UNIVERSIDAD DE SONORA

DIVISION DE CIENCIAS EXACTAS Y NATURALES

INGENIERIA EN TECONOLOGIA ELECTRONICA

Control de un Robot Móvil con FPGA

Tesis profesional presentada por

Miguel Angel Castillo Rojo

Como requisito para obtener el título de

INGENIERO EN TECNOLOGÍA ELECTRÓNICA

Directores de Tesis:

Dra. Alicia Vera Marquina

Dr. Luis Arturo García Delgado

Hermosillo, Sonora, México. **Septiembre de 2014**

Agradecimientos

A mi directora de tesis, la Dra. Alicia Vera Marquina, por su apoyo y confianza al permitirme trabajar en conjunto en el desarrollo de esta tesis y concluir así una etapa de mi vida y superarme académicamente.

Con todo mi cariño y mi amor para las personas que hicieron todo en la vida para que yo pudiera lograr mis sueños, por motivarme y darme la mano cuando sentía que el camino se terminaba, a ustedes por siempre mi corazón y mi agradecimiento.

A mi Padre y Madre

A mis hermanos que aun con las diferencias seguimos juntos y en la batalla del día a día los espero ver haciendo lo mismo adelante.

A tu paciencia y comprensión, preferiste sacrificar tu tiempo para que yo pudiera cumplir con el mío, ahora puedo decir que esta tesis lleva mucho de ti, gracias por estar siempre a mi lado en las buenas como las malas incondicionalmente.

Zulema

Al Dr. Luis Arturo García como todos mis maestros que en este andar por la vida, influyeron con sus lecciones y experiencias en formarme como una persona de bien y preparada para los retos que pone la vida, a todos y cada uno de ellos les dedico cada una de estas páginas de mi tesis. Así como también a mis compañeros universitarios que muchos de ellos me apoyaron también en su momento con dudas que a su vez también me ayudaron a aprender más aún.

Índice

Capítulo 1

Introducción

1.1 Antecedentes y Justificación

Durante los últimos cincuenta años, la robótica no sólo ha incursionado en la industria, centro de investigación, universidades y hospitales, actualmente existen fábricas completamente automatizadas mediante robots manipuladores. Hoy en día, la robótica es tan familiar que se pueden encontrar robots en el hogar realizando tareas domésticas. No obstante, se considera un área joven en constante crecimiento.

La naturaleza multidisciplinaria de la robótica permite involucrar una gran cantidad de áreas del conocimiento tales como matemáticas, física, electrónica, computación, visión e inteligencia artificial, entre otras. Por otro lado, aun cuando la robótica es un área eminentemente experimental todos sus resultados están sustentados con un estricto rigor científico.

La robótica se ha convertido en un área clave y estratégica para todo país en desarrollo, es sinónimo de modernización y coadyuva a proporcionar bienestar a la sociedad.

Los robots móviles han cobrado una importancia creciente desde los años ochenta y noventa. Un robot móvil requiere de mecanismos de locomoción que le permitan moverse ilimitadamente dentro de su entorno. En contraste, los robots manipuladores generalmente se encuentran anclados a una superficie de trabajo fija y limitada físicamente. Hay varias formas para lograr el desplazamiento de un robot, la selección del mecanismo de locomoción es un aspecto importante para el diseño de este tipo de robots.

Los robots móviles son utilizados en el hogar para limpiar y recolectar basura; en hospitales se emplean para trasladar instrumental de quirófano al área de desinfectado; en investigación científica del espacio (en la luna o en planetas) se ocupan para analizar y enviar información de piedras, arenas y atmósfera; en arqueología son empleados para enviar señales de video del interior de cavernas, túneles, pirámides, etc. Lo que permite que el ser humano no tenga que ir a lugares estrechos o peligrosos.

1.2 Objetivo

En este trabajo se presenta el diseño de un robot móvil, que para su autonomía o posible control humano se usa lenguaje de descripción de hardware, el cual funciona sobre una tarjeta controlador FPGA (del Ingles Field Programmable Gate Array). La autonomía del robot requiere de dos sensores para posicionamiento o detección de obstáculos al momento de dejarlo por si solo en el entorno. Los sensores serán uno infrarrojo y otro ultrasónico y tendrán su propio método de comunicación. La comunicación entre los sensores y la tarjeta controladora se realiza mediante el acoplamiento de señales analógicas y digitales. Gracias a la amplia gama de puertos y capacidades de la tarjeta de control es posible incluir otros tipos de sensores que amplíen el desempeño del robot, sin embargo, este trabajo se enfocará a aplicación los sensores mencionados.

Otro de los objetivos principales es utilizar programación VHDL (combinación de VHSIC y HDL, donde VHSIC es el acrónimo de Very High Speed Integrated Circuit y HDL es a su vez el acrónimo de Hardware Description Language, aplicado al lenguaje para circuitos digitales propuesto en la norma ANSI/IEEE 1076) para programar el control del robot móvil y la síntesis utilizando la tecnología FPGA, con la intención de promover el interés hacia esta tecnología.

1.3 Organización de la tesis

La presente tesis está dividida en 5 capítulos, el capítulo 1 es una introducción del motivo por el cual se desarrolla la presente tesis, así como el objetivo y justificación de la misma. En el capítulo 2 se muestra el modelado y la justificación matemática de robots móvil así como la simulación del movimiento de un robot uniciclo y cuyo sistema de locomoción es de tipo diferencial. En el capítulo 3 se presenta la descripción del sistema de control por medio de una tarjeta de desarrollo FPGA, así como el Lenguaje de Descripción de Hardware utilizado para la síntesis del sistema en el dispositivo lógico programable (FPGA). Los resultados de este trabajo de tesis se muestran en el capítulo 4. En el capítulo 5 se presentan las conclusiones respecto al trabajo desarrollado y así mismo se presenta el trabajo a futuro. Los apéndices de los programas desarrollados se muestran al final de la tesis.

Capítulo 2

Robótica Móvil

2.1 Introducción

Los primeros robots móviles aparecen a mediados del siglo XX. William Grey Walter construyo dos robots móviles autónomos, Elmer y Elsie, en 1949. Entre 1961 y 63, Beast se desarrolla en la universidad Johns Hopkins. Es capaz de identificar tomas de corriente y aproximarse a ellas para recargar su batería de forma autónoma. Y a finales de los años 60, Shakey es desarrollado en el instituto de Investigaciones de Stanford. Shakey es el primer robot móvil capaz de razonar sobre sus propias acciones [8].

Respecto a los robots móviles esto fueron las primeras apariciones de ellos en este entonces los robot eran simples cajas con piezas internas no tenían alguna forma definida solamente contaban con sensores y ejes móviles los cuales les permitían verificar su alrededor para ejercer algún movimiento.

Otra de las grandes evoluciones que se generó a través del tiempo fue el inicio de los robot humanoides de los cuales se hace mención pero no son parte del estudio en este trabajo. Uno de los proyectos más avanzados en estos días es el robot ASIMO, fabricado por la empresa de origen japonés HONDA, la cual comenzó desde un armazón de dos piernas el cual difícilmente caminaba y se mantenía en pie a un robot humanoide completo el cual puede responder a estimulo humano.

La robótica móvil ha evolucionado conjuntamente con los avances tecnológicos reciente, lo cual ha dado lugar al desarrollo de muchas aplicaciones en áreas muy diversas. Si bien, dicha evolución ha alcanzado un nivel elevado en la actualidad, aun hay que esperar para la llegada de la revolución social de la robótica, en donde los robots llegarían a ser un elemento más en el hogar, la empresa y la sociedad general.

Hoy en día el diseño y la construcción de un robot es bastante excesiva además de que el propósito es muy específico, por lo que, los estudias actuales se centran en conseguir que sean más generales y económicos, de tal manera que sean accesibles a todas las personas que quieran resolver alguna problemática cotidiana. Por lo cual el diseño e implementación de este trabajo sobre un producto accesible como una tarjeta de diseño FPGA.

Este trabajo se centra en el estudio de los robots móviles con ruedas, por lo cual será el principal estudio del mismo de las características configuración y capacidades de estos [9].

Los robots móviles pueden ser clasificados de acuerdo con el medio en el que se desplacen: terrestres, marinos y aéreos.

El desarrollo de robots móviles responde a la necesidad de extender el campo de aplicación de la robótica, su importancia radica principalmente en que poseen un espacio de trabajo ilimitado, a diferencia de los robots manipuladores fijos los cuales están restringidos a cumplir tareas dentro de un espacio de trabajo determinado por sus dimensiones físicas. Por lo tanto, con la finalidad de aumentar la movilidad del robot y de esta manera su capacidad de trabajo, se hace uso de un sistema locomotor para que el robot pueda desplazarse libremente en su espacio de trabajo. Además, estos robots tienen la capacidad de adaptarse a una gran diversidad de terrenos y actuar en ambientes no estructurados.

El término autonomía hace referencia a la capacidad del robot para responder ante situaciones cambiantes, ambiguas o impredecibles sin necesitad de supervisión humana, sugiere el paso de robots costosos y poco flexibles a robots ligeros y con mayor capacidad de adaptación a diferentes tareas. Los robots móviles se caracterizan por su capacidad de desplazarse en forma autónoma en un entorno desconocido o conocido parcialmente. Sus aplicaciones cubren una gran variedad de campos, entre los cuales se incluyen trabajos subterráneos (minería, construcción de túneles, etc.), misiones espaciales y exploración planetaria (recolección de muestras, mantenimiento de estaciones orbitales, etc.), vigilancia e intervención de seguridad (desactivación de explosivos, operación en zonas radioactivas, etc.), aplicaciones militares, entre otras. En todas estas aplicaciones la justificación más importante para el uso de la robótica móvil es la dificultad o imposibilidad de intervención humana, ya sea de manera directa o tele operada.

Los robots móviles se pueden clasificar de acuerdo con el tipo de locomoción utilizado. En general, los tres sistemas de locomoción más conocidos son: ruedas, patas y orugas. Es importante señalar que aunque la locomoción por patas y orugas han sido ampliamente estudiadas, el mayor desarrollo se presenta en los robots móviles con ruedas, por ello el presente capítulo se dedica al estudio de este tipo de robots móviles.

2.2 Robot móvil con ruedas (RMR)

Es un vehículo capaz de moverse de manera autónoma sobre una superficie, mediante la acción de las ruedas montadas en el robot. Es importante señalar que se considera que el contacto entre cada rueda y la superficie es solamente un punto de contacto de rodado. Una diferencia esencial entre los RMR´s y otros sistemas robóticos son las restricciones cinemáticas entre las ruedas y la superficie sobre la cual se desplazan. Los RMR´s constituyen una clase de sistemas mecánicos caracterizados por restricciones cinemáticas no-holonómicas, entre las velocidades en el punto de contacto y las velocidades

angulares, las cuales no son integrables en forma de relaciones algebraicas entre las variables de desplazamiento trasnacional y rotacional, por lo tanto no pueden ser eliminadas de las ecuaciones del modelo. El resultado de este problema de integrabilidad de las restricciones es la falta de una relación uno-a-uno entre las variables cartesianas y las variables articulares. Esto significa que mientras que los desplazamientos angulares en manipuladores seriales determinan la posición y orientación de un efector final, el desplazamiento angular de las ruedas de un RMR no determina la posición y orientación del cuerpo del vehículo.

2.3 Robots móviles con ruedas

Los robots móviles emplean diferentes tipos de locomoción mediante ruedas, las cuales les confieren características y propiedades diferentes respecto a la eficiencia energética, dimensiones, cargas útiles y maniobrabilidad [1].

2.3.1 Descripción de un robot móvil con ruedas.

Como el desplazamiento de los RMR se realiza mediante el accionamiento de ruedas, su energía potencial para desplazamiento en superficies planas y horizontales, donde los ejes de tracción son paralelos a la superficie, puede ser constate y su deslizamiento reducirse a cero.

Con este trabajo se pretende demostrar, a través del análisis cinemático de las diferentes arreglos de tracción y mediante el diseño, construcción y evaluación de un RMR lo antes planteado, se aborda el caso bidimensional debido a que el robot se mueve en un plano, así el problema se reduce a encontrar la terna (x, y, θ) asociada al sistema de referencia móvil del vehículo, donde las dos primeras componentes corresponden a la traslación y la tercera a la orientación del RMR (Figura 2.1).

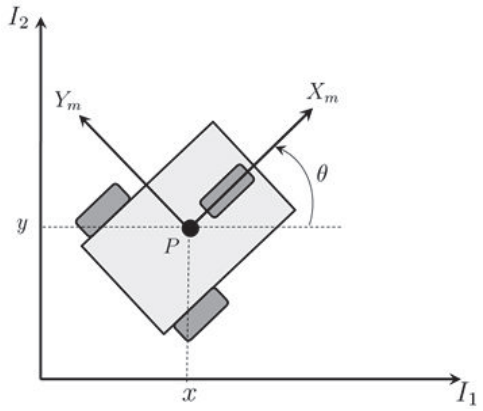


Figura 2.1 Coordenadas de la postura de un robot móvil.

La localización del robot en el plano puede ser descrita de la siguiente forma: se establece un sistema inercial arbitrario (I_1, I_2) fijo al plano del movimiento, mientras que un punto de referencia *P* en el robot representa el origen del sistema arbitrario (X_m, Y_m), de tal manera que la postura del robot queda completamente especificada por las coordenadas generalizadas (x, y, θ).

El par (x, y) representa las coordenadas generalizadas del punto de referencia *P* respecto al sistema inercial, es decir,

$$\overrightarrow{OP} = x\overrightarrow{I_1} + y\overrightarrow{I_2}, \tag{2.1}$$

Mientras que θ describe la orientación del sistema (X_m,Y_m) con respecto al sistema inercial (I_1, I_2).

Por lo tanto la postura del robot puede ser determinada por el vector

$$\xi = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \tag{2.2}$$

mientras que la matriz de rotación que define la orientación del sistema (X_m, Y_m) con respecto al sistema (I_1, I_2), tomando en cuenta que los ejes Z de ambos sistemas son paralelos, está dada por:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & sen(\theta) & 0 \\ -sen(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

2.3.2 Tipos de ruedas.

Existen dos tipos básicos de ruedas, las ruedas convencionales y las ruedas suecas. Con base en la clasificación presentada por *Camption et al*., las ecuaciones de restricción para cada uno de los tipos de ruedas están dadas por las siguientes relaciones cinemáticas.

Rueda fija

El centro de la rueda es el punto fijo A, cuya posición respecto al marco de referencia móvil puede ser obtenida en coordenadas polares, es decir, la distancia l de A a P y el ángulo α. La orientación del plano de la rueda con respecto a l está representada por el ángulo constante β (figura 2.2). Las ecuaciones de restricción para este tipo de rueda están dadas por:

- *A lo largo del plano de la rueda*

$$[-\operatorname{sen}(\alpha+\beta) \quad \cos(\alpha+\beta) \quad l\cos(\beta)]R(\theta)\dot{\xi} + r\dot{\varphi} = 0 \tag{2.4}$$

- *Ortogonal al plano de la rueda*

$$[\cos(\alpha+\beta) \quad \operatorname{sen}(\alpha+\beta) \quad l\operatorname{sen}(\beta)]R(\theta)\dot{\xi} = 0 \tag{2.5}$$

Rueda de centro orientable

Una rueda con centro orientable es aquella en la que el movimiento del plano de la rueda, con respecto a su marco de referencia, es una rotación $\beta(t)$ alrededor del eje vertical (figura 2.3), pasando a través del centro de la rueda. Sus ecuaciones de restricción son:

- *A lo largo del plano de la rueda*

$$[-\operatorname{sen}(\alpha+\beta) \quad \cos(\alpha+\beta) \quad l\cos(\beta)]R(\theta)\dot{\xi} + r\dot{\varphi} = 0 \tag{2.6}$$

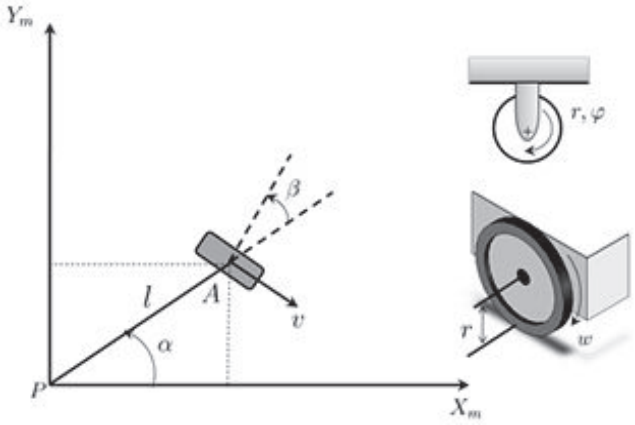


Figura 2.2 Rueda convencional fija.

- *Ortogonal al plano de la rueda*

$$[\cos(\alpha+\beta) \quad \operatorname{sen}(\alpha+\beta) \quad l\operatorname{sen}(\beta)]R(\theta)\dot{\xi} = 0 \tag{2.7}$$

Es importante hacer notar que la descripción es la misma que para una rueda fija, excepto porque ahora el ángulo de orientación β no es constante.

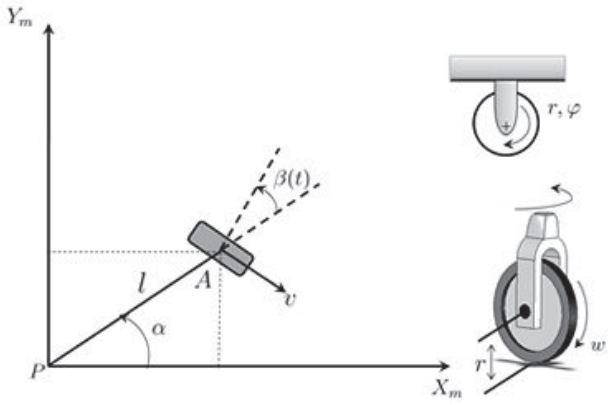


Figura 2.3 Rueda convencional de centro orientable.

Rueda de centro orientable desplazado (giratoria)

Una rueda de centro orientable desplazado es también una rueda orientable con respecto al marco de referencia, sin embargo la rotación del plano de la rueda es alrededor de un eje vertical que no pasa a través del centro de la rueda. El centro de la rueda es el punto B y está conectada al punto A mediante una varilla rígida, de longitud constante d, la cual puede rotar alrededor del eje vertical que pasa por el punto A (figura 2.4). Sus ecuaciones de restricción son:

- *A lo largo del plano de la rueda*

$$[-sen(\alpha+\beta) \quad \cos(\alpha+\beta) \quad lcos(\beta)]R(\theta)\dot{\xi} + r\dot{\varphi} = 0 \tag{2.8}$$

- *Ortogonal al plano de la rueda*

$$[\cos(\alpha+\beta) \quad sen(\alpha+\beta) \quad d + lsen(\beta)]R(\theta)\dot{\xi} + d\dot{\beta} = 0 \tag{2.9}$$

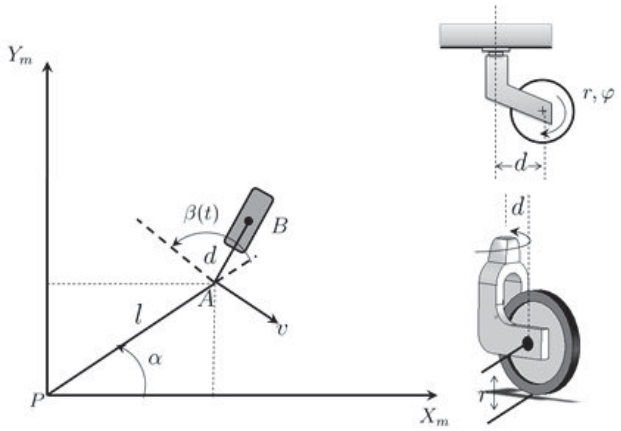


Figura 2.4 Rueda convencional de centro orientable desplazado.

Rueda omnidireccional o sueca

La rueda omnidireccional es similar a una rueda convencional, pero posee muchas ruedas direccionales pequeñas que se mueven en un ángulo de 90 grados por su superficie de rodado. Mientras la rueda completa se mueve hacia adelante y hacia atrás, las ruedas pequeñas que cruzan su superficie se mueven indirectamente, por lo tanto, la rueda puede desplazarse en cuatro direcciones (figura 2.5).

La posición de la rueda con respecto al marco de referencia es descrita como en el caso de la rueda convencional fija. Sin embargo, se requiere un parámetro adicional γ para caracterizar la dirección con respecto al plano de la rueda. Esta configuración sólo tiene una restricción:

- *A lo largo del plano de la rueda*

$$[-\operatorname{sen}(\alpha+\beta+\gamma) \quad \cos(\alpha+\beta+\gamma) \quad l\cos(\beta+\gamma)]R(\theta)\dot{\xi} + r\cos(\gamma)\,\dot{\varphi} = 0 \tag{2.10}$$

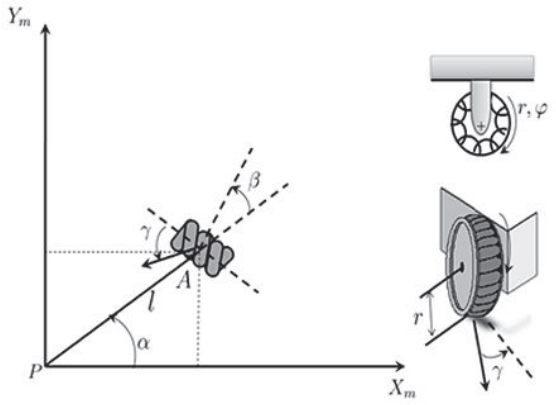


Figura 2.5 Rueda omnidireccional o sueca.

Restricciones en la movilidad de un robot

Considérese un robot móvil equipado con N ruedas de los cuatro tipos descritos con anterioridad, donde la siguiente relación representa el total de ruedas de cada clase

$$N_f + N_{co} + N_{cd} + N_o = N, \tag{2.11}$$

El subíndice f representa las ruedas fijas, co las ruedas de centro orientable, cd las ruedas de centro orientable desplazado y o las ruedas omnidireccionales.

De acuerdo con la representación anterior, las restricciones de movilidad pueden representarse mediante la siguiente forma general:

$$J_1(\beta_{co}, \beta_{cd})R(\theta)\dot{\xi} + J_2\dot{\varphi} = 0 \tag{2.12}$$

$$C_1(\beta_{co}, \beta_{cd})R(\theta)\dot{\xi} + C_2\beta_{cd} = 0 \tag{2.13}$$

De la ecuación (2.12) se tiene que

$$J_1(\beta_{co}, \beta_{cd}) = \begin{bmatrix} J_{1f} \\ J_{1co}(\beta_{co}) \\ J_{1cd}(\beta_{cd}) \\ J_{1o} \end{bmatrix} \tag{2.14}$$

Donde J_{1f}, J_{1co}, J_{1cd}, J_{1o} son matrices de dimensiones $(N_f \times 3), (N_{co} \times 3)$, $(N_{cd} \times 3)$ y $(N_o \times 3)$, cuya estructura se deriva directamente de las ecuaciones de restricción (2.4), (2.6), (2.8) y (2.10), respectivamente. Además, J_2 es una matriz diagonal de dimensión $(N \times N)$ compuesta por los radios de las ruedas, excepto para las ruedas omnidireccionales donde el radio está multiplicado por $\cos(\gamma)$.

Por otra parte, de la ecuación (2.13) se tiene que

$$C_1(\beta_{co}, \beta_{cd}) = \begin{bmatrix} C_{1f} \\ C_{1co}(\beta_{co}) \\ C_{1cd}(\beta_{cd}) \end{bmatrix} \tag{2.15}$$

$$C_2 = \begin{bmatrix} 0 \\ 0 \\ C_{2cd} \end{bmatrix} \tag{2.16}$$

Donde $C_{1f}, C_{1co} y C_{1cd}$ son matrices de dimensiones $(N_f \times 3), (N_{co} \times 3)$ y $(N_{cd} \times 3)$ formadas a partir de las ecuaciones (2.5), (2.7) y (2.9), respectivamente, mientras que C_{2cd} es una matriz diagonal de $(N_{cd} \times N_{cd})$, cuyos elementos son igual a d.

2.4 Configuración de robots móviles

Las características más significativas de los sistemas de locomoción con ruedas más comunes en robótica móvil son presentados en las siguientes secciones. Estas características se refieren a la configuración de ruedas y posiciones así como la manera de direccionamiento del robot móvil tomando en cuenta cual será el sistema es la configuración de ruedas a seguir.

2.4.1 Configuración Ackerman

Este diseño está compuesto de cuatro ruedas y proporciona una adecuada estabilidad, las ruedas direccionales no son motrices.

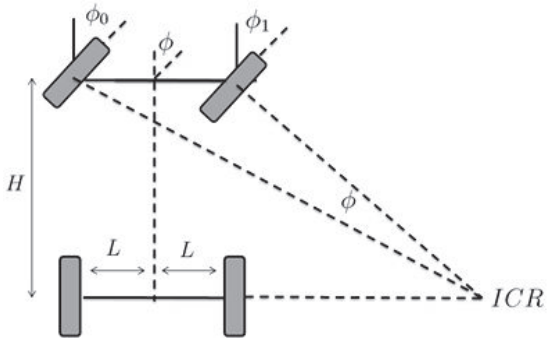


Figura 2.6 Robot móvil en configuración Ackerman.

Ackerman es el término para describir la geometría de conducción que ocasiona que la rueda delantera interior sea más firme que la rueda delantera exterior. La rueda delantera

interior gira un ángulo ligeramente superior al exterior $\phi_1 > \phi_0$ para eliminar el deslizamiento (figura 2.6).

Al prolongar los ejes de las dos ruedas delanteras, estos intersecan en un punto sobre el eje de las ruedas traseras, y este punto se conoce como Centro Instantáneo de Rotación (ICR, por sus siglas en inglés).

2.4.2 Configuración triciclo clásico

La configuración de triciclo más común tiene una rueda delantera y dos ruedas traseras, como puede observarse en la figura 2.7; esta configuración se conoce como *delta*. La premisa principal de un triciclo es proporcionar una plataforma estable. El efecto de la base en un triciclo influye en la maniobrabilidad y la distribución de peso. La base de un triciclo es la longitud entre el eje de las ruedas traseras y la rueda delantera. Una base corta genera un radio de giro del triciclo muy pequeño, mientras que una base larga hace al radio de giro más grande (ICR). Adicionalmente, un triciclo con una base corta exhibe mayor maniobrabilidad que un triciclo con una base larga.

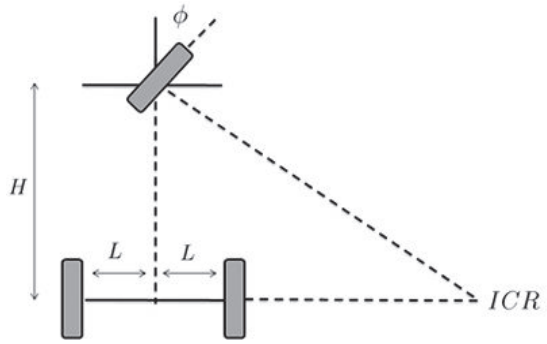


Figura 2.7 Robot móvil en configuración triciclo clásico.

2.4.3 Direccionamiento diferencial

En esta configuración, el punto (ICR) sobre el cual el robot móvil pivotea está sobre una línea perpendicular que atraviesa el centro de las ruedas (figura 2.8). El radio llega a ser mínimo cuando el punto del pivote se localiza en el punto medio de las ruedas. El espacio mínimo para que el robot gire es determinado por la distancia máxima de ese punto a cualquier otro punto en el robot móvil, normalmente la esquina delantera.

El robot puede moverse en línea recta, girar sobre sí mismo y trazar curvas. El equilibrio del robot se obtiene mediante una o dos ruedas adicionales de apoyo formando un diseño triangular o romboidal. El diseño triangular puede no ser suficiente dependiendo de la distribución de peso del robot, y el romboidal puede provocar poca adaptación al terreno si éste es irregular, lo que puede exigir alguna clase de suspensión.

La tracción se consigue mediante las ruedas laterales y la dirección por la diferencia de velocidades de las mismas ($v_L y v_R$).

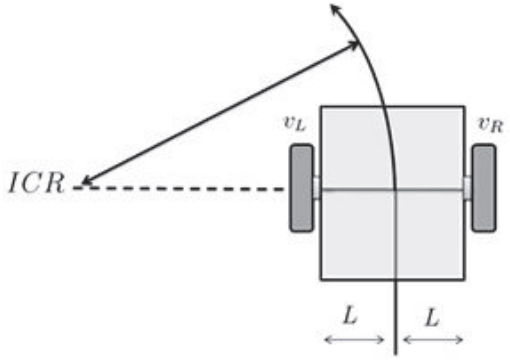


Figura 2.8 Robot móvil en configuración de direccionamiento diferencial.

2.4.4 Configuración síncrona

En este diseño todas las ruedas (generalmente tres) son tanto de dirección como de tracción; las ruedas se encuentran ubicadas de tal forma que siempre apuntan en la misma dirección. Para cambiar la dirección, el robot gira simultáneamente todas las ruedas alrededor de un eje vertical, de modo que la dirección del robot cambia, pero su chasis sigue apuntando en la misma dirección que tenía antes del giro (figura 2.10).

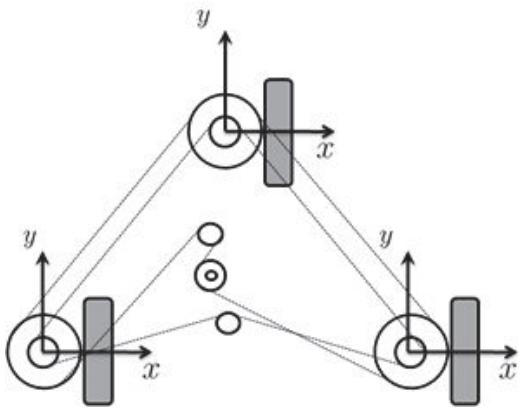


Figura 2.9 Robot móvil en configuración síncrona.

2.4.5 Movilidad y direccionalidad

El término movilidad en robótica se refiere a la habilidad que tiene un robot móvil para desplazarse o moverse con libertad en su entorno. La restricción básica de movilidad es satisfacer la condición de no deslizamiento de las ruedas. Por otra parte, la direccionalidad es intuitivamente la habilidad de un robot móvil para cambiar de direcciones durante su movimiento dentro de su entorno de trabajo. La capacidad de direccionamiento está

directamente relacionada con la cantidad de ruedas direccionales que posee un robot móvil.

Considérese un robot móvil con $N_f + N_{co}$ ruedas, para el cual las restricciones de no deslizamiento (2.13) pueden ser escritas como

$$C_{1f}R(\theta)\dot{\xi} = 0 \tag{2.17}$$

$$C_{1co}(\beta_{co})R(\theta)\dot{\xi} = 0 \tag{2.18}$$

Estas restricciones implican que el vector $R(\theta)\dot{\xi} \in \mathcal{N}(C_1^*(\beta_{co}))$ donde

$$C_1^*(\beta_{co}) = \begin{bmatrix} C_{1f} \\ C_{1c0}(\beta_{co}) \end{bmatrix} \tag{2.19}$$

Y rank$(C_1^*(\beta_{co})) \leq 3$. Sin embargo, si rank $(C_1^*(\beta_{co})) = 3$ entonces R $(\theta)\dot{\xi} = 0$ y por lo tanto cualquier movimiento en el plano es imposible.

En el rango de la matriz $C_1^*(\beta_{co})$ depende del diseño del robot móvil. El grado de movilidad δ_m de un robot móvil puede definirse como

$$\delta_m = \dim\left(\mathcal{N}\left(C_1^*(\beta_{co})\right)\right) = 3 - rago(C_1^*(\beta_{co})) \tag{2.20}$$

Por otra parte, el grado de direccionabilidad δ_d puede obtenerse a partir del rango de $C_{1co}(\beta_{co})$, es decir que

$$\delta_d = rango(C_{1co}(\beta_{co})) \tag{2.21}$$

Si un robot móvil es equipado con más de δ_d ruedas direccionables $(N_{co} > \delta_d)$, el movimiento de las ruedas extra debe ser coordinado para garantizar la existencia del ICR en cada instante.

De acuerdo con la estructura de los robots móviles, los grados de movilidad y direccionabilidad satisfacen las siguientes desigualdades

$$1 \leq \delta_m \leq 3 \tag{2.22}$$

$$0 \leq \delta_d \leq 2 \tag{2.23}$$

$$2 \leq \delta_m + \delta_d \leq 3 \tag{2.24}$$

Por lo tanto, existen solamente cinco tipos de RMR´s de acuerdo con los cinco pares de valores de δ_m y δ_d que satisfacen las desigualdades (2.22) – (2.24) y se presentan en la tabla 2.1.

Tabla 2.1 Tipos (δ_m, δ_d) de RMR´s

δ_m	3	2	2	1	1
δ_d	0	0	1	1	2

El grado de movilidad es el primer índice de maniobrabilidad, éste es igual al número de grados de libertad que pueden ser manipulados directamente desde las entradas η, sin la reorientación de las ruedas. Este número δ_m no es igual al número de grados de libertad del robot que pueden ser manipulados mediante η y ζ. De hecho este número es igual al grado de maniobrabilidad dado por

$$\delta_M = \delta_m + \delta_d \tag{2.25}$$

2.5 Cinemática de robots móviles

El movimiento de un robot móvil puede ser descrito por el siguiente sistema de ecuaciones

$$\dot{\xi} = R^T(\theta)\Sigma(\beta_{co})\eta \tag{2.26}$$

$$\dot{\beta_{co}} = \zeta \tag{2.27}$$

Donde η y ζ representan velocidades y pueden ser interpretadas como entradas de control. Las ecuaciones (2.26)-(2.27) se conocen como *modelo cinemática de postura*.

Sin embargo, es importante reconocer el efecto de las restricciones de movilidad (2.12)-(2.13) a partir de las cuales es posible obtener la evolución de las velocidades de orientación y tracción dadas por

$$\dot{\beta}_{cd} = -C_{2cd}^{-1}C_{1cd}(\beta_{cd})R(\theta)\dot{\xi} \tag{2.28}$$

$$\dot{\varphi} = -J_2^{-1}J_1(\beta_{co}, \beta_{cd})R(\theta)\dot{\xi} \tag{2.29}$$

Combinando estas ecuaciones con el modelo cinemática de postura (2.26)-(2.27), las ecuaciones para $\dot{\beta}_{cd}$ y $\dot{\varphi}$ pueden ser reescritas como

$$\dot{\beta}_{cd} = D(\beta_{cd})\Sigma(\beta_{co})\eta \tag{2.30}$$

$$\dot{\varphi} = E(\beta_{co}, \beta_{cd})\Sigma(\beta_{co})\eta \tag{2.31}$$

Donde

$$D(\beta_{cd}) = -C_{2cd}^{-1}C_{1cd}(\beta_{cd}) \tag{2.32}$$

$$E(\beta_{co}, \beta_{cd}) = -J_2^{-1}J_1(\beta_{co}, \beta_{cd}) \tag{2.33}$$

Definiendo q como el vector de coordenadas de configuración dado por

$$q = \begin{bmatrix} \xi \\ \beta_{co} \\ \beta_{cd} \\ \varphi \end{bmatrix} \tag{2.34}$$

entonces la evolución en el tiempo de este vector puede representarse en forma compacta como:

$$\dot{q} = S(q)u \tag{2.35}$$

donde

$$S(q) = \begin{bmatrix} R^T(\theta)\Sigma(\beta_{co}) & 0 \\ 0 & I \\ D(\beta_{cd})\Sigma(\beta_{co}) & 0 \\ E(\beta_{co},\beta_{cd})\Sigma(\beta_{co}) & 0 \end{bmatrix} \tag{2.36}$$

$$u = \begin{bmatrix} \eta \\ \zeta \end{bmatrix} \tag{2.37}$$

La ecuación (2.35), conocida como modelo cinemática de configuración, tiene la forma estándar del modelo cinemática de un sistema sujeto a restricciones de velocidad independientes.

2.6 Dinámica de robots móviles

Aquí se presentan las ecuaciones de movimiento que representan un modelo dinámico general en espacio de estados y describen las relaciones entre las coordenadas de configuración y los pares de los actuadores de un robot móvil con ruedas (RMR´s).

De acuerdo con la formulación de Euler-Lagrange, la dinámica de los RMR´s puede ser escrita por las siguientes ecuaciones de movimiento:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{K}}{\partial \dot{\xi}}\right)^T - \left(\frac{\partial \mathcal{K}}{\partial \xi}\right)^T = R^T(\theta)J_1^T(\beta_{co},\beta_{cd})\lambda + R^T(\theta)C_1^T(\beta_{co},\beta_{cd})\mu \tag{2.38}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{K}}{\partial \dot{\beta}_{cd}}\right)^T - \left(\frac{\partial \mathcal{K}}{\partial \beta_{cd}}\right)^T = C_2^T\mu + \tau_{cd} \tag{2.39}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{K}}{\partial \dot{\varphi}}\right)^T - \left(\frac{\partial \mathcal{K}}{\partial \varphi}\right)^T = J_2^T\lambda + \tau_{\varphi} \tag{2.40}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{K}}{\partial \dot{\beta}_{co}}\right)^T - \left(\frac{\partial \mathcal{K}}{\partial \beta_{co}}\right)^T = \tau_{co} \tag{2.41}$$

Donde $\mathcal{K}$ representa la energía cinética, λ y μ son multiplicadores de Lagrange asociados a las restricciones (2.12) y (2.13), τ_{φ} representa los pares de las ruedas de tracción, τ_{cd} los pares de los actuadores de orientación de las ruedas de centro orientable desplazado, y finalmente τ_{co} los pares para la orientación de las ruedas de centro orientable.

Con la finalidad de eliminar los multiplicadores de Lagrange de las ecuaciones de movimiento (2.38), (2.39) y (2.40), éstas deben pre multiplicarse por $\Sigma^T(\beta_{co})R(\theta)D(\beta_{cd})$, y $\Sigma^T(\beta_{co})E(\beta_{co},\beta_{cd})$, respectivamente, y después sumándolas se obtiene que

$$\Sigma^T(\beta_{co})R(\theta)[\mathcal{K}]_{\xi}D(\beta_{cd})[\mathcal{K}]_{\beta_{cd}} + E(\beta_{co},\beta_{cd})[\mathcal{K}]_{\varphi} = \Sigma^T(\beta_{co})\Big[D^T(\beta_{cd})_{\tau_{cd}} + E^T(\beta_{co},\beta_{cd})_{\tau_{\varphi}}\Big] \tag{2.42}$$

$$[\mathcal{K}]_{\beta_{co}} = \tau_{co} \tag{2.43}$$

Donde

$$[\mathcal{K}]_{\xi} = \frac{d}{dt}\left(\frac{\partial \mathcal{K}}{\partial \dot{\xi}}\right)^T - \left(\frac{\partial \mathcal{K}}{\partial \xi}\right)^T \tag{2.44}$$

$$[\mathcal{K}]_{\beta_{cd}} = \frac{d}{dt}\left(\frac{\partial \mathcal{K}}{\partial \dot{\beta}_{cd}}\right)^T - \left(\frac{\partial \mathcal{K}}{\partial \beta_{cd}}\right)^T \tag{2.45}$$

$$[\mathcal{K}]_{\varphi} = \frac{d}{dt}\left(\frac{\partial \mathcal{K}}{\partial \dot{\varphi}}\right)^T - \left(\frac{\partial \mathcal{K}}{\partial \varphi}\right)^T \tag{2.46}$$

$$[\mathcal{K}]_{\beta_{co}} = \frac{d}{dt}\left(\frac{\partial \mathcal{K}}{\partial \dot{\beta}_{co}}\right)^T - \left(\frac{\partial \mathcal{K}}{\partial \beta_{co}}\right)^T \tag{2.47}$$

La energía cinética de un RMR puede expresarse como

$$\mathcal{K} = \dot{E}^T R^T(\theta)\big[M(\beta_{cd})R(\theta)\dot{\xi} + 2V(\beta_{cd})\dot{\beta}_{cd} + 2W\beta_{co} + \dot{\beta}_{cd}^T I_{cd}\dot{\beta}_{cd} + \dot{\varphi}^T I_{\varphi} + \dot{\beta}_{co}^T I_{co}\dot{\beta}_{co}\big] \tag{2.48}$$

Con definiciones apropiadas de las matrices $M(\beta_{cd})$, $V(\beta_{cd})$, W, I_{cd}, I_{φ} e I_{co} de acuerdo con la distribución de masa del robot y los momentos de inercia de los componentes del mismo.

2.7 Control de robots móviles

El control de RMR´s continúa generando muchos desafíos o retos de investigación y desarrollo. Algunos objetivos de control son: seguimiento de trayectorias, estacionamiento de vehículos, evasión de obstáculos y colisiones, cooperación con otros robots, así como el control de formación o escolta. El problema de control más desafiante en robótica móvil consiste en diseñar leyes de control por retroalimentación que puedan estabilizar a un RMR alrededor de un punto de equilibrio. La problemática radica en que un robot no-holonómico no puede ser estabilizado por una retroalimentación de estados suave.

Esta sección se concentra en el control de seguimiento de RMR´s que resulta ser un problema de mayor importancia práctica que la regulación (estabilización alrededor de un punto de equilibrio). El control de seguimiento consiste en la estabilización alrededor de

una trayectoria y se conoce como seguimiento estable de un movimiento de referencia. Es importante mencionar que el problema de seguimiento es más fácil de resolver que problema de regulación de RMR´s. Existen diversos problemas de control que se relacionan con tareas de seguimiento para robots móviles; a continuación se describen algunos.

El problema de seguimiento de postura consiste en diseñar una ley de control por retroalimentación estable que permita el seguimiento de una postura de referencia $\xi_r(t)$ variante en el tiempo, la cual se asume que es diferenciable al menos dos veces. En términos más formales, consiste en encontrar una ley de control v acotada que logre que el error de seguimiento $\tilde{\xi}(t) = \xi(t) - \xi_r(t)$ permanezca acotado y converja hacia cero en forma asintótica, es decir;

$$\lim_{t\to\infty} \tilde{\xi}(t) = 0 \tag{2.49}$$

Por lo tanto, si $\xi(0) = \xi_r(0)$ entonces $\xi(t) = \xi_r(t) \forall t$

En algunas situaciones particulares no es necesario realizar el seguimiento de la postura completa del robot, por lo tanto resulta suficiente controlar solamente la posición de un punto fijo *P´* en la estructura del robot (figura 2.10). Las coordenadas del punto están dadas por

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x + e\ \cos(\theta + \delta) \\ y + e\ \operatorname{sen}(\theta + \delta) \end{bmatrix} \tag{2.50}$$

El problema de *seguimiento de un punto* es encontrar un controlador estable por retroalimentación de estados que logre el seguimiento de una posición de referencia $x_r'(t), y_r'(t)$ móvil dada y la cual se asume doblemente diferenciable. En términos formales, diseñar una ley de control $v(t)$ acotada que permita que los errores de seguimiento $\tilde{x}'(t) = x'(t) - x_r'(t),\ \tilde{y}'(t) = y'(t) - y_r'(t)$ tiendan asintóticamente hacia cero, es decir;

$$\lim_{t\to\infty} \tilde{x}'(t) = 0 \tag{2.51}$$

$$\lim_{t\to\infty} \tilde{y}'(t) = 0 \tag{2.52}$$

Por lo tanto, si $x'(0) = x_r'(0)$ y $y'(0) = y_r'(0)$ entonces $x'(t) = x_r'(t)$ y $y'(t) = y_r'(t) \forall t$

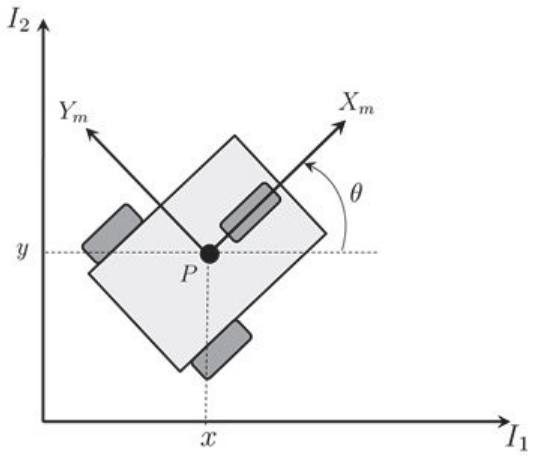


Figura 2.10 Coordenadas del punto *P'* en un robot móvil.

2.8 Simulación con MatLab

Considérese el robot móvil tipo (2,0) conocido como *uniciclo* y cuyo sistema de locomoción es de tipo diferencial. Se realizó la simulación en MatLab [4] del modelo cinemático de postura de dicho robot considerando tres casos:

1. $\dot{\varphi}_L \neq \dot{\varphi}_R$
2. $\dot{\varphi}_L = -\dot{\varphi}_R$
3. $\dot{\varphi}_L = \dot{\varphi}_R$

donde $\dot{\varphi}_R$ representa la velocidad angular de la rueda derecha y $\dot{\varphi}_L$ representa la velocidad angular de la rueda izquierda.

Para el robot tipo (2, 0), el modelo cinemático de postura está dado por

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ sen(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

donde v y ω representan las velocidades lineal y angular, respectivamente del robot móvil. Para el caso del direccionamiento diferencial dichas velocidades están dadas por:

$$v = \frac{r}{2}(\dot{\varphi}_R + \dot{\varphi}_L)$$
$$\omega = \frac{r}{L}(\dot{\varphi}_R - \dot{\varphi}_L)$$

donde r representa el radio de ambas ruedas y L la distancia desde el centro de la rueda derecha hasta el centro de la rueda izquierda. Para simular numéricamente en MatLab el modelo cinemático se consideró $r = 3\ cm$ y $L = 10\ cm$.

Para el presente trabajo de tesis y como modelado del robot móvil tipo uniciclo cuyo sistema de locomoción es del tipo diferencial, se hicieron las simulaciones en MatLab y su código se muestra en el Apéndice C.

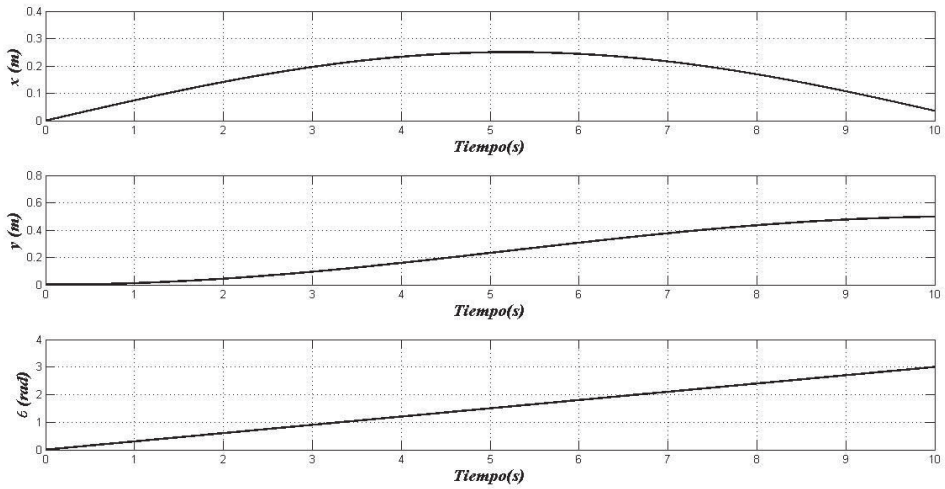


Figura 2.11 Postura de un robot uniciclo tipo (2,0) para el caso: $\dot{\varphi}_L \neq \dot{\varphi}_R$

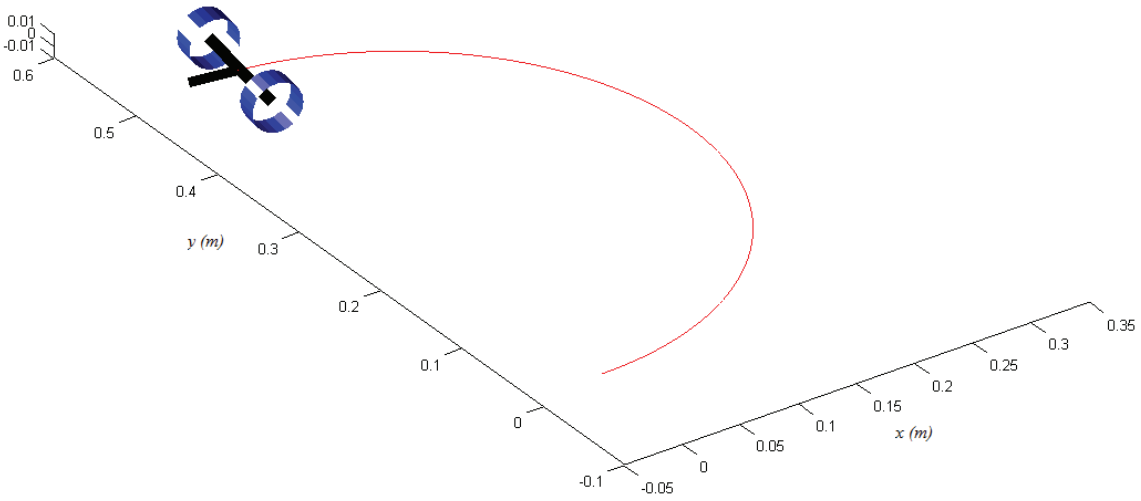


Figura 2.12 Trayectoria de un robot uniciclo tipo (2,0) para el caso: $\dot{\varphi}_L \neq \dot{\varphi}_R$

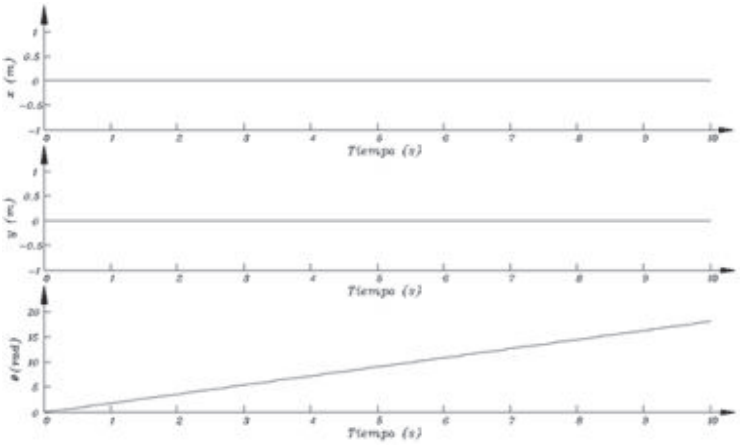


Figura 2.13 Trayectoria de un robot uniciclo tipo (2,0) para el caso: $\dot{\varphi}_L = -\dot{\varphi}_R$

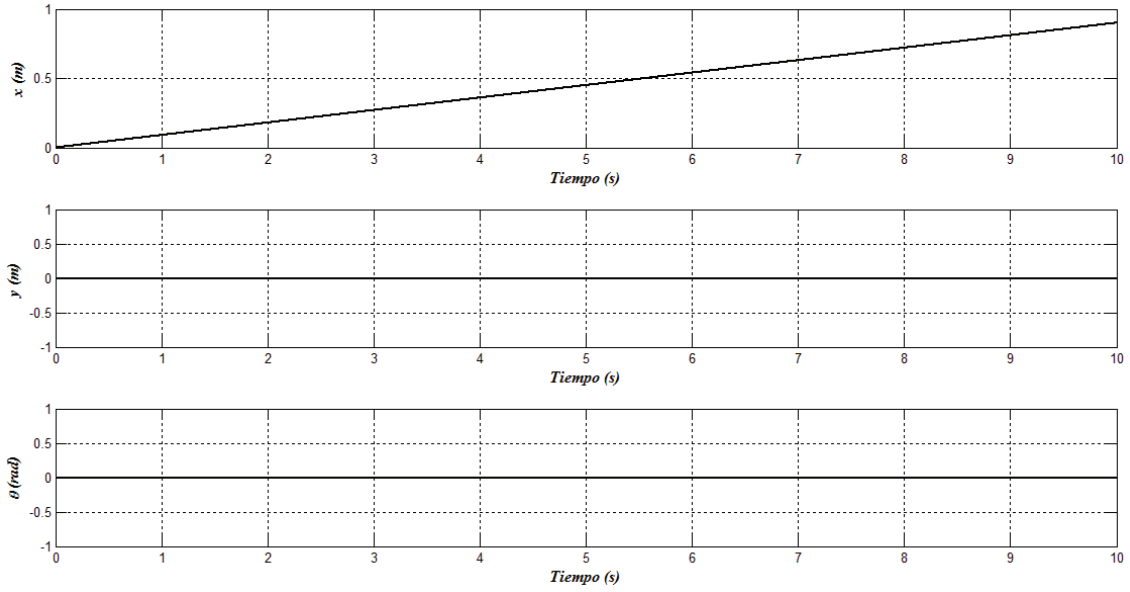


Figura 2.14 Postura de un robot uniciclo tipo (2,0) para el caso: $\dot{\varphi}_L = \dot{\varphi}_R$

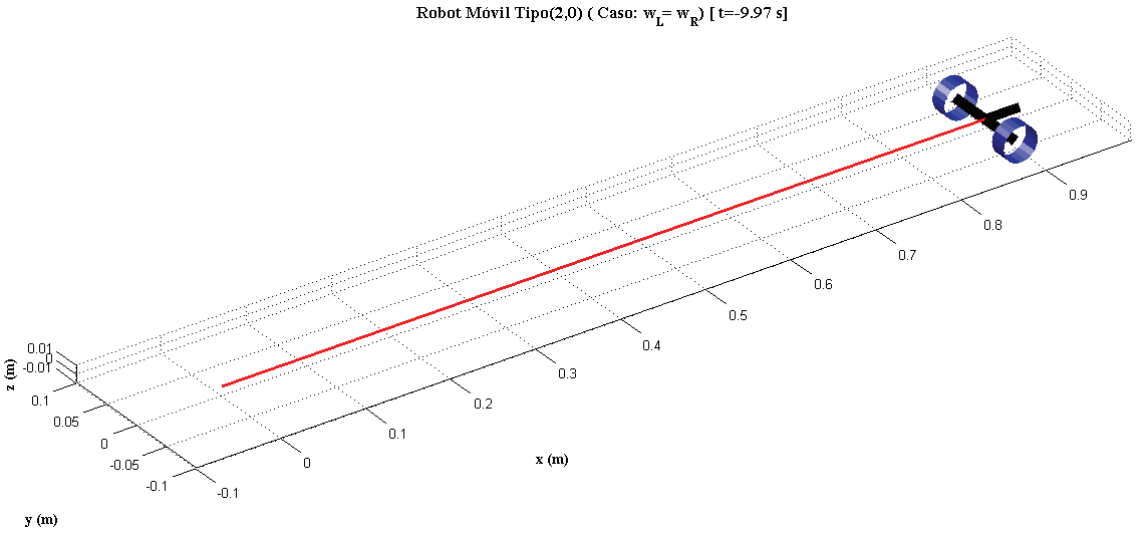


Figura 2.15 Trayectoria de un robot uniciclo tipo (2,0) para el caso: $\dot{\varphi}_L = \dot{\varphi}_R$

Los resultados obtenidos en la simulación se presentan en las figuras 2.11 a la 2.15. En el primer caso se consideró $\dot{\varphi}_R = 3\,rad/s$ y $\dot{\varphi}_L = 2\,rad/s$. La figura 2.11 representa la evolución en el tiempo de la postura del robot y se aprecia que las tres componentes(x, y, θ) muestran un comportamiento variable debido a la diferencia de las velocidades de ambas ruedas. Por lo tanto, el robot móvil debe describir un movimiento de tipo circular, lo cual se verifica en la figura 2.12.

Para el segundo caso de análisis se consideró $\dot{\varphi}_R = 3\,rad/s$ y $\dot{\varphi}_L = -3\,rad/s$. La figura 2.14 muestra la evolución en el tiempo de la postura del robot y se aprecia que las componentes lineales (x, y) permanecen constantes en la condición inicial $(0, 0)$, mientras que la componente angular θ presenta un comportamiento variable debido a la diferencia en el sentido de las velocidades de ambas ruedas. Por lo tanto, el robot móvil gira sobre su propio eje.

Finalmente para el tercer caso se consideró $\dot{\varphi}_R = \dot{\varphi}_L = 3\,rad/s$. La figura 2.15 presenta la evolución en el tiempo de la postura del robot y se aprecia que la componente x varía con el tiempo, mientras que las componentes y y θ permanecen constantes en las condiciones iniciales iguales a cero, esto se debe a que las velocidades de ambas ruedas son iguales en magnitud y sentido.

2.8.1 Control por realimentación no lineal

Es bien conocido que la linealización vía retroalimentación a través de controladores comunes tiene serias limitaciones para el control de RMR´s. En particular, no permite que un robot sea estabilizado alrededor de un punto fijo en el espacio de trabajo. En esta

sección se presenta una técnica de control por retroalimentación no lineal que permite resolver el problema de seguimiento de postura.

Considérese el robot uniciclo tipo (2,0) que se muestra en la figura 2.17, cuyo modelo cinemático de postura puede expresarse en forma compacta como:

$$\dot{z} = G(z)u = \begin{bmatrix} \cos(\theta) & 0 \\ \text{sen}(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.53}$$

Donde $z = [x\ y\ z]^T$ y las entradas de control son la velocidad lineal v y la velocidad angular ω.

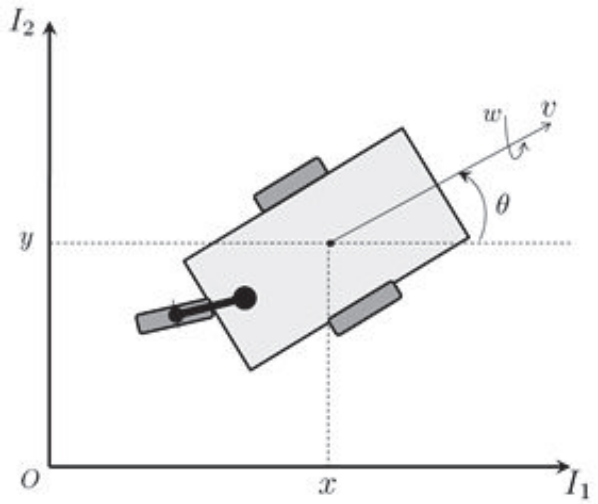


Figura 2.17 Coordenadas de postura de un robot móvil tipo (2,0).

Con la finalidad de emplear técnicas de control no lineal, es necesario realizar un cambio de coordenadas para expresar el sistema en una forma canónica, de tal manera que las nuevas coordenadas estarían dadas por:

$$\begin{aligned} x_1 &= x \\ x_2 &= \tan(\theta) \\ x_3 &= y \end{aligned} \tag{2.54}$$

Y las nuevas entradas de control serían

$$u_1 = v\cos(\theta) \tag{2.55}$$

$$u_2 = \frac{\omega}{\cos^2(\theta)} \tag{2.56}$$

Donde $\theta \in (-{}^{\pi}/_{2} + k\pi) \forall k \in Z$. Por lo tanto, el sistema (2.53) se transforma a la siguiente forma canónica de cadena

$$\begin{aligned} \dot{x}_1 &= u_1 \\ \dot{x}_2 &= u_2 \\ \dot{x}_3 &= x_2 u_1 \end{aligned} \tag{2.57}$$

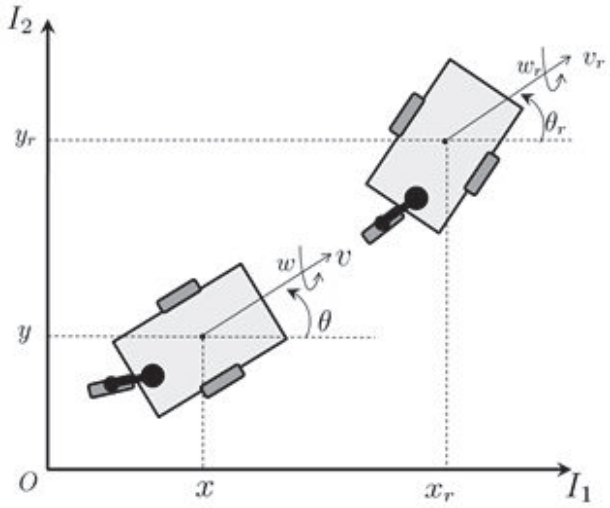


Figura 2.18 Problema de seguimiento de postura.

El problema de seguimiento de postura que se representa en la figura 2.18, consiste en encontrar una ley de control por retroalimentación

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = k(z, z_r, v_r, \omega_r) \tag{2.58}$$

De tal forma que $lím_{t\to\infty}\big(z(t) - z_r(t)\big) = 0$. Además se considera que el robot nunca está en reposo, es decir; $lím_{t\to\infty} v_r(t) \neq 0$ y $lím_{t\to\infty}\omega_r(t) \neq 0$. Para este problema de seguimiento considérese el siguiente cambio de coordenadas

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \text{sen}(\theta) & 0 \\ -\text{sen}(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix} \tag{2.59}$$

Donde e_1 y e_2 son las coordenadas del error de posición y e_3 representa el error de orientación. Además, las entradas de control están dadas por

$$u_1 = v_r \cos(e_3) - v \tag{2.60}$$

$$u_2 = \omega_r - \omega \tag{2.61}$$

Obteniéndose el siguiente sistema de ecuaciones diferenciales

$$\dot{e} = \begin{bmatrix} 0 & \omega & 0 \\ -\omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} e + \begin{bmatrix} 0 \\ \text{sen}(e_3) \\ 0 \end{bmatrix} v_r + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2.62}$$

Con la finalidad de resolver el problema de seguimiento de postura se emplea el siguiente controlador no lineal

$$u_1 = -k_1(v_r, \omega_r)e_1 \tag{2.63}$$

$$u_2 = -k_2 v_r \frac{e_2}{e_3}\text{sen}(e_3) - k_3(v_r, \omega_r)e_3 \tag{2.64}$$

Donde $k_1(v_r, \omega_r)$ y $k_3(v_r, \omega_r)$ son funciones continuas y definidas positivas, y k_2 es una constante positiva.

Capítulo 3

Sistemas Digitales Embebidos

3.1 Introducción

Los circuitos integrados son omnipresentes en gran cantidad de productos industriales. Una de sus alternativas, los circuitos tipo FPGA, presentan una característica única: están al alcance de países con un desarrollo tecnológico medio. Las FPGAs (del inglés Field-programmable Gate Arrays) aparecen en el mercado en 1984, con una idea central: permitir realizar un circuito integrado a la medida, sin los riesgos económicos asociados a las otras opciones tecnológicas. Hoy las FPGAs están presentes en campos tan diversos como la automatización, la electrónica de consumo, o la investigación espacial. La tecnología FPGA tiene una aplicación horizontal en todas las industrias que requieren computación a alta velocidad [5].

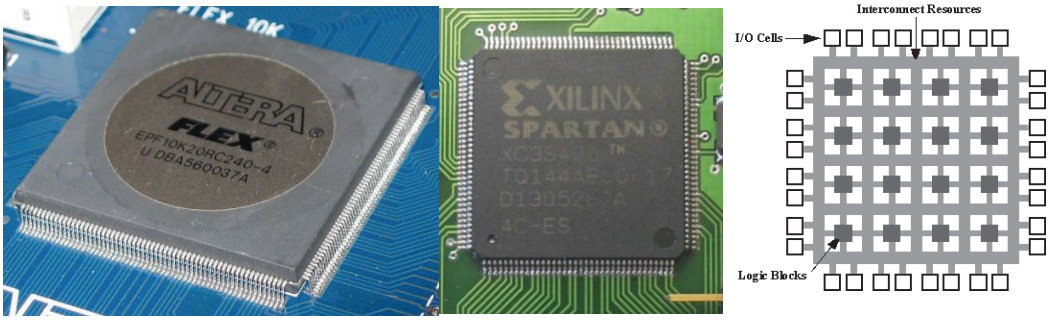
3.2 Tecnología lógica programable

Una FPGA es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad se puede programar. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip SoC (System-on-a-chip) [6].

Las FPGAs se utilizan en aplicaciones similares a los ASICs (Circuitos integrados de aplicación específica) sin embargo son más lentas, tienen un mayor consumo de potencia y no pueden abarcar sistemas tan complejos como ellos. A pesar de esto, las FPGAs tienen las ventajas de ser reprogramables (lo que añade una enorme flexibilidad al flujo de diseño), sus costos de desarrollo y adquisición son mucho menores para pequeñas cantidades de dispositivos y el tiempo de desarrollo es también menor.

Ciertos fabricantes cuentan con FPGAs que sólo se pueden programar una vez, por lo que sus ventajas e inconvenientes se encuentran a medio camino entre los ASICs y las FPGAs reprogramables.

Históricamente las FPGAs surgen como una evolución de los conceptos desarrollados en las PLAs (arreglos de compuertas programables) y los CPLDs (Dispositivos lógicos programables).En la figura 3.1 se muestran los circuitos FPGA y las celdas básicas que componen los FPGAs.



(a) (b) (c)

Figura 3.1 (a) FPGAs de marca ALTERA (b) Xilinx y (c) Celdas de un FPGA

Las FPGAs fueron inventadas en el año 1984 por Ross Freeman y Bernard Vonderschmitt, co-fundadores de Xilinx, y surgen como una evolución de los CPLDs

Tanto los CPLDs como las FPGAs contienen un gran número de elementos lógicos programables. Si medimos la densidad de los elementos lógicos programables en puertas lógicas equivalentes (número de puertas NAND equivalentes que podríamos programar en un dispositivo) podríamos decir que en un CPLD hallaríamos del orden de decenas de miles de puertas lógicas equivalentes y en una FPGA del orden de cientos de miles hasta millones de ellas.

Aparte de las diferencias en densidad entre ambos tipos de dispositivos, la diferencia fundamental entre las FPGAs y los CPLDs es su arquitectura. La arquitectura de los CPLDs más rígida y consiste en una o más sumas de productos programables cuyos resultados van a parar a un número reducido de biestables síncronos (también denominados flip-flops). La arquitectura de las FPGAs, por otro lado, se basa en un gran número de pequeños bloques utilizados para reproducir sencillas operaciones lógicas, que cuentan a su vez con biestables síncronos. La enorme libertad disponible en la interconexión de dichos bloques confiere a las FPGAs una gran flexibilidad.

Otra diferencia importante entre FPGAs y CPLDs es que en la mayoría de las FPGAs se pueden encontrar funciones de alto nivel (como sumador y multiplicador) embebidas en la propia matriz de interconexiones, así como bloques de memoria.

En este trabajo de tesis se programará un FPGA que se encuentra en una tarjeta de desarrollo (modelo DE2) para controlar un robot móvil, que generará señales digitales para activar los motores de las ruedas del robot.

3.3 Programación

La tarea del programador es definir la función lógica que realizará cada una de las Celdas Lógicas Básicas (CLB) de los FPGAs y seleccionar el modo de trabajo de cada entrada y salida para interconectarlos entre ellos.

El diseñador cuenta con la ayuda de entornos de desarrollo especializados en el diseño de sistemas a implementarse en un FPGA. Un diseño puede ser capturado ya sea como esquemático, o haciendo uso de un lenguaje de programación especial. Estos lenguajes de programación especiales son conocidos como HDL o Hardware Description Language (lenguajes de descripción de hardware) [7]. Los HDL's más utilizados son:

- VHDL
- Verilog
- ABEL

En un intento de reducir la complejidad y el tiempo de desarrollo en fases de prototipaje rápido, y para validar un diseño en HDL, existen varias propuestas y niveles de abstracción del diseño. Entre otras, National Instruments LabVIEW FPGA propone un acercamiento de programación gráfica de alto nivel.

VHDL se trata de un lenguaje de descripción de hardware, esto significa que mediante él se puede describir la forma de comportarse de un circuito electrónico. El comportamiento puede ser llevado a algún dispositivo lógico programable que dispondrá de sus propios componentes para lograr el comportamiento deseado. La forma de comportarse es independiente del hardware donde se utilice.

El VHDL es un estándar llamado IEEE 1076-1993. Sus ventajas son:

- Una disponibilidad pública
- Independencia de dispositivos y fabricantes
- Reutilización
- Diseño jerárquico

Este lenguaje más que nada es utilizado en las tarjetas FPGAs debido a la sencillez para su trabajo en ellas a la hora de declaración ya que éstos son un conjunto de bloques lógicos o celdas programables.

3.3.2 Entrada de diseño

El diseño del hardware asociado a una FPGA se puede realizar de varias maneras, existiendo la posibilidad de emplear varias de ellas simultáneamente si así se considera necesario. Para programar el hardware de un FPGA se puede realizar de la siguiente manera:

a) Por código HDL

Los lenguajes de descripción de hardware, HDL, son lenguajes de documentación de circuitos electrónicos que permiten detallar el funcionamiento y las conexiones del mismo.

Los lenguajes más extendidos son VHDL y Verilog. Ambos pueden modelar estructuras de hardware con la misma eficacia. La elección entre uno y otro suele basarse en la preferencia del diseñador, herramientas disponibles y condiciones impuestas externamente.

b) Esquemáticos

Estos describen un circuito electrónico mediante la conexión de señales y bloques. Este método para diseñar hardware presenta ventajas sobre el HDL cuando se emplea para unir bloques entre sí. Sin embargo, la descripción funcional de un bloque es más efectivo diseñarla en HDL.

c) Diagramas de estados

En el caso de existir estados en el sistema, puede resultar interesante diseñarlo mediante diagramas de estados como el mostrado en la Figura 3.2, dejando a las herramientas que generen los archivos correspondientes a partir de éstos.

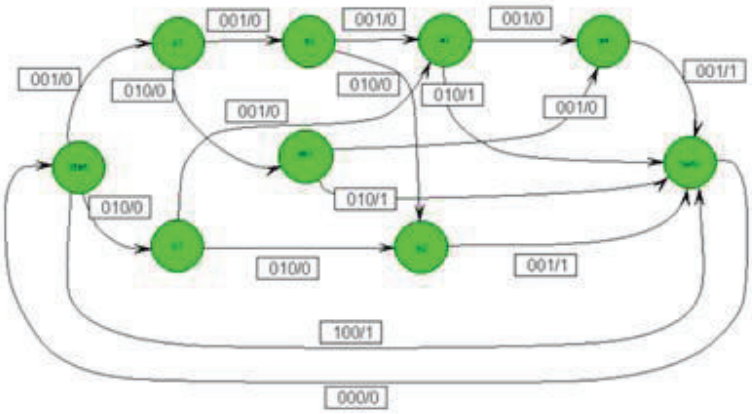


Figura 3.2 Diagrama de Estados

d) IP Cores

Los IP Cores (núcleos de propiedad intelectual) son unidades de lógica diseñadas por una entidad que dispone de derechos de propiedad intelectual sobre ésta. El uso de estos bloques simplifica y agiliza el diseño de un circuito electrónico como una FPGA o un ASIC. Ejemplos de las IP Cores son las CPUs, UARTs, interfaces PCI y controladores de Ethernet.

La finalidad de los IP Cores es minimizar la aparición de problemas durante la etapa de diseño, disminuir el time-to-market (tiempo para el mercado) del producto y reducir los costos que conlleva el desarrollo del mismo. El uso de estándares favorece la portabilidad de estos bloques. Los IPs más complejos requieren de software (en forma de APIs).

Estos se pueden clasificar en tres grandes grupos, en función de su flexibilidad y portabilidad:

- **HardCores**: cuando va ser utilizado en sistemas diferentes, debe ser posible adaptarlo sin realizar ningún cambio en el (o casi ninguno). Son por tanto muy poco flexibles, pero mure predecibles y fiables a la hora de implementarlos. incluyen información del PLACE AND ROUTE. Un buen ejemplo de este tipo son los procesadores y memorias.
- **FirmCores**: a mitad de camino entre los Hard y Soft, son bloques configurables pero que también tiene algo de información del PLACE AND ROUTE. Tienen flexibilidad limitada y predictibilidad aceptable en la implementación.
- **SoftCores**: suelen ser archivo HDL o netlist (lista de puertas lógicas y sus respectivas conexiones). Son altamente flexibles, pudiéndose personalizar de forma específica para cada aplicación. Son muy flexibles pero poco predecibles en la implementación.

e) Síntesis de diseño

Esta recopila toda la información de los archivos de entrada del diseño y los agrupa en una o varias netlist, que consisten en un listado de puestas lógicas del sistema y las conexiones entre éstas. En esta etapa es necesario conocer el dispositivo. Esta optimización es importante ya que permite aprovechar los recursos de forma más eficiente.

f) Implementación del diseño

Este es un proceso que consiste en convertir las netlist en un archivo de configuración, denominado BITSTREAM, que consiste como su propio nombre lo indica en una seria de bits de que configuran las conexiones y la lógica de la FPGA.

g) Configuración del dispositivo.

Finalmente, el BITSTREAM es descargado a la FPGA para que esta funcione de acuerdo al diseño Figura 3.3.

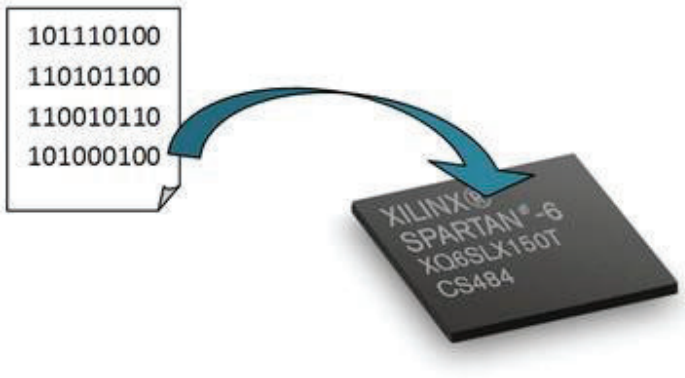


Figura 3.3 Descarga de BITSTREAM en FPGA

h) Simulación Funcional

Durante todo el proceso de diseño, es necesario ir realizando diversas simulaciones para comprobar que el diseño se ajusta a la especificaciones iníciales.

La simulación funcional consiste en comprobar que el funcionamiento del sistema es el indicado por los requisitos. Esta simulación suele realizarse habitualmente durante la entrada de diseño, aunque también se realiza en menor medida durante las fases de síntesis e implementación.

En esta simulación se asignan valores a las entradas de sistema, y se analizan las salidas, comparándolas con las salidas esperadas. Cualquier divergencia entre estas implicara la modificación del diseño.

3.4 Sistemas Embebidos

Un sistema embebido o insertado es un sistema de computación diseñado para realizar una o algunas pocas funciones dedicadas frecuentemente en un sistema de computación en tiempo real. Al contrario de lo que ocurre con los ordenadores de propósito general como lo son las computadoras personales Figura 3.4, que están diseñadas para cubrir un amplio rango de necesidades, los sistemas embebidos se diseñan para cubrir necesidades específicas. En un sistema embebido la mayoría de los componentes se encuentran incluidos en la placa base y muchas veces los dispositivos resultantes no tienen aspecto de lo que se suele asociar a una computadora. Algunos ejemplos de sistemas embebidos Figura 3.5, podrían ser dispositivos como un taxímetro, un sistema de control de acceso, la electrónica que controla una máquina expendedora o el sistema de una fotocopiadora entre otras múltiples aplicaciones.

Por lo general los sistemas embebidos se pueden programar directamente en el lenguaje ensamblador del microcontrolador o microprocesador incorporado sobre el mismo, o también, utilizando los compiladores específicos, pueden utilizarse lenguajes como C o C++; en algunos casos, cuando el tiempo de respuesta de la aplicación no es un factor crítico, también pueden usarse lenguajes interpretados como JAVA.

Puesto que los sistemas embebidos se pueden fabricar por decenas de millares o por millones de unidades, una de las principales preocupaciones es reducir costos. Los sistemas embebidos suelen usar un procesador relativamente pequeño y una memoria pequeña por ello los primeros equipos embebidos que se desarrollan fueron elaborados por IBM en los años 1980.

Los programas de sistemas embebidos se enfrentan normalmente a tareas de procesamiento en tiempo real.

Figura 3.4 Computadora Personal, PC.

Figura 3.5 Sistema Embebido.

3.5 SoC en los FPGA

System-on-chip (SoC) en una FPGA es una interesante plataforma que está empezando a aparecer más y más en todo el mercado de sistemas embebidos. Una de las razones es que integra el procesador, la memoria, la red de conexión en un solo chip, lo que provoca una menor demanda de potencia del sistema. Un SoC típico consta de un núcleo de procesador de 32 bits y muchas funciones seleccionables por un diseñador que permiten diseñar un sistema íntegramente a medida, reduciendo espacios y consumos de potencia a lo estrictamente necesario. Estas funciones incluyen la memoria, la interfaz de bus, drivers E/S, decodificadores y soporte de red. A menudo los sistemas embebidos son empleados en aplicaciones que requieren dar respuesta a eventos externos en tiempo real, por ello sistemas de tiempo-real son a menudo empleados.

Hoy muchos productos embebidos son construidos empleando varios chips, haciéndolos más grandes, más caros y con mayores requerimientos de consumo eléctrico. Los SoC han existido en largo tiempo sobre placas ASICs pero estos SoC son prácticamente nuevos sobre FPGA. Conforme las FPGA han ido mejorando la capacidad, velocidad, costo, han posibilitado la implantación de SoftProcessor, además de su capacidad intrínseca de reconfiguración que permite añadir sólo aquello que es necesitado en el diseño, éstas han ido ganando terreno a los ASICs. De tal manera que más y más compañías se están volcando en desarrollo de sistemas embebidos sobre los FPGA que hacen más fácil y económico el desarrollo y evaluación del producto, así como su futura evolución.

3.6 Flujo de diseño

Firmware es el código que define el funcionamiento de un dispositivo electrónico a bajo nivel. Este puede describir tanto el hardware del circuito, en cuyo caso vendrá dado en un lenguaje HDL o esquemáticos, o bien el software que controla dicho hardware, siendo entonces un programa desarrollado en un lenguaje como C o similar.

Existen distintos tipos de *firmware* dependiendo del dispositivo a diseñar. Aquí solo abordaremos el diseño para dispositivos de lógica programable tipo CPLD o FPGA.

3.6.1 Flujo de diseño

En la figura 3.6 se muestra un diagrama de flujo para la creación de un dispositivo electrónico

Figura 3.6 Flujo de un dispositivo Electrónico.

Lo primero que se realiza a la hora de diseñar un dispositivo electrónico es establecer de forma inequívoca su funcionalidad, establecer los requisitos que debe cumplir, etc. Una vez hecho esto se debe definir claramente qué parte de la funcionalidad se va asignar al hardware y cual al software. Se realiza el diseño tanto del hardware ya sea digital o analógico según sea lo necesario como, de existir, del software. Finalmente, se integran todas las partes y se comprueba que funciona según los requisitos.

En el hardware de diseño digital el flujo de diseño es el mostrado en la figura 3.7:

Figura 3.7 Flujo de diseño digital.

3.7 Tarjeta FPGA Cyclone II

La tarjeta FPGA Cyclone II es una tarjeta desarrolladora figura 3.5.1, la cual provee dispositivos integrados los cuales permiten al usuario el desarrollo y prueba de los diseños de rango desde simples circuitos digitales como compuertas digitales hasta el procesamiento y diseño multimedia en sus proyectos, todo sin la necesidad de implementar complejas aplicaciones para el díselo de interfaces API´s, software de control nativo o controladores de memoria flash/SRAM/SDRAM.

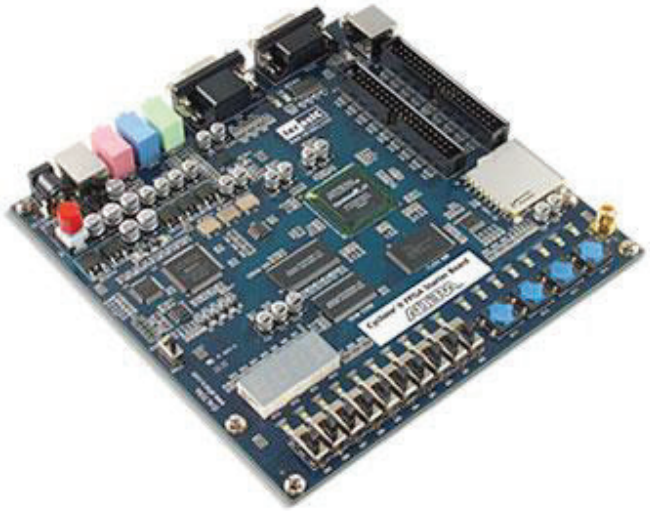


Fig3.5.1 Tarjeta Desarrolladora Cyclone II

Esta cuenta con grandes capacidades de interconexión desde botones push y switches hasta entradas de salida de video y antena para captar radio frecuencia, alguna de las capacidades de la tarjeta se enumeran a continuación:

- Cuenta con un Integrados Cyclone II EP2C20
- Puerto de Configuración Serial Altera EPCS4
- USB-Blaster que es un chip para la programación de usuario
- Módulo 512-Kbyte SRAM
- Módulo 8-Mbyte SDRAM
- Módulo 4-Mbyte de memoria Flash
- Ranura para tarjeta SD
- 4 Push botón
- 10 Switches
- 10 LED Rojos
- 8 LED Verdes
- Relojes osciladores Internos de 50Mhz, 27Mhz y 24Mhz

- CODEC de audio de 24-bit con calidad de CD de audio con entrada, salida de audio y entrada de Micrófono
- Receptor Transmisor RS-232 con conector de 9-Pins
- Conector PS/2 para teclado y mouse
- Dos conectores de 40-Pines con protección por resistencia
- Adaptador de 7.5V DC

Estas son algunos de los componentes de la tarjeta y módulos para el desarrollo figura 3.5.2.

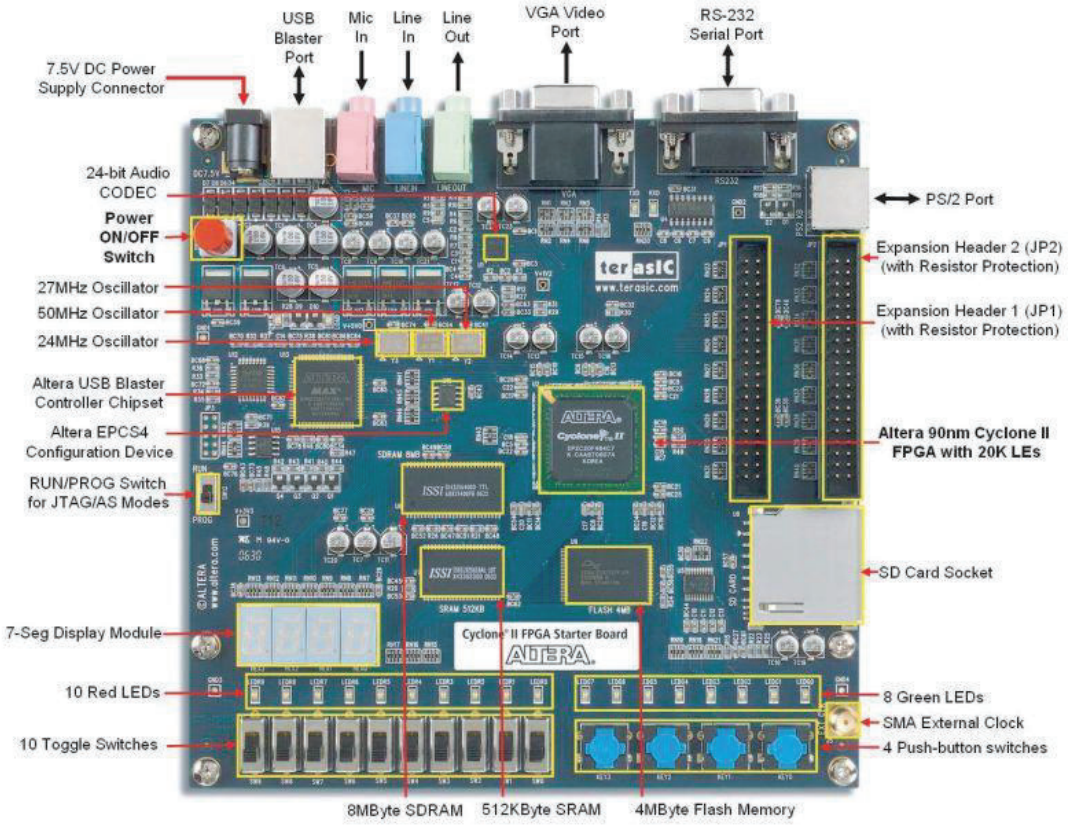


Fig3.5.2 Terminales de Tarjeta de Desarrollo.

Capítulo 4

Resultados experimentales

4.1 Introducción

Con el diseño de robots móviles existen muchos dispositivos, motores y controladores para la construcción de éstos, pero depende del criterio y las necesidades del programador para la selección de los mismos.

Uno de los primeros pasos para la construcción de un robot móvil es la elección de los tipos de motores a utilizar ya que estos definen el controlador, ya que en esta etapa se selecciona la torsión, manejabilidad y facilidad de acoplamiento a las tarjetas controladoras.

Las tarjetas controladoras pueden ser tanto dispositivos desarrolladores con múltiples controles, entradas y salidas de diferentes tipos de comunicaciones, así como también los diseñados por programadores utilizando diferentes dispositivos como microcontroladores, PLC, FPGA dependiendo del lenguaje a utilizar.

Además lo que hace llamar a estos robots móviles autónomos, es que utilizan sensores y retroalimentaciones para su posicionamiento en el entorno a conducir, siempre y cuando su aplicación lo demande. A diferencia de los robots con intervención humana que estos al detectar señales o estados fuera de su programación detienen su función y esperan acción por parte del programador.

4.2 Motores Robot móvil

Una de las necesidades al momento de diseñar un robot móvil en una tarjeta FPGA es la de utilizar motores que puedan tener un control desde el FPGA sin necesidad de intervenir o agregar algún acoplamiento electrónico.

Existen dos tipos de motores usados comúnmente en estos proyectos los motores DC (Direct Current), por sus siglas en Ingles, de corriente directa y los servomotores como los mostrados en la figura 4.1.

Motor DC Servomotor

Figura 4.1 Motores en DC y servomotores

4.2.1 Servomotor

Un servomotor es un dispositivo actuador que tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y de mantenerse estable en dicha posición. Está formado por un motor de corriente continua, una caja reductora y un circuito de control. Su margen de funcionamiento generalmente es de menos de una vuelta completa [10].

4.2.2 Motores DC

El motor de corriente continua es una máquina que convierte la energía eléctrica en mecánica, provocando un movimiento rotatorio, gracias a la acción de un campo magnético.

Una máquina de corriente continua se compone principalmente de dos partes. El embobinado da soporte mecánico al aparato y contiene los devanados principales de la máquina, conocidos también con el nombre de polos, que pueden ser de imanes permanentes o devanados con hilo de cobre sobre núcleo de hierro. El rotor es generalmente de forma cilíndrica, también devanado y con núcleo, alimentado con corriente directa mediante escobillas fijas.

El principal inconveniente de estas máquinas es el mantenimiento, muy caro y laborioso, debido principalmente al desgaste que sufren las escobillas al entrar en contacto con las la base del rotor, además de que el manejo de éstos con la tarjeta FPGA resulta ser complicado, ya que es necesario un acoplamiento de señales para poder ser controlado, porque a diferencia de los servo motores, estos necesitan un arreglo de transistores para intercambiar los voltajes para modificar la dirección [11].

4.2.3 Modulación por anchura de pulso (PWM)

La modulación por anchura de pulso, es una de las técnicas más empleadas para el control de servomotores. Este sistema consiste en generar una onda cuadrada y variar el tiempo que el pulso está a nivel alto, manteniendo el mismo período (normalmente), con el objetivo de modificar la posición del servo motor según se desee Figura 4.2.

4.2.4 Adaptando el Servo motor

Entre las modificaciones al servomotor que se utiliza en esta tesis, fue retirar el tope que evita el giro continuo y el potenciómetro que es el ajuste para las frecuencias.

Estas modificaciones son para alterar el circuito que controla la posición del servo motor, el cual indica que ha llegado donde se indica pero al no ser así éste continua con el giro continuo realizando la función deseada que es una rueda de tracción.

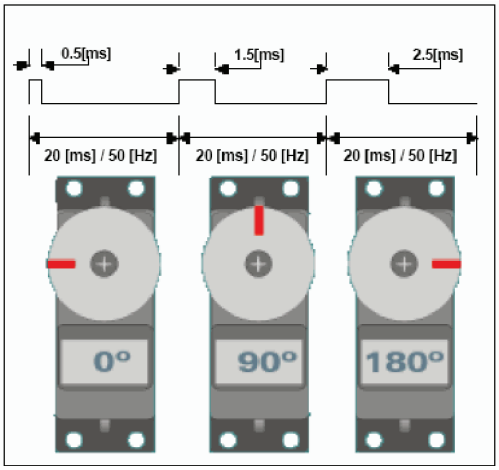
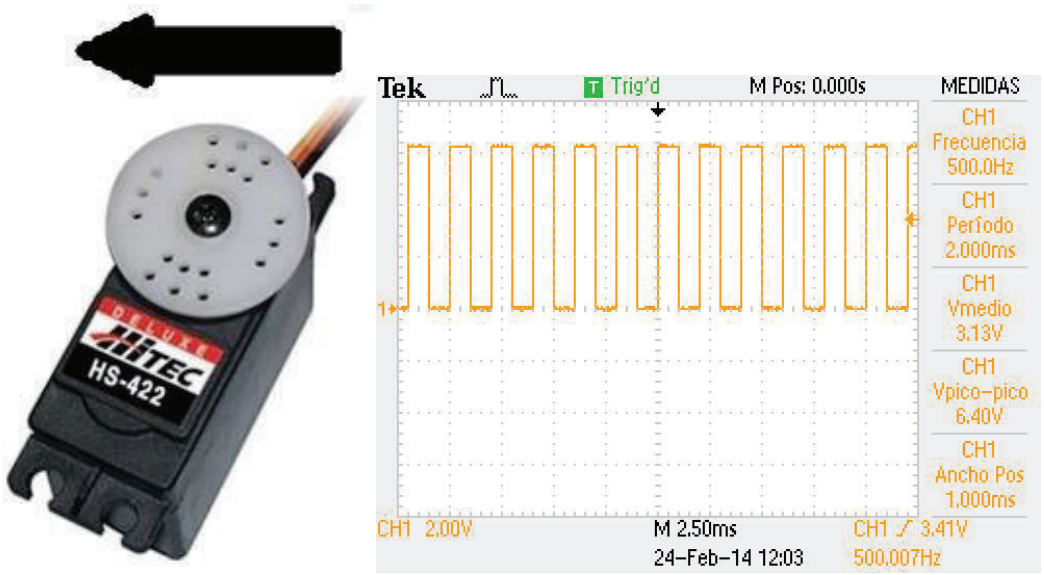


Figura 4.2 Señales PWM para control de posición.

4.3 Control de Sentido de los servos

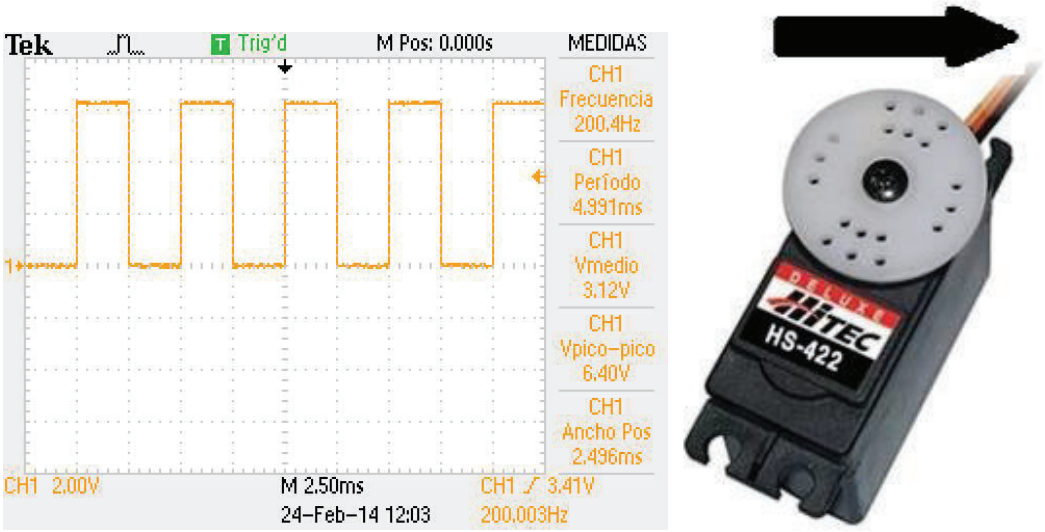
Ya con los servomotores modificados, se convierten en ruedas de tracción ya que la señal donde se indica la posición, por medio del potenciómetro ya no se encuentra presente y permite el giro continuo de los motores.

Para el control de movimiento derecho fue necesaria una frecuencia de 200Hz y para el movimiento al izquierdo una de 500Hz. La figura 4.3 (a) y (b) muestran los movimientos y las formas de onda dentro de las frecuencias generadas por nuestro PWM.



(a) (b)

Figura 4.3(a) Movimiento Izquierdo del motor con PWM, (b) señal medida con osciloscopio.



(a) (b)

Figura 4.4 (a) Movimiento Derecho del motor con PWM, (b) señal medida con osciloscopio.

4.4 Programación FPGA

De los diferentes lenguajes y software para la programación de FPGA se trabaja sobre el sistema de ALTERA Quartus II figura 4.5, este sistema ofrece una amplia gama de herramientas para el diseño así como simulación, además la posibilidad de trabajar sobre diagramas esquemáticos que son mas didactas y más fáciles de utilizar porque estos funcionan por medio de bloques o instancias que se agrupan e interconectan entre sí más fácilmente.

Figura 4.5 Software de Diseño Altera

Para generar las instancias de trabajo y poder diseñar en esquemático, es necesario la programación de estas dependiendo de la función a realizar de cada una de ellas. En este esquemático realizado para la programación se utilizaron una por cada servomotor y una más para la interfaz de entre sensores y control de motores cada una con sus simulaciones y representaciones en descripción de componentes.

Antes del diseño para el control y comportamiento del RMR es necesario realizar un diagrama de flujo el cual indique, el comportamiento a seguir dependiendo del estado de los sensores y control de las ruedas esto ayuda al momento de implementar la lógica a seguir del robot y acciones a realizar figura 4.6.

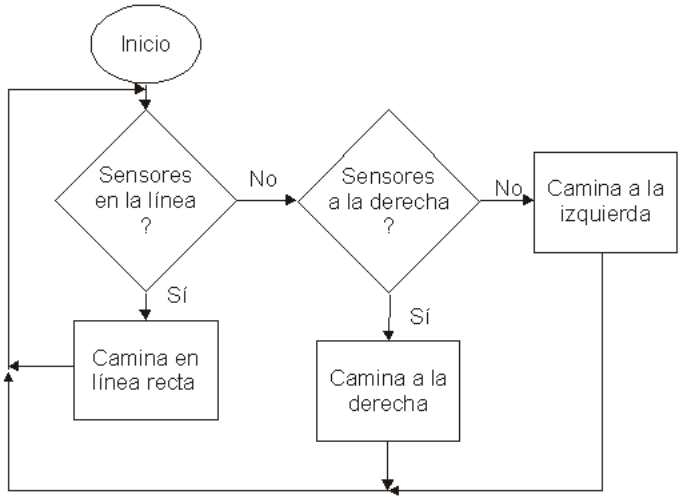
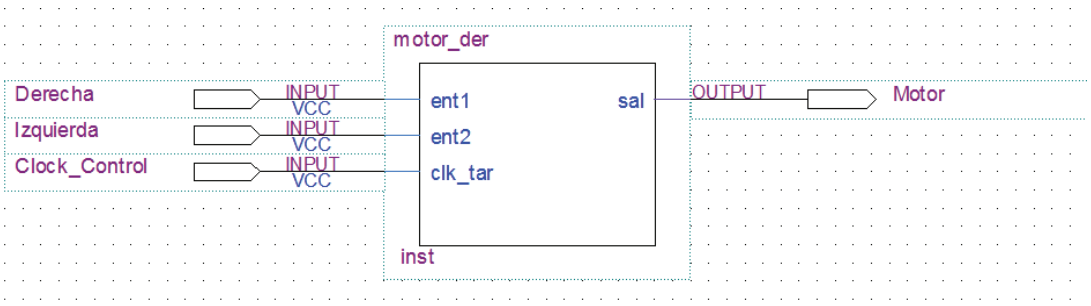
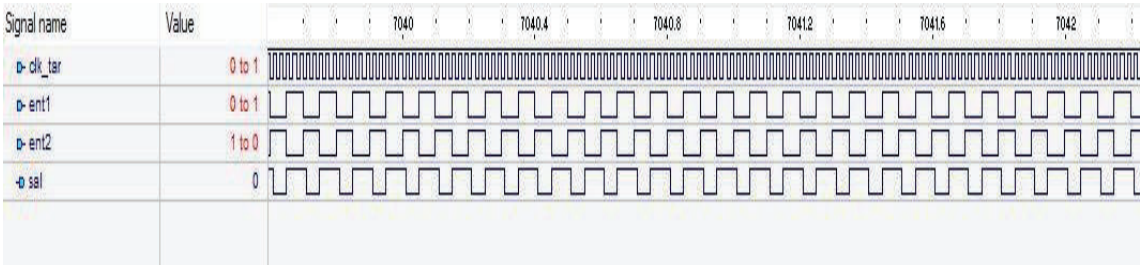


Figura4.6 Diagrama de flujo RMR.

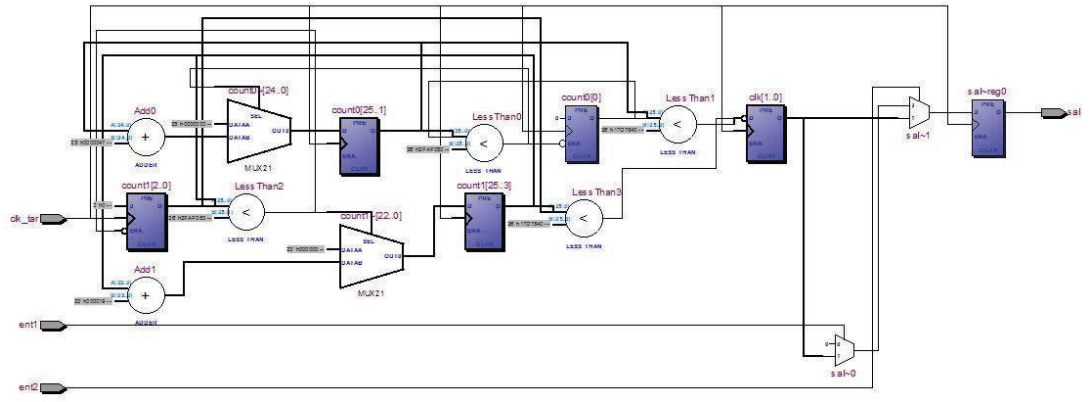
Para la instancia de control de motores se utilizó una para cada motor pero estas tienen internamente la misma programación, generan dos señales PWM a las frecuencias necesitadas para la dirección de los motores figura 4.7, (a) aquí se observa entradas y salidas. La dirección está definida por cada entrada activada y reloj es tomado del interno de la FPGA, (b) en su simulación se observan el reloj a frecuencia de la tarjeta FPGA y con comportamiento aleatorio en las entradas se observa la variación de la salida, (c) finalmente observamos la estructura interna de la instancia, como se describió el hardware dentro de la programación Apéndice 01.



(a)



(b)

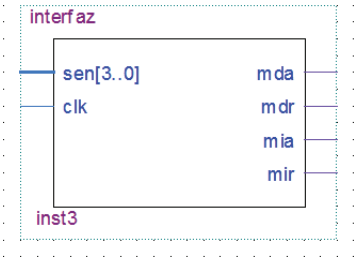


(c)

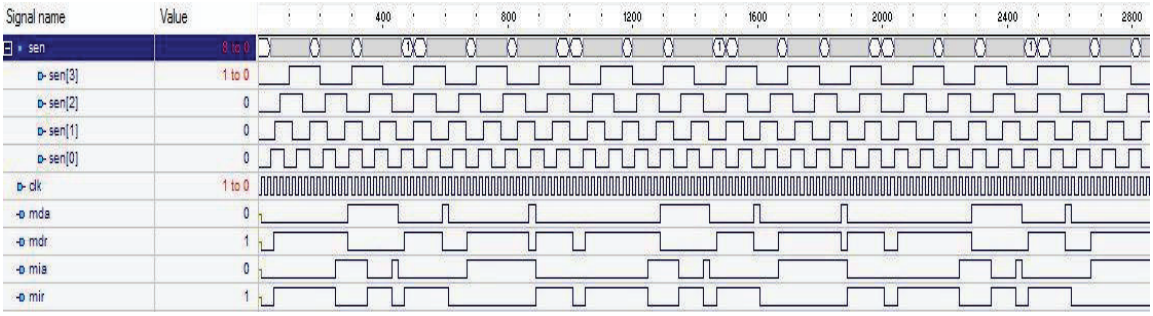
Figura 4.7 Esquemático de para Control de Motor

En la instancia de la interfaz entre motores y sensores figura 4.8, observamos cuatro salidas, una entrada bus de 4bits y una entrada de reloj la cual es la misma que en la anterior en (a) observamos la representación de la instancia en el esquemático, (b) se simulo una señal aleatoria para observar los comportamientos a diferentes respuesta de los sensores, estas salidas son la interpretación de los sensores al tomar la dirección y

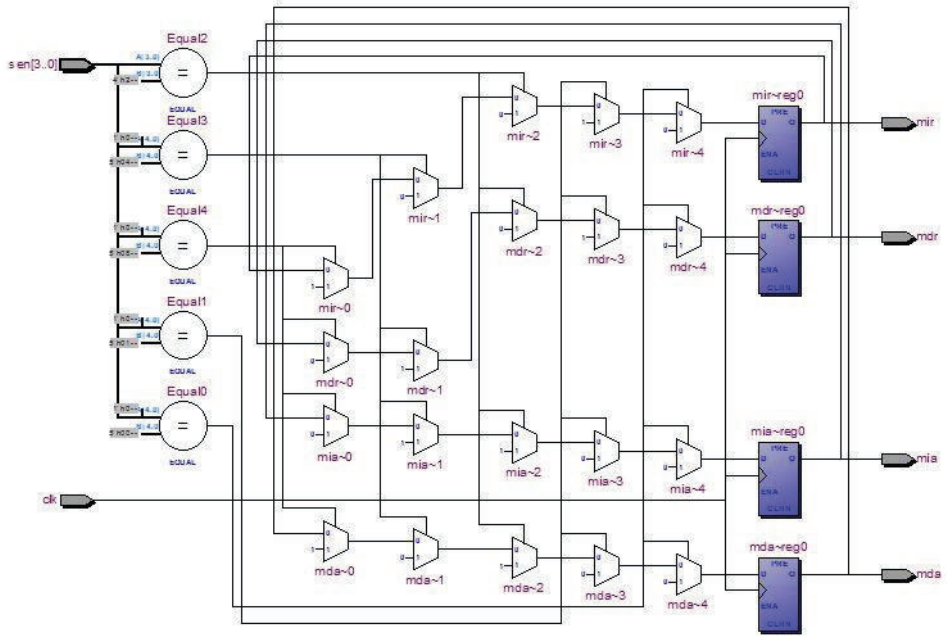
facilitamos el entendimiento de estos y por ultimo en (c) se muestra la estructura interna descrita del hardware, la lógica y los diferentes componentes utilizados durante la programación Apéndice 02.



(a)



(b)



(c)

Figura 4.8 Interfaz Entre Sensores y Motores

Esta señal será regida por dos sensores cada uno de 2 bits los cuales dependiendo de su estado se seleccionara la acción a realizar ya se retroceso o avance así como también la dirección del giro para evitar posibles conflictos.

Con los bloques ya diseñados se realiza la interconexión entre ellos estos facilita mucho el trabajo ya que se puede programar componentes individuales para el trabajo didáctico, una de las mayores ventajas es la utilización en diferentes proyectos ya que se puede diseñar un reloj el cual se puede controlar con diferentes entradas para selecciona una frecuencia deseada.

Tomando los bloques en el diseño del robot móvil se colocan los bloques y se agregan entradas y salidas para controlar los motores figura 4.9, aquí tenemos como en anteriores instancias ya nuestro diseño completo en este caso tenemos de entrada nuestro bus de 4bits y solo dos salidas una para cada motor.

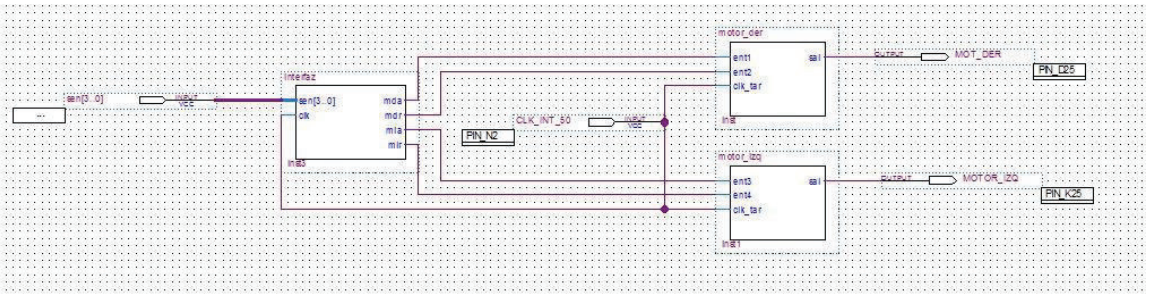


Figura 4.9 Esquemático final en el diseño

Para las pruebas en físico con la tarjeta, se utiliza la asignación de pines la cual es donde se localizan tanto entradas como salidas, para el trabajo de los servomotores y sensores figura 4.10.

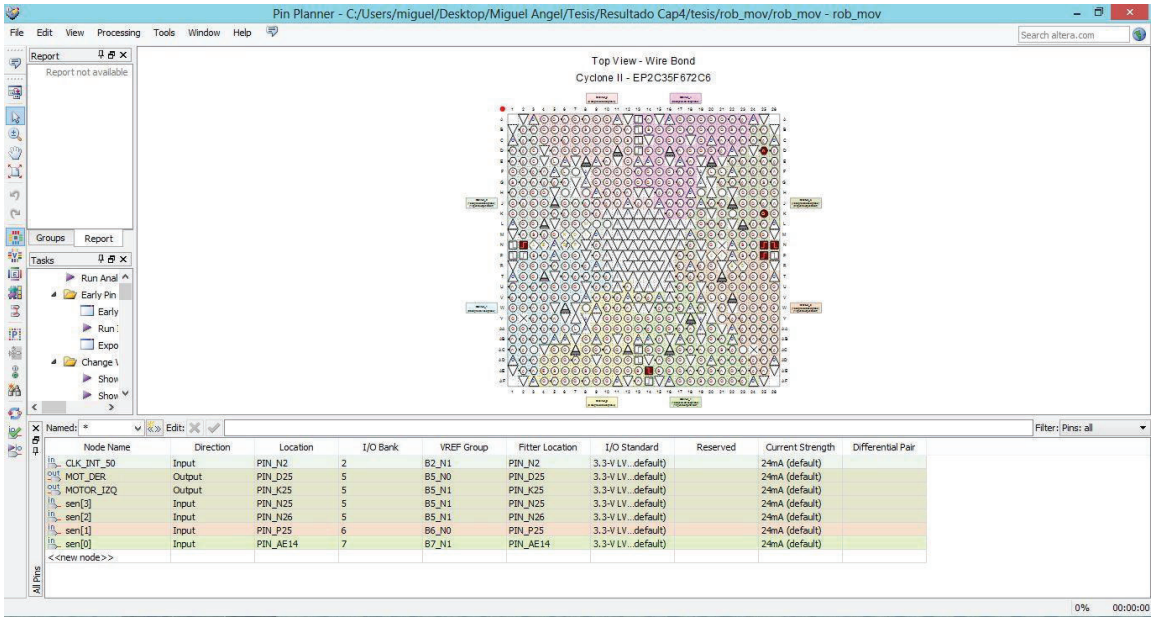


Figura 4.10 Selección de Pines en FPGA

Las pruebas se realiza sobre la tarjeta montada para observar el movimiento, para finalizar se muestran alguna imágenes con el robot figura 4.11, 4.12.

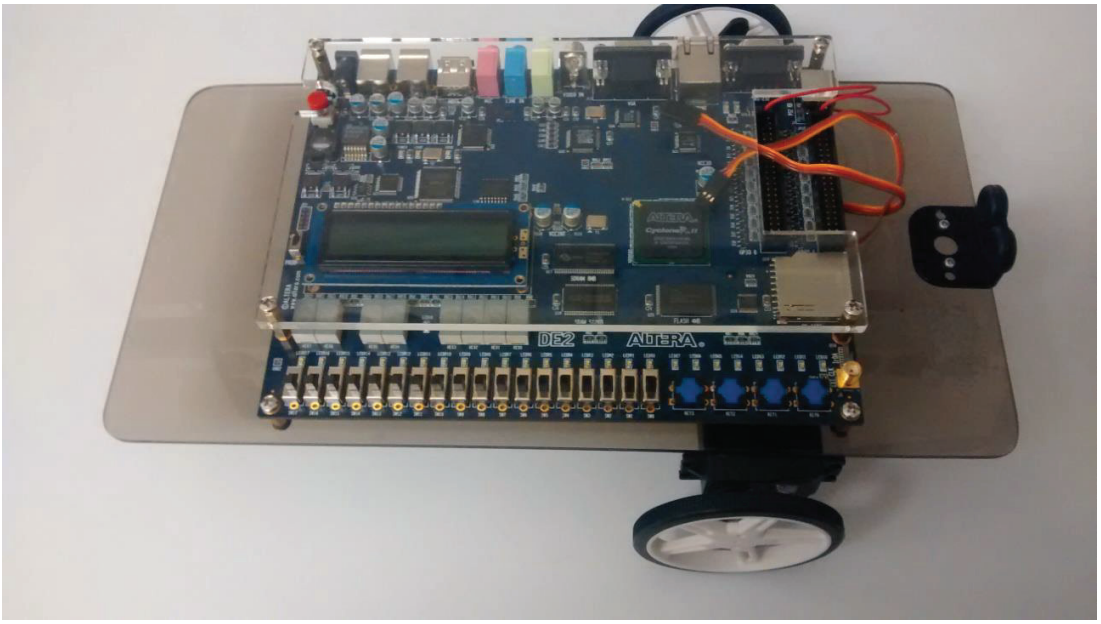


Figura 4.11 Robot Móvil

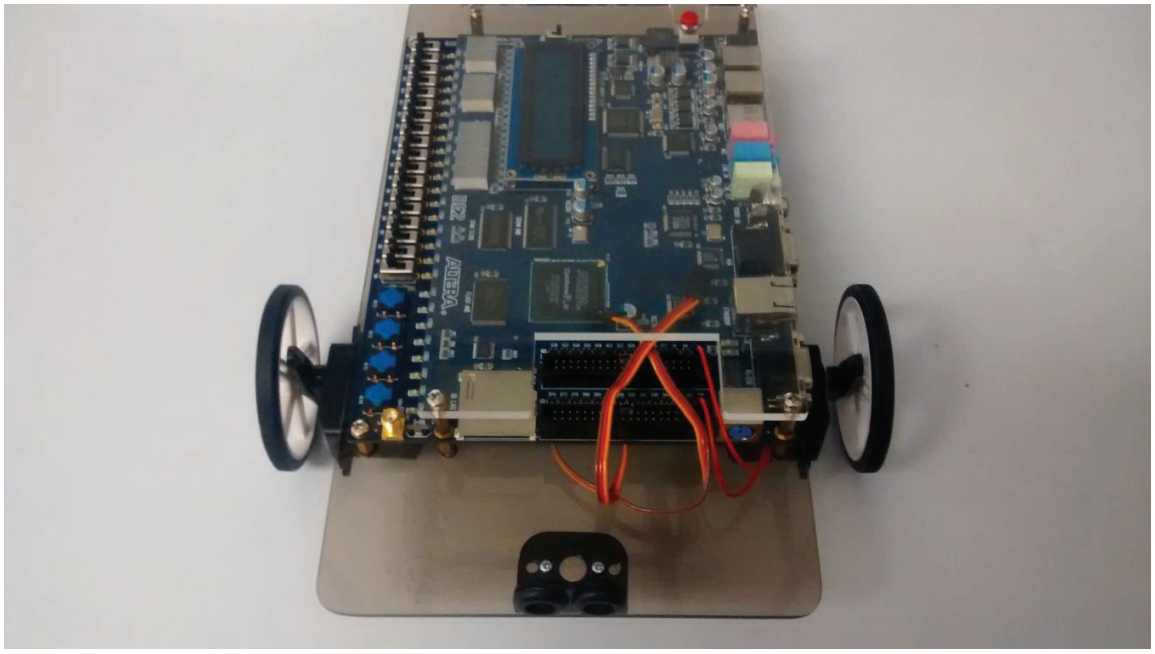


Figura 4.12 Robot Móvil

En la figura se muestra el montaje final, con servomotores y tarjeta fija. Las pruebas se realizaron con fuentes de alimentación de laboratorio solamente ya que la tarjeta maneja totalmente la dirección y control de los motores, el control se realiza por medio de los SWITCH utilizando 4 de ellos ya que indican los bits de control al acoplar los sensores.

4.5 Control de movimiento

Para la posición como se mencionó serán dos sensores lo cuales nos indicaran esos en el diagrama siguiente veremos cono serán colocados para el control del mismo.

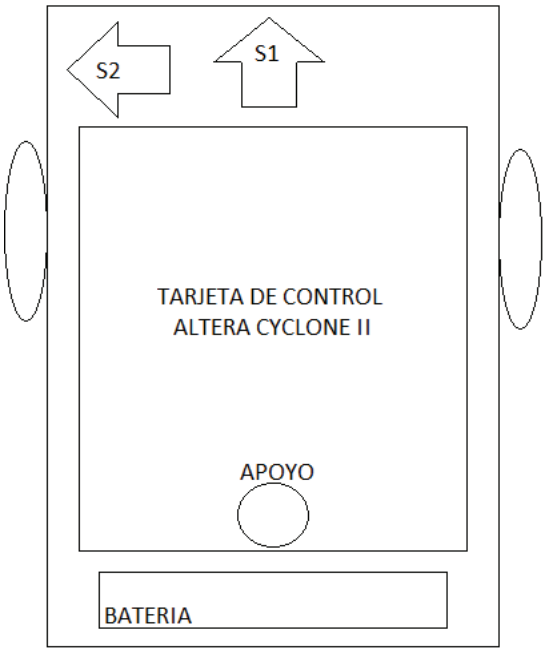


Figura 4.13 Robot en Movimiento Libre

Para el caso de la figura 4.13, el robot se encuentra en movimiento libre, los sensores colocados en la parte frontal cumplen dos funciones el S1 controla la distancia a medir para el avance y retroceso, por otra parte el sensor S2 funciona como un selector de giro midiendo la distancia se toma la decisión entre izquierda y derecha.

En la figura 4.14 se muestra un ejemplo de lo mencionado anteriormente. Al avanzar se encuentra con obstáculo frontal y el sensor S2 se encuentra bloqueado por otro objeto.

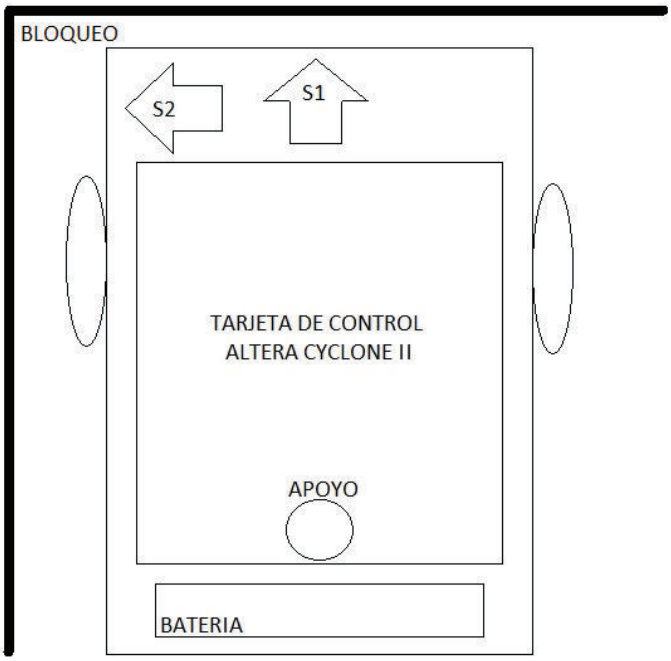


Figura 4.14 Robot con bloqueo frontal y lateral

Se realiza una medición de la distancia y se selecciona la opción más óptima. En este ejemplo vemos que el robot ha quedado muy cercano al bloqueo por un costado, entonces un retroceso y un giro a la derecha no es posible ya que no se obtiene el rango de giro deseado, esto lleva a un bloqueo del cual no podrá salir.

Para esto se selecciona un retroceso y con un giro de rueda izquierda para poder apartarse del obstáculo y así salir con giro derecho sin perder espacio para una vuelta correcta.

4.6 Ubicación de obstáculos

Los sensores serán dos medidores de distancia mediante ultrasónico específicamente el SRF-10. La imagen del sensor se muestra en la figura 4.15.

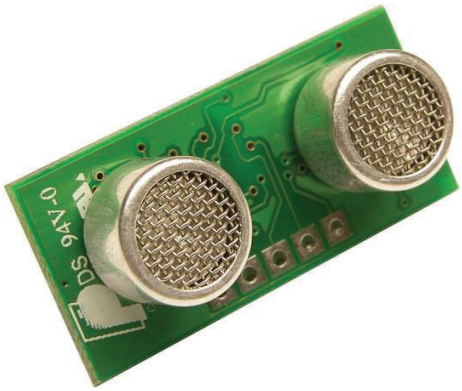


Figura 4.15 Sensor Ultrasónico

Este sensor controlado por un micro controlador es capaz de medir distancias hasta de 6m lo cual nos permite una mejor resolución debido a la sensibilidad para seleccionar la dirección a elegir.

Para controlar la distancia a medir se utilizara comunicación hexadecimal donde se asignará un valor de referencia que será el que nos indique qué tan lejos se están tanto de un objeto delantero que pueda obstruir el movimiento y que tan cerca de algo que esté en el costado izquierdo para obtener los mejores rangos de giro.

4.7 Interfaz entre sensores y FPGA

Para los sensores es necesaria una interconexión, el sensor propuesto es uno ultrasónico como se indicó anteriormente pero existen una gran variedad. Una de las ventajas de éste es que debido a su comunicación hexadecimal sólo es necesaria la implementación de un bloque de referencia para indicar los valores a medir por el FPGA.

Otro de los sensores que se pensaron utilizar para este proyecto fue un infrarrojo como el de la figura 4.16, sin embargo debido al comportamiento de dicho sensor analógico sería necesario un interfaz ADC (Analog Digital Convert) para poder realizar la comunicación.

Figura 4.16 Imagen de un sensor infrarrojo.

Para esta comunicación la adaptación que se utiliza es un integrado ADC el cual maneja un voltaje de referencia que depende del voltaje entregado por el sensor y la configuración que se le asigne. También es posible utilizar microcontroladores los cuales son más sencillos y fáciles de ajustar ya que el voltaje de referencia se puede configurar al mismo voltaje de trabajo del sensor infrarrojo.

Utilizando como referencia para el diseño de acoplador de señales el circuito integrado es el ADC0804LCN como el mostrado en la figura 4.17.

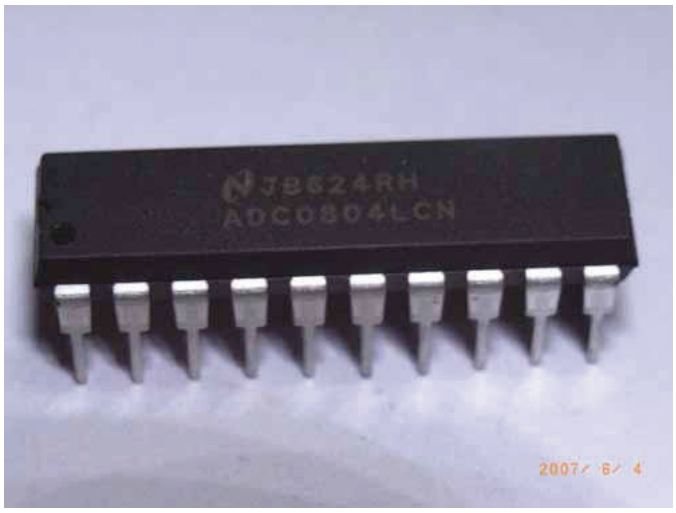


Figura 4.17 Integrado ADC

El circuito integrado ADC0804 es un convertidor de señal analógica a digital de 8 bits. Este circuito cuenta con un sólo canal de entrada analógica con una salida digital de ocho bits que puede mostrar 256 valores de medidas diferentes. El tamaño de medición se ajusta mediante el establecimiento de la tensión de referencia en el pin 9(ver figura 4.17). La

entrada de referencia de voltaje puede ser ajustado para permitir codificar cualquier rango de tensión analógica más pequeña para la totalidad de 8 bits de resolución.

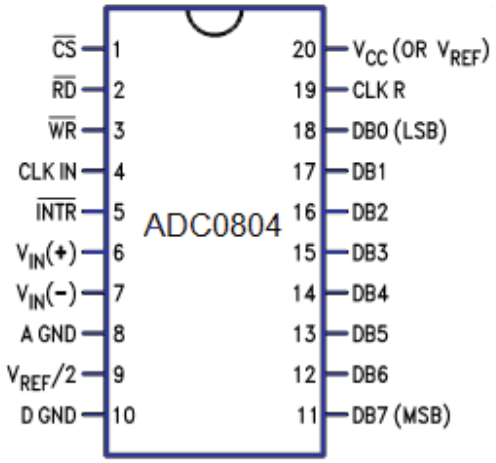


Figura 4.17 Terminales del circuito integrado ADC0804.

Capítulo 5

Conclusiones y trabajo futuro

Un aprendizaje importante con el desarrollo del prototipo RMR fue el potencial para facilitar el proceso de diseño, pruebas y correcciones que ofrecen las tarjetas del tipo FPGA, esta flexibilidad no es tan evidente en diseños basados en procesadores o microchip, que efectivamente son de costos menores, pero sus características de funcionamiento son más limitadas, un ejemplo muy visible son la capacidades de aplicación respecto a sus entradas y salidas que suelen ser pocas.

En este trabajo de tesis se desarrolló una plataforma robótica móvil diseñada para utilizar la tarjeta de desarrollo DE2. El FPGA-bot está diseñado para ser un pequeño vehículo autónomo que está programado para moverse en respuesta a una entrada sensorial. Una amplia variedad de sensores se puede conectar fácilmente a la FPGA-bot.

Durante el diseño, uno de los principales limitantes fueron los acoplamientos de las señales ya que debido a que los motores generan voltajes de retroceso, al accionarse estos pueden dañar la tarjeta, pero se soluciona con una simple alimentación externa ya que la tarjeta sólo suministra la señal de control.

Además a través de la experimentación se observa que también la aplicación de diferentes tipos de motores sería una mejoría, ya que se podrían adaptar diferentes capacidades desde un robot con mucha precisión controlando servomotores por modulación de ancho de pulso o PWM así como también motores de fuerza o velocidad que se podrían controlar cambian las entradas de señales de control a señales de activación de motores.

Con los motores utilizados se obtiene un control mucho más preciso y fino, respecto al giro y velocidad ya que se puede lograr una gran precisión de estos al momento de ejecutar algún movimiento así como también, si se aplica alguna retroalimentación de posición por pequeñas variaciones en el pulso, se pueden hacer movimientos milimétricos a diferencia de los motores DC que generan un impulso el cual puede ser algo mayor al deseado.

También en la aplicación de los sensores, la amplia gama de sensores en el mercado y la facilidad con la cual algunos ya traen ciertos acoplamientos como el caso del sensor ultrasónico prácticamente los convierte en dispositivos PLUG and PLAY debido a que ya vienen con una programación definida la cual se puede adaptar con modificar algunas líneas del código principal.

La simulación de la postura del robot considerando tres casos: ruedas girando en el mismo sentido, girando en sentido contrario y girando sola una, demuestran la validez del modelo utilizado para tal fin.

Como trabajo futuro se propone que el robot móvil pueda también ser controlado por un módulo de operación RF (X-bee) el cual es un dispositivo que permite comunicaciones mediante e programación la interpretación de señales con el propósito de demostrar la operación del robot móvil funcionando con señales de radio frecuencia controladas por computadora.

También se propone la implementación de sensores como una herramienta de control externa, la cual podría funcionar como ayuda para evitar bloqueos, así como también el control para diferentes tareas, información de retroalimentación hacia el equipo de cómputo sobre estado de FPGA-bot como la posición, orientación o tal vez alguna información de desempeño como distancia recorrida, así como algún dispositivo transmisor de imagen que permita el envío y recepción de imágenes en tiempo real.

Referencias

[1] Fernando Reyes, Control de robots manipuladores, Alfaomega, 2011, pp 509-539.

[2] G. Campion, G. Bastin& B. DAndrea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots".IEEE Trans. Robot.Autom.12(1), pp. 47-62, 1996.

[3] G. Campion, B. DAndrea-Novel &G. Bastin, "Modelling and state feedback control of nonholonomic mechanical systems". In proc. 30th IEEE Conf. on Decision and Control, UK, pp. 1184-1189, 1991.

[4] Fernando Reyes, MATLAB aplicado a robótica y Mecatrónica, Alfaomega, 2012.

[5] Eduardo BoemoScalvinoni, Estado del arte de la tecnología FPGA, cuaderno tecnológico no.1, microelectrónica, octubre de 2005

[6] J. O. Hamblen, T. S. Hall and M. D. Furman, Rapid Prototyping of Digital Systems, Springer, 2008, pp 242 -247

[7] R. de J. Romero Troncoso, Electrónica digital y lógica programable, Universidad de Guanajuato, 2007, pp.

[8]Fernández, L.M. A., Fernández, S. D. y Valmaseda, P. C. (2010). Planificación de trayectoria para un robot móvil. Universidad Complutense de Madrid. Tesis de licenciatura. Recuperado el 17 de agosto del 2014, de http://eprints.ucm.es/11301/1/MemoriaProyectoSSII.pdf.

[9]Libro. (2010). Estudio de robots móviles con ruedas. Universidad Politécnica Saleciana, Ecuador. Disponible en http://dspace.ups.edu.ec/bitstream/123456789/175/2/Capitulo%201.pdf.

[10]Revista Mental Actual, ServoMotores, Carlos Elias Sepulveda Lozano. Disponible en http://www.metalactual.com/revista/25/maquinaria_servo.pdf

[11]Universidad de OVIEDO, Tema VI: La Máquina de Corriente continúa Trabajo de Investigación. Disponible en http://www.uib.cat/depart/dfs/GTE/education/industrial/con_maq_electriques/teoria/Teoria%20Oviedo/Primer%20Parcial/Presentaciones%20en%20formato%20PDF/Tema6.pdf

Apéndices

Apéndice A. Control de Motor Derecho e Izquierdo

```
------Señales de control motor DERECHO
Libraryieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
------Identidad de motor_der
Entitymotor_deris
port(
		ent1, ent2 :in std_logic;
		sal:outstd_logic;
		clk_tar:instd_logic);
end motor_der;

-----Arquitectura behavior del motor
architecture behavior of motor_der is

SIGNAL clk:std_logic_vector(1 downto 0);
CONSTANT max:integer:=50000000;
CONSTANT half:integer:=max/2;
SIGNAL count0:INTEGER RANGE 0 TO max;
SIGNAL count1:INTEGER RANGE 0 TO max;
-------SELECCION DE DIRECCION (ent1, ent2)
begin
-------CONTROL DE FRECUENCIAS PWM
-------AVANZE
process
begin
		wait until clk_tar'event and clk_tar='1';
		if count0 < max then count0 <= count0+142;
		else count0<=0;
		end if;
		if count0 < half then clk(0) <= '0';
		else clk(0) <= '1';
		end if;
end process;
--------RETROCESO
process
begin
		wait until clk_tar'event and clk_tar='1';
		if count1 < max then count1 <= count1+200;
			else count1<=0;
		end if;
		if count1 < half then clk(1) <= '0';
		else clk(1) <= '1';
		end if;
endprocess;
--Control de Salida para Sentido
process
begin
		wait until clk_tar'event and clk_tar='1';
		if ent1='1' then sal<=clk(0);
			else sal<='0';
		end if;
		if ent2='1' then sal<=clk(1);
		end if;
end process;
end behavior;
```

ApéndiceB. Interfaz entre Sensores y Motores

```
--Interfaz entre Sensores y Motores
Libraryieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
--Identidad de Interfaz
entity interfaz is
port(
        sen:instd_logic_vector(3 downto 0);--EntradaSensores 4BITS
        clk:instd_logic;
        mda, mdr, mia, mir:outstd_logic);
        --mda=Motor Derecho Avance mdr=Motor Derecho Retroceso
        --mir=Motor Izquierdo Retroceso mia=Motor Izquierdo Avance
end interfaz;
--comenzando arquitectura de interfaz
architecture behavior of interfaz is
--Instruccion con herramientaelseif
process
begin
wait until clk'event and clk='1';
        if sen="0000" then --ALTO
                mda<='0';
                mdr<='0';
                mia<='0';
                mir<='0';
        elsifsen="0001" then --RETROCESO
                mda<='0';
                mdr<='1';
                mia<='0';
                mir<='1';
        elsifsen=("0010") then --AVANCE
                mda<='1';
                mdr<='0';
                mia<='1';
                mir<='0';
        elsifsen="0100" then --GIRO IZQUIERDO
                mda<='0';
                mdr<='1';
                mia<='1';
                mir<='0';
        elsifsen="1000" then --GIRO DERECHO
                mda<='1';
                mdr<='0';
                mia<='0';
                mir<='1';
        end if;
end process;
end behavior;
```

Apéndice C. Programación de Simulación en MATLAB

Programa Control

```
%Simulacion del control de postura de un uniciclo
%Robot movil tipo 2,0
clc;
clearall;
closeall;
animacion=1; %0:Deshabilita 1:Habilita
%----------------%
%Variables globales;
global r L W_R W_L k_2
%----------------%
%Parametros de simulacion
ts=10; %Tiempo de simulacion(segundo)
z_r0=[0;0;0];%Configuracion de referencia
z_0=[0.01;0.01;0.01];%Configuracionincial del robot
opciones=odeset('RelTol',1e-3,'InitialStep',2.5e-3,'MaxStep',2.5e-3);
%Parametros del robot:
r=0.03;%radio de las ruedas
L=0.1;%distancia entre las ruedas
%-------------------%
%Solucion del sistema:
W_R=3.0;
W_L=2.0;
k_2=10;%Ganacia del controlador
[T,X]=ode45('control_uniciclo',[0;ts],[z_r0;z_0],opciones);
n=length(T);%Numero de muestras
%-------------------%
%Formato para mostrar las figuras en pantalla
scrz=get(0,'ScreenSize');
off1=10;
off2=110;
figpos=[off1 off2 scrz(3)-off1 scrz(4)-off2];
%Figuras 1 y 2:Trayectoria y error de seguimiento de postura
f1=figure(1);
set(f1,'Color','w','Position',figpos);
g1=plot(X(:,1),X(:,2),'--k',X(:,4),X(:,5),'k');
set(g1,'LineWidth',2.0);
xlim([-0.05 0.3]);
ylim([-0.1 0.6]);
x1=xlabel('x(m)');
set(x1,'FontName','Times new roman','FontSize',12,'FontAngle','Italic','FontWeight','Bold');
y1=ylabel('y(m)');
set(y1,'FontName','Times new roman','FontSize',12,'FontAngle','Italic','FontWeight','Bold');
legend('Deseada','Actual');
grid;

f2=figure(2);
set(f2,'Color','w','Position',figpos);
```

```
subplot(3,1,1)
g1=plot(T,X(:,4)-X(:,1),'k');
set(g1,'LineWidth',2.0);
x1=xlabel('Tiempo(s)');
set(x1,'FontName','Times new roman','FontSize',12,'FontAngle','Italic','FontWeight','Bold');
y1=ylabel('x(m)');
set(y1,'FontName','Times new roman','FontSize',12,'FontAngle','Italic','FontWeight','Bold');
grid;
b1=text(-0.41,0.0063,'~');
set(b1,'FontWeight','Bold','Rotation',90);

subplot(3,1,2)
g2=plot(T,X(:,5)-X(:,2),'k');
set(g2,'LineWidth',2.0);
x2=xlabel('Tiempo(s)');
set(x2,'FontName','Times new roman','FontSize',12,'FontAngle','Italic','FontWeight','Bold');
y2=ylabel('x(m)');
set(y2,'FontName','Times new roman','FontSize',12,'FontAngle','Italic','FontWeight','Bold');
grid;
b2=text(-0.41,0.0069,'~');
set(b2,'FontWeight','Bold','Rotation',90);

subplot(3,1,3)
g3=plot(T,X(:,6)-X(:,3),'k');
set(g3,'LineWidth',2.0);
x3=xlabel('Tiempo(s)');
set(x3,'FontName','Times new roman','FontSize',12,'FontAngle','Italic','FontWeight','Bold');
y3=ylabel('x(m)');
set(y3,'FontName','Times new roman','FontSize',12,'FontAngle','Italic','FontWeight','Bold');
grid;
b3=text(-0.41,0.0046,'~');
set(b3,'FontWeight','Bold','Rotation',90);
%-----------------------------------------%
%Figura 3:Animacion
if(animacion)
timer=T(n);
fps=30;
   numero=timer*fps;
avance=round(n/numero);

   f3=figure(3);
   set(f3,'Color','w','Position',figpos);

for i=1:avance:n
clf
dibuja_uniciclo(X(i,4),X(i,5),X(i,6));
      axis equal
      axis([-0.05 0.35 -0.1 0.6 -0.01 0.01])
cameratoolbar('SetCoordSys','y')
      hold on
      view([-35.0,30.0])
      g1=plot(X(:,1),X(:,2),'--r');
```

```
      set(g1,'LineWidth',2.0);
      x1=xlabel('x(m)');
      set(x1,'FontName','Times','FontSize',10,'FontWeight','Bold');
y1=ylabel('y(m)');
set(y1,'FontName','Times','FontSize',10,'FontWeight','Bold');
z1=zlabel('x(m)');
      set(z1,'FontName','Times','FontSize',10,'FontWeight','Bold');
tit=sprintf('Robot MovilTipo(2,0)[t=%2.2fs]',T(i));
      t1=title(tit);
      set(t1,'FontName','Times','FontSize',12,'FontWeight','Bold');
      grid;
      pause(1/fps)
end
end
```

Programas de Parámetros

```
%Dibuja un cilindro
function []=cilindro(K,alto,radio,dif)
%el cilindro dibujado siempre en Z
%k es la matriz de cinematica con respecto a [0,0,0]
%alto y radio son propiedades del cilindro
%dif es la altura donde comienza el cilindro en (z=-dif)
[Xc,Yc,Zc]=cylinder(radio);
zc=(Zc*alto)-dif;
xc=Xc(1,:);
yc=Yc(1,:);
zc1=zc(1,:);
zc2=zc(2,:);
p1=[xc;yc;zc1;ones(1,length(zc1))];
p2=[xc;yc;zc2;ones(1,length(zc2))];
p1K=K*p1;
p2K=K*p2;
Xc=[p1K(1,:);p2K(1,:)];
Yc=[p1K(2,:);p2K(2,:)];
Zc=[p1K(3,:);p2K(3,:)];
surf(Xc,Yc,Zc,'FaceColor','blue','EdgeColor','none');
camlightleft;
```

```
%Control de seguimiento de postura de un uniciclo
%Robot movil tipo (2,0)
%Entradas: t tiempo
% u vector de estados
% u(:,1)=x_rposicion de referencia coordenada x
% u(:,2)=y_rposicion de referencia coordenada y
% u(:,3)=theta_rorientacion de referencia
% u(:,4)=x posicion coordenada x
% u(:,5)=y posicion coordenada y
% u(:,6)=theta orientacion de referencia
%Salida
% y=[zp_r;zp] derivada del vector de estados
% zp_r velocidad de refencia
```

```
% zp velocidad
function y=f(t,u)
%Variables globales
global r L W_R W_L k_2
%----------------------%
%Cinematica del robot de referencia
%Velocidad de entrada
v_r=(r/2)*(W_R+W_L);%Velocidad lineal
w_r=(r/L)*(W_R-W_L);%Velocidad angular
%Vector de postura
z_r=[u(1);u(2);u(3)];
theta_r=u(3);%Orientacion
%Matriz de transformacion G(z)
G_r=[cos(theta_r) 0;
   sin(theta_r) 0;
   0 1];
%Velocidad Robot
zp_r=G_r*[v_r;w_r];
%--------------------%
%Controlador no lineal
%Vector de postura del robot
z=[u(4);u(5);u(6)];
theta=u(6);%orientacion
%Error de seguimiento
R=[cos(theta) sin(theta) 0;
   -sin(theta) cos(theta) 0;
   0 0 1];
e=R*(z_r-z);
%acciones de control
u_1=-sin(v_r)*e(1);
u_2=-k_2*v_r*(e(2)/e(3))*sin(e(3))-sin(w_r)*e(3);
%--------------------%
%Cinematica del robo
%Velocidades de entrada
v=v_r*cos(e(3))-u_1;%velocidad lineal
w=w_r-u_2;%velocidad angular
%Matriz de transformacion G(z)
G=[cos(theta) 0;
   sin(theta) 0;
   0 1];
%Velocidad del robot
zp=G*[v;w];
%--------------------%
%Vector salida
y=[zp_r;zp];
```

```
%Dibujo un robot movil
%entrada :
% x posicion
% y posicion
% theta orientacion
function []=dibuja_uniciclo(x,y,theta)
%Longitud de las barras
l1=0.04;
```

```
l2=0.05;

%color y ancho de las barras
cr='k';ar=8;
%matrices de transformacion
A0=transl(x,y,0);
A1=rotz(theta)*transl(l1,0,0);
A2=rotz(theta+pi/2)*transl(l2,0,0);
A3=rotz(theta-pi/2)*transl(l2,0,0);
%calculo de cadenas de cinematicas
K0=A0;
K1=K0*A1;
K2=K0*A2;
K3= K0*A3;
%calculo de los vectores de los ejes
[po0,px0,py0,pz0]=ejes(K0,l1);
[po1,px1,py1,pz1]=ejes(K1,l1);
[po2,px2,py2,pz2]=ejes(K2,l1);
[po3,px3,py3,pz3]=ejes(K3,l1);
holdon;
%se dibujan las barras
eslabon(po0,po1,cr,ar);
eslabon(po0,po2,cr,ar);
eslabon(po0,po3,cr,ar);
%se dibujan las ruedas
radio=l1/2;alto=l1/2;
cilindro(K2*roty2(pi/2),alto,radio,alto/2);
cilindro(K3*roty2(pi/2),alto,radio,alto/2);  %*roty(pi/2)
```

```
%calculo de ejes
function[po,px,py,pz]=ejes(A,t)
po=A*([0 0 0 1]');
px=A*([t 0 0 1]');
py=A*([0 t 0 1]');
pz=A*([0 0 t 1]');
```

```
%dibuja un eslabon
function[]=eslabon(Po,Pf,c,lw)
po=Po';
pf=Pf;
plot3([po(1) pf(1)],[po(2) pf(2)],[po(3) pf(3)],c,'LineWidth',lw);
```

```
%matriz de transformacion para una rotacion sobre el eje y
function r=roty(t)
ct=cos(t);
st=sin(t);
r=[ct 0 st 0
0 1 0 0
   -st 0 ct 0
   0 0 0 2];
```

```
%archivo rotz.m
%matriz de transformacion para una rotacion sobre el eje z
function r=rotz(t)
ct=cos(t);
st=sin(t);
```

```
r=[ct -st 0 0
stct 0 0
   0 0 1 0
   0 0 0 1];
```

```
%matriz de transformacion para una translacion
function r=transl(x,y,z)
t=[x;y;z];
r=[eye(3) t;
0 0 0 1];
```