ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"ROBOT OMNIDIRECCIONAL CONTROLADO CON NIOS II"

TESINA DE SEMINARIO

Previa la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

LILIANA PATRICIA IMBAQUINGO QUY YON

RICHARD EDUARDO SALAVARRIA QUIROZ

Guayaquil – Ecuador

Año 2013

AGRADECIMIENTO

A Dios quien no se deja ganar en generosidad.

A mi familia en especial a mi mamá por luchar para darnos una buena educación, la mejor herencia que un padre puede dejar a sus hijos.

A mis abuelitos y tíos quienes siempre me han brindado su ayuda y apoyo incondicional.

A mi esposo por haberme acompañado y aconsejado en esta etapa universitaria y en esta nueva etapa que hemos empezado juntos.

Liliana Imbaquingo Quy Yon

A Dios por colmar mi vida de bendiciones.

A mi familia quienes han estado conmigo compartiendo mis éxitos, pero más aún han sido mi más grande apoyo en los momentos difíciles.

A mi amada esposa quien me demuestra cada día que para ser el mejor en todo lo más importante es siempre servir a los demás.

Y a mis amigos, compañeros y profesores por compartir su tiempo y anécdotas, y hacerme aprender algo nuevo cada día.

Richard Salavarria Quiroz

DEDICATORIA

A mi familia que es el centro de mi vida y por quienes he de esforzarme para salir adelante.

Liliana Imbaquingo Quy Yon

A mis padres, a mi esposa, a mi hija y a las personas apasionadas por la tecnología.

Richard Salavarria Quiroz

TRIBUNAL DE SUSTENTACIÓN

ING. RONALD PONGUILLO INTRIAGO

PROFESOR DE SEMINARIO DE GRADUACIÓN

PHD. DANIEL OCHOA DONOSO

PROFESOR DELEGADO POR LA UNIDAD ACADÉMICA

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesina, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral".

(Reglamento de Graduación de la ESPOL)

_____

Liliana Patricia Imbaquingo Quy Yon

_____

Richard Eduardo Salavarria Quiroz

RESUMEN

El presente documento da a conocer las diversas etapas de la construcción de un robot omnidireccional, desde la elección de las piezas hasta el ensamblaje de las mismas, así como el diseño e implementación del hardware y software del controlador, usando la FPGA incluida dentro de la tarjeta de desarrollo DE0 Nano.

El documento ha sido organizado en los siguientes capítulos:

En el capítulo 1 se habla sobre las generalidades del proyecto como son objetivos, alcances, limitaciones y justificación.

En el capítulo 2 se define técnicamente al robot omnidireccional, se hace una revisión de ruedas y tipos de estructuras y se estudia la modulación PWM y los sensores de distancia. Además se incluyen los temas más importantes relacionados con un sistema embebido basado en el procesador Nios II.

En el capítulo 3 se describen las componentes que se van a utilizar para construir el robot omnidireccional como son: la estructura principal, tipo de ruedas, motores, drivers, sensores, además de la conexión que existe entre los

componentes del robot y la FPGA. También se ilustra el proceso de montaje de las piezas usando la herramienta gráfica Solidworks.

En el capítulo 4 se detalla el proceso de diseño e implementación del hardware y software del controlador. Se explica sobre los módulos que fueron añadidos al controlador por medio de SOPC Builder y acerca de las funciones más importantes que se usaron e implementaron para controlar los movimientos del robot omnidireccional y la detección de objetos.

En el capítulo 5 se muestra la descripción de las pruebas realizadas con el robot omnidireccional, así como sus respectivos resultados mediante tablas que permiten observar y comparar el comportamiento del robot ante diferentes situaciones.

ÍNDICE GENERAL

BIBLIOGRAFÍA

ANEXOS

ABREVIATURAS

BDF	Block Diagram File
BSP	Board Support Package
DE0 NANO	Development and Education Board
DIP	Dual In-line Package
EEPROM	Electrically Erasable Programmable Read-Only Memory
ELF	Executable and Linking Format File
EPCS	Erasable Programmable Configuration Serial
FPGA	Field Programmable Gate Array
IDE	Integrated Development Environment
HAL	Hardware Abstraction Layer
HDL	Hardware Description Language
IDC	Insulation Displacement Connector
IP Core	Intellectual Property Core

MB	Mega Byte
PCB	Printed Circuit Board
PWM	Pulse Width Modulation
RISC	Reduced Instruction Set Computer
RTL	Resistor Transistor Logic
SDRAM	Synchronous Dynamic Random-Access Memory
SOC	System On Chip
SOF	SRAM Object File
SOPC	System On Programmable Chip
VHDL	Very High Speed Hardware Description Language

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

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 más 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.

CAPITULO 1

1 GENERALIDADES

1.1 Objetivos

1.1.1 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.

1.1.2 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.

1.2 Alcances y Limitaciones

El alcance de este proyecto se centra en el diseño y la construcción de un robot omnidireccional capaz de detectar y evadir obstáculos y que además demuestre que puede moverse omnidireccionalmente. Las limitaciones que se establecen dentro del alcance planteado son las siguientes.

- Los obstáculos, para ser detectados, deberán estar ubicados dentro del campo de búsqueda de los sensores o en su línea de vista.

- El robot se desplazará siempre hacia adelante hasta cuando detecte algún obstáculo que lo obligue a cambiar de dirección.

- Sólo en el modo de demostración de movimientos omnidireccionales puede moverse hacia atrás y diagonalmente.

- Todos los movimientos del robot se harán a una velocidad constante.

1.3 Justificación

Hoy en día se hace imprescindible monitorear distintos lugares que en ocasiones resultan inaccesibles e incluso peligrosos para el ser humano, como zonas de guerra, desastre químicos, etc.; pero gracias a la tecnología se han podido construir máquinas que automaticen estas tareas con el fin de eliminar el peligro que conlleva realizarlas, dentro de estas máquinas tenemos a aquellas que son móviles e independientes, llamados robots autónomos. Uno de los principales requerimientos para estos robots, es la capacidad de moverse en cualquier dirección indistintamente del lugar en el que se encuentran, los cambios bruscos en dirección pueden ser un poco complejos pero esto se ve eliminado al usar un esquema de movimiento omnidireccional. Además del movimiento, otra de las tareas de un robot autónomo es la de obtener información del ambiente que le rodea, por lo que el uso de sensores es necesario.

Por esto se ha optado diseñar y construir un robot omnidireccional, es decir, que pueda cambiar la dirección de su trayectoria sin necesidad de girar; que use sensores de distancia los cuales permitirán detectar y evadir obstáculos y que use un sistema embebido. Si bien es cierto ya se han implementado diversos modelos de robots omnidireccionales, el fin de elaborar uno a través de un sistema embebido sobre FPGA teniendo como procesador al Nios II, es poder ampliar el control y la lista de tareas del robot sin necesidad de migrar a otro dispositivo por falta de recursos ya que en este tipo de sistemas se pueden agregar módulos al controlador sin cambios de hardware en el PCB.

CAPITULO 2

2 MARCO TEÓRICO

2.1 Robot omnidireccional

Un robot omnidireccional es un robot móvil que posee máxima maniobrabilidad en el plano; en otras palabras, 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. En la figura 2.1 se observan los movimientos que un vehículo no omnidireccional y uno omnidireccional deben realizar para alcanzar un objetivo, el vehículo no omnidireccional debe realizar una trayectoria compleja mientras que el vehículo omnidireccional se dirige con mayor facilidad hacia el lugar deseado.

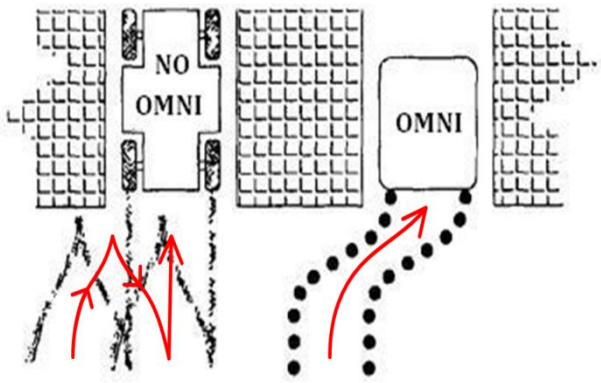


Figura 2.1 Movimientos de un móvil no omnidireccional y uno omnidireccional [1]

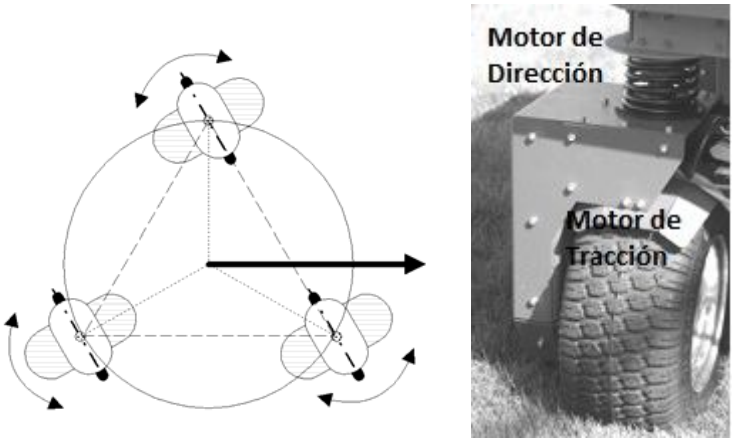
2.1.1 Sistema de locomoción

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: las ruedas orientables y ruedas especiales.

2.1.1.1 Ruedas orientables

Dentro de las ruedas orientables, aquellas que permiten un desplazamiento omnidireccional son las ruedas orientables centradas, ya que cada rueda

posee un motor que la orienta (dirección) y otro motor que le permite girar (tracción).



(a) (b)

Figura 2.2 Robot omnidireccional con ruedas orientables centradas. (a) Disposición sobre una estructura mecánica. (b) Componentes de tracción y dirección.

Todas las ruedas orientables poseen una componente activa, la cual está relacionada con la tracción de la rueda esto es si la rueda se mueve hacia adelante la dirección de la componente activa y la tracción serán hacia adelante.

En la figura 2.2(a) se presenta una versión de robot omnidireccional con tres ruedas orientables centradas,

esta disposición de ruedas permite realizar movimientos omnidireccionales pero genera cierta complejidad en el diseño y control debido al uso de dos motores (tracción y dirección) por cada rueda, como lo indica la figura 2.2 (b). Es decir, se debe ir cambiando de dirección mientras el robot se está moviendo.

2.1.1.2 Ruedas Especiales

Las ruedas especiales están basadas en la idea de incluir: una componente activa, que provee tracción en una dirección, y una componente pasiva, que así mismo provee tracción pero en una dirección diferente a la activa.

Figura 2.3 Componentes de una rueda Universal

Estas ruedas presentan entre otras ventajas: simplicidad en el control de movimiento y facilitan el diseño mecánico del robot puesto que, a diferencia de las orientables, se requiere un solo motor por rueda.

En este grupo tenemos a las ruedas Universales y a las ruedas Mecanum; ambas presentan rodillos ubicados en su periferia, pero dispuestos en ángulos diferentes, tal como se ilustra en la figura 2.4.

Fuente: http://www.robotshop.com

(a) **(b)**

Figura 2.4 Ruedas Omnidireccionales (a) Universal Doble. (b) Mecanum.

Los rodillos ubicados en la periferia de la rueda universal son los que añaden una componente pasiva perpendicular a la componente activa. Cuando tres o más de estas ruedas son utilizadas en un robot, sus componentes activas y pasivas combinadas permiten realizar movimientos omnidireccionales. En las ruedas Mecanum, la componente pasiva forma un ángulo de 45° con respecto a la activa.

A pesar de que las ruedas especiales facilitan el diseño mecánico del robot, éstas presentan ciertas limitaciones con respecto a las orientables centradas:

- Poseen carga limitada debido a que su apoyo está en los rodillos ejerciendo fuerza sobre sus ejes que son más frágiles que los ejes de la propia rueda.
- Poseen radio variable debido a los rodillos.
- Sensibilidad a desniveles y obstáculos.

Tipo de Rueda	Capacidad de carga	Diseño	Radio de la rueda	Fricción	Sensibilidad al suelo
Universal	Baja	Simple	Variable	Baja	Si
Mecanum	Baja	Complejo	Variable	Baja	Si
Orientables	Alta	Complejo	Constante	Alta	No

Tabla 2.1 Comparación de ruedas estudiadas [2]

2.1.2 Estructura de un robot omnidireccional

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 cada opción posee sus ventajas y desventajas. 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.

2.1.2.1 Robot omnidireccional de 3 ruedas

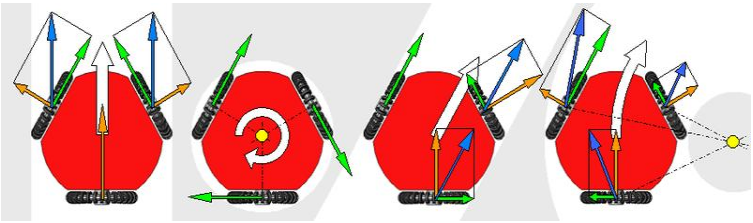
En este tipo de estructura las ruedas están dispuestas 120° entre sí, generalmente para esta clase de arreglo se usan ruedas Universales, aunque

el uso de ruedas orientables también permiten movimientos omnidireccionales.

Fuente: http://www.cytron.com.my

Figura 2.5 Robot omnidireccional de tres ruedas

Un robot omnidireccional de 3 ruedas posee la ventaja de tener un control simple, ya que para cada dirección y velocidad del vehículo existe sólo una combinación en las velocidades y direcciones de las ruedas (a diferencia de los vehículos omnidireccionales de más de 3 ruedas), lo cual se ilustra en la Figura 2.6.



Fuente: http://en.wikipedia.org

Figura 2.6 Componentes vectoriales en el movimiento de un robot omnidireccional de tres ruedas

2.1.2.2 Robot omnidireccional de 4 ruedas

Esta estructura consiste en una aproximación al omnidireccional de 3 ruedas y posee la ventaja de tener redundancia, es decir, que para lograr cualquier movimiento en el plano, existen varias combinaciones de velocidades posibles.

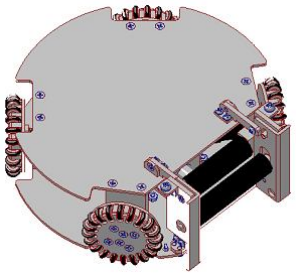


Figura 2.7 Robot omnidireccional de cuatro ruedas de tipo Universal

Las ruedas Mecanum también son escogidas para este tipo de estructuras, en donde, a diferencia de las ruedas universales, estas ruedas se ubican mostrando un arreglo similar a las ruedas de un coche.

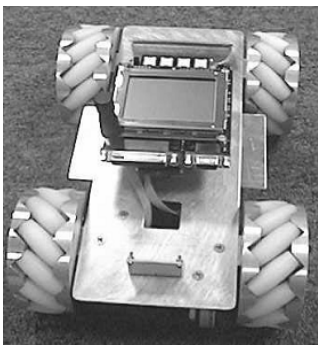


Figura 2.8 Robot omnidireccional con cuatro ruedas tipo Mecanum

Así mismo las ruedas orientables centradas son otra opción para este tipo de estructuras lo cual se observa en la Figura 2.9.

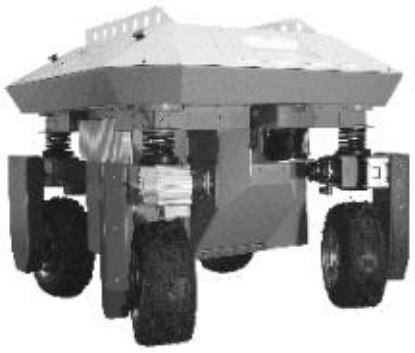
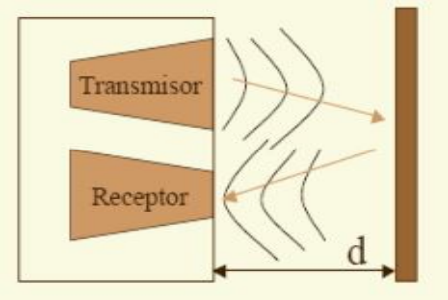


Figura 2.9 Robot omnidireccional con cuatro ruedas orientables centradas

2.2 Sensores de distancia o 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.



Fuente: http://sensorultrasonico.blogspot.com

Figura 2.10 Funcionamiento de un sensor de ultrasonido

2.3 Modulación PWM

Este tipo de modulación es muy utilizada en el campo de la robótica puesto que en la mayoría de las aplicaciones es necesario controlar la velocidad de un motor ya sea este servo, reductor o de algún otro tipo.

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.

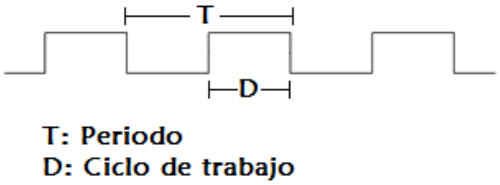


Figura 2.11 Tren de pulsos

El ciclo de trabajo de una señal periódica, es el tiempo que se mantiene en alto o en bajo (dependiendo la convención) con relación a su periodo, de esta manera la carga recibe un voltaje promedio dependiendo del valor del ciclo de trabajo.

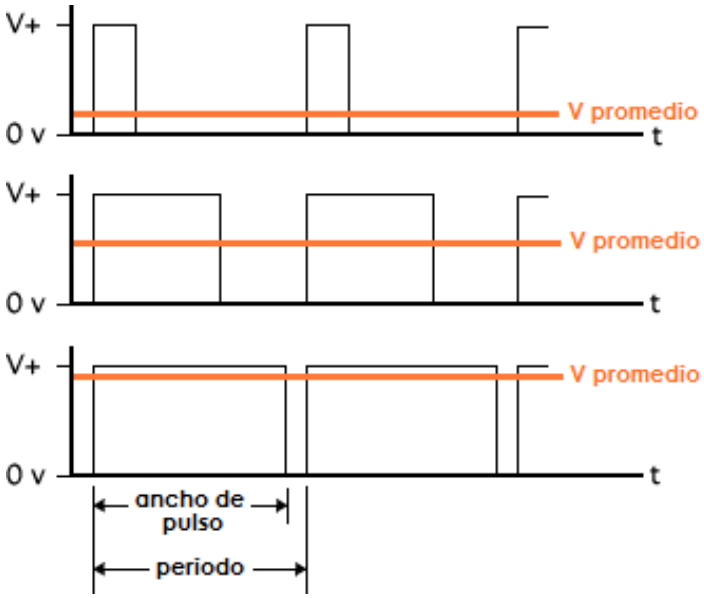


Figura 2.12 Señal PWM con diferentes ciclos de trabajo

2.4 Tecnología Utilizada

La implementación de un sistema que controle los movimientos del robot omnidireccional, active a los sensores, interprete las señales de los mismos y brinde al usuario una interfaz simple, es otro de los puntos a desarrollar en este proyecto. Por tal se hace un estudio sobre los principales temas relacionados con la tecnología de FPGA y sistemas embebidos configurables.

2.4.1 Lenguaje de descripción de hardware (HDL)

Un HDL es un lenguaje de programación de alto nivel usado para describir la entidad y arquitectura de un sistema electrónico. Los HDLs nacieron por la necesidad de diseñar sistemas complejos, los cuales a nivel de transistores eran difíciles de realizar.

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.

Dos HDLs muy conocidos en el campo de la lógica programable son: VHDL y Verilog. Verilog tiene una sintaxis similar a la del lenguaje de programación C, de manera que este lenguaje resulta familiar a los ingenieros por lo que es rápidamente aceptado. Como contraparte VHDL usa una sintaxis muy diferente a la del lenguaje C, ya que este lenguaje fue diseñado en base a los principios de la programación estructurada. La entidad (ENTITY) en VHDL es la declaración de las entradas y salidas de un módulo mientras que la arquitectura (ARCHITECTURE) es la descripción detallada de la estructura interna del módulo o de su comportamiento.

2.4.2 Sistemas embebidos configurables en FPGA

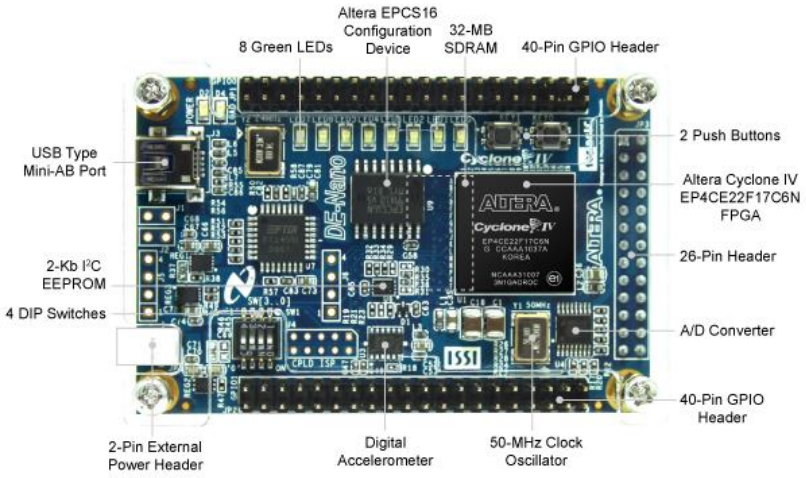
En general un sistema embebido consiste en un microcontrolador cuyo hardware y software están diseñados y optimizados para resolver un problema concreto de forma eficiente [3].

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.

2.4.3 Tarjeta de Desarrollo Altera DE0 NANO



Fuente: www.altera.com

Figura 2.13 Tarjeta 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.

Esta tarjeta cuenta con:

- Una FPGA CYCLONE IV EP4C22.
- Interfaces de entrada/salida como; puertos de expansión, convertidor analógico digital, acelerómetro de 3 ejes.
- Memoria: 32 MB de SDRAM ,2 Kb EEPROM.
- Conmutadores y Leds: 8 Leds verdes, 4 interruptores DIP, dos pulsadores sin rebote.

2.4.4 Procesador Nios II

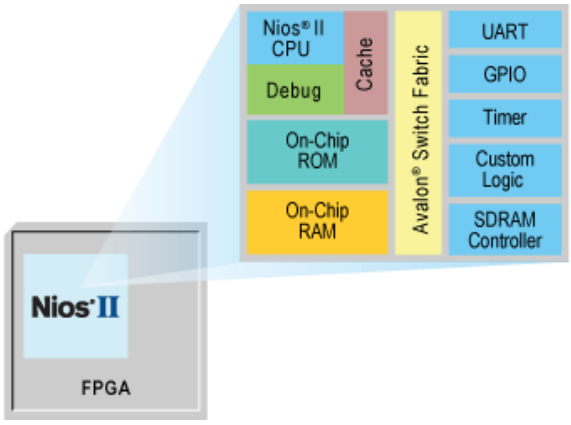
Nios II es un procesador de propósito general de tipo soft-core (puede ser configurado usando algún HDL) diseñado para varios FPGA de Altera. Este procesador posee una arquitectura tipo Harvard, debido a que usa buses separados para instrucciones y para datos.

Las principales características del procesador Nios II son las siguientes:

- Tamaño de palabra de 32 bits.
- Juego de instrucciones RISC de 32 bits.
- 32 registros de propósito general de 32 bits (r0 – r31).
- 6 registros de control de 32 bits (ctl0 - ctl5).
- 32 fuentes de interrupción externa.
- Capacidad de direccionamiento de 32 bits.
- Operaciones de multiplicación y división de 32 bits.
- Instrucciones dedicadas para multiplicaciones de 64 y 128 bits.
- Instrucciones para operaciones de coma flotante en precisión simple.
- Acceso a variedad de periféricos integrados e interfaces para manejo de memorias y periféricos externos.

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.[4]



Fuente: www.altera.com

Figura 2.14 Sistema basado en el procesador Nios II

2.4.5 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. Quartus II permite al desarrollador compilar sus diseños, realizar análisis temporales, examinar diagramas RTL y configurar el dispositivo de destino con el programador [5].

Entre sus ventajas se tiene:

- Posee editor de lenguaje simbólico.
- Posee un compilador que es capaz de recibir como entrada tanto archivos con la descripción del sistema (VHDL, VERILOG), como archivos con circuitos esquemáticos. Puede ejecutar simulaciones, usando como archivo de entrada uno creado por el editor de formas de onda.
- Puede programar un dispositivo FPGA especificando previamente su modelo.

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. SOPC Builder incorpora una biblioteca de componentes o IP CORES predefinidos, (incluyendo al procesador Nios II, los controladores de memoria, interfaces y periféricos) y una interfaz para la incorporación de otros componentes.

2.4.6 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 mediante la herramienta de programador del Quartus II. Al usar este IDE se puede programar de una manera más rápida y nítida para lograr el objetivo principal que es el de integrar el software con el hardware de los Sistemas Embebidos.

2.4.7 Interacción entre las herramientas

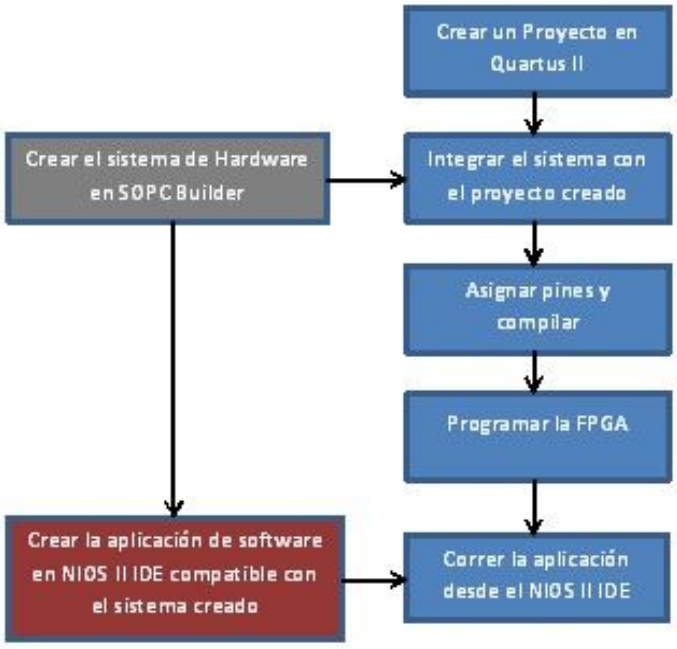


Figura 2.15 Diagrama de interacción de las herramientas usadas

La integración del sistema de hardware, creado en SOPC Builder, se puede realizar de dos maneras en Quartus II; ya sea mediante una herramienta gráfica donde se incluye al sistema en un archivo BDF o a través de la creación de un archivo de texto, usando un HDL, donde se declara la entidad y se la relaciona con las señales del sistema. Además se debe realizar la asignación de pines a la FPGA que se esté utilizando para el

desarrollo del proyecto, es muy importante incluir, en la declaración de la entidad, una señal de entrada como reloj y otra de reset y asignarlas a los pines correspondientes de la FPGA. La compilación del proyecto genera un archivo SOF que es usado para programar la FPGA con el fin de poder usarla para correr la aplicación de software creada en el NIOS II IDE.

Esta es la manera tradicional de realizar pruebas al sistema obtenido en Quartus II y a la aplicación creada en el NIOS II IDE, pero cuando el proyecto llega a término es importante conocer que se puede almacenar toda esta información de manera permanente en una memoria tipo flash, como por ejemplo la EPCS presente en la DE0 Nano.

CAPITULO 3

3 CONSTRUCCIÓN DEL ROBOT OMNIDIRECCIONAL

3.1 Estructura

Se utilizaron tres ruedas omnidireccionales de tipo universal, las cuales se encuentran ubicadas a 120° entre sí, de esta forma las ruedas coinciden con los vértices de un triángulo equilátero. La estructura 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. El marco fue construido utilizando acrílico de 6 mm de espesor, ya que a

espesores menores, el acrílico se dobla debido al peso que debe soportar.

3.2 Ruedas

Fuente: http://www.robotshop.com

Figura 3.1 Rueda omnidireccional Nexus con acoples

Se usaron ruedas de tipo universal doble, marca Nexus Robot. Cada rueda posee 8 rodillos (4 por lado) fabricados de caucho termoplástico, este material brinda alta fricción en superficies deslizantes como cerámicas y baldosas, razón por la cual se escogió este tipo de ruedas ya que se iba a trabajar en lugares cerrados y la mayoría de ellos poseen las superficies antes mencionadas. El diámetro de las ruedas es de 48 mm y su capacidad de carga es de 2 Kg. Estas ruedas

además incluyen acoples universales para servomotores, motores con caja reductora e incluso para la plataforma NXT de LEGO.

3.3 Motores y Drivers

Fuente: http://www.robotshop.com

Figura 3.2 Motores Dagu

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. Optamos por estos motores porque los acopladores que se incluye en las ruedas Nexus son compatibles con ellos. Entre las principales características de los motores tenemos:

- Voltaje de operación: 4.8 a 6 V
- Velocidad: 90 rpm a 4.8 V
- Torque: 3.5Kg/cm a 4.8 V

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. El L293D posee dos configuraciones puente H que internamente incluyen sus respectivos diodos de protección y permiten controlar dos motores de forma independiente.

3.4 Sensores ultrasónicos

Los sensores ultrasónicos empleados para realizar la detección de obstáculos son los que corresponden al código HC-SR04 y HY-SRF05. La diferencia entre ambos radica en que el SR04 consume 15mA mientras que el SRF05 posee un consumo de 2mA e incluye un nuevo modo de operación en el cual se usa un solo pin del sensor como entrada y salida.

Se utilizan un total de 5 sensores: 4 del tipo SR04 y 1 del tipo SRF05, 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.

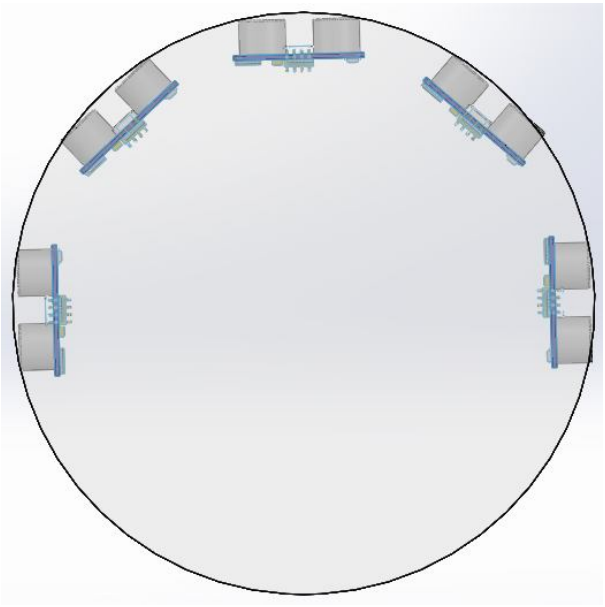


Figura 3.3 Disposición de los sensores en la estructura del robot

Ambos tipos de sensores poseen los siguientes pines:

- +VCC: Tensión positiva de alimentación.
- ECHO: Salida del pulso cuya anchura determina el tiempo del recorrido de la señal ultrasónica.
- TRIGGER: Entrada de inicio de una nueva medida. Se aplica un pulso con una duración mínima de 10 µs.

- OUT (sólo en el SRF05): Sin conexión se selecciona el modo 1 que tiene compatibilidad con el SRF04. Conectado a GND se selecciona el modo 2 de trabajo.
- GND: Tierra de alimentación.

Los dos tipos de sensores tienen en común un modo de trabajo el cual emplea los pines por separado, uno para aplicar el pulso de inicio o TRIGGER y otra para leer la anchura del pulso del ECHO medido. En el diagrama de tiempos de la Figura 3.4 se explica este modo de operación. Se aplica un pulso de disparo o TRIGGER, cuya duración mínima debe ser de 10 µs, el cual activa el sensor. El sensor transmite un tren de pulsos de 8 ciclos a 40KHz, luego la señal de salida ECHO pasa a nivel “1”. Cuando la cápsula receptora recibe la señal que ha sido reflejada al rebotar sobre un objeto, esta salida ECHO pasa de nuevo a nivel “0”. Se mide el tiempo en que la señal ECHO se mantiene en “1”, ya que este tiempo es proporcional a la distancia a la cual se encuentra el objeto.

Con objeto de que el sensor se estabilice, se debe dejar un lapso de tiempo de unos 20mS mínimo entre el momento en que la señal de ECHO pasa a “0” y un nuevo pulso de disparo que inicie el siguiente ciclo o medida. Esto permite realizar medidas cada 50 ms, ya que el

máximo si no detecta ningún objeto es de 30 ms, o lo que es igual a 20 medidas por segundo.

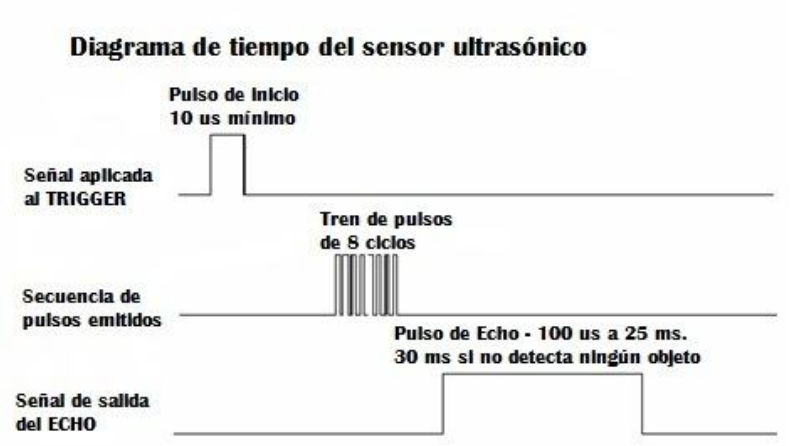


Figura 3.4 Diagrama de tiempo del sensor ultrasónico

La duración del pulso ECHO de salida varía entre 100µS y 25mS, en función de la distancia entre las cápsulas del módulo y el objeto. El inverso de la velocidad del sonido es de 29.15 µs/cm que, como realiza un recorrido de ida y vuelta, queda establecida en 58.30µS/cm. Así pues el rango mínimo que se puede medir es de 1.7 cm (100µS/58) y el máximo de 431 cm (25mS/58). [6]

3.5 Sistema de interconexión

Está basado en una placa de circuito impreso o PCB diseñado para las siguientes funciones:

- Alimentar a la tarjeta DE0 Nano, sensores y drivers (L293D) a través de dos reguladores de 5 V en paralelo.
- Conectarse con uno de los puertos de expansión de la DE0 Nano por medio de un puerto IDC de 40 pines.
- Proveer al usuario una botonera para reset, 10 puertos para comunicación de la FPGA con los sensores, 8 puertos que se conectan directamente a los drivers para controlar los motores y 6 puertos auxiliares para propósito general.

Este PCB fue diseñado en Altium a doble cara y facilita la interconexión entre todos los elementos que componen al robot con la tarjeta DE0 Nano. Se hace uso de 2 reguladores 7805 en paralelo con el fin de tener mayor capacidad de corriente ya que un solo regulador de este tipo tiene una salida de hasta 1 A. El 7805 permite un voltaje de entrada máximo de 36 V. El voltaje usado es de 9.6 V y se lo aplica mediante un conector JACK 2.5 ubicado en la periferia del PCB. El diagrama esquemático del PCB se muestra en el Anexo D.

Figura 3.5 Tarjeta de interconexión

3.6 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. Las señales de salida de los drivers que energizan a los motores reciben el voltaje del grupo de pilas.

3.7 Proceso de montaje de las piezas

Para el proceso de diseño y simulación de montaje de las partes en la estructura se utilizó el programa Solidworks con el cual creamos las piezas y las fuimos ensamblando como se muestra a continuación.

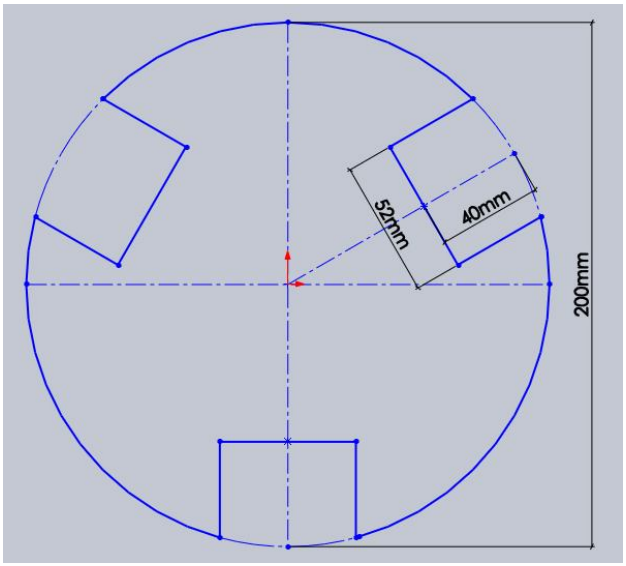


Figura 3.6 Dimensiones de la base de la estructura

La base de la estructura tiene forma circular con un radio de 10 cm y posee 3 cortes rectangulares cuyos centros se ubican uniformemente a 120° dentro del círculo, tiene también tres paredes laterales asentadas de manera que coincidan con los cortes rectangulares, en estas paredes se fijan los motores para asegurar un mejor balanceo. Los cortes se realizaron con la finalidad de que tanto motores como la mayor parte del cuerpo de las ruedas pertenezcan al nivel inferior y así no ubicarlos por debajo de la base.

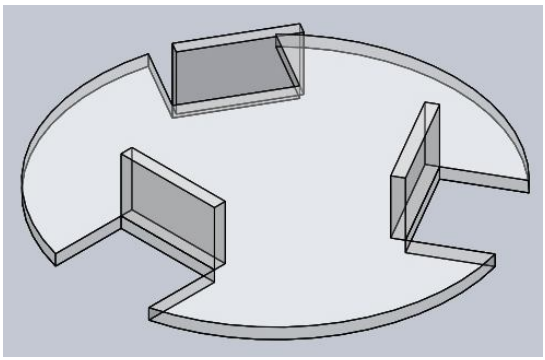


Figura 3.7 Vista de la base de la estructura

Se fijaron los motores a las paredes de la base usando de tornillos.

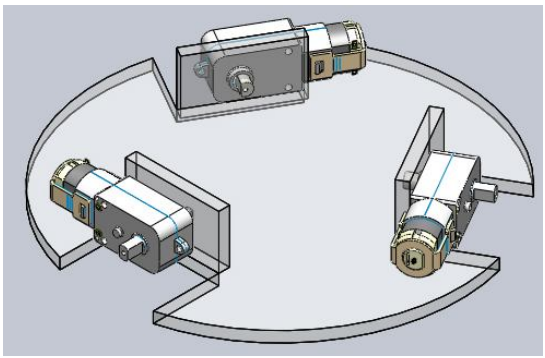


Figura 3.8 Base de la estructura con los motores incorporados

Se colocaron las baterías en el centro de la base y se acoplaron las ruedas a los motores.

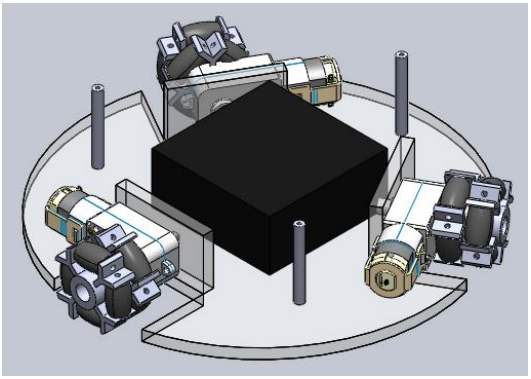


Figura 3.9 Nivel inferior de la estructura completo

Se adicionó el nivel superior que alberga a las tarjetas electrónicas y los sensores, la simetría fue un aspecto muy importante en el momento de la construcción por esta razón se trató de ubicar uniformemente todos los elementos dentro del marco, para de esta manera no tener problemas con el control de los movimientos de robot.

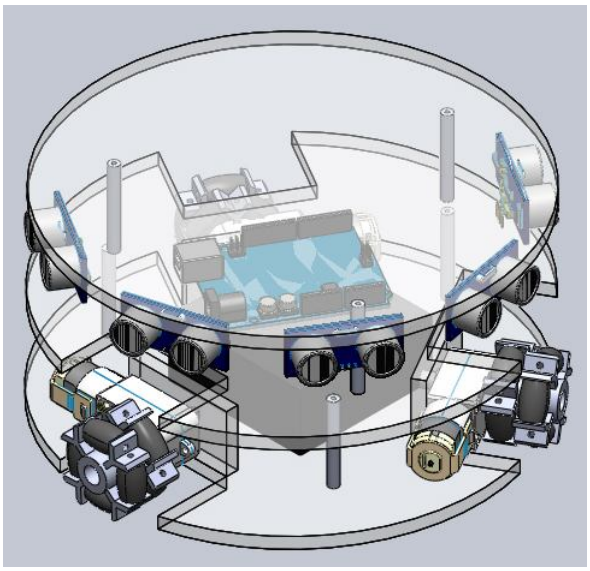
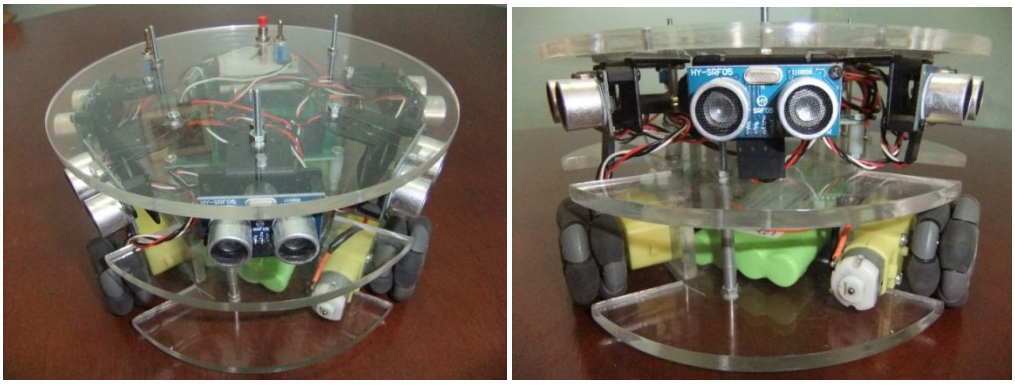


Figura 3.10 Robot omnidireccional ensamblado en Solidworks



(a) **(b)**

Figura 3.11 Robot Omnidireccional (a) Vista Superior (b) Vista Frontal

CAPÍTULO 4

4 DISEÑO E IMPLEMENTACIÓN

4.1 DISEÑO

4.1.1 Arquitectura del Hardware

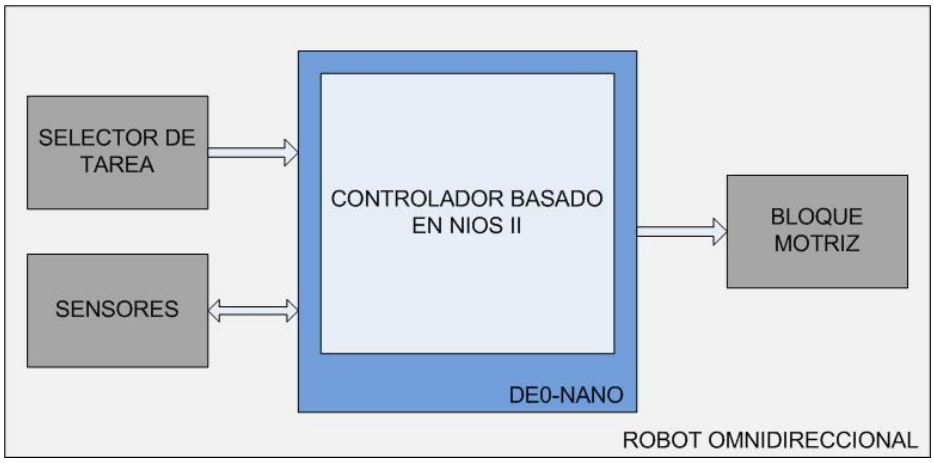


Figura 4.1 Integración componentes con el controlador

Selector de Tarea

Debido a que el robot tiene dos tareas programadas, la primera, realizar una rutina de movimientos omnidireccionales y la segunda, detectar y evadir obstáculos, se hace uso de un selector de tarea implementado con un interruptor simple.

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 mediantes 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

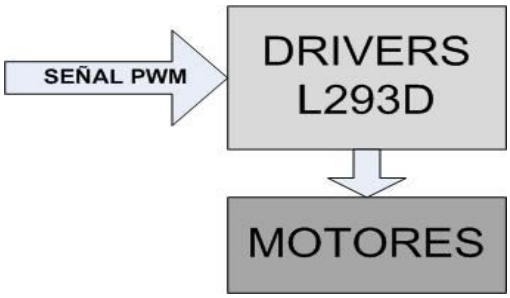


Figura 4.2 Esquema de control y fuerza

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.

4.1.2 Arquitectura del Software

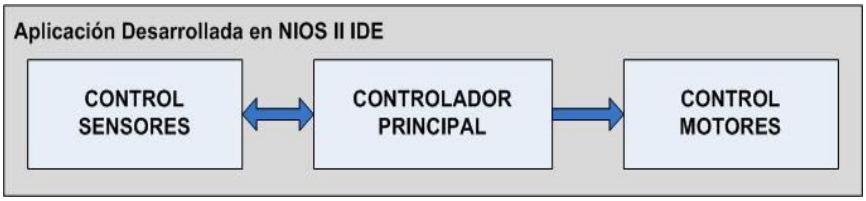


Figura 4.3 Diagrama de bloques de la arquitectura del software

Controlador Principal

Es el encargado de tomar todas las decisiones dentro de la aplicación, entre las cuales están: la distancia máxima permitida entre el robot y el obstáculo antes de cambiar de dirección, la dirección hacia donde desplazarse, el porcentaje de modulación que le corresponde a cada motor, entre otras.

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. Este bloque trabaja solamente en el modo detección y evasión obstáculos.

Control Motor

Genera las señales PWM enviadas a los drivers (posteriormente a los motores) y con esto controla los movimientos del robot, recibe del controlador principal la señal para habilitar y deshabilitar los motores además de cambiar el ciclo de trabajo y la frecuencia de las señales PWM.

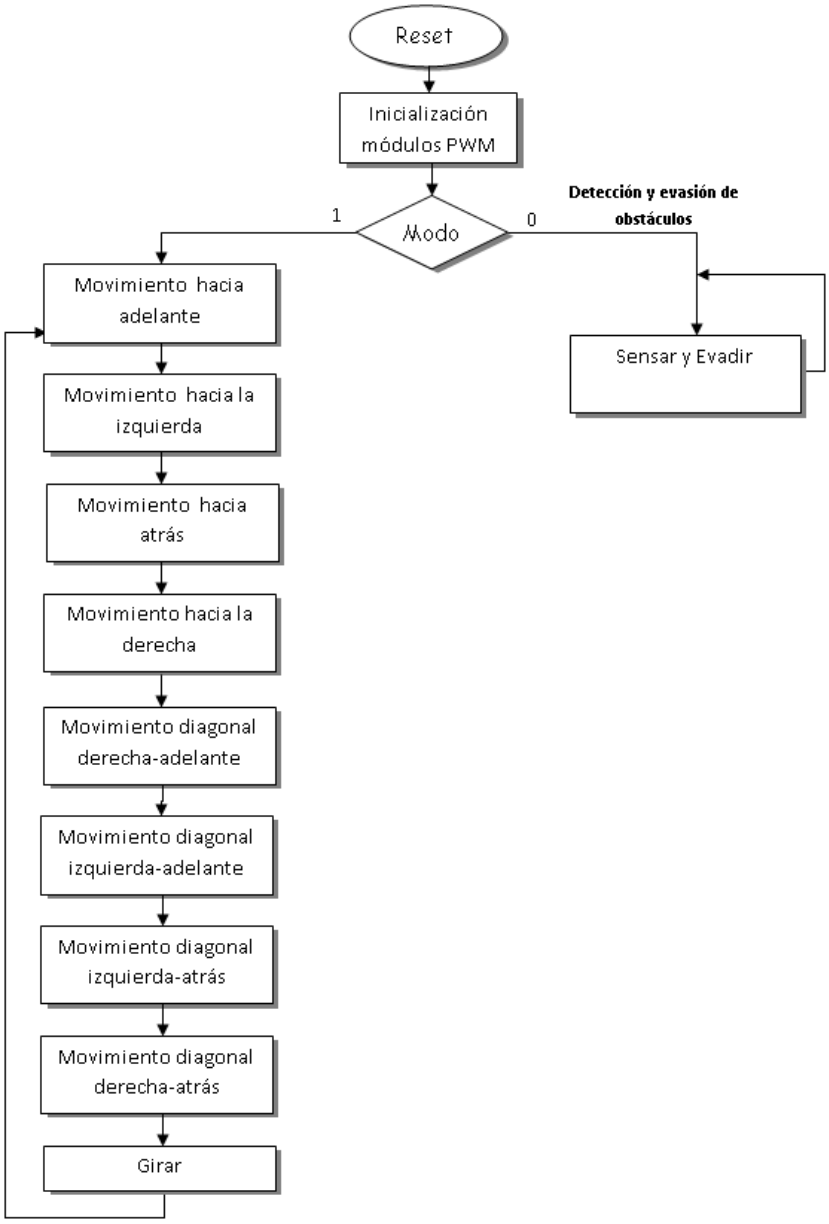


Figura 4.4 Diagrama de flujo de la aplicación en Nios II

Sensar y Evadir

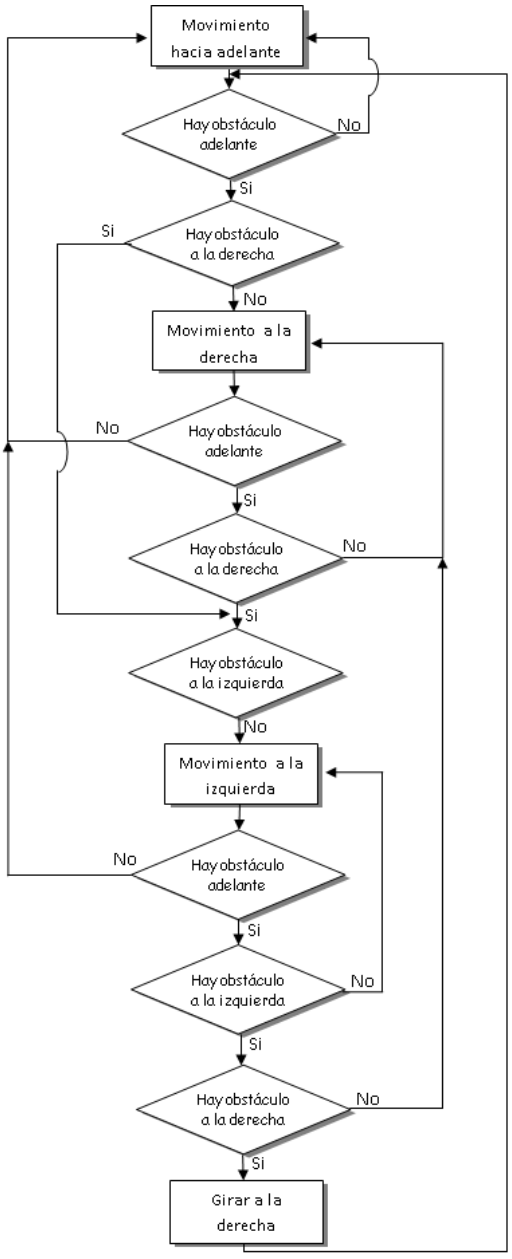


Figura 4.5 Diagrama de flujo de la evasión y detección de obstáculos

En el diagrama de flujo de la Figura 4.4 se detalla el algoritmo que el robot ejecuta, se inicializan los módulos PWM indistintamente del modo a operar, ya que en ambos se requiere el control de los motores por medio de una señal PWM.

En el caso que se haya elegido el modo de la rutina Demo Omnidireccional, el robot realiza diferentes desplazamientos, donde cada uno dura 4 segundos, la secuencia a seguir está especificada en el diagrama de flujo de la aplicación en NIOS II, en la Figura 4.4.

En el modo de evadir obstáculos, el robot inicialmente se mueve hacia adelante y si encuentra un obstáculo en frente, sensará al lado derecho, en caso de tener camino libre se moverá hacia la derecha hasta que el obstáculo de en frente haya desaparecido de su vista. Si al moverse a la derecha detecta un obstáculo, buscará en el lado izquierdo, y en caso de no existir obstáculo para avanzar a la izquierda se moverá hacia esa dirección. Si hubiera obstáculos adelante, izquierda y derecha, el robot girará hacia la derecha hasta que pueda avanzar adelante, a la izquierda o derecha.

Para cambiar de la rutina a la detección y evasión de obstáculos o viceversa se debe cambiar la posición del switch modo y presionar el botón reset.

4.2 IMPLEMENTACIÓN

4.2.1 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), esta versión fue escogida debido a que posee un alto rendimiento además de características como pipeline de 6 etapas, memoria caché para datos e instrucciones.

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. Estos CORES no están incluidos en la librería nativa del SOPC Builder por lo que se lo añadió manualmente. El procedimiento de añadir este componente se lo muestra en el Anexo C.

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. Esta memoria tiene una capacidad de 26000 bytes.

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, véase Anexo B.

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.

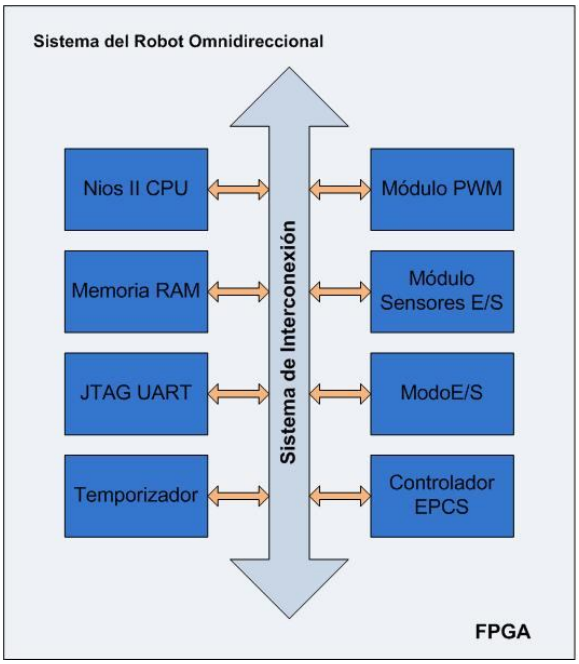


Figura 4.6 Componentes del hardware del sistema

El Sistema de interconexión que se indica en la Figura 4.6 corresponde a un bus Avalon el mismo que transparenta la comunicación entre los componentes y el CPU. Adicionalmente se tiene que configurar el procesador Nios II para que el vector de reset esté ubicado en la memoria EPCS con un offset de 0x00; de esta manera cuando ocurra un reset el procesador ejecuta las instrucciones en esta dirección de memoria y el vector de excepciones en la MEMORIA RAM con un offset de 0x20.

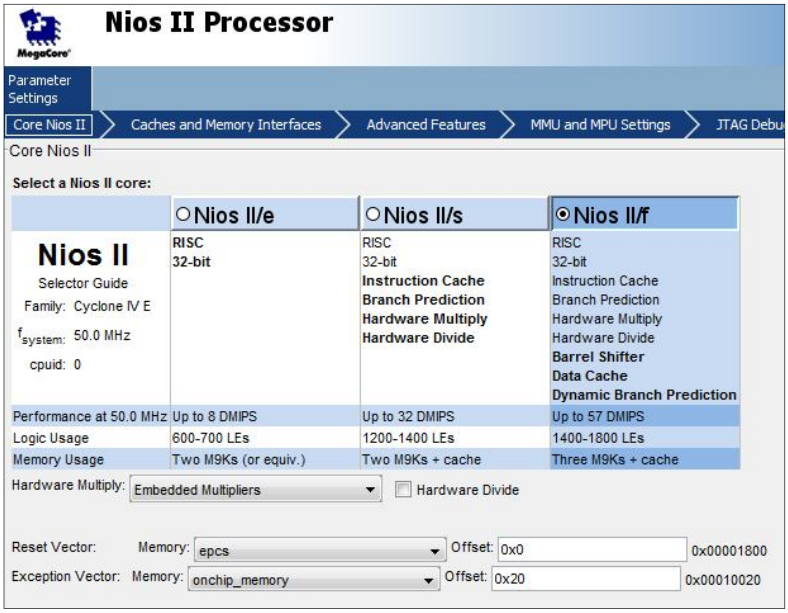


Figura 4.7 Configuración del Vector de reset y excepciones en el CPU

Cuando se selecciona como dirección del vector de reset a la dirección de la EPCS, internamente Quartus II, hace uso la memoria ROM que posee el módulo Controlador EPCS cargando en este el cargador de arranque, entonces luego de un reset el sistema ejecuta las instrucciones almacenadas en esta memoria ROM. Primero se analiza el contenido de la EPCS ya que aquí se almacena la información del sistema (SOF) pero además se guarda la aplicación creada en NIOS II, es esta aplicación la que se copia en la MEMORIA RAM y a su vez se le

pasa el control a la misma. La EPCS almacena estos dos archivos mediante el Flash Programmer, véase en la sección 4.3.

Luego de generar el sistema en SOPC Builder, para grabarlo en la FPGA, se debe tener una jerarquía de archivos para lo cual se creó un módulo usando el lenguaje de descripción de hardware Verilog, la finalidad de esta acción es de relacionar cada una de las señales de los componentes de nuestro sistema con las entradas y salidas declaradas en la entidad del módulo creado.

```
  Robot_system Robot_system_inst(
//señales globales
 .clk_50(CLOCK_50),
 .reset_n(RESETN),
//epcs
 .data0_to_the_epcs(EPCS_DATA0),
 .dclk_from_the_epcs(EPCS_DCLK),
 .sce_from_the_epcs(EPCS_NCSO),
 .sdo_from_the_epcs(EPCS_ASDO),
//pwm
 .pwm_out_from_the_pwm_avalon_interface_motor1_right(PWM_OUT1),
 .pwm_out_from_the_pwm_avalon_interface_motor1_left(PWM_OUT2),
 .pwm_out_from_the_pwm_avalon_interface_motor2_right(PWM_OUT3),
 .pwm_out_from_the_pwm_avalon_interface_motor2_left(PWM_OUT4),
 .pwm_out_from_the_pwm_avalon_interface_motor3_right(PWM_OUT5),
 .pwm_out_from_the_pwm_avalon_interface_motor3_left(PWM_OUT6),
//sensores
 .out_port_from_the_trigger_1(TRIGGER_1),
 .in_port_to_the_echo_1(ECHO_1),
 .out_port_from_the_trigger_2(TRIGGER_2),
 .in_port_to_the_echo_2(ECHO_2),
 .out_port_from_the_trigger_3(TRIGGER_3),
 .in_port_to_the_echo_3(ECHO_3),
 .out_port_from_the_trigger_4(TRIGGER_4),
 .in_port_to_the_echo_4(ECHO_4),
 .out_port_from_the_trigger_5(TRIGGER_5),
 .in_port_to_the_echo_5(ECHO_5),
  // modo
 .in_port_to_the_modo(MODO));
```

Figura 4.7 Código del módulo en Verilog

Finalmente se asignaron las entradas y salidas del módulo a los pines físicos de la FPGA usando la herramienta de Quartus II Pin Planner. Se realiza la compilación para obtener un archivo de programación SOF.

4.2.2 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 y para la detección y evasión de obstáculos se crearon funciones que se detallan a continuación.

4.2.2.1 Funciones para control de los motores

Para controlar los motores se usó la librería altera_avalon_pwm_routines.h, donde los parámetros principales para inicializar los módulos PWM son la frecuencia de la señal pwm y el ciclo de trabajo.

Las funciones utilizadas de esta librería son:

```
int altera_avalon_pwm_init(unsigned int address, unsigned int clock_divider, unsigned int duty_cycle);
```

Esta función inicializa el Módulo PWM; el parámetro clock_divider es el divisor de reloj más 1, número para el cual se divide la frecuencia a la cual está trabajando la tarjeta DE0 Nano, y permite obtener la frecuencia de la señal PWM; en este caso se trabaja con un clock_divider

de 15001; duty_cycle permite obtener el tiempo que la señal está en bajo dividiendo ese valor para la frecuencia de trabajo de la tarjeta DE0 Nano y varía de acuerdo al movimiento del robot.

int altera_avalon_pwm_enable(**unsigned int** address);

Con esta función se habilita el Módulo PWM cuya dirección base está indicada por address.

int altera_avalon_pwm_disable(**unsigned int** address);

Con esta función se deshabilita el Módulo PWM cuya dirección base está indicada por address

int altera_avalon_pwm_change_duty_cycle(**unsigned int** address, **unsigned int** duty_cycle);

Permite cambiar el ciclo de trabajo de la señal PWM, se debe tomar en cuenta que este valor no debe exceder al clock_divider con el cual fue configurado ese módulo PWM

Para usar un módulo PWM se emplean las funciones en el siguiente orden:

```
1 altera_avalon_pwm_init(MOTOR1_RIGHT_BASE,  15001, 6500);
2 altera_avalon_pwm_enable(MOTOR1_RIGHT_BASE);
3 altera_avalon_pwm_disable(MOTOR1_RIGHT_BASE);
4 IOWR_ALTERA_AVALON_PWM_DUTY_CYCLE(MOTOR1_RIGHT_BASE,6200);
```

Figura 4.8 Uso de las funciones de la librería altera_avalon_pwm_routines.h

1. Inicializa el módulo PWM con un clock_divider de 15001 y duty_cycle de 6500.

2. Activa el módulo PWM.

3. Cuando se desee deshabilitar el módulo se usa esta función.

4. Cambia el ciclo de trabajo a 6200.

4.2.2.2 Detección y evasión de obstáculos

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

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 4.9 Código de la función Read_distance

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

1. Se inicializa el Timer con un valor de 0XFFFFFFFF.

2. 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.

Distance

```
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 4.10 Código de la 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.

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 4.11.

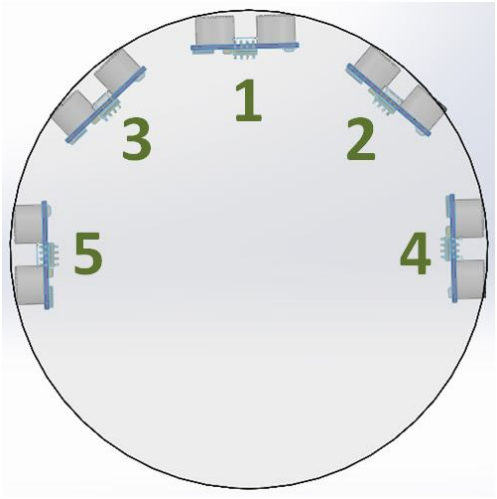


Figura 4.11 Numeración de los sensores en la estructura

4.3 PROGRAMACION DE LA MEMORIA FLASH

Al compilar la aplicación en Nios II IDE se genera un archivo ejecutable tipo ELF, el cual contiene las líneas de código, traducidas a lenguaje ensamblador para que puedan ejecutarse en el procesador Nios II del sistema.

La memoria flash (EPCS16) de la DE0 Nano se debe programar con dos archivos, el SOF y el ELF, convertidos a formato FLASH; para esto se utilizó el Flash Programmer, herramienta del Nios II IDE, en el cual se realizaron las siguientes acciones:

1. La conversión del archivo SOF a un archivo tipo FLASH compatible con la memoria EPCS16.
2. La conversión del archivo ELF a un archivo tipo FLASH compatible con la memoria EPCS16.
3. Ambos archivos convertidos se graban en la memoria Flash de la DE0 Nano.

De esta manera tanto el Hardware como el Software del controlador del robot omnidireccional se almacenan permanentemente en la memoria EPCS. Este procedimiento se lo muestra en el Anexo B.

CAPÍTULO 5

5 Pruebas y Resultados

5.1 Prueba 1: Distancia de detección de obstáculos (Sensor 1)

Descripción

En esta prueba se configuran diferentes distancias de detección de obstáculos con el fin de medir el tiempo de respuesta del robot, es decir, que pueda evadir los obstáculos sin antes colisionar con los mismos. El robot avanza con una velocidad aproximada de 0.14 m/s.

Resultado esperado

Se espera que el robot pueda esquivar correctamente un objeto cuando lo detecte a 10 cm en frente de él, si se configura una distancia menor a 10 cm, el robot no alcanzará a esquivarlo por completo.

Resultado obtenido

Escenario	Distancia[cm]	Chocó con el objeto
1	12	No
2	11	No
3	10	No
4	9	No
5	8	Si
6	7	Si

Tabla 5.1 Resultados de la prueba: Distancia de detección de obstáculos

De acuerdo a la tabla la distancia mínima a la cual el robot puede detectar y evadir completamente un objeto es 9 cm, pero se optó por configurar una distancia de 10 cm ya que 9 cm por ser la mínima

distancia el robot evade un objeto pero algunas veces pasa muy cerca de él, casi rozándolo.

5.2 Prueba 2: Movimiento omnidireccional

Descripción

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

Resultado

El robot ha realizado con éxito la trayectoria que ha sido programada. En la Figura 5.1 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 5.1 Rutina del robot omnidireccional

(a) Trayectoria en línea recta (b) Trayectoria diagonales (c) Giro sobre su propio eje

5.3 Prueba 3: Detección y evasión de obstáculos

Descripción

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

Escenario 1.1

A cada uno de los sensores del robot se ha configurado una distancia de detección de 10 cm.

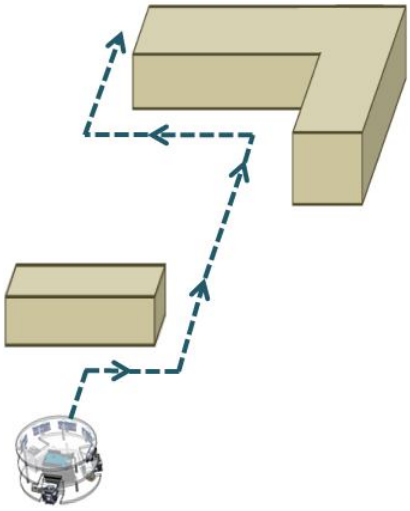
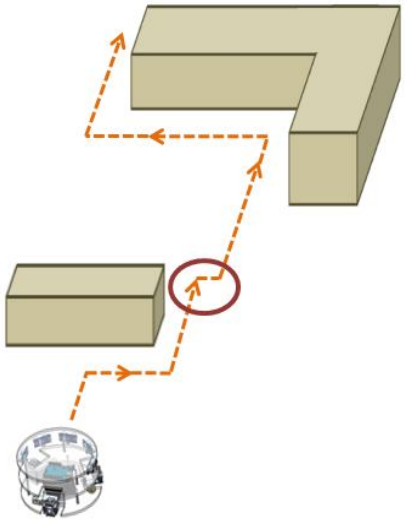
Resultado esperado	Resultado obtenido
	

Tabla 5.2 Resultados de la prueba 3 Escenario 1.1

Como se puede observar el robot no tiene colisiones en su recorrido pero no cumple con lo esperado debido a que el sensor 3 detecta el obstáculo y cambia su recorrido para esquivarlo, sin embargo se pudo visualizar que si no cambiaba la de trayectoria tampoco colisionaría con el objeto.

Escenario 1. 2

Los sensores 2 y 3 según la numeración de la Figura 4.11, son configurados con 8 cm de distancia de detección.

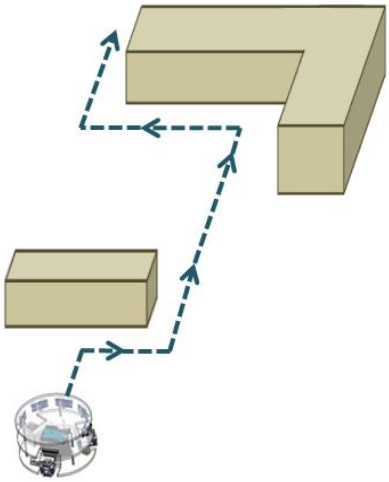
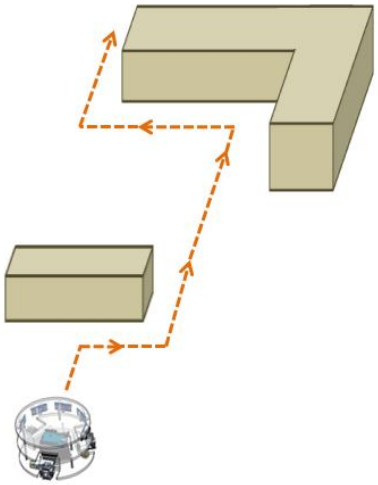
Resultado esperado	Resultado obtenido
	

Tabla 5.3 Resultados de la prueba 3 Escenario 1.2

En el mismo escenario anterior robot fue ubicado en una posición inicial diferente y se observa que al cambiar las distancias de detección de los sensores 2 y 3 se cumple con el resultado esperado, se esquiva los obstáculos con gran facilidad y no se presenta colisión alguna durante el recorrido.

Escenario 2

Se mantuvieron las distancias que fueron configuradas en el escenario 1.2 para los sensores 2 y 3.

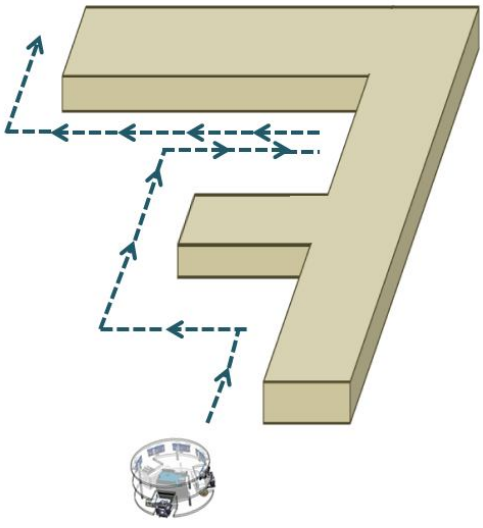
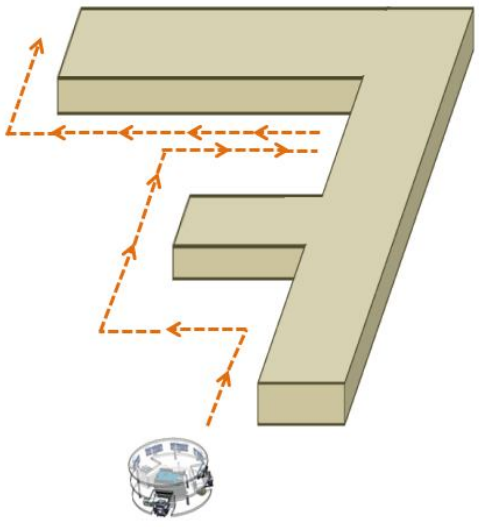
Resultado esperado	Resultado obtenido
	

Tabla 5.4 Resultados de la prueba 3 Escenario 2

En este escenario se aumentó el grado de dificultad de tal manera que se obtenga una trayectoria más compleja, para lo cual se obtuvo que; el robot cumple con lo esperado y de igual manera no se presentan colisiones en su recorrido.

Escenario 3

En este escenario el robot ingresa a un callejón sin salida por lo cual según la programación realiza un giro, sobre su propio eje, de 90º en sentido horario para luego intentar salir del callejón.

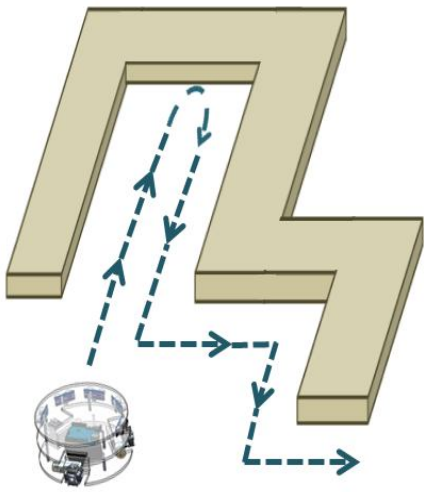
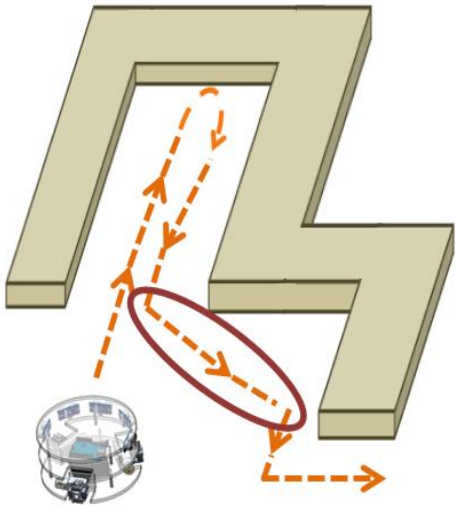
Resultado esperado	Resultado obtenido
	

Tabla 5.5 Resultados de la prueba 3 Escenario 3

En esta prueba no se obtuvo el resultado esperado ya que el robot realizo una especie de trayectoria en zigzag para poder salir del callejón ya que el giro realizado fue superior a los 90º estimados, sin embargo el robot no tuvo ninguna colisión en su recorrido de salida.

5.4 Prueba 4: Medición del tiempo de Procesamiento

Se mide el tiempo de procesamiento del procesador Nios II implementado en nuestro sistema de hardware para la función read_Distance() a diferentes distancias de detección, y además se realiza la implementación de esta función en un microcontrolador Atmel ATMega328p para de igual manera medir su tiempo de procesamiento. El código de la prueba realizada con el microcontrolador ATMega328p se encuentra en el Anexo F.

Resultado obtenido

Distancia[cm]	Tiempo de Procesamiento [µs]	
	DE0 Nano	Arduino
10	29965	31200
20	30585	31808
50	32321	33560
100	35224	36464
145	37842	39072

Tabla 5.6 Resultados de la prueba 4

El procesador Nios II que está incluido en nuestro sistema procesa instrucciones con mayor velocidad que la del procesador AVR del microcontrolador de ATMega328p.

Este resultado se debe a que el procesador Nios II ejecuta una instrucción cada ciclo de reloj y además está trabajando a 50 MHz; por otro lado, el microcontrolador ATMega328 trabajó a 8 MHz para esta prueba; cabe recalcar que puede trabajar máximo hasta 20 MHz pero aun así no habría realizado un menor tiempo de procesamiento que el procesador embebido Nios II.

Recordemos que el procesador Nios II es de tipo soft-core, es decir, podemos configurarlo para optimizar el procesamiento, a diferencia del núcleo de procesamiento de los microcontroladores que ya vienen optimizados e implementados como hardware desde fábrica.

CONCLUSIONES

1. El desarrollo de este proyecto demostró que la tarjeta de desarrollo DE0 Nano puede ser usada para implementar un sistema de control de movimientos y trayectorias de un robot omnidireccional, pero se debe tener en cuenta que es dentro de la FPGA de la DE0 Nano donde se llevan a cabo todos los procesos que comandan las funciones del robot.

2. La facultad de poder separar hardware y software para implementar el sistema de control del robot dentro de una FPGA fue de gran importancia en el desarrollo del proyecto dado que primero diseñamos a la plataforma de hardware embebida en la FPGA según nuestros

requerimientos, para luego poder elaborar fácilmente una aplicación que sea compatible con esta plataforma y que pueda hacer uso de los recursos de la misma.

3. El diseño mecánico del robot jugó un papel importante para el desarrollo de la aplicación, ya que la combinación de las señales PWM que reciben los motores dependen del tipo de ruedas utilizadas y la disposición de las mismas para realizar cada movimiento, además que cualquier error de construcción mecánica del robot se pudo corregir a través del software.

4. El algoritmo de lectura de los sensores, el algoritmo de detección de objetos junto con las distancias de detección debidamente configuradas produjeron un resultado acorde a lo esperado ya que permiten al robot tener un buen tiempo de respuesta al momento de detectar el obstáculo, lo que le permite detenerse, cambiar de dirección y evadirlo sin colisionar.

5. La memoria flash EPCS, la cual es frecuentemente utilizada para almacenar la configuración de la FPGA, pudo ser usada para guardar

el archivo ejecutable de la aplicación desarrollado en el Nios II IDE. Usar la memoria para guardar este archivo ejecutable permite al robot tener una autonomía completa ya que el mismo no se pierde cada vez que se quita energía a la tarjeta DE0 Nano.

6. Se detectaron buenos resultados en el control de los movimientos del robot sobre todo en las pruebas de detección y evasión de obstáculos las que incluían cambios bruscos de dirección con lo se comprueba la superioridad de los robot omnidireccionales frente a los convencionales.

RECOMENDACIONES

1. Se debe tener siempre en cuenta que la distancia de detección depende de la velocidad a la que se mueve el robot ya que si este se desplaza rápidamente no se podrán evitar las colisiones debido al tiempo que le toma desde procesar la lectura de los sensores hasta cambiar de dirección.

2. Las pruebas o demostraciones deben ser realizadas de preferencia en superficies planas ya que terrenos irregulares podrían causar problemas en el recorrido.

3. Se recomienda añadir un disipador a los reguladores de la PCB del sistema de interconexión, o a su vez sustituir el uso de dos reguladores en paralelo por el uso de un circuito que incluya un transistor de potencia, para evitar el calentamiento de los reguladores por la demanda de corriente.

4. Es aconsejable el uso de baterías recargables construidas para carros a control remoto o automodelismo puesto que estas presentan un desempeño muy bueno y mayor duración.

TRABAJOS FUTUROS

- El control de movimientos del robot puede ser optimizado incluyendo un controlador PID, de tal manera que sea posible obtener la velocidad de los motores y poder compensarlas de ser necesario mientras el robot se desplaza, lo que permitiría un mejor desempeño para diversas condiciones de trabajo como un cambio de nivel de voltaje de las baterías o alguna variación mecánica.

- Incorporar un sistema de comunicación por radio frecuencia que permita controlar remotamente al robot, obtener información de los sensores, nivel de baterías y velocidad desde un computador seria de mucha utilidad para futuras aplicaciones.

- Almacenar la información del entorno, y los recorridos que tiene el robot dentro de una memoria externa ayudaría al robot a no repetir rutas e incluso a poder llegar a un objetivo de forma más rápida.

BIBLIOGRAFÍA

[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] Victor Grosu Ioan Doroftei y Veaceslav Spinu. Bioinspiration and Robotics: Walking and Climbing Robots, Capitulo 29 Omnidirectional Mobile Robot - Design and Implementation, páginas 544. I-Tech Education and Publishing, Technical University of Iasi, Romania, September 2007

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

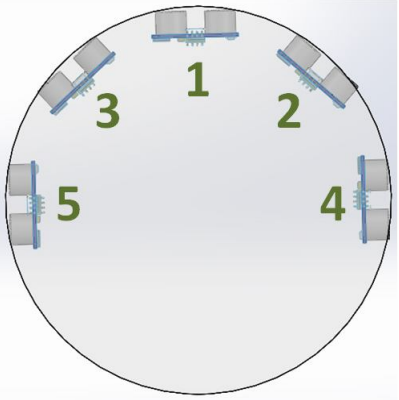
[4] Jorge Rodríguez Araujo, Estudio del microprocessor Nios II, Marzo 2010

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

[6] Sensores Ultrasónicos SFR05 y SR04, http://www.msebilbao.com/notas/downloads/Medidor%20Ultrasonico%20SRF05.pdf

ANEXO A

Tabla de combinaciones para los movimientos del robot omnidireccional



Combinación Nº	Sensor					Movimiento
	2	5	4	3	1	
0	0	0	0	0	0	Girar
1	0	0	0	0	1	Girar
2	0	0	0	1	0	Girar
3	0	0	0	1	1	Girar
4	0	0	1	0	0	Girar
5	0	0	1	0	1	Girar
6	0	0	1	1	0	Girar

7	0	0	1	1	1	Girar
8	0	1	0	0	0	Girar
9	0	1	0	0	1	Girar
10	0	1	0	1	0	Izquierda
11	0	1	0	1	1	Izquierda
12	0	1	1	0	0	Girar
13	0	1	1	0	1	Girar
14	0	1	1	1	0	Izquierda
15	0	1	1	1	1	Izquierda
16	1	0	0	0	0	Girar
17	1	0	0	0	1	Girar
18	1	0	0	1	0	Girar
19	1	0	0	1	1	Adelante
20	1	0	1	0	0	Derecha
21	1	0	1	0	1	Derecha
22	1	0	1	1	0	Derecha
23	1	0	1	1	1	Adelante

24	1	1	0	0	0	Girar
25	1	1	0	0	1	Girar
26	1	1	0	1	0	Izquierda
27	1	1	0	1	1	Adelante
28	1	1	1	0	0	Derecha
29	1	1	1	0	1	Derecha
30	1	1	1	1	0	Derecha
31	1	1	1	1	1	Adelante

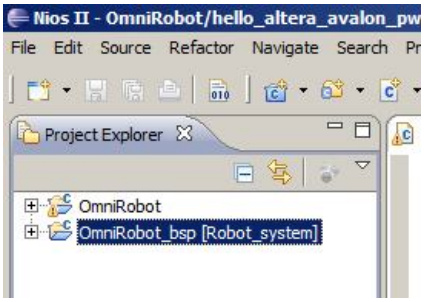
0: El sensor detecta obstáculo

1: El sensor no detecta obstáculo

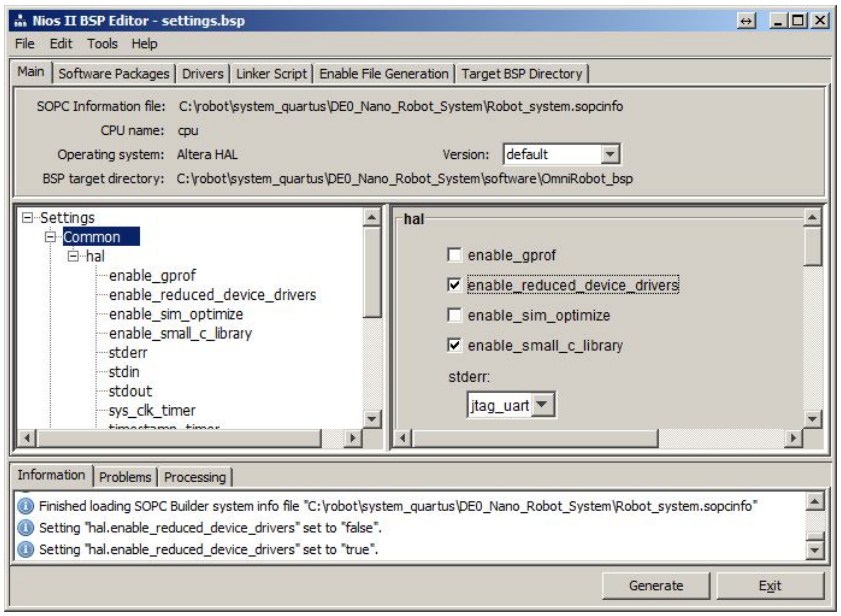
ANEXO B

Pasos para grabar en la Memoria Flash utilizando el Flash Programer

1.- Antes de compilar, dar clic derecho en OmniRobot_BSP y seleccionar **Nios II > BSP Editor.**

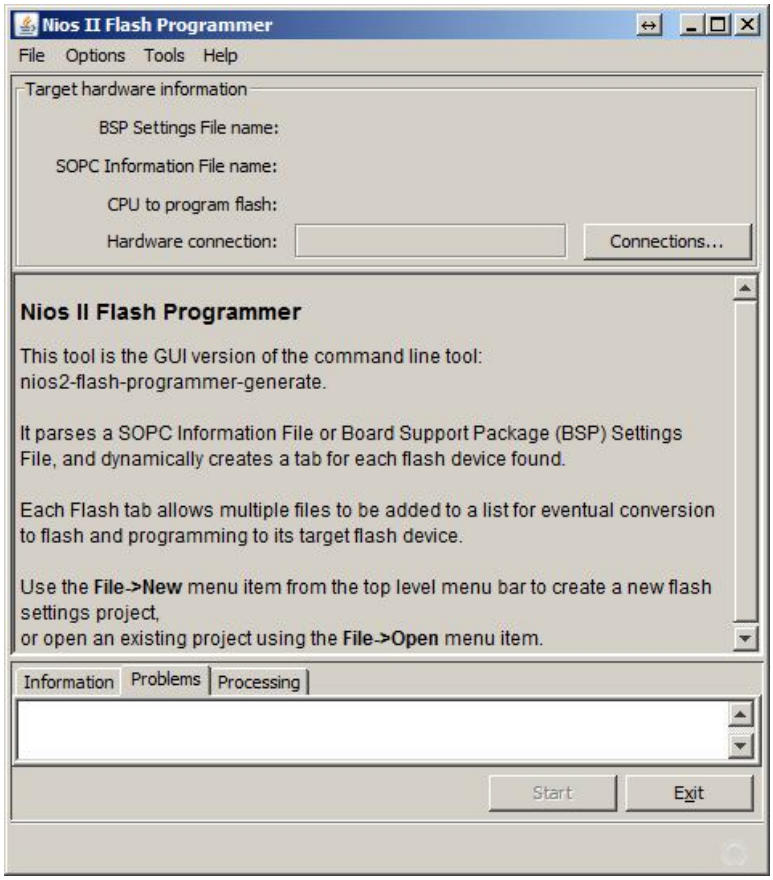


2.- En la opción hal activar **enable_reduced_device_drivers** y **enable_small_c_library,** dar clic en **Generate** y **Exit**

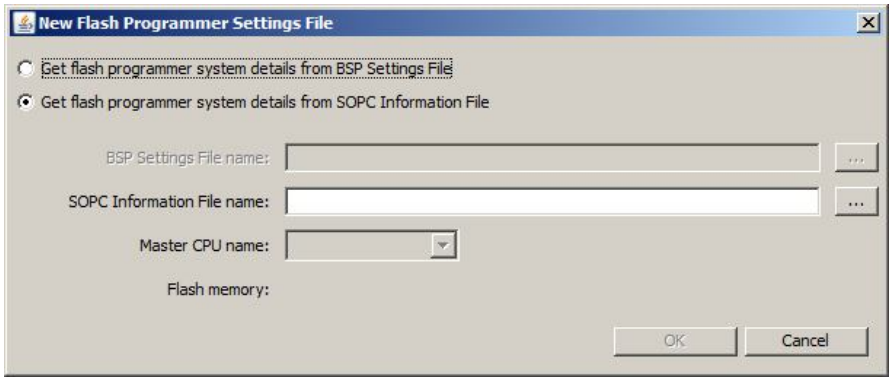


3. - Dar clic en **Build All**

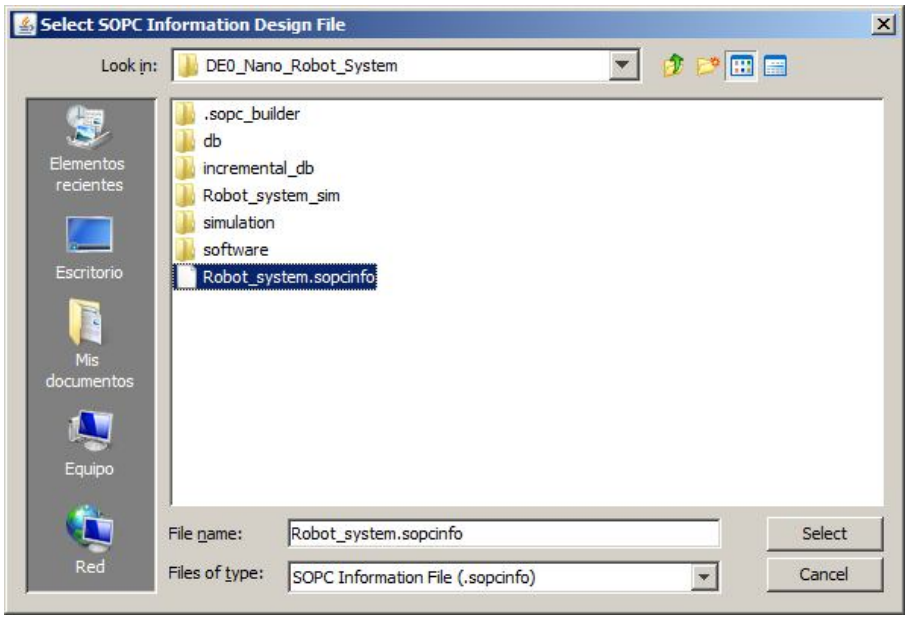
4.- En el menú de Eclipse seleccionar **Nios II > Flash Programmer**



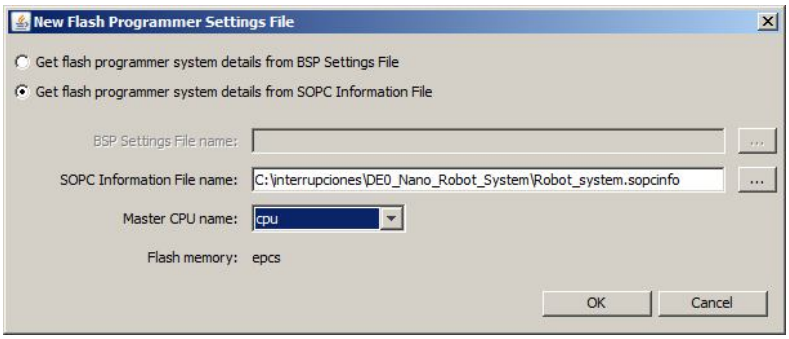
5.- Dentro del Flash Programmer seleccionar **File > New**



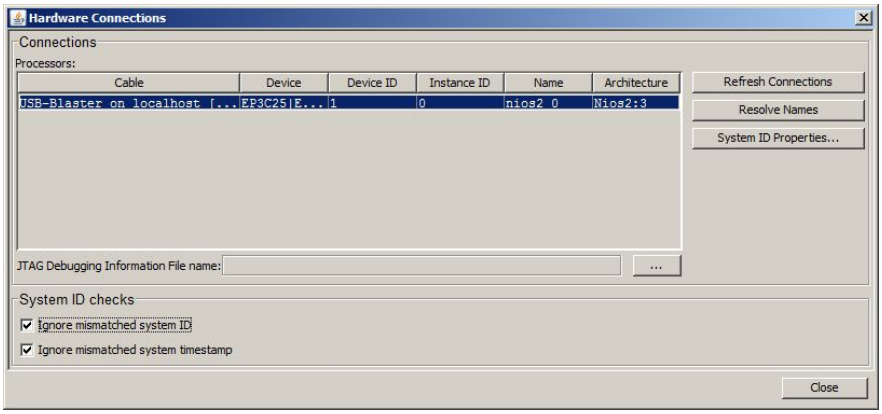
6.- Escoger **Get flash programmer system details from SOPC Information File** y **seleccionar el SOPCINFO del sistema generado en SOPC Builder.**



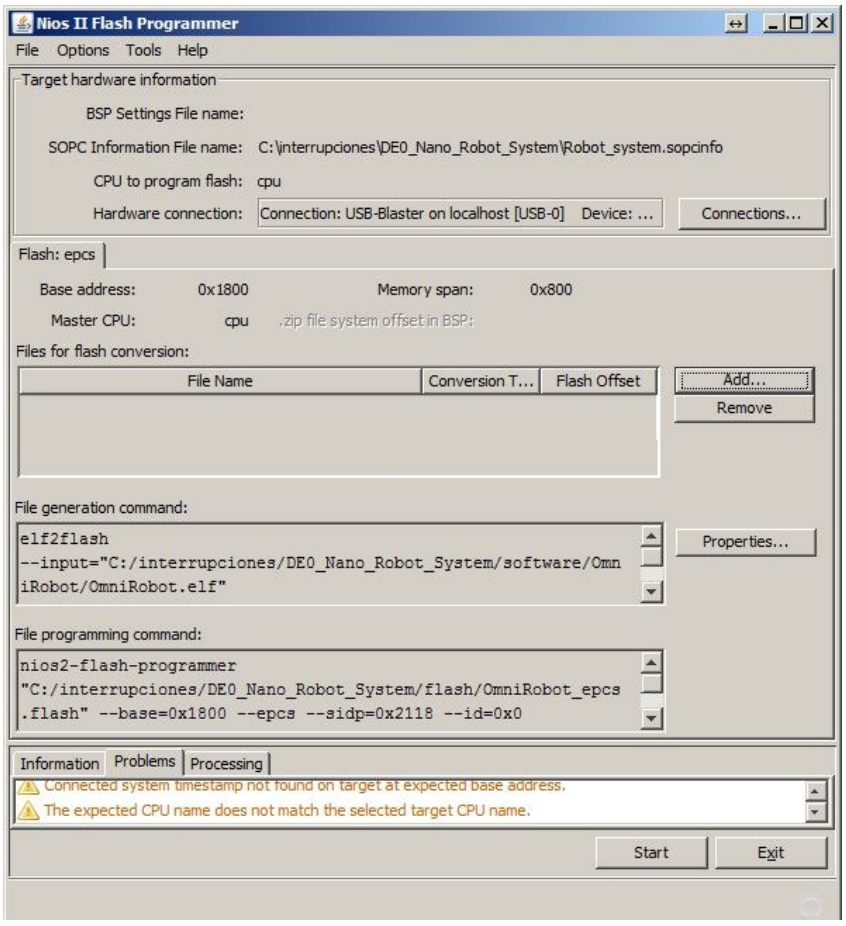
7.- Dar clic en **OK**



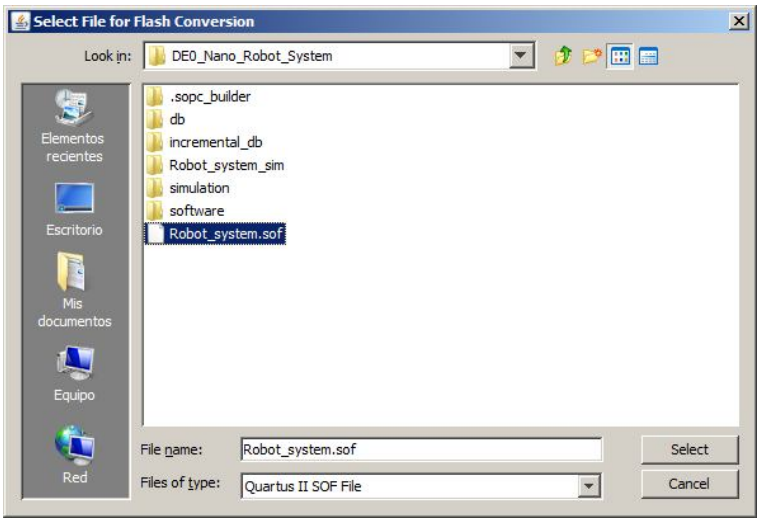
8.- Si se muestra algún mensaje de error dar click en **Connections…** y seleccionar **Ignore mismatched system ID** e **Ignore mismatched system timestamp**



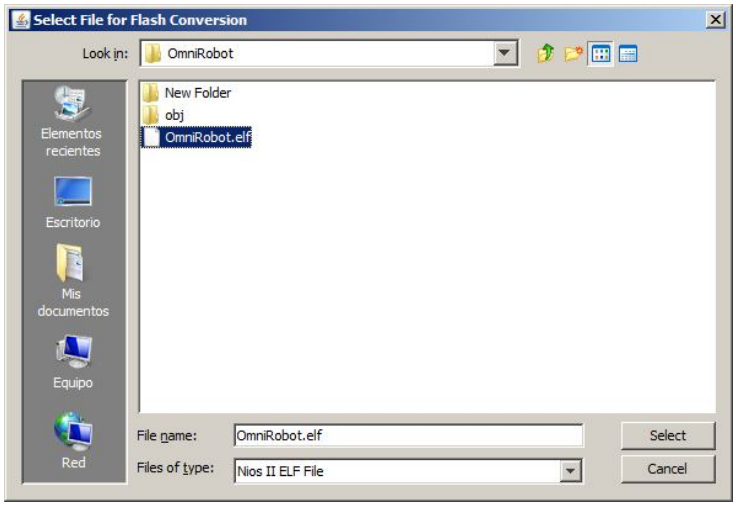
9.- Ahora en la ventana del **Flash Programmer** se muestra la información de la memoria EPSC incluida en nuestro sistema



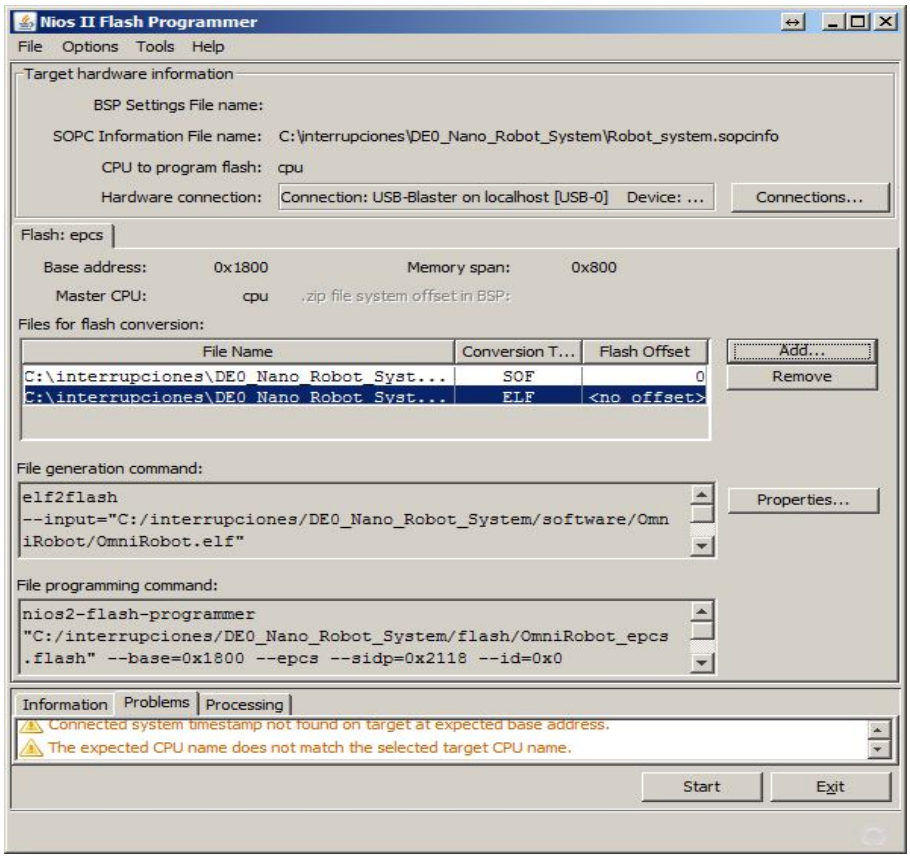
10.- Dar clic en el botón **Add..** y añadir el archivo **SOF** resultado de la compilación del proyecto en **Quartus II**



10.- Nuevamente dar clic en el botón **Add..** y añadir el archivo **ELF** resultado de la compilación del proyecto en **NIOS II IDE**



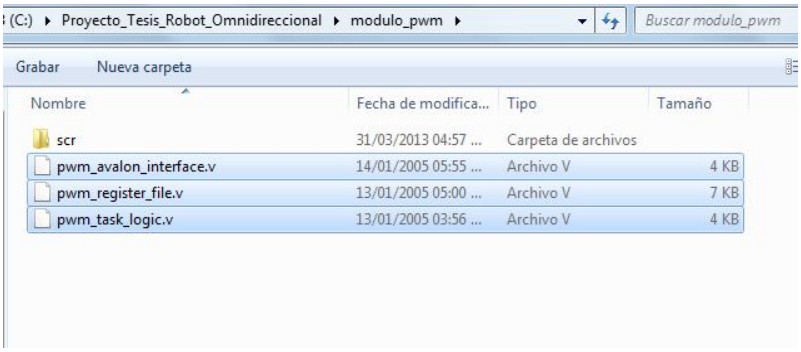
11.- Se muestran los dos archivos que se convertirán a formato **FLASH** ahora dar clic en el boton **START** para realizar la conversión y luego de esto automaticamente el **Flash Programmer** grabara los archivos convertidos en la memoria **EPCS** del sistema


Nios II Flash Programmer
File Options Tools Help
Target hardware information
BSP Settings File name:
SOPC Information File name: C:\interrupciones\DE0_Nano_Robot_System\Robot_system.sopcinfo
CPU to program flash: cpu
Hardware connection: Connection: USB-Blaster on localhost [USB-0] Device: ...
Connections...
Flash: epcs
Base address: 0x1800
Memory span: 0x800
Master CPU: cpu
.zip file system offset in BSP:
Files for flash conversion:
File Name
Conversion T...
Flash Offset
C:\interrupciones\DE0_Nano_Robot_Syst...
SOF
0
C:\interrupciones\DE0_Nano_Robot_Syst...
ELF
<no offset>
Add...
Remove
File generation command:
elf2flash
--input="C:/interrupciones/DE0_Nano_Robot_System/software/Omn
iRobot/OmniRobot.elf"
Properties...
File programming command:
nios2-flash-programmer
"C:/interrupciones/DE0_Nano_Robot_System/flash/OmniRobot_epcs
.flash" --base=0x1800 --epcs --sidp=0x2118 --id=0x0
Information
Problems
Processing
Connected system timestamp not found on target at expected base address.
The expected CPU name does not match the selected target CPU name.
Start
Exit

ANEXO C

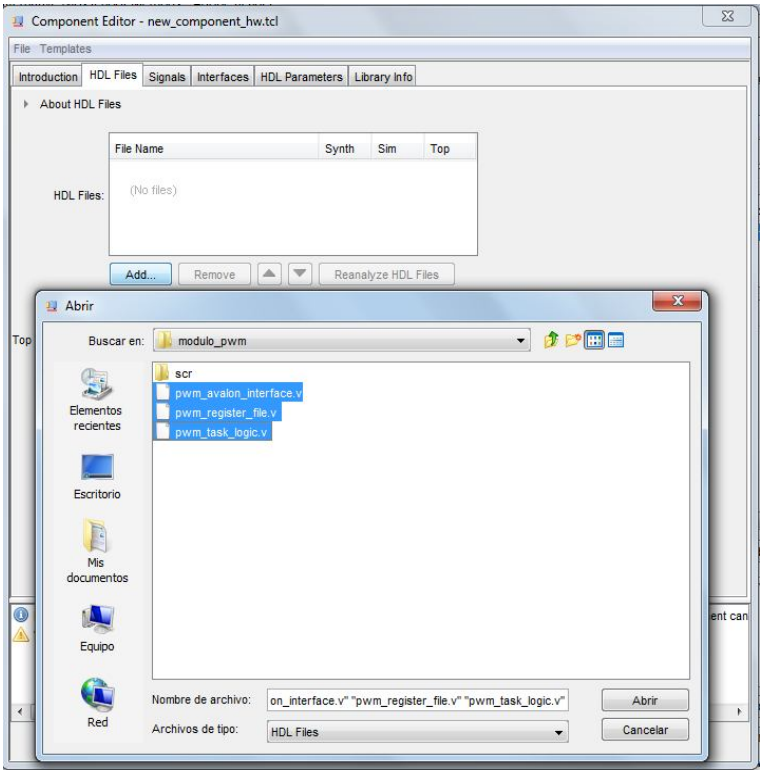
Agregando el módulo PWM al SOPC Builder

La carpeta modulo_pwm incluye archivos fuentes escritos en Verilog del módulo PWM que vamos a agregar al SOPC Builder y será usado en nuestro sistema, además dentro de la carpeta llamada scr se incluyen las librerías con la declaración de las funciones que se usarán en Nios II IDE, para el desarrollo de la aplicación, así como también sus implementaciones.

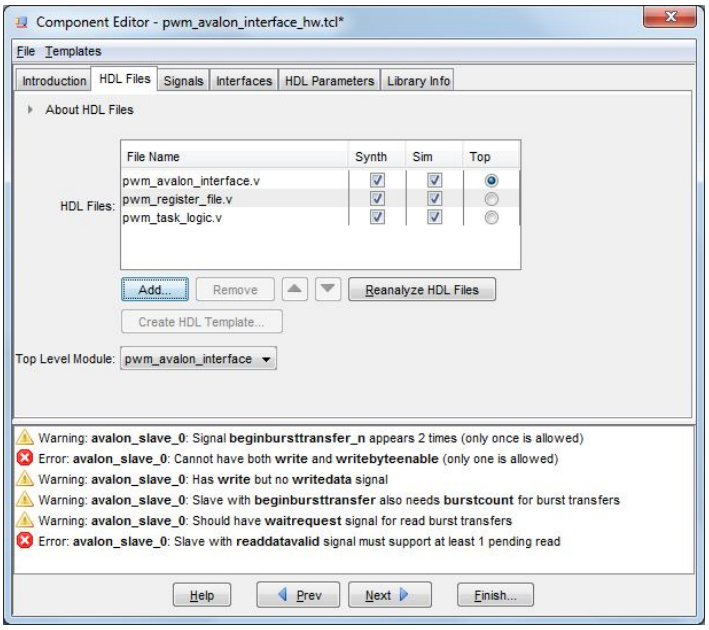


1.- En el cuadro izquierdo de la ventana principal del SOPC Builder, se da doble click a la opción ***New component..,*** luego de lo cual se abrirá la ventana del ***Component Editor*** donde seleccionamos la pestaña ***HDL Files*** y damos click

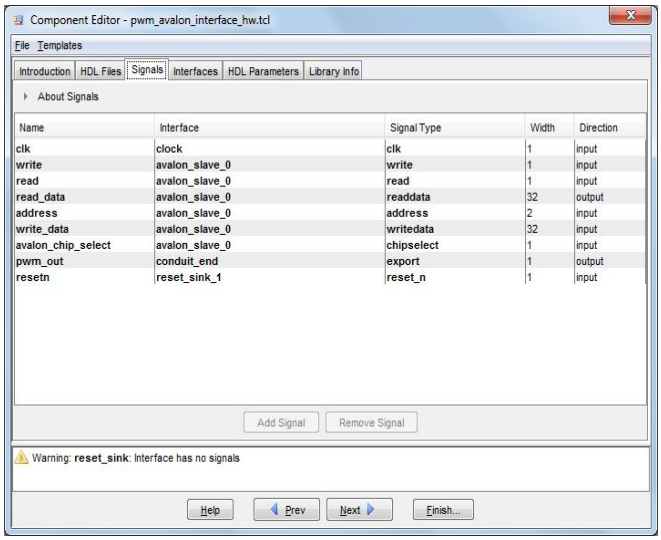
en el botón ***Add..***, se abrirá una ventana que nos permite seleccionar los archivos escritos en HDL que contienen la descripción del componente a añadir, aquí seleccionamos los tres archivos.



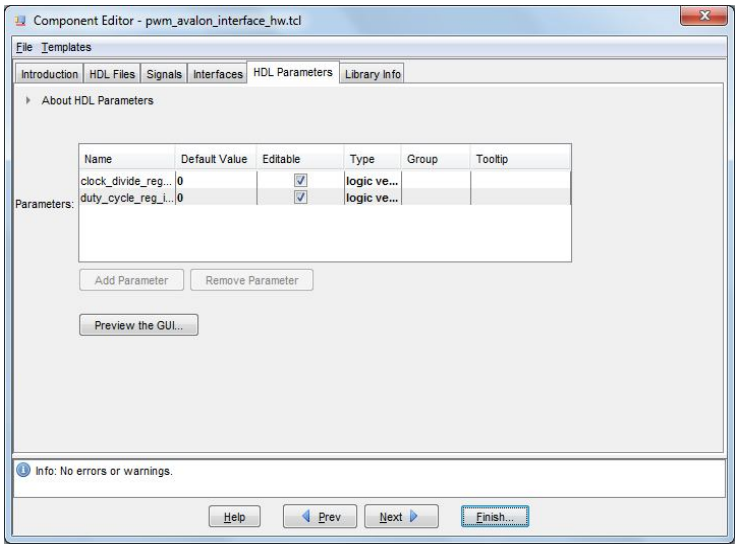
2.- Ahora se debe seleccionar el archivo principal para la creación del componente, en este caso, los mensajes de error no son de importancia en este momento ya que aún debemos configurar la entidad del componente, presionamos ***Next***.



3.- Se configuran las señales de entrada y salida como también los buses de datos y direcciones, de tal manera que coincidan con el tipo de señal requerida, en esta ventana se muestra toda la información de la entidad del componente. La señales de tipo ***export*** son aquellas disponibles para conectarlas con pines de las FPGA, el resto se conectan internamente en el sistema creado en SOPC Builder.



4.- Ahora revisamos los parámetros con los que cuenta este componente, los mismos que se modificarán desde NIOS II IDE, usando las funciones de las librerías mencionadas anteriormente. Luego de presionar el botón ***Finish..,*** el componente estará disponible para usarlo dentro de nuestro sistema.


Component Editor - pwm_avalon_interface_hw.tcl
File Templates
Introduction
HDL Files
Signals
Interfaces
HDL Parameters
Library Info
About HDL Parameters
Parameters:
Name
Default Value
Editable
Type
Group
Tooltip
clock_divide_reg...
0
logic ve...
duty_cycle_reg_i...
0
logic ve...
Add Parameter
Remove Parameter
Preview the GUI...
Info: No errors or warnings.
Help
Prev
Next
Finish...

ANEXO D

Diagrama esquemático

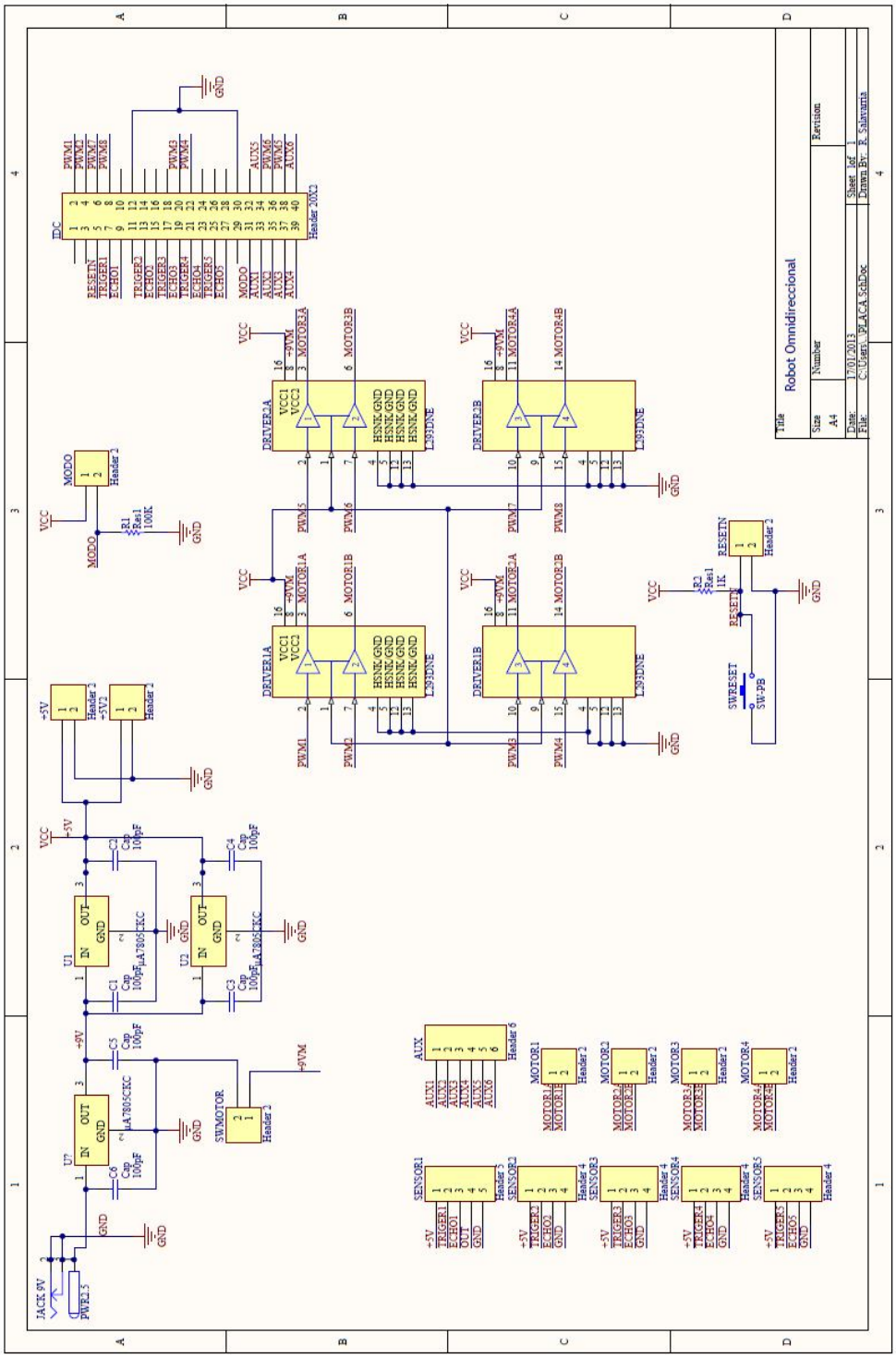
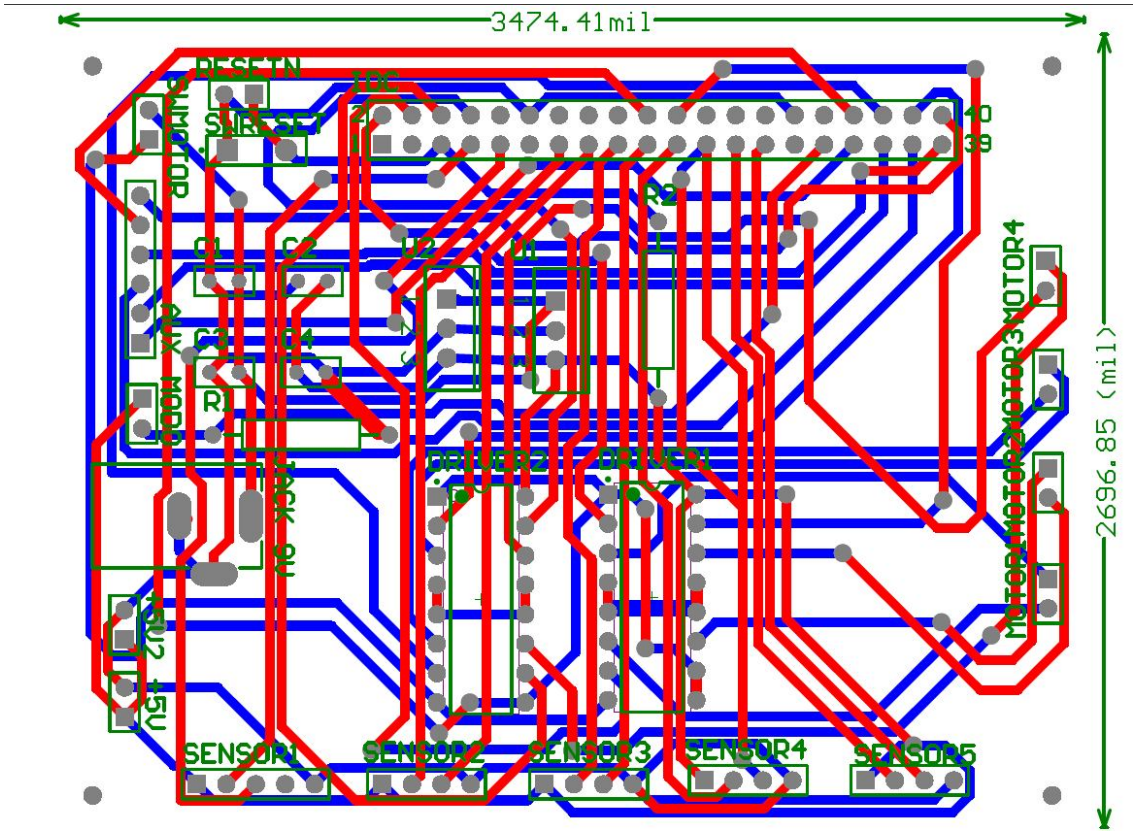


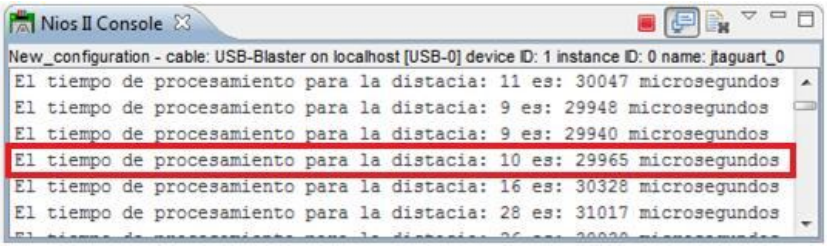
Diagrama de pistas del PCB

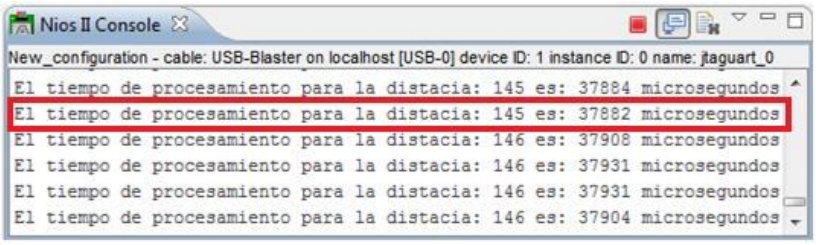


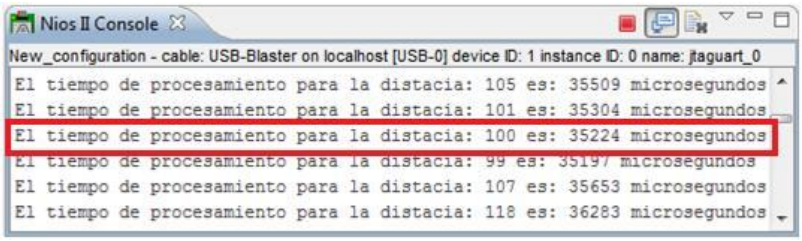
ANEXO E

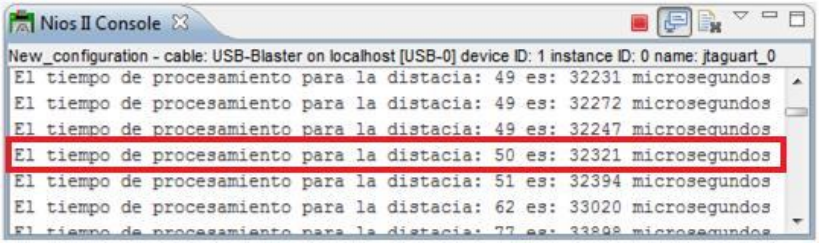
Resultados de las prueba Medición del tiempo de procesamiento

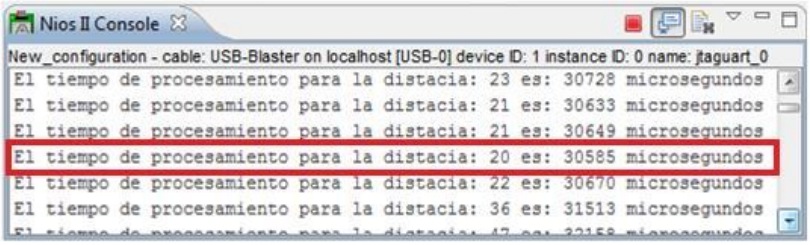
DE0 Nano



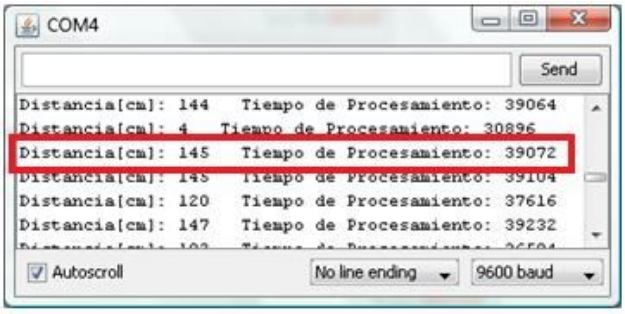


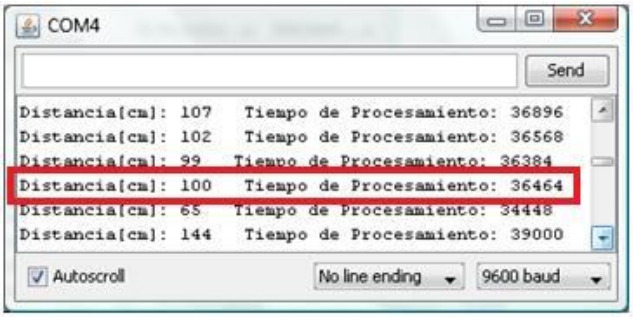


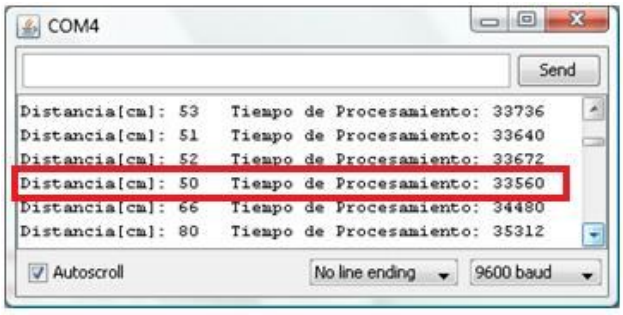


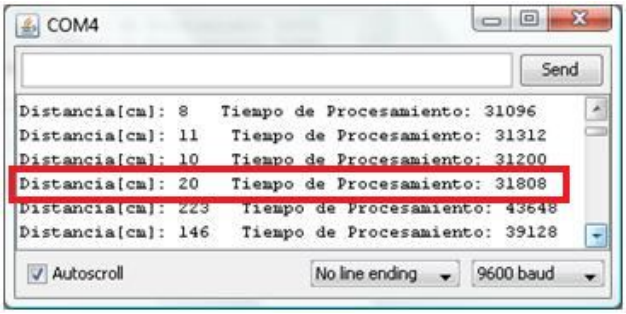

Nios II Console
New_configuration - cable: USB-Blaster on localhost [USB-0] device ID: 1 instance ID: 0 name: jtaguart_0
El tiempo de procesamiento para la distacia: 23 es: 30728 microsegundos
El tiempo de procesamiento para la distacia: 21 es: 30633 microsegundos
El tiempo de procesamiento para la distacia: 21 es: 30649 microsegundos
El tiempo de procesamiento para la distacia: 20 es: 30585 microsegundos
El tiempo de procesamiento para la distacia: 22 es: 30670 microsegundos
El tiempo de procesamiento para la distacia: 36 es: 31513 microsegundos

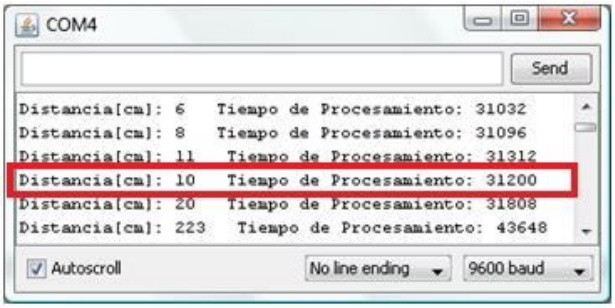
Arduino


COM4
Send
Distancia[cm]: 144 Tiempo de Procesamiento: 39064
Distancia[cm]: 4 Tiempo de Procesamiento: 30896
Distancia[cm]: 145 Tiempo de Procesamiento: 39072
Distancia[cm]: 120 Tiempo de Procesamiento: 37616
Distancia[cm]: 147 Tiempo de Procesamiento: 39232
Autoscroll
No line ending
9600 baud


COM4
Send
Distancia[cm]: 107 Tiempo de Procesamiento: 36896
Distancia[cm]: 102 Tiempo de Procesamiento: 36568
Distancia[cm]: 99 Tiempo de Procesamiento: 36384
Distancia[cm]: 100 Tiempo de Procesamiento: 36464
Distancia[cm]: 65 Tiempo de Procesamiento: 34448
Distancia[cm]: 144 Tiempo de Procesamiento: 39000
Autoscroll
No line ending
9600 baud

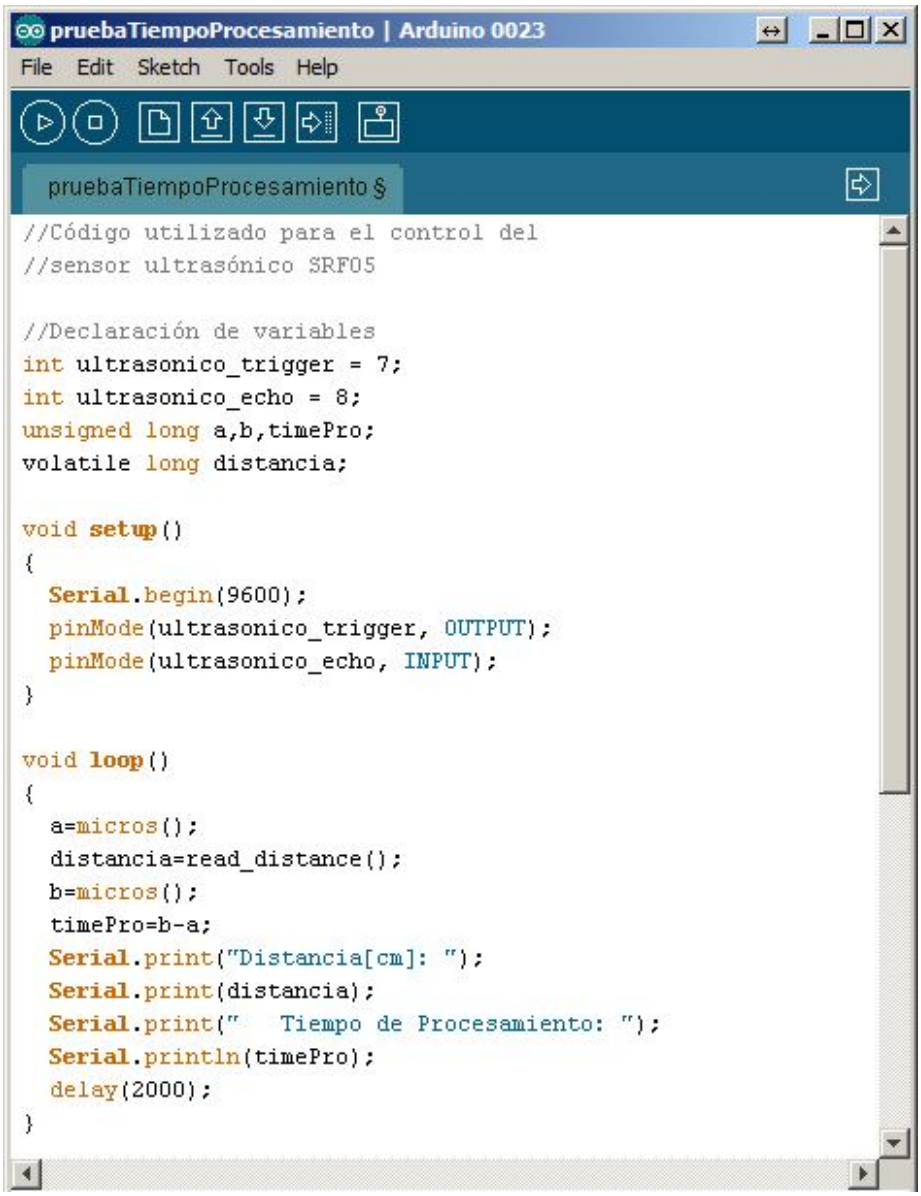

COM4
Send
Distancia[cm]: 53 Tiempo de Procesamiento: 33736
Distancia[cm]: 51 Tiempo de Procesamiento: 33640
Distancia[cm]: 52 Tiempo de Procesamiento: 33672
Distancia[cm]: 50 Tiempo de Procesamiento: 33560
Distancia[cm]: 66 Tiempo de Procesamiento: 34480
Distancia[cm]: 80 Tiempo de Procesamiento: 35312
Autoscroll
No line ending
9600 baud

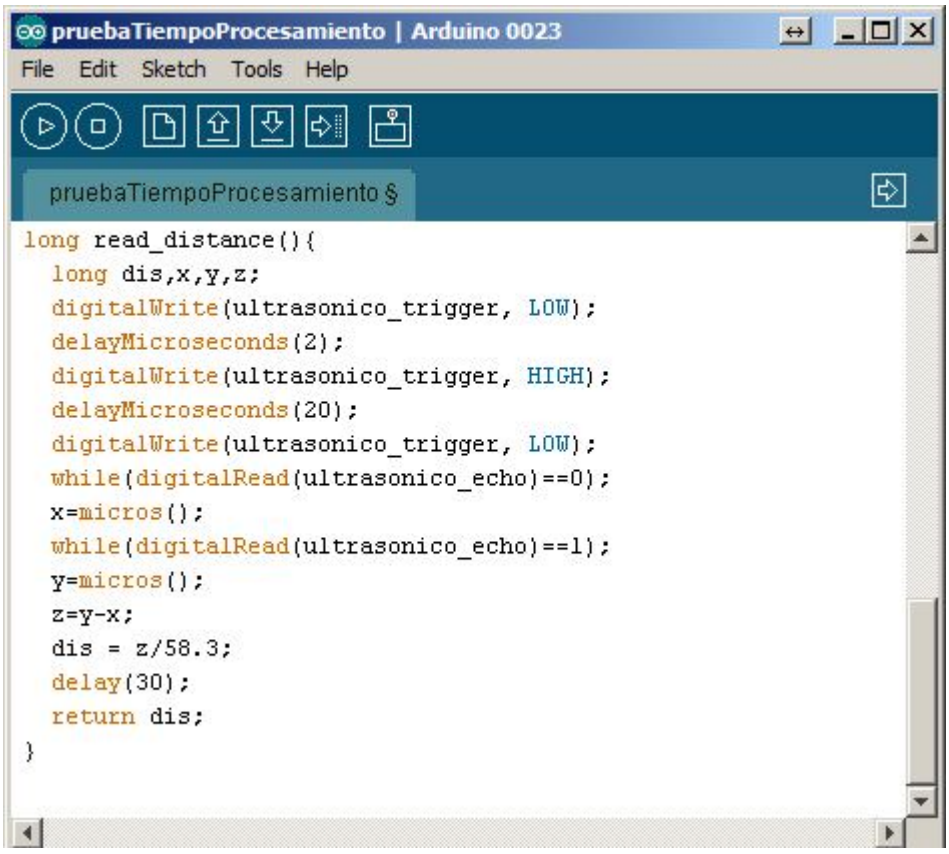

COM4
Send
Distancia[cm]: 8 Tiempo de Procesamiento: 31096
Distancia[cm]: 11 Tiempo de Procesamiento: 31312
Distancia[cm]: 10 Tiempo de Procesamiento: 31200
Distancia[cm]: 20 Tiempo de Procesamiento: 31808
Distancia[cm]: 223 Tiempo de Procesamiento: 43648
Distancia[cm]: 146 Tiempo de Procesamiento: 39128
Autoscroll
No line ending
9600 baud


COM4
Send
Distancia[cm]: 6 Tiempo de Procesamiento: 31032
Distancia[cm]: 8 Tiempo de Procesamiento: 31096
Distancia[cm]: 11 Tiempo de Procesamiento: 31312
Distancia[cm]: 10 Tiempo de Procesamiento: 31200
Distancia[cm]: 20 Tiempo de Procesamiento: 31808
Distancia[cm]: 223 Tiempo de Procesamiento: 43648
Autoscroll
No line ending
9600 baud

ANEXO F

Código de la prueba: Medición del tiempo de Procesamiento usando el microntrolador ATMega328p




pruebaTiempoProcesamiento | Arduino 0023
File Edit Sketch Tools Help
pruebaTiempoProcesamiento §
long read_distance(){
long dis,x,y,z;
digitalWrite(ultrasonico_trigger, LOW);
delayMicroseconds(2);
digitalWrite(ultrasonico_trigger, HIGH);
delayMicroseconds(20);
digitalWrite(ultrasonico_trigger, LOW);
while(digitalRead(ultrasonico_echo)==0);
x=micros();
while(digitalRead(ultrasonico_echo)==1);
y=micros();
z=y-x;
dis = z/58.3;
delay(30);
return dis;
}

ANEXO G

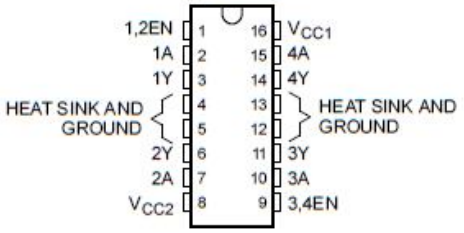
Hojas de datos de los integrados

L293, L293D
QUADRUPLE HALF-H DRIVERS

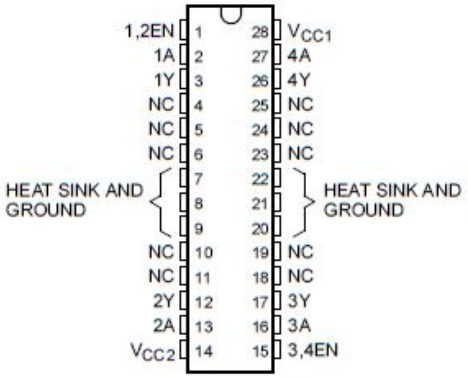
SLRS008B – SEPTEMBER 1986 – REVISED JUNE 2002

- **Featuring Unitrode L293 and L293D Products Now From Texas Instruments**
- **Wide Supply-Voltage Range: 4.5 V to 36 V**
- **Separate Input-Logic Supply**
- **Internal ESD Protection**
- **Thermal Shutdown**
- **High-Noise-Immunity Inputs**
- **Functional Replacements for SGS L293 and SGS L293D**
- **Output Current 1 A Per Channel (600 mA for L293D)**
- **Peak Output Current 2 A Per Channel (1.2 A for L293D)**
- **Output Clamp Diodes for Inductive Transient Suppression (L293D)**

N, NE PACKAGE
(TOP VIEW)



DWP PACKAGE
(TOP VIEW)



description

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

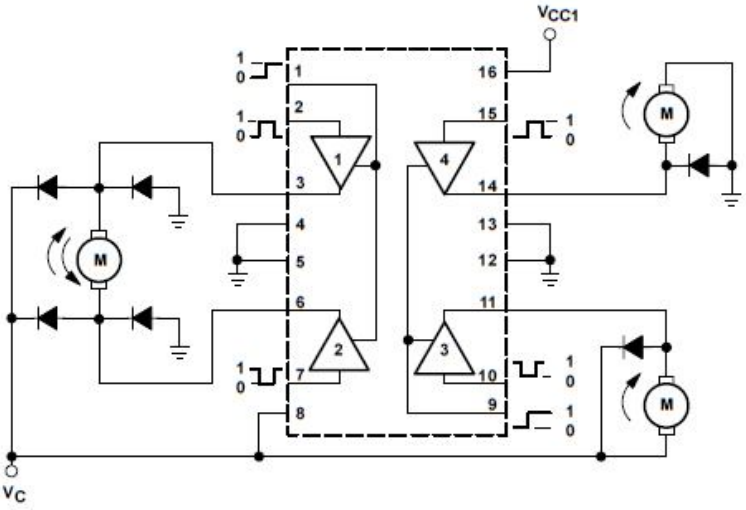
All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

On the L293, external high-speed output clamp diodes should be used for inductive transient suppression.

A V_{CC1} terminal, separate from V_{CC2}, is provided for the logic inputs to minimize device power dissipation.

The L293and L293D are characterized for operation from 0°C to 70°C.

block diagram



NOTE: Output diodes are internal in L293D.

**TEXAS INSTRUMENTS
AVAILABLE OPTIONS**

T_A	PACKAGE
	PLASTIC DIP (NE)
0°C to 70°C	L293NE L293DNE

**Unitrode Products
from Texas Instruments**

AVAILABLE OPTIONS

T_A	PACKAGED DEVICES	
	SMALL OUTLINE (DWP)	PLASTIC DIP (N)
0°C to 70°C	L293DWP L293DDWP	L293N L293DN

The DWP package is available taped and reeled. Add the suffix TR to device type (e.g., L293DWPTR).

recommended operating conditions

			MIN	MAX	UNIT
	Supply voltage	V_{CC1}	4.5	7	V
		V_{CC2}	V_{CC1}	36	
V_{IH}	High-level input voltage	$V_{CC1} \le 7$ V	2.3	V_{CC1}	V
		$V_{CC1} \ge 7$ V	2.3	7	V
V_{IL}	Low-level output voltage		–0.3†	1.5	V
T_A	Operating free-air temperature		0	70	°C

† The algebraic convention, in which the least positive (most negative) designated minimum, is used in this data sheet for logic voltage levels.

electrical characteristics, V_{CC1} = 5 V, V_{CC2} = 24 V, T_A = 25°C

PARAMETER			TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage		L293: I_{OH} = –1 A L293D: I_{OH} = –0.6 A		V_{CC2}–1.8	V_{CC2}–1.4		V
V_{OL}	Low-level output voltage		L293: I_{OL} = 1 A L293D: I_{OL} = 0.6 A			1.2	1.8	V
V_{OKH}	High-level output clamp voltage		L293D: I_{OK} = –0.6 A			V_{CC2} + 1.3		V
V_{OKL}	Low-level output clamp voltage		L293D: I_{OK} = 0.6 A			1.3		V
I_{IH}	High-level input current	A	V_I = 7 V			0.2	100	µA
		EN				0.2	10	
I_{IL}	Low-level input current	A	V_I = 0			–3	–10	µA
		EN				–2	–100	
I_{CC1}	Logic supply current		I_O = 0	All outputs at high level		13	22	mA
				All outputs at low level		35	60	
				All outputs at high impedance		8	24	
I_{CC2}	Output supply current		I_O = 0	All outputs at high level		14	24	mA
				All outputs at low level		2	6	
				All outputs at high impedance		2	4	

switching characteristics, V_{CC1} = 5 V, V_{CC2} = 24 V, T_A = 25°C

PARAMETER		TEST CONDITIONS	L293NE, L293DNE			UNIT
			MIN	TYP	MAX	
t_{PLH}	Propagation delay time, low-to-high-level output from A input	C_L = 30 pF, See Figure 1		800		ns
t_{PHL}	Propagation delay time, high-to-low-level output from A input			400		ns
t_{TLH}	Transition time, low-to-high-level output			300		ns
t_{THL}	Transition time, high-to-low-level output			300		ns

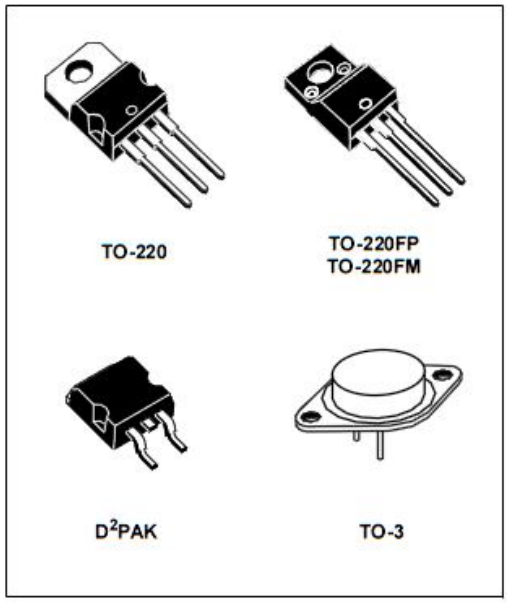
switching characteristics, V_{CC1} = 5 V, V_{CC2} = 24 V, T_A = 25°C

PARAMETER		TEST CONDITIONS	L293DWP, L293N L293DDWP, L293DN			UNIT
			MIN	TYP	MAX	
t_{PLH}	Propagation delay time, low-to-high-level output from A input	C_L = 30 pF, See Figure 1		750		ns
t_{PHL}	Propagation delay time, high-to-low-level output from A input			200		ns
t_{TLH}	Transition time, low-to-high-level output			100		ns
t_{THL}	Transition time, high-to-low-level output			350		ns

L7800 SERIES

POSITIVE VOLTAGE REGULATORS

- OUTPUT CURRENT TO 1.5A
- OUTPUT VOLTAGES OF 5; 5.2; 6; 8; 8.5; 9; 10; 12; 15; 18; 24V
- THERMAL OVERLOAD PROTECTION
- SHORT CIRCUIT PROTECTION
- OUTPUT TRANSITION SOA PROTECTION



DESCRIPTION

The L7800 series of three-terminal positive regulators is available in TO-220, TO-220FP, TO-220FM, TO-3 and D^2PAK packages and several fixed output voltages, making it useful in a wide range of applications. These regulators can provide local on-card regulation, eliminating the distribution problems associated with single point regulation. Each type employs internal current limiting, thermal shut-down and safe area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltage and currents.

Figure 1: Schematic Diagram

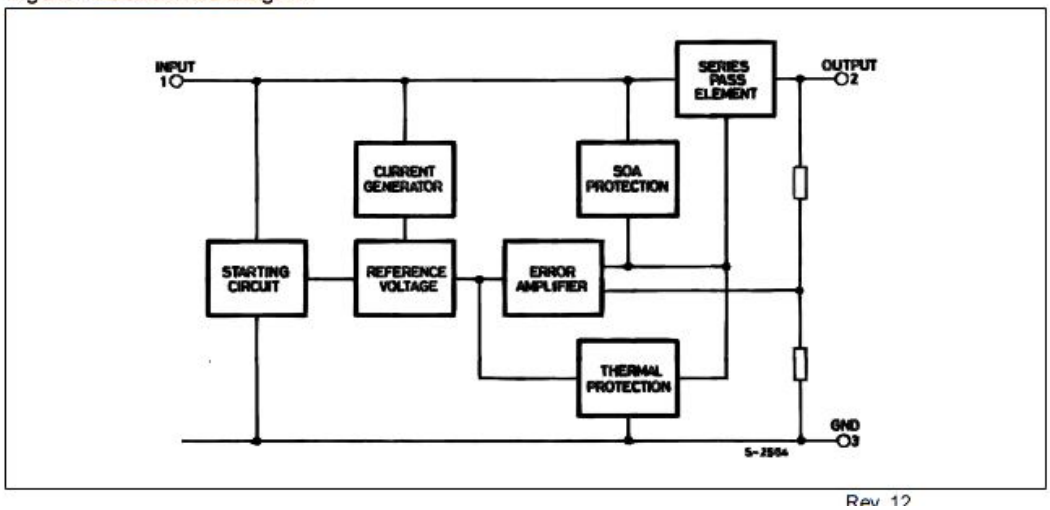


Table 4: Electrical Characteristics Of L7805 (refer to the test circuits, T_J = -55 to 150°C, V_I = 10V, I_O = 500 mA, C_I = 0.33 µF, C_O = 0.1 µF unless otherwise specified).

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_O	Output Voltage	T_J = 25°C	4.8	5	5.2	V
V_O	Output Voltage	I_O = 5 mA to 1 A $P_O \leq$ 15W V_I = 8 to 20 V	4.65	5	5.35	V
ΔV_O(*)	Line Regulation	V_I = 7 to 25 V T_J = 25°C		3	50	mV
		V_I = 8 to 12 V T_J = 25°C		1	25	
ΔV_O(*)	Load Regulation	I_O = 5 mA to 1.5 A T_J = 25°C			100	mV
		I_O = 250 to 750 mA T_J = 25°C			25	
I_d	Quiescent Current	T_J = 25°C			6	mA
ΔI_d	Quiescent Current Change	I_O = 5 mA to 1 A			0.5	mA
		V_I = 8 to 25 V			0.8	
$\Delta V_O/\Delta T$	Output Voltage Drift	I_O = 5 mA		0.6		mV/°C
eN	Output Noise Voltage	B =10Hz to 100KHz T_J = 25°C			40	$\mu V/V_O$
SVR	Supply Voltage Rejection	V_I = 8 to 18 V f = 120Hz	68			dB
V_d	Dropout Voltage	I_O = 1 A T_J = 25°C		2	2.5	V
R_O	Output Resistance	f = 1 KHz		17		mΩ
I_{sc}	Short Circuit Current	V_I = 35 V T_J = 25°C		0.75	1.2	A
I_{scp}	Short Circuit Peak Current	T_J = 25°C	1.3	2.2	3.3	A

(*) Load and line regulation are specified at constant junction temperature. Changes in V_O due to heating effects must be taken into account separately. Pulse testing with low duty cycle is used.