Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Electrónica y Automática Industrial

Trabajo Fin de Grado

Plataforma motorizada de apoyo a la movilidad para niños con discapacidad motriz

Autor: Alessandro Melino Carrero

Tutor: Juan Carlos García García

2020

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Trabajo Fin de Grado

Plataforma motorizada de apoyo a la movilidad para niños con discapacidad motriz

Autor: Alessandro Melino Carrero

Director: Juan Carlos García García

Tribunal:

Presidente: Pedro Alfonso Revenga de Toro

Vocal 1: Almudena López Dorado

Vocal 2: Juan Carlos García García

Calificación: .....

Fecha: .....

A todas las personas que me acompañaron
y apoyaron durante esta etapa

Agradecimientos

> "Las cosas no se hacen siguiendo caminos distintos para que no sean iguales, sino para que sean mejores"
>
> *Elon Musk 21st*

Llegando al fin de esta etapa, alcanzando una de las metas que tengo desde pequeño, y que desde el primer año en la universidad se ve tan lejos, uno se da cuenta de lo importante que es el apoyo que te dan las personas de tu alrededor para poder seguir aguantando ante el esfuerzo y sacrificio que supone una ingeniería. Por tanto me gustaría agradecer a cada una de esas personas haber estado ahí cuando lo necesitaba.

En primer lugar agradecer a toda mi familia que me haya soportado día a día en mis etapas de mal humor y estrés pero que siempre me han animado a continuar adelante y calmarme. Sin ellos este camino hubiera sido mucho mas duro e incluso no habría llegado a su fin, así que de nuevo, mucha gracias.

No me puedo olvidar de mis compañeros de universidad y por suerte amigos, mis *Panas*, los cuales me han llenado de risas y buenos momentos estos cuatro años de universidad, haciéndose así mucho menos duros.

Gracias además a todos mis amigos, por estar ahí desde el principio, por distraerme en mis momentos de ocio y llenarme de fuerza para seguir adelante.

Por último, agradecer a Marta y a Juan Carlos por brindarme la oportunidad de empezar a trabajar con ellos, poder realizar este TFG y aprender todo lo que he aprendido durante estos dos años.

Autor

Alessandro Melino Carrero

Resumen

El propósito de este Trabajo de Fin de Grado (TFG) es desarrollar una nueva plataforma de ayuda a la movilidad destinada a niños, bajo el nombre de Kids Assistive TEchnology (KATE).

Bajo este propósito se utiliza como base para el desarrollo una plataforma ya creada en el proyecto SAR (ref: CCG2018/EXP-059). Dicho proyecto tiene por objetivo general favorecer la independencia del colectivo de personas con dificultades de movilidad graves. Estas personas podrán beneficiarse de avances ya consolidados en la robótica móvil, como es el caso de los sistemas de navegación autónoma o semi-autónoma en entornos estructurados, trasladados y adaptados a una silla de ruedas robotizada que incorpore, además, interfaces hombre-máquina apropiados.

A partir de la estructura principal de KATE, formada por un chasis mecánico y dos motores eléctricos, se diseña el sistema de bajo nivel que ponga en funcionamiento la silla y pueda ser controlada por el usuario mediante el uso de un joystick analógico.

Con el fin de alcanzar el propósito planteado se ha dividido el trabajo en las siguientes tareas: diseño del sistema de excitación y control de los motores de la plataforma, tanto a nivel hardware como a nivel software, desarrollo de la HMI mediante la cual el usuario controlará el sistema completo, implementación de un bus CAN de comunicaciones, con el fin de que en un futuro sea posible añadir nuevos módulos y prestaciones, capaces de comunicarse entre sí gracias a este bus y, por ultimo, prueba y evaluación del sistema completo en entornos similares al de los usuarios finales.

Keywords: Silla de Ruedas,Robótica,Plataforma Motorizada,Bus CAN,Nodos.

Abstract

The purpose of this final degree project is develop a new assisting motorized platform for kids, under the name of Kids Assistive TEchnology (KATE).

With this intention it is used as start point for the project another platform already created in the SAR project (ref: CCG2018/EXP-059). This project has as general objective improve the independence of people with serious problems of mobility collective. These people could enjoy the advances of mobile robotics already consolidated, like autonomous or semi-autonomous navigation systems in structured environments, adapted to a robotic wheelchair, which, in addition, have appropriated Human Machine Interface (HMI).

From the main structure of KATE, build by a mechanic chassis and two electric engines, it is design a low level system that turn on the wheelchair and could be controlled by the user with an analogical joystick.

For reaching the objectives described, the project has been divided in the following tasks: design the power and control of the engines of the platform,the hardware level as well as software level, carry out the HMI through which the user is able to control the whole system, implement a communication Controller Area Network (CAN) bus, in order to add new modules and resources in the future, being capable of communicate between the different nodes thank to this bus. Last of all, test and evaluate the complete system inside similar environments where the final users would use the platform.

Keywords: Wheelchair,Robotics,Motorized Platform,CAN bus,Nodes.

Resumen extendido

El propósito de este Trabajo de Fin de Grado (TFG) es desarrollar una nueva plataforma de ayuda a la movilidad destinada a niños, bajo el nombre de Kids Assistive TEchnology (KATE).

El trabajo parte de una estructura mecánica que soporta al usuario, en este caso una silla adaptada para niños diseñada previamente por la iniciativa Padrino Tecnológico [1] y es adaptada en este trabajo a modo de colaboración y trabajo paralelo a dicha iniciativa. El resultado final del montaje de la plataforma se puede ver en la figura 1.

Figura 1: Vista del prototipo KATE

A esta plataforma se le ha añadido un sistema de control distribuido basado en nodos interconectados a través de una red CAN. Cada nodo está formado por un módulo programable, capaz de desarrollar un conjunto dado de funciones. El sistema mínimo en KATE consta de dos nodos, que son el de motores y el del joystick. Su diseño es explicado a continuación de forma resumida.

El diseño electrónico del nodo motores debe estar orientado en torno a las funciones que debe cumplir:

- Encendido del sistema completo.
- Conectar todos los nodos del sistema distribuido a través de una red CAN.
- Recibir los pulsos de encoder y analizarlos.
- Excitar los motores.
- Reducir la tensión de 24 a 5 voltios.

Para realizar estas funciones se diseña primero el hardware necesario, se monta sobre una placa y se interconecta todo, desde los drivers de excitación de los motores, el hub de conectores RJ45 encargados del bus CAN de comunicación entre nodos y el sistema de lectura de pulsos de encoder.

Por otro lado se utiliza una tarjeta microcontroladora LPC2129 la cual se programa para poder captar los pulsos de encoder, mandarlos por el bus CAN y además excitar los motores. Este nodo se puede ver en la figura 2.

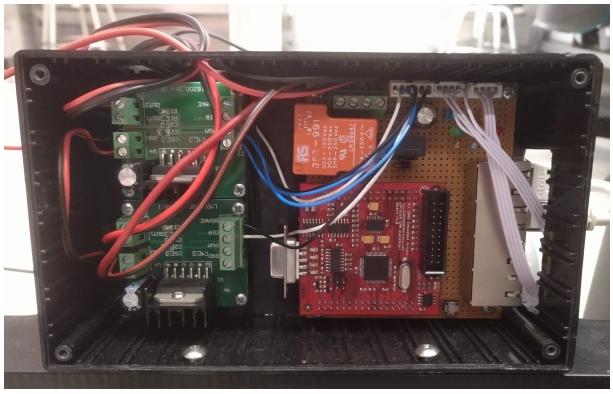

Figura 2: Hardware del nodo motores en su cajetín

En este nodo se implementa un controlador proporcional integrador de velocidad. Para ello se modela la planta a partir de los valores proporcionados por las hojas de datos de los motores, resultando la siguiente función de transferencia:

$$G_{planta} = \frac{2{,}688}{z - 0{,}8574} \tag{1}$$

Gracias a la herramienta Sisotool de Matlab se diseña el controlador apropiado con el perfil de velocidad deseado. Debido a que el controlador es proporcional integral, el error en régimen permanente sera nulo. Una vez obtenidas las constantes del controlador, se discretiza obteniendo la siguiente expresión:

$$H_z = 0{,}006 \cdot \frac{z - 0,5789}{z - 1} \tag{2}$$

La respuesta ante una entrada de tipo escalón se puede observar en la figura 3.

Este controlador se implementa en el código del microcontrolador y una vez obtenida la velocidad, se aplica a los drivers con la generación del PWM.

Toda plataforma móvil necesita una interfaz de control de la misma, tanto para realizar su encendido, como elegir entre modos de operación y ejecutar su movimiento. Ante esta necesidad se añade el nodo joystick, el cual es el punto central de la Human Machine Interface (HMI), desde donde el usuario puede controlar el sistema completo, de una manera sencilla y robusta. Las funciones que debe cumplir son:

- Orden de encendido del sistema.
- Selección de modo.
- Control de dirección y velocidad.
- Activación del claxon.
- Visualización del nivel de batería, encendido y modo.

En este nodo se realiza un circuito de acondicionamiento para adaptar las tensiones recibidas desde el joystick, up/down y left/right, para poder ser leídas por los conversores analógico-digitales del microcontrolador. El circuito es el mostrado en la figura 4.

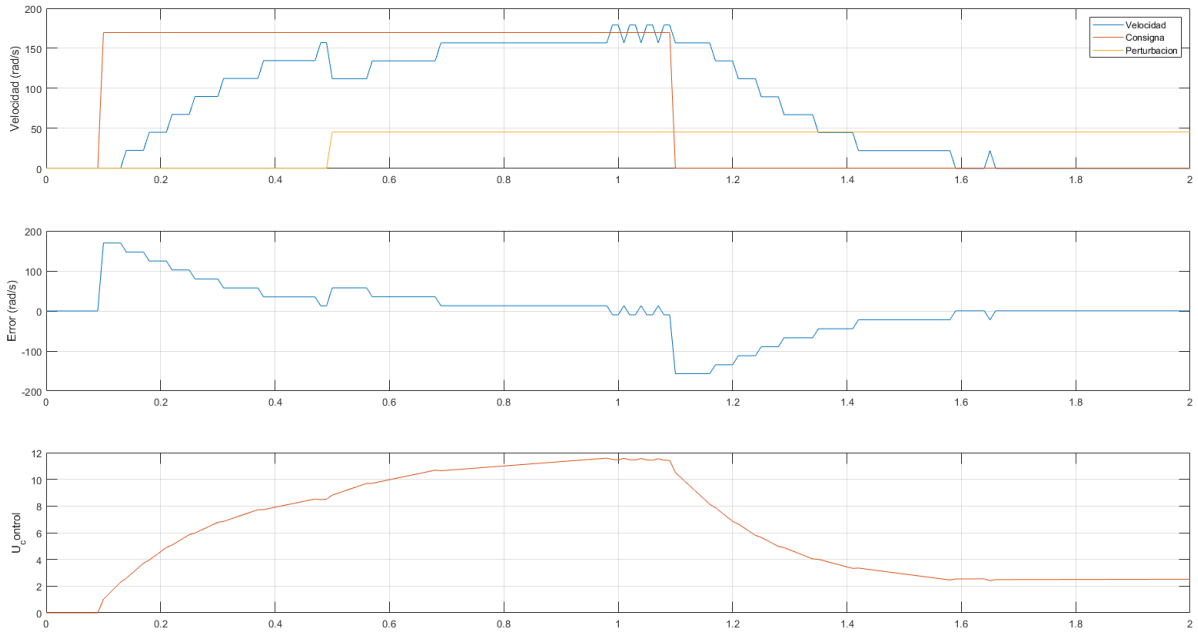

Figura 3: Medidas del sistema ante escalón con perturbación

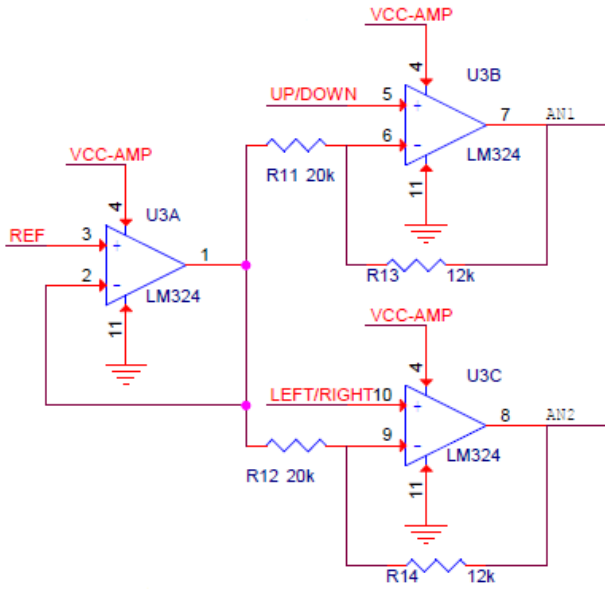

Figura 4: Circuito de acondicionamiento de las tensiones del joystick

También se desarrolla un circuito de limitación de velocidad mediante un potenciómetro, y se acondiciona de la misma forma que el joystick. Se utiliza un divisor resistivo y un amplificador operacional como se puede ver en la figura 5. Mediante este potenciómetro el usuario puede aumentar o disminuir la velocidad máxima de la plataforma.

Se integran dos pulsadores en el mando del joystick, uno para el encendido de la silla y otro para la activación del claxon. El resultado final del nodo joystick se puede observar en la figura 6.

Como prueba de funcionamiento final, se conecta la plataforma a un PC para poder muestrear las velocidades reales de los motores y con la ayuda de ROS.

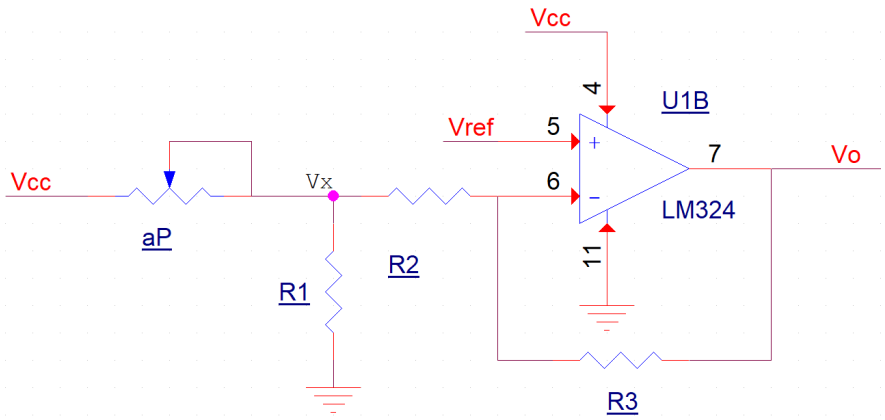

Figura 5: Esquemático del circuito de limitación de velocidad

Figura 6: Integración del nodo joystick en KATE

Como se puede observar en la figura 7 las velocidades aumentan de forma gradual y disminuyen de forma mas brusca, como consecuencia de la rampa de aceleración programada.

Por otro lado se ve que la rueda izquierda circula a mayor velocidad que la derecha, para evitar la desviación de la plataforma hacia la derecha, por un defecto de fabricación del prototipo.

Por ultimo se muestra una prueba real de funcionamiento de la plataforma en el siguiente vídeo:

Prueba de navegación con joystick de la plataforma KATE: https://youtu.be/Vtv6oAWaBnA.

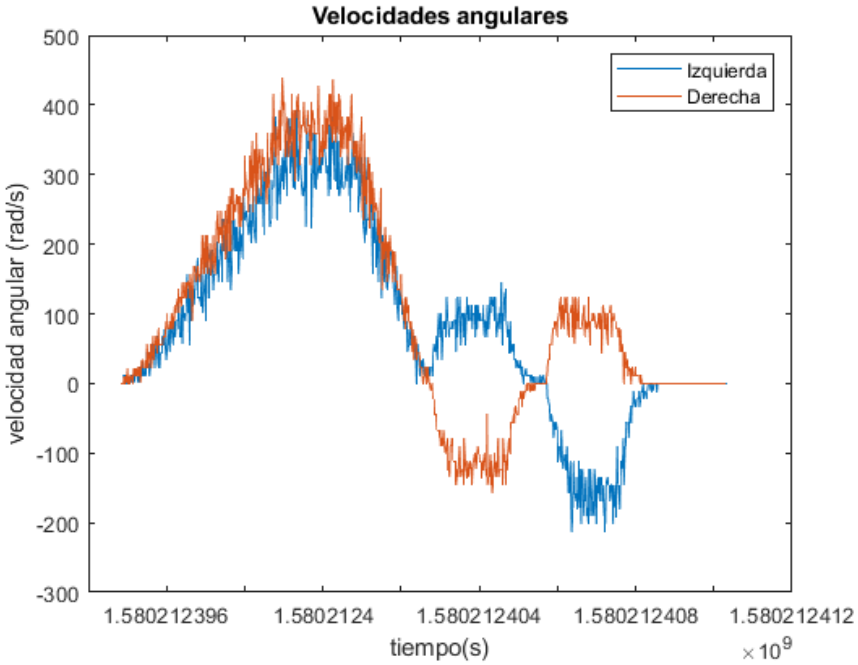

Figura 7: Observaciones de las velocidades de la rueda izquierda y derecha leídas gracias a la interfaz con un PC y ROS

Índice general

Índice de figuras

Índice de tablas

Lista de acrónimos

ADC	Analog to Digital Converter.
CAN	Controller Area Network.
GPIO	General Purpose Input Output.
HMI	Human Machine Interface.
KATE	Kids Assistive TEchnology.
LED	Light Emitting Diode.
PC	Personal Computer.
PI	Proporcional Integrador.
PWM	Pulse Width Modulation.
ROS	Robot Operating System.
SAR	Sistema de Ayuda a la movilidad Robotizado.
SARA	Sistema de Ayuda Robotizado y Avanzado.
SPNO	Single Pole Normally Opnen.
TFG	Trabajo de Fin de Grado.
ZOH	Zero Order Hold.

Capítulo 1

Introducción

1.1 Contexto del trabajo

El concepto de robótica se ha consolidado en nuestras vidas como algo indispensable y que nos facilita todo tipo de tareas. Durante estos últimos años ha experimentado un gran crecimiento y adquirido mucha importancia, tanto dentro del mundo industrial como en la robótica de servicios.

Bajo este último propósito parte el objetivo del trabajo, el cual pretende mejorar la calidad de vida y favorecer la independencia de personas con dificultades de movilidad graves. Con este objetivo se desarrolló tiempo atrás otra plataforma de ayuda a la movilidad denominado Sistema de Ayuda Robotizado y Avanzado (SARA), objeto de muchos trabajos [2], [3], [4], [5], [6], [7], [8], que tienen como objetivo principal el dotar a la silla de ruedas de un sistema de navegación autónomo y de distintos métodos de entrada para controlarla.

Figura 1.1: Silla de ruedas SARA perteneciente al proyecto SAR

SARA, como se puede ver en la figura 1.1, es una silla de ruedas que ha sido modificada para facilitar su uso a personas que presentan dificultades de movilidad. Se incorporaron dos motores con respectivos encoders, un joystick analógico (el cual por medio de programa también presenta opción de uso digital),

un sistema de conducción por soplo, una cadena de sensores de ultrasonidos, destinados a evitar colisiones, y un sistema de navegación autónoma por medio de un PC.

Dicho sistema fue objeto del Trabajo de Fin de Grado "Módulos software para un sistema de asistencia a la movilidad basado en entorno ROS"[9], el cual implementó el sistema de navegación autónomo basado en odometría y detección de obstáculos usando la tecnología desarrollada bajo ROS [10].

Toda esta arquitectura sirve como punto de partida para este Trabajo de Fin de Grado, donde tal arquitectura será replicada en una plataforma motorizada destinada a niños que presentan dificultades de movilidad.

Básicamente, se usa la estructura de nodos que presenta SARA, implementando un nodo para los motores y encendido de la silla y otro nodo para el joystick y administración de los modos de funcionamiento. Dichos nodos se comunican por medio de un bus CAN. Presenta dos modos de funcionamiento fundamentales: conducción por medio del joystick y un sistema de navegación autónoma que implementó Javier de la Roda en SARA [9]. Este sistema de navegación es instalado en una Raspberry PI4 que se usa de modo similar a un PC.

1.2 Definición de objetivos

El trabajo parte de una estructura mecánica que soporta al usuario, en este caso una silla adaptada para niños, y todo el hardware necesario para la implementación del sistema de bajo nivel. Cabe destacar dentro de la plataforma mecánica la existencia, como dispositivos de sensado hardware, de dos encoders de efecto Hall acoplados a los ejes del motor.

Figura 1.2: Plataforma móvil GRAM perteneciente a Padrino Tecnológico

Toda la estructura base, la cual se puede ver en la figura 1.2, ha sido diseñada previamente por la iniciativa Padrino Tecnológico [1], y es adaptada en este trabajo a modo de colaboración y trabajo paralelo a dicha iniciativa.

Sobre esta plataforma se incorporan dos drivers (módulos hardware) encargados de excitar los motores de continua (24V) de la plataforma mecánica, en función de las señales PWM recibidas desde un microcontrolador, incorporado para realizar tareas de control del movimiento. En dicho microcontrolador se implementará un controlador PI con el fin de regular las velocidades lineales y angulares del vehículo.

La tarjeta microcontroladora se ocupa, además, de analizar la información de comando proporcionada por un joystick analógico con el que el usuario puede controlar el movimiento de la silla. La posicion del joystick se convierte a velocidades en los motores con los drivers correspondientes.

Así, la plataforma básica, objeto de este TFG, constará de dos nodos principales:

- Nodo joystick: formado por un joystick analógico, interruptores de encendido, bocina y selector de modo, un potenciómetro para fijar la velocidad máxima y un display que muestra el nivel de la batería de la plataforma.
- Nodo motores: formado por los motores, sus encoders, drivers y la tarjeta controladora.

Sin embargo, la naturaleza modular del sistema, permite incorporar fácilmente prestaciones avanzadas de navegación autónoma al sistema KATE mediante la incorporación de nodos adicionales, como los siguientes:

- Nodo PC: formado por un ordenador, donde se instalará un núcleo ROS (Robot Operating System). Este sistema operativo será el encargado de controlar la conducción autónoma de la plataforma.
- Nodo sensores: formado por los sensores precisos para poder obtener información del entorno, por ejemplo, para una conducción reactiva del sistema. Este nodo deberá incluir además una tarjeta microcontroladora para administrar la información de los sensores y la enviará a los nodos que la precisen a través del bus CAN.

A continuación, en la figura 1.3 se representa mediante un diagrama de bloques la estructura de un sistema completo, incluyendo además de los dos nodos del sistema básico (motores y joystick) otros nodos correspondientes a ampliaciones futuras, como los señalados anteriormente (PC y sensores):

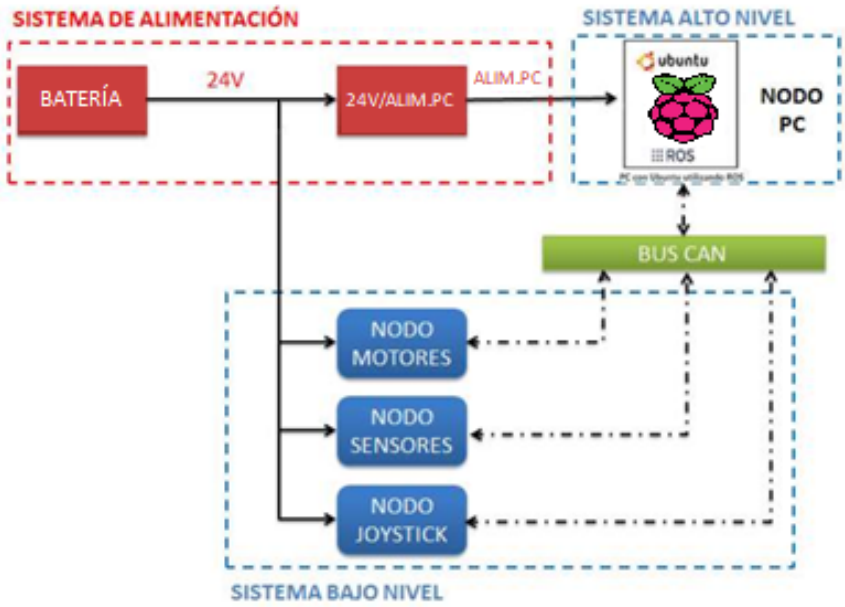

Figura 1.3: Diagrama de bloques de los nodos que forman parte de KATE

La comunicación entre los distintos nodos (motores y joystick) se realiza mediante un bus CAN. Se elige este bus dado que es fiable, robusto, permite intercambiar gran cantidad de información a alta velocidad y es usado ampliamente dentro del ámbito industrial y en automoción.

1.3 Plan de trabajo

A continuación, se describen las diferentes tareas necesarias del proceso que se va a realizar para completar el trabajo propuesto.

- Se diseñará el controlador PI encargado de regular los motores de la plataforma, además de las funciones software necesarias para conectar el controlador con los sensores de velocidad (encoders) y los actuadores de la misma (drivers). El algoritmo encargado de realizar esta función se codifica en la tarjeta microcontroladora situada en el nodo motores.
- El bus CAN se usará para que el nodo joystick envíe periódicamente las consignas de velocidad correspondientes. Para ello se medirán los valores (analógicos) extraídos de la posición del joystick y se convertirán a valores digitales mediante el ADC de la tarjeta microcontroladora de este nodo. Además, también se desarrolla el software correspondiente al cambio de modo de funcionamiento (control directo por joystick o avanzado desde el PC).
- A nivel hardware, se diseñarán, realizarán y documentarán apropiadamente los circuitos de ambos nodos. Para reducir su coste en el prototipo, cada nodo usa módulos microcontroladores comerciales adaptados mediante tarjetas diseñadas ex-profeso para poder desarrollar las funciones especificadas para cada uno de los nodos.
- Se adaptarán y probarán sobre cada módulo el código fuente preciso para los microcontroladores de cada uno de los nodos. Dicho código se adapta a las necesidades de KATE e incluye la gestión de la comunicación CAN entre los nodos. Para ello se usa como guía los trabajos anteriores indicados en la introducción de este anteproyecto.
- Se prueba la plataforma al completo para comprobar su funcionamiento en distintas situaciones que se pueden dar en el uso cotidiano.
- Todo esto es documentado y será incluido en la memoria final del trabajo.

Capítulo 2

Estructura mecánica

2.1 Introducción

Este capítulo se explican las partes que componen la estructura mecánica sobre la que se realiza el montaje del hardware de la plataforma KATE. Dicha estructura ha sido obtenida gracias a la colaboración con la iniciativa Padrino Tecnológico.

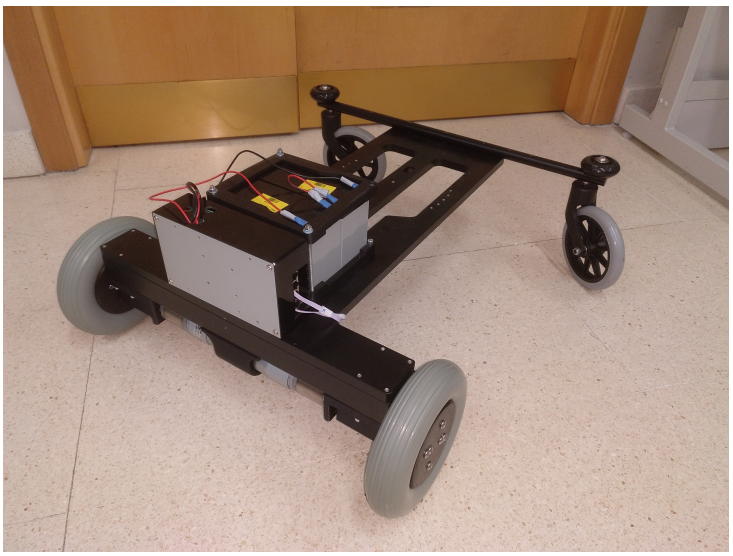

Figura 2.1: Estructura mecánica base de KATE

Dicha estructura consta de las siguientes partes:

- Chasis.
- Baterías.
- Ruedas.
- Motores y encoders.
- Piezas 3D.

2.2 Chasis

El chasis de la plataforma, como se puede observar en la figura 2.1, está formado principalmente por tres partes, una pieza central donde se distribuyen todos los elementos de la silla de ruedas y dos piezas transversales a la misma.

La pieza central es una placa de aluminio debidamente mecanizada para poder asegurar todos los elementos que formaran la silla, tales como el asiento del usuario, las baterías, los cajetines con la electrónica necesaria y las piezas transversales. Además está diseñada de una forma que sea lo suficientemente resistente para que pueda soportar el peso del usuario y de las baterías, pero a su vez ligera, con el fin de que no suponga un consumo extra de energía al moverse.

Por otro lado, las piezas transversales son barras de aluminio destinadas a la sujeción de las ruedas. La barra delantera es un tubo hueco, al cual se le han añadido unos protectores en los extremos para evitar posibles daños en caso de colisión. Además presenta la opción de añadir reposapiés para el usuario. La barra trasera esta destinada a portar todo el sistema de motores, encoders y ruedas motrices, por tanto es algo más grande y su superficie es lisa.

2.3 Baterías

Como sistema de alimentación de toda la plataforma se opta por el uso de dos baterías de 12 V y 7.2 A h por celda. Por tanto el sistema de alimentación proporciona 24V a toda la plataforma por medio del bus CAN, como se detalla en el capítulo 3.

Debido a que son muy pesadas, son colocadas en el centro de masas de la silla, para que no la desestabilicen. Además son amarradas con un set de piezas 3D, las cuales se analizan posteriormente en la sección 2.6.

Para conocer el estado de carga de las baterías, se dispone de un sistema integrado en el modulo del joystick, el cual indica el nivel de energía disponible. Si el nivel de carga es bajo, se debe realizar la carga mediante un cargador especializado en este tipo de baterías, debido a que no es recomendable que se sobrecarguen o permanezcan con muy baja carga, porque se degradan si permanecen descargadas mucho tiempo.

2.4 Ruedas

Como sistema de desplazamiento de la plataforma, se utilizan un total de 4 ruedas, dos delanteras y dos traseras, de las cuales las motrices serán las traseras.

Las ruedas traseras son adaptadas para poder ser usadas en la plataforma. Se diseña un set de piezas 3D especificas para ellas, eliminando la llanta original de fabrica y utilizando este set de piezas como llanta. Como son ruedas motrices, deben estar debidamente fijadas al eje de los motores, para conseguir la correcta transmisión de movimiento. Presentan una cierta ventaja respecto a las ruedas de SARA, debido a que no necesitan aire, por lo que no se produce error de movimiento debido a la presión, como ocurre con la otra plataforma.

Las ruedas delanteras son ruedas locas destinadas a permitir el movimiento en cualquier dirección. Sin embargo, presentan la desventaja que desvían la silla en el arranque, si no se encuentran colocadas en la dirección deseada. Una vez situadas en la dirección de avance, no presentan ningún problema adicional.

Con esta disposición de ruedas motrices, KATE presenta un sistema cinemático diferencial, descrito en el capítulo 6.

2.5 Motores y encoders

Los encargados de accionar las ruedas y conseguir que la plataforma se desplace son los motores. Esta plataforma presenta unos motores de corriente continua alimentados a 24V que pertenecen a la serie PG42775 de Twirl Motor.

Son motores pequeños pero lo suficientemente potentes para mover la plataforma con una carga semejante a una persona de poca edad. Debido a su escaso tamaño, son fácilmente integrables en la plataforma. Presentan una reductora de engranajes planetarios con una relación de reducción de 1:50.9.

Se usa un acoplador para transmitir el movimiento del eje del motor a las ruedas y queden correctamente alineados sin que se produzca excentricidad en el movimiento.

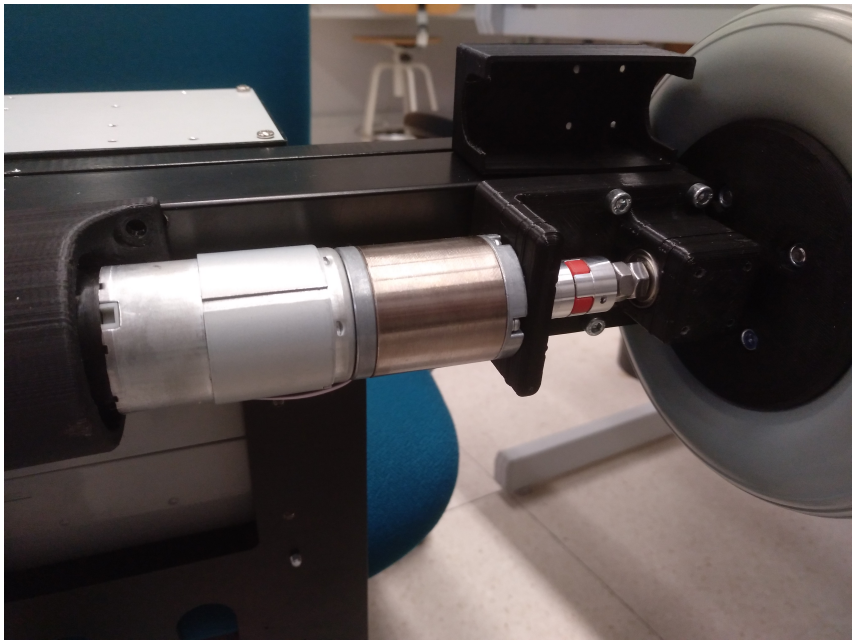

Figura 2.2: Montaje del motor en la parte trasera de KATE

Conectados a los motores se instalan dos encoder magnéticos de efecto Hall de la serie ME-775, de dos canales. Dichos encoder proporcionan 7 pulsos por revolución.

Con todos estos datos se puede conocer la máxima resolución de los encoder que se puede obtener, como se comprueba en la expresión 2.1, dado que se tienen 2 canales en el encoder, y se pueden usar 2 flancos por cada pulso que proporcione cada canal. Teniendo en cuenta que cada canal tiene una resolución de 7 pulsos por vuelta, hay un total de 28 flancos por vuelta del motor.

$$Res_{mot} = N_{canales} \cdot N_{flancos} \cdot ppr_{enc} = 2 \cdot 2 \cdot 7 = 28 flancos/vuelta \tag{2.1}$$

En función de la rueda, se debe tener en cuenta la reducción que ofrece la reductora, llegando así a los 1425,2 flancos por vuelta de la rueda.

$$Res_{rueda} = N_{red} \cdot Res_{mot} = 50{,}9 \cdot 28 = 1425{,}2 flancos/vuelta \tag{2.2}$$

Estos cálculos resultan útiles a la hora de realizar el control sobre los motores en el capítulo 3.

2.6 Piezas 3D

Se diseña un conjunto de piezas 3D para integrar de forma correcta y robusta elementos como los motores, las baterías o las ruedas en el chasis de la plataforma.

El primer set de piezas corresponde a la sujeccion de las baterías. Consta de dos piezas, una inferior donde apoyan las baterías y una superior con la cual se sujetan. Mediante unos tornillos pasantes se atornillan ambas piezas y las baterías quedan bloqueadas en su interior, evitando su desplazamiento. Además, dispone de un espacio suficiente en la parte superior para poder conectar los cables de alimentación.

El segundo set de piezas corresponde a las ruedas traseras. Se imprimen una serie de discos, previamente diseñados en un programa de modelado, como se puede ver en la figura 2.3, para poder adaptar las ruedas traseras a los ejes de los motores, dado que el formato en el que se proporcionan no es valido para su conexión. Para ello se desmontan los neumáticos de la llanta original y se utilizan las piezas como llantas adaptadas a esta aplicación. Mediante una serie de tornillos pasantes se sujetan al eje de la rueda.

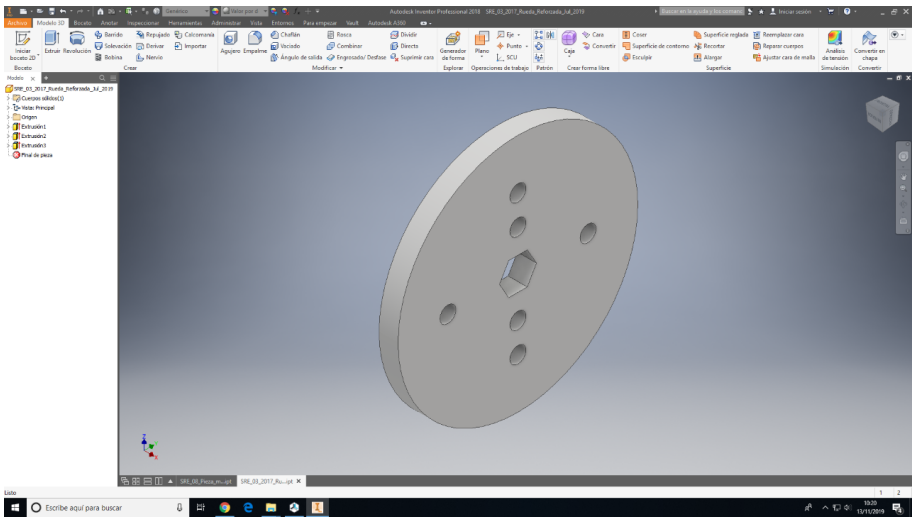

Figura 2.3: Diseño de una de las piezas 3D de las ruedas en Inventor

El tercer set de piezas es utilizado para sujetar y proteger los encoder, el cableado y el sistema de transmisión de los motores a las ruedas. Consta de un total de 5 piezas como se puede observar en la figura 2.2:

- Una pieza central de protección del cableado y los encoders.
- Dos piezas laterales, una a cada lado, como sujeción del sistema de transmisión y el eje de la rueda.
- Dos piezas laterales, una a cada lado, como protección del sistema de transmisión.

Capítulo 3

Nodo motores: nivel físico

3.1 Introducción

En este y los siguentes capítulos, se detallará la arquitectura de nodos de la plataforma, tanto a nivel físico (hardware) como lógico (software). A nivel global, la estrcutura mínima de esta plataforma consta de solo dos nodos, motores y joystick, que serán los que se detallen en este TFG. No obstante, gracias a la estructura modular y al uso de un bus CAN de comunicaciones se pueden añadir más módulos y funciones sin más que añadir nuevos módulos (nodos), como por ejemplo un nodo PC o sensores de entorno, como los incluidos en SARA.

Dichos nodos se encuentran interconectados con cable de pares del mismo tipo que el de red Ethernet, mediante el cual se alimentan y se comunican, formando así un sistema robusto, portable y fácilmente ampliable.

En la figura 3.1 se puede comprobar de forma esquemática el sistema de nodos de la plataforma.

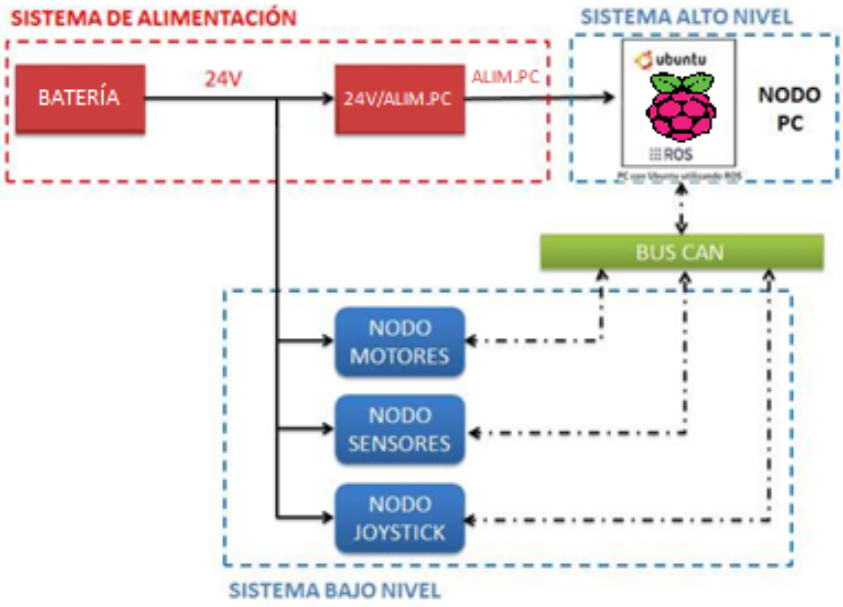

Figura 3.1: Diagrama de bloques de los nodos que forman parte de KATE

3.2 Alimentación a 24V mediante cable Ethernet

Dentro de la plataforma se usa cable Ethernet con conectores RJ45, debido a que son estándares y fácilmente enchufables. Básicamente estos cables se usan para integrar el bus CAN, en la alimentación de los diferentes nodos y otras señales indispensables en el funcionamiento de la plataforma.

Sin embargo, se podría usar otro tipo de conector y cable específico para alimentar el ordenador, como los conectores tipo DIN o similar. Este tipo de conector es roscado y por tanto menos sujeto a accidentes y desconexiones, pero su montaje es más complicado, ya que se d eben soldar hilo a hilo, además de ser menos versátil que un RJ45. Como estos conectores son específicos para alimentación, en su datasheet indica que la corriente por conexión puede conducir hasta 5 A, que alimentando a 24 V, se pueden llegar a alcanzar prácticamente los 120 W para alimentar nodos dentro del mismo bus.

La capacidad para transferir energía de los conectores RJ45 usados en la plataforma es algo menor. Cada una de las conexiones puede llegar a conducir entre 1 A y 1.5 A como máximo, aunque se puede duplicar usando cables en paralelo, en caso de ser necesario.

En el peor de los casos, conduciendo 1 A y alimentado a 24 V, la potencia máxima por conexión son 24 W.

Uno de los dispositivos que se va a conectar a la plataforma que más consume es la Raspberry PI4, con un consumo máximo de 15 W (alimentación de 5 V, con 3 A como máximo), lo que limita al resto del bus a consumir menos de 9 W. Para ello se debe realizar un análisis nodo a nodo para determinar la potencia que consume.

Por otro lado, también se debe tener en cuenta la resistencia equivalente del cable Ethernet. El usado en KATE es de categoría 5, lo que implica una resistencia de $0.19\,\frac{\Omega}{\mathrm{m}}$.

Esto es un parámetro a considerar, ya que si se calcula la potencia que disipa cada metro de cable se obtiene:

$$P_{cable} = I_{max}^2 \cdot R_{cable} = 1^2 \cdot 0{,}19 = 0.19\,\frac{\mathrm{W}}{\mathrm{m}} \tag{3.1}$$

Se estima que se usan entre 2 y 3 metros de cable Ethernet, por lo que la disipación máxima a causa del cable alcanzaría los 0.54 W, quedando 5.5 W para alimentar el resto de nodos.

Si se realizan los mismos cálculos en el mejor de los casos, donde la corriente máxima que soporte el conector RJ45 sea 1.5 A, las cifras indican que la potencia total transferible es de 36 W, donde la Raspberry consume 15 W y restan 11 W para el resto de nodos. A esto hay que añadir el efecto del cable, donde se consumen $0.405\,\frac{\mathrm{W}}{\mathrm{m}}$ y estimando la longitud de cable usada se disipan hasta 1.2 W.

La conclusión obtenida tras este análisis es que usando conectores RJ45, la plataforma no va a tener problemas de alimentación en ninguno de los nodos, incluso con la Raspberry y los motores conectados al mismo tiempo.

3.3 Diseño electrónico del nodo

El diseño electrónico del nodo debe estar orientado en torno a las funciones que debe cumplir. Por ello se tiene que tener claro qué funciones debe tener el nodo motores, las cuales se listan a continuación:

- Encendido del sistema completo.
- Conectar todos los nodos CAN.

- Recibir los pulsos de encoder y analizarlos.
- Excitar los motores.
- Reducir la tensión de 24 a 5 voltios.

El encendido de la silla funciona de la siguiente forma: el usuario debe presionar un pulsador con enclavamiento situado en el mando del joystick. Dicho pulsador cortocircuita una línea del bus (la cual se ha denominado START en los esquemáticos) con masa. Esa línea esta conectada en el nodo motores con uno de los terminales de la bobina de un relé, mientras que el otro terminal de la bobina esta conectado a la batería. La diferencia de potencial que se produce en la bobina cuando se cierra el circuito, hace que el relé cierre el contacto, permitiendo el paso de energía desde la batería hasta otra línea del bus, denominada +24VB. Esta línea del bus es la encargada de alimentar todos los nodos de la plataforma, tras presionar el botón de encendido.

Si se vuelve a presionar, se deshace la conexión a masa de la bobina del relé, por tanto el relé se abre y no permite el paso de corriente, realizando el apagado automático de la silla.

Los componentes electrónicos necesarios que se han utilizado para esta especificación son un relé normalmente abierto de un solo polo (SPNO), correctamente dimensionado para evitar problemas ante corrientes de conmutación. Se añade un diodo de protección entre terminales de la bobina, para evitar corrientes de retorno, además del pulsador con enclavamiento anteriormente comentado.

Otro punto fundamental en el diseño del circuito de los nodos es reducir la tensión de alimentación de 24 V a 5 V, debido a que la mayoría de elementos que se usan en el nodo funcionan alimentados a dicha tensión.

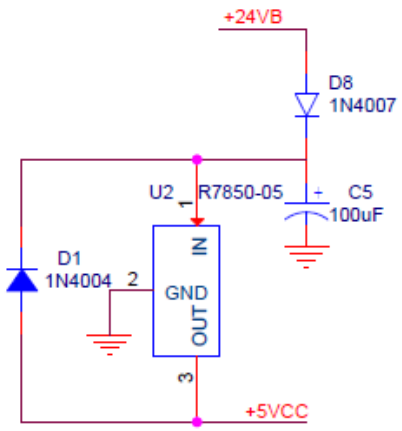

Figura 3.2: Esquemático del circuito para reducir la tensión

Así como se puede ver en la figura 3.2 se utiliza un convertidor DC/DC R-785.0-0.5, cuyo rango de tensiones de entrada varia desde 4.75 V a 32 V y cuya salida son 5 V suministrando una corriente de 0.5 A, siendo totalmente apto para la aplicación. Este convertidor es conmutado en lugar de lineal para reducir pérdidas. Además se añade un diodo de protección entre los terminales de entrada y salida, tal y como se pide en la hoja de características.

Por último, se pretende evitar problemas con la alimentación, dado que están conectados en la misma línea elementos que consumen bastante energía, así como los motores o en un futuro el PC. Con dicha finalidad se añaden el diodo D8 y el condensador C5. El condensador se carga y proporciona energía cuando un gran consumo de otros elementos provoque una caída de tensión en el cable y compromete el

funcionamiento del nodo. El diodo D8 se encarga de evitar que la energía de C5 se fugue hacia el resto de nodos. La tensión de salida del regulador es utilizada para alimentar el microcontrolador y los encoders.

En cuanto a la conexión de todos los nodos, se utiliza un hub de conectores RJ45, al cual se pueden enchufar y desenchufar fácilmente el cable Ethernet usado en la plataforma, permitiendo una gran modularización del sistema. De esta forma se pueden añadir elementos nuevos al sistema, pudiendo ser alimentados y comunicarse gracias a la presencia de este centro de conexiones.

Otras de las necesidades que debe suplir el nodo motores es recibir, filtrar y enviar los pulsos de los encoders.

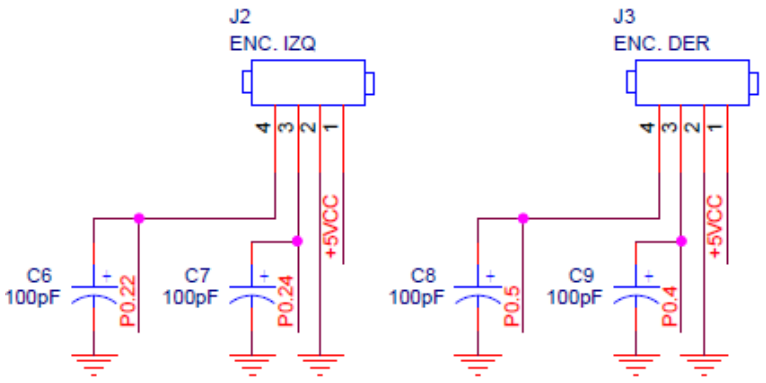

Figura 3.3: Esquemático de los conectores de los encoders

Para ello se dispone de dos conectores hembra de 4 pines, mediante los cuales se alimentan los encoder y se toman los pulsos de sus dos canales. Como se puede ver en la figura 3.3, se añade un condensador por cada canal debido a que experimentalmente se consiguen mejores resultados a la hora de realizan la lectura de los pulsos, puesto que sirven como filtros ante ruidos en la señal. Dicha señal ya filtrada se conecta a los pines del microcontrolador. Cómo se analizan estos pulsos, se explica posteriormente en el apartado 4.

Referente a la excitación de los motores, son utilizados dos módulos drivers diseñados en la universidad, los cuales presentan toda la electrónica necesaria para usarlos directamente. El centro funcional del driver es el puente en H de referencia LMD18200.

En este trabajo se ha usado un conector de 6 pines para realizar todas las conexiones entre el microcontrolador y los drivers de forma integrada, como se puede ver en la figura 3.4. Se debe realizar la conexión de las entradas DIR, PWM, GND y BRAKE. Se usan dos líneas controlables por el micro para BRAKE, dos líneas controlables por el micro para DIR, una línea conectada a VDD de 3.3 V para PWM y una línea a masa para conectarlo con GND. El uso de los pines del micro, se explica posteriormente en el apartado 4.

Se han añadido 4 resistencias con la finalidad de realizar el pull-up de las entradas DIR y BRAKE en el encendido de la plataforma. Esta decisión se toma porque cada vez que se encendía la silla se producía el accionamiento de los motores hasta que se fijaba el valor de ciclo de trabajo de PWM al 50 % (motores parados). Con la inclusión del pull-up, se asegura que dichos pines se encuentran a nivel alto hasta que el microcontrolador se encarga de ellos. De esta forma, según la tabla de verdad del LMD18200, se cortocircuitan los terminales de salida del driver, bloqueando los motores y evitando que se desplace la plataforma en el encendido.

Por otro lado, se usan 3 LED para señalizar que se consiguen los niveles correctos de tensión. Dichos LED están conectados a la toma de alimentación de 24 V procedente del bus, al terminal de salida del

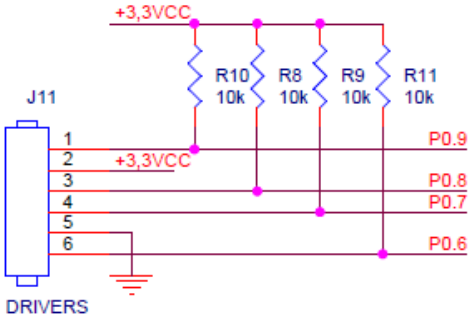

Figura 3.4: Esquemático del a los drivers

convertidor DC/DC, para conocer que se está regulando de forma correcta la tensión y, por último, al terminal ON proveniente del bus. Dicha señal indica que el microcontrolador del nodo joystick se encuentra operativo y señala que se puede continuar con el resto del encendido.

Por último, queda por introducir el núcleo del nodo, el microcontrolador, el cual se va a encargar de recibir la información de los encoder y fijar el ciclo de trabajo del PWM para conseguir la velocidad deseada.

El microcontrolador utilizado es el LPC2129 CAN QuickStart desarrollado por Embedded Artists. Dicho microcontrolador se puede ver en la figura 3.5.

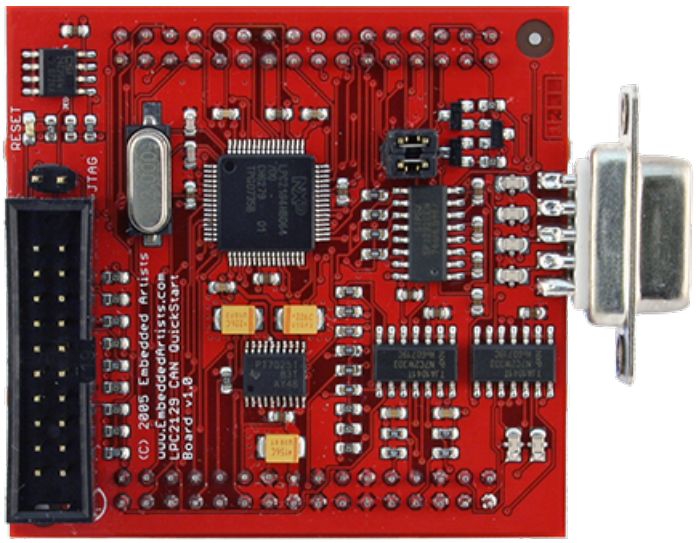

Figura 3.5: Microcontrolador LPC2129 CAN QuickStart de Embedded Artists

Este mismo microcontrolador es utilizado en cada uno de los nodos de SARA, y debido a su alta versatilidad, presencia de interfaces CAN y facilidad de programación, es elegido también para ser utilizado en este trabajo.

Dentro del nodo motores, se va a utilizar el interfaz CAN, un timer, dos módulos PWM y ciertos puertos de entrada y salida. Es integrado en el nodo por medio de dos zócalos y además se incluye un pulsador sin enclavamiento para poder realizar el reinicio del microcontrolador de forma externa, útil cuando se reprograma éste.

Las conexiones que se realizan con el resto de dispositivos del nodo son las siguientes:

- Pines P0.8 y P0.6 como GPIO conectados a las entradas BRAKE del driver.

- Pines P0.7 y p0.9 como PWM conectados a las entradas DIR del driver.
- Pines P0.4, P0.5, P0.24 y P0.28 como GPIO conectados a los canales de los encoders.
- Salida digital VDD de 3.3 V conectada a PWM del driver.
- Pines CANH-0 y CANL-0 conectados al bus CAN.

Una vez conocido el conexionado de todos los dispositivos, se describe el código del microcontrolador para que cumpla con los objetivos propuestos.

3.3.1 Control de los motores

3.3.1.1 Diseño del controlador PI

A continuación se explica cómo funciona el controlador PI implementado para regular la velocidad angular de los motores, ante posibles perturbaciones y pares que se produzcan. Dicho controlador se ha diseñado y simulado con la ayuda de MATLAB y Simulink, y se ha integrado en el microcontrolador LPC2129.

Para poder diseñar el regulador primero se debe modelar la planta. Optamos por un modelode orden 1, como el dado en la expresión 3.2. Siendo A la constante de velocidad del motor y τ la constante de tiempo del motor. Para identificar las constantes se usa la hoja de características del motor RS-775244500.

$$G_{planta} = \frac{A}{\tau \cdot s + 1} \tag{3.2}$$

La constante de τ se obtiene su valor experimentalmente, resultando ser 65 ms. Por otro lado,el valor de A responde a la expresión 3.3.

$$A = \frac{\omega_{nominal}}{V_{nominal}} = \frac{4500 \cdot 2 \cdot \pi}{60 \cdot 24} = 187.5\,\mathrm{rad/s \cdot V} \tag{3.3}$$

Este valor es teórico. Experimentalmente se comprueba que se ajusta mejor a $180\,\mathrm{rad/s \cdot V}$. Con la planta modelada se puede obtener su equivalente discreto mediante el método ZOH, obteniéndose la función de transferencia 3.4.

$$G_{planta} = \frac{2{,}688}{z - 0{,}8574} \tag{3.4}$$

Una vez definida la planta del motor que se quiere controlar, se debe diseñar el controlador. En la expresión 3.5 se puede observar la forma del controlador PI que se desea integrar.

$$H_z = K \cdot \frac{z - c}{z - p} \tag{3.5}$$

El valor del polo es 1, ya que se desea un regulador PI (proporcional integral). El valor del cero se obtiene experimentalmente. La restricciones de diseño correspondientes indican que la salida del sistema no tenga un sobreimpulso elevado y sea lo suficientemente rápido. Posteriormente también se obtiene el valor de K con el mismo fin. Una vez con el valor de la ganancia proporcional y del cero calculados, se obtiene la función de transferencia del controlador 3.6.

$$H_z = 0{,}006 \cdot \frac{z - 0{,}5789}{z - 1} \tag{3.6}$$

Por último, se añaden los elementos no lineales, tales como saturación, cuantificación de la consigna y la cuantificación de los encoder. En la figura 3.6 se representa el sistema completo diseñado con todos sus elementos.

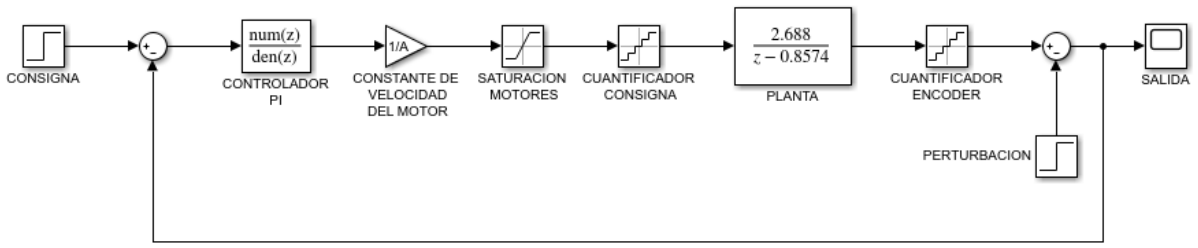

Figura 3.6: Modelo del sistema completo a simular

Para comprobar el correcto funcionamiento del sistema se simula aplicando una entrada escalón de magnitud inferior a la velocidad máxima de los motores ($W_{max} = V_s \cdot A = 226\,\mathrm{rad\,s^{-1}}$, suponiendo alimentación de 12 V) sin perturbación y posteriormente con perturbación.

Como se puede observar en la figura 3.7, la medición de velocidad presenta muy poca resolución debido a los pocos pulsos que proporcionan los encoders, aunque gracias a la baja velocidad a la que van a circular es posible realizar control en régimen permanente.

Al introducir el controlador la constante de tiempo del sistema aumenta hasta aproximadamente 240 ms y alcanza el régimen permanente en 500 ms.

Los resultados al aplicar el mismo escalón, pero con perturbación, se pueden ver en la figura 3.8.

El sistema responde correctamente, corrigiendo el error producido por la perturbación y alcanzando el valor de la consigna, aunque el sistema prácticamente satura, ya que la máxima tensión que se le podría aplicar a los motores es 12 V, en el caso bajo estudio.

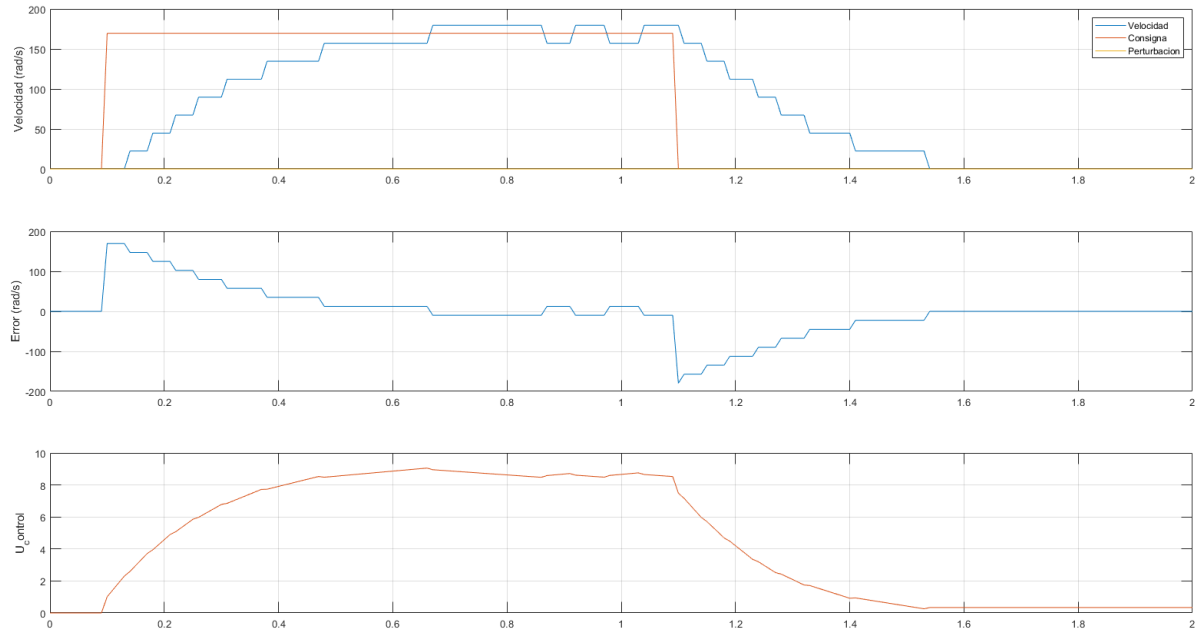

Figura 3.7: Medidas del sistema ante escalón sin perturbación

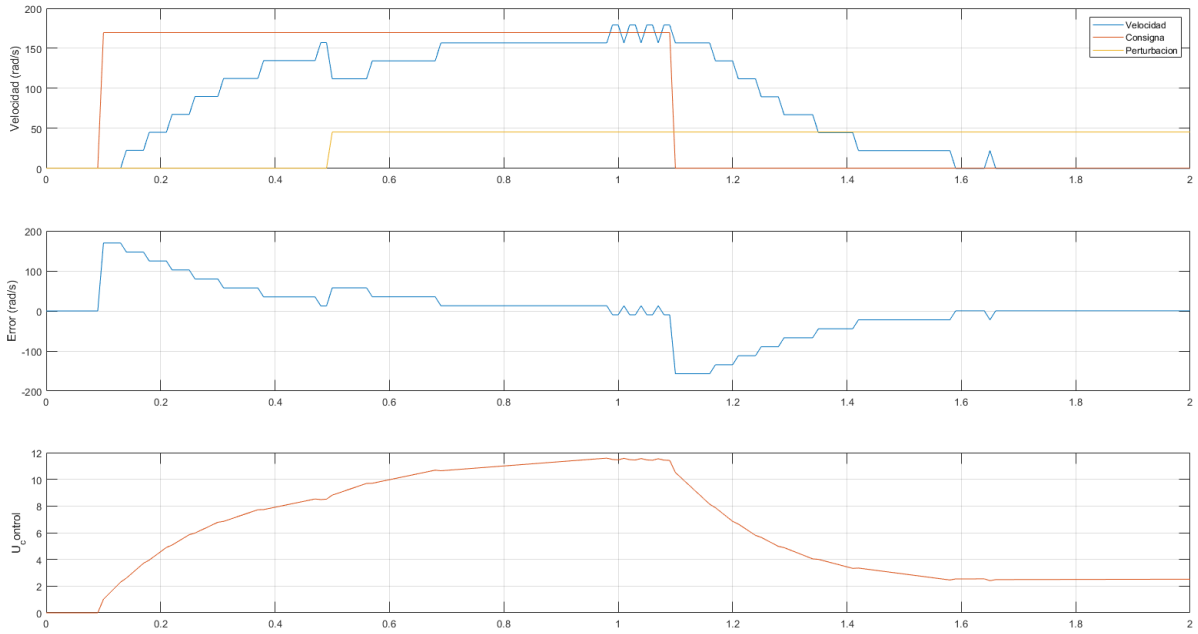

Figura 3.8: Medidas del sistema ante escalón con perturbación

3.3.1.2 Diseño de anti-Windup

Con el fin de evitar y reducir este problema de saturación, se introduce un compensador Anti-Windup, como aparece en la figura 3.9.

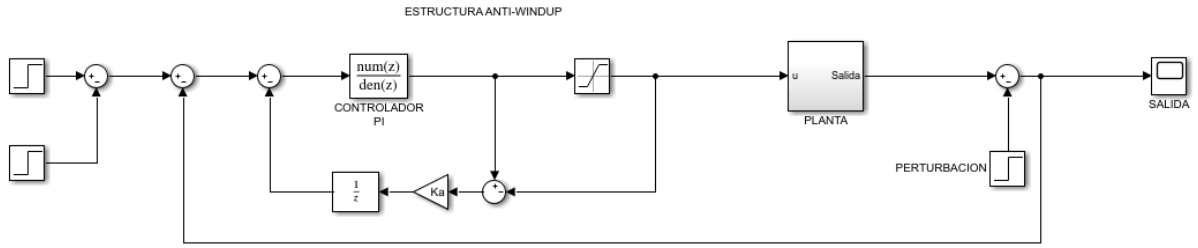

Figura 3.9: Modelo del sistema con anti-windup

Con el sistema diseñado, tan solo falta obtener el valor de la constante K_a, de manera que se reduzca la saturación, cumpliendo el objetivo de diseño. Finalmente,tras varias iteraciones, el mejor valor de K_a es 10, obteniendo como resultado final del control la gráfica 3.10.

Al aplicarse la perturbación, el nivel de la entrada de la planta asciende para mantener el valor de la velocidad requerido por la consigna, pero a diferencia de la simulación anterior el valor de la entrada a la planta es menor, reduciendo así la posibilidad de llegar a saturar el sistema y que responda de forma no deseada.

3.3.1.3 Implementación

Como última parte de la implementación del controlador, queda verificar el funcionamiento sobre los motores reales y así comprobar lo visto en la simulación. El código del microcontrolador es el presentado en el fragmento 4.10.

Con objeto de analizar los datos de velocidad y control en Matlab, se utiliza el puerto serie del

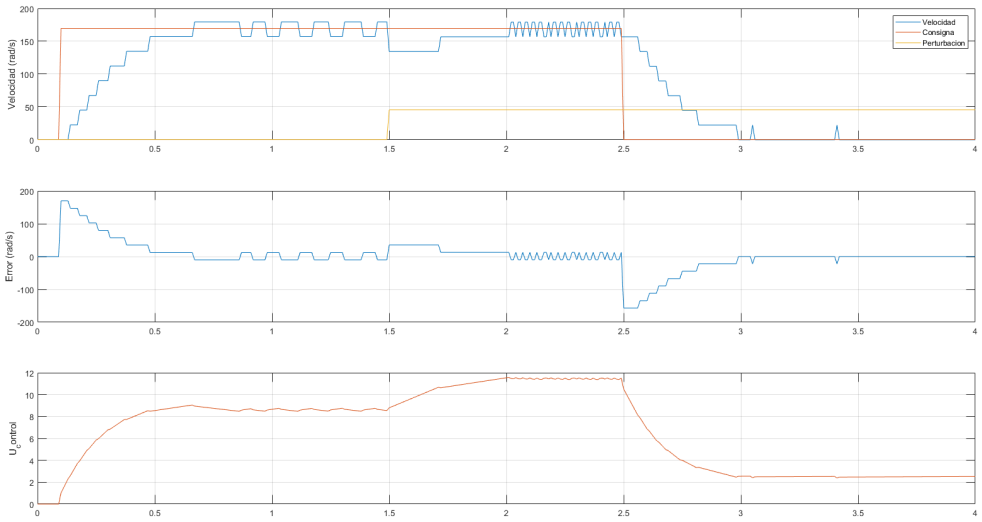

Figura 3.10: Medidas del sistema con anti-windup

microcontrolador y un conversor serie-USB para enviar estos datos al PC. En este punto se realizan diferentes pruebas para analizar el comportamiento de los motores con la implementación del controlador PI.

3.3.1.4 Medición de velocidad sin aplicar par resistente

Lo primero que se realiza es aplicar una consigna de velocidad máxima y de la mitad de su valor, en intervalos de tiempo de 2 segundos, sin aplicar carga, para observar si el motor responde correctamente y cuál es su constante de tiempo de forma experimental. En la figura 3.11 se presenta la velocidad del motor izquierdo, en color rojo junto con la consigna deseada, en color azul. Como era de esperar, la resolución de las medidas es muy baja, a causa de los pocos pulsos proporcionados por el encoder, pero se puede apreciar que el motor sigue correctamente a la consigna. Cabe comentar que esta es la velocidad del motor y no la de la rueda, debido a que se cuenta con una reductora la cual comunica la rueda con el motor, por lo tanto, la velocidad de la rueda será mucho menor a la que se puede ver en las figuras.

La constante de tiempo del sistema es 265 ms. Se ha visto modificada ya que ahora se trabaja con el motor real. Además se han incluido dos condensadores para filtrar los pulsos que se reciben y así evitar picos indeseados en la medición, lo que conlleva a que la constante de tiempo también aumente. En este punto se deben analizar otros parámetros como el error o el valor de tensión que se está aplicando.

En cuanto al valor de la consigna que se le aplica al motor, el cual se puede estudiar en la figura 3.12, llega a su máximo cuando la consigna de velocidad es máxima y es proporcional al valor de velocidad deseado. Se estudia su comportamiento más adelante, cuando se le aplique carga al motor.

Por último, el error de velocidad es la diferencia entre la consigna y la velocidad real del motor. Como se puede observar en la figura 3.13 se producen picos de error cuando el motor intenta seguir a la consigna, pero el error en régimen permanente es nulo, por lo que la funcionalidad del controlador implementado es correcta.

Igualmente, se muestran las figuras respectivas a la velocidad (3.14), la consigna de tensión (3.15) y el error del motor derecho (3.16). Los resultados obtenidos del motor derecho son idénticos al izquierdo, confirmando el funcionamiento en ambos motores.

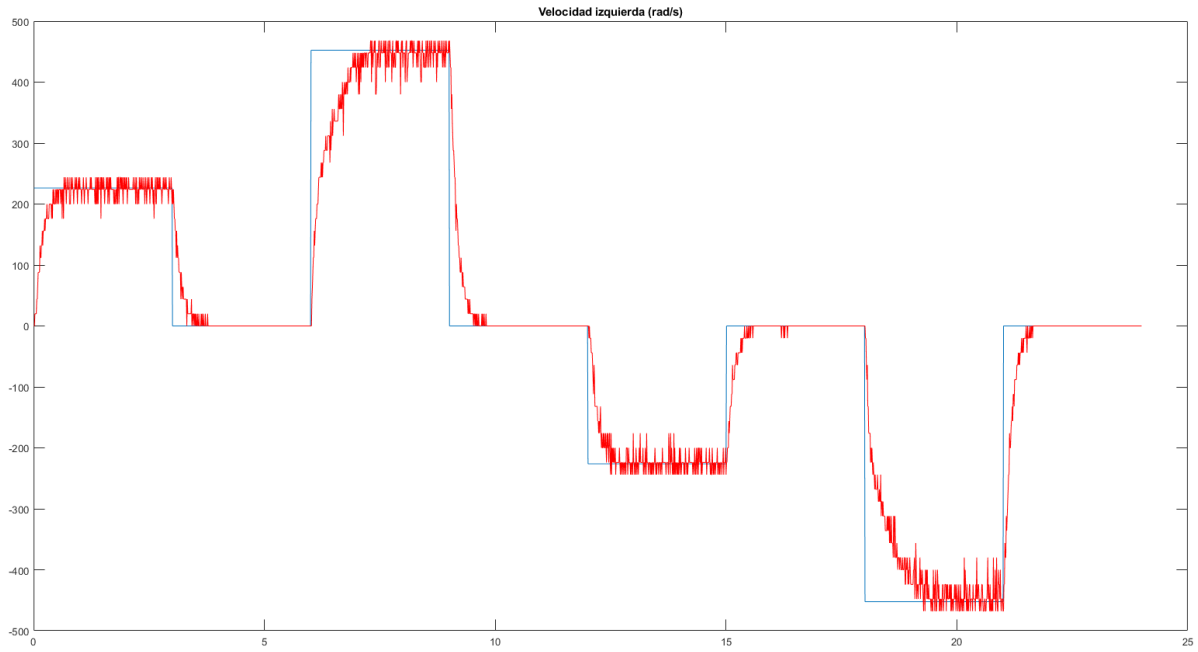

Figura 3.11: Medida de velocidad de la rueda izquierda

3.3.1.5 Medición de velocidad aplicando par resistente

En este apartado se aplicará una consigna de velocidad del mismo valor, pero durante distinto tiempo, para poder observar el comportamiento de los motores cuando se produce una perturbación, tal como un aumento de carga en la rueda. Se realizará de forma manual, por lo que el valor de la perturbación no será idéntico, pero servirá para observar el comportamiento del sistema en distintos casos.

En la prueba de la figura 3.17 durante los primeros 18 segundos de la muestra no se ha aplicado ningún tipo de carga, mientras que durante los 6 últimos siguientes se ha intentado bloquear la rueda.

El resultado observado es que al ser la consigna el valor máximo de velocidad, se ve afectada por la carga, ya que disminuye la velocidad, porque el sistema esta saturando. Esto se puede comprobar en la figura 3.18, donde se puede observar que se está aplicando el máximo valor de consigna, correspondiente a 12 V, entonces al aplicarle un par externo, se le exige aumentarlo, pero al ser inviable, la velocidad se ve reducida, porque se está alcanzando la saturación.

Estudiando el motor derecho, se comporta de igual manera, satura cuando la consigna es máxima, pero se realiza el control correctamente cuando la consigna es menor y no se eleva hasta alcanzar la saturación. De hecho se puede comprobar como se aplica algo de carga en el intervalo de 12 a 18 segundos, donde la consigna de tensión varia y se corrige la velocidad del motor, en las figuras 3.19 y 3.20.

Para contrastar toda esta información, se realiza el estudio del comportamiento del sistema en lazo abierto, donde se reflejarán las diferencias entre el sistema con realimentación y sin realimentación. Se han realizado básicamente las mismas pruebas del estudio de la velocidad de ambos motores con y sin perturbación.

En la figura 3.21 se puede observar como la velocidad sigue a la referencia sin ningún problema, debido a que no existe ningún tipo de perturbación que frene el motor. Sin embargo, si se estudia con detalle la figura, se puede observar que la constante de tiempo se ha visto modificada a aproximadamente 600 ms. En este punto se ve un aspecto negativo del sistema en lazo abierto, ya que se vuelve bastante más lento que el sistema realimentado.

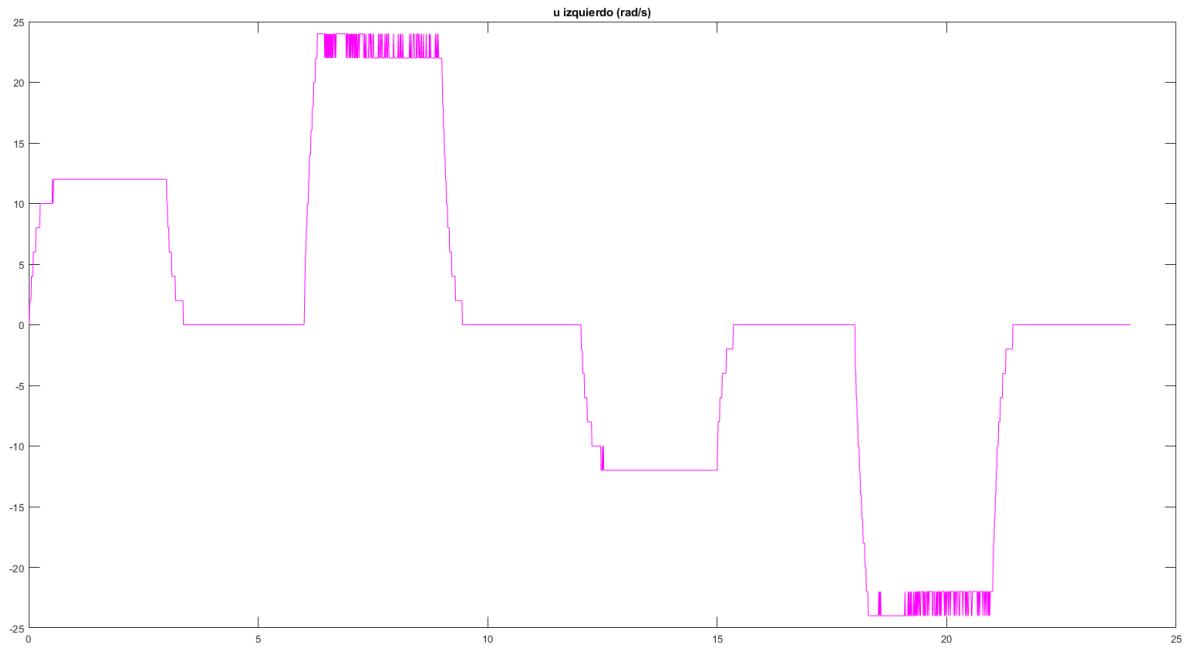

Figura 3.12: Medición de la tensión que se está aplicando al motor izquierdo

En la figura 3.22 se puede observar el comportamiento del sistema en lazo abierto y aplicándose una perturbación. Como en el ejemplo anterior, durante los primeros segundos no se aplica ningún tipo de carga, pero finalizando la prueba se puede observar como la velocidad prácticamente se hace nula, sin corregirse. Los cambios de velocidad que se observan son a causa de que la perturbación no se puede aplicar de forma uniforme, ya que se realiza de forma manual. Se repiten las mismas pruebas con el motor derecho, obteniéndose las figuras 3.23 y 3.24.

Como conclusión de este apartado se ha comprobado que el comportamiento del sistema en lazo cerrado es mucho peor que en lazo abierto, es mas rápido y además corrige los errores ante perturbación. De esta forma queda diseñado e implementado el controlador PI que regula el comportamiento de los motores de la plataforma.

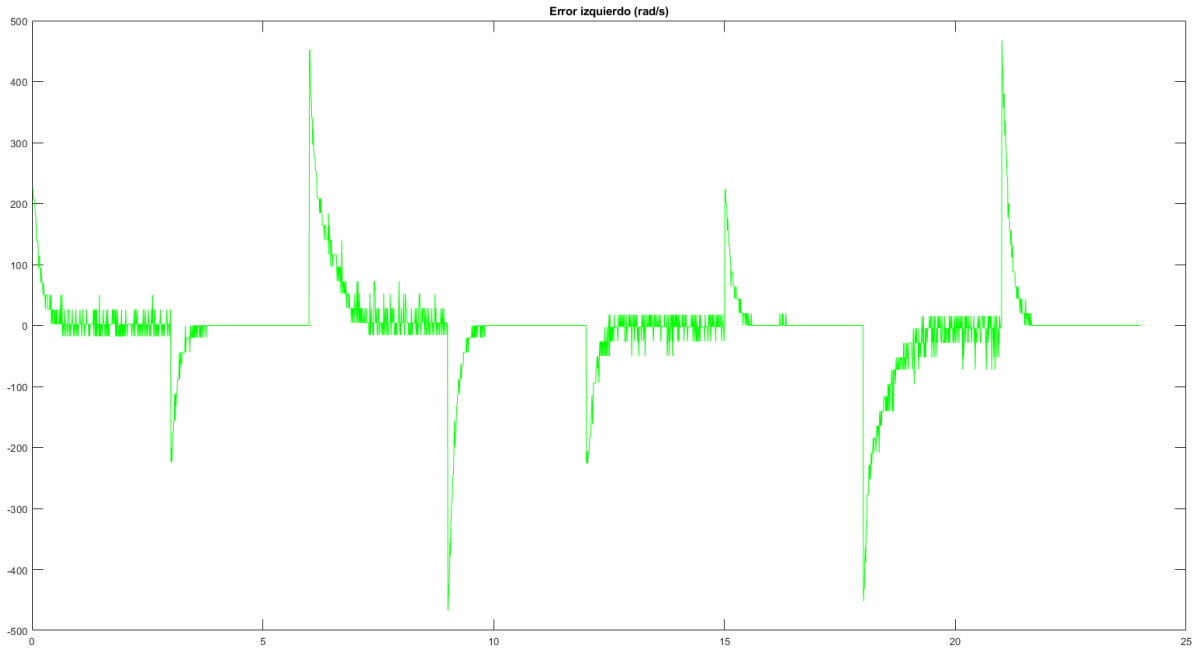

Figura 3.13: Error del motor izquierdo

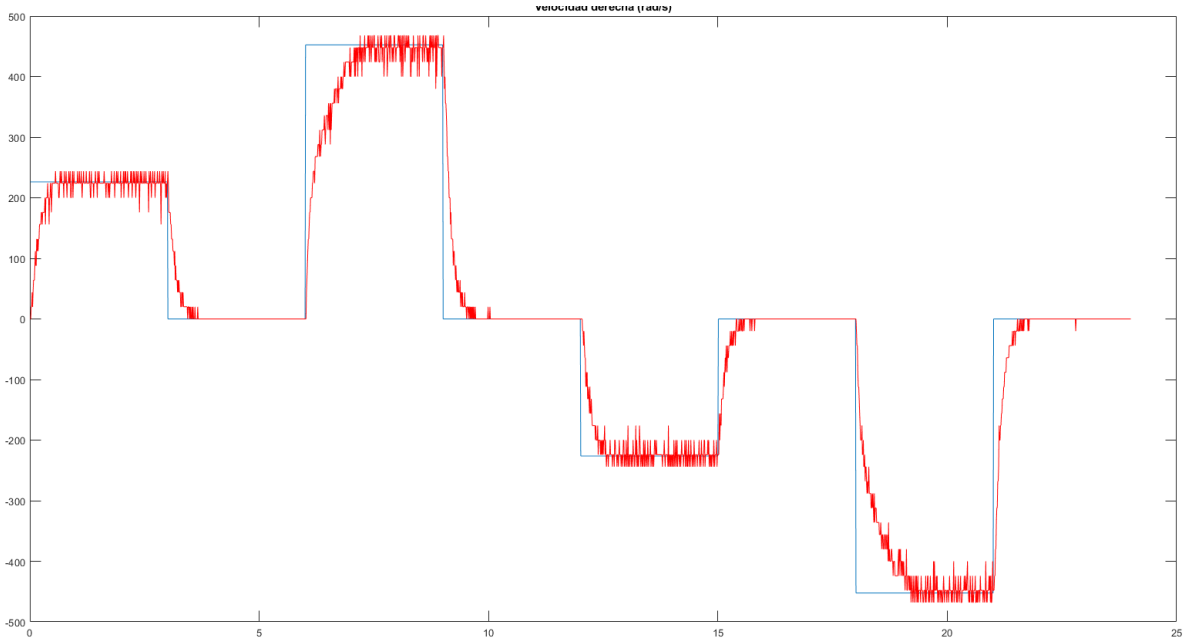

Figura 3.14: Medida de velocidad del motor derecho

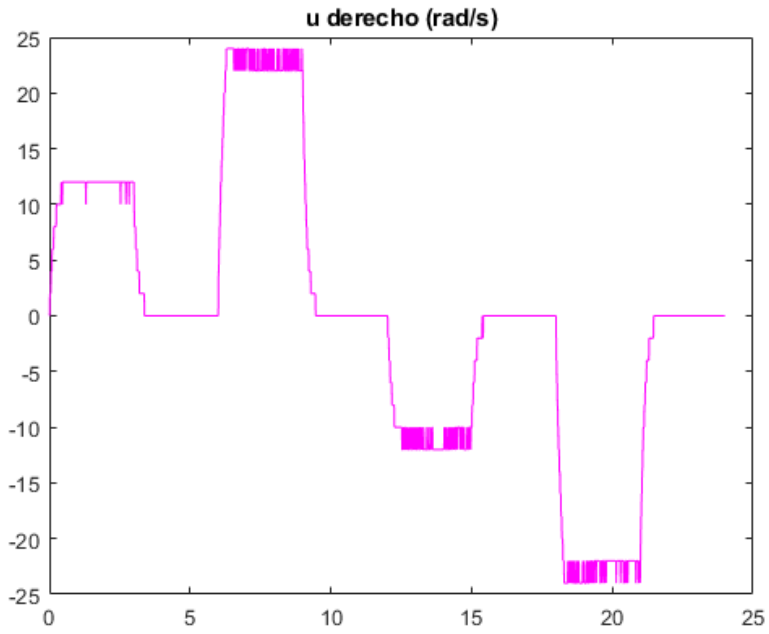

Figura 3.15: Medición de la tensión que se está aplicando al motor derecho

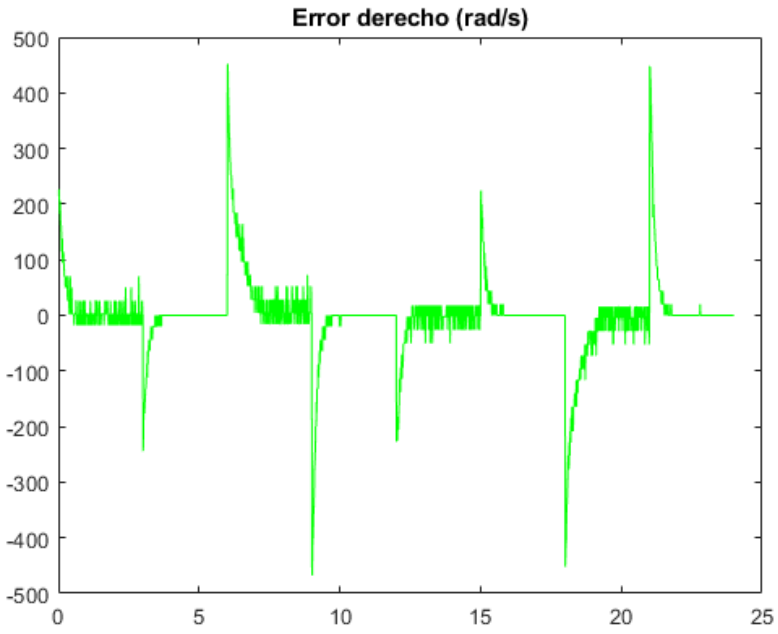

Figura 3.16: Error del motor derecho

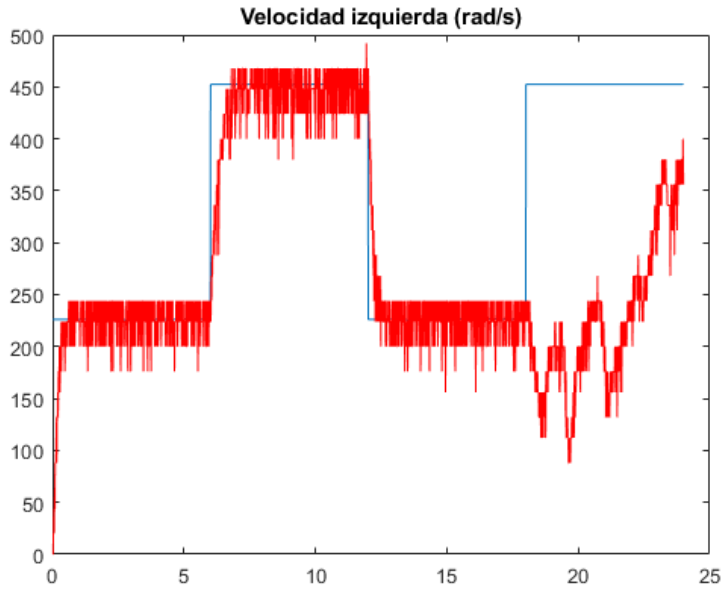

Figura 3.17: Medida de velocidad izquierda con perturbación

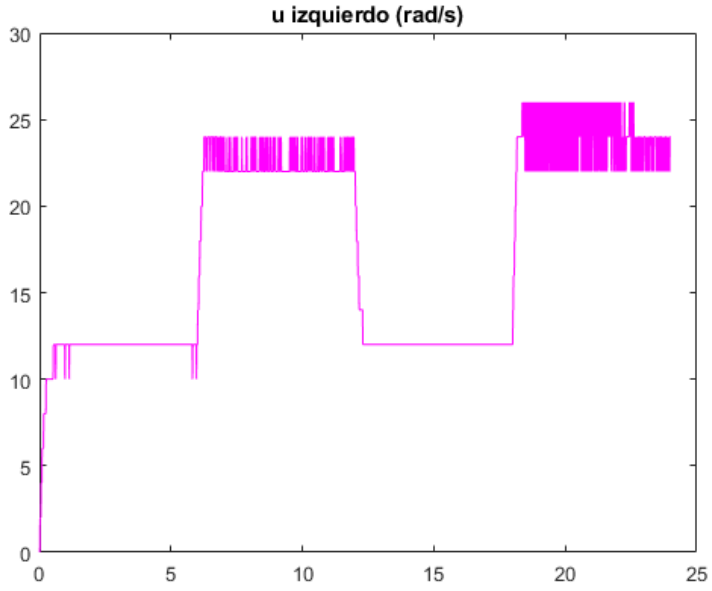

Figura 3.18: Valor de consigna aplicado al motor izquierdo

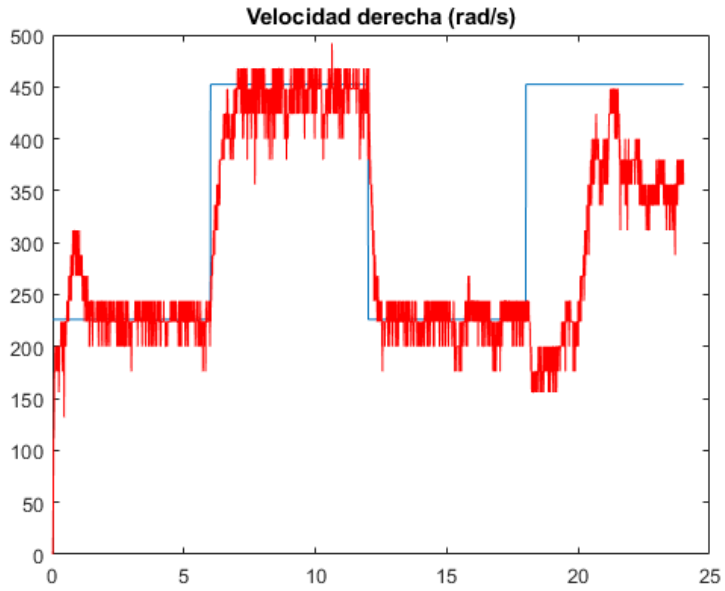

Figura 3.19: Medida de velocidad derecha con perturbación

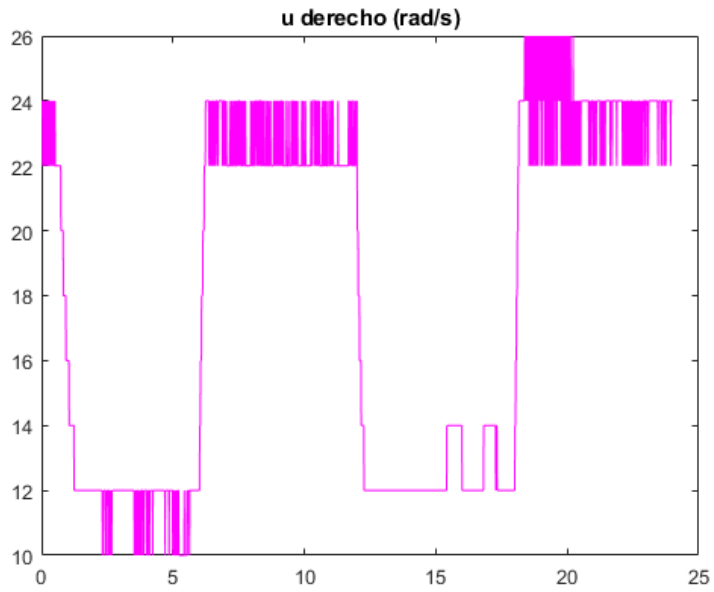

Figura 3.20: Valor de consigna aplicado al motor derecho

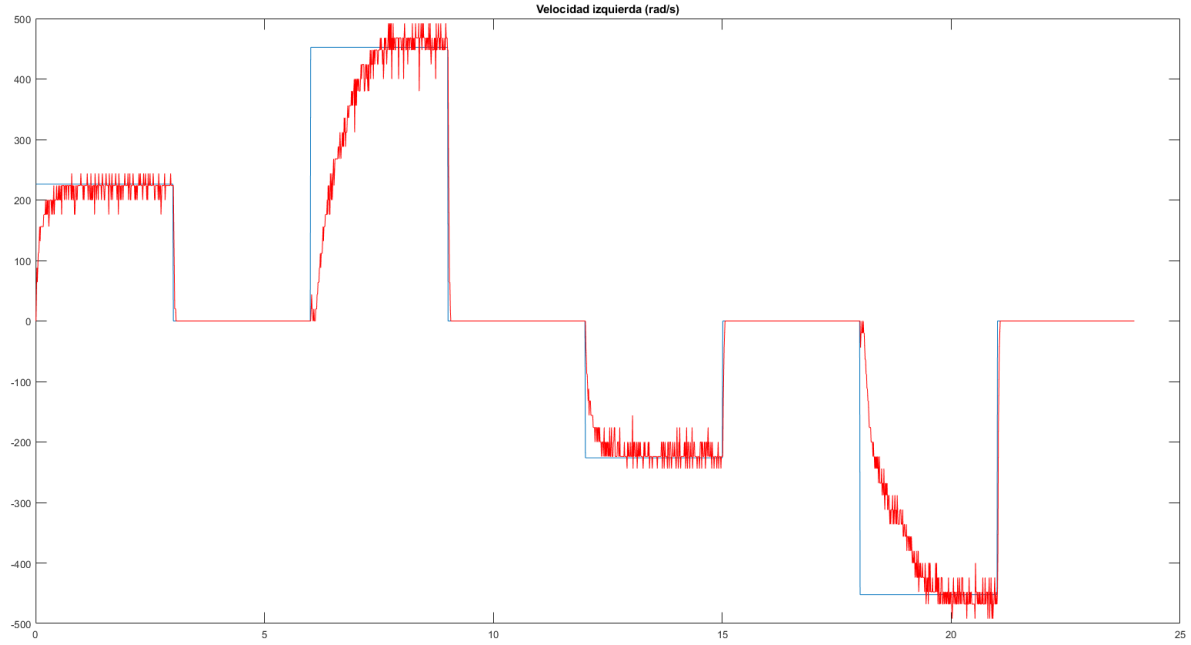

Figura 3.21: Velocidad del motor izquierdo sin perturbación y en lazo abierto

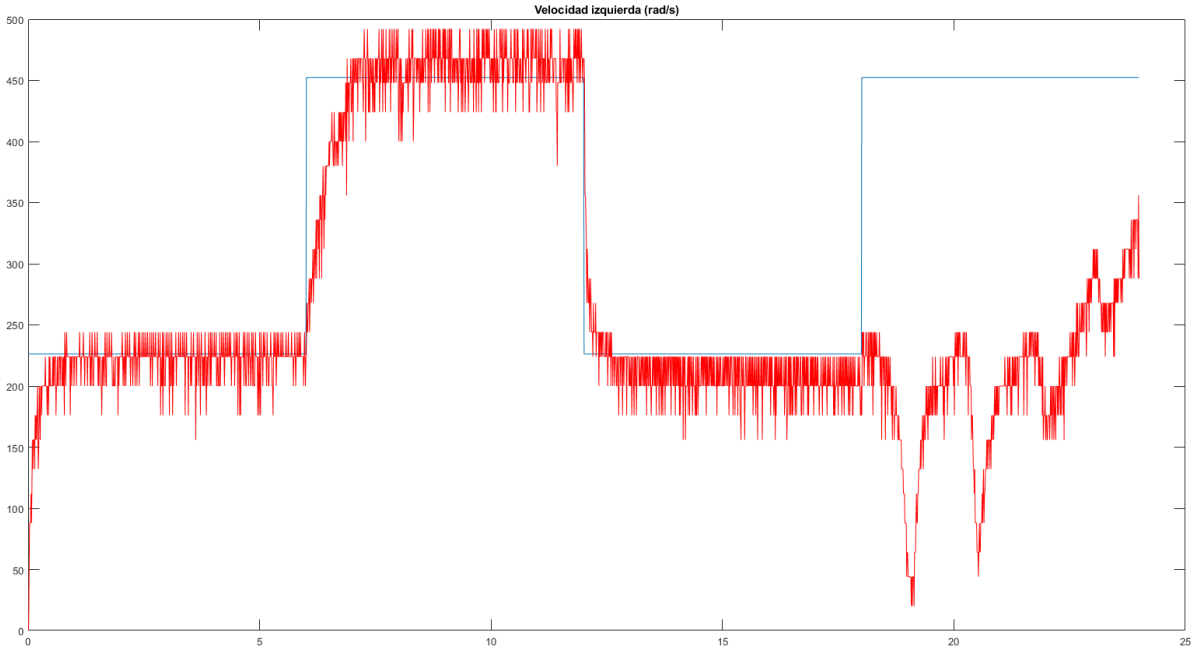

Figura 3.22: Velocidad del motor izquierdo con perturbación y en lazo abierto

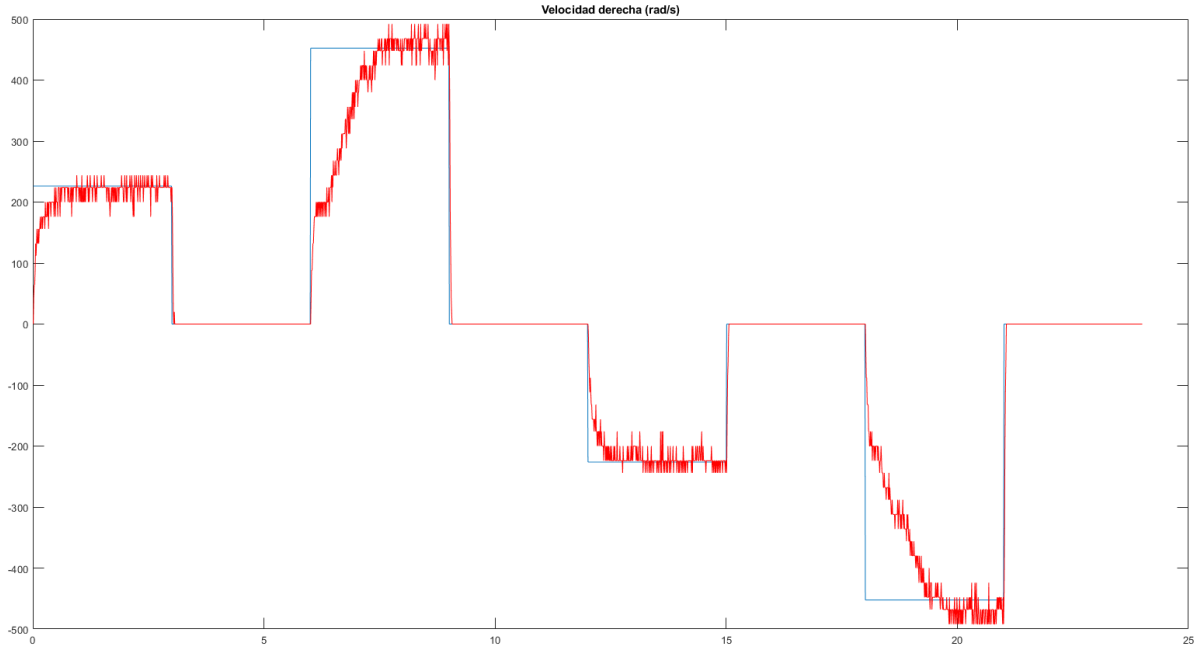

Figura 3.23: Velocidad del motor derecho sin perturbación y en lazo abierto

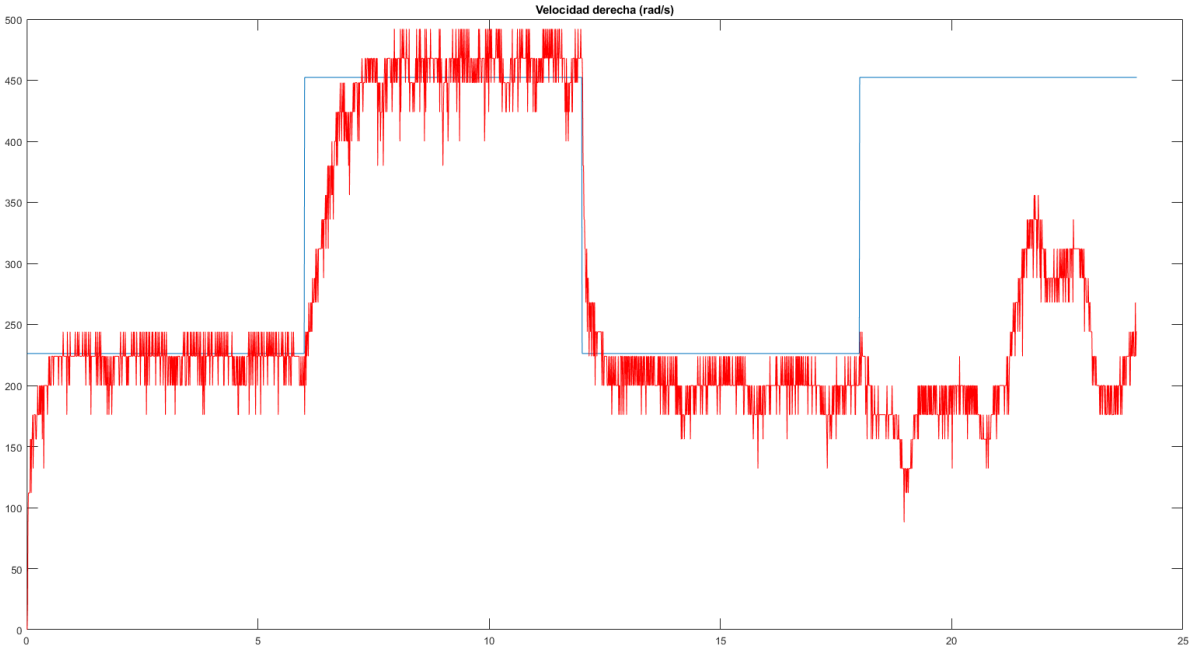

Figura 3.24: Velocidad del motor derecho con perturbación y en lazo abierto

Capítulo 4

Nodo motores: nivel lógico

El lenguaje de programación es C, muy útil, versátil y fácil de usar. Se utiliza el entorno de desarrollo proporcionado por ARM para organizar proyectos, cargar código, simular, etc denominado μVision. En dicho entorno se desarrolla el código correspondiente a los nodos del sistema y se prueban antes de cargarlos en placa en el debugger.

Previamente a la explicación del código, se expresa en el flujograma 4.1 el funcionamiento general del mismo, para así ayudar a su comprensión, resultando más fácil desarrollarlo posteriormente.

Como se puede comprobar, lo primero que se realiza al iniciar el programa, es la configuración de periféricos, comenzando por el PWM, el cual se configura con la sección de código 4.1.

Una vez configurado el PWM, se configura el interfaz CAN con el fragmento de código 4.2. En este trabajo, para facilitar el uso de dicho interfaz, se usan las librerías LPC_FULLCAN_SW, las cuales se encargan de la modificación de los registros que incorpora la tarjeta LPC2129 mediante el uso de funciones y parámetros. De esta forma se configura el periférico CAN1 de la tarjeta, estableciendo su vector de interrupción en el 0, velocidad de transmisión a $1\,\mathrm{Mb\,s^{-1}}$; en el vector 2 de interrupción su error de CAN; y en el vector 4 las interrupciones de recepción de mensajes. Por último, se establecen los filtros de CAN para actuar cuando se reciben mensajes provenientes del joystick y del PC.

Por último, se configura el timer del microcontrolador en el algoritmo 4.3. Dicho timer se desea que genere una interrupción cada 100 µs, como base de tiempos de un repertorio de timers virtuales. Se elige este método porque así se pueden tener varios temporizadores software con una sola interrupción, y no se sobrecarga el sistema de interrupciones.

Una vez configurado, se usa la rutina de interrupción del timer para incrementar los contadores de tiempo, como se puede ver en el fragmento de código 4.4. Además se utiliza una función que reactiva el timer, la cual se debe usar cada vez que se produce un flag de finalización del tiempo.

Una vez realizada la configuración completa de todos los periféricos necesarios en el sistema, se entra en un bucle infinito, en el que continuamente se llama a la función que se encarga de contar pulsos de encoder, se comprueba si se han recibido mensajes CAN y cada 10 ms se realiza el control de los motores, en el caso de encontrarse en el modo joystick.

La función encargada de contar pulsos, se divide en dos partes: primero se detecta si se produce un flanco de bajada o subida en cualquiera de los 4 canales y posteriormente, se ha producido algún flanco que se interpreta para aumentar o disminuir el contador de pulsos.

En el fragmento de código 4.5 se puede ver la primera parte de la función *Cuenta_Pulsos*, en la cual se sondean los canales A y B de cada encoder para determinar si se ha producido algún flanco.

En la siguiente parte de código se muestra como se realiza el estudio de todos los casos en los que se produce un flanco de encoder, respetando la siguiente tabla lógica 4.1.

Tabla 4.1: Lógica seguida para aumentar o disminuir el contador de pulsos

Tipo de flanco	Canal del flanco	Nivel del canal opuesto	Acción del contador
Flanco de subida	A	Alto	Aumentar contador
Flanco de subida	A	Bajo	Reducir contador
Flanco de subida	B	Alto	Reducir contador
Flanco de subida	B	Bajo	Aumentar contador
Flanco de bajada	A	Alto	Reducir contador
Flanco de bajada	A	Bajo	Aumentar contador
Flanco de bajada	B	Alto	Aumentar contador
Flanco de bajada	B	Bajo	Reducir contador

Esta tabla ha sido desarrollada siguiendo el diagrama temporal proporcionado por el fabricante del encoder, en el cual se puede interpretar el sentido de giro del motor según el tipo de flanco que se produce, en qué canal se produce y el nivel que tiene el canal opuesto del mismo encoder. En la porción de código 4.6, se muestra las sentencias usadas para realizar dicha clasificación en ambos motores, adaptada para cada caso concreto.

Una vez realizado el conteo de los pulsos de encoder, lo siguiente que se debe realizar es comprobar si se ha recibido un mensaje proveniente del bus CAN y en caso afirmativo, leerlo. El fragmento de código encargado de leer los mensajes CAN se lista en el algoritmo 4.7.

Hasta este punto, el código se ejecuta de manera cíclica continuamente a máxima velocidad. A continuación, cada 10 ms se realiza el envío de los pulsos de encoder por el bus CAN y el control de los motores.

El envío de los pulsos de encoder se realiza por turnos. De esta forma en un turno se envían los pulsos de encoder del motor derecho y en el siguiente turno los correspondientes al motor izquierdo, teniendo así información completa de la odometría cada 20 ms. Esto se hace de esta forma porque previamente se hizo de manera secuencial, enviando primero la información de un encoder y posteriormente la del otro, pero por problemas temporales los mensajes no se llegaban a enviar sincronizadamente.

La forma de publicar los datos de los encoders por el bus es de manera absoluta, es decir, desde el encendido del sistema, los pulsos se suman o se restan continuamente en un mismo contador absoluto y se envían junto con su marca temporal absoluta, la cual aumenta cada 100 µs en la rutina de interrupción del timer.

Se utilizan las funciones 4.8 para publicar dichos mensajes por el bus CAN.

Una vez publicados los mensajes CAN tan solo queda realizar el control de los motores en el caso correspondiente, lo cual se realiza con las sentencias 4.9.

La variable *modo* es definida en las lecturas del bus CAN debido a que es enviada por el nodo joystick, en función de un interruptor integrado en el mismo. Por otro lado, el modo por defecto tiene como significado la parada completa del sistema, donde se establecen las consignas de velocidad a 0 y se resetean las lecturas de los pulsos de encoder.

La explicación del código que realiza el control de los motores, se encuentra en el apartado 3.3.1.3 de implementación del controlador PI.

4.1 Algoritmos y flujogramas

Para facilitar la lectura ordenada de la estructura y secuencia del código del "nodo de motores", en esta sección se reunen todos los cuadros de algoritmos y flujogramas mencionados en este capítulo.

Algoritmo 4.1: Configuración de periféricos PWM

Result: Se configuran el PWM2 y PWM6 en los pines P0.7 y P0.9, sin interrupción, con autoreset y sin parada

$PINSEL0| = (1 << 15)|(0 << 14);$
$PINSEL0| = (1 << 19)|(0 << 18);$
$PWMPR = 0;$
$PWMMR0 = MR0\_VALUE;$
$PWMMR2 = MR0\_VALUE/2;$
$PWMMR6 = MR0\_VALUE/2;$
$PWMMCR| = (0 << 0)|(1 << 1)|(0 << 2);$
$PWMLER| = (1 << 0)|(1 << 2)|(1 << 6);$
$PWMTCR| = (1 << 3)|(1 << 0);$
$PWMPCR| = (1 << 10)|(1 << 14);$

Algoritmo 4.2: Configuración de periférico CAN

Result: Configuración de CAN1, cuyos pines son CANH-0 y CANL-0

$FullCAN\_Init(1, 0, CANBitrate001m_12MHz);$
$FullCAN\_SetErrIRQ(2);$
$VICVectAddr4 = (unsignedlong)FullCAN_CANISR_Rx1;$
$VICVectCntl4 = 0x20|26;$
$VICIntEnable = 0x02000000L;$
$FullCAN\_SetFilter(1, 0x110);$
$FullCAN\_SetFilter(1, 0x111);$
$FullCAN\_SetFilter(1, 0x120);$

Algoritmo 4.3: Configuración del timer

Result: Configuración de Timer0, generando interrupción cada 100 µs

$T0MR0 = 5999;$
$T0MCR = 3;$
$T0TCR = 1;$
$Timer1 = T1ticks;$
$T1 = 0;$
$VICVectAddr3 = (unsignedlong)Timer0ISR;;$
$VICVectCntl3 = 0x20|4;$
$VICIntEnable = 0x00000010L;$

Algoritmo 4.4: Funciones del timer

Result: Rutina de interrupción y función de activación del timer

Function *void Timer0ISR (void) __irq* **is**
    T0IR = 1;
    gTimerTick++;
    encoderITimerTick++;
    encoderDTimerTick++;
    if *Active_Timer1* **then**
        Timer1;
    end
    if *Timer1 == 0* **then**
        T1 = 1;
        Active_Timer1 = 0;
    end
    VICVectAddr = 0xFFFFFFFFL;
end
Function *void Set_T1(void)* **is**
    Timer1=T1ticks;
    T1=0;
    Active_Timer1=1;
end

Algoritmo 4.5: Sondeo de flancos

Result: Se compara el valor del canal del encoder en el instante anterior y en el actual para determinar si se ha producido un flanco

$if((prev\_A\_izq == LOW)\&\&((IO0PIN >> 4\&1) == HIGH))flanco\_subida\_A\_izq = TRUE;$
$if((prev\_A\_izq == HIGH)\&\&((IO0PIN >> 4\&1) == LOW))flanco\_bajada\_A\_izq = TRUE;$
$if((prev\_B\_izq == LOW)\&\&((IO0PIN >> 5\&1) == HIGH))flanco\_subida\_B\_izq = TRUE;$
$if((prev\_B\_izq == HIGH)\&\&((IO0PIN >> 5\&1) == LOW))flanco\_bajada\_B\_izq = TRUE;$
$if((prev\_A\_der == LOW)\&\&((IO0PIN >> 24\&1) == HIGH))flanco\_subida\_A\_der = TRUE;$
$if((prev\_A\_der == HIGH)\&\&((IO0PIN >> 24\&1) == LOW))flanco\_bajada\_A\_der = TRUE;$
$if((prev\_B_der == LOW)\&\&((IO0PIN >> 22\&1) == HIGH))flanco\_subida\_B\_der = TRUE;$
$if((prev\_B_der == HIGH)\&\&((IO0PIN >> 22\&1) == LOW))flanco\_bajada\_B\_der = TRUE;$
$prev\_A\_izq = IO0PIN >> 4\&1;$
$prev\_B\_izq = IO0PIN >> 5\&1;$
$prev\_A\_der = IO0PIN >> 24\&1;$
$prev\_B\_der = IO0PIN >> 22\&1;$

Algoritmo 4.6: Sondeo de flancos

Result: Aplicación de la tabla lógica de los pulsos de los encoder

if *flanco_subida_A_izq* **then**
    flanco_subida_A_izq = FALSE;
end
if *IO0PIN >> 5 & 1 == LOW* **then**
    Pulsos_Izq–;
else
    Pulsos_Izq++;
end

Algoritmo 4.7: Lectura de mensaje CAN

```
Result: Se obtiene una estructura de tipo FULLCAN_MSG con los datos provenientes del CAN
Function void lee_can(void) is
    FULLCAN_MSG MsgBuf;
    if FullCAN_PullMessage(1 , &MsgBuf) then
        if (MsgBuf.Dat1 & 0x07FF) == ID_MSG_JOYA then
            modo=(short)(MsgBuf.DatB & 0xFFFF);
            if modo!=3 then
                auxD=((signed char)(MsgBuf.DatA & 0xFF));
                auxI=((signed char)((MsgBuf.DatA & 0xFF00)>>8));
            end
        end
    end
end
```

Algoritmo 4.8: Envío de mensaje CAN

```
Result: Formación de la estructura FULLCAN_MSG y envío de la misma por el bus
Function void envio_canD(void) is
    FULLCAN_MSG MsgBuf;
    MsgBuf.Dat1 = 0x00080101L;
    MsgBuf.DatA = encoderDabs;
    MsgBuf.DatB = cuentaD;
    FullCAN_PushMessage(1 , &MsgBuf);
end
```

Algoritmo 4.9: Ejecución del control de los motores en función del modo que se encuentre el sistema

```
Result: Modo 0 significa parada del sistema y modo 1 controlar los motores
switch modo do
    case 0 do
        auxD=0;
        auxI=0;
        Set_Velocidad_Izq(0);
        Set_Velocidad_Der(0);
        Pulsos_Der=0;
        Pulsos_Izq=0;
    case 1 do
        Controla_Motores();
    end
    otherwise do
        auxD=0;
        auxI=0;
        Set_Velocidad_Izq(0);
        Set_Velocidad_Der(0);
        Pulsos_Der=0;
        Pulsos_Izq=0;
    end
end
```

Algoritmo 4.10: Implementación del controlador PI

Result: Se utilizan los pulsos contados para calcular la velocidad, se realiza el control y se aplican las nuevas velocidades.

Function *void Controla_Motores(void)* **is**

```
tss++;
//Motor izquierdo;
Velocidad_Izq = (Pulsos_Izq*2*PI)/(N*EDGES*Ts*0.001);
Pulsos_Izq = 0;
error_Izq = (Consigna−Velocidad_Izq)−Error_Sat_Izq;
u_Izq = G*error_Izq−c*G*error_nm1_Izq+u_nm1_Izq;
Set_Velocidad_Izq(u_Izq); Error_Sat_Izq = (u_Izq−Consigna_Sat_Izq)*Ka;
error_nm1_Izq = error_Izq; u_nm1_Izq = u_Izq; U0THR = (short)(Velocidad_Izq/2);
U0THR = (short)(u_Izq/A);
//Motor derecho;
Velocidad_Der = (Pulsos_Der*2*PI)/(N*EDGES*Ts*0.001);
Pulsos_Der = 0;
error_Der = (Consigna−Velocidad_Der)−Error_Sat_Der;
u_Der = G*error_Der−c*G*error_nm1_Der+u_nm1_Der;
Set_Velocidad_Der(u_Der); Error_Sat_Der = (u_Der−Consigna_Sat_Der)*Ka;
error_nm1_Der = error_Der;
u_nm1_Der = u_Der;
U0THR = (short)(Velocidad_Der/2);
U0THR = (short)(u_Der/A);
```

end

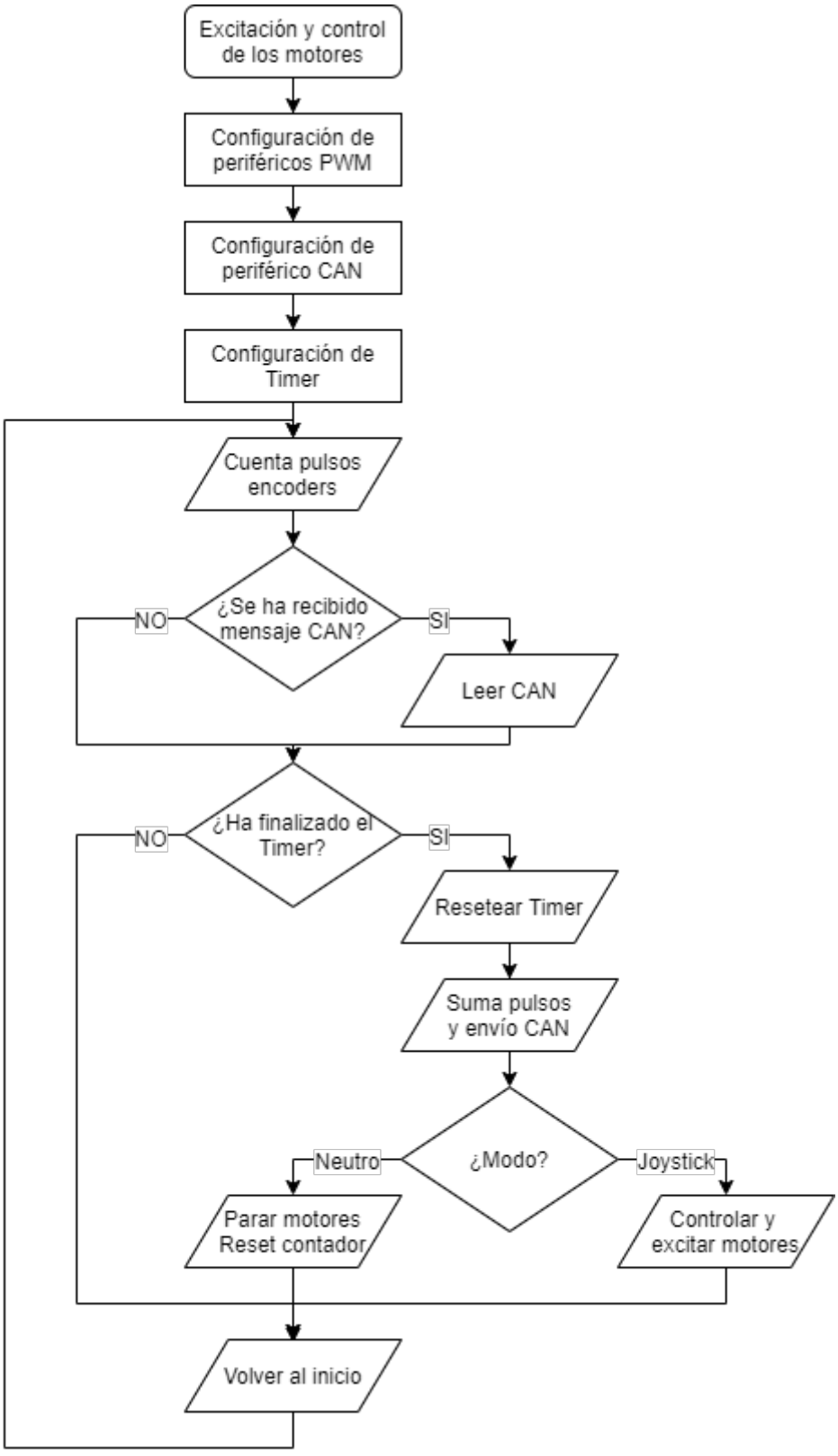

Figura 4.1: Flujograma del código del nodo motores

Capítulo 5

Nodo joystick

5.1 Introducción

Toda plataforma móvil necesita una interfaz de control de la misma, en nuestro caso para realizar tareas como el encendido, la elección entre modos de operación y comandar su movimiento. Ante esta necesidad se añade el nodo joystick, y que forma la Human Machine Interface (HMI) desde donde el usuario puede controlar el sistema completo de una manera sencilla y robusta.

Para el diseño de este nodo se deben tener presentes las tareas que debe cumplir, que son:

- Orden de encendido del sistema.
- Selección de modo de funcionamiento: manual o autónomo (con PC).
- Control de dirección y velocidad.
- Activación de una señal de audio (bocina o claxon).
- Visualización de otros parámetros como: nivel de batería, encendido y modo.

5.2 Diseño electrónico del nodo

Desde un pulsador con enclavamiento, situado en el mando del joystick, se realiza el encendido de la silla. Dicho pulsador conecta uno de los terminales del relé de encendido a masa, activando sus contactos y dando paso a la alimentación de 24 V a toda la plataforma, como ya se ha explicado anteriormente.

Tal y como ocurre en el nodo motores, los dispositivos usados en el nodo joystick funcionan con una tensión de 5 V, por lo tanto se debe reducir la tensión de alimentación suministrada desde el bus. Se utiliza la misma estructura usada en el nodo anterior, la cual se puede ver en la figura 3.2.

Cuando el microcontrolador queda alimentado y se enciende, pasa a nivel alto el pin P0.15. Dicho pin está conectado a una estructura de transistores, cuya función es actuar como interruptores de señalización de estado de la alimentación, como se puede ver en la figura 5.1. Además se añaden resistencias para ajustar las corrientes en saturación a valores moderados.

Cuando el microcontrolador activa a alto el pin P0.15 el transistor Q5 pasa a ON, permitiendo el encendido de la luz (señal ON-L). Por otro lado, Q5 activa al transistor Q4, el cual permite la emisión

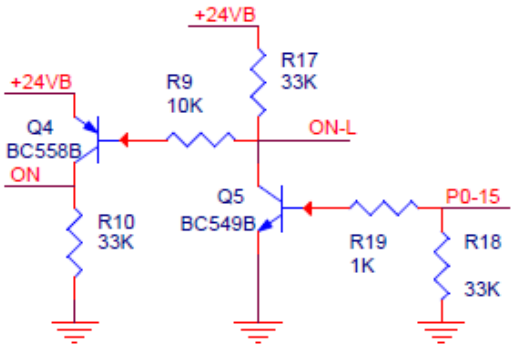

Figura 5.1: Circuito que controla la señal de ON y el encendido de la luz

de la señal ON, que se transmite por el bus de la plataforma y es usada por otros nodos para indicar que el nodo joystick se encuentra encendido.

El elemento central de este nodo es el joystick. Existen cuatro tipos de joystick: resistivos, de efecto Hall, inductivos y ópticos. Los joysticks resistivos son los mas económicos pero es descartado debido a la degradación de sus elementos internos como los contactos que con el tiempo se desgastan. Los joysticks de efecto Hall se degradan de igual forma que los resistivos, pierde sus capacidades ferromagnéticas y están muy sujetos a cambios con la temperatura.

La opción elegida es un joystick inductivo, sin contacto, de 2 ejes y bajo consumo, fabricado por APEM. Dicho joystick se alimenta a 5 V y proporciona la información de posición mediante 3 terminales: referencia, eje X (UP/DOWN) y eje Y (LEFT/RIGHT). Para la adaptación de dicha información se diseña un circuito encargado de amplificar y adaptar la tensión proporcionada por los terminales del joystick al rango de tensiones de entrada del ADC del microcontrolador, el cual va de 0 V a 3.3 V.

Con dicha función se plantea la estructura de la figura 5.2. El valor de las resistencias se ha calculado aplicando superposición a las salidas de cada uno de los amplificadores. Se utiliza el integrado LM324, el cual porta 4 amplificadores operacionales en su interior.

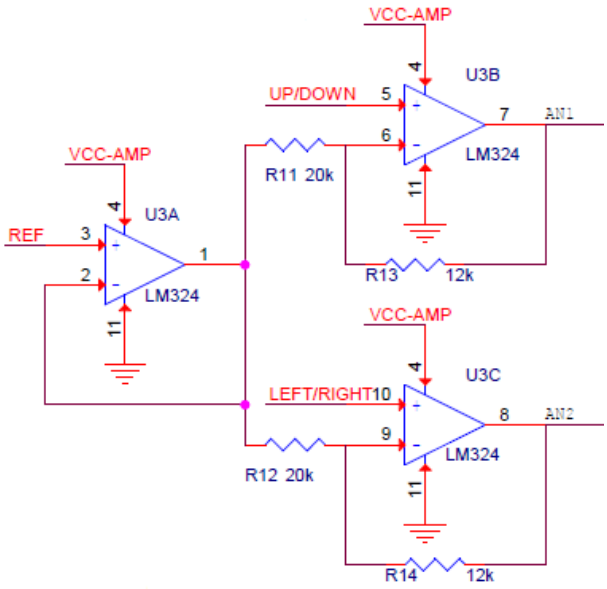

Figura 5.2: Circuito de amplificación y adaptación de la tensión del joystick

Según la hoja de características del joystick la tensión del terminal REF es la mitad del valor de la tensión de alimentación, en este caso 5 V, por tanto V_{ref} tiene un valor de 2.5 V. Se usa uno de los operacionales como seguidor de la tensión V_{REF}, para eliminar de esta forma el efecto de carga.

A continuación, se utilizan otros dos operacionales en configuración inversora para la tensión de referencia y configuración no inversora para la tensión de los ejes. Las tensiones de salida V_{AN1} y V_{AN2} responden a la expresión 5.1.

$$V_o = \left(1 + \frac{R_2}{R_1}\right) \cdot V_{eje} - V_{ref} \cdot \frac{R_2}{R_1} \tag{5.1}$$

Siendo R_2 la resistencia de realimentación y R_1 la resistencia de entrada y V_{eje} la tensión proporcionada como información de up/down y left/right.

La hoja de características indica además que la tensión de los ejes varia un 10 % desde el punto central de tensión, es decir V_{ref}, teniendo de esta forma un rango de tensiones de entrada tal y como se indica en la expresión 5.2.

$$0{,}4 \cdot V_{cc} < V_{eje} < 0{,}6 \cdot V_{cc} \tag{5.2}$$

Por tanto, la tensión de cada eje que se tiene realmente varía desde los 2 V a los 3 V. Conocidos estos valores y conocido el rango de entrada del ADC del microcontrolador, se puede calcular el valor de las resistencias, ya que despejando en la expresión 5.1 se obtiene:

$$\frac{R_2}{R_1} = \frac{V_o - V_{eje}}{V_{eje} - V_{ref}} \tag{5.3}$$

Entonces se dan dos casos deseables: que V_o valga 3.3 V cuando V_{eje} valga 3 V o que V_o valga 0 V cuando V_{eje} valga 2 V. Así se obtienen dos valores posibles para la relación $\frac{R_2}{R_1}$, 0.6 o 4. Si se sustituyen dichos valores en la expresión 5.1 se consiguen las funciones de transferencia 5.4 y 5.5 respectivamente, suponiéndose lineales. A su vez se puede ver representada en la figura 5.3.

$$V_{o1} = 1{,}6 \cdot V_{eje} - 1{,}5 \tag{5.4}$$

$$V_{o2} = 5 \cdot V_{eje} - 10 \tag{5.5}$$

Como el rango de entrada del microcontrolador es de 0 V a 3.3 V interesa utilizar la combinación de resistencias de 0,6 ya que si se utiliza la combinación de valor 4, la tensión de salida llega 5 V y no es deseable.

Una vez seleccionada la relación de resistencias de 0.6, se deben elegir valores correspondientes a la serie E24 de resistencias. Como se puede comprobar en el esquemático del circuito 5.2, las resistencias finalmente usadas son de 12 kΩ y 20 kΩ.

Usualmente, se permite al usuario, por razones de confort o confianza, aumentar o disminuir la velocidad máxima del sistema. Se desea usar el potenciómetro presente en el mando del joystick como limitador de velocidad, cuyo valor de tensión es leído por la tarjeta LPC2129, y modifica la consigna de velocidad según la posición del potenciómetro. Está configurado como un resistor variable, por lo que se debe diseñar un circuito que proporcione una tensión proporcional en función de la posición del potenciómetro. Para ello se parte del circuito mostrado en la figura 5.4.

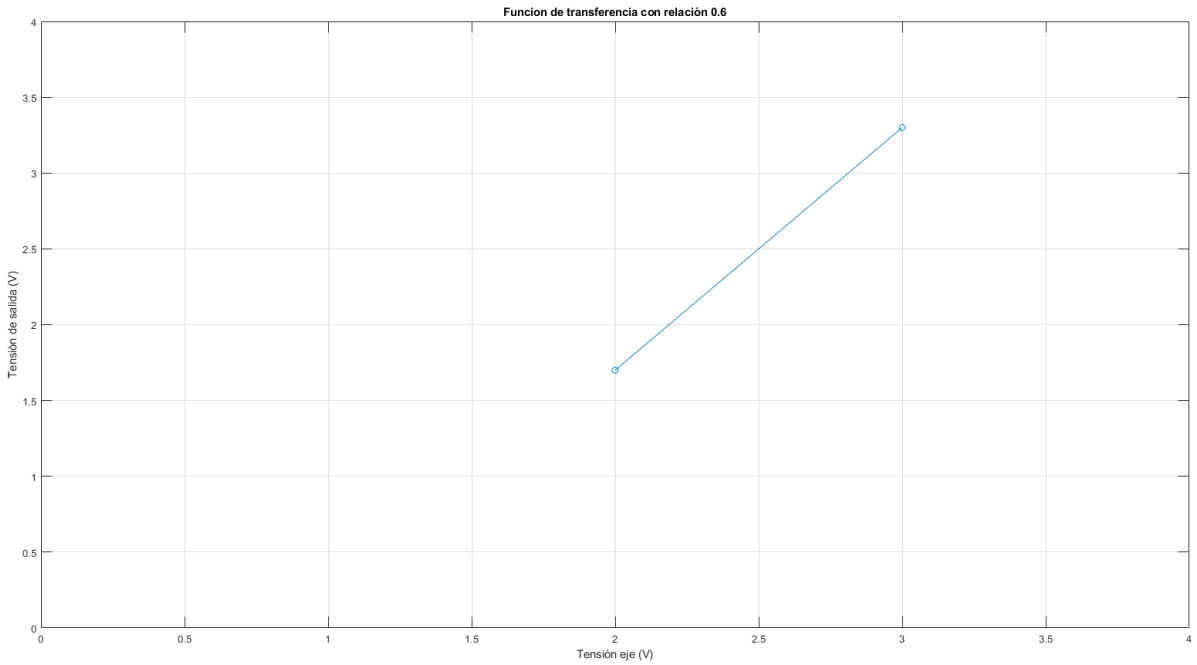

Figura 5.3: Función de transferencia con relación de resistencias 0.6

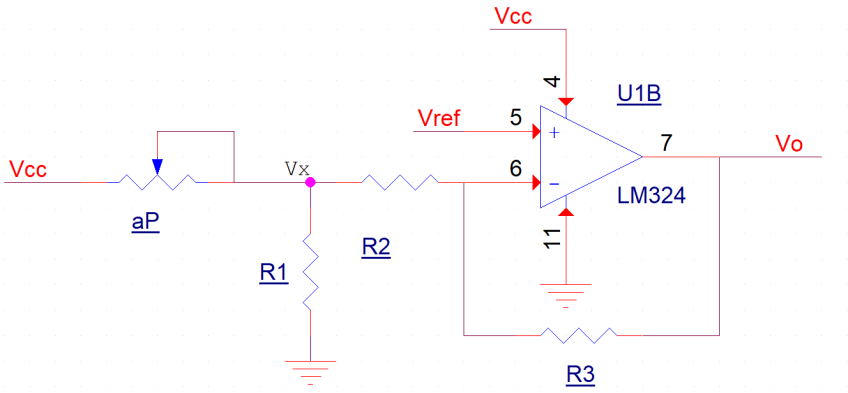

Figura 5.4: Esquemático inicial del circuito limitador de velocidad

Si la impedancia de entrada al circuito es mucho mayor que R1 se puede afirmar que la tensión V_x sigue la expresión de la ecuación 5.6.

$$V_x \approx V_{cc} \cdot \frac{R_1}{R_1 + \alpha P} \tag{5.6}$$

con P=10 kΩ y α entre 0 y 1.

El amplificador está alimentado entre V_{cc} y masa. En el terminal no inversor se introduce V_{ref} cuyo valor es $V_{cc}/2$ procedente del joystick. Operando se obtiene que la tensión de salida del circuito es la correspondiente a la ecuación 5.7.

$$V_o = V_{ref} \cdot \frac{R_3 + R_2}{R_2} - \frac{R_3}{R_2} \cdot V_x \tag{5.7}$$

suponiendo que $R_2 \gg R_1$. Sustituyendo V_x en la expresión anterior se tiene la ecuación completa de la tensión de salida:

$$V_o = \frac{V_{cc}}{2} \cdot \frac{R_3 + R_2}{R_2} - \frac{R_3}{R_2} \cdot V_{cc} \cdot \frac{R_1}{R_1 + \alpha P} \tag{5.8}$$

Como caso particular, para simplificar la ecuación, se estudia que R2 y R3 sean iguales en la siguiente expresión.

$$V_o = V_{cc} \cdot \left(1 - \frac{R_1}{R_1 + \alpha P}\right) \tag{5.9}$$

Entonces las tensiones máxima y mínima de salida (llegando a los extremos de α) son:

$$V_o max = V_{cc} \cdot \left(1 - \frac{R_1}{R_1 + P}\right) \tag{5.10}$$

$$V_o min = 0 \tag{5.11}$$

Debido a que se desea conectar al ADC de la tarjeta microcontroladora, cuyo SPAN de entrada es de 0 V a 3.3 V, la tensión máxima se iguala a 3.3 V y se obtiene el valor de R_1:

$$K = \frac{V_o max}{V_{cc}} = \frac{3{,}3}{5} = 0,66 \tag{5.12}$$

Asumiendo que $K = 1 - \frac{R_1}{R_1 + P}$. Entonces el valor de R_1 es 5151 Ω. Dado que no es un valor estándar de la serie E24, el siguiente valor de resistencia de dicha serie es 5600 Ω. Para este valor de resistencia,usando la expresión 5.10, la tensión máxima que se tiene a la salida se ve disminuida.

$$V_o max = 5 \cdot \left(1 - \frac{5600}{5600 + 10000}\right) = 3.205\,\text{V} \tag{5.13}$$

En la figura 5.5 se muestra el diseño del circuito final en PSpice con los respectivos valores de las resistencias. Por último, para comprobar la no linealidad de la tensión de salida, se representa en la figura 5.6 la función de transferencia teórica del circuito variando de nuevo el valor de α.

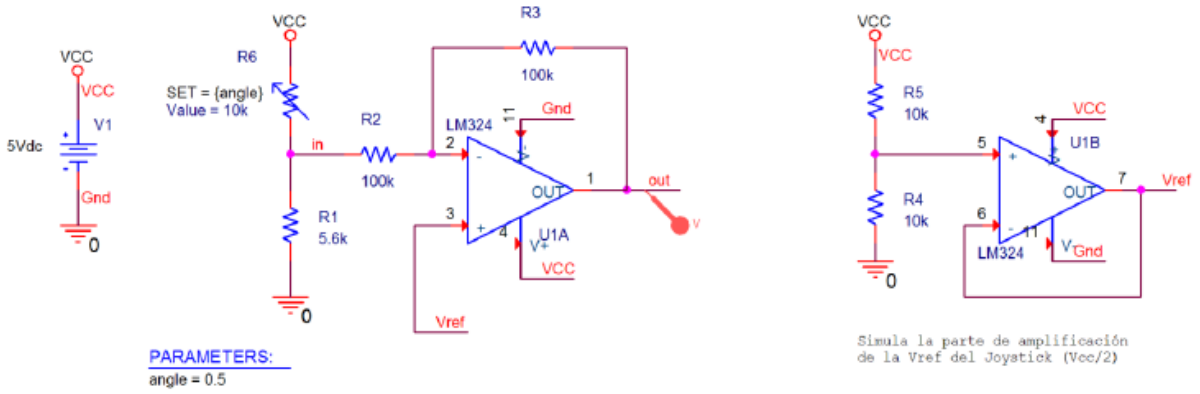

Figura 5.5: Esquemático final del circuito

Aunque se presenta de forma no lineal no se aprecia la curvatura de forma práctica, a la hora de ser usado por el usuario.

Todos estos elementos electrónicos deben ser encapsulados en un dispositivo ergonómico para el usuario, fácil de usar y robusto, de forma que no presente peligros sobre la electrónica y la integridad del

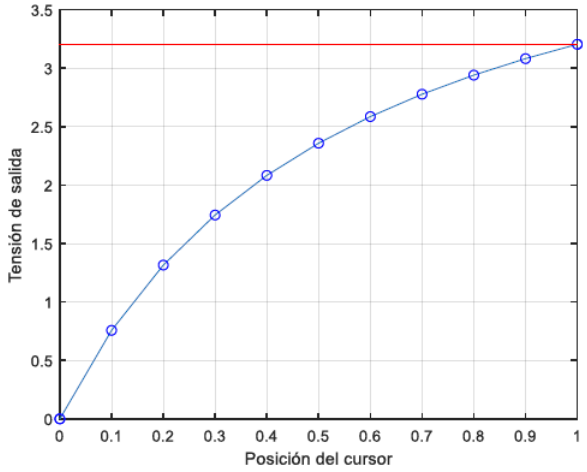

Figura 5.6: Función de transferencia teórica

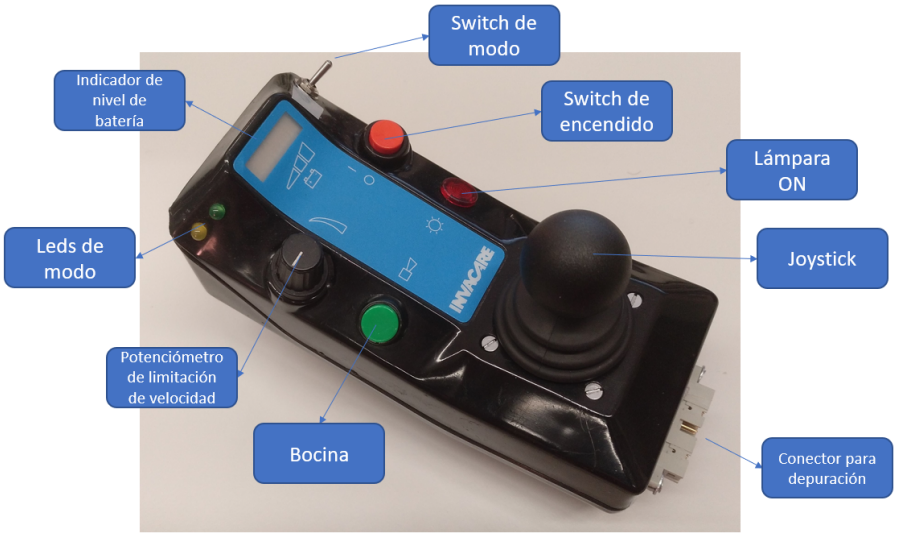

Figura 5.7: Integración del nodo joystick en KATE

sistema. De este modo se ha integrado todo el nodo joystick del mando de control que se puede observar en la figura 5.7.

Como se puede ver, en la parte izquierda se encuentran los LEDs indicadores de modo, verde para joystick y amarillo para PC (cuando sea implementado), el potenciómetro encargado de ajustar la velocidad máxima y el pulsador del claxon. En la parte central arriba se encuentra el indicador de nivel de batería y abajo el joystick, y por último en la parte derecha se encuentra el interruptor de selección de modo, el pulsador de encendido de la silla y el indicador luminoso de encendido del sistema completo.

En la parte parte inferior del mando, se encuentra el conector RJ45, al cual se conecta el cable Ethernet usado como bus en la plataforma y se lleva hasta el hub presente en el nodo motores. Por otro lado, en la parte trasera, se tiene el conector de cable paralelo usado para reprogramar la tarjeta LPC2129 usada en el nodo, de forma que en el caso que se deban realizar modificaciones software no sea necesario abrir el mando para volver a programar el microcontrolador.

5.3 Código del nodo joystick

En este apartado se va a explicar el funcionamiento general del código implementado en el nodo joystick, haciendo especial énfasis en las funciones más importantes. Para comenzar se expone el flujograma del código completo en la figura 5.9.

Así como en el nodo motores, se usa el entorno μVision para desarrollar, simular y cargar el código. Como se puede apreciar en el flujograma, lo primero que se realiza es la configuración del sistema y los periféricos.

En la primera función de configuración 5.1, se establecen ciertos pines como salidas, los cuales serán usados posteriormente. Además se activa el pin P0.15, para señalar el encendido del microcontrolador al resto de la plataforma, como se ha explicado en el apartado hardware del nodo joystick. Se establece el divisor del reloj de los periféricos a 1, por tanto su reloj funcionara con una frecuencia de 60 MHz. Por ultimo se deshabilitan todas las interrupciones con la intención de habilitar solo las necesarias cuando se configuren los periféricos que precisan de interrupción.

La función 5.2 es la de configuración de la interfaz CAN. En este prototipo, el bus CAN del nodo joystick solo se utiliza para enviar datos al resto de nodos, y no se recibe ninguno mensaje, por tanto no se establece ningún filtro para la interfaz. De esta forma, se configura el CAN1 en el vector de interrupciones 0 y se ajusta a una velocidad de $1\,\mathrm{Mb\,s^{-1}}$, respetando la velocidad del bus completo. Por ultimo se establece el error del CAN1 en el vector de interrupciones 2.

Por ultimo, la última función de configuración 5.3 pertenece al timer. Dicho temporizador es usado para leer los datos del joystick y enviarlo por el bus cada 100 ms. Se configura de la misma forma que el timer usado en el modo motores, tan solo se cambia la constante de tiempo TS_MSEGUNDOS que se usa para activar el flag, cuyo valor pasa a ser 100 en lugar de 10. Otra diferencia respecto al nodo motores, es que cuando se agota el tiempo del timer y han pasado los 100 ms, en la propia rutina de interrupción, se toman los valores analógicos de los diferentes canales del ADC y se convierten a valores digitales.

La función para adquirir los valores analógico y convertirlos se muestra en el fragmento de código 5.4. Se usan 3 canales del ADC de los 8 presentes en la tarjeta LPC2129. Dichos ADC se controlan mediante el registro ADCR, con el cual se selecciona el canal que se va a usar, el divisor del reloj de los periféricos se establece a 5, no se usa modo burst, se habilita el periférico y se comienza la conversión en el mismo instante. El valor de la conversión se lee del registro cuando ha finalizado la conversión y se guarda en las respectivas variables.

La forma de detectar si se han recibido mensajes CAN y de leerlos es idéntica a como se realiza en el nodo motores. Además dicha función no será usada en el actual nodo, debido a que se decide no implementar ningún filtro de recepción de mensaje CAN.

En función del modo en que se encuentre la plataforma, seleccionado por el usuario, se entra en modo neutro o en modo joystick. Esto se implementa con el fragmento de código 5.5, mediante la sentencia switch-case en función de la variable modo.

Si el sistema se encuentra en modo joystick, se ejecuta la función Conversion_Joystick. (la cual se explica posteriormente en el capítulo de navegación 6) y envía los datos por el bus CAN. Además, se realiza el encendido del LED correspondiente. La función Can_Envio tiene la estructura del código 5.6.

El nodo joystick usa el identificador 0x110 para enviar los datos de velocidad y modo. Primero asigna los valores a las variables de envío, realizando el cast pertinente, y posteriormente forma la estructura del mensaje a enviar.

La última parte del código que queda por analizar es la que se encarga de realizar el cambio de modo. Como se puede ver en el fragmento 5.7 continuamente se está comprobando qué entrada se encuentra a nivel alto. Estas entradas están conectadas al interruptor de selección de modo. Si una de ellas está inactiva la otra estará inactiva.

De esta forma se comprueba en qué posición se encuentra el interruptor y en función de esta posición se cambia el valor de la variable modo. Además se apaga el LED contrario al *modo* en que se encuentra el sistema.

Para una correcta comunicación, es importante conocer los mensajes que se van a enviar entre los distintos nodos y la información que porta cada uno de ellos. En la tabla 5.8 se muestra la clasificación de mensajes por identificador CAN [6]. Se usan tramas CAN estándares, las cuales tienen identificadores de 9 bits y un campo de datos de 64 bits, divididos en 2 mensajes de 32 bits. En este proyecto cada mensaje se construye con diferentes datos por ejemplo en el que tiene identificador 0x110, donde *JoyX* ocupa 16 bits, *velI* ocupa 8 bits y *velD* 8 bits. El espacio que ocupan en la tabla es proporcional al numero de bits que poseen las variables.

Emisor	Identificador	Mensaje A			Mensaje B	
Joystick	0x110	JoyX	velI	velD	JoyY	Modo
	0x111	modo (leído por PC)				
Motores	0x101	encoderDabs			tDabs	
	0x102	encoderIabs			tIabs	
PC	0x120	datoD	datoI		Lazo	

Figura 5.8: Estructuración de información en las tramas CAN

5.4 Algoritmos y flujogramas

Para facilitar la lectura ordenada de la estructura y secuencia del código del "nodo de joystick", en esta sección se reunen todos los cuadros de algoritmos y flujogramas mencionados en este capítulo.

Algoritmo 5.1: Función de configuración del sistema

Result: Configuración de pines y deshabilitación de interrupciones
Function *void config(void)* **is**
    $IODIR0| = (1 << 10)|(1 << 15)|(1 << 17)|(1 << 21);$
    $IOSET0| = (1 << 15);$
    VPBDIV = 1;
    VICIntEnClr = 0xFFFFFFFFL;
    VICIntSelect = 0x00000000L;
end

Algoritmo 5.2: Función de configuración de la interfaz CAN1

Result: CAN1 en el vector 0 de IRQs, error en el vector 2 y velocidad de $1\,\mathrm{Mb\,s^{-1}}$

```
Function void init_can (void) is
    int i;
    FullCAN_Init(1 , 0 , CANBitrate001m_12MHz);
    FullCAN_SetErrIRQ(2);
    for(i=0 ; i<200000 ; i++);
end
```

Algoritmo 5.3: Funciones del timer del nodo joystick

Result: Rutina de interrupción y función de activación del timer del nodo joystick

```
Function void tc0(void) __irq is
    T0IR = 1;
    gTimerTick++;
    if Active_Ts then
        Timer_s–;
    end
    if Timer_s == 0 then
        Active_Ts = 0;
        Adquiere_Analogicos();
        Ts = 1;
    end
    VICVectAddr = 0xFFFFFFFF;
end
Function void Set_Ts(void) is
    Timer_s = Tsticks;
    Ts = 0;
    Active_Ts = 1;
end
```

Algoritmo 5.4: Función para adquirir los valores analógicos

Result: Toma los valores analógicos del joystick y potenciómetro y los convierte a digitales

```
Function void Adquiere_Analogicos(void) is
    ADCR = 0x012E0401;
    while (valorpot & 0x80000000) == 0 do
        valorpot = ADDR;
    end
    ADCR &=  0x01000000;
    valorpot = (valorpot >> 6) & 0x03FF;
    ADCR = 0x012E0402;
    while (valorx & 0x80000000) == 0 do
        valorx = ADDR;
    end
    ADCR &=  0x01000000;
    valorx = (valorx >> 6) & 0x03FF;
    ADCR = 0x012E0404;
    while (valory & 0x80000000) == 0 do
        valory = ADDR;
    end
    ADCR &=  0x01000000;
    valory = (valory >> 6) & 0x03FF;
end
```

Algoritmo 5.5: Conversión joystick-velocidad y envío de datos en función del modo

```
Result: Ejecución de unas funciones u otras en función del modo en el que se encuentre el sistema
switch modo do
    case MODO_JOYSTICK do
        Conversion_Joystick();
        Can_Envio();
        IOSET0|=(1<<17);
    case MODO_PC do
        velI=0;
        velD=0;
        Can_Envio();
        IOSET0|=(1<<21);
    end
end
```

Algoritmo 5.6: Conversión joystick-velocidad y envío de datos en función del modo

```
Result: Ejecución de unas funciones u otras en función del modo en el que se encuentre el sistema
Function void Can_Envio(void) is
    short poty,potx,modo_envio;
    FULLCAN_MSG MsgBuf;
    velI=(char)realI;
    velD=(char)realD;
    poty=(char)valory;
    potx=(short)valorx;
    modo_envio=(short)modo;
    MsgBuf.Dat1 = 0x00080110L;
    MsgBuf.DatA = (((potx << 16) & 0xFFFF0000)|((velI << 8) & 0x0000FF00));
    MsgBuf.DatA = (MsgBuf.DatA | (velD & 0xFF));
    MsgBuf.DatB = (((poty << 16) & 0xFFFF0000) | (modo_envio & 0xFFFF));
    FullCAN_PushMessage(1 , &MsgBuf);
end
```

Algoritmo 5.7: Secuencia para determinar el modo de funcionamiento

```
Result: Cambiar de modo y apagar el LED opuesto
if (IOPIN0 >> 16) & 1 then
    modo = MODO_JOYSTICK;
    IOCLR0 |= (1 << 21);
end
if (IOPIN0 >> 20) & 1 then
    modo = MODO_PC;
    IOCLR0 |= (1 << 17);
end
```

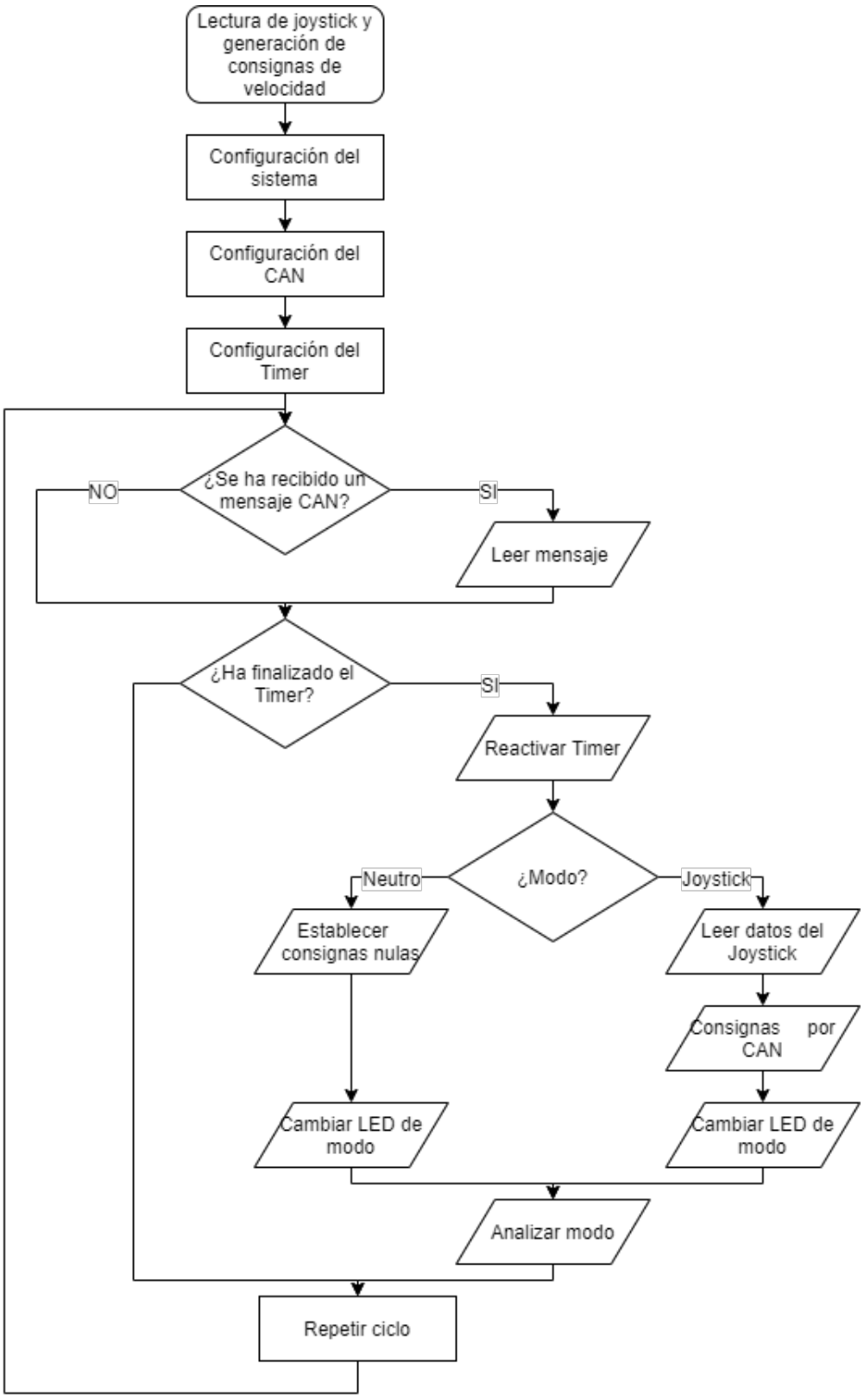

Figura 5.9: Flujograma del código del nodo joystick

Capítulo 6

Navegación

6.1 Introducción

En este capítulo se va a explicar el procedimiento seguido para desarrollar el sistema de navegación por joystick integrado en la plataforma.

Además se realiza un estudio de la silla como sistema robot para comprender los movimientos que puede realizar y las posibilidades de navegación que existen.

Se engloba tanto los elementos hardware presentes en la plataforma, tales como el joystick y el regulador de velocidad, así como los elementos software que se han utilizado y las estrategias de aceleración seguida para implementar dicho sistema de navegación.

6.2 Cinemática de la silla

Como preámbulo a este capítulo se explica la cinemática de la silla. Este estudio justifica la forma en que se aplican las consignas de velocidad y cómo se extrae y trata la información de los encoder para conocer la posición de la silla.

En esta plataforma, la cinemática consta de dos partes. La primera, en la cual se aplican las consignas de velocidad angular de las ruedas se denomina cinemática directa. La segunda parte trata de obtener la posición del robot en función de lo que ha avanzado cada rueda; esto se consigue mediante la lectura de los pulsos de encoder, y se denomina cinemática inversa u odometría.

Para poder aplicar las descripciones y ecuaciones de cualquiera de las dos partes, es necesario conocer la estructura y configuración de la silla. La plataforma consta de dos ruedas motrices separadas por una cierta distancia y con un cierto radio, por lo que cumple con el modelo cinemático de tracción diferencial. Estas distancias son importantes para los cálculos a realizar posteriormente. En la figura 6.1 se puede observar dicha configuración y los parámetros importantes para la aplicación. En el caso de KATE la distancia D es de 530 mm y la distancia R es de 90 mm.

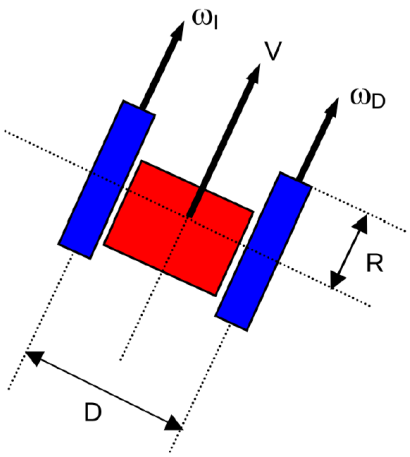

Figura 6.1: Parámetros del modelo cinemático de tracción diferencial [2].

6.2.1 Cinemática directa

La cinemática directa determina qué velocidad se debe aplicar a cada una de las ruedas motrices ω_D, rueda derecha y ω_I, rueda izquierda, para conseguir una determinada velocidad lineal V y angular Ω del móvil.

Dichas velocidades son descritas por las ecuaciones 6.1 y 6.2, las cuales dependen del radio de las ruedas, la distancia entre ruedas, la velocidad lineal deseada y la velocidad angular deseada.

$$\omega_D = \frac{1}{R} \cdot \left(V + \Omega \cdot \frac{D}{2} \right) \tag{6.1}$$

$$\omega_I = \frac{1}{R} \cdot \left(V - \Omega \cdot \frac{D}{2} \right) \tag{6.2}$$

Estas expresiones son usadas en el código de navegación con el joystick, el cual se explica en el apartado 6.3. En dicho código, se calcula la velocidad lineal y angular de la plataforma $[V, \Omega]$, se traduce a velocidades angulares de cada rueda $[\omega_D, \omega_I]$ y se envía por el bus CAN del sistema.

6.2.2 Odometría

Gracias a la odometría se puede conocer la posición de la plataforma a partir de las velocidades angulares de las ruedas. Estas velocidades angulares son calculadas a partir de los pulsos de encoder leídos en un determinado rango de tiempo. Para conocer la posición es útil obtener la velocidad lineal V y la velocidad angular Ω, que a partir de las expresiones 6.1 y 6.2 resultan ser las expresiones 6.3 y 6.4.

$$V = \frac{R}{2} \cdot (\omega_D + \omega_I) \tag{6.3}$$

$$\Omega = \frac{R}{2} \cdot (\omega_D - \omega_I) \tag{6.4}$$

En la figura 6.2 se puede observar el diagrama esquematizado de un robot diferencial, la silla, junto con los parámetros necesarios para determinar, mediante la cinemática inversa, la posición en X e Y y orientación dentro de un plano.

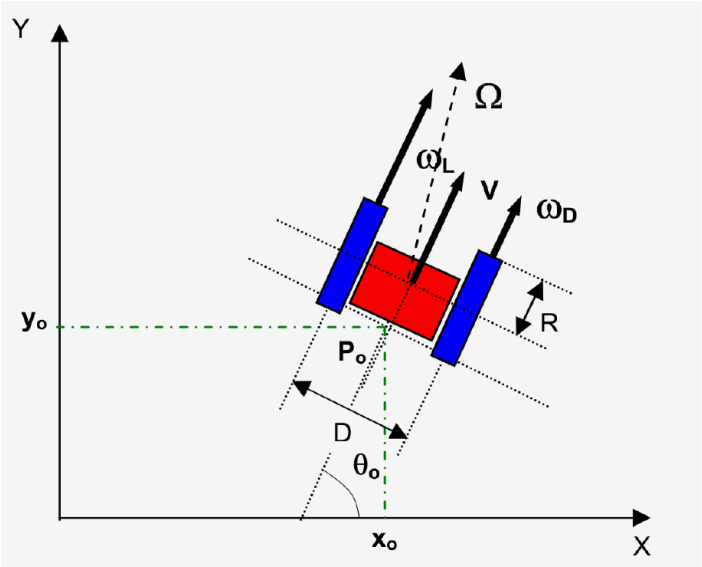

Figura 6.2: Modelo cinemático inverso de un robot diferencial: posicionamiento en 2D parámetros $[X, Y, \theta]$ [2].

La adquisición de los pulsos de encoder y el cálculo de la velocidad de cada una de las ruedas se explica en el apartado 4. Dichos datos se pueden utilizar para aplicar la cinemática inversa en la silla, sin embargo en este trabajo no se realiza.

6.3 Navegación con joystick

En esencia, el objetivo de este TFG es desarrollar una plataforma móvil que se pueda desplazar mediante el uso de un joystick analógico. La extracción y procesamiento de los datos procedentes del joystick se ha explicado el apartado 5.1, por tanto a continuación se detalla cómo se van a utilizar dichos datos para conseguir el movimiento deseado de la silla.

Con el objetivo de que la navegación sea suave y fácilmente controlable por el usuario, se establece una estrategia de aceleración la cual permite aumentar la velocidad poco a poco en intervalos de tiempo dados. Es decir, el usuario establece una velocidad comandada con el Joystick y el sistema varía la velocidad que se envía desde el nodo joystick hasta el nodo motores en forma de incrementos.

Dichos incrementos se pueden modificar manualmente en el código para que el comportamiento sea más o menos brusco. Están determinados en función del tiempo de muestreo de los valores del joystick (en nuestro caso 100 ms). En la figura 6.3 se muestra esquemáticamente el comportamiento de este método.

Como se puede ver, cuando la velocidad comandada cambia su valor, la velocidad real que se va a enviar por el bus aumenta a intervalos, en este caso de 10 unidades. Además, como consideración adicional, se deben evitar oscilaciones cuando llega al valor de la velocidad comandada, dado que ésta no tiene un valor entero tal y como la velocidad real. Por ello cuando supera el valor de la velocidad comandada se satura.

Otro factor que se implementa es un sistema de parada de emergencia. Dicho sistema funciona cuando se está aplicando una velocidad en un sentido y rápidamente se comanda una velocidad de sentido contrario con el joystick. De este modo se activa un flag de parada y la velocidad queda bloqueada a cero. el flag se desactiva cuando se lleva el joystick a la posición de reposo de nuevo, pudiéndose usar con total normalidad. Todo esto se debe implementar por código. Se realiza en la función Conversion_Joystick la

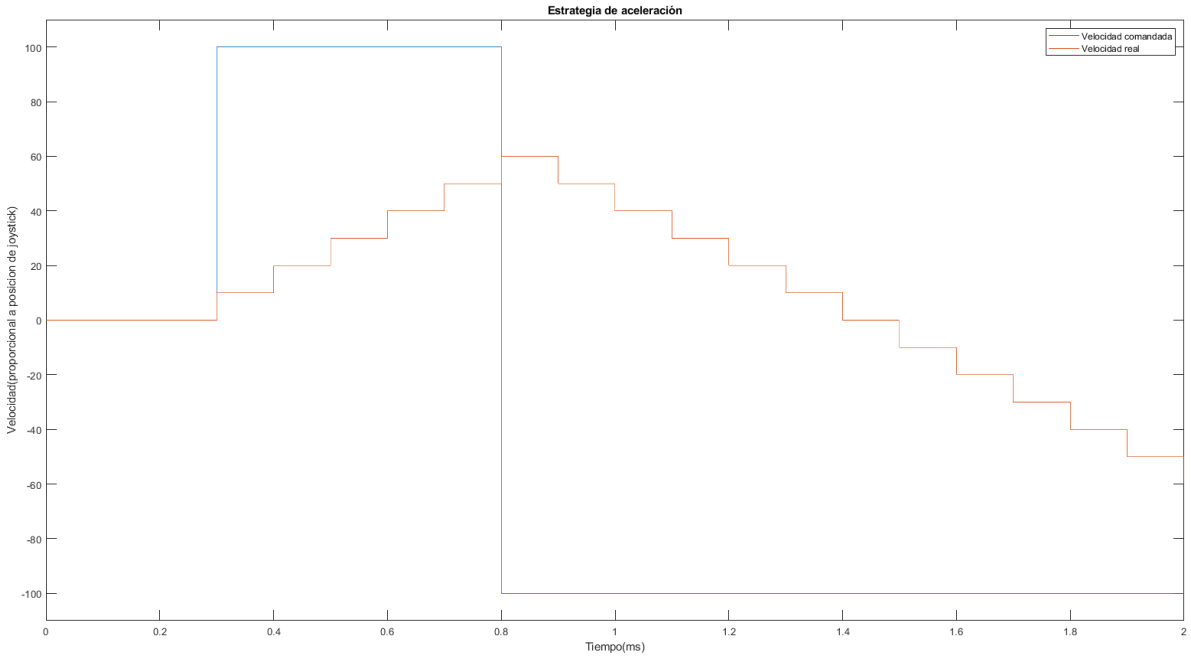

Figura 6.3: Esquema de la estrategia de aceleración

cual presenta varias partes las cuales se explican en el fragmento 6.1.

Algoritmo 6.1: Función para la conversión de valores de joystick a velocidad

Result: Convierte los valores digitales del joystick a consignas de velocidad
Function *void Conversion_Joystick(void)* **is**
    Declaración de incrementos;
    Implementación de zona muerta;
    Corrección de valores;
    Saturación de valores;
    Lógica de aceleración;
    Conversión de valores a velocidad;
    Activación de flag de parada;
end

Lo primero que se realiza, es la declaración de los valores de los incrementos y decrementos de la velocidad. Estos valores se encuentran definidos en función de los intervalos temporales entre los cuales se realizan las conversiones del joystick, para conocer de esta forma cuánto se incrementa en cada iteración.

Algoritmo 6.2: Segmento de declaración de valores incrementales

Result: Define los valores de incremento y decrementos de velocidad
incrementoV= 5*TS_MSEGUNDOS/100;
decrementoV= 10*TS_MSEGUNDOS/100;

Así como se comprueba en las lineas del código 6.2, cuanto menor sea el valor, más suave sera el movimiento, por esta razón se establece la aceleración mas lenta que la deceleración para que en el caso de que se quiera frenar la silla, se realice de forma más rápida. Los valores finales se han determinado de forma experimental.

En el fragmento de código 6.3 se muestra como se establece la zona muerta de los valores leídos del joystick. Dicha zona muerta hace que los valores cercanos al punto central (reposo) del joystick sean nulos, para evitar oscilaciones y movimientos no deseados.

Algoritmo 6.3: Segmento donde se establece zona muerta

```
Result: Establece un rango de valores cercanos a cero donde el valor del joystick es nulo
if (valorx > 770) && (valorx < 810) then
    ejex=0;
else
    ejex=valorx-785;
end
if (valory > 600) && (valory < 900) then
    ejey=0;
else
    ejey=valory-785;
end
```

Los valores del eje X del joystick (movimiento hacia delante o hacia atrás) recibidos varían desde 515 hasta 1023, por tanto se establece la zona muerta entre los valores centrales, de 770 a 810, y en caso contrario se hace que los valores en el eje X varíen desde -238 a 238.

En el caso del eje Y (movimiento izquierda o derecha) los valores recibidos varían desde 500 hasta 1023. La zona muerta que se establece es mayor, valores desde 600 a 900, debido a que cuando se está circulando resultaba muy fácil que la silla se desviara a los lados. De esta manera hay que desviar mucho más el joystick a los lados para conseguir girar.

La siguiente acción que se realiza es corregir los valores del joystick en función de los datos recibidos por los sensores de colisión (si existen), tal y como se realiza en SARA. En esta plataforma no se implementa esta opción, debido a que aun no se cuenta con la presencia de sensores para realizar dicha corrección. La saturación de valores se realiza con el fragmento de código mostrado en el listado 6.4.

Algoritmo 6.4: Segmento donde se establece la saturación de valores

```
Result: Saturar los valores de velocidad comandada para evitar oscilaciones en valores próximos
        al objetivo
if abs(correctV-realV) < decrementoV then
    correctV=abs(correctV-realV);
    decrementoV=incrementoV;
end
if abs(correctV-realV) < decrementoV then
    incrementoV=abs(correctV-realV);
    decrementoV=incrementoV;
end
```

Primero se comprueba si la diferencia entre el valor comandado y el real es menor que un decremento. Si es así se modifica el valor de la velocidad comandada a dicha diferencia y el del decremento al valor del incremento. A continuación se comprueba si este mismo valor es menor que la diferencia se modifica el valor del incremento y decremento.

Uno de los últimos pasos es aplicar la lógica de aceleración, la cual se puede observar en el código 6.5, donde se pueden observar las diferentes situaciones en las que se puede encontrar la plataforma y en

función de esas situaciones realizar un tipo de incremento u otro.

Algoritmo 6.5: Segmento donde se realiza la lógica de aceleración

```
Result: Aumenta o disminuye la velocidad comandada a intervalos
if (correctV >= 0) && (realV >= 0) then
    if correctV > realV then
        realV+=incrementoV;
    else if correctV < realV then
        realV-=decrementoV;
else if correctV < 0 && realV > 0 then
    realV-=decrementoV;
else if correctV > 0 && realV < 0 then
    realV+=incrementoV;
else if correctV <= 0 && realV <= 0 then
    if correctV < realV then
        realV-=incrementoV;
    else if correctV > realV then
        realV+=decrementoV;
```

Como se puede observar se tienen en cuenta las siguientes situaciones:

- Velocidad comandada positiva y velocidad real positiva.
  - Comandada mayor que real.
  - Real mayor que comandada.
- Velocidad comandada negativa y velocidad real positiva.
- Velocidad comandada positiva y velocidad real negativa.
- Velocidad comandada negativa y velocidad real negativa.
  - Comandada mayor que real.
  - Real mayor que comandada.

En función de cada situación se aplican incrementos y decrementos dados para conseguir que la velocidad real sea igual a la comandada y convertirse en la parte de código 6.6.

Algoritmo 6.6: Segmento donde se convierte valores a velocidad angular

```
Result: Convierte los valores del joystick a valores de velocidad angular para las ruedas
wd=((valorpot*1000)/(RADIO*990))*(realV-((correctR*DISEJES)/2000));
wi=((valorpot*1000)/(RADIO*990))*(realV+((correctR*DISEJES)/2000));
realD=wd/20;
realI=wi/20;
realD=Saturacion(realD,-127,127);
realI=(Saturacion(realI,-127,127));
```

En este punto se han aplicado los conceptos de cinemática directa explicados anteriormente, aplicando las expresiones 6.1 y 6.2. Cabe comentar que se divide entre 2000 y no entre 2 debido a que las unidades se han declarado en milímetros.

Además se realiza la limitación de velocidad en función de la posición del potenciómetro.La variable valorpot tiene un rango de valores desde 13 a 990, por tanto para limitar la velocidad se divide entre 990 para que dicho valor sea equivalente a un valor porcentual entre 0 % y 100 %.

Por último, una vez calculado el valor de velocidad, se reduce para que sus valores se puedan codificar en 8 bits y además se satura a ±127. También se puede reducir el valor de velocidad de algunas de las ruedas, debido a que experimentalmente se comprueba que la silla se desvía hacia la derecha debido a un defecto de fabricación del prototipo. De este modo al circular la rueda izquierda más despacio se corrige dicha desviación y se consigue una trayectoria recta. Este proceso de calibración se realiza posteriormente.

Falta tan solo el segmento 6.7 que se encarga de activar la parada de emergencia en caso de ser necesario. Esta parada se produce cuando los valores de velocidad comandada y la real son contrarios. Entonces anula todos los valores y se activa el flag de parada. Dicho flag condiciona el comportamiento de toda la estrategia de aceleración.

Algoritmo 6.7: Segmento donde se activa la parada

```
Result: Activa la parada en caso de que se de la situación especifica
if (realV > 0 && inputV < 0) —— (realV < 0 && inputV > 0) then
    realV=0;
    realD=0;
    realI=0;
    Parada=1;
end
```

Capítulo 7

Resultados

7.1 Introducción

En este capitulo se van a recopilar los resultados obtenidos al final del trabajo, mostrando la plataforma montada y demostrando su funcionamiento. Básicamente este capitulo se divide en los siguientes puntos:

- Montaje de la plataforma.
- Diseño y prueba del controlador de velocidad.
- Prueba de navegación con joystick.
- Medida de velocidad con PC.

7.2 Montaje de la plataforma

En este punto se muestra el montaje final de la plataforma con todos los elementos hardware. Todas las partes que integra la silla han sido explicadas en el capítulo 2. En la figura 7.1 se muestra como queda construida KATE.

Figura 7.1: Montaje final de KATE

Toda la electrónica del nodo motores queda integrada en un cajetín en la parte trasera de la silla, con la única apertura al hub de conectores RJ45 donde se podrán conectar el resto de nodos como el joystick o en un futuro el PC. La silla se encuentra amarrada al chasis, mientras que el mando del joystick lo sostendrá el usuario para su manejo.

Durante el montaje se soluciona un problema de excentricidad en la rueda derecha que causaba perturbaciones en su velocidad cada vez que daba una vuelta.

7.3 Prueba del controlador de velocidad

Como se explica en el punto 3.3.1.1, se ha diseñado un controlador PI con anti-windup para controlar la saturación del sistema y controlar la velocidad. Tras modelar la planta a partir de sus parámetros de la hoja de características y conseguir la respuesta temporal deseada mediante el uso de la herramienta de MATLAB, la función de transferencia del controlador discreto queda como se muestra en la expresión 7.1.

$$H_z = 0{,}006 \cdot \frac{z - 0{,}5789}{z - 1} \tag{7.1}$$

Además de esto, se debe tener en cuenta la acción de anti-windup, el cual introduce la constante Ka que se aplica al error de saturación. Este diseño se puede observar en la figura 7.2.

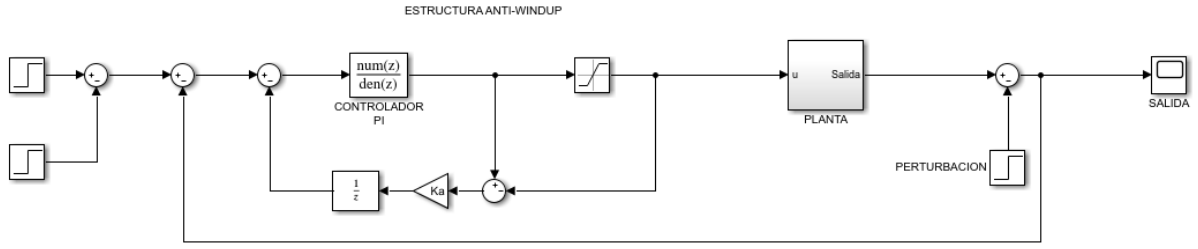

Figura 7.2: Modelo del sistema con anti-windup

Todo esto se debe escribir en código, lo cual se programa en el nodo de los motores tal y como se ve en el fragmento 4.10. Con todo el diseño realizado e implantado se prueba para observar el comportamiento real, aplicando tanto consignas negativas como positivas en ambas ruedas. Dichos resultados se observan en las figuras 7.3 y 7.4.

En este punto aun no se tiene en cuenta la rampa de aceleración implementada mas adelante, tan solo se observa el efecto del controlador ante consignas directamente aplicadas.

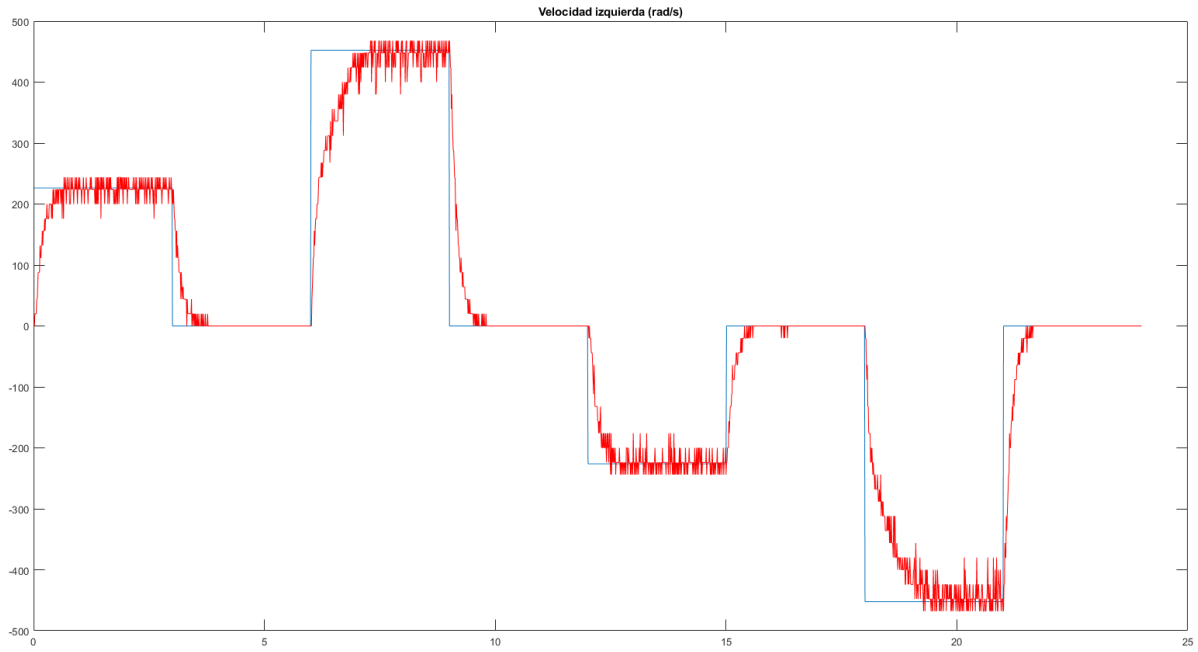

Figura 7.3: Medida de velocidad de la rueda izquierda

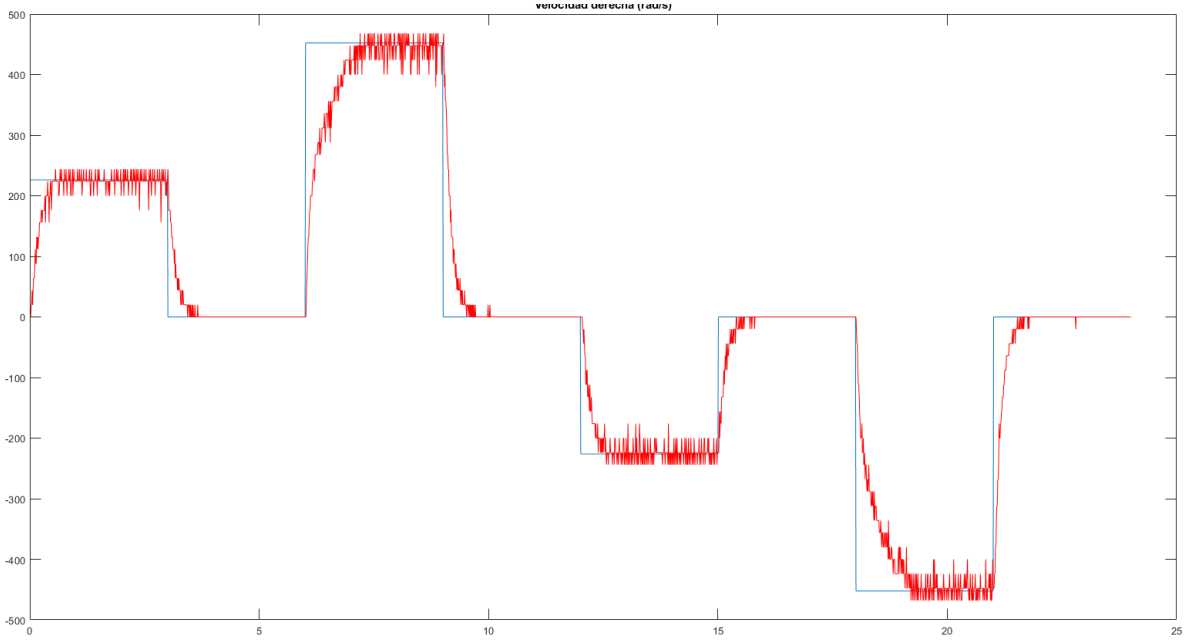

Figura 7.4: Medida de velocidad de la rueda derecho

7.4 Prueba de navegación con Joystick

Tras tener toda la plataforma montada, el código desarrollado y los microcontroladores programados, se realizan diversas pruebas de navegación para comprobar cómo es el control mediante el uso del joystick.

En una primera prueba se configuró la velocidad, debido a que ésta era muy elevada el control de la silla era muy difícil. Tras esta prueba se modificó el código que genera las consignas en el microcontrolador del joystick para que ésta fuera menor y fuera generada de forma gradual, debido a que la rampa de

aceleración no estaba funcionando correctamente. Además con el potenciómetro destinado a limitar la velocidad se consigue una velocidad adaptable al usuario, de tal forma que se pueda configurar de forma óptima.

Por otro lado, se producía una desviación hacia el lado derecho, a causa de que la rueda derecha circula a menos velocidad por deformación y excentricidad. De esta forma se puede introducir un factor de corrección a la rueda izquierda en posteriores pruebas de calibración.

Tras estas pruebas y modificaciones se puede observar en el siguiente vídeo una muestra de su funcionamiento:

Prueba de navegación con joystick de la plataforma KATE: https://youtu.be/Vtv6oAWaBnA

7.5 Medida de velocidad con PC

Por ultimo, a modo de comprobación del funcionamiento completo, se desea medir la velocidad de ambas ruedas usando el joystick y observando cómo responde en tiempo real.

Para poder obtener estas gráficas se hace uso del sistema en ROS [10] creado por Javier de la Roda [9] sobre SARA. Para poder medir la velocidad se utiliza un módulo de tipo .launch el cual lanza 3 nodos, como se puede ver en el fragmento de código 7.1.

Algoritmo 7.1: Archivo de lanzamiento para medir velocidad

Result: Lanzamiento de los nodos canusb, sara_control y rosbag

```
<launch>
 <param name="use_sim_time" value="false" />
 <include file="(find canusb)/launch/can_launch.launch/>
 <node name="SARA_interface" pkg="sara_control" type="hwinterface_script.py"
 output="screen"/>
 <node pkg="rosbag" type="record" name="rosbag_record_diag" args=" -o
 /home//sara//bagfiles/velocidad.bag /wheel_state" />
 </launch>
```

Como se puede ver, se lanza el nodo *canusb* [11] encargado de convertir la información que se envía y recibe por el usb del ordenador y transformarlo a tramas CAN. Posteriormente dichas tramas son desglosadas para poder obtener la información que contienen, que en este caso será la velocidad que es medida por el sistema PC.

Por otro lado se lanza el nodo llamado SARA_interface del tipo *hardware_interface* [12]. Dicho nodo ejecuta un script encargado de leer los topics de pulsos de encoder recibidos desde la plataforma física y convertirlos a velocidad. Dichas velocidades, de la rueda izquierda y derecha, son publicadas en el topic /wheel_state. En el script lanzado en este nodo se debe cambiar el numero de pulsos máximos obtenidos por vuelta, de 64.000 a 1425,2.

Por ultimo, se lanza el nodo *rosbag* encargado de grabar topics. En el caso de esta prueba se graba el topic /wheel_state debido a que es el que posee la velocidad calculada de ambas ruedas.

Una vez obtenido el archivo .bag, se debe extraer y representar la información que contiene. En este caso se utiliza un script de MATLAB, mostrado en 7.2. De esta forma se extraen los valores de velocidad de los topics grabados con el nodo rosbag y se representan, obteniendo la gráfica 7.5.

Como era de esperar, presentan mucho ruido o picos en la medición debido a la poca resolución que tienen los encoders. A pesar de este error de cuantificación, los datos obtenidos sirven para determinar que se elimina el error en régimen permanente ante perturbaciones.

Algoritmo 7.2: Script de Matlab para representar velocidades de un archivo .bag

```
Result: Extracción de las velocidades del archivo .bag
bag = rosbag('velocidad.bag');
wheel_state = select(bag,'Topic','/wheel_state');
wheel_state_struct=readMessages(wheel_state);
k=1;
x=1;
for i=1:length(wheel_state_struct)
    if (strcmp(wheel_state_struct{i,1}.Name{1,1},'LEFT'))
        wheel_state_data(k,1)=wheel_state_structi,1.Header.Stamp.Sec +
         wheel_state_struct{i,1}.Header.Stamp.Nsec*10^-9;
        wheel_state_data(k,2)=wheel_state_struct{i,1}.Velocity;
        wheel_state_data(k,5)=wheel_state_struct{i,1}.Header.Seq;
        k=k+1;
    elseif (strcmp(wheel_state_struct{i,1}.Name{1,1},'RIGHT'))
        wheel_state_data(x,3) = wheel_state_struct{i,1}.Header.Stamp.Sec +
         wheel_state_struct{i,1}.Header.Stamp.Nsec*10^-9;
        wheel_state_data(x,4)=wheel_state_struct{i,1}.Velocity;
        wheel_state_data(x,6)=wheel_state_struct{i,1}.Header.Seq;
        x=x+1;
    end
end
wheel_state_data=wheel_state_data(1:end-1,:);
```

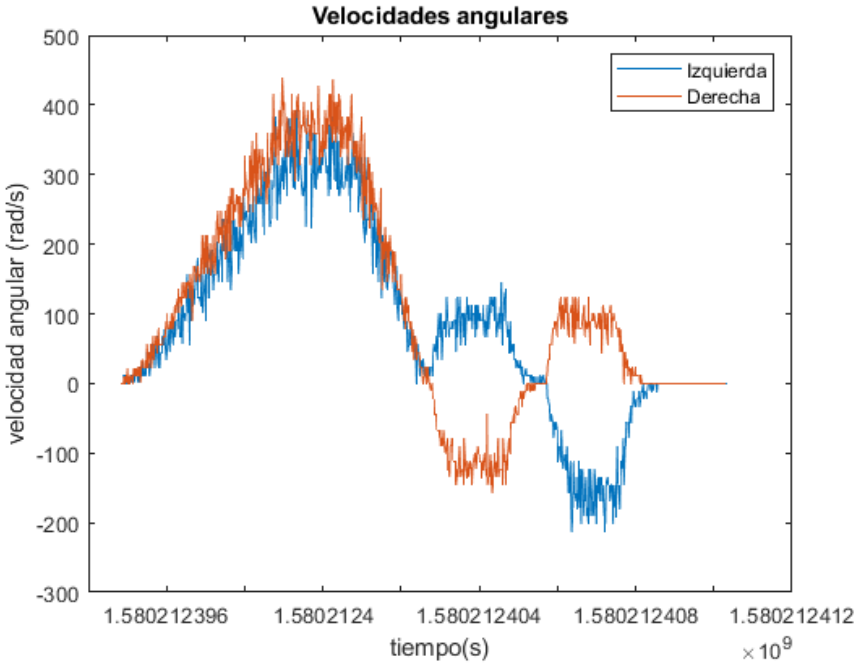

Figura 7.5: Representación de la velocidad real

En la primera parte de la gráfica se realiza un desplazamiento recto, por lo que ambas ruedas deben girar en el mismo sentido. Se puede ver que la velocidad de la rueda izquierda es menor que la derecha, por el factor de corrección introducido por código para compensar la desviación producida por la rueda derecha. El ascenso hasta la velocidad deseada (unos $350\,\mathrm{rad\,s^{-1}}$) es más lento que la bajada a la velocidad de reposo. De esta forma se puede comprobar que las rampas de aceleración y deceleración funcionan correctamente.

Por otro lado, en la segunda parte de la prueba se hace girar a la plataforma primero a la derecha y luego a la izquierda. Estas velocidades son mas inmediatas, debido a que los giros no presentan rampa de aceleración. Dichas velocidades se han obtenido de la expresión de la velocidad angular de la cinemática directa, visto en el capítulo 6, en función de la velocidad angular deseada.

Gracias a esta gráfica, se puede deducir que los resultados son buenos y se han cumplido con los objetivos planteados al principio de este TFG.

Todos los códigos y esquemáticos usados en este proyecto han sido subidos al Github [13].

Capítulo 8

Conclusiones y trabajos futuros

8.1 Conclusiones

En este TFG se fijó por objetivo crear una nueva plataforma móvil que sirva de apoyo a la movilidad a niños con discapacidad motriz, a partir de la estructura de otra plataforma ya existente perteneciente a la iniciativa Padrino Tecnológico [1], dotándola de un control en lazo cerrado en los motores y una HMI para poder ser controlada por el usuario.

Se ha respetado la estructura de modularización por nodos de la plataforma SARA, ya que se considera una forma muy eficiente y robusta de añadir nuevas funcionalidades a la silla de manera fiable, rápida y sencilla.

Se ha realizado tanto el desarrollo hardware como software de dos nodos diferentes para alcanzar dicho objetivo, los cuales se han denominado Nodo Motores y Nodo Joystick.

El Nodo Motores se encarga de aplicar consignas a los motores, realizar el control en lazo cerrado y tomar las lecturas de los encoder, además de encargarse del encendido de la silla. El Nodo Joystick es el encargado de administrar las ordenes del usuario tales como encendido, selección de modo, regulación de velocidad y navegación con joystick.

8.2 Trabajos futuros

A continuación se exponen algunas de las propuestas de lineas futuras que pueden partir como base de la nueva plataforma desarrollada en este trabajo:

- Dotación de un sistema sensorial, bien sea ultrasonidos o infrarrojos, que capacite a la plataforma de navegar de forma segura pudiendo evadir obstáculos y evitando situaciones de peligro o choque. Se propone para ello sensores de relativo bajo coste pero de prestaciones muy avanzadas, como:
  - RPLidar A1: escáner láser de 360 grados desarrollado por Slamtec [14].
  - Realsense D435: camara de profundidad desarrollada por Intel [15].
- Creación de un nuevo nodo que se encargue de realizar navegación autónoma de la plataforma, basado en un sistema operativo de tipo ROS. Para ello se propone utilizaar un controlador single board computer (SBC) de bajo coste como por ejemplo:

  - Raspberry Pi 4: ordenador de reducido tamaño y gran capacidad de procesamiento con hasta 8GB de memoria RAM [16].

- Incorporación de otros métodos de navegación tales como navegación por soplido o por joystick digital para aumentar la adaptabilidad de la plataforma al usuario.

- Incorporación de interfaces HMI gráficas y con pantalla táctil, para aprovechar las prestaciones avanzada de ROS sobre PC o SBC.

Capítulo 9

Pliego de condiciones

A continuación se detallan de forma específica los elementos mas importantes de la plataforma desarrollada, centrándose en sus especificaciones técnicas, y el software utilizado en este proyecto.

9.1 Elementos físicos

- Estructura mecánica
  - Dimensiones: 675 x 650 x 1000 mm.
  - Radio de las ruedas: 90 mm.
  - Distancia entre ruedas: 530 mm.
  - Reductora de engranajes planetarios. Reducción x50.9.
  - Batería. Dos baterías de plomo ácido de 12 V y 7.2 A h.
- Elementos electrónicos
  - Tarjeta microcontroladora LPC2129 CAN QuickStart.
  - Motores PG42775 RS-775244500.
  - Encoders ME-775.
  - Módulos driver LMD18200.
  - Joystick analógico inductivo APEM.
  - Mando para joystick, con lector de tensión para batería.
  - Material electrónico (resistencias, condensadores, transistores, etc.) indicados en los esquemáticos.
- Portátil Lenovo Yoga, para depuración y pruebas.

9.2 Software

- Procesamiento y análisis de datos: Matlab, versión 2019a, toolboxes Sisotool, Simulink y ROS.
- Paquetes de diseño de circuitos electrónicos: EasyEDA y OrCAD.

- Diagramas de flujo realizados con: https://app.diagrams.net/.
- Procesador de textos: OverLeaf (Latex).
- Sistema Operativo: Ubuntu 16.04 LTS.
- Versión de ROS: Kinetic.

Capítulo 10

Presupuesto

En este capítulo se incluye una estimación del coste total para la realización del proyecto. En las siguientes secciones los costes son divididos en función de su origen, exponiendo el subtotal en cada uno de los apartados y al final se añade el total de estas cifras.

10.1 Recursos Hardware

Se ha necesitado gran cantidad de recursos hardware para construir la plataforma y que funcione correctamente. Dichos recursos son enumerados en el siguiente listado.

- Chasis.
- Ruedas.
- Baterías.
- Motores y encoders.
- Piezas 3D.
- Material básico de electrónica (resistencias, condensadores, LEDs, etc).
- Drivers para motores.
- Tarjetas microcontroladoras LPC2129.
- Joystick analógico inductivo.

El precio y las unidades son incluidos en la tabla 10.1 junto con el total de los recursos hardware.

Concepto	Precio por Unidad	Cantidad	Subtotal
Chasis	50,00 €	1	50,00 €
Ruedas	10,00 €	4	40,00 €
Baterías	40,00 €	2	80,00 €
Motor y encoder	20,00 €	2	40,00 €
Piezas 3D	15,00 €	1	15,00 €
Material de electrónica	60,00 €	1	60,00 €
Drivers	30,00 €	2	60,00 €
LPC2129	50,00 €	2	100,00 €
Joystick	110,00 €	1	110,00 €
TOTAL			555,00 €

Tabla 10.1: Recursos hardware usados

10.2 Recursos de desarrollo y pruebas

Para llevar a cabo este trabajo es necesario el uso de determinado software, que involucra la programación de los nodos, el diseño electrónico de los mismos, la simulación de la plataforma, etc. A continuación se incluye un listado de los programas usados para dichas tareas. El precio se ha calculado en función de la vida útil del recurso (2 años) y el tiempo utilizado (2 meses).

- Matlab para diseño de controlador PI, representación de gráficas y pruebas.
- OrCAD para diseño de circuitos electrónicos.
- μVision de Keil para programación de microcontroladores.
- FreeCAD. Este programa es gratuito y se ha usado para diseño de piezas 3D y modelado de la plataforma.

En la tabla 10.2 se incluyen dichos programas con sus respectivos costes de licencia.

Concepto	Precio por Unidad	Coeficiente	Subtotal
Matlab	200,00 €	0,0833	16,66 €
μVision	400,00 €	0,0833	33,32 €
FreeCAD	0,00 €	0,0833	0,00 €
OrCAD	125,00 €	0,0833	10,41 €
Portatil Lenovo Yoga	700,00 €	0,0833	58,31 €
TOTAL			118,70 €

Tabla 10.2: Recursos de desarrollo y pruebas

10.3 Recursos Humanos

El coste proveniente a los recursos humanos del trabajo provienen de un ingeniero que desarrolla el trabajo completo. A continuación se incluyen dichos costes.

Concepto	Precio por hora	Cantidad de horas	Subtotal
Ingeniero	50,00 €	500	25000,00 €
TOTAL			25000,00 €

Tabla 10.3: Recursos humanos

10.4 Presupuesto de ejecución material

El presupuesto de ejecución material es la suma de los recursos hardware, software y humanos necesarios para llevar a cabo el trabajo.

Concepto	Subtotal
Recursos hardware	555,00 €
Recursos software	118,70 €
Coste mano de obra	25000,00 €
TOTAL	25673,70 €

Tabla 10.4: Presupuesto de ejecución material

10.5 Importe de la ejecución por contrata

Los costes de la ejecución por contrata deben incluir los gastos derivados del uso de las instalaciones donde se ha llevado a cabo el trabajo, las cargas fiscales, los gastos financiero, las tasas administrativas y las obligaciones de control del proyecto.

Dicho gasto se asume estableciendo un recargo sobre el coste del importe del presupuesto de ejecución material. Dicho recargo equivale al 22 % de dicho importe

Concepto	Subtotal
22 % del coste total de ejecución de material	5648,21 €

Tabla 10.5: Importe de ejecución por contrata

10.6 Honorarios facultativos

Se fija en este proyecto un porcentaje del 7 % sobre el coste total de ejecución por contrata.

Concepto	Subtotal
7 % del coste de ejecución por contrata	395,38 €

Tabla 10.6: Importe de los honorarios facultativos

10.7 Presupuesto Total

A continuación, en la tabla 10.7 se realiza la suma de todos los conceptos del presupuesto tenidos en cuenta en los anteriores apartados.

Concepto Precio	Subtotal
Presupuesto de ejecución material	25673,70 €
Importe de la ejecución contratada	5648,21 €
Horarios facultativos	395,38 €
TOTAL (sin IVA)	31717,29 €
IVA (22 %)	6977,80€
TOTAL	38695,10 €

Tabla 10.7: Importe del presupuesto total del proyecto

Bibliografía

[1] S. M. Bascón, "Plataforma móvil gram," https://padrinotecnologico.org/plataforma-gram/.

[2] M. M. Romera, "Sistema de posicionamiento absoluto de un robot móvil, función de sensado odométrico y visión de marcas artificiales." *Publicación*, 2002.

[3] ——, "Navegación autónoma de una silla de ruedas en interiores parcialmente estructurados." *Publicación*, 2000.

[4] P. del Moral Peña, "Control y guiado de una silla de ruedas en entornos interiores." *Publicación*, 2013.

[5] J. B. Álvarez, "Guiado de una silla de ruedas mediante joystick y soplido por bus can ." *Publicación*, 2009.

[6] Álvaro García Morcillo, "Diseño y desarrollo de módulos hardware/software para un sistema avanzado robótico de asistencia a la movilidad (sara)." *Publicación*, 2014.

[7] J. B. García, "Guiado semiautomático de una silla de ruedas comandada por un joystick de cabeza a través de bus can." *Publicación*, 2009.

[8] D. P. Ávila, "Diseño y desarrollo de módulos hardware/software para un sistema avanzado robótico de asistencia a la movilidad (sara)." *Publicación*, 2015.

[9] F. J. R. Herrero, "Módulos software para un sistema de asistencia a la movilidad basado en entorno ros." *Publicación*, 2018.

[10] "Pagina web de ros," https://www.ros.org/.

[11] "Repositorio del paquete canusb," https://github.com/spiralray/canusb.

[12] "Pagina web de hardware_interface," http://wiki.ros.org/hardware_interface.

[13] A. M. Carrero, "Github de kate," https://github.com/AlessandroMelino/SAR.

[14] "Página web de rplidar a1 de slamtec," https://www.slamtec.com/en/Lidar/A1.

[15] "Página web de cámara realsense d435 de intel," https://www.intelrealsense.com/depth-camera-d435/.

[16] "Página web de sbc raspberry pi 4," https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR