velocidad permite mantener una velocidad específica bajo diferentes situaciones sin importar las variaciones de la carga que tenga que soportar. Este modo nos permite trabajar en un rango amplio de velocidades, ya que es capaz de trabajar a velocidades muy bajas hasta velocidades muy altas. Además, la velocidad de giro se puede especificar en revoluciones por minuto (RPM) o en radianes por segundo (rad/s). Esta velocidad de giro es constante como se aprecia en la figura 4.18 y el sentido de giro y de la aceleración viene determinado por la velocidad objetivo, como viene descrito en la demostración de funcionamiento en la figura 4.19.

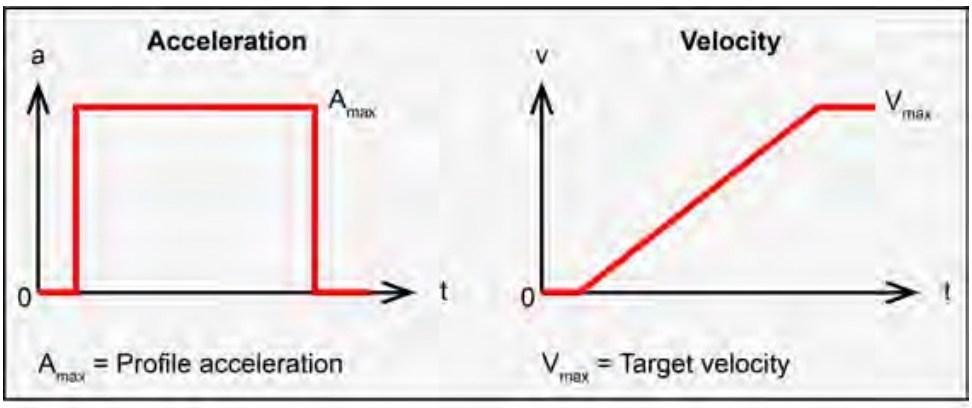


Figura 4.18: Gráficas de los parámetros del modo de velocidad

Este modo, a diferencia que el modo de posición, nos permite obtener la velocidad real a la que está girando el motor para poder ajustarla si fuera necesario. Además, mediante los parámetros de aceleración y deceleración se puede cambiar la velocidad de manera controlada.

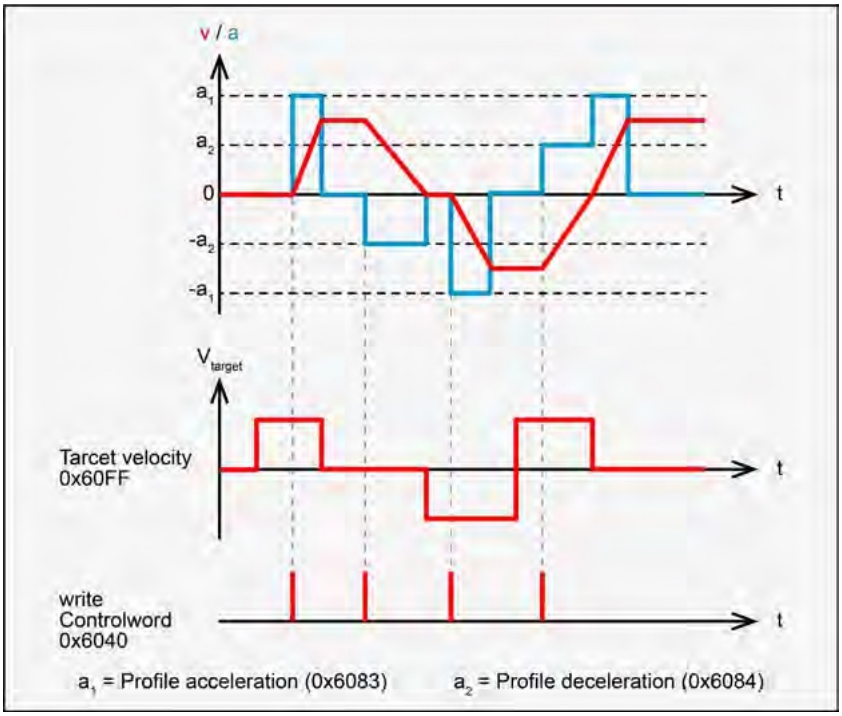


Figura 4.19: Comparación de velocidad y aceleración

Modo Homming

El modo Homming se utiliza para determinar una posición de referencia inicial para el sistema. Este modo es esencial cuando se necesita una posición de partida antes de comenzar con la operación normal.

Este modo lo que hace es realizar una búsqueda de la posición de referencia. La secuencia de búsqueda puede ser girar en una dirección hasta que la corriente supere un cierto umbral, identificando de esta manera el fin de carrera. Si se sabe la carrera total, entonces es posible fijar una referencia. Otra forma es mediante el uso de sensores que sean detectables por el motor y cuya posición conocemos que nos permitan determinar la posición en la que se encuentra el motor. Una vez esta búsqueda ha finalizado, el sistema determinará la posición final como la posición de referencia del sistema (posición cero).

4.4.3. Alimentación EPOS4

La arquitectura original con la que empezó este proyecto tenía a la controladora EPOS4 alimentada directamente desde la batería de servicio, algo que en esta nueva implementación queríamos cambiar. Uno de los objetivos de este proyecto era agrupar la alimentación de todos los dispositivos electrónicos en la PowerBox entre los que se incluye la EPOS4. Las razones para hacerlo así se dieron anteriormente, pero la fundamental es que aunque el resto de dispositivos estén apagados, la EPOS sigue funcionando, agotando la batería de servicio. Como veremos a continuación, la alimentación de la EPOS a través de la PowerBox ha dado algunos problemas no esperados.

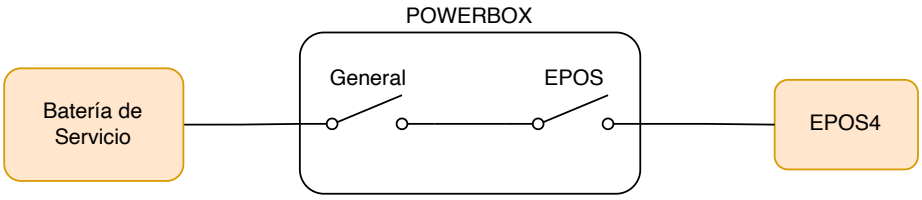


Figura 4.20: Esquema eléctrico de la EPOS

Como vemos en la figura 4.20, el objetivo final es que la alimentación de la EPOS pase por dos interruptores que se encuentran la caja de conexiones, en primera instancia tenemos el interruptor general que alimentará la PowerBox y después tenemos el interruptor específico de la EPOS. Este diseño se ha llegado a implementar y el sistema funcionaba correctamente en casi todas las situaciones, el único problema es que cuando la EPOS se encuentra en las situaciones más extremas como puede ser el cambio brusco de orientación de giro o cuando llega a las posiciones finales en los que el sistema ofrece una gran resistencia. En estos casos, la EPOS llegaba a desconectarse y no permitía seguir operando el motor.

Analizando los mensajes de error que devolvía la EPOS descubrimos que se trataba de un problema de alimentación, que no era suficiente para lo que requería en esas situaciones. Sin embargo, se sabía que no podía ser un problema de la fuente de alimentación, ya que si alimentamos la EPOS directamente desde la batería de servicio, esta funciona perfectamente hasta en las situaciones más complejas, por lo que el problema debía residir en las modificaciones que se habían hecho en la nueva versión.

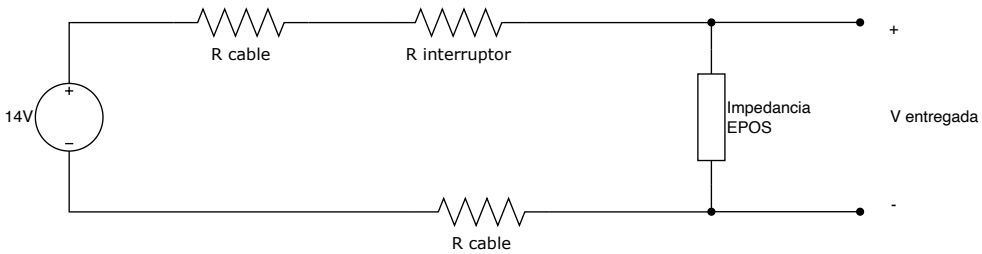


Figura 4.21: Modelado del circuito eléctrico de la entrega de potencia a la EPOS

Analizando las especificaciones de la EPOS y del motor, descubrimos que la EPOS, una vez fijada un voltaje inicial de alimentación, es capaz de soportar variaciones de voltaje de 1 V como máximo:

$$\Delta \mathrm{V} = 14\,\mathrm{V} - V_{\text{entregada}} < 1\,\mathrm{V}$$

Dado que la EPOS estaba conectada a la fuente a través de la PowerBox que estaba varios metros alejada de la batería, el problema estaba en que el cable utilizado introducía una caída de voltaje en

función de la corriente solicitada. El modelado eléctrico del cable junto con la PowerBox se puede ver en la figura 4.21. El voltaje máximo que caerá en la resistencia del cable y del interruptor se puede calcular a partir de la corriente que circule por ellos, y teniendo en cuenta que conocemos la potencia máxima que se puede entregar al motor, encontramos que esta corriente se puede calcular fácilmente:

$$I = \frac{100\,\mathrm{W}}{(14\ -\ 1)\mathrm{V}} = 7{,}7\,\mathrm{A}$$

Analizando nuestro circuito representado en la figura 4.21 tenemos que calcular la resistencia máxima en serie que podemos admitir:

$$R_{serie} = 2R_{cable}\ +\ R_{interruptor}\ < \frac{1\mathrm{V}}{7{,}7\mathrm{A}} = 0{,}13\Omega$$

Por lo tanto, lo primero que se hizo fue considerar que la resistencia del interruptor era nula y calcular un cable con una sección que nos asegurara una caída baja de voltaje, teniendo en cuenta que el cable total medía 5 m. Decidimos utilizar un cable AWG12, que se corresponde con una sección de $4\mathrm{mm}^2$ y con una resistividad proporcionada por el cobre de $\rho = 0{,}017\,\Omega\,\mathrm{mm}^2/\mathrm{m}$. De esta forma se puede calcular la resistencia total producida por el cable:

$$R_{cable} = \rho\frac{L}{S} = 0{,}017\frac{5\mathrm{m}}{4\mathrm{mm}^2} = 0{,}02125\,\Omega$$

En este punto se realizaron varias pruebas y se comprobó que el problema no se había resuelto. Después de mirar varias posibilidades se llegó a la conclusión que el problema estaba en suponer que la resistencia de los interruptores era nula. Efectivamente, los interruptores tienen una resistencia de contacto, y se puede calcular la resistencia total que pueden introducir sin más que aplicar una diferencia:

$$R_{\mathrm{interruptor}} \leq 0{,}13\,\Omega - 2 * 0{,}02125\,\Omega = 0{,}0875\,\Omega = 87{,}5\,\mathrm{m}\Omega$$

Según las especificaciones del fabricante, las resistencias de contacto de los dos interruptores en la PowerBox son menores a $50m\Omega$, pero que el polímetro nos decía que era de alrededor de 2,5Ω. Al no cumplirse claramente los requisitos de resistencia máxima en los interruptores, era fácil que se produjera fallo de alimentación. A día de hoy no ha sido posible introducir unos interruptores con una resistencia acorde a nuestras necesidades, por lo que de momento la EPOS sigue siendo alimentada directamente por la batería de servicio. Hasta que no se solucione este punto deberemos buscar otra fuente de alimentación para dar energía a la PowerBox, por este motivo es por el que se está utilizando un transformador conectado al *inverter* que se vio en el apartado 3.4.2.

4.4.4. Software

Hay dos maneras de manejar con software la controladora EPOS, una es mediante el programa desarrollado por la compañía Maxon, **EPOS Studio**, que mediante una interfaz gráfica muy visual permite manejar la controladora, poder supervisar el funcionamiento en tiempo real, configurar la controladora ajustando los parámetros necesarios, actualizar el software etc. [6]

La otra manera de controlar la EPOS es mediante scripts en C++ y en Python, aunque estos últimos utilizan las librerías disponibles de C++. Esto nos aporta una mayor capacidad de personalización, ya que podemos diseñar programas que se ajusten a nuestras necesidades sin depender de programas de terceros. Maxon suministra una API junto con su librería para que podamos comunicarnos con la controladora y poder manejarla. Este método es el que se usa para el control lateral del Twizy. [6]

Nuevo modo de funcionamiento

Hasta ahora el control lateral estaba definido mediante el modo de posición. Como se ha visto anteriormente, este modo necesita como entradas la velocidad máxima de movimiento y la posición

objetivo, entre otros. Pero con este modo de funcionamiento nos encontrábamos con el inconveniente de la desconexión del motor. Cuándo utilizábamos este modo y queríamos que girase a una velocidad elevada (superior a 3000 rpm) el motor se desconectaba y se paraba la ejecución. La razón para este comportamiento se debe a que en el modo posición la controladora antes de empezar el movimiento define una trayectoria de la posición a alcanzar en un tiempo concreto, y posteriormente, durante la operación, comprueba continuamente la posición alcanzada, de tal manera que si en un momento determinado la diferencia entre la posición leída por la controladora y la esperada es mayor que un margen predefinido, la controladora entra en modo error y se desconecta. Esto obligaba a trabajar a muy bajas revoluciones, lo que a su vez acababa impactando en la velocidad máxima a la que se puede mover el vehículo (5 km/h en proyectos anteriores) dado que el seguimiento de la línea azul es muy dependiente de la velocidad de corrección con el volante.

Ahora se ha implementado un código con control en **modo de velocidad**, que nos entrega una velocidad de giro del volante constante y no se desconecta si introducimos velocidades elevadas. La manera con la que vamos a trabajar ahora es que se va a monitorizar continuamente la posición del motor en tiempo real desde el ordenador de control en lugar de desde la controladora. El programa desarrollado tiene como entradas la velocidad deseada y la posición objetivo, el programa es capaz de comparar la posición actual con la objetivo para poder determinar el sentido de giro del motor. Después comienza a mover el motor mientras monitoriza la posición en tiempo real y la compara con la objetivo. Se ha fijado un umbral en el que si la diferencia entre la posición objetivo y la real es inferior a este umbral, significa que ya ha llegado a su destino y se da por finalizado el proceso, deteniendo el movimiento.

Este modo no es perfecto, ya que a diferencia del modo de posición tenemos un menor control en la búsqueda de la posición objetivo. Esto significa que se pueden dar más situaciones de *overshooting*, esto es, que el motor oscile continuamente alrededor de la posición final deseada.

Capítulo 5

Software

5.1. Introducción

En este apartado se hablará de los programas principales relacionados con el control de este prototipo, empezaremos con el software principal llamado `mod-con.py` que monitoriza la información que captan y envían los sensores para procesarla y tomar decisiones en función de ella. Seguidamente, se comentará el programa que se encarga del control lateral, veremos como gestiona las órdenes que recibe del módulo de control, así como la monitorización de la posición y velocidad actual del motor y para terminar, haremos un repaso de las diferentes funcionalidades de las que dispone. Todo el software que hablaremos a continuación está disponible en el repositorio de *GitHub*: `https://github.com/GCOdeveloper/Twizycontest`

Pero antes de hablar del software, cabe destacar cómo se ha conseguido solucionar una dificultad que tenían en las versiones anteriores. Esta dificultad era que el módulo de control, cuando se inicia, no asigna en el mismo orden las interfaces USB de los periféricos. La solución temporal era utilizar una secuencia concreta de encendido de los periféricos, pero a partir de ahora eso no va a ser necesario porque se va a a realizar una identificación previa de los dispositivos conectados.

5.2. Identificación de los dispositivos conectados

5.2.1. Mapeo persistente dispositivos USB

En el estado inicial de este proyecto, los periféricos no tenían asignado un nombre concreto en el módulo de control. La solución adoptada era ir encendiendo en una secuencia determinada los diferentes periféricos conectados al *hub* USB para que así el módulo de control asigne siempre el mismo identificador a cada periférico, por ejemplo: cuando encendemos varios periféricos están conectados a la Humming, el sistema operativo Linux asigna las interfaces USB en orden, comenzando por **ttyUSB0** para el primer periférico que encuentra y así sucesivamente. El orden en el que encuentra los periféricos puede variar de un encendido a otro.

Este mecanismo es efectivo pero no muy eficiente, ya que necesitas disponer de un hub USB con interruptores y cada vez que reinicias el módulo de control tienes que repetir la secuencia de encendido. Por este motivo se ha implementado una solución bastante más eficiente y sencilla, como es la de asignar alias o enlaces simbólicos a los periféricos, también conocida como mapeo persistente de los dispositivos USB.

Esto consiste en que en una primera instancia se debe averiguar cuáles son los atributos de cada periférico, como son el *idVendor* y el *idProduct*. Estos son visibles mediante el comando **lsusb** que muestra los dispositivos USB conectados junto con su nombre e identificador que está compuesto por los atributos mencionados antes. Puede darse el caso en el que dos dispositivos estén conectados mediante el mismo adaptador USB, por lo que la única forma de distinguir cada periférico es mediante

el número de serie individual. Una forma de obtener este serial es mediante el siguiente comando:

```
udevadm info -a -n /dev/ttyUSBX | grep '{serial}' | head -n1
# Resultado
ATTRS{serial}=="AI02P0F4"
```

En el que indicando el nombre de la interfaz que se quiere consultar, por ejemplo ttyUSB0, te muestra el serial del dispositivo conectado a esa interfaz, el cual es único para cada componente.

Una vez obtenidos todos los datos de cada periférico, falta establecer las reglas del mapeo de estos dispositivos. Esto se hace creando un fichero en el directorio */etc/udev/rules.d*, llamado *99-usb-serial.rules*. En él se incluyen para cada dispositivo su *idVendor* e *idProduct* y si fuera necesario su serial, además de un alias o enlace simbólico al que poder hacer referencia desde los programas de ejecución debido a que va a permanecer inalterable. Un ejemplo de este fichero es el siguiente:

```
SUBSYSTEM=="tty", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", ATTRS{serial
    }=="AI02P0F4", SYMLINK+="USB_Marchas"
SUBSYSTEM=="tty", ATTRS{idVendor}=="29fe", ATTRS{idProduct}=="4d53", SYMLINK+="
    USB_Camera"
SUBSYSTEM=="tty", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", ATTRS{serial
    }=="AH020796", SYMLINK+="USB_Acelerador"
SUBSYSTEM=="tty", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", ATTRS{serial
    }=="AB0JPVRD", SYMLINK+="USB_RFID"
SUBSYSTEM=="tty", ATTRS{idVendor}=="1a86", ATTRS{idProduct}=="7523", SYMLINK+="
    USB_Sonidos"
```

5.2.2. Uso de variables de entorno

La idea del mapeo persistente de dispositivos no dio tan buenos resultados como se esperaba, debido a que los periféricos conectados tenían el mismo *idVendor* y el mismo *idProduct* y, por lo tanto, las reglas descritas en el apartado anterior no se aplicaban correctamente. En algunas ocasiones intercambiaba los dispositivos que compartían estos parámetros y no tenía en cuenta el serial. En cambio, sirvió de ayuda para comprender mejor como poder interactuar con los periféricos conectados a la *Humming*. La forma en la que se resolvió finalmente el manejo de las interfaces USB fue la de crear variables de entorno del sistema operativo con el fin de almacenar en ellas las direcciones de los periféricos, como pueden ser, por ejemplo, /dev/ttyUSB0.

Para hacer esto posible, se ha diseñado un *script* que va recorriendo todos los periféricos conectados, que dado que ahora mismo solo tenemos cuatro periféricos conectados al *hub* irán desde ttyUSB0 hasta ttyUSB3. Este *script* abre la comunicación serie con cada periférico de manera individual y se queda esperando a que este envíe mensajes relacionados con el funcionamiento del controlador, por ejemplo, el controlador de marchas envía un mensaje en formato de cadena de caracteres indicando la marcha en la que se encuentra. Dicho esto, la Humming comprueba el mensaje recibido y dependiendo de cuál sea, asigna el nombre del periférico a la dirección del periférico actual.

Por ejemplo, abre el puerto ttyUSB3 y este envía continuamente los valores de los potenciómetros del controlador del acelerador. El *script* crea la variable Acelerador_USB con el valor de la dirección del periférico */dev/ttyUSB3*. Una vez ejecutado el *script*, si miramos las variables de entorno con el comando *printenv* tenemos, entre otras, las siguientes variables:

```
Acelerador_USB  = /dev/ttyUSB0
Sonidos_USB     = /dev/ttyUSB1
Marchas_USB     = /dev/ttyUSB2
RFID_USB        = /dev/ttyUSB3
```

Para poder recoger esta información desde el programa principal necesitamos utilizar las funciones de la librería **os** de Python como es **os.environ()**. En concreto, vamos a utilizar:

```
Aclerador_dir   = os.environ["Acelerador_USB"]
Sonidos_dir     = os.environ["Sonidos_USB"]
Marchas_dir     = os.environ["Marchas_USB"]
RFID_dir        = os.environ["RFID_USB"]
```

Por último, para abrir la comunicación con estos dispositivos se hace mediante el comando **Serial** de la librería **serial**. En particular se hace de la siguiente manera:

```
serial_acelerador   = serial.Serial(Aclerador_dir, 9600)
serial_sonidos      = serial.Serial(Sonidos_dir, 9600)
serial_marchas      = serial.Serial(Marchas_dir, 9600)
serial_RFID         = serial.Serial(RFID_dir, 9600)
```

5.3. Programa Principal

El software principal del módulo de control está implementado en Python, al igual que el correspondiente al módulo de comunicaciones. El programa asociado al módulo de control se denomina `mod-con.py` y cumple con las siguientes tareas[6]:

- Establecer la conexión con el módulo de comunicaciones.
- Identificar el estado operativo actual del vehículo.
- Procesar de manera adecuada la información proporcionada por los sensores y la cámara.
- Desarrollar un algoritmo de conducción tanto longitudinal como transversal, basado en los datos de percepción del entorno, para que el Twizy pueda llevar a cabo una conducción autónoma.

Estas tareas ya han sido implementadas en el Trabajo de Fin de Grado de Ignacio Royuela [7] y el Trabajo de Fin Máster de Samuel Pilar [6]. Ambos módulos son capaces de comunicarse entre sí, y el módulo de control está diseñado para determinar el estado del vehículo, que puede ser uno de los siguientes [6]:

0. Arranque del sistema (*Start-up*).
1. Modo normal (no autónomo).
2. Modo autónomo.
3. Modo de fallos.

Para recibir y procesar la información proveniente de múltiples sensores de manera simultánea, optaron por una arquitectura basada en procesos hijo que se ejecutan en paralelo, utilizando la librería `multiprocessing` de Python. En la Figura 5.1 se presenta el esquema de los procesos que se ejecutan dentro del programa `mod-con.py`.

El proceso hijo denominado `main_driver` será responsable de la lógica de conducción una vez que el Twizy entre en modo autónomo, momento en el que se creará este proceso. Al activarse, `main_driver` generará otros cinco procesos hijo, cada uno encargado de una función específica que veremos a continuación:

- `RFID`: Procesar el ID del *tag* RFID (conversión a ASCII) y notificar al proceso padre `main_driver` que se ha leído un *tag* y cuál es su ID. Dependiendo del mensaje recibido, `main_driver` decidirá si preparar el vehículo para una bifurcación o detenerlo.[6]
- `Cámara`: A diferencia de la versión original y como se ha comentado en varias ocasiones en esta memoria, esta implementación usa una cámara para detectar la línea a seguir en vez del sensor magnético. Este proceso se encarga de procesar la información capturada por la cámara para obtener un ángulo de giro y convertirlo al número de pasos que tiene que dar el motor del control lateral para hacer girar el volante a la posición deseada y mantener el coche centrado con respecto a la línea.

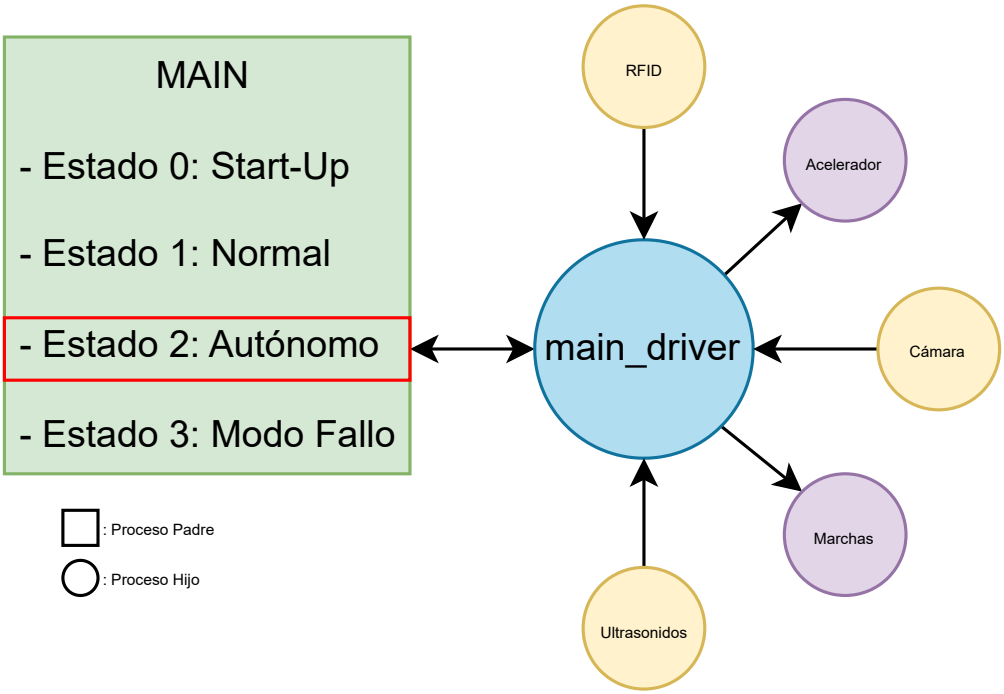


Figura 5.1: Esquema de procesos en el módulo de control

- `Ultrasonidos`: Detectar obstáculos dentro del rango de los sensores ultrasónicos e informar sobre la distancia a la que se encuentran. El proceso padre `main_driver` deberá decidir a qué distancia detener el vehículo o si existe un riesgo de colisión que requiera la interrupción de la marcha.
- `Acelerador`: Simular la señal del pedal del acelerador y determinar la velocidad a la que debe moverse el vehículo. Esta velocidad será definida por el algoritmo de control longitudinal implementado en `main_driver`.
- `Marchas`: Este proceso se encarga de controlar el cambio de marcha cuándo entramos en el modo autónomo y asignar la marcha D para poder mover el vehículo. Esto nos permite poder controlar el modo en el que queremos manejar la caja de cambios, bien sea de manera electrónica con el software o en su defecto, si salimos del modo autónomo permite cambiar de marchas de manera física con los botones del coche.

El siguiente aspecto a considerar es la lógica de conducción que deberá seguir el vehículo, la cual se programará dentro de `main_driver` y cómo este proceso se comunica con el software del control lateral. A continuación vamos a ver cómo se ha implementado este control lateral en el programa principal del vehículo y qué modificaciones se han hecho con respecto al trabajo realizado por Samuel Pilar [6].

5.4. Control lateral

Maxon, a través de su sitio web oficial, proporciona bibliotecas diseñadas para trabajar con sus motores utilizando código en C++. Esta biblioteca incluye una variedad de comandos que permiten

operar el motor en diferentes modos de funcionamiento. En la Figura 5.2, se presenta una visualización de los modos por los cuales el motor transita según la lógica implementada[6].

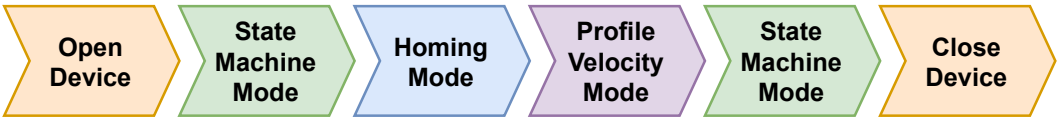


Figura 5.2: Diferentes estados de funcionamiento de la EPOS4

5.4.1. Estados de funcionamiento

Abrir y cerrar la conexión con la EPOS

Empezaremos describiendo el estado inicial y final, que, como sus nombres sugieren, se utilizan para establecer o cerrar la conexión con la unidad de control. Los dos comandos principales en estos estados son:

- `VCS_OpenDevice`: Este comando abre el puerto necesario para enviar y recibir comandos desde la EPOS4. La comunicación puede realizarse mediante interfaces CAN, USB o RS232; en este caso específico, se emplea la interfaz USB para conectar la controladora con el módulo de control.
- `VCS_CloseDevice`: Este comando cierra el puerto utilizado para la comunicación entre la controladora y el módulo de control.

Máquina de estados

El modo denominado *Machine State* describe el estado actual de la controladora, determinando qué comandos pueden ser aceptados en función de dicho estado. Durante la operación, la controladora alterna entre dos estados: activo y desactivado. Los comandos asociados a estos estados son:

- `VCS_SetEnableState`: Activa la controladora, permitiendo la transición a un modo operativo y aceptando los comandos correspondientes.
- `VCS_SetDisableState`: Desactiva el dispositivo.

Modo Homing

Antes de abordar el modo que controla el giro del motor y, en consecuencia, del volante, es crucial tener un punto de referencia de la posición del motor, que permitirá determinar la dirección y el grado de giro. Por defecto, la controladora establece este punto de referencia (también conocido como cero absoluto) en la posición física del motor cuando se abre el puerto de conexión entre la EPOS4 y el módulo de control (comando `VCS_OpenDevice`). Idealmente, este cero absoluto debería estar centrado o en un extremo controlado para proporcionar un punto de referencia preciso.

Nosotros vamos a utilizar como posición de referencia el centro del volante, con las ruedas rectas. Unos de los objetivos era continuar con el trabajo de Samuel utilizando el modo *Homing* 7 [6] pero no hemos sido capaces de obtener unos resultados precisos en todas las pruebas realizadas, ya que usar la corriente como medida de que se ha llegado al final de carrera es útil, pero en muy pocas ocasiones superaremos el umbral de corriente en la misma posición. Por lo que en cada iteración podríamos tener una posición de referencia distinta. Otra posible solución para obtener la posición de referencia es añadiendo un sensor que sea reconocible por la EPOS, estando este sensor en una posición conocida, idealmente uno de los extremos. Entonces, la EPOS hace girar el motor hasta encontrar este sensor y

por último, corrige el *offset* hasta llegar a la posición de referencia.

Hasta que se llegue a implementar esos cambios, estamos dejando el volante en la posición de referencia y cada vez que se entra en el modo autónomo se define la posición actual del motor de control lateral como la posición de referencia, este modo es el número 35 [24] y los comandos principales para este modo son:

- `VCS_ActivateHomingMode`: Activa el modo *Homing*, habilitando los comandos asociados.
- `VCS_DefinePosition`: Establece un nuevo punto de referencia.

Profile Velocity Mode

En cuánto al perfil de movimiento se va utilizar el *Profile Velocity Mode*, que permite controlar únicamente la velocidad del motor durante un periodo continuo de tiempo, sin especificar la posición final. En este modo, se definen la velocidad y la aceleración, y el motor continúa girando a esas características hasta que se indique lo contrario. Los comandos son similares a los del *Profile Position Mode*, pero enfocados en la velocidad en lugar de la posición.

- `VCS_ActivateProfileVelocityMode`: Activa el perfil de velocidad.
- `VCS_SetVelocityProfile`: Establece la aceleración y desaceleración en este modo.
- `VCS_MoveWithVelocity`: Gira el motor a la velocidad especificada.
- `VCS_GetPositionIs`: Devuelve la posición actual del motor.
- `VCS_GetVelocityIs`: Devuelve la velocidad actual del motor.
- `VCS_HaltVelocityMovement`: Detiene el motor según la desaceleración establecida.

La parte más diferencial de este modo con respecto al modo de posición que se utilizaba antes es que no podemos establecer una posición objetivo en las instrucciones que enviamos a la EPOS4. Es por esto por lo que necesitamos comprobar la posición actual del motor para poder controlar el giro del motor. El software que se encarga del control lateral hace lo siguiente y se muestra el diagrama de flujo en la figura 5.3.

1. **Activación del Modo de Velocidad:**
   - El código activa el modo velocidad usando la función `VCS_ActivateProfileVelocityMode`. Si la activación falla, se registra el error y se indica un fallo en la operación.
2. **Configuración del Perfil de Velocidad:**
   - La función `VCS_SetVelocityProfile` establece los parámetros de aceleración y desaceleración, configurados con un valor de 10.000 rpm/s.
3. **Lectura de la Velocidad y Posición Actual:**
   - En cada ciclo de la operación, se llama a la función `VCS_GetVelocityIs` para leer la velocidad actual del motor y a `VCS_GetPositionIs` para obtener la posición actual. Este ciclo de operación es lo que tarda en recorrer un bucle `while` y volver al mismo punto. A su vez hay otros procesos que están por encima gestionando el modo de velocidad y que tardan más tiempo, pero se detallarán más adelante en el apartado 5.5.
   - La posición se expresa en *steps*, donde 0 representa la posición central del volante del vehículo. Desde esta posición central, los movimientos hacia la izquierda aumentan el valor de la posición, y los movimientos hacia la derecha lo disminuyen.
4. **Control de Velocidad:**

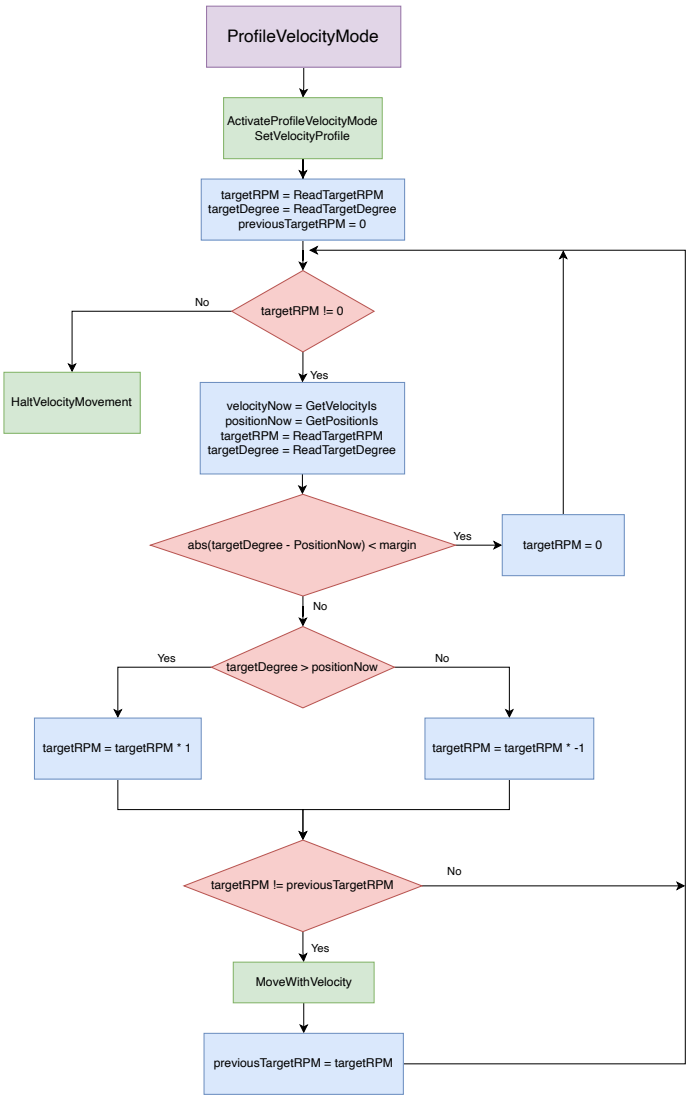


Figura 5.3: Diagrama de flujo del control lateral

- Dependiendo de la posición actual y la posición objetivo, el código decide si el motor debe girar en sentido positivo o negativo. Esto se traduce en una velocidad positiva o negativa, utilizando la lógica:
  - Si la posición objetivo es mayor que la posición actual, la velocidad objetivo (`targetRPM`) se mantiene positiva, lo que hace que el motor gire en sentido antihorario.
  - Si la posición objetivo es menor, la velocidad objetivo se convierte en negativa, girando el motor en sentido horario.
- El comando `VCS_MoveWithVelocity` se utiliza para aplicar la velocidad calculada al motor. Si la velocidad objetivo cambia, se actualiza la orden de movimiento.

5. **Control de Posición:**

   - El código verifica continuamente si el motor ha alcanzado la posición objetivo comparándola con la posición actual. Si la diferencia entre ambas es menor que un margen definido (*10000 steps*), el motor se detiene estableciendo la velocidad objetivo en 0.

6. **Detención del Motor:**

- Una vez que el bucle de control termina, esto sucede cuando se alcanza la posición objetivo o si la velocidad objetivo es superior a la velocidad máxima indicada en la hoja de datos del motor, que es de 6000 rpm. La función `VCS_HaltVelocityMovement` detiene el motor de manera controlada.
- Si el proceso de detención falla, se registra el error y se notifica al usuario.

En resumen, la función `DemoProfileVelocityMode` gestiona el control de velocidad del motor en tiempo real, ajustando la velocidad y deteniendo el motor según la posición objetivo definida por el usuario.

5.4.2. Memoria Compartida

Como ya explica Samuel en su proyecto [6], en el funcionamiento completo del sistema, se utiliza memoria compartida para la comunicación entre procesos, en concreto entre el proceso que se encarga del control lateral con el proceso padre `main_driver`. A continuación se detalla el uso de la memoria compartida en el código [6]:

- **Claves de Memoria Compartida:**
  - Se definen cuatro claves (`Clave1`, `Clave2`, `Clave3`, `Clave4`) que se obtienen utilizando la función `ftok`.
  - `Clave1` y `Clave2` corresponden a la memoria compartida para los valores objetivos de RPM y *steps* (`MemoriaRPM_target` y `MemoriaDEG_target`) que se leen en el software del control lateral y se escriben en el proceso padre.
  - `Clave3` y `Clave4` corresponden a la memoria compartida para los valores actuales de RPM y ángulo (`MemoriaRPM_current` y `MemoriaDEG_current`), que son escritos por este programa y son leídos por el `main_driver`.
- **Lectura y Escritura de Valores:**
  - Los valores de velocidad objetivo (`targetRPM`) y posición objetivo (`targetDegree`) se leen de las posiciones de memoria compartida `MemoriaRPM_target[0]` y `MemoriaDEG_target[0]` respectivamente.
  - Durante cada iteración del bucle, se actualizan los valores actuales de la velocidad (`velocityNow`) y posición (`positionNow`) en la memoria compartida, utilizando los punteros `MemoriaRPM_current` y `MemoriaDEG_current`.

En resumen, el uso de la memoria compartida en este código permite una comunicación eficiente y en tiempo real entre el proceso padre `main_driver` y el software del control lateral. [6]

5.4.3. Alternativa al uso de Memoria Compartida

Sin embargo, en este proyecto se ha logrado una optimización significativa al reestructurar el código para que las funciones necesarias en el control lateral estén definidas dentro de una clase, eliminando la necesidad de memoria compartida.

Nueva Implementación con Scripts en Python

El nuevo enfoque implementado utiliza una clase de funciones en Python que interactúa directamente con la librería de comandos de EPOS (`libEposCmd.so`) para controlar el motor. Esta implementación ofrece varias ventajas:

- **Integración Completa:** Todo el control del motor, incluyendo la configuración, activación de modos de operación, y consulta de estados (posición, velocidad, corriente), se realiza dentro de una única clase de funciones en Python. Esto simplifica la arquitectura del sistema.

- **Facilidad en la Comunicación entre Procesos:** Ahora los datos se envían entre procesos Python dentro de un mismo proyecto utilizando variables globales y llamadas directas a funciones, lo que elimina la necesidad de memoria compartida.

Detalles de la Implementación

La clase de funciones en Python incluye las siguientes funcionalidades clave:

- **Inicialización y Configuración del Dispositivo:** Se configura el dispositivo EPOS4 mediante funciones como `open_device`, que establece la conexión, y `activate_position_mode` y `activate_velocity_mode`, que activan los modos de control de posición y velocidad, respectivamente.
- **Control de Movimiento:** La clase permite mover el motor a una posición específica (`move_to_position_speed`) o mantener una velocidad constante (`move_velocity`), lo que facilita la implementación de movimientos complejos sin necesidad de otros procesos adicionales.
- **Consulta de Estado del Motor:** A través de funciones como `get_position`, `get_velocity`, y `get_current`, se pueden obtener datos en tiempo real sobre la posición, velocidad y corriente del motor.
- **Modo de Homing:** La clase también maneja el modo de Homing (`homming_mode`), que permite encontrar y definir la posición inicial del motor.

Conclusión

Gracias a esta nueva implementación en Python, se ha simplificado significativamente el sistema. Al eliminar la dependencia de la memoria compartida y centralizar toda la lógica en una clase de funciones, no solo se mejora la eficiencia, sino que también se reduce la complejidad del código y se facilita el mantenimiento.

5.5. Simulación en MATLAB

Para el modelo cinemático y los algoritmos de guiado del vehículo se van a utilizar los descritos en el Trabajo de Fin de Máster de Samuel Pilar [6]. En él, se aplica un PID sobre la controladora EPOS4 para controlar y supervisar el movimiento lateral. El PID (Proporcional-Integral-Derivativo) es un controlador utilizado en sistemas de control automático para ajustar la salida de un proceso basado en el error entre un valor deseado y el valor actual, combinando tres acciones [25]:

- K_P (Proporcional): Ajusta la respuesta en función del error actual, afectando la rapidez de la corrección.
- K_I (Integral): Corrige en función del error acumulado a lo largo del tiempo, eliminando el error de estado estacionario.
- K_D (Derivativo): Responde a la tasa de cambio del error, ayudando a amortiguar las oscilaciones y mejorando la estabilidad.

Además de estas constantes, también se va a aplicar el algoritmo de Stanley, lo que nos obliga a tener en cuenta la constante de Stanley:

- $K_{STANLEY}$: Ajusta la sensibilidad del controlador Stanley en el seguimiento de una trayectoria, afectando la precisión y la reactividad en el alineamiento con la ruta deseada.

La simulación también está regida por otros parámetros que interfieren en el resultado de la misma. Estos parámetros están relacionados con las dinámicas del vehículo, como puede ser la velocidad con la que actúa el motor de control lateral, así como los retardos de procesamiento de las imágenes capturadas por la cámara:

- **Velocidad máxima**: Este parámetro fija las revoluciones por minuto máximas a las que puede girar el motor de control lateral, que gracias a los avances mencionados durante la memoria podemos aumentar esta velocidad a 6000 rpm a diferencia de los 3000 rpm con los que trabajaba Samuel Pilar [6].
- **Aceleración máxima**: Este parámetro ajusta la rapidez con la que el motor de control lateral llega a la velocidad máxima mencionada antes. En la simulación se puede fijar este parámetro como infinito si no queremos introducir ningún retardo ligado a la aceleración, como así lo hacía Samuel. En cambio, en este proyecto se ha buscado usar los datos más próximos a la realidad y se ha fijado una aceleración máxima de 10000 rpm/s.
- **Control del motor**: También se ha modificado el modo de funcionamiento del motor de control lateral, que estaba ajustado con el modo de posición y se ha cambiado al modo de velocidad.
- **Retardo de procesamiento**: Con este parámetro se define el tiempo que tarda en procesar la Humming Board, la imagen capturada por la cámara. Se ha estimado que el tiempo de retardo es de 0.2 segundos como ya estaba en el proyecto de Samuel.
- **Periodo de Captura**: Este tiempo indica la frecuencia con la que la cámara captura la información. Que en este caso se asume también que es de 0.2 segundos.

Cabe destacar que Samuel en su proyecto demostró que a pesar de modificar los parámetros relacionados con la captura de video (Retardo y periodo) a valores ideales, el resultado de la simulación seguía sin ser el óptimo. Por lo tanto, se confirma que control lateral, con su velocidad y su aceleración, tienen un papel fundamental en el resultado de la simulación que, como veremos a continuación, son determinantes.

Vamos a partir del simulador diseñado en [6] que permite dos modos de simulación:

- Única simulación: Se introducen manualmente los valores de las constantes, velocidad y radio de giro máximo.
- Barrido paramétrico: Se introducen la velocidad, radio y el rango de valores de las constantes. La simulación consiste en encontrar la combinación óptima de estas constantes.

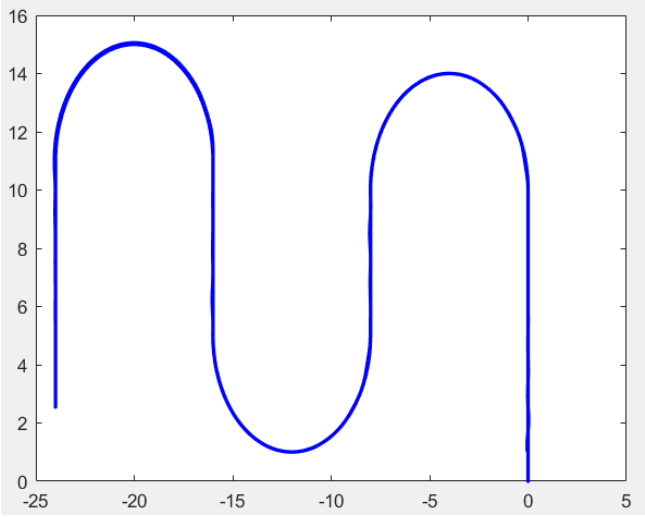


Figura 5.4: Ruta a realizar en la simulación

En la figura 5.4 se muestra la ruta que tiene que realizar el vehículo en la simulación, es una ruta que contiene 4 rectas y 3 giros de 180 grados. En este proyecto la idea es conseguir duplicar la velocidad de paso del vehículo con respecto al proyecto de Samuel Pilar, que era de 3 km/h [6] y uno de nuestros objetivos es conseguir ir a 6 km/h.

Se ha realizado el barrido paramétrico mencionado en el proyecto de Samuel [6] pero cambiando los valores extremos de las constantes. Que en este caso van a ser de:

- K_P: [3000, 15000] en saltos de 500.
- K_D: [3000, 15000] en saltos de 500.
- K_S: [0.5, 4] en saltos de 0.5.

El resultado de este barrido nos otorga que los valores que nos permiten tener la simulación más precisa posible son de:

- $K_P = 6000$
- $K_D = 8500$
- $K_S = 4$

La simulación con estos parámetros es la que se muestra en la figura 5.5:

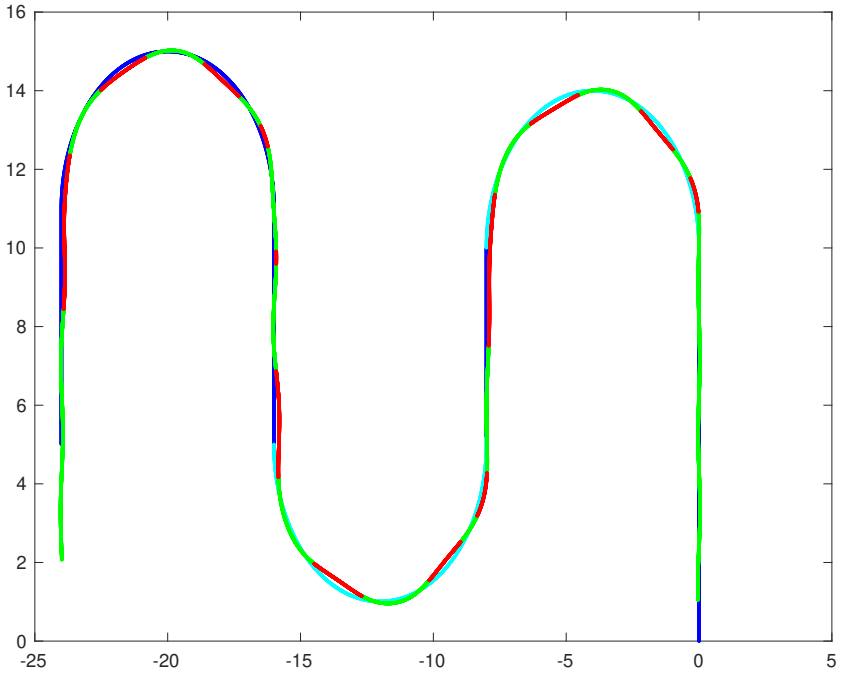


Figura 5.5: Simulación para 6km/h

En la figura 5.5 se aprecia que el vehículo puede seguir casi perfectamente la ruta marcada. La línea verde indica que la trayectoria que sigue el vehículo está dentro del margen máximo permitido que es de 0,085 m. A pesar de esto, los tramos en los que nos salimos de este margen suponen un error muy pequeño que no provocan un fallo en el seguimiento de la línea.

El problema viene cuando intentamos aumentar la velocidad que nos encontramos con limitaciones que exceden al control lateral. Y es que el uso de la cámara introduce un retardo que junto con el procesado de la propia imagen en la *Humming* hacen que el seguimiento de la trayectoria sea casi imposible cuando aumentamos la velocidad a 8-10 km/h. Como se puede ver en la figura [**?**], incluso habiendo recalculado los parámetros, si ejecutamos la simulación con 8 km/h vemos que el vehículo pierde por completo la ruta a seguir. Los nuevos valores de la simulación son:

- $K_P = 12500$

- $K_D = 8500$
- $K_S = 4{,}5$

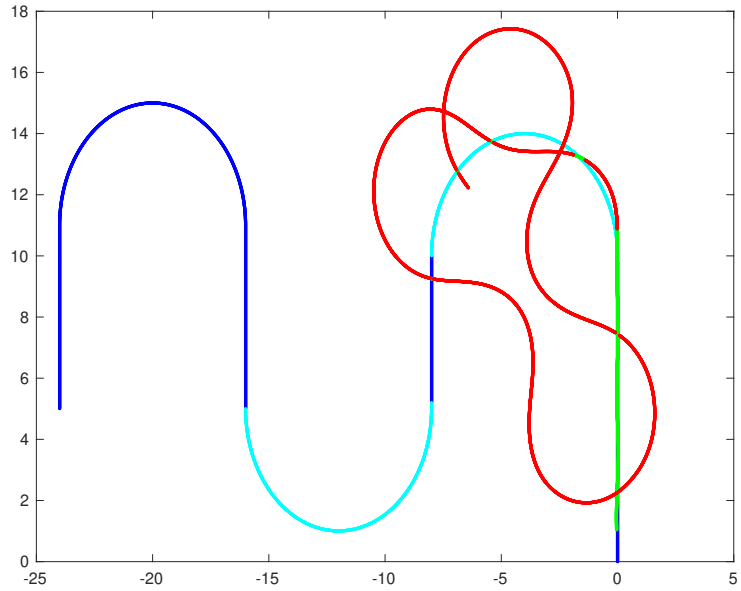


Figura 5.6: Simulación para 8km/h

En cambio, si ajustamos los parámetros de configuración relacionados con el procesado de la imagen, asumiendo que tenemos un ordenador con un mejor rendimiento que introduce un menor retardo, se aprecia que el vehículo es capaz de seguir prácticamente por completo la trayectoria marcada.

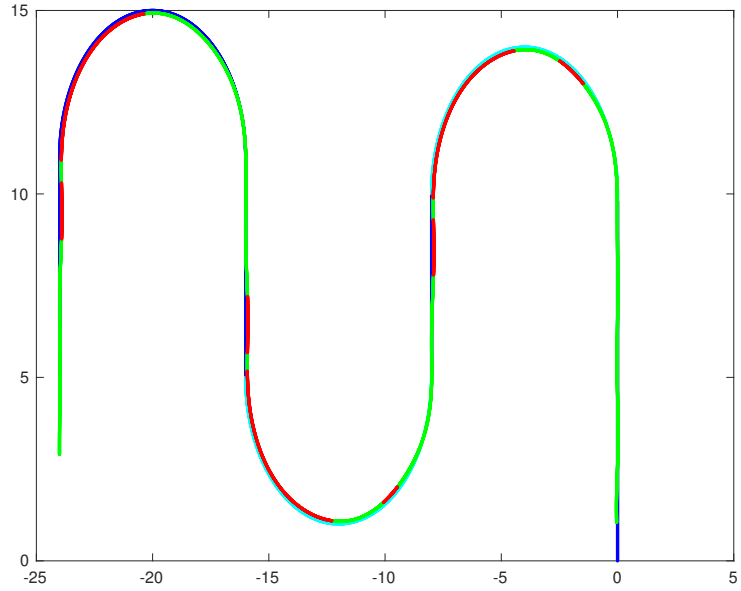


Figura 5.7: Simulación para 8km/h sin retardo de imagen

Capítulo 6

Conclusiones y líneas futuras

6.1. Conclusiones

Considerando los objetivos de este Trabajo de Fin de Máster definidos en la sección 1.2, se puede afirmar que se han cumplido las metas prefijadas. Este proyecto es una continuación del proyecto TwizyLine que, a pesar de haber finalizado, sigue sufriendo constantes implementaciones y mejoras que hace que los alumnos que trabajen en él puedan aprender y desarrollar un prototipo de vehículo autónomo.

Se ha rediseñado el conexionado eléctrico y las comunicaciones del vehículo, aportando simplicidad y claridad al cableado. Se ha partido de un vehículo que estaba en una etapa de desarrollo y no funcionaba correctamente, por lo que una gran parte del tiempo de este proyecto se ha destinado a conseguir que el vehículo funcionase de acuerdo con las versiones anteriores, como era la implementación de Ignacio Royuela [7]. Para poder hacer esto posible ha sido necesario programar el controlador de marchas, ya que no estaba operativo y requería de una nueva versión de software. Cabe destacar que, uno de los mayores problemas al que me he enfrentado ha sido el controlador del acelerador ya que nos ha supuesto una pérdida de tiempo importante debido a causas ajenas a este trabajo. Como ha sido la falta de documentación y *software* disponible que permitieran la trazabilidad del estado actual de este controlador. Pero finalmente se decidió hacer un nuevo controlador desde cero y el resultado ha sido más que positivo.

Otro punto importante del conexionado eléctrico ha sido el diseño y la implementación de la caja de conexiones (PowerBox). Esta caja ha permitido la agrupación y el control de la alimentación de todos los dispositivos electrónicos que hacen que el vehículo se pueda mover de manera autónoma. Permitiendo trabajar con los dispositivos de manera independiente, además de incluir una caja de fusibles que otorga seguridad frente posibles picos de tensión que puedan dañar los dispositivos. En cuánto a la colación de los módulos de control y comunicación, se ha decidido ponerlos en la parte trasera dentro del *rack* para poder tener un mejor acceso a ellos y evitando aglomeración de cables en las guanteras delanteras del Twizy.

Para las comunicaciones entre los dispositivos, he optado por incluir un *router* que permita la comunicación SSH entre los módulos de control y de comunicaciones, así como la posibilidad de poder conectarte con móvil vía Wifi al servidor MQTT para enviar las órdenes al vehículo. Esto ha aportado mayor agilidad en el desarrollo del *software*, ya que se disponía de comunicación directa entre los módulos y el PC en el que se estaba desarrollando el *software*.

Esta implementación ha supuesto el cambio del modo de control de la EPOS4, pasando de un control basado en la posición actual del motor a uno basado en la velocidad. Esto permite un movimiento lateral significativamente mayor una vez ya hemos sido capaces de controlar los límites de giro del motor. En concreto, en versiones anteriores con el otro modo de funcionamiento, el motor giraba a una velocidad de 3000 rpm que es casi la mitad de lo que puede girar ahora, que es de 5000

rpm. Este incremento de velocidad en el control lateral permite aumentar la velocidad longitudinal del vehículo, ya que podemos seguir la línea azul del suelo con una mayor facilidad, además de ser capaces de corregir la posición con una mayor brevedad.
Por último, me gustaría hacer una reflexión acerca de lo que me ha aportado a mí. En este proyecto me he enfrentado a problemas inesperados, los cuales, como hemos visto durante la memoria, suponen que un simple interruptor haga que no funcione un motor. O que trabajar sobre el proyecto de otro compañero y no disponer de la información acerca de cómo lo había hecho me ha supuesto un retraso a la hora de poder identificar el problema y de encontrar la solución. Pero a pesar de ser inconvenientes, también me han ayudado a prepararme mejor de cara a la vida laboral, ya que me han aportado constancia y capacidad de adaptación a nuevos problemas inesperados.

6.2. Líneas futuras

Este proyecto está en un punto en el que hay un abanico inmenso de posibles evoluciones. Ahora que la parte eléctrica está claramente definida, estable y robusta permite centrar los esfuerzos en el desarrollo de software y nuevas implementaciones como puede ser ROS como ya inició Carlota Gómez [5] o perfeccionando el programa principal del modo autónomo, añadiendo nuevas funcionalidades o casos de uso.

Además, otra vía de desarrollo puede ser la inclusión del LiDAR y balizas de ultrasonidos que, junto con un mapa del entorno, permitan localizar el vehículo y establecer su posición en el mapa. En cuánto al control lateral se pueden hacer evoluciones en la parte de la obtención de la posición de referencia, ya que no hemos conseguido obtener los resultados deseados, se podría estudiar el uso de sensores que indiquen la posición de referencia y mejorar el modo Homing de la controladora.

Por último, una línea a seguir puede ser la de buscar soluciones al problema de la inicialización del acelerador, ya que no fuimos capaces de identificar el problema. Por otro lado, también se puede prescindir de los módulos de control y de comunicaciones e integrarlo todo en un único dispositivo que se encargue de realizar todas las tareas. Reciba la información del LiDAR, el CAN del vehículo y tenga conexión con los distintos controladores, creando una especie de "cerebro" que gestione el modo autónomo del Twizy, y así también se podrían mitigar los problemas con el retardo del procesado del video.

Bibliografía

[1] TwizyLine, "http://twizyline.com, 25 de marzo de 2024," 2020.

[2] M. Martín Fernández, "http://twizyline.com/renault-and-the-university-of-valladolid-welcome-
25 de marzo de 2024," 2020.

[3] T-CUE, "Ganadores de la edición 2020 del concurso "iniciativa campus emprendedor". https://www.redtcue.es/campus/campus-2020/ganadores, 25 de febrero de 2024," 2020.

[4] S. Pilar Arnanz, "Back-end implementation for an automatized car parking," 2020.

[5] C. Gómez Diego, "Guiado de vehículo autónomo mediante tecnología lidar," 2022.

[6] S. Pilar Arnanz, "Servicios de vehículo conectado y conducción autónoma en un twizy," 2021.

[7] I. Royuela González, "Four level autonomous vehicle for an automatized parking," 2020.

[8] M. A. Kanaan, "Autonomous vehicle control by 3d camera in a twizy," 2022.

[9] N. Gordon-Bloomfield, "Renault twizy: The ultimate big boy's (and girl's) eco-toy?. https://web.archive.org/web/20190129011808/https://transportevolved.com/2014/01/17/renault-twizy-the-ultimate-big-boys-and-girls-eco-toy/, 25 de febrero de 2024," 2014.

[10] S. Cropley, "Renault twizy review. https://www.autocar.co.uk/car-review/renault/twizy, 25 de febrero de 2024," 2013.

[11] TopGear, "Renault twizy review. https://www.topgear.com/car-reviews/renault/twizy, 25 de febrero de 2024," 2015.

[12] E. Espinós, "Renault dejará de fabricar el twizy en septiembre... y ya se conoce su sucesor. https://www.autofacil.es/renault/twizy/renault-dejara-fabricar-twizy-septiembre-2023/635486.html, 25 de febrero de 2024," 2023.

[13] Renault, "Renault twizy e-tech 100 % eléctrico. https://www.renault.com.co/electricos/twizy/especificaciones.html, 25 de febrero de 2024," 2017.

[14] Waymo, "Waymo driver. https://waymo.com/intl/es/waymo-driver/, 25 de febrero de 2024," 2024.

[15] NHTSA, "Automated vehicles for safety. https://www.nhtsa.gov/vehicle-safety/automated-vehicles-safety, 25 de febrero de 2024," 2023.

[16] RACE, "Así son los cinco niveles de la conducción autónoma. https://www.race.es/niveles-conduccion-autonoma, 25 de febrero de 2024," 2022.

[17] TwizyLine, "Twizyline, drawing the future. http://twizyline.com/twizylinedrawingthefuture/, 25 de marzo de 2024," 2020.

[18] B. Jiménez Padilla, *Técnicas básicas de electricidad de vehículos (MF0624_1)*. Málaga: IC Editorial, 2012.

[19] H. Budde-Meiwes, J. Drillkens, B. Lunz, J. Muennix, S. Lehner (maiden name Rothgang), J. Kowal, and D. U. Sauer, "A review of current automotive battery technology and future prospects," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 227, 2013.

[20] C. Dunkel, "Adaption eines versuchsträgers und entwicklung von funktionsprototypen für automatisierte fahrfunktionen," 2017.

[21] M. Group, "Epos4 70/15, electrónica digital de control de posición. `https://www.maxongroup.es/maxon/view/product/control/Positionierung/594385`, 25 de mayo de 2024," 2020.

[22] M. Group, "Epos communication guide.," 2020.

[23] M. Group, "Epos firmware specification.," 2020.

[24] M. Group, "Epos command library.," 2020.

[25] T. Mansour and T. Mansour, *PID control, implementation and tuning*. Rijeka, Croatia: IntechOpen, 2011.