

CONTROL EN VEHÍCULOS

Universidad de Sevilla

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA



**Proyecto Vehículo Autónomo: Detección y
Seguimiento de Carril basado en Procesamiento
de Imagen**

Autores:
Álvaro García Lora
Sergio León Doncel
Juan Antonio Cano Vílchez

Junio 2024

Índice general

1. Objetivos y alcance	1
1.1. Introducción	1
1.2. Objetivos	2
1.3. Planteamiento del problema	2
1.4. Aplicaciones	3
2. Hardware, Sistema Electrónico y Comunicaciones	4
2.1. Coche RC original	4
2.2. Mando controlador	5
2.3. Vehículo prototipo	7
2.3.1. Electrónica bajo nivel	8
2.3.2. Electrónica alto nivel	8
2.4. Comunicaciones	10
2.5. Complicaciones abordadas	12
3. Software y Control	14
3.1. Arquitectura implementada	14
3.2. Programación del microcontrolador	16
3.3. Programación de alto nivel	17
3.3.1. Procesamiento de imagen y detección de carril	17
3.3.2. Obtención de líneas guías	19
3.3.3. Obtención de la pendiente	21
3.3.4. Lógica de control	22
4. Pruebas de funcionamiento y Resultados	24
5. Conclusiones y Trabajo Futuro	27
5.1. Limitaciones actuales	27
5.2. Ampliaciones a futuro	28
6. Archivos adjuntos	29

Resumen

Este documento pretende reportar el procedimiento y resultados conseguidos durante el desarrollo del proyecto de la asignatura Control en Vehículos para el Máster de Ingeniería Electrónica, Robótica y Automática.

Partiendo de un coche RC de juguete con configuración Ackermann, se utilizará su chasis y sus motores, añadiendo todo el sistema de componentes electrónicos necesario para dar soporte a la funcionalidad de detección y seguimiento de carril mediante cámara, empleando procesamiento de imagen y visión computacional. Esto permitirá que el vehículo se desplace dentro de una vía delimitada de manera autónoma. Los experimentos se realizarán en una pista diseñada específicamente para verificar el funcionamiento del sistema.

Capítulo 1

Objetivos y alcance

1.1. Introducción

En las últimas décadas, el campo de la conducción autónoma ha avanzado significativamente, impulsado por los rápidos progresos en tecnologías de percepción, procesamiento de datos y aprendizaje automático.

Entre las diversas tecnologías emergentes, la detección de carriles mediante cámaras se ha destacado como una solución crucial para la navegación y el control de vehículos autónomos. Este enfoque se basa en el uso de cámaras y algoritmos de visión por computadora para identificar y seguir las marcas del carril en tiempo real, permitiendo que los vehículos mantengan su trayectoria y respondan de manera adecuada a las condiciones del camino.

Los sistemas de detección de carriles han evolucionado considerablemente, desde los métodos basados en algoritmos simples de detección de bordes hasta los avanzados modelos de redes neuronales profundas que pueden aprender y adaptarse a una amplia variedad de condiciones viales y ambientales.

Estos avances han permitido una mayor precisión y fiabilidad en la identificación de carriles, incluso en situaciones complejas como carreteras con marcas desgastadas, iluminación variable y obstáculos temporales.

La integración de estas tecnologías en vehículos autónomos ofrece múltiples beneficios, incluyendo mejoras en la seguridad vial a través de la prevención de salidas involuntarias del carril y la reducción de accidentes causados por errores humanos.

Además, facilita el desarrollo de sistemas de asistencia avanzada al conductor (ADAS), que proporcionan soporte en la conducción diaria y ayudan a mitigar la fatiga y el estrés del conductor.



Figura 1.1: Tarea de seguimiento de carril en vehículos autónomos

1.2. Objetivos

El objetivo final de este proyecto es crear un sistema prototipo de coche, totalmente propio, que realice la tarea de detección y seguimiento de carril de forma autónoma.

Para ello, se plantea hacer uso del chasis y motores de un viejo coche de radiocontrol, al cual se le incorpora todo un sistema electrónico a bordo, diseñado e integrado de forma propia, necesario para cumplir la funcionalidad requerida, tanto a nivel de sensórica y actuadores, así como de soportar la capa de comunicaciones inalámbricas.

A nivel software, se desarrollarán los programas necesarios tanto a bajo como alto nivel, desde el control y comunicaciones hasta el procesamiento de imagen y toma de decisiones en la estrategia de control para el movimiento del vehículo. Esto supondrá por tanto, la programación efectiva de microcontroladores y microprocesadores, donde una adecuada adaptación a los recursos disponibles será necesaria.

Al trabajar con un sistema físico real, a diferencia de una simulación, se deben abordar diversas problemáticas asociadas a los componentes físicos y al entorno del mundo real. Entre estas problemáticas se incluyen variaciones en la luminosidad, límites de los actuadores, restricciones de los sensores, perturbaciones externas y desafíos inherentes al movimiento real del vehículo.

1.3. Planteamiento del problema

Para lograr los objetivos mencionados en el punto anterior, hemos de descomponer el problema en los siguientes aspectos:

- **Hardware y Comunicaciones:** Adquisición e integración de sistemas electrónicos para crear un mando controlador y vehículo prototipo funcional. Montaje

del sistema de comunicaciones completo. Los detalles serán mostrados en el Capítulo 2.

- **Software y Control:** Desarrollo del código de bajo nivel para comunicaciones y control de dispositivos electrónicos. En una capa de abstracción superior, procesamiento de imágenes y aplicación de una serie de técnicas de visión computacional para poder extraer la información necesaria para la toma de decisiones e implementación de un control del vehículo. Todo ello se encuentra documentado en el Capítulo 3.
- **Testing:** Pruebas de funcionamiento y extracción de resultados sobre un recinto apropiado para la tarea de vehículo autónomo que se presenta. Necesidad de fabricación de pista con carriles para generar un entorno de test y validación. Todas las ideas relacionadas son descritas en el Capítulo 4.

1.4. Aplicaciones

El desarrollo de un prototipo de vehículo equipado con una cámara capaz de detectar y seguir el carril tiene un gran interés por diversas razones.

En primer lugar, representa un paso significativo hacia la conducción autónoma, ofreciendo potenciales mejoras en la seguridad vial a través de sistemas avanzados de asistencia en carretera y ayuda a la conducción.

Estos sistemas pueden reducir la probabilidad de accidentes al mantener el vehículo dentro de su carril y reaccionar rápidamente ante imprevistos.

Además, la tecnología de detección de carriles puede facilitar la navegación en condiciones adversas, como niebla o lluvia intensa, donde la visibilidad es limitada.

También abre la puerta a innovaciones en la eficiencia del tráfico y la reducción de congestiones mediante una conducción más precisa y coordinada.

Este proyecto no solo contribuye al avance de la tecnología de vehículos autónomos, sino que también tiene el potencial de mejorar significativamente la experiencia de conducción y la seguridad en las carreteras.



Figura 1.2: Asistencia a la conducción y seguridad en carretera

Capítulo 2

Hardware, Sistema Electrónico y Comunicaciones

2.1. Coche RC original

Como punto de partida se ha tomado un coche radiocontrol inutilizado de la marca Nikko, escala 1:16, del cual se ha aprovechado el chasis, carcasa, motores y mecanismo de dirección.



Figura 2.1: Coche RC original. Punto de partida

La configuración de movimiento es Ackermann, lo cual lo hace apropiado como prototipo de vehículo real. La tracción es trasera únicamente, siendo las ruedas delanteras las encargadas de la dirección. En ambos lugares contamos con motores DC con tren de engranajes como reductora. En el caso de la dirección, presentamos una limitación sobre el giro, puesto que no puede hacerse gradual, ya que no se tiene un servomotor en el mecanismo que efectúa el movimiento de las ruedas.

Por tanto, las posiciones estables disponibles son: neutra (no hay tensión aplicada al motor), límite derecho y límite izquierdo (posiciones extremas del mecanismo correspondientes a aplicar la tensión efectiva al motor en un sentido o en su contrario).

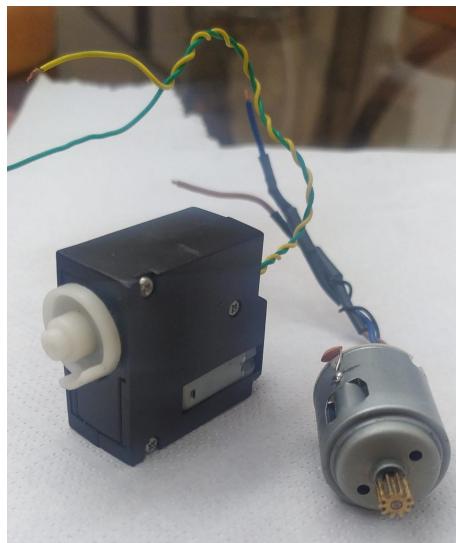


Figura 2.2: Motores originales reutilizados en el proyecto

2.2. Mando controlador

Con vistas a tener un mando controlador que nos sirva para controlar el vehículo en modo manual o actuar como modo de emergencia, se ha desarrollado la superficie de control que se presenta en esta sección. La comunicación de todo el sistema será inalámbrica vía WiFi, como se explicará más adelante en el informe, por lo que este mando desarrollado tiene la capacidad de conectarse a la red que creamos.



Figura 2.3: Mando controlador desarrollado

A continuación se listan los componentes electrónicos con los que cuenta:

- **Joysticks 2 ejes + botón:** Superficies de control para movimientos de avance / retroceso y giros. Se usa solamente un eje de cada joystick (vertical en el caso del izquierdo y horizontal en el derecho) por motivos de facilidad en el manejo del vehículo. Además los pulsadores disponibles en cada joystick son usados para funcionalidades de encender / apagar panel de luces leds y tocar el claxon (hacer sonar un buzzer interno).
- **Potenciómetro 10KOhm:** Usado para controlar la velocidad de movimiento del vehículo.
- **ADC 8-bits PCF8591:** Convertidor analógico/digital que cuenta con 3 canales, usado para convertir los niveles de tensiones analógicos de los ejes usados de los joysticks y potenciómetro en valores digitales para transmitirlos al microcontrolador central del mando mediante protocolo I2C.
- **Placa Wemos WiFi D1 Mini:** Placa ligera de 5 g de peso que cuenta con microcontrolador central ESP8266 donde se encuentra el programa de bajo nivel encargado de gestionar las señales recibidas de las superficies de control y enviar los comandos correspondientes de orden para movimiento de los motores del vehículo. La placa a su vez cuenta con un módulo WiFi integrado lo que la hace muy práctica para conexión al punto de acceso de la comunicación inalámbrica que estableceremos.

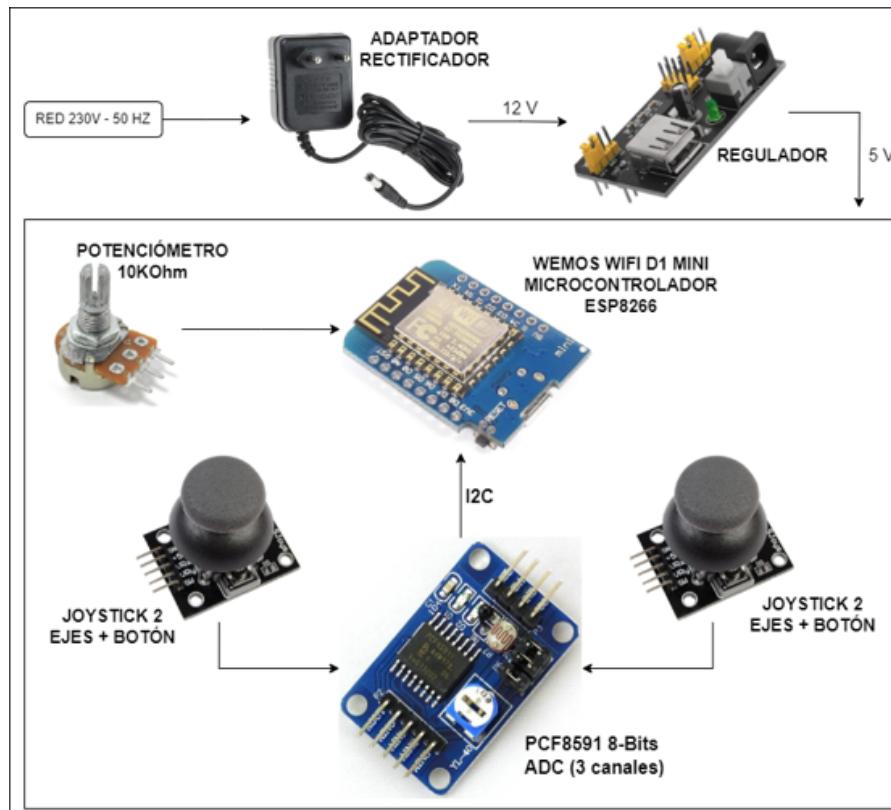


Figura 2.4: Esquema de relaciones y conexionado de componentes del mando

- **Rectificador y regulador:** Para usar la alimentación de la red se usa un adaptador rectificador con salida a 12 V. Tras esto se dispone de una etapa reguladora que baja la tensión a 5 V, la cual es necesaria para alimentar a la electrónica previamente presentada.

Nota: Es posible alimentar el mando a baterías, simplemente se sustituye el adaptador por unas baterías con tensión nominal superior a 5 V. Se han realizado pruebas con baterías LiPo 2S de 600mAh de capacidad con capacidad de descarga de 10C.

2.3. Vehículo prototipo

Nuestro sistema móvil prototipo ha sido construido de forma propia gracias al diseño e integración de su sistema electrónico de a bordo, el cual ha dado el soporte para desarrollar la capa software que se abordará en el siguiente capítulo. Para ello ha sido necesario un estudio de los componentes necesarios, sus conexionados, la organización efectiva de los espacios disponibles en el chasis original y la distribución de pesos en la estructura. Las dimensiones del prototipo resultan: 30 cm largo x 12 cm ancho x 12 cm alto. Su peso final es de aproximadamente 1 Kg.



Figura 2.5: Apariencia exterior del vehículo prototipo construido

Podemos dividir el sistema dos módulos principales: bajo y alto nivel. Cada uno de ellos cuenta con su propio procesamiento. De esta forma desacoplamos la capa de procesamiento de imagen y toda de decisiones de la capa encargada de usar la electrónica final destinada al movimiento del vehículo y otras funcionalidades añadidas.

2.3.1. Electrónica bajo nivel

El módulo de bajo nivel es el encargado de recibir los comandos procedentes del alto nivel que le indican como debe comportarse el vehículo en cada momento. Para lograr esto, se dispone de la siguiente serie de componentes:

- **Motores DC:** Motores originales del juguete RC, ya presentados con anterioridad.
- **Sensor LDR:** Detectar las condiciones de luminosidad para toma de decisión de luces automáticas.
- **Panel leds:** Conjunto de leds dispuestos en serie y limitados en corriente con resistencia diseñada para que funcionen en condiciones óptimas de polarización. Su función es aportar luz cuando se le indique o bien la medida del sensor LDR esté por debajo de cierto umbral (no haya suficiente luminosidad en el escenario).
- **Buzzer:** Usado como señal sonora de aviso para el cambio entre manual y automático. También puede usarse a modo de claxon en modo manual.
- **Driver L298:** Control de dirección y potencia para motores. Su manejo se realiza por medio de los pines disponibles usando modulación PWM (velocidades) y señales digitales para cambios de sentido de giro o parada.
- **Placa Wemos WiFi D1 Mini:** Se trata de una placa idéntica a la usada en el mando controlador. Se ha decidido volver a usar el mismo modelo al ser suficiente en número de pines, capacidad de cómputo por parte del microcontrolador ESP8266 y por contar con el módulo WiFi integrado para las comunicaciones. Su peso ligero y pequeñas dimensiones la hacen ser una solución muy versátil. Cuenta por tanto con el programa que gestiona los comandos de control que se reciben desde el mando y microprocesador de alto nivel. En función dichas órdenes, controla el driver de los motores, el buzzer y el panel de leds (según medida de sensor LDR).
- **Batería LiPo:** Bateria 2S 1300mAh para alimentar a todo el sistema de bajo nivel. Cuenta con un poder de descarga de 60C. Tiene capacidad suficiente y es capaz de gestionar las potencias variables requeridas por los motores y resto de electrónica.

2.3.2. Electrónica alto nivel

Por encima del bajo nivel, se encuentra una capa de abstracción de nivel superior, encargada de procesar las imágenes para extraer la información útil para realizar la tarea de seguimiento del carril. Este módulo cuenta con una placa con microprocesador, con mucha más capacidad que el microcontrolador de bajo nivel con el que se comunica. Los componentes presentes son:

- **RaspberryPi 4B 1GB RAM**: Famosa placa con microprocesador que supone el cerebro central de nuestro prototipo de vehículo autónomo. A ella se le conecta la cámara empleada mediante puerto USB. Se comunica con los ESP8266 del mando controlador y del bajo nivel del coche vía WiFi.
- **WebCam C270 Logitech**: Cámara usada para la visión del vehículo. Tiene una resolución HD de 720P y es capaz de funcionar a 30 fps.
- **Banco pilas 6V**: Contamos con un total de 4 pilas recargables en serie de 1.5V cada una. De esta forma tenemos alimentación independiente entre bajo y alto nivel puesto que la placa Raspberry necesita una alimentación estable que debe ser independiente de las variaciones de potencias requeridas por los motores.

En el siguiente esquema de la Figura 2.6 pueden visualizarse la disposición de los distintos elementos descritos en esta sección.

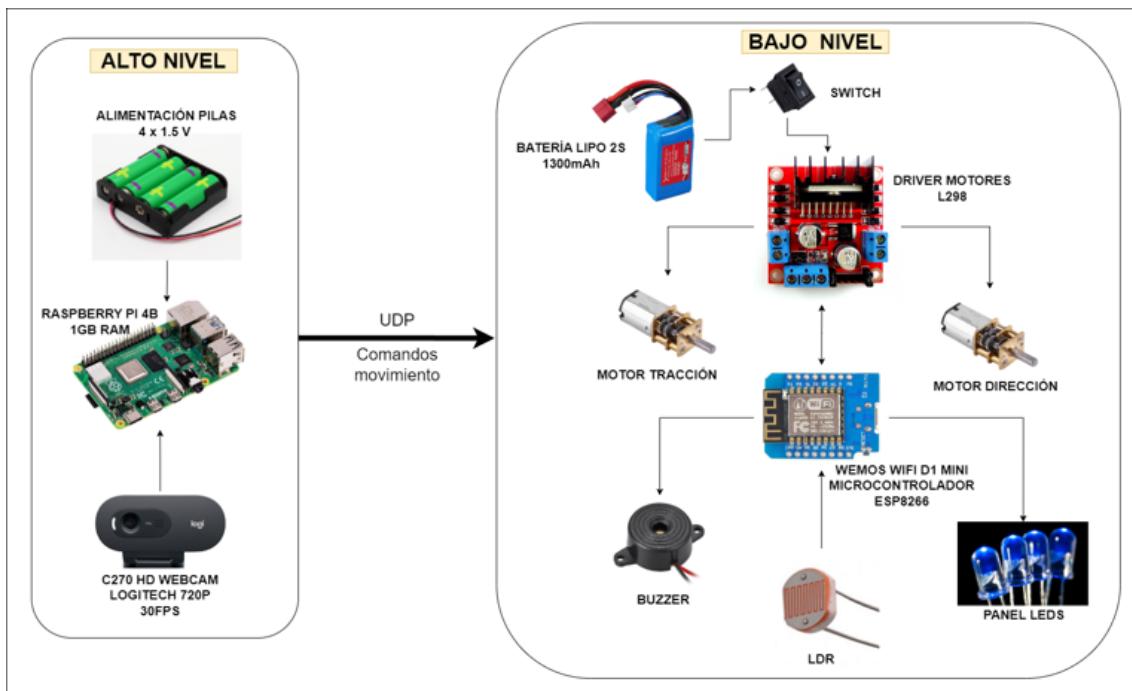


Figura 2.6: Esquema conceptual de conexionado de componentes internos del sistema (alto y bajo nivel).

Para dar al lector una mayor comprensión sobre el montaje físico que se ha realizado en este proyecto se recogen una serie de imágenes intermedias tomadas durante el proceso de implementación.

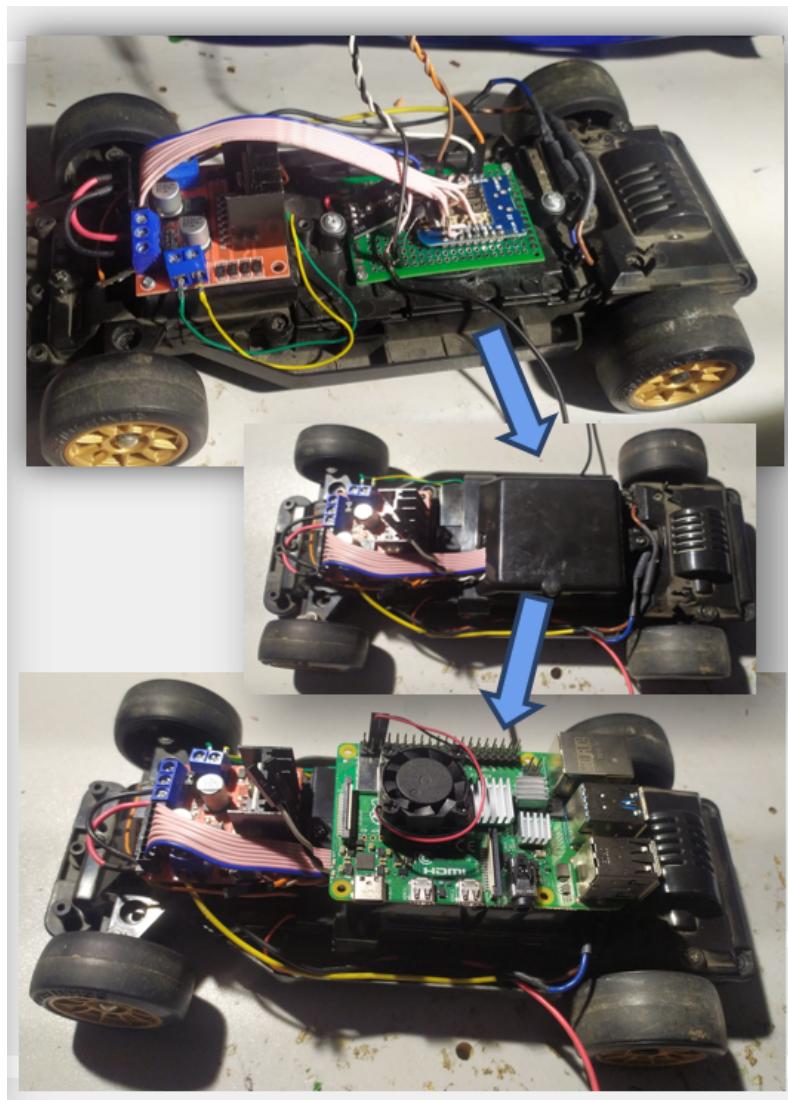


Figura 2.7: Pasos construcción: Distribución interna de componentes

2.4. Comunicaciones

La arquitectura de comunicación del sistema se ha diseñado para ser completamente inalámbrica utilizando tecnología WiFi. En el centro de esta configuración se encuentra un router que actúa como punto de acceso para todos los dispositivos involucrados en el proyecto, formándose una red propia a nivel local.

Para la transmisión de datos, se ha optado por el protocolo UDP (User Datagram Protocol). UDP se elige por su capacidad de transmitir datos de manera rápida y eficiente, sin la sobrecarga adicional que acompaña a otros protocolos como TCP (Transmission Control Protocol). Aunque UDP no garantiza la entrega de paquetes, su velocidad y baja latencia lo hacen ideal para aplicaciones en tiempo real como la detección y seguimiento de carriles, donde la rapidez de la comunicación es crucial.

El router no solo facilita la comunicación entre los dispositivos, sino que también puede gestionar la asignación de direcciones IP y el encaminamiento de paquetes de datos, asegurando que cada mensaje llegue a su destino correcto.

Esta metodología permite escalar el sistema fácilmente, añadiendo nuevos dispositivos o sensores sin necesidad de modificar la infraestructura básica de comunicación.

En la Figura 2.8 podemos ver el resumen de los dispositivos que intervienen en el sistema completo. El mando controlador envía comandos de control a la placa de alto nivel del coche quien decide si los reenvía a la placa de bajo nivel (en caso de que esté seleccionado el modo manual) o es ella misma quien se encarga de enviar los comandos de movimiento (modo autónomo).

El cambio entre manual y automático se realiza al pulsar un botón destinado para ello en el mando controlador (cuando esto se hace, el buzzer suena indicándolo).

Además contamos con un PC conectado a la misma red WiFi que controla la placa RaspberryPi por SSH y VNC. Gracias a esto tenemos una pantalla y teclado remoto para visualizar, en la GUI del SO interno del microprocesador, ventanas con imágenes tanto originales captadas por la cámara como procesadas por nosotros, en las distintas fases. Esto nos ha sido totalmente necesario para poder desarrollar, depurar y aportar resultados de funcionamiento.

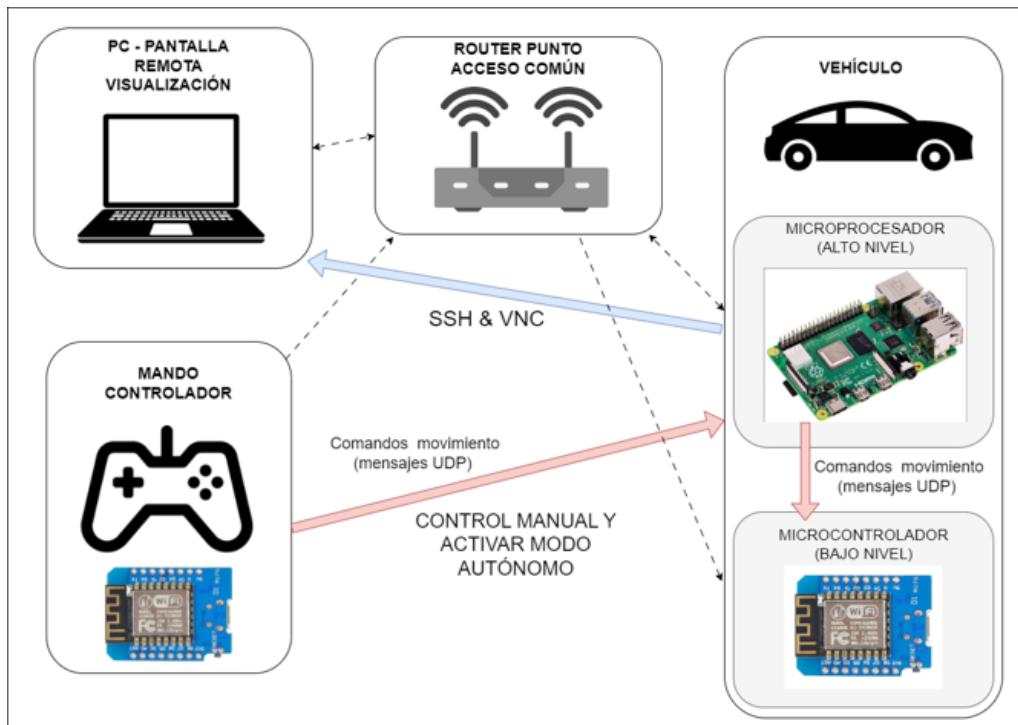


Figura 2.8: Esquema de comunicaciones y dispositivos involucrados

2.5. Complicaciones abordadas

Durante el desarrollo de la parte física y testeo de capacidades del sistema surgieron una serie de dificultades destacables que, debido al tiempo y trabajo dedicado, merecen mencionarse y tenerse en cuenta.

ESP32-CAM & Micropython: Limitación de cómputo

En una etapa temprana del proyecto, se optó por utilizar la placa ESP32-CAM, que integra un microcontrolador y una minicámara, considerando que era ideal por su tamaño compacto, sus prestaciones y su bajo coste. Por ello, se invirtió tiempo y esfuerzo en configurar la placa y flashear un firmware basado en MicroPython, permitiendo al microcontrolador acceder a funcionalidades y librerías conocidas para el procesamiento de imágenes en Python. MicroPython es una versión simplificada de Python diseñada para trabajar con microcontroladores.

Sin embargo, tras completar el montaje y configuración, las pruebas con la cámara revelaron que la capacidad de cómputo de la ESP32-CAM no era adecuada para nuestras necesidades. Esto nos obligó a cambiar a un microprocesador con mayores prestaciones: la Raspberry Pi. Inicialmente, se buscaba combinar funciones de alto y bajo nivel en un único microcontrolador, pero esta estrategia resultó inviable. Por lo tanto, decidimos implementar una solución de dos niveles, la cual se ha presentado en este informe.



Figura 2.9: ESP32-CAM

Radio de giro

Una vez construido el sistema completo, en la fase de pruebas en control manual, se comprobó que el radio de giro del coche era demasiado grande. Se necesitaba mucho espacio para poder dar vueltas completas, lo cual limitaba la capacidad de maniobrabilidad y suponía disponer de un espacio para pruebas, y por tanto una pista experimental, de dimensiones inviables. La causa de este problema residía en un desequilibrio en la distribución del peso del vehículo, lo que provocaba que las ruedas delanteras no tuvieran un agarre adecuado en el terreno, resultando en deslizamientos.

Para solucionarlo se fabricó una pieza a partir de pletina de hierro, que sirviera de contrapeso, la cual se dispuso en la parte delantera y a la vez se usó para soporte de la cámara.

Ajuste posición de cámara

La vista en primera persona del vehículo, es decir, las imágenes capturadas por la cámara, captaban demasiado cerca las líneas de la pista de pruebas, por lo que fue necesario un cambio de la posición inicial donde se dispuso. Aún así, aceptamos el reto de no poder tener la cámara demasiado alta, debido a las dimensiones reducidas del vehículo. Con una vista superior de la escena, posiblemente hubiera sido más sencillo y robusto el reconocimiento del carril.

Distribución y aprovechamiento de espacios

Debido a las pequeñas dimensiones del vehículo, implementar físicamente todo el sistema electrónico mostrado a supuesto un verdadero desafío. Además de la selección de componentes fiables que fuesen viables en términos de dimensiones y peso, se ha tenido que plantear cual era la mejor distribución de los mismos en el espacio disponible. Esto ha supuesto tener que iterar en varias ocasiones para lograr llegar a una solución compacta, equilibrada y fiable de cara al desarrollo del software y pruebas de funcionamiento.

Capítulo 3

Software y Control

3.1. Arquitectura implementada

Para alcanzar los objetivos mencionados anteriormente, se propone la implementación del **software embebido** bajo una **arquitectura master-slave**.

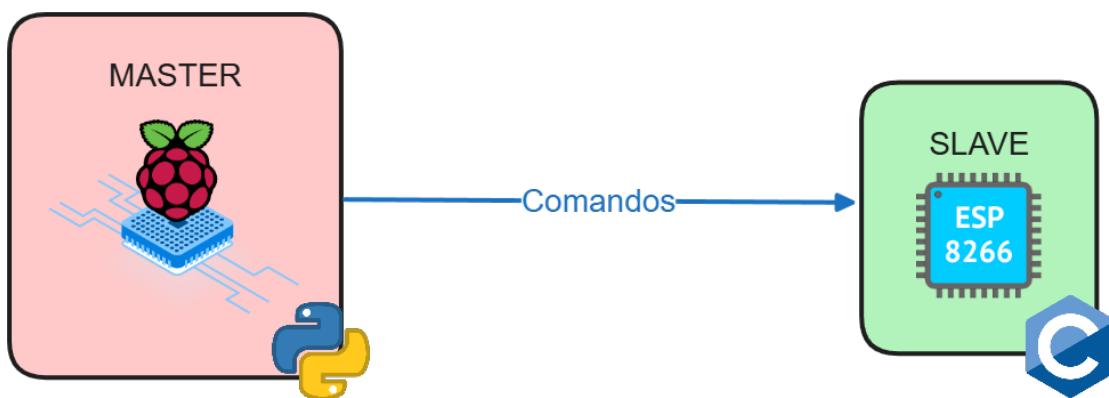


Figura 3.1: Esquema de arquitectura software

Esta se encontrará distribuida en:

- Raspberry PI: Se aprovechará la **alta capacidad de cómputo** de esta SBC para llevar a cabo el procesamiento de imagen y la toma de decisiones que requiere la solución planteada para el problema.

Su programación se llevará a cabo en el lenguaje **Python**, dada la **versatilidad** que ofrece su amplio abanico de librerías orientadas a la visión artificial.

- Wemos Wifi D1 Mini: Permitirá hacer uso a **bajo nivel de su interfaz GPIO** para asegurar la actuación sobre la electrónica del vehículo, así como dotar al sistema de **robustez en tiempo real** al no ejecutarse en paralelo con otras tareas del sistema operativo.

Por estos mismos motivos, el lenguaje elegido será **C**, ya que a pesar de que se ha estudiado la posibilidad de utilizar microPython, las desventajas en pérdida de control a bajo nivel no compensa su única ventaja, la sencillez y comodidad.

Por otro lado, para definir las especificaciones de nuestra solución, es necesario definir una **frecuencia de actualización** para cada una de las **tareas** que se llevan a cabo. Podemos identificar:

- **Procesamiento de imágenes y toma de decisiones:** Desde la perspectiva de control, una mayor tasa de refresco asegura un menor tiempo de respuesta y por tanto, un mejor comportamiento.

Por ello, la frecuencia de actualización vendrá determinada por el límite de cómputo. Sin embargo, esto también dependerá de la **resolución** de las imágenes que se utilice, que influirá en el rendimiento de la visión artificial. Gracias a que este es un parámetro a nuestra elección, se fija en **320x240 (QVGA)** para conseguir cerrar el bucle a una **frecuencia estable de 15 Hz**.

- **Recepción/Envío de mensajes:** En estos mensajes se codificará en **formato UTF-8** el byte stream necesario para alojar los comandos de actuación en el vehículo. En un principio, se podría pensar en sincronizarlo a la misma frecuencia que la tarea anterior, ya que la actuación calculada por el control no cambiará prematuramente.

Sin embargo, hay 2 motivos por los que esto sería un error:

- Primeramente, en el sistema se propone implementar la característica de un **control manual remoto** que sirva como alternativa ante emergencias. Esto hace que la acción de actuación pueda variar en cualquier instante, por lo que es importante asegurar una tasa lo suficientemente rápida como para que no se produzca un 'delay' entre un 'input' y la respuesta asociada que complique el manejo.
- Por otro lado, la comunicación se lleva a cabo por el **protocolo UDP**, el cuál no asegura la correcta recepción por parte del destinatario del mensaje. Por tanto, en la práctica es recomendable triplicar la tasa de envío para provocar redundancias que aseguren la correcta transmisión.

De las razones anteriores, la primera resulta la más exigente. En nuestro caso, se estima lógico una **tasa de 100 Hz** con este propósito.

- **Actuación a bajo nivel:** En este caso, sí es posible recurrir a la sincronización con la tarea anterior, ya que la actualización de niveles lógicos en la GPIO no tendrá sentido si no ha variado la decisión de control. Por tanto, se fija también una **frecuencia de 100 Hz**.

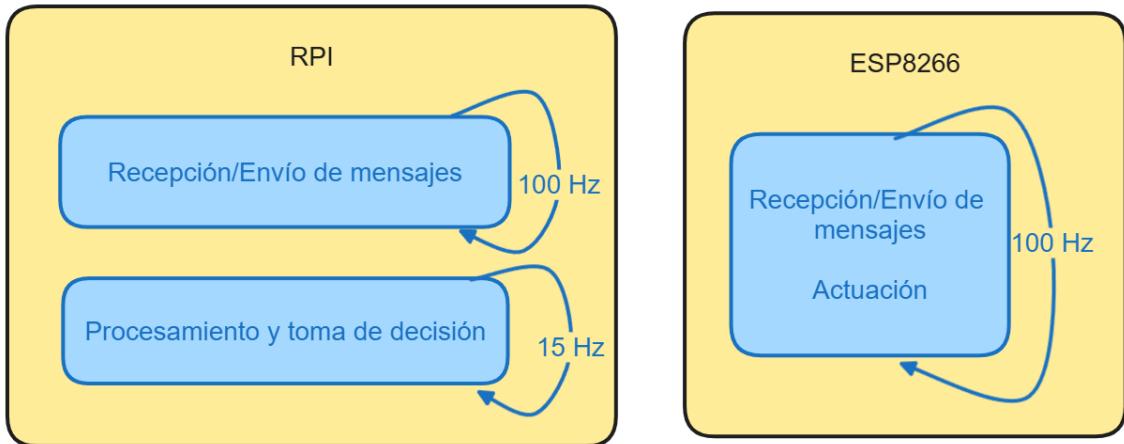


Figura 3.2: Distribución de tareas

En base al planteamiento anterior, es necesario el uso de **hilos** para conseguir la **parallelización de las tareas** y hacer uso de la **concurrencia** del microprocesador de la Raspberry Pi.

Esto a su vez conlleva a **mecanismos de sincronización** que aseguren la **integridad de la memoria compartida**, ya que existirán variables que pueden ser leídas/escritas por ambos hilos a la vez, lo que en la práctica conlleva condiciones de carrera que derivan a comportamientos inesperados.

Para solventar esto, se recurrer al uso de las **librerías Threading y Semaphore**, esta última con la intención de disponer de un mecanismo de bloqueo que no permite a varios hilos acceder a una misma variable a la vez.

3.2. Programación del microcontrolador

Partiendo de las especificaciones anteriores, se desarrolla el firmware que será flasheado en el microcontrolador a bordo del vehículo, cuyo objetivo será proporcionar la funcionalidad de control remoto del vehículo utilizando la red Wi-Fi, añadiendo a su vez capacidades adicionales como el control de luces y la generación de sonidos.

Para ello, se hace uso de las siguientes librerías:

- Arduino Framework: Capa HAL de la placa.
- Wire y SPI: Para utilizar los protocolos I2C y SPI de comunicación con periféricos externos.
- ESP8266WiFi y WiFiUDP: Para la conexión Wi-Fi y el protocolo UDP.
- Math, stdio y stdlib: Para cálculos matemáticos y manejo de cadenas.
- UDP y ESP8266_Utils: Para las funciones de conexión Wi-Fi y UDP.

Además, se incluye el archivo de cabecera 'config.h' donde se definen los datos de acceso a la red.

Por otro lado, se definen los pines en modo digital del ESP8266 asociados a cada funcionalidad del vehículo:

- IN1, IN2, IN3, IN4: Para controlar la dirección de los motores.
- ENA, ENB: Para controlar la velocidad de los motores.
- BUZZER: Para generar sonido a modo de claxon.
- LUZ: Para el apagado/encendido del array de LEDS delantero del vehículo.

Inicialmente, se realiza una configuración inicial configurando los pines de salida digital, estableciendo la conexión wifi, y abriendo el socket UDP.

Posteriormente, se ejecuta el bucle principal del programa de manera indefinida, donde se ejecuta repetidamente la siguiente secuencia:

- Recepción de comandos: En el bucle principal, se reciben y procesan comandos UDP.
- Ajuste del estado del vehículo: Decodificando el payload recibido en el mensaje en formato UTF-8, se ajustan la dirección, la tracción, el estado de las luces y la reproducción de notas.

3.3. Programación de alto nivel

Para la implementación del código de alto nivel, se segmenta el código en el archivo del programa principal, el script 'controlSW.py', el archivo de configuración con los parámetros de configuración del usuario 'userConfig.py', y la librería 'helpers.py' conteniendo la implementación de las funciones auxiliares del programa.

3.3.1. Procesamiento de imagen y detección de carril

Captura de Imágenes

Para la sección asociada al procesamiento de imagen, se emplea la librería OpenCV junto con algunas funciones adicionales para el análisis de imágenes. El proceso comienza con la captura de imágenes a través de la webcam Logitech, estableciendo una resolución por defecto de 320x240 píxeles. Cada fotograma se obtiene en formato BGR (Blue, Green, Red). El tiempo de captura de cada fotograma está determinado por el tiempo de ejecución del programa del microprocesador y la frecuencia a la que se establece la captura. La imagen de captura se muestra en la Figura 3.3.



Figura 3.3: Captura de imagen original

Conversión a Escala de Grises

El primer paso, una vez obtenido un fotograma, es convertir el formato original BGR a escala de grises (ver Figura 3.4). Esta conversión facilita la posterior manipulación de la imagen, ya que simplifica los datos al reducir la información a un solo canal de intensidad de luz.



Figura 3.4: Imagen en escala de grises

Binarización de la Imagen

El siguiente paso es aplicar una binarización a la imagen en escala de grises utilizando un umbral (threshold). Este umbral tiene un límite superior e inferior que vienen ajustados por parámetros configurados para el entorno deseado. La binarización crea una imagen en blanco y negro donde los píxeles son clasificados como 0 (negro) o 255 (blanco). Posteriormente, se invierte la imagen para que las líneas del circuito aparezcan como "1" (blanco) y permitan ser procesadas posteriormente. Comprobar la Figura 3.5.



Figura 3.5: Imagen binarizada e invertida

Operaciones Morfológicas

En los siguientes pasos, se realizan operaciones morfológicas de erosión y dilatación. Estas operaciones tienen el objetivo de eliminar el ruido y consolidar las formas en la imagen. La erosión reduce el tamaño de las regiones blancas, mientras que la dilatación las agranda, ayudando a definir mejor las estructuras relevantes y eliminando pequeños defectos de la captura. El resultado se aprecia en la Figura 3.6.



Figura 3.6: Imagen filtrada morfológicamente

3.3.2. Obtención de líneas guías

Detección de Regiones

Una vez obtenida la imagen deseada en escala de grises, la segunda parte del procesamiento se centra en la detección de regiones. En primer lugar, se realiza un etiquetado de las regiones conectadas, obteniendo información detallada de cada región.

Demarcación de Región de Interés

Para enfocar el análisis en una parte específica de la imagen, se incorpora una línea horizontal en la imagen original. Esta línea actúa como un delimitador para definir una región de interés (ROI). Posteriormente, las regiones etiquetadas se filtran basándose en su posición relativa a esta línea, conservando únicamente aquellos elementos situados por debajo de la misma y que cumplan ciertas condiciones.

Filtrado de Regiones

Las regiones etiquetadas se ordenan por su extensión en el eje Y, buscando identificar la longitud de las líneas. Se aplican criterios específicos para filtrar las regiones:

- Porcentaje de píxeles por debajo del umbral marcado por la línea horizontal.
- Área mínima de la región detectada.

Esqueletización de la Región Más Larga

Finalmente, se aplica la operación de esqueletización a la región más larga válida encontrada. La esqueletización es un proceso que reduce las regiones a su estructura fundamental. Se utiliza el algoritmo de Zhang-Suen, el cual es un método iterativo para obtener el esqueleto de una figura binaria. Este algoritmo funciona mediante la eliminación de píxeles de borde mientras preserva la conectividad topológica del objeto, hasta que se alcanza una estructura delgada del tamaño de un píxel que representa el esqueleto del objeto original.

A modo de resumen, el algoritmo de Zhang-Suen consta de los siguientes pasos:

1. Identificación de Puntos de Borde: En cada iteración, se identifican los píxeles de borde que pueden ser eliminados.
2. Condiciones de Eliminación: Se evalúan varias condiciones para asegurar que la eliminación de un píxel no rompe la conectividad del objeto. Estas condiciones verifican:
 - La cantidad de vecinos no nulos.
 - La cantidad de transiciones de blanco a negro en la vecindad del píxel.
 - Que al eliminar el píxel, la estructura conectada del objeto se mantenga.
3. Eliminación Iterativa: Se aplican las condiciones iterativamente hasta que no se puedan eliminar más píxeles sin romper la conectividad del objeto.

El resultado es una versión esqueletizada de la región más larga, preservando su estructura esencial mientras se eliminan los píxeles redundantes.

Como resultado final, se presenta la Figura 3.7, en la cual se observan las regiones detectadas en color rojo y la línea esqueletizada resultante en color amarillo.

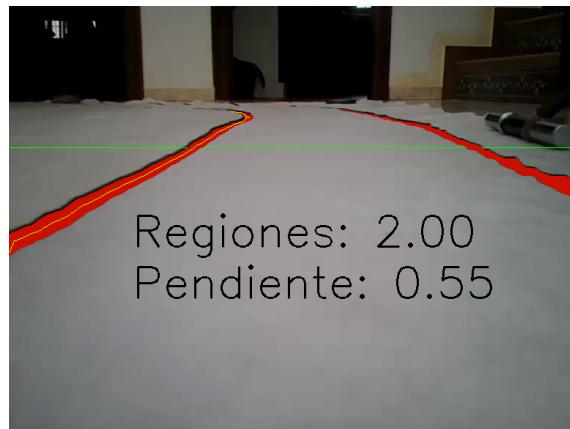


Figura 3.7: Imagen resultado final

3.3.3. Obtención de la pendiente

A partir de la lista generada con la obtención de líneas de carril anterior, se itera sobre sus elementos buscando aquellos no nulos a partir de los cuales obtener las coordenadas en píxeles de los puntos de las líneas.

A continuación, se recurre a la función de '**polyfit**' para resolver el problema de polinomio de mejor ajuste, obteniéndose los **coeficientes de la aproximación de primer grado** (línea recta), a partir del cuál puede obtenerse la **pendiente** derivando en cualquiera de sus puntos, en base a la cuál tomar la decisión de control.

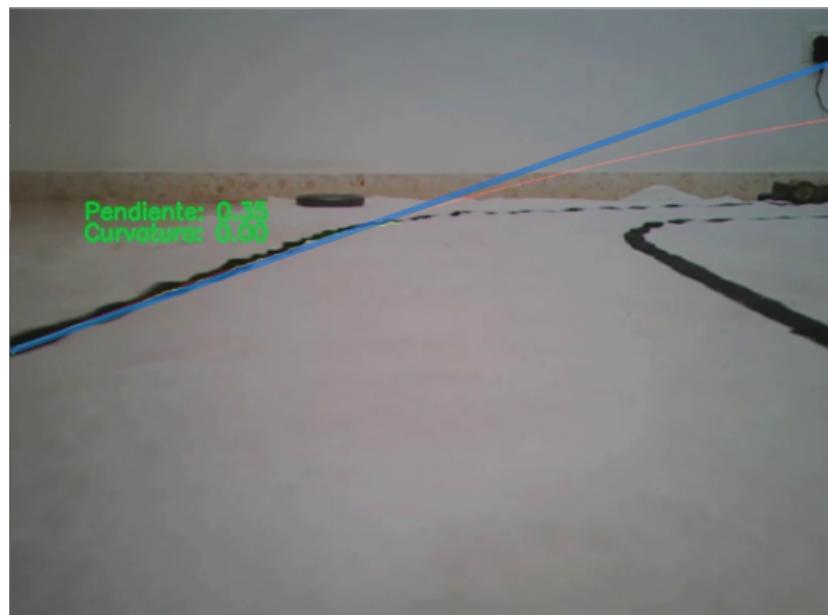


Figura 3.8: Representación de polinomio calculado sobre imagen

Además, en primeras versiones, se analizó la posibilidad de estimar la curvatura con un polinomio de 2do o 3er orden, realizando su segunda derivada. Sin embargo, en pruebas experimentales esta métrica quedó descartada por falta de robustez y dado que la actuación final sería todo/nada.

3.3.4. Lógica de control

En la lógica de control, podemos distinguir el módulo de control de la dirección todo/nada, así como el control de la potencia de tracción del vehículo.

La decisión de actuación sobre la dirección se basará en la pendiente anteriormente calculada, distinguiendo:

- **Número de Regiones detectadas:**

- Si el número de regiones detectadas es 0, se mantiene la decisión previa de control (se asume que no hay nuevas líneas detectadas y el sistema sigue con la última decisión tomada).
- Si el número de regiones detectadas es 2, se establece una **dirección neutra**, lo que significa que el vehículo sigue una trayectoria recta porque se han detectado dos líneas paralelas.

- **Pendiente:**

- Si la pendiente es menor que 0 (pendiente negativa), se toma la decisión de un **giro de dirección a la izquierda**.
- Si la pendiente es mayor que 0 (pendiente positiva), se toma la decisión de un **giro de dirección a la derecha**.

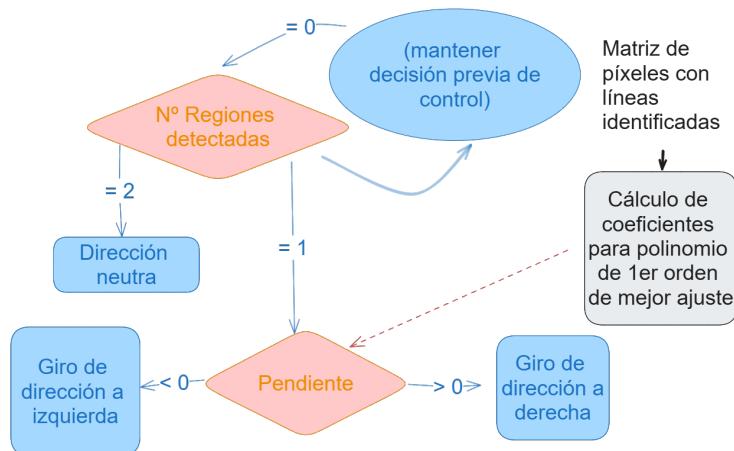


Figura 3.9: Diagrama de flujo del módulo de control de dirección

Por otro lado, la potencia de tracción a aplicar se decide en base al estado de la dirección, ya que la fuerza de resistencia que se presenta al movimiento del vehículo cambia junto a ello.

Por ello, en búsqueda de controlar a la velocidad más lenta posible (dado que de otra forma las vibraciones en la cámara harían imposible una correcta detección), se determina experimentalmente que es necesario aplicar un 10 % de ancho de pulso PWM a los motores para moverse en línea recta, y un 20 % para tomar las curvas.



Figura 3.10: Diagrama de flujo del módulo de control de tracción

Capítulo 4

Pruebas de funcionamiento y Resultados

Para desarrollar las pruebas de funcionamiento del vehículo autónomo, se ha construido una pista de pruebas sobre una tela blanca, donde se han trazado líneas con pintura negra. Las dimensiones de la pista son de 4 metros por 2.5 metros, como se puede observar en la figura 4.1. El ancho del carril se ha ajustado a aproximadamente 30 cm a lo largo de todo el recorrido.

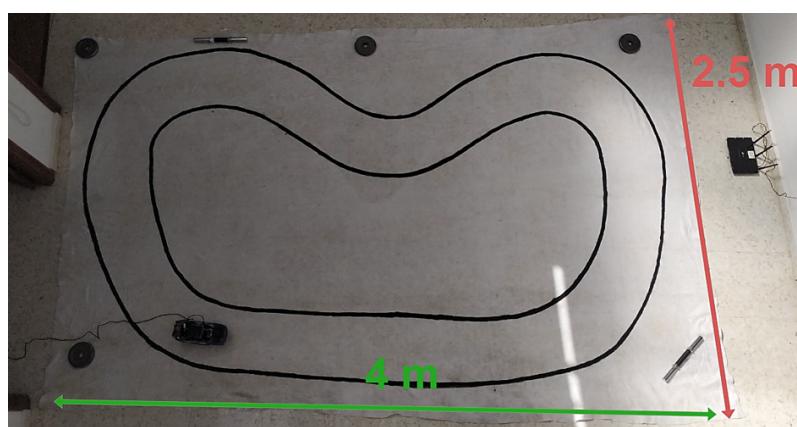


Figura 4.1: Pista de pruebas

Los resultados obtenidos se muestran en el siguiente vídeo, el cual documenta una serie de vueltas autónomas realizadas por el vehículo. En él se presenta la trayectoria completa del circuito desde una vista panorámica, así como una vista en primera persona a través de la cámara implementada en el coche, junto con el procesamiento de imágenes en tiempo real. El vídeo puede consultarse en el Capítulo 6.

Como se puede observar, los resultados de los experimentos no son perfectos y se identifican ciertas irregularidades. Los principales fallos observados son los siguientes:

- **Déficit de detección:** Se producen fallos en la parte de visión debido a que por ejemplo en un frame no se ve una de las líneas del circuito lo que provoca que falle el sistema, ver Figura 4.2 ilustrativa.

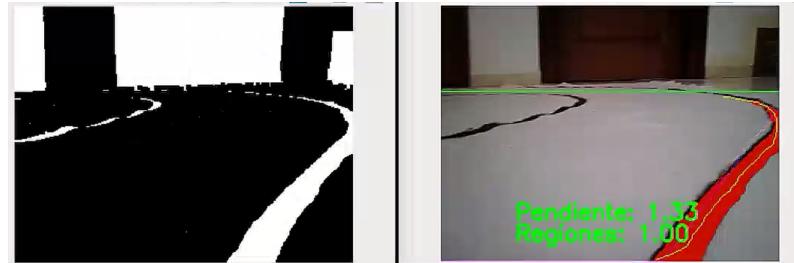


Figura 4.2: Déficit de detección

- **Detección de falso positivo:** Se detectaron objetos fuera del circuito, los cuales afectaron los cálculos asociados a las líneas detectadas, provocando desajustes en el control y el funcionamiento autónomo del coche. Ver Figura 4.3 ilustrativa.



Figura 4.3: Detección de falso positivo

- **Desajuste de control:** Se produjeron fallos puntuales en el sistema de control, particularmente en la estimación de elementos como la pendiente de la trayectoria de la línea, lo que afectó el desarrollo de la conducción autónoma. Ver Figura 4.4 ilustrativa.

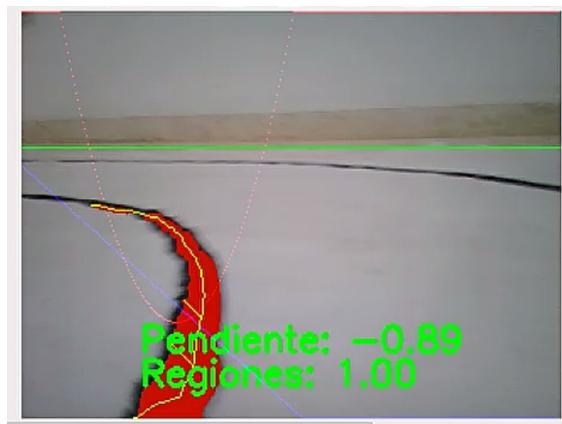


Figura 4.4: Desajuste de control

A pesar de las irregularidades identificadas durante las pruebas, los resultados generales indican que el sistema de conducción autónoma funciona de manera efectiva. Los fallos observados, aunque significativos, no impiden que el vehículo realice vueltas autónomas completas en la pista de pruebas.

Capítulo 5

Conclusiones y Trabajo Futuro

A lo largo de las pruebas de funcionamiento y análisis de resultados, se han identificado una serie de limitaciones en nuestro sistema que afectan a su desempeño. Estas limitaciones pueden ser abordadas como mejoras futuras para optimizar el rendimiento del vehículo autónomo.

5.1. Limitaciones actuales

Las principales limitaciones detectadas en nuestro proyecto son las siguientes:

1. **Mecanismo de Giro:** El actual mecanismo de giro del vehículo utiliza un motor de tipo "todo o nada", el cual opera con una serie de reductoras que permiten un giro completo hacia la izquierda o la derecha. El retorno de las ruedas a la posición recta se realiza mediante muelles auxiliares. Este sistema limita la precisión y suavidad de los giros, afectando la maniobrabilidad del coche.
2. **Detección:** El sistema de detección de líneas de carril se basa en un procesamiento de imagen sencillo utilizando técnicas morfológicas. Estos parámetros se ajustan manualmente según las condiciones específicas, lo que puede generar errores en la detección de las líneas del carril. La falta de adaptabilidad del sistema a diferentes condiciones ambientales y de iluminación es una limitación significativa.
3. **Posicionamiento:** Actualmente, nuestro coche autónomo carece de un sistema de autolocalización. Esto supone que el sistema de control no cuenta con información precisa sobre la posición del vehículo en el entorno. La ausencia de datos de posicionamiento impide implementar algoritmos de control más avanzados, afectando la precisión y estabilidad del coche durante su operación.
4. **Procesamiento:** El procesamiento de imágenes y el control del vehículo se realizan mediante un microprocesador Raspberry Pi que presenta limitaciones en términos de rendimiento y capacidad de procesamiento. Estas restricciones impiden que las imágenes se procesen a mayor resolución y a frecuencias más altas.

5.2. Ampliaciones a futuro

En base a las limitaciones identificadas, se proponen las siguientes mejoras para optimizar el rendimiento de nuestro coche autónomo:

1. **Mecanismo de Giro:** Se sugiere reemplazar el actual mecanismo de giro "todo o nada" por un sistema que utilice servomotores. Este cambio permitirá realizar giros graduales y sostenidos en ambos sentidos, mejorando significativamente la maniobrabilidad y control del vehículo.
2. **Detección:** El sistema de detección basado en operaciones morfológicas simples debe ser sustituido por una solución más avanzada. Proponemos el entrenamiento de una red neuronal convolucional (CNN) utilizando imágenes capturadas por las cámaras del vehículo. Este enfoque permitirá una detección más precisa y robusta de las líneas del carril, gracias a la capacidad de la red neuronal para aprender características relevantes a través de entrenamientos iterativos y adaptarse a diversas condiciones ambientales y de iluminación.
3. **Posicionamiento:** Para mejorar el control del vehículo, es crucial implementar un sistema de posicionamiento. Una opción es incorporar encoders en los motores de tracción, ya sea sustituyendo los motores originales por otros que incluyan encoders integrados o añadiendo sensores de efecto Hall a los motores existentes. Otra alternativa es implementar un módulo de posicionamiento GPS, especialmente diseñado para entornos interiores. La fusión de datos de estos sensores proporcionará información precisa sobre la posición del vehículo, mejorando los algoritmos de control y la navegación autónoma.
4. **Procesamiento:** Sería recomendable emplear un microprocesador de mayor capacidad. Este cambio permitirá manejar mayores volúmenes de datos y procesar imágenes a mayor resolución y frecuencia. Además, el código debe ser optimizado para aprovechar al máximo las capacidades del nuevo hardware tras las mejoras, asegurando un procesamiento más eficiente y rápido.

Capítulo 6

Archivos adjuntos

Junto a esta memoria de proyecto, se hace entrega de:

- **'software.zip'**: Comprimido zip con el código fuente del software del proyecto.
En ella, se encontrarán los siguientes directorios:
 - *'esp8266_low_level_car_receiver'* : Códigos y librerías desarrollados para el microcontrolador de bajo nivel del vehículo, , siguiendo jerarquía de compilación de la herramienta Platform IO. (C)
 - *'esp8266_low_level_controller_transmitter'* : Códigos y librerías desarrollados para el microcontrolador de bajo nivel del mando controlador, siguiendo jerarquía de compilación de la herramienta Platform IO. (C)
 - *'raspberry_pi_high_level'* : Programas desarrollados para microprocesador alto nivel del vehículo.(Python)

A distinguir:

- *'controlSW.py'*: Programa principal de alto nivel.
- *'userConfig.py'*: Asignación de valores a los parámetros del programa.
- *'helpers.py'*: Implementación de las funciones auxiliares del programa.
- **'videoDemo.mp4'**: Vídeo demostrativo de pruebas y resultados.
- **'presentacion.pdf'**: Presentación a modo resumen de los puntos principales abordados en esta memoria.