SISTEMA DE DEMOSTRACIÓN PARA SISTEMAS OPERATIVOS DE TIEMPO REAL SOBRE SISTEMAS EMPOTRADOS, CASO: UP ADAPTABLE NIOS II

César Iván Alvear Vega
Ricardo Antonio Palacios Ledezma

Trabajo de Grado en Modalidad de Desarrollo

Monografía presentada para optar al título de Ingeniero en Electrónica y Telecomunicaciones

Directora
Mag. Ing. Carolina Ríos Fuentes

Co-Directora
Mag. (c) Ing. Mary Cristina Carrascal

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Popayán, Octubre de 2010

Tabla de Contenido

Índice de Ilustraciones

Índice de Tablas

Capítulo I Introducción

1.1 Contexto

Cada vez cobra mayor importancia la interacción de la tecnología con la manera en que la sociedad se desarrolla, es así como en la actualidad es más fácil percibir esta correlación determinando de qué manera las personas se relacionan, se comunican, buscan entretenimiento, se informan, se educan; muchas veces estas acciones se efectúan de manera simultánea y siempre se espera que todo ocurra con total inmediatez, en el instante en el que se desee o necesite [1].

Toda esta serie de interacciones o requerimientos se llevan a cabo sobre una gran cantidad de medios y herramientas que acercan al hombre con el resto del mundo. Estas relaciones hacen necesario que se busquen formas de administrar de una manera eficiente tanto el tiempo en el que se llevan a efecto, como la versatilidad en el manejo de los recursos tecnológicos con que se efectúan estos procesos.

Gran cantidad de aplicaciones orientadas a resolver estas labores se encuentran diseñadas sobre sistemas con capacidad para llevar a cabo tareas de forma autónoma, programada, con límites en los tiempos de ejecución [2], denominados Sistemas en Tiempo Real (STR); los cuales permiten dar solución a un extenso número de necesidades humanas, tan diversas como en un electrodoméstico en el hogar o el control sistemas aéreos, entre otros, como se muestra en la **¡Error! No se encuentra el origen de la referencia.** a continuación [3]:

Tabla 1 Aplicaciones de los Sistemas de Tiempo Real

Campo de aplicación	Aplicación
Medicina	Robot cirujano, Proyección de imágenes medicas, Control de cirugía remota
Industria	Controlador de plantas de producción, Robot para tratamiento de material peligroso, Líneas de montaje robótico, Inspección automatizada
Construcción civil	Control de elevadores, Sistemas de automoción
Multimedia	Juegos, simuladores
Usos militares	Sistema de reconocimiento de blancos automático, Sistema de guiado de misiles y navegación
Sistemas altamente críticos	Plantas nucleares, Sistemas de aviónica, Sistemas de navegación

Se puede observar que en el ámbito local aun no se encuentra un desarrollo industrial destacado que produzca las tecnologías encargadas de suplir estas necesidades, en la mayoría de los casos, éstas son producidas en el exterior e importadas; debido a esto, se

requiere adaptaciones a las necesidades del entorno nacional, las cuales en algunos casos resultan inadecuadas. Lo anterior conlleva a la necesidad de cultivar el conocimiento que permita entender los conceptos que originan y que permiten desarrollar tecnologías relacionadas con los STR.

Uno de los más importantes campos de aplicación aportado por los STR, es el trabajo con sistemas empotrados, estos instrumentos son un motor de desarrollo tecnológico ya que son sistemas que trabajan con autonomía permitiendo el manejo confiable de múltiples procesos dedicados, son sistemas de bajo costo ya que pueden ser fabricados en serie, y en general cuentan con una arquitectura simple, haciendo que su estudio y maneo no sea complejo por lo que son herramientas apropiadas para formar parte de un curso teórico practico sobre STR.

De lo expuesto anteriormente, se puede establecer que existe un extenso número de posibilidades de creación de soluciones tecnológicas con las que se puede mejorar la calidad de vida pensando en las necesidades explicitas presentes en el país; en consecuencia, el estudio académico para abordar el desarrollo de STR sobre sistemas empotrados se convierte en una gran oportunidad para ampliar las experiencias y resaltar los conocimientos que busquen el desarrollo de éstas aplicaciones, de tal modo que el fruto de su estudio y experimentación desemboca en un aprovechamiento del potencial humano de la Facultad de Ingeniería Electrónica y Telecomunicaciones FIET en el desarrollo de la industria tecnológica del país.

1.2 Escenarios de motivación

En la actualidad es frecuente encontrar cada vez más aplicaciones en las que el tiempo ocupa un rol fundamental. Entre estas aplicaciones podemos citar tareas tan cotidianas como el uso de electrodomésticos en el hogar [4]; y en contextos determinantes como protocolos de comunicación, controladores de comandos de aviones, de robots, de procesos industriales automatizados, de dispositivos electrónicos; aplicaciones multimedia y de internet; entre otras. Muchas de estas, conllevan a manejar situaciones de una importancia crítica, en donde cualquier mal funcionamiento podría conducir a consecuencias tan graves como poner en riesgo grandes inversiones económicas e incluso vidas humanas.

Por otra parte, la FIET cuenta con algunos dispositivos de desarrollo y experimentación, en particular el Nios II Development Kit, Stratix II Edition [5], sobre el cual es posible desarrollar prácticas académicas tendientes a recrear el comportamiento de un STR; no obstante, solo se cuenta con uno en el laboratorio por lo cual es necesario contar con una herramienta que permita su administración para masificar su uso. Es así como se cuenta con la oportunidad de explotar sus características en pro de buscar el aprovechamiento constructivo del recurso disponible y así poder aportar al proceso de crecimiento académico de la facultad.

Es aquí donde ofrecer posibilidades de formación académica sobre los principios y el funcionamiento de éstos dispositivos; como también, poder contar con una herramienta adecuada de experimentación, que permita la multiplexación del acceso al recurso de manera remota y poder ser utilizado por un numeroso grupo de alumnos; permitirá que los estudiantes de la FIET puedan fortalecer su conocimiento e impulsar su creatividad; lo que abre nuevas expectativas con el propósito de buscar soluciones a los retos que

presenta hacia el futuro el desarrollo de la sociedad y su estrecho vínculo con la tecnología.

1.3 Definición del problema

En los últimos años, el desarrollo de los STR ha tenido un continuo crecimiento, prueba de esto es la gran cantidad de aplicaciones en diversos campos de la tecnología que dan solución de forma efectiva a un número ilimitado de necesidades que facilitan el continuo progreso de la sociedad; soluciones en las que estos sistemas son el eje fundamental de su funcionamiento [6].

Para conseguir tal evolución en el desarrollo de este tipo de tecnologías, se han implementado los Sistemas Empotrados, encargados de ejecutar varias tareas para las que hayan sido previamente programados; usando en conjunto una plataforma software que interactúe de manera precisa para lograr un desempeño optimo, por consiguiente es necesario el uso de Sistemas Operativos de Tiempo Real (SOTR) [7], creados para dar solución de manera eficaz en la generación y corrección de respuestas a procesos bajo parámetros de tiempo restringidos, característica determinante para hacer funcionales a toda clase de aplicaciones como se mostró en la sección 1.1.

Se debe destacar que entre las capacidades que ostentan los SOTR [8] [9], cuenta con la posibilidad de priorizar los procesos por encima de las aplicaciones de los usuarios, esta característica fundamental permite el trabajo con procedimientos de control y procesamiento de forma autónoma, en consecuencia con uno de los aspectos en los que se desempeña de manera frecuente los STR. Por otro lado, estos sistemas operativos son diseñados de modo que se consiga el máximo aprovechamiento de los recursos de procesamiento teniendo en cuenta su reducido tamaño y la simplicidad con la que se ha concebido su estructura, de manera que pueda potenciar la capacidad del Sistema Empotrado en el que se necesite incorporar.

Teniendo en cuenta la ilimitada aplicabilidad de los STR sobre Sistemas Empotrados, empleando un SOTR, se considera de gran importancia, que éstos formen parte en la aprendizaje integral para la generación de conocimiento que se lleva a cabo en los diferentes programas de formación impartidos en la FIET de tal manera que se consiga profundizar el conocimiento en los procesos que permiten el desarrollo de aplicaciones sobre STR.

Por lo tanto, se determina en un factor de gran valor ofrecer herramientas que permitan la ampliación de la labor formativa fomentando experiencias a través de prácticas académicas y en laboratorio que acerquen a los estudiantes con los recursos hardware y software con los que cuenta la facultad para realizar estudios sobre STR a través de sistemas demostrativos que involucren Sistemas Embebidos y Sistemas Operativos de Tiempo Real.

Por otra parte, debido a la naturaleza pública de la Universidad del Cauca, se dispone de recursos limitados, para adquirir el número de equipos necesarios para dotar a cada estudiante de herramientas apropiadas que posibiliten una instrucción idónea en éstos campos del conocimiento.

En consecuencia con lo expuesto anteriormente, se considera de gran importancia generar los medios para facilitar que los estudiantes de la FIET, puedan llevar a cabo pruebas demostrativas en laboratorio que permitan conocer las capacidades de los STR sobre sistemas empotrados, en particular del kit de desarrollo de Stratix II para Nios II, y de los SOTR. De esta manera, se permita generar en ellos nuevas expectativas en el desarrollo de estos conocimientos como parte de su formación académica; potenciando el uso de los limitados recursos con las que cuenta la institución, y así puedan ser aprovechados constructivamente por un gran número de estudiantes.

Con este propósito, se busca recoger de manera apropiada y en su totalidad los aspectos fundamentales que motivan la elaboración del presente trabajo de grado en la modalidad de desarrollo; guiado en la búsqueda de aportar con una herramienta que propicie la construcción de conocimiento y ser partícipe del fortalecimiento académico de los estudiantes de la FIET, dando cumplimiento a los objetivos propuestos a continuación.

1.4 Objetivos

1.4.1 Objetivo General

- Crear una herramienta Software y Hardware para la realización de prácticas de Sistemas Operativos de Tiempo Real sobre dispositivos empotrados.

1.4.2 Objetivos Específicos

- Desarrollar una interfaz de interacción para depuración de aplicaciones con SOTR sobre dispositivos empotrados.

- Generar una guía de pruebas y prácticas para el sistema Stratix II para Nios II, con soporte para al menos dos SOTR.

- Comprobar que la herramienta desarrollada sirve como soporte a las prácticas de laboratorio de un curso de Sistemas Operativos de Tiempo Real SOTR.

1.5 Trabajos relacionados

En la elaboración del presente trabajo de grado, se tiene en cuenta un grupo de proyectos que muestran aspectos de gran importancia relacionados con desarrollos sobre SOTR y Sistemas Empotrados; sin embargo, el contenido de cada uno de ellos no envuelve en su totalidad los elementos que intervienen en la construcción del proyecto aquí desarrollado.

Éstos tratan de experiencias académicas en temas que son estudiados en el presente trabajo de un modo parcial. Sin embargo, como se menciona anteriormente, todos conservan una estrecha relación y aportaron a la construcción final del presente proyecto.

1.5.1 En el Ámbito Internacional

- **The Role of Virtualization in Embedded Systems [10]**
  En este artículo el autor muestra como la virtualización ha cobrado gran importancia en el desarrollo de sistemas embebidos haciendo uso de maquinas

virtuales y como ésta no es capaz de satisfacer las necesidades especiales requeridas por este tipo de sistema, siendo una mejor solución el uso de micro núcleos de alto rendimiento, guiado hacia el futuro del desarrollo de los Sistemas Empotrados.

Se determina como una experiencia que apoya el desarrollo del presente trabajo de grado ya que esta propuesto con el fin de buscar el conocimiento y experimentación sobre una tarjeta de desarrollo que involucra las características de un Sistema Empotrado.

- **Using a Low-Cost SoC Computer and a Commercial RTOS in an Embedded Systems Design Course [11]**

  Este documento describe las experiencias del autor usando un sistema de bajo costo SoC. (system on chip) y un SOTR comercial en un curso de laboratorio de diseño de sistemas empotrados. Este curso cubre temas tanto hardware y software en sistemas empotrados, de modo que los estudiantes tengan la experiencia de laboratorio que refleje con mayor precisión las prácticas actuales del diseño en la industria, el curso culmina en un proyecto de diseño que involucra SOTR y Sistemas Empotrados.

  A diferencia de este articulo, en el trabajo de grado se implementó un sistema de demostración que permitirá a sus usuarios evaluar las características de los SOTR sobre un sistema empotrado particular.

- **Automatic Generation of Fast Timed Simulation Models for Operating Systems in SoC Design [12]**

  En este articulo, los autores presentan un método de generación de modelos de simulación de sistemas operativos, los cuales permitirán al diseñador probar más alternativas de diseño del sistema operativo, mostrando aumento de la velocidad de simulación de acuerdo a experimentos que se realizan.

  La diferencia con el presente trabajo de grado radica en que se utilizan Sistemas Operativos de Tiempo Real: FreeRTOS y Micro C/OS-II, a los cuales no se les modificara ningún tipo de funcionalidad, solo se realizaran pruebas de las características ofrecidas por estos, aplicadas sobre la placa de desarrollo Nios II con procesador adaptable Stratix II.

- **Embedded System Architecture Design Based on Real-Time Emulation [13]**

  En este articulo los autores hacen una introducción de las metodologías de última generación para el diseño de sistemas embebidos, centrándose en el disgregación hardware-software, mostrado las restricciones básicas de los enfoques clásicos, dando como solución para superarlas una nueva metodología de diseño, que consta de dos fases.

  La diferencia con el presente trabajo de grado está en que se implementó un sistema de que permita aprender de un modo demostrativo con elementos

hardware y software que están diseñados y funcionando, con el fin de permitir la realización de pruebas en circuito de estos elementos.

- **A Configurable Hardware Scheduler for Real-Time Systems [14]**

  En este artículo se tiene en cuenta el uso y desarrollo de un sistema que permite atender a posibles sobrecargas de funcionamiento del procesador en la ejecución de tareas realizadas por un Sistema Empotrado, el que trabaja sobre un Sistema Operativo en Tiempo Real (SOTR).

  Para el desarrollo del presente caso se utilizo un kit de desarrollo, el sistema Stratix II para Nios II, específicamente, y dos SOTR operando sobre este, para aprovechar las características que posee y su uso como herramienta para ser utilizado en diversas prácticas académicas de laboratorio.

- **A Modular SystemC RTOS Model for Embedded Services Exploration [15]**

  En este articulo se encuentra el planteamiento de un modelado, una caracterización de un SOTR para determinar cómo debería comportarse bajo una seria de aplicaciones especificas utilizando SystemC como lenguaje de descripción del sistema.

  En el presente trabajo se ha utilizado los recursos disponibles con los que se cuenta en la FIET, esto es, el sistema Stratix II para Nios II, y dadas sus características de fabricación ponerlo en funcionamiento utilizando dos SOTR que se determinaron según su desempeño y características de manejo de Sistemas de Tiempo Real sobre la tarjeta de desarrollo en cuestión.

- **Timber as an RTOS for Small Embedded Devices [16]**

  Para este caso, se tiene la presentación del desarrollo de un lenguaje de programación llamado Timber, que está planteado para ser utilizado en pequeños sistemas empotrados que necesiten operarse en tiempo real.

  Sin embargo para el presente trabajo de grado, en el momento de determinar el sistema operativo SOTR se tuvo en cuenta que haya sido utilizado, probado con éxito en diversas aplicaciones, así como el contenido de sus herramientas para posibilitar un trabajo apropiado sobre la tarjeta de desarrollo caso de estudio.

- **Sistema operativo para SMPs basados en MicroBlaze [17]**

  En el presente artículo se determina aplicar un sistema operativo sobre un sistema embebido específico (soft-core MicroBlaze) que utiliza FPGAs, el cual tiene como propósito los sistemas multiprocesador basados en procesadores soft-core, además de pruebas sobre un sistemas de multiprocesamiento simétrico (SMP).

  Como ya se encuentra relacionado, para el presente caso se trabaja con las herramientas con las que se dispone en la FIET, y de este modo potenciar su usabilidad en pos de la academia, de la formación cada vez más amplia de los

estudiantes en campos del conocimiento con una extensa proyección hacia el futuro como lo son los STR.

- **Impact of Embedded Systems Evolution on RTOS Use and Design [18]**

  Aquí se trata de un estudio del desarrollo de SOTRs teniendo en cuenta un objetivo claro que es el trabajar sobre la electrónica de consumo, de aplicación comercial, y, por lo tanto existe una divergencia de objetivos con el presente trabajo de grado ya que, como se ha anotado, el presente proyecto es de forma fundamental enfocado en la obtención de alcances académico - formativos hacia el interior de la FIET, utilizando las potencialidades y recursos existentes.

- **Real-Time Operating System Services for Realistic SystemC Simulation Models of Embedded Systems [19]**

  En el documento reseñado se formula el mejoramiento de la capacidad de modelado que proporciona SystemC visto como un lenguaje de descripción de sistemas, con el fin de perfeccionar su aplicabilidad sobre los SOTR y Sistemas Empotrados en general.

  En el desarrollo del presente proyecto se definieron de manera apropiadamente los SOTR FreeRTOS y Micro C/OS-II, dadas sus características de desempeño sobre el kit de desarrollo Stratix II para Nios II, con el fin de operar de manera apropiada las herramientas hardware y software que se incluyen en éste y desarrollar de manera satisfactoria las prácticas de laboratorio planteadas, Anexo A.

- **Administración de Interrupciones en Sistemas Operativos de Tiempo Real [20]**

  En el anterior documento se desarrolla un trabajo que busca analizar y dar solución a las falencias presentes en el actual funcionamiento de las llamadas interrupciones sobre los Sistemas Operativos de Tiempo Real, de modo que plantea un modelo alternativo para el tratamiento de estas.

  En el presente trabajo de grado se trabajo haciendo uso de las características de manejo de interrupciones con las que cuenta los SOTR FreeRTOS y Micro C/OS-II, de tal modo que se buscó el máximo aprovechamiento de sus posibilidades para poder ser visualizadas sobre la tarjeta de desarrollo Nios II con procesador adaptable Stratix II perteneciente a la FIET

1.5.2 En el Ámbito Nacional

- **Evaluación del Sistema Operativo µClinux para su Utilización en Sistemas Embebidos [21]**

  En este artículo se prueba el sistema operativo µClinux como alternativa para la implementación de aplicaciones en sistemas embebidos de alta complejidad que hacen uso de microprocesadores de 32 bits. Además se muestran pruebas de

evaluación del funcionamiento de SO en un sistema embebido concreto, revisando las características más importantes.

El presente proyecto, se evaluaron dos sistemas operativos FreeRTOS y Micro C/OS-II, fuero adaptados de forma adecuada al el kit de desarrollo de Stratix II para Nios II con el fin de aprovechar todas sus características, además de permitir el monitoreo de su funcionamiento.

1.6 Solución propuesta

Teniendo en cuenta lo expuesto anteriormente, en lo que se resalta la trascendencia como experiencia educativa y la innumerable aplicabilidad de los Sistemas de Tiempo Real; así como la oportunidad que brinda la FIET de contar con un equipo especializado en la experimentación académica en éste campo, y junto con esto la posibilidad de potenciar y masificar considerablemente su utilización haciendo posible su operación por parte de gran cantidad de estudiantes.

El presente trabajo de grado propone un aplicativo que permita a los estudiantes de laboratorio experimentar de manera práctica con el kit de desarrollo Stratix II con procesador Nios II perteneciente a la facultad de Ingeniería Electrónica y Telecomunicaciones haciendo uso de una interfaz software de monitoreo y depuración que hace posible llevar a cabo practicas de manera remota sobre la tarjeta de desarrollo permitiendo a un grupo de estudiantes acceder a éste recurso a distancia, desde su computador en la sala de laboratorio, siguiendo los cambios de los periféricos de salida con los que cuenta esta placa a través de video streaming utilizando una cámara web dispuesta sobre la tarjeta.

Ésta experimentación se lleva a cabo desarrollando diferentes prácticas en las que se visualizan y estudian de manera demostrativa algunos conceptos básicos que involucran los Sistemas de Tiempo Real, además de poder explorar y poner en funcionamiento las diversas características con las que cuenta el kit de desarrollo caso de estudio; utilizando dos Sistemas Operativos de Tiempo Real FreeRTOS y Micro C/OS-II. Dichas prácticas están consignadas en diferentes guías de laboratorio que incluyen diferentes objetivos a cumplir teniendo como herramienta ejemplos guía previamente codificados y probados, sobre los conceptos que atañen a la temática relacionada con cada práctica.

1.7 Contribuciones

El presente trabajo de grado brinda las siguientes contribuciones:

- Darle uso académico al kit de desarrollo Stratix II para Nios II, perteneciente a la facultad de Ingeniería Electrónica y Telecomunicaciones, de la Universidad del Cauca; buscando explotar al máximo sus características para el aprovechamiento de la mayor cantidad de estudiantes posible multiplexando su acceso a través de la interfaz de monitoreo y depuración remota.

- Fomentar el fortalecimiento de una base de conocimiento que sirva como plataforma en el crecimiento educativo como apoyo a la experimentación y puesta en práctica del conocimiento en lo relacionado a Sistemas Tiempo Real y

Sistemas Empotrados que se adhiera a la formación académica de los estudiantes de los diferentes programas de ingeniería ofrecidos por la FIET.

- Proporcionar la posibilidad de participar en experiencias académicas, de manera demostrativa en el manejo de un Sistema Empotrado, kit de desarrollo Stratix II para Nios II, trabajando con dos SOTR, FreeRTOS y Micro C/OS-II en prácticas de laboratorio.

- Generación de un aplicativo de depuración y monitoreo remoto que permite el control y seguimiento de los procesos que se estén ejecutando sobre la tarjeta de desarrollo Stratix II con procesador adaptable Nios II.

- Elaboración de diversas guías prácticas de laboratorio que exponen de una forma demostrativa a los estudiantes algunos conceptos que atañen a los Sistemas de Tiempo Real, en busca de ofrecer experiencias formativas de gran valor académico con un extenso campo de aplicabilidad.

- Introducir en la FIET la noción del concepto de la utilización de laboratorios remotos, haciendo real la posibilidad hacia el futuro de democratizar y potenciar el uso de los limitados recursos de experimentación en nuevos y diversos campos del conocimiento como lo son los Sistemas de Tiempo Real.

1.8 Contenido

En el presente trabajo de grado se desarrollan los siguientes capítulos.

Capítulo 2, Fundamentos teóricos: En este capítulo se describen los conceptos más importantes relacionados con Sistemas de Tiempo Real, Sistemas Operativos en Tiempo Real, Sistema Empotrado caso de estudio Nios II Debelopment Kit, Stratix II Edition ; que permiten una contextualización apropiada al desarrollo del presente trabajo de grado.

Capítulo 3, Diseño e Implementación del Sistema de Depuración: En este capítulo se hace una descripción detallada de cada modulo del sistema propuesto como solución al problema, además se presenta su modelado en donde se incluye los diagramas de casos de uso, secuencia y clases. Se concluye con la presentación teórica de los conceptos relacionados con las prácticas sobre sistemas operativos de tiempo real.

Capítulo 4, Prototipo y Pruebas: En este capítulo se describe el proceso de construcción y evaluación del funcionamiento del aplicativo de monitoreo y depuración propuesto en el presente trabajo de grado, en donde se incluye su validación, las pruebas realizadas sobre éste junto con los resultados obtenidos.

Capítulo 5, Conclusiones y Trabajos Futuros: En el capítulo final se presenta las conclusiones del presente trabajo de grado como también se propone algunos objetivos para trabajos futuros relacionados con Sistemas en Tiempo Real, SOTR y elementos relacionados.

Capítulo II Fundamentos Teóricos

En éste capítulo se señalan los lineamientos teóricos necesarios para la construcción del presente proyecto, es así como se trata algunos fundamentos sobre los STR debido a que su conocimiento se hace importante en la realización de las prácticas de laboratorio; además se tratan los SOTR con el fin de realizar la elección de los dos sistemas que deberán ser puestos en funcionamiento sobre la tarjeta de desarrollo; también se incluyen los conceptos y características físicas que presenta el Nios II Development Kit, Stratix II Edition ya que se deben tener en cuenta en el planteamiento de las practicas; además se especifica los tipos de proyecto que pueden ser ejecutados sobre la tarjeta de desarrollo, información esencial en el desarrollo de las practicas como también en su ejecución; se hace, también, una descripción de las características principales del entorno de desarrollo de Nios II y posteriormente de las herramientas de desarrollo incluidas en el kit de desarrollo, ya que sobre éstas es que se lleva a cabo la codificación y compilación de los proyectos; la configuración de las características del microprocesador adaptable y la programación de la tarjeta; a continuación se realiza una descripción detallada de los componentes hardware que contiene la placa de desarrollo, información de gran importancia puesto que basados en esta se realiza la personalización de la configuración del microprocesador adaptable que va a ser cargada sobre la FPGA de la tarjeta.

2.1 Sistemas de Tiempo Real STR

Los STR tienen un comportamiento que se determina tanto como por la ejecución de acciones programadas, como por los tiempos en las que ocurren y se procesan; estos lapsos se convierten en el principal parámetro que permite determinar el funcionamiento de estos sistemas, por lo tanto, estas acciones deben ser ejecutadas en periodos de tiempo específicos; además tienen la capacidad de interactuar de forma precisa con su entorno físico, este tipo de sistemas pueden realizar funciones de supervisión o control para ser utilizadas por sí mismo [22].

La característica más importante de estos sistemas es el poseer facultad de ejecutar actividades o tareas en intervalos de tiempo definidos; estos tiempos, pueden ser determinados por un esquema de activación, esto es: en intervalos regulares (periódicos), o también como respuesta a sucesos externos que ocurren de forma aperiódica, y un plazo de ejecución.

En general, están diseñados para ser utilizados cuando existen requerimientos de tiempo muy rigurosos en las operaciones o en el flujo de datos, características frecuentemente relacionadas con sistemas de control. La eficacia de los STR depende tanto de la exactitud de los resultados de cómputo, como también del momento en que los entrega. Estos sistemas pueden ser clasificados de la siguiente manera:

- ***Hard real-time****:* El no cumplimiento de los tiempos definidos para la ejecución de las actividades puede traer consecuencias catastróficas para el sistema [21].

- ***Firm real-time:*** El no cumplimiento de los tiempos definidos causa una reducción apreciable en la calidad de la respuesta del sistema [23].

- ***Soft real-time:*** Las actividades deben ser desarrolladas por el sistema lo más rápido posible, pero pueden no cumplir con los tiempos definidos [21].

2.1.1 Características de los STR

Teniendo en cuenta que en la actualidad, una gran cantidad de aplicaciones tecnológicas basadas en STR, toman cada vez mayor importancia en diversas actividades que la sociedad realiza manera cotidiana, se introducen algunos conceptos que los envuelven. Entre las características más importantes de estos sistemas se encuentran [24]:

- **Gran complejidad:** Los STR suelen ser complejos en cuanto a especificación, diseño y mantenimiento debido a las exigencias que plantea el cumplimiento de los plazos y a pesar de las latencias presentes en todo sistema. Añadido a esto, se suma que en muchas aplicaciones se encuentran geográficamente distribuidos, lo que los hace todavía más complejos.

- **Ejecución simultánea de procesos:** Para cumplir con los plazos especificados para el sistema casi siempre es necesario aprovechar el paralelismo intrínseco a casi todas las aplicaciones y avanzar trabajo en paralelo ejecutando simultáneamente varios procesos. Se debe tener en cuenta que la ejecución simultánea se lleva a cabo en un solo procesador.

- **Determinismo:** Para garantizar que el sistema cumpla con los plazos, se debe evitar cualquier fuente de aleatoriedad en él. Es decir que los tiempos de ejecución de las tareas no deben estar afectados por fenómenos estocásticos que hagan que no sean predecibles.

- **Fiabilidad y seguridad:** Es necesario que el sistema sea fiable y seguro, garantizando tolerancia a fallos en todos los casos por si alguno de los componentes del sistema no funciona adecuadamente o como se esperaba.

2.1.2 Aplicaciones de los STR

Como ha sido reseñado en la sección 1.1, existe un extenso número de aplicaciones en las que los STR son el núcleo sobre el que están construidas, por lo que se concluye que no existen fronteras en las posibilidades de desarrollo que tiene éste tipo de tecnologías dejando por sentado que el estudio de la temática fortalece de manera adecuada el campo del conocimiento de los estudiantes de los programas en Ingeniería ofrecidos por la FIET.

2.1.3 Gestión de procesos en STR

Un proceso es un conjunto de eventos o actividades pueden ser coordinados u organizados, que se ejecutan o suceden, de forma alternativa o simultánea, con un fin determinado. Un proceso puede ser la ejecución de una tarea definida en la sección 3.3.1.

El componente de gestión de procesos es responsable de la creación del proceso, su carga y control de ejecución, además de la interacción de los procesos con señales de eventos, la supervisión de procesos, la asignación de CPU y la terminación del proceso [25].

2.1.4 Gestión de memoria en STR

Los sistemas empotrados suelen tener limitada la cantidad de memoria disponible; eso ocurre por ahorro en costos de producción o restricciones relativas al conjunto del sistema, como tamaño, potencia o peso. Por esta razón, es necesario el controlar como se asigna esta memoria para garantizar una utilización eficiente. Más aun, cuando haya más de un tiempo de memoria, con diferentes propiedades de acceso, dentro del sistema, será necesario indicar al compilador que ubique ciertos tipos de datos en ciertas ubicaciones. Así, el programa será capaz de incrementar sus prestaciones y su predictibilidad, así como de interaccionar con el entorno circundante [26].

2.1.5 Comunicación y sincronización entre procesos en STR

A veces es necesario para un proceso o una *Interrupt Service Rutine* (ISR) comunicar sus datos a otro. Ésta transferencia de datos se denomina comunicación entre procesos; los datos podrá comunicarse entre éstos de dos maneras: a través de variables globales o mediante el envío de mensajes.

Al utilizar variables globales, cada proceso debe asegurarse de que tiene acceso exclusivo a éstas para evitar indeterminaciones. Si se trata de una ISR, la única manera de garantizar acceso exclusivo a las variables comunes es desactivar las interrupciones. Entre los mecanismos que permiten esta comunicación se encuentran:

- **Semáforos:** un semáforo es una variable protegida o tipo abstracto de datos que constituye un método clásico de controlar el acceso por varios procesos a un recurso común en un entorno de programación paralelo. Un semáforo generalmente toma una de dos formas: binario o contador. Un semáforo binario es un indicador de condición que registra si un recurso está disponible o no; estos solo pueden tomar dos valores "verdadero/falso" (bloqueado / desbloqueado), si esta en verdadero el recurso está disponible, y el proceso lo puede utilizar; si es falso, el recurso no está disponible y el proceso debe esperar. Y el semáforo como contador, trabaja llevando la cuenta de los recursos que se encuentren disponibles. Cualquier tipo de semáforo puede ser empleado para prevenir accesos simultáneos a un mismo recurso [25].

- **Cola de mensajes:** es uno de los mecanismos más utilizados para comunicar procesos. Éste permite que un número de mensajes, cada uno de una longitud variable, sea ordenado en una estructura de cola (*First In, First Out* FIFO o basada en prioridades) a la espera de ser leídos. Una vez que ha sido creada, cualquier proceso puede escribir y leer mensajes sobre ellas. Por tanto, es posible que varios procesos envíen mensajes a una misma cola y que múltiples procesos reciban mensajes de una de ellas. Es conveniente borrar las colas de mensajes cuando no van a ser utilizadas, ya que en caso contrario se malgastará de forma innecesaria la memoria del sistema [25].

- **Memoria compartida:** es una de las formas más eficaces que tienen los procesos para comunicarse, consiste en compartir una zona de memoria, tal que para enviar datos de un proceso a otro solo sea escribir en dicha memoria y automáticamente estos datos estarán disponibles para cualquier otro proceso. La utilización de este espacio de memoria común evita la duplicación de datos y el lento trasvase de información entre los procesos.

- **Señales:** es una forma limitada de comunicación entre procesos, es utilizada en Sistemas Operativos compatibles con POSIX (*Portable Operating System Interface*) [27], en esencia se trata de una notificación asíncrona enviada a un proceso con el fin de notificarle la ocurrencia de un evento.

Cuando se habla de sincronización se hace referencia a la coordinación de procesos que se ejecutan simultáneamente para completar una tarea, con el fin de obtener un orden de ejecución correcto y evitar así estados inesperados, independientemente del tipo de interacción existente entre los distintos procesos activos, en un sistema con multiprogramación éstos comparten un conjunto de elementos que deben ser accedidos de forma controlada para evitar situaciones de inconsistencia. Estos elementos compartidos ya sean dispositivos de E/S o zonas de memoria comunes son considerados como críticos y la parte del programa que los utiliza se conoce como región o sección crítica. Es muy importante que sólo un programa pueda acceder a su sección crítica en un momento determinado. Por esta razón, el sistema debe ofrecer mecanismos que hagan posible una correcta sincronización de los distintos procesos activos en los accesos a los recursos que comparten. Algunos de los mecanismos de sincronización utilizados por los Sistemas de Tiempo Real se presentan a continuación:

- **Exclusión mutua (Mutex):** es una forma de acceso en la que un proceso excluye temporalmente a todos los demás de utilizar un recurso compartido, con el fin de asegurar la integridad del sistema. Si el recurso compartido es una variable, la exclusión mutua asegura que, como máximo, un proceso tendrá acceso a ella durante las actualizaciones críticas. En el caso de compartir dispositivos, la exclusión mutua es mucho más obvia si se consideran los problemas que pueden derivarse de su uso incontrolado [28].

- **Monitores:** se trata esencialmente de una colección de datos y de procedimientos para su manipulación junto con una secuencia de inicialización. Un monitor puede considerarse como una estructura estática que se activa únicamente cuando alguno de sus procedimientos es llamado por un proceso en ejecución y se dice, entonces, que el proceso en cuestión entra o tiene acceso al monitor. Solamente un proceso puede estar ejecutándose en el monitor en un instante determinado.

  Una característica básica de los monitores es proporcionar control sobre las operaciones realizadas sobre los elementos compartidos con el fin de prevenir actuaciones dañinas o sin significado. De esta forma, se limitan los tipos de actuaciones proporcionando un conjunto de procedimientos de manipulación fiables y bien probados.

  Los monitores encapsulan los datos utilizados por los procesos concurrentes y permiten su manipulación sólo por medio de operaciones adecuadas y sincronizadas. Nunca existirá peligro de actualización inconsistente por entrelazamiento de llamadas concurrentes ya que los procesos del monitor siempre se ejecutarán en exclusión mutua [28].

2.1.6 Políticas de planificación en STR

Puesto que el número máximo de procesos que es posible ejecutar simultáneamente en un computador es limitado, es necesario definir un conjunto de reglas que permitan

determinar qué proceso o procesos deben ser ejecutadas en cada momento. Estas reglas constituyen los denominados algoritmos o políticas de planificación, y de su elección depende en gran medida el que se satisfagan o no las restricciones temporales impuestas al sistema [29].

Los principales objetivos que persiguen las políticas de planificación son [30]:

- Garantizar la correcta ejecución de todos los procesos críticos.
- Ofrecer un buen tiempo de respuesta a los procesos aperiódicos sin plazo.
- Administrar el uso de recursos compartidos.
- Posibilidad de recuperación ante fallos software o hardware.
- Soportar cambios de modo, esto es, cambiar en tiempo de ejecución el conjunto de procesos. Por ejemplo: un cohete espacial tiene que realizar acciones muy distintas durante el lanzamiento, estancia en órbita y regreso; en cada fase, el conjunto de procesos que se tengan que ejecutar ha de ser distinto.

2.2 Sistemas Operativos de Tiempo Real (SOTR)

En general, un Sistema Operativo (SO) es responsable de la gestión de los recursos hardware y alojamiento de aplicaciones que se ejecutan en un equipo. Un SOTR realiza estas tareas, pero está también especialmente diseñado para ejecutar aplicaciones con un tiempo de respuesta muy preciso y un alto grado de fiabilidad.

Los SOTR proporcionarn apoyo básico para la programación, gestión de los recursos, sincronización, comunicación, y manejo de entradas/salidas. Han evolucionado a partir sistemas especializados de un solo uso, hacia una amplia variedad de sistemas operativos de propósito general. También presentan un notable avance ya que en la actualidad son completamente predecibles que soportan aplicaciones de seguridad crítica hasta aplicaciones que admiten aplicaciones *soft* en tiempo real. Dicho avance incluye el concepto de calidad de servicio (QoS) para sistemas en tiempo real, en aplicaciones multimedia, así como en grandes y complejos sistemas distribuidos de tiempo real.

Estos Sistemas Operativos deben ser predecibles respecto al tiempo de respuesta en el que ejecutan las tareas. Su objetivo no es la rapidez en la que se ejecutan las tareas, sino, en cumplir con los plazos de tiempo preestablecidos para realizar las mismas. Estos sistemas deben ser exactos, garantizando así, que todas las tareas se ejecuten en un límite máximo de tiempo, independientemente de si este es corto o prolongado [31].

Lo anterior resalta la importancia que han adquirido estos Sistemas Operativos, de tal modo que es una razón más para desarrollar su potencialidad en el ámbito académico en la Facultad de Ingeniería Electrónica de la Universidad del Cauca.

2.2.1 Características de los SOTR

Los SOTR, además de establecer condiciones para cumplir las características de los STR, tienen las siguientes propiedades [32]:

- Gestión de dispositivos críticos en tiempo
- Manejo de interrupciones hardware y software
- Eficiente almacenamiento de entradas y salidas

- Permiten acceso desde los programas de usuario a los vectores de interrupción para prestar servicio a los sucesos
- Tolerancia a fallos [33]

2.2.2 Implementaciones de SOTR para Sistemas Empotrados

A continuación se presentan algunos Sistemas Operativos de Tiempo Real explorados durante el proceso de desarrollo del presente trabajo de grado. Sistemas que han sido diseñados para trabajar fundamentalmente sobre Sistemas Empotrados definidos en la sección 2.3 basados en diferentes arquitecturas de microprocesadores; entre los que se encuentra el microprocesador adaptable Nios II el cual configurado sobre la FPGA Stratix II de ALTERA, encargada del control de los recursos presentes en el kit de desarrollo, caso de estudio en el proceso de elaboración de este proyecto.

Estos Sistemas Operativos cumplen con las normas del estándar POSIX [34] que son una familia de estándares de llamadas al sistema operativo definido por la IEEE y especificados formalmente en el IEEE 1003 [35].

A continuación se presenta una tabla comparativa que incluye las características más relevantes acerca de los Sistemas Operativos en Tiempo Real que fueron explorados en el proceso de desarrollo del presente proyecto:

Tabla 2 Sistemas Operativos de Tiempo Real SOTR

Nombre	Licencia	Código	Objetivo de uso	Estado	Plataformas	Ref.
eCos	GNU GPL modificada	Código abierto	Propósitos generales	Activo	ARM/XScale, CalmRISC, 68000/ Coldfire, fr30, FR-V, H8, IA32, MIPS, MN10300, OpenRISC, PowerPC, SPARC, SuperH, V8xx	[36]
embOS	Propietaria	Cerrado	Embebidos	Activo	8/16/32 bit processors	[37]
ERIKA Enterprise	GPL, con excepción	Código abierto	Embebidos	Activo	ARM7, H8 (Hitachi), Nios2 (Altera), PIC24/dsPIC/PIC32 (Microchip), ST10 (ST Microelectronics)/C167 (Infineon), PPC z7 Mamba, AVR, Tricore1, Mico32, S12XS, H8	[38]
FreeRTOS	GNU GPL modificada	Código abierto	Embebido	Activo	ARM, AVR, AVR32, Freescale, Nios II ColdFire, HCS12, IA32, MicroBlaze, MSP430, PIC, Renesas H8/S, 8052, STM32	[39]
LynxOS	Propietario	Código fuente disponible	Embebido	Activo	Motorola 68010, x86/IA-32, ARM, Freescale PowerPC, PowerPC 970, LEON3	[40]
MaRTE OS	GNU, GPL	Código abierto	Embebido	Activo	IA-32	[41]
Neutrino	Propietario	Código fuente disponible	microkernel	Activo	ARM, MIPS, PPC, SH, x86, XScale	[42]
NucleusOS	Propietario	Código fuente disponible	Embebido	Activo	AMD Au1100, ARM, Atmel AT91 series, Atmel Nios II, Freescale iMX, Freescale MCF, Freescale MPC, Marvell PXA series, MTI, NEC uPD6111x, Sharp LH7 series,	[43]

					ST, TI OMAP, TI TMS320 series, Xilinx Microblaze	
OpenRTOS	Propietario	Código fuente disponible	Embebido	Activo	Ver FreeRTOS	[44]
QNX	Mixto	--------	Propósitos generales	Activo	IA32, MIPS, PowerPC, SH-4, ARM, StrongARM, XScale	[45]
RTLinux	GNU, GPL	Código abierto	Propósitos generales	Activo	Igual que LINUX	[46]
ThreadX	Propietario	Disponible para clientes	----------	Activo	ARC, ARM/Thumb, AVR32, BlackFin, ColdFire/68K, H8/300H, Luminary Micro Stellaris, M-CORE, MicroBlaze, PIC24/dsPIC, PIC32, MIPS, V8xx, Nios II, PowerPC, SH, SHARC, StarCore, STM32, StrongARM, TMS320C54x, TMS320C6x, x86/x386, XScale, Xtensa/Diamond, ZSP	[47]
µC/OS-II	Propietario	Disponible bajo licencia	Embebido	Activo	ARM7/9/11/Cortex M1/3, AVR, HC11 /12/S12, Coldfire, Blackfin, Microblaze, NIOS, 8051, x86, Nios II, Win32, H8S, M16C, M32C, MIPS, 68000, PIC24 /dsPIC33/ PIC32, MSP430, PowerPC, SH, StarCore, STM32	[48]
VxWorks	Propietario	----------	Embebido	Activo	ARM, IA32, MIPS, PowerPC, SH-4, StrongARM, xScale	[49]

Después de realizar una exploración concienzuda acerca del los factores más importantes que se deben tener en cuenta en el momento de determinar cuáles son los SOTR apropiados para hacer parte en proceso de construcción del presente proyecto, se observaron aspectos entre los que se resaltan:

- La disponibilidad en el mercado de los SOTR por su tipo de licencia.
- Su costo de adquisición.
- La documentación existente como apoyo a su manejo.
- El objetivo de su creación.
- Las plataformas que pueden soportarlos.
- El lenguaje de programación en el que están implementados, de modo que sea consecuente con otras asignaturas impartidas en la FIET.
- La adaptabilidad que permita para su interacción con la herramienta disponible (kit de desarrollo Nios II con procesador adaptable Stratix II).

Además teniendo en cuenta que el fin de este proyecto es el de proponer un aplicativo que facilite el aprendizaje y la experimentación en laboratorio, donde se pueda poner en práctica los conocimientos ofrecidos en un curso de Sistemas de Tiempo Real.

Teniendo en cuenta lo expuesto, se ha optado por trabajar con los Sistemas Operativos FreeRTOS y Micro C/OS-II, y además porque éstos ofrecen una gran variedad de funcionalidades tales como el manejo de semáforos, colas de mensajes, mutex, prioridades, entre otros; que permiten la implementación de aplicaciones con las especificaciones apropiadas para cumplir con los conceptos más relevantes en el estudio de los Sistemas de Tiempo Real**.**

2.2.3 Sistema Operativo en Tiempo Real: FreeRTOS

Es un SOTR libre, dirigido a pequeños sistemas embebidos en tiempo real. Dado que la mayor parte de su código está escrito en el lenguaje de programación C, es altamente portable y ha sido adaptado a distintas plataformas. Su fuerza es su pequeño tamaño, lo que hace posible ejecutar, donde la mayoría (o todos) los SOTR no se ajustan [50].

Está licenciado bajo GNU *General Public License* (GPL), con una excepción, esta permite que el código fuente de las aplicaciones usen FreeRTOS únicamente a través de la API publicada en su sitio de WEB, para que siga siendo fuente cerrada; lo que permite el uso de FreeRTOS en aplicaciones comerciales sin ser necesario que toda la aplicación sea código abierto. Si se vincula FreeRTOS a otros módulos independientes utilizando sólo la interfaz API de FreeRTOS, puede distribuir el código bajo diferentes licencias a la GPL. Además, permite usar FreeRTOS en aplicaciones comerciales sin pagar regalías. Por supuesto también puede utilizarlo sin la restricción de la API si utiliza la licencia GPL en su propio código [51].

Este sistema operativo es adaptable a gran cantidad de plataformas que han sido desarrolladas por grandes fabricantes, entre los que tenemos la familia de procesadores Nios II de Altera. Este sistema es compatible con la herramienta Nios II IDE con GCC, desarrollada sobre Eclipse.

Características

FreeRTOS es un núcleo de tiempo real escalable, diseñado específicamente para pequeños sistemas integrados. Entre sus características más destacables tenemos [52]:

- Núcleo RTOS libre – incluye opciones de configuración para trabajar como un sistema preferente o cooperativo.
- Es soportado oficialmente por 23 arquitecturas [53].
- FreeRTOS-MPU es compatible con la unidad de protección de memoria (MPU) de Cortex M3.
- Diseñado para ser pequeño, simple y fácil de usar. Normalmente, una imagen binaria del núcleo estará entre 4 y 9 Kbytes.
- Su estructura de código es muy portable, en su mayoría está escrito en C
- Es compatible con tareas y co-rutinas.
- Funcionalidad eficaz de seguimiento de ejecución.
- Opciones de detección de desbordamiento de pila.
- No tiene restricción de software en el número de tareas que pueden ser creadas.
- No tiene restricción de software en el número de prioridades que pueden utilizarse.
- No impone restricciones a la asignación de prioridades – puede asignarse la misma prioridad a más de una tarea.
- Provee los servicios de colas, semáforos binarios, semáforos como contadores, semáforos recursivos y mutexes para la comunicación y sincronización entre tareas, o entre tareas e interrupciones.
- Mutexes con herencia de prioridad.
- Herramientas de desarrollo libre (puertos Cortex-M3, ARM7, MSP430, H8 / S, AMD, AVR, x86 y 8051).
- Código fuente de software libre.
- Sin regalías.

2.2.4 Sistema Operativo en Tiempo Real: Micro C/OS-II

El Micro C/OS-II es un SOTR portable, cuyo núcleo es multitarea, determinista, rápido, escalable y ROMable, DSPs, para microprocesadores, y microcontroladores [54].

Micro C/OS-II administra hasta 250 de tareas de aplicación. Micro C/OS-II incluye: semáforos; indicadores de eventos; semáforos de exclusión mutua que eliminan inversiones de prioridad; buzones de mensaje y colas; administración de tareas, gestión de tiempo y temporizadores; y administración de bloques de memoria de tamaño fijo.

Micro C/OS-II se puede escalar (entre 5 Kbytes a 24 Kbytes) para contener únicamente las características requeridas para una aplicación específica. El tiempo de ejecución para la mayoría de los servicios prestados por Micro C/OS-II es a la vez constante y determinista; los plazos de ejecución no dependen del número de tareas que se ejecutan en la aplicación [55].

Características

A continuación se citan algunas de las características más importantes de este Sistema Operativo [56]:

- Se ejecuta en un gran número de arquitecturas de procesador con puertos fácilmente descargables.
- Escalabilidad entre 5 y 24 KBytes
- Tiempo máximo de desactivación de interrupciones: 200 ciclos de reloj.
- Robusto para satisfacer los rigurosos requisitos de seguridad de sistemas críticos.
- Código fuente implementado en ANSI C.

Este sistema operativo es adaptable a gran cantidad de plataformas que han sido desarrolladas por grandes fabricantes, entre los que tenemos la familia de procesadores Nios II de Altera. Este sistema está integrado dentro de la herramienta Nios II IDE con GCC, desarrollada sobre Eclipse [57].

El código fuente de Micro C/OS-II, y el objeto, pueden ser distribuidos libremente (para estudiantes) por universidades acreditadas sin necesidad de una licencia, siempre que no haya involucrada ninguna aplicación comercial. No necesita de consecución de licencias si Micro C/OS-II se utiliza para uso educativo. Se debe obtener una " Objeto de la Licencia de Distribución de Código" para incrustar Micro C/OS-II en un producto comercial. En otras palabras, debe obtener una licencia para poner Micro C/OS-II en un producto que se vende con la intención de obtener un beneficio.

2.3 Sistema Empotrado: Nios II Development Kit, Stratix II Edition

Teniendo en cuenta que ésta tarjeta de desarrollo es catalogada como un sistema empotrado, se hace una breve introducción acerca de éstos sistemas.

Un Sistema Empotrado es una aplicación en sistemas informáticos, a diferencia de otros tipos de sistemas, como PC o supercomputadores. Sin embargo, debido a la evolución constante y los avances en la tecnología y la disminución progresiva en el costo de la implementaciones hardware y componentes software. Una definición de Sistema

Empotrado es difícil de precisar, de tal modo que se presenta una noción general que describe el concepto que envuelve éstos sistemas [58].

- Los Sistemas Empotrados presentan mayores limitantes en la funcionalidad de hardware y/o software que un computador personal (PC).
- Un Sistema Empotrado está diseñado para realizar una función dedicada. Más dispositivos embebidos están diseñados principalmente para una función específica.
- Un Sistema Empotrado es un sistema informático de mayor calidad y requisitos de confiabilidad que otros tipos de sistemas informáticos.

Los dispositivos electrónicos en casi todos los mercados de la ingeniería se clasifican como sistemas empotrados [58]. Sin embargo, fuera de ser "tipos de sistemas informáticos", la caracterización específica sólo sugiere decir de la amplia gama de dispositivos concebidos como sistema empotrados es que no existe una definición única que refleje a todos.

A continuación se presenta una descripción abreviada de los componentes principales a nivel de hardware y software con los que cuenta el kit de desarrollo Stratix II para Nios ll con los que se ha trabajado en el desarrollo del presente proyecto, fundamentales en el desarrollo de las practicas, además de la configuración del microprocesador adaptable para posteriormente ser cargado sobre la FPGA.

2.3.1 Desarrollo de Software Sobre el Nios II

Las herramientas de generación de software de Nios II incluidas en el kit, proporcionan un entorno de desarrollo de software solido que funciona para todos los sistemas que utilizan un procesador de Nios II; contienen un conjunto de comandos de gran alcance, utilidades y scripts para administrar las opciones de compilación de las aplicaciones, los paquetes, además las librerías de software para su utilización, que, junto con una completa documentación, ayuda en línea, y diseños de referencia potencian el desarrollo de software sobre una la placa de desarrollo.

Ofrecen un entorno de desarrollo integrado plenamente desde el origen bajo Eclipse; éste está concebido en su totalidad como un plug-in de Eclipse para que pueda ser considerado como un estándar de la industria a nivel mundial. Este entorno de desarrollo se centra en mejorar la fabricación de software para aplicaciones complejas y software basado en diseño sobre placas de desarrollo, esto apunta a mejorar el rendimiento para el trabajo con un procesador Nios II.

A continuación se presenta una introducción y conceptos más importantes de las herramientas de desarrollo de software para el procesador Nios II, se trataran aspectos importantes del entorno de desarrollo de software de Nios II y una breve descripción de las herramientas software que están incluidas en el kit de desarrollo Stratix II para Nios II, y que forman parte activa en el desarrollo de las guías y practicas académicas propuestas en el presente proyecto.

Para obtener información más detallada de las siguientes secciones tener en cuenta las siguientes referencias: [59], [60], [61].

2.3.1.1 Entorno de desarrollo de software de Nios II

La EDS de Nios II proporciona un entorno de desarrollo de software que funciona para todos los sistemas con procesador Nios II. Con la EDS de Nios II ejecutándose en un equipo host, una FPGA de Altera y un cable de descarga *Joint Test Action Group* JTAG, como el cable de descarga Altera USB-Blaster con el que cuenta el Kit de desarrollo que es el caso de estudio, se puede escribir programas para hacer factible la comunicación con cualquier sistema con procesador Nios II. El acceso al procesador es el mismo, independientemente de si se implementa un dispositivo con sólo un procesador Nios II, o si está empotrado en un complejo sistema multiprocesador. Por lo tanto, no necesita gastar tiempo para crear manualmente un mecanismo de interfaz.

El EDS de Nios II proporciona dos flujos distintos de diseño e incluye muchas herramientas propietarias y de código abierto (tales como la cadena de herramientas GNU C/C ++) para la creación de programas de Nios II. El EDS de Nios II automatiza la creación de paquetes (BSP) que se utiliza como soporte de placa para sistemas basados en Nios II, eliminando la necesidad de dedicar tiempo a crear manualmente estos paquetes. El BSPs de Altera contiene la Capa de Abstracción de Hardware (HAL) de Altera, un RTOS opcional, y controladores de dispositivos. El BSP ofrece un entorno de ejecución para C / C + +.

2.3.1.2 Programas desarrollados sobre Nios II.

Cada programa que se desarrolla sobre un procesador Nios II, consiste en un proyecto de aplicación, proyectos de bibliotecas opcionales del usuario y un proyecto BSP. Generan un programa que crea un ejecutable y un archivo de enlace con formato (.elf) el cual es cargado en la placa de desarrollo y corre en el procesador Nios II.

Las herramientas de generación de software de Nios II, permiten crear proyectos de software de manera práctica y ágil, con una completa documentación tanto acerca de las herramientas con las que se trabaja, como sobre ejemplos pre elaborados, cada uno de los cuales se basa fundamentalmente en un archivo make.

2.3.1.3 Makefiles y herramientas de construcción de software

El makefile es el componente central de un proyecto de software Nios II, si el proyecto se crea con las herramientas de generación de software de Nios II, o en la línea de comandos. En éste se describe todos los componentes de un proyecto de software, cómo se compila y enlaza. Con un makefile y un conjunto completo de archivos fuente de C/C++, su proyecto de software Nios II está totalmente definido.

2.3.2 Tipos de proyecto de software de Nios II

A continuación se listan los tipos de proyecto que toman parte en el desarrollo de un proyecto de generación de hardware en el que se construye usando como objetivo un procesador Nios II:

2.3.2.1 Proyecto de aplicación

Un proyecto de aplicación de Nios II en C / C + +, consiste en una colección de código fuente, además de un Makefile. Una característica típica de una aplicación es que uno de

los archivos de código fuente contiene la función main ().Una aplicación incluye el código que llama a las funciones en las bibliotecas y el BSPs. El Makefile compila el código fuente y lo vincula con un BSP y una o más bibliotecas opcionales, para crear un archivo con extensión .elf.

2.3.2.2 Proyecto de biblioteca de usuario

Conformado por una colección de código fuente compilado, para crear un único archivo de biblioteca de usuario (.a). Las bibliotecas contienen a menudo funciones reutilizables para fines generales que pueden compartir aplicaciones en varios proyectos. Por ejemplo, una colección de funciones aritméticas comunes. Una biblioteca de usuario no contiene la función *main().*

2.3.2.3 Proyecto BSP

Un proyecto de Nios II BSP es una biblioteca especializada que contiene el código de soporte específico del sistema. Un BSP proporciona un entorno de ejecución de software personalizado para un procesador en un sistema constructor SOPC. El EDS de Nios II proporciona herramientas para modificar la configuración que controla el comportamiento del BSP.

Un BSP contiene los siguientes elementos:

- Una Capa de Abstracción de Hardware (HAL).
- Una librería estándar Newlib de C.
- Controladores de dispositivos.
- Paquetes de software opcionales.
- Un Sistema Operativo de Tiempo Real (RTOS) opcional.

- **HAL:**

  Un HAL proporciona un entorno de ejecución de C y C++, similar a UNIX. Proporciona E/s genéricas de los dispositivos, lo que permite escribir programas que tienen acceso a hardware mediante las rutinas de librerías estándar newlib de C, tales como *printf().* HAL minimiza (o elimina) la necesidad de tener acceso a registros de hardware directamente al control y comunicarse con dispositivos periféricos. El HAL sirve como un paquete de controladores de dispositivo para sistemas con procesador Nios II, que proporciona una interfaz permitiendo el uso de los periféricos en el sistema.

- **Librería estándar newlib de C:**

  Newlib es una implementación de código abierto de las librerías estándar de C destinada para su uso en sistemas empotrados, integrada en el HAL, es precisa para ser utilizada sobre el procesador Nios II, esta librería se encuentra muy bien documentada. Es una colección de rutinas comunes tales como *printf (), malloc ()* y *open ().*

- **Controladores de dispositivo:**

Cada controlador de dispositivo administra un componente de hardware. De forma predeterminada, el HAL crea una instancia de un controlador de dispositivo para cada componente en el sistema generador de SOPC que necesita un controlador de dispositivo. En el entorno de desarrollo de software de Nios II, un controlador de dispositivo está asociado con un componente específico de SOPC Builder. Además puede tener la configuración que afecta su compilación. Estos valores se convierten en parte de la configuración del BSP.

- **Paquetes de software opcional:**

  Un paquete de software es el código fuente que opcionalmente se puede agregar a un proyecto BSP para proporcionar funcionalidad adicional.

- **Sistema Operativo en Tiempo Real (RTOS) opcional**

  El IDE de Nios II incluye una implementación del RTOS externo, Micro C/OS-II, que se puede incluir opcionalmente en el BSP.

2.3.3 Entorno de desarrollo integrado de Nios II

El entorno de desarrollo de escritorio para el IDE Nios II, es el entorno de trabajo en el que se puede editar, compilar y depurar los programas. Cada entorno de trabajo contiene uno o más opciones. A continuación se presenta una breve descripción de algunos de los elementos más relevantes del entorno de desarrollo integrado de Nios II.

- **Flujo de Diseño del IDE de Nios II para la creación de Programas**

El flujo de diseño del IDE de Nios II permite crear, modificar, ejecutar y depurar programas Nios II con la interfaz gráfica de usuario (GUI) del IDE de Nios II. El IDE crea y gestiona sus makefiles para sus proyectos. El IDE de Nios II se basa en el popular entorno IDE de Eclipse y herramientas de desarrollo plug-ins de Eclipse para C/C++, (CDT). Con asistentes que proporcionan ayuda en la creación y configuración de proyectos, el IDE de Nios II es fácil de usar, y es especialmente útil para el aprendizaje en el uso de la Nios II.

- **Perspectivas, editores y vistas**

El apoyo de editores en diversos tipos de vistas, proporciona presentaciones alternativas y formas de navegar sobre la información en su entorno de trabajo. Cada ventana de edición contiene uno o más perspectivas. Cada perspectiva proporciona un conjunto de capacidades para llevar a cabo un tipo específico de tarea. Como por ejemplo, se cuenta con una perspectiva del desarrollo Nios II para trabajar con C y C++. Suministra además vistas de editores de apoyo, proporcionan presentaciones alternativas y formas de navegar la información en su entorno de trabajo. Un editor le permite abrir y editar un recurso de proyecto (es decir, un archivo, carpeta o proyecto).
El IDE de Nios II le permite ejecutar o depurar cualquier proyecto ya sea en una placa destino, en el simulador de grupo de instrucciones de Nios II (ISS), o en la herramienta de simulación *ModelSim*.

- **SOPC Builder**

SOPC Builder es una herramienta de desarrollo de sistemas de gran alcance que automatiza las tareas de integración de componentes de hardware que incluyen procesadores, periféricos, y memorias. Le permite definir y generar un completo sistema programable en-un-chip; (SOPC) en mucho menos tiempo que los métodos tradicionales de integración manual de elementos.

Usando SOPC Builder, se especifica los componentes del sistema en la GUI de Nios II, y SOPC Builder genera la lógica de interconexión de forma automática. Provee los archivos HDL que definen todos los componentes del sistema, y un archivo HDL de nivel superior que conecta todos los componentes juntos. Además, proporciona funciones para mejorar el software de grabación y acelerar la simulación del sistema. SOPC Builder está incluido en el software con el que cuenta el kit de desarrollo caso de estudio.

- **El archivo de descripción de sistema system.h**

El archivo de system.h proporciona una descripción de software completo del sistema hardware de la Nios II. Este archivo contiene la información acerca de qué hardware está presente en este sistema.

El archivo system.h describe cada uno de periféricos en el sistema y proporciona los siguientes datos:

- La configuración de hardware de los periféricos.
- La base de las direcciones.
- La prioridad IRQ (si es necesaria).
- Un nombre simbólico para cada periférico.

2.3.4 Descripción de herramientas del IDE de Nios II

A continuación se presenta una descripción general de las herramientas de las que dispone el IDE de Nios II. En una sección posterior se hará una descripción detallada junto con las instrucciones necesarias para comenzar a desarrollar cualquier tipo de proyecto de software sobre el kit de desarrollo Stratix II para Nios II, que se presenta como caso de estudio.

- **IDE Nios II**

El entorno de desarrollo integrado (IDE) Nios II es la interfaz de usuario gráfica de desarrollo software (GUI) para el procesador Nios II. Todas las tareas de desarrollo de software pueden lograrse dentro del IDE de Nios II, incluyendo la edición, construcción y depuración de programas. El IDE de Nios II es la ventana a través de la cual pueden ser lanzadas todas las otras herramientas.

El IDE de Nios II se basa en el popular marco IDE de Eclipse y el Kit de herramientas (CDT) de desarrollo Eclipse C plug-ins. El IDE de Nios II es una interfaz de usuario ligera que manipula otras herramientas detrás de su interfaz, la protege de los detalles de las herramientas de línea de comandos y presenta un entorno de desarrollo unificado. Si es necesario, los procesos de desarrollo de software puede ser escritos y ejecutados de forma independiente de la GUI.

- **Programador de Flash**

El IDE Nios II incluye una utilidad de programador flash que le permite programar los chips de memoria flash en una placa de destino. Esta herramienta soporta programación flash en cualquier placa, incluyendo las de desarrollo construidas por Altera además de las personalizadas. Este programador facilita la programación flash para los siguientes propósitos:

- Código y datos ejecutables
- Código de secuencia de inicio para copiarlo de flash a RAM y, ejecutarlo desde la RAM.
- Archivo de subsistemas HAL.
- Datos de configuración de hardware de la FPGA.

- **Simulador de conjunto de instrucciones**

Altera proporciona un Simulador de Conjunto de Instrucciones (ISS), por su sigla en Inglés, para el procesador Nios II. La ISS está disponible dentro del IDE de Nios II, permite empezar a desarrollar programas antes de la plataforma hardware destino esté lista de modo que el proceso para ejecutar y depurar programas en esta herramienta sea tan fácil como cuando se ejecuta en un dispositivo de hardware real.

- **Programador Quartus II**

El programador Quartus II es parte de la Suite Completa de diseño de Quartus II, sin embargo, el IDE Nios II puede iniciar el programador de Quartus II directamente. Proporciona un entorno de diseño multiplataforma completo que se adapta fácilmente a las necesidades específicas de cualquier diseño. El software de Quartus II también permite utilizar la interfaz gráfica de usuario de Quartus II, interfaz de la herramienta EDA o interfaz de línea de comandos para cada fase del flujo de diseño. Su programador incorporado permite descargar nuevos archivos de configuración de la FPGA hacia la placa.

2.3.5 Visión general de las características del hardware

Para obtener información más detallada de las siguientes secciones revisar las siguientes referencias: [62], [63], [64].

La placa de desarrollo Nios, Stratix II Edition, proporciona una plataforma hardware para desarrollar sistemas embebidos basados en los dispositivos Stratix II de Altera. Ésta proporciona las siguientes características organizadas en la Figura 2-1:

- Una FPGA Stratix II con más de 13.500 módulos de lógica adaptativa (ALM) y 1,3 millones de bits de memoria en el chip
- 16 MBytes de memoria flash
- 2MBytes de SRAM sincrónica
- 32MBytes de SDRAM de tasa doble de datos (DDR)
- Lógica para configurar la FPGA desde la memoria flash
- Un dispositivo Ethernet MAC/PHY y un conector RJ45

- Conector CompactFlash Tipo I para tarjetas CompactFlash
- Conector PMC de 32 bits capaz de operar a 33 MHz y 66 MHz
- Conector Mictor para depuración de hardware y software
- Cuatro interruptores conectados a pines de usuario de E/S de la FPGA
- Ocho LEDs conectados a pines de usuario de E/S de la FPGA
- 2 Conectores de expansión de prototipos (Proto1&Proto2)
- Puerto serie RS-232 DB9
- Conectores JTAG para dispositivos Altera
- Circuitos de reconfiguración en el encendido
- Doble display de 7 segmentos
- Oscilador de 50 MHz
- Entrada externa para reloj

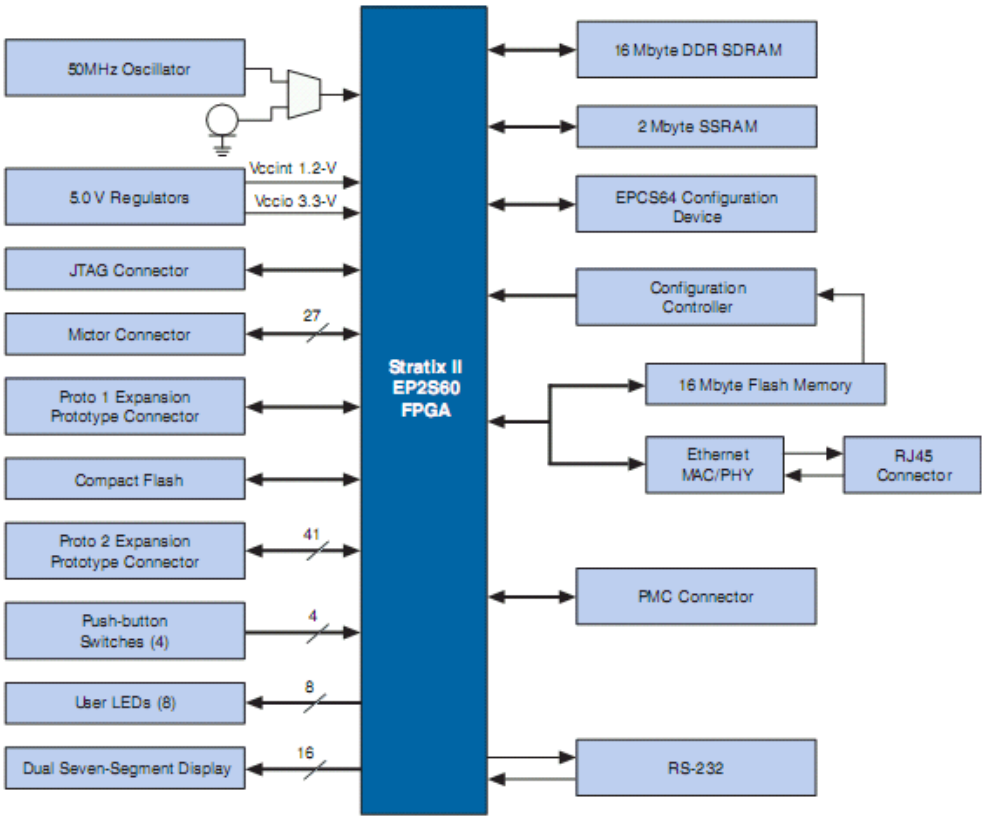


Figura 2-1 Diagrama de Bloques, placa de Desarrollo Nios, Stratix II Edition

A continuación se presentan los componentes más importantes de la placa de desarrollo Nios, en la Figura 2-2 se presenta la ubicación de cada uno de estos sobre la placa, de los cuales se realiza una breve descripción.

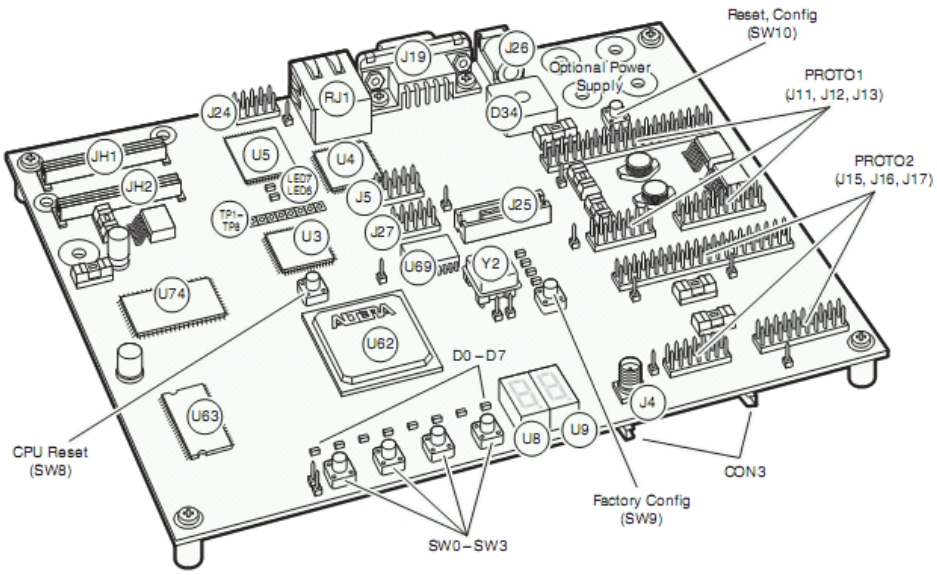


Figura 2-2 Placa de Desarrollo Nios, Stratix II Edition

a) FPGA EP2S60F672C3N Stratix II (U62)

La familia Stratix de Altera, son las FPGA de alta gama de la cual su segunda generación es la Stratix II y sus características más destacables son las siguientes:

Las FPGAs Stratix II de Altera implementan dos tipos de Phase Loched Loop PLL diferentes, en total 12. También contiene 60,440 Elementos Lógicos, 255 bloques de memoria RAM de 4 Kbits más paridad, con un total de 2, 544,192 bits de memoria RAM, 144 bloque multiplicadores y 718 pines de E/S.

b) Sistema Procesador NIOS II

Un sistema de procesador Nios II es equivalente a un microcontrolador éste sistema consiste de un núcleo procesador Nios II, un conjunto de periféricos, memoria en chip, e interfaces para memorias fuera del chip, todos implementados en un único dispositivo de Altera como se muestra en la Figura 2-3. Al igual que todas las familias de microcontroladores, los sistemas de procesador Nios II utilizan un conjunto de instrucciones coherente y un modelo de programación.

El procesador Nios II es un procesador soft-core configurable, en contraposición a un microcontrolador fijo, estándar. En este contexto, configurable significa que se puede agregar o quitar características sobre un sistema base para cumplir los objetivos de rendimiento o precio. Soft-core significa que el núcleo del procesador puede ser dirigidos a cualquier familia de FPGAs de Altera.

Por otra parte, los sistemas de procesador de Nios II tienen un conjunto periférico flexible, lo cual es una de las más notables diferencias con los microcontroladores no adaptables. Debido a la naturaleza soft-core del procesador Nios II, puede crear fácilmente sistemas de procesador Nios II hechos a medida con el conjunto exacto de periféricos necesario para las aplicaciones de destino. . Al igual que los periféricos, las instrucciones pueden

ser personalizadas para aumentar el rendimiento del sistema aumentando el procesamiento con hardware personalizado.

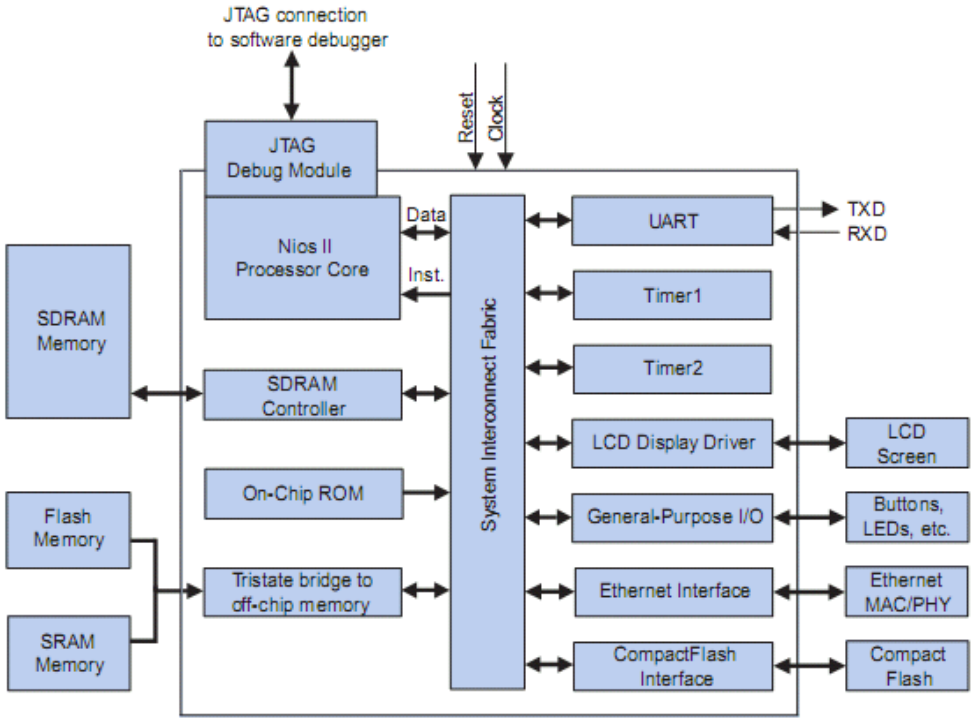


Figura 2-3 Sistema de procesador Nios II

El Nios II es un núcleo procesador configurable de 32 bits de propósito general, basado en una arquitectura tipo Harvard, dado que usa buses separados para instrucciones y datos, que se puede implementar en alguna de las tres versiones disponibles, Nios II/f ("fast"), Nios II/s ("standard"), Nios II/e ("economy"), según se busque minimizar el consumo de recursos de la FPGA o maximizar el rendimiento del procesador, para el caso en estudio se cuenta con un procesador Nios II standard, que es la versión con *pipeline* "basada en filtros" de 5 etapas que dotada de unidad aritmético-lógica ALU, busca combinar rendimiento y consumo de recursos, entre sus principales características se encuentran:

- Tamaño de palabra asignado 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.
- Unidad de gestión de memoria (MMU) para soportar sistemas operativos que la requieran.
- Operaciones de multiplicación y división de 32 bits.
- Instrucciones dedicadas para multiplicaciones de 64 y 128 bits.
- Unidad de protección de memoria (MPU).

La arquitectura Nios II define las siguientes unidades funcionales mostradas en la Figura 2-4:

- Registro de archivos
- Unidad aritmética y lógica (ALU)
- Interface para instrucciones lógicas personalizadas
- Controlador de excepciones
- Controlador de interrupciones internas y externas
- Bus de instrucciones
- Bus de datos
- Unidad de gestión de memoria (MMU)
- Unidad de protección de memoria (MPU)
- Interfaces de memoria para instrucciones y datos unidas estrechamente
- Modulo de depuración JTAG

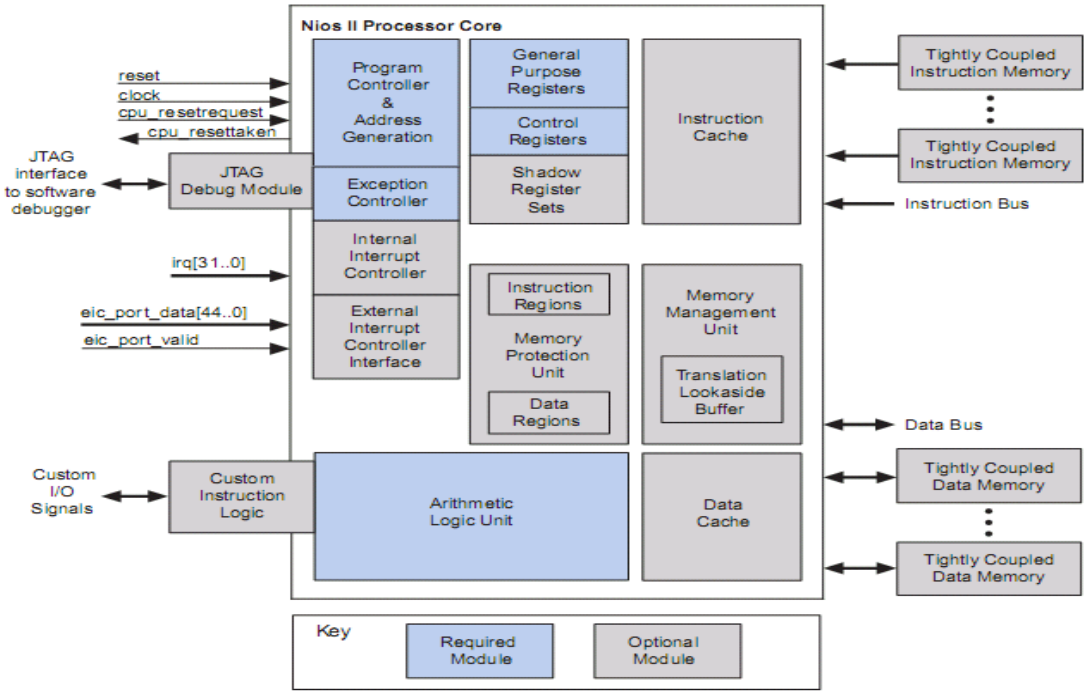


Figura 2-4 Núcleo Procesador NIOS II

c) Interruptores Push-Button SW0-SW3

SW0 - SW3 son botones interruptores de contacto momentáneo utilizados para proporcionar estímulos a los diseños programados en la FPGA. Cada interruptor está conectado con una resistencia pull-up a un pin de E/S de propósito general de la FPGA. Cada pin de E/S percibe un 0 lógico cuando el interruptor correspondiente está presionado.

d) LEDs individuales D0-D7

Esta placa de desarrollo Nios II dispone de ocho LEDs individuales conectados a la FPGA. D0 - D7 se conectan a pines de E/S de propósito general de la FPGA. Cuando un pin de E/S percibe un 0 lógico, el correspondiente LED se enciende.

e) Displays 7 segmentos U8-U9

Estos se encuentran conectados a la FPGA y cada uno de sus segmentos se encuentra controlado por un pin de propósito general de la FPGA. Cuando un pin percibe un 0 lógico el correspondiente segmento se enciende.

f) Chip SSRAM U74

Chip Cypress SSRAM de 2 MBytes y 32-bits. El chip está tasado para accesos síncronos de hasta 167 MHz. U74 se conecta a la FPGA para que pueda ser utilizado por el procesador empotrado Nios II como memoria de propósito general.

g) Chip SDRAM DDR U63

Es un chip de memoria SDRAM DDR de 32MBytes de Micron. Altera proporciona un controlador de memoria SDRAM DDR que permite al procesador NiosII acceder al dispositivo SDRAM DDR como memoria de gran tamaño, direccionable en forma lineal.

h) Memoria Flash U5

U5 es un 8-bits, AMD 16MByte dispositivo de memoria flash conectado a la FPGA. . U5 se puede utilizar para dos propósitos: el primero como memoria de propósito general y de almacenamiento no volátil y el segundo para guardar datos de configuración de la FPGA que se utilizan por el controlador de configuración para cargar la FPGA al momento del encendido. El software de desarrollo Nios II incluye subrutinas para escribir y borrar la memoria flash.

i) Ethernet MAC/PHY U4 y Conector RJ45 RJ1

El chip LAN91C111 (U4) es un control de acceso al medio Ethernet 10/100 e interfaz física (MAC / PHY). Los pines de control del chip están conectados a la FPGA para que el sistema Nios II pueda acceder a redes Ethernet a través del conector RJ-45. Las herramientas de desarrollo del Nios II incluyen componentes hardware y software que permiten los sistemas de procesador Nios II comunicarse con el dispositivo Ethernet LAN91C111 ubicados como se muestra en la Figura 2-5.

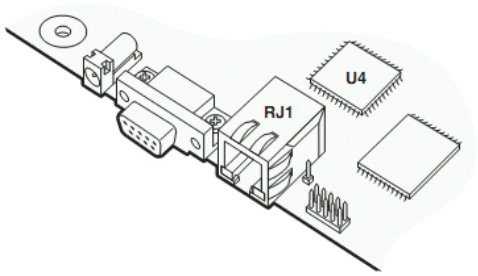


Figura 2-5 Conector Ethernet RJ-45

j) Conector Serial J19

Es un conector estándar serie DB-9. Normalmente se utiliza para comunicación entre la FPGA y un computador central a través de un cable serie RS-232. Se utiliza un driver de línea (U52) entre el conector DB-9 y la FPGA ya que esta no puede manejar los niveles de voltaje RS-232 directamente.

Este conector es capaz de transmitir todas las señales RS-232 Figura 2-6. Alternativamente, se puede configurar la FPGA para que utilice sólo las señales que necesita, como recepción de datos RXD y transmisión de datos TXD. Dos LEDs son conectados a las señales RXD y TXD para indica visualmente cuando se están transmitiendo o recibiendo datos.

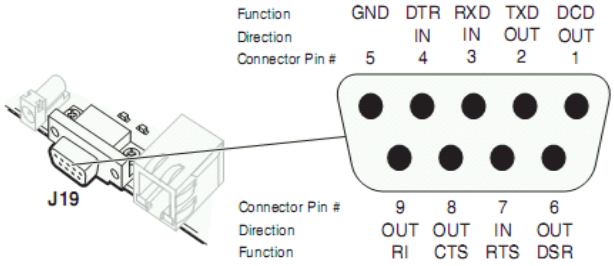


Figura 2-6 Conector Serial DB-9

k) Conectores de Prototipos de Expansión PROTO1-PROTO2

Conectores de expansión conectados a 41 pines de E/S de la FPGA, estos suministran voltajes de 3,3 V y 5,0 V para ser usados por otros periféricos. Los conectores J11, J12, J13 en conjunto forman el PROTO1 y J15, J16 y J17 como se muestran en la Figura 2-7 en conjunto forman el PROTO2.

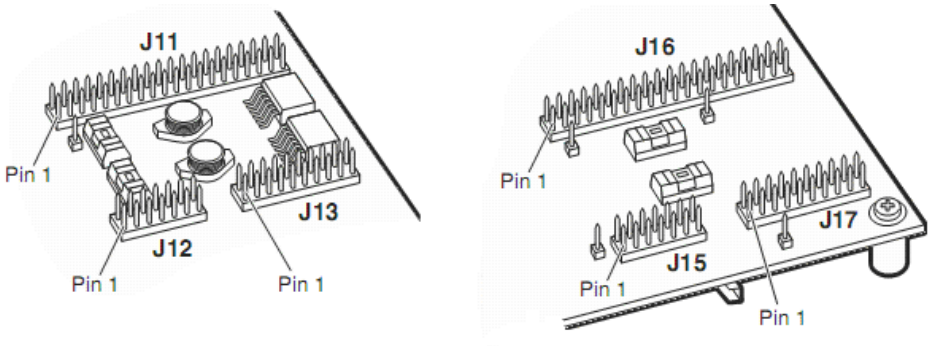


Figura 2-7 Conectores de Expansión PROTO1-PROTO2

l) Conector CompactFlash CON3

Este permite a los diseños de hardware implementados sobre la FPGA tener acceso a tarjetas *CompactFlash* Figura 2-8.

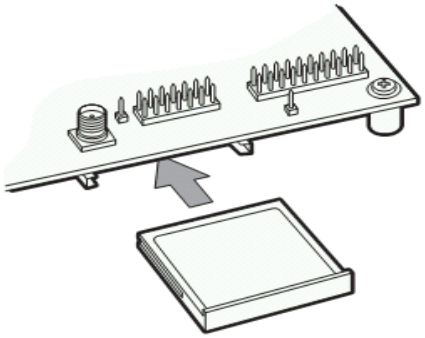


Figura 2-8 Conector *CompactFlash*

m) Conector PMC JH1- JH2

Este es un conector para tarjetas Mezzanine PCI (PMC) [555], formado por los puertos JH1 y JH2, como se muestra en la Figura 2-9, permite a los sistemas Nios II en la FPGA una interfaz con otras tarjetas usando el factor de forma PMC estándar de 32 bits, este conector es capaz de funcionar a 33 MHz o 66 MHz, y se configura como el anfitrión PMC.

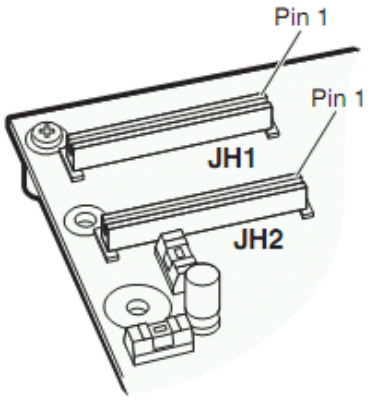


Figura 2-9 Conector PMC

n) Conector Mictor J25

Este conector de la Figura 2-10 se puede utilizar para transmitir hasta 27 señales de alta velocidad de E/S con muy poco ruido a través de un cable blindado Mictor, se puede utilizar como un puerto de depuración para el procesador Nios II o como de conector E/S de propósito general a la FPGA, veinticinco de las señales del conector Mictor se utilizan como datos, y dos señales se utilizan como entrada y salida de reloj.

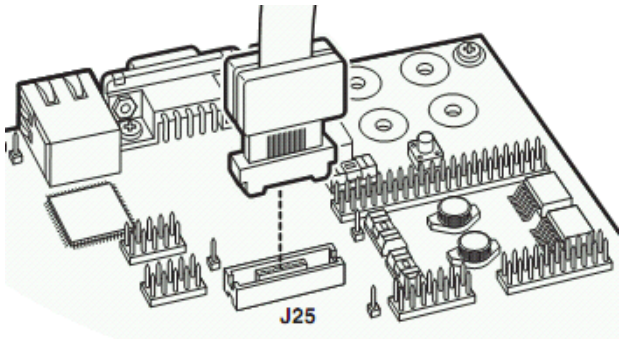


Figura 2-10 Conector Mictor

o) Puntos de Prueba TP1-TP8

Son puntos de prueba conectados a pines de E/S de la FPGA, la cual está diseñada para dirigir señales de E/S hacia estos, para la realización de pruebas de funcionamiento.

p) Dispositivo de Configuración Serial EPCS64 U69

Es un dispositivo de configuración de serial conectado a la FPGA, los dispositivos de configuración serial son memorias flash con una interfaz en serie que puede almacenar datos de configuración y cargar los datos en el FPGA al encenderse o reconfigurarse, además puede almacenar los datos de programa del Nios II.

q) Controlador de configuración de dispositivos U3

Es un dispositivo que viene pre programado para manejar las condiciones de reset de la tarjeta y configuración de la FPGA a partir de datos almacenados en la memoria flash y el dispositivo de configuración serial, este controlador se encuentra conectado a 4 LEDs que indican la configuración cargada en la FPGA cuando se enciende la tarjeta. Los cuales pueden indicar estados de error o éxito en la transferencia de datos de fábrica o de usuario desde la memoria flash a la FPGA.

r) Botones de Reset y Configuración

La tarjeta de desarrollo Nios utiliza tres switches dedicados SW8, SW9 y SW10 para las siguientes funciones:

- CPU Reset - SW8: El diseño de referencia del Nios II programado de fábrica trata este como un botón de restablecimiento de la CPU, cuando este se presiona, se restablece el diseño de referencia de Nios II y se inicia la ejecución de código desde su dirección de restablecimiento.

- Factory Config - SW9: Este botón ejecuta los comandos del controlador de configuración para volver a configurar la FPGA con la configuración de fábrica.

- Reset, Config - SW10: Es el botón de restablecimiento en el encendido. Cada vez que se presiona SW10, el controlador de configuración intenta volver a configurar la FPGA.

s) Conectores JTAG J24 y J5

La tarjeta de desarrollo Nios tiene dos conectores JTAG de 10-pines (J24 y J5) Figura 2-11 Conectores JTAG compatible con cables de descarga Altera, cada uno de estos conectores JTAG se conecta a un dispositivo de Altera, J24 se conecta a la FPGA (U60) y J5 se conecta al dispositivo Controlador de configuración (U3).

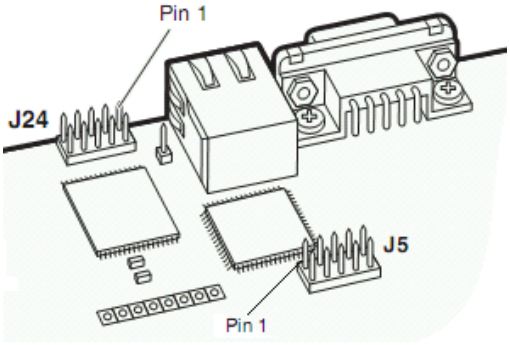


Figura 2-11 Conectores JTAG

t) Circuitos de reloj

La tarjeta de desarrollo Nios incluye un oscilador de 50 MHz (Y2) y una red de distribución de reloj punto a punto que maneja la FPGA (U60), el controlador de configuración de dispositivos (U3) y los pines de los conectores PROTO1 & PROTO2. Este oscilador puede ser cambiado o removido por el usuario, para manejar los circuitos de reloj usando el conector de reloj externo (J4), se debe retirar el oscilador Y2.

u) Circuito de Suministro de energía

La placa de desarrollo Nios funciona con un voltaje de entrada de 16.0V no regulado conectado a J26, circuitos en la tarjeta generan niveles de voltaje regulados de ± 12.0V, 5.0 V, 3,3 V, 2,5 V y 1,2. Para aplicaciones que requieren mayores niveles de tensión, por separado puede ser conectada una fuente de suministro de alimentación.

En la práctica, la mayoría de diseños de FPGA implementan alguna lógica adicional además del sistema procesador. Las FPGAs de Altera proporcionan la flexibilidad para agregar características y mejorar el rendimiento del sistema Nios II. Por el contrario, puede eliminar características innecesarias de procesador y periféricos para ajustar el diseño en un dispositivo más pequeño y de menor costo, debido a que los pines y recursos lógicos en estos dispositivos son programables, es posible realizar muchas configuraciones diferentes.

Capítulo III Diseño e Implementación del Sistema de Depuración

La Interfaz de Monitoreo y Depuración Remota para Nios II, ha sido desarrollada con el propósito de aportar una herramienta que apoye la elaboración de prácticas en laboratorio de una asignatura que trate acerca de los STR, permitiendo llevar a cabo, procesos de monitoreo y depuración de aplicaciones creadas para ser ejecutadas sobre la tarjeta de desarrollo de manera remota, de tal modo que facilite para el estudiante las experiencias educativas en laboratorio.

En este tercer capítulo se hace una descripción detallada del sistema propuesto como solución al problema. A partir de un diagrama general se muestra los diferentes componentes que toman parte en su funcionamiento, se profundiza en la estructura tanto a nivel de hardware y software utilizados en el proceso de construcción de dicha solución. Luego se describe la arquitectura del sistema, ejercicio fundamental que permite un entendimiento claro, grafico de cómo está construido el sistema; también se especifica el modelado del sistema; además se describe el objetivo y procedimiento de cada una de las 7 practicas que sirven de apoyo para un curso en STR.

Para ésta descripción se tuvo en cuenta cada uno de los módulos hardware que lo componen junto con las herramientas software ligadas a estos; se describe los sistemas encargados de la comunicación entre ellos; se detalla la herramienta de depuración desarrollada como también las tecnologías que se utilizaron en su desarrollo y se describe su funcionamiento. La parte final del capítulo está dedicada a mostrar un resumen de cada una de las prácticas planteadas con el fin de presentar el modelo propuesto de su composición, éstas han sido formuladas como apoyo al desarrollo de las experiencias académicas de aprendizaje sobre el uso del kit de desarrollo, caso de estudio, y puesta en práctica de algunos de los conceptos de STR.

3.1 Arquitectura del Sistema

La interfaz de depuración se desarrollo bajo una arquitectura Cliente/Servidor, ésta es un modelo que intenta proveer usabilidad, flexibilidad, interoperabilidad y escalabilidad en las comunicaciones. En este modelo, el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio), haciendo uso de una red LAN que hace posible la comunicación entre el equipo servidor y los clientes [65].

En la Figura 3-12 se presenta un diagrama modular del sistema, en el cual se muestra los diferentes bloques que lo componen. Después de ésta se hace una descripción detallada del funcionamiento de los elementos hardware, software e interfaces que permiten la comunicación entre estos.

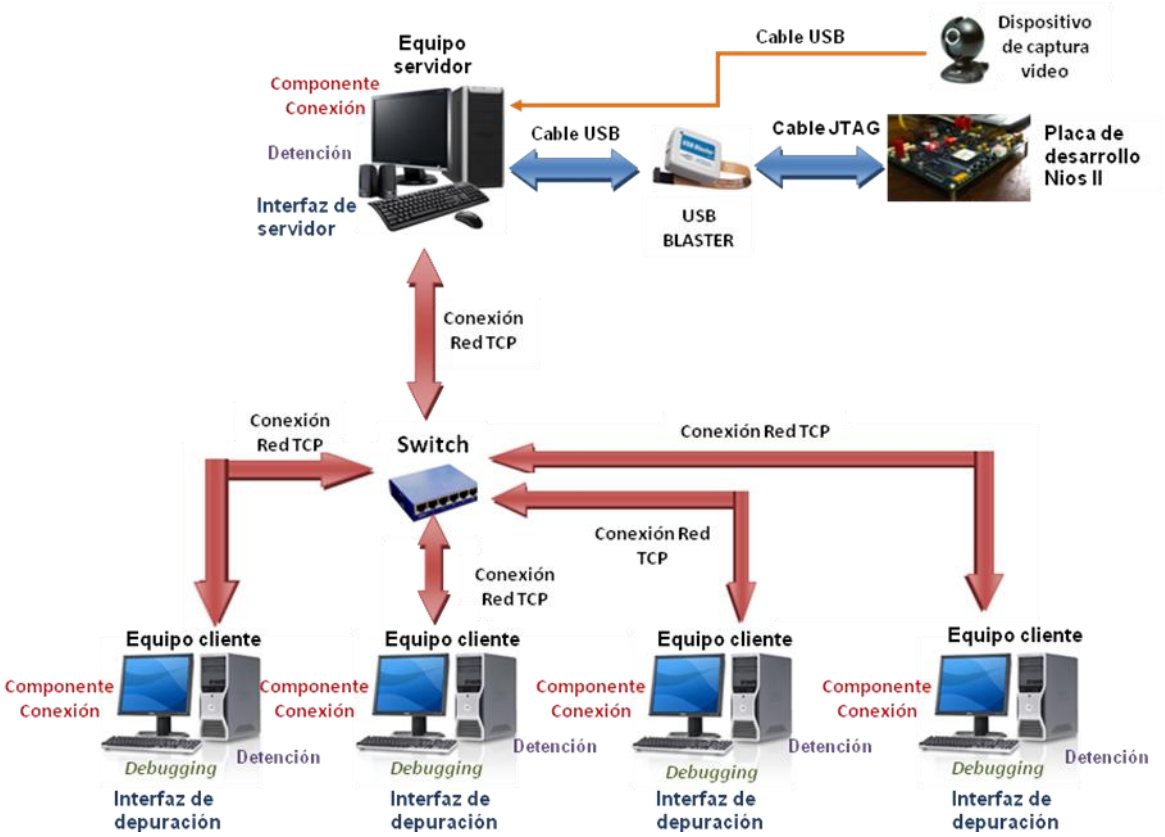


Figura 3-12 Diagrama modular del sistema

Como componentes hardware se tienen:

a) Placa de desarrollo Nios II

Parte fundamental dentro del sistema ya que se trata del kit de desarrollo Nios II edición Stratix II, equipo en propiedad de la FIET, cuya descripción detallada se efectuó en la sección 2.3 de éste documento.

b) Cable de programación USB BLASTER

El cable USB-Blaster Figura 3-13, es una interfaz de programación que se conecta desde un puerto USB del computador host hacia la FPGA de Altera montada sobre la placa de desarrollo provista para este proyecto. El cable envía los datos de configuración desde el computador a un conector estándar de 10 pines conectado a la FPGA. Puede utilizar el cable USB-Blaster iterativamente para descargar datos de configuración a un sistema durante la etapa de creación de prototipos o a los datos del programa en el sistema durante su producción. Los datos de configuración de programación se programan sobre la tarjeta con el programador de Quartus II [66].

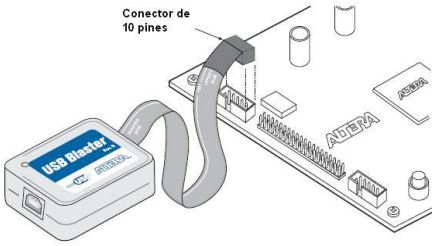


Figura 3-13 Interfaz de descarga USB-Blaster

El conector USB Blaster hace uso del estándar JTAG que es un protocolo serial, que se utiliza en emulación de circuitos y programación de memorias flash. Su estándar es el IEEE 1149.1: [67].

Este conector permite cargar los datos de programación de las aplicaciones dentro de la memoria flash de la tarjeta de desarrollo. Está conectado al modulo de depuración JTAG del la tarjeta, permitiendo ejecutar y detener las aplicaciones, establecer breakpoints[1], monitorear el estado de registros y memoria, entre otros.

c) Equipo Servidor

Es el encargado de llevar a cabo la configuración de la FPGA con el diseño hardware de la tarjeta, utilizando la herramienta de programación incluida en el software de desarrollo que proporciona el Kit de desarrollo Nios II [59].

Además sobre éste se ejecuta la aplicación que permite el acceso a los grupos de estudiantes que llevaran a cabo el monitoreo de cada una de las prácticas sobre la tarjeta de desarrollo, lo cual involucra establecer la conexión entre el grupo de estudiantes y el kit de desarrollo, cargar sobre éste el archivo ejecutable de la aplicación obtenida a partir de la compilación del código fuente desarrollado en laboratorio, e iniciar el envió de comandos de depuración recibidos desde un equipo cliente, al igual que las respuestas generadas tras la ejecución de cada uno de éstos.

Este equipo también realiza el manejo de la cámara Web encargada de capturar la señal de video que permite la visualización de las aplicaciones que están siendo ejecutadas sobre la tarjeta de desarrollo, para luego ser enviada al equipo cliente que está haciendo uso del kit de desarrollo y así facultar el monitoreo de modo remoto de las practicas en desarrollo.

d) Equipo Cliente

Sobre el Equipo Cliente los usuarios podrán llevar a cabo la creación de las aplicaciones, haciendo uso del software que acompaña al Kit de desarrollo caso de estudio, Nios II 7.2 IDE para compilar y generar el archivo ejecutable de la aplicación que permitirá llevar a cabo su depuración y prueba de funcionamiento sobre la tarjeta, el manual para esta operación se describe en el anexo D.

Por otra parte, sobre este equipo se ejecuta la herramienta de monitoreo y depuración, parte del desarrollo del presente proyecto, que permite tener acceso al servidor de

[1] *Breakpoint*: Es un punto de interrupción o parada usado en programación, especialmente en un proceso de depuración.

manera remota y así llevar a cabo el proceso de monitoreo y depuración el cual implica cargar el archivo ejecutable sobre la tarjeta, conectarse al servidor de depuración que da acceso a la placa de desarrollo, enviar comandos que efectúan diferentes operaciones con el propósito de ejecutar de manera controlada las aplicaciones que han sido implementadas con este fin, y recibir las respuestas generadas tras la ejecución de cada uno de estos comandos sobre la tarjeta, junto con la visualización de las respuestas producidas por los periféricos de salida proporcionados por el kit de desarrollo Nios II.

e) Cámara web

Dispositivo conectado al equipo servidor, encargado de capturar la señal de video que permite la visualización de las respuestas producidas por los diferentes periféricos de salida suministrados por la tarjeta de desarrollo, la cual es procesada por el servidor y así poder ser enviada al cliente que se encentre haciendo uso del recurso.

A nivel software se tienen los siguientes componentes:

a) Componente de conexión

Se necesita un componente de conexión que permita la comunicación entre el servidor y los clientes de tal modo que sea posible controlar la conexión con el recurso de manera confiable, a demás que se permita una sola conexión a la vez, y que sea posible la transmisión de video con una calidad adecuada y con una mínima presencia de retardos.

b) Componente de depuración (*Debugging*)

Se requiere de un componente que admita enviar comandos de depuración de manea remota; que permita el control de sus procesos, de tal modo que sea posible manejarlos desde una interfaz grafica; como también que éstos comandos puedan ser interpretados de manera transparente por la tarjeta de desarrollo Stratix II con procesador adaptable Nios II.

c) Sistema de depuración Remota

Éste sistema debe proporcionar las herramientas necesarias para llevar a cabo la depuración, de manera remota, de aplicaciones desarrolladas para ser programadas sobre la tarjeta de desarrollo, además debe suministrar la posibilidad multiplexar el control del recurso a un grupo de estudiantes, uno a la vez, atreves de un programa servidor que controle y administre el acceso al dispositivo según el orden de llegada de las peticiones de conexión permitiéndole el acceso al primero, y dejando en espera a los demás hasta que el recurso sea liberado, para que una vez más la primera petición que se haga en ese momento será atendida por el servidor.

Por otra parte, se utiliza el Protocolo de alto nivel TCP [68] el cual proporciona confiabilidad, control de flujo y recuperación de errores, garantizando una conexión segura entre los clientes y el servidor para poder llevar a cabo cualquier proceso de depuración como también asegurar que la transmisión del video, tomado sobre la tarjeta, sea entregado con éxito al estudiante que esté haciendo uso de ella.

d) Componente de detención (Detención)

Se requiere de un componente de este tipo en el desarrollo del sistema de monitoreo y depuración, ya que para efectuar una acción tan importante como la detención de la ejecución de una aplicación que corre sobre la tarjeta de desarrollo, requieren de un procedimiento que no termine esta ejecución definitivamente, sino que sea posible reanudarse a partir del punto en el que se detuvo para efectos de continuar con el proceso de depuración.

3.2 Modelado del sistema propuesto.

A continuación se presenta el proceso de modelado llevado a cabo para implementar el sistema propuesto. A través de los diagramas de casos de uso, de secuencia y de clases, de modo que se pueda visualizar de manera acertada la caracterización del aplicativo.

3.2.1 Diagrama de Casos de Uso

En la Figura 3-14 se presenta el diagrama de casos de uso en el equipo servidor.

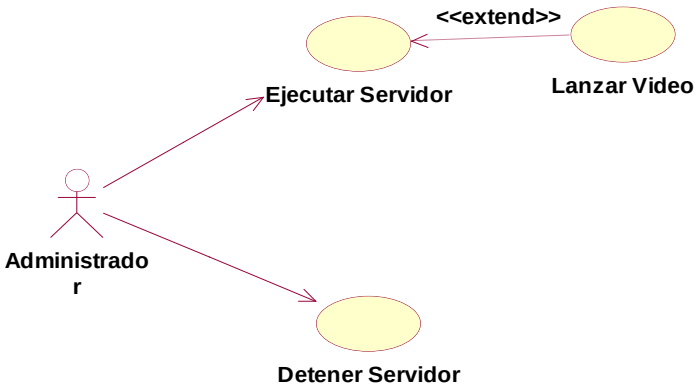


Figura 3-14 Diagrama de casos de uso servidor

A continuación se presenta la descripción de los casos de uso más relevantes del servidor.

Tabla 3 Caso de Uso: Ejecutar Servidor

Identificador	**Ejecutar Servidor**	
Descripción	Permite poner el servidor en escucha de peticiones	
Actor	Administrador	
Precondición	La tarjeta de desarrollo debe estar conectada y configurada	
Secuencia	**Paso**	**Acción**

Normal	**1**	El administrador debe poner en funcionamiento la interfaz del servidor
	2	El administrador debe dar clic en el botón <Iniciar Server>, para que el servidor inicie su funcionamiento.
Secuencia alternativa		Ninguna
Pos condición	El servidor empieza a escuchar peticiones de conexión.	

Tabla 4 Caso de Uso: Detener Servidor

Identificador	**Detener Servidor**	
Descripción	Permite detener la ejecución del servidor	
Actor	Administrador	
Precondición	El servidor tiene que estar en ejecución	
Secuencia Normal	**Paso**	**Acción**
	1	El administrador debe dar clic en el botón <Detener Server>, para que el servidor inicie su funcionamiento.
Secuencia alternativa		Ninguna
Pos condición	El servidor deja de escuchar peticiones de conexión.	

Tabla 5 Caso de Uso: Lanzar Video

Identificador	**Lanzar Video**	
Descripción	Permite iniciar la captura de video	
Actor	Sistema	
Precondición	El servidor tiene que estar en ejecución, y debe encontrase atendiendo un cliente	
Secuencia Normal	**Paso**	**Acción**
	1	El cliente carga el archivo ejecutable sobre la tarjeta de desarrollo
	2	El sistema inicia la captura de video, el video es visualizado en una ventana emergente en el servidor y el usuario remoto que se encuentra conectado.

Secuencia alternativa		Ninguna
Pos condición	El video puede ser detenido	

En la Figura 3-15 se presenta el diagrama de casos de uso del cliente.

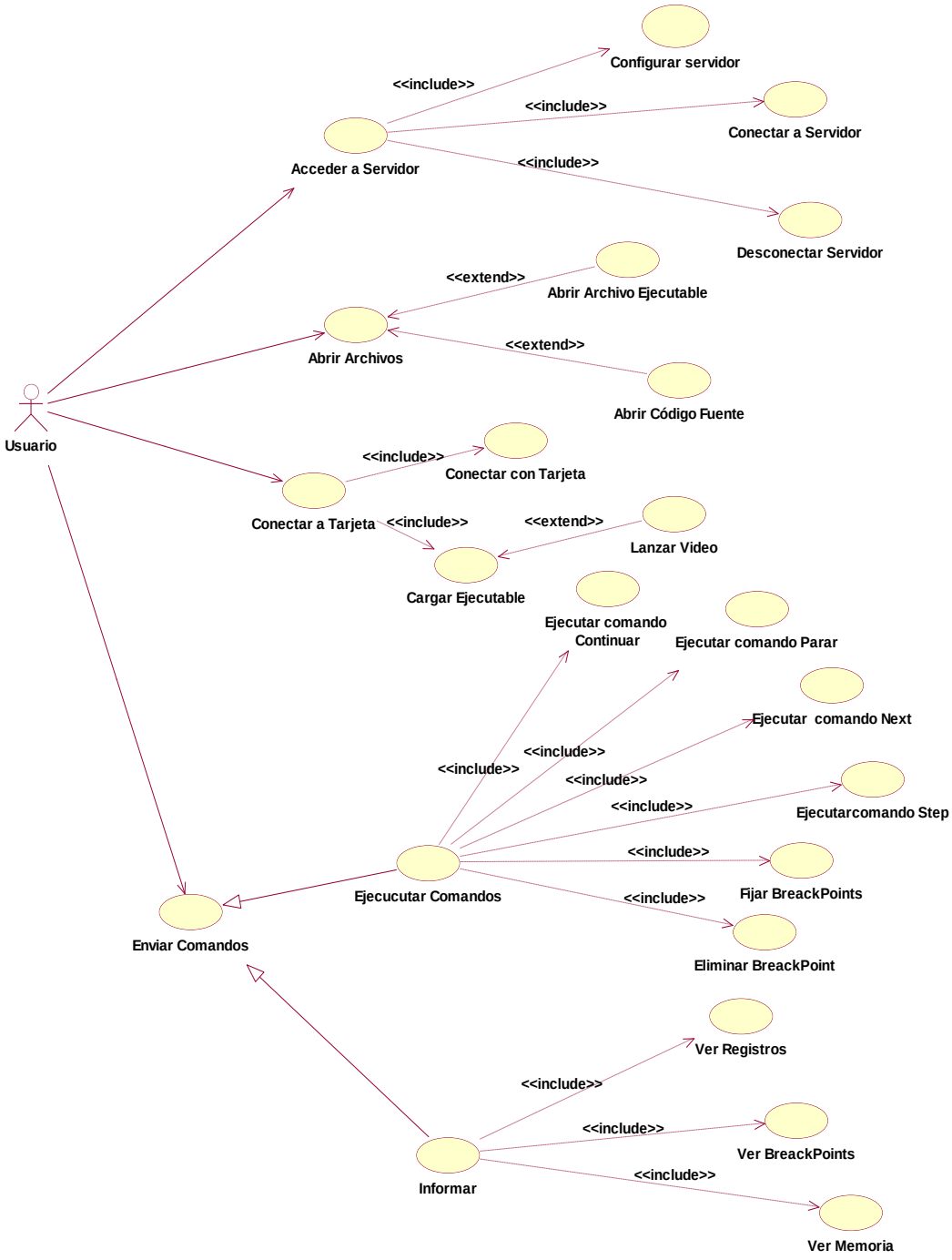


Figura 3-15 Diagrama de casos de uso

A continuación se presenta la descripción de los casos de uso más relevantes del cliente.

Tabla 6 Caso de Uso: Configurar servidor

Identificador	**Configurar el Servidor**	
Descripción	Permite configurar la dirección IP y el puerto del servidor	
Actor	Usuario	
Precondición	El servidor debe estar en funcionamiento	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe poner en funcionamiento la interfaz de depuración
	2	El usuario ingresa la dirección IP y el puerto del servidor al que se desea conectar en la ventana "Configuración Servidor" que se encuentra en la parte superior izquierda de la interfaz de la interfaz principal de usuario
	3	El usuario da clic en el botón <Aceptar>, para guardar los datos ingresados.
	4	El sistema activa la opción de conexión con el servidor
Secuencia alternativa		Ninguna
Pos condición	El usuario queda habilitado para conectarse al servidor	

Tabla 7 Caso de Uso: Conectar al Servidor

Identificador	**Conectar a Servidor**	
Descripción	Permite conectarse al servidor	
Actor	Usuario	
Precondición	El servidor que procesa las peticiones de conexión se encuentra disponible	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe presionar el botón <Conectar>
	2	El sistema indica al usuario si se realizo la conexión.
Excepción	**Paso**	**Acción**

	1	Si el servidor se encuentre ocupado, este le notifica al usuario. Con un mensaje de “servidor ocupado”
Pos condición	El usuario puede acceder al servicio que presta el servidor	

Tabla 3 Caso de Uso: Desconectar al Servidor

Identificador	**Desconectar del Servidor**	
Descripción	Permite desconectarse del servidor	
Actor	Usuario	
Precondición	El usuario debe estar conectado al servidor	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe presionar el botón de desconexión
	2	El sistema indica al usuario si se realizo la desconexión
Secuencia alternativa		Ninguna
Pos condición	El usuario termina la sesión	

Tabla 8 Caso de Uso: Abrir archivo ejecutable

Identificador	**Abrir Archivo Ejecutable**	
Descripción	Permite buscar el archivo que será cargado en la tarjeta de desarrollo	
Actor	Usuario	
Precondición	El usuario debe estar conectado al servidor	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe presionar el botón de <Abrir archivo ejecutable>, y buscarlo
	2	El usuario debe presionar el botón <Abrir>, el archivo ejecutable queda guardado
Secuencia alternativa		Ninguno
Pos condición	El usuario podrá cargar el código fuente de la aplicación	

Tabla 9 Caso de Uso: Abrir código fuente

Identificador	**Abrir Código Fuente**	
Descripción	Permite buscar el archivo de código fuente que será desplegado en la interfaz	
Actor	Usuario	
Precondición	El usuario debe haber seleccionado el archivo ejecutable	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe presionar el botón de <Abrir código fuente>, y proceder a buscarlo
	2	El usuario debe presionar el botón <Abrir>
	3	La interfaz muestra el código fuente cargado
Secuencia alternativa		Ninguno
Pos condición	El usuario puede conectare a la tarjeta de desarrollo	

Tabla 10 Caso de Uso: Conectar

Identificador	**Conectar a la tarjeta**	
Descripción	Permite conectarse a la tarjeta de desarrollo	
Actor	Usuario	
Precondición	El usuario debe haber cargado el archivo ejecutable y el código fuente	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe seleccionar la opción <Conectar a la tarjeta>
	2	El sistema indicara a través de un mensaje la conexión con la tarjeta de desarrollo
Excepción	**Paso**	**Acción**
	1	En el caso de que no se haya efectuado la conexión con la tarjeta de desarrollo, el sistema notificara al usuario con el mensaje “conexión fallida”
Pos condición	El usuario puede proceder a descargar el archivo ejecutable sobre la tarjeta de desarrollo	

Tabla 11 Caso de Uso: Cargar Ejecutable

Identificador	**Cargar Ejecutable**	
Descripción	Permite cargar el archivo a depurar sobre la tarjeta de desarrollo	
Actor	Usuario	
Precondición	El usuario debe haberse conectado a la tarjeta de desarrollo	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe seleccionar la opción <Cargar>
	2	El sistema muestra, con un mensaje, al usuario que el archivo ha sido cargado sobre la tarjeta
	3	El sistema indica al usuario si se realizo la conexión
	4	El sistema lanza la señal de video que es recibida por el usuario
Excepción	**Paso**	**Acción**
	1	En el caso de que el archivo ejecutable no se haya cargado correctamente, el sistema informara al usuario
Pos condición	El usuario puede ejecutar el archivo cargado sobre la tarjeta	

Tabla 12 Caso de Uso: Efectuar el comando Ejecutar

Identificador	**Efectuar el comando Ejecutar**	
Descripción	Permite iniciar la ejecución de la aplicación sobre la tarjeta	
Actor	Usuario	
Precondición	El archivo ejecutable debe estar correctamente cargado sobre la tarjeta.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe seleccionar la opción <Ejecutar>
	2	Se inicia la ejecución del archivo ejecutable sobre la tarjeta, sus resultados son visibles en video, y las salidas por consola con el mensaje "*Continuing*."
Secuencia alternativa		Ninguno

Pos condición	el usuario debe detener la ejecución para poder continuar el proceso de depuración

Tabla 13 Caso de Uso: Ejecutar el comando Parar

Identificador	**Ejecutar el comando Parar**	
Descripción	Permite parar la ejecución de la aplicación sobre la tarjeta	
Actor	Usuario	
Precondición	La aplicación cargada en la tarjeta debe estar en estado de ejecución	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe seleccionar la opción <Detener>
	2	Se detiene la ejecución del archivo ejecutable sobre la tarjeta, en el video se visualiza la detención, en consola aparecerá el mensaje "*Program received signal SIGTRAP, Trace/breakpoint trap. …*"
Secuencia alternativa		Ninguno
Pos condición	El usuario puede ejecutar diferentes comandos que permiten realizar la depuración y monitoreo	

Tabla 14 Caso de Uso: Ejecutar el comando *Next*

Identificador	**Ejecutar el comando *Next***	
Descripción	Permite ejecutar la aplicación sobre la tarjeta paso a paso	
Actor	Usuario	
Precondición	La aplicación en depuración debe estar detenida	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe seleccionar la opción <*Next*>
	2	La aplicación se ejecuta un paso a la vez sin entrar a las funciones, en consola aparece la línea que se ejecuta, en video se visualiza el cambio de estado de los periféricos.
Secuencia alternativa		Ninguno

Pos condición	el usuario puede ejecutar cualquier comando

Tabla 15 Caso de Uso: Ejecutar el comando *Step*

Identificador	**Ejecutar el comando *Step***	
Descripción	Permite ejecutar la aplicación sobre la tarjeta línea por línea	
Actor	Usuario	
Precondición	La aplicación en depuración debe estar detenida	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe seleccionar la opción <*Step*>
	2	La aplicación se ejecuta línea por línea, se visualiza como en el caso de uso anterior
Secuencia alternativa		Ninguno
Pos condición	el usuario puede ejecutar cualquier comando	

Tabla 16 Caso de Uso: Fijar *BreakPoint*

Identificador	**Fijar *BreakPoints***	
Descripción	Permite Fijar BreakPoints en cualquier línea del código de la aplicación	
Actor	Usuario	
Precondición	La aplicación en depuración debe estar detenida	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe seleccionar la opción < Fijar *BreakPoints* >
	2	El usuario debe introducir el numero de línea en el que desea fijar el *BreakPoint*
	3	El usuario debe presionar el botón <Aceptar> para que el *BreakPoint* quede establecido
Excepción	**Paso**	**Acción**

	1	En el caso de que el usuario ingrese un carácter no numérico, el sistema informará a través de un cuadro de texto el error cometido
	2	En el caso de que el usuario ingrese un numero de línea no existente en el código, el sistema mostrara un mensaje indicando el error cometido
Pos condición	el usuario puede ejecutar uno de los comandos	

Tabla 17 Caso de Uso: Eliminar *BreakPoint*

Identificador	**Eliminar BreakPoint**	
Descripción	Permite Eliminar un *BreakPoint* previamente establecido en cualquier línea del código de la aplicación	
Actor	Usuario	
Precondición	La aplicación en depuración debe estar detenida	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe seleccionar la opción < Eliminar *BreakPoints* >
	2	El usuario debe introducir el numero de línea del *BreakPoint* que desea eliminar
	3	El usuario debe presionar el botón <Aceptar> para que el *BreakPoint* quede selecciono sea eliminado
Secuencia alternativa	**Paso**	**Acción**
	1	En el caso de que el usuario ingrese un carácter no numérico, el sistema informara atabes de un cuadro de texto el error cometido
	2	En el caso de que el usuario ingrese un numero de línea en el que no ha sido establecido un *BreakPoint*, el sistema mostrara un mensaje indicando el error cometido
	3	En el caso de que el usuario ingrese un numero de línea no existente en el código, el sistema mostrara un mensaje indicando el error cometido
Pos condición	el usuario puede ejecutar cualquier comando	

Tabla 18 Caso de Uso: Ver Registros

Identificador	**Ver Registros**	
Descripción	Permite visualizar el estado de los registros	
Actor	Usuario	
Precondición	La aplicación en depuración debe estar detenida	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe seleccionar la opción <Ver Registros>
	2	La interfaz visualiza los estados de los registros del micro controlador
Secuencia alternativa		Ninguno
Pos condición	el usuario puede ejecutar cualquier comando	

Tabla 19 Caso de Uso: Ver *BreakPonts*

Identificador	**Ver *BreakPoint***	
Descripción	Permite visualizar la ubicación de los *BreakPoint* establecidos	
Actor	Usuario	
Precondición	La aplicación en depuración debe estar detenida	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe seleccionar la opción <Ver *BreakPoints*>
	2	La interfaz visualiza la ubicación de los *BreakPoint* establecidos con anterioridad
Secuencia alternativa		Ninguno
Pos condición	el usuario puede ejecutar cualquier comando	

Tabla 20 Caso de Uso: Ver Memoria

Identificador	**Ver estado de Memoria**
Descripción	Permite visualizar el estado de la memoria

Actor	Usuario	
Precondición	La aplicación en depuración debe estar detenida	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario debe seleccionar la opción <Ver Memoria>
	2	La interfaz visualiza los estados de la memoria
Secuencia alternativa		Ninguno
Pos condición	el usuario puede ejecutar cualquier comando	

3.2.3 Diagrama de Secuencia

A continuación se presenta un diagrama de secuencia, que representa el funcionamiento general del sistema, Figura 3-16Figura 3-16 Diagrama de secuencia, que representan los casos de uso fundamentales en el desarrollo de la solución del presente proyecto.

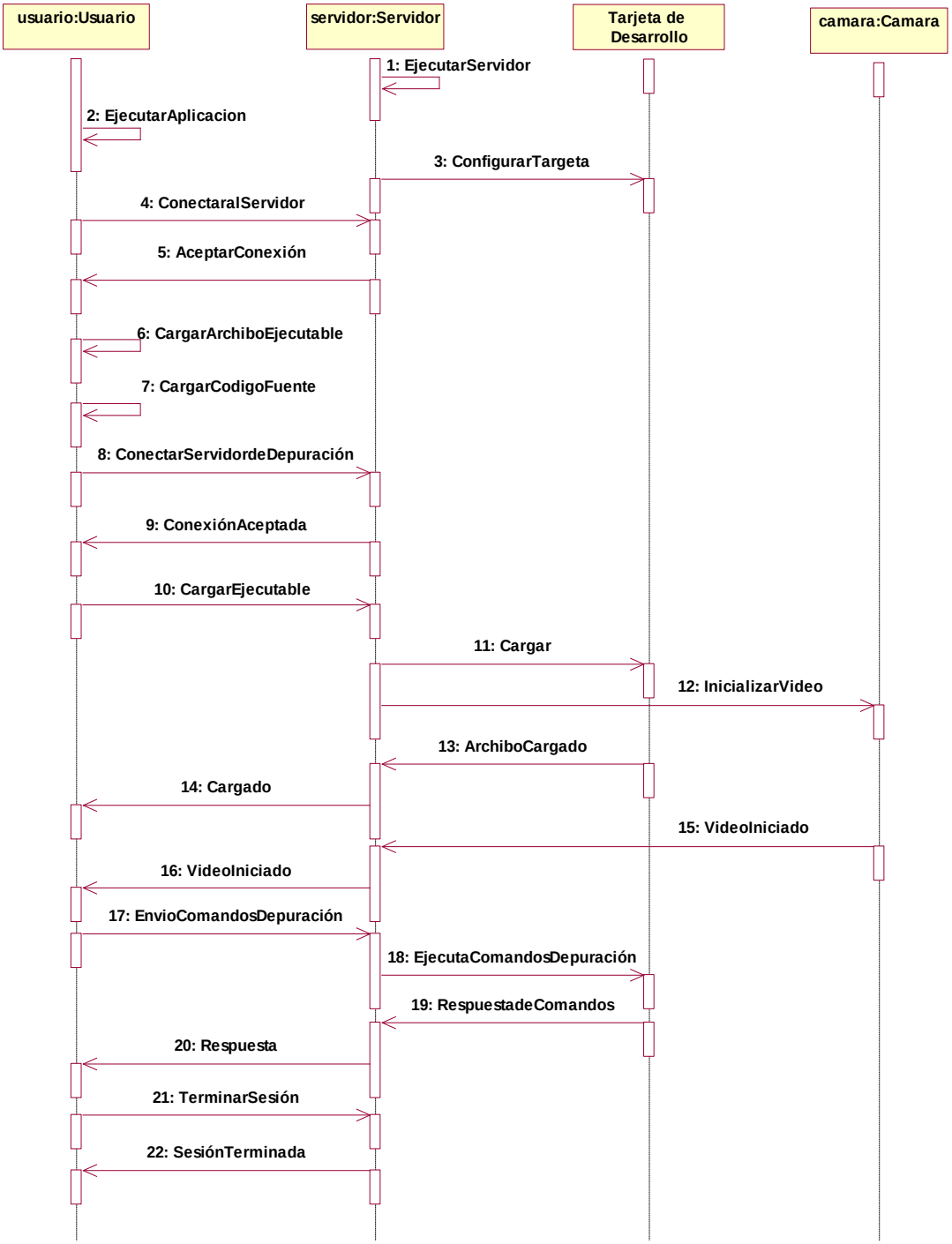


Figura 3-16 Diagrama de secuencia

3.2.4 Diagrama de Clases

En la Figura 3-17, se presenta el diagrama de Clases correspondiente al diseño del servidor.

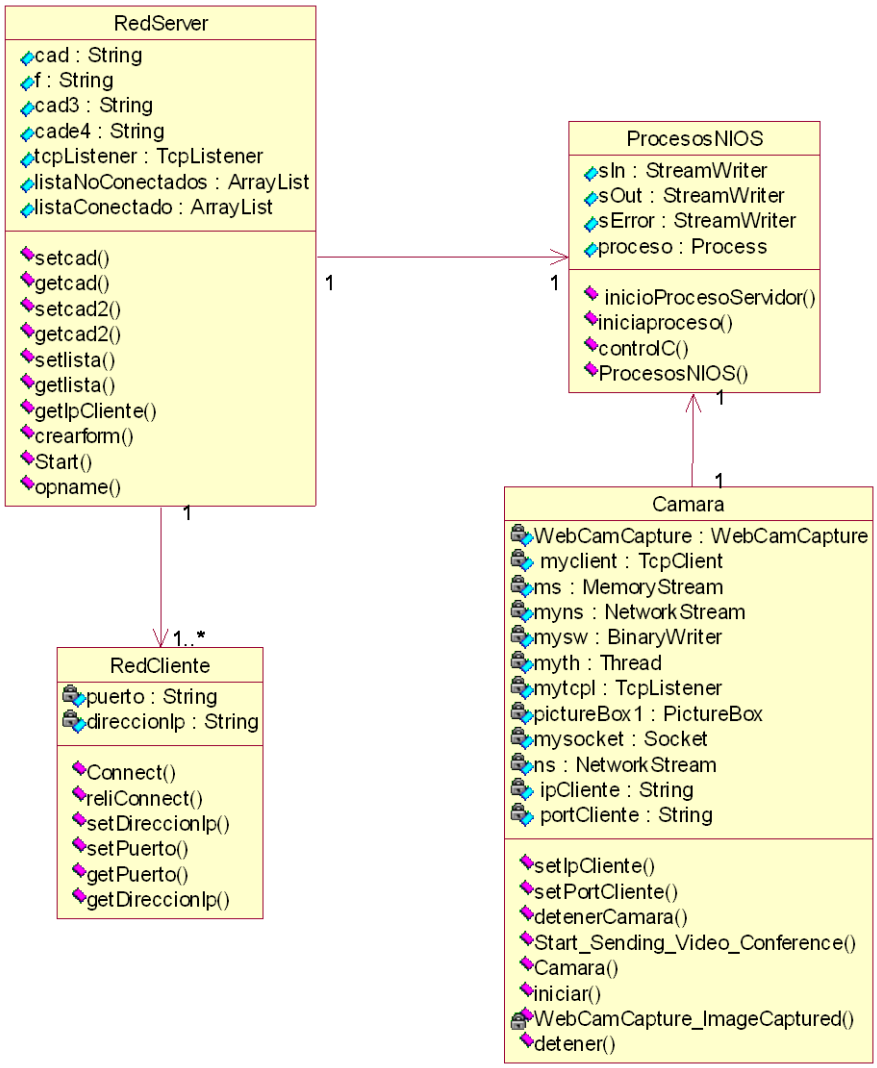


Figura 3-17 Diagrama de Clases Servidor

En la Figura 3-18, se presenta el diagrama de Clases correspondiente al diseño del equipo cliente.

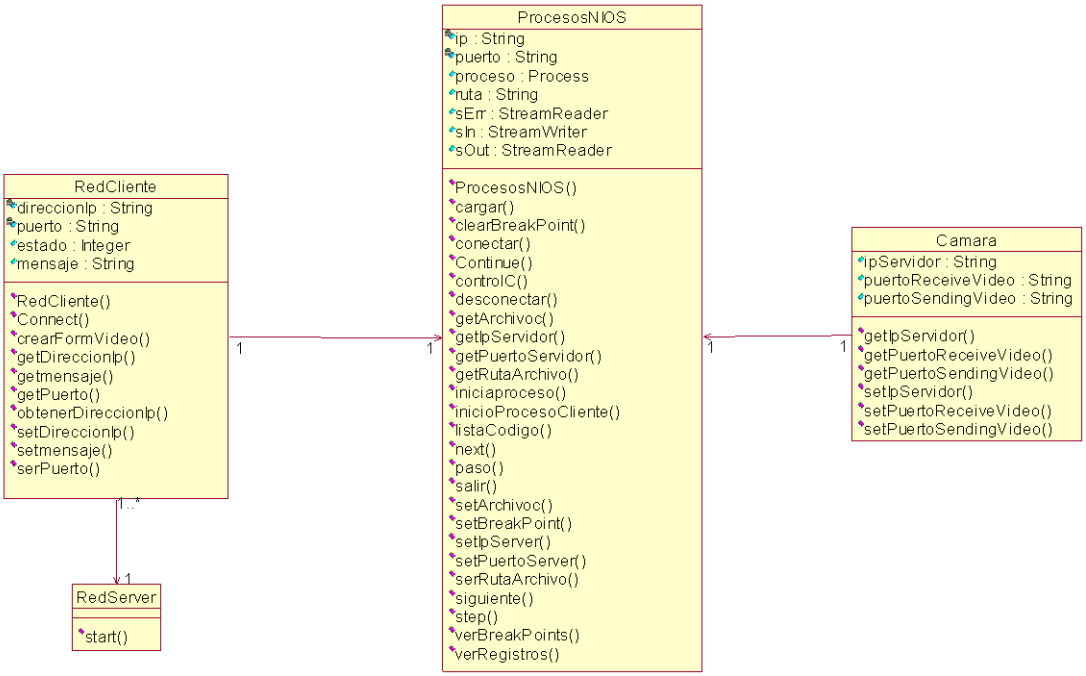


Figura 3-18 Diagrama de Clases Cliente

3.3 Practicas sobre Sistemas Operativos de Tiempo Real

En ésta sección se expondrán las prácticas desarrolladas sobre los Sistemas Operativos de Tiempo Real elegidos para ser implementados utilizando el kit de desarrollo Nios II. Dichas prácticas tienen como finalidad acercar a los estudiantes a los conceptos involucrados en el manejo de Sistemas Tiempo Real implementados sobre un Sistema empotrado; es así como se trataran temas como la administración de tareas, el manejo de tiempos, comunicación y sincronización entre procesos y planificación de tareas.

Para desarrollar esta sección se presenta los nombres, el objetivo y el procedimiento de cada una de las 7 practicas propuestas como apoyo en el desarrollo de un curso de STR, estas prácticas son detalladas en el Anexo A, en donde se incluye el código desarrollado para los dos SOTR seleccionados.

3.3.1 PRÁCTICA No. 1: GUÍA DE AMBIENTACIÓN A LAS HERRAMIENTAS DE TRABAJO

OBJETIVO:

- Lograr que el estudiante se familiarice con la interfaz de monitoreo y depuración remota para Nios II con el que se va a trabajar.

- Conocer el proceso de creación y compilación de un proyecto en el entorno de desarrollo Nios II 7.2 IDE.

- Efectuar el proceso de configuración de las características de trabajo del kit de desarrollo Stratix II con procesador adaptable Nios II.

DESCRIPCIÓN:

En esta práctica se tratarán los aspectos básicos de la edición, y compilación de códigos en el entorno de desarrollo Nios II 7.2 IDE por medio de un ejemplo, en el cual se busca crear una variable que se incrementa y muestra los cambios de los estados en LEDs. Además, se debe llevar a cabo la configuración de trabajo de la tarjeta de desarrollo utilizando el software de programación Quartus II; para terminar, se lleva a cabo una sesión de monitoreo sobre la tarjeta de desarrollo utilizando la interfaz de monitoreo depuración para Nios II.

PROCEDIMIENTO:

Para crear o editar un archivo .c, vamos a utilizar el entorno de desarrollo Nios II 7.2 IDE, que nos permite crear, editar, y compilar los programas que desarrollaremos durante el curso. Iniciaremos creando un proyecto que llamaremos "ContadorBinario" y un archivo .c con el nombre de "contbin.c", como lo indica la primera sección del manual de éste IDE; luego tomaremos como ejemplo el siguiente sencillo código, que debe ser copiado en el archivo que creamos:

3.3.2 PRÁCTICA No. 2: MANEJO DE PERIFERICOS DEL KIT DE DESARROLLO STRATIX II PARA NIOS II

OBJETIVO:

- Conocer el funcionamiento de los periféricos de salida y entrada con los que cuenta el kit de desarrollo Stratix II con procesador adaptable Nios II.

PROCEDIMIENTO:

En la práctica se busca experimentar de manera directa con el funcionamiento de los periféricos tanto de entrada como de salida, disponibles en la tarjeta de desarrollo, para lo cual se presenta como apoyo un ejemplo en el que se puede visualizar el manejo y funcionamiento de éstos periféricos.
Los estudiantes deben implementar un contador de 8 bits el cual debe mostrar su valor en los 8 LED y los *display* de 7 segmentos, y aplicar a este valor una función matemática (seno, logaritmo, raíz cuadrada, etc.) diferente de acuerdo a cada pulsador que se presione, y mostrar el resultado de estas operaciones en el LCD y en el *Hyper Terminal* utilizando el puerto serial.

3.3.3 PRÁCTICA No. 3: ADMINISTRACIÓN DE TAREAS

OBJETIVO:

- Poner en práctica los conceptos relacionados con la creación de tareas y su administración entendida como pausar, reanudar, eliminar, manejo de prioridades y ejecución de tareas.

- Profundizar los conocimientos teóricos de funcionamiento de tareas implementadas en un Sistema de Tiempo Real y su aplicabilidad en la ejecución de procesos.

PROCEDIMIENTO:

A continuación se presenta un ejemplo en el que se implementan tres tareas, se les asigna una prioridad a cada una, posteriormente se ejecutan, y se eliminan. En un segundo ejemplo, las tareas creadas se suspenden y reinician.

En el siguiente código de creación y eliminación de tareas, además se presenta el manejo de los periféricos LEDs, Display de 7 segmentos y LCD. En la primera parte, se presenta la codificación en el SOTR Micro C/OS II:

3.3.4 PRÁCTICA No. 4*:* SINCRONIZACIÓN Y COMUNICACIÓN DE TAREAS

OBJETIVO:

- Poner en práctica los conceptos relacionados con la sincronización y comunicación de tareas aquí utilizaremos los conceptos de creación y manejo de colas, mutex y semáforos.

- Profundizar los conocimientos teóricos de funcionamiento de tareas implementadas en un Sistema de Tiempo Real y su aplicabilidad en la ejecución de procesos utilizando los SOTR FreeRTOS, y Micro C/OS II.

PROCEDIMIENTO:

Después de haber estudiado los conceptos relacionados con la comunicación y sincronización de tareas, y utilizando como apoyo los siguientes ejemplos crear dos tareas que generen números aleatorios los cuales se escriben en una cola, garantizando el acceso a esta por parte de cada tarea utilizando semáforos o MUTEX, y una tercer tarea que obtenga los datos de la cola y los imprima en el LCD utilizando cada uno de los SOTR: FreeRTOS y Micro C/OS II.

A continuación se muestra un ejemplo de creación y utilización de semáforos con 3 tareas deferentes, y se despliega el resultado en los periféricos de la tarjeta de desarrollo.

En la primera parte, se presenta la codificación de un ejemplo en el cual se muestra la creación de un semáforos el cual controla el acceso a un recurso, en este caso el LCD con el que cuenta la tarjeta de desarrollo, utilizando el SOTR Micro C/OS II:

3.3.5 PRÁCTICA No. 5: ADMINISTRACIÓN DE TIEMPO

OBJETIVO:

- Afianzar los conocimientos de los conceptos adquiridos en el estudio de la administración de tiempo que están disponibles para su uso en sistemas operativos de tiempo real.

- Iniciar al estudiante para la implementación de un Sistema de Tiempo Real donde sea necesario el uso de los conceptos estudiados.

PROCEDIMIENTO:

Después de realizar un estudio de los conceptos que definen las características del manejo de tiempo en los STR. Utilizando el código ejemplo crear 4 tareas en las que se utilicen los diferentes métodos para bloquear las tareas, concluir cuales son las diferencias entre estas funciones utilizadas para este fin.

3.3.6 PRÁCTICA NO.6: EJECUTIVO CICLICO

OBJETIVO:

- Verificar la apropiación y correcto uso de los conceptos de ejecutivos cíclicos y planificación estática de tareas.
- Comprobar el manejo de las condiciones para planificar un conjunto de tareas utilizando un ejecutivo cíclico.
- Conocer el nivel de comprensión de las clases teóricas y su aplicabilidad.

PROCEDIMIENTO:

Tomando como base el ejemplo de creación de un ejecutivo cíclico presentado a continuación, se debe crear uno para 3 tareas. Estas tareas deben imprimir un mensaje con su activación y desactivación para verificar su ejecución, y verificar que se cumplan los tiempos establecidos en el diseño.

A continuación se presenta un ejemplo donde se crea un ejecutivo cíclico que planifica la ejecución de 4 tareas.

3.3.7 PRÁCTICA No. 7: PRÁCTICA FINAL

OBJETIVO:

- Poner en práctica los conceptos y experiencias obtenidas a lo largo del desarrollo de las guías realizadas en el curso.

DESCRIPCIÓN:

En esta práctica se busca realizar un ejercicio que incluya algunos de los conceptos que fueron objeto de experimentación durante el desarrollo del curso con el fin de afianzar estos conocimientos y fortalecer las habilidades del estudiante para iniciar nuevos desarrollos en los que se trabaje con Sistemas de Tiempo Real.

PROCEDIMIENTO:

De acuerdo a lo aprendido durante el desarrollo de las prácticas, proponer e implementar un proyecto donde se utilice uno de los SOTR estudiados en el curso, haciendo uso de los periféricos de entrada y salida disponibles en el kit de desarrollo Stratix II con procesador adaptable Nios II.

Capítulo IV Prototipo y Pruebas

El objetivo del sistema de Monitoreo y Depuración Remoto implementado para Nios II, Fue ayudar en el trabajo educativo en laboratorio en la FIET, donde se estudie y experimente sobre Sistemas de Tiempo Real; de tal modo que se pueda optimizar el proceso de monitoreo y depuración, debido a que ya no existiría la necesidad de hacerlo utilizando una gran cantidad de instrucciones para ejecutar los comandos sobre una consola, por lo que desarrollar las guías de practica propuestas para ser programadas sobre la tarjeta de desarrollo Stratix II con procesador adaptable Nios II, se convierta en una experiencia formativa de gran utilidad. Además, el aplicativo permite el acceso al recurso hardware a un grupo de estudiantes en la sala donde se lleva a cabo el laboratorio ya que el acceso a la tarjeta se efectúa si se encuentra conectado a una red LAN.

En este capítulo se presenta el proceso de validación del sistema de Monitoreo y Depuración Remoto para Nios II implementado. El capitulo inicia con una reseña de los instrumentos software utilizados para lograr la puesta en funcionamiento de la herramienta justificando su utilización. Continúa con una reseña de los principales inconvenientes que se debieron sortear durante todo el proceso de desarrollo del presente trabajo de grado; junto con la solución que puso fin a estos problemas. A continuación se realiza la descripción del procedimiento de evaluación aplicado, así como el resumen de cada una de las pruebas efectuadas para verificar el funcionamiento y la eficacia del aplicativo implementado, divididas en tres secciones, las pruebas hechas sobre el funcionamiento de la tarjeta de desarrollo, las que se realizaron sobre el funcionamiento de la herramienta pero en un trabajo por módulos, y se finaliza con la presentación de la prueba realizada por un grupo de alumnos de laboratorio II de Sistemas Digitales en donde se evalúa la experiencia con el uso de la interfaz y su percepción de su usabilidad.

4.1 Herramientas Software

A continuación se presenta un descripción de las herramientas software que fueron utilizadas en el desarrollo del sistema de monitoreo y depuración haciendo posible aportar una solución satisfactoria, uno de los objetivos específicos del presente proyecto.

En la Figura 4-19 se presenta un diagrama modular en el que se muestra los componentes software que fueron utilizados en el desarrollo del presente trabajo de grado

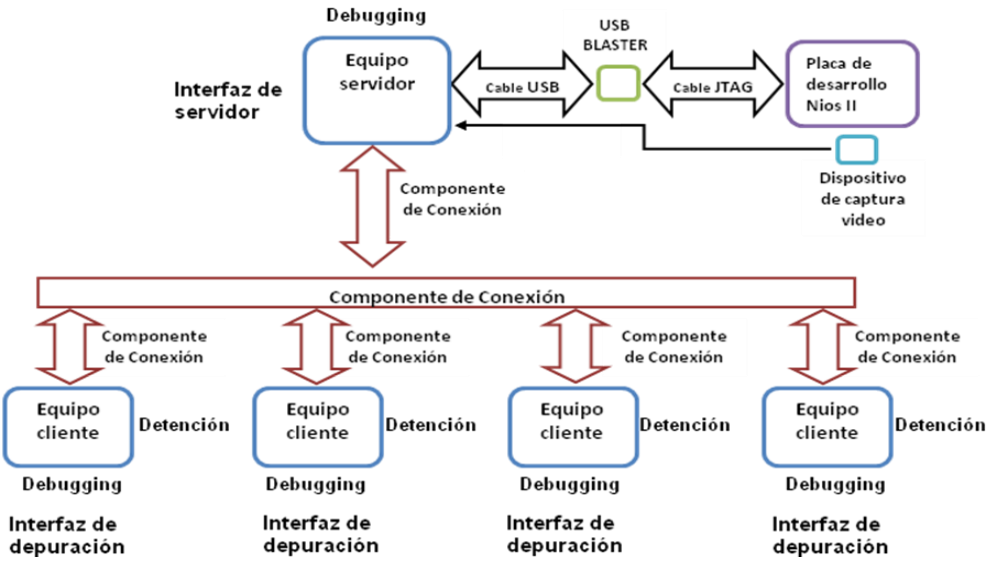


Figura 4-19 Diagrama modular de herramientas software

a) Componente de conexión

Para desarrollar este componente se optó por utilizar los Sockets, debido a que presentaban las características que se requerían para llevar a cabo la comunicación entre los clientes y el servidor; esto es, los Socket permiten el manejo de usuarios de manera eficiente y confiable, como también es posible a través de ellos la transmisión de video de manera que no se presenten retardos considerables, y además en la FIET se cuenta con las un manejo de redes basadas sobre el protocolo de comunicación TCP/IP, sistema sobre el cual trabajan los Sockets utilizados en el desarrollo del presente proyecto.

Los sockets son mecanismos de comunicación entre programas a través de una red TCP/IP, estos mecanismos pueden tener lugar dentro de la misma máquina o utilizando una red, se usan en arquitectura cliente – servidor. Desde el punto de vista del servidor, éstos llevan asociado un puerto, y de lado del cliente se requiere la dirección del servidor y un número de puerto. Una vez establecida la conexión es posible intercambiar datos utilizando flujos. Los sockets son puntos finales de enlaces de comunicaciones entre procesos.

Para el establecimiento de la comunicación entre el cliente y el servidor se utilizan socket TCP los cuales proporcionan un servicio orientado a conexión, donde los datos son transferidos sin encuadrarlos en registros o bloques y el socket servidor atiende peticiones de conexión y el cliente efectúa solicitudes. En caso de pérdida de la transferencia de información entre procesos estos son notificados de tal suceso para que tomen las medidas oportunas como la retransmisión de los datos entre otras [69].

b) Componente de depuración (*Debugging*)

Para la construcción de este componente fue utilizado el *Gdb Debugger*, como parte del desarrollo de la herramienta de depuración debido a que éste es una funcionalidad que se encuentra incorporada en el paquete de software licenciado incluido en el kit de desarrollo, por tanto, se convirtió en uno de los objetivos del proyecto el generar la interfaz

grafica que permitiera llevar a cabo los procesos de monitoreo y depuración de forma intuitiva, amigable interactiva evitando el uso de la ejecución de una serie de comandos sobre consola.

El *Gdb Debugger* es una herramienta de depuración desarrollado por la FSF (*Free Software Foundation*) que se usa principalmente en Linux y bajo licencia GPL (GNU *Public Licence*). Es un depurador portable que se usa en varias plataformas Linux y funciona para varios lenguajes de programación como C, C++, Fortran. Además, ofrece la posibilidad de trazar y modificar la ejecución de un programa, permitiendo al usuario controlar y alterar los valores de las variables internas del programa, GDB no contiene su interfaz gráfica de usuario, por lo que se controla con una interfaz de comandos [70], entre sus características más importantes se encuentran:

- *Debugging* de programas complejos con múltiples archivos.
- Capacidad para detener el programa o ejecutar un comando en un punto específico, según una condición o al llegar una señal.
- Capacidad para mostrar valores de expresiones cuando el programa se detiene automáticamente,
- Examinar la memoria y/o variables de diversas formas y tipos, incluyendo estructuras, arreglos y objetos.
- Cambiar los valores de las variables para estudiar el comportamiento del programa sin necesidad de recompilar.
- *Debugging* a programas en ejecución,
- *Debugging* a programas que han finalizado.

Esta herramienta ofrece un modo remoto para la depuración de aplicaciones de sistemas empotrados, utilizando un protocolo serial especifico del GDB, como también utilizando el protocolo TCP/IP, y soportando gran variedad de procesadores como ARM, X86, Motorola, Nios II, SPARC, entre otros [71] [72].

El Gdb hace uso de un protocolo remoto serial, en el que todos los comandos y sus respuestas se envían como un paquete, cada paquete es introducido con el símbolo '*$*', los paquetes de datos terminan con el caracter '#', seguido por una suma de comprobación de dos dígitos (*$packet-data#checksum)*.

La suma de comprobación de dos dígitos se calcula como el módulo suma 256de todos los caracteres entre el primero '$' y el final '#' (un checksum sin signo de 8 bits). Cuando el host o el equipo de destino recibe un paquete, la primera respuesta recibido es un acuse de recibo: ya sea '+' (para indicar que el paquete se recibió correctamente) o '-' (a la retransmisión de la solicitud) [73].

c) Herramientas de desarrollo Software

En el proceso de desarrollo del sistema de Monitoreo y Depuración Remota para Nios II, se opto por elegir entre los dos lenguajes de programación más conocidos y trabajados en la FIET, C Sharp y Java. A partir de las cuales se eligió la que se acercaba mejor a las necesidades del proyecto.

En la siguiente Tabla 21, se presenta una comparación en la que se incluyen algunas características de los lenguajes de C# y Java, vistos como lenguajes de programación

orientados a objetos, para lo cual se partirá de ver la forma en la cual cada uno de ellos implementa las características de la Programación Orientada a Objetos (POO) [74].

Tabla 21 Tabla comparativa entre las herramientas de desarrollo

Característica	**C#**	**Java**
Tipos de estructuras que el lenguaje le permite crear a un programador	Clases, interfaces, *struct* y *enum*.	Clases e interfaces.
Elementos que pueden definirse dentro de una clase	Atributos, métodos, clases internas, propiedades, eventos y *delegates*.	Atributos, métodos y clases internas.
Niveles de encapsulamiento	*public, private, internal, protected* y la combinación de estos dos últimos. En caso de omisión se asume *private*.	*public, private, protected* y visibilidad de paquete, este último se asume cuando se omite.
Herencia	No se permite la herencia múltiple, se puede simular a través del uso de interfaces. Por omisión se hereda de *Object.*	No se permite la herencia múltiple, se puede simular a través del uso de interfaces. Por omisión se hereda de *Object*.
Polimorfismo	Se permite que una clase sobrecargue o sobre escriba métodos definidos por su clase padre, a menos que la clase padre lo impida empleando la palabra reservada *sealed*, en el encabezado del método. Si un objeto de una clase hija es referenciado a través de una referencia a su clase padre, su comportamiento, al invocar un método sobre escrito, dependerá de los permisos establecidos por la clase padre, y de la decisión tomada por quien definió la clase. Por omisión se comportará como lo definió la clase padre.	Se permite que una clase sobrecargue o sobre escriba métodos definidos por su clase padre, a menos que la clase padre lo impida mediante la palabra reservada *final,* en el encabezado del método. Si un objeto de una clase hija es referenciado a través de una referencia a su clase padre, su comportamiento, al invocar un método sobre escrito, será el que definió la clase a la cual él pertenece.
Sobrecarga de operadores para una clase:	No permite la sobrecarga de ninguno de los operadores básicos.	Se permite la sobrecarga de algunos de los operadores básicos: Unitarios: **+**, **-**, **~, !**, **++**, **--**, **true**, **false** Binarios: **+**, **-**, *****, **/**, **%**, **&**, **|**, **^**, **<<**, **>>**, **==**, **!=** , **>**, **<**, **>=**, **<=**

Teniendo en cuenta estas características que influyen directamente en la construcción de la herramienta, se optó por trabajar sobre C Sharp, además el manejo de procesos sobre el sistema operativo Windows en más transparente, permitiendo enviar datos o instrucciones, capturar sus respuestas, detenerlo y reiniciarlo de manera controlada sin perder la conexión; que se convierte en una característica fundamental que permitió el desarrollo del aplicativo.

C Sharp (C#) es un lenguaje de programación orientado a objetos [75] [76] desarrollado y estandarizado por Microsoft como parte de su plataforma .NET [77], el cual ha sido aprobado como un estándar por la ECMA [78] e ISO [79].

d) Componente de detención (Detención)

Se utilizó el lenguaje de programación C para el desarrollo de este componente, ya que para la ejecución de un comando fundamental en el aplicativo "*detene*r", requería del envió de una señal que indicara la interrupción en la ejecución de las aplicaciones sobre la tarjeta de desarrollo, para conseguir ésta señal se creó un ejecutable codificado en este lenguaje que la genera, posteriormente, cuando se requería de ejecutar este comando, se realiza una llamada e interpretada desde la herramienta de monitoreo.

C [80], es un lenguaje orientado a la implementación de Sistemas Operativos, software de software de sistemas y en la creación de aplicaciones. Su estandarización surjo en dos etapas la primera fue por ANSI, y posteriormente en 1990, fue ratificado como estándar ISO [79], es apreciado por la eficiencia del código que produce, si los programas creados lo siguen, el código creado en C es portátil entre plataformas y/o arquitecturas. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, con fuertes características que permiten construcciones de lenguaje con un control a muy bajo nivel [81].

e) Entorno de desarrollo

En el proceso de elaboración del sistema de depuración remota, se utilizo el entorno de desarrollo Visual Estudio 2008, ya que ofrece todas las herramientas para desarrollar aplicaciones tipo Windows, a demás éste es el entorno recomendado por los desarrolladores del lenguaje de programación C Sharp (C#), ya que también son ellos los constructores de éste.

4.2 Problemas encontrados en el desarrollo del sistema

En el desarrollo del sistema se afrontaron varios retos entre los que se encuentran:

- Presencia de retardos considerables en la señal de video recibida en el cliente, con respecto a la señal transmitida por el servidor.

  Este problema se solucionó disminuyendo el tamaño de la imagen tanto como lo permitiera la funcionalidad de este componente, además se debió utilizar un formato de video de menor calidad, de modo que se necesitara transmitir la menor cantidad posible de información.

- El no reconocimiento del comando que detiene la ejecución de la práctica sobre la kit de desarrollo.

  Se solucionó, mediante la implementación de un aplicativo externo que permitía que enviarle la instrucción de detención de modo que fue interpretado correctamente por la tarjeta de desarrollo.

- No era posible controlar el envío comandos desde el equipo cliente hacia la aplicación corriendo en la tarjeta de desarrollo utilizando JAVA.

Se debió desarrollar el aplicativo en el lenguaje de desarrollo C#, el cual si permitió el control a voluntad la ejecución de comandos de forma independiente.

- Presencia de nuevos retardos en la recepción de video cuando se efectuaba la petición de conexión por más de un usuario.

  Se decidió aislar la red en la que estaban conectados el equipo servidor y el cliente, con el fin de evitar el tráfico externo manifestado por la presencia de múltiples equipos haciendo uso de la red de Internet.

En la sección 4.3, de pruebas, se presenta las estrategias empleadas para mitigar los problemas presentados con mayor detalle.

4.3 Procedimiento de Evaluación del Prototipo y Pruebas

En ésta sección se describe el procedimiento llevado a cabo en la evaluación del aplicativo, así como las pruebas efectuadas con el fin de constatar el funcionamiento de la Sistema de Monitoreo y Depuración Remota para Nios II, pruebas llevadas a cabo a lo largo del desarrollo proyecto, en búsqueda de un apropiado funcionamiento de la herramienta aquí generada, al término de construcción del presente trabajo de grado.

Para llevar a cabo el proceso de evaluación, se dividieron las pruebas en tres aspectos diferentes: primera sección, utilizando el GDB Debugger se realizaron test del funcionamiento básico; en la segunda parte las pruebas se llevaron a cabo sobre funcionalidades de la interfaz de depuración; y en la sección final se efectuaron pruebas sobre el funcionamiento de la interfaz en red especificadas de la siguiente manera:

- En la primera parte se evaluó el funcionamiento del kit de desarrollo Stratix con procesador adaptable Nios II, utilizando la consola de comandos proporcionada por el GDB Debugger como herramienta para la depuración sobre la tarjeta. En esta sección se efectuaron pruebas de funcionamiento de la placa de desarrollo, de conectividad, de manejo de sus periféricos de salida, y adaptación de los SOTR: FreeRTOS y Micro C/OS-II sobre ella.

- En la segunda sección, las prácticas se efectuaron sobre prototipos no terminados de la interfaz, pero que sin embargo, ya permitían testear el desempeño de su conectividad cliente/servidor, la efectividad en la ejecución de los comandos de depuración y de la transmisión de video entre estos dos módulos.

- En la etapa final, se efectúa la evaluación de desempeño del las funcionalidades del aplicativo y todo el proceso que se lleva a cabo en una sesión de monitoreo, y duración, efectuando la conexión entre un único equipo cliente, a través del servidor, y la tarjeta de desarrollo; se procede a continuación con una prueba similar a la anterior, pero ahora utilizando cuatro equipos cliente, probando, el encolamiento de las peticiones de conexión y la notificación de la liberación del recurso (tarjeta de desarrollo), de modo que los equipos encolados puedan competir por la conexión con la placa.

  Se finaliza con una prueba en la que un grupo de estudiantes de la asignatura Circuitos Digitales II evalúan el desempeño de la herramienta poniéndola en

funcionamiento desarrollando la practica No. 1 del anexo A, en la que se puede visualizar claramente el funcionamiento de la tarjeta de desarrollo.

4.3.1 Pruebas sobre la tarjeta de desarrollo Nios II

Las siguientes pruebas se desarrollaron, como se especificó en la sección anterior, con el fin de verificar el correcto funcionamiento de la tarjeta de desarrollo Nios II.

- **Prueba 1: funcionamiento de los periféricos de salida de la tarjeta de desarrollo**

Objetivo: comprobar el correcto funcionamiento de los periféricos de salida con los que cuenta el kit de desarrollo, como son 8 LEDs, dos despliegues de 7 segmentos y un LCD.

Procedimiento: se realizo el proceso de conexión y configuración de la tarjeta de desarrollo desde un computador host encargado de su programación Figura 4-20.La prueba consistió en cargar un programa sobre la tarjeta que permitiera visualizar el funcionamiento de los diferentes periféricos de entrada y salida con los que cuenta esta tarjeta.

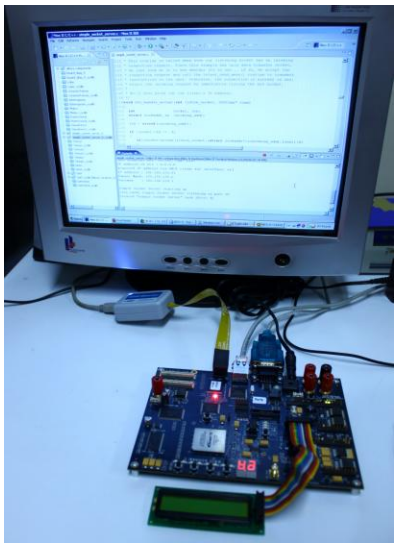


Figura 4-20 Tarjeta de prueba conectada y configurada

Resultado: los periféricos: leds, display de 7 segmentos, LCD, los pulsadores, el puerto Serial, y el puerto Ethernet, funcionan de forma correcta Figura 4-20

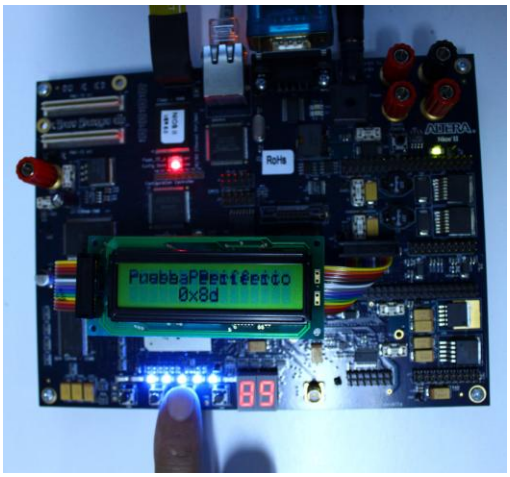


Figura 4-21 Funcionamiento de periféricos de entrada y salida

- **Prueba 2: Depuración remota utilizando línea de comandos.**

Objetivo: Comprobar que la conexión remota que permite la depuración sobre la tarjeta de desarrollo se efectúe de manera correcta.

Procedimiento: 1. La tarjeta de desarrollo es conectada en un equipo host encargado de su configuración, Anexo D. Se ejecuta el servidor de depuración en la ruta: *Inicio> todos los programas> Altera> Nios II EDS 7.2> Nios II 7.2 Command Shell*; seguido del comando: **nios2-gdb-server –tcpport 2342 –tcppersist,** en este punto el servidor se encuentra esperando una conexión, Figura 4-22.

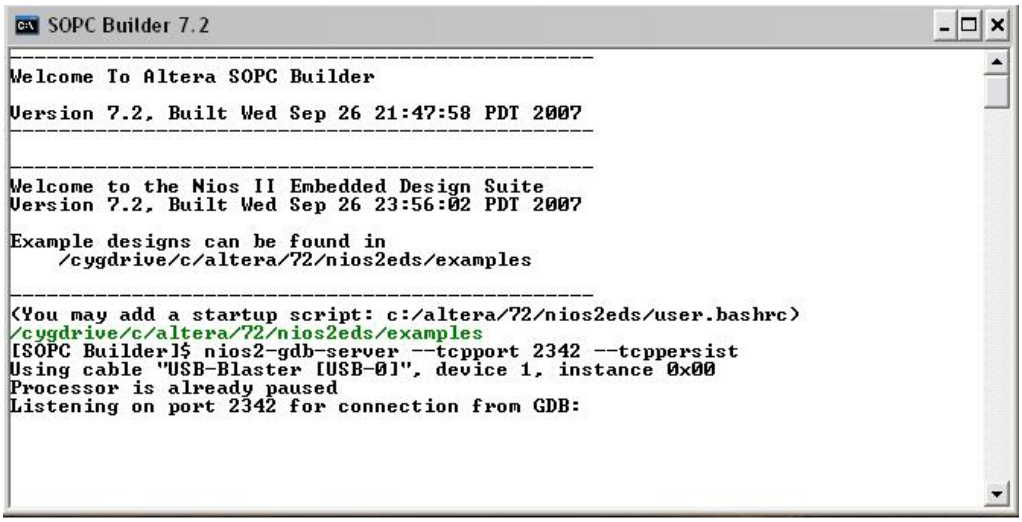


Figura 4-22 Servidor en consola de comandos en espera de una conexión

2. Desde un equipo cliente se inicia conexión con el host utilizando línea de comandos con el mismo programa del índice anterior, seguido de los comandos: **nios2-elf-gdb –cd=”ruta del archivo ejecutable .elf” “nombre del archivo .elf”, target remote “ip equipo host:2342”,** Figura 4-23.

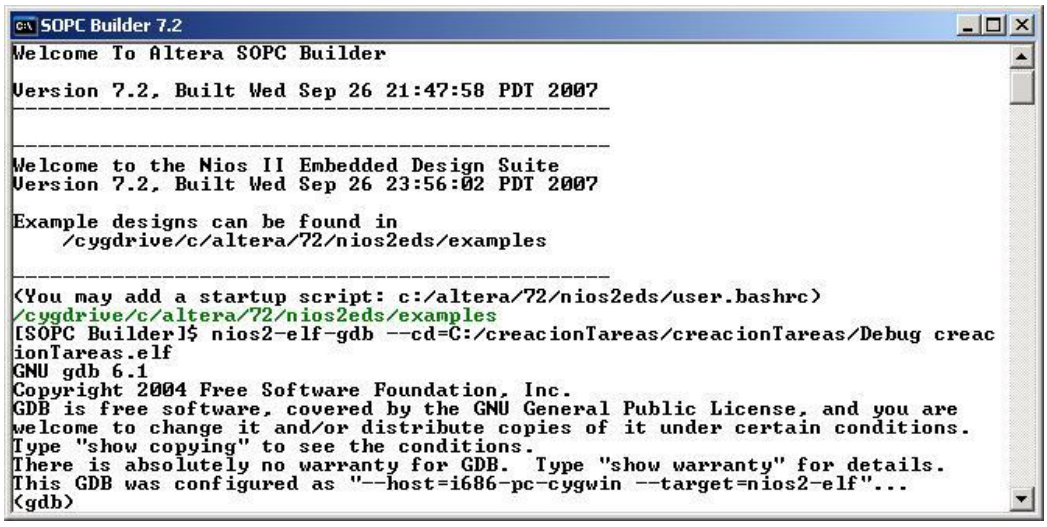


Figura 4-23 Cliente en consola de comandos conectado al servidor

Ya establecida la conexión se permite la descarga de un programa de prueba sobre la tarjeta con el comando: **load**. A continuación son enviados los comandos de depuración Figura 4-24.

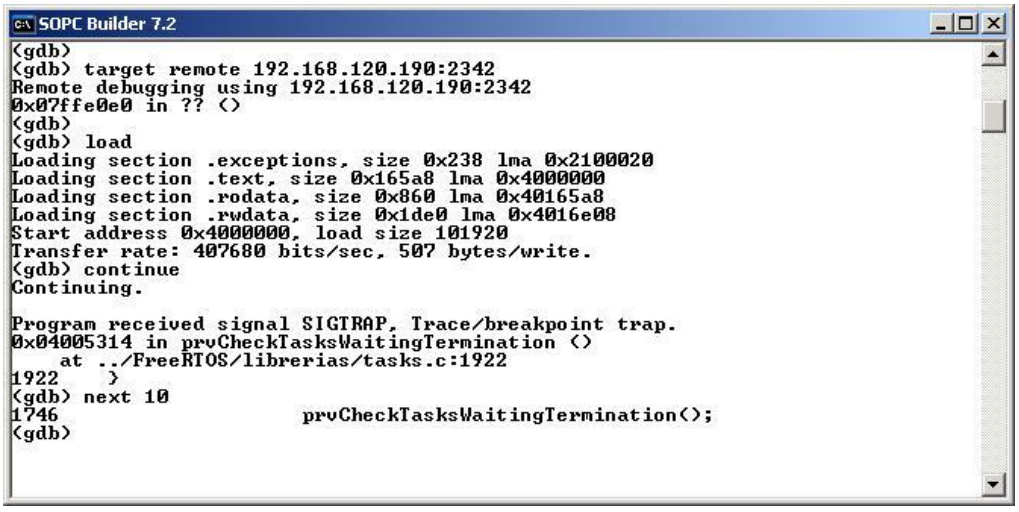


Figura 4-24 Cliente depurando en consola de comandos

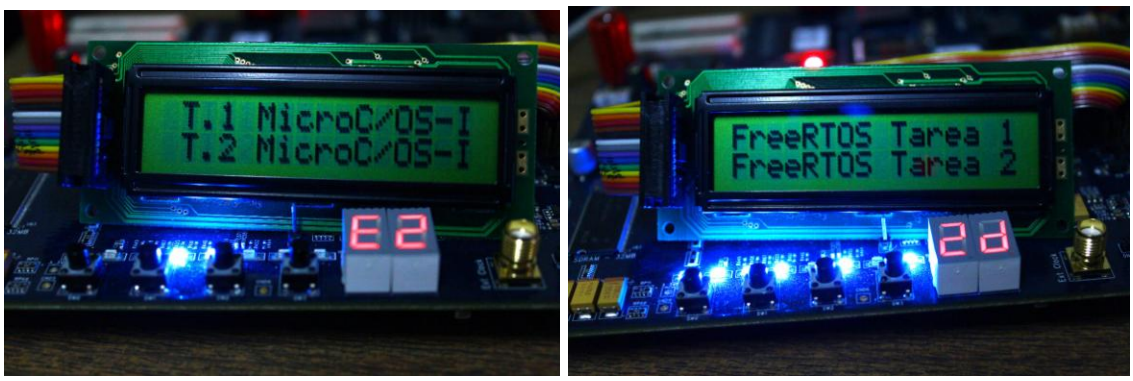
Resultado: se ha podido verificar que la conexión ha sido efectuada entre el equipo cliente y la tarjeta de desarrollo, además se lleva a cabo de manera satisfactoria la depuración de un ejemplo utilizando línea de comandos.

- **Prueba 3: Funcionamiento de los Sistemas Operativos en Tiempo Real**

Objetivo: Probar el funcionamiento de los Sistemas Operativos en Tiempo Real seleccionados sobre la tarjeta de desarrollo Nios II, caso de estudio.

Procedimiento: Después de conectar y configurar correctamente la tarjeta de desarrollo, se procede a configurar los Sistemas Operativos Figura 4-25 b. FreeRTOS, y Figura 4-25 b. y Micro C/OS II como se describe en el Anexo C, y así poder ser integrados al Nios II

IDE; luego se carga y ejecutada una práctica ejemplo en el que se crean dos tareas con diferentes prioridades de cada Sistema Operativo sobre la placa de desarrollo, Figura 4-26.



a. Prueba S. O. Micro C/OS II **b. Prueba S. O. FreeRTOS**

Figura 4-25 Ejecución de práctica de los Sistemas Operativos en Tiempo Real sobre la tarjeta de desarrollo

Resultado: Se comprobó el correcto funcionamiento de cada uno de los Sistemas Operativos: FreeRTOS y Micro C/OS, sobre la placa de desarrollo Nios II, pudiendo ver la ejecución de cada una de las tareas de acuerdo a las prioridades de tiempo asignadas a éstas.

4.3.2 Pruebas sobre funcionalidades de la interfaz de depuración

Como se especifico anteriormente, las siguientes pruebas buscan testear el funcionamiento de las funcionalidades básicas de la interfaz de monitoreo y depuración remoto ejecutadas de forma independiente.

- **Prueba 4: transmisión de video**

Objetivo: Evaluar la calidad del la recepción del video enviado desde el equipo host hacia el equipo cliente.

Procedimiento: Se inicia la aplicación que comunica al servidor Figura 4-26 a., y cliente Figura 4-26 b., que permite transmitir el video entre éstos.

a. Señal de video en el equipo servidor **b. Retardo de video en el equipo cliente**

Figura 4-26 Señal de video con retardo considerable

Observaciones: Se observa la presencia de retardos considerables, de aproximadamente 5 segundos en la señal recibida por el cliente, con respecto a la señal transmitida por el host, lo cual hace poco útil esta funcionalidad ya que lo que se espera es poder ver la ejecución de las aplicaciones sobre la tarjeta de desarrollo para evaluar su correcto funcionamiento.

Solución: Para solucionar este inconveniente se utilizo para realizar la captura de imagen un formato de video de menor peso, y se redujo el tamaño de la imagen.

Resultado: El retardo en la imagen recibida por el equipo cliente se reduce de manera considerable, éste es de aproximadamente 0.3 segundos, de modo que se hace posible utilizar el video como una propiedad que ayuda el trabajo de depuración de manera remota, Figura 4-27.

a. Video en equipo cliente **b. Video en equipo host**

Figura 4-27 Señal de video con retardo disminuido

- **Prueba 5: Conexión de la interfaz grafica con la consola de depuración**

Objetivo: Verificar la recepción y ejecución de comandos de depuración enviados desde la interfaz de depuración por parte de la consola de comandos.

Procedimiento: Utilizando el lenguaje de programación JAVA, se crea una instancia de la clase “process” [82] a través del cual se realiza la ejecución de la consola de depuración y comandos reconocidos por ésta.

Observaciones: Procediendo de esta manera solo fue posible el envío de comandos de manera secuencial seleccionándolos de forma predeterminada, de modo que imposibilitaba el control de la sesión de depuración ya que una vez ejecutados estos comandos, se debía iniciar un nuevo proceso.

Solución: Se opto por utilizar el lenguaje de programación "*C Sharp"* el cual permite ejecutar de igual manera la consola de depuración dentro de un proceso permitiendo el envió de comandos al proceso de ejecución de manera independiente sin necesidad de interrumpirlo o reiniciarlo.

- **Prueba 6: Ejecución de comandos desde la interfaz grafica**

Objetivo: Verificar la correcta ejecución de los comandos de depuración haciendo uso de la interfaz de depuración.

Procedimiento: Se inicia la aplicación servidor en el equipo host, previamente se debe configurar la tarjeta de desarrollo como lo indica el Anexo D; y la aplicación cliente se ejecuta en su respectivo equipo. Se efectúa el proceso de configuración del cliente según lo indicado en el Anexo A, para iniciar el proceso de depuración de una práctica ejemplo sobre la tarjeta de desarrollo Nios II.

Observaciones: Los comandos de configuración e inicio de la sesión de depuración, de captura de información y control se efectuaron de manera correcta a excepción del comando que detiene la ejecución de la práctica sobre la tarjeta.

Solución: Debido a que este comando no se puede representar como una cadena de caracteres, como los demás, el flujo de entrada del proceso no lo logra interpretar de manera adecuada. Luego de investigar la forma en que la consola de depuración reconoce este comando, se determino que éste funciona como una señal de interrupción del proceso. Dado este análisis se opto por crear un proceso codificado en el lenguaje de programación C que permite el envió de este tipo de señales.

Resultado: El proceso llevado a cabo para la resolución de este problema fue satisfactorio, dado que el comando para lograr la detención de la práctica en ejecución sobre la tarjeta de desarrollo funciona de manera adecuada.

4.3.3 Pruebas sobre la interfaz de desarrollo y el manejo de clientes

En la etapa final de la evaluación del funcionamiento de la interfaz de monitoreo, se realizaron pruebas que permiten determinar su desempeño mientras debe atender uno o varios usuarios al tiempo. Y la prueba final que determina su usabilidad siendo utilizado por un grupo de estudiantes de laboratorio de digitales II, quienes después de haber llevado a cabo la prueba, evalúan su usabilidad llenando un test en el cual se ve reflejada su opinión.

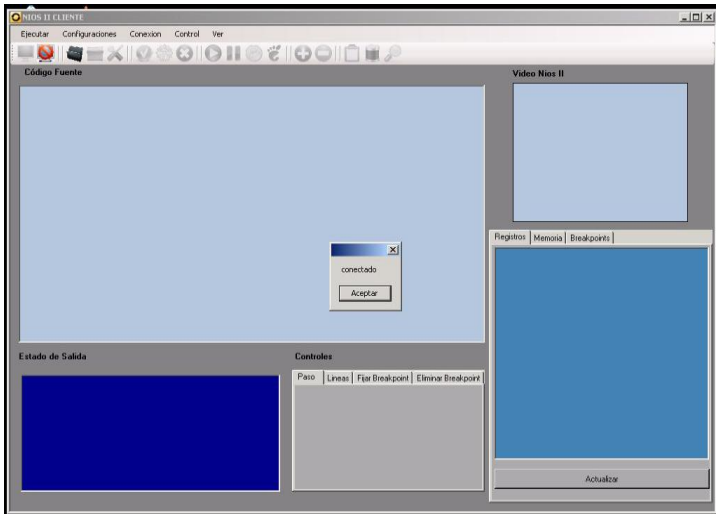
Prueba 7: Ejecución de la interfaz de depuración remota con un solo cliente

Objetivo: Verificar el funcionamiento de la interfaz de depuración remota haciendo la conexión con un equipo cliente.

Procedimiento: Se inicia la aplicación servidor en el equipo host, previamente se debe configurar la tarjeta de desarrollo como lo indica el Anexo D; y la aplicación cliente se ejecuta en su respectivo equipo. Se efectúa el proceso de configuración del cliente según lo indicado en el Anexo B, para iniciar el proceso de depuración de una práctica ejemplo sobre la tarjeta de desarrollo Nios II.

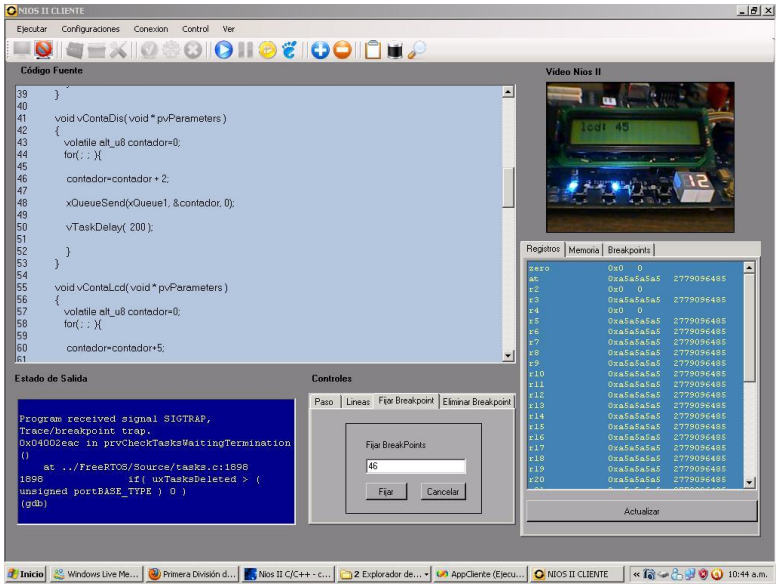
Resultado: Cuando éste procedimiento se lleva a cabo con un solo cliente, se logro efectuar la conexión, como se muestra en la Figura 4-28 a. y b. respectivamente, y llevar a cabo el proceso de depuración Figura 4-28 c., de un ejemplo de practica sobre la tarjeta de desarrollo Nios II con de manera satisfactoria con el cliente y el servidor,

.



a. Conexión de la aplicación cliente

b. Conexión con aplicación servidor



c. Aplicación cliente en depuración

Figura 4-28 Proceso de ejecución de la aplicación de depuración remota

Prueba 8: Ejecución de la interfaz de depuración remota con cuatro clientes

Objetivo: Verificar el funcionamiento de la Interfaz de Monitoreo y Depuración Remota llevando a cabo la conexión con equipo servidor de cuatro equipos cliente.

Procedimiento: Se inicia la aplicación servidor en el equipo host; previo a esto, se debe configurar la placa de desarrollo como lo indica el Anexo B. La aplicación cliente se ejecuta en cada uno de los equipos dispuestos para este fin. Se efectúa el proceso de configuración del cliente en cada equipo, según lo indicado en el Anexo D se ejecuta la opción de "*conectar*" para iniciar el proceso de depuración de una práctica ejemplo sobre la tarjeta de desarrollo Nios II.

Observaciones: Cuando se acepta la petición de conexión hecha por uno de los equipos cliente y se inicia el proceso de monitoreo de la práctica ejemplo sobre la tarjeta de desarrollo, el video recibido por el equipo conectado presenta un incremento en el retardo de tal modo que éste recurso termina por paralizarse terminando por volverse inoperante.

Solución: Se optó por aislar la conexión de los equipos en prueba en una red LAN única establecida para efectuarla; ya que se conjeturó que el tráfico presente en la red durante la prueba, provocaba un efecto de sobrecarga a la hora de transmitir video. En consecuencia se presentaba fallas en su funcionamiento.

Resultado: Cuando se realizo el proceso de prueba dese el inicio bajo esta nueva condición, el retardo en la señal de video, de 0.4 segundos aproximadamente, recibida por cada equipo cliente cuando se encontraba conectado a la tarjeta, se consideró permisible para el caso de funcionamiento.

Prueba 9: Evaluación del desempeño del aplicativo por un grupo de estudiantes

Objetivo: Efectuar la evaluación del desempeño de la interfaz de monitoreo y depuración remota para Nios II por un grupo de estudiantes de laboratorio de Circuitos Digitales II.

Procedimiento: En una sala de cómputo en la que se imparte la asignatura de laboratorio II de Circuitos Digitales, se configuran cuatro equipos como clientes y un equipo como servidor como lo muestra la Figura 4-29, se propone a un grupo de 8 estudiantes de éste curso, que lleven a cabo la guía No. 1 del Anexo A, en la que se debe manejar las características de la herramienta y midan su usabilidad llenando un test de desempeño.

Figura 4-29 Configuración de cuatro equipos cliente y un equipo servidor

Después de hacer una introducción para aclarar el manejo y objetivo de la herramienta los estudiantes procedieron a utilizar el sistema según las instrucciones del manual consignado en el Anexo A.

Resultado: La prueba fue exitosa debido a que cada estudiante tuvo la oportunidad de interactuar, por intermedio de la interfaz, con la tarjeta de desarrollo Stratix II con procesado adaptable Nios II, programando sobre ésta un ejemplo de práctica, y llevando a cabo el proceso de depuración sin inconvenientes.

Por otro lado los resultados arrojados en la evaluación, medidos a través de la encuesta cuyo formato se encuentra en el Anexo E, indicaron que la interfaz cumple con los requisitos de usabilidad satisfactoriamente, como se muestra en la siguiente tabla donde se muestra el número de estudiantes por característica a evaluar:

Tabla 22 Resultados de la evaluación efectuada por estudiantes

Característica en evaluación	Calificación			
	Excelente	Bueno	Regular	Malo
Funcionalidad	6	2	0	0
Confiabilidad	5	3	0	0
Usabilidad	5	3	0	0
Eficiencia	7	1	0	0
Mantenibilidad	5	3	0	0
Funcionalidad	7	1	0	0
Total	34	14	0	0

De los anteriores datos se concluye que de un total de 48 valoraciones posibles; 34 o un porcentaje del 70.84%, opinan que la interfaz de depuración en conjunto, tiene una calificación de excelente; y que 14 o el 29.16%, opina que es buena. Además, hicieron manifiesta su aprobación verbalmente e interés por las potencialidades que tiene el presente trabajo de grado, como se puede observar en las siguientes figuras.

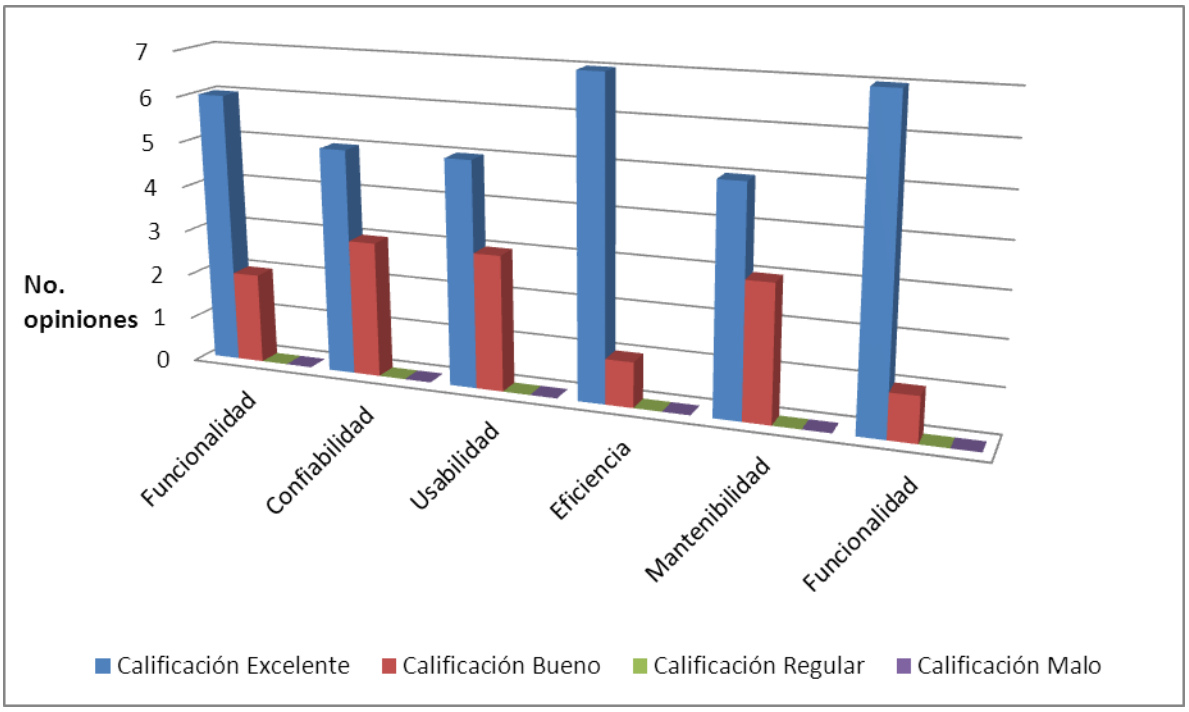


Figura 4-30 Resultado de evaluación con estudiantes

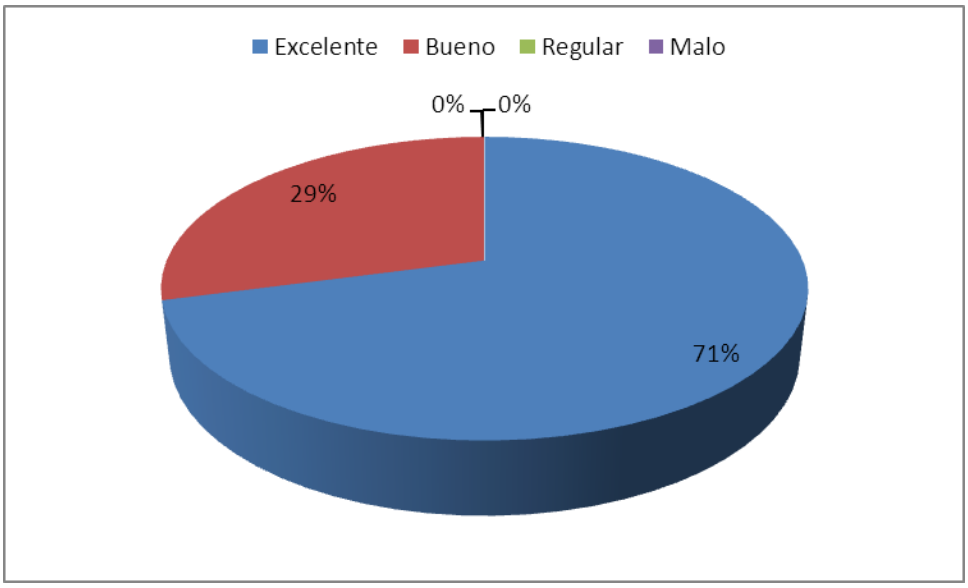


Figura 4-31 Porcentaje del nivel de aceptación

Capítulo V Conclusiones y Trabajos Futuros

En el capítulo que a continuación se desarrolla, se describen las conclusiones relacionadas con presente trabajo de grado, como también se propone algunos trabajos futuros que involucran diferentes aspectos que tomaron parte en la construcción del proyecto.

5.1 Conclusiones

En este aparte se generan algunas conclusiones que describen los aportes y objetivos alcanzados por este trabajo.

- Una consideración muy importante a la que se llega con la construcción del presente trabajo de grado, es la potencialidad que tiene los Sistemas de Tiempo Real, por cuanto se puede concluir que prácticamente en todas las actividades que se llevan a cabo cotidianamente, en cualquier sociedad, sin importar el nivel de complejidad, se tiene contacto con un aplicativo diferente de estos sistemas; por lo que su estudio detallado se convierten en una herramienta de gran valor por su proyección hacia el futuro en la formación integral de ingenieros en la FIET de la Universidad del Cauca.

- Mediante el desarrollo del presente trabajo de grado se logra darle un aprovechamiento mayor a un recurso físico, el kit de desarrollo: Nios II Development Kit, Stratix II Edition, con el que se cuenta en la FIET, haciéndolo accesible a un gran número de estudiantes de los diferentes programas académicos impartidos en la facultad, proporcionándoles nuevas posibilidades de experiencias en las que se busque una profundización del conocimiento y experimentación sobre Sistemas de Tiempo Real, de modo que algunos de sus conceptos más importantes puedan se ser llevados a la práctica haciendo didáctica su comprensión.

- En el proceso de construcción del presente proyecto se pudo determinar que como éste kit de desarrollo, existen muchos otros recursos disponibles en la FIET de diferente índole, que pueden ser aprovechados de múltiples maneras en pos de buscar incrementar las herramientas educativas como a poyo a diversas actividades de formación académica, masificando el conocimiento en prácticas laboratorios, practicas de campo, entre otras formas. Y como complemento a lo anterior, se puede resaltar la importancia de apoyar y buscar el desarrollo de proyectos como éste, que buscan aportarle de manera recursiva al enriquecimiento de la academia en la Universidad en general.

- Se puede concluir que es posible llevar a cabo una práctica de laboratorio de manera remota contando con las herramientas adecuadas para este fin, por lo que ésta posibilidad se convierte en un modo efectivo, simple y de bajo costo para la administración de los limitados recursos físicos con los que puede contar una universidad pública, como lo es la Universidad del Cauca, permitiendo una experimentación completa y a la vez optimizando la conservación de los equipos prolongando su utilización.

- Después de llevar a cabo las pruebas de usabilidad de la interfaz de monitoreo y depuración con un grupo de estudiantes del Laboratorio II de Sistemas Digitales, se concluye que es de gran importancia contar con este tipo de herramientas y motivar su evolución ya que en este escenario se pudo confirmar su gran usabilidad y gran cantidad de posibles aplicaciones.

5.2 Trabajos Futuros

En esta sección se presenta algunas propuestas de trabajos futuros relacionados con las diferentes temáticas desarrolladas en el presente proyecto de grado buscando complementar y potenciar las posibilidades que se aportan en este documento.

- **Complementación de las aplicaciones de la Interfaz de Monitoreo y Depuración Remota para Nios II.**

  Dado que el alcance que presenta el desarrollo del presente proyecto es el de generar una interfaz de depuración y monitoreo remota circunscrita para ser utilizada en una red LAN. Se propone complementar estas expectativas incrementando la aplicabilidad de la herramienta incluyendo la posibilidad de construcción y compilación de código, como también hacer posible su utilización desde cualquier ubicación implementando su modo de acceso a través la red de Internet.

- **Estandarización del la Interfaz de Monitoreo y Depuración Remota para Nios II.**

  En vista de que la FIET cuenta con otras tarjetas de desarrollo del mimo fabricante de la tarjeta de desarrollo caso de estudio, como la Cyclone y Cyclone III, y dada la posibilidad de nuevas adquisiciones de otros equipos como estos. Se convierte en una opción interesante estandarizar el aplicativo para que pueda ser utilizado por otros equipos con similares características.

- **Implementación de nuevos módulos de prueba sobre la tarjeta de desarrollo.**

  Teniendo en cuenta las características del kit de desarrollo Stratix II con procesador adaptable Nios II, en cuanto al manejo de periféricos de entrada. Se propone la implementación de nuevos módulos de prueba que puedan ser conectados a esta, con sus respectivas prácticas de laboratorio, que permitan un seguimiento en tiempo real como sensores de temperatura, de presión; dispositivos de transmisión de información en tiempo real, señales de control inteligente de tráfico vehicular, entre otros.

- **Desarrollo de estudios e implementación de laboratorios remotos**

  Siendo esta una opción de gran interés en la masificación del uso de los recursos con los que se cuentan en la FIET y más aún, en la Universidad en general. Se propone el estudio y aplicación de los conceptos de laboratorio remoto ya que haciendo uso de éstos muchas de las experiencias de laboratorio podrían llevarse a cabo desde sitios distantes de la ubicación de los equipos de experimentación, sin perder el carácter de interacción que busca una práctica en laboratorio.

- **Desarrollar aplicaciones construidas sobre Sistemas de Tiempo Real**

  En consecuencia con la búsqueda del manejo académico de los Sistemas de Tiempo Real se presentan las posibilidades de llevar todo este conocimiento a la práctica desarrollando proyectos que busquen generar aplicaciones que solucionen problemas de la cotidianidad como el monitoreo de signos vitales, manejo de tráfico vehicular, o el monitoreo de la ubicación de niños a distancia, entre otras.

Referencias Bibliográficas

[1] Mónica Casalet,Felipe Lara Rosano, “Tecnología: concepto, problemas y perspectivas”, 1ra edición, Siglo Veintiuno Editores S. A., 1998 disponible en: http://books.google.com.co/books?id=SSqDGtPR7T0C&printsec=frontcover&dq=tecnologia&hl=es&ei=WLmgTKDWK4P78AaV-OVR&sa=X&oi=book_result&ct=result&resnum=1&ved=0CCcQ6AEwAA#v=onepage&q=tecnologia&f=false, [Citado 8 de marzo de 2010]

[2] Mario Aldea Rivas “*Planificación de Tareas en Sistemas Operativos de Tiempo Real Estricto para Aplicaciones Empotradas” Trabajo de Título de Tesis doctoral en Ciencias (fisicas), Universidad de Cantabria, Noviembre 2002, disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.9880&rep=rep1&type=pdf, [Citado 28 de marzo de 2010]

[3] Phillip A. Laplante, “REAL-TIME SYSTEMS DESIGN AND ANALYSIS”, 3th ed, IEEE Press Editorial Board, 2004 [Citado 3 de marzo de 2010]

[4] John Goyder, “Technology and society”, Segunda edición, Broadview Press Ltd., 2005, disponible en: http://books.google.com.co/books?id=fJLXSt4qBr0C&printsec=frontcover&dq=technology+and+society&hl=es&ei=TzmTLuFGoH98AaBveH8AQ&sa=X&oi=book_result&ct=result&resnum=1&ved=0CCoQ6AEwAA#v=onepage&q&f=false, [Citado 12 de julio de 2010]

[5] Nios II Development Kit, Stratix II Editionhttp://www.altera.com/products/devkits/altera/kit-niosii-2S60.html, [Citado noviembre de de 2009]

[6] Editado por Jorge Real,Tullio Vardanega, “Reliable Software Technologies - Ada-Europe 2010: 15th Ada-Europe 2010”, Universidad Politécnica de Valencia, 2010, disponble en: http://books.google.com.co/books?id=0FmuFnNFtbUC&pg=PP11&dq=evolution+of+%22real-time+systems%22&hl=es&ei=NrCjTMP2OIKB8gaH-eXCCQ&sa=X&oi=book_result&ct=result&resnum=4&ved=0CDoQ6AEwAzgK#v=onepage&q=evolution%20of%20%22real-time%20systems%22&f=false, [Citado 14 de Agosto de 2010].

[7] Chowdary Venkateswara Penumuchu, “Simple Real-Time Operating System: A Kernel Inside View for a Beginner”, Trafford Publishing, 2007, disponible en: http://books.google.com.co/books?id=3EjaPFRv2IgC&printsec=frontcover&dq=%22real-time+operating+system%22&hl=es&ei=erujTMcDwv3wBrLNvKEK&sa=X&oi=book_result&ct=result&resnum=1&ved=0CCkQ6AEwAA#v=onepage&q&f=false, , [Citado 10 de Julio de 2010].

[8] Qing Li, Caroline Yao, “Real-time concepts for embedded systems”, CMPBOOKS, 2003, disponible en: http://books.google.com.co/books?id=c_F2ckT-ZVsC&printsec=frontcover&dq=Operating+System+%22real-time+operating+system%22&hl=es&ei=t7-jTNTjJcK88gaV2LXnCg&sa=X&oi=book_result&ct=result&resnum=4&ved=0CDYQ6AEwAw

#v=onepage&q=Operating%20System%20%22realtime%20operating%20system%22&f=false, [Citado 3 de julio de 2010]

[9] Prasad, “mbedded Real Time Systems: Concepts, Design & Programing”, Hilmal impressions, New Delhi, 2009, disponible en: http://books.google.com.co/books?id=YHk43y8mSIsC&pg=PA181&dq=Operating+System+%22realtime+operating+system%22&hl=es&ei=t7jTNTjJcK88gaV2LXnCg&sa=X&oi=book_result&ct=result&resnum=9&ved=0CFQQ6AEwCA#v=onepage&q=Operating%20System%20%22realtime%20operating%20system%22&f=false, [Citado 25 de Julio de 2010].

[10] Open Kernel Labs and NICTA and University of New South Wales Sydney, Australia. “The Role of Virtualization in Embedded Systems” Documento en formato pdf disponible en: http://ertos.org/publications/papers/Heiser_08.pdf. [Citado 10 de Agosto de 2009]

[11] lEEE TRANSACTIONS ON EDUCATION, VOL. 51, NO. 3, AUGUST 2008. “Using a Low-Cost SoC Computer and a Commercial RTOS in an Embedded Systems Design Course”. Documento en formato pdf disponible en: http://users.ece.gatech.edu/~hamblen/ papers/IEEETRANED2008.pdf [Citado 18 de Agosto de 2009].

[12] SLS Group, TIMA Laboratory. “Automatic Generation of Fast Timed Simulation Models for Operating Systems in SoC Design”. Documento en formato pdf disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.1148 &rep=rep1&type=pdf. [Citado 22 de Agosto de 2009]

[13] Carsten Nitsch, Karlheinz Weiss, Thorsten Steckstor, Wolfgang Rosenstiel: “Embedded System Architecture Design Based on Real-Time Emulation”. Documento en formato pdf disponible en: http://xoraya.net/publicationen/rsp2000.pdf. [Citado 18 de Agosto de 2009].

[14] School of Electrical and Computer Engineering, Georgia Institute of Technology Atlanta, Georgia: “A Configurable Hardware Scheduler for Real-Time Systems”. Documento en formato pdf disponible en: http://www.ittc.ku.edu/~dandrews/690/files/mooney2.pdf. [Citado 22 de Agosto de 2009].

[15] ENSEA, University of Cergy-Pontoise, France: “A Modular SystemC RTOS Model for Embedded Services Exploration”. Documento en formato pdf disponible en: http://publi-etis.ensea.fr/2007/HMV07/HMV07.pdf. [Citado 22 de Agosto de 2009].

[16] Luleå University of Technology: “Timber as an RTOS for Small Embedded Devices”. Documento en formato pdf disponible en: http://www.sics.se/realwsn05/papers/kero05timber.pdf. [Citado 26 de Agosto de 2009].

[17] Escuela Técnica Superior de Ingeniería Informática, URJC, Madrid, España “Sistema operativo para SMPs basados en MicroBlaze”. Documento en formato pdf disponible en: http://www.escet.urjc.es/~phuerta/pdf/articulo_JCRA_2008.pdf.[Citado 26 de Agosto de 2009].

[18] David Andrews, Iain Bate, Thomas Nolte, Clara M. Otero P´erez, Stefan M. Petters: “Impact of Embedded Systems Evolution on RTOS Use and Design”. Documento en

formato pdf disponible en: https://wiki.ittc.ku.edu/hybridthread/images/6/64/ Ospert.pdf. [Citado 26 de Agosto de 2009].

[19] Integrated Circuits and Systems Laboratory Department of Computer Science - Technische Universität Darmstadt, Alemania: " Real-Time Operating System Services for Realistic SystemC Simulation Models of Embedded Systems". Documento en formato pdf disponible en: http://www.iss.tu-darmstadt.de/staff/klaus/publications /FDL04.pdf. [Citado 28 de Agosto de 2009].

[20] Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Departamento de Computación: "Administración de Interrupciones en Sistemas Operativos de Tiempo Real". Documento en formato pdf disponible en: http://www.cs.cinvestav.mx/Estudiantes/TesisGraduados/2008/tesisLuisLeyva.pdf. [Citado 31 de Agosto de 2009].

[21] Universidad de Antioquia, Grupo de Microelectrónica y Control: "Evaluación del Sistema Operativo µClinux para su Utilización en Sistemas Embebidos". Documento en formato pdf disponible en: http://microe.udea.edu.co/proyectos/DMA/enlaces/ articulo/EvaluacionuClinux.pdf. [Citado 12 Septiembre de 2009].

[22] Jean J. Labrosse, "MicroC/OS-II: the real-time kernel", 2th ed, Cmp Books, 2002.

[23]Dr. Ing. Álvaro Rendón Gallón, conferencia: "Introducción a los Sistemas de Tiempo Real", Universidad del Cauca, Departamento de Telemática, Popayán, septiembre de 2006

[24] D. Ionescu, A. Cornell, "Real-Time Systems Modeling, Design, and Applications", AMAST Series in Computing: vol. 8, World Scientific Publishing Co. Pte. Ltd. 2007.

[25] Yanbing Li, Miodrag Potkonjak, and Wayne Wolf, "Real-Time Operating Systems for Embedded Computing", Department of Electrical Engineering, Princeton University Department of Computer Science, UCLA, 1997, [Consultada: Marzo 6 de 2010]

[26] Alan Burns, Anty Wellings, "Sistemas de Tiempo Real y Lenguajes de Programación", 3ª ed, Addison Wesley, 2003, disponible en: Biblioteca central Universidad del Cauca 005.133 B967 3ED [Consultada: Enero 18 de 2010].

[27] Michael González Harbour," REAL-TIME POSIX: AN OVERVIEW", Departamento de Electrónica, Universidad de Cantabria, June 1993, pp 1-7, [Consultada: 11 de abril de 2010].

[28] Pablo Ruiz Múzquiz, "Sistemas Operativos", ed 0.5.0, alqua, abril 2004, Disponible en: http://www.sindominio.net/metabolik/alephandria/txt/SSOO-0_5_0.pdf, [Consultada: Enero 30 de 2010].

[29] Mario Aldea Rivas "Planificación de Tareas en Sistemas Operativos de Tiempo Real Estricto para Aplicaciones Empotradas", Universidad de Cantabria, Facultad de Ciencias,

Departamento de Electrónica y Computadores, noviembre de 2002, disponible en: http://marte.unican.es/documentation/tesis-mario.pdf, [Consultada: Enero 18 de 2010].

[30] José Ismael Ripoll Ripoll, “Planificación en sistemas de Tiempo Real”, Universidad Politécnica de Valencia, pp 2-3, Dpto. de Informática de Sistemas y Computadores, disponible en: http://www.gii.upv.es/personal/iripoll/str/planificacion/planif.pdf, [Consultado: Enero 18 de 2010].

[31] John A. Starnkovic, R. Rajkumar “Real-Time Operation Systems”, University of Virginia, Carnegie Mellon University, kluwer Academic Publishers, 2004 pp 1-2, disponible en: http://www.secs.oakland.edu/~ganesan/old/courses/CSE666%20F06/ RT% 20operating%20Systems%20Jan%202005.pdf, [Consultada: Enero 18 de 2010].

[32] Milan Milenkovich, "Sistemas Operativos: Conceptos y Diseño". Mc Graw Hill, Madrid, España. 1994, disponible en: biblioteca central Universidad del Cauca, cód. 001.642, [Consultado: Enero 18 de 2010].

[33] William Stallings, "Sistemas Operativos: Principios de Diseño e Interioridades", 4a ed., Prentice Hall, Madrid, España 2001, disponible en: biblioteca central Universidad del Cauca, cód.005.43, [Consultado: Enero 18 de 2010].

[34] Stephen R. Walli, “The POSIX Family of Standards”, StandardView Vol. 3, No. 1, SRW SOFTWARE, KITCHENER, ONT, marzo 1995, disponible en: http://stephesblog.blogs.com/papers/acm-posix.pdf, [Consultado: Enero 18 de 2010].

[35] The Open Group, “lEEE Std 1003.1, 2004 Edition”, disponible en: http://www.unix.org/version3/ieee_std.html, [Consultado: Febrero 4 de 2010].

[36] eCos Home Page, “eCos”, disponible: http://ecos.sourceware.org/, [Consultado: Febrero 4 de 2010].

[37] segger, “SEGGER Microcontroller - embOS”, disponible en: http://www.segger.com/ cms/embos.html, [Consultado: Febrero 4 de 2010].

[38] ERIKA Enterprise, “Erika Enterprise and RT-Druid”, disponible en: http://erika.tuxfamily.org/, [Consultado: Febrero 4 de 2010].

[39] FreeRTOS, “FreeRTOS-A Free RTOS for ARM7, ARM9” disponible en: http://www.freertos.org/index, [Consultado: Mayo 23 de 2010].

[40] LynuxWorks, “RTOS - Real-time operating systems for embedded real-time systems, from LynuxWorks”, disponible en:http://www.lynuxworks.com/rtos/, [Consultado: Enero 18 de 2010].

[41] MaRTE OS, “MaRTE OS Home Page”, disponible en:http://marte.unican.es/, [Consultado: Mayo 22 de 2010].

[42] QNX Software Systems, “Operating Systems”, disponible en: http://www.qnx.com/ products/index.html, [Consultado: Enero 18 de 2010].

[43] Mentor Graphics, “/products/embedded_software/nucleus_rtos/”, disponible en: http://www.mentor.com/products/embedded_software/nucleus_rtos/, [Consultado: Mayo 22 de 2010].

[44] WITTENSTEIN High Integrity Systems, “index”, disponible en: http://www.openrtos .com/index, [Consultado: Mayo 27 de 2010].

[45] QNX Software Systems, “QNX Realtime Operating System (RTOS) software”, disponible en: http://www.qnx.com/, [Consultado: Mayo 26 de 2010].

[46] RTLinuxFree, “Wind River : RTLinuxFree”, disponible en:http://www.rtlinuxfree.com/, [Consultado: Mayo 3 de 2010].

[47] Express Logic, “RTOS - Real-Time Operating Systems for Embedded Development, Real Time System By Express Logic”, disponible en: http://www.rtos.com/page/product.php?id=2, [Consultado: Mayo 3 de 2010].

[48] Micrum, “Micrium - µC/OS-II Kernel”, disponible en: http://micrium.com/page/products/ rtos/os-ii, [Consultado: Mayo 10 de 2010].

[49] Wind River, “Wind River Home”, disponible en:http://www.windriver.com/, [Consultado: Junio 5 de 2010].

[50] The FreeRTOS Project, “FreeRTOS-A Free RTOS for ARM7, ARM9, Cortex-M3”, disponible en: http://www.freertos.org/index.html?http://www.freertos.org/FreeRTOS-quick-start-guide.html, [Consultado: Junio 5 de 2010].

[51] FreeRTOS,” FreeRTOS-A Free RTOS for ARM7, ARM9, Cortex-M3,” http://www.freertos.org/index.html?http://www.freertos.org/%20a00114.html, [Consultado: Junio 5 de 2010].

[52] FreeRTOS, “FreeRTOS Documentation and Book”, disponible en: http://www.freertos.org/Documentation/FreeRTOS-documentation-and-book.html, [Consultado: Junio 25 de 2010].

[53] Mycrum, “FreeRTOS-A Free RTOS for ARM7, ARM9, Cortex-M3, MSP430”, http://www.freertos.org/index.html?http://www.freertos.org /a00090.html, [Consultado: Junio 25 de 2010].

[54] Mycrum, “Micrium - µC/OS-II Kernel”, disponible en: http://micrium.com/page /products/rtos/os-ii, [Consultado: Enero 18 de 2010], [Consultado: Junio 25 de 2010].

[56] mycrum, ”Micrium - Site map”, disponible en: http://micrium.com/newmicrium/uploads /file/ datasheets/ucosii_datasheet.pdf, [Consultado: Julio 4 de 2010].

[57] Mycrum, “Micrium - Partners”, disponible en: http://micrium.com/page/partners, [Consultado: Septiembre 2 de 2010],

[58] Tammy Noergaar, “Embedded Systems Architecture, A Comprehensive Guide for Engineers and Programmers”, Newnes, 2005, [Consultado: Julio 2 de 2010].

[59] ALTERA, ”n2sw_nii5v2.pdf (application/pdf Objeto)” Nios II Software Developer’s Handbook, disponible en: http://www1.cs.columbia.edu/~sedwards/classes /2009/4840/n2sw_nii5v2.pdf, [Consultado: Julio 4 de 2010].

[60] Altera, “Quartus II Version 7.2 Handbook, Volume 1: Design and Synthesis,” disponible en: http://www.cs.columbia.edu/~sedwards/classes/2010/4840/qts_qii5v1.pdf, [Consultado: Julio 4 de 2010].

[61] ALTERA, “Literature: Nios II Processor”, disponible en: http://www.altera.mobi/literature/lit-nio2.jsp#related _documentation, [Consultado: Julio 4 de 2010].

[62] ALTERA, “Nios II Processor Reference Handbook”, http://www.altera.com/literature /hb/nios2/ n2cpu_nii5v1.pdf, [Consultado: Julio 15 de 2010].

[63] ALTERA, “Nios Development Board Stratix II Edition, Reference Manual”, http://www.altera.com/literature/manual/mnl_nios2_board_stratixII_2s60_rohs.pdf, [Consultado: Julio 15 de 2010].

[64] Wikipedia, “PCI Mezzanine Card - Wikipedia, the free encyclopedia”, disponible en: http://en.wikipedia.org/wiki/PCI_Mezzanine_Card, [Consultado: Julio 22de 2010].

[65] Dhiman Deb Chowdhury, “High speed LAN technology handbook”, Springer, 2004, disponible en: http://books.google.com.co/books?id=Dgyo8iIv1ZMC&printsec=frontcover&dq=LAN&hl=es&ei=H0ysTLC0FYL58Aaf8PyWCA&sa=X&oi=book_result&ct=result&resnum=2&ved=0CDEQ6AEwAQ#v=onepage&q&f=false, [Consultado junio 15 de 2010].

[66] ALTERA, “ug_usb_blstr.pdf (application/pdf Objeto)”, disponible en:, http://www.altera.com/literature/ug/ug_ usb_blstr.pdf, [Consultado: Julio 22 de 2010].

[67] IEEE 1149.1 JTAG Boundary-Scan Testing, “literature”, disponible en: http://www.altera.com/literature/lit-index.html, [Consultado: Julio 22 de 2010].

[68] Ing. Francisco Javier Terán C., “ENFASIS III, REDES TELEMATICAS II”, Popayán 2008

[69] Los Sockets, “Sockets.pdf (application/pdf Objeto)”, disponible en: http://www.angelfire.com/trek/storwald/Sockets.pdf, [Consultado: Julio 25 de 2010].

[70] Debugging with gdb, “gdb”, disponible en: http://sourceware.org/gdb/current/onlinedocs/gdb/, [Consultado: Julio 25 de 2010].

[71] Debugging Remote Programs, http://sourceware.org/gdb/current/onlinedocs/gdb/Remote-Debugging.html# Remote-Debugging, [Consultado: Julio 25 de 2010].

[72] Pat Eyler, “Guia Avanzada Redes Linux con TCP/IP”, Pearson Educacion, S.A., Madrid, 2001.

[73] GDB protocol, gdb Remote Serial Protocol, “GDB: The GNU Project Debugger”, disponible en: http://sourceware.org/gdb/current/onlinedocs/gdb/Remote-Protocol.html# Remote-Protocol, [Consultado: Julio 28 de 2010].

[74]A COMPARISON OF MICROSOFT'S C# PROGRAMMING LANGUAGE TO SUN MICROSYSTEMS' JAVA PROGRAMMING LANGUAGE http://www.25hoursaday.com/csharpvsjava.html, [Consultado: Julio 28 de 2010].

[75] POO; “Lenguajes de programación, programación Orientada a Objetos”, disponible en: http://www.lenguajes-de-programacion.com/programacion-orientada-a-objetos .shtml, [Consultado: Julio 28 de 2010].

[76] Luis R. Izquierdo, “Introducción a la Programación Orientada a Objetos”, articulo en formato pdf disponible en: http://luis.izqui.org/resources/ProgOrientadaObjetos.pdf

[77] Microsoft, “.NET Framework: Overview”, disponible en: http://www.microsoft.com/net/overview.aspx, [Consultado: Julio 28 de 2010].

[78] Ecma International, “What is Ecma International”, disponible en: http://www.ecma-international. org/memento/index.html, [Consultado: Agosto 4 de 2010].

[79] ISO organisation, “ISO - About ISO”, disponible en: http://www.iso.org/iso/about.htm, [Consultado: Agosto 24 de 2010].

[80] By Brian W. Kernighan and Dennis M. Ritchie, “The C programming Language”, Prentice-Hall, 1988, disponible en: http://net.pku.edu.cn/~course/cs101/2008/resource/The_C_Programming_Language.pdf, [Consultado: agosto 8 de 2010]

[81] Escrito por Harvey M. Deitel, “Cómo programar en C/C++ y Java”, cuarta edición, Prentice Hall, 4004, disponible en: http://books.google.com.co/books?id=H9zwxk6jsMoC&pg=PA6&dq=lenguajes+de+programacion+de+alto+nivel&hl=es&ei=s6arTJOAB8KB8gaqjcmoCA&sa=X&oi=book_result&ct=result&resnum=7&ved=0CEUQ6AEwBg#v=onepage&q&f=false, [Consultado: agosto 26 de 2010]

[82] ORACLE, “Process (Java 2 Platform SE v1.4.2)”, disponible en: http://download.oracle.com/javase/1.4.2/docs/api/java/lang/Process.html., [Consultado: marzo 3 de 2010].