UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

APLICACIÓN DE SOFTWARE DE VISIÓN Y SIMULACIÓN EN EL SEGUIMIENTO DE TRAYECTORIAS EN ROBÓTICA MÓVIL

T E S I S

Que para obtener el título de

INGENIERO MECATRÓNICO

P R E S E N T A

SANTIAGO BLACKALLER LEDESMA

DIRECTOR DE TESIS:
DR. VÍCTOR JAVIER GONZÁLEZ VILLELA

MÉXICO D.F. **ENERO 2012**

Jurado Asignado

Presidente:

Secretario:

Vocal:

Primer Suplente:

Segundo Suplente:

Lugar donde se realizó la tesis:

Facultad de Ingeniería, UNAM.

TUTOR DE TESIS:
DR. VÍCTOR JAVIER GONZÁLEZ VILLELA

_____

FIRMA

DEDICATORIA

A mis padres María del Carmen y Santiago por su incondicional apoyo y confianza en todo momento, por ser fuente de inspiración y motivación para lograr alcanzar ésta meta.

A mi hermana Mercedes, por haber compartido conmigo desde la infancia bellos momentos y mi ilusión de estudiar Ingeniería.

A Fernanda por llenar de inmenso amor y alegría mi corazón y por haber compartido conmigo una gran etapa de mi vida.

A Alejandro, Arturo, Cristhian, David, Diego, Eduardo, Emmanuel, Ivan, Jorge, Juan Antonio, Luis Alberto, Mario, Miguel Ángel, Pedro Enrique, Pedro Mariano, Roberto Carlos, Sahel y Víctor, por haber formado parte de mi vida universitaria.

Santiago Blackaller Ledesma
Enero 2012

AGRADECIMIENTOS

A la Universidad Nacional Autónoma de México, por haber sido la cuna de mi formación profesional.

A los profesores de la Facultad de Ingeniería, en especial a los del Departamento de Mecatrónica, por sus enseñanzas, su dedicación y su tiempo.

Al Dr. Víctor Javier González Villela, al Mtro. Octavio Díaz Hernández y al Mtro. Patricio Martínez Zamudio por su motivación, ayuda y confianza en la realización de esta Tesis.

Agradezco en lo que le corresponde a la DGAPA, UNAM, por el apoyo brindado para la realización de este trabajo, a través del proyecto PAPIIT IN115811: "Investigación y desarrollo en sistemas mecatrónicos: robótica móvil, robótica paralela, robótica híbrida y teleoperación (2011-2013)".

ÍNDICE GENERAL

RESUMEN 7

1. INTRODUCCIÓN 8

1.1. Generalidades 8

1.2. Aproximaciones previas 9

1.3. ¿Por qué la “Llanta Virtual y el sistema de visión externo”? 10

1.4. Objetivos 11

1.5. Justificación 11

2. Robot Móvil de Llantas Tipo (1,1) 12

2.1. Generalidades 12

2.2. Definiciones 12

2.2.1. Grado de movilidad (r_m) 12

2.2.2. Grado de manejabilidad (r_s) 12

2.2.3. Grado de maniobrabilidad (r_M) 12

2.2.4. Tipo de Robot Móvil 12

2.3. Representación Esquemática 13

2.4. Ecuaciones que lo modelan 16

2.5. Descripción General del Robot Móvil 17

2.6. Descripción Específica de los elementos que componen al Robot Móvil 18

2.6.1. Motores de corriente directa 18

2.6.2. Tarjeta MD-25 20

2.6.3. Servomotores 21

2.6.4. Microcontrolador Arduino Duemilanove 23

2.6.4.1. Comunicación I^2C entre el Arduino Duemilanove y la tarjeta MD-25 24

2.6.4.2. Generación de señal PWM para el control de posición de los servomotores

2.6.5. Módulos de Radiofrecuencia XBee para comunicación serial inalámbrica..... 28
2.6.5.1. Comunicación..... 28
2.6.5.2. Protocolo Zigbee..... 28
2.6.5.3. Módulos XBee..... 30
2.6.5.4. Configuración de los módulos XBee..... 32
3. El sistema de visión. 35
3.1. Visión..... 35
3.2. Arquitectura general del sistema de visión..... 35
3.3. Descripción de los elementos que conforman el sistema de visión..... 36
3.3.1. Cámara..... 36
3.3.2. Software ReacTIVision y Cliente TUIO..... 37
3.3.3. Integración con Simulink® de MATLAB®..... 39
4. Simulaciones..... 41
4.1. Análisis de los resultados de simulación..... 50
5. Pruebas y Resultados..... 53
5.1. Pruebas con retroalimentación de datos de simulación (Lazo Semi-Cerrado)..... 53
5.2. Pruebas con retroalimentación de datos provenientes del sistema de visión (Lazo-Cerrado)..... 55
5.3. Resultados..... 57
5.3.1. Resultados obtenidos de las pruebas en Lazo Semi-Cerrado..... 57
5.3.2. Resultados obtenidos de las pruebas en Lazo Cerrado..... 59
6. Conclusiones y Trabajo a Futuro..... 69
APÉNDICES..... 72
APÉNDICE A CÓDIGO PROGRAMADO EN EL MICROCONTROLADOR..... 72
APÉNDICE B CÓDIGO DEL BLOQUE DE VISIÓN EN SIMULINK®..... 74
REFERENCIAS..... 77

RESUMEN

Esta tesis presenta el diseño de un plataforma de visión por computadora basada en los programas para computadora ReacTIVision y Simulink® (MATLAB) cuyo objetivo es servir de lazo de retroalimentación de la posición, la orientación y la velocidad de desplazamiento de un robot móvil de llantas independientemente actuadas en su tracción (las traseras) y en su orientación (las delanteras). La información obtenida por el sistema de visión es introducida al controlador previamente diseñado por V.J González Villela y parcialmente adaptado por el autor del presente documento. El controlador calcula, basado en las entradas provistas por el sistema de visión, las posiciones de cada una de las llantas delanteras y las velocidades de las llantas traseras, y las envía a través de comunicación serial inalámbrica utilizando señales de radiofrecuencia Zigbee (X-Bee) al microcontrolador montado sobre el robot móvil, éste comanda directamente las instrucciones de la posición a las llantas delanteras utilizando señales PWM, y envía a través de comunicación serial I^2C las instrucciones que permiten que la tarjeta actuadora de los motores de las llantas traseras establezca la velocidad de cada una de ellas.

Para probar la validez de funcionamiento del sistema dos pruebas diferentes fueron realizadas. La primera utilizando un lazo semi-cerrado y la segunda un lazo cerrado, observándose evidentes mejoras utilizando el sistema de visión.

1. INTRODUCCIÓN

1.1. Generalidades

La robótica móvil es un área de la robótica que se encarga de estudiar a aquellos robots que no tienen ningún eslabón anclado a un medio físico no móvil como podría ser la tierra. Los hay acuáticos, voladores y terrestres. Todos ellos, por la manera en que están configurados, y por la intrínseca relación que guardan con las disciplinas de la ingeniería mecánica, electrónica y el control, son considerados productos inherentemente mecatrónicos [1].

De los robots móviles terrestres, los utilizan llantas para desplazarse, constan en su mayoría de un chasis en el que se encuentran montadas las llantas, ya sean direccionables y/o de tracción, los sensores, los actuadores, los elementos que proveen energía al sistema y los circuitos de control, que normalmente son microcontroladores pre-programados que brindan autonomía al robot, microcontroladores con comunicación hacia una PC o bien una computadora personal montada sobre el robot [1].

Resulta fundamental para la robótica móvil tener control sobre el robot, ya que este desempeña su labor en un ambiente humano, y debe ser capaz de tomar decisiones. El robot puede ser controlado a distancia por un humano, como teleoperación [2], o contar con elementos que le permitan obtener información de su ambiente y le faciliten la toma de decisiones, es decir lo provean de autonomía [2], éstos pueden ser: sensores infrarrojos, sensores ópticos, sensores ultrasónicos, sistemas de posicionamiento global (GPS) [3], u otros sensores que permitan al robot obtener información sobre su posición en alguna determinada área de trabajo.

Toda esta información proveniente de los sensores debe ser procesada utilizando ciertos algoritmos o teorías, de tal manera que el robot sea capaz de controlar los elementos que rigen su cinemática, que normalmente es la orientación y/o velocidad de sus ruedas, para poder desarrollar tareas específicas.

El presente trabajo pretende comprobar físicamente una de estas teorías, desarrollada en 2006 por el Dr. V.J. Gonzalez-Villela [4], utilizando un robot móvil de 4 llantas, las dos frontales independientemente actuadas en su orientación y las dos posteriores en su tracción; y un sistema de sensado externo al robot basado en visión por computadora.

1.2. Aproximaciones previas

El problema de coordinación de llantas, tanto en su orientación como en su velocidad, con el objetivo de seguir una trayectoria establecida utilizando diversos sistemas de sensado, ha sido estudiado anteriormente con diferentes enfoques.

En el estudio realizado por O. Díaz en 2010 [2] se utilizaba una configuración de robot móvil de llantas con la misma distribución utilizada para el presente estudio, con la diferencia de que el sistema de sensado de la posición del robot provenía de una estimación odométrica basada en el giro de las ruedas, el cual era medido con encoders. Esta configuración resultaba útil, pero al haber eventualmente desplazamiento de las ruedas sin movimiento del robot, se incurría en errores que no era posible corregir.

Otros sistemas han sido implementados, en 2008 S.S Mehta et.all [5] desarrollaron un sistema para el desplazamiento de un robot móvil dentro de un invernadero basado en visión, el sistema de visión se encontraba montado sobre el robot y le permitía conocer la orientación y posición del mismo utilizando técnicas de fotogrametría. El inconveniente de ésta técnica es la necesidad de marcas de referencia, dentro del área de trabajo, para que el sistema de visión pueda realizar su función.

En 2010, Maier, D. y Kleiner, A. [3] desarrollaron un sistema para la navegación de robots móviles terrestres basados en GPS, su aportación fundamental es la movilidad que otorgan al robot móvil dentro de ambientes urbanos, donde el GPS podría incurrir en errores por la presencia abundante de construcciones. Este sistema es comparable con un sistema de visión externo al robot, que le indique su

posición dentro de cierta área determinada y le facilite el seguimiento de trayectorias.

Por otra parte, V.J. Gonzalez-Villela desarrolla una teoría unificadora para el análisis cinemático y dinámico de robots móviles [4] que permite coordinar las llantas del robot tanto en su velocidad como en su orientación basado en la teoría de la llanta virtual, para que el robot sea capaz de seguir trayectorias.

Este trabajo utiliza la teoría de [4] aplicada a un robot móvil de llantas, que junto con el desarrollo de un sistema de visión externo al robot, permitirá que éste siga trayectorias predefinidas sin necesitar de sensores internos y utilizando únicamente la retroalimentación por visión y el lazo de control diseñado en 2006 por V.J. Gonzalez-Villela, que fue adaptado por el autor de este documento para su aplicación con un robot móvil físico que conviva con el sistema de visión.

1.3. ¿Por qué la "Llanta Virtual y el sistema de visión externo"?

Al día de hoy, como se pudo apreciar en el apartado anterior, son pocos los que han trabajado experimentalmente con la teoría de la llanta virtual, por lo que se pretende no solamente comprobar la validez de las ecuaciones que la describen, sino evitar los errores que se han presentado con anterioridad al utilizar solamente sensores internos, proponiendo un sistema de visión. Este trabajo desea montar un sistema capaz de sensar externamente variables de suma importancia como son la posición, orientación y velocidad del robot dentro de un área de trabajo, de tal forma que con esta información el controlador sea capaz de corregir la velocidad y orientación de las llantas del robot para seguir de la mejor manera posible una trayectoria previamente establecida.

1.4. Objetivos

- Montar un sistema de experimentación, basado en visión y Simulink®, que permita obtener la posición, orientación y velocidad del robot dentro de un área de trabajo.

- Utilizar la información obtenida del sistema de visión para retroalimentar el lazo de control previamente diseñado y simulado por V.J González-Villela (2006), con la finalidad de que el robot móvil siga una trayectoria previamente programada.

- Comprobar que la teoría de la llanta virtual funciona sobre el robot móvil propuesto

1.5. Justificación

Los resultados de este trabajo son de alto impacto en aplicaciones industriales permitiendo el traslado de materiales u objetos dentro de algún complejo, así como en la industria automotriz, permitiendo generar vehículos con dirección y tracción independiente, pero no solo eso, además capaces de seguir alguna trayectoria previamente establecida, como podría ser el traslado de algún pasajero de un punto a otro utilizando sistemas de navegación como el GPS.

2. Robot Móvil de Llantas Tipo (1,1)

2.1. Generalidades

Un robot móvil se encuentra equipado con motores que son controlados por una computadora o microcontrolador. Cuenta con un chasis rígido y llantas no deformables en las que no hay deslizamiento con respecto al suelo, y que se mueven en un plano horizontal. Cada rueda se mantiene vertical y gira alrededor de un eje horizontal [4]. Con estas consideraciones y algunas definiciones se describirá el robot móvil que se va a utilizar a lo largo de estas pruebas.

2.2. Definiciones

Para poder hacer una mejor descripción del robot móvil, es necesario definir claramente los siguientes conceptos:

2.2.1. Grado de movilidad (r_m)

Se refiere a los grados de libertad instantáneos que un robot puede manipular desde sus entradas, sin orientar ninguna de sus ruedas. [4]

2.2.2. Grado de manejabilidad (r_s)

Se encuentra definido por el número de llantas direccionables centradas que son libres para ser controladas. [4]

2.2.3. Grado de maniobrabilidad (r_M)

El grado de maniobrabilidad está definido como $r_M = r_m + r_s$. Define el total de grados de libertad que el robot puede manipular. [4]

2.2.4. Tipo de Robot Móvil

Como consecuencia de las tres definiciones anteriores, los robots móviles con llantas pueden ser clasificados en cinco configuraciones no singulares, tomando en cuenta el grado de movilidad y de manejabilidad. El grado de maniobrabilidad, por sí mismo, no define el tipo de robot móvil, porque dos robots con el

mismo r_M , pero diferente r_m no son equivalentes. Por esto, para definir el tipo de robot móvil se utilizan el r_m y el r_s, generando una nomenclatura como la siguiente "*robot móvil tipo (r_m,r_s)*"[4]

Las cinco configuraciones se muestran en la Tabla 1.

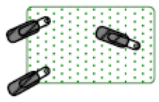
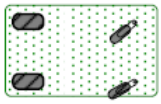
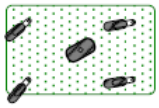
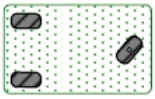
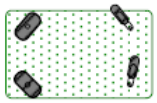
		Tipo (r_m,r_s)				
		(3,0)	(2,0)	(2,1)	(1,1)	(1,2)
Grado	r_m	3	2	2	1	1
	r_s	0	0	1	1	2
	r_M	3	2	3	2	3
Configuración del Robot						
Llantas Motorizadas		 Fijas	 Centradas	 Descentradas		
Llantas No Motorizadas		 Pasivas Descentradas	 Pasivas Centradas			

TABLA 1. Tipos de configuraciones de robots móviles con llantas. Adaptado de González-Villela 2006

2.3. Representación Esquemática

Para lograr un mejor entendimiento del robot móvil en cuestión, resulta fundamental representarlo de manera esquemática, dentro de un sistema de referencia y con todos sus parámetros muy bien definidos. Para ello, partiré de un esquema del robot tipo carro tomado de González-Villela, 2006.

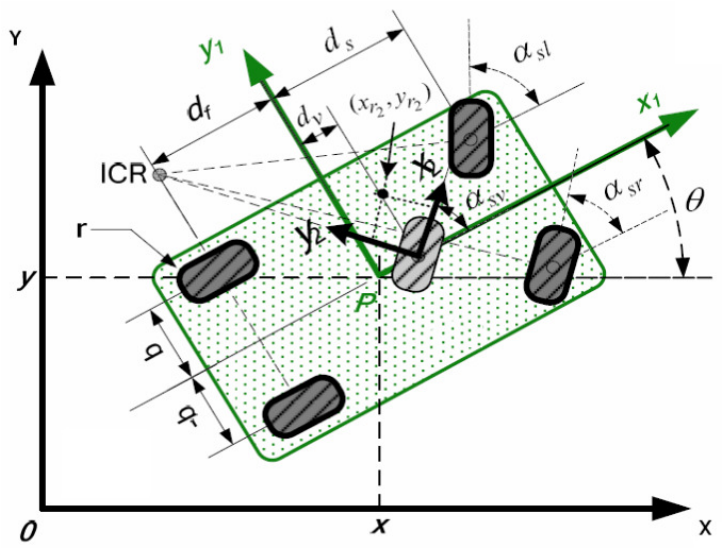


Figura 0 Representación Esquemática del Robot Móvil [4]

En la figura 0 podemos observar, visto desde arriba, al robot móvil. Este se encuentra dentro de un marco de referencia fijo (X,Y). Exactamente al centro de la plataforma del robot, podemos observar el sistema de referencia (x1,y1) el cual tiene como origen al punto P de coordenadas (X,Y). El eje x1 se encuentra alineado con el eje longitudinal de la plataforma del robot, y el y1 es perpendicular a éste. El ángulo que existe entre el sistema de referencia fijo (X,Y) y el móvil (x1,y1) está representado por el ángulo θ. Claramente se pueden apreciar cada una de las llantas que conforman al robot; en la parte trasera, dos llantas centradas fijas motorizadas de forma independiente, en la delantera, dos llantas centradas fijas direccionables independientemente, cada una de ellas tiene asignada un ángulo de giro asociado con su orientación. La de la llanta derecha se representa por la variable α_{sr} y la de la llanta izquierda está representada por α_{sl} . Aproximadamente al centro (separado una distancia "dv" desde el punto P y sobre el eje x1) se puede observar una llanta considerablemente más clara, esta es la ***llanta virtual***, es decir no posee masa ni tiene motor [2].Esta llanta además posee una orientación α_{sv} y exactamente al centro de la llanta, se ubica el sistema de referencia (x2,y2). Con respecto a este sistema se localiza el

punto de control el cual tiene coordenadas (x_{r2},y_{r2}) y será el punto que seguirá a la trayectoria programada. Finalmente, hay variables asociadas a la geometría del robot. El radio de las llantas está definido por la variable "r", la mitad de la distancia que existe entre las llantas traseras se representa por la variable "b", la distancia desde el punto P a lo largo del eje x1, y hasta el eje de las llantas traseras está definida por la variable "d_f", y finalmente, la distancia desde el punto P a lo largo del eje x1 y hasta el eje de las llantas delanteras está definido por la variable "d_s".

Los valores reales, medidos en el robot, de todos estos parámetros se encuentran contenidos en la Tabla 2.

PARÁMETROS CINEMÁTICOS DEL ROBOT MÓVIL		
Símbolo	Descripción	Valor [m]
r	Radio de las llantas	0.03
b	Distancia de las llantas al eje x1	0.0875
d_f	Distancia desde el eje y1 al eje de las llantas traseras	-0.065
d_s	Distancia desde el eje y1 al eje de las llantas delanteras	0.065
d_v	Distancia desde el eje y1 al eje de la llanta virtual	0
x_{r2}	Distancia medida sobre el eje x2	0.001
y_{r2}	Distancia medida sobre el eje y2	0

Tabla 2. Parámetros cinemáticos del robot móvil

El conocimiento de todos estos elementos y parámetros resulta fundamental para el posterior modelado cinemático del robot, y la

parametrización de las velocidades y posiciones de las llantas en función de la velocidad y posición de la llanta virtual.

2.4. Ecuaciones que lo modelan

El proceso de selección de sistemas de referencia, modelado de las llantas del robot, y de cada uno de los elementos que lo componen, incluyendo el concepto de la llanta virtual, fue desarrollado por V.J. González-Villela en 2006 [4]. Al ser uno de los objetivos de esta tesis demostrar experimentalmente la validez de algunas de las ecuaciones presentadas por el mencionado investigador, más no su desarrollo, se procederá a reescribir las ecuaciones que resultaron fundamentales para este proyecto.

La orientación de cada una de las llantas frontales direccionables en función de la orientación de la llanta virtual.

$$\alpha_{sr}(\alpha_{sv}) = \tan^{-1}\left[\frac{-(d_f - d_s)\sin(\alpha_{sv})}{(b\sin(\alpha_{sv}) - (d_f - d_v)\cos(\alpha_{sv}))}\right] \quad (1)$$

$$\alpha_{sl}(\alpha_{sv}) = \tan^{-1}\left[\frac{-(d_f - d_s)\sin(\alpha_{sv})}{-b\sin(\alpha_{sv}) - (d_f - d_v)\cos(\alpha_{sv})}\right] \quad (2)$$

La velocidad de cada una de las llantas traseras fijas en función de la posición y velocidad de la llanta virtual.

$$\dot{\varphi}_{fr}(\alpha_{sv}, \dot{\varphi}_{sv}) = \left[\frac{(d_v - d_f)\cos(\alpha_{sv}) + b\sin(\alpha_{sv})}{(d_v - df)}\right]\dot{\varphi}_{sv} \quad (3)$$

$$\dot{\varphi}_{fr}(\alpha_{sv}, \dot{\varphi}_{sv}) = \left[\frac{(d_v - d_f)\cos(\alpha_{sv}) - b\sin(\alpha_{sv})}{(d_v - df)}\right]\dot{\varphi}_{sv} \quad (4)$$

2.5. Descripción General del Robot Móvil

El robot móvil a utilizar durante el desarrollo de este trabajo, es uno tipo(1,1) el cual cuenta con los siguientes elementos: Un chasis rígido de aluminio conformado por dos placas unidas perpendicularmente. En la parte posterior de una de ellas, la que hace las veces de plataforma, se encuentran las dos llantas motorizadas centradas fijas, las cuales son actuadas de forma independiente utilizando un motor de corriente directa para cada una. El par generado por cada motor es transmitido a su respectiva llanta a través de un sistema de engranes helicoidales. También en la placa base se encuentra colocada la batería, la cual provee de energía a todos los motores y a los circuitos de control. En la parte anterior de la plataforma es donde se encuentra la unión con la otra placa, esta segunda hace las veces de soporte de las llantas frontales, las cuales son centradas direccionables, y de los servomotores que las actúan.

Sobre la plataforma se encuentran atornillados tres espaciadores de bronce, los cuales nos permiten, en un nivel superior, colocar una plataforma de acrílico que sirve de base para el microcontrolador y sus periféricos que permiten la comunicación con la computadora. Haciendo uso de la misma clase de espaciadores y placas de acrílico, se agregan dos niveles más. El inmediato superior contiene el regulador de voltaje para alimentar los servomotores y también es donde está colocada la tarjeta controladora de los motores de corriente directa (todos los elementos que conforman al robot serán detallados posteriormente). Finalmente en la plataforma superior se agrega el perfil de un "*fiducial*", que resulta fundamental para el sistema de visión y que se describirá a profundidad más adelante.

Una foto del robot móvil utilizado se muestra en la Figura 1.

2.6. Descripción Específica de los elementos que componen al Robot Móvil

A continuación se describirá, detalladamente, cada uno de los elementos que conforman al robot móvil y que permiten que se comporte de la manera deseada.

2.6.1. Motores de corriente directa

Cada una de las llantas traseras es actuada con un motor de corriente directa modelo EMG30 [6] mostrado en las Figuras 2 y 3. Estos motores funcionan con un voltaje nominal de 12V, y se encuentran equipados con encoders de cuadratura y una caja reductora con una relación 30:1. Cuentan con un capacitor, como supresor de ruido, entre las líneas de alimentación del motor.

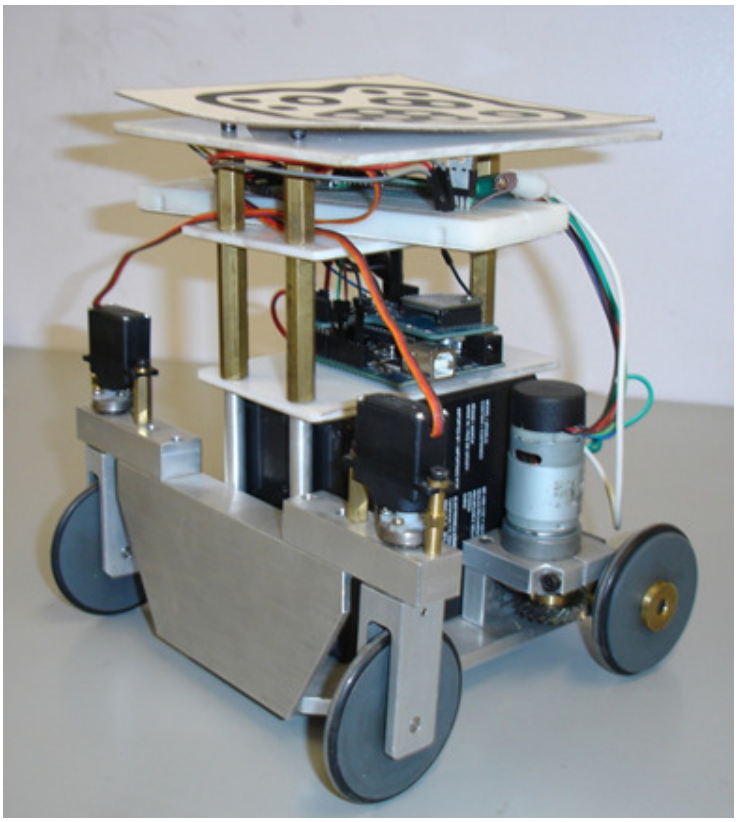


Figura 1. Fotografía del robot móvil utilizado en las pruebas

Figura 2. Motor de corriente directa modelo EMG30 [6]

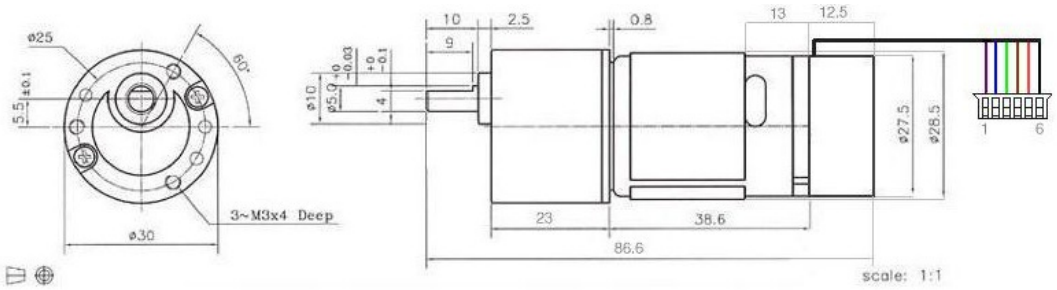


Figura 3. Representación esquemática y dimensiones del motor EMG30 [6]

Las especificaciones del motor se muestran en la Tabla 3

Especificaciones del Motor EMG30	
Voltaje nominal [V]	12
Torque nominal [kg/cm]	1.5
Velocidad nominal[rpm]	170
Corriente nominal[mA]	530
Velocidad sin carga[rpm]	216
Corriente sin carga[mA]	150
Potencia de salida nominal[W]	4.22
Pulsos del encoder por vuelta del eje	360

Tabla 3. Especificaciones del motor EMG30

2.6.2. Tarjeta MD-25

La tarjeta MD25 [7] es un controlador robusto diseñado para manejar dos motores EMG30, a través de comunicación serial o I^2C con un microcontrolador. La figura 4 muestra la tarjeta con sus componentes. Sus características principales son:

- Es capaz de leer los encoders de los motores para conocer la distancia recorrida y la dirección
- Puede controlar dos motores de forma independiente o combinada
- Es capaz de obtener la corriente requerida por cada motor
- Únicamente requiere de una alimentación de 12[V]
- Un regulador de 5[V] montado sobre la tarjeta es capaz de alimentar circuitería externa a una corriente continua de 300[mA]
- Permite establecer diferentes rectas de aceleración, y regular la potencia de cada motor.

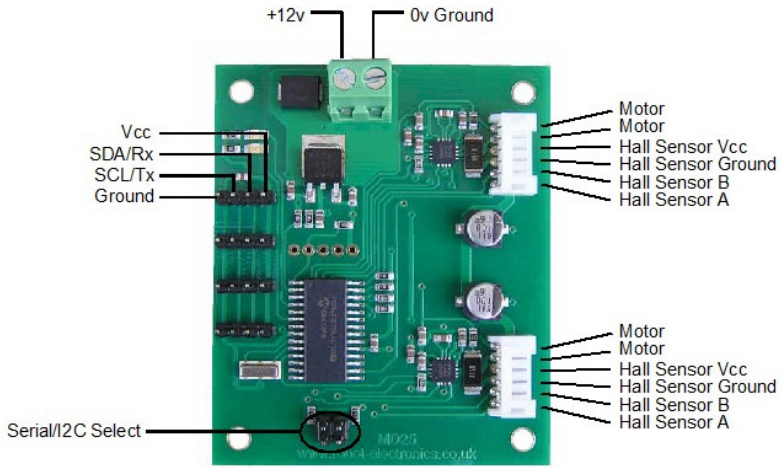


Figura 4. Tarjeta MD-25 y los componentes que la conforman. [7]

Como se menciona anteriormente esta tarjeta permite comunicación serial o I^2C con el microcontrolador. En nuestro caso se decide utilizar la segunda para dejar la serial libre para la comunicación entre el microcontrolador y la computadora. El funcionamiento de la comunicación I^2C se describirá a detalle posteriormente.

2.6.3. Servomotores

Esta clase de motores son dispositivos actuadores con la capacidad de ubicarse y mantenerse estables en alguna posición dentro de su margen de operación. Se conforman por un motor de corriente continua, una caja reductora y un circuito de control [8], y su margen de funcionamiento normalmente es de menos de una vuelta completa.

Los servomotores utilizados en el robot son los "Vigor VS-3" (Figura 5.) que utilizan un circuito de control para ubicar su eje en una posición deseada; el controlador utilizado es proporcional, y funciona de la siguiente manera: La referencia, que es el valor de posición deseado para el motor, se introduce mediante una señal de control PWM [9]. El ancho de pulso de la señal indica el ángulo de posición: una señal con pulsos más anchos(es decir, de mayor duración) ubicará al motor en un ángulo mayor, y viceversa [8]. En principio, un amplificador operacional calcula el error de posición, que es la diferencia entre la referencia y la posición en que se encuentra el motor. Si el servomotor se encuentra en la posición deseada por el usuario, el error será cero, y no habrá movimiento [10]. Para que el amplificador operacional pueda calcular el error de posición, debe restar dos valores de voltaje analógicos. La señal de control PWM es entonces convertida en un valor analógico de voltaje mediante un convertidor de ancho de pulso a voltaje. El valor de posición del motor se obtiene usando un

potenciómetro acoplado mecánicamente a la caja reductora del eje del motor: cuando el motor rote, el potenciómetro también lo hará, variando el voltaje que se introduce al amplificador operacional [8]. Un diagrama del circuito de control empleado es mostrado en la Figura 6.

Figura 5. Servomotor Vigor VS-3

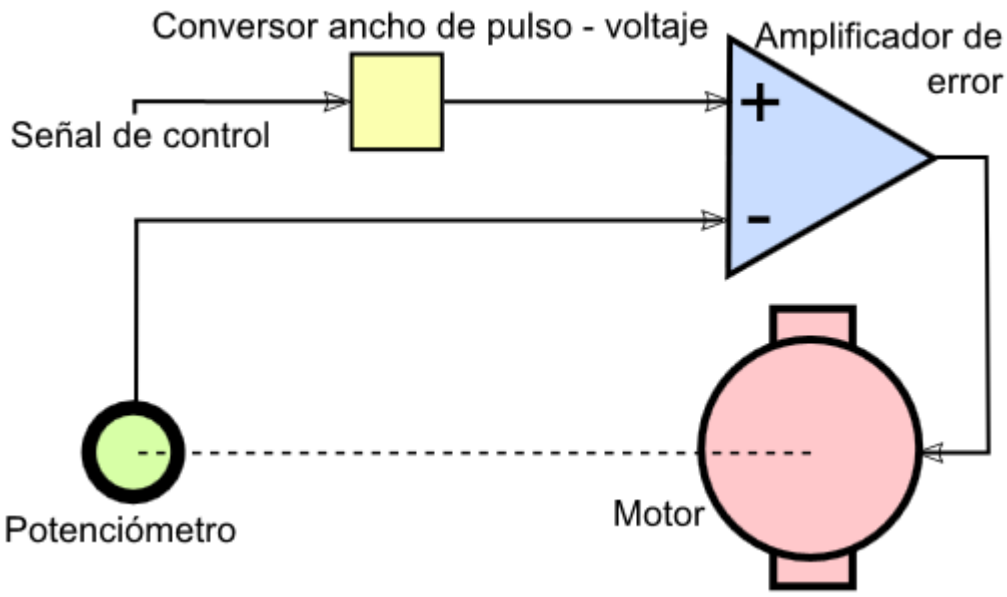


Figura 6. Diagrama del circuito de control implementado en un servomotor [8]

La señal PWM para controlar la posición de los motores es generada en el microcontrolador, es explicado a profundidad más adelante.

2.6.4. Microcontrolador Arduino Duemilanove

El Arduino Duemilanove (Figura 7.) es una placa con un microcontrolador basada en el ATmega328 [11]. Tiene 14 pines con entradas/salidas digitales (seis de las cuales pueden ser usadas como salidas de PWM), 6 entradas analógicas, un cristal oscilador a 16 [MHz], conexión USB, entrada de alimentación, una cabecera ISCP y un botón de reset [12].

La placa Arduino Duemilanove cuenta con las especificaciones mostradas en la Tabla 4.

Especificaciones del Arduino Duemilanove	
Microcontrolador	ATmega328
Voltaje de funcionamiento	5[V]
Voltaje de entrada(recomendado)	7-12[V]
Voltaje de entrada(límite)	6-20[V]
Pines Entradas/Salidas Digitales	14 (6 proporcionan salida PWM)
Pines de entrada analógica	6
Corriente por pin	40[mA]
Corriente por pin a 3.3[V]	50[mA]
Memoria Flash	32[kB] de los cuales 2[kB] son usados por el gestor de arranque
SRAM	2[kB]
EEPROM	1[kB]
Velocidad de reloj	16[MHz]

Tabla 4. Especificaciones del Arduino Duemilanove

Cada uno de los 14 pines digitales del Duemilanove pueden utilizarse como entradas o salidas. Los pines operan a 5[V] y pueden proporcionar una corriente máxima de 40[mA].

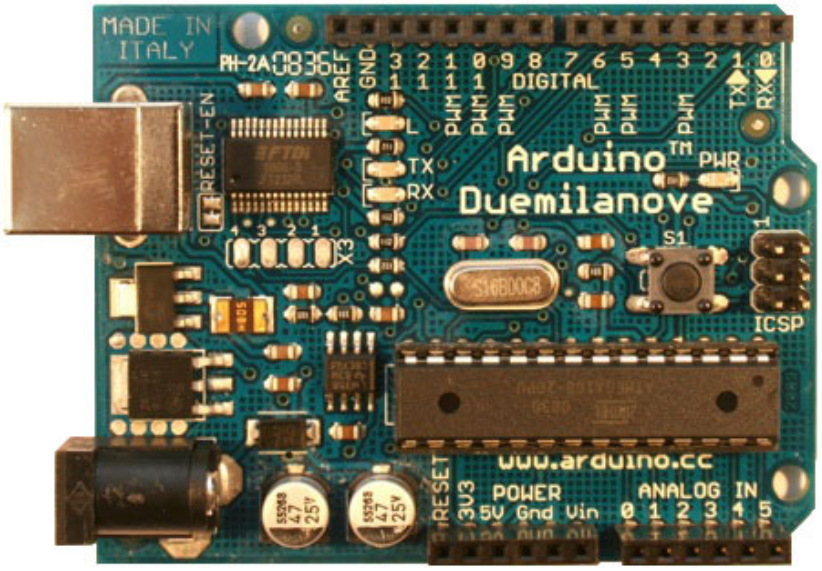


Figura 7. Arduino Duemilanove [12]

2.6.4.1. Comunicación I^2C entre el Arduino Duemilanove y la tarjeta MD-25

Para simplificar la interconexión de dispositivos a un microprocesador, Philips® desarrolló un sencillo bus bidireccional basado en dos hilos por el que se transmiten los datos vía serie [13].

La versión 1.0 se publicó en 1992 y posteriormente en el 2000 fue liberada la versión 2.1, actualmente se encuentra en la versión 3.0 publicada en 2007. En modo estándar es capaz de transmitir a 100[Kbit/s] aunque también permite una mayor velocidad llegando a los 3.4 [Mbit/s] [14].

La línea SDA es la de datos y la SCL es la señal de reloj. Cada dispositivo conectado al bus tiene una dirección que lo identifica, cada uno de ellos puede operar transmitiendo o recibiendo datos. Cuando comienza la conexión se define el rol de cada dispositivo, pueden ser Maestros o Esclavos. El dispositivo maestro es aquel que inicia la conexión, además es el encargado de generar la señal de reloj. La figura 8 muestra una representación de la conexión I^2C entre un microcontrolador y varios dispositivos.

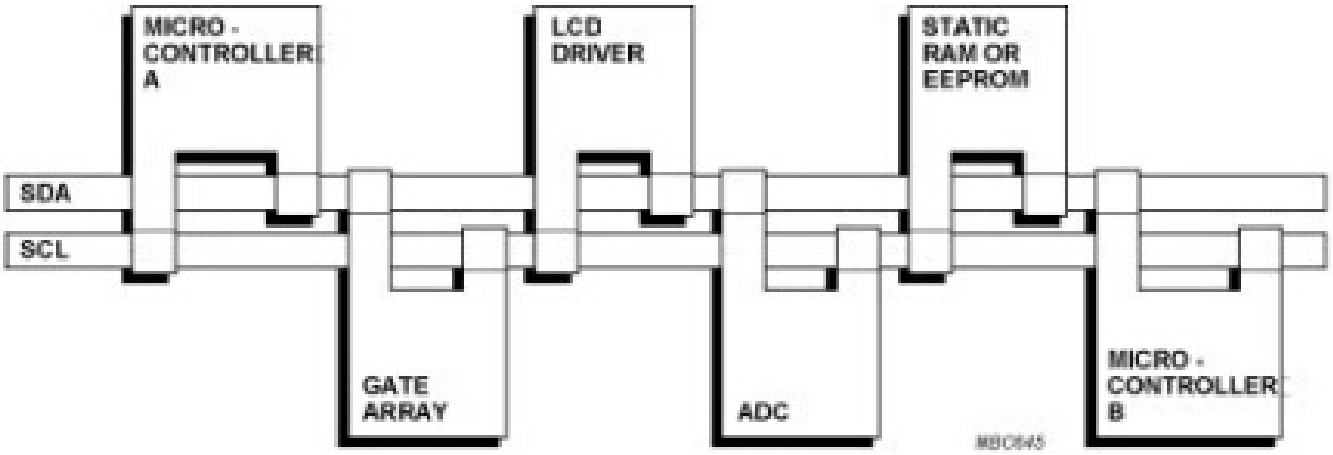


Figura 8. Representación del protocolo de comunicación I^2C [13]

En el problema particular que tenemos, deseamos comunicar a través de éste protocolo el microcontrolador Arduino y la tarjeta controladora de los motores MD-25, para poder establecer la velocidad a la que deben girar cada uno de ellos.

El Arduino Duemilanove cuenta en dos de sus pines analógicos, el 4 y 5, con los canales de comunicación SDA y SCL respectivamente, de la misma manera la tarjeta MD-25 posee ambas líneas por lo que la conexión física entre la tarjeta MD-25 y el Arduino Duemilanove queda como lo indica la figura 9.

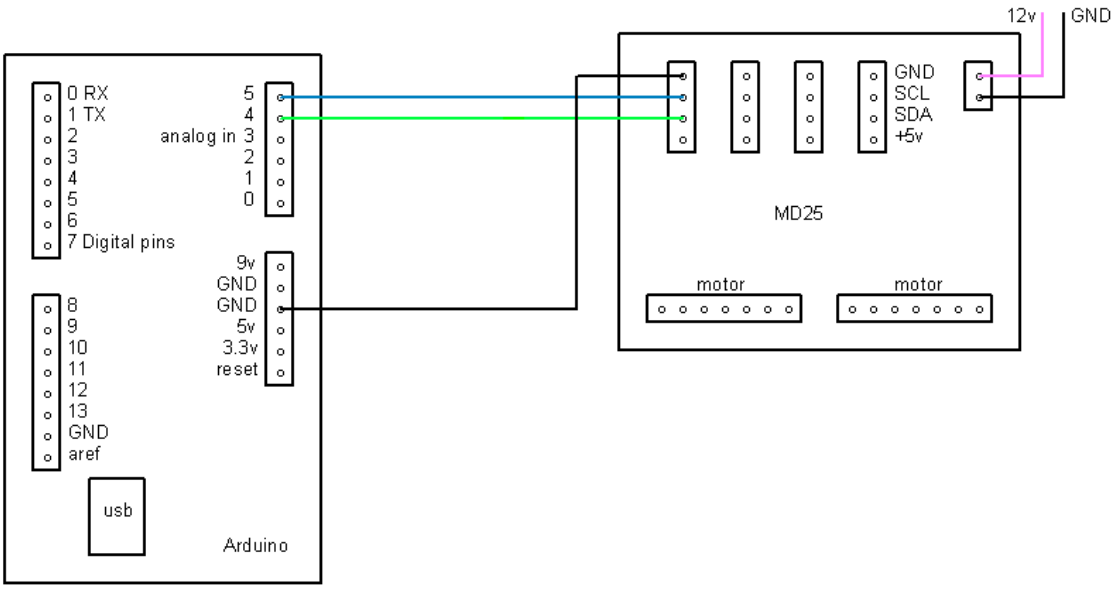


Figura 9. Esquema de la conexión entre el Arduino Duemilanove y la tarjeta MD-25. Adaptado de [15]

Una vez conectados ambos dispositivos, es preciso enviar instrucciones desde el microcontrolador hacia la tarjeta MD-25. Para ello se utilizó la librería "*Wire.h*". Esta librería fue diseñada específicamente para utilizar la comunicación I^2C del Arduino hacia otros dispositivos. El código utilizado se muestra en el Apéndice A de este documento.

2.6.4.2. Generación de señal PWM para el control de posición de los servomotores

La modulación por ancho de pulso (PWM, por sus siglas en inglés) de una señal o fuente de energía es una técnica en la que se ajusta el ciclo de trabajo de una señal periódica (normalmente una señal cuadrada) con la finalidad de controlar la cantidad de energía que se envía hacia algún dispositivo, o para transmitir información a través de un canal de comunicaciones [14].

Como se explicaba anteriormente, la señal de control de los servomotores es una señal cuadrada PWM, la cual posee una frecuencia fija, pero a la que se le modifica la duración de los pulsos. Cada posición del eje del motor está relacionada con una duración del ciclo de trabajo, de tal manera que podemos encontrar una relación directa entre el ciclo de trabajo del motor y la posición del eje este, como se muestra en la figura 10.

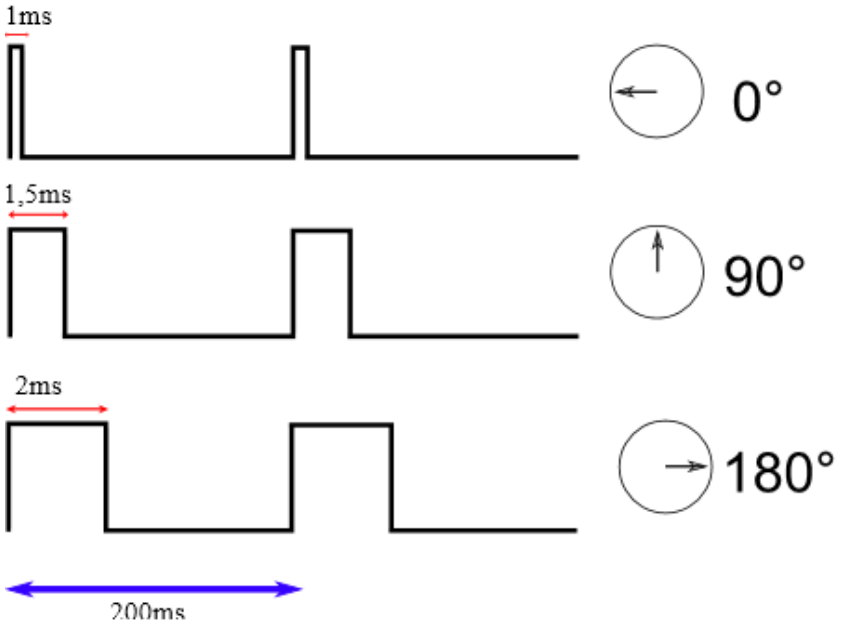


Figura 10. Relación existente entre el ciclo de trabajo de la señal PWM y la posición del servomotor [8]

Conociendo esta relación, y sabiendo que requerimos de la generación de este tipo de señal para controlar los servomotores, se implementa en el microcontrolador un código, basado en la librería "*Servo.h"* la cual nos permite generar en los pines 9 y 10 del Arduino una salida PWM de 8 bits de resolución. La propia librería *Servo.h* contiene comandos para establecer la posición del motor de manera directa al utilizar una serie de líneas de programación. El código utilizado puede ser consultado en el Apéndice A de este documento.

2.6.5. Módulos de Radiofrecuencia XBee para comunicación serial inalámbrica

2.6.5.1. Comunicación

La comunicación es un proceso imprescindible para cualquier robot, por muy poca autonomía (entendiendo por autonomía a la capacidad de no depender más que de sí mismo para realizar alguna tarea) de la que éste goce. Según la Real Academia Española, comunicar es transmitir señales mediante un código común al emisor y receptor. El robot puede ser emisor y receptor indistintamente ya que puede enviar y recibir información a/de otro receptor/emisor, respectivamente.

La comunicación inalámbrica es aquella en la que los extremos de la comunicación no se encuentran unidos por un medio de propagación físico, sino que se utiliza la modulación de ondas electromagnéticas a través del espacio. De acuerdo al órgano regulador de la IEEE existen distintos tipos de protocolos de comunicación, los cuales están estandarizados según la velocidad de transmisión y su alcance [16].

2.6.5.2. Protocolo Zigbee

Zigbee es una plataforma global abierta estandarizada bajo la norma 802.15.4 de la IEEE diseñada para soportar capacidades avanzadas de acoplamiento de enrutados. Las especificaciones de Zigbee son desarrolladas por un creciente consorcio de compañías que conforman la “Alianza Zigbee”. Esta alianza está conformada por más de 300 miembros.

Zigbee define tres diferentes dispositivos: coordinador, router y dispositivo final. Sus características son las siguientes [16].

El coordinador:

- Selecciona el canal y el PAN ID (64-bit y 16-bit) para comenzar la red.
- Puede permitir a routers y dispositivos finales unirse a la red.
- Puede asistir en el enrutado de datos.
- No puede *dormir*. Debe estar siempre activo.

El router:

- Debe unirse a un PAN-Zigbee antes de poder transmitir, recibir o enrutar datos.
- Después de unirse, puede permitir a routers y dispositivos finales a unirse a la red.
- Después de unirse, puede asistir en el enrutado de datos.
- No puede *dormir*. Debe estar siempre activo

El dispositivo final:

- Debe unirse a una Pan-Zigbee antes de poder transmitir o recibir datos.
- No puede permitir a dispositivos unirse a la red.
- Siempre debe transmitir y recibir datos de RF a través de su *pariente*. No puede enrutar datos.

- Puede entrar a niveles bajos de potencia para conservarla y puede ser energizado por baterías.

En una red Zigbee, el coordinador debe seleccionar un PAN ID así como el canal para empezar la red. Después de eso se comporta prácticamente como un router.

2.6.5.3. Módulos XBee

Los módulos XBee (Figura 11.) son módulos de radiofrecuencia que trabajan en la banda de los 2.4 [GHz] con un protocolo de comunicación 802.15.4 (Zigbee) estandarizado por la IEEE. Son fabricados por la empresa Digi®.

Normalmente son utilizados en automatización de casas, sistemas de seguridad, monitoreo de sistemas remotos, aparatos electrodomésticos, etc. Cuentan con un alcance en interiores de hasta 30 [m] y en exteriores de hasta 100 [m] [17]

Figura 11. Módulo de radiofrecuencia XBee [18]

Para este proyecto en particular se utilizan dos módulos XBee, uno de ellos hace las veces de emisor, y se encuentra

conectado, a través de una interfaz USB a la computadora, que lo detecta y configura como un puerto serial. El otro módulo, el receptor, se encuentra acoplado al microcontrolador Arduino a través de un "*shield*" el cual está diseñado *ex profeso* para conectar el XBee con el Arduino, la figura 12 muestra el Xbee montado sobre el *shield*.

Figura 12. XBee montado sobre el *shield* para el microcontrolador Arduino [19]

Estos módulos nos permiten establecer una comunicación inalámbrica entre la computadora y el microcontrolador, lo cual es fundamental para las pruebas. La computadora será quien determine, a través de cálculos precisos, la velocidad y posición de cada una de las llantas del robot, esta información será empaquetada y enviada por medio de comunicación inalámbrica hacia el microcontrolador, quien finalmente decodificará las instrucciones y realizará las acciones pertinentes para posicionar las llantas frontales y para establecer la velocidad de las llantas traseras.

2.6.5.4. Configuración de los módulos XBee

Para que la comunicación entre la computadora y el microcontrolador funcione correctamente, es preciso configurar cada uno de los módulos. Existen algunas formas de hacerlo. Una de ellas es programar cada uno de los XBee a través de una *Hyperterminal* y una interfase serial con un MAX3232 y una serie de comandos llamados AT, pero este método puede resultar sumamente complicado. Al contar nosotros con una interfase USB (figura 13.) podemos utilizar un software distribuido gratuitamente por el fabricante llamado X-CTU para realizar la programación.

Figura 13. Explorador USB para XBee [20]

Como el protocolo de comunicación usado por los módulos XBee es el Zigbee, configuraremos al emisor como "coordinador" y al receptor como "dispositivo final".

Debido a que la finalidad de la comunicación es simplemente sustituir a un cable para la comunicación serial, los configuraremos de forma "transparente" es decir, el dispositivo emisor, enviará únicamente a su receptor la información, y ningún otro dispositivo, aun cuando esté configurado como "dispositivo final" será capaz de recibirla.

Para configurar al coordinador, dentro del programa X-CTU (Figura 14.), seleccionamos la pestaña "*Modem Configuration*", en esta seleccionamos el tipo de modem que es el "XB24-B" y en la casilla de "*Function Set"* seleccionamos "*ZNET 2.5 COORDINATOR AT"* una vez seleccionados estos parámetros procedemos a dar click sobre el botón "*Write"* en la esquina superior izquierda de la ventana y esperamos a que la configuración seleccionada sea "grabada" en el módulo.

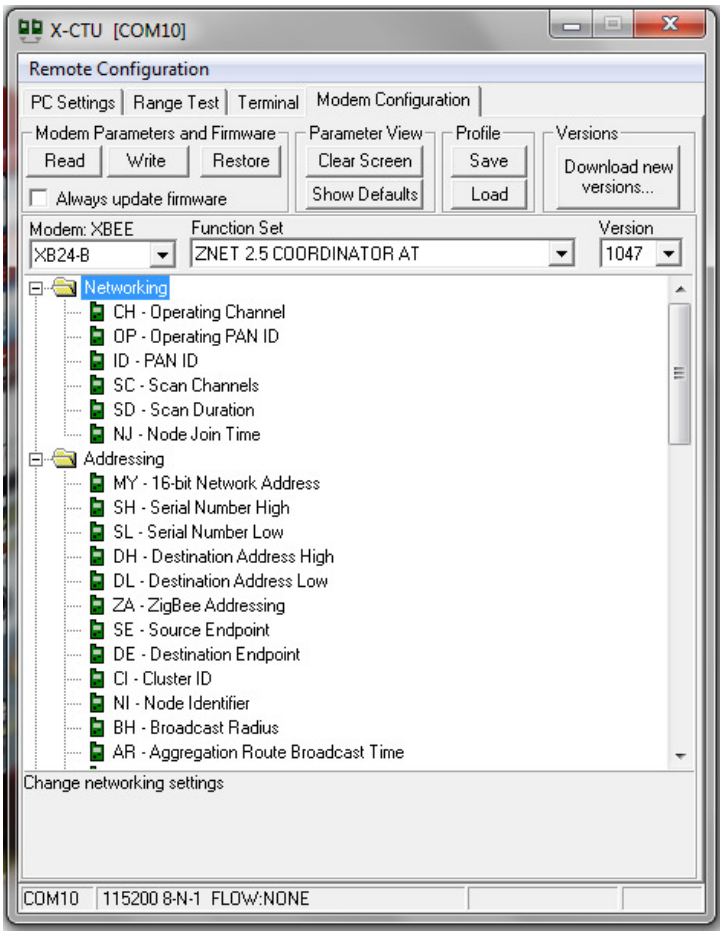


Figura 14. Interfaz del software X-CTU para configuración de módulos XBee

Para configurar el dispositivo final, los pasos son muy parecidos a la configuración del coordinador, únicamente que en la casilla de "*Function Set"* se selecciona "*ZNET 2.5 ROUTER/END DEVICE AT"*

Resulta fundamental, para el buen funcionamiento de la comunicación, que ambos módulos XBee se encuentren en el mismo canal y dentro de la misma PAN ID. Además, el parámetro SH (Serial Number High) del coordinador debe ser el SL (Serial Number Low) del dispositivo final y viceversa.

3. El sistema de visión.

3.1. Visión

De acuerdo a la Real Academia Española (RAE), la visión es la acción y efecto de percibir por los ojos los objetos mediante la luz [21] [22].La visión no es una capacidad exclusiva del ser humano, muchos seres vivos en el planeta la han desarrollado, y el hombre en su afán por lograr generar "ojos artificiales", ideó y fabricó la cámara. Haciendo un símil, y utilizando la definición de la RAE, podríamos afirmar que una cámara es capaz de percibir objetos mediante la luz, tal como un par de ojos lo harían. De esta manera, si enfocamos una cámara sobre un área de trabajo definida, donde el robot móvil se desplaza, ésta nos brindaría información útil sobre la postura del robot, que para fines de esta tesis resulta fundamental. Para lograr esto es necesario contar con un "sistema de visión" capaz de procesar y acondicionar la información captada por la cámara para su posterior utilización, en otras palabras, el sistema de visión cumple la función de sensor externo, capaz de brindarnos información útil e importante sobre el comportamiento del robot dentro de su área de trabajo.

Con esto en mente se definirá la arquitectura general del sistema de visión y cada uno de los elementos que lo conforman.

3.2. Arquitectura general del sistema de visión

El *sistema de visión* (Figura 15.) está conformado por cuatro elementos interconectados entre sí, ya sea físicamente a través de cables, o mediante comunicación entre programas en una computadora.

La cámara de video se encuentra fija atornillada al techo de la habitación donde se realizan las pruebas, su lente enfoca el área de trabajo sobre la cual el robot móvil se desplaza. El robot móvil tiene en su "techo" una ameba *fiducial* (se explicará a detalle más adelante). Las imágenes adquiridas por la cámara son enviadas a través de un cable Firewire

(IEEE 1394) hacia una computadora de escritorio en la que se ejecutan en paralelo los siguientes programas: ReacTIVision, un Cliente TUIO JAVA y Simulink® de MATLAB®.

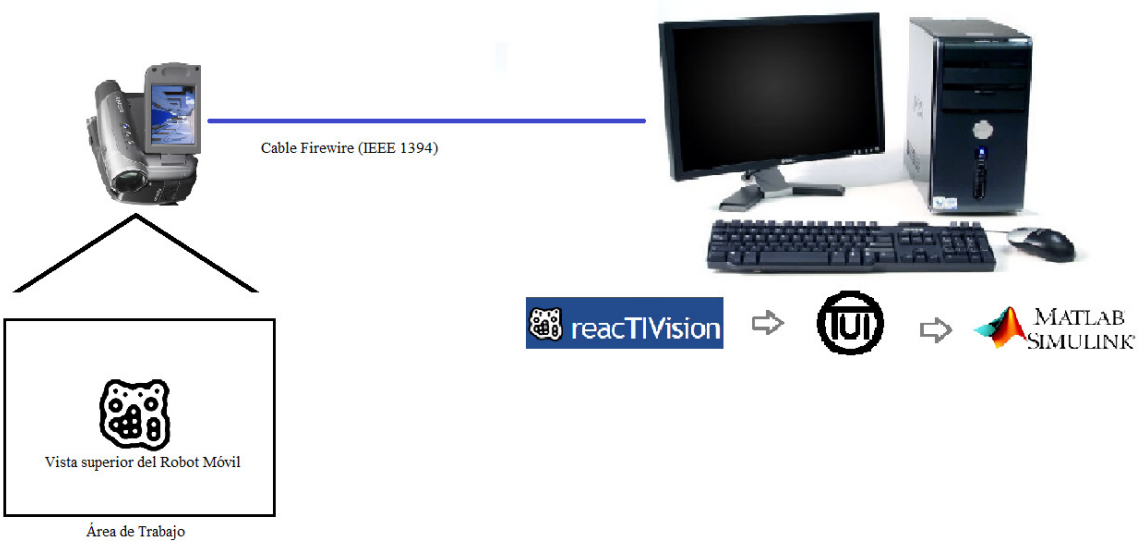


Figura 15. Arquitectura General del Sistema de Visión.

3.3. Descripción de los elementos que conforman el sistema de visión

3.3.1. Cámara

La cámara utilizada para esta trabajo es la Sony® HandyCam® modelo DRC-HC28 (Figura 16.) conectada a la computadora a través de un cable Firewire. La cámara brinda una resolución NTSC de 720 x 480 pixeles. La pérdida de la linealidad asociada a la curvatura del lente de la cámara es despreciable por la cercanía de ésta al área de trabajo que enfoca.

Resulta fundamental para la calidad de la imagen adquirida, que la cámara se encuentre muy bien enfocada sobre la superficie de trabajo, que el tiempo de exposición sea el adecuado a fin de evitar una imagen borrosa o desvanecida y que la superficie de trabajo este muy bien iluminada.

Figura 16. Cámara Sony® HandyCam® modelo DRC-HC28 [23]

3.3.2. Software ReacTIVision y Cliente TUIO

Tradicionalmente el reconocimiento y procesamiento de imágenes por computadora se asocia con un proceso difícil que consume una buena cantidad de los recursos del sistema, además de requerir equipos especializados de alto costo. Sin embargo, en la actualidad existen ciertas herramientas que facilitan la mencionada tarea, reduciendo la cantidad de recursos computacionales requeridos y el costo de los equipos de video. Una de estas herramientas es la plataforma ReacTIVision que se describirá a continuación.

ReacTIVision es una plataforma de visión por computadora, que fue diseñada para interactuar con objetos tangibles, es decir existentes físicamente, en el plano. Su código es abierto y principalmente está diseñada para la construcción de interfaces tangibles bidimensionales con el usuario [16].

La plataforma utiliza marcadores visuales especialmente diseñados (símbolos *fiducial*) que pueden ser asociados a un objeto físico. Los marcadores son reconocidos y localizados por un algoritmo de visión por computadora optimizado para el diseño particular de los marcadores, el cual mejora considerablemente la velocidad y la robustez del proceso de reconocimiento. Los

símbolos *fiducial* (Figura 17.) permiten distinguir una gran cantidad de identidades únicas, como el ángulo de rotación y la velocidad de desplazamiento [16].

Figura 17. Símbolo *fiducial* [24]

El software ReacTIVision envía el "estado" del o los *fiducials* (su posición, su orientación, su velocidad de desplazamiento, etc.) de manera codificada utilizando mensajes TUIO a través del puerto UDP 3333, hacia cualquier cliente TUIO disponible. Estos mensajes transmiten constantemente la presencia, posición y ángulo de los símbolos reconocidos, así como sus parámetros derivados. En el cliente TUIO estos mensajes son decodificados y la información puede ser utilizada por otro software para realizar acciones o tomar decisiones en función de la información obtenida [24].

En el caso específico de este trabajo, se eligió el cliente TUIO programado en JAVA, porque nuestra intención era lograr introducir toda la información obtenida del *fiducial* a Simulink® de MATLAB®, el cual es capaz de procesar código escrito en este lenguaje de programación.

3.3.3. Integración con Simulink® de MATLAB®

Simulink® de MATLAB® es un ambiente para simulación y diseño basado en modelos, para sistemas dinámicos y embebidos. Provee un ambiente gráfico interactivo y un conjunto de librerías de bloques personalizables que permiten al usuario diseñar, simular, implementar y probar una gran variedad de sistemas variantes en el tiempo, incluyendo comunicaciones, control, procesamiento de señales, video e imágenes [25].

Para fines de este proyecto se desea introducir la información codificada en mensajes TUIO, proveniente de ReacTIVision, a Simulink®. Para ello se utilizó un Cliente TUIO Java. Este cliente permite, mediante un código escrito en una función S y asignado a un bloque "función" en Simulink®, decodificar el mensaje y obtener los atributos del fiducial (su posición, orientación, velocidad, etc.) para su posterior utilización dentro del entorno de Simulink®.

El código está basado en instrucciones previamente establecidas por el cliente TUIO para la decodificación del mensaje. La estructura del código sigue la sintaxis para Java (programación orientada a objetos). En principio se debe crear un "objeto", posteriormente se pueden obtener los atributos de ese objeto y asignárselos a variables previamente definidas. El ente objeto en la programación está ligado con un fiducial, y los atributos, son precisamente la información que nos interesa como su posición y su orientación.

Una vez escrito el código que nos permite obtener la información valiosa, es preciso introducirlo en un bloque de Simulink®, porque es en esta plataforma donde se tienen las simulaciones realizadas previamente por el Dr. V.J. González-Villela y que se desean llevar

a la vida real con el robot móvil. Para lograr esto, y aprovechando la capacidad de MATLAB® de ejecutar nativamente código de Java, se introdujo el código previamente obtenido a una función S, la cual cuenta con una definición de entradas-salidas, temporizadores y bloques de ejecución. Una vez obtenido el código de la función S, se ligó a un bloque de "función" en Simulink® el cual cada vez que es llamado para su ejecución, corre el código que tiene ligado. De esta manera, en cada iteración de ejecución del código, el bloque nos arroja a través de su salida los atributos que nosotros deseamos saber del *fiducial* que se encuentra dentro de nuestra área de trabajo.

Precisamente por ello el fiducial se encuentra en el "techo" del robot móvil, para que se desplace junto con el robot, y el sistema de visión al darnos la posición, orientación y velocidad del *fiducial*, indirectamente nos estará brindando la posición, orientación y velocidad del robot móvil. De esta manera habremos cumplido el objetivo de generar un sistema externo capaz de sensar la postura del robot móvil, y cuya información será utilizada para las pruebas de este trabajo.

4. Simulaciones

Como se ha mencionado con anticipación, el presente proyecto parte de un conjunto de ecuaciones que modelan a un robot móvil de llantas en función de la llanta virtual. Este trabajo desarrollado en 2006 por González-Villela, fue acompañado por simulaciones desarrolladas en Simulink® las cuales demuestran, virtualmente, la validez de las ecuaciones y la teoría que las sustenta.

El presente trabajo tiene como uno de sus objetivos demostrar físicamente, es decir aplicado sobre un robot móvil de llantas real, la validez de esas ecuaciones, y es partiendo de las mencionadas simulaciones que se pretende dar solución a este objetivo.

Las simulaciones, como su nombre lo indica, pretenden simular de la manera más realista posible el comportamiento de un robot móvil el cual se desplaza dentro de un área de trabajo y al que se le comanda la instrucción de seguir una trayectoria predefinida. Para ello el mencionado investigador introdujo dentro del ambiente de Simulink®, utilizando bloques de programación y bloques de función, los diversos modelos que componían al sistema. En primer lugar se definen los parámetros que definen la trayectoria que seguirá el robot, en este caso se trata de una trayectoria senoidal. Una vez definidos, se introducen a una función pre-programada que calcula, para cada iteración el par ordenado (X,Y) de la trayectoria. También se definen las condiciones iniciales del robot, es decir su posición (X,Y) y su orientación θ iniciales. (Figura 18.) Toda esta información inicial es utilizada para la primera iteración de la simulación.

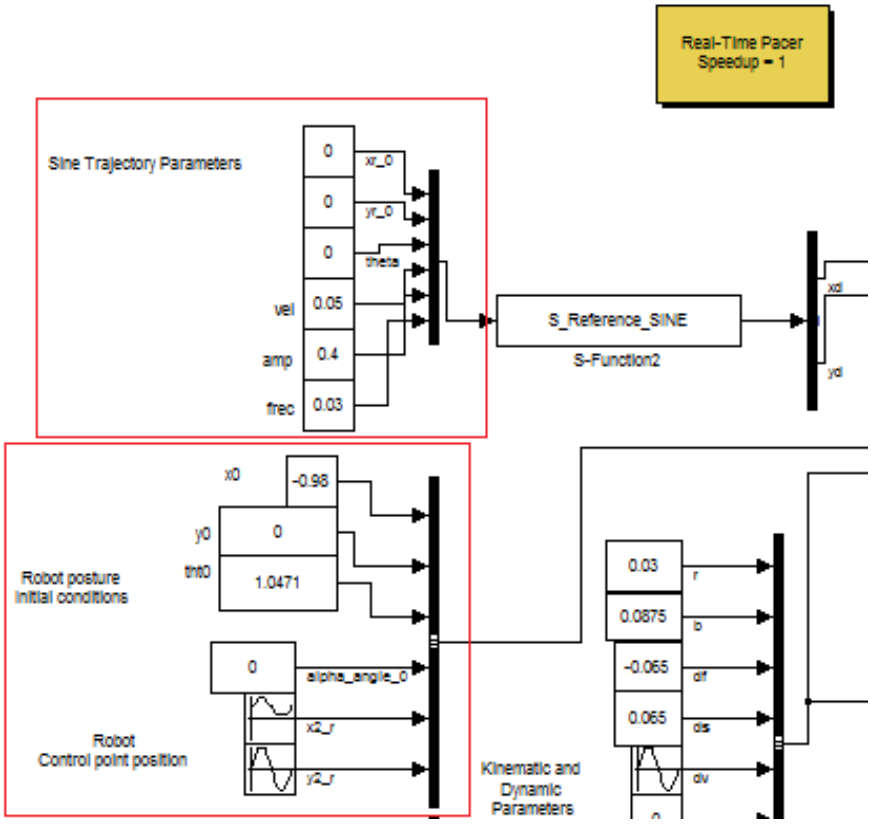


Figura 18. Definición de parámetros de trayectoria y condiciones iniciales (en rojo).

Los datos X y Y de la trayectoria programada se introducen posteriormente a un controlador (Figura 19.), el cual calcula dos funciones de error (una para X y otra para Y) en función de la posición "real" del robot y de la posición deseada (que es la de la trayectoria). Este error es posteriormente derivado y multiplicado por una ganancia y a la vez es sumado con una compensación proporcional del mismo error. Al ser operadas cada una de las funciones de error, se obtienen dos parámetros que representan las aceleraciones, tanto en X como en Y que posteriormente son alimentadas a una linealización entradas-salidas.

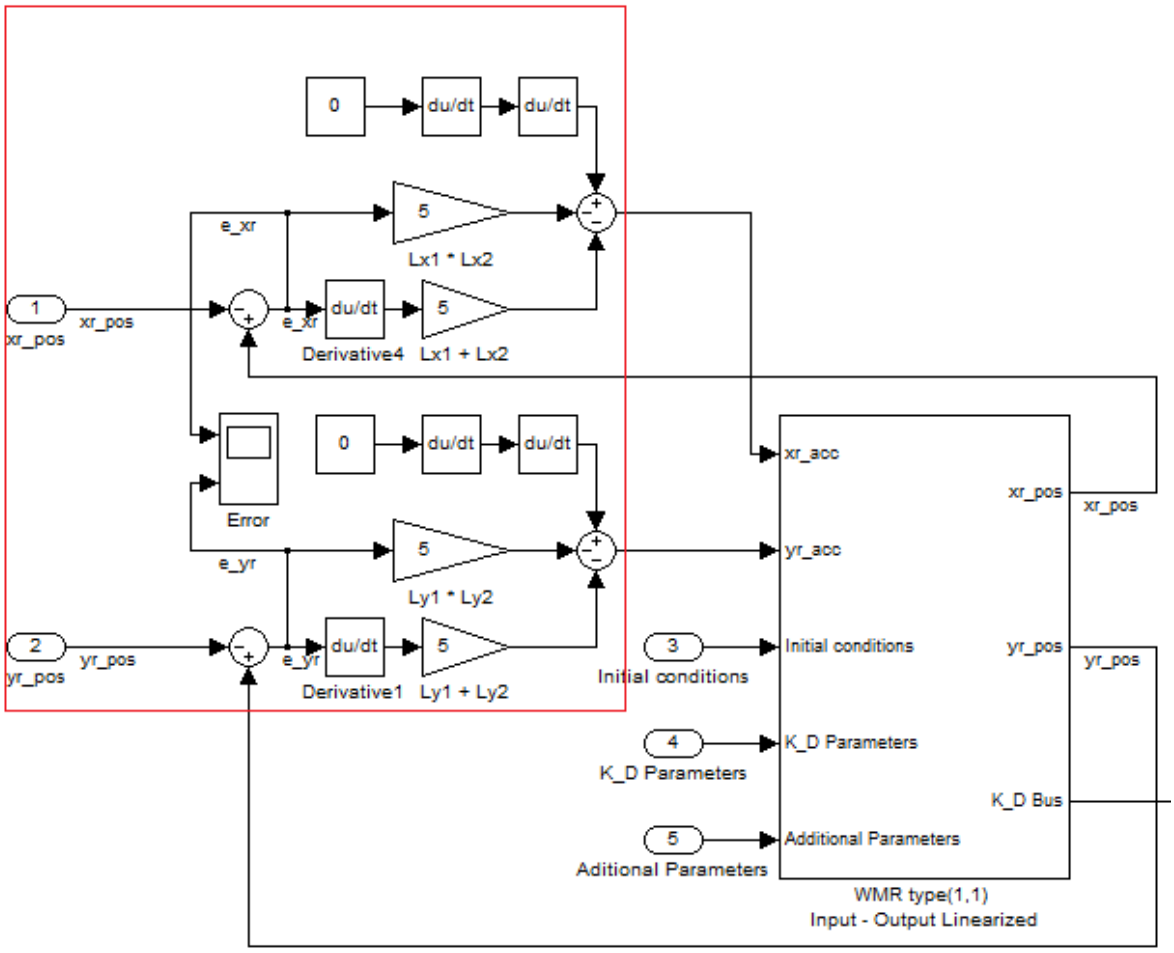


Figura 19. Controlador (en rojo) para obtener las aceleraciones en X y en Y

Esta linealización (Figura 20.) arroja dos aceleraciones, la primera de ellas es la aceleración en la dirección de la llanta virtual, y la segunda es la aceleración angular de la llanta virtual. Ambos parámetros son ahora introducidos a un modelo que representa tanto la parte cinemática (Figura 21.) como la dinámica (Figura 22.) de un robot móvil de llantas tipo (1,1). Este modelo es una representación, basada en ecuaciones del robot móvil, y es una de las partes más importantes de la simulación porque es aquí donde se recrea el comportamiento del robot móvil. Al recibir este modelo las aceleraciones, son integradas numéricamente para obtener velocidades, ambas son introducidas a la representación cinemática del robot móvil, el cual calcula los valores “siguientes” de su posición, orientación, velocidades

tanto lineales como angulares, así como la orientación y velocidad de la llanta virtual. Podríamos entender al bloque del modelo cinemático del robot móvil como un estimador de los valores siguientes en función de los valores presentes, la ventaja fundamental es que el estimador estaría representado por todas las ecuaciones que modelan al robot móvil, haciendo que esta "estimación" sea exacta.

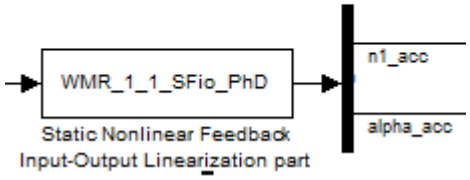


Figura 20. Bloque de la linealización entradas-salidas.

Una vez que se han calculado los nuevos valores que describen al robot móvil son "regresados" al controlador inicial, para calcular nuevamente el error existente entre la nueva posición deseada y la nueva posición del robot. Este proceso se repite iterativamente, dentro de un intervalo de tiempo definido, para que el robot se desplace de un punto A a uno B a lo largo de la trayectoria programada.

Para dejar más en claro lo que realiza la simulación, una vez explicado su funcionamiento, se mostrarán los resultados de la misma. Para la prueba se programó una trayectoria senoidal cuyos parámetros se detallan en la Tabla 5. Se observó como el robot móvil, representado por un bloque color "cyan" en el que se aprecian las llantas delanteras, las traseras, así como la virtual en color blanco, se desplaza dentro del área de trabajo siguiendo la trayectoria programada.

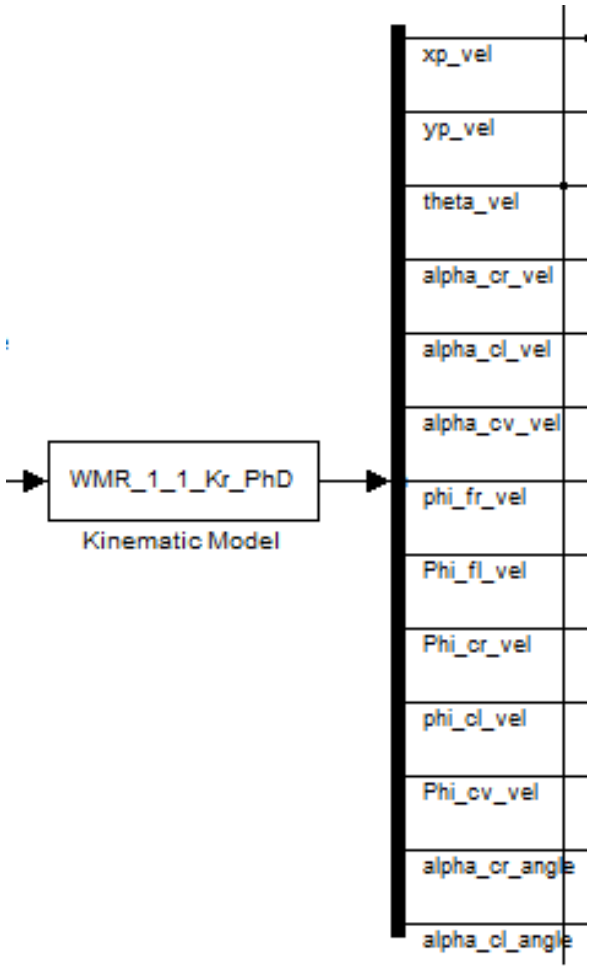


Figura 21. Bloque que representa la parte cinemática del robot móvil de llantas

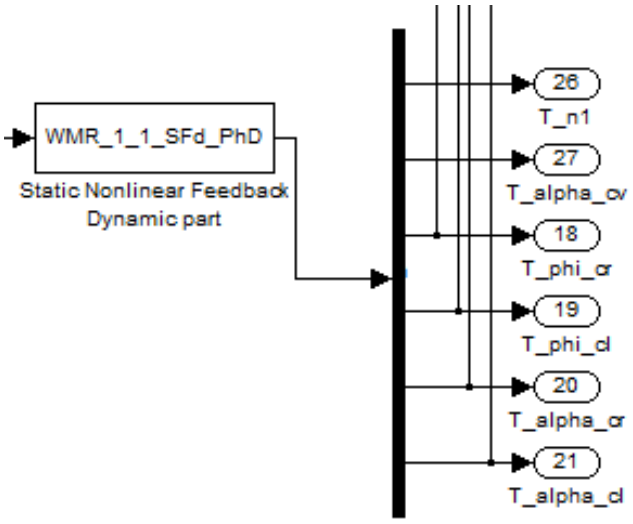


Figura 22. Bloque que representa la parte dinámica del robot móvil de llantas

PARÁMETROS DE LA TRAYECTORIA SENOIDAL	
Velocidad de desplazamiento en el eje X	0.062 [m/s]
Amplitud	0.45 [m]
Frecuencia	0.028 [Hz]

Tabla 5. Parámetros de la trayectoria senoidal a seguir por el robot móvil

Utilizando los valores de la tabla 5, las ecuaciones de la trayectoria (tanto en X como en Y) parametrizadas en función del tiempo resultaron ser las siguientes:

$$x_{ref} = 0.062 \cdot t - 0.98 \quad (5)$$

$$y_{ref} = 0.45 \cdot \sin(0.056\pi \cdot t) \quad (6)$$

Finalmente, antes de mostrar los resultados resulta conveniente retomar el nombre de las variables que fueron descritas en el capítulo 2 de este documente para facilitar su comprensión. La relación entre la nomenclatura y su significado se muestran en la Tabla 6.

Variables utilizadas en los resultados y su definición	
x_r	Valor sobre el eje X del punto de referencia del robot móvil
y_r	Valor sobre el eje Y del punto de referencia del robot móvil
x_d	Valor sobre el eje X de la trayectoria de referencia
y_d	Valor sobre el eje de la trayectoria de referencia
θ	Ángulo existente entre la plataforma del robot móvil y el sistema de referencia(X,Y)
α_{cv}	Ángulo de orientación de la llanta virtual
α_{cr}	Ángulo de orientación de la llanta direccionable derecha
α_{cl}	Ángulo de orientación de la llanta direccionable izquierda
e_{xr}	Error en X entre la posición del punto de control del robot y la trayectoria
e_{yr}	Error en Y entre la posición del punto de control del robot y la trayectoria

Tabla 6. Variables utilizadas en los resultados y su definición

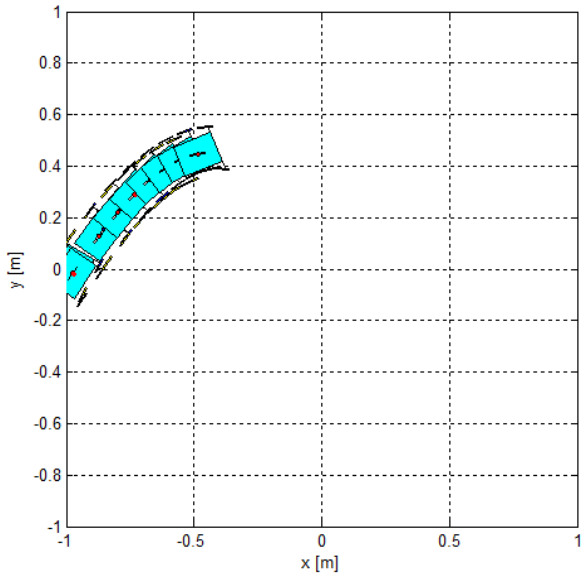


Figura 23. Desplazamiento del Robot Móvil sobre la trayectoria senoidal a los 8 [s] de haber iniciado el movimiento.

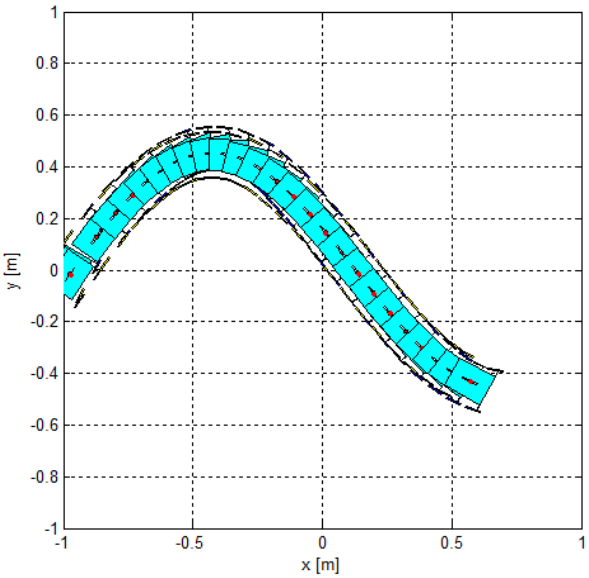


Figura 24. Desplazamiento del Robot Móvil sobre la trayectoria senoidal a los 26 [s] de haber iniciado el movimiento.

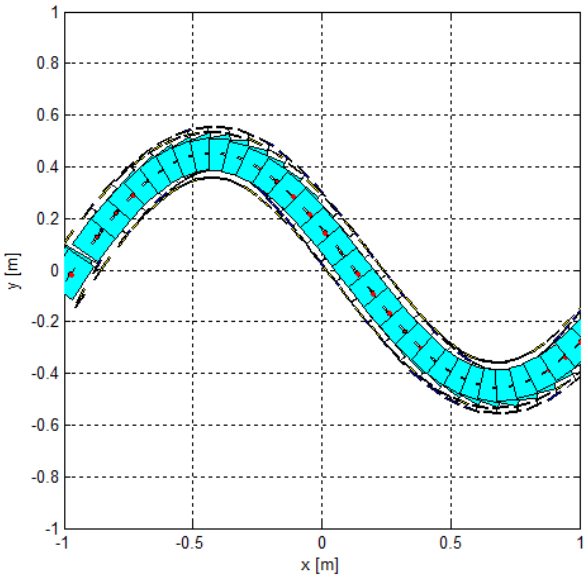


Figura 25. Desplazamiento del Robot Móvil sobre la trayectoria senoidal a los 35 [s] de haber iniciado el movimiento.

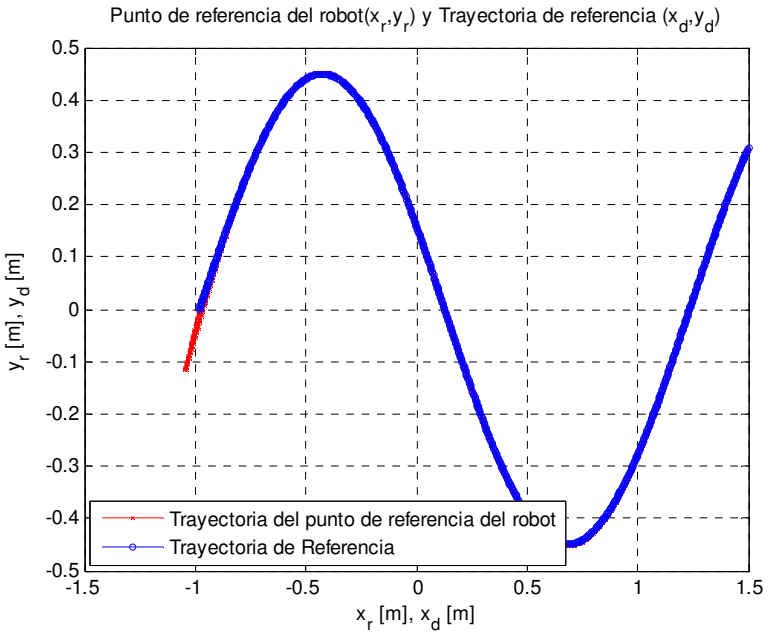


Figura 26. Trayectoria del robot móvil y Trayectoria de Referencia

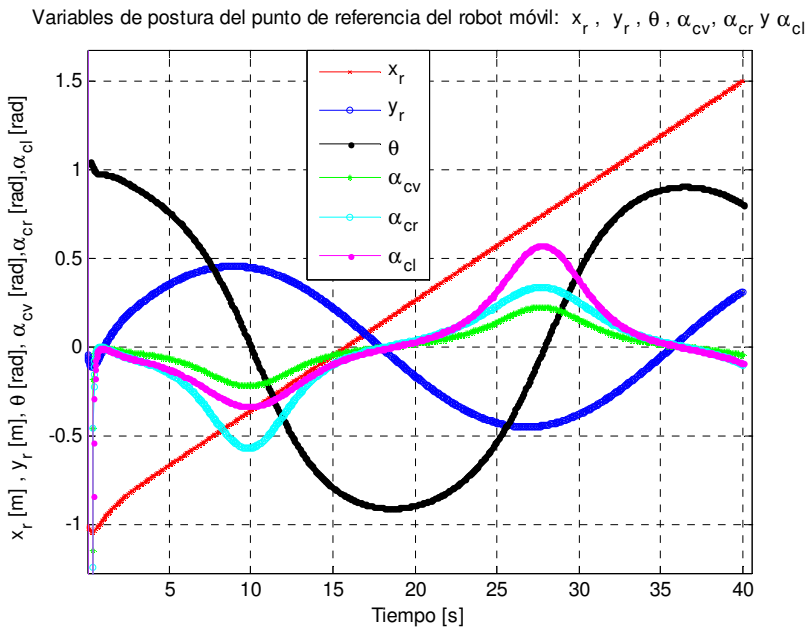


Figura 27. Variables de postura del punto de referencia del robot móvil

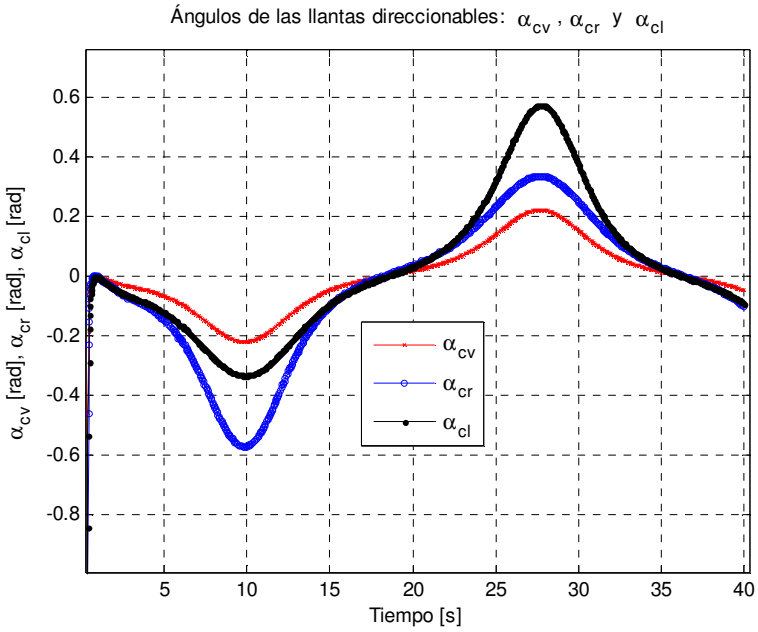


Figura 28. Ángulos de las llantas direccionables del robot móvil

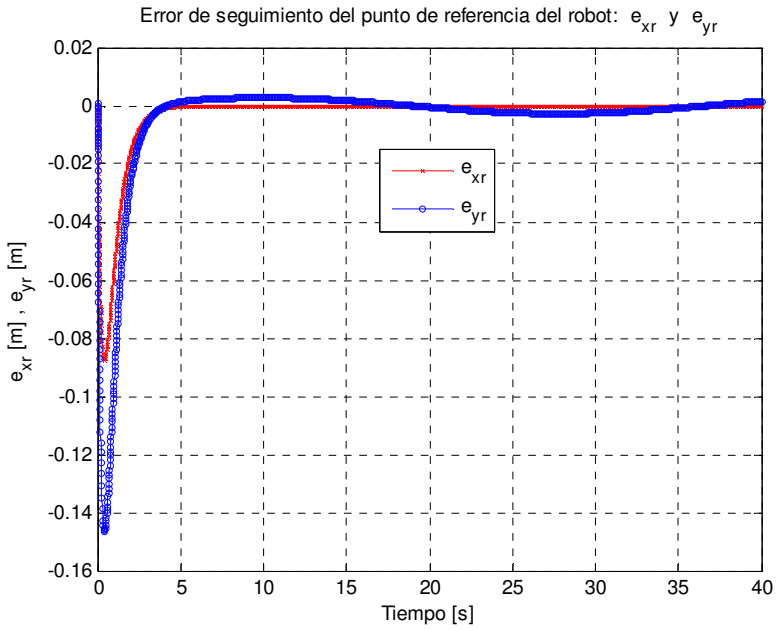


Figura 29. Error de seguimiento del punto de referencia del robot móvil

4.1. Análisis de los resultados de simulación

Para dejar más en claro el contenido de las gráficas arriba mostradas, se interpretará su contenido. En la figura 26 se pueden apreciar dos curvas, la roja es la trayectoria del punto de control del robot, y la azul muestra la trayectoria senoidal que se desea que siga el robot y que resulta así por las ecuaciones (5) y (6). La curva roja inicia en un lugar diferente a la azul por las condiciones iniciales de posición del robot móvil, pero se puede observar claramente cómo es que esta desaparece debajo de la azul, lo que indica que el seguimiento es prácticamente perfecto.

La figura 27 muestra 6 variables sumamente representativas del robot móvil. La primera de ellas es la trayectoria que sigue el robot a lo largo del eje X, se observa como inicia en una posición de -1 [m] (con respecto al sistema de referencia mostrado en las figuras 23, 24 y 25) y termina en 1.5 [m], lo que nos confirma que el robot nunca va hacia atrás, es

decir a lo largo de su desplazamiento mientras sigue la trayectoria únicamente va avanzando de la izquierda y hacia la derecha del área de trabajo. La siguiente curva que se muestra es la trayectoria que sigue el robot a lo largo del eje Y, es claro el perfil de una senoidal con una amplitud de 0.45 [m] y un periodo de 35 [s] aproximadamente, ambos datos coinciden con la ecuación (6). Posteriormente se muestra, de color negro, la orientación del robot móvil. Esta curva refleja cómo va rotando el robot con respecto a un eje perpendicular al área de trabajo y que lo atraviesa en su centro (punto P), es claro el movimiento de vaivén, que refleja perfectamente el comportamiento del robot. Las últimas tres curvas mostradas, están íntimamente relacionadas entre sí y muestran la rotación de la llanta virtual, de la llanta orientable derecha y de la llanta orientable izquierda. Se observa como la orientación de las llantas es independiente, en ningún momento tienen el mismo valor y mientras el robot gira hacia un lado las otras dos también lo hacen en el mismo sentido lo que refleja que la coordinación de las llantas es correcta. La figura 28, muestra nuevamente estas 3 curvas de manera aislada para que resulte más sencillo visualizarlas.

Finalmente la figura 29 muestra el error de seguimiento, que resulta ser la diferencia entre la trayectoria descrita por el robot móvil y la trayectoria de referencia. Se representan dos curvas, la primera describe el error a lo largo del eje X y la segunda a lo largo del eje Y. Se observa durante los primeros 5 segundos de trayectoria que el error es considerablemente grande aproximadamente de 14 [cm] en el eje Y y 8 [cm] en el eje X, esto es debido a que el robot no se encuentra posicionado donde inicia la trayectoria y por tanto el error es máximo. Mientras pasan los segundos podemos observar como el error va disminuyendo en forma gradual, hasta llegar a ser técnicamente cero a lo largo del eje X, y aproximadamente 3 [mm] a lo largo del eje Y. lo que

nos indica que la trayectoria establecida es seguida con suma precisión por el robot móvil.

5. Pruebas y Resultados.

Una vez explicado el procedimiento utilizado para las simulaciones, resulta más sencillo explicar el utilizado para llevar a cabo las pruebas físicas. Fue fundamental la arquitectura mencionada en el capítulo III la cual se recapitula brevemente a continuación.

El robot móvil de llantas, que porta una ameba "*fiducial*" en su parte superior, se coloca sobre una superficie lisa y plana la cual es enfocada por una cámara suspendida en el techo de la habitación. La imagen captada por la cámara es procesada por el software "ReacTIVision" el cual detecta el "*fiducial"* montado sobre el robot, y calcula la posición (X,Y) y la orientación θ de la ameba y por consiguiente del robot. Esta información esta codificada y debe de ser decodificada para poder ser "entendida" por algún otro programa. Para ello se utiliza un Cliente TUIO programado en JAVA el cual decodifica e introduce en Simulink® de MATLAB® la información de posición y orientación del robot. Una vez estando la información dentro de Simulink® y utilizando el mismo código de las simulaciones se calcula la orientación de cada una de las llantas frontales y la velocidad de las llantas traseras. Esta información es concatenada dentro de un bus de datos y enviada a través del puerto serie, utilizando comunicación inalámbrica mediante módulos XBee. La información es recibida por el microcontrolador del robot, quien finalmente comanda a cada uno de los actuadores su posición o su velocidad, logrado que el robot se desplace. Este ciclo se repite iterativamente desde el inicio y hasta el final de la trayectoria.

5.1. Pruebas con retroalimentación de datos de simulación (Lazo Semi-Cerrado)

Una primera aproximación para la realización de las pruebas fue utilizar datos provenientes de la propia simulación como entradas de retroalimentación, es decir, los datos calculados en la simulación eran enviados como instrucciones para comandar las acciones sobre el robot móvil, asimismo la retroalimentación del sistema provenía de los cálculos

o “estimaciones” que el propio software hacía sobre la posición del robot, en lugar de utilizar algún sistema de sensado para proveer esta retroalimentación.

Esta estructura será denominada Lazo Semi-Cerrado, por presentar la estructura cerrada de un lazo de control con la excepción de que la retroalimentación no proviene del sistema a controlar sino de una estimación.

Para realizar la prueba fue necesario en principio tener la simulación funcionando correctamente, posteriormente únicamente se agregaron los bloques necesarios para enviar los datos de posición y velocidad al robot móvil. Los bloques referidos son: el de configuración de la comunicación serial, y el de envío de datos a través del puerto serial.

Algo importante de mencionar, es que para la realización de esta prueba fue necesario “alentar” el tiempo de simulación para que éste coincidiera con el tiempo real, ya que los datos enviados al robot móvil debían estar temporizados y enviarse en el momento preciso para que la prueba fuese lo más fidedigna posible. Para lograr esto un bloque de Simulink® llamado “Real-Time Pacer” fue utilizado y configurado para que mantuviera una relación 1/1 del tiempo de simulación con el tiempo real.

Para hacer más explícita la arquitectura del sistema utilizada la el lazo semi-cerrado, esta es mostrado en la Figura 30.

Figura 30. Arquitectura del sistema en Lazo Semi-Cerrado

El objetivo primordial de esta configuración era poder realizar pruebas con el robot y establecer su funcionamiento sin ningún sistema de sensado, para posteriormente utilizarlo como referencia y poder medir, cualitativamente, el grado de mejora de la arquitectura una vez introducido el sistema de visión para la retroalimentación.

5.2. Pruebas con retroalimentación de datos provenientes del sistema de visión (Lazo-Cerrado)

Una vez realizadas las pruebas con el lazo semi-cerrado, se decidió construir la arquitectura final del sistema, utilizando el sistema de visión descrito en el Capítulo III del presente texto.

La arquitectura final, utilizaba el sistema de visión para sensar la posición y orientación del robot dentro de una superficie previamente calibrada y utilizaba esa información como lazo de retroalimentación para el sistema de control.

Para realizar esta prueba fue necesario sincronizar todos los elementos que participaban en el sistema. En primer lugar la simulación debía, al

igual que en la prueba en lazo semi-cerrado, estar temporizada para que coincidiera con el tiempo real. La segunda consideración a tomar en cuenta era el tiempo que le tomaba al sistema de visión capturar la imagen, codificar la información obtenida de ella, transmitirla entre Reactivision y Simulink y posteriormente decodificar la trama de datos para obtener de ella la información útil. Después de varias pruebas para lograr una correcta sincronización se llegó a la conclusión que el tiempo de muestreo que debía usarse para que el sistema funcionara de la mejor forma posible era de 0.9 segundos. Esto significa que cada 0.9 segundos el programa en Simulink® recibía una nueva posición y orientación del robot para realizar una iteración más en el control y arrojar el valor de las variables que el robot necesitaba para comandar las posiciones y velocidades de cada una de las llantas. Para la realización de esta prueba fue necesario mantener fijos los resultados de la iteración anterior, para que 0.9 segundos después fueran refrescados y de nuevo mantenidos, realizando iterativamente este mismo proceso a lo largo de la trayectoria.

La estructura del sistema en lazo cerrado se muestra en la Figura 31.

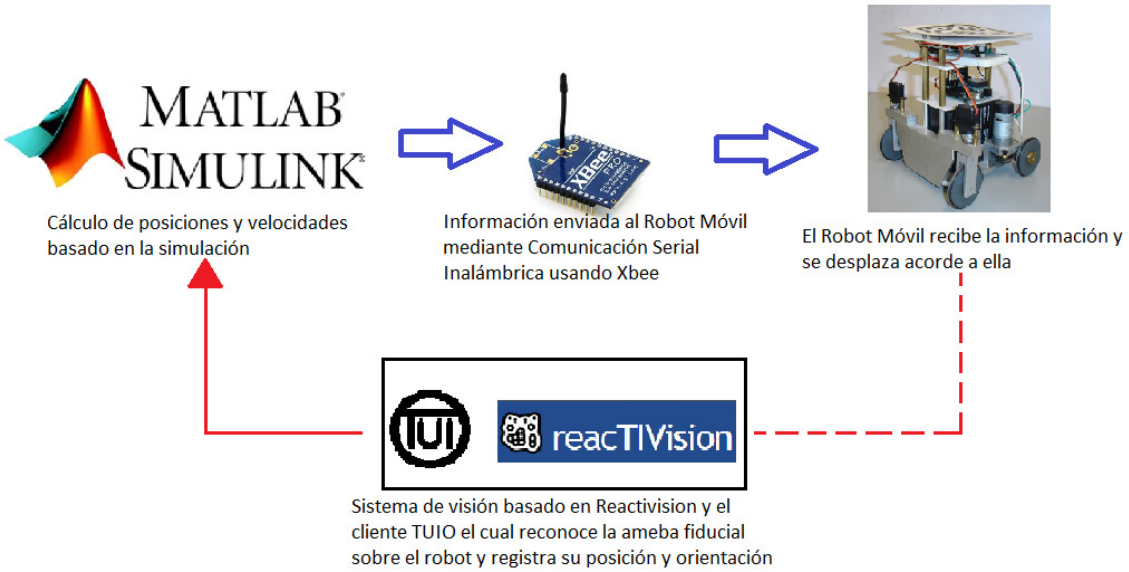


Figura 31. Arquitectura del Sistema en Lazo Cerrado

Una vez descritas ambas configuraciones se procederá a mostrar los resultados obtenidos de cada una de las pruebas realizadas. No hay que olvidar que el objetivo de realizar ambas pruebas era comparar la ventaja de utilizar un sistema de sensado externo al robot y basado en un sistema de referencia fijo el cual pudiera proveer información fidedigna sobre el desplazamiento del robot en el área establecida.

5.3. Resultados

A continuación se muestran los resultados obtenidos de cada una de las pruebas. Es importante mencionar que todas pruebas, tanto en lazo semi-cerrado como en lazo cerrado, se realizaron bajo las mismas condiciones de iluminación y calibración del área de trabajo y de la cámara.

5.3.1. Resultados obtenidos de las pruebas en Lazo Semi-Cerrado

Para la prueba en lazo semi-cerrado se programó una trayectoria senoidal con una frecuencia de 0.03 [Hz], una amplitud de 0.3 [m] y una velocidad de desplazamiento de la trayectoria a lo largo del eje X de 0.04 [m/s]. De esta prueba se hicieron dos experimentos, en cada uno de ellos se almacenaron los datos de la trayectoria seguida por el robot móvil, y la trayectoria que se le pidió seguir o trayectoria de referencia. Al estar la prueba en lazo semi-cerrado basada en la simulación, y no haber retroalimentación de la posición del robot, las gráficas que muestran el comportamiento de las velocidades y posiciones resultan ser exactamente las mismas que en la simulación, por lo que realmente únicamente es posible mostrar el comportamiento en el desplazamiento del robot comparado contra la trayectoria predefinida.

En la Figura 32, se observa como la trayectoria de referencia inicia en las coordenadas (-1,0), mientras el robot inicialmente se encontraba en una

posición cercana a (-0.9, -0.25). Resulta evidente que el robot no es capaz de sensar su posición con respecto a la trayectoria preestablecida, por lo que corregir su posición es imposible, únicamente se limita a seguir las instrucciones comandadas desde la simulación y aunque sí realiza una trayectoria con una geometría parecida a una senoidal, pareciera que la amplitud y la frecuencia de la onda cambiaran a lo largo del tiempo.

Resulta evidente que la trayectoria seguida por el robot está lejos de la de referencia.

Al finalizar la primera prueba, se decidió realizar una segunda para establecer claramente el comportamiento del robot y la manera en que este seguía las instrucciones preestablecidas. El resultado de la prueba se muestra en la Figura 33.

Es posible observar nuevamente la trayectoria de referencia con las mismas características que en la prueba anterior, mientras que la trayectoria seguida por el robot cambia con respecto a la primera prueba. En esta ocasión la trayectoria del robot inició aproximadamente en las coordenadas (-0.9, -0.15) y de nueva cuenta se observa una aparente variación en la amplitud y frecuencia de la trayectoria descrita, mostrando un evidente error con respecto a la trayectoria de referencia. Es importante aclarar que la posición inicial del robot se trataba de acercar lo más posible al (0,0) de la trayectoria de referencia, pero resultaba bastante complicado por la curvatura del lente de la cámara y por las propias dimensiones de la ameba *fiducial* iniciar exactamente en esa posición.

Finalmente es posible establecer una referencia clara de la cual partir sobre el comportamiento del sistema sin retroalimentación alguna para posteriormente realizar una comparación con los resultados de las pruebas en lazo cerrado. Los resultados obtenidos fueron los esperados,

resultaría muy insensato pensar que el robot sería capaz de seguir una trayectoria establecida sin ninguna herramienta que le permitiese corregir el error que presentaba con respecto a la referencia.

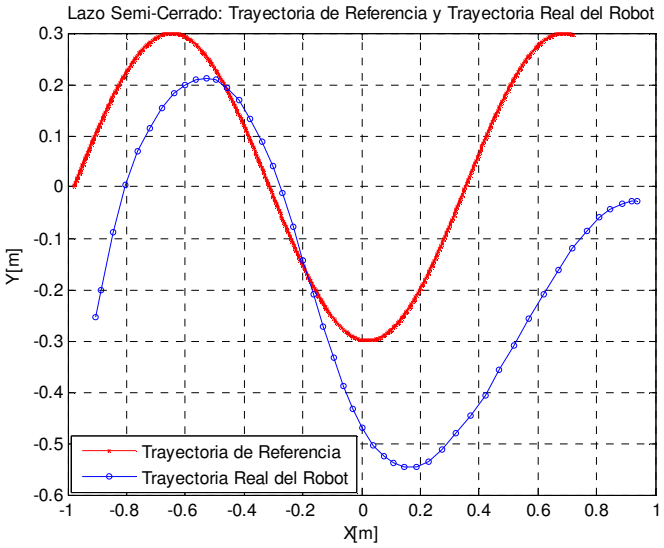


Figura 32. Resultado de la primera prueba en lazo semi-cerrado

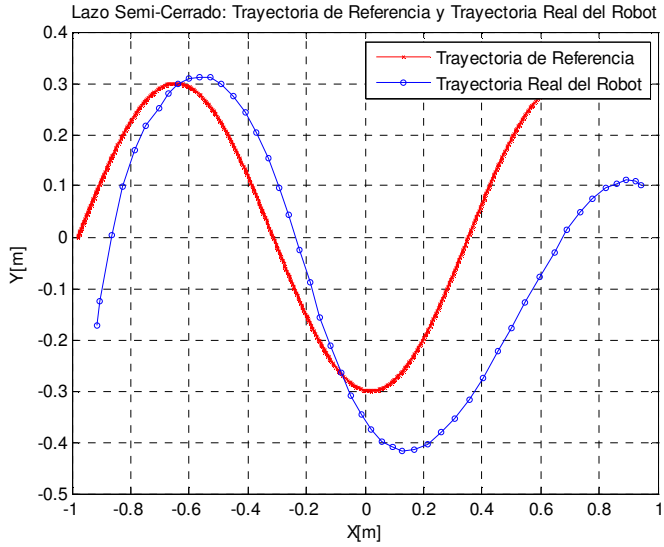


Figura 33. Resultado de la segunda prueba en lazo semi-cerrado

5.3.2. Resultados obtenidos de las pruebas en Lazo Cerrado

Para la prueba en lazo cerrado se programó una trayectoria senoidal con una frecuencia de 0.028 [Hz], una amplitud de 0.45 [m] y una velocidad

de desplazamiento de la trayectoria a lo largo del eje X de 0.062 [m/s]. De esta prueba se hicieron dos experimentos, en cada uno de ellos se almacenaron los datos de la trayectoria seguida por el robot móvil, y la trayectoria que se le pidió seguir o trayectoria de referencia. De igual manera, las variables de postura del robot, tanto sus desplazamientos, orientación del robot y ángulos de rotación de las llantas frontales fueron almacenados. En este caso, a comparación del experimento en lazo semi-cerrado, sí fue posible almacenar los datos y mostrarlos en comparación con los de la simulación porque al haber retroalimentación en cada iteración los cálculos arrojaban resultados diferentes basados en las entradas provistas por el sistema de visión.

En la Figura 34, es posible observar tanto la trayectoria de referencia como la trayectoria real del robot para la primera prueba en lazo cerrado. Es posible observar que la referencia inicia en las coordenadas (-1,0) mientras que el robot se encuentra cerca, aproximadamente en (-0.7, -0.02). En la imagen es posible percibir como la línea azul trata de seguir en todo momento a la línea roja, aun cuando no iniciaron en el mismo punto. Justo después de la primera curva pronunciada, en la cresta de la senoidal, el robot logra reducir considerablemente el error, y mantenerse prácticamente sobre la línea roja hasta que nuevamente en la siguiente curva pronunciada, el valle de la senoidal, incrementa un poco el error. En términos generales es posible observar un comportamiento de seguimiento de trayectoria muy bueno, considerando todos los "inconvenientes" que se encuentran afectando al sistema. No hay que olvidar que el tiempo de muestro es de 0.9 segundos, y que mientras se refresca la información de la posición del robot desde la información obtenida con visión se mantienen los resultados de la última posición incrementando un poco el error. También es de considerar el tiempo procesamiento de la imagen y de los cálculos, el envío de los datos

codificados por comunicación serial y la aplicación de las instrucciones en el robot.

De esta prueba también fue posible obtener las variables de postura del robot, los ángulos de las llantas frontales y la virtual, y el error de seguimiento de la trayectoria tanto en el eje X como en el eje Y. Los resultados se muestran en las Figuras 35, 36 y 37.

En la Figura 35, se observan los desplazamientos tanto en el eje X como en el eje Y del robot móvil, así como su orientación a lo largo de la trayectoria. Es posible distinguir los “brincos” debidos a la discretización de los datos, y al refresco de los datos provenientes de visión cada 0.9 segundos.

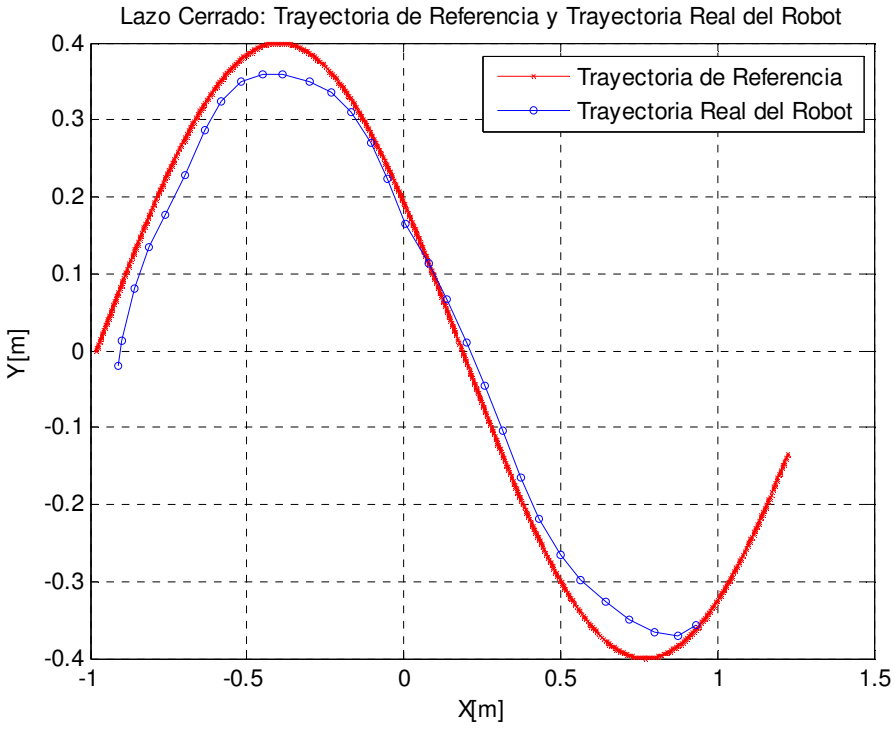


Figura 34. Trayectoria de Referencia y Trayectoria Real del Robot

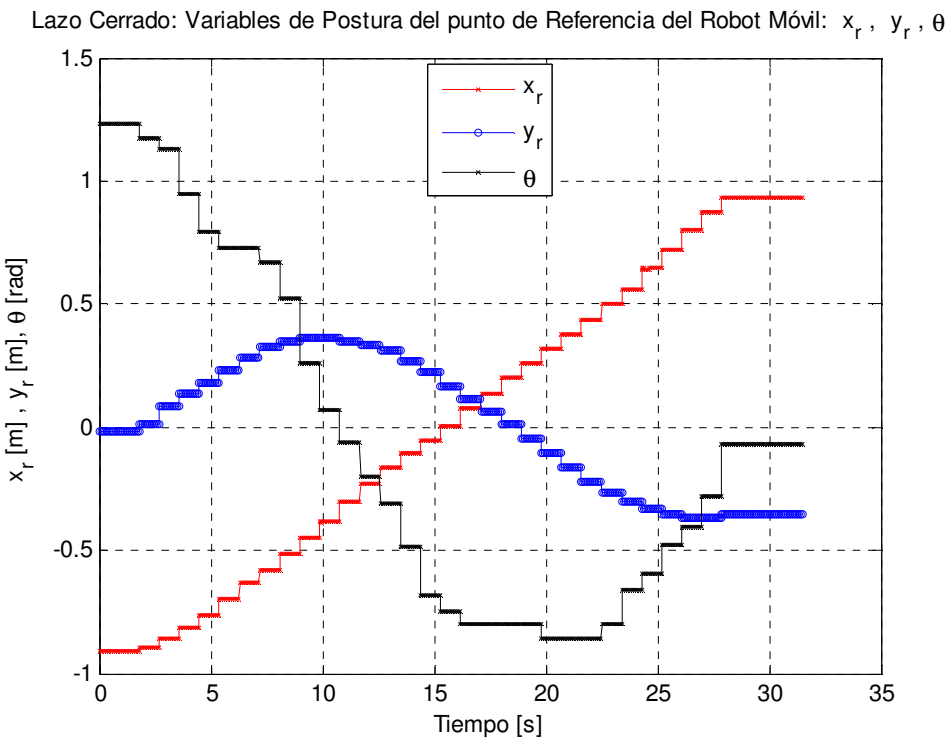


Figura 35. Variables de Postura del punto de Referencia del Robot Móvil

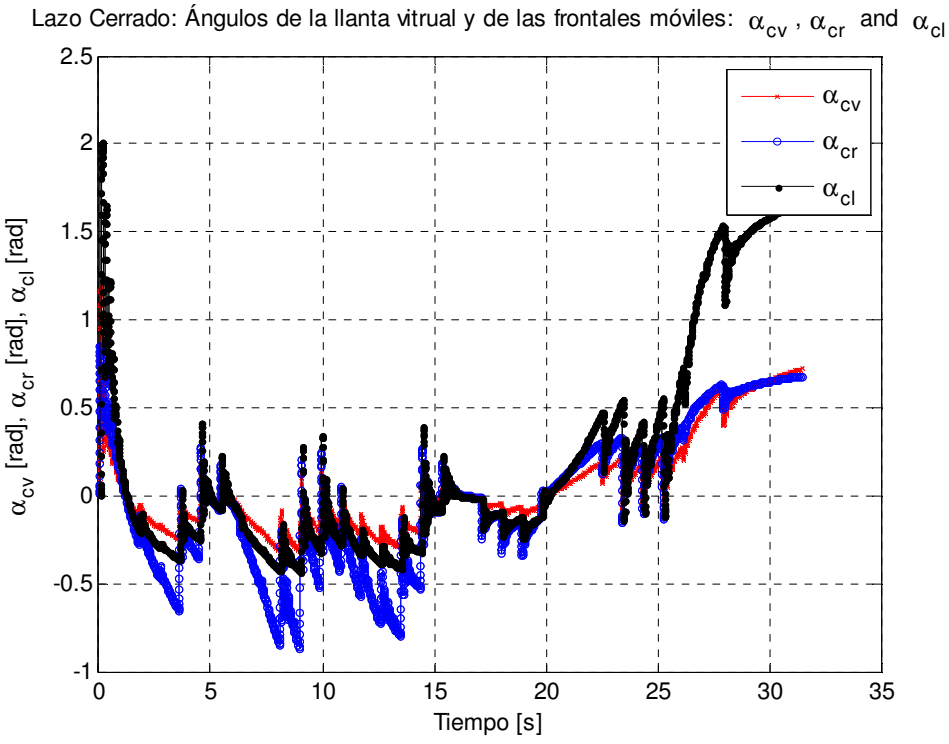


Figura 36. Ángulos de la llanta virtual y de las llantas frontales móviles

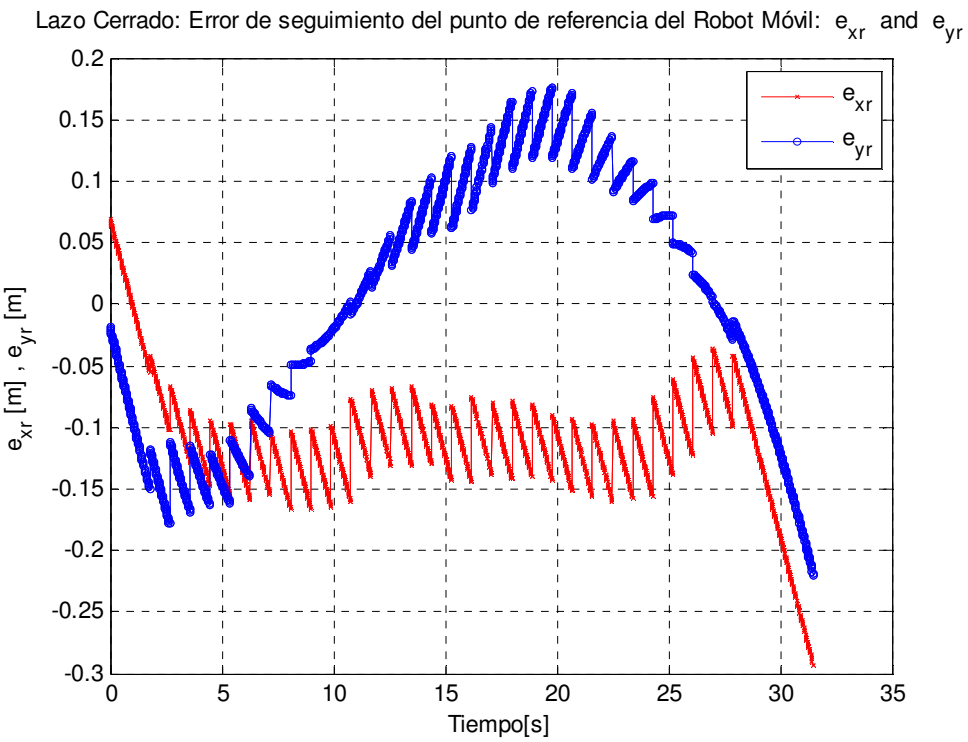


Figura 37. Error de seguimiento del punto de referencia del Robot Móvil.

En la Figura 36, se observa el comportamiento de la llanta virtual así como de las llantas frontales direccionables. El comportamiento muestra muchas variaciones, pero estos ajustes están hechos en función de la retroalimentación de la visión con el objetivo de corregir el error entre la trayectoria de referencia la posición real del robot.

Finalmente en la Figura 37 se muestra el error de seguimiento para la trayectoria en el eje X y para la trayectoria en el eje Y, es posible notar que el error nunca es mayor a 0.15 [m], lo cual considerando la propia curvatura del lente de la cámara y lo complicado de la calibración del área de trabajo es realmente pequeño comparado con el tamaño del área de trabajo de aproximadamente 3 [m^2].

La segunda prueba en lazo cerrado consistió en el mismo experimento, se recabaron nuevamente los datos y se obtuvieron las mismas gráficas

que para la primera prueba en lazo cerrado. El objetivo de replicar el experimento era darle validez a la prueba realizada.

En la Figura 38 se puede observar la trayectoria de referencia y la trayectoria real del robot, y nuevamente aun cuando las trayectorias no inician en el mismo punto, se puede observar como el robot sigue de manera más fiel la trayectoria de referencia, haciendo el error prácticamente cero en la zona comprendida entre la cresta y el valle de la senoidal.

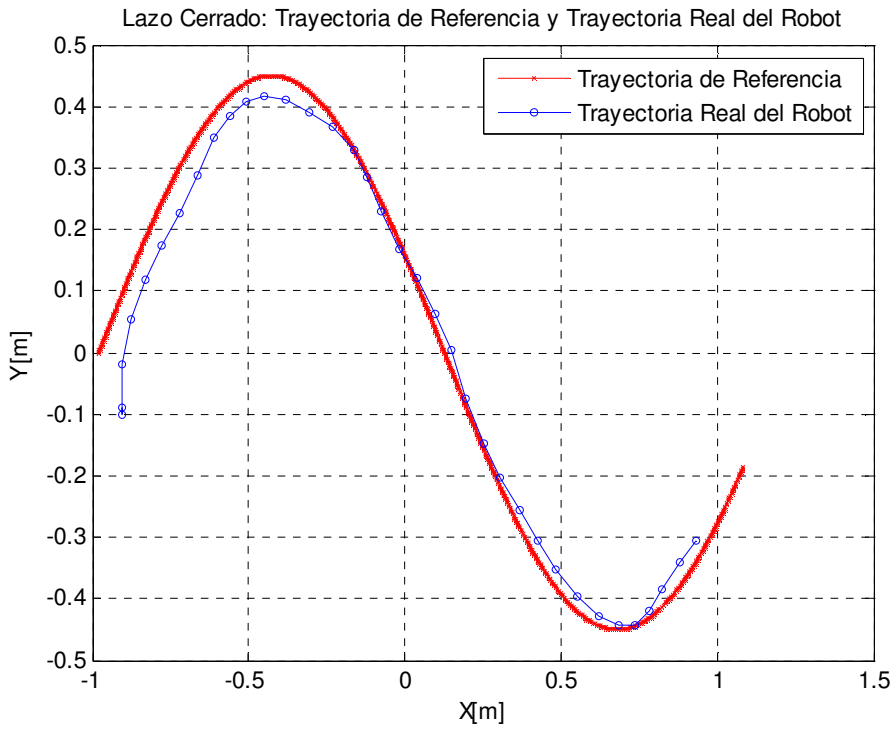


Figura 38. Trayectoria de Referencia y Trayectoria Real del Robot

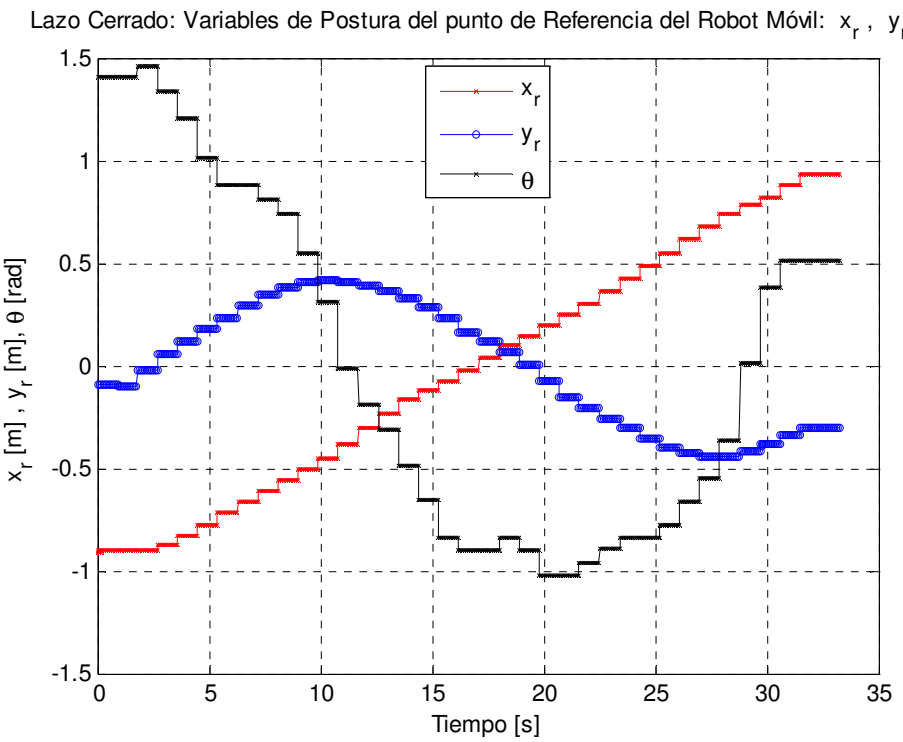


Figura 39. Variables de Postura del punto de Referencia del Robot Móvil

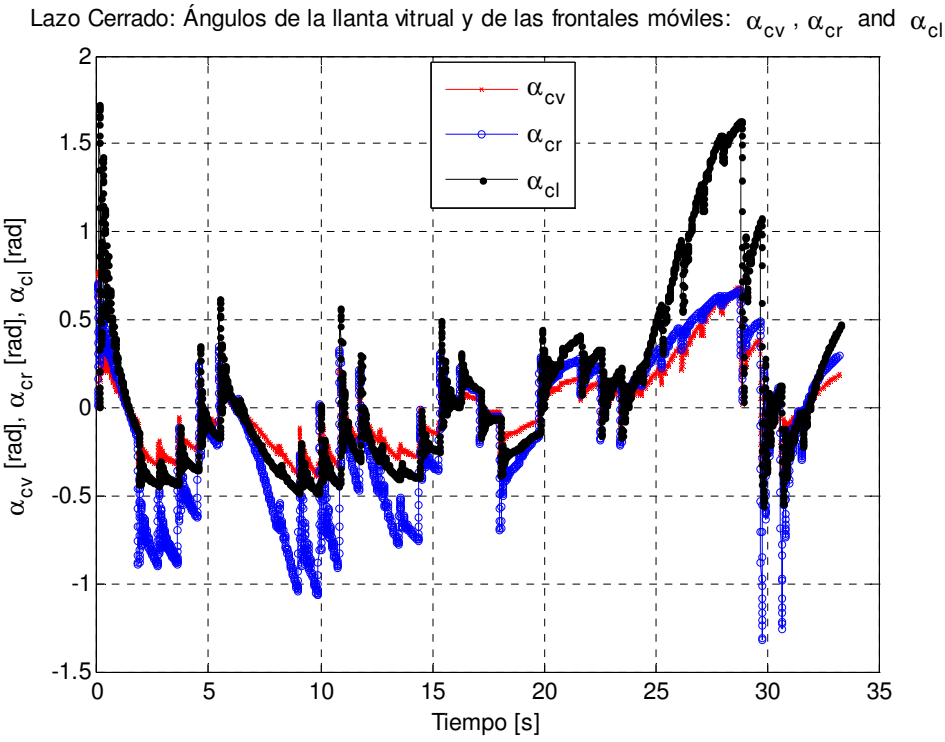


Figura 40. Ángulos de la llanta virtual y de las llantas frontales móviles

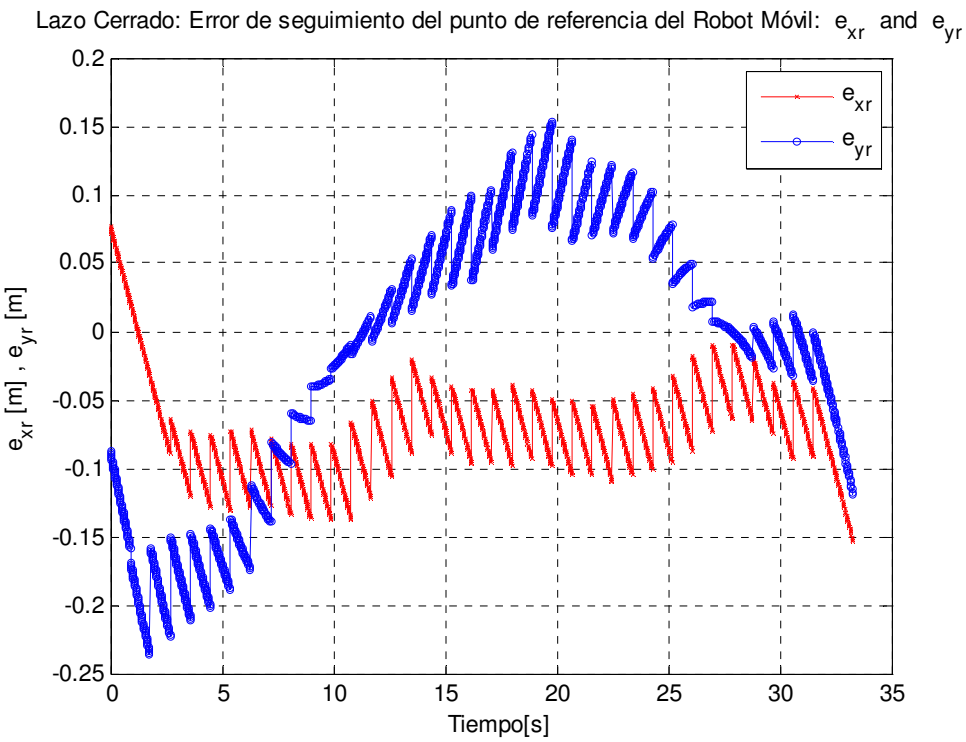


Figura 41. Error de seguimiento del punto de referencia del Robot Móvil.

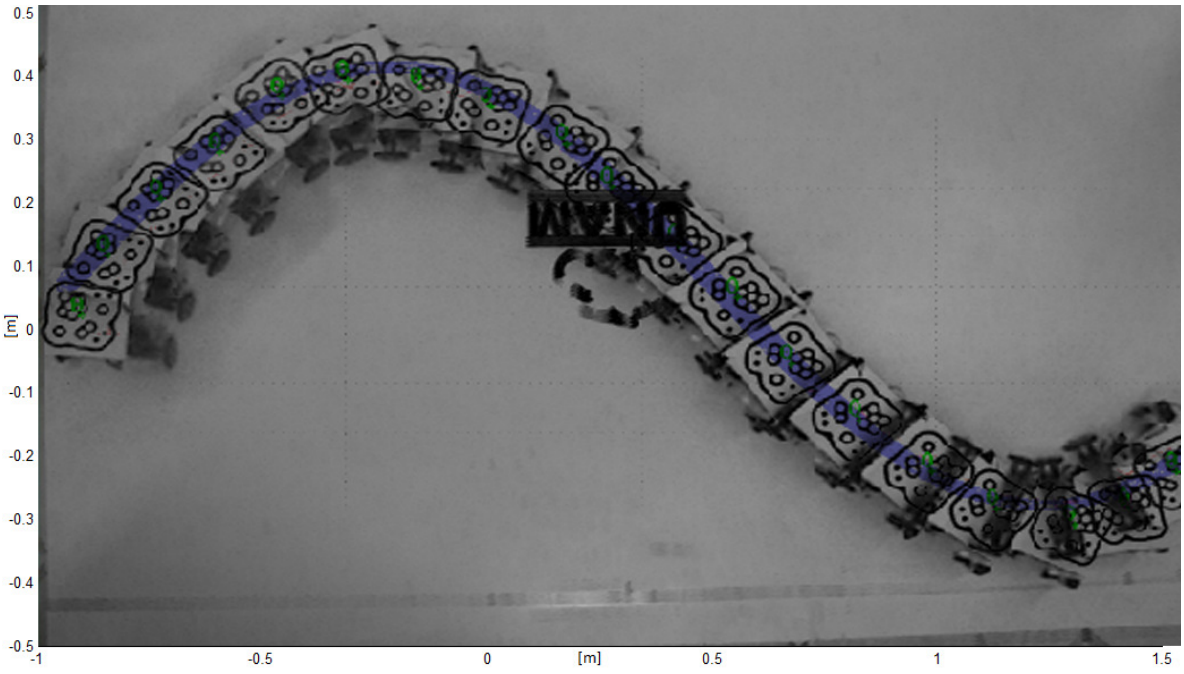


Figura 42. Secuencia de imágenes obtenidas del video que muestran el desplazamiento del robot móvil y en azul la trayectoria de referencia.

La Figura 39 muestra el comportamiento de las variables de postura del robot, tanto sus desplazamientos como la orientación del robot móvil y se observa un comportamiento muy parecido a las gráficas mostradas en la Figura 35.

Las Figuras 40 y 41 muestran los ángulos tanto de la llanta virtual como de las móviles delanteras, y el error de seguimiento entre la referencia y la trayectoria real del robot tanto con respecto al eje X como con respecto al eje Y, respectivamente.

Finalmente, a manera ilustrativa y mucho más visual, se muestra una secuencia de imágenes tomadas de la trama de video y superpuestas para lograr mostrar, en una sola imagen, el desplazamiento del robot móvil sobre la superficie y como éste se movió con respecto a la trayectoria de referencia mostrada en azul.

Una vez mostrados los resultados obtenidos, tanto de las pruebas en lazo semi-cerrado como de las pruebas en lazo cerrado, es notable inmediatamente la mejora en los experimentos cuando se utiliza un sistema de sensado externo al robot, que cuando se pretende estimar el comportamiento del mismo. Resulta evidente a la vista como en ambas pruebas en lazo cerrado el robot sigue prácticamente la trayectoria predefinida, ajustando la posición y velocidad de sus llantas para lograrlo.

Finalmente es posible afirmar, que los objetivos de este trabajo fueron cumplidos cabalmente. Se logró montar un sistema basado en visión y Simulink®, que permitiera obtener la posición y orientación del robot dentro de un área de trabajo, se utilizó la información obtenida del sistema de visión para retroalimentar el lazo de control diseñado y simulado por V.J. González-Villela [4] con la finalidad de seguir una

trayectoria previamente programada, y se comprobó que usando una llanta virtual direccionable fue posible coordinar el movimiento de las restantes en su velocidad (tracción) y orientación (dirección) sobre un robot móvil propuesto.

6. Conclusiones y Trabajo a Futuro

El presente documento muestra el desarrollo de un sistema de visión para sensar parámetros del robot móvil compatible con el software Simulink®y comprueba físicamente, utilizando todo el sistema diseñado, que usando una llanta virtual direccionable es posible coordinar el movimiento de las restantes en su velocidad y orientación sobre un robot móvil.

Al finalizar las pruebas, y ver en retrospectiva el trabajo realizado para reportarlo en estas páginas, resulta claro que se lograron cumplir los objetivos propuestos. El primero de ellos establecía la necesidad de desarrollar un sistema de visión que nos permitiera obtener las coordenadas, orientación y velocidad de un "objeto" que se desplazara dentro del área enfocada por la cámara y posteriormente introdujera las lecturas de los datos a Simulink®. El reto fue mayúsculo y fue el objetivo que requirió de mayor tiempo y dedicación para ser resuelto, aun cuando se utilizaron herramientas preexistentes como el software ReacTIVision y el protocolo TUIO que facilitaron ciertas tareas como el procesamiento de la imagen y la propia obtención de los datos de la misma, pero había que resolver la comunicación del protocolo TUIO y Simulink®. Finalmente fue posible establecer la comunicación entre ambas piezas de Software y completar el sistema de visión.

El segundo objetivo era utilizar el sistema de visión, cambiar el simple objeto por un robot móvil desplazándose sobre la superficie e introducir en control diseñado por V.J. Gonzalez Villela [4] los datos provenientes del sistema de visión para cerrar el lazo de control con la finalidad de seguir una trayectoria programada. Una vez que el sistema de visión funcionaba correctamente, y el robot móvil por su parte era capaz de ser comandado a través de instrucciones comunicadas inalámbricamente desde una computadora, el paso final fue ensamblar todo el sistema en uno solo, que contuviera el control, la codificación de las instrucciones hacia el robot, la comunicación inalámbrica y el sistema de visión. Todo funcionó exitosamente.

Y finalmente, se comprobó que una llanta virtual direccionable era capaz de controlar las restantes del robot móvil para que este lograra seguir una trayectoria predefinida.

A la vez que se llevaron a cabo las pruebas, me percaté de que el trabajo realizado podría incorporar mejoras a futuro, las cuales menciono a continuación:

Incorporación de sensores en el robot móvil: Dependiendo de la aplicación que se requiera del robot móvil, podría resultar útil incorporar sensores que evitaran colisiones y que trabajaran en coordinación con el sistema de visión.

Utilización de una computadora con mayor capacidad de procesamiento: El equipo utilizado para realizar las pruebas resulta fundamental para la velocidad de procesamiento de datos. Un equipo con mayores prestaciones en memoria y procesador podría mejorar la velocidad de respuesta del sistema y reducir el tiempo de muestreo de 0.9 segundos a al menos unos 0.5 segundos.

Utilización de un equipo de cómputo dedicado al procesamiento de imágenes: Resultaría muy benéfico para las pruebas en tiempo real desarrollar un sistema dedicado que procesara las imágenes y enviara a través de algún protocolo de comunicación la información recabada a la computadora central, evitando la dedicación de tiempo en el procesamiento, reduciendo el tiempo de muestreo y mejorando la resolución para el experimento.

Incorporación del módulo de "Tiempo Real" de Simulink®: El software Simulink® cuenta con un "*Toolbox"* o herramienta dedicada para crear programas que funcionen en tiempo real, que mejoran de manera importante la comunicación serial y por tanto mejorarían la comunicación con el robot. El único inconveniente es que se podría requerir de un equipo de cómputo más robusto que permitiese la ejecución tanto del módulo en tiempo real

como del sistema de visión, o en caso contrario, utilizar la recomendación previa y el módulo de tiempo real.

Estas son en general las recomendaciones que sugeriría para aquel que decida incursionar nuevamente en la robótica móvil y la visión como sistema de sensado externo.

Para finalizar me parecería importante mencionar que toda la experimentación en esta tesis desarrollada, podría ser escalada y utilizada para aplicaciones útiles a la humanidad. Pensemos por ejemplo en un robot "agricultor" que cense las condiciones de humedad, temperatura y acidez de algún terreno con fines agrícolas y que fuese guiado con una trayectoria preprogramada, utilizando herramientas de geoposicionamiento como GPS en lugar de visión. Este sistema nos permitiría que el robot siguiese trayectorias complejísimas sin utilizar, por ejemplo, algún método físico para marcar el camino a seguir; no serían necesarias líneas de magnetos o de colores específicos, sino que desde una computadora podría ser posible determinar una trayectoria basado en un sistema coordenado, y el robot al estar referenciado al mismo sistema sería capaz de incorporarse a la trayectoria y seguirla, brindando completa flexibilidad al usuario.

Otro ejemplo sería el uso del concepto de la llanta virtual para controlar vehículos automotores, cambiando la tradicional dirección de Ackermann por sistemas que permitiesen controlar al vehículo utilizando señales eléctricas en lugar de mecanismos, y que actualmente son conocidos como "drive-by-wire".

APÉNDICES

APÉNDICE A CÓDIGO PROGRAMADO EN EL MICROCONTROLADOR

```
#include <Wire.h>
#include <Servo.h>
Servo myservo1;
Servo myservo2;

#define i2cAddress 0x58               // Dirección i2c

// Comandos de la Tarjeta MD25
#define speed1 0x00                   // Velocidad del primer motor
#define speed2 0x01                   // Velocidad del segundo motor
#define softwareVer 0x0D              // software version

int x=128, x2=128,pos1=90,pos2=90;// Inicialización de las variables de posicion y velocidad
long x_t;                          //angulos en 90 grados y velocidad en 0 rpm

#define BUFFERSIZE 20

char buffer[BUFFERSIZE];         // Comunicación serial de entrada
char *parseptr;
char bufferidx;

void setup()
{
  Serial.begin(115200);       // Configuracion de la comunicación serial a 115200 bps
  pinMode(13, OUTPUT);
  Wire.begin();
  delay(200);                // Tiempo de espera de 200 ms para que todos los dispositivos
respondan

  Serial.print("MD25 version:");
  int softVer = getSoft();             // Obtiene la version de Software de la tarjeta MD23
  Serial.println(softVer, DEC);
  myservo1.attach(9);
  myservo2.attach(10);
  encodeReset();                      // Reset de los encoderes de los motores a cero
  Serial.flush();

}

void loop()
{

  if (Serial.available()>0)        //Dato entrante en el puerto serial
  {
    pos2=Serial.read();            //El primer dato es una posición de llanta frontal
    delay(5);
    pos1=Serial.read();            //El segundo dato es una posición de llanta frontal
    delay(5);
    x2=Serial.read();              //El tercer dato es una velocidad de llanta trasera
    delay(5);
    x=Serial.read();               //El cuarto dato es una velocidad de llanta trasera
    delay(5);
  }

      digitalWrite(13,HIGH);

      myservo1.write(pos1);       //Se envía la primera posicion al servomotor
      myservo2.write(pos2);       //Se envía la segunda posicion al servomotor

      Wire.beginTransmission(i2cAddress); // Comanda la velocidad al motor de tracción x
      Wire.send(speed1);
      Wire.send(x);
      Wire.endTransmission();
```

```
      Wire.beginTransmission(i2cAddress); // Comanda la velocidad al motor de tracción x2
      Wire.send(speed2);
      Wire.send(x2);
      Wire.endTransmission();



      delay(10);

      digitalWrite(13,LOW);

}

int getSoft(){                            // Función que obtiene la versión de Software
  Wire.beginTransmission(i2cAddress);     //  Envía un byte para leer la versión de software
  Wire.send(softwareVer);
  Wire.endTransmission();

  Wire.requestFrom(i2cAddress, 1);        // Solocita un byte de la tarjeta MD25
  while(Wire.available() < 1);            // Espera a que esté disponible
  int software = Wire.receive();          // Se lee el byte

  return(software);
}

void encodeReset(){                       // Esta función resetea los enconders a cero
  Wire.beginTransmission(i2cAddress);
  Wire.send(cmdByte);
  Wire.send(0x20);                        // El valor 0x20 resetea los encoders
  Wire.endTransmission();
}

void readline(void) {
  char c;
  bufferidx = 0; // se inicia al principio

  while (1) {
    c = Serial.read();
    if (c == -1)
      continue;
    //Serial.print(c);

    if (c == '\n')
    {
      Serial.print("*OK*");
      break;
    }

    {
      if ((bufferidx == BUFFERSIZE-1) || (c == '\r')) {
        buffer[bufferidx] = 0;
        return;
      }
    }
    buffer[bufferidx++]= c;
  }
}

long parsedecimal(char *str) {
  long d = 0;

  while (str[0] != 0) {
    if ((str[0] > '9') || (str[0] < '0'))
      return d;
    d *= 10;
    d += str[0] - '0';
    str++;
  }
  return d;}
```

APÉNDICE B CÓDIGO DEL BLOQUE DE VISIÓN EN SIMULINK®

```
function [sys,x0,str,ts] = visioncode(t,x,u,flag)

switch flag,

  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;

  case 1,
    sys=mdlDerivatives(t,x,u);

  case 2,
    sys=mdlUpdate(t,x,u);

  case 3,
    sys=mdlOutputs(t,x,u,flag);

  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);

  case 9,
    sys=mdlTerminate(t,x,u);

  otherwise
    error(['Unhandled flag = ',num2str(flag)]);

end

% end sfuntmpl

%
%=============================================================================
====
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-
function.
%=============================================================================
====
%
function [sys,x0,str,ts,T_samp]=mdlInitializeSizes

%
sizes = simsizes;

sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 4;
sizes.NumInputs      = 0;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;   % at least one sample time is needed

sys = simsizes(sizes);
```

```
x0  = [];

str = [];

ts  = [0.9 0];

% end mdlInitializeSizes

%
%=============================================================================
====
% mdlDerivatives
% Return the derivatives for the continuous states.
%=============================================================================
====
%
function sys=mdlDerivatives(t,x,u)

sys = [];

% end mdlDerivatives

%
%=============================================================================
====
% mdlUpdate
% Handle discrete state updates, sample time hits, and major time step
% requirements.
%=============================================================================
====
%
function sys=mdlUpdate(t,x,u)

sys = [ ];

% end mdlUpdate

%
%=============================================================================
====
% mdlOutputs
% Return the block outputs.
%=============================================================================
====
%

function sys=mdlOutputs(t,x,u, flag)

global a1 x1 y1 s1;
import TUIO.*;
tc = TuioClient();
tc.connect();
```

```
for i = 1 : 4

    c=tc.getTuioObjects().size();

    if (c > 0)
        a1 = tc.getTuioObjects().get(0).getAngle();
        x1 = tc.getTuioObjects().get(0).getPosition().getX();
        y1 = tc.getTuioObjects().get(0).getPosition().getY();
        s1 = tc.getTuioObjects().get(0).getMotionSpeed();
    end;
    pause(0.1);
end;

tc.disconnect();

sys = [a1 (x1*1.96)-0.98 (y1*1.47)-0.735 s1];




% end mdlOutputs

%
%=============================================================================
====
% mdlGetTimeOfNextVarHit
% Return the time of the next hit for this block.  Note that the result is
% absolute time.  Note that this function is only used when you specify a
% variable discrete-time sample time [-2 0] in the sample time array in
% mdlInitializeSizes.
%=============================================================================
====
%
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sys = [];
% sampleTime = 1;    %  Example, set the next hit to be one second later.
% sys = t + sampleTime;

% end mdlGetTimeOfNextVarHit

%=============================================================================
====
% mdlTerminate
% Perform any end of simulation tasks.
%=============================================================================
====
%
function sys=mdlTerminate(t,x,u)

sys = [];

% end mdlTerminate
```

REFERENCIAS

1. Gopalakrishnan, B., S. Tirunellayi, and R. Todkar, *Design and development of an autonomous mobile smart vehicle: a mechatronics application.* Mechatronics, 2004. **14**(5): p. 491-514.
2. Díaz-Hernández, O., *Experimentación con una configuración de robot móvil de llantas frontales independientes en su direccionabilidad*, in *Departamento de Mecatrónica*. 2010, Universidad Nacional Autónoma de México: México D.F. p. 71.
3. Maier, D. and A. Kleiner. *Improved GPS sensor model for mobile robots in urban terrain*. in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. 2010.
4. Gonzalez-Villela, V.J., *Research on a semiautonomous mobile robot for loosely structured enviroments focused on transporting mail trolleys*, in *Mechanical and Manufacturing Engineering*. 2006, Loughborough University: Loughborough, UK.
5. Mehta, S.S., T.F. Burks, and W.E. Dixon, *Vision-based localization of a wheeled mobile robot for greenhouse applications: A daisy-chaining approach.* Computers and Electronics in Agriculture, 2008. **63**(1): p. 28-37.
6. Robot-Electronics. *EMG-30 DC MOTOR*. [cited 2011 14 Junio]; Available from: http://www.robot-electronics.co.uk/htm/emg30.htm.
7. Robot-Electronics. *MD-25*. [cited 2011 14 Junio]; Available from: http://www.robot-electronics.co.uk/htm/md25tech.htm.
8. Wikipedia. *Servomotor de Modelismo*. [cited 2011 14 Junio]; Available from: http://es.wikipedia.org/wiki/Servomotor_de_modelismo.
9. Netrino. *PWM Pulse Width Modulation*. [cited 2011 14 Junio]; Available from: http://www.netrino.com/Embedded-Systems/How-To/PWM-Pulse-Width-Modulation.
10. Behnke, S. and M. Schreiber, *Digital Position Control for Analog Servos*, in *Humanoid Robots Group, Computer Science Institute*. 2006, University of Freiburg: Freiburg, Germany. p. 5.
11. Atmel. *ATMega328 Microcontroller*. [cited 2011 14 Junio]; Available from: http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdf.
12. Arduino. *Arduino Duemilanove*. [cited 2011 14 Junio]; Available from: http://www.arduino.cc/es/Main/ArduinoBoardDuemilanove.
13. Granada, U.d. *Bus I2C*. [cited 2011 14 Junio]; Available from: http://atc.ugr.es/~afdiaz/fich/bus_i2c.pdf.
14. Wikipedia. *Modulación por Ancho de Pulso*. [cited 2011 14 Junio]; Available from: http://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_ancho_de_pulsos.
15. Robot-Electronics. *Arduino Examples*. [cited 2011 14 Junio]; Available from: http://www.robot-electronics.co.uk/htm/arduino_examples.htm.
16. Ángeles-García, A.M. and D. Lima-Robleda, *Sistema semi-autónomo de robots móviles colaborativos para el manejo de materiales*, in *Departamento de Mecatrónica*. 2011, Universidad Nacional Autónoma de México: Mexico D.F. p. 140.
17. DEC. *Xbee*. [cited 2011 15 Junio]; Available from: http://decelectronics.com/html/XBEE/XBEE.htm.
18. Software Design, a.e.s.t. *Xbee*. [cited 2011 15 Junio]; Available from: http://embedsoftdev.com/embedded/xbee/.
19. BricoGeek. *Arduino Xbee Shield*. [cited 2011 15 Junio]; Available from: http://www.bricogeek.com/shop/14-arduino-xbee-shield.html.

20. Robot-Gear. *Xbee Explorer USB*. [cited 2011 15 Junio]; Available from: http://www.robotgear.com.au/Product.aspx/Details/401.
21. Real Academia de la Lengua Española. *Visión*. [cited 2011 17 Junio]; Available from: http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=visión.
22. Real Academia de la Lengua Española. *Ver*. [cited 2011 17 Junio]; Available from: http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=ver.
23. Microsoft. *SonyDCRHC28 VideoCamera*. [cited 2011 21 Junio]; Available from: http://www.microsoft.com/Windows/compatibility/windows-7/en-us/Details.aspx?type=Hardware&p=Sony%20Handycam%20DCR-HC28%202.5LCD%2020x%20Camcorder&v=Sony&uid=DCRHC28&l=en-US&pf=1&pi=9&c=Cameras%20%26%20Photo&sc=Camcorders&os=64-bit.
24. ReacTIVison. *ReacTIVision*. [cited 2011 21 Junio]; Available from: http://reactivision.sourceforge.net/.
25. Mathworks. *Simulink*. [cited 2011 21 Junio]; Available from: http://www.mathworks.com/products/simulink/.