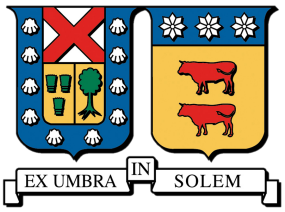
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTRÓNICA

VALPARAÍSO - CHILE



"INTEGRACIÓN DE SISTEMA DE VISIÓN POR COMPUTADOR A PLATAFORMA ROBÓTICA MÓVIL CON TRACCIÓN DIFERENCIAL"

POLET ALMENDRA CASTILLO DEL PINO

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO MENCIÓN CONTROL AUTOMÁTICO

PROFESOR GUÍA:	DR. FRANCISCO VARGAS PARRA
PROFESOR CORREFERENTE:	DR. GONZALO CARVAJAL BARRERA
PROFESOR CORREFERENTE:	DR.JUAN CARLOS AGÜERO VÁSQUEZ

DICIEMBRE 2022

Agradecimientos

Este trabajo de título no podría haberse realizado sin el constante apoyo del mi profesor guía Dr. Francisco Vargas, el cual gracias a su *feedback* permitió que la plataforma desarrollada evolucionara sin perder su foco.

También destacar al profesor Dr. Gonzalo Carvajal con sus importantes aportes y acotaciones sobre el área de Computadores, en conjunto a consejos para llevar a cabo la implementación física de la plataforma.

Además, agradecer al profesor Andrés Peters por su invaluable ayuda para diseñar en 3D, por las gestiones realizadas referente a la PCB desarrollada y por la facilitación de todos aquellos componentes extra que necesité para desarrollar la plataforma.

Gracias por el apoyo desde la Universidad Técnica Federico Santa María y al Dr. Juan Carlos Agüero, que a través del Advanced Center for Electrical and Electronic Engineering (AC3E) - Centro Basal ANID FB0008, y al proyecto interno PI-LII-2020-38 "Diseño de técnicas de control para un sistema piloto de vehículos a escala autónomos considerando restricciones de comunicación entre vehículos", me brindaron de motivación para concluir el trabajo de título.

Por último pero no menos importante, quiero agradecer a mi pareja Diego Pandolfa por su apoyo incondicional; y a mi familia que constantemente me motivó bajo el argumento de que "*hasta los caballos terminan las carreras*".

Sin su ayuda, este trabajo no hubiera sido posible. A todos, muchas gracias.





Integración de sistema de visión por computador a plataforma robótica móvil con tracción diferencial

Polet Castillo

Memoria para optar al título de Ingeniero Civil Electrónico,
Mención en Control Automático

Profesor Guía: Francisco Vargas

Diciembre 2022

Resumen

A partir del trabajo de título que lleva por nombre *Diseño, Implementación y Evaluación de una Plataforma Experimental para el Estudio del Control de Flotas de Robots Móviles* [9] desarrollado en la Universidad Técnica Federico Santa María, se llevó a cabo la adecuación de un entorno propicio para la integración de visión por computador en dicha plataforma. Para ello, se reemplazó la unidad de control central por una Raspberry Pi 4 Modelo B [15], lo que conllevó a una serie de modificaciones, como la incorporación de nuevos componentes, alteraciones en la estructura de la plataforma, requisitos energéticos distintos y la utilización de Python como lenguaje de programación, entre otros.

En este trabajo de titulación primero se llevó a cabo un estudio de la plataforma base con el objetivo de integrar la nueva unidad de control central. Luego, se realizaron mejoras destinadas a aprovechar el potencial que ofrece la Raspberry Pi, tales como el procesamiento *multi-threading* y la incorporación de una cámara digital compatible. Con la nueva unidad se replicaron las funciones de la plataforma base, pero además se habilitó la posibilidad de realizar seguimiento de trayectorias en base a visión por computador. En este trabajo también se estudió la capacidad de la nueva plataforma de detectar, usando la cámara, las líneas que definan la ruta o delimiten el carril.

Palabras clave: Plataforma robótica móvil, Robot, Raspberry Pi, Python, Multi-threading, Computer vision.





Integration of computer vision system to mobile robotic platform with differential traction.

Polet Castillo

Final Project Report towards the fulfillment of the Ingeniero Civil Electrónico, Minor in Automatic Control degree.

Advisor: Francisco Vargas

December 2022

Abstract

Based on the final year project titled *Design, Implementation, and Evaluation of an Experimental Platform for the Study of Mobile Robot Fleet Control* [9] carried out at the Federico Santa Maria Technical University, an environment suitable for the integration of computer vision into the platform was developed. To achieve this, the central control unit was replaced with a Raspberry Pi 4 Model B [15], which led to a series of modifications, such as the addition of new components, changes in the platform's structure, different energy requirements, and the use of Python as the programming language, among other changes.

In this work, firstly a study of the base platform was carried out with the aim of integrating the new central control unit. Then, improvements were made to take advantage of the potential offered by the Raspberry Pi, such as *multi-threading processing* and the incorporation of a compatible digital camera. With the new unit, the functions of the base platform were replicated, but the possibility of performing trajectory tracking based on computer vision was enabled. Additionally, this work studied the ability of the new platform to detect, using the camera, the lines that define the route or delimit the lane.

Keywords: Mobile robotic platform, Robot, Raspberry Pi, Python, Multi-threading, Computer vision.

Glosario

- **AI:** *Inteligencia artificial* (Artificial Intelligence)
- **CV:** *Visión por computador* (Computer vision)
- **GPIO:** *Entrada/Salida de propósito general* (General Purpose Input/Output)
- **I2C:** *Circuito inter-integrado* (Inter-Integrated Circuit)
- **LIDAR:** *Sistema de medición y detección de objetos mediante láser* (Laser Imaging Detection and Ranging)
- **Motor DC:** *Motor de Corriente Continua* (Direct Current Motor)
- **MQTT:** *Transporte de telemetría en cola de mensajes* (Message Queing Telemetry Transport)
- **OEM:** *Onda Electromagnética*
- **OSS:** *Código abierto* (Open source software)
- **PCB:** *Circuito impreso* (Printed Circuit Board)
- **SLAM:** *Localización y modelado simultáneo* (Simultaneous localization and mapping)
- **UDP:** *Protocolo de datagramas de usuario* (User datagram protocol)

Índice de Contenidos

Agradecimientos I
Resumen II
Abstract III
Glosario IV
Índice de Contenidos VI
Índice de Figuras VII
Índice de Tablas X

1 Introducción **1**
1.1. Motivación 1
1.2. Objetivo 4
1.3. Alcance 4

2 Estudio de la plataforma base **6**
2.1. Componentes 6
2.1.1. Unidad de control central. 6
2.1.2. Actuadores. 6
2.1.3. Sensores. 8
2.1.4. Alimentación. 8
2.1.5. Estructura. 9
2.2. Diagrama de bloques. 9
2.3. Modelo matemático 11
2.3.1. Controlador 12
2.4. Funcionalidades 13

3 Modificaciones a la plataforma base **15**
3.1. Modificaciones de la plataforma base 15
3.1.1. Cambio en el modelo matemático 15
3.1.2. Cambio en los sensores 15
3.2. Adición de sensores a la plataforma base 16
3.2.1. Conversor análogo digital. 16
3.2.2. Unidad de medición inercial. 16
3.2.3. Cámara digital. 17
3.3. Variación en el requerimiento energético de la plataforma 17

3.4. Cambios en la estructura de la plataforma 19
3.5. Adición de una nueva PCB a la plataforma 20

4 Descripción e implementación de la plataforma actualizada 22
4.1. Estructura 22
4.2. Componentes 23
4.2.1. Adquisición de datos 23
4.3. Diagrama de bloques 29
4.4. Controlador 31
4.4.1. Esquema de control para móvil sucesor. 31
4.4.2. Esquema de control para móvil líder. 32
4.4.3. Modificación del esquema de control mediante comunicación entre la flota. 32
4.4.4. Implementación del controlador en tiempo discreto 33
4.4.5. Parámetros de los controladores 35
4.5. Comunicación 36
4.5.1. Control de parámetros 36
4.5.2. Reporte a un agente monitor y entre la flota 37

5 Resultados 39
5.1. Funcionamiento basal 39
5.1.1. Comunicación - Reporte 39
5.1.2. Comunicación - Control 42
5.1.3. Seguimiento de velocidad y trayectoria 43
5.1.4. Seguimiento de distancia 46
5.2. Implementación de Computer Vision 52
5.2.1. Limitantes observadas para el funcionamiento del segmentador . 54
5.2.2. Evaluación de la factibilidad de segmentación en tiempo real . . 56

6 Conclusiones 58

7 Referencias 60

8 Anexos 62
8.1. Repositorio 62
8.2. Aspectos de interés para la implementación de sistemas multiagentes en plataformas experimentales. 62
8.2.1. Control de distanciamiento 63
8.2.2. Control de seguimiento 63
8.2.3. Comunicación entre la flota 63
8.2.4. Actuadores, sensores y capacidad de cómputo 64
8.3. Fotografías del proyecto 65
8.4. Códigos 67

Índice de Figuras

1.1. Diagrama de una caravana de camiones en formación de pelotón. 2
1.2. Algunas plataformas multipropósito con enfoque educacional. 3

2.1. Plataforma RUPU [9]. 7
2.2. Tarjeta de desarrollo D1R32 Wemos. 7
2.3. Componentes de actuación del robot diferencial. 7
2.4. Sensores del móvil diferencial. 8
2.5. Estructura del móvil. 9
2.6. Diagrama de bloques que incluye señales de alimentación, de la plataforma. 10
2.7. Diagrama de bloques que incluye señales de medición y control, de la plataforma. 10
2.8. Diagrama de velocidad lineal, velocidades angulares y ángulo de la trayectoria en un móvil. 13
2.9. Diagrama de medición de distancia entre móviles y comparación de la trayectoria definida (línea punteada blanca) con la trayectoria del móvil, $\theta(t)$. 13
2.10. Diagrama de bloques del controlador planteado para el sistema. 14

3.1. Raspberry Pi 4 modelo B. 16
3.2. Componentes extras para la plataforma. 16
3.3. Alternativas de alimentación para la Raspberry Pi. 18
3.4. Estructura del móvil con tracción diferencial. 20
3.5. Estructura de soporte para adición de cámara, permitiendo regular su altura y grado de inclinación 20
3.6. Gráfico 3D de la PCB adicional. 21
3.7. Diagrama de conexionado de la nueva PCB. 21

4.1. Fotografía de la nueva versión del móvil con tracción diferencial. 23
4.2. Fotografías de la distribución de la estructura de la nueva plataforma. . . 23
4.3. Vista inferior del móvil diferencial para visualizar los puerto de conexión al multiplexor de cada sensor de los Arrays IR. 25
4.4. Ejemplo de medición de cantidad de desviación del móvil. 28
4.5. Diagrama de bloques que incluye señales de medición y control de la actualización de la plataforma. 29

4.6. Diagrama de bloques que incluye señales de alimentación de la actualización de la plataforma, opción 1. 30
4.7. Diagrama de bloques que incluye señales de alimentación de la actualización de la plataforma, opción 2. 30
4.8. Alternativas de esquemas de control para los distintos móviles del pelotón. 31
4.9. Captura del Dashboard de control desarrollado dentro de *Ubidots*. 37

5.1. Gráfico del esquema de comunicación utilizado para el experimento de comunicación con tres agentes y un monitor. 40
5.2. Gráfico del esquema de comunicación utilizado para el experimento de compatibilidad de comunicación entre plataformas. 41
5.3. Gráfica del recorrido utilizado para el experimento de seguimiento de la trayectoria. 43
5.4. Gráfica del seguimiento de velocidad del móvil con tracción diferencial. . . 44
5.5. Gráfica del seguimiento de trayectoria del móvil con tracción diferencial. . 45
5.6. Gráfica del seguimiento de distancia al móvil Líder. 46
5.7. Gráfica del seguimiento de distancia al móvil Líder, con acercamiento. . . 47
5.8. Gráfica de la velocidad de referencia del móvil seguidor. 48
5.9. Gráfica del seguimiento de velocidad del móvil seguidor. 49
5.10. Gráfica de la activación de control de distancia del móvil seguidor. 50
5.11. Gráfica del seguimiento de distancia de referencia del móvil seguidor. . . . 50
5.12. Gráfica del seguimiento de velocidad de referencia del móvil seguidor en conjunto a la activación del controlador de distancia. 51
5.13. Prueba de captura de video desde distintas posiciones. Captura del horizonte de la trayectoria. 52
5.14. Prueba de captura de video desde distintas posiciones. Captura de la trayectoria en curva. 52
5.15. Captura de un video proveniente de la cámara (izquierda) en comparación con el video segmentado utilizando *OpenCV* (derecha). 53
5.16. Captura de un video proveniente de la cámara (izquierda) en comparación con el video segmentado utilizando *OpenCV* (derecha), en situación de poco óptima iluminación. 55
5.17. Captura de un video proveniente de la cámara (izquierda) en comparación con el video segmentado utilizando *OpenCV* (derecha), en situación de confusión por falta de murallas. 55
5.18. Captura de un video proveniente de la cámara (izquierda) en comparación con el video segmentado utilizando *OpenCV* (derecha), en situación de ausencia de captura de trayectoria en el recorrido. 56
5.19. Captura de un video proveniente de la cámara (izquierda) en comparación con el video segmentado utilizando *OpenCV* (derecha), en situación de objetos ajenos al recorrido. 56
5.20. Captura de la demostración desarrollada por Xiaotian para seguimiento de un objeto de color específico [25]. 57

8.1. Fotografía del móvil con tracción diferencial, vista lateral derecha. 65
8.2. Fotografía del móvil con tracción diferencial, vista ortogonal. 65
8.3. Fotografía del móvil con tracción diferencial, vista lateral izquierda. 66

8.4. Fotografía del móvil con tracción diferencial, vista posterior. 66

Índice de Tablas

2.1. Parámetros del sistema. 12
2.2. Parámetros del controlador del sistema. 14

3.1. Descripción de requerimiento de corriente de los componentes de la plataforma. 18

4.1. Resumen de los componentes que utiliza la actualización de la plataforma. 24
4.2. Direcciones utilizadas para leer puertos de entrada del Multiplexor. 26
4.3. Variables que recibe de entrada cada función que implementa los controlador. 36

5.1. Agentes que forman parte del experimento de comunicación y reporte. . . 39
5.2. Parámetros del controlador del sistema utilizados en los experimentos. . . 43

Capítulo 1

Introducción

En este capítulo se expone la motivación detrás de este trabajo de título, que se enfoca en la búsqueda de plataformas asequibles a baja escala que permitan estudiar sistemas multiagente, en particular el *platooning*. Además, se describen las áreas de interés y se detallan los objetivos y alcances del presente trabajo.

1.1. Motivación

Cada vez más, los vehículos están incorporando sistemas de control automático para ayudar en la conducción, con el objetivo de reducir accidentes y mejorar la calidad de vida de las personas. Algunos vehículos de lujo, como los de Tesla y Mercedes-Benz, ya han alcanzado el nivel 3 de los 6 niveles de autonomía definidos por la SAE[6], lo que significa que son capaces de tomar decisiones de conducción, pero aún requieren la presencia del conductor en caso de emergencia. En el sector logístico, el transporte de mercancías por tierra también presenta desafíos en cuanto a la eficiencia y seguridad de los viajes, tanto para el conductor como para la carga transportada. Un escenario de interés es el de varios camiones en caravana, ya que esto permite un ahorro de combustible y, por lo tanto, una menor emisión de contaminantes por kilómetro recorrido[11] [26]. Esto se debe a que los vehículos en caravana generan una fuerza de arrastre entre ellos. Sin embargo, este escenario presenta el desafío de coordinar la conducción de una flota de camiones para maximizar la eficiencia y la seguridad del viaje.

Este escenario en particular es una aplicación de un sistema multiagente, el cual se compone de entidades *inteligentes* que interactúan para lograr un objetivo a nivel del sistema global. El término "inteligente" se utiliza para describir estas entidades ya que tienen la capacidad de adaptarse a diversas condiciones conocidas, como su interacción con entidades externas al sistema.

El *platooning* o *pelotón* es un caso específico de sistema de navegación multiagente, en el que se busca que una flota de vehículos, principalmente camiones, viaje simultáneamente, uno detrás del otro, como se muestra en la Figura 1.1. Estudios realizados en Estados Unidos han demostrado que esta tecnología mejora la seguridad vial y la



Figura 1.1: Diagrama de una caravana de camiones en formación de pelotón.

eficiencia de las autopistas sin necesidad de hacer cambios drásticos en las carreteras existentes [16].

Dada la importancia de la industria logística en todo el mundo, es interesante estudiar los desafíos de los vehículos autónomos desde la perspectiva del control automático. Sin embargo, una limitación es que aunque existen vehículos a escala real que ya implementan sistemas de control automático para la conducción asistida, los datos no están disponibles para su estudio y los costos para adquirir vehículos, infraestructura y poner en marcha escenarios a escala real son muy elevados, requiriendo una inversión de decenas de miles de dólares y varios años de desarrollo.

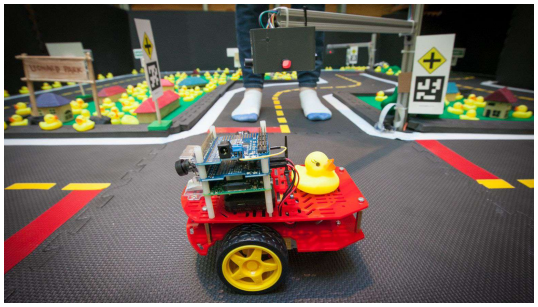
Por lo tanto, es necesario contar con una plataforma a pequeña escala y bajo costo que permita emular diferentes escenarios y adquirir datos con fines académicos. En la Sección 8.2, se detallan los aspectos de interés que definen las características mínimas que debe tener esta plataforma.

Aunque existen varias plataformas comerciales multipropósito y algunas con enfoque educacional en el mercado, ninguna de ellas cumple con las características requeridas para estudiar el *platooning*.

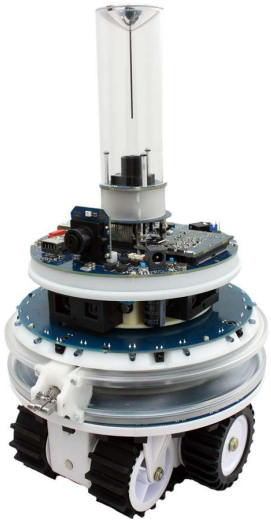
La plataforma comercial multipropósito DuckieTown [7, 17] (Figura 1.2a), de código abierto y desarrollada con fines educativos, ofrece comunicación entre móviles, planificación de rutas, reconocimiento de imágenes y objetos en la ruta mediante algoritmos de IA, entre otras prestaciones. Sin embargo, su precio de alrededor de 350 dólares por móvil es comparativamente alto frente a otras plataformas experimentales de menos de 100 dólares por móvil, aunque estas últimas tengan menos sensores y estén limitadas para realizar experimentos de platooning en situaciones específicas.

Otro ejemplo es el proyecto MarXbot [3] (Figura 1.2b), que desarrolla robots con tracción diferencial modular todo-terreno capaces de experimentar en diversos escenarios y formaciones multiagentes mediante distintos tipos de sensores, como giroscopios, múltiples cámaras y de tipo LIDAR. Sin embargo, el precio de esta plataforma es superior a los 250 dólares por móvil debido a las prestaciones que ofrece.

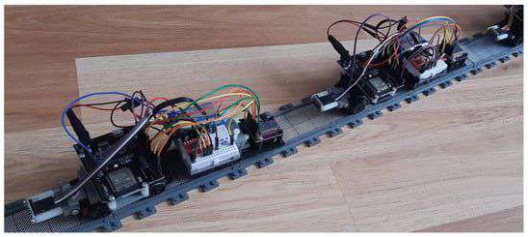
En contraste, PL-TOON [18] (Figura 1.2c) es una plataforma educativa de bajo costo (menos de 100 dólares por móvil) que se utiliza para estudiar el control cooperativo descentralizado mediante trenes programables que se desplazan a través de un riel LEGO®. Aunque esta plataforma es útil para estudiar problemas longitudinales de control, es decir, que cada agente mantenga una distancia deseada con su predecesor, presenta limitantes en el desplazamiento de los móviles debido a que su trayectoria está fijada



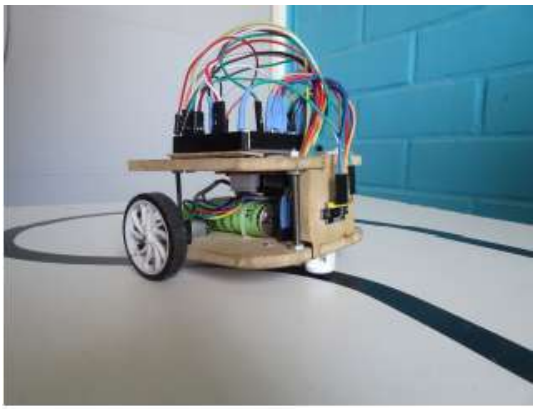
(a) DuckieTown [7][17]



(b) MarXbot [3]



(c) PL-TOON [18]



(d) RUPU [9]

Figura 1.2: Algunas plataformas multipropósito con enfoque educacional.

por los rieles, lo que la inhabilita para el estudio de otras estructuras de formación.

Para remediar esta limitante, se desarrolló la plataforma RUPU [9] (Figura 1.2d), que fue diseñada para el estudio del control de flotas de robots móviles. RUPU incorpora tracción diferencial bajo 2 ruedas independientes, sensores que permiten detectar una trayectoria definida por una línea en el suelo y un sensor de distancia frontal, lo que le permite abandonar la necesidad de un riel y ofrecer a los móviles un mayor grado de libertad en su movimiento.

No obstante, RUPU tiene limitantes las cuales fueron detectadas al momento de estudiar esta plataforma, las cuales se describen a continuación.

- El recorrido de la flota se encuentra acotado al diseño de la pista por medio de una línea en el suelo.
- La plataforma utiliza sensores de distancia frontal lineales de acotado ángulo de visión, por lo que en seguimiento en curvas cerradas el móvil sucesor pierde de vista

al móvil predecesor. Esto provoca, a nivel de controlador, que el error asociado al seguimiento aumente lo que genera como salida un aumento de velocidad, que en reiteradas veces culmina en una colisión entre ambos móviles.

En resumen, aunque las plataformas descritas logran cubrir aspectos de interés para el estudio de sistemas multiagentes a baja escala, sería de utilidad ampliar sus capacidades para abarcar una mayor cantidad de aspectos, como la planificación de rutas (*path-planning*).

1.2. Objetivo

El presente trabajo de título utiliza como base la plataforma RUPU [9], con el fin de expandir sus capacidades. Así, el objetivo de este trabajo de título es la creación de una nueva versión de la plataforma RUPU, otorgándole mayor capacidad de cómputo por medio de la sustitución del procesador utilizado.

Como objetivos específicos se tienen:

1. Replicar las funcionalidades de la versión original de la plataforma.
   - El móvil es capaz de seguir una línea en el piso, aplicando técnicas de control.
   - El móvil es capaz de mantener una distancia fija con respecto a un objeto frente a él, aplicando técnicas de control.

2. Incorporar nuevas funcionalidades, implementación de procesamiento de video:
   - Añadir soporte para cámara.
   - Detección de líneas en la pista.

1.3. Alcance

A continuación se describe el alcance que posee el presente trabajo de título:

1. Reemplazar la unidad de control central de la plataforma RUPU a una Raspberry Pi 4 modelo B [15] la cual cuenta con un procesador ARM Cortex-A72 de 1.5 GHz 64-bit quad core y 4GB de RAM.

   Esta posee las prestaciones necesarias para la interconexión de los componentes de la plataforma original permitiendo replicar las funcionalidades de la plataforma original, además de aportar en velocidad de cómputo y capacidad de procesamiento manteniéndose bajo los 150 dolares por tarjeta.

2. Adicionar la cámara RPI V2 [14] de 8MP a la plataforma.

   Esta, en conjunto con la nueva unidad de control central, permite incluir procesamiento de video a la plataforma.

3. Replicar el funcionamiento de la plataforma con la nueva unidad de control central, y extenderla para detectar líneas de trayectoria predefinida utilizando una cámara específica.

Cabe destacar que dentro de este trabajo de título se desarrolla un algoritmo básico para detectar líneas utilizando una cámara, pero queda afuera del alcance de esta memoria utilizar este resultado para realizar control del móvil.

Capítulo 2

Estudio de la plataforma base

El presente trabajo comienza con el estudio de la plataforma RUPU, la cual fue desarrollada durante los años 2020 y 2021 por Carlos Escobar en su trabajo de título Diseño, Implementación y Evaluación de una Plataforma Experimental para el Estudio del Control de Flotas de Robots Móviles [9].

Esta plataforma robótica de tracción diferencial fue creada con el objetivo de estudiar sistemas multiagente, en particular el *platooning*. Los agentes son capaces de seguir de manera autónoma una trayectoria definida por una línea en el suelo, mantener una distancia constante con respecto al móvil predecesor y corregir su velocidad para cumplir con el objetivo propuesto. En la Figura 2.1 se muestra una imagen ilustrativa del funcionamiento del proyecto base, donde se observa el seguimiento de línea y el comportamiento multiagente.

2.1. Componentes

2.1.1. Unidad de control central.

La plataforma cuenta con una tarjeta de desarrollo D1 R32 [23] como unidad de control central, presente en la Figura 2.2. Esta tarjeta está basada en el microcontrolador ESP32 WROOM 32 SMD, que incorpora WiFi y Bluetooth (BLE). La CPU es un Dual core Tensilica LX6 (32 bit) con frecuencia de 240 MHz, memoria flash de 4 MB, voltaje de operación de 7 a 16 $[V]$ DC y consumo de corriente máximo de 250 $[mA]$. Además, posee interfaces de comunicación para los protocolos de UART, SPI, I2C, entre otros.

2.1.2. Actuadores.

La plataforma cuenta con dos motores DC modelo N20 MicroGear [20] con voltaje de 6 a 12 $[V]$, 80 $[RPM]$ con carga, torque de 2 $[kg/cm]$, corriente base de 0.07 $[A]$ y 1.6 $[A]$ cuando se encuentra bloqueado a un voltaje de 12 $[V]$. Además, estos motores poseen una caja reductora de 1:100 y un *enconder* de cuadratura que se caracterizará más adelante.

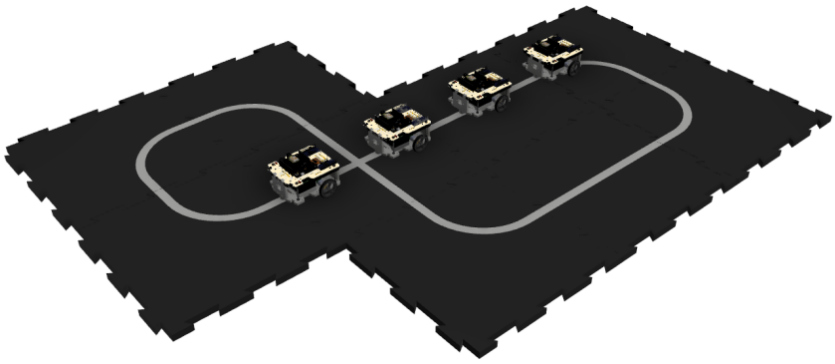


Figura 2.1: Plataforma RUPU [9].

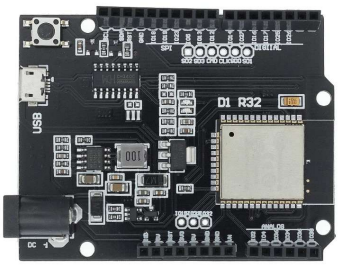


Figura 2.2: Tarjeta de desarrollo D1R32 Wemos.

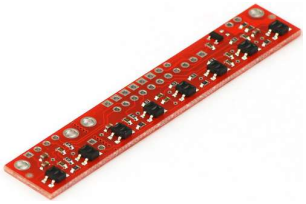


(a) Motor (b) Driver

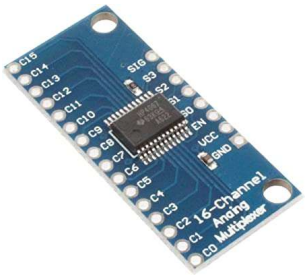
Figura 2.3: Componentes de actuación del robot diferencial.

Los motores, a través de la caja reductora, están conectados a ruedas con exterior de goma de 1.5 $[cm]$ de radio.

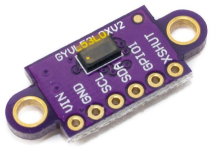
Para controlar los motores se utiliza el *driver* TB6612FNG Dual H-Bridge [21], el cual soporta dos motores DC a una corriente constante de 1.2 $[A]$ o 3.2 $[A]$ de peak, controlándolos mediante PWM por sus pines de entrada.



(a) Sensor IR.



(b) Multiplexor.



(c) Sensor de distancia.

Figura 2.4: Sensores del móvil diferencial.

2.1.3. Sensores.

Los motores utilizados en el proyecto incluyen un sensor *encoder* C6101 de efecto *hall* que mide la velocidad del motor mediante el posicionamiento de un imán dentro del mismo. Las emanaciones de OEM del imán son detectadas por el sensor, y al contar la cantidad de pulsos recibidos, conocer la relación de reducción del motor y el radio de la rueda, se puede calcular la distancia recorrida y la velocidad lineal del robot diferencial. Es importante destacar que estos pulsos se transforman en señales digitales.

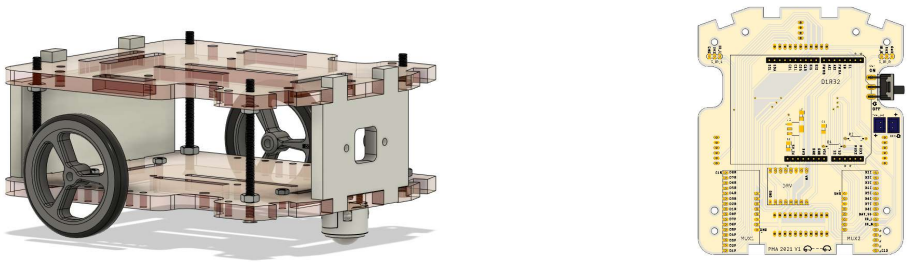
La detección de la trayectoria, definida por una línea en el suelo, se lleva a cabo mediante dos arreglos de sensores IR reflectivos QTR-8RC [19], ubicados en la parte inferior del móvil, uno en la parte frontal y el otro en la parte posterior, como se observa en la Figura 4.3. Estos sensores utilizan LED IR y fotoresistencias para medir la cantidad de luz IR reflejada en el suelo y determinar si el robot se encuentra sobre una línea diferente al suelo. El consumo máximo de corriente es de 17 $[mA]$, considerando la corriente utilizada por la placa y los 8 LED IR a su máxima luminosidad. Cada arreglo de sensores tiene 8 pines de salida analógicos.

Debido a que se utilizan dos arreglos de 8 sensores IR (para un total de 16 sensores), se requiere un multiplexor para obtener todas las lecturas con un número menor de GPIOs. Se utiliza el multiplexor CD74HC4067 [5] de 16 canales analógicos, que permite seleccionar el canal en la salida del multiplexor mediante el control de 4 pines digitales de entrada. Este componente consume 50 $[mA]$ a 3.3 $[V]$.

Para medir la distancia entre los móviles se utiliza el sensor VL53L0X [22], un sensor de distancia por tiempo de vuelo que emite un haz de luz láser a intervalos regulares. Al encontrar un objeto en su camino, el haz es reflejado y el sensor mide el tiempo que transcurre entre la emisión del haz y la detección de su reflejo para estimar la distancia hasta el objeto detectado. Esta medida se utiliza como entrada para controlar la distancia entre los agentes. Este sensor consume 10 $[mA]$ a 3.3 $[V]$ y se comunica con la unidad de control central a través de la interfaz I2C.

2.1.4. Alimentación.

La plataforma cuenta con dos baterías de litio 18650 [12], las cuales son celdas de 3.7 $[V]$ y tienen una capacidad típica de 2550 $[mAh]$. Estas están conectadas en serie para proporcionar un voltaje total de 7.4 $[V]$, el cual se utiliza para alimentar directamente los drivers de motor y la entrada de voltaje de la tarjeta de desarrollo D1 R32.



(a) Estructura. (b) PCB.

Figura 2.5: Estructura del móvil.

Todos los demás componentes de la plataforma son alimentados desde las salidas reguladas de voltaje de la tarjeta D1 R32, que proporciona un voltaje constante de 3.3 $[V]$.

Es importante destacar que el regulador de la tarjeta D1 R32 es capaz de suministrar un total de 500 $[mA]$.

2.1.5. Estructura.

El móvil se compone de dos placas de acrílico, una superior y otra inferior, que proporcionan la estructura necesaria para sostener todos los componentes, sujetadas mediante tornillos, tuercas y espaciadores, como se puede apreciar en la Figura 2.5a. La estructura completa mide 12 $[cm]$ de largo, 10 $[cm]$ de ancho y 4.5 $[cm]$ de alto.

Además de la estructura, se ha diseñado una placa de circuito impreso (PCB) específica para el proyecto, con las mismas dimensiones que las placas acrílicas, como se muestra en la Figura 2.5b. Esta placa realiza todas las conexiones de la plataforma y dispone de interruptores y encabezados para los diferentes componentes.

La plataforma también incluye un portapilas central para alojar las dos baterías de litio 18650.

Finalmente, la plataforma se completa con dos ruedas de 1.5 $[cm]$ de radio y una rueda tipo bola de giro sin fin que proporciona un punto de apoyo central a la estructura.

2.2. Diagrama de bloques.

Para comprender de manera funcional los componentes descritos anteriormente, se presentan diagramas de bloques que representan los distintos componentes de la plataforma y cómo interactúan entre sí.

En la Figura 2.6, se puede observar la interconexión de los componentes desde la perspectiva de la potencia. Los recuadros de la figura representan los distintos componentes de la plataforma, y las flechas indican la dirección que sigue la potencia. El voltaje de 7.4 $[V]$, que proviene de las baterías de litio, alimenta tanto la tarjeta del microcontrolador como el driver de motor. Desde el driver, el voltaje se modula mediante PWM (línea de color magenta) para accionar los motores. Además, la tarjeta del microcontrolador cuenta con un regulador de voltaje que suministra un voltaje de 3.3 $[V]$ (línea de color verde) a los sensores de distancia, IR, encoder derecho e izquierdo, multiplexor y circuito lógico del driver de motor.

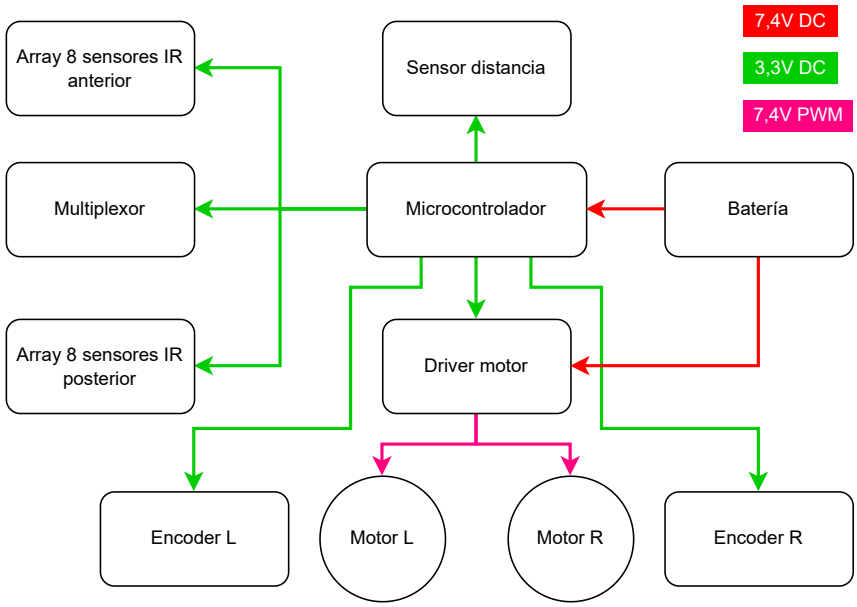


Figura 2.6: Diagrama de bloques que incluye señales de alimentación, de la plataforma.

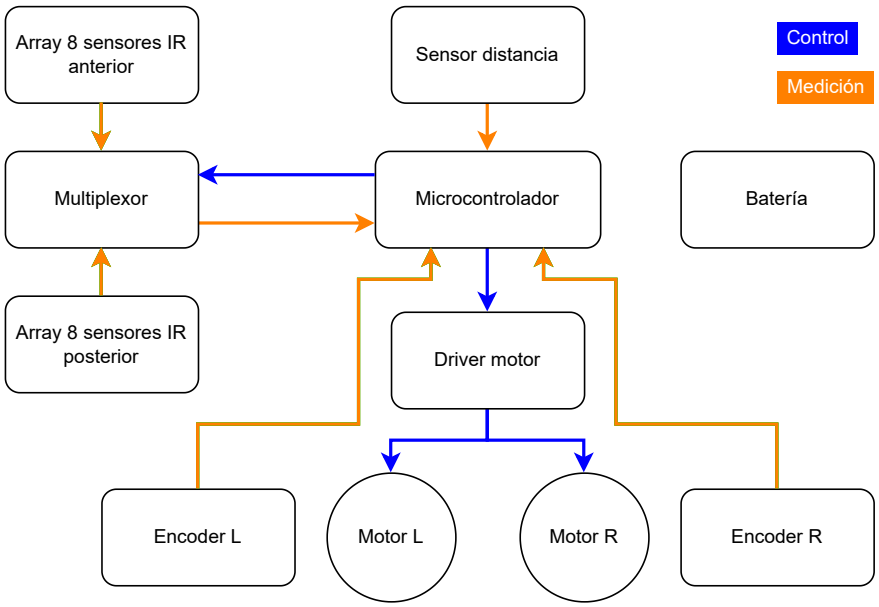


Figura 2.7: Diagrama de bloques que incluye señales de medición y control, de la plataforma.

Por otro lado, en la Figura 2.7 se presenta la interacción de los componentes desde la perspectiva de las señales de medición y control. En este caso, los recuadros representan los distintos componentes de la plataforma, y las flechas indican la dirección de las señales. El microcontrolador controla el multiplexor para elegir un canal específico de lectura y también controla el driver de motor para indicar el ciclo de trabajo para accionar los motores (línea de color azul). Además, la unidad de control central recibe las señales del sensor de distancia y los pulsos de los encoders derecho e izquierdo (línea

de color naranja). El multiplexor recibe 8 señales de cada arreglo de sensores IR y entrega una de estas señales a la unidad de control central para la medición.

2.3. Modelo matemático

Este trabajo de título utiliza el modelo desarrollado para la plataforma RUPU [9]. A continuación se plasman los resultados obtenidos luego de la derivación del modelo, presente en el escrito antes mencionado.

El modelo matemático utilizado refiere a un modelo en variables de estado presente en la ecuación (2.1).

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{2.1}$$

Considerando lo siguiente.

$$x = \begin{bmatrix} \dot{\phi}_r(t) & \dot{\phi}_l(t) & \dot{Q}_{mr}(t) & \dot{Q}_{ml}(t) & \phi_r(t) & \phi_l(t) \end{bmatrix}^T \tag{2.2}$$

Y definición de las variables.

- $\dot{\phi}_r(t)$ y $\dot{\phi}_l(t)$: Corresponden a las velocidades angulares de las ruedas derecha e izquierda, respectivamente, del robot. Su unidad es $[rad/s]$.
- $\dot{Q}_{mr}(t)$ y $\dot{Q}_{mr}(t)$: Corresponden a las corrientes de armadura de los motores derecho e izquierdo, respectivamente. Su unidad es $[A]$.
- $\phi_r(t)$ y $\phi_l(t)$: Corresponden a las posiciones angulares de las ruedas derecha e izquierda, respectivamente. Su unidad es $[rad]$.
- $u_{mr}(t)$ y $u_{ml}(t)$: Corresponden a las señales de entrada al sistema a través de los motores derecho e izquierdo, respectivamente. Su unidad es $[V]$.
- $d(t)$ corresponde a la distancia entre dos agentes consecutivos. Su unidad es $[cm]$.
- $\theta(t)$: Corresponde a la orientación del robot medida como el ángulo entre el eje central del robot y la recta tangente a la trayectoria de navegación. Su unidad es $[rad]$.

Así, utilizando los parámetros del móvil de la Tabla 2.1 y realizando distintas simulaciones, en [9] se obtienen los valores de las matrices del modelo en variables de estado.

$$A = \begin{bmatrix} -3.606 & 0.274 & 331.290 & -25.209 & 0 & 0 \\ 0.274 & -3.606 & -25.209 & 331.290 & 0 & 0 \\ -57.791 & 0 & -4595.856 & 0 & 0 & 0 \\ 0 & -57.791 & 0 & -4595.856 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.3}$$

Tabla 2.1: Parámetros del sistema.

Símbolo	**Parámetro**	**Valor**
r	Radio de las ruedas	0.0215 $[m]$
l	Distancia entre las ruedas del móvil	0.056 $[m]$
m	Masa del robot	0.445 $[kg]$
j	Momento de inercia del móvil	1.1968e-3 $[kgm^2]$
j_{mr}, j_{ml}	Momento de inercia de los motores	6.7738e-9 $[kgm^2]$
N_r, N_l	Módulo de transformación mecanismo reductor	0.01 $[-]$
η_r, η_l	Eficiencia de los motores	0.2 $[-]$
kv_r, kv_l	Constante de fuerza electromotriz de los motores	1.5844e-3 $[Vs/rad]$
kt_r, kt_l	Constante de torque de los motores	1.5844e-3 $[A/Nm]$
b_r, b_l	Constante de roce viscoso	3.45e-4 $[Ns/rad]$
L_{mr}, L_{ml}	Inducción de armadura de los motores	2.7416e-3 $[H]$
R_{mr}, R_{ml}	Resistencia de armadura de los motores	12.6 $[\Omega]$

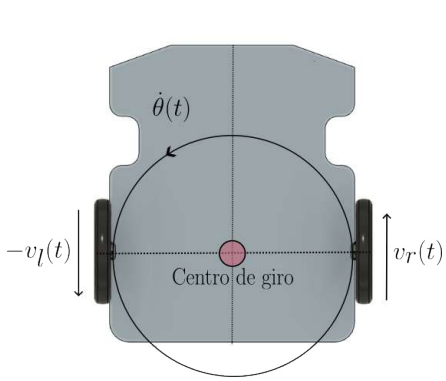
$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 364.750 & 0 \\ 0 & 364.750 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.4}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & -2.0750 & -1.0750 \\ 0 & 0 & 0 & 0 & 0.191 & -0.191 \end{bmatrix} \tag{2.5}$$

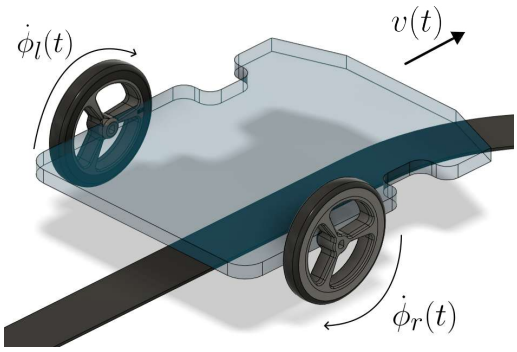
$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.6}$$

2.3.1. Controlador

Se implementa un esquema de control PID en cascada, como se muestra en la Figura 2.10. Así, se tienen tres controladores; uno para la distancia que existe entre dos agentes del pelotón, otro que controla el seguimiento de velocidad constante, y finalmente el que controla que el móvil se mantenga dentro de la trayectoria determinada.



(a) Medición de $\dot{\theta}$ y velocidades v_l y v_r.



(b) Medición de velocidad del móvil y velocidades angulares ϕ_l y ϕ_r.

Figura 2.8: Diagrama de velocidad lineal, velocidades angulares y ángulo de la trayectoria en un móvil.

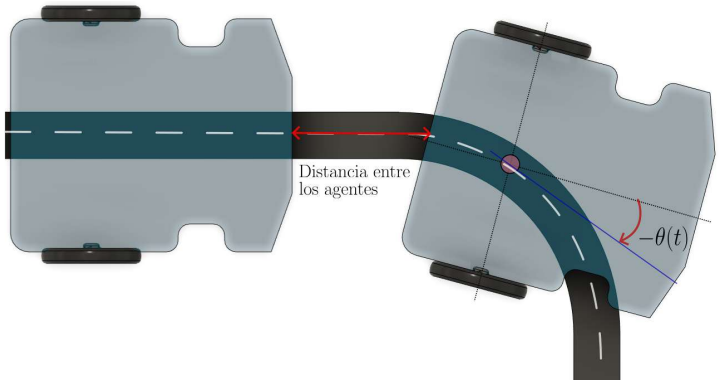


Figura 2.9: Diagrama de medición de distancia entre móviles y comparación de la trayectoria definida (línea punteada blanca) con la trayectoria del móvil, $\theta(t)$.

2.4. Funcionalidades

A continuación se presentan las características de la plataforma base con la que cuenta este proyecto de título, a nivel funcional:

- Capacidad de implementar controladores continuos de velocidad, trayectoria y distancia, en tiempo discreto.
- Capacidad de seguir una trayectoria definida por una línea de 2 [cm] de ancho, dibujada sobre una estructura plana, cuyo recorrido contenga curvas de más de 5 [cm] de radio.
- Capacidad de mantener una velocidad constante mientras se sigue dicha línea.
- Capacidad de mantener una distancia constante a un objeto (u otro móvil) antecesor al móvil.
- Capacidad de registrar mediciones de distancia al objeto frontal, velocidad lineal, distancia recorrida y detección de curvas en la trayectoria.

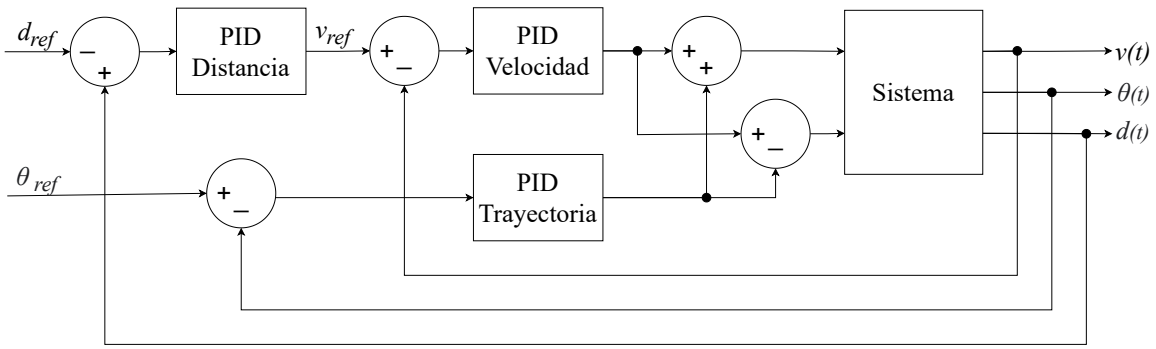


Figura 2.10: Diagrama de bloques del controlador planteado para el sistema.

Tabla 2.2: Parámetros del controlador del sistema.

Controlador	**Parámetro**	**Valor**
Velocidad	K_p	20.2912
	K_i	145.2779
	K_d	0
Distancia	K_p	-8.0013
	K_i	-5.6025
	K_d	-1.0077
Trayectoria	K_p	2755.3558
	K_i	6250.5619
	K_d	303.6521

- Capacidad de notificar las variables propias del móvil a un agente monitor, que reciba variables de distintos móviles.
- Capacidad de comunicar datos de velocidad, distancia, ángulo de seguimiento, a otro móvil de idénticas características por medio del protocolo de comunicación UDP. A su vez, es capaz de recibir estos valores y modificar parámetros de seguimiento.

Capítulo 3

Modificaciones a la plataforma base

Después de haber presentado la plataforma RUPU en el Capítulo 2, se ha formulado una nueva plataforma que replica las características y funcionalidades de la plataforma base, con una unidad de control central en una Raspberry Pi 4 modelo B, que se muestra en la Figura 3.1. Esta es un ordenador de placa reducida que cuenta con puertos GPIO que permiten la interacción física con el sistema, y que posee la capacidad de procesamiento necesaria para agregar nuevas prestaciones a la plataforma, como la posibilidad de realizar reconocimiento de imágenes para identificar la trayectoria a seguir.

Para implementar el proyecto en la nueva plataforma, se deben realizar varios ajustes en diferentes ámbitos de la plataforma, incluyendo las modificaciones realizadas en la plataforma base y los componentes nuevos que se deben agregar.

3.1. Modificaciones de la plataforma base

3.1.1. Cambio en el modelo matemático

Al realizar estos ajustes a la plataforma base, el modelo matemático y la dinámica del robot no se ven afectados. Los cambios solo afectan parámetros específicos, como la masa del móvil y la posición del centro de masa.

En la Tabla 2.1 se especifica una masa de 0.445 $[kg]$ para el robot en la plataforma base, la cual varía al trasladar la plataforma debido al cambio del dispositivo de procesamiento.

3.1.2. Cambio en los sensores

Se requieren cambios en la adquisición de datos, debido a que la plataforma base emplea señales digitales y analógicas de entrada (descritas en la Sección 2.1.3), lo que es incompatible con la Raspberry Pi, cuyos GPIO son exclusivamente digitales. Por lo tanto, es necesario utilizar un convertidor analógico-digital (ADC) para transformar estas señales analógicas en digitales con una resolución específica. En particular, se necesita convertir la señal analógica del multiplexor. Por lo tanto, se modifica el flujo de

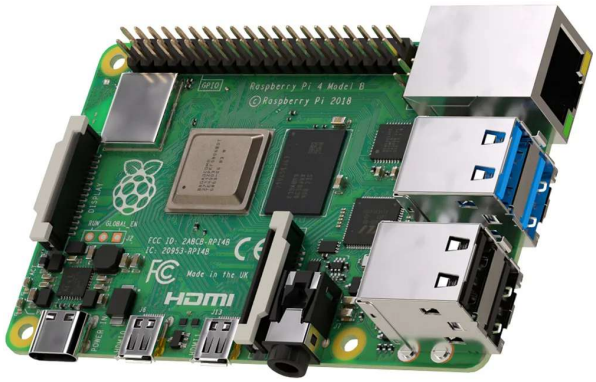
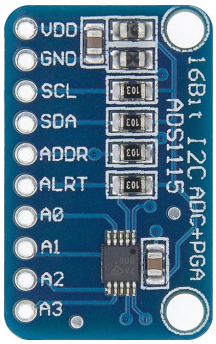
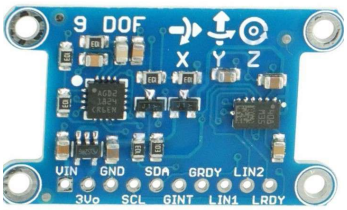


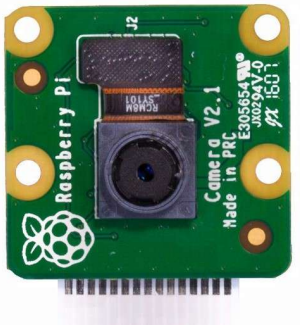
Figura 3.1: Raspberry Pi 4 modelo B.



(a) Módulo ADC ADS1115



(b) Módulo IMU 9DOF.



(c) Cámara RPi V2.1

Figura 3.2: Componentes extras para la plataforma.

información entre la unidad de control central y el multiplexor (presente en la Figura 2.7) para incorporar este convertidor. Los detalles de esta modificación se describen en la Sección 3.2.

3.2. Adición de sensores a la plataforma base

3.2.1. Conversor análogo digital.

Es necesario agregar un componente adicional para adquirir datos de los sensores IR utilizados en la plataforma base, ya que los puertos GPIO de la Raspberry Pi solo aceptan señales digitales, mientras que la salida del sensor es analógica. Para solucionar esto, se incorpora el módulo ADS1115 [2], que es un convertidor análogo-digital de 16 bits basado en el circuito integrado ADS1115. Este componente permite obtener hasta 860 lecturas por segundo a través de un bus de comunicación I2C y consume solo 150 $[\mu A]$ en modo de medición continua. La Figura 3.2a muestra el aspecto del componente.

3.2.2. Unidad de medición inercial.

Se agrega a la plataforma una unidad de medición inercial (IMU), la cual combina las lecturas del giroscopio y el acelerómetro de 3 ejes para estimar la ubicación relativa del móvil. El módulo 9DOF IMU [1] que se utiliza cuenta con los sensores LSM303DLHC

(acelerómetro y magnetómetro de 3 ejes) y L3GD20 (giroscopio de 3 ejes), una corriente base de 3.2 $[mA]$ y una interfaz de comunicación con protocolo I2C. Se puede ver la IMU en la Figura 3.2b.

Es importante tener en cuenta que la cantidad máxima de dispositivos que se pueden conectar a un bus que utiliza el protocolo de comunicación I2C está determinada por la cantidad de bits utilizados para definir la dirección de los dispositivos, generalmente 7 bits, lo que da un total de 128 posibles direcciones. Sin embargo, 16 de esas direcciones están reservadas para fines especiales, lo que significa que el máximo de nodos conectados al bus I2C en este caso es de 112, siempre y cuando no se encuentren duplicadas las direcciones de los componentes conectados. Además, esta cantidad también está limitada por la capacitancia permitida del bus, que es de 400 $[pF]$.

3.2.3. Cámara digital.

Con el fin de ampliar las prestaciones de la plataforma base y habilitar la opción de navegar por el escenario usando visión por computador, es que se agrega una cámara digital modelo RPI V2 [14] de 8 MP, que utilizando el sensor Sony IMX219 presente en la Figura 3.2c, es capaz de capturar imágenes de 3280 x 2464 píxeles y también soportar capturas de video de 1080p30, 720p60 y 640x480p60/90. Este módulo se conecta a la Raspberry Pi mediante un puerto CSi dedicado.

Adicionalmente, se diseña una estructura capaz de soportar y fijar la cámara a la plataforma base, la cual se describe en la Sección 3.4.

3.3. Variación en el requerimiento energético de la plataforma

Debido al cambio de la unidad de control central y adición de distintos sensores a la plataforma, es necesario realizar un cálculo del requerimiento energético actual y así identificar la capacidad de la batería necesaria a incorporar.

En la Tabla 3.1 está descrito cada componente y su correspondiente corriente requerida para su funcionamiento. Cabe destacar que hay algunos componentes que se encuentran por pares dentro de la plataforma, como lo son los *encoders*, motores y arreglos de 8 sensores IR.

A partir de los datos presentados en la Tabla 3.1, se puede estimar que la plataforma base necesita un total aproximado de 1.13 $[A]$. Si se asume un consumo adicional de 500 $[mA]$ por parte de la Raspberry Pi para ejecutar un código, se requeriría un total de 1.63 $[A]$ para que el móvil funcione correctamente. Sin embargo, la capacidad energética de la batería utilizada en la plataforma base es de 2550 $[mAh]$, lo que indica que aún puede ser utilizada en esta actualización.

Es importante mencionar que la Raspberry Pi necesita alrededor de 1 $[A]$ para operar correctamente, especialmente si se activa la interfaz gráfica. Aunque una Raspberry Pi 4 puede suministrar 16 $[mA]$ por pin GPIO, se debe tener en cuenta que los dispositivos

[1] La corriente consumida por los motores y encoders se encuentra considerada en la corriente del Driver motor.

[2] Esta corriente fue medida suponiendo que la Raspberry Pi 4 no tiene periféricos conectados y se encuentra en suspensión.

Tabla 3.1: Descripción de requerimiento de corriente de los componentes de la plataforma.

Función	**Componente**	**Corriente [mA]**
Seguimiento de línea	Array 8 sensores IR	17
	Multiplexor	10
	ADC	0.15
Seguimiento de distancia	Sensor de distancia	10
Ubicación relativa	Sensor IMU	3.2
Movimiento	Driver motor	250
	Motor	-[1]
	Encoder	-
Controlador	Raspberry Pi 4	575 [2]



(a) Módulo regulador de voltaje *step-down* DC-DC LM2596 [13]



(b) Fuente de voltaje externa de 5 [V].

Figura 3.3: Alternativas de alimentación para la Raspberry Pi.

conectados deben estar por debajo de este límite [3]. Sin embargo, es importante destacar que esta estimación no implica que cada pin GPIO pueda suministrar dicha cantidad de corriente en simultáneo, ya que la suma de los pines que utilizan lógica de 3.3 [V] debe ser menor a 50 [mA].

Ahora, la Raspberry Pi necesita una fuente de voltaje de 5 [V] puesto que no posee regulador de voltaje interno. Entonces, es necesario realizar un cambio con respecto a la alimentación de este, debido a que la batería de la plataforma tiene de salida 7.4 [V].

Para esto, se consideran las siguientes dos opciones de solución.

1. Incorporación de un regulador de voltaje que disminuya el voltaje de salida de la batería de 7.4 a 5 [V].

   Se considera el módulo con regulador de voltaje *step-down* DC-DC LM2596 [13] presente en la Figura 3.3a, el cual permite un voltaje de entrada entre 4.5 a 35 [V]

[3] Cabe destacar que el supuesto de que los Array 8 sensores IR requieran 17 [mA] no se cumple en este proyecto, ya que se utiliza una corriente menor debido a la máxima luminosidad de los LEDs IR.

y es regulado a un voltaje de salida entre 1.25 a 30 $[V]$. Mediante un potenciómetro multivueltas presente en el módulo se puede fijar la salida entre 1.25 y el voltaje de entrada, por lo que, en nuestro caso aplica para disminuir los 7.4 $[V]$ de la batería a los 5 $[V]$ requeridos por la Raspberry Pi. Así mismo, este módulo provee una corriente de salida de 3 $[A]$ nominales y un *peak* de 5 $[A]$ máximo, lo cual es suficiente para esta aplicación.

2. Incorporación de una fuente de voltaje externa e independiente, de 5 $[V]$.

   Se considera una batería externa portátil de 5 $[V]$, 5000 $[mA]$ y 100 $[g]$ de peso. Se puede observar en la Figura 3.3b.

- **Implementación de una alternativa de alimentación.**

  Se implementa la opción de integrar una fuente de voltaje adicional de 5 $[V]$. Para esto, se utiliza una *Powerbank* de 5000 $[mA]$ de bajo peso y dimensiones, la que se ubica junto a la Raspberry.

  Se realiza una prueba de duración de batería, la cual considera las siguientes funcionalidades:

  - La plataforma sigue una trayectoria definida.
  - La plataforma se encuentra informando sus parámetros por UDP.
  - La plataforma se encuentra escuchando a un puerto a través del protocolo MQTT.
  - La plataforma tiene habilitada su interfaz gráfica, además de mantener abierta una sesión de escritorio remoto.
  - La plataforma permanece capturando y enviando flujo de video desde la cámara frontal.

  En esta prueba, la batería externa de 5 $[V]$ duró un aproximado de 3 horas y 23 minutos, momento en que la plataforma deja de avanzar. Prueba de duración de batería de la plataforma.

  El voltaje de la batería que utilizan los motores bajó de 7.4 a 7.23 $[mA]$.

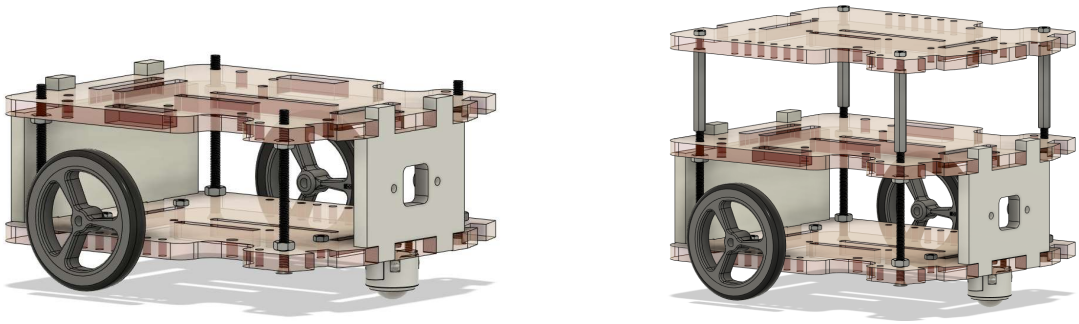
3.4. Cambios en la estructura de la plataforma

Para la estructura de la plataforma, se utiliza como base la ya desarrollada para RUPU, presente en la Figura 3.4a, adicionando un tercer nivel separado por espaciadores de 3 $[cm]$ y compuesto por una placa de acrílico que permite posicionar a la nueva unidad de control central, tal y como se puede observar en la Figura 3.4b.

Además se diseña un soporte para ubicar la cámara, la que se puede observar en la Figura 3.5.

Esta consiste en dos abrazaderas de metal que se fijan en los dos espaciadores frontales o posteriores del móvil, las que se conectan con el eje central del soporte de 10 $[cm]$ de largo, y que posibilita ajustes en la inclinación de la cámara en conjunto con tuercas para asegurar sujeción manteniendo en su posición a la cámara mientras el móvil se encuentra en movimiento.

En la parte central del eje se ubica el soporte donde se fija la cámara Modelo V2 [14], el cual consta de 2 partes del tamaño de la esta y por cuyo centro pasa el eje.



(a) Estructura base (b) Nueva estructura

Figura 3.4: Estructura del móvil con tracción diferencial.

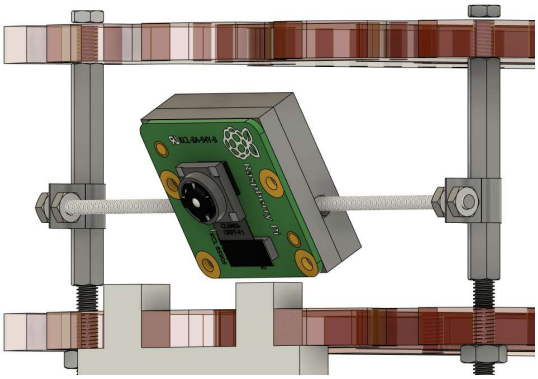


Figura 3.5: Estructura de soporte para adición de cámara, permitiendo regular su altura y grado de inclinación

3.5. Adición de una nueva PCB a la plataforma

Para adaptar la plataforma a la nueva versión incorporando la Raspberry Pi, se diseña una nueva PCB que captura los pines antiguamente utilizados en la PCB de la plataforma base, los que se distribuyen y ordenan para que sean compatibles con la disposición de los GPIO que posee la Raspberry Pi.

En la Figura 3.6 se observa un gráfico 3D de la PCB diseñada, mientras que en las Figuras 3.7a y 3.7b se observan las capas superior e inferior de la PCB.

A continuación se presenta una lista de las características de la nueva PCB:

- Incluye un cambio en la disposición de los pines antiguamente utilizados para el microcontrolador ESP32 para acondicionarlos a la ubicación de los GPIO de la Raspberry Pi.
- Incorpora el módulo de regulador de voltaje step-down DC-DC LM2596 [13], descrito en la Sección 3.3. Además, posee la opción de que permanezca desconectado en caso de realizar pruebas de alimentación.
- Disponibiliza pines con 3.3 $[V]$, GND y puertos SCL y SDA para facilitar la incorporación de nuevos sensores a través del protocolo de comunicación I2C.

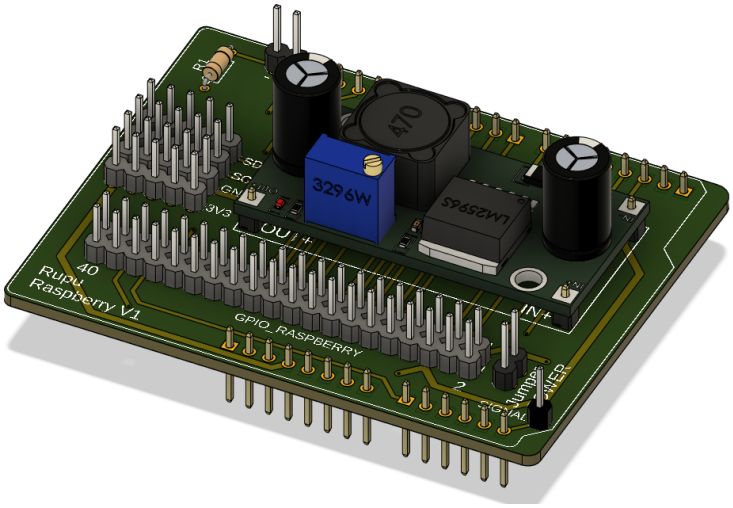
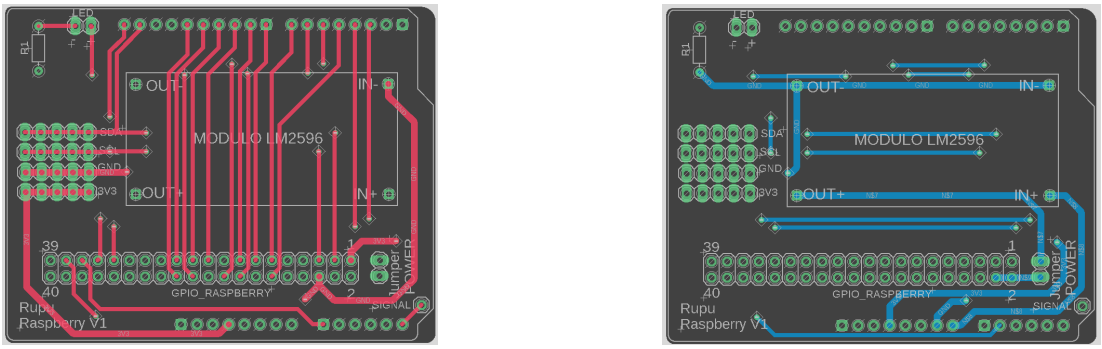


Figura 3.6: Gráfico 3D de la PCB adicional.



(a) Capa TOP de la PCB.

(b) Capa BOTTOM de la PCB.

Figura 3.7: Diagrama de conexionado de la nueva PCB.

- Incorpora conexiones necesarias para incluir un LED frontal que permita iluminar la proximidad del móvil para una mejor captura de vídeo, controlado por la Raspberry Pi.

Capítulo 4

Descripción e implementación de la plataforma actualizada

El propósito de este capítulo es describir detalladamente la consolidación de una nueva versión de la plataforma RUPU [9], destacando los aspectos clave de la actualización, como la estructura, componentes, algoritmos, flujos de control y de potencia, controladores y su implementación en la plataforma, y comunicación entre la flota, entre otros. Este capítulo culmina con pruebas de funcionamiento y rendimiento en la Sección 5.

En la Figura 4.1 se puede observar el móvil con tracción diferencial, que es el resultado del presente trabajo de título. Se utilizó una batería externa mencionada en las opciones de alimentación de la Sección 3.3 como fuente de energía.

Además, en la Sección 8.3 se muestran fotografías adicionales del móvil desde varios ángulos de vista.

4.1. Estructura

La estructura de la plataforma consta de tres niveles, los cuales están detallados en la Sección 3.4, compuestos por placas acrílicas.

En el primer nivel se encuentran el porta batería, los motores, y los sensores IR y de distancia. El segundo nivel incluye la PCB de conexión a la unidad de control central, el soporte y la cámara de la plataforma. La primera aproximación de la realización de este nivel se puede observar en la Figura 4.2b. Por otro lado, en la Figura 4.2a se puede apreciar la distribución del segundo nivel del móvil, en el cual se destaca la PCB diseñada para la actualización. Finalmente, en el tercer nivel se ubican la unidad de control central y una batería externa opcional.

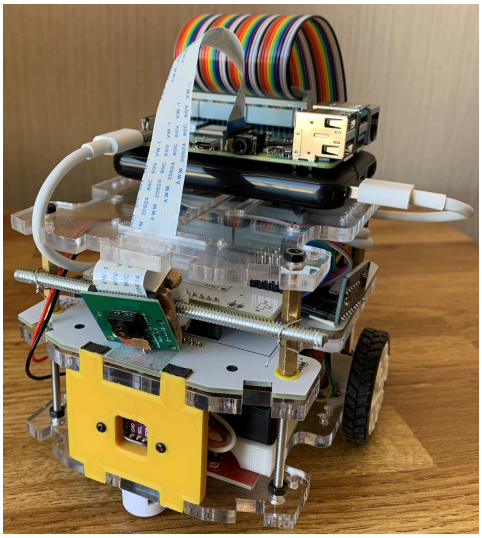
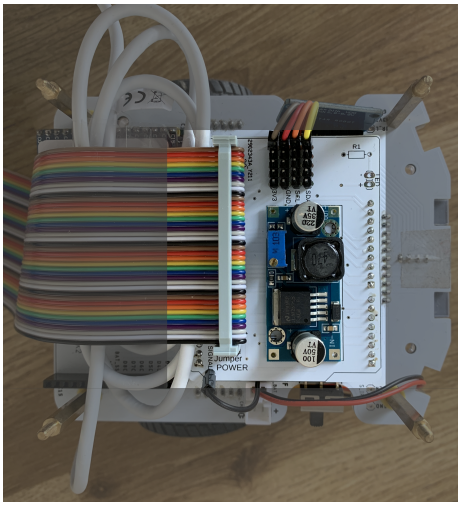
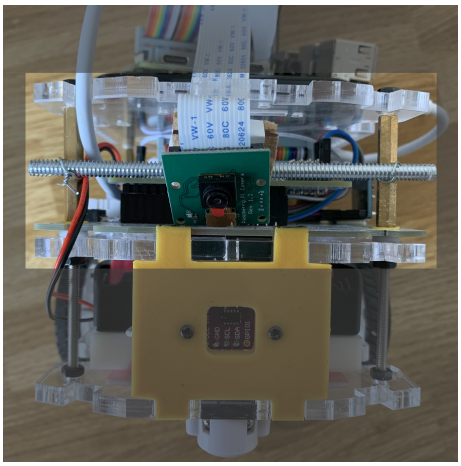


Figura 4.1: Fotografía de la nueva versión del móvil con tracción diferencial.



(a) PCB diseñada.



(b) Soporte para cámara.

Figura 4.2: Fotografías de la distribución de la estructura de la nueva plataforma.

4.2. Componentes

A modo de resumen, en la Tabla 4.1 se presentan los componentes utilizados en la plataforma actualizada, para posteriormente describir su implementación dentro de la plataforma.

4.2.1. Adquisición de datos

A continuación se describe a alto nivel la lógica de la adquisición de datos por parte de los sensores. Cabe destacar que los códigos de la plataforma se encuentran en *Python*.

[1]Se consideran ambas opciones de alimentación: regulador de voltaje a 5V o la utilización de una batería con voltaje de salida 5V.

Tabla 4.1: Resumen de los componentes que utiliza la actualización de la plataforma.

Agrupación funcional	**Componente**
Controlador	Raspberry Pi 4 Model B
Actuadores	Driver de motor TB6612FNG
	Motor DC N20 MicroGear
Sensores	Encoder C6101
	Array IR reflectivo QTR-8RC
	Multiplexor CD74HC4067
	ADC ADS1115
	Sensor láser VL53L0X
	IMU 9DOF
	Cámara RPI V2
Alimentación	Batería litio 18650
	Regulador de voltaje LM2596[1] o Batería 5V

Cámara RPI V2

Para la cámara que posee la plataforma actualizada, se utiliza la biblioteca *picamera*, la cual es capaz de entregar una vista real de la captura de la cámara, además de grabar vídeos con distintas resoluciones por cantidades de tiempo específicas. Luego, utilizando las bibliotecas de *opencv* y *numpy* es posible separar el flujo de video en *frames* para realizar operaciones de detección de objetos.

Encoder C6101

Se utiliza la biblioteca *Encoder*, la cual cuenta los pulsos que el sensor Encoder mide mientras gira la rueda del móvil con tracción diferencial. Como se conoce la relación de reducción que posee la caja de reducción que tiene incorporada el motor y también se conoce el radio de la rueda, entonces, la cantidad de pulsos medidos se pueden convertir en ángulo de rotación, velocidad angular, distancia lineal, y velocidad lineal de la rueda. Esta última resulta de nuestro interés y se puede describir bajo la Ecuación (4.1).

$$VelocidadLineal = \frac{2\pi \cdot RadioRueda}{\Delta Tiempo} \cdot \frac{PulsosEncoder}{CuentasPorRevolucion} \left[\frac{cm}{s}\right] \tag{4.1}$$

Sensor láser VL53L0X

Utiliza la biblioteca VL53L0X que entrega lecturas de distancia en $[cm]$, las cuales pueden ser corregidas agregando un *off-set* para cada sensor.

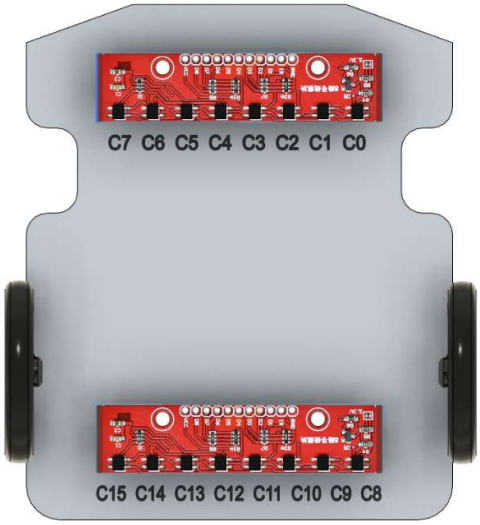


Figura 4.3: Vista inferior del móvil diferencial para visualizar los puerto de conexión al multiplexor de cada sensor de los Arrays IR.

ADC ADS1115

Corresponde al conversor análogo digital de la plataforma, y es empleado para convertir la señal análoga de la salida del multiplexor ocupado.

Utiliza la biblioteca ADS1X15 la cual registra las lecturas del sensor por medio del protocolo de comunicación I2C.

Multiplexor CD74HC4067

Las señales en la salida de este componente se miden por medio del ADC ADS1115. Es capaz de recibir señales de 16 canales análogos de forma alternada. Para escoger el canal a leer, posee 4 pines de entrada denominados S0, S1, S2 y S3. Así, los dos arreglos de sensores IR que posee el móvil con tracción diferencial se encuentran conectados a los 16 canales del multiplexor (denominados bajo el nombre de C0, C1, C2, ... , C15), siguiendo el orden de conexionado presente en la Figura 4.3.

Para elegir el canal específico a leer dentro de los 16 que posee, realiza una operación de *and* entre sus pines de entrada obteniendo un número decimal entre 0 y 15, arrojando el canal a leer. Esta lógica se puede observar en la Tabla 4.2.

Arreglo de sensores IR reflectivos QTR-8RC

Los datos del arreglo de sensores IR son adquiridos mediante el multiplexor CD74HC4067 en conjunto con el ADC ADS1115 que realiza la conversión del valor análogo a una señal digital. En el móvil con tracción diferencial se utilizan dos arreglos IR, ubicados en el frente y en la zona posterior del agente, dando en total 16 sensores que se conectan al multiplexor. En la Figura 4.3 se observa el pin de conexión que tiene cada sensor con respecto a las entradas del multiplexor.

Los valores que entrega cada sensor IR corresponden a un rango de 0 a 255 para indicar el porcentaje de absorción de luz IR por la trayectoria. Cabe destacar que las

Tabla 4.2: Direcciones utilizadas para leer puertos de entrada del Multiplexor.

Pin de dirección Multiplexor				**Pin de entrada a**
S3	S2	S1	S0	**leer del Multiplexor**
0	0	0	0	C0
0	0	0	1	C1
0	0	1	0	C2
0	0	1	1	C3
0	1	0	0	C4
0	1	0	1	C5
0	1	1	0	C6
0	1	1	1	C7
1	0	0	0	C8
1	0	0	1	C9
1	0	1	0	C10
1	0	1	1	C11
1	1	0	0	C12
1	1	0	1	C13
1	1	1	0	C14
1	1	1	1	C15

mediciones varían de sensor a sensor ya que no son exactamente iguales. En el Algoritmo 4.1 se ejemplifica el proceso de calibración de cada uno de los sensores del Array.

Algoritmo 4.1: Procedimiento para calibrar el arreglo IR.

```
1  Inicio
2  Inicializar variables: valor_minimo_sensor = [0]*16; valor_maximo_sensor =
       [32000]*16; valor = [0]*16
3  Mientras el movil gira 4 veces en su centro de giro, con velocidad de los
       motores de magnitud iguales pero de sentidos contrarios, Hacer:
4      Para s = 0 Hasta 15 con Paso de 1, Hacer:
5          Pin S0 del multiplexor = s&0x01
6          Pin S1 del multiplexor = s&0x02
7          Pin S2 del multiplexor = s&0x04
8          Pin S3 del multiplexor = s&0x08
9          Se lee la entrada del ADC y se almacena en la variable valor[s]
10         Si (valor[s] < valor_minimo_sensor[s])
11             Entonces valor_minimo_sensor[s] = valor[s]
12         Sino valor_minimo_sensor[s] mantiene su valor
13         Si (valor[s] > valor_maximo_sensor[s])
14             Entonces valor_maximo_sensor[s] = valor[s]
15         Sino valor_maximo_sensor[s] mantiene su valor
16 Pin S0 del multiplexor = 4&0x01
17 Pin S1 del multiplexor = 4&0x02
18 Pin S2 del multiplexor = 4&0x04
19 Pin S3 del multiplexor = 4&0x08
```

```
20 Se lee la entrada del ADC y se almacena en la variable valor[4]
21 Mientras (100 - 100*(valor[4] - valor_minimo_sensor[4])/(valor_maximo_sensor
       [4]-valor_minimo_sensor[4]) <80), Hacer:
22     Girar con velocidad constante
23     Se lee la entrada del ADC y se almacena en la variable valor[4]
24 Parar motores
25 Fin
```

Observando el algoritmo descrito en 4.1, se destaca que se realiza un ciclo utilizando la variable auxiliar *s* para poder obtener los valores de todos los sensores de los dos arreglos que utiliza el móvil diferencial. De allí, se almacena para cada sensor, un valor máximo y mínimo que luego servirá para discriminar si el móvil se encuentra encima de la trayectoria predefinida por una línea. Luego, entre las líneas 16 a 19 del algoritmo, se fija en el multiplexor la dirección 0100 por medio de operaciones al bit. La dirección corresponde al sensor C4 de la Figura 4.3, ubicado en la parte central del arreglo frontal del móvil. En las líneas 20 y 23 se vuelven a activar los motores y se mide el valor del sensor C4 hasta que este estime que se encuentra en el centro de la trayectoria, y con esto se termina la rutina de calibración de los sensores IR.

Algoritmo 4.2: Procedimiento para estimar la posición de la trayectoria a través del arreglo IR.

```
1  Inicio
2  Inicializar variables: sensor_normalizado = 0; area_local = 0; suma = 0;
       area_total = 0; parar = "no"
3  Para s = 0 Hasta 7 con Paso de 1, Hacer:
4      Pin S0 del multiplexor = s&0x01
5      Pin S1 del multiplexor = s&0x02
6      Pin S2 del multiplexor = s&0x04
7      Pin S3 del multiplexor = s&0x08
8      Se lee la entrada del ADC y se almacena en la variable valor
9      Se normaliza la variable valor para que se encuentre entre 0 a 100, se
       almacena en la variable sensor_normalizado
10     Si (sensor_normalizado < 30)
11         Entonces sensor_normalizado = 0
12 Para t = 0 hasta 7 con Paso de 1, Hacer:
13     area_local = 8*sensor_normalizado - 0.04*sensor_normalizado*
       sensor_normalizado
14 Se realiza una suma ponderada de la variable area_local de cada sensor, con un
        factor que depende de la posicion del sensor en el arreglo, de la forma
       -28*C0 - 20*C1 - 12*C2 - 4*C3 + 4*C4 + 12*C5 + 20*C6 + 28*C7, y se
       almacena en la variable suma
15 Se realiza una suma de la variable area_local de cada sensor y se guarda en la
        variable area_total
16 Si area_total = 0
17     Entonces el movil no se encuentra encima de la linea, se fija la variable
       parar en "si" para que se detenga
18 Se calcula la division suma / area_total y se devuelve de la funcion,
       entendiendo que mas cercano al 0, mas centrado en la linea se encuentra
19 Fin
```

- **Estimación de la posición del móvil con respecto a la trayectoria predefinida.**

  Se especifica el pseudo-código que sigue la plataforma para estimar la posición del móvil con respecto a la trayectoria predefinida, en el Algoritmo 4.2.

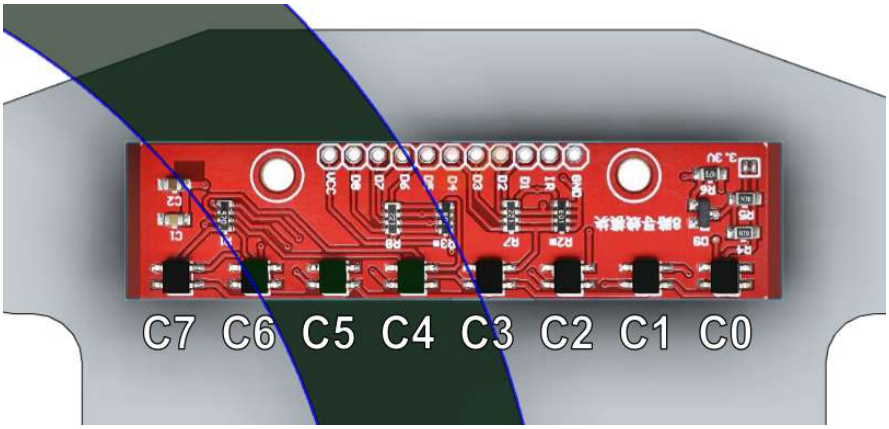


Figura 4.4: Ejemplo de medición de cantidad de desviación del móvil.

Por ejemplo, se tiene la situación de la Figura 4.4, donde se obtienen los valores C0:0, C1:0, C2:0, C3:20, C4:100, C5:100, C6:80 y C7:0, normalizados en el rango 0-100 producto de la calibración del sensor.

Luego, se pasa a calcular el área local de cada sensor según $A_i = 8 * C_i - 0.04 * C_i^2$, después se calcula una estimación de la posición de la línea ponderando los valores por la posición relativa al centro del arreglo IR, considerando una separación de 8 $[mm]$ entre sensor. Así, se realiza la operación $-28 \cdot A_0 - 20 \cdot A_1 - 12 \cdot A_2 - 4 \cdot A_3 + 4 \cdot A_4 + 12 \cdot A_5 + 20 \cdot A_6 + 28 \cdot A_7$ dando un valor de 13504, indicando que la línea se encuentra inclinada hacia el lado derecho del móvil por ser un valor positivo.

Conjuntamente, se calcula la suma de estos valores de área local dando un valor de 1320, sugiriendo que aproximadamente 4 sensores se encuentran detectando la línea, dado que el área máxima de cada sensor es de 400.

Luego, la cantidad de desviación se calcula como la razón entre la suma ponderada y el área total detectada, dado un valor de 10,2 [mm] de desviación con respecto al centro del arreglo. Este valor será cercano a cero cuando la línea se encuentre bajo el centro del sensor.

Finalmente, esta medición será dividida por la distancia entre el arreglo de sensores y el eje de las ruedas para estimar una aproximación del ángulo de desviación que se usará como entrada para el controlador de trayectoria, que para este caso de ejemplo da un valor de 0.2 $[rad]$ de desviación del robot respecto de la trayectoria.

- **Estimación de zona de curva.**

  Según la lectura del arreglo en conjunto con el *Encoder*, es posible estimar si el móvil se encuentra transitando por una curva. Esto tomando como supuesto que cuando el móvil se encuentra en un recta, las distancias recorridas por cada rueda serán muy similares, por lo que, si se toma como parámetro relevante la diferencia entre estas, se puede saber que se encuentra en una curva cuando el valor supere un cierto umbral. Ahora bien, como esta diferencia queda dependiendo de la velocidad a la que se desplace el móvil, es que se normaliza usando la distancia promedio recorrida por las dos ruedas.

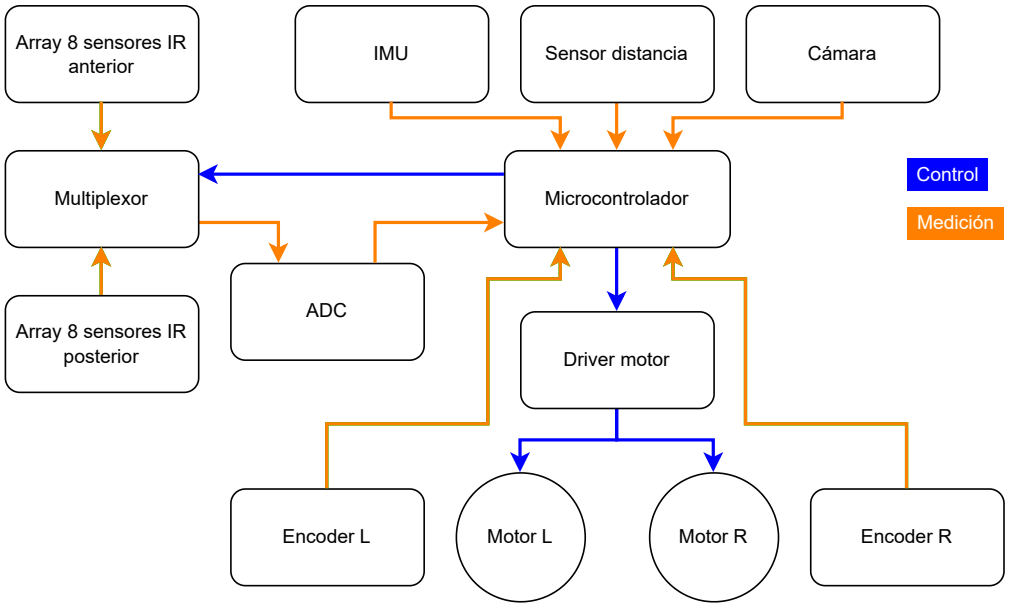


Figura 4.5: Diagrama de bloques que incluye señales de medición y control de la actualización de la plataforma.

También cabe destacar que debido a perturbaciones en las mediciones, irregularidades de la pista, recta imperfecta, oscilaciones en la actuación de los motores, etc. es probable que la magnitud normalizada antes mencionada posea oscilaciones que en algunos casos superen el umbral momentáneamente generando falsos positivos a la detección de una curva, por lo que, se implementa una redundancia con 3 variables auxiliares, y cuando la magnitud de la curvatura es mayor al umbral consistentemente en las 3 variables auxiliares al mismo tiempo, entonces, se sabe que se ha detectado una curva.

IMU 9DOF

Se utilizan distintas bibliotecas propietarias de *Adafruit*, las cuales son:

- *Adafruit LSM303DLH Mag* que entrega lecturas del magnetómetro en los tres ejes en $[Gauss]$.
- *Adafruit LSM303 Accel* que entrega lecturas del acelerómetro en los tres ejes en $[m/s^2]$.
- *Adafruit L3GD20* que entrega lecturas del giroscopio en los tres ejes en $[rad/s]$.

4.3. Diagrama de bloques

Para la actualización de la plataforma, se presentan dos diagramas de bloques para explicar el flujo de energía entre los distintos componentes de la plataforma, además de el flujo de control y medición.

En las Figuras 4.6 y 4.7 se observan dos opciones de flujo de energía. La primera opción, Figura 4.6, considera la incorporación de una nueva batería de 5 $[V]$ que alimenta

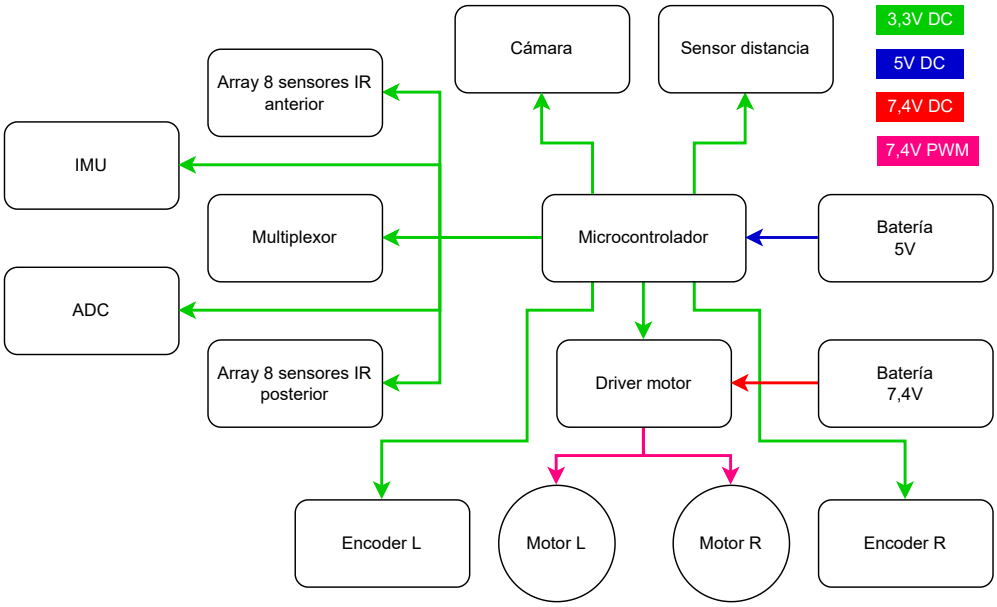


Figura 4.6: Diagrama de bloques que incluye señales de alimentación de la actualización de la plataforma, opción 1.

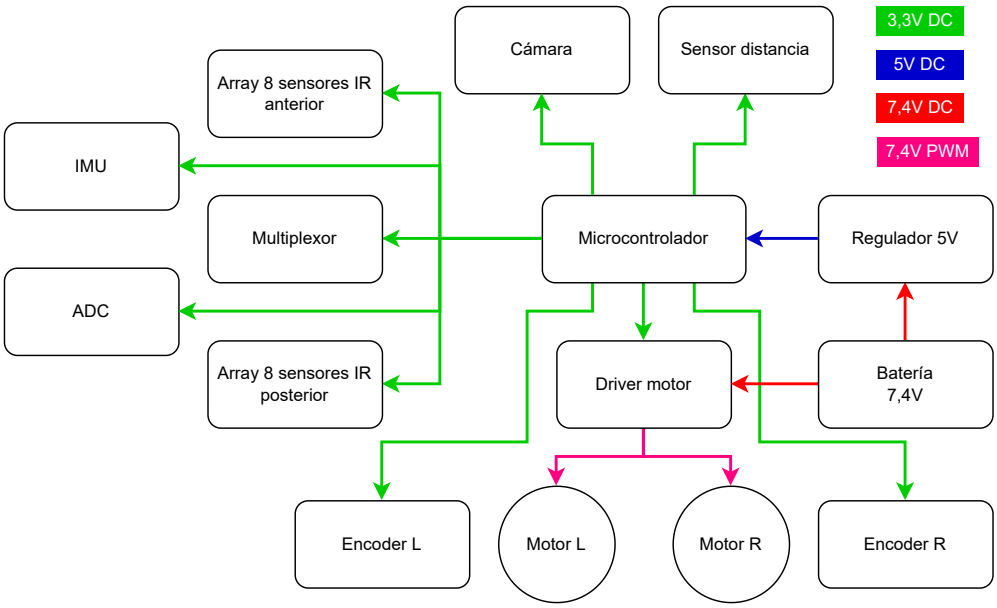
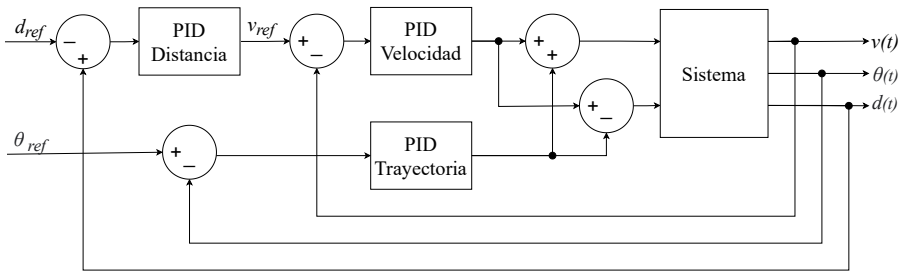


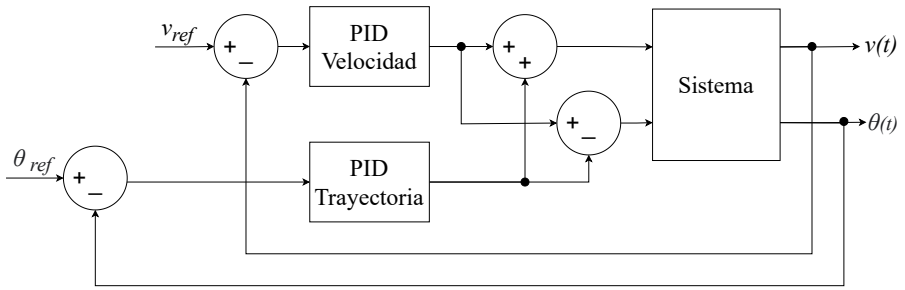
Figura 4.7: Diagrama de bloques que incluye señales de alimentación de la actualización de la plataforma, opción 2.

a la unidad de control central, funcionando en simultáneo a la batería de 7.4 $[V]$ utilizada originalmente.

En cambio, la Figura 4.7 considera la utilización de una única batería de 7.4 $[V]$, la que alimenta por una parte a los motores a través del driver, y por otra parte a un regulador *step-down* el que regula el voltaje de la batería a 5 $[V]$ para alimentar a la unidad de control central.



(a) Esquema de control para móviles sucesores.



(b) Esquema de control para móvil líder.

Figura 4.8: Alternativas de esquemas de control para los distintos móviles del pelotón.

Y la Figura 4.5 muestra el flujo de control y medición presente en la actualización de la plataforma. En ella se observan los siguientes cambios en comparación con la plataforma original presente en la Figura 2.7.

- El flujo de medición del componente multiplexor ahora pasa a través del ADC, que convierte la señal análoga en digital, debido a que la Raspberry Pi solo tiene puertos digitales.
- Se agregan los componentes IMU y Cámara.

4.4. Controlador

El controlador que se implementa en la plataforma está representado en la Figura 4.8, el cual consiste de dos alternativas según la posición del móvil dentro del pelotón.

4.4.1. Esquema de control para móvil sucesor.

El esquema de control para móviles sucesores se encuentra en la Figura 4.8a. En este se observan tres controladores PID ubicados en cascada que regulan tres variables: Distancia con respecto a un móvil predecesor, Velocidad del móvil y seguimiento de Trayectoria.

Controlador de distancia.

El controlador de distancia recibe como entrada el error de seguimiento con respecto a la distancia de referencia, y entrega como salida la velocidad de referencia que ha de

seguir el móvil, y en base a esta se calcula el error de seguimiento para el siguiente controlador de velocidad acorde a la velocidad medida.

Controlador de velocidad.

El controlador de velocidad tiene como entrada la diferencia entre la velocidad de referencia (salida del controlador de distancia) y la velocidad medida del móvil, y entrega de salida una variable acondicionada para los rangos de operación del motor, que corresponden de 0 a 1024.

Controlador de trayectoria.

Por otra parte, se tiene al controlador de trayectoria que recibe el ángulo de desviación a la trayectoria θ para calcular el error de seguimiento a esta variable y sirva de entrada al controlador. Cabe destacar que θ_{ref} debe ser nulo para asegurar seguimiento de la trayectoria. Luego, la salida del controlador, en conjunto con la salida del controlador de velocidad, entra al sistema dictando la velocidad de cada motor.

4.4.2. Esquema de control para móvil líder.

El esquema de control para el móvil líder se encuentra en la Figura 4.8b. Este se deriva del esquema de control para móviles sucesores, ya que utiliza la misma lógica de los controladores, solo que obviando el controlador de distancia.

Así, la velocidad de referencia del móvil líder ya no es el resultado de un controlador, si no que corresponde a un parámetro predefinido.

4.4.3. Modificación del esquema de control mediante comunicación entre la flota.

De forma adicional, utilizando la funcionalidad de comunicación entre los agentes de la flota presente en la Sección 4.5.2, se implementa una lógica complementaria para los móviles sucesores. Esta se basa en que cuando el móvil predecesor detecta que se encuentra en un trayecto de curva, este comunica un indicador al agente predecesor para que desactive momentáneamente su controlador de distancia, manteniendo su última velocidad de referencia. Luego, cuando acaba la curva y comienza un tramo recto, el móvil se lo indica al predecesor para que se vuelva a activar el seguimiento de distancia. Esta lógica se encuentra en el Algoritmo 4.3.

Algoritmo 4.3: Procedimiento para definir la velocidad de referencia de un móvil seguidor en zonas de curvatura.

```
1 Inicio
2 Si (movil lider esta en recta) & (movil seguidor esta en recta)
3     Se utiliza el controlador PID de distancia
4 Sino
5     No se utiliza el controlador PID de distancia
6     Se hace cero el error acumulado para el PID de distancia
7     Se mantiene el ultimo valor de velocidad de referencia
```

4.4.4. Implementación del controlador en tiempo discreto

Se implementan tres funciones dentro del código que utiliza la plataforma, mediante *Python*. Estas corresponden a los controladores de cada variable de interés. En el Algoritmo 4.4 se presenta un pseudo-código que ilustra la lógica a seguir para implementar el controlador en tiempo discreto.

Algoritmo 4.4: Procedimiento para la implementación de controladores PID en tiempo discreto.

```
1  Inicializar variable: tiempo_transcurrido_ultimo_calculo; actuacion = 0
2  Funcion calculo PID(salida, referencia, error_anterior, error_integral, kp, ki
       , kd, limite, direccion):
3  Si direccion = "DIRECTO"
4      error = referencia - salida
5  Si direccion = "INVERSO"
6      error = salida - referencia
7  error_integral = error_integral + error * tiempo_transcurrido_ultimo_calculo
8  Si error_integral * ki > limite
9      error_integral = limite / ki
10 Si error_integral * ki < - limite
11     error_integral = - limite / ki
12 actuacion = kp * error + ki * error_integral + kd * (error - error_anterior)/
       tiempo_transcurrido_ultimo_calculo
13 error_anterior = error
14 Si actuacion > limite
15     Retornar limite
16 Si actuacion < - limite
17     Retornar - limite
18 Sino
19     Retornar actuacion
20 Fin Funcion
```

Antes de explicar el pseudo-código, es importante definir ciertos aspectos que son relevantes para la implementación del controlador. En primer lugar, se debe considerar la diferencia entre la salida de la planta y el valor de referencia como el error en la implementación. En segundo lugar, las posibles entradas a la función son variables medidas dentro del móvil. En tercer lugar, la salida de la función son valores de entrada para los motores del móvil, que se traducen en velocidades lineales de cada rueda. Por lo tanto, la salida de actuación del controlador está limitada o saturada.

A continuación se realiza una explicación y descripción de la función implementada.

La **línea 2** define la función llamada *calculo PID*, la cual recibe los siguientes parámetros:

- *salida*: corresponde al valor actual de salida de la planta.
- *referencia*: valor de referencia de la variable de interés a controlar.
- *error_anterior*: valor de error anterior (iteración previa) de la variable de interés a controlar, a utilizarse para el aspecto derivativo del controlador.
- *error_integral*: valor de error anterior (iteración previa) de la variable de interés a controlar, a utilizarse para el aspecto integral del controlador.
- *kp*: constante de proporcionalidad del controlador.
- *ki*: constante de integración del controlador.

- *kd*: constante derivativa del controlador.
- *limite*: valor máximo (saturador) de la salida del controlador.
- *direccion*: valor que define la dirección de la acción del controlador. Puede adquirir los valores "DIRECTO" o "INVERSO".

Las **líneas 3 a 6** definen el comportamiento del error en función de la dirección de la acción del controlador. Si la dirección es directa, un aumento en la señal de la salida de la planta produce un aumento en la actuación del controlador. Si la dirección es inversa, un aumento en la señal de salida produce una disminución en la actuación del controlador. Por ejemplo, en un controlador de velocidad, se desea que el controlador aumente la actuación cuando la velocidad aumente (dirección directa), mientras que en un controlador de distancia, se desea que la velocidad disminuya para obtener mediciones mayores de distancia (dirección inversa).

A partir de la **línea 7**, se describe el funcionamiento del controlador. En la línea 7, se calcula el error integral utilizando la variable *error_integral* para acumular el error integral hasta el momento de ejecución de la función. Se suma el error multiplicado por el Δt, que es el tiempo entre ejecuciones de la función medido por el temporizador *tiempo_transcurrido_ultimo_calculo*, que corresponde a la frecuencia de muestreo de la implementación (40 [ms]).

Después de obtener el error integral, se calcula la salida de la parte integral del controlador (**líneas 8 a 11**). Aquí se comprueba que la salida del integrador no sobrepase el límite impuesto, saturándolo si es necesario.

En la **línea 12**, se calcula la salida o actuación del controlador. Primero, se calcula el aporte proporcional multiplicando el error por la constante de proporcionalidad *kp*. Luego, se calcula el aporte integral multiplicando el error integral por *ki*. Finalmente, se calcula el aporte derivativo simulando la derivada del error. Para hacerlo, se realiza la diferencia entre el error actual y el anterior, se divide por el Δt correspondiente a *tiempo_transcurrido_ultimo_calculo* y se multiplica por *kd*.

En la **línea 13**, se almacena el error de la iteración anterior en la variable *error_anterior*.

Por último, entre las **líneas 14 y 19**, la función retorna *limite* si la actuación está en estado de saturación, en caso contrario devuelve la actuación.

Variación en el controlador de distancia

La implementación del controlador de distancia difiere en el caso en que el sucesor pierde de vista al móvil delantero. Este problema se debe a que el sensor de distancia mide valores muy altos al no tener nada delante, lo que provoca que la salida del controlador de distancia se dispare generando velocidades de referencia muy grandes para el controlador de velocidad en cascada. Como resultado, el móvil acelera de forma drástica y no puede reaccionar a tiempo para evitar una colisión con el predecesor. Además, durante una curva, también se acumula el error integral, lo que agrava la situación.

Para remediar esta situación, se aplican ciertos cambios que siguen la siguiente estrategia: cuando se detecta que varios móviles consecutivos están en una curva, se "apaga" el controlador de distancia y se fija la salida en el último valor calculado, al mismo tiempo que se amortigua el error integral acumulado. Además, se modifica el rango de actuación del controlador de distancia, permitiendo generar velocidades de salida (referencia del controlador de velocidad) en torno a +/-10 % de la velocidad de referencia del

predecesor. Esto evita las aceleraciones bruscas del sucesor cuando el predecesor sale de la curva.

En el Algoritmo 4.5 se pueden observar los cambios aplicados.

Algoritmo 4.5: Procedimiento para la implementación de controlador PID de distancia, en tiempo discreto.

```
1  Funcion calculoPID_d(salida, referencia, error_anterior, error_integral, kp,
       ki, kd, limite, MODO, out_manual, direccion):
2  Si MODO == "MANUAL"
3      Retornar out_manual
4  Sino
5      ...
6
7      Si actuacion > limite * 1.1 + 1
8          Retornar limite * 1.1 + 1
9      Si actuacion < limite * 0.9 - 1
10         Retornar limite * 0.9 - 1
11     Sino
12         Retornar actuacion
13 Fin Funcion
```

Del Algoritmo 4.5, se observa que el rango de actuación pasa de ser de $2 * limite$ a ser $2 + 0.2 * limite$

Originalmente, la actuación del controlador varía entre $-limite$ hasta $limite$ provocando un rango de actuación de dos veces el límite. En cambio, con las modificaciones, se tiene que la actuación varía entre $0.9 * limite - 1$ hasta $1.1 * limite + 1$ arrojando un rango de $2 + 0.2 * limite$, el cual, para valores de $limite$ mayores a $\frac{10}{9}$ es siempre menor al caso original. Cabe mencionar que el offset de $+/-1$ está para el caso en que el predecesor indique una velocidad de referencia igual a 0, es decir, esté detenido, y el sucesor pueda tener un margen para acomodarse y alcanzar la distancia de referencia fijada.

Se utiliza esta saturación únicamente para el controlador de distancia debido a la estructura en que están implementados los controladores, cuyo diagrama de bloques se encuentra en la Figura 4.8. En esta se observa que la salida del controlador de distancia corresponde a la entrada del controlador de velocidad como se mencionó anteriormente.

4.4.5. Parámetros de los controladores

Luego de describir cómo funciona la implementación en tiempo discreto del controlador PID, es necesario especificar las distintas variables que recibe cada controlador, que aparecen en la Tabla 4.3.

Cabe destacar que el periodo de muestreo para la implementación se estos controladores PID es de 40 $[ms]$.

Finalmente, el driver de motor recibe un valor entre 0 y 1024 para cada motor, el cual se traduce en una señal PWM de ciclo de trabajo variable.

Así, y acorde a la Figura 4.8, la salida del PID Distancia corresponde a la velocidad de referencia para el PID Velocidad. Luego, la suma de los PID Velocidad y PID trayectoria corresponde a la entrada de un driver de motor, mientras que el motor del lado opuesto recibe como entrada la salida del PID Velocidad restada la salida del PID Trayectoria.

Tabla 4.3: Variables que recibe de entrada cada función que implementa los controlador.

Variable	**PID Distancia**	**PID Velocidad**	**PID Trayectoria**
salida	Medición del sensor de distancia $[cm]$.	Medición de velocidad lineal promedio entre las ruedas en $[cm/s]$.	Medición del ángulo de desviación desde el centro del arreglo de sensores IR a la trayectoria en $[rad]$.
referencia	Distancia de referencia.	Velocidad de referencia.	Desviación de referencia.
error_ anterior	Error de distancia de la iteración anterior.	Error de velocidad de la iteración anterior.	Error de trayectoria de la iteración anterior.
error_ integral	Error integral de distancia de la iteración anterior.	Error integral de velocidad de la iteración anterior.	Error integral de trayectoria de la iteración anterior.
kp	8.0013	20.2915	2755.3558
ki	5.6025	145.2779	625.5619
kd	1.0077	0	303.6521
limite	velocidad de referencia del predecesor +/- 1	1023	1023
direccion	Inverso	Directo	Directo

4.5. Comunicación

Se implementan dos formas simultáneas y complementarias de comunicación descritas a continuación. Cabe destacar que la implementación de estas se realiza mediante *multi-threading*, donde el procesador es capaz de ejecutar varias tareas de forma simultánea, evitando *deadlocks* o bloqueos.

4.5.1. Control de parámetros

Utilizando el protocolo MQTT, se implementa comunicación inalámbrica con el servidor *Ubidots*, para poder enviar desde su *dashboard* operacional constantes necesarias para el controlador. Así, se pueden enviar los parámetros requeridos por el controlador para realizar de forma dinámica la calibración del mismo, mientras que el móvil sigue su recorrido. Tal *dashboard* se puede observar en la Figura 4.9.

La actualización de los parámetros realizados en el *dashboard* surgen efecto de inmediato en el móvil, debido a que este se encuentra suscrito a los tópicos referentes a cada variable de interés. Cuando surge una actualización de esos tópicos, el hilo creado para esta comunicación recibe automáticamente las novedades del tópico suscrito.

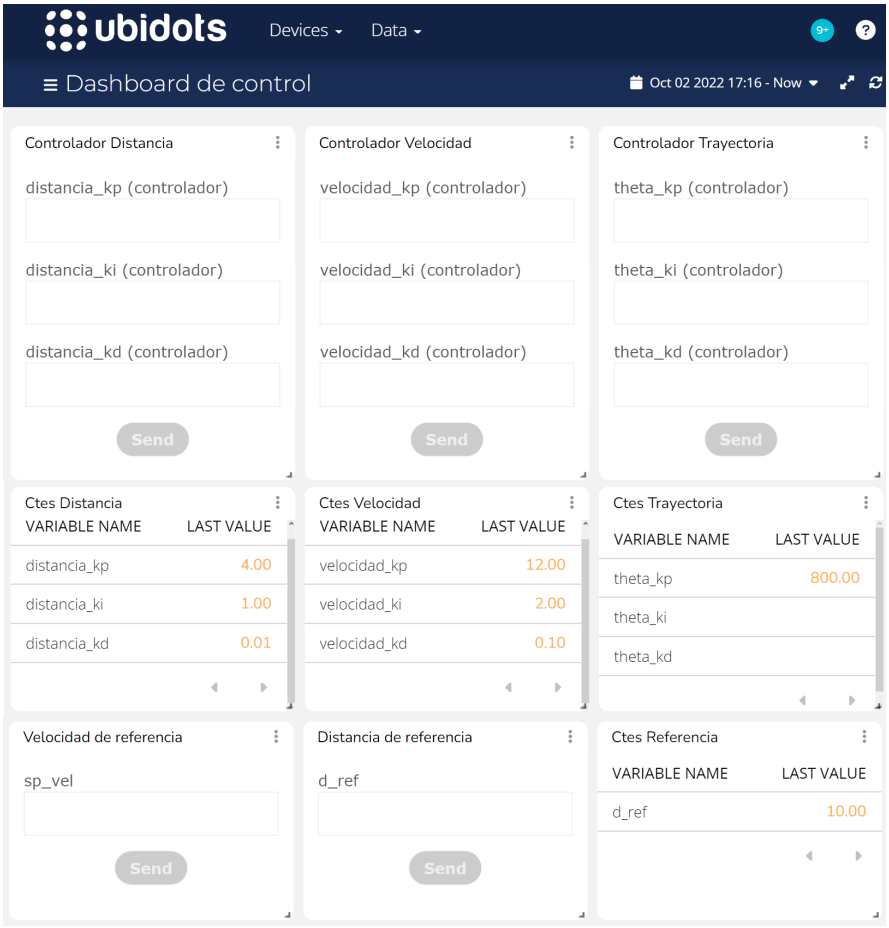


Figura 4.9: Captura del Dashboard de control desarrollado dentro de *Ubidots*.

Así, se abre la posibilidad de variar en tiempo real los siguientes parámetros del móvil:

- Constantes del controlador PID Distancia.
- Constantes del controlador PID Velocidad.
- Constantes del controlador PID Trayectoria.
- Velocidad de referencia.
- Distancia de referencia.

4.5.2. Reporte a un agente monitor y entre la flota

De forma paralela se implementa un algoritmo que permite generar y enviar un reporte de estado de los móviles con el resto de la flota, el cual puede constituirse por vehículos formados con Raspberry Pi o ESP32, utilizando el protocolo UDP.

Se utiliza un protocolo distinto al de control (MQTT) debido a que, en este caso, se envía información de velocidad, distancia y ángulos, variables necesarias para el recorrido

de los demás móviles y que tienen una frecuencia de actualización de aproximadamente 16 $[Hz]$ (60 $[ms]$).

En este caso, se mantienen las siguientes comunicaciones:

- Reporte a un agente monitor, externo a la flota de móviles. El monitor recibe de cada móvil una cadena de texto compuesta por los siguientes datos:
  - Código de identificación de la posición del móvil en la flota. Puede ser Líder, Seguidor 1, Seguidor 2, etc.
  - Medición de distancia actual al predecesor.
  - Distancia de referencia al predecesor.
  - Medición de velocidad actual.
  - Velocidad de referencia.
  - Medición de ángulo de trayectoria actual.
  - Ángulo de trayectoria de referencia.
- Reporte a un agente predecesor, para informar del estado de la trayectoria. El móvil envía una cadena de texto compuesta por los siguientes datos:
  - Código booleano que indica si el móvil se encuentra recorriendo la trayectoria, o si está detenido.
  - Medición de distancia actual al predecesor.
  - Medición de velocidad actual.
  - Indicador de si el móvil se encuentra recorriendo una curva.

Capítulo 5

Resultados

En este capítulo se presentan los resultados obtenidos tras verificar el correcto funcionamiento de cada característica necesaria para la nueva versión de la plataforma. Asimismo, se evalúa el rendimiento de las nuevas aplicaciones integradas.

5.1. Funcionamiento basal

5.1.1. Comunicación - Reporte

Para comprobar la funcionalidad de comunicación entre la flota con un agente externo a la flota utilizando el protocolo UDP, se realizan los siguientes experimentos.

Experimento de comunicación entre agentes.

Se cuenta con 4 agentes para el experimento, los cuales están descritos en la Tabla 5.1.

Tabla 5.1: Agentes que forman parte del experimento de comunicación y reporte.

Agente	**Dirección IP**	**Descripción**
Monitor	192.168.100.9	Externo a la flota.
Líder	192.168.100.18	Móvil con tracción diferencial primero en el pelotón.
Seguidor1	192.168.100.20	Móvil con tracción diferencial sucesor al Líder.
Seguidor2	192.168.100.22	Móvil con tracción diferencial sucesor al Seguidor1 y último en el pelotón.

Se sigue la siguiente lógica de reporte:

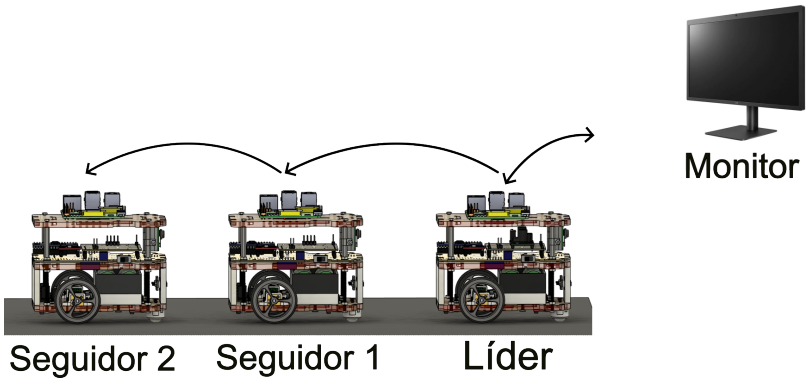


Figura 5.1: Gráfico del esquema de comunicación utilizado para el experimento de comunicación con tres agentes y un monitor.

1. Monitor le solicita datos al agente Líder.
2. Agente Líder recibe la petición del Monitor y le informa con una cadena de variables preestablecida.
3. Agente Líder informa a Seguidor1 una cadena de variables preestablecida.
4. Agente Seguidor1 recepciona la cadena enviada por Líder, e informa a Seguidor2 una cadena de variables prestablecida.
5. Agente Seguidor2 recepciona la cadena enviada por Seguidor2.

Inicialmente, se realiza un experimento con tres móviles que contienen la Raspberry Pi, cuyos resultados se presentan en el video Demostración comunicación UDP de 3 móviles con Rapsberry Pi, el cual muestra capturas de las interfaces gráficas de cada agente.

En la parte superior izquierda del video se puede observar la interfaz gráfica del Monitor, desde donde se solicitan datos al móvil diferencial con IP 192.168.100.18, correspondiente al agente Líder, mediante el envío de la cadena L/". A continuación, se recibe la cadena V/si/19/19/6" desde el Líder.

Por otra parte, en la parte superior derecha del video, se puede ver la interfaz gráfica del Líder, donde se recibe la cadena L/" que indica que el Monitor está solicitando datos, y se le responde con la cadena V/si/19/19/6", la cual coincide con la recibida inicialmente por el agente Monitor. Posteriormente, el Líder envía la cadena "V/si/9/20/20" al móvil con IP 192.168.100.20, correspondiente al Seguidor1.

En la parte inferior izquierda del video, se observa la interfaz gráfica del Seguidor1, donde se recibe la cadena enviada por el Líder y se envía a continuación la cadena "V/si/8/15/18" al agente Seguidor2, tal como se muestra en la parte inferior derecha del video.

En conclusión, se verifica que la característica de comunicación entre la flota de reporte de la plataforma actualizada se encuentra funcionando correctamente.

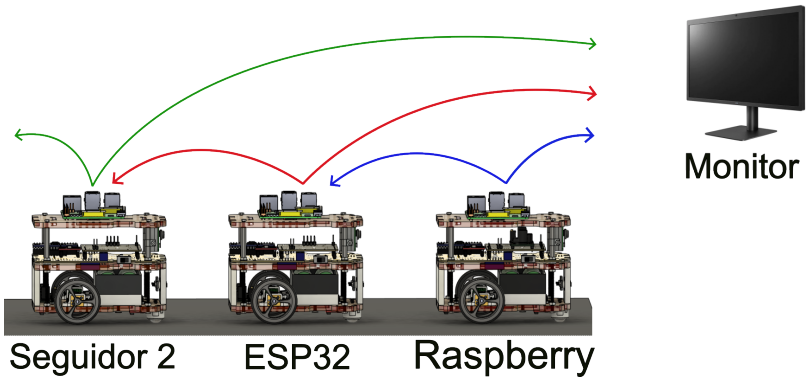


Figura 5.2: Gráfico del esquema de comunicación utilizado para el experimento de compatibilidad de comunicación entre plataformas.

Experimento de compatibilidad de comunicación entre plataformas.

Se desarrolla además un segundo experimento para comprobar la compatibilidad de la nueva plataforma con la original, que posee el microcontrolador ESP. Para este, se utiliza la siguiente lógica de reporte, presente en la Figura 5.2.

1. Agente Monitor recibe en todo momento los reportes de los miembros de la flota.
2. El móvil con tracción diferencial correspondiente a la plataforma actualizada, denominado agente Rapsberry, reporta sus datos constantemente al Monitor y a su agente sucesor.
3. El móvil con tracción diferencial correspondiente a la plataforma original, denominado agente ESP32, reporta sus datos constantemente al agente Monitor y a un agente sucesor a él en la flota.
4. Todos los demás miembros de la flota reportan al Monitor y a su sucesor.

En el video Demostración de la comunicación UDP entre móviles con Raspberry Pi y ESP32, se presentan los resultados del experimento a través de las interfaces gráficas de cada agente.

En la sección inferior del video se encuentra la interfaz gráfica del Monitor, que recibe los reportes de ambos agentes participantes.

En la porción izquierda del video se muestra una captura de la interfaz gráfica del agente con la plataforma actualizada, que envía constantemente cadenas al Monitor y al otro agente del pelotón.

Mientras que en la porción derecha del video, se puede observar la interfaz gráfica del agente con la plataforma original, en la que también se envían constantemente reportes.

Por lo tanto, se puede concluir que hay compatibilidad entre las plataformas para la comunicación.

Es importante destacar que el agente Monitor tiene la capacidad de almacenar los registros que recibe o envía a través del protocolo UDP en un archivo CSV para su posterior análisis.

5.1.2. Comunicación - Control

Para comprobar la funcionalidad de recibir parámetros vía inalámbrica desde un *dashboard* de control, primero se debe destacar que la actualización de la plataforma utiliza el protocolo de comunicación MQTT junto con el *dashboard* de control Ubidots, explicados en detalle en la Sección de Comunicación (4.5). Así, se dispone de dos móviles con tracción diferencial, y se realizan cambios desde el *dashboard* de control, observando las interfaces gráficas de ambos móviles en conjunto al *dashboard*.

El resultado del experimento se puede observar en el video Demostración funcionamiento con dashboard de control Ubidots. En este, su parte inicial consiste en una captura al *dashboard* de control, donde se envían los siguientes cambios en las variables: Velocidad de referencia con el valor de 10; constante del controlador de velocidad Kd al valor de 100 para luego bajarlo nuevamente a 0.01; y finalmente, volver a variar la velocidad de referencia a 30. Cabe destacar que en la parte inferior del *dashboard*, se observan los últimos valores de cada variable.

Ahora, continuando con el video, se observan las interfaces gráficas de los móviles con tracción diferencial. Para ambos, el resultado del experimento es el mismo ya que todos los agentes de la flota actualizan automáticamente cualquier variación de los parámetros presentes en el *dashboard*.

A continuación se pasa a revisar algunas líneas de código relevantes:

- `Suscrito a: /v1.6/devices/controlador/+/lv`

  Esta línea indica que los agentes se encuentran suscritos a una dirección en específico, que proviene de la plataforma *Ubidots*. Así, cualquier cambio que exista en los componentes que pertenecen a esa dirección se verán reflejados de forma automática en los dispositivos suscritos. Cabe destacar que el comando + significa que toma en cuenta cualquier sub-tópico dentro de `/v1.6/devices/controlador/`, y el último comando `lv` indica que se suscribe al sub-tópico que almacena únicamente los últimos valores de la variable.

- `sisub: msg received with topic: /v.16/devices/controlador/.../lv`

  `and payload: b'...'`

  Esta línea indica, primero, que se recibió una actualización de un tópico específico suscrito. Dentro de las líneas del código aparecen distintos sub-tópicos como lo son `theta_kp`, `theta_ki`, etc. Luego, indica el valor de la carga útil (*payload*) o actualización del valor del tópico suscrito, indicando con una `b` que el mensaje corresponde a una cadena de bytes, la que luego se debe decodificar en base a UTF-8.

De esta forma, los móviles al momento de comenzar su funcionamiento, se suscriben a todos los tópicos que son mantenidos en *Ubidots*, y luego, cada vez que existan modificaciones en sus valores, estos son actualizados.

Así, queda comprobado el correcto funcionamiento de la comunicación con objetivo de control, para la actualización de la plataforma.

Tabla 5.2: Parámetros del controlador del sistema utilizados en los experimentos.

Controlador	**Parámetro**	**Valor**
Velocidad	K_p	12
	K_i	2
	K_d	0.1
Distancia	K_p	4
	K_i	1
	K_d	0.01
Trayectoria	K_p	800
	K_i	30
	K_d	0.1

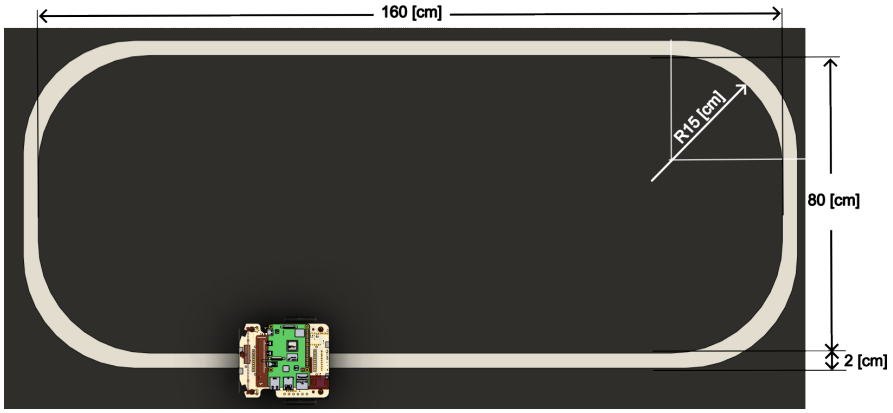


Figura 5.3: Gráfica del recorrido utilizado para el experimento de seguimiento de la trayectoria.

5.1.3. Seguimiento de velocidad y trayectoria

Para verificar el correcto funcionamiento del seguimiento de velocidad referencial y la trayectoria de los móviles con tracción diferencial, se utiliza la funcionalidad de comunicación y reporte. Cada agente informa sus variables al Monitor, que se utiliza para recopilar los datos con fines de análisis. Desde el dashboard de control *Ubidots*, se varía la velocidad de referencia que debe seguir el móvil.

Para llevar a cabo la prueba, se coloca un móvil con tracción diferencial en una superficie negra con una línea blanca de 2 $[cm]$ de ancho que indica la trayectoria a seguir, como se muestra en la Figura 5.3. La trayectoria es un rectángulo de 160 $[cm]$ de ancho y 80 $[cm]$ de profundidad, con aristas redondeadas bajo una circunferencia de 15 $[cm]$ de radio.

Cabe destacar que durante los primeros 18 $[s]$ de la prueba, el móvil comienza con una velocidad de referencia de 30 $[cm/s]$, se cambia a 0 $[cm/s]$, y luego se varía en 5 $[cm/s]$ hasta regresar a 30 $[cm/s]$, para finalmente detenerse.

Los controladores del móvil, que se describen en la Sección 4.4, incluyen los paráme-

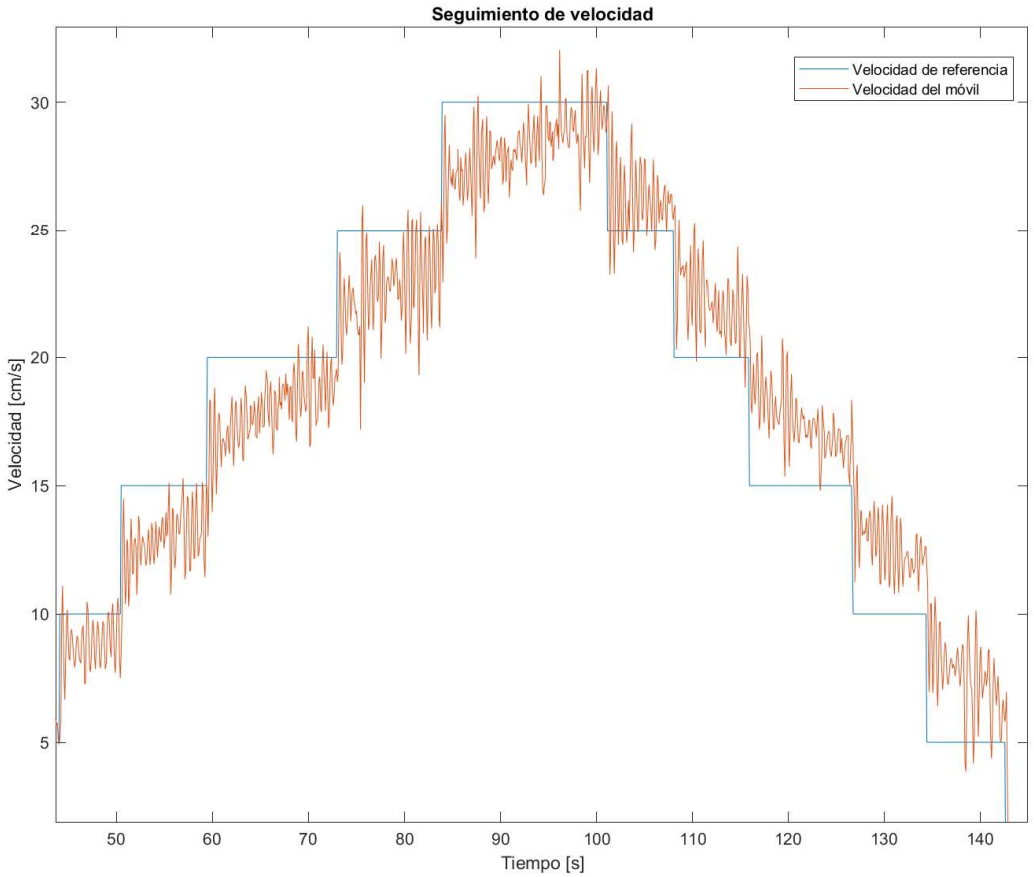


Figura 5.4: Gráfica del seguimiento de velocidad del móvil con tracción diferencial.

tros de la Tabla 5.2.

Seguimiento de velocidad.

La Figura 5.4 muestra el seguimiento de la velocidad de referencia (línea azul) que debe seguir el móvil, junto con la velocidad medida del mismo (línea roja), calculada a través de la ecuación (4.1) utilizando el sensor *Encoder* de la plataforma.

Al inicio del experimento, la velocidad de referencia es de 30 $[cm/s]$, y el móvil parte de una velocidad nula, tomando alrededor de 18 $[s]$ para acercarse a la velocidad de referencia. Luego, se fija la velocidad de referencia en 0 $[cm/s]$ y, posteriormente, en 5 $[cm/s]$. El móvil tarda aproximadamente 20 $[s]$ en alcanzar la nueva velocidad de referencia y, al aumentar la velocidad de referencia en 5 $[cm/s]$ escalonadamente, el móvil oscila en torno a dicha velocidad, tardando cerca de 5 $[s]$ en estabilizarse.

Es importante destacar que debido a las curvas e imperfecciones de la trayectoria a seguir, se observan fluctuaciones en la velocidad medida, ya que el móvil debe ajustar su velocidad para mantenerse en la línea. No obstante, en general, se puede observar que el móvil es capaz de seguir una velocidad de referencia establecida.

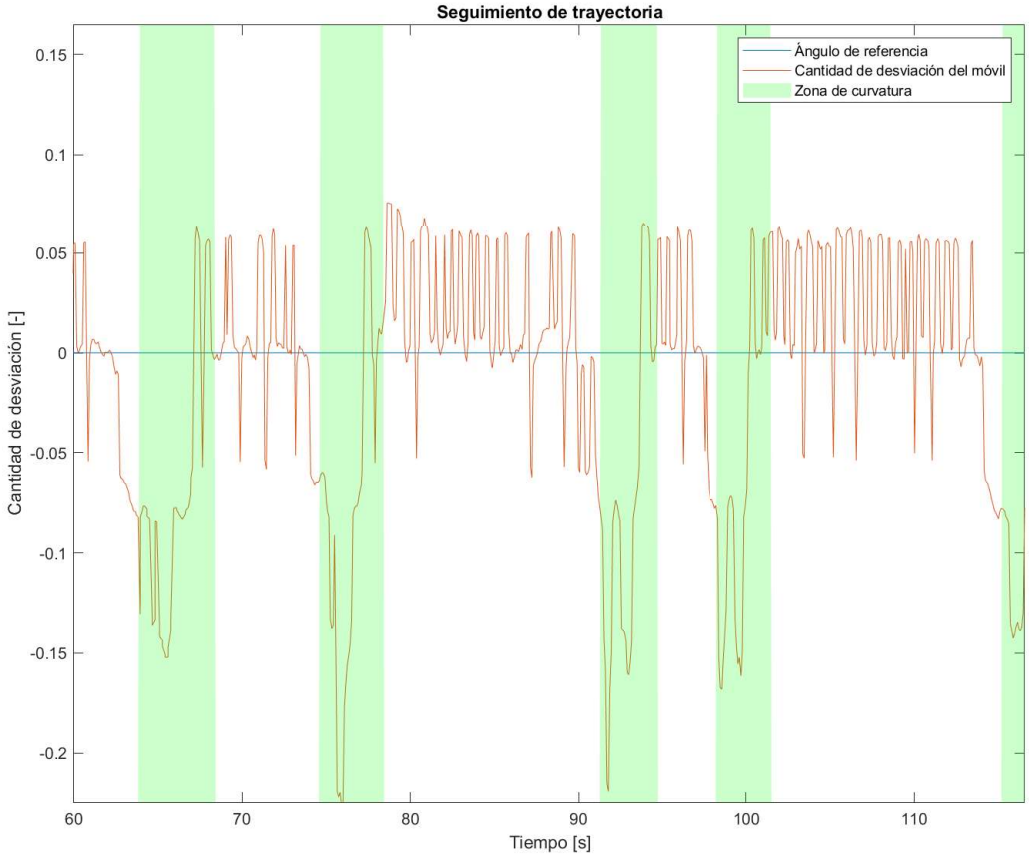


Figura 5.5: Gráfica del seguimiento de trayectoria del móvil con tracción diferencial.

Seguimiento de trayectoria.

En la Figura 5.5 se puede ver que el ángulo de referencia que debe seguir el móvil está representado en azul, mientras que la cantidad de desviación en radianes con respecto a la trayectoria se muestra en rojo. Esta desviación se calcula utilizando el procedimiento del Algoritmo 4.2.

Al observar detenidamente el gráfico de la Figura 5.5, se puede notar que la línea roja representa la cantidad de desviación del móvil con respecto a la trayectoria esperada. Esta desviación oscila en torno al cero, pero persiste hacia valores positivos, lo que podría ser indicativo de un posible error de medición o una inclinación del móvil hacia un lado debido a la construcción de los motores, ruedas, cajas reductoras, etc. Estas diferencias en la velocidad generan desviaciones en la trayectoria del móvil. Además, se puede notar que las mayores desviaciones se producen en las zonas de curvatura, lo que se refleja en las zonas de color verde.

Es importante mencionar que, en este experimento, se ha obtenido la máxima desviación de seguimiento de trayectoria a una velocidad de 25 $[cm/s]$. Esto destaca la relación entre la velocidad de recorrido y el rendimiento del seguidor de línea.

En resumen, se puede afirmar que el móvil es capaz de seguir una trayectoria con desviaciones menores a 0.1 $[rad]$ en tramos rectos y menores a 0.25 $[rad]$ en tramos curvos, lo que indica que la plataforma tiene la capacidad de seguir una trayectoria

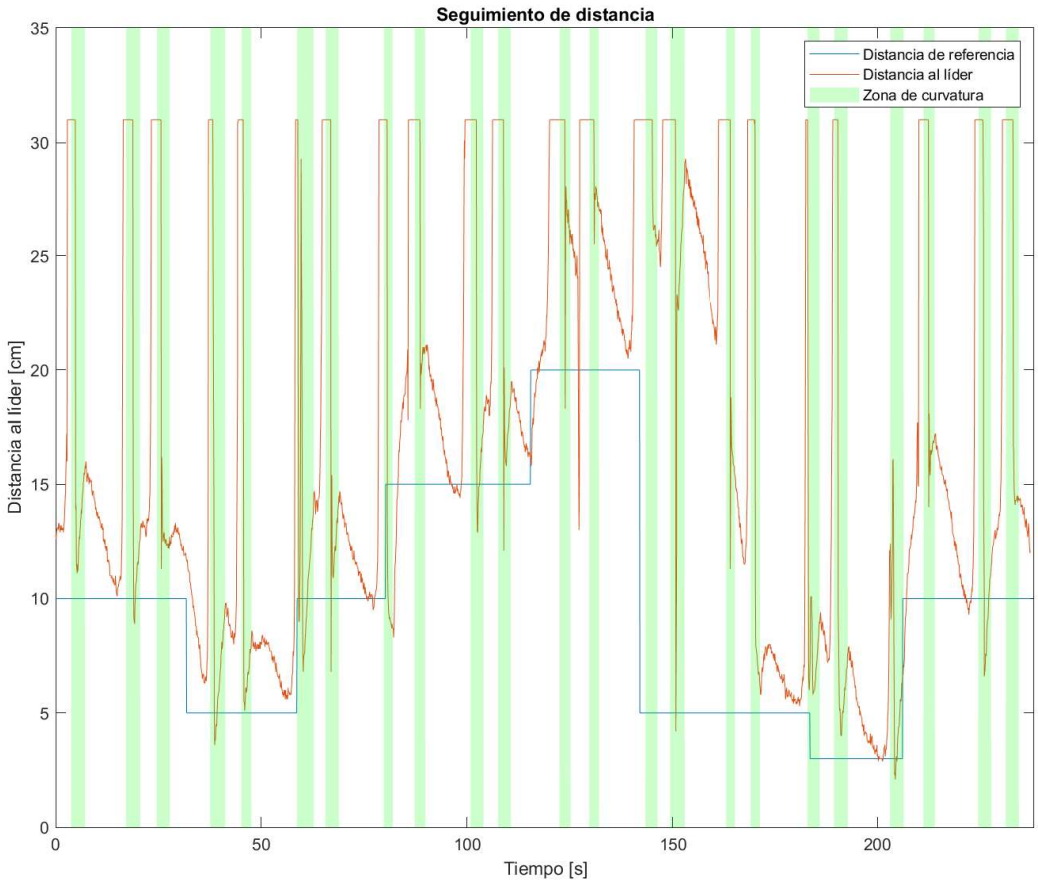


Figura 5.6: Gráfica del seguimiento de distancia al móvil Líder.

dada.

El resultado del experimento se puede observar en el video Experimento 1 móvil - Seguimiento de velocidad de referencia.

5.1.4. Seguimiento de distancia

Se realiza una prueba para verificar el correcto funcionamiento del seguimiento de distancia referencial en una situación de pelotón. Para ello, se varía la distancia que debe mantener el móvil seguidor respecto al líder desde el dashboard de control *Ubidots*, mientras se monitorea la velocidad, distancia y seguimiento de línea de cada móvil. En la Figura 5.3, se muestra el escenario utilizado en el experimento, donde dos móviles recorren la trayectoria predefinida, uno en el frente como líder y el segundo como seguidor. Ambos móviles comparten los mismos parámetros de control, presentes en la Tabla 5.2. El móvil líder mantiene una velocidad de referencia constante de 30 $[cm/s]$.

La Figura 5.6 muestra el resultado del experimento de seguimiento de distancia de referencia. La línea de color azul representa la distancia de referencia, que comienza en 10 $[cm]$ respecto al móvil líder y varía entre 5 y 20 $[cm]$. La línea de color rojo indica la distancia medida por el sensor de distancia del móvil seguidor. Aunque el móvil seguidor se acerca a la referencia, se observan saltos periódicos de medición a 31 $[cm]$, correspondientes a momentos en que el móvil seguidor deja de detectar al móvil líder al

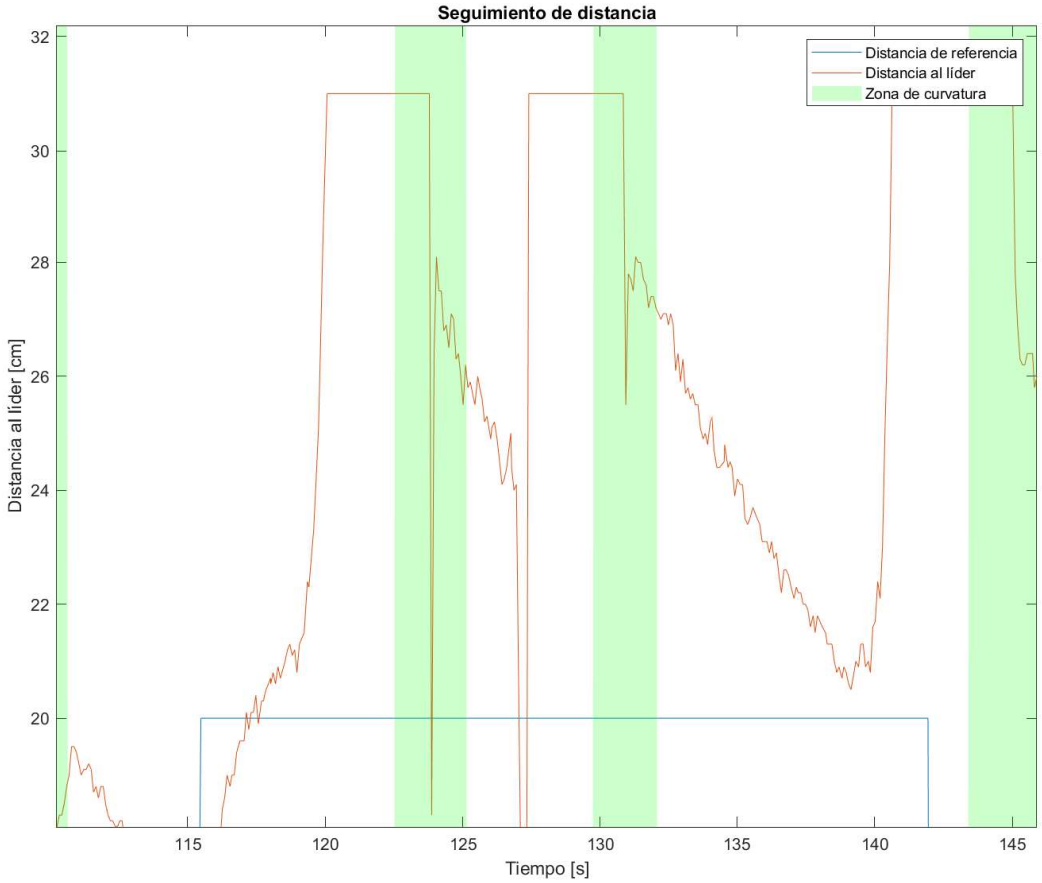


Figura 5.7: Gráfica del seguimiento de distancia al móvil Líder, con acercamiento.

estar en una curva. Los segmentos de color verde indican los momentos en que el móvil seguidor detecta que está en una curva.

En la Figura 5.7 se realiza un acercamiento a un tramo del experimento donde se tiene una distancia de referencia de 20 [cm]. En esta, se observa que al aumentan la distancia de referencia, el móvil intenta seguirla, pero comienza a medir el máximo debido a que el móvil líder entró en la curva, para luego el móvil seguidor también comience a detectar la zona de curva. Antes de salir de la curvatura, vuelve a detectar al líder por lo que aumenta de velocidad para reducir la distancia medida y acercarse a la referencia, pero antes que pueda lograrlo, entra nuevamente a una zona de curvatura.

Por otra parte, se registra la velocidad de referencia que posee el móvil seguidor, la cual corresponde a la salida del controlador PID de distancia, y entrada del PID de velocidad. Además, se registra el modo de control en que se encuentra el controlador PID de distancia, manual o automático. La lógica que se utiliza aquí es la presente en el Algoritmo 4.3.

En la Figura 5.7 se puede observar un acercamiento a un tramo del experimento en el cual se establece una distancia de referencia de 20 [cm]. En esta imagen, se aprecia cómo el móvil intenta seguir la distancia de referencia al aumentar la misma, pero comienza a medir la distancia máxima debido a que el móvil líder entra en una zona de curva, lo que provoca que el móvil seguidor también detecte la curvatura. Después de salir de la curvatura, el móvil seguidor logra detectar al líder nuevamente y, para acercarse a la

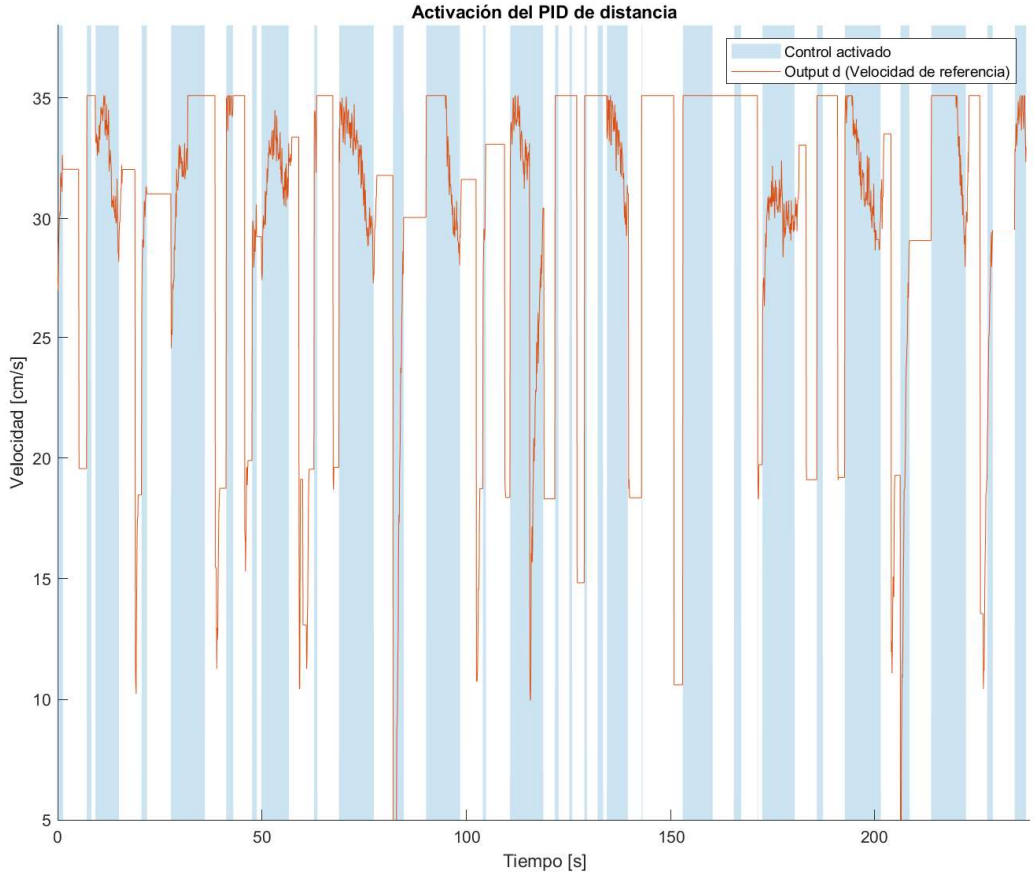


Figura 5.8: Gráfica de la velocidad de referencia del móvil seguidor.

distancia de referencia, aumenta su velocidad, pero antes de poder alcanzar la distancia deseada, entra nuevamente en una zona de curvatura.

Además, se registran la velocidad de referencia del móvil seguidor, que corresponde a la salida del controlador PID de distancia y la entrada del controlador PID de velocidad, así como también el modo de control en el que se encuentra el controlador PID de distancia, que puede ser manual o automático. La lógica empleada se encuentra en el Algoritmo 4.3.

El resultado del experimento se puede observar en el video Experimento 2 moviles - Seguimiento de distancia de referencia.

De forma análoga, se realiza nuevamente el experimento, pero esta vez utilizando el limitador de salida del controlador PID de Distancia (descrito en el Algoritmo 4.5) donde se obtienen los resultados de las Figuras 5.12, 5.10 y 5.11

El nuevo experimento permite observar que el controlador de distancia tiene una limitación del $+/-10\,\%$ respecto a la velocidad de referencia del móvil predecesor. Al comparar los rangos de actuación de la Figura 5.8 con la Figura 5.10, se nota una evidente reducción de saturación. Para este caso, la velocidad de referencia del móvil predecesor es de 30 $[cm/s]$, lo que implica que la saturación máxima se fija en $((30+1)*1.1+1) = 35.1$[cm/s] y la saturación mínima en $((30+1)*0.9-1) = 26.9$[cm/s]. Este cambio tiene un impacto positivo en el rendimiento del controlador de distancia, como se puede observar en la Figura 5.11, en comparación con la Figura 5.7.

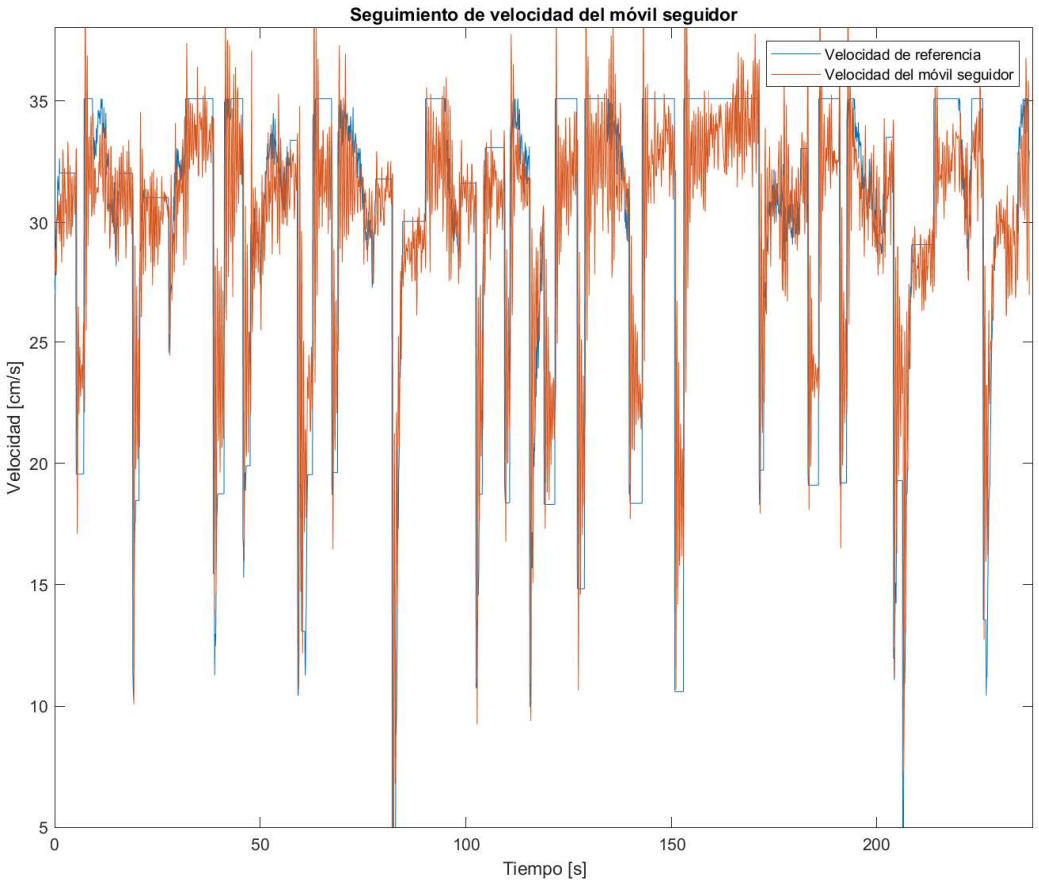


Figura 5.9: Gráfica del seguimiento de velocidad del móvil seguidor.

En la Figura 5.12 se puede apreciar que el móvil seguidor sigue la referencia generada por el controlador de distancia con una desviación de $+/-2$ $[cm/s]$ (7 % de error app.), lo cual es aceptable para esta aplicación. El video de este experimento está disponible en Experimento 2 moviles - Seguimiento de distancia de referencia con saturación.

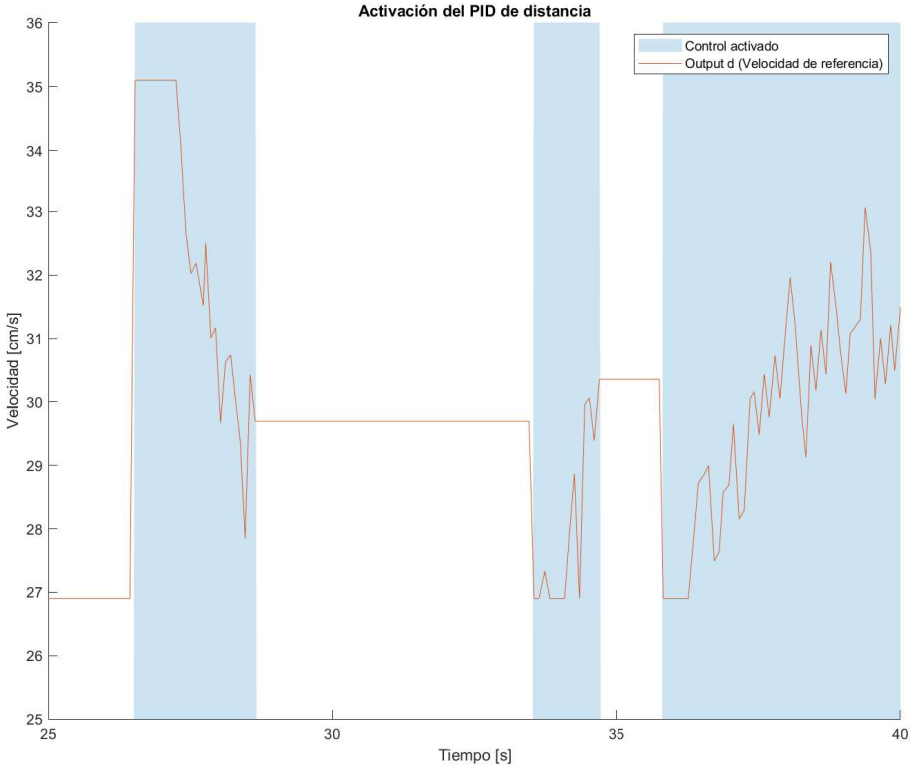


Figura 5.10: Gráfica de la activación de control de distancia del móvil seguidor.

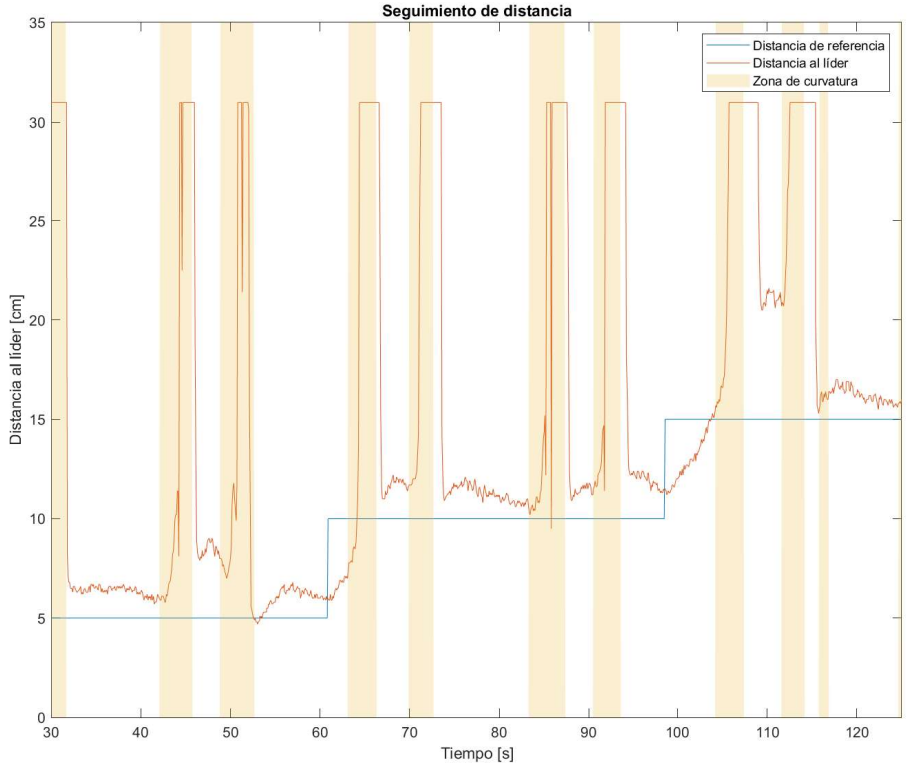


Figura 5.11: Gráfica del seguimiento de distancia de referencia del móvil seguidor.

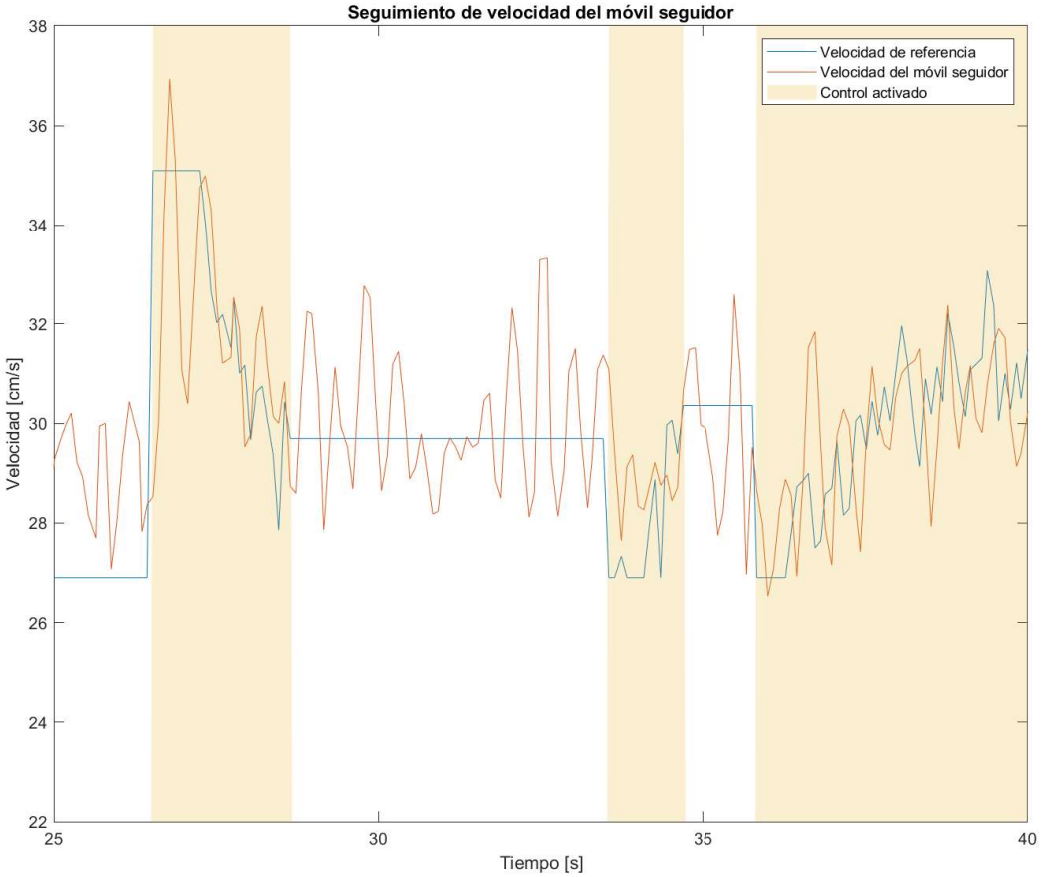
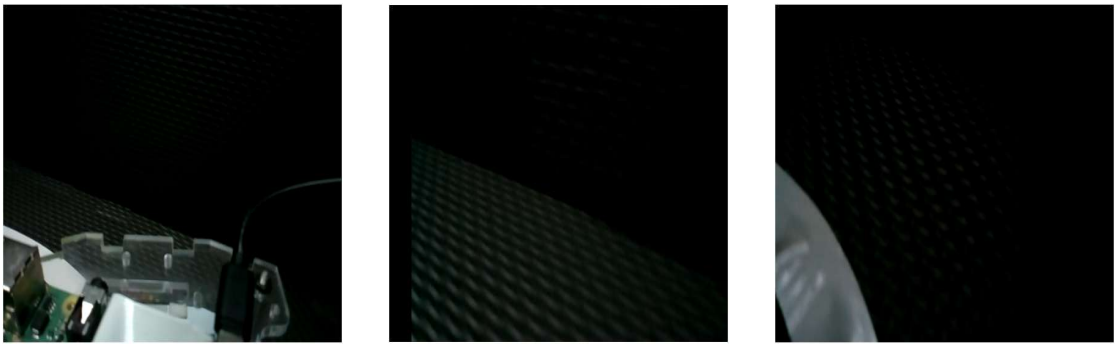


Figura 5.12: Gráfica del seguimiento de velocidad de referencia del móvil seguidor en conjunto a la activación del controlador de distancia.



(a) Posición: Altura (b) Posición: Frontal abajo (c) Posición: Posterior abajo

Figura 5.13: Prueba de captura de video desde distintas posiciones. Captura del horizonte de la trayectoria.



(a) Posición: Altura (b) Posición: Frontal abajo (c) Posición: Posterior abajo

Figura 5.14: Prueba de captura de video desde distintas posiciones. Captura de la trayectoria en curva.

5.2. Implementación de Computer Vision

Con el fin de integrar la visión por computador en la plataforma, es necesario definir la posición de la cámara dentro de la misma. Para lograr esto, se llevan a cabo diversas pruebas en las que se varía tanto la posición como el ángulo de visión de la cámara en relación con la trayectoria que debe seguir el móvil.

En las Figuras 5.13 y 5.14 se muestran capturas de video del móvil mientras sigue su trayectoria a velocidad constante. Se realizaron pruebas con la cámara ubicada en tres posiciones diferentes: en altura (aproximadamente 5 [cm] por encima de la plataforma), frontal abajo (ubicada en el segundo nivel de la plataforma, a unos 5 [cm] del suelo) y posterior abajo (a la misma altura que la posición frontal, pero mirando hacia la trayectoria ya recorrida). En la Figura 5.13 se puede apreciar la calidad de captura de la trayectoria en los tres escenarios. Se observa que la cámara en altura logra captar más elementos de la trayectoria, pero también puede confundirse con objetos que no forman parte de la trayectoria. Por lo tanto, se considera que la posición frontal abajo es más adecuada en este caso. En la Figura 5.14, se muestran capturas de video durante la realización de curvas. En este caso, la posición posterior abajo logra captar la trayectoria en más tramos que las otras dos posiciones. Tras la prueba, se decidió utilizar la posición

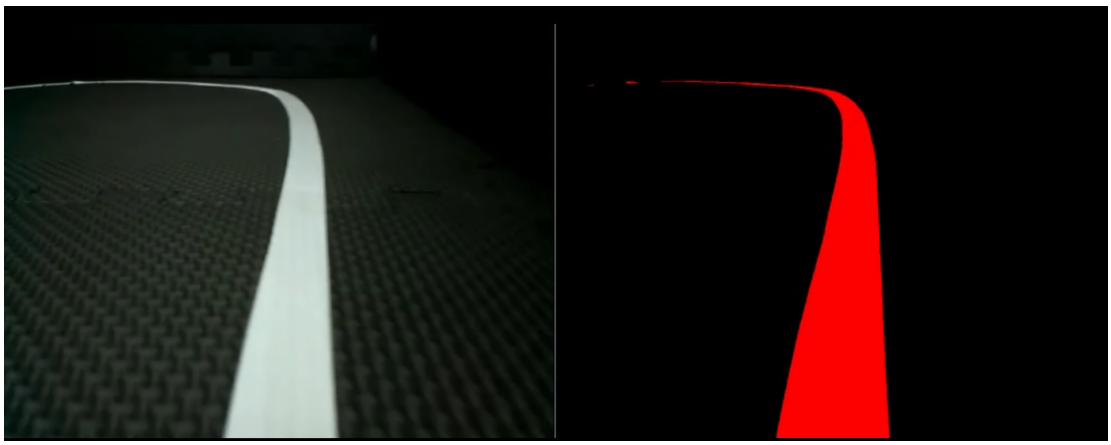


Figura 5.15: Captura de un video proveniente de la cámara (izquierda) en comparación con el video segmentado utilizando *OpenCV* (derecha).

frontal abajo para la cámara, con la opción de ajustar el ángulo de captura según sea necesario. En caso de ser necesario, se puede utilizar la vista frontal en conjunto con la posterior. El resultado de la prueba de posición de la cámara se puede observar en el video Prueba posición de la cámara.

Después de determinar la posición de la cámara, se procede a implementar algoritmos de visión por computadora en la Raspberry Pi utilizando la biblioteca de código abierto OpenCV junto con Numpy.

En primer lugar, se graba un video de aproximadamente 30 segundos del móvil siguiendo la trayectoria predefinida, colocando la cámara en la parte frontal del móvil y luego, en una segunda instancia, en la parte posterior.

A continuación, se procesa el video. Primero se lee el video en *OpenCV* y se convierte en un vector de matriz de imágenes.

Luego, se escala el vector a una imagen en escala de grises para reducir el número de canales de color y hacerlo compatible con las funciones restantes de la biblioteca.

Para cada *frame* se aplica una secuencia de funciones. Se evalúa cada píxel según un nivel fijo o *threshold* y se le asigna un valor máximo o mínimo según su comparación con dicho nivel. Se utiliza un *threshold* adaptativo que varía el nivel en función de las lecturas anteriores. Este es la suma ponderada de los valores de peso de los píxeles vecinos, cuyos pesos son asignados de forma Gaussiana. La salida de esta función es binaria, obteniendo 0 o 255.

Luego, se aplica un efecto de desenfoque o *blur* en cada *frame* para que el algoritmo se centre en lo general en lugar de en los detalles de cada captura de video.

A continuación, se vuelve a aplicar la función de *threshold* en cada *frame*, esta vez no adaptativa. Todos los píxeles que sean mayores que el nivel definido de *threshold* se fijan en 255. Esto permite separar la trayectoria, definida por una línea blanca sobre un fondo negro o una línea negra en un fondo blanco, del resto de elementos del video.

La diferencia de colores entre la línea de la trayectoria y los demás elementos es fundamental para una nueva segmentación del video.

Finalmente, se concatenan todos los *frames* procesados para obtener un video del recorrido del móvil de dos colores, donde la trayectoria se destaca en un fondo unicolor. Este resultado se puede observar en la Figura 5.15.

Teniendo para las pruebas una superficie negra con una línea de trayectoria blanca, se obtiene lo expuesto a la izquierda de la Figura 5.15.

En el lado derecho de la Figura 5.15 se aprecia el resultado del algoritmo de segmentación, donde en este caso se aplica una máscara de color rojo para resaltar la trayectoria predefinida.

Cabe destacar que el procesamiento de video se realiza una vez capturado el mismo, los que suelen durar aproximadamente 30 segundos. También, se implementa una interfaz al momento de ejecutar el algoritmo que permite especificar el nombre del video a procesar y el video de resultado. Dicho Código 8.1 se encuentra en la sección Anexo.

5.2.1. Limitantes observadas para el funcionamiento del segmentador

Se realizan diversas pruebas al segmentador de trayectoria para comprobar su funcionamiento durante la operación normal del móvil. En primera instancia se realizan pruebas en un ambiente controlado, es decir, homogeneizando la sección de escenario que es captada por la cámara por medio de paredes del mismo material que el suelo, además de quitando cualquier otro objeto ajeno a la trayectoria.

Se obtiene que el segmentador reconoce la trayectoria mientras se tiene visión de esta, en cambio, en zonas de curvatura la trayectoria queda fuera del ángulo de visión de la cámara, lo que provoca que el segmentador identifique como trayectoria otras partes del escenario. El resultado de esta prueba se puede observar en la Figura 5.15, y la prueba completa se puede encontrar en el video Prueba del segmentador de trayectoria.

Además, se realizan pruebas en la operación normal del móvil bajo un ambiente semi controlado, es decir, sin murallas en el escenario y en algunos casos, con objetos ajenos a la trayectoria dentro del ángulo de visión de la cámara.

A continuación se mencionan distintas situaciones donde se observa que el funcionamiento del segmentador implementado resulta insuficiente para detectar solo la trayectoria.

Variación en la condición lumínica de la captura de video.

En la Figura 5.16 se observa un caso en el que el segmentador reconoce partes de la superficie negra como trayectoria predefinida. Esto sucede debido a la baja calidad de iluminación que existía al momento de grabar el video. La actualización de la plataforma considera una fuente lumínica (LED) para mantener estables las condiciones lumínicas durante el recorrido del móvil.

Recorrido no acotado por murallas.

En la Figura 5.17 se observa que el segmentador reconoce como trayectoria parte del suelo, fuera de la superficie negra habilitada para el recorrido, debido a que es de color claro similar a la trayectoria definida.

Una solución a este problema es acotar la visión de la cámara para evitar elementos distractivos, cercando la superficie a recorrer con murallas de color contrario al color de la línea de trayectoria. En este caso, en pruebas se utilizan murallas del mismo material que la superficie.

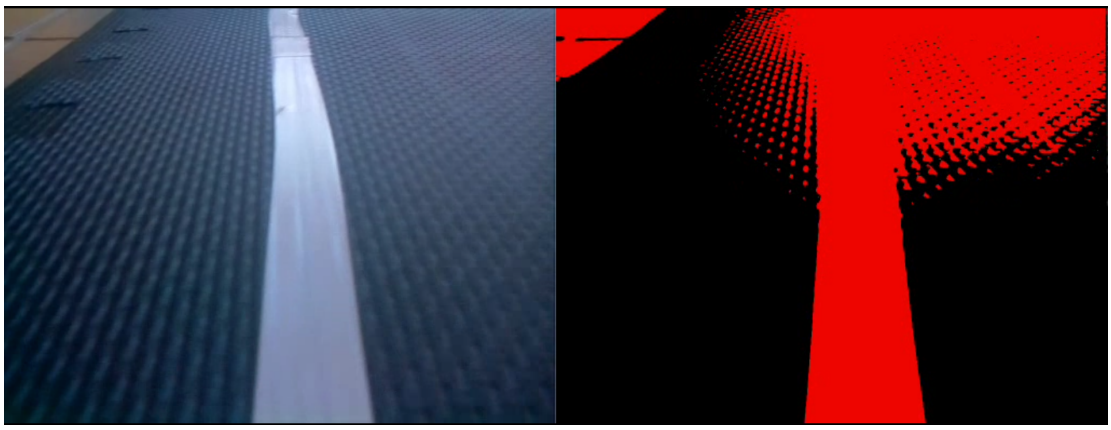


Figura 5.16: Captura de un video proveniente de la cámara (izquierda) en comparación con el video segmentado utilizando *OpenCV* (derecha), en situación de poco óptima iluminación.

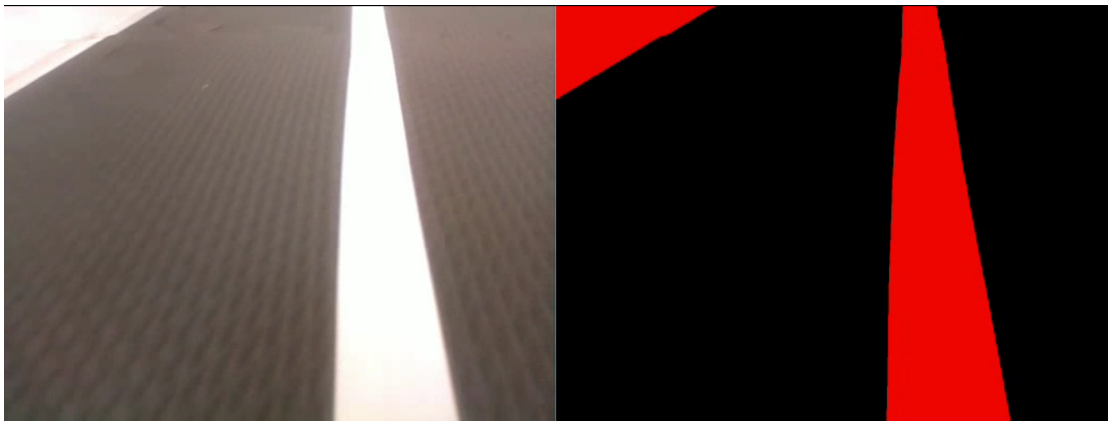


Figura 5.17: Captura de un video proveniente de la cámara (izquierda) en comparación con el video segmentado utilizando *OpenCV* (derecha), en situación de confusión por falta de murallas.

Falta de captura de curvas de la trayectoria

En la Figura 5.18 se observa que la cámara no logra captar la línea correspondiente a la trayectoria predefinida, provocando que el segmentador deje de funcionar.

Para solucionar este caso, se pueden seguir distintos acercamientos. Se puede reemplazar la cámara por una que posea mayor rango de visión y con ello asegurar que la línea sea captada en todo momento, probar distintas posiciones de la cámara en la estructura del móvil, o incluso utilizar más de una cámara (por ejemplo una cámara frontal y otra posterior) para que en conjunto, puedan brindar una visión panorámica del recorrido.

Presencia de objetos cercanos a la trayectoria

En la Figura 5.19 se observa que existe un objeto cercano a la trayectoria predefinida, el cual provoca que el segmentador lo considere dentro de la capa para definir la línea.

Para solucionar esta problemática, se puede utilizar el mismo enfoque de casos ante-

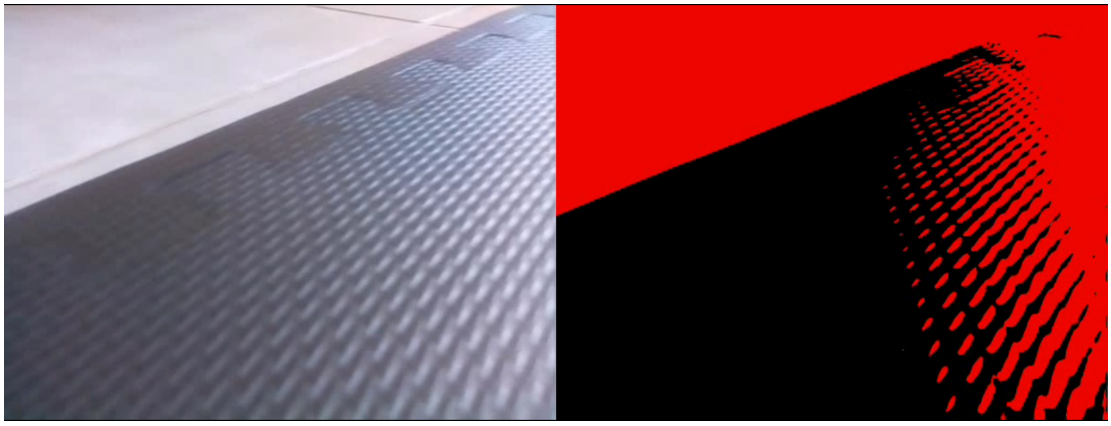


Figura 5.18: Captura de un video proveniente de la cámara (izquierda) en comparación con el video segmentado utilizando *OpenCV* (derecha), en situación de ausencia de captura de trayectoria en el recorrido.

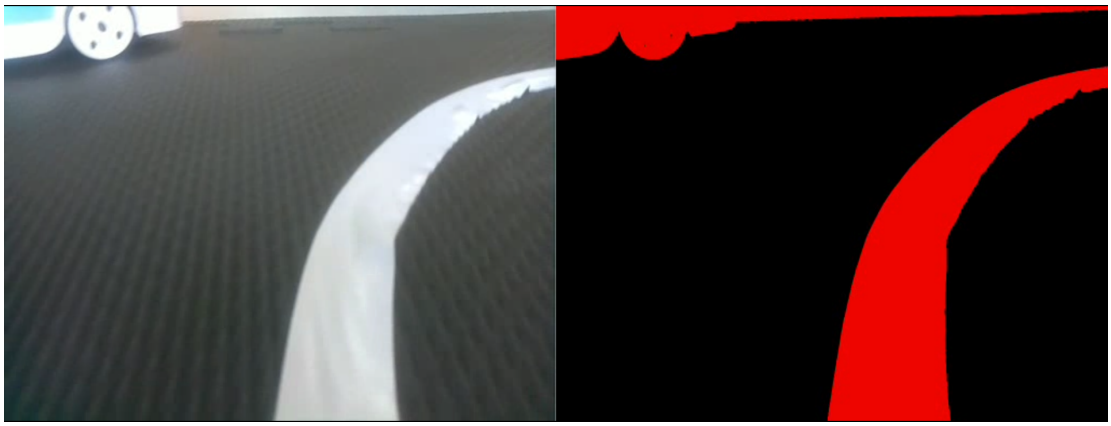


Figura 5.19: Captura de un video proveniente de la cámara (izquierda) en comparación con el video segmentado utilizando *OpenCV* (derecha), en situación de objetos ajenos al recorrido.

riores encerrando la superficie que contiene la trayectoria con murallas de color similar al suelo y removiendo cualquier objeto ajeno al recorrido. Pero en caso de que se estén realizando pruebas con más de un agente, se vuelve infactible esta solución.

Entonces, es necesario explorar distintos algoritmos de segmentación del video, para que sea capaz de ignorar objetos extraños y enfocarse en identificar la línea correspondiente a la trayectoria predefinida.

5.2.2. Evaluación de la factibilidad de segmentación en tiempo real

En el repositorio [25] de Xiaotian Dai se recopila un proyecto con enfoque a detección y seguimiento de objetos utilizando la Raspberry Pi en tiempo real.

Para esto, utiliza la Raspberry Pi como unidad de control central y una cámara USB como hardware. Por otra parte, se emplea la misma biblioteca utilizada en esta sección, $OpenCV - Python$.

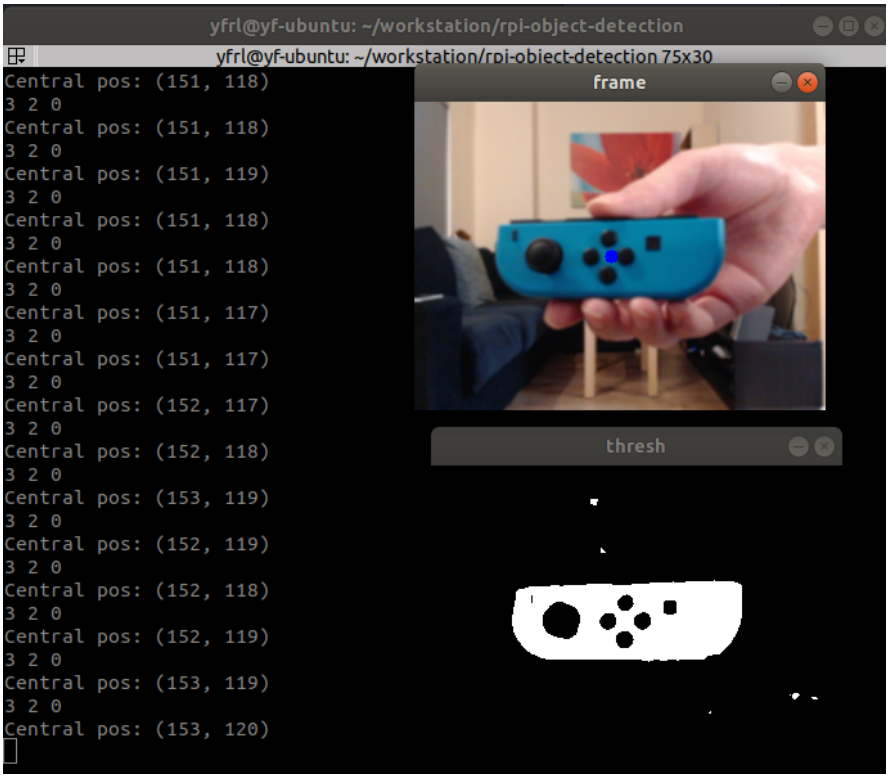


Figura 5.20: Captura de la demostración desarrollada por Xiaotian para seguimiento de un objeto de color específico [25].

En este, Xiaotian desarrolla un detector de movimiento, comparando 2 *frames* consecutivos de un flujo de video entrante, y al detectar cambios levanta una alerta.

También desarrolla un detector y seguidor de objetos de un color específico, declarado al iniciar el programa. En la Figura 5.20 se observa una captura de la demostración desarrollada en el proyecto. En esta, primero se declara el color que el segmentador buscará dentro del flujo de video.

Luego, al determinar la posición del objeto, muestra un flujo de video segmentado utilizando una máscara binaria de colores blanco y negro. Junto con esto, arroja la posición del centroide del objeto detectado, en coordenadas con referencia al video. Así, una vez identificado el objeto, es capaz de seguirlo mientras se encuentre dentro del ángulo de visión de la cámara, reportando su ubicación.

Considerando este proyecto, se puede concluir que sí es posible realizar segmentación de video en base a máscaras binarias utilizando *OpenCV*, en una Raspberry Pi en tiempo real, pero es necesario saber la cantidad de $frames$ por segundo que es capaz de procesar, buscando obtener un video fluido y estable que sea de utilidad para determinar la trayectoria a seguir por el móvil.

Por otra parte, cabe destacar que el presente proyecto además utiliza la Raspberry Pi para accionar motores, adquirir datos de distintos sensores y establecer comunicación inalámbrica, por lo que también es necesario estudiar la factibilidad de ejecutar ambos procesos de forma simultánea.

Capítulo 6

Conclusiones

En base a lo expuesto durante este trabajo de título en conjunto con los resultados obtenidos de los diversos experimentos realizados, se concluye que es factible realizar una nueva versión de la plataforma RUPU [9] mediante el reemplazo de su unidad de control central, asegurando que se mantengan las funcionalidades que la plataforma base posee, y además incorporando captura de video y post procesamiento del mismo.

La actualización de la plataforma a través de la adopción de una nueva unidad de control central permite aperturar las posibilidades y potencial que la plataforma RUPU [9] en su versión original posee. Posibilita, en el futuro, abandonar la limitante de seguir una trayectoria predefinida mediante la modificación de la superficie por la cual el móvil transite, si no que logra incorporar una cámara que realizaría reconocimiento de imágenes y dictamine el trayecto a seguir. También, gracias al aumento de la capacidad de cómputo que posee la nueva unidad de control central, logra ejecutar distintos programas en simultaneo debido a su capacidad de realizar procesos *multi-threading* mejorando su rendimiento.

Sin embargo, es necesario realizar diversos cambios específicos para cumplir con los requerimientos de la Raspberry Pi. Se debe considerar la característica digital estricta que poseen sus GPIO por lo que se adicionan módulos convertidores de señales análogas a digital (ADC). Además, trae consigo un requerimiento energético de mayor exigencia en comparación con la unidad de control central antigua, ESP32, por lo que se debe adicionar una fuente de voltaje externa, o bien un regulador de voltaje que suministre 5 $[V]$, voltaje de entrada a la Raspberry Pi.

Además, se observó que es necesario realizar distintas pruebas para posicionar la cámara digital dentro de la plataforma, evaluando ángulos de inclinación para asegurar la visión de la trayectoria durante todo el recorrido del móvil y así lograr que el segmentador implementado detecte el todo momento la trayectoria.

Si bien en este trabajo de título se logra la incorporación y adaptación de la Raspberry Pi para que sea compatible con la plataforma base, quedan aún puntos de mejora a trabajar a futuro. Estos son los siguientes.

- Incorporar un sistema de control de velocidad basado en la transmisión de video

de la cámara incluida en la actualización de la plataforma.

- Reemplazar la PBC original de la plataforma base para integrar directamente la Raspberry Pi, incluso incorporando el regulador de voltaje adicional necesario para la alimentación.
- Investigar diferentes algoritmos de control de velocidad, como el aprendizaje automático.

Capítulo 7

Referencias

[1] *9 DOF IMU breackout*. 2021. URL: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-9-dof-imu-breakout.pdf.

[2] *ADS1115 16 bits ADC*. 2018. URL: https://www.ti.com/lit/ds/symlink/ads1115.pdf?ts=1661637464915.

[3] Michael Bonani y col. «The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research». En: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, págs. 4187-4193. DOI: 10.1109/IROS.2010.5649153.

[4] Castillo. *GitHub - pcastillodp/elo308: Documentos y codigos usados para trabajo de titulo*. 2022. URL: https://github.com/pcastillodp/elo308.

[5] *CD75HC4067 multiplexor 16CH*. 2003. URL: https://www.ti.com/lit/ds/symlink/cd74hc4067.pdf.

[6] SAE On-road Automated Vehicles Standards Committee y col. *SAE J3016: Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems*. 2018.

[7] *DuckieTown*. 2018. URL: https://www.duckietown.org.

[8] Yuta Esaka y col. «Platooning method for multiple mobile robots using laser-based SLAM». En: *SICE Annual Conference 2011*. 2011, págs. 251-254. URL: https://ieeexplore-ieee-org.usm.idm.oclc.org/document/6060613.

[9] Carlos Escobar. «Diseño, implementación y evaluación de una plataforma experimental para el estudio del control de una flota de robots móviles».

[10] Chunhua Hong y col. «A Joint Design of Platoon Communication and Control Based on LTE-V2V». En: *IEEE Transactions on Vehicular Technology* 69.12 (2020), págs. 15893-15907. DOI: 10.1109/TVT.2020.3037239.

[11] Patrick Hong y col. *Drag Forces Experienced By Two, Full-scale Vehicles At Close Spacing*. Institute of Transportation Studies at UC Berkeley, 1998. URL: https://escholarship.org/uc/item/50q2p3sn.

[12] *Lithium-ion battery 18650*. 2010. URL: https://www.ineltro.ch/media/downloads/SAAItem/45/45958/36e3e7f3-2049-4adb-a2a7-79c654d92915.pdf.

[13] *LM2596 SIMPLE SWITCHER Power Converter 3-A Step-Down Voltage Regulator*. 1999. URL: https://www.ti.com/lit/ds/symlink/lm2596.pdf.

[14] Raspberry Pi Ltd. *Raspberry Camera Module 2*. 2016. URL: https://www.raspberrypi.com/products/camera-module-v2/.

[15] Raspberry Pi Ltd. *Raspberry Pi 4 Model B*. 2019. URL: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.

[16] Matteo Muratori, Jacob Holden y Michael Lammert. *Potentials for Platooning in U.S. Highway Freight Transport*. SAE World Congress Experience, 2017. URL: https://www.nrel.gov/docs/fy17osti/67618.pdf.

[17] Liam Paull y col. «Duckietown: An open, inexpensive and flexible platform for autonomy education and research». En: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, págs. 1497-1504. DOI: 10.1109/ICRA.2017.7989179.

[18] Andrés A. Peters, Francisco J. Vargas y Garrido. «PL-TOON: A Low-Cost Experimental Platform for Teaching and Research on Decentralized Cooperative Control». En: *Sensors* 21.6 (2021). ISSN: 1424-8220. DOI: 10.3390/s21062072. URL: https://www.mdpi.com/1424-8220/21/6/2072.

[19] *Pololu - QTR-8RC Reflectance Sensor Array*. 2016. URL: https://www.pololu.com/product/961.

[20] RIC MOTOR COMPANY LIMITED. *N20 micro gear motor | 12mm dc gear motor for model train - RIC MOTOR*. Dic. de 2020. URL: https://www.ricmotor.com/details/12mm-dc-gear-motor.

[21] *SparkFun Motor Driver - Dual TB6612FNG (1A)*. Dic. de 2017. URL: https://www.sparkfun.com/products/14451.

[22] *VL53L0X*. 2016. URL: https://www.st.com/resource/en/datasheet/vl53l0x.pdf.

[23] *Wemos® D1 R32 w/ ESP32 UNO R3*. 2016. URL: https://www.botnroll.com/en/esp/3639-wemos-d1-r32-w-esp32-uno-r3-pinout.html.

[24] Shixi Wen y Ge Guo. «Control of Leader-Following Vehicle Platoons With Varied Communication Range». En: *IEEE Transactions on Intelligent Vehicles* 5.2 (2020), págs. 240-250. DOI: 10.1109/TIV.2019.2955899.

[25] Xiaotian Dai. *GitHub - automaticdai/rpi-object-detection: Real-time object detection and tracking using Raspberry Pi and OpenCV!* Inglés americano. 2021. URL: https://github.com/automaticdai/rpi-object-detection (visitado 05-10-2022).

[26] Michael Zabat, Nick Stabile y Stefano Farascaroli. *The Aerodynamic Performance Of Platoons: A Final Report*. California Partners for Advanced Transportation Technology, 1995. URL: https://escholarship.org/uc/item/8ph187fw.

Capítulo 8

Anexos

8.1. Repositorio

Documentación, códigos y videos de pruebas desarrollados para el presente trabajo de título se pueden encontrar en el repositorio de GitHub [4].

8.2. Aspectos de interés para la implementación de sistemas multiagentes en plataformas experimentales.

A continuación se destacan distintos aspectos de interés que permiten el correcto funcionamiento de sistemas multiagentes compuestos por móviles con tracción diferencial, es específico para formación de *platooning* o pelotón.

El primero, es que se debe mantener una distancia predeterminada entre los móviles diferenciales, la que puede ir variando de agente a agente y en el tiempo. Entonces, es necesario que cada móvil sea capaz de alcanzar y mantener dicha distancia, tema que se denomina **control de distanciamiento**.

Otro tema, se desprende del hecho que puede que alguno de los móviles diferenciales sean tripulados, mientras que otros sean autónomos. Esto arroja una gran incógnita: ¿Cómo mantienen la formación predeterminada los agentes? Este tema se denomina **control de seguimiento**.

De este último, surge la posibilidad de intercambiar información entre la flota, la cual podría ser la velocidad, la aceleración, la ruta a seguir e incluso la geolocalización, lo que se denomina **comunicación entre la flota**.

Y se llega al último tema que se refiere a la **capacidad de cómputo, actuadores y sensores** que deben tener los móviles diferenciales de la flota para abordar los temas presentados anteriormente, por ejemplo, el tipo de microcontrolador que utiliza, la cantidad de estos, el tipo, entre otras variables.

Para aproximarse a desarrollos que permitan estudiar a los sistemas multiagentes, primero se describe en detalle cada tema anteriormente expuesto, además brindándole

una aproximación general de cómo se ha implementado en las distintas plataformas estudiadas.

8.2.1. Control de distanciamiento

Refiere a la capacidad de mantener constante la distancia que existe entre dos o más móviles diferenciales pertenecientes a la plataforma. Este parámetro puede ser establecido al comienzo del viaje, y puede o no variar según sea necesario.

Existen distintas formas de medir esta distancia: utilizando un sensor incorporado en los móviles que mida de forma física la distancia, tales como sensores ultrasónicos, sensores láser o infrarrojo; por medio de la publicación de geolocalización de cada agentes; por medio de sensores externos a los móviles (por ejemplo cámaras en la autopista) que reportan a una unidad central que a su vez reporta de vuelta la distancia entre cada móvil.

8.2.2. Control de seguimiento

Refiere a la capacidad del móvil diferencial para seguir una trayectoria predeterminada, acorde a su posición dentro de la flota.

En el caso del *platooning*, este tema queda implementado mediante limitaciones en el camino (por ejemplo una línea dibujada en el suelo, rieles [18] donde los móviles no tienen libertad de recorrido, o mantenerse dentro de las líneas delimitadoras de un carril, asemejándose a los dictámenes de las leyes del tránsito para automóviles).

Un caso más realista es cuando el seguimiento se realiza a través de una cámara frontal que posee cada agente. Aquí, el móvil líder es tripulado y dictamina la trayectoria a seguir, mientras que los móviles sucesores, de forma autónoma y utilizando reconocimiento de imágenes, identifica al líder para iniciar un control de seguimiento al mismo.

Similar a esto, se tiene la tecnología SLAM [8] (Localización y modelado simultáneo), donde el líder transmite a los seguidores imágenes escaneadas a través de un láser tipo LIDAR e información odométrica. Luego, los seguidores construyen un mapa de ocupación basado en la información del líder, y estima la posición del líder usando una correlación con el mapa que posee.

El seguir al móvil de adelante puede ser resuelto mediante una gran cantidad de sensores y tecnologías, como por ejemplo IR, láser, ultrasónico, procesamiento digital de imágenes, entre otros.

8.2.3. Comunicación entre la flota

Refiere al protocolo y a los datos que se pueden enviar entre los miembros de la flota, para lograr un mejor seguimiento de cada posición predeterminada. Un ejemplo de esto es comunicar variables propias de cada móvil, como podría ser velocidad, posición relativa o absoluta, o la aceleración.

La comunicación en todos los casos es inalámbrica, pero difiere de los protocolos utilizados. Por ejemplo, se utiliza un protocolo de LTE vehículo-a-vehículo (LTE-V2V) [10], donde se disemina la información del líder, y luego se disemina la información de los seguidores.

Los esquemas también utilizados [24] son: líder-seguidor (líder transmite sus datos a cada uno de los seguidores); predecesor-seguidor (líder transmite al seguidor próximo, y este trasmite su información al siguiente de forma sucesiva); y finalmente de forma bidireccional.

8.2.4. Actuadores, sensores y capacidad de cómputo

Refiere a todos aquellos componentes que permiten interactuar con el medio, como son los motores del móvil para el caso de los actuadores, sensores de distancia, de reconocimiento de trayectoria o de imágenes, de geolocalización, entre otros, como componentes adquisidores de datos.

Dependiendo de la aproximación elegida, se estudia la complejidad, costo y cantidad de los sensores a utilizar. Esto lleva directamente al estudio de la capacidad de cómputo disponible que posee la unidad de control central, para interpretar los sensores y operar en los actuadores.

8.3. Fotografías del proyecto

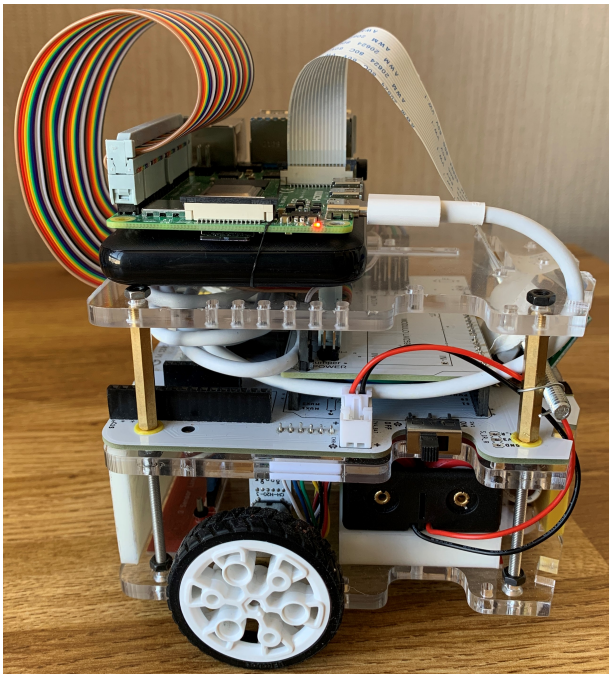


Figura 8.1: Fotografía del móvil con tracción diferencial, vista lateral derecha.

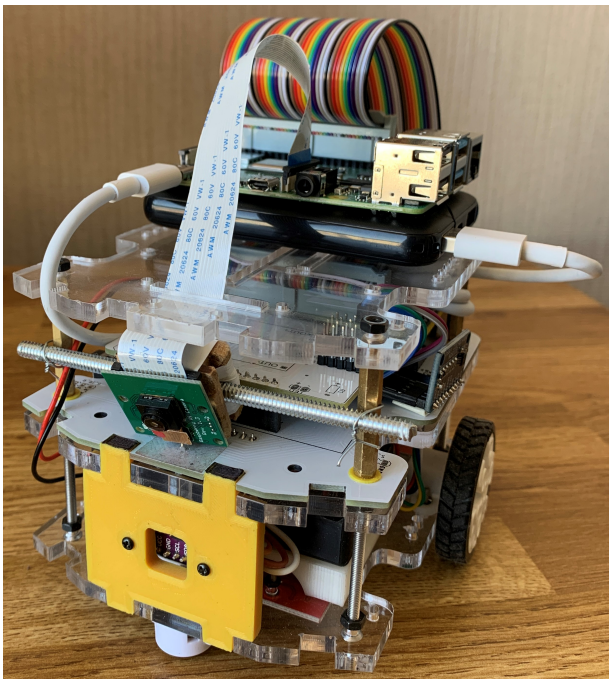


Figura 8.2: Fotografía del móvil con tracción diferencial, vista ortogonal.

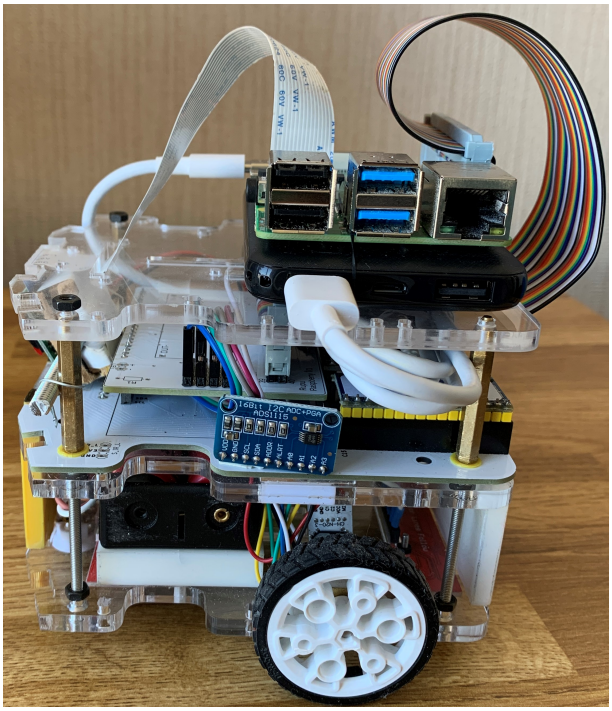


Figura 8.3: Fotografía del móvil con tracción diferencial, vista lateral izquierda.

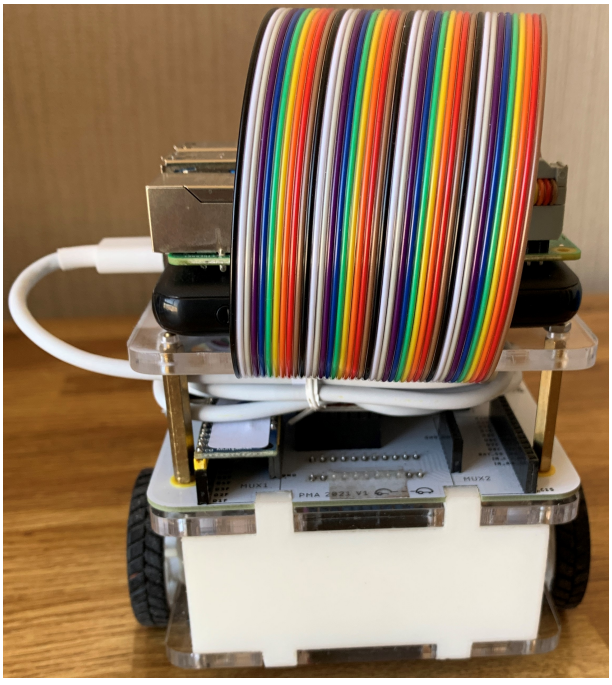


Figura 8.4: Fotografía del móvil con tracción diferencial, vista posterior.

8.4. Códigos

A continuación se presentan ciertos códigos los cuales son comentados y referidos dentro del trabajo de título.

```
import cv2 as cv
import numpy as np
import time
import argparse

parser = argparse.ArgumentParser()
parser.add_argument("-i", "--input", help="video de entrada como video.mp4")
parser.add_argument("-o", "--output", help="video de salida como video_seg.mp4")
parser.add_argument("-c", "--camera", help="ID de camara como 0")
args = parser.parse_args()
video_input = cv.VideoCapture()
output_name = ""

if((args.camera is None) and (args.input is None)):
    print("No se ha indicado un id de camara ni un video de entrada.\nPor favor vuelva a intentarlo indicando la opcion -i, -c o -h para ayuda")
    video_input.release()
    exit(0)
elif((args.camera is not None) and (args.input is not None)):
    print("Solo debe indicarse uno de estos argumentos: -i o -c, no ambos al mismo tiempo.\nPor favor reintentar ...")
    video_input.release()
    exit(0)
elif(args.camera is not None):
    print("El id de la camara es: ", args.camera, ". Procesando flujo de video entrante en tiempo real.\nPulsa la tecla 'Q' para terminar procesamiento ...")
elif(args.input is not None):
    print("Procesando video: ", args.input, "\nPulsa la tecla 'Q' para terminar procesamiento ...")
    video_input.release()
    video_input = cv.VideoCapture(str(args.input))
else:
    print("algo malo ocurrio ... ")
    exit(-1)
if(args.output is None):
    output_name = str(args.input).split(".")[0] + "_seg.mp4"
    print("No se ha indicado nombre para el video de salida, asi que este sera: ", output_name)
else:
    output_name = str(args.output)
    print("El output indicado es: ", output_name)

print("iniciando analisis ...")
ret, frame = video_input.read()
frame_gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
zeros = np.zeros(frame_gray.shape, dtype='uint8')

video_seg_out = cv.VideoWriter(output_name,cv.VideoWriter_fourcc(*'MP4V'), 25, (frame_gray.shape[:2][1],int(frame_gray.shape[:2][0]/2)))
```

```
44
45 time_start = int(time.time())
46 last_time = time_start
47
48 while video_input.isOpened():
49     ret, frame = video_input.read()
50     if time.time() - last_time > 1:
51         last_time = time.time()
52         print(last_time - time_start)
53     if ret == True:
54         frame_gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
55         th1 = cv.adaptiveThreshold(frame_gray, 255, cv.
    ADAPTIVE_THRESH_GAUSSIAN_C, cv.THRESH_BINARY, 11, 2)
56         blur = cv.GaussianBlur(frame_gray,(5,5),0)
57         ret2,th2 = cv.threshold(blur,0,255,cv.THRESH_BINARY+cv.
    THRESH_OTSU)
58         th3 = cv.merge([zeros,zeros,th2])
59
60         small_frame = cv.resize(frame, (0, 0), None, .5, .5)
61         small_th3 = cv.resize(th3, (0, 0), None, .5, .5)
62
63         concat_frame = np.hstack((small_frame,small_th3))
64
65         video_seg_out.write(concat_frame)
66
67         cv.imshow("segmentation", concat_frame)
68
69         if cv.waitKey(5) & 0xFF == ord('q'):
70             break
71     else:
72         break
73 print("Procesamiento terminado")
74 video_input.release()
75 video_seg_out.release()
76 print("video guardado como: ", output_name)
77 cv.destroyAllWindows()
```

Código 8.1: Código para implementación de segmentación de video utilizando *Computer Vision* en Python