SISTEMA PARA ELABORACION DE MAPAS DE NAVEGACION PARA ROBOTS MOVILES UTILIZANDO NIOS II

Ronny Carvajal [1] José Kam [2] Ronald Ponguillo [3]
Facultad de Ingeniería en Electricidad y Computación [1] [2] [3]
Escuela Superior Politécnica del Litoral (ESPOL) [1] [2] [3]
Campus Gustavo Galindo, Km 30.5 vía Perimetral [1] [2] [3]
Apartado 09-01-5863. Guayaquil-Ecuador [1] [2] [3]
ronancar@espol.edu.ec [1] kam@espol.edu.ec [2] rponguil@espol.edu.ec [3]

Resumen

El proyecto consiste en la implementación de un sistema con el programa NIOS II trabajando sobre la tarjeta FPGA, para lo siguiente realizamos la programación en Lenguaje C para elaborar un mapa de navegación móvil mostrada en una interfaz gráfica. El sistema recepta y procesa la información mediante comunicación inalámbrica bluetooth, esta información es enviada por un robot armado con el Kit Mindstorms NXT 2.0 que recorre un entorno cerrado seleccionado por el usuario. El robot mediante el uso de sensores del kit realizara un recorrido sin colisión a través de los diferentes obstáculos ubicados en el ambiente siendo capaz de dimensionar y ubicar la posición de los objetos, esto será mostrado por medio de una interfaz simulando un mapa con coordenadas exactas de cada obstáculo. El sistema almacena la información enviada durante el recorrido del robot que tendrá como limitante un punto de inicio y fin descrito por el usuario, luego de esto el robot mediante una serie de algoritmos tendrá una función autónoma seleccionando apropiadamente el camino para llegar al objetivo dado por el usuario mostrando el mapa del entorno recorrido

.

Palabras Claves: *NIOS II, FPGA.*

Abstract

The project involves the implementation of a system with the NIOS II program working on the FPGA board, we realize the C programming language to develop a mobile navigation map displayed in a graphical interface. The system recept and processes information using Bluetooth wireless communication, this information is sent by a robot armed with Mindstorms NXT 2.0 kit that runs a closed trace selected by the user. The robot by using sensors kit will cycle without collision through the various obstacles located in the environment being able to size and locate the position of objects, this will be shown using an interface simulating a map with exact coordinates of each obstacle. The system save the information received during the course of the robot, who will be limiting a starting point and purpose described by the user, after that the robot through a series of algorithms will have an autonomous function properly selecting the path to reach the target given by user map showing the route environment

Keywords*: NIOS II, FPGA.*

1. Introducción

La robótica se ha posicionado como una de las expresiones de ingeniería más dominantes en el campo de las aplicaciones y prototipos, ya que integra muchas disciplinas que aportan al desarrollo de robots que actualmente perciben e interactúan con el ambiente, si se utilizan robots para interactuar con el entorno, se están integrando los campos de la inteligencia ambiental con la robótica autónoma lo cual da lugar a ingresar en un campo más complejo de investigar que converge en la inteligencia artificial.

Sin lugar a dudas, uno de los pasos imprescindibles para lograr que los robots puedan estar presentes de forma natural en entornos inteligentes, es que adquieran habilidades similares a las humanas. Además de nuestra inteligencia una de las capacidades más interesantes que el ser humano posee es la comprensión del espacio y la posibilidad de realizar cualquier tarea espacial con gran precisión.

Esta capacidad se debe a la constante necesidad de interactuar con el entorno que nos rodea, con éste gran precedente esta capacidad ha evolucionado y se ha perfeccionado a través del tiempo. Por lo tanto, uno de los requerimientos necesarios para cualquier robot que se desee ingresar en entornos humanos es que sea capaz de comprender, interpretar y representar el entorno de manera eficiente y consistente.

2. Hardware

Para efecto de este proyecto se utilizaron cuatro componentes de hardware principales como son la De0-nano, el robot LEGO MINDSTORM NXT, el módulo USB-UART FT232R y el módulo bluetooth RN-42, los cuales detallaremos a continuación cada una de sus características principales.

2.1 Tarjeta de desarrollo De0-nano

La Tarjeta de Desarrollo DE0-Nano diseñada por la empresa Terasic presenta una plataforma FPGA de tamaño compacto adecuado para el desarrollo de diseños de circuitos prototipos tales como robots y proyectos "portátiles".

La placa está estrictamente diseñada para ser utilizada en la aplicación más sencilla posible, dirigida al dispositivo de ciclón IV, cuenta con una colección de interfaces, incluyendo dos GPIO headers externos para extender más allá de los diseños del tablero de0-Nano, contiene dispositivos de memoria SDRAM y EEPROM para mayor almacenamiento de datos y almacenamiento en búfer, así como el uso general de los periféricos como LEDs y pulsadores.

Las ventajas del DE0-Nano son su tamaño y peso, así como su capacidad para ser reconfigurado sin llevar hardware superfluo, para distinguirse de otras tarjetas de desarrollo. Además, para los diseños móviles donde el poder portátil es crucial, el DE0-Nano proporciona a los diseñadores tres opciones de combinación de energía que incluyen un mini-USB AB puerto, 2-pin headers de alimentación externa y dos DC 5V pin

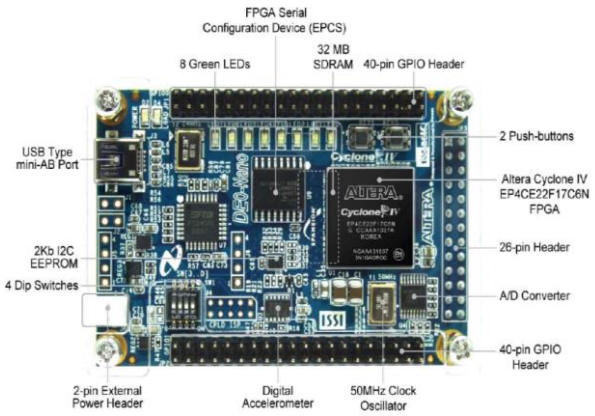


Figura 2.1 Tarjeta DE0-NANO

La tarjeta DE0-NANO está basada 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 embebidos en la tarjeta, permite al usuario controlar todos los componentes usando un sistema NIOS II almacenado en la FPGA Cyclone IV.

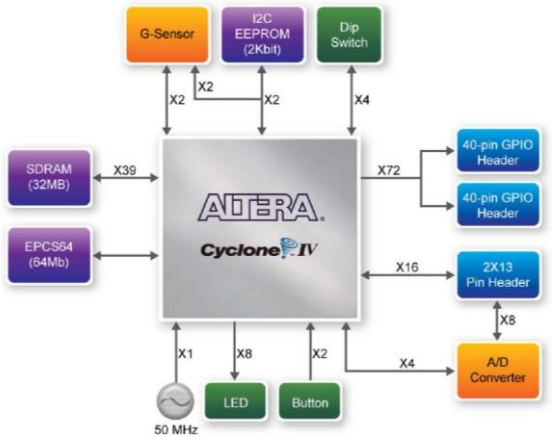


Figura 2.2 Diagrama de bloques de la tarjetaDE0-NANO

La tarjeta DE0-NANO tiene las siguientes características:

- FPGA Altera Cyclone IV EP4CE22 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 para fuente de poder externa 3.6v - 5.7V

2.2 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 Error! No text of specified style in document.**.3** 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.

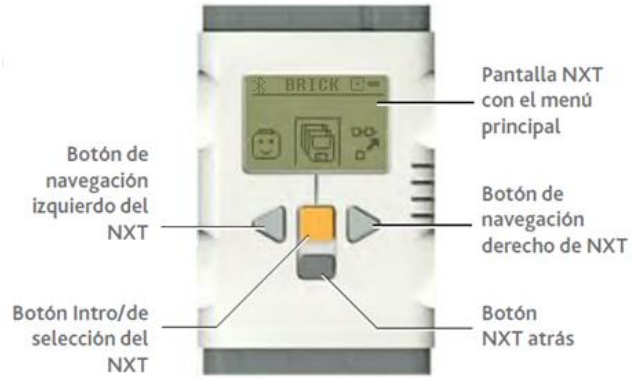


Figura Error! No text of specified style in document.**.4** Bloque NXT

SIMBOLO	DESCRIPCION
	Bluetooth encendido
	Bluetooth visible para otros dispositivos
	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 Error! No text of specified style in document.**.1** Simbología de la interfaz de usuario del bloque NXT

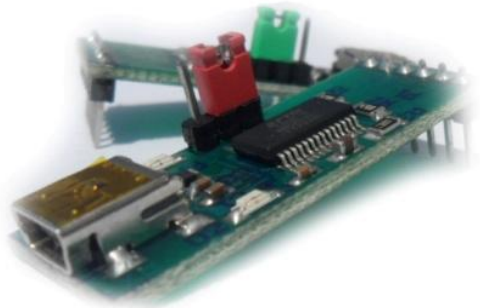
Figura Error! No text of specified style in document.**.5**
Opciones del menú del ROBOT LEGO NXT

Características técnicas:

- Microprocesador Atmel ARM7 de 32 bits 48Mhz
- Memoria FLASH de 256 Kbytes, 64 Kbytes de Memoria RAM
- 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.

2.3 Módulo USB - UART FT232R

Es un módulo de comunicación serial por puerto USB para hacer interface con un computador o PC, permite alimentar circuitos de 5V.



.

Figura Error! No text of specified style in document.**.6**
Modulo FT232R

ESPECIFICACIONES:

- Cable mini –USB.
- Leds indicadores de TX y RX.
- Transferencia de datos 300 Baud hasta 3MBaud.
- Soporta 7 a 8 bits datos, 1 o 2 bits stop, y odd/even/mark/space/no parity.
- Datos serial con amplitud seleccionable de 5V/3.3V.

CARACTERÍSTICAS

- Fuente de Alimentación
- Alimentación por interfaz USB desde la PC.
- Señales de Control:
- VCC: Salida de voltaje 5V
- TX: Transmisión de Datos
- RX: Recepción de Datos
- GND: Tierra
- APLICACIONES:
- Interface de comunicación serial con el computador
- Adquisición y envío de datos desde circuitos con microcontroladores hacia el computador.

2.4 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 Error! No text of specified style in document.**.7**
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
- 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 está 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

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.

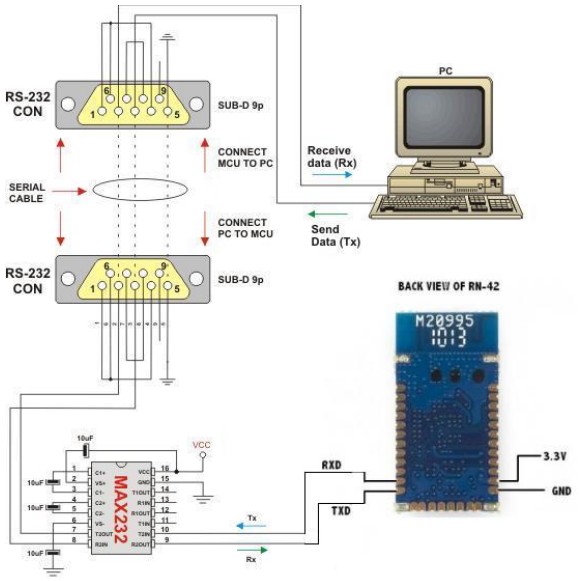


Figura Error! No text of specified style in document.**.8**
Conexión con el PC para la configuración del Módulo Bluetooth RN-42

Una vez realizado el circuito abrimos el AccessPort y seleccionamos el número de puerto COM y elegimos baud rate 115200bps.

Para entrar en el modo de configuración del módulo se debe enviar 3 signos de dólar $$$.

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.

Ingrese los comandos 1-4 que se encuentran en la Tabla 2-1 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 Error! No text of specified style in document.**.2**
Comandos del módulo bluetooth

La Tabla 2-2 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.

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

Tabla Error! No text of specified style in document.**.3**
Pines del módulo bluetooth usados en el proyecto

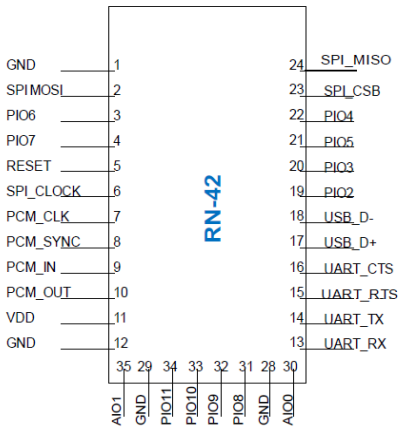


Figura Error! No text of specified style in document.**.9**
Distribucion de pines del módulo bluetooth

3. Software e Implementación

3.1 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.

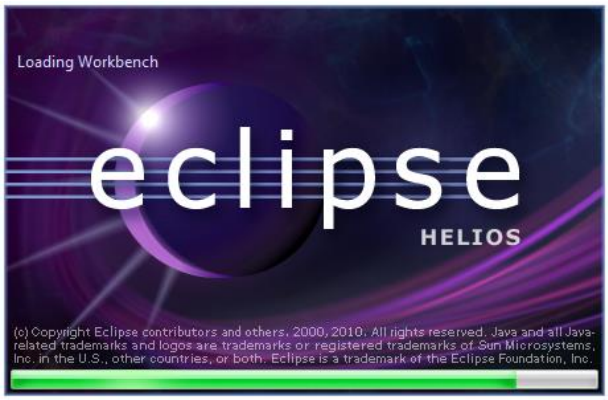


Figura 3.1 Presentación del entorno de desarrollo ECLIPSE

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

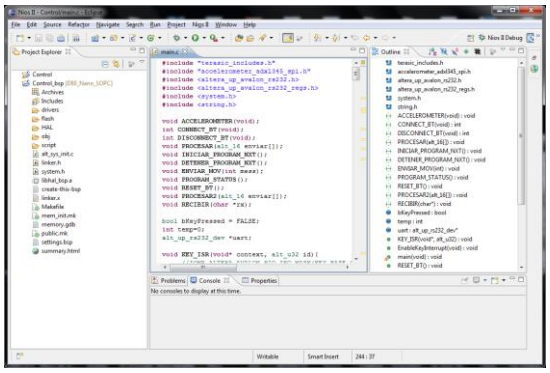


Figura 3.2 Interfaz de usurario de NIOS II SOFTWARE BUILD TOOLS

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.

3.2 NETBEANS IDE

NetBeans IDE es un entorno de desarrollo integrado (IDE), modular, de base estandar (normalizado), escrito en el lenguaje de programación Java.

El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general (framework) para compilar cualquier tipo de aplicación.

Figura 3.3 Presentación del entorno de desarrollo ECLIPSE

Características principales:

- Suele dar soporte a casi todas las novedades en el lenguaje Java. Cualquier preview del lenguaje es rápidamente soportada por Netbeans.

- Asistentes para la creación y configuración de distintos proyectos, incluida la elección de algunos frameworks.
- Buen editor de código, multilenguaje, con el habitual coloreado y sugerencias de código, acceso a clases pinchando en el código, control de versiones, localización de ubicación de la clase actual, comprobaciones sintácticas y semánticas, plantillas de código, coding tips, herramientas de refactorización,… y un largo etcétera. También hay tecnologías donde podemos usar el pulsar y arrastrar para incluir componentes en nuestro código.

3.3 Implementación

Para controlar el ROBOT LEGO MINDSTORM NXT con la FPGA debemos crear dentro de ella una mini-computadora con la herramienta Qsys la cual nos permite añadir una serie de librerías de componentes requeridos para nuestro proyecto.

Los componentes básicos requeridos para un correcto funcionamiento de nuestra mini-computadora son:

- **System ID Peripheral:** Es uno de los periféricos más importantes que debe contener el sistema, ya que permite a la herramienta de software de NIOS II saber si la aplicación desarrollada pertenece al hardware previamente configurado. Es esencial que solo exista un solo periférico de este tipo en cada sistema.

- **NIOS II Processor:** Encargado de ser el cerebro para todo el sistema, ya que todo el software será interpretado por él y este a su vez enviara las señales necesarias y correspondientes a cada periférico para su correcto funcionamiento.

- **On-Chip Memory:** Esta es una memoria que se implementará en la FPGA con la cual almacenaremos el software del sistema y también todos los datos necesarios en el programa.

- **JTAG UART:** Este periférico nos permite establecer la comunicación, es esencial incluir este módulo ya que de no ser así no se podría realizar ningún cambio a la configuración del chip Cyclone IV.
- **SDRAM Controller:** Este módulo nos ayuda a acceder y controlar la memoria SDRAM externa incluida en la tarjeta DE0-Nano, la cual tiene la capacidad de almacenar 32Mbytes de datos.

- **Interval Timer:** Este módulo nos ayudara a medir intervalos de tiempo necesarios para nuestro programa.

- **PIO (Parallel I/O):** Es de gran importancia ya que es mediante este módulo que se obtienen los datos del exterior para poder ser procesados. De igual manera se entregan datos al exterior para el control de otros dispositivos acoplados al sistema o proporcionar información útil al usuario.

- **RS232 UART:** Es de gran importancia ya que es mediante este módulo que se realizara la comunicación entre nuestro sistema embebido u el bloque NXT y además la comunicación con nuestra aplicación JAVA.

Luego de haber colocado cada módulo y realizar las conexiones necesarias entre los distintos módulos, se muestra en la Figura:

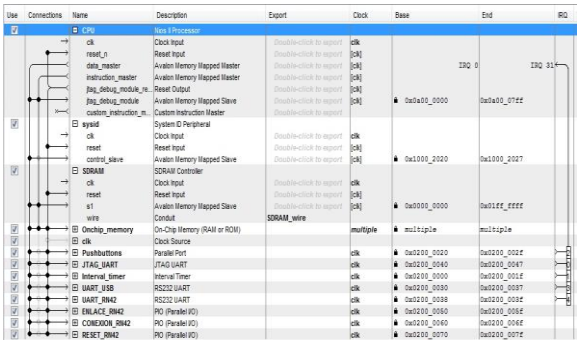


Figura Error! No text of specified style in document.**.3** Diseño e interconexión de componentes para la computadora básica

Luego de haber creado la mini-computadora en Qsys, procedemos a observar el bloque creado de nuestro sistema utilizando Quartus II.

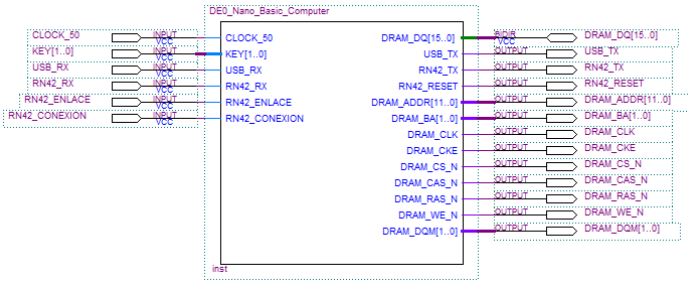


Figura Error! No text of specified style in document.**.4** Diagrama esquematico de la computadora básica

Luego en el mismo programa Quartus II le damos a la opción de PIN PLANNER el cual nos permitirá asignar los pines para el correcto funcionamiento de cada uno de los módulos y periféricos que conforman la mini-computadora.

Primero asignamos los pines para usar el reloj a 50 Mhz y también el KEY0 de la De0-Nano que servirá como un reset. (Tabla 3.1)

Node Name	Location
CLOCK_50	PIN_R8
KEY 0	PIN_J15

Tabla Error! No text of specified style in document..**1** Configuración de pines para las señales globales

Además también añadimos los pines utilizados al establecer la comunicación entre el robot LEGO NXT MINDSTORM y el módulo bluetooth RN-42. (Tabla 3.3).

RN-42	
Node Name	Location
RN42_RX	PIN_L15
RN42_TX	PIN_K16
RN42_CONEXION	PIN_P15
RN42_ENLACE	PIN_R14
RN42_RESET	PIN_R16

Tabla Error! No text of specified style in document..**2** Configuración de pines para el Módulo de Comunicación Bluetooth

Y por último agregamos los pines utilizados para la comunicación SERIAL-USB.

SERIAL-USB	
Node Name	Location
USB_RX	PIN_A4
USB_TX	PIN_B5

Tabla Error! No text of specified style in document..**3** Configuración de pines para el Módulo de Comunicación Serial

3.4 Código Del Programa Principal

Empezamos con las librerías necesarias para utilizar cada componente requerido.

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include "system.h"
#include "altera_up_avalon_rs232.h"
#include "altera_up_avalon_rs232_regs.h"
```

Figura Error! No text of specified style in document..**5** Librerias necesaria para cada componente

Ahora hablaremos brevemente de cada función creada para el funcionamiento del programa

```
void RN42_Enviar(char *data, int n_data);
void RN42_Recibir();
void RN42_Decodificar();
void USB_Enviar(char *data, int n_data);
void USB_Recibir();
void Leer_Sensor1();
void Leer_Sensor2();
void Iniciar_Tablero();
void Comando(int Fila, int Columna, int Estado);
```

Figura Error! No text of specified style in document..**6** Prototipos de funciones a usar en el programa

RN42_Enviar: Esta función nos permite enviar datos desde el módulo bluetooth al robot LEGO MINDSTORM NXT

```
void RN42_Enviar(char *data, int n_data){
    int index;
    for(index=0 ; index<n_data ; index++)
        alt_up_rs232_write_data(RN42, data[index]);
}
```

Figura Error! No text of specified style in document..**7** Contenido de la función para enviar datos mediante el modulo RN42

RN42_Recibir: Esta función se encarga de recibir los mensajes enviados por el robot LEGO MINDSTORM NXT

```
void RN42_Recibir(){
    while(IORD_ALT_UP_RS232_RAVAIL(UART_RN42_BASE)==0);
    // Capturamos respuesta
    printf("Recibiendo respuesta\n");
    index_rn42=0;
    while(IORD_ALT_UP_RS232_RAVAIL(UART_RN42_BASE)!=0){
        string_rn42[index_rn42] = (int)IORD_ALT_UP_RS232_DATA(UART_RN42_BASE);
        usleep(10000);
        index_rn42++;
    }
    string_rn42[index_rn42] = '\0';
}
```

Figura Error! No text of specified style in document..**8** Contenido de la función para recibir datos mediante el modulo RN42

RN42_Decodificar: Esta función se encarga de interpretar el mensaje enviado por el robot LEGO MINDSTORM NXT para que así nuestro módulo bluetooth sepa que hacer.

```
void RN42_Decodificar(){
    if(string_rn42[4]==-64)
        printf("Respuesta: ERROR\n");
    else if(string_rn42[4]==0)
        printf("Respuesta: OK\n");
    else
        printf("Respuesta: %s",string_rn42);
}
```

Figura Error! No text of specified style in document..**9** Contenido de la función para interpretar datos recibidos del modulo RN42

USB_Enviar: Esta función se encarga de enviar datos mediante el convertidor USB-Serial desde la FPGA a la interfaz en JAVA.

```
void USB_Enviar(char *data, int n_data){
    int index;
    for(index=0 ; index<n_data ; index++){
        alt_up_rs232_write_data(USB, data[index]);
        usleep(20000);
    }
}
```

Figura Error! No text of specified style in document..**10** Contenido de la función para enviar datos mediante el modulo FT232R

USB_Recibir: Esta función se encarga de recibir los mensajes enviados por la interfaz JAVA y transmitirlo a la FPGA

```
void USB_Recibir(){
    while(IORD_ALT_UP_RS232_RAVAIL(UART_USB_BASE)==0);
    // Capturamos respuesta
    printf("Recibiendo respuesta\n");
    index_usb=0;
    while(IORD_ALT_UP_RS232_RAVAIL(UART_USB_BASE)!=0){
        string_usb[index_usb] = (char)IORD_ALT_UP_RS232_DATA(UART_USB_BASE);
        usleep(20000);
        index_usb++;
    }
    string_usb[index_usb] = '\0';
    printf("Respuesta: %s",string_usb);
}
```

Figura Error! No text of specified style in document..**11** Contenido de la función para recibir datos mediante el modulo FT232R

Leer_Sensor1: Esta función es la encargada de recibir la información del sensor ultrasónico frontal del robot, utilizando algunas funciones anteriormente descritas, genera un pulso por el cual espera y procede a realizar un cálculo de la distancia a la cual se encuentra un obstáculo.

```
void Leer_Sensor1(){
    printf("Ordenando lectura en Sensor1\n");
    RN42_Enviar(ORDER_SENSOR1,9);
    RN42_Recibir();
    RN42_Decodificar();
    printf("Esperando calculos\n");
    RN42_Enviar(STATUS_SENSOR1,5);
    RN42_Recibir();
    RN42_Decodificar();
    printf("Leyendo Valor\n");
    RN42_Enviar(VALUE_SENSOR1,5);
    RN42_Recibir();
    RN42_Decodificar();
    Lectura_Sensor1 = (int)string_rn42[6];
    if(Lectura_Sensor1 < 0)
        Lectura_Sensor1 = 255;
}
```

Figura Error! No text of specified style in document..**12** Descripcion del proceso de lectura del sensor 1

Leer_Sensor2: Esta función realiza lo mismo que la anterior descrita pero lo hace con el segundo sensor (lateral) que posee nuestro robot.

```
void Leer_Sensor2(){
    printf("Ordenando lectura en Sensor2\n");
    RN42_Enviar(ORDER_SENSOR2,9);
    RN42_Recibir();
    RN42_Decodificar();
    printf("Esperando calculos\n");
    RN42_Enviar(STATUS_SENSOR2,5);
    RN42_Recibir();
    RN42_Decodificar();
    printf("Leyendo Valor\n");
    RN42_Enviar(VALUE_SENSOR2,5);
    RN42_Recibir();
    RN42_Decodificar();
    Lectura_Sensor2 = (int)string_rn42[6];
    if(Lectura_Sensor2 <0)
        Lectura_Sensor2 = 255;
}
```

Figura Error! No text of specified style in document..**13** Descripcion del proceso de lectura del sensor 2

Iniciar_Tablero: esta función se encarga de dibujar las dimensiones del tablero virtual en JAVA y de ir pintando cada cuadro de acuerdo como vaya indicando el robot.

```
void Iniciar_Tablero(){
    int i,j;
    for(i=0;i<DIM;i++){
        for(j=0;j<DIM;j++){
            Tablero[i][j]=-1;
        }
    }
    Tablero[DIM-1][DIM-1]=VACIO;
    Fila=DIM-1;
    Columna=DIM-1;
    Direccion=NORTE;
}
```

Figura Error! No text of specified style in document..**14** Descripcion del proceso de inicialización del tablero

Comando: Esta función al comunicar el robot con JAVA permite conocer tres valores necesarios para marcar en el tablero virtual los cuales son la fila, columna y el estado en que se encuentra ubicado nuestro robot.

```
void Comando(int Fila, int Columna, int Estado){
    CMD_JAVA[1] = Fila;
    CMD_JAVA[2] = Columna;
    CMD_JAVA[3] = Estado;
    USB_Enviar(CMD_JAVA,4);
    Tablero[Fila][Columna]=Estado;
}
```

Figura Error! No text of specified style in document.**.15**
Descripcion de la trama de envio a JAVA

Para poder comunicar la FPGA con el robot LEGO MINDSTORM NXT tuvimos que recurrir al lenguaje utilizado por el robot, empezando con una frecuencia de 115200 y luego la siguiente secuencia de funciones obtenidas del primer manual de la primera versión creada del robot, el cual lo puede encontrar en la sección de anexos.

```
char CMD_MODE [] = {'$','$','$'};
char CONECTAR_NXT [] = {'C',13,10};
char CONFIG_SENSOR1 [] = {5,0,0,5,0,11,0};
char ORDER_SENSOR1 [] = {7,0,0,15,0,2,1,2,66};
char STATUS_SENSOR1 [] = {3,0,0,14,0};
char VALUE_SENSOR1 [] = {3,0,0,16,0};
char CONFIG_SENSOR2 [] = {5,0,0,5,3,11,0};
char ORDER_SENSOR2 [] = {7,0,0,15,3,2,1,2,66};
char STATUS_SENSOR2 [] = {3,0,0,14,3};
char VALUE_SENSOR2 [] = {3,0,0,16,3};
char MOVE_NXT [] = {13,0,128,4,255,15,7,1,0,32,0,0,0,0,0};
char STOP_NXT [] = {13,0,128,4,255,0,0,0,0,0,0,0,0,0,0};
char RIGHT1_NXT [] = {13,0,128,4,2,15,7,1,0,32,0,0,0,0,0};
char RIGHT2_NXT [] = {13,0,128,4,1,-15,7,1,0,32,0,0,0,0,0};
char LEFT1_NXT [] = {13,0,128,4,1,15,7,1,0,32,0,0,0,0,0};
char LEFT2_NXT [] = {13,0,128,4,2,-15,7,1,0,32,0,0,0,0,0};
char CMD_JAVA [] = {'*',0,0,0};
int Tablero[DIM][DIM];
int Finalizar=0;
```

Figura Error! No text of specified style in document.**.16**
Codigos usados para poder enviar ordenes al bloque NXT mediante bluetooth

Explicadas todas las funciones del programa procedemos con el "main" en el cual lo primero que realiza es dar un reset al módulo bluetooth. Luego de esto se ingresa al modo comando enviando ($,$,$).

```
int main(void){

    // Archivos para comunicacion serial
    RN42 = alt_up_rs232_open_dev(UART_RN42_NAME);
    USB = alt_up_rs232_open_dev(UART_USB_NAME);

    do{
        // Damos Reset al modulo RN42
        printf("Reset RN42\n");
        IOWR(RESET_RN42_BASE,0,0);
        usleep(250000);
        IOWR(RESET_RN42_BASE,0,1);
        usleep(1000000);
        // Enviamos codigo para entrar en Modo-Comando y esperamos p
        printf("Ingresando al Modo Comando\n");
        usleep(100000);
        RN42_Enviar(CMD_MODE,3);
        RN42_Recibir();
        RN42_Decodificar();
        usleep(100000);
        if(strcmp(string_rn42,"CMD\r\n")!=0)
            printf("Estado: Error al ingresar al Modo Comando\n");
        else
            printf("Estado: Ingreso al Modo Comando Exitoso\n");
    // Se espera la secuencia CMD
    }while(strcmp(string_rn42,"CMD\r\n")!=0);
    usleep(1000000);
```

Figura Error! No text of specified style in document.**.17**
Proceso inicial de ingreso al modo comando en RN42

Luego de un ingreso exitoso al modo comando, procedemos a realiza la conexión entre el De0-NANO con el NXT.

```
// Ordenamos la Conexion entre DE0-NANO y NXT
printf("Iniciando Conexion DE0-NANO <<-->> NXT\n");
usleep(100000);
RN42_Enviar(CONECTAR_NXT,3);
RN42_Recibir();
RN42_Decodificar();
usleep(100000);
// Esperamos a que se establesca la conexion
while(IORD(CONEXION_RN42_BASE,0)==0);
printf("Conexion DE0-NANO <<-->> NXT establecida...\n");
usleep(1000000);
```

Figura Error! No text of specified style in document.**.18**
Proceso de conexión DE0-NANO con bloque NXT

Una vez establecida aquella conexión, procedemos a realizar la conexión entre DE0-NANO con la interfaz gráfica JAVA.

```
do{ // Ordenamos la Conexion entre DE0-NANO y JAVA
    printf("Iniciando Conexion DE0-NANO <<-->> JAVA\n");
    usleep(100000);
    //do{   // Ordenamos la Conexion entre DE0-NANO y JAVA
        USB_Enviar("*JAVA?",6);
    //}while(IORD_ALT_UP_RS232_RAVAIL(UART_USB_BASE)!=1);
    // Capturamos respuesta desde JAVA
    USB_Recibir();
    usleep(100000);
    if(strcmp(string_usb,"JAVA_OK\n")!=0)
        printf("Estado: Error al conectar con JAVA\n");
    else
        printf("Estado: Conexion con JAVA Exitosa\n");
}while(strcmp(string_usb,"JAVA_OK\n")!=0);
printf("Conexion DE0-NANO <<-->> JAVA establecida...\n");
usleep(1000000);
```

Figura Error! No text of specified style in document.**.19**
Proceso de conexión entre DE0-NANO y la aplicación JAVA

Iniciamos el NXT con una primera lectura de sus sensores.

```
// Enviamos comando para iniciar el programa NXT
printf("Iniciando Programa NXT\n");
USB_Enviar("*NXT_RUN",8);
// Inicializamos el Tablero para empezar recorrido
printf("Estableciendo Condiciones Iniciales\n");
Iniciar_Tablero();
// Configuracion de Sensor en Puerto1
printf("Configurando Sensor en Puerto1\n");
RN42_Enviar(CONFIG_SENSOR1,7);
RN42_Recibir();
RN42_Decodificar();
// Configuracion de Sensor en Puerto4
printf("Configurando Sensor en Puerto4\n");
RN42_Enviar(CONFIG_SENSOR2,7);
RN42_Recibir();
RN42_Decodificar();
usleep(1000000);

Leer_Sensor1();
Leer_Sensor2();
```

Figura Error! No text of specified style in document.**.20**
Envio de señales para proceso inicial

Luego de que todas las conexiones se hayan establecidos correctamente, preguntamos por los sensores, y sus cálculos de distancia, respecto con aquellos datos se decide si se avanza o se gira.

```
// Lazo infinito para lectura de datos NXT
Direccion = NORTE;
while(Finalizar!=1){
    usleep(20000);
    Comando(Fila,Columna,VACIO);
    Leer_Sensor1();
    Leer_Sensor2();
    printf("Distancia Frontal : %dcm\n", Lectura_Sensor1);
    printf("Distancia Lateral : %dcm\n", Lectura_Sensor2);
    // Preguntamos si se puede Girar a la Derecha
    if(Lectura_Sensor2>30){
        RN42_Enviar(RIGHT1_NXT,15);
        RN42_Enviar(RIGHT2_NXT,15);
        usleep(501200);
        RN42_Enviar(MOVE_NXT,15);
        usleep(1680000);
        RN42_Enviar(STOP_NXT,15);
        switch(Direccion){
            case NORTE: Direccion = ESTE; Columna++; break;
            case SUR: Direccion = OESTE; Columna--; break;
            case ESTE: Direccion = SUR; Fila++; break;
            case OESTE: Direccion = NORTE; Fila--; break;
        }
        Tablero[Fila][Columna] = 1;
    }
```

Figura Error! No text of specified style in document.**.21**
Proceso de adquisición y envio de datos hasta realizar el recorrido completo del entorno

```
// Preguntamos si se puede Avanzar
else if(Lectura_Sensor1>30){
        if(Direccion==SUR && Columna==DIM-1 && Fila+1==DIM-1){
            Finalizar=1;
            break;
        }
        if(Direccion==ESTE && Columna+1==DIM-1 && Fila==DIM-1){
            Finalizar=1;
            break;
        }
        RN42_Enviar(MOVE_NXT,15);
        usleep(1680000);
        RN42_Enviar(STOP_NXT,15);
        switch(Direccion){
            case NORTE: Fila--;
                        if(Columna > 0 && Columna < DIM-1)
                            Comando(Fila,Columna+1,OCUPADO);
                        break;
            case SUR:   Fila++;
                        if(Columna > 0 && Columna < DIM-1)
                            Comando(Fila,Columna-1,OCUPADO);
                        break;
            case ESTE:  Columna++;
                        if(Fila > 0 && Fila < DIM-1)
                            Comando(Fila+1,Columna,OCUPADO);
                        break;
            case OESTE: Columna--;
                        if(Fila > 0 && Fila < DIM-1)
                            Comando(Fila-1,Columna,OCUPADO);
                        break;
        }
}
```

Figura Error! No text of specified style in document.**.22**
Proceso en caso de poder avanzar

Si no se puede avanzar, se buscará girar hacia la izquierda primeramente, ya que el robot siempre bordeará el tablero.

```
    // Ultima Opcion Girar a la Izquierda
    else{
        RN42_Enviar(LEFT1_NXT,15);
        RN42_Enviar(LEFT2_NXT,15);
        usleep(480000);
        RN42_Enviar(STOP_NXT,15);
        switch(Direccion){
            case NORTE: Direccion = OESTE;
                        if(Fila > 0)
                            Comando(Fila-1,Columna,OCUPADO);
                        break;
            case SUR:   Direccion = ESTE;
                        if(Fila < DIM-1)
                            Comando(Fila+1,Columna,OCUPADO);
                        break;
            case ESTE:  Direccion = NORTE;
                        if(Columna < DIM-1)
                            Comando(Fila,Columna+1,OCUPADO);
                        break;
            case OESTE: Direccion = SUR;
                        if(Columna > 0)
                            Comando(Fila,Columna-1,OCUPADO);
                        break;
        }
    }
}
```

Figura Error! No text of specified style in document.**.23**
Proceso a seguir en caso de no poder avanzar y poder girar a la izquierda

Este proceso se repite durante todo el recorrido, pero si el siguiente cuadro es aquel de donde empezó a moverse el robot, este automáticamente se detendrá, indicando que ha llegado al final de su recorrido.

```
printf("Final de Recorrido \n");
usleep(1000000);
//USB_Enviar("*Z",2);

return 0;
```

Figura Error! No text of specified style in document.**.24**
Muestra del proceso final de mapeo del entorno

4. Funcionamiento y Resultados

4.1 Análisis de Distancia

Los sensores ultrasónicos juegan un papel fundamental en nuestro proyecto debido a eso le realizamos unas pruebas de calibración para observar su comportamiento, pudiendo realizar las siguientes adquisiciones.

Figura Error! No text of specified style in document..**25**
Toma de medidas para el sensor ultrasonico

PRUEBA SENSOR ULTRASONICO				
DISTANCIA	SENSOR1	ERROR	SENSOR2	ERROR
7	7	0,00%	7	0,00%
10	10	0,00%	10	0,00%
14	14	0,00%	14	0,00%
21	20	4,76%	20	4,76%
30	28	6,67%	27	10,00%
34	31	8,82%	32	5,88%

Tabla Error! No text of specified style in document..**2**
Resumen de mediciones teóricas y experimentales

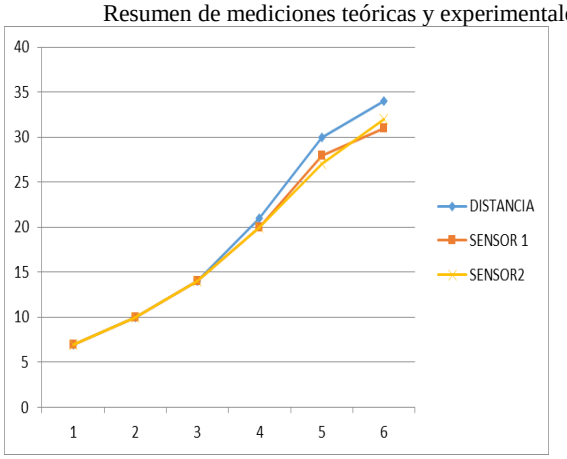


Figura Error! No text of specified style in document..**26**
Estadistica de mediciones con el sensor

4.2 Muestreo

En entornos interiores, uno de los problemas habituales que afrontan los robots es mantenerse localizados. Este proyecto maneja dos frentes de percepción (frontal y lateral derecho) para lo cual necesitamos giros precisos de 90 grados que varían los cuatro estados de avance (norte, sur, este, oeste) las ruedas y la superficie por la que se desplaza puede modificar el correcto funcionamiento del recorrido.

Se Realizan 3 pruebas del funcionamiento completo del sistema. Donde pondremos a prueba la orientación, capacidad de percepción y evasión de obstáculos el color blanco indica una casilla vacía por la que puede circular un robot con características similares, el cuadro en color negro localiza un obstáculo en la posición exacta donde fue ubicado que impide el transito del robot. Creamos un espacio físico cuadrado de 150x150 cm con división cuadricular de 30x30cm y obstáculos de similares características. Estas medidas se muestran a escala con el programa en JAVA.

PRUEBA 1.

Figura Error! No text of specified style in document..**27**
Resultado de recorrido de entorno 1

El primer entorno muestra un recorrido que implica un retorno por casillas marcadas poniendo a prueba la

orientación del robot, la parte superior derecha muestra un desafío a la detección y cambio constante de orientación.

PRUEBA 2.

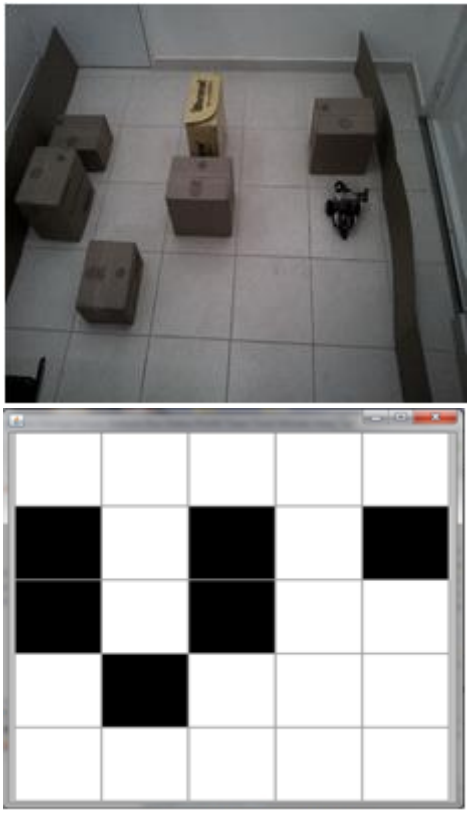


Figura Error! No text of specified style in document..**28**
Resultado de recorrido de entorno 2

Variando la ubicación de los obstáculos con un ambiente más desafiante tratamos de que el robot pierda orientación. Hay que recordar que mientras más giros realice el robot se pueden cometer errores que dejarían sin validez el mapa.

PRUEBA 3.

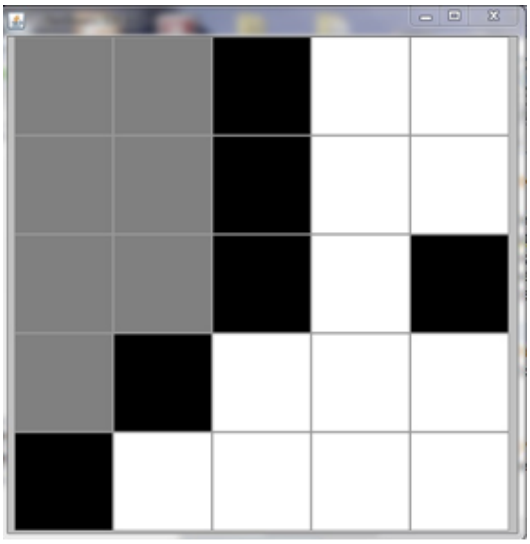


Figura Error! No text of specified style in document..**29**
Resultado de recorrido de entorno 3

Ahora ponemos a prueba las capacidades de la aplicación Java montando un ambiente que solo represente aproximadamente el 50% del área total. Como se muestra en la figura las casillas grises representan el área no explorada

4.3 Tiempo de Respuesta

Para esta prueba tomaremos el tiempo que ocupa nuestro programa en NIOS II con los distintos periféricos tales como: entrar al modo comando del módulo bluetooth RN42, el tiempo de conexión entre nuestra DE0-NANO y con el robot LEGO MINDSTORM NXT, el tiempo que le toma al programa configurar automáticamente ambos sensores ultrasónicos y por último el tiempo de lectura que ocupa los sensores desde el momento en que emite el pulso hasta que procesa el dato recibido.

Para las pruebas realizadas tomamos 6 medidas y tomamos los datos de cada una de ellas a 1, 2, 5, 10, 15, 20 metros de distancia en ambientes abiertos.

Para obtener los tiempos en nuestro programa, tuvimos que crear dos nuevas función, la cual no está descrita en el capítulo anterior, ya que solo lo utilizamos para tomar los tiempos en este capítulo de prueba. Estas nuevas funciones son **void Iniciar_Timer**(); y **void Detener_Timer**();

```
void Iniciar_Timer(){

    IOWR(INTERVAL_TIMER_BASE,2,0xFFFF);
    IOWR(INTERVAL_TIMER_BASE,3,0xFFFF);
    IOWR(INTERVAL_TIMER_BASE,1,4);
}
```

Figura Error! No text of specified style in document..**6**
Funcion Iniciar_Timer

```
void Detener_Timer(){
    long Parte_baja, Parte_alta, Numero_Flancos;

    IOWR(INTERVAL_TIMER_BASE,1,8);      //STOP COUNTER
    IOWR(INTERVAL_TIMER_BASE,4,0);      //CAPTURA PARTE BAJA
    IOWR(INTERVAL_TIMER_BASE,5,0);      //CAPTURA PARTE ALTA
    Parte_baja = 0xFFFF & IORD(INTERVAL_TIMER_BASE,4);
    Parte_alta = 0xFFFF & IORD(INTERVAL_TIMER_BASE,5);
    Numero_Flancos = 0xFFFFFFFF & ((Parte_alta<<16) + Parte_baja);
    Contador_useg = ((0xFFFFFFFF - Numero_Flancos)/50);
}
```

Figura Error! No text of specified style in document..**7**
Funcion Detener_Timer

Lo único que se debe realizar con estas funciones es colocarla entre las otras funciones de las cuales deseamos obtener el tiempo de ejecución. Y como resultado nos muestra esto dentro del programa NIOS II.

Los mensajes mostrados al ejecutar nuestro código en NIOS II sobre el ingreso al modo Comando del módulo bluetooth y el tiempo de 0.1475 segundos que toma para alcanzar con éxito este objetivo.

```
Ingresando al Modo Comando
Recibiendo respuesta
Respuesta: CMD
Estado: Ingreso al Modo Comando Exitoso
Tiempo de Ingreso a modo Comando: 147529 [uSeg]
```

Figura Error! No text of specified style in document..**8**
Tiempo de Ingreso a Modo Comando

Ahora ejecutamos la sección de la conexión entre la DE0-NANO y nuestro LEGO MINDSTORM NXT y podemos observar los mensajes del NIOS y el tiempo de 2.2524 segundos que toma lograr la conexión.

```
Iniciando Conexion DE0-NANO <<-->> NXT
Recibiendo respuesta
Respuesta: TRYING
Conexion DE0-NANO <<-->> NXT establecida...
Tiempo de conexion DE0-NANO <<-->> NXT: 2252468 [uSeg]
```

Figura Error! No text of specified style in document..**9**
Tiempo de conexión DE0NANO con NXT

Una vez realizadas todas las comunicaciones procedemos a configurar los sensores ultrasónicos de nuestro robot, para esto también lo realiza de manera autónoma nuestro programa en NIOS II. Ahora nos muestra el tiempo que le toma configurar ambos sensores, 0.2244 segundos para el sensor 1 (frontal), y 0.2015 para el sensor 2 (lateral derecho).

```
Iniciando Programa NXT
Estableciendo Condiciones Iniciales
Configurando Sensor en Puerto1
Recibiendo respuesta
Respuesta: OK
Tiempo de configuracion Sensor1: 224467 [uSeg]
Configurando Sensor en Puerto4
Recibiendo respuesta
Respuesta: OK
Tiempo de configuracion Sensor2: 201536 [uSeg]
```

Figura Error! No text of specified style in document..**10**
Tiempo configuración sensor 1 y sensor 2

Y por último tomamos el tiempo que ocupa los sensores desde que emitieron un pulso hasta que lo recibieron. En los mensajes del NIOS II nos muestra que el sensor 1 demora 1.1086 segundos y con el sensor 2 demora 1.0259 segundos.

```
Recibiendo respuesta
Recibiendo respuesta
Tiempo de lectura Sensor1: 1108654 [uSeg]
Recibiendo respuesta
Recibiendo respuesta
Recibiendo respuesta
Tiempo de lectura Sensor2: 1025900 [uSeg]
Recibiendo respuesta
```

Figura Error! No text of specified style in document..**11**
Tiempo de lectura sensor 1 y sensor 2

Una vez obtenido los tiempos de prueba con las 6 distancias antes descritas tomadas como referencias, mostraremos las siguientes tablas.

Como observamos el tiempo no varía de acuerdo a la distancia.

TIEMPO CONEXION MODO COMANDO	
DISTANCIA(m)	**TIEMPO (us)**
1	147560
2	147594
5	147660
10	147529
15	147588
20	147532

Tabla Error! No text of specified style in document..**2**
Tiempo de Conexión a Modo Comando

El tiempo si varía de acuerdo a la distancia debido a que el modulo bluetooth debe emitir más, para lograr localizar el receptor.

TIEMPO CONEXION DE0NANO CON NXT	
DISTANCIA (m)	**TIEMPO (us)**
1	2492000
2	2545149
5	2642474
10	4243966
15	4759155
20	117782373

Tabla Error! No text of specified style in document..**3** Tiempo de conexión De0Nano con NXT

Una vez lograda la comunicación bluetooth, el tiempo que le toma al programa NIOS II configurar los sensores, no varía significativamente.

TIEMPO CONFIGURAR SENSORES		
DISTANCIA (m)	**SENSOR1 (us)**	**SENSOR2 (us)**
1	201391	182079
2	220774	182370
5	236827	187662
10	211435	273332
15	216828	246504
20	193911	308625

Tabla Error! No text of specified style in document..**4** Tiempo para configurar sensores

El tiempo de lectura de los sensores tampoco varia significativamente, es decir que NIOS II lo realiza de manera autónoma todo el proceso.

TIEMPO LECTURA DE SENSORES		
DISTANCIA (m)	**SENSOR1 (us)**	**SENSOR2 (us)**
1	1093559	1070320
2	1112937	1105584
5	1089846	1073708
10	1238382	1221289
15	1495530	1268473
20	1188856	1863055

Tabla Error! No text of specified style in document..**5** Tiempo de lectura de sensores

5. Conclusiones

1. Se ha conseguido realizar un robot completamente autónomo con las características necesarias para realizar un mapeo en el cual se identifique claramente los obstáculos y los caminos libres para recorrer.

2. La estimación de la posición está basada en la extracción de la información percibida por los sensores del robot con respecto al entorno en tiempo real, fusionar los datos y expresarlos en un sistema de referencia común para ambos frentes.

3. Se ha desarrollado un método para extraer características en el espacio sensorial y definir regiones de interés para la toma de decisión del siguiente movimiento.

4. El tiempo utilizado para realizar todo el proceso, desde el ingreso al modo comando del módulo bluetooth, hasta la lectura de ambos sensores para predecir el siguiente paso, no varía de manera significativa, teniendo retardos de apenas micro segundos, imperceptibles al momento de la práctica.

6. Referencias

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

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

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

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

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

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

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

[8] Altera, UART Core - Quartus II 9.1 Handbook, Volume 5,

http://www.altera.com.cn/literature/hb/nios2/n2cpu_nii51010.pdf.

[9] Roving Networks, Bluetooth Module RN-42, http://www.rovingnetworks.com/products/RN42.

[10] Microelectronicos, Trabajando con módulos Bluetooth RN-41 y RN-42, http://www.microelectronicos.net/?p=1075.

[11] Lego, Lego Mindstorms NXT Hardware Developer Kit, http://mindstorms.lego.com/en-us/support/files/default.aspx.

[12] Lego, Lego Mindstorms NXT Bluetooth Developer Kit, http://mindstorms.lego.com/en-us/support/files/default.aspx.

[13] Lego, Lego Mindstorms NXT User Guide, http://mindstorms.lego.com/en-us/support/files/default.aspx.

[14] Altera, Introduction to the Quartus II Software, http://www.altera.com/literature/manual/archives/intro_to_quartus2.pdf.

[15] Altera, Quartus II Handbook v12.1.0 Complete Three-Volume Set, http://www.altera.com/literature/lit-qts.jsp

[16] Altera, Introduction to SOPC Builder, http://www.altera.com/literature/hb/qts/qts_qii54001.pdf.

[17] Altera, SOPC Builder User Guide, http://www.altera.com/literature/ug/ug_sopc_builder.pdf.T

[18] Altera, SOPC Builder Support, http://www.altera.com/support/software/system/sopc/sof-sopc_builder.html.

[19] Altera, Nios II Software Build Tools for Eclipse Support, http://www.altera.com/support/ip/processors/nios2/ide/ips-nios2-ide.html.

[20] Altera, Nios II Software Developer's Handbook, http://www.altera.com/literature/hb/nios2/n2sw_nii5v2.pdf.

[21] Altera, Software Development Tools for the Nios II Processor, http://www.altera.com/devices/processor/nios2/tools/ide/ni2-ide.html.

[22] Rosenberg Neil, NXT Programming For Beginners, http://www.rocwnc.org/Beginning_NXT_Programming_Workshop.pdf.

[23] Floyd James, LEGO MINDSTORMS NXT-G Programming Guide Second Edition

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

[25] Lego, Appendix 1-LEGO MINDSTORMS NXT Communication Protocol, http://mindstorms.lego.com/en-us/support/files/default.aspx.

[26] Lego, Appendix 2–LEGO MINDSTORMS NXT Direct commands, http://mindstorms.lego.com/en-us/support/files/default.aspx.

[27] Poliakoff Pierre, Communicating with LEGO NXT via Bluetooth in C#, http://www.codeproject.com/Articles/18857/Communicating-with-LEGO-NXT-via-Bluetooth-in-C.

[28] Roving Networks, Bluetooth RN-42 Command Reference, http://www.rovingnetworks.com/resources/download/47/Advanced_User_Manual.

[29] Altera, Nios II Flash Programmer User Guide, http://www.altera.com/literature/ug/ug_nios2_flash_programmer.pdf.

[30] Altera, Serial Configuration (EPCS) Devices, http://www.altera.com/literature/hb/cfg/cyc_c51014.pdf