CT-730-SS1
Don. 2013

xx(209420.1)

Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional
Unidad Guadalajara

Diseño e Implementación de una Red Inalámbrica de Sensores

Tesis que presenta:
Miriam Alejandra Carlos Mancilla

para obtener el grado de:
Maestro en Ciencias

en la especialidad de:
Ingeniería Eléctrica

Director de Tesis
Dr. Luis Ernesto López Mellado

CINVESTAV del IPN Unidad Guadalajara, Guadalajara, Jalisco, Diciembre de 2012.

Diseño e Implementación de una Red Inalámbrica de Sensores

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Miriam Alejandra Carlos Mancilla
Licenciatura en Informática
Universidad de Guadalajara 2006-2010

Becario de Conacyt, expediente no. 243189

Director de Tesis
Dr. Luis Ernesto López Mellado

CINVESTAV del IPN Unidad Guadalajara, Diciembre de 2012.

DEDICATORIA

Esta tesis está dedicada a mi familia que me dio el apoyo en todo momento y siempre confió en mí para salir adelante, pero sobre todas las cosas gracias a mi madre que siempre está conmigo.

AGRADECIMIENTOS

A mi familia por siempre apoyarme.

A mis amigos por compartir mi triunfo y acompañarme durante esta travesía de mi vida.

A mi madre por ser un ejemplo de vida para mí.

A mi padre por su cariño y apoyo.

A mis compañeros de generación, por apoyarme y siempre obtener de ellos una sonrisa en los buenos y malos momentos que compartimos juntos.

A mis compañeros de laboratorio de computación, por ayudarme con las pruebas de campo.

A mis mejores amigos, que siempre me apoyaron y me dieron una palabra de aliento cuando sentía que todo estaba mal y aún así no me dejaron sola.

A dios, por darme la oportunidad de seguir adelante y darme la fuerza diaria de seguir con mis sueños.

A los doctores por su paciencia, atención y los conocimientos que me brindaron.

A mi asesor el Doctor Luis Ernesto López Mellado por su paciencia, ayuda y conocimientos que me guiaron durante la maestría.

Y por último y no menos importante a Conacyt por apoyarme durante mi estancia dentro del Cinvestav.

Diseño e Implementación de una Red Inalámbrica de Sensores

RESUMEN

En esta tesis se presenta el diseño e implementación de una red inalámbrica de sensores usando el dispositivo Freescale MC1321X. El procedimiento de gestión de la red diseñado opera en dos etapas; la primera consiste en la formación de la red bajo un criterio de reducción del consumo de energía y la segunda etapa donde se realiza la medición y concentración de la información sensorial. La formación de red se logra utilizando un algoritmo distribuido, previamente definido, basado en una estrategia de auto-organización. Un segundo algoritmo fue diseñado para el manejo de los sensores en una tarea simple de medición donde se colectan las temperaturas mínima, máxima y promedio de los sensores en la red. El algoritmo construye primeramente un árbol a partir del concentador (Sink) sobre la red formada; posteriormente efectúa las tareas de medición, procesamiento y transmisión de la información hacia el nodo Sink. La implementación conjunta de ambos algoritmos presentó ventajas en el ahorro de energía total en la red, reorganización y confiabilidad. Las pruebas se realizaron bajo diferentes escenarios en espacios cerrados y abiertos.

Design and Implementation of a Wireless Sensor Network

ABSTRACT

In this thesis the design and implementation of a wireless sensor network using the device Freescale MC1321X is presented. The designed network management procedure operates in two steps; the first one consists in the network formation following an approach of power consuming reduction, and the second one in which the measurements and collection of the sensory data is performed. The network formation is achieved using a previously conceived distributed algorithm, which is based on a self-organization strategy. A second algorithm is designed for sensor management within a simple measurement task in which temperatures minimal, maximal, and average are collected. This algorithm first builds a tree from the sink over the formed network; then it performs the tasks involving measurement, processing, and transmission to the sink node. The implementation of both algorithms provided evident advantages regarding low energy consumption, reorganization and reliability. The tests have been performed under different scenarios in open and closed environments.

Índice General

Capítulo III.
ALGORITMO DE AUTO-ORGANIZACIÓN

Capítulo IV.
DESARROLLO DE UN ALGORITMO PARA UNA RED DE SENSORES

Capítulo V.
IMPLEMENTACIÓN Y PRUEBAS

Introducción

Las redes de sensores han tenido gran aceptación para resolver diversos tipos de problemas de monitoreo de parámetros distribuidos en diferentes tipos de entornos, sea bien estructurados como edificios inteligentes, sea sin estructura predefinida como bosques o sembradíos. En el segundo caso, donde además es posible que los sensores cambien de ubicación, se requiere que la comunicación sea inalámbrica. Este tipo de aplicaciones presentan retos interesantes que surgen de la forma de interacción de los dispositivos: calidad variable de la comunicación, interferencia del entorno, movilidad de los dispositivos, etc. Esto plantea problemas de selección de dispositivos sensores, formación y mantenimiento de las redes, ahorro de energía para prolongar el tiempo de vida de la red y eficiencia en la medición y colección de datos sensoriales, los cuales son objeto de estudio en diferentes comunidades científica y tecnológica.

Recientemente se han desarrollado propuestas que atienden a algunos de esos problemas, principalmente en el aspecto de formación de la red. Por ejemplo en [*Van Dyck, 2002*] se presenta un algoritmo centralizado en el que las decisiones distribuidas se toman de manera local; su objetivo es fusionar decisiones locales dentro de una decisión global en el clúster; esta decisión es enviada vía multi-saltos a toda la red utilizando algoritmos de ruteo en una red ad-hoc móvil (MANET).

Otros trabajos proponen soluciones distribuidas basadas en la formación de grupos (clusters) utilizando auto-organización. En [*Lehsaini, 2010*] se presenta un algoritmo eficaz (***ECSA*** Efficient Cluster Based Self-Organisation Algorithm), donde se divide la red en clusters operando en forma jerárquica y se reduce el tiempo de servicio del líder del grupo para optimizar la energía. Un problema similar es abordado en [*Park, 2007*] donde el principal objetivo es la formación de grupos.

Con el mismo objetivo, en [*Olascuaga, 2011*] se propone una estrategia de agrupamientos (clustering) basada en auto-organización, la cual construye un backbone principal entre dispositivos móviles. La estrategia soluciona los problemas de segmentación y recuperación de redes. En su estrategia de auto-organización los nodos toman diversos roles como LEADER, MEMBER, GATEWAY Y BRIDGE, lo cuales realizan diferentes labores dentro de la red.

En [*Bandara, 2007*] también se propone la formación de grupos mediante un algoritmo denominado Generic Top-down Cluster and Cluster Tree Formation(**GTC**) Algorithm para formar un árbol de clústeres con la red. El algoritmo GTC es configurable, independiente de la topología de la red y no requiere de información de vecinos a priori, ni conciencia de localización o tiempo de sincronización. Más tarde, se incorporan en mecanismos para facilitar la intra e inter comunicación. Las VSNs (Virtual Sensor Network) soportan funcionalidades colaborativas, eficiencia de recursos

y redes multipropósito. El uso, formación, y mantenimiento de VSNs requieren la implementación de muchas funciones y protocolos, que tendrán que adaptarse a los cambios de la red: modificaciones, eliminaciones, adiciones, detección de múltiples VSN, combinación, divisiones y tendrán también que facilitar la comunicación con y a través de los VSNs. El esquema introducido en [*Bandara, 2008b*] proporciona una gran funcionalidad debido a que se organiza desde un conjunto de 3 escenarios distintos. Este funciona por áreas donde se detecta un evento y se propaga por los nodos vecinos hasta llegar al nodo raíz o nodo leader de ese grupo.

Otro trabajo reciente propone una nueva arquitectura implementando un algoritmo bio-inspirado ACO [*Salem, 2009*], donde se utiliza el método para encontrar una ruta óptima en una WSN multi-saltos. Los nodos trabajan de manera descentralizada, para recolectar datos, o detectar algún evento y llevar la información a través de comunicación multi-saltos.

Así como estos, encontramos muchos trabajos que están relacionados con el proyecto que se desarrollo en esta parte y se tomaron ideas de algunos de ellos para mejorar el nuestro.

En el presente trabajo nos hemos interesado en el desarrollo de una red de sensores inalámbricos en la que la localización de éstos no es conocida a priori. El funcionamiento de este tipo de redes requiere primeramente la formación de la red y posteriormente, la medición y recolección de la información sensorial. La solución propuesta incluye estas dos etapas. Para la formación de la red se adoptó un estrategia local basada en auto-organización para la formación de la red [*Olascuaga, 2011*]; el algoritmo se adaptó para su implementación en los sensores Freescale MC1321X. Posteriormente se diseñó un algoritmo para la medición y colección de los valores medidos; este algoritmo construye un árbol a partir del nodo concentrador (sink) sobre la red formada durante la primera fase, la cual tiene redundancias, y organiza el ruteo de la información sensorial hacia el sink, en una tarea simple de medición de temperaturas y cálculo distribuido de los valores máximo, mínimo y promedios de las mediciones.

La tesis está organizada como sigue. En el capítulo 1 se incluyen algunos conceptos básicos sobre redes y se presenta una revisión de trabajos actuales sobre las redes de sensores inalámbricas y algoritmos de ruteo. En el capítulo 2 se presentan los sensores de Freescale MC1321X, sus características y se describe el protocolo utilizado en el proyecto. En el capítulo 3 se presenta el algoritmo de auto-organización [Olascuaga, 2011] sobre el que se basa el proyecto para realizar la formación de la red. En el capítulo 4 se presenta la propuesta de un algoritmo para la propagación y recolección de información a través de un árbol de nodos formado en base a la estructura que presenta el algoritmo de auto-organización. Finalmente, en el capítulo 5 se presentan 3 casos de estudio que muestran el funcionamiento de los algoritmos, se describen los escenarios usados durante las pruebas y los resultados obtenidos.

CAPÍTULO 1

REDES DE SENSORES

Resumen. En este capítulo se presentan conceptos básicos del paradigma de las redes, tipos de redes y topologías. Se presenta también una revisión de métodos para el modelado de redes de sensores Ad-Hoc, procesos de comunicación y ruteo en este tipo de redes, incluidos aquellos que motivaron a la realización de esta tesis.

1.1 Conceptos Básicos

Las redes de sensores en estos tiempos tienen un gran auge en la utilización de la tecnología, por lo que cada vez hay que optimizar su rendimiento, manera de trabajar, desempeño y tiempo en que realizan una actividad.

A continuación incluimos un conjunto de definiciones básicas.

Una *red* es un conjunto de dispositivos autónomos interconectados, donde su principal trabajo es el intercambio de información [*Tanenbaum, 2003*].

Un *sensor* es aquel dispositivo que recolecta información del medio y será procesada para los fines que sean necesarios.

Una *red de sensores* es aquella que consta de dispositivos (sensores), tienen una arquitectura compuesta de un microprocesador, un sensor, un Transmisor/Receptor, batería, entre otras.

Estos dispositivos se unen para lograr un objetivo en común, transmitir información a toda la red. Un problema viene a dar lugar, encontrar cuál es la mejor ruta para transmitir la información, economizando y aprovechando los recursos de la red y la infraestructura, tanto de la red como de los sensores.

Un *Sink* es un nodo, el cual tiene asignada la tarea de recolectar la información de la red. Definición utilizada para este trabajo.

El *enrutamiento en las redes* de sensores es necesario en estos tiempos, ya que no todos los nodos se comunican de manera directa, y se realizan cada vez algoritmos más complejos que mucho dependen del tipo de red al que se apliquen.

El backbone son las principales conexiones troncales de la red.

Un *deadlock* en la red, es causado cuando se forma un ciclo, los buffers están llenos y un nodo espera por los demás, la información puede seguir avanzando pero sin realizar ninguna actividad más, es decir, no hay un subconjunto de estados alcanzable que provoquen un avance [*Tanenbaum, 2003*].

Un *Livelock* se da cuando 2 procesos están ejecutándose a la vez y al cambiar sus estados en respuesta al proceso paralelo al otro, ambos procesos quedan bloqueados y nunca terminan su ejecución. Para resolverlo se puede utilizar la exclusión mutua, es decir, sólo un proceso a la vez [*Ríos & Alvino, 2011*].

Codebase es un conjunto de archivos fuente, archivos de conFiguración, y generación de reglas que sirven como un repositorio, para los demos de BeeKit, templates, y otras aplicaciones generadas.

En esta tesis se presentan dos algoritmos, uno de ellos de auto-organización [*Olascuaga, 2011*] y el siguiente de ruteo, que trabaja con una búsqueda en amplitud y propagación de la información a partir de un Sink.

1.1.1 Generalidades

Las redes se encuentran determinadas por sus diferentes tamaños, formas y figuras. Existe una gran confusión entre una red de computadoras y un sistema distribuido. Un sistema distribuido esta constituido de un conjunto de computadoras independientes con un sistema consistente, un modelo que se presenta a los usuarios y una capa de software que se ejecuta sobre el sistema operativo denominado Middleware, éste contiene dicho modelo.

Mientras que en una red de computadoras no existe esta consistencia, modelo ni software. Los usuarios están expuestos a las máquinas reales y el sistema no realiza ningún intento porque las máquinas se vean y actúen de manera similar. El hardware y el software son completamente transparentes al usuario.

Existen aplicaciones de negocios, familiares, empresariales para una red de computadoras, que pueden consistir de diferentes sistemas operativos y su objetivo principal es el de compartir recursos.

La clasificación de las redes depende de muchos conocimientos y maneras de abordar el problema al que se enfrentan; las redes se clasifican por factores como *área de instalación (*LAN, MAN, WAN, etc.) *infraestructura* (Centralizadas o Descentralizadas), *puntos de conexión* (Topologías de la red), *requerimientos de aplicación*, etc.

1.1.2 Topologías De La Red

Existen diversas topologías (Figura 1.1) que se han utilizado a los largo de la vida de las redes y éstas se adoptan a la aplicación en la que se implementan.

La comunicación se realiza por medio de los links a los diferentes nodos que componen la red, determina su topología la manera de trabajar una vez que ya se establecieron sus links. La red puede ser cableada o inalámbrica.

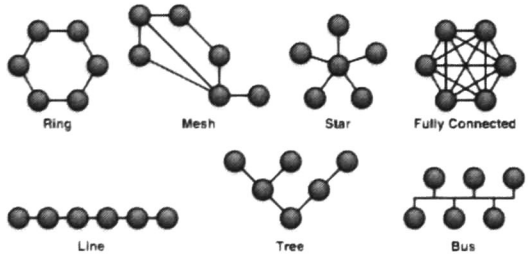


Figura 1.1 Topologías de Red [IPFire,2012]

A continuación se dan algunas características de algunas topologías de la red.

Red en Bus, en este tipo de red todos los nodos están conectados a un backbone, su señal no se reproduce y sólo transmite una computadora a la vez. La comunicación es de tipo broadcast, cuando

un nodo escucha, verifica si el mensaje es para él, si es así lo toma, sino lo ignora. Desventaja: si la comunicación falla la red se cae.

Red en Malla, un nodo está conectado a todos los demás (Full conected) o contiene diversas conexiones con nodos, ésta característica le permite ser una red redundante y si se interrumpen las comunicaciones o algún nodo falla la red no falla. No se requiere de un nodo central pero su desventaja es que requiere de mucho cableado.

Red en Estrella, está conectada por medio de dispositivo llamado "hub" ó "concentrador", que es el encargado de transmitir la comunicación al nodo destino. Si un nodo falla la red sigue funcionando, el verdadero problema es que si falla el hub la red se cae.

Red en Anillo, se le llama así por la manera en la cual está conectada. El paso de mensajes en esta red es por medio de un token, el nodo que lo posea es el transmisor en ese momento. La información viaja en una sola dirección a través del anillo; si se rompe un enlace la red se cae, en caso de que se quiere tener redundancia, se agrega otro cableado (Anillo doble).

Redes Híbridas, son una combinación de 2 o más redes de las anteriores con el fin de mejorar la topología y redundancia de la red; un ejemplo de éstas, son las redes en árbol que son una mezcla de las redes en estrella y la de bus, etc. Estas redes suelen tener un coste de mantenimiento elevado por la estructura que adoptan.

1.1.3 Arquitectura De Las Redes

La arquitectura de la red se encarga de organizar como es la comunicación entre las diferentes partes que componen una empresa ya sea entre pisos, departamentos, edificios, entre otros. Se toman en cuenta las características que necesita la red y su estado general para garantizar que no falle la transferencia de información.

1.1.3.1 Distribución Geográfica

Este tipo de distribución toma en cuenta el estado físico la red, el servicio que provee y como se distribuye. Se evalúan aspectos como switch's de la capa de acceso, capa de distribución de la aplicación y las computadoras de los usuarios.

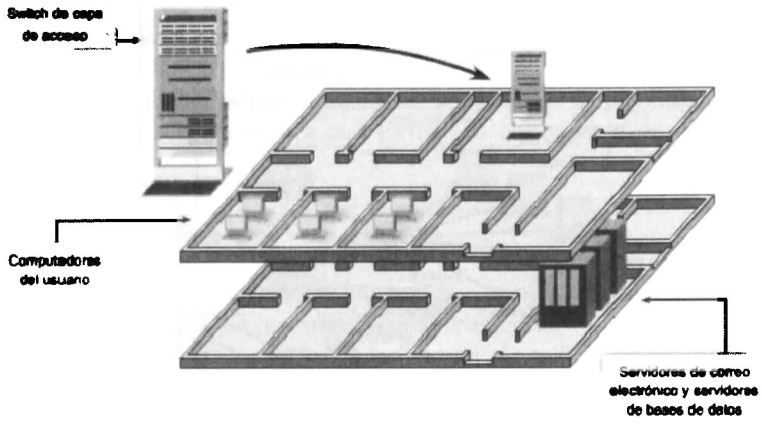


Figura 1.2 Distribución Física [CISCO, 2011]

En la distribución física (Figura 1.2) se tienen las siguientes clasificaciones:

LAN (*Redes de Área Local*) son redes de propiedad privada que se encuentran en un solo edificio o en un campus de pocos kilómetros de longitud. Se utilizan ampliamente para conectar PC y estaciones de trabajo, en oficinas de una empresa, fábrica, negocio, etc., para compartir recursos. Están restringidas por tamaño y tiempo de transmisión, que en el peor de los casos es limitado y conocido. El hecho de conocer este límite permite utilizar ciertos tipos de diseño, lo cual no sería posible de otra manera.

MAN (*Red de Área Metropolitana*) son redes que se encuentran dentro de una ciudad, pueden tener alcances de más de 4 kms y están conectadas con buses unidireccionales; un ejemplo de estas redes es la TV por cable.

WAN (*Red de Área Amplia*) abarcan una gran área geográfica con frecuencia un país o un continente. Contiene un conjunto de máquinas diseñadas para programas (aplicaciones) de usuario "host". Estos hosts están conectados por una subred de comunicación, que se encuentra dentro de las PC de los usuarios, mientras que las compañías telefónicas o proveedores de servicios de internet operan la subred de comunicación. La función de una subred es llevar mensajes de un *host* a otro [*Tanenbaum, 2003*].

1.1.3.2 Distribución Lógica

En la distribución lógica cada capa está compuesta por diversas actividades, las cuales se componen de pasos o tareas (Figura 1.3). Cada capa se explica a continuación:

La capa de acceso: es la encargada de realizar la conexión de los dispositivos con la interfaz final, tales como impresoras, pc's, teléfonos, router's, switch's, hub's; cuyo objetivo es aportar la conexión al resto de la red.

La capa de distribución: agrega los datos recibidos a los switch's de la capa de acceso antes de que sean transmitidos a la capa de núcleo. También es la encargada de administrar el flujo de redundancias en la información.

La capa de núcleo: es el backbone principal de la red, su función es realizar la transferencia de la información tan rápido como le sea posible.

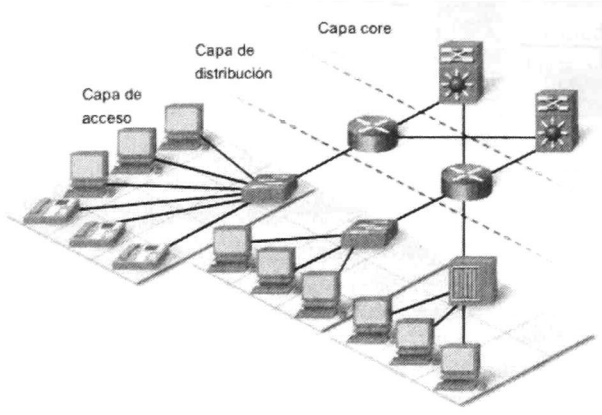


Figura 1.3 Capas de la Red Jerárquica [CISCO, 2011]

1.2 Redes Inalámbricas

Las redes inalámbricas permiten la interconexión entre dos o más puntos, nodos o estaciones, por medio de ondas electromagnéticas que viajan a través del espacio y llevan información de un lugar a otro. Para lograr el intercambio de información existen diferentes mecanismos de comunicación o protocolos que establecen reglas que permiten el flujo confiable de información entre nodos [*Tanenbaum, 2003*].

Algunos de esos protocolos son *WiFi, WiMax, Bluetooth, etc*; éstas permiten la comunicación dentro de una misma localidad y *WiMax* establece la conexión satelital.

Como primera aproximación, las redes inalámbricas se pueden dividir en tres categorías principales:

⇒ *Interconexión de sistemas* se refiere a la interconexión de componentes de una computadora que utiliza radio de corto alcance.

⇒ *LANs inalámbricas (***WLAN***)* son las redes inalámbricas de corto alcance, permiten la comunicación con el único requisito de que los dispositivos involucrados se encuentren dentro del alcance de la red. Cada computadora tiene un módem de radio y una antena mediante los que se puede comunicar con otros sistemas.

⇒ *WANs inalámbricas* **(WWAN)**, son redes que conectan a diferentes WLAN, éstas pueden conectarse a distancias mayores e incluso áreas donde no tiene acceso el ser humano.

La interacción de redes inalámbricas podría permitir la recolección de información, monitoreo útil para temas ambientales, desarrollo agrícola, prevención de desastres o hasta el uso en una empresa para la transferencia de información.

1.2.1 Redes Distribuidas

Las redes inalámbricas con administración descentralizada le permiten a cada computador administrar su información con respecto a la información que reciben de las conexiones con él. Cuentan con:

⇒ Dispositivos autónomos.
⇒ Configuración individual de funcionalidades.
⇒ Cada nodo envía su información a los demás.
⇒ Adecuado en aplicaciones que no pueden ser centralizadas.
⇒ Las aplicaciones son más veloces, ya que no cargan con toda la información de la red.
⇒ Evitan el uso de routers o de puntos de acceso que gestionen la comunicación.
⇒ Se utilizan en ambientes hostiles, etc.

Ningún nodo posee la capacidad de filtrar qué información circula por la red ni cuál reciben los demás nodos de la red (Figura 1.4). Por esa razón se desarrollan algoritmos cada vez más robustos que son capaces de asegurar que la información enviada sea la suficiente necesaria para realizar una tarea y los resultados obtenidos sean competitivos a una red con comportamiento global.

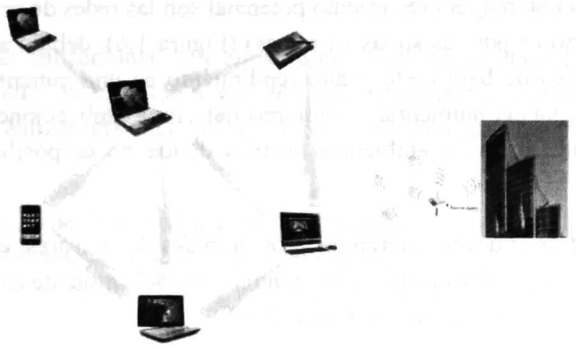


Figura 1.4 Red Inalámbrica Distribuida

1.2.2 Redes Centralizadas

Las redes inalámbricas (Figura 1.5) con administración centralizada son ideales para empresas que implementarán una red con varios Access Point, con esta tecnología podrán:

⇒ Administrar centralizadamente todos los AP que se tengan.
⇒ Los AP actúan inteligentemente regulando automáticamente su potencia dependiendo de la recepción de los equipos contiguos.
⇒ Permite Roaming dentro de la misma red de la empresa.
⇒ Facilita el análisis de cobertura de la red inalámbrica.
⇒ Localización precisa de los recursos

Figura 1.5 Red Inalámbrica Centralizada

1.3 Redes De Sensores

Debido al crecimiento acelerado en las tecnologías se deben encontrar nuevas metodologías que se adapten a los sistemas cambiantes y sistemas más flexibles, escalables y seguros.

Una de las áreas con mayor crecimiento potencial son las redes de sensores inalámbricas (**WSN** *Wireless Sensor Network* por sus siglas en inglés) (Figura 1.6), debido al hecho de que pueden ser utilizadas con modelos de bajo coste y alto rendimiento en una inmensidad de aplicaciones tales como seguridad, vigilancia ambiental, monitoreo del clima, aplicaciones inteligentes, domótica y también podemos utilizarlas en ambientes hostiles donde no es posible el acceso a personas y muchos más.

En la actualidad se conocen diferentes tipos de redes de sensores, estas dependen de los fines de la aplicación para la que fue diseñada, las condiciones del ambiente en donde se vaya a implantar y los recursos con los que se cuenta [*Ibnkahla, 2008*].

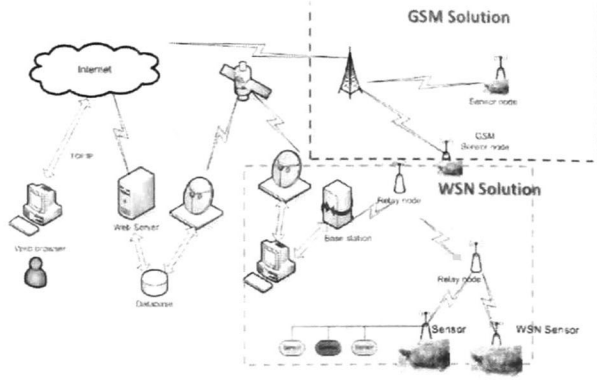


Figura 1.6 Redes de Sensores inalámbricas [Ibnkahla, 2008].

Las WSN's requieren de diversas disciplinas para su aplicación. Una de las motivaciones de este trabajo es la auto-organización de la red en la que por medio de roles se realiza la formación de clústeres, se reorganizan, agregan y eliminan nodos durante el proceso [*Olascuaga, 2011*]. Por otro lado el ruteo o propagación de la información es por medio de asignación de niveles [*Tao, 2011*], por los cuales se transmite la información. Se toman en cuenta escalabilidad, confiabilidad, energía consumida y procesos de comunicación.

1.3.1 Nodos Sensores

Un nodo sensor es aquel dispositivo que nos permite realizar una extracción de información del exterior, enviarla a través de otros dispositivos o a uno final, procesarla y tomar decisiones en base a la información obtenida. Algunas de las aplicaciones esenciales de los nodos son:

⇒ Meteorología (monitoreo ambiental)
⇒ Aplicaciones médicas
⇒ Aplicaciones militares
⇒ Alarmas
⇒ Domótica, etc.

Se realizan diversas aplicaciones con ellos, tales como mediciones de presión, temperatura, humedad, movimiento, etc., se manejan nodos de forma solitaria o en conjuntos para formar redes. Estos sensores pueden realizar una sola aplicación o combinarlas para que realicen múltiples tareas a la vez.

Debido al gran avance de la tecnología, en la actualidad se utilizan sensores inalámbricos; su interconexión permite grandes facilidades, pero a su vez están delimitados por las características del propio sensor, es decir, consumo de energía, programación, funcionalidad, tiempo de vida, etc.

La energía en los sensores se aprovecha de 3 maneras: *comunicación, el procesamiento* y *sensado*, el ahorro de está en una implementación de red de sensores es de vital importancia; se utilizan hoy en día diferentes métricas para su ahorro, conservando los diferentes factores que hacen útil la vida de la red en los fines para los cuales fue diseñada.

Los nodos sensores dependen tanto de su tipo como de su funcionalidad, actúan de diferentes maneras en un mismo ambiente y pueden utilizar técnicas de ahorro de energía como: apagarse cada cierto tiempo, reducir la frecuencia de operación, etc.

El software encargado del control del sistema suele utilizar uno de los tres estados siguientes para el sensor:

⇒ ***Activo***: se encuentra en este estado cuando funciona continuamente aunque no se le soliciten peticiones, esta manera de trabajar tiene la ventaja de disponibilidad de tiempo para atender alguna petición, lo que conlleva a una desventaja ya que gasta energía innecesaria.

⇒ ***Proactivo***: el nodo no trabaja siempre sólo cuando es necesario o cada determinado tiempo, así permite el ahorro de energía. Este tipo de sensor utiliza energía sólo cuando la necesita suspendiendo toda actividad al no estar activo y permite que el tiempo de vida se prolongue.

⇒ ***Inactivo o monitor***: el nodo monitorea el entorno y el canal de comunicación en espera de instrucciones pero no transmite ningún dato, [*Arreola, 2012*].

1.3.2 Métodos De Formación De Redes Ad-Hoc

1.3.2.1 Redes De Sensores Centralizadas

En [*Van Dyck, 2002*] se enfoca en el problema de detección distribuida, su objetivo es fusionar decisiones locales dentro de una decisión global en el clúster; esta decisión local es enviada vía multi-saltos a toda la red utilizando algoritmos de ruteo en una red ad-hoc mobile (MANET).

Para el consumo de la energía y baja probabilidad se requiere que se minimicen el número de bits enviados a través de nodos locales. Al mismo tiempo la probabilidad de error global también se debe minimizar para que permita lograr la misma probabilidad de error como en un procesador central. Esto se realiza por múltiples iteraciones llamadas parleys.

Este trabajo es una extensión del método de Swaszek and Wallet [*Swaszek and Wallet, 1995*].

Algoritmo Descentralizado

En LCA se utilizan tramas TDMA para la comunicación, el número de sensores es fijo y una parte se asigna a cada sensor. El algoritmo esta compuesto de dos etapas principales:

⇒ *La formación de los clúster*: consiste en construir una matriz de conectividad. Se utilizan dos frames; el primero de ellos envía paquetes que contienen los nodos y escuchan la información de un frame anterior y en el segundo cada nodo hace broadcast a su fila completa.
⇒ *El enlace de clústers usando gateways*. comienza cuando se termina la transmisión anterior; se utiliza la matriz para clasificarse como: Cluster Head, Gateway u Ordinary.

Se forma un backbone de conexión con los CH pasando por los gateways si es necesario.

Soft-decision Parley Algorithm

✓ Cada uno de los N sensores mide un fenómeno y realizan una función de probabilidad $\Lambda(r_i)$; para cada sensor de $i=1, \ldots, a\ N$, el óptimo centralizado esta dado por:

$$\prod_{i=1}^{N} \Lambda(r_i) \underset{H_0}{\overset{H_1}{<>}} \lambda \text{ donde } \lambda \text{ es el umbral.}$$

La decisión descentralizada de cada sensor en cada iteración esta dada por:

$$\Lambda(r_i) \underset{u_{i,m}=0}{\overset{u_{i,m}=1}{<>}} \lambda_{i,m}$$

Donde $u_{i,m}$ es la decisión difícil y $\lambda_{i,m}$ es el umbral local del sensor en la iteración m.

- ✓ Se dice que se elige un consenso si todos los nodos eligen la misma hipótesis. El producto de todas las funciones de probabilidad locales es tan grande o menos que la el producto del umbral local.
- ✓ "Si el umbral de cada iteración satisface $\prod_{i=1}^{N}\lambda_{i,m} = \lambda$ alcanza una decisión de consenso que se iguala a la decisión centralizada"

- ✓ El umbral óptimo local está determinado por:

$$\lambda_{i,m+1} = \left(\lambda \prod_{k=1}^{N} \frac{\Pr(S_{k,m} \leq \Lambda(r_k) \leq t_{k,m} | H_o)}{\Pr(S_{k,m} \leq \Lambda(r_k) \leq t_{k,m} | H_1)} \right)^{1/N}$$

$$* \frac{\Pr(S_{i,m} \leq \Lambda(r_i) \leq t_{i,m} | H_1)}{\Pr(S_{i,m} \leq \Lambda(r_i) \leq t_{i,m} | H_0)}.$$

$s_{k,m}$ y $t_{k,m}$ son el valor mínimo y máximo de $\Lambda(r_k)$, dado por las decisión es difíciles $u_{k,n}$, $n=1, \ldots, m-1$. λ es el valor de la detección centralizada.

- ✓ Se elijen los valores de probabilidad mínimos y máximos de cada función de probabilidad de los nodos y se compara con el umbral local que es calculado utilizando probabilidades a posteriori, determinada numéricamente por las funciones de probabilidad de otros nodos entre el mínimo y máximo valor; se realiza una soft decisión y se envía por broadcast a todos los nodos.
- ✓ Se selecciona una hipótesis con un bit de cuantificación y se guarda un valor confidencial para su hipótesis. Si el 80% está de acuerdo entonces el proceso de parley se detiene.

Robustness: Sensor Performance versus Distance

Para calcular la curva gaussiana se requiere de un punto de acceso donde se realicen las comparaciones y probabilidades del procesador central. Un error será la cuota inferior del desempeño de cualquier algoritmo de detección. Las hipótesis están dadas por:

$$\Pr(r) = \frac{1}{\sqrt{2\pi\sigma}} \exp \frac{-(r \pm s)^2}{2\sigma^2}$$

Donde s es la hipótesis uno (H1) y –s es la hipótesis cero (H0). Alternativamente se pueden cambiar los valores de manera que para la HO sea 0 y para H1 sea m=2m. La función de probabilidad está dada por:

$$\Gamma(R) = \frac{\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma}} \exp \frac{-(R_i - m)^2}{2\sigma^2}}{\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma}} \exp \frac{-R^2}{2\sigma^2}}$$

Para calcular la atenuación del medio se utiliza $m' = \frac{m}{ad^{\beta}}$ donde d es la distancia física del emisor al sensor. α, β son factores de atenuación que determinan la propagación del medio.

Las hipótesis se pueden mostrar de la manera siguiente:

$$H0 = l = N(0,1)$$

$$H1 = l = N\left(\sqrt{N}\frac{m}{\sigma},1\right)$$

Donde$d = \sqrt{N\,m/\sigma}$ y n dependen del costo y probabilidades a priori. Para calcular la probabilidad de error, se asume que las probabilidades de las probabilidades se conocen a priori.

El algoritmo se basa principalmente en decisiones locales, para tomar una decisión global utilizando decisiones soft y una robusta comunicación.

1.3.2.2 Redes De Sensores Inalámbricas Distribuidas (DWSN)

Una red de sensores distribuida inalámbrica consiste en un conjunto de sensores situados a una cierta distancia donde pueden ser manipulados; no dependen del tipo de red, ni de propiedades de la misma.

Estas redes son las más utilizadas en la actualidad por las características que posee. El problema de la distribución tiene diversas estrategias y se basan primordialmente en que el control de la red no depende de un solo dispositivo, sino que cada uno contiene ciertas responsabilidades para actuar y tomar decisiones en base a su estado; además comparten la información sólo con un limitado número de sensores que se encuentran en su rango de alcance.

1.3.2.2.1 Redes de Sensores Basadas En Auto-Organización

Este tipo de redes permiten que en la interacción se conduzca a la aparición de funcionalidades que en una sola unidad no son capaces de ser explicadas. Estos algoritmos trabajan con redes ad hoc y aprovechan su carencia de información de la red global, trabajan sólo con la información que tienen de sus vecinos. Este tipo de redes provee muchas ventajas ya que aumenta la escalabilidad de la red, refiriéndose a la incorporación de nuevos nodos y modificación de los mismos durante su funcionalidad.

1.3.2.2.1.1 ACO (Antz Clustering Optimization)

En el campo de Redes de Sensores Inalámbricas (WSNs) es aplicable utilizar los algoritmos que son bio-inspirados, ya que ayudan a resolver los problemas que surgen en las WSNs; uno de ellos es el ACO que muestra un excelente desempeño en este tipo de redes. En él se consideran factores como SO, calidad del enlace, nivel de energía, pérdida de tasa de transmisión, etc., para tomar decisiones y ayudar a la convergencia del algoritmo.

En [*Colonies, 91*] una sola ant tiene capacidades simples. El comportamiento completo de la red es altamente estructurado.

Se le llama *ant* a la interacción entre los agentes; *ant-algoritmo* al algoritmo que se define; se compara el trabajo realizado con la estrategia del algoritmo TSP (*Travelling Salesman Problem*). Se definen principalmente 3 conceptos para dicho algoritmo:

⇒ *ANT-density:* se llama así cuando una ant va de i a j y deja Q_2 cantidad de feromonas por cierta unidad de longitud.

$$\Delta\tau_{ij}^{k}(t,t+1) = \begin{Bmatrix} Q_2 \; si \; la \; k-th \; ant \; va \; de \; i \; a \; j \; entre \; t \; y \; t+1. \\ 0 \; de \; otra \; manera \end{Bmatrix}$$

⇒ ***ANT-quantity***: es una cantidad Q_1 que deja una ant cada vez que va de una ciudad i a una ciudad j.

$$\Delta\tau_{ij}^{k}(t,t+1) = \begin{Bmatrix} \frac{Q_1}{d_{ij}} & si\ k-th\ ant\ va\ de\ i\ a\ j\ entre\ t\ y\ t+1 \\ 0 & de\ otra\ manera \end{Bmatrix}$$

⇒ ***ANT-cycle***: Se actualiza su valor al dar una vuelta completa y esta dado por:

$$\Delta\tau_{ij}^{k}(t,t+n) = \begin{Bmatrix} \frac{Q_3}{L^k} & si\ la\ \ k-th\ ant\ \ ha\ completado\ un\ tour \\ 0 & de\ otra\ manera \end{Bmatrix}$$

Donde Q_3 es una constante y L^k es la longitud del ciclo de la ***k-th*** ant.

En el caso de ants, el medio por el cual ellos deciden comunicarse y toman la decisión de formar caminos es mediante el rastreo de feromonas (*pheromone trail*). Si una segunda ant busca un camino dado por probabilidad, así esta decide si toma el camino anteriormente marcado o encuentra uno mejor disponible; entre más ants pasen por el mismo camino más probabilidad hay de que las demás recorran el camino hacia el destino.

El comportamiento colectivo que emerge es forma de un comportamiento auto catalítico o alelo mimesis, que están dados por la mayor cantidad de ants seguidoras de una pista y en el que el camino se vuelve atractivo para las ellas (Figura 1.7).

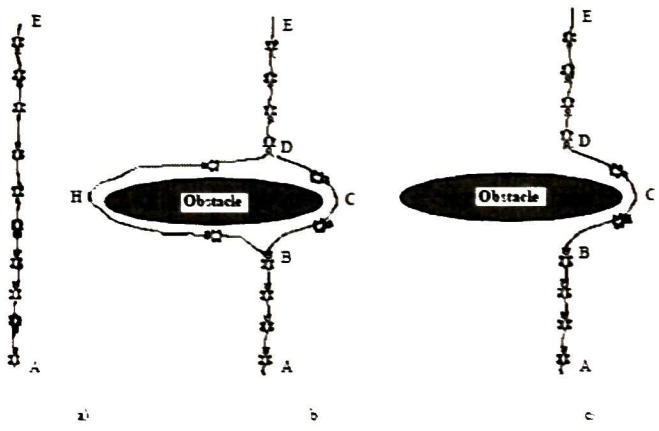


Figura 1.7 Se presenta un ejemplo de cómo alelo mimesis puede llevar a la identificación del camino más corto aun con un obstáculo.

Dado un conjunto de n Ciudades, el TSP comienza a buscar el camino mínimo visitando cada ciudad una sola vez. Tenemos $b_i(t)$ $(i=1, \ldots, n)$ es el número de ants en una ciudad i en un tiempo t. El total de ants esta determinado por $m = \sum_{i=1}^{n} b(t)$.

Llamamos $path_{ij}$ al camino más corto entre una ciudad i y otra ciudad j. Para obtener esta se utiliza la distancia Euclidiana entre dos puntos:

$$dij = \sqrt{(x_1^i - x_1^j)^2 + (x_2^i - x_2^j)^2}$$

Se tiene $\tau_{ij}(t+1)$ como la intensidad de la pista en el $path_{ij}$ en un tiempo $t+1$, dado por la formula:

$$\tau_{ij}(t+1) = \rho\,\tau_{ij}(t) + \Delta\tau_{ij}(t,t+1)$$ donde ρ es el *coeficiente de evaporación.*

$$\Delta\tau_{ij}(t,t+1) = \sum_{k=1}^{m} \Delta\,\tau_{ij}^{k}(t,t+1)$$

Donde $\Delta\tau_{ij}^{k}(t,t+1)$ es la cantidad por unidad de longitud de la pista (pheromone in real ants) puesta sobre el $path_{ij}$ por la *k-th* ant entre el tiempo t y $t+1$.

Llamamos visibilidad de cantidad a $\eta_{ij}=1/d_{ij}$. La probabilidad de probabilidad de transición de una ciudad i a una ciudad j esta definida por:

$$p_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}]}{\sum_{j=1}^{n}[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}$$

Donde α y β son parámetros que permiten a un usuario una pista contra la visibilidad, ciudades cercanas deben ser altamente probables y la intensidad de la pista dice que si un $path_{ij}$ tiene mucho tráfico entonces es altamente deseable.

Cada ant tiene una estructura de datos llamada *tabulist* que memoriza la ciudad que ya se visito en un tiempo t, para así evitar que se hagan dobles visitas y se visiten de nuevo antes de terminar con todas las ciudades.

La solución converge en un tiempo exponencial utilizando procesos auto-catalíticos, en los cuáles se obtienen resultados favorables al realizar ciclos, pero la transmisión y cálculo de esas mediciones no juzga la batería limitada con la que los sensores cuentan. Además de que no está bien definido a qué clase de problemas se aplica este trabajo.

ACO asegura que llega a una convergencia cuando su probabilidad es 1. Esta convergencia se obtiene por medio de espacio de valores posibles de acuerdo al escenario con el que se trabaja y tiene diversas aplicaciones, desde la realización de grupos, manera de comunicación, hasta el ruteo de diferentes redes, etc.

1.3.2.2.1.2 Basadas en Clustering

En el trabajo de [*Lehsaini, 2010*] la SO se trabaja mediante un algoritmo eficaz (***ECSA*** Efficient Cluster Based Self-Organisation Algorithm), de manera que se divide la red en clústeres que trabajan de forma jerárquica.

La red de sensores inalámbricos se ve como un grafo no dirigido $G = (V,E)$, donde V son los nodos. $E \subseteq V^2$ es el conjunto de aristas que permiten la comunicación: una arista $e=(u,v)$ pertenece a E si permite la transmisión de mensajes de u a v y viceversa.

Cada sensor tiene un identificador $Node_{id}(u)$. El conjunto de vecinos de u están representados por $N_1(u)$. El tamaño del conjunto de los vecinos se conoce como el grado de u, $\delta_1(u)$. La k-densidad de un nodo u representa el radio entre el número de enlaces en el k-salto de vecindario.

El tamaño de los clúster esta dado entre 2 umbrales:

$Tresh_{Lower}$ que representa el número mínimo de sensores que puede tener un clúster.
$Tresh_{Upper}$ es el número máximo de sensores que pueden pertenecer a un clúster.

Excepto en los casos donde el valor es más bajo que $Thresh_{Lower}$, y además el CH es el encargado de decidir si se agrega un nodo nuevo basado en su capacidad sin afectar a los miembros.

La información es transmitida a los CH quienes se encargan de retransmitir a un Sink, los vecinos se encuentran a 2 saltos. El proceso de elección de CH se realiza cada cierto tiempo que es llamado $Time_{Service}$ para optimizar la distribución de energía.

Cada nodo se identifica con el siguiente vector: ($Node_{Id}$, $Node_{CH}$, *Weight*, *Hop*, *Size*, $Thresh_{Lower}$, $Thresh_{Upper}$).Cada nodo es responsable de mantener una tabla $Table_{Cluster}$ donde la información local de los miembros del clúster se almacena y contiene ($Node_{Id}$, $Node_{CH}$, *Weight*). Se realiza todo el proceso en 2 fases que son:

⇒ *Set up phase*, en la que se eligen los CH's y los miembros del clúster mediante una serie de mensajes (Figura 1.8)

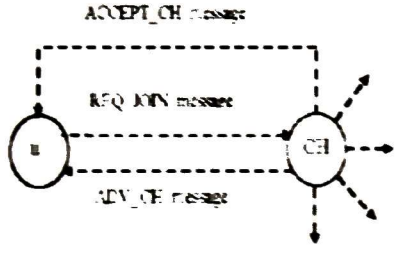


Figura 1.8 Fase Set up

⇒ *Re-affiliation phase*, es posible que haya clúster con un tamaño menos a $Thresh_{Lower}$, en esta etapa se trata de minimizar el número de clúster y reorganizarlos (Figura 1.9).

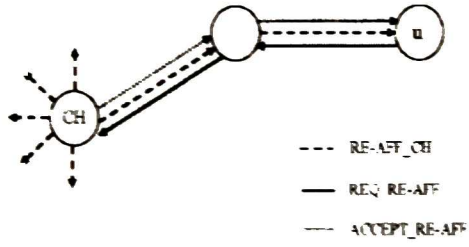


Figura 1.9 Fase de Re-afiliación

Cluster maintenance, se dispara cuando se pierde el CH o se termina la batería. Es similar a la etapa de *Set up* entre los miembros del clúster.

El tiempo de convergencia del algoritmo depende de la cantidad de nodos y la manera en que se realicen los clústers. Este trabajo contiene un cambio excesivo de mensajes, debido a ello sólo tomamos ideas que nos facilitan nuestro trabajo. Unas de ellas son reconfiguración, utilización de

una estación base (Sink) y la manera en que se realiza la auto organización. Este trabajo presenta muchas ventajas, sin embargo, no nos ayudan del todo para abordar la problemática de nuestro proyecto.

En el trabajo de [*Park, 2007*] el principal objetivo es la optimización en la formación de clúster maximal (Figura 1.10).

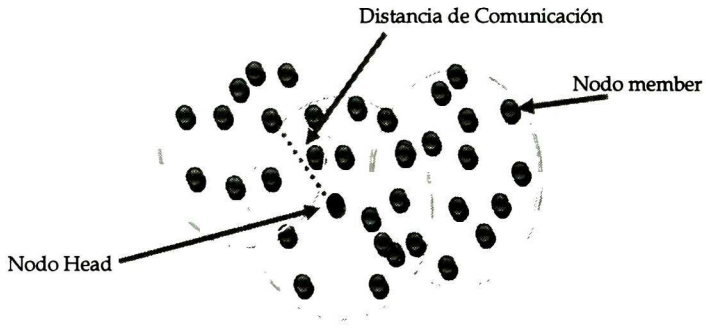
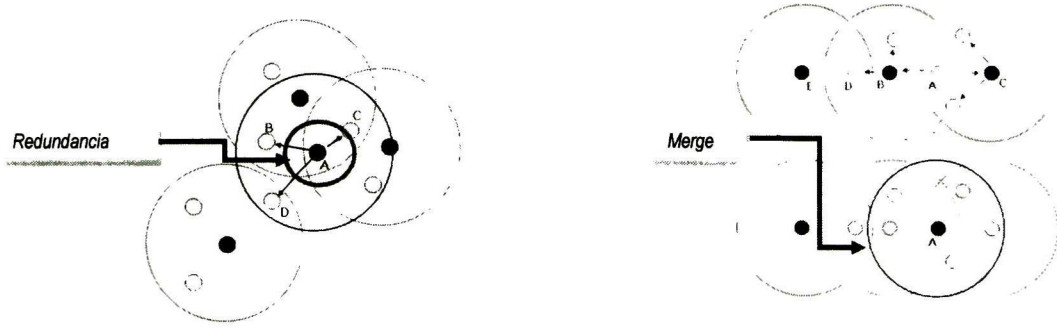


Figura 1.10 Formación de un clúster

En este algoritmo se utilizan 2 procesos (Figura 1.11):

- *Merge* se usa para sobreponer un clúster en un área donde hay muchos.
- *Dimises* se utiliza para eliminar redundancias en CH.



a) Redundancia b) Merge

Figura 1.11 Ejemplos de casos que pueden ser optimizados

El proceso llamado *Dismiss* se trata de la siguiente manera. Comienza con el nodo CH que esta duplicado o redundante, este envía un mensaje broadcast con la siguiente información: *Is it all right if i give up being a head of you*?, los nodos que tienen conexión a él tendrán que verificar sus circunstancias, si encuentran algún otro CH cerca de su rango de transmisión envían un mensaje de OK. Si el nodo CH recibe de todos sus hijos mensajes OK se realiza el proceso de Dismiss y los nodos hijos entonces se unen a otro líder (Figura 1.12).

a) Envió de mensaje Broadcast del nodo redundante A b) Proceso Finalizado

Figura 1.12 Ejemplo del proceso Dismiss

En el caso de *merging* de dos clústers; el proceso comienza con un nodo member disparado por su reloj local a los nodos que se encuentren dentro de su rango de alcance (*Figura 13(a)*), si el nodo reconoce que tiene más de un CH en su rango de comunicación, el nodo que se disparo envía el siguiente mensaje: *Is it posible for me to be a head node instead of you two*?, así los CH's que reciben el mensaje envían un broadcast a sus miembros con su ID y el Id del nodo que quiere ser CH, (*Figura 13 (b)*), si el nodo que quiere ser CH's recibe de ambos CH's una confirmación, en este ejemplo el nodo A se convertirá en CH y los CH´s se convertirán en miembros de él (*Figura 13 (c)*). Reconfigurándose la red. (*Figura13 (d)*).

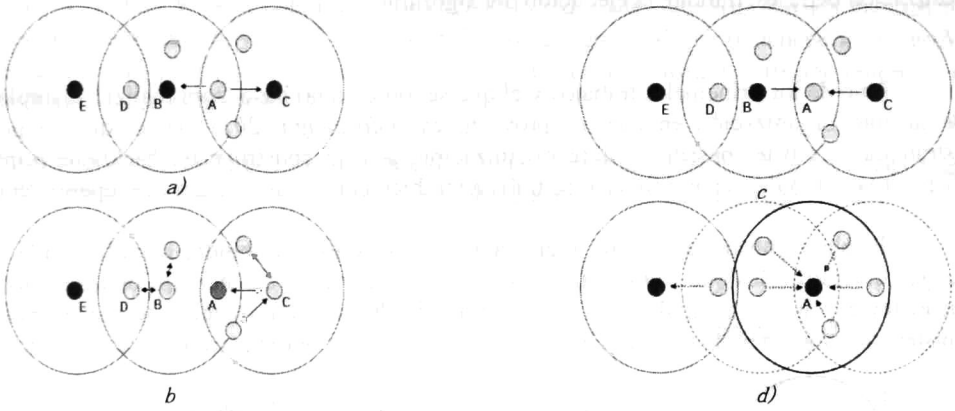


Figura 1.13 Proceso de Merging

Un nodo es el encargado de realizar el algoritmo una vez que este activa toda la red, este nodo se ejecuta de manera aleatoria.

La red se modifica por factores como:

Un nodo cambia de miembro a CH o viceversa.
Un nodo cambia a CH.
Un nodo es inicializado.
Un nodo muere.

Se utiliza un proceso de utilización de cache para evitar la sobrecarga de mensajes. La optimización de la red se realiza en 2 fases: *Survey* y *Execution*.

Survey phase. en esta fase un nodo investiga a los nodos que se encuentran en su rango de comunicación con el fin de saber si puede realizar alguno de los procedimientos: merge o dismiss. basado en las replicas de los nodos. Cuando no se realiza ningún cambio se utiliza la memoria cache ya que almacena la información conveniente de sus nodos vecinos. para la siguiente notificación.

Execute phase basado en la fase survey realiza la optimización.

Este algoritmo mostro en algunas pruebas mejores resultados que **ACE** (*Algorithm for Cluster Establishment*) [*Chan. 2004*] y **SOS** (*Self-Organizing Sensor*) [*Shin. 2006*] que son dos algoritmos de SO más.

El desempeño de ACE se basa en dos parámetros, los cuales son introducidos manualmente y ajustados de acuerdo al tamaño y forma de la red de sensores para realizar la formación de clústers.

SOS en cambio tiene una debilidad estructural y siempre necesita en enlace con todos los nodos para formar la red.

Este algoritmo que se presenta es una mejora sin duda a los dos anteriores, sin embargo contiene algunas deficiencias, como el cambio excesivo de mensajes en las etapas de merging y dismiss. No toma en cuenta la energía residual de un CH hasta que se muere o se agrega un nuevo nodo. Sin duda se toman ideas del algoritmo para la reconfiguración de todos los nodos y la realización de fases, durante la ejecución del algoritmo.

Uno de los principales trabajos y el que se tomo como base para realizar la implementación de la auto organización en nuestro proyecto es [*Olascuaga. 2011*] en el que se propone una estrategia de clúster basada en auto organización, la cual construye un backbone principal entre dispositivos móviles, en el proyecto se trabaja también con segmentación y recuperación de redes.

El algoritmo trabaja sobre interacciones locales, es distribuido, reconfigurable. Los nodos juegan diversos roles que son dinámicos durante su tiempo de ejecución. Los resultados obtenidos en el trabajo son comparados con el algoritmo de Prim. La formación se realiza por medio de clústers que son liderados por un CH (*Leader*), permite que se optimicen los recursos de la red.

Este trabajo transmite un número mínimo de mensajes tomando en cuenta la limitación de energía en estas redes. Los roles que juegan los nodos son los siguientes:

- ⇒ *MEMBER*: es el rol más simple y pertenecen a un solo líder.
- ⇒ *LEADER*: es el encargado de un clúster y quien hace posible la comunicación entre diferentes clústers de la red.
- ⇒ *GATEWAY*: es el responsable de la comunicación entre clústers a través de los líderes.
- ⇒ *BRIDGE*: realiza la comunicación entre diferentes segmentos de la red, a través de miembros de diferentes clústers que se encuentren en su rango de transmisión.

El leader se elige de acuerdo a diversos parámetros, tales como energía residual, cantidad de memoria, capacidad de procesamiento, número de vecinos, calidad de enlace, etc., que son calculados localmente por cada nodo para realizar esta elección entre los miembros del clúster al que pertenecen.

Para este trabajo se toman en cuenta las siguientes restricciones:

⇒ Cada agente tiene un identificador único.
⇒ Cada nodo conoce su vecino a un salto.
⇒ Un nodo puede dejar, llegar o moverse dentro de la red.
⇒ Cada nodo puede ajustar su poder de transmisión de acuerdo al rol de vecino para conservar la energía.
⇒ Cada nodo puede escuchar la información que se transmite por la red, para realizar resguardo de información y reducir paso de mensajes.
⇒ Cada nodo u mantiene una tabla de vecindario y un peso (*weight*) que es igual a el producto de la energía residual e y el número de vecinos N (*weight*= $e_u * N_u$).
⇒ Los nodos no conocen su posición
⇒ Se encuentran en un plano de 2 dimensiones.

Cada uno de los nodos ejecuta de manera independiente el algoritmo, su comportamiento aparenta al de un algoritmo global. Al inicio del algoritmo cada nodo tiene asignado el rol de leader, surge entonces conflicto que se resuelve con la función llamada *Solve_Conflict()*, ésta se encarga de definir quién se queda con el rol de leader.

En dado caso de que un nodo tenga más de un vecino leader, este se convierte en gateway, para que se logre la comunicación entre diversos grupos. Un rol de bridge se elige cuando se tiene un nodo $u \in$ V, donde V es el conjunto de nodos en la red, dado que $N(u)$ son los vecinos de u, Role(u) es el rol del nodo u, y $id_g(u)$ es el identificador del grupo al que pertenece u, si $\exists v \in N(u)$: Role(v)=member, $id_g(v) \neq id_g(u)$ y $\nexists w \in N(u)$: Role(w)= Gateway o Role(w)=Bridge entonces u se convierte en un bridge. (Figura 1.14).

La conservación de la energía depende del rol de cada nodo. Los roles de los nodos pueden cambiar si se presenta alguno de estos casos: un nodo se agregue, se mueva o deje la red.

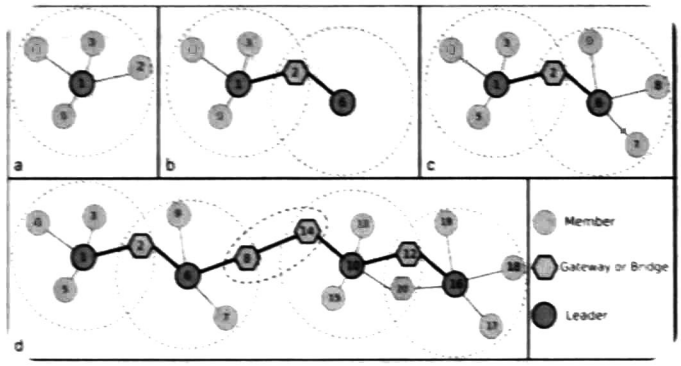


Figura 1.14 Proceso Auto-Organización

Los nodos son móviles. Los nodos que no están en funcionamiento se apagan para conservar la energía. Cuando en un mismo tramo existen dos o más gateways que conectan al mismo par de leaders, se realiza un procedimiento, el cuál decide que nodo es el más capacitado para el rol de gateway y los demás nodos se apagan para conservar la energía.

⇒ Los members solo necesitan comunicarse con los leaders; su gasto de energía no es elevado.
⇒ Los leaders necesitan energía para realizar sus tareas y mantener estructurada la red, actualizar la tabla de ruteo y comunicarse con los demás clústers a través de los nodos que juegan el papel de gateways, si este pierde un α de su energía y no es suficiente para cumplir con sus tareas, renuncia y se convierte en un member, entonces el algoritmo se vuelve a reestructurar.
⇒ Los gateways o bridges, solo se comunican con los leaders de los clústers y los que no son requeridos o son redundantes se apagan.
⇒ Los bridges se utilizan para arreglar el problema de segmentación.

La representación de la red es por medio de un grafo, las aristas no tienen dirección y corresponden a la comunicación entre nodos G = (V,E), V es el conjunto de nodos (agentes) y E es la relación de aristas y vértices; cada nodo cuenta con un radio de comunicación E ⊂ V x V. Para cada arista $(u,v) \in E$ tienen un peso el cuál se representa con $w(u,v)$, que es el costo de la arista u a v.

Este trabajo se compara con dos algoritmos centralizados, éstos son MST (Minimum Spanning Tree) y PRIM [*Nieberg, 2004*].

Este proyecto fue realizado en simulación bajo NS2 con un total de 100 nodos y se obtienen resultados favorables y con mejoras respecto a MST y PRIM.

Este trabajo se toma como base para realizar la implementación del primer algoritmo de este proyecto. Se explica con más detalle en la sección 3.

1.3.2.2.1.3 Basadas en Árboles

En el proyecto de [*Bandara, 2008a*] para facilitar la intra e inter comunicación, se presenta un clúster de árbol.

Las VSNs soportan funcionalidades colaborativas, eficiencia de recursos, redes multipropósito, etc y se utilizadas principalmente en tres áreas que son:

⇒ Aplicaciones de sobre posición geográficas.
⇒ Redes de sensores multipropósito lógicamente separadas.
⇒ Mejorar la eficiencia de sistemas que siguen fenómenos dinámicos.

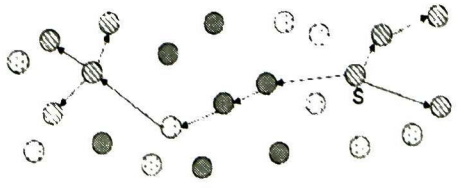


Figura 1.15 DosVSN's superpuestas

Cuando un nodo detecta por primera vez un evento relevante envía un mensaje hacia la raíz del árbol del clúster para avisarlo y este a su vez busque nodos similares que colaboren con el nodo detector. El nodo se une a un VSN existente o se junta con más nodos para formar uno.

El uso de un árbol en clúster o una estructura similar garantiza que dos o más nodos verán el mismo fenómeno; se realiza una comparativa con el algoritmo de Rumor Routing [*Braginsky, 2002*].

El uso, formación y mantenimiento de VSNs requieren la implementación de muchas funciones y protocolos que tendrán que adaptarse a los cambios de la red en los que se pueden tener: modificación, eliminación, adición, detección de múltiples VSN, combinación y división de nodos que permitan facilitar la comunicación con y a través de los VSNs.

En redes multifuncionales un nodo puede pertenecer a múltiples VSNs si mantiene información de diferentes VSNs para el mismo vecino. Si un nodo no detecta el fenómeno, entonces enviará un mensaje para separarse del VSN, *Unsubscribe_VSN(msg).*

Para la intra e inter comunicación se utilizan 3 tipos de mensajes:

⇒ *Unicast_VSN*(destination, type, data): Se necesita cuando se manda un mensaje a un miembro específico de un VSN.
⇒ *Muticast_VSN*(type, data): envía mensaje a todos los miembros de un VSN.
⇒ *Broadcast_VSN*(data): envía mensaje a todos.

Para la formación del árbol de clúster se utiliza el algoritmo denominado Generic Top-down Cluster and Cluster Tree Formation(**GTC**) Algorithm, [*Bandara, 2007*] para formar un árbol de clúster con la red.

El algoritmo GTC es configurable, independiente de la topología de la red y no requiere de información de vecinos a priori, ni conciencia de localización o tiempo de sincronización. En este proyecto se utiliza el algoritmo Hop a head Hierarchical Clustering (**HHC**), que es una especialización del anterior en el que la raíz puede tener hasta 6 clústers de hijos, mientras que las demás ramas sólo 3.

Para la formación de un árbol de clúster basado en una Red virtual de sensores se utiliza HHC para realizar la comunicación a través de los VSNs. Cuando un nodo detecta un evento para el primer tiempo envía un mensaje de *VSN formation/discovery* hacia el nodo raíz y mantiene cierta información (Como se muestra en la Figura 1.16 (b)).

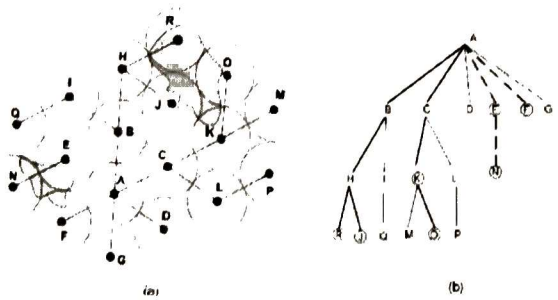


Figura 1. 16 Formación de VSN.

Cada CH mantiene un rastro de sus hijos mientras se está formando el árbol virtual, sin utilizar un esquema de direccionamiento. Los mensajes Unicast facilitan la comunicación en los VSN y necesita asegurarse que los CH's detectan el mismo fenómeno; cada CH le envía a todos los miembros CH's del VSN su dirección, de esta manera se conocen las direcciones de los nodos a partir de la raíz que los une (*Esquema Jerárquico*). [*Bandara, 2008b*].

Este esquema muestra gran funcionalidad debido a que se organiza en tres escenarios distintos. Se detectan diferentes tipos de eventos distribuidos en diferentes lugares del esquema, se muestran buenos resultados, pero a su vez pierde gran cantidad de mensajes.

Se incrementa la carga de la red que depende de las áreas donde se detecta el evento; el ahorro de energía no tiene gran prioridad.

De este trabajo se retoman las ideas de la formación de un árbol para realizar la propagación de la información a fin de mejorar la estructura de una red y se toman en cuenta las capacidades de los sensores.

1.4 Ruteo De Redes de Sensores

En este tipo de algoritmos en WSNs como su nombre lo dice, son los encargados de decidir por donde viaja la información, de un punto determinado a otro tomando en cuenta diversos factores como: cantidad de nodos, velocidad de transmisión, alcance de la información, tiempo de transmisión, tasa de transmisión, camino más corto, etc.

Los algoritmos de enrutamiento en las WSNs deben cumplir al menos con especificaciones como:

- ⇒ Mantener una tabla de enrutamiento.
- ⇒ Elegir la mejor ruta ya sea en base al costo, tiempo, confiabilidad, velocidad, etc. Depende de la aplicación para lo que se utilice.
- ⇒ Tablas para soportar cambios en la red.
- ⇒ Tiempo para converger.
- ⇒ Utilizar menor cantidad de transmisión de mensajes.
- ⇒ Ahorro de energía, etc.

1.4.1 Tipos De Ruteo

Para elegir el tipo de ruteo que se necesita, se solicitan características de la aplicación tales como tasa de transmisión, tiempo de vida de la red, tipo de hardware, software que se requiere, gasto de energía, etc. [*Serna, 2007*]

En las WSN regularmente los nodos no tienen conocimiento del tipo de red con el que se trabaja, está en su deber descubrir cómo trabaja la red actual; ejemplo de ello: anunciar cuando un nodo se agregará a la red y determinar la manera de actuar de los miembros en la red.

La organización de la red debe decidir quién será el encargado de gestionar sus recursos para comenzar a transmitir. Se utilizan diversos protocolos que nos ayudan en la administración de la red. Dentro de este apartado veremos algunos de los más comunes y su funcionamiento.

Modelo One Hop.

Se trata de la comunicación peer to peer, es decir, la comunicación directa entre 2 nodos; se comunican con una estación base y los nodos tienen un rango de red limitado. Este modelo es inseguro en las redes inalámbricas.

Modelo Multi-Hop.

En este modelo la información se retransmite de nodo a nodo hasta llegar a su destino, esta manera de trabajar es una de las más utilizadas en la construcción de WSNs porque muestra efectividad en la convergencia de los algoritmos.

Modelo basado en Clúster.

En esta aproximación los nodos se agrupan en clústeres que son liderados por un nodo, quién generalmente es el responsable de realizar la comunicación entre diferentes líderes de clústeres o a una estación base. La comunicación es multi-hop y este método es mejor que los anteriores, ya que proporciona entre cada sensor amplitud y mayor cantidad de relaciones, comunicación entre su grupo y redundancia [*Palma, 2009*].

Enseguida se presentan los tipos de enrutamiento más comunes que utilizan estas técnicas para el ruteo y propagación de información.

1.4.1.1 Ruteo Jerárquico

Se utilizan en redes en donde la información es dinámica. Se creó con el fin de reducir el uso de memoria en topologías muy grandes; se necesita definir la topología de la red a fin de utilizar el enrutamiento jerárquico y darle a los nodos un direccionamiento jerárquico. Los nodos sólo tienen conocimiento de los nodos a su nivel, los datos se envían entonces al router para que este a su vez los transmita, permitiendo a la tabla de enrutamiento reducirse de n^2 a $log\ n$. [*Romero y Muños, 2011*] [*Parekh, 2002*].

1.4.1.2 Ruteo Distribuido

En este tipo de ruteo cada nodo realiza su funcionalidad independiente de los demás, toma sus propias decisiones en base a su entorno y cada uno tiene una tabla de ruteo de nodos vecinos; la información que se maneja es local [*Tanenbaum, 2003a , Douglas 1995*].

Hay 2 maneras en la que se trabaja este tipo de ruteo:

⇒ *Vector de Distancias*: tiene 2 parámetros principales: el *nodo* y la *distancia* a la que esta del nodo actual.

⇒ *Estado de enlaces:* Cada nodo difunde a los nodos que se encuentran dentro de su alcance las distancias que él tiene con respecto a sus vecinos.

1.4.1.3 Ruteo Por Propagación (Flooding)

Cada mensaje se envía por todos los lugares posibles de salida menos por la línea de donde llego. Cada paquete contiene la dirección del destino y un contador que se decrementa en cada reenvío, al llegar a 0 ese mensaje deja de tomarse en cuenta. Es robusto por la duplicidad de mensajes que existe, aunque suele tener excesiva pérdida de memoria. Siempre elige el camino más corto y se eliminan los mensajes duplicados en el destino.

Para evitar aumentar la carga de la red, cada nodo recuerda la identidad de los paquetes que ya han sido reenviados. Se intentan todas las rutas posibles, todos los nodos se visitan y se utiliza en redes donde se requiere enviar información de gran prioridad.

1.4.1.4 Ruteo Centralizado

Este ruteo tiene información acerca de toda la red. Hay un nodo que se encarga de la toma de decisiones en base al estado de la red, como se muestra en la Figura 1.17. Es simple y frágil ante fallos.

Figura 1.17 Red de Ruteo Centralizada

Se presentan diversos trabajos, en [*Salem, 2009*] se propone una nueva arquitectura mediante la implementación de un algoritmo bio-inspirado ACO, donde se utiliza el método para encontrar una ruta óptima en una WSN multi-saltos. Los nodos trabajan de manera descentralizada para recolectar datos o detectar algún evento y llevar la información a través de comunicación multi-saltos. En ACO se trabaja de 2 maneras: *Forward Ants,* o *BackwardAnts.*

Forward Ants cuyo objetivo es explorar el camino y recolección de información del nodo fuente al nodo destino; se construye un árbol para transmitir la información. Tiene 2 factores principales para su funcionamiento: *rastreo de feromonas* que son depositados a lo largo de las aristas y el *potencial de los nodos* los cuales proveen un estimado de cuanto es lo más lejano que tiene que viajar la ant para llegar a su destino *d* o agregar datos con otro nodo.

Backward Ants viajan desde su destino a su nodo fuente, su función de desempeño actualiza la información en el paso por nodo.

El tipo de algoritmos ACO llegan a tener una convergencia total del sistema debido a que imitan el comportamiento de las ants, son competitivos y eficientes.

El desarrollo principal del proyecto son máquinas de estado que verifican la funcionalidad de este tipo de problemas, como administración de ruteo (*management routing*) el cual busca el camino que lleve al nodo destino más rápido, administración del vecindario (*neighborhood management*) que encuentra al mejor nodo que llevará la informacióny administración de la energía (*power management*) verifica la energía de los nodos, de acuerdo a sus condiciones ambientales (Figura 1.18).

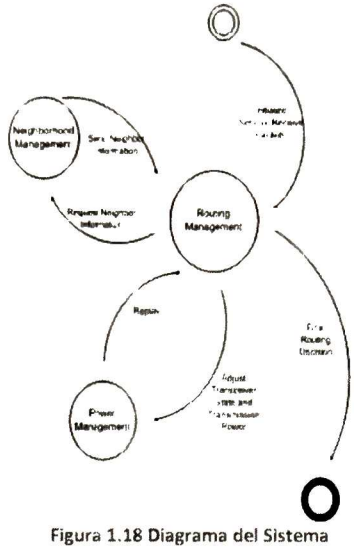


Figura 1.18 Diagrama del Sistema

Este algoritmo mejora el rendimiento de datos, también es capaz de evitar ciclos y utiliza reforzamiento de aprendizaje.

[*Singh, 2004*] En su trabajo provee un algoritmo on-line ACO usando técnicas de AntNet [*Di Caro, 1998*] para MSDC en donde se mejora el algoritmo agregando un nuevo tipo de Ants "*las aleatorias*"; su principal tarea consiste en disipar información entre los nodos de otro vecindario. Este algoritmo muestra mejores resultados que un algoritmo centralizado, pero la desventaja que presenta es que al utilizar Forward Ant gasta mucho tiempo en esperar las respuesta y mientras éste transcurre puede ocurrir un deadlock.

En [*Zhang,2004*] se realiza una comparativa de algoritmos para WSNs, en donde muestra cómo funcionan diversos algoritmos en las aplicaciones, estos son:

⇒ *Sensor-Driven and Cost-Aware Ant Routing* (SC):El principal problema de los algoritmos Ant- Routing es que se tardan en encontrar la solución debido a que no saben al principio cual es su destino; utilizan forward ants. En este algoritmo las ants tienen sensores y pueden oler en donde inicia la comida (camino) y se basan principalmente en ruteo basado en dirección. Cada nodo estima el costo desde su camino al destino de cada uno de los vecinos. Se utilizan características como conocimiento geográfico, etc.

El costo generaliza el objetivo del camino más corto y puede aplicar diferentes métricas de ruteo, como las que se mencionaron al inicio de este apartado.

El costo de estimación es Qn para un vecino n, el costo se calcula con la siguiente fórmula: $C = min_{n\epsilon N}(c_n + Q_n)$ donde c_n es la función de costo local y Qn puede obtenerse de 2 maneras, la primera es con por medio de:

$$k\sqrt{(x_d - x)^2 + (y_d - y)^2}$$

donde(x, y) es la posición del nodo actual y $(x_d - y_d)$ es la posición del nodo destino.

La otra manera es que entre cada salto de vecinos se calcula un costo, Qn es la suma de los costos entre 2 nodos, siempre y cuando el destino sea conocido.

La función de probabilidad es:

$$p_n \leftarrow \frac{e^{(C - Q_n)^\beta}}{\Sigma_{n\,\epsilon\,N} e^{(C - Q_n)^\beta}}$$

⇒ *Flooded Forward AntRouting* (FF): Las forwards ants pueden equivocarse debido a obstáculos o al movimiento de destinos. Este algoritmo explota el canal debido a las inundaciones de mensajes broadcast, cuando inicia un ant a buscar comida avisa a sus vecinos quienes realizan su propagación por los nodos vecinos, hasta que el destino es encontrado. Cuando regresan a su fuente dejan pistas de feromonas. Esto sólo se puede realizar una vez que se tiene de cerca a la comida. Los enlaces de probabilidades son usados para la estimación. Presenta problemas con las colisiones debido a que cada nodo presenta un retraso para transmitir.

⇒ *Flooded Piggy backed Ant Routing* (FP):Cuando se realizan caminos simples, se tiende a perder gran cantidad de información. El ruteo multi-caminos tales como inundación es un algoritmo robusto y no tiene tanta perdida de datos. En este algoritmo se combinan Forward Ants y Data Ants, para ejecutar una búsqueda de caminos como el algoritmo anterior y a su vez descubrir el mejor camino. Data Ants no sólo se encarga de llevar la información del nodo fuente al destino, sino también de llevar un registro de las rutas que se realizan. Su desventaja es el alto consumo de energía.

1.5 Protocolos De Redes Inalámbricas

En las WSN se presentan grandes retos que no se consideraban en las redes cableadas, entre ellos están la movilidad de los nodos, ruido en el medio, baterías limitadas, pérdida de mensajes, la conectividad, etc. (Figura 1.19).

Las WSN, suponen un primer reto al respecto: considerando miles de nodos sin una estructura fija se organizan para realizar una tarea en concreto y toman decisiones de manera autónoma, tomando en cuenta el medio, los recursos limitados y la diversidad de dispositivos que existen en la actualidad. Este tipo de redes sin embargo, deben ofrecer funcionalidad, rapidez, seguridad, entre otras cosas. Para que esto sea posible, se requieren de protocolos que nos facilitan la manera de comunicarse entre diversidad de nodos.

Figura 1.19 Red Inalámbrica

Dentro de las redes se utiliza entonces la subcapa MAC, perteneciente a la capa de enlace de datos, se encarga del control de acceso al medio y es responsable de transmitir los paquetes, validar las tramas que recibe, comprobar errores en la transmisión, tasa de transmisión, control de flujo, confirmar la recepción de tramas al emisor, etc.

Los protocolos que se utilizan para el ruteo de la información trabajan a este nivel de capa, debido a que influyen de manera directa en la manera en la que acceden al medio para transmitir. Estos protocolos se pueden agrupar en dos clases básicas que son:

⇒ *Los protocolos basados en slots o ranurados:* dividen el tiempo en intervalos (*frames o slots*) y dividen los estados de un nodo en: *transmitir, recibir o apagarse*. Se utilizan tiempos de sincronización para manejar estos estados. Los costes de la sincronización y mantenimiento penalizan el consumo energético y el ancho de banda. Entre ellos se incluyen los protocolos TDMA, IEEE 802.15.4, S-MAC, T-MAC, entre otros.

⇒ *Los protocolos basados en muestreo*: al contrario que los ranurados, se apagan la mayor parte del tiempo y sólo encienden en periodos en busca de actividad en el canal, en caso de encontrar algo, comienzan a recibir datos, sino, se apagan de nuevo para ahorrar energía. La detección se puede basar en el nivel de energía del canal o en la detección de la portadora. Estos suelen ser flexibles pues permiten comunicarse con cualquier sensor dentro de su rango de alcance, pero no siempre, porque no tienen sincronización. Entre ellos están Aloha, B-MAC ,WiseMAC, transceiver de ChipconCC2500 y la plataforma mica de Berkeley.

En este trabajo sólo nos enfocaremos a los protocolos basados en slots.

1.5.1 TDMA

El principio de TDMA es básicamente simple. Tradicionalmente, los canales de voz han sido creados dividiendo el espectro de la radio en portadoras de frecuencia RF (canales), con una conversación que ocupa un canal dúplex. Esta técnica es conocida como FDMA (*frecuency división multiple access*). TDMA divide a los portadores de la radio en una sucesión repetida de pequeñas ranuras de tiempo (canales). Una trama (Figura 1.20) es una sucesión de N intervalos, la transmisión se organiza en tramas de duración *Ti*, la duración de un intervalo es *T=Ti/N*. La información se transmite en forma de un tren de bits llamado ráfaga (burst). Cada conversación ocupa justo una de estas ranuras de tiempo así en lugar de sólo una conversación, cada portador de la radio lleva varias conversaciones. [*Hernández, 2011*] [*Soto, 2011*].

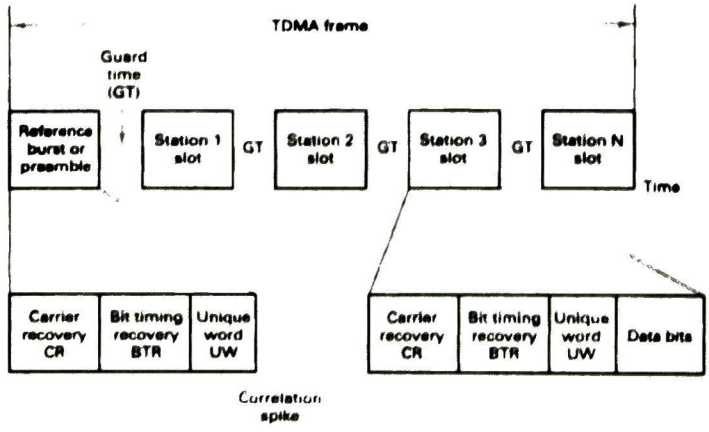


Figura 1.20 FrameTDMA [Hernández, 2011]

1.5.2 Zigbee/802.15.4

[*Andreu, 2011*] 802.15.4 se considera estándar desde el 2003 y surge por la necesidad de un protocolo estándar de bajo consumo, flexible y bajo ancho de banda para redes de sensores. En este protocolo se soportan sólo dos tipos de topologías:

⇒ Topología en *estrella*: se utiliza cuando se implementen redes que necesiten baja potencia.
⇒ Topología *"Peer-to-Peer"* Se implementan cuando se requieren redes amplias y con mayor precisión.

En este protocolo se trabajan 3 tipos de nodos, cada uno con funciones específicas en función de la topología de red, éstos son [*WSN, 2010*]:

⇒ **RFD** (*Reduced Function Device*): Dispositivo de Función Reducida. Limitado a la topología en estrella, los nodos sólo pueden comunicarse con los coordinadores de la red y no pueden desempeñar el papel de coordinador.

⇒ **FFD**(*Full Function Device*): Dispositivo de Función Completa. Este dispositivo puede desempeñar cualquier tarea y comunicarse con todos los nodos de la red. Un coordinador de la red será FFD.

⇒ **PAN Coordinador**(*Personal Area Network*) actúan como ruteadores y administran la carga de la red.

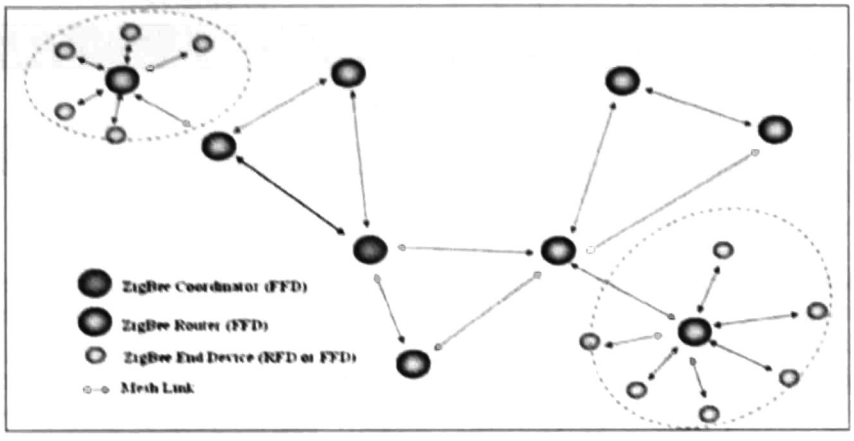
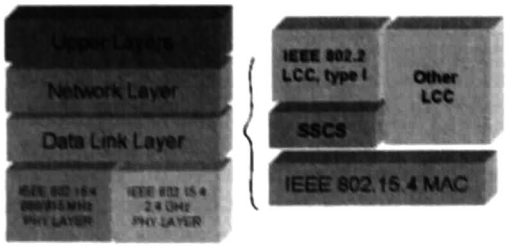


Figura 1.21 Red ZigBee/802.15.4

Capa de Enlace de Datos (**Data Link Layer, DLL**).



[*Mayne, 2011*][*Andreu, 2011*] IEEE 802 divide a la Capa de Enlace de Datos en 2 subcapas, la capa de Control de Acceso al Medio (MAC Medium Access Control), y la de Control de Enlaces Lógicos (Logical Link Control, LLC) (Figura 1.22)

Figura 1.22 Relación de 802.15.4 con Modelo OSI

MAC IEEE 802.15.4 cuenta con características como son: asociación/disociación, reconocimientos de entrega de trama (ACK), mecanismos de acceso al canal, validación de trama, control de garantía de ranuras de tiempo (Slot Time), control de guías (Beacon) y barrido de canal. El manejo de servicios MAC se accede por medio de la capa MAC de manejo de identidades (MLME-SAP).

El formato general de las tramas MAC se diseñó para ser muy flexible y se ajustará a las necesidades de las diferentes aplicaciones, con diversas topologías de red, al mismo tiempo que se mantenía un protocolo simple.

⇒ ***Data Frame***: usado para todas las transferencias de datos. Se le denomina **PPDU** (PhyProtocol Data Unit), la trama empieza con un encabezado de sincronización (**SHR**, Synchronization HeadeR), seguido de un encabezado de capa física para indicar la longitud

del paquete (**PHR**, PhyHeadeR), y seguida de la capa física de la unidad de servicio de datos (**PSDU**, PhyService Data Unit, PSDU) (Figura 1.23).

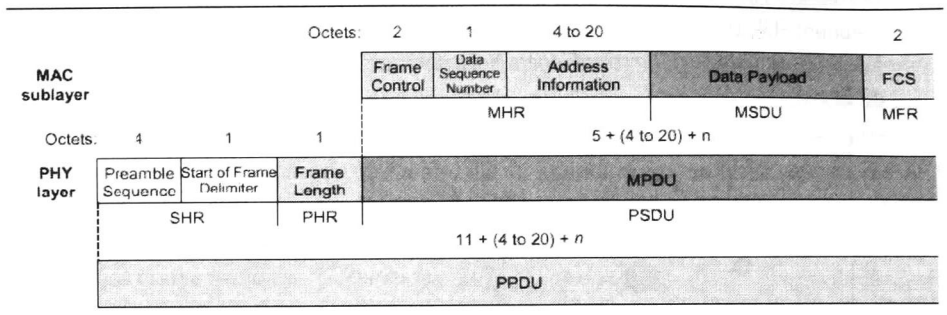


Figura 1.23 Formato Trama MAC

⇒ ***Acknowledgment Frame***(Figura 1.24): usado para confirma la recepción exitosa de la trama.

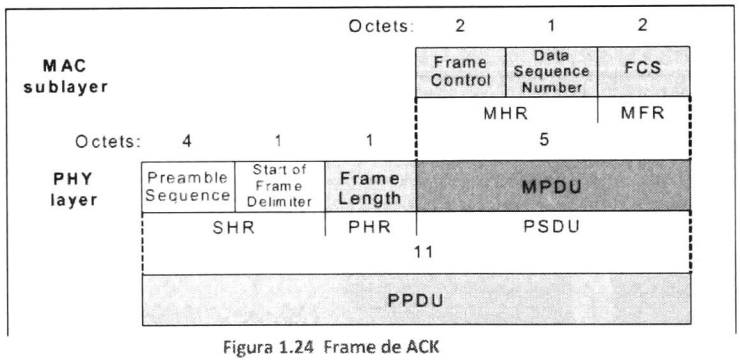


Figura 1.24 Frame de ACK

⇒ ***MAC Command Frame***: usado para manejar el control de entidad MAC o configuración a distancia de los dispositivos de los nodos. Permite que un director de la red centralizado pueda configurar a los dispositivos individualmente sin importar cuan grande que sea la red (Figura 1.25).

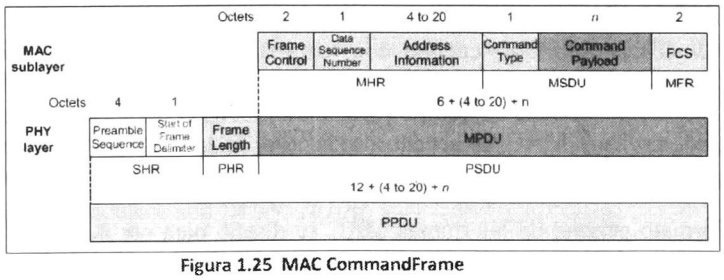


Figura 1.25 MAC CommandFrame

⇒ ***Beacon Frame***: usado por un coordinador para transmitir "beacons" Los dispositivos de los nodos pueden despertarse solamente cuando es transmitida una señal de guía o "beacon", escuchar su dirección y si no la escucha volver al estado dormido con el consecuente ahorro de energía.

Las tramas Beacon son importantes en las redes "*mesh*" y "*cluster tree*" para mantener los nodos sincronizados sin requerir que consuman energía de la batería mientras se encuentran en modo escucha durante largos periodos de tiempo (Figura 1.26).

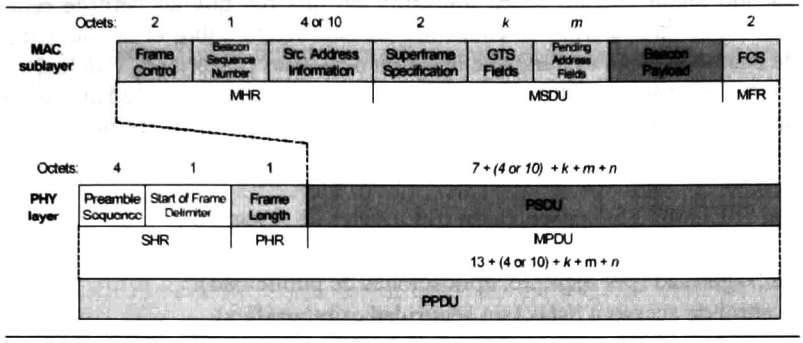


Figura 1.26 BeaconFrame

En una *red con señal de guía o "beacon"*, cualquier dispositivo que desee transmitir durante el periodo de acceso de contención, espera a que empiece el siguiente "time slot" y después determina si algún otro dispositivo se encuentra transmitiendo en el mismo "time slot". Si algún otro dispositivo se encuentra transmitiendo en dicho momento, el dispositivo se repliega a un número aleatorio de slots o indica un fallo en la conexión después de varios intentos. Además en una red con señal de "beacon", las tramas ACK no utilizan CSMA.

Esta estructura de la trama Beacon contiene las denominadas Superframes (Figura 1.27). Algunas aplicaciones requieren anchos de banda dedicados para lograr grandes estados latentes para un muy bajo consumo de energía. Para lograr dichos estados latentes el IEEE 802.15.4 puede operar este modo opcional, llamado Superframes.

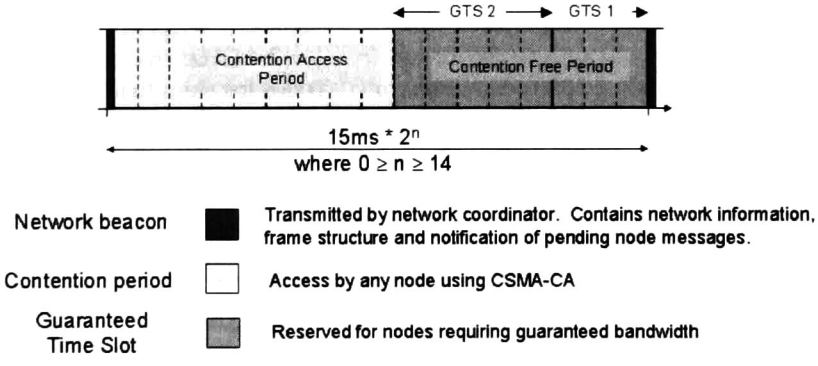


Figura 1.27 Superframe de 802.15.4

En un Superframe, un coordinador de red denominado coordinador PAN transmite Superframes de guía (beacon) en intervalos definidos. Estos intervalos pueden ser tan cortos como 15 ms o tan largos como 245 s. El tiempo entre cada uno de ellos se divide en 16 "time slots", independientes a la duración de cada Superframe. El coordinador de PAN puede asignar intervalos o "time slots" a un solo aparato que requiera un determinado ancho de banda permanente o transmisiones de baja latencia. Estos "time slots" asignados son llamados "time slots" de garantía (**GTS**- *Guaranted Time Slots*).

La trama que contiene el ACK se manda de retorno inmediatamente después de que se hace una validación exitosa de la trama de entrada. Las tramas de guía (*beacon frames*) enviados por el coordinador del PAN nunca son respondidas con algún ACK.

En redes *sin Beacon* se utiliza el estándar CSMA-CA. Estas redes trabajan de la siguiente forma: cuando algún aparato desea transmitir en una red que no permite señales de guía, la red primero revisa si otro aparato se encuentra transmitiendo sobre el mismo canal. Si es el caso, el intento de acceso al canal se tiene que hacer en ocasiones posteriores, o indica una falla de conexión después de varios intentos fallidos. La trama ACK confirma si una transmisión previa no utiliza los mecanismos de CSMA dado que estos se mandan inmediatamente después de cada paquete de información.

El estándar IEEE 802.15.4 proporciona tres niveles de seguridad:

⇒ Sin seguridad (por ejemplo, aplicaciones de publicidad).
⇒ Control de acceso a listas (sin seguridad criptográfica).
⇒ Seguridad con clave simétrica.

Para minimizar costos para dispositivos que no lo requieran, el método de distribución de clave no se especifica en el estándar pero se debe de incluir en capas superiores de las aplicaciones apropiadas.

Las características más importantes del estándar 802.15.4 son las siguientes:

Características	Propiedades
Bandas de frecuencia	868 MHz: 20kb/s;
Rango de transmisión de	915 MHz: 40kb/s;
datos	2.4 GHz: 250 kb/s.
Alcance	10-20 m
Latencia	Por debajo de 15 ms
Canales	868/915 MHz: 11 canales.
	2.4 GHz: 16 canales.
Modo de direccionamiento	Todos los chips tienen 64 bits
	IEEE de direccionamiento
Canal de acceso	CSMA-CA
Seguridad	128 AES
Red	Hasta 2^{64} dispositivos
Rango de Temperatura	-40° a +85° C

Tabla 1.1 Propiedades del Estándar 802.15.4

El estándar Zigbee se enfoca a un segmento del mercado no atendido por los estándares existentes, con baja transmisión de datos y bajo ciclo de servicio de conectividad. La razón de promover un nuevo protocolo como un estándar, es para permitir la interoperabilidad entre dispositivos fabricados por compañías diferentes.

ZigBee es un estándar de hardware y software basado en el recientemente ratificado estándar IEEE 802.15.4, es un estándar de PAN (*Personal Area Network*). Este importante estándar define el hardware y el software, el cual ha sido descrito en los términos de conexión de redes como la capa físicas (PHY) y la capa de control de acceso al medio (MAC). La alianza ZigBee ha añadido las especificaciones de las capas red (NWK) y aplicación (APL) para completar la pila o stack ZigBee.

ZigBee se ha implementado en la banda mundial de 2.4GHz, sin necesidad de licencia o una de las bandas regionales de 868/900 MHz. Las bandas de 2.4 GHz y 868/900 MHz fueron escogidas por el estándar IEEE 802.15.4 por sus características de propagación. La propagación hace referencia a la manera en que las ondas de radio actúan en el medio ambiente.

El hardware y la redes Zigbee deben permitir una transferencia de datos de 10 a 115 kbps, que representa cantidad de datos que pueden ser transferidos en cuanto el protocolo de cabecera sea retirado. El hardware Zigbee debe comunicarse sobre un rango entre 10 a 75 metros. ZigBee mantiene el coste bajo mientras que añade potencia con la conexión de redes en malla, una característica que no se encuentra en la mayoría de los estándares de conexión de redes de radio.

1.5.3 SMAC

[*Equihua, 2009*] [*Freescale, 2008*]El protocolo S-MAC (*Sensor Medium Access Control*) es similar al 802.15.4 salvo algunas diferencias notables en sus mecanismos.

S-MAC está basado en un esquema RTS-CTS (*Requestto Send/Clear toSend*) para evitar el problema de la terminal escondida; es de bajo consumo de energía para redes de sensores. A través del paso de mensajes, un nodo puede transmitir un conjunto de paquetes usando un único RTS/CTS para negociar (*handshake*). Todo paquete debe ser por tanto reconocido y describir la duración de su transmisión para determinar cuando un nodo puede dormir.

S-MAC periódicamente duerme, despierta, escucha el canal y después vuelve a dormir (*dutycycle)* (Figura 1.28). Se diseño para funcionar como una "caja negra", ya que no dispone de ninguna interfaz, primitiva o funcionalidad que permita alterar el *dutycycle* ó sus parámetros.

A cambio de esto ofrece:

⇒ Encaminamiento,
⇒ Organización,
⇒ Sincronización y
⇒ Servicios de fragmentación de paquetes que forman parte del protocolo.

El periodo de actividad es fijo en S-MAC es de 115ms, pero el periodo en el que duerme es variable. La longitud del periodo de dormir dictamina el *dutycycle*. La red se divide en celdas y cada nodo perteneciente a una celda intercambia información necesaria para la sincronización de sus envíos, los nodos que pertenecen a más de una celda deben mantener la información de sincronización de ambas celdas.

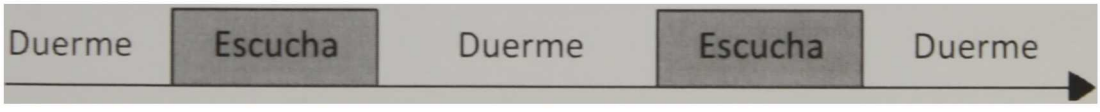


Figura 1.28 Ciclos de S-MAC

S-MAC requiere sincronización. Los nodos vecinos se intercambiarán mensajes de sincronización periódicamente (SYNC). Las marcas de tiempo intercambiadas con este SYNC son relativas.

Sensor MAC es el primer protocolo de la capa MAC desarrollado para redes de sensores y ha tenido como era de esperar muy en cuenta las limitaciones de energía.

La principal idea del S-MAC es ahorrar energía encendiendo y apagando periódicamente las radios de los nodos, apareciendo el término de ciclo de trabajo, en lugar de tener las radios siempre escuchando posibles transmisiones [Lewis, 2004].

1.6 Síntesis

En este capítulo se presenta la definición de una red, a tipos de red, aplicaciones, ruteo y tipos de propagación.

Las redes de sensores inalámbricas son en la actualidad puntos de partida importantes, cada día contamos con más y mejores dispositivos que se conectan a través de todo el mundo para navegar, realizar negociaciones, etc.

Una red de sensores verifica su canal de comunicación, tasa de transmisión, tiempo de transmisión, etc.

Se toman como trabajo base la implementación de un algoritmo de auto-organización [*Olascuaga, 2011*] previamente definido para la realización de este proyecto y se revisan diferentes algoritmos de ruteo, para complementar el trabajo de la red.

Actualmente el objetivo del ahorro de energía supone el eje principal en la investigación con protocolos de control de acceso al medio (MAC). Los protocolos de encaminamiento en redes inalámbricas de sensores se pueden clasificar generalmente en dos grupos, de acuerdo a la *estructura de la red* o al *criterio de encaminamiento utilizado*.

En nuestro caso trataremos con Auto-Organización de WSN, y después de ello se propone un algoritmo de propagación por la red por la estructura de la red, para realizar el paso de la comunicación de manera eficiente, el protocolo que se utiliza para este segundo algoritmo es el SMAC y el 802.15.4 son los más rentables, de menor coste, conservador de energía y de fácil manejo.

En los capítulos siguientes se formaliza el trabajo y de se explica de manera detallada, el trabajo realizado a lo largo de la aplicación de esta aplicación.

CAPÍTULO 2

Sensores MC1321X: Arquitectura y Configuración Para Uso en Red

---

Resumen. En este capítulo se describen los sensores Freescale MC1321X, su configuración, limitaciones, alcances y modificaciones que se realizaron a lo largo de la implementación. Se presenta la arquitectura de los sensores y la funcionalidad de estos. Finalmente se describe el protocolo utilizado SMAC y sus directivas.

2.1 Descripción Funcional de los Sensores MC1321X.

Las redes de sensores inalámbricas se basan en dispositivos de bajo costo y consumo, cuyo trabajo es recolectar información del exterior para procesarla localmente y comunicarla a través de otros sensores que se encuentran distribuidos espacialmente, para realizar tareas como medición de temperatura, humedad, luz, etc. [*Aakvaag& Frey, 2006*].

Cada sensor cuenta con una arquitectura general (Figura 2.1), compuesta de un Transmisor (TX) / Receptor (RX), un micro controlador, un procesador de energía y un sensor.

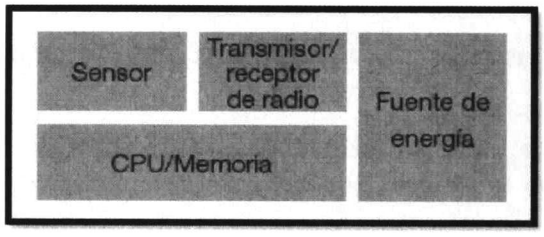


Figura 2.1 Arquitectura General de los Sensores[Aakvaag& Frey, 2006]

A los nodos también se les conoce con el nombre de motas, debido a que están compuestos por una placa de sensores o de adquisición de datos y un "mote" o mota que es una placa de procesador TX/RX.

Los sensores tienen la capacidad de restaurar la comunicación si alguno falla (Robusto a fallos), así este conformada por un número pequeño de dispositivos.

Las propiedades de Auto-diagnostico, Auto-Configuración, Auto-Organización y reparación son manejables para realizar una red más segura, y solventar problemas actuales que dependen de la aplicación de la red.

Los dispositivos de Freescale tienen la memoria suficiente para soportar la pila de protocolos y la aplicación y sólo utilizan el 50% del consumo de energía.

Entre las aplicaciones que se incluyen en los sensores están:

⇒ Automatización residencial y comercial
⇒ Control Industrial
⇒ Cuidados de la salud
⇒ Clientes

Una de las características principales de este tipo de redes además de que realizan una tarea en particular, es que no necesitan de la intervención humana, debido a ello, son muy eficientes y proporcionan muchas ventajas al utilizarlas en ambientes hostiles. Los nodos de la red normalmente se comunican con sensores llamados gateways.

Para el trabajo realizado se toman en cuenta las siguientes características:

⇒ Enlace de calidad.

⇒ Tiempo de ejecución del algoritmo.

⇒ Cantidad de sensores.

⇒ Rango de alcance de los sensores.

Se utilizan a lo largo de la estructura de este proyecto, se tienen algunas limitaciones y se realizan suposiciones, tales son:

⇒ Las pruebas se realizan con 6 sensores.

⇒ Cada sensor ejecuta el algoritmo de manera independiente.

⇒ Una vez que se cumplen determinada cantidad de ACKBROADCAST recibidos, se detiene el primer algoritmo (Auto-Organización) para iniciar el segundo (Propagación), al finalizar esta condición, debe existir al menos un líder y un miembro perteneciente a él.

⇒ El tiempo de escucha es aleatorio.

⇒ El tiempo de ejecución del algoritmo no es fijo.

⇒ El número de sensores con que se cuenta es limitado, las pruebas que se realizan son con una red pequeña.

Las redes inalámbricas no son seguras puesto que los nodos pueden fallar fácilmente, sin embargo, se pueden implementar medidas para que sean más confiables. En las redes distribuidas se realizan diversos caminos de comunicación del origen al destino, evitando causar *deadlocks* o algún *Livelock*.

Una limitante de este tipo de redes es la administración de energía, ya que los sensores inalámbricos se alimentan regularmente de baterías. Algunas de las soluciones a este problema son algoritmos que se enfoquen en la conservación, generación o administración de energía. Con el fin de prolongar el tiempo de vida de la red.

La administración de estas redes depende de la aplicación para la que se definen. Por ejemplo, en la auto-organización, se requieren estándares para su implementación por su complejidad. Los nodos despiertan, detectan a otros que están sólo en el rango de su alcance y forman la red mediante las interacciones, después reportan la información a una o diversas estaciones base.

2.2 Configuración para uso en una red e implementación de algoritmos

2.2.1 Sensores Freescale

[*Reference Manual, 2010*] Sensor 1321X es un dispositivo basado en Freescale MC1322X, diseñado para proveer una alta integración, una solución total con altas capacidades de procesamiento y bajo consumo de energía.
Estos dispositivos están provistos con:

⇒ Un radio 802.15.4.
⇒ Interfaz de usuario.
⇒ Capacidades de debug.
⇒ Conexión a computadoras y otros dispositivos.
⇒ Sensores.
⇒ Portabilidad.

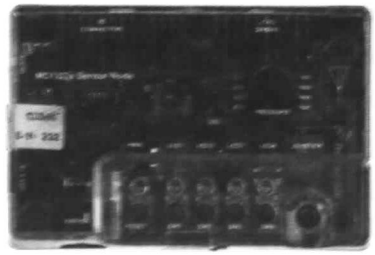


Figura 2.2 Dispositivo MC1322X [Reference Manual, 2010]

Algunas de las aplicaciones que incluye son:

⇒ Automatización residencial y comercial
  - Control de luz. Seguridad, Control de Acceso
  - Calefacción, Ventilación y Aire Acondicionado (HVAC) por sus siglas en inglés, etc.

⇒ Control Industrial
  - Seguimiento de activos y monitoreo, Administración de la industria, Control y monitoreo del ambiente, etc.

⇒ Cuidados de la salud
  - Monitoreo de adultos y Monitoreo de gimnasio.

⇒ Clientes
  - Control Remoto, Sistemas de Entretenimiento y Ataques a teléfonos celulares.

*Para ver más características ver Anexo A.

Freescale provee un poderoso ambiente de software llamado *Freescale BeeKit Wireless Connectivity Toolkit*. BeeKit es un codebase de librerías de redes inalámbricas, templates y ejemplos de aplicaciones. De la que se tomo una de las aplicaciones para la realización del proyecto.

2.2.2 Módulo de Comunicación y Control

Los sensores cuentan con 3 tipos de memoria:

⇒ 96 Kbyte de RAM: para soportar las capas inferiores de los protocolos (SMAC, ZIgBee y RF4CE) y el software de aplicación,

⇒ 128 Kbyte de memoria FLASH que es direccionada a la memoria RAM,

⇒ Memoria ROM de 80 Kbyte disponible para software de arranque, capas superiores del protocolo IEEE 802.15.4 MAC y para el software de capas de comunicación.

Acceso a memoria DMA para transmitir y recibir paquetes de información.

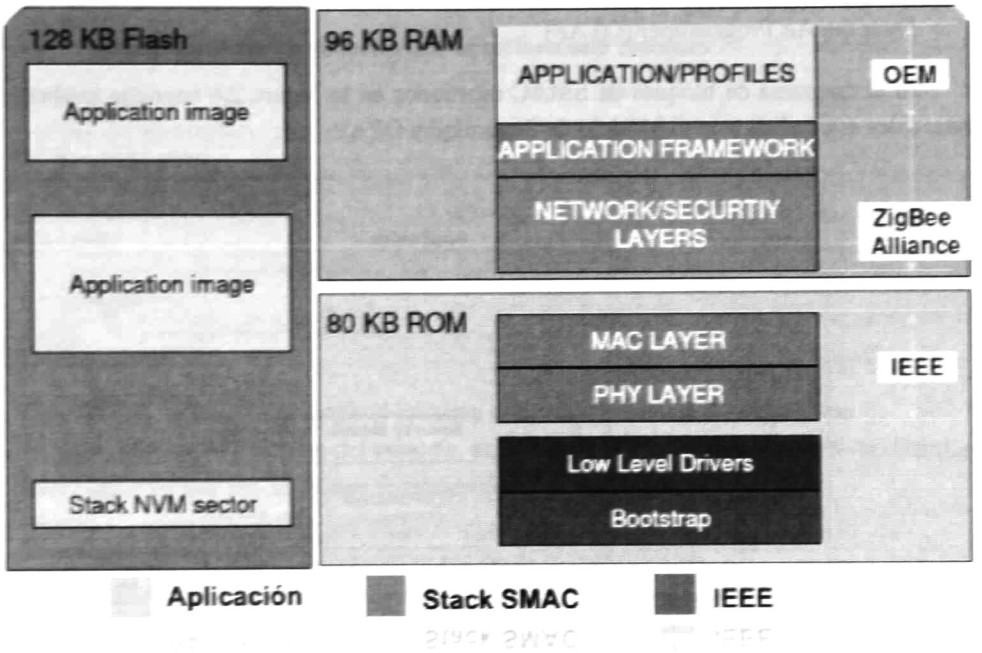


Figura 2.3 Distribución del Software y Protocolo [Reference Manual, 2010]

El micro controlador como módulo de control se encarga de hacer la comunicación con el módulo sensorial por medio de su interfaz SPI (Serial Peripheral Interface) para monitorear e interpretar la información proveniente del módulo sensorial.

El módulo de comunicación está encargado de establecer la comunicación confiable con el coordinador utilizando el transceptor integrado de RF, para enviar la información adquirida.

2.2.3 SMAC en los Sensores Freescale

Algunas de las aplicaciones disponibles en los sensores Freescale soportadas por SMAC, generadas por los proyectos de BeeKit Solution son:

- ⇒ Basic Packet Error Rate(PER)
- ⇒ Wireless UART
- ⇒ Accelerometer V 3.0
- ⇒ Range
- ⇒ Lighting
- ⇒ Test Mode
- ⇒ Repeater
- ⇒ Simple Protocol Test Client
- ⇒ Over the Air Programmer (OTAP)

En el diagrama de bloques de SMAC mostrados en la Figura 2.4 contiene módulos, los más destacables entre ellos son: el Módulo de Seguridad y OTAP.

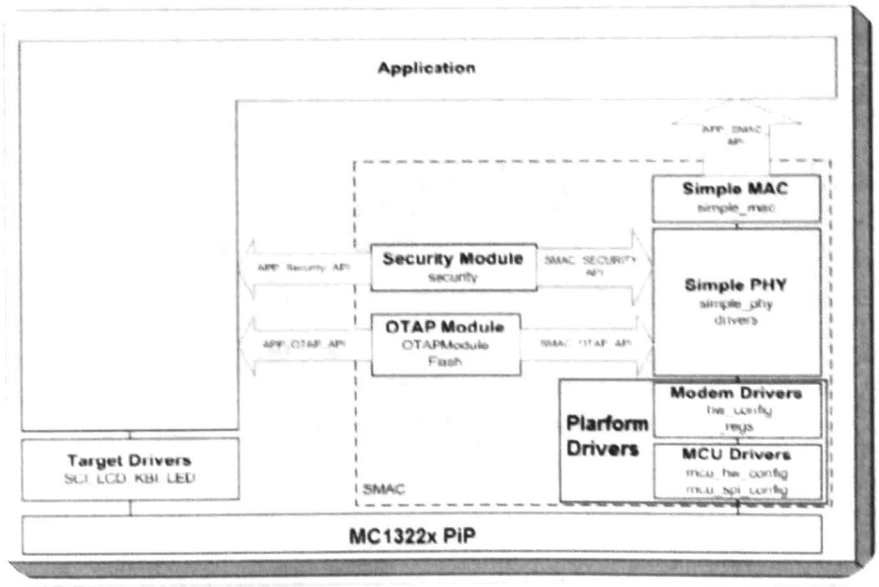


Figura 2.4 Dispositivos MC1322X en SMAC [Reference Manual, 2010]

SMAC trabaja bajo las siguientes estructuras de datos:

tTxPacket: Estructura que define los paquetes que se van a transmitir, es la siguiente:

```
typedefstruct {
UINT8u8DataLength;    número de bits a transmitir.
UINT8 *pu8Data;        puntero al buffer de datos a transmitir.
} tTxPacket;
```

Donde

- *u8DataLength,* es la longitud del paquete que se enviará
- *pu8Data*, contiene los datos que se transmitirán.

tRxPacket : La estructura de los paquetes que se reciben será entonces:

```
typedefstruct {
UINT8u8MaxDataLength;
UINT8u8DataLength;
UINT8 *pu8Data;
UINT8u8Status;
} tRxPacket;
```

Donde

- *u8MaxDataLength,* es la longitud máxima que puede recibirse de un paquete
- *u8DataLength*,longitud total del paquete, puede llegar a ser la misma que u8MaxDataLength
- *pu8Data*, contiene los datos que se transmitirán
- *u8Status*, contiene el estado del paquete que se recibe, pueden ser:
    - *SUCESS* cuando un paquete se recibe de manera exitosa
    - *TIMEOUT* después de un tiempo de escucha, el paquete se retrasa, cayendo en este estado.

Los tipos de datos que se manejan en SMAC son:

⇒ UINT8 Unsigned 8 bit definition
⇒ UINT16 Unsigned 16 bit definition
⇒ UINT32 Unsigned 32 bit definition
⇒ INT8 Signed 8 bit definition
⇒ INT16 Signed 16 bit definition
⇒ INT32 Signed 32 bit definition

2.2.3.1 Directivas de SMAC

Las primitivas correspondientes a SMAC que se van a utilizar son las siguientes:

⇒ **MCPSDataRequest**: función se usa para enviar un paquete.
⇒ **MCPSDataIndication:** es una función llamada CallBack, esta función se llama al recibir un paquete y es procesado por la aplicación para asignarle el status SUCESS o TIMEOT.
⇒ **MLMESetChannelRequest**: configura la frecuencia actual en que el radio transmite y en la cuál recibe.
⇒ **MLMERXEnableRequest**: configura el radio en modo recibir sobre el canal que fue seleccionado de la función MLMESetChannelRequest ().
⇒ **MLMEMC13192PAOutputAdjust**: esta función ajusta la potencia de salida del transmisor. Por medio de parámetros como NOMINAL_POWER, MAX_POWER ó MIN_POWER.
⇒ **MLMELinkQuality:** lee la calidad del enlace del último mensaje recibido.
⇒ **MLMEMC13192ResetIndication():** función callback, también necesitada por SMAC, esta se ejecuta cuando SoftReset a ocurrido.
⇒ **MLMEMC13192PAOutputAdjust**: esta función ajusta la energía de salida del transmisor.

Para ver con más detalle las funciones y su utilización ver Anexo C. Las funciones que no están declaradas se explican en las secciones siguientes, fueron las agregadas y utilizadas para los algoritmos.

2.2.4 Arquitectura De Los Sensores Freescale MC1321x

Los sensores 1321X cuentan con una arquitectura propia, donde se pueden identificar los elementos básicos de un nodo, compuestos de memoria, procesamiento, funciones de sensor, módulo de Rx/ TX, Administración de energía, etc, como se puede apreciar en la Figura 2.5.

Con la combinación optimizada de la memoria ROM, RAM y memoria FLASH, los usuarios se aseguran de satisfacer sus requerimientos sobre los dispositivos. La plataforma única en el Paquete (PiP *Platform in Package*) de diseño integra los componentes RF, simplifica el diseño de las redes inalámbricas, para permitir al usuario conectar una antena de 50 Ohm, reducir el coste y tiempo del sistema

Esta arquitectura presenta ventajas por su estructura y la manera en que trabaja, nos beneficia por el ahorro de energía y características que se mencionaron en secciones anteriores. Se pretendió obtener el mayor provecho de la estructura. Y a partir de la aplicación utilizada"PER TEST" realizar las adaptaciones suficientes para el manejo del dispositivo.

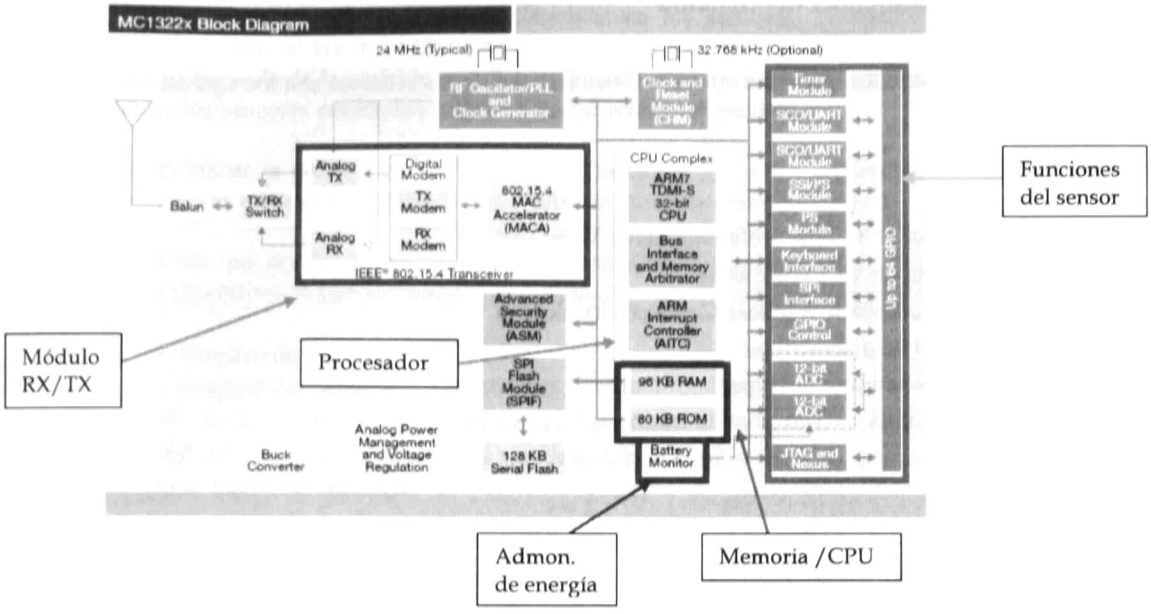
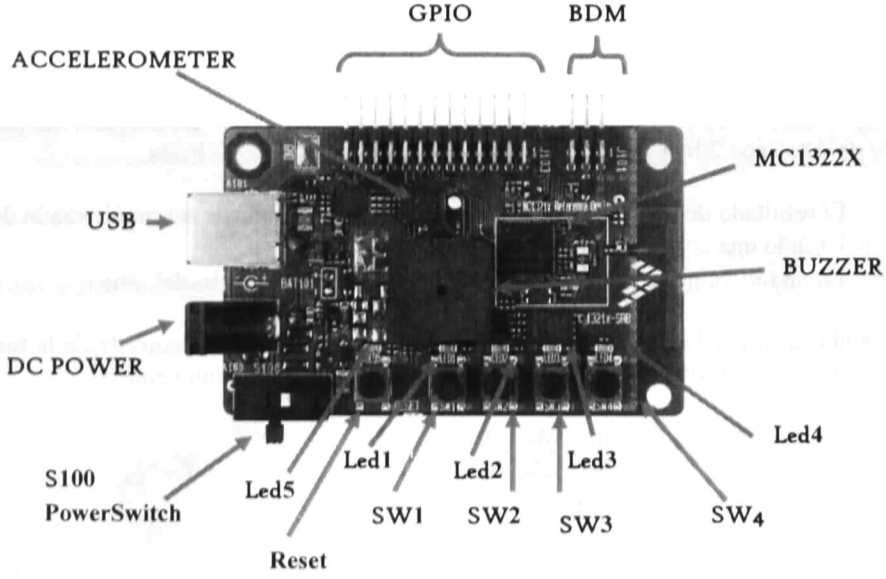


Figura 2.5 Partes de la Arquitectura de los Sensores [Reference Manual, 2010]

Los sensores contienen elementos principales dentro de su estructura, los cuales se muestran en la Figura 2.6.



2.6 Partes del sensor Freescale MC1322X

2.2.5 Requerimientos del Sistema

Los requerimientos para una correcta funcionalidad de los sensores son los siguientes:

Software

⇒ Windows de 32 bits, preferentemente Windows XP.
⇒ CodeWarrior v6.1 o posterior.
⇒ Freescale BeeKit Wireless Connectivity Toolkit.
⇒ Terminal de dispositivos
  - Windows →HyperTerminal.
  - Linux → Terminal de Linux.

Hardware

⇒ P&E's USB BDM Multilink, compartible con Windows XP.
⇒ Sensores MC1321X.
⇒ Cables son serial/USB.
⇒ Una computadora por cada sensor.

2.2.6 Implementación de PER Test en los sensores

Basic Packet Error Rate (PER) test es una prueba unidireccional, cuya tarea es la de enviar 1000 paquetes de un transmisor a un receptor por un único canal. La longitud del paquete es de 18 bytes de datos con 2 bytes de cabecera SMAC, para un total de 20 bytes.

El resultado de los paquetes recibidos por el receptor puede ser monitoreado de dos maneras
⇒ Usando una conexión de PC con USB.
⇒ En un modo independiente, leyendo los leds sobre el tablero del sensor.

La aplicación se basa en una máquina de estados, que son manejados desde la función principal main, y la directiva MCPCDataIndication, cada vez que se recibe un paquete.

Se utilizan 2 aplicaciones diferentes:
- *PER Test TX* para el transmisor
- *PER Test RX* para el receptor.

En este proyecto sólo se utilizó la del transmisor, debido a que presenta la ventaja de comunicación con sensores TX y RX, no siendo así para el receptor que sólo se comunica con sensores tipo TX.

El canal de comunicación de la aplicación puede modificarse por medio de los sw´s que contiene el sensor, lo esencial para su correcto funcionamiento es que los sensores trabajen en el mismo canal.

Para comenzar la aplicación ya instanciada en los sensores, se oprime el primer LED5 (RESET), al oprimir el botón LED1 este comenzará la transmisión de TX a RX de los paquetes, cuyo número depende de la variable TEST_NUM, que contiene el total de paquetes a enviar (1000). Las luces de los sensores encienden cada vez que se recibe un paquete.

Sí al iniciar la aplicación, las luces de los sensores TX y RX encienden al mismo tiempo indican que se encuentran sincronizados en el mismo canal de comunicación.

Una vez que se comienzan a mandar los mensajes se imprimen en la pantalla de la terminal del Sistema Operativo en que se trabaje, los parámetros se muestran el la Figura 2.7 y son:

⇒ ***N***: Número de paquete
⇒ ***L:*** longitud del mensaje.
⇒ ***LQ***: (Link Quality): Calidad del Enlace (en dBm).
⇒ ***CRC*** (Check Redundancy Ciclic): si llego un mensaje se imprime como bien.
⇒ ***Data***: Datos del mensaje.

```
N:980 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:981 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:982 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:983 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:984 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:985 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:986 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:987 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:988 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:989 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:990 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:991 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:992 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:993 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:994 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:995 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:996 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:997 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
N:998 L:18 LQI=-24 CRC=1 Data=000102030405060708090A0B0C0D0E0F1011
Good:   999/1000

Halt
```

Figura 2.7 Salida del UART durante el proceso de PER Test

El proceso que se realiza básicamente en un proceso unidireccional (Figura 2.8):

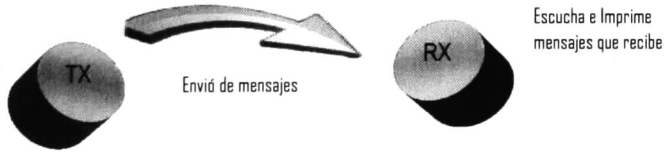


Figura 2.8 Diagrama de la aplicación PER Test.

La aplicación se basa en una máquina de estados que determinan las acciones que deberán tomarse para la realización de las tareas, según convenga. La máquina de estados para PER Test TX, se compone de los estados que se muestran en la Figura 2.9:

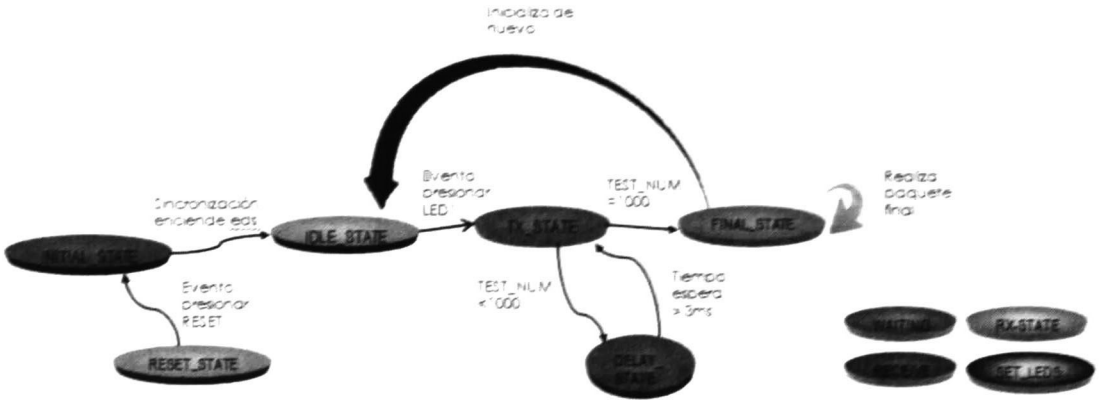


Figura 2.9 Máquina de Estados de PER Test TX

⇒ INITIAL_STATE, se inicializan algunas variables generales para comenzar a trabajar, enciende los leds para avisar que ya esta listo.

⇒ IDLE_STATE: inicializa radio, y configuración de canal, etc.

⇒ RESET_STATE: solo se llama cuando se oprime RESET a los sensores, para iniciar de nuevo la aplicación.

⇒ SET_LEDS: este estado es llamado por la función *MCPSDataIndication*, una vez que ya se emitió la información, para indicar que ya transmitió un paquete y con que potencia se envía. No se encuentra activo en la aplicación ya que no recibe ACK.

⇒ TX_STATE: es el encargado de formar los paquetes a enviar y comenzar su transmisión.

⇒ DELAY_STATE: retarda por un pequeño lapso, los envíos de mensajes.

⇒ RECEIVER_ALWAYS_ON: este estado se utiliza si se quiere poner siempre en escucha al sensor. No esta activo en esta aplicación.

⇒ WAITING_FOR_ACK: espera un tiempo para que se reciba un ACK. No esta activo en la aplicación.

⇒ RX_STATE: como lo indica su nombre, sirve para cambiar el estado en modo Receive. Tampoco se encuentra activo.

⇒ FINAL_STATE: encargado de terminar la aplicación una vez que termino de transmitir.

Mientras que la máquina de estados para PER Test RX, se muestra en la Figura 2.10.

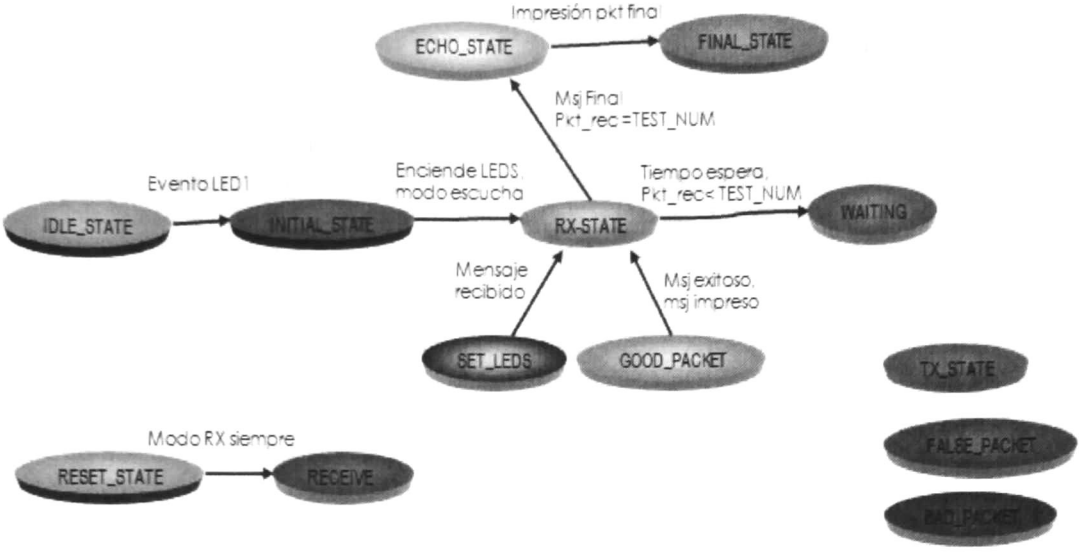


Figura 2.10 Máquina de Estados de PER Test RX

Los estados que tienen en común PER Test TX y PER Test RX, realizan las mismas tareas, la diferencia es que esta aplicación cuenta con algunos estados más y algunos estados no se encuentran definidos en ella.

Por ejemplo, el estado DELAY se elimina y los estados que se suman a esta aplicación se explican a continuación:

- ⇒ GOOD_PACKET: se utiliza cada vez que un paquete llega exitoso e imprime su contenido.
- ⇒ FALSE_PACKET: aunque está declarada en la máquina de estados y no se utiliza en la aplicación, este puede definir acciones si se detecta un paquete falso.
- ⇒ BAD_PACKET: al igual que el anterior, no se utiliza en la aplicación, pero contiene funciones si la información en el paquete no es la que se esperaba, o el paquete llega en mal estado.
- ⇒ ECHO_STATE: es llamado una vez que se detecta que llego el último paquete, imprime cuantos paquetes se recibieron en total.
- ⇒ RECEIVE_ALWAYS_ON, se encuentra activo en esta aplicación y TX_STATE no.

2.3 Síntesis

Freescale es la herramienta que nos facilita tanto los dispositivos como la plataforma que se utiliza. Se presenta la aplicación elegida para el proyecto "PER Test TX" la cuál nos permite tomar la parte de TX y adaptarla a las necesidades del proyecto, ya que su estructura nos permite comunicarnos con cualquier sensor a diferencia de la aplicación RX que sólo se comunica con el transmisor.

La aplicación PER Test TX trabaja mediante una máquina de estados que nos permite adaptar los cambios necesarios bajo la IDE de Codewarrior.

El lenguaje de programación utilizado es C/C++, y el protocolo de comunicación es SMAC, por las facilidades de comunicación que permite, y no requiere de una trama específica del estándar 802.15.4.

En el capítulo se muestra el funcionamiento general y la explicación de la aplicación sobre la cual se basa el proyecto para obtener mejoras.

CAPÍTULO 3

Algoritmo de Auto-Organización

Resumen. En este capítulo se presenta la implementación de una estrategia distribuida para la formación de una red ad-hoc de dispositivos inalámbricos móviles, sobre la cuál se desarrollará un algoritmo para la red de sensores.

Se describen las adecuaciones de un algoritmo de auto-organización para ser utilizado en los sensores Freescale MC1322X.

3.1 Descripción del Algoritmo de Auto-Organización

El algoritmo de auto-formación de la red basado en estrategias de auto-organización se ejecuta en cada sensor, cada nodo tiene una tabla de vecinos, en la que se almacena la siguiente información (*Id, Type, Role, Measure, VarCount*) y se describen a continuación:

- ⇒ *Id:* se almacena el id del nodo que envía la información, este campo es el único que no cambia, a diferencia de los demás, los cuales cambian constantemente de acuerdo a la información que se recibe.
- ⇒ *Type:* almacena el tipo de mensaje que recibe, para asegurarnos de mandar mensajes "ACK" y "B"
- ⇒ *Role:*es el role actual del nodo, se utiliza para decidir que rol va a tomar el nodo que recibe la información en base al rol del mensaje que se recibió o viceversa.
- ⇒ *Score:* nos ayuda a tomar la decisión de quien será el líder en el grupo.
- ⇒ *Measure:* la medición en este caso es la temperatura del sensor que se envía para obtener los promedios.
- ⇒ *VarCount:* esta variable se utiliza para saber que nodo ya envió su información al nodo líder y este pueda transmitirla al nodo Sink, una vez terminado el algoritmo de SO.

El algoritmo que se utiliza para la formación de la red reportado en [Olascuaga, 2011] es el siguiente:

Algoritmo1 *Self_Organization()*

```
if Neigs!=0 then
if Leaders = 0 then
ROLE ← Leader
else {existe al menos un leader}
        if ROLE = leader then
            Solve_conflict()
        else {si mi rol no es lider}
            if Leaders = 1 then
                ROLE ← member
                Verify_consistency()
            else {existe más de 1 leader}
                 ROLE ← Gateway
            end if
        end if
end if
else
    ROLE ←Any
```

Para explicar el algoritmo implementado en nuestro proyecto,se definen 2 tipos de mensajes existentes:

⇒ *BROADCAST* (**B**): se envía a todos los nodos cuyo alcance permita escucharlo.
⇒ *ACKBROADCAST* (**ACK**): se envía como respuesta a los mensajes broadcast.

El algoritmo una vez adaptado a nuestro trabajo, consiste en lo siguiente:

Ejecución para inicializar algoritmo auto-organización

```
1  ROLE←Any
2  Nodo_i TX_STATE (TYPE← BROADCAST)
3  if Nodo_j esta en rango_alcance Nodo_i then
4      Nodo_j TX_STATE(TYPE ←ACKBROADCAST)
5  endif
6  if Nodo_i recibe "ACKBROADCAST" then
7      agregar_vecino()
8  endif
9  if numvecinos > 0then
10     self_organization()
11 endif
12 if nodo_i recibe msj de nodo_k then
13     repetir pasos 1 al 11
14 endif
15 if nodo_i recibe msj nodo_j then
16     informacion_actualiza
17 endif
```

El nodo decidirá en base al rol y el score, que rol juega dentro del grupo. En caso de que llegue un mensaje con el role de leader y ya exista un nodo leader dentro del grupo, se realiza un procedimiento llamado *solve_conflict()*, que asegura que no existan 2 nodos leader en un mismo clúster; se explica más adelante con detalle.

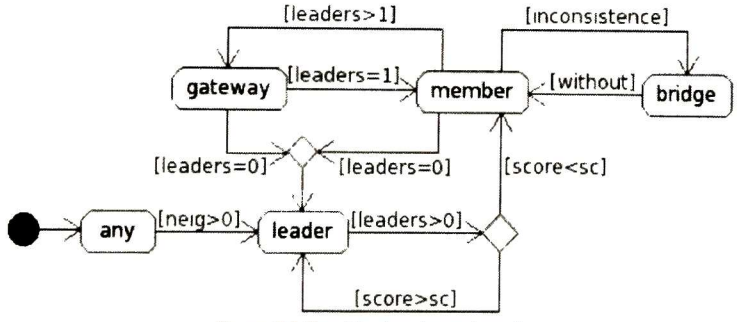


Figura 3.1 Elección de roles en la red

La elección de los roles es de la siguiente manera:

⇒ LEADER: este es el encargado de coordinar las actividades del clúster al que pertenece, sólo puede existir uno por clúster y su elección es en base al *score* más alto.

Leader = May (Score).

⇒ MEMBER: son los nodos más comunes, pues no realizan alguna tarea más que la de comunicarse con su respectivo leader. Este se selecciona una vez que ya existe un leader o cuando el leader ya no tiene la suficiente energía para mantener sus tareas, para ello renuncia a ser leader convirtiéndose en member, no sin antes asegurarse de que alguien más tome su lugar.

⇒ GATEWAY: es el encargado de establecer la comunicación entre diferentes grupos de clústeres por medio de los nodos leader. Se elige cuando en su rango de comunicación detecta a más de un nodo leader que pertenecen a diferentes clústeres.

⇒ BRIDGE: es el encargado de unir segmentos de red, es decir, cuando existe un nodo con el rol de member y dentro de su rango de comunicación detecta a otro nodo cuyo rol también es member y pertenece a un leader diferente, entonces se unen los dos miembros haciendo un lazo de comunicación, y ambos nodos cambian su rol a Bridge.

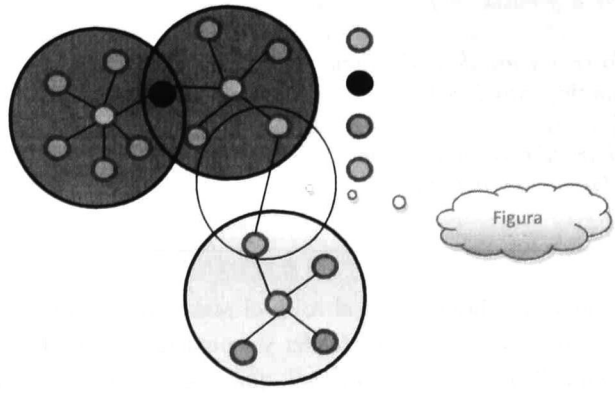


Figura 3.2 Roles en la Red.

En la Figura 3.1 se muestra un ejemplo de cómo queda definida una red una vez que se ejecuta el algoritmo, tomando en cuenta cómo reacciona cada nodo según el rol que juega en la red.

El paquete que se utiliza en [*Olascuaga, 2011*], es el siguiente:

Id	*Type*	*Role*	*Score*	*PTx*

Donde:

⇒ *Id*: es el id del agente.
⇒ *Type*: es el tipo de mensaje que se envía.
⇒ *Role*: se envía el rol del agente.
⇒ *Score*: es el score de cada nodo que se envía para realizar la decisión del rol.
⇒ *PTx*: es el poder de transmisión del mensaje

El *score* se calcula de la siguiente manera: $score=n + e$, donde la n es el número de vecinos y e es la energía residual.

La manera en que se resuelve el conflicto de la elección de leaders una vez que ya existe uno y otro nodo quiere tomar su lugar, es mediante la función *Solve_conflict()*, que es la encargada de verificar quien tiene el mayor *score* para que tome el rol de líder. El score se calcula en cada nodo una vez que se recibe un mensaje.

La aplicación perdura por un tiempo definido, reorganizándola cada vez que se realizan cambios, tales como, agregar, quitar o mover un nodo en la red.

El algoritmo para resolver el conflicto de los leaders es el siguiente:

***Algoritmo** Solve_Conflict( )*

If *score>scr* **then**
    ROLE ←MEMBER {cuando el score de un agente vecino es mayor}
else if *score =Scr* **then**
if *rx_packet.ID< ID* **then** *{menor ID}*
    ROLE ←MEMBER;
endif
endif

El algoritmo *solve_conflict()* no resolvía el conflicto cuando dos scores tienen la misma magnitud, debido a ello se realizó la siguiente modificación:

- ✓ Cada nodo contiene un score diferente.
- ✓ En caso de que dos nodos llegarán a tener un score de la misma magnitud, toma el role como leader el nodo cuyo ID sea menor.

La configuración en el ambiente CodeWarrior y en la terminal se explican en el Anexo D.

3.2 Implementación Del Algoritmo De Auto-Organización

Al iniciar con las modificaciones pertinentes al sistema, nos enfrentamos con algunos problemas tales como:

- ❖ Comunicación unidireccional.
- ❖ La estructura de la plataforma con la que trabajan los sensores era nueva para nosotros.
- ❖ Algunas funciones no explicaban su funcionamiento.
- ❖ No se tenía conocimiento de la funcionalidad del código, etc.

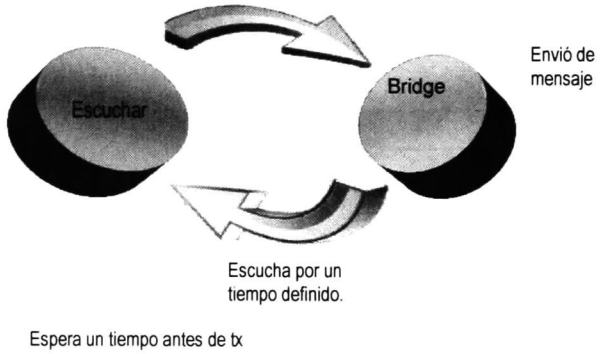
Para resolver el problema de la comunicación unidireccional primeramente, se agregaron y eliminaron estados de la aplicación PER Test Tx, ya que sólo tenía activos algunos y otros que se necesitaban no se encontraban en la aplicación. Los estados resultantes, los cuáles se explican con detalle más adelante, son los siguientes:

⇒ IDLE_STATE
⇒ INITIAL_STATE
⇒ TX_STATE
⇒ RX_STATE
⇒ GOOD_PACKET
⇒ SET_LEDS

Se verificaron funciones y procedimientos que se encuentran en la aplicación para modificarlos y permitir que la comunicación se realice en dos sentidos, es decir, Transmitir y Escuchar, ya que la comunicación era unidireccional.

Para definir la estructura de la implementación, se realizó una investigación y se analizó el código de la aplicación para entender los estados y funciones cuya forma de trabajar no se encontraba definida. Se les dio seguimiento para conocer cuál era su importancia, los detalles se resolvieron con estudio y algunos de ellos a prueba y error.

En la aplicación modificada los nodos una vez que envían el paquete por el canal de la radio, y otro dispositivo nodo lo escucha, inmediatamente envía la respuesta para hacer saber que este mensaje si fue recibido. Después de ello, pasa un tiempo en espera antes de volver a transmitir. El comportamiento se explica con el diagrama de estados de la Figura 3.3.



El paquete que se envía en la aplicación con los cambios realizados, tiene la estructura mostrada en la Figura 3.4.

Figura 3.3 Diagrama de Función de los Sensores

En donde:

- ✓ **ID** es el identificador del nodo.

- ✓ **TYPE** maneja 2 tipos de mensajes en este primer algoritmo, el primero de ellos es un mensaje de Hello "B", que es enviado para conocer el vecindario de cada nodo. El otro tipo de mensaje es de respuesta "ACK" TYPE Se utiliza para decidir que tipo de mensaje es el que se enviará, asegurándose que si se recibe un ACK entonces se envíe un "B" y viceversa.

- ✓ **ROLE** puede tomar los valores LEADER, MEMBER, BRIDGE ó GATEWAY. Este valor se determina conforme llegan los mensajes. Este cambia, si se recibe un mensaje con un mayor score, si la red se reconfigura, si llega un nuevo nodo, o si uno de los nodos muere en la red.

- ✓ [redacted] es el número de vecinos que tiene un nodo, su valor aumenta conforme el número de vecinos que se registren en un nodo. Sólo aumenta si el nuevo mensaje que llego es de un vecino diferente a los anteriores, sino sólo se refrescan los datos del ID al que pertenezca.

- ✓ [redacted] (Link Quality),es la calidad del enlace del mensaje que se recibió. Esta se calcula cada vez que un mensaje nuevo ha llegado.

- ✓ [redacted] Es la medida de la temperatura que los sensores calculan en cada ciclo, esta variable se envía solo como información en este primer algoritmo.

- ✓ [redacted] Se utiliza para asignar niveles a los sensores, es utilizado en el segundo algoritmo. El nivel se asigna en el algoritmo de auto-organización de manera paralela a los roles, se asigna cuando se asegura de que el padre sea un líder y con nivel más bajo al actual, a excepción del Sink, que contendrá el Level uno por default. Este campo cambia cuando el paquete que se recibe es enviado de un nodo líder y de menor nivel, o para asignar por primera vez un nivel.

- ✓ [redacted] Se utiliza para enviar el ID del nodo, hace referencia al nodo receptor quien envía la información. Una vez finalizado el algoritmo de auto-organización, se envía como bandera que sirve como confirmación de la condición de paro.

  Una vez que un nodo termina de recibir la cantidad de mensajes ACK que espera, cambia esta variable a una bandera para que un nodo al recibirla no espere más mensajes ACK o B y cambie las variables para comenzar con el algoritmo de propagación. Esto con el fin de ahorrar tiempo de espera, energía o evitar que algún nodo se quede esperando indefinidamente.

El score no se envía como parte del paquete debido a que se calcula una vez que llega el mensaje en cada nodo receptor. Los factores que tomamos en cuenta para el cálculo del *score* de un nodo *i* son los siguientes: LQ_k (Link Quality) calidad del enlace de un mensaje k y N_i es el número de vecinos del nodo *i*.

$$Score_i = LQ_k * N_i$$

Estos parámetros cambian en cada mensaje, debido a ello se calculan en cada ciclo de ejecución. La transmisión de cada mensaje se realizaba de manera manual al oprimir LED1, para ello, se tuvo que reconfigurar los sensores y utilizar este LED1 como timer, es decir, realiza la función de escuchar y transmitir de manera automática, sin la necesidad de oprimir cada vez que se quiera hacer una transmisión.

Los tiempos de escucha se realizan en determinado tiempo que es dado por el timer. Este se encuentra adaptado a cada sensor de manera diferente debido a su estructura, ya que si se utiliza el mismo rango de tiempo en los sensores, por su configuración hay mucha pérdida de mensajes.

La fórmula para el cálculo de cada timer en los sensores es la siguiente:

$$timer = (UINT32)(((rand()\%4)+1) * time);$$

Donde:

- ϕ *timer* es el tiempo total que el nodo se pondrá en modo escucha.
- ϕ *time* es la variable de configuración que se modifica en cada sensor.

Cada nodo tiene asignado un *id* fijo, los demás campos en el paquete se actualizan en cada envío y recepción de mensaje; esto permite que la información que se recibe sea actualizada.

Para la aplicación del algoritmo se tomó en cuenta si un nodo ya estaba registrado o no; de ser así sólo se actualizan los datos, de otra forma se inserta el nodo como vecino y se reconfigura la red. Los datos son tomados en cuenta para calcular el score, realizar la auto-organización, la propagación, etc.

Después de realizar las adaptaciones adecuadas sobre la aplicación PER Test TX, la máquina de estados y sus relaciones correspondientes se muestran en la Figura 3.5.

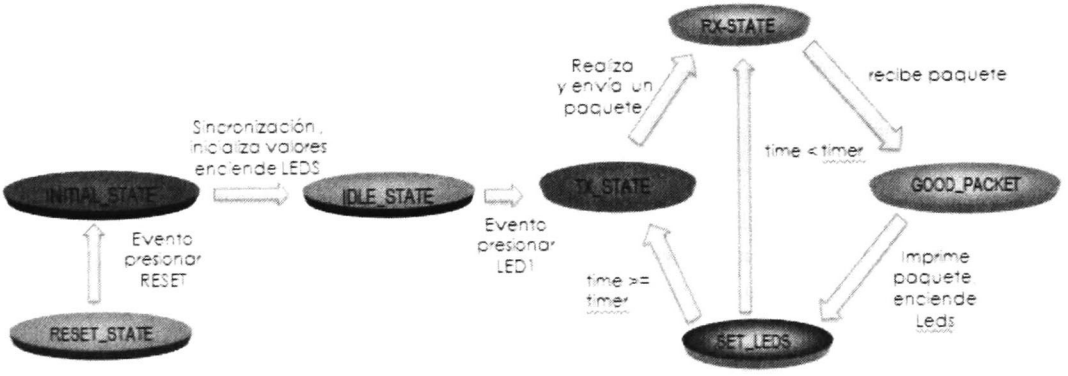


Figura 3.5 Máquina de estados Final.

Las funcionalidades de los estados sufren los cambios que se explican a continuación:

⇒ INITIAL_STATE: en este estado se encienden los leds para indicar que ya está listo y sincronizado el nodo; se enciende la radio.

⇒ IDLE_STATE: se inicializan variables, canal de comunicación, y espera por el evento LED1 para comenzar con el proceso de comunicación.

⇒ TX_STATE: en este estado se realizan 2 procesos; el primero es utilizado para el algoritmo de SO y el segundo es para el proceso de propagación. La diferencia con los procesos es la trama que se envía ya que esta cambia dependiendo del proceso actual de ejecución.

⇒ RX_STATE: es el estado el cual establece al nodo en modo escucha un determinado tiempo, en el que se pueden recibir uno o más mensajes. Este tiempo se encuentra diferente en cada sensor y es aleatorio, debido a la configuración de los sensores.

⇒ GOOD_PACKET: se encarga de mostrar los mensajes en una terminal, asignada por puertos COM a cada sensor, si estos tienen una conexión alámbrica.

⇒ SET_LEDS: es el encargado de verificar la calidad del enlace de cada uno de los mensajes que se reciben. Si se enciende el LED1 es porque se recibió el ACK. La intensidad con que llegue el mensaje, es el número de leds que encienden en el sensor, en la tabla 3.1 se muestran las relaciones de intensidad y encendido de LEDS.Una vez que se encuentra en este estado y no se ha terminado el tiempo de escucha, regresa al estado RX_STATE, en caso contrario a TX_STATE.

⇒ RESET_STATE: regresa a los sensores a su primer estado de inicialización, para reiniciar el proceso o detenerlo.

Casos/leds	LED1	LED2	LED3	LED4
Caso 0 -88				
Caso 1 < -80				
Caso 2 < -60				
Caso 3 < -40				

Tabla 3.1 Medidas en la calidad del enlace de mensajes recibidos.

De manera paralela con la modificación del autómata, también se modificaron algunas funciones, y se agregaron aquellas que eran indispensables para la ejecución del algoritmo.

La primera función Callball que se modificó fue *MCPCDataIndication* la cual, como ya se había explicado, es la encargada de verificar el estado que tomará un mensaje en el tiempo que llegue, es decir, las acciones que se realizan si el paquete llega en tiempo o fuera de él {TIMEOUT, SUCCESS}. En esta función se clasifican las acciones que se realizan con el tipo de mensaje que se reciben.

Si llega un mensaje en tiempo "*TIMEOUT*" se manda al nodo a retransmitir; en cambio si el mensaje llega de manera exitosa "*SUCESS*", se realizan diferentes modificaciones.

⇒ Si llega un mensaje de tipo "HelloB", cambia el tipo de mensaje a enviar "ACK" y se manda a transmitir este como respuesta al anterior.
⇒ Si llega un mensaje de tipo "ACK" se cambia el tipo para que se envié un mensaje "B", al recibir un mensaje "ACK", se utiliza una variable para llevar la cuenta de cuantos ACK tiene el sensor hasta ese momento, variable que se utiliza después como condición de paro.
⇒ Al momento de recibir un mensaje "ACK" se mandan llamar las funciones que se encargan de la auto-organización y se ejecuta hasta que la condición de paro se cumpla, siempre y cuando reciba mensajes ACK.

Las funciones que se agregaron en la implementación y permiten que se realice la auto-organización son las siguientes:

⇒ *Message_Broadcast(tRtpacketrx_packet)*; está función se encarga de verificar un mensaje ACK en la tabla de vecinos, si es la primera vez que se recibe este mensaje, se registra al nodo por medio de la información que se recibe, en caso contrario se actualiza la información correspondiente al nodo y se muestran los datos obtenidos.

⇒ *Self_Organization(tRtpacketrx_packet)*; es la función que se encarga de realizar el proceso de SO, cambia el rol conforme a la información que tiene registrada y que recibe; decide que rol juega un nodo dentro de un clúster.

⇒ *BFS(tRtpacketrx_packet)*; esta función es la encargada de realizar la asignación de niveles. El proceso comienza una vez que el Sink envía un mensaje a sus vecinos de donde se encuentra, mediante su nivel que siempre será uno. Un nivel sólo se podrá asignar en los siguientes casos:

- Se asigna cuando un nodo no tiene asignado nivel.
- Si un nodo ya tiene un nivel asignado, sólo podrá reasignar su nivel si se recibe un mensaje con un nivel menor en dos unidades al actual y el mensaje recibido tiene el rol de líder.

Los vecinos de un nodo N, sólo podrán aumentar si al recibir un mensaje ACK de otros vecinos, este no se encuentra registrado en la tabla de vecinos.

La tabla de vecinos se verifica cada vez que un nuevo mensaje se recibe, para asegurarse de que si ya existe el nodo se actualicen los datos o agregar al nodo como vecino.

Los sensores cuentan con 3 potencias de transmisión:

⇒ MAX_POWER: +3 a +5 dBm.
⇒ OUT_POWER: 0 dBm.
⇒ MIN_POWER: ~(-16bBm).

Se decidió ajustar los sensores a la mínima potencia, con el fin de que se pudieran formar diferentes redes en la etapa de auto-organización.

Cada sensor es el encargado de ejecutar el algoritmo para decidir que rol jugará dentro de la red. La variable que nos ayuda con la condición de paro en el algoritmo de auto-organización se modifica proporcionalmente al número de sensores que se tengan en la red.

La transmisión y recepción de los mensajes par el proceso de auto-organización se repite hasta que se cumpla la condición de paro (especificada en el capítulo 4) en todos los sensores o algún otro haya terminado y por lo tanto avise a sus vecinos para que estos al enterarse cambien las variables que permitan comenzar con el algoritmo de propagación.

3.3 Síntesis.

En este capítulo se presentó la implementación de una estrategia de auto-organización para la formación de una red ad- hoc.

Se describen los procedimientos realizados en la implementación; las modificaciones en las funciones y en la máquina de estados.

Se presentan también, las configuraciones que se llevaron a cabo con el fin de obtener una mejor funcionalidad en los sensores.

La implementación del algoritmo de auto-organización se realizó de manera exitosa, se logra la formación de diferentes redes ad-hoc, y se presentan los resultados obtenidos en las terminales, se explica como y más a detalle en el capítulo 4 y el caso de estudio como algunos resultados se muestran en el capítulo 5.

CAPÍTULO 4

Desarrollo de un algoritmo para una red de sensores

Resumen. En este capítulo se presenta un algoritmo para la integración de la información sensorial en una red de sensores inalámbrica. Se describen las configuraciones necesarias para ser implementado en los sensores Freescale MC1321X.

4.1 Estrategia General

El problema planteado consiste en la formación de una red de sensores en donde un nodo (Sink) estará encargado de recolectar los datos correspondientes a las mediciones realizadas. En este caso en particular las mediciones son la temperatura en la ubicación del nodo. Además se realizarán operaciones simples para calcular las temperaturas máxima, mínima y promedio en algunos nodos que forman la red.

El procedimiento de colección de datos debe operar sobre la red formada por la ejecución del algoritmo de auto-organización con el fin de ahorrar energía, es decir, una vez que se estabiliza el procedimiento de formación del backbone se inicia la colección y transmisión de temperaturas.

La estrategia utilizada es simple; se genera un árbol de manera paralela con la formación de la red tomando como raíz el nodo Sink. A través del árbol formado se realiza la transmisión a partir de los nodos hoja. Durante la propagación de la información se realizan los cálculos necesarios y se transmiten los datos por los nodos cuyo rol es leader o Gateway, hasta el nodo Sink.

4.2 Formación de la red de sensores

4.2.1 Formación del Árbol

La formación del árbol depende del backbone que se construye con el algoritmo de auto-organización. En el algoritmo propuesto se eliminan redundancias existentes en la red para la formación de las ramas del árbol, esto con el fin de ahorrar energía y hacer del árbol un medio para que fluya la información de manera directa hacia el nodo Sink.

Primeramente se define la jerarquía de los niveles y se asignan de la siguiente manera:

- ⇒ El nivel más bajo (nivel 1) corresponde al nodo Sink; a partir de él se comienza la formación del árbol.
- ⇒ El nivel mayor será entonces el que se encuentre a mayor distancia del nodo Sink y tenga nodos intermedios que jueguen los roles de Gateway o Leader. Las hojas del árbol corresponderán a los nodos con rol member.

La figura 4.1 ilustra lo anterior; aunque se muestra un árbol balanceado no necesariamente las hojas están al mismo nivel.

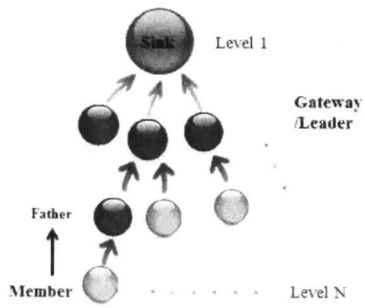


Figura 4.3 Asignación de Niveles

A continuación se presenta el algoritmo encargado de asignar los niveles a cada nodo el cual verifica que se haya formado el backbone. El procedimiento espera hasta que haya terminado el algoritmo de auto-organización.

Algoritmo Formación del árbol *(Paquete * rx_packet)*

If *rx_packet.father = LEADER and rx_packet.level <= milevel and rx_packet.level >0* **then**
level← rx_packet.level + 1
father← rx_packet.Id
existfather← TRUE
else if *existfather = FALSE* **then**
if *rx_packet.level < (milevel-1) and rx_packet.level >0 or milevel=0 and rx_packet.level >0* **then**
level← rx_packet.level + 1
father← rx_packet. Id
endif
endif

En el algoritmo de asignación de niveles se toma en cuenta lo siguiente:

⇒ Se debe respetar la jerarquía de los roles: los nodos leader, gateway y bridge pueden asignar niveles; los nodo member no pueden asignar niveles a menos que sea un nodo Sink.

⇒ El algoritmo toma en cuenta casos como:

- Se le puede asignar nivel a un nodo si al recibir un mensaje, éste ya tiene un nivel asignado, de lo contrario sólo se actualizan los datos en la tabla de vecinos, como se muestra en la figura 4.2.

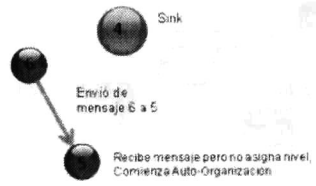


Figura 4.2 Mensaje Sin Nivel Asignado.

- Se le asignará un nivel a un nodo *i* si al recibir un mensaje de un nodo *j*, este ya tiene un nivel asignado, entonces se asigna a *i* un nivel mayor al que tiene *j* y el padre de *i* será el id del nodo *j* y sólo si el nodo *j* tiene el rol de LEADER.

- Una vez que ya se asignó un nivel a un nodo *i*, puede suceder que llegue un mensaje de un nodo *j*, con un menor nivel cuyo rol es LEADER, si esto llegará a suceder, entonces el algoritmo debe reasignar el nivel y el padre del nodo *i*, con base a los datos del nodo *j*. Como se muestra en la Figura 4.3.

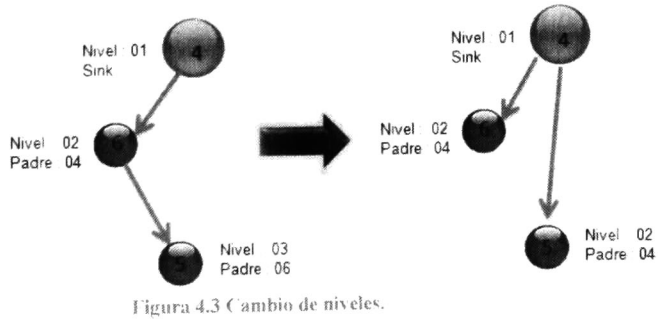


Figura 4.3 Cambio de niveles.

- Cuando existe un conflicto en un rol Gateway al comunicarse con 2 nodos leaders para asignarle nivel, se toma en cuenta el id de los nodos y se asigna como su padre el nodo cuyo id sea el más bajo, ejemplo de ello se muestra en la Figura 4.4.

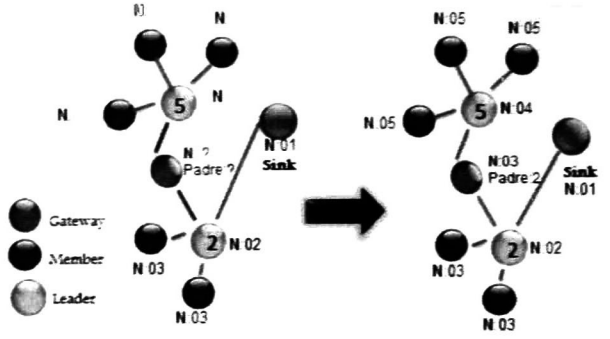


Figura 4.4 Conflicto en algoritmo de propagación en gateways

El proceso de la asignación de niveles se muestra en la figura 4.5. Los niveles se podrán reasignar durante un tiempo definido y después de ese tiempo comenzará la medición-colección de información.

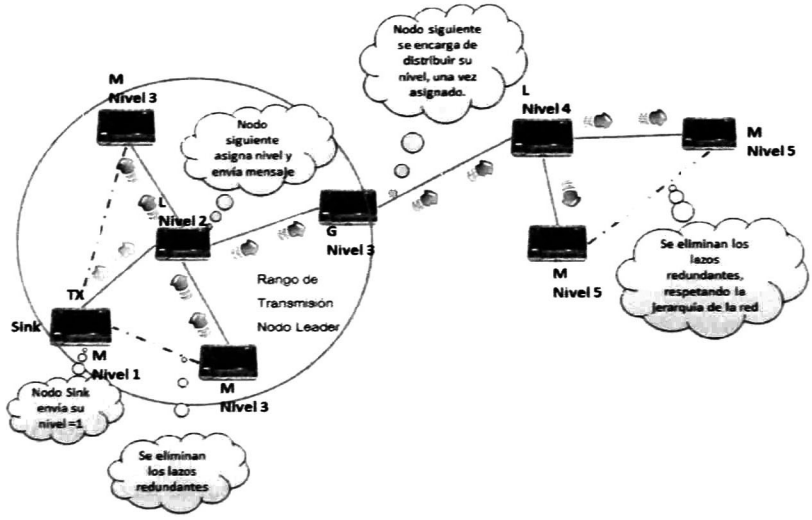


Figura 4.5 En la asignación de niveles se eliminan los lazos redundantes de la red.

4.2.2 Medición y propagación de la información hacia el Sink

La transmisión de la información una vez cumplida la condición de paro y la formación del árbol se realiza de la siguiente manera y se ilustra en la figura 4.6:

⇒ Los nodos de rol member (*nodos hoja*) realizan la medición y comienzan a transmitir, ya que no esperan información de algún otro nodo.

⇒ Los nodos intermedios (gateway y leader) realizan a medición, reciben información de los nodos hijos, efectúan los cálculos y transmiten el resultado del cálculo al nodo que sea su padre. Este procedimiento se efectúa después de cierto tiempo en modo escucha.

⇒ **La transmisión se realiza por intervalos de tiempo de los nodos hijos hacia el nodo padre.**

⇒ **El proceso formación de backbone-medición y colección de datos sensoriales se realiza periódicamente en un lapso de tiempo predefinido.**

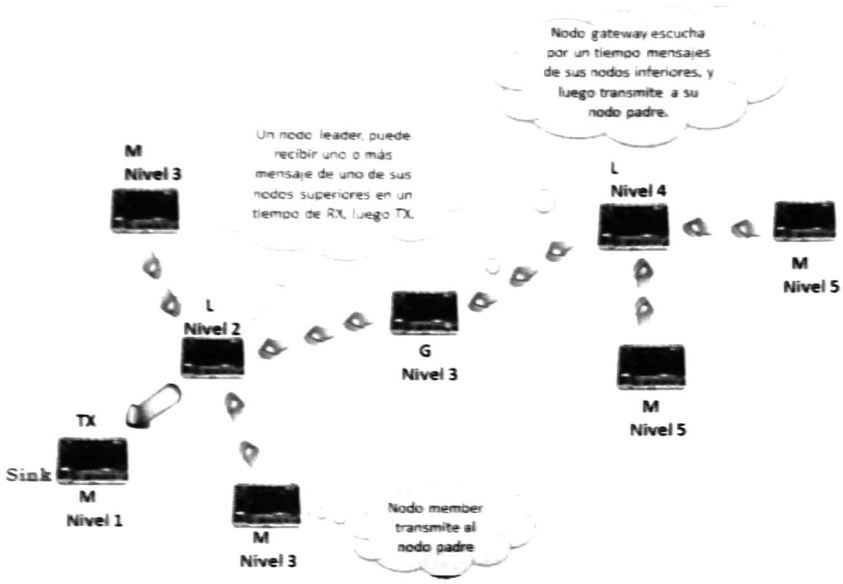


Figura 4.6 Transmisión de la información por los roles que juegan en la red.

A continuación se presenta el algoritmo encargado de realizar el proceso de transmisión de cada nodo según su rol:

Algoritmo *Propagación (tRxPacket *rx_packet)*

```
Imprime Role
Imprime LEVEL
1  if Sink = true and LEVEL=1 then //Primer caso
2       SiSink← true
3       if ID= rx_packet.Id then
4             estado← GOOD_PACKET
5             imprime Temp Promedio
6       else if
7             estado←RX_STATE
8       end if
9       Prom ← Temp
10 else if father ≠ 0 then //Segundo caso
11      if  ROLE =MEMBER and Sink =FALSE then
12            estado←TX_STATE
13      end if
14      if  ROLE =GATEWAY or  ROLE = LEADER then
15            estado←GOOD_PACKET
16      end if
17 else if SiSink =FALSE then
18      estado←SET_LEDS
19 endif
```

En el algoritmo anterior se describe el proceso presentado al inicio de este apartado y forma parte del algoritmo que se propone para la medición-colección de la información sensorial. La variable *estado* toma los valores: transmitir, recibir y procesar, los cuales describen el estado de la aplicación. *Sink* es una variable que determina si el nodo tiene el rol de Sink (*true*) antes de que comience la propagación y *SiSink* es una variable, manipulada por GOOD_PACKET la cual determina las acciones a realizar del nodo Sink.

El proceso que realiza el nodo Sink es independiente al rol que posee, y procesa los mensajes cuyo origen sea de sus hijos.

En el segundo caso del algoritmo se decide que acción se realizará tomando en cuenta el rol del nodo, estas acciones están definidas por la máquina de estados de la figura 3.5.

4.2.3 Medición de temperaturas

En la tarea de los sensores la cual consiste en tomar la temperatura y procesarla se realizan las siguientes actividades:

⇒ Los nodos member u hoja se encargan de medir la temperatura y enviar sus datos hacia su nodo padre.

⇒ Los nodos leader o gateways realizan su medición y después esperan por un tiempo mensajes de sus nodos hijos. Cuando lo reciben recalculan la temperatura promedio con su medición. Eventualmente actualizan las temperaturas menor y mayor para transmitirlas a su padre después de que finalice su tiempo de escucha.

⇒ El nodo Sink realiza la misma tarea pero no transmite, sino que muestra la información obtenida de la red. Un ejemplo de este ejercicio es el que se muestra en la Figura 4.7.

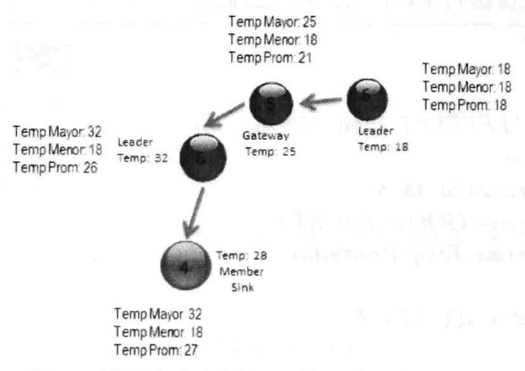


Figura 4.7 Flujo de la información en la red.

A continuación se presenta el algoritmo encargado de obtener los promedios de cada nodo y las mediciones mínima y máxima de la temperatura.

Algoritmo *Average(tRxPacket *rx_packet)*

```
if ROLE ≠ MEMBER or Sink ≠ 0 then
        Prom← Prom/2
end if
if rx_packet.Min < Min then
        Min ← rx_packet.Min
else if rx_packet.May > May then
        Min ← rx_packet.May
endif
```

Los valores de las temperaturas menor, mayor y promedio se inicializan con la temperatura actual del nodo; éstos van cambiando conforme a la ejecución del algoritmo y con los mensajes entrantes; una vez obtenidas la menor y mayor temperatura de 2 nodos se almacena y se envía como parte del mensaje. El Sink es el encargado de obtener los promedios finales, temperatura menor y mayor total de los nodos de los que recibió información.

El funcionamiento global del algoritmo de medición-colección de información sensorial se ilustra en la figura 4.8.

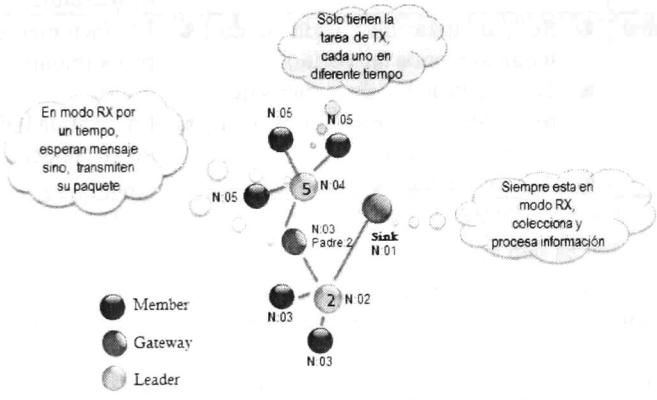


Figura 4.8 Tiempos de procesamiento en la red.

4.3 Implementación sobre MC1321X

En el algoritmo de medición-colección es necesaria definir la condición de paro. Para evaluar las posibilidades y la opción más viable se tomaron en cuenta las siguientes condiciones:

⇒ Cantidad de mensajes enviados.
⇒ Cantidad de mensajes recibidos.
⇒ Tiempo de ejecución en el algoritmo de auto-organización.

De las opciones que se plantean, se toman en cuenta algunas ventajas y desventajas de cada una de ellas las cuales fueron determinadas de manera experimental utilizando los sensores Freescale MC1321X. Estas se presentan en la Tabla 4.1.

	Ventajas	Desventajas
Mensajes Enviados	☯ Menor tiempo de espera. ☯ Menor cantidad de mensajes transmitidos. ☯ Ahorro de energía.	☯ No se asegura que los nodos vecinos escuchen los mensajes enviados, ya que los tiempos de escucha son aleatorios. ☯ La cantidad de mensajes enviados es proporcional a la cantidad de nodos. ☯ Los mensajes enviados no son suficientes.
Tiempo de Ejecución	☯ Reduce tiempo de espera. ☯ No tiene un control de número de mensajes. ☯ No gasta energía ni memoria en ello.	☯ No se conoce a priori el tiempo de formación de la red. ☯ Como los tiempos de escucha en los sensores son aleatorios, puede ser que no se realice la formación. ☯ Si es demasiado el tiempo, se puede gastar mucha energía. ☯ Control de tiempo en los sensores es inestable.
Mensajes Recibidos	☯ Se garantiza que cada nodo tenga al menos un vecino. ☯ La cantidad de mensajes recibidos es proporcional a la cantidad de nodos. ☯ Ahorro de energía.	☯ El incremento de mensajes es proporcional a la cantidad de nodos. ☯ La cantidad exacta de mensajes no se conoce.

Tabla 4.2 Evaluación de las Condiciones de paro

Se eligió como condición de paro la *cantidad de mensajes recibidos "ACK"*; se asegura que una vez que se cumple esta condición deben existir al menos, un nodo member y un nodo leader en la red.

Una vez que se cumple la condición de paro en un sensor, se activa una bandera que se transmite como parte del paquete en el campo ID_SENSOR (*IdToSendAck)* a los nodos hijos. Este campo indica que la condición se ha cumplido y el nodo que recibe el mensaje cambiará la estructura del paquete mostrada en la figura 4.9 para comenzar con el algoritmo de medición-colección.

Figura 3.4 Trama del algoritmo de Auto-Organización

Los campos del paquete utilizado en la medición-colección son los siguientes:

⇒ *ID, ROLE y LEVEL* almacenan los mismos valores correspondientes al paquete del algoritmo de auto-organización, es decir, id del nodo, rol y nivel.

⇒ *FATHER* se envía como parte del paquete para hacer referencia del padre actual del nodo.

⇒ *May, Min, Prom* son respectivamente las temperaturas mayor, menor y promedio calculadas en el nodo; éstas se envían para que los nodos que reciban el paquete realicen los cálculos correspondientes, actualicen sus datos y se transmitan al nodo padre siguiente.

⇒ *idToSendAck* se utiliza para identificar la procedencia del nodo que envía la información. La figura 4.10 ilustra el uso de este campo.

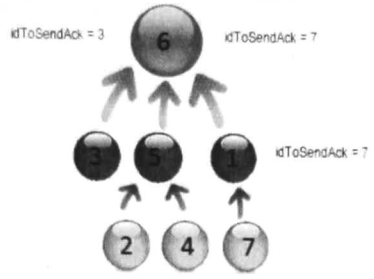


Figura 4.9 Paquete del Algortimo de medición-colección.

Durante los experimentos se presentaron algunos problemas relacionados con la configuración de los nodos con los controladores y los sensores para la medición de las temperaturas en el ambiente. En las primeras pruebas, para solucionar este problema se asignó un valor de temperatura fijo (remplazando la medición) a cada sensor, con el fin de verificar la funcionalidad al calcular las temperaturas *min, may* y *prom*.

El algoritmo de medición-colección se ejecuta de acuerdo a su ubicación en el árbol. Los nodos hoja ejecutan la medición y transmisión periódicamente, mientras que los nodos intermedios (leader y gateway) lo hacen cada vez que reciben un mensaje de sus nodos hijos, con el fin de ahorrar energía y evitar pérdida de mensajes. Las pruebas se reportan en el capítulo siguiente.

4.4 Operación

El algoritmo de medición-colección de información puede terminar su ejecución de dos maneras:

- ⇒ Por tiempo definido. El algoritmo se termina una vez que se cumple el tiempo que fue estipulado por el usuario.
- ⇒ Por elección del usuario. El usuario puede finalizar la ejecución del algoritmo antes de que se cumpla el tiempo establecido.

En ambas opciones, una vez finalizado el algoritmo de medición-colección el proceso de auto-organización comienza de nuevo, obteniendo posiblemente un backbone diferente.

4.5 Síntesis

La principal motivación para la realización de este algoritmo fue aprovechar la estructura que surge del algoritmo de auto-organización para el ahorro de energía. La formación del árbol sobre el backbone permite iniciar rápidamente la fase de medición-colección.

En este proyecto se realiza una implementación no sólo de la auto-organización, sino también de la propagación y recolección de la información a través de la formación de un árbol.

Con el fin de observar la funcionalidad en los sensores se implementó una aplicación que obtiene las temperaturas menor, mayor y promedio en los nodos intermedios que conforman la red. Estos nodos son gateways, leader y finalmente el propio Sink.

CAPÍTULO 5

Implementación y Pruebas

Resumen. En este capítulo se presenta el desarrollo de una red de sensores. Se describe la implementación de los algoritmos de formación de la red y de medición y colección de información sensorial. Se muestran las pruebas realizadas en diversos casos de estudio y se comentan los resultados.

5.1 Implementación de los algoritmos

Los algoritmos para la formación de red y las mediciones de temperatura se implementaron en los sensores MC1321X utilizando las facilidades de la plataforma BeeKit y la herramienta CodeWarrior. Los algoritmos fueron codificados en C/C++ y transferidos a los sensores a través de USB BDMMultilink como se muestra en la Figura 5.1. Las características de la plataforma Beekit y la herramienta CodeWarrior se describen en el Anexo B.

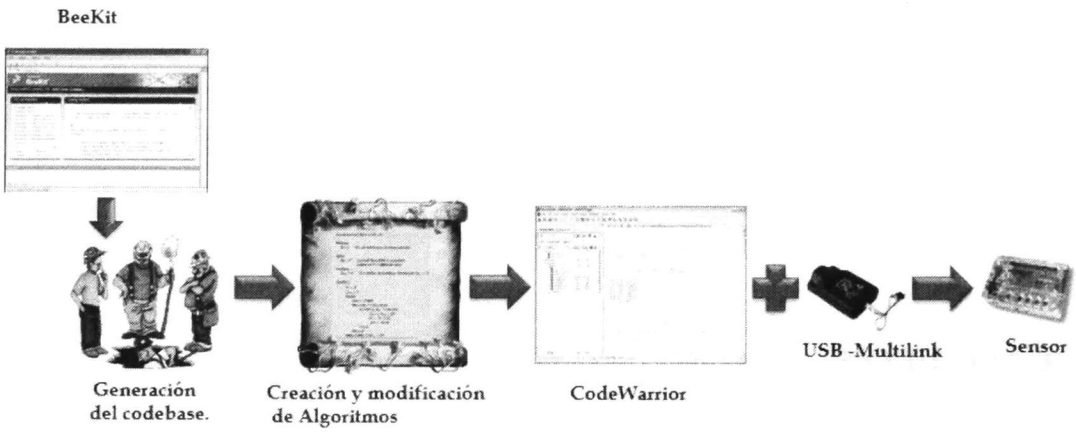


Figura 5.1 Proceso de generación y utilización de los sensores Freescale MC1321X.

En la implementación, el parámetro de la potencia de transmisión puede tomar tres valores constantes definidos como MIN_POWER, MAX_POWER y OUT_POWER.

Los sensores fueron ajustados al valor mínimo (MIN_POWER) con el fin de mostrar diferentes formaciones de redes dentro del laboratorio.

Los sensores operan con el uso de un temporizador, el cual permite que una vez terminada la transmisión el sensor se ponga en modo escucha durante un tiempo. Dicho tiempo es determinado mediante la función: *timer = (UINT32)(((rand()%6)+1) * time)*, en donde el parámetro *time* es especificado en ms y configurado con un valor diferente en cada sensor; razón por la cuál existe una pérdida mínima de mensajes, así como se explicó en el capítulo 3.

Debido a las temporizaciones que se llevan a cabo durante la ejecución, las mediciones de las temperaturas mínima, máxima y promedio se ven afectadas sólo en los tiempos de escucha. En este tiempo una vez que se recibe un mensaje se realiza la medición y los cálculos, los cuales se transmiten al nodo padre.

La cantidad de mensajes "ACK" que se utilizaron para la condición de paro es de quince.

5.2 Pruebas

Las pruebas se realizaron utilizando los 6 sensores funcionales disponibles, distribuidos en diferentes ubicaciones dentro del laboratorio de computación del CINVESTAV. La distancia aproximada entre cada sensor oscila entre 2 y 10 metros; esto es porque el rango de transmisión tiene una variación grande de acuerdo al ambiente en donde se encuentran los sensores.

Los factores que pueden alterar una buena comunicación en nuestras pruebas son:

⇒ Obstáculos (personas, equipos, objetos) que se encuentren entre la ubicación de los sensores

⇒ Condiciones propias de los sensores.

Para mostrar los resultados fue necesario conectar cada sensor a una computadora, de modo que permitiera observar las salidas correspondientes a cada sensor, debido a que no se contaba con un programa para captar y procesar las señales inalámbricas de los sensores.

5.2.1 Ejecución de los algoritmos

La tarea que realizan los sensores se ve modificada por diversas etapas a lo largo de la ejecución. El ciclo de funcionamiento de cada sensor se muestra en la Figura 5.2.

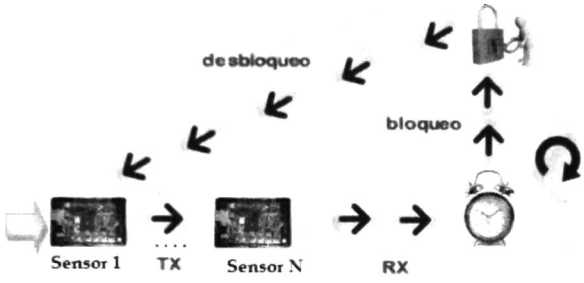


Figura 5.2 Etapas de los sensores en la transmisión.

Un sensor comienza la transmisión hacia los demás sensores que se encuentren dentro del rango de su alcance. Una vez que transmite, el sensor cambia a modo escucha por un tiempo que es aleatorio y después del tiempo de escucha el sensor se bloquea por un tiempo pequeño para luego volver a repetir el proceso.

El procedimiento se ejecuta en dos etapas. La primera de ellas es la formación de la red mediante el algoritmo de auto organización (Figura 5.3).

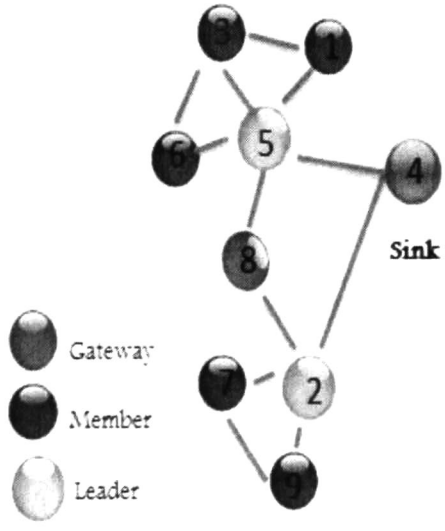


Figura 5.3 Primera fase, algoritmo de formación de red.

La segunda etapa es propiamente el algoritmo de medición y colección de datos de los sensores. Aquí se forma el árbol sobre el backbone y se realizan la medición y transmisión de los valores de temperatura medidos. Las hojas corresponden a los nodos member; éstos solamente miden la temperatura y la transmiten a sus nodos padre. Cada nodo con rol leader, Gateway y el propio Sink, además de realizar la medición, calculan la temperatura menor, mayor y promedio que se envían a los nodos padres (Figura 5.4).

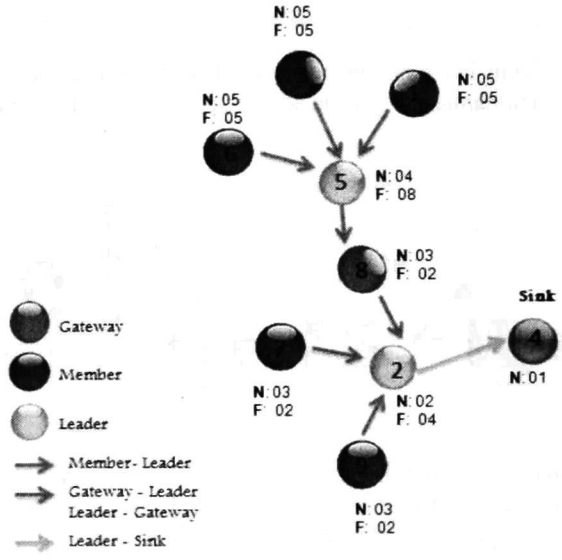


Figura 5.4 Eliminación de redundancias, formación del árbol

En la Figura 5.5 se muestra como es la transmisión de información en el árbol, una vez que comienza el algoritmo de medición-colección.

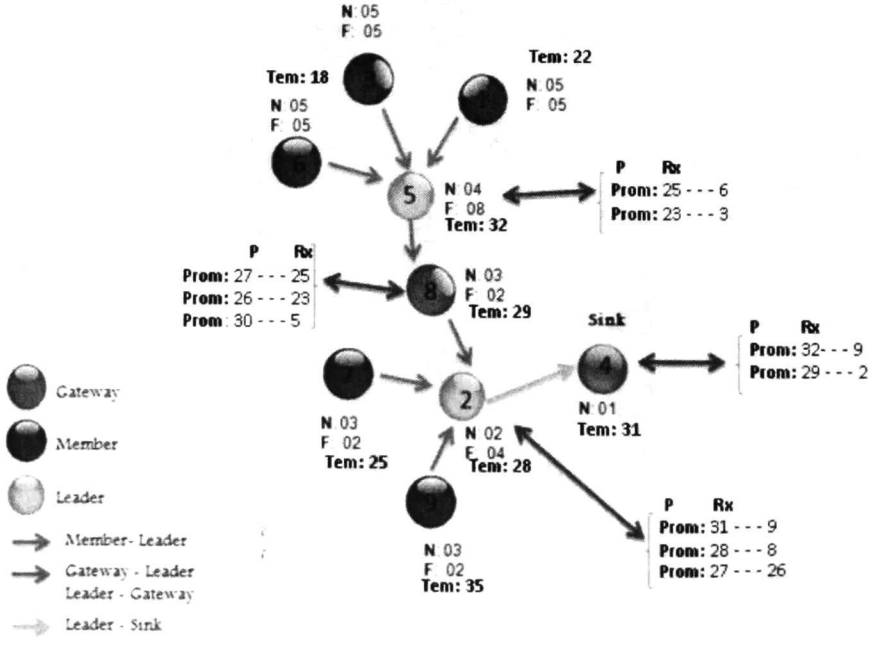


Figura 5.5 Propagación de la Información.

El cálculo de las temperaturas máxima, mínima y promedio se realiza como sigue:

1. Los nodos hoja realizan su medición y la envían a su padre; esta medición constituye la temperatura máxima, mínima y promedio del nodo.
2. Cada nodo intermedio (leader, gateway y Sink) recalcula la temperatura máxima y mínima en base a los valores recibidos de sus hijos, comparándola con su propio valor medido.
3. El valor promedio en cada nodo intermedio se calcula localmente con los valores recibidos de los hijos y el propio valor medido; este promedio se transmite a su padre. Esta forma de realizar el promedio se aproxima al promedio calculado mediante a un procedimiento aproximado global.

En la siguiente sección se muestran los resultados de la implementación. Se realizaron dos casos de estudio, en los que se logró la formación de diferentes redes. En este documento sólo se presentan tres formaciones de red diferentes.

Cada escenario se encuentra representado a una escala de 1:100.

5.2.2 Escenario 1

En este escenario la ubicación geográfica de los sensores se muestra en la figura 5.6.

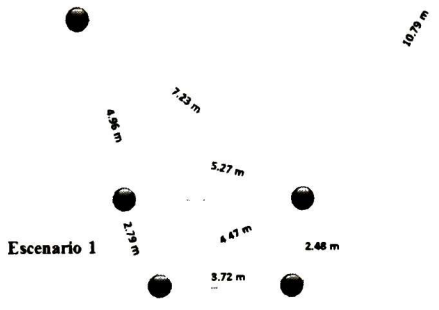


Figura 4.10 Distribución de idToSendAck al Sink.

Con este escenario se logró la formación de varias redes, pero algunas tienen una estructura similar, de manera que todo el conjunto puede ser representado con sólo dos, que se presentan a continuación.

Formación 1.1

La red que se formó, por su estructura es la red más simple ya que consta del nodo Sink que es a la vez el líder de la red y los miembros que son el resto de los sensores. La red, el árbol y los resultados se presentan a continuación.

ϕ Red

En esta primera red formada, se encuentran algunas redundancias entre los nodos que se muestran con líneas punteadas (Figura 5.7).

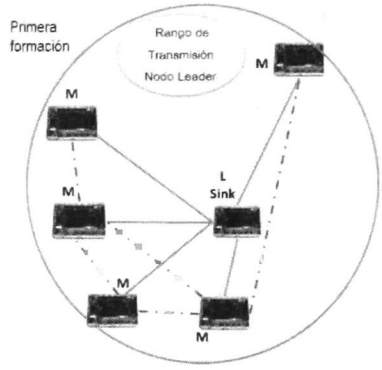


Figura 5.6 Posición Geográfica de los sensores en el primer escenario.

ϕ Árbol

En la formación del árbol las redundancias se eliminan y una vez que se termina la formación, se realiza la medición-colección de la información, mostrada en la Figura 5.8.

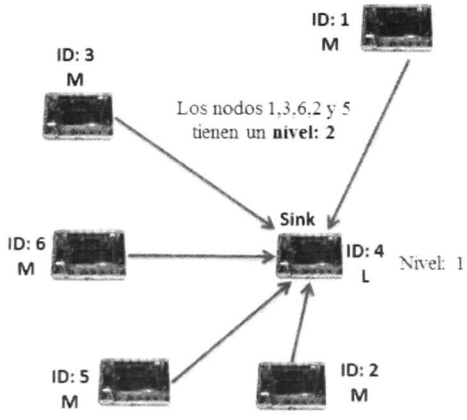


Figura 5.7 Visualización General de la primera formación de red.

ϕ Resultados

En la Figura 5.9 se muestra el resultado obtenido de la implementación en los sensores.

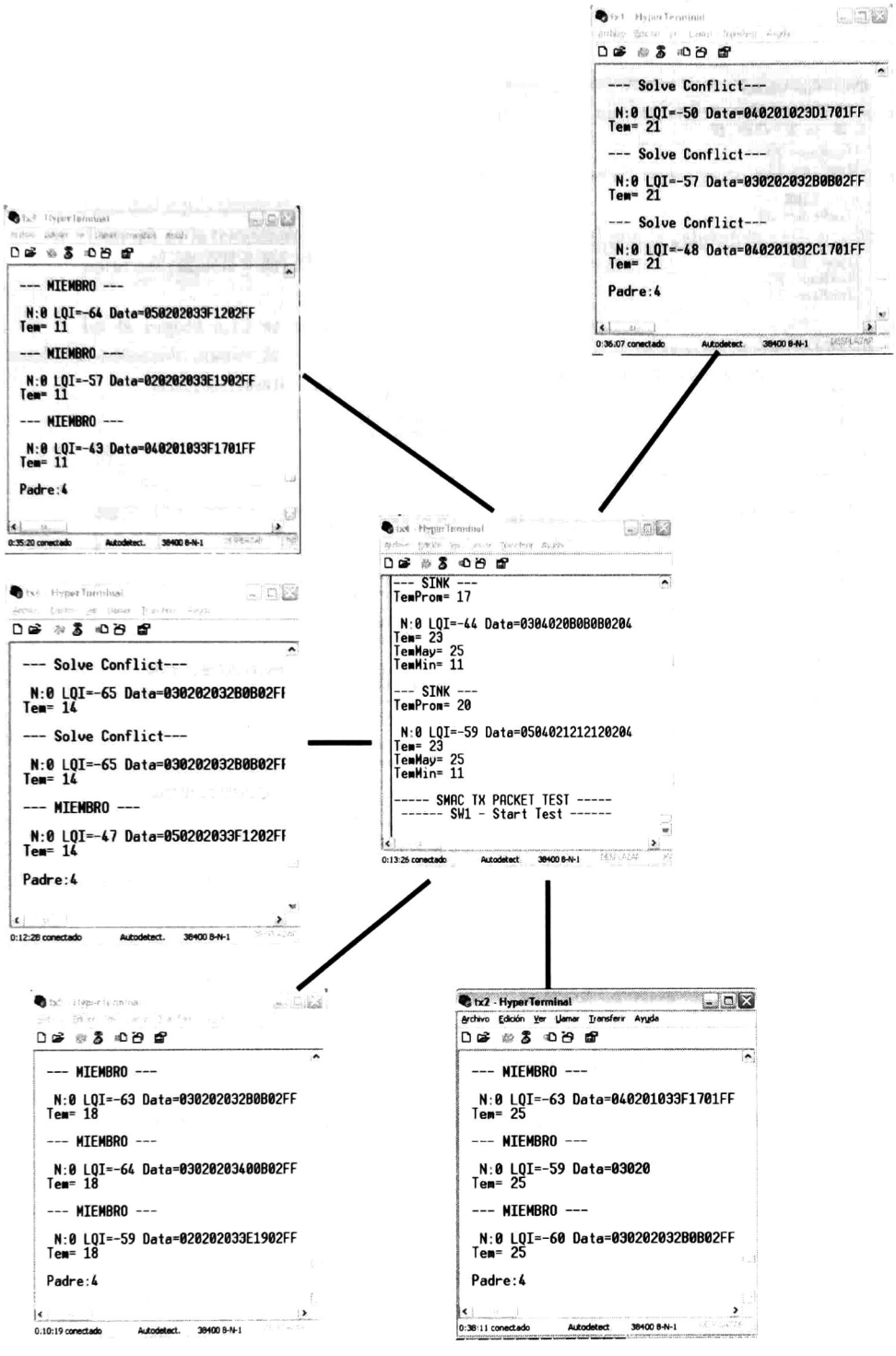


Figura 5.9 Primera formación de red.

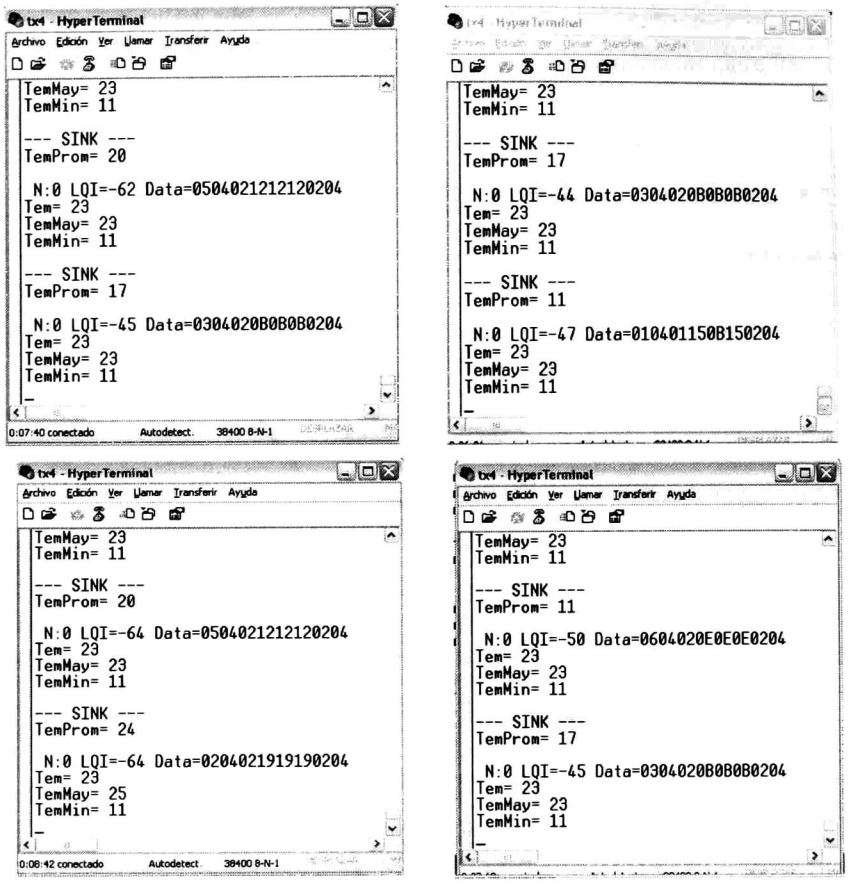


Figura 5.10 Algunas pantallas del Sink en el algoritmo de propagación, primera formación.

En la Figura 5.9 se muestran los resultados de cada sensor durante la fase de formación de la red y del árbol. En la Figura 5.10 se muestran algunos de los resultados obtenidos en la fase de medición-colección. El nodo Sink recibe información de todos los nodos. La información que se presenta en la implementación se explica a continuación:

En la Figura 5.11 se observan con detalle los datos que se muestran en la terminal:

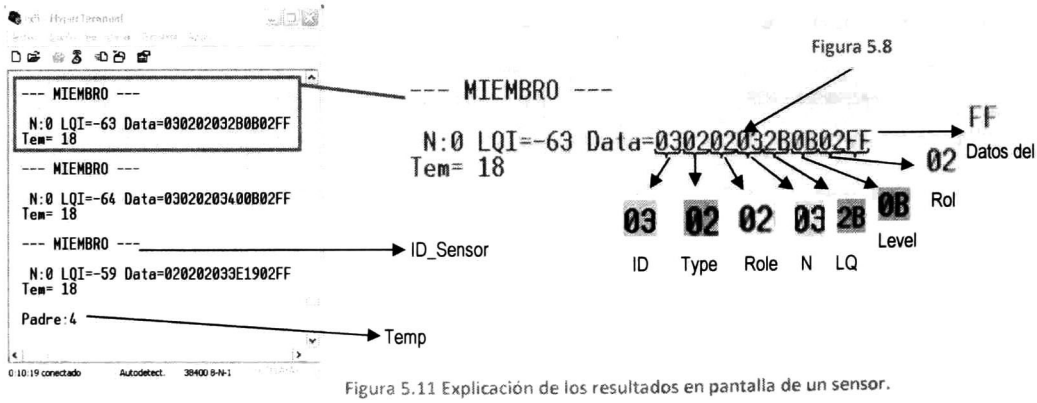


Figura 5.11 Explicación de los resultados en pantalla de un sensor.

⇒ **N:0** indica el número de mensaje que se transmite. En nuestro caso siempre será 0 debido a que en cada ciclo de transmisión sólo se manda uno.

⇒ **LQI= -63** indica la calidad de enlace del mensaje; se envía como parte del paquete y se muestra la cantidad equivalente en decimales.

⇒ **Tem:18** es la temperatura captada por el sensor; al igual que la calidad de enlace se transmite como parte del paquete y se muestra la cantidad equivalente en decimal.

En la Figura 5.12 se muestran los resultados que presenta el sink una vez que el algoritmo de medición-colección comienza.

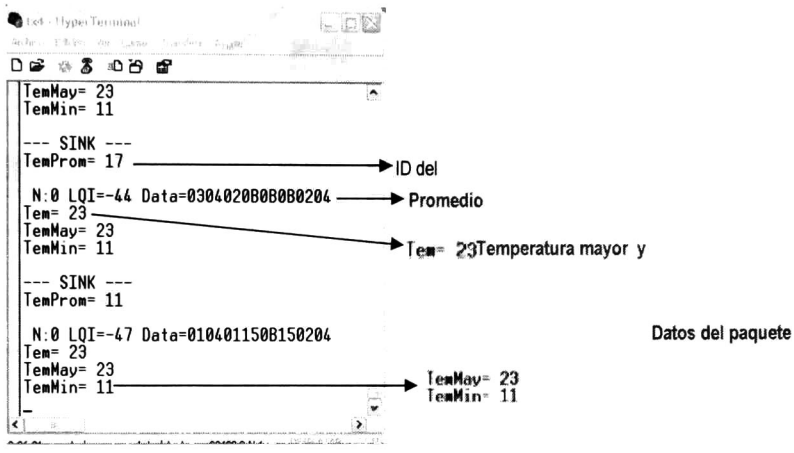


Figura 5.12 Información mostrada en el Sink.

⁂ Formación 1.2

Se presenta a continuación una segunda formación de red obtenida a partir del escenario 1.

ϕ Red

En este caso, se muestra una red distinta, en donde un nodo toma el liderazgo del clúster y el Sink juega el rol de member (Figura 5.13).

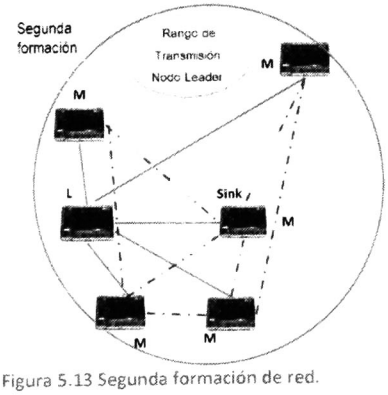


Figura 5.13 Segunda formación de red.

ϕ Árbol

El árbol de esta segunda formación muestra un esquema un poco más complejo que el anterior; en esta formación el nodo sink juega el rol de member en la red, el nodo leader colecciona la información que le envían sus nodos hijos y éste a su vez la transmite hacia el nodo Sink en el algoritmo de medición-colección. La formación del árbol se muestra en la Figura 5.14.

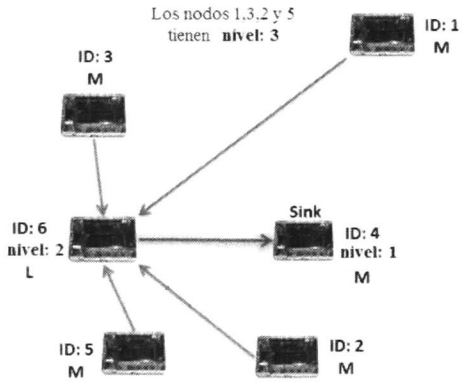


Figura 5.14 El nodo 6 es leader, recibe mensajes de los nodos 1,3, 5 y 2. El nodo 4-Sink recibe la información del nodo leader.

ϕ Resultados

Los resultados obtenidos en esta red tras la implementación se muestran en la figura 5.15.

Las pantallas se ubican al igual que la prueba anterior en una posición similar a la que se encontraban los sensores originalmente; las pruebas son satisfactorias. En la primera etapa del algoritmo la red se auto-organiza y se muestran los roles que toma cada nodo. Después los sensores que son miembros comienzan la transmisión hacia el nodo líder, quien a su vez envía la información al nodo Sink.

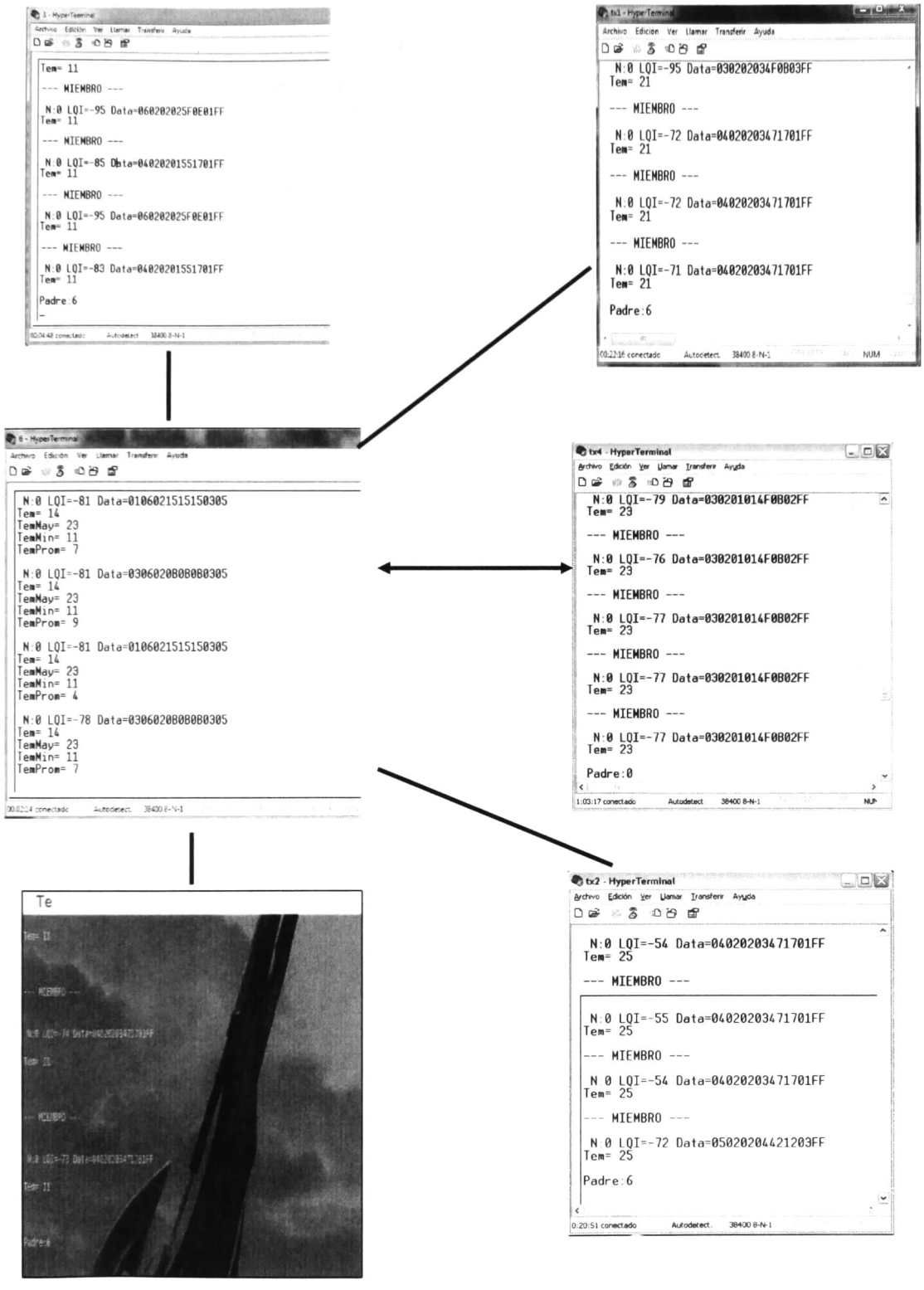


Figura 5.15 Segunda formación, resultados obtenidos en la implementación.

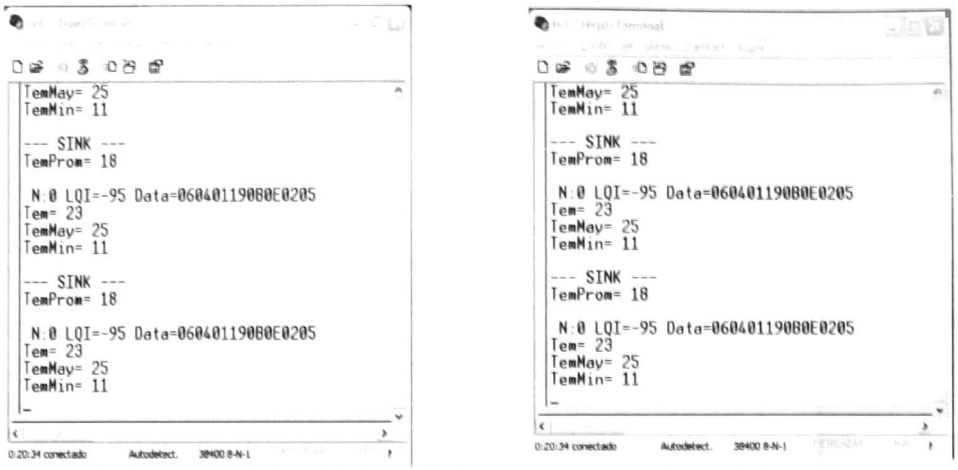


Figura 5.16 Algunos resultados del Sink en el algoritmo de propagación en la segunda formación

En las Figura 5.16 se muestran algunos mensajes que son enviados del nodo leader hacia el nodo Sink. La ejecución de los algoritmos termina después de un tiempo, para comenzar de nuevo con el proceso de auto-organización.

5.2.3 Escenario 2

En este segundo caso de estudio se muestra un acomodo de los sensores un poco diferente, con distancias relativas más diversas. En este escenario se muestra una red más compleja en estructura y funcionamiento. La distribución se muestra en la Figura 5.17.

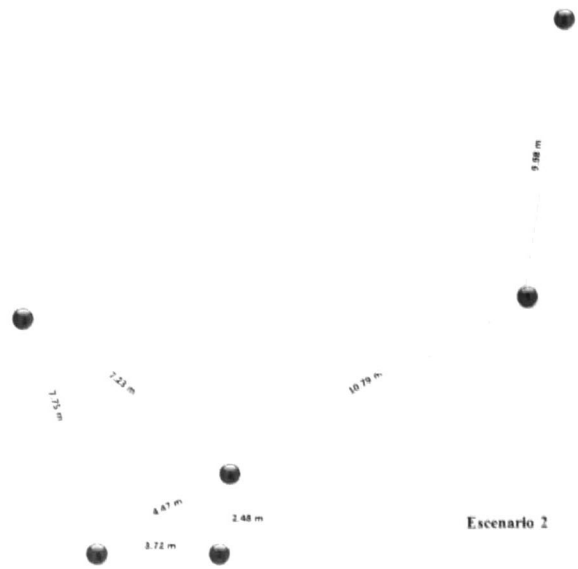


Figura 5.17 Segundo caso de estudio, ubicación geográfica de los nodos.

Formación 2.1

Red

En esta red se muestra el rol gateway que pertenece al nodo tres; por lo tanto dos nodos juegan el rol de leader: el Sink y el nodo seis. Los nodos uno, dos y cinco forman parte del clúster del nodo Sink (Figura 5.18).

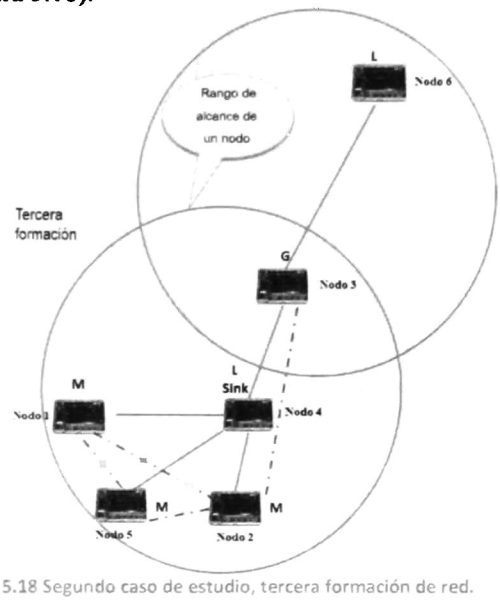


Figura 5.18 Segundo caso de estudio, tercera formación de red.

Árbol

En la Figura 5.19 se muestra la estructura de la formación del árbol y cómo es el proceso de transmisión de información una vez que comienza la medición-colección.

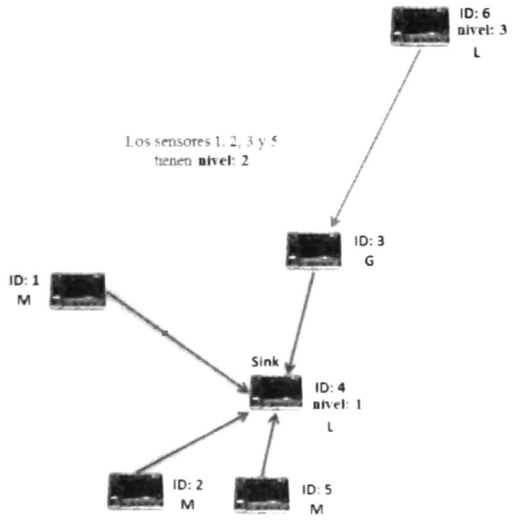


Figura 5.19 Backbone de la tercera formación.

ϕ Resultados

5

Figura 5.20 Resultado de la

En la Figura 5.20 se ilustra la evolución de la ejecución de ambos algoritmos en las terminales tanto de Windows como de Linux.

El nodo 5 cambia de rol durante la ejecución del algoritmo de auto-organización y se queda con el rol de member; esto se pueden apreciar en la Figura 5.21.

5

Una vez que termina el algoritmo de auto-organización, el nodo con el ID 3, cuyo rol es de gateway, tiene que escuchar por un tiempo definido para después transmitir al Sink; en la Figura 5.22 se muestra el comportamiento.

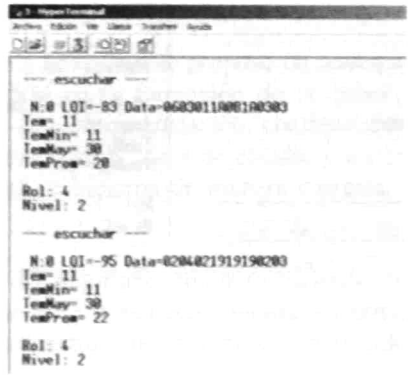


Figura 5.21 Nodo 5 cambia de rol a member.

Una vez terminado el proceso de auto-organización, el nodo Sink comienza a recibir información de los nodos que están asociados a él, realiza sus cálculos y se imprimen los resultados (Figura 5.23).

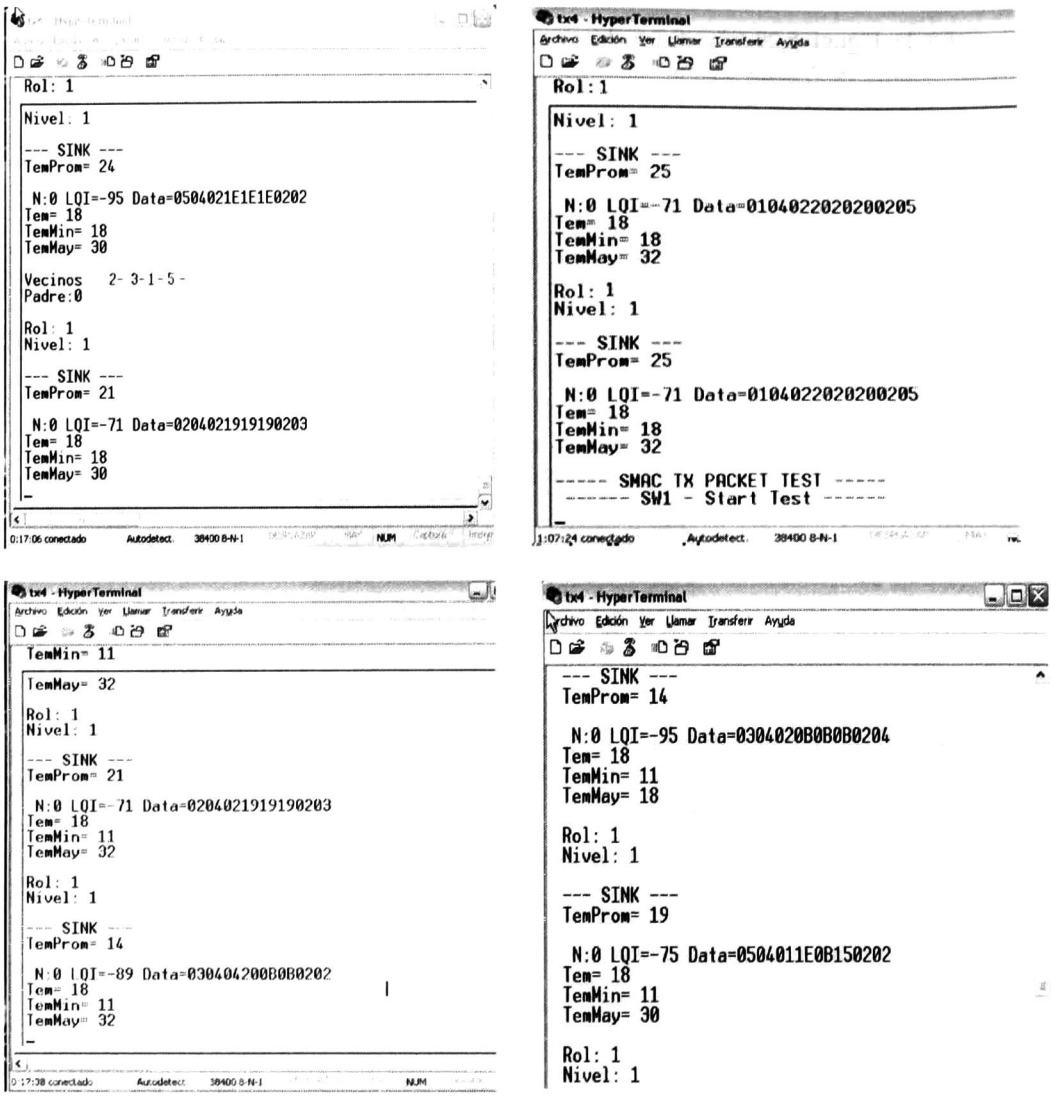


Figura 5.22 Nodo 3 Rol de Gateway.

5.3 Síntesis

En este capítulo se abordan 2 casos de estudio en los que se presentan 3 formaciones de red. En ellos se ilustran los algoritmos explicados en los capítulos 3 y 4. En los casos de estudio se mostró cómo la implementación resultó ser eficiente. Se obtuvieron los resultados deseados, de manera que cada nodo juega su papel de manera independiente y el nodo Sink colecciona la información de sus nodos hijos.

El algoritmo se mostró efectivo en el uso y ahorro de energía y permitió mostrar cómo a partir de una auto-organización puede realizarse un modelo más simple (árbol) para transmitir la información.

Conclusiones

En esta tesis se desarrollo una red de sensores utilizando los dispositivos Freescale MC1321X. La solución propuesta incluye la adaptación de un algoritmo de auto organización y el diseño de un algoritmo para la medición y ruteo de la información colectada.

Primeramente realizó un análisis de las Redes Inalámbricas de sensores a partir del cual se evaluaron trabajos realizados desde la época de los noventas hasta los más recientes. Como resultado de dicho estudio se detectaron las propuestas más interesantes disponibles en la actualidad y se tomaron como base para realizar nuestro proyecto.

Dentro del campo de estudio de las Redes de Sensores Inalámbricas, se centró el trabajo en algoritmos de auto-organización y encaminamiento de información. Del estudio realizado se extrajeron valiosas conclusiones a partir de las cuales se decidió con que sensores trabajar y el protocolo a utilizar para su implementación. Entre las características más destacadas están el ahorro de consumo de energía y la elección de la ruta de encaminamiento dirigida por los nodos cuya energía almacenada sea la mayor para prolongar el tiempo de vida de la red.

Uno de los objetivos en este trabajo ha sido incrementar la robustez en las comunicaciones para mantener la red operativa un mayor tiempo en presencia de fallos, ya que esta cuestión es decisiva en ámbitos de aplicación re redes inalámbricas en ambientes abiertos en la actualidad.

En esta tesis se realizó la adaptación e implementación de un algoritmo de auto-organización [Olascuaga, 2011]. Esto es de gran importancia, ya que permite manejar en la red ahorro de energía, robustez, reorganización autónoma, distribución con menor manejo de información a través de la red, flexibilidad, escalabilidad, comunicación acotada por el tiempo y seguridad.

Para la tarea de medición y colección se propuso un nuevo algoritmo de encaminamiento y recolección de información basado en la formación de un árbol, que aprovecha los resultados obtenidos a partir del algoritmo de auto-organización, continuando con el ahorro de energía y los beneficios que ofrece. Se realizaron varios casos de estudio y los resultados obtenidos cumplieron con lo esperado: las formaciones se hicieron de manera correcta, así como la transmisión de la información.

El tiempo de ejecución del algoritmo conjunto depende de dos factores: el usuario y la aplicación. Esta última mostró un buen desempeño durante las pruebas, en las cuales se definieron distintos escenarios y diversos parámetros que pueden ser modificados en trabajos futuros.

Como trabajo futuro, se pretende dar más flexibilidad en la implementación de ambos algoritmos para que permita adaptarse a cualquier ambiente, mejorar las mediciones toma de temperatura y ampliar la aplicación. Debido al tiempo y las herramientas con las que contábamos no se realizaron comparaciones con demás algoritmos, por ello queda como referencia a trabajo futuro. Habiendo cumplido todos los objetivos fijados, consideramos que el propósito final de la presente tesis ha sido alcanzado satisfactoriamente, concluyéndose por tanto la presente memoria.

Referencias

[***Aakvaag& Frey, 2006***] Niels Aakvaag Noruega, Jan-Erik Frey Sweden"Redes de Sensores Inalámbricas" Nuevas soluciones de interconexión para la automatización industrial. Revista ABB, 2006.

[***Andreu, 2011***] David Andreu, "*Implementation and Performance Evaluation of IEEE 802.15.4 Protocol.* "Master's Degree Project Supervisor: Pangun Park, Stockholm, Sweden 2011.

[***Arreola, 2012***] NicolásArreola, RedesInalámbricas de Sensores, SistemasDistribuidos, www.dsi.fceia.unr.edu.ar/downloads/distribuidos/material/monografias/RedesInalambricasSensores_1.pdf

[***Bandara, 2007***] H. M. N. D. Bandara and A. P. Jayasumana, "*An enhanced top-down cluster and cluster tree formation algorithm for wireless sensor networks.*"In Proc. ICIIS 2007, Sri Lanka, Aug. 2007, pp. 37-42.

[***Bandara, 2008a***] H. M. N. DilumBandara, Anura P. Jayasumana,"*Cluster Tree Based Self Organization of Virtual Sensor Network*" Department of Electrical and Computer Engineering, Colorado State University and Tissa H. Illangasekare Division of Environmental Science and Engineering, Colorado School of Mines This paper appears in: GLOBECOM Workshops, 2008 IEEE.

[***Bandara, 2008b***] H. M. N. D. Bandara, "*Top-down clustering based self-organization of collaborative wireless sensor networks*," Master's Thesis, Department of Electrical and Computer Engineering, Colorado State University, 2008.

[***Bragisnsky, 2002***] D. Braginsky and D. Estrin, "*Rumor routing algorithm for sensor networks*", In Proc. 1st ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, Sep. 2002, pp. 22-31.

[***Chan, 2004***] Chan, H.; Perrig, A. "*ACE: An Emergent Algorithm for Highly Uniform Cluster Formation*". In 2004 European Workshop on Sensor Networks 2004, 154-171.

[***Colonies, 91***] Colonies Alberto Colorni, Marco Dorigo, Vittorio Maniezzo. "*Distributed Optimization by Ant.*"Dipartimento di Elettronica, Politecnico di Milano Piazza Leonardo da Vinci 32, 20133 Milano, Italy. Appears in proceedings of ecal91 European Conference on Artificial Life. Paris, France, Elsevier Publishing, 134–142.

[***Di Caro, 1998***] G. Di Caro and M. Dorigo."*Antnet: Distributed stigmergetic control for communication networks.* Journal of ArtificialResearch, 9:317 – 365, 1998.

[***Equihua, 2009***] Edgar Alfredo Equihua González, "Red Inálalambrica de Relevadores de Estado Sólido" Centro de Investigación y Desarrollo de Tecnología Dígital, Tijuana Baja California, México, Diciembre 2009.

[***Freescale, 2008***] Freescale, Semiconductor, "Simple Media Access Controller", User's Guide, Document Number SMACRM Rev 1.5, 03/2008.

[*Hernández, 2011*] Carlos Hernández, "TDMA"
http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/hernandez_c_a/capitulo2.pdf

[*Hussein, 2003*] O. Hussein and T. Saadawi, "*Ant routing algorithm for mobile ad-hoc networks (ARAMA)*".In Proc. IEEE International Conference on Performance,Computing, and Communications, Apr. 2003, pp 281-290.

[*Ibnkahla, 2008*] Mohamed Ibnkahla, PhD, "*Wireless Sensor Network Projects*"PEngAssociateProfessorDepartment of Electrical and Computer Engineering Queen's University.

[*Lehsaini, 2010*] Lehsaini, M., Guyennet, H. and Feham, M. "*A Novel Cluster-based Self-organization Algorithm for Wireless Sensor Networks.*"LIFC – Laboratory of Computer Engineering of Franche-Comté University, Proceding.Journal.Sensor Networks, Vol. 7, Nos. 1/2, pp.85–94. February 2010.

[*Lewis, 2004*] F. L. LEWIS.,Wireless Sensor Networks Smart Environments: Technologies, Protocols, and Applications Ed. D.J. Cook and S.K. Das, John Wiley, New York, 2004 &http://arri.uta.edu/acs/networks/WirelessSensorNetChap04.pdf

[*Mayne, 2011*] Jordi Mayne, SILICA Aplicaciones IEEE 802.15.4/ ZigBee. Freescale.PDF.52767696-jordi-mayne.http://www.bairesrobotics.com.ar/data/ieee_zigbee_silica.pdf

[*Nieberg, 2004*] T. Nieberg and J. Hurink, "*Wireless communication graphs*," in Proc. 2004 Intelligent Sensors, Sensor Networks, Information Processing Conf., Dec. 2004, pp. 3

[*Olascuaga, 2011*] J. Guadalupe Olascuaga Cabrera, Ernesto LópezMellado, Andres Mendez Vazquez, and Félix Francisco Ramos Corchado, "*A Self-Organization Algorithm for Robust Networking of Wireless Devices*", CINVESTAVGdl, Zapopan, Jal.México.This paper appears in: IEEE Sensors Journal Vol. 11, No.3, Marzo 2011.

[*Palma, 2009*] Antonio Manuel Palma Gómez, "*Análisis de Protocolos de Enrutamiento Para Redes De Sensores Inalámbricas*", Universidad Carlos III De Madrid Escuela Politécnica Superior, Departamento De Teoría De La Señal Y Comunicaciones, Diciembre, 2009.

[*Parekh, 2002*] Abhay Parekh, Hierarchical Routing", Topincs in Routing.http://robotics.eecs.berkeley.edu/~wlr/228a02/Lecture%20Slides/routing3.pdf

[*Park, 2007*] Sungyun Park Kwangecheol Shin, Ajith Abraham and Sang Yong Han, "*Optimized Self Organized Sensor Networks.*" School of Computer Science and Engineering, Chung-Ang University 221, Heukseok-dong, Dongjak-gu, Seoul 156-756, Korea.Center of Excellence for Quantifiable Quality of Service (Q2S), Norwegian University of Science and Technology, Norway.Sensors, Molecular Diversity Preservation International , Switzerland, Volume 7, Issue 5, pp. 730-742, 2007.

[*Reference Manual, 2010*] 1322X Reference Manual Freescale Company, Document Number 1322XSNRM, Rev. 1.5, 2010

[*Ríos & Alvino, 2011*] Rovetto Ríos, Carlos Alvino, " *Métodos basados en Redes de Petri para el diseño de algoritmos en encaminamiento Adaptativos Libres de Bloqueo*" Universidad de Zaragoza,Departamento de Informática e Ingeniería de Sistemas, Septiembre 2011.

[***Romero y Muños, 2011***] Eduardo Luis Romero Sauceda, Jesús Israel Muños Padilla, "*Enrutamiento Jerárquico*", 2011 http://es.scribd.com/doc/54747607/Enrutamiento-Jerarquico.

[***Salem, 2009***] K.Saleem, N.Fisal, S.Hafizah,S.Kamilah and Rozeha A. Rashid, "*Ant Based Self-organized Routing Protocol For Wireless Sensor Networks*" Faculty of Electrical Engineering, UniversitiTeknologi Malaysia. Proceeding in International Journal of Communication Networks and Information Security (IJCNIS) Vol. 1, No. 2, August 2009.

[***Serna, 2007***] Jesús Serna Sánchez. "*Redes de Sensores Inalámbricas*", 2007.

[***Shin, 2006***] Shin, K.; Abraham, A.; Han, S. Y, "*Self-Organizing Sensor Networks Using Intelligent Clustering.*"Lecture Notes in ComputerScience2006, 3983, 40-49.

[***Singh, 2004***] G. Singh, S. Das, S. Pujar, and S. Gosavi, "*Ant Colony Algorithms for Steiner Trees: An application to Routing in Sensor Networks*," IGI press, 2004.

[***Soto, 2011***] Diana Carolina Soto, Luis Carlos Montoya, Andrés Felipe Ceballos, "Comunicaciones *Móviles*" Universidad Piloto De Colombia, Julio 2011. http://es.scribd.com/doc/60719812/1-TDMA

[***Swaszek and Willet, 1995***] P F. Swaszek and P. Willet, "*Parley as an approach to Distributed Detection*", IEEE Trans.On Aerospace and Electronic System, vol. 31.No.1, pp 447-457, Jan, 1995.

[***Tanenbaum, 2003***] Andrew S. Tanenbaum, "Redes de Computadoras", 4taedición, Editorial Pearson, Prentice Hall, *VrijeUniversiteItAmsterdam, TheNetherlands* 2003.

[***Tanenbaum, 2003a ,Douglas 1995***] Redes Centralizadas, Facultad Regional de Córdoba Departamento de Ingeniería en Sistemas de Información http://www.google.com.mx/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=0CFUQFjAD&url=http%3A%2F%2Fwww.profesores.frc.utn.edu.ar%2Fsistemas%2Fingcura%2FArchivos_Redes%2FEnrutayCCong.htm&ei=ps7yT-yNMuPS2gWT1KHOCw&usg=AFQjCNEU7UYy9SYBt1EP0nRiBrvk_iDs9w

[***WSN, 2010***] Wireless Sensors Australia is an independent Australian distributor of wireless networking products to systems integrators, resellers and end users in Australia, New Zealand, Asia and the Pacific.http://wirelesssensors.com.au/index.php?option=com_content&view=article&id=75:rfd-and-ffd&catid=50:faq&Itemid=91

[**Zhang, 2004**] Y. Zhang, L. D. Kuhn, and M. P. J. Fromherz, "*Improvements on Ant Routing for Sensor Networks*," M. Dorigo et al. (Eds.): ANTS 2004, Springer-VerlagBerlin Heidelberg 2004, vol. LNCS 3172, pp. 154-165, 2004.

IMÁGENES

[***Aakvaag& Frey, 2006***] Figura 2.1 Niels Aakvaag, Jan-Erik Frey "Redes de Sensores Inalámbricas" Nuevas soluciones de interconexión para la automatización industrial. Revista ABB, 2006.

[*CISCO, 2011*] Figura 1.2 CISCO: conmutación. "Principios de diseño de redes jerárquicas" Capítulo 1, 2011.

[*CISCO, 2011*] Figura 1.3 CISCO: conmutación "Principios de diseño de redes jerárquicas" Capítulo 1, 2011. http://es.scribd.com/doc/56290077/15/Principios-de-diseno-de-redes-jerarquicas

[*Ibnkahla, 2008*] Figura1.4 Mohamed Ibnkahla, PhD. "*Wireless Sensor Network Projects*"PEngAssociateProfessorDepartment of Electrical and Computer Engineering Queen's University.http://wsncanada.ca/index.php?page=turtles-at-risk

[*IPFire, 2012*] Figura1.1 IPFire. Org, 2012.
http://wiki.ipfire.org/_media/de/networking/networking_networktopologies.png

[*Reference Manual, 2010*] Figura 2.21 322X Reference Manual Freescale Company, Document Number 1322XSNRM, Rev. 1.5, 2010

ANEXOS

ANEXO A Sensores Freescale MC1321X

Algunas de las características más importantes con la que cuentan los sensores son las siguientes:

- ❖ Operación de banda 2.4 GHz ISM
- ❖ 16 canales seleccionables
- ❖ Transmisor programable, output power (-30 dBm a +4 dBm)
- ❖ < -96 dBm typical receiver sensitivity using DCD mode (<1% PER, 20-byte packets)
- ❖ < -100 dBm typical receiver sensitivity using NCD mode (<1% PER, 20-byte packets)

•Aceleración de hardware para aplicaciones 802.15.4
- ⇒ MAC accelerator (sequencer and DMA interface)
- ⇒ Advanced encryption/decryption hardware engine (AES 128-bit)

• Soporte del estándar IEEE 802.15.4 con una tasa de datos de 250 kbps.
- ⇒ 32-bit ARM7TDMI-S CPU core con un desempeño programable arriba de 26 MHz (24 MHz typical)
- ⇒ Extensive on-board memory resources
- ⇒ 128 Kbyte serial FLASH memory (will be mirrored into RAM)
- ⇒ 96 Kbyte SRAM
- ⇒ 80 Kbyte ROM

- Disipación de energía en Best-in-class.
- ⇒ 22 mA typical RX current draw (DCD mode) with radio and MCU active
- ⇒ 29 mA typical TX current draw with radio and MCU active (coin cell capable)
- ⇒ 3.3 mA typical current draw with MCU active (radio off)
- ⇒ 0.8 mA typical current with MCU idle (radio off)
- ⇒ 0.85 μ A typical Hibernate current (retain 8 Kbyte SRAM contents)
- ⇒ 0.4 μ A maximum Off current (device in reset)

• Control y variación del modo sleep extensible.
- ⇒ Hibernate and Doze low power modes
- ⇒ Programmable degree of power down
- ⇒ Clock management
- ⇒ Onboard 2 kHz oscillator for wake-up timer.
- ⇒ Optional 32.768 kHz crystal oscillator for accurate real-time sleep mode timing and wake-up with a possible sleep period greater than 36.4 hours
- ⇒ Wake-up through programmable timer, external real-time interrupts, or ADC timer

- Dedicated 802.15.4 modem/radio interface module (RIF)
- ⇒ Dedicated NVMSPI interface for managing FLASH memory
- ⇒ Two dedicated UART modules capable of 2 Mbps with CTS/RTS support
- ⇒ SPI port with programmable master and slave operation

• Características del sistema de protección
- ⇒ Detección de batería baja
- ⇒ Ver el gtimer (COP)

⇒ Sleep mode timer

•Optional buck converter for better battery life
• -40 °C to +105 °C en rango de temperatura

La interfaz de BeeKit (*Graphical User Interface* **GUI**) permite crear, modificar, y actualizar implementaciones de redes inalámbricas.

Los dispositivos mencionados soportan los protocolos Simple MAC, (SMAC), el protocolo IEEE 802.15.4 MAC, Protocolo de Pila Synkro™ y el Protocolo de Pila BeeStack™ ZigBee.

Los sensores están compuestos por un módulo de bajo consumo de potencia, un transceptor RF de 2.4 GHz, un núcleo con una arquitectura de ARM7 de 32- bits y un acelerador de hardware para aplicaciones inalámbricas.

Una fuente de poder controlada de 3.3 Vdc alimenta al microprocesador y al módulo sensorial, por ello puede funcionar con las baterías.

MC1321X son microcontroladores que permiten incorporar una plataforma completa para soluciones inalámbricas y de control, ofrecen un poder de procesamiento superior en aplicaciones inalámbricas debido a su núcleo ARM7TDMI-S de 32 bits, que puede operar a más de 26 MHz.

En Freescale el Controlador Simple de Acceso al Medio (22xSMAC) es una pila simple de códigos basada en ANSI C disponible como un ejemplo de código fuente. Puede usarse SMAC para desarrollar transmisor RF usando MC1322X.

Algunas de las características de SMAC son:

- Solo Lectura - Código: ~5 Kb
- Solo Lectura – Datos: ~0.5 Kb
- Solo Lectura – Datos: ~2.5 Kb
- Transferencias de Datos de 10 bytes sin bloqueos.
- Bajo consumo de potencia
- Enlace de comunicación RF Bidireccional
- Soporta capa de seguridad AES128 para una transmisión/recepción segura.
- Programación sobre el aire OTAP(Over the Air Programmer).
- Utiliza 2k Flash
- Código fuente y ejemplos de aplicaciones
- Compartible con MC1319x, MC1320x, and MC1321x
- ANSI C basado en el núcleo HCS08
- Puede ignorar paquetes del estándar 802.15.4

ANEXO B Herramientas Para los Sensores Freescale MC1321X

Freescale BeeKit Wireless Connectivity ToolKit (FBWCT)

Freescale es una empresa que produce y diseña sistemas embebidos, contiene tanto software como hardware para redes, incluye productos como micro controladores, micro procesadores, procesadores de señales digitales, sensores, etc.

Freescale BeeKit Wireless Connectivity Toolkit, es la plataforma en la cual se generan los proyectos apropiados y estructuras de los códigos fuentes de los estándares IEEE 802.15.4, SMAC y BeeStack de Zigbee (*BeeKit Wireless Connectivity Toolkit User's Guide* BKWCTKUG) estas proyecciones se ligan a otra de las herramientas utilizadas CodeWarriorv6.1 para la programación y manejo de la tarea que realizarán los sensores.

Para comenzar con la aplicación buscamos las aplicaciones de SMAC, seleccionamos del Menú Archivo/ Select CodeBase (Figura B.1).

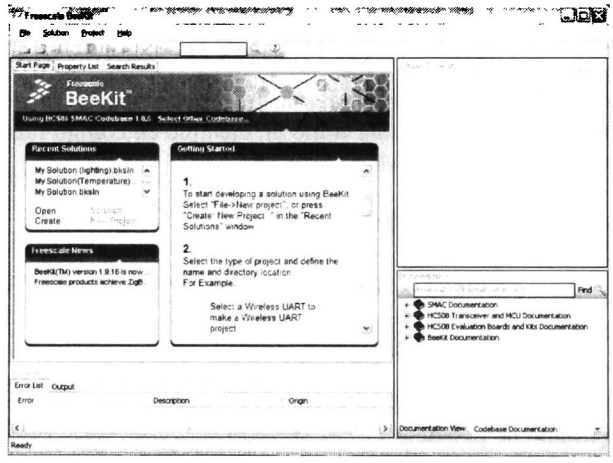


Figura B.1 IDE FreescaleBeeKit

⇒ Damos clic a HCS08 SMAC Codebase/Aceptar.

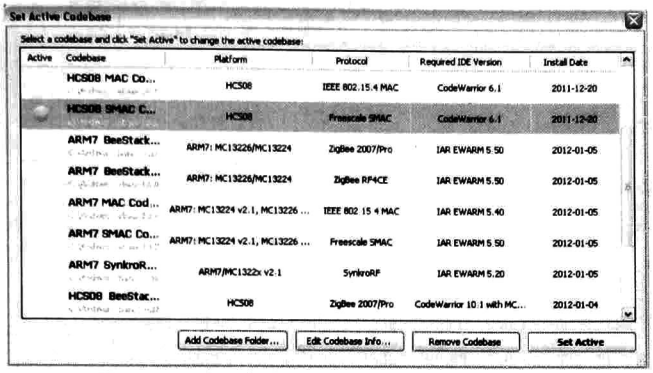


Figura B.2 Selección de HCS08SMACCodebase.

⇒ Se elige la aplicación que se utilizará, en este caso es PER Test TX, Aceptar y enseguida se generarán las librerías como códigos necesarios para comenzar con todo el procedimiento (Figura B.3).

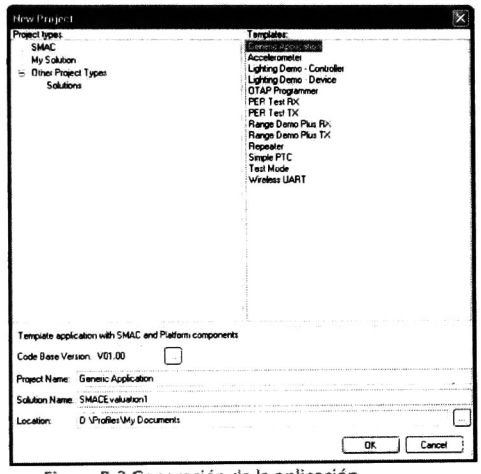


Figura B.3 Generación de la aplicación.

⇒ A la aplicación, damos clic izquierdo Add Project como se muestra en la Figura B.4, o Export Solution Code Warrior. Enseguida se generará la aplicación y se abrirá el proyecto en la ventana de CodeWarrior para comenzar a utilizarla.

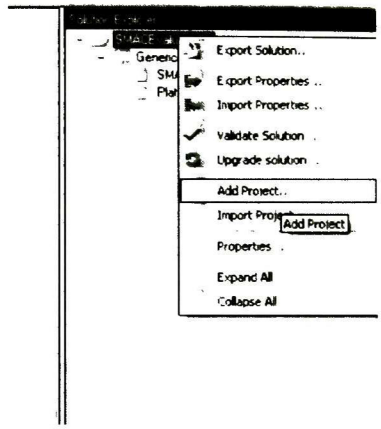


Figura B.4 Exportación de BeeKit a CodeWarrior.

Freescale MC1321X PiP es una solución robusta y flexible para aplicaciones y control de redes de sensado, tiene un alto rendimiento en microcontroladores de 32 bits, ofrece un amplio espacio y soporte para ejecutar las aplicaciones que requieren almacenar batería. Tiene periféricos extendidos, incluyendo más de 64 múltiples interfaces seriales GPIO y un 12-bit ADC, que conectan a una variedad de dispositivos.

CodeWarrior

CodeWarrior (CW) IDE es un completo sistema para editar, compilar y enlazar, simular, programar y depurar para los DSC de Freescale, en la Figura B.5 se muestra el IDE de trabajo principal.

El IDE de CW soporta tres tipos de lenguajes: Assembly, C y C++ también es posible hacer la mezcla de ellos en el mismo proyecto.El IDE es una aplicación de software que integra la mayoría de las herramientas de programación en una única pieza de software.

Con frecuencia se proporciona el protocolo de comunicación con objeto de utilizar los recursos disponibles dentro de los límites especificados y que ningún elemento esté energizado, si no es imprescindible. El trabajo se reduce a activar y desactivar sensores con la temporización apropiada.

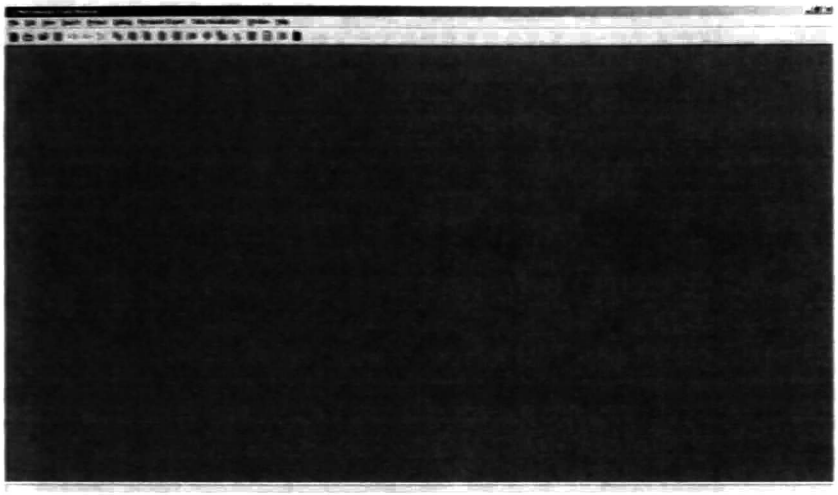


Figura B.5 IDECodeWarrior.

Para comenzar a trabajar con la aplicación, al abrir el IDE de CW, oprimimos Menú Archivo/ Abrir/Buscamos la ruta de la aplicación que vamos a modificar, y una vez encontrada, seleccionamos el archivo con extensión *.mcp*, en este caso es *PER Test TX.mcp* (Figura B.6).

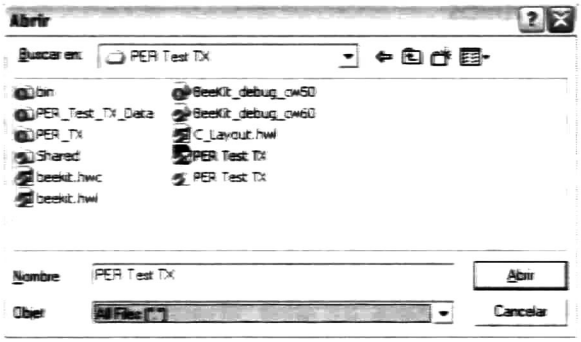


Figura B.6 Seleccion de la Aplicacion PER Test.

Al abrirse la aplicación, se elige el código principal, para trabajar, regularmente se encuentra en la Carpeta que se llama igual a la aplicación, para nuestro caso es PER_TX 'Sources/*smac_per_tx.c*, como se muestra en la Figura B.7.

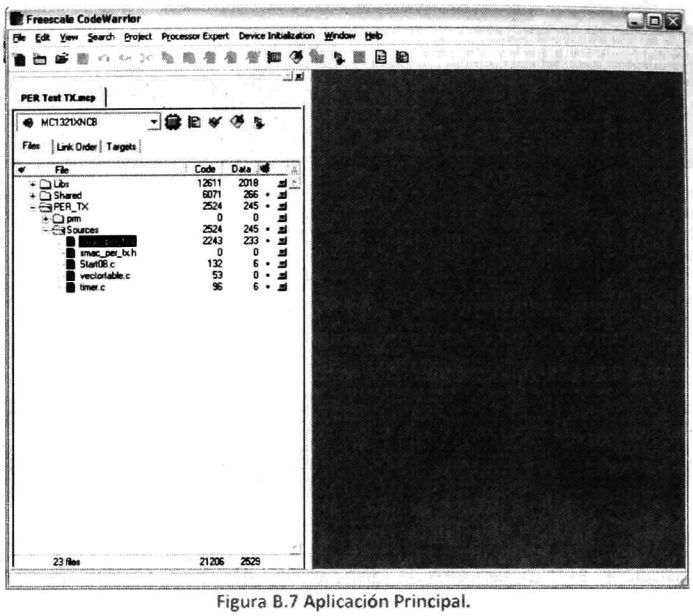


Figura B.7 Aplicación Principal.

Terminal de Dispositivos

La terminal de dispositivos, se utiliza para observar el proceso de comunicación de los dispositivos en una pantalla, es un medio útil para configurar y probar el módem o examinar la conexión con otros sitios o dispositivos. Para nuestro trabajo se utiliza la HyperTerminal de Windows o la terminal de Linux (Figura B.8).

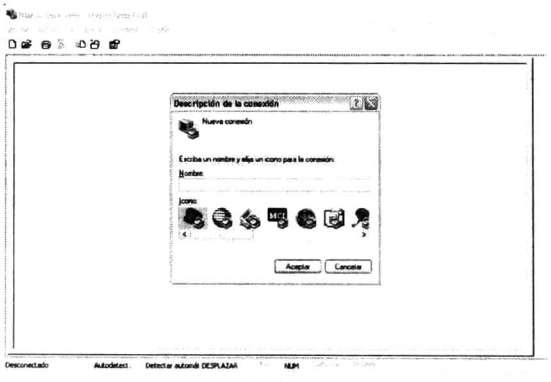


Figura B.8 IDE de la HyperTerminal.

ANEXO C Primitivas de SMAC

Se presentan a continuación las primitivas utilizadas con su funcionamiento y parámetros utilizados:

⇒ **MCPSDataRequest**: esta función es usada para enviar un paquete.
- El prototipo es: *UINT8 MCPSDataRequest(tTxPacket *msg);*msg es el paquete a transmitir.
- Regresa: *Success.*

⇒ **MCPSDataIndication:** esta función es llamada CallBack, se encuentra localizada en la aplicación del usuario, y es necesitada por SMAC, esta función se llama al recibir un dato y es procesado por la aplicación.
- El prototipo de la función es: *Void MCPSDataIndication(tRxPacket *gsRxPacket);* el parámetro *gsRxPacket* es un puntero donde se recibe el dato.
- No regresa valor.
- Trata a un paquete que llega, con uno de los siguientes valores que son:
  SUCCESS — si se recibe paquete, verifica el TYPE del mensaje para determinar que acción realizar.
  - *TIMEOUT* — Ocurre si el receptor esta encendido por una cantidad de tiempo. Ver función MLMERXEnableRequest.

⇒ **MLMESetChannelRequest**: configura la frecuencia actual en que el radio transmite y en la cuál recibe.
- El prototipo es: *UINT8 MLMESetChannelRequest(UINT8u8Channel)*; espera un canal, con valores entre 0 y 15.
- La función regresa uno de los siguientes valores:
  Success: Si el canal está entre los solicitados,
  Error: si el canal es mayor a la cantidad esperada.

⇒ **MLMERXEnableRequest**: pone al radio en modo recibir sobre el canal que fue seleccionado de la función MLMESetChannelRequest ().
- El prototipo de la función es: *UINT8 MLMERXEnableRequest(tRxPacket *psRxPacket, UINT32u32Timeout)*; Los parámetros son
  - *tRxPacket**: Localización del buffer donde se almacenará el paquete.
    UINT32:valor del timeout de 32 bits
- Los valores que regresa son: *Success ó Error.*

⇒ **MLMEMC13192PAOutputAdjust**: esta función ajusta la potencia de salida del transmisor. Por medio de parámetros como NOMINAL_POWER, MAX_POWEROMIN_POWER.
- El prototipo es: *UINT8 MLMEMC13192PAOutputAdjust(UINT8u8PaValue);* donde el valor esperado es de 0-15.
- Regresa: *Success ó Overflow.*

⇒ **MLMELinkQuality:** lee la calidad del enlace del último mensaje recibido.
- El prototipo es *UINT8 MLMELinkQuality (void);* regresa el valor calculado.
- La función de cálculo es la siguiente: **dBm** = (-Link Quality/2).

⇒ **MLMEMC13192ResetIndication():** función CallBack, también necesitada por SMAC, esta se ejecuta cuando Soft Reset a ocurrido.

- El prototipo es igual a la función.
- Inicializa el proceso de nuevo.

⇒ **MLMEMC13192PAOutputAdjust**: esta función ajusta la energía de salida del transmisor.

- Prototipo**:** *UINT8 MLMEMC13192PAOutputAdjust(UINT8u8PaValue);* el valor esperado es entre 0-15.
- Regresa**:**

  Sucess: cuando se realiza exitosamente.

  Overflow: cuando el valor que recibe es mayor a 15.

ANEXO D Configuración del Sistema de Ambiente Codewarrior a los Sensores Freescale MC1321X

Para cargarle el programa a los sensores se realiza lo siguiente:

Al terminal la programación o realizar una modificación y ver la funcionalidad en los sensores, se realiza la compilación del archivo, el USB BDMMULTILINK debe estar conectado al puerto serial del sensor y al puerto USB de la PC, para que esta interfaz pueda transmitirle los datos de la PC al sensor.

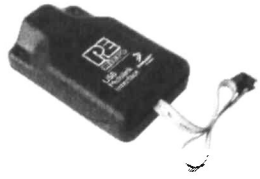


Figura D.1 PE Micro Multilink

Para que se pueda realizar la transferencia de información de manera adecuada, se debe asegurar que ambos leds del USB BDMMULTILINK estén encendidos. Como se muestra en la Figura D.2.

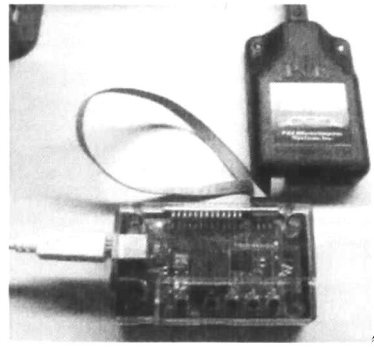


Figura D.2 USB BDMMULTILINK conectado al sensor y estos a su vez a la PC para cargar el programa.

Una vez asegurados que se encuentren bien conectados los dispositivos para compilar el proyecto se debe presionar el botón  El proyecto no debe generar errores, pero en caso de haberlos es necesario corregirlos verificando la sintaxis del programa.

Para iniciar la depuración es necesario ir al menú **Project_Debug,** o bien de forma rápida mediante el botón  . (Figura D.3).

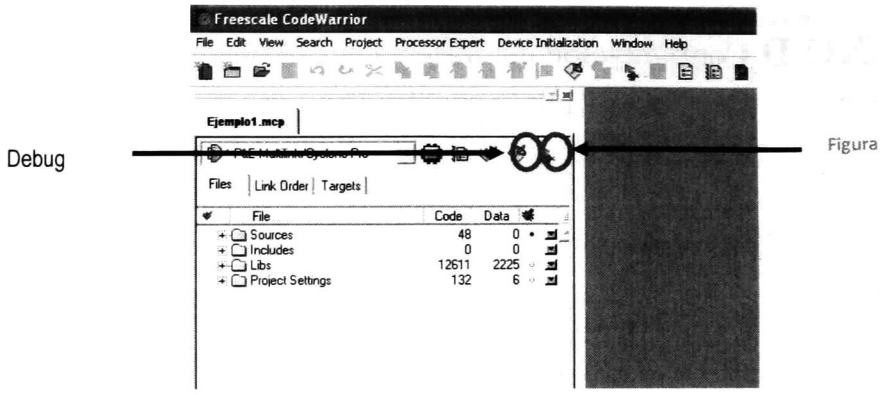


Figura D.3 Ventana Principal de CodeWarrior

Una vez emitido el debug, abrirá una nueva ventana mostrando que se establecerá la comunicación entre el multilink y los sensores, oprimimos el botón **Connect (Reset)** Paso 2, que se muestra en la figura D.4.

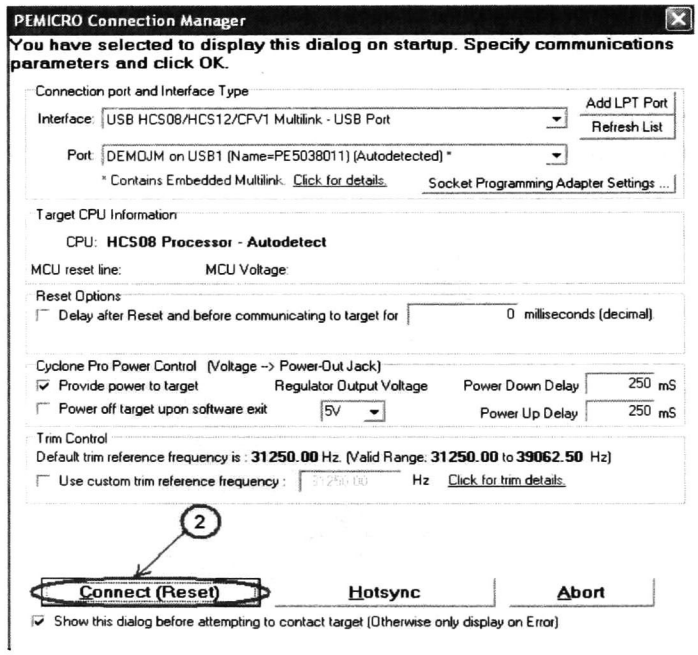


Figura D.4 Conexión con el DEMOJM

Se muestra otra ventana como ingreso al depurador. Esta ventana indica que se va a borrar la FLASH del microcontrolador y regrabarla, elegimos la opción Yes. Paso 3 mostrado en la Figura D.5

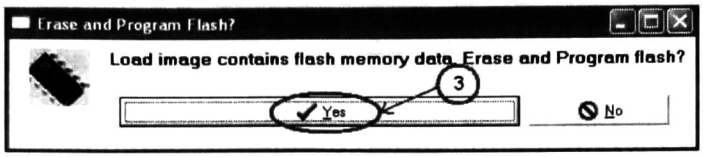


Figura D.5 Afirmación para re-escribir la memoria FLASH

Aparecerá una tercera ventana nos dice indica que ya se termino de cargar el programa y entonces podemos comenzar a configurar la terminal para de los sensores, que se encuentran ya cargados con la nueva información. (Figura D.6).

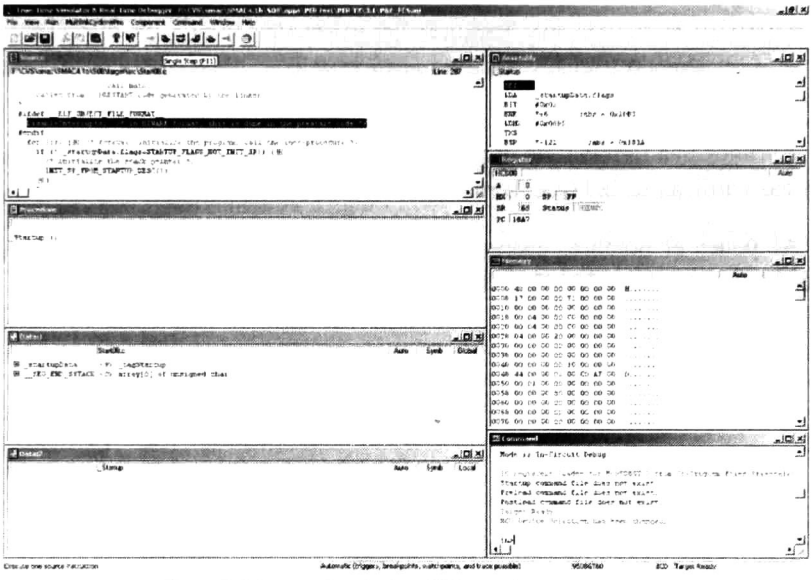


Figura D.6Ventana Principal de HiWaveDebugger

Configuración de los sensores en la terminal de dispositivos.

Primeramente tenemos que observar que número de puerto COM es el sensor que vamos a utilizar, para ello, abrimos inicio/Clic derecho en equipo/Propiedades/Administrador de dispositivos/ Puertos COM y ahí observamos el número de dispositivo que esta asignado al sensor, si se utilizan diversos sensores, se debe de tener cuidado en la numeración, debido a que se utiliza para las terminales. (Figura D.7).

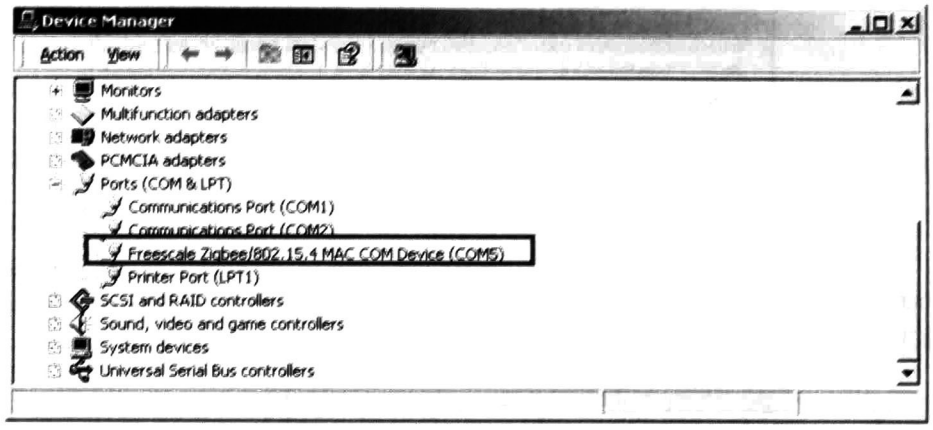


Figura D.7 Determinación del puerto COM en la administración de dispositivos.

Abrimos una terminal de dispositivos en el Sistema Operativo que estemos utilizando, para nuestro caso utilizamos la HyperTerminal de Windows.

Al poner el nombre, aparecerán las propiedades del dispositivo, cambiamos la tasa de velocidad por segundos a 38400, y el Control de Flujo a ninguno. (Figura D.8)

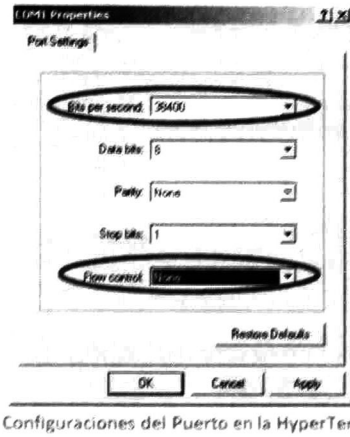


Figura D.8 Configuraciones del Puerto en la HyperTerminal.

En la pestaña de Archivo, Propiedades/Configuración/ Configuración ASCII, seleccionamos el checkbox que tiene la opción "*Append line feeds to incoming line ends*", que se muestra en la figuraD.9:

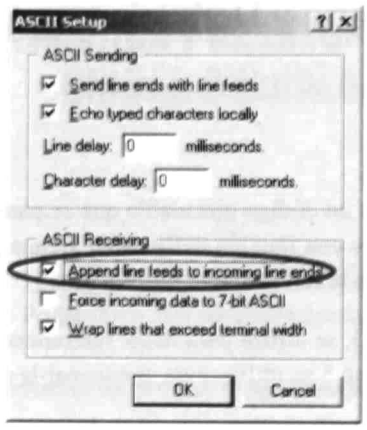


Figura D.9 Configuración de la recepción de la información a la HyperTerminal

Así una vez terminadas con las configuraciones anteriores podemos comenzar a ver que cambios se le realizaron en la programación.

ANEXO E Glosario y Definiciones

Definiciones

Una función *CallBack* es un código ejecutable que se pasa su resultado como argumento a otro código, permite que se llame a una función definida en alguna parte del código de manera autónoma y en estas funciones permiten la reutilización de código

May visto en el capítulo 3, se utiliza para hacer referencia al mayor score obtenido para realizar la elección de líder. Y en el cap 5 se utiliza para almacenar la mayor temperatura obtenida en la red.

Tiempo de ejecución es el tiempo que duran los algoritmos funcionando en los sensores, y en el que se muestran resultados.

Glosario de Terminos

- **ACK** =Acknowledgment Frame (Trama de reconocimiento)
- **Ant** = Hormigas
- **CH** = Cluster Head (Cabecera de clúster)
- **CSMA-CA** = Carrier Sense Multiple Access with Collision Avoidance (Acceso múltiple por detección de portadora con evasión de colisiones)
- **LAN** = Local Area Network (Redes de Área Local)
- **Link**= Enlace
- **LQ** = Link Quality (Calidad del enlace)
- **MAN**=Metropolitan Area Network (Redes de Área Metropolitana)
- **PC**= Personal Computer (Computadora Personal)
- **Sink**= pozo (Estación base)
- **SO** = Self-Organization
- **Weight** = Peso
- **WSN** = Wireless Sensor Network (Redes de Sensores Inalámbricas)
- **WAN**= Wide Area Network (Red de Área Amplia)

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Diseño e Implementación de una Red Inalámbrica de Sensores

del (la) C.

Miriam Alejandra CARLOS MANCILLA

el día 13 de Diciembre de 2012.

Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3C
CINVESTAV Unidad Guadalajara

Dr. Félix Francisco Ramos Corchado
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Mario Angel Siller González
Pico
Investigador CINVESTAV 2C
CINVESTAV Unidad Guadalajara