Robot Omnidireccional Controlado con NIOS II

Richard Salavarria Q. [(1)] Liliana Imbaquingo Q. [(2)] Ing. Ronald Ponguillo I. [(3)]
Facultad de Ingeniería en Electricidad y Computación
Escuela Superior Politécnica del Litoral (ESPOL)
Campus Gustavo Galindo, Km 30.5 vía Perimetral
Apartado 09-01-5863. Guayaquil-Ecuador
riedsala@espol.edu.ec [(1)], lilpaimb@espol.edu.ec [(2)], rponguil@espol.edu.ec [(3)]

Resumen

El presente proyecto da a conocer el proceso de diseño y construcción de un robot omnidireccional basado en el procesador Nios II. Debido a que el robot posee la característica de ser móvil elegimos la tarjeta de desarrollo DE0 Nano que por su tamaño y peso ofrece la ventaja de ser portable; esta tarjeta de desarrollo consiste en un sistema embebido basado en una FPGA Cyclone IV de Altera.

La primera etapa del proyecto consiste en la construcción de la estructura física del robot omnidireccional la cual contiene tres ruedas y consiste en una estructura circular de dos niveles, el robot tiene la opción de realizar dos tareas: una rutina de movimientos omnidireccionales y la detección y evasión de obstáculos, esta última tarea la realiza gracias a los sensores de ultrasonido que posee en la periferia delantera del nivel superior.

La segunda etapa del proyecto se encarga del diseño e implementación del controlador donde el hardware del controlador se lo implementa con la herramienta SOPC Builder y el software se lo describe usando el entorno de desarrollo NIOS II IDE.

Palabras Claves: *Robot omnidireccional, FPGA, Nios II, SOPC Builder*

Abstract

This project teaches the process of designing and building an omnidirectional robot based on the Nios II processor. Because the robot has the characteristic of being mobile we choose DE0 Nano development board for its size and weight offers the advantage of being portable, this development board is an embedded system based on an FPGA Cyclone IV from Altera.

The first stage of the project involves the construction of the physical structure of the omnidirectional robot which has three wheels and is a two-story circular structure, the robot has the option to perform two tasks: a routine of omnidirectional movement and the obstacle detection and avoidance, the latter task is performed by ultrasound sensors which are located in the periphery upper level front.

The second stage of the project involves the design and implementation of the controller where the controller hardware is implemented with SOPC Builder tool and the software is described using the development environment NIOS II IDE.

Keywords*: Omnidirectional robot, FPGA, Nios II SOPC Builder*

1. Introducción

Dentro del campo de la electrónica digital, los dispositivos lógicos programables han tenido una gran acogida debido a que permiten optimizar recursos físicos y lógicos al momento de diseñar e implementar la solución a un circuito, de igual manera estos dispositivos han venido evolucionando con el tiempo dando paso a tecnologías con mayor capacidad y velocidad de procesamiento.

En este grupo de dispositivos se encuentra la familia de las FPGA las cuales brindan al diseñador

una amplia variedad de características importantes, que van desde implementar un circuito integrado particular hasta un sistema capaz de procesar imágenes, audio y video.

En la robótica es muy necesario que exista una unidad de control principal la cual se encarga de procesar y comandar todos los dispositivos secundarios del sistema, las FPGA están adquiriendo cada vez mayor utilidad al momento de implementar la unidad de control principal puesto que además de lo descrito anteriormente, son reconfigurables lo que facilita el diseño y las pruebas antes de obtener el producto final.

2. Objetivos

2.1. Objetivo general

Desarrollar una plataforma Hardware/Software basado en el procesador Nios II, que controle los movimientos de un robot omnidireccional y le permita evadir obstáculos.

2.1. Objetivos específicos

- Crear el sistema de hardware necesario para controlar el robot usando SOPC Builder.
- Utilizar la modulación PWM para controlar la velocidad de los motores del robot.
- Diseñar funciones que controlen los sensores e interpreten sus señales para la detección de objetos.
- Emplear una rutina de movimientos que demuestre que el robot puede moverse en forma omnidireccional.

3. Marco teórico

3.1. Robot Omnidireccional

Un robot omnidireccional es un robot móvil que posee máxima maniobrabilidad en el plano; es decir, puede moverse en cualquier dirección sin necesidad de rotar o reorientarse, a diferencia de otros tipos de robots móviles que requieren girar y cambiar de dirección para llegar a un punto deseado.

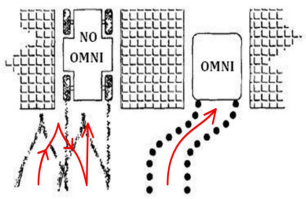


Figura 1. Movimientos de un móvil no omnidireccional y uno omnidireccional [1]

El sistema de locomoción es una de las principales propiedades de un robot móvil, para el caso de un robot omnidireccional este sistema está basado en ruedas y se divide en dos importantes grupos: ruedas orientables y ruedas especiales.

(a) (b)

Figura 2. Ruedas Omnidireccionales (a) Universal Doble. (b) Mecanum.

Además del sistema de locomoción una decisión importante en el diseño y construcción de un robot omnidireccional es la cantidad de ruedas a utilizar, las opciones más comunes son 3 o 4 ruedas donde los vehículos de 3 ruedas poseen control y dirección simple pero estabilidad limitada mientras que los de 4 ruedas presentan mecánica y control complejos pero mayor estabilidad y tracción.

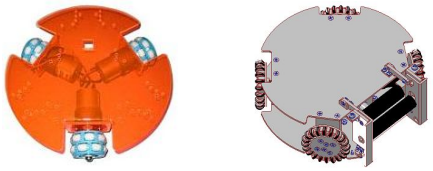


Figura 3. Estructuras de un robot omnidireccional (a) tres ruedas (b) cuatro ruedas

3.2. Sensores de proximidad

Son utilizados para medir las distancias a las cuales están ubicados los objetos, dentro de éstos se encuentran los sensores ultrasónicos que trabajan a frecuencias entre 38 y 50Khz; en estos tipos de sensores el emisor lanza un tren de pulsos y el receptor espera el rebote de la señal; el tiempo que demora en llegar la onda reflejada se lo relaciona con la velocidad del sonido para de esta manera determinar la distancia a la cual se encuentra el objeto que hizo rebotar dicha onda.

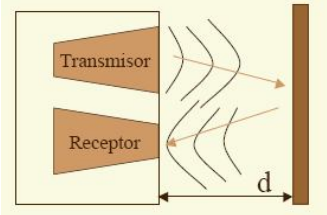


Figura 4. Funcionamiento de un sensor de ultrasonido

3.3. Modulación PWM

La modulación PWM consiste en generar una señal periódica, por lo general un tren de pulsos, a la cual podemos modificar el ciclo de trabajo, ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

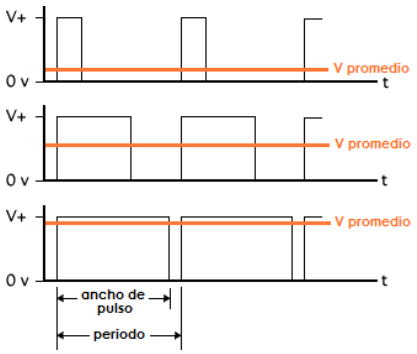


Figura 5. Señal PWM con diferentes ciclos de trabajo

3.4. Tecnología utilizada

3.4.1. Lenguaje de Descripción de Hardware.

Un HDL es un lenguaje de programación de alto nivel usado para describir la entidad y arquitectura de un sistema electrónico, el objetivo de un HDL es realizar la descripción de un circuito mediante un conjunto de instrucciones de alto nivel; para que de esta manera el programa de síntesis genere o ensamble el circuito que será utilizado físicamente.

3.4.2. Sistema embebido configurable.

En general un sistema embebido consiste en un microcontrolador cuyo hardware y software están diseñados y optimizados para resolver un problema concreto de forma eficiente [2].

Un FPGA es un circuito integrado que contiene bloques de lógica programable, es decir, su interconexión y funcionalidad (entidad y arquitectura) pueden ser configuradas mediante un HDL. Los FPGAs tienen la gran ventaja de ser reprogramables lo que otorga mucha flexibilidad en el diseño.

Al usar sistemas embebidos en FPGAs estamos refiriéndonos a que podemos diseñar a nuestra conveniencia un computador que como tal incluya procesador, memoria y circuitos de entrada y salida, y que además podemos modificarlo usando algún HDL.

3.4.3. Tarjeta de Desarrollo Altera DE0 Nano.

La tarjeta de desarrollo DE0 Nano posee una FPGA Cyclone IV EP4CE22F17C6N y tiene como fin ser empleada en proyectos educativos aunque debido a su versatilidad también se presta para aplicaciones en la industria. Su principal ventaja frente a otras tarjetas de desarrollo de su misma marca es su peso y tamaño, lo cual le permite ser utilizada en proyectos portables o móviles.

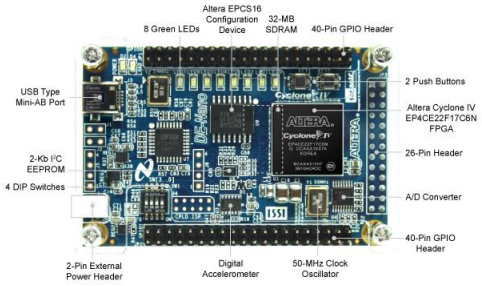


Figura 6. Tarjeta DE0 Nano

3.4.4. Procesador NIOS II.

La tarjeta de desarrollo DE0 Nano posee una FPGA Cyclone IV EP4CE22F17C6N y tiene como fin ser empleada en proyectos educativos aunque debido a su versatilidad también se presta para aplicaciones en la industria. Su principal ventaja frente a otras tarjetas de desarrollo de su misma marca es su peso y tamaño, lo cual le permite ser utilizada en proyectos portables o móviles.

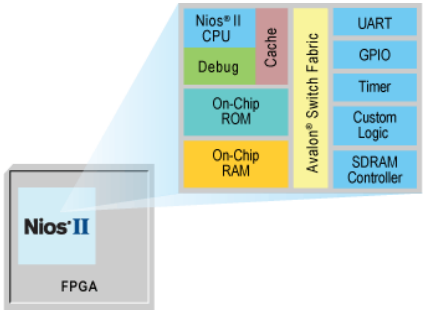


Figura 7. Sistema basado en el procesador Nios II

Nios II es un procesador de propósito general de tipo soft-core diseñado para varios FPGA de Altera. Este procesador posee una arquitectura tipo Harvard, debido a que usa buses separados para instrucciones y para datos.

Además posee tres versiones disponibles, dependiendo de cómo se quiera optimizar los recursos de la FPGA o maximizar el rendimiento del procesador:

El NIOS II/f (“rápido”) es la versión diseñada para alto rendimiento, y que con un “pipeline” de 6 etapas proporciona opciones para aumentar su desempeño, como memorias caché de instrucciones y datos, o una unidad de manejo de memoria (MMU, Memory Management Unit).

El NIOS II/s (“estándar”) es la versión con “pipeline” de 5 etapas que dotada de una unidad aritmética lógica (ALU, Arithmetic Logic Unit) busca combinar rendimiento y consumo de recursos.

El NIOS II/e (“económico”) es la versión que requiere menos recursos de la FPGA, sin “pipeline” y muy limitada, dado que carece de las operaciones de multiplicación y división. [3]

3.4.5. Quartus II – SOPC Builder. Quartus II es una herramienta de software producido por Altera para el análisis y la síntesis de los diseños en HDL. [4]

SOPC Builder también es una herramienta producida por Altera a la cual podemos acceder desde Quartus II, esta herramienta realiza la conexión interna de componentes de hardware para crear un sistema de hardware completo que se ejecuta en cualquiera de sus diversas FPGA.

3.4.6 NIOS II IDE. Es un entorno de desarrollo para lenguajes de alto nivel. El código descrito en este entorno será ejecutado en los sistemas de hardware que fueron generados previamente por el SOPC Builder y grabado en la FPGA. Al usar este IDE se puede integrar el software con el hardware de los Sistemas Embebidos.

4. Construcción del robot omnidireccional

La estructura del robot tiene forma circular y se divide en dos niveles; en el nivel inferior se encuentran: el sistema de energía y los motores acoplados directamente con las ruedas, mientras que en el nivel superior están ubicados: 5 sensores ultrasónicos, la tarjeta DE0 Nano y un PCB que sirve para conectar todos los elementos del robot con la FPGA.

Para la construcción del robot omnidireccional se utilizaron tres ruedas omnidireccionales de tipo universal doble, marca Nexus Robot las cuales se encuentran ubicadas a 120° entre sí.

Los motores usados son de marca Dagu, los mismos que incluyen una caja reductora la cual posee un eje de giro desplazado a 90° con respecto al eje del motor.

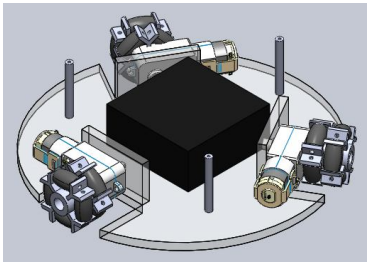


Figura 8. Nivel inferior de la estructura completo

Se utilizó el integrado L293D para la parte de control y fuerza de los motores, el cual proporciona una salida de hasta 30 V y 2 A. Para controlar cada motor el L293D recibe dos señales de control provenientes de la FPGA y envía al motor las mismas señales pero con un nivel de voltaje diferente.

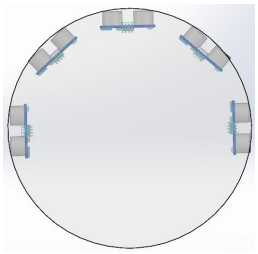


Figura 9. Disposición de los sensores en la estructura del robot

Se utilizan un total de 5 sensores: 4 del tipo HC-SR04 y 1 del tipo HY-SR05, los cuales se ubicaron en la periferia frontal del nivel superior del marco del robot, como lo indica la Figura 3.3, para de esta manera asegurar la correcta detección de objetos a la distancia requerida.

Para el sistema de energía se utilizaron 8 pilas recargables de Ni-MH de 1.2 V y 1600 mAh cada una, en total 9.6 V, este voltaje se regula a 5 V para alimentar a la tarjeta DE0 Nano, sensores y drivers.

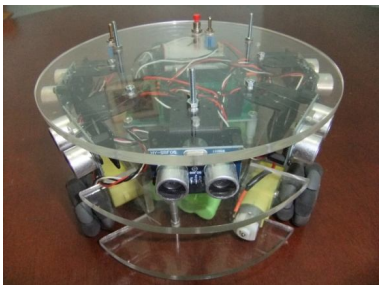


Figura 10. Robot Omnidireccional

5. Diseño e implementación del controlador

5.1. Diseño

5.1.1. Arquitectura de hardware

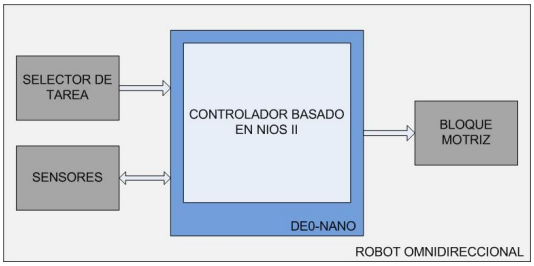


Figura 11. Integración componentes con el controlador

Selector de Tarea: Está implementado con un interruptor simple y se utiliza para elegir entre la rutina de movimientos y la detección de objetos.

Sensores: Los sensores de ultrasonido son usados cuando se selecciona la tarea de detección y evasión de obstáculos, cada sensor se comunica con el controlador mediante dos de sus pines, TRIGGER y ECHO, usados para inicializar al sensor y medir la distancia del objeto respectivamente.

Controlador basado en NIOS II: Representado por la tarjeta DE0 Nano, la cual posee dos puertos de expansión para propósito general a través de los cuales se conectan los bloques externos directamente al FPGA donde se implementa el sistema embebido que controla todas las funciones del robot.

Bloque Motriz: En este bloque se encuentran los drivers y motores. Los drivers reciben una señal PWM del FPGA denominada señal de control y mediante su circuitería interna entregan a los motores la misma señal PWM (señal de fuerza) pero con una mayor cantidad de corriente.

5.1.2. Arquitectura de Software

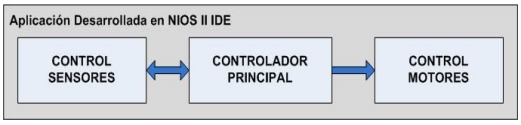


Figura 12. Diagrama de bloques de la arquitectura del software

Controlador Principal: Es el encargado de tomar todas las decisiones dentro de la aplicación.

Control Sensores: Recibe del controlador principal la señal para habilitar los sensores, y le envía la información correspondiente a las distancias leídas por cada uno de los sensores.

Control Motor: Genera las señales PWM enviadas a los drivers (posteriormente a los motores) y con esto controla los movimientos del robot.

5.2. Implementación

5.2.1. Implementación del hardware. El hardware del controlador fue creado usando la herramienta SOPC Builder en la cual se seleccionaron los siguientes CORES o componentes:

Nios II CPU: Está implementado con un procesador Nios II/f (rápido).

JTAG UART: Sirve para comunicar de forma serial a la computadora con el sistema creado con el SOPC Builder y para la ejecución y depuración de las aplicaciones.

Módulos PWM: Son 6 módulos que generan señales PWM cada uno, las mismas que se aplican a los 3 motores del robot omnidireccional.

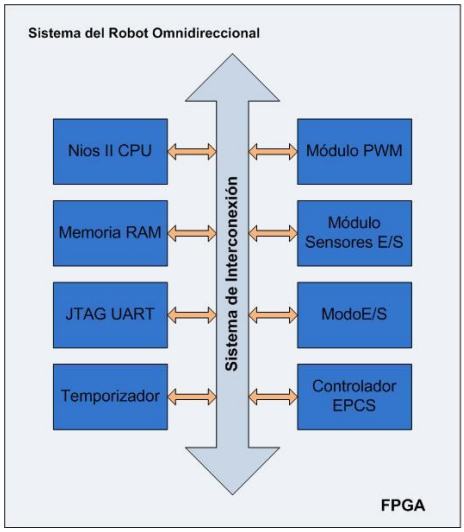


Figura 13. Componentes del hardware del sistema

Módulos Sensores: Dado a que se usan 5 sensores de ultrasonido para la detección de obstáculos, se agregó un módulo conformado por 10 PIO de un bit cada uno, 5 de estos configurados como salida para habilitar a los sensores y los 5 restantes configurados como entrada para recibir el pulso de los sensores que indica la distancia del obstáculo.

Modo: Es un PIO configurado como entrada con el fin de verificar si el usuario desea ejecutar la rutina programada o la detección y evasión de obstáculos.

Memoria RAM: Es una memoria creada dentro de la FPGA que tiene como función almacenar las instrucciones de la aplicación de forma temporal para poder ejecutarlas en el procesador.

Controlador EPCS: Este componente tiene como objeto controlar la memoria de tipo flash EPCS16 ubicada en la PCB de la DE0 Nano y además es usado por el Flash Programmer para almacenar la información del hardware y del software de forma permanente en la EPCS.

Temporizador: Es un timer embebido, utilizado para medir el tiempo en el que la señal del pin ECHO de los sensores se encuentra en alto, en este caso el timer es un contador de ciclos de trabajo y luego el valor de conteo es relacionado con la frecuencia del reloj global para obtener el tiempo en segundos.

El Sistema de interconexión que se indica en la Figura 13 corresponde a un bus Avalon el mismo que transparenta la comunicación entre los componentes y el CPU.

5.2.2. Implementación del Software. La implementación del software del controlador se la realizó en la herramienta NIOS II IDE. Para controlar los motores y así los movimientos del robot se usa la librería ***altera_avalon_pwm_routines.h.***

Para realizar la detección de obstáculos con los sensores se implementaron las funciones: Read_distance y Distance

Función Read_distance

```
float read_distance(unsigned int trigger_base,
                    unsigned int echo_base){
    int counter_us=0;
    float distance;
    long numclow, numchigh, numclks;

1   IOWR(TIMER_BASE,2,0Xffff);
    IOWR(TIMER_BASE,3,0Xffff);
2   IOWR(trigger_base,0,0);
    IOWR(trigger_base,0,1);
    usleep(20);
    IOWR(trigger_base,0,0);
3   while(!IORD(echo_base,0));
4   IOWR(TIMER_BASE,1,4);
5   while(IORD(echo_base,0));
6   IOWR(TIMER_BASE,1,8);
7   IOWR(TIMER_BASE,4,0);
    IOWR(TIMER_BASE,5,0);
8   numclow=0xffff & IORD(TIMER_BASE,4);
    numchigh=0xffff & IORD(TIMER_BASE,5);
    numclks=0xffffffff &(numclow +(numchigh<<16));
9   counter_us=((0xffffffff-numclks)/50);
10  usleep(30000);
11  distance=counter_us/58.3;
    return distance;
}
```

Figura 14. Código de la función Read_distance

Esta función retorna la distancia en cm a la cual se encuentra un objeto, para lo cual:

Se inicializa el Timer con un valor de 0XFFFFFFFF.

Se genera un pulso de 20 µs en el pin trigger del sensor para activarlo.

3. Se espera mientras la señal del pin echo esté en bajo.

4. Se activa el Timer para contar el tiempo que la señal echo se mantiene en alto.

5. Se espera mientras la señal echo esté en alto

6. Una vez que la señal del pin echo regrese al nivel bajo se detiene el Timer.

7. Se guarda el valor del timer en los registros snapshotL y snapshotH.

8. Se obtiene el valor del timer al leer los dos registros y se lo ensambla en una variable de 32 bits.

9. Se obtiene el tiempo en microsegundos.

10. Se espera 30 ms, necesarios para que el sensor se estabilice.

11. Se determina la distancia en centímetros a la cual se encuentra el objeto.

Función Distance

1. Se lee la distancia de los cinco sensores al objeto más próximo.

2. A la variable dis se le agrega 1 si el objeto se encuentra a una distancia mayor que la distancia enviada como parámetro, 0 en caso contrario. Se desplaza un bit a la izquierda para añadir el dato de cada sensor.

```
int distance(float d1,float d2,float d3,float d4,float d5){

    int dis=0;
1   float distancia1=read_distance(TRIGGER_1_BASE,ECHO_1_BASE);
    float distancia2=read_distance(TRIGGER_2_BASE,ECHO_2_BASE);
    float distancia3=read_distance(TRIGGER_3_BASE,ECHO_3_BASE);
    float distancia4=read_distance(TRIGGER_4_BASE,ECHO_4_BASE);
    float distancia5=read_distance(TRIGGER_5_BASE,ECHO_5_BASE);

    dis=distancia2>d2;
2   dis=(dis<<1)+(distancia5>d5);
    dis=(dis<<1)+(distancia4>d4);
    dis=(dis<<1)+(distancia3>d3);
    dis=(dis<<1)+(distancia1>d1);
    return dis;
}
```

Figura 15. Código de la función Distance

Se agregan los bits en el siguiente orden: sensor 2, sensor 5, sensor 4, sensor 3 y sensor 1; donde la numeración de los sensores se muestra en la Figura 16.

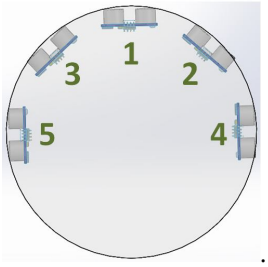


Figura 16. Numeración de los sensores en la estructura

6. Pruebas y Resultados

6.1. Prueba: Movimiento omnidireccional

Descripción

El robot ejecuta la rutina programada en el Nios II IDE para comprobar que puede moverse de forma omnidireccional.

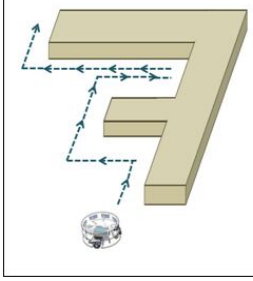
Resultado

Se observa que primero realiza un cuadrado comprobando los movimientos adelante, izquierda, atrás y derecha; luego realiza los movimientos diagonales formando un rombo y finalmente gira sobre su propio eje.

Figura 17. Rutina del robot omnidireccional (a) Trayectoria en línea recta (b) Trayectoria diagonales (c) Giro sobre su propio eje

6.2. Prueba: Detección y evasión de obstáculos

Tabla 1. Resultados de la prueba: Detección y evasión de obstáculos

Resultado esperado	Resultado obtenido
	

Descripción

Se provee de un camino para probar que el robot puede evadir obstáculos indistintamente del camino en el que se encuentra.

Resultado

El robot cumple con lo esperado y no se presentan colisiones en su recorrido.

7. Conclusiones

- Trabajar con sensores y motores demuestra las diversas aplicaciones que se les pueda dar a un sistema embebido basado en NIOS II.
- El movimiento omnidireccional del robot está estrechamente relacionado con el diseño mecánico.
- El uso de la memoria EPCS para almacenar el hardware y software permite al sistema no depender de un computador cuando se enciende la FPGA.
- NIOS II IDE permite obtener un nivel de abstracción al poder desarrollar fácilmente una aplicación sin necesidad de conocer a detalle el hardware.

8. Referencias

[1] Mark West y Haruhiko Asada, In Design of a Holonomic Omnidirectional Vehicle, International conference on robotics and automation, volume 3, páginas 97-103, Mayo 1992

[2] Introducción a los Sistemas Embebidos, http://es.scribd.com/doc/111023290/Introduccion-a-Sistemas-Embebidos

[3] Jorge Rodríguez Araujo, Estudio del microprocessor Nios II, Marzo 2010

[4] Quartus II, http://es.wikipedia.org/wiki/Quartus_II