ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA - ICAI

Grado en Ingeniería en Tecnologías Industriales (GITI)

PROYECTO DE FIN DE GRADO

Navegación de Robots Móviles en Entornos Dotados de Sistemas de Localización Externos

Autor: Adriana Marroquín Rodríguez

Director: Juan Luis Zamora Macho

Madrid 2021

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Navegación de Robots Móviles en Entornos Dotados de Sistemas de Localización Externos

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico **2020/2021** es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Adriana Marroquín Fecha: 05 / 07 / 21

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Fecha: 13 / 07 / 2021

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA - ICAI

Grado en Ingeniería en Tecnologías Industriales (GITI)

PROYECTO DE FIN DE GRADO

Navegación de Robots Móviles en Entornos Dotados de Sistemas de Localización Externos

Autor: Adriana Marroquín Rodríguez
Director: Juan Luis Zamora Macho

Madrid 2021

Agradecimientos

Este trabajo de fin de grado representa el final de mi paso por la universidad. Haber llegado hasta aquí ha supuesto un reto para mí que, desde luego, no habría sido posible alcanzar sin todas aquellas personas con las que he tenido la suerte de compartir estos años y que han sido mi apoyo en los momentos más duros. Sería injusto no reconocer que mis logros son en gran parte gracias a vosotros.

En primer lugar, agradecer todo el apoyo incondicional de mis padres y hermanos, que han estado a mi lado, ayudándome y apoyándome siempre que les he necesitado. Gracias por creer en mí, incluso cuando yo dudaba de mí misma. A mi abuela Ofelia y mi abuelo Miguel, quienes siempre me han servido de ejemplo para todo en la vida.

A mi abuela Mari Carmen y a todos mis tíos, por acogerme y cuidarme durante los cuatro años de carrera. Gracias por haberme hecho sentir tan querida y apoyada. Sin vosotros, seguramente habría sido mucho más difícil.

A todos mis amigos y amigas, aquellos que empezaron a formar parte de mi vida cuando comencé la universidad, y también a todos aquellos que llevan a mi lado desde mucho antes. Especialmente a Bea, María, Laura y Marta que no han dejado de confiar en mi ni un solo momento y que han sido mi vía de escape cuando lo he necesitado. Bea, gracias por estar a mi lado durante toda la carrera, estudiando y celebrando siempre que podíamos, sin ti ICAI no habría sido lo mismo.

A todo el personal docente que me ha ayudado con mi formación, tanto personal como profesional, a lo largo de estos cuatro años. En especial a mi director de proyecto, Juan Luis Zamora Macho, que creyó en mí desde el primer momento y me dio la confianza que necesitaba para alcanzar todos los retos que fueron planteándose durante el desarrollo del proyecto. Trabajar con él ha hecho que este proyecto sea una gran experiencia, en la que he podido aprender una gran cantidad de cosas que tengo claro que me servirán durante toda mi carrera profesional, y que han hecho de este proyecto la despedida perfecta para cerrar esta etapa.

No puedo olvidarme de Jorge, gracias por estar a mi lado desde el primer día, apoyándome en los malos momentos y celebrando los buenos juntos. Gracias por confiar en mí siempre y ayudarme cuando todo se complicaba.

Una última vez, gracias a todos por ayudarme a estar cada vez más cerca de la ingeniera que quiero llegar a ser y de la que espero, podáis sentiros orgullosos.

Resumen

En este proyecto se ha desarrollado un sistema de navegación autónoma para robots móviles en un entorno limitado y provisto de un sistema de percepción externo formado por un conjunto de cámaras infrarrojas.

El sistema de navegación incluye la detección de la posición y orientación del vehículo mediante el sistema de posicionamiento externo, la comunicación inalámbrica de esta información desde las cámaras al robot en tiempo real y los algoritmos de control que permiten seguir la trayectoria planificada.

Palabras clave: Optitrack, robot móvil, sistema de posicionamiento, navegación autónoma.

1. Introducción

El objetivo principal de este trabajo de fin de grado es el diseño e implementación de un sistema de control digital de navegación del vehículo basado en el kit de robótica Balboa 32U4 de la empresa Pololu, Figura 1.1. Por lo tanto, se busca desarrollar el control de un vehículo no tripulado que sea capaz de moverse en un entorno limitado de forma independiente.

El vehículo usará la información obtenida a través de los *encoders* y del sistema de cámaras externo para navegar desde un punto de origen a un destino seleccionado, siendo esta la única información que se le proporciona al sistema.

2. Breve descripción del vehículo y del sistema de cámaras

El hardware del vehículo está compuesto por elementos que se explican a continuación.

La electrónica del vehículo está formada por dos dispositivos encargados de llevar a cabo las acciones de control sobre el robot y que están comunicados entre sí mediante comunicación I^2C. El vehículo consta de una Raspberry Pi 3B+ programada con Matlab y Simulink, que actúa como el ordenador principal, adquiriendo las medidas de la IMU y del resto de sensores, y siendo el responsable de las tareas de mayor complejidad como por ejemplo la gestión del resto de comunicaciones [1] [2]. Además, cuenta con un microcontrolador ATMega32U4 programado con la IDE de Arduino, que actúa como el dispositivo "esclavo", gestionando las tareas de menos nivel[3].

Por otra parte, Balboa 32U4 incluye un conjunto de sensores inerciales que constituyen la unidad de medida inercial (IMU, *Inertial Measurement Unit*), que combina un acelerómetro y un giróscopo de 3 ejes utilizados para detectar la posición del vehículo midiendo la tasa de cambio en la aceleración lineal y la rotación angular respectivamente [4].

Además, el vehículo incluye una serie de *encoders* magnéticos integrados que permiten obtener una medición continua de la velocidad y posición de las ruedas respecto a la posición inicial.

Mediante las lecturas de los *encoders* (ubicados en el eje de cada motor), se realizan los ajustes necesarios a la velocidad del motor para que este trabaje en las condiciones necesarias.

Por último, el vehículo cuenta con dos motores de corriente continua de 6V, que son los principales actuadores del vehículo. El vehículo también cuenta con dos sensores de distancia, que no han sido usados en el desarrollo de este trabajo puesto que la navegación se realiza a través de la información proveniente del sistema de cámaras.

El sistema de posicionamiento de cámaras OptiTrack, es el sistema de localización externo usado en este trabajo. Se trata de un sistema capturador de movimiento, que permite obtener la posición y orientación del vehículo en el espacio tridimensional. Se obtienen las medidas mediante cámaras infrarrojas y los reflectores de luz infrarroja que se encontrarán en el propio vehículo. Definen el sistema de referencia absoluto [5].

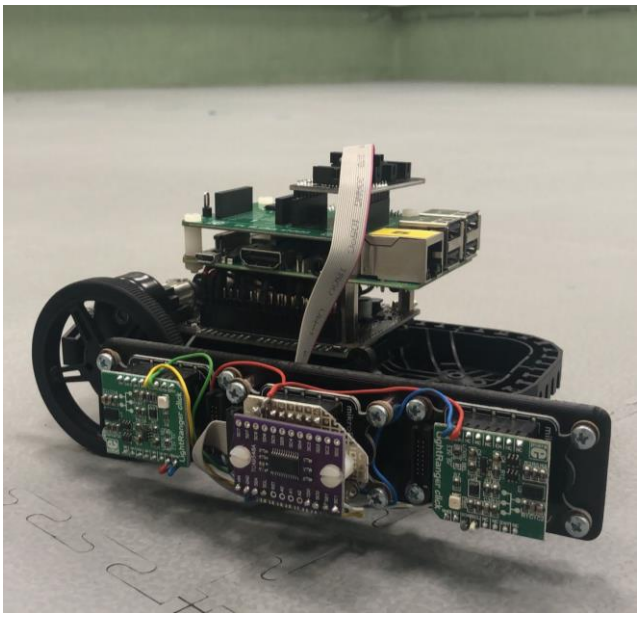

Figura 1.1. Balboa 32U4 con Raspberry Pi

3. Objetivos

Los objetivos fundamentales de este proyecto son los mencionados a continuación:

- Estudio del sistema de cámaras Optitrack, del software de control del mismo (*Motive*) y de su conexión a Simulink para la comunicación de los datos de las cámaras al PC y de este al vehículo [17][5][6].

- Diseño de la estructura del sistema de comunicaciones del proyecto: incluye el envío de información entre el sistema de cámaras y el ordenador, la comunicación vía

Bluetooth de estos datos al vehículo, y por último la comunicación entre vehículo y ordenador vía Wifi, mostrado en la Figura 1.2.

- Diseño y optimización del control y del sistema de Navegación, para que el vehículo sea capaz de seguir una ruta indicada desde un punto de origen a otro de destino.

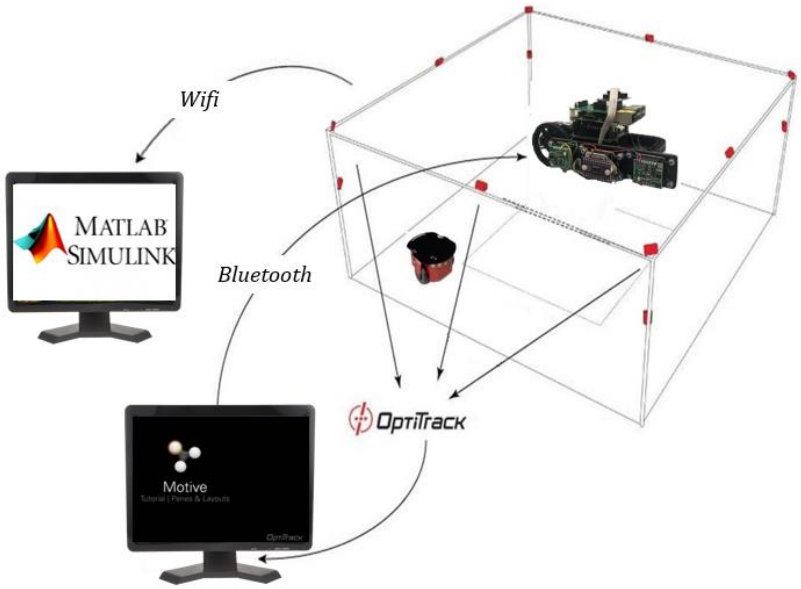

Figura 1.2. Estructura del sistema de comunicaciones del proyecto

4. Solución

Para el diseño del sistema de navegación del vehículo en el plano XY, ha sido necesario diseñar primero tres controles de velocidad de giro, velocidad de avance y ángulo de guiñada.

Por lo tanto, el sistema de navegación cuenta un control de velocidad y otro de giro que permiten al vehículo recorrer trayectorias definidas por puntos de paso, conocidos como *waypoints*. Para ello, se emplea una estructura de control en cascada, con dos lazos de control: el lazo de control interno y el externo.

El recinto de navegación del vehículo es un cuadrado de lado L igual a 2m, cuyos extremos, puntos medios de los lados y por último el punto central del cuadrado, forman los puntos de paso por los que el vehículo puede pasar. De esta forma se pueden definir diferentes trayectorias variando el orden de los *waypoints* en la secuencia que indica el camino a seguir.

El control del vehículo se encuentra integrado en un fichero de Simulink, que se ejecuta en la Raspberry Pi y que permite llevar a cabo la monitorización del robot durante la navegación, mostrado en la Figura 1.3.

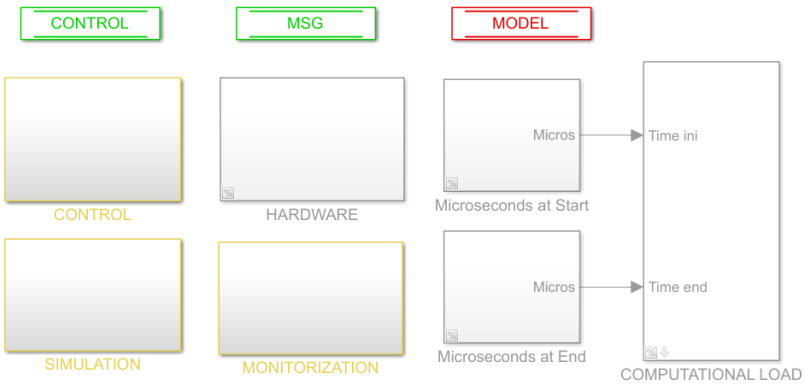

Figura 1.3. Diagrama de Simulink del control del vehículo

El planificador de trayectorias empleado en el diseño del sistema de navegación se trata de un planificador de trayectorias basado en *splines*, es decir, basado en una aproximación por funciones polinómicas derivables.

5. Referencias

[1] Xataka. *https://www.xataka.com/makers/cero-maker-todo-necesario-para-empezar-raspberry-pi*. [Último acceso: 29/06/2021]

[2] RaspberryPi. *https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/.* [Último acceso: 29/06/2021]

[3] Atmel, Febrero 2014, "8-bit Microcontroller with 16/32K Bytes of ISP Flash and USB Controller, ATmega16U4, ATmega32U4, Preliminary", Atmel. Disponible en: *https://docs.rs-online.com/32ab/0900766b80eee70f.pdf*. [Último acceso: 10/07/2021]

[4] C. Jiménez Cortés, "Control de un vehículo equilibrista mediante una Raspberry Pi", Trabajo de fin de grado, Ingeniería en Tecnologías Industriales, Escuela Técnica Superior de Ingeniería ICAI Universidad Pontificia de Comillas, Madrid, España, 2019.

[5] Optitrack. *https://optitrack.com/.* [Último acceso: 29/06/2021]

[6] J. S. Furtado, H.H.T.Liu et al., "Comparative Analysis of OptiTrack Motion Capture Systems", *Proceedings of The Canadian Society for Mechanical Engineering International Congress, CSME International Congress*, 2018, Toronto, Canada.

Abstract

In this project, it has been developed an autonomous navigation system for mobile robots in a limited environment and provided with an external perception system formed by a set of infrared cameras.

The navigation system includes: the detection of the vehicle's position and orientation by using the external positioning system, the wireless communication of this information from the cameras to the robot in real time and the control algorithms that allow it to follow the planned trajectory.

Keywords: Optitrack, mobile robot, positioning system, autonomous navigation.

1. Introduction

The main objective of this project is the design and implementation of a digital control system for the navigation of a vehicle based on the Balboa 32U4 robotics kit from Pololu (Figura 1.4). Therefore, it is sought to develop the control of an unmanned vehicle that is capable of moving in a limited environment independently.

The vehicle will use the information obtained through the *encoders* and the external camera system to navigate from a point of origin to a selected destination, being this information the only one provided to the system.

2. Brief description of the vehicle and the camera system

The vehicle's hardware is made up of the items explained below.

The electronics of the vehicle is composed of two devices that are in charge of carrying out the control actions on the robot and that are communicated with each other through I^2C communication. The vehicle consists of a Raspberry Pi 3B+ programmed with Matlab and Simulink, which acts as the main computer, acquiring the measurements of the IMU and the rest of the sensors, and being responsible for the more complex tasks such as managing the rest of the communications [1][2].In addition, it has an ATMega32U4 microcontroller programmed using the Arduino IDE, which acts as the "slave" device, managing lower-level tasks [3].

On the other hand, Balboa 32U4 includes a set of inertial sensors that constitute the inertial measurement unit (IMU), which combines a 3-axis accelerometer and a 3-axis gyroscope used to detect the position of the vehicle by measuring the rate of change in linear acceleration and the angular rotation respectively [4].

In addition, the vehicle includes a series of integrated magnetic *encoders* that allow obtaining a continuous measurement of the speed and position of the wheels with respect to the initial position.

Through the readings of the *encoders* (located on the shaft of each motor), the necessary adjustments are made to the motor speed so that it works under the necessary conditions.

Finally, the vehicle has two 6V DC motors, which are the main actuators of the vehicle. The vehicle also has two distance sensors, which have not been used in the development of this project since navigation is carried out through the information from the camera system.

The OptiTrack camera positioning system is the external location system used in this project. It is a motion capture system, which allows obtaining the position and orientation of the vehicle in three-dimensional space. Measurements are obtained using infrared cameras and infrared light reflectors that will be found on the vehicle itself. They define the absolute reference system [5].

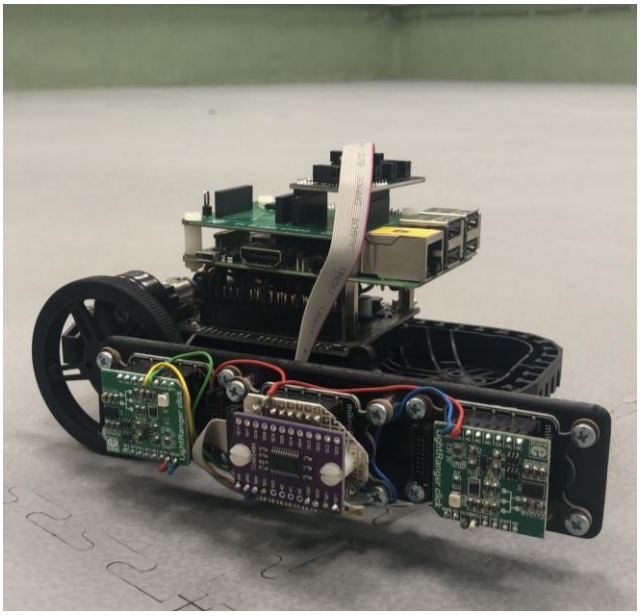

Figura 1.4. Balboa 32U4 with Raspberry Pi

3. Objectives

The main objectives of this project are as follows:

- Study of the Optitrack camera system, its control software (Motive) and its connection to Simulink for communicating the data from the cameras to the PC and from this to the vehicle [5][6].

- Design of the project's communications system structure: includes the sending of information between the camera system and the computer, the communication via

Bluetooth of this data to the vehicle, and finally the communication between the vehicle and the computer via Wifi, (Figura 1.5).

- Design of the control and the Navigation system, so that the vehicle is able to follow an indicated route from one point to another in the environment.

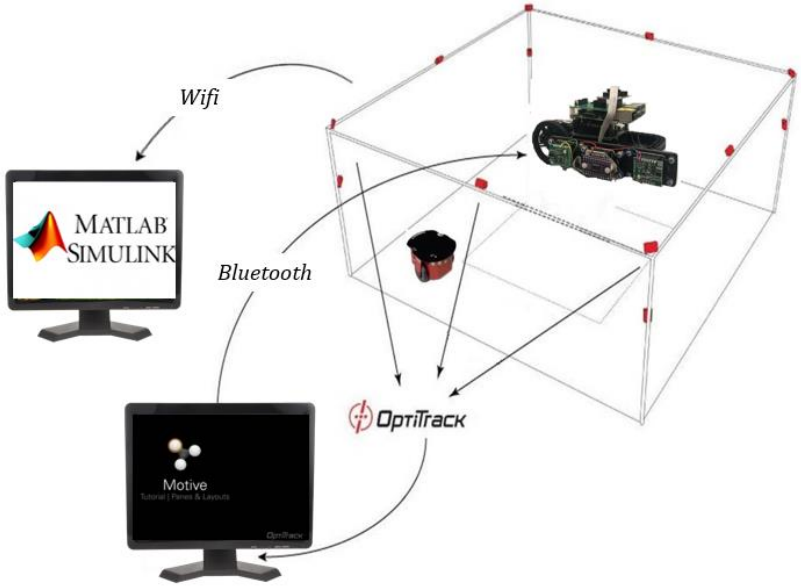

Figura 1.5. Project's communications system structure

4. Solution

For the design of the vehicle navigation system in the XY plane, first it has been necessary to design three controls for forward velocity, yaw rate and yaw angle.

Therefore, the navigation system has a speed control and a yaw control that allow the vehicle to travel paths defined by waypoints. To do this, a cascade control structure has been used in this project, with two control loops: the internal and external control loop.

The navigation area of the vehicle is formed by a square with side L equal to 2m, whose ends, midpoints of the sides and finally the central point of the square, form the points of passage through which the vehicle can pass. In this way, different trajectories can be defined by varying the order of the waypoints in the sequence that indicates the path to follow.

The vehicle's control is integrated into a Simulink file, which runs on the Raspberry Pi and which allows the monitorization of the robot during navigation, which is shown in Figura 1.6.

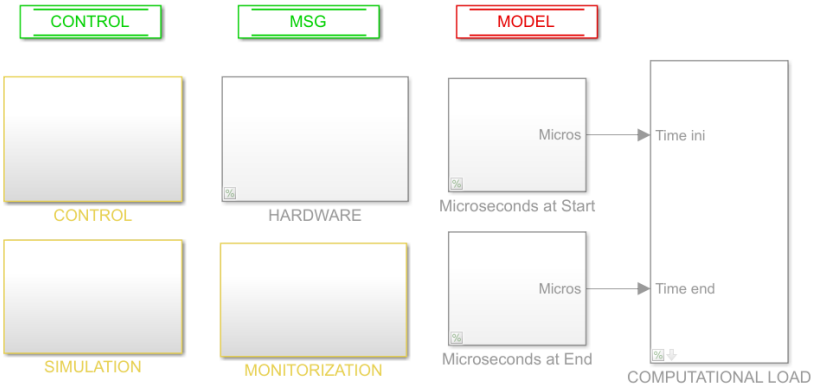

Figura 1.6. Simulink diagram of the vehicle's control

The trajectory planner used in the design of the navigation system is a trajectory planner based on *splines*, that is based on an approximation by differentiable polynomial functions.

5. References

[1] Xataka. *https://www.xataka.com/makers/cero-maker-todo-necesario-para-empezar-raspberry-pi*. [Último acceso: 29/06/2021]

[2] RaspberryPi. *https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/.* [Último acceso: 29/06/2021]

[3] Atmel, Febrero 2014, "8-bit Microcontroller with 16/32K Bytes of ISP Flash and USB Controller, ATmega16U4, ATmega32U4, Preliminary", Atmel. Disponible en: *https://docs.rs-online.com/32ab/0900766b80eee70f.pdf*. [Último acceso: 10/07/2021]

[4] C. Jiménez Cortés, "Control de un vehículo equilibrista mediante una Raspberry Pi", Trabajo de fin de grado, Ingeniería en Tecnologías Industriales, Escuela Técnica Superior de Ingeniería ICAI Universidad Pontificia de Comillas, Madrid, España, 2019.

[5] Optitrack. *https://optitrack.com/.* [Último acceso: 29/06/2021]

[6] J. S. Furtado, H.H.T.Liu et al., "Comparative Analysis of OptiTrack Motion Capture Systems", *Proceedings of The Canadian Society for Mechanical Engineering International Congress, CSME International Congress*, 2018, Toronto, Canada.

Índice de la memoria

Índice de figuras

Anexo A

Anexo B

Parte I: MEMORIA

DESCRIPTIVA

Capítulo 1. INTRODUCCIÓN

1.1 MOTIVACIÓN DEL PROYECTO

La sociedad avanza hacia una realidad en la que la forma de relacionarnos con nuestro entorno será muy diferente a la actual. Los objetos que nos rodean estarán conectados a internet para hacernos el día a día mucho más sencillo, realizando las tareas que tienen asignadas de una forma automática. Este concepto se denomina Internet de las Cosas o Internet of Things (IoT).

En algunos casos, estos objetos necesitarán estar dotados de un sistema que les permita moverse de forma autónoma para poder cumplir con las funciones que tengan asignadas. Este sería el caso de los robots-aspiradora, que necesitan un sistema de navegación que les permita moverse por el hogar, además de un sistema de control que les indique cómo realizar su tarea de la forma más eficiente.

En este proyecto se va a desarrollar el sistema de navegación para un robot móvil con un sistema de localización externo. Esta tecnología puede ser muy útil no solo para ayudar a automatizar las tareas del hogar, sino también en aplicaciones industriales como el transporte de mercancías en almacenes.

Por las razones descritas anteriormente, el reto personal que supone y debido al gran impacto que puede tener en el futuro, se ha decidido llevar a cabo este proyecto.

1.2 OBJETIVOS

Este proyecto gira en torno al desarrollo de un sistema de navegación para un pequeño vehículo autónomo, con el objetivo de que éste sea capaz de moverse en un entorno limitado y seguir las trayectorias que se le indiquen. La estructura planteada para poder alcanzar el objetivo propuesto se muestra a continuación, junto con una explicación de cada una de las partes:

1) Análisis de la posición y orientación del vehículo en el espacio:

Se realizó un estudio del hardware que se usó en el proyecto como sistema de posicionamiento del vehículo: el sistema capturador de movimiento Optitrack. Además, se llevó a cabo un análisis en mayor profundidad del funcionamiento del software de control del mismo, Motive; puesto que a través de este se pudo obtener la información sobre la posición y orientación del coche en el espacio tridimensional delimitado por las ocho cámaras infrarrojas que forman el sistema de localización externo.

Para poder llevar a cabo el seguimiento del vehículo en tiempo real, previamente fue necesario preparar y definir el entorno en el que el vehículo fuese a navegar, llevando a cabo varias calibraciones de las cámaras y definiendo el plano de referencia usado durante el rastreo. Se tuvo que definir el vehículo como un cuerpo rígido sobre el que poder realizar el estudio, para lo que se usaron unos reflectores de luz infrarroja, conocidos como marcadores.

Este sistema es el encargado de proporcionar la información sobre la posición y orientación del coche. Además, se realizó un estudio para poder llevar a cabo la transmisión de datos entre Simulink y Motive, y poder así realizar el rastreo en tiempo real del vehículo. Por otra parte, se realizó un análisis de la comunicación entre el ordenador y el vehículo, decidiendo entre Wifi o Bluetooth como vía de transmisión de datos.

También fue necesaria la programación de los *scripts*, que permitieran enlazar el sistema de cámaras y el PC, para poder controlar y filtrar la información sobre la localización del vehículo. También se analizó la transmisión de estos datos del ordenador al vehículo mediante el uso de Bluetooth. Se realiza una explicación más detallada de las comunicaciones en el Capítulo 4.

2) Diseño del sistema de comunicaciones del proyecto:

Toda la información sobre las comunicaciones del proyecto se explica de forma más detallada en el Capítulo 4.

Para el correcto funcionamiento del proyecto ha sido necesario crear una estructura de comunicación robusta, que permita en todo momento establecer las comunicaciones entre el ordenador receptor de la información proveniente de las cámaras, el vehículo y el ordenador encargado de monitorizar.

Ha sido necesario filtrar y tratar la información proveniente de las cámaras puesto que no toda esta resultaba útil para el proyecto o en el caso de la posición, no estaba en el sistema de coordenadas correcto.

Se ha escogido la transmisión vía Bluetooth de la información de las cámaras al vehículo, mientras que la comunicación entre el ordenador de monitorización y el vehículo se lleva a cabo vía Wifi.

3) Diseño del control y del Sistema de Navegación:

En la última etapa del proyecto, se llevan a cabo los diseños de control de velocidades de avance y giro, además del control del ángulo para la navegación. Para ello es necesario realizar una serie de ensayos y mediante la información obtenida a través de estos hacer la identificación de los parámetros del modelo del vehículo. Se realiza la identificación a través de ensayos llevados a cabo tanto en lazo abierto como en lazo cerrado.

Tras el diseño de los controles ha sido necesario realizar varios ensayos para poder comprobar el correcto funcionamiento de los mismos y que el sistema no fuese inestable, haciendo una comparación entre los ensayos y las simulaciones hechas anteriormente. Toda la información sobre los controles del vehículo se describe en el Capítulo 5.

1.3 RECURSOS

Puesto que se trata de un proyecto que engloba varias disciplinas, entre ellas la mecánica, electrónica o robótica, fue necesario el uso de varios elementos hardware. Estos componentes se dividen en dos grupos: los elementos que conforman el vehículo y el sistema de posicionamiento externo.

1) Recursos hardware:

A continuación, se explica brevemente el funcionamiento de cada uno de los elementos que componen el vehículo, que se tendrán en cuenta a la hora de desarrollar el sistema de control del mismo. Se realiza una descripción más detallada de todos ellos en el Capítulo 3.

- Microcontrolador ATMega32Ua: dispositivo "esclavo" encargado de llevar a cabo las tareas de bajo nivel, relacionadas con el tratamiento de la información proveniente de los sensores y actuadores.

- Raspberry pi 3B+: microcontrolador principal, encargado de las tareas que sean más complejas y conlleven una mayor carga computacional.

- Unidad de medida inercial (IMU, *Inertial Measurement Unit):* Incluido en la placa de ATMega32U4, se trata de un chip ST LSM6DS33 que combina un acelerómetro y un giróscopo, ambos de 3 ejes.

- *Encoders* magnéticos: Se pueden utilizar para rastrear la velocidad de rotación y dirección de las ruedas del robot.

- Dos motores *75:1 Micro Metal Gear Motor HPCB 6V*: Se trata de dos motores de DC con escobillas de carbón de alta duración y una caja de cambios metálica de 75,81:1, que son los principales actuadores del vehículo.

El sistema de localización externo usado en este proyecto, será el encargado de proporcionar la información sobre la posición y orientación del vehículo. El recurso empleado ha sido Optitrack, cuya explicación más detallada se encuentra en el Capítulo 3.

- Optitrack es un sistema capturador de movimiento, que nos permite rastrear la posición de cualquier objeto, en el caso de este proyecto se trata del vehículo, a través de cámaras infrarrojas.

2) Recursos software:

A continuación, se muestra una breve descripción del conjunto de programas que se han empleado durante la realización de este proyecto:

- Motive: Se trata del software encargado del control del sistema de posicionamiento empleado en este proyecto, que está formado por ocho cámaras infrarrojas, Figura 1.1. A través de este, se realizó la transmisión de los datos sobre el posicionamiento del robot al PC.

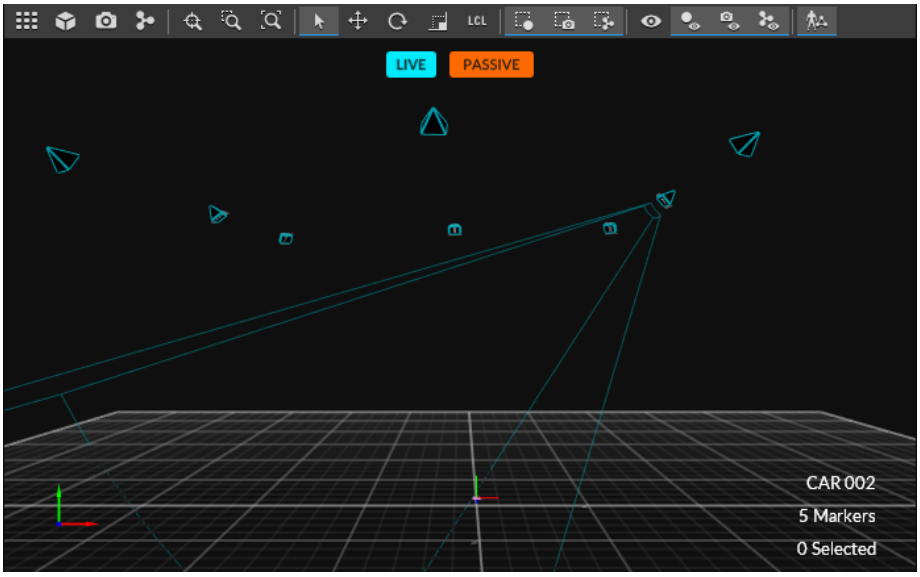

Figura 1.1. Entorno de Motive, visualización de las cámaras infrarrojas

- Matlab-Simulink: Han permitido programar los ficheros necesarios para configurar la lógica el vehículo y establecer las comunicaciones entre los diferentes recursos hardware. A través de Simulink se han podido realizar y monitorizar los ensayos que se han ido llevando a cabo.

- Arduino-IDE: Se ha empleado para configurar los módulos bluetooth necesarios para llevar a cabo la comunicación entre el PC desde el cual se realizaba el rastreo del vehículo y el propio vehículo.

1.4 METODOLOGÍA

Para satisfacer el objetivo principal del proyecto, la ejecución del mismo se ha dividido en las etapas que se describen en esta sección. Durante el desarrollo del proyecto, las etapas fueron modificándose en función de la evolución del trabajo y de las dificultades que fueron apareciendo.

Estudio del sistema de posicionamiento externo

Durante el inicio del proyecto fue necesario recoger toda la documentación sobre el sistema de posicionamiento externo que se iba a utilizar, Optitrack y sobre su software de control, Motive. Se realizó un estudio del software para poder adaptarlo a las necesidades del proyecto y obtener a través de él la información que permitiese ubicar al vehículo en el espacio tridimensional donde se realizaron los ensayos.

Mediante el sistema de cámaras infrarrojas y los reflectores de luz infrarroja colocados en el vehículo, se pudo obtener la posición del centro de masas del vehículo respecto a los tres ejes de coordenadas (X, Y, Z), además de los tres ángulos necesarios para posicionar completamente al vehículo en el espacio 3D (guiñada, cabeceo y alabeo).

Sin embargo, para el desarrollo del sistema de planificación de trayectorias del vehículo, bastaba con las coordenadas en los ejes X e Y, puesto que la navegación del vehículo se realizaría en el plano formado por estos dos ejes. Además, sólo nos resultaba útil el ángulo de guiñada, ya que este indica el giro del vehículo sobre su eje vertical. Por lo tanto, fue necesario desarrollar las rutinas que permitiesen filtrar la información proveniente de las cámaras a través de Matlab-Simulink.

Comunicaciones entre el vehículo y el ordenador

Una vez obtenida la información sobre la ubicación del vehículo y tras adaptarla a las necesidades del proyecto, fue necesario establecer un sistema de comunicación entre el ordenador que realizaba el control de las cámaras y el propio vehículo. Para ello se decidió usar tecnología Bluetooth frente a Wifi, puesto que la transmisión de datos entre dispositivos posicionados uno cerca del otro es mejor, siempre y cuando el ancho de banda sea pequeño, como en el caso de este proyecto.

Para establecer la comunicación de ambos se emplearon dos módulos bluetooth y un conversor USB UART, cuyo funcionamiento y configuración se explica de forma más detallada en el Capítulo 4. En un primer momento se realizaron pruebas con otra Raspberrypi 3B+ diferente a la empleada como microcontrolador principal en el vehículo, por lo que también fue necesario configurar esta Raspberry desde el inicio. Tras esto se realizaron pruebas enviando un único número, para comprobar la correcta configuración de los módulos bluetooth desde Simulink. Una vez establecida la comunicación, se adaptó el fichero de

Simulink para poder emplearlo en el proyecto, enviando al vehículo la información sobre su posición en tiempo real.

Para cerrar el lazo de transmisión de datos, se llevaron a cabo una serie de pruebas para comprobar que la información sobre la posición que recibía el propio vehículo se enviaba correctamente de nuevo al PC de monitorización.

Las comunicaciones del proyecto se describen de forma más detallada en el Capítulo 4.

Diseño de los controles PID de velocidad

Una vez establecido el sistema de comunicaciones del proyecto, se procedió a diseñar los controles de velocidad de avance y giro, además del control del ángulo necesario para la navegación.

En un primer lugar se realizaron una serie de ensayos con el vehículo que permitieron estimar los parámetros del modelo del vehículo y determinar de esta forma las funciones de transferencia de las plantas, necesarias para diseñar los tres controles: velocidad de avance, velocidad de giro y ángulo de giro. Se llevaron a cabo ensayos en lazo abierto y lazo cerrado, por lo que se realizó la identificación del modelo para ambos ensayos.

Tras la identificación de los modelos se diseñaron los controles de velocidad de avance y giro y el control del ángulo de giro necesario para la navegación. Estos controles se estudiaron y analizaron mediante simulaciones. Más tarde, a través de ensayos se realizaron una serie de comparaciones para comprobar si el funcionamiento de los controles era el esperado.

El diseño y análisis de estos controles se encuentra recogido y explicado en el Capítulo 5.

Sistema de Navegación

Para el diseño del sistema de navegación, además de los tres controles de velocidad de avance, giro y ángulo de guiñada diseñados, se ha empleado un planificador de trayectorias basado en *splines,*

Se ha explicado el funcionamiento del planificador de trayectorias de forma más detallada en el Capítulo 6. En este mismo capítulo se encuentran las simulaciones que se han llevado a cabo del sistema de navegación del vehículo junto con las explicaciones necesarias.

Capítulo 2. ESTADO DEL ARTE

Actualmente el vehículo autónomo se considera una de las tecnologías emergentes más importantes, como se puede observar en los ciclos de sobreexpectación (Hype Cycle) que la empresa Gartner presenta anualmente Capítulo 8. . Además, se producirán otros avances tecnológicos a lo largo de esta década que afectarán a la evolución de la conducción autónoma, como el Internet de las Cosas (IoT) o la llegada del 5G que permitirá conexiones mucho más rápidas y de menor latencia.

Los fabricantes clásicos de la industria del automóvil ya han apostado por el vehículo autónomo, como es el caso de Toyota o Volkswagen. Además, empresas que han surgido a partir de la idea del coche autónomo se han convertido en pioneras, como es el caso de Tesla y su proyecto "Full Self-Driving" (FSD). Cabe destacar la importancia de esta tecnología y sus posibles aplicaciones, ya que las grandes empresas tecnológicas como Google, con su proyecto Waymo; Amazon, con la compra de la empresa Zoox especializada en vehículos autónomos o Apple, que está desarrollando su propio proyecto, han decidido apostar por esta idea.

2.1 NAVEGACIÓN AUTÓNOMA

El coche tradicional ha ido evolucionando poco a poco e incorporando diferentes mejoras que han permitido automatizar ciertos procesos que antes eran manuales como por ejemplo el aparcamiento asistido, la velocidad de crucero o el frenado de emergencia. Para clasificar el nivel de autonomía de un vehículo, la Autoridad de Tráfico de Estados Unidos (National Highway Traffic Safety Administration, NHTSA por sus siglas en inglés) divide en 6 niveles la capacidad de este para circular de forma independiente [2].

Nivel 0: No Automatizado.

Este nivel de conducción se corresponde con la navegación tradicional en la que el conductor humano lleva a cabo toda la conducción a tiempo completo, teniendo el control en todo momento.

Nivel 1: Asistencia al Conductor.

Un sistema de asistencia al conductor (ADAS, Advanced Driver-Assistance Systems) ayuda con la dirección o el frenado/aceleración, pero no puede actuar sobre ambas acciones de forma simultánea. El conductor sigue realizando la mayoría de tareas en la conducción y el vehículo simplemente facilita la conducción realizando algunas de ellas de forma semiautomática. Un ejemplo de esto sería el control de crucero o los controles de estabilidad.

Nivel 2: Automatización Parcial.

En este caso, el sistema de asistencia al conductor (ADAS) puede ayudar al conductor con la dirección y con el frenado/aceleración de forma simultánea en algunas circunstancias, aunque este proceso debe seguir estando supervisado por el conductor en todo momento.

Nivel 3: Automatización Condicional.

Para el nivel 3, el sistema de conducción automatizado (ADS, Automated Driving Systems) está dotado de lo necesario para poder realizar todas las tareas de la conducción, pero únicamente en condiciones favorables. El conductor debe seguir atento para tomar el control del vehículo cuando el ADS lo solicite, como al circular por una carretera en mal estado o en condiciones meteorológicas adversas.

Nivel 4: Alta Automatización.

En el nivel 4, con ayuda del ADS, el vehículo puede realizar de forma independiente todas las tareas de la conducción y supervisar el entorno cuando las circunstancias son favorables. La diferencia con el nivel anterior es que, en estas condiciones, no es necesario que el conductor preste atención.

Nivel 5: Automatización Completa.

En este último nivel, las personas que circulan en el vehículo dejan de ser conductores y pasan a ser solo pasajeros. El vehículo lleva a cabo todas las tareas de conducción de forma completamente autónoma gracias al ADS.

En este proyecto el nivel de conducción autónoma del vehículo se encuentra entre el nivel 4 y el nivel 5, puesto que se trata de un vehículo no tripulado que es capaz de moverse en un entorno limitado de forma independiente. El vehículo usará la información obtenida a través de sus sensores y del sistema de cámaras externo para navegar desde un punto de origen a un destino seleccionado, siendo esta la única información que se le proporciona al sistema.

2.2 TECNOLOGÍAS CLAVE

Las cuatro tecnologías con las que cuenta un vehículo autónomo actualmente son un sistema de navegación, un sistema que permita planificar la rutas, un sistema de percepción del entorno y un sistema de control del automóvil [3].

Sistema de navegación

Los vehículos autónomos cuentan normalmente con un sistema de navegación que les permite ubicarse en el entorno que les rodea. La ubicación de un vehículo puede ser absoluta, relativa o híbrida, además los sistemas de posicionamiento varían en función de si el vehículo circula en un entorno limitado o no.

Para conocer su ubicación absoluta en un entorno no limitado, la gran mayoría usa sistemas de posicionamiento basados en satélites, como GPS, GLONASS y Galileo. Estos sistemas tienen un problema, ya que las señales de los satélites pueden sufrir interferencias por ejemplo al circular por un túnel o en menor medida, debido a las condiciones meteorológicas, por lo que esta tecnología puede no ser precisa siempre. Por otra parte, los vehículos autónomos pueden contar con un sistema de navegación inercial (INS, Inertial Navigation System), que se trata de un sistema de ubicación relativa formado por giróscopos y acelerómetros. La tendencia actual es usar ambos sistemas en conjunto para obtener una información más fiable, es lo que se conoce por ubicación híbrida.

En cambio, es más sencillo conocer la posición de vehículos cuyo movimiento está limitado. Para conocer el posicionamiento indoor de los robots autónomos, también se usan técnicas odométricas, que utilizan datos de sensores colocados en las ruedas para conocer la posición relativa. La información obtenida por métodos odométricos no siempre es fiable, por lo que también resulta útil obtener una posición absoluta del robot por ejemplo mediante landmarks o referencias externas [4].Otro método puede ser el empleado en este proyecto: un sistema de cámaras infrarrojas y reflectores de luz infrarroja colocados en el vehículo.

Sistema de planificador de rutas

Una vez conocida la ubicación del vehículo, es necesario determinar el camino que este va a seguir, de ahí la importancia de la planificación de rutas. La planificación de rutas permite a los vehículos autónomos generar un camino óptimo tras detectar un obstáculo. Por ejemplo, en el caso de los robots-apiradora, esto se traduce en un gasto energético mínimo [5].

La planificación de rutas puede ser estática o dinámica. En el caso de los vehículos autónomos, la planificación de rutas suele ser dinámica ya que los objetos u obstáculos que los rodean pueden estar en movimiento, hecho que dificulta en gran medida el proceso. Mediante esta planificación se obtienen rutas factibles que cumplen con los requisitos de tiempo y seguridad, ambos muy importantes en cualquier navegación [6].

En la Sección 2.4 se realiza una breve descripción de los sistemas de planificación de trayectorias empleados actualmente en los proyectos de robótica y más concretamente el empleado en este proyecto.

Sistema de percepción del entorno

Cualquier vehículo autónomo está dotado de un sistema que le permite percibir el entorno que le rodea. Además de su propia posición, el vehículo debe ser capaz de observar y predecir el comportamiento del resto de cuerpos que están presentes durante la navegación. Para ello el vehículo suele contar con un conjunto de sensores, cada uno de ellos con sus propias características [7]. Los más usados en la industria actualmente son:

- Los sensores ultrasónicos, cuyo rango operativo es de 2 metros y que suelen usarse en la función de aparcamiento asistido.
- Un sistema de cámaras, necesarias para el reconocimiento de imágenes.
- Un sistema de cámaras infrarrojas, que permiten la navegación nocturna.
- Los sensores de radar, usados en el control de crucero automatizado, los sistemas de frenado de emergencia, la detección de colisiones y la asistencia para cambio de carril. Su uso es tan amplio debido a su robustez y a su largo alcance.
- El LIDAR, que permite crear mapas 3D detallados del entorno que rodea al vehículo. Aunque se trata de una tecnología cara, aporta información muy útil y precisa [8].
- Los *encoders*, cuya función es determinar el valor de la posición, velocidad o aceleración mediante el valor medido del movimiento de las ruedas.
- Los acelerómetros y giroscopios, utilizados para detectar la posición del vehículo midiendo la tasa de cambio en la aceleración lineal y la rotación angular respectivamente.

Sistema de control

La lógica de control es la responsable de determinar el flujo de ejecución, decidir qué tareas han de desarrollarse en cada instante y según la reacción del sistema y del entorno que está siendo intervenido, cuáles deberán realizarse en un futuro [9].

En el caso de la percepción artificial, existe un gran desafío debido a la inmensa variabilidad de las condiciones visuales que rodean al vehículo. Los módulos de percepción desarrollados para obtener información sobre carriles, detectar y clasificar vehículos o incluso peatones deben evaluarse, optimizarse y calibrarse en condiciones realistas. Esto supone un problema puesto que el número de diferentes situaciones que se puede dar en la realidad, son casi infinitas, por lo que es muy complicado tenerlas a todas en cuenta [10].

El desafío que supone el sistema de control necesario en la navegación autónoma y las posibles soluciones que tiene se explican más detalladamente en la Sección 2.3.

2.3 PROBLEMAS Y SOLUCIONES AL SISTEMA DE CONTROL

El control de los vehículos autónomos supone un reto debido en parte a que los sistemas de navegación autónomos basados en técnicas tradicionales de procesamiento de imágenes y reconocimiento de patrones, suelen funcionar bien en determinadas condiciones, pero

presentan problemas en cuanto el entorno está menos controlado o no está limitado. Es decir, su efectividad se reduce considerablemente cuando las variables a tener en cuenta se comportan de forma aleatoria.

Esto significa, que el control en entornos limitados es más sencillo, debido a que en este caso no se producen cambios en el entorno que impliquen un problema para su navegación. En cambio, en el caso de los vehículos que circulan en entornos no limitados, existen agentes externos que se comportan de manera aleatoria y dificultan la navegación exigiéndole al sistema tomar decisiones en milésimas de segundo.

En cualquier es necesario tener un conjunto de acciones básicas que permitan al robot interaccionar con el entorno que le rodea. Estas rutinas han sido desarrolladas por personas provenientes del campo del control clásico o por aquellos cuya base se centra en el campo de la inteligencia artificial [11].

En cuanto al control de los robots industriales, a lo largo del tiempo hayan ido surgiendo distintas corrientes o paradigmas en el diseño de sistemas de control de robots. Tradicionalmente se encuentran divididos en tres grandes bloques: los sistemas deliberativos, los reactivos y los híbridos [12][13][14],poseyendo cada uno ellos sus propias ventajas e inconvenientes.

La principal diferencia en las arquitecturas se puede encontrar en su computabilidad, la cual se mide en el tiempo de respuesta, capacidad de aprendizaje y un modelado previo del entorno que les rodea.

El control reactivo, se caracteriza por tener un tiempo de respuesta muy rápido, debido a que no utiliza un modelo del entorno para ejecutar las acciones del robot. Los sensores del mismo, son los encargados de proveer la información necesaria para determinar la acción a realizar sobre los actuadores, lo que definirá el comportamiento final del robot. El problema que tienen estos controles es que no siempre son los más precisos, porque el uso de sensores únicamente para la toma de decisiones hace que no sean muy fiables.

El control deliberativo por otra parte tiene su origen en las primeras aplicaciones de la robótica y está basado en la obtención de un modelo preciso del entorno en el cual debe desenvolverse el robot. A partir de este modelo del espacio que le rodea, se realiza una planificación que determina las acciones que tiene que ejecutar el robot en cada instante para poder alcanzar su objetivo. El principal inconveniente de estos controles es que el mundo real es demasiado complejo y tiene demasiados detalles para poder obtener modelos exactos del mismo. Por lo que su posible aplicación queda reducida a entornos estáticos, donde no se requiera mucha capacidad de adaptación ante cambios no previstos por parte del robot. Además, es necesaria una gran capacidad de almacenamiento para poder guardar estos modelos, lo que también supone un problema.

Por último, los sistemas híbridos combinan aspectos de los métodos reactivos y deliberativos permitiendo obtener controles más robustos y a la vez flexibles.

Actualmente, la mayoría de los manipuladores robotizados son sistemas altamente no-lineales, cuyo funcionamiento depende directamente de su modelo dinámico y de la efectividad del controlador que empleen. Por lo tanto, el análisis y diseño de las estrategias de control para robots requiere el desarrollo de ecuaciones dinámicas eficientes.

A continuación, se resumen brevemente algunas de las múltiples estrategias de control empleadas en manipuladores robotizados en la industria actual: PID, técnica del par calculado [33] y modelo de referencia adaptativo [34], [35].

PID

El control PID es la técnica más empleada para controlar manipuladores industriales. Los controladores PID están constituidos por tres tipos de compensaciones: Control Proporcional (P), control Integral (I) y control Derivativo (D).

En servomecanismos que utilizan control proporcional, el problema esencial es que en los actuadores del manipulador no se produce un torque que cambie su sentido de giro antes de que el error de posición sea cero, por lo que el sistema está condenado a sobrepasarse severamente. Por otra parte, el control derivativo puede aplicarse para controlar la pendiente de la respuesta de posición, con lo que el tiempo de respuesta será menor. Por último, el propósito del control integral es eliminar el error [32].

Para los controles de este vehículo se ha escogido este tipo de control, puesto que permite obtener controles robustos y menos tediosos de diseñar que el resto de los propuestos a continuación.

Técnica del par calculado

Esta técnica de control se basa en las ecuaciones de movimiento de *Lagrange-Euler* o *Newton-Euler*, que describen el comportamiento dinámico de un manipulador robotizado. La técnica del par calculado es un control de compensación anticipativo, cuya principal desventaja es que la convergencia del vector de error de posición depende de los coeficientes dinámicos y de la ecuación de movimiento, lo cual exige un modelo muy preciso de la dinámica del manipulador, y tiene mucha sensibilidad frente a los errores.

Modelo de referencia adaptativo

Debido a los inconvenientes que presenta la técnica del par calculado, diversas aproximaciones se han propuesto para desarrollar controladores más robustos de manera que su desempeño no sea sensible a errores de modelado. Dentro de estas aproximaciones se encuentra el control adaptivo, en el cual la ley de control se actualiza de manera continua para cambiar ante los parámetros del sistema que está siendo controlado.

Los algoritmos de control adaptativo representan una mejoría respecto a los algoritmos clásicos de control en robótica, ya que las leyes de control adaptativo permiten un seguimiento de trayectorias más rápido y robusto; con mayor precisión, independientemente de los parámetros del sistema variantes en el tiempo.

Los métodos de control adaptivos están basados en la suposición de que los parámetros del sistema que está siendo controlado no cambian tan rápidamente en comparación con las constantes de tiempo del sistema. Dichas técnicas han sido probadas y han resultado bastante efectivas en algunas aplicaciones específicas. como es el caso de aquellos procesos donde los parámetros sufren cambios graduales, como por ejemplo, los procesos químicos.

2.4 PLANIFICADOR DE TRAYECTORIAS PARA ROBOTS

En cuanto a la planificación de trayectorias de robots móviles, la mejora tecnológica ha permitido que los robots puedan realizar trayectorias cada vez más complejas, al poder ser éstas calculadas previamente.

La planificación de trayectorias se basa en la búsqueda de una sucesión de posiciones para un robot, que permitirán llevarlo desde un estado inicial a uno final, entendiéndose por estado a la descripción de la ubicación del robot referida a un marco de referencia absoluto.

Generalmente, se recurre a una representación realizada a partir de la discretización del espacio del ambiente de trabajo, con lo que se extrae una representación segura, con la garantía de que el espacio libre podrá ser ocupado por el robot (sin riesgo de colisión). Por lo tanto, es necesario que tal discretización se haga en base a las características geométricas, tanto del robot como de los obstáculos.

Existen dos formas básicas para especificar el movimiento:

- Suministrando puntos consecutivos e ignorando la trayectoria espacial que describe el robot entre cada dos puntos.
- Especificando el camino que debe unir los puntos mediante una determinada trayectoria, tal como una línea recta o un círculo, que debe describir el robot en el espacio de trabajo.

La primera alternativa, denominada tradicionalmente control punto a punto, sólo tiene interés práctico cuando los puntos están suficientemente separados, ya que, en caso contrario, la parte de especificación sería muy tediosa. Por otra parte, los puntos tampoco pueden estar muy separados puesto que en ese caso el riesgo de que se generen movimientos imprevisibles o no controlados es mucho mayor.

Las trayectorias punto a punto sólo están implementadas en robots muy simples o con sistemas de control muy limitados.

La segunda estrategia se denomina control de trayectoria continua. En este caso, el sistema de control debe hacer que el robot reproduzca lo más fielmente posible la trayectoria especificada. Suponiendo que la trayectoria que debe seguir el robot se especifica en el espacio cartesiano, existen dos alternativas para su ejecución:

1. Definir los lazos de control directamente en el espacio cartesiano y controlar el robot para que se anule el error de seguimiento de trayectoria en este espacio.

2. Transformar la trayectoria del espacio cartesiano al espacio de las variables articulares y controlar la evolución de cada una de las variables articulares definiendo los lazos de control en este espacio.

El problema de la planificación de trayectorias debe resolverse en tiempo real. Por lo tanto, también se trata de que la generación de trayectorias sea computacionalmente eficiente. En robots manipuladores, la generación de trayectorias articulares suele realizarse en tiempos de orden de los milisegundos o decenas de milisegundos.

Por otra parte, existe la planificación de trayectorias basada en *splines.* Estas son curvas paramétricas con valores vectoriales que se utilizan comúnmente en el diseño asistido por ordenador. Aunque recientemente, ha aumentado su aplicación en proyectos de robótica debido a la eficacia de su uso en la planificación en tiempo real y maniobras complejas [15].

Hay muchas curvas de interpolación para estos puntos dados discretos, como curva *spline* cúbica normal, curva de Bézier, curva B-*spline*, curva *spline* Bézier triangular cúbica, curva NURBS, etc. Concretamente en este proyecto se emplea un planificador de trayectorias basado en *splines* con curvas cúbicas, mediante este método se pueden obtener las reglas del movimiento articular optimizado. El problema principal radica en que la cantidad de cálculo con este método aumenta a medida que se añaden puntos intermedios en la ruta y puede resultar en una ruta menos suave [16].

Capítulo 3. HARDWARE

En este capítulo se describe el funcionamiento y la labor que desempeña cada elemento hardware empleado, en el desarrollo total del proyecto.

En un primer lugar se describe el sistema de posicionamiento externo empleado para la obtención de los datos de posicionamiento del vehículo, además se realizará una breve explicación del sistema software empleado para controlar el sistema de localización externo.

A continuación, se realizará una introducción del vehículo empleado, para después explicar de forma más detallada el conjunto de componentes hardware que lo forman y qué función desempeñan.

3.1 DESCRIPCIÓN DEL SISTEMA DE LOCALIZACIÓN

Los sistemas de localización tienen un gran valor práctico en aplicaciones de robótica móvil para interiores, pero es muy importante elegir correctamente el sistema de captura de movimiento adecuado para una aplicación particular. A continuación, se muestra una explicación del sistema de cámaras empleado durante el desarrollo del proyecto y del sistema software que se ha usado para el tratamiento de los datos y su posterior envío [17].

3.1.1 HARDWARE: OPTITRACK

El sistema de localización empleado en este proyecto: Optitrack, se trata de un sistema de seguimiento 3D y captura de movimiento, líder en la industria dedicada al diseño de videojuegos, animación, realidad virtual, robótica y ciencias del movimiento, debido al seguimiento óptico de alto rendimiento y tener a los precios más asequibles de la industria, Figura 3.1. La línea de productos OptiTrack incluye software de captura de movimiento y cámaras de seguimiento de alta velocidad y actualmente está siendo utilizado en una gran variedad de mercados que van desde películas y juegos hasta entrenamiento deportivo y biomecánica [18].

Figura 3.1. Optitrack en la industria de los videojuegos.

El sistema de cámaras con el que cuenta la universidad y que se ha usado durante el desarrollo del proyecto, está formado por un conjunto de ocho cámaras del modelo Flex 13, colocadas formando un rectángulo, Figura 3.2.

Figura 3.2. Cámaras Optitrack modelo Flex 13

Este capturador de imágenes de Optitrack ofrece una resolución de 1.3 MP, cuatro veces más que la resolución que tienen otras cámaras pertenecientes a la industria como por ejemplo el modelo Flex 3. Además, tiene una gran capacidad de rastreo cuando se combina con los algoritmos de procesamiento en tiempo real propios de Optitrack, pudiendo rastrear movimientos con una tolerancia de menos de 0.5 mm.

Por otra parte, cada cámara es capaz de alcanzar un área de captura que se encuentra dentro de las más grandes de la industria, lo que permite obtener grandes volúmenes de captura con un número menor de cámaras. Estas características hacen que sea perfecto para aplicaciones de rastreo rigurosas y en seguimientos de movimientos complejos, como es el caso de este proyecto.

Debido a que las cámaras OptiTrack Flex 13 pueden funcionar en modo de escala de grises, el sistema ofrece una mayor calidad en cuanto a capturas de movimientos más pequeñas en tiempo real (como podría ser movimientos faciales en humanos o pequeños cambios de posición en objetos), respecto a otros sistemas de la industria [19].

Además, la captura de movimientos en escala de grises MJPEG se realiza a una velocidad de hasta 120 fotogramas por segundo, una velocidad cinco veces mayor a la velocidad normal de reproducción de las películas. Esto permite tener una mayor precisión en tiempo real durante la obtención de los datos de seguimiento.

3.1.2 SOFTWARE: MOTIVE

Motive es una plataforma de software diseñada para controlar los sistemas de captura de movimiento en varias aplicaciones de seguimiento, Figura 3.3. Motive no solo permite al usuario calibrar y configurar el sistema, sino que también proporciona interfaces para capturar y procesar datos 3D [20].

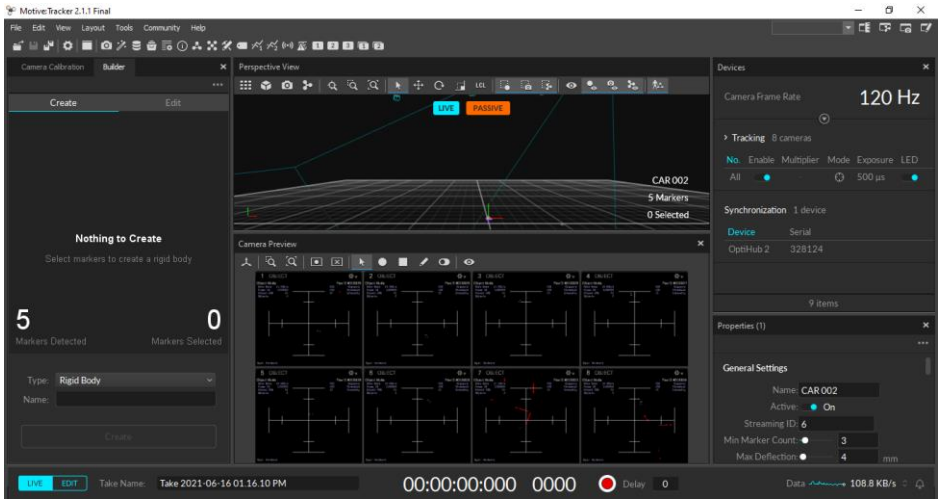

Figura 3.3. Entorno de captura de Motive

Los datos capturados a través del sistema de cámaras se pueden grabar o transmitir en tiempo real a otros programas como es el caso de este proyecto. Cabe destacar que Motive obtiene la información 3D de la posición del objeto a través de la Reconstrucción, que es el proceso de compilar múltiples imágenes 2D de marcadores, colocados en los objetos, para posteriormente obtener sus coordenadas 3D [21].

Para poder llevar a cabo el rastreo del vehículo primero es necesario preparar el entorno de navegación, para ello se debe realizar una calibración y definir el origen de coordenadas. A continuación, es necesario colocar los marcadores en el vehículo y el propio programa se encarga de calcular el centroide del mismo (a través de los marcadores), donde estará el origen de coordenadas del vehículo.

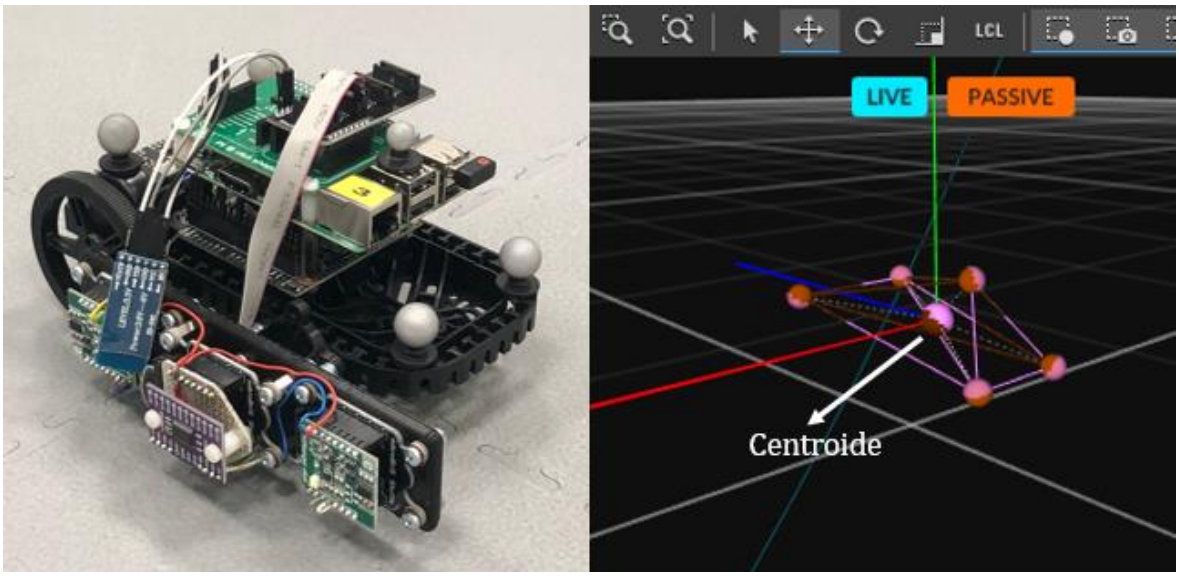

Figura 3.4. Colocación de los marcadores en el vehículo y visualización de los mismos en Motive.

En la Figura 3.4, se puede observar la representación en Motive de los marcadores, y el centroide que calcula basándose en la forma geométrica en la que se han colocado. Debido a esto es importante hacer una colocación correcta de los marcadores, ya que el centroide marcará el origen de coordenadas del sistema de referencia del vehículo

Usando las coordenadas 3D de los marcadores rastreados, Motive puede obtener los datos de los seis grados de libertad (posición y orientación 3D) para múltiples cuerpos rígidos y esqueletos, y permitir el seguimiento de movimientos complejos en el espacio 3D.

Para el desarrollo de este proyecto, únicamente era necesaria la información comprendida en el plano de navegación del vehículo, es decir, las coordenadas en los ejes X e Y y el ángulo de guiñada, que indica el giro del vehículo sobre su eje vertical. La explicación del tratamiento de la información y de la transmisión de datos al vehículo se encuentra explicada con más detalle en el Capítulo 4.

Por último, cabe destacar que ha sido necesario consultar varias guías de uso y fichas técnicas del software y hardware, para poder aprender y controlar el funcionamiento del mismo. Además, se ha desarrollado una pequeña guía de uso de Motive, para facilitar el aprendizaje de futuros alumnos que puedan necesitar usarlo en proyectos o clases de laboratorio; que se encuentra en el ANEXO A: Motive.

3.2 INTRODUCCIÓN AL VEHÍCULO

Durante este proyecto se ha empleado el vehículo basado en el kit de robótica Balboa 32U4 de la empresa Pololu, Figura 3.5, que está formado por dos microcontroladores que se reparten las tareas de control y gestión del vehículo. Los dos microcontroladores son una Raspberry pi

y un ATMega32U4. La Raspberry Pi 3B+ es la encargada de llevar a cabo las tareas de control del robot de alto nivel mientras confía en la placa de control Balboa 32U4 para tareas de bajo nivel, como hacer funcionar motores, leer codificadores e interactuar con otros dispositivos analógicos o sensibles a la sincronización dispositivos.

Figura 3.5. Vehículo basado en el kit de robótica Balboa 32U4 de la empresa Pololu

Además, puesto que el vehículo inicial del kit de Pololu se trata de un vehículo equilibrista se ha añadido el Kit de Conversión de Estabilidad, Figura 3.6, para poder usarlo en modo de navegación. Este kit de estabilidad proporciona un tercer punto de contacto para aplicaciones tradicionales de accionamiento diferencial.

Figura 3.6. Kit de Conversión de Estabilidad

El resultado final, tras añadir el Kit de Conversión de Estabilidad, se muestra en la Figura 3.7.

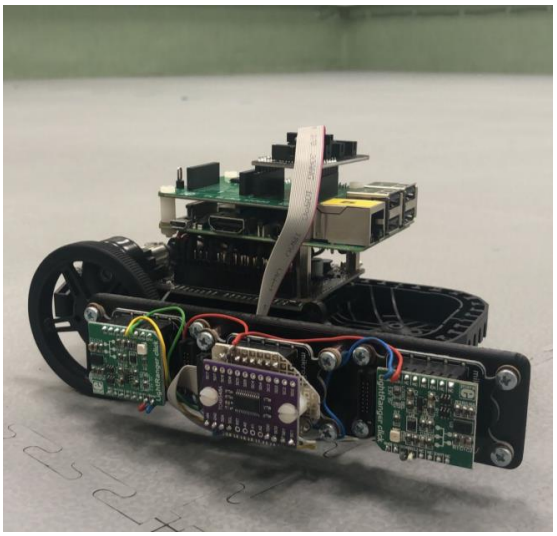

Figura 3.7. Versión final del vehículo empleado en el proyecto

Los sensores que forman parte del hardware del vehículo son los siguientes:

- La unidad de medida inercial (IMU), que está formada por un giróscopo y un acelerómetro ambos de 3 ejes. El acelerómetro mide la aceleración en cada uno de los 3 ejes de coordenadas espaciales, mientras que el giróscopo mide la velocidad angular respecto a los 3 ejes del espacio.

- Dos *encoders* magnéticos que se encuentran ubicados en los ejes de los motores. Estos *encoders* se encargan de medir la velocidad y sentido de giro de los motores, mediante los cuales se estima la velocidad de avance y de giro del vehículo.

Además, el vehículo cuenta con tres pulsadores ubicados en la parte inferior de la placa ATMega, que se emplean como señales digitales de entrada en la máquina de estados.

Por otra parte, los actuadores son los explicados a continuación:

- Dos motores 75:1 Micro Metal Gear Motor HPCB 6V. Se trata de dos motores de DC con escobillas de carbón de alta duración.
- Tres LEDs: amarillo, verde y rojo, ubicados en la placa ATMega.

Por último, es importante mencionar que las comunicaciones entre los diferentes componentes hardware del vehículo (microcontroladores, sensores y actuadores) se realiza a través del bus de comunicaciones I^2C, siendo la Raspberry Pi el máster de la comunicación y el resto de los elementos los esclavos.

3.3 RASPBERRY PI 3B+

La Raspberry Pi es una serie de ordenadores de placa reducida, ordenadores de placa única u ordenadores de placa simple (SBC) de bajo coste desarrollado en el Reino Unido por la Raspberry Pi Foundation, con el objetivo de hacer más accesible a las personas de todo el mundo la informática y la creación digital. La Raspberry Pi está compuesta por un SoC, CPU, memoria RAM, puertos de entrada y salida de audio y vídeo, conectividad de red, ranura SD para almacenamiento, reloj… por lo que se considera un "miniordenador" [22].

Desde la creación de la primera Raspberry Pi hasta la actualidad, el dispositivo ha ido evolucionando y mejorando sus prestaciones, y es por eso que, actualmente forma parte de una gran cantidad de proyectos relacionados la electrónica, automática y domótica.

En concreto la Raspberry Pi 3 B+, Figura 3.8, que es el microcontrolador encargado de la mayor parte de la carga computacional del proyecto, apareció en marzo del 2018 para actualizar el modelo anterior la Raspberry Pi 3 Model B y entre sus mejoras cuenta con un nuevo procesador y mejor conectividad. Concretamente, cuenta con un procesador de cuatro núcleos de 64 bits, 1.4 Ghz de CPU y 1 GByte de memoria SDRAM [23].

Figura 3.8. Microcontrolador Raspberry Pi 3B+

Las tareas que desempeña la Raspberry Pi son las siguientes:

- Carga computacional de la lógica de control.
- *Máster* en la comunicación I^2C con el ATMega32U4 y la IMU.
- Integración de los diferentes elementos Hardware.
- Comunicaciones con la estación base.

El código que permite a las Raspberry Pi llevar a cabo todas estas tareas está recogido en diferentes *scripts* de Matlab, todos ellos integrados en un fichero de Simulink. Este fichero se vuelca en la Raspberry Pi y se ejecuta de forma constante hasta que se decida finalizar el proceso.

3.4 ATMEGA 32U4

3.4.1 DESCRIPCIÓN

El microcontrolador Atmel ATMega32U4, Figura 3.9, es un dispositivo megaAVR de 8 bits basado en la arquitectura RISC mejorada de los AVR. Este dispositivo se he fabricado utilizando la tecnología de memoria no volátil de alta densidad de Atmel. El microcontrolador empleado en este proyecto consta de 32KBytes de memoria flash, además de 2.5 KBytes de RAM a 16MHz.

La placa de control Balboa 32U4 fue diseñada para ser fácil de conectar con una placa única Raspberry Pi y expandir de esta forma la potencia de procesamiento de Balboa[24].

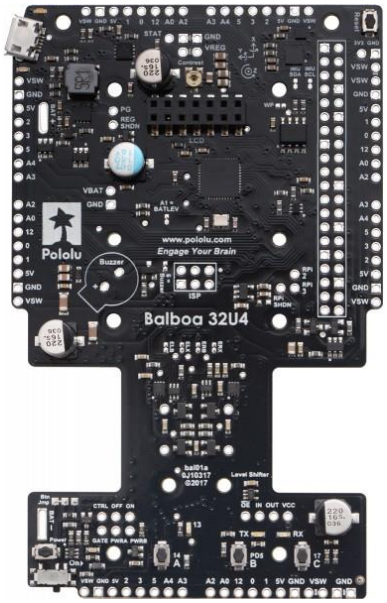

Figura 3.9. Microcontrolador Atmel ATMega32U4

Por otra parte, puesto que la Raspberry Pi es la encargada de realizar las tareas de alto nivel, se le han asignado unas tareas de bajo nivel al ATMega, que se muestran a continuación:

- Interacción con los *encoders*.
- Lectura del estado de los pulsadores de la placa.
- Redacción de la tensión de cada uno de los motores

Las rutinas necesarias para poder llevar a cabo estas tareas han sido programadas en Arduino, y se ejecutan de forma periódica a una frecuencia de 100Hz, por lo que el tiempo de muestreo empleado tanto en este microcontrolador como en el resto del proyecto es de diez milisegundos.

Además, el fabricante de este vehículo cuenta con una serie de librerías de Arduino donde se recogen todas las funciones necesarias para el desarrollo de las tareas y que se han diseñado de forma específica para este microcontrolador. Las librerías son:

- *Balboa32U4.h*: Donde se encuentran las funciones necesarias para poder interactuar con el hardware implementado en la tarjeta del Balboa.
- *PololuRpiSlave.h*: Donde se recogen las funciones para configurar al ATMega como el *esclavo* I^2C de la Raspberry Pi.

El programa consiste definir, escribir y leer una estructura que recoge toda la información necesaria para interactuar con el resto de elementos hardware del vehículo. Los campos que se han incluido en esta estructura son [26]:

- Batería en mV: Campo de tipo int16 escrito por el ATMega, que registra el valor en tiempo real de la tensión proporcionada por las seis baterías que utiliza el robot. Se trata de un valor a tener en cuenta debido al elevado consumo que tiene este proyecto.

- Cuentas del *encoder* izquierdo: Elemento de tipo int16 escrito por el ATMega. Almacena el valor absoluto de las cuentas del *encoder* izquierdo.

- Cuentas del *encoder* derecho: Elemento de tipo int16 escrito por el ATMega. Almacena el valor absoluto de las cuentas del *encoder* derecho.

- Error de lectura del *encoder* izquierdo: Dato de tipo *boolean* escrito por el ATMega, habitualmente denominado "Flag". Se pone a uno cuando se ha producido algún error en la medida del *encoder* izquierdo.

- Error de lectura del *encoder* derecho: Dato de tipo *boolean* escrito por el ATMega, habitualmente denominado "Flag". Se pone a uno cuando se ha producido algún error en la medida del *encoder* derecho.

- Pulsadores A, B y C: Campos de tipo *boolean* escrito por el ATMega, que se activan mientras se presione el pulsador correspondiente.

- Leds amarillo, verde y rojo: Elementos de tipo *boolean* escritos por la Raspberry Pi. El ATMega encenderá los leds para los que la Raspberry Pi haya escrito un 1 en su posición dentro de la estructura.

- Tensión a aplicar al motor izquierdo: Dato tipo int16 entre -300 y +300. Multiplicando este dato por 0.018, se obtiene la tensión que el ATMega debe aplicar al motor izquierdo.

- Tensión a aplicar al motor derecho: Dato tipo int16 entre -300 y +300. Multiplicando este dato por 0.018, se obtiene la tensión que el ATMega debe aplicar al motor derecho.

3.4.2 COMUNICACIÓN ENTRE RASPBERRY PI Y ATMEGA32U4

En cuanto a las comunicaciones, también existe una biblioteca Arduino para la familia de placas 32U4 que les permite actuar como esclavos I^2C y proporciona un marco para la comunicación entre el ATmega32U4 y la Raspberry Pi que actúe como máster.

Cuando se envía la estructura a las Raspberry Pi, esta se recibe como una estructura de dieciocho elementos tipo *uint8,* de tal forma que la información se organiza de la siguiente manera:

Nº Byte	Campo de la estructura
0	Batería mV LSB
1	Batería mV MSB
2	Enconder izquierdo LSB
3	Encoder izquierdo MSB
4	Encoder derecho LSB
5	Encoder derecho MSB
6	Error lectura encoder izquierdo
7	Error lectura encoder derecho
8	Pulsador A
9	Pulsador B
10	Pulsador C
11	LED amarillo
12	LED verde
13	LED rojo
14	Tensión motor izquierdo LSB
15	Tensión motor izquierdo MSB
16	Tensión motor derecho LSB
17	Tensión motor derecho MSB

Debido a la nueva distribución de los datos, es necesario realizar la reconstrucción de los datos del tipo *int16*, de tal forma que se pueda obtener el valor real. De esta forma se usa la función *typecast* de Matlab que recibe como entrada los dos bytes que definen el dato, concatenados en un único vector [LSB MSB], y devuelve el número resultante en 16 bits [26].

3.5 IMU

3.5.1 DESCRIPCIÓN

La IMU (Inertial Measurement Unit, por sus siglas en inglés), Figura 3.10, está formada por un giróscopo y un acelerómetro, ambos actuando sobre los tres ejes. Los dos se encuentran integrados en el chip LSM6DS33. El sensor proporciona seis lecturas de velocidad de rotación

y aceleración independientes cuyas sensibilidades se pueden configurar en los rangos mostrados a continuación, disponibles a través de interfaces I^2C y SPI:

- ± 2/ ±4/±8/± 16 g's para la gravedad
- ± 125 / ±245/±500/±1000/ ± 2000 dps

Para la consecución de este proyecto se han utilizado las escalas de ± 2g y ± 125 dps, para obtener la máxima resolución.

El *CI* LSM6DS33 también incluía una serie de filtros digitales paso bajo, que se configuraron a la frecuencia de corte menos restrictiva (mayor frecuencia de corte), para no alterar las medidas crudas y así poder filtrar las mismas con los filtros programados en Matlab-Simulink [26].

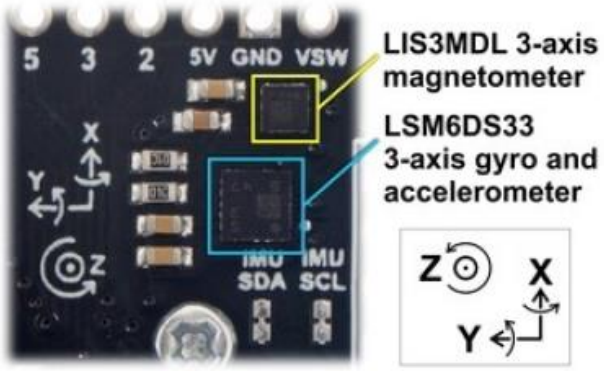

Figura 3.10. Unidad inercial de medidas.

3.5.2 MODO DE EMPLEO DE LA IMU

Como ya se ha mencionado en el Apartado 3.5.1, la IMU contiene un acelerómetro y un giróscopo, encargados de medir respectivamente la aceleración lineal y la velocidad angular en cada uno de los tres ejes.

Por otra parte, tal y como se explica en la Sección 3.3, como el chip LSM6DS33 funciona en modo esclavo I^2C de la Raspberry Pi, la información de estas medidas se recibe a través de I^2C. Estos datos se leen en los registros del 22 al 27 en el caso del giróscopo, y del 28 al 33 para el acelerómetro. Cada pareja de registros se corresponde con la medida en cada uno de los ejes, por lo que la medida real es resultado de la unión de ambos datos como uno único de tipo *int16* [26].

Se realiza un muestreo de la IMU cada diez milisegundos, de esta forma se puede obtener el valor medio de cada medida con diez muestras.

Por último, es importante mencionar que, aunque en la Figura 3.10 aparece un magnetómetro de 3 ejes, este no se empleará en el proyecto. Un magnetómetro es un sensor electrónico que mide y cuantifica las fuerzas magnéticas, es decir se trata de un instrumento encargado de medir la fuerza y dirección de un campo magnético. Esto permite saber el rumbo de la navegación, puesto que actúa como si fuese una brújula, lo que podría ser útil para el proyecto. El problema radica en que, al navegar en el interior de un laboratorio, la presencia de otros campos magnéticos parásitos dificulta su calibración y uso [25].

3.5.3 PROCESAMIENTO DE LA INFORMACIÓN

Tras recibir las medidas, es necesario aplicarles unas ganancias de 9.81/1000 en el caso de la medida del acelerómetro y de π/180/1000 en el caso del giróscopo; para obtener ambas medidas en m/s^2 y en rad/s , respectivamente. Estas medidas cuentan también con un filtrado adicional paso bajo de primer orden, similar al que se usa en el *encoder* magnético, cuya frecuencia de corte es igual a 5Hz.

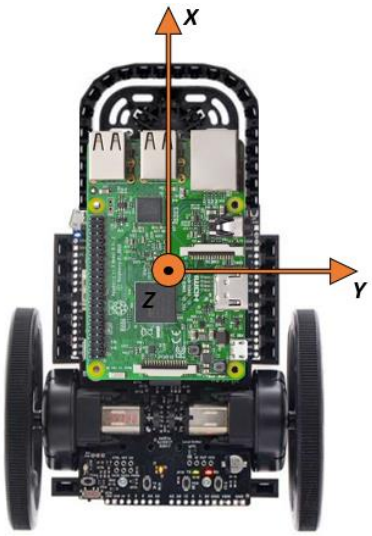

Figura 3.11. Sistema de referencia del acelerómetro del vehículo.

Por otra parte, es necesario transformar las medidas obtenidas a través del bus I^2C, ya que los ejes predeterminados de la IMU no coinciden con los ejes del sistema de referencia del vehículo.

Los valores que resulta interesante medir son la velocidad angular de giro y la aceleración de avance. Estos valores se pueden medir en los ejes X y Z de la Figura 3.11, respectivamente. Para obtener estas medidas transformadas es necesario realizar dos pasos:

1. **Cambio de la orientación del sistema de referencia.** Para que los sistemas concuerden, es necesario realizar un intercambio de los ejes X y Z, de tal forma que se siga obteniendo un triedro ortonormal. Las matrices necesarias son:

   Acelerómetro

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_{Vehículo,} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_{IMU}$$

   Giróscopo

$$\begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix}_{Vehículo,} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix}_{IMU}$$

   Siendo $[a_x\ a_y\ a_z]_{Vehículo,}$ y $[g_x\ g_y\ g_z]_{Vehículo,}$ los valores medidos a través de la IMU, expresados en el sistema de referencia del vehículo.

El elemento (3,3) de la matriz de rotación del acelerómetro y del giróscopo no son coincidentes, debido a que el código para la implementación se había diseñado de forma previa para ese sistema de referencia.

3.6 MOTORES

3.6.1 DESCRIPCIÓN

El vehículo tiene como actuadores principales dos motores HPCB de 6V. El modelo exacto es 75:1 Micro Metal Gear Motor HPCB 6V, Figura 3.12. Este modelo proporciona una relación de transmisión de 75.81:1, y cuenta con una resistencia interna de valor 4 Ω.

Figura 3.12. Motor 75:1 Micro Metal Gear Motor HPCB 6V.

Este motorreductor es un motor de CC, que es capaz de girar a 430 rpm sin carga y con una alimentación de 6 V, consumiendo una corriente de 100mA. Para la situación de motor bloqueado, este ofrece un par de 1.1 $kg \cdot cm$ consumiendo una corriente de 1.5 A [27].

En términos de tamaño, esta versión con escobillas de carbón (HPCB) cuenta con unos terminales y tapas de extremo, cuyas dimensiones son ligeramente diferentes a las versiones con escobillas de metales [28].

Para el control de ambos motores, se emplea el *driver* DRV8838 de la empresa estadounidense *Texas Instruments.* El microcontrolador ATMega emplea los siguientes pines para el control del *driver* [29]:

- El **pin digital 15**, o PB1: controla el sentido de giro del motor derecho (LOW impulsa el motor en sentido positivo, ALTO lo impulsa en dirección contraria).
- El **pin digital 16**, o PB2: controla la dirección del motor izquierdo (Se comporta de la misma forma que el pin digital 15).
- El **pin digital 9**, o PB5: controla la velocidad del motor derecho con el PWM (modulación de ancho de pulso) generado por el Timer1 del ATmega32U4.
- El **pin digital 10**, o PB6: controla la velocidad del motor izquierdo con el PWM (modulación de ancho de pulso) generado por el Timer1 del ATmega32U4.

Para el desarrollo del proyecto se ha tomado como sentido de giro positivo el correspondiente al caso que hace avanzar al vehículo en el sentido en el que se encuentra ubicada la tapa de la batería.

En la Figura 3.13 se puede observar la curva del motor que tiene el vehículo que se ha empleado en el proyecto.

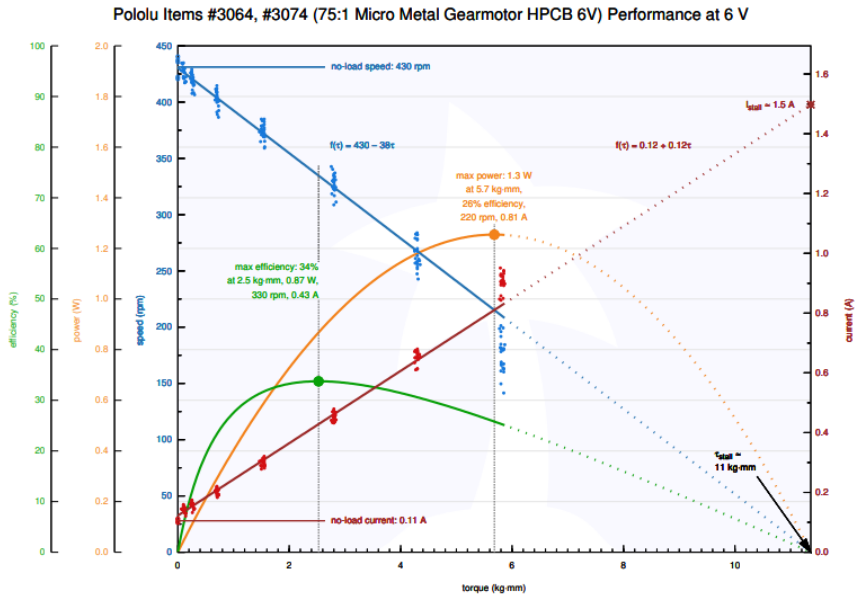

Figura 3.13. Curva de caracterización del motor.

La familia de dispositivos DRV883x son controladores de *Puente en H* que pueden impulsar motores de CC u otros dispositivos como solenoides. Estos dispositivos integran los *drivers* de FET necesarios y los circuitos de control de FET en un solo dispositivo.

Además, esta familia de dispositivos DRV883x agrega características de protección más allá de las implementaciones discretas tradicionales: bloqueo por infratensión, protección contra sobrecorriente y apagado por sobretemperatura. Estas protecciones saltan cuando: la tensión de alimentación es menor de 1.7V, la corriente de salida es superior a 1.9A y la temperatura excede los 150ºC.

El *driver*, Figura 3.14, está formado por cuatro FETs con pequeña resistencia parásita (alrededor de 2.8 m) y con un diodo de recirculación. Soportan una corriente máxima de 1.8 A de corriente directa. Este *driver* está especialmente indicado para alimentar cargas tales como cámaras, lentes DSLR, robótica, juguetes, equipos médicos y cargas similares que se comporten como solenoides [29].

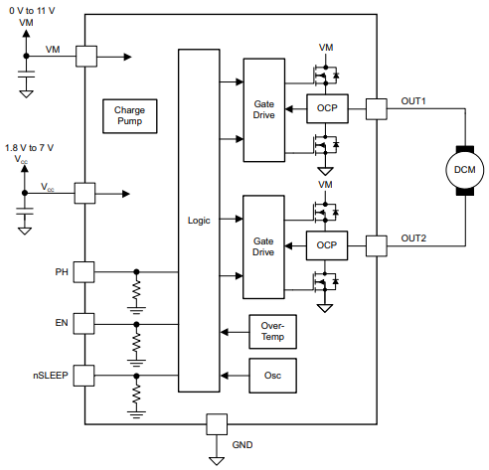

Figura 3.14. Driver del motor.

3.6.2 FUNCIONAMIENTO DEL DRIVER DRV8838

Para el control de las salidas del *driver*, en lugar de usar el PWM, estas se controlan mediante una interfaz Fase/Habilitar. Este sistema tiene una ventaja frente al tradicional PWM que le convierte una mejor opción: a diferencia que el PWM donde se controla cada salida con un pin de entrada; la interfaz de Fase/Habilitar emplea únicamente dos pines. El pin xPHASE se encarga de controlar la dirección de la corriente y el pin xENBL para habilitar o deshabilitar el puente en H.

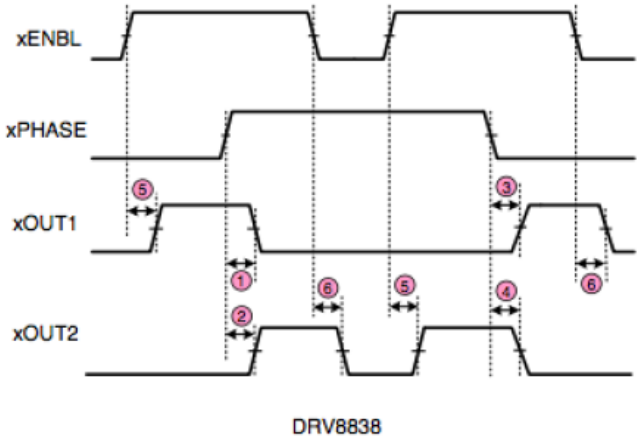

Figura 3.15. "Timing" de las entradas y salidas del driver DRV8838.

El funcionamiento del dispositivo se muestra en la Figura 3.15. Cuando el pin de habilitación está en nivel alto y el pin de fase a nivel bajo, el pin *xOUT1* se pone a nivel alto,

mientras que el pin *xOUT2* se pone a nivel alto cuando tanto el pin de habilitación como el de fase están a nivel alto.

Cuando el pin *xOUT1* está a nivel alto, la corriente llegar al motor en sentido directo. En cambio, si es el pin *xOUT2* el que se encuentra a nivel alto, el motor es alimentado con corriente inversa.

La corriente deja de llegar al motor cuando el pin de habilitación está a nivel bajo, en cuyo caso tanto el pin *xOUT1* como el pin *xOUT2* se encuentran a nivel bajo.

En la Figura 3.15, aparecen unos círculos de color rosa que muestran los diferentes retrasos que se producen durante la conmutación del estado de los diferentes pines. La implementación de este *driver* en la placa de ATMega32U4 se muestra en la Figura 3.16.

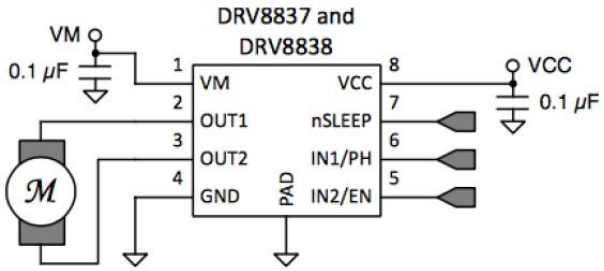

Figura 3.16. Aplicación del driver DRV8838 a un motor de corriente continua.

3.7 ENCODERS

3.7.1 DESCRIPCIÓN

El vehículo empleado en el proyecto tiene dos *encoders* de cuadratura magnéticos, que se encuentran en los ejes de los motores, Figura 3.17. Estos *encoders* cuentan con dos discos magnéticos sujetos a una extensión del eje de giro del motor y un par de sensores de efecto Hall que proporcionan 12 cuentas por revolución del eje del motor, ubicados en la placa [30].

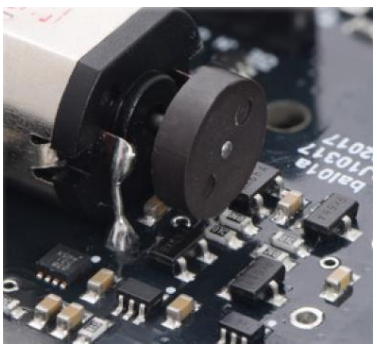

Figura 3.17. Colocación del disco magnético y los sensores.

Los sensores de efecto Hall son un tipo de transductores que generan una tensión de salida cuando se encuentran en presencia de campo magnético variable. Los *encoders* de cuadratura del kit Balboa funcionan gracias al principio de efecto Hall, generando dos señales cuadradas desfasadas 90^0 entre sí.

Los *encoders* de cuadratura se controlan normalmente a través de la monitorización de ambos canales, canales A y B en la Figura 3.18, directamente. Pero en el caso de este proyecto, se usa un XOR entre ambos canales, para obtener de esta forma un único pin de interrupción. Por otra parte, se conecta un segundo pin al canal B, lo que permite reconstruir la señal del canal A mediante una operación XOR entre ambos pines [26]:

$$(A\ xor\ B)\ xor\ B = A$$

El microcontrolador ATMega32U4 es el encargado de realizar la tarea de lectura y obtención de las medidas de los *encoders*, que posteriormente se incluyen dentro de la estructura de campos que se describe y explica en la Sección 3.4.

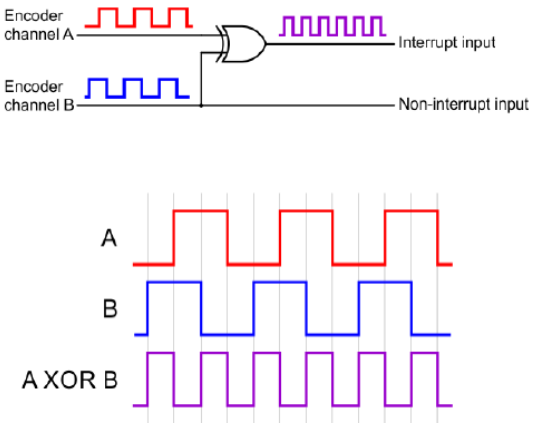

Figura 3.18. Lectura de los canales de los encoders.

3.7.2 ESTIMACIÓN DE LA VELOCIDAD DE AVANCE Y VELOCIDAD ANGULAR DE GUIÑADA

En la Sección 3.4, se explica la transmisión de datos a la Raspberry, que recibe dos bytes con las medidas o cuentas de ambos *encoders*. Tras procesar las medidas *crudas* se obtienen las velocidades lineales común y diferencial.

Inicialmente se convierte el valor que se recibe a través del bus de datos I^2C de *cuentas de encoder* a grados. Para realizar esta transformación, es necesario calcular la resolución del

encoder mediante la información que se encuentra en el *datasheet* del Balboa32U4[24]. Se obtiene la ecuación:

$$r_{enc} = 12 \cdot m_{rt} \cdot r_{rt}$$

Donde cada elemento es:

- r_{enc}: La resolución del *encoder.*
- m_{rt}: La relación de transmisión del motor.
- r_{rt}: La relación de transmisión de la rueda.

Por otra parte, el valor de la medida se puede obtener en grados mediante:

$$enc_g = enc_c \cdot {}^{360}/_{r_{enc}}$$

Donde cada elemento es:

- enc_g: Valor del *encoder* en grados.
- enc_c: Valor del *encoder* en cuentas.
- r_{enc}: La resolución del *encoder*.

A continuación, la **velocidad** común o de **avance** se puede obtener mediante:

$$\boldsymbol{v_c} = \frac{\Delta enc_l + \Delta enc_r}{2} \cdot \frac{1}{t_s} \cdot \frac{\pi}{180} \cdot R$$

Donde cada elemento es:

- v_c: Velocidad común (de avance) del vehículo.
- Δenc_l: Incremento del valor del *encoder* izquierdo.
- Δenc_r: Incremento del valor del *encoder* derecho.
- R: El radio de la rueda $[m]$.
- t_s: El tiempo de muestreo $[s]$.

Por último, la **velocidad** diferencial o **de guiñada** se obtiene a través de:

$$\boldsymbol{v_d} = (\Delta enc_l - \Delta enc_r) \cdot \frac{1}{t_s} \cdot \frac{\pi}{180} \cdot R \cdot \frac{1}{D}$$

Donde cada elemento es:

- v_d: Velocidad diferencial (de giro) del vehículo.
- Δenc_l: Incremento del valor del *encoder* izquierdo.
- Δenc_r: Incremento del valor del *encoder* derecho.
- R: El radio de la rueda $[m]$.

- t_s: El tiempo de muestreo [s].
- D: La distancia de separación entre las ruedas [m].

Como se buscaba que en el sentido horario (sentido positivo del eje Z), la velocidad de giro resultara positiva, esta se ha definido como el incremento del *encoder* izquierdo menos el derecho.

Capítulo 4. COMUNICACIONES

El objetivo de esta sección es describir y explicar el diseño que se ha realizado del sistema de comunicaciones del proyecto. Para el correcto funcionamiento del proyecto ha sido necesario crear una estructura de comunicación robusta, que permita en todo momento establecer las comunicaciones entre el ordenador receptor de la información proveniente de las cámaras, el vehículo y el ordenador encargado de monitorizar la navegación.

En primer lugar, se va a realizar una explicación de la transmisión de datos del sistema de localización externo a Simulink, programa encargado del posterior tratamiento de estos datos para el correcto uso de los mismos en etapas posteriores del proyecto.

A continuación, se va a explicar la comunicación entre el ordenador receptor de la información proveniente de las cámaras y el vehículo. Ha sido necesario realizar varias pruebas y ensayos iniciales, para poder obtener el sistema Bluetooth final. En este apartado, se explicará el funcionamiento del decodificador necesario para recibir los datos de las cámaras en el vehículo.

Por último, se explica la transmisión de información vía Wifi entre el vehículo y el ordenador encargado de monitorizar y desde el cual se vuelca el código de control al propio vehículo.

4.1 COMUNICACIÓN ENTRE MOTIVE Y SIMULINK:

Como se ha comentado en la Sección 3.1 de este documento, Motive es la herramienta software a partir de la cual se lleva a cabo el control de las cámaras que forman el sistema de localización externa de este proyecto. A través de Motive se realiza el seguimiento del vehículo y se obtienen las coordenadas del mismo en el espacio tridimensional en tiempo real.

Estas coordenadas son los datos necesarios para diseñar el sistema de navegación del vehículo, por lo que es necesario transmitirlas al vehículo para que posteriormente este las envíe al ordenador donde se encuentran los códigos de control y de navegación.

El software Motive debe ser empleado por el usuario para calibrar y configurar el sistema, crear los objetos tridimensionales, y procesar en 3D los datos obtenidos. Junto con Motive, OptiTrack facilita el entorno de desarrollo *NatNet*, que cuenta con un conjunto de herramientas cliente-servidor que emplean protocolo UDP (en inglés, *User Datagram Protocol*) y comunicación de tipo *multicast* o *unicast*. El protocolo UDP permite la transmisión de datos sin conexión previa; de esta manera, es posible enviar información de una forma muy rápida, sin necesidad de esperar la confirmación de la conexión.

Empleando estas herramientas pueden enviarse los datos registrados en las pruebas de navegación desde un servidor (Motive) a aplicaciones cliente (en este proyecto, Matlab-Simulink).

4.1.1 SIMULINK PARA LA RECEPCIÓN Y ENVÍO DE INFORMACIÓN DEL SISTEMA DE LOCALIZACIÓN

Para la recepción, tratamiento de los datos de las cámaras y su posterior envío al vehículo se ha empleado el fichero de Simulink *MCS_CONTROL_STATION,* de la Figura 4.1.

La información se almacena en el bus de *Control*, que es el *Data Store Memory* que aparece en la parte superior del diagrama. Trabajar con buses permite tener controladas y almacenadas todas las variables para poder acceder a ellas y modificarlas desde cualquier otra parte del programa. Esto implica que los buses se inicializan en los scripts de configuración, permitiendo definir sus dimensiones y tipo de dato almacenado.

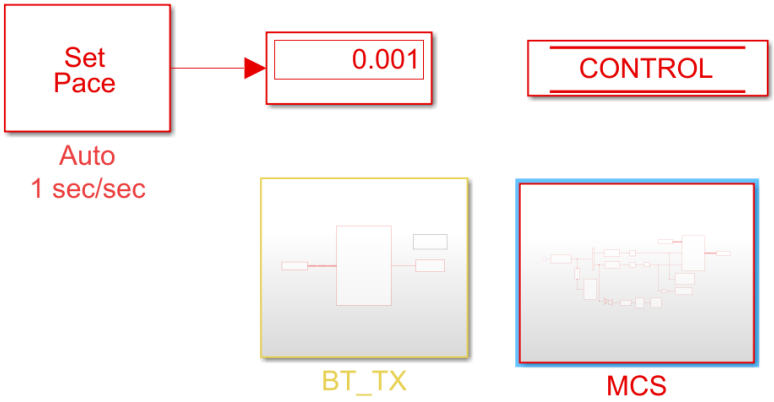

Figura 4.1. Diagrama de Simulink MCS_CONTROL_STATION.

- El subsistema de *BT_TX* es el encargado de realizar el envío de las coordenadas en tiempo real, a la Raspberry Pi del vehículo vía Bluetooth. Su funcionamiento se explica en la Sección Capítulo 1. de este documento.

- El subsistema de *MCS,* Figura 4.3, es el encargado de obtener las coordenadas del vehículo en el espacio 3D a través de la función *FuncMotive* que conecta el programa de Motive y Simulink permitiendo un envío continuo de datos de posicionamiento.

Para recibir los datos en Simulink desde Motive se emplea una función (*FuncMotive*) que utiliza la API facilitada por *NatNet*, que permite obtener las características del cuerpo rígido detectado por las cámaras. A partir de estas herramientas se obtienen las posiciones tridimensionales y los cuaterniones que definen la orientación espacial del vehículo. Estos últimos son transformados en ángulos de Euler, tal y como se observa en la Figura 4.2.

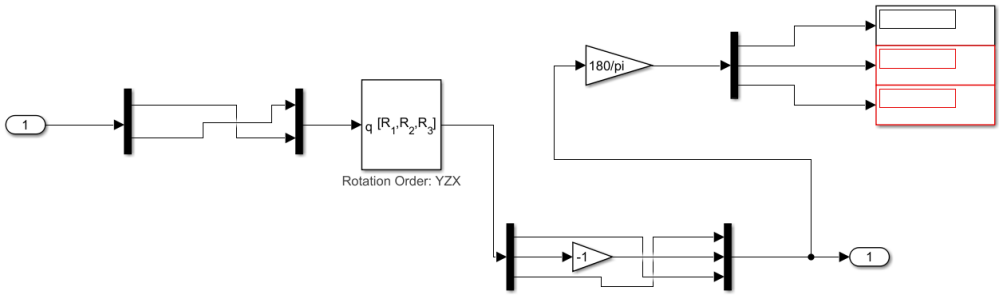

Figura 4.2. Bloque de Simulink "Euler Angles".

Aunque a través de *FuncMotive* se obtienen los valores de la posición del vehículo respecto a los tres ejes de coordenadas y los tres ángulos de Euler, más conocidos en navegación como los ángulos de *Tait-Bryan;* únicamente necesitamos las coordenadas de los ejes X e Y y el ángulo de guiñada. Por eso, se emplea un selector de la información, como se observa en la Figura 4.3, para guardar en el bus de *Control* solo la información que resulta interesante.

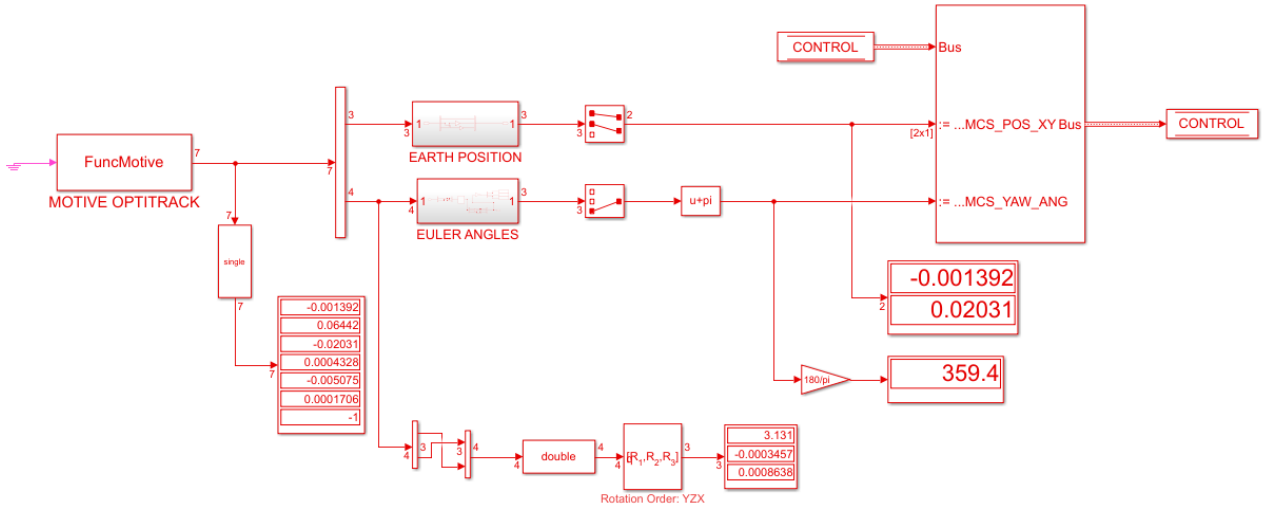

Figura 4.3. Subsistema "MCS" del diagrama MCS_CONTROL_STATION.

Cabe mencionar que el sistema de referencia que se emplea en Motive es *y-up right-hand coordinate system,* Figura 4.4, por lo que ha sido necesario adaptar los datos obtenidos por las cámaras y transformarlos al sistema de referencia del vehículo.

Para ello los datos obtenidos de la posición respecto al eje Z según Motive, se han invertido, para obtener así los datos respecto al eje Y del sistema de referencia del vehículo, Figura 4.5.

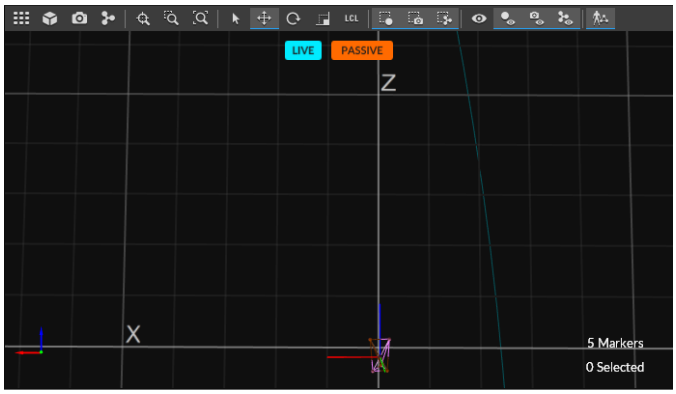

Figura 4.4. Sistema de referencia de Motive.

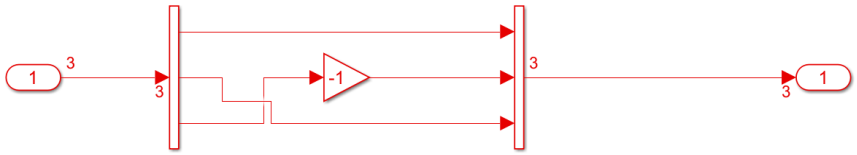

Figura 4.5. Transformación de los datos recibidos al sistema de referencia del vehículo.

4.2 TRANSMISIÓN BLUETOOTH DEL ORDENADOR DE RASTREO A LA RASPBERRY PI

En esta sección se explica de forma detallada el sistema de comunicación empleado para transmitir los datos de las coordenadas de las cámaras a la Raspberry Pi. Para ello se realizaron pruebas enviando únicamente un byte, para posteriormente modificarlo y poder enviar las tres medidas: coordenadas en los ejes X e Y y el ángulo de guiñada.

4.2.1 PRUEBAS INICIALES

Inicialmente se decidió probar la comunicación Bluetooth entre una Raspberry Pi externa al vehículo y el ordenador, enviando un único byte y comprobando que este se recibía de nuevo en el ordenador, demostrando así el correcto funcionamiento de la comunicación una vez cerrado el lazo.

Para poder realizar las pruebas y ensayos, fue necesario configurar una nueva Raspberry Pi desde cero. Además, para poder llevar a cabo la transmisión de datos vía Bluetooth fue necesario configurar dos módulos Bluetooth HC05 para que actuasen como *esclavo* y como *máster* mediante Arduino. Toda la información necesaria para llevar a cabo estos pasos se

documentó para que futuros alumnos pudiesen usarla en sus proyectos; se encuentra recogida en el ANEXO B: Configuración de la Raspberry Pi.

Para llevar a cabo el envío del byte, se emplearon dos ficheros de Simulink. Uno de ellos era el que se volcaba en la Raspberry Pi a la que se debe conectar el módulo Bluetooth HC05 que actúa como *esclavo,* Figura 4.6, mientras que el otro fichero enviaba el byte y lo recibía de nuevo, actuando como ordenador de monitorización.

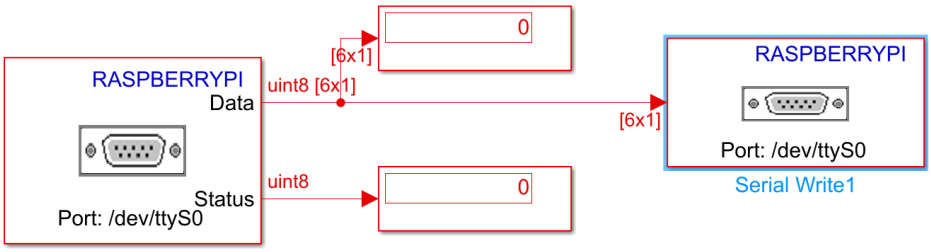

Figura 4.6. Diagrama de Simulink de la Raspberry Pi.

Una vez comprobada la comunicación y que el envío de un byte se realizaba de forma correcta. Se modificó el fichero de Simulink para que se enviara 6 bytes, para comprobar entre otras cosas, que estos llegaban siempre en el mismo orden en el que se enviaban.

En la prueba final que se realizó, se enviaron 6 bytes: [25, 26, 27, 28, 29, 30] desde el ordenador, y se recibieron de la misma forma en el ordenador de nuevo, Figura 4.7.

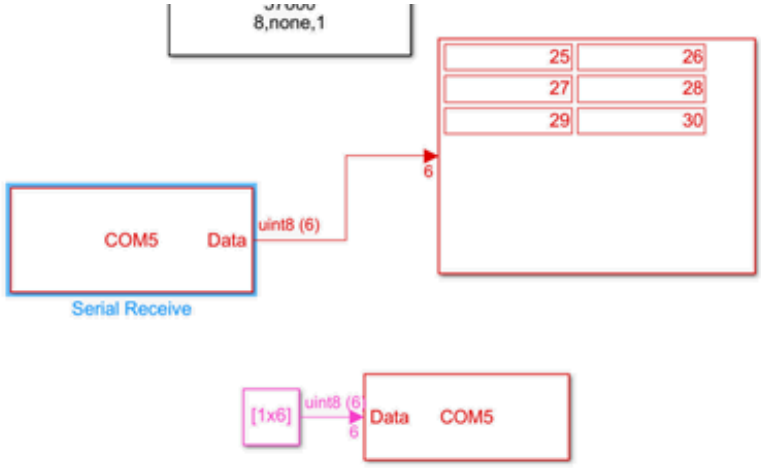

Figura 4.7. Prueba de envío de 6 bytes.

4.2.2 PRUEBA FINAL DE LA TRANSMISIÓN VÍA BLUETOOTH

Una vez comprobado el correcto funcionamiento de los módulos Bluetooth a través de las pruebas iniciales, ha sido necesario diseñar el subsistema del fichero de Simulink *MCS_CONTROL_STATION,* denominado *BT_TX,* Figura 4.8, necesario para enviar los valores de los datos medidos a la Raspberry Pi del vehículo.

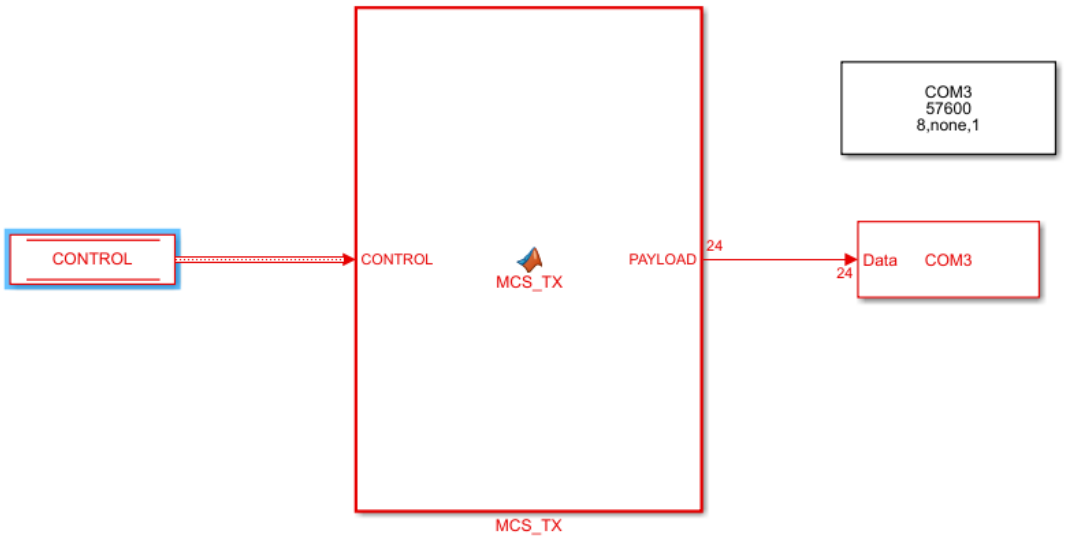

Figura 4.8. Subsistema "BT_TX" del diagrama MCS_CONTROL_STATION.

El bloque *MCS_TX* se trata de un codificador que recibe del bus de *Control* los tres valores de medidas de la posición X, Y y el ángulo de guiñada de tipo *double*, y los convierte a tipo *unit8* para poder realizar el posterior envío a la Raspberry Pi.

Para probar el buen funcionamiento del subsistema *BT_TX* se conectaron los módulos Bluetooth a la Raspberry Pi, el que trabajaba como *esclavo,* Figura 4.9, y finalmente al ordenador el que trabajaba como *máster* en la comunicación mediante un conversor USB UART.

Para realizar la comprobación final fue necesario cerrar por completo el bucle y diseñar la parte de recepción de la información de la Raspberry Pi en un subsistema, *BT_RX* dentro del fichero de Simulink que se vuelca en la Raspberry Pi, *CAR_CONTROL_SYSTEM* que se explica de forma detallada en el Capítulo 5. Por otra parte, la descripción del funcionamiento del subsistema de transmisión Bluetooth se explica en la Sección 4.3.

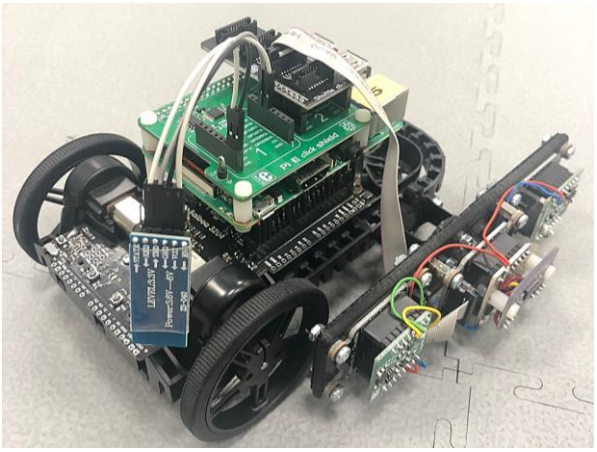

Figura 4.9. Conexión del módulo bluetooth que actúa como esclavo a la Raspberry Pi.

4.3 RECEPCIÓN BLUETOOTH DE LA RASPBERRY PI

En esta sección se explica el funcionamiento de la parte receptora del sistema de comunicación Bluetooth que existe entre el ordenador y la Raspberry Pi.

Para el funcionamiento de esta, ha sido necesario incluir dentro de las comunicaciones del vehículo un bloque de recepción de información Bluetooth, *BT_RX* de la Figura 4.10, para que la Raspberry Pi del vehículo sea capaz de recibir los datos que se transmitan desde el ordenador de las cámaras.

Para ello ha sido necesario el uso de un decodificador, *MCS_DECODER*, cuyo funcionamiento se explica en el Apartado 4.3.1.

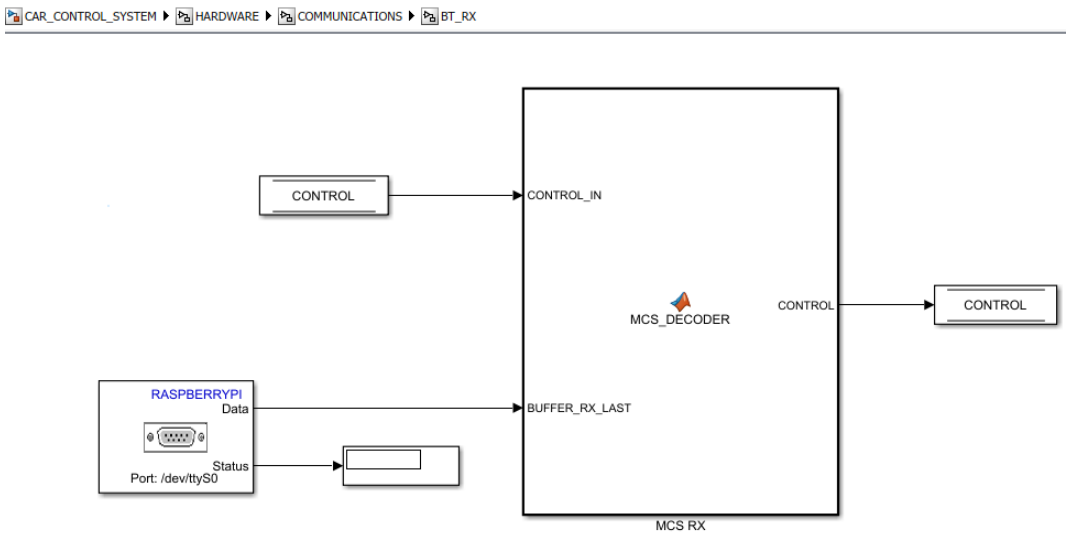

Figura 4.10. Subsistema "BT_RX" del diagrama CAR_CONTROL_SYSTEM.

4.3.1 FUNCIONAMIENTO DEL DECODIFICADOR

Para que la comunicación se pueda realizar de manera correcta ha sido necesario el uso de un decodificador que reciba los datos provenientes de la Raspberry Pi y los almacene en un buffer de 200 bytes.

De esta forma se asegura que el mensaje llega correctamente y no falta o sobra información, ya que sin este formato la información se recibía de forma desordenada e inconexa.

El funcionamiento de este decodificador es el siguiente: el buffer del decodificador, cuya capacidad es de 200 bytes, se va rellenando mientras busca el mensaje. La Raspberry Pi envía un mensaje de 50 bytes entre los cuales se encuentran los 24 bytes que almacenan los datos de la posición X e Y y el ángulo de guiñada precedidos de un *header.* El resto de los 50 bytes se rellenan con ceros.

El decodificador busca entre los bytes que se almacenan en el buffer de 200 bytes el *header* que marca el inicio del mensaje, una vez detecta 50 bytes, incluyendo el *header* en la primera posición, el decodificador ha encontrado el mensaje por lo que lo guarda y vacía el buffer de 200 bytes para prepararlo para la siguiente recepción de datos. Este proceso se repite de forma continua.

4.3.2 COMPROBACIÓN DEL FUNCIONAMIENTO DEL DECODIFICADOR

Una vez creado el código del decodificador, se realizó una prueba inicial en la que no intervino la comunicación Bluetooth, es decir, la prueba se llevó a cabo sin usar la Raspberry Pi. El objetivo era comprobar que el codificador funcionase según lo esperado.

Para ello se empleó un fichero de Simulink que no forma parte del conjunto de ficheros del proyecto si no que se creó exclusivamente para realizar esta prueba, Figura 4.11.

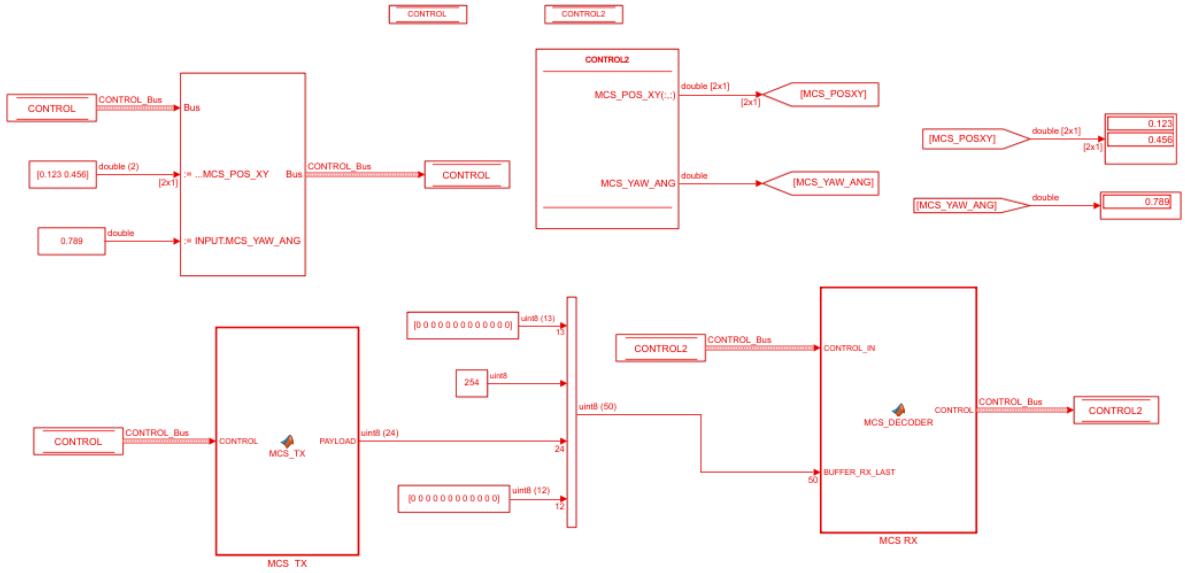

Figura 4.11. Diagrama de Simulink de prueba del funcionamiento de decodificador sin el uso del Bluetooth.

En la Figura 4.12 se puede observar que el fichero cuenta con un bus de *Control* en el cual se almacenan tres valores de tipo *double*, que se corresponden con los valores de la posición X e Y y el ángulo de guiñada que se obtienen a partir del sistema de localización externo. Para la prueba inicial, se escogieron tres valores aleatorios: 0.123, 0.456 y 0.789.

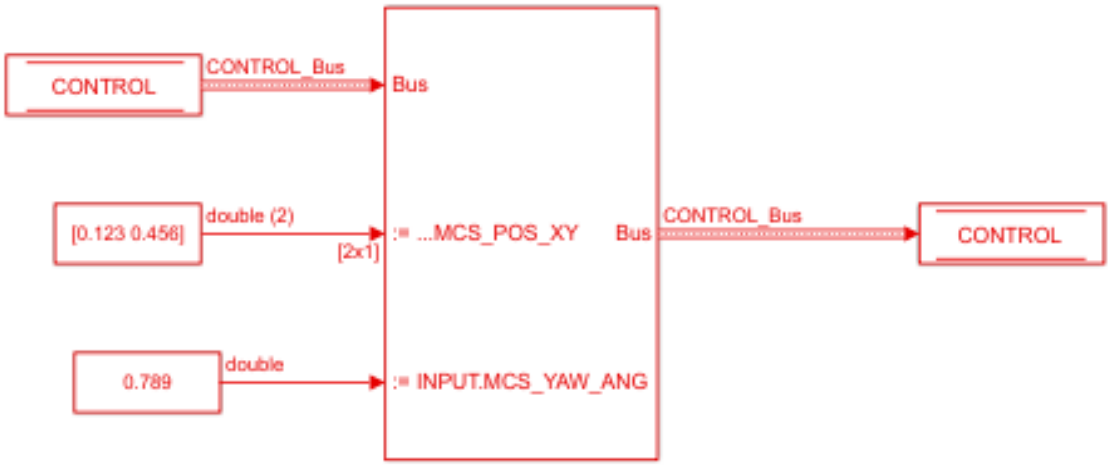

Figura 4.12. Almacenamiento de los tres valores aleatorios en el bus de Control.

El bloque de *MCS_TX* se corresponde con el codificador empleado en el fichero de *MCS_CONTROL_STATION* que se explica en la Sección 4.2. Los tres valores se guardan en un vector *PAYLOAD* de tipo *uint8* y de tamaño 24, Figura 4.13. Tal y como se ha explicado, el

decodificador debe recibir un vector de tamaño 50 bytes en el cual se encuentre el *PAYLOAD* introducido por un '254', el *header* que marca el inicio del mensaje de envío. Para esta prueba se introdujo el *PAYLOAD* en una posición central y se rellenó el resto del vector con ceros.

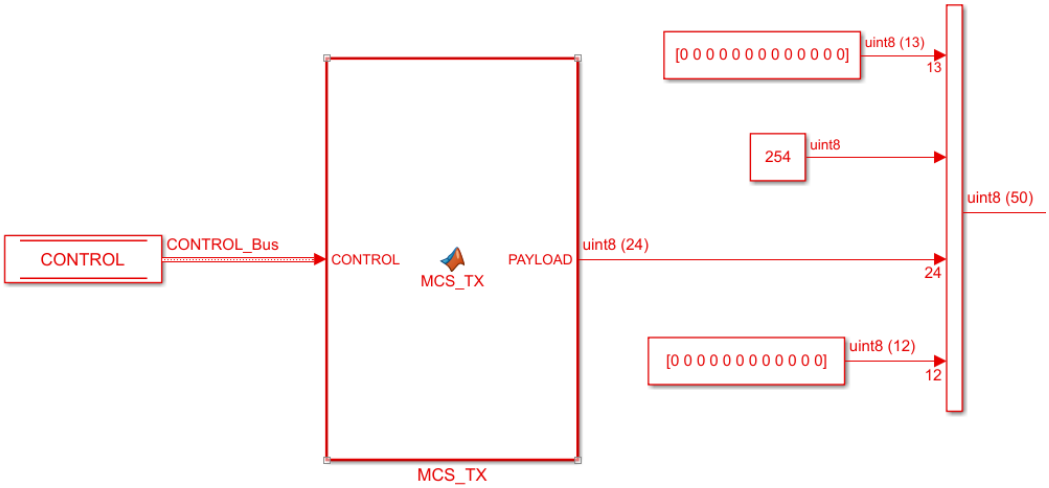

Figura 4.13. Codificador empleado en la transmisión.

A la entrada del decodificador se recibe un vector de 50 bytes entre los cuales se encuentran los valores que se han enviado. El decodificador se encarga de "encontrarlos" y almacenarlos en el bus *Control2*, Figura 4.14.

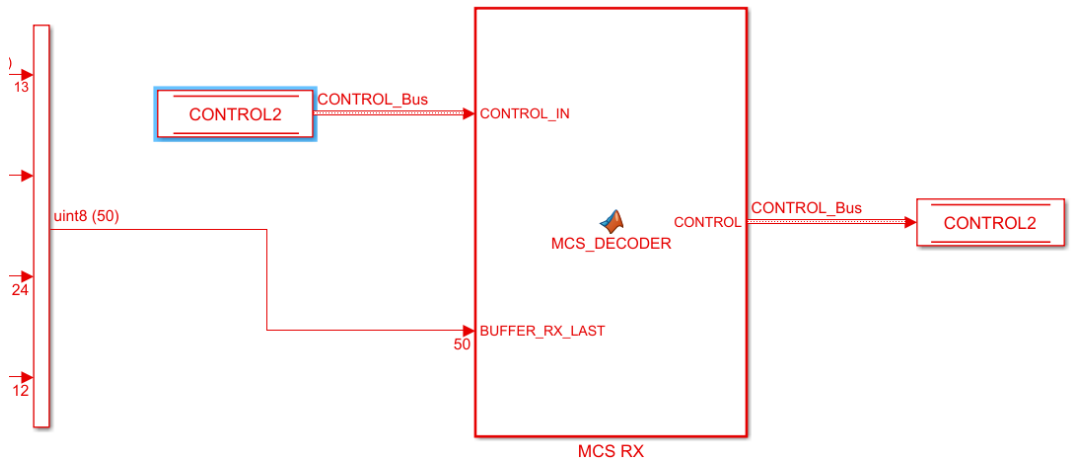

Figura 4.14. Decodificador empleado en la recepción.

Para finalizar la prueba, se seleccionan del bus de control las medidas que nos interesan: *MCS_POS_XY* (donde se encuentran los valores de las coordenadas en los ejes X e Y del centro

de masas del vehículo) y *MS_YAW_ANG* (donde se encuentra el valor del ángulo de guiñada medido por las cámaras). Estos valores se muestran en unos *displays* y se observa que efectivamente coinciden con los valores iniciales que se han escogido: 0.123, 0.456 y 0.789, Figura 4.15. Se comprueba así el correcto funcionamiento del decodificador.

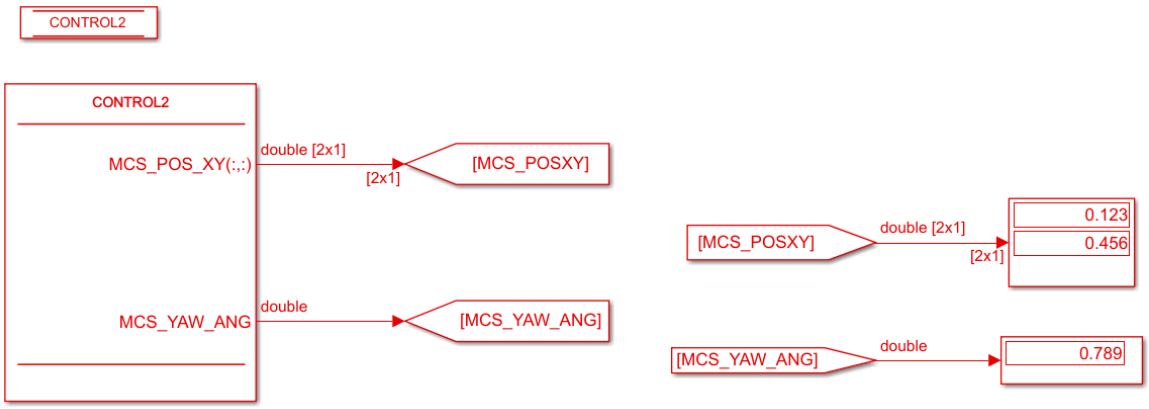

Figura 4.15. Comprobación de los valores recibidos.

4.3.3 PRUEBA FINAL. CERRANDO EL LAZO

Una vez comprobado el correcto funcionamiento del decodificador empleado en el subsistema *BT_TX*, se realizó una prueba con Bluetooth, incluyendo a la Raspberry Pi con el objetivo de cerrar el lazo de transmisión de información y verificar de forma definitiva el funcionamiento correcto de este sistema.

Para ello, de nuevo se han empleado dos ficheros de Simulink que no se encuentran dentro del conjunto de ficheros del proyecto, sino que se han creado exclusivamente para realizar esta prueba y que han servido para crear la versión final y oficial del sistema de comunicación Bluetooth del vehículo.

El fichero del ordenador es aquel que envía los valores a la Raspberry Pi, Figura 4.16. Para realizar la prueba, se envían tres valores aleatorios que se corresponderían con las posiciones X e Y y el ángulo de guiñada medidos por las cámaras. Los valores escogidos son: 0.987, 0.654 y 0.321.

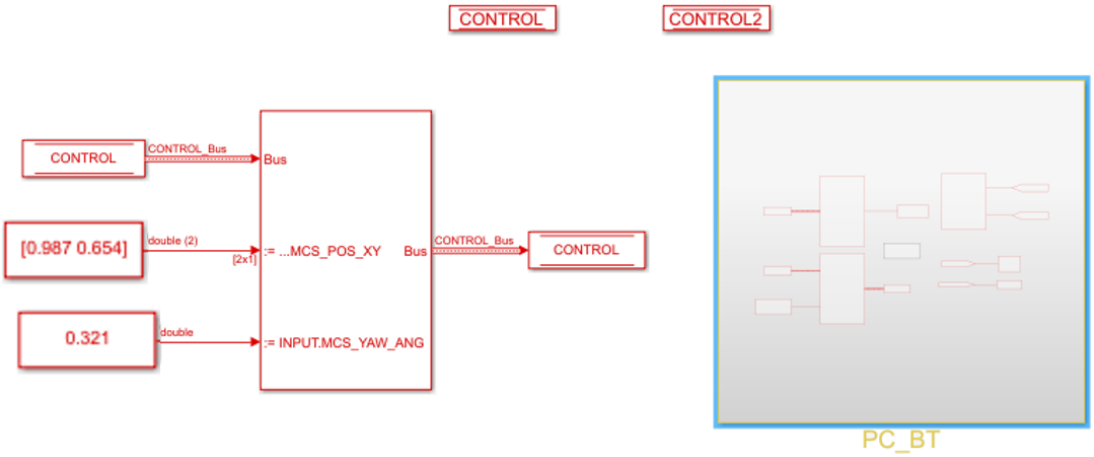

Figura 4.16. Diagrama de Simulink de transmisión Bluetooth desde el ordenador.

Estos valores de almacenan en el bus de *Control*, y tras pasar por el codificador *MCS_CODER* se envían vía Bluetooth a la Raspberry Pi, Figura 4.17.

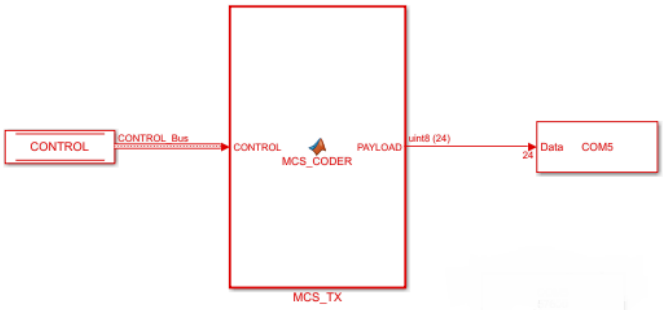

Figura 4.17. Transmisión de datos a la Raspberry Pi.

El segundo fichero es aquel que se vuelca a la Raspberry, donde llegan los datos y tras decodificarse a través del *MCS_DECODER*, se almacenan en el bus de *Control2*, Figura 4.18.

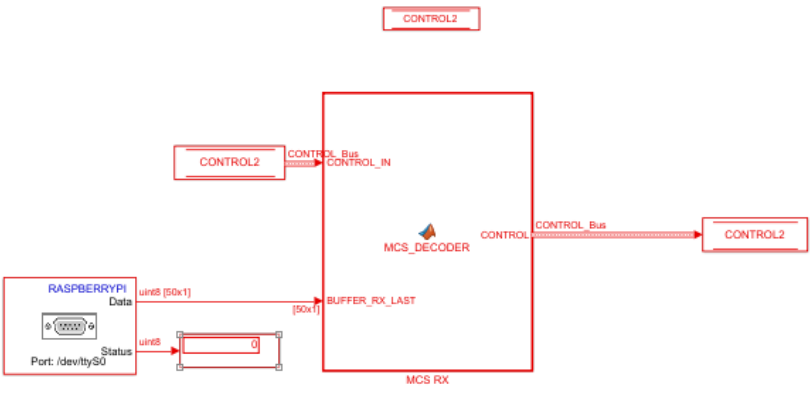

Figura 4.18. Diagrama de recepción de la Raspberry Pi.

Para cerrar el lazo, estos datos almacenados en el bus *Control2* deben enviarse de nuevo al ordenador, con el objetivo de poder visualizarlos, Figura 4.19. Por lo tanto, es necesario codificar de nuevo los datos mediante el codificador *MCS_CODER* y concatenarlos con el '254' que marca el inicio, para que el decodificador del ordenador pueda funcionar.

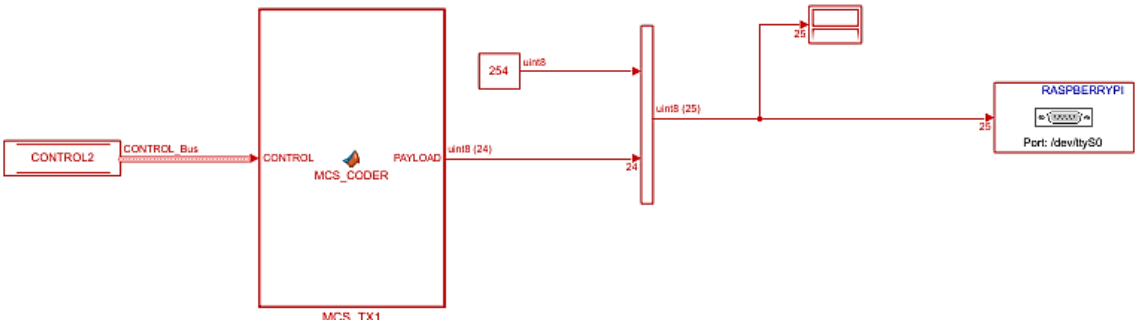

Figura 4.19. Envío desde la Raspberry Pi de los datos recibidos.

Por último, en el subsistema del fichero del ordenador de *PC_BT*, Figura 4.20, se reciben los datos de la Raspberry Pi y tras decodificarlos, estos se almacenan en el bus *Control2* del fichero del PC.

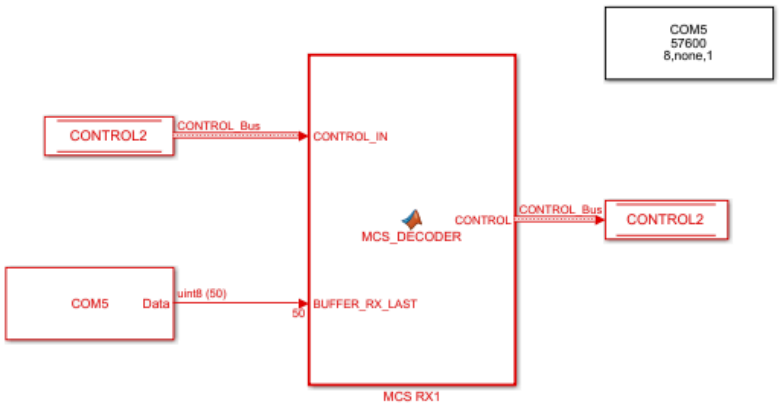

Figura 4.20. Decodificador empleado en el ordenador para la recepción de los datos de la Raspberry Pi.

El paso final de la verificación es seleccionar los valores que nos interesan del bus *Control2*: *MCS_POS_XY* (donde se encuentran los valores de las coordenadas en los ejes X e Y del centro de masas del vehículo) y *MS_YAW_ANG* (donde se encuentra el valor del ángulo de guiñada medido por las cámaras). Al mostrar estos valores en unos *displays*, se demuestra que efectivamente coinciden con los valores iniciales que se envían a la Raspberry Pi, Figura 4.21.

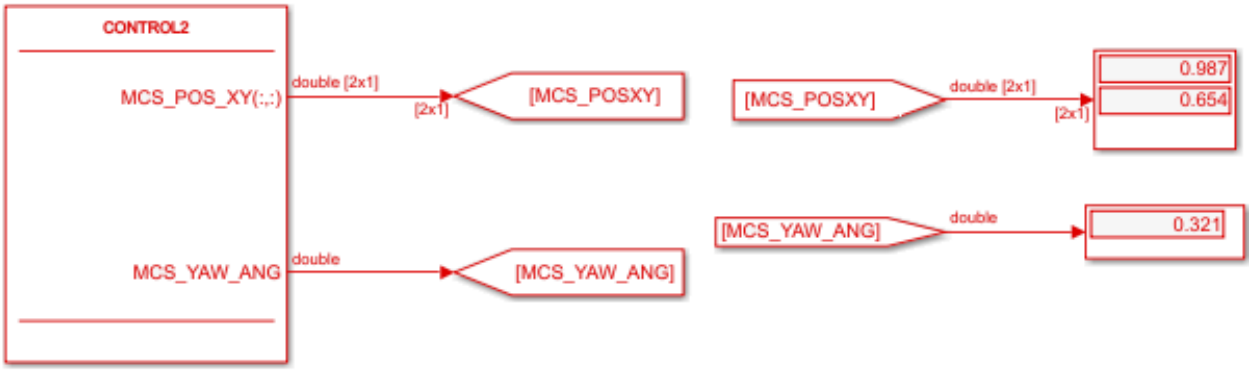

Figura 4.21. Visualización de los valores recibidos de la Raspberry Pi.

Se observa por lo tanto que los valores que envía la Raspberry Pi (coincidentes con los que recibe directamente del ordenador) son los mismos que se envían inicialmente: 0.987, 0.654 y 0.321. De esta forma se cierra el lazo de transmisión y se comprueba el correcto funcionamiento de todos los elementos que componen el sistema de transmisión Bluetooth.

Gracias a esta prueba final, se ha demostrado que el sistema de comunicación Bluetooth de la Raspberry Pi es un sistema robusto y fiable, que funciona correctamente.

4.4 COMUNICACIÓN WIFI ENTRE LA RASPBERRY PI Y EL ORDENADOR DE MONITORIZACIÓN

La comunicación entre el vehículo y el ordenador de trabajo se realiza mediante una conexión Wifi a través de un *router* inalámbrico al que deben estar ambos conectados. En el ordenador de trabajo se encuentran los ficheros de Simulink que contienen la lógica del vehículo; es también desde donde se modifican los parámetros y se monitorizan y registran las variables de interés. Por lo tanto, la carga del *firmware* del vehículo se hace también mediante Wifi.

Capítulo 5. SISTEMA DE CONTROL

El objetivo final de este capítulo es el diseño del sistema de navegación del vehículo que le permita circular por el entorno previamente delimitado indicándole únicamente el punto de origen y el destino. Para poder alcanzar el objetivo, ha sido necesario llevar a cabo varias tareas previas que permitieran diseñar los controles del vehículo y que se describen a lo largo de este Capítulo.

En primer lugar, se realizaron una serie de ensayos con el vehículo en lazo abierto, que permitieron estimar los parámetros del modelo del vehículo y determinar de esta forma las funciones de transferencia de las plantas, necesarias para para llevar a cabo un primer diseño de los tres controles: velocidad de avance, velocidad de giro y ángulo de giro.

Una vez realizada la identificación de los modelos se diseñaron los controles de velocidad de avance y giro y el control del ángulo de giro necesario para la navegación. Estos controles sirvieron de base, para poder realizar de nuevo los ensayos, esta vez en lazo cerrado, y poder obtener así un modelo más preciso.

Tras rediseñar los tres controles, se analizaron y estudiaron a través de simulaciones para comprobar que estos eran similares a lo esperado. Una vez hechas las comprobaciones, se realizaron una vez más los ensayos con los nuevos controles y se llevó a cabo una comparación de estos con sus respectivas simulaciones.

Con los tres controles ya diseñados, se procede a diseñar el sistema de navegación del vehículo, que se explica de forma más detallada en el Capítulo 6.

Con los ensayos que se realizan en lazo abierto inicialmente, se lleva a cabo la identificación del modelo. Para esa primera identificación se diseñan los tres controles y se ensayan esta vez, en lazo cerrado.

Estos controles sirven de base para el diseño de los controles definitivos. Con los ensayos en lazo cerrado se realiza de nuevo la identificación del modelo, que resulta ser más precisa que la obtenida inicialmente. Se lleva a cabo el diseño de los controles definitivos para esta segunda identificación, esta vez obtenida a través de unos ensayos realizados en lazo cerrado.

5.1 MODELADO Y OBTENCIÓN DE PARÁMETROS

Para poder llevar a cabo el diseño del sistema de navegación del vehículo, primero se deben identificar los parámetros del modelo. Para ello es necesario estimarlos mediante ensayos en lazo abierto y determinar las funciones de transferencia necesarias.

De esta forma se obtienen las funciones de transferencia entre la tensión común u_c aplicada a los motores y la velocidad de avance v del vehículo, entre la tensión diferencial u_d de los motores y la velocidad de rotación del vehículo w y por último, la función de transferencia entre la tensión diferencial u_d y el ángulo de guiñada del vehículo.

Las tres funciones de transferencia siguen la misma estructura:

$$P(s) = \frac{K_m}{(1 + T_m s) \cdot (1 + T_f s)}$$

En el proceso de identificación se estiman los parámetros de los motores:

- Par de fricción estático
- Resistencia eléctrica
- Constante de par contraelectromotriz
- Constante de fuerza contraelectromotriz

También se lleva a cabo la estimación de los parámetros mecánicos del vehículo:

- Momento de inercia
- Fricciones viscosas

Además, se estima la caída de tensión del convertidor PWM que alimenta cada motor.

Para poder realizar la identificación de los modelos, hay que hacer tres ensayos: velocidad de avance, velocidad de giro y ángulo de guiñada; y obtener a partir del método de optimización de Gauss-Newton (ajuste por mínimos cuadrados) los parámetros nombrados anteriormente.

El problema que presenta la identificación es que como no se dispone de un modelo preciso y correcto en el cuál basarse para el diseño del control, se vuelve necesaria la realización de ensayos para poder llevar a cabo un análisis de las variables del sistema.

5.1.1 IDENTIFICACIÓN DEL MODELO DE VELOCIDAD DE AVANCE

Para llevar a cabo la identificación del modelo, es necesario realizar un ensayo a partir del cual se estimen los parámetros del modelo detallado que afectan exclusivamente al avance del vehículo mediante un ajuste por mínimos cuadrados. En este caso, la entrada es el mando (tensión común de los motores, u_c) y la salida es la velocidad de avance medida, v .

Los parámetros estimados son:

- La caída de tensión común en el convertidor PWM que alimenta cada motor
- El par de fricción estático
- La resistencia eléctrica
- Constante de par contraelectromotriz de los motores

- Constante de fuerza contraelectromotriz de los motores

Una vez realizado el ensayo necesario, se llevan a cabo los siguientes pasos para poder hacer la identificación:

Preparación de los datos

El ensayo realizado queda registrado en Matlab. Una vez finalizado, es necesario seleccionar la sección del ensayo que mejor represente el comportamiento del vehículo, puesto que esta será la empleada durante la optimización, siendo preferible evitar las partes que contengan anomalías o irregularidades.

Optimización

La optimización se realiza mediante Matlab, y se lleva a cabo dentro de un bucle *while* que simula la salida obtenida para combinaciones diferentes de parámetros. Para ello se emplea un diagrama de Simulink diseñado de forma específica para llevar a cabo la caracterización del modelo.

El algoritmo se ejecuta en un bucle *while* mientras la variación de la función objetivo, en porcentaje, sea mayor que la tolerancia definida previamente; o cuando el máximo incremento de los parámetros, también en porcentaje, sea mayor que su tolerancia. Cuando alguna de estas dos condiciones no se cumpla, se detiene el programa. Mientras tanto el programa va representando en un mismo gráfico la respuesta en frecuencia real (obtenida en el ensayo), la obtenida a través del modelo y la diferencia entre ambas, Figura 5.1.

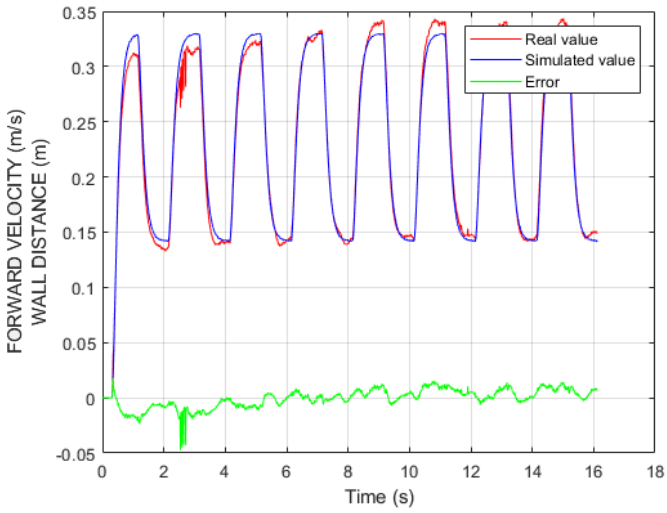

Figura 5.1. Respuesta en frecuencia real (obtenida en el ensayo), la obtenida a través del modelo y la diferencia entre ambas

Una vez detenido el programa, se obtienen los parámetros que ofrecen un valor mejor de la función de coste, se muestran por pantalla y se prepara para realizar una nueva iteración, cuyo punto de partida serán los últimos parámetros obtenidos.

Los valores que se obtienen mediante el algoritmo de iteración, no son los valores reales de los parámetros, sino que son los coeficientes que multiplican a sus valores iniciales.

5.1.2 IDENTIFICACIÓN DEL MODELO DE VELOCIDAD DE GIRO

Una vez hecha la identificación del modelo de velocidad de avance y a partir de los parámetros estimados del modelo, se lleva a cabo la identificación del modelo de velocidad de giro.

Para llevar a cabo la identificación del modelo, es necesario realizar un ensayo a partir del cual se estimen los parámetros del modelo detallado que afectan exclusivamente al giro del vehículo, de nuevo mediante el método de Optimización de Gauss-Newton (ajuste por mínimos cuadrados). En este caso, la entrada es el mando (tensión diferencial de los motores, u_d) y la salida es la velocidad de giro medida, w .

Los parámetros estimados son:

- La caída de tensión diferencial en el convertidor PWM que alimenta cada motor
- El momento de inercia
- La fricción viscosa del vehículo

Una vez realizado el ensayo y registrado los datos, se lleva a cabo el mismo proceso de estimación de los parámetros que se describe en el Apartado 5.1.1 y que se divide en dos partes.

La estimación de los parámetros es similar a excepción de los modelos usados en ambos casos. Aunque ambos modelos no lineales están parametrizados y los coeficientes de ambos se actualizan en cada iteración del algoritmo de Gauss-Newton; cada modelo tiene como salida una variable distinta.

Tras llevar a cabo la identificación del modelo de giro y una vez finalizado el programa se obtienen las tres funciones de transferencia necesarias para diseñar los controles de velocidad de avance, velocidad de giro y ángulo de guiñada.

5.2 DISEÑO DE LOS CONTROLES

El diseño de los controles del vehículo se realiza a través de Matlab, mediante un fichero que permite al usuario acceder a una interfaz gráfica de diseño interactivo de controles PID mediante respuesta en frecuencia, Figura 5.2. A través de esta herramienta, se pueden realizar diseños para distintas especificaciones y comprobar la respuesta en frecuencia del sistema para cada combinación de valores, pudiendo escoger la que se considere más adecuada.

Esta interfaz gráfica permite escoger el tipo de control a diseñar: P, PD, PI y PID, entre otras cosas. Además, en función del tipo de control seleccionado, te permite modificar las especificaciones y decidir si diseñar por margen de fase o por margen de ganancia.

Para llevar a cabo el diseño, es necesario seleccionar la planta del sistema, que en el caso de este proyecto se trata de las tres funciones de transferencia para la velocidad de avance, velocidad de giro y ángulo de guiñada, que se han obtenido en la identificación del modelo a través de los ensayos realizados en lazo abierto.

Inicialmente se realizan los diseños de los controles mediante la identificación para los ensayos en lazo abierto. Una vez diseñados los tres controles, se llevan a cabo los ensayos, esta vez en lazo cerrado, para realizar una nueva identificación. Finalmente se diseñan los que serán los controles definitivos para la identificación con los ensayos en lazo cerrado.

El motivo principal de repetir el proceso es que el ensayo realizado en lazo cerrado tiene más información en el rango de frecuencias donde se va a diseñar el control. Este hecho se traduce en una considerable mejora de la precisión del modelo en ese rango de frecuencias, que es el que resulta realmente interesante en este proyecto.

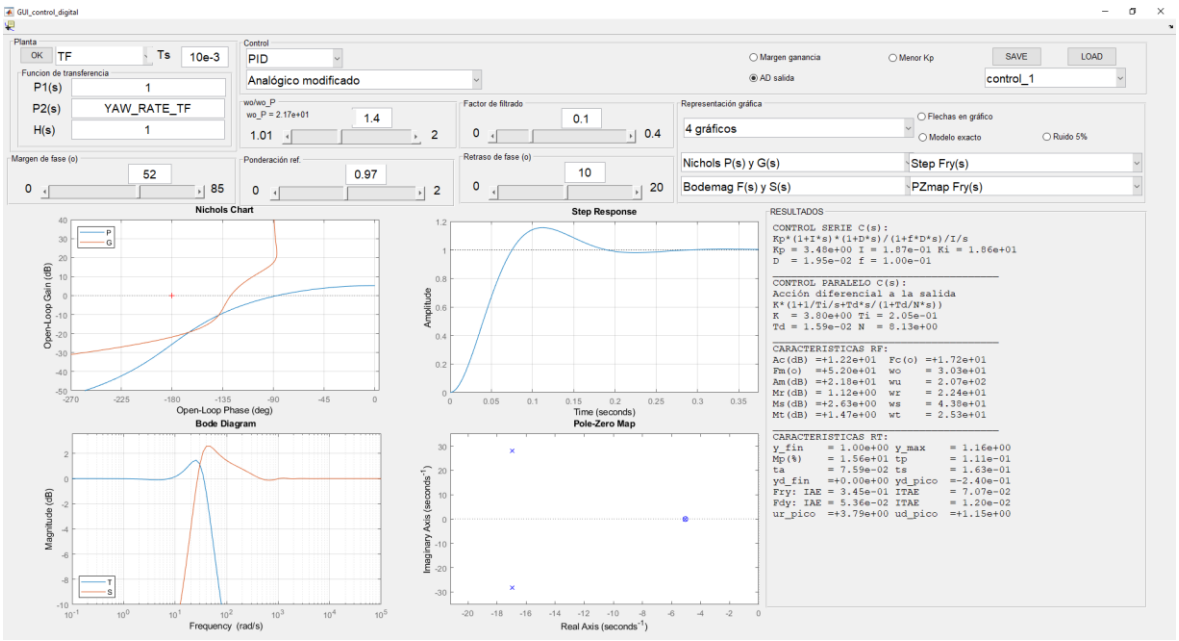

Figura 5.2. Interfaz gráfica de diseño de los controles.

5.3 MODELADO Y OBTENCIÓN DE PARÁMETROS EN LAZO CERRADO

Una vez realizado el primer diseño de los controles, se ha decidido repetir la identificación de los parámetros del modelo, pero esta vez para los ensayos realizados en lazo cerrado.

Para estimar los parámetros y obtener las funciones de transferencia de la velocidad de avance, velocidad de giro y ángulo de guiñada, se debe repetir el proceso explicado en el Apartado 5.1.1 que se divide en dos partes: preparación de los datos y optimización, con la excepción de que esta vez los ensayos se realizarán en lazo cerrado.

Tras registrar los datos y obtener los valores estimados de los parámetros y las nuevas funciones de transferencia, se procede a realizar el nuevo diseño de los controles, esta vez para los parámetros del modelo obtenidos en la identificación mediante los ensayos en lazo cerrado.

5.4 DISEÑO DE LOS CONTROLES DEFINITIVOS

Para el diseño de los controles se emplea de nuevo la interfaz de diseño descrita en la Sección 5.2. El objetivo es escoger los parámetros adecuados para que se cumplan las especificaciones de estabilidad y precisión deseados para la función de transferencia.

En esta sección se van a mostrar los tres diseños realizados mediante la interfaz gráfica para los tres controles: velocidad de avance, velocidad de giro y ángulo de guiñada; y se realizará una breve explicación sobre las especificaciones escogidas en cada caso y cómo afectan a la respuesta en frecuencia del sistema.

A continuación, se mostrarán los resultados obtenidos al simular los tres controles y se compararán con los esperados según la interfaz de diseño, realizando las modificaciones necesarias en caso de que no coincidan. En las gráficas que se muestran, se representan la referencia, la respuesta real obtenida y la respuesta filtrada.

Finalmente se realizará un ensayo para cada control con las especificaciones escogidas durante el diseño de los tres controles PID. En cada apartado de diseño de los controles se explican las referencias que se aplican en cada ensayo.

Una vez realizados los ensayos, estos se comparan con las simulaciones para verificar que el control se ha diseñado de forma correcta y la respuesta en frecuencia obtenida es similar a la de la simulación.

5.4.1 CONTROL DE VELOCIDAD DE AVANCE

Diseño del control

Para el diseño del control de velocidad de avance a través de la interfaz gráfica, se emplea la función de transferencia que relaciona la tensión común u_c aplicada a los motores y la velocidad de avance del vehículo v, *FORWARD_VEL_TF*.

También es necesario indicar el periodo de muestreo, que en el caso de este proyecto será en todo momento de 10 ms. Para el diseño de todos los controles se ha seleccionado el tipo de control PID.

En cuanto a las especificaciones escogidas para este primer control, Figura 5.3, se ha decidido diseñar por margen de fase ($\emptyset_m$) puesto que este es más restrictivo que el margen de ganancia (A_m) y más fiable [31]. Por otra parte, los valores recomendados son entre 45° y 60°, aunque por lo general cuanto mayor sea el margen de fase, mayor será la estabilidad del sistema.

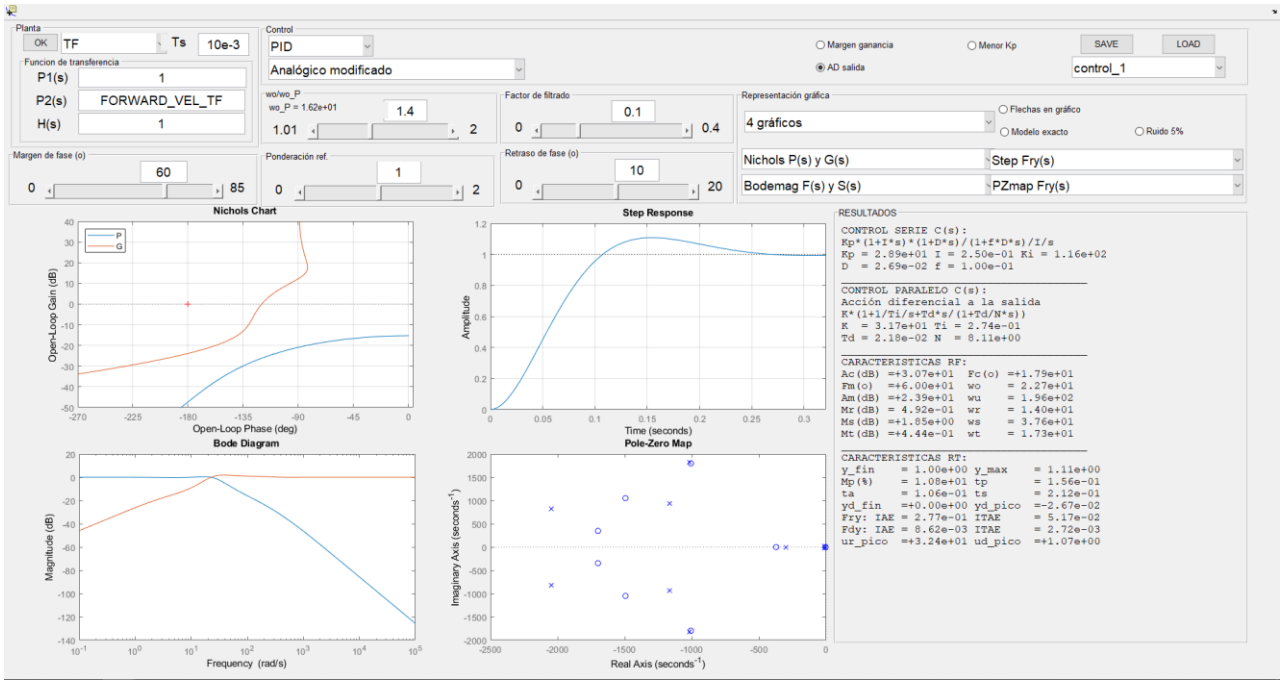

Figura 5.3. Primer control de velocidad de avance diseñado.

Para diseñar el control PID se ha escogido un margen de fase de $\emptyset_m = 60°$, una relación de velocidades de $w_o/w_{o_P} = 1.4$, factor de filtrado $f = 0.1$, ponderación a la referencia $b = 1$ y un retraso de fase por acción integral de $\varphi_I = -10°$. Por otra parte, la acción diferencial se aplica a la salida.

Para este diseño del control, se obtienen las características de respuesta temporal y en frecuencia mostradas en la Figura 5.3. El análisis de estos parámetros permite conocer la respuesta de la señal en régimen permanente en términos de velocidad, amortiguamiento y también la estabilidad en lazo cerrado.

La respuesta a un escalón unitario en referencia muestra un sobrepaso moderado y una amortiguación adecuada. El sistema cuenta con un sobrepaso (M_p) del 10.8% y un tiempo de establecimiento (t_s) de 0.212s.

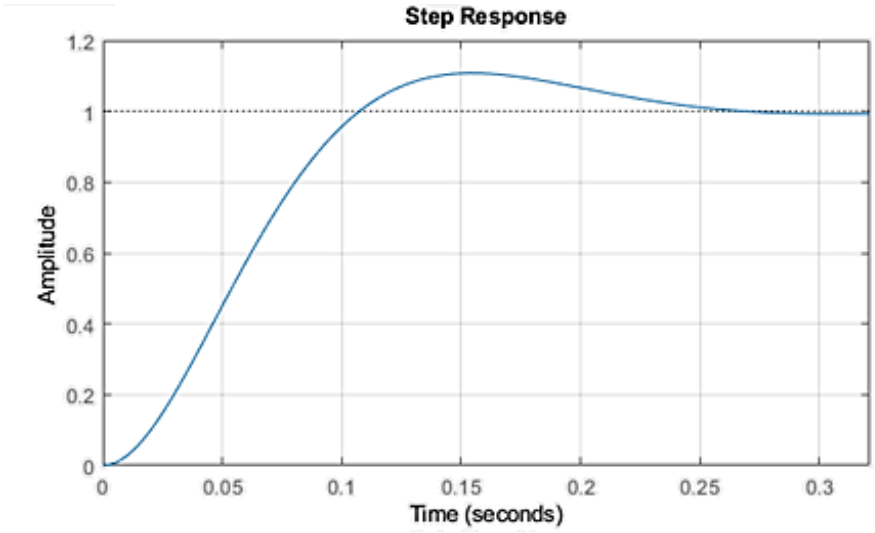

Figura 5.4. Respuesta a un escalón unitario en referencia del control de velocidad de avance.

Una vez terminado el diseño de control, se realiza la simulación del mismo y la respuesta se compara con la respuesta esperada según la interfaz de diseño, Figura 5.4. La respuesta que se obtiene en la simulación para las especificaciones escogidas no resulta similar a la respuesta que se espera, contando con un sobrepaso mucho mayor e igual a $M_p = 32.7\%$, por lo que se hace necesario repetir el diseño del control esta vez buscando obtener un sobrepaso mucho menor.

En el segundo diseño que se realiza, Figura 5.5, las especificaciones escogidas son: un margen de fase de $\emptyset_m = 70°$, una relación de velocidades de $w_o/w_{o_P} = 1.4$, factor de filtrado $f = 0.1$, ponderación a la referencia $b = 1.025$ y un retraso de fase por acción integral de $\varphi_I = -10°$. La acción diferencial se sigue aplicando a la salida. Se decide aumentar el margen de fase para que disminuya el sobrepaso, pero es necesario aumentar la ponderación a la referencia para que la respuesta sea adecuada.

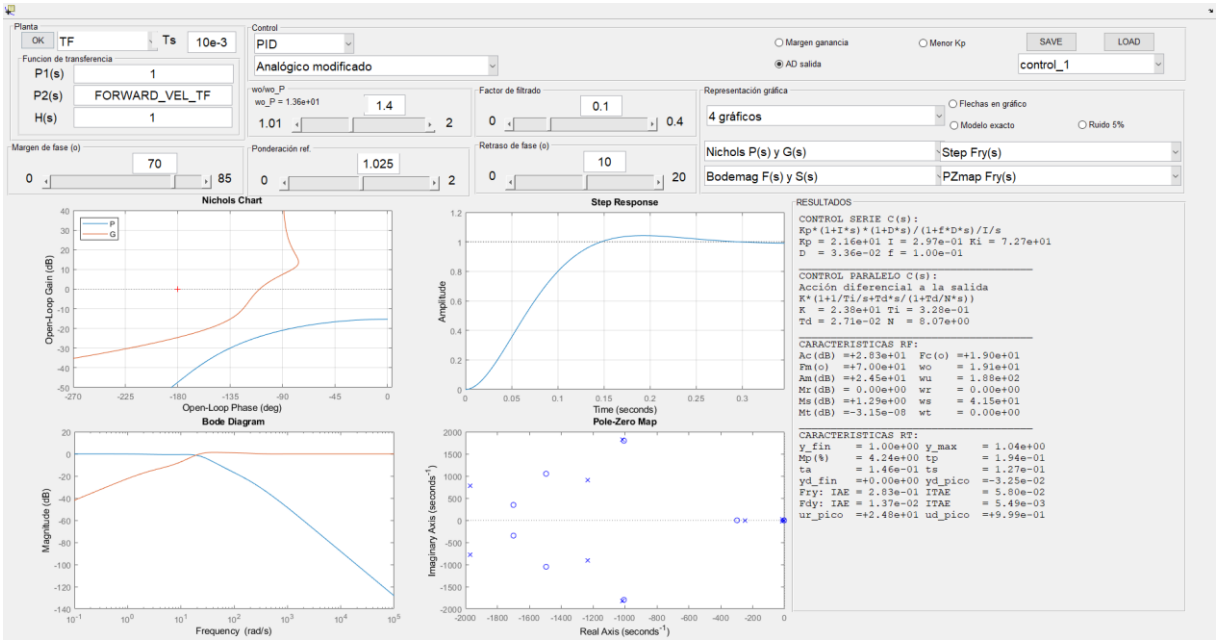

Figura 5.5. Segundo control de velocidad de avance diseñado.

En este caso la respuesta a un escalón unitario en referencia que se obtiene muestra un sobrepaso moderado y una amortiguación adecuada, Figura 5.5. El sistema cuenta con un sobrepaso (M_p) menor que en el diseño anterior y de valor igual al 4.24% y con un tiempo de establecimiento (t_s) de 0.127s también menor que en el caso anterior.

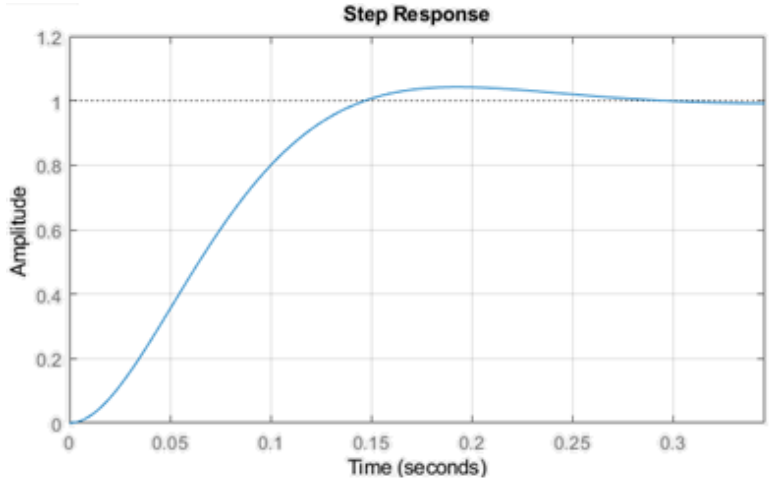

Figura 5.6. Respuesta a un escalón unitario en referencia del control definitivo de velocidad de avance.

El diseño definitivo del control de avance, Figura 5.6, resulta similar al obtenido en la simulación y cuenta con una respuesta de sobrepaso y amortiguación adecuadas, además de una rapidez considerable.

Simulación

Tal y como se comenta en el apartado de diseño, en el primer intento la simulación no se corresponde con lo esperado según la interfaz de diseño. El sobrepaso obtenido en la simulación, Figura 5.7, es mucho mayor al esperado (cuyo valor es de 10.8%), siendo igual a:

$$M_p = \frac{y_{máx} - y_\infty}{y_\infty - y(0)} = \frac{0.3564 - 0.3}{0.3 - 0.1} = 28.2\%$$

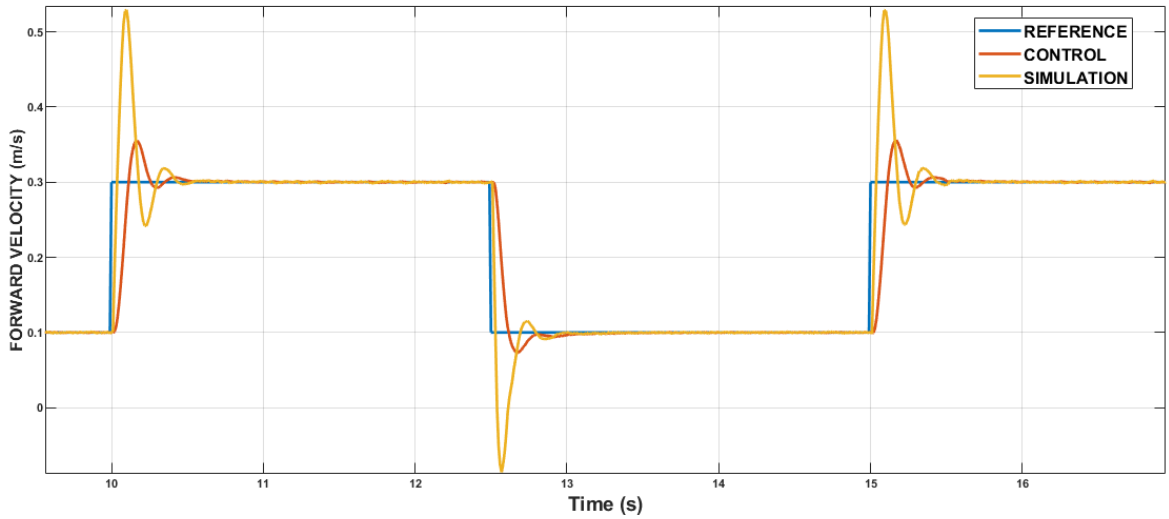

Figura 5.7. Simulación del primer control de velocidad de avance diseñado.

Las medidas se corresponden con la señal del *control* de color rojo, Figura 5.8.

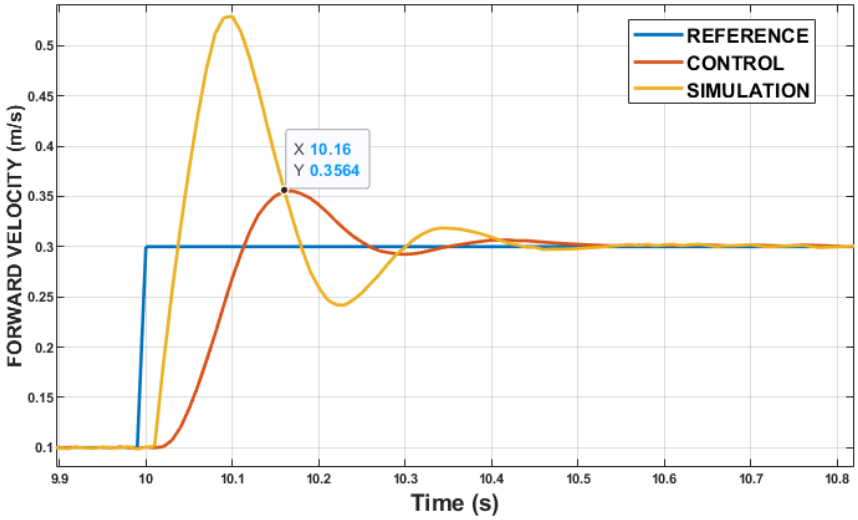

Figura 5.8. Sobrepaso del primer diseño del control de velocidad de avance.

Debido a la poca similitud entre la simulación y el diseño que se esperaba, se decide realizar el nuevo diseño que se comenta en el apartado de diseño y que cuenta con un sobrepaso mucho menor de valor $M_p = 4.24\%$. La simulación de este nuevo control, Figura 5.9, obtiene una respuesta mucho más parecida a la esperada cuyo sobrepaso tiene un valor de:

$$M_p = \frac{y_{máx} - y_\infty}{y_\infty - y(0)} = \frac{0.3263 - 0.3}{0.3 - 0.1} = 13.15\%$$

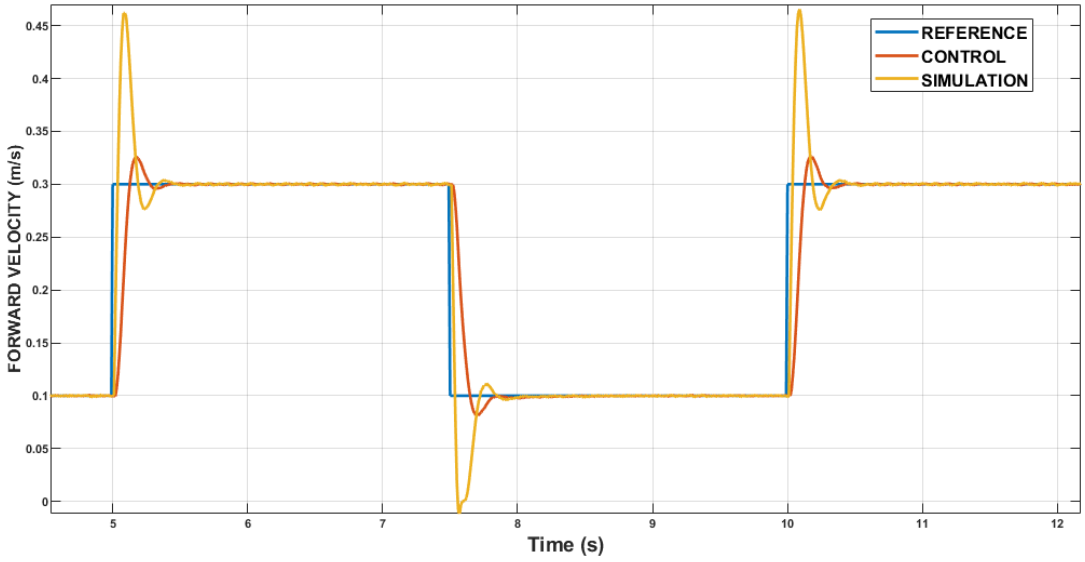

Figura 5.9. Simulación del segundo control de velocidad de avance diseñado.

Las medidas se corresponden con la señal del *control* de color rojo, Figura 5.10.

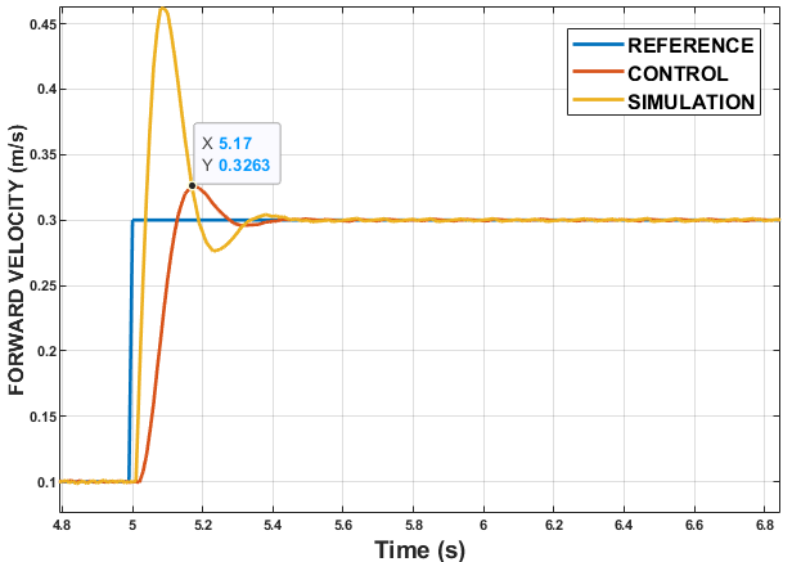

Figura 5.10. Sobrepaso del segundo diseño del control de velocidad de avance.

Ensayo

Una vez realizada la comprobación de que la respuesta obtenida en la simulación es similar a la esperada, se procede a realizar un ensayo para las especificaciones escogidas en el segundo diseño del control de velocidad de avance, que será el definitivo.

Las referencias que se aplican en el ensayo, Figura 5.11, son una onda cuadrada en la referencia de velocidad de avance entre 0.1 m/s y 0.3 m/s y período igual a 5 s y una referencia nula para la velocidad de giro y ángulo de guiñada.

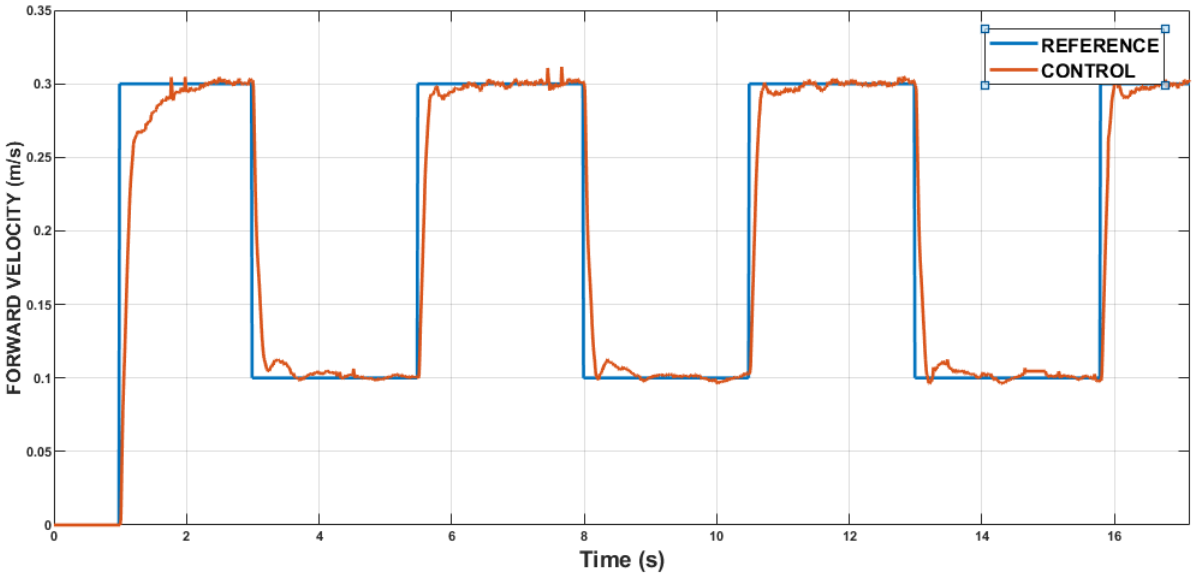

Figura 5.11. Ensayo del control de velocidad de avance.

Se puede observar que la respuesta sigue la referencia de forma adecuada a pesar de las pequeñas irregularidades que se muestran y que podrían ser debidas a la sensibilidad al ruido que presenta el control a causa de la acción derivativa.

Se ha estudiado la posibilidad de emplear un control PI para que la acción derivativa no afecte a la sensibilidad frente al ruido, pero el control se vuelve demasiado lento por lo que se decide que es preferible mantener la condición de velocidad a pesar de que pueda afectar a la sensibilidad frente al ruido.

Comparación del ensayo y la simulación

Una vez realizado el ensayo se procede a realizar una comparación de la respuesta obtenida durante el ensayo con la obtenida en la simulación, Figura 5.12.

- En color *rojo* se muestra la simulación.
- En color *amarillo* se muestra el ensayo.

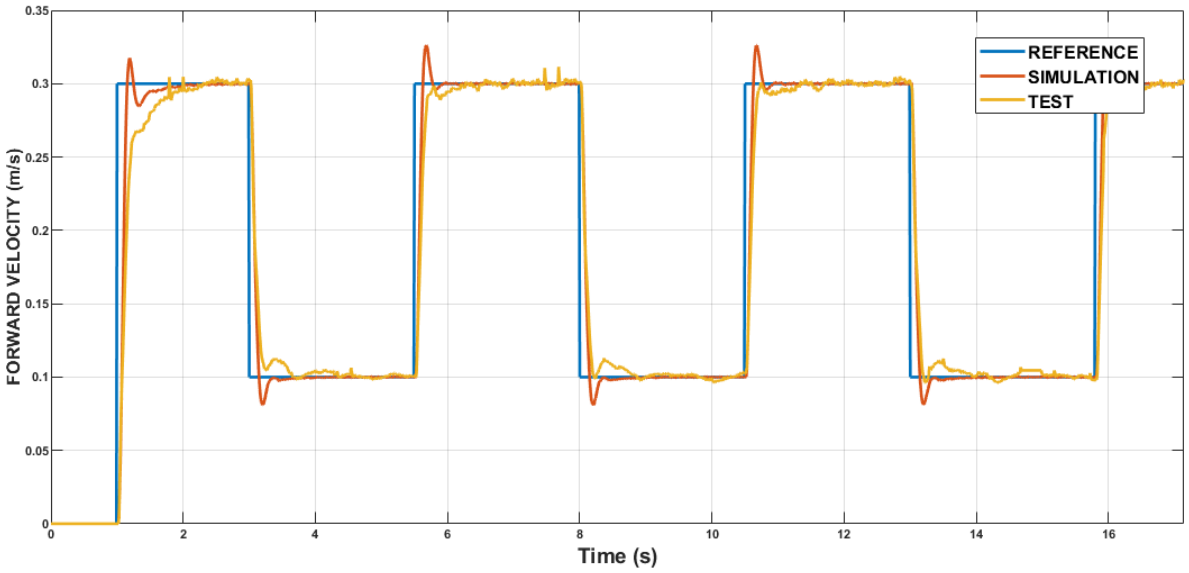

Figura 5.12. Comparación del ensayo y la simulación del control de velocidad de avance.

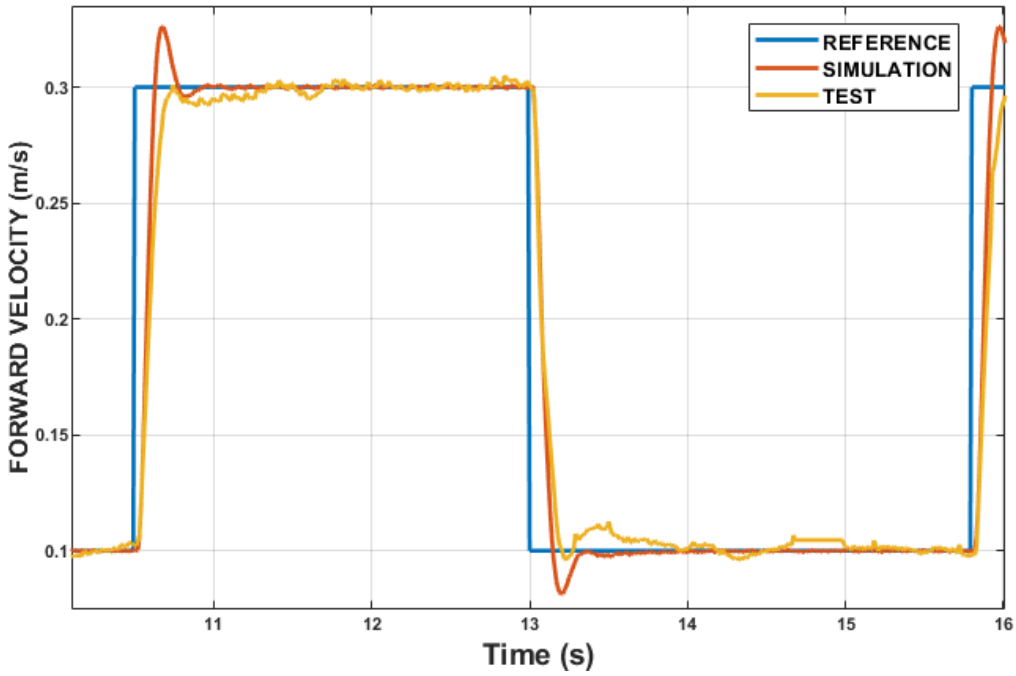

Figura 5.13. Comparación pendiente de la simulación y el ensayo.

La respuesta obtenida en el ensayo es muy similar a la simulada, especialmente durante la parte de subida y bajada, donde se muestra que ambos tienen la misma pendiente, Figura 5.13. Sin embargo, el control cuenta con un sobrepaso demasiado pequeño, por lo que se decide cambiar el valor de la ponderación a la referencia de la simulación, únicamente para la comparación, para poder comprobar si es este parámetro el que afecta a la respuesta.

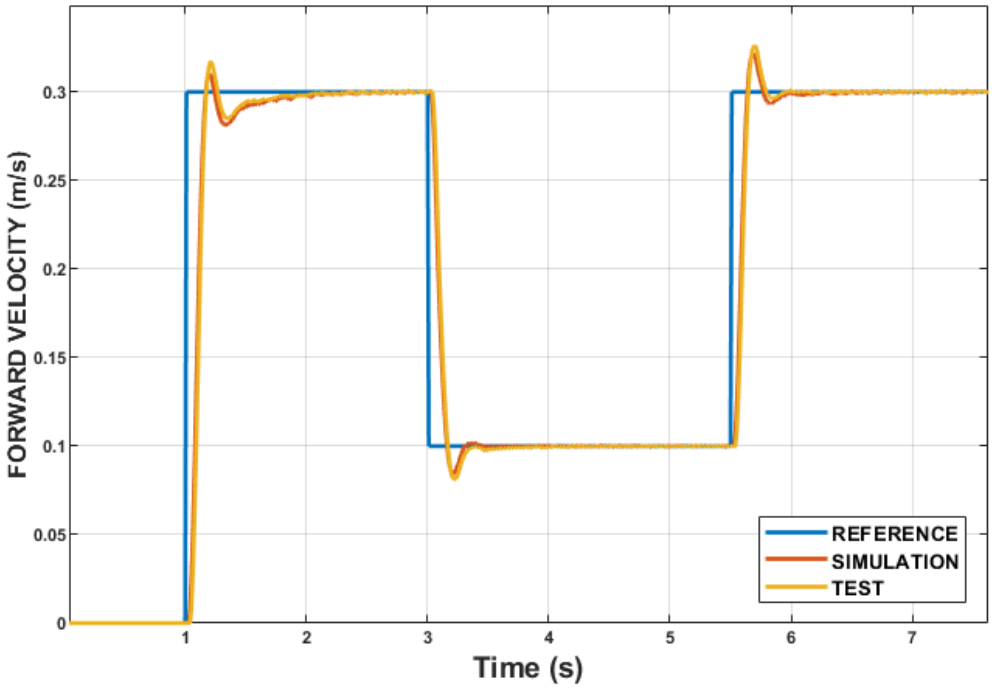

Figura 5.14. Comparación entre el ensayo y la simulación cambiando el valor de la ponderación a la referencia.

Al modificar ligeramente el valor de la ponderación a la referencia se observa que las respuestas se vuelven casi idénticas, Figura 5.14. Sin embargo, el control se mantiene con los parámetros escogidos en el segundo diseño y que se comentan en el apartado de diseño.

5.4.2 CONTROL DE VELOCIDAD DE GIRO

Diseño del control

Para el diseño del control de velocidad de giro a través de la interfaz gráfica, se emplea la función de transferencia que relaciona la tensión diferencial u_d de los motores y la velocidad de rotación del vehículo w, *YAW_RATE_TF*.

También es necesario indicar el periodo de muestreo, que será de 10 ms tal y como se ha comentado en el Apartado 5.4.1. Para el diseño de todos los controles se ha seleccionado el tipo de control PID.

Para diseñar el control PID, Figura 5.15, se ha escogido un margen de fase de $\emptyset_m = 60°$, una relación de velocidades de $w_o/w_{o_P} = 1.4$, factor de filtrado $f = 0.1$, ponderación a la referencia $b = 1$ y un retraso de fase por acción integral de $\varphi_I = -10°$. Por otra parte, la acción diferencial se aplica a la salida.

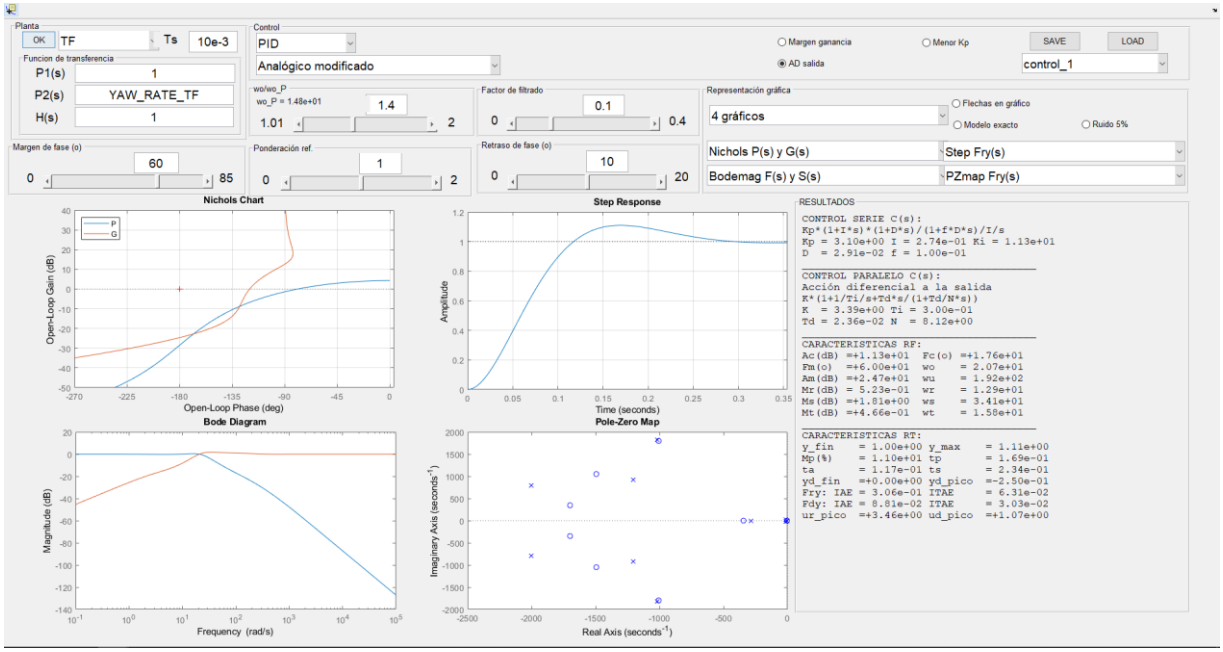

Figura 5.15. Primer control de velocidad de giro diseñado.

La respuesta a un escalón unitario en referencia muestra un sobrepaso moderado y una amortiguación adecuada, ambos similares a los obtenidos en el diseño del control de velocidad de avance, Figura 5.16. El sistema cuenta con un sobrepaso (M_p) del 11% y un tiempo de establecimiento (t_s) de 0.234s.

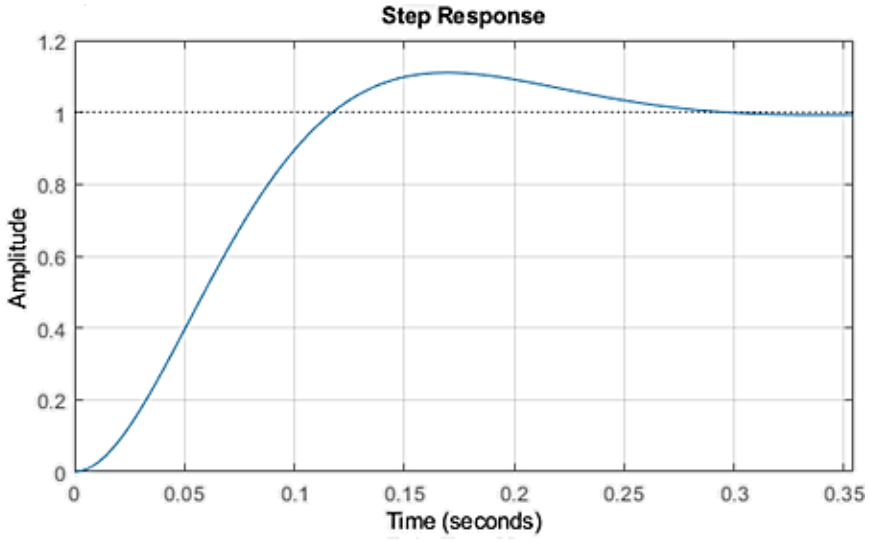

Figura 5.16. Respuesta a un escalón unitario en referencia del control de velocidad de giro.

Sin embargo, una vez terminado el diseño de control, se realiza la simulación del mismo y la respuesta se compara con la respuesta esperada según la interfaz de diseño. De la misma

forma que ocurre con el control de velocidad de avance, la respuesta que se obtiene en la simulación para las especificaciones escogidas no resulta similar a la respuesta que se espera, contando con un sobrepaso mucho mayor, por lo que de nuevo se hace necesario repetir el diseño del control esta vez buscando obtener un sobrepaso mucho menor.

En el segundo diseño que se realiza, las especificaciones escogidas son: un margen de fase de $\emptyset_m = 75°$, una relación de velocidades de $w_o/w_{oP} = 1.4$, factor de filtrado $f = 0.1$, ponderación a la referencia $b = 1.075$ y un retraso de fase por acción integral de $\varphi_I = -10°$. La acción diferencial se sigue aplicando a la salida. De la misma forma que con el control de velocidad de avance, se decide aumentar el margen de fase para que disminuya el sobrepaso, pero es necesario aumentar la ponderación a la referencia para que la respuesta sea adecuada.

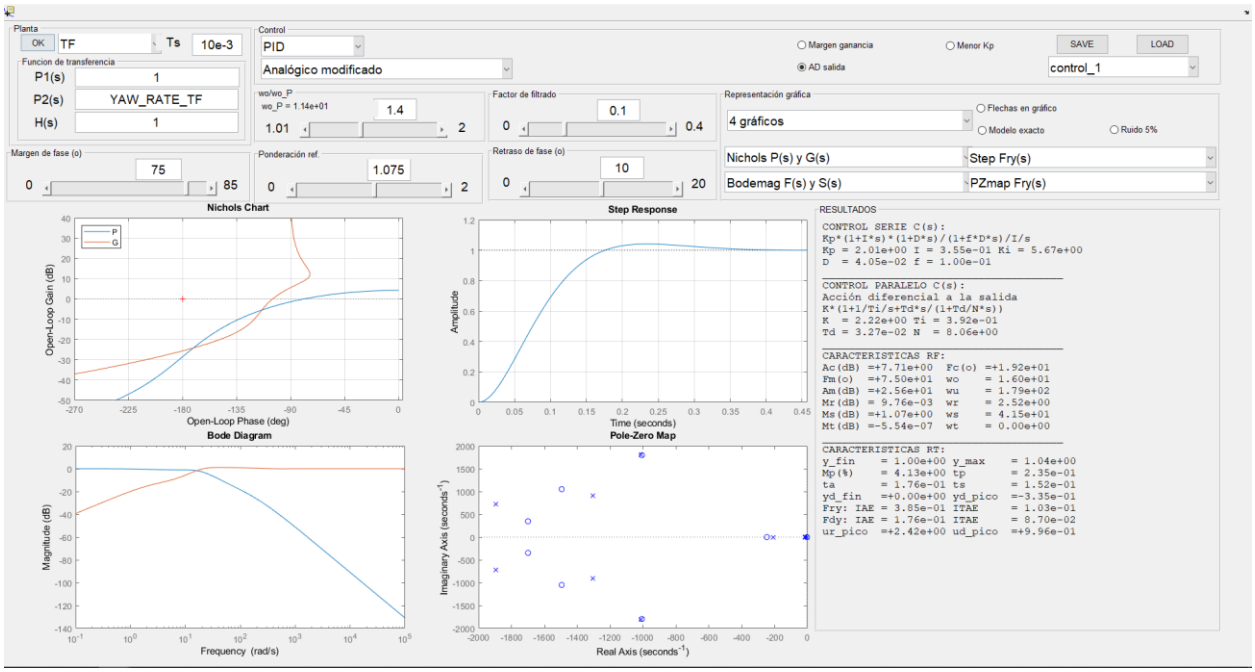

Figura 5.17. Segundo control de velocidad de giro diseñado.

Para este segundo diseño del control, la respuesta a un escalón unitario en referencia que se obtiene muestra un sobrepaso menor y una amortiguación igualmente adecuada, Figura 5.17. El sobrepaso (M_p) que se obtiene con este diseño tiene un valor igual al 4.13% y un tiempo de establecimiento (t_s) de 0.152s bastante menor que en el caso anterior.

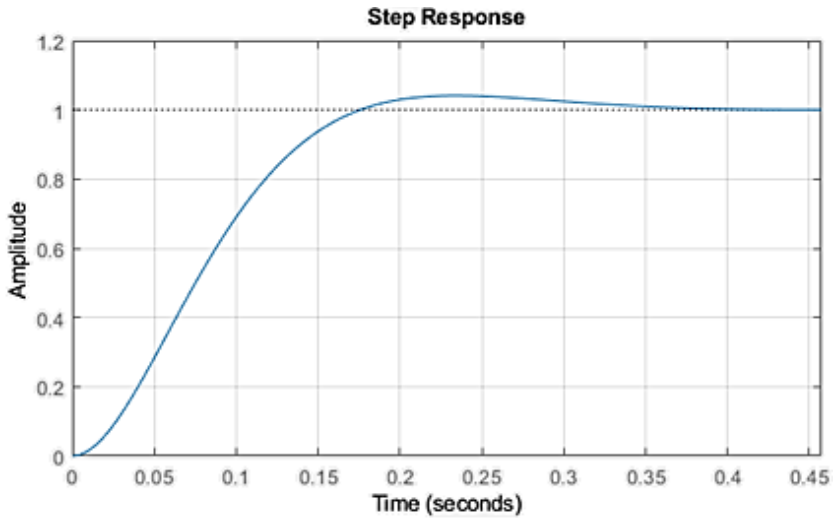

Figura 5.18. Respuesta a un escalón unitario en referencia del control definitivo de velocidad de giro.

Este es el diseño definitivo del control de velocidad de giro, Figura 5.18, cuyas especificaciones se emplearán durante el resto del proyecto. La respuesta que se obtiene en la simulación es similar a la esperada.

Simulación

De la misma forma que para el diseño del control de avance, una vez realizado el diseño es necesario compararlo con la respuesta obtenida en la simulación para comprobar que se trata de un diseño adecuado.

Tal y como se comenta en el apartado de diseño, en el primer intento la simulación no se corresponde con lo esperado según la interfaz de diseño. El sobrepaso obtenido en la simulación es mucho mayor al esperado, por lo que se diseña el segundo control que será el definitivo.

En la simulación de este nuevo control, Figura 5.19, se obtiene una respuesta mucho más parecida a la esperada ($M_p = 4.13\%$) cuyo sobrepaso tiene un valor de:

$$M_p = \frac{y_{máx} - y_\infty}{y_\infty - y(0)} = \frac{1.279 - 1.047}{1.047 - (-1.047)} = 11.08\%$$

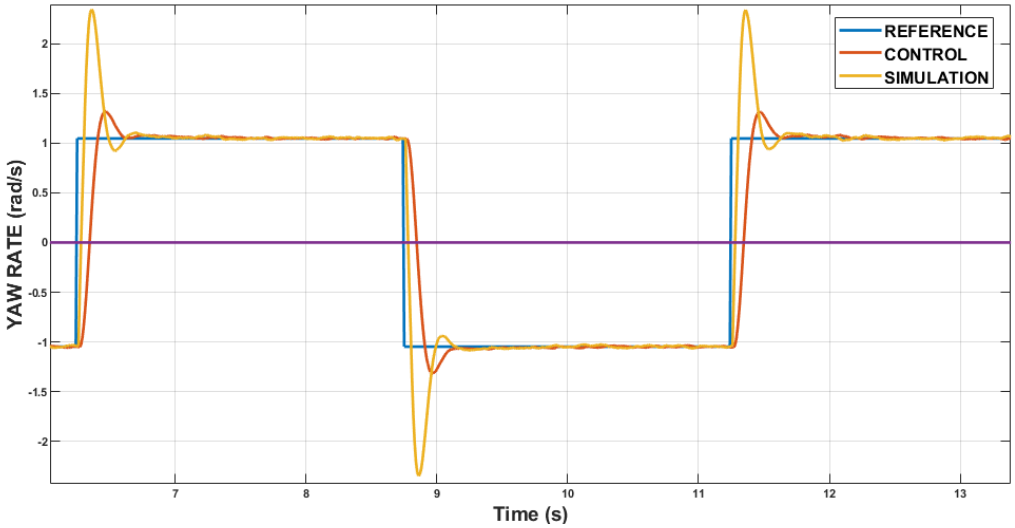

Figura 5.19. Simulación del segundo control de velocidad de giro diseñado.

Las medidas se corresponden con la señal del *control* de color rojo, Figura 5.20.

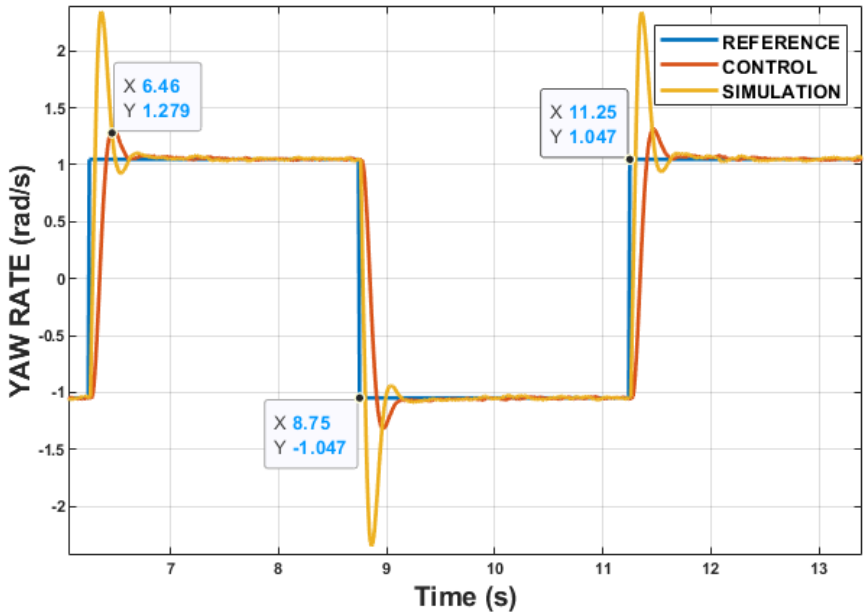

Figura 5.20. Sobrepaso del segundo diseño del control de velocidad de giro.

Ensayo

Se repite el proceso realizado para el diseño de velocidad de avance, una vez realizada la comprobación de que la respuesta obtenida en la simulación es similar a la esperada, se procede a realizar un ensayo con las especificaciones escogidas en el segundo diseño del control de velocidad de avance, que será el definitivo.

Las referencias que se aplican en el ensayo, Figura 5.21, son una onda cuadrada en la referencia de velocidad de giro entre 1.074 rad/s y -1.074rad/s y período igual a 5 s y una referencia nula para la velocidad de avance y ángulo de guiñada.

De nuevo se observa que la respuesta sigue la referencia de forma adecuada a pesar de las pequeñas irregularidades que se muestran y que podrían ser debidas a la sensibilidad al ruido que presenta el control a causa de la acción derivativa.

En este caso también se ha estudiado la posibilidad de emplear un control PI para que la acción derivativa no afecte a la sensibilidad frente al ruido, pero se llegan a las mismas conclusiones que para el control de velocidad de avance y se decide que es preferible mantener la condición de velocidad a pesar de que pueda afectar a la sensibilidad del control frente al ruido.

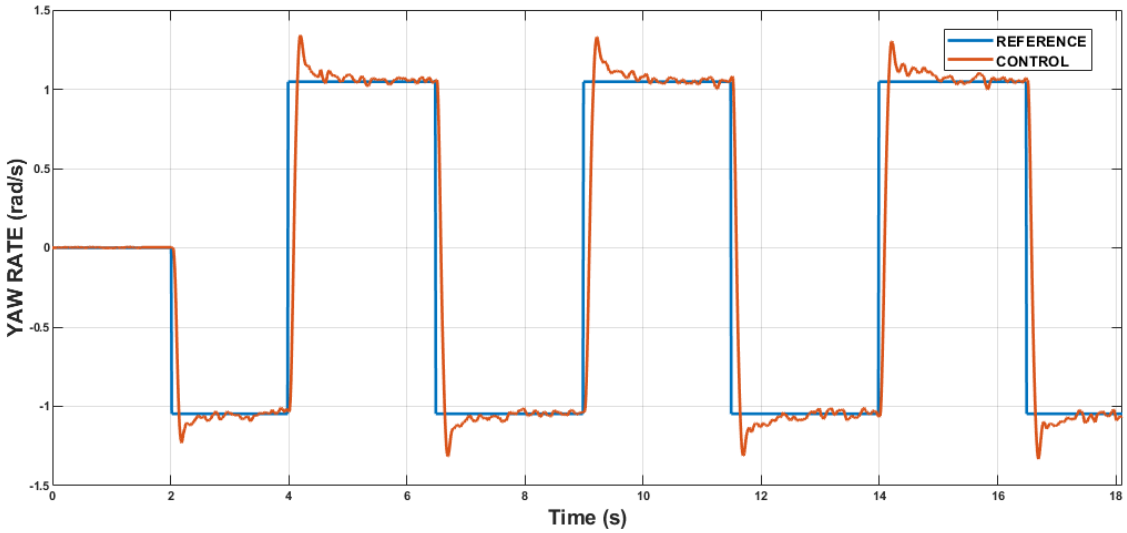

Figura 5.21. Ensayo del control de velocidad de giro.

Comparación del ensayo y la simulación

Una vez realizado el ensayo se procede a realizar una comparación de la respuesta obtenida durante el ensayo con la obtenida en la simulación, Figura 5.22.

- En color *rojo* se muestra la simulación.
- En color *amarillo* se muestra el ensayo.

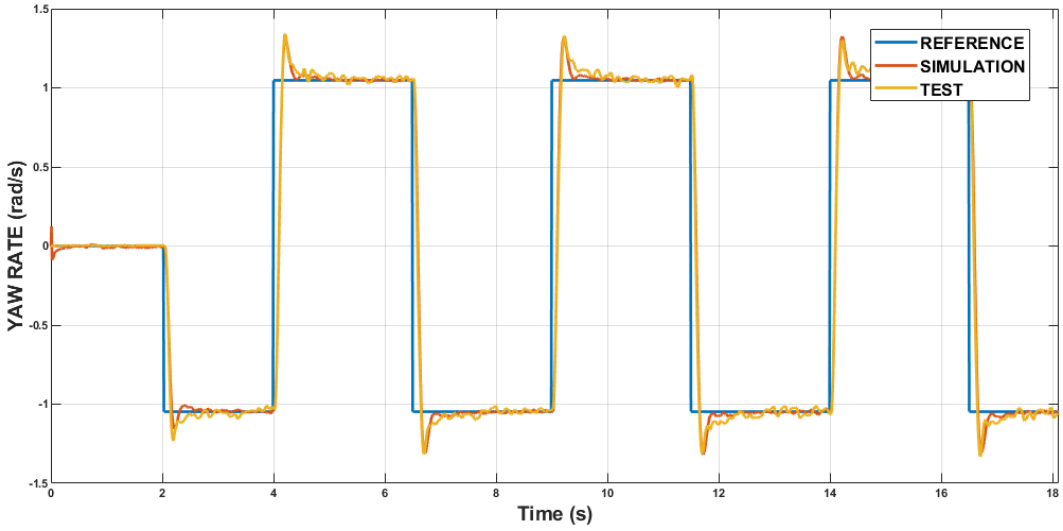

Figura 5.22. Comparación del ensayo y la simulación del control de velocidad de giro.

La respuesta obtenida en el ensayo es casi idéntica a la simulada, Figura 5.23, lo que significa que el diseño del control de velocidad de giro se ha realizado de forma correcta. Este diseño se empleará durante el resto del proyecto.

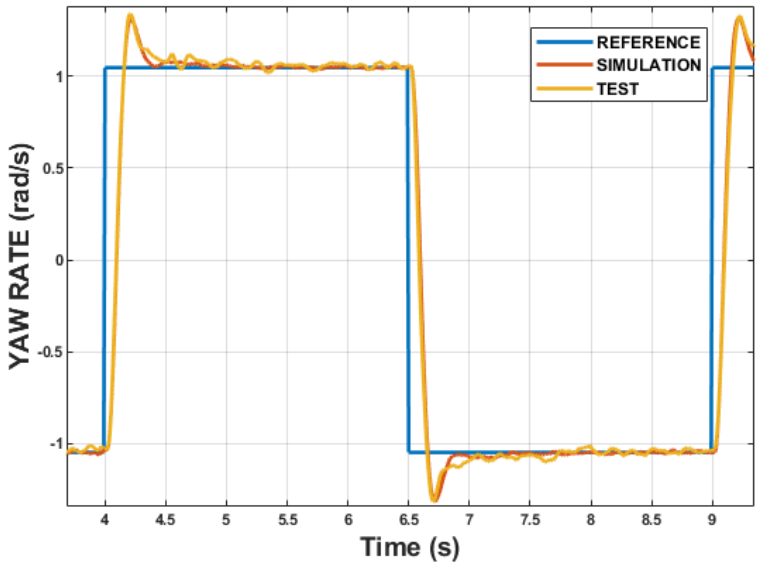

Figura 5.23. Comparación pendiente de la simulación y el ensayo para el control de velocidad de giro.

5.4.3 CONTROL DEL ÁNGULO DE GUIÑADA

Diseño del control

En esta sección se explica el proceso de diseño del último de los tres controles necesarios para llevar a cabo el diseño del sistema de navegación del vehículo. Como en el caso de los otros dos controles, se describe también el proceso que se ha llevado a cabo para comprobar que el control diseñado es el adecuado.

Para el diseño del control de ángulo de guiñada a través de la interfaz gráfica, se emplea la función de transferencia que relaciona la tensión diferencial u_d y el ángulo de guiñada del vehículo, *YAW_ANG_TF*.

También es necesario indicar el periodo de muestreo, que será de 10 ms tal y como se ha comentado en el Apartado 5.4.1. Para el diseño de todos los controles se ha seleccionado el tipo de control PID.

Para el diseño del control PID, Figura 5.24, se ha escogido un margen de fase de $\emptyset_m = 55°$, una relación de velocidades de $w_o/w_{o_P} = 1.4$, factor de filtrado $f = 0.1$, ponderación a la referencia $b = 0.75$ y un retraso de fase por acción integral de $\varphi_I = -10°$. Por otra parte, la acción diferencial se aplica a la salida, al igual que en los otros dos controles diseñados.

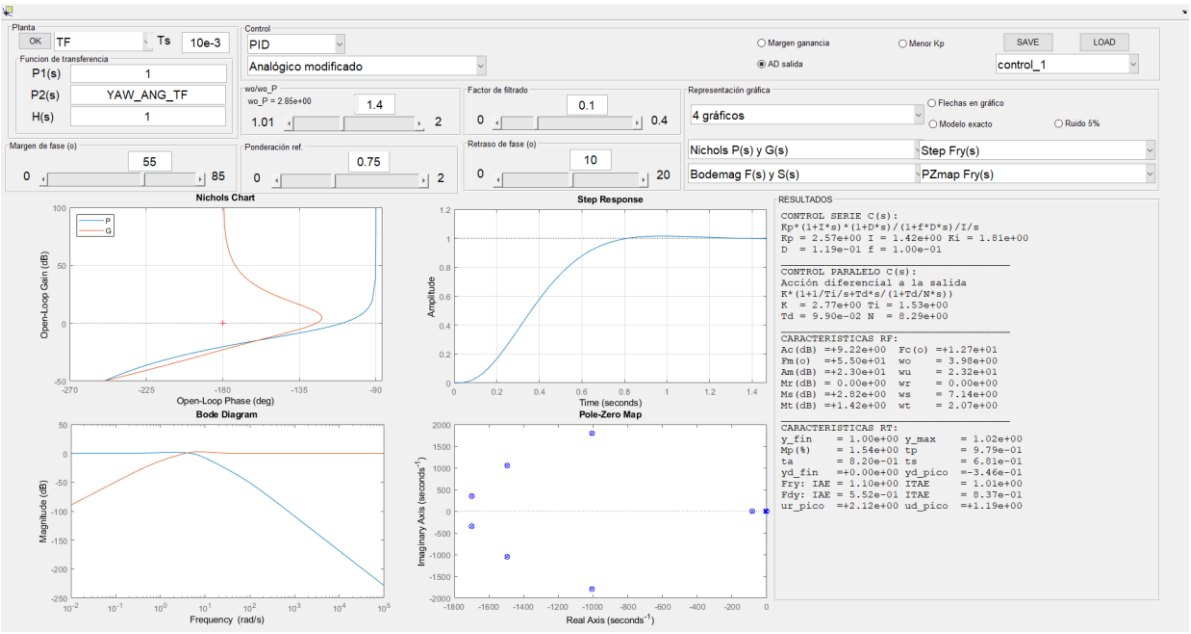

Figura 5.24. Diseño del control de ángulo de guiñada

La respuesta a un escalón unitario en referencia muestra un sobrepaso moderado y una amortiguación adecuada, ambos similares a los obtenidos en el diseño del control de velocidad

de avance, Figura 5.24. El sistema cuenta con un sobrepaso (M_p) del 1.54% y un tiempo de establecimiento (t_s) de 0.681s. Aunque este control es un poco más lento que los otros dos controles diseñados, tiene la rapidez necesaria para que funcione correctamente una vez ensayado.

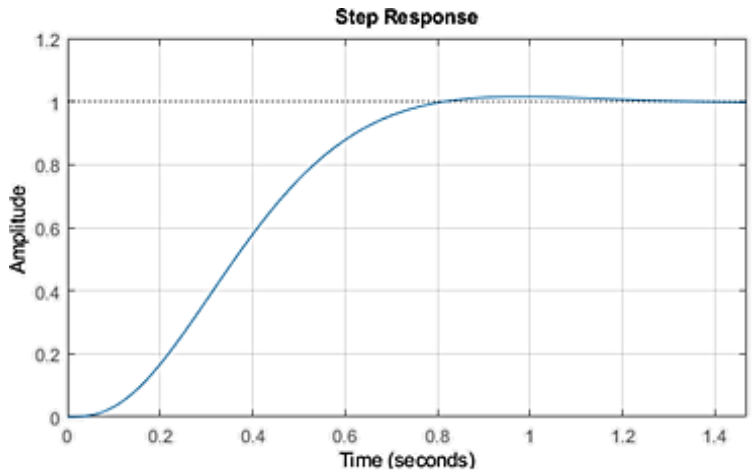

Figura 5.25. Respuesta a un escalón unitario en referencia del control de ángulo de guiñada.

Al contrario que en el caso de los otros dos controles, no ha sido necesario realizar un segundo diseño, puesto que este ha funcionado correctamente al simularlo y se ha obtenido una respuesta adecuada, Figura 5.25.

Simulación

De la misma forma que para los otros dos diseños de control, una vez realizado el diseño es necesario compararlo con la respuesta obtenida en la simulación para comprobar que se trata de un diseño adecuado.

En la simulación de este control, Figura 5.26, se obtiene una respuesta muy parecida a la esperada ($M_p = 1.54\%$) cuyo sobrepaso tiene un valor de:

$$M_p = \frac{y_{máx} - y_\infty}{y_\infty - y(0)} = \frac{66.27 - 60}{60 - (-60)} = 5.225\%$$

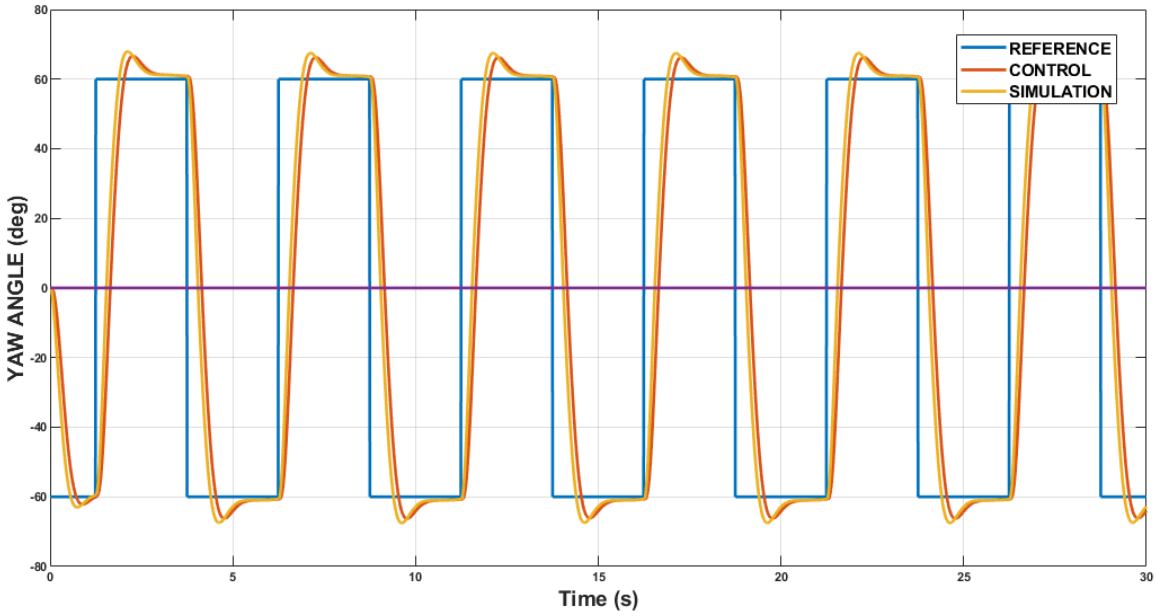

Figura 5.26. Simulación del control de ángulo de guiñada diseñado.

Las medidas se corresponden con la señal del *control* de color rojo, Figura 5.27.

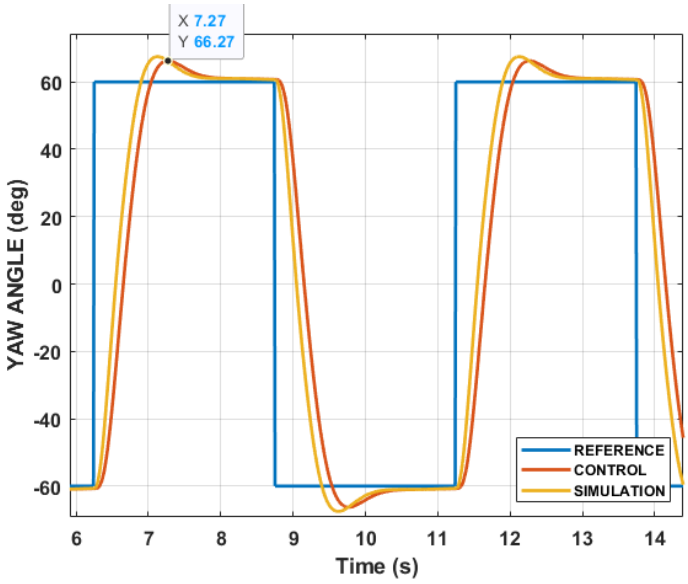

Figura 5.27. Sobrepaso del control de ángulo de guiñada diseñado.

Ensayo

Se repite el proceso realizado para el diseño de los otros dos controles, una vez realizada la comprobación de que la respuesta obtenida en la simulación es similar a la esperada, se

procede a realizar un ensayo con las especificaciones escogidas en el segundo diseño del control de velocidad de avance, que será el definitivo.

Esta vez ha sido necesario utilizar también el sistema de localización externo, para llevar un control del ángulo de guiñada a través de las cámaras. Esto ha servido para comprobar el funcionamiento completo de todos los componentes del proyecto a la vez, puesto que hasta el momento los ensayos que se habían realizado habían sido separando la parte del sistema de localización del resto de elementos del proyecto.

Las referencias que se aplican en el ensayo, Figura 5.28, son una onda cuadrada en la referencia de velocidad de giro entre 60^0 y -60^0 y período igual a 5 s y una referencia nula para la velocidad de avance y giro.

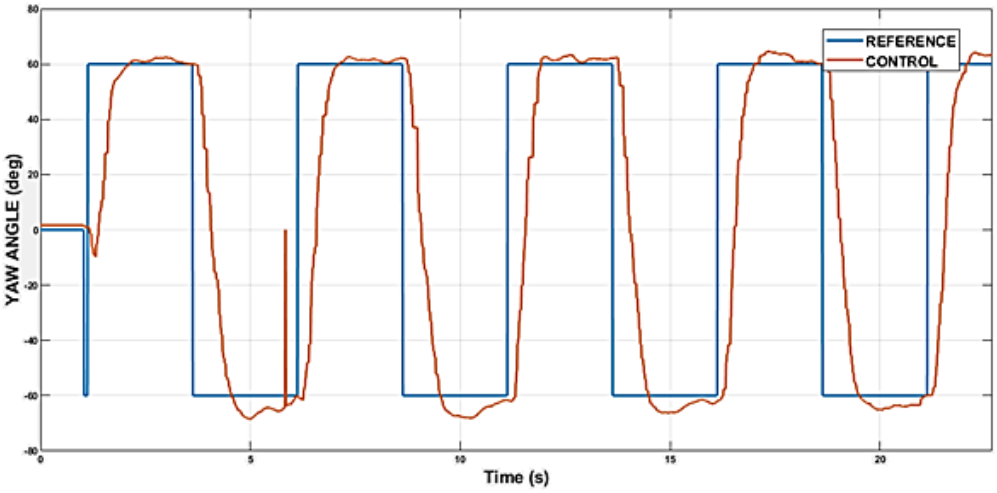

Figura 5.28. Ensayo del control de ángulo de guiñada.

De nuevo se observa que la respuesta sigue la referencia de forma adecuada a pesar de las pequeñas irregularidades que se muestran y que podrían ser debidas a la sensibilidad al ruido que presenta el control a causa de la acción derivativa.

Además, la respuesta no se comporta de forma simétrica respecto al eje X, si no que la parte "negativa" resulta más similar a la onda simulada. Esto puede ser debido a alguna no linealidad del vehículo, por la asimetría propia de los motores que podría no estar contemplada en el modelo.

Comparación del ensayo y la simulación

Una vez realizado el ensayo se realiza la última etapa del proceso llevándose a cabo la comparación de la respuesta obtenida durante el ensayo con la obtenida en la simulación, Figura 5.29.

- En color *rojo* se muestra la simulación.
- En color *amarillo* se muestra el ensayo.

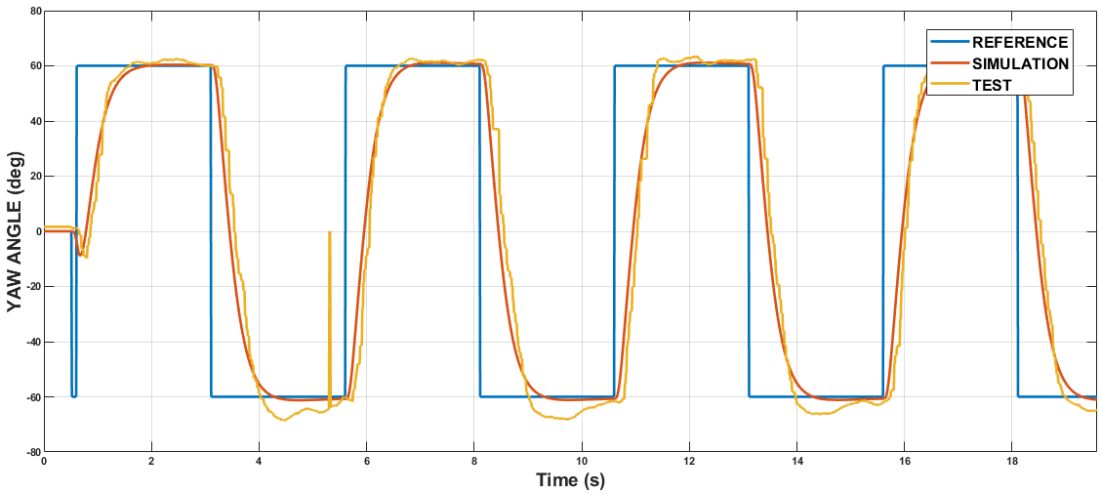

Figura 5.29. Comparación del ensayo y la simulación del control de ángulo de guiñada.

La respuesta obtenida en el ensayo no es tan similar a la simulada como en el caso de los otros dos controles, Figura 5.30. Al comparar el ensayo y la simulación, se observa que la parte "positiva" es casi idéntica en ambos casos, sin embargo, la parte "negativa" no se asemeja tanto.

Aun así, es lo suficientemente parecido como para considerar que el diseño del control del ángulo de guiñada se ha realizado de forma correcta. Este diseño se empleará durante el resto del proyecto.

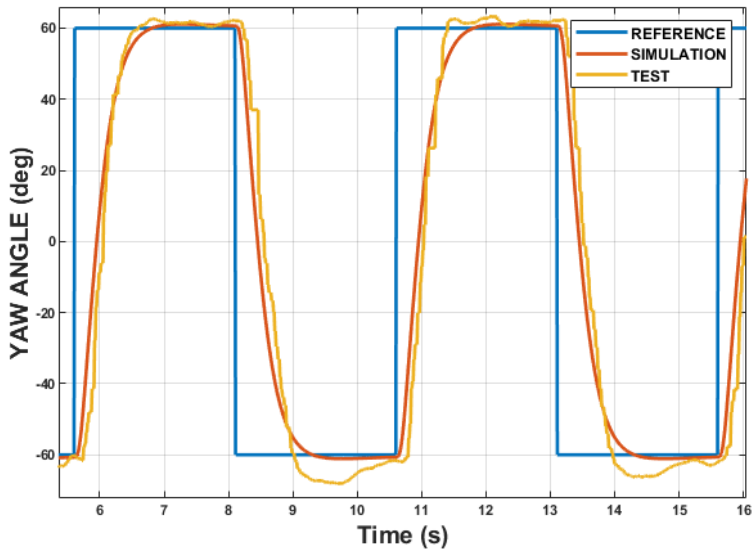

Figura 5.30. Comparación pendiente de la simulación y el ensayo para el control de ángulo de guiñada.

5.5 *SIMULINK PARA EL CONTROL DEL VEHÍCULO*

Para el desarrollo de todas las etapas explicadas en la Sección 5.4 se ha empleado el fichero de Simulink que integra todo lo necesario para controlar el vehículo *CAR_CONTROL_SYSTEM*, Figura 5.31.

De la misma forma que en el diagrama de *MCS_CONTROL_STATION* explicado en la Sección 4.1, en este fichero la información se almacena en los buses que se encuentran en la parte superior del sistema: *Control, Model* y *MSG.* De esta forma se tiene un mayor acceso y control de todas las variables contenidas en los buses, lo que simplifica su tratamiento.

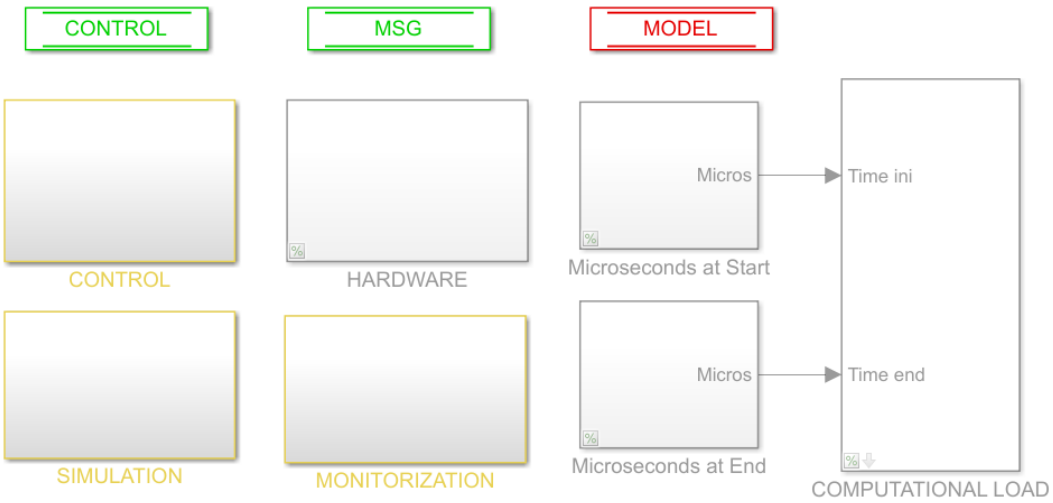

Figura 5.31. Diagrama de Simulink de CAR_CONTROL_SYSTEM.

- El bus *Control* almacena las variables propias del control del vehículo, donde se incluyen tanto las medidas de los sensores como los valores que se estiman mediante el Filtro de Kalman Extendido; además de los valores de las salidas y los mandos que genera el control.

- El bus *Model* contiene los parámetros necesarios para modelar el vehículo, sensores y actuadores.

- En el bus *MSG* se encuentran las variables que intervienen en las comunicaciones entre el ordenador de monitorización y la Raspberry Pi.

Los subsistemas que forman el diagrama se comentan o descomentan en función del modo en que se implemente el programa. En el caso de la Figura 5.31, el modo de implementación es de simulación. Esto se puede saber por los subsistemas que se encuentran comentados: *Hardware* y *Computational Load*, que son aquellos subsistemas en los que intervienen elementos físicos.

Por otra parte, los subsistemas que forman este diagrama son los siguientes:

- El subsistema *Control,* Figura 5.32, contiene la máquina de estados y el control del vehículo, que permite calcular los valores de los mandos que se deben aplicar a través de las medidas, las estimaciones del EKF y el control diseñado.

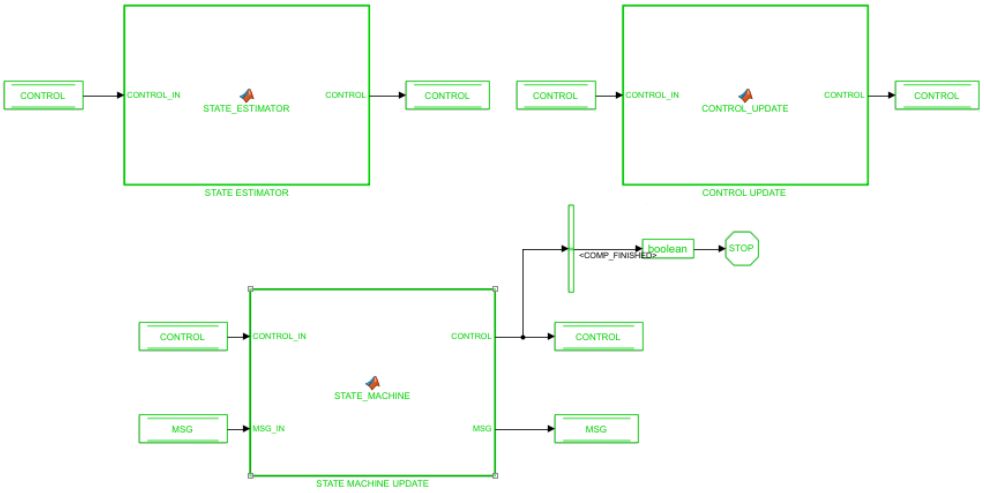

Figura 5.32. Subsistema de Control.

- El subsistema *Hardware,* Figura 5.33, contiene otros tres subsistemas: *Sensors, Actuators* y *Communications*. Estos almacenan los *scripts* encargados de controlar los elementos hardware que los componen y también las funciones de Matlab que controlan las comunicaciones del vehículo, entre ellas la encargada de la comunicación Bluetooth con las cámaras, *BT_RX*.

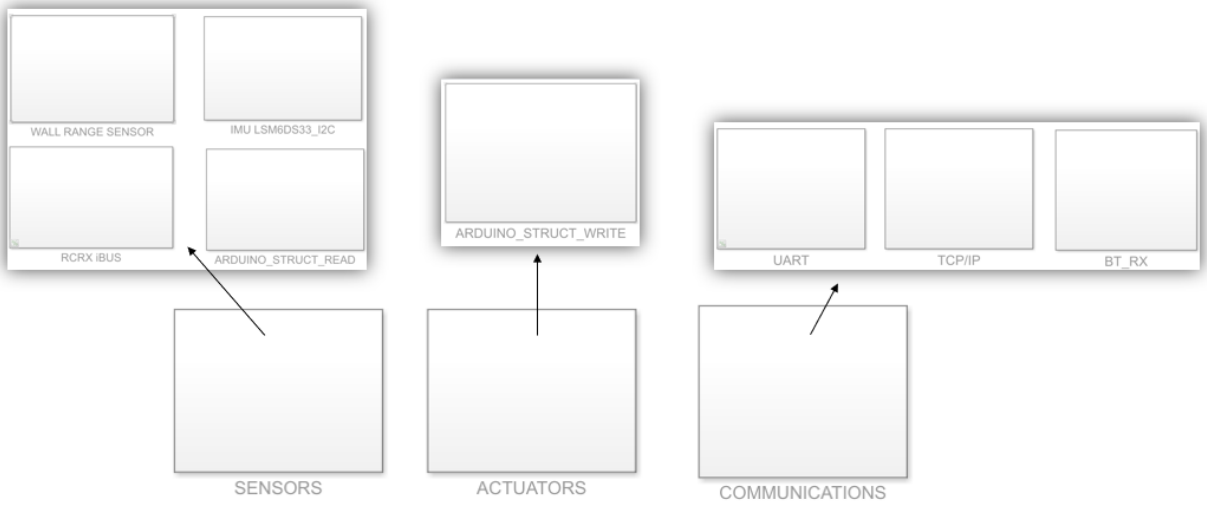

Figura 5.33. Subsistema Hardware.

- El subsistema *Monitorization*, Figura 5.34, contiene 6 subsistemas encargados de monitorizar mostrando a través de *displays* o *scopes* las variables de interés según el modo de implementación del programa. De esta forma se registran y controlan las diferentes variables que puedan resultar interesantes para el desarrollo del proyecto. Los subsistemas que se han empleado en el proyecto han sido: *TEST:SCOPES, HARDWARE:SCOPES* y *SIMULATION:SCOPES.*

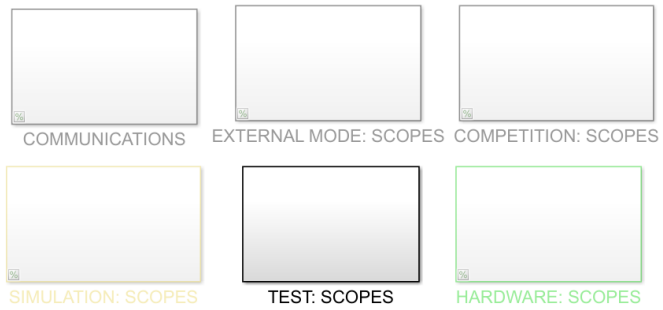

Figura 5.34. Subsistema Monitorization.

- El subsistema *Simulation,* Figura 5.35, que contiene las funciones de Matlab donde se definen los modelos del vehículo y de sus sensores. Además, se encuentra la configuración del funcionamiento de los tres botones con los que cuenta el vehículo y que se emplean durante los ensayos para poner en marcha, parar y devolver al estado inicial al vehículo.

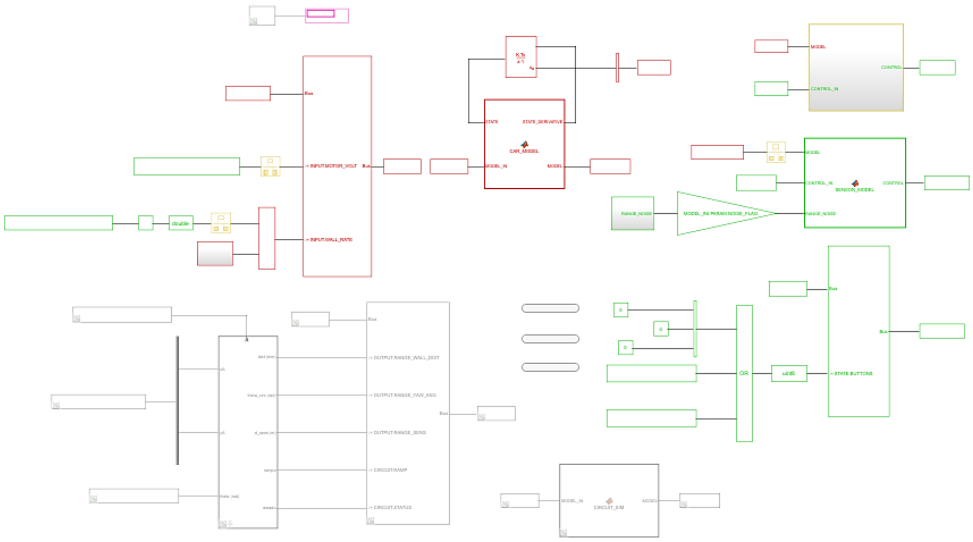

Figura 5.35. Subsistema de Simulación.

Capítulo 6. NAVEGACIÓN DEL VEHÍCULO

En este capítulo se explica y describe el proceso que se ha llevado a cabo para diseñar el sistema de navegación en el plano XY del vehículo, que es el objetivo final de este proyecto.

En primer lugar, se explica detalladamente la estructura de control del vehículo que se trata de un control en cascada con dos lazos de control, uno interno y otro externo, que constituye el sistema de navegación. Se describe también el entorno de navegación donde se encuentran los *waypoints* o puntos de paso que formarán la secuencia que debe seguir el vehículo.

Por último, para comprobar el funcionamiento del control, han llevado a cabo dos simulaciones para dos trayectorias previamente definidas y diferentes y se ha comprobado el funcionamiento del sistema de navegación.

6.1 INTRODUCCIÓN AL SISTEMA DE NAVEGACIÓN

El control de navegación que se diseña en este proyecto se trata de un control de velocidad y giro que permite recorrer trayectorias definidas por puntos de paso, *waypoints*.

Con este fin, se utiliza una estructura de control en cascada, con dos lazos de control, Figura 6.1:

- Lazo de control interno que contiene los controles PID diseñados y descritos en el Capítulo 5. Se trata de los controles de velocidad de avance y ángulo de guiñada. Las variables que controla este lazo son la velocidad de avance v y el ángulo de giro θ que actúan sobre las tensiones común y diferencial de los motores. Se requiere la función de transferencia de lazo cerrado del control interno para diseñar el lazo de control externo.

- Lazo de control externo. Las variables de control o mando son las referencias del lazo de control interno, es decir las referencias de la velocidad de avance v_{ref} y del ángulo de giro θ_{ref}. Se puede establecer una analogía de conducción donde la velocidad de avance se asemeja al acelerador del vehículo y el ángulo de guiñada representa el volante. Se emplean los valores de velocidad y posición en los ejes X e Y, además de las coordenadas de los *waypoints* definidas en los ejes X e Y.

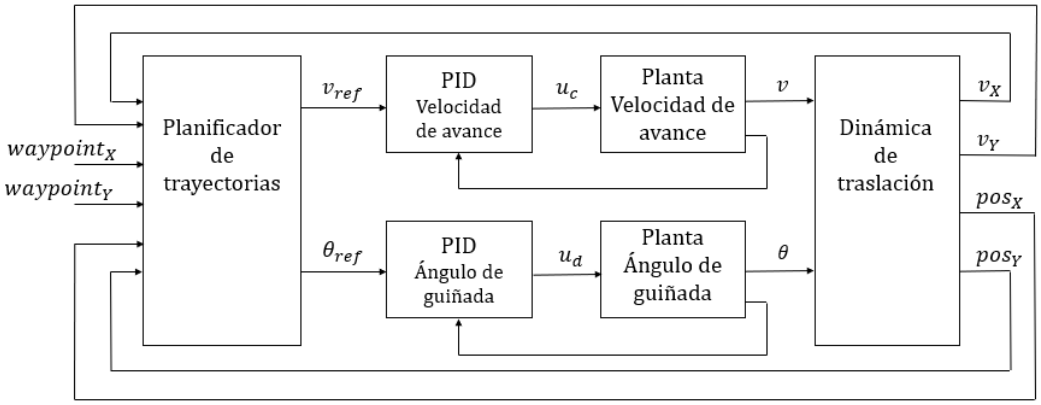

Figura 6.1. Estructura de control del vehículo.

El objetivo final consiste en recorrer una trayectoria definida por unos puntos de paso o *waypoints* en el menor tiempo posible, por lo que resulta necesario optimizar los controles.

La trayectoria se define en un recinto cuadrado cuyo lado es de longitud L=2m, y donde los puntos de paso son los extremos y los puntos medios de cada lado y el punto central del recinto, Figura 6.2. De esta forma es posible definir múltiples trayectorias dependiendo de la secuencia seleccionada de los puntos de paso. En este proyecto se ha supuesto que cualquier trayectoria siempre empieza y acaba en el centro del recinto, *waypoint* número uno.

Por otra parte, en el proyecto se ha considerado que se alcanza un punto de paso, cuando el origen del sistema de referencia del vehículo (centro de masas) atraviesa el círculo de radio R=0.05m cuyo centro es el punto de paso correspondiente.

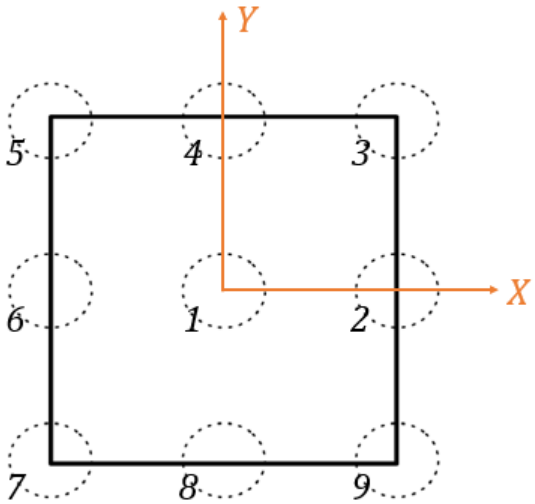

Figura 6.2. Recinto donde se encuentran los waypoints que forman las trayectorias del vehículo.

6.2 PLANIFICADOR DE TRAYECTORIAS

El planificador de trayectorias se encuentra dentro del lazo externo y es el componente que a partir de la posición $\vec{r}[k]$ y la velocidad $\vec{v}[k]$ actuales, de los dos puntos siguientes de paso $\vec{r}wp(n)$ y $\vec{r}wp(n+1)$ y de la velocidad máxima deseada $vMAX$, determina la secuencia futura de las referencias de la velocidad ($\vec{v}ref[k+1]\cdots\vec{v}ref[k+P]$) durante el horizonte de predicción, Figura 6.3.

Se emplea por lo tanto un planificador de trayectorias basado en *splines*. Se utiliza la optimización numérica para encontrar los parámetros de la curva que obedecen las *constraints* o limitaciones de la ruta que debe seguir.

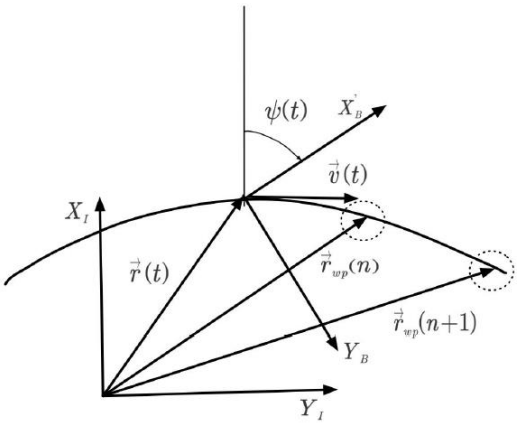

Figura 6.3. Planificación de la trayectoria de referencia para el control de navegación

El software de Matlab/Simulink que se emplea en el proyecto, incluye una versión básica del planificador de trayectoria que se basa en una aproximación por funciones polinómicas derivables, en el caso de este proyecto se trata de un polinomio de tercer orden. En la Figura 6.4 se representa el fundamento de este planificador de trayectorias para el eje X. Para el eje Y, se aplica otro planificador exactamente igual.

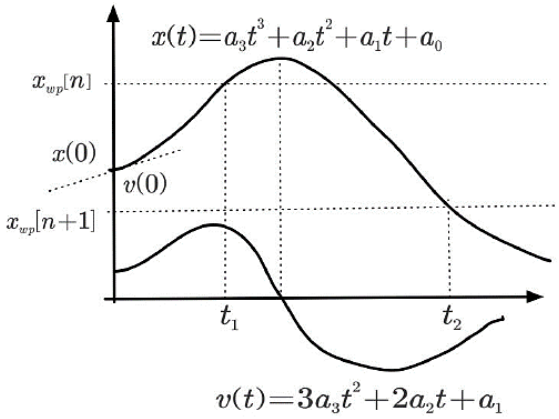

Figura 6.4. Planificador de la trayectoria en el eje X basado en polinomios cúbicos

El funcionamiento de este planificador de trayectorias es el siguiente:

Se busca definir la trayectoria en cada eje mediante un polinomio de tercer orden en función del tiempo que pase por la coordenada X de la posición actual $x(0)$ y por las coordenadas X de los dos próximos puntos de paso $xwp[n]$ y $xwp[n+1]$ en los instantes de tiempo $t1$ y $t2$, respectivamente.

La derivada en el instante inicial, es decir la pendiente de la recta, debe coincidir con la velocidad actual $v(0)$ en el eje X. A partir de estas condiciones se pueden determinar todos los coeficientes del polinomio $x(t) = a_3t^3 + a_2t^2 + a_1t + a_0$.

Una vez obtenido este polinomio, se pueden determinar los valores futuros de la velocidad en el eje X. Para ello, se calcula la derivada del polinomio anterior $x(t)$, dando resultado al polinomio $v(t) = 3a_3t^2 + 2a_2t + a_1$, en los instantes correspondientes a las muestras del horizonte de predicción desde ts hasta Pts, siendo ts el período de muestreo del control de navegación.

Para determinar los instantes $t1$ y $t2$ correspondientes a los dos siguientes puntos de paso, se aplica el siguiente algoritmo iterativo:

1. Se calculan las distancias $d1$ entre la posición actual $\vec{r}(t)$ y el siguiente punto de paso $\vec{r}wp(n)$ y la distancia $d2$ entre los dos siguientes puntos de paso $\vec{r}wp(n)$ y $\vec{r}wp(n+1)$.

2. Se estiman los tiempos que hay que invertir para recorrer esas distancias a la velocidad máxima deseada v_{MAX}: $t_1 = d_1/v_{MAX}$ y $t_2 = t_1 + d_2/v_{MAX}$.

3. Con esta estimación inicial de tiempos, se determinan los polinomios que definen la posición y la velocidad en cada eje, se evalúan en el horizonte de predicción y se calcula la velocidad máxima en dicho horizonte.

4. Si la velocidad máxima calculada en el horizonte es inferior a la velocidad máxima deseada $vMAX$, se termina el proceso. En caso contrario, se amplían los intervalos temporales $t1$ y $t2-t1$, multiplicándolos por un factor igual a cociente entre la velocidad máxima calculada y la velocidad máxima deseada, es decir de forma proporcional, y se vuelve al punto 3.

6.2.1 SIMULACIONES DE DOS TRAYECTORIAS DIFERENTES

Antes de realizar los ensayos necesarios para comprobar el correcto funcionamiento del planificador de trayectorias, se han llevado a cabo dos simulaciones para dos trayectorias diferentes.

Primera trayectoria

La primera trayectoria definida sigue la siguiente secuencia de *waypoints*: 1 ,2, 3, 4, 5, 1, 9, 8, 7, 6, 1, 1. Teniendo en cuenta la ubicación de los *waypoints* que se muestra en la Figura 6.2, los movimientos que debe realizar el vehículo, tomando como punto de partida el *waypoint* 1, se muestran en la Figura 6.5.

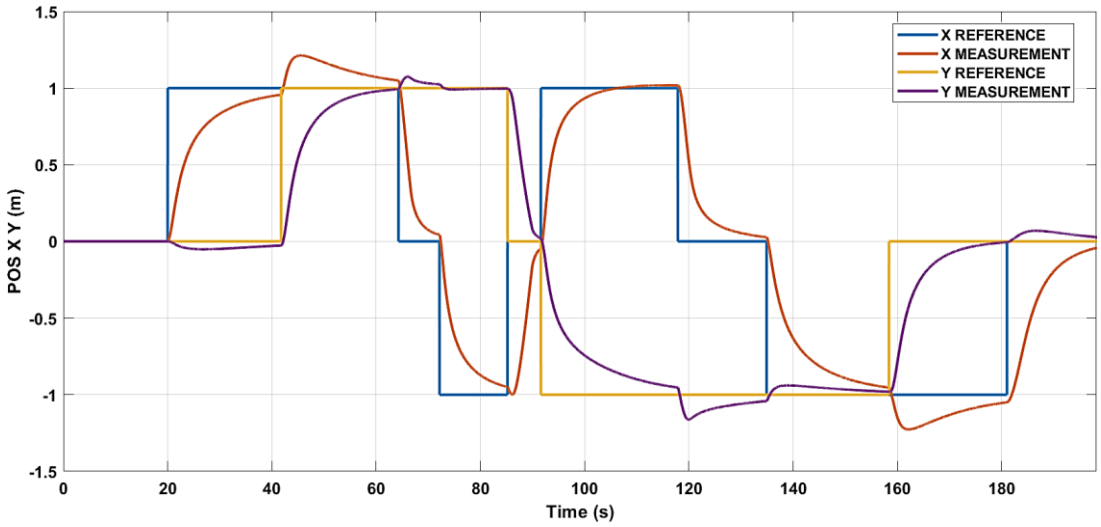

Figura 6.5. Simulación de la primera trayectoria definida.

- En color *azul* se muestra la referencia en el eje X.
- En color *amarillo* se muestra la referencia en el eje Y.
- En color *rojo* se muestra la posición real del vehículo durante la navegación, en el eje X.
- En color *morado* se muestra la posición real del vehículo durante la navegación, en el eje Y.

La trayectoria que sigue el vehículo, contenida en el plano XY, se muestra en la Figura 6.6.

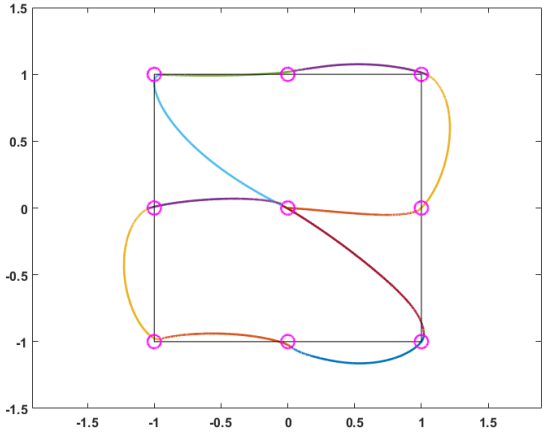

Figura 6.6. Trayectoria del vehículo contenida en el plano XY.

Segunda trayectoria

La segunda trayectoria definida sigue la siguiente secuencia de *waypoints*: 1 ,6, 4, 2, 8, 5, 3, 2, 1, 7, 1, 1. Teniendo en cuenta la ubicación de los *waypoints* que se muestra en la Figura 6.2, los movimientos que debe realizar el vehículo, tomando como punto de partida el *waypoint* 1, se muestran en la Figura 6.7.

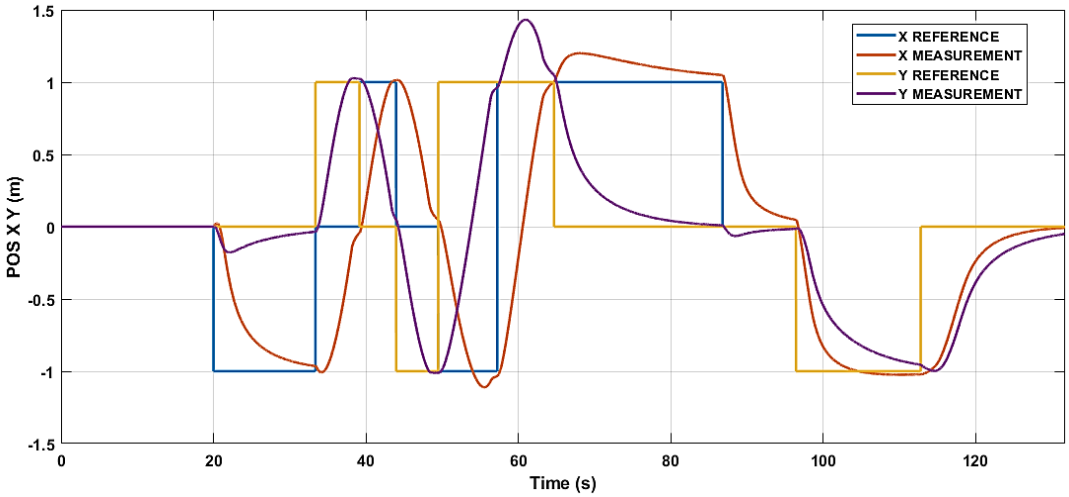

Figura 6.7. Simulación de la segunda trayectoria definida.

- En color *azul* se muestra la referencia en el eje X.
- En color *amarillo* se muestra la referencia en el eje Y.
- En color *rojo* se muestra la posición real del vehículo durante la navegación, en el eje X.
- En color *morado* se muestra la posición real del vehículo durante la navegación, en el eje Y.

La trayectoria que sigue el vehículo, contenida en el plano XY, se muestra en la Figura 6.8.

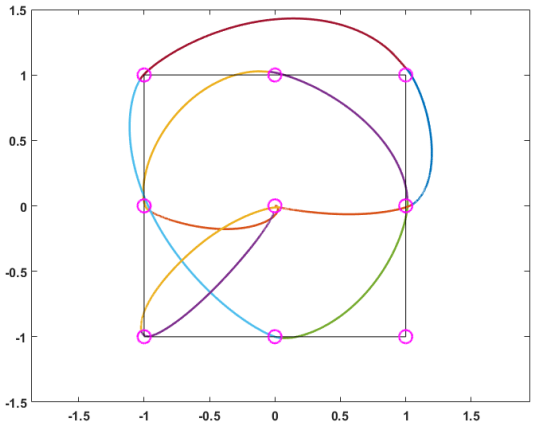

Figura 6.8. Trayectoria del vehículo contenida en el plano XY.

Aunque parezca que la primera trayectoria es menos complicada, el vehículo tarda un tiempo mayor en recorrerla, aproximadamente 200s, frente a 140s aproximados que se tarda en recorrer la segunda trayectoria.

Una vez realizadas las simulaciones y comprobado el correcto funcionamiento del sistema de navegación, se llevaron a cabo varios ensayos. Durante la realización de los ensayos, surgieron varios problemas que no se pudieron solucionar puesto que no se contaba con el tiempo necesario.

Capítulo 7. CONCLUSIONES, APORTACIONES Y FUTUROS DESARROLLOS

Para finalizar la memoria, en este último capítulo se desarrollan las conclusiones del proyecto, donde se explican las dificultades e inconvenientes que se han presentado durante el desarrollo del mismo.

También se explican las aportaciones que se han podido realizar, que podrían servir de ayuda a futuros proyectos.

Por último, se presentan posibles trabajos de mejora o continuación de este proyecto que se podrían llevar a cabo.

7.1 CONCLUSIONES

Durante la realización del proyecto se han presentado diferentes inconvenientes que han requerido de más o menos tiempo en función de su dificultad y que se describen a continuación:

La parte más problemática ha sido el sistema de localización externo

Se ha dedicado un tiempo mayor al esperado puesto que para poder comprender el funcionamiento completo del sistema de cámaras infrarrojas, primero fue necesario leer muchos documentos técnicos.

Por otra parte, una vez conocido el sistema de localización, fue necesario llevar a cabo la transmisión de los datos de captura mediante el fichero *MCS_CONTROL_STATION*. También se dedicó mucho tiempo a esta parte del proyecto, puesto que los códigos de Matlab-Simulink no eran simples y dieron muchos problemas antes de comenzar a funcionar de manera adecuada. Fue necesario invertir más tiempo del esperado en arreglar los problemas que fueron surgiendo, como en el caso de la transmisión en tiempo real, que finalmente se ha llevado a cabo mediante un bloque de Simulink de *Simulation Pace* que simula el tiempo real.

Además, una vez solucionados los problemas y comprobado su correcto funcionamiento, cuando se probó de nuevo esta parte con el resto de elementos del proyecto volvieron a surgir problemas, esta vez con la parte de transmisión Bluetooth del fichero de Simulink *MCS_CONTROL_STATION.*

La comunicación Bluetooth resultó ser menos trivial de lo esperado

No se había probado antes la comunicación bluetooth con este vehículo, por lo que fue necesario empezar desde cero, por lo que también se le dedicó más tiempo del esperado.

Para ello, inicialmente hubo que configurar los módulos bluetooth empleados en la comunicación y realizar las previas pruebas necesarias para verificar su correcto funcionamiento. Además, hubo que programar los *scripts* del codificador y decodificador empleados, y crear los ficheros de Simulink necesarios para la transmisión y recepción de los datos.

Tanto la parte de recepción como la de transmisión resultaron ser más complicadas que lo que se esperaba. El decodificador programado es más complejo del que inicialmente se pensó hacer, puesto fue necesario mejorarlo para que la recepción se realizará de forma correcta.

7.2 APORTACIONES Y OBJETIVOS ALCANZADOS

Las principales aportaciones de este proyecto son las siguientes:

Estudio y análisis del sistema de localización externo Optitrack

Fue necesario estudiar la herramienta software, Motive, que permitiese controlar al sistema de localización externo formado por cámaras infrarrojas y que constituye uno de los elementos más importantes y necesarios del sistema de navegación.

Durante el análisis y estudio de su control y la transmisión de los datos de captura, se fueron registrando los conocimientos adquiridos en una guía que cuenta con toda la información necesaria para poder emplear el sistema en proyectos futuros y que se encuentra resumida en el ANEXO B: Configuración de la Raspberry Pi.

Programación en Matlab-Simulink de los ficheros necesarios para la obtención de los datos de captura

Para poder utilizar los datos de captura del sistema de localización externo, primero había que filtrar y transformar esa información para que resultase útil y fácil de emplear en los controles y el sistema de navegación. Todo esto se ha hecho a través de un fichero de Simulink, *MCS_CONTROL_STATION*, que contiene todo lo necesario para obtener la información, transformarla y enviarla al vehículo.

Estos ficheros podrían emplearse en otros robots para la detección y transmisión de su ubicación en tiempo real sin realizar prácticamente ninguna modificación.

Configuración del sistema de comunicación Bluetooth

Para cerrar el sistema de comunicaciones del proyecto, fue necesario emplear además de la comunicación Wifi, una estructura Bluetooth que permitiera conectar el ordenador desde el que se rastrea al vehículo con el propio vehículo.

Para ello hubo que configurar los módulos bluetooth y programar los ficheros en Matlab-Simulink necesarios para el funcionamiento. Concretamente se ha creado un fichero de

transmisión Bluetooth para enviar las coordenadas en tiempo real al vehículo y otro que se incluye dentro de los archivos de control del vehículo de recepción de datos. Ambos se podrían emplear en futuros proyectos en los que interviniese la transmisión Bluetooth.

Diseño de los controles del vehículo mediante Matlab-Simulink

Se han diseñado tres controles para el vehículo: control de velocidad de avance, velocidad de giro y ángulo de guiñada, que se han simulado y ensayado con el fin de comprobar el correcto funcionamiento. Se han llevado a cabo varios diseños y finalmente se han escogido aquellos que permitían obtener una mejor respuesta en frecuencia.

Diseño del sistema de navegación del vehículo

Por último, se ha diseñado el sistema de navegación del vehículo y se ha simulado el planificador de trayectorias para comprobar el correcto funcionamiento del mismo. Para el diseño de este sistema de navegación, se han empleado los controles nombrados anteriormente, junto con el planificador de trayectorias basado en *splines*. De nuevo, este sistema de navegación podría resultar útil para proyectos futuros.

7.3 FUTUROS TRABAJOS

Programación del archivo .tlc del fichero MCS_CONTROL_STATION

Actualmente, la transmisión de datos del sistema de localización externo al vehículo se lleva a cabo mediante un bloque de *Simulation Pace* que simula el tiempo real. Esto se debe a que la función empleada para recibir los datos en Simulink desde Motive (*FuncMotive*) que utiliza la API facilitada por *NatNet*, necesita un archivo *.tlc* para que la transmisión de datos se realice en tiempo real.

Aunque con este bloque el funcionamiento del fichero es el correcto y cumple con los requisitos necesarios, resultaría interesante estudiar el posible desarrollo del archivo *.tlc* que permitiese dejar de correr el fichero en modo de simulación y empezar a hacerlo como el resto del sistema, en tiempo real. El envío de datos sobre las coordenadas del vehículo sería posiblemente más preciso y por lo tanto podría mejorar considerablemente el funcionamiento del sistema de navegación.

Realizar ensayos con el equipo real del sistema de navegación

Uno de los futuros desarrollos tiene que ver con la optimización y depuración del sistema de navegación del proyecto, puesto que, aunque se ha comprobado mediante simulación que el funcionamiento es el correcto, no se ha llegado a ensayar obteniendo resultados satisfactorios.

Se deberían solventar los problemas que han surgido durante los ensayos y que han impedido corroborar el funcionamiento. Por otra parte, este mismo sistema podría servir como modelo o base para futuros proyectos que empleen otros robots y que tengan plantas diferentes.

Probar otros sensores en la navegación

El sistema de posicionamiento externo no tiene muchas aplicaciones puesto que resulta caro y aparatoso. Sería una buena idea probar a llevar a cabo la navegación con ayuda de otros sensores como podría ser un LIDAR o incluso colocar cámaras en el propio coche que identificaran los objetos que rodean al vehículo durante la navegación.

Capítulo 8. BIBLIOGRAFÍA

[1] K. Panetta. "5 Trends Appear on the Gartner Hype Cycle for Emerging Technologies, 2019". Gartner. *https://www.gartner.com/smarterwithgartner/5-trends-appear-on-the-gartner-hype-cycle-for-emerging-technologies-2019/* [Último acceso: 25/06/2021]

[2] NHTSA. *https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety#topic-road-self-driving* [Último acceso: 25/06/2021]

[3] J. Zhao et al., "The key technology toward the self-driving car", Escuela de Ingeniería en Automoción y Transporte, Politécnica Shenzhen, Shenzhen, China, 2017.

[4] M. J. González., "Análisis de la tecnología de posicionamiento indoor aplicada a robots autónomos móviles", Trabajo de fin de grado, Dpto. de Ingeniería de Sistemas y Automática, Universidad Carlos III de Madrid, Madrid, España, 2013.

[5] N. Buniyamin et al., "A Simple Local Path Planning Algorithm for Autonomous Mobile Robots", *International Journal of Systems Applications, Engineering & Development*, vol. 5, nº 2, pp. 151-159, 2011.

[6] O. Hachour, "Path planning of Autonomous Mobile robot", *International Journal of Systems Applications, Engineering & Development*, vol. 2, nº 4, pp. 178-190 , 2008. [En línea]. Disponible en: *http://www.universitypress.org.uk/journals/saed/saed-45.pdf* [Último acceso: 10/07/2021]

[7] M. Hirz y B. Walzel,"Sensor and object recognition technologies for self-driving cars.", *Computer-Aided Design and Applications*, vol 15, nº4, pp. 1-8, 2018. [En línea]. Disponible en: *https://www.researchgate.net/publication/322368082_Sensor_and_object_recognition_technologies_for_self-driving_cars* [Último acceso: 10/07/2021]

[8] Y. Alkhorshid et al., "Camera-Based Lane Marking Detection for ADAS and Autonomous Driving", Image Analysis and Recognition. ICIAR 2015. Lecture Notes in Computer Science, vol 9164. Springer, 2015. [En línea]. Disponible en: *https://doi.org/10.1007/978-3-319-20801-5_57* [Último acceso: 10/07/2021]

[9] F. Maciá Pérez et al,. "Sistema de control para robots inspirado en el funcionamiento y organización de los sistemas neurorreguladores humanos", Dpto. de tecnología Informática y Computación, Universidad de Alicante, Alicante, España. [En línea]. Disponible en: *https://rua.ua.es/dspace/bitstream/10045/13610/1/sistemas%20de%20control%20para%20robots.pdf* [Último acceso: 10/07/2021]

[10] Ü. Dogan, J. Edelbrunner and I. Iossifidis, "Autonomous driving: A comparison of machine learning techniques by means of the prediction of lane change behavior," 2011 *IEEE International Conference on Robotics and Biomimetics*, 2011, pp. 1837-1843, doi: 10.1109/ROBIO.2011.6181557.

[11] J.M. Molina López, V. Matellán Olivera, "Robots autónomos: Arquitecturas y control", Laboratorio de Agentes Inteligentes (LAI), Dpto. de Informática, Universidad Carlos III de Madrid, Madrid, España. [En línea]. Disponible en: *https://upcommons.upc.edu/bitstream/handle/2099/9648/Article007.pdf?sequence=1&isAllowed=y* [Último acceso: 10/07/2021]

[12] Murphy, R. R. Introduction to AI robotics. MIT Press, 2000.

[13] Arkin, R. C. Behavior based robotics. MIT Press, 1998.

[14] Coste-Maniére, E. y Simmons, R.G. Architecture, the backbone of robotic systems. *In Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pp. 67–72, San Francisco, CA (USA), April 2000.

[15] Elbanhawi, Mohamed, Simic, Milan. "Continuous-Curvature Bounded Trajectory Planning Using Parametric Splines". *Frontiers in Artificial Intelligence and Applications*, Chania, Greece, 2014, vol. 262, pp. 513-522, doi: 10.3233/978-1-61499-405-3-513.

[16] B. Su, L. Zou, "Manipulator Trajectory Planning Based on the Algebraic-Trigonometric Hermite Blended Interpolation Spline", *Procedia Engineering,* vol. 29, pp. 2093-2097, 2012.

[17] J. S. Furtado, H.H.T.Liu et al., "Comparative Analysis of OptiTrack Motion Capture Systems", *Proceedings of The Canadian Society for Mechanical Engineering International Congress, CSME International Congress*, 2018, Toronto, Canada.

[18] Optitrack. *https://optitrack.com/*. [Último acceso: 29/06/2021]

[19] Tracklab. *https://tracklab.com.au/products/brands/optitrack/optitrack-flex-cameras/optitrack-flex-13/.* [Último acceso: 29/06/2021]

[20] Motive. *https://v22.wiki.optitrack.com/index.php?title=Motive_Documentation.* [Último acceso: 29/06/2021]

[21] G. Nagymáté, R.M. Kiss. "Application of OptiTrack motion capture systems in human movement analysis. A systematic literature review". Dpto. de Mecatrónica, Óptica, e Ingeniería Mecánica Informática. Universidad de Budapest en Tecnologías y Economía, Budapest, Hungría. *Recent Innovations in Mechatronics* (RIiM), vol. 5, nº 1, p. 1-9 ,2018.

[22] Xataka. *https://www.xataka.com/makers/cero-maker-todo-necesario-para-empezar-raspberry-pi.* [Último acceso: 29/06/2021]

[23] RaspberryPi. *https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/*. [Último acceso: 29/06/2021]

[24] Atmel, Febrero 2014, "8-bit Microcontroller with 16/32K Bytes of ISP Flash and USB Controller, ATmega16U4, ATmega32U4, Preliminary", Atmel. Disponible en: *https://docs.rs-online.com/32ab/0900766b80eee70f.pdf*. [Último acceso: 10/07/2021]

[25] 5Hertz.*https://www.5hertz.com/index.php?route=tutoriales/tutorial&tutorial_id=1 K*. [Último acceso: 29/06/2021]

[26] C. Jiménez Cortés, "Control de un vehículo equilibrista mediante una Raspberry Pi", Trabajo de fin de grado, Ingeniería en Tecnologías Industriales, Escuela Técnica Superior de Ingeniería ICAI Universidad Pontificia de Comillas, Madrid, España, 2019.

[27] Pololu. *https://www.pololu.com/product/3064*. [Último acceso: 29/06/2021]

[28] Pololu Corporation, Diciembre 2019, "Micro Metal Gearmotors", Pololu. Disponible en: *https://www.pololu.com/file/0J1487/pololu-micro-metal-gearmotors-rev-4-2.pdf* [Último acceso: 10/07/2021]

[29] Texas Instrument Incorporated, Junio 2012, Abril 2021, "DRV883x Low-Voltage H-Bridge Driver", Texas Instrument. Disponible en: *https://www.ti.com/lit/ds/symlink/drv8838.pdf* [Último acceso: 10/07/2021]

[30] Pololu. *https://www.pololu.com/product/3081*. [Último acceso: 27/06/2021]

[31] F. L. Pagola y de las Heras, *Regulación Automática*, 1ª ed. Lugar de publicación: Universidad Pontificia de Comillas (publicaciones), Octubre 2006.

[32] M. F. Khan, R. ul Islam and J. Iqbal, "Control strategies for robotic manipulators," *2012 International Conference of Robotics and Artificial Intelligence*, 2012, pp. 26-33, doi: 10.1109/ICRAI.2012.6413422.

[33] T. Osuna Altamirano, "Navegación y Control de Robot Móvil", Trabajo de fin de grado, Maestría en Ciencias en Sistemas Digitales, Instituto Politécnico Nacional, Centro de Investigación y Desarrollo de Tecnología Digital, Tijuana, B.C., México, 2010.

[34] J.A. Martín Hernández, "Estudios sobre sistemas adaptativos con aplicaciones en la robótica autónoma y los agentes inteligentes", Tesis doctoral, Ingeniería en Informática, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, España, 2009.

[35] S. Torres et al,. "Acción autoadaptativa aplicada a controlador robusto en robots manipuladores", Grupo de Computadoras y Control. Dep. de Física F. y E., Electrónica y Sistemas, Universidad de La Laguna, Tenerife, España.

[36] ONU. *https://www.un.org/sustainabledevelopment/es/development-agenda/* [Último acceso:05/07/2021]

[37] ONU. *https://www.un.org/sustainabledevelopment/es/infrastructure/* [Último acceso: 05/07/2021]

[38] ONU. *https://www.un.org/sustainabledevelopment/es/cities/* [Último acceso: 05/07/2021]

ANEXO A: MOTIVE

Este anexo incluye una explicación más detallada de los posibles usos y aplicaciones de Motive, el software de Optitrack. Además, explica de una forma más minuciosa el uso del programa: cómo se debe configurar el entorno de captura de movimiento en la Sección A.1, los pasos a seguir después de la captura de vídeo en la Sección A.2 y finalmente, cómo llevar a cabo la transmisión de datos en la Sección A.3, punto clave en la realización de este proyecto. Se ha llevado a cabo la redacción de este Anexo ya que se considera que puede servir de ayuda y guía para alumnos que necesiten usar el programa de localización externo en algún proyecto o laboratorio.

A.1 CONFIGURACIÓN DEL ENTORNO DE CAPTURA

CALIBRACIÓN DE LAS CÁMARAS

Para poder comenzar con "el rastreo", primero es necesario calibrar todas las cámaras. Mediante la calibración de la cámara, Motive calcula tanto la posición y la orientación de las cámaras, como la cantidad de distorsiones de la lente en las imágenes capturadas. Usando los resultados de la calibración, Motive construye un volumen de captura 3D y, dentro de este volumen, se consigue el seguimiento del movimiento.

Al comenzar, se debe escoger si realizar una nueva calibración de las cámaras o por lo contrario se quiere seguir con una ya existente, Figura A. 1.

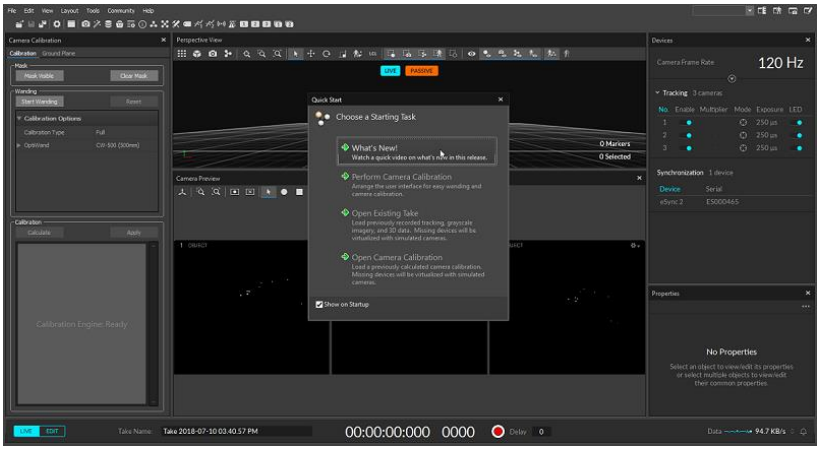

Figura A. 1. Menú principal de Motive

Todas las herramientas de calibración se pueden encontrar en el panel Calibración ("Calibration pane"). A continuación, se describen los pasos necesarios para llevar a cabo la calibración.

- ***MASKING:***

Es necesario comprobar que no haya ningún objeto que interfiera o reflejos de luz infrarroja en el espacio de captura, que puedan causar problemas.

A continuación, hay que abrir el panel de Calibración en Motive. Haciendo click en el menú *Layout* → *Calibrate* se accede al diseño de calibración. Después, pulsar  en "Camera Preview Pane".

Una vez se ha preparado el entorno, se puede enmascarar cualquier reflejo que siga permaneciendo (como las interferencias de luz infrarroja de otras cámaras) usando las "masking tools". Para enmascararlo pinchar en "Block Visible" del *Calibration Pane*, o pulsar en  en *el Camera Preview Pane* para aplicar las máscaras de forma automática.

Una vez se han aplicado todas las máscaras, todos los reflejos aparecerán ahora cubiertos con pixeles rojos.

- ***WANDING:***

Para este paso es necesario usar la *varilla de calibración*. A continuación, se debe asegurar que en el *Calibration Pane*, en "Type" pone "Full" y en la sección *OptiWand* se especifica el tipo correcto de varilla. Tras esto para comenzar, hay que pulsar en "*Start Wanding*" en el *Calibration Pane*. Después se debe mover la varilla de calibración por todo el entorno de las cámaras para que estas vayan recogiendo muestras, Figura A. 2.

Cuando el sistema indique que se han recogido suficientes muestras, hay que pulsar en el botón "Calculate" para comenzar el cálculo (puede tardar unos minutos). Se puede dejar calculando todo el tiempo que se quiera, cuanto más tiempo, más precisa será la calibración. Las muestras empezarán a volverse azules a medida que vaya mejorando la calidad de la calibración. Cuando se vuelva completamente azul, el cálculo habrá terminado.

Una vez aparezca el mensaje "Ready to Apply", se debe pulsar el botón "Apply Result".

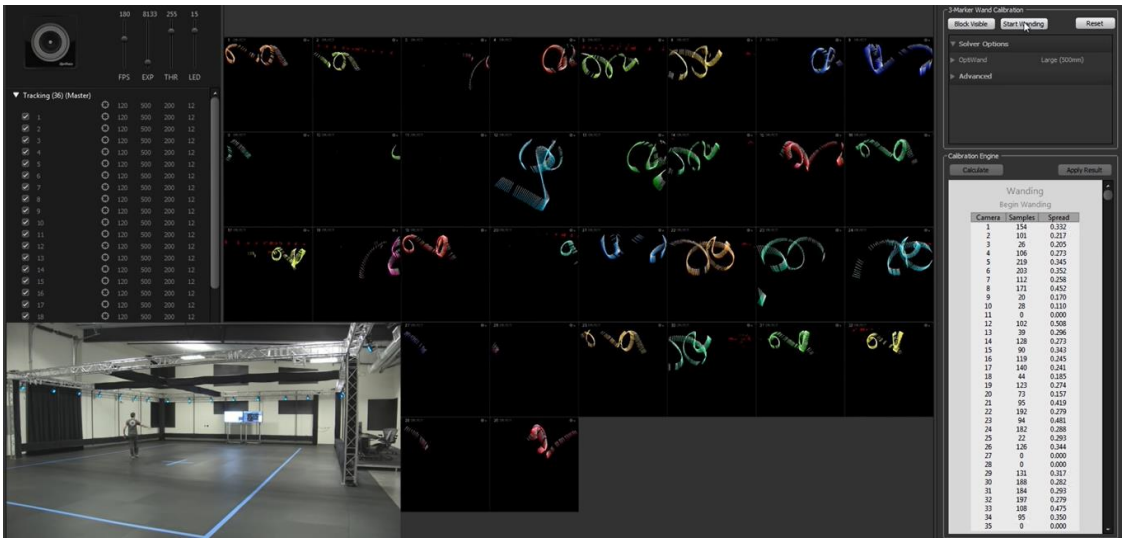

Figura A. 2. Proceso de "Wanding".

En este momento, aparecerá una venta de resultados de calibración. Después de haber analizado el resultado, hay que hacer click en "Apply" y así aplicar la calibración:

- ***SETTING THE GROUND PLANE:***

Una vez se han calibrado las cámaras, hay que definir el plano "de tierra" (en plano del suelo) del entorno de captura. Para ello hay que seguir las siguientes indicaciones:

En primer lugar, colocar el cuadrado de calibración en cualquier punto que se encuentre dentro del entorno de captura. Para que funcione de manera correcta, es necesario colocar el cuadrado de calibración de forma que el VÉRTICE esté justo en el que deseamos que sea el ORIGEN GLOBAL del sistema.

Para orientar de manera correcta el Cuadrado de calibración, es necesario colocar el "brazo" más largo en la posición del eje Z (positivo) y el brazo más corto en la posición del eje Y (positivo) que queramos que tenga nuestro sistema de referencia dentro del entorno, Figura A. 3.

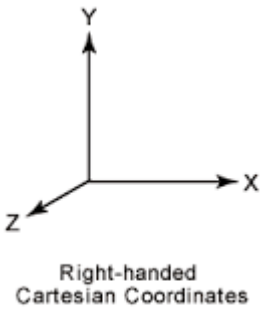

Figura A. 3. Sistema de coordenadas de Motive

Hay que asegurarse de que el cuadrado de calibración se encuentre en una superficie completamente plana y paralela al suelo o plano de "tierra".

A continuación es necesario acceder a la pestaña "Ground Plane" del Calibration Pane. Mientras los marcadores del cuadrado de calibración se seleccionan, hay que hacer click en "Set Ground Plane" en la sección Ground Plane Calibration (de esta forma se estará escogiendo el plano seleccionado como plano de tierra u plano de origen del sistema de referencia). Para finalizar, hay que guardar el archivo de calibración en la carpeta correspondiente.

CONFIGURAR UNA SESIÓN DE CAPTURA

- ***CAPTURE RECORDINGS:***

En Motive, cada captura se guarda en un archivo *.tak* y los archivos que guardan una relación se agrupan en "carpetas de sesión". Para empezar a grabar hay que seguir las indicaciones que se muestran a continuación:

Para comenzar es necesario crear una carpeta de sesión en el directorio que se desee. Tras esto hay que cargar la carpeta en "Data Management Pane" haciendo click en el icono  o simplemente arrastrando esta carpeta y soltándola en "Data Management Pane".

Si no se carga ninguna carpeta de sesión, se creará una de forma automática en el directorio de documentos. Las grabaciones que se hagan se guardarán dentro de la carpeta de sesión que se haya seleccionado, que aparecerá marcada con el símbolo:  .

- ***MOTIVE PROFILE:***

Las configuraciones de software de Motive se guardan en "Motive Profiles". Todas las configuraciones relacionadas con la aplicación se pueden guardar en los perfiles de Motive, y se pueden exportar e importar estos archivos manteniendo fácilmente las mismas configuraciones de software.

MARKER UP

Para comenzar, es necesario colocar los marcadores en el sujeto (cuerpo rígido, si se trata de un objeto o esqueleto, en el caso de que se busque rastrear movimientos humanos). Es importante comprobar que estos marcadores estan colocados de manera correcta. A través de ellos, se reconstruye el objeto en 3D.

Para el seguimiento de los cuerpos rígidos o esqueletos, primero hay que abrir el "Builder Pane" haciendo click en  en la sección *"View"*, ir a las opciones de creación de esqueletos y elegir el conjunto de marcadores que se va a utilizar. Para ello hay que seguir el diagrama del

esqueleto e ir colocando los marcadores. En la Figura A. 4 se muestra un ejemplo de cómo se deben colocar los marcadores:

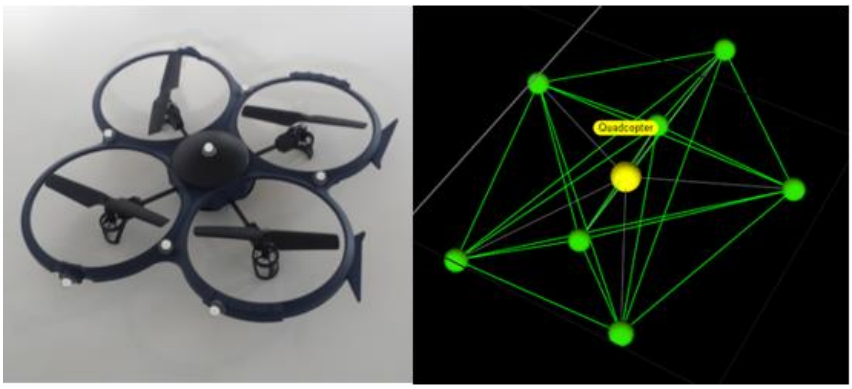

Figura A. 4. Marcadores colocados en un dron y el cuerpo rígido correspondiente definido en Motive.

En el "Builder Pane" se puede, tanto crear un nuevo cuerpo rígido/esqueleto al que rastrear, como modificar uno ya existente.

CÓMO DEFINIR CUERPOS RÍGIDOS

En primer lugar, para crear nuevos cuerpos a los que poder rastrear, hay que hacer click en *Layout* → *Creat Menu* para acceder a la función de diseño de creación del modelo.

- ***CREACIÓN DE UN CUERPO RÍGIDO:***

Para crear un cuerpo rígido, acceder al "Builder Pane" y seleccionar en primer lugar "*Rigid Body"* en la parte inferior del panel y a continuación *"Create"* en la parte superior.

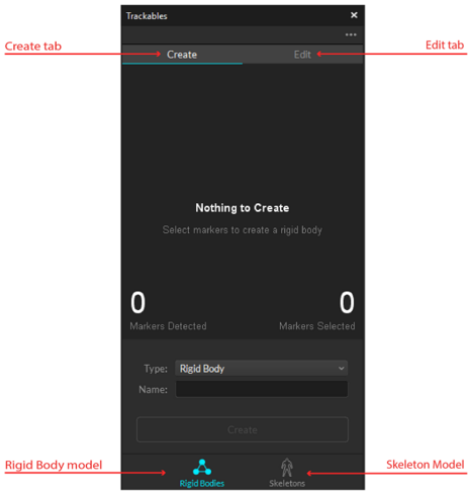

Figura A. 5. "Builder Pane" con opciones de crear o editar.

En primer lugar, hay que seleccionar todos los marcadores asociados del cuerpo rígido en la ventana gráfica 3D. Tras esto en el "Builder Pane", Figura A. 5, hay que confirmar que los marcadores seleccionados coinciden con los marcadores que se han colocado en el objeto, a través de los cuales se desea definir el cuerpo rígido. A continuación es necesario hacer click en *"Create"* para definir el cuerpo rígido, Figura A. 6.

Una vez creado el cuerpo rígido, los marcadores se "colorearán" (etiquetarán) y se interconectarán entre sí. Ahora el cuerpo rígido recién creado, debería aparecer en el "Assets Pane".

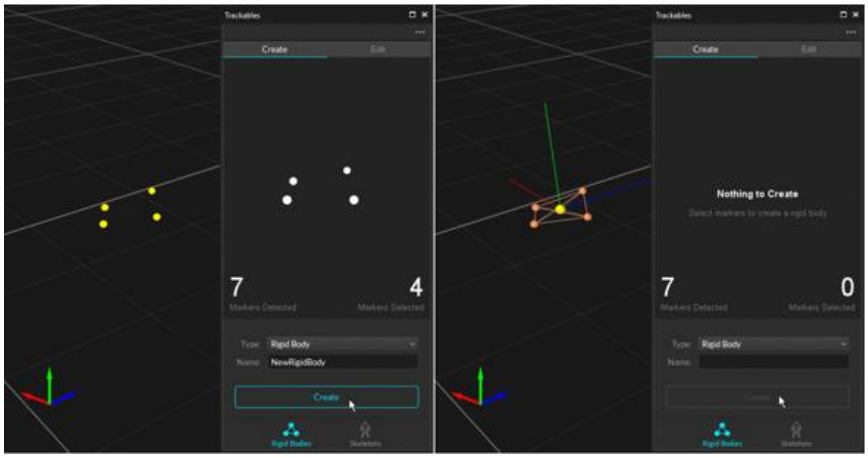

Figura A. 6. Creación de un cuerpo rígido.

CÓMO EDITAR CUERPOS RÍGIDOS

Para crear un cuerpo rígido, acceder al "Builder Pane" y seleccionar en primer lugar "*Rigid Body"* en la parte inferior del panel y a continuación *"Edit"* en la parte superior. Tras esto, se abrirán lasopciones para poder editar un cuerpo rígido.

- ***REFINE:***

La herramienta de refinamiento de cuerpos rígidos mejora la precisión del cálculo de cuerpos rígidos en Motive. Esta herramienta permite a Motive recolectar muestras adicionales en el "Live Mode" para conseguir resultados de seguimiento más precisos. Concretamente, se mejora el cálculo de las ubicaciones esperadas de los marcadores de cuerpo rígido, así como la posición y orientación del cuerpo rígido. Para ello seguir los siguientes pasos:

En primer lugar, es necesario abrir el "Builder Pane" pulsando en la pestaña "*View"* y seleccionar la opción *"Rigid body"* en la parte inferior del panel para a contiuación hacer click en *"Edit"* en la parte superior. En "Live Mode", se debe seleccionar un activo de cuerpo rígido existente que se desee refinar. En este momento, es necesario mantener el cuerpo rígido en el centro de volumen de captura, para que todas las cámaras puedan capturar mejor los marcadores del cuerpo rígido.

Cuando el cuerpo rígido esté bien colocado, hacer click en *"Start Refine"* en el "Builder Pane". Tras esto, hay que ir rotando lentamente el cuerpo rígido para que se recojan muestras desde diferentes orientaciones.

Una vez se hayan recogido todas las muestras necesarias, aparecerán los resultados del refinamiento

- ***LOCATION/ORIENTATION:***

La pestaña "*Edit*" se utiliza para aplicar traslación o rotación al punto desde el que pivota el cuerpo rígido seleccionado. Un punto de pivote de un cuerpo rígido representa tanto la POSICIÓN (x, y, z), como la ORIENTACIÓN (cabeceo, inclinación) del mismo.

1. LOCATION: Esta herramienta se usa para trasladar un punto de pivote en el eje x / y / z (en mm). También permite restablecer la traslación para volver a establecer el punto de pivote en el centro geométrico del cuerpo rígido.
2. ORIENTATION: Esta herramienta se usa para aplicar rotación al sistema de coordenadas del cuerpo rígido seleccionado. También puede restablecer la orientación para alinear el eje de coordenadas del cuerpo rígido y el eje global.

RECORD DATA

Para comenzar a grabar, es necesario acceder al *Layout → Capture Menu.* Una vez que se ha calibrado el volumen de captura y se ha definido el cuerpo rígido, ya se pueden hacer las grabaciones.

En primer lugar, en "Control Deck" que se encuentra en la parte inferior, hay que presionar el botón rojo que hace que se comience a grabar, o simplemente pulsar la barra espaciadora cuando esté en "Live Mode" para empezar con la captura.

Cuando se comience a grabar, el botón rojo se "iluminará" indicando que hay una grabación en curso. Para detener la grabación simplemente hay que hacer click de nuevo en el botón rojo. El archivo de captura que se haya grabado (archivo .tak) se guardará dentro de la carpeta de la sesión actual.

Una vez guardado el archivo de la grabación, se pueden reprodrucir capturas, reconstruirlas, editarlas y exportar sus datos en una gran variedad de fomatos, para poder hacer un análisis adicional.

A.2 DESPUÉS DE LA CAPTURA

EDICIÓN DE LOS DATOS

Tras terminar de grabar, los datos 3D se pueden post-procesar utilizando herramientas de edición de datos que se pueden encontrar en "Edit Tools Pane". Se puede acceder a este Panel haciendo click en la pestaña "View" y luego en  .

Estas herramientas proporcionan funciones que nos permiten eliminar trayectorias pocos fiables, suavizar trayectorias seleccionas e interpolar posiciones de marcadores faltantes. Esta edición de los datos puede suponer una mejora en la calidad de los datos de seguimiento. Los pasos generales de edición son los siguientes:

En primer lugar, es necesario revisar los marcos generales de una toma para conocer los marcos y marcadores que deberían "limpiarse". Lo siguiente es consultar en "Labeling Pane", Figura A. 7, e inspeccionar los porcentajes de espacio en cada marcador. Tras esto, se debe seleccionar un marcador que esté normalmente "bloqueado" o "perdido", es decir, que aparezcan lo suficiente en las capturas.

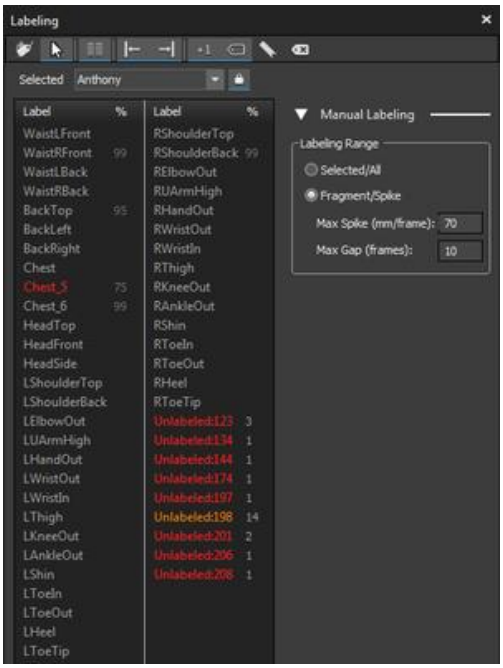

Figura A. 7."Labeling Pane".

A continuación, hay que mirar los marcos en "Graph View Pane" e inspeccionar los espacios que pueda haber en la trayectoria. La "Tracks View" es una vista simplificada que puede revelar espacios, cambios de marcadores y otros problemas básicos de etiquetado que se pueden solucionar rápidamente fusionando varias trayectorias de marcadores. Cuando se seleccionan varios marcadores, las etiquetas se pueden fusionar usando  y  .

Para cada espacio en los marcos, se debe buscar un marcador sin etiquetar en la ubicación esperada, cerca de la posición del marcador resuelto. Tras esto, hay que volver a asignar la etiqueta de marcador adecuada si existe el marcador sin etiquetar.

Se puede utilizar la función "*Trim Tails"* para recortar ambos extremos de la trayectoria en cada espacio. A continuación, se debería recortar algunos fotogramas donde puedan existir errores de seguimiento. De estar forma se preparan las trayectorias que puedan haber sido "bloqueadas" o "perdidas" para rellenar sus huecos.

Una vez se han detectado los huecos que es necesario rellenar, hay que utilizar la función "*Fill Gaps"* para modelar las trayectorias estimadas de los marcadores "bloqueados" o "perdidos".

MARKER LABELING

En Motive, los marcadores capturados se reconstruyen en coordenadas 3D. Estos marcadores reconstruidos deben ser etiquetados para que Motive pueda distinguir las diferentes reconstrucciones dentro de la captura. Las trayectorias de las reconstrucciones anotadas se pueden exportar individualmente o usarse (resueltas por completo) para rastrear los movimientos de los sujetos objetivo.

Los marcadores asociados con cuerpos rígidos se etiquetan automáticamente mediante el proceso de etiquetado automático. Se debe tener en cuenta que estos marcadores se pueden etiquetar automáticamente tanto durante el modo en vivo (antes de la captura) como en el modo de edición (después de la captura).

Los marcadores individuales también se pueden etiquetar, pero cada marcador debe etiquetarse manualmente en el post-procesamiento mediante los "MarkerSet assets" y el "Labeling Pane". Estas herramientas de etiquetado manuales también se pueden utilizar para corregir cualquier error de etiquetado.

A modo de resumen se podría decir que el etiquetado AUTOMÁTICO, etiqueta automáticamente conjuntos de marcadores de un cuerpo rígido utilizando las definiciones de activo correspondientes. Por otra parte, el etiquetado MANUAL, etiqueta los marcadores de forma manual usando el "Labeling Pane" asignando etiquetas definidas en MarkerSet cuerpo rígido. Para reasignar de forma manual etiquetas que pueden estar dando problemas, se puede usar la función "Quick Labeling Mode".

- ***CÓMO ETIQUETAR DE FORMA MANUAL:***

  Usando este "modo" de funcionamiento, se pueden etiquetar los marcadores con sólo hacer un click sobre ellos. Antes de etiquetar los marcadores, primero hay que establecer el rango de etiquetado. En "Labeling Pane", hay dos tipos de rangos entre los que puedes escoger:

  Por defecto, se usa el ajuste *Fragment/Spike*. Esta configuración, etiqueta los rangos de trayectoria que están limitados por huecos y picos, lo que es un comportamiento típico de errores de etiquetado, Figura A. 8.

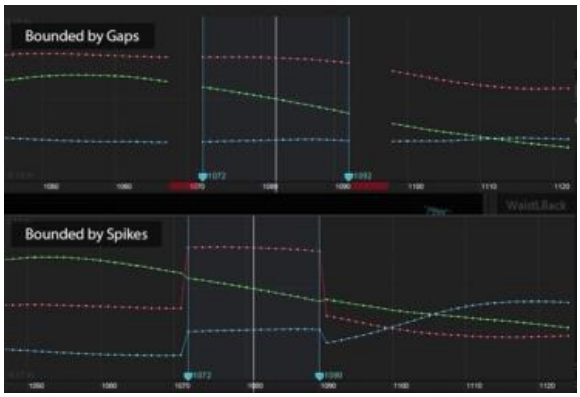

Figura A. 8. Errores típicos de etiquetado "Gaps" y "Spikes"

La configuración *Selected/All,* Figura A. 9, asignan etiquetas tanto a una trayectoria completa como a rangos de marcos específicos seleccionados de la línea de tiempo.

Se debe escoger la configuración que encaje mejor con las trayectorias de las tomas. Tras escoger la configuración deseada, hay que ir mirando toda la captura y detectando los huecos que haya. Cuando se detecte un hueco, simplemente hay que activar la función "Quick Labeling Mode" y pinchar sobre el marcador que no esté etiquetado. Hay que repetir este proceso con todos los marcadores que estén "perdidos".

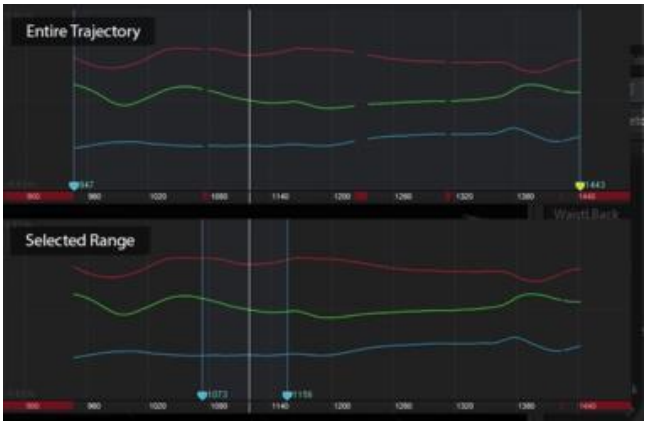

Figura A. 9. Configuración "Selected/All".

A.3 TRANSMISIÓN DE DATOS

Motive exporta datos de seguimiento 3D reconstruidos en varios formatos de archivo. Para exportar datos de seguimiento, hay que seleccionar la toma que se desee exportar y abrir la ventana de diálogo de exportación, a la que se puede acceder desde *File → Export Tracking Data.* Otra forma sería, haciendo click en el botón derecho en la toma que se desea exportar y *Take → Export Tracking data* del Data Mangement Pane, Figura A. 10.

Desde la ventana de diálogo de exportación se pueden configurar la velocidad de fotogramas, la escala de medición y el rango de fotogramas de los datos exportados. Los rangos

de fotogramas también se pueden especificar seleccionando un rango de fotogramas en "Graph View Pane" antes de exportar un archivo.

En la ventana de diálogo de exportación, las opciones de exportación correspondientes están disponibles para cada formato de archivo.

Motive ofrece múltiples opciones para transmitir datos de seguimiento (en tiempo real) a aplicaciones externas. Las aplicaciones comunes de captura de movimiento se basan en el seguimiento en tiempo real, y el sistema OptiTrack está diseñado para entregar datos con una latencia extremadamente baja.

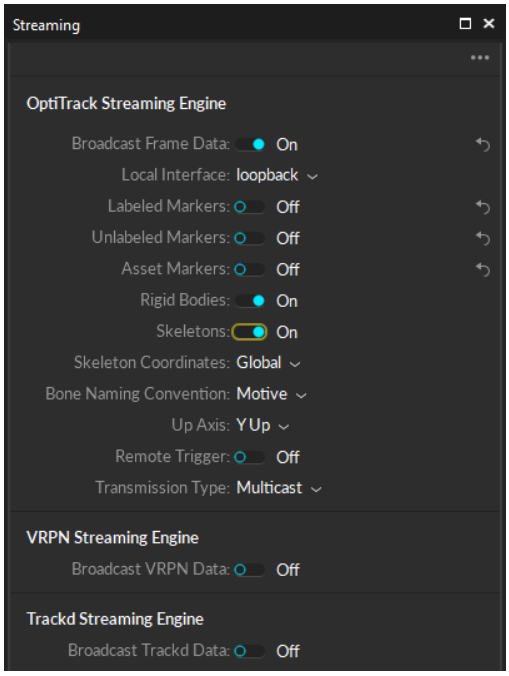

Figura A. 10. "Data Management Pane".

A continuación, se va a describir la configuración de Motive para la transmisión de datos a través de una red de servidor seleccionada.

STREAMING IN MOTIVE

En primer lugar, hay que abrir el "*Data Streaming Pane*" de Motive, al cual se puede acceder haciendo click en el icono  de la barra principal de herramientas. A continuación, hay que seleccionar la dirección de interfaz de la red para la transmisión de datos. Es necesario seleccionar los tipos de datos deseados para transmitir en las opciones de transmisión.

Tras esto, hay que marcar "*Broadcast Frame Data*" que hay en la parte superior del *"Data Streaming Pane*". Después hay que configurar los ajustes de transmisión e indicar la dirección IP correspondiente a la que se desea enviar los datos.

Se pueden enviar tanto datos en tiempo real como capturas de vídeo que se hayan realizado anteriormente.

STREAMING IP ADDRESS

Es importante seleccionar el adaptador de red (interfaz, dirección IP) para la transmisión de datos. La mayoría de las PC Motive Host tienen varios adaptadores de red: uno para la red de la cámara y uno (o más) para la red de área local (LAN)

Motive solo transmite datos a través del adaptador (interfaz) seleccionado. Hay que seleccionar la interfaz deseada utilizando el "Data Streaming Pane" en Motive. La interfaz puede estar en una red de área local (LAN) o en el ordenador (localhost, local loopback).

Si tanto el servidor (Motive) como la aplicación a la que se desea enviar los datos se ejecutan en el mismo ordenador, hay que configurar la interfaz de red en la dirección de bucle de retorno local.

Cuando se transmiten los datos a través de una LAN, hay que seleccionar la dirección IP del adaptador de red conectado a la LAN. Esta es la misma dirección que utiliza la aplicación que recibe los datos para conectarse a Motive.

ANEXO B: CONFIGURACIÓN DE LA RASPBERRY PI

Este anexo incluye una explicación más detallada del proceso de configuración de la Raspberry Pi y del resto de elementos necesarios en la comunicación Bluetooth empleada en este proyecto como vía de transmisión de los datos del ordenador de rastreo al vehículo.

En la Sección B.1 se describe de forma minuciosa el proceso de configuración de la Raspberry Pi desde cero, mientras que en la Sección B.2 se explica paso a paso cómo llevar a cabo la configuración de los módulos Bluetooth empleados en la comunicación y en último lugar en la Sección B.3 se muestra la conexión final entre los módulos Bluetooth del vehículo y el ordenador. Al igual que con el Anexo A, se ha llevado a cabo la redacción de este Anexo porque se considera en un futuro, puede servir de ejemplo o ayuda para la consecución de proyectos similares.

B.1 CONFIGURACIÓN DE LA RASPBERRY PI

Para poder empezar con la configuración de la Raspberry Pi desde cero, es necesario en primer lugar cargar Raspbian (sistema operativo recomendado para Raspberry Pi y basado en una distribución de GNU/Linux llamada Debian) en la tarjeta microSD que se introducirá en la Raspberry.

En este proyecto se ha empleado Matlab-Simulink para cargar Raspbian en la Raspberry. Los pasos que se han seguido han sido los siguientes:

Para comenzar, desde Matlab pinchar en el símbolo de *Add-Ons*. En la ventana que se abre tras pinchar en "Get Add-Ons", escribir en el buscador: "Simulink for Raspberry" y "Matlab for Raspberry". A continuación, instalar ambos: "MATLAB Support Package for Raspbery Pi Hardware" y "Simulink" Support Package for Raspbery Pi Hardware"

Tras instalarlos, es necesario pinchar de nuevo en Add-Ons, pero esta vez se debe abrir "Manage Add-Ons". En la ventana que se abre, aparecen los elementos instalados. Hay que pinchar en "Simulink Support Package for Raspberry Pi Hardware" en el símbolo de los tres "puntitos" y luego "Setup".

A continuación, se debe elegir el modelo de Raspberry correspondiente, Figura B. 1, pinchar en "Next" y en la siguiente ventana seleccionar la opción "Setup hardware with MathWorks Raspbian image".

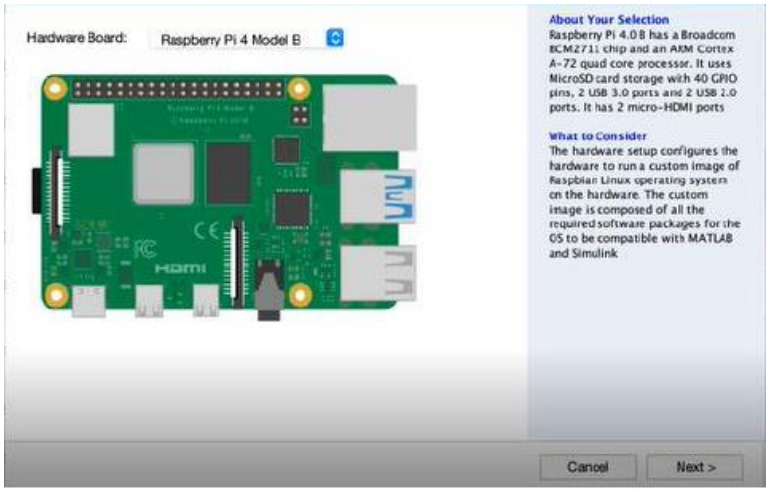

Figura B. 1. Selección del tipo de Raspberry.

En la ventana siguiente se muestran dos opciones de descarga para la imagen de Raspbian. En este proyecto se ha escogido la opción que no contiene Deep Learning, puesto que no era necesaria. Tras descargarla, la imagen aparece en una carpeta .zip en la carpeta de "Descargas" de nuestro PC. En la siguiente ventana hay que seleccionar la imagen y darle a "Validate". Una vez validada la imagen, esta aparece con un tick.

En la siguiente ventana, hay que escoger la opción de "Connect directly to host Computer" y elegir el puerto al que esté conectada la microSD. Una vez finalizado el proceso de carga de Raspbian en la microSD, aparece un mensaje indicando la finalización del proceso.

Una vez cargado el sistema operativo en la Raspberry, se lleva cabo la configuración. Para ello es necesario conocer los componentes de la Raspberry Pi, Figura B. 2.

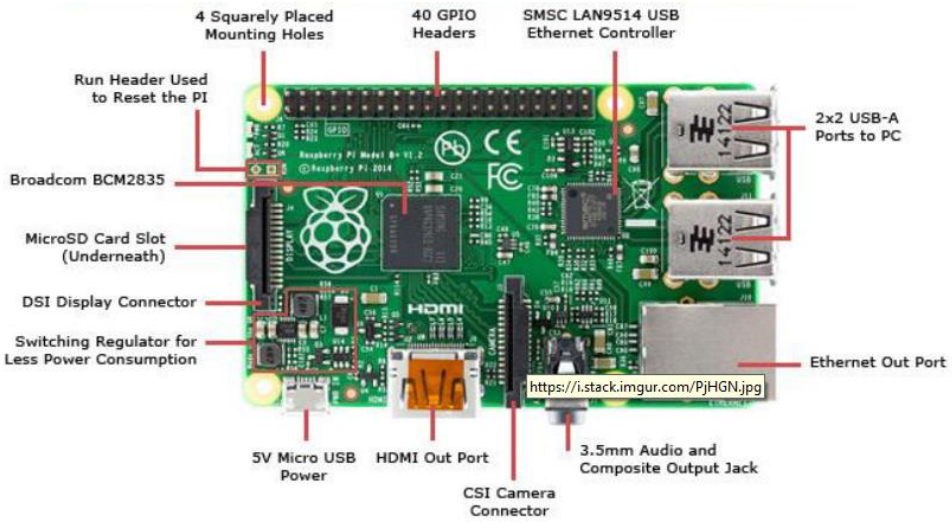

Figura B. 2. Componentes de la Raspberry Pi 3B+.

Al iniciar la Raspberry Pi por primera vez, aparece una aplicación que va guiando al usuario para poder llevar a cabo la configuración inicial. Al comenzar, es necesario configurar y establecer el país, idioma, uso horario y el usuario y contraseña de la Raspberry Pi. Por último, se debe conectar la Raspberry a una red Wifi, la misma a la que esté conectado el ordenador.

B.2 CONFIGURACIÓN DE LOS MÓDULOS BLUETOOTH HC05

En esta sección se describe el proceso a seguir para poder realizar la configuración de los dos módulos bluetooth HC05 empleados en el proyecto.

Para usar ambos módulos bluetooth, primero es necesario configurarlos como *master* y *slave*. Un *slave* solo puede conectarse a un único *master*. Cada dispositivo tiene una dirección IP y un nombre para identificarlo. Para configurarlos es necesaria una placa de Arduino, Figura B. 3.

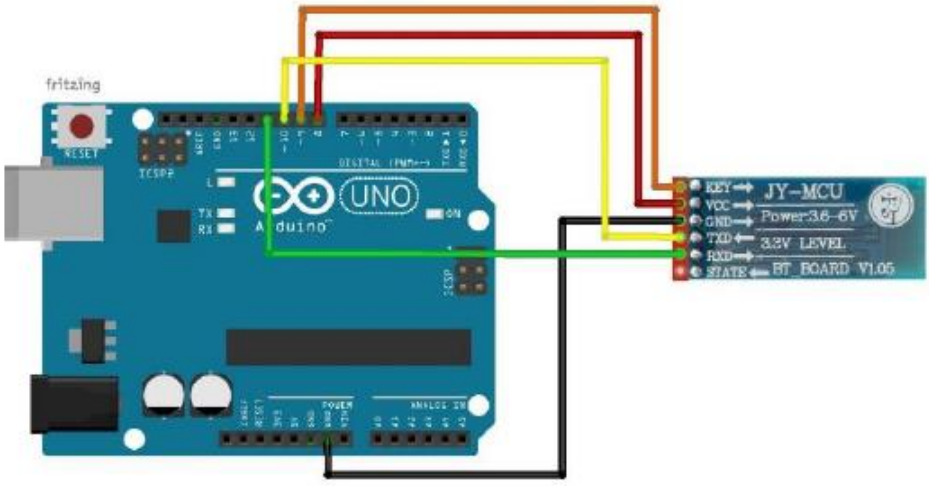

Figura B. 3. Conexión del módulo Bluetooth a la placa de Arduino.

Para poder configurarlos es necesario hacerlo desde el modo AT. Para ello hay que mantener pulsado el *push-button*, que se encuentra en el módulo HC05 en la parte contraria a la que aparecen los nombres de los pines, mientras se conecta el módulo a VCC.

Los leds deberían encenderse con una frecuencia de 2s, si es así hemos accedido correctamente al modo AT.

Tras esto en el software de Arduino, en la pestaña de Herramientas, hay que seleccionar el puerto COM y también el tipo de placa que estamos usando, Figura B. 4.

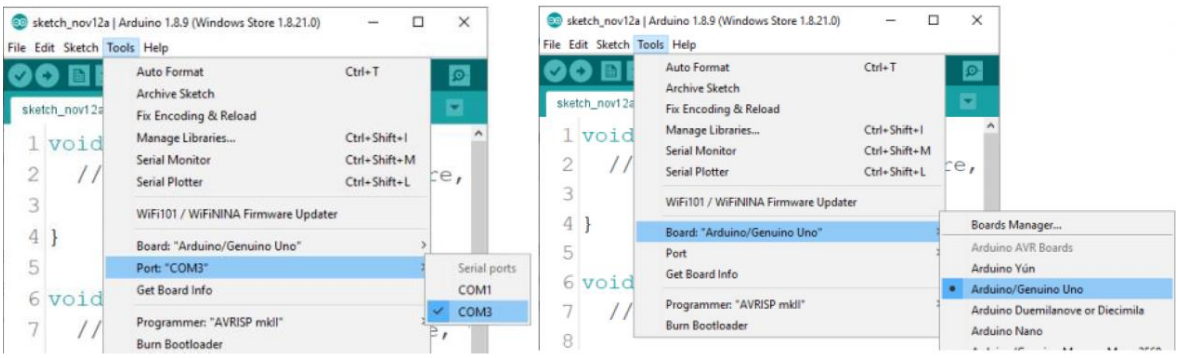

Figura B. 4. Selección del puerto COM y del tipo de placa.

Tras esto, hay que acceder al MONITOR SERIE y usar los comandos necesarios para configurar cada módulo como *slave* y *master*. En la Figura B. 5, se muestra una lista de comandos que podrían resultar útiles.

Command	Description
AT	Test the serial communication. Returns "OK"
AT+VERSION?	Returns the firmware version "VERSION:3.0-20170601"
AT+ORGL	Reset to original configuration
AT+RESET	Restart the module
AT+NAME?	Returns the name of the module
AT+NAME=newName	Rename the module to "newName".
AT+ADDR?	Returns the MAC-Address as Hexadecimal values
AT+STATE?	Returns the current state: INITIALIZED READY PAIRABLE PAIRED INQUIRING CONNECTING CONNECTED DISCONNECTED UNKNOWN
AT+ROLE?	Returns the current ROLE: 0 = slave mode 1 = master mode 2 = slave echo mode

AT+ROLE=x	Switch the current Role: 0 = slave mode 1 = master mode 2 = slave echo mode
AT+CMODE?	Returns the current mode. For CMODE to work, the module has to be in Master Modus (see AT+ROLE command). 0 = Manual connection. The device address must be provided 1 = Auto connect 2 = Slave loop Mode
AT+PSWD?	Returns the current PIN
AT+PSWD=xxxx	Set the PIN-Code for pairing.
AT+PAIR=address,timeout	Pair with other Bluetooth device. the address format is: 1234, 56, abcdef. The timeout - value is in seconds

Figura B. 5. Lista de comandos para la configuración de los módulos Bluetooth.

Si al introducir AT en el MONITOR SERIE, salta un Ok, significa que está funcionando y que se puede continuar con la configuración. Es importante que ambos módulos tengan los mismos *baudios*, sino la configuración no se realizará de forma correcta.

Inicialmente, se debe obtener la dirección de ambos módulos usando el comando: *AT+ADDR* puesto que esta se usará más adelante.

❖ CONFIGURACIÓN DE LOS MÓDULOS BLUETOOTH HC05 COMO *SLAVE*:

Para configurar el módulo bluetooth como *slave* hay que utilizar los siguientes comandos en el MONITOR SERIE:

- AT
- AT+ROLE = 0
- AT+CMODE=0
- AT+UART=57600,1,0
- AT+ ADDR
- AT+BIND = *address* del *master* (separada por ',' no por ':')

Si todo ha salido bien los LEDs deberían "parpadear dos veces" y luego quedarse 1 segundo apagados. Si en cambio parpadean todo el rato, significa que algo no ha funcionado.

❖ **CONFIGURACIÓN DE LOS MÓDULOS BLUETOOTH HC05 COMO *MASTER*:**

Para configurar el módulo bluetooth como Slave hay que utilizar los siguientes comandos en el MONITOR SERIE:

- AT
- AT+ROLE = 1
- AT+CMODE=0
- AT+UART=57600,1,0
- AT+ ADDR
- AT+BIND = *address* del *slave* (separada por ',' no por ':')

Si todo ha salido bien los LEDs deberían "parpadear dos veces" y luego quedarse 1 segundo apagados. Si en cambio parpadean todo el rato, significa que algo no ha funcionado.

B.3 COMUNICACIÓN ENTRE PC Y RASPBERRY PI

Para llevar a cabo la comunicación entre el ordenador y la Raspberry Pi, es necesario emplear un conversor USB UART.

En primer lugar, se deben que conectar ambos módulos bluetooth al ordenador y a la Raspberry. La conexión del módulo *slave* con la Raspberry es la mostrada en la Figura B. 6.

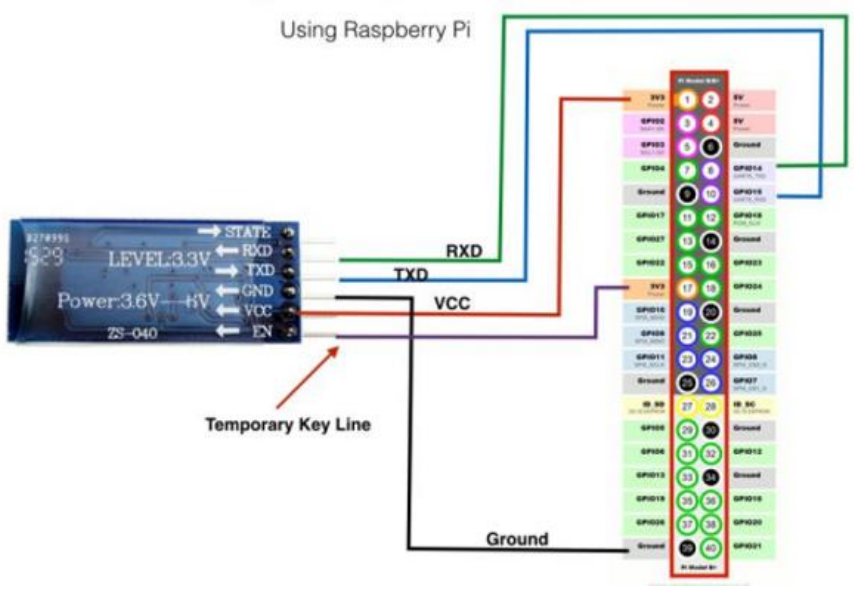

Figura B. 6. Conexión del módulo slave con la Raspberry Pi.

Hay que conectar *RXD* del módulo bluetooth con el *TXD* de la Raspberry y el *TXD* del módulo bluetooth con el *RXD* de la Raspberry tal y como se muestra en la Figura B. 6.

Por otra parte, la conexión del módulo *master* con el conversor USB UART es la que se indica en la Figura B. 7. Es importante conectar VCC a 3,3V.

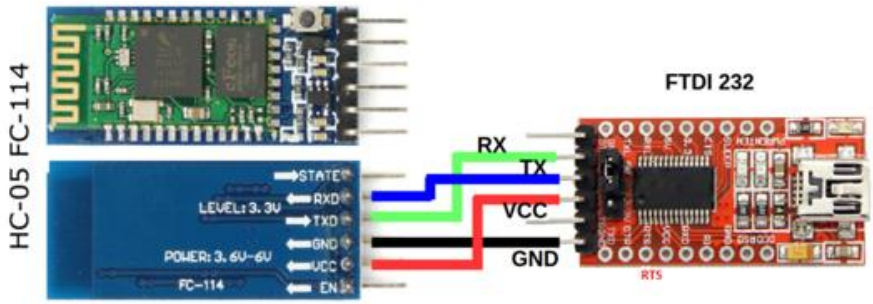

Figura B. 7. Conexión del módulo master al convertidor USB UART.

Si están configurados correctamente, al conectar ambos módulos, los LEDs deberían parpadear más o menos a la vez de la siguiente forma: Dos pulsos encendidos y 2 segundos (aproximadamente) apagados (repitiéndose todo el rato).

ANEXO C: CONTRIBUCIÓN A LOS OBJETIVOS DE LA ONU PARA EL DESARROLLO SOSTENIBLE

Se va a incluir en este anexo una reflexión sobre el alineamiento de este trabajo con los Objetivos de Desarrollo Sostenible (ODS) de Naciones Unidas, los cuales son un proyecto clave para acabar con la desigualdad mundial que encontramos en la actualidad.

Se va a explicar brevemente cuáles son algunos de estos Objetivos de Desarrollo Sostenible, concretamente el 9 y el 11, y la relación de estos con este proyecto de fin de grado.

Los Objetivos de Desarrollo Sostenible (ODS) constituyen un llamamiento universal a la acción para poner fin a la pobreza, proteger el planeta y mejorar las vidas y las perspectivas de las personas en todo el mundo. En 2015, todos los Estados Miembros de las Naciones Unidas aprobaron 17 Objetivos como parte de la Agenda 2030 para el Desarrollo Sostenible, en la cual se establece un plan para alcanzar los Objetivos en 15 años [36].

Entre estos ODS propuestos por la ONU podemos relacionar este trabajo con los objetivos número 9 y 11, más cercanos al desarrollo industrial y la sostenibilidad en las nuevas tecnologías.

El objetivo número 9 sobre industria, innovación e infraestructuras trata de conseguir una industrialización inclusiva y sostenible que, junto con la innovación y la infraestructura, pueden dar rienda suelta a las fuerzas económicas dinámicas y competitivas que generan el empleo y los ingresos. Estas desempeñan un papel clave a la hora de introducir y promover nuevas tecnologías, facilitar el comercio internacional y permitir el uso eficiente de los recursos [37].

En el contexto del trabajo, el desarrollo de sistemas de navegación como el que se ha realizado en este proyecto está muy enfocado a la innovación y creación de nuevas infraestructuras dentro de la industria. Sistemas como este permiten la instalación de robots inteligentes en cadenas de montaje, para facilitar ciertas tareas concretas, o carritos autónomos que transporten materiales o productos en grandes fábricas.

En cuanto al objetivo 11 sobre ciudades y comunidades sostenibles, el mundo cada vez está más urbanizado y desde 2007 más de la mitad de la población mundial ha estado viviendo en ciudades. Además, se espera que dicha cantidad aumente hasta el 60% para 2030 [38].

Las ciudades y las áreas metropolitanas son centros neurálgicos del crecimiento económico, ya que contribuyen al 60% aproximadamente del PIB mundial. Sin embargo,

también representan alrededor del 70% de las emisiones de carbono mundiales y más del 60% del uso de recursos [38].

La sostenibilidad en las ciudades va a llegar muy de la mano del crecimiento del internet de las cosas (IoT), que mediante dispositivos conectados a internet permitirá gestionar de una forma mucho más eficiente cada uno de los procesos dentro de estas.

Además, el crecimiento del vehículo autónomo en las ciudades está directamente relacionado con el motor eléctrico, mucho más respetuoso con el medio ambiente que el motor de combustión usado en los coches convencionales. En este proyecto se ha diseñado el sistema que permita a un pequeño robot moverse de forma autónoma y que podría simular en pequeña escala el sistema de navegación de los vehículos autónomos.

A modo de conclusión, es muy positivo que todos los proyectos universitarios o de carácter académico o de investigación, tengan en cuenta estos diecisiete Objetivos de Desarrollo Sostenible, para conseguir un mundo con menos desigualdad y más sostenible, teniendo siempre en cuenta la innovación y el desarrollo industrial.