ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DATALOGGER USANDO NIOS II"

TESINA DE SEMINARIO

Previa la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

INGENIERO EN TELEMÁTICA

Presentado por:

Luis Enrique Campoverde Rugel

Washington Adrián Velásquez Vargas

GUAYAQUIL – ECUADOR

AÑO 2013

AGRADECIMIENTO

Agradezco a Dios por guiarme durante estos 5 años de lucha constante para obtener mi título profesional.

A Fanny y Martha mis madres que siempre están guiándome por el camino del bien.

A Carlos y Talía por su apoyo constante, consejos y buenos ejemplos que me sirvieron de ayuda durante mis estudios universitarios.

Al Ing. Ronald Ponguillo que con su guía pudimos sacar este proyecto adelante.

Luis Campoverde R.

Quiero agradecer a Dios, por iluminarme siempre y nunca desampararme en los momentos difíciles, a mi familia sobre todo a mi madre Cruz María Vargas Rodríguez por el apoyo incondicional que tuve, por los consejos que recibí, a todos mis amigos que siempre estuvieron apoyándome y por la dedicación mutua que teníamos por salir adelante en nuestras carreras y a las personas que conocí que me proporcionaron su ayuda para salir adelante en las buenas y las malas.

Un especial agradecimiento a mi profesor de materia, Ing. Ponguillo por confiar en mí en que este proyecto lo íbamos a realizar con éxito, a todos ellos gracias.

Washington Velásquez V.

DEDICATORIAS

Dedicado para Astrid mi hermana una persona muy especial que está a punto de iniciar su vida universitaria, espero que este proyecto le sirva de ejemplo e inspiración y pueda culminar su carrera con total éxito.

Luis Campoverde R.

Se lo dedico a una persona que ha sido muy importante en toda la trayectoria de mi carrera universitaria, gracias al apoyo incondicional y la estimación personal de la Srta. Karen Escalante.

Washington Velásquez V.

TRIBUNAL DE SUSTENTACIÓN

_____

Ing. Ronald Ponguillo

PROFESOR DEL SEMINARIO DE GRADUACIÓN

_____

Ing. Víctor Asanza

PROFESOR DELEGADO POR LA UNIDAD ACADÉMICA

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)

_____

Luis Enrique Campoverde Rugel

_____

Washington Adrián Velásquez Vargas

RESUMEN

El presente proyecto consiste en la implementación de un Datalogger utilizando el microprocesador NIOS II el cual está embebido en una FPGA CYCLONE II que se encuentra integrada a la tarjeta de desarrollo ALTERA DE2, además de obtener datos de distintos sensores y almacenarlos en una tarjeta SD Card.

Para la realización del proyecto se aplican cuatro etapas. La primera etapa está basada en obtener los datos mediante el uso de sensores y la transmisión usando un PIC, la siguiente etapa se basa principalmente en la recepción de los datos, la tercera etapa consiste en utilizar la pantalla LCD de la tarjeta DE2 para mostrar los datos obtenidos y una última etapa donde por medio de algoritmos implementados en lenguaje C, se procede a almacenar los datos en una tarjeta SD Card.

Al proyecto se lo ha estructurado en 4 capítulos como se lo explica a continuación:

En el primer capítulo, se exponen los objetivos generales y específicos del proyecto, además de especificar claramente los alcances y limitaciones.

En el segundo capítulo, se da a conocer la parte teórica, explicando los conceptos básicos de los elementos y software usados para el desarrollo del

proyecto, además de conocimientos adicionales que se tuvo en cuenta con lo que respecta a transmisión y recepción de datos.

En el tercer capítulo, se muestra mediante el uso de diagramas de bloques y diagramas funcionales como se desarrolló el proyecto, se describe paso a paso las distintas etapas que se tuvieron que implementar para cumplir con los objetivos propuestos.

Para el cuarto capítulo se cuenta con varios escenarios para poner a prueba la funcionalidad del proyecto, dichos escenarios se implementaron en un ambiente de laboratorio. Los resultados obtenidos fueron analizados usando gráficas las cuales permitieron obtener conclusiones sobre la funcionalidad del proyecto.

En base a las conclusiones obtenidas estamos en la capacidad de dar una conclusión general de toda la funcionalidad del proyecto, la misma que nos permite proporcionar recomendaciones que servirán como base para futuras aplicaciones o mejoras que se le quiera dar al presente proyecto.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS

ADC	ANALOG-TO-DIGITAL CONVERTER
ALU	ARITHMETIC LOGIC UNIT
BUS	BACK UP SYSTEM
CAD	COMPUTER AIDED DESIGN
CPU	CENTRAL PROCESSING UNIT
DAC	DIGITAL-TO-ANALOG CONVERTER
DDR	DOUBLE DATA RATE
DE2	DEVELOPMENT AND EDUCATION BOARD II
EEPROM	ELECTRICALLY ERASABLE PROGRAMMABLE READ ONLY MEMORY
EUSART	ENHANCED UNIVERSAL SYNCHRONOUS / ASYNCHRONOUS RECEIVER / TRANSMITTER
FPGA	FIELD PROGRAMMABLE GATE ARRAY
GUI	GRAPHICAL USER INTERFACE

HDL	HARDWARE DESCRIPTION LANGUAGE
HW	HARDWARE
IrDA	INFRARED DATA ASSOCIATION
JTAG	JOINT TEST ACTION GROUP
LCD	LIQUID CRYSTAL DISPLAY
LED	LIGHT-EMITTING DIODE
MIPS	MICROPROCESSOR WITHOUT INTERLOCKED PIPELINE STAGES
MUX	MULTIPLEXOR
NRZ	NON RETURN TO ZERO
NTSC	NATIONAL TELEVISION SYSTEM COMMITTEE
PAL	PHASE ALTERNATION BY LINE
PCB	PRINTED CIRCUIT BOARD
RAM	RANDOM ACCESS MEMORY
RISC	REDUCED INSTRUCTION SET COMPUTER

ROM	READ ONLY MEMORY
RS-232	RECOMENDED STANDARD-232
SDRAM	SYNCHRONOUS DYNAMIC RANDOM ACCESS MEMORY
SOPC	SYSTEM ON A PROGRAMMABLE CHIP
SPI	SERIAL PERIPHERAL INTERFACE
SRAM	STATIC RANDOM ACCESS MEMORY
SW	SOFTWARE
TTL	TRANSISTOR TRANSISTOR LOGIC
UART	UNIVERSAL ASYNCHRONOUS RECEIVER / TRANSMITTER
USART	UNIVERSAL SYNCHRONOUS / ASYNCHRONOUS RECEIVER / TRANSMITTER
USB	UNIVERSAL SERIAL BUS
VGA	VIDEO GRAPHICS ARRAY
VHDL	VHSIC HARDWARE DESCRIPTION LANGUAGE

INTRODUCCIÓN

Con el avance de la tecnología que estamos viviendo en los actuales momentos, nos podemos dar cuenta que la electrónica digital se está convirtiendo en un estándar en casi todos los campos de la ingeniería. Los sistemas digitales en la actualidad se encuentran en constante evolución siendo cada día más fáciles de manejar optimizando recursos, tiempo y dinero.

Algunas de las aplicaciones de los sistemas digitales, se encuentran el monitoreo en tiempo real de parámetros relevantes dentro de alguna aplicación específica; por tanto, la mayoría de empresas o industrias tienen la necesidad de tener sistemas de almacenamiento de datos.

La posibilidad de tener datos históricos, hacer proyecciones y predicciones en base a los datos guardados es lo que impulsa la utilización de estos sistemas. Si bien un monitoreo diario, semanal o mensual, de una o de diferentes variables podría hacerse en forma manual, una mejor alternativa es contar con un sistema que recolecte los datos de manera automática y controlada.

Las dificultades que se presentan para la captura de datos ha motivado la búsqueda constante de sistemas que faciliten la recolección. Dentro de este esfuerzo se enmarca este proyecto: implementar una alternativa óptima para

muestrear en forma automática señales analógicas de temperatura, utilizando como principal instrumento la tarjeta de desarrollo DE2 de Altera para la recepción de datos y dispositivos electrónicos para la transmisión.

CAPÍTULO 1

1. GENERALIDADES

1.1. JUSTIFICACION

Un DATALOGGER es un instrumento que se encarga de realizar mediciones para luego almacenarlas en su memoria; manteniendo un registro de distintas mediciones en un instante de tiempo. Teniendo esto en mente nos planteamos la pregunta más significativa, **¿Por qué un DATALOGGER?**

Si bien es cierto que estos aparatos electrónicos ya se encuentran en el mercado, siendo su uso ilimitado en el proceso de monitoreo de variables

en el campo industrial, doméstico o simplemente como un ensayo de laboratorio educacional, nos dimos cuenta que el uso de estos instrumentos es de fundamental importancia en la vida diaria. Si tomamos en cuenta el campo industrial veremos que existen variables que son de gran importancia al hacer un análisis específico, estas variables pueden ser de Temperatura, Humedad, Salinidad del Agua, Potencia de una Señal RF entre otras. Para obtener un análisis claro se debe llevar un registro de las mismas, para lo cual uno de los posibles métodos de almacenamiento es usar un Datalogger ya que nos ofrece seguridad y movilidad de los datos almacenados. Teniendo en cuenta lo mencionado anteriormente decidimos implementar un Datalogger digital el cual tiene como núcleo principal el microprocesador NIOS II que será embebido en una FPGA CYCLON II de la compañía Altera.

1.2. IDENTIFICACIÓN DEL PROBLEMA

Llevar un registro de varios parámetros de importancia en cierta aplicación es una necesidad que ha surgido últimamente y se lo puede observar en cualquier campo de ingeniería.

Hacer uso de dispositivos con capacidad de llevar un registro de diferentes tipos de parámetros en tiempo real, permite a las personas

tener información importante para realizar análisis que ayude a prevenir posibles errores presentes en dicha aplicación.

Para comodidad de los usuarios es necesario que estos dispositivos sean lo más pequeños posibles permitiendo portabilidad además de tener una interfaz amigable la cual facilite su uso.

El dispositivo electrónico que se esté desarrollando debe tener la capacidad de almacenar diferentes tipos de parámetros, capacidad de generar un archivo diferente por cada lectura que se tome y sobre todo debe ser capaz de interactuar con el usuario mediante una interfaz gráfica.

El uso de FPGA en la actualidad es una alternativa que están implementando los desarrolladores de hardware y software para optimizar el uso de recursos, tiempo de procesamientos en una comunicación y sobre todo dinero.

Para ir a la par con esta evolución tecnológica se presenta el desarrollo de un sistema que permita almacenar en tiempo real valores de parámetros que sean de interés, para luego analizar dichos valores y verificar la funcionalidad del sistema implementado.

1.3. OBJETIVOS

1.3.1. OBJETIVOS GENERALES

- Obtener habilidades en el diseño e implementación de soluciones hardware/software con circuitos de lógica configurables (FPGA) para dar solución a problemas específicos que se presentan en nuestra sociedad.
- Comprender el uso del microprocesador NIOS II junto con sus diferentes periféricos de E/S para lograr diseñar un microcontrolador que cumpla con las especificaciones basadas en el diseño previo y de esta forma obtener un dispositivo capaz de interactuar de forma autónoma y realizar las tareas asignadas vía software.

1.3.2. OBJETIVOS ESPECÍFICOS

- Comprender el manejo de la tarjeta de desarrollo DE2 de Altera
- Construir un sistema embebido basado en el microprocesador NIOS II y bloques de lógica reconfigurable que funcione como un datalogger.
- Implementar una solución HW/SW que permita leer información de sensores y almacenarla en una memoria SDCard.

- Desarrollar un proyecto basado en el microprocesador NIOS II que sirva como punto de partida para futuras investigaciones de tal forma que se logre implementar una interfaz amigable para el usuario.

1.4. ALCANCE Y LIMITACIONES

La tarjeta de desarrollo DE2 de Altera nos brinda algunas alternativas para poder cumplir con los requisitos básicos de nuestro datalogger, pero siendo este un proyecto con finalidad académica se presentan a continuación alcances y limitaciones del mismo:

ALCANCE

- El datalogger permitirá a los usuarios poder visualizar en la LCD de la DE2 la temperatura actual que sensan los dos sensores de temperatura (LM35).
- La información obtenida por los sensores de temperatura será almacenada en una SDCard.
- Los datos obtenidos serán almacenados en formato de texto para una mejor interpretación.
- Se realizará un análisis teórico-gráfico con los datos obtenidos.

LIMITACIONES

- Las pruebas del Datalogger serán realizadas en un ambiente de laboratorio.
- El tiempo de almacenamiento de datos no sobrepasará los 30min debido a la capacidad de la tarjeta SD.

1.5. HARDWARE Y SOFTWARE A UTILIZAR

HARDWARE

- Tarjeta DE2 – Altera
- Sensores de Temperatura – LM35
- Microcontrolador 16F887

SOFTWARE

- Quartus II 11.1
- NIOS II SBT
- Proteus 7.6
- MikroC 5.0
- Altium Designer 10.0

CAPÍTULO 2

2. MARCO TEÓRICO

En este capítulo se muestra el concepto teórico, la explicación técnica del hardware y software para el desarrollo de nuestro proyecto. Se presenta de manera básica el funcionamiento de la tarjeta de desarrollo DE2 de Altera, así como la función de cada una de los módulos usados dentro de este proyecto además se muestra las características principales del microcontrolador PIC 16F887 y del sensor de temperatura LM35.

2.1. TARJETA DE DESARROLLO DE2

2.1.1. PARTES Y CARACTERÍSTICAS

La tarjeta DE2 cuenta con varias características que permiten al usuario poner en práctica sus conocimientos para el desarrollo de una variedad de circuitos digitales, circuitos que van desde lo más simple como un arreglo de puertas lógicas, hasta diseños que demanda el uso de la mayoría de sus periféricos.

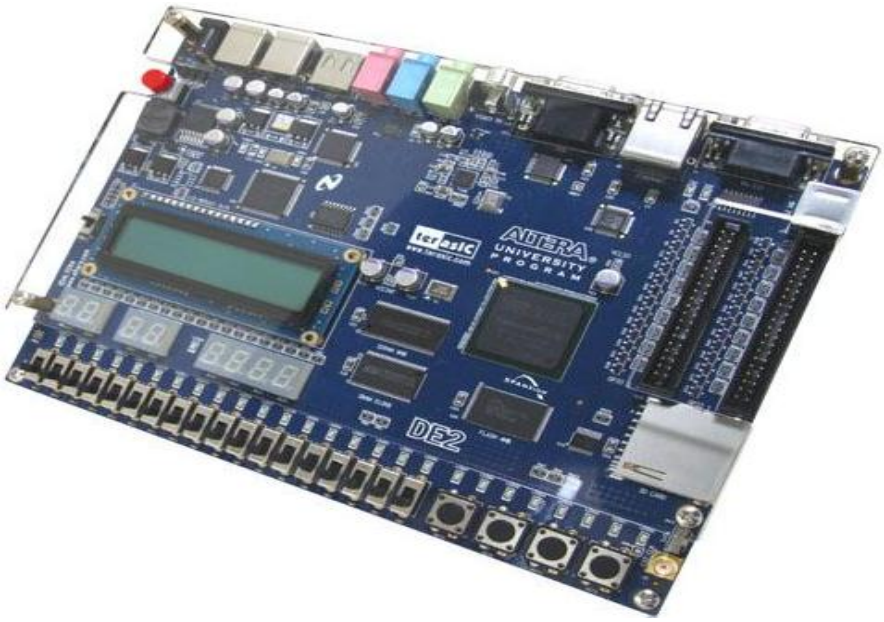


Figura 2.1 - TARJETA DE DESARROLLO DE2 DE ALTERA [1]

La tarjeta de desarrollo DE2 es un módulo ideal para el aprendizaje de lógica digital, organización de computadoras y FPGA; esta tarjeta cuenta con una FPGA CYCLONE II para fines educativos y de investigación. La tarjeta DE2 es de gran importancia para tener una mayor comprensión de los conceptos de lógica digital, conceptos que se aplican en la elaboración de diseños.

A continuación se muestran las partes que conforman a la tarjeta DE2 de Altera:

FPGA

- Altera Cyclone II

Interfaces de Comunicación

- Dispositivos de Configuración Serial (EPCS16) para Cyclone II 2C35
- USB Blaster para configurar la FPGA.

Memorias

- 512-Kbyte SRAM
- 8-Mbyte SDRAM
- 4-Mbyte Memoria Flash

Leds y Botones

- 4 Pulsadores
- 18 Switches
- 18 Leds Rojos
- 9 Leds Verdes

Fuentes de Reloj

- Oscilador de 50-MHz
- Oscilador de 27-MHz

Audio y Video

- Linea In/Out, Micrófono (24-bit Audio CODEC).
- VGA DAC de 10 bits
- Decodificador de TV (NTSC/PAL)

Ethernet

- 10/100 controlador Ethernet con socket

USB Host/Esclavo

- USB tipo A
- USB tipo B

Conectores

- Conector para tarjetas SD
- Interfaz RS 232 con conector DB9 de 9-pines.
- Conector PS/2 (ratón, teclado).
- 2 conectores de expansión de 40 pines

Varios

- Módulo LCD de 16x2
- 8 Displays de 7 segmentos
- Transceptor Infrarrojo IrDA

2.1.2. INTERFAZ LCD

Figura 2.2 - MÓDULO LCD 16X2 [2]

La pantalla de cristal líquido o LCD (Liquid Crystal Display) es un dispositivo controlado de visualización gráfica para la presentación de caracteres, símbolos o incluso dibujos (en algunos modelos), el modelo

presente en la tarjeta DE2 es de 2 filas de 16 caracteres donde cada una dispone de una matriz de 5x7 puntos (pixeles).

Las características principales que posee la pantalla de cristal líquido de la tarjeta DE2 se muestran a continuación:

- Pantalla de caracteres ASCII
- Desplazamiento de los caracteres hacia la izquierda o la derecha
- Movimiento del cursor

Para llevar a cabo las características mencionadas anteriormente la pantalla LCD cuenta con 11 señales internas las cuales son conectadas a la FPGA tal como se muestra en la figura 2.3

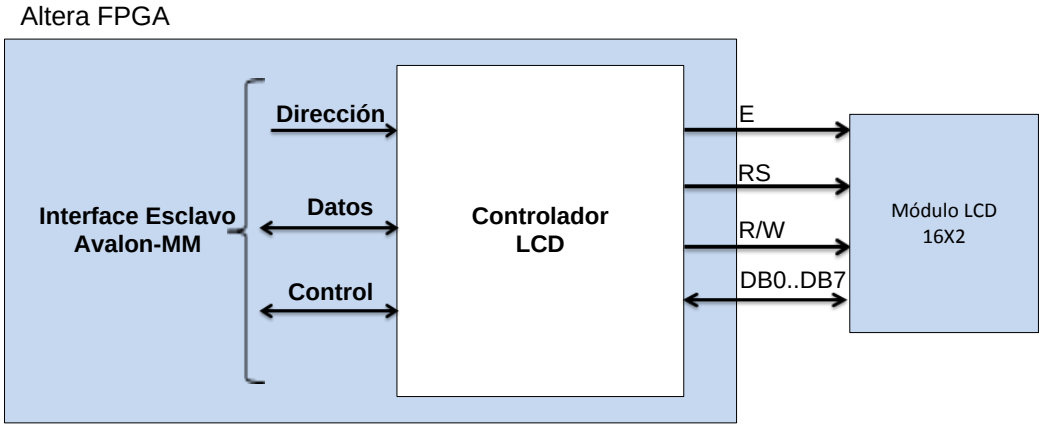


Figura 2.3 - DIAGRAMA DE CONEXIÓN DEL MÓDULO LCD16X2 [3]

Como podemos observar en la figura 2.3 la pantalla LCD necesita de un controlador para poder interactuar con el microprocesador NIOS II que se encuentra embebido en la FPGA. El controlador LCD está acompañado de los siguientes archivos generados por software, estos archivos definen la interfaz de bajo nivel con el hardware y proporciona los controladores HAL. Se recomienda que los desarrolladores no modifiquen estos archivos

altera_avalon_lcd_16207_regs.h

Este archivo proporciona las declaraciones de funciones y constantes simbólicas para acceder al hardware de bajo nivel.

altera_avalon_lcd_16207.c

Este archivo implementa las funciones necesarias para el manejo del dispositivo LCD, los cuales se generan a partir de la librería de sistema HAL.

La interface de comunicación entre el microprocesador NIOS II y la pantalla LCD se conoce como Interface Avalon-MM Slave

2.1.3. MÓDULO UART

El módulo UART permite convertir los datos recibidos en forma paralela, a forma serial mediante una secuencia de caracteres entre un sistema embebido en una FPGA y un dispositivo externo. Se encuentra implementado en base al protocolo de transmisión RS-232 que ofrece varias tasas de transmisión estándar, bits de paridad, bits de datos y señales opcionales de flujo de control como RTS/CTS. El conjunto de características de este protocolo es configurable, permitiendo a los diseñadores aplicar sólo la funcionalidad necesaria para un sistema dado.

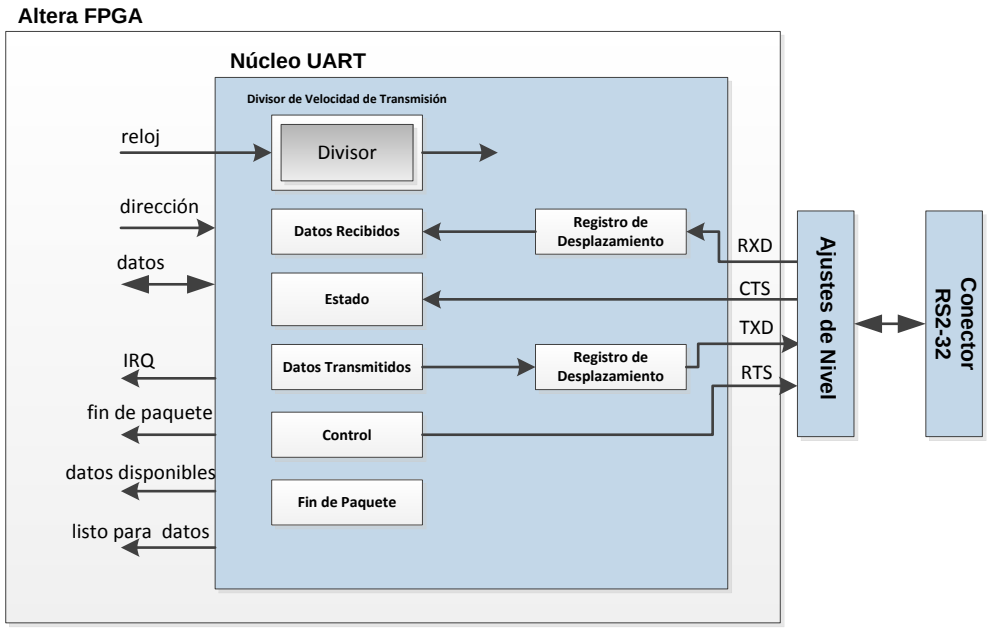


Figura 2.4 - MÓDULO UART [4]

Como podemos observar en la figura 2.4 la comunicación entre el microprocesador NIOS II y el módulo UART se la realiza mediante la Interface Avalon – MM la cual permite acceder a los diferentes registros del módulo UART. También se observan los 7 registros que conforman el módulo UART además de un divisor de Baudios, cada registro permite almacenar datos de 16 bits.

2.1.3.1. LÓGICA DE TRANSMISIÓN

La transmisión UART se compone de 7, 8 o 9 bits de datos los cuales ingresan serialmente al registro de desplazamiento TXdata antes de ser transmitidos, los mismos 7,8 o 9 bits son enviados hacia otro registro de desplazamiento para luego ser transmitidos de forma paralela. Los datos son escritos en el registro de desplazamiento TXdata a través de la interfaz Avalon MM Slave, durante la transmisión el dato se carga en el registro TXdata de manera automática siempre y cuando no se esté realizando la transmisión de otro dato. La lógica de transmisión inserta automáticamente el número de bits de inicio, de parada y de paridad en los datos transmitidos serialmente.

2.1.3.2. LÓGICA DE RECEPCIÓN

La lógica de recepción es exactamente igual a la lógica de transmisión, los datos recibidos pueden ser de 7, 8 o 9 bits. Estos datos son almacenados en un registro de desplazamiento cuando son recibidos para luego ser enviados al registro de desplazamiento RXdata, la interface Avalon MM Slave es la encargada de llevar los datos recibidos al módulo UART para su respectiva interpretación. Para que un nuevo dato pueda ser recibido ambos registros deben estar vacíos, mientras los dos registros se encuentren ocupados ningún dato podrá ser receptado. La lógica de recepción verifica automáticamente los bits de inicio, parada y paridad en los datos recibidos serialmente.

2.1.4. ESTÁNDAR RS232

RS-232 (Recommended Standard 232, también conocido como Electronic Industries Alliance RS-232C) es una interfaz que designa una norma para el intercambio serie de datos binarios entre un DTE (Equipo terminal de datos) y un DCE (Data Communication Equipment, Equipo de Comunicación de datos).

El RS-232 incluye un conector tipo DB-25 (de 25 pines), aunque es normal encontrar la versión de 9 pines (DE-9), más barato e incluso más

extendido para cierto tipo de periféricos (como el ratón serie del PC). En la configuración más básica tres hilos son suficientes para una comunicación full duplex.

2.1.4.1. CONECTOR RS232D - CONECTOR DB9

La Figura 2.5 muestra el conector de 9 pines

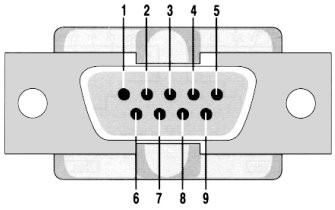


Figura 2.5 - CONECTOR DB9

A continuación en la tabla 2.1 se muestra la descripción de todos los pines

PIN	NOMBRE	DESCRIPCIÓN
1	CD	Detección de Portadora
2	RXD	Recepción de Datos
3	TXD	Transmisión de Datos
4	DTR	Terminal de Datos Preparado
5	GND	Señal de Tierra
6	DSR	Dispositivo Preparado
7	RTS	Petición de Envío
8	CTS	Preparado para Transmitir
9	RI	Indicador de Llamada Entrante

Tabla 2.1 - DESCRIPCIÓN DE PINES PARA UN CONECTOR DB9 HEMBRA

2.1.5. MÓDULO SDCARD

Una tarjeta SD es un dispositivo de almacenamiento usado comúnmente en cámaras digitales, teléfonos celulares, computadoras, tabletas etc. Sirve para el almacenamiento de todo tipo de archivos, es portable lo cual permite que los datos almacenados sean transferidos con facilidad hacia otros dispositivos. La tarjeta de desarrollo DE2 cuenta con un puerto para conexión de tarjetas SD facilitando cualquier diseño que implemente el uso de una tarjeta SD.

El módulo de control para la interfaz SDCard es proporcionado por el programa Universitario de Altera, el cual nos da acceso al hardware necesario para el manejo de este dispositivo, el núcleo se ha diseñado para ser implementado mediante el generador SOPC Builder

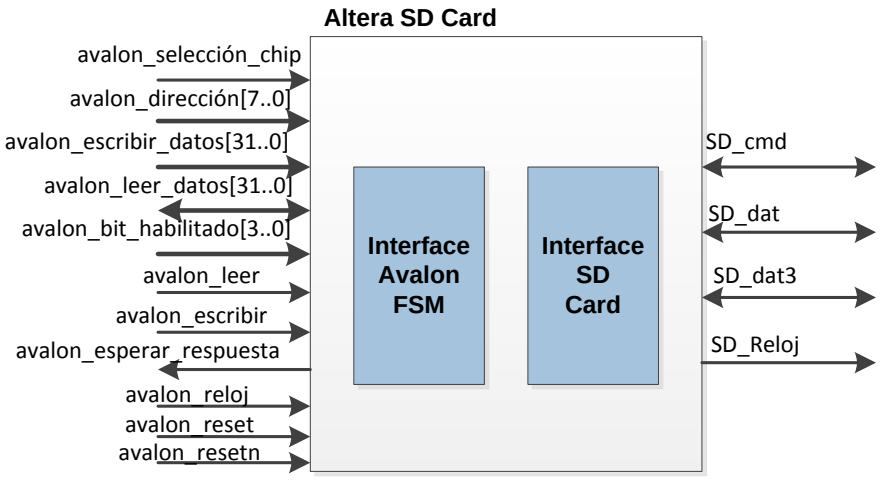


Figura 2.6 - MÓDULO SDCARD [5]

Un diagrama de bloques de alto nivel de muestra en la figura 2.6, las señales que se muestran en la parte izquierda del diagrama de bloques son las interconexiones con la interfaz Avalon, los requerimientos de lectura y escritura que son interpretados como comandos o datos por interfaz Avalon de estados finitos Avalon que se muestra en la figura 2.6. Existen dos tipos de requerimientos o solicitudes que permiten el acceso al módulo SD para su respectiva configuración y almacenamiento de datos.

- **Solicitudes de comando**

Son utilizadas para configurar el módulo SD y especifican los lugares de la memoria a los cuales se desean acceder

- **Solicitudes de datos**

Son usadas para almacenar los datos en las localidades de memoria especificadas por el diseñador.

2.2. FPGA CYCLONE II

FPGA

FPGA (Field Programable Gate Array) es un dispositivo semiconductor, el cual contiene en su interior un conjunto de circuitos digitales

reprogramables. Puede entenderse conceptualmente como un arreglo bidimensional de bloques lógicos fijos.

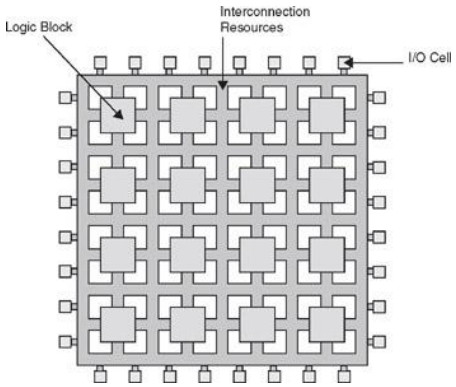


Figura 2.7 - ESTRUCTURA DE UNA FPGA [6]

Las FPGA cuentan con cableado reprogramable entre bloques lógicos. Una FPGA puede emular desde Circuitos lógicos básicos como Sumadores, MUX, Decodificadores entre otros hasta procesadores (Intel, MIPS). Puede programarse de modo que implemente diferentes circuitos lógicos aislados unos de otros, desempeñando tareas completamente diferentes, en paralelo. Son ampliamente utilizados en etapas de prueba de circuitos lógicos complejos (procesadores y sistemas embebidos).

La adopción de chips FPGA en las industrias ha sido impulsada por el hecho de que las FPGA combinan lo mejor de los circuitos integrados de

aplicación específica (ASICs) y de los sistemas basados en procesadores.

Los beneficios de usar FPGA en una aplicación son los siguientes:

- Tiempos más rápidos de respuesta de E/S y funcionalidad especializada.
- Rápida generación de prototipos y verificación sin el proceso de fabricación del diseño personalizado de ASIC
- Implementar funcionalidad personalizada con la fiabilidad de hardware determinístico dedicado

CYCLONE II

Figura 2.8 - FPGA CYCLONE II [7]

La familia CYCLONE II ofrece las siguientes características:

- Arquitectura de alta densidad con 4,608 - 68,416 elementos lógicos

- Bloques embebidos de Memoria M4K.
- Hasta 1.1Mbits de memoria RAM útil, sin reducir la lógica disponible.
- 4,096 bits de memoria por bloque.

Multiplicadores Embebidos

- Hasta 150 multiplicadores de 18x18 bits cada uno configurables como dos independiente de 9x9 bits.
- Registros de entrada y salida opcionales.

Soporte avanzado de E/S

- Soporte externo de memoria de alta velocidad incluyendo DDR, DDR2 y SDR, SDRAM y QDRII SRAM soportados por las funciones de Altera IP MegaCore.
- Tres registros dedicados para cada elemento de E/S: un registro de entrada, un registro de salida y un registro habilitador de salida.

Circuito de administración flexible de Reloj

- Red de jerarquía de reloj hasta 402.5 MHz de rendimiento
- Hasta 16 líneas de reloj globales en la red global de reloj a lo largo de todo el dispositivo

Configuración de Dispositivo

- Configuración serial de alta velocidad que permite tiempos de configuración inferiores a 100ms
- Soporta diferentes modos de configuración: Serial Activo, Serial Pasivo y JTAG
- La configuración del dispositivo soporta múltiples voltajes (3.3, 2.5 o 1.8 V)

2.3. MICROPROSESADOR EMBEBIDO NIOS II

El fabricante Altera proporciona una infraestructura completa para crear sistemas embebidos según las necesidades del diseñador, por medio de la combinación de una serie de componentes configurables sobre sus FPGA [23].

Para ello, proporciona un entorno específico, al que denomina SOPC Builder, que permite la definición y configuración a medida de un sistema basado en el microprocesador NIOS II, y que con la herramienta de síntesis Quartus II puede ser implementado directamente sobre una FPGA.

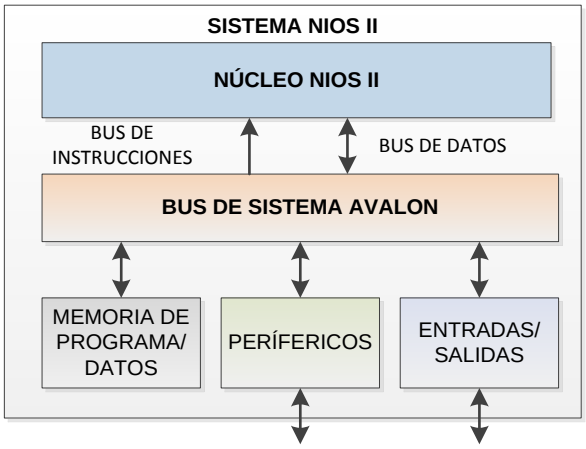


Figura 2.9 - SISTEMA BASADO EN EL MICROPROCESADOR NIOS II [8]

La figura 2.9 muestra un sistema NIOS II el cual se compone de lo siguiente:

- Procesador NIOS II
- Memoria interna de programa
- Memoria interna de datos
- Periféricos integrados
- Interfaces para memoria externa y/o entrada/salida.

Al tratarse de un sistema flexible que puede ser adaptado a las necesidades específicas de cada diseño, no sólo posibilita ajustar su tamaño al de un determinado dispositivo, sino que deja decidir al diseñador cuando una implementación se debe realizar en software y

cuando en hardware, permitiendo elevar el rendimiento y ajustar los costes.

2.3.1. VERSIONES DEL MICROPROCESADOR NIOS II

El microprocesador NIOS II puede ser implementado en cualquiera de las 3 versiones en la que se encuentra disponible, las 3 versiones disponibles del microprocesador NIOS II se muestran a continuación:

- El NIOS II/f (“fast”) es la versión diseñada para alto rendimiento, y que con un “pipeline” de 6 etapas proporciona opciones para aumentar su desempeño, como memorias caché de instrucciones y datos, o una unidad de manejo de memoria (MMU, Memory Management Unit).
- El NIOS II/s (“standard”) es la versión con “pipeline” de 5 etapas que dotada de una unidad aritmética lógica (ALU, Arithmetic Logic Unit) busca combinar rendimiento y consumo de recursos.
- El NIOS II/e (“economy”) es la versión que requiere menos recursos de la FPGA, sin “pipeline” y muy limitada, dado que carece de las operaciones de multiplicación y división.

Cada una de estas versiones se complementa con una serie de componentes, memoria y periféricos, por medio de su interconexión a

través de un bus de sistema al que denominan “Avalon Switch Fabric”, para obtener un sistema NIOS II completo en un chip (SOC, System On Chip) [24].

2.3.2. CARACTERÍSTICAS DEL MICROPROCESADOR NIOS II

El NIOS II es un procesador de propósito general de 32 bits, basado en una arquitectura tipo Harvard, dado que usa buses separados para instrucciones y datos, cuyas principales características son:

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

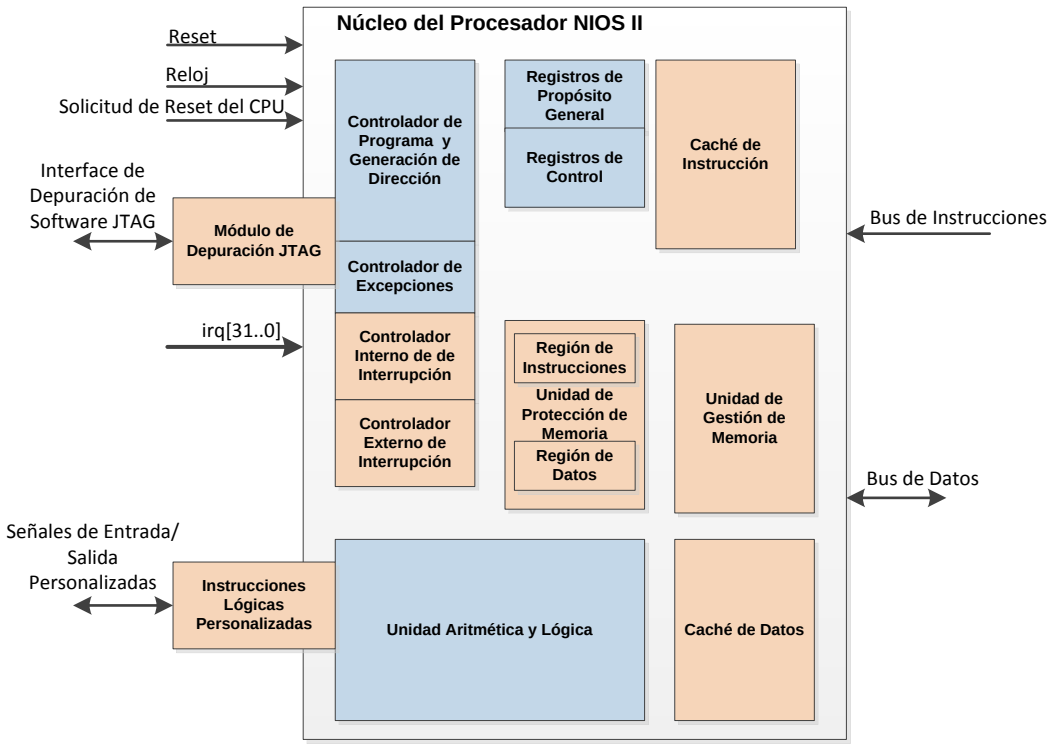


Figura 2.10 - ESTRUCTURA DEL MICROPROCESADOR NIOS II [9]

La figura 2.10 muestra la arquitectura interna del microprocesador NIOS II en la cual se puede observar la unidad lógica aritmética, la región donde se procesan las instrucciones, controladores internos entre otras.

2.3.3. CONEXIÓN Y ACCESO A MEMORIA

El microprocesador NIOS II utiliza 32 bits para realizar un direccionamiento por byte, poseyendo buses separados para instrucciones y datos, este tipo de direccionamiento corresponde a una

arquitectura tipo Harvard. En este tipo de conexión, los buses para la memoria de instrucciones se utilizan para leer el programa, que debe ser ejecutado por el procesador, mientras que el bus para la memoria de datos proporciona acceso a los diferentes bloques de memoria y a los periféricos del sistema [25].

Sin embargo, existen tres modos posibles de conexión con memoria:

- Conexión de memoria a través de la red Avalon, tanto de memoria interna como externa.
- Conexión directa de bloques de memoria interna, con lo que se proporciona un rápido acceso a memoria.
- Conexión de memoria a través de caché.

El microprocesador NIOS II puede incluir tanto caché de datos como de instrucciones, implementadas en los bloques de memoria de la FPGA. La inclusión de estas memorias caché aumenta el rendimiento de forma significativa, sobre todo cuando la mayoría de la memoria principal se sitúa en un chip SDRAM externo.

La memoria caché se organiza, al igual que la RAM, en un arreglo tipo tabla, formado por filas y columnas. En este caso, los tamaños de caché son configurables, aunque la caché de instrucciones se organiza en

líneas de 8 bytes, mientras que la caché de datos se organiza en líneas de 4, 16 o 32 bytes. La gestión de caché se realiza por software, mediante las instrucciones definidas para este propósito según se trate de caché de instrucciones o datos. Se diferencian dos tipos de instrucción: para la carga y asociación de una línea de caché con una determinada dirección de memoria y para llevar y vaciar el contenido de una línea de caché asociada con una determinada dirección de memoria. Aunque por norma, se usará la caché cuando exista, es posible saltarse el acceso a través de caché y realizarlo de forma directa. Pero no deben mezclarse ambos métodos, dado que se puede provocar que la coherencia de los datos de memoria y caché se vea comprometida.

2.3.4. CONEXIÓN Y ACCESO A PERIFERICOS

En el microprocesador NIOS II, la memoria de datos y los periféricos comparten la capacidad de direccionamiento de 32 bits. Así, el acceso de entrada/salida (I/O) se realiza a través del mapa de memoria, del mismo modo que cualquier otra dirección de memoria (acceso a través de las mismas instrucciones) [9].

En la figura 2.11 se muestra los niveles de acceso a memoria así como a los periféricos conectados al microprocesador NIOS II

Figura 2.11 - CONEXIÓN ENTRE PERIFÉRICOS [10]

Dado que el conjunto de periféricos y la capacidad de memoria son configurables, el mapa de direcciones de memoria y periféricos depende del sistema diseñado, estableciéndose en el momento de generar el sistema, al igual que las direcciones de reset e interrupción.

2.3.4.1. PERIFÉRICOS ESTÁNDAR

Altera proporciona un conjunto de periféricos comúnmente usados, como: temporizadores, interfaces de comunicación serie, entradas/salidas (I/O) de propósito general, controladores SDRAM y otras interfaces de memoria.

2.3.4.2. PERIFÉRICOS A MEDIDA

La verdadera potencia de los sistemas embebidos configurables, implementables sobre FPGA, aparece en el momento en que el diseñador tiene la posibilidad de decidir que partes del sistema serán implementadas en software y cuales en hardware.

Así, todas aquellas tareas cuyo rendimiento sea crítico, pueden ser implementadas en hardware a través del desarrollo de un periférico a medida. Con lo que se obtiene un mayor desempeño gracias al carácter concurrente de la implementación hardware y se libera al procesador para realizar otras funciones en paralelo, mientras los demás periféricos procesan los datos. La integración de periféricos a medida con el microprocesador NIOS II presenta una serie de alternativas según las necesidades específicas del diseño, pasando en la mayoría de los casos, por implementar una determinada interfaz de la red Avalon.

Al igual que se pueden definir periféricos a medida, el microprocesador NIOS II también ofrece la posibilidad de añadir instrucciones propias a la unidad aritmético lógica (ALU), sobre todo cuando se utilizan aplicaciones de procesamiento digital de señales (DSP, Digital Signal Processing).

2.3.5. RED AVALON

Para la interconexión de los diferentes componentes, o bloques de diseño, que conforman un sistema hardware se pueden identificar, tradicionalmente, cuatro arquitecturas de conexión diferentes:

- Conexión en “Bus”: se basa en la conexión de maestros y esclavos por medio de una unidad de arbitraje común, lo que permite operar a frecuencias relativamente altas a costa de perder la concurrencia.
- Conexión “Full Crossbar Switch”: se basa en la conexión directa de maestros y esclavos por medio de una matriz de conexiones, lo que permite transacciones concurrentes entre los diferentes elementos del sistema, siendo uno de sus usos principales las aplicaciones de computación de alto rendimiento.
- Conexión “Partial Crossbar Switch”: se basa en la conexión directa de un maestro con un determinado grupo de esclavos, lo que permite ahorrar recursos en la creación de la matriz de conexiones.
- Conexión en “Streaming”: se basa en la conexión directa entre fuente (“source”) y sumidero (“sink”), creando un flujo de datos unidireccional para la transferencia de datos a alta velocidad, dado

que elimina la unidad de arbitraje al crear una conexión punto a punto, siendo uno de sus principales usos el procesamiento de vídeo.

Altera, basándose en una arquitectura “Partial Crossbar Switch”, implementa una serie de interfaces, a las que designa como Avalon como se muestra en la figura 2.12, las cuales facilitan la interconexión de componentes en sistemas complejos, y permiten la interconexión de sistemas concurrentes, gracias a su estructura multimaestro.

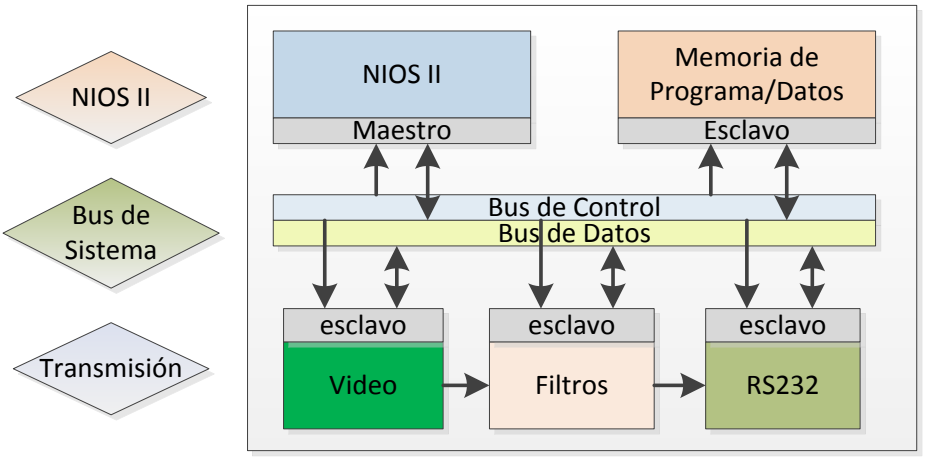


Figura 2.12 - INTERCONEXIÓN DE PERIFÉRICOS CON LA RED AVALON [11]

Por otro lado, la conexión al bus del sistema permite que el control sea realizado por el microprocesador y facilita el intercambio de información con memoria, posibilitando el post-procesado mediante su programación

en un lenguaje de alto nivel, lo que eliminaría las dificultades y costes asociados a su desarrollo en hardware, y daría una mayor flexibilidad al sistema.

2.3.5.1. INTERFACES AVALON

Existen seis tipos diferentes de interfaz:

- “Avalon Memory Mapped Interface” (Avalon-MM) es una interfaz de lectura/escritura basada en direcciones, típica de conexiones maestro-esclavo.
- “Avalon Streaming Interface” (Avalon-ST) es una interfaz que soporta un flujo de datos unidireccional, incluyendo streams multiplexados, paquetes y datos DSP.
- “Avalon Memory Mapped Tristate Interface” es una interfaz triestado de lectura/escritura basada en direcciones para el soporte de periféricos externos al chip. Así, múltiples periféricos pueden compartir buses de datos y direcciones para reducir el número de terminales de interconexión a la FPGA o el número de rutas en la PCB.
- “Avalon Clock” es una interfaz que envía o recibe señales de reloj y reset para sincronizar interfaces y proveer conectividad de reset.

- “Avalon Interrupt” es una interfaz que permite que los componentes envíen señales de eventos a otros componentes.
- “Avalon Conduit” es una interfaz que proporciona señales para ser llevadas fuera del nivel del SOPC Builder donde poder conectar otros módulos diseñados sobre la FPGA o a los terminales de la misma.

Así, un componente puede incluir una o varias de estas interfaces, o varias instancias de la misma interfaz, posibilitando la interacción de sistemas heterogéneos complejos.

2.4. MICROCONTROLADOR PIC 16F887

El PIC16F887 es un producto conocido de la compañía Microchip. Dispone de todos los componentes disponibles en la mayoría de los microcontroladores modernos. Por su bajo precio, un rango amplio de aplicaciones, alta calidad y disponibilidad, es una solución perfecta para controlar diferentes procesos en la industria, en dispositivos de control de máquinas, para medir variables de procesos, etc.

Este microcontrolador soporta un set de 35 instrucciones tipo RISC, instrucciones necesarias para facilitar su manejo además cuenta con 256 bytes de memoria EEPROM de datos, posee un módulo de comunicación

serial para comunicarse con otros dispositivos y otras características que se describirán a continuación las cuales lo hicieron ideal en la implementación de este proyecto.

2.4.1. DIAGRAMA DE BLOQUES

A continuación se muestra un diagrama de bloques del micro controlador PIC16F887 donde se puede apreciar todos los módulos que integra y la conexión de los mismos.

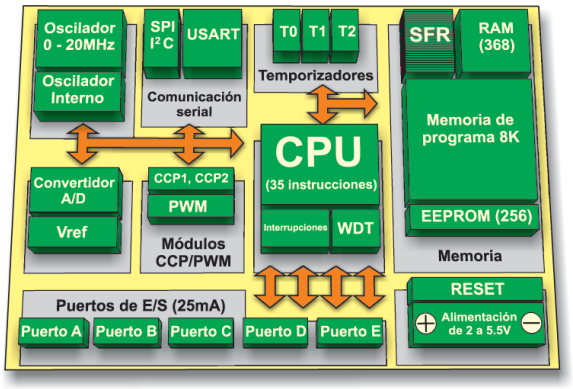


Figura 2.13 - DIAGRAMA DE BLOQUES PIC 16F887 [12]

2.4.2. ARQUITECTURA HARVARD

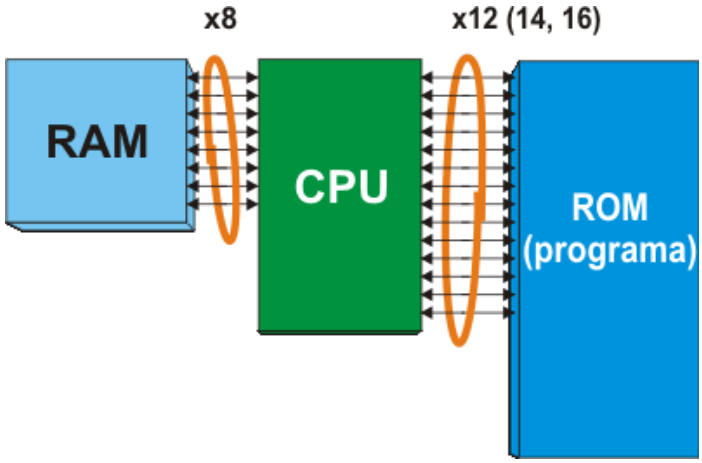


Figura 2.14 - ARQUITECTURA PIC 16F887 [13]

Los microcontroladores que utilizan esta arquitectura disponen de dos buses de datos diferentes. Uno es de 8 bits de ancho y conecta la CPU con la memoria RAM. El otro consiste en varias líneas (12, 14 o 16) y conecta a la CPU y la memoria ROM. Por consiguiente, la CPU puede leer las instrucciones y realizar el acceso a la memoria de datos a la vez. Puesto que todos los registros de la memoria RAM son de 8 bits de ancho, todos los datos dentro del microcontrolador que se intercambian son de la misma anchura. Durante el proceso de la escritura de programa, sólo se manejan los datos de 8 bits. En otras palabras, todo lo que usted podrá cambiar en el programa y a lo que podrá afectar será de 8 bits de ancho. Todos los programas escritos para estos microcontroladores serán almacenados en la memoria ROM interna del

microcontrolador después de haber sido compilados a código máquina. No obstante, estas localidades de memoria ROM no tienen 8, sino 12, 14 ó 16 bits. 4, 6 ó 8 bits adicionales representan una instrucción que especifica a la CPU qué hacer con los datos de 8 bits.

2.4.3. PARTES Y CARACTERÍSTICAS DEL PIC16F887

Las características básicas se muestran a continuación:

Arquitectura RISC

- El microcontrolador cuenta con solo 35 instrucciones diferentes
- Todas las instrucciones son uni-ciclo excepto por las de ramificación

Frecuencia de operación 0-20 MHz

Oscilador interno de alta precisión

- Calibrado de fábrica
- Rango de frecuencia de 8MHz a 31KHz seleccionado por software

Voltaje de la fuente de alimentación de 2.0V a 5.5V

- Consumo: 220uA (2.0V, 4MHz), 11uA (2.0 V, 32 KHz) 50nA (en modo de espera)

Ahorro de energía en el Modo de suspensión

Brown-out Reset (BOR) con opción para controlar por software

35 pines de entrada/salida

- Alta corriente de fuente y de drenador para manejo de LED
- Resistencias pull-up programables individualmente por software
- Interrupción al cambiar el estado del pin

Memoria ROM de 8K con tecnología FLASH

- El chip se puede re-programar hasta 100.000 veces

Opción de programación serial en el circuito

- El chip se puede programar incluso incorporado en el dispositivo destino.

256 bytes de memoria EEPROM

- Los datos se pueden grabar más de 1.000.000 veces

368 bytes de memoria RAM

Convertidor A/D:

- 14 canales

- Resolución de 10 bits

3 temporizadores/contadores independientes

Temporizador perro guardián

Módulo comparador analógico con

- Dos comparadores analógicos
- Referencia de voltaje fija (0.6V)
- Referencia de voltaje programable en el chip

Módulo PWM incorporado

Módulo USART mejorado

- Soporta las comunicaciones seriales RS-485, RS-232 y LIN2.0
- Auto detección de baudios

Puerto Serie Síncrono Maestro (MSSP)

- Soporta los modos SPI e I2C

2.4.4. EMPAQUETADO

Este microcontrolador está disponible en dos tipos de empaquetado, cada empaquetado tiene variantes, especialmente en lo relativo a las dimensiones del espesor del paquete, en general se pueden encontrar paquetes tipo PDIP (Plastic Dual In Line Package), PLCC (Plastic Leaded Chip Carrier), QFP (Quad Flat Package) y SOIC (Small Outline I.C.). El empaquetado seleccionado fue el tipo PDIP ya que es el más fácil de encontrar en el mercado ecuatoriano y por su utilización en proyectos anteriores con excelentes resultados.

En la figura 2.15 se puede apreciar el tipo de empaquetado así como la descripción de los 40 pines que integran este microcontrolador.

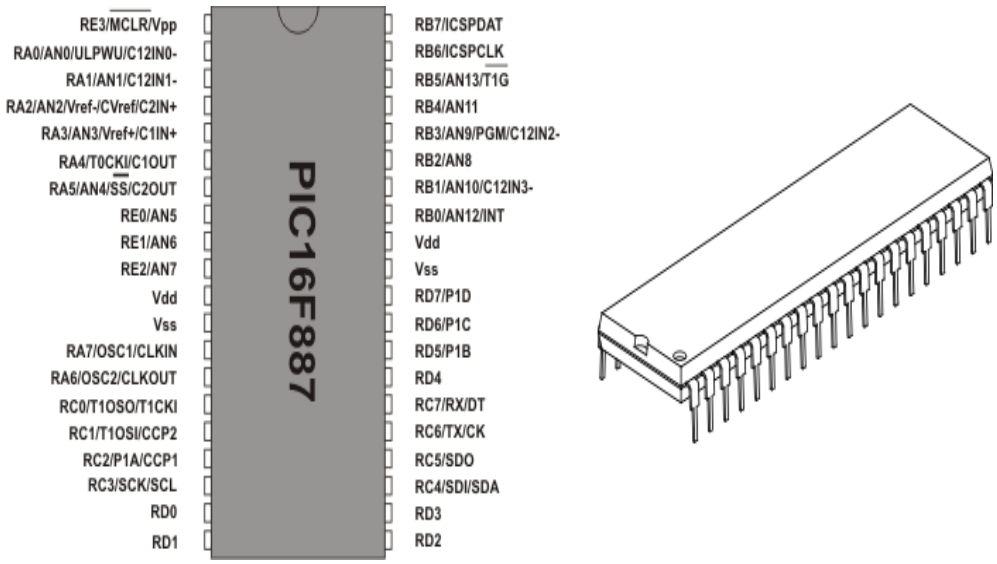


Figura 2.15 - EMPAQUETADO TIPO DIP PIC 16F887 [14]

2.4.5. MÓDULO ADC PIC 16F887

Para que el convertidor A/D alcance su exactitud especificada, es necesario proporcionar un cierto tiempo muerto entre seleccionar una entrada analógica específica y la medición misma. Se utiliza una ecuación para hacer cálculo de tiempo de adquisición con precisión, cuyo valor mínimo es de 20uS aproximadamente

El módulo del convertidor A/D dispone de las siguientes características:

- El convertidor genera un resultado binario de 10 bits utilizando el método de aproximaciones sucesivas y almacena los resultados de conversión en los registros ADC (ADRESL y ADRESH);
- Dispone de 14 entradas analógicas separadas;
- La resolución mínima o calidad de conversión se puede ajustar a diferentes necesidades al seleccionar voltajes de referencia Vref- y Vref+.

En la figura 2.16 se muestra el diagrama de bloques del módulo ADC presente en el PIC 16F887

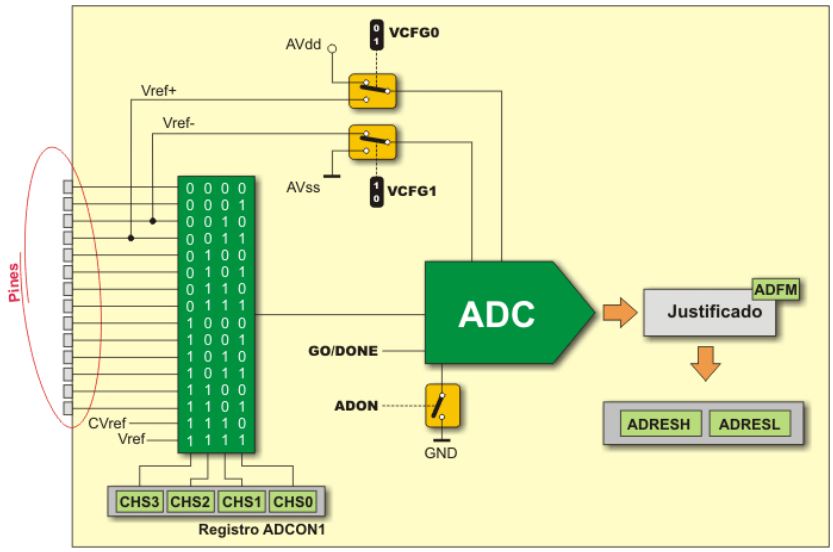


Figura 2.16 - MÓDULO ADC PIC 16F887 [15]

El funcionamiento del convertidor A/D está bajo el control de los bits de cuatro registros:

- ADRESH Registro alto del resultado de la conversión A/D;
- ADRESL Registro bajo del resultado de la conversión A/D;
- ADCON0 Registro de control 0; y
- ADCON1 Registro de control 1.

2.4.5.1. REGISTROS ADRESH Y ADRESL

El resultado obtenido después de convertir un valor analógico en digital es un número de 10 bits que se almacenará en los registros ADRESH y ADRESL. Hay dos maneras de manejarlo: justificación a la izquierda y a

la derecha tal como se observa en la figura 2.17, que simplifica en gran medida su uso. El formato del resultado de la conversión depende del bit ADFM del registro ADCON1. En caso de que no se utilice el convertidor A/D, estos registros se pueden utilizar como registros de propósito general.

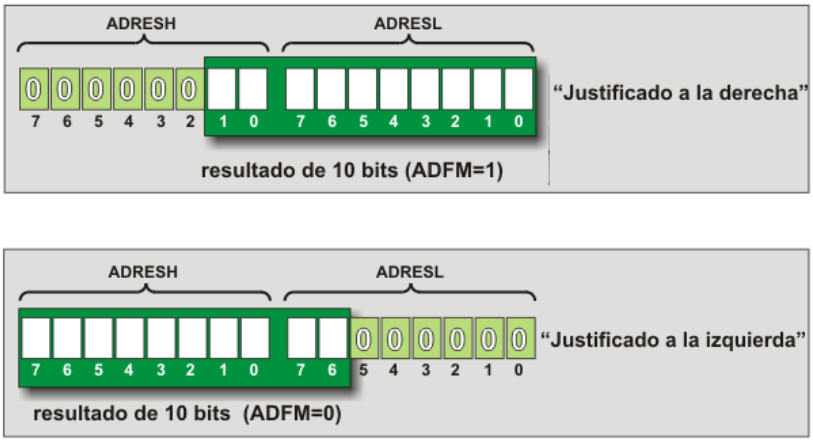


Figura 2.17 - REGISTROS DE ALMACENAMIENTO DE LOS DATOS CONVERTIDOS [16]

2.4.6. MÓDULO DE COMUNICACIÓN SERIE

El USART es uno de los primeros sistemas de comunicación serie. Las versiones nuevas de este sistema están actualizadas y se les denomina un poco diferente - EUSART.

2.4.6.1. EUSART

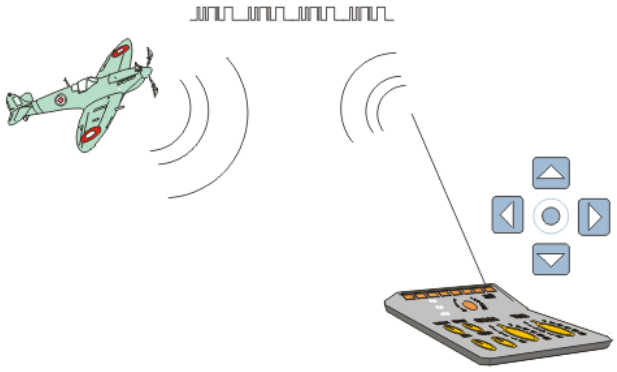


Figura 2.18 - COMUNICACIÓN ENTRE DISPOSITIVOS DIFERENTES [17]

La figura 2.18 describe simplificada mente el proceso de comunicación entre dos dispositivos diferentes usando el protocolo de comunicación serie.

El módulo Transmisor/Receptor Universal Síncrono/Asíncrono mejorado (Enhanced Universal Synchronous Asynchronous Receiver Transmitter - EUSART) es un periférico de comunicación serie de entrada/salida. Asimismo es conocido como Interfaz de comunicación serie (Serial Communications Interface - SCI). Contiene todos los generadores de señales de reloj, registros de desplazamiento y búferes de datos necesarios para realizar transmisión de datos serie de entrada/salida, independientemente de la ejecución de programa del dispositivo. Como

indica su nombre, aparte de utilizar el reloj para la sincronización, este módulo puede establecer la conexión asíncrona.

El EUSART integrado en el PIC16F887 posee las siguientes características:

- Transmisión y recepción asíncrona en modo Full-duplex;
- Caracteres de anchura de 8 – 9 bits programables;
- Detección de dirección en modo de 9 bits;
- Detección de errores por saturación del búfer de entrada; y
- Comunicación Half Duplex en modo síncrono.

2.4.6.2. EUSART EN MODO ASÍNCRONO

El EUSART transmite y recibe los datos utilizando la codificación de no retorno a cero - NRZ (non-return-to-zero). Como se muestra en la figura 2.19, no se utiliza una señal de reloj y los datos se transmiten de forma muy simple:

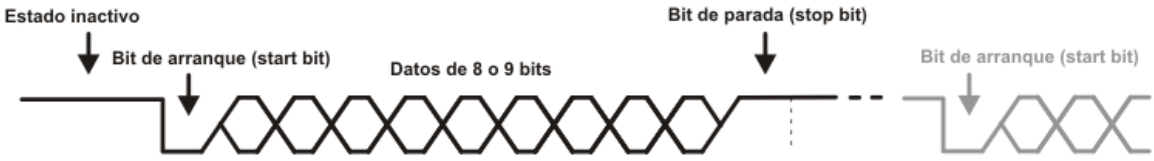


Figura 2.19 - MÉTODO DE TRANSMISIÓN SERIAL [18]

Cada dato se transmite de la siguiente forma:

- En estado inactivo la línea de datos permanece en estado alto (1);
- Cada transmisión de datos comienza con un bit de arranque (START), el cual, siempre es cero (0);
- Cada dato tiene un ancho de 8 o 9 bits (primero se transmite el bit menos significativo - LSB); y
- Cada transmisión de datos termina con un bit de parada (STOP), el cual, siempre es uno (1) La siguiente figura muestra cómo conectar de manera habitual un microcontrolador PIC que utiliza el módulo EUSART.

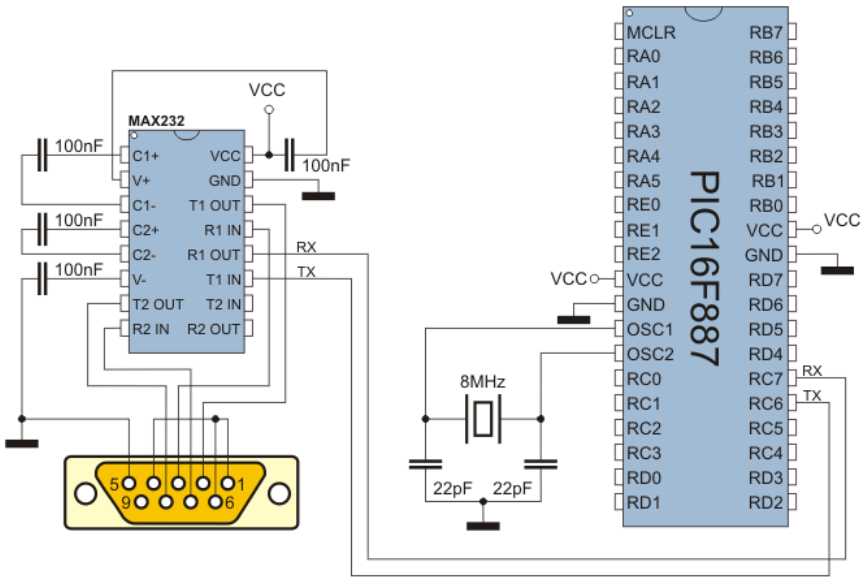


Figura 2.20 - ESQUEMÁTICO DE CONEXIÓN RS232 [19]

La figura 2.20 muestra el diagrama esquemático utilizado en este proyecto, para la conexión entre el PIC 16F887 y el conector DB9. Debido a que el PIC nos proporciona niveles TTL +5V o -5V es necesario el uso de un driver que nos permita adaptar dichos niveles de voltajes a los niveles establecidos por la norma RS232. El driver que se observa en la figura 20 corresponde a un MAX232 el cual permite adaptar los niveles TTL del PIC a niveles de +15V o -15V especificados en la norma RS232.

2.5. MÓDULO XBEE

Los módulos Xbee trabajan mediante el protocolo Zigbee. Zigbee es un protocolo de comunicaciones inalámbrico basado en el estándar de

comunicación para redes inalámbricas IEEE_802.15.4. Zigbee permite que dispositivos electrónicos de bajo consumo puedan realizar sus comunicaciones inalámbricas. Las comunicaciones Zigbee se realizan en la banda libre de 2.4GHz. A diferencia de bluetooth, este protocolo no utiliza FHSS (Frequency hooping), sino que realiza las comunicaciones a través de una única frecuencia; es decir, de un canal. Normalmente puede escogerse un canal de entre 16 posibles. El alcance depende de la potencia de transmisión del dispositivo así como también del tipo de antenas utilizadas (cerámicas, dipolos, etc) El alcance normal con antena dipolo en línea vista es de aproximadamente de 100m y en interiores de unos 30m. La velocidad de transmisión de datos de una red Zigbee es de hasta 256kbps. Una red Zigbee la pueden formar, teóricamente, hasta 65535 equipos; debido a que el protocolo está preparado para poder controlar en la misma red esta cantidad enorme de dispositivos.

2.5.1. APLICACIONES

Los módulos Xbee, pueden ser ajustados para usarse en redes de configuración punto a punto, punto-multipunto o peer-to-peer. Un ejemplo se muestra en la Figura 2.21, donde se muestra una conexión multipunto, con un coordinador, conectado a varios nodos.

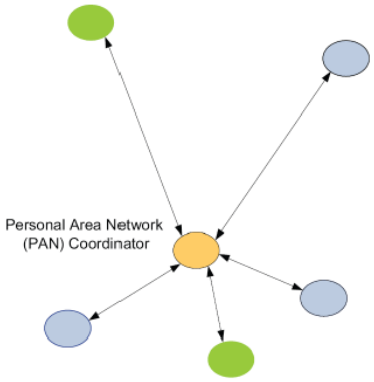


Figura 2.21 - COORDINADOR PAN CON MÚLTIPLES NODOS [20]

En la Figura 2.22 se observan los elementos del XBEE. El Chip de la antena en la parte superior, el conector para la antena RF, y el conector para una antena integrada Whip.

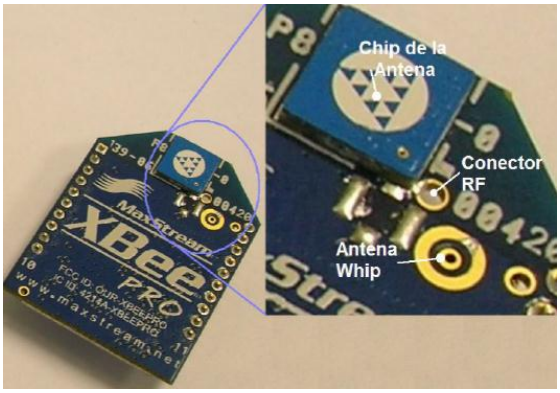


Figura 2.22 - ELEMENTOS DEL XBEE [21]

Los módulos Xbee PRO de la Serie 1 se diferencian de los módulos de la serie 2 en la capacidad de alcance, permitiendo en algunos casos doblar la distancia de transmisión, ya que poseen una mayor potencia en la

señal. Con los módulos Xbee PRO de la Serie 2, es posible crear redes más complejas, como las llamadas MESH. Estas permiten acceder a un punto remoto, utilizando módulos intermedios para llegar como routers. En la figura 2.23 se muestra la configuración de una red MESH.

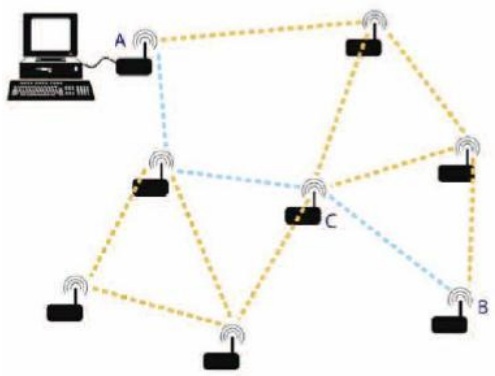


Figura 2.23 - RED MESH IMPLEMENTADA CON MÓDULOS XBEE CARACTERISTICAS [22]

Los módulos Xbee tienen las siguientes características:

- Alimentación de 3.3VDC
- Velocidad de transmisión máxima de 250kbps
- Rango de transmisión de 100m en línea de vista y 30m con obstáculos
- 6 entradas analógicas para conversión ADC a una resolución de 10 bits
- 8 entradas digitales

- Soporta comandos AT

2.6. HERRAMIENTAS DE DESARROLLO ASISTIDA POR COMPUTADOR

2.6.1. SOFTWARE QUARTUS II

Este software incluye el procesador NIOS II y la herramienta de creación SOPC para facilitar las implementaciones de sistemas complejos. Además, el software ofrece una interfaz gráfica de usuario así como una interfaz de comandos para cada una de las fases de diseño. La interfaz gráfica del software se presenta a continuación:

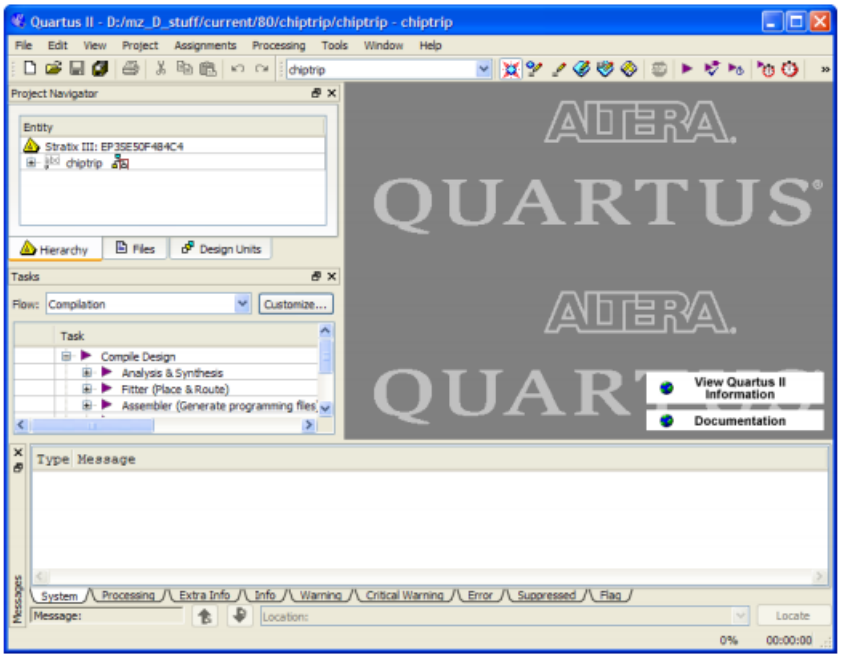


Figura 2.24 - VENTANA DE TRABAJO QUARTUS II

Quartus II nos permite realizar varias tareas al momento de la simulación de un circuito lógico, desde la descripción de los componentes del circuito en lenguaje VHDL, hasta la simulación de diagramas de tiempo del comportamiento del circuito. Quartus II nos ofrece muchas ventajas al momento de desarrollar un proyecto usando el microprocesador NIOS II ya que con la herramienta integrada SOPC BUILDER podemos generar el hardware que requerimos para la implementación.

SOPC BUILDER

Figura 2.25 - VENTANA DE INICIO SOPC BUILDER

SOPC builder es una herramienta de desarrollo que permite definir y generar sistemas SOPC (Sistemas Completos en un chip programable) ahorrando tiempo comparado con los métodos tradicionales. Esta

herramienta de desarrollo está integrada a Quartus II, la figura 2.25 muestra la ventana de inicio de SOPC Builder.

SOPC Builder automatiza la tarea de integrar los componentes de hardware del sistema. Al usar los métodos tradicionales de diseño se debe escribir manualmente los módulos HDL para conectar los elementos del sistema. Mediante SOPC builder, se especifican los componentes del sistema que se está implementando y genera archivos HDL de forma automática en los cuales se especifica la interconexión lógica de los componentes del sistema.

2.6.2. NIOS II SBT

NIOS II SBT para Eclipse es una herramienta fácil de usar con una interfaz gráfica que automatiza la gestión de las librerías generadas mediante SOPC Builder. Además integra un editor de texto, un depurador, programador NIOS II flash y el programador de Quartus II. La aplicación ofrece plantillas por defecto que vienen incluidas en la interfaz gráfica para comodidad de los programadores al momento de desarrollar el software que correrá en el sistema a implementar.

Nios II SBT para Eclipse se basa en la estructura del Framework Eclipse y Eclipse Development Toolkit Plugins (CDT). En la figura 2.26 se presenta la interfaz gráfica para el usuario.

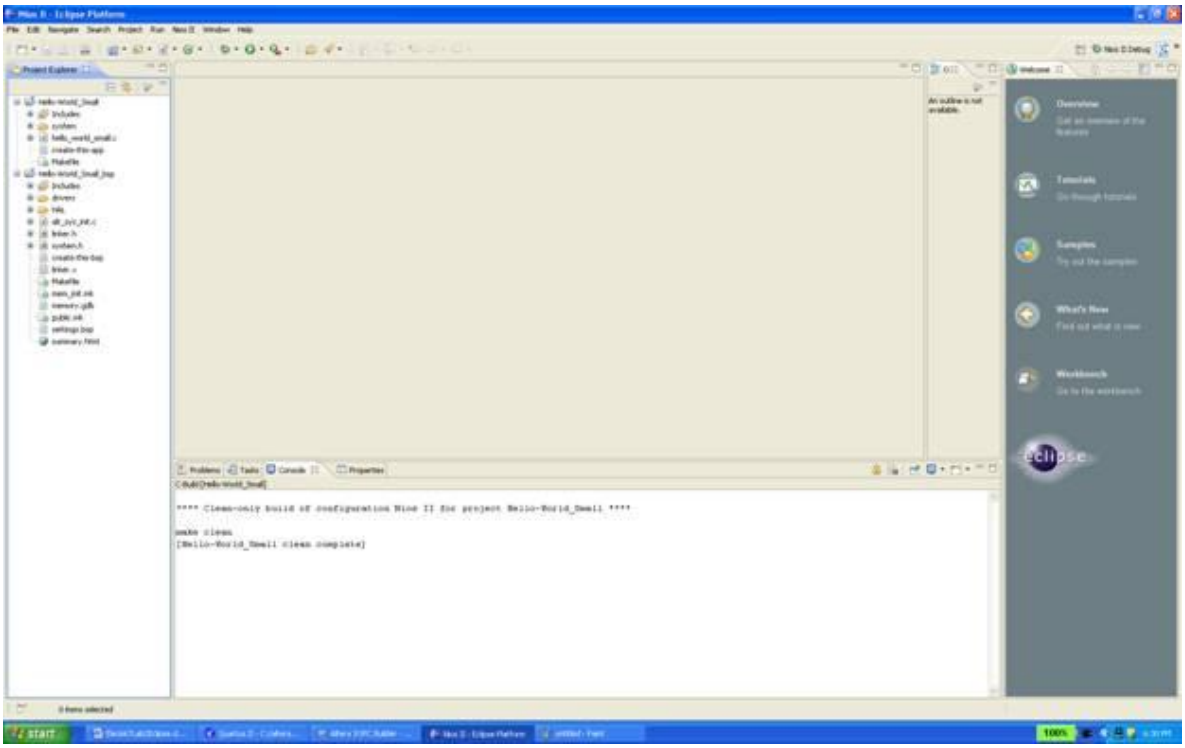


Figura 2.26 - VENTANA DE TRABAJO DE ECLIPSE NIOS II SBT

En la parte izquierda se muestra el explorador de carpetas del sistema que se está implementado, en la parte central se muestra el ambiente de trabajo donde se escribe el código en lenguaje C que mandara al microprocesador NIOS II y los periféricos conectados a él.

2.6.3. PROTEUS

Es una compilación de programas de diseño y simulación electrónica, desarrollado por Labcenter Electronics que consta de dos programas principales: Ares e Isis, y el módulo VSM . La figura 2.27 muestra la ventana de inicio del programa Proteus.

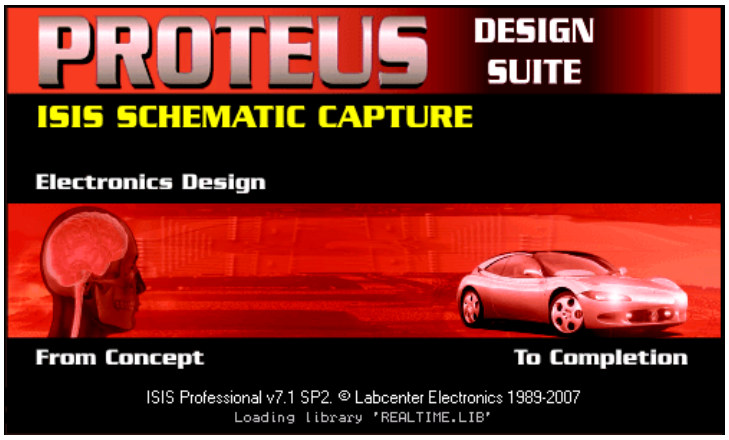


Figura 2.27 - DESIGN SUITE PROTEUS

ISIS (“Intelligent Schematic Input System”) que es el módulo de captura de esquemas.

VSM (“Virtual System Modelling”) es el módulo de simulación.

ARES (“Advanced Routing Modelling”) es el módulo para realización de circuitos impresos (PCB).

PROSPICE es la versión SPICE incluida en PROTEUS y desarrollada a partir del modelo Berkeley, con extensiones para la simulación analógica y digital conjunta y la animación de circuitos.

El módulo ISIS es un programa que nos permite dibujar sobre un área de trabajo un circuito que posteriormente podremos simular. Las utilidades que posee este software son entre otras:

- Librerías de componentes.
- Conexión automática entre 2 puntos denominado Netlist, compatible con la mayoría de los programas de realización de PCB.
- Enumeración automática de componentes.

2.6.4. ALTIUM

Altium Designer es un conjunto de programas para el diseño electrónico en todas sus fases y para todas las disciplinas, ya sean esquemas, simulación, diseño de circuitos impresos, implementación de FPGA, o desarrollo de código para microprocesadores. La figura 2.28 muestra la ventana de inicio de Altium Designer.

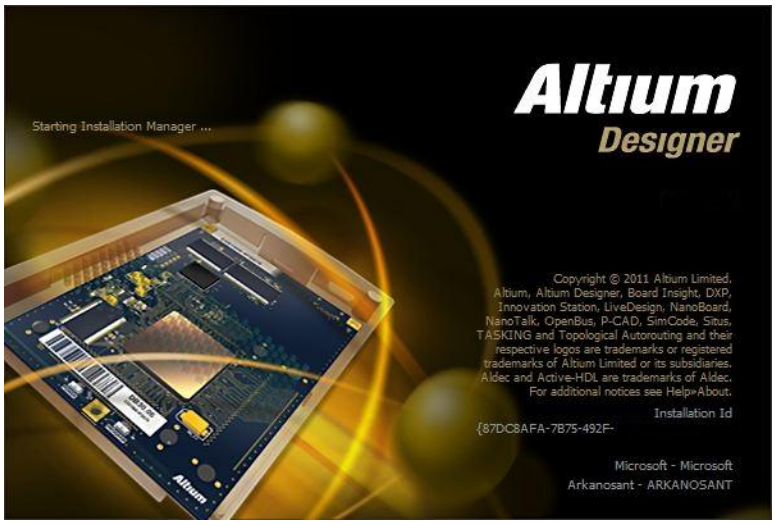


Figura 2.28 - ALTIUM DESIGNER

Incluye la parte básica dónde se crean los distintos proyectos para los distintos objetivos, ya sea un esquema para un circuito impreso, un programa para un microprocesador o un diseño para ser simulado. Las características más relevantes de este módulo son:

- Conexión a base de datos
- Visores de PCB y Gerbers
- Simulador mixto SPICE
- Signal integrity (reflexiones y diafonía)basado en esquemas
- Síntesis y simulación FPGA.
- Conexión JTAG

2.6.5. MIKROC

El desarrollo de código para dispositivos PIC ya que incluye un compilador avanzado, librerías y herramientas adicionales que ayudarán en el desarrollo de una aplicación.

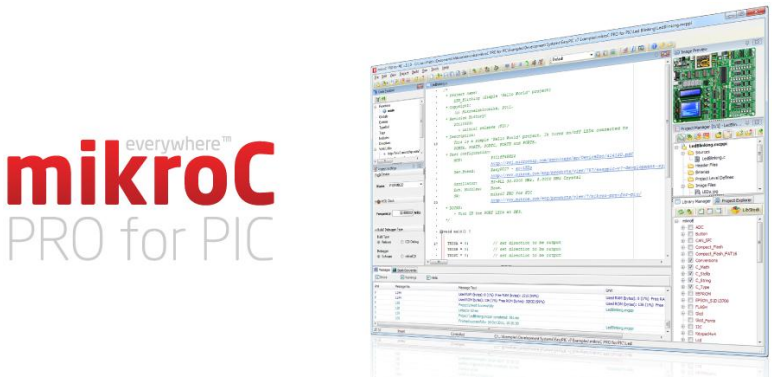


Figura 2.29 - SOFTWARE DE PROGRAMACIÓN MIKROC

MikroC Pro organiza aplicaciones en los proyectos que consisten en un solo fichero con extensión .mcppi o en uno o más ficheros. Los cuales se pueden compilar solo si forman parte del proyecto, ficheros con extensión .c. Los ficheros fuentes son denominados cabeceras de programa. MikroC Pro permite manejar varios proyectos a la vez.

Un fichero de proyecto contiene lo siguiente:

- Nombre del proyecto y la descripción opcional.
- Tipo de microcontrolador.

- Frecuencia de reloj del microcontrolador.
- Lista de ficheros fuentes del proyecto.
- Ficheros binarios (*.mcl)

CAPÍTULO 3

3. DISEÑO E IMPLEMENTACIÓN

3.1. ANÁLISIS DE REQUERIMIENTOS

Para llevar a cabo el funcionamiento completo del Datalogger se debe cumplir con ciertas especificaciones establecidas previamente las cuales nos servirán de guía a lo largo de todo el proceso de diseño e implementación final. Estas especificaciones se detallan a continuación:

- El núcleo central del proyecto es el microprocesador NIOS II.
- La información es obtenida de sensores.

- Implementar un protocolo de comunicación.
- Visualizar los datos recibidos.
- La información debe ser almacenada en una tarjeta SD.
- Los datos almacenados deben ser interpretados mediante un software.

La obtención de la información proporcionada por los sensores externos se la realiza mediante un microcontrolador de la familia microchip “16F887”, el mismo que posee los módulos requeridos para el procesamiento de la información que será enviada al microprocesador NIOS II.

Debido a que el microprocesador NIOS II se encuentra embebido en una FPGA, que está integrada a la tarjeta DE2 junto con los periféricos de entrada y salida de datos, fue necesario establecer un protocolo de comunicación entre el microcontrolador y la tarjeta DE2. Se utilizó la comunicación serial estándar RS232 que nos proporcionas estándares previamente establecidos como; velocidad de transferencia de datos, el control que utiliza dicha transferencia, los niveles de voltajes, los conectores, etc.

Los datos recibidos por la tarjeta DE2 son visualizados y almacenados en tiempo real, uno de los periféricos utilizados para visualizar los datos obtenidos por los sensores fue la pantalla LCD que está en la tarjeta. Para llevar un registro de los datos recibidos se utilizó una tarjeta SD, la misma que debe ser conectada al puerto SDCard de la tarjeta DE2, para luego comenzar con el almacenamiento de los datos recibidos en un archivo de texto.

Como fase final se realiza un análisis con los archivos almacenados en la tarjeta SD, por lo que se decidió elaborar un software que permita ingresar el archivo creado en la tarjeta SD y genere una gráfica que servirá para obtener conclusiones del comportamiento de los sensores.

3.2. DIAGRAMA FUNCIONAL

En la figura 3.1 se muestra un diagrama funcional de nuestro proyecto, que describe cada etapa que se utilizó en la implementación del mismo. A continuación se da una breve explicación del funcionamiento de cada etapa.

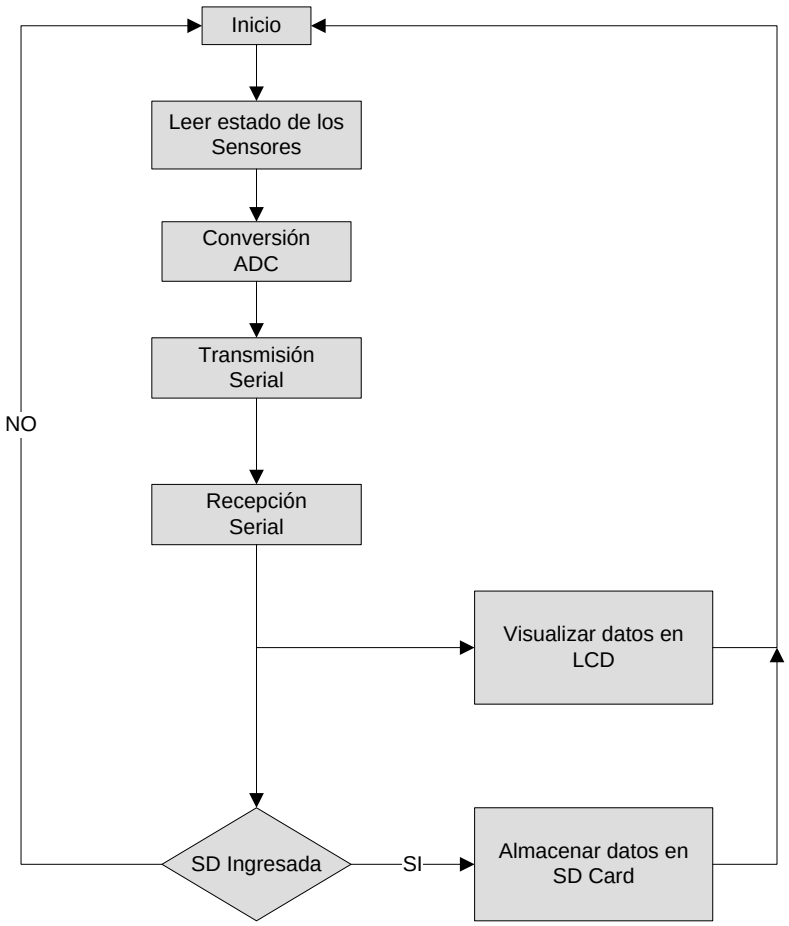


Figura 3.1 - DIAGRAMA FUNCIONAL DE LA SOLUCIÓN IMPLEMENTADA

Una vez iniciado el sistema, se debe obtener los respectivos valores proporcionados por los sensores externos, cabe mencionar que los sensores entregan valores analógicos ya que se seleccionó como parámetro de interés a la temperatura.

Como segunda etapa se realiza una digitalización de los datos obtenidos mediante sensores de temperatura con el objetivo de preparar cada dato para que sea transmitido a la tarjeta DE2. Una vez realizada la digitalización de los datos proporcionado por cada sensor de temperatura se procede a realizar la transmisión de los mismos, se utilizó el protocolo de comunicación serial estándar RS232 con los siguientes parámetros:

- Velocidad de Transmisión: 9600 bps
- Niveles de Voltajes Requeridos para la transmisión/recepción: +15V,-15V
- Tipo de Conector entre los dos módulos: Conector DB9

Los datos proporcionados por los sensores de temperatura son recibidos por la tarjeta DE2, se los visualiza en tiempo real mediante la LCD permitiendo al usuario advertir cualquier error que presente durante la recepción de los mismos.

Como etapa final se tiene el proceso de almacenamiento de los datos recibidos en una tarjeta SD, el usuario puede elegir en que momento comenzar con el proceso de almacenamiento siempre y cuando la tarjeta se encuentre insertada en el puerto SD de la tarjeta DE2. Se tiene que accionar un switch para iniciar el proceso de almacenamiento de los

datos recibidos, cada accionamiento implica una nueva lectura y se genera un nuevo archivo con los valores correspondientes a cada sensor de temperatura.

3.3. DIAGRAMA DE BLOQUES

Para representar de manera más clara el funcionamiento del Datalogger, se realizó un diagrama de bloques. En la figura 3.2 se puede observar lo siguiente:

- Obtención de los valores analógicos proporcionados por los sensores de temperatura
- Etapa de digitalización mediante el módulo ADC del PIC16F887,
- Transmisión serial de los valores digitalizados usando el PIC 16F887.
- Recepción de los valores transmitidos por el PIC en la tarjeta DE2
- Visualización de los datos mediante la LCD
- Almacenamiento de los datos recibidos en una tarjeta SD

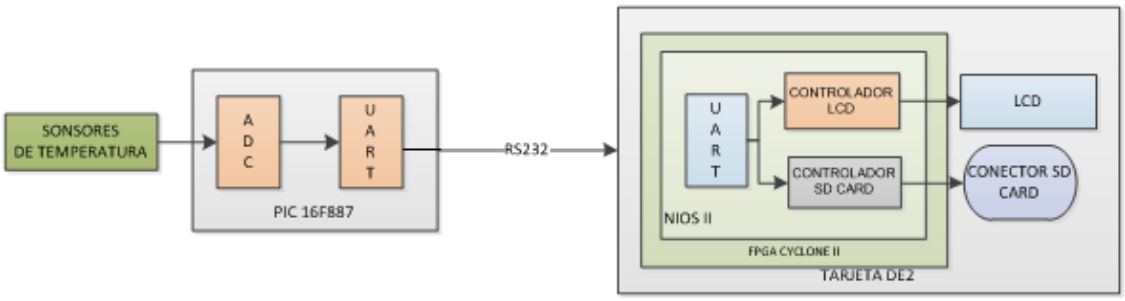


Figura 3.2 - DIAGRAMA DE BLOQUES DEL DATALOGGER

En el diagrama de bloques que se puede observar que se presentan dos etapas bien definidas, una conformada por el PIC 16F887 que corresponde al módulo transmisor y la otra etapa conformada por la tarjeta DE2 que conforma el módulo receptor. A continuación se describen con más detalle el diseño de ambos módulos.

3.4. ESQUEMÁTICO DEL MÓDULO TRANSMISOR

El diseño del hardware que posee el módulo transmisor se lo ha desarrollado de tal forma que realice lo siguiente:

- Obtener los valores analógicos proporcionados por los sensores de temperatura
- Digitalizar valores analógicos
- Transmisión serial de los valores digitalizados

En la figura 3.3 se muestra el esquemático que se realizó para llevar a cabo la implementación del módulo de transmisor, está conformado por 4 etapas:

- Alimentación
- Sensores
- Red de Estabilidad
- Digitalización-Transmisión realizadas con el PIC 16F887

Las 4 etapas mencionadas anteriormente son detalladas a continuación:

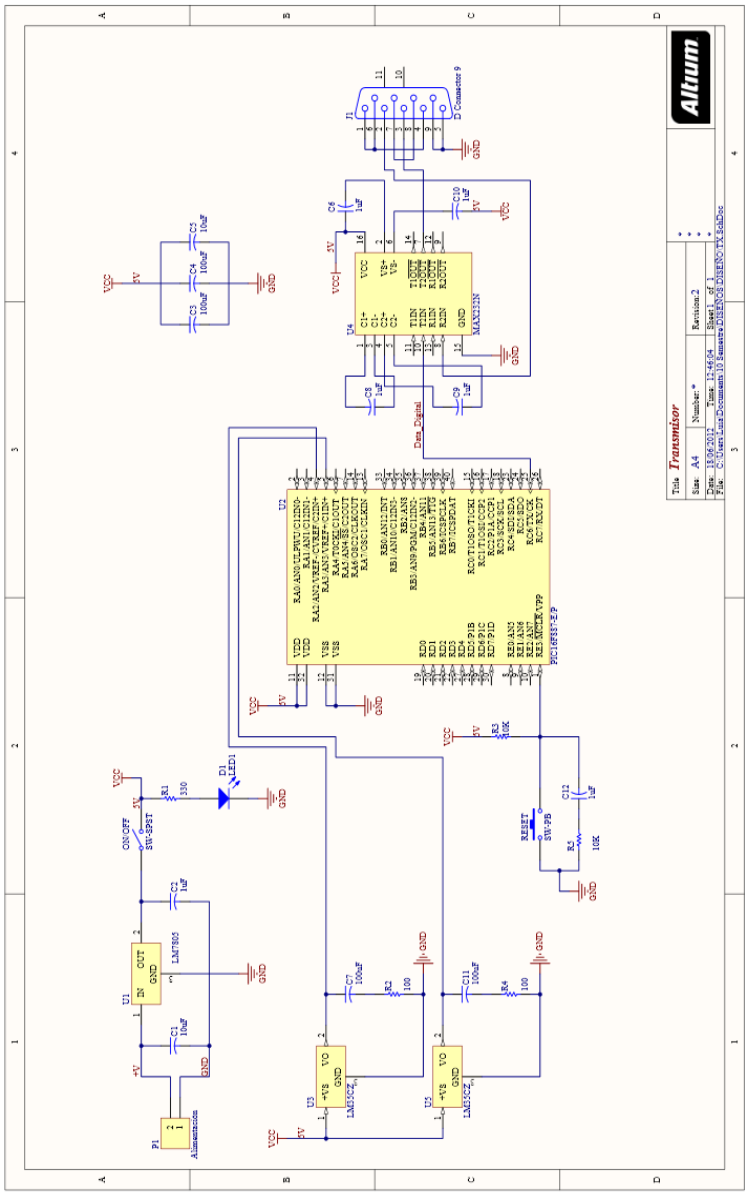


Figura 3.3 - ESQUEMÁTICO MÓDULO TRANSMISOR

3.4.1. ALIMENTACIÓN

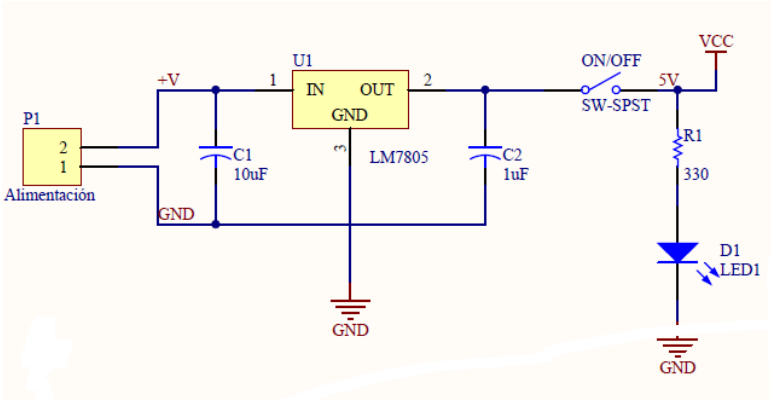


Figura 3.4 - ETAPA DE ALIMENTACIÓN

La etapa de alimentación es de gran importancia en todo hardware ya que permite el adecuado desempeño de los componentes electrónicos que integran el diseño de dicho hardware. Para el módulo transmisor se realizó el diseño de una fuente de alimentación la cual se puede observar en la figura 3.4. Esta fuente de alimentación tiene las siguientes características:

- Etapa de filtrado para el voltaje de entrada
- Voltaje de salida Regulado de 5VDC

Para cumplir con las características mencionadas anteriormente se implementó una configuración usando un regulador de voltaje positivo

LM7805, el cual nos entrega un voltaje de salida de 5VDC siempre y cuando se cumpla la siguiente condición:

$$(3.1)\ V_{inmin} = V_o + 2V$$

Esta expresión nos indica que para tener un voltaje a la salida del regulador de 5VDC, nuestro voltaje de entrada mínimo deberá ser de 7VDC. Para la etapa de filtrado se coloca en el pin de entrada del regulador (Pin 1) un capacitor de $10uF$ con el fin de filtrar cualquier ruido proveniente de la alimentación principal, en el pin de salida del regulador (Pin 3) se coloca un capacitor de $1uF$ con el objetivo de obtener un voltaje de salida completamente regulado.

3.4.2. SENSORES

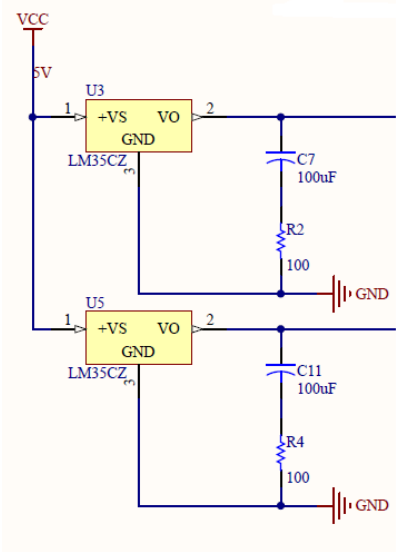


Figura 3.5 - SENSORES DE TEMPERATURA LM35

En la figura 3.5 se puede observar la configuración implementada para el funcionamiento de los sensores de temperatura LM35, esta configuración es proporcionada por el fabricante en la hoja de especificaciones. Las características del sensor LM35 se muestran a continuación:

- Escala de medición en Celsius
- Variación por cada grado centígrado de $\mathbf{10mV}$
- Rango de medición de $-\mathbf{55C^0}$hasta $+\mathbf{150C^0}$
- Voltaje de operación de 4V hasta 30V

La red RC que se observa en la figura 3.5 sirve para que los sensores capten temperaturas mayores a 0°, si se desea que los sensores capten temperatura por debajo de los 0° se debe implementar una configuración diferente, que se puede consultar en la hoja de especificaciones.

3.4.3. RED DE ESTABILIDAD

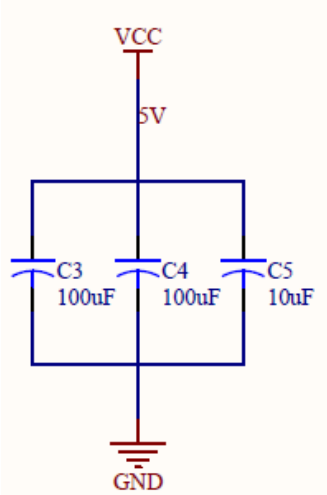


Figura 3.6 - RED DE ESTABILIDAD

La red de estabilidad sirve para garantizar que el módulo transmisor esté alimentado con 5VDC además de eliminar corrientes parásitas que se pueden presentar a la entrada de los pines de alimentación del microcontrolador, esta red esa formada por 3 capacitores paralelo $C3 = 100uF, C4 = 100uF\ y\ C5 = 10uF$ dando como resultado una capacitancia total de $(3.2)\ C_T = C3 + C4 + C5 = 210uF$.

3.4.4. DIGITALIZACIÓN - TRANSMISIÓN

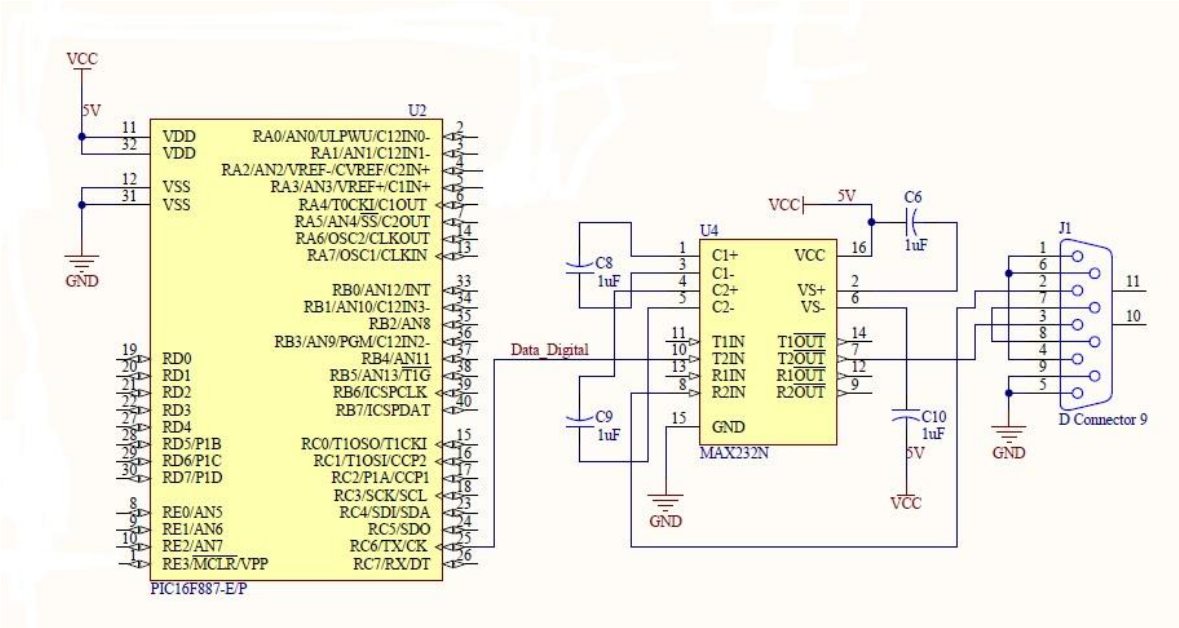


Figura 3.7 - ETAPA DE TRANSMISIÓN

En esta etapa se realiza el proceso de digitalización y transmisión de los datos proporcionados por los sensores de temperatura. La figura 3.7 describe el diagrama esquemático implementado el mismo que consta de:

- Microcontrolador PIC16F887
- MAX232
- Conector DB9 Hembra.

Los datos obtenidos de los sensores son procesados por el microcontrolador mediante una conversión ADC con 10 bits de resolución, para luego ser transmitidos hacia la tarjeta DE2 a una velocidad de

9600bps. La conexión entre la tarjeta DE2 y el microcontrolador se la realiza mediante un cable serial, el cual debe ir conectado a un conector DB9.

El MAX232 se encarga de ajustar los niveles TTL entregados por el PIC a través del PIN 25 a los niveles requeridos por el estándar RS232 +12VDC o -12VDC, con el fin de obtener una comunicación entre la tarjeta DE2 y el módulo transmisor.

3.5. DISEÑO DEL HARDWARE - MÓDULO RECEPTOR EN SOPC BUILDER

SOPC BUILDER es una herramienta de desarrollo de hardware que es usado para implementar sistemas embebidos que usan el microprocesador NIOS II y viene integrada en el software Quartus II.

El microprocesador NIOS II es propio de Altera, es definido en lenguaje de descripción de hardware, el mismo que puede ser implementado en dispositivos FPGA de Altera usando Quartus II. Para lograr un sistema embebido usando el microprocesador NIOS II, es necesario añadir módulos que interactúen en conjunto con este microprocesador como por ejemplo memorias, interfaces de entrada/salida, interfaces de comunicación entre otras.

En la figura 3.8 se muestra el diseño del módulo receptor implementado en SOPC Builder y a continuación se describen las partes más importantes de este diseño.

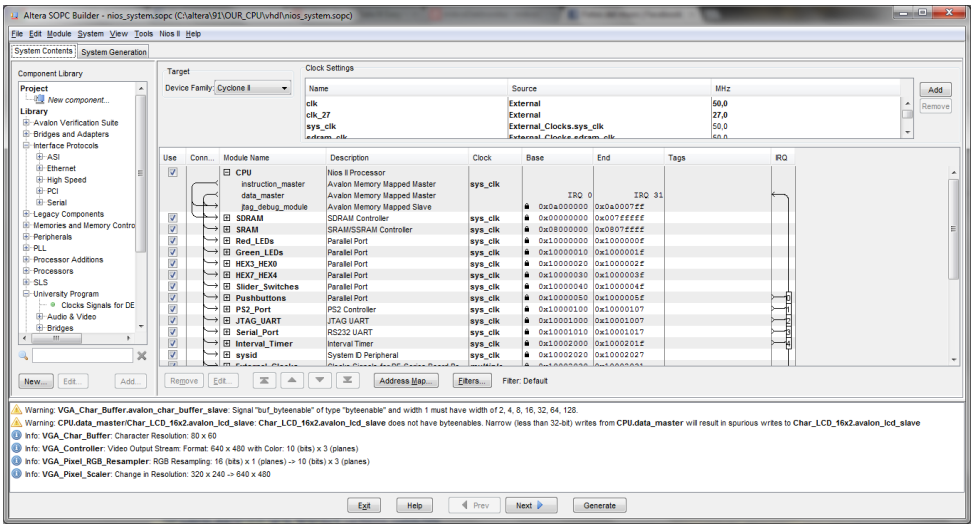


Figura 3.8 - DISEÑO DEL MÓDULO RECEPTOR EN SOCP BUILDER

3.5.1. MICROPROCESADOR NIOS II

Todo sistema embebido debe ser controlado mediante un procesador, el procesador es el encargado de interactuar con diferentes interfaces conectadas a él, procesar la información que recibe y generar las diferentes señales de control para cada interfaz.

En la figura 3.9 se muestra la configuración del procesador NIOS II con las siguientes especificaciones:

- Arquitectura RISC
- Bus de datos de 32 bits
- Multiplicador implementado en Hardware
- Divisor implementado en Hardware
- Frecuencia de operación de 50MHz

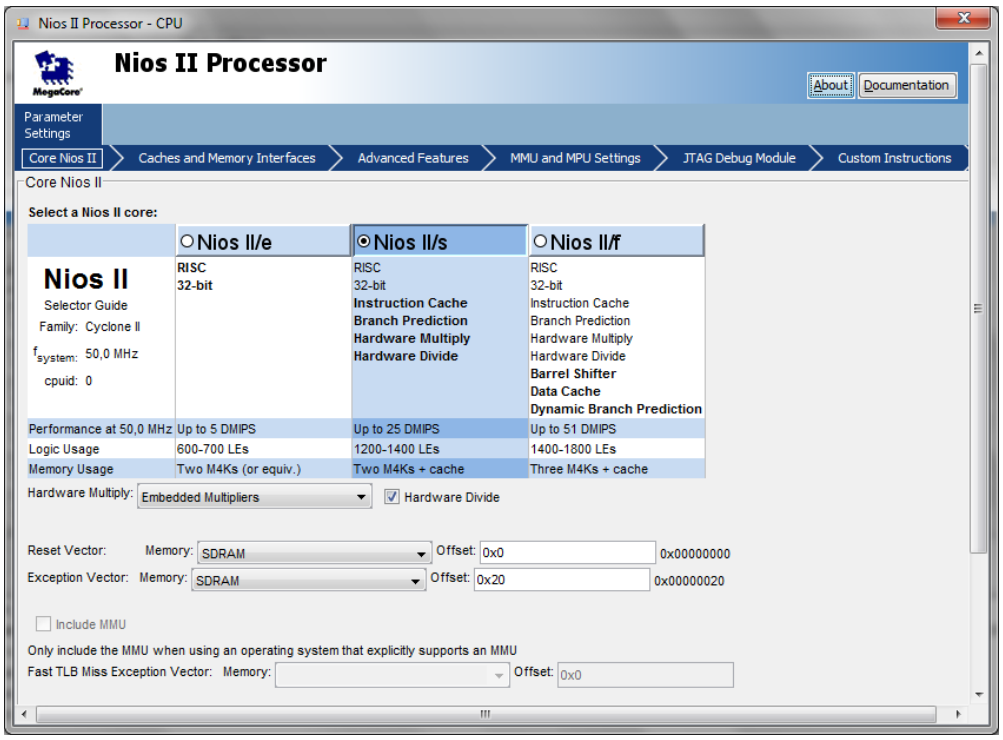


Figura 3.9 - CONFIGURACIÓN DEL MICROPROCESADOR NIOS II

3.5.2. DIPLAYS DE 7 SEGMENTOS

Funciona como un indicador para saber si los datos que se reciben se están almacenando. En sus segmentos se dibuja la palabra ON cuando los datos se están almacenando correctamente en la tarjeta SD, cuando se dibuja la palabra OFF indica que los datos no se están almacenando y solo se están visualizando en la pantalla LCD. La figura 3.10 muestra la configuración de 4 displays de 7 segmentos en SOPC Builder.

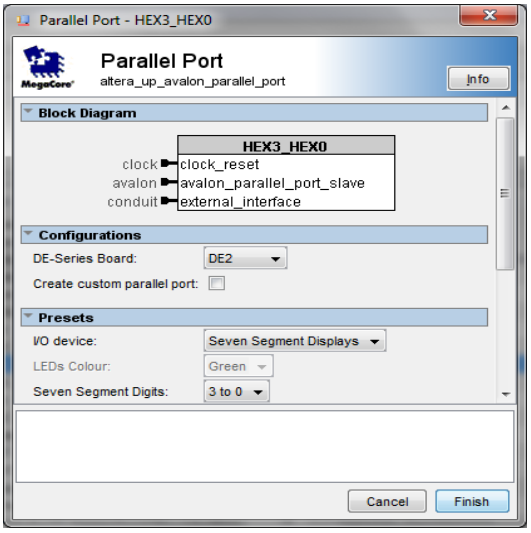


Figura 3.10 - DISPLAYS DE 7 SEGMENTOS

3.5.3. SWITCHES DESLIZANTES

Los switches deslizantes implementados en nuestro sistema son utilizados para indicarle al procesador NIOS II que debe enviar señales de control a la interfaz SDCard y comenzar con el almacenamiento de los

datos recibidos. Por cada accionamiento del respectivo switch se genera un archivo de texto diferente con los respectivos valores recibidos por la tarjeta DE2. En la figura 3.11 se muestra la configuración de los switches deslizantes en SOPC Builder.

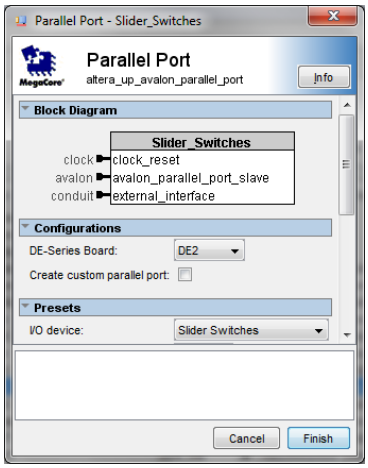


Figura 3.11 - SWITCHES DESLIZANTES

3.5.4. PUERTO SERIAL RS232

La interfaz serial es la que permite la conexión entre el módulo transmisor y el módulo receptor mediante el estándar RS232. En la figura 3.12 se muestra la configuración de la interfaz serial de la tarjeta DE2 en SOPC builder la misma que presenta los siguientes parámetros para realizar la conexión de forma correcta:

- Velocidad de transmisión a 9600 bps

- 8 bits para los datos
- 1 bit de parada

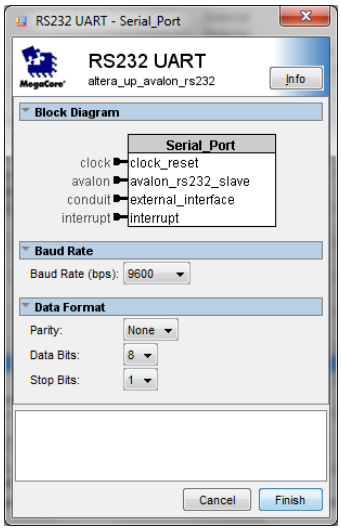


Figura 3.12 - CONFIGURACIÓN DEL MÓDULO UART

3.5.5. LCD 16X2

Los datos que recibe la tarjeta DE2 son visualizados en la pantalla LCD, el control de la visualización de los datos se lo realiza por medio del procesador NIOS II en conjunto con el módulo SDRAM. En la figura 3.13 se muestra la configuración de la interfaz LCD en SOPC builder. En SOPC builder se indica los parámetros de configuración a nivel de hardware para la interfaz LCD, el control de la visualización de los datos se lo realiza mediante software usando las librerías que se generan una vez terminado el diseño en SOPC builder.

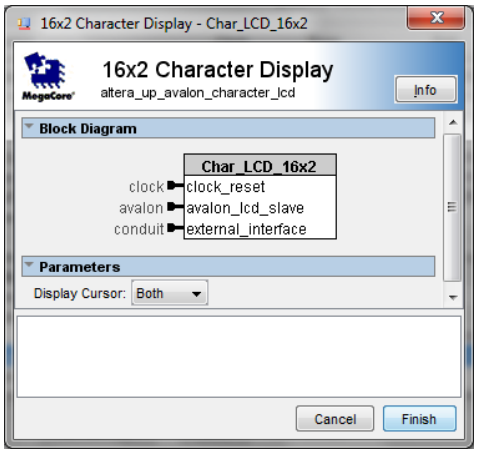


Figura 3.13 - MÓDULO LCD 16X2

3.5.6. SDCARD

Para realizar el almacenamiento de datos se usa una tarjeta SD la misma que debe ser insertada en el conector SD Card presente en la tarjeta DE2. A nivel de software se establecen los comandos que debe ejecutar el procesador NIOS II para realizar los siguientes procedimientos:

- Verificar que se encuentre una tarjeta SD insertada en el conector SD Card.
- Almacenamiento de datos.
- Verificar que la tarjeta haya sido extraída del conector SD Card.

En la figura 3.14 se muestra la configuración de la interfaz SD Card en SOPC Builder, esta configuración es a nivel de hardware.

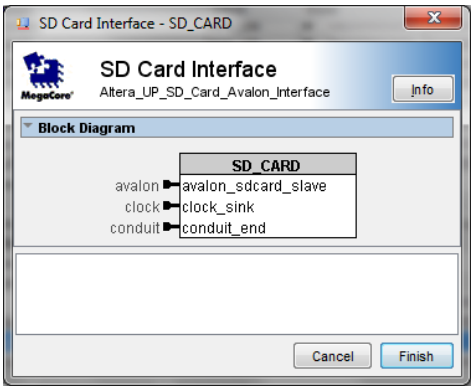


Figura 3.14 - INTERFAZ SD CARD

3.6. PROGRAMACIÓN EN LENGUAJE C USANDO MIKRO C Y ECLIPSE

En esta sección se presentan la descripción de las funciones utilizadas en la programación de ambos procesadores, estas funciones son necesarias para la implementación del código de la solución en nuestro proyecto.

Para el PIC se implementó la función para la conversión de valores analógicos a digitales la misma que es descrita a continuación

- **ADC_Read**

Esta función lee el valor analógico presente en el cualquier pin configurado como entrada analógica del microcontrolador y convierte ese valor analógico en un valor digital.

Para el funcionamiento de módulo UART se implementaron 2 funciones, una para inicializar el UART y la otra para el envío serial de los datos digitalizados las mismas que son descritas a continuación

- **UART1_Init**

Esta función permite inicializar el módulo UART para la transmisión de datos seriales con una velocidad en baudios por segundos la cual es definida por el programador

- **UART1_Write**

Esta función nos permite enviar los datos digitalizados por el pin TX del microcontrolador

El procesador NIOS II es un dispositivo programable al cual se le debe decir como interactuar con los diferentes periféricos de entrada salida que se encuentran conectados a él, el Programa Universitario de Altera proporciona una serie de librerías para cada módulo que se conecta con el procesador NIOS II, estas librerías son necesarias para el desarrollo de nuestro datalogger además se implementaron funciones adicionales las cuales se detallan a continuación.

3.6.1. PANTALLA LCD

Para la pantalla LCD se usaron las siguientes librerías:

altera_up_avalon_character_lcd.h

altera_up_avalon_character_lcd_regs.h

Las librerías mostradas anteriormente contienen la descripción de funciones necesarias para el manejo de la pantalla LCD mediante el procesador NIOS II. A continuación se muestra las funciones que se usaron en la implementación de la solución en nuestro proyecto.

- **void show_data(alt_up_character_lcd_dev * lcd,float data1, float data2)**

Esta función permite mostrar los datos (data1, data2) con su respectivo formato en la pantalla LCD de la tarjeta DE2.

- **void lcd_clear_screen(alt_up_character_lcd_dev * lcd,int len)**

Esta función permite borrar el contenido que se está visualizando en la pantalla LCD, dependiendo del valor que posea la variable **len.**

- **char * convertIntChar(float num)**

Esta función permite convertir los datos recibidos por los sensores a valores adecuados para su correcta visualización en la pantalla LCD.

3.6.2. INTERFAZ RS232

Para el módulo UART se usaron las siguientes librerías:

altera_up_avalon_rs232.h

altera_up_avalon_rs232_regs.h

Las librerías mostradas anteriormente contienen la descripción de funciones necesarias para el manejo del módulo UART mediante el procesador NIOS II. A continuación se muestra las funciones que se usaron en la implementación de la solución en nuestro proyecto.

- **float converR232ValorTemp(int valor)**

Esta función nos permite convertir de enteros a valores decimales, los datos que se reciben mediante el estándar RS232 para que sean visualizados en la pantalla LCD así como almacenarlos en una tarjeta SD.

3.6.3. SD CARD

Para el módulo SD CARD se usaron las siguientes librerías

altera_up_sd_card_avalon_interface.h

La librería mostrada anteriormente contienen la descripción de funciones necesarias para el manejo del módulo SD CARD mediante el procesador

NIOS II. A continuación se muestra las funciones que se usaron en la implementación de la solución en nuestro proyecto.

- **int converNum(int num)**

Para poder escribir en la tarjeta SD necesitamos enviarle valores hexadecimales, por tal motivo se implementó esta función, la cual recibe un valor entero y retorna el valor hexadecimal del mismo.

- **void writeDataSDCard(short int archivo,int valor)**

Esta función permite almacenar valores en un archivo .txt, el nombre del archivo es enviado como parámetro.

3.7. IMPLEMENTACIÓN FÍSICA

En esta sección se muestra la implementación física del módulo transmisor y receptor así como su respectiva simulación.

3.7.1. SIMULACIÓN DEL MÓDULO TRANSMISOR

Utilizando el programa PROTEUS el cual nos brinda un entorno amigable al momento de realizar un circuito en particular, se presenta a continuación la simulación implementada para llevar a cabo con el proceso descrito anteriormente para sensar temperatura y transmitirla hacia la tarjeta DE2.

Como primera etapa del proceso de implementación se realizó la comunicación entre dos microcontroladores PIC 16F887, el transmisor recibe las señales analógicas proporcionadas por los sensores, esos valores son digitalizados y enviados al receptor usando el protocolo de comunicación RS232. El receptor se encarga de recibir los valores enviados por el transmisor, luego pasan por un proceso de conversión para que puedan ser visualizados en la pantalla LCD.

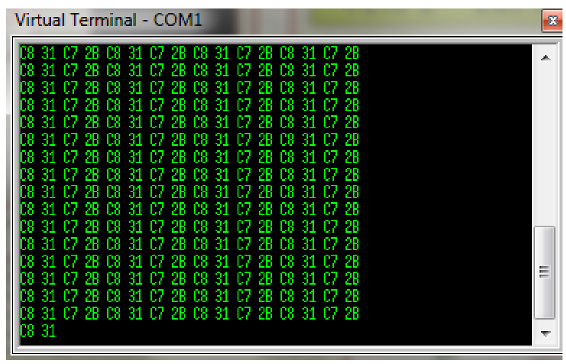


Figura 3.15 - PANTALLA DE SIMULACIÓN

La figura 3.15 corresponde a la pantalla del terminal virtual que posee PROTEUS, la misma que nos permite observar el envío de los datos digitalizados. Se observa que el formato de envío de datos se los realiza en hexadecimal, además se puede ver claramente dos datos que se repiten con frecuencia a esos datos se les llama identificadores los cuales son:

- Identificador C8 que corresponde al sensor 1
- Identificador C7 que corresponde al sensor 2

Estos valores nos sirven para reconocer de qué sensor estamos recibiendo información para luego procesarla de forma correcta y realizar los análisis respectivos.

Como siguiente paso se debe mostrar la recepción de los datos recibidos en la pantalla LCD, en la figura 3.16 se muestra el circuito implementado donde están definidas las etapas de transmisión y recepción, al momento de la simulación los valores que se estaban recibiendo eran de:

- Sensor 1 con un valor de temperatura igual a 23.9 ℃
- Sensor 2 con un valor de temperatura igual a 20.9 ℃

Los valores de temperatura que se observan en la figura 3.17 pueden variar dependiendo del estado de cada sensor y cada sensor puede sensar un valor distinto de temperatura dependiendo de la aplicación en la que se los esté usando.

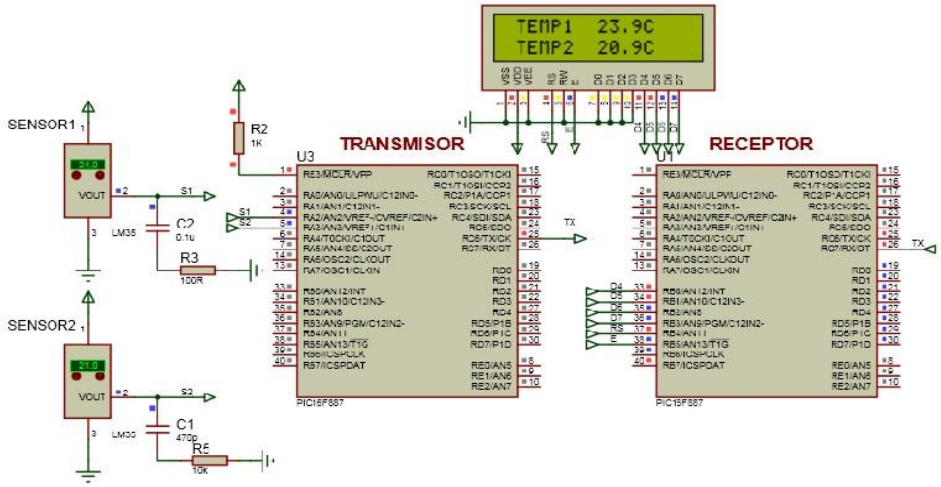


Figura 3.16 -CIRCUITO SIMULADO EN PROTEUS

3.7.2. TRANSMISIÓN DE PIC A PIC

Luego de haber realizado la simulación del sistema que se desea implementar el siguiente paso es pasar a la implementación física del circuito que se simuló. La figura 3.17 muestra la implementación en protoboard del circuito simulado en Proteus, se puede observar que los valores registrados por los sensores son visualizados en la pantalla LCD dichos valores fueron tomados en el laboratorio de sistemas digitales los cuales son:

- Sensor 1 registro un valor de 26.8 ℃
- Sensor 2 registro un valor de 26.8 ℃

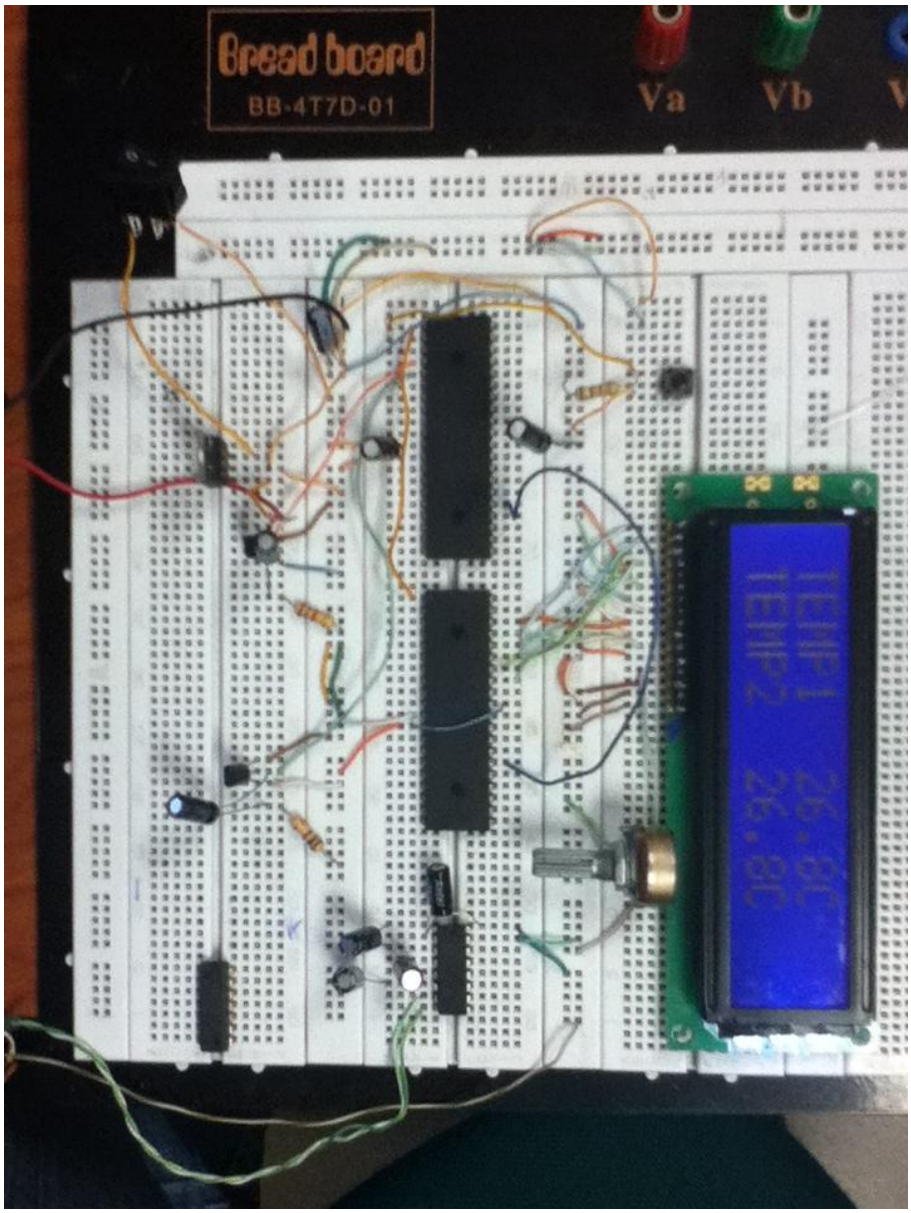


Figura 3.17 - TRANSMISIÓN DE SEÑAL

3.7.3. COMUNICACIÓN PROTOBOARD - USART MIKROC

El objetivo que se persigue es realizar la conexión entre el transmisor y la tarjeta DE2 mediante el protocolo RS232, una etapa previa de comprobación fue realizar la comunicación entre el circuito implementado en protoboard y una PC. La implementación se presenta a continuación:

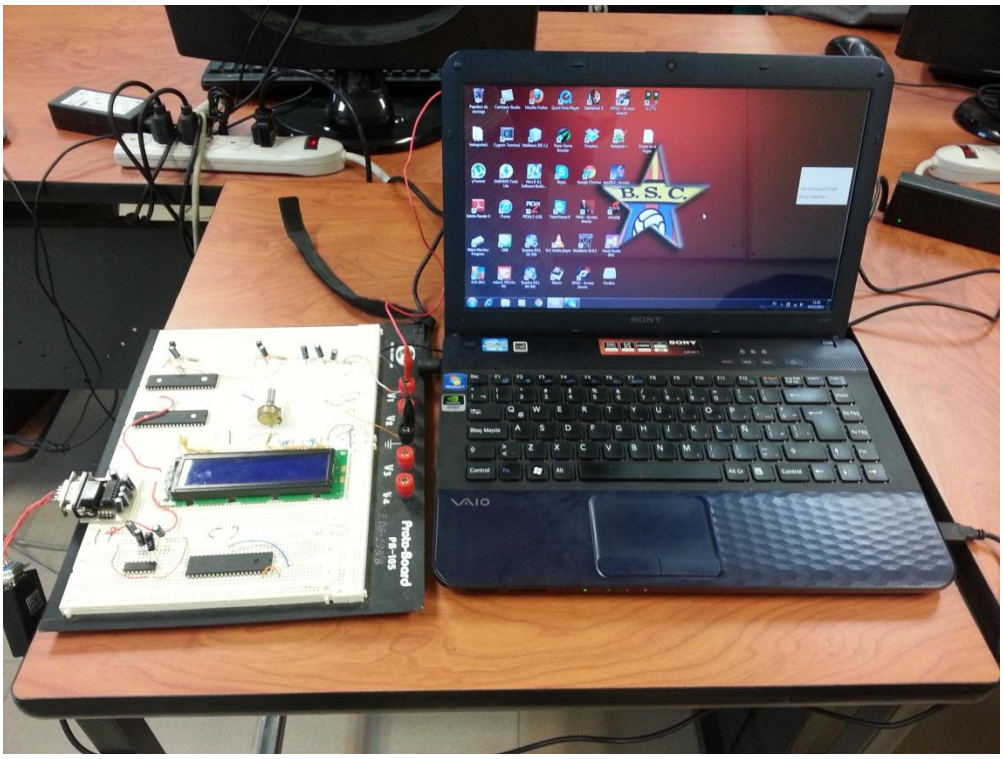


Figura 3.18 - COMUNICACIÓN ENTRE EL TRANSMISOR Y LA TARJETA DE2

En la figura 3.18 se observa la conexión entre el protoboard y la PC a través del cable serial, los datos enviados desde el protoboard son

recibidos por la PC. En la figura 3.19 se pueden observar los datos recibidos por la PC los cuales son enviados desde el protoboard.

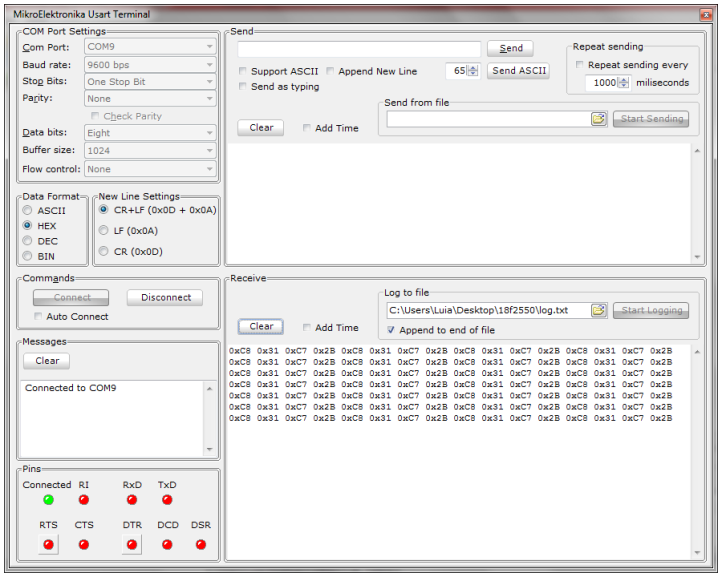


Figura 3.19 - VISUALIZACIÓN DE LOS DATOS ENVIADOS DESDE EL PROTOBOARD

La comunicación se realizó de manera exitosa lo cual nos indica que no habría problemas al momento de implementar la comunicación RS232 con la tarjeta DE2 teniendo en cuenta que la velocidad de transmisión sea la misma para ambos módulos.

3.7.4. COMUNICACIÓN INALÁMBRICA

La comunicación inalámbrica es una alternativa en este proyecto, ya que podemos implementar una aplicación en la cual los parámetros que deseamos sensar sufran una alteración al momento de realizar una comunicación inalámbrica, debido a la distancia entre el transmisor y receptor implicando una alteración en los resultados esperados. Por tal motivo podemos hacer uso de módulos inalámbricos facilitando la transmisión de los datos ya que dependiendo del módulo que usemos podemos transmitir datos a distancias de 30m, 50m, 3km garantizando que los datos recibidos no sufran alguna alteración.

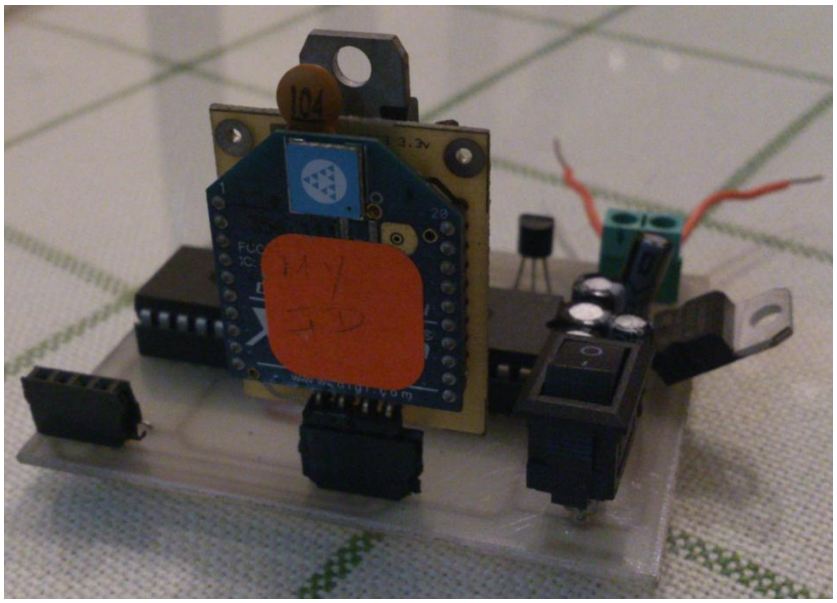


Figura 3.20 - MÓDULO DE COMUNICACIÓN INALÁMBRICA

En la figura 3.20 se muestra el circuito implementado para la transmisión inalámbrica de los valores proporcionados por los sensores de temperatura, el módulo que se observa es un XBee de un 1mW con una antena tipo chip que presenta las siguientes características:

- Alimentación de 3.3VDC
- Velocidad de transmisión máxima de 250kbps
- Rango de transmisión de 100m en línea de vista y 30m con obstáculos
- 6 entradas analógicas para conversión ADC a una resolución de 10 bits
- 8 entradas digitales
- Soporta comandos AT

3.7.5. GENERACIÓN DEL ARCHIVO SOPC INFO

El archivo SOPC INFO contiene toda la configuración sobre el sistema que se está implementando en base al procesador NIOS II, es decir contiene toda la configuración necesaria de cada módulo conectado al procesador NIOS II para luego embeber todo el sistema sobre una FPGA. Cualquier módulo que se desee agregar o quitar al sistema antes de generar el archivo SOPC INFO debe estar previamente configurado.

Para generar el archivo SOPC INFO se hace uso de la herramienta SOPC Builder que está integrada a Quartus II, con esta herramienta la generación de sistemas digitales basados en el microprocesador NIOS II es sencilla ya que solo se necesita saber que módulo se va a usar sobre el sistema, luego proceder a buscarlo en la sección de componentes disponibles para el procesador NIOS II, arrastrarlo a la venta de trabajo y configurarlo. Una vez que la configuración del sistema está completa verificando que no haya errores, se procede a la generación del archivo SOPC INFO para luego realizar la compilación del proyecto la cual se la realiza en Quartus II.

CAPÍTULO 4

4. PRUEBAS Y RESULTADOS

En el presente capítulo se muestran las pruebas realizadas que permiten mostrar el funcionamiento del proyecto. Se observa los datos de temperatura recibidos en la pantalla LCD de la tarjeta DE2, se ve que pasa cuando se configura incorrectamente los módulos Xbee y la forma de configurarlos correctamente. Además se presenta los valores de temperatura recibidos cuando algún sensor presenta problemas de conexión con el módulo transmisor. Para entender lo mencionado anteriormente se escogieron tres escenarios relevantes que permitirán al

lector entender fácilmente el funcionamiento del presente proyecto. Los escenarios seleccionados se muestran a continuación:

- Escenario A: Desconexión de un sensor de temperatura.
- Escenario B: Configuración incorrecta del módulo Xbee.
- Escenario C: Almacenamiento exitoso de los valores de temperatura.

4.1. ESCENARIO A: SOBRECALENTAMIENTO DE UN SENSOR DE TEMPERATURA

Para verificar el funcionamiento de los sensores de temperatura se realizó una conexión entre el módulo transmisor y la PC con lo cual se puede observar los datos transmitidos, los mismo que deben estar e formato hexadecimal. Para la visualización de los datos se usó el terminal USART del software Mikro C; En la figura 4.1 se observa los valores transmitidos a la PC, los cuales están fuera del rango establecido. Cabe señalar que el rango establecido fue sensar temperaturas mayores a 0 grados.

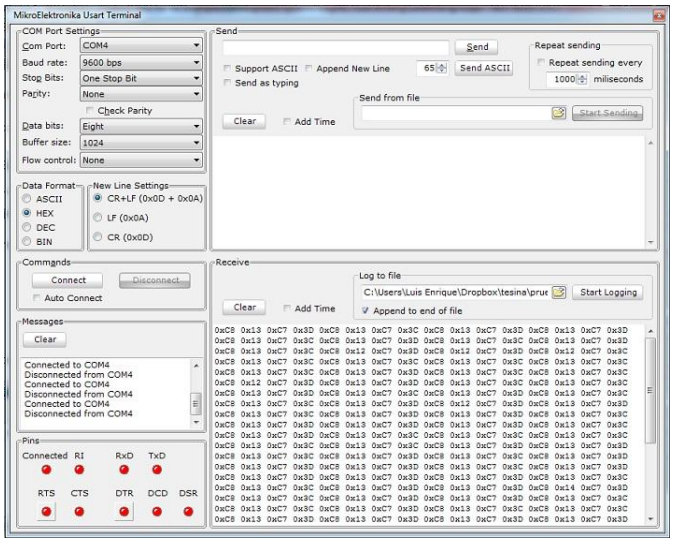


Figura 4.1 - REGISTRO DE LOS VALORES DE TEMPERATURA DE CADA SENSOR FUERA DEL RANGO ESTABLECIDO

En la tabla 4.1 se muestran dos aplicaciones en las cuales tendríamos problemas con uno o ambos sensores de temperatura si se sobrecalientan.

Tipo de Prueba	Error que puede presentar el sensor
1) Monitorear Temperaturas mayores a 0 grados	Recibir valores de temperaturas menores a 0 grados debido a que se le aplica al sensor un voltaje de alimentación mayor al requerido.
2) Monitorear Temperaturas Positivas y Negativas	Visualización de temperaturas fuera del rango esperando debido a una falla en su configuración interna, básicamente en su diodo zener.

Tabla 4.1 - APLICACIONES QUE PUEDEN PRESENTAR ALGÚN ERROR AL MOMENTO DE LA IMPLEMENTACIÓN

Para nuestro caso la aplicación que se implementó fue sensar temperaturas mayores a 0 grados, los resultados obtenidos se pueden observar en la figura 4.2

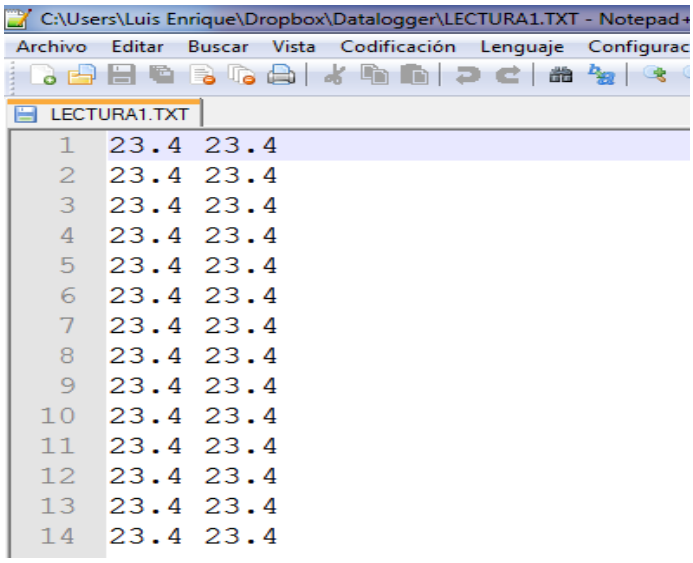


Figura 4.2 - VALORES DE TEMPERATURAS ALMACENADOS EN UN ARCHIVO DE TEXTO

Al sobrecalentarse un sensor de temperatura se pueden presentar los errores descritos en la tabla 4.1, es recomendable tomar las precauciones adecuadas para no quemar los sensores, de lo contrario tendremos problemas al momento de estar monitoreando temperaturas.

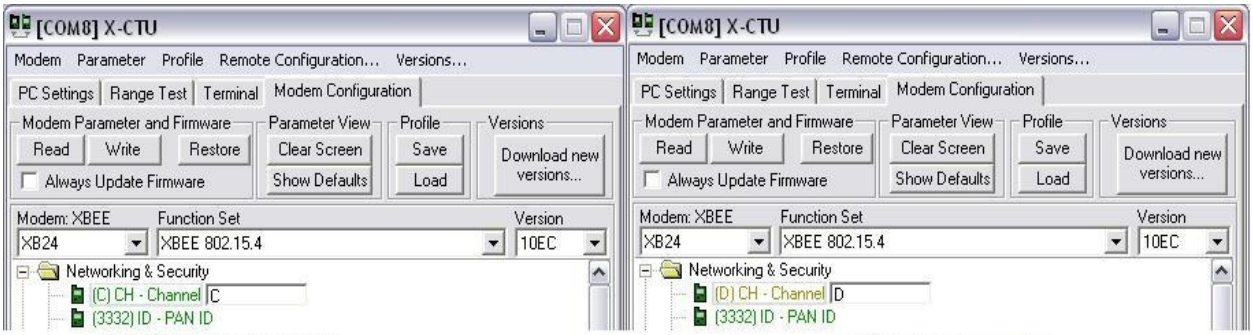
4.2. ESCENARIO B: CONFIGURACIÓN INCORRECTA DEL MÓDULO XBEE

Para la transmisión inalámbrica de los valores de temperatura se hace uso de los módulos Xbee, los mismos que deben ser previamente configurados.

Para poder establecer una comunicación ambos módulos deben tener la siguiente configuración:

- Deben funcionar en el mismo Canal.
- Deben transmitir a la misma velocidad de baudios.
- Deben tener la misma configuración de los bits de paridad, stop y datos.

En este escenario no será posible realizar una comunicación entre ambos módulos ya que no se encuentran funcionando en el mismo canal, la configuración realizada para el módulo transmisor y para el receptor se puede observar en la figura 4.3.



a) Módulo Transmisor b) Módulo Receptor

Figura 4.3 - CONFIGURACIÓN INCORRECTA DEL CANAL DE COMUNICACIÓN DEL MÓDULO XBEE. A) MÓDULO TRANSMISOR EN EL CANAL C. B) MÓDULO RECEPTOR EN EL CANAL D

Debido a esta configuración incorrecta no es posible que la tarjeta DE2 reciba algún dato de temperatura. En la tabla 4.2 se muestra la configuración detallada de cada módulo Xbee, ésta configuración impide la comunicación entre el módulo transmisor y el receptor.

PARAMETRO	MODULO TRANSMISOR	MODULO RECEPTOR
CANAL	C	D
VELOCIDAD DE BAUDIOS	9600	14400
NUMERO DE BITS	8	7
PARIDAD	1	2
BITS DE PARADA	2	1

Tabla 4.2 - PARÁMETRO DE CONFIGURACIÓN PARA LOS MÓDULOS XBEE

Para establecer la comunicación entre ambos sensores los parámetros de la tabla 4.2 deben ser los mismos para ambos módulos.

4.3. ESCENARIO C: ALMACENAMIENTO EXITOSO DE LOS VALORES DE TEMPERATURA

En este escenario se presenta el correcto funcionamiento del proyecto, se observa todo el proceso de envío, recepción, almacenamiento y análisis de los valores de temperatura registrados en la tarjeta SD.

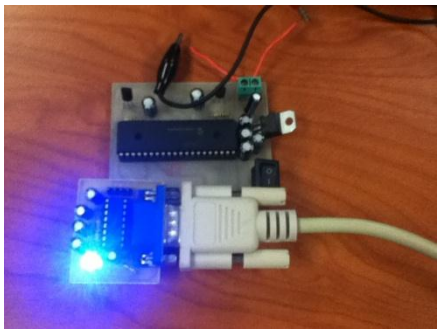


Figura 4.4 - MÓDULO TRANSMISOR

En la figura 4.4 se muestra el módulo transmisor utilizado para la transmisión de los datos de temperatura, se observa también la conexión con la tarjeta DE2 a través del cable serial. El led indica que se están transmitiendo los datos de temperatura digitalizados por el PIC al driver MAX232.

Los datos transmitidos son recibidos en la tarjeta DE2 y mostrados en la pantalla LCD. Si queremos almacenar los datos recibidos debemos de

ingresar una tarjeta SD en el puerto SDCard tal como se ilustra en la figura 4.5.

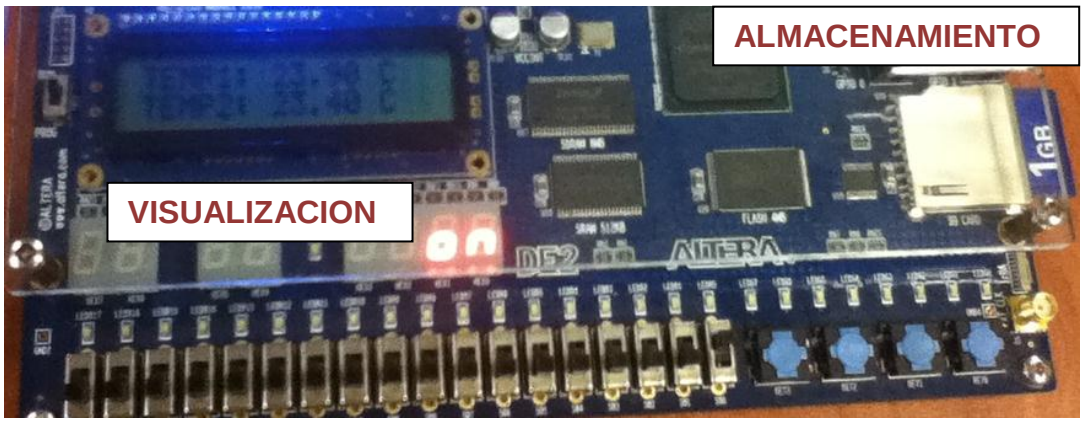


Figura 4.5 - VISUALIZACIÓN Y ALMACENAMIENTO DE TEMPERATURA

Para comenzar con el proceso de almacenamiento de la temperatura proporcionada por los sensores es necesario además de insertar una tarjeta SD accionar el switch 0. Al accionar el switch 0 se muestra la palabra ON en los displays de 7 segmentos, esto nos indica que se comienza a llevar un registro de la temperatura proporcionada por cada sensor. Cada vez que se accione el switch 0 se crea un archivo diferente en la tarjeta SD esto nos permite tener un registro independiente de la temperatura para luego realizar un análisis con dichos datos.

Todo el proceso mencionado anteriormente nos garantiza el almacenamiento de los datos de temperatura proporcionados por cada

sensor, el siguiente paso es realizar un análisis gráfico de cada lectura registrada, el mismo que es mostrado a continuación:

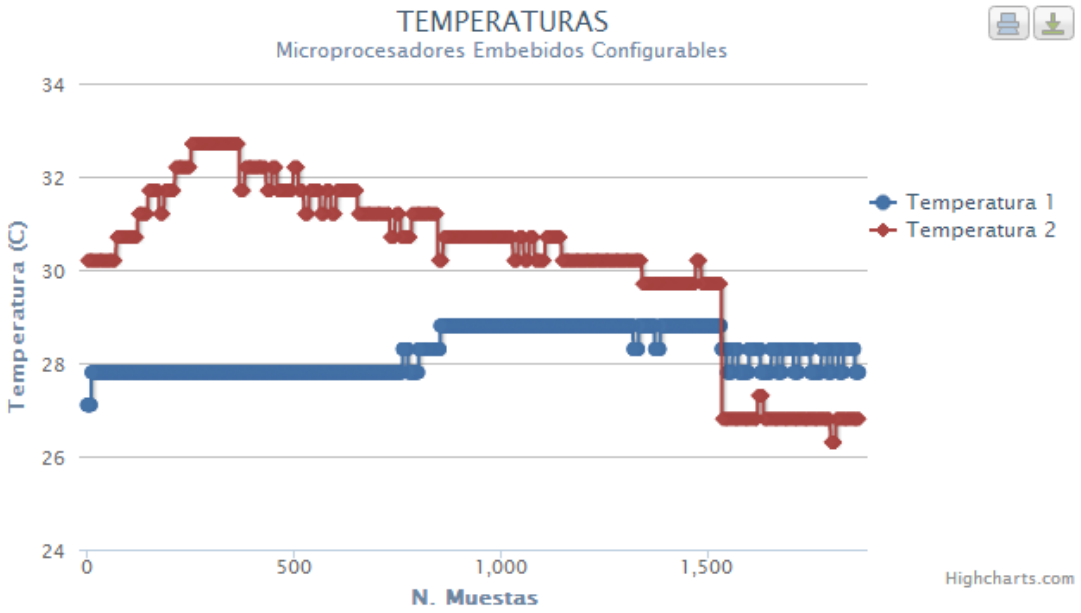


Figura 4.6 - GRAFICA TEMPERATURA VS NÚMERO DE MUESTRAS

En la figura 4.6 se observa la gráfica de los datos de temperatura registrados en la tarjeta SD, se observa claramente que la temperatura inicial del sensor1 representado por la curva de color azul es de 27.1 °C y del sensor2 representado por la curva color rojo es de 31.4 °C**.** A medida que se van registrado más muestras y el tiempo transcurre los datos van cambiando, en la muestra 250 se observa un cambio en la temperatura del sensor. Dependiendo de la aplicación que se esté

implementado la gráfica va a cambiar debido a los datos de temperatura que se registren.

CONCLUSIONES Y

RECOMENDACIONES

CONCLUSIONES

1. En base los objetivos planteados se pudo generar un sistema para una tarjeta de desarrollo NIOS II, y a su vez compilar un software en lenguaje C que permita cumplir con los requisitos básicos de un Datalogger.
2. El uso del lenguaje C como herramienta para desarrollar una serie de instrucciones que sirven para controlar el funcionamiento del proyecto, demuestra la versatilidad que posee este lenguaje de alto nivel para obtener una implementación en hardware acorde a los requerimientos establecidos del proyecto.
3. Se implementó un hardware básico usando la tarjeta DE 2 de Altera que puede ser usado como guía para realizar futuras mejoras como; enviar los datos a través de la red, controlar el sistema de manera remota, entre otras funcionalidades, por lo tanto; se puede concluir que se consiguió un sistema versátil y eficiente con posibilidades de ampliar su capacidad y funcionalidad en diseños futuros.

RECOMENDACIONES

1. Se recomienda revisar los diagramas esquemáticos de la tarjeta DE2 cuando se vaya a conectar algún dispositivo externo para evitar corto circuitos, problemas de conexión, daños en los módulos integrados a la tarjeta y sobre todo para evitar algún daño permanente que deje inutilizable por completo a la tarjeta DE2.
2. Se recomienda acudir a los foros de altera cuando se presente dudas al momento de realizar la configuración de cualquier módulo de la tarjeta DE2 en Quartus II o en NIOS II IDE.
3. Se recomienda tener precaución al codificar las líneas de control en lenguaje C, debido a que el uso de los puertos que posee la Tarjeta de Altera se realiza mediante direcciones de memoria, por lo que se podría presentar un mal funcionamiento al momento de ejecutar el código.
4. Se recomienda el uso del depurador de NIOS II IDE para observar la ejecución del código, debido a que se pueden presentar fallas en el direccionamiento de memoria que se esté usando.
5. Verificar la hoja técnica de cualquier dispositivo externo a usar en el proyecto, como es el caso del módulo inalámbrico XBEE, microprocesador PIC 16f887 entre otros elementos usados en la

implementación del Datalogger para evitar daños por algún voltaje mal establecido.

6. Se recomienda usar software de ayuda para poder visualizar la comunicación entre el módulo y la tarjeta a la hora de realizar el proyecto, de esta forma observaríamos errores de programación o electrónicos y así podríamos evitar gastos innecesarios.

ANEXOS

ANEXO A

CODIGO FUENTE EN LENGUAJE C

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <io.h>
#include "system.h"
#include "altera_up_avalon_character_lcd.h"
#include "altera_up_avalon_character_lcd_regs.h"
#include "altera_up_avalon_rs232.h"
#include "altera_up_avalon_rs232_regs.h"
#include "altera_up_sd_card_avalon_interface.h"
#include "altera_up_avalon_parallel_port.h"
#include "altera_up_avalon_parallel_port_regs.h"

/*Definiciones*/
#define  intToChar(entero) entero+0x30

/*Funciones Implementadas*/
void lcd_clear_screen(alt_up_character_lcd_dev * lcd,int len);
void show_data(alt_up_character_lcd_dev * lcd,float data1, float data2);
char * convertIntChar(float num);
float converR232ValorTemp(int valor);
int converNum(int num);
void writeDataSDCard(short int archivo,int valor);
```

FUNCIONES Y PROCEDIMIENTOS

```
/*
	lcd_clear_screen
	parámetros:   alt_up_character_lcd_dev * lcd,int len
	Este procedimiento limpia la pantalla de LCD
*/
void lcd_clear_screen(alt_up_character_lcd_dev * lcd,int len){
	alt_up_character_lcd_set_cursor_pos(lcd, 0, 0);
	alt_up_character_lcd_write(lcd, "                ", len);
	alt_up_character_lcd_set_cursor_pos (lcd, 0, 1);
	alt_up_character_lcd_write(lcd, "                ", len);
}
```

```
/*
	show_data
	parametros:   alt_up_character_lcd_dev * lcd,float data1, float data2
	Este procedimiento ayuda a mostrar los datos en la pantalla de la LCD
*/

void show_data(alt_up_character_lcd_dev * lcd,float data1, float data2){
	char temp1[16]="TEMP1: ";
	char temp2[16]="TEMP2: ";
	char valor1,valor2;
	char tm[2]="";

	strcat(temp1,convertIntChar(data1));
	strcat(temp2,convertIntChar(data2));
	strcat(temp1," C");
	strcat(temp2," C");
	lcd_clear_screen(lcd,16);
	//Coloca los datos en la pantalla LCD
	alt_up_character_lcd_set_cursor_pos(lcd, 0, 0);
	alt_up_character_lcd_write(lcd,temp1,strlen(temp1));
	alt_up_character_lcd_set_cursor_pos (lcd, 0, 1);
	alt_up_character_lcd_write(lcd,temp2,strlen(temp2));
	alt_up_character_lcd_cursor_off(lcd);

}



/*
	convertIntChar
	parámetro:    float num
	Esta función convierte un número a char, esta conversión se la realiza
	para poder colocar el valor en la pantalla LCD
*/

char * convertIntChar(float num)
{
	char numConv[5]="",temporal;
	int i=0,longitud,j;
	float car;
	char valCon;
	sprintf(numConv, "%.2f", num);
	return numConv;
}
```

```
/*
	converR232ValorTemp
	parámetro;	int valor
	Proceso que se debe realizar al momento que se lee un
	dato del RS232, para poder presentar en la LCD un valor
	de temperatura
*/

float converR232ValorTemp(int valor){
	int temp = valor;
	float ultimo;
	float valorT=0;
	temp = temp * 5000;
	temp = temp / 1024;
	ultimo = temp % 10;
	temp = temp / 10;
	ultimo = ultimo * 0.1;
	valorT = temp + ultimo;
	return valorT;
}



/*
	writeDataSDCard
	parámetros:	short int archivo,int valor
	El valor recibido como parámetro es almacenado en la
	SDK en un archivo de texto
*/

void writeDataSDCard(short int archivo,int valor){
	int temp = valor;
	int ulti,penul,prime;
	int valorT=0;
	temp = temp * 5000;
	temp = temp / 1024;//Valor
	ulti = temp % 10;
	temp = temp / 10;
	penul = temp % 10;
	temp = temp / 10;
	prime = temp % 10;
	alt_up_sd_card_write(archivo,converNum(prime));
	alt_up_sd_card_write(archivo,converNum(penul));
	alt_up_sd_card_write(archivo,0x2E);
	alt_up_sd_card_write(archivo,converNum(ulti));
	alt_up_sd_card_write(archivo, 0x20);

}
```

```
/*
	converNum
	parámetros:   int num
	Esta función convierte un numero decimal a Hexadecimal
	esto nos ayuda al momento de almacenar los datos en la SDK
*/
int converNum(int num){
	switch(num){
	case 0:
			return 0x30;
		break;
	case 1:
			return 0x31;
		break;
	case 2:
			return 0x32;
		break;
	case 3:
			return 0x33;
		break;
	case 4:
			return 0x34;
		break;
	case 5:
			return 0x35;
		break;
	case 6:
			return 0x36;
		break;
	case 7:
			return 0x37;
		break;
	case 8:
			return 0x38;
		break;
	case 9:
			return 0x39;
		break;
	default:
		return 0;
	}
}
```

PROGRAMA PRINCIPAL

```
int main(){
       //Variables Locales
   short int archivo;
   alt_up_sd_card_dev * sdk = NULL;
   int conec=0;
   alt_up_character_lcd_dev * lcd45;
   alt_up_rs232_dev * uart;
   int len=15;
   float data1=0,data2=0;
   int datatemp1=0,datatemp2=0,cmpcad;
   int i,sw,loop,recivido,encArc;
   int *check=0x00,contsegundos=0;
   int anterior=0,flagAlmacenar=0,contp=0,flagCerrar=0,contarchi=0;
   char
ascii,archivocadtemp[50]="LECTURA",valorcad[10]="0",archivocad[50]="LECTURA",cadEn
[50]="";
   volatile int *rx=SERIAL_PORT_BASE + 0;
   volatile int *psw = SLIDER_SWITCHES_BASE;
   volatile int *pHEX7SEG = HEX3_HEX0_BASE;
   int switchp = *psw;
   *pHEX7SEG = 0x003F3F71;

   /*Enlazamos los punteros a los dispositivos*/
       lcd45 = alt_up_character_lcd_open_dev(CHAR_LCD_16X2_NAME);
       IORD_ALT_UP_RS232_DATA(SERIAL_PORT_BASE);
       uart = alt_up_rs232_open_dev(SERIAL_PORT_NAME);
    sdk = alt_up_sd_card_open_dev(SD_CARD_NAME);

    /*Mensaje de Bienvenida en la LCD*/
    alt_up_character_lcd_set_cursor_pos(lcd45, 0, 0);
    alt_up_character_lcd_write(lcd45, "WELCOME NIOS II", len);
    alt_up_character_lcd_set_cursor_pos (lcd45, 0, 1);
    alt_up_character_lcd_write(lcd45, "...  ESPOL  ...", len);
    usleep(400000);
    lcd_clear_screen(lcd45,16);
    usleep(2000);
    if(sdk!= NULL){
      while(1){
         usleep(2000);
         recivido=alt_up_rs232_read_data(uart,rx,0);
         usleep(2000);
         if(anterior == 0xC8 && anterior != *rx){
              data1 = converR232ValorTemp(*rx);
              datatemp1=*rx;
              show_data(lcd45,data1,data2);
         }
         if(anterior == 0xC7 && anterior != *rx){
```

```
                data2 = converR232ValorTemp(*rx);
                datatemp2=*rx;
                show_data(lcd45,data1,data2);
            }
            anterior = *rx;
            usleep(200000);
            sw = *psw & 0x01;
            if( sw == 1){
                   *pHEX7SEG = 0x00005C54;
                   flagAlmacenar=1;
                   contp++;
                          if(alt_up_sd_card_is_Present()== true){
                               printf("Card Conec");
                               if (alt_up_sd_card_is_FAT16()) {
                                  printf("FAT16 file system detected.\n");
                                  if(flagAlmacenar==1 && contp==1){
                                        strcpy(archivocad,archivocadtemp);
                                        strcat(archivocad,valorcad);
                                        strcat(archivocad,".TXT");
                                        encArc=alt_up_sd_card_find_first(".",cadEn);
                                        do{
                                         contarchi++;
                                         cmpcad= strcmp(cadEn,archivocad);
                                         if(cmpcad==0){
                                                sprintf(valorcad,"%d",contarchi);
                                                strcpy(archivocad,archivocadtemp);
                                                strcat(archivocad,valorcad);
                                                strcat(archivocad,".TXT");

                                         }
                                         encArc = alt_up_sd_card_find_next(cadEn);
                                        }while(encArc == 0 && cmpcad!=0);
                                        archivo=alt_up_sd_card_fopen(archivocad,true);
                                  }
                                  //Escribo en SDK
                                  if(contsegundos==22){
                                         writeDataSDCard(archivo,datatemp1);
                                         writeDataSDCard(archivo,datatemp2);
                                         alt_up_sd_card_write(archivo,0x0D);
                                         contsegundos=0;
                                  }else
                                         contsegundos++;
                               } else {
                                  printf("Unknown file system.\n");
                               }
                                  conec = 1;
                          }else if ((conec == 1) && (alt_up_sd_card_is_Present() ==
false)) {
                                    printf("Card disconnected.\n");
```

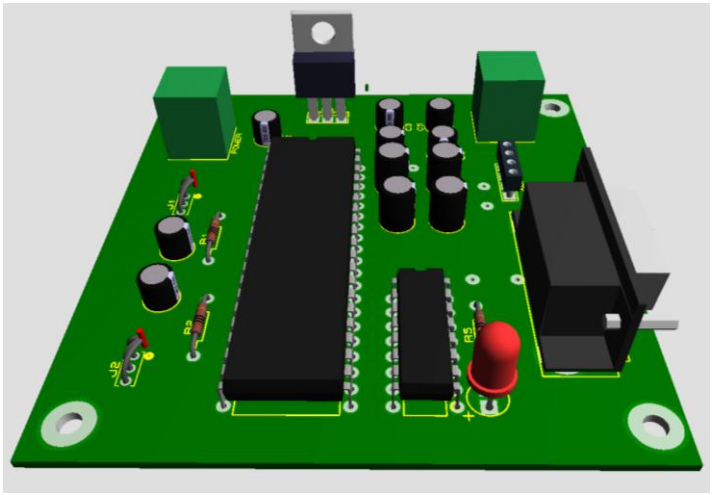
```
                                conec = 0;
                                flagAlmacenar=0;
                                contp=0;
        contarchi=0;
                        }
                         flagCerrar=1;
          }else{
                 if(flagCerrar==1)
                        alt_up_sd_card_fclose(archivo);
                 if ((conec == 1) && (alt_up_sd_card_is_Present() == false)) {
                                printf("Card disconnected.\n");
                                conec = 0;
                 }
                 flagAlmacenar=0;
                 contp=0;
                 flagCerrar=0;
                 contarchi=0;
                 *pHEX7SEG = 0x003F3F71;
          }
        }

    }
    return 0;
}
```

ANEXO B

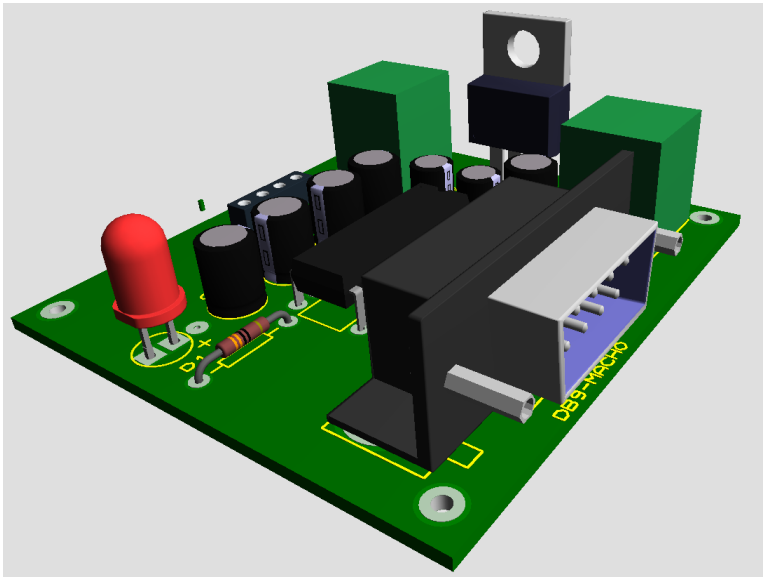
CODIGO FUENTE DEL MICROCONTROLADOR

```
unsigned int adc_rd;

void Move_Delay() {
  Delay_ms(100);
}
void main()
{
 OSCCON=0X75;
INTCON = 0;
ANSEL =   0x0C; // PIN4 Y PIN5 Como entradas analógicas
ANSELH =  0;
ADCON1 = 0x82; // Configuramos el Modulo ADC
TRISA  = 0xFF; // Pines del Puerto A como entradas

UART1_Init(9600);  // Inicializamos el módulo UART a 9600 bps

Delay_ms(1000);

while(1)
        {
           adc_rd  = ADC_read(2); // Conversión de valor analógico del PIN4
           Delay_ms(300);
           UART1_Write(0xC8); // Enviamos el Indicador del Sensor 1
           Delay_ms(300);
           UART1_Write(adc_rd);   // Enviamos el valor digitalizado del PIN4
           Delay_ms(1000);
           adc_rd  = ADC_read(3); // Conversión de valor analógico del PIN5
           Delay_ms(300);
           UART1_Write(0xC7); // Enviamos el Indicador del Sensor 2
           Delay_ms(300);
           UART1_Write(adc_rd); // Enviamos el valor digitalizado del PIN5
           Delay_ms(1000);
        }
}
```

ANEXO C



VISTA EN 3D DEL MODULO TRANSMISOR



VISTA EN 3D DEL MODULO RECEPTOR

ANEXO D

HTML, PHP Y JAVASCRIPT

Las tecnologías mencionadas se las utilizó para diseñar una pequeña aplicación que muestre de manera gráfica los datos obtenidos por el Datalogger.

Esta aplicación web usa HTML para la estructura de las páginas, PHP como servidor que procesa la solicitud del usuario al momento de enviar el archivo que contiene los datos y JAVASCRIPT para generar de manera automática el gráfico.

PÁGINA DE INICIO DE LA APLICACIÓN

BIBLIOGRAFIA

[1]. TARJETA DE DESARROLLO DE2 DE ALTERA

https://paruro.pe/productos/tarjeta-altera-de2

Fecha de Consulta: Junio 2012

[2]. MÓDULO LCD 16X2

http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=78&No=396

Fecha de Consulta: Junio 2012

[3]. DIAGRAMA DE CONEXIÓN DEL MÓDULO LCD16X2

Altera Corporation Embedded Peripherals IP User Guide - Optrex 16207 LCD Controller Core – Pág. 98

Fecha de Consulta: Junio 2012

[4]. MODULO UART

Altera Corporation Embedded Peripherals IP User Guide- UART Core - Pág. 69

Fecha de Consulta: Junio 2012

[5]. MÓDULO SDCARD

Altera Corporation Altera University Program Secure Data Card IP Core - Pág 2

Fecha de Consulta: Junio 2012

[6]. ESTRUCTURA DE UNA FPGA

http://beta.globalspec.com/reference/81250/203279/chapter-29-field-programmable-gate-arrays-fpgas

Fecha de Consulta: Junio 2012

[7]. FPGA CYCLONE II

http://www.ictradenet.com/Vendor/Altera/Index13.htm

Fecha de Consulta: Junio 2012

[8]. SISTEMA BASADO EN EL MICROPROCESADOR NIOS II

Estudio del Microprocesador NIOS II Jorge Rodríguez Araújo 14 de Marzo del 2010- Pág. 2

Fecha de Consulta: Junio 2012

[9]. ESTRUCTURA DEL MICROPROCESADOR NIOS II

Alera Corporation Nios II Processor Reference Handbook- Pág. 18

Fecha de Consulta: Julio 2012

[10]. CONEXIÓN ENTRE PERIFÉRICOS

Estudio del Microprocesador NIOS II Jorge Rodríguez Araújo 14 de Marzo del 2010 – Pág. 9

Fecha de Consulta: Julio 2012

[11]. INTERCONEXIÓN DE PERIFÉRICOS CON LA RED AVALON

Estudio del Microprocesador NIOS II Jorge Rodríguez Araújo 14 de Marzo del 2010 - Pág. 10

Fecha de Consulta: Julio 2012

[12]. DIAGRAMA DE BLOQUES PIC 16F887

Milan Verle PIC Microcontrollers - Programming in C microelectronic; 1st edition (2009) – Pág. 30

Fecha de Consulta: Julio 2012

[13]. ARQUITECTURA PIC 16F887

Milan Verle PIC Microcontrollers - Programming in C mikroElektronika; 1st edition (2009) - Pág. 24

Fecha de Consulta: Julio 2012

[14]. EMPAQUETADO TIPO DIP PIC 16F887

Milan Verle PIC Microcontrollers - Programming in C mikroElektronika; 1st edition (2009) - Pág. 29

Fecha de Consulta: Julio 2012

[15]. MODULO ADC PIC 16F887

Milan Verle PIC Microcontrollers - Programming in C mikroElektronika; 1st edition (2009) - Pág. 125

Fecha de Consulta: Julio 2012

[16]. REGISTROS DE ALMACENAMIENTO DE LOS DATOS CONVERTIDOS

Milan Verle PIC Microcontrollers - Programming in C mikroElektronika; 1st edition (2009) - Pág. 125

Fecha de Consulta: Julio 2012

[17]. COMUNICACIÓN ENTRE DISPOSITIVOS DIFERENTES

Milan Verle PIC Microcontrollers - Programming in C mikroElektronika; 1st edition (2009) - Pág. 96

Fecha de Consulta: Julio 2012

[18]. MÉTODO DE TRANSMISIÓN SERIAL

Milan Verle PIC Microcontrollers - Programming in C mikroElektronika; 1st edition (2009) - Pág 97

Fecha de Consulta: Julio 2012

[19]. ESQUEMÁTICO DE CONEXIÓN RS232

Milan Verle PIC Microcontrollers - Programming in C mikroElektronika; 1st edition (2009) - Pág. 262

Fecha de Consulta: Julio 2012

[20]. COORDINADOR PAN CON MÚLTIPLES NODOS

Andrés Oyarce Xbee Series 1 Guía del Usuario – Pág. 9

Fecha de Consulta: Julio 2012

[21]. ELEMENTOS DEL XBEE

Andrés Oyarce Xbee Series 1 Guía del Usuario - Pág. 10

Fecha de Consulta: Octubre 2012

[22]. RED MESH IMPLEMENTADA CON MÓDULOS XBEE

Andrés Oyarce Xbee Series 1 Guía del Usuario - Pág. 11

Fecha de Consulta: Octubre 2012

[23]. Estudio del microprocesador NIOS II, Jorge Rodríguez Araujo 2010 pág. 2

Fecha de Consulta: Julio 2012

[24]. Estudio del microprocesador NIOS II, Jorge Rodríguez Araujo 2010 pág. 3

Fecha de Consulta: Julio 2012

[25]. Estudio del microprocesador NIOS II, Jorge Rodríguez Araujo 2010 pág. 7

Fecha de Consulta: Julio 2012