UNIVERSIDAD DE MÁLAGA

ESCUELA DE INGENIERÍAS INDUSTRIALES

DPTO. DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

UNIVERSIDAD
DE MÁLAGA

TRABAJO FIN DE GRADO

CONTROL DE UNA PLATAFORMA OMNIDIRECCIONAL PARA UN MANIPULADOR MÓVIL

GRADO EN INGENIERÍA DE TECNOLOGÍAS INDUSTRIALES

AUTOR: ALONSO SERRANO FLORES

TUTOR: JESÚS MANUEL GÓMEZ DE GABRIEL

MÁLAGA, JUNIO DE 2021

Declaración de autoría

Yo, Alonso Serrano Flores, con DNI 31012355-Y, alumno del Grado en Ingeniería de Tecnologías Industriales de la Escuela de Ingenierías Industriales de la Universidad de Málaga, declaro ser autor del texto entregado y que no ha sido presentado con anterioridad, ni total ni parcialmente, para superar materias previamente cursadas en esta u otras titulaciones de la Universidad de Málaga o cualquier otra institución de educación superior u otro tipo de fin.

Así mismo, declaro no haber transgredido ninguna norma universitaria con respecto al plagio ni a las leyes establecidas que protegen la propiedad intelectual, así como que las fuentes utilizadas han sido citadas adecuadamente.

Fdo: Alonso Serrano Flores
Málaga, Junio de 2021

Acrónimos

EII	Escuela de Ingenierías Industriales
PID	Proporcional Integral Derivativo
TFG	Trabajo Fin de Grado
UMA	Universidad de Málaga
RMR	Robots Móviles con Ruedas

Palabras Clave

- Cinemática
- Control
- Controladora
- Dinámica
- Hardware
- Mecanismo
- Mecanum
- Modelo
- Omnidireccional
- PID
- Plataforma
- Robot móvil
- Rueda
- Trayectoria

Resumen

Este trabajo se centra en el diseño y montaje de un robot omnidireccional con cuatro ruedas Mecanum posicionadas como las ruedas de un automóvil. Para ello se ha realizado un estudio del tipo de rueda que se iba a utilizar, y seguidamente un desarrollo de la cinemática de un robot configurado de la forma mencionada anteriormente. Seguidamente se ha fabricado y montado todas las partes mecánicas de la plataforma. Después de esto se ha elegido toda la electrónica en función de las funciones que queremos desempeñar con este robot y se ha realizado su montaje simultáneamente al desarrollo del software necesario.

Respecto al software se han desarrollado diferentes formas de manejo como se verán mas adelante. Finalmente se han realizado una serie de experimentos para verificar la validez del sistema creado y a su vez corregir errores.

Para finalizar el trabajo se han presentado unas conclusiones y se han propuesto ciertas mejoras de cara al futuro, de forma que el robot sea más eficiente y tenga mayor maniobrabilidad.

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción y visión general

Contenido

1.1. Introducción

En los últimos años la robótica se ha introducido en casi todos los campos de nuestro entorno, desde la domótica, hasta la medicina. Sus aplicaciones son innumerables, y cada día aparecen nuevas ideas en las que los robots se incorporan a los distintos ámbitos de nuestra vida [1].

En este entorno surgen los robots móviles. El principal objetivo a cumplir de un robot móvil es el de generar trayectorias viables y dirigir su movimiento. Para llevar esto a cabo exige varios niveles de procesamiento de la información, con distintos grados de complejidad. La solución más básica utiliza simplemente un modelo cinemático, que no tiene en cuenta los elementos de su entorno [2].

En el campo de la robótica móvil, con el objeto de mejorar la maniobrabilidad se han desarrollado nuevos tipos de ruedas dando lugar a los robots omnidi-

reccionales. Un robot omnidireccional, también conocido como holonómico [3], es un tipo de robot móvil con ruedas, cuya configuración le permite desplazarse en cualquier dirección sin la necesidad de alcanzar previamente una orientación específica. Es decir, es capaz de realizar movimientos en cualquiera de las componentes del plano, bien sean traslaciones (hacia adelante, en reversa, laterales) o rotaciones, a partir de un estado de movilidad. Todo esto se consigue a expensas de una mayor complejidad en su control [4] [5].

En este trabajo se va a utilizar un tipo de ruedas omnidireccionales, denominadas ruedas Mecanum, como las que se pueden ver en la siguiente imagen:

Figura 1.1: Rueda Mecanum.

El diseño de estas ruedas se basa en una rueda convencional que dispone de una serie de rodillos en su circunferencia. Los rodillos presentan una configuración de 45º respecto al plano de la rueda [6]. Asimismo, estos rodillos tienen una forma curvada de forma que, vista desde el lateral, la rueda mantiene un perfil circular. Y mediante cuatro de estas ruedas se diseñará el control del movimiento de un robot [7] [8].

1.2. Motivación del trabajo

En el entorno de la robótica doméstica hay mucho por avanzar aún, pero está claro que sea cual sea el objetivo de un robot este deberá tener el mayor número de funcionalidades posible de manera que se aproveche al máximo su capacidad. Este trabajo nace de la necesidad de que un brazo robótico destinado a labores

domésticas tenga la capacidad de desplazarse por lugares pequeños y simples como una casa. Esta simple característica aumentará mucho su funcionalidad.

1.3. Objetivos

Los objetivos de este trabajo son los que se enumeran a continuación:

1. Diseño e implementación del sistema de control de movimientos de una plataforma con ruedas omnidireccional
2. Elección de componentes e integración de la electrónica de control
3. Realización de pruebas de verificación y ajuste de controladores
4. Programación de un sistema de control de velocidad cartesiano
5. Programación de un sistema de obtención de datos de odometría
6. Programación de un sistema de teleoperación local inalámbrica
7. Realización de experimentos de validación.
8. Documentar el sistema, incluyendo una guía para los desarrolladores de niveles superiores de control de la plataforma

1.4. Fases del proyecto

El propósito de este trabajo es por tanto el de diseñar y construir un robot móvil omnidireccional que dote de la capacidad de movimiento al robot doméstico. Para ello se realizará un estudio de la cinemática de las ruedas y la contrucción y montaje tanto de la mecánica como de la electrónica. Se construirá una plataforma que contendrá todos los elementos y sobre la que irá el brazo robótico.

De forma análoga se desarrollará el software necesario para el control del robot mediante un microcontrolador Arduino mediante el entorno de Arduino IDE. Para su control se implementarán diferentes modos de control como serán el uso del puerto serie de Arduino, un joystick convencional basado en potenciómetros y un mando de una videoconsola común.

Finalmente se realizarás una serie de experimentos de forma que se definan ciertas trayectorias a seguir por el robot y a su vez tomemos los valores reales de desplazamiento, con el fin de que podamos comprobar la validez de los mismos.

1.5. Marco de realización

El grupo de investigación de Ingeniería de Sistemas y Automática de la Universidad de Málaga cuenta con una extensa experiencia en la investigación, contando con un amplio personal cualificado, numerosas patentes, laboratorios y equipamiento. Las líneas de investigación de este grupo van desde robots móviles autónomos y teleoperados hasta sistemas robóticos de asistencia a la cirugía pasando por sistemas de control inteligente y procesado de imágenes.

Este Trabajo Fin de Grado forma parte de un proyecto en el que se está desarrollando un brazo robótico con finalidades domésticas y de asistencia a personas. En este marco surge la necesidad de que dicho robot cuente con la capacidad de realizar trayectorias factibles en entornos sencillos como son los domésticos.

De esta manera, este trabajo ayudará a aumentar en un gran número las funcionalidades del manipulador.

1.6. Metodología

A continuación se enumeran los pasos seguidos para la realización de este trabajo, describiendo brevemente cada uno:

1. **Diseño y construcción de la plataforma.** Lo primero a realizar será diseñar y fabricar la plataforma, así como cada elemento de sujeción de los motores y la distribución de estos.

2. **Estudio cinemático de las ruedas Mecanum.** Obtención de un modelo del robot, de manera que dispongamos de las ecuaciones necesarias para convertir tres valores de velocidad (vx,vy y wz) en las cuatro velocidades de giro de los motores.

3. **Diseño del software en Arduino.** Se diseña el código necesario para el control del robot en el entorno de Arduino IDE. Dicho software será el encargado de leer los valores recibidos ya sea a través del puerto serie, del joystick o del mando de videoconsola, transformarlos en un vector de velocidad de tres componentes, a su vez transformarlo en las cuatro velocidades que deben tomar cada motor, y enviarlas a las controladoras. Paralelamente el microprocesador se encargará de ir midiendo valores reales de desplazamiento con el objetivo de que se sepa en todo momento la posición real.

4. **Fabricación y montaje de la electrónica.** Se fabrican todos los cables y conectores necesarios para el conexionado de las controladoras, el microprocesador, los encoders y la alimentación de los motores.

5. **Generación de un controlador adecuado para cada motor.** A partir de la interfaz de usuario de las controladoras de potencia se ajusta un controlador PID para cada motor, de manera que la respuesta sea la más adecuada. Se realizan pruebas de control mediante los tres modos de manejo y se corrigen errores de código.

6. **Realización de experimentos.** Se generan una serie de trayectorias a ser realizadas por el robot, con el objetivo de comparar la trayectoria ideal con la real y así demostrar la funcionalidad del robot móvil.

1.7. Estructura de la memoria

En este apartado se enumeran las partes del trabajo y se comentan los temas que se tratan en cada una.

Capítulo 1, Introducción. En esta primera parte del trabajo se hablará sobre la actualidad de la robótica móvil y se planteará el motivo y el objetivo del proyecto.

Capítulo 2, Estado del arte. En este capítulo se hace una revisión de los distintos métodos de planificación de trayectorias para así conocer las distintas alternativas y escoger alguna para utilizarla en el presente trabajo.

Capítulo 3, Fundamentos teóricos. En este capítulo se hablará sobre el tipo y configuración de rueda usada y se realizará el análisis cinemático de una plataforma con cuatro ruedas Mecanum.

Capítulo 4, Diseño y fabricación. Se explicará tanto la parte mecánica, como el hardware y el software diseñado para el robot.

Capítulo 5, Experimentos. En este capítulo se analizarán una serie de experimentos para corregir errores y para comprobar la validez del control creado.

Capítulo 6, Conclusiones y líneas futuras. En este capítulo resumen los resultados obtenidos y se proponen posibles mejoras de cara al futuro.

Capítulo 2

Contexto

Contenido

2.1. Introducción

En este capítulo se pretende mostrar una visión general de la robótica móvil en la actualidad y sus diferentes aplicaciones tanto ahora, como en un futuro próximo.

2.1.1. Actualidad

La tecnología mejora día a día debido a las nuevas necesidades que aparecen y a las exigencias de la sociedad. En los últimos años, la robótica ha implicado un gran cambio en campos como la medicina y sobre todo la industria [9]. La creación de robots médicos ha facilitado la realización de operaciones en las que el factor humano era muy determinante, y el incremento de producción y mejora de la industria debido a la robótica es notable y aumenta cada día.

En campos como estos es donde surge una de las tendencias mas populares de la actualidad, que es el uso de la robótica móvil. El uso de robots móviles aumenta en gran medida las funcionalidades de los robots, ya que estos pueden diseñarse para numerosas acciones al no tener que estar fijados en una

ubicación determinada, lo cual restringe enormemente sus capacidades.

Los primeros robots que se construyeron eran estáticos, debido a esto, su diseño se ajustaba a la función que debía desempeñar, pero con la llegada de los robots móviles comenzó a ser fácil desarrollar robots con diferentes propósitos, con herramientas intercambiables de manera que con un único diseño se consiguiesen numerosas labores. De esta forma, se está mejorando cada vez más el estilo de vida, y los procesos laborales.

Industria 4.0

En este punto, cabe mencionar también el desarrollo de la Industria 4.0, también conocida como Ïndustria Conectada", que ha consolidado el uso de la conectividad y la robótica colaborativa en los procesos industriales, creando espacios de trabajo en los que humanos y robots pueden trabajar juntos de manera segura, además de compartir información para optimizar los procesos, obteniendo así mejores resultados [10]. El mayor activo es el intercambio de información gracias a la integración de las últimas tecnologías inteligentes en la robótica, como el Internet de las cosas, Inteligencia artificial o Big Data. Todo esto se traduce en más eficiencia, mejor uso de los recursos y por tanto mayor productividad. La robótica móvil en combinación con esta tecnología ha marcado un nuevo comienzo en ámbitos de la industria y domésticos. Varios ejemplos en los que ha habido mejoras notables son en las tareas logísticas de pick and place, tareas de alimentacion de piezas, metrología, control de calidad, limpieza, pulido, atornillado, etc. Estas son tareas den las que la repetibilidad y la uniformidad de las acciones son clave para ahorras costes y agilizar los procesos.

Tres características muy importantes son:

1. **Automatización inteligente.** El robot es capaz de leer, generar y gestionar información, por tanto, además de no necesitar presencia humana para su funcionamiento, este es capaz de tomar decisiones. Esto es algo muy importante en el campo de la realización de trayectorias factibles, ya que el propio robot decidirá que camino seguir para llegar a un lugar, en función de obstáculos y de la manera más eficiente posible.

2. **Conectividad.** Se posibilita la comuncicación máquina a máquina, y a través de interfaces diseñadas para ello, la interacción con los humanos.

3. **Flexibilidad.** Como he comentado anteriormente, la robótica móvil, sumado a este nuevo tipo de industria, hace que un robot sea fácilmente adaptable a otra forma de trabajo u otra tarea, en función de las exigencias que

se necesiten en cada momento. Ejecutar diferentes tareas, adaptar su velocidad, crear rutas alternativas o cambiar de sección en tiempo real están dentro de las capacidades de un robot móvil con navegación inteligente [10].

Nuevos sistemas de desplazamiento

Paralelamente a todo esto, se han desarrollado y se sigue estudiando cada día en nuevos sistemas de desplazamiento, nuevas configuraciones de robots, que permitan mayor maniobrabilidad y faciliten la realización de mayor número de trayectorias. Esta será una parte muy importante de este trabajo.

Los robots móviles se pueden clasificar por el tipo de locomoción utilizado [11]; en general, los tres más usados son: por ruedas *a)*, por patas *b)* y por orugas *c)* 2.1. Si bien la locomoción por patas y orugas ha sido ampliamente estudiada, el mayor desarrollo está en los Robots Móviles con Ruedas (RMR), esto es debido a las ventajas que presentan las ruedas respecto a las patas y a las orugas. Dentro de los atributos mas relevantes de los RMR, destacan su eficiencia en cuanto a energía en superficies lisas y firmes, a la vez que no causan desgaste en la superficie donde se mueven y requieren un número menor de partes, normalmente menos complejas en comparación con los robots de patas y de orugas, lo que permite que su construcción sea más sencilla [11].

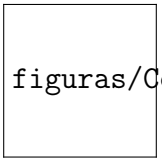


Figura 2.1: Tipos de locomoción en un robot móvil.

En función de la configuración cinemática, los RMR utilizan cuatro tipo de ruedas: omnidireccionales, convencionales, tipo castor y ruedas de bola. En los estudios del modelado, a fin de simplificar, se hacen ciertas suposiciones como son:

- Las partes dinámicas del RMR se consideran insignificantes.
- No contiene partes flexibles, así se pueden aplicar mecanismos de cuerpo rígido.
- Asumir que todos los ejes de dirección son perpendiculares a la superficie, así todos los movimientos se realizan en un solo plano.

Respecto a los actuadores más usados para generar el movimiento, se suelen usar motores de corriente continua, ya que su modelo es lineal, y facilita enormemente su control.

En la robótica móvil, lo más importante se puede resumir en el posicionamiento, seguimiento de una trayectoria y evasión de obstáculos. Respecto al posicionamiento, son muy usados los sensores de ultrasonidos. En [12] se pueden ver varios métodos de implementación en los que se usan este tipo de sensores. De igual forma, en [13] se aborda el problema de estimación del posicionamiento de un robot móvil, mediante la cuantificación de errores de odometría de tipo sistemáticos. En [14] se integran dos métodos de mapeo para la navegación de un robot dentro de un ambiente cerrado y estructurado, mapeo enrejado bidimensional y mapeo topológico, logrando una mayor precisión y eficiencia en el desplazamiento del robot.

En lo que respecta a la evasión de obstáculos, existen distintos métodos para llevar a cabo esta tarea. Los más relevantes son: por detección de bordes [15], por descomposición en celdas [16], construcción de mapas [17] y campos potenciales [18].

2.2. Conclusión

Como conclusión de este capitulo podemos decir que los avances en la robótica móvil en las últimas décadas han sido impresionantes, y que actualmente, está siendo incorporada totalmente en la vida cotidiana, desde la industria hasta la medicina, y aunque mas rezagado, también en el entorno doméstico.

Sumado a las nuevas tecnologías de comunicación, la Industria 4.0, el internet de las cosas, Big Data, etc, va a seguir suponiendo una auténtica revolución en todos los aspectos de nuestra vida, introduciendo la robótica cada vez más en cada tarea de nuestro día a día.

Capítulo 3

Fundamentos teóricos

Contenido

3.1. Introducción

En este capítulo se describen los fundamentos teóricos que han sido estudiados para el desarrollo de la plataforma móvil omnidireccional. El objetivo en esta fase del trabajo era el de elegir los componentes físicos de la plataforma y diseñar sus dimensiones.

3.2. Tipo de rueda

Uno de las principales cuestiones a abordar fue la elección del tipo de rueda más adecuada para la finalidad que buscamos. Para diseñar un robot móvil hay que tener en cuenta las funciones que debe llevar a cabo para de esta manera determinar el diseño y componentes que mas se ajustan a los objetivos. En

nuestro caso la función más importante y que más nos va a limitar a la hora del diseño de rueda es la movilidad omnidireccional, es decir, el desplazamiento en cualquiera de las direcciones del plano.

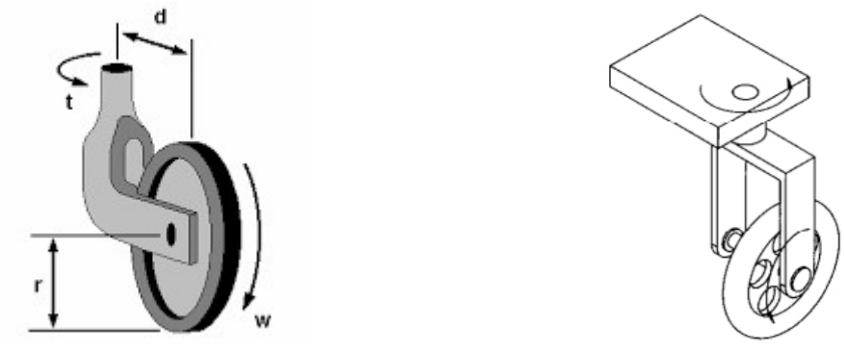
Al comienzo del trabajo se plantearon una serie de configuraciones y se hizo una valoración de cada una, de cara a elegir la mas adecuada.

3.2.1. Ruedas disponibles

Las ruedas de un robot omnidireccional son el elemento más importante, ya que es el que permitirá que la plataforma realice un conjunto de desplazamientos determinado sin necesidad de giro. Una de las funcionalidades que más destacará será el movimiento simultáneo de traslación y rotación. Esto supone una simplificación de la trayectoria seguirá por un robot ya que permite establecer tanto la posición como la orientación en un mismo movimiento. Esto para un robot doméstico como es el caso permitirá un mayor número de trayectorias posibles.

Ruedas convencionales

Este tipo de ruedas, combinadas de la manera adecuada permiten el desplazamiento en cualquiera de las direcciones, incluyendo la rotación sobre si mismo. Los dos tipos mas comunes son los siguientes:



(a) Tipo 1.

(b) Tipo 2.

Figura 3.1: Ruedas convencionales.

En la siguiente figura se muestra una posible configuración con este tipo de ruedas, de manera que se posibilita cualquier desplazamiento y en plano [19]:

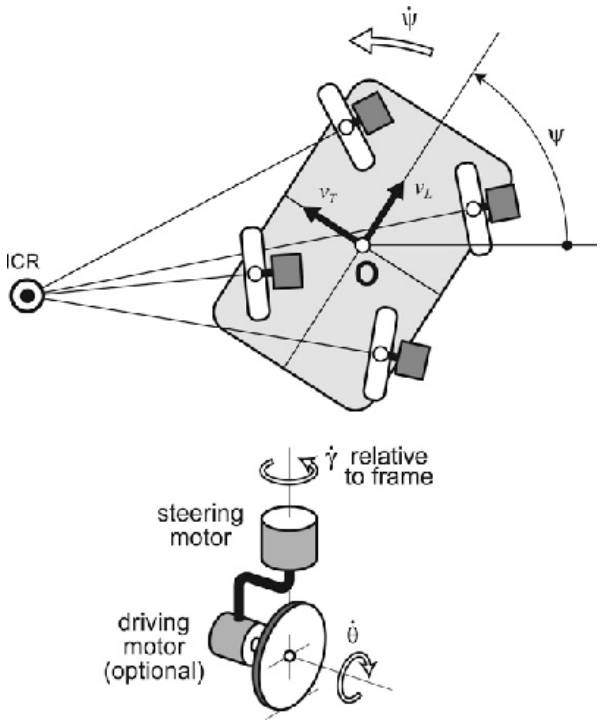


Figura 3.2: Esquema robot omnidireccional con ruedas convencionales.

Este solo es un ejemplo, pero existen numerosas convinaciones más, tanto de 3, como de 4 ruedas. El principal problema que presentan los robots móviles con estas configuraciones es que no llegan a tener una controlabilidad total de su movimiento, ya que a la hora de realizar cambios bruscos en la trayectoria, no son capaces de realizar el cambio en un ángulo recto, si no que describirían una curva. Esto es debido a que para ese cambio de dirección, las ruedas deben tomar la nueva dirección, lo cual implica un intervalo de tiempo hasta que quedan orientadas correctamente. Otro gran problema de este diseño, es que para conseguir la mayor maniobrabilidad posible, cada rueda debe contar con dos motores como se ve en la figura 3.2, ya que necesita uno para el desplazamiento y otro para la orientación.

Ruedas especiales

Las ruedas especiales juegan con la pasividad, de manera que generando fuerza en unas y aprovechando la pasividad de otras se pueden generar una gran cantidad de movimientos. Esto es gracias a que con estas configuraciones pasamos a tener un mayor número de grados de libertad controlables. En este apartado vamos a comentar las mas usadas, que son las ruedas universales, las

ruedas Mecanum y las esféricas.

- **Ruedas universales** Este tipo de ruedas también se les denomina omnidireccionales. Están formadas por una rueda principal, la cual tiene en su circunferencia una serie de rodillos pasivos, que no generan velocidad por si solos, pero que gracias a ellos y a la combinación de velocidades del conjunto de todas las ruedas del robot se consiguen una infinidad de movimientos en el plano. Al disponer de estos rodillos, cada rueda tiene 3 grados de libertad (girar sobre su eje, sobre el eje vertical y además el desplazamiento lateral gracias a los rodillos).

Figura 3.3: Rueda universal.

Este tipo de ruedas presenta ciertas ventajas y desventajas:

- Desventajas:
  - Al disponer de estos rodillos en su perímetro, la superficie de contacto se reduce a un punto, de ahí que se reduzca la carga máxima soportada por la plataforma.
  - El hecho de tener su superficie compuesta de esos rodillos hace que sea irregular. Esto puede llegar a provocar vibraciones en mayor o menos medida dependiendo de la cantidad de rodillos que formen la superficie. Esto se reduce con diseños de de doble o triple rueda en paralelo.
  - Con este tipo de ruedas, al realizar movimientos laterales tendremos el problema de que al encontrarnos cualquier obstáculo este será más difícil de sobrepasar, debido a que no estará influenciado por el radio de la rueda principal, si no por el radio de los rodillos de su superficie.

- Ventajas:
  - Una de las principales es la simplicidad en el diseño, ya que se pueden construir robots con únicamente 3 ruedas y que tengan todas las funcionalidades de un robot omnidireccional.
  - Aunque la reducción de superficie sea una desventaja a la hora de la carga, también es una gran ventaja, ya que la fricción se reduce enormemente.

- **Ruedas Mecanum** Este tipo de ruedas es muy similar a las ruedas universales. La diferencia esta en que los rodillos de su superficie, en lugar de estar alineados con el plano de la rueda, forman 45 grados con este. Asimismo, estos rodillos tienen una forma curvada de forma que, vista desde el lateral, la rueda mantiene un perfil circular [6].

Figura 3.4: Rueda Mecanum.

El fundamento de estas ruedas es que la fuerza, gracias a los rodillos, se transmite a 45 grados respecto a los ejes principales de la rueda. como podemos ver en la siguiente imagen, la fuerza se descompone en dos componentes, pero solo la componente F_A genera movimiento.

Para aprovechar al máximo este tipo de ruedas la combinación más adecuada es usar 4 ruedas, colocadas de manera similar a las ruedas de un coche. Esto, combinado con la velocidad independiente de cada motor hace que se pueda conseguir cualquier dirección

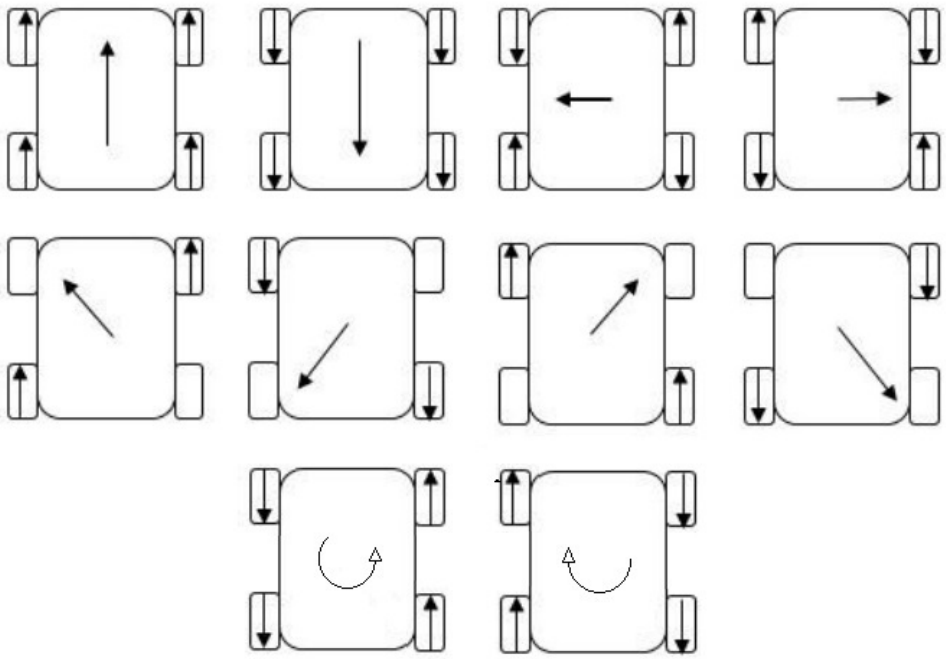


Figura 3.5: Diferentes movimientos de la plataforma.

Cabe destacar que este tipo de diseño presenta un gran inconveniente, reflejado en una perdida de fuerza. Como hemos comentado anteriormente la componente F_B no genera moviemiento, es una fuerza desaprovechada. Esto a su vez conlleva que la velocidad alcanzable sea menor. Este tipo de rueda presenta las mismas ventajas y desventajas que las ruedas universales, con el añadido de que el diseño es más complejo. Además esta configuración de ruedas frente a la configuración clásica, como por ejemplo un coche, hace necesario que cada rueda tenga su propio motor, por lo cual pasaríamos de tener un único motor a tener 3 o 4 en función del diseño del robot. Este diseño también presenta límite menos de carga y ciertas vibraciones, pero debido a la transferencia más suave de las superficies de contacto un se pueden soportar cargas más altas en comparación a las ruedas universales.

- **Ruedas oruga** Las orugas constituyen una alternativa sólida a otro tipo de sistemas y desde principios del siglo XX han demostrado sus bondades en vehículos tripulados. Son un tipo de rueda muy usado para condiciones exteriores, con terrenos en los que un robot debe sortear muchos obstáculos [20].

Figura 3.6: Robot móvil con orugas.

Su principal ventaja es su robustez frente a variaciones en el terreno, algunos diseños incluso siendo capaces de subir escaleras. Pero presentan desventajas como que requieren mucha energía para cambiar de dirección, ya que tienen alto torque (se basan en el principio de resbalamiento). Otro de sus problemas es que generalmente su odometría es más inexacta que en robot con ruedas normales.

Al analizar este tipo de locomoción observamos que un robot con este diseño puede realizar giros sobre si mismo, y desplazarse hacia delante y hacia atrás, combinando el desplazamiento con el giro, pero volvemos a encontrarnos con el problema de no poder realizar trayectorias dentadas sin tener que parar el desplazamiento y posterior orientación del robot. Por lo tanto volvemos a tener el problema de esa pérdida de tiempo dedicada a la orientación, ya que no podemos combinar el movimiento de traslación y rotación.

3.2.2. Conclusión

Con este apartado concluimos que aunque las ruedas especiales presentan ciertas desventajas frente a ruedas convencionales, para la finalidad de nuestro trabajo este tipo de ruedas implican ciertas funcionalidades que implicarán una mejor maniobrabilidad en un ambiente doméstico.

Vistas todas las características de varios tipo de ruedas, se decide optar por un diseño de cuatro ruedas Mecanum configuradas de la misma forma que las ruedas de un coche. Con esta configuración tendremos una omnidireccionalidad completa y se podrán hacer infinidad de trayectorias, incluyendo combinaciones simultaneas de desplazamiento y rotación de la plataforma.

3.3. Modelado del comportamiento cinemático del robot

En este apartado se van a analizar las relaciones cinemáticas de una plataforma configurada con cuatro ruedas Mecanum de la misma forma que usaremos en este trabajo. El objetivo es obtener tanto la cinemática directa como la inversa de manera que contemos con las ecuaciones necesarias para obtener las velocidades de cada una de las ruedas en función de la velocidad deseada, y a la vez sepamos traducir la velocidad del robot a partir de las de cada motor. De esta forma contaremos con las herramientas necesarias para realizar una lectura de la velocidad y desplazamiento real del robot, con el fin de minimizar errores en la trayectoria.

3.3.1. Cinemática

En la siguiente figura podemos observar la configuración de ruedas que utilizaremos para la plataforma, así como todos los parámetros que se van a usar en el cálculo de la cinemática [7] [8]:

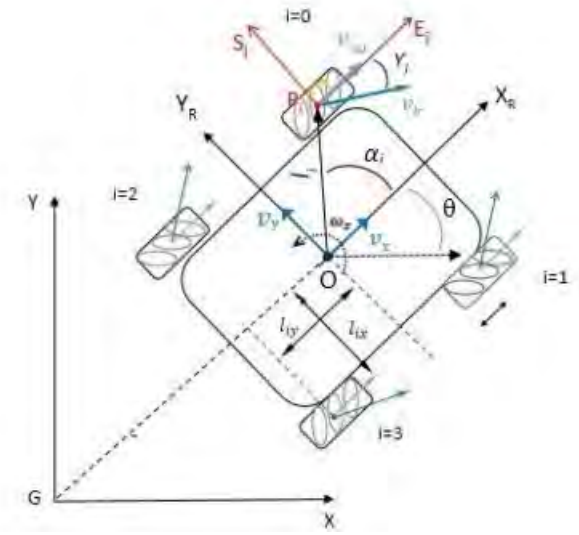


Figura 3.7: Configuración de ruedas y parámetros necesarios.

Paso a definir cada una de las variables usadas:

- x, y, θ, posición del robot (x, y) y su ángulo de orientación (θ).

- X, G, Y, sistema de referencia.
- X_R, Y_R, O, sistema de referencia del robot; sistema de coordenadas cartesianas asociado con el movimiento del centro del cuerpo.
- S_i,P_i,E_i, sistema coordenado de la rueda *i* en el punto central de la rueda.
- O, P_i, base inercial del robot en el sistema de referencia y $P_i = [X_{P_i}, Y_{P_i}]$ el centro de los ejes de rotación de la rueda i.
- $\vec{OP_i}$, vector que indica la distancia entre el centro del robot y el centro de la rueda *i*
- l_{ix} es la mitad de la distancia entre los centros de rueda paralelos y l_{iy} es la mitad de la distancia entre los centros de las ruedas delanteras y traseras.
- l_i, distancia entre las ruedas y el centro del robot.
- r_i radio de la rueda i
- r_r, radio de los rodillos.
- α_i, angulo entre $\vec{OP_i}$ y X_R.
- β_i, ángulo entre S_i y X_R.
- γ_i, ángulo entre v_{ir} y E_i.
- $\omega_i[rad/s]$, velocidad angular de la rueda.
- $v_{iw}[m/s], (i = 0, 1, 2, 3) \in R$, vector de velocidad correspondiente a las revoluciones de la rueda.
- v_{ir}, velocidad de los rodillos.
- $[w_{si}, w_{Ei}, w_i]^T$, velocidad generalizada del punto P_i en el sistema S_i, P_i, E_i.
- $[v_{si}, v_{Ei}, w_i]^T$, velocidad generalizada del punto P_i en el sistema $X_R O Y_R$.
- $v_x, v_y[m/s]$ velocidad lineal del robot.
- $\omega_z[rad/s]$, velocidad angular del robot.

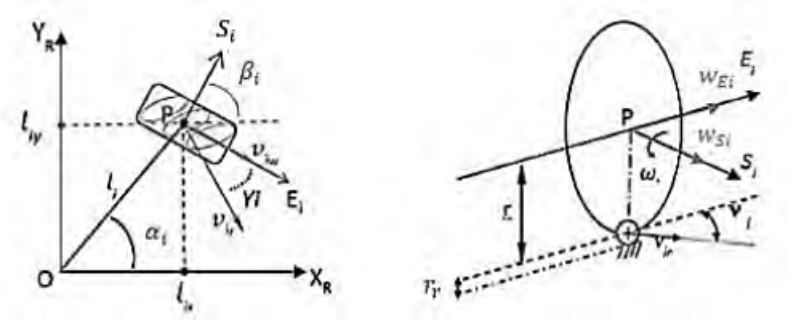


Figura 3.8: Parámetros de la rueda.

Podemos calcular la velocidad de la rueda *i* y la velocidad tangencial del rodillo que toca el suelo.

$$v_{ir} = \frac{1}{cos45} r_i \omega_i, \quad w_{Ei} = r_i \omega_i, \quad i = 0, 1, 2, 3 \tag{3.1}$$

De acuerdo a la figura 3.8 y considerando la ecuación 3.1, la velocidad de la rueda *i* en el sistema $S_i P_i E_i$ puede ser derivada por:

$$v_{S_i} = v_{ir} sen\gamma_i$$
$$v_{E_i} = w_i ri + v_{ir} cos\gamma_i$$

$$\begin{pmatrix} v_{S_i} \\ v_{E_i} \end{pmatrix} = \begin{pmatrix} 0 & sen\gamma_i \\ r_i & cos\gamma_i \end{pmatrix} \begin{pmatrix} \omega_i \\ v_{ir} \end{pmatrix} = w_i T_{P_i} \begin{pmatrix} \omega_i \\ v_{ir} \end{pmatrix} \tag{3.2}$$

La matriz de transformación de las velocidades de la *i*-ésima rueda a su centro:

$$w_i T_{P_i} = \begin{pmatrix} 0 & sen\gamma_i \\ r_i & cos\gamma_i \end{pmatrix} \tag{3.3}$$

De acuerdo a la figura 3.8 y la figura 3.7 la velocidad del centro de la rueda trasladada a sistema $X_R O Y_R$ puede ser calculada por la ecuación 3.7:

$$\begin{pmatrix} v_{iX_R} \\ v_{iY_R} \end{pmatrix} = \begin{pmatrix} cos\beta_i & -sen\beta_i \\ sen\beta_i & cos\beta_i \end{pmatrix} \begin{pmatrix} v_{S_i} \\ v_{E_i} \end{pmatrix} = w_i T_{P_i} P_i T_R \begin{pmatrix} \omega_i \\ v_{ir} \end{pmatrix} \tag{3.4}$$

Después, la matriz de transformación del centro de la *i*-ésima rueda a el sistema de coordenadas del robot puede ser calculado de la ecuación 3.5

$$P_i T_R = \begin{pmatrix} cos\beta_i & -sen\beta_i \\ sen\beta_i & cos\beta i \end{pmatrix} \tag{3.5}$$

Teniendo en cuenta que el movimiento del robot es plano también tenemos:

$$\begin{pmatrix} v_{iX_R} \\ v_{iY_R} \end{pmatrix} = \begin{pmatrix} 1 & 0 & -l_{iy} \\ 0 & 1 & l_{ix} \end{pmatrix} \begin{pmatrix} v_X \\ v_Y \\ \omega \end{pmatrix} = T' \begin{pmatrix} v_{X_R} \\ v_{Y_R} \\ \omega_R \end{pmatrix} \tag{3.6}$$

Donde

$$T' = \begin{pmatrix} 1 & 0 & -l_{iy} \\ 0 & 1 & l_{ix} \end{pmatrix} \tag{3.7}$$

De la ecuación 3.3 y 3.5, el modelo cinemátivo inverso puede ser calculado:

$$w_i T_{P_i} P_i T_R \begin{pmatrix} \omega_i \\ v_{ir} \end{pmatrix} = T' \begin{pmatrix} v_{X_R} \\ v_{Y_R} \\ \omega_R \end{pmatrix}, \quad i = 0, 1, 2, 3 \tag{3.8}$$

Como

$$r_i \neq 0, 0 < |\gamma_i| < \frac{\pi}{2}, det(P_i T_R) \neq 0, det(w_i T_{P_i}) \neq 0$$

Por lo tanto al combinar las ecuaciones 3.4 y 3.6 la velocidad de la base relacionada a la velocidad de rotación de la *i*-ésima rueda puede ser calculada con la ecuación 3.9.

$$\begin{pmatrix} \omega_i \\ v_{ir} \end{pmatrix} = w_i T_{P_i}^{-1} \cdot P_i T_R^{-1} \cdot T' \begin{pmatrix} v_{X_R} \\ v_{Y_R} \\ \omega_R \end{pmatrix}, \quad i = 0, 1, 2, 3 \tag{3.9}$$

De acuerdo a la ecuación 3.3 y 3.4 hay una relación entre variables en cada uno de los sistemas de las ruedas del robot y su centro. Y con la cinemática inversa, la velocidad del sistema puede ser calculada implementando v_{ir} la velocidad linear y ω_i la velocidad angular de la rueda *i*-ésima en la ecuación 3.10 y la contraria en la ecuación 3.11.

$$\begin{pmatrix} v_{X_R} \\ v_{Y_R} \\ \omega_z \end{pmatrix} = T^{+} \begin{pmatrix} \omega_i \\ v_{ir} \end{pmatrix} \tag{3.10}$$

$$\begin{pmatrix} \omega_i \\ v_{ir} \end{pmatrix} = T \begin{pmatrix} v_{X_R} \\ v_{Y_R} \\ \omega_R \end{pmatrix} \tag{3.11}$$

Donde:

$$T = w_i T_{P_i}^{-1} \cdot P_i T_R^{-1} \cdot T', \quad T^+ = (T^T T)^{-1} T^T.$$

$$T = \begin{pmatrix} cos\beta_i & -sen\beta_i \\ sen\beta_i & cos\beta_i \end{pmatrix}^{-1} \cdot \begin{pmatrix} 0 & sen\gamma_i \\ r_i & cos\gamma_i \end{pmatrix}^{-1} \cdot \begin{pmatrix} 1 & 0 & -l_{iy} \\ 0 & 1 & l_{ix} \end{pmatrix}$$

considerando el hecho de que $l_{ix} = l_i cosa_i$ y $l_{iy} = l_i sena_i$, y asumiendo que las ruedas tienen el mismo tamaño, la matriz de transformación es:

$$T = \frac{1}{-r} \begin{pmatrix} \frac{cos(\beta_i - \gamma_i)}{sen(\gamma_i)} & \frac{sen(\beta_i - \gamma_i)}{sen(\gamma_i)} & \frac{l_i sen(-\alpha_i + \beta_i - \gamma i)}{sen(\gamma_i)} \\ -\frac{rcos(\beta_i)}{sen(\gamma_i)} & -\frac{rsen(\beta_i)}{sen(\gamma_i)} & -\frac{l_i sen(-\alpha_i + \beta_i)r}{sen(\gamma_i)} \end{pmatrix} \tag{3.12}$$

$$T^+ = \frac{1}{l_i^2 + 1} \begin{pmatrix} -\frac{1}{2}(l_i^2 sen(\beta_i) - l_i^2 sen(-\beta_i + 2\alpha_i) + 2sen(\beta_i))r & \frac{1}{2} l_i^2 sen(\gamma_i - \beta_i + 2\alpha i) - \frac{1}{2} sen(-\gamma_i + \beta_i) l_i^2 - sen(-\gamma_i + \beta_i) \\ \frac{1}{2} r(l_i^2 cos(\beta_i) - l_i^2 cos(-\beta_i + 2\alpha_i) + cos(\beta_i)) & -\frac{1}{2} l_i^2 cos(\gamma_i - \beta_i + 2\alpha i) + \frac{1}{2} cos(-\gamma_i + \beta_i) l_i^2 + cos(-\gamma_i + \beta_i) \\ cos(\alpha_i - \beta i) l_i r & cos(\alpha_i - \beta_i + \gamma_i) l_i \end{pmatrix} \tag{3.13}$$

Obtenemos la cinemática inversa con la ecuación 3.14

$$\begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix} = \frac{-1}{r} \begin{pmatrix} \frac{cos(\beta 1 - \gamma_1)}{sen\gamma 1} & \frac{sen(\beta_1 - \gamma_1)}{sen\gamma_1} & \frac{l_1 sen(\beta_1 - \gamma_1 - \alpha_1)}{sen\gamma 1} \\ \frac{cos(\beta 2 - \gamma_2)}{sen\gamma 2} & \frac{sen(\beta_2 - \gamma_2)}{sen\gamma_2} & \frac{l_2 sen(\beta_2 - \gamma_2 - \alpha_2)}{sen\gamma 2} \\ \frac{cos(\beta 3 - \gamma_3)}{sen\gamma 3} & \frac{sen(\beta_3 - \gamma_3)}{sen\gamma_3} & \frac{l_3 sen(\beta_3 - \gamma_3 - \alpha_3)}{sen\gamma 3} \\ \frac{cos(\beta 4 - \gamma_4)}{sen\gamma 4} & \frac{sen(\beta_4 - \gamma_4)}{sen\gamma_4} & \frac{l_4 sen(\beta_4 - \gamma_4 - \alpha_4)}{sen\gamma 4} \end{pmatrix} \tag{3.14}$$

La ecuación 3.15 muestra la matriz Jacobiano para la cinemática inversa del sistema:

$$T = \frac{-1}{r} \begin{pmatrix} \frac{cos(\beta 1 - \gamma_1)}{sen\gamma 1} & \frac{sen(\beta_1 - \gamma_1)}{sen\gamma_1} & \frac{l_1 sen(\beta_1 - \gamma_1 - \alpha_1)}{sen\gamma 1} \\ \frac{cos(\beta 2 - \gamma_2)}{sen\gamma 2} & \frac{sen(\beta_2 - \gamma_2)}{sen\gamma_2} & \frac{l_2 sen(\beta_2 - \gamma_2 - \alpha_2)}{sen\gamma 2} \\ \frac{cos(\beta 3 - \gamma_3)}{sen\gamma 3} & \frac{sen(\beta_3 - \gamma_3)}{sen\gamma_3} & \frac{l_3 sen(\beta_3 - \gamma_3 - \alpha_3)}{sen\gamma 3} \\ \frac{cos(\beta 4 - \gamma_4)}{sen\gamma 4} & \frac{sen(\beta_4 - \gamma_4)}{sen\gamma_4} & \frac{l_4 sen(\beta_4 - \gamma_4 - \alpha_4)}{sen\gamma 4} \end{pmatrix} \tag{3.15}$$

Y para la cinemática directa de acuerdo a la ecuación 3.10, tenemos:

$$\begin{pmatrix} v_x \\ v_y \\ \omega_z \end{pmatrix} = T^+ \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix} \tag{3.16}$$

3.3.2. Solución del robot omnidireccional con cuatro Mecanum

En la siguiente tabla se resumen los parámetros que tenemos con la configuración de la figura 3.7:

i	Rueda	α_i	β_i	γ_i	l_i	l_{ix}	l_{iy}
0	1	$\pi/4$	$\pi/2$	$-\pi/4$	l	l_x	l_y
1	2	$-\pi/4$	$-\pi/2$	$\pi/4$	l	l_x	l_y
2	3	$3\pi/4$	$\pi/2$	$\pi/4$	l	l_x	l_y
3	4	$-3\pi/4$	$-\pi/2$	$-\pi/4$	l	l_x	l_y

Tabla 3.1: Parámetros del conjunto de ruedas Mecanum.

Sustituyendo los valores de la 3.1 en la ecuación 3.15 y teniendo en cuenta la ecuación 3.14:

$$T = \frac{1}{r}\begin{pmatrix} 1 & -1 & -(l_x+l_y) \\ 1 & 1 & (l_x+l_y) \\ 1 & 1 & -(l_x+l_y) \\ 1 & -1 & (l_x+l_y) \end{pmatrix} \tag{3.17}$$

$$T^{+} = \frac{r}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -\frac{1}{(l_x+l_y)} & \frac{1}{(l_x+l_y)} & -\frac{1}{(l_x+l_y)} & \frac{1}{(l_x+l_y)} \end{pmatrix} \tag{3.18}$$

De acuerdo a las ecuaciones 3.10 y 3.11:

$$\begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix} = \frac{1}{r}\begin{pmatrix} 1 & -1 & -(l_x+l_y) \\ 1 & 1 & (l_x+l_y) \\ 1 & 1 & -(l_x+l_y) \\ 1 & -1 & (l_x+l_y) \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ \omega_z \end{pmatrix} \tag{3.19}$$

$$\begin{aligned} \omega_1 &= \tfrac{1}{r}(v_x - v_y - (l_x+l_y)\omega), \\ \omega_2 &= \tfrac{1}{r}(v_x + v_y + (l_x+l_y)\omega), \\ \omega_3 &= \tfrac{1}{r}(v_x - v_y - (l_x+l_y)\omega), \\ \omega_4 &= \tfrac{1}{r}(v_x + v_y + (l_x+l_y)\omega). \end{aligned}$$

(3.20)

y por último:

$$\begin{pmatrix} v_x \\ v_y \\ \omega_z \end{pmatrix} = \frac{r}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -\frac{1}{(l_x+l_y)} & \frac{1}{(l_x+l_y)} & -\frac{1}{(l_x+l_y)} & \frac{1}{(l_x+l_y)} \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix} \tag{3.21}$$

Ya tenemos las ecuaciones de la velocidad longitudinal:

$$v_x(t) = (\omega_1 + \omega_2 + \omega_3 + \omega_4) \cdot \frac{r}{4} \tag{3.22}$$

Velocidad transversal:

$$v_y(t) = (-\omega_1 + \omega_2 + \omega_3 - \omega_4) \cdot \frac{r}{4} \tag{3.23}$$

Y velocidad de rotación:

$$\omega_z(t) = (-\omega_1 + \omega_2 - \omega_3 + \omega_4) \cdot \frac{r}{4(l_x + l_y)} \tag{3.24}$$

El software que se ha diseñado para el control con el microprocesador incluirá estas ecuaciones de manera que en todo momento esté convirtiendo las velocidades longitudinal, transversal y de rotación en las velocidades angulares de los cuatro motores. Al mismo tiempo usará las ecuaciones de la cinemática inversa para calcular la posición real en todo momento.

Capítulo 4

Implementación

Contenido

4.1. Introducción

Una vez realizado el estudio de la cinemática del robot con las cuatro ruedas Mecanum pasamos a diseñar y fabricar la plataforma móvil sobre la que irá colocado el robot doméstico. En este apartado se va a comentar el diseño, sus dimensiones y materiales. Posteriormente se analizará el hardware usado y la fabricación de ciertas partes de la electrónica. Por último se comentará el software creado para el control.

4.2. Diseño y montaje de la plataforma

Para el diseño de la plataforma se ha utilizado el software de modelado SolidWorks. A continuación se puede ver una imagen del modelo diseñado:

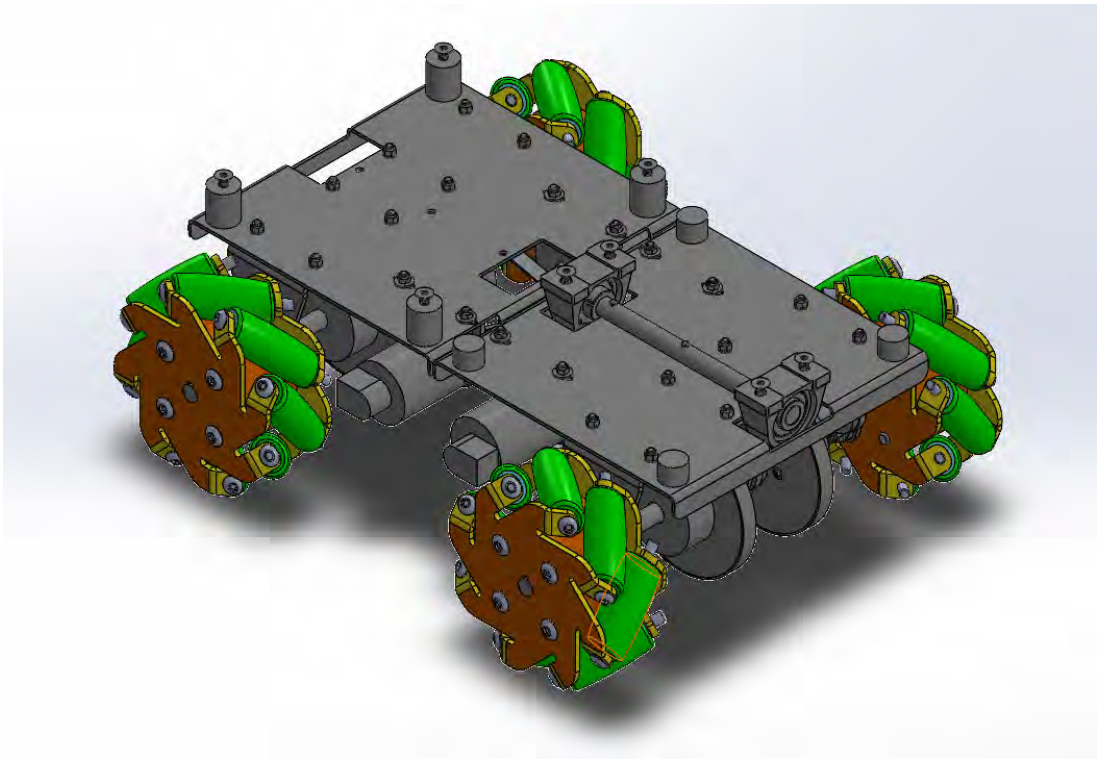
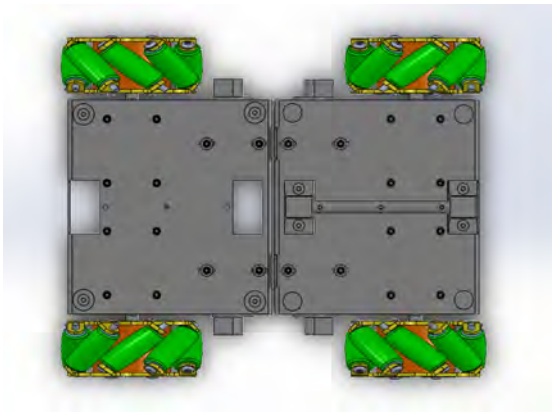


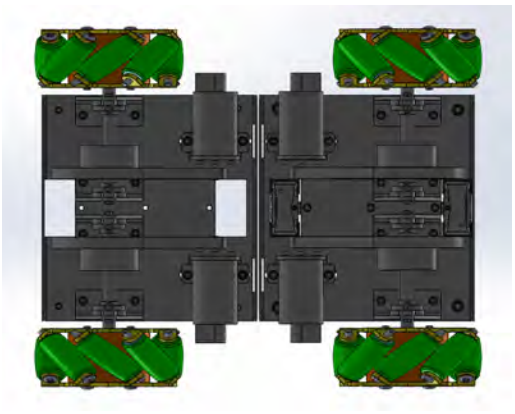
Figura 4.1: Modelo de la plataforma en SolidWorks.

Se ha decidido utilizar una transmisión mediante correas usando ruedas dentadas y una correa para cada motor, con un factor de reducción de 8.5. De esta forma conseguiremos tener más par en cada rueda y así el control realizado sea mucho más suave. Al tener este factor de reducción lo que estamos consiguiendo también es mucha más resolución de los encoders y de esta forma poder medir mejor la velocidad de las ruedas y como hemos mencionado esto ayudara a un control mucho más suave que con una transmisión directa.

Durante el diseño de la plataforma surge uno de los problemas que conlleva este tipo de configuración con 4 ruedas Mecanum, y es que en el momento que una de las ruedas deje de estar en contacto con el suelo, el sistema de control realizará trayectorias erróneas debido a que una de las lecturas de encoder estará dando error. Es por ello que dos de las ruedas han sido diseñadas de forma que quedarán situadas en una superficie basculante. Así, el robot podrá sobrepasar ciertos obstáculos sin llevar a este error.

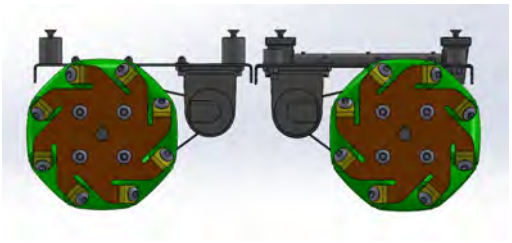


(a) Vista planta.

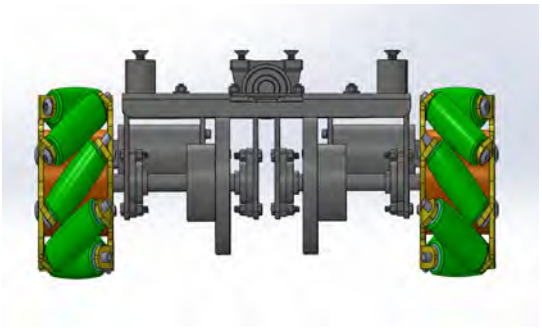


(b) Vista planta inferior.

Figura 4.2: Planta de la plataforma.



(a) Perfil.



(b) Alzado.

Figura 4.3: Perfil y alzado de la plataforma.

A continuación se muestran unas imágenes de la plataforma despues de su montaje:

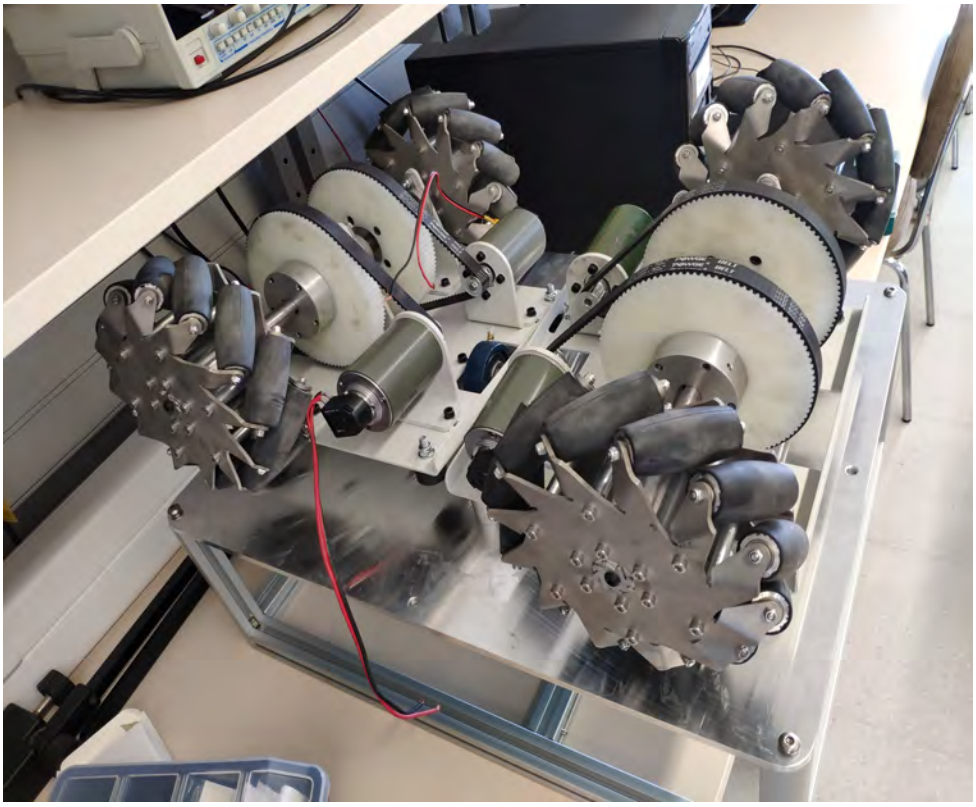


Figura 4.4: Plataforma del robot.

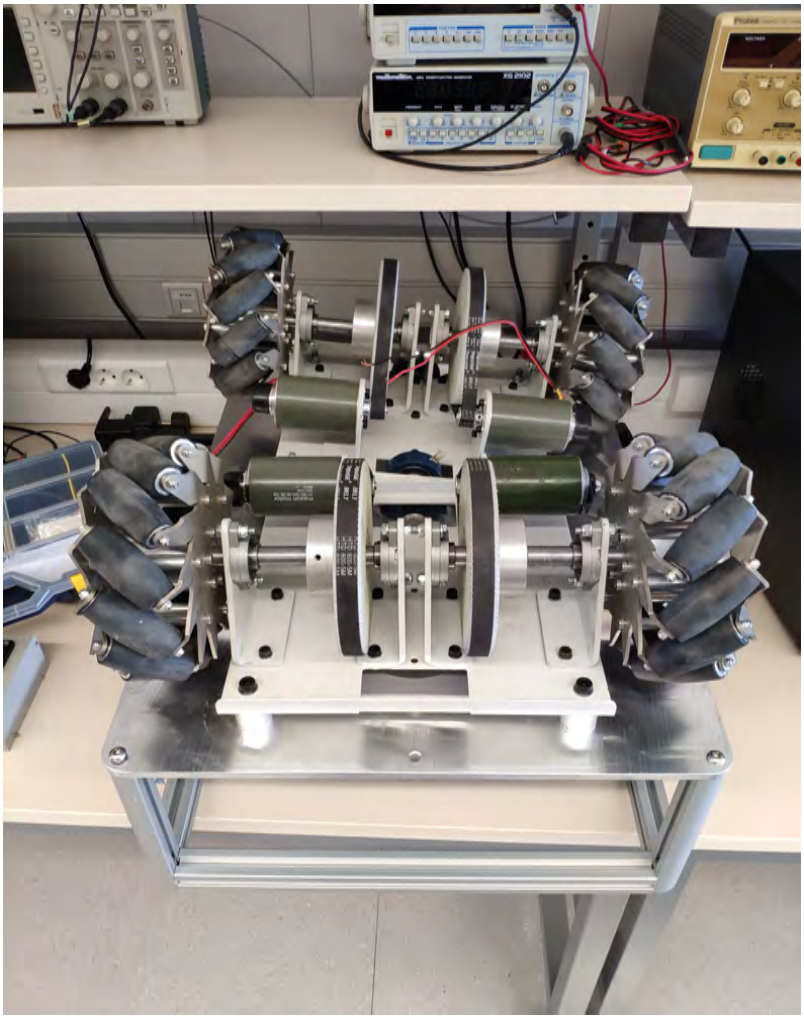


Figura 4.5: Plataforma del robot (otra vista).

Las dimensiones que a nuestro estudio nos interesan son:

- Distancia entre el centro de dos de las ruedas paralelas: 40 cm.
- Distancia entre el centro de las ruedas delanteras y traseras: 40 cm.
- Diámetro de las ruedas: 8 pulgadas.

En la figura 4.5 se puede ver la superficie basculante de las ruedas delanteras de la que se hablo anteriormente. Dicha plataforma evitará que una de las ruedas quede en suspensión al sortear obstáculos.

4.3. Electrónica

La parte electrónica de la plataforma móvil es tan importante como la mecánica. En este apartado se comenta todo cada uno de los componentes electrónicos que se han elegido, así como el software diseñado.

4.3.1. Hardware

En función de las funcionalidades que debe cumplir la plataforma móvil, se han elegido una serie de componentes que se ajustan a las necesidades, tanto para el control como para la medición de valores reales. Además para realizar todo el conexionado de la electrónica y de la potencia de los motores se han fabricado todos los cables necesarios.

Motores eléctricos

Para el accionamiento de cada una de las ruedas se utilizan cuatro motores eléctricos. Se han usado cuatro motores de 24 V y 60 W como los de las imágenes siguientes:

Figura 4.6: Motor utilizado en la plataforma.

Encoders

Para la monitorización constante de la velocidad, además del control de la misma como veremos más adelante, se han usado 4 encoders HEDS-5540 del

fabricante Avago Technologies. Se decidió utilizar este tipo de encoder ya que cuenta con una resolución suficiente para la labor que debe desempeñar nuestro robot. Se trata de un codificador óptico incremental en cuadratura que cuenta con 500 PPR (pulsaciones por vuelta) y 30000 rpm de velocidad máxima, lo cual para nuestra aplicación nos basta. Es de fácil montaje realizado por tornillo (como se ve en la figura 4.8c)

(a) Imagen delantera.

(b) Imagen trasera.

Figura 4.7: Encoder HEDS-5540 A11.

Para el montaje de los encoders ha sido necesario el diseño de una pieza para el acople entre el encoder y el motor. A continuación se muestran unas imágenes, tanto de la pieza como del montaje mediante tornillo del encoder.

(a) Pieza de acople

(b) Acople

(c) Montaje por tornillo

Figura 4.8: Acople y montaje encoder

Para el conexionado de los encoders a las controladoras se han fabricado cables con las dimensiones adecuadas, crimpando sus entremos y añadiendo los conectores adecuados. Sellándolos por último mediante tubo termoretráctil con la ayuda de un decapador. En la siguiente imagen se muestra el resultado:

Figura 4.9: Cableado de los encoders.

En nuestro caso los pulsos del encoder serán leídos mediante las controladoras, las cuales disponen del software preparado para dicha tarea.

Microcontrolador

El microcontrolador que se ha elegido para este trabajo es un Arduino MEGA 2560 modificado, el cual incluye el módulo wifi ESP8266 (4.5.

Figura 4.10: Arduino MEGA 2560 + ESP8266.

Este módulo es un chip integrado con conexión WiFi y compatible con el protocolo TCP/IP [21]. El objetivo principal es dar acceso al microcontrolador a una red. Se ha optado por usar este microcontrolador debido a ciertas ventajas:

- Es un microcontrolador muy potente de 8 bits
- Gran cantidad de pines en comparación a otros microcontroladores, lo cuál de cara a futuras modificaciones pueden ser necesarios.
- Cuenta con cuatro unidades UART TTL 0V / 5V, de las cuales necesitamos 3, ya que cada controladora, como veremos más adelante, usará una de ellas, y seguiremos teniendo disponibles para la comunicación con un ordenador para enviar lecturas de posición reales.
- Varias posibilidades de alimentación.
- Bajo coste.
- Cuenta con el módulo wifi, que aunque en el desarrollo de este trabajo no se usará, si que es una buena opción a tener en futuras modificaciones del control del robot.

Mediante el microcontrolador nos encargaremos de leer los valores de velocidad que se desean, ya sean por lectura de un vector por el puerto serie, o bien por el joystick o mando de videoconsola, y las convertiremos usando la cinemática directa en las velocidades angulares de cada motor. Mediante los puertos serie el microcontrolador se comunicará enviándole dichas velocidades. A su vez, el microcontralor será el encargado de solicitar a las controladoras las lecturas de los encoders cada cierto intervalo de tiempo, y a través de estas lecturas ir calculado el desplazamiento real que sufre el robot. Así mismo podrá ir solicitando parámetros importantes como temperatura de las controladoras, voltaje, intensidad, etc.

Controladoras

Para el control de potencia de los motores se decidió usar dos controladoras de la marca Roboclaw, como las de la figura 4.11:

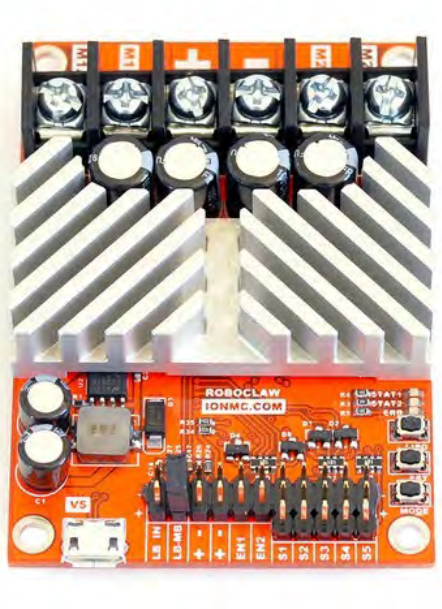


Figura 4.11: Controladora de motor Roboclaw.

Este modelo de controladoras pueden controlar hasta dos motores con una corriente de 15 amperios continuos por cada canal. Soportando picos de hasta 30 amperios. Una de las ventajas, y a su vez motivos por los que se ha optado por usar este modelo de controladora son las varias formas que tiene para controlarla. Se pueden manejar mediante USB, radio RC, PWM, serie TTL, analógicos y

microcontroladores, como es nuestro caso.

Además de todo esto, la controladora incluye su propio software, de manera que permite realizar un control de los motores tanto de posición como de velocidad, así como realizar la constante lectura de velocidad, posición y aceleración de los encoders; temperatura, voltaje y amperaje de cada canal, etc. En el apartado de experimentos se comentarán ciertas opciones a las que da opción la controladora, configurables desde un ordenador con su propio software o desde cualquiera de los canales de comunicación.

Cabe destacar también que la controladora RoboClaw incorpora varias características de protección, incluyendo temperatura, corriente, sobretensión y voltaje.

La controladora alimenta a los encoders, y se conecta a sus canales para hacer las mediciones necesarias. Para la conexión de la controladora al microcontrolador también se ha fabricado el cable necesario y en este caso se ha envuelto en una malla trenzada para tener mayor comodidad a la hora de hacer cambios en el cable (figura 4.12)

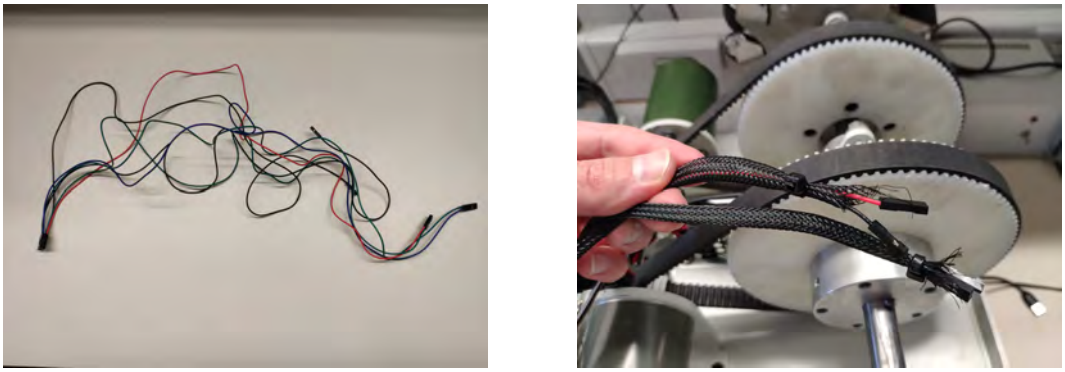


Figura 4.12: Cables fabricados para la controladora.

Elementos de control

- Mando PS2

  Se decidió usar un mando inalámbrico por vía bluetooth, de manera que el único componente que debe permanecer próximo al robot es el receptor del mando, el cual es alimentado a 3.3V por el mismo microcontrolador. Mediante el uso de ciertas librerías se realiza la lectura de todos los botones del mando, lo cual nos deja la ventaja de poder configurar mucho más la maniobrabilidad, así como diseñar ciertas trayectorias (como las reali-

zadas más adelante para los experimentos) al disponer de muchas más combinaciones que con el joystick simple [22].

Figura 4.13: Programación de un mando de videoconsola.

4.3.2. Software

En este apartado se va a comentar el código utilizado para el control del robot, diseñado en el entorno de Arduino IDE. El microcontrolador será el encargado de realizar todas las comunicaciones entre el sistema de control (puertos serie, joystick o mando), la controladora y los encoders. A su vez el código se puede descomponer en varias partes.

Lectura del sistema de control

En cada ciclo del microcontrolador, este leerá, en función del sistema de control, una serie de valores, y los convertirá en el vector de velocidades $[v_x, v_y, \omega_z]$.

Una vez obtenido dicho vector, usará las ecuaciones de la cinemática directa del robot vistas en el apartado 3. De manera que se obtienen cuatro velocidades

angulares, correspondientes a nuestras cuatro ruedas. En nuestro caso, los motores no están directamente conectados a la rueda, es decir, la transmisión no es directa, si no que el movimiento es transmitido a través de dos ruedas dentadas y una correa, teniendo una relación de reducción de 8.5. Por lo que habrá que incluirlo en las ecuaciones.

Por último, cabe destacar que la controladora lee los valores de velocidad en pulsaciones por segundo del encoder. En nuestro caso como el encoder tiene 2000 cambios de flanco por vuelta, se realizan los cálculos necesarios para obtener las cuatro velocidades angulares en pulsos por segundo.

Este proceso se realiza constantemente, a fin de que los cambios de velocidad solicitados se realicen lo más rápido posible.

Paso a comentar las funciones implementadas en el mando de PS2 ya que ha sido el sistema más utilizado y que más combinaciones ofrece. En la figura **??** se específica la función asignada a cada botón:

Figura 4.14: Funciones del mando PS2.

- Los botones de la izquierda se han asignado para movimiento a una velocidad fija. Cabe destacar que se pueden conseguir hasta 8 direcciones con ellos.
- El joystick izquierdo se ha asignado una velocidad longitudinal y transversal variable en función del valor del potenciómetro. Lo mismo para el joystick derecho pero asignado para el giro.

- Los botones L1, R1 y R2 se han configurado con dos finalidades, primero hacer de botones de seguridad, de modo que si ninguno de los 3 está pulsado, el robot no recibirá velocidad. Independientemente de esto, a cada uno de ellos se le ha dado una constante, de manera que si pulsamos L1 obtendremos una cierta velocidad (tanto con los botones fijos como con los joysticks), si pulsamos R1 dicha constante será mayor, así que el rango de velocidades también lo será, etc.
- El botón cuadrado y equis, generan rotación del robot a una velocidad fija.
- Por último, a L2 se le ha asignado la función de trayectoria. Si este botón es pulsado, y posteriormente se pulsa cuadrado, triángulo o círculo, el robot generará una trayectoria igual a la del símbolo de su botón. Estas trayectorias nos servirán en los experimentos para detectar errores.

Envío de información a las controladoras

Al igual que la lectura del sistema de control, cada ciclo del microcontrolador se envían los valores de velocidad calculados en pulsaciones por segundo a las controladoras. Esta comunicación se hace a través de los puertos serie del Arduino.

Otra de las ventajas de la controladora Roboclaw es que dispone de su propia librería en Arduino, lo cual simplifica mucho su utilización. De esta forma, y con un código muy simple, el microcontrolador enviará a cada controladora (cada una con una dirección) la velocidad a la que debe controlar cada uno de sus motores.

Solicitud de información a las controladoras

Para minimizar el error en los desplazamientos, se deben realizar lecturas cada cierto tiempo, a fin de conocer la posición y orientación del robot en todo momento.

Para esto se ha diseñado una temporización, de manera que cada 10 ms se realice una lectura de los valores de los encoders. Esto, al igual que el envío de la velocidad, se realiza de forma muy sencilla gracias a la librería de Roboclaw. Por tanto, cada 10 ms obtendremos una medida de la posición de los encoders, y el microcontrolador se encargará de comparar dichas medidas con las anteriores, calcular la velocidad media en ese intervalo de tiempo, y posteriormente, mediante las ecuaciones de la cinemática inversa, calcular el vector de velocidades $[v_x, v_y, \omega_z]$ en ese tiempo y así obtener el desplazamiento realizado. Mediante

unos contadores de posición que se irán actualizando tenemos la posición real del robot, la cual puede ser distinta a la ideal, debido al error.

Software de control

El código diseñado para realizar el control se deja a disposición del lector en el *Apéndice A* de este trabajo.

Diseño general del sistema

Para finalizar este capítulo se ha realizado un esquema básico del funcionamiento del sistema, de manera que se pueden apreciar los elementos conectados, desde los actuadores, hasta el microcontrolador del robot:

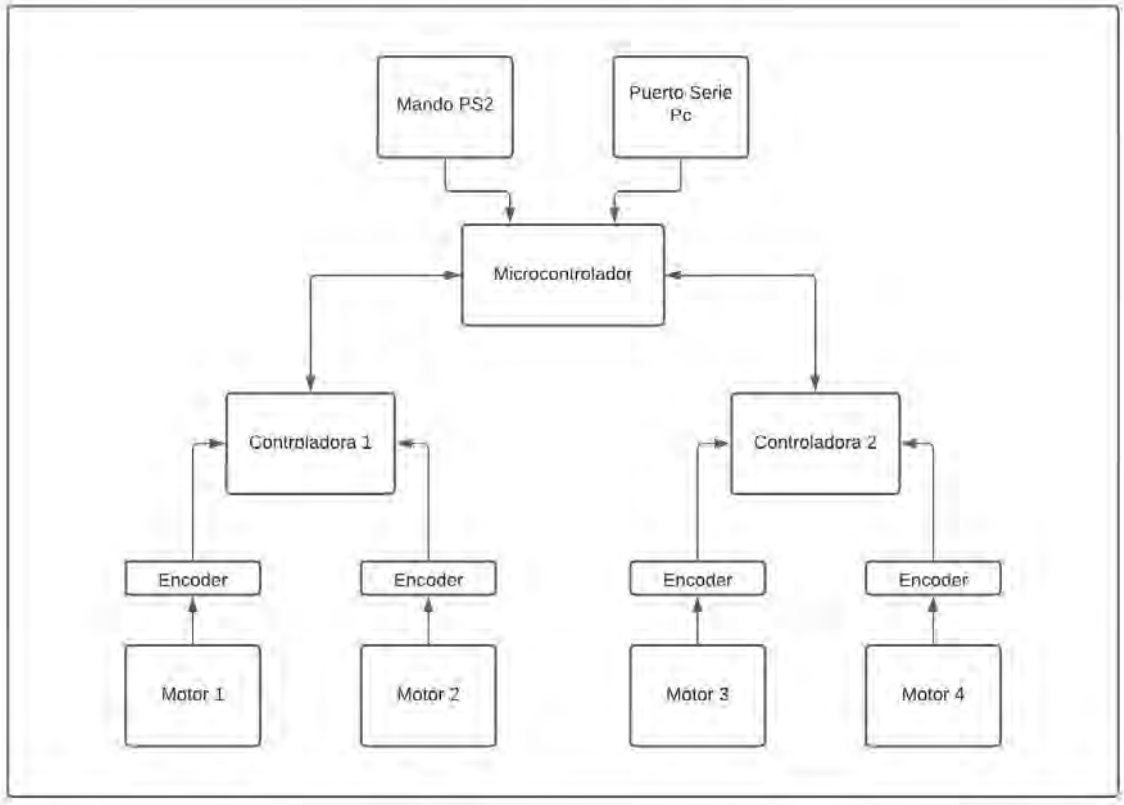


Figura 4.15: Diseño general del sistema.

Capítulo 5

Uso y configuración de las controladoras

Las controladoras usadas en este trabajo disponen de un software propio, BasicMicro Motion Studio, el cual permite modificar fácilmente los parámetros del PID desde un ordenador. Cabe destacar que dicho software permite numerosos ajustes como por ejemplo, establecer velocidades y aceleraciones máximas, valores máximos de intensidad y voltaje, establecer control de velocidad y de posición, modificar la forma en que va a ser controlada cada Roboclaw (en nuestro caso mediante comunicaciones seriales), etc.

En un primer lugar se realizó un primer control mediante la herramienta *Auto Tuning* de BasicMicro [23]. La cual proporciona un contralador bastante bueno. Sin embargo, durante la realización de pruebas se han ido modificando sus parámetros de cara a obtener respuestas más rápidas y adecuadas para la finalidad de este robot móvil. Dichas modificaciones se han realizado basándose en la teoría de la automática, variando valores de manera experimental. A continuación se observan algunas imágenes del software utilizado para los ajustes de las controladoras:

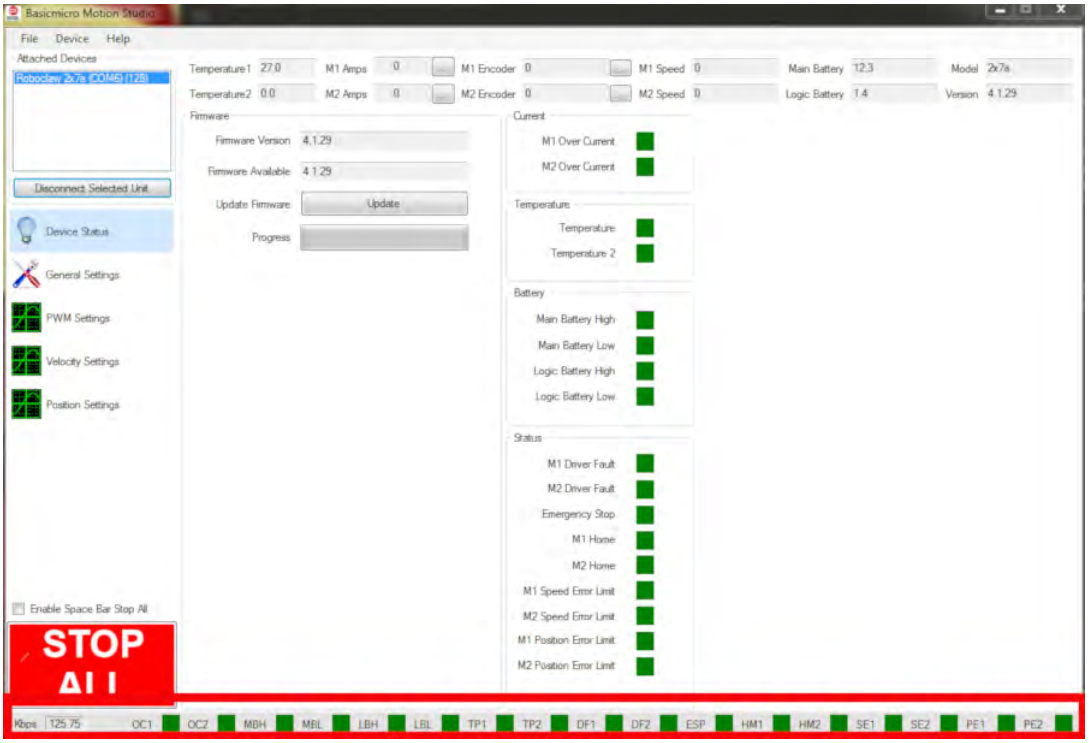


Figura 5.1: Estado del dispositivo.

En la pestaña de estado del dispositivo, podemos ver que todos los parámetros del dispositivo son correctos. En el caso de un mal funcionamiento es fácil acudir a este menú y observar si se ha producido algún error. Aquí podemos también realizar actualizaciones del firmware.

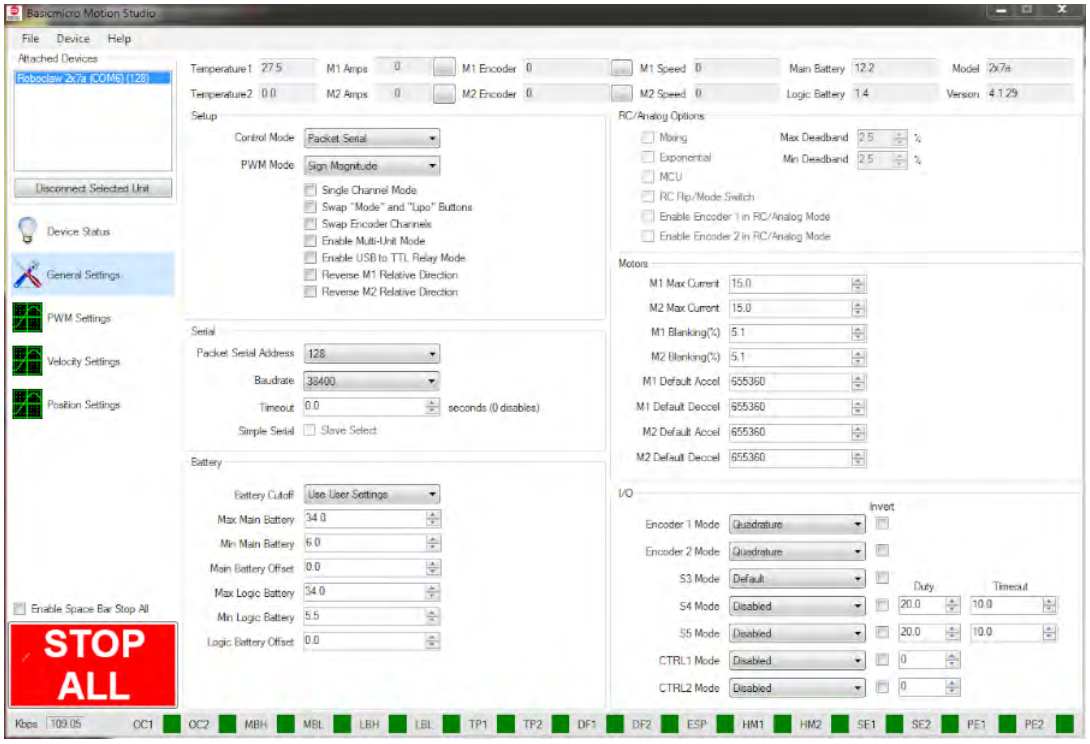


Figura 5.2: Ajustes generales.

En la pestaña de ajustes generales podemos configurar numerosos parámetros. Para empezar hay que seleccionar el modo de control que se usará, en nuestro caso comunicación serie. A cada controladora se le adjudica una dirección, a la que más tarde se enviarán los datos, y el baudaje. Además de esto podemos configurar parámetros como valores máximos de corriente, velocidad, aceleración y deceleración. Por último también podemos configurar las acciones de una serie de pines de los que dispone la controladora, así como el tipo de encoder.

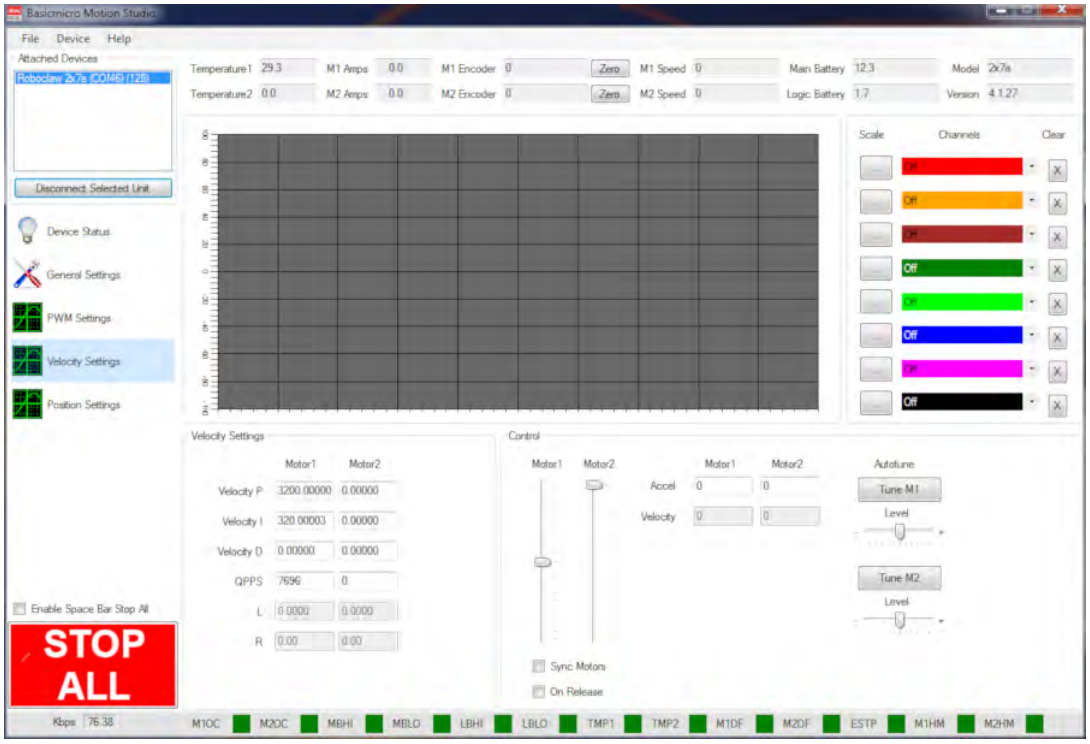


Figura 5.3: Ajustes de velocidad.

Con este menú podemos realizar un control de velocidad, tanto manual como automático como se ha comentado anteriormente. Además permite probar los motores de cada canal para poder ver la respuesta de estos ante la entrada de una cierta velocidad. Mediante esta herramienta hemos realizado el control de nuestro robot.

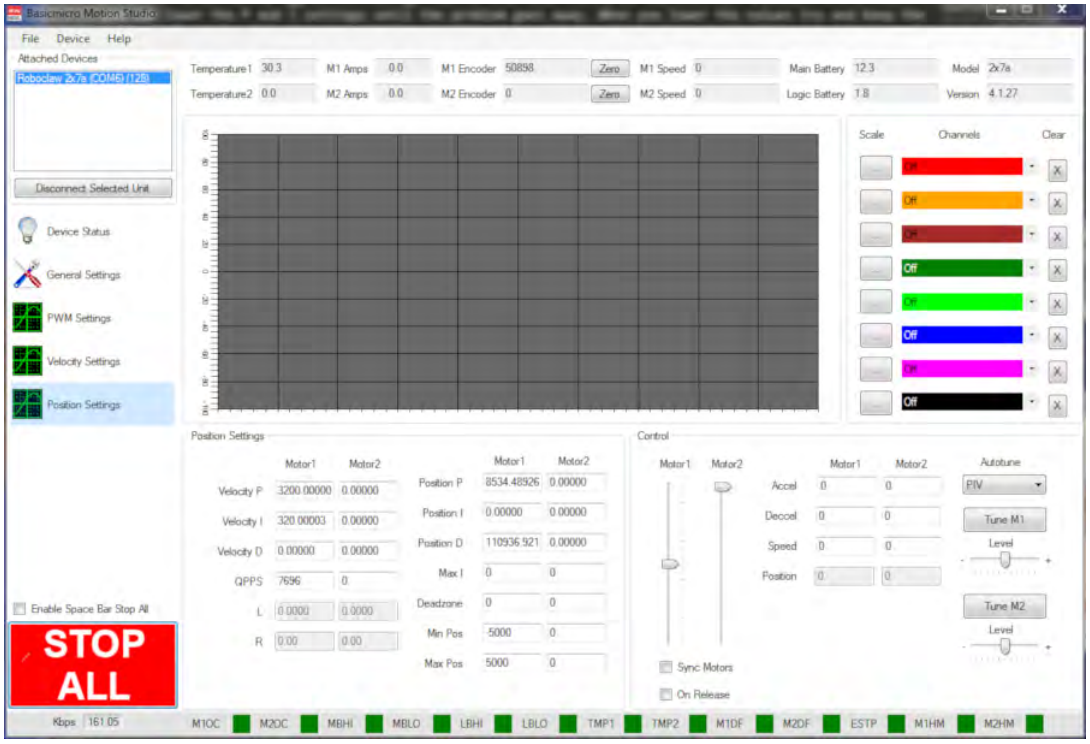


Figura 5.4: Ajustes de posición.

Por último, también se puede realizar un control de posición en lugar de velocidad. En este trabajo el objetivo se ha planteado de forma que el control sea de velocidad, es decir, no enviaremos posiciones al robot, sino que enviaremos las velocidades longitudinal, transversal y de rotación. Pero con esta herramienta es igual de fácil realizar un control de posición para futuras modificaciones de la plataforma.

Capítulo 6

Experimentos

Contenido

6.1. Introducción

Una vez construida la plataforma, y diseñado y montado toda la electrónica, incluyendo el software, procedemos a realizar ciertos experimentos con el fin de cuantificar el error que existe. Como se comentó anteriormente, se configuró el mando con ciertas combinaciones de botones que generen ciertas trayectorias. La finalidad de este apartado es mostrar los experimentos realizados y las mediciones reales obtenidas para compararlas con las ideales.

6.2. Experimentos

Para la realización de los experimentos se ha modificado el código levemente, con la finalidad de que se muestren por pantalla a través del puerto serie de Arduino los valores de lectura de cada encoder en forma de vector. De esta forma, una vez realizado, es fácil copiar estos vectores y trasladarlos a un fichero de matlab en el que representaremos la posición en cada instante de medición.

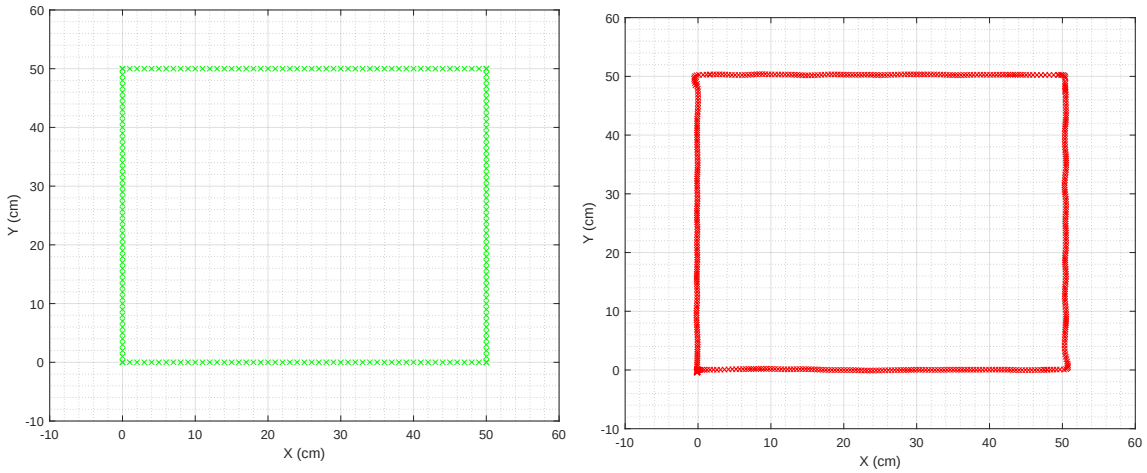
```
    pos1 = (roboclaw.ReadEncM1(address1)) ; //posición en pulsos
    pos2 = (roboclaw.ReadEncM2(address1));
    pos3 = (-1) * (roboclaw2.ReadEncM2(address2));
    pos4 = (-1) * (roboclaw2.ReadEncM1(address2));

    tiempo = millis() - tiempoActual;
    tiempoActual = millis();

    Serial.print("Medidas("); Serial.print(cont); Serial.print(",1)=");
    Serial.print(pos1); Serial.print(";");
    Serial.print("Medidas("); Serial.print(cont); Serial.print(",2)=");
    Serial.print(pos2); Serial.print(";");
    Serial.print("Medidas("); Serial.print(cont); Serial.print(",3)=");
    Serial.print(pos3); Serial.print(";");
    Serial.print("Medidas("); Serial.print(cont); Serial.print(",4)=");
    Serial.print(pos4); Serial.print(";");
    Serial.print("Medidas("); Serial.print(cont); Serial.print(",5)=");
    Serial.print(tiempo); Serial.println(";");
    cont++;

  }
}
```

Figura 6.1: Código añadido para experimentos.

Durante la realización de los experimentos se han llevado a cabo ciertas correcciones y modificaciones del control del sistema. Para ello se ha utilizado el software de las controladoras Roboclaw (BasicMicro Motion Studio), como se ha visto en el capítulo anterior.

6.2.1. Experimento 1: Seguimiento de un camino cuadrado

El primer experimento que se ha llevado a cabo ha sido el de realizar una trayectoria cuadrada de 0.5 metros de lado. A continuación se muestran dos gráficas. En primer lugar la gráfica de la trayectoria ideal, si el robot la realizase perfectamente, a velocidad constante. Y en segundo lugar la gráfica de la trayectoria real, calculada a partir de las lecturas de los encoders.



(a) Camino ideal.

(b) Camino real.

Figura 6.2: Camino cuadrado.

Para que se pueda apreciar mejor el error se representan ambas gráficas superpuestas:

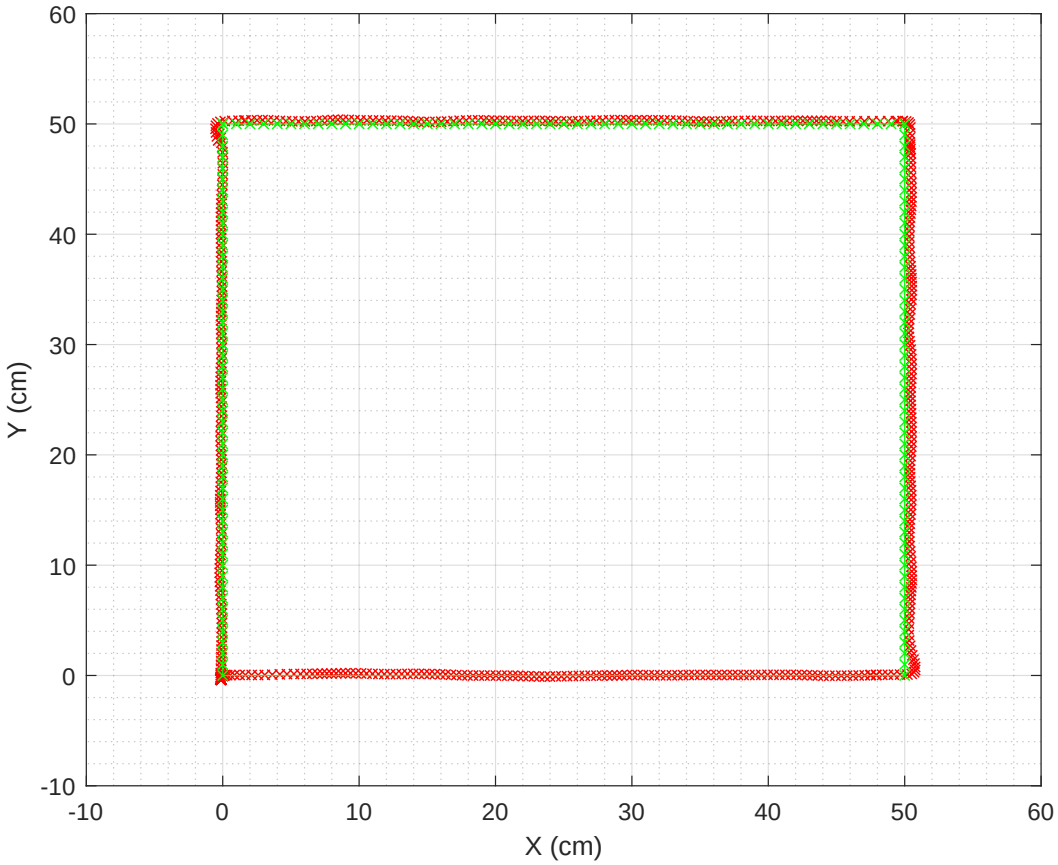


Figura 6.3: Comparación de caminos.

Como se ha realizado un control en velocidad, podemos apreciar ciertos errores a la hora de cambiar de dirección, ya que el robot invierte cierto tiempo en frenar su velocidad. Por tanto durante cierto intervalo de tiempo existe una aceleración negativa que lleva a un pequeño error. Sin embargo se puede apreciar que el resultado es bastante correcto.

Para observar los desplazamientos y la rotación realizadas con respecto al tiempo se ha hecho la siguiente representación:

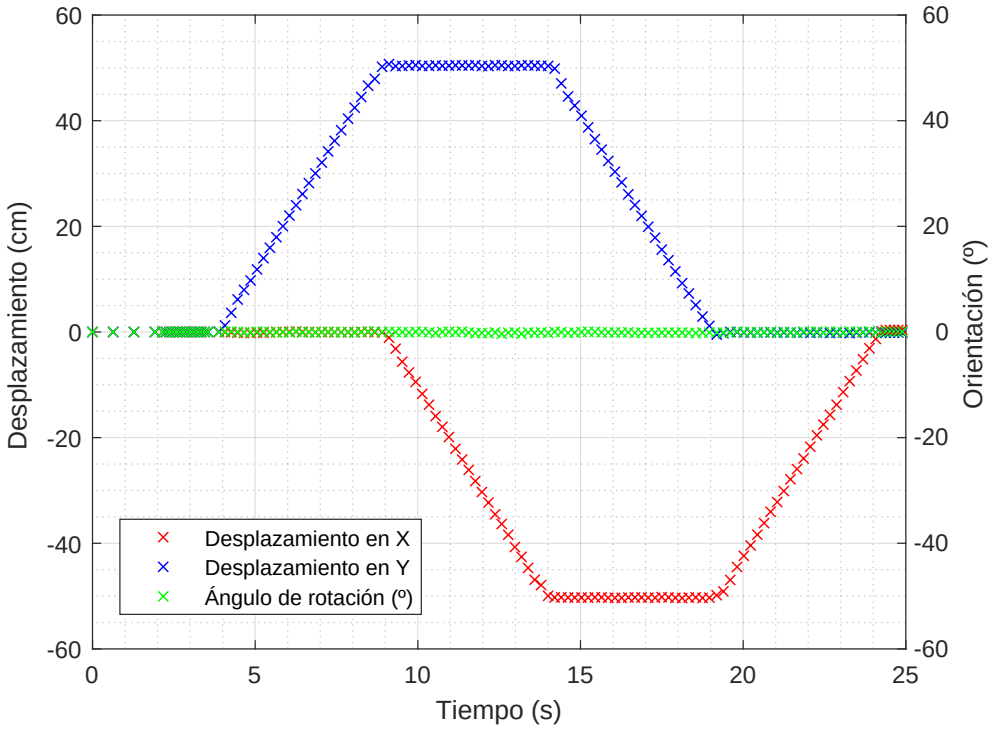


Figura 6.4: Trayectoria.

Si representamos únicamente la orientación es posible observar que el máximo error que se origina está en torno a 0,4º. Para dicha representación nos hemos ayudado de la herramienta *Curve Fitting Tool* de Matlab:

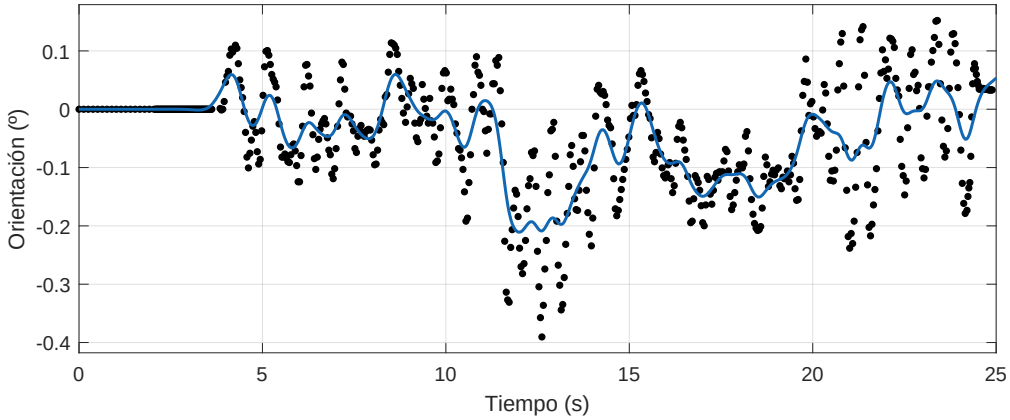


Figura 6.5: Orientación durante la realización del camino.

6.2.2. Experimento 2: Realización de un giro completo sobre su centro

El segundo experimento pretende realizar un giro de 360º sobre el centro de la plataforma. Para ver los resultados vamos a representar la rotación realizada con respecto al tiempo:

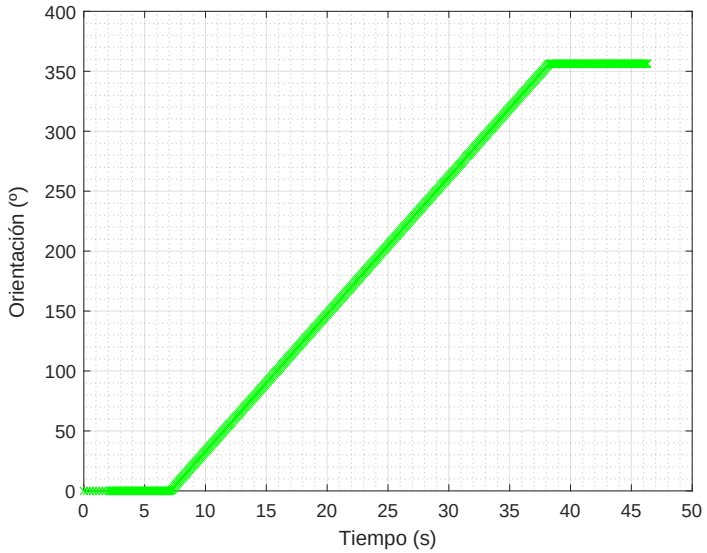


Figura 6.6: Representación orientación en º.

Para analizar el desplazamiento del centro, de cara a ver el error, hemos utilizado Matlab para representar tanto el desplazamiento en x, como el desplazamiento en y, siendo estos de aproximadamente 0,3 cm. Lo cual se ha considerado un buen resultado. En las siguientes figuras, se puede apreciar:

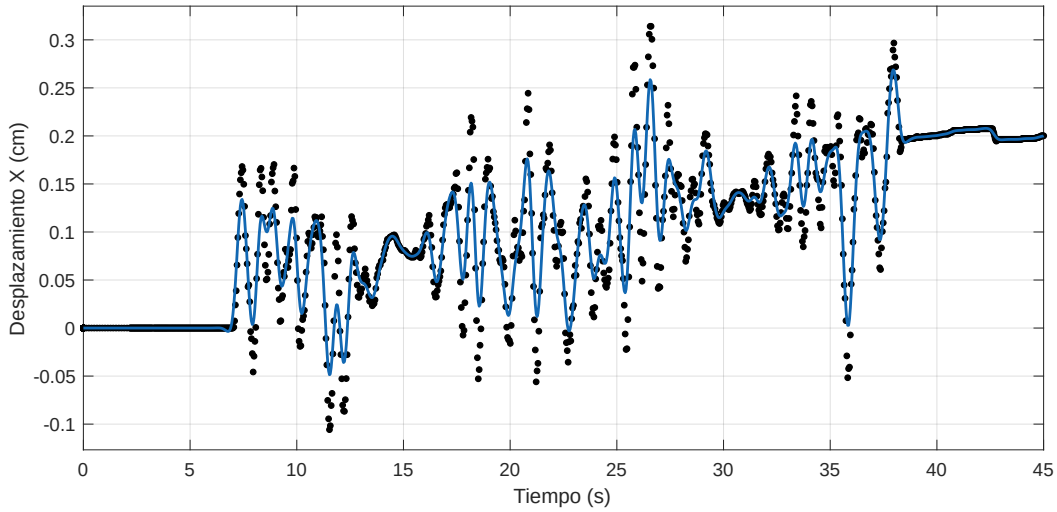


Figura 6.7: Desplazamientos en X provocados en el giro.

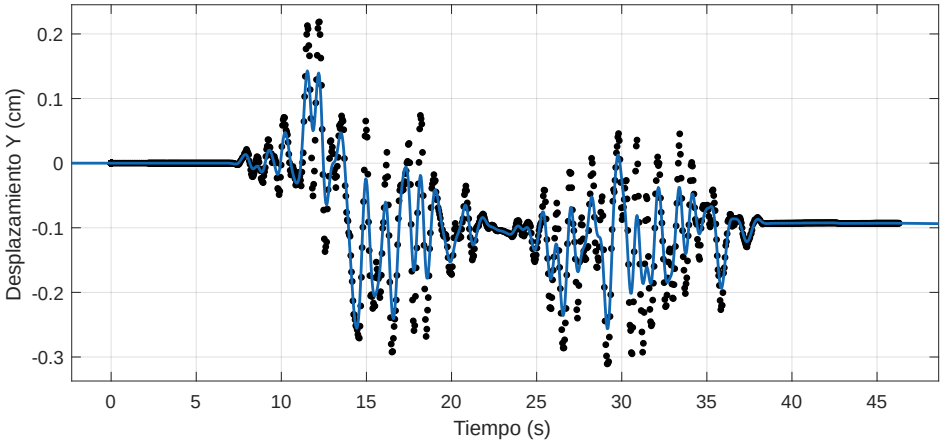
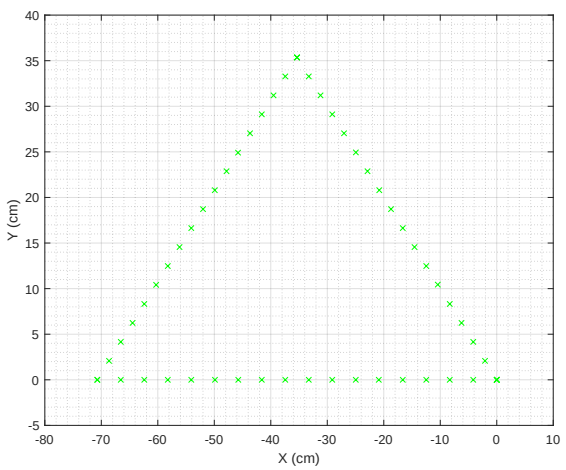


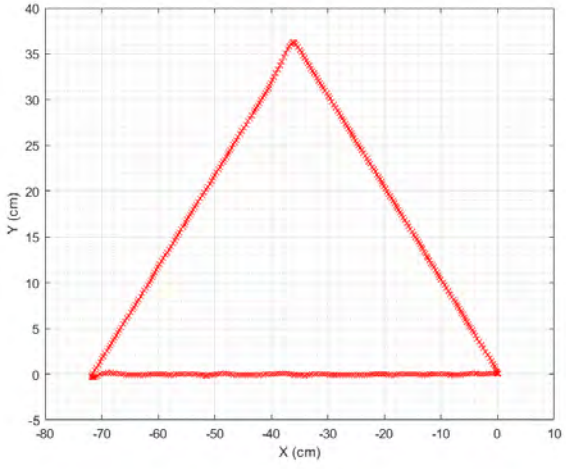
Figura 6.8: Desplazamientos en Y provocados en el giro.

6.2.3. Experimento 3: Seguimiento de camino triangular

El tercer experimento realizado ha sido seguir una trayectoria con forma de triángulo isósceles de 1 metro de longitud (lados iguales) y 45 grados. Es decir, dos desplazamientos diagonales y uno transversal.



(a) Camino ideal.



(b) Camino real.

Figura 6.9: Camino triangular.

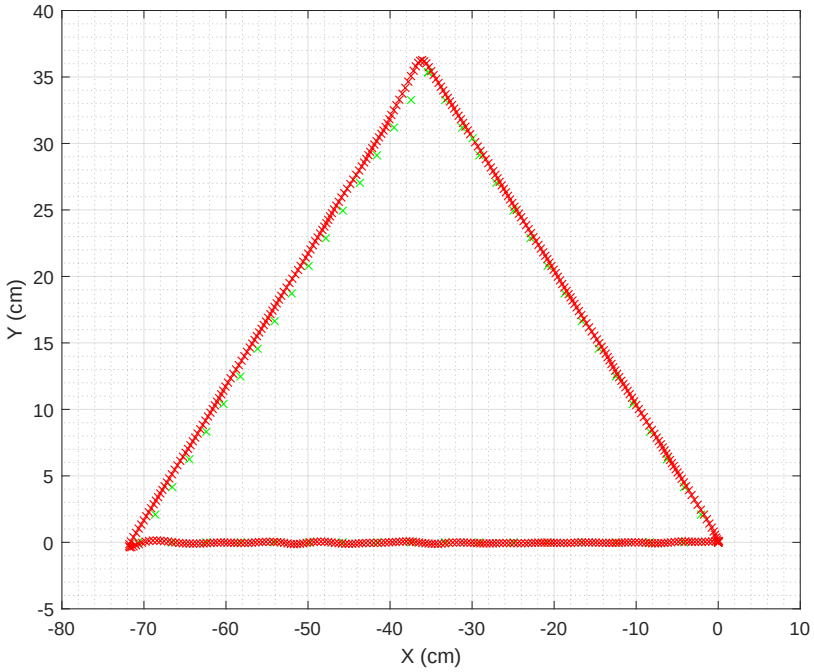


Figura 6.10: Comparación de caminos.

Al igual que en el cuadrado podemos apreciar errores debido al tiempo de frenada de la plataforma, en los cuales sobrepasa la distancia establecida.

Hacemos la representación de los desplazamientos y de la rotación con respecto al tiempo:

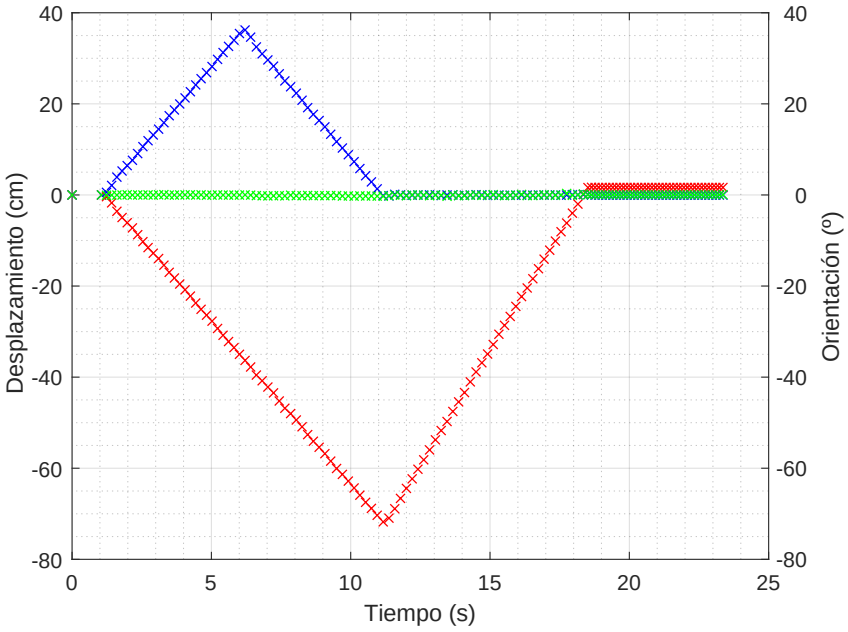


Figura 6.11: Trayectoria.

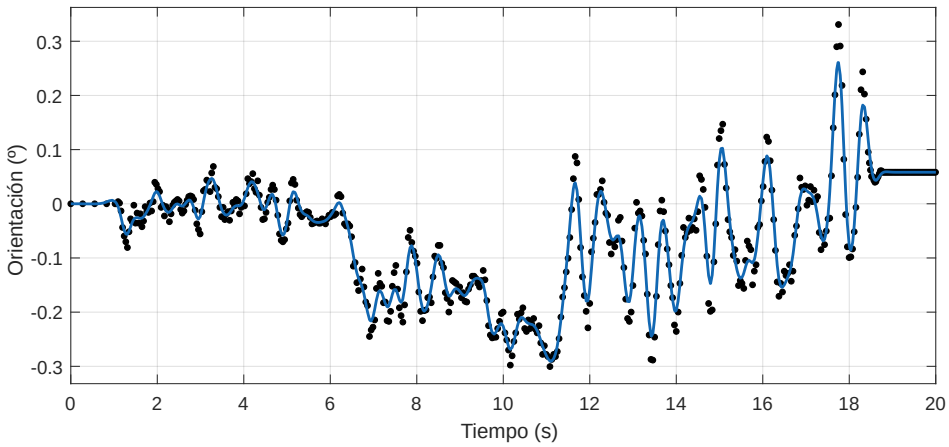
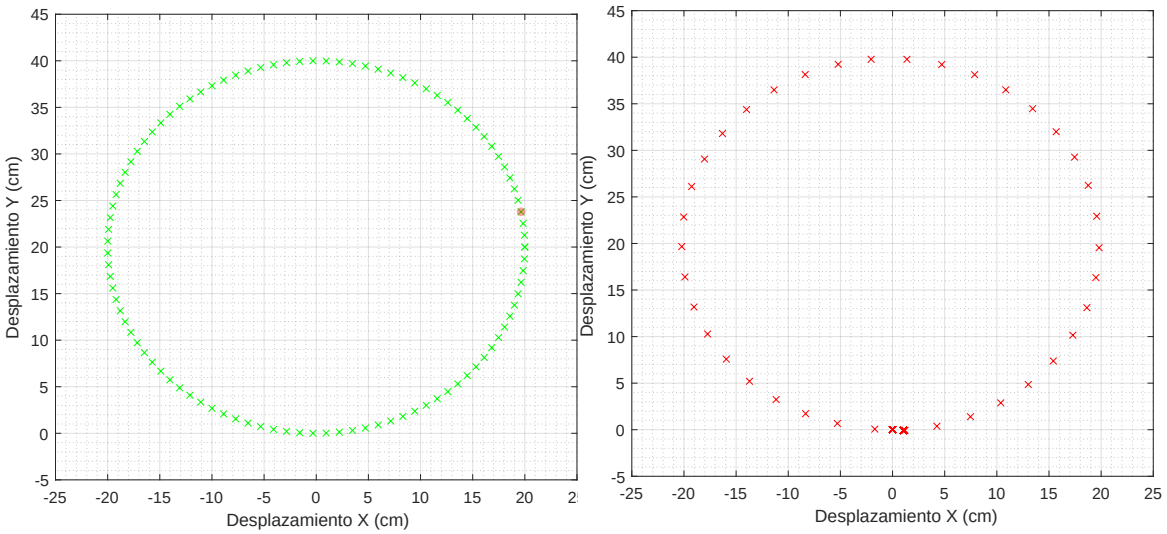


Figura 6.12: Orientación con respecto al tiempo.

Al igual que con los experimentos anteriores apreciamos que la rotación máxima se da en torno a 0,3º

6.2.4. Experimento 4: Realización de camino circular

Como último experimento se decidió realizar un camino más complejo, que combina desplazamiento y rotación. Este tipo de movimiento es el que más nos interesa, y más maniobrabilidad nos dan este tipo de configuración de ruedas. Se realizó un giro sobre el punto central del frontal de la plataforma, de forma que el centro del robot realizara una circunferencia de 40 cm de diámetro, y una rotación de 360º:



(a) Camino ideal.

(b) Camino real.

Figura 6.13: Camino circular.

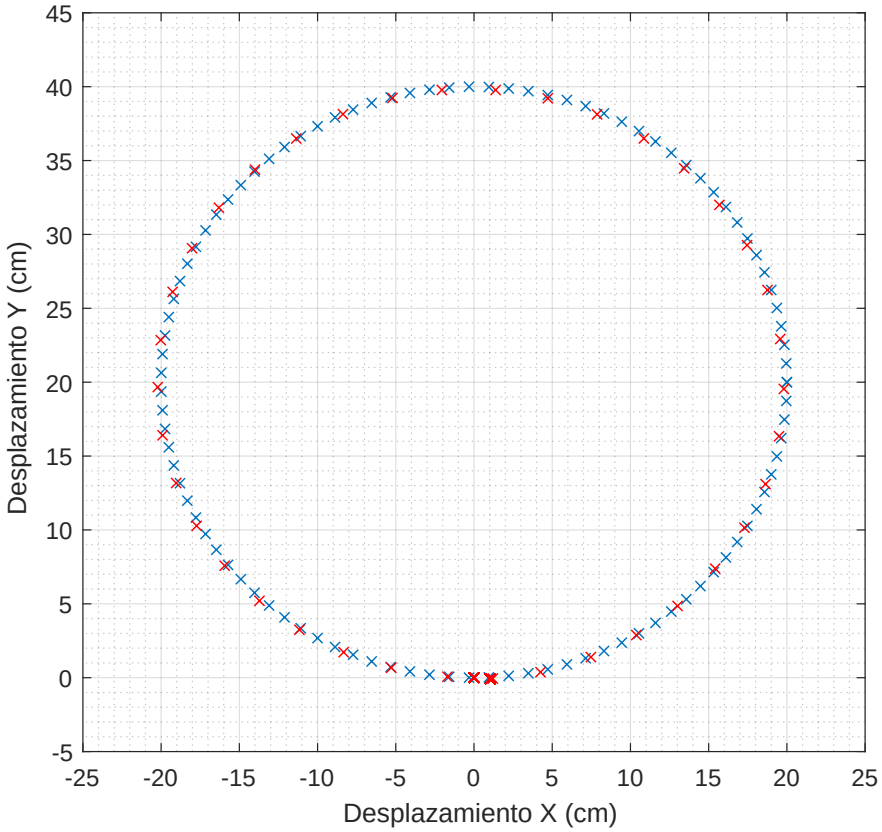


Figura 6.14: Comparación de caminos.

Se puede apreciar que el resultado es bastante exacto, habiendo unos errores máximos de décimas de centímetro.

Se representa también los desplazamientos y la rotación con respecto al tiempo:

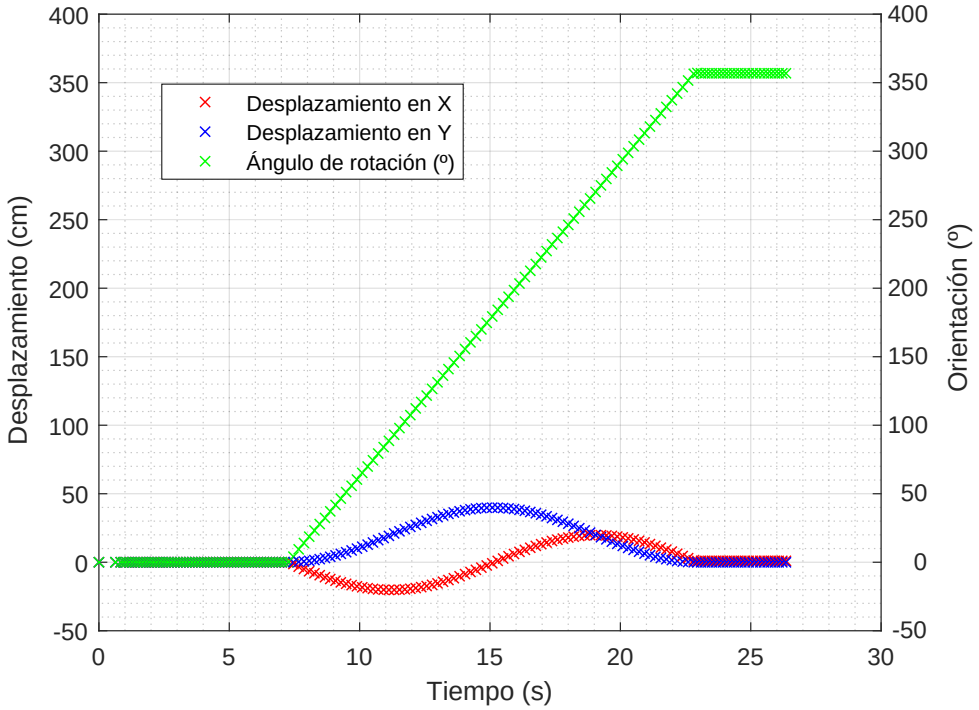


Figura 6.15: Trayectoria triangular.

Como se observa, tanto los desplazamientos como la rotación dan muy buenos resultados:

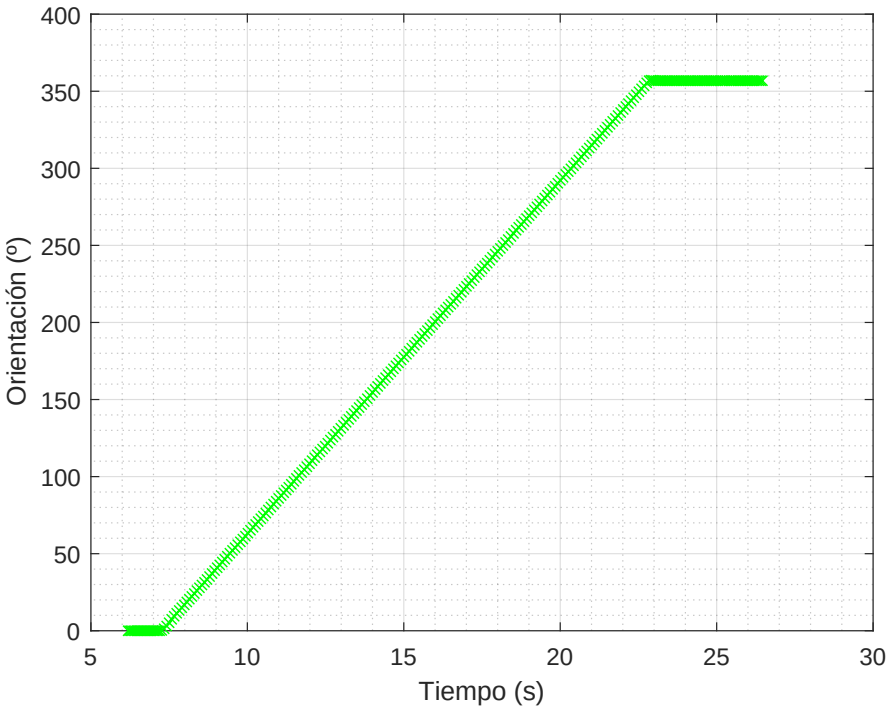


Figura 6.16: Orientación con respecto al tiempo.

6.2.5. Conclusiones

Con estos experimentos hemos obtenido resultados satisfactorios, de manera que podemos confirmar que el robot realiza los movimientos que se le exigen con un error mínimo. Concluimos también que la medida de la posición real mediante el software de lectura de los encoders es bastante exacto también. Se ha determinado que el error obtenido oscila ente el 1 % y el 3 % del desplazamiento total.

Capítulo 7

Conclusiones y líneas futuras

Contenido

7.1. Conclusiones

En este trabajo, en primer lugar, se ha realizado un estudio de la cinemática del tipo de ruedas elegidas. A continuación, se han diseñado, elegido y/o fabricado, todas las partes mecánicas y electrónicas del robot. Seguidamente se ha realiazado el montaje y el diseño del software de control. Y finalmente se ha realizado una serie de experimentos con el consiguiente ajuste de los controladores, obteniendo unos resultados satisfactorios.

Si hacemos un rápido análisis de los objetivos propuestos al inicio del trabajo podemos concluir que la plataforma móvil construida es totalmente funcional, el algoritmo de funcionamiento cinemático *1* desarrollado en *3.3* funciona de manera correcta y realiza todos los movimientos que se solicitan, de manera eficaz, siendo un diseño de fácil configuración para futuras modificaciones. Se ha realizado una elección de todos los componentes electrónicos *2*, como se ha visto en *4.3* de manera que se satisfacen todos los requerimientos. El objetivo *3* se realizó de manera correcta gracias a las facilidades que nos aportan las controladoras Roboclaw y su software, como se ha descrito en *6.2*.

Como se ha descrito en *4.3.2* se ha diseñado de manera satisfactoria un sistema de control de velocidad cartesiano *4* el cual nos permite una maniobra-

bilidad total de la plataforma. Al igual que se ha obtenido el sistema de control, se ha diseñado un sistema de obtención de datos de odometría *5*, visto en *4.3.2*, que como se ha visto en los experimentos, realiza un seguimiento muy acertado de la trayectoria real. En el mismo capítulo se describe como se ha solventado el objetivo *6* de manera que se ha dotado al sistema de una herramienta de control totalmente inalámbrica. No hay que olvidar que la finalidad de este robot móvil es la de realizar trayectorias factibles en el entorno doméstico, por lo cual concluimos que con los experimentos realizados, objetivo *7* de este trabajo, y la maniobrabilidad de que dispone la plataforma, será capaz de satisfacer las necesidades adecuadamente. Dichos experimentos, descritos en *5*, nos permiten afirmar el correcto funcionamiento. Especificar que el hardware que se ha utilizado es el necesario para desempeñar las funciones solicitadas, aunque como veremos en el siguiente apartado se pueden realizar ciertas mejoras.

Sirva el presente documento, en conjunto con el *Apéndice A*, como guía para los desarrolladores de niveles superiores de control de la plataforma, siendo este el último objetivo propuesto en este trabajo *8*.

Por último cabe destacar que para la realización de este trabajo se han utilizado conocimientos de distintos campos de la ingeniería, dando como resultado un robot omnidireccional con una configuración de 4 ruedas Mecanum totalmente funcional.

7.2. Líneas Futuras

Con el objetivo de aprovechar al máximo tanto la electrónica como la mecánica de este robot, se proponen ciertas mejoras de cara al futuro.

Para empezar, como se ha comentado anteriormente, la electrónica usada ha sido la necesaria, sin embargo, se podrían introducir algunos componentes que mejorarían el control y el posicionamiento real del robot. Una notable mejora sería la introducción de cámaras y sensores de proximidad, de manera que se implemente la capacidad de sortear obstáculos más fácilmente.

Otra mejora podría ser la de implementar en el software un control de posición en combinación con el control de velocidad realizado en este trabajo. De manera que se pueda elegir entre un modo u otro de manejo, ya que en ciertos momentos será útil un control de posición más exacto, que no presente los errores de aceleración y deceleración que se han observado en la realización de los experimentos de este trabajo.

Otro aspecto a mejorar, sería el incluir comunicaciones vía internet. Como

se comentó, uno de los motivos de elegir el microcontrolador escogido fue la disponibilidad de un módulo wifi. Este tipo de comunicaciones supondrían una gran mejora sobre un robot doméstico como este, a la hora de manejarlo desde fuera de casa.

Estas mejoras son solo propuestas que han surgido durante la realización del trabajo, pero queda abierto a la introducción de cualquier otra funcionalidad que tenga como objetivo aumentar la precisión, optimización de energía, aprovechamiento del software ya incluido, adición de nuevo, etc.

Bibliografía

[1] Daniela Rus. Robótica: una década de transformaciones. https://www.bbvaopenmind.com/articulos/robotica-una-decada-de-transformaciones/, 2019. Último acceso 10 Mayo 2021.

[2] Victor Fernando Muñoz Martinez. *Planificación de Trayectorias para Robots Móviles*. Escuela de Ingenierías Industriales, 1995.

[3] Wikipedia. Holónomo (robótica). https://es.wikipedia.org/wiki/Holónomo_(robótica). Último acceso 14 Mayo 2021.

[4] Raúl Rojas Jochen Brunhorn, Oliver Tenchio. *A Novel Omnidirectional Wheel Based on Reuleaux-Triangles*. Springer, Berlin, Heidelberg, 1 edition, 2007.

[5] Carolina Chang Lu y Danilo Díaz Tarascó. Robot omnidireccional. http://wikitronica.labc.usb.ve/index.php/Robot_omnidireccional#:~:text=Un%20robot%20omnidireccional%2C%20tambi%C3%A9n%20conocido,alcanzar%20previamente%20una%20orientaci%C3%B3n%20espec%C3%ADfica.&text=Por%20lo%20general%2C%20presentan%20una,de%20tres%20o%20cuatro%20ruedas. Último acceso 21 Mayo 2021.

[6] Luis Llamas. Robot cn mecanum wheel controlado por arduino. https://www.luisllamas.es/robot-con-mecanum-wheel-controlado-por-arduino/. Último acceso 14 Mayo 2021.

[7] Nurallah Ghaeminezhad Hamid Taheri, Bing Qiao. *Kinematic Model of a Four Mecanum Wheeled Mobile Robot*. International Journal of Computer Applications, 2015.

[8] Patrick F. Muir y Charles P. Neuman. *Kinematic Modeling for Feedback Control of an Omnidirectional Wheeled Mobile Robot*. Carnegie Mellon University, 1987.

[9] Vinssa Industrial Solutions. Robots móviles: la madurez de la tecnología. https://blog.vinssa.com/robots-moviles-la-madurez-de-la-tecnologia. Último acceso 31 Mayo 2021.

[10] Robotnik. La robótica móvil en la industria 4.0. https://www.interempresas.net/Robotica/Articulos/323571-La-robotica-movil-en-la-Industria-40.html. Último acceso 31 Mayo 2021.

[11] José Rafael; Silva Ortigoza Ramón Barrientos Sotelo, Víctor Ricardo; García Sánchez. *Robots Móviles: Evolución y Estado del Arte*. Polibits, Distrito Federal, México, 1 edition, 2007.

[12] H. R. Everett J. Borenstein and L. Feng. *Where am I? sensors and methods for mobile robot positioning*. The University of Michigan, 1996.

[13] J. Borenstein and L. Feng. *UMBmark: A benchmark test for measuring odometry errors in mobile robots*. SPIE Conference on Mobile Robots, Philadelphia, 1995.

[14] A. Bucken and S. Thrun. *Learning maps for indoor mobile robot navigation*. Carnegie Mellon University, Pittsburgh, 1996.

[15] J. L. Crowley. *World modeling and position estimation for a mobile robot using ultrasonic rangin*. IEEE International Conference on Robotics and Automation, Scottsdale, Arizona, 1 edition, 1989.

[16] J. T. Schwartz y M. Sharir. *On the piano movers problem: II. General technique for computing topological properties of real algebraic manifolds*. Advanced in applied Mathematics, New York, 1983.

[17] N. Nilsson. *A mobile automaton: an application of artificial intelligence techniques*. Proceedings of the International Joint Conference on Artificial Intelligence, Menlo Park, California, 1969.

[18] O. Khatib. *Real-Time obstacle avoidance for manipulators and mobile robots*. IEEE International Conference on Robotics and Automation, St. Louis, MO, USA, 1985.

[19] Joaquim A. Batlle. Robot with only steering wheels. https://www.researchgate.net/figure/Robot-with-only-steering-wheels_fig5_220143477. Último acceso 18 Mayo 2021.

[20] Javier Ruiz del Solar. Robots con orugas y robots robots con orugas y robots articulados. https://www.u-cursos.cl/ingenieria/2008/2/EL710/1/material_docente/detalle?id=186565. Último acceso 24 Mayo 2021.

[21] Luis del Valle Hernández. Esp8266 todo lo que necesitas saber del módulo wifi para arduino. https://programarfacil.com/podcast/esp8266-wifi-coste-arduino/#Que_es_el_ESP8266. Último acceso 10 Abril 2021.

[22] Juan Carlos Macho. Controlando el rover con un mando ps2. https://www.prometec.net/controlador-ps2/. Último acceso 12 Abril 2021.

[23] BasicMicro. Auto tuning with motion studio. https://resources.basicmicro.com/auto-tuning-with-motion-studio/. Último acceso 25 Abril 2021.

Apéndice A

Software del microcontrolador

```
#define DEBUG_ARRAY(a) {for (int index = 0; index < sizeof(a...
    ) / sizeof(a[0]); index++) {Serial.print(a[index]); ...
    Serial.print('\t');} Serial.println();};
#include <TimerOne.h> //Libreria para temporizaciones
#include <PS2X_lib.h> //Libreria para el uso del mando PS2
#include "RoboClaw.h" //Libreria para controladoras Roboclaw

//Se definen las direcciones de cada controladora
#define address1 0x80
#define address2 0x81

//Se definen las dimensiones de la plataforma
#define lx 20.0
#define ly 20.0
#define r 10.0

// Creamos la clase del mando PS2
PS2X ps2x;

//Se definen los puertos de comunicaci n con cada ...
    controladora
RoboClaw roboclaw(&Serial1, 10000);
RoboClaw roboclaw2(&Serial2, 10000);


int error = 0;
int cont = 1;

//Pines de lectura del joystick
const int analogPin0 = A0;
const int analogPin1 = A1;
```

```
29 const int analogPin2 = A2;
30
31 //variables que almacena la lectura anal gica del joystick ...
      o mando PS2
32 int value0;
33 int value1;
34 int value2;
35
36 //Variables que almacenan la velocidad en cm/s y la ...
      velocidad de rotaci n en rev/min
37 float vely;
38 float velx;
39 float velw;
40
41 float b = 0; //variable para nivel de velocidad
42
43 //Constantes y cadena para la lectura de vectores por puerto...
      serie
44 String str = "";
45 const char separator = ',';
46 const int dataLength = 3;
47 int data[dataLength];
48
49 //Variables para los valores de velocidad en cm/s y rad/s
50 float vx;
51 float vy;
52 float wz;
53
54 //Variables para la velocidad de cada motor en rad/s
55 float w1, w2, w3, w4;
56
57 //Variables para la velocidad de cada motor en rev/s
58 float w1rev, w2rev, w3rev, w4rev;
59
60 //Variables para la velocidad de cada motor en pulsaciones/s...
      del encoder
61 float w1pulsos, w2pulsos, w3pulsos, w4pulsos;
62
63
64 //Variables para realizar el control
65 float ref1, ref2, ref3, ref4;
66
67 //Variables de la posicion real
68 float x = 0;
69 float y = 0;
70 float a = 0;
71
72 //Variables de la velocidad real
73 volatile float vxreal, vyreal, wzreal;
```

```
74
75 //Variables de la velocidad real en rad/s
76 float w1real;
77 float w2real;
78 float w3real;
79 float w4real;
80
81 //Variable que almacena la posicion de los encoders
82 int long pos1;
83 int long pos2;
84 int long pos3;
85 int long pos4;
86
87 //Contadores para el recuento de tiempo para el calculo de ...
     velocidad real y desplazamiento
88 unsigned long tiempo = 0;
89 unsigned long tiempoActual = 0;
90
91 //Variables para la comparacion de contadores de encoders
92 int long Enc1 = 0;
93 int long Enc2 = 0;
94
95 void setup()
96 {
97   //Se indica que pines son usados por el mando PS2
98   error = ps2x.config_gamepad(28, 24, 26, 22, true, true); ...
      //(clock, command, attention, data, Pressures, Rumble)
99
100  //Se inicializan a 0 los valores de control para que al ...
      arrancar el microcontrolador no haya ningun valor ...
      almacenado
101  ref1 = ref2 = ref3 = ref4 = 0;
102
103  //Abrimos los puertos Serie, configuramos los datos a ...
      38400 bps
104  roboclaw.begin(38400);
105  roboclaw2.begin(38400);
106  Serial.begin(38400);
107
108  Timer1.initialize(10000); // Dispara la interrupcion cada ...
      10 ms
109  Timer1.attachInterrupt(ISR_LecVel); // Activa la ...
      interrupcion y la asocia a ISR_LecVel
110
111 }
112
113
114 void loop(void)
115 {
```

```
116  //Esta parte del codigo se utiliza para leer vectores a ...
       traves del puerto serie.
117  //Se deja como comentario ya que usaremos los otros modos
118  /*OBTENER EL VECTOR MEDIANTE PUERTO SERIE
119    if (Serial.available())
120    {
121     str = Serial.readStringUntil('\n');
122     for (int i = 0; i < dataLength ; i++)
123     {
124        int index = str.indexOf(separator);
125        data[i] = str.substring(0, index).toInt();
126        str = str.substring(index + 1);
127     }
128     DEBUG_ARRAY(data);
129    }
130    Serial.print("vector 1: ");
131    Serial.println(data[0]);
132     Serial.print("vector 2: ");
133     Serial.println(data[1]);
134        Serial.print("vector 3: ");
135        Serial.println(data[2]);
136
137    vx=data[0];
138    vy=data[1];
139    wz=data[2];*/
140
141  //Esta parte se usa para leer leer los valores de ...
       velocidad de un joystick
142  //Tambien se deja como codigo ya que en este trabajo no se...
       usa
143  /*
144    //Lectura de los valores anal gicos y transformaci n ...
       al rango de velocidad que queremos
145    value0 = analogRead(analogPin0);
146    vely = map(value0, 400, 620, -50, 50); // convertimos a ...
       velocidad de cm/s
147    value1 = analogRead(analogPin1);
148    velx = map(value1, 400, 620, -50, 50);
149    value2 = analogRead(analogPin2);
150    velw = map(value2, 0, 540, -20, 20); //convertimos a rev...
       /min
151
152    //Con esto dejamos un rango de holgura del joystick sin ...
       que el robot se mueva
153    if (velx < 2.5 && velx > -2.5)
154      vx = 0;
155    else
156      vx = velx;
157    if (vely < 2.5 && vely > -2.5)
```

```
158         vy = 0;
159       else
160         vy = vely;
161       if (velw < 2.5 && velw > -2.5)
162         wz = 0;
163       else
164         wz = 2 * 3.1416 * velw / 60; //a wz le doy el valor en...
          rad/s
165     */
166
167     //Control con mando PS2
168     //Realizamos la lectura del mando
169     ps2x.read_gamepad();
170
171     //Si L2 est  pulsado entramos a esta parte del codigo ...
        destinada a realizar trayectorias
172     //Si no lo esta pasamos al control por botones
173     if (ps2x.Button(PSB_L2) )
174     {
175       //Si se pulsa el boton X se realiza un giro sobre el ...
        punto central del frontal de la plataforma
176       if (ps2x.Button(PSB_BLUE))
177       {
178         delay(3000);
179         //Tomamos el valor de uno de los encoders
180         Enc1 = roboclaw.ReadEncM2(address1);
181
182         //realizamos esto mientras que la diferencia entre los...
          contadores sea inferior a 100500 lo cual equivale a la ...
        rotacion completa del robot
183         do {
184           //Establecemos la velocidad a cada motor
185           roboclaw.SpeedM1(address1, -6375);
186           roboclaw.SpeedM2(address1, 6375);
187           roboclaw2.SpeedM2(address2, 2125);
188           roboclaw2.SpeedM1(address2, -2125);
189
190           //Se actualiza con el valor actual del encoder en ...
        cada ciclo
191           Enc2 = roboclaw.ReadEncM2(address1);
192
193           //Llamamos a esta funcion que mide las posiciones de...
          los encoders y en tiempo entre cada medicion, para ...
        despues calcular valores reales
194           medirposicion();
195         }
196         while ((Enc2 - Enc1) < 100500);
197
```

```
198       //Establecemos todos los motores a velocidad 0 una vez...
      realizada la trayectoria
199       roboclaw.SpeedM1(address1, 0);
200       roboclaw.SpeedM2(address1, 0);
201       roboclaw2.SpeedM2(address2, 0);
202       roboclaw2.SpeedM1(address2, 0);
203     }
204
205     //Si se pulsa el boton cuadrado se realiza la ...
      trayectoria de un cuadrado
206     if (ps2x.Button(PSB_PINK)) //Robot realiza un cuadrado
207     {
208       //Mismo procedimiento que con la trayectoria anterior
209       Enc1 = roboclaw.ReadEncM1(address1);
210
211       //Primer desplazamiento longitudinal
212       do {
213         roboclaw.SpeedM1(address1, 2705);
214         roboclaw.SpeedM2(address1, 2705);
215         roboclaw2.SpeedM2(address2, -2705);
216         roboclaw2.SpeedM1(address2, -2705);
217
218         Enc2 = roboclaw.ReadEncM1(address1);
219         medirposicion();
220
221       } while ((Enc2 - Enc1) < 5 * 2705);
222
223       roboclaw.SpeedM1(address1, 0);
224       roboclaw.SpeedM2(address1, 0);
225       roboclaw2.SpeedM2(address2, 0);
226       roboclaw2.SpeedM1(address2, 0);
227
228       Enc1 = roboclaw.ReadEncM2(address1);
229       //Primer desplazamiento transversal
230       do {
231         roboclaw.SpeedM1(address1, -2705);
232         roboclaw.SpeedM2(address1, 2705);
233         roboclaw2.SpeedM2(address2, -2705);
234         roboclaw2.SpeedM1(address2, 2705);
235
236         Enc2 = roboclaw.ReadEncM2(address1);
237         medirposicion();
238
239       } while ((Enc2 - Enc1) < 2705 * 5);
240
241       roboclaw.SpeedM1(address1, 0);
242       roboclaw.SpeedM2(address1, 0);
243       roboclaw2.SpeedM2(address2, 0);
244       roboclaw2.SpeedM1(address2, 0);
```

```
245
246         Enc1 = roboclaw2.ReadEncM2(address2);
247         //Segundo desplazamiento longitudinal
248         do {
249           roboclaw.SpeedM1(address1, -2705);
250           roboclaw.SpeedM2(address1, -2705);
251           roboclaw2.SpeedM2(address2, 2705);
252           roboclaw2.SpeedM1(address2, 2705);
253           Enc2 = roboclaw2.ReadEncM2(address2);
254           medirposicion();
255
256         } while ((Enc2 - Enc1) < 2705 * 5);
257
258         roboclaw.SpeedM1(address1, 0);
259         roboclaw.SpeedM2(address1, 0);
260         roboclaw2.SpeedM2(address2, 0);
261         roboclaw2.SpeedM1(address2, 0);
262
263         Enc1 = roboclaw.ReadEncM1(address1);
264         //Segundo desplazamiento transversal
265         do {
266           roboclaw.SpeedM1(address1, 2705);
267           roboclaw.SpeedM2(address1, -2705);
268           roboclaw2.SpeedM2(address2, 2705);
269           roboclaw2.SpeedM1(address2, -2705);
270           Enc2 = roboclaw.ReadEncM1(address1);
271           medirposicion();
272
273         } while ((Enc2 - Enc1) < 2705 * 5);
274
275         roboclaw.SpeedM1(address1, 0);
276         roboclaw.SpeedM2(address1, 0);
277         roboclaw2.SpeedM2(address2, 0);
278         roboclaw2.SpeedM1(address2, 0);
279       }
280
281       //Si se pulsa el boton triangulo se realiza la ...
      trayectoria de un triangulo
282       if (ps2x.Button(PSB_GREEN)) //Robot realiza un triangulo
283       {
284         Enc1 = roboclaw.ReadEncM2(address1);
285         //Primer desplazamiento diagonal
286         do {
287           roboclaw.SpeedM1(address1, 0);
288           roboclaw.SpeedM2(address1, 3826);
289           roboclaw2.SpeedM2(address2, -3826);
290           roboclaw2.SpeedM1(address2, 0);
291           Enc2 = roboclaw.ReadEncM2(address1);
292           medirposicion();
```

```
293       } while ((Enc2 - Enc1) < 3826 * 5);
294
295       Enc1 = roboclaw2.ReadEncM1(address2);
296       //Segundo desplazamiento diagonal
297       do {
298         roboclaw.SpeedM1(address1, -3826);
299         roboclaw.SpeedM2(address1, 0);
300         roboclaw2.SpeedM2(address2, 0);
301         roboclaw2.SpeedM1(address2, 3826);
302         Enc2 = roboclaw2.ReadEncM1(address2);
303         medirposicion();
304       } while ((Enc2 - Enc1) < 3826 * 5);
305
306       Enc1 = roboclaw.ReadEncM1(address1);
307       //Desplazamiento transversal
308       do {
309         roboclaw.SpeedM1(address1, 2706);
310         roboclaw.SpeedM2(address1, -2706);
311         roboclaw2.SpeedM2(address2, 2706);
312         roboclaw2.SpeedM1(address2, -2706);
313         Enc2 = roboclaw.ReadEncM1(address1);
314         medirposicion();
315       } while ((Enc2 - Enc1) < 2706 * 7.1);
316
317       roboclaw.SpeedM1(address1, 0);
318       roboclaw.SpeedM2(address1, 0);
319       roboclaw2.SpeedM2(address2, 0);
320       roboclaw2.SpeedM1(address2, 0);
321     }
322
323     //Si se pulsa el bot n circulo se realiza una rotacion ...
      sobre si mismo de 360 grados
324     if (ps2x.Button(PSB_PAD_RIGHT))//Robot realiza un giro ...
      sobre si mismo
325     {
326       Enc1 = roboclaw.ReadEncM2(address1);
327       do {
328         roboclaw.SpeedM1(address1, -2125);
329         roboclaw.SpeedM2(address1, 2125);
330         roboclaw2.SpeedM2(address2, 2125);
331         roboclaw2.SpeedM1(address2, -2125);
332         Enc2 = roboclaw.ReadEncM2(address1);
333         medirposicion();
334       } while ((Enc2 - Enc1) < (67000));
335
336       roboclaw.SpeedM1(address1, 0);
337       roboclaw.SpeedM2(address1, 0);
338       roboclaw2.SpeedM2(address2, 0);
339       roboclaw2.SpeedM1(address2, 0);
```

```
340       }
341     }
342
343     //En el caso de que L2 no este pulsado directamente ... entramos aqui
344     else {
345       //En funci n de si L1,R1 o R2 estan pulsados se da un ... valor a la constante b
346       b = 0;
347       if (ps2x.Button(PSB_L1))
348         b = 1;
349       if (ps2x.Button(PSB_R1))
350         b = 1.5;
351       if (ps2x.Button(PSB_R2))
352         b = 2;
353
354       //Se realiza una lectura analogica de los joysticks del ... mando
355       value0 = ps2x.Analog(PSS_LX);
356       value1 = (-1) * ps2x.Analog(PSS_LY) + 255; //se ... intercambia el valor ya que el joystick da valores de ... menos a mayor de arriba a abajo
357       value2 = (-1) * ps2x.Analog(PSS_RY) + 255;
358
359       //Se transforman los valores de 0-255 a las velocidades ... que nos interesan
360       vely = map(value0, 0, 255, -b * 35, b * 35); //cm/s
361       velx = map(value1, 0, 255, -b * 35, b * 35);
362       velw = map(value2, 0, 255, -b * 6, b * 6); //rev/min
363
364       //Rango de holgura de los joysticks sin que el robot se ... mueva
365       if (velx < 2.5 && velx > -2.5)
366         vx = 0;
367       else
368         vx = velx;
369       if (vely < 2.5 && vely > -2.5)
370         vy = 0;
371       else
372         vy = vely;
373       if (velw < 2.5 && velw > -2.5)
374         wz = 0;
375       else
376         wz = 2 * 3.1416 * velw / 60; //a wz le doy el valor en... rad/s
377
378       //Si no se usan los joystick se realiza una lectura de ... los botones de direccion y se genera una velocidad ... constante al contrario que los joysticks
```

```
379     if (ps2x.Button(PSB_PAD_UP) && !ps2x.Button(PSB_PAD_DOWN...
        ) && !ps2x.Button(PSB_PAD_LEFT) && !ps2x.Button(...
        PSB_PAD_RIGHT))
380     {
381       vx = b * 15;
382       vy = wz = 0;
383     }
384     if (!ps2x.Button(PSB_PAD_UP) && ps2x.Button(PSB_PAD_DOWN...
        ) && !ps2x.Button(PSB_PAD_LEFT) && !ps2x.Button(...
        PSB_PAD_RIGHT))
385     {
386       vx = -b * 15;
387       vy = wz = 0;
388     }
389     if (!ps2x.Button(PSB_PAD_UP) && !ps2x.Button(...
        PSB_PAD_DOWN) && ps2x.Button(PSB_PAD_LEFT) && !ps2x....
        Button(PSB_PAD_RIGHT))
390     {
391       vy = b * 15;
392       vx = wz = 0;
393     }
394     if (!ps2x.Button(PSB_PAD_UP) && !ps2x.Button(...
        PSB_PAD_DOWN) && !ps2x.Button(PSB_PAD_LEFT) && ps2x....
        Button(PSB_PAD_RIGHT))
395     {
396       vy = -b * 15;
397       vx = wz = 0;
398     }
399     if (ps2x.Button(PSB_PAD_UP) && !ps2x.Button(PSB_PAD_DOWN...
        ) && ps2x.Button(PSB_PAD_LEFT) && !ps2x.Button(...
        PSB_PAD_RIGHT))
400     {
401       vy = b * 15;
402       vx = b * 15;
403       wz = 0;
404     }
405     if (ps2x.Button(PSB_PAD_UP) && !ps2x.Button(PSB_PAD_DOWN...
        ) && !ps2x.Button(PSB_PAD_LEFT) && ps2x.Button(...
        PSB_PAD_RIGHT))
406     {
407       vy = -b * 15;
408       vx = b * 15;
409       wz = 0;
410     }
411     if (!ps2x.Button(PSB_PAD_UP) && ps2x.Button(PSB_PAD_DOWN...
        ) && ps2x.Button(PSB_PAD_LEFT) && !ps2x.Button(...
        PSB_PAD_RIGHT))
412     {
413       vy = b * 15;
```

```
414         vx = -b * 15;
415         wz = 0;
416       }
417       if (!ps2x.Button(PSB_PAD_UP) && ps2x.Button(PSB_PAD_DOWN...
          ) && !ps2x.Button(PSB_PAD_LEFT) && ps2x.Button(...
          PSB_PAD_RIGHT))
418       {
419         vy = -b * 15;
420         vx = -b * 15;
421         wz = 0;
422       }
423       if (ps2x.Button(PSB_RED) && !ps2x.Button(PSB_PINK))
424       {
425         wz = -b * 0.6;
426         vx = vy = 0;
427       }
428       if (!ps2x.Button(PSB_RED) && ps2x.Button(PSB_PINK))
429       {
430         wz = b * 0.6;
431         vx = vy = 0;
432       }
433       //Si no se pulsa nada simplemente se mantiene a 0 las ...
          velocidades
434       if (!ps2x.Button(PSB_PAD_UP) && !ps2x.Button(...
          PSB_PAD_DOWN) && !ps2x.Button(PSB_PAD_LEFT) && !ps2x....
          Button(PSB_PAD_RIGHT) && !ps2x.Button(PSB_L1) && !ps2x....
          Button(PSB_RED) && !ps2x.Button(PSB_PINK) && !ps2x.Button...
          (PSB_R1) && !ps2x.Button(PSB_R2) )
435         vx = vy = wz = 0;
436
437       //Recibimos el vector [vx, vy , wz] y a partir de el ...
          calculamos w1,w2,w3 y w4, velocidades de las ruedas
438       w1 = (1 / r) * (vx - vy - (lx + ly) * wz); //r=radio de ...
          las ruedas, lx=mitad distancia entre rueda izq y der, ly=...
           mitad distancia rueda delantera y trasera
439       w2 = (1 / r) * (vx + vy + (lx + ly) * wz);
440       w3 = (1 / r) * (vx + vy - (lx + ly) * wz);
441       w4 = (1 / r) * (vx - vy + (lx + ly) * wz);
442
443       //El valor esta en rad/s lo pasamos a rev/min
444       w1rev = w1 / (2 * 3.1416);
445       w2rev = w2 / (2 * 3.1416);
446       w3rev = w3 / (2 * 3.1416);
447       w4rev = w4 / (2 * 3.1416);
448
449       //Lo pasamos a pulsaciones por segundo (2000 pulsos por ...
          vuelta)
450       ref1 = w1rev * 2000 * 8.5; //He usado 8.5 como el factor...
           de reducci n
```

```
451       ref2 = w2rev * 2000 * 8.5;
452       ref3 = w3rev * 2000 * 8.5;
453       ref4 = w4rev * 2000 * 8.5;
454
455       //Se mandan las velocidades a las controladoras
456       roboclaw.SpeedM1(address1, ref1);
457       roboclaw.SpeedM2(address1, ref2);
458       roboclaw2.SpeedM2(address2, -ref3);
459       roboclaw2.SpeedM1(address2, -ref4);
460       medirposicion()
461     }
462 }
463
464 //Funcion que mide la posicion de cada encoder y el tiempo ...
      entre cada lectura
465 void medirposicion()
466 {
467   //Tomamos los valores de cada encoder
468   pos1 = (roboclaw.ReadEncM1(address1)) ; //posici n en ...
      pulsos
469   pos2 = (roboclaw.ReadEncM2(address1));
470   pos3 = (-1) * (roboclaw2.ReadEncM2(address2)); //...
      multiplicado por -1 debido a que el encoder est  en ...
      sentido contrario
471   pos4 = (-1) * (roboclaw2.ReadEncM1(address2));
472
473   //Tomamos el tiempo entre cada medicion y la anterior
474   tiempo = millis() - tiempoActual;
475   tiempoActual = millis();
476
477   //Imprimimos por puerto serie los valores, en formato para...
      facilmente pasar a matlab
478   Serial.print("Medidas("); Serial.print(cont); Serial.print...
      (",1)=");
479   Serial.print(pos1); Serial.print(";");
480   Serial.print("Medidas("); Serial.print(cont); Serial.print...
      (",2)=");
481   Serial.print(pos2); Serial.print(";");
482   Serial.print("Medidas("); Serial.print(cont); Serial.print...
      (",3)=");
483   Serial.print(pos3); Serial.print(";");
484   Serial.print("Medidas("); Serial.print(cont); Serial.print...
      (",4)=");
485   Serial.print(pos4); Serial.print(";");
486   Serial.print("Medidas("); Serial.print(cont); Serial.print...
      (",5)=");
487   Serial.print(tiempo); Serial.println(";");
488   cont++;
489 }
```

```
490
491
492 //Funcion que actualiza los valores reales de posicion y ...
      orientacion de la plataforma
493 void ISR_LecVel(void)
494 {
495   //Se leen las velocidades de los encoders y se transforman...
       a rad/s
496   w1real = (roboclaw.ReadEncM1(address1)) * 2 * 3.1416 / 200...
      0;
497   w2real = (roboclaw.ReadSpeedM2(address1)) * 2 * 3.1416 / 2...
      000;
498   w3real = (roboclaw.ReadSpeedM2(address2)) * 2 * 3.1416 / 2...
      000;
499   w4real = (roboclaw.ReadSpeedM1(address2)) * 2 * 3.1416 / 2...
      000;
500
501   //Se calculan las velocidades longitudinal, transversal y ...
      de rotaci n
502   vxreal = (w1real + w2real + w3real + w4real) * r / 4; //...
      unidades en funcion de r
503   vyreal = (-w1real + w2real + w3real - w4real) * r / 4;
504   wzreal = (-w1real + w2real - w3real + w4real) * r / (4 * (...
      lx + ly));
505
506   //Se actualizan sus coordenadas
507   x = x + vxreal * 0.01; //actualizacion posicion x
508   y = y + vyreal * 0.01; //actualizacion posicion y
509   a = a + wzreal * 0.01; //actualizacion giro
510
511 }
```

Código A.1: Software de control