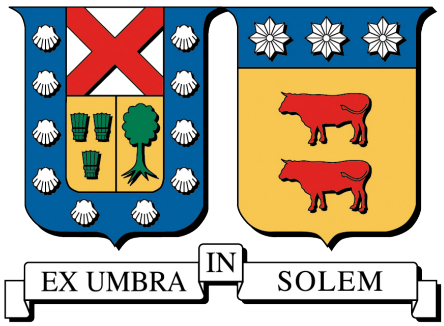
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAISO - CHILE



DISEÑO E IMPLEMENTACIÓN DE INTERFAZ DE USUARIO Y CASOS DE USO PARA PLATAFORMA EXPERIMENTAL DE SISTEMA MULTI-AGENTES

CATALINA CHAUFLEUR CONCHA

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERA CIVIL ELECTRÓNICA

PROFESOR GUÍA	:	GONZALO CARVAJAL B.
PROFESOR CORREFERENTE	:	FRANCISCO VARGAS P.

OCTUBRE 2023

Índice general

Índice de tablas

Índice de figuras

1 Introducción

El presente trabajo de título contribuye en el desarrollo de la plataforma experimental RUPU, diseñada en el Departamento de Electrónica con el propósito específico de evaluar algoritmos de control distribuido para pelotones de vehículos. Habiéndose diseñado, fabricado, y probado las componentes principales de la plataforma a nivel funcional como parte de trabajos previos, el trabajo desarrollado en esta memoria se enfocó en mejorar la usabilidad de la plataforma para facilitar la ejecución de experimentos reproducibles y repetibles para fines docentes y de investigación. Con este objetivo, se consideró la generación de documentación para el uso a nivel de componentes individuales y a nivel sistema, el desarrollo de una interfaz gráfica para configuración de experimentos y recolección de datos, y la preparación de un conjunto de configuraciones predefinidas que sirven para actividades de difusión y como demostradores tecnológicos. Además, siendo hasta el momento una plataforma experimental única en su tipo y de bajo costo, se busca establecer una identidad visual que incorpore los elementos gráficos como logos y tipografía que permitan la identificación visual del proyecto RUPU para efectos de difusión y su posicionamiento dentro del espectro de herramientas educacionales que permiten evaluar el desempeño de algoritmos de control asociados a la conducción autónoma.

1.1. Motivación y contexto

El avance en tecnologías de conducción autónoma ha dado pie a diferentes líneas investigativas asociadas al control de los vehículos que forman parte de dichas tecnologías, entre ellas, los sistemas multi-agente (MAS). En el contexto de conducción autónoma, los MAS corresponden a un conjunto de vehículos independientes, o agentes, que coordinan sus acciones manteniendo una formación predefinida sin la necesidad de un acoplamiento físico entre ellos. Las formaciones de MAS representan un tópico de gran interés en términos de investigación y desarrollo tecnológico, debido a los desafíos que presentan respecto a los algoritmos de control que se deben implementar, además de la comunicación y sincronización entre agentes. En particular, en este trabajo el foco es la formación de vehículos en pelotón, la cual consta de una fila de vehículos que siguen a un líder manteniendo una distancia configurable entre ellos mientras se mueven a lo largo de un camino predefinido como se muestra en la Figura 1.1. Se espera que este tipo de configuración de seguimiento de camino en pelotón constituya la base para futuras tecnologías ITS *(Intelligent Transportation Systems)* [1] y conceptos relacionados como los son las *Automated Highway Systems (AHS)*, *Cooperative Adaptive Cruise Control* (CACC) [2], A*utomated Driving Assistance Systems* (ADAS) [3], entre otros, trayendo mejoras en la seguridad, manejo de tráfico, eficiencia energética, reducción de emisiones, etc. [4–6].

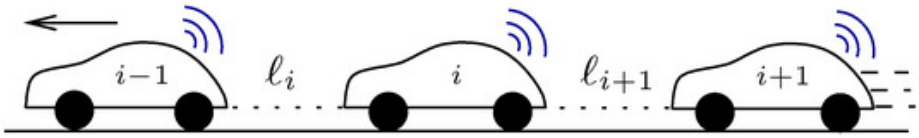


Figura 1.1: Representación formación en pelotón.

Desde una perspectiva de control, la formación de un pelotón seguidor de camino puede representarse como un problema doble [7]: (i) *el problema lateral*, el cual establece que cada vehículo en el pelotón avance manteniéndose dentro de un camino predefinido; y (ii) *el problema longitudinal*, que establece el requisito de que cada vehículo mantenga una distancia deseada con respecto al vehículo precedente. El estudio de los problemas lateral y longitudinal de manera simultánea ha recibido mucha atención en los últimos años [8–11]. Abordar simultáneamente ambos problemas de control para cada vehículo en el pelotón es un desafío, ya que alcanzar un objetivo puede perjudicar al otro debido a las dinámicas acopladas. Esta interacción entre los objetivos de control puede abordarse naturalmente aplicando técnicas de la teoría de control multivariable.

Existen múltiples enfoques para estudiar el problema de control de pelotones, los cuales dependen del tipo de vehículos, los grados de libertad de movimiento, los sensores disponibles, la capacidad de cómputo, la información compartida entre los vehículos u otros factores [12]. En escenarios prácticos, son muchos los factores que pueden afectar el desempeño del pelotón, entre ellos se encuentra una percepción ambiental poco confiable debido al ruido o fallas en los sensores locales, interferencias en la línea de visión debido a la curvatura o inclinación del camino [13–16], el manejo de redes comunicacionales sujetas a retraso, corrupción o pérdida de paquetes, entre otros desafíos.

Con el objetivo de probar la efectividad de los algoritmos de control bajo diferentes condiciones operacionales, es esencial validar estos controladores en escenarios reales donde puedan surgir problemas no modelados y que no sean capturados mediante simulaciones. En este contexto, los conglomerados más importantes han implementado y probado configuraciones de pelotón utilizando vehículos a escala real e infraestructura de carreteras [17]. Sin embargo, estos experimentos son orientados principalmente a la demostración tecnológica, y la información y resultados no se encuentra abiertamente disponible para el escrutinio e investigación académica. Por otro lado, el alto costo asociado con estos experimentos impide la investigación experimental de nuevos algoritmos de control y las pruebas de escalabilidad con pelotones de varios vehículos, donde surgen problemas como la estabilidad de cuerda [18]. Una alternativa adecuada a los experimentos a escala real es considerar plataformas a escala reducida que capturen la esencia de los problemas de control y permitan realizar pruebas de forma rápida y segura, considerando los desafíos en las tecnologías computacionales y de comunicación asociadas. Si bien, existen plataformas comerciales y educativas multipropósito para pruebas de vehículos autónomos, estas no cuentan con los sensores necesarios para tratar simultáneamente el problema lateral y longitudinal, o bien, tienen un alto costo que limita su accesibilidad a investigadores [19, 20].

A partir de lo anterior, se presenta la plataforma RUPU como una solución de bajo costo para probar y evaluar experimentalmente el desempeño de los algoritmos de control diseñados para los problemas que presenta la formación en pelotón. RUPU es una plataforma desarrollada por académicos y estudiantes de la Universidad Técnica Federico Santa María [21, 22] y consta de

un pelotón de vehículos a escala que operan sobre una pista de goma EVA ensamblable que permite modificar la ruta a seguir, la cual se encuentra demarcada por una línea blanca.

Al momento de iniciar este trabajo de memoria, la plataforma RUPU ya ha sido utilizada como base para experimentar y validar algoritmos de control enfocados en resolver el problema lateral y longitudinal que conforman el seguimiento de línea en formación de pelotón [22]. Sin embargo, a pesar de las mediciones que demuestran el desempeño de la plataforma en la experimentación, en su versión actual cuenta con una usabilidad limitada, en el sentido de que el procedimiento de configuración de experimentos debe realizarse manualmente y requiere conocimiento específico sobre las componentes de cada vehículo, lo cual lo vuelve tedioso y propenso a errores, incluso para los mismos diseñadores de la plataforma. En específico, la falta de una interfaz de usuario para ajustar los parámetros o enviar instrucción dificultan la operación de la plataforma, mientras que la inadecuada documentación requerida para poder reproducir los experimentos por otro grupo de personas limita la extensión de su uso. Esto se plasma en la imposibilidad de reproducir los resultados obtenidos en experimentos ya reportados debido a que no hay guías de uso o configuración que expliquen la forma de encender o conectar a la red los dispositivos de la plataforma, lo cual a su vez afecta la calidad y confiabilidad de los resultados utilizados para docencia e investigación.

1.2. Planteamiento del problema

Para formalizar el problema a tratar en esta memoria de título, se adoptarán las definiciones propuestas por la *Association for Computing Machinery* (ACM) para la revisión de artefactos y su respectivo *badging* [23], las cuales se describen a continuación:

- **Repetibilidad (mismas personas, misma configuración experimental):** Las medidas se pueden obtener con cierta precisión por el mismo equipo de personas utilizando el mismo procedimiento de medición, el mismo sistema de medición, bajo las mismas condiciones de funcionamiento y en el mismo lugar en múltiples pruebas.
- **Reproducibilidad (diferentes personas, misma configuración experimental):** Las medidas pueden ser obtenidas con cierta precisión por un equipo diferente, utilizando el mismo procedimiento de medición, bajo las mismas condiciones de funcionamiento y en el mismo lugar en múltiples pruebas.
- **Replicabilidad: (diferentes personas, diferente configuración experimental):** La medición puede ser obtenida con cierta precisión por un equipo diferente, un sistema de medición diferente, en una ubicación diferente en múltiples pruebas.

Considerando las definiciones presentadas, se establece como objetivo principal de este trabajo extender la infraestructura de la plataforma RUPU para mejorar la usabilidad y habilitar la capacidad de reproducibilidad de experimentos a través de la creación de una interfaz de usuario integrada para configuración y monitoreo en línea de experimentos, generación de documentación y manuales de uso, y preparación de configuraciones estándar con un enfoque *plug-and-play* para efectos de demostración y difusión.

Este proyecto se realizó a través de los siguientes pasos:

- **Estudio de la plataforma Duckietown.** Se comienza por indagar de forma empírica en el estado del arte de las plataformas de sistemas multi-agente autónomos a escala con fines

educativos. En particular, a partir de la investigación realizada en la asignatura previa a la memoria de título, se selecciono como caso de estudio la plataforma Duckietown, debido a que cuenta con la más amplia y detallada documentación, siendo además uno de los proyectos más citados en artículos de investigación y sobre el cual el MIT realiza sesiones virtuales para enseñar sobre conducción autónoma . A través de este análisis, se realizan una serie de ejemplos de casos de uso propuestos como tutoriales verificando las capacidades reales del dispositivo en cuanto a y logrando extraer los elementos y características trasladables a RUPU para mejorar su usabilidad.

- **Estudio de la versión original de RUPU y reproducción de experimento.** Esta etapa consiste en reunir la documentación disponible para operar los vehículos disponibles y tratar de reproducir los experimentos y resultados reportados en [21, 24]. Se encontró que no era posible reproducir los experimentos a partir de la documentación, por lo que se requirió el apoyo de los desarrolladores originales de la plataforma, encontrándose además dificultades para validar la reproducibilidad de resultados.
- **Desarrollo de documentación e interfaz gráfica**. Una vez se consigue lograr el funcionamiento de la plataforma, se documentan los códigos utilizados, las versiones de las librerías requeridas y las herramientas de software que permiten manejar la plataforma, quedando todo el proceso documentado en un tutorial en HTML con el objetivo de que pueda ser complementado y extendido en versiones futuras de la plataforma. Además, para facilitar el manejo de la plataforma y agilizar la configuración de parámetros y visualización de señales, se desarrolló una interfaz gráfica de usuario. Tanto la documentación como el código de la interfaz gráfica se encuentran disponibles en un repositorio GIT [25].
- **Evaluación experimental y desarrollo de demostraciones.** Finalmente, como última etapa, se realizan experimentos que permiten validar la repetibilidad y reproducibilidad de la plataforma.

1.3. Alcances y contribuciones

De acuerdo a lo realizado en etapas previas a este trabajo de título, y según determinaciones tomadas en su transcurso, se especifican los siguientes alcances:

- Para el desarrollo de esta memoria, se utilizarán los robots ya construidos anteriormente, tanto el modelo basado en ESP32 [21] como el modelo basado en Raspberry Pi [26], sin incorporar modificaciones físicas a la plataforma ni a los agentes móviles, asumiendo que cada uno de los componentes sobre los cuales están construidos los agentes funcionan correctamente.
- Este trabajo de memoria no busca implementar nuevos algoritmos de control ni alterar los códigos originales que gobiernan la plataforma. En su lugar, se verifica la reproducibilidad de los experimentos que ya se encuentran documentados desde la perspectiva de control en [22], asumiendo el funcionamiento correcto de cada agente en la totalidad de componentes que posee.
- El desarrollo de la interfaz gráfica favorecerá la funcionalidad y facilidad de uso de la plataforma, dejando para etapas futuras las mejoras en el rendimiento y eficiencia del código.

- La documentación disponible en la página web de RUPU se centra en las instrucciones necesarias para poner en funcionamiento la plataforma en su versión ESP32. Además, se brindan sugerencias sobre secciones pendientes, que incluyen la guía de ensamblado de la plataforma y la documentación relacionada con la versión basada en Raspberry Pi.

Entre las contribuciones de este trabajo, se espera que tanto la interfaz gráfica como la documentación y códigos generados sean de utilidad para quienes deseen someter a evaluaciones experimentales sus algoritmos de control para pelotones de robots, preparar actividades llamativas para fines de difusión, y también para quienes deseen continuar perfeccionando la plataforma RUPU e implementando mejoras que permitan explorar otros campos de investigación.

A partir del trabajo realizado y presentado en este informe, se preparó un artículo el cual fue enviado a la Conferencia IEEE ChileCon 2023 [27] y, al momento de entregado este informe, se encuentra aceptado para ser presentado en diciembre de 2023.

1.4. Organización del informe

Este proyecto se centra en un enfoque experimental, donde los elementos clave a proporcionar constituyen códigos y documentación específica relacionada a la utilización de la plataforma, los cuales se encuentran disponibles en un repositorio GIT [25] para ser empleados y validados en RUPU.

El contenido de este documento complementa los entregables principales del trabajo final, teniendo como propósito entregar información adicional sobre el contexto del trabajo, el proceso de desarrollo y decisiones que fue necesario tomar en el camino, además de resultados de experimentos específicos que validan la funcionalidad de las herramientas desarrolladas.

Los siguientes capítulos se estructuran según se describe a continuación:

- En el **Capítulo 2** se describe el trabajo previo, el cual consiste en el sondeo de las principales plataformas educativas orientadas a la conducción autónoma, lo que se encuentra complementado por la experimentación con la plataforma Duckietown. Además, en esta sección se describen las principales características de RUPU.
- En el **Capítulo 3** se realiza una exhaustiva exploración para la implementación de la interfaz gráfica en todas sus etapas de desarrollo. Se abarcan desde las primeras iteraciones hasta las modificaciones finales. Este capítulo brinda una visión completa del proceso de diseño y mejora de la documentación recopilada y plasmada en una página web para posteriores usos de RUPU.
- En el **Capítulo 4** se presentan los resultados experimentales obtenidos a través de un análisis de la plataforma. Este capítulo no sólo se dedica a exponer los resultados en sí, sino que también se adentra en el proceso de evaluación que condujo a estos resultados.
- En el **Capítulo 5** se resumen las principales conclusiones de este trabajo entregando directrices y recomendaciones para los trabajos futuros.

2 Antecedentes

A partir del trabajo realizado previamente en la asignatura Proyecto de Título (ELO307), para la cual se realizó el estudio de diferentes plataformas orientadas a la evaluación práctica y experimentación con algoritmos de conducción autónoma, se identificó Duckietown como la más relevante de ellas. En este capítulo se presentarán las conclusiones obtenidas de dicho trabajo respecto a los elementos a incorporar en RUPU con el objetivo de facilitar su uso.

Además se detallan las características de RUPU y se presenta información sobre su funcionamiento en el estado previo al desarrollo de este trabajo, resumiendo aspectos técnicos para la implementación de mejoras en la usabilidad que se describe en los siguientes capítulos.

2.1. Resumen de plataformas experimentales educativas

Como primer paso para abordar el problema de usabilidad inherente a RUPU se realiza un estudio de las plataformas de vehículos autónomos a escala que se encuentran disponibles en el mercado, comenzando por Duckietown y TurtleBot, las cuales se encuentran identificadas como las principales plataformas en cuanto a vehículos experimentales a escala [22]. Se desglosan las principales características de Duckietown con el fin de realizar una búsqueda más exhaustiva de plataformas ampliando la investigación fuera del ámbito educativo donde también se encuentra ZUMI [28], abarcando las plataformas de concepto *open* tanto de hardware como de código entre las cuales se encuentran Qcar [29] y DonkeyCar [30] y finalmente las plataformas orientadas al desarrollo de algoritmos a través de competencias como AWS DeepRacer o F1Tenth.

De todo este análisis, se corrobora la información que posiciona a Duckietown como la plataforma experimental más refinada en el campo de la educación, documentación y capacidad de extender su desarrollo gracias a los potentes elementos de hardware que la componen, procediendo de esta forma a una evaluación práctica de la plataforma.

2.2. Duckietown

Esta plataforma desarrollada por el *Massachusetts Institute of Technology* (MIT) cuyo objetivo es construir una plataforma a pequeña escala y accesible, pero que conserve los desafíos científicos inherentes a una plataforma de robots autónomo a escala real [31], implementando algoritmos de conducción autónoma y reconocimiento de imagen entre sus características principales.

El trabajo realizado en ELO307 contempla la utilización de un Duckiebot ya ensamblado previamente, y se enfoca en el objetivo de caracterizar la plataforma en términos de usabilidad por sobre su potencial en el desarrollo de algoritmos. Por esta razón, se comienza instalando y configurando el entorno de trabajo que permite controlar el robot donde el principal material de apoyo para esta tarea corresponde al manual de *set up* [32] mostrado en la Figura 2.1 en el cual se establece como restricción de funcionamiento óptimo el uso de un computador con sistema operativo Linux.

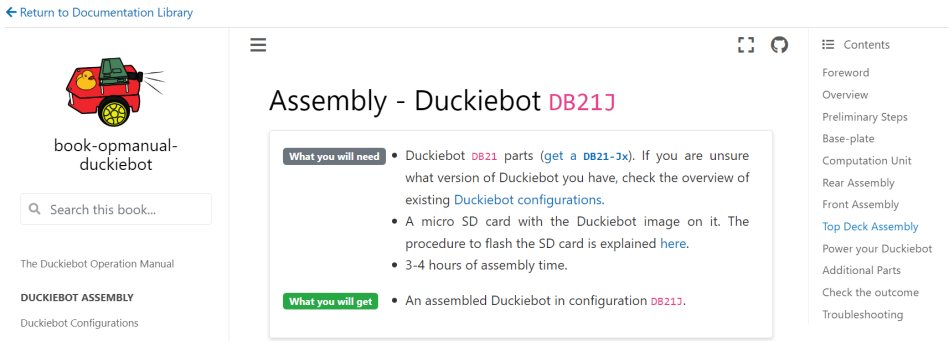


Figura 2.1: Página web de documentación de Duckietown.

Una vez instalados todos los programas necesarios y siguiendo los requerimientos de hardware indicados se puede operar finalmente con Duckietown, por lo que se comienzan a realizar los primeros tutoriales y demostraciones validando el funcionamiento de la plataforma para alguno de los experimentos propuestos. Los resultados de dichos experimentos fueron reportados en ELO307 y su aporte en este contexto son las herramientas que permiten y facilitan el uso de la plataforma.

Entre estas herramientas se encuentra la *dashboard* de Duckietown, presentada en la Figura 2.2 y está compuesta por dos elementos principales, el control que permite mover el vehículo y una página web que se aloja en un puerto local del computador del usuario y que cuenta con dos pestañas, una que se encarga de mostrar el estado del robot a través de gráficos de temperatura interna, utilización de CPU y batería, y otra que muestra las señales medidas por el robot tales como distancia, velocidad y movimiento de los motores, entre otras.

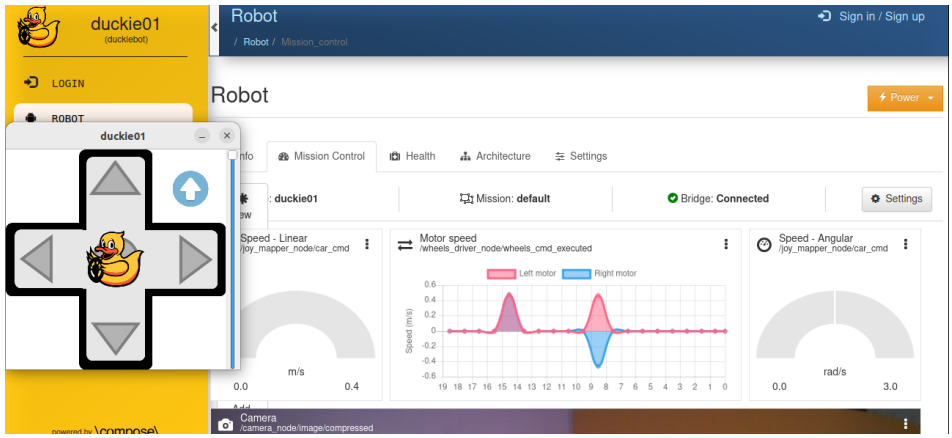


Figura 2.2: Dashboard de Duckietown.

En base a la experiencia obtenida en la evaluación de Duckietown, se destaca que la disponibilidad de una pagina de documentación, disponibilidad de un repositorio GIT con códigos de demostración, y una interfaz de usuario intuitiva para controlar y monitorear los vehículos, representan factores diferenciadores de Duckietown respecto a otras plataformas evaluadas.

2.3. PLToon

Como precursora al desarrollo de RUPU, se diseñó la plataforma PLToon, la que se encuentra descrita en profundidad en el artículo *"PL-TOON: A Low-Cost Experimental Platform for Teaching and Research on Decentralized Cooperative Control"* [33], en este se introduce una plataforma experimental económica diseñada para respaldar la enseñanza de teoría de control y permitir la emulación de resultados teóricos relacionados con el control de un pelotón de vehículos no tripulados en una dimensión, la cual se puede observar construida en la Figura 2.3.

Dicho artículo entrega detalles sobre el diseño y construcción de la plataforma, además de presentar el diseño del lazo de control y describir el esquema de comunicación entre los vehículos, sin embargo, dado que se limita a resolver únicamente el problema longitudinal, motiva a la construcción de la plataforma RUPU, la que contempla tanto el problema longitudinal como la incorporación del problema lateral para formaciones en pelotón de vehículos no tripulados.

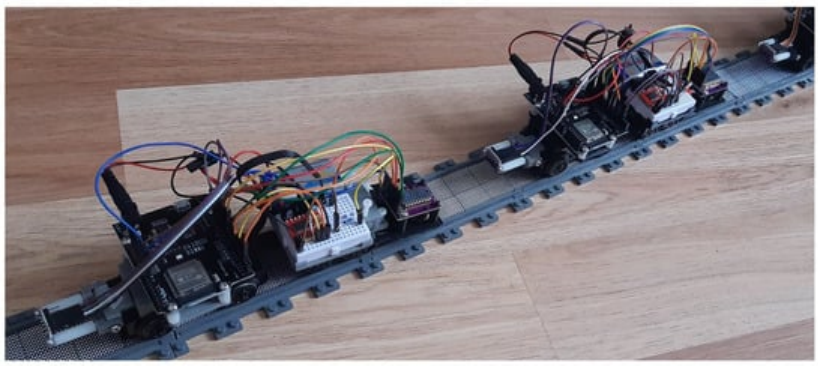


Figura 2.3: Pista y vehículos PLToon.

2.4. Descripción de RUPU

En esta gama de soluciones para desarrollar y experimentar con algoritmos de conducción autónoma se introduce RUPU, esta plataforma ha sido diseñada para estudiar problemas de control en configuraciones de pelotón, especialmente los problemas lateral y longitudinal. Este propósito, junto al bajo costo de construcción y su capacidad de extensión en campos de investigación como el reconocimiento de imagen para la detección de caminos, le entregan a RUPU características únicas que lo posicionan como una solución innovadora y versátil en su campo, a diferencia de Duckietown, que no se enfoca en los problemas de pelotón y su accesibilidad es reducida.

2.4.1. Descripción de la plataforma

La plataforma se compone de dos elementos principales, el primero de ellos corresponde a los agentes móviles que se comunican entre ellos, para entregarse información sobre su estado, y a un monitor/operador encargado de enviar instrucciones y visualizar parámetros, los agentes pueden estar basados en ESP32 [21] o Raspberry Pi [26]. La primera de ellas es la precursora de la siguiente, cuenta con mayor número de vehículos y ha sido utilizada para demostraciones en actividades de difusión, mientras que la versión en Raspberry Pi se encuentra en etapas tempranas de desarrollo y busca ampliar las funcionalidades de su versión anterior al incorporar una Raspberry Pi como unidad de procesamiento central, esta versión de la plataforma es actualmente funcional, pero su uso se encuentra vinculado a una cuenta en la plataforma Ubidots, la que hace de dashboard para visualizar las señales del dispositivo y leer los mensajes UDP con las instrucciones, obstaculizando en su código la comunicación UDP con otros dispositivos. Cada una de las versiones se encuentra ilustrada en la Figura 2.4. Por su parte, el segundo elemento de RUPU corresponde a la pista ensamblable de goma EVA compuesta por tapetes negros de $29 \times 29 \times 0.8$ [cm] marcados con una línea blanca que indica el camino conformando una pista con sus agentes como se presenta en la Figura 4.1.

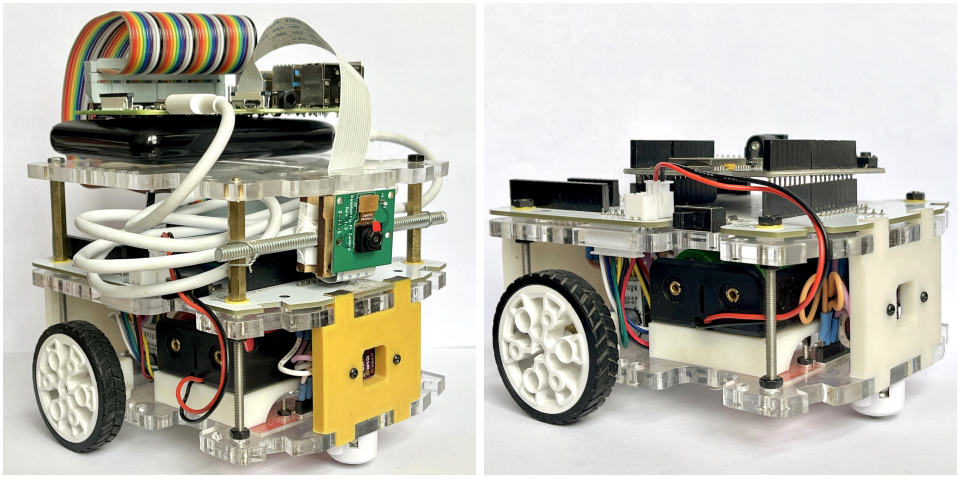


Figura 2.4: RUPU versión Raspberry Pi y ESP32.

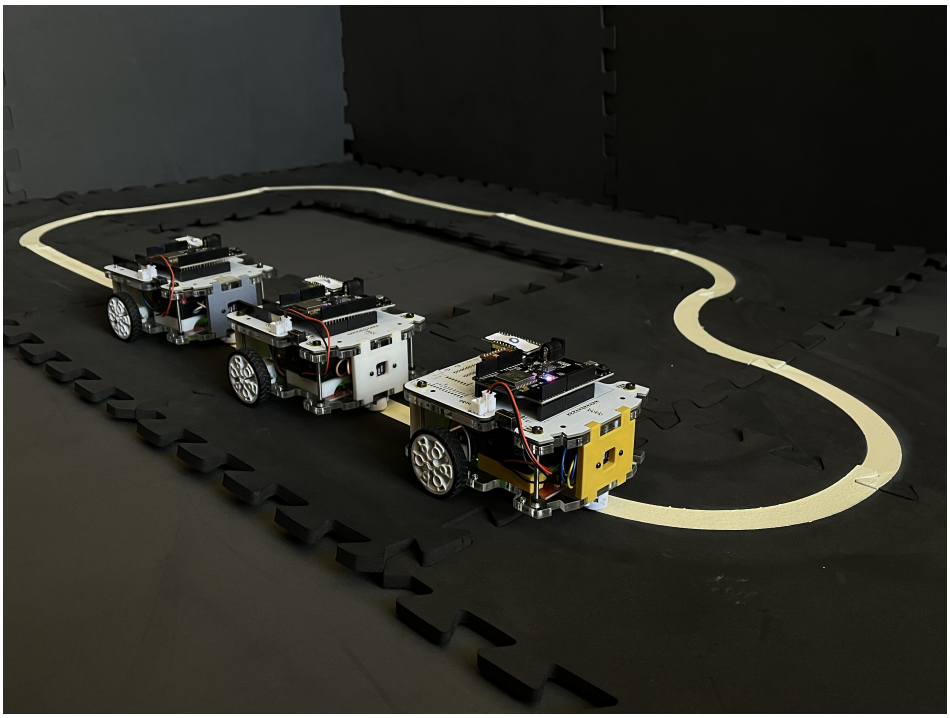
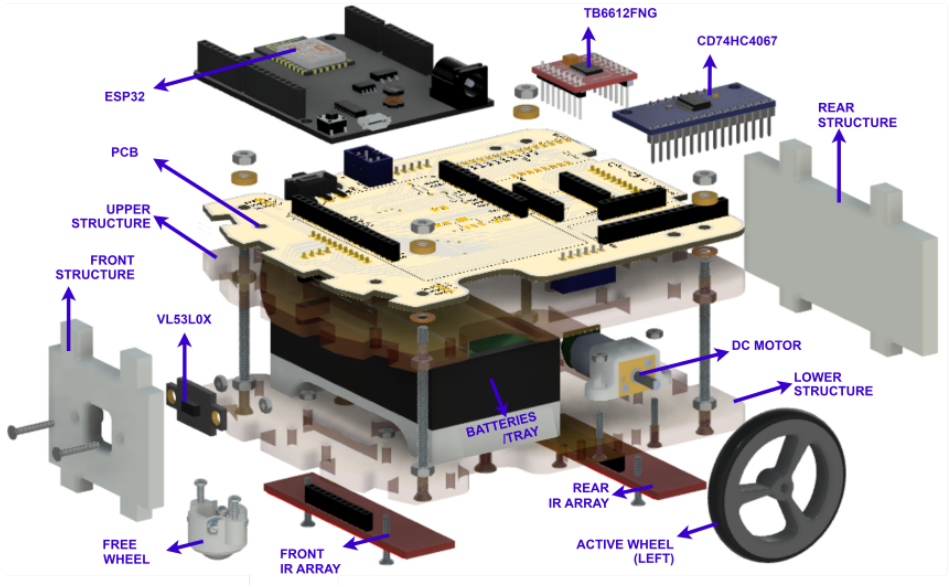
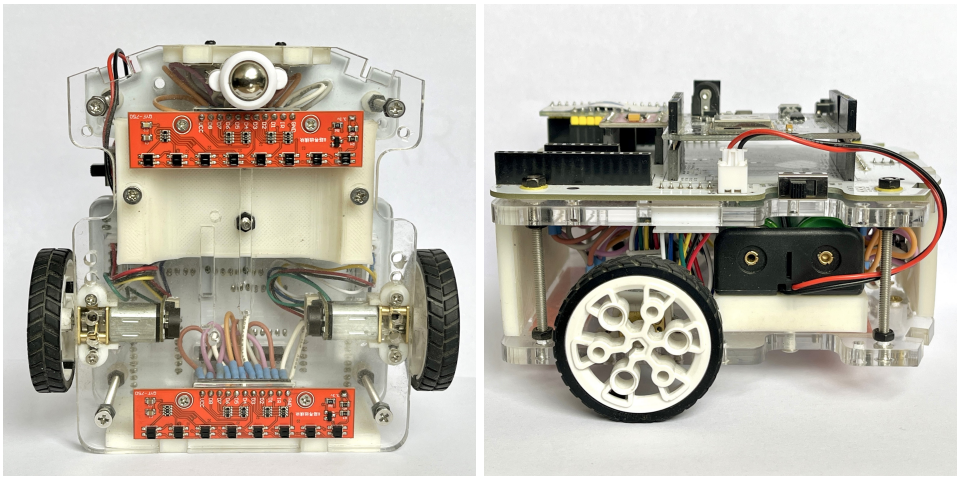


Figura 2.5: Pista con 3 vehículos.

Este trabajo de memoria se basa en el modelo ESP32, ya que como se explicó anteriormente se encuentra en etapas de desarrollo más avanzadas, permitiendo ser usada actualmente de forma demostrativa y sin dependencias a una cuenta de Ubidots. En esta plataforma, cada agente se implementa como un robot móvil de accionamiento diferencial (DDMR), la Figura 2.6a muestra una vista de las componentes mecánicas y electrónicas de un agente. El chasis del DDMR está compuesto principalmente por cuatro piezas planas (estructura delantera, trasera, inferior y superior). El agente se mueve a través de dos ruedas activas de 43 [mm] de diámetro controladas de forma independiente, permitiendo movimiento hacia atrás, adelante, girar, doblar o mantener una trayectoria curva. La versión actual utiliza dos motores DC N20, cada uno contiene un *encoder* embebido y una caja de cambios a razón de 100:1. Las componentes electrónicas están montadas en una placa de circuito impreso (PCB), la cual se acopla a la estructura superior. En la parte inferior se incorpora una bola rodante para estabilizar la estructura y un contenedor para baterías. La Figura 2.6b muestra las vistas inferior y lateral de un agente ensamblado, cuyas dimensiones son 10×12 [cm].



(a)



(b)

Figura 2.6: Componentes principales de un agente móvil. Vista inferior y lateral de un agente ensamblado.

La Figura 2.7 muestra un diagrama de interconexión de las principales componentes electrónicas para el accionamiento, sensado y cómputo. Un controlador de motor TB6612FNG provee dos canales independientes para controlar ambas ruedas. Cada agente cuenta con un sensor de distancia VL53L0X que permite al vehículo medir la distancia a un objeto ubicado en su línea de vista frontal, dado que esta medición no es confiable en zonas de curvas, se cuenta con un algoritmo que de forma off-line es capaz de procesar un video con vista cenital del experimento para estimar las distancias reales entre los vehículos. La orientación de cada agente respecto a la línea blanca en el camino se mide a través de dos arreglos con ocho sensores infrarrojos QRE113 ubicados en la parte inferior de cada agente, uno en la parte delantera y el otro en la trasera. Los sensores permiten determinar si los agentes se encuentran sobre la línea y conocer la orientación con respecto a su eje de traslación. Se utiliza un multiplexor análogo CD74HC4067 para manejar la información de los arreglos de sensores. La velocidad del agente

es medida a través de encoders *Hall-effect* ubicados dentro de cada motor DC, los que entregan información sobre la posición angular, rotación, y velocidad angular de cada rueda. Cada agente incorpora una placa de desarrollo Wemos D1 R32 basada en un microcontrolador ESP32 de bajo costo con WiFi integrado, lo cual permite la comunicación entre agentes y también la interacción en línea entre el agente y un computador externo para monitoreo y configuración de parámetros.

Todas las componentes son alimentados por dos baterías *Lithium-Ion* en formato cilíndrico 18650, entregando un voltaje nominal de 7.2 [V] y máximo de 8.4 [V]. Su capacidad promedio es de 2200 [mAh], suficiente para proveer alrededor de 90 minutos de autonomía a cada vehículo.

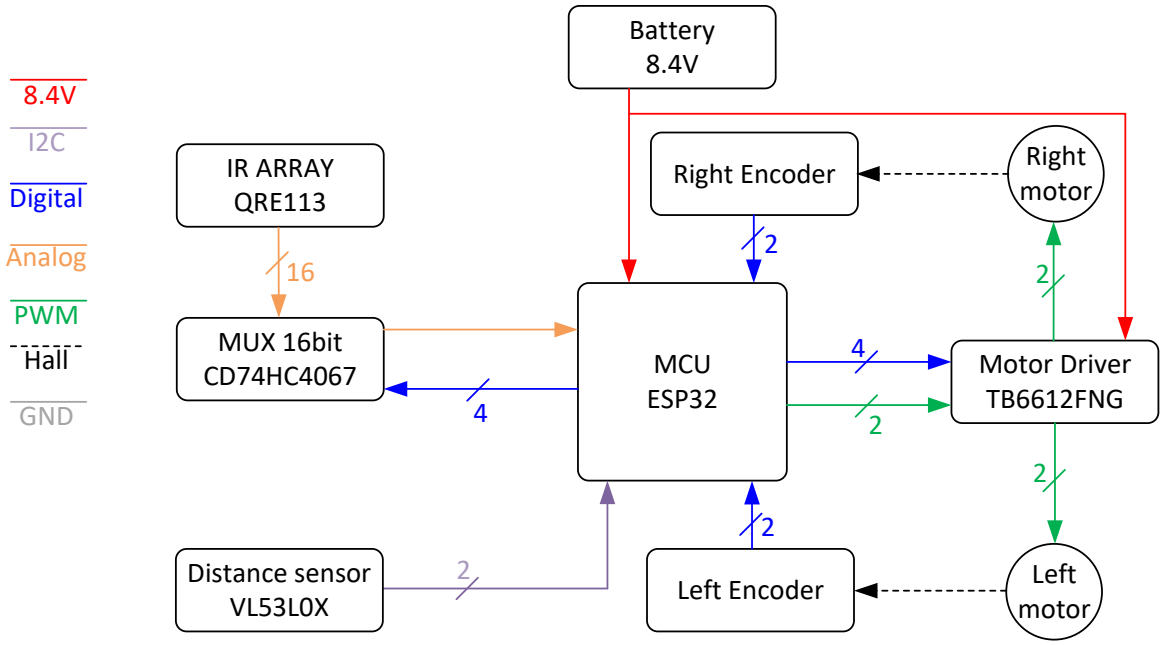


Figura 2.7: Esquema de interconexión de las principales componentes de los agentes móviles.

2.4.2. Esquema de control

Como complemento a las funcionalidades de la interfaz gráfica que permiten modificar los parámetros del controlador que se presentarán en el Capítulo 3, se describe en esta sección el esquema del lazo de control implementado en la plataforma.

Este esquema de control, se encuentra basado en controladores PID, para las variables de velocidad, orientación y distancia del DDMR y se encuentran configurados como se muestra en la Figura 2.8 por la notación C_1, C_2 y C_3. Cada uno de estos controladores puede ser configurado en sus tres parámetros; proporcional, integral y derivativo y además, cuentan con un bloque de saturación que actualmente se encuentra establecido en un 10 % y que solo puede ser modificado en el código .ino que permite programar cada agente.

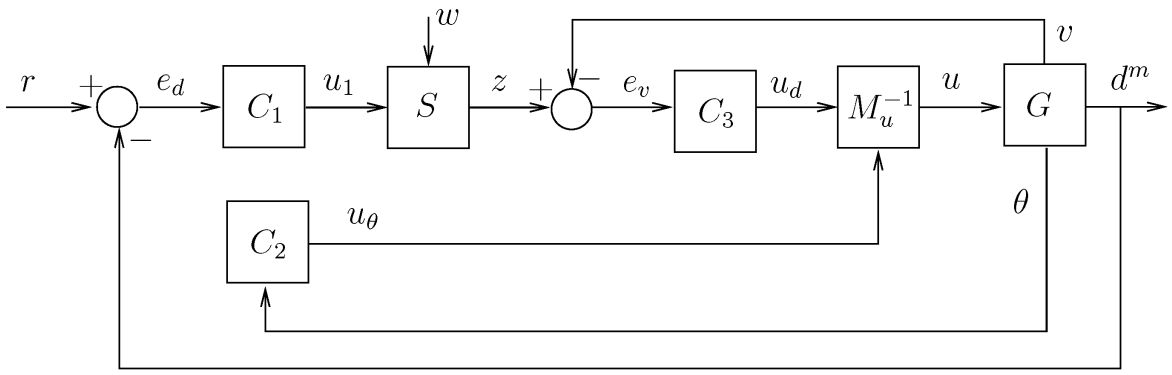


Figura 2.8: Esquema de control de RUPU.

El detalle del diseño de este esquema de control y los experimentos que entregan resultados sobre su desempeño se encuentran documentados en [22].

2.4.3. Comunicación de los agentes

Para dar soporte al trabajo detallado en el Capítulo 3 se requiere comprender a fondo la operación de los agentes en términos de comunicación, por lo que utilizando como referencia el trabajo de memoria que explica la elaboración de RUPU [21] se da cuenta del funcionamiento de los dispositivos con enfoque en la transmisión y el monitoreo, lo que representa los tópicos de relevancia para el desarrollo de la interfaz.

Configuración remota de parámetros

En esta sección se busca comprender la interacción configurada en el trabajo previo a esta memoria entre los agentes y el dispositivo externo que controla su funcionamiento, sin indagar en la comunicación entre los agentes. Para esto, se encuentra implementada una topología "estrella" donde cada uno de los agentes se encuentra conectado a través de un dispositivo central conectado a la red local, el cual se denomina "Operador", que se comunica con cada uno de los robots utilizando su respectiva dirección IP y puerto a través del protocolo UDP, esta topología implica que cualquier cambio de parámetro puede ser realizado únicamente por dicho "Operador", al cual los agentes pueden enviar su respectiva información como se muestra en la Figura 2.9.

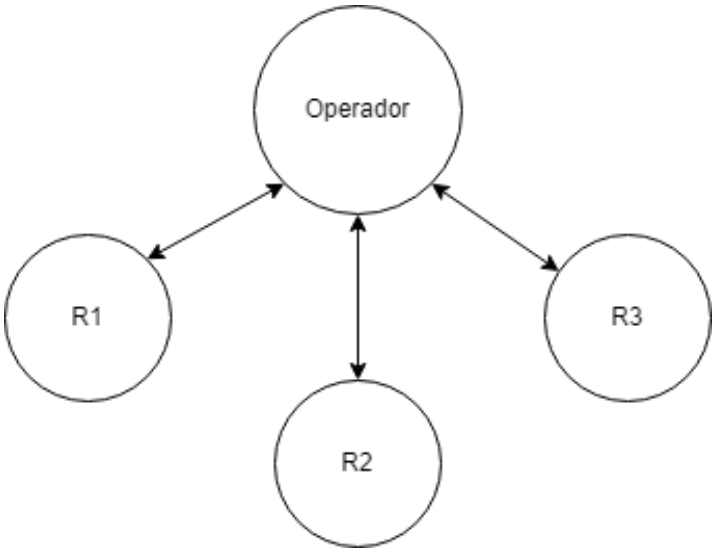


Figura 2.9: Diagrama de comunicación.

La estructura de los mensajes UDP válidos y decodificables por los agentes se encuentra detallada en la expresión 2.1 donde E y *variable* son parámetros específicos que se encuentran documentados con anterioridad en [21] donde se presenta la Tabla 2.1 con la lista de comandos habilitados y donde el parámetro *valor* es el único que queda a determinación del usuario.

$$\text{E/variable/valor} \tag{2.1}$$

Tabla 2.1: Comandos definidos para el operador de la plataforma experimental.

Comando	**Descripción**
E/co_p/valor	Define la ganancia proporcional del controlador de orientación
E/co_i/valor	Define la ganancia integral del controlador de orientación
E/co_d/valor	Define la ganancia derivativa del controlador de orientación
E/cv_p/valor	Define la ganancia proporcional del controlador de velocidad
E/cv_i/valor	Define la ganancia integral del controlador de velocidad
E/cv_d/valor	Define la ganancia derivativa del controlador de velocidad
E/cv_ref/valor	Define la referencia de velocidad
E/cd_p/valor	Define la ganancia proporcional del controlador de distancia
E/cd_i/valor	Define la ganancia integral del controlador de distancia
E/cd_d/valor	Define la ganancia derivativa del controlador de distancia
E/cd_ref/valor	Define la referencia de distancia
E/calibrar/valor	Permite calibrar los arreglos de sensores IR si valor=1
E/parar/valor	Permite iniciar el ciclo de control (valor=si), o detenerlo (valor=no)

Monitoreo

Aprovechando la capacidad de los agentes para enviar información remota a través de comunicación UDP, cada robot comparte sus variables internas a un dispositivo "Monitor" a través de la red local en el formato descrito en la expresión 2.2.

$$\text{carácter identificador, variable 1, variable 2, ..., variable j} \tag{2.2}$$

El Algoritmo 1 describe los pasos realizados por cada agente RUPU en el proceso de envío de datos al Monitor.

Algorithm 1: Envío de datos para monitoreo.

Entrada: Definir la dirección IP y el puerto del dispositivo monitor

1: Iniciar la escritura de datos UDP hacia el dispositivo monitor (IP + puerto remota)
2: Definir el mensaje con los datos de interés, según la expresión 2.2
3: Construir el datagrama (dirección-mensaje) y enviar hacia la dirección remota (dispositivo monitor)

Al momento de comenzar este trabajo se contaba con resultados que validaban el correcto funcionamiento de las interfaces y esquemas de comunicación en RUPU que se han documentado en la Tabla 2.1 y el Algoritmo 1; sin embargo, a pesar de contar con esta información y tener claro el funcionamiento teórico de la plataforma, no era posible ponerla en marcha, dando cuenta del problema respecto a la usabilidad de la misma.

No existía documentación sobre como manejar las herramientas y aplicaciones para configurar la plataforma, por lo que luego de varios intentos de prueba error en los que no se obtuvieron resultados, se decidió contactar al diseñador de la plataforma quien realizó una sesión de capacitación respecto a su uso. En esta, se traduce de forma práctica lo documentado en la Sección 2.4.3 a utilizar un teléfono celular con una aplicación que permite enviar mensajes UDP a modo de "Operador", configurando uno a uno cada agente. Esto implicaba cambiar de IP en la aplicación por cada vez que se quisiera enviar un mensaje a un robot diferente, calibrando a cada robot por separado y añadiendo tiempo y complejidad en la operación.

En el caso de querer monitorear o guardar los datos registrados por los dispositivos a través del "Monitor" mencionado en la Sección 2.4.3, era necesario utilizar un computador, que a través de un código de Python, genera un archivo de extensión .csv con las expresiones en el formato indicado en 2.2 para su posterior análisis, requiriendo finalmente de al menos dos equipos conectados a la red local para poder realizar la medición y configuración de parámetros de la plataforma, pero además, impidiendo la visualización de señales en línea debido al procesamiento *off-line*.

Además, en esta sesión de capacitación se entrega información respecto a la rutina de programación de cada agente, la cual contempla el código de Arduino que se carga en cada uno, las variables que se editan de este código y el orden en el cual debe ser subido a cada robot, dicha información era de conocimiento exclusivo del diseñador de la plataforma y resultaba crucial para ponerla en funcionamiento.

Tanto la dificultad en la configuración de parámetros de RUPU, como su rutina de programación, sumado a lo inconveniente del post-procesamiento de señales de interés para su visualización, es que resulta necesario plantear las mejoras a implementar en la plataforma con el fin de facilitar su uso.

2.5. Especificaciones de mejoras

A partir de las funcionalidades provistas y las debilidades detectadas a la hora de poner en funcionamiento la plataforma RUPU, se define como resultado esperado para mejorar su usabilidad la implementación de una interfaz gráfica que cumpla con los siguientes requerimientos:

- Unificar monitor y operador, contando con una interfaz que pueda ejercer ambos roles.
- Contar con un número configurable de agentes.
- Configurar los parámetros de referencia y controlador en cada uno de los agentes.
- Visualizar las señales de interés, principalmente velocidad, distancia al predecesor y orientación a través de la interfaz.
- Capturar datos para posibles análisis futuros.

Como se mencionó en la descripción de RUPU, su diseño accesible y su posibilidad de aportar en el campo de la experimentación con algoritmos de conducción autónoma para formaciones en pelotón lo favorecen como plataforma demostrativa y académica. Por esta razón, complementar su capacidad técnica con un enfoque centrado en la usabilidad y reproducibilidad se vuelve un paso natural en su desarrollo y expansión de su uso.

3 | Implementación de mejoras

El potencial de RUPU como herramienta de docencia y experimentación motiva a mantener su desarrollo especialmente para actividades demostrativas, por lo que a contar del presente año se han unido al equipo nuevos estudiantes con intenciones de impulsar el proyecto. Sin embargo, el primer obstáculo presentado es la falta de manuales o instructivos que cuenten con información detallada respecto al uso de la plataforma, imposibilitando la reproducción de los experimentos documentados con anterioridad [22]. De esta forma, según lo propuesto en el Capítulo 2 se busca extender las funcionalidades de la plataforma por medio de la incorporación de una interfaz gráfica que facilite la configuración de parámetros y monitoreo de señales de interés de cada robot.

El propósito de este capítulo es describir el proceso de desarrollo de la GUI desde sus primeras versiones y presentar los resultados de la documentación del uso de la plataforma plasmado en la página web de RUPU, disponible en el repositorio GIT [34]. Tanto el código de la interfaz gráfica como la página web son material autocontenido que busca ser complementado y actualizado con los trabajos posteriores, correspondiendo al principal entregable de este trabajo de memoria. Por su parte, este informe sirve para dar un contexto general al material desarrollado.

3.1. Primeras implementaciones de la interfaz gráfica de usuario

Como se mencionó en el Capítulo 2, el objetivo es implementar una interfaz de usuario que a través del protocolo UDP, sea capaz de controlar a los agentes RUPU enviando instrucciones que permitan la configuración de sus parámetros. Además, se establece que debe ser capaz de monitorear señales de interés como lo son la distancia, velocidad y orientación de cada uno de los agentes, generando una GUI con un concepto similar al *dashboard* de Duckietown.

Para esto, se comenzó explorando la implementación de la interfaz en MATLAB, los motivos que impulsaban esta decisión eran, por un lado la herramienta nativa de MATLAB llamada AppDesigner, que permite de forma simple y guiada incorporar los elementos y funcionalidades del software en una interfaz gráfica basada en programación orientada a objetos mientra que por otro lado, se consideró el soporte del Add-on *Hardware Support* disponible para facilitar el manejo de Raspberry Pi y ESP32 a través de MATLAB.

3.1.1. Implementación en Raspberry Pi

Considerando que Raspberry Pi era la evolución más reciente de la plataforma RUPU y que cuenta con mayor documentación del complemento Hardware Support [35], encargado de establecer conexión con Raspberry Pi, se decidió comenzar por implementar la interfaz para esta versión.

A través del complemento, MATLAB habilita la opción de enviar y recibir información controlando el dispositivo de forma inalámbrica vía SSH, por lo que los primeros pasos consistieron en integrar la visión por cámara y el accionamiento de pines a través de una interfaz montada con App Designer, sin incluir los códigos diseñados para el control en formación de pelotón como se muestra en la Figura 3.1.

Los códigos de MATLAB asociados a la interfaz desarrollada se encuentran disponibles en el repositorio GIT de este proyecto, sin embargo, no fueron utilizados en la versión final de la interfaz al presentar complicaciones con el protocolo de comunicación escogido y la versión de RUPU ESP32. Cabe destacar que para el caso de Raspberry Pi, esta interfaz podría adaptarse y mejorarse, pero en esta etapa, compromete la funcionalidad del proyecto al no ser compatible con ESP32.

Finalmente, se documentan las versiones utilizadas en este desarrollo, las cuales son; MATLAB R2023a, Linux RaspberryPi 5.15.32-v7l y Python 3.

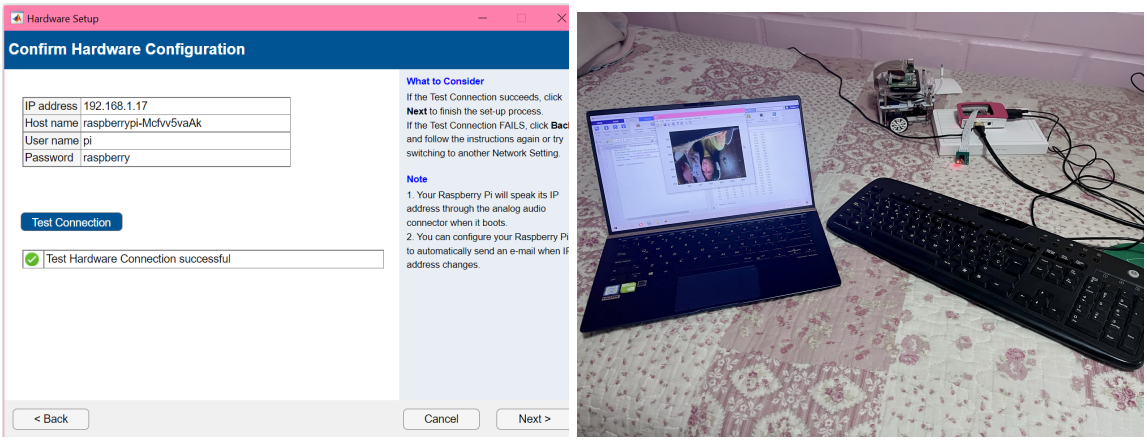


Figura 3.1: MATLAB Hardware Support para Raspberry Pi.

3.1.2. Implementación en ESP32

De la misma forma, se procedió a trabajar con la ESP32, sin embargo, luego de instalar y utilizar Hardware Support para Arduino, se determinó que su uso implicaba la reescritura de los códigos, pues la función de este complemento es hacer de puente entre un código de MATLAB y la ESP32, sin necesidad de pasar por el IDE de Arduino y dejando fuera los códigos escritos en C++. Evidentemente, esto resultaba en un trabajo adicional que no aportaba valor al desarrollo de la plataforma, instando a buscar soluciones alternativas para visualizar las señales y configurar los parámetros desde MATLAB.

Dicho camino alternativo consistió en controlar la ESP32 enviando instrucciones a través

de protocolo TCP, el cual es fácil de implementar en MATLAB y no requiere de ninguna librería adicional como se muestra en la Figura 3.2. Finalmente, a pesar de que es posible establecer comunicación con la ESP32 utilizando los códigos de prueba disponibles, se descarta esta opción debido a que el propósito es establecer comunicación a través del protocolo UDP, pues corresponde a un alcance de esta memoria no modificar los dispositivos ya construidos y programados, lo que incluye sus respectivos esquemas de comunicación y control. Si bien, se puede establecer comunicación UDP como una función exclusiva de la librería de pago *Instrument Control Toolbox*, se compromete la accesibilidad a la plataforma.

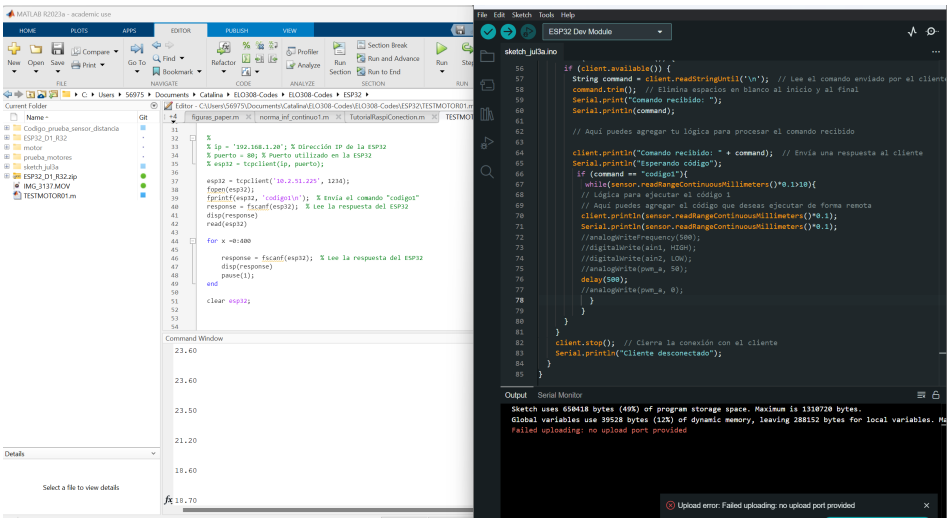


Figura 3.2: Comunicación TCP MATLAB-ESP32.

Considerando las limitaciones encontradas en el uso de MATLAB, que no significan un impedimento, pero si escapan de los alcances propuestos, se explora el uso de Python para la implementación de la interfaz.

Esta alternativa no se escoge de forma arbitraria, es a raíz de la falta de conocimiento y documentación respecto a la puesta en marcha de los robots que el diseñador de ellos realizó una capacitación para comprender el funcionamiento de los dispositivos, en dicha capacitación, además de presentar el paso a paso necesario para activar los robots, indicó una serie de códigos de prueba, entre ellos un código de Python que lee y grafica la velocidad instantánea del robot líder.

Este código implicaba resolver uno de los requerimientos de la interfaz gráfica, la lectura y muestreo de señales de forma gráfica. Conocer dicho código y su funcionamiento permitía ampliar la lectura a otras señales, o bien, a otros agentes en el pelotón. Por lo que a partir de este punto, se comienza a trabajar en una nueva versión de la interfaz gráfica, basada en Python, lo que va en línea con mantener accesibilidad de la plataforma, permitiendo mayor extensibilidad y utilización de recursos computacionales.

3.2. Implementación de la interfaz gráfica de usuario

Una vez escogido el entorno de trabajo que dará soporte al desarrollo de la interfaz, se exploran las librerías disponibles para su implementación [36], de las cuales se decide utilizar Tkinter, tanto por la experiencia previa de quien desarrolla esta memoria con la librería como por la ventaja de que corresponde a un paquete preinstalado de Python facilitando su portabilidad entre computadores independiente del sistema operativo utilizado por el usuario.

En su versión inicial, se contempló incorporar todos los parámetros configurables del vehículo en una sola pestaña unificada, esto incluye la configuración del número de vehículos y la IP de cada uno, además de los comandos de inicialización en una misma vista. Al querer incluir la posibilidad de agregar más agentes, fue necesario incorporar una segunda ventana para que se pudieran visualizar todos los controles.

Dado que el monitoreo utilizaba la función de animación de la librería Matplotlib, la cual abría necesariamente otra pestaña, fue necesario ajustar el tamaño de la ventana de configuración para dar espacio a la visualización de señales.

Por último, se incorporaron elementos de la librería CustomTkinter disponible para Python y cuya estructura replicaba los elementos de Tkinter, permitiendo cambiar la visualización de los elementos de la GUI de una forma más estética, donde la disposición final de los elementos se encuentra presentada en la Figura 3.3, la cual muestra la composición de las pestañas que se encuentran en la ventana de configuración.

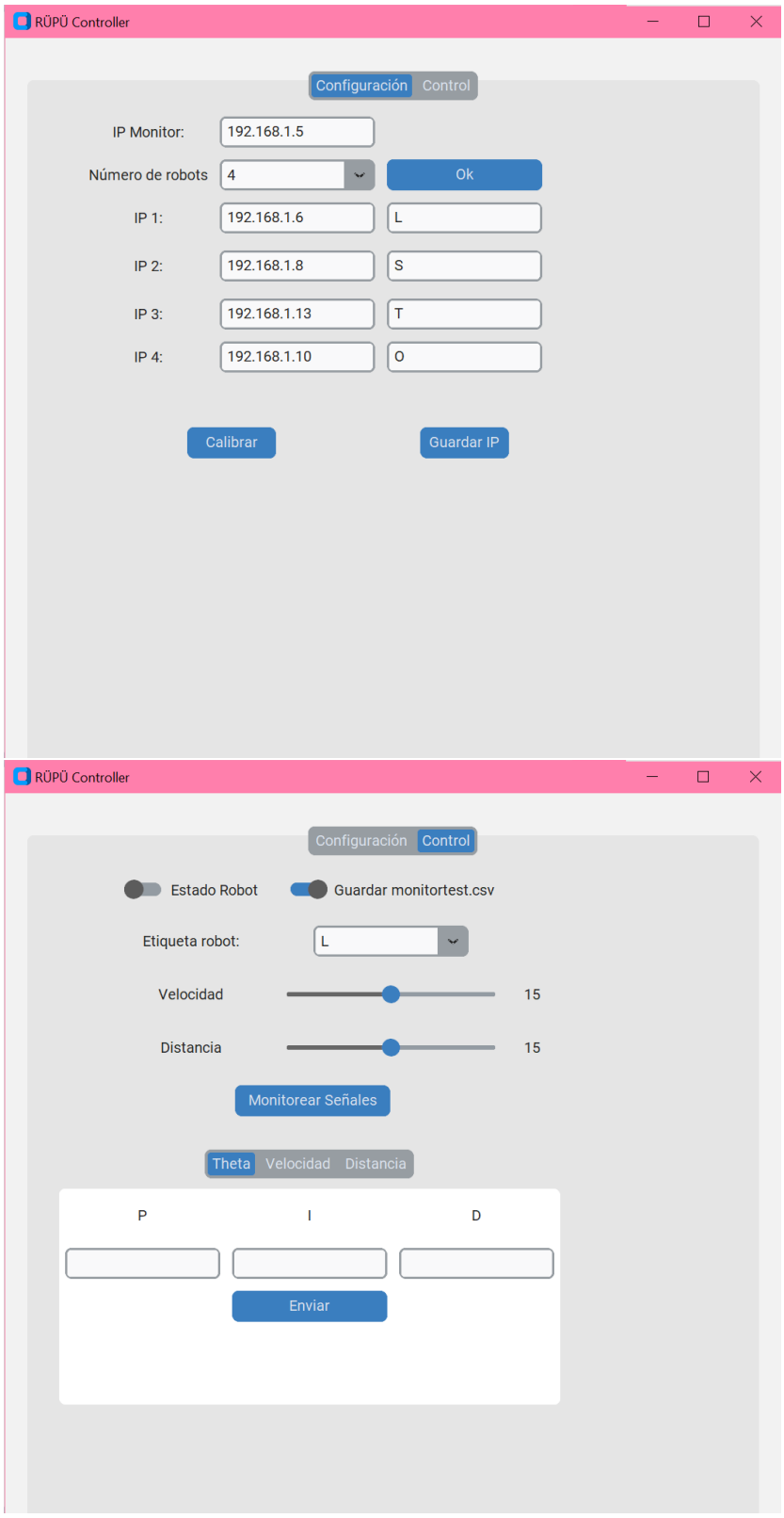


Figura 3.3: Pestañas de Configuración y Control GUI.

A continuación, se presenta la Tabla 3.1 que resume la conexión entre las instrucciones descritas en la Tabla 2.1 y los elementos presentas en la GUI que se encargan de realizarlas.

Tabla 3.1: Funcionalidad elementos GUI

Elemento GUI	**Comando**
Pestaña controlador orientación	E/co_p/valor
Pestaña controlador orientación	E/co_i/valor
Pestaña controlador orientación	E/co_d/valor
Pestaña controlador velocidad	E/cv_p/valor
Pestaña controlador velocidad	E/cv_i/valor
Pestaña controlador velocidad	E/cv_d/valor
Slider velocidad de referencia	E/cv_ref/valor
Pestaña controlador distancia	E/cd_p/valor
Pestaña controlador distancia	E/cd_i/valor
Pestaña controlador distancia	E/cd_d/valor
Slider velocidad de referencia	E/cd_ref/valor
Botón calibrar	E/calibrar/valor
Switch encender/apagar	E/parar/valor

Con la implementación de botones, entradas y sliders que permitieran cambiar las referencias se completa el hito de lograr una configuración de parámetros remota y centralizada a través de una interfaz fácil de operar, por lo que corresponde abordar el monitoreo de señales en línea.

Dado que ya se contaba con un código que permite leer y graficar la velocidad del líder en línea a través de la implementación de hilos con la librería *Threading*, se procede a extender la funcionalidad para monitorear también las señales.

3.2.1. Alcances de la interfaz desarrollada

La interfaz desarrollada contempla los siguientes alcances en su funcionamiento:

- Se pueden configurar hasta 10 vehículos RUPU, y se asegura su uso efectivo a través de pruebas para 4 vehículos, donde los límites de las referencia de velocidad se encuentra definido en 30 [cm/s] como se indica en el artículo [22], mientras que el máximo de la distancia de velocidad se encuentra en 30 [cm] para evitar que los agentes pierdan de vista a su predecesor.
- La topología de comunicación implementada es de tipo predecesor-following, la cual, de querer modificarse, debe ser a través del código que se carga a cada uno de los agentes, por lo que su modificación es independiente del uso de la interfaz e implicaría modificaciones en ella, pues se encuentra ajustada a la topología actual de comunicación.
- La interfaz permite graficar en línea las señales de velocidad, distancia e inclinación instantánea, en el caso particular de la distancia, esta medición se encuentra alterada por las zonas de curvas y esto se compensa a través de un algoritmo de post-procesamiento que estima la distancia entre los vehículos. Dicho algoritmo no se encuentra incorporado en la interfaz, por lo que se realiza de forma *off-line* una vez finalizados los experimentos.

- En términos del controlador, actualmente la interfaz se encuentra ajustada al controlador PID que contempla el diseño actual de RUPU, un cambio en dicho controlador, implica también que deben agregarse nuevas instrucciones de configuración de parámetros, y por lo tanto, nuevas funcionalidades y opciones de configuración a la interfaz. Dicho controlador también cuenta con parámetros de ajuste de saturación, el cual no puede variado a través de la interfaz ya que no cuenta con instrucciones listadas en el formato de la Tabla 2.1 que sean decodificables por los agentes y esto corresponde a una modificación que escapa a los alcances de este trabajo de memoria.

Todos estos puntos no resultan un impedimento para el uso de la interfaz, pero si permiten comprender los alcances de la misma. Por otro lado, pueden servir de referencia como trabajos futuros que le entreguen mayor versatilidad a la plataforma.

3.3. Documentación y reproducibilidad

De forma paralela e imprescindible para la implementación de una interfaz gráfica, se busca resolver la dificultad para utilizar la plataforma, pues, hasta antes de realizarse este trabajo de memoria, la plataforma podía ser operada exclusivamente por la persona que la desarrolló y realizó los experimentos inicialmente, ya que no se contaba con ninguna documentación con las indicaciones de uso que dieran información respecto a los códigos que se deben cargar en los agentes, en qué orden, qué parámetros configurar o si se requería de alguna herramienta adicional para complementar su uso. Dada esta dependencia plataforma, fue necesaria la realización de una demostración por parte del creador de la plataforma para documentar los pasos a seguir al momento de operar con RUPU en su versión ESP32.

De esta sesión demostrativa, se consolidó la documentación en HTML de la plataforma en tres etapas fundamentales disponibles en la carpeta “Página Web” del repositorio GIT [34], la cual se encuentra autocontenida y comentada en su respectivo código.

3.3.1. Entorno de trabajo

En primer lugar se documenta el entorno de trabajo, lo que corresponde a los software necesarios y las librerías con sus correspondientes versiones de instalación resumidas en la Tabla 3.2, es importante destacar que no utilizar las versiones correctas, o bien, librerías de autores diferentes a los indicados a continuación comprometen el funcionamiento del código, ya que, antes de instalar las versiones correctas, la incompatibilidad de las librerías impedía que los robots implementaran correctamente los algoritmos de control, imposibilitando el uso de la plataforma, e incluso impidiendo cargar el código en los robots.

Toda las instrucciones documentadas contemplan la creación de videos ilustrativos que facilitan la implementación de los pasos a seguir y se encuentran incorporados a la página web en formato .GIF como se muestra en la Figura 3.4.

Tabla 3.2: Versiones de instalación Arduino.

Arduino		
Librería	**Creador**	**Version**
VL53L0X	Pololu	1.3.1
analogWrite	ErroPix	0.1
Encoder	Paul Stoffrengen	1.4.2
Board Manager	**Creador**	**Versión**
esp32	Espressif Systems	1.0.4

Tabla 3.3: Versiones de instalación Python.

Python	
Nombre	**Versión**
customtkinter	5.2.0
tkinter	8.6
matplotlib	2.2.9
paho.mqtt	1.6.1
threading	3.8

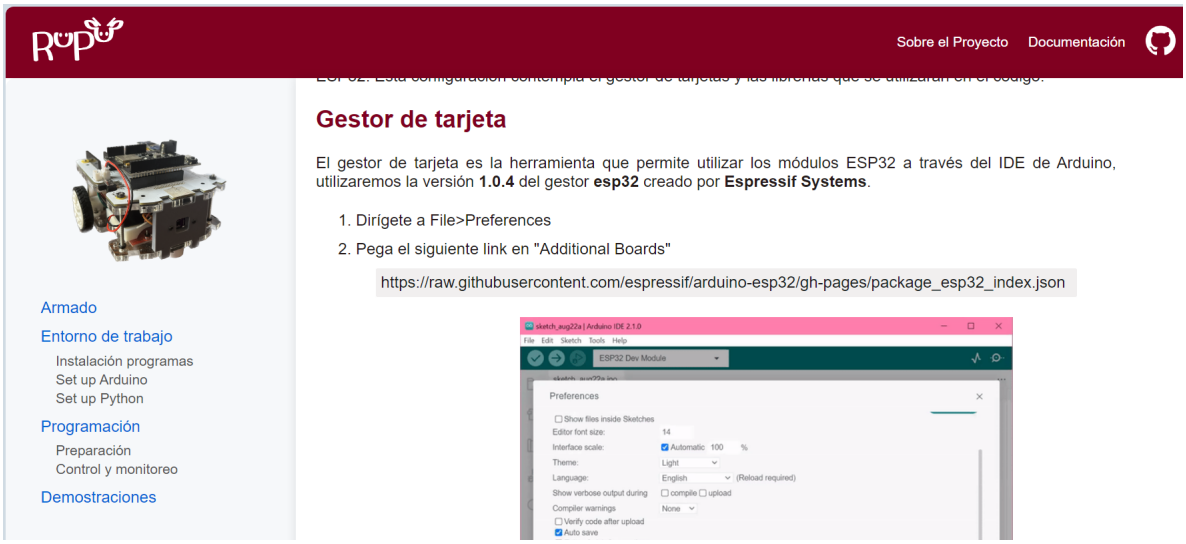


Figura 3.4: Documentación Entorno de Trabajo.

3.3.2. Programación

La segunda etapa de la documentación contempla las indicaciones a seguir para poner en marcha los agentes, dicha etapa se documenta en la página "Programación". Esta sección se divide en dos partes. La primera se centra en la preparación inicial de los vehículos, esto implica editar y subir los códigos para cada robot, en qué orden hacerlo y qué detalles se deben tener en cuenta.

```
1 #define ssid "NombreRedWifi"
2 #define password "Password"
```

```
3
4  #define IP_monitoreo "192.168.1.101"  //ip del monitor/GUI
5  #define puerto_monitoreo 1234         //puerto de envio UDP monitoreo
6  #define IP_sucesor "192.168.1.106"    //ip del robot sucesor
7  #define puerto_sucesor 1111           //puerto del robot sucesor
8  #define puerto_local 1111              //puerto local
9
10 #define EtiquetaRobot "L"   // "L": Robot lider, "S": robot2, "T": Robot3
```

Listing 3.1: Configuración control_curvatura.ino

La segunda parte se centra en describir el uso de la interfaz gráfica, cómo inicializarla, qué parámetros entregarle y cómo funciona cada elemento dispuesto en ella, tal como se muestra en la Figura 3.5

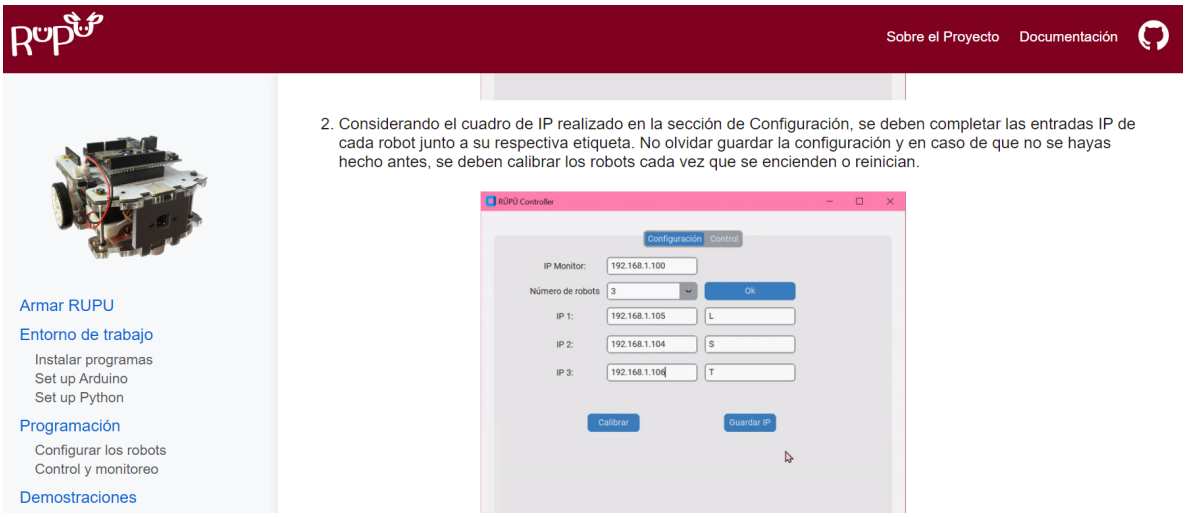


Figura 3.5: Documentación programación.

3.3.3. Demostraciones

La última sección de la documentación contempla los códigos demostrativos preparados para la plataforma con sus respectivos videos. Estas demostraciones permiten validar el correcto funcionamiento de la plataforma utilizando la interfaz gráfica desarrollada y los procedimientos descritos en el manual de uso.

Figura 3.6: Documentación Demostraciones.

Finalmente, para concluir este capítulo, es necesario destacar el apoyo de los demás integrantes del proyecto, quienes fueron los encargados de validar la información documentada y el funcionamiento de la interfaz de usuario a través del uso y experimentación con la plataforma, cuyos resultados se encuentran en el Capítulo 4.

4 | Experimentos y resultados

El presente capítulo se enfoca en detallar las pruebas experimentales realizadas con la plataforma para validar a nivel de sistema su funcionamiento e implementar pruebas enfocadas en verificar si la plataforma permite realizar experimentos repetibles y reproducibles, ya que como se mencionó en la introducción, a nivel de agente, cada uno de los componentes se encuentra probado en reportes anteriores como en [21] y [22], donde se valida el funcionamiento de cada vehículo.

En particular, los experimentos reportados en [22] servirán de base para realizar las validaciones de repetibilidad y reproducibilidad de la plataforma RUPU, las cuales se basarán en la configuración experimental reportada en el artículo e incorporarán el análisis de métricas de error para diferentes iteraciones de cada prueba, con el fin de obtener conclusiones que entreguen información sobre la posibilidad de repetir y reproducir los experimentos y, en caso de ser posible, precisar el error asociado a cada una de estas mediciones.

4.1. Validación de funcionamiento

El experimento llevado a cabo considera un pelotón de tres vehículos que se desplazan a lo largo del circuito cerrado, compuesto por tramos rectos y curvas, como se ilustra en la Figura 4.1. En esta configuración, se establece la presencia de un vehículo líder y dos vehículos seguidores que avanzan siguiendo la trayectoria marcada por la línea blanca en el circuito. Es importante destacar que, previo al inicio de los experimentos, cada vehículo ejecuta una secuencia de calibración que lo posiciona con su eje central alineado con una sección recta de la línea blanca y a una distancia inicial del vehículo predecesor, este procedimiento se encuentra documentado en [22]. Por último, una de las principales condiciones a cumplirse para la realización de los experimentos, es la carga adecuada de los robots antes de ejecutar cualquier código, ya que un agente con batería baja puede distorsionar todos los datos obtenidos a nivel sistema, debido a un accionamiento deficiente que perjudique la velocidad, mediciones erradas o, incluso, problemas de recepción y envío de mensajes UDP.

Para la realización del experimento la referencia de distancia configurada entre cada seguidor y su predecesor es de 10 [cm]. Por su parte, la referencia de velocidad se cambia cada 30 [s] iniciando en 15 [cm/s], luego 25 [cm/s] y finalmente 10[cm/s], completando cada experimento en 90 [s]. El registro audiovisual del experimento se encuentra disponible en [37], y los detalles de resultados se describen en el resto de esta sección.

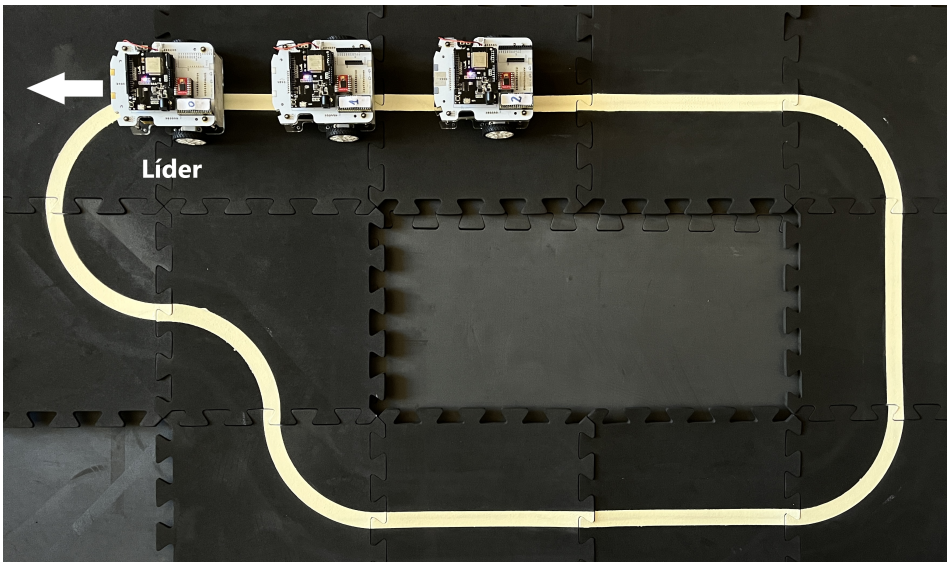


Figura 4.1: Configuración experimental para pelotones de seguimiento de línea en un camino cerrado.

Para obtener mediciones consistentes, fue necesario estandarizar las condiciones ambientales de cada prueba, para esto, se acondicionó una habitación con iluminación natural filtrada a través de cortinas, la intensión de las cortinas era evitar la luz directa que pudiera proyectar la sombra de los robots en pista perturbando la calibración. La luz tampoco podía provenir de una luz artificial directa sobre la pista, ya que esta se refleja sobre las placas de los robots, perjudicando el post procesamiento de los videos que permiten conocer la distancia estimada entre los agente.

Estos videos que fueron posteriormente analizados y procesados por medio de un algoritmo que utiliza como referencia un punto de la placa para estimar la distancia, que para el caso de este experimento dicho punto era un LED, fueron grabados con un Iphone 13, las ventajas de esta grabación, respecto a las grabaciones con una GoPro Hero 11 son principalmente los ajustes de color, que permiten una mejor captura del LED de la placa, la imagen no necesita ser grabada desde lejos, dado que el lente principal del teléfono no genera distorsión de gran angular y finalmente, la menor cantidad de frames capturados, agiliza el procesamiento del video sin perjudicar los resultados de la medición.

Por el lado de los vehículos, es un requerimiento fundamental que ellos se encuentren cargados antes de realizar los experimentos, la falta de carga altera la velocidad de los agentes y su capacidad de envío y recepción de información perjudicando el desempeño general de la plataforma. Otro punto a considerar es que se encuentren correctamente calibrados los sensores de distancia, lo cual puede ser verificado por medio de los códigos de prueba provistos en el repositorio de RUPU [25].

Una vez aclaradas las condiciones y requerimientos sobre los que se desarrollan los siguientes experimentos, se proceden a presentar los resultados experimentales que den cuenta de la validación funcional de la plataforma RUPU, comenzando por la Figura 4.2, la cual muestra las mediciones de velocidad instantánea para cada vehículo en el pelotón junto con su referencia a lo largo del experimento, estas mediciones fueron recibidas mientras los agentes ejecutaban su rutina en el circuito y guardadas en un archivo .csv, el que fue posteriormente graficado

como se muestra en la Figura 4.2, los datos capturados para este gráfico son recibidos por el computador desde el cual se ejecuta la secuencia de instrucciones de cambio de velocidad en un ciclo asociado a un hilo que se ejecuta paralelamente al envío de instrucciones.

Se debe notar que cada vehículo sigue una referencia de velocidad interna, generada a partir de las estimaciones de velocidad de su predecesor que son recibidas por comunicación inalámbrica. El objetivo de esta configuración es que cada vehículo pueda limitar su velocidad en los instantes en que las mediciones de distancia obtenidas a partir del sensor local son erráticas debido a que el predecesor se pierde de la línea vista, lo cual ocurre en las regiones curvas.

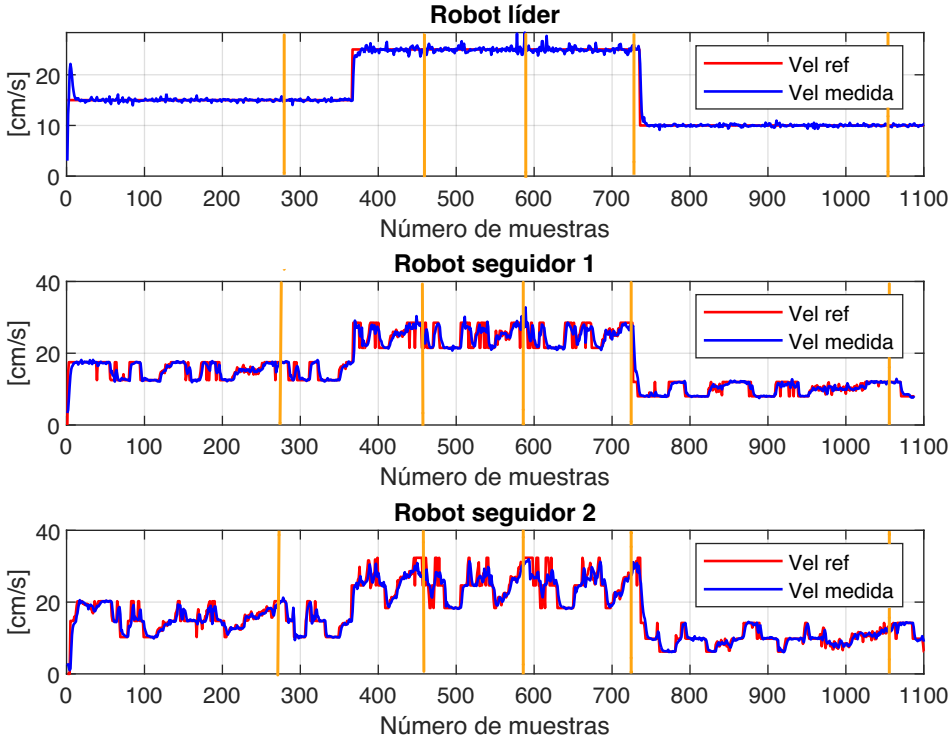


Figura 4.2: Velocidad instantánea y referencia.

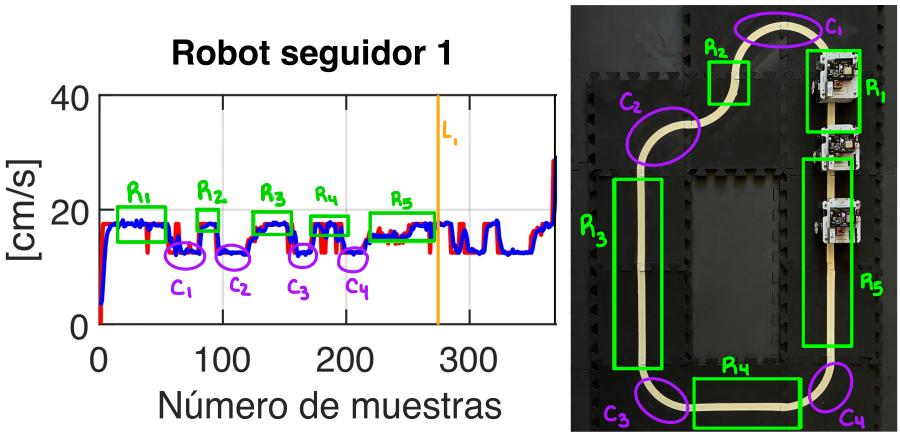


Figura 4.3: Velocidad instantánea y referencia.

La Figura 4.4 muestra las mediciones de distancia obtenidas del sensor local de cada vehículo seguidor con respecto a su predecesor, donde se encuentran marcadas verticalmente y de color naranjo los momentos en que el agente completa una vuelta completa de medición. Para dar claridad respecto a los gráficos de velocidad y su relación con las zonas curvas y rectas de la pista, se presenta la Figura 4.3, la cual corresponde a un *zoom* a la Figura 4.4 junto a la pista en donde se marcan las zonas curvas en morado y las rectas en verde respecto a la primera vuelta que dan los agentes tanto en la imagen de la pista como en la gráfica de velocidad.

En las secciones curvas, cada vehículo seguidor satura su velocidad en base a la velocidad recibida desde el predecesor para evitar potenciales colisiones, retomando la distancia con respecto al predecesor en cuanto se alcance nuevamente una zona recta del camino. Para analizar el comportamiento del pelotón en las zonas en que los sensores no son confiables, la plataforma incluye una herramienta de procesamiento de video que permite estimar la verdadera distancia entre vehículos durante todo el experimento usando técnicas de visión computacional [24], lo cual facilita el posterior análisis del desempeño del algoritmo de control y la verificación de comportamientos de interés, como la presencia o ausencia de colisiones. Es importante enfatizar que esta estimación solo se obtiene luego de ejecutado el experimento, y no se usa para efectos de control. La Figura 4.5 muestra la estimación de la distancia utilizando procesamiento de video, donde se pueden observar intervalos en que la distancia estimada se desvía del valor de referencia, lo cual se asocia a las zonas curvas y se considera un comportamiento esperado dado que el controlador fue diseñado para utilizar la velocidad recibida desde el predecesor y saturar la velocidad en estas condiciones, reduciendo la posibilidad de colisiones con respecto al comportamiento de un algoritmo no cooperativo [24].

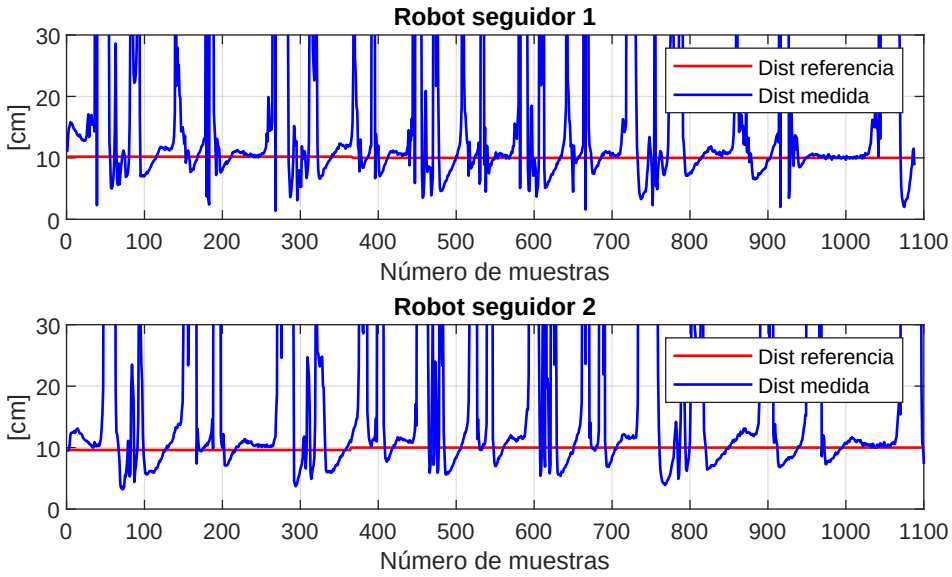


Figura 4.4: Medición de distancia de los sensores locales.

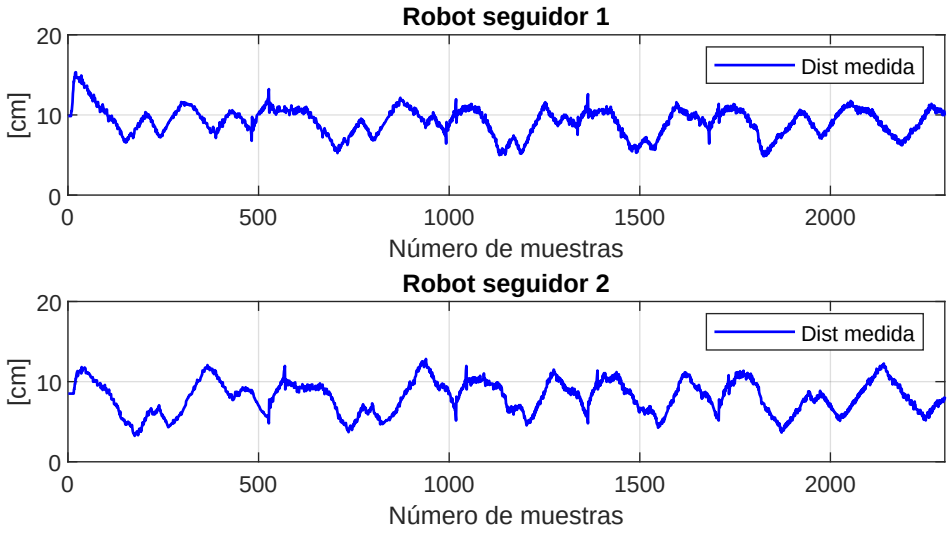


Figura 4.5: Medición de distancia medida a través de procesamiento de video.

La realización de estas mediciones, permiten validar el correcto funcionamiento de la plataforma a nivel de sistema multi-agentes y permite trabajar sobre esta configuración en la ejecución de los experimentos que permitirán validar la repetibildad y reproducibilidad de RUPU.

4.2. Repetibilidad

La definición de repetibilidad indica que los experimentos realizados por un mismo grupo de personas con un misma configuración experimental deben presentar resultados equivalentes bajo cierto grado de precisión. Según esta definición, se repite el experimento de validación realizado en tres ocasiones y se presentan las mediciones de velocidad instantánea del robot seguidor uno en la Figura 4.6.

Se busca contrastar la tendencia de las señales medidas en cada iteración del experimento con el objetivo de verificar la repetibilidad del mismo. Lo primero a notar es la tendencia similar de cada una de las señales, aumentando y disminuyendo su magnitud en las mismas proporciones, sin embargo, esto se ve afectado por un desfase temporal esperable producido por la captura de datos. Este desfase temporal induce alteraciones en las métricas de error que permiten obtener conclusiones sobre la confiabilidad de los experimentos realizados y por lo tanto es necesario implementar estrategias computacionales que mitiguen su efecto.

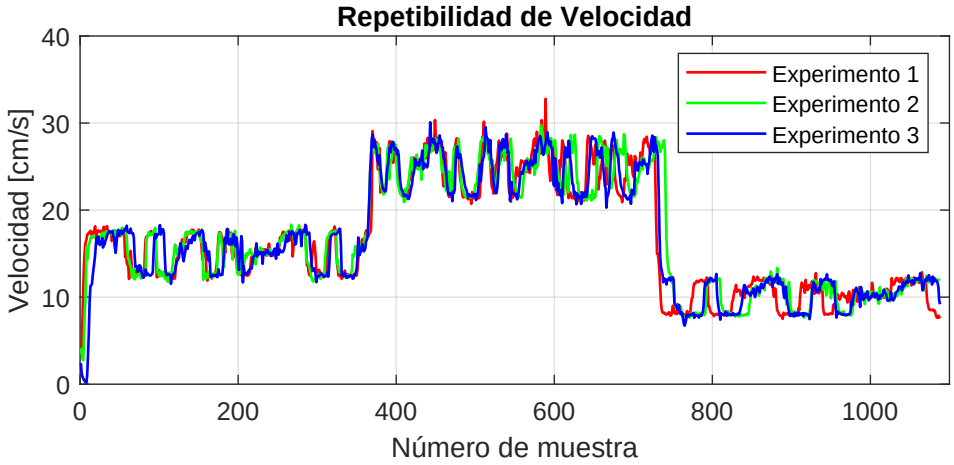


Figura 4.6: Medición instantánea de velocidad del seguidor 1.

Para contrarrestar el efecto del desfase temporal en las mediciones, se utilizó el algoritmo *Dynamic Time Warping*, el cual busca la alineación óptima entre dos señales a través de la minimización de la suma de distancias euclidianas. La Figura 4.7, muestra la aplicación de dicho algoritmo contrastando las iteraciones 2 y 3 con el experimento base y obteniendo las gráficas con las tendencias ajustadas, quitando el efecto del desfase temporal.

A partir de los vectores proporcionados por el algoritmo, es posible calcular las métricas de error de las señales alineadas, cada una de estas se obtiene de comparar el experimento base con sus iteraciones siguientes, las cuales se presentan en la Tabla 4.1 y se definen a continuación.

- **Error cuadrático medio (ECM):** Representa el promedio de los errores al cuadrado y se obtiene de sumar la resta los valores del experimento base y_i con la siguiente iteración $\hat{y}_i$, llevarla al cuadrado y dividirla por el número de muestras n.

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- **Raíz del error cuadrático medio (RECM):** Se obtiene de aplicar la raíz cuadrada al error cuadrático medio.

$$RECM = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- **Raíz del error cuadrático medio normalizado**: Esta métrica se obtiene dividiendo el valor RECM en el promedio de la señal original del experimento base N.

$$RECMNormalizado = \frac{1}{N}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

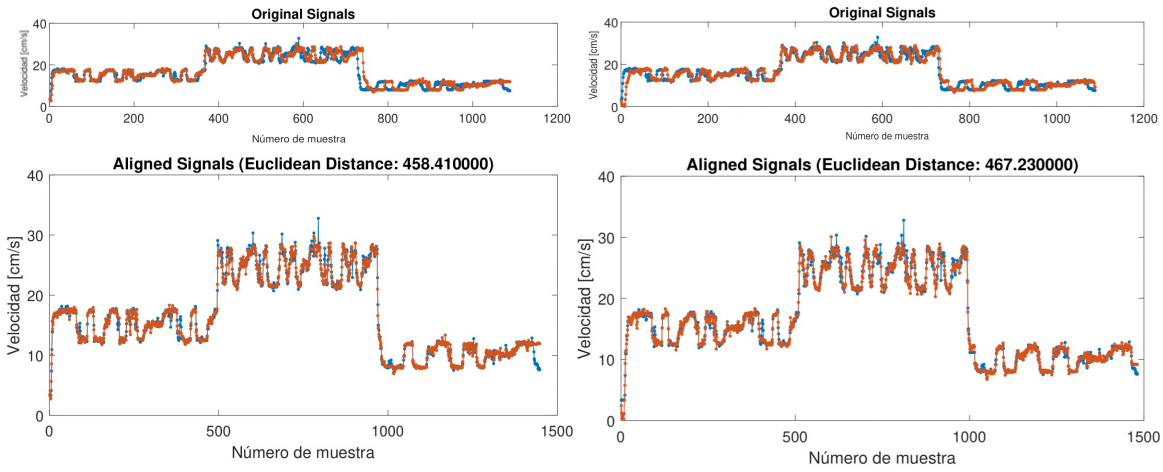


Figura 4.7: Dynamic time warping para iteraciones 2 y 3 respecto a la base.

Tabla 4.1: Error experimento de repetibilidad.

Métricas	**Experimento 1-2**	**Experimento 1-3**
Suma Dist. Eucl.	458.41	467.23
ECM	0.37643	0.30826
RECM	0.61354	0.55522
RECM Normalizado	3.67 %	3.32 %

En la Tabla 4.1 se presentan las métricas de error anteriormente descritas junto a la suma de distancias euclidianas que resulta de la comparación entre el experimento base y las iteraciones 2 y 3, la suma de distancias euclidianas es el valor que minimiza el algoritmo de DTW para optimizar el ajuste de curva, por lo que entrega información respecto al desfase de la función y no corresponde a un valor que guarde relación directa con las métricas de error definidas anteriormente y sobre las cuales se realiza el análisis de resultados.

Por último, al proceder con la obtención de cada una de las métricas de error, es posible concluir que la repetibilidad se alcanza con una precisión menor al 4 %, lo que resulta de dividir cada una de las raíces el ECM, por el promedio de la señal del experimento base, el cual es de 16.706 [cm/s]. Con esta información establecida, se procede a realizar el siguiente experimento a comprobar la reproducibilidad de la plataforma.

4.3. Reproducibilidad

A diferencia de la repetibilidad, para validar la reproducibiliadad de un experimento, es necesario contar con un equipo diferente de personas que utilicen la misma configuración experimental. Para esto, se realiza una secuencia de comandos por parte de quien desarrolla esta memoria registrándose los datos, y luego, otro miembro del equipo, desde su computador personal descarga dicho código para ejecutarlo él mismo. Para validar la reproducibilidad del experimentos, ambas gráficas deben mostrar una tendencia claramente similar.

La Figura 4.8 muestra la comparativa entre el experimento base y los experimentos realizados por la segunda persona. Al igual que en el caso presentado anteriormente, se observa similitud entre la forma de las señales, pero obtener métricas de error directamente desde los datos inducirá a conclusiones alteradas producidas por el desfase esperable entre las señales.

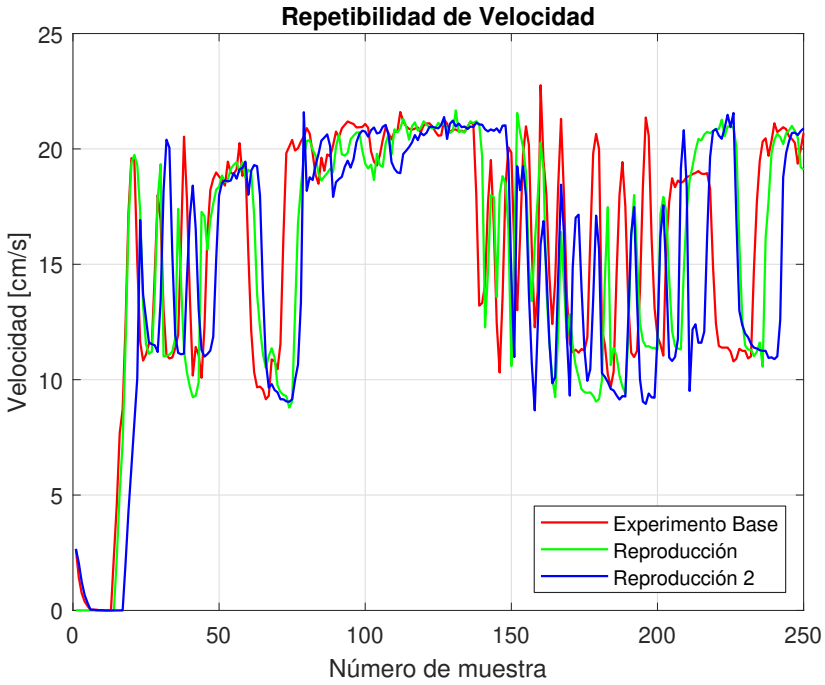


Figura 4.8: Experimento de reproducibilidad.

Nuevamente se aplica el algoritmo *Dynamic Time Warping* para ignorar el efecto del desfase en el análisis del error entre las mediciones, presentándose la alineación de las señales en las Figuras 4.9 y 4.10. En la Tabla 4.2 se presentan las mismas métricas definidas para el experimento de repetibilidad para las iteraciones 2 y 3 realizadas por el otro miembro del equipo.

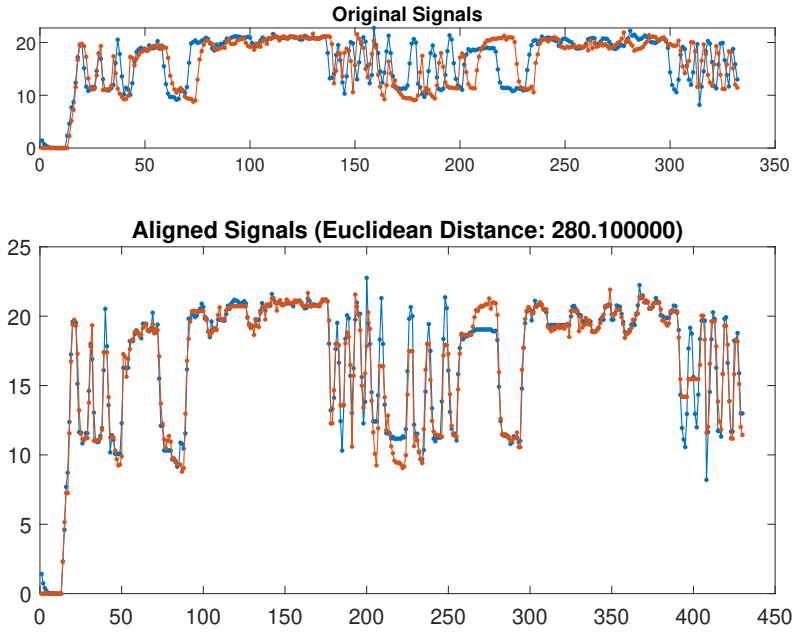


Figura 4.9: Dynamic time warping en reproducción 1.

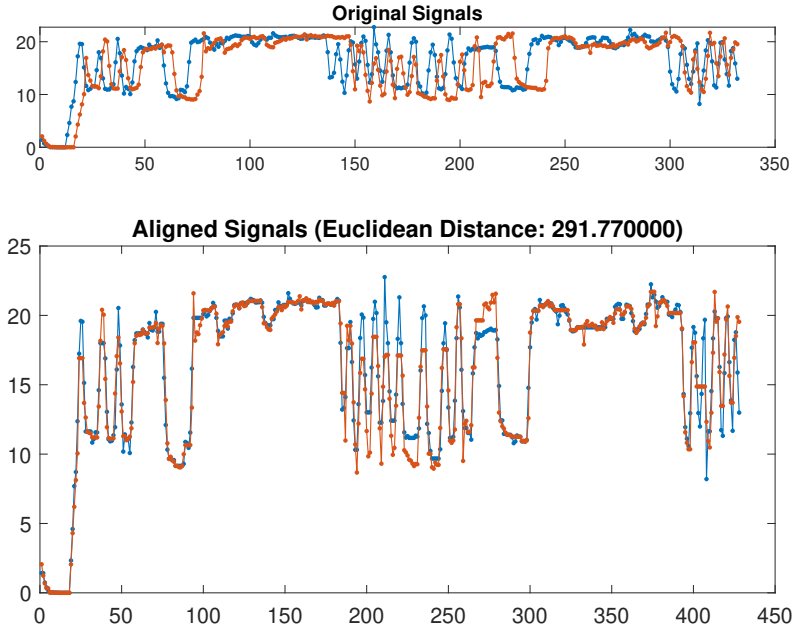


Figura 4.10: Dynamic time warping en reproducción 2.

Tabla 4.2: Error experimento de reproducibilidad.

Métricas	**Experimento 1-2**	**Experimento 1-3**
Suma Dist. Eucl.	282.76	291.74
ECM	1.1338	1.343
RECM	1.0648	1.1589
RECM Normalizado	6.47 %	7.05 %

En este experimento, las métricas de error presentan una magnitud mayor a las del experimento de repetibilidad, pero manteniéndose por debajo del 8 % de error normalizado, siendo el promedio de la señal base igual a 16.448 [cm/s], el leve aumento en la magnitud del error puede atribuirse a la posición de los robots en la pista que puede haber sido similar, mas no exactamente la misma realizada en el experimento base, las conidiciones climáticas que alteran la iluminación de la habitación o incluso la capacidad del equipo personal con el cual se ejecutó el código por parte del miembro del equipo que dio soporte a este experimento.

Aún con esta variación, se concluye que la plataforma es reproducible con una precisión menor al 8 %, lo cual entrega confiabilidad a RUPU en el reporte de sus experimentos debidamente documentados.

5 | Conclusiones y trabajo futuro

5.1. Conclusiones

Este trabajo de memoria ha tenido dos ejes principales de desarrollo, el primero de ellos corresponde a mejorar la usabilidad de la plataforma RUPU, implementando una interfaz que agiliza el proceso de configuración de parámetros y visualización de señales con fines demostrativos. De este eje, se puede concluir que la interfaz de usuario implementada permite incorporar las funcionalidades creadas para RUPU en [21] de manera intuitiva para el usuario, entregando soporte a las necesidades de configuración y monitoreo de los agentes. La instalación y correcto funcionamiento de la interfaz fue comprobado por diferentes miembros del equipo tanto para computador con sistema operativo Windows como Linux.

Respecto a los experimentos de repetiblidad, y en base a las definiciones propuestas por la *Association for Computing Machinery* (ACM) para la revisión de artefactos, el criterio de repetibilidad establece que las mediciones deben ser obtenidas con cierto grado de precisión para repetidas ejecuciones del experimento realizadas por lo mismos investigadores, usando los mismos procedimientos, instrumentos de medición, y condiciones operacionales [23]. Dada la naturaleza de la plataforma RUPU y su orientación como plataforma académica y educativa, se establece la repetibilidad como un requisito para incentivar el análisis y escrutinio de resultados por parte de otros investigadores.

Para verificar la repetibilidad, se realizaron tres repeticiones del experimento reportado en la sección anterior y se analizaron los registros de las mediciones obtenidas en cada caso. La Figura 4.6 muestra los gráficos de las mediciones de velocidad para el primer seguidor en la primera repetición del experimento completo. Se puede observar que las mediciones son consistentes entre repeticiones del experimento, con algunas desviaciones temporales pero que siguen esencialmente las mismas tendencias y magnitudes. Dada esta correlación en los datos, se puede considerar que la plataforma permite obtener resultados consistentes en todas las iteraciones realizadas, validando la repetibilidad del experimento con una precisión menor al %4.

Por otro lado, para validar la reproducibilidad de los experimentos se establece que las mediciones deben ser obtenidas con cierto grado de precisión para repetidas ejecuciones del experimento realizadas por diferentes investigadores, usando los mismos procedimientos, instrumentos de medición, y condiciones operacionales [23].

Para verificar este ítem, se contó con el apoyo de otro miembro del equipo, quien sin haber utilizado previamente la interfaz gráfica o ejecutado los códigos demostrativos para la plataforma, fue capa de obtener resultados similares a los obtenidos en iteraciones anteriores por quien

presenta esta memoria en dos iteraciones del experimento, dichos resultados se encuentran expuestos en la Figura 4.8, donde la clara tendencia junto a las métricas de error que entregan resultados inferiores al %8, permiten validar la reproducibilidad del experimento.

5.2. Recomendaciones para trabajo futuro

Ciertamente, el desarrollo de RUPU se encuentra en etapas de maduración y aún no está cerca de ser una plataforma comercial como lo es Duckietown, sin embargo, su potencial como plataforma experimental incentiva a seguir desarrollándola, por esta razón y a raíz del trabajo realizado en esta memoria, se sugieren las siguientes tareas a realizar.

- **Documentar el proceso de ensamblado de RUPU:** parte importante del proceso de replicabilidad requiere que los vehículos se puedan armar por otra personas, para que sea probado con un dispositivo diferente y para ello se requiere documentación precisa y detalla. Para esto se dejó una sección en la documentación para ser completada con esta información.
- **Eliminar la dependencia a Ubidots en el código de Raspberry Pi:** la versión adaptada en Python del código principal se encuentra enlazada a una cuenta Ubidots con la cual se monitorean las señales y decodifica las instrucciones recibidas, esto implica que no es posible ejecutar los comandos de instrucción sin vincularlo a esa cuenta, impidiendo la compatibilidad con la interfaz gráfica.
- **Replicar la documentación para Raspberry Pi:** una vez se hayan resuelto las tareas anteriores, es necesario generar una documentación que de cuenta del paso a paso de configuración de la plataforma desde su ensamblado hasta las demostraciones que validen su funcionamiento, tal como se ha hecho con la versión en ESP32.
- **Estandarizar el controlador:** la investigación de algoritmos de control es la motivación principal para la creación de la plataforma, y si bien, es cierto que se han logrado esquemas de control robustos bajo ciertas condiciones de operación, los cambios de iluminación, la elección de la pista u otras perturbaciones pueden causar que los experimentos no puedan ser reproducidos o incluso repetidos, por esta razón, es necesario investigar y documentar minuciosamente el desempeño del controlador y los parámetros escogidos realizando reiteradas pruebas.

Como última observación, todo el material desarrollado en esta memoria puede y debe ser actualizado con el fin de mantener vigente la información sobre la plataforma motivando a nuevos estudiantes a formar parte de este proyecto.

Anexos

La presente sección contempla la descripción y resultados del experimento preliminares denominado "Test de estrés", que consiste en aumentar progresivamente el envió de secuencias de instrucciones a los vehículos para forzar su desempeño al límite y conocer las limitaciones de la plataforma.

5.3. Test de estrés

En este experimento se buscó medir la repetibilidad sometiendo a los vehículos a una prueba de estrés, este experimento consistió en realizar 3 experimentos enviando comandos de configuración para las referencias de velocidad cada un 0.5, 0.25 y 0.125 [s] verificando si la plataforma es capaz de responder de forma consistente a tales cambios de referencia. Como resultado de este experimento se obtuvo que los robots no son capaces de procesar instrucciones cada 0.125 [s] deteniendo su ejecución.

A pesar de ello, los agentes sí son capaces de procesar los comandos cada 0.25 [s], los resultados de esta medición se encuentran documentados en el Capítulo 5.2 y de ellos se extrae que al enviar comandos con tan alta frecuencia las mediciones de repetibilidad deben ser sometidas al *Dynamic time warping* para observarse las tendencias con claridad.

Los siguientes gráficos corresponden a elementos que complementan el Capítulo 4 y cada uno de ellos entrega la siguiente información:

- La Figura 5.1 presenta la velocidad del robot líder y sus seguidores en el experimento base realizado enviando instrucciones cada 0.5 segundos y se utiliza para validar la correcta captura de datos.
- La Figura 5.2 muestra la velocidad instantánea de todos los experimentos realizados utilizando la frecuencia de envío de instrucciones cada 0.5 segundos, donde el propósito es determinar si se observan similitudes en las tendencias del gráfico.
- Las Figuras 5.3, 5.4, 5.5 presentan la alineación del experimento base y sus siguientes iteraciones a través del algoritmo *Dynamic Time Warping*.
- La Figura 5.6 presenta la velocidad del robot líder y sus seguidores en el experimento base realizado enviando instrucciones cada 0.25 segundos y se utiliza para validar la correcta captura de datos y funcionamiento de este experimento.
- La Figura 5.7 presenta la alineación de las señales por medio del algoritmo *Dynamic Time Warping* para poder evitar el efecto del desfase en las métricas de error que se

puedan extraer.

Si bien este experimento no fue completado en su totalidad para presentar un análisis detallado de sus resultados, puede servir como precedente para algún trabajo futuro que busque medir con precisión el desempeño de la plataforma en situaciones que involucren un alto flujo de instrucciones.

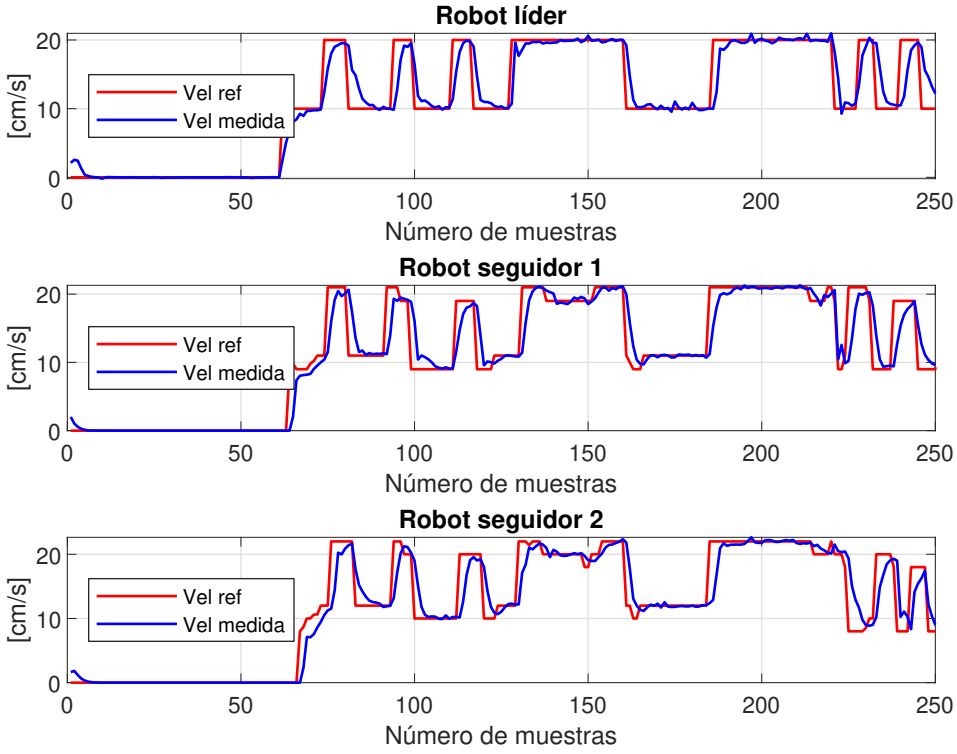


Figura 5.1: Velocidad instantánea experimento base con cambios mínimos de 0.5 segundos.

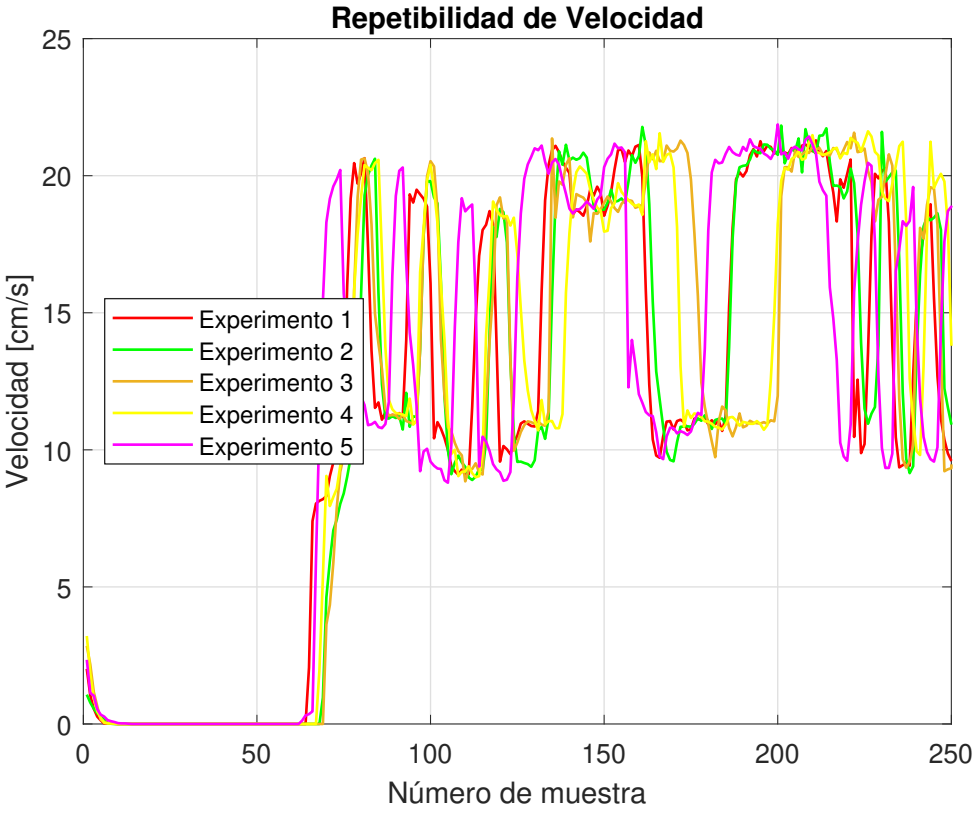


Figura 5.2: Repetibilidad con cambios mínimos de 0.5 segundos.

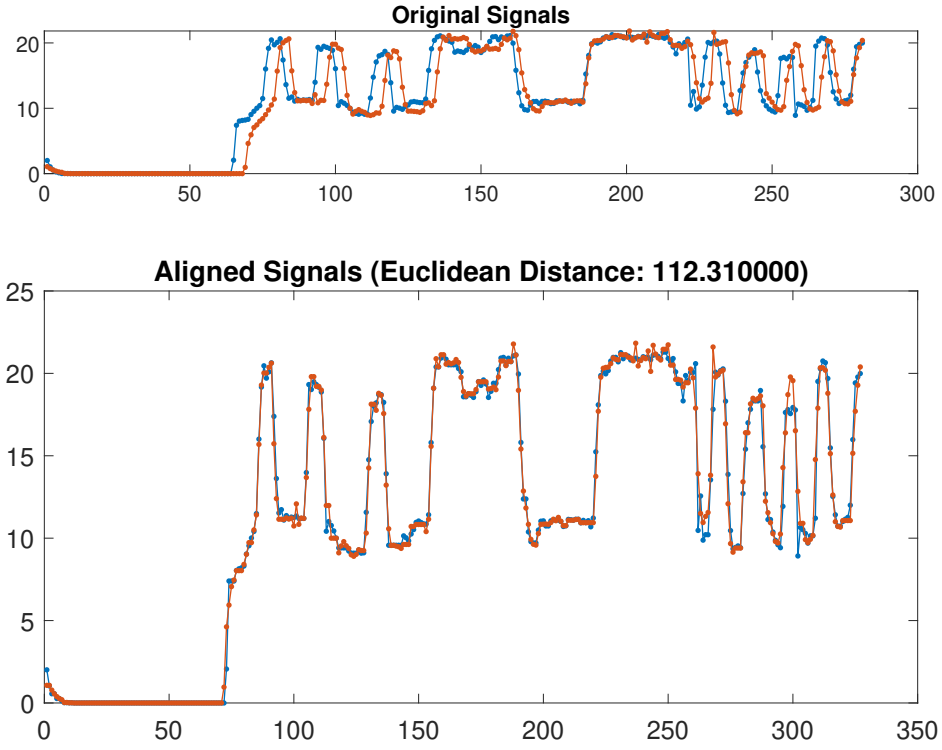


Figura 5.3: Dynamic time warping iteración 2 con cambios mínimos de 0.5 segundos.

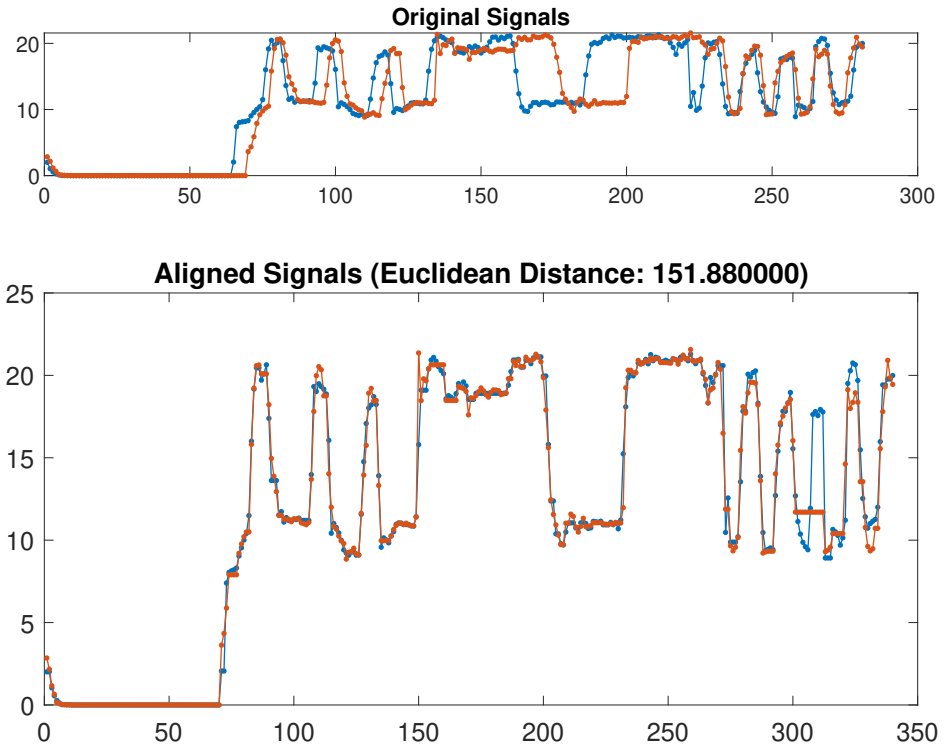


Figura 5.4: Dynamic time warping iteración 3 con cambios mínimos de 0.5 segundos.

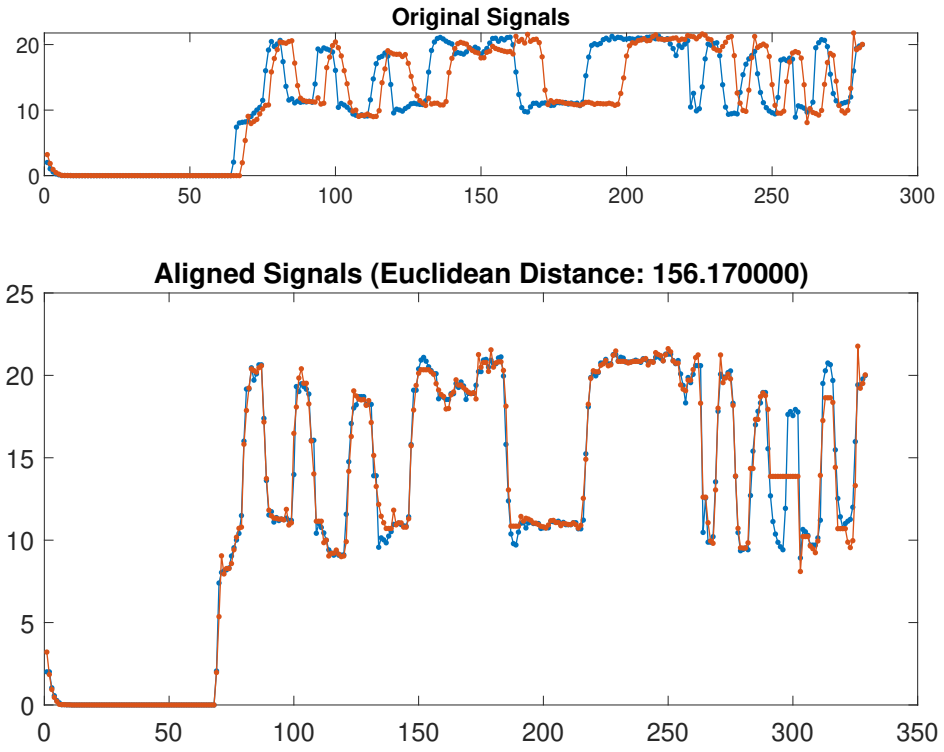


Figura 5.5: Dynamic time warping iteración 4 con cambios mínimos de 0.5 segundos.

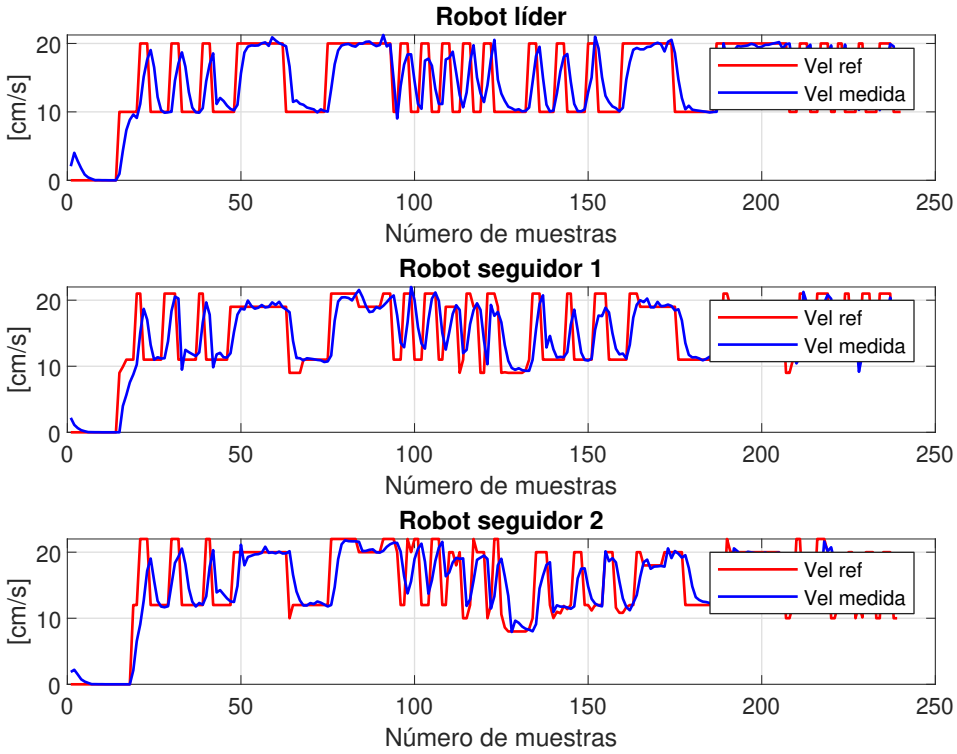


Figura 5.6: Velocidad instantánea con cambios mínimos de 0.25 segundos.

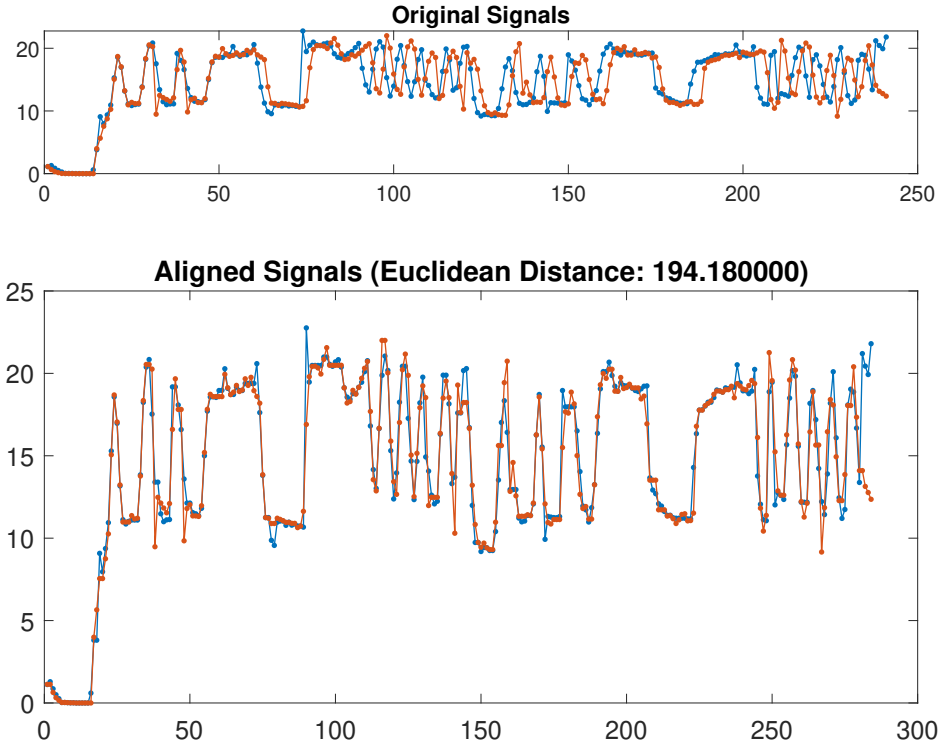


Figura 5.7: Dynamic time warping iteración 2 con cambios mínimos de 0.25 segundos.

Bibliografía

[1] J. Guerrero-Ibáñez, S. Zeadally, and J. Contreras-Castillo, "Sensor technologies for intelligent transportation systems," *Sensors*, vol. 18, no. 4, p. 1212, 2018.

[2] K. C. Dey, L. Yan, X. Wang, Y. Wang, H. Shen, M. Chowdhury, L. Yu, C. Qiu, and V. Soundararaj, "A review of communication, driver characteristics, and controls aspects of cooperative adaptive cruise control (cacc)," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 491–509, 2015.

[3] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, "Three decades of driver assistance systems: Review and future perspectives," *IEEE Intelligent transportation systems magazine*, vol. 6, no. 4, pp. 6–22, 2014.

[4] V. Turri, B. Besselink, and K. H. Johansson, "Cooperative look-ahead control for fuel-efficient and safe heavy-duty vehicle platooning," *IEEE Trans. on Control Systems Technology*, vol. 25, no. 1, pp. 12–28, 2016.

[5] L. Xu, L. Y. Wang, G. Yin, and H. Zhang, "Communication information structures and contents for enhanced safety of highway vehicle platoons," *IEEE Trans. on Vehicular Technology*, vol. 63, no. 9, pp. 4206–4220, 2014.

[6] S. Thormann, A. Schirrer, and S. Jakubek, "Safe and efficient cooperative platooning," *IEEE Trans. on Intelligent Transportation Systems*, 2020.

[7] C. Bergenhem, S. Shladover, E. Coelingh, C. Englund, and S. Tsugawa, "Overview of platooning systems," in *Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria (2012)*, 2012.

[8] C. Latrech, A. Chaibet, M. Boukhnifer, and S. Glaser, "Integrated longitudinal and lateral networked control system design for vehicle platooning," *Sensors*, vol. 18, no. 9, 2018.

[9] S. Wei, Y. Zou, X. Zhang, T. Zhang, and X. Li, "An integrated longitudinal and lateral vehicle following control system with radar and vehicle-to-vehicle communication," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1116–1127, 2019.

[10] A. Bayuwindra, J. Ploeg, E. Lefeber, and H. Nijmeijer, "Combined longitudinal and lateral control of car-like vehicle platooning with extended look-ahead," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 790–803, 2020.

[11] L. Yu, Y. Bai, Z. Kuang, C. Liu, and H. Jiao, "Intelligent bus platoon lateral and longitudinal control method based on finite-time sliding mode," *Sensors*, vol. 22, no. 9, p. 3139, 2022.

[12] S. Feng, Y. Zhang, S. E. Li, Z. Cao, H. X. Liu, and L. Li, "String stability for vehicular platoon control: Definitions and analysis methods," *Annual Reviews in Control*, vol. 47, pp. 81–97, 2019.

[13] F. A. Papoulias, "Guidance and control laws for vehicle pathkeeping along curved trajectories," *Applied ocean research*, vol. 14, no. 5, pp. 291–302, 1992.

[14] T. I. Fossen, K. Y. Pettersen, and R. Galeazzi, "Line-of-sight path following for dubins paths with adaptive sideslip compensation of drift forces," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 2, pp. 820–827, 2014.

[15] A. Roy and M. M. Noel, "Design of a high-speed line following robot that smoothly follows tight curves," *Computers & Electrical Engineering*, vol. 56, pp. 732–747, 2016.

[16] M. Velasco-Villa, R. D. Cruz-Morales, A. Rodriguez-Angeles, and C. A. Domínguez-Ortega, "Observer-based time-variant spacing policy for a platoon of non-holonomic mobile robots," *Sensors*, vol. 21, no. 11, p. 3824, 2021.

[17] "Ensemble." `https://platooningensemble.eu`.

[18] Z. Li, B. Hu, M. Li, and G. Luo, "String stability analysis for vehicle platooning under unreliable communication links with event-triggered strategy," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2152–2164, 2019.

[19] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang, *et al.*, "Duckietown: an open, inexpensive and flexible platform for autonomy education and research," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1497–1504, IEEE, 2017.

[20] R. Amsters and P. Slaets, "Turtlebot 3 as a robotics education platform," in *Robotics in Education*, (Cham), pp. 170–181, Springer International Publishing, 2020.

[21] C. Escobar, "Diseño, implementación y evaluación de una plataforma experimental para el estudio del control de una flota de robots móviles," 2021.

[22] C. Escobar, F. J. Vargas, A. A. Peters, and G. Carvajal, "A cooperative control algorithm for line and predecessor following platoons subject to unreliable distance measurements," *Mathematics*, vol. 11, no. 4, 2023.

[23] Association for Computing Machinery, "Artifact review and badging." `https://www.acm.org/publications/policies/artifact-review-badging`, Aug. 2020.

[24] C. Escobar, F. J. Vargas, A. A. Peters, and G. Carvajal, "A cooperative control algorithm for line and predecessor following platoons subject to unreliable distance measurements," *Mathematics*, vol. 11, no. 4, p. 801, 2023.

[25] "Git rupu_gui." `https://github.com/catalinachaufleur/RUPU_GUI`.

[26] P. Castillo, "Integración de sistema de visión por computador a plataforma robótica móvil con tracción diferencial," 2022.

[27] "Chilecon 2023." `https://site.ieee.org/chilesur/ieee-chilecon-2023/`.

[28] "Zumi." `https://www.robolink.com/products/zumi`.

[29] "Qcar." `https://www.quanser.com/products/qcar/`.

[30] "Donkeycar." https://www.donkeycar.com.

[31] "Duckietown history." https://www.duckietown.org/about/history.

[32] "Setup laptop." https://docs.duckietown.com/daffy/opmanual-duckiebot/setup/setup_laptop/index.html.

[33] A. A. Peters, F. J. Vargas, C. Garrido, C. Andrade, and F. Villenas, "Pl-toon: A low-cost experimental platform for teaching and research on decentralized cooperative control," *Sensors*, vol. 21, no. 6, 2021.

[34] "Git rupu_gui/página web." https://github.com/catalinachaufleur/RUPU_GUI/tree/main/PÃągina%20Web.

[35] "Matlab hardware support - raspberry." https://la.mathworks.com/hardware-support/raspberry-pi-matlab.html.

[36] "Interfaces gráficas (gui)." https://wiki.python.org.ar/interfacesgraficas/.

[37] "Rupu project." https://youtube.com/@RupuProject.