INSTITUTO TECNOLOGICO DE COSTA RICA

VICERRECTORÍA DE INVESTIGACIÓN Y EXTENSIÓN

DIRECCIÓN DE PROYECTOS

Y

ESCUELA DE INGENIERÍA ELECTRÓNICA

INFORME FINAL

PROYECTO

"Desarrollo tecnológico de un sistema de adquisición de datos ambientales para su uso en proyectos de investigación científica: Arquitectura abierta CRTecMote"

Código: 5402-1360-2201

INVESTIGADOR

Ing. Johan Carvajal Godínez

2009-2010

Contenido

Resumen

El avance en el desarrollo de la microelectrónica y las teorías de sistemas han hecho posible el diseño de complejos sistemas de monitorización con un mayor nivel de integración y una mayor eficiencia en el consumo energético. Gracias a esto, se ha logrado el desarrollo de tecnologías móviles que abren una ventana de oportunidad a aquellos proyectos que se habían visto limitados desde un punto de vista tecnológico.

Los países desarrollados han hecho una gran inversión en el desarrollo de tecnologías portátiles o móviles para aplicaciones militares, sin embargo, en el contexto de la realidad nacional, el enfoque que se ha dado a las tecnologías móviles va enfocado al desarrollo de sistemas de monitorización ambiental orientados a resolver tres problemáticas principales: la primera son las aplicaciones de monitorización y protección ambiental, la segunda aplicación es prevención de desastres mediante la predicción de problemas en la infraestructura nacional y la tercera es el desarrollo de indicadores de calidad de agua, con lo que se pretende mejorar la calidad de vida de los costarricenses.

A inicios del 2009 se comenzó con el desarrollo de una plataforma tecnológica de arquitectura abierta que hiciera posible desarrollar herramientas para atacar las tres prioridades que se expusieron anteriormente, es por ello que CRTECMote viene a ser la respuesta natural que satisface las necesidades de un conjunto de investigadores en torno al desarrollo de sistemas de medición inteligentes, escalables y de bajo costo.

Palabras Clave: Red Inalámbrica de Sensores, protocolos de comunicación, monitorización ambiental, Dispositivos de bajo consumo energético.

Abstract

The growth of microelectronic device technology and the development of system´s theory has made possible the design and implementation of complex monitoring systems, with a higher level of integration and more energy efficient. Thanks to these advancements, is that mobile technologies have become part of human lifestyles solving problems that in the past were impossible to overcome due to technological showstoppers.

The developed countries have made a huge investment on development of mobile technologies focusing on military and industrial applications; however in the national context it is important to use those for addressing environmental issues. Based on that reasoning, this project looks for focusing on three key applications: (1) environmental monitoring and protection systems, (2) natural disaster prevention systems, and (3) water monitoring systems.

The end goal of the project was to increase the quality of citizen´s life by providing a technological platform for addressing common issues.

The Project started on 2009 with de design of an open architecture for a wireless sensor network node, the continued with the implementation of an operative system for managing the node recourses and was completed with the successful implementation of an experimental sensor network setup with CRTECMOTE nodes.

These results satisfy and exceed the objectives defined at the beginning of the project and bring the opportunity to environmental researchers to develop smart metering systems for their application with the state of the art technology and with a low cost.

Keywords: Wireless sensor networks, communication protocols, environmental monitoring, low power electronic systems

Datos generales del proyecto

Nombre del proyecto: “Desarrollo tecnológico de un sistema de adquisición de datos ambientales para su uso en proyectos de investigación científica: Arquitectura abierta CRTECMote” Código 5402-1360-2201

Vigencia del proyecto: Enero 2009 a Diciembre 2010

Participantes:

	Nombre completo	**Jornada semestral [horas/semana]**	**Universidad**
Responsables	Lic. Johan Carvajal Godínez	20	ITCR
	Dr. Pablo Alvarado Moya	0	ITCR
Asistentes estudiantiles	Mario Alfaro Ortega	10	ITCR
	Norwing Leiva Delgado	10	ITCR
	Sebastián López González	10	ITCR
	Sofía Ortiz Argüello	10	ITCR
	Alexander Valverde Serrano	10	ITCR
	Dennis Rodríguez Rodríguez	10	ITCR
	Cristhian Villegas Benavidez	10	ITCR
	Mariano Jimenez Brenes	10	ITCR
	Miguel Fonseca Porras	10	ITCR
	Jose Pablo Fallas Zúñiga	10	ITCR
Personas que apoyaron el proyecto	Ing. Adolfo Chaves Jiménez	0	ITCR
	Dr. Ólman Murillo Gamboa	0	ITCR
	Dr. Carlos Meza Benavidez	0	ITCR

Capítulo I: Introducción

Introducción a las redes inalámbricas de sensores (WSN)

Una red inalámbrica de sensores (WSN; por sus siglas en inglés) consiste en un conjunto de dispositivos con capacidad del procesamiento, autonomía energética, y comunicación limitados. Dichos dispositivos se encuentran equipados con interfaces de tanto de comunicación inalámbrica como de medición de procesos o variables ("sensores") que le permiten configurarse como un sistema de adquisición de datos distribuido. [1] [2]

A los dispositivos antes mencionados se denomina nodos sensoriales o simplemente nodos de una red inalámbrica, y por lo que general se distribuyen espacialmente sobre una región bajo estudio, donde cada nodo es responsable de medir, codificar y transmitir información del entorno tales como humedad relativa, temperatura, presión, luminosidad, etc. Se podría también transmitir mensajes de eventos particulares determinados por la combinación de condiciones que disparen una condición de respuesta específica.

Una WSN se compone por un arreglo de nodos distribuidos espacialmente, que a lo interno se pueden caracterizar por bloques de operación con funciones específicas para:

- Procesamiento de datos
- Suplir energía para el consumo local
- Comunicación con otros nodos de la red
- Conectar sensores de variables particulares.
- Administrar los recursos de hardware y software del nodo.
- Proveer algoritmos de procesamiento especializado, ejemplo FPGA.

Los avances tecnológicos en microelectrónica, sistemas micro-electromecánicos (MEMS) y en sistemas empotrados registrados a partir del año 2005, han permitido extender el rango de aplicaciones de las WSN y han hecho posible su utilización en aplicaciones de monitorización de zonas agrícolas y forestales, con

el fin de tener una mejor comprensión y control sobre las variables que influyen en el comportamiento de entornos ambientales, tanto abiertos como cerrados.

Dentro de los retos que se plantean en las redes inalámbricas de sensores se encuentran la cobertura geográfica de los nodos de medición, así como su autonomía energética. También se debe un esfuerzo muy fuerte en incrementar la eficiencia de los protocolos de comunicación utilizados, pues las WSN generan y almacenan la información sensorial y utilizan sus conexiones inalámbricas para hacerla llegar a otros nodos, que pueden ser móviles o estáticos.

Considerando la característica anterior, se puede definir que los datos son los elementos que agregan valor al sistema, por lo tanto se requiere asegurar un nivel de integridad y confiabilidad a la hora de procesar y transmitir las mediciones realizadas. [3]

Las estrategias para mejorar la eficiencia en la utilización de la energía por parte de los nodos de medición, buscan desde la utilización de tecnología de bajo consumo eléctrico hasta sistemas adaptativos que se autoregulan en cuando a rendimiento con la finalidad de extender el tiempo de operación del sistema.

Para lograr hardware con menor consumo eléctrico se echa mano de los avances en la tecnología de manufactura de semiconductores [4]. Mientras que desde el punto de vista de optimización operativa, se modifica el software que se ejecuta en los nodos para la obtención de protocolos de comunicación que consuman la menor cantidad de energía posible para realizar una operación específica. [5]

Para mantener una operación adecuada los nodos están equipados con una reserva de energía que se implementa a través de uso de una pila o baterías, sin embargo es tarea del nodo obtener energía de alguna manera para garantizar la sostenibilidad de la operación de la red inalámbrica de sensores. Para ello se han explorado metodología de cosecha energética que utilizan sistemas de carga basados en tecnología fotovoltaica para obtener energía y almacenarla en sistemas reservorio de cargas eléctricas, tales como baterías o ultracapacitores y de esta manera satisfacer de energía el nodo de medición de la red. [6]

Las redes inalámbricas de sensores han surgido como una tecnología innovadora que se proyecta que cambiará la manera en que medimos en la actualidad. En la revista Technology Review ("Techreview") del MIT se seleccionó a las Redes Inalámbricas de Sensores (WSN) como una de las 10 tecnologías emergentes que cambiaran el mundo debido a su amplio espectro de aplicación para resolver problemas en la industria, salud, ambiente, urbanismo, prevención de desastre naturales, e investigación científica, entre muchos otros.

En cuanto al desarrollo de la tecnología de redes inalámbricas de sensores se encuentran trabajos de postgrado que han determinado paradigmas de diseño de aplicaciones de monitorización. Uno de los más reconocidos es fue el trabajo desarrollado por Jason Hill de Universidad de California en Berkeley [7] quien propuso una arquitectura de hardware y software para la implementación de los para nodos medición de una WSN. Su propuesta parte del hecho de que las redes inalámbricas de sensores son sistemas reducidos que generan eventos. Para administrar dichos sistemas plantea la implementación de un sistema operativo optimizado para operar en dichos nodos o polillas ("motes") que llamó con el nombre de TinyOS. Con este concepto funda una compañía dedicada a las redes inalámbricas de sensores (www.Xbow.com) que se encarga de desarrollos con redes inalámbricas de sensores para aplicaciones de trazabilidad de vehículos, paquetes y sistemas militares.

El tamaño de las redes inalámbrica de sensores puede variar pero los protocolos más usados en la implementación de dichas configuraciones establecen que se pueden tener redes de pequeña escala hasta redes de miles de nodos, con la oportunidad de que en el futuro se tengan redes de cientos de miles a millones de nodos.

Con lo discutido anteriormente se establecen factores de diseño [1] que incluyen una serie de consideraciones que se citan a continuación:

- Una red tolerante a fallas de alguno o algunos de los nodos
- Capacidad de escalabilidad y modularidad de los nodos
- Costos asociados a la manufactura

- Balance de las características de operación del nodo
- Topología de implementación de la WSN
- Capacidad de cosecha de energía y entorno de operación de la red

Es muy importante para caracterizar el problema de medición al que se quiere aplicar una red inalámbrica de sensores como solución, pues de ello dependerá la estrategia que se utilizará, dado que se pueden tener diferentes enfoques al implementar una red inalámbrica de sensores.

Para ello se define un árbol de clasificación de las redes inalámbricas de sensores según su aplicación que se presentan en la figura 1.

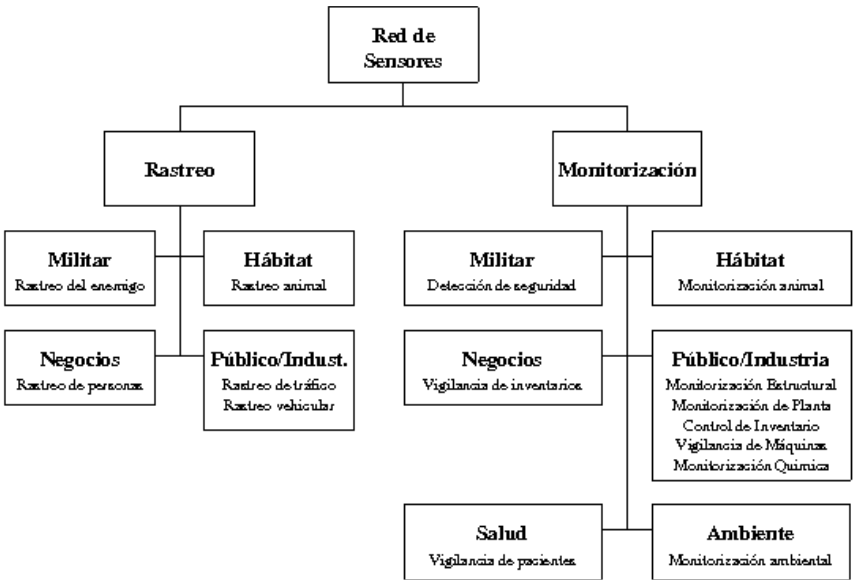


Figura 1. Clasificación de WSN según aplicaciones [2]

Adicionalmente de acuerdo a la estrategia de distribución de los nodos espacialmente se identifica dos categorías desde el punto de vista de la instalación de la infraestructura. Para ello habla de que una WSN puede verse como un sistema estructurado, cuando los nodos se instalan de forma planeada para maximizar el área de cobertura, con una cantidad de nodos no muy densa. La otra categoría asociada a la forma de instalar los nodos es la configuración no estructurada: caracterizada por un alto número de nodos no atendidos, instalados en forma no planeada, interconectados como en una red ad-hoc.[2]

Antecedentes de redes inalámbricas de sensores en el TEC

En el Instituto Tecnológico de Costa Rica (TEC), el desarrollo de redes inalámbricas para la monitorización de ambientes abiertos fue iniciada por el investigador M.Sc. Néstor Hernández Hostaller de la Escuela de Ingeniería Electrónica en el año 2004, con el proyecto denominado: “Diseño de una red inalámbrica de telecomunicaciones para la protección ambiental en el bosque”, el cual produjo prototipos de nodos de monitorización para la detección de fuegos forestales y caza furtiva.

A partir de las ideas surgidas en el “proyecto del bosque”, el Dr. Alfonso Chacón Rodríguez desarrollo su tesis doctoral denominada “Circuitos integrados de bajo consumo para detección y localización de disparos de armas de fuego”, el cual tenía como propósito el diseño de sensores para detectar la caza ilegal. La meta sería utilizar los sensores diseñados por el Dr. Chacón en la implementación de una red inalámbrica de sensores que pudiese alertar a los guardaparques sobre la presencia ilegal de cazadores en los parques nacionales de Costa Rica.

Como producto del estudio de los académicos antes mencionados se genera una publicación en sociedad de circuitos y sistemas de la IEEE en la que se discuten los algoritmos utilizados para la detección de disparos en entornos abiertos y su implementación por medio de técnicas de diseño de circuitos integrados. [8]

En paralelo a este trabajo se realizó otro proyecto liderado por el Dr. Pablo Alvarado Moya de la Escuela de Ingeniería Electrónica en conjunto con el Dr. Apolinar González y el Dr. Luis Villaseñor de la Universidad de Colima en México. El proyecto denominado “D2R” tenía como objetivo el desarrollo de una arquitectura de redes inalámbricas de sensores para ambientes protegidos y en campo. [9]

Otros académicos del ITCR entre los que nombro al Dr. Freddy Araya Rodríguez, Dr. Olman Murillo Gamboa y Dr. Carlos Meza Benavidez han venido propulsando la utilización de tecnologías de redes inalámbricas de sensores para su utilización en la monitorización, modelado y control de jardines de clonación, acueductos rurales y cultivos de área extensiva, por lo que puede afirmar que el estudio de la

redes inalámbricas de sensores, su mejoramiento y aplicación son una línea de investigación formal que el Tecnológico de Costa Rica persigue activamente.

La necesidad de una arquitectura abierta para la investigación científica

Cuando se habla de arquitecturas abiertas se debe comprender que se hace referencia a la forma de administrar el conocimiento creado por un grupo de científicos o investigadores acerca de un tópico en particular.

Una ventaja fundamental de los proyectos de arquitectura abierta es que paralelizan la ejecución de los mismos, acortando el ciclo de desarrollo de la tecnología y garantizando una retroalimentación de muchos profesionales lo que garantiza un nivel de calidad adecuado.

Existen promotores y detractores del paradigma de arquitectura abierta, pues plantean una discusión sobre la protección de la propiedad intelectual contra el nivel aplicabilidad y escalabilidad de una tecnología. Es importante, esclarecer que el desarrollo de arquitecturas abiertas se puede dar entre consorcios de compañías que buscan establecer una base de desarrollo común, es decir se construye un cuerpo de conocimiento que favorece a todos los involucrados y a partir de este se diseñan y manufacturan productos que van a satisfacer necesidades particulares de un segmento de mercado propio de cada compañía, con lo cual se minimizan costos de investigación y desarrollo, así como se reduce el tiempo necesario para alcanzar el mercado.

Para la comunidad de investigadores, las arquitecturas abiertas de redes inalámbricas de sensores plantean una oportunidad para desarrollar aplicaciones de recolección, análisis y síntesis de datos que van a facilitar el desarrollo de los modelos que describen sus objetos de estudio. Es por eso, que se puede afirmar que el desarrollo de un sistema de adquisición de datos ambientales con arquitectura abierta facilitaría las tareas de investigación, pues serían sistemas flexibles de programar, fáciles de instalar y de bajo costo, lo cual permite contar con suficientes dispositivos para conducir una investigación tanto en ambientes cerrados como en áreas extendidas como por ejemplo en cultivos agrícolas.

Capítulo II: Descripción del Proyecto

Aspectos generales

El proyecto de desarrollo de un sistema de adquisición de datos ambientales surge como una necesidad de poseer una plataforma de hardware y software que les permita a los investigadores del TEC la implementación de sistemas de muestreo configurable, confiable, fácil de utilizar y de bajo costo.

Para solventar esta necesidad, se planteó el diseño de una plataforma de red inalámbrica de sensores de arquitectura abierta, que contemple una lista de requerimientos y restricciones definidas a partir del estudio de las potenciales aplicaciones que podría tener la plataforma CRTecMote.

Algunos aspectos que se consideraron al desarrollar el proyecto en mención fueron los siguientes:

1. Uso de tecnología de bajo costo
2. Modularidad del diseño
3. Utilizar en la medida de lo posible herramientas de código abierto o libres para minimizar el costo de implementación.
4. Estrategias de centralización de la información recolectada para su posterior análisis.
5. Interoperabilidad de la tecnología desarrollada con otras tecnologías para garantizar una estandarización de la solución planteada.

La metodología que se utilizó para abordar el problema fue la estrategia de diseño de ingeniería [10], la cual aborda el problema de estudio y plantea una solución como si fuese un producto comercializable.

Para ello, el método de investigación se plantea como un ciclo de generación de conocimiento que se puede ilustrar por la figura 2, en el que se parte de una necesidad, se sintetiza un problema o un conjunto de problemas, se hace una revisión del estado del arte, se desarrolla una solución articulada que satisfaga el conjunto de requerimiento y supuestos pre-establecidos y finalmente se construye el prototipo, el cual se evalúa para identificar un conjunto de áreas de mejora o en

el peor de los casos se plantea un rediseño de la solución y se comienza con el ciclo de generación de conocimiento nuevamente.

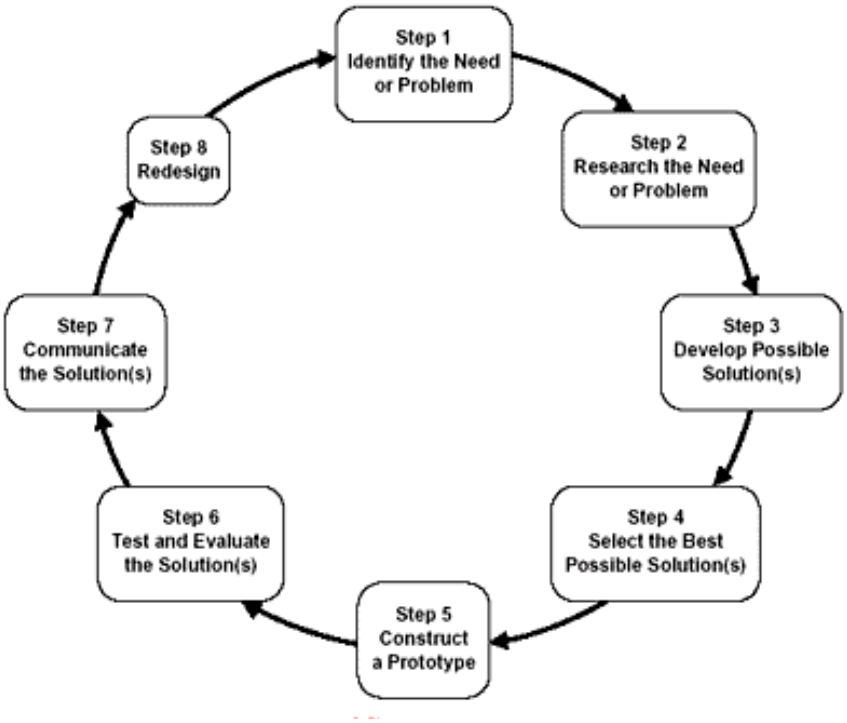


Figura 2. Ciclo de desarrollo de una investigación utilizando el método de diseño de ingeniería [11]

Para la consecución del proyecto se buscó la implementación de una red inalámbrica de sensores aplicada a la resolución de un problema de monitorización ambiental en particular, el cual fue la monitorización de jardines de clonación forestal. Esta implementación se realizó a nivel de prototipo, con lo cual se tendría la oportunidad de evaluar el diseño e identificar un conjunto de áreas de mejora que se abordarían en futuros proyectos de investigación que utilicen los resultados de la arquitectura abierta CRTecMote como insumo.

Con lo expuesto anteriormente, se probaría la hipótesis planteada al inicio del proyecto que afirmaba que era posible la implementación de una infraestructura de adquisición de datos ambientales por medio del diseño e implementación de una red inalámbrica de sensores de arquitectura abierta flexible, modular y de bajo costo para su aplicación en proyectos de investigación científica.

Antecedentes del proyecto

Como se mencionó anteriormente, desde el año 2004 la Escuela de Ingeniería en Electrónica ha venido incursionando en el desarrollo de tecnologías para mejorar las prestaciones de las redes inalámbricas de sensores. En el año 2007, se plantea el diseño de un nodo básico de adquisición de datos por parte del investigador, Ing. Adolfo Chaves Jimenez. Este nodo se fabricó a nivel de prototipo y se evaluó su aplicabilidad a la resolución de problemas de monitorización distribuida, en donde se pudo concluir que el nodo carecía de suficiente capacidad de procesamiento de información y carecía de la interfaz de comunicación inalámbrica que haría posible la sincronización y recolección de la información muestreada en campo. Es por ello se plantea este proyecto CRTecMote para solucionar las deficiencias encontradas al nodo original y convertirlo en una plataforma de recolección de datos más robusta.

A continuación se plantean los objetivos perseguidos por el proyecto CRTecMote.

Objetivos

Objetivo General

Desarrollar una plataforma de hardware y software de propósito general para obtención de datos de variables ambientales.

Objetivos específicos

1. Adaptar la red inalámbrica de sensores CRTECMote para garantizar su funcionamiento adecuado en condiciones ambientales exteriores.
2. Corregir los problemas detectados en los proyectos de investigación que utilizan la plataforma CRTECMote para la obtención de datos en una segunda versión del dispositivo.
3. Adaptar el sistema CRTECMote, para utilizarlo como base para los sistemas de obtención de datos de variables ambientales de otros proyectos de desarrollo en el ITCR
4. Poner a disposición de la comunidad científica internacional, el desarrollo realizado para permitir que los usuarios a nivel mundial contribuyan con su desarrollo dentro del contexto de arquitectura abierta.

Marco Teórico

El siguiente apartado presenta los fundamentos teóricos utilizados para el diseño y la implementación de la arquitectura de adquisición de datos CRTecMote

Redes inalámbricas de sensores (WSN)

Las redes inalámbricas de sensores (WSN por sus siglas en inglés), son un conjunto de dispositivos electrónicos interconectados los cuales tiene la capacidad de medir variables específicas del entorno donde se encuentran ubicados, comunicarse e interactuar por medio de enlaces inalámbricos. [12]

El transporte de la información en una red inalámbrica de sensores busca el envío de la información desde los distintos puntos de captura hacia un único punto de recolección de información al cual se le denomina “nodo sumidero”, el cual funciona como punto de contacto con otras redes de datos basadas en paquetes como lo es internet. [13]

Esta configuración con un único punto de recolección de datos o nodo sumidero, presenta la desventaja que cuando se requiere la implementación de muchos nodos (más de 1000), es poco eficiente por la complejidad de los protocolos de direccionamiento requeridos para alcanzar la salida a través de múltiples saltos. Esto implica que las redes inalámbricas de sensores de gran escala deben tener múltiples sumideros de información para ser escalables. [14]

La figura 3 muestra una representación de un escenario de nodo sumidero simple contra una representación de nodo sumidero múltiple.

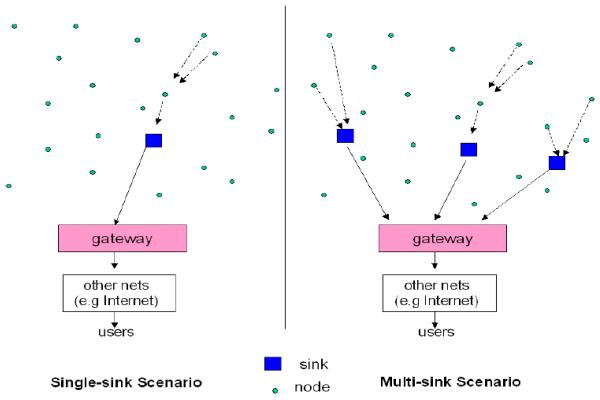


Figura 3. Escenarios de flujo de datos en una WSN [15]

Consideraciones generales para el diseño de WSN

Las redes inalámbricas pueden tener una gama de aplicación muy amplia por lo que una buena especificación de los requerimientos de diseño garantizará un desempeño adecuado de la WSN para el entorno en el cual se quiere operar.

En el diseño de las WSN uno de los requerimientos constantes es la restricción en la cantidad de energía que existe disponible para que el nodo opere, lo que plantea el reto de realizar un diseño lo más eficiente posible en términos de la energía requerida para las operaciones de monitorización, procesamiento y transmisión de la información recolectada. Normalmente se tienen los nodos alimentados con baterías, y se tienen enlaces de comunicación de bajo ancho de banda, además se debe considerar que la WSN debe ser tolerante a las fallas de uno o más nodos en el arreglo. Según sea el entorno de operación también se deben tener características de seguridad para evitar que la red pueda ser interferida o atacada por entes externos. Para lograr eficiencia energética se puede abordar la optimización desde varios enfoques. Se puede trabajar en mejoramiento de la eficiencia de la capa física, la capa de enlace de datos, los protocolos de enrutamiento o a nivel de la aplicación. En paralelo se puede mejorar la eficiencia energética mediante la selección adecuada de la tecnología apropiada para la implementación física de la red inalámbrica de sensores. [16]

Otras consideraciones en torno al diseño de redes inalámbricas de sensores es el costo de la manufactura de los nodos, pues se esperaría que fuese lo más baja posible para que la implementación de grandes WSN sea viable económicamente.

También se debe considerar el tipo de usuario de la información recolectada por la red inalámbrica de sensores en el sentido de que en algunas aplicaciones la información se recolecta para su posterior análisis, mientras que existe aplicaciones donde la información recolectada se usa para la toma inmediata de decisiones, lo cual implica diferentes abordajes al plantear la arquitectura de la red inalámbrica de sensores que se debe implementar.

Adicionalmente a los requerimientos y restricciones planteados por los usuarios de la red inalámbrica se debe analizar el entorno para identificar características

especiales que pudieran afectar la puesta en marcha del sistema de monitorización.

En cuanto al diseño de redes inalámbricas de sensores (WSN) se puede plantear una comparación contra los aspectos considerados en el diseño de redes inalámbricas tradicionales (WiFi) [17]

Tabla 1. Comparación de las características de las redes inalámbricas convencionales (WiFi) contra las Redes inalámbricas de sensores (WSN)

WiFi	**WSN**
Redes de propósito general	Redes de propósito específico
Enfocadas en rendimiento	Enfocadas en eficiencia energética
Diseñadas para encajar en estructuras pre-establecidas, es decir poseen un diseño estructurado	Diseñadas para ser funcionales en entornos no estructurados
Los dispositivos utilizados operan en ambientes controlados	Los dispositivos operan en ambientes abiertos y con condiciones inestables
El mantenimiento y la reparación se dan en condiciones de fácil acceso	El acceso físico a los nodos es difícil o inclusive imposible lo cual dificulta el mantenimiento o reparación
El costo de los dispositivos es elevado	El costo de los dispositivos es bajo
Utilizan plataformas computacionales convencionales	Se utilizan plataformas computacionales especializadas
Arquitectura de información centralizadas	Arquitectura de información descentralizada

Protocolos y topologías de comunicación para WSN

Para todos es conocido que el estándar de comunicaciones inalámbricas de dispositivos de comunicación es la familia del IEEE 802.11, donde se destacan distintas versiones. En la banda de los 2.4 GHz, el 802.11b y el 802.11g se han destacado, mientras que el 802.11a utiliza la banda de los 5 GHz. Al inicio del diseño de las redes inalámbricas de sensores se utilizó la banda de los 5 GHz, sin embargo con el tiempo, se migró al uso de la banda de los 2.4 GHz ya que es una banda mundialmente reservada para actividades industriales, científicas y médicas (ISM). Otro aspecto que motivo el cambio de banda y de protocolo de comunicación, fue que la familia de estándares 802.11, no estaban optimizadas para mantener un consumo bajo de energía lo cual se contrapone con uno de los elementos esenciales para el diseño de WSN [17].

Esta necesidad de obtener protocolos de comunicación que satisficieran los requerimientos de bajo consumo energético y baja tasa de transferencia, llevó al diseño del IEEE 802.15.4 que especifica los protocolos de comunicación a nivel de capa física y de enlace de datos para redes de comunicación de bajo consumo energético y baja tasa de transferencia de datos. Para 802.15.4 se han diseñado una serie de transceptores que soportan los protocolos de comunicación definidos por este estándar, mientras permiten la agregación de procesamiento de información y toma de decisiones.

Otro aspecto a considerar para la comunicación eficaz de la red inalámbrica de sensores es la topología de conexión de los nodos. Se pueden clasificar en dos grupos: aquellas llamadas topologías simples, como por ejemplo la conexión en topología punto a punto (P2P), o en topología estrella. Se pueden un segundo grupo de topologías denominadas complejas, las cuales pueden ser de árbol extendido (“Cluster Tree”) o de malla extendida (“Mesh”). Para diferenciarlas, las primeras solo necesitan realizar un salto de nodo para alcanzar el nodo sumidero, mientras que las segundas deben realizar dos o más saltos para alcanzar el nodo sumidero.

En la figura 4 y figura 5 se pueden ilustrar ambos grupos de topologías.

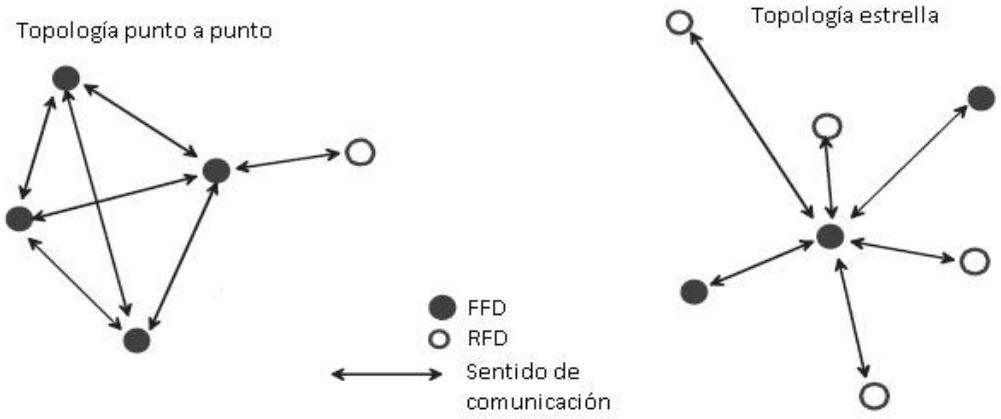


Figura 4. Topologías simples para la implementación de WSN [18]

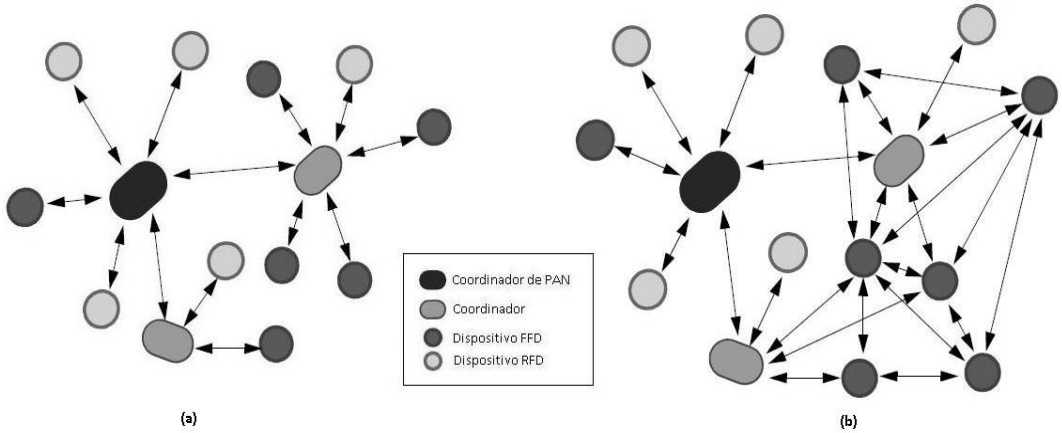


Figura 5. Topologías complejas para la implementación de WSN [18]

En la figura 4 los nodos de medición están denotados por nodos RFD (Reduced Function Device), mientras el nodo sumidero está denotado por los nodos FFD (Full Function Device).

En la figura 5, los nodos de medición son igualmente denotados por los nodos RFD, mientras el sumidero está denotado por el nodo coordinador PAN.

Las figuras anteriores diferencian las WSN con topologías “uni-salto” de “multi-salto” la cual va a determinar la pila de protocolos adecuada para cada aplicación en particular.

Arquitectura de los nodos utilizados en una WSN

Como se definió anteriormente, los nodos de una red inalámbrica definen el rendimiento de la red inalámbrica de sensores para procesar información.

Un nodo de una red inalámbrica de sensores consiste un conjunto de bloques que dentro de los cuales se puede definir la lógica de sensado, en la cual destaca un bloque de conversión de señales de analógico a digital. Cabe destacar que la resolución con la que se mida la señal va a estar determinada por la capacidad del convertidor de señales de analógico a digital (ADC). Otro bloque que se identifica dentro de la arquitectura de un nodo de una WSN es el módulo de comunicación, el cual se encarga de obtener información empaquetada por el procesador y enviarla a otro nodo, ya sea para que este la reenvíe o para que la sintetice dentro de la lógica de fusión de datos. Para conectar el bloque de comunicación con el el procesador del nodo se utilizan protocolos de conexión punto a punto que sean full dúplex, como por ejemplo el protocolo SPI. Otros transceptores de comunicación se conectan directamente a señales del procesador y se comunican mediante protocolos propietarios. El bloque de comunicación permite la ejecución de los protocolos de comunicación de datos y de enrutamiento de la información. Existe adicionalmente al bloque de sensado, comunicación y procesamiento un bloque de acondicionamiento de la energía para el nodo de manera que se le provea a los distintos bloques con las tensiones eléctricas y corrientes necesarias para un adecuado funcionamiento. Para lograr dicha funcionalidad se recurre a la implementación de convertidores de potencia DC-DC.

En cuanto al procesador utilizado por un nodo de una WSN, se tiene que seleccionar de acuerdo a las características del problema que se plantea resolver. En algunas ocasiones basta con tener un procesador de 8 bits para el muestreo y envío de información, mientras que en otras ocasiones es más conveniente el pre-procesamiento de la información, por lo que requiere tener procesadores más potentes, ya sea de 16 y hasta de 32 bits. [17]

La mayoría de los nodos utilizan sistemas en chip que poseen integrados los controladores de interfaz para conexión con los demás bloques del nodo.

Un ejemplo de una arquitectura implementada para el nodo de una WSN es el XYZ el cual presenta los cuatro subsistemas discutidos anteriormente: una unidad de procesamiento basada en el microcontrolador ARMTDMI, el cual puede operar a una velocidad máxima de 58 MHz, tanto en 16 como en 32 bits. Este microcontrolador posee controladores de acceso directo a memoria, interfaz de comunicación para protocolos SPI, I^2C, USART, SIO entre otras. El nodo también posee un módulo de comunicación basado en el transceptor CC2420, el cual es compatible con el estándar IEEE 802.15.4 y funciona en el espectro de los 2.4 GHz, y se conecta con el procesador por medio del protocolo de comunicación serie SPI. Tanto el procesador como el transceptor de radio tienen características de bajo consumo de energía (Wake up mode y Sleep Mode)

El bloque de control de la potencia entregada al nodo, se hace a través de convertidores de corriente directa y un sistema supervisor que garantiza el nivel de tensión eléctrica provista al nodo.

La figura 6 muestra la arquitectura completa del nodo de WSN denominado XYZ:

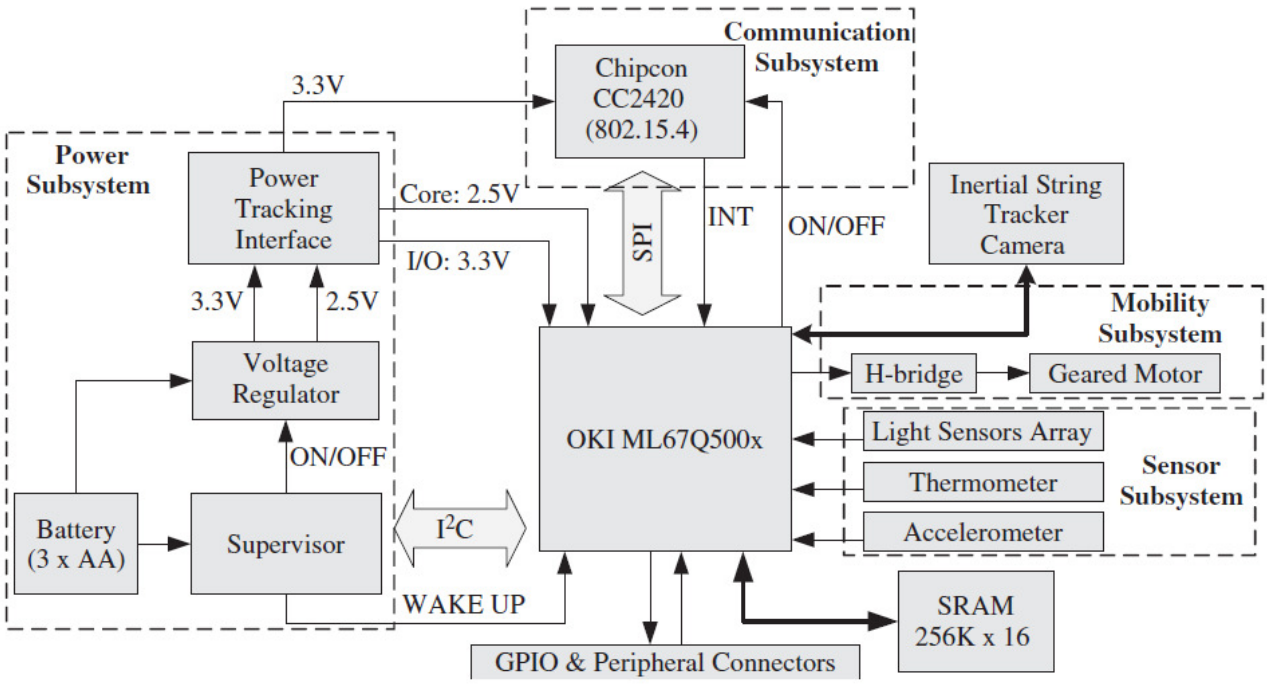


Figura 6. Arquitectura del nodo XYZ [17]

Sistemas operativos para nodos de una WSN

En la sección anterior se discutieron los aspectos relativos al diseño del hardware de un nodo para una WSN. En esta sección se discute el mecanismo de administración de recursos del nodo y los diferentes abordajes que se han realizado para obtener un sistema de administración de recursos por software. Uno de los requerimientos principales con respecto al software de un nodo es que este debe tener un tamaño reducido, dados los recursos reducidos con los que cuenta un nodo de una WSN [17]. Es por ello se busca un modelo de administración de recursos muy eficiente que provea los siguientes servicios:

- Administración de la memoria
- Administración del consumo de energía
- Administración del manejo de archivos
- Funciones de comunicación en red
- Controlar el acceso a los dispositivos de Entrada-Salida.
- Proveer un entorno de desarrollo de aplicaciones específicas.

En cuanto al abordaje del control de las distintas tareas que realiza un sistema operativo para un nodo de una WSN, se pueden tener sistemas operativos que realizan una única tarea a la vez (monotarea) o aquellos que pueden ejecutar varios procesos simultáneamente o multitarea. La selección de uno u otro mecanismo depende de la complejidad de los procesos que se ejecuten en el nodo, y de la cantidad de recursos con la que se cuente en el nodo.

Normalmente en redes inalámbricas de sensores, se utilizan con frecuencia sistemas operativos multitarea, dada la necesidad de interacción en tiempo real de los sub-sistemas de sensado, comunicación y procesamiento.

En cuanto a los aspectos funcionales que se deben tomar en cuenta a la hora de desarrollar o implementar un sistema operativo para un nodo de una WSN tenemos los siguientes:

Tipos de datos: En una WSN, la comunicación entre módulos es fundamental, por lo que se debe garantizar la consistencia del ancho de los datos para garantizar la mejor precisión posible. Con regularidad se limita el tipo de datos a aquellos que

sean soportados por el compilador que se utilice para codificar el sistema operativo.

Calendarización de procesos o tareas: Para lograr la atención adecuada de los procesos que involucra la operación de una red inalámbrica de sensores, se deben establecer políticas de calendarización y el diseño de estructuras que las soporten. Por ejemplo, se debe poseer capacidad de manejo de colas, cálculo de prioridades, entre otras. Otro aspecto que determina la estrategia de calendarización es la necesidad de respuesta en tiempo real, pues ello requerirá elementos adicionales que permitan medir si se alcanza el tiempo límite para la ejecución de una tarea ("deadline"). Algunos algoritmos comúnmente usados son el de cola (FIFO), el de round robin, el de prioridad fija, entre otros. También se debe considerar la posibilidad de estrategias que involucren la suplantación de procesos, por lo que se debe garantizar integridad de las tareas que se ejecutan mediante estructuras de control (PCB).

Tamaño de las pilas: en caso de utilizar un sistema operativo multitarea, se debe asignar a cada tarea una región de memoria para que sea utilizada como pila, en caso de tener muchas tareas los requerimientos de memoria se aumentarían lo que limitaría el acceso a los recursos del nodo.

Manejo de interrupciones: el manejo de interrupciones es requerido cuando se atienden dispositivos con baja frecuencia de petición de acceso a los recursos. La petición de los dispositivos es asincrónica y el nodo de la red inalámbrica debe poder identificar la fuente de interrupción para dar el tratamiento adecuado al dispositivo de entrada-salida. Una de las principales preocupaciones es en sistemas operativos multitarea donde el nodo debe suplantar la ejecución de las tareas para poder atender las peticiones generadas por los dispositivos que utilizan las interrupciones del sistema.

Programación orientada a procesos vs programación orientada a eventos: en una WSN, el soporte a tareas concurrentes es vital, por lo que se justifica el uso de sistemas operativos multitarea. En este caso se genera otro dilema sobre el paradigma de programación a utilizar, si el orientado a tareas o el orientado a

eventos. El paradigma de programación orientado a tareas, utiliza múltiples hilos dentro de un programa y un único espacio de direcciones de memoria. Lo anterior implica que cada tarea debe tener asignada una pila que le permita mantener almacenados los valores del entorno de ejecución de cada tarea con la finalidad de mantener consistencia en su ejecución. De esta forma tareas con dependencias de dispositivos de entrada o salida pueden ser bloqueadas para dar paso a otras cuyas dependencias están resueltas o no tienen dependencias de datos para ser ejecutadas. Para lograrlo el sistema operativo debe poder proteger las estructuras de datos compartidos, sincronizar la ejecución de procesos y mantener la consistencia del flujo del programa. Esto es costoso en términos energéticos pero permite realizar procesos más complejos en el nodo.

En cuanto al paradigma de programación orientada a eventos, se consideran dos elementos propios de este esquema de programación: los eventos y los manejadores de eventos. Los eventos actualizan el estado de la cola de calendarización del sistema operativo y esta decide cual evento atender de acuerdo a un esquema de prioridad previamente establecido. Una desventaja que se posa sobre este esquema de programación es que es poco eficiente para aplicaciones con requerimientos de tiempo real, dada la sobrecarga (“overhead”) de programación requerido para resolver la fuente de solicitud de eventos. Se recomienda este paradigma de programación para aplicaciones que no requieran ejecución de procesos en tiempo real.

Administración de la memoria: la memoria es un recurso preciado en WSN, dado que normalmente se utilizan microcontroladores para la implementación de subsistema de procesamiento, es un recurso sumamente limitado. Si a eso se le suma que la ejecución de un sistema operativo por parte del nodo tiene un costo en términos de recursos del sistema. En cuanto a la administración de memoria esta puede ser estática o dinámica y tiene lugar ya sea cuando el programa comienza, o cuando las tareas comienzan. Para entornos de WSN se hace necesaria la flexibilidad en el manejo de la memoria (dinámico), sin embargo esto acarrea costos adicionales en la administración de las estructuras, por lo que se vuelven poco atractiva para ambientes con alta restricción de recursos.

Otros aspectos: se discutieron en las secciones anteriores aspectos relacionados con la funcionalidad de los sistemas operativos utilizados en las redes inalámbricas de sensores, sin embargo adicionalmente a éstos, se deben otros aspectos que afectan la operación del software de la WSN. Se debe considerar aspectos como la separación entre nodos, la portabilidad de los programas y sistemas operativos a otras arquitecturas de nodo, la reprogramación dinámica de los nodos, etc. Los factores anteriormente citados en muchas ocasiones determinan el éxito en la implementación de una WSN, por lo que no pueden obviarse o descartarse.

Ejemplos de sistemas operativos para WSN:

- Tiny OS [19]: Es el sistema operativo más utilizado en la implementación de redes inalámbricas de sensores. Su arquitectura orientada a eventos lo hace muy atractivo para aplicaciones de monitorización. Conceptualmente el Tiny OS consiste de un calendarizador y un conjunto de componentes que se pueden adecuar a la aplicación [7]. Los componentes se clasifican en dos grupos, los componentes de configuración y los módulos. Los componentes de configuración permiten la conexión de varios módulos y varias configuraciones describen un programa de aplicación. Las aplicaciones son estáticas, es decir no pueden cambiar durante tiempo de ejecución. La figura 7 muestra la arquitectura del Tiny OS.

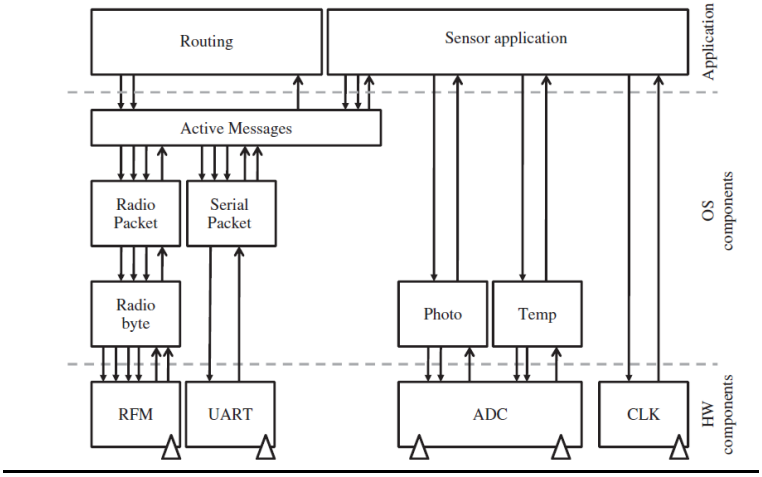


Figura 7. Arquitectura del sistema operativo Tiny OS [17]

- SOS [20]: al igual que TinyOS es un sistema operativo orientado a eventos, con la particularidad que permite la reconfiguración dinámica de las aplicaciones. Para ello es SOS posee un kernel y un conjunto de módulos que puede cargar o descargar según la necesidad de la aplicación. Esta característica permite establecer configuraciones dinámicas de módulos con lo cual se logra que el nodo sea reprogramable dinámicamente.
- Contiki [21]: Es un sistema operativo híbrido, pues su kernel funciona orientado a eventos, mientras que permite la implementación de múltiples tareas a través de bibliotecas asociadas. Esto le da características de programación reconfigurable. En cuanto al manejo de memoria, el kernel mantiene su bloque único, mientras que cada tarea posee su propio bloque de memoria que puede compartir entre procesos. Este sistema operativo introduce el concepto de proto-hilos ("protothreads"), los cuales son un híbrido entre un evento y un hilo.
- LiteOS[22]: es un sistema operativo orientado a tareas o hilos que permite la ejecución de múltiples aplicaciones. Está basado en la premisa que las aplicaciones se ejecutan en un espacio separado al kernel, por lo que se pueden distinguir las llamadas del sistema de las llamadas de los usuarios a los múltiples servicios. Una característica que distingue al Lite OS de los otros sistemas operativos presentados que permite la administración de un sistema de archivos distribuidos, de esta manera cada nodo puede acceder a los datos y directorios de la red inalámbrica de sensores. La figura 8 muestra la arquitectura de LiteOS [22]

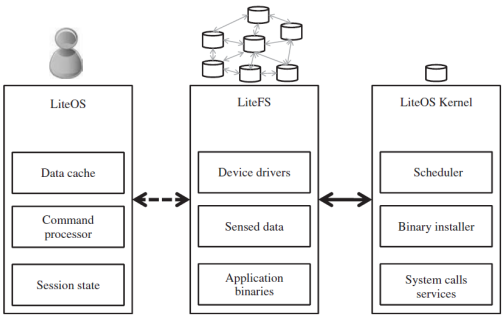


Figura 8. Arquitectura del sistema operativo Lite OS [17]

FreeRTOS

El FreeRTOS es un mini Kernel de tiempo real basado en prioridades, multitarea, de código abierto que puede ser utilizado y modificado bajo la licencia GPL. FreeRTOS puede ser utilizado en un sistema embebido para una aplicación específica, además dicho RTOS está escrito para ser portado a múltiples arquitecturas, entre las cuales están ARM, ColdFire, x86, PIC32MX, entre otros. La licencia presenta una excepción opcional que permite al usuario mantener su código utilizado en el FreeRTOS como un sistema cerrado, eso quiere decir que no puede ser modificado ni redistribuido por terceros sin los debidos permisos del creador. Otras características del FreeRTOS es que es simple y requiere una cantidad relativamente pequeña de memoria de código para ser almacenado. Dicho Kernel consta únicamente de alrededor de cinco archivos fuentes, escritos en lenguaje ANSI C, y algunos archivos de cabeceras (*.h), esto permite contar con un código portable relativamente fácil de mantener y modificar. Solo unas cuantas funciones son escritas en lenguaje ensamblador, además éstas pertenecen mayormente a rutinas que se manejan dentro del calendarizador. Dependiendo del compilador y de las opciones de compilación, el código objeto del Kernel del FreeRTOS puede ser reducido a un tamaño menor que 4 Kilobytes en el uso de memoria. El FreeRTOS es ejecutado de manera predeterminada a una frecuencia de operación de 80MHz para el reloj principal y la mitad del mismo para el reloj del bus sistema que utilizan los periféricos. Pero dicho valor de frecuencia puede ser modificado para permitir un consumo determinado de corriente bajo condiciones de operación nominales. Además, presenta un quantum o Tick del sistema igual a 1KHz, esto significa que el Kernel del RTOS creará 1000 secciones de ejecución para tareas en un intervalo de tiempo igual a 1 segundo. Si el RTOS se configura como apropiativo, las tareas se cederán los recursos del sistema cada 1ms de tiempo. Este parámetro también puede ser modificado según los requerimientos del sistema a utilizar.

Estructura interna del FreeRTOS

Entre los módulos básicos que presenta el FreeRTOS están el planificador de tareas (en este RTOS a los hilos de ejecución se les llaman tareas), el gestor de la

memoria principal, el manipulador de interrupciones, controladores de dispositivos de entrada / salida externos, manejo de recursos compartidos y el administrador de colas, las cuales tienen una gran variedad de usos.

Planificación de procesos en el FreeRTOS

Presenta un planificador de tareas que puede ser configurado como apropiativo o cooperativo. Un calendarizador es apropiativo cuando es capaz de interrumpir y suspender la ejecución de un proceso dado, en cualquier momento basado en reglas que garantizan el uso equitativo de los recursos del sistema. El FreeRTOS utiliza la regla llamada quantum, donde todos los procesos tienen un plazo de tiempo determinado para ser ejecutados por el procesador y utilizar los recursos que necesite, una vez que este plazo de tiempo ha terminado, el Kernel del sistema lo interrumpe para luego bloquearlo, cediendo los recursos del sistema a otro proceso. El calendarizador es ejecutado al final de cada intervalo de tiempo de ejecución de cualquier tarea. Existe una interrupción periódica (creada a partir del TIMER1 del microcontrolador) que se utiliza para tal fin, llamada “Tick Interrupt”. El tamaño del intervalo de tiempo o la frecuencia de esta interrupción está dado por la variable llamada configTICK_RATE_HZ, valor dado en Hertzios. El planificador de tareas de dicho Kernel también está basado en el uso de listas de prioridades, donde la menor prioridad corresponde a la tarea denominada “Idle Task”, cuyo valor numérico es 0. Cada tarea deber ser iniciada con una prioridad, y esta puede ser cambiada en tiempo de ejecución. Por otro lado el FreeRTOS presenta un valor máximo de prioridad, llamado “configMAX_PRIORITIES”. Es recomendable mantener este valor en el mínimo posible, ya que valores altos de este número puede ocasionar mayor consumo de memoria RAM por parte de los manejadores del Kernel.

El calendarizador siempre seleccionará a la tarea que presente la prioridad más alta y que esté lista para ser ejecutada, para llevarla al estado de Ejecución. Cuando hay varias tareas con el mismo nivel de prioridad listas para ser ejecutadas el calendarizador selecciona la primera de la lista de tareas listas y utiliza el modelo de planificación llamado Round Robin, por ello se puede decir

que dicho RTOS presenta una estructura de planificación hibrida entre Listas de prioridades y Round Robin.

Hilos de ejecución o Tareas en el FreeRTOS

Las tareas en el FreeRTOS son implementadas como funciones en el lenguaje C. Cada tarea es un pequeño programa el cual tiene un punto de entrada y se ejecutan infinitamente, esto debido a que se obliga a que presenten un bloque de ejecución infinita, por ejemplo el uso de los comandos for o while. El código que presenta la funcionalidad de la tarea debe ir dentro del bloque infinito. La tarea debe implementar un “break out” con el objetivo de salir del bloque infinito y poder eliminar la tarea creada antes de finalizar la función dada. Una aplicación puede consistir de varias tareas. Cuando el microcontrolador contiene un solo procesador entonces solo una tarea puede ser ejecutada a la vez en un intervalo de tiempo dado. Lo anterior implica que una tarea solo puede estar en uno de dos estados, ejecutándose o no ejecutándose. Este es un simple modelo pero el estado no ejecutándose contiene un número de sub-estados. Cuando una tarea está en el estado de ejecutándose el procesador ejecuta su código. Cuando está en el estado de no ejecutándose ella se encuentra “dormida” esperando a que el calendarizador la active de nuevo para ser ejecutada. Cuando el calendarizador activa una tarea dada, está ejecuta la instrucción que estaba después de la última en ser ejecutada. En el FreeRTOS el calendarizador es el único que puede cambiar de estado una tarea. Cuando no hay alguna tarea lista para ser ejecutada el calendarizador en el FreeRTOS selecciona la tarea “IdleTask”, la cual tiene una prioridad de 0, la prioridad más baja. Lo anterior con el objetivo de tener al procesador siempre ocupado. Esta tarea es automáticamente creada al iniciar el calendarizador, esto con el objetivo de tener siempre al menos una tarea por ejecutar y así lograr que el Kernel este en ejecución siempre. Al usar el modo apropiativo, si existe una tarea con nivel de prioridad mayor o igual que la “IdleTask”, el calendarizador automáticamente cede el CPU a esta tarea. Por otro lado, cada tarea se puede comunicar con otras más a través de la transmisión de mensajes o datos por medio de un conducto de información llamado cola en el FreeRTOS. Estas colas permiten utilizar mecanismos de comunicación y

sincronización entre tareas. Así una tarea puede colocar un nuevo elemento en la cola y en otro instante de tiempo una segunda tarea puede obtener el dato colocado y trabajar sobre ellos. Las colas también permiten el paso de punteros a memoria de datos para el manejo de grandes cantidades de datos.

Manejo de memoria en el FreeRTOS

Cuando de manejo de memoria se trata el Kernel puede asignar memoria dinámicamente a una tarea, una cola o un semáforo creado. Las funciones malloc y free de la librería estándar pueden ser usadas, pero se debe tomar en cuenta que se pueden sufrir algunos de los siguientes problemas, No siempre están disponibles en un sistema embebido pequeño, No son deterministas. La cantidad de tiempo que se necesita para la ejecución de estas funciones cambia de llamada a llamada, Pueden sufrir fragmentación de memoria, entre otros.

En el FreeRTOS cuando el Kernel requiere de memoria RAM, éste llama a la función llamada pvPortMalloc, en lugar de llamar a la función malloc. Cuando necesita liberar memoria RAM el Kernel llama a la función llamada vPortFree, en lugar de llamar a free. El sistema presenta tres maneras de manejar el reparto de memoria, para ello existen tres archivos de código fuente en .C llamados heap_1, 2 y 3. El heap_1 no libera memoria, utiliza un bloque de tamaño estático que utiliza para brindar memoria a quién la necesite, el tamaño de cada bloque que se solicita puede ser variable. El heap_2 también utiliza un bloque estático como espacio de memoria libre, pero si presenta la opción de liberar memoria, pero no presenta la función que permita combinar bloques de memoria adyacentes liberados. Y finalmente el heap_3 simplemente hace una llamada a las funciones estándar malloc y free para el manejo de memoria dinámica.

Manejo de interrupciones en el FreeRTOS

El FreeRTOS permite al usuario añadir controladores o drivers al sistema con el objeto de crear un sistema sensible a eventos externos, logrando además obtener información que debe ser manipulada por medio de los hilos de ejecución o tareas sobre dicho RTOS. Las interrupciones pueden ser configuradas con un nivel de prioridad dado, logrando de esta manera que siempre que ocurra un evento

externo pueda ser atendido en un lapso de tiempo determinista. Es recomendable que toda información que se obtiene a través de las interrupciones deba ser procesada por medio de tareas en el FreeRTOS, esto se puede lograr por medio de la decodificación de dicha información en la rutina de interrupción y enviar mensajes a través de colas de datos a otras tareas que procesen las peticiones generadas. Por otro lado, dado que el FreeRTOS permite la configuración de la prioridad de las interrupciones, se permite el anidamiento de una interrupción que presenta mayor prioridad dentro de otra, logrando de esta manera mantener el determinismo en la atención de eventos externos que lo ameriten.

Manejo de recursos en el FreeRTOS

Por ser el FreeRTOS un sistema multitarea existe el riesgo de que múltiples tareas generen conflictos a la hora del uso de un mismo recurso del sistema. Por ejemplo cuando un proceso empieza con el uso de un recurso dado pero no completa el acceso antes de ser bloqueado por el planificador al cumplirse el plazo de tiempo otorgado, quantum. Por lo que si dicha tarea deja al recurso en un estado inconsistente, entonces llegará otra tarea o rutina de interrupción que podría resultar con el manejo de datos corruptos o algún error particular. Para evitar problemas como los anteriores, el FreeRTOS hace uso de la llamada “Sección Crítica”, la cual garantiza el uso ilimitado de un recurso, hasta completar la operación sobre el mismo, suspendiendo el planificador temporalmente, logrando que ninguna otra tarea se apropie del mismo recurso por cumplirse el plazo de tiempo de la tarea que lo adquirió inicialmente. Esta sección crítica de código también tiene la opción de deshabilitar las interrupciones externas para lograr el mismo resultado antes planteado. Otra solución que presenta dicho RTOS es el uso de Mutex y semáforos binarios. Un Mutex es un tipo de semáforo que manipula el control de acceso a un recurso que es compartido por dos o más tareas. El uso de Mutex y semáforos binarios permite a una tarea que utiliza un recurso bloquear otras tareas que están a la espera de la liberación del mismo recurso. Un semáforo siempre debe estar libre cuando ninguna tarea está utilizando un recurso dado, y cualquier tarea puede tomarlo siempre y cuando éste se encuentre libre.

Sistemas de fusión de datos para WSN

En redes inalámbricas de sensores, es tan importante obtener información por medio de la infraestructura de medición, como la capacidad de gestión de dicha información. Por ello se habla de la necesidad de definir una estrategia de almacenamiento, y ordenamiento de los datos obtenidos por medio de la WSN.

A dicha estrategia se le conoce con el nombre de sistema de fusión de datos, y consiste en un sistema de interconexión con los nodos de una red inalámbrica de sensores para la recolección de la información, síntesis de la información y transporte hacia un punto de interés donde sea accesible por personas que tengan interés en el análisis y/o toma de decisiones con dicha información. La idea de fusionar los datos permite reducir la cantidad de información transmitida haciendo más eficiente la utilización de la energía por parte de los nodos de medición.

Para el diseño de un sistema de fusión de datos en una red inalámbrica de sensores, se plantean 3 preguntas claves que deben de resolverse durante la implementación del mismo [23]:

1. ¿En qué momento se solicita la toma de muestras o mediciones de los sensores?
2. ¿Qué estrategia sigue un nodo para realizar la fusión de datos, de manera que múltiples muestras puedan ser representadas por un reporte de información?
3. ¿Qué arquitectura de fusión de datos se debe utilizar?

En cuanto a la primera pregunta, se pueden identificar tres estrategias para definir el momento para la solicitud de muestreo a los sensores de la red:

a. Periódicamente el nodo de medición hace lecturas desde los sensores y las reporta al nodo sumidero.
b. El nodo sumidero envía un mensaje de solicitud de muestras al nodo de medición y este reporta la información medida por los sensores.
c. Se detecta un evento en el nodo de medición que determina la necesidad de reportar la información medida en los sensores al nodo sumidero.

Para responder la segunda pregunta, depende de la capacidad del nodo para realizar síntesis de información, y de la precisión con la que se quiera reportar la información. Se han utilizado desde el envío de resúmenes de datos (medias y desviación estándar), hasta modelos bayesianos basados en probabilidad. El modelo de fusión de datos debe estar dado como requerimiento de diseño de cada aplicación en particular.

En cuanto a la arquitectura del sistema de fusión de datos, se pueden identificar 3 modelos, los cuales están estrictamente ligados con la topología de conexión de los nodos de la WSN. Se discuten en seguida, algunas de las ventajas y desventajas de las tres principales arquitecturas reportadas [24]:

a. Fusión de datos centralizada
    - Ventajas:
        - Simple de implementar
        - Fácilmente se detectan reportes erróneos desde los nodos de medición.
        - No hay interdependencia entre nodos de fusión.
    - Desventajas:
        - Inflexible
        - La carga de trabajo se centraliza, lo que aumenta el riesgo de falla de WSN si falla el nodo de fusión de datos.
b. Fusión de datos descentralizada
    - Ventajas
        - Escalable
        - Tolerante a la falla de nodos de fusión, pues no se depende de un único nodo de fusión
    - Desventajas
        - Requiere mayor poder de procesamiento en los nodos de medición.
        - Algoritmos de implementación más complejos.
        - Mayor costo energético en los nodos de medición

c. Fusión de datos jerárquica
    - Ventajas
        - Balance de las cargas entre nodos de fusión de cada subred
        - Mejor balance de complejidad-flexibilidad de implementación
    - Desventajas
        - Si un nodo de fusión de una subred se desconecta, el conjunto de datos recolectados por ese conjunto de datos se pierde.

Otro problema que se tiene que resolver por medio de la estrategia de fusión de datos es maximizar la cantidad de tiempo que el nodo de medición se mantiene enviando información. Para medir este indicador se utiliza un parámetro llamado MLDA (Maximum Lifetime Data Aggregation), por sus siglas en inglés, que mide el tiempo que un grupo de sensores dura enviando información antes de que se reporten fallos de energía en alguno de los nodos de medición. Para ello se proponen diferentes estrategias que buscan una manera eficiente de recolectar y enviar la información desde los nodos de medición hacia el nodo sumidero.

Adicionalmente, se debe considerar aspectos de sincronía en la generación de datos desde los nodos de medición, hacia el nodo sumidero. Esto porque en la mayoría de estrategias fusión, los nodos deben recopilar un conjunto de datos desde los nodos de medición antes de aplicar algún algoritmo de fusión de datos. El problema principal de la desincronización de los nodos de medición es que afecta la integridad de los reportes de datos fusionados lo que causa desconfianza en las mediciones tomadas y que se generen inexactitudes sobre la ocurrencia de ciertos eventos cuando se analizan sobre una base de tiempo. Por eso la o duración desde el nodo más lejano de la red hacia el centro de fusión se utiliza como parámetro para determinar los periodos de muestreo y así garantizar que se evitan los problemas de sincronía dentro de la red inalámbrica de sensores. En cuanto almacenamiento de la información recolectada por una WSN, se ha registrado el uso de bases de datos, así como versiones simplificadas de esquemas de almacenamiento que faciliten la recolección de la información obtenida desde de los nodos de medición y fusionada a través los nodo sumidero.

Capítulo III: Metodología

La estrategia utilizada para la ejecución del proyecto se basó en la metodología de marco lógico, en la cual se identificó un árbol de problemas y un árbol de soluciones, con las que se procedió a definir un cronograma de actividades con duración máxima de un semestre, a partir de ello se pudo mapear un conjunto de entregables asociados a los productos definidos durante la etapa de anteproyecto.

Para la resolución de los problemas particulares se utilizó la metodología de diseño de ingeniería discutida en capítulo 2.

Así fue como para el primer semestre del 2009 se definieron y ejecutaron las siguientes tareas:

1. Se inicia con el estudio del estado del arte de redes inalámbricas de sensores mediante una investigación bibliográfica para fundamentar el marco teórico.
2. Se propone la lista de sensores ambientales que serían de utilidad en las aplicaciones de monitorización ambiental.
3. Se ubican potenciales clientes/socios interesados en la utilización de la plataforma mediante la identificación de proyectos de investigación o actividades que requieran la recolección y análisis de datos ambientales en el TEC.
4. Se realiza la documentación de requerimientos para dichas aplicaciones mediante reuniones de discusión de las necesidades de los potenciales clientes y las posibilidades tecnologías para el diseño del sistema de adquisición de datos ambientales.
5. Se completa el diseño teórico del nodo de la red inalámbrica CRTecMote, basado en las necesidades identificadas de los potenciales clientes.
6. Se realiza la compra de equipo y componentes destinados a la implementación del diseño propuesto.

Para el segundo semestre del 2009 se realizan las siguientes actividades:

1. Se diseña página web del proyecto en coordinación con el web institucional para dar a conocer a otros investigadores los alcances del proyecto CRTecMote.
2. Se implementa el nodo de sensado CRTECMote, mediante la integración de distintos componentes de hardware que componen los subsistemas del nodo.
3. Se realiza una investigación bibliográfica para la valoración de sistemas operativos de tiempo real de código abierto con la finalidad seleccionar uno para la implementación de la red inalámbrica CRTecMote.
4. Se adapta el sistema operativo de tiempo real y de código abierto FreeRTOS para su implementación sobre el nodo CRTecMote. Se consideran aspectos de estructura, tamaño del código y eficiencia energética como indicadores de éxito en la adaptación de dicho sistema operativo para el nodo CRTecMote.
5. Se desarrolla un sistema de almacenamiento de la información o sistema de fusión de datos con el propósito de organizar los datos recolectados de una manera lógica, que permitan a los investigadores diseñar sus sistemas de información para el análisis y toma de decisión adecuada. Dicho aplicación se denomina LOADPOINT y se implementa sobre el motor de bases de datos MySQL y la plataforma de programación .NET
6. Se valora la arquitectura de comunicación de datos que se debería utilizar de acuerdo del tamaño en la cantidad de nodos de las aplicaciones propuestas, mediante un estudio del esquema de muestro planteado por los clientes de la red CRTecMote.

Durante el primer semestre del 2010 se plantearon las siguientes actividades:

1. Acople del nodo CRTecMote para que pudiera operar y enviar datos a través de la red de telefonía móvil col propósito de obtener comunicación remota con el sistema LOADPOINT y drenar la información desde el punto de recolección hasta el punto de carga.
2. Implementación del software controlador (“drivers”) para los módulos de sensado y comunicaciones del nodo CRTecMote con el propósito de que

fuera capaz de formar una red inalámbrica de sensores en topología estrella y transmitir datos desde los distintos puntos de medición (nodos RFD) hacia los centrales (FFD) de la WSN donde accedían al LOADPOINT.

3. Realizar mejoras al sistema operativo de los nodos CRTecMote, con el propósito garantizar las características de tiempo real e incrementar la eficiencia energética de los nodos de la red.

Durante el segundo semestre del 2010 se realizaron las siguientes actividades:

1. Desarrollo de una aplicación de monitorización ambiental de las característica de un túnel de ambiente controlado para Genfores, con el propósito de caracterizar el desempeño de la humedad, temperatura e iluminación presentes para la consolidación de los datos para su utilización en el análisis y definición de modelos por parte del Dr. Olman Murillo Gamboa. Para la implementación se adapta la red CRTecMote para su operación en exteriores mediante el diseño de un empaquetado plástico que protege los dispositivos electrónicos de las condiciones de exceso de humedad.
2. Se identificaron otras aplicaciones para la red CRTecMote en la monitorización de procesos y sistemas. Para ello se plantearon nuevas iniciativas en el área de predicción de fallas en puentes, con ayuda de Virginia Tech.
3. Se realizó un modelo de simulación para estudiar la viabilidad del escalamiento de la red CRTecMote mediante grupos de subredes de recolección de datos interconectadas a través de internet por medio de la red de telefonía móvil (GPRS y EDGE). Eso se realizó por medio de simulaciones con la plataforma Omnet ++ de los escenarios de conexión posibles.
4. Se trabajó en la publicación de resultados del diseño de la red CRTecMote en el taller HEDISC donde se presentó un taller sobre las tecnologías desarrolladas y se obtuvo retroalimentación de investigadores de Costa Rica, Brasil, Francia, Panamá, Ecuador, Suiza y Bolivia.

Capítulo IV: Resultados y su análisis

En el capítulo anterior se especificó el método de abordaje de los distintos problemas en los que se dividió el diseño y la implementación de la red inalámbrica de sensores CRTecMote.

En seguida se presentan cada uno de los resultados obtenidos durante el desarrollo del proyecto de investigación.

Lista de requerimientos y restricciones para el diseño de CRTecMote

Como punto de partida para el diseño de la red inalámbrica CRTECMote, se realizó un análisis de las necesidades para las aplicaciones típicas de monitorización que se realizan en el TEC. En primera instancia determinó que el área de cobertura requerida una potencial red de sensores era 30 mil metros cuadrados en espacios abiertos, es decir sin obstáculos. Eso significaba definir un punto y garantizar la cobertura para el área encerrada 100 metros a la redonda. En la figura 9 se muestra el esquema de distribución requerido para la red inalámbrica de sensores de pequeña escala, que se propuso implementar.

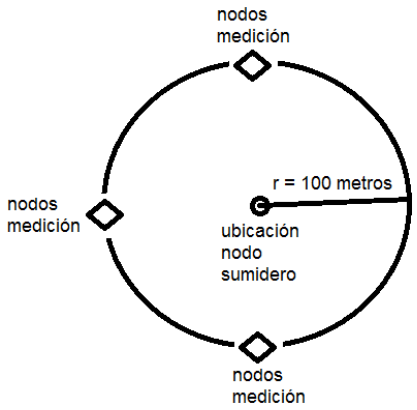


Figura 9. Esquema de instalación propuesto para la red CRTecMote.

Al calcular el área cubierta por el nodo sumidero, se obtiene un área potencial de cobertura definida por la formula:

$$A_{cobertura\ CRTecMote} = \pi r^2 \quad (1)$$

Donde r = 100 metros, con lo que se obtiene un área de cobertura de 31 415 metros cuadrados, siendo ésta mayor al área solicitada por los usuarios de la red inalámbrica de sensores.

El requerimiento anterior plantea la necesidad de una topología de red en estrella, donde los nodos sumideros funcionarían como punto central de la red. Para cumplir el requerimiento se seleccionó el estándar IEEE 802.15.4 como conjunto de protocolos para la conexión entre los nodos de la red CRTecMote, debido a su característica de bajo consumo de potencia eléctrica.

Para el diseño de los nodos CRTecMote se definieron los siguientes requerimientos:

- Capacidad de formar una red estrella de al menos 5 nodos con capacidad de conexión de hasta 100 nodos.
- Sistema operativo de código abierto y de tiempo real
- Rango de conexión de hasta 100 metros
- Optimizar el consumo de energía en los nodos de medición como de fusión.
- Capacidad de conexión a la red de telefonía móvil vía mensajes SMS.
- Capacidad de conexión de al menos 10 sensores por nodo de medición.
- Capacidad de almacenamiento de los datos por medio de base de datos.
- Capacidad de alimentación tanto por batería de 9V como por adaptador de corriente continúa.
- Procesador con capacidad de procesamiento de mediana-alta escala para aplicar algoritmos de pre-proceso avanzados.
- Diseño modular
- Capacidad de operación en ambientes abiertos de alta humedad.
- Capacidad de conexión de sensores comerciales de temperatura, humedad, luminosidad, PH, Turbidez y conductividad de agua.
- Capacidad de almacenamiento local en una unidad de almacenamiento flash.

Con la lista de requerimientos planteada anteriormente se procedió a realizar el diseño y la implementación de los nodos CRTecMote.

Diseño del nodo CRTecMote

Con los requerimientos definidos en la sección anterior se procede al diseño de los distintos subsistemas que conforman el nodo CRTecMote.

Selección del microcontrolador para el nodo CRTecMote

Para la implementación del módulo de procesamiento se eligió la utilización de un microcontrolador, dado que presenta la integración del procesador, la memoria y los dispositivos controladores requeridos para la implementación del nodo una red inalámbrica de sensores.

Sin embargo, existen gran variedad de microcontroladores (MCU), por lo que se deben elegir criterios adicionales para la selección adecuada del nodo CRTecMote.

Los criterios elegidos para la selección del microcontrolador fueron [25]:

1. Consumo de energía del núcleo de procesamiento del MCU (mW/MHz): para determinar la eficiencia entre microcontroladores con características operativas similares, como voltaje de operación, frecuencia de operación o cantidad de instrucciones por ciclo de reloj, se hace una comparación de la relación potencia/frecuencia para calificar la eficiencia en el procesamiento.
2. Capacidad para la ejecución de algoritmos complejos, como por ejemplo operaciones básicas de procesamiento digital de señales (DSP)
3. Instrucciones para el manejo de modos de energía: se consideró si los microcontroladores candidatos poseían estructuras de hardware para el control del consumo de energía del microcontrolador.
4. Disponibilidad de herramientas de desarrollo y depuración: disponibilidad de compiladores y herramientas para la depuración del código que se fuese a implementar en el nodo de la red CRTecMote.
5. Disponibilidad de soporte técnico por parte del fabricante del dispositivo.

Al evaluar los criterios antes descritos en los tres mayores fabricantes de microcontroladores: Texas Instruments (ARM7TDMI), Atmel (AVR32) y Microchip

(PIC32) se obtuvo la siguiente tabla de evaluación, en donde se compara el desempeño de los microcontroladores.

Tabla 2. Comparación de los núcleos de los microcontroladores candidatos para la unidad de procesamiento del nodo CRTecMote

Núcleo del MCU	**Relación P/F (mW/MHz)**	**DSP**	**Instrucciones para ahorro de energía**	**Plataforma de desarrollo**
ARM CORTEX-M3	0.084 (**)	*	WFI, WFE(***)	RealView ®
AVR32	1.25 (*)	***	Se requiere biblioteca (*)	AVR ® 32 Studio
M4K PIC32	0.066 (***)	**	SLEEP, IDLE (**)	MPLAB IDE ®

Al evaluar los tres microcontroladores candidatos, se decidió utilizar el microcontrolador con núcleo MIPS M4K que fabrica la compañía microchip y comercializa con el nombre de familia PIC32MX.

Este microcontrolador posee las siguientes características operativas:

- Procesador RISC de 32 bits con núcleo de procesamiento MIPS M4K.
- Convertidor analógico a digital de 10 bits de resolución y 16 canales
- Procesador multietapa de 5 niveles.
- 32 registros de propósito general 32 bits.
- Controlador DMA
- Unidades rápidas de multiplicación/división
- Soporte para protocolos CAN, I^2C, SPI, USART,USB, Ethernet.
- Dos modos de ahorro de energía.
- Escalamiento dinámico de la frecuencia de operación.

Selección del transceptor para módulo de comunicación

La etapa de comunicación de los nodos de la red inalámbrica CRTecMote se diseñó mediante el análisis, la evaluación y la selección de hardware requerido la implementación de conjunto de protocolos IEEE 802.15.4.

Debido a que existen varios transceptores comerciales con diferentes atributos, se realizó un análisis comparativo de las especificaciones técnicas de algunos de ellos para identificar cuál de ellos cumple con los requisitos de una red personal para nodos CRTECMOTE. El estudio abarcó los siguientes parámetros: frecuencia de operación, potencia de transmisión, sensibilidad de recepción, tasas de transmisión, consumo de potencia, tipo de modulación, soporte del estándar IEEE 802.15.4, interfaces de conexión y costo. La tabla de la figura 3, muestra los resultados obtenidos en dicho análisis.

Tabla 3. Cuadro comparativo de transceptores de radiofrecuencia

Transceptor	**MRF24J40**	**MRF49XA**	**CC1000**	**TR1000**	**Chipcon CC2420**
Banda	2.4GHz	433, 868 y 915 MHz	300 a 1000 MHZ	916.5 MHz	2.4GHz
Potencia TX	0 dBm	7 dBm	-20 a 10 dBm	1.5 dBm	-25 a 0 dBm
Sensibilidad	-94 dBm	-110 dBm	-110 dBm	-106 dBm	-95 dBm
soporta IEEE 802.15.4	Si	No	No	No	Si
MiWi	Si	Si	No	No	-
Modulación	O-QPSK	FSK	FSK - OOK	OOk - ASK	O-QPSK
Valor RSSI	Si	Si	Si	No	O-QPSK
Consumo	Rx: 19mA Tx: 23 mA Sleep: 2 µA	Rx: 11 mA Tx: 15 mA Sleep: 0.3 µA	Rx: 7.4 mA Tx: 10.4 Sleep:	Rx: 3 mA Tx: 12 mA Sleep: 0.7 µA	Rx: 19.7 mA Tx: 17.4 mA Sleep:
Tasa de TX	250 Kbps	115.2 a 256 kbps	76.8 kBaud	115.2 kbps	250 kbps
Costo	18.95 $	40 $	-	13.57 $	45$/5u

De la tabla 5 anterior se identificó que el dispositivo de radiofrecuencia que cumple con los requisitos de los nodos para el diseño CRTecMote es el MRF49XA, producido por la empresa Microchip Technology Inc. Este presenta el menor

consumo de corriente en recepción y transmisión, la mayor velocidad de transmisión y sensibilidad. Sin embargo, el costo del dispositivo, sugirió la utilización del transceptor MRF24J40MA, en la implementación de nodo de sensado, pues cumplía con los requerimientos y valía menos de la mitad que el transceptor MRF49XA.

Para la interfaz de conexión del transceptor MRF24J40MA al módulo de procesamiento (MCU) se utilizó el protocolo SPI, por lo que se procedió a la creación de 3 funciones dedicadas al manejo apropiado del módulo de puerto SPI del microcontrolador PIC32MX. Se trata de las funciones de inicialización del módulo, transmisión y recepción de datos. Para estas funciones se tomó en cuenta las siguientes características que requiere el transceptor para establecer una interfaz adecuada:

- Se transferirían datos con un ancho de 8 bits entre el MCU y el transceptor MRF24J40MA.
- Máxima frecuencia de operación del módulo de SPI: 2 MHz
- Polaridad y fase de la señal de reloj: modo 0,0
- Utilizaría la característica de "interrupción en cambio" del microcontrolador (específicamente INT1) para indicar una recepción, final de transmisión, salir del modo dormido, entre otras.
- Se requería de una señal de RESET para reiniciar el transceptor y otra de WAKE para salir de modo "dormido"

El MRF24J40MA maneja unos registros de control y almacenamiento que pueden controlar direcciones largas de 10 bits o cortas de 6 bits. Estos registros son los encargados de la administración eficiente del transceptor. Por ejemplo, a través de uno de estos registros se determinó el origen de interrupciones habilitadas. El acceso a ellos se realizó a través de los servicios de la capa física del protocolo.

De esta manera, la selección del protocolo de comunicaciones a utilizar, se redujo a dos posibles: MiWi^{TM} y MiWi^{TM} P_2P. Ambos cumplen con el requisito establecido anteriormente de crear una red en topología estrella para los nodos CRTecMote, sin embargo la simplicidad y modularidad del MiWi^{TM} P_2P lo posicionó como el

candidato óptimo. El estudio realizado sobre ambos protocolos demostró que los dos poseen la misma implementación a nivel de la capa física, pero no así de la de control de acceso al medio (MAC). Al ser ambos complemento del estándar IEEE 802.15.4, ofrecen los servicios de algunas de las capas superiores aunque bajo enfoques de implementación diferentes. De la misma forma, los dos protocolos fueron desarrollados mediante la creación de una interfaz de aplicación de programa y su código abierto permite la modificación para adecuarse a la necesidad de CRTecMote.

Una diferencia muy importante entre estos 2 protocolos y que marcó el punto de selección de uno de ellos, fue el hecho de que MiWi^{TM} P_2P fue programado a nivel de capas, es decir, se puede diferenciar fácilmente los límites de una y el comienzo de la otra. Específicamente el protocolo provee la programación de las capas física, MAC y de aplicación. En la primera, se ofrecen los servicios de acceso y control del transceptor, a través del cual se puede obtener información sobre la gestión de este nivel de red. Luego, en la capa de acceso al medio, MiWi P_2P presenta atributos como control de la potencia, selección del canal, manejo de los modos de operación del transceptor, valoración del canal, direccionamiento y control de flujo de la trama de datos en transmisión y recepción. Por último, en la capa de aplicación, este protocolo permite al usuario el control de la información que circula por la red y la creación de la red misma.

En el protocolo MiWi^{TM}, aunque se proveen muchos de estos servicios, no se distingue su implementación dentro de las capas. Por esta programación modular y sencilla, sumada al bajo consumo de potencia se escogió MiWi^{TM} P_2P para la creación de una red en topología estrella de la red de sensores CRTecMote.

Diseño, implementación y validación del módulo de acople de sensores

Los sensores que se seleccionaron para la medición de variables ambientales con CRTecMote producían corrientes de salida en el rango de 4mA a 20mA como respuesta los cambios en las variables medidas y se debía acondicionar dicha señal de corriente a un rango de tensión eléctrica permitida por el convertidor

analógico a digital (ADC) del microcontrolador (PIC32), el cual estaba definido en el rango de 0V a 3.3V.

Habiendo dicho esto, se procedió a resolver dos problemas, el primero convertir la señal eléctrica de intensidad de corriente a tensión eléctrica y el segundo ajustar la señal de tensión eléctrica para no sobrepasar los rangos establecidos por el convertidor analógico a digital (ADC) del microcontrolador usado en CRTecMote.

En seguida se presentan los resultados obtenidos en cuanto al diseño de la etapa de acople de sensores ambientales para su funcionamiento con el nodo CRTecMote.

Existen diversos circuitos para convertir corriente eléctrica a tensión eléctrica [26], sin embargo para este diseño el primero que se evaluó fue el seguidor de corriente:

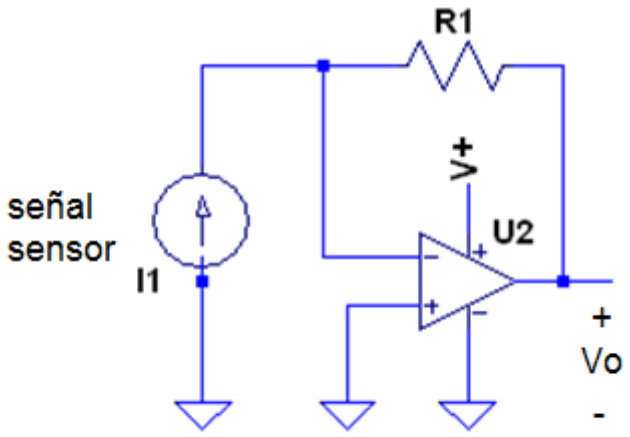


Figura 10. Circuito de conversión de corriente a tensión eléctrica

Donde la tensión eléctrica de salida (V_O) está dada por:

$$V_O = -I_1 * R_1 \quad \textbf{(2)}$$

Para este diseño V_O = -3.3V con una I_1 de 20mA, despejando de la ecuación 2 donde R1 =165Ω, sin embargo al no haber tener disponible resistencias comerciales con dicho valor escogió una resistencia con un valor de 120Ω y una

tolerancia del 5%, por lo que V_O = -2.4V, el cual no es conveniente según las especificaciones del diseño que se solicitó, por su polaridad negativa. Para corregir este efecto, planteo la implementación de un circuito seguidor de tensión que se conectara a una tensión de referencia generada a partir de la fuente de corriente, es decir se convertiría la corriente en una tensión, haciéndola pasar por una resistencia en paralelo a la fuente de corriente.

Se utilizó una resistencia en paralelo para generar una dicha tensión de referencia. Este método lo recomendó el proveedor de los sensores y definió una resistencia en paralelo R = 120Ω, por lo que el rango de voltaje que se puede generar es:

$$V = I * R \qquad \textbf{(3)}$$

Con los siguientes valores de tensión máxima y tensión mínima.

$$Vmin = 4mA * 120 = 0.48V$$

$$Vmax = 20mA * 120 = 2.4V$$

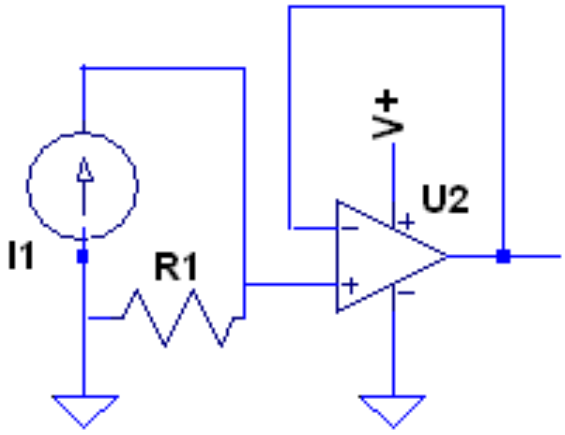


Figura 11. Circuito seguidor de tensión propuesto.

A la salida del seguidor de tensión se conecta un circuito amplificador el cual toma la tensión eléctrica obtenida por la resistencia y la acondiciona para cumplir con el rango de tensión de 0 V a 3.3 V especificado por el módulo de convertidor de analógico a digital (ADC) del microcontrolador de CRTecMote. Para el diseño del circuito amplificador se debe linealizar la curva de tensión eléctrica para obtener

3.3V como máximo y 0V como mínimo, a partir de las tensiones de salida producidas por el circuito seguidor de tensión antes descrito.

En la figura 12 se muestra la implementación final del circuito utilizado para ajustar las señales recibidas desde los sensores de variables ambientales cuya salida de corriente de 4 mA a 20 mA dicha señal se denominó “I sensor” y se generaba la salida con el nombre de V_{out}(ADC). , las cuales tenían un rango de 0V a 3.3V.

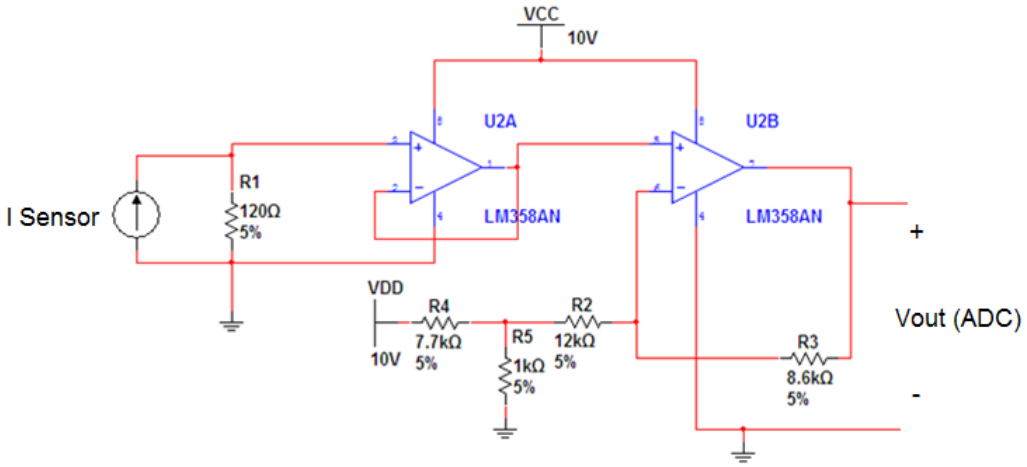


Figura 12. Circuito acondicionador de señal (CAS) propuesto para CRTecMote.

En la figura 13 se muestra el valor obtenido a la salida cuando se estimula el circuito a la entrada con el valor mínimo de corriente (4 mA). Para ello se utiliza el simulador.

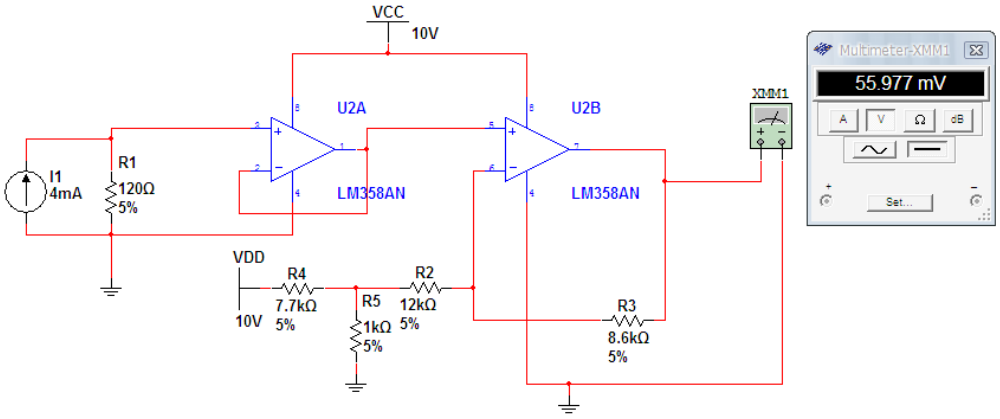


Figura 13. Simulación de valor mínimo obtenido con propuesta de CAS.

En la figura 14 se muestra el valor obtenido a la salida cuando se estimula el circuito a la entrada con el valor máximo de corriente (20 mA) que podrían generar los sensores.

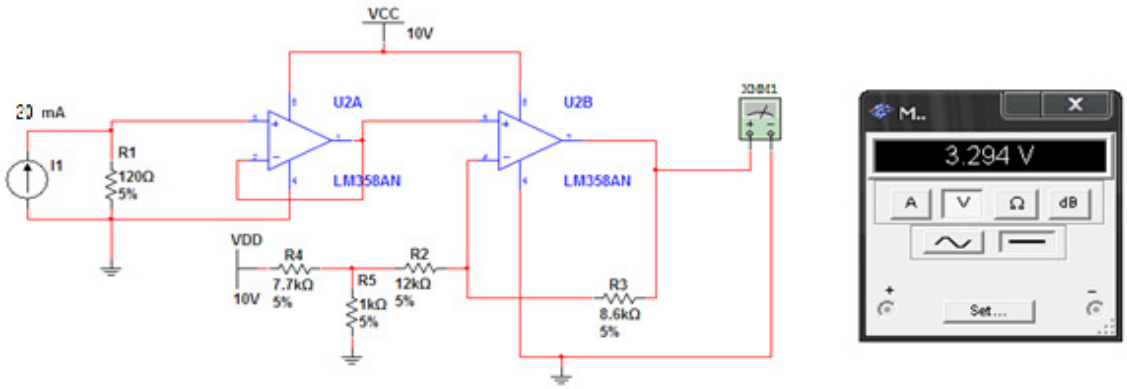


Figura 14. Simulación de valor máximo obtenido con propuesta de CAS.

Se obtuvo también un gráfico de respuesta de frecuencia del circuito de sensado de señales que se diseñó. La figura 15 muestra el resultado obtenido mediante simulación, donde se aprecia que la frecuencia de corte se ubica en los 350 kHz.

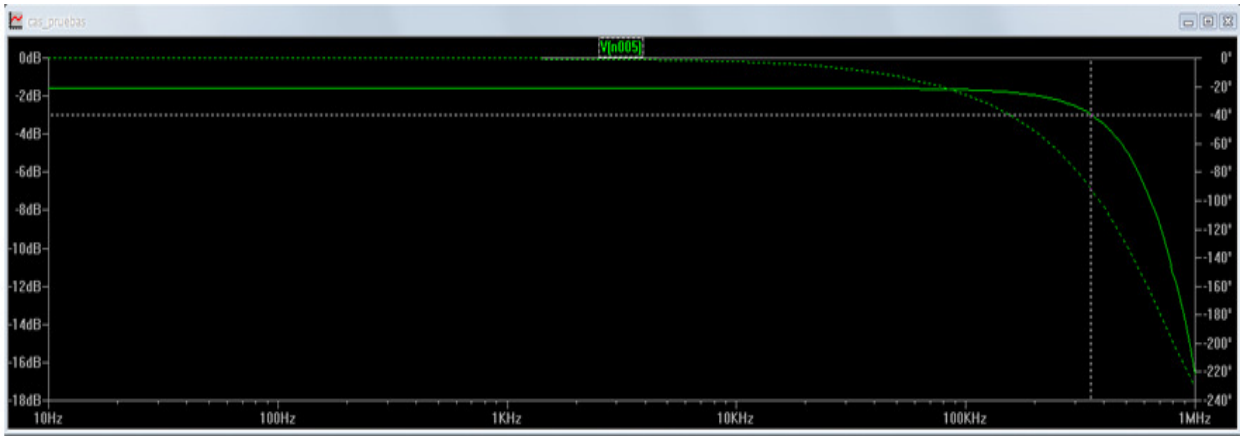


Figura 15. Simulación de respuesta de frecuencia obtenida con propuesta de CAS.

Una vez probado el circuito a nivel de simulación se procedió a su implementación física para los cual se diseño una placa de circuito impreso (PCB)

En las figuras 16 se muestra el diseño del PCB realizado utilizando el software de CAD/CAM EAGLE.

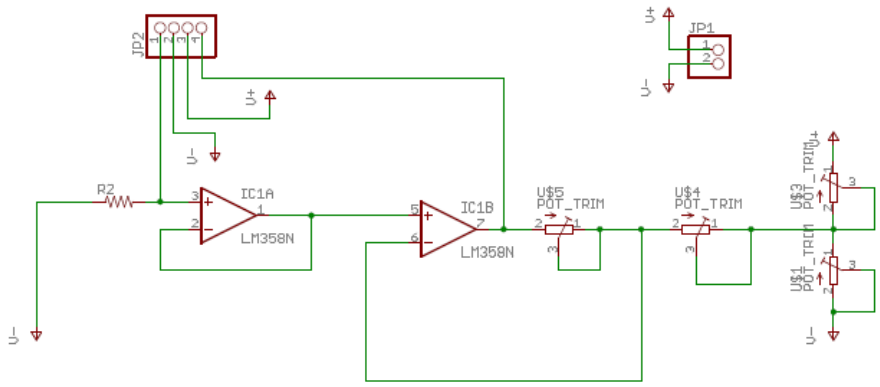


Figura 16. Diagrama esquemático de circuito acondicionador de señal implementado con EAGLE

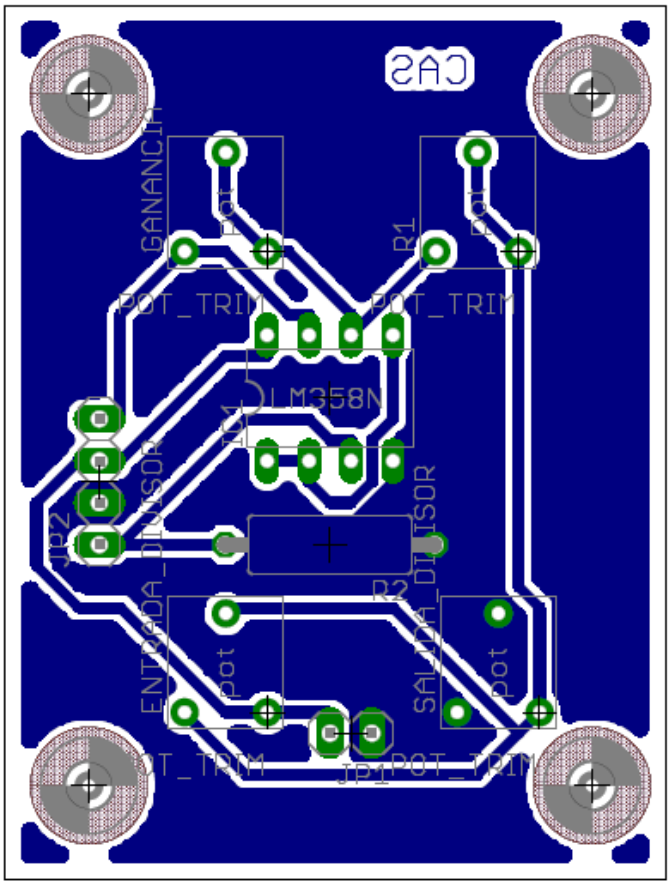


Figura 17. Diseño de la placa PCB obtenida a partir del esquemático con EAGLE.

A partir de la implementación del PCB de la figura 17, se realizó el ensamble y la validación de parámetros de salida de tensión eléctrica en función de los parámetros de entrada de corriente. La tabla 4 muestra los resultados obtenidos contra los valores simulados y sus respectivos porcentajes de error.

Tabla 4. Datos experimentales del CAS con el amplificador LM358N.

I sensor [mA]	V IN [V]	V_{OUT}(ADC) medido [V]	V REF [V]	V_{OUT} (ADC) simulado [V]	% Error
4,250	0,539	0,158	1,112	0,051	208,714
4,490	0,570	0,207	1,115	0,101	105,646
4,750	0,602	0,257	1,118	0,154	66,602
4,990	0,633	0,309	1,123	0,204	51,665
5,210	0,660	0,350	1,125	0,249	40,509
5,510	0,696	0,409	1,127	0,311	31,536
5,760	0,729	0,463	1,129	0,362	27,731
5,990	0,758	0,508	1,130	0,410	23,933
6,250	0,790	0,560	1,132	0,464	20,820
6,560	0,830	0,625	1,134	0,527	18,504
6,720	0,850	0,657	1,136	0,560	17,239
7,050	0,892	0,725	1,138	0,628	15,367
7,760	0,980	0,867	1,143	0,775	11,900
8,700	1,096	1,044	1,149	0,969	7,785
9,500	1,197	1,217	1,156	1,134	7,365
10,680	1,344	1,455	1,165	1,377	5,681
11,500	1,447	1,620	1,171	1,546	4,797
12,480	1,568	1,817	1,178	1,748	3,955
13,110	1,646	1,941	1,182	1,878	3,368
14,000	1,756	2,121	1,189	2,061	2,899
14,860	1,863	2,292	1,195	2,239	2,388
15,550	1,948	2,430	1,200	2,381	2,067
16,350	2,047	2,590	1,205	2,546	1,740
17,020	2,128	2,721	1,210	2,684	1,384
17,590	2,202	2,843	1,215	2,801	1,487
17,860	2,232	2,891	1,216	2,857	1,189
18,360	2,296	2,997	1,220	2,960	1,247
18,880	2,357	3,090	1,223	3,067	0,740
19,200	2,396	3,156	1,226	3,133	0,725
19,470	2,428	3,205	1,227	3,189	0,504
19,970	2,485	3,295	1,231	3,292	0,091

Diseño del módulo de conversión de energía

Al principio de este capítulo se presentaron los requerimientos necesarios para la implementación del nodo de red inalámbrica de sensores CRTecMote. Uno de las necesidades que se planteó fue la capacidad del nodo para operar tanto con alimentación externa a través de un adaptador de corriente alterna que tomara como entrada 110V a 60Hz y generara una tensión de corriente directa de 9V. También pidió que el nodo pudiese ser alimentado como por medio de una pila. Para lograr lo anteriormente descrito, se procedió a diseñar un circuito eléctrico que pudiese ser capaz de cumplir con dichos requerimientos. El diseño planteado resultó en un circuito de dos etapas: la primera etapa toma, se implementa un bloque de protección contra polarización invertida, y se implementa un convertidor de corriente directa con entrada de tensión eléctrica de 9V a 12V y salida fija con un valor de tensión eléctrica de 5V requerida para alimentar el bloque de acople de sensores. La figura 18 muestra el circuito eléctrico obtenido para esta etapa

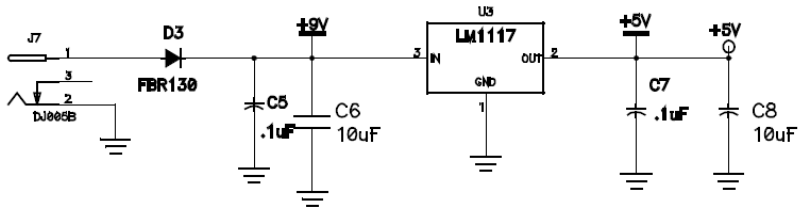


Figura 18. Circuito de regulación de tensión eléctrica de 9V a 5V

La segunda etapa consistió en un circuito de conversión de tensión eléctrica de 5V a 3.3V que era la tensión de operación requerida para la operación del microcontrolador utilizado para el nodo CRTecMote. La figura 19, muestra el circuito obtenido para esta etapa del acondicionador de energía.

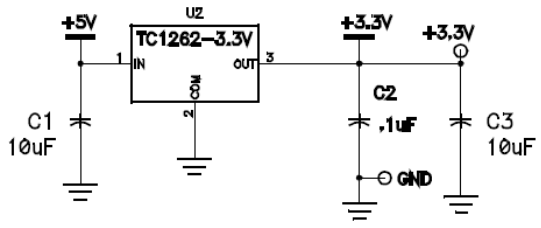


Figura 19. Circuito de regulación de tensión eléctrica de 5V a 3.3V

Implementación final del nodo CRTecMote

Luego de realizar el proceso de diseño del nodo CRTecMote y la selección de sus componentes, se procedió a la implementación del nodo. Para ello se diseñó un circuito electrónico para conectar los módulos de microcontrolador, comunicación, energía y un conjunto de lógica de soporte para la programación e interfaz con otros dispositivos. El módulo de acople de sensores se implementó en una tarjeta impresa aparte para mejorar aspectos de escalabilidad y reducir el tamaño de la implementación final del nodo.

La figura 20, muestra el circuito esquemático obtenido para la implementación del nodo CRTecMote.

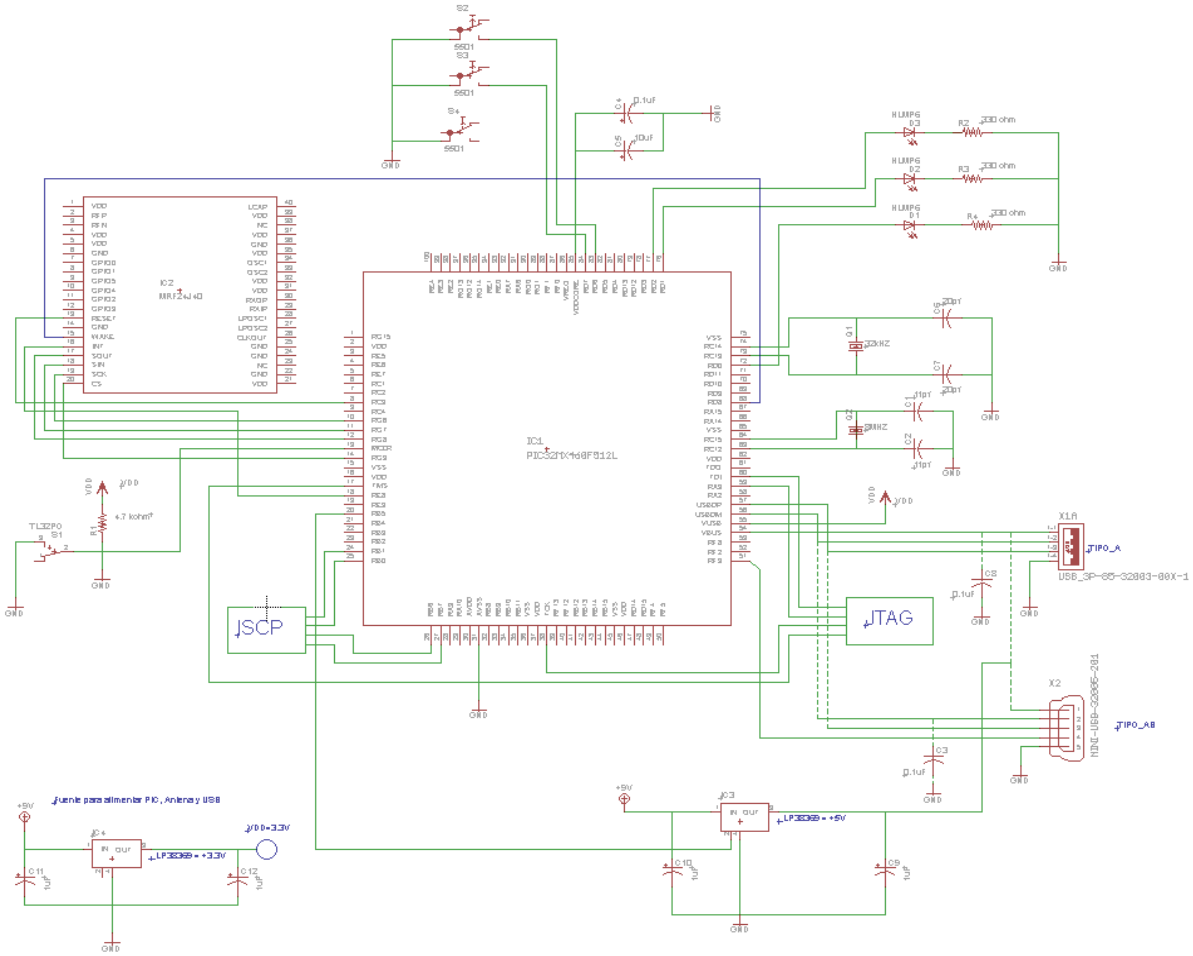


Figura 20. Diagrama esquemático del núcleo del nodo CRTecMote

A partir del circuito esquemático presentado en la figura 20 se procede a generar una placa de circuito impreso (PCB) para la integración funcional de los distintos componentes del nodo.

En la figura 21 se muestra la implementación del circuito impreso desde una vista superior, donde se puede apreciar la distribución de componentes y su conexión ("Layout").

Para generar el circuito impreso a partir del circuito esquemático se utilizó el software EAGLE.

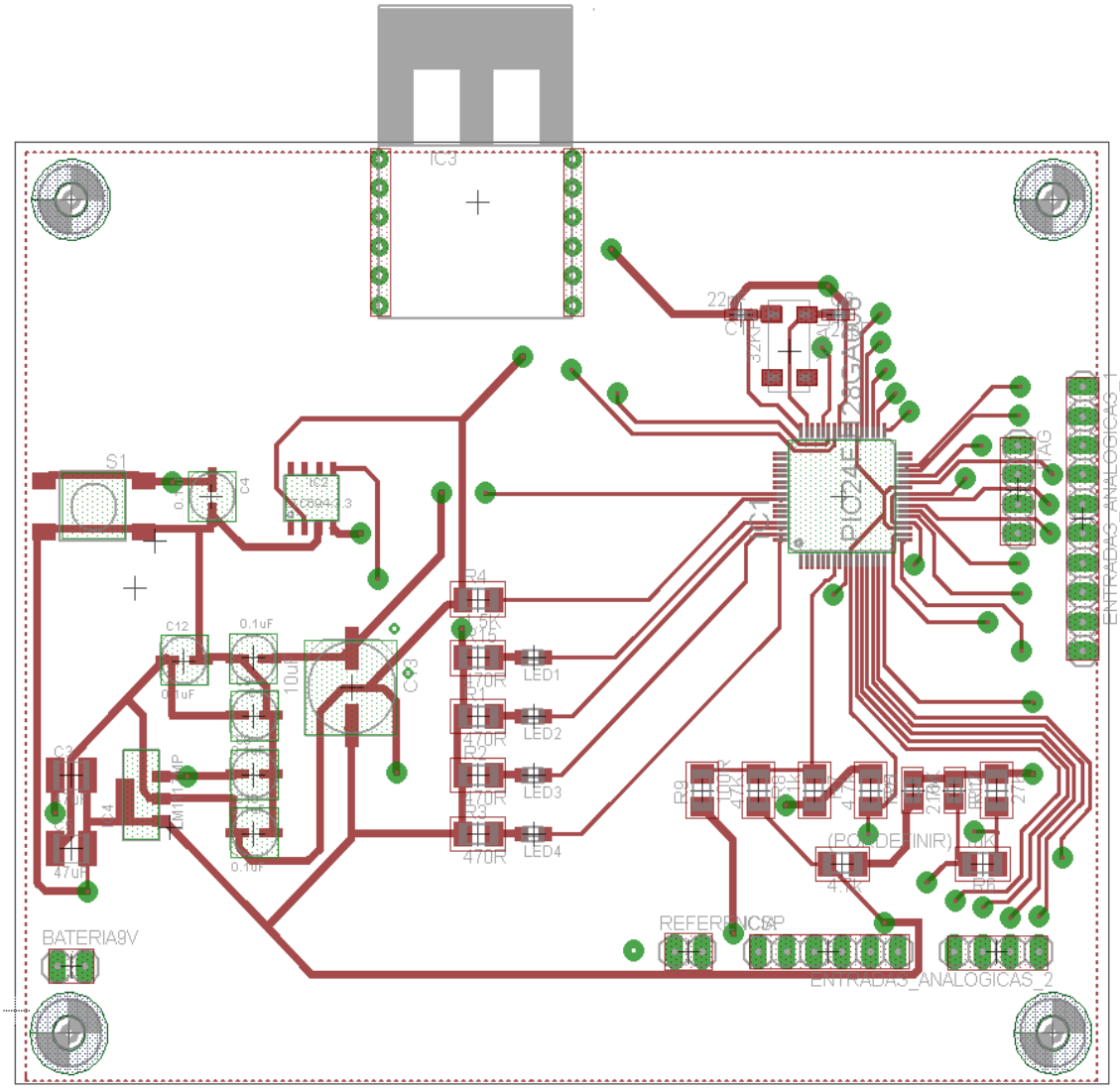


Figura 21. Placa de circuito impreso generado para la implementación de CRTecMote.

Escalamiento de los nodos CRTecMote

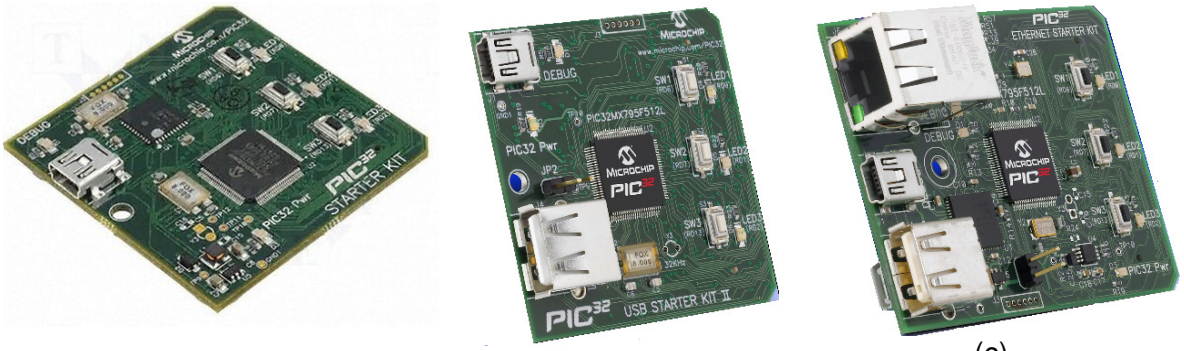
Una vez descrito el proceso de diseño e implementación del nodo se tenía que definir una estrategia de escalamiento de la tecnología con la cual se pudiera replicar las características del nodo para su utilización en una aplicación de campo que pudiese validar la operatividad de la red de sensores.

Para lograr mantener la consistencia modular del nodo, se procedió a identificar hardware que cumpliera con las características descritas durante el proceso de diseño, pero que pudiese interconectarse con facilidad para lograr la integración funcional del nodo.

A continuación se describe el proceso de selección de los componentes para llevar a cabo el proceso de escalamiento del nodo CRTecMote.

1. **Núcleo de procesamiento:** para el módulo de procesamiento de CRTecMote se escogieron tres módulos de hardware para llevar a cabo su escalamiento, según la función del nodo dentro de la red.
   - Nodo de medición: Para aquellos nodos de medición de la red CRTecMote se utilizará el módulo PIC32 starter Kit (P/N: DM320001), con un costo de 50 USD.
   - Nodos sumidero integrable a Red de telefonía Móvil: se requiere que el nodo de procesamiento provea una interfaz de conexión a un módulo “data card”. Para estos nodos se lección el hardware denominado Starter Kit II (P/N: DM320003-2) con un costo de 55 USD.
   - Nodos sumideros con capacidad de integración a redes de datos (TCP/IP): para ellos se escogió el módulo PIC32 Ethernet Starter Kit (P/N: DM320004) con un costo de 72 USD.

   La figura 22 muestra los distintos núcleos de procesamiento utilizados para la implementación de CRTecMote. En la figura 21.a se muestra el núcleo para nodos de medición, en la figura 21.b se muestra el núcleo para nodos sumideros con conexión a red de telefonía móvil y en la figura 21.c se muestra el nodo sumidero con capacidad de conexión a red de datos TCP/IP, por medio de Ethernet.



(a) (b) (c)

Figura 22. Núcleos de procesamiento utilizados para la implementación de CRTecMote.

2. **Módulos de comunicación:** para la implementación del módulo de comunicación con el transceptor seleccionado durante la etapa de diseño, se procedió a realizar investigar los distintos tipos de interfaces con los que se podía realizar la implementación. Se escogió para el escalamiento del CRTecMote el módulo de comunicación MRF24J40MA con un costo de 18 USD. La figura 23 muestra la vista superior del módulo antes descrito.

Figura 23. Módulo de comunicación utilizado para la implementación de CRTecMote

3. **Placa base de integración de módulos:** para la integración de los nodos CRTecMote se requería una placa que permitiera la conexión de los distintos módulos del nodo, así como el bloque de acondicionamiento de energía. Para ello se escogió la placa PIC32 I/O Expansion Board (P/N: 320002) con un costo 72 dólares. La figura 24 muestra una vista superior de dicha placa. A esta placa se puede conectar cualquiera de los módulos de procesamiento, así como el módulo de comunicaciones inalámbricas.

También posee interfaz para conexión desde la placa de sensores de 4-20 mA, así como otros sensores a través del protocolo I^2C, UART y SPI.

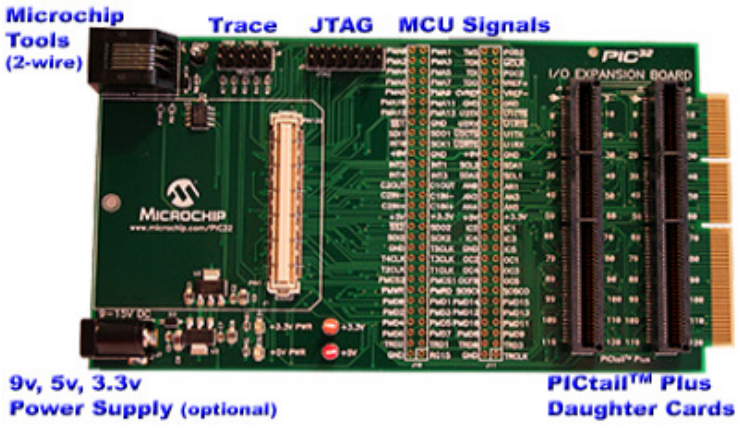


Figura 24. Placa base de integración de módulos para la implementación de CRTecMote

A partir del hardware anteriormente descrito se procedió al escalamiento del diseño de la red inalámbrica de sensores con nodos CRTecMote.

La figura 25 muestra la implementación de un nodo de medición (RFD) al cual se le conecta un conjunto de sensores por medio del protocolo I^2C. Se puede observar que el nodo fue adaptado para su operación en ambientes exteriores. Para ello se adaptó una caja de plástico con aislamiento de humedad, con lo que se logró proteger el nodo de condiciones de humedad excesiva y de la lluvia.

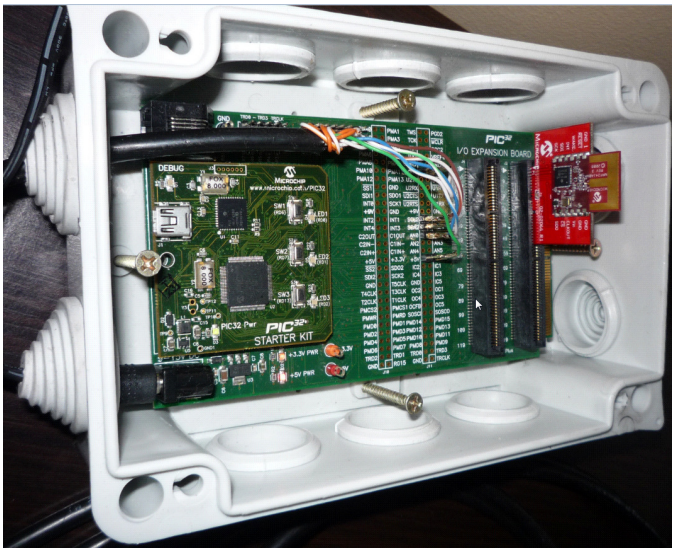


Figura 25. Nodo de medición CRTecMote

La figura 26 muestra la implementación final del nodo para su operación en ambientes abiertos.

Figura 26. Vista exterior de nodo de medición CRTecMote

En la figura 27 se muestra un nodo CRTecMote funcionando como sumidero de datos y con acceso a internet.

Para este nodo se tiene la capacidad de respaldar los datos por medio de una unidad de memoria flash conectada al puerto de USB del nodo. En este caso no se conectan sensores al nodo, sino que este funciona como punto de acceso para la red de sensores.

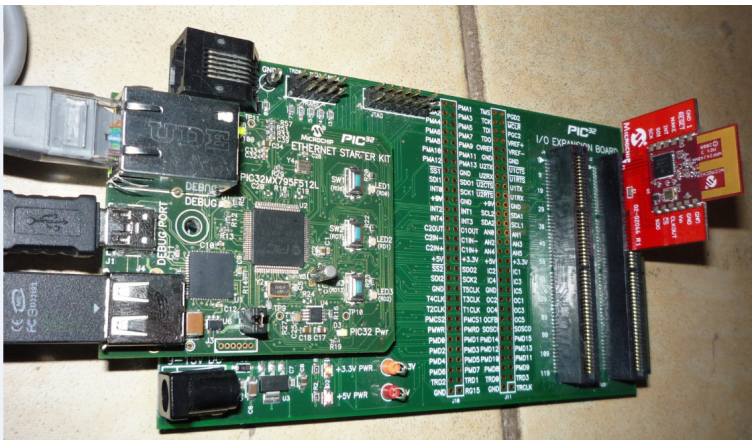


Figura 27. Implementación final del nodo sumidero para la red CRTecMote

Sistema operativo para los nodos CRTecMote

Para la selección del sistema operativo encargado de administrar los recursos del nodo CRTecMote, se tuvo que tomar como primera consideración que éste fuera portable a la arquitectura de microcontrolador PIC32MX, de manera que cualquier sistema operativo que no soportara esta arquitectura quedó descartado del proceso de implementación de CRTecMote.

Para la escogencia se realizó una investigación sobre las características de desempeño de los distintos sistemas operativos del mercado que podían portarse a la arquitectura PIC32MX.

La tabla 5 presenta un resumen de los parámetros evaluados para la selección de sistema operativo y su respectiva ponderación (entre más alta mejor).

Tabla 5. Benchmark de sistemas operativos para PIC32 [27]

Parámetro	TreadX	µC RTOS II	TNKernet	AVA	FreeRTOS
Código Abierto y Libre	No	No	Si	No	Sí
Calendarización Apropiativa	4,870,885	3,909,085	3,359,814	1,724,948	3,717,913
Procesamiento de interrupciones	6,918,050	5,259,998	5,497,238	5,207,762	1,881,892
Procesamiento de mensajes	6,928,383	No datos	4,146,914	2,761,154	484,691
Procesamiento de sincronización	15,337,354	10,293,318	7,353,579	7,514,799	1,989,999
Procesamiento de memoria	12,863,624	6,814,817	5,933,761	10,235,182	No datos

Al observar los datos de la tabla anterior, tanto el TNKernet como el FreeRTOS son de código abierto, el primero bajo la licencia FreeBSD-like y el segundo bajo el esquema de licencia GPL. Para el nodo CRTecMote, se decidió realizar una nueva distribución de sistema operativo, basada en el sistema FreeRTOS, ya que se contaba con suficiente información de soporte, manuales y comunidades trabajando con esta plataforma. A la distribución nueva se le denominó SIWA-RTOS.

Para el desarrollo de la distribución SIWA-RTOS como sistema operativo para los nodos CRTecMote se tenía como meta:

- Reducción del consumo de potencia en un 50% con respecto a FreeRTOS
- Restructuración del código fuente para reducir código redundante
- Implementación de nuevos controladores de software (“drivers”)
- Capacidad de respuesta en tiempo real

Restructuración del código fuente de FreeRTOS

El FreeRTOS presentaba inicialmente un conjunto de archivos de código fuente escritos en ANSI C y archivos de cabecera que permitían verificar el funcionamiento de los módulos internos del microcontrolador PIC32MX, así como la utilización de sus módulos periféricos. Además se tenía código que estaba destinado a la administración de las interrupciones con el fin de manipular eventos asincrónicos externos al nodo.

Como ejemplo de lo anterior se presentan los siguientes casos: existían tareas que generaban escenarios para verificar el uso exitoso en cuanto al manejo de colas de mensajes dentro del RTOS, para ello se creaban una serie de tareas donde unas de ellas enviaban mensajes por medio de las colas y otras salían del estado de bloqueo para realizar la lectura de dichos mensajes. Con ello se corroboró el uso exitoso de la API para crear tareas y planificarlas, para crear colas y gestionarlas, el uso del estado de bloqueo, así como la verificación del manejo de las colas entre tareas, una a una o una a varias tareas. Otro ejemplo fue la creación de ciertas tareas que permanecían bloqueadas a la espera de datos provenientes de una cola que era manipulada por una rutina de servicio de interrupción (ISR), por medio de la cual se comprobaba el uso adecuado de las colas de interrupciones, el manejo de interrupciones externas, el cambio de contexto, entre otros.

Se comprobó que si se tenían más de 20 tareas ejecutándose a la vez en el RTOS, se producirá un exceso de trabajo sobre el CPU, generando por consiguiente un consumo de potencia mucho mayor respecto a ejecutar pocas. Así lo primero que se efectuó fue la eliminación de todos los códigos fuentes .C y bibliotecas .H que contenían declaraciones de tareas que no representaban un servicio funcional y que además generaban un consumo significativo de potencia.

De aquí se obtuvo un RTOS modificado que contenía únicamente las funciones (APIs) necesarias para la administración de tareas y la gestión de los recursos de hardware propios del nodo CRTecMote.

Todo lo anterior representaba el mini Kernel del sistema, que se caracterizaba por poseer un planificador apropiativo que trabajaba con niveles de prioridad, donde las tareas se organizaban en listas independientes con cierto nivel de prioridad, por lo cual el calendarizador siempre gestionaba la lista con la prioridad mayor y que tuviese al menos una tarea lista para ser ejecutada, además cada lista se administraba por el algoritmo de Round Robin con periodo de cambios de contexto (quantum) que era configurable por el usuario durante la iniciación del módulo. El hecho de excluir ciertos archivos fuente .C y .H no significó que el SIWA-RTOS careciera de todas las funcionalidades que se poseían al principio en el Kernel del del FreeRTOS. Con la restructuración del código se obtuvo una reducción en el consumo de potencia de cerca del 10%.

El Kernel modificado quedó con los siguientes módulos:

- Planificador de tareas, a través de listas de prioridad y algoritmo Round Robin.
- Despachador de tareas, permitiendo el cambio de contexto entre tareas e interrupciones externas.
- Gestor dinámico de memoria, permitiendo asignar memoria así como eliminar la misma de ser necesario.
- Administrador de recursos, manejo dinámico de mútex y semáforos binarios para gestionar el acceso a recursos compartidos.
- Gestor de interrupciones, permitiendo la interrupción del sistema por eventos externos y ejecución de rutinas de servicio de interrupción (ISR).
- Comunicación entre procesos (IPC), permitiendo el paso de mensajes e información entre tareas a través de colas.

La figura 28 muestra la estructura obtenida para los distintos bloques del sistema operativo (SIWA-RTOS) implementado en el nodo CRTecMote. Esta versión obtenida a partir de FreeRTOS se le denominó SIWA-RTOS. La

palabra SIWA se tomó del una lengua indígena de Costa Rica y se traduce como brisa. La idea era tener un sistema operativo liviano como la brisa, para que fuese ejecutado por los nodos CRTecMote con la mayor eficiencia energética posible.

La implementación final consumió un espacio de memoria de código de 78KB y un espacio de memoria de datos de 40KB.

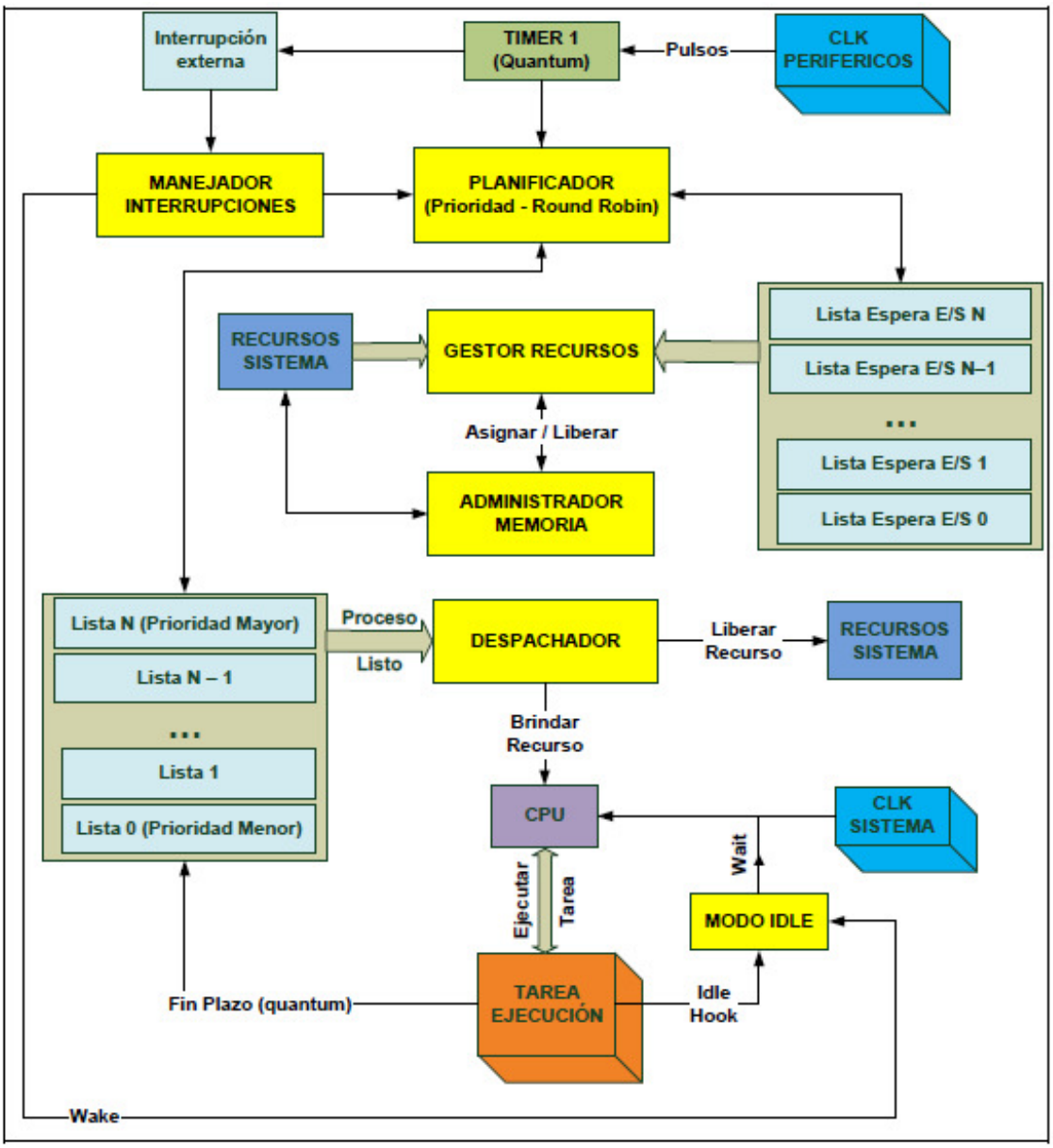


Figura 28. Arquitectura de la implementación final del sistema operativo para un nodo sumidero de la red CRTecMote

Implementación de los modos de bajo consumo energético para CRTecMote

Entre las razones que permitieron realizar esta modificación al Kernel del sistema, es que el PIC32MX permitía el uso de los modos de bajo consumo de energía, estos se dividen en dos categorías, la primera se puede utilizar mientras el CPU está activo, la cual consiste en la variación dinámica de la frecuencia de operación del sistema y la del bus para los periféricos. La segunda manera para reducir el consumo de potencia eléctrica fue por medio del uso instrucciones para detener el reloj interno del CPU. Para ello se podían usar dos métodos: el modo IDLE con diferentes fuentes para el reloj del núcleo de procesamiento (CPU) y los periféricos o el modo SLEEP, donde el reloj de todo el sistema es detenido, pero algunos periféricos pueden continuar trabajando siempre y cuando tengan un reloj independiente para ellos, este modo reduce al máximo el consumo de potencia, sin embargo el tiempo de recuperación si un periférico interrumpe el sistema es mayor lo que afectaba la característica de tiempo real. En SIWA-RTOS se utilizó el manejo del modo de bajo consumo de potencia a través del uso del modo IDLE debido a que permitía ahorrar energía cuando el nodo no estaba realizando ninguna operación de monitorización o comunicación y permitía un tiempo de recuperación mayor que el modo SLEEP anteriormente discutido.

Tampoco se utilizó la variación dinámica de la frecuencia de operación debido a que esto afecta directamente a todos los periféricos del sistema, lo que puede generar problemas de sincronización entre dispositivos externos como el transceptor y los periféricos del microcontrolador PIC32MX.

En cuanto al planificador de SIWA-RTOS, éste trabajaba por listas de prioridad, en conjunto con el algoritmo de Round Robin. Para evitar el problema de inanición de las tareas con más baja prioridad, se debía trabajar las tareas por eventos periódicos, donde cada una de ellas ejecutaba sus instrucciones por un intervalo de tiempo reducido para después entrar al estado de bloqueo en espera del siguiente período de ejecución. Con la manera de trabajar anteriormente descrita se presentaba el caso en donde el CPU ejecutaba la tarea “Idle” la mayor parte del tiempo. Tomando ventaja del caso anterior se configuró el Kernel para que

trabajara con el modo IDLE, así cuando se entrara a la ejecución de la tarea “Idle” el sistema realizara la instrucción en Lenguaje Ensamblador llamada WAIT, la cual permitía al CRTecMote entrar al modo IDLE. Para activar esta característica en el nodo, se debía colocar un cero lógico en el bit SLPEN dentro del registro de configuración del microcontrolador llamado OSCCON, logrando de esta manera detener el reloj que alimentaba al CPU del microcontrolador.

La modificación que se realizó sobre el SIWA-RTOS fue hacer uso de la tarea “Idle Task Hook”, por medio de la definición de configUSE_IDLE_HOOK a uno lógico en el archivo FreeRTOSConfig.h. Es posible añadir una funcionalidad especifica directamente dentro de la tarea “Idle” a través del uso de una función “hook” (o call-back), una función que es llamada por la tarea “Idle” por cada iteración del ciclo infinito dentro de dicha tarea. Dentro de los usos de las funciones hook están:

- Ejecutar tareas de baja prioridad que necesitan procesamiento continuo en segundo plano.
- Poner el procesador en modo de baja potencia, esto provee un método para ahorrar energía, cuando no haya alguna tarea por realizar.

Las funciones hook nunca deben entrar al estado de bloqueo o suspendidas. Además, se debe tomar en cuenta que si el RTOS hace uso de la API llamada vTaskDelete, entonces las funciones hook debe siempre retornar dentro de un período de tiempo razonable.

Desempeño de consumo energético del núcleo de SIWA-RTOS

Como se comentó anteriormente, a partir de los cambios introducidos a FreeRTOS, se generó una nueva distribución que fue utilizada en la implementación de la red inalámbrica de sensores CRTecMote.

Para caracterizar el consumo energético se utilizó un conjunto de tareas comunes a FreeRTOS y SIWA-RTOS con el fin de obtener una comparación entre ambas implementaciones del sistema operativo sobre el nodo CRTecMote.

A partir de las mediciones de consumo se obtuvieron los siguientes gráficos de desempeño.

Se parte de que tanto el nodo de CRTecMote ejecutando FreeRTOS con el nodo ejecutando SIWA-RTOS están alimentados con una tensión de entrada al microcontrolador de 3.3V y una frecuencia de reloj de 80 MHz. Ambos nodos ejecutan el mismo conjunto de tareas y la frecuencia intercambio de tareas es la misma para ambos casos con un valor igual a 1 kHz.

El gráfico de la figura 29, representa el consumo de corriente para un nodo CRTecMote ejecutando el conjunto de tareas de referencia por medio de FreeRTOS. Se puede observar que el consumo de corriente promedio está cercano a los 130 mA.

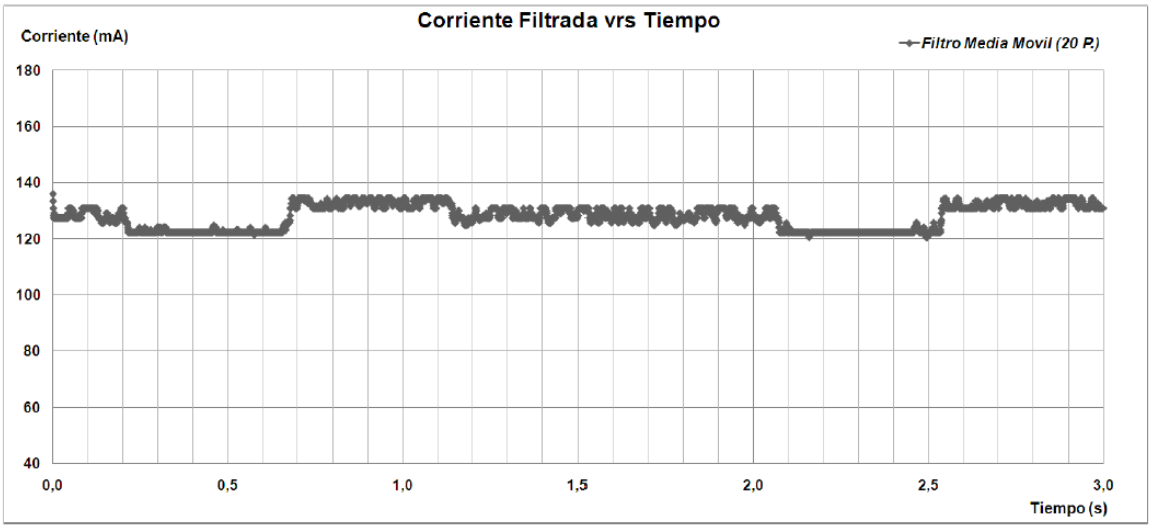


Figura 29. Consumo de corriente promedio de un nodo CRTecMote ejecutando un conjunto de tareas con el sistema operativo FreeRTOS.

El gráfico de la figura 30 presenta el consumo de corriente para un nodo CRTecMote ejecutando el conjunto de tareas de referencia, pero en esta ocasión utilizando el sistema operativo de tiempo real modificado SIWA-RTOS.

Se puede observar como el consumo de corriente electrica promedio para el nodo se encuentra cerca de los 75 mA contra los 130 mA registrados para el mismo conjunto de tareas ejecutado en el mismo hardware, solo que con el sistema operativo FreeRTOS.

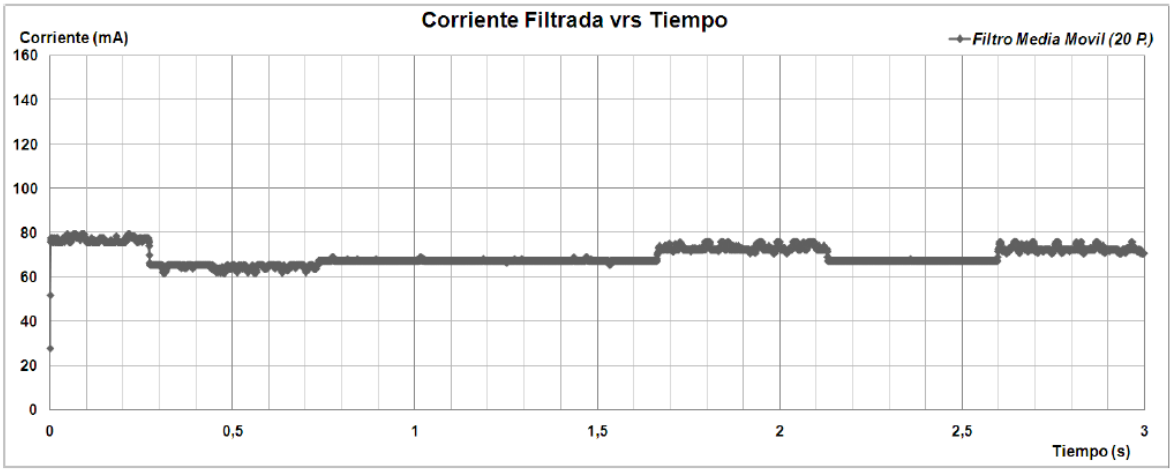


Figura 30. Consumo de corriente promedio de un nodo CRTecMote ejecutando un conjunto de tareas con el sistema operativo SIWA-RTOS.

Adicionalmente las tablas 6 y 7 muestran la información de consumo de potencia eléctrica consumida por los nodos cuando se realizó un barrido de frecuencias de operación del CRTecMote tanto para su operación con FreeRTOS como con SIWA-RTOS respectivamente para su comparación posterior en el análisis.

Tabla 6. Consumo de potencia eléctrica al variar la frecuencia de operación para el CRTecMote ejecutando FreeRTOS

Frecuencia Operación (MHz)	Potencia de Consumo (mW)
10	179.469
20	216.762
30	260.969
40	283.420
48	312.282
60	360.663
72	405.053
80	427.518
96	450.527

Tabla 7. Consumo de potencia eléctrica al variar la frecuencia de operación para el CRTecMote ejecutando SIWA

Frecuencia Operación (MHz)	Potencia de Consumo (mW)
30	180.551
40	185.327
48	196.935
60	200.723
72	216.788
80	224.150
96	241.327

Controladores de software desarrollados para CRTecMote

En este capítulo se ha venido discutiendo el proceso de implementación de la red inalámbrica de sensores CRTecMote y los resultados obtenidos. Hasta acá, se ha descrito la implementación del hardware, así como el sistema operativo utilizado para la administración de los recursos del nodo, sin embargo para el desarrollo de una aplicación funcional se requirió el desarrollo de un conjunto de controladores de software o "drivers" para poder acceder a servicios como el de conexión de red de los nodos, conexión a la red de datos TCP/IP, conexión a la red de telefonía móvil entre otros.

En esta sección se documenta la implementación de los servicios más relevantes para la operación de los nodos CRTecMote en la adquisición y procesamiento de datos ambientales.

Controlador de administración de red de área local

Considerando las especificaciones físicas y de programación, establecidas en la etapa de diseño del nodo, así como el transceptor y el protocolo escogidos, se procedió a la implementación de un controlador de software para los las comunicaciones de área local sobre un nodo CRTecMote ejecutando SIWA-RTOS. El proceso consistió, en primer lugar, en la inclusión de los archivos fuente (extensión "*.c") y de cabecera (extensión ".h") al código que conforma el SIWA-RTOS. En seguida se realizaron los cambios necesarios para generar la compatibilidad entre el protocolo de comunicaciones MiWi™ y el sistema operativo SIWA-RTOS.

Como se mencionó en una sección anterior, el transceptor se controla a través del acceso a sus registros. Para ello, fue necesaria la implementación de 4 funciones especiales para poder leer y escribir registros cortos y largos, eso sí, teniendo en cuenta que se transfieren datos de 8 bits por medio del protocolo SPI desde el microcontrolador hasta el transceptor de comunicaciones. Dichas funciones debían seguir la secuencia del diagrama de flujo documentado en la figura 31.

Con el fin de identificar y delimitar la capa física de las demás, al nombre de cada una de estas 4 funciones, se les agregó el prefijo "PHY" dentro del archivo de

cabecera llamado MRF24J40.h, en éste también se definió la dirección de todos los registros largos y cortos del transceptor.

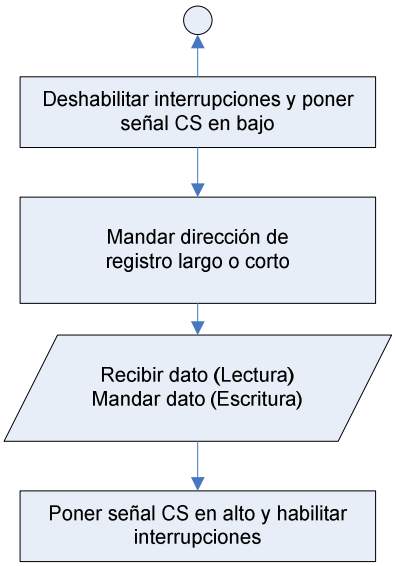


Figura 31. Diagrama de flujo para el acceso a los registros largos o cortos del MRF24J40MA

Para efectos de prueba de la capa física, se realizó el proceso de inicialización del dispositivo, la cual consistió en la especificación de parámetros de operación del transceptor, como por ejemplo la potencia de salida, el canal de operación, dirección de identificación del dispositivo, habilitación de interrupciones deseadas, habilitar el valor RSSI, especificación del tipo de nodo sobre el cual se trabaja, es decir si era un nodo de medición (RFD) o si se trataba de un nodo sumidero (FFD)

El siguiente paso fue la implementación de la capa de control de acceso al medio (MAC), con el propósito para que fuese accesible y controlable desde SIWA-RTOS. El conjunto de funciones de programación (API) que conforma esta capa proveía los siguientes servicios implementados mediante las operaciones de:

- **Transmisión de tramas**: en esta función se determinaba la forma de transmisión de la información. Por ejemplo, se establecía la estructura y el tamaño de la trama de datos a ser transmitida. Dependiendo de si se trata un comando, un dato o una señal de reconocimiento, se agregan los respectivos bits de configuración al encabezado de cada trama de datos.

También se establecía el tipo de direccionamiento a utilizar, considerando el destino de la información que se desea transmitir. Además, se incluía información sobre el dispositivo que transmitía los datos.

- **Inicialización de la capa MAC**: esta función inicializa todos los registros y variables pertenecientes a la capa de acceso al medio (MAC). Asimismo realiza la inicialización del transceptor.
- **Recepción de tramas**: esta función verificaba si se había recibido una trama de datos en el transceptor. Si este era el caso, se almacenaba toda la trama en una estructura de datos global para que fuese manipulada por las capas superiores del protocolo de comunicación MiWi™ P_2P.
- **Valoración del canal de operación**: Esta función determinaba el nivel de interferencia electromagnética presente en el canal sobre el que operaba la red mediante el cálculo de valor RSSI del canal.
- **Direccionamiento alterno:** los dispositivos en una red compatible con el estándar IEEE 802.15.4, requieren una dirección de identificación única de 8 bytes y una alterna de 4 bytes. Esta función establece la dirección alterna del transceptor. Para lograrlo se implementó una función que pudiese asignar el direccionamiento según parámetros operativos de la red.

También se desarrollaron y adaptaron funciones para el cambio del modo de operación del transceptor, el establecimiento del canal de operación y el control de la potencia de salida del transceptor con el fin de ofrecer diferentes posibilidades de configuración del transceptor de comunicación según fuese requerido.

A los nombres de las funciones creadas en esta segunda capa se le agregó el prefijo “MiMAC” para una identificación más efectiva de los procedimientos de esta capa. Todas estas funciones estaban incluidas en un archivo de código fuente llamado MRF24J40MA.c. En este archivo también se implementaron las funciones de la capa física. También se desarrolló un archivo de cabecera llamado MCHP_MAC.h que contenía la estructura global de la trama recibida y otras estructuras de datos para las tramas a transmitir. También declaraban acá los prototipos de las funciones “MiMAC”.

Hasta este momento se ha presentado la implementación del conjunto de funciones relacionado a las capas físicas (PHY) y de acceso al medio (MAC). El controlador de software o "driver" MiWi, implica el acceso a las funciones de estas dos capas, sin embargo para poder realizar pruebas de alcance de la comunicación y validar la estabilidad del controlador se procedió a implementar un conjunto de funciones (API) concernientes a la capa de aplicación cuya implementación se detalla en seguida.

La capa de aplicación constituye el punto de contacto más directo que tiene un desarrollador para crear y administrar la red de nodos CRTecMote. Las funciones de aplicación permiten a su vez, el manejo de las funciones de la capa de control de acceso al medio (MAC) y la capa física (PHY). Esto le permite al desarrollador controlar los servicios de dicha capa, como el tipo de información que se envía o recibe, con la ventaja de que no se tiene que preocupar por la estructuración de las tramas ni de la compleja administración de los registros del transceptor. Basándose en el protocolo de conexión punto a punto, se implementó esta capa con las características que se detallan a continuación.

Una función principal llamada P2Ptask, es la encargada de mantener el funcionamiento de la red. El diagrama de flujo que sigue esta función, se muestra en la figura 32.

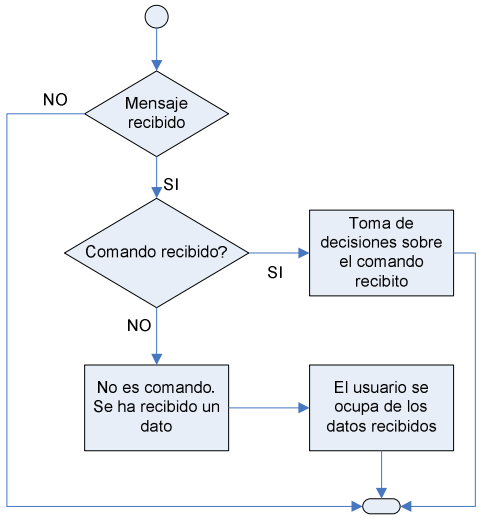


Figura 32. Diagrama de flujo del funcionamiento general de la función P2Ptask

En la capa de aplicación fue donde se implementaron las funciones para la asociación de los dispositivos a la red de la red CRTecMote. Esto se logró al desarrollar una función llamada *Addconnection*, la cual se encargaba de la administración de una tabla de conexiones, donde se almacenaba la información de los nodos que habían sido agregados a la red por parte de un nodo coordinador (FFD). Además esta función decidía el tipo de respuesta de petición de conexión a enviar a los nodos de medición (RFD) que solicitaran incorporarse a la red.

La función implementada para este propósito sigue el algoritmo del diagrama de flujo mostrado en la figura 33.

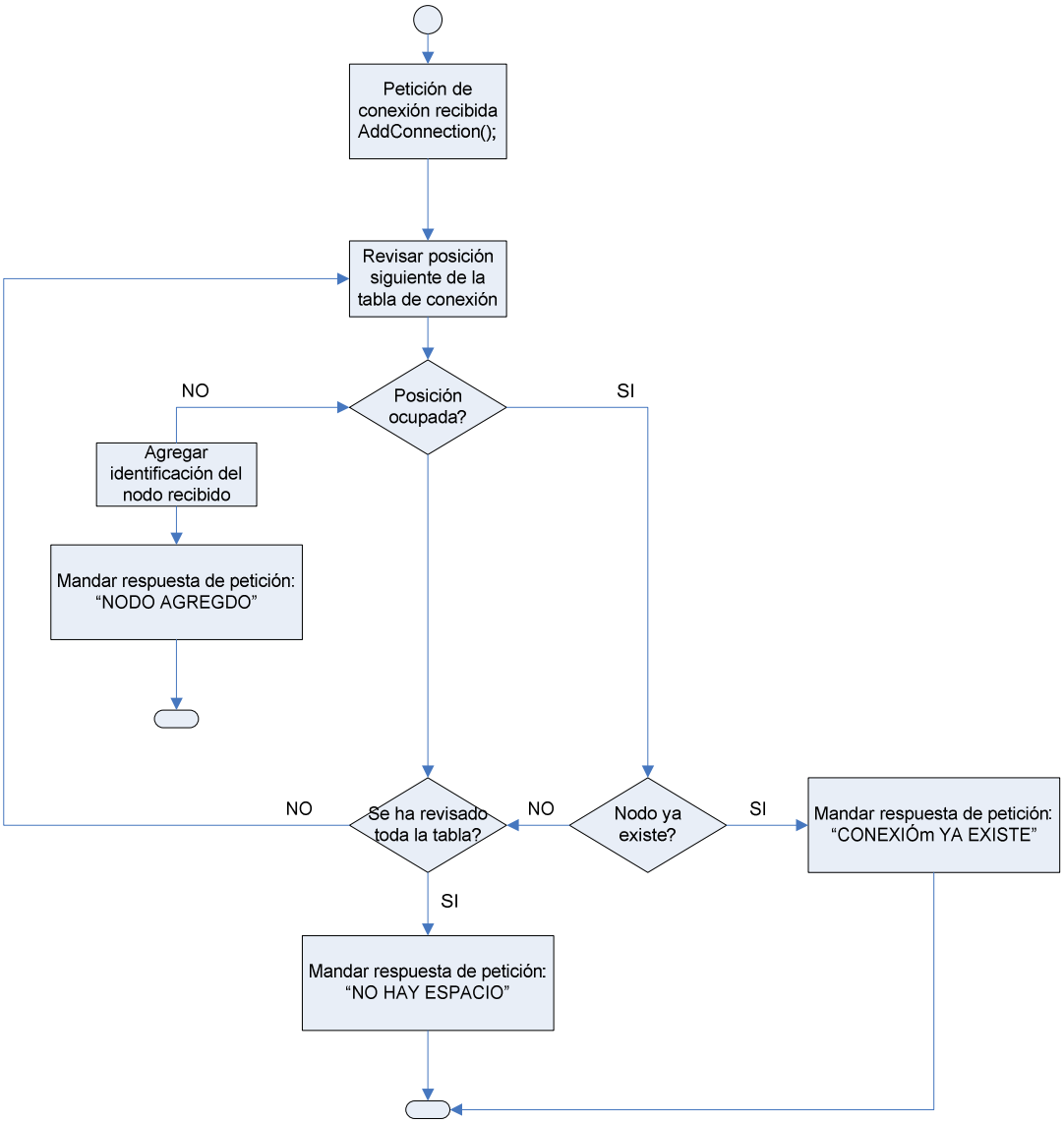


Figura 33. Diagrama de flujo del proceso de asociación de dispositivos en una red CRTecMote

Ante una petición de conexión por parte de un nodo de medición (RFD), el nodo sumidero (FFD) que funcionaba como punto de acceso a la red verificaba primeramente, la cantidad de bytes que traía consigo la petición, si eran menos de 3 bytes, lo que corresponde a una petición de conexión reducida, quería decir que ésta no contenía información del dispositivo que la enviaba la información, por lo que se asumía que este dispositivo estaba en proceso de búsqueda de redes disponibles, por lo que se enviaba una respuesta de tipo STATUS_ACTIVE_SCAN. Luego de verificar el tipo de petición, se determinaba si el nodo que realiza la petición ya estaba registrado dentro de la tabla de conexiones activas, en cuyo caso le respondía el comando STATUS_EXISTS.

Si el nodo no existe, pero sin embargo no hay espacio, se manda una respuesta de tipo STATUS_NOT_ENOUGH_SPACE. Por último si hay espacio disponible, se agrega el nodo a la tabla y se manda una respuesta de tipo STATUS_SUCCESS.

También se encuentran algunas otras particularidades de la capa:

Inicialización del protocolo: esta función se encarga de la inicialización de toda la red, variables globales, estructuras de recepción, transmisión, de mensajes indirectos, tablas de conexiones e inicialización de la capa de acceso al medio.

Creación de una red, realizada por el coordinador; o búsqueda de redes disponibles, realizada por un RFD. Esta última se realiza mandando peticiones de conexión reducidas, las cuales son recibidas por cualquier coordinador, el cual le envía una respuesta, indicando así la existencia de una red personal.

Exploración de energía para determinar niveles de interferencia: se utiliza una función “MiMAC” para determinar niveles de ruido electromagnético en todos los canales. Al final se escoge el de menor interferencia.

Transmisión de mensajes indirectos: dado que los RFD tienen la posibilidad de pasar a modo de bajo consumo dormido, estando en este estado no serían capaces de recibir datos, por lo que en estos casos, el coordinador almacena la información destinada para este RFD específico durante cierto tiempo, esperando por una petición de datos.

Emisión de mensajes generales (en inglés: Broadcast): esta característica permite transmitir información sin especificar la dirección de destino, es decir, va dirigida hacia todos los nodos RFD de la red.

Re-sincronización de dispositivos: si un RFD pierde conexión con la red, por ejemplo, si se sale del alcance de la red, el dispositivo intenta comunicarse de nuevo, mandando peticiones de conexión reducidas, es decir, sin la información propia del nodo RFD. El número de intentos se define en los archivos de cabecera. Esta función requirió modificaciones, ya que solo soportaba el caso en el que la pérdida de conexión se daba por encontrarse fuera del alcance del coordinador. Otro inconveniente es que ante este tipo de peticiones, cualquier dispositivo, sea coordinador o no, puede contestarle al nodo que intenta re-sincronizarse. La figura 34 muestra el diagrama de flujo para este proceso.

Al igual que para las capas inferiores, los nombres de las funciones creadas en esta capa de aplicación, contienen el prefijo “MiApp” para la identificación de la capa. Las funciones están incluidas en un archivo fuente llamado P2P.c, en un archivo de cabecera llamado P2P.h; y en otro denominado MCHP_API.h, se definen las estructuras de recepción transmisión de datos desde la capa de acceso al medio, así como los prototipos de las funciones del API de la capa de aplicación.

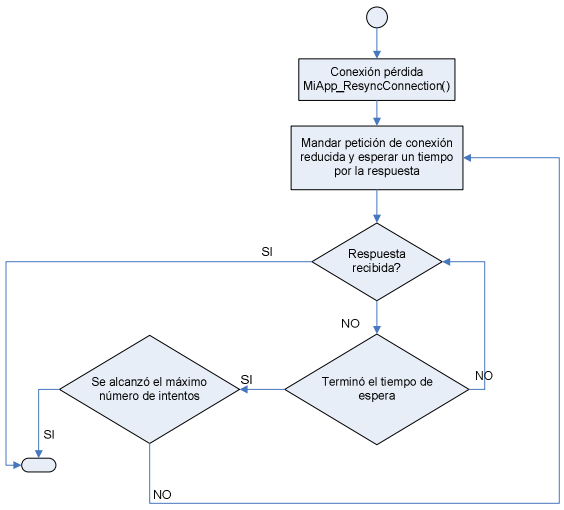


Figura 34. Diagrama de flujo del proceso de re-sincronización de dispositivos en una red CRTecMote.

Para obtener una migración funcional del protocolo, teniendo en cuenta su aplicación final, se realizaron algunas modificaciones. Esto se debe a que originalmente el protocolo de conexión punto a punto fue desarrollado para aplicaciones de este tipo, por lo que no contempla por ejemplo asociaciones de varios dispositivos controlados por un solo nodo. En esta sección se detalla la modificación del protocolo de red para habilitar el servicio de direccionamiento dinámico.

Como se mencionó en secciones anteriores, los dispositivos, poseen una dirección específica y única de 8 bytes. Esto es el equivalente a las direcciones IP manejadas en los protocolos de internet. Sin embargo, dentro del protocolo MiWi™ P2P, la asignación de estas direcciones se realiza a nivel de programación en el nodo coordinador, es decir, estáticamente, teniendo que reprogramar el dispositivo si se deseara cambiar el conjunto de direcciones a utilizar en algún momento.

Como los nodos CRTECMOTE están diseñados para distintas aplicaciones, se decidió implementar la asignación de direcciones de forma dinámica tal y como lo hace el protocolo de redes DHCP (siglas en inglés para Dynamic Host Configuration Protocol). De esta manera, luego de recibir la respuesta de conexión, cada nodo esclavo recibe una nueva dirección que lo identifica dentro de la red. Para lograr esto se agregó y modificó código del protocolo, en varios puntos diferentes dentro del archivo fuente P2P.c, tanto para el coordinador como para los nodos. La figura 35 se muestra el diagrama de flujo que sigue el algoritmo de asignación dinámica de direcciones a nivel de la función *Addconnection*.

El proceso de asignación dinámica inicia con la programación de una misma dirección genérica en cualquier dispositivo, la cual consiste en un arreglo de 8 posiciones inicializado con un carácter hexadecimal cualquiera. Una vez que el coordinador crea la red, si se recibe una petición de conexión, se detecta la dirección genérica del solicitante. Luego se asocia el dispositivo como se describió anteriormente y se almacena en alguna posición de la tabla de conexiones la nueva dirección para el dispositivo creada según la posición en esta tabla y el arreglo general para nuevas direcciones.

Finalmente se envía la nueva dirección al dispositivo como parte de la respuesta de petición de conexión “STATS_SUCCES”.

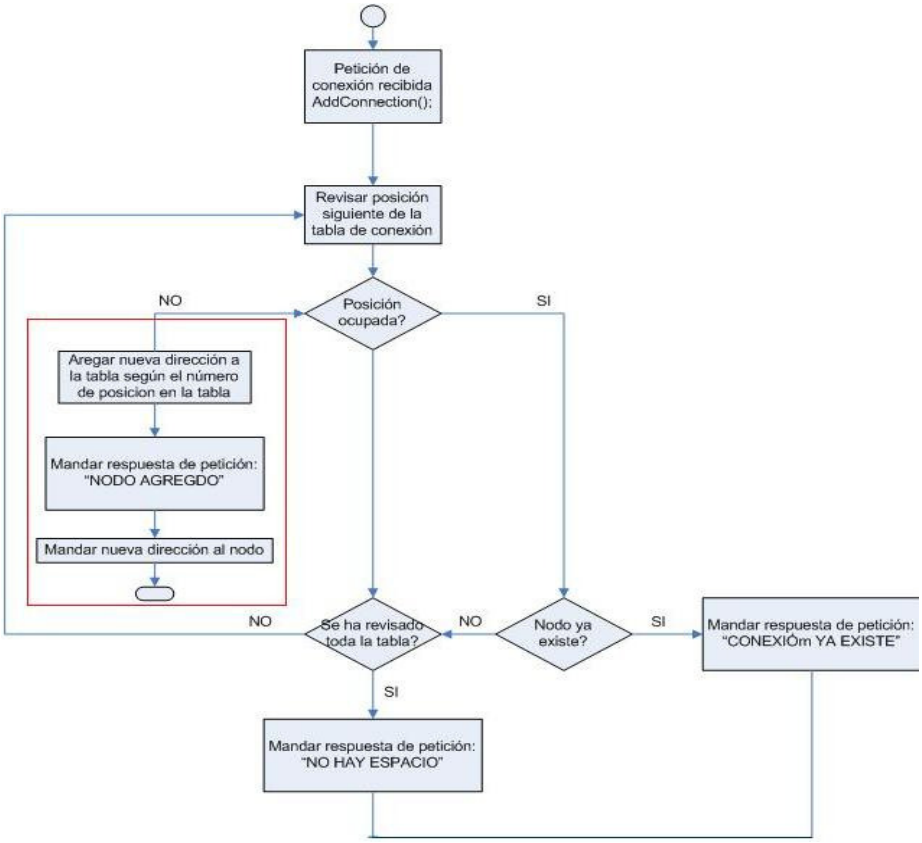


Figura 35. Diagrama de flujo del procedimiento de asignación dinámica de direcciones en una red CRTecMote.

La función original de re-sincronización cubría únicamente el caso en el que la conexión se perdía si se salía del alcance del coordinador. Sin embargo, existen casos más generales de pérdida de conexión e igual de comunes que el anterior. En esta sección se describe la modificación realizada a esta función para contemplar también la situación en la que, por alguna razón, el coordinador debe reiniciarse o ha cambiado el canal de operación. Si este es el caso, se reinicia la tabla de conexiones y desaparecen todos los nodos agregados, ocasionando pérdida de conexión del RFD.

Para solucionar este inconveniente, se modificó el código de re-sincronización para que, en cada intento, se realice el proceso de asociación de nodo, o sea, mandar una petición de conexión completa y recibir la respectiva respuesta de

éxito. De esta forma se agrega nuevamente el nodo a la red y mediante el algoritmo de direccionamiento dinámico, se le asigna una nueva dirección al dispositivo. También se eliminó el número máximo de intentos y se agregó una exploración de todos los canales para contemplar el caso en el que el coordinador simplemente cambió de canal de operación. El algoritmo se muestra en la figura 36.

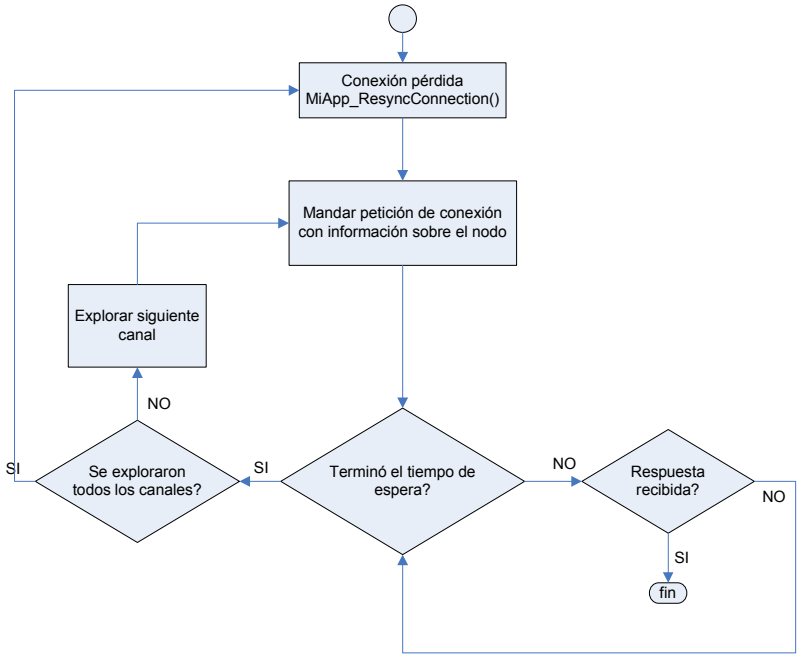


Figura 36. Diagrama de flujo de re-sincronización modificado.

Una vez implementados los algoritmos necesarios para el controlador de red de área local se procedió a realizar una caracterización del consumo energético de dicho controlador integrado al sistema operativo SIWA RTOS. Para ello se definieron un conjunto de pruebas que se detallan a continuación.

La primera prueba fue medir el consumo de potencia eléctrica consumida por el nodo y la segunda fue una caracterización de la intensidad de señal recibida para distintos entornos operativos de la red inalámbrica de sensores CRTECMOTE.

Para la primera de las pruebas descritas anteriormente se alimentó el nodo con una tensión eléctrica de 3.3V y se midió la corriente consumida por un nodo que se definió como coordinador desde el momento en que inicia la configuración del microcontrolador hasta el punto en el que asocia un nodo a la red y le envía,

exactamente, dos peticiones de datos con el promedio de las últimas 5 muestras del ADC, cada una.

La gráfica de la figura 37 muestra la variación de corriente en función del tiempo para un nodo coordinador. Las muestras de corriente fueron tomadas con el equipo Source Meter Unit modelo 1636A de la marca Keithley, con un periodo de muestreo de 0.03495 segundos

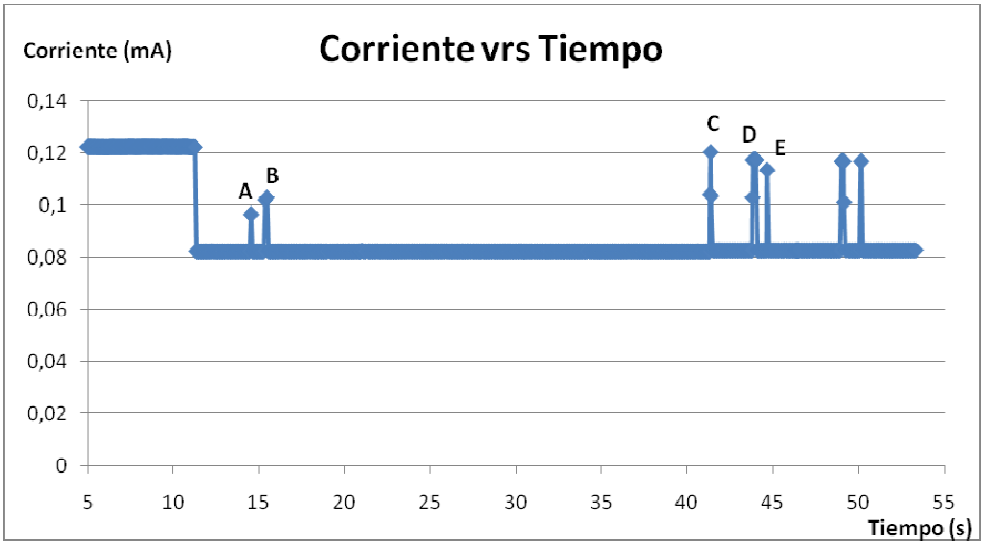


Figura 37. Gráfico de variación de consumo de corriente contra tiempo para un nodo CRTecMote RFD ejecutando SIWA con el controlador de red de área local.

De la gráfica de la figura 37 se pueden destacar dos aspectos importantes; El primero consiste en el análisis del proceso de creación de una red y la asociación de un dispositivo RFD. Como puede apreciarse, en los primeros 6 segundos de muestreo, se observa un escalón de corriente. Éste es el producto de la inicialización del dispositivo, seguido de la exploración de energía para determinar el canal óptimo de operación y la creación de la red. Luego, si se avanza en el tiempo, se encuentran dos picos de corriente, correspondientes a los puntos A y B de la gráfica. Dado que inmediatamente después de la creación de la red, viene el proceso de asociación, estos picos corresponden a las petición reducida que se manda durante una detección activa de redes disponibles realizada por parte del nodo de medición, cuyo proceso dura aproximadamente 28 segundos, por lo que

se puede decir que estas peticiones fueron recibidas en el coordinador, en uno de los primeros 4 canales del espectro de 2.4GHz. De lo contrario, los picos se encontrarían más adelante en el tiempo.

En seguida, se encuentra otro pico de corriente denominado punto C en la gráfica. Si se descuentan, aproximadamente 2 segundos, que equivalen al intervalo de tiempo que va desde el momento en que el coordinador termina la creación de la red, hasta el momento en que se alimenta al nodo de medición (RFD), se puede ver que el punto C corresponde al punto B de esta última y es por lo tanto el instante en el que se realiza la asociación del dispositivo.

De esta manera, los picos de corriente señalados en D y E, corresponden entonces, al envió de una petición de datos por parte del coordinador hacia un nodo RFD y la recepción del promedio de las últimas 5 muestras del ADC de este nodo. Del análisis de los tiempos de muestreo, se determinó que este proceso de petición dura aproximadamente 859.45 milisegundos, mientras que el proceso de asociación dura 6.524 milisegundos.

El otro aspecto importante a resaltar de esta gráfica es el consumo de corriente. Como la gráfica representa una operación real del nodo coordinador, se puede decir que el consumo máximo se da en los instantes de inicialización. Posteriormente se obtendrán picos de corriente no mayores a los 350 milisegundos. A partir de un muestreo de corriente se puede afirmar que si se excluye el intervalo de inicialización, se obtiene un promedio de corriente de 82.265 mA lo cual si se considera una alimentación fija de 3.3V, se traduce en un consumo de potencia promedio de 272.731 mW por parte del nodo coordinador.

Al aire libre, se realizó un proceso de medición para estimar la intensidad de potencia de las señales recibidas a partir de la medición del valor RSSI, el cual tiene un rango de 0 a 255 y es calculado por el mismo transceptor. La visualización de este valor se puede obtener por medio de una herramienta de depuración que permite imprimir mensajes durante la operación del microcontrolador. Desde 0 hasta los 150 m aproximadamente, manteniendo la línea vista entre los dispositivos, se obtuvieron los resultados de la tabla 5.7.

Como los datos fueron recibidos luego de una petición del coordinador, en la columna derecha se anotó el número de intentos necesarios para recibir el mensaje. De la tabla se aprecia que antes de los 100m exactos, se obtiene un 100% de la integridad de la información, ya que fue necesario únicamente un solo intento de petición. Sin embargo, más allá de los 100m, se empezó a tener dificultades en la recepción, teniendo que presionar el botón de petición hasta 3 veces. Esto concuerda con las especificaciones de la hoja de datos del MRF24J40, en donde se definen un rango de 100m al aire libre y en línea vista. Según las mediciones, se comprueba que después de esta distancia se pierde la integridad de la información.

Existe además, un valor más representativo para determinar la calidad del enlace o de la señal recibida desde otro dispositivo. Este valor es el indicador de la calidad del enlace (LQI), que consiste en un grado de correlación entre la propagación de secuencias del transceptor y las tramas recibidas. El rango de variación es de 0 a 255 en donde 0 indica un pésimo enlace y 255 representa un enlace de muy alta calidad. Como se puede apreciar de los resultados de la tabla 2, este valor se mantiene mucho más constante que la estimación de potencia a través del valor RSSI. Además, los resultados obtenidos muestran un tipo de enlace intermedio, que al igual que el valor RSSI se degrada con la distancia, aunque no tan rápidamente. Por esto se puede decir que la calidad del enlace para la red implementada se mantiene relativamente constante en un nivel intermedio para una distancia menor a los 150m.

Tabla 8. Caracterización cobertura de una red CRTecMote

Distancia hasta el nodo coordinador [m]	RSSI	Potencia (dBm)	LQI	Número de intentos
0	240	-40	110	1
5	158	-56	115	1
10	127	-63	110	1

15	112	-66	113	1
20	101	-68	113	1
25	71	-74	115	1
30	60	-76	114	1
35	16	-84	117	1
40	4	-87	111	1
45	0	<-90	103	1
50	7	-87	113	1
55	9	-86	111	1
60	2	-88	114	1
65	0	<-90	101	1
70	2	-88	106	1
75	9	-86	111	1
80	0	<-90	109	1
85	0	<-90	108	1
90	0	<-90	108	1
95	0	<-90	108	1
100	0	<-90	110	1
110	0	<-90	106	2
120	0	<-90	109	3
130	0	<-90	108	3

En resumen, se obtuvo un módulo de comunicaciones para un nodo CRTecMote caracterizado por el protocolo de redes inalámbricas MiWi^{TM} P_2P. El nodo está en capacidad de crear una red en topología estrella y aceptar las conexiones de 4 nodos como máximo, aunque esta cantidad es configurable. La red posee las características habituales de transmisión y recepción de una trama de datos con un límite de 127 bytes, a la cual se le agregan, posteriormente, bits de control y sincronización. La red también presenta la característica de direccionamiento dinámico típica de la capa 3 del modelo OSI para redes. También se encuentran funciones de re-sincronización, identificación del destino de mensajes, transmisión de mensajes indirectos, modos de operación de bajo consumo para nodos esclavos, búsqueda de canales con menos nivel de ruido electromagnético, búsqueda de redes disponibles, entre otras. En cuanto al protocolo en general, la programación a nivel de capas permite una mayor comprensión y simplicidad del protocolo.

Controlador de administración de red de área extendida vía MODEM

Para este propósito se desarrolló un controlador para un dispositivo de conexión a una red de telefonía móvil o "*data card*", con el propósito que el nodo coordinador de CRTecMote (FFD) se pudiese controlar a distancia por medio de un dispositivo móvil.

Para la solución del problema de conexión a la red de telefonía móvil se utilizó un nodo CRTecMote FFD con microcontrolador PIC32MX460512. Esto debido a su capacidad de funcionar como administrador de dispositivos de tipo USB y también por la capacidad de soportar un sistema operativo de tiempo real como SIWA-RTOS.

La aplicación USB seleccionada para la solución fue la de tipo CDC-USB-HOST de la empresa Microchip, ya que es una buena aproximación al comportamiento que se desea del MODEM EDGE-USB. La aplicación CDC-USB-HOST USB funciona bajo el modelo USB Embedded Host Firmware de la figura 38. En este modelo se basó la solución planteada al problema de comunicación por medio de la red de telefonía móvil.

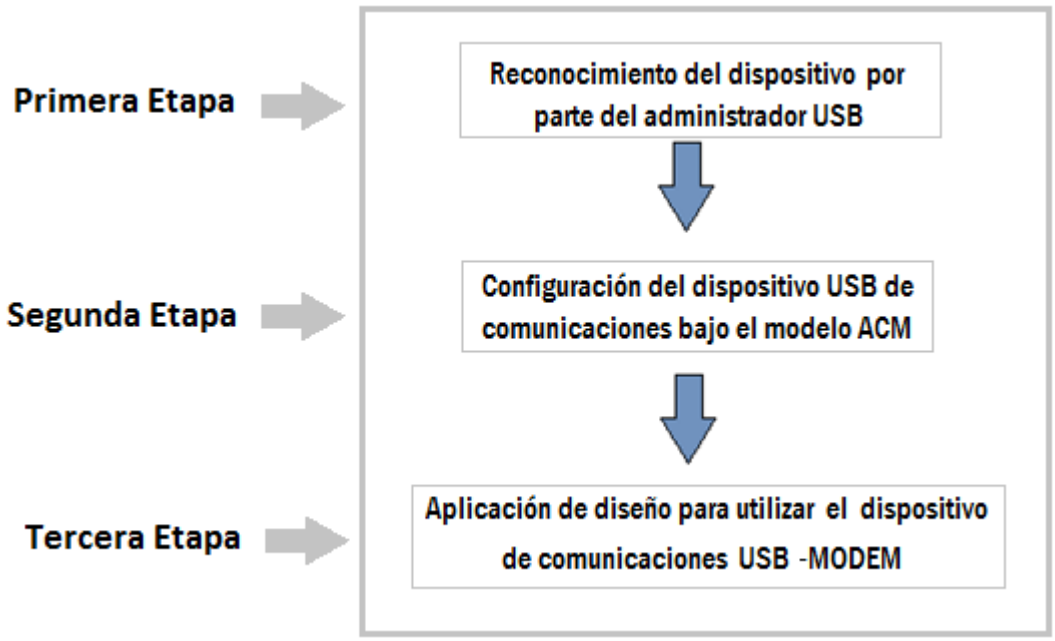


Figura 38. Arquitectura del la pila de aplicaciones Host-USB de Microchip.

El dispositivo seleccionado como interfaz de comunicaciones fue un Modem de la marca EDGE, ya que está diseñado para ser controlado por comandos AT Hayes y también por funcionar en las frecuencias de operación de las redes de telefonía celular del Instituto Costarricense de Electricidad.

Por último se seleccionó el sistema operativo SIWA-RTOS ya que es un sistema diseñado para ser de bajo consumo de potencia, lo cual es un factor determinante en el diseño y desarrollo de futuras aplicaciones de la plataforma de redes inalámbricas de sensores CRTECMote.

Para verificar el estado de reconocimiento del modem EDGE-USB, se localizó la primera bandera de interrupción de conexión del dispositivo, con la cual se logró comprobar que efectivamente la lógica de resistencia de pull-up encargada de informar al controlador sobre una nueva conexión en el bus USB estaba funcionando.

```
if (((usbHostState & STATE_MASK) != STATE_DETACHED) && !U1IRbits.ATTACHIF)
```

Figura 39. Instrucción de control para la interrupción por hardware

Luego de verificar que la interrupción de hardware se llevó a cabo con éxito, se procedió a identificar los estados de enumeración por software que contempla el protocolo USB 2.0; específicamente la aplicación USB-CDC Host. En el diagrama de la figura 40 se presenta la secuencia completa de los pasos de enumeración de un dispositivo USB conectado al nodo CRTecMote. Cada uno de estos procesos fue etiquetado con mensajes equivalentes a break-points de programación, con el objetivo de imprimir en el Output Window de la herramienta de desarrollo (MPLAB) el estado del proceso y controlar el estado del programa.

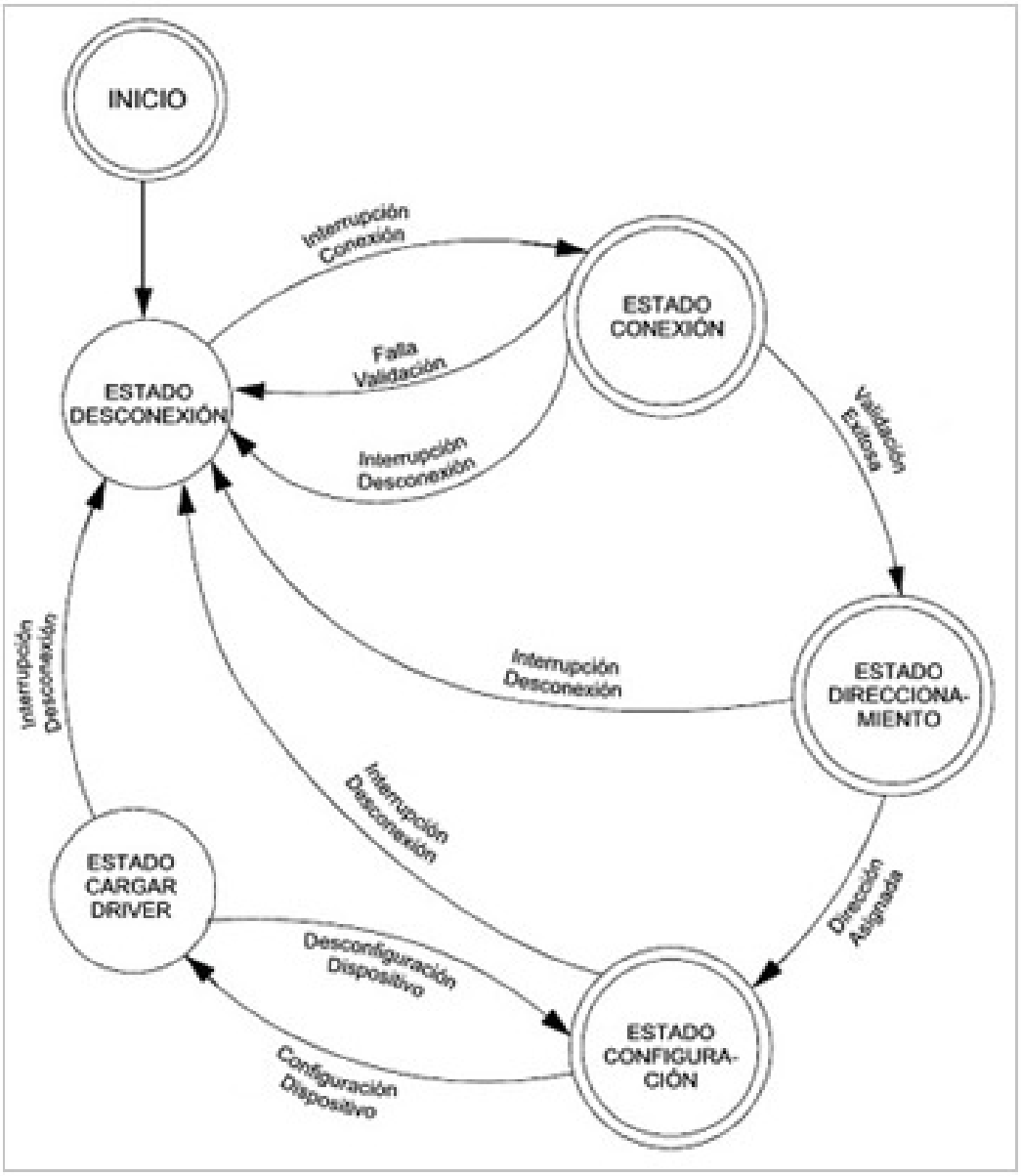


Figura 40. Estados de inicialización de un dispositivo USB

Además se insertaron etiquetas de depuración de datos las cuales se crearon para conocer el valor de los registros asociados a las solicitudes de tipo USB-Host > USB Device y viceversa.

Estas etiquetas se insertaron en los procesos de enumeración descritos en el esquema de la figura 40, y de esta manera se logró controlar el estado de reconocimiento del dispositivo por parte del Host USB y decidir si el dispositivo estaba listo para ser configurado.

Como parte de los resultados obtenidos en la etapa de investigación, se determinó que un dispositivo USB enumerado como Comunication Device Class (Dispositivo de Comunicaciones), está diseñado para identificarse como un dispositivo de almacenamiento masivo de datos.

Por lo tanto se procedió a reconocer cada una de las etapas de petición de descriptores con el objetivo de identificar el paso en el cual el Host USB decide qué tipo de interface se utilizará para controlar el dispositivo y así poder seleccionar la interfaz de comunicación de datos en lugar de la interfaz de almacenamiento de datos. A continuación se presenta el desglose de las peticiones de descriptores más relevantes del proceso de enumeración:

Obtener Descriptor del Dispositivo: La solicitud se realizó como una transacción dirigida al End-Point cero, y seguidamente se verificó que los datos recibidos fueran congruentes con los datos recopilados en la etapa de investigación por medio de los analizadores de protocolo USBlyzer y USBTrace. La figura 41 muestra la implementación de la función de reconocimiento del descriptor de dispositivo.

```
case SUBSTATE_GET_DEVICE_DESCRIPTOR_SIZE:

_USB_InitControlRead( usbDeviceInfo.pEndpoint0, pEP0Data, 8, pEP0Data, 8 );
_USB_SetNextSubSubState();
SendMessage( 2, "HOST: Getting Device Descriptor size.\r\n" );
```

Figura 41. Código de revisión de descriptor de interfaz del dispositivo

Validar VID & PID, o Class Type: para realizar la validación del dispositivo se plantearon dos soluciones, la primera de ellas fue la de tratar de validarlo por sus etiquetas Vendor ID y Product ID (VID-PID). Los valores de las etiquetas VID-PID para el dispositivo de comunicaciones EDGE-USB fueron asignados como 0x0471 y 0x1237 de su valor en hexadecimal. Estos valores se introdujeron en la primera tabla de identificación del dispositivo de la figura 42.

```
USB_TPL usbTPL[NUM_TPL_ENTRIES] =
{
//    VID & PID or                          Client
//    Class, Subclass & Protocol      Config Driver  Flags
    { INIT_VID_PID( 0x0471, 0x1237 ),   1,     0,    {TPL_SET_CONFIG} }  // Validación por VID-PID
};
```

Figura 42. Función para la validación por etiquetas VID-PID

Una vez insertados los valores de las etiquetas, se procedió a verificar el proceso de reconocimiento de dispositivo por medio de una declaración global del código denominado *ALLOW GLOBAL_VID_PID.*

```
#ifdef ALLOW_GLOBAL_VID_AND_PID
if (((usbTPL[i].device.idVendor  == pDesc->idVendor ) &&
     (usbTPL[i].device.idProduct == pDesc->idProduct)) ||
    ((usbTPL[i].device.idVendor  == 0xFFFF) &&
     (usbTPL[i].device.idProduct == 0xFFFF)))
#else
if ((usbTPL[i].device.idVendor  == pDesc->idVendor ) &&
    (usbTPL[i].device.idProduct == pDesc->idProduct)   )
#endif
{
    SendMessage( 2, "HOST:Device validated VID-PID\r\n" );
    usbDeviceInfo.flags.bfUseDeviceClientDriver = 1;
}
```

Figura 43. Estructura de control de identificación del dispositivo por etiquetas VID-PID

El segundo intento de validación se creó a partir de una tabla que lista de códigos predeterminados por el estándar USB 2.0 para la clase de tipo Comunication Class Device (CDC).

Los códigos que identifican al dispositivo de una forma más concreta según la aplicación para la cual este diseñado permite al programado definir la estrategia de atención. Dicha estructura de datos se puede visualizar en la figura 44.

```
USB_TPL usbTPL[] =
{
    { INIT_CL_SC_P( 2ul, 2ul, 1ul ), 0, 0, {TPL_CLASS_DRV} }, // Communication Interface
     // Communication Device/Interface Class , Abstract Control Mode , Common AT Commands
    { INIT_CL_SC_P( 0x0Aul, 0ul, 0ul ), 0, 0, {TPL_CLASS_DRV} }  // Data Interface
};
```

Figura 44. Validación por etiquetas por clase de dispositivo

Este tipo de validación también fue autenticado en la rutina de identificación de dispositivo por medio de de la etiqueta “Host: Device validated by class” que se muestra en la figura 45.

```
if ((usbTPL[i].device.bClass    == pDesc->bDeviceClass    ) &&
    (usbTPL[i].device.bSubClass == pDesc->bDeviceSubClass) &&
    (usbTPL[i].device.bProtocol == pDesc->bDeviceProtocol)    )
{
                SendMessage( 2, "HOST:Device validated by class\r\n" );
    #ifdef DEBUG_MODE
        UART2PrintString( "HOST: Device validated by class\r\n" );
    #endif
    usbDeviceInfo.flags.bfUseDeviceClientDriver = 1;
}
```

Figura 45. Estructura de control de identificación del dispositivo por etiquetas clase

Estructura de la memoria para dar soporte al dispositivo: Al localizar la etapa de direccionamiento, se reconoció como prioridad identificar la dirección de memoria que el controlador asignó al dispositivo. También se analizaron las peticiones específicas al END-POINT cero relacionadas con la etapa de direccionamiento y tamaño de espacio en memoria para controlar el dispositivo.

```
case SUBSUBSTATE_SEND_SET_DEVICE_ADDRESS:
            SendMessage( 2, "HOST: Setting device address.\r\n" );
    usbDeviceInfo.deviceAddress = USB_SINGLE_DEVICE_ADDRESS;

    // Set up and send SET ADDRESS
    pEP0Data[0] = USB_SETUP_HOST_TO_DEVICE | USB_SETUP_TYPE_STANDARD | USB_SETUP_RECIPIENT_DEVICE;
    pEP0Data[1] = USB_REQUEST_SET_ADDRESS;
    pEP0Data[2] = usbDeviceInfo.deviceAddress;
    pEP0Data[3] = 0;
    pEP0Data[4] = 0;
    pEP0Data[5] = 0;
    pEP0Data[6] = 0;
    pEP0Data[7] = 0;
    _USB_InitControlWrite( usbDeviceInfo.pEndpoint0, pEP0Data, 8, NULL, 0 );
    _USB_SetNextSubSubState();
    break;
```

Figura 46. Petición de direccionamiento del dispositivo

Configuración a partir de los descriptores: Se procedió a autenticar el estado de configuración del dispositivo. Con este estado de configuración el controlador PIC32MX460512 del nodo CRTecMote verifica si es capaz de controlar el MODEM USB por medio de los controladores previamente configurados para la aplicación. Esta petición de configuración llevó la etiqueta "Host: Set Configuration" y con ella se identifica si efectivamente el nodo alcanzó a realizar la petición con éxito. En la figura 47 se muestra el resultado de la implementación de dicha rutina.

```
case SUBSUBSTATE_SEND_SET_CONFIGURATION:
    SendMessage( 2, "HOST: Set configuration.\r\n" );
    #ifdef DEBUG_MODE
        UART2PrintString( "HOST: Set configuration.\r\n" );
    #endif

    // Set up and send SET CONFIGURATION.
    pEP0Data[0] = USB_SETUP_HOST_TO_DEVICE | USB_SETUP_TYPE_STANDARD | USB_SETUP_RECIPIENT_DEVICE;
    pEP0Data[1] = USB_REQUEST_SET_CONFIGURATION;
    pEP0Data[2] = usbDeviceInfo.currentConfiguration;
    pEP0Data[3] = 0;
    pEP0Data[4] = 0;
    pEP0Data[5] = 0;
    pEP0Data[6] = 0;
    pEP0Data[7] = 0;
    _USB_InitControlWrite( usbDeviceInfo.pEndpoint0, pEP0Data, 8, NULL, 0 );
    _USB_SetNextSubSubState();
    break;
```

Figura 47. Rutina implementada para configuración del dispositivo

Por último, se insertó una etiqueta denominada "HOST: Configuration Complete. EDGE MODEM (GSM-SMS)" con el objetivo de comprobar que la etapa de enumeración había concluido. En la figura 48 se muestra el código implementado para dicho propósito.

```
case SUBSUBSTATE_SET_CONFIGURATION_COMPLETE:
    SendMessage( 2, "HOST:Configuratión Complete. EDGE MODEM (GSM-SMS).\r\n" );
    // Clean up and advance to the next state.
    _USB_InitErrorCounters();
    _USB_SetNextSubSubState();
    break;
```

Figura 48. Rutina implementada para confirmación de estado de configuración del modem USB utilizado en el nodo coordinador CRTecMote.

Servicio de transmisión de datos utilizando mensajes de texto SMS

El proceso de transmisión de datos desde el nodo coordinador hacia un nodo sumidero que funcionara como centro de fusión de los datos se diseñó utilizando para ello las capacidades de mensajería de texto de la red de telefonía móvil existente así como mediante la implementación de una serie de rutinas dentro del nodo coordinador CRTecMote capaz de interactuar con los nodos de recolección de datos para pre-procesar la información de los sensores de campo.

Esto se logró al migrar la aplicación creada para el gestionar el servicio de mensajería de texto (SMS) utilizando el PIC32 hacia el sistema operativo de tiempo real SIWA-RTOS. La incorporación de las funcionalidades de conexión de red se realizó mediante el desarrollo de un controlador de software o "driver" compatible con las estructuras de ejecución del SIWA RTOS. En seguida se describen los procedimientos realizados así como los resultados obtenidos.

1. Se movieron todos los Header Files contenidos en la aplicación CDC-USB-HOST a la carpeta de los archivos a incluir de SIWA-RTOS.
2. Se creó con MPLAB el workspace del SIWA-RTOS.
3. Se cargó los archivos USB_host.c, USB_config.c, USB_CDC_Host.c el código de aplicaciones del SIWA-RTOS.
4. Se crearon las nuevas direcciones de búsqueda de los archivos a incluir. La aplicación creada para comprobar los procedimientos mencionados se creó en base al diagrama de flujo presentado en la figura 49.

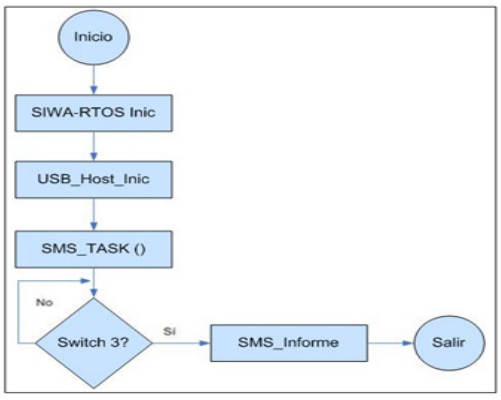


Figura 49. Rutina implementada para integración de la biblioteca CDC a SIWA-RTOS

Por medio de la tarea SMS_GSM, se creó una aplicación que envía un mensaje de texto con una notificación sobre el estado actual del sistema operativo. En la figura 50 se muestra la estructura de la tarea empleda para este propósito.

```
/*-----------------------------------------------------------*/

void SMS_GSM( void *pvParameters )
{
const char *pcTaskName = "SMS_GSM\n";
volatile unsigned long ul;
    for( ;; )
    {   UART2Init();
        vPrintString( pcTaskName );
        if (!mGetSW3){
                        SMS_TASK();
                        }
        for( ul = 0; ul < mainDELAY_LOOP_COUNT; ul++ )
        {

        }
    }
}
```

Figura 50. Estructura de la tarea creada en SIWA-RTOS para el envío de mensajes de texto

En la figura 51 se muestra el resultado de implementación de la rutina de autenticación con la red de telefonía móvil del instituto costarricense de electricidad (ICE).

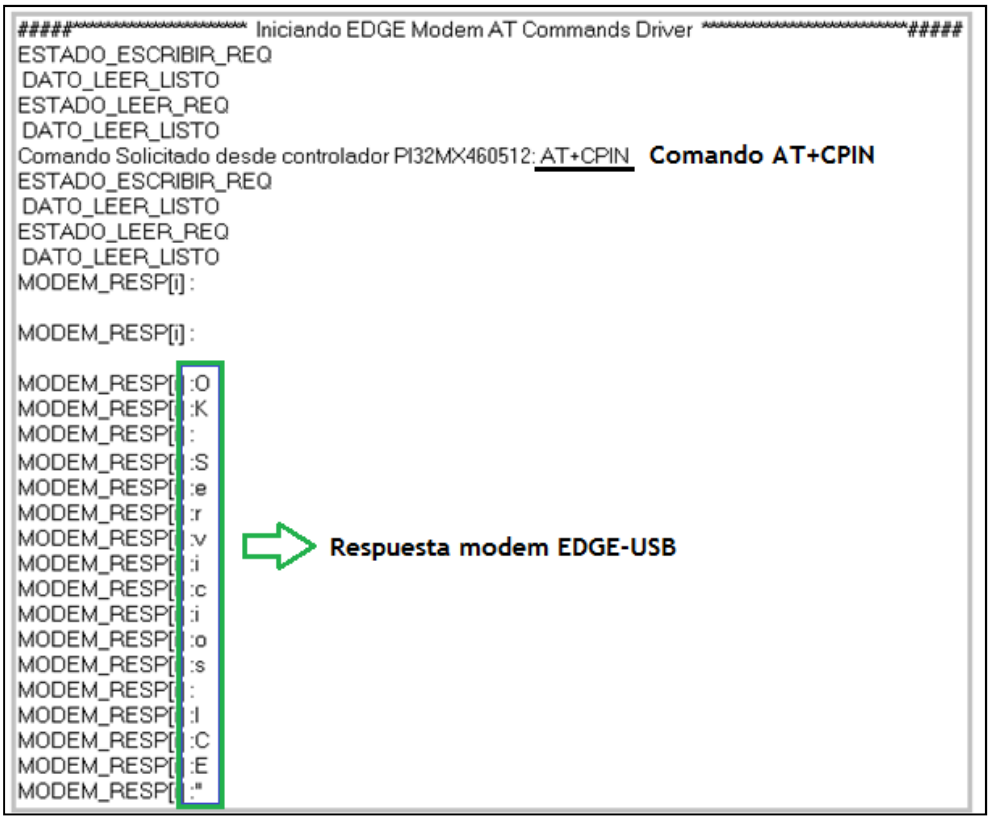


Figura 51. Mensaje de autenticación del nodo CRTecMote con la red de telefonía móvil del ICE

Una vez que se logró que el nodo de la red inalámbrica CRTecMote se conectara a la red de telefonía móvil se procedió a probar las rutinas de mensajería de texto que se habían desarrollado en la implementación del controlador de comunicación vía MODEM GSM.

Para tal propósito se envió un mensaje desde un teléfono móvil conectado a una aplicación de computador (NOKIA PC SUITE) hacia el nodo CRTecMote, con el fin de verificar si éste lograba recibirlo exitosamente. Para ello se revisó la cadena de datos CMTI en el MODEM. La figura 52 muestra tanto el mensaje enviado desde el computador, como el mensaje recibido en el nodo CRTecMote.

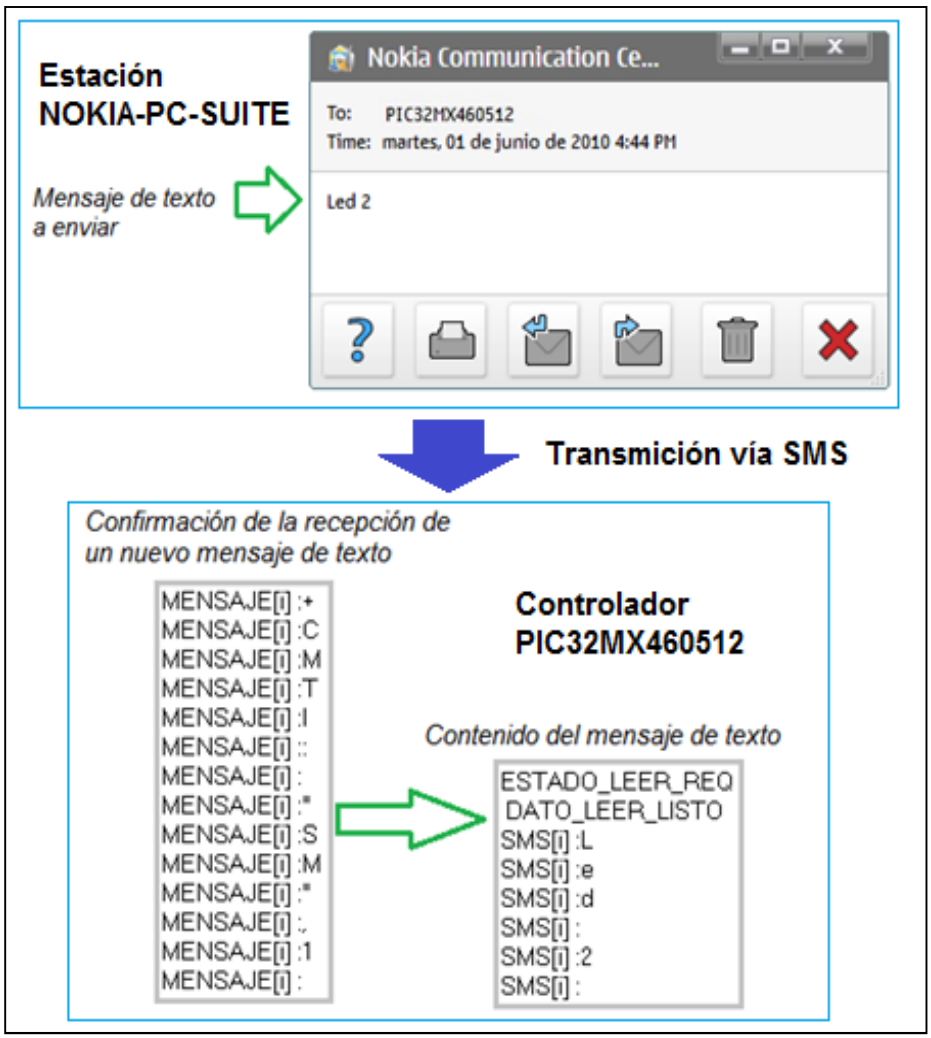


Figura 52. Mensaje de prueba de enlace hacia el nodo CRTecMote desde una terminal móvil

Una vez que se recibió el mensaje en el nodo CRTecMote, se comprobó si se podía generar comunicación desde el nodo CRTecMote hacia el nodo de fusión de datos. Para ello se envió un conjunto de mensajes de prueba con el esquema que muestra en la figura 53.

En esta prueba se puede apreciar la confirmación de la transmisión del modem EDGE-USB por medio de la cadena de datos CMGS.

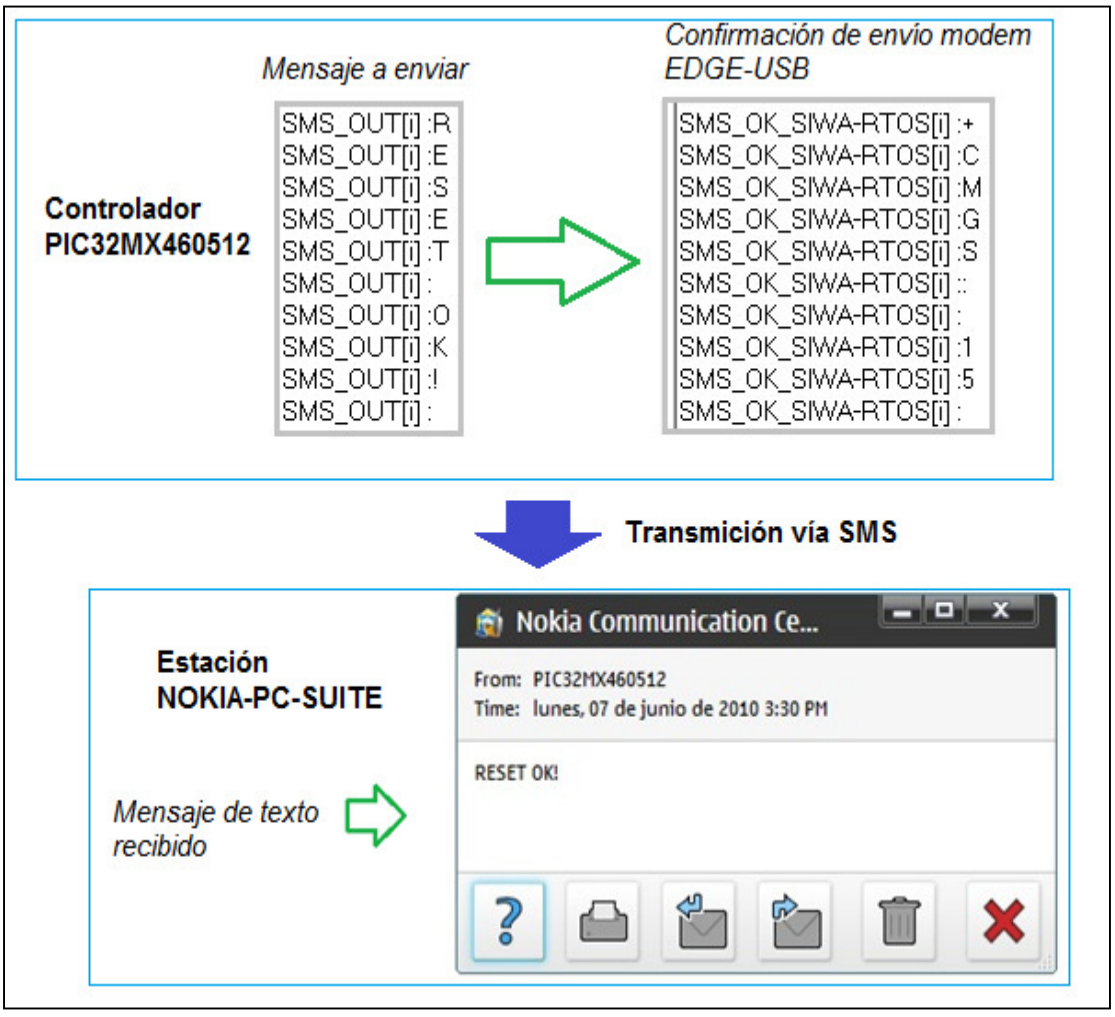


Figura 53. Mensaje de prueba de enlace desde el nodo CRTecMote hacia una terminal móvil

Se puede visualizar tanto la información enviada desde el nodo sumidero, como la información recibida en el teléfono móvil conectado a una PC emulando la terminal de fusión de datos.

Como resultado del diseño del bloque de comunicación, se cuantificó el consumo de potencia, así como la confiabilidad del envío de un conjunto de mensajes, con el propósito de determinar el nivel de confiabilidad del modulo y con diferentes parámetros operativos y con ello escoger el que se ajustara mejor a la red inalámbrica de sensores CRTecMote.

En la figura 54 se presenta la gráfica del estudio del consumo de corriente del controlador en un período de 72 segundos.

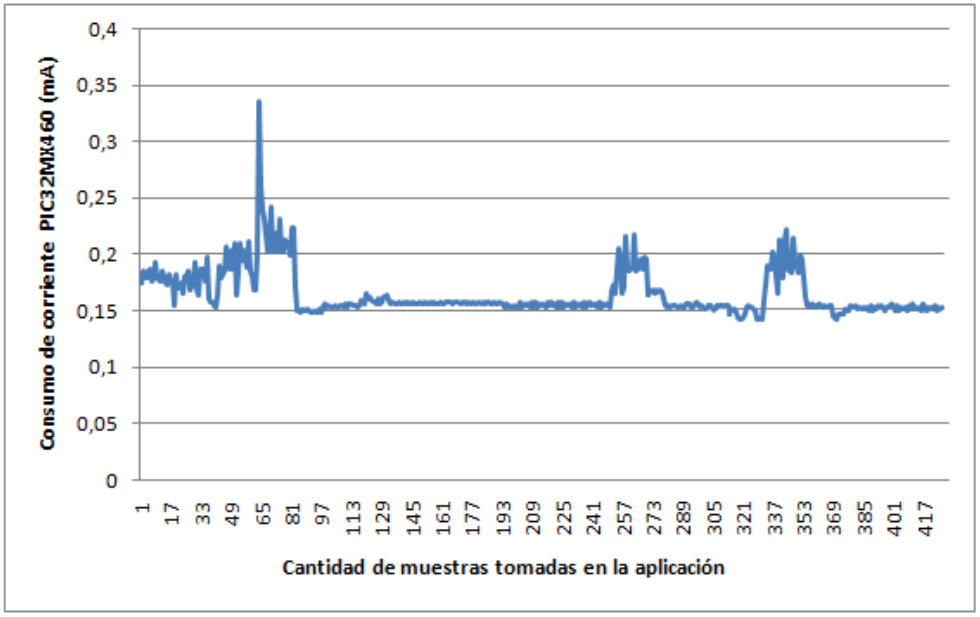


Figura 54. Consumo de corriente de un nodo CRTecMote ejecutando el controlador de acceso a la red de telefonía móvil

El perfil de consumo de corriente de la figura 54 se dividió en 4 subgrupos distintos de acuerdo a la función que se realizaba con el propósito de distinguir dentro de la aplicación cual proceso se estaba dando y con ello caracterizar la el consumo del nodo de acuerdo a la función que estuviese realizando.

Los cuatro procesos que se pueden identificar en la gráfica 54 corresponden a:

- Inicialización de la conexión
- Inicialización de la tarea del SIWA RTOS
- Envío de mensaje SMS
- Recepción de SMS

Tabla 9. Caracterización del consumo de potencia eléctrica de las distintas funciones del controlador de comunicación de área extendida del nodo CRTecMote.

Ejecución	Rango de Muestras (s)	Consumo de corriente promedio (mA)	Consumo de Potencia promedio (mW)
1. Conexión ICE	0.00-11.7	250	100
2. USB Tasks en SIWA RTOS	11.7-43.5	150	324
3. Nuevo SMS	43.48.15-	200	43.0
4. Salida SMS	57.5-62.2	200	42.0

Luego de caracterizar el consumo de potencia del nodo CRTecMote ejecutando el controlador de software con SIWARTOS, se procedió a determinar el periodo óptimo entre envío de mensajes de manera que se tuviese menos porcentaje de error. El tamaño de los mensajes era de 160 bytes y se varió el tiempo entre envíos de los mensajes, en este caso se utilizó un conjunto de 5 mensajes.

La tabla 10 muestra los resultados obtenidos para la prueba mencionada antes.

Tabla 10. Caracterización de la confiabilidad de la transmisión de los nodos CRTecMote

Tiempo entre envío de mensajes [s]	Cantidad de mensajes enviados	Cantidad de mensajes recibidos	% error
3	5	0	100%
5	5	3	40%
7	5	4	20%
8	5	5	0%

Otros controladores de software implementados

Además de los controladores mencionados en las secciones anteriores, durante el desarrollo del nodo CRTecMote se desarrollaron controladores que se habilitaron dentro del SIWA-RTOS con el propósito de:

- Administración de los controladores analógico a digital, permitiendo la selección de canales, el ajuste de la resolución y los tiempos de muestro entro otros.
- Incorporación de dispositivos de actuación por medio de la técnica de PWM con el cual se controlaba la velocidad de un ventilador.
- Incorporación de un controlador de software para administrar la interfaz de un nodo sumidero capaz de conectarse a Ethernet, con el cual pudiesen extraerse datos desde la red de sensores y enviarlo a través de internet. Además del controlador se implementó un servidor empotrado de páginas web, para que los usuarios pudieran interactuar con la red.
- Se implementaron controladores para interfaces de comunicación por el puerto serie mediante el protocolo RS232C. Esta interfaz se utilizó para conexión con equipos de depuración y prueba de la red inalámbrica de sensores.
- El protocolo USB fue implementado para conectar los nodos sumidero con una computadora portátil y extraer información de la red en tiempo real, así como su configuración.
- Un protocolo para el manejo de sistemas de archivos en una unidad de almacenamiento masivo. Con este se puede almacenar localmente una copia de los datos de forma que si pierde conectividad la información no se pierda y se pueda extraer tanto físicamente como por medio de la Web.

Todos los controladores desarrollados quedan a disposición de los desarrolladores de nuevas aplicaciones, de manera que la curva de desarrollo de sistemas de recolección de datos basados en redes inalámbricas CRTecMote se minimice y con ello el tiempo de desarrollo se explote en la integración de nuevas funcionalidades para la plataforma.

Diseño del sistema de fusión de datos para la red CRTecMote

El tercer bloque fundamental que se incluye en el diseño de la red inalámbrica CRTecMote fue el sistema de fusión de datos. Este sistema se implementó en una computadora personal o PC a la cual se le adaptó uno de los MODEM USB que se describió en la sección de conexión de área extendida. A este sistema se le denominó "LOADPOINT" en alusión a que es el punto de recolección, organización y fusión de los datos recolectados desde la red inalámbrica de sensores CRTecMote.

El sistema de organización de la información y fusión de datos se ejecuta en un computador que contiene una base de datos para almacenar la información y un software de control de un dispositivo inalámbrico para la conexión al sistema global de comunicaciones móviles. Dicho software es la interfaz para obtener los datos del dispositivo GSM y almacenar en la base de datos "mySQL". La figura 55 detalla la arquitectura del sistema implementado para organizar y almacenar la información proveniente de la red de sensores CRTecMote.

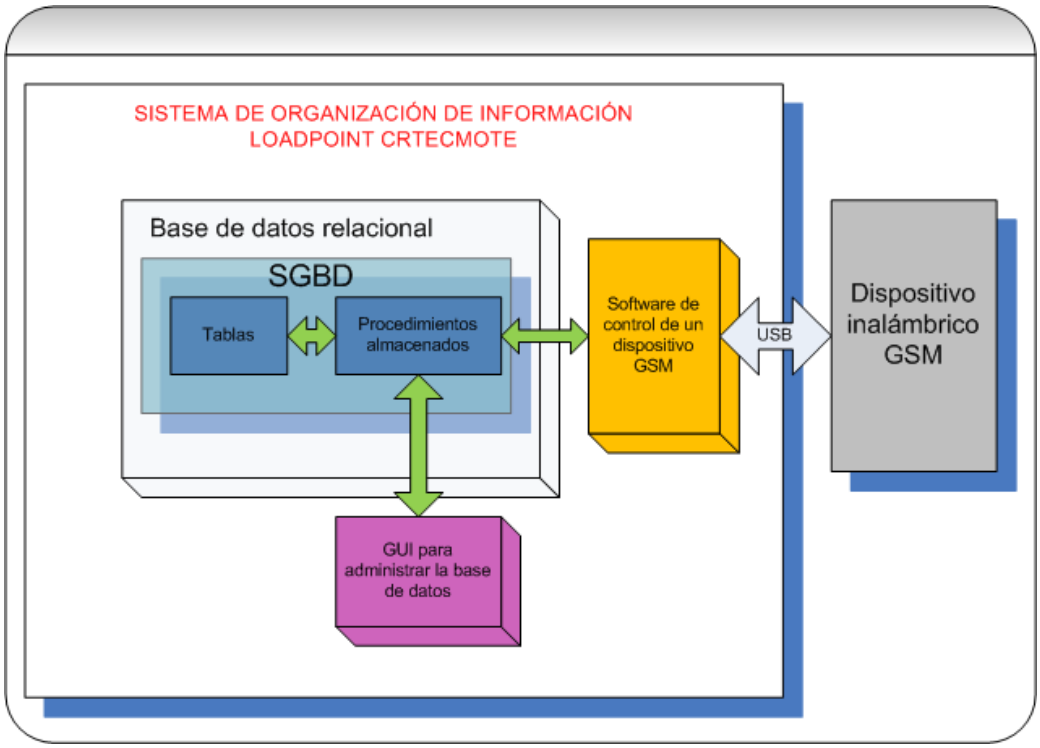


Figura 55. Arquitectura del sistema de recolección, procesamiento y almacenamiento de datos de la red inalámbrica de sensores CRTecMote.

La base de datos implementada sigue el modelo relacional para almacenar información y se compone de un sistema gestor de bases de datos (SGBD) donde se construyeron las tablas para organizar la información y los procedimientos almacenados en el SGBD para realizar tareas de administración de las tablas y su contenido.

Para verificar la estructura de la base de datos relacional se realizó una interfaz gráfica de usuario (GUI) en el lenguaje de alto nivel C#. Con éste mismo lenguaje se implementó las rutinas para el programa que controla el hardware de acceso a la red GSM. El dispositivo GSM se conectó al sistema por un puerto del computador que soporta el protocolo del bus serial universal (USB).

El diseño para la organización de la información proveniente de una red inalámbrica de sensores consiste en la aplicación del modelo relacional de una base de datos. Para aplicar el modelo, se procedió al establecimiento de grupos de datos de un mismo tipo o contexto mediante tablas. Las tablas son llamadas "tuplas" y están constituidas por "filas" (entradas) y columnas. Cada fila tiene una o más columnas. Los dos tipos de tuplas implementadas son basadas en el esquema de "padre" e "hijo". Una tabla "padre" contiene un grupo de datos del mismo tipo de la cual una tabla "hijo" necesita una o más filas. Una tabla "hijo" se convierte en una tabla "padre" a su vez, cuando otra tabla hace referencia a alguna de sus filas.

Las relaciones de las tablas unas con otras se manejó bajo el esquema de uniones lógicas llamadas "llaves". Cada tupla tiene una "llave primaria" (primary key) y es única. Ésta llave es la identificación de la tabla. Una tupla "padre" pura solamente tiene llave primaria. Una tabla "hijo", además de la llave primaria, tiene llaves secundarias llamadas "llaves extranjeras" (foreign keys) y son utilizadas para apuntar de manera lógica a las llaves primarias de las tablas tipo padre. El diseño del modelo descrito se muestra en la figura 56.

El esquema de la figura 56 muestra en detalle las tablas creadas para el modelo de bases de datos relacional en el sistema de organización de la información generada por una red inalámbrica de sensores CRTecMote.

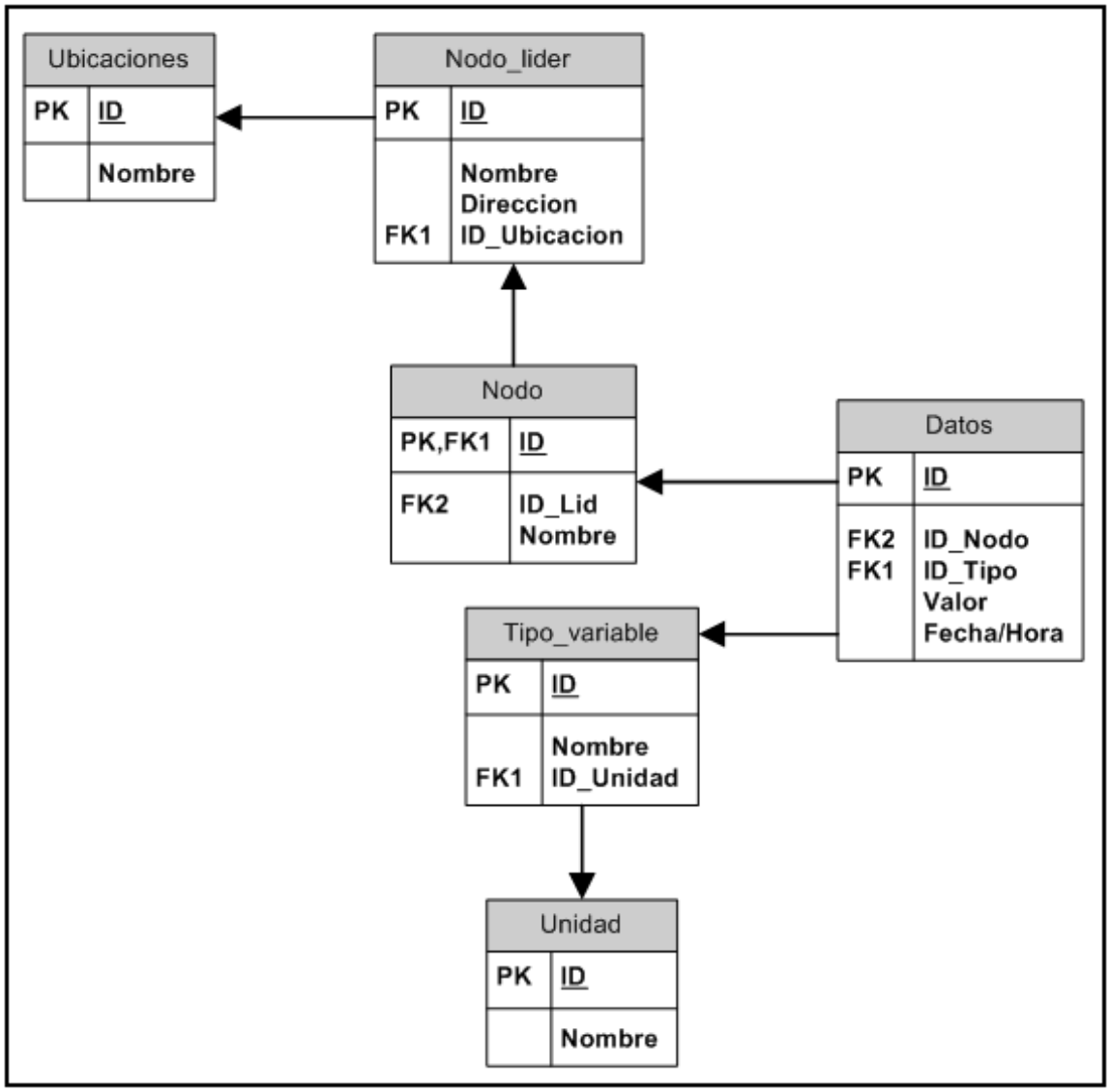


Figura 56. Esquema relacional de tablas creadas para el almacenamiento de la información proveniente de la red inalámbrica CRTecMote.

Según la figura 56, las tablas de tipo padre puras son “Ubicaciones” y “Unidad”. La primera entidad tiene una única columna que se encarga de almacenar la ubicación de un nodo de la red. Cuando una persona se encargue de distribuir los nodos sensores de la red inalámbrica es imperativo que anote la ubicación de dicho nodo para que sea ingresado en esta tabla. Nótese que es una tabla padre pura debido a la ausencia de llaves extranjeras. De igual manera, la segunda entidad se encarga de guardar la información correspondiente a las unidades de

los parámetros que se desean medir con los nodos sensores; sea el caso de un dato de temperatura en la que su unidad en el sistema internacional de unidades es el “Celsius”.

La tupla “Nodo_líder” o nodo sumidero son del tipo hijo ya que apunta a su padre que es “Ubicaciones”. La función de dicha tabla es almacenar el nombre de los nodos tipo líder en la red inalámbrica y la dirección de dicho nodo para ubicar el área donde se hizo la distribución de los nodos sensores bajo el mando del nodo líder. Una llave extrajera en esta tabla apunta a las ubicaciones de cada uno de los nodos de la red.

La tabla “Nodo” es una tabla hijo de la tabla “Nodo_líder” y en su defecto de “Ubicaciones” como se observa en la figura 56, posee una llave extranjera que apunta a su tabla padre directo pero también hereda un apuntador implícito a la tabla de “Ubicaciones”. Esta tabla se encarga de guardar los nombres de los nodos distribuidos de la red. Con la llave extranjera se hacen las relaciones de la información correspondiente a un nodo sensor en específico.

La entidad llamada “Tipo_variable” es un hijo de “Unidad” y se encarga de depositar la información referente al tipo de medición que hace un sensor. Sus columnas corresponden al nombre del tipo de medición que se está haciendo (temperatura, humedad, acidez entre muchas más) y a la llave extranjera que apunta a su unidad correspondiente en la tabla padre.

Finalmente, la tupla llamada “Datos” es una tabla “hijo” pura que almacena directamente la información proveniente de la red inalámbrica de sensores mediante sus columnas del valor medido, la fecha/hora en que se midió dicho valor, el número de identificación del nodo que hizo la medición y el número de identificación del tipo de variable. Estos últimos datos corresponden a las llaves extranjeras para relacionar el dato entrante con la información contenida en las tablas descritas anteriormente.

Para la implementación de una base de datos relacional se escogió un sistema gestor de bases de datos MySQL que utiliza el motor de almacenamiento InnoDB.

Para poner en marcha el sistema gestor de bases de datos relacional MySQL fue necesario conocer el hardware y el sistema operativo en el cual se deseaba ejecutar el sistema gestor de la base de datos. En el caso del presente proyecto se utilizó una computadora personal HP Compaq nc6400 con procesador Intel Core Centrino® Duo con 2GB de memoria RAM y 2 GHz de velocidad. El sistema operativo es el ambiente Windows Vista® de 32 bits. Así el SGBD utilizado es la versión MySQL 5.0 para Windows®

La instalación de MySQL 5.0 para Windows® se realizó mediante una interfaz gráfica para su configuración. MySQL presenta un archivo ejecutable para levantar dicha interfaz y empezar con la instalación del SGBD. Para lograr el funcionamiento del gestor de bases de datos, en primera instancia por medio de la interfaz, se procedió a la instalación del sistema gestor de manera manual de tal forma que se pudiera especificar las características más importantes para el funcionamiento deseado de la base de datos. En las opciones de instalación se decidió agregar además del SGBD, la ventana de comandos y la documentación de ayuda para los comandos. Con las opciones listas se procedió a la instalación.

Una vez instalado, se continuó con la configuración del SGBD con la ayuda de la interfaz gráfica. Se escogió hacer una configuración detallada con el fin de definir el tipo de gestor necesario. Para el proyecto se utilizó un gestor de base de datos para el desarrollo de código como muestra la figura 57.

Figura 57. Pantalla de instalación del gestor de bases de datos utilizado en LOADPINT

Para crear una tabla bajo el modelo de base de datos relacional se diseñó el algoritmo de la figura 58.

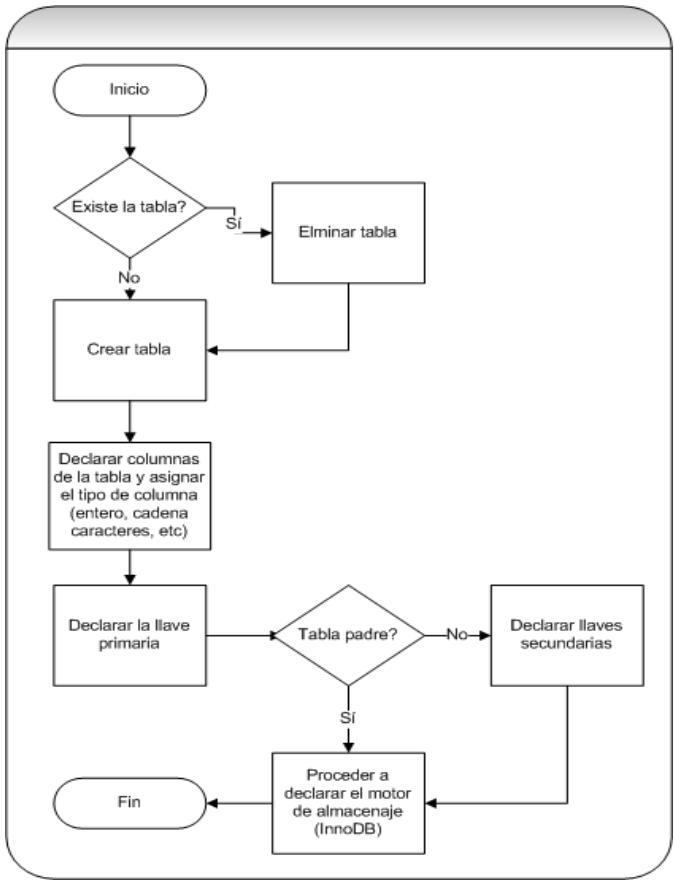


Figura 58. Procedimiento para crear una tabla bajo el modelo de base de datos relacional

Para implementar cada una de las tablas del modelo de la figura 56 se utilizó el lenguaje estructurado de consultas (SQL) en el SGBD siguiendo la secuencia de pasos descritos en el diagrama de flujo de figura 58.

Una vez creada la base de datos, se requería definir el formato de envió de la información desde el nodo sumidero o nodo líder de la red inalámbrica de sensores CRTecMote, por lo que se procedió a diseñar un protocolo de comunicación entre el nodo sumidero y la estación LOADPOINT de forma que se suministrara la información con el nivel de integridad adecuado.

El diseño del protocolo de comunicaciones se basó en la necesidad de establecer la forma en que los datos iban a ser recibidos y comprendidos por el sistema de organización de la información (LOADPOINT). Debido a que los nodos sensores

en una red inalámbrica comparten información fue necesario especificar el tipo de dato que va a recibir el sistema de organización de la información. Para tipificar el tipo de dato se estructuró la forma que debe de tener un paquete para que la estación LOADPOINT pueda entenderlo y poder así darle tratamiento para que sea almacenado en la base de datos correctamente. Debido a que la información es estructurada de una manera definida, al momento de leer la estructura para su comprensión en el nodo de carga se consideró la integridad de los datos.

Para definir la estructura de los datos en el modelo relacional se consideró que los datos medidos desde un nodo de medición (RFD) se colocarían en el campo de "valor". Dicho valor puede corresponder a múltiples tipos de variables dependiendo del ambiente en que se encuentre ubicada la red de sensores y el propósito de dicho sistema. La figura 59 muestra el formato de la trama de datos que se diseñó para hacer consistente la comunicación entre los nodos sumidero y la estación de carga de datos (LOADPOINT)

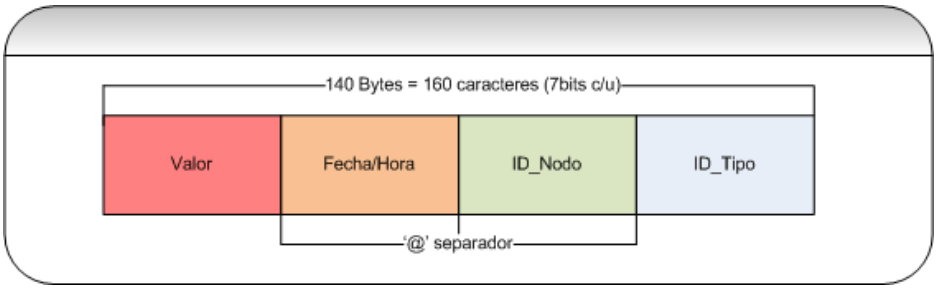


Figura 59. Formato de la trama de datos transmitida entre los nodos sumidero y la estación de carga de datos de la red inalámbrica CRTecMote

Bajo el principio de funcionamiento de que la red inalámbrica de sensores se planea para su utilización en sistemas de monitorización ambiental, a partir de ello se definió que la estructura posible de la información proveniente de un nodo de la red puede ser un dato correspondiente a un valor, la fecha y hora en que se mide dicho valor, un identificador del nodo sensor y finalmente el identificador del tipo de dato (temperatura, humedad, presión, fuerza, etc.) que se está midiendo del ambiente. De manera conjunta todos estos datos forman un paquete de datos como se muestra en la figura 59. Se denota que el paquete de datos o trama tiene

un máximo de 140 bytes de tamaño, lo que significa que la información enviada debe de tener un máximo de 160 caracteres.

De éstos 160 caracteres se deben de restar al menos 3 caracteres que corresponden al separador (se definió el separador '@') de cada uno de los datos enviados en una trama. La cantidad de bytes enviados puede ser variable pero debe de ser menor a los 140 bytes. Los separadores se utilizan para conocer el final de un dato y el inicio de otro en la trama de información. Estos separadores son usados en una rutina para leer dicha información.

Para garantizar la integridad de los datos se definió el uso de un código para detección de errores. El algoritmo que se utilizó es el cálculo de un código "Hash" mediante una biblioteca contenida en el lenguaje de programación C#.

Dicho algoritmo se ejecuta para la cadena de texto o trama definida en la figura 59 y el resultado es un código hash. El código se agrega al paquete de datos definido (encapsulamiento) mediante un nuevo separador escogido como el caracter "^". La Fig. 5.13 muestra el encapsulamiento del paquete de datos con el hash. El código es des-encapsulado en el nodo central y se ejecuta el algoritmo hash sobre el paquete de datos original nuevamente para verificar que este no se encuentra corrupto.

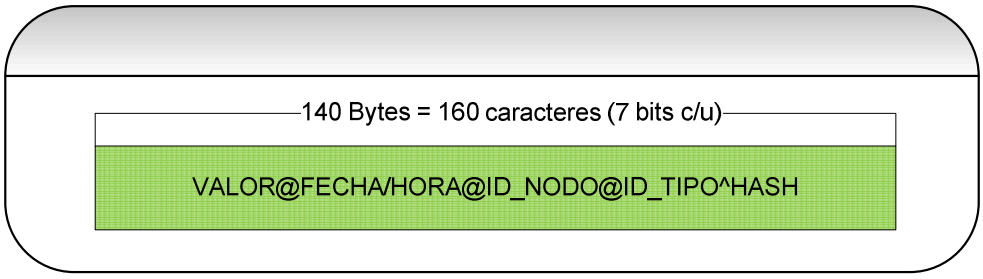


Figura 60. Formato de la trama de datos transmitida entre los nodos sumidero y la estación de carga de datos de la red inalámbrica CRTecMote con el esquema de verificación de integridad incorporado

Una de las piezas claves del sistema de organización de la información generada por una red inalámbrica de sensores es la capacidad de conexión al medio inalámbrico. Se escogió el envío de información utilizando la plataforma de

mensajes cortos del sistema global para comunicaciones móviles. GSM tiene la facilidad de que es la tecnología escogida por el país para telefonía móvil y tiene la particularidad de que cubre casi la totalidad de Costa Rica, lo que permite la escalabilidad de cualquier sistema que utilice el servicio. Para conectar el sistema de organización de información al servicio inalámbrico se escribió una biblioteca de software de un dispositivo funcional sobre la red GSM.

Para inicializar el dispositivo móvil GSM se requirió el uso de los comandos AT para verificar la operación así como establecer la conexión con el dispositivo móvil especificando los parámetros para una conexión serial.

Una vez que el registro de modo de los mensajes se cambió se procedió a configurar en el dispositivo el registro de indicación de nuevo mensaje CNMI (new message indication command), para ello se escoge en tres el valor de este comando para que todo mensaje nuevo sea indicado por medio de una notificación al equipo terminal, luego se indica con un uno el campo para que el aviso tenga la posición en memoria del mensaje en el dispositivo GSM y el resto de los campos se ponen por defecto.

Para realizar la acción del envío de información utilizando el servicio de mensajes cortos se introdujo al dispositivo GSM el comando “AT+CMGS= #destino”, inmediatamente después se envió la información en forma de texto finalizando con el código en ASCII para el retorno de carro. Si el mensaje se envió con éxito la rutina devuelve un uno, de lo contrario la rutina devuelve un cero. Esta es la rutina utilizada para simular el envío de datos desde un nodo líder al nodo central.

Luego se procedió a implementar la rutina de recepción de tramas desde un nodo CRTecMote sumidero. Esta es la rutina más importante del sistema ya que es la que controla la recepción de información del hardware de acceso al medio GSM para el sistema de organización de información. En esta rutina se concentra la comunicación con el dispositivo GSM en el nodo central para la correcta recepción de la información enviada por la rutina anterior. Se diseñó bajo el principio de funcionamiento de que el dispositivo móvil se inicializa para notificar sobre el arribo de un nuevo mensaje.

La figura 61 muestra el procedimiento implementado para la recepción del paquete de datos.

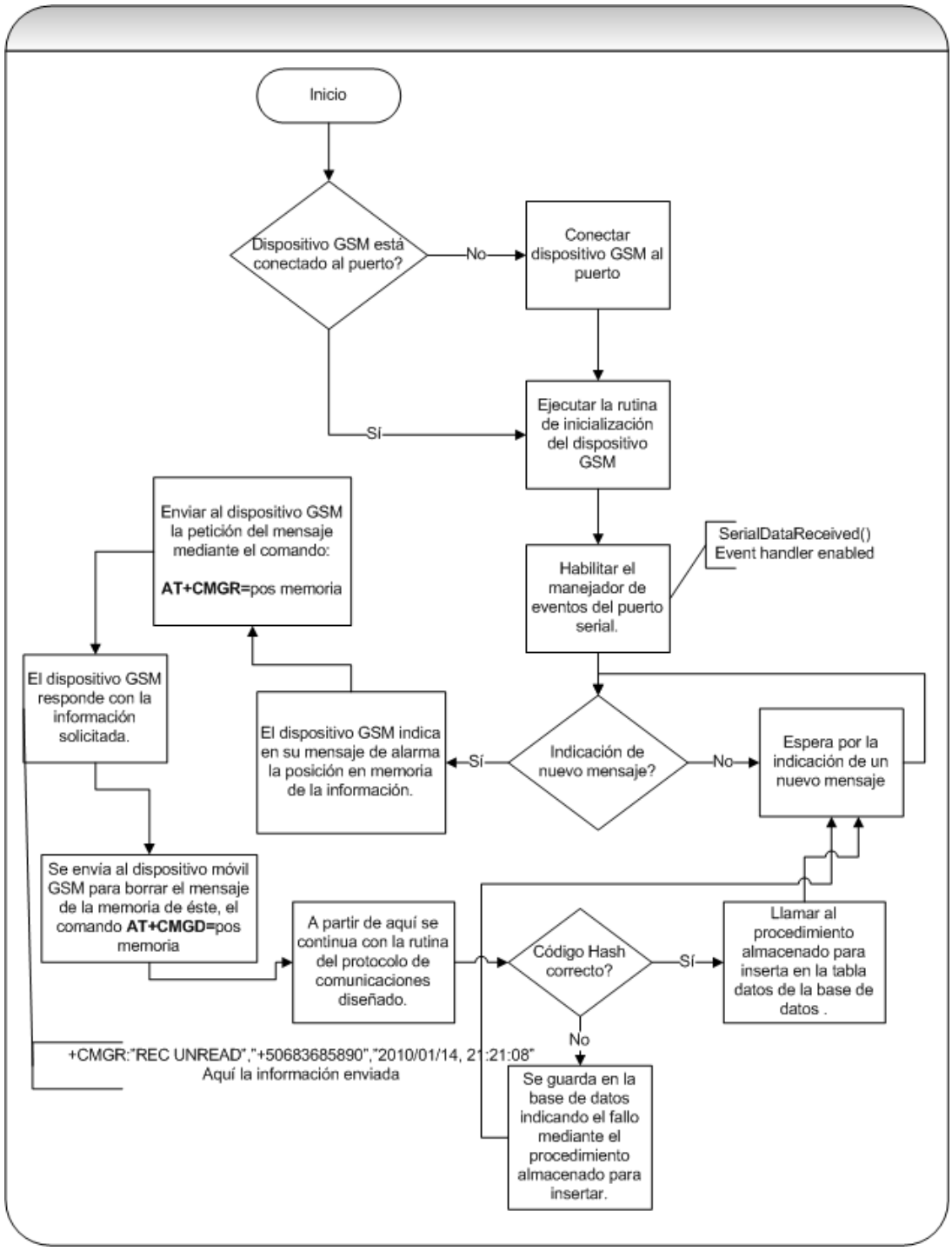


Figura 61. Procedimiento implementado para la recepción de una trama de datos desde un nodo sumidero CRTecMote

Con un manejador de eventos habilitado en el puerto serial se lee la notificación que indica la posición en memoria de la nueva información. Entonces, se envió al modem GSM el comando “AT+CMGR=pos memoria” para leer de memoria el nuevo mensaje. El dispositivo respondió con el mensaje y es leído por la rutina. Luego es almacenado temporalmente en una lista enlazada. Se envió al modem el comando “AT+CMGD=pos memoria” para borrar el mensaje almacenado en el dispositivo y así liberar el espacio de la memoria cuando se dé un funcionamiento crítico del sistema. De ésta forma se aplican las condiciones del protocolo de comunicaciones para separar e interpretar la información recibida, calculando el código hash para validar la integridad de los datos. Si los datos se validan correctamente, se procede a almacenar la información en la base de datos llamando al procedimiento almacenado de insertar del SGBD en la tupla “Datos” del modelo relacional. En su defecto se guarda indicando el fallo en la integridad de los datos.

Para demostrar el funcionamiento del sistema de organización de la información diseñado, se realizó diversas pruebas de campo. Para lograr la realización de las pruebas de funcionamiento se debió escribir código adicional a las rutinas explicadas en el capítulo para tener la facilidad de una interfaz gráfica de usuario.

Una vez implementadas las rutinas de recepción y almacenamiento de datos se procedió a probar la implementación del sistema de organización y fusión de datos LOADPOINT.

Para demostrar que la estructura del modelo de la base de datos relacional diseñada es correcta se creó la interfaz gráfica de la figura 62. La interfaz gráfica hace uso de un conector ODBC desarrollado por MySQL para poder acceder a la base de datos relacional. Este conector funciona de interfaz entre la aplicación y el sistema gestor de bases de datos para poder enviar comandos SQL al SGBD. El conector se utilizó en el presente proyecto como administrador del SGBD MySQL ya que se pueden tener varios SGBD y muchas bases de datos en cada sistema gestor. La aplicación se escribió en lenguaje de alto nivel C# utilizando las bibliotecas disponibles en .NET Framework.

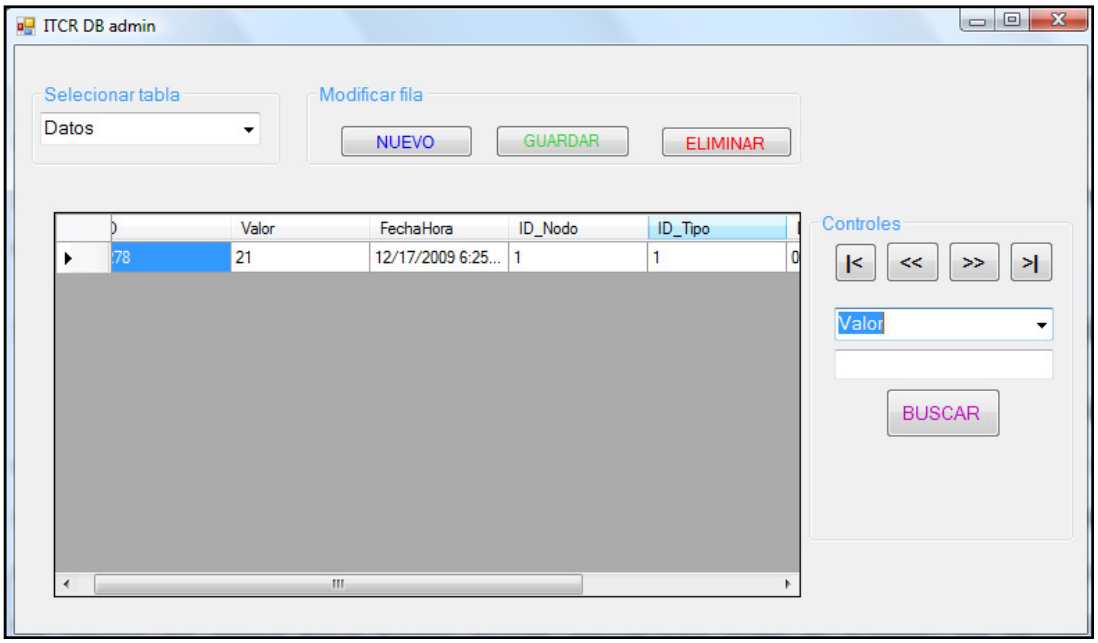


Figura 62. Interfaz de su usuario implementada para visualizar el contenido de la base de datos del sistema de fusión de datos LOADPOINT.

Como se muestra en la figura 62, con esta aplicación es posible ver cada una de las tablas diseñadas en el modelo relacional seleccionando el nombre de cada tabla en el menú “Seleccionar tabla”. En cada tabla se pueden realizar tres funciones básicas expuestas en el menú “Modificar filas” que utilizan los procedimientos almacenados en el SGBD MySQL diseñados para administrar el modelo. Se puede insertar un nuevo registro o fila, guardar o actualizar una fila o columna de una fila y eliminar una fila seleccionada. Los “Controles” son para navegar por las filas de cada una de las tablas. Finalmente, se utilizan algunos procedimientos almacenados para hacer búsquedas en cada una de las tablas.

Se utilizó la interfaz anterior para comprobar el diseño del modelo relacional y su característica de protección de datos mediante los siguientes casos de prueba:

- Dependencia de la tabla “Datos” con “Nodo” y “Tipo_variable”: insertar en la tabla “Datos” un dato con la llave ID_Nodo (llave que apunta a alguna llave primaria de la tabla “Nodo”) y la llave ID_Tipo (llave que apunta a alguna llave primaria de la tupla “Tipo_variable”) cuando no existe la llave primaria correspondiente en la tabla “Nodo” y/o la tabla “Tipo_variable”.

- Dependencia de la tabla “Nodo” con “Nodo_líder”: insertar en la tabla “Nodo” el nombre de un nodo y la llave extranjera ó ID_Lid sin que exista el nodo líder correspondiente.
- Dependencia de la tabla “Nodo_líder” o sumidero con “Ubicaciones”: insertar en “Nodo_líder” el nombre del nodo líder y su dirección sin tener el ID_Ubic que ubica el nodo sensor al cual el nodo líder gobierna.
- Dependencia de la tabla “Tipo_variable” con “Unidad”: insertar en la tabla “Tipo_variable” un tipo de dato (temperatura, presión, humedad, etc.) apuntando con el ID_Uni la unidad del tipo de dato sin haber adicionado previamente su unidad en la tabla “Unidad”.
- Llaves primarias duplicadas: Insertar en cualquier tabla una llave primaria o ID duplicado.
- Independencia de la llave primaria en “Ubicaciones”: borrar en la tabla “Ubicaciones” un ID o llave primaria en la cual una llave secundaria de la tabla “Nodo_lider” esté utilizando.
- Independencia de la llave primaria en “Nodo_líder”: borrar en la tabla “Nodo_líder” un ID a la cual una llave secundaria de la tabla “Nodo” esté utilizando.
- Independencia de la llave primaria en “Nodo”: borrar en la tabla “Nodo” un ID a la cual la llave extranjera de la tabla “Datos” esté apuntando.
- Independencia de la llave primaria en “Unidad”: borrar en la tupla “Unidad” un ID a la cual la llave secundaria de la tabla “Tipo_variable” esté apuntado.
- Independencia de la llave primaria en “Tipo_variable”: borrar en la tabla “Tipo_variable” una llave primaria a la cual una llave extranjera de la tabla “Datos” esté apuntando.

Para verificar la correcta creación de la estructura del modelo de base de datos se verificó que el proceso de la figura 63 se cumpliera basándose en la definición de los casos mencionados anteriormente.

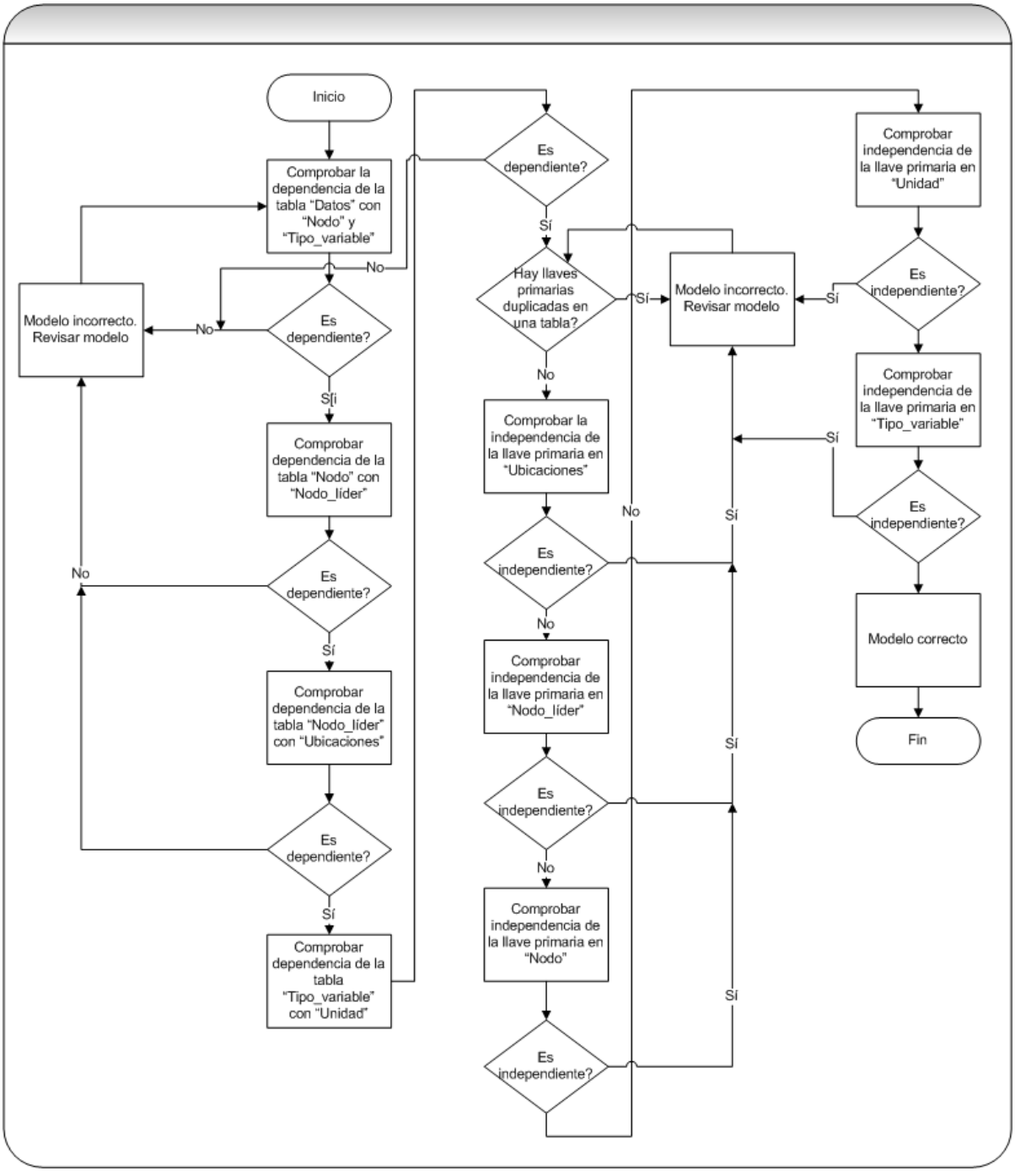


Figura 63. Diagrama de flujo para verificar la estructura del modelo de base de datos relacional de LOADPOINT

Para comprobar el funcionamiento del sistema de organización de la información ante un envío significativo de mensajes se diseñó una interfaz gráfica capaz de enviar una seguidilla de mensajes utilizando el software controlador del hardware

de acceso al medio inalámbrico GSM. La figura 64 muestra la interfaz de usuario gráfica diseñada que utiliza el software controlador del dispositivo inalámbrico para enviar información. Esta interfaz se puede ejecutar en cualquier computador personal que tenga .Net Framework instalado. Este sistema se implementó para simular la carga de trabajo variable de la red inalámbrica de sensores CRTecMote. Este programa simula un nodo sumidero o nodo líder enviando tramas a LOADPOINT para su procesamiento.

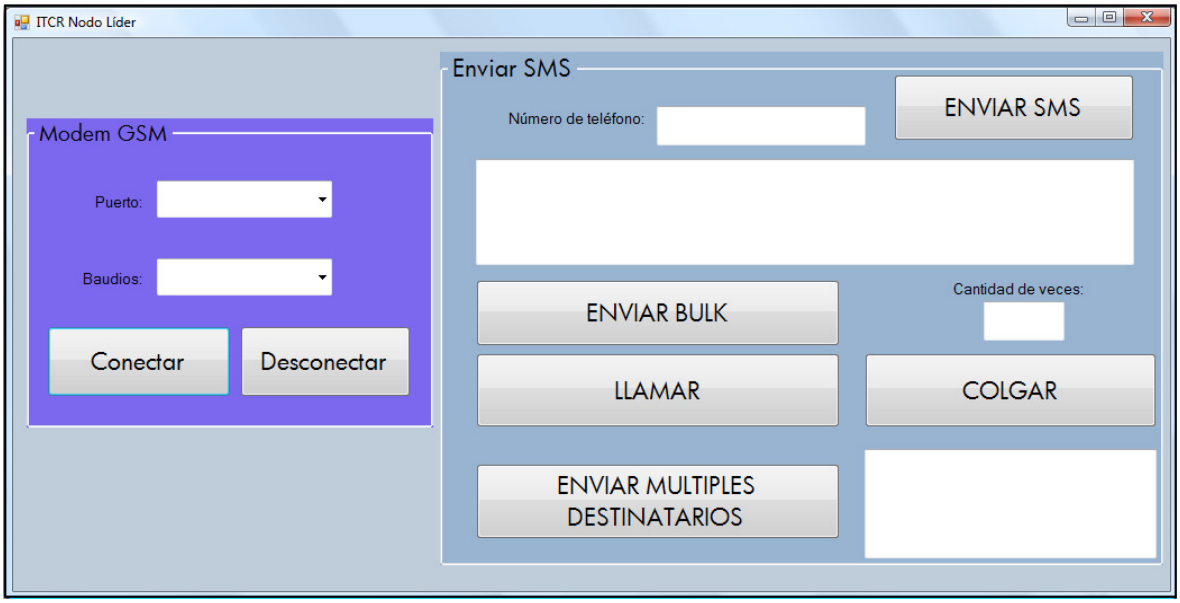


Figura 64. Interfaz de usuario diseñada para emular el envío de conjuntos de datos desde un nodo sumidero o líder CRTecMote al sistema de fusión de datos LOADPOINT

A la rutina de control del dispositivo GSM se le agregó una modificación para el envío de mensajes en forma masiva ("bulk"). Con propósito de determinar la capacidad del sistema a aplicaciones de tiempo real.

Para probar la funcionalidad de LOADPOINT se midió la cantidad de mensajes enviados y la cantidad de mensajes recibidos para obtener el éxito del sistema ante una gran carga de transmisión.

Para comprobar la integridad de la información se utilizó las definiciones del protocolo de comunicaciones. Antes de enviar un dato se calculó el código hash de la información y se encapsuló el código en el mensaje. Cuando el mensaje se recibe en el nodo central el hash se des-encapsula y se vuelve a calcular el hash

sobre el mensaje recibido. Ambos códigos se comparan para determinar si son iguales y validar la integridad de los datos.

Para esta prueba se agregó una columna a la tupla “Datos” llamada “Hash” para almacenar el resultado de la integridad de la información. Si el hash recibido y el calculado son iguales se valida el dato como “correcto” y se guarda en la base de datos esta palabra en la nueva columna de la tabla. Si es “incorrecto” se guarda el dato en la base de datos con dicha palabra en la nueva columna.

Se diseñó una prueba para medir la diferencia de tiempo que existe entre el momento en que se envía un dato y el momento en que se recibe el dato en el sistema LOADPOINT.

Utilizando la interfaz de envío de mensajes para la simulación del nodo líder se toma el tiempo del sistema computacional (tiempo inicial) en el momento justo antes de enviar el mensaje utilizando el controlador del dispositivo inalámbrico GSM. Este tiempo es encapsulado al mensaje total mediante una rutina que adiciona el tiempo de retraso al encapsulamiento del código hash utilizando el separador “^”. Cuando el mensaje llega al sistema de organización de información se toma el tiempo del sistema computacional como el tiempo final, se separa el tiempo inicial de la trama de datos.

Se adicionó para esta prueba una columna en la tupla de “Datos” llamada “Delta” en el modelo relacional con el fin de almacenar la duración de envío y recepción de la información.

En las tablas 11, 12 y 13 se muestran los resultados obtenidos para el envío de conjuntos de datos masivos (“bulk”) para diferentes horas del día con el propósito de verificar una operación estable del sistema LOADPOINT a lo largo del día.

Para cada conjunto de mensajes se calculó el tiempo de duración desde que se envió el mensaje del nodo sumidero, hasta que se procesó su recepción en el sistema LOADPOINT.

A continuación se muestran los resultados obtenidos para cada uno de los tres escenarios de operación planteados.

Tabla 11. Tiempo de retardo y resultado de la comparación del hash de 50 paquetes de información enviados el 4 de Enero del 2010 a las 6:30 am.

Trama	Tiempo de retardo (s)	Resultado del hash
1	70.5	CORRECTO
2	71.3	CORRECTO
3	71.1	CORRECTO
4	69.8	CORRECTO
5	73.1	CORRECTO
6	73.7	CORRECTO
7	72.1	CORRECTO
8	73.7	CORRECTO
9	71.1	CORRECTO
10	69.5	CORRECTO
11	72.1	CORRECTO
12	70.2	CORRECTO
13	69.7	CORRECTO
14	73.3	CORRECTO
15	69	CORRECTO
16	73.3	CORRECTO
17	71.5	CORRECTO
18	72.8	CORRECTO
19	73.8	CORRECTO
20	71.8	CORRECTO
21	73.7	CORRECTO

22	74	CORRECTO
23	73.7	CORRECTO
24	74	CORRECTO
25	73.8	CORRECTO
26	73.7	CORRECTO
27	72.2	CORRECTO
28	71.3	CORRECTO
29	71.1	CORRECTO
30	70.9	CORRECTO
31	72.6	CORRECTO
32	73.5	CORRECTO
33	73.1	CORRECTO
34	71.5	CORRECTO
35	72.9	CORRECTO
36	73.7	CORRECTO
37	74.5	CORRECTO
38	73.7	CORRECTO
39	71.6	CORRECTO
40	73.8	CORRECTO
41	73.7	CORRECTO
42	72.5	CORRECTO
43	73.7	CORRECTO
44	71.4	CORRECTO
45	69.8	CORRECTO

46	73.7	CORRECTO
47	68.9	CORRECTO
48	73.7	CORRECTO
49	69.7	CORRECTO
50	69	CORRECTO

Tabla 12. Tiempo de retardo y resultado de la comparación del hash de 50 paquetes de información enviados el 04 de Enero del 2010 a las 12:10 pm.

Trama	Tiempo de retardo (s)	Resultado del hash
1	70.5	CORRECTO
2	71.3	CORRECTO
3	71.1	CORRECTO
4	69.1	CORRECTO
5	69.8	CORRECTO
6	71.1	CORRECTO
7	72.1	CORRECTO
8	70.9	CORRECTO
9	71.1	CORRECTO
10	69.5	CORRECTO
11	72.1	CORRECTO
12	70.2	CORRECTO
13	69.7	CORRECTO
14	69.4	CORRECTO
15	69	CORRECTO

16	72	CORRECTO
17	70.5	CORRECTO
18	72.1	CORRECTO
19	71.4	CORRECTO
20	71.8	CORRECTO
21	70.8	CORRECTO
22	71.2	CORRECTO
23	71.5	CORRECTO
24	72.1	CORRECTO
25	70.8	CORRECTO
26	70.37	CORRECTO
27	70.7	CORRECTO
28	71.1	CORRECTO
29	70.9	CORRECTO
30	70.9	CORRECTO
31	72.6	CORRECTO
32	72.2	CORRECTO
33	72.1	CORRECTO
34	71.5	CORRECTO
35	72	CORRECTO
36	71.8	CORRECTO
37	70.9	CORRECTO
38	70.8	CORRECTO
39	71.6	CORRECTO

40	69.9	CORRECTO
41	70.1	CORRECTO
42	71	CORRECTO
43	71.1	CORRECTO
44	69.8	CORRECTO
45	72.5	CORRECTO
46	72.5	CORRECTO
47	69.1	CORRECTO
48	72.1	CORRECTO
49	70	CORRECTO
50	69	CORRECTO

Tabla 13. Tiempo de retardo y resultado de la comparación del hash de 50 paquetes de información enviados el 4 de Enero del 2010 a las 05:00 pm.

Trama	**Tiempo de retardo (s)**	**Resultado del hash**
1	70.5	CORRECTO
2	71.3	CORRECTO
3	71.1	CORRECTO
4	69.1	CORRECTO
5	69.8	CORRECTO
6	71.1	CORRECTO
7	72.1	CORRECTO
8	70.9	CORRECTO
9	71.1	CORRECTO

10	69.5	CORRECTO
11	72.1	CORRECTO
12	70.2	CORRECTO
13	69.7	CORRECTO
14	69.4	CORRECTO
15	69	CORRECTO
16	72	CORRECTO
17	70.5	CORRECTO
18	72.1	CORRECTO
19	71.4	CORRECTO
20	71.8	CORRECTO
21	70.8	CORRECTO
22	71.2	CORRECTO
23	71.5	CORRECTO
24	72.1	CORRECTO
25	70.8	CORRECTO
26	70.37	CORRECTO
27	70.7	CORRECTO
28	71.1	CORRECTO
29	70.9	CORRECTO
30	70.9	CORRECTO
31	72.6	CORRECTO
32	72.2	CORRECTO
33	72.1	CORRECTO

34	71.5	CORRECTO
35	72	CORRECTO
36	71.8	CORRECTO
37	70.9	CORRECTO
38	70.8	CORRECTO
39	71.6	CORRECTO
40	69.9	CORRECTO
41	70.1	CORRECTO
42	71	CORRECTO
43	71.1	CORRECTO
44	69.8	CORRECTO
45	72.5	CORRECTO
46	72.5	CORRECTO
47	69.1	CORRECTO
48	72.1	CORRECTO
49	70	CORRECTO
50	69	CORRECTO

Los resultados de dichas tablas se procesaran luego para analizar la duración promedio y el porcentaje de integridad de los datos recibidos en LOADPOINT.

Hasta este momento se ha discutido la implementación individual de los módulos que conforman la red inalámbrica CRTecMote, sin embargo en los objetivos del proyecto se plantea poner a disposición de la comunidad científica la tecnología desarrollada para que su utilización en el diseño de sistemas de adquisición de datos ambientales para alguna aplicación específica. La siguiente sección se encarga de mostrar una aplicación específica de la red inalámbrica CRTecMote.

Implementación de prueba piloto de la red CRTecMote

GENFORES es programa que vincula la academia y la industria, creado a finales el año 2001 por el Dr. Olman Murillo Gamboa, profesor en la Escuela de Ingeniería Forestal del Tecnológico de Costa Rica y especialista en Mejoramiento Genético Forestal. GENFORES es la primera cooperativa de mejoramiento forestal en Costa Rica y cuenta con la participación de 11 empresas privadas que aportan recursos económicos y conocimiento, propiciando un intercambio tecnológico muy amplio entre todos los integrantes del grupo. Su misión es promover la exploración, conservación, utilización racional y mejoramiento de los recursos genéticos forestales.

El programa GENFORES tiene operaciones en las instalaciones del Tecnológico de Costa Rica en la sede de Santa Clara, en este lugar se cuenta con una serie de túneles con ambiente controlado, o mini jardines clonales para la siembra de los clones de árboles creados por ellos. Estos espacios deben presentar características climatológicas adecuadas para el óptimo desarrollo de los árboles en su etapa inicial tales como humedad relativa, temperatura y cantidad de luz, entre otras. Algunas de estas condiciones son manipuladas mediante sistemas de riego por micro-aspersión, que se realiza de manera automática mediante temporizadores y sistemas de apertura de válvulas, sin embargo las decisiones de la frecuencia de riego son hechas a partir de la experiencia de los encargados y no de datos científicos.

Actualmente el desempeño logrado en las recámaras con ambientes controlados es satisfactorio ante los requerimientos de GENFORES, sin embargo, no se cuenta con medidas de las características ambientales que permitan cuantificar el desempeño de este con datos recopilados en el lugar, ante esta situación, los encargados de GENFORES acuden al personal de la Escuela de Ingeniería Electrónica, específicamente al Ing. Johan Carvajal, coordinador del proyecto CRTECMOTE para analizar la posibilidad de implementar un sistema que permita tomar mediciones y así cuantificar el desempeño del ambiente de desarrollo de los árboles en su etapa temprana para buscar potenciales mejoras al sistema actual.

Con tal propósito se procedió a implementar una red inalámbrica CRTecMote con el objetivo de cuantificar las variables ambientales de interés del invernadero y a la vez probar la tecnología de datos desarrollada para la red inalámbrica de sensores CRTecMote.

En la figura 65 se puede observar un diagrama de la topología implementada. En esta figura se muestra un nodo central o FFD que se comunica con tres nodos secundarios o RFDs por medio del estándar IEEE 802.15.4, específicamente el protocolo MIWI de comunicación inalámbrica. Cada RFD está conectado a dos grupos de sensores, el primer grupo son los que miden humedad relativa y temperatura que se comunican por medio del protocolo serial I^2C y el segundo grupo son sensores analógicos de luz que se leen por medio del ADC de cada nodo.

En la parte superior de la figura se observa el enlace entre el FFD y una PC, este se realiza por medio de Ethernet.

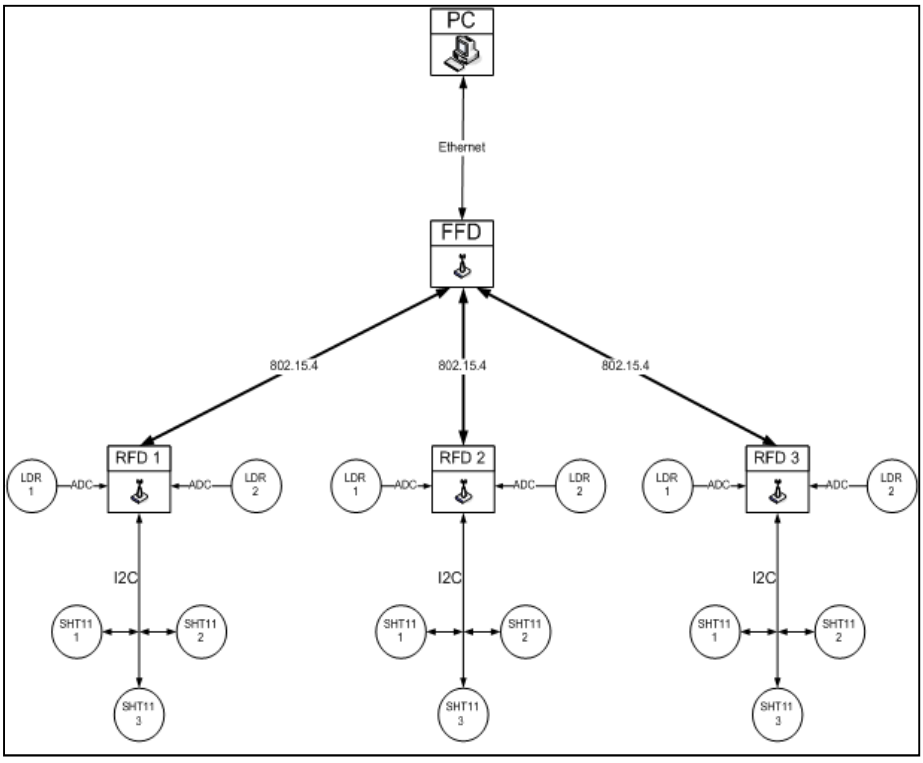


Figura 65. Arquitectura de red utilizada para probar la tecnología CRTecMote en los mini jardines clonales de GENFORES

Entre sensores que se conectaron a los nodos de medición de datos (RFD) para aplicación de caracterización de mini jardines clonales destacan los de temperatura y humedad relativa mediante comunicación serial I2C y los sensores de luz por medio de dos canales del convertidor analógico a digital (ADC). En la figura 66 se muestra un diagrama general del montaje físico de los nodos RFD.

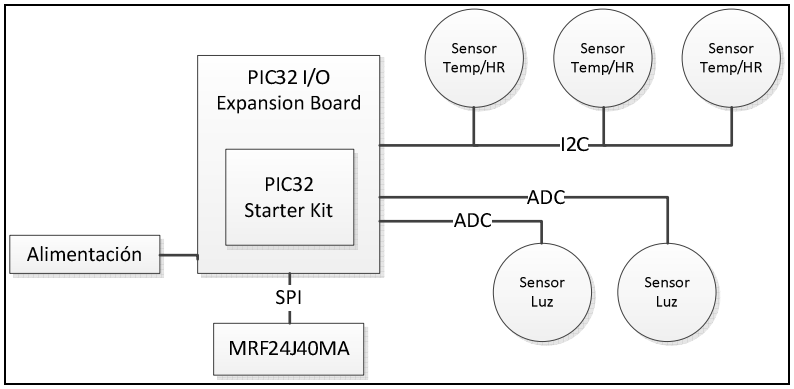


Figura 66. Arquitectura del nodo de recolección de datos CRTecMote utilizado en la caracterización mini jardines clonales del GENFORES

Los sensores seleccionados para este proyecto fueron los HYT-271 comercializados por la empresa Hygrosens Instruments, de origen estadounidense. Sus principales características se listan en la tabla 14.

La conexión eléctrica de los dispositivos al bus I^2C debe incluir resistencias de Pull-Up en cada una de las líneas SCL y SDA para asegurar que cuando no se está realizando ninguna transferencia se mantengan las líneas en alto (“1” lógico). La longitud máxima de un bus I2C depende de la capacitancia de la línea aunque existen maneras de alargarlas, mediante buffers especializados. En la figura 67 se puede observar un ejemplo de conexión de dispositivos a un bus I2C. En el caso particular de este proyecto se comprobó experimentalmente que no sería necesaria la utilización de los buffers para la extensión del bus, las resistencias de pull up utilizadas fueron de 2.4 kΩ tal como lo sugiere el fabricante y se colocó un par de ellas para cada sensor utilizado.

Tabla 14. Resumen de características sensor HYT-271

Característica	Valor
Rango de medición %HR	0% - 100%
Rango de medición de temperatura	-40ºC – 125ºC
Precisión HR	±1.8 % HR
Precisión de temperatura	±0.2 ºC
Resistente a químicos	Sí
Resistente a condensación	Si
Voltaje de operación	2.7V – 5V
Comunicación	I2C

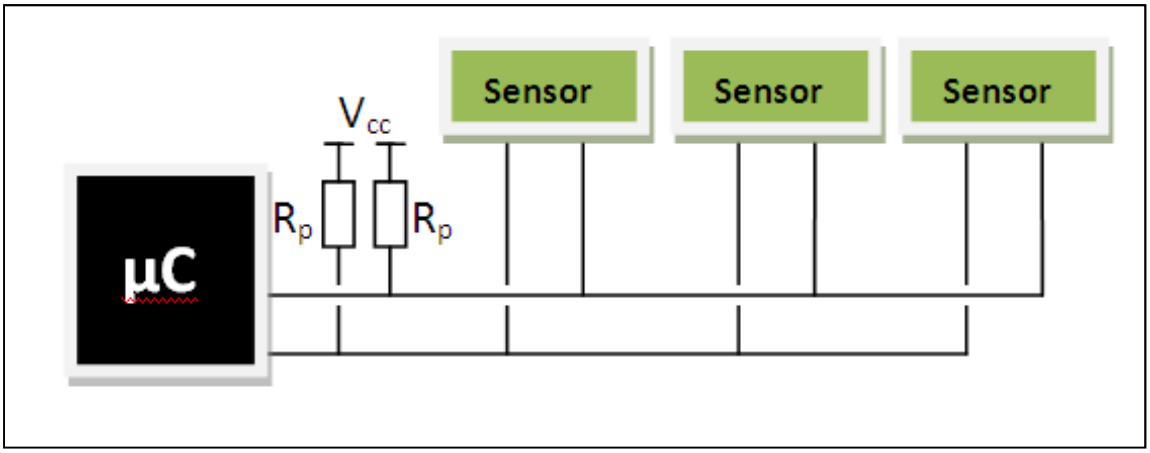


Figura 67. Conexión de sensores bajo el protocolo de comunicación I^2C para un nodo de medición CRTecMote

Con respecto a los sensores de luz utilizados, se optó por fotodiodos de silicio BPW34 ya que como lo muestra la figura 68 son sensibles a la luz visible, ubicada entre 400 y 700 nm, además son pequeños, baratos y de fácil implementación. La lectura de estos se realizó por medio del ADC del microcontrolador, de manera que se midiera el voltaje sobre una resistencia de 10 kΩ conectada en serie al fotodiodo.

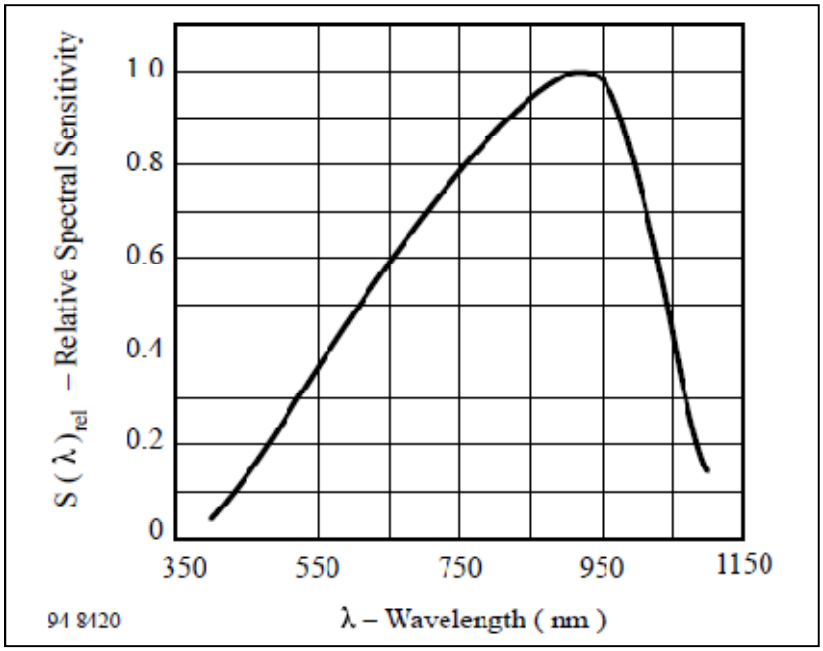


Figura 68. Curva de sensibilidad de los sensores de luz utilizados

Para la obtención de la respuesta del sensor a la luz se realizó un experimento donde se expuso al fotodiodo a diferentes intensidades lumínicas, aplicadas variando la distancia entre una lámpara de alta intensidad y el circuito de prueba, así se midió el voltaje sobre la resistencia. Se utilizó como referencia las mediciones de un luxómetro expuesto a la misma cantidad de luz en el mismo momento que el circuito de prueba. Como resultado de este experimento se obtuvo la gráfica de la figura 69 de la cual se deriva la ecuación que se utilizó posteriormente para la interpretación de los datos de voltaje leídos por el ADC, esta ecuación de polinomios que se muestra en la parte superior de la figura 69.

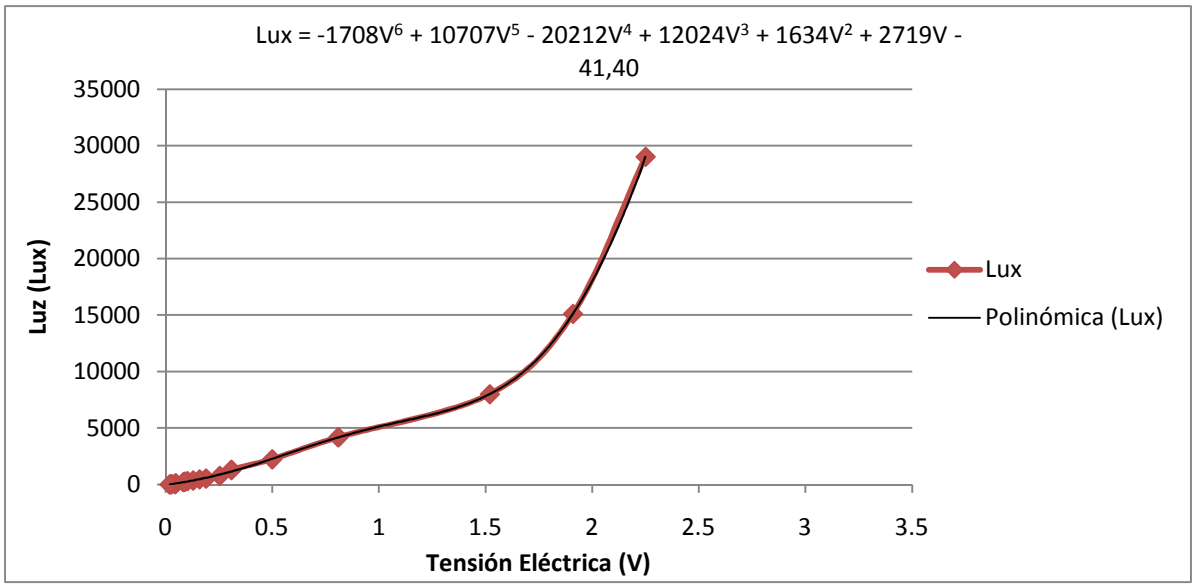


Figura 68. Caracterización experimental de los sensores de luz utilizados en la implementación de la red inalámbrica CRTecMote

El nodo CRTecMote FFD constituye el núcleo del sistema, es este el encargado de solicitar los datos a los nodos CRTecMote RFD en el momento justo, administrar la red inalámbrica, almacenar la información recopilada en memoria no volátil, almacenar y soportar la página web desde la cual se accede a los datos, en fin es el nodo maestro y de mayor importancia en el sistema. El nodo CRTecMote FFD utilizado cuenta con todas las interfaces de hardware necesarias para la implementación de la aplicación solicitada por GENFORES. El dispositivo de memoria utilizado fue una memoria FLASH de 2GB de capacidad. La figura 69 muestra la arquitectura del nodo CRTecMote implementado como coordinador o sumidero de la red.

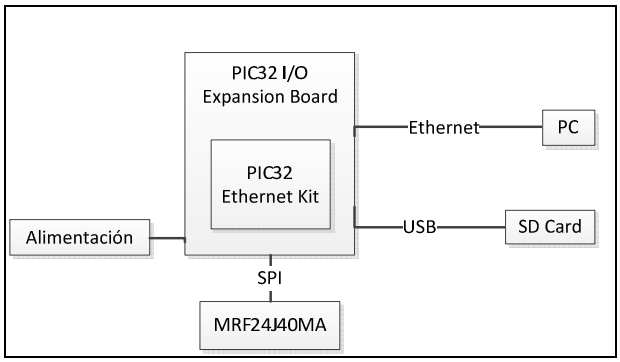


Figura 69. Arquitectura del nodo CRTecMote sumidero o FFD utilizado en la aplicación

Todas las rutinas de software implementadas en este proyecto se debieron diseñar para ser ejecutadas en el sistema operativo de tiempo real SIWA-RTOS, La escritura del código para correr sobre este sistema operativo implicó la creación de tareas que son ejecutadas según los algoritmos de calendarización, que garantizan un uso compartido del procesador por cada tarea. Seguidamente se hará una descripción de las rutinas o tareas que se implementaron en cada nodo, tanto en los RFD como en el FFD.

La rutina principal o "main" de los nodos es bastante simple, nada más debe inicializar el hardware necesario para cada aplicación, crear las tareas correspondientes y finalmente iniciar el calendarizador. La inicialización del calendarizador implica que el sistema nunca saldrá de esa rutina a no ser que se dé un problema de que no existe suficiente memoria en la pila para soportar esta rutina. En la figura 70 se muestra un diagrama donde se aprecia con claridad las acciones realizada en la rutina principal.

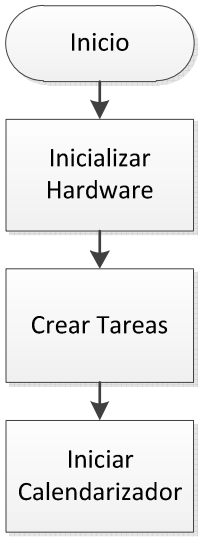


Figura 70. Flujo del programa principal de los nodos CRTecMote

Esta se puede decir que es la tarea principal de los nodos RFD, en ella se hace el manejo de la comunicación con el nodo central, se interpretan las instrucciones de este y se coordina la toma de datos y envío de los mismos.

Esta tarea es prácticamente igual en cada nodo RFD con la salvedad de las variables que identifican a cada nodo como NODO 1, NODO 2 o NODO 3. En la figura 71 se presenta el detalle del algoritmo seguido por esta tarea.

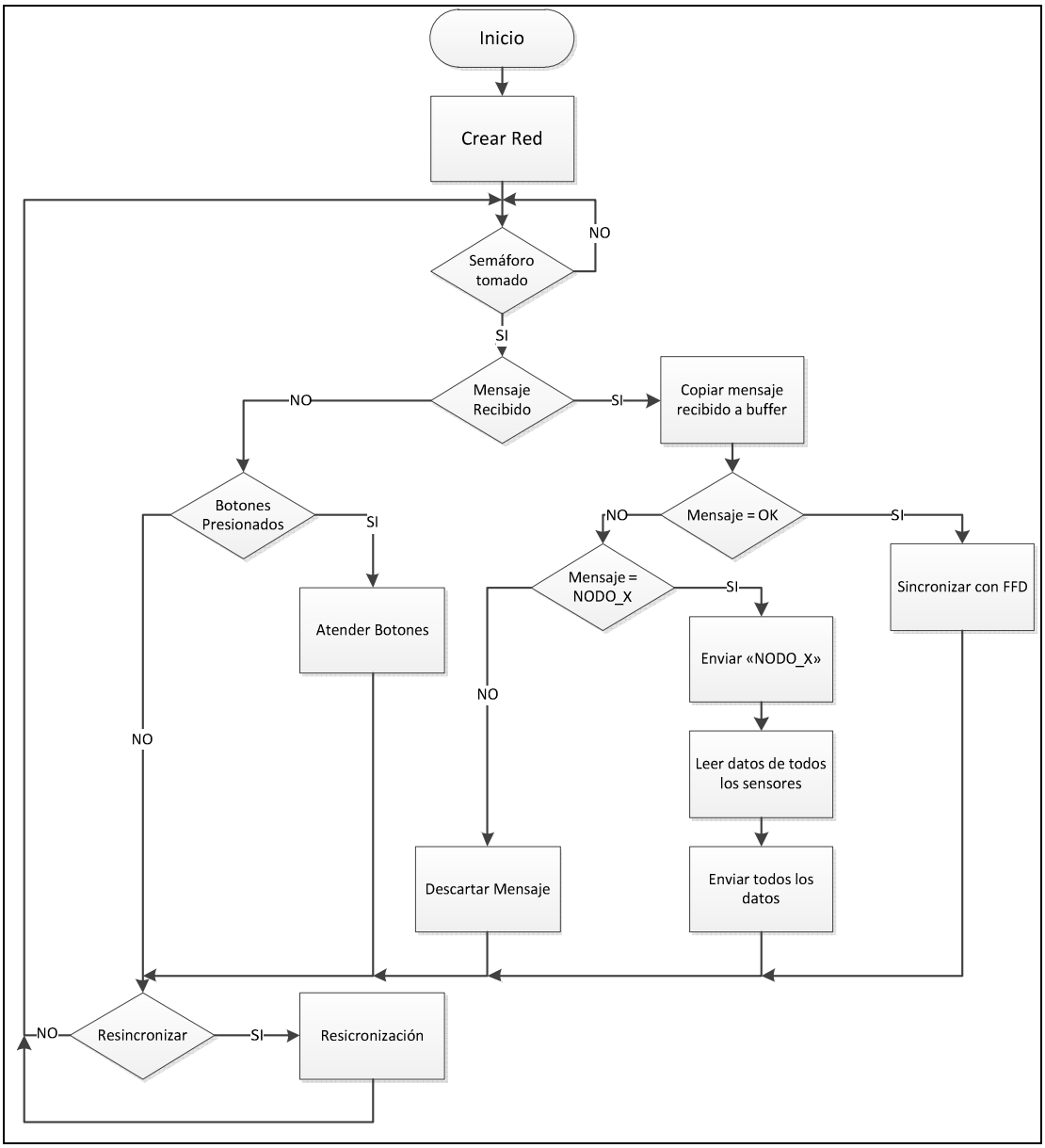


Figura 71. Diagrama de flujo de la tarea de conexión de área local de los nodos CRTecMote denominada taskMIWI

En el diagrama de flujo de la tarea taskMIWI de la figura 71, se observa que cuando llega un mensaje con el nombre asignado al nodo receptor se responde

inmediatamente con el mismo nombre emulando un reconocimiento del mensaje, luego se leen los datos de todos los sensores.

La lectura de estos datos se refiere a la ejecución de la rutina de manejo del bus I^2C para extraer los datos de los sensores de humedad relativa y temperatura, además de la lectura de los canales del ADC donde se encuentran conectados los sensores de luz. De la lectura de los ADC se hablará en el siguiente punto de este apartado, por ahora se tratarán los detalles de la implementación de las rutinas de manejo del protocolo I^2C

Se partió de un ejemplo de código brindado por Microchip denominado "I2C Peripheral Library Example". De este ejemplo se toman 3 funciones y además se diseñan otras 4. Los métodos creados son MeasurementRequest, DataFetch, Read_One y Real_All. MeasurementRequest y DataFetch fueron diseñadas siguiendo los lineamientos establecidos por el fabricante en la hoja de datos [22] y permiten solicitar al sensor que inicie su ciclo de medición y extraer las mediciones. Read_One y Real_All fueron creadas para funcionar como APIs y así darle mayor facilidad al usuario en la interacción con los sensores, solo estas dos funciones son llamadas desde la tarea taskMIWI.

En las siguientes páginas se describirán todas las funciones necesarias para interactuar con los sensores HYT-271, comenzando por las más básicas y terminando en las que poseen mayor grado de abstracción.

Al igual que para los nodos RFD, la tarea taskMIWI es la de mayor importancia en el nodo CRTecMote central, sumidero o FFD, es en esta donde se crea y mantiene la red inalámbrica, desde la que se hacen las solicitudes de datos y se sincroniza la operación de las demás tareas.

Para mayor claridad en la explicación de la implementación de esta tarea se puede observar el diagrama de su flujo mostrado en la figura 72. Este diagrama muestra información básica acerca de la secuencia de actividades realizadas, no así los detalles de su implementación. Detalles acerca de cómo se realizan las acciones citadas en este diagrama podrán ser encontradas más adelante en esta misma sección de este informe.

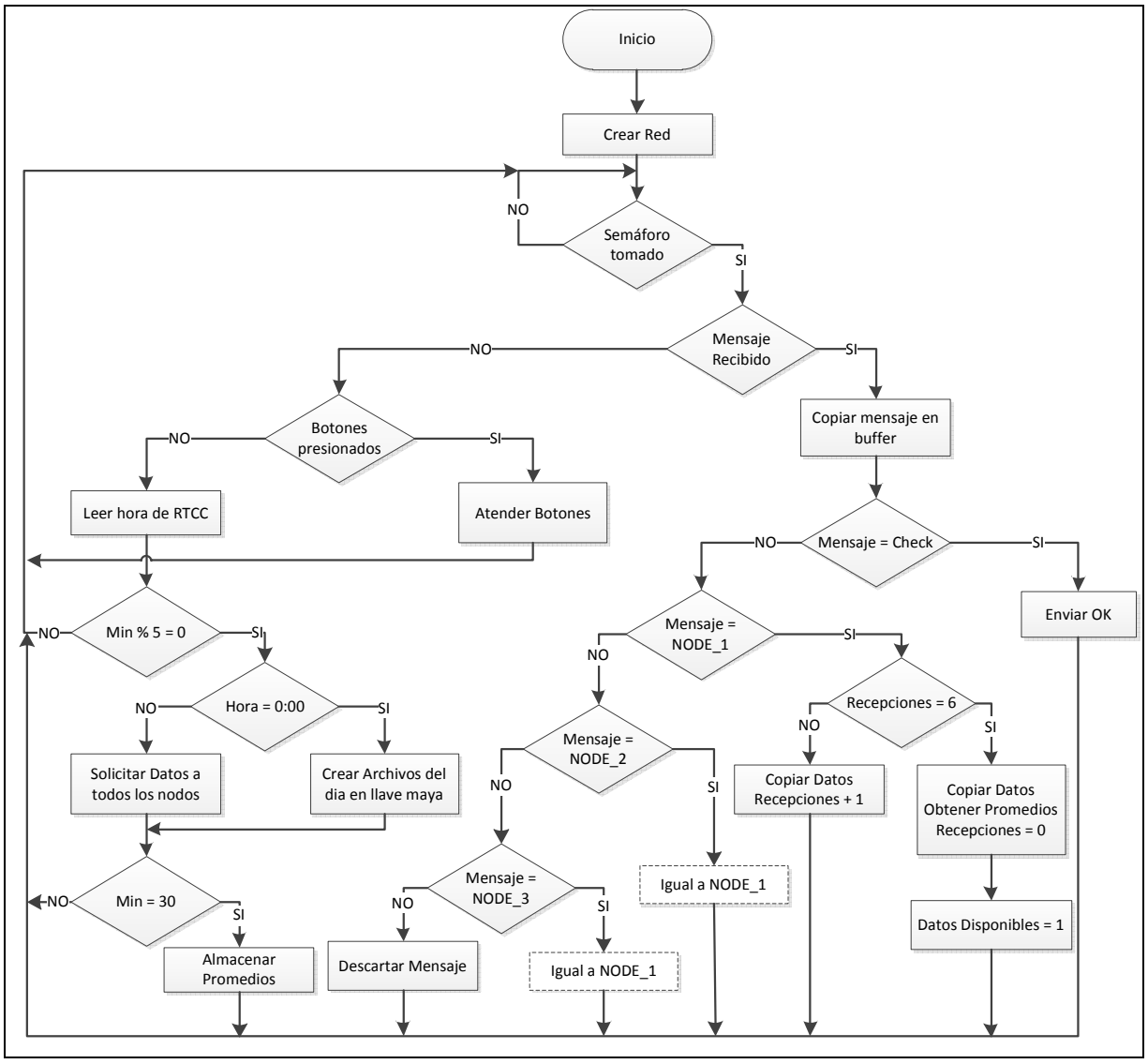


Figura 72. Diagrama de flujo del proceso de conexión de red ejecutado en el nodo sumidero de CRTecMote

Existen varios aspectos que destacar del diagrama anterior. El primero es cuando se recibe un mensaje “CHECK”, esta es una modificación en la comunicación entre el nodo central y los nodos periféricos que debió ser agregada ya que cuando se desconectaba uno de los nodos RFD el nodo central este no lo registraba de manera tal que al re-sincronizarse el nodo perdido, se interpretaba como una conexión nueva lo que eventualmente llenaba la tabla de conexiones existentes y deshabilitaba la comunicación. Como solución al problema anterior se

implementó una función en los nodos RFD que envía un código interpretado como “CHECK”, al recibirse este código se responde con el código “OK”. SI no se recibe el “CHECK” se interpreta que el nodo está perdido y realizando labores de resincronización.

Otro aspecto de gran importancia es que cada 5 minutos se realizará una solicitud de los datos recopilados por los nodos RFD. Las labores de temporización se realizan gracias al módulo RTCC (Reloj de tiempo real y calendario) del cual se discutirá más adelante. Al enviar la petición de datos, esta dispara la lectura de los sensores de cada nodo y el consecuente envío de la información departe de cada RFD, el mensaje entrante al nodo central está encabezado por el código “NODE_X” (siendo X el número de nodo) y dependiendo de este encabezado se ejecutará la rutina indicada para cada nodo.

Seguidamente de cada recepción se llama al método Copy_Data el cual recibe como parámetros el número de solicitud y el nodo que está realizando la llamada, esta función se encarga de copiar los datos desde el buffer de entrada a buffers dedicados para cada nodo, conforme van copiando los datos se realizan las operaciones matemáticas necesarias para convertir la información a las unidades requeridas, ya sea grados centígrados para temperatura, porcentaje de humedad relativa y lux para la cantidad de luz. Los datos convertidos se almacenan en variables del tipo flotante, al pasar 30 minutos se tendrá un total de 6 valores en este arreglo y se llamará además de la función Copy_Data (para almacenar el sexto valor) a la función Promedios, esta calcula los valores promedio de cada sensor para un nodo en específico y almacena estos promedios en un nuevo arreglo de datos.

Tambien se desarrolló un conjunto de rutinas para el almacenamiento de la información recolectada. Esta tarea se denominó “Storage”.

Dentro de las funciones más destacadas llevadas a cabo en Storage se encuentran la llamada a la función que configura el RTCC, la lectura de datos desde la llave maya para ser desplegados en la página web, la creación de

directorios y archivos en la llave maya y el almacenamiento de información cada vez que se requiere por la tarea taskMIWI.

A continuación se presenta el diagrama de flujo general (figura 73) de esta tarea y seguidamente la descripción detallada de las principales acciones realizadas en ella.

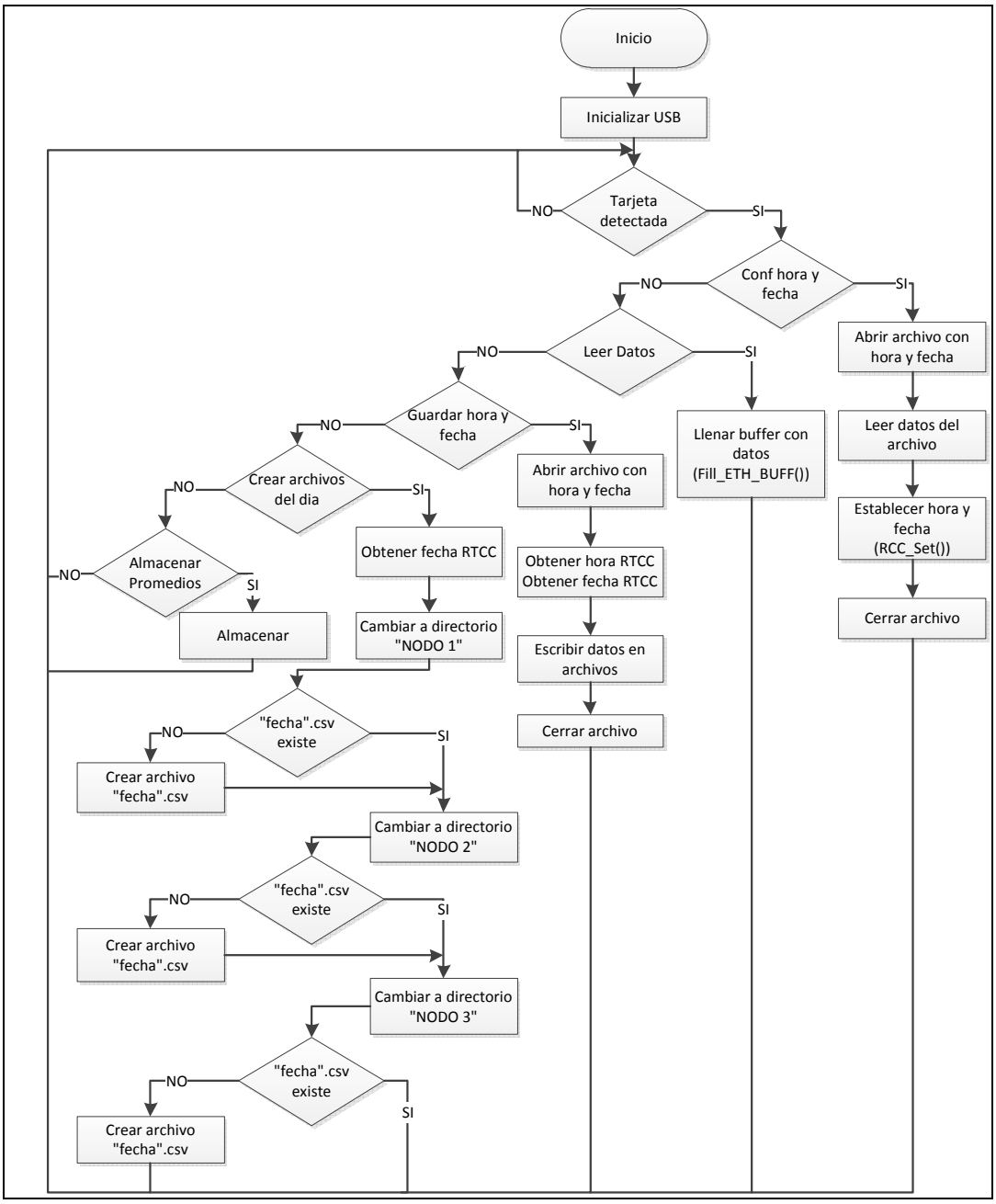


Figura 73. Diagrama de flujo de la tarea de almacenamiento implementada en el nodo sumidero

Para el manejo de la hora y fecha para etiquetar la recolección de datos se debió diseñar una tarea que permitiera la administración del tiempo en la red CRTecMote. Esta tarea se realiza al inicio de la ejecución del programa y sólo se lleva a cabo 1 vez. Su función es leer la hora y fecha almacenadas en un archivo en la llave maya y configurar el RTCC con esta información. Se creó dada la necesidad de que el sistema pudiera arrancar por sí mismo en caso de una falla de energía o desconexión voluntaria. Utiliza tanto funciones de la biblioteca de USB como del SIWA-RTOS. Durante su creación se debieron cuidar detalles como el formato de los datos ya que cuando se lee de la llave maya lo que se obtiene es información como caracteres ASCII, estos deben ser pasados a valores BCD para ser finalmente asignados al RTCC ya que así lo requiere este módulo.

Una vez recolectada y almacenada la información se debía poder mostrar al usuario de la red CRTecMote. Para ello se diseñó un conjunto de tareas que permitieran la conexión al nodo Sumidero CRTecMote a través de la interfaz Ethernet que este posee. Esta función se lleva a cabo bajo petición directa del usuario final por medio un botón de la página web de visualización de los datos. Al obtenerse la bandera se llama a la función FILL_ETH_BUFF que recibe como parámetros el número de nodo requerido y la fecha en la que fueron tomadas las mediciones.

Esta es la tarea que soporta las funciones del protocolo TCPIP en la tarjeta de desarrollo. Para su implementación se utilizó el conjunto de librerías facilitadas por Microchip denominado TCPIP_Stack y un ejemplo de uso de esta librería. Se modificó una página web incluida en el ejemplo para que se ajustara a las necesidades del proyecto y el código en C que permite al programa principal interactuar con esta página. Las principales funciones de esta tarea de visualizan a partir de las interfaces gráficas de las distintas páginas creadas, estas son la página principal, la página de visualización de la tabla de valores del día para cada sensor, la página de visualización de la gráfica de valores de los parámetros por día, la página de envío de correo electrónico y una última que permite el cambio de la contraseña de ingreso a la página. A continuación se hará una breve descripción de cada una de estas páginas.

A partir de la funciones de comunicación de red, se crea una página web que permite al usuario interactuar con la red.

Esta página es la que recibe al usuario luego de introducir el usuario y contraseña correctos, su apariencia es como la mostrada en la figura 74 y permite ver la ubicación espacial de cada sensor a lo largo del túnel mediante indicadores de diferentes colores, también muestra los valores mínimo, máximo y el promedio de cada día para el sensor seleccionado. Para ver los datos se debe antes haberlos cargado desde la página de visualización de la tabla histórica.

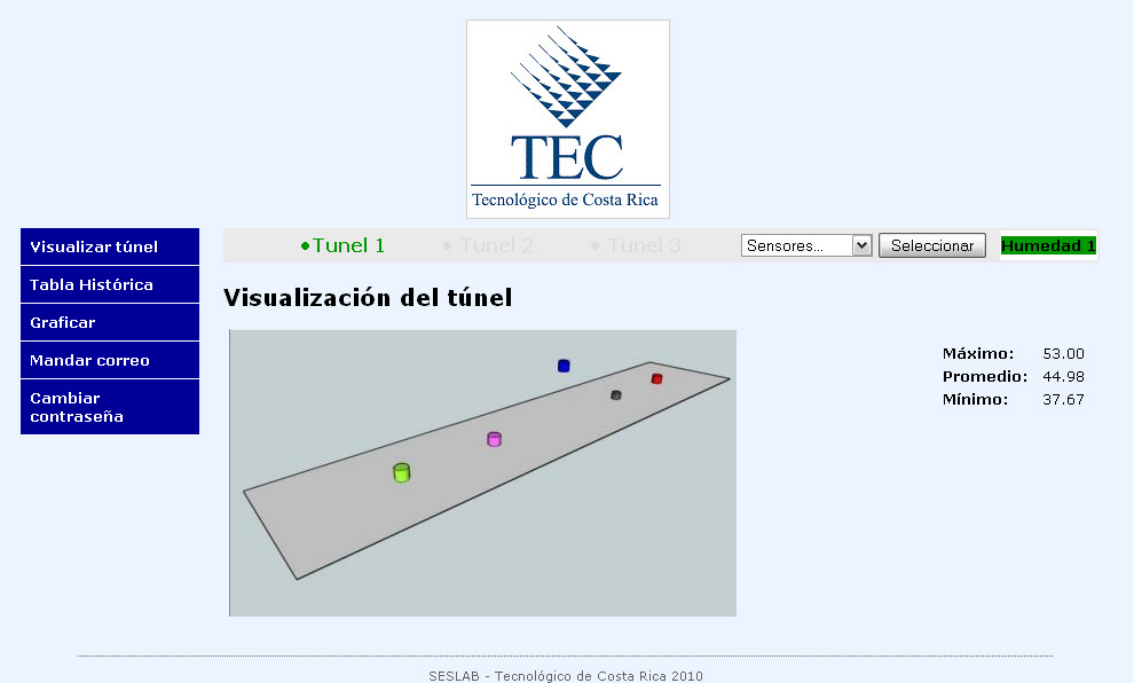


Figura 74. Interfaz de usuario diseñada para la aplicación de monitorización de mini-jardines clonales de GENFORES utilizando la red CRTecMote.

Además de este formato de visualización se diseño un formato de visualización por medio de tabla para que los usuarios del sistema pudieren determinar puntos de comportamiento anormales dentro del jardín.

Es en esta página donde se brinda la información de los datos recopilados para cada sensor en una fecha especificada por el usuario. Los datos se cargan en la página mediante la introducción de la fecha requerida en los campos destinados para este fin, además se debe haber marcado el túnel del cual se desea visualizar

la información. Ya con los primeros datos cargados que están predefinidos para ser los del sensor de temperatura 1, se puede cambiar la selección de sensor para ver los datos de los demás sensores. Un muestra de cómo se ve esta página se encuentra en la figura 75. En esta se puede observar claramente que se están visualizando los datos del sensor de luz 1 colocado en el túnel 1 para la fecha del 6 de abril del 2011.

Al presionar el botón “Seleccionar” se levanta la bandera que permite a la tarea Storage pasar los datos de la llave maya a la página web, también se obtienen los parámetros requeridos por la función FILL_ETH_BUFF en esta interfaz tal como se mencionó anteriormente.

Figura 75. Tabla de datos generada con la red inalámbrica CRTecMote

Con esta figura se muestra el funcionamiento de una red CRTecMote para una aplicación real de monitorización. En el capítulo de apéndices y anexos se adjuntan fotos de la implementación de esta aplicación en el invernadero de GENFORES ubicado en la sede del TEC en San Carlos.

Tesis y publicaciones generadas a partir del proyecto

Gracias al proyecto de investigación se logró el desarrollo de una serie de proyectos de graduación de grado de licenciatura para estudiantes de la escuela de Ingeniería Electrónica que se especifican a continuación.

- **II semestre del 2009**

Título de la tesis	Autor
Sistema de administración en tiempo real para los recursos de hardware y software de un nodo de sensado	Norwing Alexander Leiva Delgado
Organización de la información generada por una red inalámbrica de sensores CRTecMote: LOADPOINT	Dennis Rodríguez Rodríguez

- **I semestre del 2010**

Título de la Tesis	Autor
Optimización del módulo de planificación de procesos del sistema operativo de tiempo real SIWA-RTOS utilizado en los nodos CRTecMote	Alexander Valverde Serrano
Desarrollo de un sistema convertidor de protocolos para comunicación inalámbrica vía GSM, arquitectura abierta CRTECMote	Miguel Angel Fonseca Porras
Diseño e implementación del módulo de comunicación inalámbrica de un nodo CRTECMOTE para enlaces de tipo punto-multipunto	Sebastián Lopez González

- **II semestre del 2010**

Título de la Tesis	Autor
Sistema de caracterización de parámetros climáticos en Mini Jardines Clonales Forestales bajo ambiente protegido	Jose Pablo Fallas Zúñiga
Simulación de los protocolos GPRS para redes de inalámbricas de sensores que utilizan redes de telefonía móvil para comunicación de área extendida	Sofía Ortiz Argüello

También se han generado las siguientes ponencias en congresos y conferencias:

Nombre de la ponencia	Tipo y nombre de la actividad	Tipo de ponencia
Diseño de un nodo con arquitectura abierta para redes inalámbricas de sensores: CRTecMote	Jornadas de Ingeniería Electrónica, Ecuador, junio 2009	Presentación Oral
SIWA: An Open RTOS for embedded Systems	2nd Hedisc Workshop, Costa Rica December 2010	Presentación Oral
Kernel Optimization for RTOS used in energy restricted environment	Technology for Sustainable Development Conference, Costa Rica February 2011	Poster

Implementación de protocolos de comunicación de bajo consumo energético para redes inalámbricas de sensores	Embedded Technology Conference, March 2011	Artículo y Presentación

Además se presentó el siguiente artículo para Revista Tecnología en Marcha publicado el segundo cuatrimestre del 2011:

"Diseño de un nodo con arquitectura abierta para aplicaciones con redes inalámbricas de sensores (CRTECMOTE)". Tecnología en Marcha, Segundo cuatrimestre 2011.

Se plantea enviar un par de artículos adicionales con los resultados obtenidos con el proyecto y que estén relacionados a:

- Simulación de sistemas redes de sensores utilizando telefonía móvil
- Sistemas de fusión de datos para redes inalámbricas de sensores

Sin embargo no se han identificado las conferencias o revistas para este propósito.

Capítulo VI: Conclusiones y recomendaciones

Conclusiones

- Se logró la implementación funcional de una arquitectura de red inalámbrica de sensores capaz de operar en ambientes abiertos expuestos a condiciones ambientales de alta humedad.
- Se diseñó una red con topología estrella con nodos de medición RFD y nodos de coordinación o sumideros FFD capaz de establecer comunicación entre eficaz ellos.
- Existe una relación no lineal entre el consumo de potencia de un nodo CRTECMOTE con respecto a la frecuencia de operación.
- Una variación en el quantum o Tick del SIWA-RTOS no generó un cambio relativamente significativo entre un valor de consumo de potencia de los nodos CRTecMote.
- El empleo del modo de bajo consumo de potencia, IDLE, permitió reducir el consumo de potencia del microcontrolador al ejecutar el RTOS modificado (SIWA-RTOS) en aproximadamente 8%, respecto a la ejecución del FreeRTOS original.
- La reestructuración del código fuente del SIWA-RTOS generó una reducción en el consumo de potencia promedio del nodo CRTecMote en aproximadamente 12%, respecto al sistema original.
- La ejecución del SIWA-RTOS por parte del nodo CRTecMote obtuvo un porcentaje de reducción en el consumo de potencia promedio de aproximadamente el 47.5% respecto al valor original cuando se ejecutaba el FreeRTOS.
- El promedio teórico de incremento en la duración de la carga de una batería de 9V al ser utilizada como fuente de alimentación para el CRTecMote cuando ejecuta el SIWA-RTOS fue del 94.5% respecto al valor para el FreeRTOS.

- Para un nodo coordinador funcionando dentro de una red en topología estrella, el mayor consumo de corriente se da en los instantes de inicialización del dispositivo, de la pila de protocolo de red y de la creación de la red.
- Para un nodo CRTECMOTE esclavo funcionando dentro de una red en topología estrella, el mayor consumo de corriente se da en los instantes de inicialización del dispositivo, de la pila de protocolo de red y de búsqueda de redes disponibles.
- A una distancia menor de 100 metros, 2 nodos CRTECMOTE mantienen la calidad del enlace entre dispositivos sin pérdida de conexión.
- La adaptación de la aplicación para enviar y recibir mensajes de texto desde el sistema operativo de tiempo real SIWA-RTOS, permitió brindar un servicio de comunicación de largas distancias a la plataforma de redes inalámbricas CRTECMote.
- La transmisión de datos por medio de el acople a nivel de hardware y software entre el CRTecMote y el modem EDGE-USB tuvo un porcentaje de efectividad del 100%.
- El consumo de potencia durante el envío de un mensaje de texto se cuantificó en 42mW mientras que el consumo durante recepción de un mensaje fue de 43mW.
- La información enviada desde el nodo sumidero CRTecMote es la misma recibida en la estación de control, un 100% de integridad de datos.
- La reestructuración al modelo de configuración ACM-Serial Emulation, permitió ejecutar comandos bajo el protocolo V2.5ter AT en un modem de tipo GSM-USB.
- El diseño de un sistema de organización de la información de datos de launa red de sensores depende de la estructura de los datos, del medio para obtener los datos y de la aplicación que se busca del sistema.
- Se comprobó que el funcionamiento de la estructura de un modelo de base de datos depende de la correcta implementación de una rutina para crear cada una de sus tablas.

- La implementación de un modelo para organizar la información es simplificado mediante el uso de un sistema gestor de bases de datos y de los procedimientos almacenados para administrar el contenido de su estructura.
- El porcentaje de éxito en la recepción de mensajes en el sistema de organización de la información fue de un 100 % para las tres pruebas realizadas en un día.
- El funcionamiento del sistema de organización de la información permitió obtener que el tiempo promedio que tarda un mensaje desde el momento en que se envía al momento en que se recibe es de 1 minuto y 12 segundos en una misma celda de la red GSM.
- Los sensores de humedad relativa y temperatura digitales disponibles en el mercado brindan mejores prestaciones en cuanto a exactitud que los sensores analógicos comerciales.
- La arquitectura de red CRTecMote puede extender su funcionalidad al incluir dispositivos de almacenamiento masivo USB y funciones de red mediante interfaz Ethernet.
- Los sensores de humedad relativa HYT-271 brindan datos de humedad relativa y temperatura con muy baja variabilidad dado que son sensores digitales.
- Los sensores analógicos de luz están limitados en su margen de medición inferior por consideraciones físicas del circuito de conexión al microcontrolador por lo que el sistema no se considera indicado para medir cantidades menores a 500 lux.
- La red inalámbrica de sensores CRTecMote permite el acople de sensores tanto analógicos como digitales.
- Se logró establecer una tecnología de redes inalámbricas de sensores para su utilización en otros proyecto de investigación científica como por ejemplo el proyecto eBRIDGE.
- Se logró promocionar la tecnología de redes inalámbricas de sensores como una línea de investigación del laboratorio SESLab.

- Se logró promocionar la tecnología desarrollada en proyecto para su utilización tanto a nivel institucional como a nivel regional con otras universidades en panamá y Brasil.
- Se logró adaptar el nodo para su operación en ambientes exteriores con condiciones operativas extremas.
- Se logró el desarrollo de 7 tesis de Licenciatura de Ingeniería Electrónica por medio de la realización del proyecto.

Recomendaciones

- Durante la consecución del proyecto se utilizaron varias versiones del ambiente de desarrollo MPLAMTM y de librerías de Microchip Inc. Sin embargo, en las constantes actualizaciones de las librerías, en su mayoría, se modifican aspectos a nivel de compilador, por lo que es recomendable trabajar sobre las versiones más actualizadas del ambiente de desarrollo.
- Existe una tendencia a crear tareas sobre el sistema operativo SIWA-RTOS con una capacidad mínima de la pila de memoria. Sin embargo, para la tarea que mantiene la pila de red debe tener un tamaño de pila de memoria mayor a 500 palabras, dado que el proceso de interrupciones, por manejar gran cantidad de datos, puede causar desbordamiento de la pila.
- Para que las tareas de captura de datos no interfieran con la tarea de la pila red, esta última debe tener una prioridad igual a tskIDLE_PRIORITY + 1, mientras que las de captura deben tener una prioridad igual a configKERNEL_INTERRUPT_PRIORITY + 1.
- Para cumplir con las características de sistema operativo multitarea, se debe manejar la comunicación entre procesos y manejar eventos periódicos o aperiódicos.
- Para reducir aún más el consumo de potencia en un nodo CRTECMOTE esclavo, periódicamente, se puede poner en modo de bajo consumo tanto al transceptor como al microcontrolador.
- Para obtener un mayor alcance, se puede utilizar el transceptor MRF24J40MB de la empresa Microchip Technology Inc. que presenta las

mismas características que el MRF24J40MA con la adición de un amplificador de salida que proporciona una potencia de salida de 20 dBm, comparado con los 0dBm del MRF24J40MA.

- Para darle más flexibilidad, se debe encontrar la forma de soportar la característica de multi-salto de mensajes para alcanzar dispositivos en un rango más amplio.
- Es necesario someter la interfaz de visualización de los datos a la opinión de los funcionarios de GENFORES para que brinden realimentación en cuanto a las formas en las que desean interactuar con el sistema.
- La sustitución de los sensores analógicos de luz por sensores digitales con interfaces seriales podría ayudar a aumentar el rango de medición de luz, de forma que sea posible realizar mediciones de baja iluminación.
- Para efectos de ahorro económico, se debe buscar la forma de desarrollar una tarjeta de circuito impresa que permita acoplar los componentes de los nodos CRTecMote entre ellos y con futuros componentes, de forma que se independice el sistema de los kit de desarrollo comerciales y se aprovechen al máximo las prestaciones del hardware utilizado.
- Se debe buscar una base para conectar los sensores a las líneas de datos que cumplan tanto con los requerimientos técnicos y de seguridad como con los principios básicos de la estética.

Capítulo VII: Referencia Bibliográficas

[1] Ian F. Akyildiz, Weilian Su, Zogesh Sankarasubramaniam, y Erdal Cayirci. A survey on sensor networks. IEEE Communications Magazine, pags. 102-114, Agosto 2002.

[2] Jennifer Yick, Biswanath Mukherjee, y Dipak Ghosal. Wireless sensor network survey. Computer Networks, 52:2292-2330, 2008.

[3] A. Baggio. Wireless sensor networks in precision agriculture. In Proceedings of the Workshop on Real-World Wireless Sensor Networks (REALWSN'05), Junio 2005.

[4] Andrew S. Tanenbaum, Chandana Gamage, y Bruno Crispo. Taking sensor networks from the lab to the jungle. IEEE Computer, pags. 98-100, August 2006.

[5] Holger Karl y Andreas Willig. Protocols and Architectures for Wireless Sensor Networks. Wiley, 2005.

[6] Vijay Raghunathan, Curt Schurgers, Sung Park, y Mani B. Srivastava. Energy-aware wireless microsensor networks. IEEE Signal Processing Magazine, pags. 40-50, Marzo 2002.

[7] Hill, Jason L. System Architecture for Wireless Sensor Networks. PhD thesis, University of California, Berkeley, Spring 2003.

[8] Chacon-Rodriguez, A. Julian, P. Castro, L. Alvarado, P. Hernandez, N. Evaluation of gunshots detection algorithms. Circuits and Systems I: Transactions. Vol. 58 issue 2 pp. 363-373. 2011.

[9] Pablo Alvarado, Apolinar González, y Luis Villaseñor. Propuesta de aplicación de redes de Sensores en el modelado de cultivos protegidos y en campo. In Memorias. "Workshop on Sensor Networks and Applications", Gramado, Brasil, Setiembre 2008

[10] Cross, Nigel, Engineering design methods: strategies for product design. John Wiley & Sons Inc., 3rd ed. 2005

[11] Science and Technology-Engineering Curriculum Framework. Spring 2001. Visitado el 15-03-2011. Actualizado Mayo del 2001.

http://www.doe.mass.edu/frameworks/scitech/2001/standards/strand4.html

[12] Raghavendra, C.; Sivalingam, K.; Znati, T. Wireless Sensor Networks. Springer: New York, NY,USA, 2004.

[13] Culler, D.; Estrin, D.; Srivastava, M. Overview of sensor networks. IEEE Comput. 2004, 37, 41–49.

[14] Lin, C.; Tseng, Y.; Lai, T. Message-Efficient In-Network Location Management in a Multi-sink Wireless Sensor Network. In Proceedings of IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, Taichung, Taiwan, 2006; pp. 1–8.

[15] Buratti Chiara; Conti Andrea; Dardari Davide; Verdone Roberto. An Overview on wireless sensor networks Technology and Evolution. Sensors. Vol 9. 2009.

[16] Verdone, R.; Dardari, D.; Mazzini, G.; Conti, A. Wireless Sensor and Actuator Networks; Elsevier: London, UK, 2008.

[17] Dargie, Waltenegus; Poellabauer, Christian. Fundamentals of Wireless Sensor Networks: Theory and Practice. Wiley and Sons. 2010. Pp 9-15.

[18] AN1204, “Microchip MIWI™ P2P Wireless Networking Protocol” (DS01204), Microchip Technology Inc., 2008.

[19] Gay, D., Levis, P., and Culler, D. (2007) Software design patterns for TinyOS. ACM Trans. Embed. Comput. Syst. 6 (4), 22.

[20] Han, C.C., Kumar, R., Shea, R., Kohler, E., and Srivastava, M. (2005) A dynamic operating system for sensor nodes. MobiSys ’05: Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services (pp. 163–176). ACM, New York, NY, USA.

[21] Dunkels, A., Gronvall, B., and Voigt, T. (2004) Contiki: A lightweight and flexible operating system for tiny networked sensors. LCN ’04: Proceedings of the

29th Annual IEEE International Conference on Local Computer Networks (pp. 455–462). IEEE Computer Society, Washington, DC, USA.

[22] Cao, Q., and Abdelzaher, T. (2006) LiteOS: A lightweight operating system for C++ software development in sensor networks. SenSys '06: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (pp. 361–362). ACM, New York, NY, USA.

[23] Khan, Asheq. "Data Fusion in Sensor Networks". Presentation online. University of New York. Visited February 2011.

http:// www.cse.buffalo.edu/~qiao/cse620/fall04/Data_Fusion.ppt

[24] K. Dasgupta, K. Kalpakis and P. Namjoshi, "An Efficient Clustering-based Heuristic for Data Gathering and Aggregation in Sensor Networks," IEEE WCNC, 2003.

[25] Berkeley Design Technology Inc. staff, "An independent analysis of the: MIPS Technologies MIPS32® M4K™ synthesizable processor core". www.BDTI.com , 2007.

[26] Coughlin Robert, Driscoll Frederick. Amplificadores operacionales y circuitos integrados lineales. Quinta Edición. Pearson Education, 2000.

[27] PIC32 RTOS comparison. Visitado el 02 de junio de 2009.

http://rtos.com/news/detail/Express_Logics_ThreadXMCU_RTOS_Scores_Top_Marks_in_Microchip_Technologys_PIC32_Benchmarks/

Capítulo VIII: Apéndices y Anexos

Fotos de la implementación de CRTecMote en los mini Jardines clonales de GENFORES (1/2)

Fotos de la implementación de CRTecMote en los mini Jardines clonales de GENFORES (2/2)