UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Estabilización experimental de un robot móvil diferencial alrededor de trayectorias tipo C2

T E S I S

Que para obtener el título de

INGENIERO MECATRÓNICO

P R E S E N T A

JOSÉ AGUSTÍN ROMERO ESQUINCA

DIRECTOR DE TESIS:
DR. VÍCTOR JAVIER GONZÁLEZ VILLELA

MÉXICO, D.F. NOVIEMBRE 2013

Dedicatoria

A Pita por mostrarme la valía de ser auténtico.

A mis padres José y Magdalena por el apoyo y amor incondicionales durante toda mi vida y, especialmente, durante mi formación profesional.

A Reyna por su amor, confianza, honestidad y ejemplo.

A Tía Nena, María Elena, resto de mi familia y amigos por el cariño que me han demostrado a lo largo de mi existencia.

Agradecimientos

A la Universidad Nacional Autónoma de México y a la Facultad de Ingeniería por las facilidades recibidas durante mi formación académica.

Al Dr. Víctor Javier González Villela por el asesoramiento, paciencia y apoyo durante la realización de esta tesis.

A la Dirección General de Asuntos del Personal Académico (DGAPA), UNAM, a través del proyecto IN115811 del PAPIIT, con título: "Investigación y desarrollo en sistemas mecatrónicos: robótica móvil, robótica paralela, robótica híbrida y teleoperación", por el apoyo para el desarrollo de este trabajo.

Resumen

El presente trabajo hace una introducción a la robótica y en específico a la robótica móvil, habla sobre los tipos de robots móviles, los cuales se clasifican de acuerdo al medio en el que se desenvuelven y conforme a su configuración. También aborda el problema de seguimiento de trayectorias C2 para una configuración en particular, que es la del robot móvil diferencial. Este trabajo comprueba experimentalmente la validez de una ley de control basada en una linealización del error y que utiliza un sistema de visión como realimentación de la posición y de la orientación del robot. Los resultados muestran que la planta fue capaz de seguir dos trayectorias distintas con diferentes valores de ganancias con una buena precisión.

Abstract

This work makes an introduction to robotics and specifically to mobile robotics. Mobile robots are classified according to their environment and configuration. This work discusses a control law that stabilizes a differential configuration mobile robot over a C2 trajectory. The system uses a camera to close the loop and wireless radios to communicate with the mobile robot. The results show that the robot was able to follow two different non-linear trajectories with precision.

Contenido

Capítulo 1

Introducción

La robótica es un campo de estudio relativamente reciente dentro de la tecnología moderna y cruza los límites de la ingeniería tradicional. Entender la complejidad de la robótica requiere de sólidos conocimientos en ingeniería eléctrica, ingeniería mecánica, ingeniería en sistemas, ciencias de la computación, matemáticas y física [1].

El término robot fue introducido por el escritor Karel Capek en 1920, mediante una obra de teatro titulada "Robots universales Rossum", la palabra *robot* deriva de la palabra *robota*, que significa "trabajo" o "esclavo", en checo o eslavo respectivamente. Tras el éxito de Capek y sus obras, la expresión fue introducida en múltiples lenguas. Desde entonces, esta palabra ha sido empleada para denominar un gran número de dispositivos mecánicos, como tele-operadores, vehículos sumergibles, etc. Prácticamente cualquier dispositivo que opere con algo de autonomía, bajo el control de una computadora, ha sido llamado robot. Una definición emitida por el *Robot Institute of America es :*"un robot es un manipulador reprogramable y multifuncional, diseñado para mover partes, herramientas o dispositivos especiales mediante movimientos variables programados para la ejecución de diversas tareas" [1].

Aunque es una definición que acota en demasía el conjunto de dispositivos que pueden ser considerados como robots, entre ellos los robots móviles, hace mención de la capacidad de ser reprogramados, lo que otorga a estos dispositivos una alta adaptabilidad.

Desde sus inicios con la tele-operación y sistemas maestro-esclavo hasta el presente con robots móviles autónomos, la existencia de un control ha sido fundamental para la correcta interacción del robot con el entorno. Para cumplir con dicha interacción, el robot debe ser capaz de recibir estímulos del medio y así poder planear de manera eficiente y precisa sus movimientos.

El objetivo de este trabajo es estabilizar experimentalmente alrededor de una trayectoria tipo C2 a un robot móvil con configuración diferencial, utilizando un control lineal y un sistema de visión como realimentación, dentro de este objetivo, se tiene la meta de obtener un error de 2 cm o menor, puesto que esta medida representa el 1% del espacio de trabajo del robot y se considera suficientemente preciso. Por otro lado, se desea analizar las diferencias en los resultados entre la simulación y el prototipo funcional, así como examinar el impacto en la respuesta del sistema ante diferentes valores en los parámetros del controlador

Para ello, el capítulo dos hace una introducción a la robótica móvil, habla de los tipos y configuraciones de los robots móviles y se detalla el modelo cinemático del robot utilizado en este experimento. Asimismo, se describe el procedimiento para obtener la ley de control en lazo abierto y en lazo cerrado que permite la estabilización del robot alrededor de trayectorias tipo C2.

En el capítulo tres se describen los elementos que conforman al robot móvil, el sistema de visión, se explica el acoplamiento y procesamiento de datos, su integración con Simulink® de MATLAB®, así como la comunicación entre todos los componentes que en conjunto permiten la realización de simulaciones y pruebas experimentales.

En el capítulo 4 se presentan los resultados de la simulación de un control de seguimiento de un robot móvil de configuración diferencial para una trayectoria sinusoidal y una trayectoria con forma de ocho, Asimismo, la simulación es útil para analizar las entradas requeridas y con ello saber si el sistema es capaz de alcanzar los

valores de entrada requeridos por el control con los parámetros ingresados por el usuario.

El capítulo 5 expone los resultados de las pruebas experimentales realizadas en el laboratorio con el robot móvil, el sistema de realimentación y el controlador expuesto en el capítulo 2. De igual manera, se discute el impacto que tuvo la variación de los parámetros del controlador realimentado en la respuesta del sistema real y la precisión que otorgó en cada experimento realizado.

Finalmente, en el capítulo 6 se presentan las conclusiones de la simulación y de los experimentos hechos con el prototipo funcional. También menciona las propuestas para mejorar el experimento.

Capítulo 2

Robot móvil con configuración diferencial y su estabilización alrededor de trayectorias tipo C2

2.1 Robótica móvil

2.1.1 Introducción

Un robot móvil es la combinación de dispositivos físicos y computacionales, en términos de los dispositivos físicos, también conocido como hardware, puede ser considerado como un conjunto de subsistemas, los cuales son:

- **Motriz:** la forma en que el robot se mueve en su área de trabajo
- **Sensado:** la forma en que el robot recibe información del medio que le rodeo o de él mismo
- **Control:** la forma en que el robot transforma estas mediciones en acciones
- **Comunicación:** la forma en que el robot interactúa con su operador.

Para que un robot autónomo o vehículo pueda moverse, es necesario aplicarle alguna fuerza. El estudio del movimiento donde dichas fuerzas son consideradas se le conoce como dinámica, mientras que la cinemática estudia el movimiento sin tomar en cuenta a dichas fuerzas, es decir, la cinemática analiza las relaciones geométricas que gobiernan al sistema y la dinámica incluye en su estudio las fuerzas asociadas al movimiento.

Con base en la aplicación, han surgido algunas categorías de robots móviles [2]:

- **Terrestres:** viajen sobre la superficie
- **Acuáticos:** operan en el agua
- **Aéreos:** a menudo imitan aviones o aves
- **Espaciales:** diseñados para trabajar en condiciones de baja gravedad o de gravedad cero.

Sin una fuente de energía, un robot móvil no es más que una pieza de hardware inerte. La gran mayoría de dispositivos autónomos depende de baterías para proveer energía eléctrica al sistema, estas baterías están especificadas acorde a su capacidad de almacenamiento en términos de amperios-hora $(A \cdot h)$. Lamentablemente, su capacidad nominal se ve disminuida rápidamente por el proceso de carga y descarga, temperatura, etc. [2]; esto es un problema persistente en dispositivos no sólo asociados con la robótica, sino en aparatos de uso común, como teléfonos celulares y computadoras personales.

Es necesario transformar la energía eléctrica en energía mecánica para que el robot móvil se mueva, en el caso de los robots móviles con ruedas, los motores de corriente directa o de CD, se han mostrado efectivos para cumplir con esta tarea. Dispositivos mecánicos como ruedas, actuadas por motores, se aprovechan de la fricción generada con la superficie para lograr que el robot móvil se desplace.

Se distinguen dos tipos básicos de ruedas, las convencionales, que incluyen las ruedas fijas, orientables centradas y orientables no centradas y las no convencionales.

Las ruedas fijas son aquellas que su orientación angular es constante con respecto al marco de referencia del robot y su posición está definida por las coordenadas *(x,y,α)* [5].

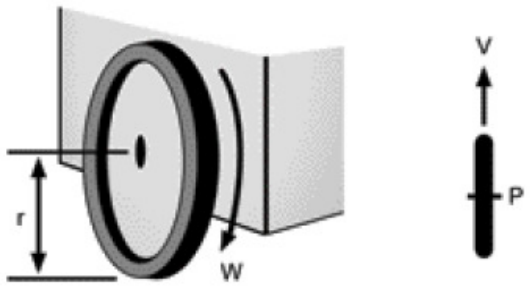


Figura 1 Rueda fija.

Las ruedas orientables centradas se comportan de una manera similar a las fijas, pero en este caso, su orientación angular con respecto al marco de referencia del robot es variable. El eje de giro de la orientación es perpendicular al eje de giro de la rueda [5].

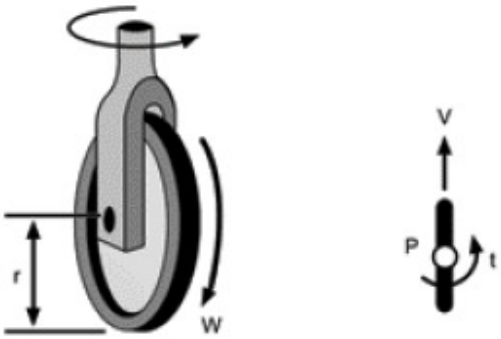


Figura 2 Rueda orientable.

Las ruedas orientables no centradas tienen una orientación variable con respecto al marco de referencia del robot y el eje de giro de la orientación no se interseca con el eje de giro de la rueda [5].

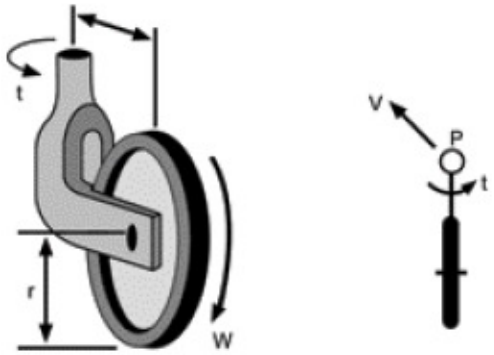


Figura 3 Rueda orientable no centrada.

Algunos vehículos ligeros están equipados con mecanismos adicionales o puntos de contacto que proveen soporte pero no contribuyen a la capacidad de orientarse o a la propulsión. Conocidas popularmente como ruedas locas, estos mecanismos pueden ser ignorados en el modelado cinemático del vehículo [2].

Las ruedas no convencionales están basadas en el concepto de tracción en un sentido y que permite un movimiento pasivo en otra dirección, por tanto, permiten mayor flexibilidad en ambientes congestionados. Estos diseños incluyen a la rueda universal, la rueda sueca, entre otras.

La rueda universal provee una combinación movimientos libres y con restricciones durante las vueltas. El mecanismo consiste en pequeños rodamientos localizados en el diámetro exterior de la rueda montados perpendicularmente al eje de rotación de la rueda.

La rueda sueca es similar a la universal en diseño, excepto que sus rodamientos están montados en ángulo. Esta configuración transmite una parte de la fuerza en la dirección de rotación de la rueda a una fuerza normal a la dirección de la misma.

2.1.2 Configuraciones comunes de robots móviles

De acuerdo al tipo de ruedas y al arreglo de las mismas, se pueden identificar configuraciones comunes dentro de la robótica móvil:

- De configuración diferencial
- Síncrono
- Triciclo
- Ackerman o tipo automóvil.

2.1.2.1 Robot móvil con configuración diferencial

Se le considera la configuración de accionamiento más simple para un robot móvil con ruedas. Es usado en pequeños y baratos robots destinados a tareas en interiores. Un

robot móvil con configuración diferencial consiste en dos ruedas montadas dos ejes colineales, cada una controlada por un motor diferente.

Bajo esta configuración, para que cada rueda produzca movimiento, el robot debe rotar alrededor de un punto ubicado en el eje de las ruedas denominado centro instantáneo de rotación o CIR Variando la velocidad relativa de ambas ruedas, el punto de rotación puede ser modificado, provocando que el robot describa diferentes trayectorias. Es importante notar que si la velocidad en ambas ruedas es la misma, el radio de la trayectoria descrita respecto al ICC tiende a infinito, por lo que el robot realizará una trayectoria recta. Si las velocidades son iguales en magnitud pero contrarias en sentido, el radio de la trayectoria descrita respecto al ICC es cero y el robot girará sobre un eje ubicado en medio de ambas ruedas del robot [2].

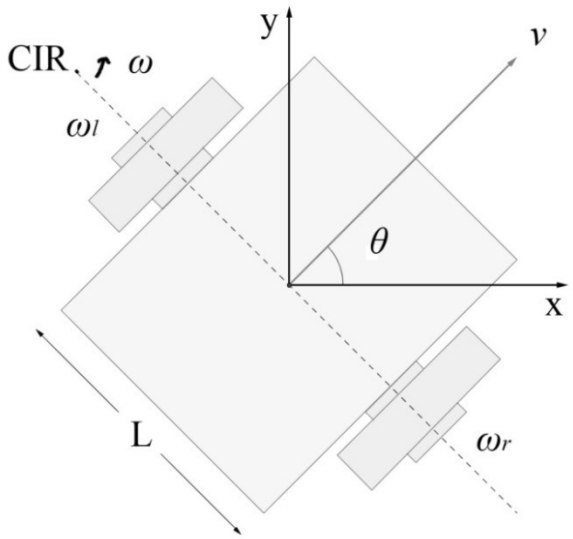


Figura 4 Robot móvil con configuración diferencial.

2.1.2.2 Robot móvil síncrono

En un robot móvil síncrono todas las ruedas giran y producen impulso al unísono, todas las ruedas apuntan en la misma dirección y giran al mismo tiempo. Estos vehículos controlan la dirección de las ruedas y la velocidad a la que giran. En esta configuración,

cada rueda es capaz tanto de producir un impulso como de modificar su orientación. Configuraciones típicas incluyen un arreglo de tres ruedas situadas en los vértices de un triángulo equilátero. Una rueda direccionable es aquella que permite controlar la orientación de su eje de rotación.

Una solución común en robots de este tipo es utilizar dos motores independientes, uno para producir desplazamiento y otro para rotar las ruedas.

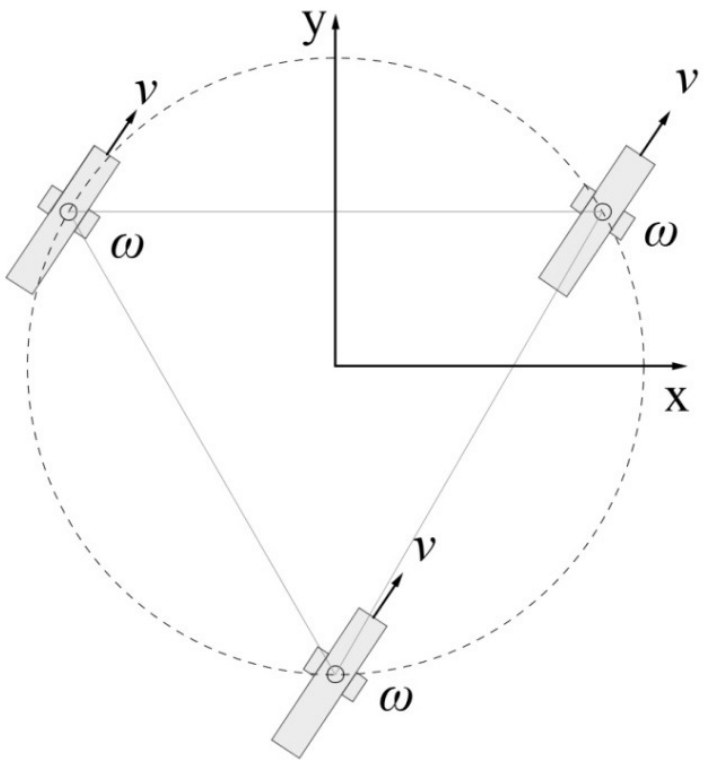


Figura 5 Robot móvil síncrono.

2.1.2.3 Robot móvil tipo triciclo

Un robot tipo triciclo clásico tiene tres ruedas, las dos traseras y la delantera que incorpora capacidad de orientación e impulso. El movimiento del robot es controlado por la rotación y giro de la llanta delantera [2].

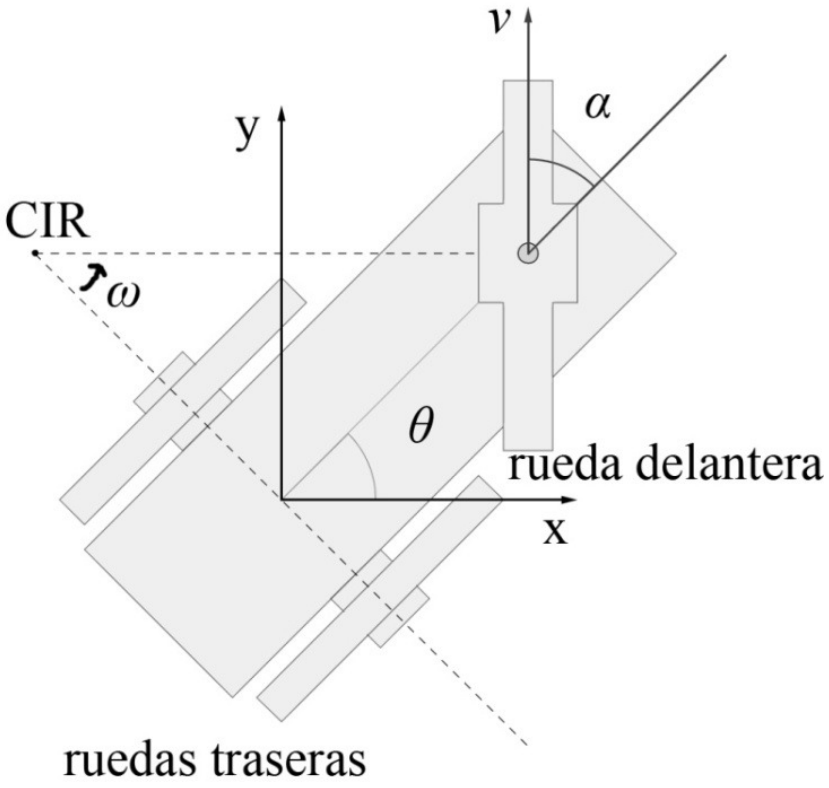


Figura 6 Robot móvil triciclo.

2.1.2.4 Robot móvil con configuración Ackerman

Este tipo de configuración se puede observar en la mayoría de los automóviles. En el modelo de dirección Ackerman, cada rueda delantera de dirección rota en brazos separados con el objetivo de permitir tener diferentes ángulos en cada rueda respecto al CIR. La rueda interna debe girar un ángulo mayor que la exterior y la rueda interior recorre una distancia menor que la externa.

La dirección Ackerman es el mecanismo preferido para vehículos grandes de los cuales se espera rueden sobre caminos existentes o sirvan para trasladar grandes cantidades de carga [2].

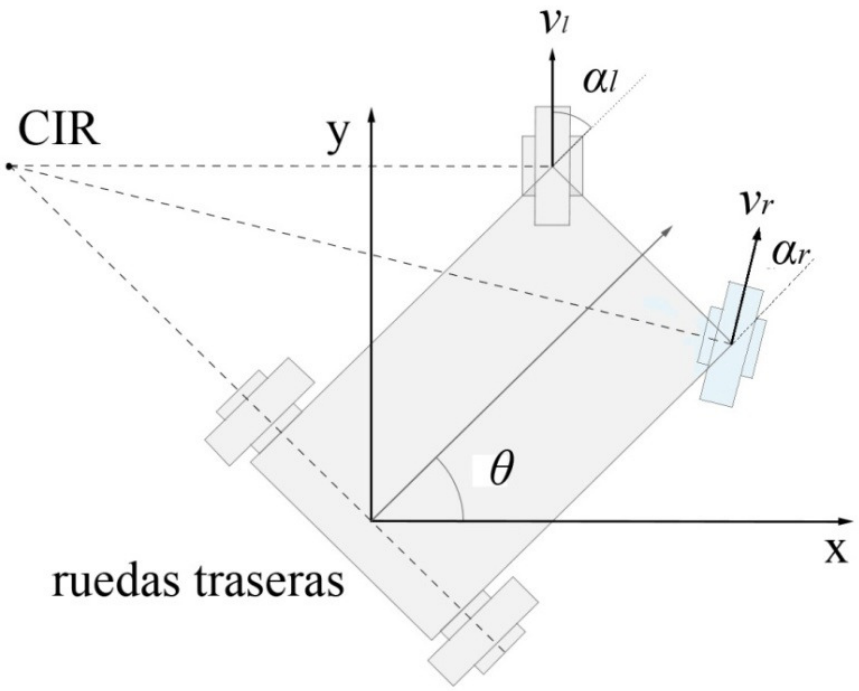


Figura 7 Robot móvil con configuración Ackerman.

2.1.3 Propiedades cinemáticas de los robots móviles con ruedas

González Villela [4] hace mención de tres propiedades cinemáticas de movimiento de un robot móvil.

Grado de movilidad (r_m)

Define el número de variables que se pueden controlar y, como consecuencia, los grados instantáneos de libertad que el robot puede manipular desde sus entradas sin orientar ninguna de sus ruedas.

Grado de manejabilidad (r_s)

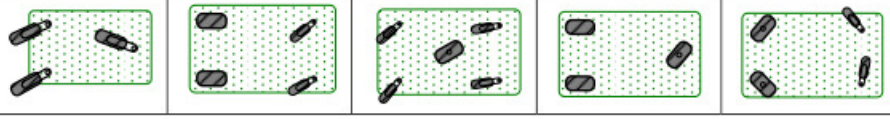
Define el número de ruedas orientables centradas que pueden ser controladas.

Grado de maniobrabilidad (r_M)

Define el total de grados de libertad que el robot puede manipular.

$$r_M = r_m + r_s \tag{1}$$

Tabla 1 Tipos de configuraciones de robots móviles con ruedas [4].

		Tipo(r_m, r_s)				
		(3,0)	(2,0)	(2,1)	(1,1)	(1,2)
Grado	r_m	3	2	2	1	1
	r_s	0	0	1	1	2
	r_M	3	2	3	2	3
Configuración del robot						
Ruedas motorizadas		 Fijas Centradas No centradas				
Ruedas no motorizadas		 Pasiva no centrada Pasiva centrada				

Como consecuencia de las definiciones anteriores, los robots móviles con ruedas son clasificados en cinco configuraciones no singulares, de acuerdo al grado de movilidad (grados instantáneos de libertad), a su grado de manejabilidad (número de grados de manejabilidad) y a la suma de ambos, definida como el grado de maniobrabilidad (total de grados de libertad). El grado de maniobrabilidad por sí mismo no define el tipo de robot móvil, debido a que dos robots con el mismo r_M pero diferente r_m no son equivalentes. Los valores r_m y r_s son necesarios para definir el tipo de robot como $tipo(r_m, r_s)$ [4].

Tipo (3,0)

Estos robots no tienen ruedas fijas ni centradas, únicamente tienen ruedas orientables no centradas o suecas. Estos robots son llamados omnidireccionales debido a que tienen movilidad total en el plano, lo que significa que se pueden mover en cualquier instante de tiempo en cualquier dirección y con cualquier orientación.

Tipo (2,0)

Estos robots no tienen ruedas centradas, tienen una o varias ruedas fijas en el mismo eje.

Tipo (2,1)

Estos robots no tienen ruedas fijas y tienen al menos una rueda centrada.

Tipo (1,1)

Estos robots tienen una o varias ruedas fijas en un único eje. Asimismo, tienen una o varias ruedas centradas con la condición de que el centro de una de ella no esté localizado en el eje de las ruedas fijas.

Tipo (1,2)

Estos robots no tienen ruedas fijas y tienen al menos dos ruedas centradas.

Cabe señalar que el único robot omnidireccional es el tipo *(3,0)*, debido a que su $r_m = 3$, mientras que los otros 4 mencionados tienen un $r_m < 3$.

2.1.4 Configuración seleccionada y postura cinemática

El robot utilizado para la prueba es un robot móvil con configuración diferencial, tipo *(2,0)* el cual no cuenta con ruedas centradas, únicamente con un par de ruedas fijas, alineadas en el mismo eje y una rueda no motorizada que da soporte al sistema.

Si una rueda gira alrededor de su eje *x*, el movimiento sobre su eje *y* es conocido como rodamiento, cualquier componente de movimiento en la dirección del eje *x* perpendicular a la dirección del rodamiento, es conocida como deslizamiento lateral. Una rueda ideal no presenta deslizamiento lateral, solamente se mueve en la dirección del rodamiento. Este modelo de ruedas es el considerado en el modelo cinemático utilizado en este trabajo.

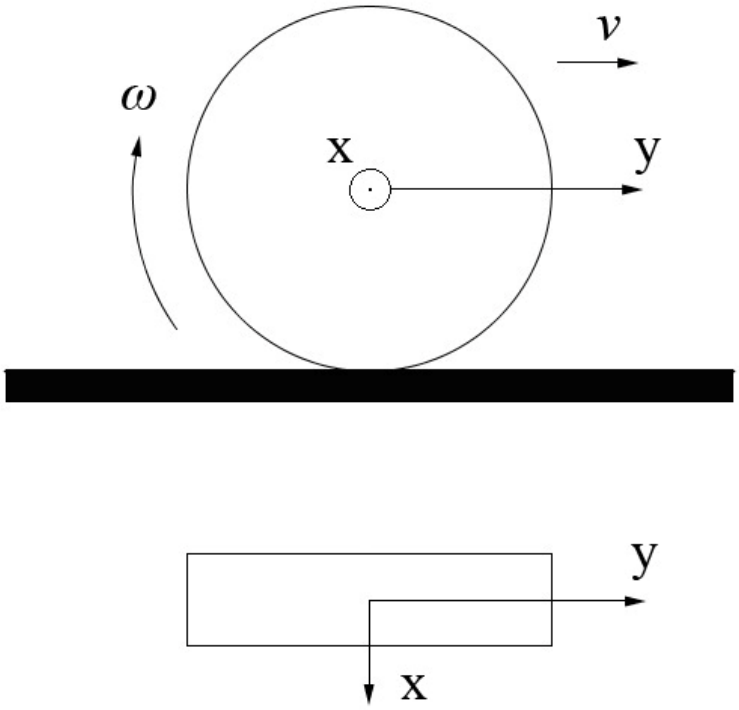


Figura 8 Rueda ideal.

La rueda tiene un radio *r*, una velocidad lineal *v* y una velocidad angular ω.

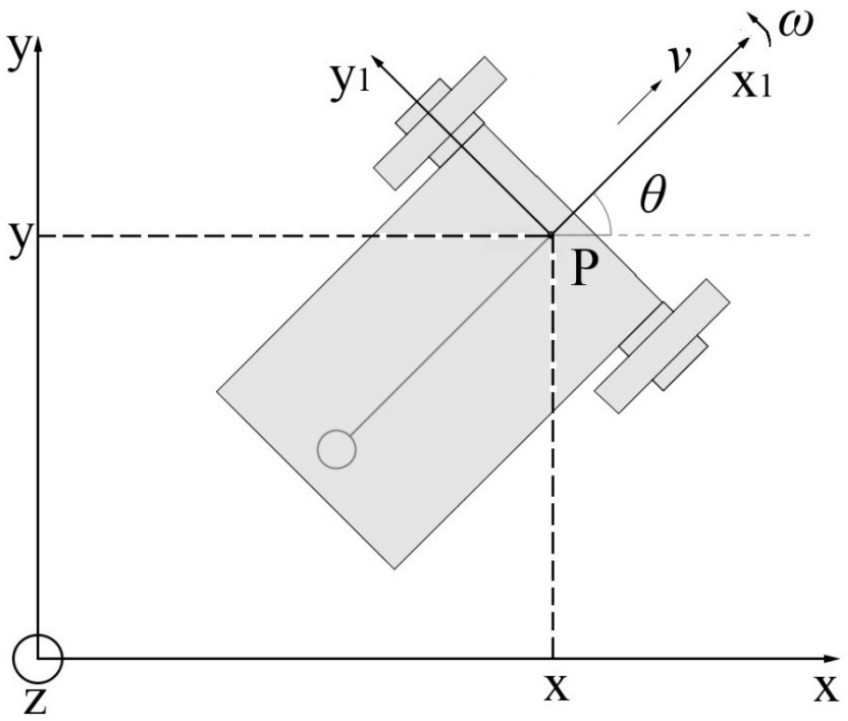


Figura 9 Postura cinemática del robot móvil con configuración diferencial.

Un robot móvil con ruedas tiene dos marcos de referencia que son útiles para su análisis y diseño. El primero es el marco de referencia del mundo real, representado por el sistema coordenado *{x, y}*. Contiene todo lo que está en el ambiente del robot. El segundo es el marco de referencia del robot y se mueve con él mismo, está representado por el sistema coordenado $\{x_1, y_1\}$ adjunto al punto *P* que está fijo a la plataforma del robot y sobre el eje de las ruedas fijas, por tanto, la postura del robot [4] está totalmente descrita por el vector:

$$q = [x \quad y \quad \theta]^T \tag{2}$$

donde q describe la posición del punto $P = (x, y)$ y la orientación θ del marco de referencia del robot $\{x_1, y_1\}$. La orientación θ es el ángulo formado por el eje *x* del marco de referencia del mundo real y el eje x_1 del marco de referencia del robot.

Asimismo, se define la matriz de rotación $R(\theta)$ para mapear el marco de referencia del mundo real en el marco del robot $R(\theta) \in \mathcal{R}^{3x3}$, y viceversa $R^T(\theta) \in \mathcal{R}^{3x3}$ [4].

$$R(\theta) = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$R(\theta) \cdot R^T(\theta) = I \tag{4}$$

En el marco de referencia del robot móvil, el modelo cinemático está definido por:

$$\begin{bmatrix} v_x(t) \\ v_y(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} r/2 & r/2 \\ 0 & 0 \\ -r/L & r/L \end{bmatrix} \begin{bmatrix} \omega_l(t) \\ \omega_r(t) \end{bmatrix} \tag{5}$$

donde $v_x(t)$ es la velocidad en x, $v_y(t)$ es la velocidad en y, $\omega(t)$ es la velocidad angular del robot, r es el radio de las ruedas y L es la distancia entre las mismas, $\omega_l(t)$ es la velocidad angular de la rueda izquierda y $\omega_r(t)$ es la velocidad angular de la rueda derecha. Este modelo es de utilidad para el control de velocidad del robot y resulta evidente que el robot no puede desplazarse en dirección de su eje *y*.

En el marco de referencia del mundo real, las ecuaciones que describen el comportamiento cinemático del robot móvil de configuración diferencial son:

$$\begin{aligned} \dot{x}(t) &= v(t)\cos\,\theta(t) \\ \dot{y}(t) &= v(t)\sin\,\theta(t) \\ \dot{\theta}(t) &= \omega(t) \end{aligned} \tag{6}$$

donde $\dot{x}(t)$es la velocidad sobre el eje *x*, $\dot{y}(t)$ es la velocidad sobre el eje *y*, $v(t)$ es la velocidad lineal del robot, $\theta(t)$ es la posición angular del robot respecto al mundo real y $\omega(t)$ es la velocidad angular del robot.

Integrando la expresión 6 respecto al tiempo, se obtiene el valor de la posición en el eje *x*, la posición en el eje *y* y la posición angular del robot respecto del mundo real.

$$\begin{aligned} x(t) &= \int_0^t v(t)\cos\ \theta(t)dt \\ y(t) &= \int_0^t v(t)\sin\ \theta(t)dt \\ \theta(t) &= \int_0^t \omega(t)dt \end{aligned} \tag{7}$$

Finalmente, se expresa el modelo cinemático del robot en el espacio de estados, de la forma $\dot{q} = S_T(q)u$ [4]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\ \theta & 0 \\ \sin\ \theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{8}$$

donde v y ω son velocidad lineal y angular respectivamente.

2.2 Estabilización alrededor de trayectorias tipo C2

De manera coloquial, el diseño de sistemas de control busca resolver el problema de hacer que un sistema físico se comporte de acuerdo a cierto conjunto de especificaciones. El producto final del diseño de sistemas de control es un dispositivo físico tal que al ser conectado al sistema, garantice que este se va a comportar de acuerdo a las especificaciones.

2.2.1 Ley de control en lazo abierto

La acción de control en lazo abierto es independiente de la salida del sistema, es decir, las entradas al sistema están predefinidas y no toma en cuenta información arrojada por la planta.

Las ecuaciones de movimiento que definen al sistema están descritas por la expresión 8, donde $v\ y\ \omega$ son las variables manipuladas, es decir, son variables que se modifican con el objetivo de afectar las variables controladas que son las que se miden y controlan, representadas por:

$$q = [x \quad y \quad \theta]^T$$

Para una trayectoria de referencia $\big(x_r(t), y_r(t)\big)$ definida en $t \in [0, T]$, se puede obtener una ley de control en lazo abierto. Para ello, se deriva respecto al tiempo, obteniendo un vector de la velocidad en *x* y uno para la velocidad en *y* para cada punto de la trayectoria de referencia. Así pues, la velocidad de referencia del robot para cada punto de la trayectoria es obtenida al calcular el módulo de la suma de ambos vectores.

$$v_r(t) = \pm\sqrt{\dot{x}_r^2(t) + \dot{y}_r^2(t)} \tag{9}$$

El ángulo de la tangente a cada punto de la trayectoria se obtiene mediante:

$$\theta_r(t) = arctan\left(\frac{\dot{y}_r(t)}{\dot{x}_r(t)}\right) \tag{10}$$

Debido a la naturaleza de la función arco tangente, es necesario implementar una función de transferencia que mapee los valores entregados por la función trigonométrica inversa a un valor que vaya de *0* a *2nπ*, donde *n* es el número de vueltas que la trayectoria ha dado en sentido anti horario.

Finalmente, para obtener la velocidad angular, se deriva la posición angular respecto al tiempo para cada punto de la trayectoria.

$$\omega_r(t) = \frac{d}{dt}\theta_r(t) = \frac{\dot{x}_r(t)\ddot{y}_r(t) - \dot{y}_r(t)\ddot{x}_r(t)}{\dot{x}_r^2(t) + \dot{y}_r^2(t)} \tag{11}$$

Con las expresiones 9, 10 y 11, se define la trayectoria de referencia $q_r(t) = [x_r\,(t), y_r\,(t), \theta_r\,(t)]^T$ y las entradas al robot virtual $v_r(t)$ y $\omega_r(t)$. Para esta ley de control es requisito que la trayectoria sea de tipo C2, es decir, que sus segundas derivadas sean continuas, asimismo, el error inicial debe ser cero y no deben existir perturbaciones.

2.2.2 Aproximación lineal y análisis de controlabilidad

Para muchos sistemas no lineales, la aproximación lineal se puede usar como un buen método de control por su simplicidad, asimismo, aporta información sobre la controlabilidad en lazo cerrado del sistema no lineal, es decir, si la aproximación lineal es controlable, el sistema no lineal original también lo es, al menos para una vecindad del punto de operación [3].

El objetivo es pasar de un sistema de la forma $\dot{q} = f(q, u)$, donde q es el vector de variables de estado del sistema y el vector u son las entradas al sistema a una representación en espacio de estados de la forma $\dot{q} = Aq + Bu$ equivalente, para un punto de operación o vecindad del mismo. Para linealizar el sistema, se debe obtener el jacobiano del sistema no lineal evaluado en el punto de operación.

$$J_{(f,q)} = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \cdots & \frac{\partial f_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial q_1} & \cdots & \frac{\partial f_m}{\partial q_n} \end{bmatrix}_{PO}$$

$$J_{(f,u)} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_j} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial u_1} & \cdots & \frac{\partial f_m}{\partial u_j} \end{bmatrix}_{PO} \tag{12}$$

Para el caso de estudio, la aproximación lineal del sistema suya expresión es 8, en el punto de equilibrio ($\vec{q} = 0, \vec{u} = 0$) esta dado por:

$$\dot{q} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} u \tag{13}$$

Análisis de Controlabilidad

Sea la ecuación de estados $\dot{q} = Aq + Bu$, se dice que el par *(A,B)* es CONTROLABLE si para cualquier q_0 y cualquier estado final $q(t_1)$ existe $u(t)$ tal que transfiera al sistema de q_0 a $q(t_1)$ en un tiempo finito t_1. De otra forma, se dice que el sistema o el par *(A, B)* es NO CONTROLABLE [7].

Teorema [7]

El par *(A, B)* es controlable si la matriz de controlabilidad *C* es de rango completo [7].

$$C = [B \;\; A \cdot B \ldots A^{n-1} \cdot B] \tag{14}$$

Para la aproximación lineal, la matriz de controlabilidad *C* está definida por:

$$C = B = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{15}$$

El rango de esta matriz es dos, por tanto, información del modelo no lineal se pierde al utilizar una aproximación lineal del sistema.

2.2.3 Estabilización utilizando como guía un robot virtual

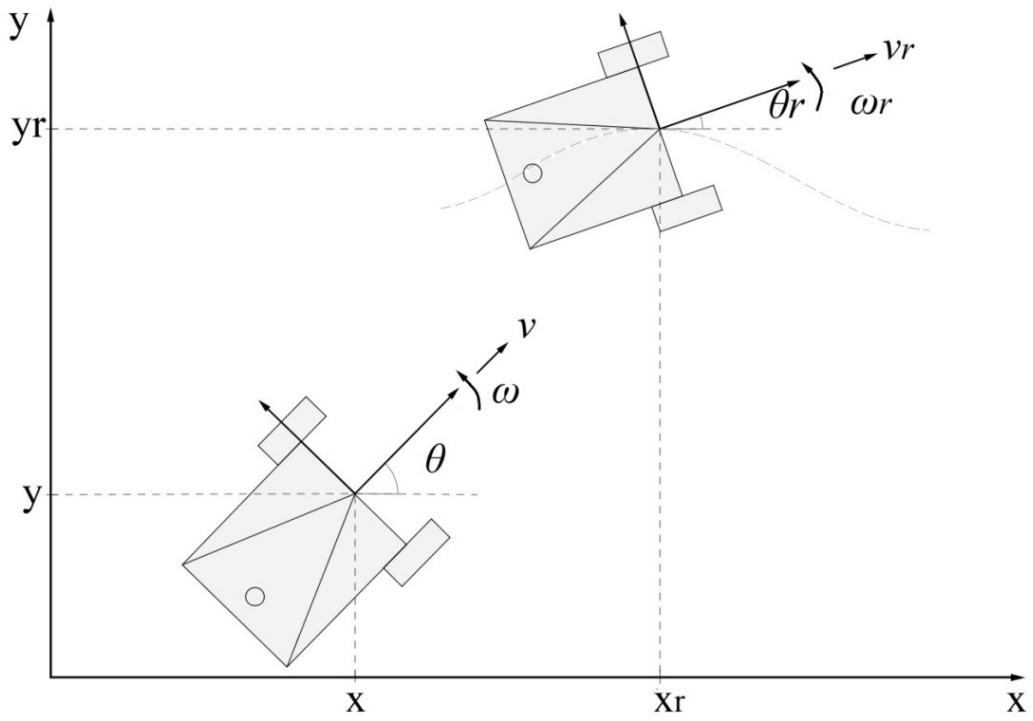


Figura 10 El robot real y el robot virtual.

Se considera un robot virtual de referencia, cuyas ecuaciones son:

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} cos\theta_r & 0 \\ sin\theta_r & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ \omega_r \end{bmatrix} \tag{16}$$

En la Figura 10 el robot de referencia es un robot imaginario que idealmente sigue la trayectoria de referencia. Sin embargo, el robot real al compararlo con el robot virtual, tiene cierto error al seguir la trayectoria, así pues, la ley de control debe ser diseñada para forzar al robot real a seguir la trayectoria de referencia de manera precisa [6].

Se busca una ley de control

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = k(q, q_r, v_r, \omega_r)$$

tal que

$$\lim_{t\to\infty}(q_r(t) - q(t)) = 0 \tag{17}$$

Así pues, el problema de seguimiento de trayectoria involucra ecuaciones de error que describen la evolución de $q_r - q$ en el tiempo; dicho error, expresado en el marco de referencia del robot real, está descrito por:

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix} \tag{18}$$

En [3] se propone el siguiente cambio de variable en las entradas del sistema:

$$\begin{aligned} u_1 &= v_r \cos e_3 - v \\ u_2 &= \omega_r - \omega \end{aligned} \tag{19}$$

Derivando la expresión 18 respecto al tiempo y con el cambio de entradas de 19, se obtiene el modelo cinemático no lineal asociado al error de seguimiento [3]:

$$\dot{e} = \begin{bmatrix} 0 & \omega & 0 \\ -\omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} e + \begin{bmatrix} 0 \\ \sin e_3 \\ 0 \end{bmatrix} v_r + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{20}$$

donde v y ω incluidas en las expresiones 19 y 20 son las entradas en lazo cerrado.

2.2.3.1 Linealización

Debido a que es de interés que el error e sea cero o cercano a cero, se linealiza el modelo cinemático no lineal asociado al error (20) alrededor del punto de operación $(e = 0, u = 0)$, tal que se obtiene el modelo linealizado de la forma $\dot{e} = Ae + Bu$:

$$J_{(f,e)} = \begin{bmatrix} 0 & \omega_r & 0 \\ -\omega_r & 0 & v_r \\ 0 & 0 & 0 \end{bmatrix}; \quad J_{(f,u)} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$
$$\therefore \ \dot{e} = \begin{bmatrix} 0 & \omega_r & 0 \\ -\omega_r & 0 & v_r \\ 0 & 0 & 0 \end{bmatrix} e + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{21}$$

donde v_r y ω_r son las entradas en lazo abierto, mientras que v y ω son las entradas en lazo cerrado.

2.2.3.2 Análisis de controlabilidad

Según [7], para el sistema de la expresión 21 que tiene 3 variables controladas y, por tanto, $n = 3$, la matriz de controlabilidad *C* es:

$$C = [B \quad A \cdot B \quad A^2 \cdot B] = \begin{bmatrix} 1 & 0 & 0 & 0 & -\omega_r^2 & v_r\omega_r \\ 0 & 0 & -\omega_r & v_r & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{22}$$

Para que la matriz *C* sea de rango completo y para que el sistema sea controlable, v_r y ω_r deben ser diferentes de cero. Este es otro motivo por el que la trayectoria de referencia q_r debe ser de tipo C2 [6].

2.2.3.3 Control lineal por realimentación de estados

El sistema descrito por la expresión 21 tiene dos entradas, que son la velocidad lineal *v* y la velocidad angular *ω*, tiene tres variables controladas que son los errores asociados a

la postura del robot, así pues, un controlador lineal por realimentación de estados para dicho sistema esta descrito por [7]:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = -ke, \;\; k \epsilon R^{2x3} \tag{23}$$

Para reducir el error en la dirección de la velocidad del robot, e_1, dicha velocidad debe ser modificada. De manera similar, el error de orientación, e_3, puede ser manipulado eficientemente con la velocidad angular del robot. Finalmente, el error ortogonal a la velocidad del robot puede ser reducido modificando la velocidad angular y la velocidad lineal del robot. En [3] se propone el siguiente controlador lineal por realimentación de estados:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = -\begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & k_3 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \tag{24}$$

Para determinar el valor de las ganancias se utiliza la técnica de asignación de polos, los polos deseados están definidos por el polinomio característico deseado que, para sistemas de tercer orden es:

$$\begin{gathered} P_d = (s + 2\xi\omega_n)(s^2 + 2\xi\omega_n s + \omega_n^2) \\ P_d = s^3 + 4\xi\omega_n s^2 + (\omega_n^2 + 4\xi^2\omega_n^2)s + 2\xi\omega_n^3 \end{gathered} \tag{25}$$

donde ξ es el factor de amortiguamiento y ω_n es la frecuencia natural del sistema.

El polinomio característico del sistema en lazo cerrado con el controlador de la expresión 24 está descrito por:

$$P_c = det\big(sI - (A - Bk)\big)$$

$$P_c = det\left(\begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & s \end{bmatrix} - \left(\begin{bmatrix} 0 & \omega_r & 0 \\ -\omega_r & 0 & v_r \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & k_3 \end{bmatrix}\right)\right)$$

$$P_c = s^3 + (k_1 + k_3)s^2 + (k_1k_3 + \omega_r^2 + k_2v_r)s + k_3\omega_r^2 + k_1k_2v_r \tag{26}$$

Igualando términos de las expresiones 25 y 26:

$$\begin{aligned} k_1 + k_3 &= 4\xi\omega_n \\ k_1k_3 + \omega_r^2 + k_2v_r &= \omega_n^2 + 4\xi^2\omega_n^2 \\ k_3\omega_r^2 + k_1k_2v_r &= 2\xi\omega_n^3 \end{aligned} \tag{27}$$

Resolviendo la expresión 27 se obtiene el valor de las ganancias en términos del factor de amortiguamiento ξ y de la frecuencia natural ω_n:

$$\begin{aligned} k_1 = k_3 &= 2\xi\omega_n \\ k_2 &= \frac{\omega_n^2 - \omega_r^2}{v_r} \end{aligned} \tag{28}$$

El valor de k_2 depende de la dirección de v_r, por tanto, se puede reescribir k_2 como:

$$k_2 = \frac{\omega_n^2 - \omega_r^2}{|v_r|} \tag{29}$$

Considerando la expresión 29, el controlador lineal descrito por la ecuación 24 para el sistema cinemático linealizado asociado al error queda definido por [3]:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} -k_1 & 0 & 0 \\ 0 & -sign(v_r)k_2 & -k_3 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \tag{30}$$

Finalmente, tomando en cuenta las expresiones 8,16,18,19,28,29 y 30, se obtiene una ley de control no lineal variante con el tiempo, que permite modificar las entradas con objeto de estabilizar al robot móvil alrededor de una trayectoria tipo C2. Dicha ley de control es:

$$\begin{aligned} v(t) = {} & v_r(t)\cos\big(\theta_r(t)-\theta(t)\big) - 2\xi\omega_n\cos\big(\theta(t)\big)\big(x_r(t)-x(t)\big) \\ & - 2\xi\omega_n\sin\big(\theta(t)\big)\,(y_r(t)-y(t)) \\ \omega(t) = {} & \omega_r(t) - sign\big(v_r(t)\big)\frac{\omega_n^2-\omega_r^2(t)}{|v_r(t)|}\cos\big(\theta(t)\big)\,\big(y_r(t)-y(t)\big) \\ & + sign\big(v_r(t)\big)\frac{\omega_n^2-\omega_r^2(t)}{|v_r(t)|}\sin\big(\theta(t)\big)\,\big(x_r(t)-x(t)\big) \\ & - 2\xi\omega_n\big(\theta_r(t)-\theta(t)\big). \end{aligned} \tag{31}$$

Capítulo 3

Implementación del sistema

3.1 Robot móvil con configuración diferencial

3.1.1 Descripción general

Se utilizó un robot móvil con configuración diferencial tipo (*2,0)*, el cual cuenta con un eje delantero en cuyos extremos se encuentran dos ruedas fijas actuadas, con una tercera rueda orientable no actuada centrada en la parte trasera que le proporciona estabilidad. Tiene base de aluminio, sobre la cual se montaron dos motores de CD EMG30 con el eje en posición vertical, el par es transmitido mediante engranes de los motores a las ruedas.

El robot cuenta con dos pisos de acrílico sobre la base metálica, soportados por pilares metálicos. Entre la base y el primer piso de acrílico se incorporó una batería de 12 V marca STEREN modelo br1204, que se encarga de proveer la energía a los motores y a la electrónica. En el piso intermedio se colocó la tarjeta MD25 cuya función es la de controlar los motores, asegurando que la velocidad de los mismos sea la solicitada por el controlador.

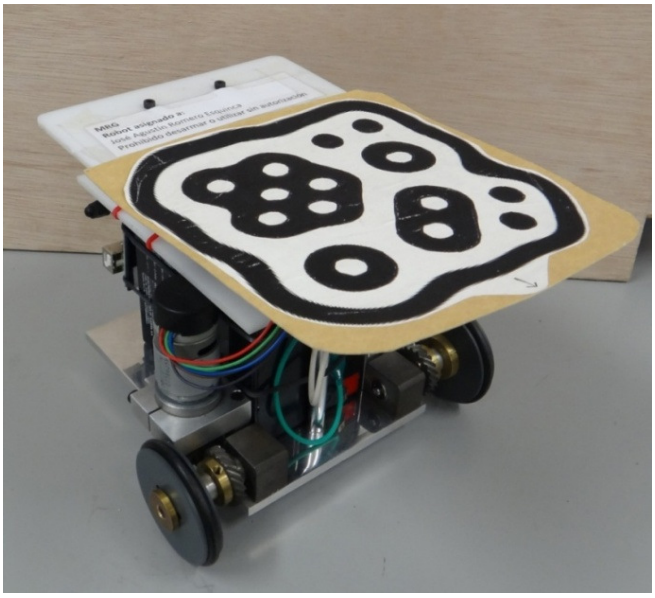


Figura 11 Robot móvil utilizado.

Finalmente, anclado a los pilares metálicos se encuentra el micro-controlador Arduino Duemilanove y el receptor inalámbrico, cuyas funciones son recibir datos de forma inalámbrica provenientes del controlador y enviarlos a la tarjeta MD25 mediante comunicación I^2C para hacer funcionar al sistema. Las especificaciones del robot son mostradas en la Tabla 2.

Tabla 2 Especificaciones del robot móvil.

Motores	2 EMG30 de 12 V de CD
Controlador de motores	MD25
Tarjeta controladora	Arduino Duemilanove
Módulo de comunicación inalámbrica	Radio Xbee
Largo	15 cm
Ancho	11 cm
Alto	15 cm
Distancia entre ruedas	17 cm
Radio de las ruedas	2.95 cm

3.1.2 Motores EMG30

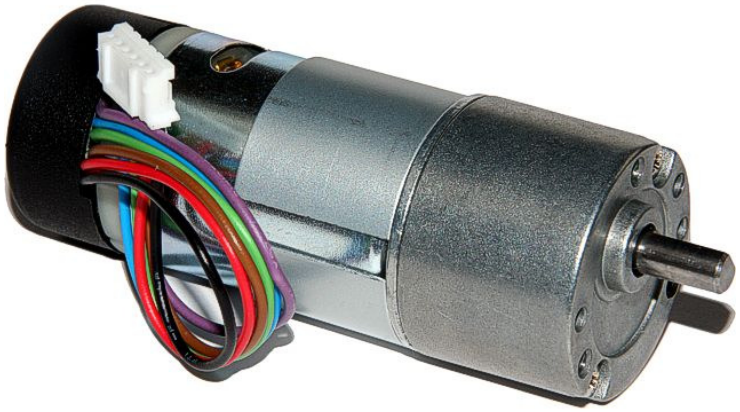


Figura 12 Motor EMG30.

Se aplicó par a las ruedas utilizando motores de 12 V de CD modelo EMG-30, con una reducción de 30:1; estos motores cuentan con un encoder de efecto Hall integrado. Sus especificaciones nominales están dadas en la tabla 3 [8].

Tabla 3 Especificaciones del motor EMG30.

Voltaje nominal	12 V
Par nominal	1.5 $kg_f \cdot cm$
Velocidad nominal	170 rpm
Corriente nominal	530 mA
Velocidad sin carga	216 rpm
Corriente sin carga	150 mA
Corriente con motor bloqueado	2.5 A
Potencia nominal	4.22 W
Pulsos por vuelta del encoder	360

3.1.3 Controlador de motores MD25

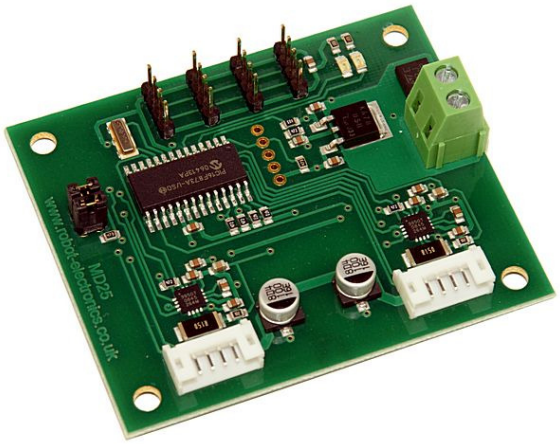


Figura 13 Controlador para motores MD25.

Esta tarjeta es desarrollada por Devantech, LTd. Es un controlador de doble puente H para motores EMG30. Sus principales características son [9]:

- Entradas para lectura de encoders acoplados a los motores EMG30.
- Lectura de los encoders, la cual proporciona datos sobre el desplazamiento producido por los motores y su sentido de giro.
- Puede controlar dos motores de forma independiente.
- Se alimenta con un rango de voltaje de 9 a 14 V.
- Regula 5 V a 300 mA para alimentación de circuitería auxiliar.
- Suministra hasta 3 A a cada motor.
- Comunicación serie o I^2C.

La comunicación entre este controlador y el micro-controlador fue mediante I^2C, debido a la facilidad que representa implementar dicha comunicación con Arduino y debido a que éste solamente cuenta con un módulo pre-establecido para comunicación serial, el cual está asignado a la recepción de datos mediante radiofrecuencia.

3.1.4 Tarjeta controladora

La tarjeta controladora que se utilizó en el prototipo funcional fue un Arduino Duemilanove [10]. Arduino es una plataforma de fuente abierta (open source) para la creación de prototipos basada en software y hardware flexibles y fáciles de usar.

Arduino puede tomar información del entorno a través de sus pines de entrada analógicos de toda una gama de sensores y puede interactuar con el medio que le rodea, controlando luces, motores, servomotores y otros actuadores, así como comunicándose de forma alámbrica e inalámbrica con otros sistemas.

El micro-controlador de la placa Arduino se programa mediante el lenguaje de programación C, y el entorno de desarrollo Arduino está basado en Processing. La tarjeta Arduino Duemilanove, el cual salió al mercado en 2009, es una placa con el micro-controlador ATmega328P, tiene 14 pines con entradas/salidas digitales, seis de las cuales pueden ser usadas como salidas PWM, seis entradas analógicas, un cristal oscilador a 16 MHz, conexión USB, entrada de alimentación, una cabecera ISCP y un botón de reset [10].

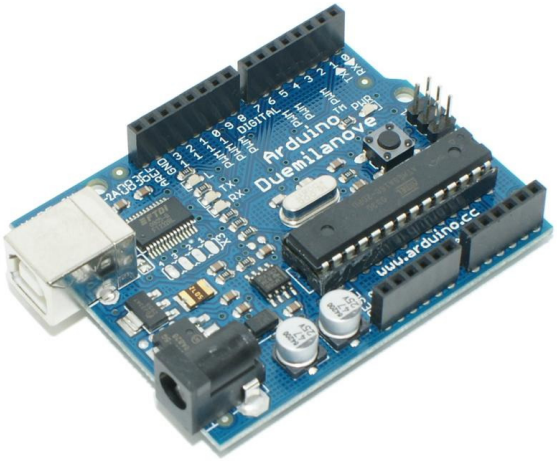


Figura 14 Tarjeta Arduino Duemilanove.

El ATmega328P tiene 32 kB de memoria flash para almacenar código, 2 kB de los cuales son usados para el arranque del sistema, también conocido como bootloader. El ATmega328 tiene 2 kB de memoria SRAM y 1 kB de EEPROM [11].

El Arduino Duemilanove facilita en varios aspectos la comunicación con la computadora, ya que proporciona comunicación serie UART TTL (5 V) disponible a través de los pines digitales 0(RX) y 1(TX). Un chip FTDI FT232RL integrado en la placa canaliza esta comunicación a través del puerto USB. Asimismo, soporta la comunicación I^2C y SPI. El software Arduino incluye un archivo de biblioteca Wire para simplificar el uso del bus I^2C. Las especificaciones están resumidas en la Tabla 4.

Tabla 4 Especificaciones de la tarjeta Arduino Duemilanove

Micro-controlador	ATmega328P
Voltaje de funcionamiento	5 V
Voltaje de entrada	7-12 V
Pines de entrada/salida	14
Pines PWM	6
Pines de entrada analógicos	6
Intensidad de corriente máxima por pin	40 mA
Memoria flash	32 kB
SRAM	2 kB
EEPROM	1 kB
Frecuencia de operación	16 MHz

3.1.5 Fuente de alimentación

La fuente de alimentación instalada en el robot móvil fue una batería recargable sellada, de ácido-plomo, marca STEREN, modelo br1204, de 4 A·h, sus especificaciones se muestran en la Tabla 5.

Figura 15 Batería BR1204.

Tabla 5 Especificaciones de batería br1204.

Voltaje	12 V
Dimensiones	9x10.1x7 cm
Temperatura máxima	300°C
Peso	1.7 kg_f
Efecto de memoria	No
Protección para carga excesiva	Si

3.2 Sistema de visión

3.2.1 Descripción general

El prototipo implementado utiliza un sistema de visión, el cual cumple el rol de sensor, es decir, es el dispositivo o sistema que aporta información sobre las variables de interés en tiempo real, en este caso, la localización del robot móvil en el espacio de trabajo y permite reingresar dichos datos al controlador para que éste haga las correcciones pertinentes y el robot siga la trayectoria deseada.

Una cámara capta, desde una perspectiva aérea, al robot en el espacio de trabajo, el campo de visión de la cámara delimita el espacio en el que el robot pude moverse y el espacio en que es posible obtener información de su localización. Una vez captada, los datos son procesados y enviados a Simulink donde los cálculos del control son llevados a cabo.

3.2.2 Arquitectura

El sistema de visión se puede descomponer en subsistemas, los cuales son: cámara, software de identificación de imagen ReacTIVision y cliente TUIO.

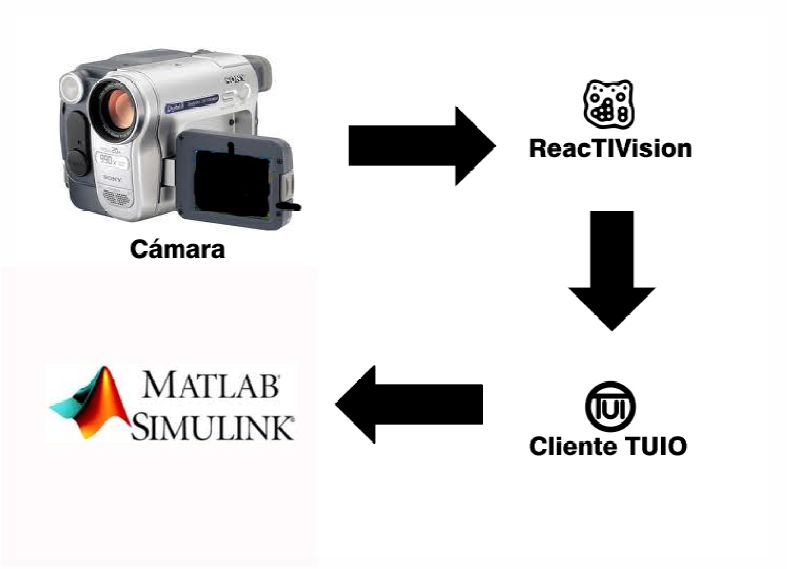


Figura 16 Arquitectura del sistema de visión.

La cámara obtiene una fotografía, la cual es procesada por un software de identificación de imagen llamado ReacTIVision, este software, mediante un cliente TUIO, es capaz de enviar la información captada a Simulink® de MATLAB®, que es el software utilizado para llevar al cabo las tareas de procesamiento y cálculos de los controladores.

3.2.3 Cámara

Figura 17 Cámara Sony DCR-TRV316.

La cámara para la captación de imágenes en tiempo real utilizada fue una Sony Handycam DCR-TRV361, la cual se conectó mediante un cable *firewire* a la

computadora; cabe señalar que esta conexión permitió una mayor velocidad en la transferencia de datos con respecto a tecnologías más populares como USB y USB2.0.

3.2.4 Software ReacTIVision

ReacTIVision es un programa de código abierto y multiplataforma, diseñado para el rastreo rápido y robusto de objetos físicos mediante símbolos fiduciales, los cuales son adheridos a dichos objetos.

ReacTIVision rastrea los símbolos fiduciales diseñados de forma especial, mediante video en tiempo real. La imagen fuente es convertida a una imagen en blanco y negro vía un algoritmo de umbral adaptativo. Después, esta imagen es segmentada en un árbol de regiones negras y blancas, lo que genera una representación llamada gráfica de región adyacente. En este esquema se buscan secuencias específicas que son codificadas en un marcador *fiducial*. Por último, estas secuencias se relacionan mediante un catálogo de símbolos para obtener un número de identificación único. Esta tecnología permite la identificación del centro del marcador y su orientación [12].

Una vez identificado el marcador, ReacTIVision envía mensajes TUIO a cualquier cliente mediante el puerto UDP 333.

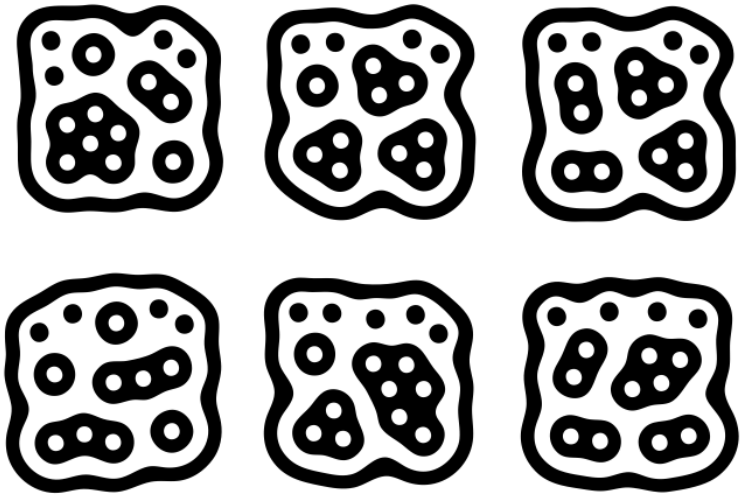


Figura 18 Ejemplo de marcadores fiduciales.

3.2.5 Cliente TUIO

ReacTIVision envía mensajes TUIO a cualquier aplicación o programa que cuente con un cliente TUIO, el cual es un marco abierto que define un protocolo común para superficies *multitouch* y aplicaciones de visión por computadora. Este protocolo codifica datos de control provenientes de aplicaciones, como ReacTIVision, y los envía a otra aplicación, con su respectivo cliente TUIO, que es capaz de decodificar dicho protocolo y acceder a la información [13].

Existe un creciente número de aplicaciones compatibles con el protocolo TUIO así como archivos de biblioteca para diversos ambientes de programación que han acelerado el desarrollo de superficies *multitouch* y software de visión [13].

3.3 Acoplamiento y procesamiento de datos

3.3.1 Simulink® de MATLAB®

Es un entorno de programación visual, que funciona sobre el software de programación MATLAB. Es un ambiente de programación de más alto nivel de abstracción que el lenguaje interpretado MATLAB. Simulink genera archivos con extensión .mdl, haciendo referencia a modelos. Simulink es una herramienta de simulación de modelos o sistemas, que provee un editor gráfico, bibliotecas de bloques personalizables y solucionadores para la modelación y simulación de sistemas dinámicos. Se emplea en ingeniería mecatrónica en temas relacionados con el procesamiento digital de señales, telecomunicaciones, robótica y control [14].

3.3.2 Funciones s

Las funciones s permiten agregar algoritmos propios a modelos de Simulink. Los algoritmos pueden ser escritos en MATLAB o en C. Siguiendo una serie de reglas simples, es posible implementar estos algoritmos en una función s. Una función s es la descripción de un sistema dinámico en términos de un lenguaje de programación. Estas funciones utilizan una sintaxis especial que le permite interactuar con los

solucionadores de ecuaciones de Simulink. Esta interacción es muy similar a la que ocurre entre los solucionadores y los bloques integrados en Simulink [15].

La forma de una función s es muy general y permite interactuar con sistemas continuos, discretos e híbridos. Como resultado, casi todos los modelos de Simulink pueden describirse como funciones s. Las funciones s se incorporan a Simulink haciendo uso del bloque *s-function* ubicado en la biblioteca *Nonlinear Block.* Se debe utilizar el cuadro de diálogo de dicho bloque para especificar el nombre de la función s asociada al mismo, como se muestra en la Figura 19.

El uso más común que se le da a las funciones s es crear bloques personalizados que permitan implementar código C ya existente a la simulación, describir el sistema como un conjunto de ecuaciones matemáticas e incorporar gráficas y animaciones.

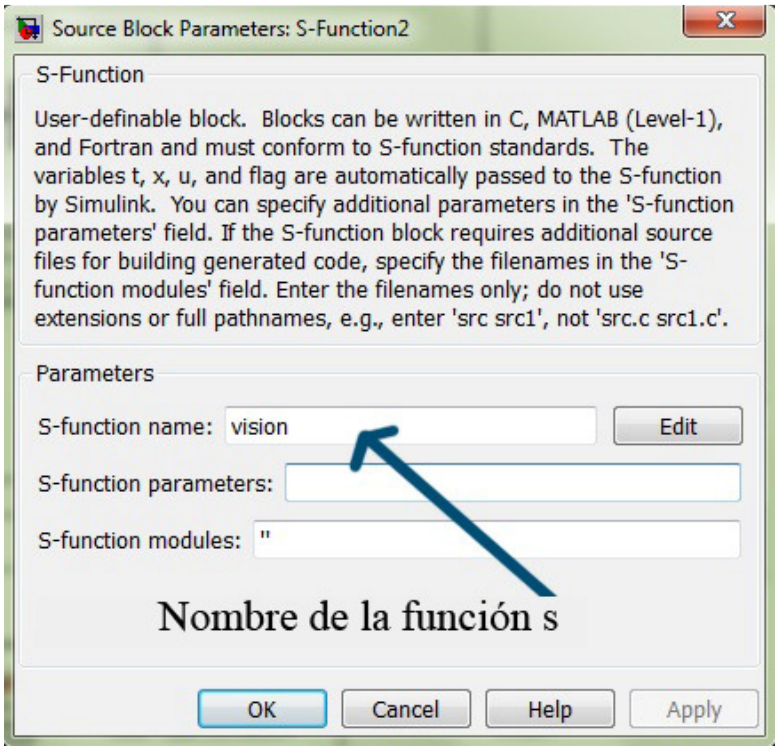


Figura 19 Ventana de parámetros de configuración de una función s.

3.3.3 Cliente TUIO y su integración con Simulink® de MATLAB®

Como ya se mencionó, el software de visión ReacTIVision envía mensajes TUIO, por tanto, fue necesario instalar dicho cliente a Simulink para que pudiera decodificar dichos mensajes e hiciera uso de la información encriptada. Para instalar el cliente TUIO se recomienda seguir las siguientes instrucciones:

1 Seleccionar el lenguaje de programación del cliente TUIO; existe para C#, C++, Java, Processing, entre otros. Para este experimento, se utilizó Java.
2 Descargar el archivo TUIO_JAVA.zip de la página de ReacTIVision [12] y extraerlo a una carpeta temporal.
3 Copiar los archivos TuioDemo.jar y libTUIO.jar de la carpeta temporal a la carpeta MATLAB\versión\java\jar.
4 Ejecutar MATLAB como administrador y, en la ventana de comandos, escribir: "*edit classpath.txt*" y presionar la tecla *enter*. Se abrirá un editor con direcciones a diversos archivos .jar, demos agregar las siguientes líneas: $matlabroot/java/jar/TuioDemo.jar y $matlab/java/jar/libTUIO.jar.
5 Guardar los cambios en *classpath.txt*.

Ahora, mediante una función s, se pudieron importar los datos enviados por ReacTIVision. Cabe señalar que en dicha función s se debe agregar la biblioteca TUIO, luego crear un objeto y finalmente establecer la comunicación entre el objeto creado y la función s; todo esto está integrado en la función de visión, la cual puede ser consultada en el Apéndice A1. Las instrucciones que permitieron hacerlo son las siguientes:

- import TUIO.*;
- tc=TuioCliente();
- tc.conect();

Con el objeto creado, fue posible obtener diferentes datos de uno o varios fiduciales como su posición en *x*, posición en *y*, orientación, velocidad, etc. Para este experimento solamente se necesitó conocer tres datos de un único fiducial, los cuales son, posición en *x*, posición en *y* y su orientación.

Las instrucciones que se utilizaron para obtener dicha información dentro de la función s fueron las siguientes:

- x1=tc.getTuioObjects().get(0).getPosition().getX();
- y1=tc.getTuioObjects().get(0).getPosition().getY();
- angulo=tc.getTuioObjects().get(0).getAngle();

3.3.4 Funcionamiento en tiempo real

Como se verá más adelante, las ecuaciones que describen las trayectorias de referencia para el robot móvil son funciones del tiempo; para efectos de simulación este hecho pierde importancia debido a que el objetivo de las mismas es hacer pruebas para diferentes trayectorias y parámetros del sistema en menos tiempo, y debido a que Simulink tiene su propio reloj interno que alimenta las diferentes etapas del experimento, es decir, el proceso de datos, la generación de señales y la simulación sobre el robot, se ejecutan en la misma plataforma.

Al llevar a cabo pruebas con el robot real, es de suma importancia que el tiempo de simulación coincida con el tiempo real. Este problema es relativamente fácil de resolver; únicamente se debe hacer más lento el tiempo de simulación y así sincronizar los cálculos hechos con Simulink y el robot móvil real.

Se utilizó el bloque *Real Time Pacer* para cumplir con esta tarea, el cual está disponible para descarga gratuita en la red. Este bloque permite tener una relación constante entre el tiempo de simulación y el tiempo real. Para este experimento, dicha relación es de 1.

El bloque está implementado con una función s, escrita en código m, lo que permite que funcione en cualquier plataforma de Simulink; asimismo, el bloque utiliza la función pausa de MATLAB para hacer más lenta la simulación; este comando libera al procesador durante la pausa, por tanto, el rendimiento no decae [16].

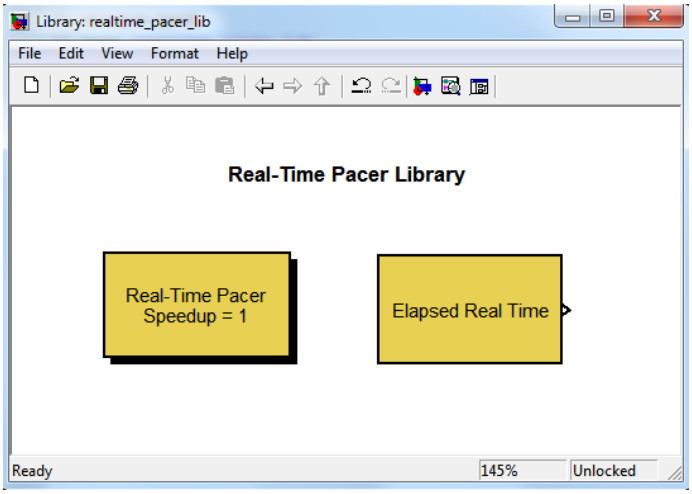


Figura 20 Biblioteca Real-Time Pacer.

3.3.5 Acoplamiento de datos

Los datos obtenidos del sistema de visión a través del cliente TUIO, así como las entradas calculadas por el controlador que se inyectan al robot móvil no pueden ser utilizadas directamente, se debe acoplar dicha información al sistema para que ésta sea de utilidad, tanto en el proceso de lectura de estados como en el de envío de datos al robot.

Los datos provenientes del sistema de visión están graduados, para establecer la posición en el eje x y en el eje y de 0 a 1. Esto se soluciona de manera sencilla, solamente hace falta implementar una función que mapee los valores entregados por el sistema de visión, que van de 0 a 1, a valores de longitud correspondientes al espacio de trabajo. El ángulo entregado por ReacTIVision es de 0 a 2π y el controlador necesita el valor del ángulo de $-2n\pi$ a $2n\pi$ y las derivadas respecto al tiempo de las ecuaciones que describen el comportamiento en el eje x y en el eje y tanto de la trayectoria del robot como de la trayectoria de referencia, el código que corrige este problema se presenta en el Apéndice 1, en el correspondiente a la función de visión.

Asimismo, el controlador entrega dos señales que se introducen al robot, éstas son la velocidad lineal y la velocidad angular. La manera de manipular el movimiento del

robot es a través de la velocidad de giro de cada una de sus ruedas actuadas, por tanto, se necesita que las entradas se traduzcan a velocidad angular de cada una de sus ruedas.

La relación entre velocidad lineal, velocidad angular del robót y las velocidades angulares en las ruedas está expresada en las siguientes ecuaciones:

$$\omega_l = \frac{v - \frac{L}{2}\omega}{r}$$
$$\omega_r = \frac{v + \frac{L}{2}\omega}{r} \tag{32}$$

donde r es el radio de las ruedas, L es la distancia entre las ruedas, v es la velocidad lineal, ω es la velocidad angular, ω_l y ω_r son la velocidad angular de la rueda izquierda y derecha, respectivamente.

Finalmente, se requiere transformar la velocidad angular de cada rueda en una señal que el controlador de motores MD25 pueda entender y por tanto, actúe los motores correctamente.

La tarjeta MD25 puede leer una señal de 8 bits para cada motor, de manera que, 0 corresponde a la máxima velocidad de los motores en reversa, 128 detiene los motores y 255 pondrá los motores a girar a su máxima velocidad hacia adelante [9].

La relación entre velocidad angular en las ruedas y la señal de 8 bits enviada al driver es la siguiente:

$$señal\ de\ 8\ bits = 6.4081 * velocidad\ angular\ + 127.67 \tag{33}$$

3.3.6 Diagramas implementados en Simulink® de MATLAB®

Se utilizaron tres diagramas en Simulink para llevar a cabo simulaciones y experimentos con el robot real. El diagrama empleado para las pruebas con el robot real consta de los siguientes bloques:

- Generador de trayectorias, donde x y y están definidos en función del tiempo, es decir, x=f(t) y y=f(t)
- Bloque *Real Time Pacer* para coordinar el tiempo de simulación con el tiempo real
- Bloque de entrada de datos proveniente de la visión
- Bloques que integran el controlador
- Bloques de acondicionamiento de señal
- Bloques de envío de datos al robot.

El diagrama usado en simulaciones no tiene bloque de visión, acondicionamiento de datos ni envío de los mismos, pero incorporó un bloque que simula al robot, el cual entrega las señales que realimentan al sistema de control. Tampoco incorporó el bloque *Real Time Pacer* debido a que no interesa hacer simulaciones en tiempo real, sino hacer simulaciones más rápido para ahorrar tiempo. Los tres diagramas incluyeron gráficas para el análisis de los datos arrojados tanto en simulaciones como en experimentos con el robot real. Los diagramas pueden ser consultados en el Apéndice 1.

3.4 Comunicación inalámbrica

Una computadora personal con Simulink instalado fue la encargada de recibir la información proveniente del módulo de visión, procesar los datos y enviarlos al robot; debido a la naturaleza de los robots móviles, es poco práctico establecer una comunicación alámbrica, por tanto, fue necesario implementar una comunicación inalámbrica entre dicha computadora y el robot. Simulink es capaz de enviar datos vía comunicación serial de manera sencilla con un bloque denominado *serial send*, y mediante un módulo de radiofrecuencia, se envió dicha información a un receptor instalado en el robot, el cual recibe la cadena de datos y los envía mediante comunicación serial al micro-controlador. Los radios utilizados en este experimento fueron un par de Xbee Series 2.

3.4.1 Radios XBee Series 2

Los módulos de radiofrecuencia XBee Series 2 fueron diseñados para operar bajo el protocolo de comunicación ZigBee y para cumplir con los estándares de bajo costo y

baja potencia de las redes inalámbricas. Estos módulos requieren poca energía y ofrecen una entrega de datos confiable y robusta entre dispositivos remotos; sus características principales se muestran en la Tabla 6 [17].

Tabla 6 Especificaciones de la radio XBee Series 2.

Rango en interiores	40 m
Rango en exteriores, en línea de visión	120 m
Velocidad de transferencia	250 kbps
Voltaje de operación	2.8 – 3.4 V
Intensidad de corriente de operación	40 mA
Dimensiones	2.438 x 2.761 cm
Banda de frecuencia de operación	ISM 2.4 GHz

Un módulo opera como emisor, conectado a la computadora, y envía los datos de salida obtenidos por el controlador programado en Simulink, un segundo módulo se conecta a la tarjeta Arduino Duemilanove y recibe información, la cual es enviada al controlador de los motores mediante el protocolo I^2C.

3.4.2 Protocolo ZigBee

ZigBee es una tecnología desarrollada como un estándar global abierto para hacer frente a las necesidades únicas de las redes inalámbricas de bajo costo y bajo consumo de energía. ZigBee opera con las especificaciones de radio 802.15.4 IEEE y en bandas de frecuencia sin licencia, incluyendo 2.4 GHz, 900 MHz y 868 MHz.

Las especificaciones 802.15.4 fueron ratificadas por el *Institute of Electrical and Electronics Engineers* (IEEE) en 2003. Este protocolo permite a los dispositivos comunicarse en una variedad de topologías de red como son punto a punto, punto a puntos múltiples o redes de malla. Debido a su bajo ciclo de trabajo, proporciona una larga vida a las baterías que lo alimentan [17].

Para configurar los módulos, se utiliza un software proporcionado por la empresa Digi, llamado X-CTU. Es necesario utilizar un adaptador USB para configurar las radios y

conectarlos a la computadora. La radio que funge como emisor se configura como ZNET 2.5 COORDINATOR AT y la radio receptora como ZNET 2.5 ROUTER/END DEVICE AT. Se debe configurar la misma velocidad de transferencia para ambos módulos.

Figura 21 Radio XBee Series 2.

Capítulo 4

Simulación

Dentro de los objetivos de la simulación está el de comprobar el correcto funcionamiento de las ecuaciones que describen tanto al sistema como al controlador; asimismo, permitieron modificar variables de manera rápida y observar el impacto de dichos cambios sin poner en riesgo a la planta real.

4.1 Trayectoria sinusoidal

Se hizo la simulación para una trayectoria sinusoidal definida por las ecuaciones paramétricas:

$$\begin{aligned} x(t) &= 2t \\ y(t) &= 50 \sin(0.1t) \end{aligned} \tag{34}$$

donde t esta en s, $x(t)$ y $y(t)$ en cm.

El tiempo de simulación para esta trayectoria fue de 80 segundos. Se seleccionaron los valores de dos parámetros que permitieron modificar las ganancias del controlador en lazo cerrado, estos son el factor de amortiguamiento ξ y la frecuencia natural ω_n.

4.1.1 Simulación para $\boldsymbol{\xi = 1,\ \omega_n = 0}$

Para la primera corrida, se deseaba una respuesta críticamente amortiguada, por tanto, el factor de amortiguamiento fue $\xi = 1$ y la frecuencia natural se seleccionó de forma arbitraria, así pues, $\omega_n = 0.5$. Se llevó a cabo la simulación donde la postura inicial del robot fue $(-1, -1, 0.65)$, esta postura se seleccionó de manera arbitraria buscando tener un error inicial pequeño y se obtuvieron los siguientes resultados:

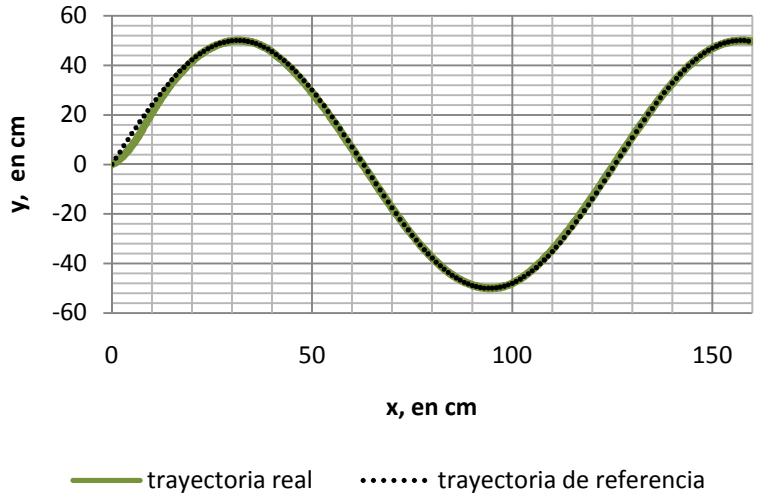


Figura 82 Posición en el área de trabajo.

En la Figura 22 se muestra en negro la trayectoria de referencia y en verde la trayectoria descrita por el robot; se puede observar que el robot siguió la trayectoria de referencia con una muy buena precisión. Se aprecia la etapa de transición y no existe sobrepaso, tal como se esperaba de acuerdo a la selección de las ganancias del controlador.

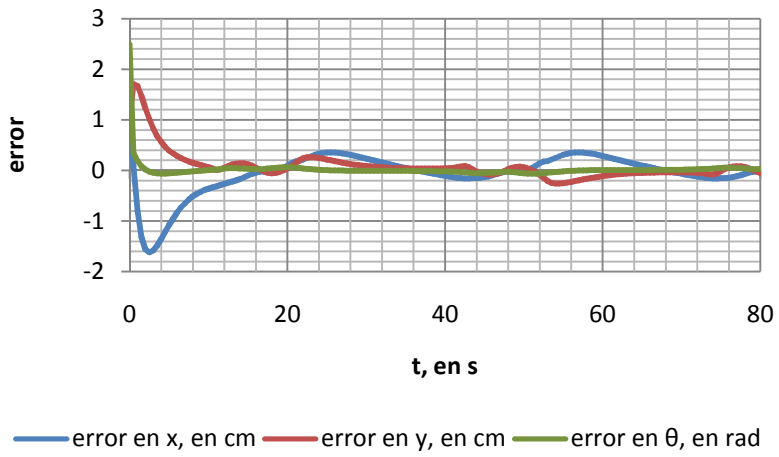


Figura 23 Error de seguimiento.

En la Figura 23 se muestra el error de seguimiento del robot durante toda la simulación; es claro que después de la etapa transitoria, el sistema se estabilizó y el error de posición no superó los 0.5 cm. El tiempo de asentamiento fue aproximadamente de 15 s y se considera una respuesta lenta.

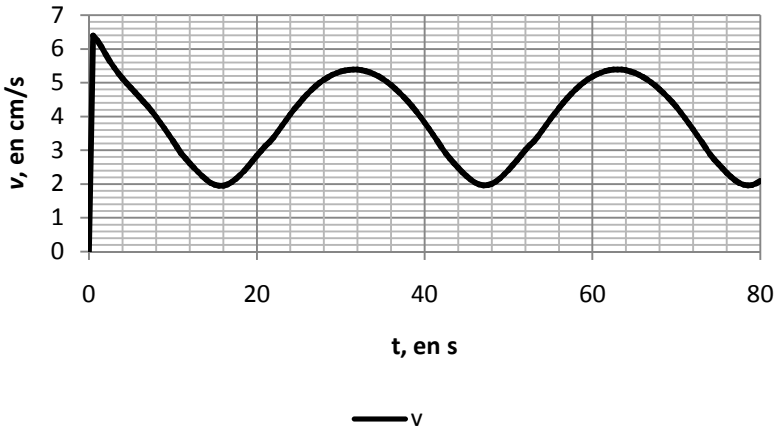


Figura 24 velocidad lineal.

La Figura 24 muestra la velocidad lineal v introducida al robot. Se observa que el pico máximo solicitado al sistema es de 6.3 cm/s, siendo éste un valor dentro del rango de operación del robot real.

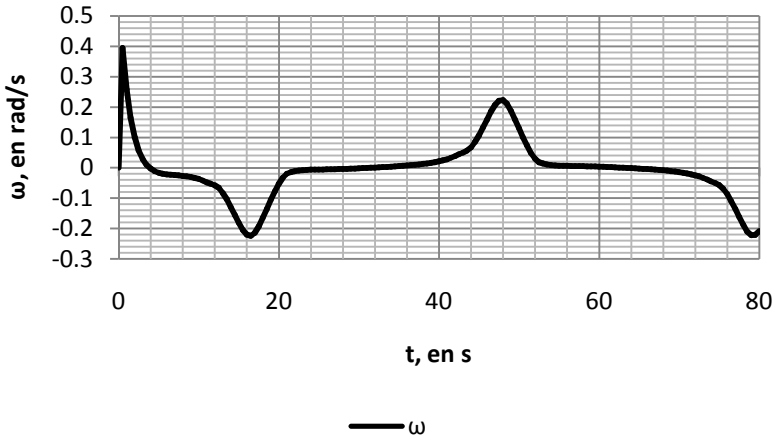


Figura 25 Velocidad angular.

La Figura 25 muestra el comportamiento de la velocidad angular ω ingresada al robot móvil durante los 80 segundos de simulación. Se aprecia el pico máximo que es de 0.39 rad/s, que es un valor dentro del rango de operación de la planta real.

De los datos arrojados en esta simulación y mostrados en las figuras, se concluye que es posible probar esta trayectoria en el robot real con los parámetros antes mencionados. El tiempo de asentamiento fue grande, por tanto, se debió probar con un valor mayor de frecuencia natural, lo cual hizo que fueran más negativos los polos deseados y en consecuencia, se obtuvo un controlador más rápido.

4.1.2 Simulación para $\xi = 0.8$, $\omega_n = 0.7$

Se buscó obtener un controlador más veloz ya que fue de interés de este trabajo que el sistema se estabilizara de manera rápida y precisa sobre la trayectoria deseada.

Se propuso cambiar el valor del factor de amortiguamiento a $\xi = 0.8$ y aumentar el valor de la frecuencia natural a $\omega_n = 0.7$. Los resultados que se obtienen de la simulación con estos nuevos valores se describen a continuación.

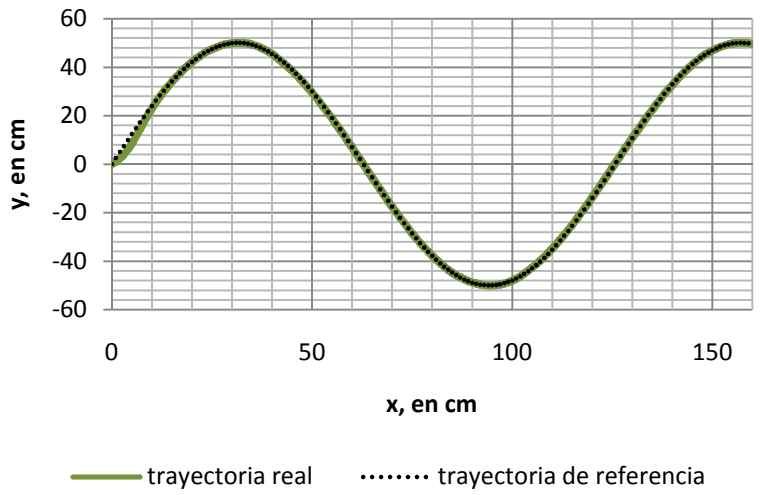


Figura 26 Posición en el área de trabajo.

La Figura 26 muestra la trayectoria de referencia en negro y la trayectoria descrita por el robot en el espacio de trabajo en color verde. Se observó que una vez que el robot se estabilizó sobre la trayectoria, éste la siguió de manera precisa a pesar del error inicial.

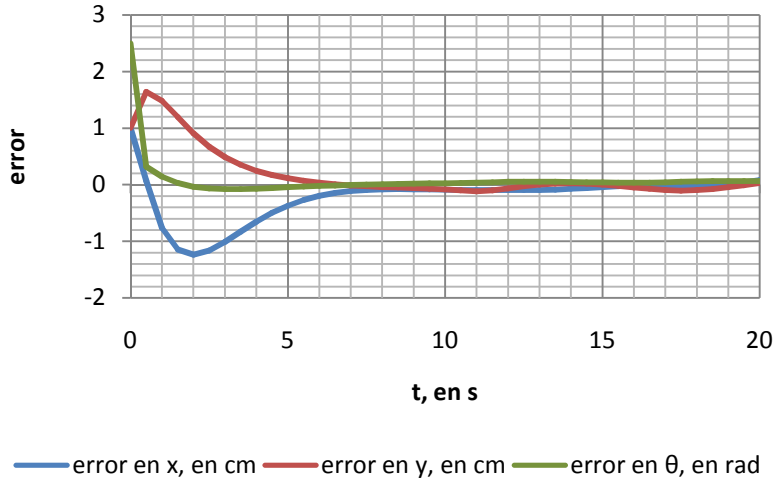


Figura 27 Error de seguimiento en 20 s.

La Figura 27 muestra el error de seguimiento del robot durante los primeros 20 s de la simulación, se aprecia que el tiempo de asentamiento fue aproximadamente de 7 s, el

error de posición en la etapa transitoria no rebasó los 3 cm y una vez estabilizado el sistema, el error de posición no superó los 0.2 cm. Con los nuevos valores asignados al factor de amortiguamiento y a la frecuencia natural del controlador, se obtuvo una respuesta más rápida y precisa que la obtenida para $\xi = 1,\ \omega_n = 0.5$.

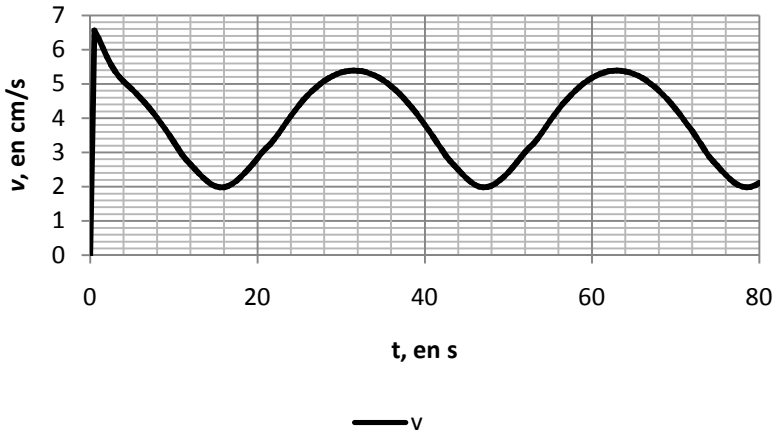


Figura 28 Velocidad lineal.

La Figura 28 muestra la velocidad lineal de entrada v ingresada al robot durante los 80 s de simulación, la máxima velocidad solicitada por el controlador para la postura inicial $(-1, -1, 0.65)$ fue de 6.55 m/s, posteriormente osciló entre 1.9 y 5.38 m/s.

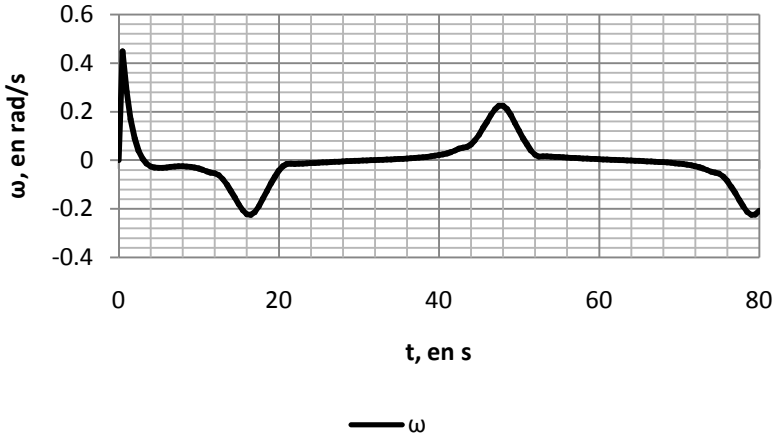


Figura 29 Velocidad angular.

En la Figura 29 se observa la velocidad angular ω introducida al sistema real por el controlador, el valor máximo requerido fue de 0.44 rad/s y en el resto de la simulación, dicho valor llegó hasta -0.22 rad/s que fue cuando el robot llegó a la primera cresta de la trayectoria y posteriormente se ingresó una velocidad angular positiva de 0.22 rad/s que fue cuando el robot alcanzó el valle de la trayectoria.

La velocidad lineal v y la velocidad angular ω solicitados por el controlador estuvieron dentro del rango de operación de la planta real, por tanto, fue posible probar la estabilización del robot real sobre esta trayectoria sinusoidal para los valores de ganancia simulados.

4.2 Trayectoria con forma de ocho

Se hicieron dos simulaciones para la trayectoria 2, la cual es una trayectoria en forma de ocho y que está definida por las ecuaciones paramétricas:

$$\begin{aligned} x &= 70\sin(2t_2) \\ y &= 50\sin(4t_2) \\ t_2 &= \frac{t\pi}{180} \end{aligned} \tag{35}$$

donde x y y están en cm, t en s.

El tiempo de simulación fue de 180 s, con este tiempo el robot hizo un ocho completo y terminó la trayectoria en la posición cero del área de trabajo. Si se requiere que el robot dé más de una vuelta (un ocho completo), se debe multiplicar el tiempo de simulación por el número de vueltas deseado.

4.2.1 Simulación para $\xi = 1,\ \omega_n = 0.5$

De manera similar a la trayectoria sinusoidal, se quiso obtener una respuesta críticamente amortiguada, es decir, que no exista sobrepaso en la respuesta del sistema; por tanto, se propuso asignar el valor de 1 al factor de amortiguamiento ξ y de 0.5 para la frecuencia natural ω_n.

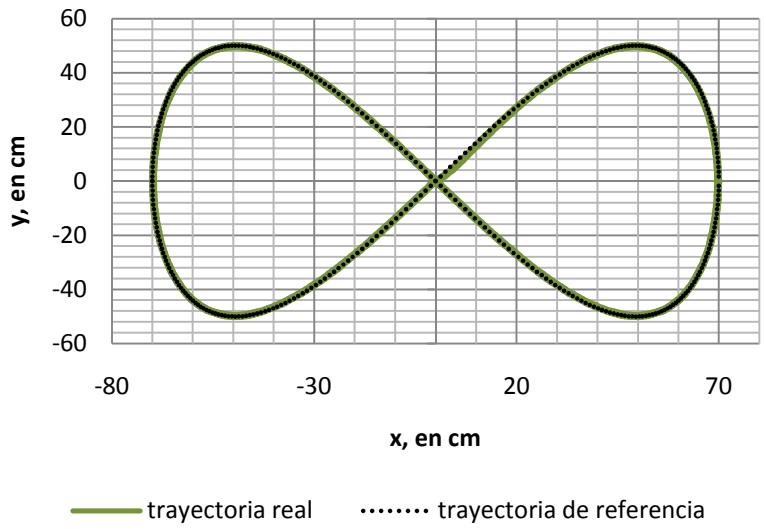


Figura 30 Posición en el área de trabajo.

La Figura 30 muestra en negro la trayectoria de referencia y en verde la trayectoria descrita por el robot; se observa la respuesta transitoria del sistema pues el robot no se estabilizó inmediatamente sobre la trayectoria pero sí hizo correcciones moderadas para que, una vez alcanzada la trayectoria, la siguiera de manera precisa y confiable.

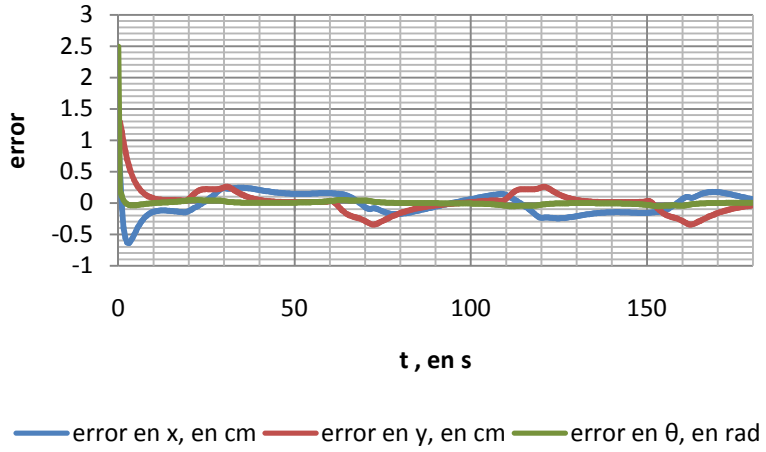


Figura 31 Error de posición.

La figura 31 muestra el error de seguimiento del robot en la simulación; para este controlador, el tiempo aproximado de asentamiento fue de 10 s, posterior a los cuales, el

error de posición no superó los 0.5 cm y el error relacionado con el ángulo fue muy cercado a cero. Se hace notar que el tiempo de asentamiento es considerablemente menor en esta simulación y esto era lo esperado, pues este controlador es más rápido aunque demanda valores de velocidad lineal y velocidad angular mayores a los solicitados en la primera simulación.

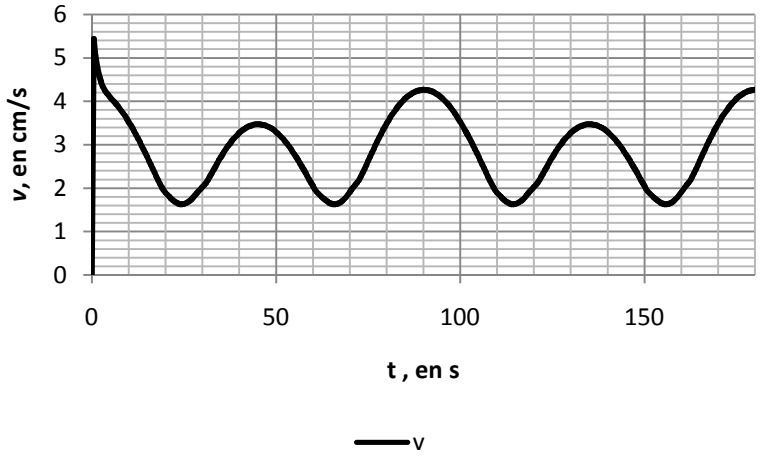


Figura 32 Velocidad lineal.

La Figura 32 muestra la velocidad lineal v introducida al robot; en un inicio tiene un pico de 5.43 cm/s y posteriormente se hizo oscilar entre 1.6 y 4.2 cm/s. Estos valores se encuentran dentro del rango de operación del robot móvil real.

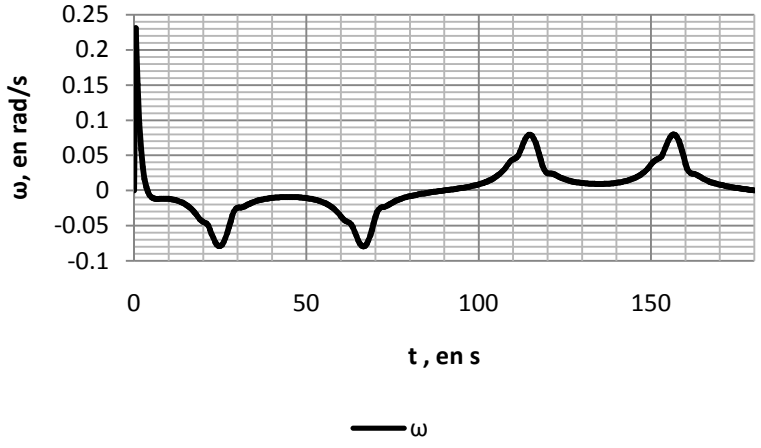


Figura 33 Velocidad angular.

La Figura 33 muestra la velocidad angular ω ingresada al robot; fue notorio que al inicio el controlador solicitó una velocidad de 0.23 rad/s para corregir la dirección de robot y posteriormente se suavizó y pidió valores más pequeños. La entrada solicitada en esta simulación pudo ser desarrollada por el robot real.

4.2.1 Simulación para $\xi = 0.8,\ \omega_n = 0.5$

Luego se modificó el valor del factor de amortiguamiento ξ, el cual fue de 0.8 y la frecuencia natural ω_n se mantuvo igual a la simulación anterior.

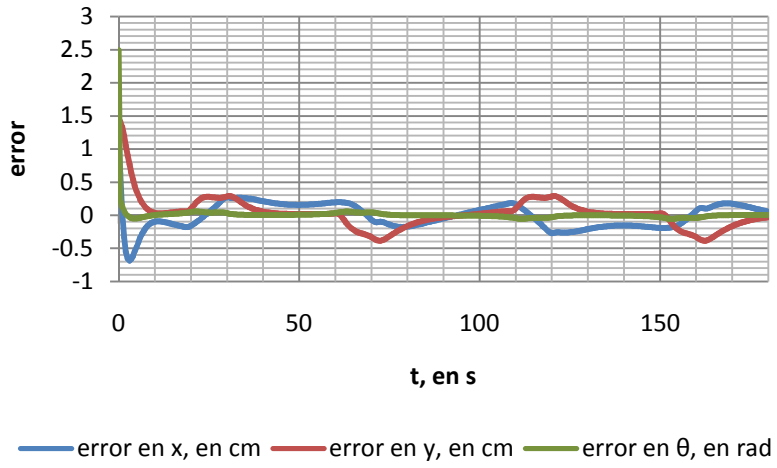


Figura 34 Error de seguimiento.

La Figura 34 describe el comportamiento del error de seguimiento con respecto al tiempo, se observa que los primeros 10 s de la simulación correspondieron a la respuesta transitoria, desde la posición inicial del robot hasta que alcanzó un valor cercano a la trayectoria de referencia. Posteriormente, el error no sobrepasó los 0.5 cm y el error de orientación fue muy cercano a cero.

La Figura 35 muestra la velocidad lineal v que se introdujo al robot en la simulación; el valor máximo de velocidad requerido para este control fue de 5.2 m/s, siendo 0.23 cm/s menor a lo solicitado con $\xi = 1$.

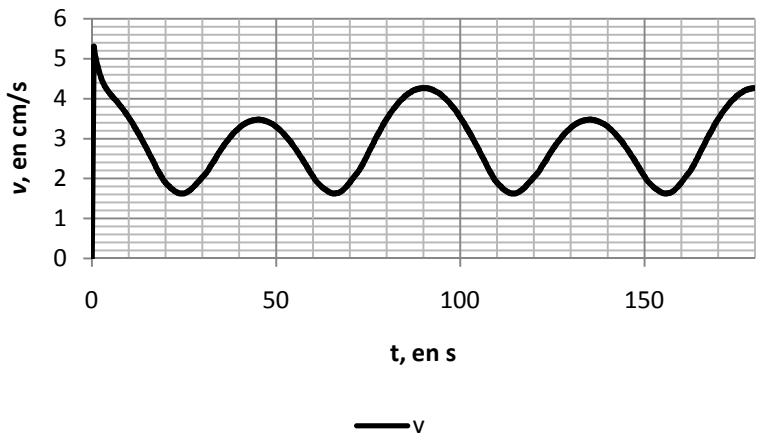


Figura 35 Velocidad lineal.

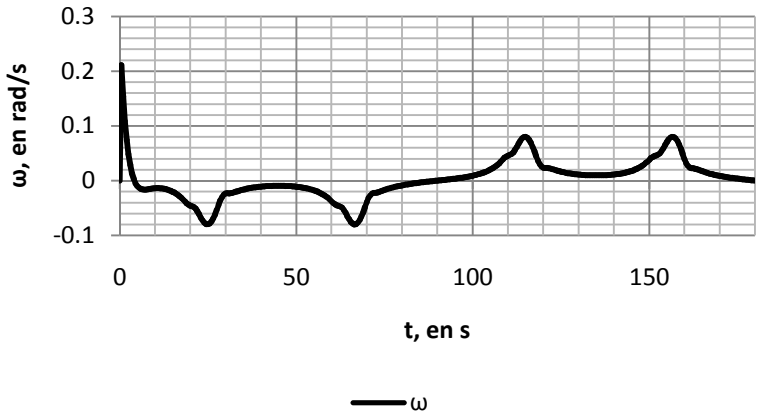


Figura 36 Velocidad angular.

La Figura 36 muestra la velocidad angular de entrada ω con respecto al tiempo de simulación, el valor máximo solicitado por el controlador fue de 0.21 rad/s, siendo éste un valor menor al obtenido en la simulación para $\xi = 1$ que igualmente permitió al robot seguir la trayectoria de manera suave pese al error inicial. Estos valores fueron fácilmente desarrollados por el robot real.

Capítulo 5

Resultados experimentales

Se llevaron a cabo cuatro experimentos en total, de los cuales dos de ellos correspondieron a la estabilización sobre una trayectoria sinusoidal y dos sobre una trayectoria con forma de ocho.

Se presentan gráficas de la trayectoria real y la trayectoria de referencia en el espacio de trabajo, gráficas del error de seguimiento, gráficas de la postura del robot y esquemas de las velocidades de entrada solicitadas por el controlador al robot móvil durante la estabilización alrededor de las trayectorias mencionadas.

5.1 Trayectoria sinusoidal

Se realizaron dos pruebas experimentales para la trayectoria definida en el tiempo por las ecuaciones paramétricas 34, el valor de los parámetros del controlador para las pruebas experimentales fue definido en el capítulo 4, esto para conocer el valor teórico de las entradas $(v\ y\ \omega)$ y verificar que el robot móvil con configuración diferencial real fuera capaz de desarrollar dichas velocidades.

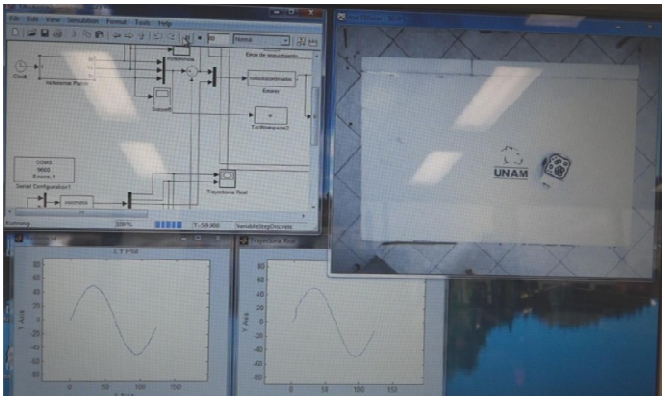


Figura 37 Entorno de pruebas para una trayectoria sinusoidal.

5.1.1 Estabilización con $\xi = 1,\ \omega_n = 0.5$

La posición inicial del robot para este experimento fue $x = -1$ cm, $y = -1.2$ cm y la orientación inicial fue $\theta = 0.64$ rad.

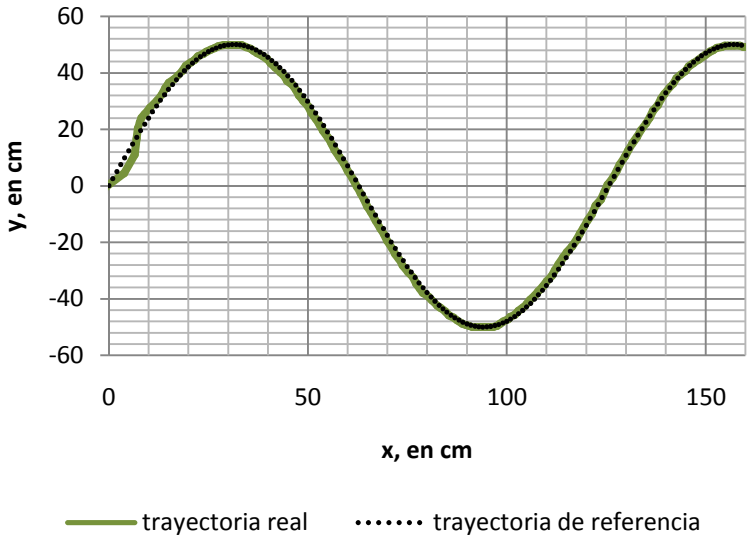


Figura 38 Posición en el área de trabajo.

En la Figura 38 se muestra la trayectoria de referencia en negro y la trayectoria descrita por el robot móvil en verde, el área graficada representa el espacio de trabajo del robot. Se aprecia que al inicio del recorrido es cuando el robot presentó el mayor error, puesto que existió un error inicial. Posterior a esta etapa transitoria, el robot siguió la trayectoria con buena precisión.

A pesar de que se seleccionó el factor de amortiguamiento $\xi = 1$ para que no existiera sobrepaso, es claro que cuando el robot corrigió su posición para seguir la trayectoria desde su postura inicial, se pasó de la misma e hizo una nueva corrección girando en sentido horario.

La Figura 39 muestra de azul el error de posición en x, en rojo el error de posición en y en verde el error de orientación θ. El error de orientación se mantuvo muy cercano a cero durante todo el experimento salvo al inicio del mismo que corresponde a la respuesta transitoria del sistema. El error de posición en x alcanzó un valor máximo de 4.06 cm que corresponde a la etapa transitoria de la prueba y una vez estabilizado en la trayectoria, el error máximo fue de 1.16 cm. El error de posición en y alcanzó un valor máximo de 8.28 cm durante la respuesta transitoria y de 2 cm una vez estabilizado en la trayectoria deseada.

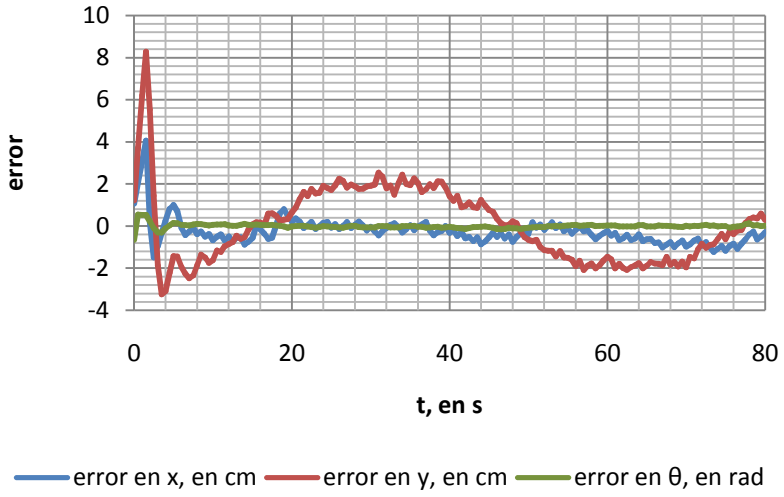


Figura 39 Error de seguimiento.

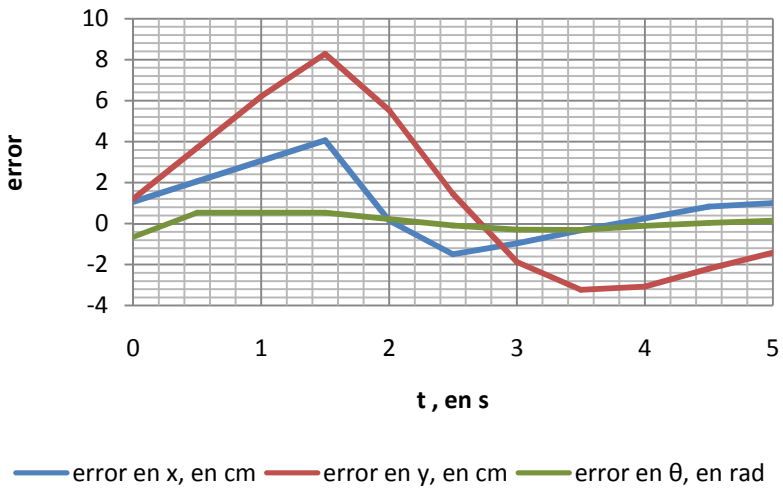


Figura 40 Error de posición del robot de 0 a 5 s.

En la Figura 40 se muestra el error de posición y de orientación del robot respecto a la trayectoria de referencia durante los primeros cinco segundos del experimento. Se aprecia que el tiempo de asentamiento para estos parámetros del controlador fue cercano a 5 s, que es cuando el error de posición fue menor o igual a 2 cm y se considera que es suficientemente pequeño y, por tanto, que el seguimiento de la trayectoria fue preciso. Este valor está alejado de lo obtenido en la simulación donde el tiempo de asentamiento fue de 15 s.

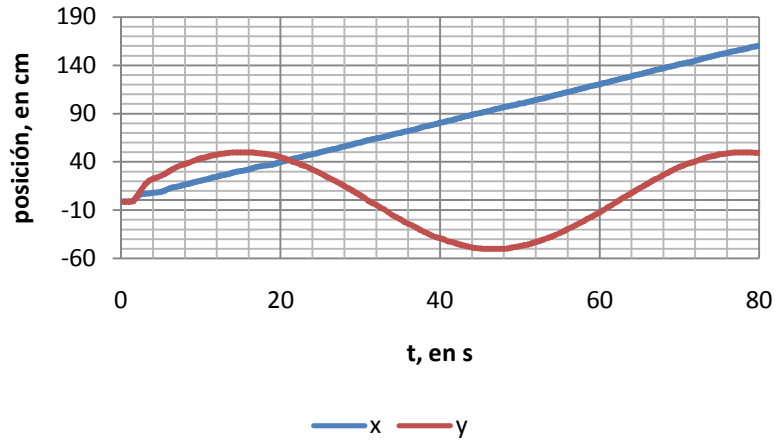


Figura 41 Posición en x y en y del robot.

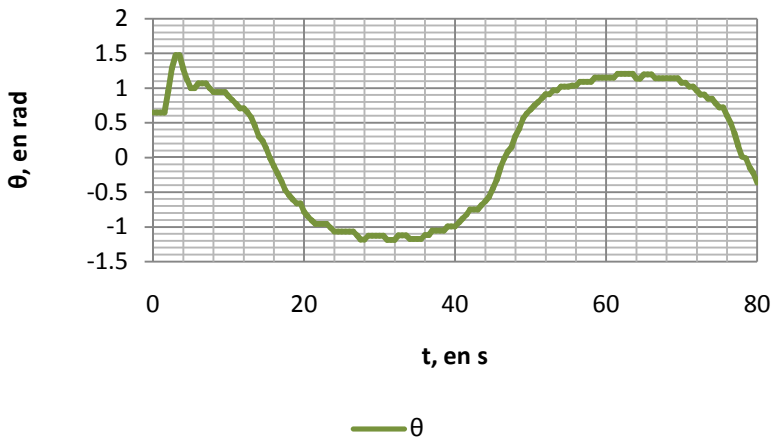


Figura 42 Posición angular del robot.

La figura 41 muestra la posición del robot durante la prueba, la posición en x se mantuvo como el doble del tiempo del experimento y la posición en y describió una sinusoidal con amplitud de 50 cm. Este comportamiento coincide con la trayectoria de referencia definida por las ecuaciones paramétricas 34.

En la Figura 42 se expone el comportamiento del ángulo de orientación θ, el cual osciló entre 1.17 y -1.17 rad. En los primeros diez segundos se observó que el comportamiento del ángulo de orientación estuvo en una etapa transitoria, debido al error inicial del robot móvil.

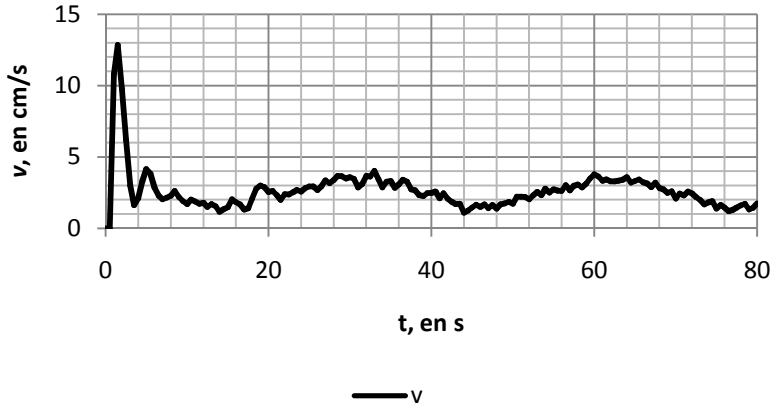


Figura 43 Velocidad lineal del robot.

La Figura 43 muestra la velocidad lineal solicitada por el controlador en esta prueba experimental. Se apreció un pico de velocidad de 12.8 cm/s al inicio del experimento, este valor fue considerablemente alto pues en la simulación fue de 6.3 m/s. Se observó que la velocidad fue menor en los picos y en los valles de la trayectoria cercana a 1.1 cm/s y fue mayor en las partes rectas llegando a valores de 4 cm/s.

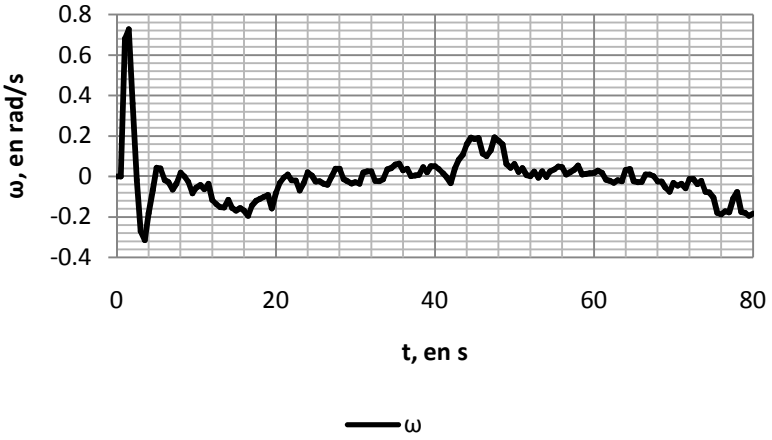


Figura 44 Velocidad angular del robot.

La Figura 44 muestra la velocidad angular solicitada por el controlador. Se aprecia que la velocidad angular ω fue negativa cuando el robot giró en sentido horario y positiva cuando lo hizo en sentido anti-horario. El valor máximo solicitado por la ley de control fue de 0.72 rad/s.

5.1.2 Estabilización con $\xi = 0.8,\ \omega_n = 0.7$

La posición inicial del robot para este experimento fue $x = -1.2$ cm, $y = -1.5$ cm y la orientación inicial fue $\theta = 0.65$ rad, esto con el fin de tener un error inicial moderado.

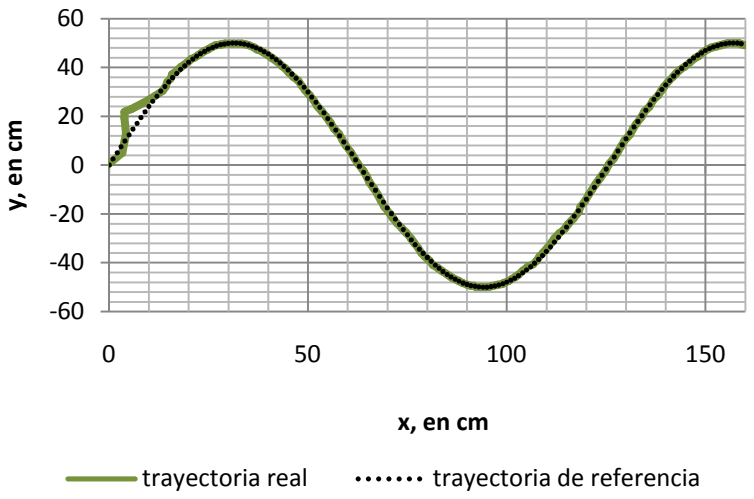


Figura 45 Posición del robot en el área de trabajo.

La Figura 45 muestra la trayectoria que describió el robot móvil real con referencia a la trayectoria ideal. Se aprecia un claro sobrepaso al inicio de la trayectoria por un sobregiro en sentido anti-horario de la planta, el cual fue posteriormente corregido. Se observa el efecto en la disminución del valor del factor de amortiguamiento ξ y el aumento de la frecuencia natural ω_n respecto a la prueba pasada pues el sistema reaccionó más rápido y de manera más brusca ante el error, lo que provocó un sobrepaso mayor que el obtenido en el experimento anterior. Se apreció que pese al error inicial, el robot móvil siguió de manera precisa la trayectoria de referencia.

La Figura 46 muestra el error de la postura del robot respecto a la postura de referencia. Durante la respuesta transitoria del robot, se apreció un error máximo de 4.2 cm en el eje x y de 8.6 cm en el eje y. Ya que el sistema se estabilizó, el error máximo sobre el eje x es de 1 cm y de 2.3 cm en el eje y. El error de orientación máximo fue de 0.53 rad durante la etapa transitoria y de 0.11 rad cuando la planta se estabilizó.

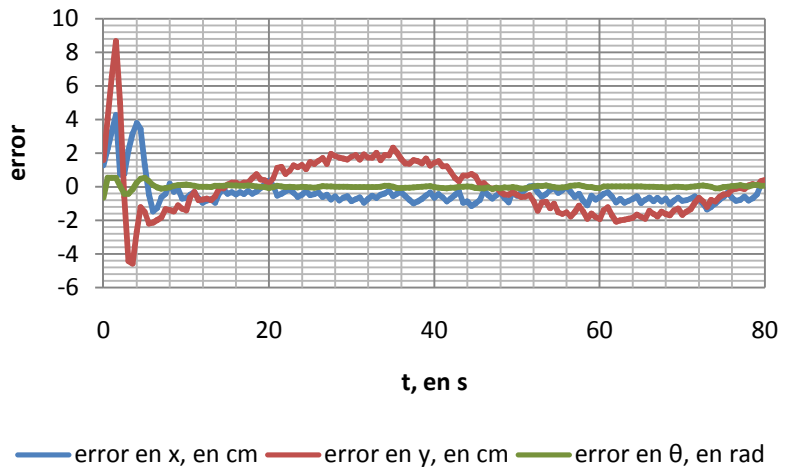


Figura 46 Error de seguimiento del robot.

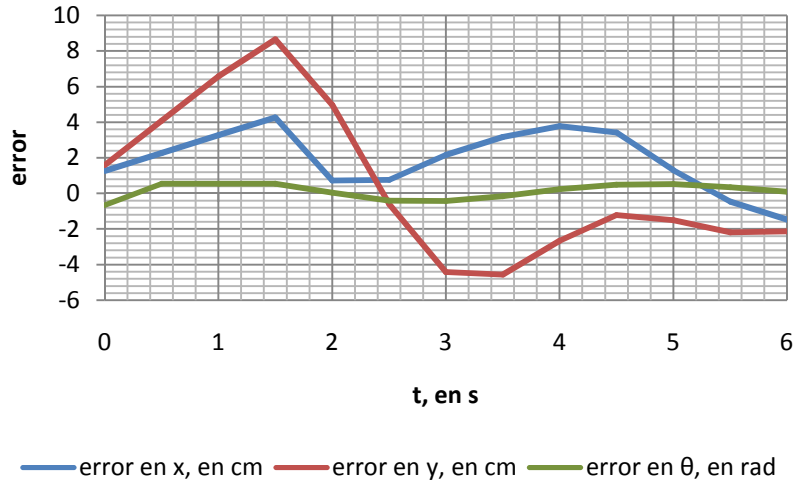


Figura 47 Error de seguimiento del robot de 0 a 6 s.

La Figura 47 muestra el error de la postura del robot durante los primeros seis segundos de la prueba experimental. Se aprecia que el tiempo de asentamiento para estos parámetros del controlador fue aproximadamente de 6 s, siendo un tiempo mayor a pesar de que fue un control más rápido, esto debido a la sobre-corrección del robot ante el error inicial.

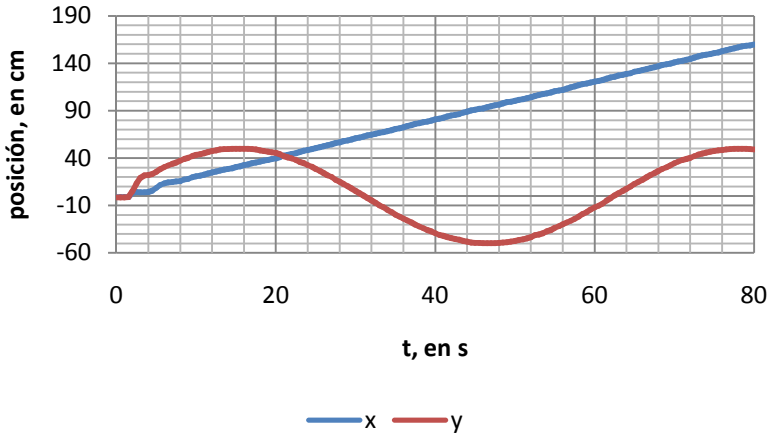


Figura 48 Posición en *x* y en *y* del robot.

La Figura 48 muestra la posición del robot en el eje x y en el eje y durante los 80 s que duró el experimento.

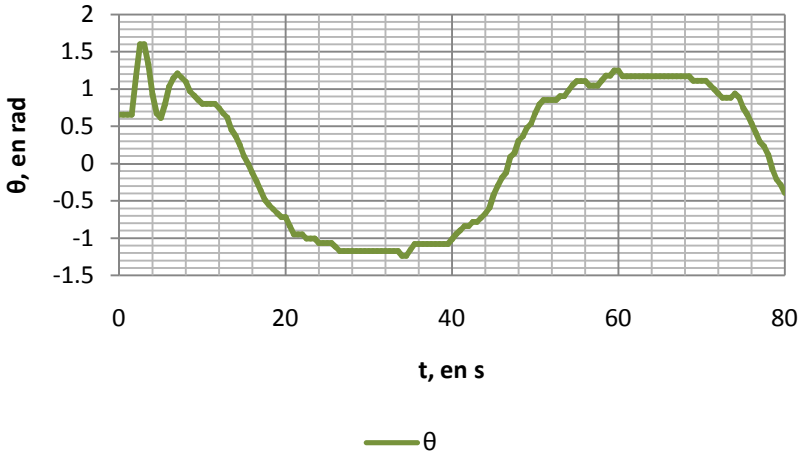


Figura 49 Posición angular del robot.

En la Figura 49 se muestra el comportamiento del ángulo de orientación θ. Se aprecia que, al inicio del experimento dicho valor osciló entre 0.65 rad y 1.59 rad; posteriormente regresó a un valor cercano a 0.65 rad en un lapso de tiempo de 8 s aproximadamente.

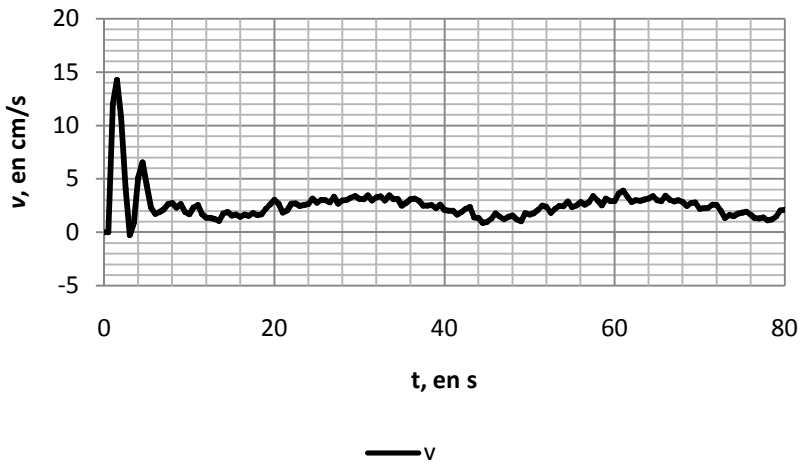


Figura 50 Velocidad lineal del robot.

La Figura 50 muestra la velocidad lineal solicitada por el controlador a la planta real, se aprecia un pico de 14.2 cm/s en la etapa transitoria y fue mayor al obtenido en el primer experimento, esto era esperado ya que un control más rápido demanda valores de velocidad más grandes y, en este caso, provocó una respuesta transitoria más larga y con mayor error durante la misma. Posteriormente, los valores de velocidad fueron muy parecidos a los vistos en la prueba anterior.

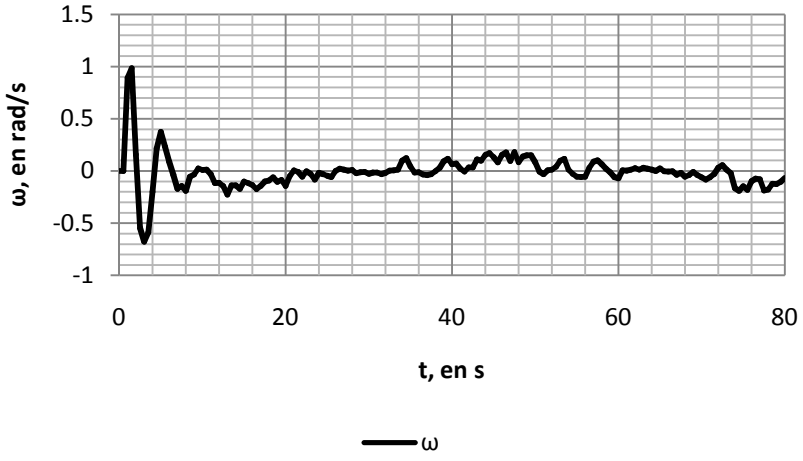


Figura 51 Velocidad angular del robot.

En la Figura 51 se muestra la velocidad angular ω solicitada por el controlador a la planta real. Se aprecia un salto de 0 a 0.98 rad/s y luego de este último valor hasta un

valor negativo de -0.67 rad/s, que corresponde al sobregiro del robot al inicio del experimento y a la corrección para estabilizarlo alrededor de la trayectoria sinusoidal.

Figura 52 Prototipo funcional en el ambiente de pruebas experimentales.

5.2 Trayectoria con forma de ocho

Se realizaron dos experimentos para la estabilización del robot móvil real alrededor de la trayectoria descrita por las ecuaciones paramétricas 35. En el primero, los valores de los parámetros del controlador fueron $\xi = 1, \omega_n = 0.5$, para el segundo los valores fueron $\xi = 0.8$ y $\omega_n = 0.5$.

5.2.1 Estabilización con $\boldsymbol{\xi = 1,\ \omega_n = 0.5}$

La posición inicial del robot para este experimento fue $x = -0.6$ cm, $y = -0.2$ cm y la orientación inicial es $\theta = 0.61$ rad. Este punto se eligió de manera arbitraría con el objetivo de tener un error inicial pequeño.

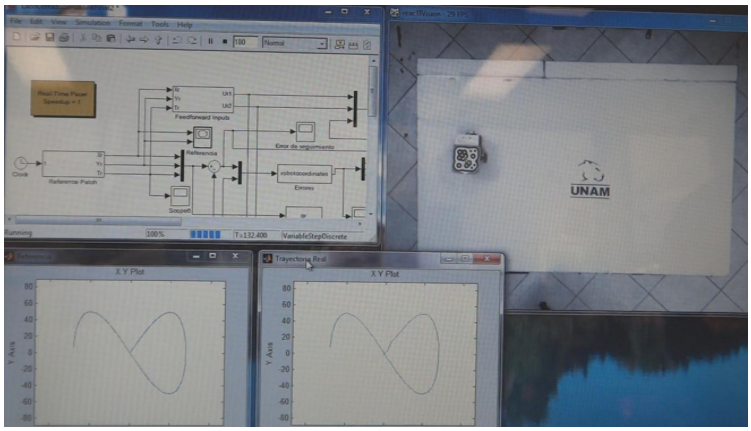


Figura 53 Entorno de pruebas para una trayectoria con forma de ocho.

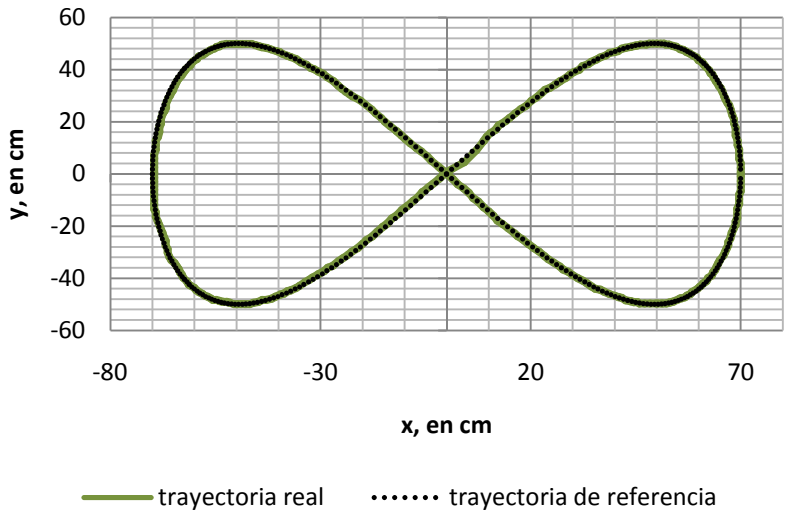


Figura 54 Posición del robot en el espacio de trabajo.

La Figura 54 muestra la trayectoria de referencia en negro y la trayectoria descrita por el robot en verde, se aprecia que al inicio del experimento el error fue mayor, esto debido al error inicial que el controlador corrigió para acercarse de manera suave a la trayectoria de referencia. Ya que el robot alcanzó la trayectoria, la siguió con una buena precisión hasta el final de la prueba experimental.

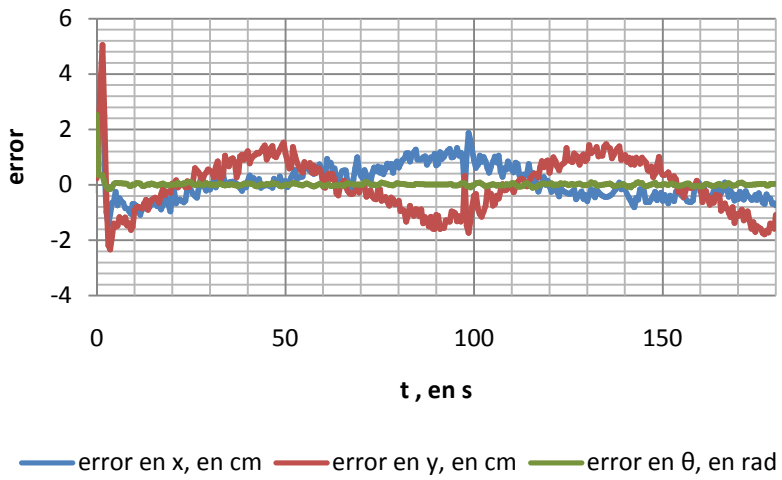


Figura 55 Error de seguimiento del robot.

En la Figura 55 se observa el error de posición y de orientación de la planta respecto a la trayectoria de referencia, se observa de azul el error correspondiente al eje x el cual tiene un pico máximo de 4.3 cm durante la respuesta transitoria del sistema y posteriormente, el error fue muy pequeño, de entre 0.5 y 1 cm. El error sobre el eje y se observa en color rojo, mismo que alcanzó su valor máximo durante la respuesta transitoria del robot con un valor de 5 cm y, ya que el robot se estabilizó sobre la trayectoria, el error no superó los 2 cm.

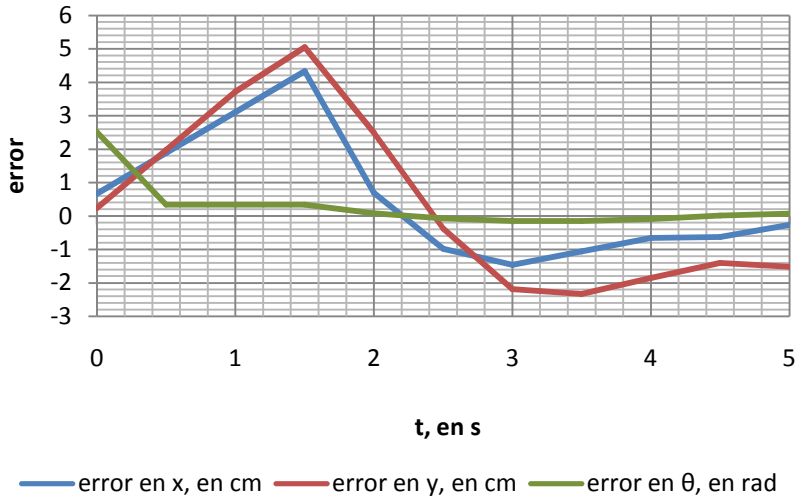


Figura 56 Error de seguimiento del robot de 0 a 5 s.

La Figura 56 muestra el error de posición y de orientación durante los primeros 5 s del experimento, se aprecia que el tiempo de asentamiento del sistema fue de 3 s aproximadamente; posterior a ellos, el error de posición no superó los 2 cm y el error de orientación se mantuvo muy cercano a cero.

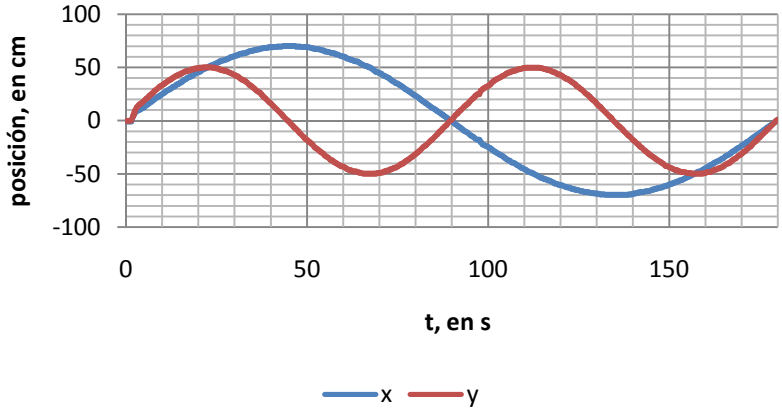


Figura 57 Posición en *x* y en *y* del robot.

La Figura 57 muestra la posición del robot en el eje x y en el eje y. Se pueden apreciar claramente las funciones sinusoidales que definen esta trayectoria y que están representadas por la expresión 35 donde el periodo de la función $y = f(t)$ tiene el doble de frecuencia que el de la función que define $x = f(t)$; que era lo esperado.

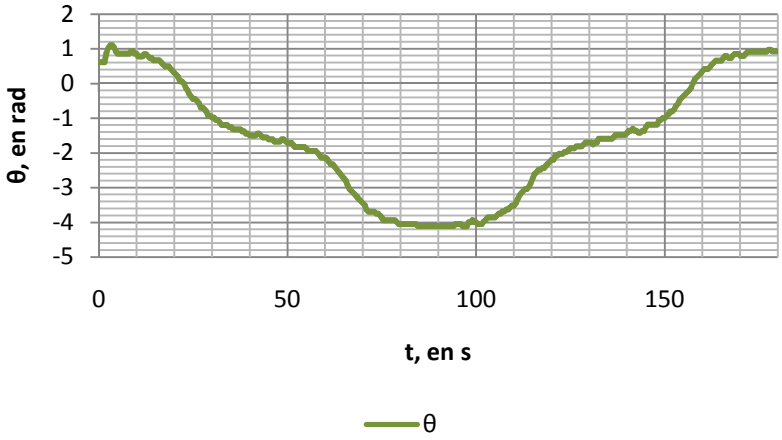


Figura 58 Posición angular del robot.

En la Figura 58 se muestra la orientación del robot durante la prueba experimental, se observa que a la mitad de la prueba hubo un cambio la orientación del robot, es decir, el robot giró en sentido anti-horario para completar la segunda parte de la trayectoria y regresó al origen del espacio de trabajo.

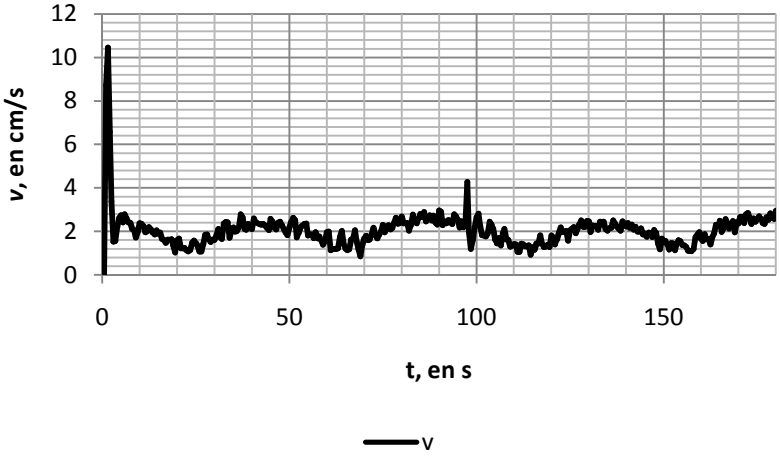


Figura 59 Velocidad lineal del robot.

La Figura 59 muestra la velocidad lineal solicitada por el controlador al robot móvil para que éste siga la trayectoria, se aprecia un pico correspondiente a la parte transitoria en la que *v* alcanzó su valor máximo en 10.4 m/s, posteriormente, el comportamiento de esta velocidad osciló entre 2.7 y 1.3 m/s de forma cíclica, con una frecuencia aproximada de 22 mHz.

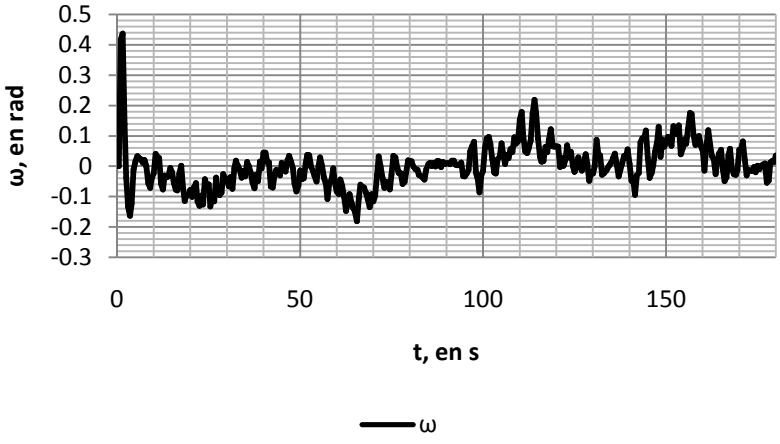


Figura 609: Velocidad angular del robot.

La Figura 60 muestra la velocidad angular ω introducida por el controlador a la planta real, se aprecia un pico máximo de 0.43 rad/s en lo que correspondió a la respuesta transitoria del controlador. Se aprecia que dicha velocidad se mantuvo en valores negativos en la primera mitad del experimento, esto era esperado pues los primeros giros del robot fueron en sentido horario y, la segunda parte coincidió con valores positivos pues el robot giró en sentido anti-horario.

5.2.2 Estabilización con $\xi = 0.8,\ \omega_n = 0.5$

La posición inicial del robot para este experimento fue $x = -0.7$ cm, $y = -0.47$ cm y la orientación inicial fue $\theta = 0.63$ rad. Se escogió esta posición inicial de manera arbitraria por la finalidad de tener un error inicial moderado.

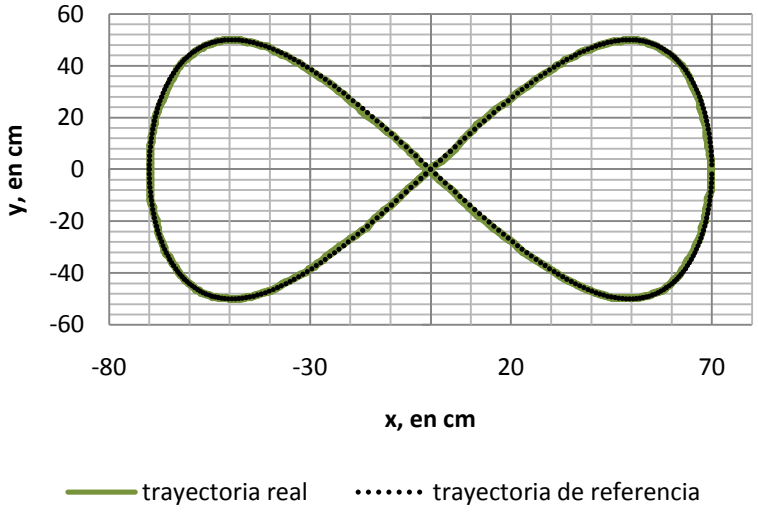


Figura 61 Posición del robot en el área de trabajo.

La Figura 61 muestra la trayectoria que describió el robot móvil con configuración diferencial en el área de trabajo y se comparó con la trayectoria de referencia o trayectoria ideal. El robot siguió de manera precisa la trayectoria durante todo la prueba, salvo por los primeros segundos que correspondieron a la respuesta transitoria del sistema.

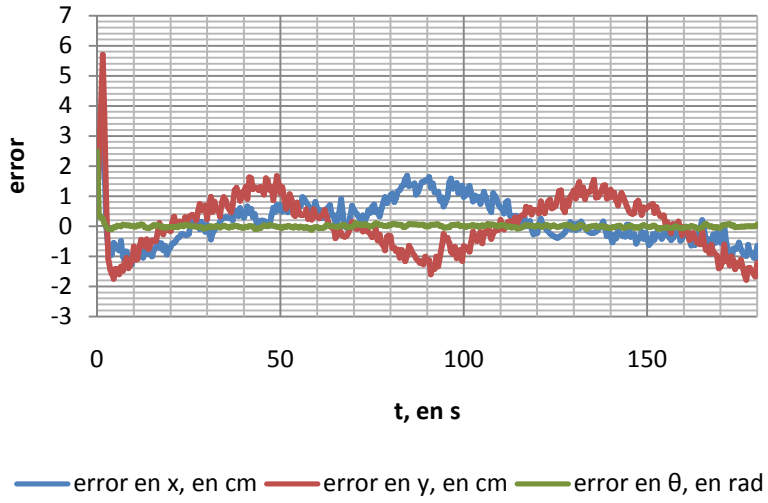


Figura 62 Error de seguimiento del robot.

En la Figura 62 se muestra el error de posición y de orientación del robot durante este experimento. El error máximo fue de 4.7 cm y 5.6 cm para el eje x y y, respectivamente. Posterior a la etapa transitoria, el error de seguimiento no rebasó los 2 cm lo cual se encuentra dentro de los objetivos de este trabajo. El error de orientación máximo fue de 0.32 rad y se presentó durante la respuesta transitoria del robot.

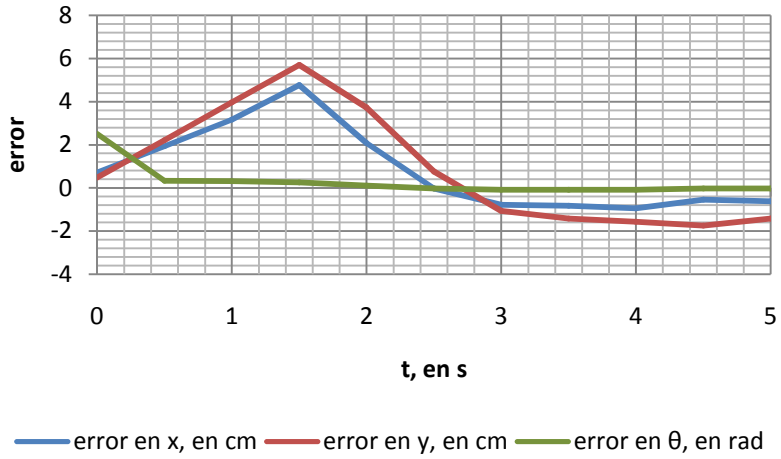


Figura 63 Error de seguimiento del robot de 0 a 5 s.

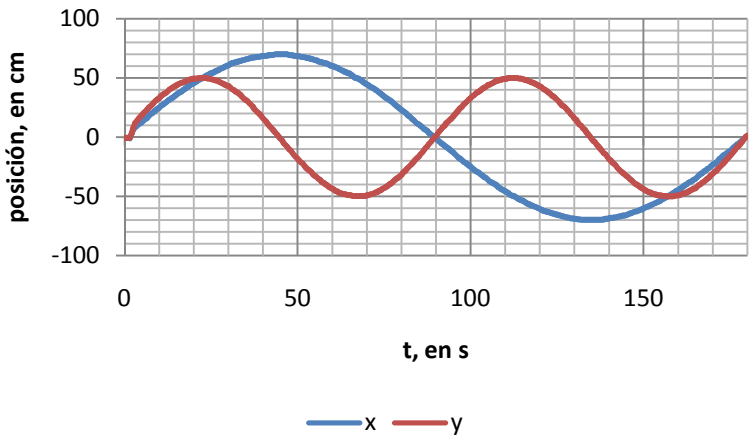


Figura 64 Posición del robot en *x* y en *y*.

La Figura 63 muestra el error de seguimiento durante los primeros cinco segundos del experimento. Se observa que el tiempo de asentamiento fue aproximadamente de 3 s, se obtuvo una respuesta más rápida y más precisa que la que se consiguió en la prueba experimental anterior, a pesar de que, teóricamente éste controlador fue más lento.

La Figura 64 muestra el comportamiento de la posición del robot en los ejes *x* y *y*, asimismo, se observa nuevamente que el comportamiento coincidió con lo descrito por las ecuaciones paramétricas que definen la trayectoria de referencia.

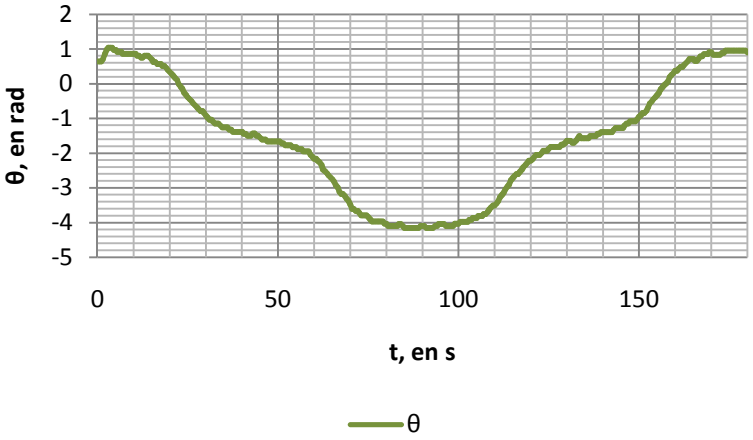


Figura 65 Posición angular del robot.

En la Figura 65 se muestra el comportamiento del ángulo de orientación θ. Se observa que, en caso de que se desee que el robot móvil haga más de una trayectoria en forma de ocho, el comportamiento muestra periodicidad con una frecuencia de 5 [mHz].

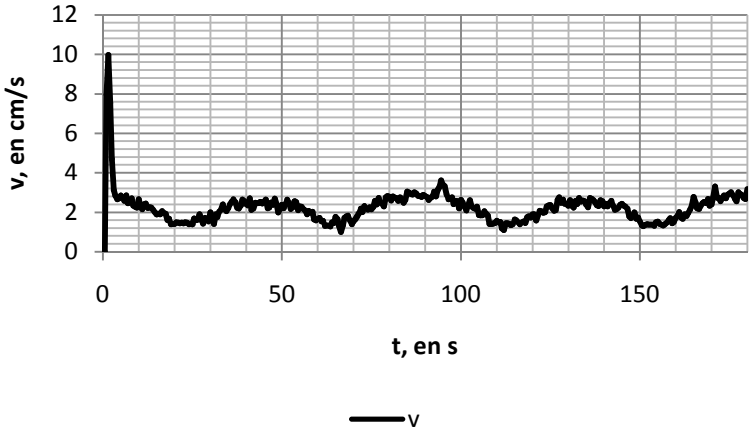


Figura 66 Velocidad lineal del robot.

La Figura 66 muestra que la velocidad lineal máxima solicitada por la ley de control para estabilizar al robot fue de 9.9 cm/s, este valor fue menor que el solicitado en el experimento anterior y esto era esperado debido a que se utilizó un controlador más lento, el cual exigió velocidades menores, no obstante, la precisión en esta prueba fue mayor, el tiempo de asentamiento y el sobrepaso fue menor.

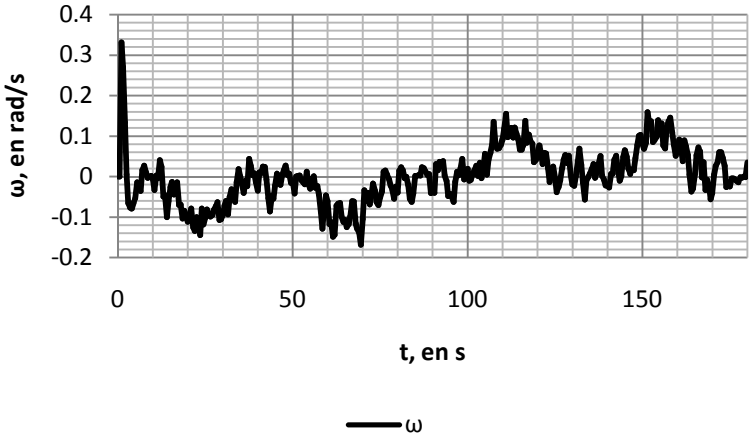


Figura 67 Velocidad angular del robot.

En la Figura 67 se muestra la velocidad angular ω introducida a la planta. Se aprecia un valor máximo de 0.33 rad/s durante la etapa transitoria, este valor fue notablemente inferior al máximo solicitado por la ley de control en el experimento anterior. Para valores menores del factor de amortiguamiento y frecuencia natural para el controlador, se obtuvieron valores de velocidades menores. El valor de esta velocidad fue negativo en la primera mitad del experimento debido a que el robot giró únicamente en sentido horario, posteriormente, este valor fue positivo y coincidió con los giros en sentido anti-horario del robot.

Capítulo 6

Conclusiones y trabajo a futuro

6.1 Conclusiones

Se llevaron a cabo cuatro simulaciones en las que se comprobó la validez del modelo cinemático en espacio de estados del robot móvil con configuración diferencial, así como la del controlador que lo estabiliza alrededor de una trayectoria C2; en dichas simulaciones se observó que, mediante la manipulación del factor de amortiguamiento ξ y la frecuencia natural ω_n, se puede modificar los valores de las ganancias del controlador; para valores mayores de estos parámetros se obtuvo un controlador más rápido, es decir, el tiempo de asentamiento del robot era menor, aunque se le ingresaran velocidades más altas, de manera contraria, para valores menores se obtuvo un controlador más lento que exigía velocidades más pequeñas. Las simulaciones fueron útiles para determinar valores de factor de amortiguamiento ξ y frecuencia natural ω_n, que no exigieran al robot real una velocidad que no fuera capaz de desarrollar.

Se llevaron a cabo dos pruebas experimentales para una trayectoria sinusoidal. En el primer experimento, se observó un sobrepaso del robot a pesar de que se buscaba una respuesta críticamente amortiguada. En el segundo experimento, se esperaba una

respuesta más rápida con la existencia de sobrepaso: El sobrepaso existió de forma evidente y el tiempo de asentamiento fue mayor, a pesar de que teóricamente debía ser menor, pero para ambos casos, el robot se estabilizó sobre la trayectoria de manera exitosa con un error cercano o menor a 2 cm a pesar de la existencia de un error inicial. De esta forma, se comprobó experimentalmente la validez del modelo cinemático del robot móvil con configuración diferencial y del controlador de seguimiento de trayectorias C2 en lazo cerrado.

En el experimento para la estabilización alrededor de una trayectoria sinusoidal se observó un sobrepaso grande. Esto sucede debido a que dicho sistema no mide la posición del robot de forma continua, es decir, hay lapsos de tiempo en los que el sistema de visión no envía datos al controlador, por tanto, el controlador no modifica las entradas que ingresan al sistema y esto ocasiona que la estabilización sea menos precisa o que incluso el robot pueda perderse. Por tanto, se concluye que este error será menor para trayectorias que demanden menor velocidad lineal y angular, es decir, en las que el robot se tenga que mover más lento.

Se llevaron a cabo dos experimentos para una trayectoria con forma de ocho. El robot móvil con configuración diferencial se estabilizó sobre dicha trayectoria de manera precisa y rápida, con tiempos de asentamiento de 3 s y un error no mayor a 2 cm a pesar de la existencia de un error inicial. Mediante las cuatro pruebas experimentales se puede concluir que el controlador expuesto es eficiente, confiable y efectivo, siempre que la trayectoria sea tipo C2.

Las pruebas experimentales difieren de las simulaciones debido a perturbaciones existentes en la superficie sobre la cual se mueve el robot, el tiempo de muestreo del sistema de visión, que funge como sensor de posición y de orientación del robot, el cual es de 500 ms, la precisión del sistema de visión, la diferencia entre el valor de velocidad demandado por el controlador y el desarrollado por la planta real, la inercia del robot, la incertidumbre de las dimensiones de las ruedas y de la distancia entre ellas, el posible error en la localización del punto P leido y la verdadera posición del mismo en el robot y la capacidad de cómputo utilizada.

El uso del modelo cinemático del robot móvil diferencial en espacio de estados fue suficiente para estabilizarlo con buena precisión alrededor de las trayectorias propuestas.

6.2 Trabajo a futuro

Las principales fallas que presentó la implementación de este experimento se relacionaron con la existencia de un tiempo de muestreo grande, el cual se pudo reducir hasta un valor de 500 ms pero que, para ciertas trayectorias significó un gran lapso en el cual el sistema se queda ciego, es decir, no fue capaz de realimentar al controlador. La siguiente propuesta podría ser la implementación de un sistema de realimentación más eficiente que permita un tiempo de muestreo menor; asimismo, dado que la cámara deforma la imagen en las orillas reduciendo la precisión de las mediciones, se podría sugerir la implementación de algún algoritmo o método para rectificar la imagen y eliminar la deformación.

Otro punto que no fue tomado en cuenta en este trabajo es el modelo dinámico del robot: se da por hecho que los motores son suficientemente potentes para mover al robot de acuerdo a cómo y cuándo lo solicite el controlador en lazo cerrado, lo cual es un error pues aunque los motores llevan un control PID para alcanzar la velocidad solicitada, existe una etapa transitoria en la respuesta de los mismos ante la entrada. En consecuencia, la implementación de un control dinámico podría incrementar la precisión del control aunque aumentaría de manera significativa la complejidad del problema.

Apéndices

A1 Códigos de funciones s y diagrámas implementados en Simulink de MATLAB

Función s de visión

```
function [sys,x0,str,ts]=visiondos(t,x,u,flag) %Inicia función s
switch flag,
    case 0
        [sys,x0,str,ts]= mdlInitializeSizes;
    case 3
        sys = mdlOutputs(t,x,u);
    case {1,2,4,9}
        sys = [];
    otherwise
        error(['Unhandled flag=',num2str(flag)]);
end;
function [sys,x0,str,ts]=mdlInitializeSizes

sizes = simsizes;
sizes.NumContStates=0; %
sizes.NumDiscStates=0;
sizes.NumOutputs=5;  %Asigna cinco salidas al sistema
sizes.NumInputs=2;   %Asigna dos entradas al sistema
sizes.DirFeedthrough=1;
sizes.NumSampleTimes=1;
sys = simsizes(sizes);
```

```
x0=[];
str=[];
ts=[0.5 0];   %Asigna un tiempo de muestreo de 500 ms

function sys=mdlOutputs(t,x,u)
global x1 y1 angulo angulo2 na angant n ; %Define 7 variables globales
na=u(1);       %Asigna a la variable na el valor de la entrada 1
angant=u(2);   %Asugna a la variable angant el valor de la entrada 2
import TUIO.*; %Inicia protocolo TUIO
tc = TuioClient();
tc.connect();   %Inicia comunicación con el cliente TUIO
for i = 1 : 2   %Inicia un ciclo infinito
    c = tc.getTuioObjects().size();  %cuenta los objetos
    if (c > 0)  %Si existe al menos un objeto
        f   =   tc.getTuioObjects().get(0).getSymbolID();   Obtiene   el
número de identificación del marcador fiducial
        switch f
            case 0 %Si el marcador tiene el número de identificación 0
                x1 = tc.getTuioObjects().get(0).getPosition().getX();
%Obtiene el valor de la posición en el eje x del marcador cero
                y1 = tc.getTuioObjects().get(0).getPosition().getY();
%Otiene el valor de la posición en el eje y del marcador cero
                angulo = tc.getTuioObjects().get(0).getAngle();
%Obtiene el ángulo del marcador cero
        end
    end
    pause(0.1);
end
if ((2*pi-angulo)-angant>5) %Comprobación para saber si se debe restar
una vuelta al angulo absoluto
    n=na-1;
elseif ((2*pi-angulo)-angant<-5) %Comprobación para saber si se debe
sumar una vuelta al ángulo absoluto
    n=na+1;
else
    n=na;
end
angulo2=(2*pi-angulo)+2*n*pi;
tc.disconnect(); %Termina comunicación con el cliente TUIO
sys   =   [(236*x1)-40   (177*(1-y1)-88.5)   angulo2   n   2*pi-angulo]   %De
definen las salidas del sistema
```

Función s de contador de vueltas para ángulo absoluto

```
function [sys,x0,str,ts]=testangulo(t,x,u,flag)
switch flag,
    case 0
        [sys,x0,str,ts]= mdlInitializeSizes;
    case 3
        sys = mdlOutputs(t,x,u);
    case {1,2,4,9}
        sys = [];
    otherwise
        error(['Unhandled flag=',num2str(flag)]);
end;

function [sys,x0,str,ts]=mdlInitializeSizes

sizes = simsizes;
sizes.NumContStates=0;
sizes.NumDiscStates=0;
sizes.NumOutputs=1; %Asigna una salida al sistema
sizes.NumInputs=2;  %Asigna dos entradas al sistema
sizes.DirFeedthrough=1;
sizes.NumSampleTimes=1;

%Load the sys vector with the sizes information.

sys=simsizes(sizes);
x0=[];
str=[];
ts=[0 0]; %No existe tiempo de muestreo, es una función contínua

function sys=mdlOutputs(t,x,u)
global dx dy la bandera ang;  % Declara 5 variables de tipo global
dx = u(1);  %Asigna a la variable dx el valor de la entrada 1
dy = u(2);  %Asigna a la variable dy el valor de la entrada 2
if (dx>0) && (dy>0)
    if (bandera < 3)
        la=la+1;
    else
        la=la;
    end
    ang = atan(dy/dx)+2*la*pi;
    bandera=5;
elseif (dx<0) && (dy>0)
```

```
    ang = pi + atan(dy/dx)+2*la*pi;
    bandera=3;
elseif (dx<0) && (dy<0)
    ang = pi + atan(dy/dx)+2*la*pi;
    bandera=3;
elseif (dx>0) && (dy<0)
    if (bandera > 4)
        la=la-1;
    else
        la=la;
    end
    ang = 2*pi + atan(dy/dx)+2*la*pi;
    bandera=2;
elseif (dx==0) && (dy>0)
    ang = pi/2+2*la*pi;
    bandera=3;
elseif (dx==0) && (dy<0)
    ang = 3*pi/2+2*la*pi;
    bandera=3;
elseif (dy==0) && (dx>0)
   ang = 0+2*la*pi;
   bandera = 3;
else
   ang = pi+2*la*pi;
   bandera=3;
end

sys = [ang];  %Define las salidas del sistema
```

Función s de error del robot

```
function [sys,x0,str,ts]=robotcoordinates(t,x,u,flag)
switch flag,
    case 0
        [sys,x0,str,ts]= mdlInitializeSizes;
    case 3
        sys = mdlOutputs(t,x,u);
    case {1,2,4,9}
        sys = [];
    otherwise
        error(['Unhandled flag=',num2str(flag)]);
end;
```

```
function [sys,x0,str,ts]=mdlInitializeSizes

sizes = simsizes;
sizes.NumContStates=0;
sizes.NumDiscStates=0;
sizes.NumOutputs=3;   %Asigna tres salidas del sistema
sizes.NumInputs=4;    %Asigna cuatro entradas al sistema
sizes.DirFeedthrough=1;
sizes.NumSampleTimes=1;

sys=simsizes(sizes);

x0=[];
str=[];
ts=[0 0]; %No hay tiempo de muestreo, es una función contínua
function sys=mdlOutputs(t,x,u)

e(1)=cos(u(4))*u(1)+sin(u(4))*u(2); %Ddefine el valor de e1
e(2)=-1*sin(u(4))*u(1)+cos(u(4))*u(2); %Define el valor de e2
e(3)=u(3); %Define el valor de e3
sys = [e(1) e(2) e(3)]; %Define las salidas del sistema
```

Diagrama para la simulación del control

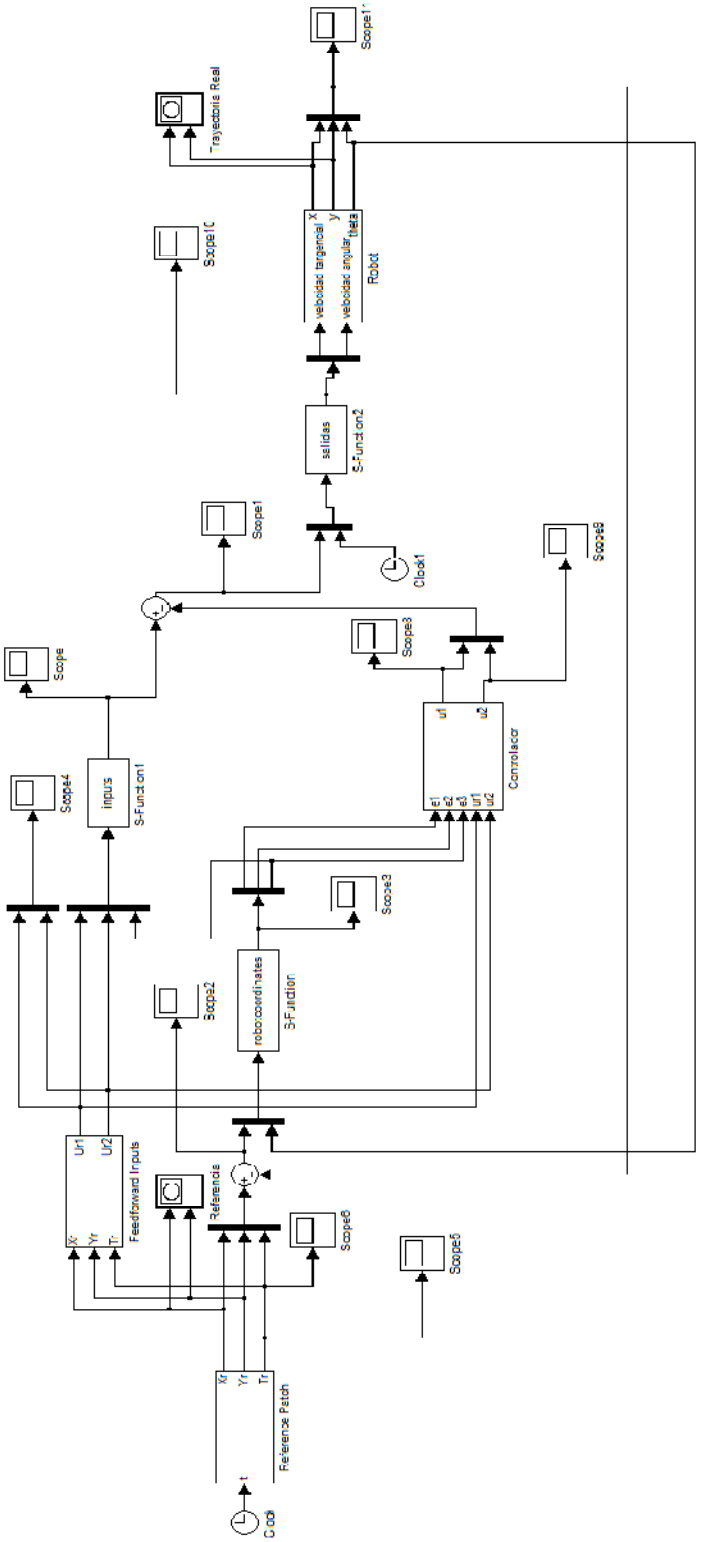
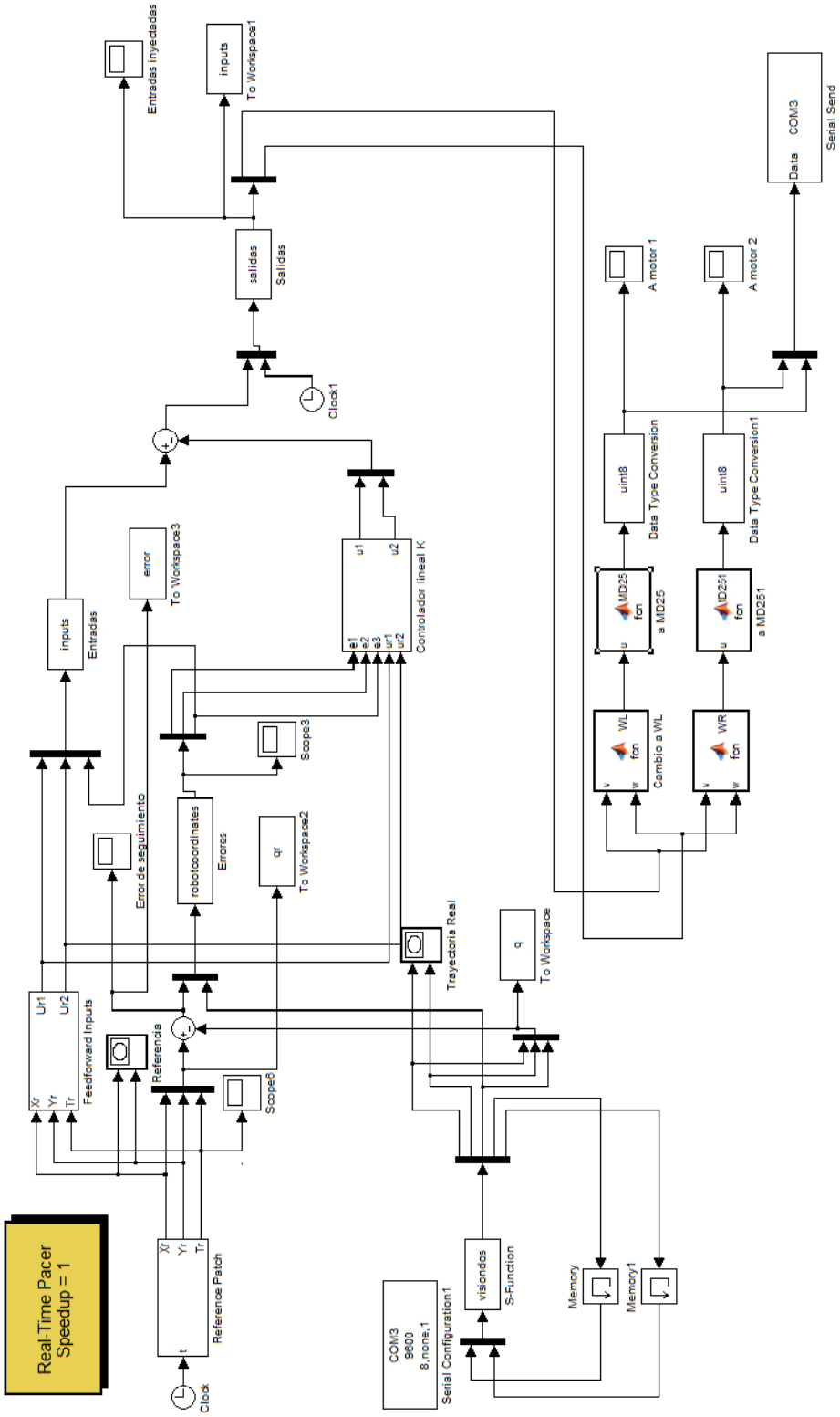


Diagrama para las pruebas experimentales



A2 Código de la tarjeta Arduino Duemilanove y diagrama de conección con el controlador MD25

Código implementado en la tarjeta Arduino Duemilanove para la recepción de datos y envío a la tarjeta MD25

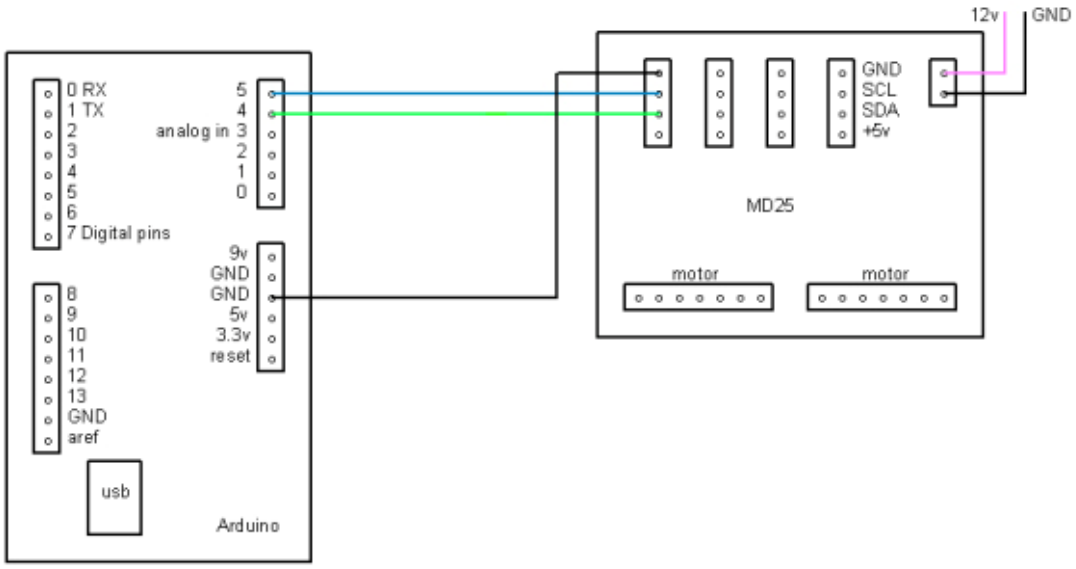
```
#include <Wire.h>
#define md25 0x58          //Dirección MD25
#define motor1 0x01        //Dirección del motor 1
#define motor2 0x00        //Dirección del motor 2
byte m1=128, m2=128, i=0;  //Define 3 bytes y su valor inicial
byte rev[2]={128,128};     //Define 2 bytes y su valor inicial

void setup(){              //Función de configuración
      Serial.begin(9600);  //Inicio de comunicación serial
      Wire.begin();        //Inicio de comunicación i2c
      actuar(m1,m2);       //Para los motores
}
void loop(){               //Función principal
      if(Serial.available()==2){  //Si hay información en el serial
      for(i=0;i<2;i++){
      rev[i]=Serial.read();  //Lee la información y asignala a rev
      }
Serial.flush();  //Desecha la información leída
}
      m1=rev[0];  //Asigna a m1 el valor leido y almacenado en rev[0]
      m2=rev[1];  //Asigna a m2 el valor leido y almacenado en rev[1]

      actuar(m1,m2);  //Llama a la función actuar que envia comandos
de velocidad a las ruedas del robot
}
void actuar(byte v1,byte v2){  //Definición de la función "actuar"
        v1=map(m1,128,255,128,0);
      //Motor 1
      Wire.beginTransmission(md25);//Inicia  comunicación  i2c  con  la
MD25
      Wire.write(motor1);
      Wire.write(v1);  Envía el comando de velocidad en un byte a la
dirección del motor 1
      Wire.endTransmission(); //Termina el envío de datos i2c
      //Motor 2
      Wire.beginTransmission(md25);//Inicia  comunicación  i2c  con  la
MD25
```

```
      Wire.write(motor2);
      Wire.write(v2); //Envía el comando de velocidad en un byte a la
dirección del motor 2
      Wire.endTransmission(); //Termina envio de datos i2c
      }
```

Diagrama de conección del controlador MD25 y la tarjeta Arduino Duemilanove



Referencias

[1] Spong, M.W, Hutchinson, S, Vidyasagar, M, *Robot modeling and control*, Wiley, Nueva Jersey, EEUU, 2006.

[2] Dudek, G., Jenkin, M., *Computational principles of mobile robotics*, Cambridge University Press, Nueva York, EEUU, 2010.

[3] Canudas de Wit, C, Siciliano, B, *Theory of robot control*, Springer, Berlín, Alemania, 1997.

[4] González-Villela, V. J, *A unifying theory on conventional wheeled mobile robots. Research on semiautonomous mobile robots for loosely structured environments focused on transporting mail trolleys*, Loughborough University, Loughborough, Reino Unido, 2006.

[5] Méndez-Sandoval, A, *Experimentación en tiempo real sobre deformación reactiva de trayectorias para robótica móvil.* UNAM. Distrito Federal, México. 2011.

[6] Klancar, G., Matko, D., Blazic, S. *Mobile robot control on a reference path*. Mediterranean Conference on control and automation, Chipre, 2005.

[7] Chen, Chi-Tsong, *Linear system theory and design*, Oxford University Press, Nueva York, EEUU, 1995.

[8] http://www.robot-electronics.co.uk/htm/emg30.htm consultada en febrero de 2013

[9] http://www.robot-electronics.co.uk/htm/md25tech.htm consultada en febrero de 2013

[10] http://arduino.cc/es/Main/arduinoBoardDuemilanove consultada en febrero de 2013

[11] http://www.atmel.com/devices/atmega328.aspx consultada en marzo de 2013

[12] http://reactivision.sourceforge.net/ consultada en marzo de 2013

[13] http://www.tuio.org/ consultada en marzo de 2013

[14] http://www.mathworks.com/products/simulink/ consultada en mayo de 2013

[15] MathWorks, Inc. *Simulink, Dynamic system simulation for MATLAB, Writing s functions*, EEUU, 1998.

[16] http://www.mathworks.com/matlabcentral/fileexchange/29107-real-time-pacer-for-simulink consultada en mayo de 2013

[17] http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module?tab=productdocs&pid=3265#docs consultada en febrero de 2013