Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Electrónica

Estudio del estado del arte para el núcleo de un nodo sensorial de la red inalámbrica de telecomunicaciones para la protección del ambiente

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura

José Pablo Guadamuz Calderón

Cartago, 5 de Marzo de 2009

INSTITUTO TECNOLOGICO DE COSTA RICA

ESCUELA DE INGENIERIA ELECTRONICA

PROYECTO DE GRADUACIÓN

TRIBUNAL EVALUADOR

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Ing. Pablo Alvarado Moya

Profesor lector

Ing. Nestor Hernandez Hollster

Profesor lector

Ing. Johan Carvajal Godínez

Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

Cartago, 5 de Marzo de 2009

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Cartago, 5 de Marzo de 2009

Firma: 

José Pablo Guadamuz Calderón

Cédula: 11156 0894

Resumen

Una red de sensores inalámbrica es un grupo de computadoras con capacidades sensitivas y de comunicación inalámbrica; estas computadoras se les conocen como nodos sensoriales. Existen muchos tipos de redes de sensores que tiene capacidad de organizarse autónomamente, independientemente de la situación. En las redes de sensores se busca hacer un uso eficiente de la energía, es por esto que este proyecto se ha centrado en la necesidad de encontrar un procesador de bajo consumo energético y con capacidad de procesamiento tal que pueda ejecutar los algoritmos desarrollados para la detección de fuego, sonidos de motosierras y disparos.

Abstract

The wireless sensor networks (WSN) are groups of embedded systems with capabilities like sensing and wireless communication; these computers are known as sensorial nodes. There are many kinds of sensor networks, including those that can autonomously organize them. One of the key capabilities of WSN node is power efficiency, it is for this reason that this project has been oriented to find a processor with a tradeoff among energy consumption and processing capability, so this hardware can support the algorithms required to detect fire, chainsaw and gun shot sounds.

Palabras claves: redes de sensores inalámbricas, nodo sensorial, microcontrolador.

Dedicadoria

Dedico este trabajo a todas las personas que confiaron en mí y me dieron siempre su apoyo incondicional.

Agradecimientos

Tendría que nombrar a muchas personas cosa que no puedo hacer, solo sepan les agradezco infinitamente su apoyo y confianza en todo momento.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1: Introducción

1.1 Problema existente e importancia de su solución

El Instituto Tecnológico de Costa Rica está trabajando en soluciones tecnológicas y económicas que permitan al país prevenir los daños causados a los bosques, esto a través de la Escuela de Ingeniería en Electrónica.

Específicamente la Escuela de Ingeniería Electrónica trabaja en el diseño e implementación de una red de sensores que permita detectar incendios forestales, tala de árboles, casería ilegal, y otros agentes causantes de daño al ambiente.

En años anteriores se han logrado diseñar e implementar algoritmos para detectar la presencia de fuego, motosierras o disparos en las zonas protegidas de nuestro país, esto a través de detectores analógicos o circuitos digitales. También se ha logrado establecer un protocolo para poder transmitir las señales de alarma generadas por los sensores, esto desde el sensor más lejano hasta el más cercano [24, 25, 26, 27, 28].

Actualmente no se ha diseñado aún el sistema empotrado capaz de administrar, agrupar y comunicar los módulos de un nodo de la red de sensores.

Al diseñar e implementar el sistema empotrado se podrá disponer con un nodo sensorial con las características necesarias para poder implementar los algoritmos de detección. Se pretende incorporar en el nodo sensorial un sistema operativo que administre los diferentes algoritmos, esto para poder hacer mediciones de eficiencia y consumo de potencia.

1.2 Solución seleccionada

A través del desarrollo del proyecto se realizó una investigación previa donde se estudiaron diversos nodos sensoriales y sus características. Luego se procedió a investigar las características de los algoritmos desarrollados por la escuela y las necesidades de hardware y software de estos. En síntesis se determinó que el procesador del nodo sensorial requerido por la escuela sería un procesador de 32 bits, con arquitectura RISC, de la marca ARM, modelo ARM7TDMI y versión v4. Una vez determinado el procesador se procedió a escoger los periféricos deseados para el procesador, entre ellos convertidores analógicos digitales, puertos E/S y puertos para implementar protocolos de comunicación, por ello se decidió en escoger un microcontrolador con un procesador ARM7 marca Philips el cual se encontraba en una tarjeta de desarrollo modelo LPC L2294, en esta tarjeta se hicieron pruebas para manejar los diferentes puertos de entrada y/o salida, además de otras características propias. Como última etapa se logró instalar un sistema operativo de tiempo real y hacer pruebas sobre manejo de tareas y administración de prioridades sobre ellas.

Capítulo 2: Meta y Objetivos

2. Meta

Determinar el estado del arte para el microprocesador con el que se debería desarrollar un nodo sensorial para la implementación de la red para la protección del ambiente.

2.1 Objetivo general

Determinar el núcleo del microprocesador/microcontrolador con el que se pueda desarrollar un nodo sensorial capaz de cumplir los requerimientos planteados para la ejecución de los algoritmos necesarios para la implementación de la red de protección ambiental.

Indicador: sistema empotrado con al menos un protocolo de comunicación, un convertidor analógico digital, y un sistema operativo.

2.2 Objetivos específicos

1. Determinar el conjunto de características requerido para que un microprocesador pueda funcionar como unidad central de proceso de un nodo sensorial.

Indicador: Lista de características válidas para un procesador de un nodo sensorial.

2. Seleccionar un microprocesador comercial con las características necesarias para llenar los requerimientos para el nodo sensorial planteado.

Indicador: Tabla comparativa con diferentes procesadores, características eléctricas y set de instrucciones.

3. Elegir una plataforma de desarrollo para probar las aplicaciones necesarias para el nodo en un sistema empotrado.

Indicador: Tabla comparativa con diversas plataformas de desarrollo para las aplicaciones del sistema empotrado.

4. Seleccionar y evaluar un sistema operativo empotrado capaz de administrar los algoritmos necesarios para la operación del nodo de la red de sensores del bosque.

Indicador: conexión del sistema operativo con todos periféricos del nodo sensorial.

Capítulo 3: Marco teórico

3.1 Redes de sensores y actuadores inalámbricos (WSAN)

Las redes de sensores y actuadores inalámbricos (WSAN) son grupos de sensores y actuadores asociadas a través de dispositivos de comunicación inalámbricas [10]. Los sensores permiten obtener información del mundo físico, mientras que los actuadores toman decisiones y realizan acciones sobre el ambiente donde se encuentran. Estas redes permiten a los usuarios sensar y actuar en el medio, a distancia. Para tener mecanismos efectivos en la medición con sensores y las acciones realizadas por los actuadores se deben poseer métodos de coordinación entre los sensores y actuadores. Y más allá para llevar acabo las acciones correctas los actuadores deben poseer información veraz y actualizada en el momento justo de actuar.

Los avances en la tecnología han hecho posible poseer redes distribuidas de sensores y actuadores capaces de observar el mundo físico, procesar datos y realizar acciones apropiadas según la información obtenida. Estas redes pueden ser parte de sistemas tales como detectores en campos de batalla, control de microclimas, control en plantas nucleares, detección ante ataques biológicos o químicos, automatización de hogares (domótica) y monitoreo ambiental.

Un sistema con una unidad proveedora de energía, una unidad de procesamiento, una unidad de transmisión – recepción inalámbrica y un actuador o un sensor se le conoce como nodo. Cuando la unidad posee una unidad de procesamiento y un sensor se le conoce como nodo sensorial.

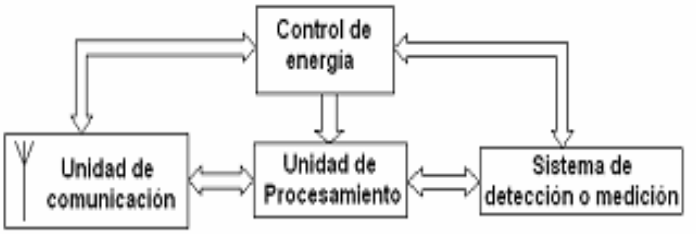

Figura 1 Nodo sensorial básico

En una WSAN los nodos con sensores son de bajo coste, de bajo consumo energético, con capacidad de procesamiento y comunicación inalámbrica. Por su parte los nodos con actuadores son ricos en recursos, equipados con mayores capacidades de procesamiento, potencias de transmisión mayores y mejor disponibilidad de energía.

En síntesis los nodos con sensores miden y transmiten la información, los nodos con actuadores procesan la información recibida y realizan acciones según el análisis realizado sobre la información. Por supuesto es posible tener unidades equipadas con ambos, sensor y actuador.

Las redes de sensores inalámbricas se caracterizan por su facilidad de despliegue y por ser autoconfigurables. En cieOS contextos se le llama adaptabilidad, que es la capacidad de adaptarse a las diferentes situaciones.

Existen diversos tipos de redes de sensores. Dependiendo de la aplicación pueden existir redes con cientos o miles de nodos. Los nodos incluso pueden tener el tamaño de un grano de arena [50].

En un contexto muy general cuando se tiene una red de nodos sensoriales cada nodo sensorial o nodo fuente de una red de sensores se encarga de obtener información del medio, procesarla y retransmitirla a una estación base de forma aleatoria, como se muestra en la figura 2.

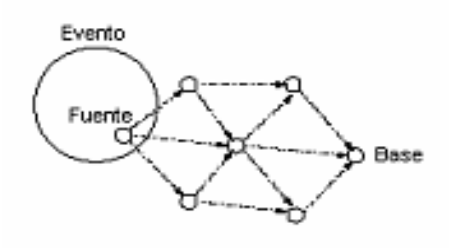

Figura 2 Funcionamiento de una red de sensores

3.2 Antecedentes

En la Escuela de Ingeniería Electrónica se ha trabajado en el diseño e implementación de las diferentes etapas de la red de sensores. A continuación un resumen de los avances obtenidos hasta el momento en el proyecto de protección ambiental:

3.2.1 Fuego

En [23] se ha trabajado en la detección de fuego a través de cámaras digitales.

3.2.2 Tala y disparos

Para la detección de tala de árboles y de disparos se han desarrollado cuatro proyectos, primero en [24] se trabajó en el diseño de un sensor de motosierras. En éste trabajo se realiza la recepción de la señal de audio producida por la motosierra a través de un micrófono. En una etapa posterior se procede a analizar si la señal recibida es o no proveniente de una motosierra. Si se detecta la presencia de una motosierra se procede a indicar la presencia de ésta en el lugar. Este prototipo se desarrolló de forma analógica. De manera similar un segundo trabajo [25] se dedicó a la detección de disparos, siguiendo un procedimiento semejante al realizado en [24].

En [26] se trabajó en la detección acústica de motosierras y disparos. Específicamente se realizó el reconocimiento de patrones acústicos con la teoría de wavelets. En [27] se continuó el trabajó realizado en [6], mejorando los clasificadores y las características empleadas en los procesos de reconocimiento.

3.2.3 Módulos de Comunicación

Como el proyecto "Red inalámbrica de telecomunicaciones para la protección del ambiente" consiste en una red de sensores inalámbrica se tuvo que incursionar en el diseño de la comunicación inalámbrica entre los sensores de la red. En [28] y [29] se ha trabajado en esta sección del proyecto. En estos trabajos se ha logrado establecer el protocolo de comunicación para que un microcontrolador reciba las señales de los sensores de forma inalámbrica, y con esto se logra indicar en un sumidero o punto central la presencia de un daño ambiental.

En [28] se dedicó a la fase de comunicación de la red de tele-detectores en el bosque, para esto se utilizó un microcontrolador de la empresa microchip (PIC16LF874A) y transceptores AC4490 – 1000M de la compañía Aerocomm, con los cuales desarrolló los protocolos de las comunicaciones y el manejo de los datos.

En [29] se trabajó en el diseño e implementación de un protocolo de comunicación para una red de sensores que permite el monitoreo de variables diversas en el bosque, se buscó mejoras en el diseño anterior. Para este proyecto se utilizaron los transceptores AC4490 – 1000M de la compañía Aerocomm y un microcontrolador PIC 16F877. Con estas herramientas se desarrolló el protocolo para la interconexión de los diferentes sensores (fuego, sierras, disparos).

3.3 Investigación previa

Para determinar cuales son las características mínimas con las que debe contar el procesador del nodo sensorial desarrollado por la Escuela de Ingeniería en Electrónica se procedió a la investigación de las características de los algoritmos con que cuentan los sensores desarrollados hasta el momento (fuego, disparos, motosierras) y sus diferentes necesidades de hardware.

Al realizar la investigación se determinó que las características de mayor importancia son capacidad de procesamiento, consumo de potencia, manejo de al menos un sistema operativo, implementación de protocolos estándares de comunicación, convertidores analógicos digitales y tamaño en memoria de los algoritmos.

3.3.1 *Conceptos básicos*

Antes de mencionar las características mínimas con las cuales debe de contar la unidad de proceso central del nodo sensorial, se presenta una explicación de algunos factores que influyen en el desempeño y consumo de potencia de un procesador.

3.3.2 Rendimiento de un procesador

El rendimiento de un procesador se ve afectado por aspectos tales como su frecuencia de reloj, tamaño de lo buses de datos, el número de bits de la ALU[1], arquitectura (RISC[2], CISC[3]) [1], cantidad de registros, memoria caché, set de instrucciones y otros aspectos del diseño.

Existen procedimientos para medir el rendimiento de un procesador, conocidos como “benchmarks”. Estos emplean diferentes unidades de medida, entre ellas MIPS, MIPS relativos, MFLOPS, y otros [2, 7, 20].

3.3.3 Buses de datos y memoria caché

El tamaño de los buses de datos, la velocidad máxima de transmisión en el bus de datos y la velocidad de la memoria principal afectan el ancho de banda[4] [3, 7] de un procesador y por ende su rendimiento. Los procesadores tienen buses de datos

---

1 ALU = unidad lógica aritmética

2 RISC = computadora con set de instrucciones reducido.

3 CISC = computadora con set de instrucciones complejo.

4 Ancho de banda = frecuencia de procesador X tamaño del bus de datos.

internos y buses de datos externos. Por factores físicos los buses de datos internos son más veloces que los buses externos y que la memoria principal. Para compensar estas diferencias de velocidades se utilizan varias técnicas de balance de ancho de banda. Entre ellas está la manipulación de la frecuencia de operación del procesador que no siempre es posible utilizarla, otra técnica es incluir estados de espera entre el procesador y la memoria principal, y la más utilizada es la implementación de memorias cachés que hacen que el bus de datos externo y la velocidad de la memoria principal sea invisible para el procesador [7]. Las dos primeras disminuyen de forma disminuyen de forma significativa el rendimiento de un procesador [4, 7, 51].

Hay procesadores que manejan diferentes tamaños en los buses de datos internos y buses externos. Un caso específico es el procesador 8088 de INTEL que tiene un bus de datos interno al procesador de 16 bits y un bus externo de 8 bits. Esto influye en el rendimiento del procesador ya que para realizar la lectura de un dato de 16 bits el procesador tiene que hacer dos lecturas de memoria, lo que implica gasto de tiempo [4,19] y una disminución del ancho de banda [3].

El tamaño de los buses de datos también afecta la funcionalidad de un procesador y el rango de los números a manejar [7, 19]. Por ejemplo el procesador 80286 con su set de instrucciones de 32 bits logró implementar nuevas instrucciones de dos operandos fuentes y uno de destino, acción que no era posible realizar en sus predecesores de 16 bits [4, 19]. En otras familias de procesadores como ARM sucede algo similar: existen diferencias en tiempos de ejecución y funcionalidades con sus juegos de instrucciones de 16 bits (Thumb) y de 32 bits [5].

3.3.4 ALU

La cantidad de bits de la ALU afecta el rendimiento de un procesador. Por ejemplo el 8086 de INTEL con ALU de 16 bits tarda más en realizar operaciones en datos de 32 bits de lo que tarda un procesador de 32 bits como es el caso de 80386 [4, 7, 19], esto por varias razones como la frecuencia de operación, el tamaño de bits de la

ALU, el procesador 80386 tiene un bus de datos de 32 bits, a una mayor frecuencia y tiene una ALU de 32 bits.

3.3.5 Capacidad de direccionamiento

Un procesador maneja una cantidad determinada de memoria externa. El manejo de la memoria externa lo hace a través de su bus de direcciones, es por eso que a medida que crecen los buses de direcciones de los procesadores crece su capacidad de direccionamiento. Por ejemplo un procesador con un bus de direcciones de 16 bits puede manejar 64 kB de memoria externa, y un procesador de 32 bits de direcciones puede manejar 4 GB [4,19].

3.3.6 Consumo de potencia

El consumo de potencia de un procesador se ve afectado por la frecuencia de reloj [6], arquitectura (CISC o RISC) [7], cantidad de transistores, modos de ahorro de energía (stand by, idle, sleep, halt y otros), buses de datos. Hablando en forma más general, por ejemplo en un sistema mínimo (procesador, memoria y periféricos) influyen aspectos como el tamaño de los programas a ejecutar, los compiladores, eficiencia de los sistemas operativos [8] y otros.

3.3.7 Arquitecturas RISC Y CISC

Los procesadores se diseñan para que adopten una de dos arquitecturas. Estas dos arquitecturas son CISC y RISC. La primera se le conoce como computadora con set de instrucciones complejos (CISC). En ésta la mayoría de instrucciones tardan más de un ciclo de reloj en ejecutarse y requiere de un hardware suficientemente capaz de ejecutar las instrucciones. La segunda se le conoce como computadora con set de instrucciones reducido (RISC), en esta la meta es que las instrucciones se ejecuten en un ciclo de reloj máximo y está orientado a un software eficiente con un hardware menos complejo que el del CISC [11].

Los términos complejo y reducido van más allá del set de instrucciones, estos términos también consideran la complejidad del hardware del procesador [12].

Con tecnologías de semiconductores comparables e igual frecuencia de reloj, un procesador RISC típico tiene una capacidad de procesamiento de dos a cuatro veces mayor que la de un CISC. La estructura de un procesador RISC se puede realizar en una fracción de la superficie ocupada por el circuito integrado de un procesador CISC [11,12].

Los procesadores RISC tienden a desplazar a los procesadores CISC, aunque no es proceso tan simple ya que para aplicar una determinada arquitectura de microprocesador son decisivas las condiciones de realización técnica y sobre todo la rentabilidad, incluyendo los costos de software, este último aspecto en un procesador RISC tiene grandes costos. Existen razones de compatibilidad para desarrollar y utilizar procesadores de estructura compleja (CISC) así como un extenso conjunto de instrucciones [11,12].

En investigaciones hechas a mediados de la década de los setentas, con respecto a la frecuencia de utilización de una instrucción en un procesador CISC y al tiempo utilizado en su ejecución, se observó que alrededor del 20% de las instrucciones ocupa el 80% del tiempo total de ejecución de un programa. En este punto nació la idea de diseñar procesadores RISC, procesadores con pocas instrucciones, muy potentes y con un hardware muy sencillo [11,12, 14].

La relativa sencillez de la arquitectura de los procesadores RISC conduce a ciclos de diseño más coOS cuando se desarrollan nuevas versiones, lo que posibilita la aplicación de las más recientes tecnologías de semiconductores. Por ello, los procesadores RISC no solo tienden a ofrecer una capacidad de procesamiento del sistema de dos a cuatro veces mayor, sino que los saltos de capacidad que se producen de generación en generación son mucho mayores que en los CISC [11,12, 14].

Ahora bien, si se quisiera comparar un procesador CISC con uno RISC tratando de

realizar una tarea en particular, por ejemplo para la multiplicación “a = a * b”, siendo a y b datos en memoria principal, la instrucción para el procesador CISC sería:

Mult a,b

Para el procesador RISC sería:

LOAD REG1, a

LOAD REG2, b

Prod REG1, REG2

Store a, REG1

Una de las principales ventajas del sistema CISC es que el compilador tiene que hacer muy poco trabajo trasladando los programas a lenguaje ensamblador. El tamaño del código es relativamente pequeño, necesitan poca memoria para almacenar las instrucciones. El énfasis es poner a ejecutar las instrucciones directamente en el hardware, conforme aumentan el tamaño del set de instrucciones crece la complejidad del hardware [13].

A primera vista se diría que el procesador RISC es mucho menos eficiente por la forma que realiza la multiplicación. Se necesita más memoria para ensamblar la multiplicación “a = a * b”, además el compilador requiere realizar un trabajo fuerte para convertir la instrucción de alto nivel a lenguaje ensamblador. Sin embargo, la estrategia del procesador RISC brinda algunas ventajas importantes. Primero cada instrucción requiere de solo un ciclo de reloj para ejecutarse, luego el programa puede ejecutarse aproximadamente en el mismo tiempo que se ejecuta la instrucción “mult a, b” del procesador CISC. El set de instrucciones requiere menos hardware complejo para ejecutarse, lo que deja más espacio para los registros de propósito general. Como todas las instrucciones se llevan a cabo en un ciclo de reloj la segmentación es posible [13].

Lo que se logra en realidad con separar las instrucciones "LOAD" y "STORE" es

reducir la cantidad de trabajo que la computadora debe realizar [7, 11,13].

A continuación un resumen de las arquitecturas CISC y RISC:

Tabla 1 Características de las arquitecturas CISC y RISC*

Arquitectura	Orientación de la arquitectura	Ciclos por instrucción	Movimiento de datos	Tamaño de códigos	Uso de los transistores	Uso de memoria en tiempo de ejecución	Peso en la ejecución de programa
CISC	Se centra en mejoras al hardware, esto para que ejecuten las instrucciones complejas.	Las instrucciones tardan varios ciclos de reloj ejecutándose.	Está orientada a movimientos de datos Memoria – Memoria: "LOAD" y "Store" están incorporadas en las propias instrucciones.	Códigos de tamaño pequeño.	Los transistores son usados para el hardware encargado de ejecutar las instrucciones complejas.	Menos memoria RAM para ejecución de programas.	El procesador tiene un trabajo fuerte en la ejecución de las instrucciones.
RISC	Se centra en mejoras al software. Se tienen pocas instrucciones muy potentes. El hardware que no se invierte en ejecutar instrucciones complejas, se invierte en construir registros internos.	Las instrucciones se ejecutan en un ciclo de reloj	Orientado a movimientos de datos Registro – Registro: "LOAD" y "STORE" son instrucciones independientes.	Tamaños de códigos grandes.	Los transistores son usados para tener muchos registros.	Más memoria RAM para ejecución de programas.	El compilador tiene un trabajo fuerte en la traducción de los programas.

* = Tabla tomada de [13].

Con certeza no se puede decir cuál de las dos arquitecturas brinda el mejor desempeño. Se deben estudiar casos específicos de comparación. Los procesadores RISC han ganado terreno en el mundo comercial, especialmente en los dispositivos hand held (portátiles), debido al bajo consumo de potencia y capacidad de procesamiento [7, 11, 12, 13, 21].

3.3.7.1 Ecuación de desempeño

La siguiente ecuación es usada para expresar el desempeño de una computadora:

$$T_{programa} = N_{instrucciones} * CPI * T_{clk}$$

La ecuación anterior, indica que el tiempo en que se ejecuta un programa está dado por la cantidad de instrucciones de un programa ($N_{instrucciones}$), la cantidad de ciclos promedio por instrucción (CPI) y el tiempo de un ciclo de reloj (T_{clk}) [7,14].

Para un procesador CISC se intenta minimizar el número de instrucciones en el programa, y para el procesador RISC se busca reducir la cantidad de ciclos por instrucción, aunque esto genera un costo en la cantidad de instrucciones del programa.

3.4 Requerimientos del sistema: características de nodo sensorial

Una vez que se han descrito algunas características con las que cuentan los procesadores con respecto a capacidad de procesamiento y consumo de potencia, se pueden mencionar cuáles características específicas son deseadas para la unidad de proceso central del nodo sensorial de la red de protección al ambiente.

a) El procesador que se escoja para el nodo sensorial debe ser de 32 bits, esto para poder ejecutar alguno de los sistemas operativos existentes para sistemas empotrados de 32 bits como uCLinux, uCos II, IAR PowerPac, OS, RTLinux, Partikle, eCOS y otros. La idea de tener un procesador de 32 bits es poder acelerar el procesamiento de los algoritmos y hacer un uso más

eficiente de los recursos del nodo sensorial.

b) Se ha determinado a través del estudio de los procesadores con arquitectura CISC y RISC, que la mejor opción para el nodo sensorial es tener un procesador tipo RISC, los cuales, aunque necesitan más memoria de programa, utilizan de forma más eficiente los recursos energéticos y son de mayor uso en dispositivos portátiles.

c) El nodo sensorial no todo el tiempo debe estar en actividad, esto para ahorrar energía, por tanto el procesador debe de tener al menos un modo que le permita al sistema ahorrar energía (stand by, sleep, idle u otros).

d) El procesador del nodo sensorial debe interfazarse con algún módulo de comunicación estándar como Zigbee, IEEE 800.12 u otros. Es por eso que el procesador del nodo sensorial debe tener la capacidad de implementar algún protocolo de comunicación estándar (TCP/IP, RS232, DH-485, etc.) que se comunique con módulos externos de transmisión y recepción [9, 16, 17].

e) El procesador del nodo sensorial debe contar con al menos un convertidor analógico digital (ADC) o con un periférico que realice esta función. Esto porque en el nodo están presentes transductores que convierten señales físicas en señales eléctricas, que posteriormente se deben procesar a través de algoritmos digitales [10]. Se desea muestrear señales acústicas con frecuencias máximas de 11kHz [15] y 8 bits mínimo.

f) Los sistemas operativos que existen para sistemas empotrados de 32 bits ocupan de 12 kB a 350 kB de memoria no volátil [16,17]. En algunos casos. Otros como uCLinux incluso necesitan 2 MB de memoria no volátil y los algoritmos desarrollados ocupan como mínimo 2 MB de memoria no volátil [18], por esto el procesador debe tener la capacidad de direccionar más de 2 MB de memoria.

g) La unidad de proceso central puede opcionalmente contar con una unidad de puntoflotante.

Tabla 2 Requerimientos mínimos del nodo sensorial

Tipo de procesador	**Número de bits**	**Modos de manejo de energìa**	**Posibles protocolos de comunicación**	**Convertidores analógicos digitales**	**Capacidad de direccionamiento**
RISC, puede o no contar con unidad de punto flotante.	32	Entre ellos idle, stand by, sleep, etc.	SPI, RS232, Ethernet, I2C	Con capacidad de muestrear al menos audio a 11 kHz	Mínimo 2 MB para ejecutar un algoritmo o guardar muestras de este.

3.5 Escogencia de unidad de proceso central

En el mercado existen familias de procesadores RISC con núcleos de 32 bits. Las diferentes familias presentan características con respecto a arquitectura, consumo de potencia, modos de ahorro de energía, set de instrucciones, manejo de periféricos, capacidad de procesamiento (MIPS, MFLOPS, etc.), y otras características.

Para el diseño de este nodo sensorial el principal parámetro a tomar en cuenta para la escogencia del procesador del nodo sensorial es el consumo de potencia. No se debe sobreponer la capacidad de procesamiento al consumo de potencia. Al realizar un estudio de las principales familias de procesadores de 32 bits RISC se ha decidido que el procesador del nodo sensorial debe ser de la familia de procesadores ARM, los cuales son los procesadores de 32 bits RISC de menor consumo de potencia en el mercado y de mayor uso en aplicaciones portátiles. A continuación una tabla comparativa con el consumo de potencia de diversas familias de procesadores RISC de 32 bits.

Tabla 3 Familias de procesadores RISC de 32 bits, consumo de potencia [22, 29, 30, 31, 32, 33, 34, 35].

Marca	Procesador	Potencia mínima de consumo (W)
Sun systems	SPARC	2.5 (lo mínimo en microSPARC I)
IBM	PowerPC	13 – 27, PowerPC
DEC	Alpha	9 en LCA4
MIPS technologies	MIPS	4 en R3000
HP	PA-RISC	7 en PCX-L
ARM	ARM	25 mW en ARM7TDMI

5.5.1 Procesadores ARM

Los procesadores RISC de 32 bits más utilizados en el mercado son de la marca ARM (Advanced Risc Machine[5], originalmente ACORN RISC MACHINE). Se dice que un 75% de los dispositivos portátiles (hand held) en el mundo utilizan los procesadores ARM, esto por su bajo consumo de potencia y eficiente set de instrucciones (RISC) [21, 22].

La implementación de la familia de procesadores ARM más exitosa ha sido ARM7TDMI, con miles de millones vendidos. ARM7TDMI ha sido especialmente utilizada en los reproductores de música y video digital iPod de Apple [21].

La arquitectura más común soportada en teléfonos inteligentes (smart phones), PDA's (Asistente Digital Personal) y otros dispositivos hand held es la ARMv4 (ARM7TDMI es arquitectura ARMv4). También la versión ARMv4 se ha utilizado en el patentado StrongARM de Intel [21].

[5] ARM = computadora avanzada con arquitectura RISC.

Los procesadores ARM no alcanzan altas temperaturas y tiene bajos requerimientos de energía [21].

Acorn Computers actualmente solo da soporte para los procesadores de las familias ARM7, ARM9, ARM10 y CORTEX [21,22].

3.5.2 Escogencia final

Al observar la tabla 3 se ha decidido por los requerimientos de ahorro de energía del nodo sensorial que el procesador del nodo sensorial debe ser un núcleo de marca ARM. ARM tiene núcleos que implementan sistemas operativos de 32 bits como IAR PowerPac, uCLinux, Partikle, eCos, uCos II, Windows CE, RTLinux, OS y otros. Los núcleos ARM son de muy bajo consumo de potencia (en el orden de los mW). Al ser orientados a mercados de microcontroladores los fabricantes integran convertidores digitales analógicos y protocolos de comunicación en un solo paquete microcontrolador. Por estas características los procesadores ARM son ampliamente utilizados en aplicaciones portátiles y sistemas empotrados, de allí que la unidad de proceso central de nodo sensorial desarrollado por la escuela de ingeniería electrónica tendrá un procesador ARM.

En la tabla 4 [22] se puede observar un resumen con las principales características de las familias de procesadores ARM. Al observar esta tabla se puede notar como la familia ARM7 es la de menor consumo de potencia, por esta razón la unidad de proceso del nodo central debe ser de la familia ARM7.

Tabla 4 Síntesis características núcleos ARM.

Núcleo	VERSIÓN DEL NÚCLEO ARM	ARQUITECTURA	RENDIMIENTO (MIPS/MHz)	MIPS PROMEDIO	THUMB	JAZELLE	DSP	MMU (unidad de manejo de memoria)	CACHÉ	FPU (unidad de punto flotante)	APLICACIONES
ARM7	V4 (es a la versión más antigua que se le da soporte actualmente)	Pipeline de 3 etapas	0.9	130	SÍ	ARM7EJ-S	ARM7 EJ-S	ARM720T	NO	no	Nintendo game boy advance, iPOD, Acorn RISC, PC 70, Psion 5.
ARM9	v4	Hardvare con pipeline de 5 etapas	1.05	300	SÍ	no	no	SÍ	SI	no, aunque puede interfazase con VFP9-S	
ARM9E	v5	Hardvare con pipeline de 5 etapas,Coprocesador opcional VFP9, puede alcanzar 215 MFLOPS	1.1	300	SÍ	SÍ, 8x (ARM926 EJ-S).	SÍ, dsp mejorado	SÍ, en algunos casos mejor MPU	SI	no, aunque puede interfazarse con VFP9-S	Nintendo DS, Nokia N-Gage.

Tabla 4 Continuación Síntesis características núcleos ARM

Núcleo	Versión del núcleo ARM	Arquitectura	Rendimiento (MIPS/MHz)	MIPS promedio	THUMB	JAZELLE	DSP	MMU	CACHÉ	FPU (unidad de punto flotante)	Aplicaciones
ARM10	v5	Hardvare con pipeline de 8 etapas	1.25	400	SÍ	SÍ	SÍ, dsp mejorado	MMU Y MPU (unidad de manejo de energía)	SI	No Existe la posibilidad de interfazarse con VF10 logra hasta 650 MFLOPS	
ARM11	v6	Hardvare con pipeline de 8 etapas	DEPENDE, YA QUE HAY SISTEMAS MULTIPROCESADOR	740	Thumb 2 mejora al thumb	SÍ	SÍ, dsp mejorado	MPU Y MMU	SI	Coprocesador integrado de punto flotante, ideal para aplicaciones 3D ARM1176JZ(F)-S, ARM1156T2-S	Procesadores multimedia OMAP de Texas Instruments

Tabla 5 Modos de ahorro de energía procesadores ARM.

Núcleo	Consumo de potencia (mW/MHz)	Rango de frecuencias de operación (MHz)	Modos de manejo de potencia	Notas	
ARM7	(0.05-0.20) en procesos de 0.13 um	50-133	El núcleo como tal no implementa modos de manejo de potencia, pero las marcas especializadas si lo hacen. Se mencionan idle, stand by y otros.	Microcontroladores de marcas como Atmel, STR, Philips LPCxxxx. Por ejemplo AT91SAM7, lh79520. El núcleo ARM 7 es ampliamente distribuido por múltiples fabricantes.	Son comunes stand by e idle. El modo idle apaga el procesador y algunos periféricos y espera regresar a operación con una interrupción.
	(0.25-0.65) en procesos de 0.18 um	75-110			

Tabla 5 Continuación Modos de ahorro de energía procesadores ARM.

Núcleo	Consumo de potencia (mW/MHz)	Rango de frecuencias de operación (MHz)	Modos de manejo de potencia	Notas	
ARM9	0.8 en procesos de 0.18 um	200	Ibidimen	AT91RM9200	
	0.25-0.26 en procesos de 0.13 um	250-258		STR91XF	
ARM9E	0.11 - 0.36 en procesos de 0.13 um	180-276			
	0.06 - 0.24 en procesos de 90 nm	230-530			
ARM10	1.05 con caché y 0.6 sin cachè en procesos de 0.13 um	266	Ver tabla 6		
	0.45 con caché y 0.40 sin cachè en procesos de 90 nm	280-540			
ARM11	0.37-0.42 sin caché y 0.43 - 0.51 para procesos de 90 nm	620			

Tabla 6 Instrucciones para ahorro de energía en procesadores ARM 10 y ARM11.

Modo	**Suceso en el núcleo**	**Memoria**	**Potencia**	**Forma de entrar al modo**	**Salida a estado normal de operación**
Run	clocks on, power on	clocks on, power on	Depende de la aplicación		
Stand by (parecido a idle)	clocks off, power on	clocks off, power on	Fugas estáticas	Espera por interrupción (instrucción)	Interrupción: requerimiento de depuración (1 us)
Dormant	clocks off, power off	clocks off, power on	Fugas estáticas de los arreglos de memoria	Deshabilita interrupciones, salva los registros de estado, drena buffer de escritura, solicitud de dormant.	Reset suave de al menos 100ms
Shut down	clocks off, power off	clocks off, power off	Cero	Deshabilita interrupciones, salva los registros de estado, drena buffer de escritura, solicitud de shut down.	Reset por sistema operativo mayor a 500 ms

3.6 Tarjetas de desarrollo

En el mercado existen diversas tarjetas de desarrollo que incorporan microcontroladores con núcleos ARM de la familia ARM7. En la mayoría de casos se encuentran los núcleos ARM7TDMI-S[6], estas pueden encontrarse en sitios como Web como www.lpctools.com, www.sparkfun.com, www.olimex.com, www.keil.com, www.iar.com y otros. Las tarjetas de desarrollo incorporan microcontroladores de diferentes marcas como ATMEL, Philips NXP, STR microelectronics, Analog Devices, entre otros. Cada tarjeta está orientada a diversos fines como aplicaciones de control industrial, comunicaciones industriales, sistemas embebidos, aplicaciones gráficas y otras. Al hacer un estudio de las tarjetas de desarrollo se determinó que las más aptas para llevar a cabo el proyecto son las que incorporan memoria FLASH y SRAM externas ya que evitan los montajes en protoboard de memorias externas, son una buena opción si lo que se desea es ganar tiempo en el desarrollo de las aplicaciones. Además, esta memoria adicional sirve para montar los algoritmos de detección desarrollados, así como sistemas operativos que requieran más memoria de la que tienen empotrada los microcontroladores o simplemente para guardar la información proveniente de los sensores para que otro dispositivo como un FPGA pueda leer estos datos y procesarlos.

En la tabla 7 [36] se pueden ver dos tarjetas de desarrollo que incorporan memoria flash y RAM externa, que son las que se apegan más a características deseadas en el nodo sensorial. Ambas tarjetas poseen procesadores LPC2294 de la marca NXP Philips, tienen la característica de poder manejar 4 bancos de memoria externos de 64 MB cada uno, en modos de 8, 16 o 32 bits.

El microcontrolador LPC2294 incorpora los siguientes modos de ahorro de energía:

- Idle: Detiene procesador, pero no los periféricos. El procesador reanuda la

[6] TDMI-S, T = instrucciones Thumb, d = depuradora incorporado, M = multiplicador por hardware de 32 X 32 bits, I = ICE interno, S = procesador modelado utilizando VHDL.

ejecución de programa a través de una interrupción.

- Sleep: Detiene el reloj de los periféricos y el procesador, el consumo de potencia es muy bajo, las salidas se mantienen en su último estado, los valores de la memoria SRAM son preservados, queda activa la salida del IRC (oscilador interno RC), no se detiene el RTC (reloj de tiempo real), esto porque las interrupciones que funcionan a través del RTC son una de las opciones para reanudar la operación del procesador.
- Power Down: similar al modo sleep pero se apaga el IRC y la memoria flash.

Tabla 7 Tarjetas de desarrollo con memoria externa incorporada [36].

KIT	Fabricante	Procesador / microcontrolador	Memoria Interna		Memoria Externa		Periféricos	Precio ($)	Modos ahorro energía	EMC (external memory controller)	Interfaces de programación
			Flash (kB)	RAM (kB)	Flash(MB)	RAM(MB)					
LPC L2294	Olimex	ARM7TDMI-S 16/32 bits	256	16	2 ó 4	1 ó 8	RTC, RS232 (2 x UART), CAN, SD/MMC, 4 x 10 bits ADC 2.44 us, Ethernet (CS8900A)I2C bus 400 kbit/s, 2 x SPI, 2 x 32 bits Timers, 7 x CCR, 6 x PWM, VIC, WDT (perro guardián)	1MB $100	Idle	Sí. Hasta 4 bancos externos de 16 MB c/u	ISP
											IAP
								8MB $232	Power down		JTAG

Tabla 7 continuación. Tarjetas de desarrollo con memoria externa incorporada.

KIT	Fabricante	Procesador / microcontrolador	Memoria Interna		Memoria Externa		Periféricos	Precio ($)	Modos ahorro energía	EMC (external memory controller)	Interfaces de programación
			Flash (kB)	RAM (kB)	Flash(MB)	RAM(MB)					
LPC H2294	Olimex	ARM7TDMI-S 16/32 bits	256	16	4	1	RTC, ISB 2 RS232 , CAN, 4 x 10 bits ADC 2.44 us,I2C bus 400 kbit/s, 2 x SPI, 2 x 32 bits Timers, 7 x CCR, 6 x PWM, VIC, WDT (perro guardián)	$77	Idle Power down	Sí. Hasta 4 bancos externos de 16 MB c/u	ISP IAP JTAG

3.7 Plataformas de desarrollo

En el mercado existen diversas plataformas de desarrollo que ofrecen a desarrolladores o principiantes una gran variedad de posibilidades para poder llevar a cabo proyectos basados en procesadores ARM. Algunas de estas plataformas son sin fines de lucro y conllevan mayor tiempo de aprendizaje para su uso, brindan todas las herramientas necesarias para el desarrollo y pruebas de aplicaciones y sin restricción en el tamaño de códigos. Existen otras plataformas con valores desde $900 y ofrecen ambientes gráficos amigables con el usuario, con menús y barras de herramientas de fácil uso, brindan la opción de utilizar versiones de evaluación para desarrollar aplicaciones de hasta 32 kB y son altamente difundidos.

A continuación se presenta un resumen las plataformas más utilizadas en el mercado, tanto las no gratuitas como las de software libre y sus características.

3.7.1 IAR WORKBENCH

Es una completa plataforma de desarrollo, que se ejecuta en el sistema operativo Windows en las versiones 98SE/ME/NT4/2000/XP [37].

Características:

- Permite desarrollar aplicaciones para todos los dispositivos con núcleos ARM (ARM7, ARM9, ARM9E, ARM10, ARM11, SecureCore, Intel Xscale).
- Incorpora las definiciones de los registros de función especial (SFR's) para los dispositivos de las marcas Analog Devices, Atmel, Cirrus logic, Freescale, Intel, NetSilicon, OKI, Philips (NXP), Samsung, sharp, ST microelectronics y Texas instruments.
- Incluye proyectos de ejemplo para la mayoría de dispositivos y tarjetas de evaluación.
- EWARM es un compilador altamente optimizado de C/C++:

- Soporta C, EC++ y EC++ extendido.
- Incluye templates, namespaces, librería estándar (STL), etc.
- Incluye chequeo automático de reglas MISRA C.
- Extensiones de lenguaje para aplicaciones embebidas con soporte para dispositivos específicos:
    - Palabras claves extendidas para definición de datos/funciones y declaración de atributos con memoria/tipos, tales como __irq, __fig, __arm, __thumb, etc.
    - Directivas pragma para controlar el comportamiento del compilador.
    - Funciones intrínsecas para el acceso directo a código fuente C para operaciones del procesador de bajo nivel, tales como la generación de instrucciones coprocesador de lectura/escritura (MCR/MRC).
- Soporte para el coprocesador de punto flotante VFP9-S.
- Soporte para aplicaciones de 4 GB en modos ARM y Thumb.
- Soporte para números long long de 64 bits (QWORD).
- Código reeentrante.
- Datos de punto flotante en formato IEEE de 32 y 64 bits.
- Múltiples niveles de optimización en códigos y velocidad de ejecución permitiendo diferentes transformaciones con funciones inlining, loop unrolling, etc.
- Optimizador avanzado global y de aplicación específica que genera el código más estable y compacto.

- Depurador (STATE OF THE ART C-SPY):
    - Puntos de ruptura (Breakpoints) para datos y código.

- Análisis en tiempo de ejecución para pilas, con un monitor que permite determinar el consumo de pilas y su integridad.
- Depuradora multi - núcleo vía IAR J-Link.
- Herramientas de análisis para cobertura de desempeño en códigos y perfiles.
- Utilidad TRACE con expresiones como variables y valores de registros que permiten examinar la historia de ejecución.
- Monitoreo versátil de registros, estructuras, llamadas en cadena, registros de periféricos, variables locales y globales.
- Simulación de entradas y salidas (I/O) e interrupciones.
- STL inteligente desplegado en una ventana independiente.
- Edición en tiempo de ejecución.
- Modelo de arrastre y posicione (Drag and drop).

- Sistema de depuración en dispositivos con las siguientes interfaces:
  - Simulador.
  - Emulador (JTAG):
    - IAR J-Link conexión USB o servidor TCP/IP.
    - RDI (Remote depuradora interfase), con interfaces como BD11000 y BD12000, EPI Majic, Ashling Opella, AIDI Open ICE.
    - Interfaces Macraigor JTAG: macraigor Raven, Wiggler, mpdemon y USBDemon.
  - ROM monitor:
    - Monitor IAR ROM para kit IAR KickStart con tarjetas de evaluación LPC210X, Analog Devices y OKI.
    - Monitor para depuradora Ángel para tarjetas de evaluación Atmel, Cirrus Logia, etc.

- Trace:
    - Prueba IAR J-Trace.
    - Interface Signum JTAGjet-Trace ETM.

- Soporte para sistemas operativos:
    - Built in plugings:
        - CMX-RTX
        - CMX-Tiny +
        - ThreadX OS.
        - OSE Epsilon
        - OSEK (ORTI)
        - Segger emboss
    - Vendor plugins:
        - RTXC Quadros.
        - µC/OS – II OS
        - Fusion OS
        - MiSPO NORTi
        - SMX OS.

- Ensamblador IAR:
    - Set versátil de operadores y directivas.
    - Diseñado en lenguaje preprocesador C, aceptando todas las macro definiciones C.

- Asociador IAR XLINK:
    - Completa asociación, relocalización y generación de formato para producir códigos de FLASH/PROMable.
    - Comandos para segmentos que permiten detallado control en la

localización de códigos y datos.

- Asociación optimizada para remover códigos y datos no útiles.
- Asociación directa con imágenes binarias, para archivos multimedia.
- Chequeo de suma (Checksum) opcional para generar chequeos en tiempo de ejecución.
- Referencia completa y dependiente en los mapas de memoria.
- Soporta hasta 30 formatos con estándares industriales incluyendo ELF/DWARF versión 2, compatible con la mayoría de emuladores y depuradores.

- IAR J-Link para ARM (herramienta depuradora de hardware) es una herramienta JTAG que se conecta vía puerto USB, se ejecuta en un computador con un ambiente Windows y es compatible con IAR EWARM, cuenta con las siguientes características:
  - Interfase manejada a través del puerto USB.
  - Soporta todos los dispositivos ARM7 y ARM9.
  - Velocidades de descarga hasta de 580 kB/seg.
  - No requiere fuente alimentación, se alimenta a través del puerto USB.
  - Velocidad de JTAG 12MHz.
  - Reconocimiento con autovelocidad.
  - USB2 de alta velocidad (12Mbit/seg).
  - Reloj adaptativo basado en la señal de RTCK JTAG.
  - Conector estándar de 20 pines.
  - Se ejecuta en Windows 2000 y Windows XP.
- Versión de evaluación para edición de códigos con un máximo de 32 kB.

3.7.2 µKeil Vision 3

Es un MDK[7] IDE[8] de Keil software (una compañía de ARM) [38], combina la administración de proyectos, edición de código fuente, depuradora de programas, y programación de dispositivos con memorias flash para dispositivos ARM de las familias ARM7, ARM9, Cortex-M1 y Cortex M3 en un solo y poderoso ambiente [38] . Entre sus características:

- Configuración de proyectos:
  - Selección de dispositivos (alrededor de 260 dispositivos).
  - Herramientas de configuración.
  - Autoconfiguración de herramientas al seleccionar un dispositivo.
- Administración de proyectos:
  - Agrupación de archivos: asocia archivos en grupos según las necesidades, o bloques funcionales.
  - Asignación de proyectos: permite crear múltiples programas o grupos en un único proyecto.
- Explorador de códigos y editor:
  - Contiene todas las barras de herramientas para rápido acceso a funciones como depuración y cambios en códigos, además presenta múltiples pantallas con información de los símbolos y variables en el programa.
- uVision 3 asegura un manejo fácil y consistente de proyectos.
- Un solo proyecto almacena los nombres de los archivos y guarda las configuraciones para el compilador, lenguaje ensamblador, asociador, depurador, cargador en flash y otras utilidades.

---

[7] MDK: microcontroller development kit.

[8] IDE: integrated development environment.

- Facilidades del editor para creación, modificación y corrección de programas.
- Depurador para tarjetas de evaluación o CPU y simulación de periféricos.
- Depurador y simulador de dispositivos: El depurador soporta puntos de ruptura (con expresiones condicionales y lógicas) y puntos de ruptura con acceso a memoria para operaciones de escritura/lectura desde una dirección o para un rango. ARM depurador también simula microcontroladores a través de un completo set de instrucciones y periféricos en el chip. Este simulador provee beneficios y colabora de manera confiable en el desarrollo y verificación de software. Características del depurador y el simulador:
  - Depurador:
    - Muestra pantalla con código desensamblado.
    - Botones para ejecución código paso a paso y otros.
    - Analizador lógico.
    - Ventana con variables arrastre y pegue (drag and drop).
    - Visores de memoria y variables.
  - Simulador:
    - Simulación de software sin presencia de hardware.
    - Permite simular hasta 260 dispositivos embebidos con procesadores ARM.
    - Aumenta la confiabilidad en el software.
    - Permite puntos de ruptura que los depuradores por hardware no permiten.
    - Permite simular entradas sin adición de ruido.
    - Permite generar funciones de señales óptimas que permiten emular el mundo real.
    - Simulación de algoritmos DSP paso a paso.

- Soporte a kernel RTX OS.
- Herramienta de hardware para depuradora (uLink2).
- Posee demarcador de código fuente, navegador de funciones, editor de plantillas, búsquedas incrementales, ayudante para configuración, analizador lógico, simulación para periféricos CAN e I2C, programación para memoria flash de dispositivos y depuradora con interfaz JTAG.
- Desempeño compilador RealView: RealView MDK está basado en las herramientas de compilación de ARM RealView, que entrega más alto desempeño en la generación de código para dispositivos ARM. Se incluye la librería MICROLIB, utilizada para optimizar sistemas embebidos.
- Versión de evaluación para edición de códigos con un máximo de 32 kB.
- Compilador C/C++:
    - Generación de códigos para modos ARM y Thumb.
    - Optimizaciones para el aprovechamiento de la memoria.
    - Funciones para soporte de hardware: __irq (interrupt request) y __swi (software interrupt handler).
    - Ensamblador Embebido: logra incluir lenguaje ensamblador a través de funciones macro de C.
    - Funciones inlining: utilizado para evitar la sobrecarga de funciones cuando se utilizan frecuentemente en un programa.
    - Estándar IEEE 754 para números en punto flotante de precisión simple y de doble precisión.
    -

3.7.3 Software Libre – Plataforma Eclipse - Yagarto

La mayoría de plataformas usadas para desarrollo de aplicaciones de procesadores ARM son paquetes comerciales, tales como IAR EWARM, Rowley (Cross Works) o ARM RealView (uVision 3) tienen un valor desde $900 y las versiones de evaluación

permiten crear aplicaciones con tamaños de 32 kB como máximo y limitan el uso de algunas herramientas.

Estos paquetes son fáciles de instalar, tienen soporte eficiente, pero el problema son sus precios que muchas veces no están al alcance de estudiantes, aficionados o ingenieros que tienen presupuesto limitado. Por eso es importante conocer que existen herramientas de código abierto y libre disponibles en la WEB que pueden ensamblar paquetes de desarrollo en pocas horas [39].

3.7.3.1 Sistema cruzado de desarrollo para software ARM

Existen algunas herramientas para escribir códigos en C como notepad de Windows, paralelamente se puede usar la consola de Windows (Windows Command Prompt) para compilarlos y hacer asociaciones ("linkearlos") con herramientas GNU[9] escribiendo comandos, esto es un proceso tedioso y es un trabajo en vano si se dispone de herramientas gráficas como Eclipse. Eclipse permite editar y modificar programas en C en un moderno editor de software. También provee herramientas de compilación y depuración.

Eclipse [39] utiliza el compilador y asociador Free Software Foundation GNU toolchain[10] for ARM. El compilador GNU C se caracteriza por universalidad y por su capacidad para generar códigos para las más populares arquitecturas de microcomputadoras. Adicionalmente GNU C toolchain incorpora un ensamblador, asociador, utilidad make, depurador, librerías y otras utilidades.

El compilador GNU C provee un desempeño muy cercano en velocidad y densidad de código a los compiladores profesionales de ARM, KEIL, HITEX, IAR y otros. GNU C y su utilidad make es usada por Eclipse para administrar archivos creados y ejecutar las herramientas de compilación en las secuencias correctas. El depurador

---

[9] GNU (no es Unix): Licencia pública general

[10] Toolchain: es un conjunto de herramientas, binarios y librerías necesarias para la compilación cruzada (binutils, gcc, glibc).
GCC: GNU compiler collection, es una colección de compiladores GNU. GCC es un software libre y con licencia GPL.

GNU GDB está completamente integrado a Eclipse IDE, este proporciona un ambiente gráfico animado durante la depuración y con la facilidad de puntos de ruptura (breakpoints), ejecución paso a paso y una inspección sofisticada de variables y estructuras de datos.

El compilador GNU C (versión ARM) y el ensamblador GNU (versión ARM) son usados para compilar y ensamblar los archivos fuente. Para crear archivos descargables en memoria flash o RAM se sigue el siguiente proceso:

- El compilador GNU C crea los archivos objeto que son archivos muy cercanos a las instrucciones de lenguaje de máquina que serán ejecutados por el procesador ARM, pero las direcciones no han sido asignadas. Estas direcciones son asignadas luego por el asociador, lo que le da la capacidad al usuario de cargar el programa en la localidad de memoria que desee.
- El asociador GNU se utiliza para reunir archivos objeto, además de algunos módulos de bibliotecas, resolviendo todas las direcciones, y combinándolas en un archivo descargable con extensión ".out". Un comando asociador con extensión ".cmd" se utiliza para especificar el orden y destino para localización en memoria de los módulos creados.
- El archivo ".out" se compone de información para el depurador y de las instrucciones ejecutables de lenguaje de máquina. Normalmente este archivo es usado para descargarse en memoria para ejecución exclusiva en memoria RAM o simplemente para ayudar al depurador Eclipse/GDB en la identificación de símbolos y sus direcciones de memoria, etc. EL asociador también produce un archivo ".map" el cual es útil para determinar el tamaño de los módulos, su ubicación en memoria, etc.
- Si se desea crear un archivo para descargar en memoria flash, entonces se requiere crear un archivo binario puro que es requerido por el depurador OpenOCD JTAG y por los depuradores por hardware como Segger J-Link, Macraigor Wiggler, Olimex ARM USB OCD. Este archivo se crea pasando el archivo ".out" a través de la utilidad GNU ObjCopy para crear el archivo binario ".bin".

- Inherentemente se crea un archivo hexadecimal (“.hex”) que puede utilizarse para descargarlo en memoria a flash usando programas como Flash Magic, Ethernet flash utility de NXP y otros.

Los pasos mencionados se pueden ir ejecutando a través de los comandos dentro del Windows Command Prompt. Sin embargo Eclipse utiliza la herramienta GNU make para automatizar todo el proceso. Make busca un makefile que prepara y ejecuta las acciones mencionadas de forma automática.

Tabla 8 Síntesis plataformas de desarrollo para procesadores ARM.

Plataforma	Licencia	Sistema Operativo	Versión de Evaluación		Depurador	Simulador	Soporte de dispositivos	Lenguaje
			Codigo limitado	Tamaño Máximo				
IAR EWARM	Sí	Windows 2000/XP	Sí	32kB	Sí JTAG, TRACE	Sí	Dispositivos con procesador ARM embebido (ARM7, ARM9, ARM9E, ARM10, ARM11, SecureCore, Intel Xscale).	C, EC++ asm
KEIL	Sí	Windows 2000/XP	Sí	32kB	Sí JTAG	Sí	Alrededor de 260 dispositivos con algún núcleo AEM (ARM7, ARM9, Cortex-M1 y Cortex M3)	C, EC++ asm
ECLIPSE	No	Windows 2000/XP Linux kernel 2.4.0	No	Ilimitado	Sí JTAG	Sí	Dispositivos ARM de las familias ARM7 y ARM9	C, EC++ asm

3.8 Sistemas Operativos (OS)

Los sistemas operativos inician su historia a partir de los años 50. Nacen como una necesidad de que un computador pudiera disponer de múltiples usuarios y al mismo tiempo que cada usuario lograra tener múltiples tareas. En forma más general nacen como una necesidad de aprovechar de una forma más eficiente los recursos del computador [40, 41].

Con la aparición de los sistemas operativos empieza a generarse una amplia terminología, a continuación una breve historia de los sistemas operativos y algunos conceptos [40, 41]:

- Open shop: nace en IBM la idea de los sistemas operativos (1954) [29].
- Batch processing (1961): se introducen estaciones con cintas magnéticas, lo que eliminó el cuello de botella de las tarjetas perforadas, los usuarios iniciaron a programar secuencias (Batch processing), aunque se continuaba con la problemática de que el computador solo podía ejecutar una tarea a la vez [29,30].
- En 1961 nace el sistema BKS en la computadora IBM 709, fue un primer intento porque una computadora realizara una lista de tareas [29].
- Multiprogramación: Para los años 60 ya habían núcleos con más memoria, memoria secundarias con acceso aleatorio, canales de datos e interrupciones por hardware lo que cambio radicalmente el rumbo de los sistemas operativos. Las interrupciones le permitieron al procesador simular la ejecución concurrente de múltiples programas y operaciones simultáneas de entrada/salida. Esta forma de concurrencia se le conoce como multiprogramación [29,30].
- Time sharing (tiempo compartido), MIT Boston, año 1960: con la aparición de la multiprogramación y de las memorias secundarias fue posible desarrollar sistemas operativos que administraran un flujo continuo de entrada, procesamiento y salida en un solo tambor (disco (drum)), a esta configuración se le llamó “spooling”. “Spooling” no requería cintas magnéticas lo que

posibilitó tener una prioridad de trabajos en memoria RAM, con prioridades como “se ejecuta la tarea más pequeña” [40].

En pruebas realizadas en un computador IBM 7094 se lograron tener 32 usuarios y cada usuario tenía al menos 0.1 s cada 3 segundos del procesador de la máquina (time sharing) [41].

Se introduce el concepto “swap”, en el cual cuando un usuario se apropia de la máquina y se guarda el trabajo de los demás usuarios en el disco (drum) [40, 41].

Actualmente el concepto de spooling es sinónimo de “batch processing”.

La multiprogramación optimizó el uso del CPU y el tiempo compartido es un intento a optimizar el uso de los programadores.

- Paginación y memoria virtual: La computadora Atlas construida por la universidad de Manchester alrededor de 1960 fue la primera en tener lo que se llamó “un nivel de almacenaje”. Esta combinaba 16 k palabras de memoria real, además de almacenar en disco. El sistema operativo y una “paginación” especial en el hardware hacían parecer la máquina Atlas como una máquina con un millón de palabras, a esta idea se le llamó “memoria virtual” [40, 41].
- Manejadores de dispositivos e interrupciones: para fines de los años 60 y principios de los 70 ya se contaba con manejadores de dispositivos (device drivers), estas son rutinas que organizaban las transferencias de datos a y desde periféricos. Los manejadores de interrupciones (interrupt handlers) se usaban para conocer cuál dispositivo había terminado una transferencia de datos [41].
- Administración de procesos: parte del código ya se podía dedicar a la administración de procesos (process management), por ejemplo para poner en diferentes horarios la asignación del CPU, los programas y otros. Este código permitía saber cuáles programas estaban listos en el CPU y cuáles estaban esperando por terminales de entrada salida, discos u otros. Si el sistema soportaba usuarios interactivos con tiempo compartido (time sharing),

este código de horarios debía hacer que un programa no se ejecutara por mucho tiempo en el CPU para poder darle algunos ciclos de reloj a los demás programas [40, 41].

- Interprete de control de trabajos (JCL): son estados para especificar cual secuencia de procesos se requieren para que una tarea se lleve a cabo [41].
- Administración de archivos: se creó con la necesidad de permitir a los usuarios crear archivos para guardar datos en el disco, entre sus características conocía que usuario tenía privilegios para realizar estas acciones y cuanto espacio tenía disponible [40, 41].
- Administración de memoria: se utiliza para colocar los programas de los usuarios en memoria. Se interlazó con el concepto de memoria virtual [40, 41].
- Kernel: es un mecanismo para comunicar procesos en paralelo a través de mensajes. Concepto creado con RC 4000 en Dinamarca [40, 41].

3.8.1 Terminología

A continuación se presenta una terminología específica relacionada con los sistemas operativos:

- Tarea: cualquier programa ejecutándose en el núcleo del CPU. Sin un kernel multitarea solo una tarea puede ser ejecutada por el CPU a la vez, a esto se le llama sistema mono tarea [42].
- Un sistema operativo permite la ejecución de múltiples tareas en un mismo CPU. Toda tarea se ejecuta como si fuera propietaria del CPU. Las tareas son programadas, lo que quiere decir que el OS puede activar y desactivar cada tarea [40, 41, 42].
- Sistemas mono tarea (súper lazos): básicamente es un programa que se ejecuta en un lazo sin fin, llamando funciones del OS para ejecutar las operaciones apropiadas (nivel de tarea). Un kernel de tiempo real no es usado, entonces las interrupciones deben ser usadas para ejecutar las partes del software de tiempo real o las partes críticas del software (nivel de

interrupciones). Este tipo de sistema típicamente es pequeño, no es complejo y el comportamiento de tiempo real no es importante. Existen cieOS problemas de sincronización y desplazamiento de tareas con aplicaciones mono tarea. Como no se usa un kernel, solo existe una pila en memoria ROM, lo que hace que la memoria ROM sea más pequeña y menos memoria RAM es usada para pilas [42].

Los súper lazos se vuelven difíciles de manejar si los programas son muy grandes, esto porque un componente de software solo puede ser interrumpido por otro componente a través de ISR (solicitud de servicio por interrupción). El tiempo de reacción de un componente depende de la ejecución de todos los otros componentes en el sistema; por tanto el comportamiento en tiempo real es pobre

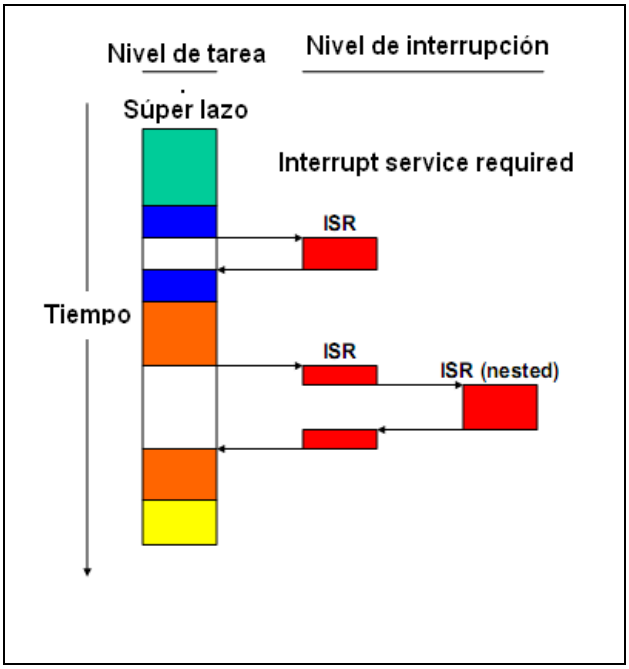

Figura 3 Sistema monotarea [42].

- **Sistema multitarea:** son aquellos sistemas que cuentan con algún algoritmo de programación de tareas, en los cuales la potencia de cálculo del CPU puede ser distribuida entre tareas [42].

- **Multitarea cooperativo:** estos sistemas esperan cooperación de todas las tareas. Las tareas solo pueden estar suspendidas por una función de sistema operativo. Si no es así el sistema “cuelga”, lo que quiere decir que otras tareas no tienen oportunidad de ser ejecutadas por el CPU mientras la primera tarea está siendo llevada fuera del sistema. Incluso si una ISR (solicitud de servicio por interrupción) tiene una tarea de mayor prioridad lista para ejecutarse, la tarea interrumpida retornará y finalizará antes la tarea que estaba realizando [42].

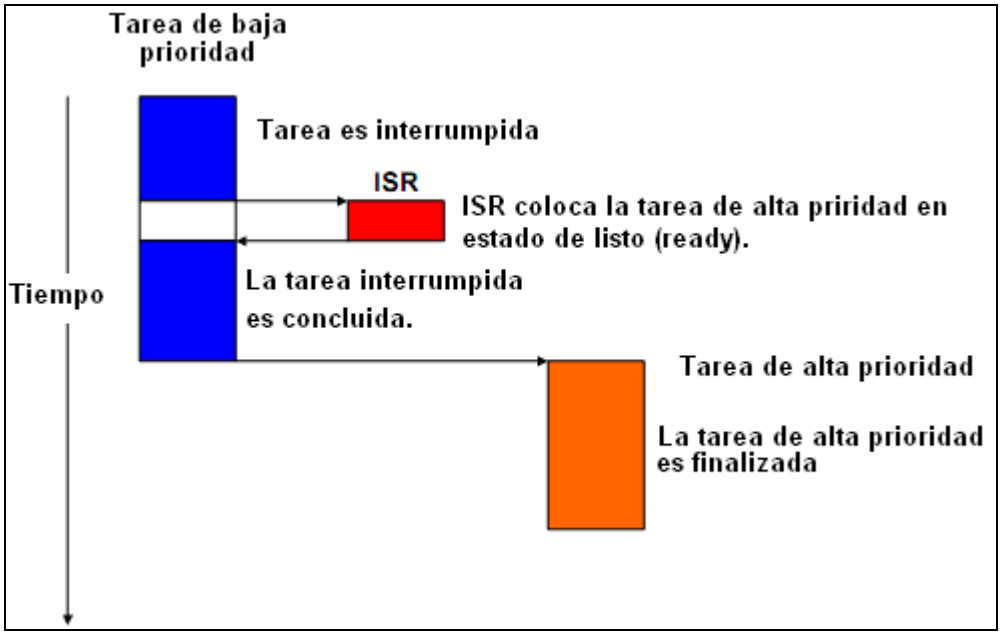

Figura 4 Sistema de cooperación multitarea [42].

- **Multitarea desplazante:** los sistemas de tiempo real operan solo con multitarea desplazante. Un sistema operativo de tiempo real necesita una interrupción periódica de un temporizador para interrumpir tareas en un tiempo determinado para llevar a cabo cambios de tarea si es necesario. La tarea de más alta prioridad en el estado de listo (Ready State) es por tanto siempre ejecutada, si es o no una tarea interrumpida. Si una ISR

tiene una tarea de más alta prioridad lista para ejecutarse, un cambio de tarea ocurrirá y la tarea será ejecutada antes de regresar a la tarea interrumpida [42].

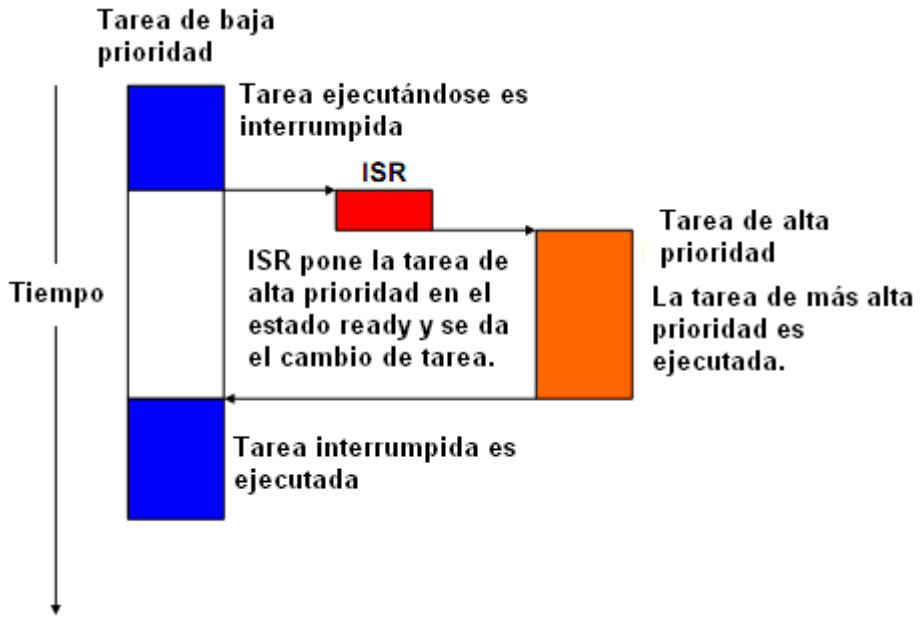

Figura 5 Sistema multitarea desplazante [42].

- **Programación (horarios):** existen diferentes algoritmos para saber cuál tarea se ejecuta, a estos algoritmos se le conoce como programadores (schedulers). Todos los programadores tienen algo en común: distinguen cuál tarea está lista para ejecutarse (en el estado de listo (ready state)) y cuál tarea está suspendida por alguna razón (retardos, espera por un mailbox, espera por semáforo, espera de un evento, y algunas otras). Los programadores seleccionan una de las tareas en el estado de listo y la activan, o sea ejecutan el programa de esta tarea. La tarea que actualmente está ejecutándose es referida como la tarea activa. La principal diferencia entre los programadores está en cómo distribuyen el tiempo de ejecución entre las tareas que se encuentran en el estado de

listo (ready state) [42].

- **Algoritmo de programación Round-Robin (de forma ordenada):** con la programación Round Robin el programador tiene una lista de tareas, y cuando desactiva la tarea activa, activa la siguiente tarea en el estado de ready. La programación Round robin puede ser utilizada en ambos esquemas de multiprogramación, desplazante y cooperativa. Este algoritmo trabaja bien si no se requiere garantizar el tiempo de respuesta, si el tiempo de respuesta no es una cuestión importante o si todas las tareas tienen la misma prioridad.

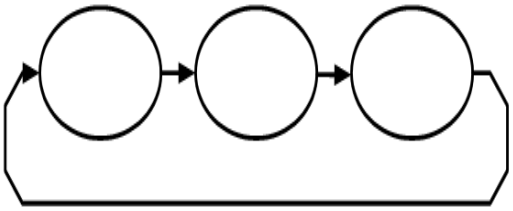

Figura 6 Algoritmo Round robin.

Todas las tareas están en el mismo nivel; la posesión del CPU cambia periódicamente después de un tiempo de ejecución predefinido. A este tiempo se le llama porción de tiempo (timeslice), y puede ser definido individualmente para cada tarea.

- **Algoritmo de programación controlado por prioridad:** en las aplicaciones del mundo real, las diferentes tareas requieren diferentes tiempos de respuesta. Por ejemplo, en aplicaciones que requieren controlar un motor, recibir caracteres de un teclado, o imprimir en una pantalla LCD, usualmente el motor requiere una reacción más rápida que el teclado o la pantalla. Mientras que la pantalla es actualizada, el motor necesita ser controlado. Esto hace a un sistema multitarea desplazante un deber. El algoritmo Round Robin podría trabajar, pero no garantizaría un tiempo específico de reacción, una mejora a este algoritmo puede ser usada.

En la programación controlada por prioridad, a cada tarea se le asigna una prioridad. El orden de ejecución depende de esta prioridad. La regla es muy simple: “el programador activa la tarea que tiene la prioridad más alta en el estado de listo (ready state)”. Esto significa que cada vez una tarea con más alta prioridad esté activa en el estado de listo (ready state) se convierte en la tarea activa. Sin embargo, el programador puede ser desactivado en partes de un programa donde los cambios de tarea son prohibidos, a estas partes de un programa se les conoce como regiones críticas.

Round Robin es un algoritmo le permite al sistema operativo evitar decidir si una tarea es más importante que otra. Las tareas con prioridad idéntica no pueden bloquear otra tarea por más tiempo que el indicado por su porción de tiempo. Round Robin también cuesta tiempo si dos o más tareas con la misma prioridad están listas y no hay una tarea de más alta prioridad lista, esto debido a que continuamente una tarea estaría interrumpiendo a la otra y viceversa, por eso es más eficiente asignar diferentes prioridades a cada tarea para evitar cambios innecesarios de tareas.

3.8.2 Comunicación entre tareas

En un programa multitarea (multi-hilo), múltiples tareas trabajan completamente separadas. Como todas las tareas trabajan en la misma aplicación, algunas veces será necesario intercambiar información entre tareas.

3.8.2.1 Variables globales

La manera más fácil de comunicar tareas es a través de variables globales. En ciertas situaciones hacer esto tiene sentido aunque tiene desventajas. Por ejemplo, si se desea sincronizar una tarea para iniciar cuando cambie el valor de una variable

global, se debe monitorear constantemente esta variable, desperdiciando potencia de cálculo y tiempo, y el tiempo de reacción depende de que tan a menudo se observe la variable.

3.8.2.2 Sistemas eficientes de comunicación

Cuando múltiples tareas trabajan en conjunto generalmente intercambian datos, se sincronizan con otras tareas o se aseguran que los recursos no están siendo usados por otra tarea. Por estas razones se crearon mecanismo de comunicación entre los cuales se encuentran correos (buzones), colas (queues), semáforos y eventos.

- **Buzones y colas:** un buzón básicamente es un buffer de datos administrado por el OS y es usado para enviar mensajes a una tarea. Este trabaja sin conflictos incluso si múltiples tareas e interrupciones intentan accesarlo simultáneamente. Un OS activa de forma automática cualquier tarea que está esperando por un mensaje en un buzón, en el momento que se tienen nuevos datos si es necesario se produce el cambio de tarea.

  Una cola funciona de manera similar, pero permite manejar mensajes con una longitud mayor, y cada mensaje puede tener su propia longitud.

- **Semáforos:** existen dos tipos de semáforos que son usados para sincronizar tareas y administrar recursos. Estos son los semáforos de recursos y los semáforos de conteo.

- **Eventos:** una tarea puede esperar por un evento en particular sin usar ningún tiempo de cálculo. La idea es simple y convincente. No hay que desperdiciar tiempo en monitorización si simplemente se quiere activar una tarea cuando suceda un evento. Con esto se evita gastar potencia de cálculo y asegura que la tarea responda al evento sin retardo alguno. Típicamente aplicaciones que utilizan eventos son aquellas que en las cuales las tareas esperan datos, teclas presionadas, un comando recibido o un carácter, o el pulso

de un reloj externo de tiempo real.

3.8.3 Tareas en un OS

Un sistema multitarea ejecuta múltiple tareas como si fueran programas únicos, cuasi – simultáneamente en un solo CPU. Una tarea en un sistema multitarea se compone de tres partes:

- El código del programa, residente en memoria ROM, aunque no siempre tiene que ser así.
- Una pila, que residente en memoria RAM y es accesada por el puntero de pila (stack pointer).
- Un bloque de control de tarea (TCB), residente en memoria RAM.

La pila tiene la misma función que la pila de un sistema mono-tarea. Almacenar la dirección de regreso cuando se dan llamadas de funciones, parámetros y variables locales, y almacenaje temporal de cálculos intermedios y valores de registros.

El bloque de control de tareas (TCB) es una estructura de datos asignada a una tarea cuando es creada. Esta estructura contiene información del estado de la tarea, incluyendo el valor del puntero de pila, la prioridad de la tarea, estado actual de la tarea (lista, esperando, razones para estar suspendida) y otros manejos de datos. Toda esta información le permite a una tarea interrumpida seguir ejecutándose exactamente en el punto que fue dejada. Los TCB solo son accesados por el OS.

3.8.4 Intercambio entre tareas

En la siguiente figura se muestra cómo el programador (Scheduler) desactiva la tarea que será suspendida (Tarea 0) salvando los registros en su propia pila. Luego el programador activa la pila de mayor prioridad (Tarea n) cargando el puntero de pila desde los valores de los registros almacenados en la pila de la tarea n.

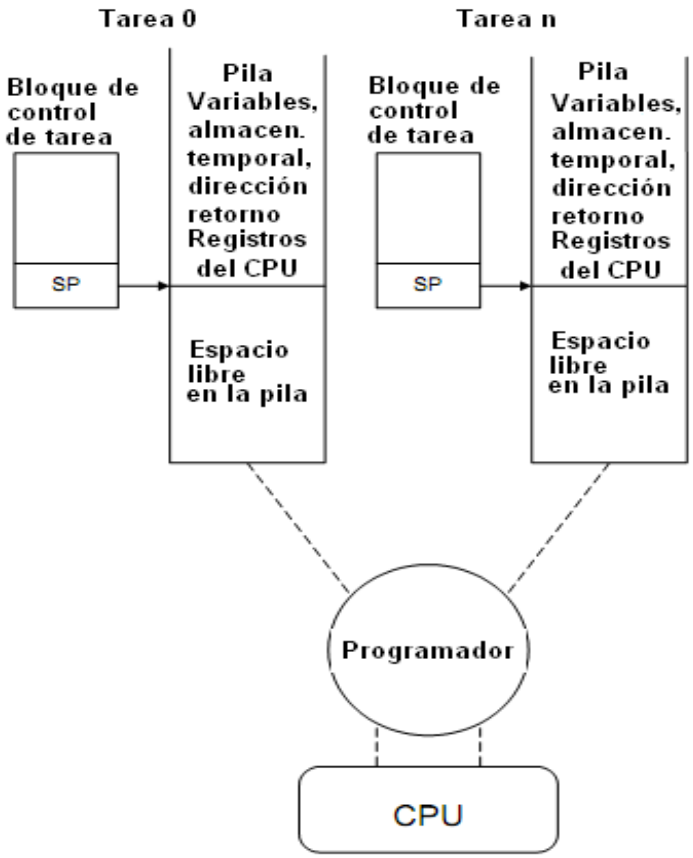

Figura 7 Cambio de tareas en un OS.

3.8.5 Cambio de estado en una tarea

Una tarea puede estar en uno de muchos estados en un momento dado. Cuando una tarea es creada automáticamente se pone dentro del estado de listo (ready state).

Una tarea es activada en el estado de listo tan pronto como no haya otra tarea con prioridad mayor. Solo una tarea puede estar activa a la vez. Si una tarea de mayor prioridad está lista, esta tarea es activada y la tarea desplazada es recordada en el estado de listo.

Una tarea puede ser retardada hasta un tiempo especificado, en este caso la tarea se coloca en un estado de retardo y la siguiente tarea con mayor prioridad es activada en el estado de listo.

La tarea activa también puede tener que esperar por un evento, un semáforo, un buzón, o una cola. Si el evento no ha ocurrido, la tarea es colocada en el estado de espera y la siguiente tarea de mayor prioridad es en el estado de listo es activada.

A continuación todos los estados posibles de una tarea y las transiciones entre ellos.

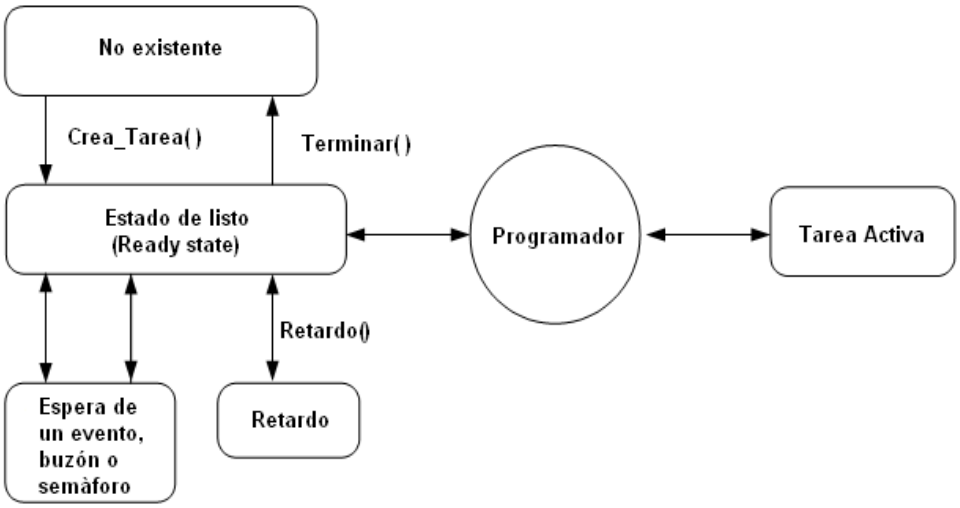

Figura 8 Estados posibles de una tarea.

3.8.7 Cómo el sistema operativo gana control

Cuando el CPU es reiniciado, los registros de función especial (SFR) son cargados con sus respectivos valores. Después del reinicio la ejecución del programa da comienzo. El contador de programa (PC) toma el valor de la dirección de inicio definida por el vector de inicio o la dirección de inicio (dependiendo del CPU). Esta dirección se encuentra en el módulo de inicio enviado por el compilador, y es parte de las librerías estándar. El código de inicio desarrolla las siguientes acciones:

- Carga los apuntadores de pila con los valores por defecto, así es para la mayoría de CPU's.
- Inicializa todos los segmentos de datos con sus respectivos valores,
- Se llama la rutina principal (main).

En un sistema mono-tarea la rutina principal es parte del programa que toma control inmediato después del inicio del CPU. En un OS la rutina principal se utiliza para

crear una o más tareas, luego se inician los módulos multitarea, y el OS toma control de cual tarea debe ser ejecutada. En un OS Se recomienda crear las tareas en la rutina principal y luego dar inicio al sistema operativo. También se recomienda crear semáforos y buzones en la rutina principal para tener un adecuado control, además se recomienda también practicar la programación modular para reutilizar de código.

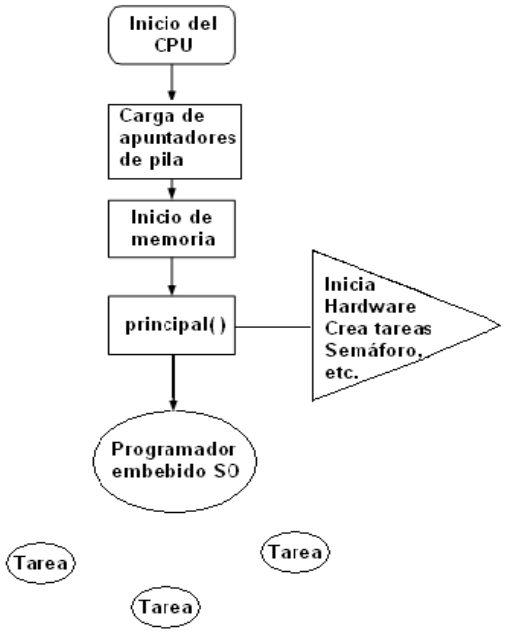

Figura 9 Inicio general de un SO.

3.8.8 Sistemas operativos para procesadores ARM

Existen diversos sistemas operativos para procesadores ARM, algunos son propietarios y se deben adquirir licencias para tener acceso a los códigos fuente principales (kernel), incluso en algunos casos no permiten crear más de un cierto número de tareas, otros OS son de código libre y abierto (free and open software), estos permiten crear todas las tareas que el sistema permita y se tiene acceso a toda información que el usuario necesite conocer. A continuación una lista con sistemas operativos y sus características [43, 44, 45, 46, 47, 48, 49].

Tabla 9 Sistemas operativos para procesadores ARM

SO	Código Abierto	Código gratis	Tipo (cooperativo, reemplazante)	Tamaño	Tareas (también depende la memoria RAM y ROM disponible)	Prioridades	CPU	Soporte	Lenguaje
IARPOWER PACK	No	Versión de evaluación	Reemplazante, con algoritmo Round Robin para tareas con idéntica prioridad.	Aprox. 15 kB	255 (versión de evaluación 3 máximo)	127	ARM7, ARM9, ARM9E, ARM10E, ARM11, SecurCore, Cortex M3, y XScale	USB, Ethernet, FTP, son funcionalidades que se deben comprar	C, C++
uCos II	No	Sí	Reemplazante. Con programador de taza monotónico	2.5 kB	64 (56 disponibles para el usuario 56), programador de taza monotónico	56 , 4 más bajas y 4 más altas están reservadas	ARM7 y ARM9, entre otros	TCP/IP, ModBus, USB	C, C++
Partikle	Sí	Sí	POXIS, reemplazante	1.5 kB	Ilimitado	Prioridades fijas, se maneja por políticas de administración de Partikle.	ARM7 (Específicamente LPC2138)		C, C++, Java, ADA
OS	Sí	Sí	Reemplazante, cooperativo y combinaciones híbridas	4.3KB en ARM7	Ilimitado	ilimitadas, se manejan por software y pueden existir tareas con la misma prioridad.	Cortex-M3, ARM7, MSP430, H8/S, AMD, AVR, x86 y 8051	TCP/IP	C, C++

Tabla 9 Continuación. Sistemas operativos para procesadores ARM

uCLinux	Sí	Sí	Reemplazante	0.5 MB a 1 MB	Ilimitado	Ilimitado	ARM7, ARM9,Coldfire (versiones de DragonBall), MIPS 4000	TCP/IP, Modbus, USB	C, C++
eCos	sí	Sí	Reemplazante		Ilimitado	Ilimitado	ARM7 y ARM9, Hitachi H8300, Intel x86, MIPS, Matsushita AM3x, Motorola 68k, PowerPC, SuperH, SPARC y NEC V8xx	USB, TCP/IP, GDB	C
Salvo	sí	No	Cooperativo con algoritmo round Robin y explícitamente se debe indicar la tarea que se ejecuta.	1 a 1.5 kB	ilimitadas	15	8051, ARM7, Cortex M3, PICs (12,16,17,18) y DsPICs	TCP/IP	C++

Capítulo 4: Procedimiento Metodológico

4.1 Reconocimiento y definición del problema

En años anteriores se ha trabajado en el diseño de algoritmos para la detección de fuego, motosierras y disparos. También se ha trabajado en el diseño y la implementación de algunos circuitos para la transmisión de señales de forma inalámbrica a través de una red de dispositivos. El problema es la no existencia de un nodo sensorial completo que pueda tomar información del medio para procesarla y poder determinar la existencia de un daño al ambiente.

4.2 Obtención y análisis de la información

Este es un proyecto orientado a la investigación, para su desarrollo fue necesario estudiar tesis desarrolladas de los algoritmos de detección, tesis de los sistemas de comunicación inalámbrica desarrollados, esto para tener una base sólida de lo que se requería en el sistema.

Una vez establecidas las necesidades del sistema, se inició una investigación en páginas WEB, artículos y libros para obtener la información sobre las características que influyen en el rendimiento y consumo de potencia de un procesador. El valor agregado de esta parte de la investigación es que permitió tener una visión más amplia sobre diversas características que podían ayudar a la escogencia del procesador deseado para el nodo sensorial.

La investigación llevo a la necesidad de buscar artículos y libros con información de aspectos varios de las arquitecturas de procesadores. Se buscó información de comparaciones entre arquitecturas CISC y RISC, también se buscaron artículos de estudios de procesadores RISC de diversas marcas. La información obtenida aquí principalmente estaba relacionada con consumo de potencia, benchmarks y aplicaciones.

La información de los procesadores ARM se obtuvo directamente de artículos y especificaciones dadas en la página www.arm.com. Esta página brinda información de datos técnicos y aplicaciones de estos procesadores, esto para todas las familias ARM7, ARM9, ARM10, ARM11 y Cortex.

La investigación también llevó a sitios web como www.nxp.com, www.atmel.com, www.olimex.com, www.lpctools.com, www.sparkfun.com, en estos sitios se encontraron bases de datos con la información necesaria para escoger un microcontrolador con un núcleo ARM7 y una tarjeta de desarrollo con todas las características deseadas para el desarrollo del proyecto. Aquí se realizó un análisis de las diversas posibilidades y características de cada tarjeta de desarrollo.

Se visitaron sitios web como www.iar.com, www.keil.com y www.yagarto.de, donde se encontraron plataformas de desarrollo y sus características. Con las plataformas de desarrollo y con la tarjeta de desarrollo ya se disponía de todas las herramientas necesarias para iniciar las pruebas de hardware y software.

Sitios web como www.iar.com, www.partikle.org, www.uclinux.org, www.ecos.com, www.OS.org, y otros, brindaron la información y programas de sistemas operativos desarrollados para procesadores ARM, esto fue de gran ayuda ya que se encontró teoría de sistemas operativos y las diversas características de cada sistema operativo.

Parte de la información obtenida en esta investigación fue necesaria obtenerla de la página www.ieee.com, esto a través de los profesores de la escuela.

4.3 Evaluación de las alternativas y síntesis de una solución

Definidas las necesidades de hardware y software de los algoritmos de detección desarrollados por la escuela de ingeniería electrónica, se definieron características de la unidad de proceso central y las características mínimas del nodo sensorial y se procedió a escoger un procesador RISC de 32 bits, de la marca ARM y familia

ARM7. Existen otras gamas de posibilidades de procesadores 32 bits RISC, la limitante es que sus consumos de potencia son altos a comparación de procesadores ARM, esto debido a que son procesadores que funcionan a frecuencias mayores y con tecnologías de fabricación que demandan más potencia.

Se decidió escoger un procesador ARM7 debido a que son los procesadores de 32 bits RISC de menor consumo de potencia en el mercado, aunque no necesariamente son los que brindan el mejor rendimiento. Una condición de la escogencia de la unidad de proceso central es que nunca la capacidad de procesamiento debía sobreponerse al consumo de potencia.

En el caso especial del núcleo ARM7 este no integra unidad de punto flotante (FPU) esto porque este proyecto plantea que se utilizará un dispositivo externo para realizar cálculos, por ejemplo un FPGA.

Durante el proceso de investigación de los procesadores ARM de 32 bits se encontró que estos se integran a sistemas microcontrolador de marcas como ATMEL, Philips, Cirus y otras. Estos sistemas microcontrolador entre otras características incluyen diversos protocolos de comunicación como RS232, USB, ETHERNET, I2C, SPI, además de ADC's, modos de manejo de energía, controladores externos de memoria (EMC), esto para manejar bancos de memoria mayores a los integrados en el chip propiamente.

Conociendo la existencia de los procesadores ARM y de los sistemas microcontrolador que los integran, se investigó sobre las diversas tarjetas de desarrollo existentes en el mercado que incorporaran el sistema microcontrolador y que incluyeran en un solo paquete las diversas partes de hardware faltantes del sistema.

Se buscaron tarjetas de desarrollo con un mínimo de memoria externa flash de 2 MB. La razón de escoger una tarjeta de desarrollo para el proyecto fue ahorrar tiempo de desarrollo en prototipos.

La escogencia final fue la tarjeta de desarrollo LPC L2294, puertos RS232, USB, ETHERTNET y otros, además de 4 MB de memoria RAM y 8 MB de memoria flash.

Existen otras tarjetas como la LPC H2294 que incorpora 1 MB de memoria flash y 4 MB de memoria RAM, pueden ser suficientes pero los algoritmos de detección de sonido necesitan incluso 2MB de memoria no volátil, por esta razón no se escogió.

Por cuestiones de investigación y pruebas se trabajó con dos plataformas de desarrollo para procesadores ARM, específicamente EWARM de IAR y Eclipse de Yagarto, estas son plataformas para los sistemas microcontrolador. IAR EWARM se trabajó con su versión de evaluación de 32 kB, de gran utilidad por orden en la programación y simulación. Eclipse se trabajó con el compilador de software libre arm-toolchain-elf.

La parte final del trabajo se centró en instalar un sistema operativo para procesador ARM7, por esto fue necesario investigar los diversos sistemas operativos existentes para procesadores ARM.

Entre otros sistemas operativos para procesadores ARM se pueden mencionar uCLinux, Partikle, eCos, uCos II, IAR POWERPACK y otros. El estudio de los sistemas operativos se basó en características como tamaño, tipo, soporte, tareas y prioridades, tipo de código y precio. Se decidió trabajar con el sistema operativo IAR POWERPACK por su amplia documentación y ejemplos, además la posibilidad de simulación en la plataforma EWARM.

4.4 Implementación de la solución

Para poder instalar el sistema operativo IAR POWERPACK en el kit de desarrollo LPC L2294 se tuvieron que realizar varias tareas previas, entre ellas entender el uso de las plataformas de desarrollo IAR EWARM y Eclipse, estas utilizan lenguaje C estándar y algunas otras consideraciones para procesadores ARM. Se investigó el procedimiento de cómo los compiladores generaban los códigos hexadecimales, que son de utilidad para poder grabar los programas en la memoria flash del microcontrolador.

Por otro lado se tuvo la necesidad de estudiar los registros de función especial del microcontrolador LPC2294 (SFR’s), esto para poder comprender aspectos varios de la arquitectura.

Como primeras pruebas se configuraron los diversos parámetros del PLL interno para funcionamiento del reloj del sistema y fuentes de reloj para los periféricos como el puerto USB, UART, ADC's y otros.

Se probaron los registros para configuraciones de pines de entrada salida y se configuró la función específica de cada pin. También se configuraron los temporizadores, dos protocolos de comunicación (RS232 y SPI) y como último paso se trabajó con el controlador vectorizado de interrupciones (VIC).

Los pasos anteriores sirvieron para entender el funcionamiento de la arquitectura de estos microcontroladores, esta información es de gran valor ya que luego se entendería las acciones del sistema operativo sobre el hardware.

Se decidió utilizar el sistema operativo IAR POWERPACK. Este sistema operativo maneja 127 prioridades, es un OS desplazante, también presenta una guía de uso que permite conocer aspectos básicos de sistemas operativos, además tiene la ventaja de tener un plugin en la plataforma de desarrollo EWARM que permite observar en tiempo de ejecución los valores de los diferentes registros, tareas, semáforos, buzones, eventos, temporizadores por software.

Las pruebas a las tareas fueron orientadas a uso de recursos como el puerto RS232, para esto se hizo uso de semáforos, buzones y eventos.

La desventaja es que al ser una versión de evaluación solo permite crear tres tareas máximo.

4.5 Reevaluación y rediseño

Por razones de defectos de fábrica de la tarjeta de desarrollo LPC L2294 en este proyecto se tuvo la necesidad de trabajar con una tarjeta LPC P2378, esta tarjeta incorpora el microcontrolador LPC2378, que es de una familia de microcontroladores

similar a la familia LPC2294. La tarjeta LPC P2378 tiene la desventaja de no poseer memoria externa. Por eso debe considerarse adquirir de nuevo otra tarjeta de desarrollo LPC L2294 para poder hacer uso del EMC del microcontrolador LPC2294 y probar la memoria externa. Sistemas operativos como uCLinux utilizan esta memoria para montar su imagen que tiene un tamaño aproximado de 1 MB.

Por otro lado es necesario instalar otros sistemas operativos. Ya se tiene una lista con las características de los OS existentes para procesadores ARM, también se tiene la base de la arquitectura, lo que ahorra tiempo de investigación y brinda las bases de lo que el sistema operativo hace. Es importante trabajar en la tarea de grabación de datos a memoria flash o RAM, esto para que los algoritmos de detección hagan uso de esta información.

Si se desean hacer pruebas de rendimiento se deben adquirir interfaces JTAG que permitan la depuración en tiempo real y medición de parámetros de los diferentes sistemas operativos.

No se trabajó con los controladores SPI para memorias tipo SD/MMC (secure device, multimedia card), estas memorias también permiten montar imágenes de sistemas operativos y son otra opción para guardar las mediciones y cálculos de sistemas los sistemas operativos. Por tanto es un punto a evaluar para determinar que tan eficiente y útil es para el sistema.

Capítulo 5 Descripción detallada de la solución

5.1 Descripción del hardware

Una vez definida las características del nodo sensorial y con el procesador

ARM7TDMI seleccionado como unidad de proceso central, se procedió a escoger un microcontrolador LPC2294 (con equivalente LPC2378) que incorpora entre otras características 8 ADC's, protocolos de comunicación (RS232, USB, ETHERNET, I2C, SPI, SPP), esto incorporado al kit de desarrollo LPC L2294 con 4MB de memoria RAM externa y con 8MB de memoria flash externa.

A continuación detalles específicos de este controlador.

5.1.1 Arquitectura LPC23xx [50]

En el siguiente diagrama se muestra la arquitectura del microcontrolador LPC2378 utilizado para el desarrollo del proyecto.

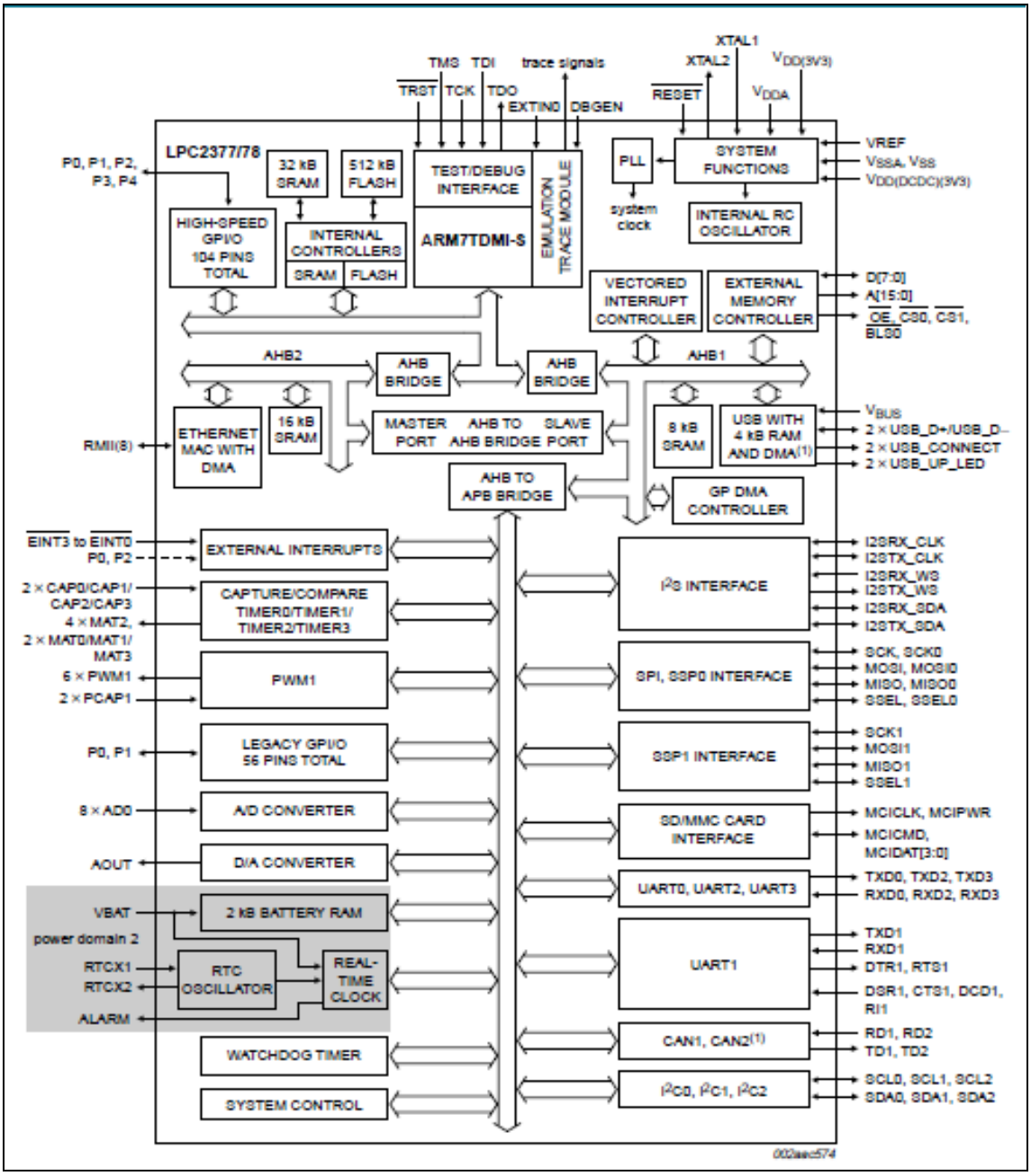

Figura 10 Arquitectura microcontrolador LPC2378

De la figura anterior cabe destacar varios puntos:

1. Núcleo ARM7TDMI-S.
2. Puentes de Buses AHB[11] a buses APH[12] y viceversa: estos puentes

[11] AHB = bus avanzado de alto desempeño, permite conectar los núcleos ARM con periféricos,

permiten interfazar los periféricos (ADCs, UARTs, etc.) con la memoria principal y el procesador ARM7TDMI-S.

3. Interfaz Ethernet con memoria de 16 kB que permite interconectarse con un controlador externo de Ethernet.
4. Controlador interno para USB 2.0 a 48 MHz.
5. Tres fuentes de reloj: Oscilador de cristal para alta frecuencia XTAL, cristal para reloj de tiempo real de 32768 Hz, oscilador interno RC. Se tiene la funcionalidad de PLL integrado para multiplicación de frecuencia con un CCO (oscilador controlado por corriente).
6. VIC (controlador vectorizado de interrupciones) para manejo de 32 interrupciones por hardware.
7. Ocho entradas para convertidor analógico digital con entrada de 0 a 3V, en 10 bits y tiempo de conversión de 2.34 us.
8. Convertidor digital analógico de 10 bits.
9. Interfaces SPI, I2C, SPP.
10. Controlador externo de memoria (EMC) para 2 bancos de 16 kB.
11. Alimentación externa de 3.3 V para periféricos y de 1.8 V para núcleo ARM.
12. Cuatro temporizadores de 32 bits.
13. Cuatro pines para interrupción externa, INT0 e INT1 son pines para sacar el CPU del modo sleep.
14. Interfaces JTAG y TRACE para programación y depuradorager.

5.1.1.2 Mapa de memoria.

En la siguiente figura se puede observar el mapa de memoria del microcontrolador LPC2378, este procesador tiene la característica de iniciar en la dirección cero luego

---

memoria e IPs.

[12] APH = es un bus más sencillo que el AHB y está diseñado para interfazase con timers, contadores, UARTs y otros.

de un reinicio o al encendido del chip. En los últimos 512 kB de memoria maneja todos los registros de función especial (SFR's).

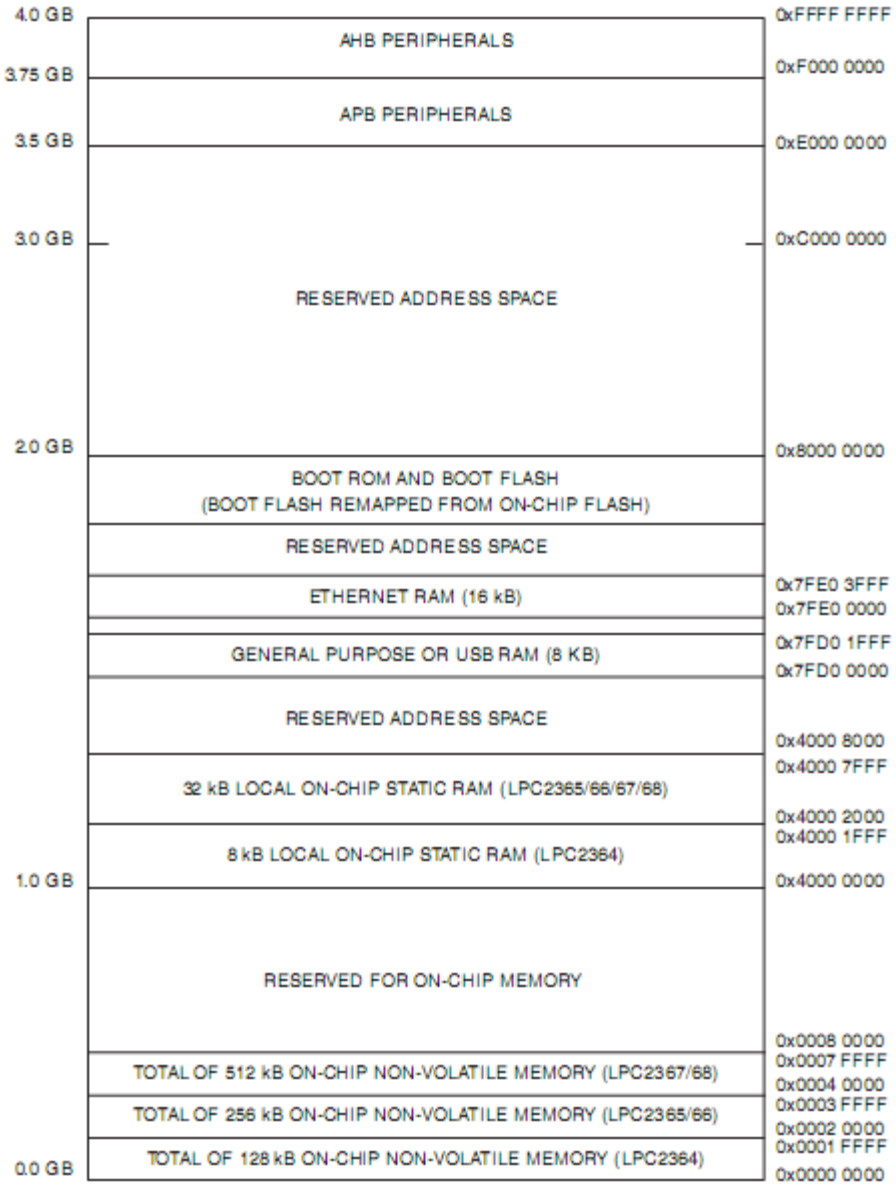

Figura 11 Mapa de memoria para LCP2378.

5.1.1.3 Configuración para PLL.

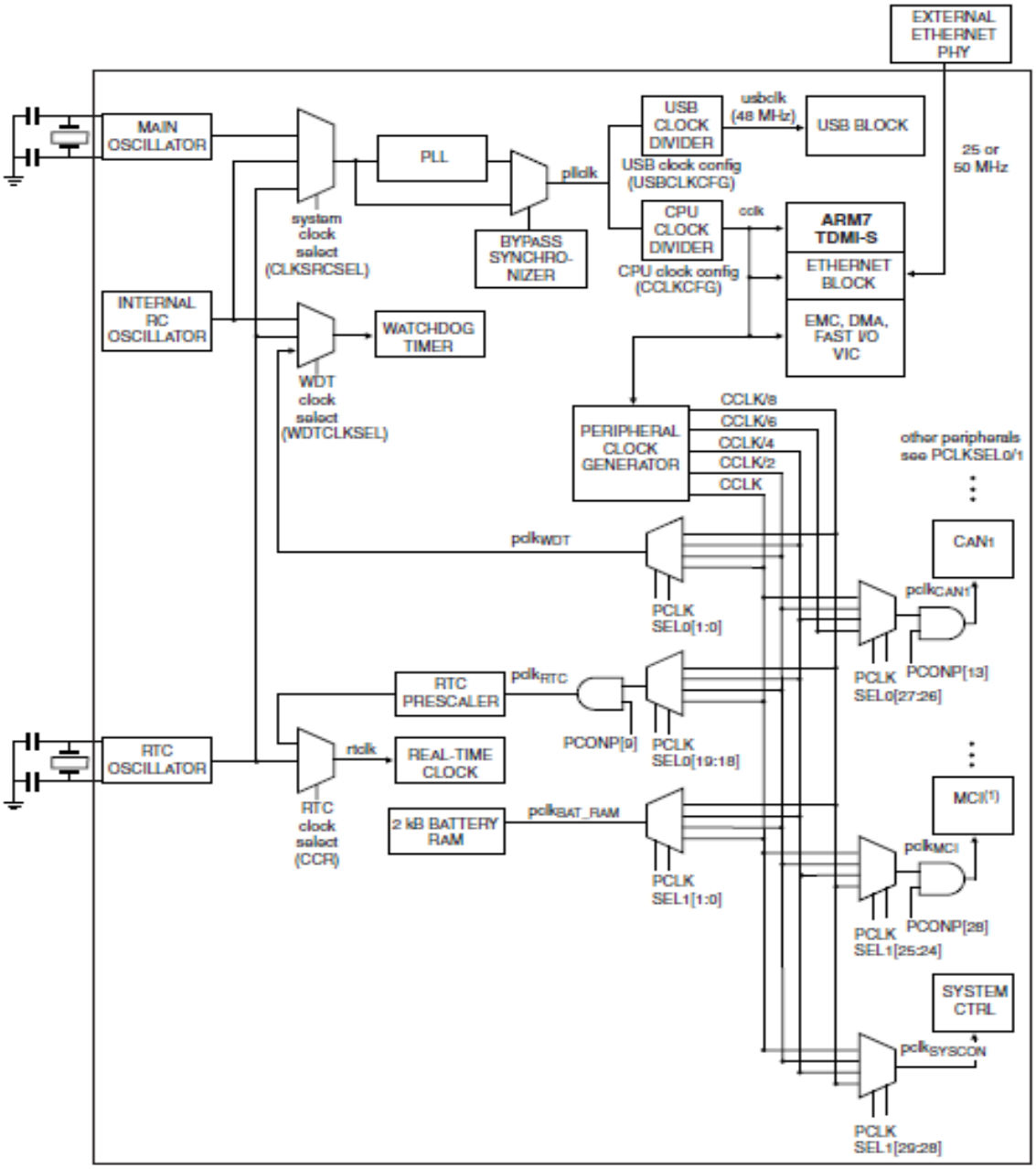

Figura 12 PLL y etapas de reloj para sistema LPC2378.

Se puede observar del diagrama de la figura 12 que el sistema tiene tres fuentes de reloj: el RTC (32768 Hz), el cristal XTAL externo y el oscilador RC interno (4MHz nominales) que pueden seleccionarse a través del registro CLKSRCSEL (0xE01FC10C). El bloque PLL puede utilizarse o no, haciendo uso del registro PLLCON (0xE01FC080).

El PLL admite frecuencias de entrada en el rango de 32kHz a 50 MHz. Estas frecuencias pueden dividirse por un valor N con rango de [1, 256]. El resultado de esta división provee una gama de 256 diferentes frecuencias de salida a partir de una misma frecuencia de entrada.

La frecuencia luego del divisor puede multiplicarse por un factor M en el rango de [1,32768] utilizando un CCO (oscilador controlado por corriente), la frecuencia resultante debe estar en un rango de [275,550] MHz. El multiplicador funciona dividiendo la salida del CCO por el valor de M, y luego utilizando un circuito detector de fase y frecuencia para comparar el valor de salida del CCO. El valor del error es usado para ajustar la frecuencia del CCO.

Hay divisores adicionales a la salida del PLL para llevar la frecuencia a un valor tan bajo como lo requiera el CPU, el bloque USB u otros periféricos.

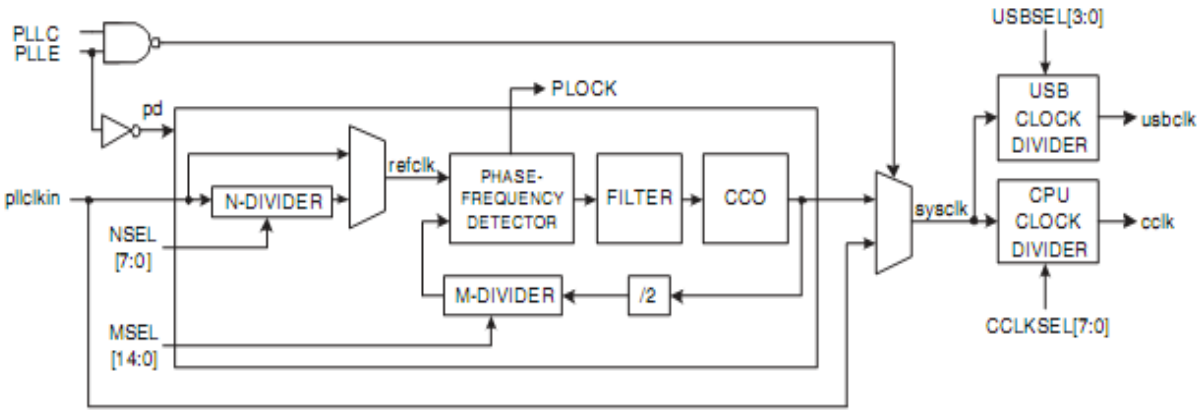

Figura 13 Diagrama de PLL para LPC2378.

La frecuencia de salida del PLL se calcula con la siguiente formula:

$$F_{CCO} = \frac{2 * M * F_{IN}}{N} \qquad (2)$$

Los siguientes registros están relacionados con el funcionamiento del PLL:

- SCS (0xE01FC1A0) R/W: registro de control y estado del sistema, permite establecer las conexiones del oscilador principal (XTAL).
- PCLKSRCSEL (0xE01FC10C) R/W: registro para selección de fuente de reloj

del PLL.

- PLLCON (0xE01FC080) R/W: registro de control de PLL, este registro activa y conecta el PLL, los valores de este registro no se actualizan hasta que no se dé una secuencia PLLFEED.
- PLLCFG (0xE01FC084) R/W: registro de configuración de PLL, en este registro se escriben los valores de MSEL (M-1) y NSEL (N-1), de igual manera necesitan una secuencia PLL feed para actualizar sus valores.
- PLLSTAT (0xE01FC088) RO: registro de lectura para conocer estado del circuito PLL, permite saber si el lazo PLL está conectado y cerrado.
- PLLFEDD (0xE01FC08C) W/R: registro para cargar y actualizar los valores en los registros PLLCON y PLLCFG. La secuencia PLLFEED es primero escribir 0xAA y luego 0x55 a este registro.

En la figura 14 se presenta un diagrama de flujo para la inicialización del PLL. Los valores en los diferentes registros son un ejemplo para tener una frecuencia de salida en el PLL de 288MHz, tener un reloj del sistema en 56.8 MHz y un reloj de USB en 48MHz. Se debe recordar que la frecuencia máxima de operación del núcleo ARM7 es de 60MHz.

El circuito divisor para frecuencia del sistema divide la frecuencia de salida del PLL por un factor CLKCFG (0xE01FC104) que tiene valores en un rango de [0,255], se recomienda que tenga un valor 0 o impar, en caso de tener un valor par no se asegura el correcto funcionamiento del sistema.

Los registros PCLKSEL0 y PCLKSEL1 se utilizan para escoger la frecuencia de operación de los diversos periféricos después del circuito generador de frecuencias, este circuito tiene como entrada la salida del circuito divisor de reloj del sistema y divide la frecuencia por factores de 1, 2, 4 y 8.

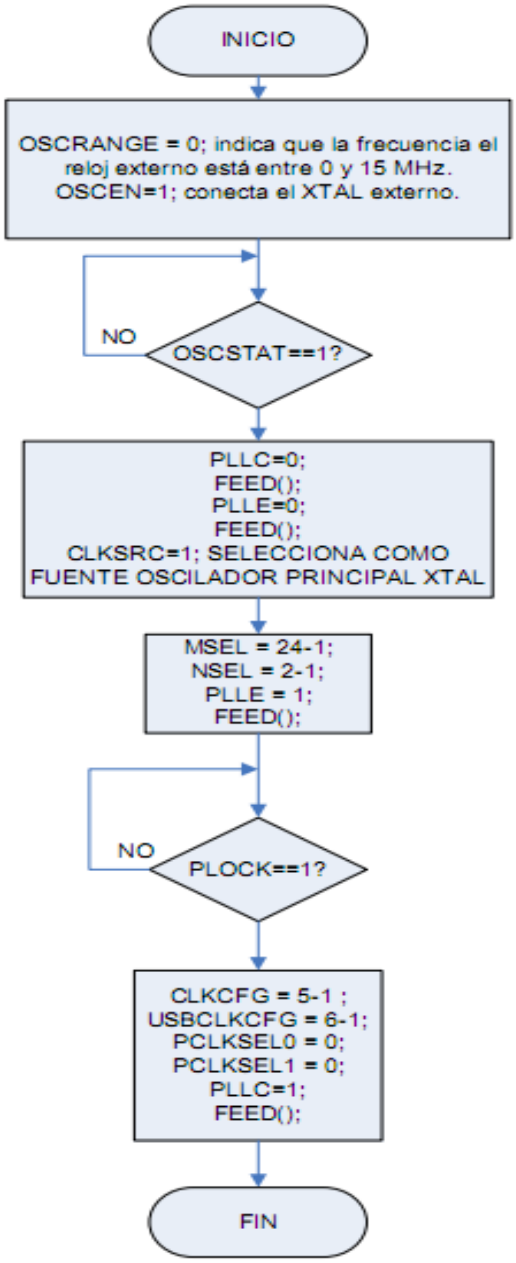

Figura 14 Diagrama de flujo para inicialización de PLL con reloj externo XTAL.

5.1.1.4 Puertos I/O

El CI LPC2378 es un microntrolador y por tanto multiplexa los pines como entradas y salidas. Cada pin maneja de una a cuatro funciones. Los pines de A0 a A15 y D0 a D8 son exclusivos para uso del controlador externo de memoria (EMC).

A continuación los registros de configuración de los pines de I/O.

- (F)IOxDIR (x = [0,4]), selecciona en cada bit si un pin es entrada o salida, con 0 es una entrada y con 1 es una salida.
- (F)IOxSET (x = [0,4]): escribe uno a los bits seleccionados de un registro IO.
- (F)IOxCLR (x = [0,4]): escribe cero a los bits seleccionados de un registro IO
- (F)IOxPIN (x = [0,4]): permite escribir o leer a un registro IO.
- IOxMASK (x = [0,4]): cuando se escriben unos a los bits se inhibe cualquier acción de escritura en los registros de entrada salida.
- PINSELx (x = [0,10]): estos registros escogen una de cuatro funciones que tiene cada pin del microcontrolador.

5.1.1.5 ADC

El microcontrolador LPC2378 incorpora seis canales para conversión analógica digital, multiplexa estas entradas para un solo un convertidor analógico digital de 10 bits que funciona con un reloj máximo de 4,3 MHz, el máximo tiempo de conversión con 10 bits a esta frecuencia es de 2,44 μs. Se presenta la opción de cambiar la cantidad de bits de conversión, y de iniciar la conversión por diferentes métodos. Este módulo presenta la característica de poder apagarse a través de un bit del registro PCONP.

5.2 Descripción del software

5.2.1 Plataforma de desarrollo

Durante el desarrollo de este proyecto se decidió trabajar con dos plataformas de desarrollo, esto bajo el sistema operativo Windows XP. Las plataformas son IAR EWARM bajo su paquete de evaluación de 32 kB y Yagarto IDE bajo su plataforma Eclipse. EWARM utiliza un compilador propietario y Eclipse utiliza una emulación del arm-elf-tool GNU en Windows.

Ambas plataformas son de utilidad si lo que se desea es realizar aplicaciones que pongan a disposición los diversos periféricos y utilidades de los procesadores disponibles. Además en ambas se trabaja bajo el lenguaje de programación C estándar. Existe una desventaja con la plataforma IAR y es cuando se trabaja con códigos de tamaños superiores a 32 kB ya que IAR EWARM no permite crear códigos mayores a este tamaño, es en este punto que herramientas de software libre como GNU ARM-ELF-TOOL se convierten en una alternativa, ya que permiten generar códigos para procesadores ARM7 y ARM9, sin importar el tamaño de los códigos y es gratuito. Incorporando esta utilidad a un IDE como Yagarto-Eclipse se tienen las herramientas necesarias para el desarrollo de aplicaciones para procesadores ARM. Eclipse es una plataforma que permite trabajar tanto en Microsoft Windows como en Linux. Eclipse es una buena alternativa si lo que se desea es trabajar con software libre.

5.2.2 IAR POWERPACK

Para este proyecto se trabajó con este sistema operativo debido a que la plataforma de desarrollo IAR EWARM tiene soporte para este sistema operativo en tiempo de ejecución y depuración, esto permite observar en tiempo de simulación las tareas, colas y pilas, temporizadores y variables, esto lo hace ideal para iniciar con un sistema operativo. Otro aspecto importante es que brinda una amplia documentación para la creación y destrucción de tareas, manejo de recursos y archivos.

Este sistema operativo al igual que otros está diseñado para funcionar con diferentes puertos (ports). Un puerto es una arquitectura específica, en este caso se utilizó para el puerto de la familia LPC23xx.

El sistema operativo IAR POWERPACK cuando se utiliza de manera gratuita permite máximo la creación de tres tareas, y el manejo de un archivo. Los creadores de este sistema operativo no brindan el núcleo del sistema (kernel), si se desea tener acceso a este debe comprarse una licencia. Para el usuario es transparente el kernel, se pueden observar algunas instrucciones ensambladas del kernel solo en tiempo de ejecución o de simulación. Existe un archivo “RTOS.h” el cual brinda la definición de funciones y variables utilizadas por el sistema operativo.

Capítulo 6 Descripción de los resultados obtenidos

6.1 Resultados Experimentales

6.1.1 Inicialización del sistema operativo

Para la operación del sistema operativo se sigue siguiente secuencia en la rutina del main:

1. Se inhiben las interrupciones.
2. Inicia el kernel del sistema operativo. Se debe crear el bloque del tamaño de memoria dinámica que utilizará el sistema operativo.
3. Se inicia el hardware: Inicia el bloque VIC (controlador vectorizado de interrupciones) e inicializa vectores de interrupción, inicializa PLL, puertos, y drivers necesarios como UART, SPI, y otros.
4. Se crean las colas, buzones y semáforos, si es el caso de que existan.
5. Se crean las tareas a manejar por el sistema operativo.
6. Se inicia el sistema operativo.

En este caso el sistema operativo da la opción de configurar el PLL usando el IRC o el XTAL externo, con un valor de M = 36 y N = 1 en el primer caso, en el segundo caso toma M = 12 y N = 1.

El sistema operativo interrumpe el sistema a través de las interrupciones del temporizador 0.

6.1.2 Creación y destrucción de tareas.

Antes de crear una tarea se deben tener en cuenta varios aspectos, entre ellos:

- Una tarea no recibe ni retorna ningún tipo de parámetro. Sintaxis: static void Tarea (void).
- Una tarea nunca debe llamarse directamente de la rutina principal o alguna parte del programa.
- Una tarea debe ejecutarse en un ciclo infinito.

- Sintaxis de la tarea:

```
static void Tarea (void){
                    while (1){
                    dosomething ();
                    OS_Delay (x); //Espere un tiempo “x” en
                                  //milisengundos para
                                  //ejecutarse de nuevo.
                    }
                    }
```

- Una tarea debe tener una pila para guardar punteros y registros. Sigue la siguiente sintaxis: OS_STACKPTR int PILA [x].
- Una prioridad de 0 a 255.
- Una TCB para control y estados de la tarea. Sintaxis: OS_TASK TCB.
- Un nombre para uso del sistema operativo. Sintaxis: “Nombre de la tarea”.
- Y se sigue el siguiente esquema para la creación de la tarea:
  OS_CREATETASK(&TCB, "Nombre de la tarea", Tarea, Prioridad, Pila);

Si se desea destruir una tarea se debe aplicar el siguiente procedimiento: OS_Terminate (Tarea).

La función OS_Delay(x) permite especificar un tiempo de espera en milisegundos para la tarea de 0 a 65535.

6.1.3 Uso de periféricos y sincronización entre tareas

6.1.3.1 Semáforos.

Cuando se quiere que dos o más tareas utilicen un periférico sin choque entre ellas se debe utilizar alguno de los mecanismos que incorpora el sistema operativo. En especial se utilizan los semáforos de recursos y los semáforos contadores, el primero se utiliza para que una tarea se apropie de un recurso, la segunda es una señal entre tareas para indicarle a otra tarea cuando se desocupa un periférico.

Sintaxis:

Declaración: OS_RSEMA SEMAFORO;

Rutina principal: OS_CREATESEMA(&SEMAFORO);

Uso por parte de la tarea con mayor prioridad o la que se apropio primero del recurso:

```
OS_Use(&SEMAFORO);
Rutina_Periferico();
OS_Unuse(&SEMAFORO);
```

6.1.3.2 Buzones.

Los buzones (mailboxes) son estructuras de datos que utiliza el sistema operativo para comunicar tareas y transferir datos entre ellas en tiempo real. Básicamente un buzón es un buffer que es manejado por el sistema operativo. Este buffer se comporta como un buffer normal, se colocan datos en el buffer (mensaje) y luego pueden recuperarse. Un buzón maneja una estructura FIFO (primero que entra, primero que sale). POWERPACK puede crear mensajes con un tamaño entre 1 y 127 bytes y maneja de 1 a 32767 buzones.

Los buzones típicamente se utilizan para tener buffer de teclas presionadas en un teclado. Una tarea, un temporizador por software o un manejador de interrupción chequea el teclado. Cuando se detecta una tecla presionada se procede a guardar el dato de la tecla dentro del buzón. El mensaje luego es recuperado por la tarea que se encarga de manejar la entrada de datos del teclado. Lo normal en estos casos es usar un buffer de un byte.

Otra aplicación típica es un buffer serial de E/S. En la mayoría de situaciones esto es hecho por un manejador de interrupción. La comunicación con estos manejadores se realiza a través del uso de buzones. Los manejadores de interrupción y las tareas se encargan de almacenar y recuperar datos a y desde un mismo buzón.

Un buzón requiere crear un objeto tipo OS_MAILBOX y un buffer.

OS_MAILBOX MAILBOX; char buffer [tamaño].

Sintaxis de creación: OS_CREATE (OS_MAILBOX, tamaño del mensaje, número máximo de mensajes, buffer);

Para hacer uso del buzón se tienen varias funciones:

Funciones para colocar datos en un buzón:
OS_PutMail / OS_PutMail1 / OS_PutMailCond / OS_PutMailCond1
Funciones de lectura para sacar datos desde un mailbox:
OS_GetMail / OS_GetMail1 / OS_GetMailCond / OS_GetMailCond1

6.1.3.3 Temporizadores por software

POWERPACK ofrece interrupciones por software a través de temporizadores. Los temporizadores por software ejecutan tareas específicas cada vez que llegan a la cuenta establecida para el temporizador. Cada vez que se cumple la cuenta un temporizador debe reinicializarse.

Los temporizadores pueden iniciar su cuenta de dos maneras, una de forma directa con la función OS_CREATETIMER. Si se utiliza la función OS_CreateTimer con la función OS_StartTimer() se puede iniciar un temporizador.

Con la función OS_StopTimer() se detiene la cuenta de un temporizador.

Sintaxis para temporizador con inicialización implícita:

```
OS_TIMER TIMER; // objeto tipo temporizador
void Tarea() { OS_RetriggerTimer(&TIMER);}
```

El temporizador se debe inicializar en la rutina principal:
OS_CREATETIMER(&TIMER, Tarea, Tiempo en milisegundos del timer);

6.1.3.4 Eventos

Existen otros métodos de sincronización entre tareas y son los eventos, los eventos son asíncronos. El propósito de un evento es hacer esperar una tarea por un evento en particular o varios. Una tarea puede suspenderse hasta que suceda un evento marcado por otra tarea, un temporizador por sofware o un manejador de interrupción. Un evento puede ser cualquier acción como un cambio en una señal, la expiración de un temporizador, una tecla presionada, la recepción de un carácter. Cada tarea tiene un byte de máscara (8 bits disponibles), lo que quiere decir que una tarea tiene ocho diferentes eventos para distinguir.

Haciendo uso de la función OS_WaitEvent(), una tarea espera por alguno de los bits de la máscara, si se da el evento la debe indicarse con la función OS_SignalEvent(), inmediatamente la tarea que estaba esperando ya puede ejecutarse, como es obvio será ejecutada según las reglas del programador tan pronto como sea la tarea de más alta prioridad lista para ejecutarse.

Sintaxis para marca y espera de un evento:

Tarea que espera evento:
OS_WaitEvent (0bxxxx xxxx); // espera por un evento o varios

Tarea que señala evento:
OS_SignalEvent(0bxxxx xxxx, &TCB_Tarea_En_Espera); //se marca un evento en cualquier bit del 0 al 8 y se indica la tarea que lo espera.

6.2 Análisis de resultados

La escuela de ingeniería en electrónica ha incursionado en el diseño de un nodo sensorial. Para este nodo sensorial se establecieron las características mostradas en la tabla 2, esto a partir de los requerimientos de las aplicaciones desarrolladas para la detección de fuego, sonidos de motosierras y disparos y otras disposiciones.
Una vez establecidas las características del nodo se procedió a investigar cual procesador cumplía los requerimientos para el nodo sensorial. Una característica especial de este procesador y por las necesidades de la aplicación es que consumiera muy baja potencia, por ello la investigación para determinar cual era el procesador deseado del nodo sensorial se centró en el mercado de los sistemas empotrados portátiles, donde se determinó que los procesadores de la marca ARM tipo RISC son los procesadores de 32 bits de menor consumo de potencia, son procesadores con consumos de potencia en el orden de los mW, abarcan un 75% del mercado de aplicaciones portátiles y poseen un eficiente set de instrucciones. Estos procesadores en general tienen un buen balance entre consumo de potencia y capacidad de procesamiento.

Un procesador ARM7TDMI con frecuencia de operación de 100 MHz consume alrededor de 25 mW y puede ejecutar 90 MIPS. Una de las aplicaciones más conocida que utilizan procesadores ARM es el reproductor de video y música iPod de Apple que incluye un procesador ARM7 v4.

Como se puede observar de la tabla 3 los núcleos ARM son los procesadores con menor consumo de potencia, por esta razón se decidió que éste fuera el núcleo de la unidad de proceso central del nodo sensorial, específicamente se escogió un procesador con arquitectura ARM7 que es el de menor consumo de potencia de los procesadores de las familias ARM.

A través de investigaciones se encontró que fabricantes como Philips, Atmel, Cirus y otros, empotran procesadores ARM a sistemas microcontrolador, esto debido a que los procesadores ARM están orientados al mercado de las aplicaciones empotradas portátiles. Estos diseños de microcontroladores incluyen interfaces de comunicación

como RS232, SPI, CAN, ETHERNET, USB y otros, además de convertidores analógicos digitales, señales IO, unidades de manejo de memoria externa (EMC), que son parte de los requerimientos para el nodo sensorial.

Para acelerar el tiempo de desarrollo de la aplicación y tener un set de periféricos disponibles en un solo paquete se decidió adquirir un kit de desarrollo con un mínimo de recursos necesarios para la ejecución de los algoritmos desarrollados de detección, por eso se escogió para la compra un kit de desarrollo marca olimex modelo LPC L2294, el cual trae incorporado un microcontrolador ARM7TDMI-S, el hardware necesario para interfaces RS232, Ethernet, CAN, SPI, SD/MMC, además de 4MB de memoria RAM y 8 MB de memoria Flash externa, interfaz para depuración JTAG, convertidores analógicos digitales con tiempo mínimo de muestreo de 2,44 µs para muestras en 10 bits y conectores de buses externos. LPC2294 es un microcontrolador marca Philips capaz de manejar 4 bancos de 16 MB externos, también incluye modos de ahorro de energía como idle, stand by y sleep que son requeridos por la aplicación.

El mercado de los microcontroladores con núcleo ARM es altamente difundido y existen diversas plataformas de desarrollo para estos. Algunas de ellas IAR EWARM, uKeil vision 3, Yagarto. IAR EWARM y uKeil vision 3 ofrecen versiones gratuitas que permiten desarrollo de aplicaciones de 32 kB de código como máximo, las licencias para estos programas tiene valores mayores a $900. La principal ventaja del uso de estos programas es el soporte en aplicaciones, datos técnicos, definición de registros de función especial, simuladores, depuradores y otras características. Yagarto por su lado es una plataforma gratuita que en conjunto con las herramientas ARM-ELF-TOOL se convierte en un ambiente gráfico de desarrollo y depuración ya sea en Windows o en Linux para procesadores ARM7 y ARM9. Existe una gran número de aplicaciones desarrollas para este ambiente de desarrollo y su compilador ARM, es un software gratuito y no limita el tamaño de las aplicaciones. Para el desarrollo y pruebas del proyecto se trabajó con las plataformas EWARM y Yagarto bajo el lenguaje de programación C, ambas son de gran utilidad si lo que desea es el

desarrollo de aplicaciones de forma gratuita.

Una vez que se adquirió el kit de desarrollo y se escogieron las plataformas de desarrollo EWARM y Yagarto se trabajó en el estudio y la programación de módulos I/O, multiplexación de entradas/salidas, selección de funciones en cada pin, configuración de PLL interno y frecuencia de operación, manejo de periféricos como el UART, monitores de señales analógicas, manejo de interrupciones, prioridades y vectores de interrupción para el microcontrolador.

Al ser los procesadores ARM parte de un mercado de aplicaciones empotradas existen diversos sistemas operativos implementados para este tipo de procesadores, orientados al eficiente uso de los recursos del sistema en tiempo real. La tabla 10 muestra algunos sistemas operativos desarrollados para núcleos ARM7, que son núcleos sin unidad de manejo de memoria (MMU).

Como una primera prueba se decidió trabajar con el sistema operativo IAR POWERPACK que en conjunto con la plataforma EWARM permite observar el comportamiento del sistema operativo en tiempo de ejecución, además de tareas, temporizadores y otros.

POWERPACK es un OS reemplazante que cuenta con funciones como creación y destrucción de tareas, asignación de prioridades, manejo de pilas por software, manejo de recursos a través de semáforos, comunicación de tareas a través de buzones, colas y eventos, interrupciones por software con temporizadores, drivers de periféricos como UART, manejo de distintos tipos de datos, instalación de interrupciones vía hardware, manejo de archivos en memoria, modos de manejo de energía. Todas estas características en conjunto con la plataforma IAR EWARM se convirtieron en los factores de peso para escoger el sistema operativo POWERPACK como el primer sistema operativo en ser montado y ejecutado en la aplicación.

Las diversas pruebas lograron demostrar como el sistema operativo gana control sobre el uso del procesador y los periféricos. Con el uso de un sistema operativo los recursos de un procesador se utilizan de forma paralela entre tareas, estas tareas adquieren el control del CPU por un lapso y por orden de prioridad, utilizan los recursos del procesador de forma ordenada a través de algoritmos de comunicación entre tareas.

Cuando se tienen tareas de igual prioridad el sistema operativo utiliza el algoritmo Round-Robin y porciones de tiempo específicas para cada tarea para decidir cuál tarea se ejecuta y por cuánto tiempo. Esto le permite al sistema no tener conflictos entre tareas de igual prioridad.

Capítulo 7 Conclusiones y recomendaciones

7.1 Conclusiones

1. Se determinaron las características requeridas para desarrollo del nodo sensorial de la escuela de ingeniería electrónica del ITCR.
2. Se escogió el procesador ARM7TDMI-S de 32 bits como unidad de proceso central del nodo sensorial, que es un procesador con consumo de potencia en el orden de los mW, de los más bajos del mercado.
3. Se adquirió un kit de desarrollo LPC L2294, con microntrolador LPC2294 (núcleo ARM/TDMI-S). Este kit de desarrollo reúne todas las características necesarias para desarrollar las aplicaciones de detección, administración y comunicación del nodo sensorial.
4. Se cuenta con plataformas de desarrollo como IAR EWARM, uKeil Vision3 y Yagarto para el desarrollo de las aplicaciones del nodo sensorial.
5. Se logró adaptar el sistema operativo IAR POWERPACK al nodo sensorial en el paquete LPC2378.
6. Haciendo uso del sistema operativo se lograron hacer lecturas de un sensor a través y transmitir información haciendo uso de un protocolo de comunicación.
7. Un sistema operativo reemplazante y con algoritmo Round Robin como POWERPACK ejecuta tareas según la prioridad de estas y de forma paralela.
8. El sistema operativo brinda diversos algoritmos para la comunicación eficiente entre tareas.
9. Se tiene un sistema con posibilidad de depuración a través de interfaces JTAG, una herramienta muy utilizada en el mercado para

depuración de software.

7.2 Recomendaciones

1. Se deben adquirir interfaces JTAG para facilitar el rastreo de código.

2. Deben instalarse otros sistemas operativos para poder hacer pruebas de rendimiento y tener parámetros de comparación, y con esto determinar cuál sistema brinda el mejor rendimiento y menor consumo de potencia.

3. Al tratarse los algoritmos de detección aplicaciones con operaciones en punto flotante, deben considerarse dos opciones en caso de que no se desee ejecutar estos algoritmos en el procesador: una es buscar otro procesador de la familia ARM con unidad de punto flotante, o como es una idea inicial interfazar el procesador a un dispositivo con una capacidad de procesamiento fuerte y de bajo consumo energético como lo es un FPGA.

8. Bibliografía

[1] Feldnam J. Computer architecture, A designer's text based on a generic RISC, Editorial Mc Graw Hill, USA, 1994.

[2] Rodríguez C. Microprocesadores RISC, Evolución y tendencias. Editorial AlfaOmega, México, año 2000.

[3] URL: http://www2.canalaudiovisual.com/ezine/books/acjirINFORMATICA/3info05.HTM, consultado 20 de Julio de 2008.

[4] Vargas C. Arquitectura del microprocesador y lenguaje ensamblador, Universidad de Costa Rica, tercera edición, año 2004

[5] Arm instruction set, Quick Reference Card.

[6] Del Alba M. Estimación del Consumo de Potencia Dinámica en un Microprocesador Superscalar, Instituto Tecnológico y de Estudios Superiores de Monterrey, Marzo 2005.

[7] Hamacher C. Computer organization, 2da edición, Mc Graw Hill, USA, 1984.

[8] Dick R.: Power Analysis of Embedded Operating Systems, Departamento de ingeniería eléctrica, Universidad de Princeton, año 2000.

[9] Eidson J. A Research Prototype of Networked Smart Sensor System, Measurement Systems Department Instruments and Photonics Laboratory, August 1995.

[10] Kyildiz I.; Kasimoglu I. Wireless sensor and actor networks: research challenges, Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Enero 2004.

[11] Dowd K. y Severance C. High performance Computing, Segunda edición, Editorial O'Relly y asociados, Estados unidos, Año 1998.

[12] Vega J.: Arquitectura RISC vs CISC (en línea), Universidad Autónoma Metropolitana,

México, consultado 29 de julio de 2008, disponible en http://www.azc.uam.mx/publicaciones/enlinea2/num1/1-2.htm.

[13] Chen C., Novick G., Shimano K.: RISC Arquitecture, RISC vs CISC (en línea), Universidad de Stanford, Año 2000, disponible en http://cse.stanford.edu/class/sophomore-college/projects-00/risc/risccisc/, consultado 10 de julio de 2008.

[14] Dos Santos V. RISC vs. CISC – The Post-RISC, Departamento de Informática, Universidad de Minho, 2002.

[15] Stanley W. digital signal processing, segunda edición, Editorial Prentice Hall, año 1984, Virginia Estados Unidos.

[16] Lingxia C. A survey embedded operating systems.

[17] Baskiyar S.; Mejhatlam N. A survey of contemporary operating systems. Auburn University. USA. 2004.

[18] Zen H.; Toda T. An Overview of Nitech HMM-based Speech Synthesis System or Blizzard Challenge 2005 in Lisboa Portugal. Instituto de tecnología de Nagoya, Japan.

[19] Brey B. Los microprocesadores INTEL: arquitectura, programación e interfaz de los procesadores 8086/8088, 80186/80286, 80386, 80486, PENTIUM, PENTIUM PRO y PENTIUM II. Quinta edición, México, editorial Pearson Education, año 2001.

[20] Weiss A. Dhrystone Benchmark: History, Analysis, "Scores" and Recommendations, Noviembre 2002, ECL benchmarks Texas, California.

[21] R. Leonid. The ARM Architecture, cse Junio 2006.

[22] Processor Overview, (en línea), disponible **en** www.arm.com, visitado 15 mayo de 2008.

[23] Villalobos F. Leonardo, Captura de rayos Infrarrojos mediante una WEBCAM, Tesis de licenciatura, año 2006.

[24] Salas P. Walter, Diseño e implementación de un sensor para la detección de motosierras de la red inalámbrica de telecomunicaciones para la protección Ambiental en el bosque, Tesis de licenciatura, año 2004.

[25] Leiva R. José, Diseño de la etapa de detección de disparo de armas de fuego para una red inalámbrica de telecomunicaciones para la protección ambiental en el bosque, Tesis de licenciatura, año 2004.

[26] Sáenz M. Gabriela, Reconocimiento de patrones acústicos para la protección del ambiente utilizando wavelets y Modelos Ocultos de Markov, Tesis de licenciatura, año 2005.

[27] Smith C. Esteban, Reconocimiento digital en línea de patrones acústicos para la protección del ambiente por medio de HMM, Tesis de licenciatura, año 2007.

[28] Navarro M. Emmanuel, Diseño de un controlador embebido para cámara USB para la detección de incendios forestales, Tesis de licenciatura, año 2006.

[29] Slater Michael, The microprocessor today, 1996

[30] Pratt Ian, Comparative Architectures, 2004

[31] IEEE Computer society, (en línea), disponible en: http://www.computer.org/portal/site/ieeecs/index.jsp, visitado 10 de agosto de 2008.

[32] Sun Systems: Ultra Sparc Processors, (en línea), disponible en: http://www.sun.com/products/microelectronics/products.jsp, visitado 10 de agosto de 2008.

[33] IBM: Power Architecture offerings, (en línea), disponible en: http://www-01.ibm.com/chips/techlib/techlib.nsf/productfamilies/PowerPC, visitado 10 de agosto de 2008.

[34] HP: Alpha processors, Archived technical documentation library, (en línea), disponible en: URL: http://h18002.www1.hp.com/alphaserver/technology/chip-docs.html, visitado 10 de agosto de 2008.

[35] Langens T.: MIPS Technologies: MIPS, (en línea), disponible en: http://www.langens.eu/tim/ea/mips/mipsverslag.pdf, visitado 10 de agosto de 2008.

[36] Kits de desarrollo: Philips NXP, (en línea), disponible en: http://www.olimex.com, visitado 30 de setiembre de 2008.

[37] IAR Embedded Workbech for ARM, año 2006(en línea), disponible en: www.iar.com, visitado 10 de octubre de 2008.

[38] ukeil vision 3: (en línea), disponible en: http://www.keil.com/demo/, visitado 20 de octubre de 2008.

[39] Yagarto: (en línea), disponible en http://www.yagarto.de/, visitado 22 de octubre de 2008.

[40] PowerPac OS User Guide, año 2006, (en línea), disponible en: www.iar.com, visitado 15 de noviembre de 2008.

[41] Brinch H. Per, The Evolution of the operating Systems, año 2000, New York.

[42] Operating Systems, (en línea), disponible en: http://www.uow.edu.au/~nabg/ABC/C3.pdf, visitado 15 de noviembre de 2008.

[43] IAR Systems: RTOS, TCP/IP and USB, (en línea), año 2008, disponible en: http://www.iar.com/website1/1.0.1.0/352/1/index.php, visitado 17 de noviembre de 2008.

[44] Micrium: Operating System, (en línea), año 2008, disponible en: http://www.micrium.com/, visitado 20 de noviembre de 2008.

[45] Peiro S.: Partikle OS: a replacement for the coreo f the RT-Linux-GPL, (en línea), disponible en: http://www.e-rtl.org/partikle/fileshare/files/5/PaRTiKle-OS.pdf , visitado 25 de noviembre de 2008.

[46] FreeRTOS: Designed for microcontrollers, (en línea), disponible en: http://www.freertos.org , visitado 26 de noviembre de 2008.

[47] uClinux: Embedded Linux/Microcontroller Project, (en línea), disponible en:

http://www.uclinux.org , visitado 28 de noviembre de 2008.

[48] eCos: operating system, (en línea), año 2008, disponible en: http://ecos.sourceware.org , visitado 29 de noviembre de 2008.

[49] Salvo: the RTOS that runs uin tiny places, (en línea), año 2008, disponible en: http://www.pumpkininc.com , visitado 20 de noviembre de 2008.

[50] Khan J., Boser B.: SMART DUST Autonomous sensing and communication in a cubic millimeter, (en línea), año 2001, disponible en: http://robotics.eecs.berkeley.edu/~pister/SmartDust , visitado 17 de abril de 2008.

[51] Patterson, David A. and John L. Hennessy. Computer Organization & Design. 2nd ed. Morgan Kaufmann Publishers: San Francisco, 1998.