ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"CONTROL DE LOS MOVIMIENTOS DE UN ROBOT USANDO UN ACELERÓMETRO Y NIOS II"

TESINA DE SEMINARIO

Previa a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentada por:

Segundo Jeancarlos Bastidas Carrillo

Pablo Geovanny Palacios Játiva

GUAYAQUIL – ECUADOR

2013

AGRADECIMIENTO

Quiero agradecer a Dios, por darme esta familia la cual me ha apoyado en buenos y malos momentos: mi padre Ángel Palacios, mi madre Silvia Játiva, mi hermana Krhistin Palacios han sido fundamentales para la culminación de esta etapa de mi vida, su ayuda en todo aspecto ha sido la base de mis logros.

A toda mi familia que siempre me ha dado sus consejos para seguir adelante, a todas las personas, amigos que han formado parte de mi vida para bien, un agradecimiento especial al Ing. Ronald Ponguillo por habernos brindado sus conocimientos y tiempo para la culminación de nuestro proyecto

Pablo Palacios Játiva

Primero que nada agradezco a Dios que siempre está presente conmigo, agradezco a mi familia por el apoyo brindado y por los valores que me han inculcado, por su trabajo y esfuerzo diario para que yo cumpla mis objetivos, su apoyo a sido primordial, también agradezco a la Escuela Superior Politécnica del Litoral por acogerme en sus aulas, sus profesores, amigos y compañeros que conocí que de una u otra manera han contribuido con esta meta, de igual modo al Ing. Ronald Ponguillo por compartir sus conocimientos y llevarnos bien encaminados en el desarrollo de este proyecto

Segundo Jeancarlos Bastidas Carrillo

DEDICATORIA

Este logro profesional se lo dedico a mis padres mi familia, y todas esas personas que de una u otra manera han formado parte de mi vida sea que sigan en ella o no, su apoyo y consejos me han dado la fuerza para culminar este proyecto

Pablo Palacios Játiva

Dedicado a mis padres y hermanos que han contribuido conmigo a lo largo de mi vida universitaria, me han transmitido consejos y me han encaminado a la culminación de esta etapa.

Jeancarlos Bastidas Carrillo

TRIBUNAL DE SUSTENTACIÓN

Ing. Ronald Ponguillo

PROFESOR DEL SEMINARIO DE GRADUACIÓN

Ing. Víctor Asanza

PROFESOR DELEGADO POR LA UNIDAD ACADEMICA

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesina, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL". (Reglamento de exámenes y títulos profesionales de la ESPOL).

Pablo Geovanny Palacios Játiva

Segundo Jeancarlos Bastidas Carrillo

RESUMEN

El presente proyecto “CONTROL DE LOS MOVIMIENTOS DE UN ROBOT USANDO UN ACELEROMETRO Y NIOS II” es nuestro tema del Seminario de Graduación “PROCESADORES EMBEBIDOS CONFIGURABLES”.

Lo que se ha realizado en este proyecto mediante el uso de una FPGA que se encuentra en la tarjeta de desarrollo DE0-NANO es implementar un control remoto que permita dirigir los movimientos de un robot LEGO MINDSTORM NXT y demostrar que es posible reemplazar cantidades de circuitos digitales en un solo chip para implementar un sistema complejo de control en una sola tarjeta de desarrollo ahorrando tiempo y costos de implementación

Para la programación del software del control remoto y del robot LEGO se usó el entorno de desarrollo NIOS II SOFTWARE BUILD TOOLS que permite crear aplicaciones en lenguaje C y consta de las librerías necesarias para el proyecto, QUARTUS II con su herramienta SOPC BUILDER permite crear el sistema basado en el procesador NIOS II y

pasarlo a lenguaje VERILOG y el software LEGO NXT 2.0 PROGRAMMING permite crear la aplicación que va a ejecutar el robot.

Para establecer comunicación entre el control remoto y el robot se usa un módulo bluetooth RN-42 el cual se encuentra conectado a la tarjeta DE0-NANO mediante los pines del puerto GPIO1.

El proyecto está dividido en 4 capítulos que detallan el diseño y funcionamiento del sistema:

En el capítulo 1, se presenta el marco teórico, dividido en dos partes: primero se detalla el hardware que se usa para el proyecto, entre los que se especifican: la tarjeta DE0-NANO, la FPGA CICLONE IV, el ACELEROMETRO ADXL345, el modulo bluetooth RN-42 y el robot LEGO MINSTORM NXT. La segunda parte detalla el software usado para la programación, entre los que se encuentran el NIOS II SBT, QUARTUS II usado en conjunto con SOPC BUILDER y para la programación del robot el software LEGO NXT 2.0 PROGRAMMING.

En el capítulo 2, mostramos el diseño de nuestro proyecto dividido en dos partes: el diseño del hardware del sistema NIOS II y del robot LEGO, además mostramos el diseño del software que controla la FPGA y el robot LEGO con sus respectivos diagramas de flujo para su mejor entendimiento.

El capítulo 3, muestra la implementación de nuestro proyecto dando a conocer la programación de la FPGA (código fuente) para que realice el control de movimiento y también la programación del robot LEGO para que detecte las señales enviadas por la tarjeta DE0-NANO.

En el capítulo 4, realizamos las pruebas con los escenarios propuestos que son la distancia de recepción y el tiempo de respuesta del robot, realizamos las comparaciones con los datos del fabricante y analizamos los resultados.

Por último se ofrecen las conclusiones obtenidas de acuerdo al cumplimiento de los objetivos propuestos además las recomendaciones y trabajos futuros que se pueden realizar.

ÍNDICE GENERAL

ÌNDICE DE FIGURAS

INDICE DE TABLAS

ABREVIATURAS

ASIC	Application-Specific Integrated Circuit
CPLD	Complex Programmable Logic Device
DSP	Digital Signal Processor
EEPROM	Electrically Erasable Programmable Read-Only Memory
EPCS	Erasable Programmable Configurable Serial
FHSS	Frequency Hopping Spread Spectrum
FPGA	Field Programmable Gate Array
GFSK	Gaussian Frequency-Shift Keying
GSM	Global System for Mobile Communications
HAL	Hardware Abstraction Layer
HDL	Hardware Description Language
I2C	Inter Integrated Circuit

JTAG	Joint Test Action Group
MIT	Massachusetts Institute of Technology
PCB	Printed Circuit Board
PDA	Personal Digital Assistant
PLD	Programmable Logic Device
PLL	Phase Lock Loop
RPM	Revoluciones Por Minuto
RS-232	Recommended Standard 232
SDRAM	Synchronous Dynamic Random-Access Memory
SMT	Surface-mount technology
SOPC	System on a Programmable Chip
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver-Transmitter
VGA	Video Graphics Array
VHDL	VHSIC Hardware Description Language

INTRODUCCIÓN

La tecnología basada en FPGA continúa siendo impulsada año tras año debido a las distintas aplicaciones que en ella se encuentra no solo en el ámbito de los sistemas electrónicos digitales, sino en sistemas de comunicaciones, sistemas de seguridad y sistemas de control, todo esto con el objetivo de minimizar costos, hacer más fácil y rápido las implementaciones y miniaturizar los dispositivos a usarse.

Dentro de todas estas aplicaciones antes mencionadas se encuentra la de control, la cual, mediante el uso de una FPGA y la combinación de múltiples dispositivos, todos estos formando un solo sistema programable podemos llegar a controlar movimientos de cámaras, de sensores e incluso de sistemas robóticos.

En nuestro proyecto usamos la tecnología de la FPGA embebida en una tarjeta de desarrollo DE0 nano que cuenta con dispositivos muy útiles tales como un acelerómetro con el cual podemos controlar los movimientos de un robot NXT de LEGO a nuestra voluntad mediante comunicación bluetooth entre el robot y la tarjeta, demostrando una aplicación bastante útil en nuestros días.

OBJETIVOS

JUSTIFICACIÓN

Este proyecto tiene sus bases en todo lo practicado y aprendido en la carrera universitaria y en el Seminario de Graduación “PROCESADORES EMBEBIDOS CONFIGURABLES”. Se escogió este proyecto porque presentó un reto para nosotros en la parte de investigación de temas como protocolos de comunicación, programación de robots, manejo de acelerómetro entre otros, además para la verificación de todo lo aprendido de manera teórica, en la práctica e implementación.

OBJETIVOS GENERALES

Implementar mediante el uso de la tarjeta de desarrollo DE0-NANO y su acelerómetro, un sistema capaz de controlar los movimientos de un robot MINDSTORM NXT de LEGO mediante comunicación bluetooth.

OBJETIVOS ESPECÍFICOS

- Uso, manejo y comprensión de la tarjeta de desarrollo DE0-NANO.
- Conocer el manejo del acelerómetro ADXL345 embebido en la tarjeta DE0-NANO
- Diseñar el sistema de comunicación entre el modulo bluetooth RN-42 y la tarjeta DE0-NANO
- Diseñar el sistema de comunicación entre la tarjeta DE0-NANO y el robot NXT mediante bluetooth

ALCANCE Y LIMITACIONES DEL PROYECTO

Entre los alcances del proyecto se tiene:

- Procesar la información del acelerómetro, para poder controlar el movimiento del robot en base a esa información.
- Comunicación serial (RS232) entre la FPGA y el módulo bluetooth.
- Comunicación SPI entre el acelerómetro y la FPGA para el envío y recepción de datos.
- Decodificación de los datos recibidos por el robot NXT de LEGO
- Mostrar en la pantalla del robot NXT de LEGO los detalles del movimiento del robot.

Entre las limitaciones se tienes las siguientes:

- Debido a las características mecánicas de los motores del robot NXT de LEGO no se puede lograr alcanzar velocidades mayores a 170 rpm, además influye en la precisión del movimiento del robot.
- El robot NXT de LEGO cuenta con un módulo bluetooth clase B el cual limita el alcance a solo 20 metros.

CAPÍTULO I

1. MARCO TEORICO

En este capítulo se abarca los conceptos básicos y se hace una pequeña introducción a los programas y tecnologías que se usaron para la realización del proyecto.

1.1 HARDWARE

1.1.1 TARJETA DE DESARROLLO DE0-NANO

La tarjeta DE0-NANO es una tarjeta de desarrollo educacional diseñada por la empresa Terasic que permite el desarrollo de una amplia gama de proyectos basados en una plataforma FPGA. Las ventajas de la tarjeta incluyen su tamaño y peso, así como la capacidad de ser reconfigurado sin la necesidad de realizar cambios en el hardware.

Además cuenta con varias interfaces integradas incluyendo 2 cabeceras de expansión GPIO de 40 pines que permiten agregar nuevo hardware a la tarjeta, convertidor A/D, acelerómetro, leds, botones y memoria SDRAM y EEPROM para almacenamiento de datos y “frame buffering”.

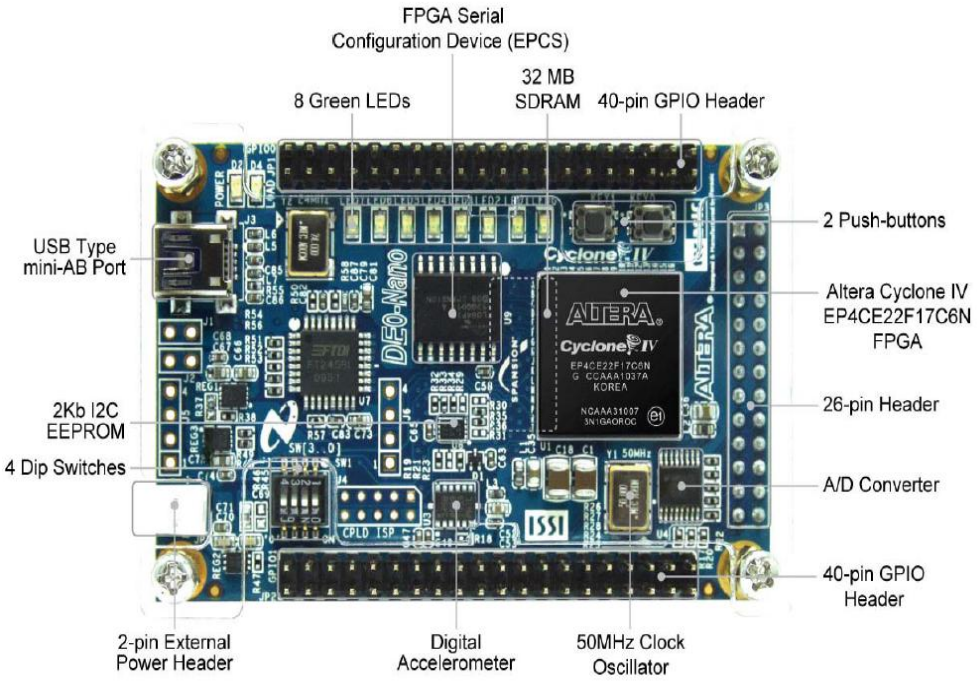


Figura 1- 1 Tarjeta DE0 NANO

La tarjeta DE0-NANO se basa en la Familia Lógica Programable FPGA Cyclone IV, chip desarrollado por Altera que tiene 153 pines de I/O en donde están conectados todos los dispositivos que tiene embebida la tarjeta, permitiendo al usuario controlar todos los componentes usando un sistema NIOS II almacenado en la FPGA Cyclone IV.

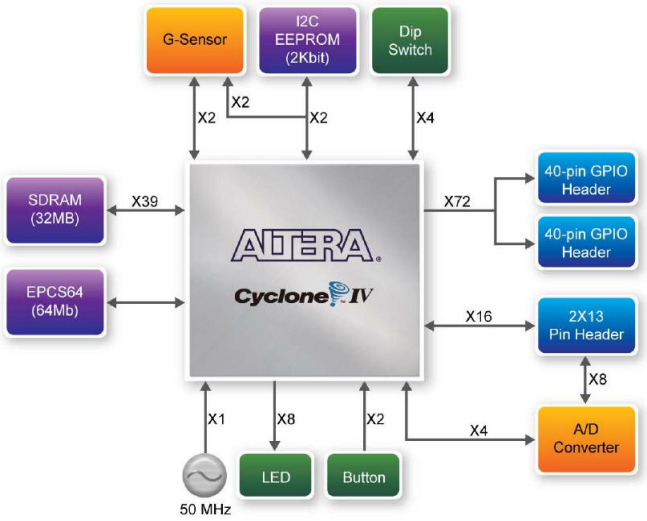


Figura 1- 2 Diagrama de bloques de la Tarjeta DEO-NANO

La tarjeta DE0-NANO tiene las siguientes características:

- FPGA Altera Cyclone IV EP4CE22F17C6N con 153 pines I/O
- Circuito USB-Blaster para configurar la FPGA
- Dispositivo de configuración serial EPCS16 de 16Mbits
- 2 cabeceras de expansión de 40 pines

- 1 cabecera de expansión de 26 pines
- Memoria SDRAM de 32MB
- EEPROM de 2Kbits con comunicación I2C
- 8 LEDs verdes
- 2 Pulsadores
- 4 interruptores DIP
- Acelerómetro de 3 ejes con una resolución de 13 bits
- Convertidor A/D de 8 canales y 12 bits de resolución
- Reloj de 50MHz
- Puerto USB tipo mini-AB (5V)
- Conector de 2 pines para fuente de poder externa 3.6v - 5.7V

1.1.2 FPGA CYCLONE IV

Una FPGA (Field Programmable Gate Array) es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada 'in situ' mediante un lenguaje de descripción especializado. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinatorial hasta complejos sistemas en un chip.

Figura 1- 3 FPGA CYCLONE IV

Una jerarquía de interconexiones programables permite a los bloques lógicos de una FPGA, ser interconectados según la necesidad del diseñador del sistema, algo parecido a un breadboard programable. Estos bloques lógicos e interconexiones pueden ser programados después del proceso de manufactura por el usuario/diseñador, así que el FPGA puede desempeñar cualquier función lógica necesaria.

Las FPGAs tienen la ventaja de ser reprogramables, lo que añade una flexibilidad al flujo de diseño, sus costes de desarrollo y adquisición son mucho menores para pequeñas cantidades de dispositivos y el tiempo de desarrollo es también menor.

El diseñador cuenta con la ayuda de entornos de desarrollo especializados en el diseño de sistemas digitales en FPGA. Un diseño puede ser capturado ya sea como esquemático, o haciendo uso de un

lenguaje de programación especial. Estos lenguajes de programación especiales son conocidos como HDL (Hardware Description Language). Los HDLs más utilizados son:

- VHDL
- Verilog
- ABEL

La tarjeta DE0-NANO contiene un FPGA Cyclone IV EP4CE22, que se puede programar mediante la interface JTAG. Esto permite a los usuarios configurar la FPGA con un diseño específico utilizando el software QUARTUS II. El diseño programado seguirá siendo funcional en la FPGA, siempre y cuando la placa esté encendida, caso contrario la información de configuración se perderá. Para evitar este inconveniente la tarjeta DE0-NANO cuenta con un dispositivo de configuración serial EPCS16 que permite almacenar permanentemente la configuración de FPGA. Este chip nos va ha permitir cargar la configuración del diseño de la computadora automáticamente dentro de la FPGA cada vez que la tarjeta sea energizada.

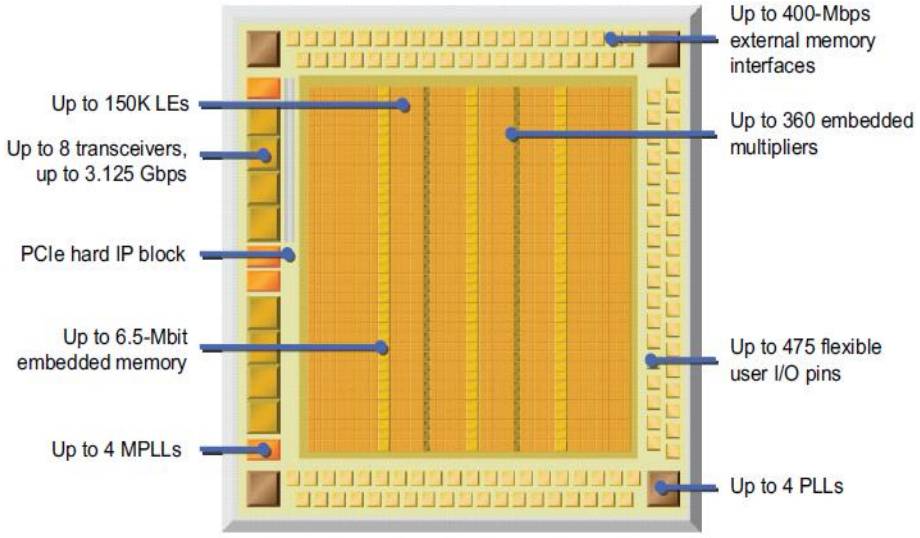


Figura 1- 4 Arquitectura de la FPGA Cyclone IV

El núcleo de la FPGA Cyclone IV consiste de elementos lógicos compuestos de 4 entradas de tabla de consultas LUTs, bloques de memoria M9K y multiplicadores. Cada bloque de memoria M9K ofrece 9Kbits de memoria SRAM embebida y es posible configurar cada bloque como un “single port”, “simple dual port” o “true dual port RAM”, así como también buffer FIFO o ROM. Los multiplicadores pueden implementarse como un único bloque de 18x18 o como 2 bloques de 9x9 multiplicadores.

Además incluye una red de hasta 20 clock globales (GCLK), soporta hasta 4 PLLs con 4 salidas por PLL y ofrecer soporte para memorias SDR, DDR, DDR2 SDRAM y QDRII SRAM. Los datos de configuración

de la FPGA Cyclone IV se almacenan en celdas de memoria SRAM cada vez que el dispositivo es energizado.

La FPGA Cyclone IV EP4CE22 ofrece las siguientes características:

- Voltaje del núcleo 1.0v y 1.2v
- 22,320 elementos lógicos
- 594Kbits de memoria embebida
- 66 multiplicadores embebidos de 18x18
- 4 PLLs de propósito general
- Red global de 20 clock
- 8 bancos de I/O
- 153 pines I/O
- Soporte para DDR2 SDRAM de hasta 200MHz

1.1.3 ACELEROMETRO ADXL345

El ADXL345 es un acelerómetro de 3 ejes, pequeño, delgado, de baja potencia y con alta resolución (13 bits) que permite realizar mediciones en un rango de ±2g, ±4g, ±8g y ±16g. Los datos digitales están en formato complemento a dos de 16bits a través de la interfaz I2C o SPI de 3 o 4 cables.

El ADXL345 se adapta muy bien para medir aceleración estática en aplicaciones de censado de inclinación, así como aceleraciones dinámicas resultantes de movimiento o impactos. Su alta resolución 4mg/LSB permite mediciones de inclinación con cambios menores a 1.0°.

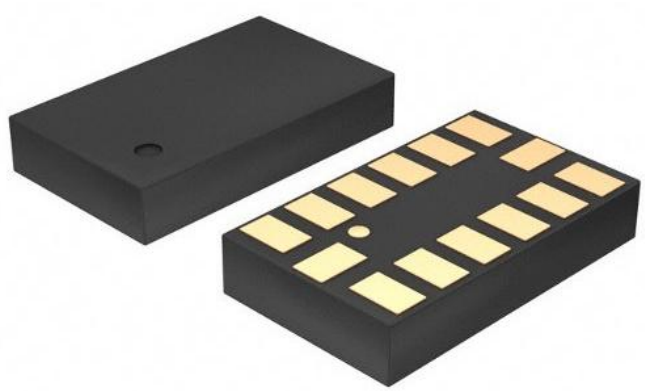


Figura 1- 5 Acelerómetro ADXL345

Así mismo posee varias funciones especiales como censado de actividad o inactividad para detectar la presencia o ausencia de movimiento. También se puede detectar si la aceleración excede un umbral determinado por el usuario en alguno de los ejes. En caída libre este sensor detecta si el dispositivo se está cayendo.

Características:

- Voltaje de alimentación 2.0 - 3.6VDC

- Consumo de ultra baja potencia: 40uA en medición y 0.1µA en standby
- Detección de caída libre
- 8 funciones de interrupción
- Interfaz SPI y I2C

La tarjeta DE0-NANO tiene embebido un acelerómetro ADXL345 que está conectado directamente a la FPGA Cyclone IV. La comunicación entre la FPGA y el acelerómetro se realiza a través de la interfaz SPI. La figura muestra la conexión entre el acelerómetro y la FPGA.

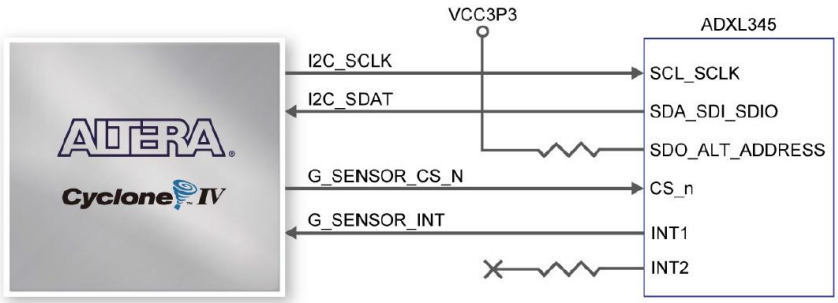


Figura 1- 6 conexión entre el ADXL345 y la FPGA CYCLONE IV

El sensor es una superficie de polisilicio-micromecanizado, estructura construida en la parte superior de una oblea de silicio. Muelles de polisilicio suspenden la estructura sobre la superficie de la oblea y proporcionan una resistencia contra las fuerzas debidas a la aceleración aplicada. La deflexión de la estructura se mide utilizando condensadores diferenciales independientes que constan de placas

fijas y placas fijadas a la masa en movimiento. La aceleración desvía la masa de prueba y desequilibra el condensador diferencial, lo que resulta en una salida del sensor cuya amplitud es proporcional a la aceleración.

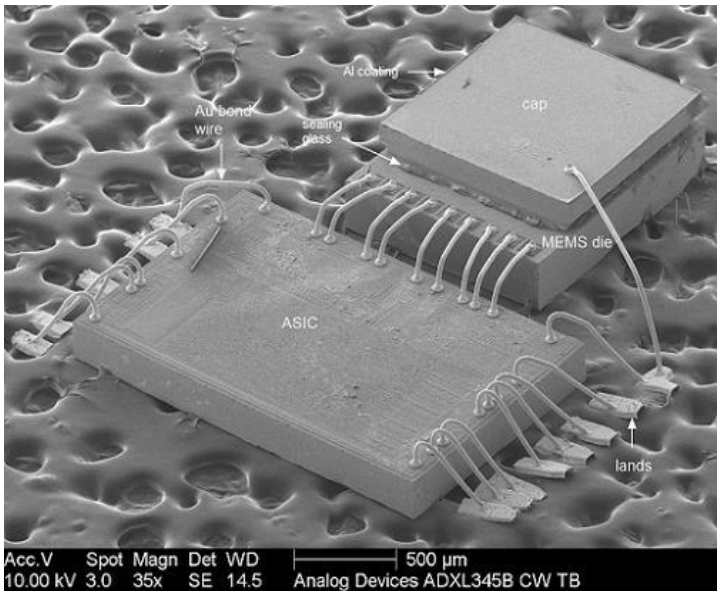


Figura 1- 7 Vista interna a 500 µm del ADXL345

El ADXL345 cuenta con dos pines de interrupciones programables INT1 e INT2 con un total de ocho funciones de interrupción disponible. Cada interrupción puede ser activada o desactivada de forma independiente, con la opción de mapear ya sea el pin INT1 o INT2. Todas las funciones se pueden utilizar simultáneamente la única limitante es que algunas funciones pueden necesitar compartir pines de interrupción.

Dependiendo de la inclinación del acelerómetro obtenemos mediciones en los distintos ejes que van en un rango de ±2g, ±4g, ±8g y ±16g rangos que son configurados a nivel de software dependiendo del tipo de medición que se quiere realizar.

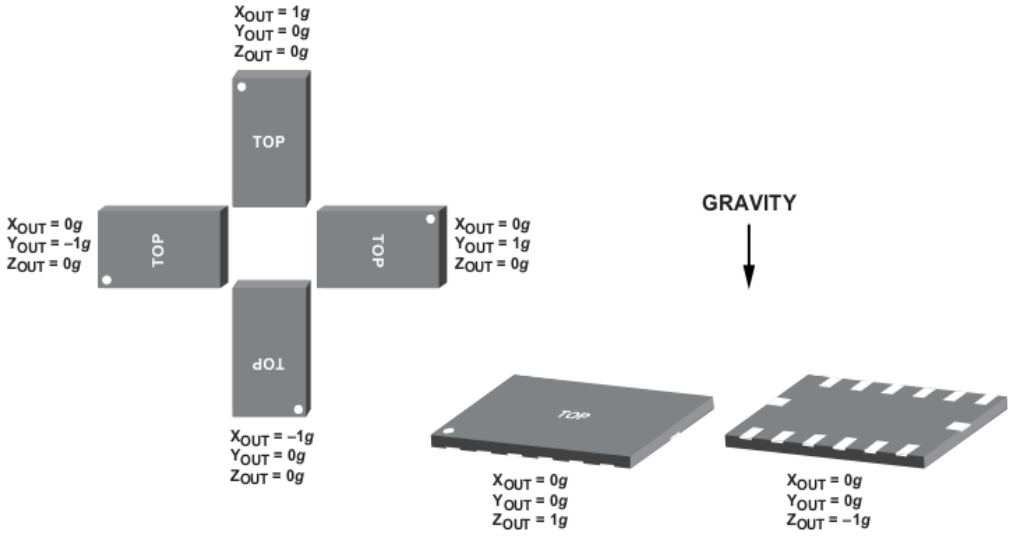


Figura 1- 8 inclinación vs orientación de la gravedad del Acelerómetro ADXL345

1.1.4 COMUNICACIÓN UART RS-232

El corazón del sistema de comunicaciones serie es la UART, “Universal Asynchronous Receiver/Transmitter”. Es un controlador cuya misión principal es convertir los datos recibidos del bus del PC o en este caso los datos recibidos del bus Avalon de la DE0-NANO en

formato paralelo, a un formato serie que se utiliza en la transmisión hacia el exterior.

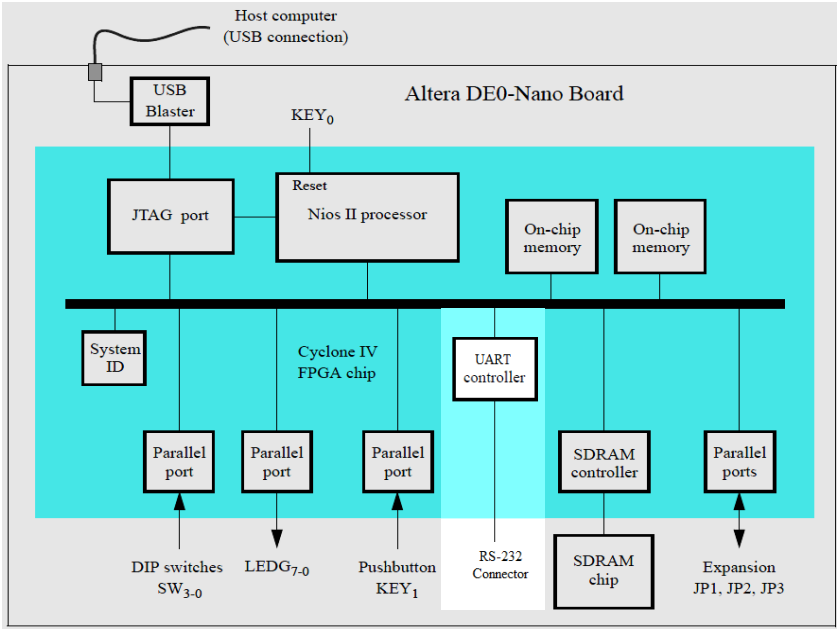


Figura 1- 9 controlador UART en un sistema NIOS II

El controlador UART implementa el protocolo RS-232 y es un dispositivo configurable en el que pueden establecerse las condiciones que se utilizarán para la transmisión (velocidad, paridad, longitud y bits de parada). Además de las líneas de transmisión TX y recepción RX, las comunicaciones seriales poseen otras líneas de control de flujo RTS/CTS (Hand-shake), donde su uso es opcional dependiendo del dispositivo a conectar.

El driver provee un simple mapa de registro Avalon Memory-Mapped ESCLAVO que permite al procesador NIOS II comunicarse con él a

través del bus Avalon Memory-Mapped MASTER para poder leer y escribir registros de control y datos.

A nivel de software, la configuración principal que se debe dar a una conexión a través de puertos seriales RS-232 es básicamente la selección de la velocidad en baudios (1200, 2400, 4800, 9200, etc.), la verificación de datos o paridad (paridad par o paridad impar o sin paridad), los bits de parada luego de cada dato (1 ó 2), y la cantidad de bits por dato (7 u 8), que se utiliza para cada símbolo o caracter enviado. Toda esta configuración se la realiza desde el software SOPC BUILDER durante la implementación de los componentes que conforman la FPGA.

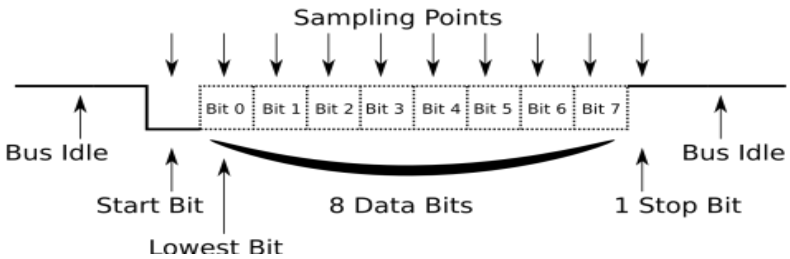


Figura 1- 10 trama de la comunicación UART

El UART toma bytes de datos y transmite los bits individuales de forma secuencial. En el destino, un segundo UART re-ensambla los bits en bytes completos. La transmisión serie de la información digital (bits) a través de un cable único u otros medios es mucho más efectiva en

cuanto a costo que la transmisión en paralelo a través de múltiples cables. Se utiliza un UART para convertir la información transmitida entre su forma secuencial y paralela en cada terminal de enlace. Cada UART contiene un registro de desplazamiento que es el método fundamental de conversión entre las formas serie y paralelo.

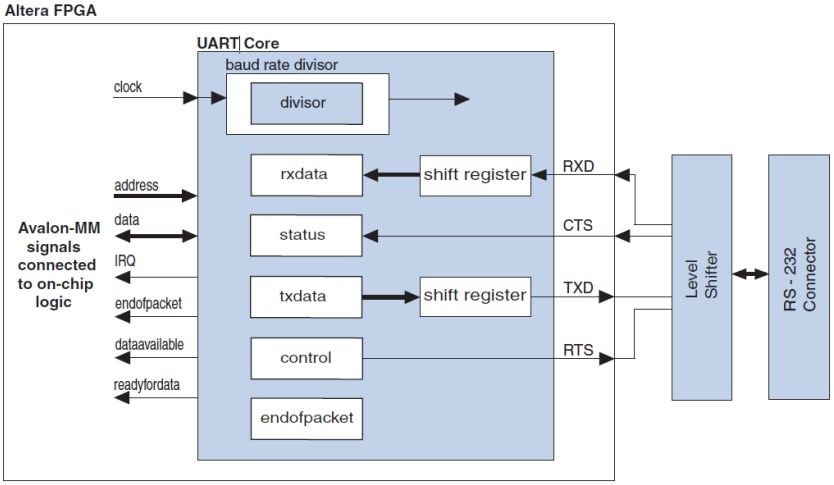


Figura 1- 11 diagrama de bloques del CORE UART de SOPC BUILDER

Para nuestro proyecto usamos el controlador UART RS-232 que se encuentra disponible en las librerías de SOPC BUILDER listo para ser integrado en la FPGA de la tarjeta DE0-NANO.

La tarjeta de trabajo DE0-NANO no cuenta con un conector RS-232 tipo DB-9 pero utilizaremos 3 pines del puerto de expansión GPIO-1 para comunicar la tarjeta DE0-NANO y el modulo Bluetooth. Consulte la Tabla 1-1.

1.1.5 MODULO BLUETOOTH RN-42

El Bluetooth es una tecnología orientada a la conectividad inalámbrica entre distintos dispositivos como PCs, PDAs, teléfonos móviles, electrodomésticos, equipos de sonido, etc. El Bluetooth, aparte de ser una nueva tecnología, es también una especificación abierta para comunicaciones inalámbricas de voz y datos.

Está basado en un enlace de radio de bajo coste y corto alcance, el cual proporciona conexiones instantáneas (ad-hoc) tanto para entornos de comunicaciones móviles como estáticos. La principal ventaja que ofrece esta tecnología es la conectividad sin cables de todos los dispositivos, pero más que reemplazar los incómodos cables, esta tecnología ofrece un puente entre las redes de datos hoy existentes y el exterior.

El Bluetooth, al ser un estándar abierto, pretende conectar una amplia gama de dispositivos sin importar su marca. Sus principales características son:

- Robustez
- Bajo coste
- Necesidad de poca potencia
- Baja complejidad

- Es un estándar global

El modulo Bluetooth RN-42 de la compañía Roving Networks es un dispositivo de clase 2 que provee un rango de alcance entre 10 y 20 metros con un bajo consumo de energía. Es perfecto para aplicaciones de corto alcance alimentado mediante batería. Usa solamente 26µA en modo "sleep" mientras permanece aún conectado y reconocible. Soporta múltiples perfiles Bluetooth tales como SPP y HCI y provee una interfaz de comunicación UART lo cual facilita su integración simple con sistemas embebidos o para su conexión con dispositivos existentes.

Figura 1- 12 Módulo Bluetooth RN-42

Las Características más importantes son:

- Módulo Bluetooth Clase II v2.1 + módulo EDR
- Soporta Módulo Bluetooth 2.1/2.0/1.2/1.1

- Interfaces de conexión de datos UART (SPP o HCI) y USB (sólo HCI)
- Soporta velocidades de datos en modo SPP - 240Kbps slave, 300Kbps master
- Soporta velocidades de datos en modo HCI - 1.5Mbps slave, 3.0Mbps master
- Antena tipo chip
- Alcance: hasta 20m con línea de vista.
- Modulación: FHSS/GFSK (79 canales a intervalos de 1MHz)
- Comunicación segura, encriptación de 128 bits
- Potencia de salida: 4dBm
- Sensitividad: -80dBm
- Consumo de corriente en transmisión/recepción: 25mA/25mA
- Voltaje de alimentación: 3V ~ 3.6V
- Tamaño: 13.4mm x 25.8mm
- Tipo de montaje: SMT

Antes de poner en funcionamiento el módulo bluetooth es necesario realizar una configuración de varios parámetros que garantizarán el correcto funcionamiento en el sistema que se esta implementando.

Este módulo tiene 2 modos de funcionamiento:

- **Data Mode:** Es el modo de transmisión en el que todos los comandos son ignorados.
- **Command Mode:** Es el modo de configuración en el que determinados comandos pueden configurar ciertos parámetros de funcionamiento del módulo. Dentro de este modo solo tenemos 60 segundos para realizar la configuración, pasado este tiempo el módulo regresa a Data Mode e ignorara los comandos.

El módulo se lo puede conectar por RS-232 con los acoples respectivos como MAX232 y el divisor de voltaje a un puerto serie por medio de un DB9, se debe abrir el hyperterminal o cualquier programa que permita leer y enviar comandos AT.

La segunda opción es prender el bluetooth de un computador o laptop y por medio de algún programa que controle bluetooth crear un COM virtual que le permita al hyperterminal enviar y recibir datos de forma inalámbrica (de esta manera se conecta TX y RX).

El método de configuración que usamos será a través del puerto RS-232 del PC y usamos el software AccessPort (similar a Hyperterminal); para este método necesitamos crear un circuito para poder comunicar

la PC con el módulo. Cuando se lo conecta viene calibrado de fábrica a una velocidad de 115200bps con nombre de fábrica según el serial y en modo esclavo (este modo permite ser descubierto y conectado por otros dispositivos bluetooth y también permite iniciar comunicaciones).

Para nuestros propósitos nos toca cambiarle el modo de operación de esclavo a master. En modo master el dispositivo no será descubierto por otros dispositivos bluetooth y solo permitirá iniciar comunicaciones, también cambiaremos el nombre del dispositivo y usaremos el baud rate que viene por defecto.

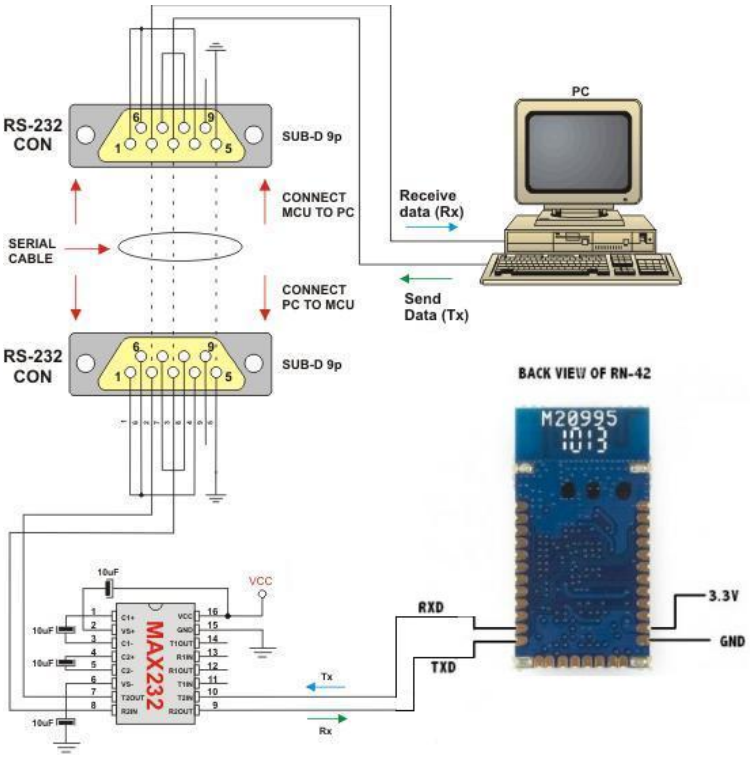


Figura 1- 13 conexión con el PC para la configuración del Módulo Bluetooth RN-42

1. Una vez realizado el circuito abrimos el AccessPort y seleccionamos el número de puerto COM y elegimos baud rate 115200bps.
2. Para entrar en el modo de configuración del módulo se debe enviar 3 signos de dólar $$$.
3. Tenemos 60 segundos para enviar los comandos al dispositivo, ya que si no entra en Data mode y los comandos son ignorados. Si los datos fueron leídos, el modulo responderá con CMD y después del envío de instrucciones nos regresa un AOK. La forma de ver si el modulo está bien, es mirar el led de status, siempre debe estar parpadeando, después de entrar en Command mode la oscilación del led es más rápida y cuando esta enlazado con algún dispositivo el led deja de parpadear y también se enciende el led de estado conectado.
4. Ingrese los comandos 1-4 que se encuentran en la Tabla 1-1 para configurar el módulo bluetooth de acuerdo los requerimientos de nuestro proyecto.

	COMANDO	DESCRIPCION
1	$$$	Modo comando
2	SM,1	Modo master
3	SN,DE0-NANO	Cambiar nombre
4	R,1	Forzar reinicio
5	C	Iniciar conexión
6	K	Terminar conexión
7	H	Muestra lista de comandos
8	D	Muestra configuración básica
9	E	Muestra configuración avanzada

Tabla 1- 1 comandos del módulo bluetooth

La Tabla 1-2 muestra la distribución de pines del módulo bluetooth. Para nuestro proyecto solo usamos 7 pines y construimos un PCB en donde soldamos el módulo SMT y lo acoplamos al puerto GPIO1 de la tarjeta DE0-NANO. Solo detallamos los pines que se utilizan en el proyecto, la información de los otros pines podrá ser consultada en el datasheet.

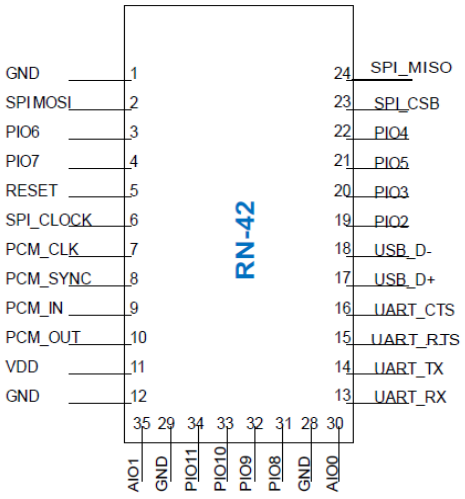


Figura 1- 14 distribución de pines del módulo bluetooth

PIN	NOMBRE	DESCRIPCION
1	GND	Señal de tierra
5	RESET	Reset. Activo en bajo
11	VDD	Señal de voltaje 3.3v
19	PIO2	Estado. Alto cuando está conectado
21	PIO5	Estado. Cambia según el estado. Bajo conectado
13	UART_RX	Entrada de recepción UART
14	UART_TX	Salida de transmisión UART

Tabla 1- 2 pines del módulo bluetooth usados en el proyecto

1.1.6 ROBOT LEGO MINDSTORM NXT

Lego Mindstorms es una herramienta educacional de robótica fabricado en conjunto por la empresa Lego y el MIT, el cual posee elementos básicos de las teorías robóticas, como la unión de piezas y la programación de acciones en forma interactiva. Este robot fue comercializado por primera vez en septiembre de 1998. Puede ser usado para construir un modelo de sistema integrado con partes electromecánicas controladas por computador. Prácticamente todo puede ser representado con las piezas tal como en la vida real, como un elevador o robots industriales.

Figura 1- 15 ROBOT LEGO MINDSTORM NXT

Para nuestro proyecto usamos el robot LEGO MINDSTORM NXT ya que cuenta con varias características que se requieren para

implementar el proyecto como bluetooth incorporado lo cual facilita la comunicación con la tarjeta DE0-NANO, servos motores que permiten controlar el movimiento del robot y una fácil programación gráfica basada en LabView.

El bloque de NXT puede comunicarse con el computador mediante la interfaz de USB que posee, además para comunicarse con otros robots en las cercanías usa una interfaz Bluetooth que es compatible con la Clase II v2.0.

Esta conectividad con Bluetooth no sólo permite conectarse con otros bloques, sino también con computadores, palms, teléfonos móviles y otros dispositivos con esta interfaz de comunicación.

La programación del Lego Mindstorms NXT se realiza mediante el entorno de programación gráfico llamado NXT-G el cual usa lenguaje gráfico basado en LabView de National Instrument, también es posible programarlo en RoboLab y directamente en LabView.

El bloque NXT cuenta con 4 botones que nos permite movernos a través del menú de forma fácil y rápida. Posee una amigable interfaz gráfica donde el usuario puede elegir las distintas opciones que presenta el menú. En la parte superior de la pantalla del NXT, podemos ver el tipo de conexión que estamos usando (Bluetooth y/o

USB), el nombre de nuestro robot, el símbolo que indica que está en operación y finalmente el estado de la batería.

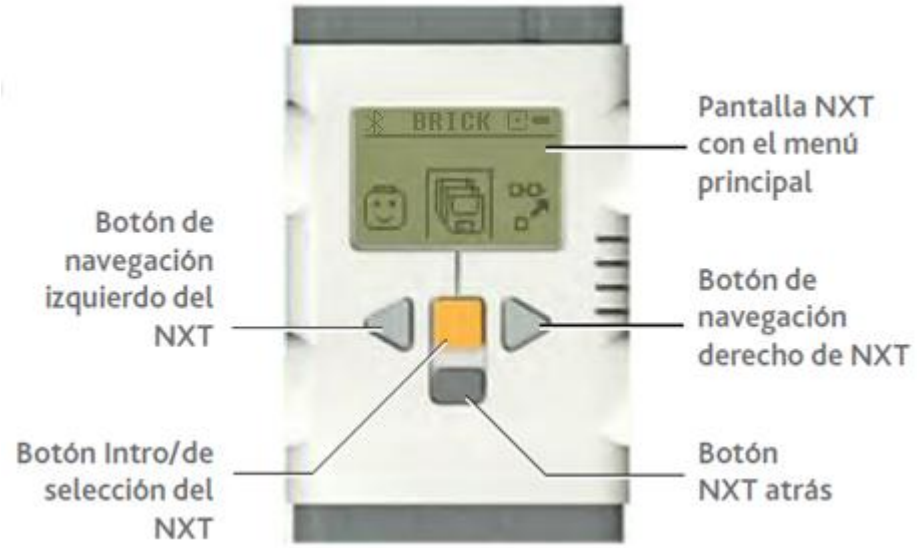


Figura 1- 16 Bloque NXT

SIMBOLO	DESCRIPCION
ᛒ	Bluetooth encendido
ᛒ<	Bluetooth visible para otros dispositivos
ᛒ◇	Bluetooth conectado a otro dispositivo
USB	USB conectado y trabajando bien
⊣⊢	USB conectado, con problemas
⊡	NXT operando correctamente
▭	Batería baja
▬	Batería bien
BRICK	Nombre del NXT

Tabla 1- 3 simbología de la interfaz de usuario del bloque NXT

Figura 1- 17 opciones del menú del ROBOT LEGO NXT

Características técnicas:

- Procesador Principal: Microprocesador Atmel ARM7 de 32 bits 48Mhz
- Memoria FLASH de 256 Kbytes, 64 Kbytes de Memoria RAM
- Co-procesador: Microprocesador Atmel AVR de 8 bits 8Mhz
- Memoria FLASH de 4Kbytes, 512 Bytes de RAM
- Comunicación Inalámbrica Bluetooth Clase II v2.0
- Puerta de alta velocidad USB (12 Mbit/s)
- Cuatro puertos de entrada de seis contactos
- Tres puertos de salida de seis contactos
- Pantalla LCD de 100x64 pixeles
- Parlante, calidad de sonido 8KHz
- Fuente de poder: Batería de Litium recargable o 6 baterías AA.

1.2 SOFTWARE

1.2.1 QUARTUS II

Quartus II es una herramienta de software producida por Altera para el análisis y la síntesis de diseños realizados en HDL que permite al diseñador compilar sus diseños de circuitos lógicos, realizar análisis de tiempo, examinar diagramas RTL, simular la reacción de un diseño a diferentes estímulos y configurar el dispositivo de destino con el programador JTAG. Para la realización de este proyecto usamos la versión 12.1 lanzada al mercado en Julio del 2012.

Se trata de un entorno completo para el diseño de SOPC “System-On-a- Programable-Chip” que incluye soluciones para todas las fases de diseño de FPGA y CPLD.

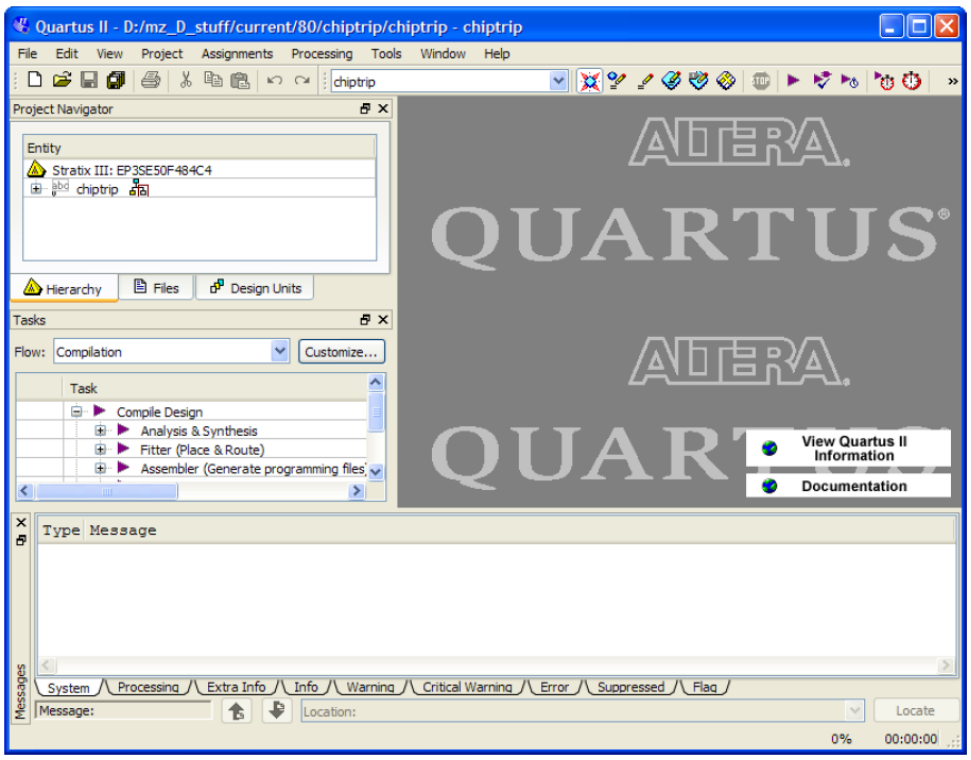


Figura 1- 18 interfaz de usuario del QUARTUS II

Quartus II utiliza una implementación de lenguaje VHDL o Verilog para realizar la descripción de hardware. Ambos lenguajes definidos por la IEEE son usados por ingenieros para describir circuitos digitales. Aunque pueden ser usados de forma general para describir cualquier circuito, se usan principalmente para programar PLD (Programable Logic Device - Dispositivo Lógico Programable), FPGA (Field Programmable Gate Array), ASIC y similares.

Estos lenguajes están diseñados para cubrir una serie de necesidades en el proceso de diseño ya que nos permite la descripción de la estructura de un sistema, es decir, la forma en que se descompone en subsistemas y como estos subsistemas están interconectados. Además, permite la especificación de la función de un sistema mediante las conocidas formas de lenguaje de programación, permitiendo el diseño de un sistema para ser simulado antes de ser fabricado, por lo que los diseñadores pueden rápidamente comparar alternativas poniendo a prueba la corrección sin demora y comparando gastos en la creación de un prototipo de hardware.

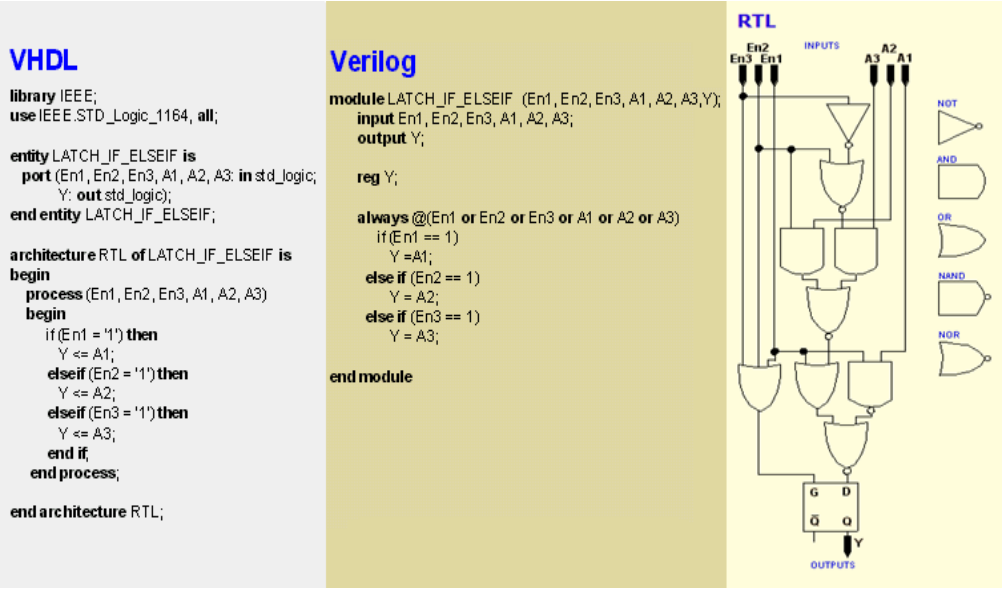


Figura 1- 19 descripción de un mismo circuito en lenguaje VHDL, VERILOG y RTL

1.2.2 SOPC BUILDER

Altera proporciona una infraestructura completa para crear sistemas con microprocesadores embebidos, completamente a medida según las necesidades del diseñador por medio de la combinación de una serie de componentes configurables sobre sus FPGAs.

Para ello, proporciona un entorno específico, al que denomina SOPC BUILDER que se utiliza en conjunto con el software Quartus II permitiendo al usuario diseñar sistemas basados en el procesador NIOS II, simplemente seleccionando las unidades funcionales deseadas y especificando sus parámetros y puede ser implementado directamente sobre una FPGA de la marca.

Para implementar un sistema útil es necesario añadir otras unidades funcionales. SOPC Builder incorpora una biblioteca de componentes pre-configurados incluyendo el procesador NIOS II, controladores de memoria, interfaces I/O, temporizadores, interfaces de comunicación y periféricos que son interconectados por medio de una red de interconexión llama la matriz de conmutación Avalon (Avalon Switch Fabric), resultando un sistema virtual que puede ser conectado con el

mundo exterior a través de los pines programables de la FPGA o conectado internamente a otros componentes.

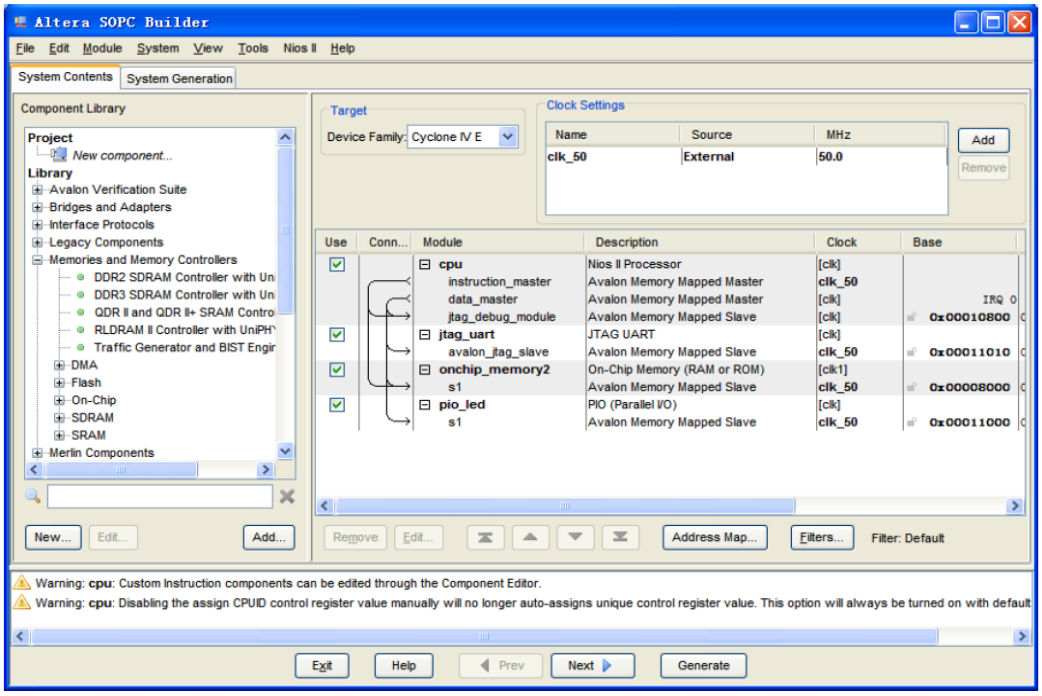


Figura 1- 20 interfaz de usuario de SOPC BUILDER

1.2.3 NIOS II SOFTWARE BUILD TOOLS FOR ECLIPSE

NIOS II Software Build Tools for Eclipse es una pequeña capa de interfaz gráfica que presenta un entorno de desarrollo unificado que funciona para todos los procesadores NIOS II. Es una herramienta que se puede integrar con ECLIPSE un entorno de desarrollo integrado de código abierto multiplataforma desarrollado originalmente por IBM. Se

puede llevar a cabo todas las tareas de desarrollo de software dentro de Eclipse incluyendo creación, edición, construcción, funcionamiento, depuración y perfilado de programas. Por otro lado, Eclipse proporciona amplias capacidades de interacción, depuración y gestión de archivos.

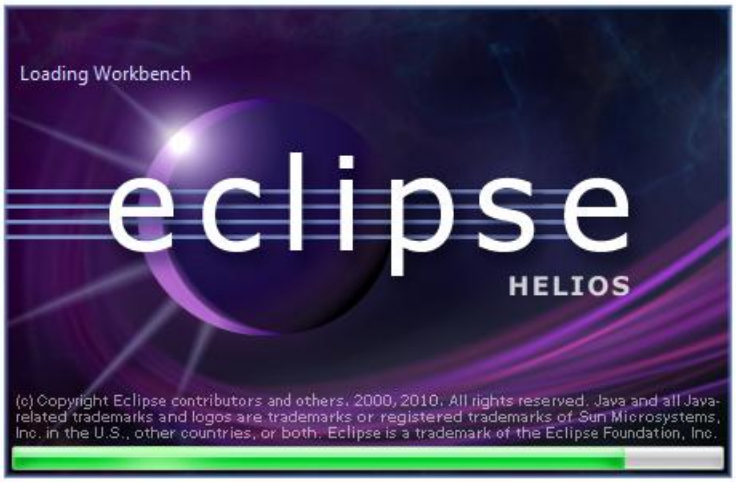


Figura 1- 21 presentación del entorno de desarrollo ECLIPSE

NIOS II Software Build Tools nos permite crear las librerías indispensables para programar sistemas basados en NIOS II. Al utilizar este entorno se emplea una capa de software HAL (Hardware Abstraction Layer) que oculta los detalles de la configuración del hardware, haciendo transparente al programador el desarrollo de aplicaciones.

Estas librerías son creadas a partir del archivo SOPC Information File (.sopcinfo) generado por SOPC BUILDER que contiene el diseño interno del hardware para la FPGA. Una vez generadas las librerías, NIOS II nos permite utilizar lenguaje C/C++ para desarrollar aplicaciones que se encarga de controlar el procesador NIOS II. Estas aplicaciones generan un archivo ejecutable .ELF que se almacena en la FPGA para ser ejecutado por el procesador NIOS II.

Muchos sistemas basados en procesadores NIOS II utilizan memoria flash externa para almacenar:

- Código de programa
- Datos del programa
- Datos de configuración de la FPGA
- Sistema de archivos

Nios II Software Build Tools for Eclipse proporciona una utilidad que permite programar los chips de memoria flash. El programador permite grabar y administrar programas en cualquier tarjeta, incluyendo las placas de desarrollo Altera.

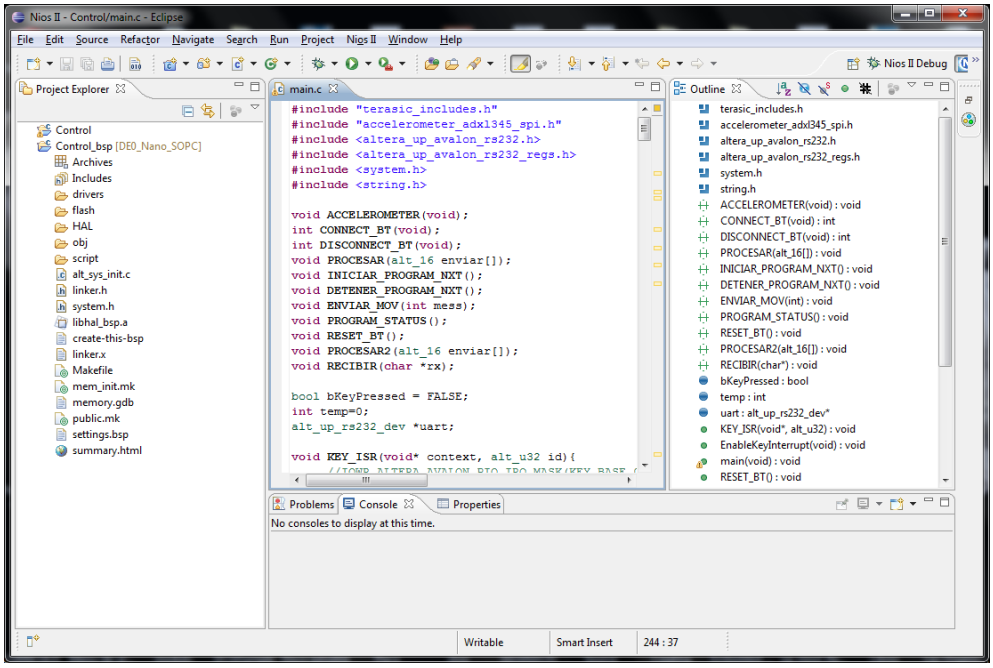


Figura 1- 22 Interfaz de usurario de NIOS II SOFTWARE BUILD TOOLS

1.2.4 NXT 2.0 PROGRAMMING

El software de LEGO MINDSTORMS NXT que usamos para nuestro proyecto es el NXT 2.0 Programming el cual nos permite programar el robot NXT y descargar sus programas al NXT por medio del puerto de USB o por conexión de Bluetooth. Este software, que funciona por medio de dar clics y de arrastrar bloques, el cual es impulsado por LabVIEW de National Instruments, incluye instrucciones de desarrollo y guías de programación para empezar a desarrollar y programar fácilmente robots Lego MINDSTORMS NXT.

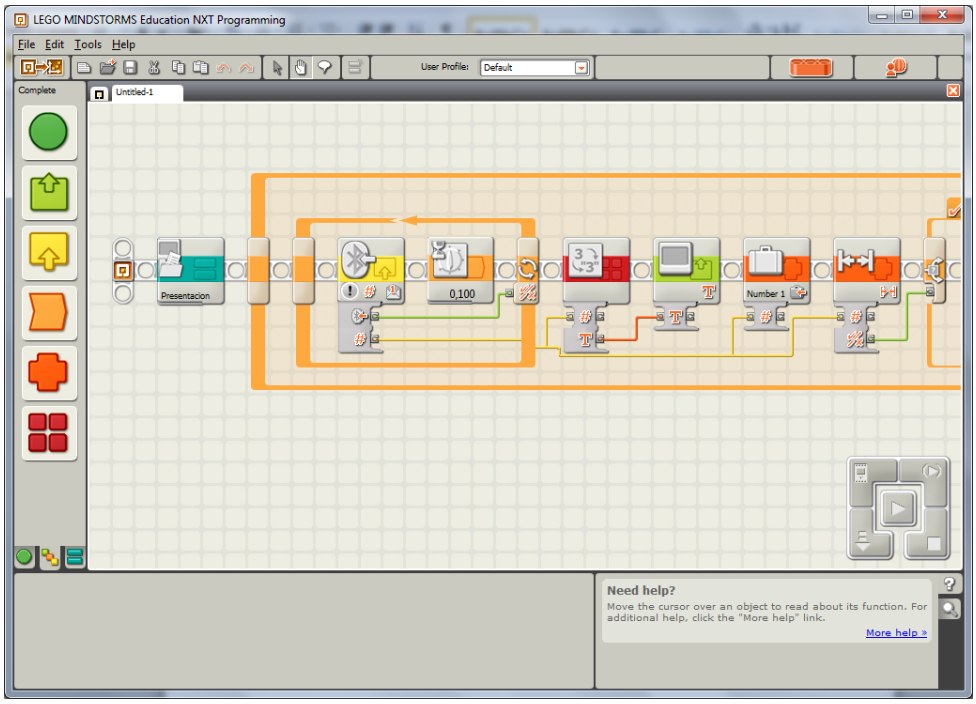


Figura 1- 23 interfaz de usuario del software NXT 2.0 PROGRAMMING

Desarrollar los programas en el entorno del NXT 2.0 Programming es sencillo ya que consta con bloques de funciones creados listos para usarse de acuerdo a la aplicación que queramos implementar.

Figura 1- 24 bloques del programa NXT 2.0 PROGRAMMING

Simplemente se debe dar clic y arrastrar los bloques del lado izquierdo de la pantalla al diagrama. Cada bloque desempeña una función única

como mover los motores, mostrar mensaje en pantalla, detectar sonido o medir temperatura.

Figura 1- 25 controlador y modo de conexión con la computadora

El entorno de desarrollo también permite descargar las aplicaciones programadas al robot Lego a través del programador que incluye. El programa compilado se envía al NXT utilizando conexión inalámbrica bluetooth o conectando el cable USB. El NXT en ese momento ejecuta el código que recibe de la computadora anfitriona.

CAPÍTULO 2

2. DISEÑO

Este capítulo muestra el diseño del hardware y software que se utilizó para alcanzar cada uno de los objetivos del proyecto. En primera instancia se describe de una manera muy general el diseño del hardware utilizado, para luego mostrar la parte medular del proyecto que es el diseño del software. Para la explicación se ha elaborado un diagrama de bloques simplificado que resume el funcionamiento básico del proyecto

2.1 HARDWARE

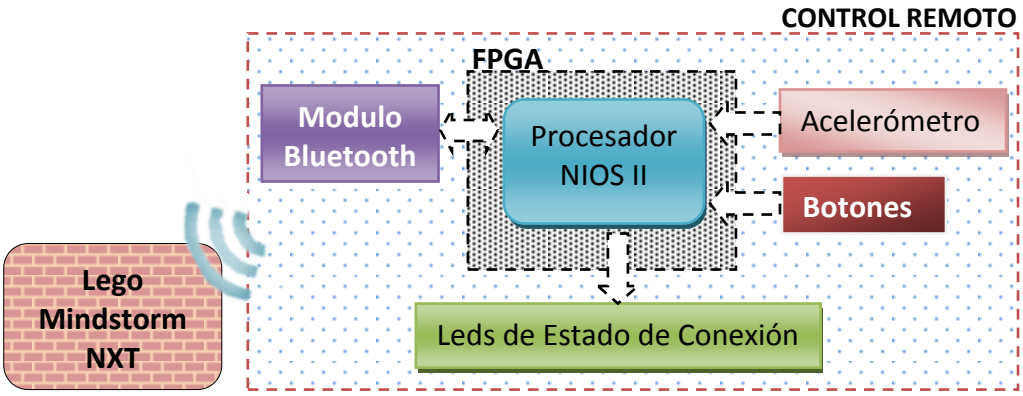


Figura 2-1 diagrama de bloques del control remoto

El proyecto en general consiste en controlar el movimiento de un robot LEGO MINDSTORM NXT con un acelerómetro a través de bluetooth. El proyecto consta de 2 partes:

- El diseño del hardware y software del control remoto
- El diseño del software del robot LEGO MINDSTORM NXT

Para el diseño del control remoto usamos la tarjeta DE0-NANO que tiene embebido un acelerómetro, puertos I/O que nos permitirán comunicar el modulo bluetooth con el procesador NIOS II, una FPGA donde grabamos el sistema de procesamiento de datos basado en el procesador NIOS II, botones y leds que nos permiten establecer comunicación y saber el estado de conexión entre el control remoto y el robot LEGO MINDSTORM NXT.

Figura 2-2 modelo de comunicación entre el robot y el control remoto

El proyecto funciona de la siguiente manera:

Al encender el control remoto un led de estado se encuentra parpadeando lentamente indicando que no hay comunicación entre en robot y el control. Para establecer comunicación se presiona el botón 2, inmediatamente el led que parpadea lentamente empieza a parpadear rápidamente indicando que se está tratando de establecer comunicación entre el robot y el control. El proceso de establecimiento de comunicación entre el robot y el control dura aproximadamente 3 segundos. Una vez establecida la comunicación, desde el control se envía automáticamente un comando al robot para ejecutar el software que recibe los movimientos del acelerómetro.

Dependiendo de la inclinación del control, el acelerómetro genera datos que son enviados al procesador NIOS II a través de comunicación SPI para obtener rangos de valores adecuados que puedan ser interpretados por el software del robot como movimientos.

	Posibles Rangos	Descripción
Velocidad	 0 - 100	**0:** Parado **100:** Máxima velocidad
Dirección	 True/False	**True=** Hacia adelante **False=** Hacia atrás
Giro	 -100 - 100	**<0:** Dirigir motor hacia la izquierda **>0:** Dirigir motor hacia la derecha **=0:** Movimiento recto.

Tabla 2- 1 descripción de los rangos de movimiento que usa el robot lego

Una vez procesados, los datos son enviados al robot por medio del módulo bluetooth RN-42. Este módulo usa el protocolo UART para comunicarse con el procesador NIOS II. Este módulo se encuentra conectado con el procesador NIOS II a través de los pines de la FPGA y los pines del puerto GPIO-1.

2.1.1 DISEÑO DEL SISTEMA NIOS II

Para que el control remoto funcione se debe diseñar un sistema de procesamiento basado en el procesador NIOS II dentro de la FPGA. Este sistema nos permite controlar todo el hardware que se encuentra conectado a los pines de la FPGA de la tarjeta DE0-NANO. También permite establecer comunicación con el PC usando el software QUARTUS II, el cual permite guardar el diseño construido en SOPC

BUILDER en la FPGA, realizar modificaciones de las asignaciones de pines de la FPGA a los distintos controladores que usamos.

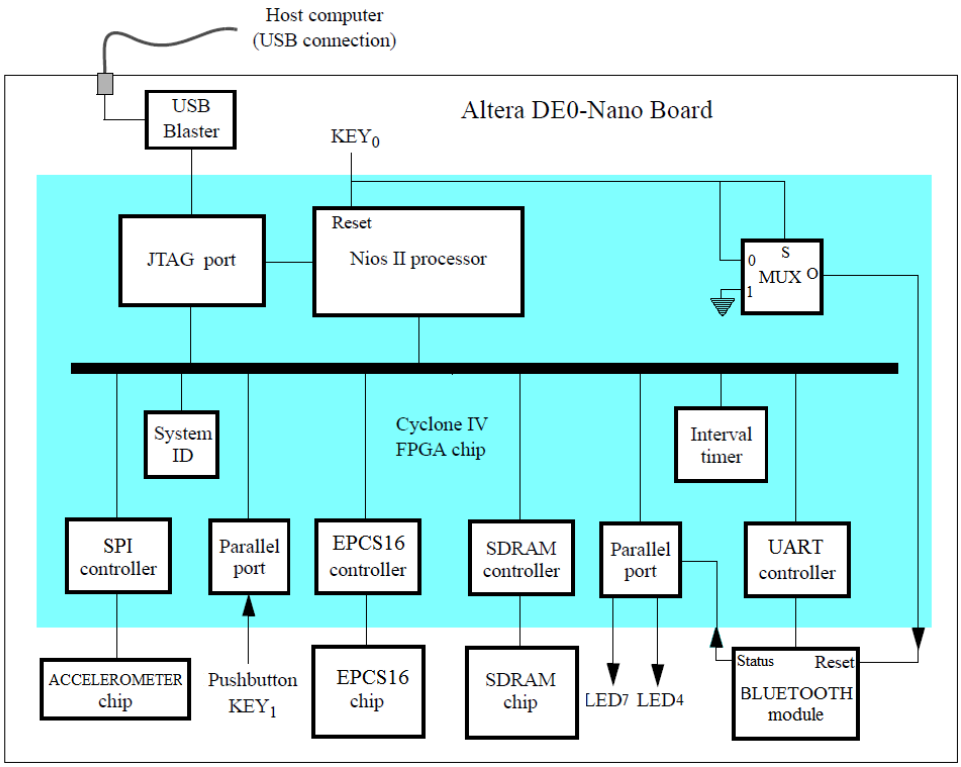


Figura 2-3 diagrama de bloques del sistema NIOS II

Como muestra la figura 2-3 el diseño consta de un procesador NIOS II, bus Avalon, puerto JTAG y varios controladores que permiten establecer comunicación entre los chips y el procesador. Estos dispositivos se comunican con el microprocesador a través del bus Avalon excepto el multiplexor debido a que este componente solo controla el Reset del procesador y del módulo bluetooth.

El controlador EPCS16 permite establecer comunicación con el chip EPCS16 para poder grabar con el programador que incluye NIOS II Software Build Tools la configuración de la FPGA y el código del programa que ejecuta el procesador NIOS II. Este controlador contiene una pequeña memoria on-chip tipo ROM donde se almacena un pequeño programa “bootloader” el cual se ejecuta al encender la tarjeta DE0-NANO y se encarga de:

- Copiar desde el chip EPCS16 los datos de configuración de la FPGA a la FPGA Cyclone IV de la tarjeta DE0-NANO.
- Copiar desde el chip EPCS16 el código del programa que ejecuta el procesador NIOS II a la memoria SDRAM.

Los controladores UART y SPI permiten la comunicación del módulo bluetooth y el acelerómetro con el procesador NIOS II. Los parallel port permiten al procesador NIOS II verificar el estado del botón 1 y el estado de conexión del módulo bluetooth. También se utiliza para mostrar en el LED7 y LED4 el estado de conexión bluetooth entre el robot y el control remoto sin intervención del procesador NIOS II. El controlador SDRAM permite al procesador NIOS II usar la memoria SDRAM para ejecutar el programa que controla el sistema.

El JTAG port permite comunicar la FPGA con el PC usando el software QUARTUS II o el programador flash que incorpora NIOS II Software Budil Tools for Eclipse para poder realizar pruebas, modificaciones y depuraciones del software del sistema.

2.1.2 DISEÑO DEL CONTROL REMOTO

El diseño del control remoto consta de 3 partes:

- Tarjeta DE0-NANO
- Modulo Bluetooth RN-42.
- Fuente de poder externa recargable NOKIA BL-5B

El módulo bluetooth esta soldado a un PCB el cual permite conectarlo con facilidad al puerto GPIO-1 de la tarjeta DE0-NANO. La fuente de poder externa está ubicada en la parte inferior de la tarjeta y va conectada al puerto de entrada de voltaje de 2 pines. No se requirió diseñar ni construir circuitos electrónicos adicionales debido a que todo lo que necesitamos ya esta embebido en la tarjeta DE0-NANO.

PCB BLUETOOTH		DE0-NANO GPIO1		FPGA	
PIN	**NOMBRE**	**PIN**	**NOMBRE**	**PIN**	**NOMBRE**
2	RX	**22**	GPIO_117	PIN_K16	TX-UART
3	RESET	**23**	GPIO_118	PIN_R16	RESET
4	TX	**24**	GPIO_119	PIN_L15	RX-UART
5	PIO2	**25**	GPIO_120	PIN_P15	LED6
7	PIO5	**27**	GPIO_122	PIN_R14	LED5
9	VDD	**29**	VCC3P3	---	---
10	GND	**30**	GND	---	---

Tabla 2- 2 asignación de pines entre el PCB del bluetooth, la DE0 NANO y la FPGA

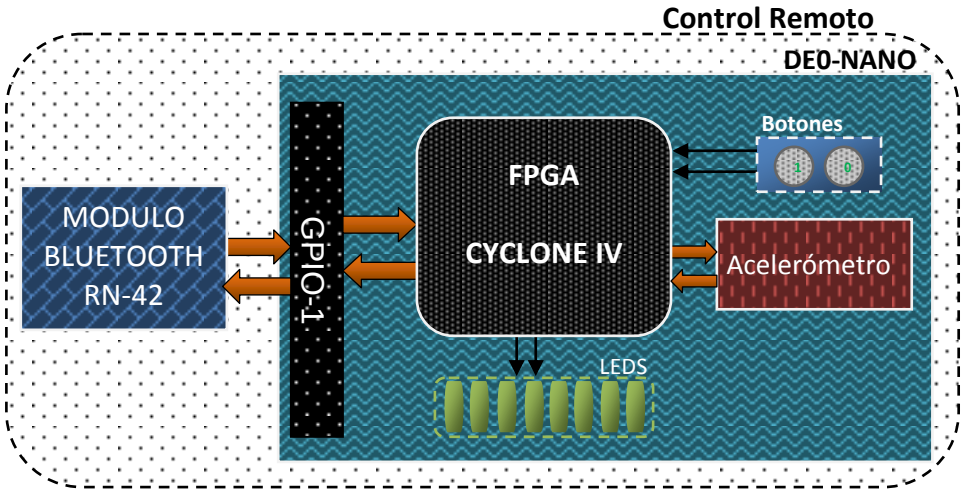


Figura 2-4 componentes del control remoto

2.1.3 DESCRIPCION DEL HARDWARE DEL ROBOT LEGO

El LEGO MINDSTORMS NXT posee un microcontrolador ARM7 de 32 bits a 48MHz que incluye 256Kb de memoria Flash y 64Kb de RAM, el cual permite mayores capacidades de ejecución de programas,

evitando que los procesos inherentes de varios paquetes de datos colisionen y produzcan errores y un posible error en la ejecución del software. Contiene un chip bluetooth Bluecore 4.0 el cual es implementado como máquina virtual con un intérprete de comandos permitiendo al procesador ARM intercambiar comandos y secuencias de datos a través del protocolo UART.

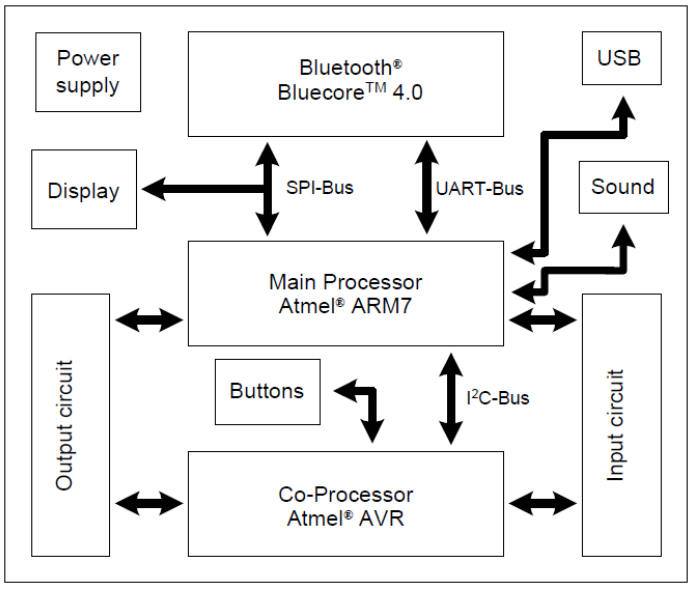


Figura 2-5 diagrama de bloques del hardware del ROBOT LEGO

El bloque NXT contiene cuatro entradas que se encuentran en la parte inferior del bloque los cuales permiten conectar distintos tipos de sensores y tres salidas de energía (A, B, C) localizadas en la parte superior del bloque permitiendo conectar los motores. Los sensores que se pueden conectar al bloque NXT son sensor de luz, sensor de temperatura, sensor de contacto, sensor de giro y sensor ultrasónico.

En la parte frontal del bloque NXT se encuentra un control de 4 botones de goma y una pantalla LCD de 100x64 pixeles que se encuentra conectada al procesador principal por medio de un bus SPI. También incluye un amplificador de audio y un altavoz de 16Ω con un ancho de banda de 2-16Khz.

El modelo NXT usa servo motores, los cuales permiten saber a que velocidad se están moviendo los motores, y corregirla si es necesario. Además permite saber en todo momento cuantos grados a girado el motor de manera exacta.

Figura 2-6 dispositivos externos que usa el ROBOT LEGO

Además incluye un módulo Bluetooth Clase II interno, el cual ofrece un rango efectivo de trabajo aproximado de 20m y permite conectarse a otros dispositivos que implementen el Serial Port Profile (SPP) proporcionando una comunicación maestro/esclavo de 4 canales. El

canal 0 lo usa cuando trabaja como esclavo y los otros 3 al trabajar como maestro, esto significa que en modo master el bloque NXT se puede comunicar con 3 o más dispositivos.

2.2 SOFTWARE

2.2.1 DIAGRAMA DE FLUJO DEL SOFTWARE DE LA FPGA

El software es el programa que permite hacer funcionar el sistema basado en el procesador NIOS II. Permite al procesador establecer comunicación, procesar datos y tener el control total de los dispositivos externos que se encuentran conectados a los pines de la FPGA.

El diagrama de flujo muestra de forma general el funcionamiento del software que fue implementado en lenguaje C usando Nios II Software Build Tools.

El software funciona de la siguiente manera:

Al encender el control remoto, el software es ejecutado por el procesador NIOS II y se muestra el Led 7 parpadeando indicando que no se ha establecido conexión con el robot. Para establecer

conexión con el robot se debe presionar el botón 1, una vez establecida la comunicación, se envía un comando al robot para ejecutar el programa que recibe los movimientos del control. Además el Led 7 se apaga y se enciende el Led 5 indicando conexión con el robot.

Mientras el programa en el robot este ejecutándose y exista conexión bluetooth entre el control remoto y el robot, el software permite controlar el movimiento del robot enviando los rangos de velocidad y giro dependiendo de la inclinación del control remoto indicada por el acelerómetro.

Finalmente si se desea parar la ejecución del programa se debe volver a presionar el botón 1 el cual termina la conexión con el robot y se enciende el Led 7 indicando desconexión.

Usamos interrupciones para obtener el estado del botón 1, el cual permite iniciar o terminar la conexión con el robot. Los Leds indicadores de estado de conexión bluetooth no son controlados por el procesador NIOS II, están directamente conectados a los pines de estado del módulo bluetooth a través de la FPGA.

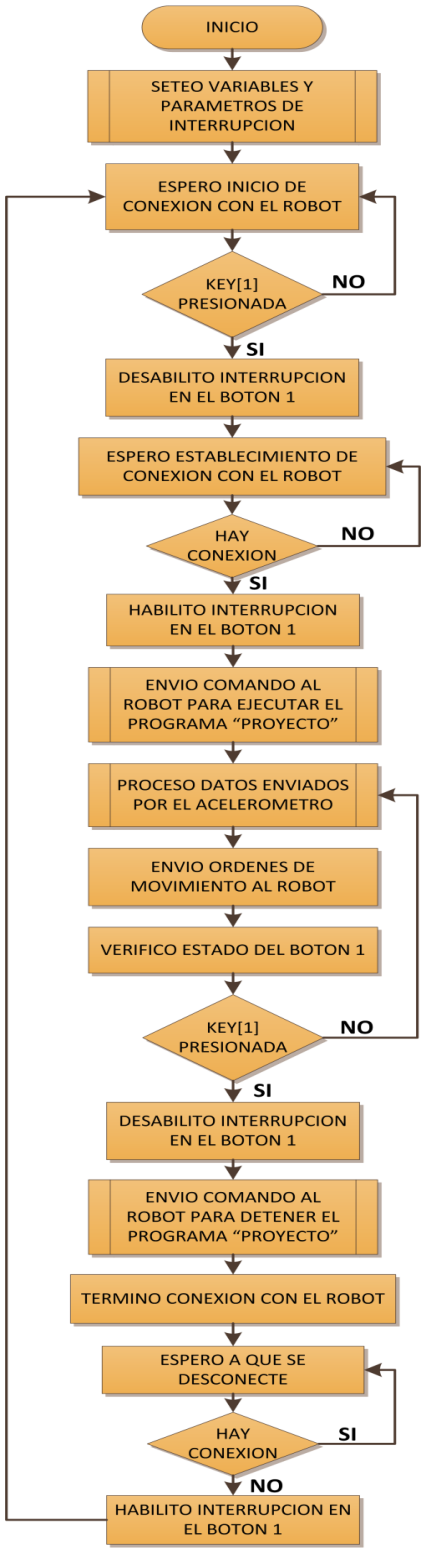


Figura 2-7 diagrama de flujo del software de la FPGA

2.2.2 DIAGRAMA DE FLUJO DEL SOFTWARE DEL ROBOT

El programa que se diseña para el robot, se encarga de recibir los datos que representan la inclinación en que se encuentra el control. Estos datos son interpretados por este software como movimientos hacia las distintas direcciones o como velocidad dependiendo de los rangos de valores recibidos.

El programa funciona de la siguiente manera:
Este software es ejecutado remotamente desde el control remoto, el cual una vez ejecutado espera recibir los datos enviados por el control para proceder a determinar si corresponden a dirección de movimiento o velocidad y los almacena en distintas variables que son usadas como parámetros de configuración de los motores haciendo posible el movimiento del robot.

También muestra en la pantalla del robot información sobre el movimiento como el rango de velocidad, dirección y sentido; estos datos están cambiando constantemente mientras exista comunicación con el robot. El programa se sigue ejecutando de manera infinita hasta que el usuario decida terminar la comunicación con el robot presionando el botón 1 en el control remoto.

El rango de velocidad de 0-100 corresponde a valores de porcentaje 0% a 100% de la velocidad máxima de los motores del robot Lego Mindstorms NXT 170rpm.

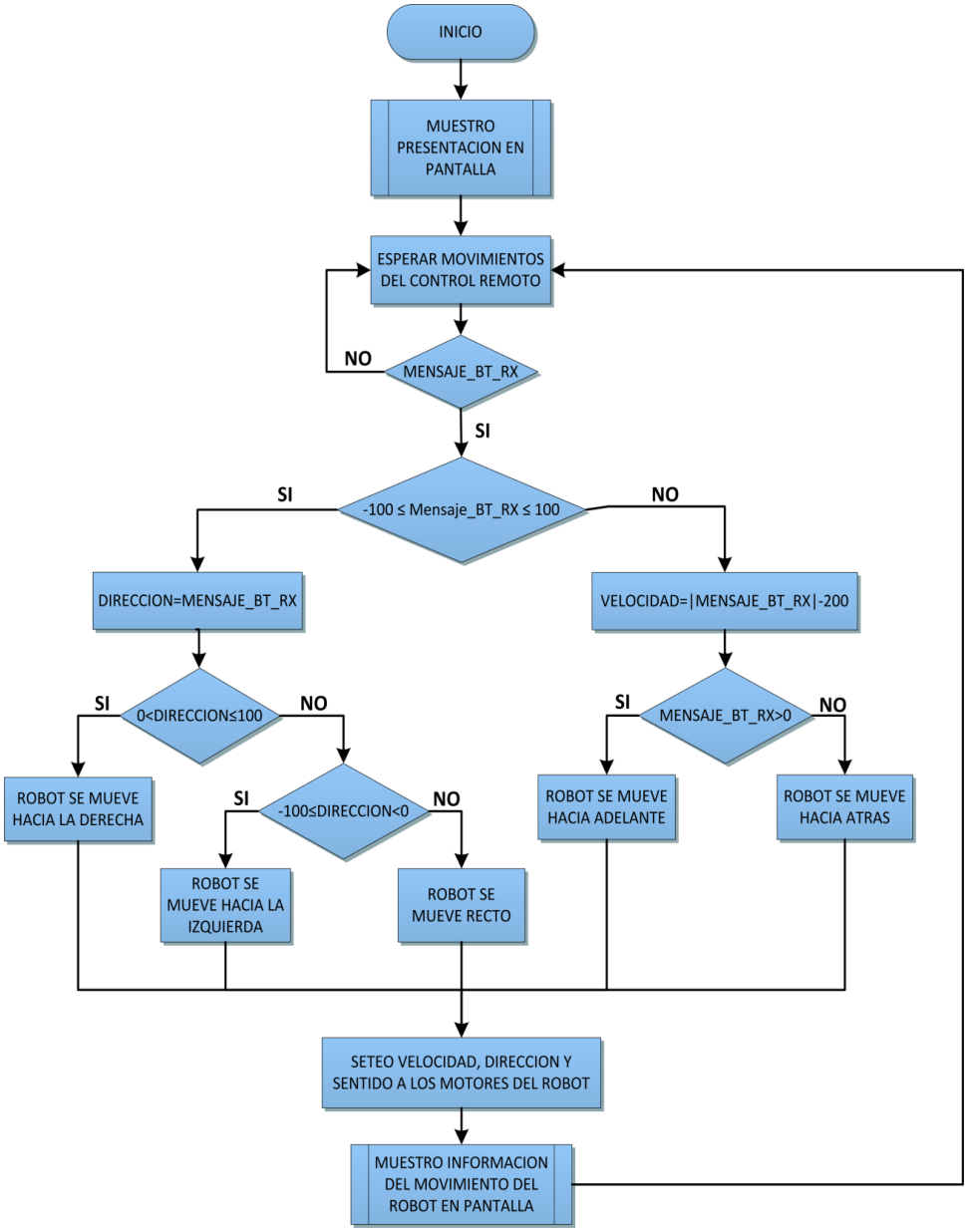


Figura 2-8 diagrama de flujo del software del robot

CAPÍTULO 3

3. IMPLEMENTACIÓN

3.1 IMPLEMENTACIÓN DEL SISTEMA NIOS II

Para realizar la implementación del sistema basado en el procesador NIOS II en la FPGA usamos la herramienta SOPC BUILDER que forma parte de QUARTUS II. Esta herramienta permite construir el hardware interno de la FPGA usando componentes que permiten controlar el hardware externo conectado a los pines de la FPGA. El proyecto está compuesto de los siguientes componentes:

COMPONENTE	DESCRIPCIÓN
CPU	Procesador NIOS II/fast de 100Mhz RISC de 32 bits
SDRAM	Memoria RAM de 32MB que usa el procesador NIOS II
JTAG_UART	Interfaz de comunicación serial entre el PC y el procesador NIOS II
CLK_50	Oscilador externo de 50Mhz
SYSID	Número identificador único del sistema NIOSII generado
ALTPLL_SYS	Módulo PLL
CLOCK_CROSSING_IO	Bus de comunicación Avalon Memory-Mapped entre el procesador NIOS II y los controladores
RS232_0	Interfaz de comunicación serial UART entre el procesador NIOS II y el módulo bluetooth
STADO_BT	Interfaz entre el LED 7 y el pin de estado de conexión del módulo Bluetooth
RESET_BT	Interfaz entre KEY[0] y el pin reset del módulo Bluetooth
EPCS_FLAHS_CONTROLLER	Interfaz de comunicación serial entre el procesador NIOS II y el chip EPCS16
GSENSOR_SPI	Interfaz de comunicación SPI entre el procesador NIOS II y el acelerómetro ADXL345

Tabla 3- 1 lista de componentes usados en el sistema NIOS II

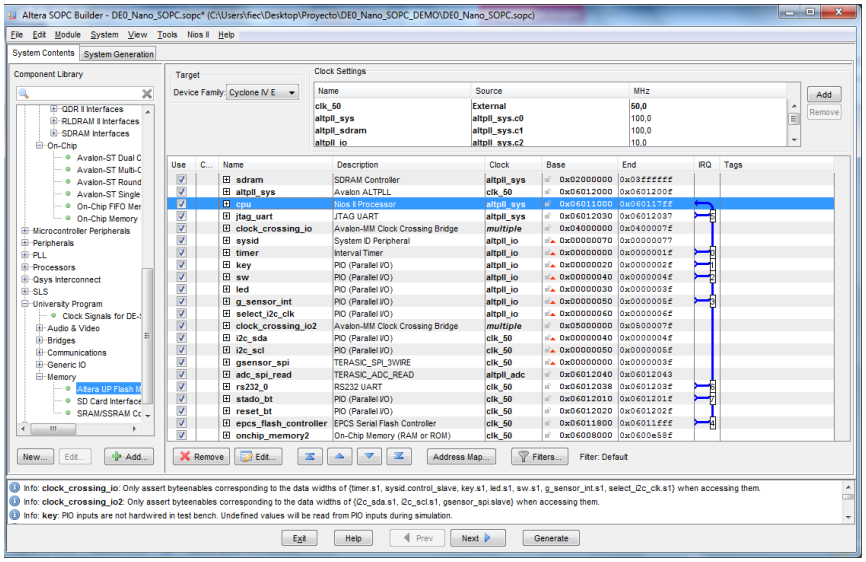


Figura 3- 1 componentes usados en SOPC BUILDER

El CPU es un procesador NIOS II/f de 32bits que se encuentra conectado a los distintos controladores a través del bus de comunicación AVALON. Cada controlador está conectado a su respectivo hardware externo a través de los pines de la FPGA. La asignación de los pines a estos controladores se lo realiza con la herramienta PIN PLANNER en QUARTUS II. Una vez que se ha terminado de construir el hardware, SOPC BUILDER genera la maquina usando lenguaje VHDL o Verilog y genera un archivo “DE0_Nano_SOPC.sopcinfo” que contiene toda la configuración del sistema.

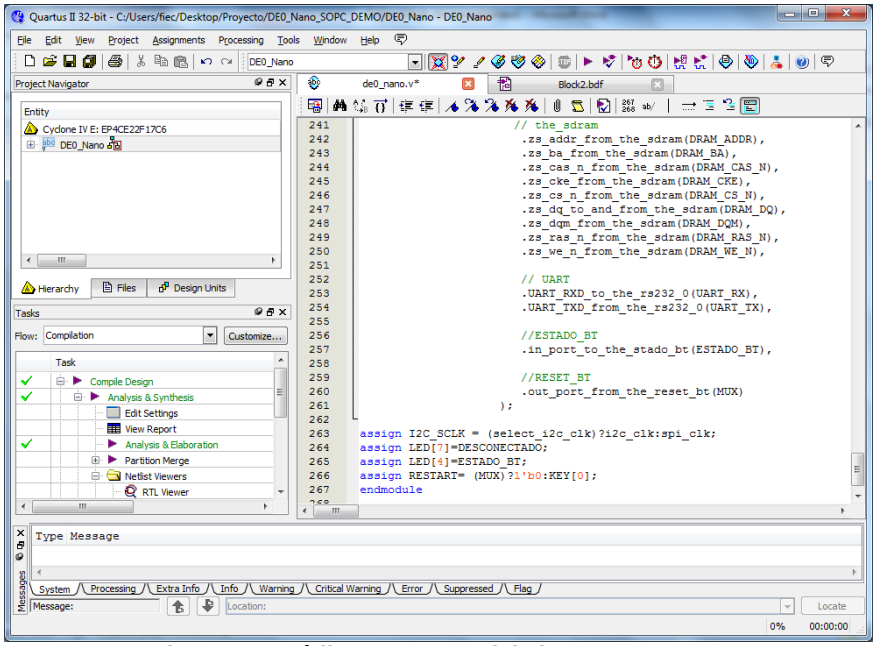


Figura 3- 2 Código VERILOG del sistema NIOS II

3.2 PROGRAMACION DEL SOFTWARE DE LA FPGA

La programación del control remoto se la realiza en el entorno de desarrollo NIOS II Software Build Tools for Eclipse, el cual permite generar los SDKs a partir del archivo .sopcinfo. Estas librerías son generadas en lenguaje C y contienen todo lo necesario para realizar la programación. Esta herramienta también permite guardar la configuración de la FPGA archivo .sof y el archivo binario del programa .elf en el chip EPCS16 usando la herramienta NIOS II Flash Programmer. El programa consta de varias funciones que se

encargan de realizar la comunicación con el robot y procesar los datos obtenidos del acelerómetro.

FUNCION	DESCRIPCION
`void ACCELEROMETER(void);`	Función que se encarga de recibir los datos del acelerómetro.
`int CONNECT_BT(void);`	Función que establece la conexión bluetooth con el robot.
`int DISCONNECT_BT(void);`	Función que se encarga de realizar la desconexión bluetooth entre el control y el robot
`void PROCESAR(alt_16 enviar[]);`	Función que procesa los datos del acelerómetro en movimientos IZQ y DER
`void INICIAR_PROGRAM_NXT();`	Envía comando al robot para ejecutar el programa que recibe los movimientos
`void DETENER_PROGRAM_NXT();`	Detiene el programa en el robot
`void ENVIAR_MOV(int mess);`	Envía al robot los movimientos procesados. Usa el formato del protocolo de comunicación de Lego.
`void PROGRAM_STATUS();`	Permite saber si el programa en el robot se esta ejecutando
`void RESET_BT();`	Resetea el módulo bluetooth
`void PROCESAR2(alt_16 enviar[]);`	Función que permite aumentar o disminuir la velocidad del robot
`void RECIBIR(char *rx);`	Función que recibe caracteres del protocolo UART y los almacena en un puntero char
`void KEY_ISR(void* context, alt_u32 id);`	Función que es llamada por la interrupción del botón KEY 1
`void EnableKeyInterrupt(void);`	Función que habilita la interrupción en el botón KEY 1

Tabla 3- 2 Lista de funciones que usa la aplicación programada en NIOS II SBT

3.2.1 PROTOCOLO DE COMUNICACIÓN DEL LEGO MINDSTORMS NXT

La comunicación bluetooth entre el robot Lego Mindstorms NXT y el control remoto se la realiza usando un protocolo de comunicación que permita al robot entender los mensajes enviados desde el control remoto. Cada mensaje inicia con 2 bytes que indican cuantos bytes incluye el mensaje y deben tener un límite de 64 bytes sin incluir los 2 primeros bytes.

Length, LSB	Length, MSB	Command Type	Command	Byte 5	Byte 6	Etc.

Figura 3- 3 trama de mensaje usado en la comunicación bluetooth

El tercer byte **Command Type** indica el tipo de mensaje:

- **0×00:** Direct Command, con respuesta
- **0×01:** System Command, con respuesta
- **0×02:** Respuesta
- **0×80:** Direct Command, sin respuesta
- **0×81:** System Command, sin respuesta

- Los ***Direct Command*** son para controlar aspectos relacionados con acciones del usuario como ejecutar un programa, controlar sensores, motores, etc.

- Los ***System Command*** son específicos para controlar la FLASH como abrir, leer, escribir ficheros, reiniciar el sistema, etc.

El cuarto byte **Command** indica el comando propiamente dicho. Cada comando tiene un código hexadecimal que lo representa. Para una lista completa de los comandos que usa el robot Lego, consulte "Appendix 2–LEGO MINDSTORMS NXT Direct commands".

COMANDO	HEXADECIMAL	DESCRIPCION
STARTPROGRAM	0X00	Inicia un programa en el robot
STOPPROGRAM	0x01	Cierra un programa en el robot
MESSAGEWRITE	0x09	Escribe un mensaje

Tabla 3- 3 lista de comandos usados para controlar el robot

El resto de bytes dependen del tipo de comando. En nuestro caso:

- El comando **MESSAGEWRITE** requiere que se especifique el "mailbox" remoto y el tamaño del mensaje que enviamos, incluyendo el '\0' al final, ya que se tratan las cadenas como en C.

- El comando **STARTPROGRAM** requiere que se especifique el nombre del archivo a ejecutar. Este nombre debe estar en formato ASCIIZ-String con un tamaño máximo de [15.3 caracteres] + '\0'.

```
//LIBRERIAS
#include "terasic_includes.h"
#include "accelerometer_adxl345_spi.h"
#include <altera_up_avalon_rs232.h>
#include <altera_up_avalon_rs232_regs.h>
#include <system.h>
#include <string.h>

//DECLARACION DE FUNCIONES
void ACCELEROMETER(void);
int CONNECT_BT(void);
int DISCONNECT_BT(void);
void PROCESAR(alt_16 enviar[]);
void INICIAR_PROGRAM_NXT();
void DETENER_PROGRAM_NXT();
void ENVIAR_MOV(int mess);
void PROGRAM_STATUS();
void RESET_BT();
void PROCESAR2(alt_16 enviar[]);
void RECIBIR(char *rx);

//VARIABLES GLOBALES
bool bKeyPressed = FALSE;
int  temp=0;
alt_up_rs232_dev *uart;

void KEY_ISR(void* context, alt_u32 id){
        IOWR_ALTERA_AVALON_PIO_EDGE_CAP(KEY_BASE,0);
      if (id == KEY_IRQ){
        if (bKeyPressed==1)
                bKeyPressed=0;
        else{
                bKeyPressed=1;
         }
       IOWR_ALTERA_AVALON_PIO_EDGE_CAP(KEY_BASE,0);
    }
}

void EnableKeyInterrupt(void){
    int error;
    // habilitainterrupcion en el boton KEY[1]
    IOWR_ALTERA_AVALON_PIO_IRQ_MASK(KEY_BASE,0x02);
    // limpia la bandera de interrupcion
    IOWR_ALTERA_AVALON_PIO_EDGE_CAP(KEY_BASE,0);

    bKeyPressed = 1;
    // Asocia a la interrupcion la function a ejecutar
    error = alt_irq_register (KEY_IRQ, 0, KEY_ISR);
    if (error)
        printf("Error al registrar la interrupcion\r\n");
}

void main(void){

char estado[10];
EnableKeyInterrupt();   //HABILITA LA INTERRUPCION
printf("inicio\n");
while(1){
        while(bKeyPressed){       //ESPERANDO QUE SE PRESIONE EL BOTON KEY[1]
        }
        if (CONNECT_BT()!=0){
                RESET_BT();
                break;
        }
        IOWR_ALTERA_AVALON_PIO_IRQ_MASK(KEY_BASE,0x00);
        while(IORD_ALTERA_AVALON_PIO_DATA(0x6012010)==0){//ESPERO QUE SE CONECTE EL CONTROL CON
ROBOT
                IOWR_ALTERA_AVALON_PIO_EDGE_CAP(KEY_BASE,0);
        }
        IOWR_ALTERA_AVALON_PIO_IRQ_MASK(KEY_BASE,0x02);
        INICIAR_PROGRAM_NXT();
        RECIBIR(&estado);
        if (estado[4]!=0)
                printf("error\n");
        ACCELEROMETER();
        IOWR_ALTERA_AVALON_PIO_IRQ_MASK(KEY_BASE,0x00);
        DETENER_PROGRAM_NXT();
```

```
        RECIBIR(&estado);
        if (estado[4]!=0)
                printf("error\n");
        if (IORD_ALTERA_AVALON_PIO_DATA(0x6012010)==1){
                if (DISCONNECT_BT()==-1){
                        printf("Error Desconexion\n");
                }
        }
        while(IORD_ALTERA_AVALON_PIO_DATA(0x6012010)==1){//ESPERO QUE SE DESCONECTE EL CONTROL CON
ROBOT
                IOWR_ALTERA_AVALON_PIO_EDGE_CAP(KEY_BASE,0);
        }
        IOWR_ALTERA_AVALON_PIO_IRQ_MASK(KEY_BASE,0x02);
        printf("desconectado\n");
}
}

void RESET_BT(){
        IOWR_ALTERA_AVALON_PIO_DATA(0x6012020,1);
        usleep(5000);
        IOWR_ALTERA_AVALON_PIO_DATA(0x6012020,0);
}

int CONNECT_BT(){
        int error=0;
        char rx[10];
        uart = alt_up_rs232_open_dev("/dev/rs232_0");
        error=alt_up_rs232_write_data(uart, '$');
        error=alt_up_rs232_write_data(uart, '$');
        error=alt_up_rs232_write_data(uart, '$');
        RECIBIR(&rx);
        if ((strcmp(rx,"CMD\r\n\0")!=0) && error!=0)
                error=-1;
        else{
                error=alt_up_rs232_write_data(uart, 'C');
                error=alt_up_rs232_write_data(uart, '\n');
        }
        printf("Error: %d",error);
        return(error);
}

int DISCONNECT_BT(){
        int error=0;
        char rx[10];
        uart = alt_up_rs232_open_dev("/dev/rs232_0");
        error=alt_up_rs232_write_data(uart, '$');
        error=alt_up_rs232_write_data(uart, '$');
        error=alt_up_rs232_write_data(uart, '$');
        RECIBIR(&rx);
        if ((strcmp(rx,"CMD\r\n\0")!=0) && error!=0)
                error=-1;
        else{
                error=alt_up_rs232_write_data(uart, 'K');
                error=alt_up_rs232_write_data(uart, ',');
                error=alt_up_rs232_write_data(uart, '\n');
        }
        printf("\nError %d\n",error);
        return(error);
}

void ACCELEROMETER(void){
    bool bSuccess;
    alt_16 szXYZ[3];
    alt_u8 id;
    const int mg_per_digi = 4;

    IOWR(SELECT_I2C_CLK_BASE, 0, 0x00);

    // configura accelerometro con +-2g e inicio la captura delos datos del acelerometro
    bSuccess = ADXL345_SPI_Init(GSENSOR_SPI_BASE);
    if (bSuccess){
        bSuccess = ADXL345_SPI_IdRead(GSENSOR_SPI_BASE, &id);
        if (bSuccess)
            printf("id=%02Xh\r\n", id);
    }

    if (bSuccess)
```

```
        printf("Capturando datos del Acelerometro\r\n");

    while(bSuccess && !bKeyPressed && IORD_ALTERA_AVALON_PIO_DATA(0x6012010)){//!
        if (ADXL345_SPI_IsDataReady(GSENSOR_SPI_BASE)){
            bSuccess = ADXL345_SPI_XYZ_Read(GSENSOR_SPI_BASE, szXYZ);
            if (bSuccess){
                 printf("X=%d mg, Y=%d mg, Z=%d mg\r\n", szXYZ[0], szXYZ[1], szXYZ[2]);
                PROCESAR(szXYZ);
                if (temp==0){
                 PROCESAR2(szXYZ);
                }
                usleep(1000*200);
            }
        }
    }
    bKeyPressed=1;
    if (!bSuccess)
        printf("Error al acceder al accelerometro\r\n");
}



void ENVIAR_MOV(int mess){
        int B=0,OK=-1;
        unsigned char message[4]={0};
        unsigned int i=0,length=0;
        unsigned char command[64];
        uart = alt_up_rs232_open_dev("/dev/rs232_0");
        length=sizeof(message);
                message[0]=(char)(mess & 0x000000FF);
                mess=(int)mess>>8;
                message[1]=(char)(mess & 0x000000FF);
                mess=(int)mess>>8;
                message[2]=(char)(mess & 0x000000FF);
                mess=(int)mess>>8;
                message[3]=(char)(mess & 0x000000FF);
        command[0]=length+5;
        command[1]=0x00;
        command[2]=0x80;
        command[3]=0x09;
        command[4]=0x00;
        command[5]=length+1;
        while(i<length){
                command[i+6]=message[i];
                i++;
        }
        command[i+6]='\0';
        while(B<length+7){
                OK=alt_up_rs232_write_data(uart,command[B]);
                B++;
        }
}

void PROCESAR(alt_16 enviar[]){
        int A=0;
        while(A<=110 && enviar[0]/2<=110 && enviar[0]/2>=-110){
                if (abs(enviar[0])/2<=A && abs(enviar[0])/2>=(A-10)){
                        if (enviar[0]>0){
                                printf("%d",A-10);
                                if (temp!=A-10){
                                        ENVIAR_MOV(A-10);
                                        temp=A-10;
                                }
                        }
                        else{
                                printf("%d",-(A-10));
                                if (temp!=A-10){
                                        ENVIAR_MOV(-(A-10));
                                        temp=-(A-10);
                                }
                        }
                        break;
                        //A=100;
                }
                else{
```

```
                        A=A+10;
                }
        }
}

void PROCESAR2(alt_16 enviar[]){
        int A=0;
        if ((enviar[2]>=170 && enviar[2]<=180)||enviar[2]<170){
                A=200;
        }
        else if(enviar[2]>=180 && enviar[2]<=190){
                A=225;
        }
        else if (enviar[2]>=190 && enviar[2]<=200){
                A=250;
        }
        else if (enviar[2]>=200 && enviar[2]<=210){
                A=275;
        }
        else if (enviar[2]>=210 && enviar[2]<=220){
                A=300;
        }
        ENVIAR_MOV(A);
}

void INICIAR_PROGRAM_NXT(){
        int B=0,OK=-1;
        char message[]="Proyecto.rxe";
        unsigned int i=0,length=sizeof(message);
        unsigned char command[64];
        uart = alt_up_rs232_open_dev("/dev/rs232_0");
        command[0]=length+2;
        command[1]=0x00;
        command[2]=0x00;
        command[3]=0x00;
        while(i<length){
                command[i+4]=message[i];
                i++;
        }
        while(B<length+4){
        OK=alt_up_rs232_write_data(uart,command[B]);
        B++;
        }
}

void DETENER_PROGRAM_NXT(){
        int B=0,OK=-1;
        unsigned char command[64];
        uart = alt_up_rs232_open_dev("/dev/rs232_0");
          command[0]=0x02;
          command[1]=0x00;
          command[2]=0x00;
          command[3]=0x01;
          while(B<4){
                OK=alt_up_rs232_write_data(uart,command[B]);
                B++;
          }
}

void PROGRAM_STATUS(){
        int B=0,OK=-1;
        unsigned char command[64];
        uart = alt_up_rs232_open_dev("/dev/rs232_0");
          command[0]=0x02;
          command[1]=0x00;
          command[2]=0x00;
          command[3]=0x11;
          while(B<4){
                OK=alt_up_rs232_write_data(uart,command[B]);
                B++;
          }
}

void RECIBIR(char *rx){
        int A=0;
        while(IORD_ALT_UP_RS232_RAVAIL(0x6012038)==0){}
                while((IORD_ALT_UP_RS232_RAVAIL(0x6012038))!=0)
```

```
                {
                        A=IORD_ALT_UP_RS232_DATA(0x6012038);
                        *rx=(char) A;
                usleep(25000);
                *rx++;
                }
        *rx='\0';
}
```

3.3 PROGRAMACION DEL SOFTWARE DEL ROBOT LEGO

El diseño del programa que se encarga de recibir los movimientos del control remoto es implementado utilizando el software LEGO MINDSTORMS Education NXT Programming v2.0 que se incluye con este robot. Este entorno de desarrollo permite programar aplicaciones para el robot LEGO MINDSTORMS NXT usando lenguaje de programación gráfica basado en LabView. También permite actualizar el firmware del robot y grabar las aplicaciones realizadas usando el programador.

La aplicación que implementamos consta de 5 partes:

1. Muestra mensaje de bienvenida al inicio
2. Espera mensajes dentro de un lazo infinito
3. Procesa los mensaje recibidos y determina velocidad y dirección de movimiento

4. Configura valores de los motores
5. Muestra información sobre el movimiento del robot en pantalla

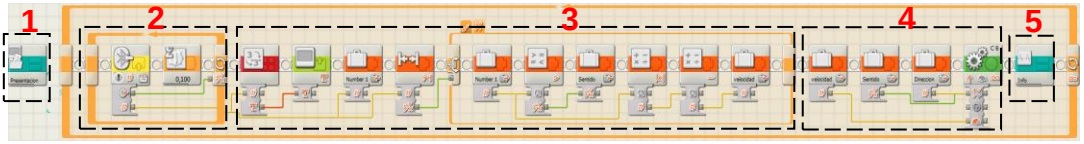


Figura 3- 4 código fuente del programa del ROBOT LEGO

Figura 3-5 Bloque Presentación

El bloque "Presentación" es un subprograma que contiene 5 bloques "Display" que permiten mostrar el mensaje de bienvenida al ejecutar el programa. Cada bloque escribe en cada línea las palabras del mensaje.

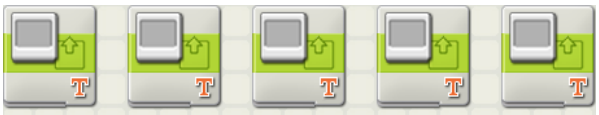


Figura 3- 6 bloque "Display" y mensaje de bienvenida

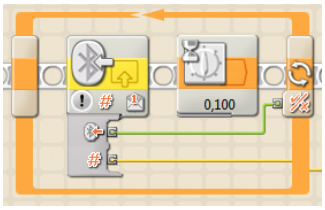


Figura 3- 7 bloque de Recepción

La segunda parte consiste en un lazo infinito que espera a que lleguen los mensajes enviados por el control remoto. El bloque de recepción de mensajes bluetooth nos indica si ha llegado algún mensaje y el bloque de retardo permite monitorear mensajes cada 0.1s.

El procesamiento de los mensajes recibidos se lo realiza en la tercera parte. Los mensajes contienen solo números que indican rango de dirección o velocidad de los motores del robot Lego. El valor recibido se lo guarda en la variable “Number1”, luego el bloque de Rango permite determinar si el valor recibido esta entre -100 y 100.

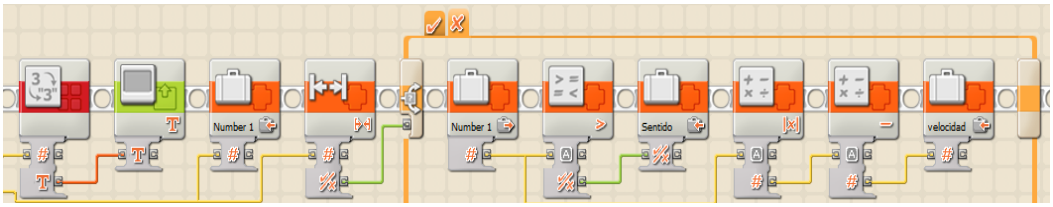


Figura 3- 8 procesamiento de los mensajes

Si el valor se encuentra dentro del rango entonces el valor recibido corresponde a dirección caso contrario el valor corresponde a velocidad. Cada valor será guardado en distintas variables con su respectivo nombre. La variable “sentido” indica el sentido de movimiento del robot. Si el valor recibido es mayor a cero entonces el

robot se mueve hacia adelante. Caso contrario el robot se mueva hacia atrás.

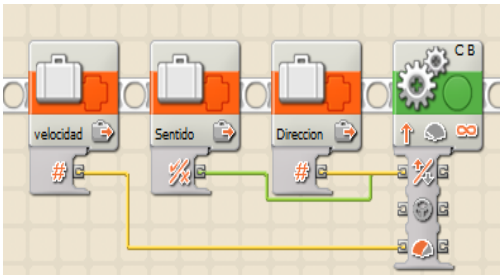


Figura 3- 9 parámetros de configuración de los motores

Una vez determinado si el mensaje contiene valores de velocidad o dirección, se configura los parámetros de los motores para iniciar el movimiento del robot. Estos parámetros están almacenados en las variables “velocidad”, “Dirección” y “Sentido” y se los introduce en el bloque “Move”. Una vez introducidos los parámetros el robot empieza a moverse.

Figura 3- 10 Bloque “Info”

Finalmente se muestra información del movimiento del robot como la velocidad, dirección y sentido de movimiento. Cabe aclarar que el valor de velocidad mostrado no es un valor en RPM sino es el valor del rango 0-100 que se usa para configurar los valores del motor en el bloque “Move”. Lo mismo ocurre con el valor de dirección, solo indica el rango que es de -100 a 100. La letra D o I indica movimiento hacia la derecha o hacia la izquierda.

Figura 3- 11 información del movimiento del robot

El bloque “Info” es un subprograma que contiene los bloques necesarios para mostrar la información en pantalla. Estos conjuntos de bloques se encargan de procesar la información y darle formato para presentarla en pantalla.

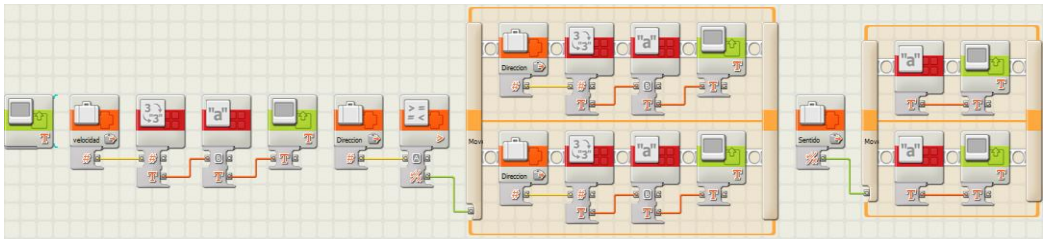


Figura 3- 12 bloques necesarios para mostrar la información del movimiento del robot en pantalla

CAPÍTULO 4

4. PRUEBAS Y RESULTADOS

En este capítulo mostramos las pruebas realizadas con el sistema los cuales los hemos divido en dos: una prueba de distancia de recepción con sus escenarios respectivos las cuales se realizaron en espacio abierto con línea de vista para poder realizar las mediciones y otra prueba de tiempo de respuesta del robot al cambio de dirección de movimiento tomando datos a varias distancias entre 5 y 35 metros.

4.1 PRUEBA DE DISTANCIA DE RECEPCIÒN

Los escenarios para esta prueba son los siguientes:

4.1.1 ESCENARIO A: MEDIR LA DISTANCIA DE RECEPCION CON ESPACIO LIBRE DE OBSTACULOS

Figura 4- 1 escenario A

Según los datos del fabricante del módulo bluetooth RN-42 el radio de alcance máximo es de 20 metros sin obstáculos.

Las pruebas se las realizó en un área de 40 metros sin obstáculos permitiendo línea de vista entre el control remoto y el robot.

Según las pruebas se superó el radio de alcance máximo del fabricante llegando a los 35.5 metros en línea recta y sin obstáculos, teniendo el control total del robot y lográndolo conectar y desconectar cuando deseemos.

Una vez que se sobrepasa esa distancia perdemos el control del robot y comienza a realizar movimientos repetitivos (en círculos) esperando una señal.

Es importante recordar que la potencia de la señal es inversamente proporcional a la distancia. Cuanto más lejos esté el emisor del receptor, más baja será la señal

4.1.2 ESCENARIO B: MEDIR LA DISTANCIA DE RECEPCION CON OBSTACULOS (PUERTAS DE VIDRIO)

Figura 4- 2 escenario B

En este segundo escenario tenemos dos puertas cerradas de vidrio como obstáculos como se puede apreciar en la figura 4-2.

Las redes inalámbricas en este caso la señal bluetooth también son susceptibles a ciertas obstrucciones que pueden provocar una señal baja. Lo que ocurre en estos casos, es que la señal se refleja, refracta o es absorbida por la obstrucción. Fuentes comunes de impedimento u obstrucción son: armarios, espejos, vidrios, objetos de

metal, paredes y techos gruesos. En la tabla 4-1 mostramos algunas pérdidas de acuerdo al material

MATERIAL	PERDIDAS
Vidrio en una pared de Ladrillo	2dB
Marco de Metal, Pared de vidrio en interior	6dB
Pared de una oficina	6dB
Puerta metálica empotrada en pared	6dB
Pared con cemento	4dB
Puerta metálica en pared de ladrillo	12.4dB
Pared de Ladrillo junto a puerta de metal	3dB

Tabla 4- 1 tabla de pérdidas de potencia de acuerdo al material

El vidrio no bloquea la señal, pero su índice de refracción de n=1.52 refracta la señal haciendo que exista una pérdida de potencia al llegar al robot, eso se lo pudo comprobar al realizar esta prueba ya que con el obstáculo de dos puertas de vidrio con bordes metálicos observamos que el alcance máximo que tenemos es de 24.4 metros, teniendo el control del robot y logrando hacer la conexión y desconexión a esa distancia. A su vez también logramos distancias menores e iguales a la anterior si llevamos conectado el robot desde nuestra posición.

Figura 4- 3 realización de la prueba en el escenario B

4.1.3 ESCENARIO C: MEDIR LA DISTANCIA DE RECEPCION CON OBSTACULOS (PAREDES DE CEMENTO)

Figura 4- 4 escenario C

En este escenario tenemos un obstáculo el cual corresponde a una pared de cemento y dos paredes de aluminio y vidrio que conforman un cubículo los cuales se encuentran ubicados en la FIEC como lo muestra la figura 4-4.

Calculamos la medida del obstáculo completo y tenemos que es de 3.50 metros.

Según la tabla de pérdidas de acuerdo al material colocada en la tabla 4-1 al tener pared de cemento y dos paredes de aluminio y vidrio tendremos una pérdida de potencia de aproximadamente 18db lo cual incidirá directamente en la distancia de recepción.

La distancia que logramos medir en la cual podemos controlar el robot incluso con este obstáculo es de 10.95 metros, lográndolo conectar y desconectar cuando deseemos.

Figura 4- 5 realización de la prueba en el escenario C

RESUMEN DE RESULTADOS DE LA PRUEBA			
DISTANCIAS	**ESCENARIO A**	**ESCENARIO B**	**ESCENARIO C**
FABRICANTE	20 m	16.76 m	10.97 m
PRUEBA	35.5 m	24.4 m	10.95 m

Tabla 4- 2 resumen de resultados de la prueba de distancia de recepción

4.2 TIEMPO DE RESPUESTA DEL ROBOT

En esta prueba medimos el tiempo que le toma al robot realizar distintas órdenes que le envía el control remoto variando la distancia de separación entre ellos. En el punto 4.1 realizamos pruebas de distancia de recepción entre el control y el robot y obtuvimos un valor de 35.5m como alcance máximo en un ambiente sin obstáculo. Basado en estos datos realizamos pruebas de tiempos de respuesta a 5, 10, 15, 20, 25, 30 y 35 metros de distancia en ambientes sin obstáculos. Para realizar la prueba, evaluamos los tiempos que tardan en ejecutarse los siguientes procesos:

- Tiempo de conexión y desconexión bluetooth entre control remoto y robot
- Tiempo de respuesta del robot al cambio de dirección de movimiento

Para realizar estas pruebas hay que agregar al sistema NIOS II diseñado, un nuevo componente “Timer” desde SOPC BUILDER. Este componente es un contador de 32bits que permite realizar mediciones de tiempos.

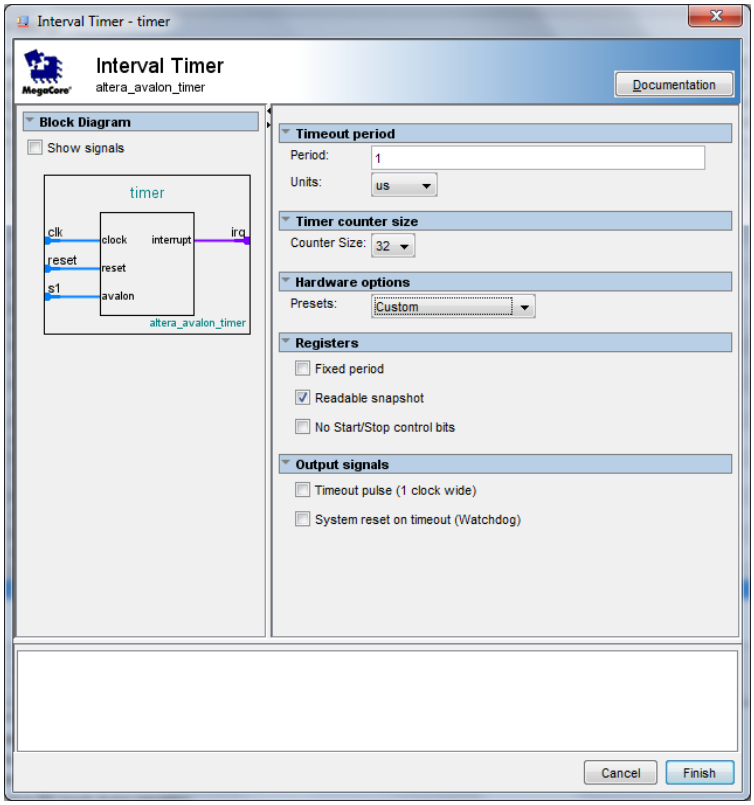


Figura 4- 6 componente "TIMER"

Una vez agregado el Timer al sistema NIOS II, procedemos a configurarlo de acuerdo a la Figura 4-6, asignamos un clock de 50Mhz, generamos la maquina en SOPC BUILDER y volvemos a compilar la nueva configuración de la FPGA de nuestro proyecto en QUARTUS II. Una vez generada la nueva configuración de la FPGA, creamos un nuevo proyecto en NIOS II y generamos las nuevas librerías que permiten tener el control del Timer.

Para realizar las mediciones de tiempo usamos las siguientes funciones que se encuentran en “alt_timestamp.h” de las HAL de NIOS II:

FUNCION	DESCRIPCIÓN
`alt_timestamp()`	Captura el valor del contador al llamar a la función.
`alt_timestamp_start()`	Encera el contador e inicia el conteo.
`alt_timestamp_freq()`	Devuelve la frecuencia a la que trabaja el Timer.

Tabla 4- 3 funciones para mediciones de tiempo

La función **`void Tiempo()`** permite medir el tiempo que tarda una función cualquiera en ejecutar la acción. Dentro de la función **`Tiempo()`** debemos colocar la función a evaluar.

```
void Tiempo(){
      int t1,t2;
      t1=alt_timestamp_start();
      FUNCION(); //FUNCION A EVALUAR
      t2=alt_timestamp();
      printf("Tiempo en seg: %f\n",(float)t2/alt_timestamp_freq());
}
```

4.2.1 TIEMPO DE CONEXIÓN Y DESCONEXIÓN BLUETOOTH

Esta prueba consiste en medir el tiempo que tarda en establecer conexión y desconexión bluetooth entre el control remoto y el robot LEGO variando la distancia entre ellos. La prueba se la realizó en un ambiente sin obstáculos, para garantizar línea de vista.

DISTANCIA (m)	TIEMPO CONEXIÓN (s)	TIEMPO DESCONEXION (s)
5	3.39	0.37
10	3.80	0.36
15	4.87	0.43
20	4.97	0.74
25	5.00	1.34
30	5.25	1.34
35	6.03	1.98

Tabla 4- 4 tiempos de conexión y desconexión según distancia

Como podemos observar los tiempos de conexión aumentan a medida que aumenta la distancia de separación entre ellos, pero los tiempo de desconexión no varían mucho y no son tan altos debido a que el control y el robot ya han establecido una comunicación y solo se envía el comando para iniciar la desconexión, mientras que durante el proceso de establecimiento de conexión los dispositivos inician un proceso de descubrimiento e identificación en la cual intercambian información adicional a los comandos de conexión.

4.2.2 TIEMPO DE RESPUESTA DEL ROBOT AL CAMBIO DE MOVIMIENTO

Esta prueba consiste en enviar al robot un cambio de dirección de movimiento y capturar el tiempo que tarda el robot en ejecutar el cambio. La medición del tiempo se la realiza desde el momento en que se envía el comando **ENVIAR_MOV(70)** desde el control remoto hasta que el robot responde. La función **void GIRO()** realizara de manera automática el proceso de cambio de movimiento del robot de manera infinita y permitirá obtener los valores del tiempo de respuesta a medida que variamos la distancia entre ellos.

```
void GIRO(){
char estado[10];
int t5,t6;
while (1){
      ENVIAR_MOV(250);//Robot empieza a moverse recto con una
                              velocidad del 50%
      usleep(5000000);//esperamos 5 segundos
      t5=alt_timestamp_start();//Inicia medición de respuesta
      ENVIAR_MOV(70);//Robot inicia movimiento hacia la derecha.
      RECIBIR(&estado);//Esperamos a que el robot responda
      if (estado[0]==3)//Verificamos respuesta del robot
            t6=alt_timestamp();//Capturamos el tiempo de respuesta al
                        cambio de movimiento
      else
            printf("error\n");//Muestra error si la respuesta del
                              robot  no es correcta
      printf("Tiempo en s: %f %d\n",(float)
      t6/alt_timestamp_freq(),t6);//mostramos el tiempo en la consola
      de NIOSII
      usleep(3000000);//Esperamos 3 segundos
      ENVIAR_MOV(200);//Detenemos el robot. Velocidad 0%
      usleep(500);//Retardo de 500useg
      ENVIAR_MOV(0);//Dirección del movimiento del robot recto.
      usleep(3000000);//Esperamos 3 segundos y se repite la función
      }
}
```

La tabla 4-5 muestra los tiempos de respuesta obtenidos y se observa que no cambian mucho a medida que aumenta la distancia de separación entre el control y el robot. Esto confirma que el sistema trabaja bien en entornos sin obstáculos permitiendo tener un buen control del robot.

Los tiempos de respuesta los vemos expresados en milisegundos (ms) debido a la capacidad de procesamiento que es limitada por el procesador del robot. La velocidad de procesamiento de la tarjeta DE0 nano es de 100 Mhz mientras que el procesador del robot es un ATMEL de 41 Mhz esto produce retardo en el procesamiento que queda demostrado en la prueba.

DISTANCIA (m)	TIEMPO DE RESPUESTA (s)
5	0.168
10	0.168
15	0.172
20	0.170
25	0.179
30	0.184
35	0.206

Tabla 4- 5 tiempos de respuesta según la distancia

CONCLUSIONES

1. Se logró realizar el acoplamiento del módulo bluetooth con la tarjeta DE0-NANO mediante el uso del puerto GPIO1 – JP2 para conectarlo físicamente a la FPGA embebida en la tarjeta, además se utilizó el protocolo UART para la comunicación entre el procesador NIOS II y el modulo bluetooth RN-42.

2. Para realizar la comunicación entre el control remoto diseñado en la tarjeta DE0-NANO y el robot LEGO MINDSTORM NXT, se implementó un protocolo de comunicación que permite al robot entender los mensajes recibidos, para hacerlo se siguió un formato de mensajes ya establecido que debían contener los paquetes enviados desde el control remoto.

3. Al lograr medir la distancia de recepción del robot LEGO MINDSTORM NXT se observa que cumple perfectamente con los parámetros establecidos por el fabricante, con obstáculos y sin ellos, además se ratifica porque se usó un módulo bluetooth clase II y no uno clase I.

RECOMENDACIONES

1. Se recomienda para futuras mejoras el uso de un robot que nos permita alcanzar mayores distancias de recepción, mayores velocidades y mejor desplazamiento en distintos tipos de suelos; esto sería excelente para aplicaciones de control de robots espías o robots exploradores los cuales necesitan mayor alcance.

2. Se puede adaptar otro tipo de comunicación entre la tarjeta DE0-NANO y el robot que permita obtener mayor alcance, se puede usar módulos de radiofrecuencia o módulos GSM, uno adaptado a la tarjeta y otro adaptado al robot, sea este LEGO o cualquier otro tipo de sistema robótico.

3. Se recomienda colocar una batería en la tarjeta como se lo hizo en el proyecto debido a que nos permite mejor movilidad al momento de controlar el robot.

BIBLIOGRAFÍA

[1] Terasic, DE0-Nano Development and Education Board, http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=139&No=593, fecha de consulta abril 2012

[2] Altera, Cyclone4 Handbook, http://www.altera.com/literature/hb/cyclone-iv/cyclone4-handbook.pdf, fecha de consulta abril 2012

[3] Altera, Nios II Processor Reference Handbook, http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf, fecha de consulta abril 2012

[4] Altera, Nios II Software Developer's Handbook, http://www.altera.com/literature/hb/nios2/n2sw_nii5v2.pdf, fecha de consulta abril 2012

[5] Altera, Documentation: Nios II Processor, http://www.altera.com/literature/lit-nio2.jsp, fecha de consulta junio 2012

[6] Altera, Nios II Hardware Development, http://www.altera.com/support/examples/nios2/exm-hardware-tutorial.html, fecha de consulta junio 2012

[7] Analog Devices, Digital Accelerometer ADXL345, http://www.analog.com/static/imported-files/data_sheets/ADXL345.pdf, fecha de consulta julio 2012

[8] Cursomicros, UART y la Interface RS-232, http://www.cursomicros.com/avr/usart/estandar-rs232.html, fecha de consulta julio 2012

[9] Altera, UART Core - Quartus II 9.1 Handbook, Volume 5, http://www.altera.com.cn/literature/hb/nios2/n2cpu_nii51010.pdf, fecha de consulta julio 2012

[10] Roving Networks, Bluetooth Module RN-42, http://www.rovingnetworks.com/products/RN42, fecha de consulta julio 2012

[11] Microelectronicos, Trabajando con módulos Bluetooth RN-41 y RN-42, http://www.microelectronicos.net/?p=1075, fecha de consulta julio 2012

[12] Lego, Lego Mindstorms NXT Hardware Developer Kit, http://mindstorms.lego.com/en-us/support/files/default.aspx, fecha de consulta mayo 2012

[13] Lego, Lego Mindstorms NXT Bluetooth Developer Kit, http://mindstorms.lego.com/en-us/support/files/default.aspx, fecha de consulta mayo 2012

[14] Lego, Lego Mindstorms NXT User Guide, http://mindstorms.lego.com/en-us/support/files/default.aspx, fecha de consulta mayo 2012

[15] Altera, Introduction to the Quartus II Software, http://www.altera.com/literature/manual/archives/intro_to_quartus2.pdf, fecha de consulta junio 2012

[16] Altera, Quartus II Handbook v12.1.0 Complete Three-Volume Set, http://www.altera.com/literature/lit-qts.jsp, fecha de consulta junio 2012

[17] Altera, Introduction to SOPC Builder, http://www.altera.com/literature/hb/qts/qts_qii54001.pdf, fecha de consulta julio 2012

[18] Altera, SOPC Builder User Guide, http://www.altera.com/literature/ug/ug_sopc_builder.pdf, fecha de consulta julio 2012

[19] Altera, SOPC Builder Support, http://www.altera.com/support/software/system/sopc/sof-sopc_builder.html, fecha de consulta julio 2012

[20] Altera, Embedded Design Handbook, http://www.altera.com/literature/hb/nios2/edh_ed_handbook.pdf, fecha de consulta julio 2012

[21] Altera, Nios II Software Build Tools for Eclipse Support, http://www.altera.com/support/ip/processors/nios2/ide/ips-nios2-ide.html, fecha de consulta agosto 2012

[22] Altera, Nios II Software Developer's Handbook, http://www.altera.com/literature/hb/nios2/n2sw_nii5v2.pdf, fecha de consulta agosto 2012

[23] Altera, Software Development Tools for the Nios II Processor, http://www.altera.com/devices/processor/nios2/tools/ide/ni2-ide.html, fecha de consulta agosto 2012

[24] Rosenberg Neil, NXT Programming For Beginners, http://www.rocwnc.org/Beginning_NXT_Programming_Workshop.pdf, fecha de consulta agosto 2012

[25] Floyd James, LEGO MINDSTORMS NXT-G Programming Guide Second Edition, Apress 2nd Ed, 2010

[26] Griffin Terry, The Art of LEGO MINDSTORMS NXT-G Programming, No Starch Press 1st Ed, 2010

[27] Lego, Appendix 1-LEGO MINDSTORMS NXT Communication Protocol, http://mindstorms.lego.com/en-us/support/files/default.aspx, fecha de consulta julio 2012

[28] Lego, Appendix 2–LEGO MINDSTORMS NXT Direct commands, http://mindstorms.lego.com/en-us/support/files/default.aspx, fecha de consulta julio 2012

[29] Poliakoff Pierre, Communicating with LEGO NXT via Bluetooth in C#, http://www.codeproject.com/Articles/18857/Communicating-with-LEGO-NXT-via-Bluetooth-in-C, fecha de consulta agosto 2012

[30] Roving Networks, Bluetooth RN-42 Command Reference, http://www.rovingnetworks.com/resources/download/47/Advanced_User_Manual, fecha de consulta julio 2012

[31] Altera, Nios II Flash Programmer User Guide, http://www.altera.com/literature/ug/ug_nios2_flash_programmer.pdf, fecha de consulta octubre 2012

[32] Altera, Serial Configuration (EPCS) Devices, http://www.altera.com/literature/hb/cfg/cyc_c51014.pdf, fecha de consulta noviembre 2012

[33] eStuffz, Nios II Flash Programmer on Altera DE0-Nano step-by-step, https://sites.google.com/site/fpgaandco/NiosII-standalone, fecha de consulta noviembre 2012

ANEXOS

ANEXO A – ESQUEMATICO Y PCB DE LA PLACA DEL MODULO BLUETOOTH

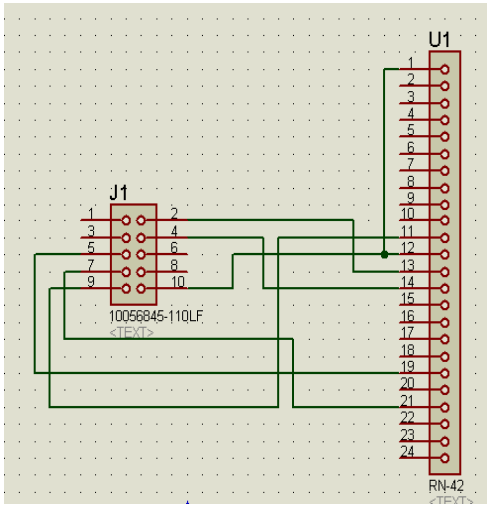


Figura A- 10 diagrama esquemático de la placa del módulo bluetooth

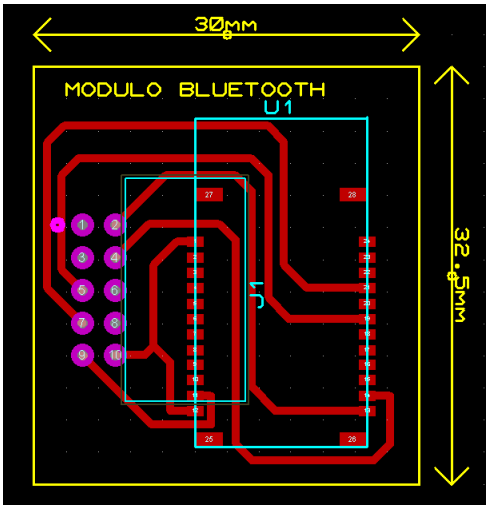
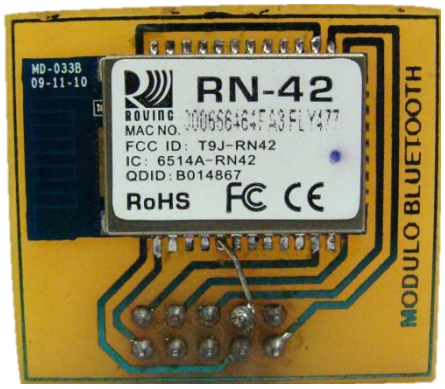
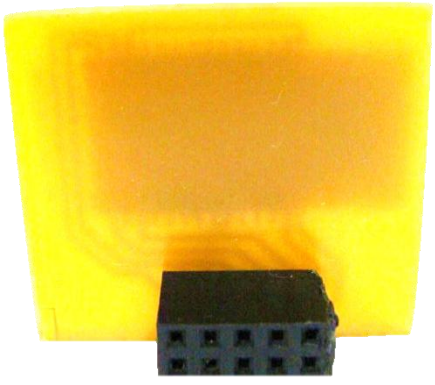


Figura A- 11 PCB de la placa del módulo bluetooth

ANEXO B – MODULO BLUETOOTH SOLDADO EN LA PLACA



a)

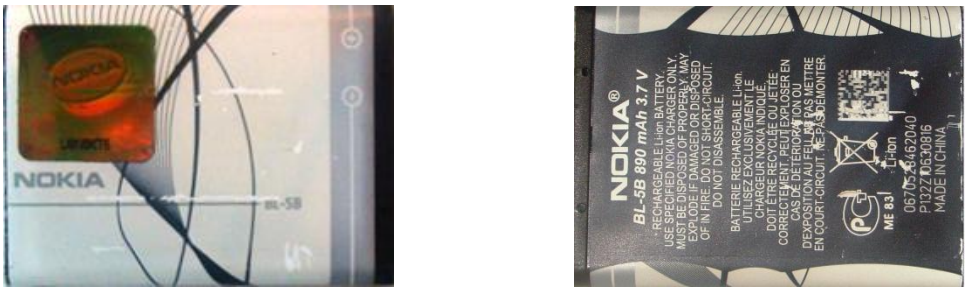


b)

Figura A- 12 a) vista frontal de la placa

b) vista trasera de la placa

ANEXO C – FUENTE DE PODER RECARGABLE



a) **b)**

Figura A- 13 a) vista frontal de la batería **b) vista trasera de la batería**

ANEXO D – PROYECTO TERMINADO

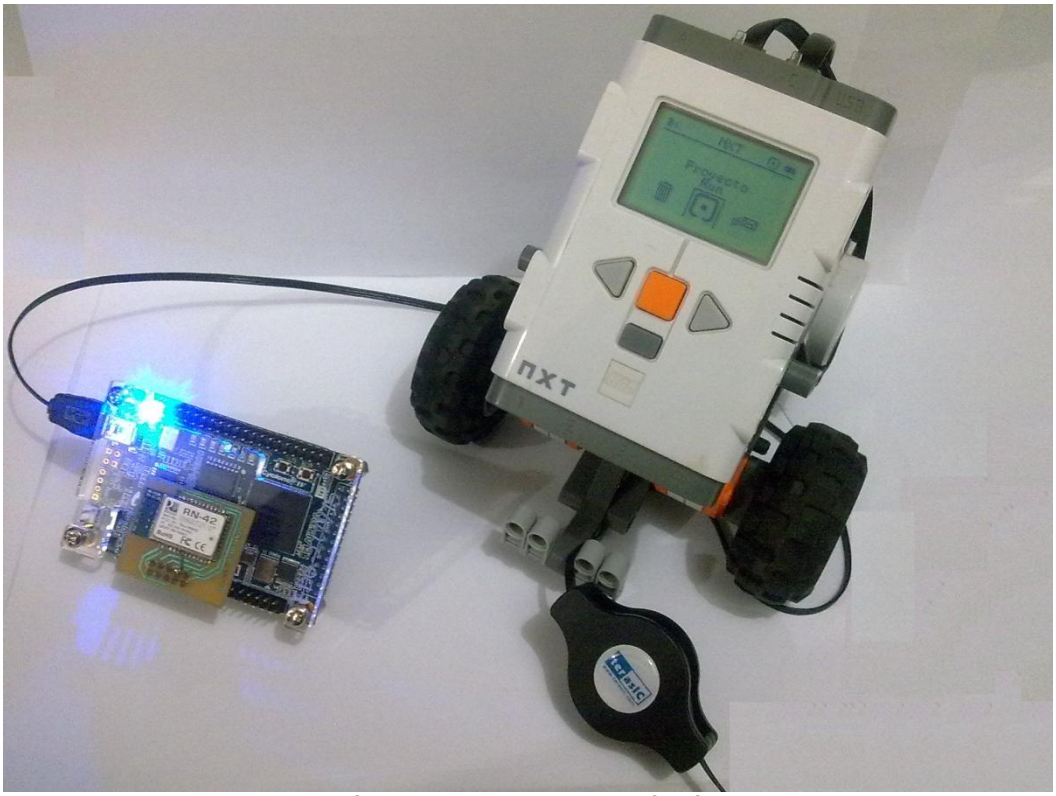


Figura A- 14 proyecto terminado

ANEXO E – GRABACION DEL PROYECTO EN LA MEMORIA FLASH EPCS16

El entorno de desarrollo NIOS II SOFTWARE BUILD TOOL FOR ECLIPSE contiene una herramienta “NIOS II Flash Programmer” que permite almacenar en la memoria flash EPCS16 la configuración de la FPGA y el software que controla el sistema NIOS II. Para poder utilizar el programador se debe tener corriendo en la FPGA el sistema SOPC a grabar con al menos los siguientes componentes:

- Procesador NIOS II con JTAG y módulo debug de nivel 1 o superior
- System ID Peripheral
- Avalon-MM Tristate Bridge
- EPCS Serial Flash Controller

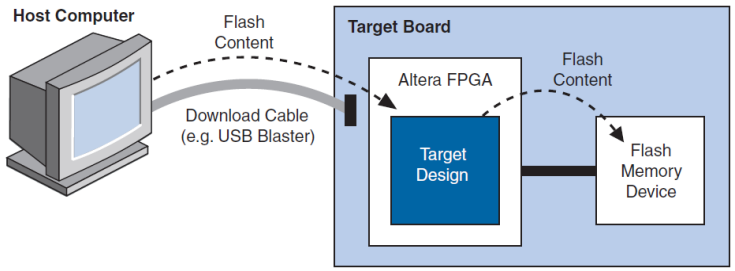


Figura A- 15 modo de operación del programador flash de NIOS II SBT

Estos componentes se los agrega al sistema NIOS II desde SOPC BUILDER en QUARTUS II. El “Avalon-MM Tristate Bridge” no es requerido cuando se

va a programar una memoria EPCS y el “System ID Peripheral” permite al programador flash validar el sistema antes de grabarlo en la memoria flash.

Para que el sistema grabado en la memoria flash inicie correctamente al momento de encender la tarjeta DE0-NANO se debe configurar en SOPC BUILDER el “Reset Vector” y el “Exception Vector” en los parámetros de configuración del procesador NIOS II.

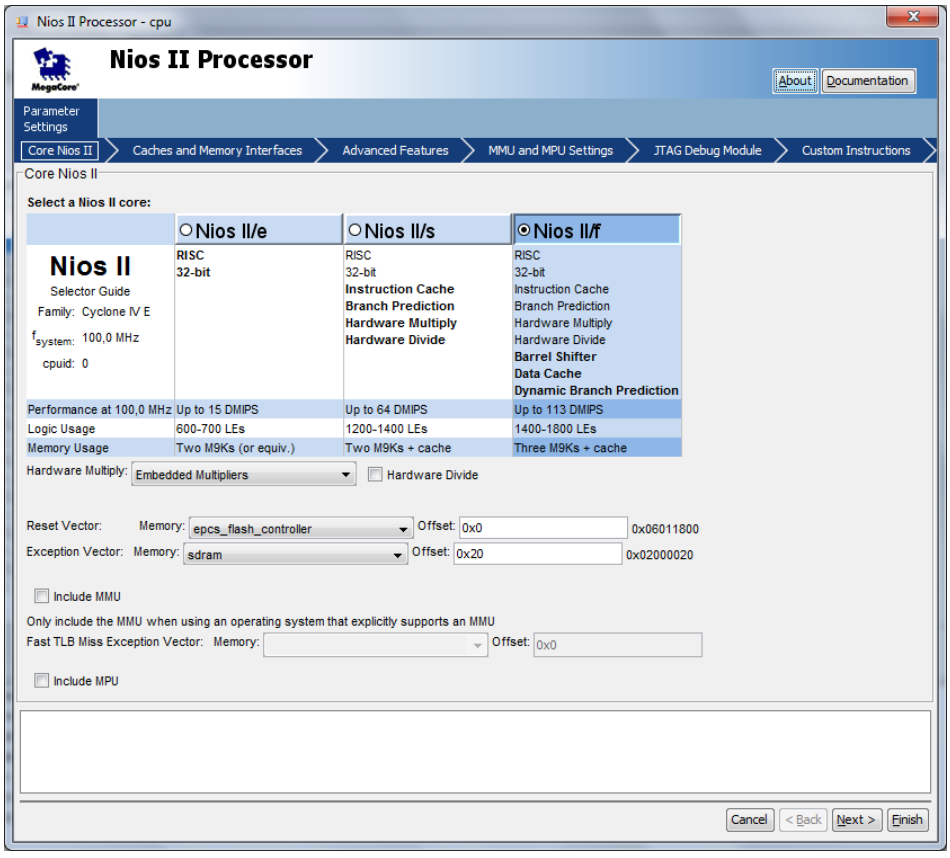


Figura A- 16 configuración del "reset vector" y "exception vector" del procesador NIOS II

Como muestra la Figura A-7 el “Reset Vector” apunta a la memoria flash EPCS16 que es donde esta guardada la configuración de la FPGA y el “Exception Vector” apunta a la memoria SDRAM que es donde se ejecuta el software que controla el sistema NIOS II.

- El “Reset Vector” indica al procesador NIOS II desde que dirección de memoria se debe iniciar a ejecutar software después que se enciende tarjeta o se presiona el botón de Reset.
- El “Exception Vector” indica al procesador NIOS II desde que dirección de memoria se debe ejecutar software cuando ocurre una excepción de software o un evento de interrupción.

Para grabar el sistema NIOS II usamos la memoria flash EPCS16 que se encuentra embebida en la tarjeta DE0-NANO y tiene una capacidad de 16Mbits equivalente a 2MB. Para almacenar correctamente todo el sistema NIOS II tanto hardware como software debemos tener los siguientes archivos:

- Archivo de configuración de la FPGA. “Archivo .sof”
- Archivo del software ejecutable NIOS II. “Archivo .elf”

Desde el NIOS II IDE hacemos clic en el menú NIOS II y seleccionamos NIOS II Flash Programmer, luego hacemos clic en File->New y seleccionamos el archivo “.sopcinfo”. Este archivo contiene información del sistema NIOS II y de la memoria flash EPCS16 que se usa para programar.

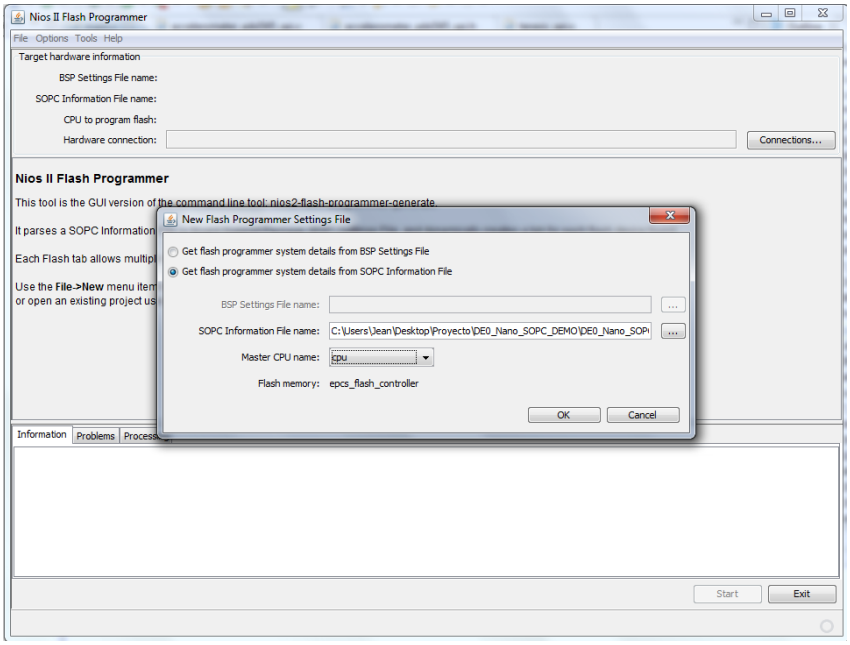


Figura A- 17 información sobre la memoria flash EPCS16

Desde el botón “Connections” seleccionamos el USB-Blaster que representa la conexión de la tarjeta DE0-NANO en la PC, luego agregamos el archivo de configuración de la FPGA “.sof” y el archivo del software que ejecuta el procesador NIOS II “.elf”. Presionamos el botón “Start” e inicia el proceso de grabación en la memoria flash EPCS16.

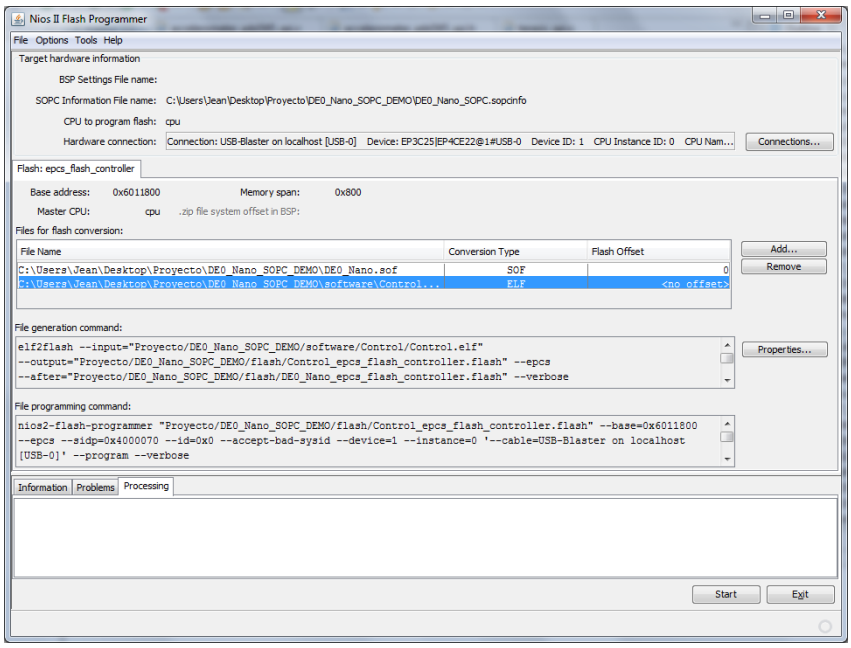


Figura A- 18 Archivos .sof y .elf listos para grabar