COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

NAVEGACIÓN DE UN VEHÍCULO EQUILIBRISTA MEDIANTE EL SEGUIMIENTO DE PARED O CÁMARAS EXTERNAS

Autor: María Ángeles González Castro

Director: Juan Luis Zamora Macho

Madrid

Septiembre 2022

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

Navegación de un vehículo equilibrista mediante el seguimiento de pared o cámaras externas

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

curso académico 2022/23 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.: María Ángeles González Castro Fecha: 27 / 09 / 2022

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Juan Luis Zamora Macho Fecha: 27 / 09 / 2022

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

NAVEGACIÓN DE UN VEHÍCULO EQUILIBRISTA MEDIANTE EL SEGUIMIENTO DE PARED O CÁMARAS EXTERNAS

Autor: María Ángeles González Castro

Director: Juan Luis Zamora Macho

Madrid

Septiembre 2022

Agradecimientos

A mi familia, que ha sido mi apoyo incondicional desde que entré en ICAI. Por ofrecerme realizar mis estudios en esta Universidad, lejos de casa. Por permitirme caer y darme ánimos cuando se me hacía cuesta arriba.

A mis compañeros y amigos, por ser un pilar fundamental a lo largo de toda la carrera.

A mi director, Juan Luis, por su infinita paciencia con este proyecto y todas las facilidades que me ha dado.

TÍTULO DEL TFG

Autor: González Castro, María Ángeles.
Director: Zamora Macho, Juan Luis.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

La presente memoria describe el diseño de un control de estabilización del ángulo de cabeceo y el diseño de un control de navegación mediante el seguimiento de pared para un vehículo equilibrista basado en una Raspberry Pi. La estrategia de control utilizada fue el control por realimentación de estados.

Palabras clave: equilibrista, control, EKF, segway

1. Introducción

Los vehículos equilibristas son aquellos que han de mantener el equilibrio e incorporan un sistema de control para dicho fin. El primer vehículo equilibrista fue el Segway Personal Transporter, inventado por Dean Kamen en 2001. [1]. Además de vehículos de transporte que mejoran las capacidades de sus usuarios, al permitirles desplazarse más rápido o cargar más peso, existen variantes del segway dirigidas a personas con capacidades reducidas de movimiento y facilitan su día a día. Otro ámbito en el que se utilizan los vehículos equilibristas es el de la docencia, puesto que son una plataforma a través de la que se puede estudiar distintas estrategias de control y es muy ilustrativa. En este trabajo se utilizará un robot que pertenece al grupo de TWABR (Two Wheeled Automatic Balancing Robot) con el fin de poner a prueba distintos controles sobre un vehículo equilibrista destinado a su uso en los laboratorios de la universidad.

2. Definición del Proyecto

Este trabajo trata de navegar un vehículo con auto balanceo. Para ello primero se ha de identificar el modelo dinámico del vehículo. Posteriormente, se diseñan de dos controles diferentes. Un primer control ha de equilibrar el vehículo manteniéndolo en su posición vertical y, una vez conseguido el primer objetivo, se diseña un segundo control de navegación mediante el seguimiento de pared. El objetivo final es la realización de una vuelta completa en un circuito cerrado por el vehículo con el control de seguimiento de pared, en la cual el vehículo mostrará su estabilización en rampas de subida y de bajada y el seguimiento de pared tanto en tramos rectos como curvos.

3. Descripción del sistema

El vehículo cuenta con dos motores de corriente continua cuya tensión administrada es el actuador y la salida del control. Para obtener el valor del

actuador, se utiliza información de las variables de estado de cada control. En el caso del control de estabilización, esas variables son el ángulo de cabeceo, la velocidad angular de cabeceo y la velocidad de avance, mientras que el seguimiento de pared son la distancia a la pared, el ángulo de orientación con esta y la velocidad de guiñada del vehículo. Se calculará el valor de tensión que se ha de aplicar a los motores en cada instante como una combinación lineal del valor de las variables de estado y del error del control. Puesto que el ángulo de cabeceo del primer control no se puede medir con sensores, se obtendrá una estimación de este con un estimador de estados. A lo largo del proyecto se alternarán dos estimadores de estado: un filtro complementario y un filtro extendido de Kalman.

4. Resultados

Los resultados del trabajo han sido satisfactorios. Se ha cumplido con los objetivos de estabilización del vehículo y el control de navegación mediante el seguimiento de pared ha sido exitoso, ya que el vehículo es capaz de trazar un circuito cerrado sin chocar contra las paredes. En la Figura 1 se muestran los resultados de un ensayo para el control de estabilización. Las oscilaciones del ángulo de cabeceo son menores a cinco grados.

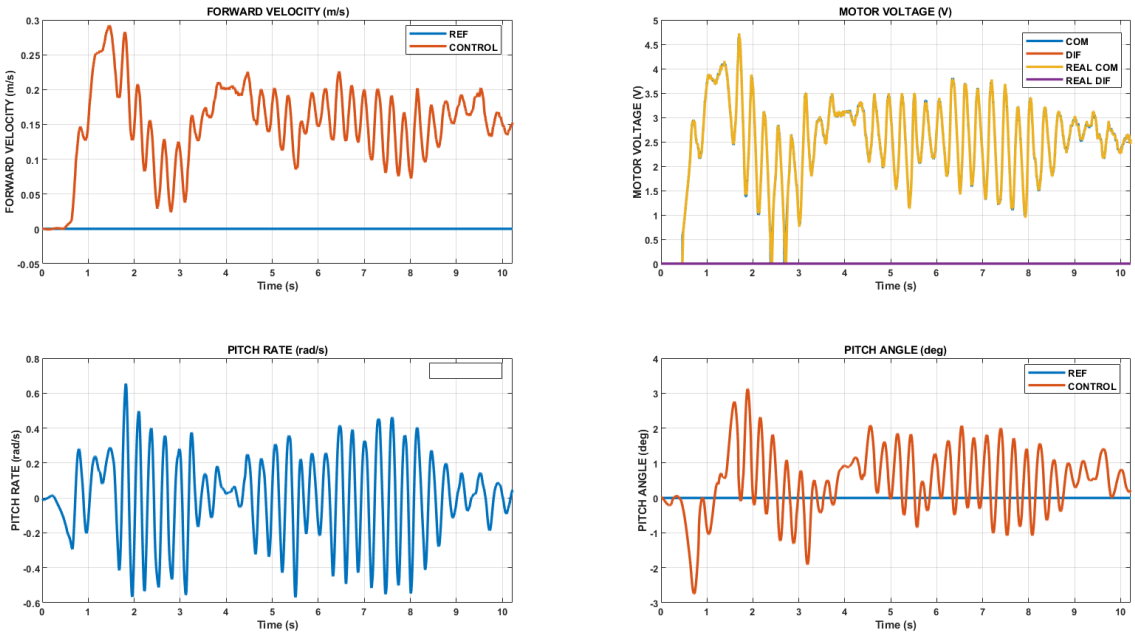


Figura 1: resultado del control de estabilización del vehículo

Para la realización del trazado del circuito cerrado se decidió utilizar un mando emisor de radiofrecuencias para controlar la velocidad del vehículo. Dado que el vehículo está destinado a usarse en el laboratorio de Control Digital, el uso del mando permitirá realizar competiciones de velocidad entre los alumnos, en las que entrará en juego la estabilidad del control, especialmente en los cambios de tramos rectos a rampas y viceversa.

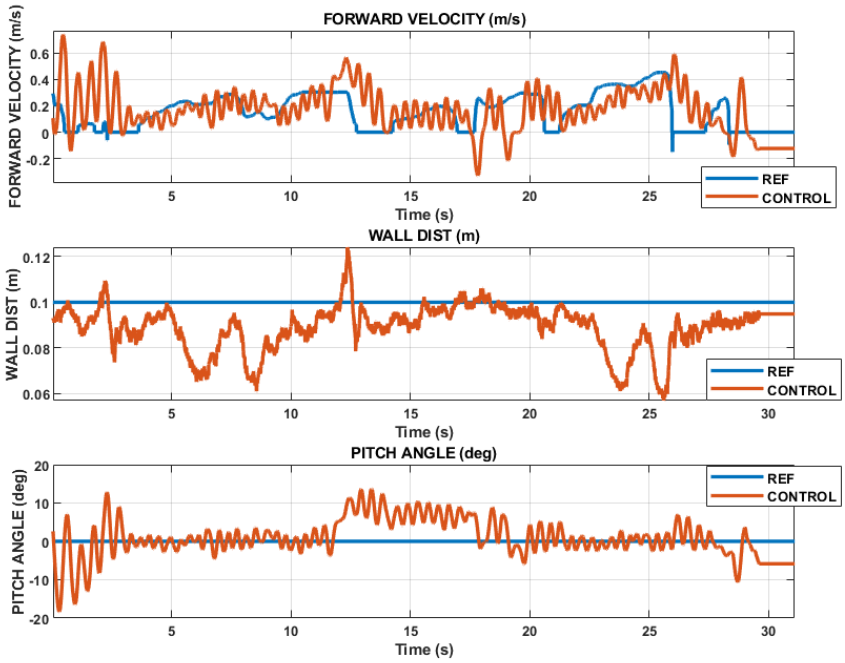


Figura 2: resultados del trazado del circuito cerrado

Tras la realización del proyecto, se considera que el control por realimentación de estados es indicado para estabilizar y navegar el vehículo equilibrista.

5. Referencias

[1] Segway-Ninebot, «La historia de Segway-Ninebot,» [En línea]. Available: https://es-es.segway.com/about-the-brand.

[2] Jiménez, F., Ruge, I., & Jiménez, A. (2020). Modeling and Control of a Two Wheeled Self-Balancing Robot: a didactic platform for control engineering education. *LACCEI Inc.*

TÍTULO DEL TFG (EN INGLÉS)

Author: González Castro, María Ángeles.
Supervisor: Zamora Macho, Juan Luis.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

This report describes the design of a pitch angle stabilization control and the design of a wall-tracking navigation control for a balancing vehicle based on a Raspberry Pi. The control strategy used was state feedback control.

Keywords: Self-balancing, control, EK, segway

1. Introduction

Self-balancing vehicles are those who must maintain balance and incorporate a control system for this purpose. The first balancing vehicle was the Segway Personal Transporter, invented by Dean Kamen in 2001. [1]. In addition to transport vehicles that improve the capabilities of their users, by allowing them to move faster or carry more weight, there are variants of the segway aimed at people with reduced movement abilities and facilitate their day to day. Another area in which balancing vehicles are used is that of teaching, since they are a platform through which different control strategies can be studied and is very illustrative. In this work, a robot belonging to the TWABR (Two Wheeled Automatic Balancing Robot) group will be used in order to test different controls on a balancing vehicle intended for use in the laboratories of the university.

2. Project definition

This work is about navigating a vehicle with self-balancing. To do this, the dynamic model of the vehicle must first be identified. Subsequently, they are designed from two different controls. A first control must balance the vehicle keeping it in its vertical position and, once the first objective is achieved, a second navigation control is designed through wall tracking. The ultimate goal is to perform a full lap on a closed circuit by the vehicle with the wall tracking control, in which the vehicle will show its stabilization on ramps up and down and wall tracking in both straight and curved sections.

3. Description of the system

The vehicle has two direct current motors whose managed voltage is the actuator and the control output. To obtain the value of the actuator, information from the state variables of each control is used. In the case of the stabilization control, those variables are the pitch angle, the angular pitch speed and the forward speed, while the wall tracking is the distance to the wall, the angle of orientation with it and the

yaw speed of the vehicle. The voltage value to be applied to the motors at each instant shall be calculated as a linear combination of the value of the state variables and the control error. Since the pitch angle of the first control cannot be measured with sensors, an estimate of this will be obtained with a state estimator. Two state estimators will be alternated throughout the project: a complementary filter and an extended Kalman filter.

4. Results

The results of the work have been satisfactory. Vehicle stabilization objectives have been met and navigation control by wall tracking has been successful, as the vehicle is able to trace a closed circuit without hitting walls. In Figure 1 the results of a stabilization control test are shown. Pitch angle oscillations are less than five degrees.

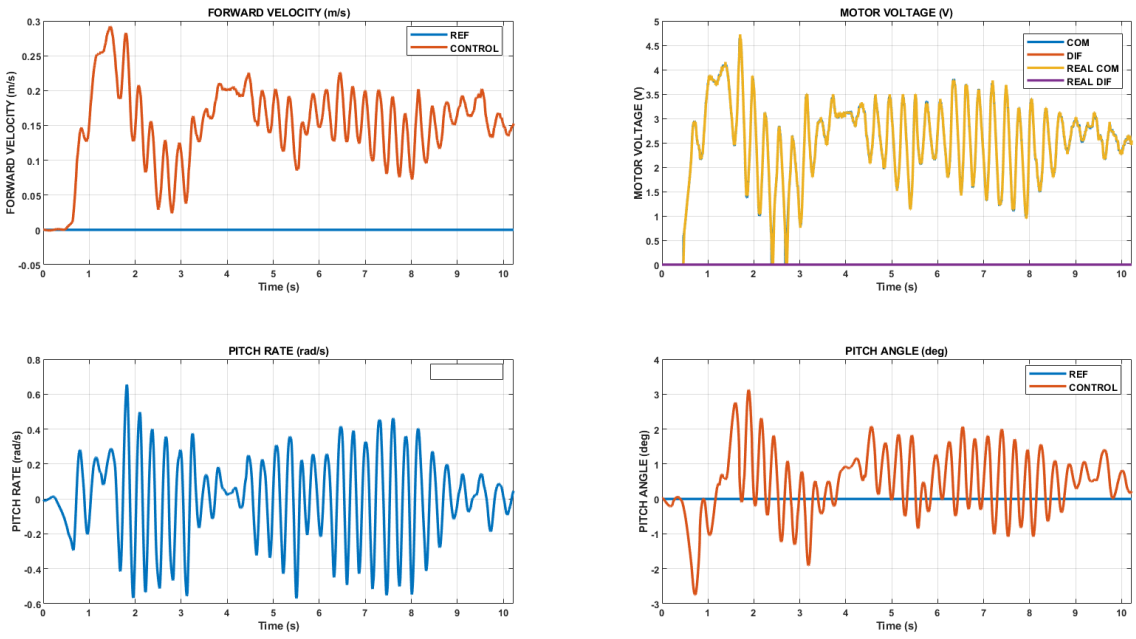


Figure 1: vehicle stabilisation control result

For the realization of the layout of the closed circuit it was decided to use a radio frequency emitting control to control the speed of the vehicle. Since the vehicle is intended for use in the Digital Control laboratory, the use of the controller will allow speed competitions to be held between students, in which the stability of the control

will come into play, especially in the changes from straight sections to ramps and vice versa.

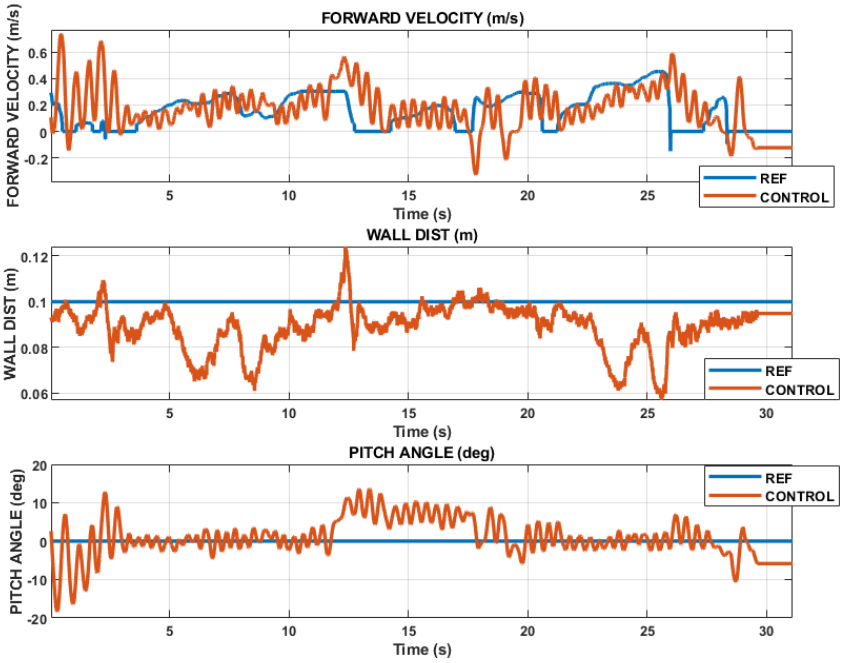


Figure 2: tracking control test in a circuit result

After the completion of the project, it is considered that the control by feedback of states is indicated to stabilize and navigate the balancing vehicle.

5. References

[1] Segway-Ninebot, «La historia de Segway-Ninebot,» [En línea]. Available: https://es-es.segway.com/about-the-brand.

[2] Jiménez, F., Ruge, I., & Jiménez, A. (2020). Modelling and Control of a Two Wheeled Self-Balancing Robot: a didactic platform for control engineering education. *LACCEI Inc.*

Índice de la memoria

Índice de figuras

Siglas

GPIO: General Purpose Input/Output, entrada/salida de propósito general

FC: Filtro Complementario

EKF: Filtro Extendido de Kalman

SBV: Self-balance Vehicle, vehículo auto balanceado

ToF: Time of Flight

1. Introducción

La transformación de la movilidad en las ciudades tiende en los últimos años a apostar por los vehículos eléctricos. Esto se debe, entre otros motivos, a la gran subida de los precios del combustible y a las normativas gubernamentales orientadas a la disminución de la contaminación. Más allá de los coches, vehículos como los patinetes eléctricos han ganado popularidad, inundando las calles. No es de extrañar: no solo son poco contaminantes y económicos, sino que son una forma divertida de evitar los atascos, ya que pueden hacerte llegar a tu destino en hora punta antes que un coche. Mucho antes de la popularización masiva del patinete eléctrico, al comienzo de la década de los 2000, Dean Kamen supo de la necesidad de un vehículo adaptado a las grandes ciudades y tuvo una idea revolucionaria: inventó el primer vehículo con auto balanceo del mercado, el Segway. [1] Fue lanzado al mercado en 2002. Se trataba de un vehículo eléctrico compuesto por una plataforma con dos ruedas y un manillar. Lo novedoso de este vehículo era su funcionamiento: para desplazarse hacia adelante o hacia atrás, el ocupante había de inclinar su cuerpo en dicha orientación. Se trataba de vehículos con alta autonomía llegando a alcanzar los 20 km/h. Las expectativas con respecto a este producto fueron altas, se predijo un número de ventas mayor a 60.000 unidades durante su primer año en el mercado y fue alabado por figuras como Steve Jobs, quien predijo que el Segway sería “tan importante como un ordenador”. Sin embargo, su elevado precio impidió que se cumpliesen tales expectativas.

A pesar del fracaso de ventas que tuvo Segway, este fue el precursor del resto de vehículos con auto balanceo o equilibristas. Posteriormente salieron al mercado multitud de vehículos derivados: algunos más ligeros, como los *hoverboards*, que no cuentan con manillar; otros orientados a las personas que tienen capacidades reducidas de movimiento, a modo de sustitutos de sillas de ruedas. Debido al éxito de ventas de los patinetes eléctricos y a su continua integración en las ciudades, se puede prever un futuro con formas de movilidad diferentes, de propulsión eléctrica y ligeros. Se trata, por tanto, del momento idóneo para la introducción de nuevos vehículos, entre ellos, los equilibristas. Por ello resulta de interés investigar la teoría de control que llevan detrás los vehículos equilibristas, en este Trabajo de Fin de Grado expuesta.

El funcionamiento de los vehículos equilibristas está basado en el problema de la teoría clásica de control del péndulo invertido unido a un carro. En él, una masa unida por una varilla rígida a un carro ha de mantenerse en posición vertical. La masa en el punto más alto encuentra un punto estable, aunque con una pequeña perturbación pierde el equilibrio y su estabilidad. Por ello precisa de un sistema de control que mantenga la estabilidad ante perturbaciones, que se encargue de ejercer la fuerza necesaria al carro para mantener a la masa en su posición de equilibrio. A pesar de la popularidad y sencillez de los controles PID, estos no resultan adecuados para plantas inestables como la del vehículo equilibrista, por lo que se precisa de una estrategia de control de realimentación de estados.

Con todo esto, teniendo en cuenta las posibilidades que ofrece la tecnología del auto balanceo, este Trabajo de Fin de Grado está destinado a estudiar el Control Digital que se aplica a estos vehículos, poniéndolo en práctica en un vehículo destinado a la docencia basado en una Raspberry Pi. Se busca además facilitar la implantación del vehículo a utilizar en este trabajo para el laboratorio de Control Digital en el curso posterior, y sustituir al lego NXT utilizado hasta este momento.

1.1 Objetivos del proyecto

Se establecen tres objetivos para este Trabajo de Fin de Grado, en orden creciente de dificultad.

1. **Estabilización del vehículo.** Se trata de un primer control por realimentación de estados cuya función sea la de mantener el vehículo en vertical impidiendo que las perturbaciones provoquen su caída al suelo. No se controla por tanto el recorrido que el vehículo pueda realizar sobre el suelo.

2. **Control integral de la velocidad de avance.** El regulador anterior permitirá equilibrar el vehículo, sin embargo no mantendrá la velocidad de avance en el valor deseado. Aunque se requiera velocidad nula, el regulador no lo garantiza si existen perturbaciones, como por ejemplo un error de estimación en el ángulo de cabeceo. El segundo objetivo del proyecto será diseñar un control por realimentación de estado que incluya una acción integral para seguir de forma precisa el valor de referencia de la velocidad de avance. Esta estrategia de control asegura error de seguimiento nulo en régimen permanente, incluso en presencia de un error de estimación en el ángulo de cabeceo.

3. **Control de seguimiento de pared.** El fin último del proyecto consiste en la realización de una vuelta completa a un circuito cerrado consistente en dos tramos rectos y dos curvas de 180 grados. Para ello se diseñará un control de seguimiento de pared que permita seguir una referencia de distancia concreta, lo cual será válido tanto en paredes rectas como en el trazado de curvas.

2. Estado del arte

Se van a diferenciar los vehículos auto balanceados según su función en vehículos para el transporte y vehículos para la docencia.

1.2 Vehículos para el transporte

Segway

Esta empresa mundialmente conocida fue fundada por el inventor estadounidense Dean Kamen. Tras ver a una persona en silla de ruedas intentar subir con dificultad una acera, Dean cayo en la cuenta de que el mundo estaba creado para personas que podían mantener el equilibrio. Para dar independencia a las personas que carecían de esa capacidad, creó el Independence IBOT™ Mobility System en 1999, un dispositivo

Figura 3: Dean Kamen sobre un Segway acompañado de un iBot en el escenario. CC por redjar en Flickr https://flickr.com/photos/87434398@N00/113100422

de movilidad auto equilibrado capaz de subir escaleras, circular por superficies irregulares y ponerse de pie sobre dos ruedas. [1] Inspirado en este primer vehículo y con la intención de extender su tecnología a personas con movilidad total, Dean creó el Segway PT. Este vehículo se compone de una plataforma sobre dos ruedas que cuenta con un manillar. El ocupante se ha de posicionar sobre la plataforma, de forma que la manera de conducir el dispositivo es la de inclinar el cuerpo hacia la dirección en la que se desee moverse. El segway, gracias a los microprocesadores, sensores de inclinación y giróscopos adopta dicha dirección.

El segway llegó al mercado en 2002. Pese a contar con altas expectativas de ventas, factores como el precio, peso, habilidad para su conducción o exposición a los elementos de las vías públicas, no tuvo el éxito esperado. En un intento se salvar la marca, en 2015 Segway se fusionó con la empresa china Ninebot. Tras el paso de los años otras compañías tomaron la idea de la micro-movilidad urbana, como los patinetes y bicicletas eléctricas, que al ser más ligeros y asequibles, opacaron al Segway en ventas. Finalmente, casi dos décadas después de su invención, en 2020 se anunció el cese de la producción de vehículos segway debido a su reducido número de ventas. A pesar de no ser la revolución anunciada, el Segway fue el precursor de la avalancha de nuevos dispositivos con motor, así como el ejemplo más popular del problema del péndulo invertido y de implementación del control por realimentación de estados.

Omeo Technology

Con origen en Nueva Zelanda, Omeo es un vehículo de movilidad personal manos libres que utiliza la inclinación del cuerpo como el mando que dirige la dirección de su movimiento. A diferencia del segway, el ocupante va sentado. Está diseñado para utilizarse como silla de ruedas adicional, vehículo para aquellas personas que tienen dificultades para moverse o aquellas que prefieren no caminar. Se le puede añadir un kit complementario todoterreno que amplía la gama de terrenos por las que se puede utilizar este vehículo. [2]

Vehículos de tipo hoverboard

El *hoverboard* es básicamente un segway sin manillar. Fue lanzado al público en 2013. Su principio de funcionamiento es el del péndulo invertido, al igual que el segway. Sin embargo, son vehículos más pequeños y menos pesados. El mecanismo de giro es diferente, ya que en el segway el giro se realiza con el manillar, mientras que en el caso de los hoverboards el giro se realiza con los pies, y dependiendo de la inclinación de cada pie gira hacia un lado o hacia el otro. La ausencia de manillar también implica la necesidad de práctica hasta alcanzar el equilibrio necesario para el uso de estas plataformas con ruedas.

Figura 4: vehículo hoverboard. CC0 por Nulall, accesible en https://upload.wikimedia.org/wikipedia/commons/4/4a/Hoverboard_2.jpg

1.3 Vehículos para la docencia

Existen muchos proyectos DIY que se pueden encontrar en internet para construir tu propio robot equilibrista. Muchos proporcionan el código de control ya hecho para iniciar a todo aquel que quiera en el mundo STEM, pero son de código abierto para poder ofrecer la posibilidad de configurar tu propio código. A continuación, se muestran algunos de ellos.

Lego Mindstorm NXT

Es un set programable de la marca LEGO que salió al mercado en 2006. Con él, se construyó el vehículo hasta ahora utilizado en la universidad como planta del control por realimentación de estados. Este kit de robótica permite construir y programar

robots usando motores, sensores, ruedas y otros componentes. Su elemento principal es el ladrillo programable NXT al que se le pueden conectar hasta cuatro sensores y con él controlar tres motores. Cuenta con un microcontrolador de 32 bits, 256 KB de memoria y 6 KB de RAM

OSOYOO 2WD Balance Car Robot

Figura 5: Vehículo equilibrista NXT de LEGO

Se trata de un kit educacional concebido por ingenieros canadienses que incluye todas las piezas necesarias para construir un robot de auto balanceo, desde el microprocesador hasta los motores y las ruedas. Está basado en una tarjeta compatible con Arduino denominada OSOYOO UNO y es controlable desde una aplicación para Android a través del chip Bluetooth que incorpora. Contiene un código de control preinstalado que es modificable. [3]

ELEGOO Tumbller Self-Balancing Robot

Se trata de un kit compatible con Arduino IDE que ofrece funciones como el control por infrarrojos del vehículo, seguimiento de objetos o la evasión automática de obstáculos. Puede ser controlado desde una la aplicación de la marca ELEGOO.

Figura 6: kit del vehículo equilibrista Tumbller de la marca ELEGOO

3. Software y hardware

Actualmente el vehículo que se utiliza en la asignatura de Control Digital en la universidad es el del kit *Mindstorm* de LEGO. Este kit dejó de recibir soporte de Matlab en 2015. Sin embargo, los recursos para la Raspberry Pi no dejan de actualizarse. [4] Es por ello por lo que se está realizando el cambio del NXT por el vehículo objeto de este Trabajo de Fin de Grado, como ya ha ocurrido en las asignaturas de Regulación Automática y Control Avanzado. La novedad en este proyecto consiste en usarlo a modo vehículo equilibrista para el estudio de controles en representación de estado. El objetivo de este capítulo es el de explicar cómo se conforma el vehículo a nivel software y hardware.

3.1 Hardware

A continuación se enumeran y explican brevemente los componentes físicos del vehículo, de abajo a arriba.

Dos motores DC (EMG30): motores de 12 voltios equipados con encoders y reductor 30:1, que impulsan dos ruedas de 100mm de diámetro. También incluyen un condensador de supresión de ruido estándar en los devanados del motor. [5]

Batería con interruptor: proporciona 5 voltios de hasta 2 amperios a la vez que 12 voltios de hasta 3 amperios. Alimenta a la Raspberry Pi y a los motores a través de driver.

Drivers MD25 (PWM): se trata de un único controlador diseñado para trabajar con los dos motores EMG30. Lee la información de los encoders de los motores para determinar distancias y direcciones a partir de ella. También puede proporcionar la tensión que reciben los motores. Se alimenta a 12 voltios. [5]

Conversor de nivel lógico: permite conectar la Raspberry Pi, que trabaja a 3,3V con el driver MD25, que lo hace a 5V, adaptando la lógica que permite la comunicación bidireccional entre estos dos elementos.

Raspberry Pi 3B+: miniordenador que coordina todas las acciones del vehículo y las comunicaciones con el exterior. Se trata de la tercera generación de la marca. Reúne las medidas de todos los sensores, controla los motores y señaliza el estado del vehículo mediante un led RGB.

Shuttle Click: es un expansor de conectores que permite conectar hasta cuatro click boards en un solo mikroBUS, permitiendo así aumentar el número de sensores que se pueden conectar a la Raspberry Pi.

MikroBUS Shuttle: es una tabla añadida que se conecta al Shuttle click a través de cables planos de 16 pines hembra para aumentar el número de conectores mikroBUS.

Pi-EzConnect: permite conexiones fáciles a cualquier GPIO o cualquier otro pin en una Raspberry Pi, garantizando conexiones seguras y fáciles de usar. En el vehículo s

eutiliza para la conexión de los pulsadores y del LED, además de los cables que comunican la Raspberry al driver MD25.

Pi E Click Shield: es una extensión para Raspberry Pi que hace que esta sea compatible con infinidad de tableros de la marca mikroBUS con sensores, transceptores, drivers de motor, codificadores, puertos de conexión…

IMU (Inertial Measurement Unit): contiene un acelerómetro de tres ejes, que mide la aceleración de avance, y un giróscopo de tres ejes, que mide la velocidad de giro.

Receptor FLYSKY FS-A8S: Es el receptor de radiofrecuencia que comunica el vehículo con el mando radiocontrol, a partir del cual se le mandan referencias de navegación. Con estas referencias se puede modificar la dirección y la velocidad, lo que será útil para la realización de competiciones como parte de las prácticas de laboratorio.

Conversor serie-USB: permite la comunicación entre el receptor Bluetooth cuya información se transmite en serie con la Raspberry a través de un puerto USB.

Display TFT: es la pantalla LCD conectada a la Raspberry del vehículo. Gracias a mostrar la dirección IP permite conectar el vehículo a un router para establecer la conexión con el ordenador.

LED RGB: indica el estado el vehículo. El color rojo indica que el vehículo no está listo, el color azul que se está calibrando la IMU y el color verde que está listo para ponerse en movimiento.

Dos sensores laterales VL6180X ToF (Time of Flight) que permiten medir la distancia y la orientación con respecto a cualquier superficie vertical.

Punto de apoyo esférico: este existe para la utilización del vehículo sin auto balanceo, fuera del objetivo de este proyecto.

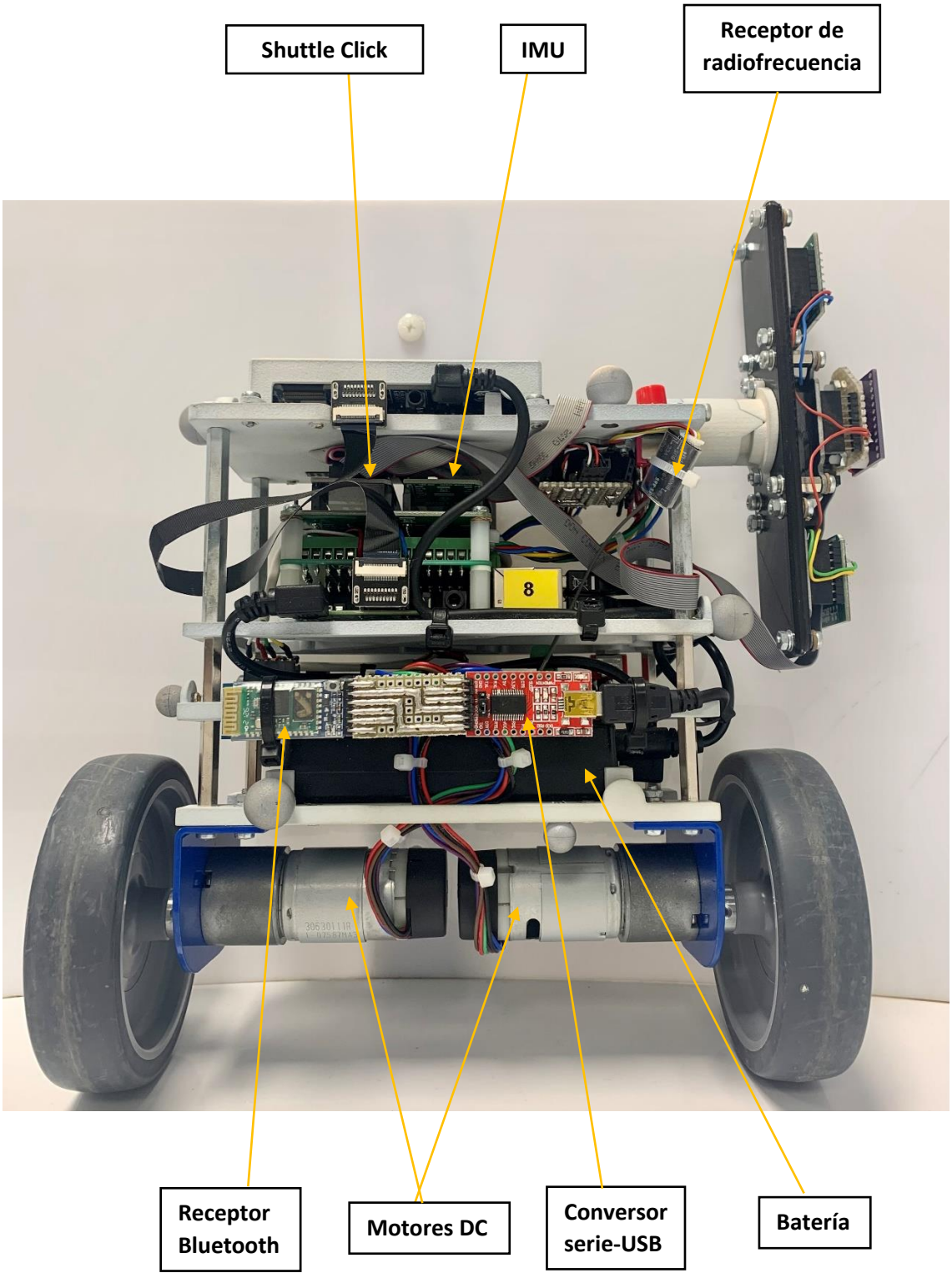


Figura 7: vista frontal del vehículo

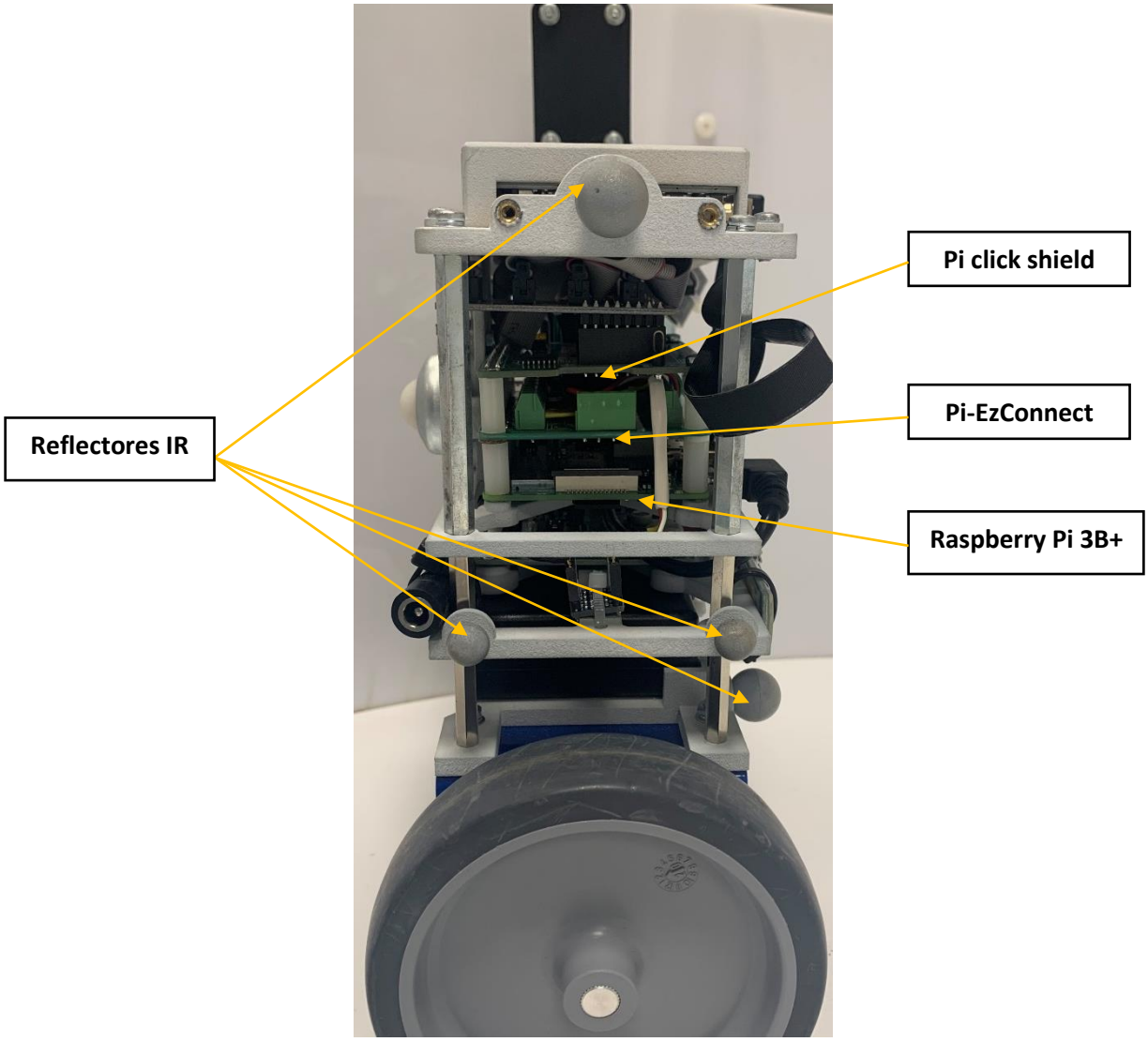


Figura 8: vista lateral del vehículo

3.2 Software

Como ya se ha comentado en el inicio de este capítulo, el cambio de vehículo para las prácticas del laboratorio en la universidad ha sido necesario debido a que MATLAB dejó de dar soporte al vehículo de LEGO. Sin embargo, las actualizaciones para la Raspberry Pi no cesan. MATLAB y Simulink se unen para permitir al usuario la lectura, escritura y análisis de datos procedentes de sensores de la Raspberry Pi, así como el desarrollo de algoritmos que se ejecuten de forma independiente en el microordenador. [6] Fue necesario instalar el paquete de soporte de MATLAB para la Raspberry Pi para comunicar el programa con la Raspberry a través de una red WIFI, así como el paquete de soporte de Simulink para la Raspberry Pi, que amplía Simulink con bloques para configurar la Raspberry.

En definitiva, el uso de MATLAB para la programación en la Raspberry Pi permite analizar los datos provenientes de la microcomputadora gracias a las funciones incorporadas, así como su representación a través de gráficas. La versión utilizada de este programa ha sido la 2022A. Por otra parte, el uso de Simulink está destinado al procesamiento de señales, diseño de control o lógica de estados. Asimismo, cabe destacar que permite ajustar y optimizar los parámetros mientras se ejecuta el programa en la Raspberry.

El control del vehículo es posible gracias a los códigos de MATLAB y diagramas de Simulink generados por el Profesor del Departamento de Electrónica y Automática Juan Luis Zamora Macho, que proporcionó la carpeta CAR_PROJECT.

Entorno de trabajo: MATLAB

Todos los archivos que se necesitan para el control del vehículo se encuentran en una carpeta denominada CAR_PROJECT. Dentro de ella se encuentran doce carpetas más en las que se organizan el resto de archivos.

A continuación, se comenta el contenido de las carpetas utilizadas en el proyecto.

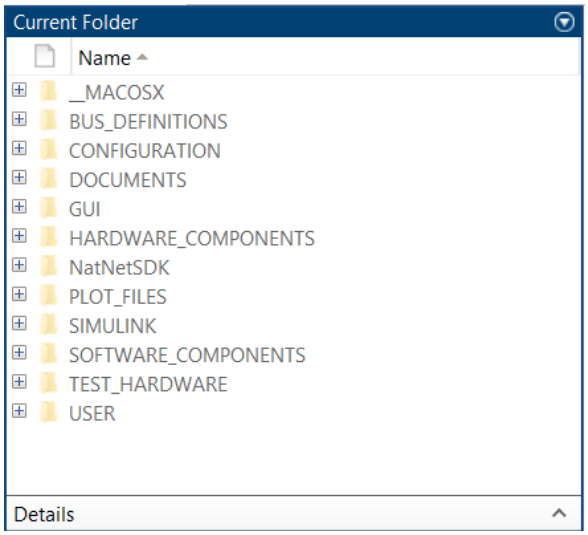


Figura 9: estructura interna de la carpeta CAR_PROJECT

CONFIGURATION: contiene entre otros archivos los dos más usados, ya que se ejecutan para poner en marcha el vehículo y para apagarlo.

CONFIG_CAR.m es archivo que se ejecuta siempre y desde el cual se llama al resto de códigos. Al ejecutarlo se abre también Simulink. En las líneas de código se ha de especificar la IP del vehículo que se está usando. En el caso de usar los ordenadores del laboratorio, la IP de cada coche será el número 192.168.0.1XX, sustituyendo XX por el número del coche. En el caso de este proyecto se utiliza el coche número 8, por tanto, la IP asignada al trabajar en el laboratorio es 192.168.0.108. Debido a que el proyecto se ha realizado durante los meses de verano, se han utilizado distintos ordenadores conectados a distintas redes, por lo que la IP se ha tenido que ir modificando con cada una de ellas. Además de la dirección IP, en CONFIG_CAR.m se determina el modo de ejecución del vehículo, es decir, se especifica si se quiere realizar una simulación, implementar el programa en la Raspberry Pi o realizar un ensayo, entre otros.

SHUTDOWN_CAR es el archivo que apaga la Raspberry Pi a nivel de software. Es necesario siempre apagarla de este modo antes de cortar su alimentación, ya que de otro modo se podría corromper el programa y habría que instalarlo de nuevo.

SOFTWARE_COMPONENTS: dentro de la carpeta "control" se encuentra el archivo CONFIG_CONTROL.m, desde el que se define la configuración y especificaciones del sistema de control que se aplica al vehículo. Se trata de un código extenso, de más de dos mil líneas, organizado en secciones, lo cual facilita su comprensión y modificación. A continuación se explican las secciones utilizadas en el proyecto:

Dentro de CONTROL MODE se indica el tipo de control que se aplicará al vehículo. Para la etapa de estabilización del vehículo se utilizará el modo 8, de auto equilibrado. Para la etapa de control de avance y giro, se utilizará el modo 2. Para la etapa final de seguimiento de pared, el modo de control será el 4.

```
%% CONTROL MODE
%--------------------------------------------------------------------------
% / 0. OPEN LOOP / 1. FORWARD VELOCITY AND YAW ANGLE /
% / 2. FORWARD VELOCITY AND YAW RATE / 3. NAVIGATION /
% / 4. WALL FOLLOWER / 5. WALL FOLLOWING COMPETITION /
% / 6. NAVIGATION COMPETITION / 7. FORWARD VELOCITY COMPETITION /
% / 8. SELF-BALANCE
CONTROL.STATE.CONTROL_MODE = uint8(8);
```

En CONTROL TYPES se define qué controles se aplican. A lo largo de los distintos apartados se irá añadiendo el control de avance y el control de seguimiento de pared, manteniendo el control de cabeceo en un regulador por realimentación de estados en todo momento.

```
%% CONTROL TYPES
%-------------------------------------------------------------------------
% FORWARD VELOCITY CONTROL TYPE
CONTROL.STATE.FORWARD_VEL_CONTROL_TYPE = uint8(1);
%-------------------------------------------------------------------
% YAW RATE CONTROL TYPE
CONTROL.STATE.YAW_RATE_CONTROL_TYPE = uint8(0);
%-------------------------------------------------------------------
% YAW ANGLE CONTROL TYPE
CONTROL.STATE.YAW_ANG_CONTROL_TYPE = uint8(0);
%-------------------------------------------------------------------
% WALL FOLLOWER CONTROL TYPE
CONTROL.STATE.WFL_CONTROL_TYPE = uint8(2);
%-------------------------------------------------------------------
% PITCH ANGLE CONTROL TYPE
CONTROL.STATE.PITCH_ANG_CONTROL_TYPE = uint8(1);
```

En REFERENCE DEFINITIONS se establecerá el tipo de referencia que se aplica a las distintas variables, que serán el objetivo a conseguir por los controles. Por ejemplo, se le puede pedir al vehículo equilibrista seguir una velocidad de avance constante, o seguir un tren de pulsos con lo que varíe su velocidad. Dentro de este apartado se definen también la fuente de las referencias, ya que en este proyecto se utilizará en ocasiones un mando de radiofrecuencia para dar las consignas de velocidad de avance y giro al vehículo. También se define el ángulo de cabeceo inicial del vehículo en 5 grados.

Desde MEASUREMENT FILTERING AND STATE ESTIMATION se elegirá el modo de observador entre el filtro complementario y el filtro extendido de Kalman.

```
%% MEASUREMENT FILTERING AND STATE ESTIMATION
%-----------------------------------------------------------
% OBSERVER MODE
% / 0. FILTERED MEASUREMENT / 1. EKF
CONTROL.STATE.OBSERVER_MODE = uint8(1);
```

Le sigue el apartado OPERATING POINTS, desde el que se definen distintas ganancias de velocidad. Será necesario modificar el vector cuando se requiera una velocidad concreta que no se corresponda con esos puntos de operación.

```
%% OPERATING POINTS (FORWARD VELOCITY)
%-------------------------------------------------------------
% DEFINITION OF OPERATING POINTS (GAIN SCHEDULING)
FORWARD_VEL_OP = [0.2 0.3 0.4 0.5];
```

Por último, se usarán los apartados correspondientes al diseño de cada control. Son tres: PITCH ANGLE: STATE FEEDBACK REGULATOR, para el regulador que estabilice el vehículo; VELOCITY: SBV STATE FEEDBACK CONTROL para el control de velocidad de avance y WALL FOLLOWER: STATE FEEDBACK REGULATOR para el seguidor de pared. El contenido de cada código se verá en mayor detalle en los capítulos de diseño de cada control.

Dentro de la carpeta de SOFTWARE COMPONENTS también se encuentra la carpeta MODEL, que se usará para la identificación del modelo dinámico del vehículo. El archivo MODEL_IDENT contiene el algoritmo por mínimos cuadrados que se utiliza para encontrar los parámetros del vehículo y de los motores que intervienen en el modelo. En la función CONFIG_MODEL se escribirá el valor de los parámetros estimados.

PLOT_FILES: el archivo PLOT_SCOPE_PC es el que permite guardar los datos de un ensayo. Se utilizó a la hora de obtener los parámetros del modelo del vehículo y para imprimir todas las gráficas de los ensayos que se muestran en esta memoria. El archivo PLOT_SCOPE_SIM realiza la misma función para el scope de las simulaciones.

Entorno de trabajo: Simulink

Al ejecutar el archivo CONFIG_CAR se abre automáticamente Simulink en dos pestañas: CAR_CONTROL_SYSTEM y PC_CONTROL_STATION.

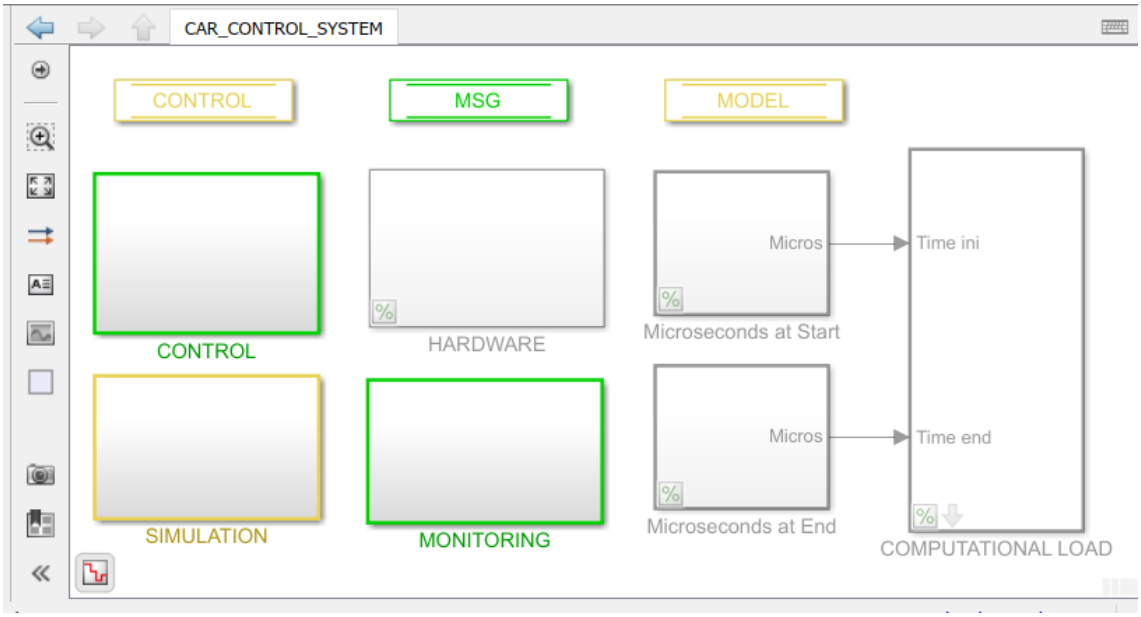


Figura 10: ventana principal del fichero de Simulink CAR_CONTROL_SYSTEM

En la Figura 10 se muestra la pestaña de CAR_CONTROL_SYSTEM. Es el fichero que permite realizar las simulaciones de los controles. Dentro del bloque MONITORING se encuentra el scope donde se pueden ver las simulaciones.

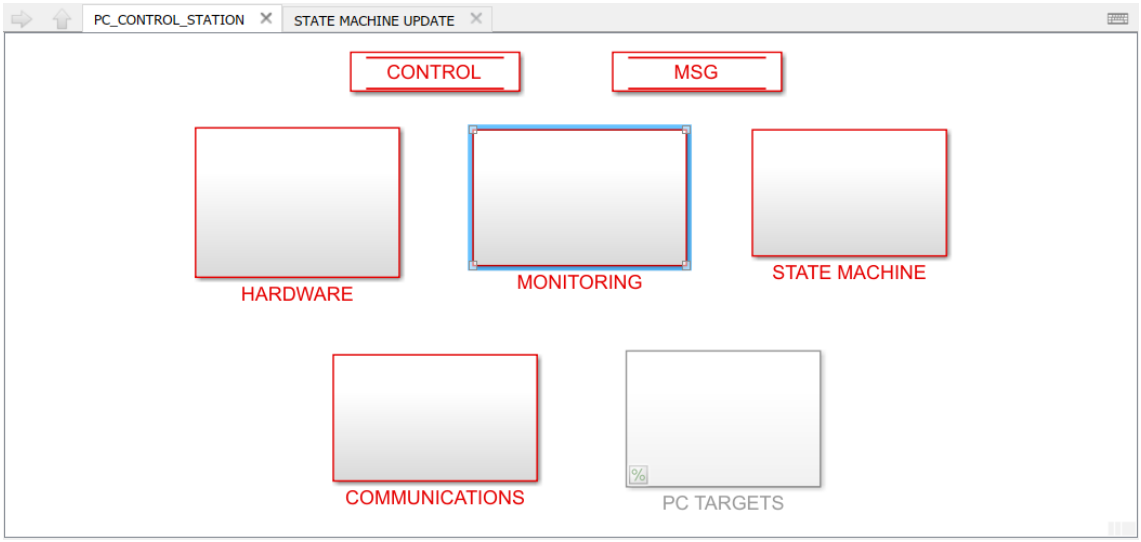


Figura 11: ventana principal del fichero de Simulink, PC_CONTROL_STATION

Dentro del fichero PC_CONTROL_STATION, el bloque HARDWARE únicamente se tuvo que modificar a la hora de usar el vehículo fuera del laboratorio, para configurar la comunicación a través de un router distinto. Esto se realiza clickando en HARDWARE->COMMUNICATIONS->RT_MONITORING-> TCP RSP. En este punto se repite el mismo proceso para los bloques MSG_TX (de transmisión de datos) y MSG_RX (de recepción), que consiste en meterse en el bloque *Packet Output* (y *Packet Input*, respectivamente), clickar en *Board setup* y escribir dentro del campo de *Remote host name or IP address* la dirección IP asignada al vehículo. Esta se puede obtener con un ratón conectado a la Raspberry Pi, posicionándolo en el icono de redes inalámbricas gracias a la pantalla LCD incorporada en el vehículo.

El bloque MONITORING contiene los scopes en los que monitorizar los ensayos realizados. En él, se pueden observar las gráficas de velocidad, tensiones, ángulos de guiñada y cabeceo, posición y distancia a la pared. Este bloque además permite conocer si el estado de comunicación entre el vehículo y Simulink es correcto.

El resto de bloques no se han utilizado de forma activa en la realización de este proyecto.

4. Modelado y obtención de parámetros

El objetivo de este apartado es la obtención de un modelo matemático no lineal, en tiempo continuo y en espacio de estado que represente el movimiento del vehículo a dos ruedas. Será clave para el diseño del control que se adapte al sistema.

El PID es sin duda la estrategia de control más extendida en la industria por su simplicidad y eficacia a la hora de dar solución a la gran mayoría de problemas de control que se plantean en la práctica. Se trata de un control simple, eficaz y robusto. Sin embargo, este tipo de control no proporciona una solución satisfactoria a la planta de un vehículo equilibrista debido a las dinámicas complejas de esta. La principal alternativa que se plantea al PID en el caso de una planta mono variable inestable como la del vehículo auto balanceado objeto de este proyecto consiste en el control por realimentación de estados.

El control por realimentación de estados utiliza la información de un conjunto de variables denominadas variables de estado para generar los mandos de los actuadores. Las variables de estados se agrupan en el vector de estado. Los mandos se generan a partir de una combinación lineal del vector de estados y de la integral de los errores de control. Estos últimos son la diferencia entre las referencias y los valores de las salidas del sistema, o variables que se quieren controlar. El control por realimentación de estados tiene como ventaja fundamental frente al PID la libertad de elección de las dinámicas del sistema en lazo cerrado en el proceso de diseño. No obstante, precisa de un modelo detallado de la planta. En este capítulo, que se divide en dos secciones, se explica en primer lugar cómo se obtiene el modelo de la planta del vehículo equilibrista. En segundo lugar, se trata la identificación de los parámetros del vehículo que en él aparecen mediante el algoritmo de mínimos cuadrados aplicado a unos ensayos.

4.1 Modelo del vehículo equilibrista

Debido a la no linealidad del sistema, se puede trabajar con una planta linealizada alrededor del punto de trabajo. Este será el estado vertical del vehículo. Por ende, para que el control funcione correctamente el ángulo con la vertical ha de ser menor a un ángulo crítico, que se ha de calcular mediante ensayos o simulaciones.

Los parámetros eléctricos, geométricos y mecánicos que intervienen en el modelo del vehículo son los siguientes:

- Los parámetros del motor (referidos a su eje de salida):
  - resistencia R_m
  - constante eléctrica K_e

- constante mecánica K_t
- inercia del motor + rueda en el eje del motor I_m
- constante de fricción viscosa D_m
- relación de engranajes del motor n_1
- par de fricción máximo T_{rmax}

• La masa M del chasis.

• La masa m de cada rueda, incluyendo el rotor del motor.

• El momento de inercia I_z (de giro del vehículo con respecto al eje vertical o guiñada).

• La fricción viscosa D_z (de giro del vehículo con respecto al eje vertical o guiñada).

• La distancia W de separación entre las ruedas.

• El radio R de las ruedas.

• La relación de engranajes n_2 de la rueda

En las ecuaciones se van a tratar variables relacionadas al motor derecho (con el subíndice r) y al motor izquierdo (con el subíndice l), por separado. También se pueden presentar en modo común y diferencial. A continuación, se muestran las ecuaciones que relacionan los valores de una variable cualquiera *x* asignados a los motores izquierdo l y derecho r con las componentes común c y diferencial d.

$$x_c = \frac{x_l + x_r}{2} \qquad (1)$$

$$x_d = \frac{x_l - x_r}{2} \qquad (2)$$

$$x_l = x_c + x_d \qquad (3)$$

$$x_r = x_c - x_d \qquad (4)$$

Se busca la representación de estado del modelo, que es de la forma:

$$\frac{dX}{dt} = F(X, U)$$

$$Y = G(X, U)$$

Donde X es el vector de estado $X = [x_1\ x_2 \dots x_n]^T$; U es el vector de entradas $U = [u_1\ u_2 \dots u_p]^T$ e Y es el vector de salidas $Y = [y_1\ y_2 \dots y_p]^T$.

En el modelo del vehículo se tienen como entradas las tensiones aplicadas a cada motor (u_l y u_r para el izquierdo y derecho, respectivamente), y como variables de estado la velocidad de avance v_c, la velocidad angular de cabeceo w_θ, el ángulo de cabeceo θ, la velocidad angular de guiñada ω_ψ y el ángulo de guiñada ψ.

Se relacionan la velocidad diferencial de las ruedas y las velocidades angulares común y diferencial de las ruedas con las variables de estado:

$$v_d = \frac{W}{2}\omega_\psi$$

$$\omega_c = \frac{v_c}{R}$$

$$\omega_d = \frac{v_d}{R} = \frac{W\omega_\psi}{2R}$$

(5)

También las velocidades angulares de los motores (subíndice m) a partir de las velocidades de las ruedas:

$$\omega_{mc} = n_2\omega_c + \frac{\omega_\theta}{n_1} \quad (6)$$

$$\omega_{md} = n_2\omega_d$$

Se obtienen los pares netos de los motores a partir de las ecuaciones eléctricas, electromecánicas, los pares de fricción estáticos y los pares de rozamiento con el suelo:

$$T_c = T_{mc} - T_{dc} - T_{rc} - F_{rc}\frac{R}{n_2} \quad (7)$$

$$T_d = T_{md} - T_{dd} - T_{rd} - F_{rd}\frac{R}{n_2} \quad (8)$$

Dentro de las ecuaciones $T_c = T_{mc} - T_{dc} - T_{rc} - F_{rc}\frac{R}{n_2}$ (7) y $T_d = T_{md} - T_{dd} - T_{rd} - F_{rd}\frac{R}{n_2}$ (8) se encuentran los siguientes parámetros:

- $T_{mc}\ y\ T_{md}$: pares motores provocados por las corrientes de los motores en modo común y diferencial. Siendo i_c y i_d las corrientes que circulan por el motor en modo común y diferencial, estos pares se obtienen de la siguiente forma:

$$T_{mc} = K_t i_c$$

$$T_{md} = K_t i_d$$

- $T_{dc}\ y\ T_{dd}$: pares de fricción viscosa de los motores en modo común y diferencial, a partir del coeficiente de fricción viscosa y las velocidades angulares de los motores:

$$T_{dc} = D_m \omega_{mc}$$

$$T_{dd} = D_m \omega_{md}$$

- $T_{rc}\ y\ T_{rd}$: pares de fricción estática común y diferencial, a partir de los pares de fricción estática de cada rueda, aplicando las ecuaciones (1) y (2). Los pares T_{rl}y T_{rr} dependen de si la velocidad del motor es nula y el par motor en valor absoluto es inferior al par de fricción máximo $T_{rmáx}$, en cuyo caso son iguales a los pares motores correspondientes. Si el motor gira, lo que ocurre cuando el par motor supera al par de fricción máximo, el par de fricción se mantiene en su valor máximo con el signo del par motor.

$$T_{rc} = \frac{T_{rl} + T_{rr}}{2}$$

$$T_{rd} = \frac{T_{rl} - T_{rr}}{2}$$

- F_{rc} y F_{rd} son las fuerzas de rozamiento de las ruedas con el suelo, R el radio de las ruedas y n_2 la relación de engranajes entre la rueda y el motor.

Una vez se tienen los pares totales de los motores, se puede plantear la ecuación dinámica de los motores en modo común y diferencial:

$$I_m \frac{d\omega_{mc}}{dt} = T_c \quad (9)$$

$$I_m \frac{d\omega_{md}}{dt} = T_d \quad (10)$$

Para simplificar los cálculos se desacoplan el avance y el giro. El avance estará gobernado por la tensión común que llega a los motores mientras que el giro será debido a la componente diferencial.

4.1.1 Modelo de avance y cabeceo

El objetivo es conseguir las ecuaciones de estado, que serán de la forma

$$\frac{dv_c}{dt} = f(v_c, \omega_\theta, \theta)$$

$$\frac{d\omega_\theta}{dt} = f(v_c, \omega_\theta, \theta)$$

$$\frac{d\theta}{dt} = f(v_c, \omega_\theta, \theta\ )$$

Para luego linealizarlas y obtener las matrices de estado A y entrada B en el punto de operación.

Continuando el desarrollo de la ecuación (9)$I_m \frac{d\omega_{mc}}{dt} = T_c$ (9 sustituyendo la expresión para el par neto de la ecuación $T_c = T_{mc} - T_{dc} - T_{rc} - F_{rc}\frac{R}{n_2}$ (7)(7) queda la siguiente expresión:

$$T_c = T_{mc} - T_{dc} - T_{rc} - F_{rc}\frac{R}{n_2} = I_m\frac{d\omega_{mc}}{dt} \qquad (11)$$

Por otro lado, se plantea la ecuación dinámica de traslación del vehículo:

$$F_{rl} + F_{rr} = 2F_{rc} = (2m + M)\frac{dv_c}{dt} - Mh\frac{d\omega_\theta}{dt}cos(\theta) + M\omega_\theta^2 hsen(\theta) \qquad (12)$$

Ambas expresiones tienen la fuerza de rozamiento como variable en común. Despejando esta variable en las dos ecuaciones e igualándolas queda:

$$(2m + M)\frac{dv_c}{dt} - Mh\frac{d\omega_\theta}{dt}\cos(\theta) + M\omega_\theta^2 hsen(\theta) = \frac{2n_2}{R}(T_{mc} - T_{dc} - T_{rc} - \frac{I_m}{n_1}\frac{d\omega_\theta}{dt})$$

Reorganizando esta expresión queda:

$$\frac{2n_2}{R}(T_{mc} - T_{dc} - T_{rc}) - M\omega_\theta^2 hsen(\theta) =$$
$$\left(2m + M + \frac{2n_2^2 I_m}{n_{1R}}\right)\frac{dv_c}{dt} - \left(Mh\,cos(\theta) - \frac{2n_2^2 I_m}{n_{1R}}\right)\frac{d\omega_\theta}{dt} \qquad (13)$$

Finalmente, se plantean las ecuaciones de traslación y rotación que tienen en cuenta la rotación del chasis sobre el eje del sistema de tracción.

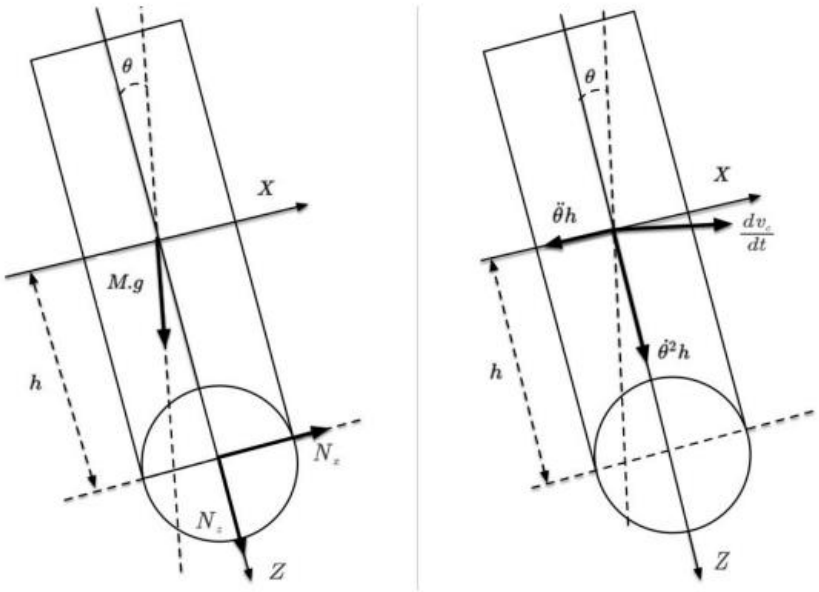


Figura 12: Fuerzas y aceleraciones en el sistema de referencia del chasis [15]

Las fuerzas que intervienen son el peso, las fuerzas normales de reacción en la unión con el eje del sistema de tracción y el par neto de reacción del motor dividido por la relación de transformación del motor.

$$N_x - Mgsen(\theta) = Mcos(\theta)\frac{dv_c}{dt} - Mh\frac{d\omega_\theta}{dt}$$

$$hN_x + \frac{2}{n_1}T_c = I_y\frac{d\omega_\theta}{dt}$$

Se igualan ambas ecuaciones despejando N_x con lo que queda:

$$h\left(M\frac{dv_c}{dt}\cos(\theta) - Mh\frac{d\omega_\theta}{dt} + Mgsen(\theta)\right) + \frac{2}{n_1}T_c = I_y\frac{d\omega_\theta}{dt}$$

Reorganizando la ecuación:

$$\frac{2}{n_1}T_c + Mgsen(\theta) = -Mh\frac{dv_c}{dt}\cos(\theta) + (I_y + Mh)\frac{d\omega_\theta}{dt}$$

Sustituyendo el par total según la ecuación ($T_c = T_{mc} - T_{dc} - T_{rc} - F_{rc}\frac{R}{n_2} = I_m\frac{d\omega_{mc}}{dt}$ (1111)

$$\frac{2}{n_1}(\mathrm{I_m}\frac{\mathrm{d\omega_{mc}}}{\mathrm{dt}}) + Mghsen(\theta) = -Mh\frac{dv_c}{dt}\cos(\theta) + (I_y + Mh^2)\frac{d\omega_\theta}{dt}$$

Sustituyendo ω_{mc} de la ecuación $\omega_{mc} = n_2\omega_c + \frac{\omega_\theta}{n_1}$ (6 (6)

$$\frac{2}{n_1}(\mathrm{I_m}\frac{n_2 d\omega_c}{dt} + \frac{I_m d\omega_\theta}{n_1\, dt}) + Mghsen(\theta) = -Mh\frac{dv_c}{dt}\cos(\theta) + (I_y + Mh^2)\frac{d\omega_\theta}{dt}$$

Y finalmente ω_c de la ecuación (5)

$$\frac{2}{n_1}(\mathrm{I_m}\frac{n_2 dv_c}{R\, dt} + \frac{I_m d\omega_\theta}{n_1\, dt}) + Mghsen(\theta) = -Mh\frac{dv_c}{dt}\cos(\theta) + (I_y + Mh^2)\frac{d\omega_\theta}{dt}$$

Recopilando las ecuaciones anteriores y eliminando variables intermedias, resulta el siguiente modelo no lineal:

$$T_{mc} = \frac{K_t}{R_m}\left(u_c - K_e\left(\frac{n_2 v_c}{R} + \frac{\omega_\theta}{n_1}\right)\right)$$

$$\frac{2n_2}{R}\left(T_{mc} - D_m\left(\frac{n_2 v_c}{R} + \frac{\omega_\theta}{n_1}\right) - T_{rc}\right) - Mh\omega_\theta^2 sen(\theta) = \left(2m + M + \frac{2n_2^2 I_m}{\mathrm{R\^{}2}}\right)\frac{dv_c}{dt} - \left(Mh\,cos(\theta) - \frac{2n_2^2 I_m}{n_1 R}\right)\frac{d\omega_\theta}{dt}$$

$$Mghsen(\theta) = -(Mh\cos(\theta) + \frac{2\mathrm{n}_2 I_m}{n_1 R})\frac{dv_c}{dt} + (I_y + Mh^2 - \frac{2I_m}{n_1^2})\frac{d\omega_\theta}{dt}$$

Del que se pueden despejar las derivadas temporales $\frac{dv_c}{dt}$ y $\frac{d\omega_\theta}{dt}$. Sea conocido que $\frac{d\theta}{dt} = \omega$, se tendría ya entonces el vector de derivadas de las variables de estado.

El modelo no lineal en espacio de estado queda así:

$$p_{11} = 2m + M + \frac{2n_2^2 I_m}{R^2} \qquad p_{12} = -\left(Mh\cos(\theta) - \frac{2n_2 I_m}{n_1 R}\right)$$

$$p_{21} = -\left(Mh\cos(\theta) + \frac{2n_2 I_m}{n_1 R}\right) \qquad p_{22} = I_y + Mh^2 - \frac{2I_m}{n_1^2}$$

$$q_1 = \frac{2n_2}{R}\left(T_{mc} - D_m\left(\frac{n_2 v_c}{R} + \frac{\omega_\theta}{n_1}\right) - T_{rc}\right) - Mh\omega_\theta^2 \text{sen}(\theta) \qquad q_2 = Mgh\text{sen}(\theta)$$

$$\frac{dv_c}{dt} = \frac{p_{22}q_1 - p_{12}\,q_2}{p_{11}p_{22} - p_{12}p_{21}}$$

$$= \frac{\left(I_y + Mh^2 - \frac{2I_m}{n_1^2}\right)\left(\frac{2n_2}{R}\left(\frac{K_t}{R_m}\left(u_c - \overbrace{K_e\left(\frac{n_2 v_c}{R} + \frac{\omega_\theta}{n_1}\right)}^{e_c}\right) - D_m\left(\frac{n_2 v_c}{R} + \frac{\omega_\theta}{n_1}\right) - T_{rc}\right) - M\omega_\theta^2 h\text{sen}(\theta)\right) + Mgh\text{sen}(\theta)\left(Mh\cos(\theta) - \frac{2n_2 I_m}{n_1 R}\right)}{\left(2m + M + \frac{2n_2^2 I_m}{R^2}\right)\left(I_y + Mh^2 - \frac{2I_m}{n_1^2}\right) - \left(Mh\cos(\theta)\right)^2 + \left(\frac{2n_2 I_m}{n_1 R}\right)^2}$$

$$\frac{d\omega_\theta}{dt} = \frac{-p_{21}q_1 + p_{11}\,q_2}{p_{11}p_{22} - p_{12}p_{21}}$$

$$= \frac{\left(Mh\cos(\theta) + \frac{2n_2 I_m}{n_1 R}\right)\left(\frac{2n_2}{R}\left(\frac{K_t}{R_m}\left(u_c - \overbrace{K_e\left(\frac{n_2 v_c}{R} + \frac{\omega_\theta}{n_1}\right)}^{e_c}\right) - D_m\left(\frac{n_2 v_c}{R} + \frac{\omega_\theta}{n_1}\right) - T_{rc}\right) - M\omega_\theta^2 h\text{sen}(\theta)\right) + Mgh\text{sen}(\theta)\left(2m + M + \frac{2n_2^2 I_m}{R^2}\right)}{\left(2m + M + \frac{2n_2^2 I_m}{R^2}\right)\left(I_y + Mh^2 - \frac{2I_m}{n_1^2}\right) - \left(Mh\cos(\theta)\right)^2 + \left(\frac{2n_2 I_m}{n_1 R}\right)^2}$$

$$\frac{d\theta}{dt} = \omega_\theta$$

Considerando el punto de operación el estado vertical del vehículo, se anulan en dicho punto todas las variables de estado ($v_{cop}, \omega_{\theta op}, \theta_{op} = 0$). Con esto se podría linealizar el sistema y obtener la matriz de transferencia entre la tensión común aplicada a ambos motores y el vector de estado.

4.1.2 Modelo de giro

Si bien el avance del coche viene determinado por la tensión común, el giro se decide con la tensión diferencial, que es la mitad de diferencia entre la tensión aplicada al motor izquierdo y la tensión aplicada al motor derecho $u_d = \frac{u_l - u_r}{2}$. Despreciando la inductancia de dispersión, las ecuaciones electromecánicas de los motores en modo diferencial quedan:

$$i_d = \frac{u_d - e_d}{R_m}$$

$$e_d = K_e\omega_{md} = K_e n_2 \omega_d = \frac{K_e n_2 W}{2R}\omega_\psi$$

$$F_{rd} = \frac{n_2}{R}\left(T_{md} - T_{dd} - T_{rd} - I_m\frac{d\omega_{md}}{dt}\right) = \frac{n_2}{R}\left(T_{md} - T_{dd} - T_{rd}\right) - \frac{Wn_2^2 I_m}{2R^2}\frac{d\omega_\psi}{dt}$$

$$T_{md} = K_t i_d = K_t \frac{u_d - e_d}{R_m} = K_t \frac{u_d - K_e \omega_{md}}{R_m} = \frac{K_t}{R_m} u_d - \frac{K_t K_e W n_2}{2 R_m R} \omega_\psi$$

$$T_{dd} = D_m \omega_{md} = D_m n_2 \omega_d = \frac{D_m n_2}{R} v_d = \frac{W D_m n_2}{2R} \omega_\psi$$

$$F_{rd} = \frac{n_2 K_t}{R R_m} u_d - \frac{W n_2^2}{2R^2} \left(\frac{K_t K_e}{R_m} + D_m \right) \omega_\psi - \frac{n_2}{R} T_{rd} - \frac{W n_2^2 I_m}{2R^2} \frac{d\omega_\psi}{dt} \quad (14)$$

Además, se plantea la ecuación dinámica del giro del vehículo con respecto al eje vertical. Siendo la fuerza de rozamiento sobre las ruedas la única fuerza externa que produce aceleración angular, la expresión queda así:

$$F_{rd} W = I_z \frac{d\omega_\psi}{dt} + D_z \omega_\psi \quad (15)$$

Donde $D_z \omega_\psi$ es el par de fricción viscoso asociado al giro de guiñada del vehículo en posición vertical e $I_z$el momento de inercia asociado a dicho giro. Si supone que el control cumple con un ángulo de cabeceo despreciable y se sustituye la expresión de la ecuación $F_{rd} = \frac{n_2 K_t}{R R_m} u_d - \frac{W n_2^2}{2R^2} \left(\frac{K_t K_e}{R_m} + D_m \right) \omega_\psi - \frac{n_2}{R} T_{rd} - \frac{W n_2^2 I_m}{2R^2} \frac{d\omega_\psi}{dt}$ (14) en la ecuación$F_{rd} W = I_z \frac{d\omega_\psi}{dt} + D_z \omega_\psi$ (15) se obtiene la siguiente expresión:

$$\frac{n_2 W K_t}{R R_m} u_d - \frac{n_2 W}{R} T_{rd} = \left(I_z + \frac{W^2 n_2^2 I_m}{2R^2} \right) \frac{d\omega_\psi}{dt} + \left(D_z + \frac{W^2 n_2^2}{2R^2} \left(\frac{K_t K_e}{R_m} + D_m \right) \right) \omega_\psi$$

Agrupando parámetros y simplificando la expresión queda la ecuación diferencial de primer orden:

$$\tau_m \frac{d\omega_\psi}{dt} + \omega_\psi = K_m \left(u_d - \frac{R_m}{K_t} T_{rd} \right) \quad \Rightarrow \quad \begin{cases} \tau_m = \dfrac{I_z + \dfrac{n_2^2 W^2 I_m}{2R^2}}{D_z + \dfrac{n_2^2 W^2}{2R^2} \left(\dfrac{K_t K_e}{R_m} + D_m \right)} \\ K_m = \dfrac{\dfrac{n_2 W K_t}{R R_m}}{D_z + \dfrac{n_2^2 W^2}{2R^2} \left(\dfrac{K_t K_e}{R_m} + D_m \right)} \end{cases}$$

4.2 Estimación de los parámetros del vehículo

Una vez se tiene el modelo dinámico del vehículo en modo equilibrista se han de obtener los parámetros que caracterizan dicho modelo. Estos parámetros varían ligeramente para cada vehículo del laboratorio, siendo el número 8 el que se usa para este proyecto. Para lograr un control más preciso para cada vehículo es conveniente la identificación de los parámetros de cada coche. A continuación, se muestra el proceso detallado, de manera que se pueda repetir este proceso para cualquier otro.

La identificación de sistemas trata de estimar los parámetros desconocidos de los sistemas mediante el uso de los datos de entrada-salida medidos. En otras palabras, la identificación consiste en construir modelos matemáticos de sistemas dinámicos basados en los datos observados de dichos sistemas [6]. En primer lugar es necesario realizar tres ensayos en el laboratorio. El primero de ellos ensayará la velocidad de avance del vehículo, el segundo la velocidad de giro y el tercero el cabeceo en modo equilibrista. Se guardarán los datos tras cada ensayo en un archivo con la extensión .mat gracias al código del archivo PLOT_SCOPE_PC.m dentro de la carpeta PLOT_FILES. Luego, se aplicará el método de mínimos cuadrados hasta que con los parámetros estimados se consigan realizar unas simulaciones con resultados muy similares a los ensayos reales.

4.2.1 Ensayos de identificación de los parámetros del vehículo

Lo primero será configurar el código para realizar los ensayos de identificación de parámetros. Se comienza con el ensayo de avance. Para ello se establece dentro de CAR_PROJECT\CONFIGURATION\CONFIG_CAR, dentro del apartado GENERAL CONFIGURATION la variable RUN_MODE al valor 4, que es el que se utiliza para realizar ensayos con el vehículo. Seguidamente se cambia IDENT_TEST a “true” y, para los dos primeros ensayos, el modo de vehículo a coche (VEHICLE_MODE = 0). Para el último ensayo, el de identificación de la inercia con respecto al eje y se usará el vehículo en modo equilibrista (VEHICLE_MODE = 1).

```
% RUN_MODE DEFINITION
% / 0. REAL-TIME SIMULATION / 1. FAST SIMULATION / 2. TEST VERIFICATION
% / 3. IMPLEMENTATION / 4. ALREADY DEPLOYED / 5. RPI SIMULATION
RUN_MODE = 4;
% TEST FOR MODEL IDENTIFICATION
IDENT_TEST = true;
% VEHICLE MODE
% / 0. CAR / 1. SELF-BALANCING VEHICLE
VEHICLE_MODE = 0;
```

Acto seguido se cambia al archivo CONFIG_CONTROL para establecer el control en modo 2, que es de velocidad de avance y de giro.

```
%% CONTROL MODE
%--------------------------------------------------------------------------
% / 0. OPEN LOOP / 1. FORWARD VELOCITY AND YAW ANGLE /
% / 2. FORWARD VELOCITY AND YAW RATE / 3. NAVIGATION /
% / 4. WALL FOLLOWER / 5. WALL FOLLOWING COMPETITION /
% / 6. NAVIGATION COMPETITION / 7. FORWARD VELOCITY COMPETITION /
% / 8. SELF-BALANCE
CONTROL.STATE.CONTROL_MODE = uint8(2);
```

El tipo de referencia aplicado a los controles se edita desde el apartado REFERENCE DEFINITION de CONFIG_CONTROL. Para el primer ensayo se ha de establecer la referencia de velocidad de avance en una serie de pulsos, lo que se corresponde al tipo de referencia número 2 según el código, mientras se mantienen las referencias de giro en tipo constante.

```
% FORWARD VELOCITY REFERENCE TYPE
% / 0. CONSTANT / 1. PULSE / 2. SQUARE / 3. PRBS / 4. RAMP
CONTROL.STATE.FV_TARGET_TYPE = uint8(2);
%-------------------------------------------------------------
% YAW RATE REFERENCE TYPE
% / 0. NONE / 1. PULSE / 2. SQUARE / 3. PRBS
CONTROL.STATE.YR_TARGET_TYPE = uint8(0);
%-------------------------------------------------------------
% YAW ANGLE REFERENCE TYPE
% / 0. NONE / 1. PULSE / 2. SQUARE / 3. PRBS
CONTROL.STATE.YA_TARGET_TYPE = uint8(0);
```

En este punto el código de MATLAB está listo para el primer ensayo. Para hacer un ensayo, primero se ejecuta CONFIG_CAR, nunca CONFIG_CONTROL ya que este código es una función que es llamada desde el y posteriormente corriendo el programa desde el otro. Para cuando el código se termina de ejecutar ya se han debido abrir distintas pestañas de Simulink automáticamente. PC_CONTROL_STATION será la que se utilice a continuación, en la cual se ha de dar a correr programa clickando en "run in real time". Se recomienda hacerlo desde MONITORING\PC: SCOPES, ya que ahí se muestra una tabla cuyos tres primeros valores son un 1 y el resto 0 cuando la transmisión de datos al vehículo se ha realizado correctamente. Es entonces cuando se puede poner el

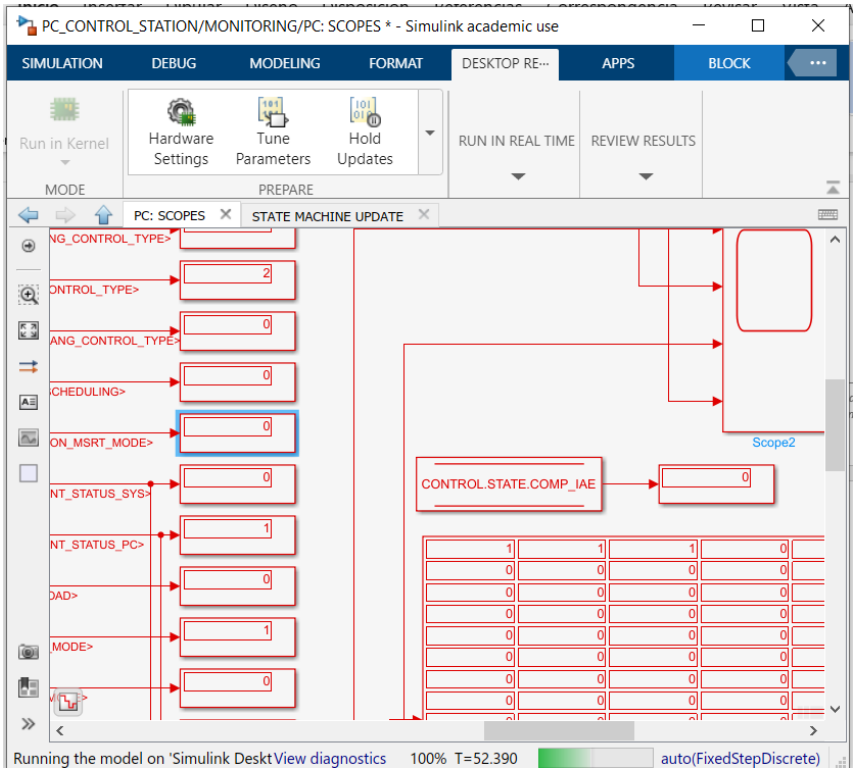


Figura 13: display dentro de PC:scopes indicando que la conexión del vehículo con el programa es correcta

vehículo en marcha.

La secuencia para iniciar el movimiento del vehículo es la siguiente: primero, se coloca el coche en el suelo y se deja inmóvil. Luego, se presiona el botón negro, lo que

provocará que el LED se ilumine de color azul. Esta es la señal de que se está calibrando la IMU. Cuando este proceso acaba, el LED cambia a color verde. En este punto, y dado que el ensayo que se está realizando es con el vehículo modo coche, el vehículo está listo para ponerse en marcha, lo que se consigue pulsando de nuevo el botón negro. Si el vehículo se utilizase en modo equilibrista, como se hará de modo habitual en este proyecto, hay un paso intermedio, que es la estimación del ángulo de cabeceo. Entonces, cuando el LED se pone en verde tras la calibración de la IMU se ha de poner en su posición vertical y volver a pulsar el botón negro, cambiando de nuevo el color del LED a azul. Tras unos 2 segundos, se pondrá de color verde si el ángulo de cabeceo es menor a 5 grados, y el control del ángulo de cabeceo comenzará tras presionar el botón negro una última vez.

Una vez hecho el ensayo se procede a guardarlo en un archivo .mat. Para ello se utiliza el código PLOT_SCOPE_PC dentro de CAR_PROJECT\PLOT_FILES. Dentro del código modificamos FILE_NAME con el nombre que se desee asignar al archivo. Después de correr el programa se guarda automáticamente el archivo del ensayo en la misma carpeta.

4.2.2 Identificación de los parámetros del vehículo a partir de los ensayos

El método de mínimos cuadrados destaca en la estimación de parámetros debido a su simplicidad. Este método busca minimizar la suma de los cuadrados del error que se produce al comparar los valores obtenidos del modelo y los experimentales. El resultado de la identificación del sistema será el conjunto de parámetros que mejor se ajuste a los datos registrados del sistema. Este método aplicado al proyecto, utiliza un grupo de los datos obtenidos de los ensayos de velocidad de avance, velocidad de giro y cabeceo. Para cada muestra hace la diferencia entre el valor estimado y el valor medido, la eleva al cuadrado y suma todos los cuadrados, y ese es el valor que busca minimizar, provocando así que las simulaciones y los ensayos converjan.

Lo primero que se ha de hacer es cargar el archivo del ensayo de avance (.mat) en MATLAB. Para ello basta con hacer doble click en él desde la ventana Current Folder. Después, en CONFIG_CAR se establece la ejecución del programa en TEST VERIFICATION, la variable ident_test a true y el modo del vehículo y de control en CONFIG_CONTROL igual a la del ensayo correspondiente.

```
% RUN_MODE DEFINITION
% / 0. REAL-TIME SIMULATION / 1. FAST SIMULATION / 2. TEST VERIFICATION
% / 3. IMPLEMENTATION / 4. ALREADY DEPLOYED / 5. RPI SIMULATION
RUN_MODE = 2;
% TEST FOR MODEL IDENTIFICATION
IDENT_TEST = true;
% VEHICLE MODE
% / 0. CAR / 1. SELF-BALANCING VEHICLE
VEHICLE_MODE = 0;
```

Tras esto, se debe abrir dentro de la carpeta MODEL en SOFTWARE COMPONENTS el archivo MODEL_IDENT.m.

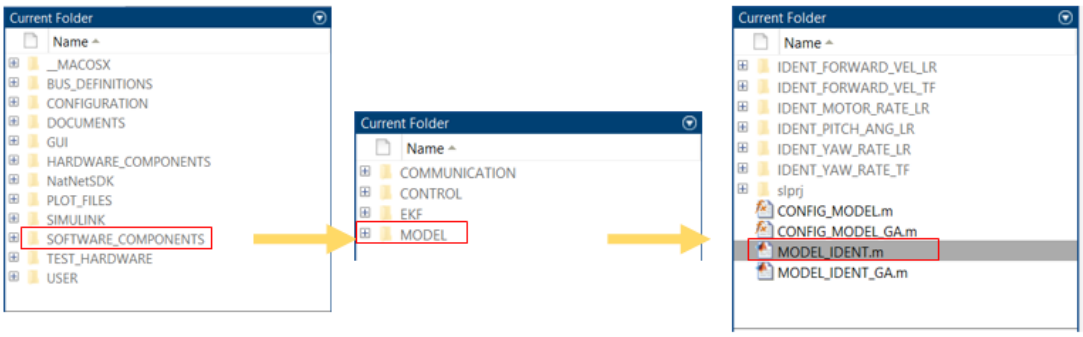


Figura 14: localización del archivo MODEL_IDENT.m

En el código se ha de modificar el valor de la variable FIT_MODE según el ensayo que se haya cargado para identificar. Para el de avance se usará el valor 6, para el de giro el valor 5 y para el de cabeceo el valor 8.

```
%% FIT MODE
%-------------------------------------------------------------
% 1  : MOTOR RATE LEFT
% 2  : MOTOR RATE RIGHT
% 3  : MOTOR RATE COM
% 4  : MOTOR RATE DIF
% 5  : YAW RATE
% 6  : FORWARD VELOCITY
% 7  : PITCH RATE
% 8  : PITCH ANGLE
% 9  : WALL DISTANCE & YAW RATE
% 10  : FORWARD VELOCITY AND YAW RATE
FIT_MODE = 6;
```

Después, se ha de abrir la pestaña CAR_CONTROL_SYSTEM.

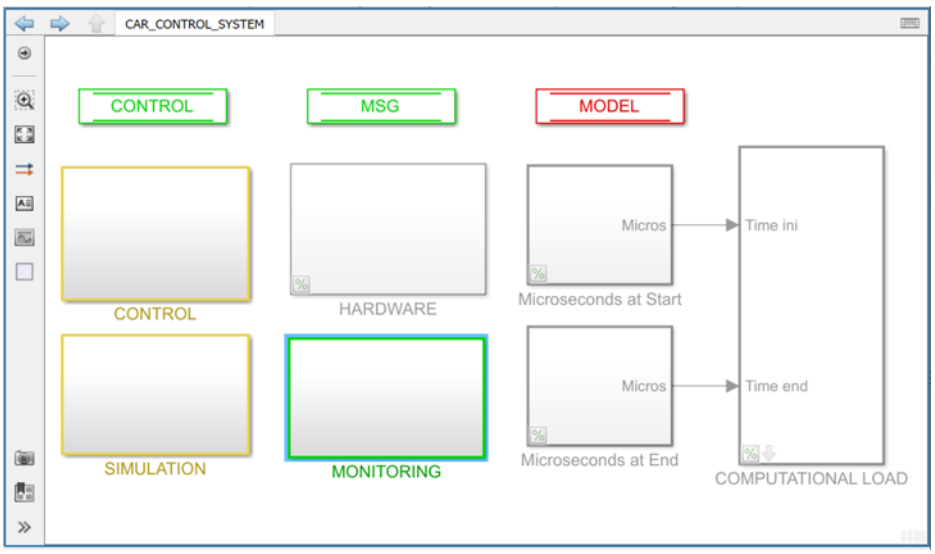


Figura 15: interfaz de CAR_CONTROL_SYSTEM en Simulink:

Para lo que se va a realizar, ha de estar abierto el SCOPE TEST, que se abre de manera automática al correr CONFIG_CAR por primera vez. Si se hubiera cerrado esta pestaña, se puede abrir de nuevo desde MONITORING -> TEST: SCOPES clickando sobre el scope de la derecha.

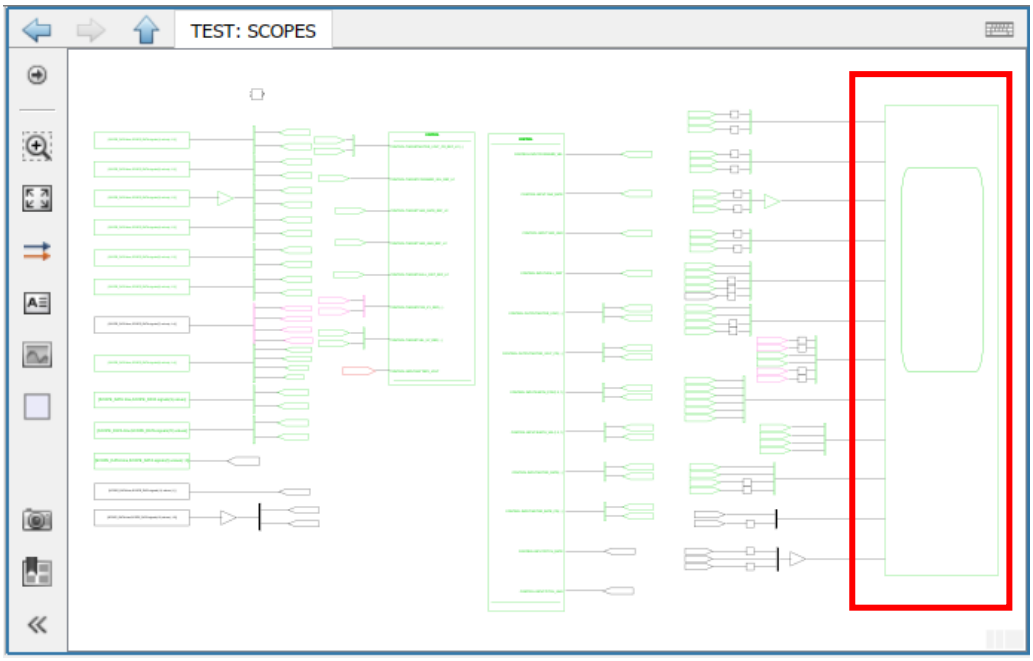


Figura 16: localización del SCOPE_TEST

Por último, se ejecuta el archivo MODEL_IDENT. Si todo va bien, deben aparecer en el SCOPE_TEST las simulaciones.

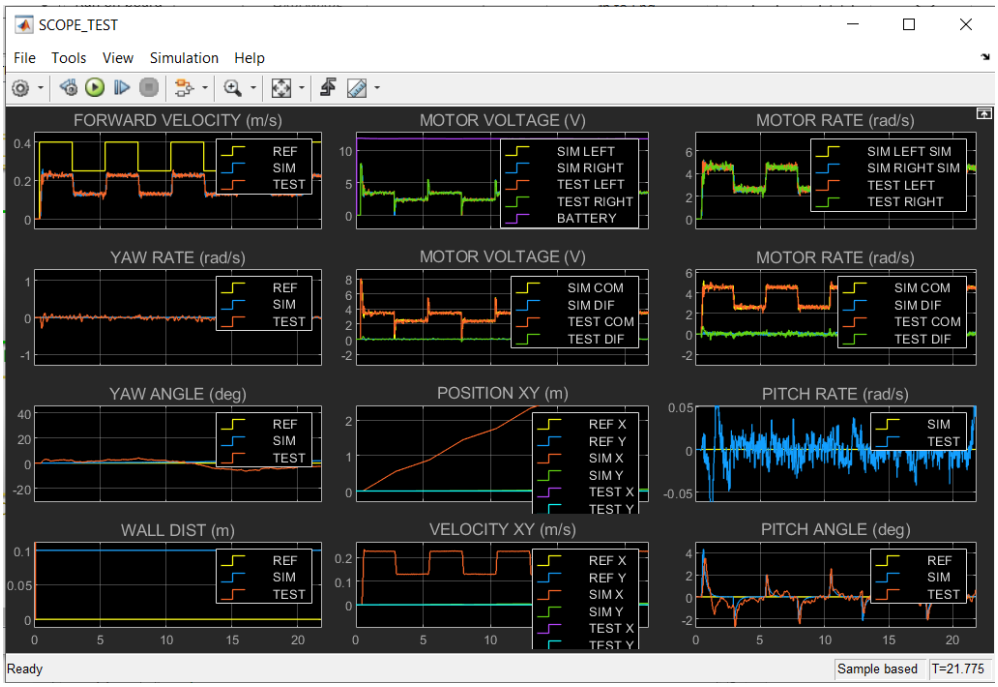


Figura 17: SCOPE_TEST ejecutando el archivo MODEL_IDENT

Para el coche 8, el resultado de la identificación de parámetros para el modelo de avance quedó como se muestra en la Figura 18. Para llegar a él solo fue necesaria una iteración del algoritmo.

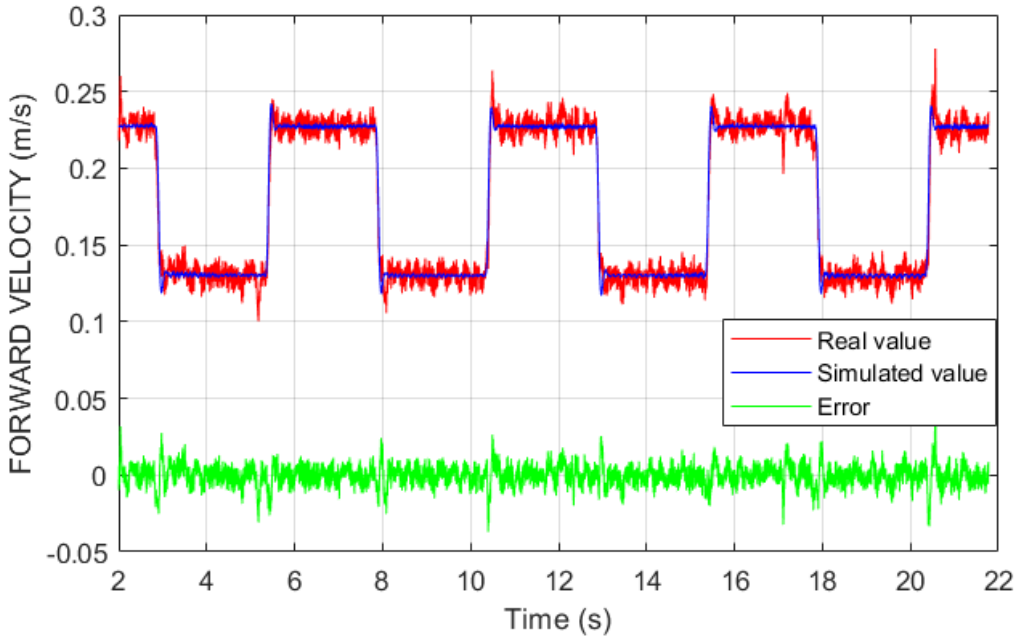


Figura 18: resultado de la identificación del modelo de avance

En la figura se muestra en color rojo la medida real registrada durante el ensayo, en color azul la medida simulada con el modelo y en color verde el error entre ambas gráficas.

Se copian los parámetros que resultaron de esta primera identificación en el apartado ESTIMATED PARAMETERS dentro de la función CONFIG_MODEL.m que se encuentra en la carpeta MODEL dentro de SOFTWARE_COMPONENTS (ver Figura 14). Se debe copiar directamente el código generado en el Command Window desde la parte de

```
%--------------------------------------------------------------
% MOTOR PARAMETERS
%--------------------------------------------------------------
```

hasta llegar a

```
MODEL.PARAM.IMU_PITCH_OFFSET = 0;
```

estando esas líneas incluidas, y pegar sobre el mismo código del apartado indicado, de manera que lo acabe sustituyendo con los nuevos valores estimados. Una vez actualizado el código, se repite el mismo proceso esta vez con los datos del ensayo de velocidad de giro. En este paso se van a actualizar exclusivamente los parámetros que

afectan a la velocidad de giro. No debe olvidarse modificar los parámetros de las referencias en CONFIG_CONTROL a los del ensayo y la variable FIT_MODE a 5, tras cargar los datos del ensayo.

Tras ocho iteraciones que tomaron 33 minutos, el resultado de la identificación del ensayo de velocidad de giro es el que se muestra en la Figura 19.

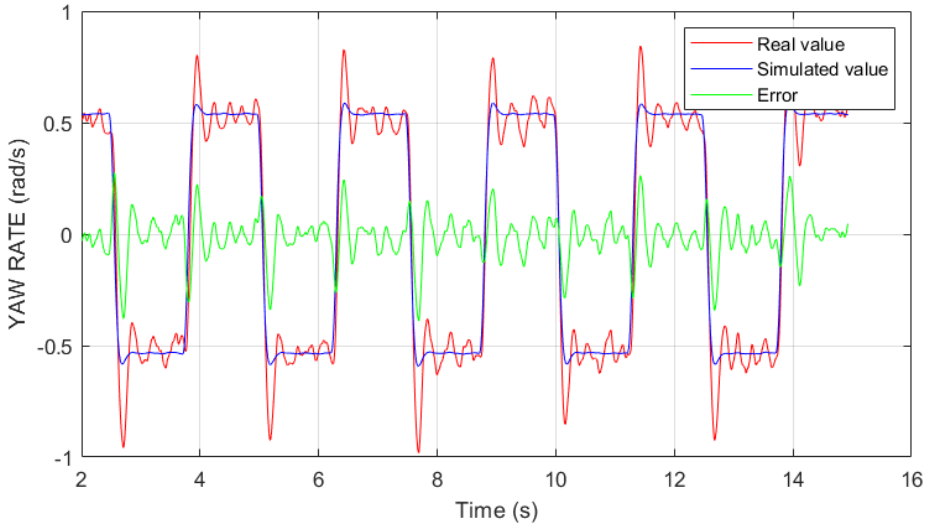


Figura 19: resultado de la identificación de velocidad de giro

Tras actualizar de nuevo el código con los parámetros estimados, se han identificado todos los parámetros que tienen que ver con el vehículo en modo coche. Falta finalmente la identificación de parámetros relacionados con el vehículo equilibrista. Se pasa por tanto a la identificación del momento de inercia de cabeceo, para el que se cargará en MATLAB el archivo del ensayo de cabeceo. Se debe tener en cuenta que hay que cambiar el modo del vehículo a equilibrista, manteniendo el modo de ejecución al 2 y la variable IDENT_TEST en true. En CONFIG_CAR se establece el modo de control de auto balanceo (modo 8), y se elimina la referencia de velocidad de giro del ensayo anterior. Para esta identificación, hay que ir a SOFTWARE_COMPONENTS -> MODEL -> IDENT_PITCH_ANG_LR, y abrir el fichero ident_pitch_ang.m y ejecutarlo.

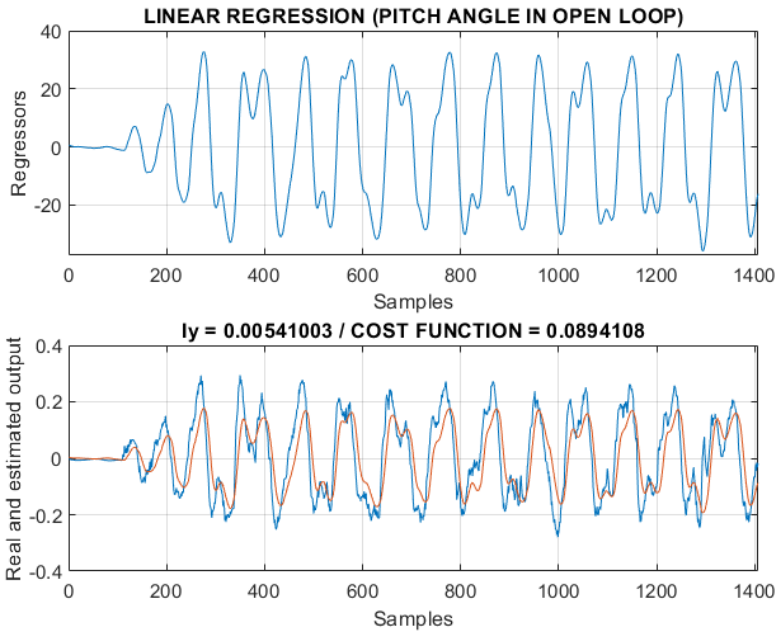


Figura 20: resultado identificación del momento de cabeceo

En la gráfica inferior de la Figura 20 se tiene el cabeceo del coche. Se realiza el método de mínimos cuadrados pero únicamente estimando un parámetro, que se muestra en el título de la gráfica inferior. De nuevo se copia la lista de parámetros del Command Window y se mete en el archivo CONFIG_MODEL. Quedaría un parámetro a estimar, el momento de inercia de giro cuando el vehículo está en modo equilibrista. Este parámetro se obtiene del modelo CAD del vehículo. Los valores estimados finales son los siguientes:

MOTOR PARAMETERS

Rated voltage (V)

MODEL.PARAM.BATTERY_VOLT = 12;

Rotor resistance (ohm)

MODEL.PARAM.MOTOR_RESISTANCE = 7.101;

Motor back-emf constant (V.s/rad)

MODEL.PARAM.MOTOR_BEMF_CONSTANT = 0.540626;

Torque constant (N.m/A)

MODEL.PARAM.MOTOR_TORQUE_CONSTANT = 0.540626;

Motor inertia (kg.m^2)

MODEL.PARAM.MOTOR_INERTIA = 0.000322251;

Viscous friction constant (N.m.s/rad)

MODEL.PARAM.MOTOR_VISCOUS_FRICTION_COEFF = 0.000931;

Static friction torque (N.m)

MODEL.PARAM.MOTOR_STATIC_TORQUE = 0.0736695;

Motor converter voltage drop (V)

MODEL.PARAM.MOTOR_VOLT_DROP_COM = 0;

MODEL.PARAM.MOTOR_VOLT_DROP_DIF = -0.00604305;

Motor converter delay (s)

MODEL.PARAM.MOTOR_DELAY = 0.05;

Motor gear ratio

MODEL.PARAM.MOTOR_GEAR_RATIO = 1;

CAR PARAMETERS

Wheel mass (kg)

MODEL.PARAM.WHEEL_MASS = 0.147;

Wheel radius (m)

MODEL.PARAM.WHEEL_RADIUS = 0.05;

Wheel distance (m)

MODEL.PARAM.WHEEL_DISTANCE = 0.2037;

Body mass (kg)

MODEL.PARAM.BODY_MASS = 1.5-2*MODEL.PARAM.WHEEL_MASS';

Body inertia in X axis (kg.m^2)

MODEL.PARAM.BODY_INERTIA_X = 0.00983486;

Body inertia in Y axis (kg.m^2)

MODEL.PARAM.BODY_INERTIA_Y = 0.00541003;

Body inertia in Z axis (kg.m^2)

MODEL.PARAM.BODY_INERTIA_Z = 0.0368737;

Body viscous friction coefficient in X axis (N.m.s/rad)

MODEL.PARAM.BODY_VISCOUS_FRICTION_COEFF_X = 0.17572;

Body viscous friction coefficient in Z axis (N.m.s/rad)

MODEL.PARAM.BODY_VISCOUS_FRICTION_COEFF_Z = 0.020664;

Distance from wheel axis to body COG [m]

MODEL.PARAM.BODY_DISTANCE = 0.065;

Wheel gear ratio

MODEL.PARAM.WHEEL_GEAR_RATIO = 1;

Pitch offset (rad)

MODEL.PARAM.IMU_PITCH_OFFSET = 0;

5. Sistema de control

En este capítulo se introduce la teoría del control por realimentación de estados y se explica por qué es la estrategia de control adecuada para el vehículo equilibrista. Posteriormente se muestran los pasos para obtener los controles que equilibran el vehículo y mantienen su velocidad según la referencia dada, así como los resultados obtenidos de cada control.

El objetivo básico del sistema de control es modificar la dinámica del vehículo para obtener un punto de equilibrio estable en su posición vertical, modificando la velocidad de avance y la velocidad de giro mediante la tensión suministrada a los motores del vehículo. El vehículo se ha de mantener en su posición vertical a pesar de las pequeñas perturbaciones que se le puedan aplicar o de que exista una ligera inclinación inicial. El control PID es el más extendido en la industria, ya es intuitivo, robusto, y da solución a la mayoría de los problemas de control. Sin embargo, en el caso de una planta inestable como la del vehículo equilibrista, no proporciona una solución satisfactoria. La principal alternativa para el PID es el control por realimentación de estados.

El control por realimentación de estados utiliza la información de un grupo de variables del sistema, denominadas variables de estado, para generar los mandos de los actuadores. En el caso del vehículo equilibrista, las variables de estado serán el ángulo de cabeceo, la velocidad de avance y la velocidad de giro, y al introducir el control de seguimiento de pared se sumará la distancia a la pared. El control se fundamenta en la realimentación de las variables de estado en el sistema, para realizar una combinación lineal entre ellas y el error de control y así generar los mandos. Este tipo de control utiliza información muy detallada del sistema, lo que involucra un número elevado de sensores y los consecuentes errores de medición que estos conllevan. Es por ello que en muchas ocasiones se introducen estimadores de estado en los controles. Durante la realización de este proyecto se alternará el uso de dos estimadores de estado: un filtro complementario y un filtro extendido de Kalman.

5.1 Estimadores de estado

El ángulo de cabeceo no se puede medir directamente. La IMU del vehículo contiene un acelerómetro de tres ejes, que mide la aceleración lineal, y un giróscopo de tres ejes, que mide la velocidad angular. Los acelerómetros son sensibles a las vibraciones y a la diferencia entre el campo gravitacional y la aceleración lineal del sensor. Ante la ausencia de aceleración lineal se pueden usar los acelerómetros para determinar los ángulos de orientación, por lo que se utilizan filtros para poder separar la componente gravitacional de la lineal en la aceleración. La ecuación a partir de la cual se obtiene el ángulo de orientación a partir de las medidas de aceleración del acelerómetro, siendo $A_{X_{out}}$, $A_{Y_{out}}$, $A_{Z_{out}}$las medidas registradas de aceleración en los ejes X, Y y Z:

$$\theta = \operatorname{atan}\left(\frac{A_{X_{out}}}{\sqrt{A_{Y_{out}}^2 + A_{Z_{out}}^2}}\right)$$

[7]Por otra parte, los giroscopios obtienen la velocidad angular, e integrando esta se puede obtener el ángulo de orientación. La desventaja es que el proceso de integración conlleva errores que crecen con el tiempo y se precisa conocer las condiciones iniciales.

$$\theta = \int \omega \, dt$$

Una solución habitual para el problema que presenta la medida del ángulo de orientación es el denominado filtro complementario, una combinación de ambos sensores con filtros de paso alto y paso bajo. Dado que el error del giroscopio suele ser a frecuencias bajas y el error del acelerómetro a frecuencias altas, se puede usar la estimación del acelerómetro cuando la velocidad del sistema es baja y la del giroscopio a velocidades altas.

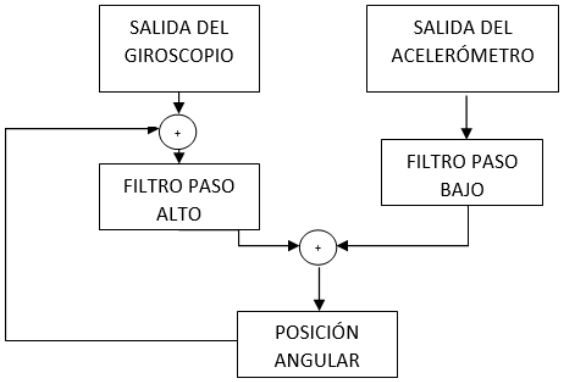


Figura 21: esquema de la implantación de un filtro complementario

Una solución más sofisticada, pero con mayor peso computacional es el filtro de Kalman. El filtro de Kalman es un algoritmo de procesamiento de datos recursivo óptimo, que puede estimar el estado actual de un sistema a partir de una serie de mediciones incompletas que contienen ruido. Este filtro puede compensar el error dinámico del acelerómetro y el error de deriva del giroscopio, en un proceso de filtrado que se puede descomponer en una actualización de tiempo o Predicción, y una actualización de medición o Corrección. Las ecuaciones de predicción utilizan las estimaciones de covarianza de estado y error anteriores para estimar el estado actual. Las ecuaciones de corrección combinan el estado actual estimado y el estado actual medido para obtener la estimación óptima del estado del sistema.

5.2 Control de estabilización del vehículo

En este apartado se va a diseñar un regulador por realimentación de estados con el fin de que el vehículo se auto balancee, es decir, que se mantenga en vertical. No se aplicarán restricciones sobre su velocidad de avance, ya que esto es objeto del apartado 5.3. Se van a diseñar dos vertientes del control de estabilización. El primero utilizará un filtro complementario para estimar el ángulo de cabeceo. El segundo sustituirá el filtro complementario por un filtro de Kalman. Se comprobará el funcionamiento de ambos reguladores con sendas simulaciones y posteriormente se comprobará experimentalmente con ensayos.

El control actúa sobre la tensión común que se aplica sobre los motores que, como se explicó en el apartado 4.1.1 Modelo de avance y cabeceo", es el mando que determina la dinámica de avance y cabeceo del vehículo. La calidad del control depende de la asignación de polos en lazo cerrado. Sin embargo, no es trivial juzgar qué asignación es mejor que otra. Un buen control es aquel que actúa con rapidez y da una respuesta lo suficientemente amortiguada, sin amplificar demasiado la sensibilidad frente a errores de modelado y la amplificación en el mando del ruido de medida. Es posible medir esa sensibilidad mediante procesos matemáticos, aunque con MATLAB y Simulink se pueden obtener simulaciones que permitan establecer como criterio de calidad la amplitud de las oscilaciones del ángulo de cabeceo y de la velocidad de avance. [8]

El diseño se realizará modificando los autovalores de la matriz de estado en lazo cerrado del control por realimentación de estado. A partir de esta matriz se calculan las ganancias que se aplican a las variables de estado para calcular el mando. A continuación se muestra el fragmento de código que se ha de modificar para el diseño del control. Este se encuentra dentro del archivo CONFIG_CONTROL.m.

```
%% PITCH ANGLE: STATE FEEDBACK REGULATOR
%-------------------------------------------------------------
% Design method
% 1. Pole placement / 2. LQR
PITCH_ANG_SFC.design_method = 1;
if CONTROL.STATE.OBSERVER_MODE == 1 % EKF
    % Damping factor
    PITCH_ANG_SFC.damping_factor = 0.85;
    % Natural frequency factor: closed-loop wn / open-loop wn
    PITCH_ANG_SFC.wn_factor = 0.55;
    % Third pole module / closed-loop wn
    PITCH_ANG_SFC.p_factor = 20;
    % LQR state weighting matrix
    PITCH_ANG_SFC.matQ = [30 1.25 1];
    % LQR MV weighting matrix
    PITCH_ANG_SFC.matR = 0.3;
else % Complementary filter
    % Damping factor
    PITCH_ANG_SFC.damping_factor = 0.9;
    % Natural frequency factor: closed-loop wn / open-loop wn
```

```
    PITCH_ANG_SFC.wn_factor = 0.60;
    % Third pole module / closed-loop wn
    PITCH_ANG_SFC.p_factor = 20;
    % LQR state weighting matrix
    PITCH_ANG_SFC.matQ = [20 1.3 1];
    % LQR MV weighting matrix
    PITCH_ANG_SFC.matR = 0.75;
end
```

Entre "if" y "else" se encuentra el código asociado al control con el filtro de Kalman, mientras que se modificarían los parámetros entre "else" y "end" si se está realizando el diseño para el filtro complementario. Para indicar el observador se ha de modificar la variable CONTROL.STATE.OBSERVER_MODE que se encuentra en el apartado MEASUREMENT FILTERING AND STATE ESTIMATION, colocando un 0 en el paréntesis de uint() para el filtro complementario y un 1 para el filtro de Kalman.

```
%% MEASUREMENT FILTERING AND STATE ESTIMATION
%------------------------------------------------------------
% OBSERVER MODE
% / 0. FILTERED MEASUREMENT / 1. EKF
CONTROL.STATE.OBSERVER_MODE = uint8(1);
```

5.2.1 Regulador con filtro complementario

Se ajusta el código de CONFIG_CAR.m estableciendo el modo de simulación (RUN_MODE = 1), asegurando que esté configurado el modo equilibrista del vehículo (VEHICLE_MODE = 1) y la variable IDENT_TEST = false para que se aplique el control diseñado. Luego, se selecciona el modo de control equilibrista. El código ya está preparado para pasar a realizar el diseño del regulador.

Acto seguido, se han de buscar los parámetros que consigan estabilizar el vehículo a dos ruedas. Se parte de los siguientes parámetros iniciales:

$$\xi = 0.85$$

$$\frac{w_{n_{cl}}}{w_{n_{ol}}} = 0.55$$

$$w_{n_{3rdpole}} = 20$$

Con estos parámetros el control se desestabilizaba, como se muestra en las gráficas de la simulación en la Figura 22. Se aprecia que pasados los 7 segundos el coche pierde el equilibrio y cae.

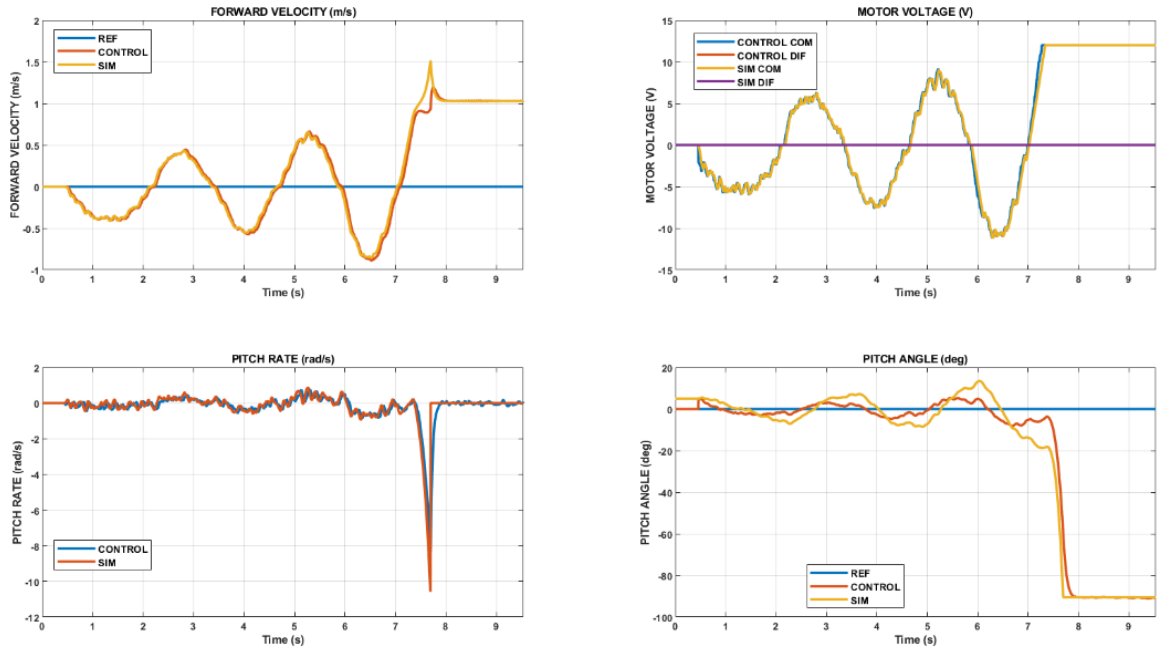


Figura 22: Simulación inicial del regulador con filtro complementario

Se editan los parámetros hasta que se consigue una simulación en la que el coche se mantiene estable. Esto se obtiene tras utilizar un factor de pulsación natural mayor y aumentar la seta, con lo que con los parámetros

$$\xi = 0.9$$

$$\frac{w_{n_{cl}}}{w_{n_{ol}}} = 0.6$$

$$w_{n_{3rdpole}} = 20$$

queda un control estable, como se puede ver en la Figura 23. Según la simulación, el vehículo mantendrá el equilibrio oscilando entre +8 grados y -8 grados con respecto a la vertical.

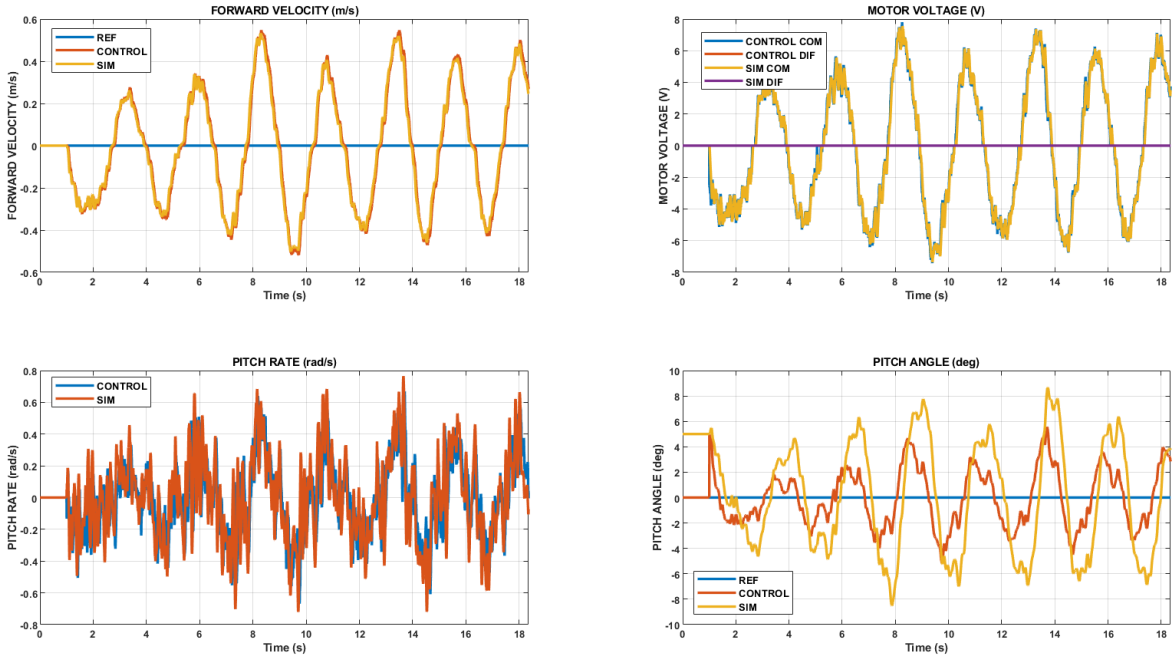


Figura 23: simulación final del regulador con FC

Posteriormente, se pone a prueba el regulador realizando un ensayo con el vehículo sobre el suelo del laboratorio de Control.

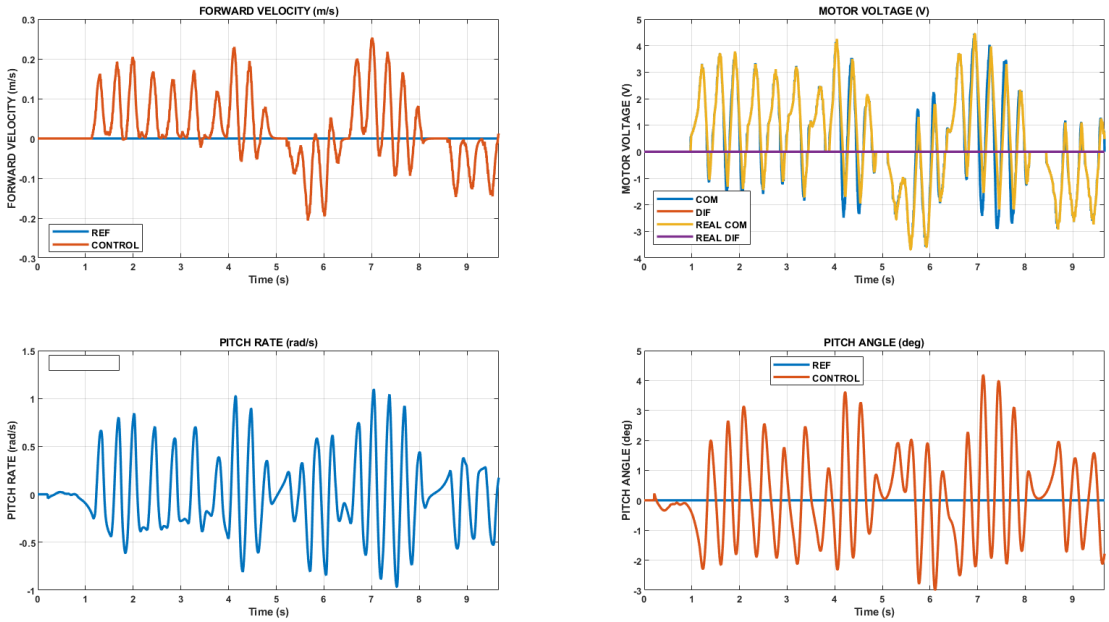


Figura 24: ensayo del regulador con FC

5.2.2 Regulador con filtro de Kalman

Se modifica el modo del observador al 1, como se indica en la introducción del capítulo 0Se sigue el mismo procedimiento que en el apartado anterior. Los resultados obtenidos en la simulación y ensayo del regulador se muestran en la Figura 25 y la Figura 26, obtenidas para los siguientes parámetros:

$$\xi = 0.85$$

$$\frac{w_{n_{cl}}}{w_{n_{ol}}} = 0.55$$

$$w_{n_{3rdpole}} = 20$$

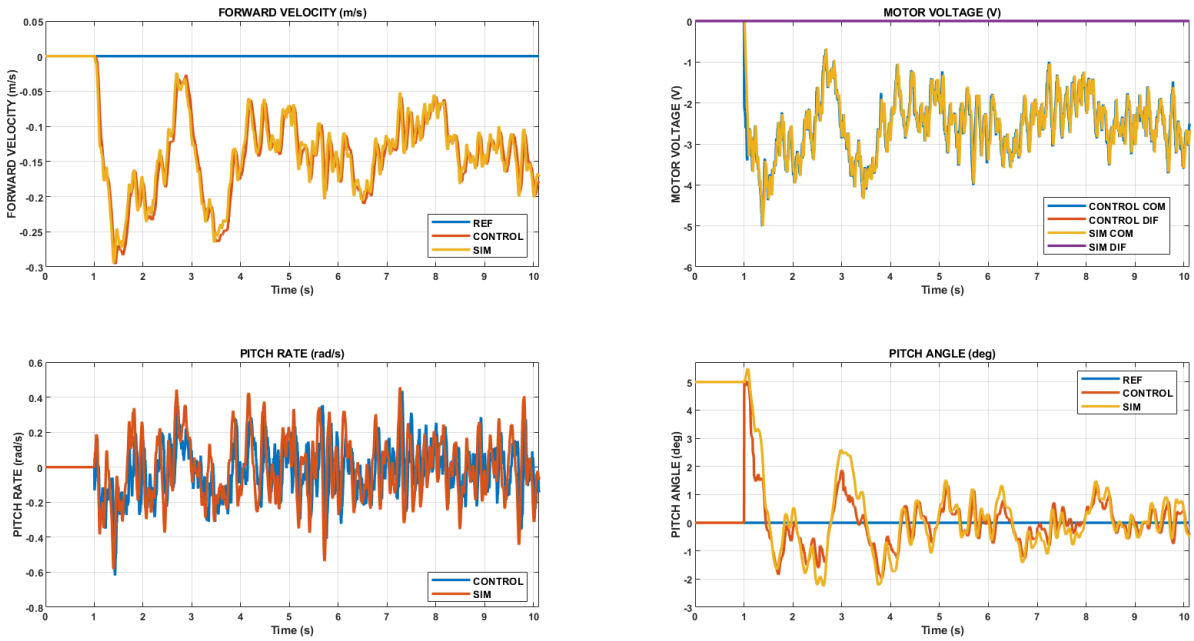


Figura 25: simulación del regulador con EKF

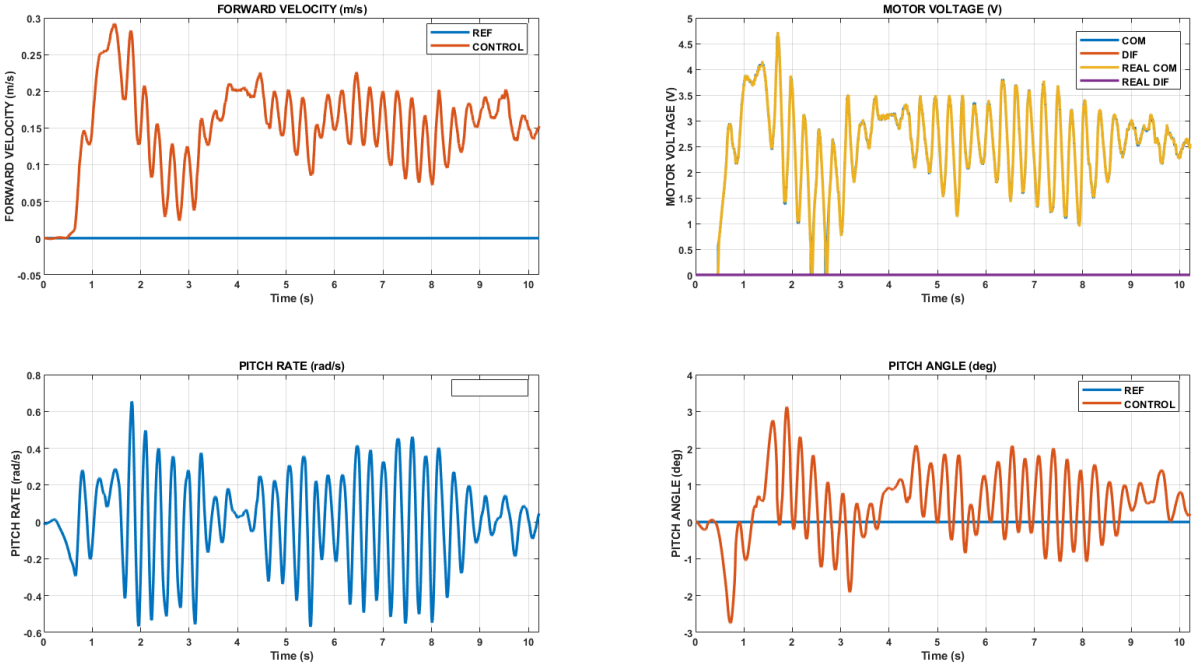


Figura 26: ensayo final del regulador con EKF

5.3 Control de avance y giro

En este punto ya se ha conseguido el primer objetivo que era estabilizar el vehículo. Sin embargo, no se actúa sobre la velocidad de avance. El siguiente objetivo es el de diseñar un control que incorpore una acción integral para garantizar una velocidad concreta de avance, así como una velocidad nula, si se desea. Al igual que en el apartado 5.2, se va a implementar este control en el vehículo en primer lugar con un filtro complementario y posteriormente con un filtro extendido de Kalman. Para ambos observadores se realizarán primero simulaciones que permitirán tantear hasta encontrar unos parámetros que estabilicen el vehículo, y luego se comprobará experimentalmente el buen funcionamiento de los controles con ensayos sobre el vehículo. De forma adicional, se incorporará el uso del mando por radiofrecuencia para controlar la navegación del vehículo, tanto en avance como en giro.

Antes de proceder con el diseño del control, se configura el código de MATLAB para este tipo de control. Se mantiene en CONFIG_CAR la configuración de vehículo equilibrista (VEHICLE_MODE = 1) y la variable IDENT_TEST en false. En el directorio CAR_PROJECT \SOFTWARE_COMPONENTS \CONTROL \CONFIG_CONTROL se modifica el modo de control, que ya no será el modo de vehículo equilibrista sino un control de avance y giro:

```
%% CONTROL MODE
%------------------------------------------------------------------------
% / 0. OPEN LOOP / 1. FORWARD VELOCITY AND YAW ANGLE /
% / 2. FORWARD VELOCITY AND YAW RATE / 3. NAVIGATION /
% / 4. WALL FOLLOWER / 5. WALL FOLLOWING COMPETITION /
% / 6. NAVIGATION COMPETITION / 7. FORWARD VELOCITY COMPETITION /
% / 8. SELF-BALANCE
CONTROL.STATE.CONTROL_MODE = uint8(2);
```

Dentro del apartado CONTROL TYPES se especifica un control por realimentación de estado a la variable velocidad de avance:

```
% FORWARD VELOCITY CONTROL TYPE
% / 0. PID  / 1. STATE FEEDBACK CONTROL / 2, FIRST_ORDER DEAD BEAT
% / 3. SECOND-ORDER DEAD BEAT
CONTROL.STATE.FORWARD_VEL_CONTROL_TYPE = uint8(1);
```

El fragmento de código que contiene los parámetros que se han de modificar en el diseño del control es el del apartado denominado VELOCITY: SBV STATE FEEDBACK CONTROL.

5.3.1 Control integral por realimentación de estado con filtro complementario

En el apartado 5.2 se indicó cómo cambiar la configuración del uso del observador. Se ha de elegir por tanto la del filtro complementario. En primer lugar, se realizan las simulaciones pertinentes hasta dar con el conjunto de parámetros que estabilicen el control. Ahora se han de modificar cuatro parámetros y no tres, debido al polo añadido de la acción integral. Dentro de los vectores 1x2, se ha de cambiar el primer elemento, que se corresponde a la velocidad de avance. El segundo elemento corresponde a la velocidad de giro, y no se ha de modificar.

```
else % Complementary filter
    % Damping factor
    VEL_SFC.damping_factor = [0.6 0.7];
    % Natural frequency factor: closed-loop wn / open-loop wn
    VEL_SFC.wn_factor = [0.85 0.8];
    % Third pole module / closed-loop wn (only forward velocity)
    VEL_SFC.p3_factor = 0.1;
    % Fourth pole module / closed-loop wn (only forward velocity)
    VEL_SFC.p4_factor = 10;
```

El primer parámetro es el factor de amortiguamiento, el segundo el factor de pulsación natural en lazo cerrado con respecto al lazo abierto. El tercer y el cuarto parámetro se corresponden a un factor aplicado a la pulsación natural de los polos complejos en lazo cerrado.

Se procede a encontrar la combinación de los cuatro parámetros que estabilizan el vehículo. Primero se aplica una referencia de velocidad nula. Una vez se estabilice el vehículo se probarán referencias con valores no nulos para comprobar que se siguen dichos valores. Los parámetros de los que se parte son $\xi = 0.6$; $\frac{w_{n_{cl}}}{w_{n_{pl}}} = 0.85$; $\frac{w_{n_3}}{w_n} = 0.1$; $\frac{w_{n_4}}{w_n} = 10$. Los parámetros con los que se estabiliza el control y cuya simulación se muestra en la Figura 27 son los siguientes:

$$\xi = 0.7$$

$$\frac{w_{n_{cl}}}{w_{n_{pl}}} = 0.99$$

$$\frac{w_{n_3}}{w_n} = 0.02$$

$$\frac{w_{n_4}}{w_n} = 10$$

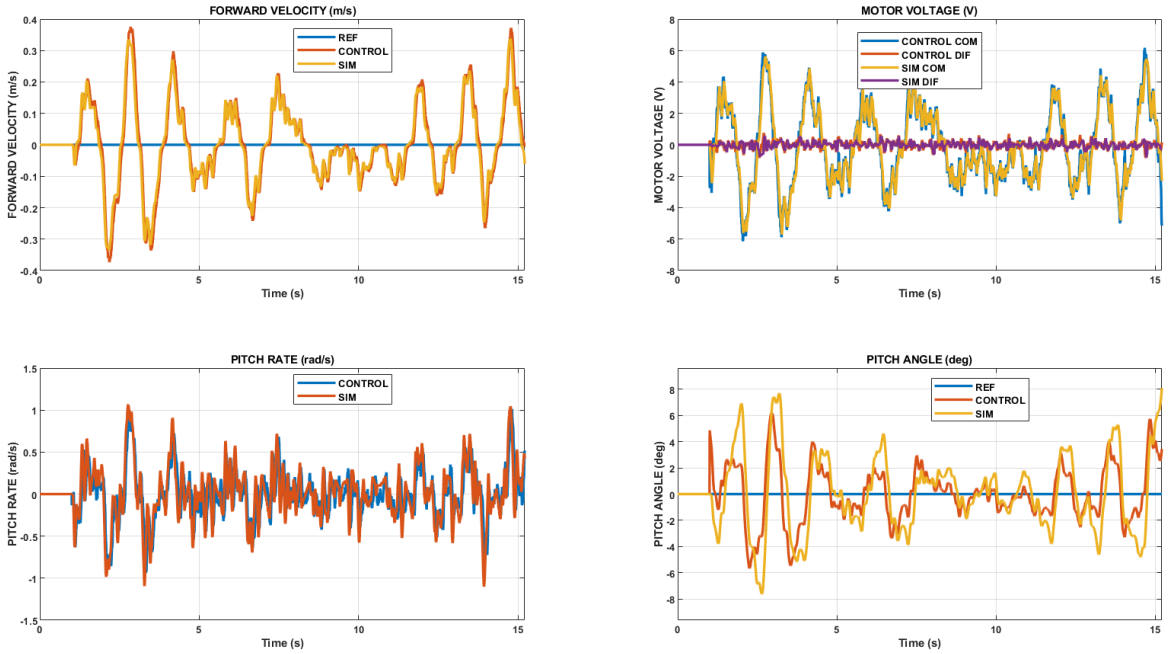


Figura 27: simulación del control por realimentación de estados con seguimiento de referencia nula con FC

Tras comprobar en la Figura 27 que el vehículo se mantiene en equilibrio, se quiere poner a prueba el seguimiento de referencia cambiando el valor constante de referencia de velocidad de avance a un valor positivo. Se utiliza el segundo valor del vector de punto de operaciones, el cual se corresponde con una velocidad de 0,3 m/s. Para actualizar dicho valor de ha de ir al apartado REFERENCE DEFINITION de CONFIG_CONTROL.m:

```
% FORWARD VELOCITY CONSTANT REFERENCE VALUE (0-4)
CONTROL.STATE.FV_TARGET_VALUE = uint8(2);
```

La información sobre la velocidad de cada punto de operación se encuentra en OPERATING POINTS. El primer valor del vector sería el valor de referencia 1, el segundo el 2, el tercero el 3 y el cuarto el 4. El valor de referencia 0 es el de velocidad nula.

```
OPERATING POINTS (FORWARD VELOCITY)
%-------------------------------------------------------------
% DEFINITION OF OPERATING POINTS (GAIN SCHEDULING)
FORWARD_VEL_OP = [0.2 0.3 0.4 0.5];
```

El resultado de la simulación para una referencia de 0,3 m/s de velocidad de avance se muestra en la Figura 28. Se trata de un resultado satisfactorio ya que el control aplicado sobre el vehículo permite que al aplicar una referencia la velocidad de avance cambie de un valor nulo hasta oscilar sobre el valor objetivo.

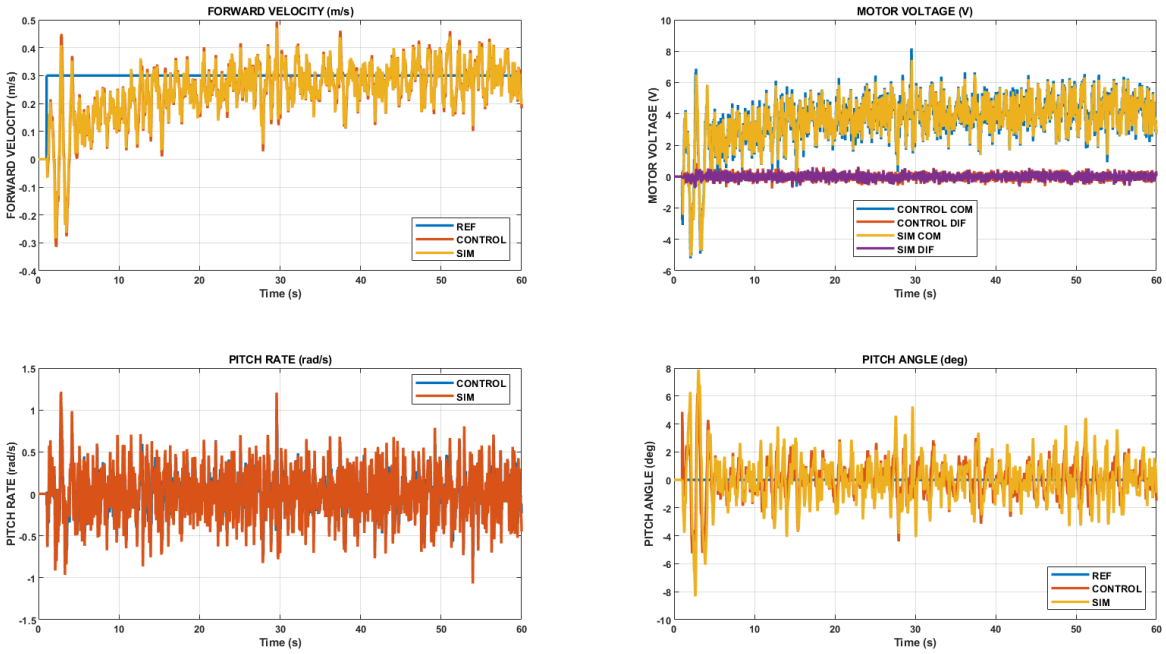


Figura 28: simulación del control por realimentación de estados con seguimiento de referencia no nula con FC

5.3.2 Control por realimentación de estado con filtro extendido de Kalman

Se establece de nuevo el filtro extendido de Kalman y se elimina la referencia de velocidad. Se realiza una simulación para los valores que se muestran en el fragmento del código siguiente, que se encuentra en el archivo CONFIG_CONTROL.m precediendo al del apartado 5.3.1 Control integral por realimentación de estado con filtro complementario

```
if CONTROL.STATE.OBSERVER_MODE == 1 % EKF
    % Damping factor
    VEL_SFC.damping_factor = [0.6 0.7];
    % Natural frequency factor: closed-loop wn / open-loop wn
    VEL_SFC.wn_factor = [0.85 0.8];
    % Third pole module / closed-loop wn (only forward velocity)
    VEL_SFC.p3_factor = 0.1;
    % Fourth pole module / closed-loop wn (only forward velocity)
    VEL_SFC.p4_factor = 10;
    % LQR state weighting matrix
    VEL_SFC.forward_vel_matQ = [12 6 0.85 1];
```

```
VEL_SFC.yaw_rate_matQ = [0.1 1];
% LQR MV weighting matrix
VEL_SFC.forward_vel_matR = 2.5;
VEL_SFC.yaw_rate_matR = 0.1;
```

El resultado de la simulación es conforme, ya que como se muestra en la Figura 29, mantiene el vehículo en el punto de operación vertical además de una velocidad de avance nula, tal y como se le configuró la referencia.

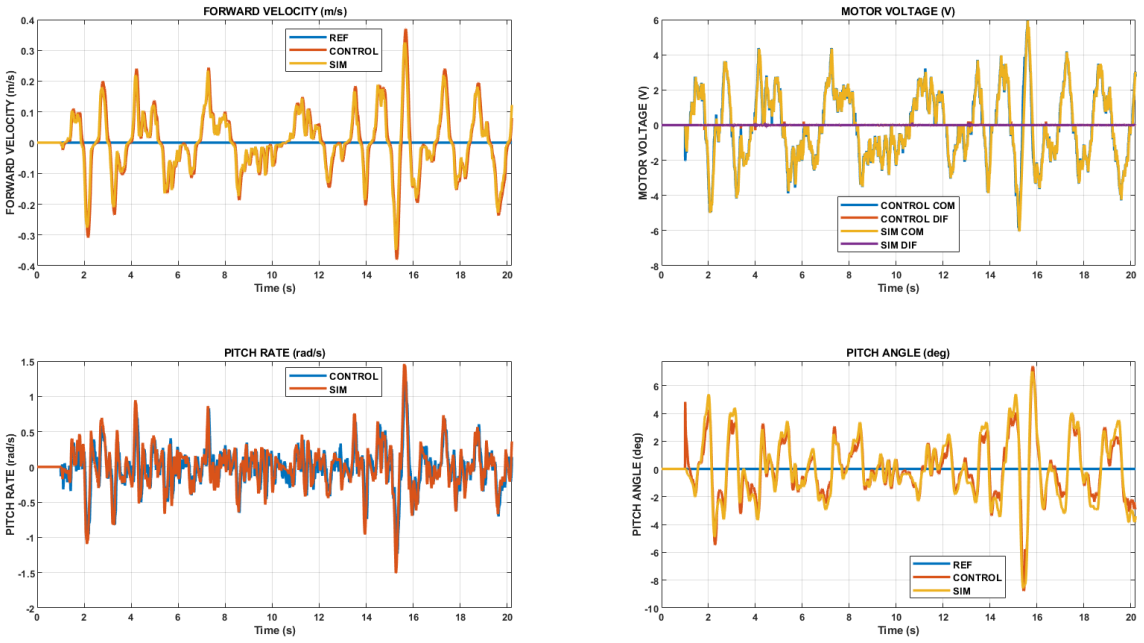


Figura 29: Simulación del control con seguimiento de referencia nula con EKF

Al igual que en el apartado anterior, se realiza una segunda simulación, esta vez con referencia no nula que asegure el correcto funcionamiento del control de velocidad. El control cumple con las expectativas ya que se aprecia como la velocidad de avance sigue la referencia constante de velocidad. Por último se realiza el ensayo del control para referencia nul y se comprueba como el vehículo se mantiene en su sitio, manteniendo el equilibrio y con unas oscilaciones del ángulo de cabeceo mínimas, entre -3 y +5 grados. Los resultados del ensayo se adjuntan en la **¡Error! No se encuentra el origen de la referencia.**.

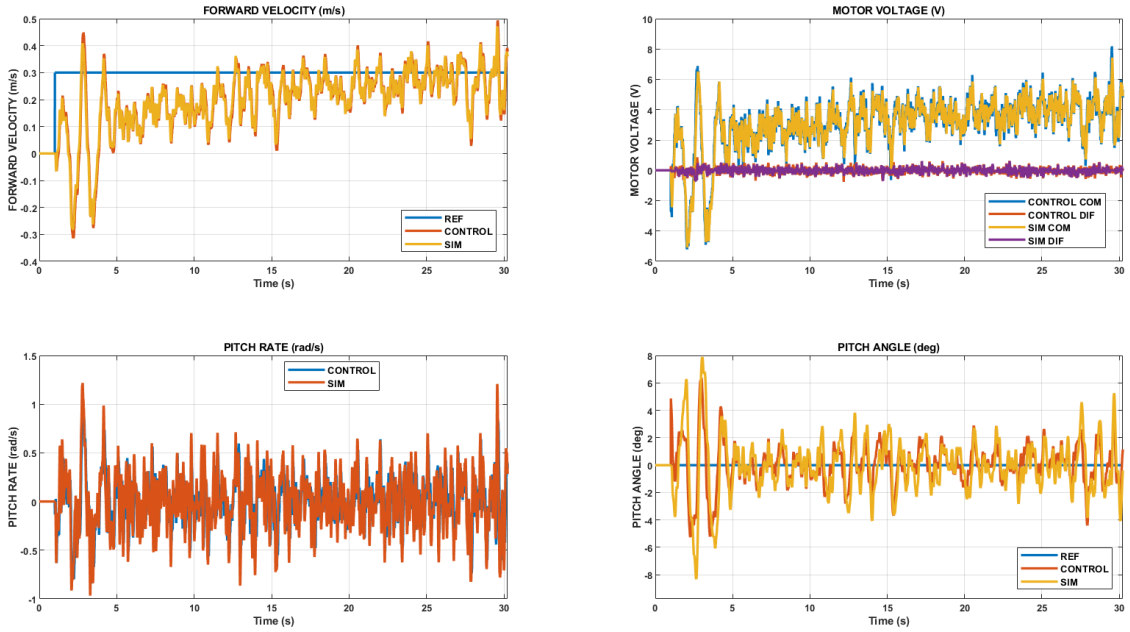


Figura 30: simulación del control con seguimiento de referencia no nula con EKF

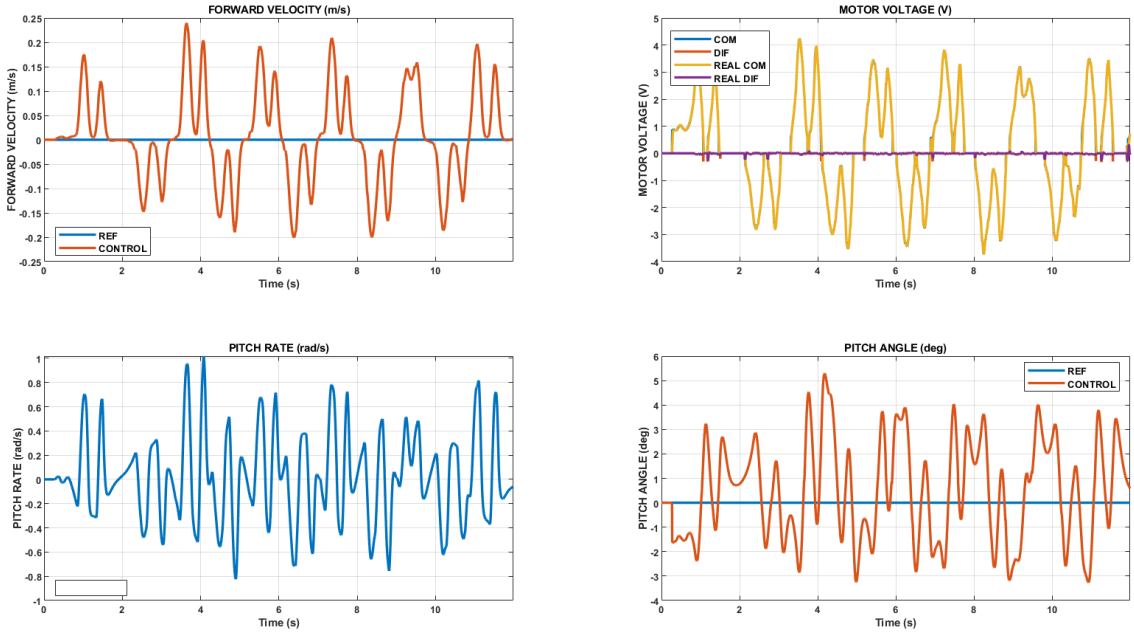


Figura 31: ensayo del control con seguimiento de referencia nula con EKF

5.3.3 Navegación del vehículo equilibrista mediante el mando de radiofrecuencias

Para darle un valor añadido al proyecto, se propone en este punto incorporar el uso del mando por radiofrecuencia para asignar las referencias de velocidad de avance. Se ha de modificar en el código de MATLAB la fuente de referencias. Esto se consigue desde el apartado REFERENCE DEFINITION, en el archivo CONFIG_CONTROL.m, en el cual se ha de modificar las variables CONTROL.STATE.FV_TARGET_SOURCE y CONTROL.STATE.YR_TARGET_SOURCE:

```
% REFERENCE SOURCE
% / 0. LOCAL / 1. PC  / 2. RC JOYSTICK / 3. RC SWITCHES
CONTROL.STATE.MV_TARGET_SOURCE = uint8([0 0])';
CONTROL.STATE.FV_TARGET_SOURCE = uint8(2);
CONTROL.STATE.YR_TARGET_SOURCE = uint8(2);
CONTROL.STATE.YA_TARGET_SOURCE = uint8(0);
CONTROL.STATE.WD_TARGET_SOURCE = uint8(0);
```

El resultado del ensayo se recoge en la Figura 32. Se puede comprobar cómo se le aplica inicialmente una referencia de velocidad de avance negativa, o dicho de otra forma, se le pide al vehículo ir hacia atrás, para luego cambiar el sentido de dicha velocidad. Es por ello que primero las oscilaciones de FORDWARD VELOCITY se desplazan hacia el semieje negativo. También se observa el giro que realiza el vehículo pasados 17 segundos desde el inicio del ensayo. Todo esto ocurre a la vez que se mantiene un ángulo de cabeceo que no supera un valor absoluto de 6 grados.

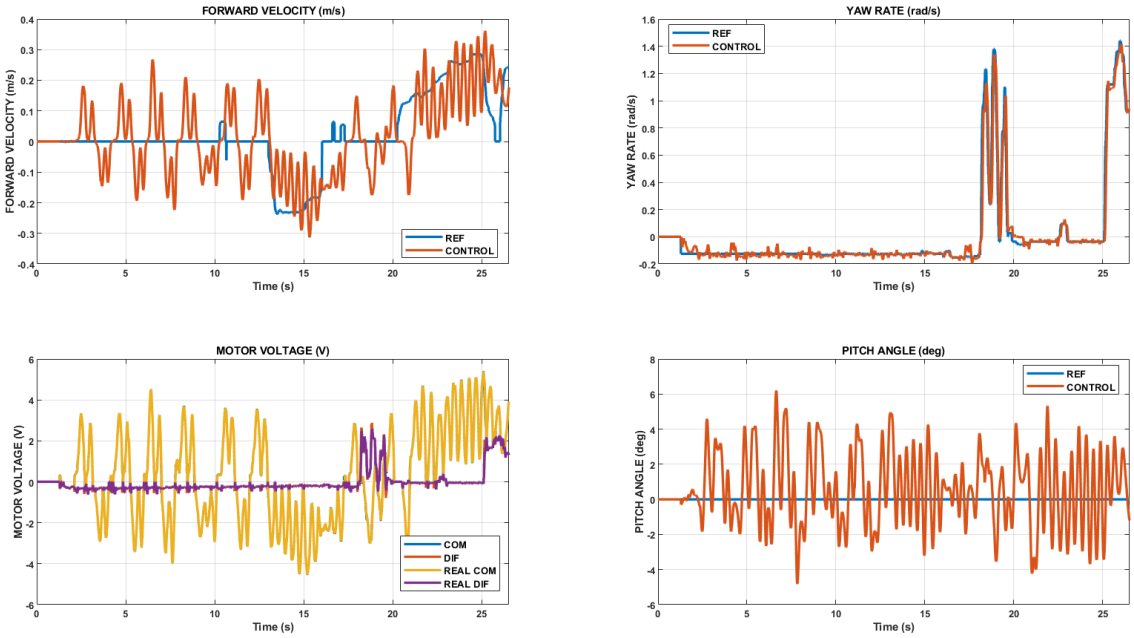


Figura 32: Ensayo de navegación en modo equilibrista con el mando de radiofrecuencia

6. Navegación del vehículo equilibrista mediante el seguimiento de pared

En este apartado se diseña un regulador por realimentación de estado para seguir la pared. El objetivo del regulador es mantener una distancia específica con una superficie vertical a pesar de las perturbaciones que aparezcan, en este caso la asimetría de los motores, la variación de la referencia de velocidad de avance o el recorrido de una curva. El control se pondrá a prueba en tres niveles: el primero será el recorrido en línea recta en paralelo a una pared, en el que el vehículo mantendrá una referencia de distancia constante. El segundo nivel consiste en cambiar la referencia constante de distancia a la pared por un tren de pulsos. En este ensayo, el vehículo debe hacer un recorrido en el que se acerque y aleje de la pared siguiendo los pulsos. Por último, el tercer nivel trata de dar una vuelta completa a un circuito cerrado que consta de dos tramos rectos y dos curvas cerradas, además de una rampa de subida y otra de bajada. Para lograr los distintos objetivos se probarán los controles primero con simulaciones y luego se implantarán en el vehículo para la comprobación de su funcionamiento con ensayos.

6.1 Modelo de vehículo para el seguimiento de pared

El vehículo consta de dos sensores de distancia a la pared del modelo VL6180X ToF, colocados en su lateral. Estos sensores miden el tiempo de vuelo que tarda la luz láser infrarroja en alcanzar la pared y reflejarse en un detector con una resolución de hasta un milímetro, pudiendo medir distancias de hasta 60 cm. Cada sensor proporciona una medida de distancia, por lo que se diferenciarán d_1y d_2, y se tomará d_s como la separación entre ambos sensores. Con estos datos, el ángulo de orientación con la pared es $\alpha = \text{atan}\,(\frac{d_1-d_2}{d_s})$. Si se tiene en cuenta el punto medio entre ambos sensores, al que se denominará A, y se quiere obtener la distancia mínima a la pared de dicho punto, se tendría que obtener el valor medio de las distancias medidas por los sensores y proyectar dicha distancia. Ese valor está representado como d_n y se calcula pues como $d_n = \frac{d_1+d_2}{2}\cos(\alpha)$. Esta distancia también puede calcularse como la integral de la componente normal de la velocidad del punto A, que es la velocidad del vehículo. Con la notación $v_a$velocidad de avance del vehículo, ω velocidad angular de rotación y x_a e y_a las coordenadas del vehículo para un sistema de referencia solidario a él con origen en el punto medio del eje que conecta las ruedas, las componentes de la velocidad quedan:

Eje x: $v_{Ax} = v_A - y_A\omega$

Eje y: $v_{Ay} = x_A\omega$

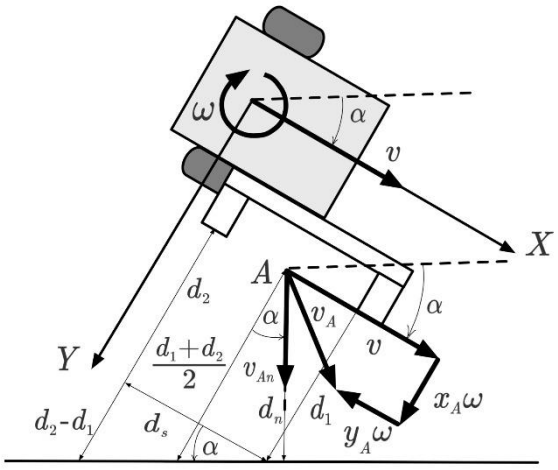


Figura 33: Localización de los sensores ToF y medidas con respecto a la pared [8]

Con esos ejes, la componente normal a la pared de la velocidad de A se calcula como

$$v_{An} = v_{Ax} sen(\alpha) + v_{Ay} \cos(\alpha)$$

Y así la distancia a la pared se puede determinar integrando dicha expresión:

$$d(t) = d(0) + \int v_{An}(t) dt$$

Por otro lado, se tiene una ecuación diferencial de primer orden que relaciona la tensión diferencial del mando con la velocidad angular del vehículo:

$$\frac{d\omega}{dt} = -\frac{1}{\tau_{m_\omega}} \omega + \frac{K_{m_\omega}}{\tau_{m_\omega}} u_d$$

En la que K_{m_ω}es la ganancia estática y τ_{m_ω}es la constante de tiempo de la función de transferencia de primer orden entre u_d y ω. De estas expresiones se obtiene el modelo no lineal.

$$\frac{d\omega}{dt} = -\frac{1}{\tau_{m_\omega}} \omega + \frac{K_m}{\tau_{m_\omega}} u_d$$

$$\frac{d\alpha}{dt} = \omega - \omega_{pared}$$

$$\frac{dd_n}{dt} = -(v - \omega y_A)\text{sen}(\alpha) - \omega x_A \cos(\alpha)$$

La 'velocidad de rotación de la pared' es una perturbación consistente en un pulso:

- Duración del pulso: $T_p = \frac{\pi r_{curva}}{v}$

- Amplitud del pulso: $\omega_{pared} = \frac{\pi}{T_p} = \frac{v}{r_{curva}}$ en rad/s

Estableciendo el punto de operación $u_d = 0$, $\omega = 0$, $\alpha = 0$, $v = v_o$ y $d_n = d_o$, se linealiza el modelo y queda:

$$\frac{d\Delta\omega}{dt} = -\frac{1}{T_m}\Delta\omega + \frac{K_m}{T_m}\Delta u_d$$

$$\frac{d\Delta\alpha}{dt} = \Delta\omega - \Delta\omega_{pared}$$

$$\frac{d\Delta d_n}{dt} = -x_A\Delta\omega - v_o\Delta\alpha$$

$$\frac{d}{dt}\begin{bmatrix}\Delta\omega\\ \Delta\alpha\\ \Delta d_n\end{bmatrix} = \overbrace{\begin{bmatrix}-\frac{1}{T_m} & 0 & 0\\ 1 & 0 & 0\\ -x_A & -v_o & 0\end{bmatrix}}^{\mathbf{A}}\begin{bmatrix}\Delta\omega\\ \Delta\alpha\\ \Delta d_n\end{bmatrix} + \overbrace{\begin{bmatrix}\frac{K_m}{T_m} & 0\\ 0 & -1\\ 0 & 0\end{bmatrix}}^{\mathbf{B}}\begin{bmatrix}\Delta u_d\\ \Delta\omega_{pared}\end{bmatrix}$$

En el modelo de seguimiento de pared, se tienen como variables de estado la velocidad angular de giro del vehículo, el ángulo de orientación con la pared y la distancia normal a la pared.

Finalmente, la función de transferencia $P(s)$ entre la tensión de los motores en modo diferencial u_d y la separación con la pared d_n puede definirse como el producto de la función de transferencia $P_1(s)$ entre la tensión diferencial y el ángulo con la pared y la función de transferencia $P_2(s)$ entre el ángulo con la pared y la distancia a la pared.

$$P(s) = \overbrace{\frac{K_m}{(1+T_m s)s}}^{P_1(s)} \times \overbrace{\frac{-v_o\left(1+\frac{x_A}{v_o}s\right)}{s}}^{P_2(s)} = \frac{-v_o K_m\left(1+\frac{x_A}{v_o}s\right)}{(1+T_m s)s^2}$$

Se puede ver en la expresión que la función de transferencia P(s) tiene un doble integrador. Esto complica la aplicación de un control PID, aunque al haber un cero negativo ($x_A > 0$) se suaviza este problema. En cambio, si cambia el signo de la velocidad de avance v_o, aparece un cero positivo que complica aún más el problema de control.

6.2 Seguimiento de pared recta

El primer objetivo es conseguir realizar una simulación en la que al vehículo se le indique mantener una cierta distancia con la pared. Para ello, se configuran los parámetros del código en MATLAB. En primer lugar se va al archivo CONFIG_CAR.m y dentro del apartado GENERAL CONFIGURATION se indica que se va a realizar una simulación con el vehículo en modo equilibrista. Se verifica que la variable de la identificación del modelo es "false".

```
CAR_IP = '172.20.10.8';
% RUN_MODE DEFINITION
% / 0. REAL-TIME SIMULATION / 1. FAST SIMULATION / 2. TEST VERIFICATION
% / 3. IMPLEMENTATION / 4. ALREADY DEPLOYED / 5. RPI SIMULATION
RUN_MODE = 1;
% TEST FOR MODEL IDENTIFICATION
IDENT_TEST = false;
% VEHICLE MODE
% / 0. CAR / 1. SELF-BALANCING VEHICLE
VEHICLE_MODE = 1;
```

Luego, en CONFIG_CONTROL.m, se indica que el modo de control será de seguimiento de pared en el apartado CONTROL MODE.

```
%% CONTROL MODE
%--------------------------------------------------------------------------
% / 0. OPEN LOOP / 1. FORWARD VELOCITY AND YAW ANGLE /
% / 2. FORWARD VELOCITY AND YAW RATE / 3. NAVIGATION /
% / 4. WALL FOLLOWER / 5. WALL FOLLOWING COMPETITION /
% / 6. NAVIGATION COMPETITION / 7. FORWARD VELOCITY COMPETITION /
% / 8. SELF-BALANCE
CONTROL.STATE.CONTROL_MODE = uint8(4);
```

Dentro del mismo archivo, se activa el control por realimentación de estados para el seguimiento de pared en el apartado CONTROL TYPES. También se aplicará este control a la velocidad de avance y al ángulo de cabeceo.

```
%% CONTROL TYPES
%--------------------------------------------------------------------------
% FORWARD VELOCITY CONTROL TYPE
% / 0. PID  / 1. STATE FEEDBACK CONTROL / 2, FIRST_ORDER DEAD BEAT
% / 3. SECOND-ORDER DEAD BEAT
CONTROL.STATE.FORWARD_VEL_CONTROL_TYPE = uint8(1);
…

%-------------------------------------------------------------
% WALL FOLLOWER CONTROL TYPE
% / 0. SINGLE LOOP  / 1. CASCADE / 2. STATE FEEDBACK CONTROL
CONTROL.STATE.WFL_CONTROL_TYPE = uint8(2);
%-------------------------------------------------------------
% PITCH ANGLE CONTROL TYPE
% / 0. PID  / 1. STATE FEEDBACK REGULATOR / 2. STATE FEEDBACK INTEGRAL
CONTROL
CONTROL.STATE.PITCH_ANG_CONTROL_TYPE = uint8(1);
```

Para comprobar que el vehículo sigue a la pared, ha de desplazarse por lo que se le ha de asignar una velocidad de avance. Esta velocidad se le puede dar con el mando de radiofrecuencia o con una referencia fija desde el ordenador. En el apartado 6.3 de la memoria se utilizará el mando, mientras que en este se le asignará de forma local. Esto se configura en el apartado REFERENCE DEFINITION.

```
%% REFERENCE DEFINITION
%-------------------------------------------------------------
% FORWARD VELOCITY REFERENCE TYPE
% / 0. CONSTANT / 1. PULSE / 2. SQUARE / 3. PRBS / 4. RAMP
CONTROL.STATE.FV_TARGET_TYPE = uint8(0);
%-------------------------------------------------------------
CONTROL.STATE.YR_TARGET_TYPE = uint8(0);
%-------------------------------------------------------------
CONTROL.STATE.YA_TARGET_TYPE = uint8(0);
%-------------------------------------------------------------
% WALL DISTANCE REFERENCE TYPE
% / 0. CONSTANT = 0.1 m / 1. PULSE / 2. SQUARE / 3. PRBS / 4. CURVE
CONTROL.STATE.WD_TARGET_TYPE = uint8(0);
%-------------------------------------------------------------
% FORWARD VELOCITY CONSTANT REFERENCE VALUE (0-4)
CONTROL.STATE.FV_TARGET_VALUE = uint8(1);
```

Nótese que la variable CONTROL.STATE.FV_TARGET_TYPE es la que indica el tipo de señal de referencia que se le da al vehículo: si se quiere un valor constante, un pulso, una señal cuadrada… En este caso se quiere una velocidad de avance constante, por lo que se le asigna un 0. Por otra parte, la variable CONTROL.STATE.FV_TARGET_VALUE asigna el valor de la referencia, es decir, que tan rápido se quiere que vaya el vehículo.

Una vez configurada la simulación del seguimiento de pared, se procede al diseño del controlador. Para ello, el método de diseño es la localización de los polos. En el apartado WALL FOLLOWER: STATE FEEDBACK REGULATOR de CONFIG_CONTROL.m Se encuentran las especificaciones de los parámetros que se han de editar hasta encontrar aquella combinación que consiga estabilizar el vehículo y siga las referencias de distancia a la pared.

```
%% WALL FOLLOWER: STATE FEEDBACK REGULATOR
%-------------------------------------------------------------
% WALL FOLLOWER SFR: SPECIFICATIONS
% Design method
% 1. Pole placement / 2. LQR
WFL_SFC.design_method = 1;
% Closed-loop wn / Open-loop wn
WFL_SFC.natural_freq = [5 5 5 5];
% Damping factor
WFL_SFC.damping_factor = [0.99 0.99 0.99 0.99];
% Third pole module / closed-loop wn
WFL_SFC.p_factor = [5 5 5 5];
```

Se tienen vectores debido a que existen cuatro puntos de operación. En este proyecto se modifica el primer valor de cada matriz debido a que se corresponde con

el punto de operación que se quiso especificar en el apartado OPERATING POINTS de CONFIG_CONTROL.m.

```
%% OPERATING POINTS (FORWARD VELOCITY)
%-----------------------------------------------------------------
FORWARD_VEL_OP = [0.1 0.3 0.4 0.5];
```

En primer lugar se comprueba la estabilización del vehículo y el seguimiento de pared con una referencia de distancia constante e igual a 0,1 m, y una referencia de velocidad de 0,1 m/s. Una vez los resultados sean satisfactorios se procederá a cambiar la referencia a un tren de pulsos y comprobar el ajuste de la separación del vehículo con la pared siguiendo los pulsos. En la Figura 34 se observa el éxito del control, que mantiene la distancia a la pared con una precisión milimétrica, a la vez que ajusta la velocidad de avance del vehículo a la referencia constante dada de 0,1 m/s y mantiene un ángulo de cabeceo, obviando las oscilaciones en t=3s, con un valor máximo de 4 grados. El diseño elegido del control es el que contiene los parámetros:

```
WFL_SFC.natural_freq = 5
WFL_SFC.damping_factor = 0.99
WFL_SFC.p_factor = 5
```

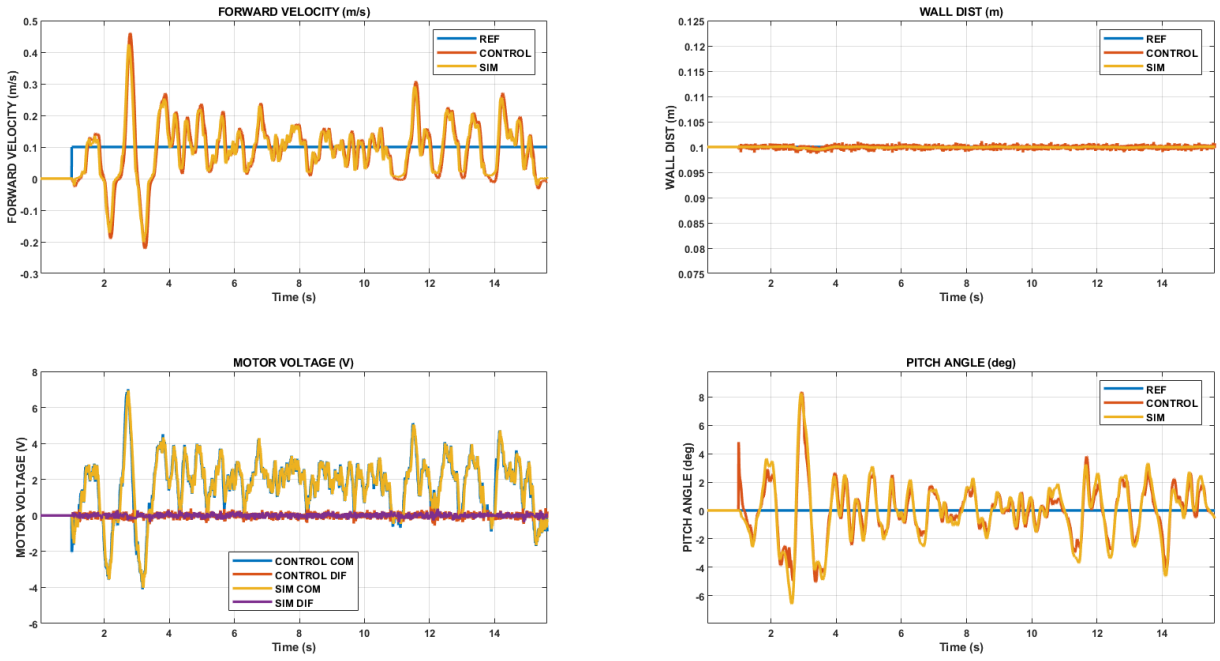


Figura 34: Simulación del control de seguimiento de pared con referencia constante y velocidad no nula

A continuación, se cambia el tipo de referencia de distancia a la pared. Puesto que queremos una velocidad de avance constante, con variaciones en la referencia de distancia a la pared para ver si el vehículo se ajusta a dichas variaciones (se aleja y se acerca de la pared), se le da una señal cuadrada como referencia. Para este control hubo que aumentar el valor de la referencia de velocidad de avance, puesto que

manteniéndola en 0,1 m/s no le daba tiempo a ajustar su distancia a la pared antes de que cambiase el valor de la referencia. Se probó con 0,2 m/s, consiguiendo que se estabilizase durante unos 10 segundos. Aumentando a 0,3 m/s de velocidad de avance fue como se consiguió estabilizar el control. En la Figura 35 se muestra el resultado final de la simulación, con un resultado muy satisfactorio pues se puede observar el seguimiento de la señal cuadrada de referencia de distancia a la pared.

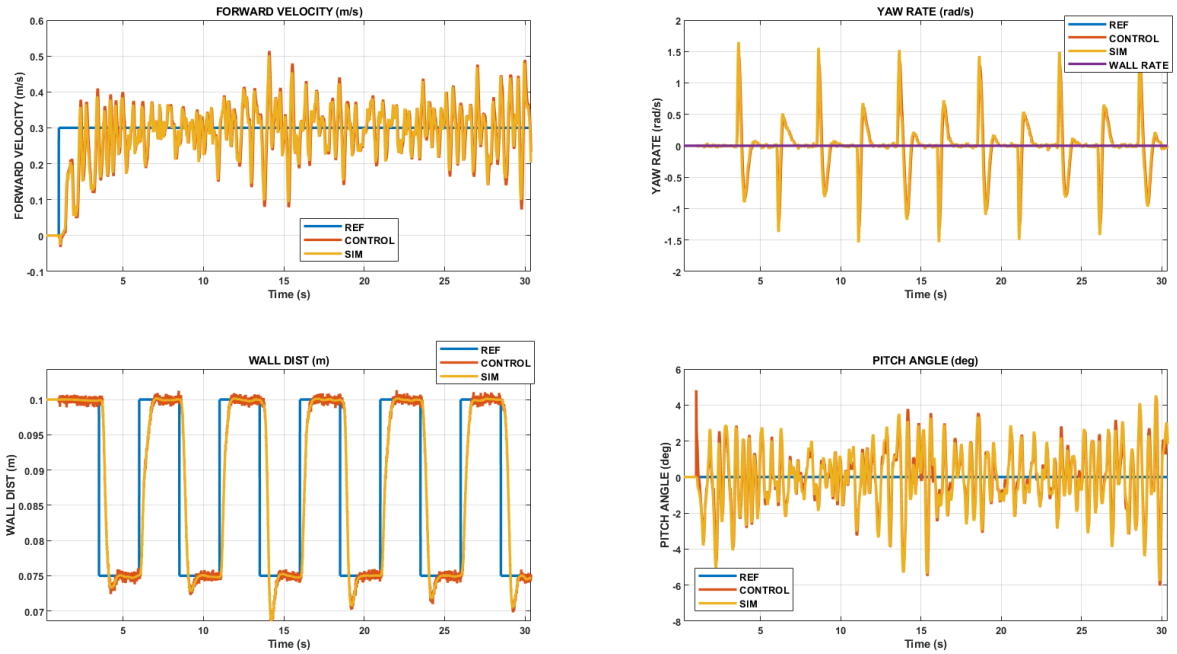


Figura 35: Simulación del control de seguimiento de pared con referencia variable

6.3 Seguimiento de pared en un circuito cerrado

El objetivo final de este proyecto es el de realizar una vuelta completa en el circuito cerrado del laboratorio de Control. El circuito añade una complicación al control de seguimiento de pared, ya que debe realizar las curvas al detectar una diferencia en el valor de los dos sensores de distancia que incorpora el vehículo.

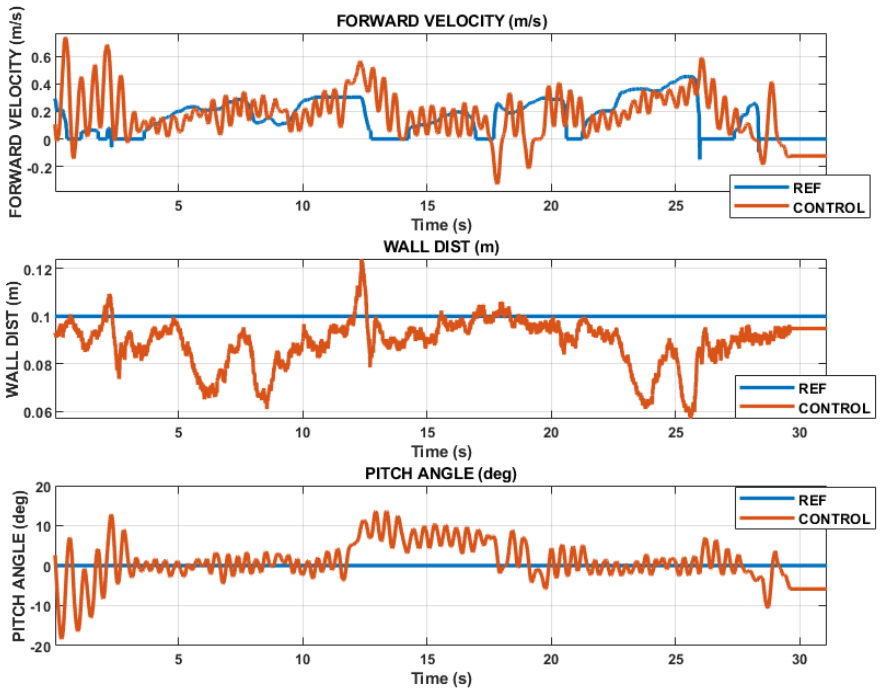
Figura 36: circuito cerrado del laboratorio

El código de control no ha de cambiar, pues es el mismo seguimiento de pared que el del apartado anterior, lo que cambian son las condiciones del ensayo. Se intenta poner en marcha el vehículo, pero este se inicia con una velocidad demasiado alta, por lo que cae. Tras un par de intentos, se decide utilizar el mando de radiofrecuencia para controlar la velocidad. Esto es muy útil si se pretende en el curso que se inicia en 2022 utilizar el vehículo en la asignatura de Control Digital, para realizar una competición de velocidad del control entre los alumnos. El ensayo consiste, pues, en el control automático de la distancia a la pared, que se fija en 10 cm, y la transmisión del valor de referencia de la velocidad de avance con el mando. Se configura el código para este fin, desde el apartado REFERENCE DEFINITIONS de CONFIG_CONTROL.m, estableciendo el mando como fuente de referencias para la velocidad de avance y de giro.

```
%-------------------------------------------------------------
% REFERENCE SOURCE
% / 0. LOCAL / 1. PC  / 2. RC JOYSTICK / 3. RC SWITCHES
CONTROL.STATE.MV_TARGET_SOURCE = uint8([0 0])';
CONTROL.STATE.FV_TARGET_SOURCE = uint8(2);
```

```
CONTROL.STATE.YR_TARGET_SOURCE = uint8(2);
CONTROL.STATE.YA_TARGET_SOURCE = uint8(0);
CONTROL.STATE.WD_TARGET_SOURCE = uint8(0);
```

El resultado final fue el recorrido completo del circuito sin chocarse, pudiendo regular la velocidad del vehículo. El resultado del ensayo se muestra en la Figura 37. El registro de datos se comienza cuando el vehículo se encuentra bajando la rampa. Se ve en la primera gráfica, correspondiente a la velocidad de avance, como se pone dicha referencia con el mando de radiofrecuencias a cero, momento en el que el vehículo abandona la rampa y es conveniente reducir la velocidad. En t =5s, el coche se encuentra a punto de tomar una curva de 180º, por lo que se aprecia entre t=5s y t=10s como la distancia a la pared se reduce, momento que se aprovecha para aumentar la velocidad de avance. En t=12s se aprecia como el vehículo se inclina hacia adelante. En este punto se encuentra subiendo la rampa. Luego, se vuelve a reducir la



distancia a la pared, al trazar la segunda curva del circuito.

Figura 37:Ensayo de seguimiento de pared realizando el circuito del laboratorio

7. Conclusiones, aportaciones y futuros desarrollos

7.1 Conclusiones

Algunas simulaciones no son completamente fiables. En el caso del control de avance, se consiguieron simulaciones para las que el vehículo se comportaba según el objetivo. Sin embargo, al implantar el control en el vehículo y realizar el ensayo, este no pudo mantener el equilibrio.

7.2 Aportaciones

El desarrollo de este proyecto ha sido la prueba previa a la implantación del nuevo vehículo equilibrista para las clases prácticas de Control Digital en la Universidad. Con él queda comprobado el buen funcionamiento de los controles diseñados.

En esta memoria se recopila toda la información relativa al vehículo equilibrista y será de ayuda para los alumnos que cursen Control Digital a partir del curso 2022/2023. Aunque ya existían documentos sobre el hardware o el software del robot, en este documento se aúna la información y sirve de guía para obtener, para cada coche del laboratorio, el diseño de los mismos controles.

7.3 Futuros desarrollos

El objetivo no cumplido de este proyecto fue la navegación autónoma del vehículo con el sistema de cámaras Optitrack, que podría ser objeto de siguientes Trabajos de fin de Grado. Además, podría ser de utilidad ampliar la utilización de la pantalla LCD del vehículo. Para este proyecto, únicamente se ha utilizado para obtener la dirección IP asignada al vehículo y comprobar la conexión del vehículo con el router. En un futuro, sería conveniente que se mostrasen por pantalla fallos de funcionamiento, como por ejemplo, un error de conexión con el ordenador. Otras futuros desarrollos interesantes serían la creación de una aplicación móvil, al igual que usan los vehículos de las marcas ELEGOO y OSOYOO, y la incorporación de una función de seguimiento de objetos. Se trataría de diseñar un control de distancia con el objeto delantero, de manera que el vehículo tenga velocidad nula cuando el objeto seguido se detenga, y velocidad de avance positiva cuando el objeto seguido se desplace.

8. Referencias

[1] Segway-Ninebot, «La historia de Segway-Ninebot,» [En línea]. Available: https://es-es.segway.com/about-the-brand.

[2] «Omeo technology,» [En línea]. Available: omeotechnology.com.

[3] OSOYOO, «OSOYOO Two Wheel Self Balancing Car Kit,» [En línea]. Available: https://osoyoo.com/2018/07/18/osoyoo-balancing-car/#Overview. [Último acceso: 20 09 2022].

[4] C. Jiménez Cortés, «Control de un vehículo equilibrista mediante una raspberry pi,» 2019.

[5] Devantech, «EMG30 mounting bracket and wheel specification,» [En línea]. Available: http://www.robot-electronics.co.uk/htm/emg30.htm. [Último acceso: 2022 8 26].

[6] MathWorks, «Programación en la Raspberry Pi con MATLAB y Simulink,» [En línea]. Available: https://es.mathworks.com/discovery/raspberry-pi-programming-matlab-simulink.html. [Último acceso: 27 08 2022].

[7] G. Arenciba Castellanos, F. E. Hernández Montero, J. Menéndez Álvarez, J. R. Rodríguez Suárez y A. Pérez Molinet, «Estimación de orientación, basada en filtro de Kalman, usando unidad de medición inercial sin magnetómetro,» *Resvista Investigación Operacional,* vol. 41, nº 3, pp. 369-378, 2020.

[8] J. L. Zamora Macho, «Proyecto 2 – Diseño de un control digital para un vehículo equilibrista».

[9] J. L. Zamora Macho, «Vehículo de dos ruedas con tracción diferencial,» 2022.

[10] C. Ladrón de Guevara Tapia, «Construcción y control de un vehículo equilibrista basado en una raspberry pi,» 2018.

[11] G. Welch y G. Bishop, «An introduction to the Kalman Filter,» University of North Carolina at Chapel Hill, 1995.

[12] Z. Alezones Campos, «Introducción al filtro de Kalman en robótica móvil,» Universidad de los Llanos, Villavicencio-Colombia, 2016.

[13] A. Becker, «Introducción al filtro de Kalman,» [En línea]. Available: Kalmanfilter.net. [Último acceso: 12 julio 2022].

[14] «EMG30, mounting bracket and wheel specification,» [En línea]. Available: https://www.robot-electronics.co.uk/htm/emg30.htm.

[15] D. Cubillo Llanes, «Navegación autónoma de un vehículo terrestre mediante una cámara lidar,» Madrid, 2022.

[16] Y. Gu y R. Ding, «A least squares identification algorithm for a state space,» Applied Mathematical Letters, 2013.

[17] J. L. Zamora Macho, *Vehículo equilibrista,* 2022.