ALGORITMO DE LOCALIZACION DE NODOS EN UNA RED DE SENSORES INALAMBRICOS

FENER IVAN ORDOÑEZ

DANIEL FELIPE VELASQUEZ CASTRO

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERIA ELECTRONICA Y TELECOMUNICACIONES

DEPARTAMENTO DE TELEMATICA

POPAYAN, JULIO DE 2009

ALGORITMO DE LOCALIZACION DE NODOS EN UNA RED DE SENSORES INALAMBRICOS

FENER IVAN ORDOÑEZ

DANIEL FELIPE VELASQUEZ CASTRO

Trabajo de grado presentado como requisito para optar al título de Ingeniero en Electrónica y Telecomunicaciones

Director
HECTOR FABIO JARAMILLO ORDOÑEZ
Magister en Ingeniería Telemática

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERIA ELECTRONICA Y TELECOMUNICACIONES

DEPARTAMENTO DE TELEMATICA

POPAYAN, JULIO DE 2009

Tabla de contenido

Lista de Figuras

Tablas

INTRODUCCIÓN

La evolución de las redes de telecomunicaciones ha seguido el paso de las necesidades de la sociedad, para facilitar el desarrollo de sus culturas, economías y demás aspectos socio-culturales indispensables en el mundo de hoy. Las nuevas aplicaciones serán sistemas autónomos que permitirán interactuar con el entorno de una manera directa y sin intervención constante de una persona, capaces tanto de monitorear como de actuar en el medio en que se encuentren. Esta es la filosofía de las redes de sensores inalámbricos o WSN (*Wireless Sensor Network*), un concepto que surgió en la Universidad de Berkeley [1]; miles de dispositivos autónomos interconectados inalámbricamente, capacitados para operar en un medio. Sus aplicaciones en el campo militar, agrícola, cuidado de especies, control de calidad, prevención de desastres naturales entre otros, son motivo de su desarrollo y apoyo por parte de la IEEE, fabricantes y grupos de investigación.

La diversidad de aplicaciones refleja la importancia de WSN, pero también muestra el momento de desarrollar nuevos componentes que complementen y mejoren los sistemas actuales. Una necesidad percibida en muchas aplicaciones es la ubicación geográfica de los nodos de una red de sensores inalámbricos. Las soluciones deben ser acordes a las limitaciones que presentan estas redes, bajo costo, bajo consumo de energía, tamaño reducido de los dispositivos, redes con miles de nodos, carencia de antenas inteligentes y movilidad de los nodos en la mayoría de las aplicaciones. Una posibilidad es utilizar GPS (Global Positioning System) para obtener la posición de cada nodo, pero esto implica un incremento substancial tanto en el costo de cada nodo como en el consumo de energía y por otro lado su limitación en entornos cerrados *(indoor)*, por estas razones el uso de GPS no es viable. La telefonía celular hace uso de la triangulación para determinar la posición, en una WSN es factible usar este mecanismo pero implica configurar miembros de manera estática y su área de cobertura es limitada. Este panorama refleja la necesidad de tener un algoritmo capaz de mostrar la ubicación de cada nodo utilizando solo mensajes entre sus miembros, este sistema le permite movilidad a cada uno de sus nodos y sinergia con otras funciones.

Este documento presenta un algoritmo con el cual se pretende brindar una solución óptima a la necesidad de localización de nodos en una red de sensores, dicho algoritmo está basado en la intensidad de señal recibida (RSSI; *Received Signal Strength Indication*) de las señales emitidas por los nodos para determinar la distancia entre ellos y mediante un sistema de estimación de posición (Trilateración) crear un marco de referencia geográfico de toda la red, permitiendo movilidad a cada uno de sus nodos y reduciendo costos de red.

Este documento está organizado de la siguiente manera:

- Capítulo 1. Se presentan los fundamentos teóricos y las herramientas necesarias para el desarrollo de este proyecto. Se muestra una visión general de las redes de sensores, se explican sus características básicas centrándose en los protocolos MAC, de enrutamiento y de comunicaciones (Zigbee), luego se hace una introducción a las herramientas software

necesarias, empezando por el sistema operativo TinyOS y el lenguaje NesC en el que se desarrollaron las aplicaciones para los nodos, finalmente se hace una introducción a los sistemas de tiempo real.

- Capítulo 2. Se exponen los sistemas y mecanismos de localización y la forma como opera el algoritmo. Allí se describen las características de los mecanismos de localización, haciendo hincapié en las técnicas de cálculo de posición principales. Después se presentan los diagramas de casos de uso, componentes, diagramas de secuencia y diagrama de flujo del algoritmo implementado.
- Capítulo 3. Se presentan los resultados de las pruebas realizadas en diferentes escenarios y bajo configuraciones diferentes del sistema completo de localización de nodos.
- Capítulo 4. Se muestran las conclusiones respecto a los objetivos alcanzados en el proyecto y se proponen nuevos proyectos a desarrollar sobre el tema.
- Anexo A. Implementación de cada uno de los componentes del sistema de localización de nodos en una Red de Sensores Inalámbricos.
- Anexo B. Artículo Algoritmo Colaborativo para localización de nodos en una Red de Sensores Inalámbricos. Desarrollado sobre los resultados obtenidos del proyecto para la revista Avances en Sistemas e Informática de la Universidad Nacional de Colombia sede Medellín, ISN 1657-7663 .

CAPITULO 1

RED DE SENSORES Y HERRAMIENTAS SOFTWARE

Este capítulo presenta la base conceptual y las herramientas software con las cuales se desarrollo el algoritmo de localización de nodos en una red de sensores inalámbricos. Se hace una descripción general de las redes de sensores inalámbricos, su protocolo de acceso al medio, de enrutamiento y Zigbee como protocolo de comunicación, posteriormente se hace una descripción de las herramientas software utilizadas.

1.1 Red de Sensores Inalámbricos (Wireless Sensor Network, WSN)

Las Redes de Sensores Inalámbricos están formadas por pequeños dispositivos nodales totalmente autosuficientes, que equipados con sensores son capaces de recoger todo tipo de datos, como temperatura, humedad, movimiento, entre otros y capaces de transmitirlos inalámbricamente.

En el desarrollo del algoritmo de Localización se utilizó una *WSN* (véase la Figura 1.1), con tres elementos fundamentales, los nodos, la Gateway y el computador (*Personal Computer*, *PC*). Las características de estos dispositivos son:

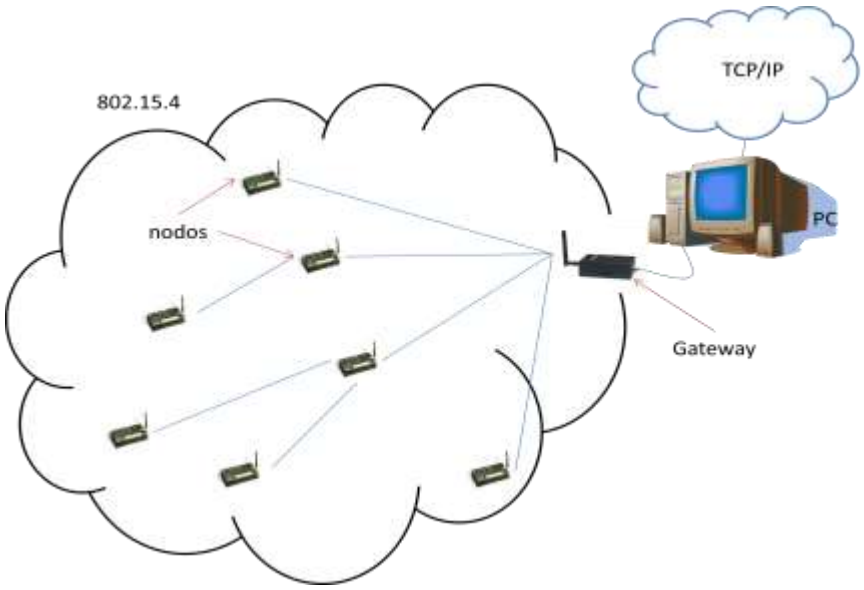

Figura 1.1. Red de sensores inalámbricos.

- *Nodos*: Toman los datos del medio utilizando sus sensores y envían la información a la Gateway para su posterior análisis. Cada nodo contiene un procesador que les permite realizar operaciones locales necesarias para el procesamiento de los datos adquiridos y para la comunicación con la red, un módulo de radio, una fuente de energía y módulos adicionales como sensores o actuadores dependiendo de la funcionalidad de la red. Los

aspectos más relevantes de cada nodo son su pequeño tamaño, reducido costo, bajo consumo energético, limitada capacidad de procesamiento y estrecho ancho de banda.

- *Gateway*: Elemento necesario para la interconexión entre la red de sensores y un PC (*estación base*). Su función principal es la de recolectar los datos enviados por los nodos y transmitirlos hacia la estación base. Contiene las mismas características de los nodos pero adicionalmente cuenta con una interfaz USB que le permite comunicarse con el computador y obtener la energía para su funcionamiento.
- *PC o Estación base*: Su función es la de procesar los datos obtenidos por la WSN. Sus altas capacidades permiten la operación del algoritmo de localización de los nodos, como también ofrece las herramientas para el almacenamiento de los datos y la interfaz para visualizar las posiciones.

La red inalámbrica utiliza el protocolo de comunicaciones *Zigbee* que está basado en el estándar *IEEE* 802.15.4, fue diseñado por la *Zigbee Alliance* para comunicaciones de corto alcance y es un modelo para redes inalámbricas de área personal (*WPANs*). Su origen radica en la necesidad de satisfacer al mercado de un protocolo de red inalámbrica flexible, el cual ofrezca un bajo consumo de energía, fiabilidad, interoperabilidad y seguridad para el control y monitorización de aplicaciones con tasas de datos bajas o moderadas [1]. La decisión de tomar este protocolo para el desarrollo del proyecto, se realizó luego de analizar la información disponible en internet por fabricantes y grupos de investigación sobre WSN, con lo cual Zigbee y las otras opciones son descritas más adelante.

Una característica adicional de una WSN es garantizar movilidad y autoconfiguración para cada uno de sus nodos, no se deben tener limitantes para nuevos nodos o si algún nodo falla el sistema debe reacondicionarse y continuar operando sin necesidad de intervención externa [2].

Ya que los nodos que conforman una WSN tienen altas restricciones en sus componentes *hardware*, los protocolos para estas redes deben garantizar un uso eficiente de los recursos, a continuación se hace una descripción de los principales protocolos y opciones utilizados por una WSN.

1.1.1. Protocolos de acceso al medio (*Medium Access Control,* MAC)

La función de un protocolo *MAC* es servir de base para protocolos de más alto nivel. En las redes de sensores, en la capa superior tendríamos el protocolo de enrutamiento, que usará las funciones implementadas en la MAC para enviar y recibir paquetes, sincronizar sus operaciones, etc.

La función principal del protocolo *MAC* es "escuchar" el medio y si está libre, transmitir. Este concepto es un punto crítico para garantizar un óptimo funcionamiento de una WSN, debido a que los nodos deben consumir la menor cantidad de energía para permitir el funcionamiento de las aplicaciones por tiempos prolongados de hasta meses sin necesidad de sustituir su fuente de alimentación.

Tomando en cuenta las restricciones de los nodos, los objetivos principales que se exigen a un protocolo *MAC* son:

- Evasión de colisiones efectiva.
- Reducir el retardo.
- Asegurar fiabilidad.
- Utilización del canal eficiente a bajas y altas tasas de datos.
- Tolerante a los cambios de frecuencia y condiciones de la red.
- Asegurar la posibilidad de establecer enlaces de comunicación y conseguirlo con el menor gasto de consumo energético posible.

Teniendo en cuenta estos objetivos se encontró una serie de protocolos que se podrían agrupar básicamente en dos clases, los protocolos basados en *slots* o ranuras de tiempo y los basados en muestreo.

***Protocolos de ranuras**:* Los nodos dividen el tiempo en intervalos discretos (ranuras de tiempo o slots), y los planifican dependiendo de si la radio está en modo recepción, modo transmisión o apagada. La sincronización de los nodos permite que estos enciendan la radio solamente cuando sea necesario, con lo que se reduce considerablemente la escucha ociosa. Estos protocolos a menudo son estáticos; después de que se planifique un determinado horario o sincronización para los slots, un nodo solo puede comunicarse con otros nodos dentro del mismo periodo de *slot*s previamente establecido. Entre los principales protocolos de ranuras se encuentran el IEEE 802.15.4, SMAC y TMAC [1], los dos últimos se describen a continuación y el protocolo IEEE 802.15.4 se detalla posteriormente:

- SMAC (*Sensor MAC*): Usa multiplexación por división de tiempo para otorgarle a cada nodo un slot para su transmisión y recepción, además tiene dos periodos, uno en el que todos los nodos están activos y se hacen las comunicaciones en su respectivo slot y otro periodo en el cual todos los nodos pasan a estado inactivo, de tal manera que la eficiencia de energía se ve reflejada en la relación del periodo activo con respecto al inactivo. Este protocolo maneja nodos sincronizadores, la elección de este nodo se realiza cuando todas las nodos están inactivas y aleatoriamente escogen un tiempo, el primer nodo en finalizar dicho periodo transmite un paquete SYNC el cual contiene el tiempo de inactividad, los nodos vecinos permanecen ese tiempo en estado *sleep* y luego despiertan y escuchan el canal, si alguno tiene información para enviar lo debe hacer en su slot, y si los datos no fueron transmitidos por el tiempo reducido para enviar entonces debe esperar al próximo periodo de actividad. Debido al limitado radio de operación de cada nodo se tienen varias celdas, cada una con su respectivo sincronizador, y en los bordes comunes entre celdas se hallan los nodos que operan como puentes entre celdas para hacer enrutamiento, ellos deben trabajar en los periodos activos de ambas celdas para poder enrutar paquetes que vayan de un lado para otro, obviamente este esfuerzo sacrifica la energía de dicho nodo y es un punto sensible de la red [1].

- TMAC (*Time MAC*): Hace una fusión de conceptos entre los protocolos SMAC y *RTS/CTS* el cual plantea el intercambio de pequeños mensajes entre nodos para informar del destino de la información y de la cantidad de bytes del mensaje, de tal manera que en el slot

correspondiente un nodo puede entablar una comunicación con un vecino y el resto puede pasar a estado inactivo, esto minimiza el gasto de energía pero acarrea inconvenientes provenientes de la esencia del protocolo RTS/CTS, los cuales son mitigados usando un mensaje FRTS (*Future-Ready-to-Send*) que informa a un nodo que luego de cierto periodo el recibirá información, esto permite que ese nodo no pase al estado *sleep* por mucho tiempo y se generen sobrecargas de información para enviar [1].

***Protocolos basados en muestreo**:* En estos protocolos en lugar de coordinar las ranuras de tiempo, los nodos despiertan periódicamente en busca de actividad en el canal de radio, si la detectan, comienzan a recibir los datos. Dentro de los protocolos basados en muestreo están: Aloha con preámbulo y WiseMAC, sus características principales son:

- Aloha con preámbulo: Antes de enviar los datos se transmite una cadena de bits que sólo pretende dar a conocer a la red el interés de un nodo de enviar información y el receptor devuelve un ACK para informar de su conexión exitosa. Este mecanismo informa a los otros nodos de la red que permanezcan en estado inactivo o *sleep*, lo cual permite que se utilice alrededor de 500 veces menos energía con respecto al estado activo, pero incrementa el tiempo de transmisión lo que afecta la energía del transmisor y el uso del canal [1].
- WiseMAC: Hace una mejora substancial al protocolo Aloha con preámbulo sincronizando todos los nodos para escuchar el medio en ciertos instantes, lo que permite a un transmisor reducir el tamaño del preámbulo, pero debido a que no se tiene una fuente global para la sincronización se opta por tener una tabla en cada nodo con los tiempos de escucha de cada vecino, estos tiempos son comunicados por cada nodo cuando se envía un ACK donde se informa el tiempo que permanecerá en estado *sleep* [1].

En contraste con los protocolos de ranuras, los protocolos de muestreo son muy flexibles, pero esta flexibilidad tiene un costo, mientras que los protocolos de ranuras envían los paquetes de datos regularmente, en muestreo hay que enviar mensajes largos y costosos energéticamente para “despertar” a un vecino. Los protocolos de ranuras ofrecen la mejor opción para una WSN por su bajo consumo de energía, es por tal hecho que nuestra implementación hace uso de este tipo de protocolo.

Los protocolos anteriormente descritos fueron desarrollados para redes inalámbricas, pero hacen uso intensivo de energía si se tiene como parámetro una *WSN*, donde es vital mantener el consumo energético bajo, tener alta densidad de nodos, permitir movilidad y ofrecer diferentes roles dentro de la red. Estudiando estas características en los protocolos, se observó que el protocolo que mejor se adecua a nuestro proyecto es el IEEE 802.15.4 por lo que se hace una descripción detallada de este a continuación.

Protocolo MAC IEEE 802.15.4

Este protocolo define la capa física y MAC para redes inalámbricas con bajas tasas de transmisión, también es la base para otros estándares como ZigBee, que construye los niveles superiores para brindar una solución a las necesidades de las WSNs.

Unas de las características más importantes en este protocolo es su flexibilidad de red y bajo consumo de energía frente a otros protocolos, como las redes de sensores basadas en IEEE 802.11.

Las topologías de red establecidas por este protocolo son la topología tipo estrella y la topología *"Peer-to-Peer"*. La topología que se maneje depende de la aplicación en la que se va a utilizar la red de sensores. Es necesario contar con roles para coordinar la red, en este caso se tienen tres tipos de dispositivos (nodos), dispositivo de funcionalidad completa (FFD), dispositivo de funcionalidad reducida (RFD), y el controlador de red (Figura 1.2 [3]). Dependiendo de la aplicación se puede organizar la red con las siguientes topologías [3]:

- ***Topología "peer to peer":*** Todos sus nodos pueden ser FFD, ya que todos tienen la misma prioridad de acceso al medio, y aunque existe un coordinador *PAN*, éste no tiene la función de señalización, por lo que se produce una lucha por el acceso al medio.
- ***Topología en estrella:*** Hay un nodo único FFD que recibe el nombre de coordinador *PAN* (*Personal Area Network*), su función es regular el acceso al medio mediante un mecanismo de señalización.
- ***Topología en malla:*** Se requiere de un coordinador PAN para regular las comunicaciones y de varios FFD capaces de enrutar los mensajes provenientes de dispositivos de funcionalidad reducida (RFD) hacia su punto de destino.
- ***Topología en árbol***: Se utilizan dispositivos FFD para que sirvan como puentes entre los RFD cercanos y un punto distante, donde se requieren de varios saltos entre FFD para llegar hasta él.

En la implementación del proyecto se utilizó la topología en malla debido a la naturaleza de la red implementada, donde todos los nodos tienen la posibilidad de moverse libremente, pueden entrar y salir de la red sin condicionamientos, estas características implican que cada uno de ellos tenga la capacidad de ser FFD para que puedan adoptar funcionalidad completa en cualquier momento.

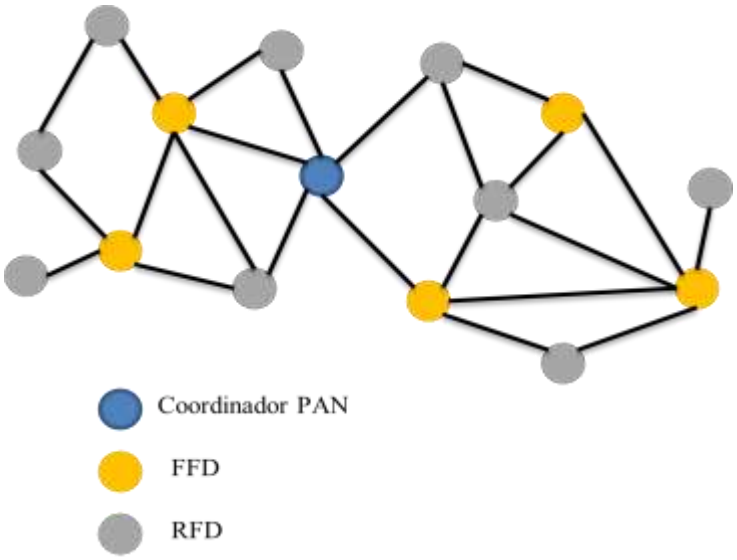

Figura 1.2*. Topología en malla del estándar IEEE 802.15.4.*

Modos de funcionamiento.

El protocolo 802.15.4 permite dos modos de funcionamiento: con *beacons* y sin *beacons*.

***Con beacons**:* Consiste en que un determinado nodo, mandará unas señales, *balizas*, de manera periódica marcando una serie de divisiones de tiempo (*slots*) entre dos *beacons* (16 *slots* en este caso). Los nodos que quieran transmitir lo hacen en uno de estos *slots*. La señal, indica a un nodo que puede transmitir, si no tiene nada para trasmitir se apaga. En el caso de que el nodo quiera transmitir, espera un periodo de contención aleatorio de varios slots denominados backoff. Tras ese periodo, si el canal está libre, el nodo transmitirá alineado con el siguiente slot. Como mecanismo de acceso al medio se utiliza *CSMA-CA* ranurado en donde las ranuras de *backoff* están alineadas con el comienzo de un *beacon*. Cada vez que un dispositivo desea transmitir, primero tiene que alinearse con el siguiente *slot* de *backoff* y entonces tiene que esperar un número aleatorio de ranuras de *backoff*. Si el canal está libre, en el siguiente *slot* comenzaría a transmitir. Si el canal está ocupado, dejara pasar otro número aleatorio de ranuras de *backoff*. Los únicos paquetes que no están sometidos a *CSMA-CA* son los *ACK's* y los *beacons* [4].

***Sin beacons**:* Aquí el mecanismo de acceso al medio es *CSMA-CA* no ranurado, esto implica que los dispositivos transmiten en el momento que es necesario. Cada vez que un dispositivo quiere transmitir, espera un tiempo aleatorio, si encuentra el canal libre espera un tiempo de *backoff*, pasado este tiempo intenta transmitir. Si el canal siguiera ocupado después del periodo de *backoff* volverá a esperar otro periodo aleatorio de tiempo y otro de *backoff* [4].

El sistema implementado no utiliza *beacons,* cada nodo transmite su información cuando la Gateway lo autoriza, pero para esto, no se realiza una división del tiempo en slots, el canal estará disponible solo para aquel nodo que deba transmitir su información, esta configuración permite administrar eficientemente el canal y evita interferencia entre mensajes de nodos vecinos.

Arquitectura del protocolo IEEE 802.15.4

Este protocolo define qué operaciones o responsabilidades tiene tanto la capa física como la capa MAC.

Capa física: Es la responsable del encendido y apagado del transceptor, detecta la energía en el canal, calcula la calidad del enlace, selecciona los canales, detección de canal limpio y transmitir o recibir los paquetes a través del medio físico. El estándar especifica como frecuencias base de transmisión 2.4GHz, 916MHz, y 868MHz, los dispositivos de Crossbow utilizados en este proyecto transmiten en la frecuencia de 2.4GHz. Esta capa también se encarga de detecta el enlace indicando su estado a través de los indicadores de calidad del enlace y de detección de energía. Por ejemplo, mediante el detector de energía se puede saber si un canal esta libre, el detector de calidad del enlace se puede utilizar para seleccionar el siguiente salto cuando se enrutan paquetes. Por esto, es importante que estas métricas estén accesibles para capas superiores de la arquitectura de red y así en función de estos parámetros optimizar su funcionamiento.

Capa MAC: Esta proporciona las primitivas necesarias para establecer el acceso al medio. También proporciona los comandos para garantizar la transferencia de datos directa entre dos nodos,

mediante el uso de *ACK´s* y primitivas para poder implementar a nivel de aplicación mecanismos de seguridad.

1.1.2. Protocolos de enrutamiento.

La responsabilidad de un protocolo de enrutamiento es garantizar el traslado de un mensaje de un punto A a un punto B, encontrando el camino más confiable para alcanzar el destino deseado teniendo en consideración la distancia, el requerimiento mínimo de energía, el tiempo de vida del enlace inalámbrico y reparación de los enlaces caídos gastando el mínimo de potencia de procesamiento y ancho de banda [5].

Los nodos de una WSN no tienen un conocimiento de la topología de la red, por lo cual deben descubrirla. Cuando un nuevo nodo aparece en la red, informa a sus vecinos de su presencia y se informa acerca de los nodos a su alcance y de la manera de enrutarse a través de ellos, también puede anunciar al resto de nodos que pueden ser accedidos desde él. Transcurrido un tiempo, cada nodo sabrá que nodos tiene alrededor y la forma de alcanzarlos.

Algunos de los principales requisitos que tienen que cumplir los protocolos de enrutamiento son:

- Elegir las mejores rutas para alcanzar los destinos.
- Tener una tabla de enrutamiento pequeña y dinámica.
- Realizar constantes actualizaciones de las tablas de enrutamiento (debido al posible movimiento, aparición o caída de algunos nodos).

Las formas de enrutar un mensaje en una WSN son [1]:

De un salto: La comunicación de los nodos y la central es directa. Lo cual implica que la red no puede extenderse porque de lo contrario los mensajes no alcanzarían la estación base y su cobertura está vinculada con la capacidad del transceptor de la central. Este tipo de protocolo limitaría el funcionamiento de nuestro algoritmo a solo aquellos nodos vecinos a la Gateway, por tanto fue desestimado.

Basada en clústeres: Los nodos se agrupan en clústeres con un nodo central, que es el responsable de enviar toda la información del clúster a los nodos centrales de otros clústeres o a la estación base. Aunque la información salta de un nodo a otro, se está haciendo de una forma tal que se cubren mayores distancias. Esto hace que la información se transfiera más rápido a la central. Este mecanismo no ofrece ventajas en redes donde sus nodos están en movimiento continuo porque su configuración cambiaría constantemente y eso demanda un mayor uso del canal para señalizaciones y por ende un incremento en el gasto de energía, por estas razones no es opción efectiva para nuestra aplicación [5].

Multi-salto: Con este sistema los nodos que se encuentran fuera del alcance de la estación central transmiten su información a través del reenvió en uno de sus vecinos que se encuentra más próximo a la estación central, este proceso se realiza hasta que el mensaje llegue a la estación central. Este sistema tiene desventajas por las tablas de enrutamiento que deben tener los nodos, pero permite el cambio rápido de roles de los nodos dentro de la red, de esta manera si un nodo cambia de posición o es apagado, los mensajes que pasaban por él hacia la estación base ahora lo

harán por el vecino que ofrezca las mejores condiciones, estas características son propias de la red implementada [5].

El sistema *Multi-salto* es utilizado por *Zigbee,* protocolo descrito en la sección 1.1.3 que se implantó en el proyecto y que es la base para protocolos como son *Dissemination* y *Collection*, ellos son módulos que hacen parte del sistema operativo *TinyOS*, del cual se mencionarán sus características en la sección 1.2.1. En el protocolo *Dissemination* se divulga el valor de una variable a todos los nodos de la red, este valor representa en el sistema un identificador de un nodo, aquel que debe enviar su información a la *Gateway*. El protocolo *Collection* establece un nodo como raíz o sumidero, de tal manera que todos los mensajes que se envíen hacia él llegaran de manera directa o por medio de saltos, con este mecanismo los nodos envían su información a la *Gateway*.

1.1.3. Zigbee

Zigbee es un protocolo de comunicaciones inalámbricas diseñado por la Zigbee Alliance para comunicaciones de corto alcance y bajo consumo energético como es el caso de las *WSN*s, está basado en el estándar IEEE 802.15.4. Opera en las bandas libres *ISM(Industrial, Scientific and Medical.*) en las frecuencias de 2.4 GHz (mundial), 868 MHz (Europa) y 915 MHz (EEUU). Con una transferencia de datos de 250 Kbps en la banda de 2.4GHz (16 canales con espaciamiento de 5 MHz), 40kbps en 915MHz (10 canales con espaciamiento de 2 MHz) y a 20kbps en la de 868MHz (un solo canal). Las distancia de transmisión pueden variar desde los 10 hasta 300 metros (esto depende de la potencia y el entorno donde se encuentren los dispositivos) [6].

Este protocolo de comunicación es Multi-salto como se mencionó anteriormente, por lo que existe comunicación entre dos nodos aun cuando estén fuera del rango de transmisión, mientras existan otros nodos intermedios que los interconecten, debido a esto se incrementa significativamente el área de cobertura de la red [7]. En una red con este protocolo se pueden observar tres tipos de dispositivos (Figura 1.3 [5]):

***Coordinador*:** debe ser un dispositivo FFD y solamente puede haber un coordinador por red. Este dispositivo inicia una red, y forma la raíz de ésta. Es capaz de almacenar información de su red, como las llaves de seguridad.

***Router*:** son dispositivos FFD, este puede estar asociado a otro *Router* o a un coordinador como también pueden actuar como un coordinador. Es el encargado del enrutamiento hacia otros dispositivos.

***Dispositivo final*:** dispositivos RFD, es un elemento básico de la red con funcionalidades necesarias para hablar con su nodo padre; pero no pueden servir como Router.

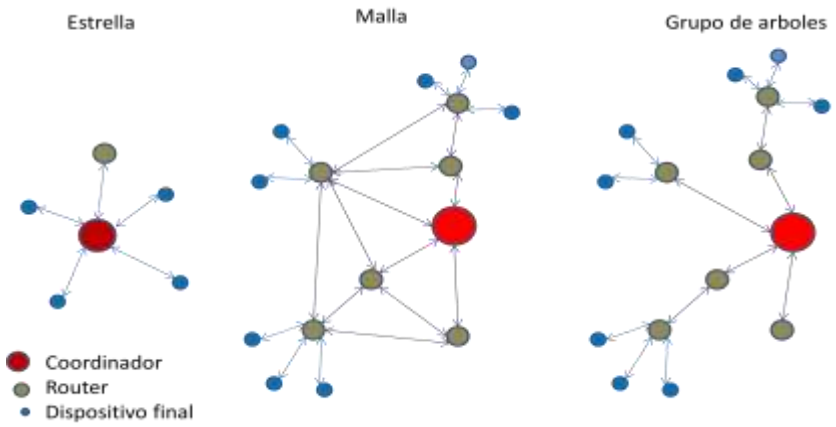

Figura 1.3 Tipos de dispositivos en una red Zigbee.

La red implantada cuenta con la Gateway como coordinador y los nodos de la red como posibles Routers o dispositivos finales dependiendo de la situación.

Arquitectura

Zigbee se asemeja al modelo de referencia OSI con una pila de capas donde cada una de ellas cumple una serie de funciones específicas independiente de las demás. Las dos capas inferiores, la capa física y la capa de acceso al medio (MAC), se basan en el estándar IEEE 802.15.4. La siguiente es la capa de red que junto con la capa de soporte de aplicación son el aporte de Zigbee (Figura 1.4 [6]).

Figura 1.4. Capas del protocolo Zigbee.

Capa de red: Sus principales características son:

- Sirve como interfaz entre la capa de aplicación y la capa MAC para lo cual brinda dos tipos de servicios, un servicio de datos y uno de control. El servicio de datos permite comunicarse con otros dispositivos de la misma red, el de control permite la comunicación entre la capa de aplicación y las demás capas.

- En esta capa se brindan los métodos necesarios para crear o unirse a una red, enrutar, filtrar, cifrar y autenticar paquetes.

- En esta capa se implementan las distintas topologías de red que Zigbee soporta.

Capa de soporte de aplicación: Mantiene la relación con los demás dispositivos con los que la aplicación interactúa y simplifica el envío de datos a estos dispositivos. También realiza un filtrado de paquetes a nivel de aplicación.

Capa de aplicación: En el nivel conceptual más alto se encuentra la capa de aplicación que no es otra cosa que la aplicación misma y de la que se encargan los fabricantes o desarrolladores. En esta se define el papel que juega el nodo en la red, si es un coordinador, Router o dispositivo final.

Empaquetamiento

En Zigbee existen cuatro tipos de paquetes: datos, ACK, comandos y beacons (Figura 1.5 [6]).

Paquete de datos: Usado para las trasmisiones de datos.

Paquete ACK: Usado para la confirmación exitosa de paquetes, la transmisión de este se hace inmediatamente después de recibir un paquete.

Paquete de Comandos: Utilizado para el control o configuración de la red por parte del coordinador centralizado.

Paquete de beacons: Este paquete "despierta" los dispositivos para que "escuchen" y luego vuelven a "dormirse". Es importante para mantener sincronizados los dispositivos sin tener que tenerlos todo el tiempo encendidos.

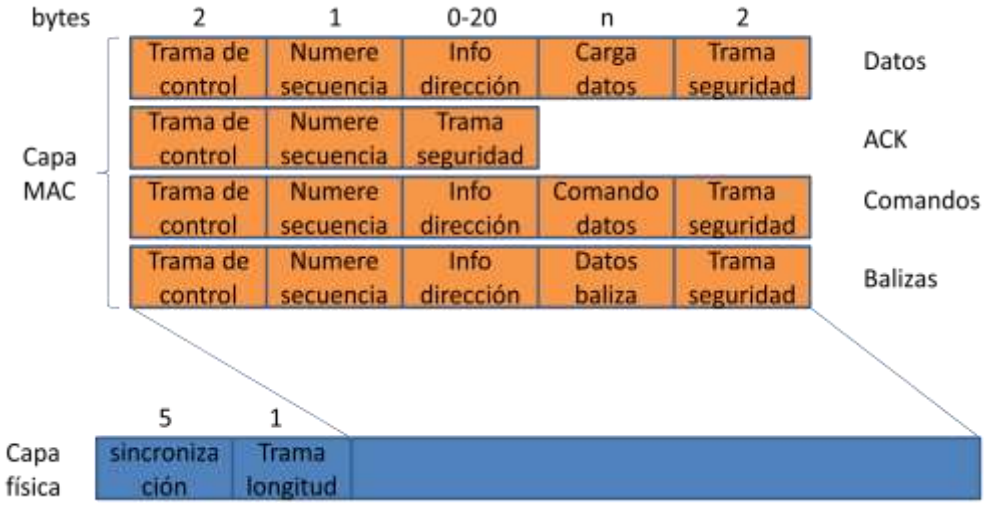

Figura 1.5. Estructura de Paquetes.

Descripción de los campos:

Trama de control: especifica como es el resto de la trama de datos y que es lo que contiene.
Numero de secuencia: verifica la integridad de la trama.

Información de direccionamiento: contiene el nodo de origen y destino dependiendo del tipo de trama, puede utilizar entre 0 y 20 bytes, esto permite tener paquetes lo más reducidos posible.
Carga de datos: De longitud variable, con un máximo de 104 bytes.
Trama de seguridad: FCS (Frame Check Sequency), es una trama de chequeo de 16 bits CRC (Cyclic Redundancy Chech).
Trama longitud: Indica la longitud de la trama de máximo 127bytes.

1.2. Herramientas software

El proyecto se desarrollo utilizando un *PC* con sistema operativo Ubuntu 8.04.1 con un Kernel 2.6.24-21-rt de tiempo real y un Kit Profesional de la empresa *Crossbow* que contiene una Gateway, seis nodos con sensores, un nodo de pruebas y una placa para programar.

El sistema operativo sobre los nodos es *TinyOS,* especialmente diseñado para las características intrínsecas de los nodos sensores y sus potenciales aplicaciones de red, y en él se programó las aplicaciones para los nodos sensores y la Gateway en el lenguaje de programación *NesC*. El sistema operativo *TinyOS* y el lenguaje *NesC* se describen en la sección 1.2.1.

La aplicación que contiene el algoritmo de localización de nodos se desarrollo para un sistema de tiempo real flexible sobre *RTJava*, las características de un sistema en tiempo real y del API de *RTSJ (Real Time Specification for Java)* se describen en la sección 1.2.3.

El entorno de desarrollo utilizado para la aplicación en el PC fue *Netbeans* 6.0.1, de la empresa *Sun Microsystem*, y para las aplicaciones de la Gateway y los nodos se utilizo *Eclipse* 3.4.1 de *Eclipse Fundation* con la ayuda del *API* para TinyOS.

Las coordenadas de los nodos son almacenadas en una base de datos utilizando el motor *MySQL* 5.0.51 de la empresa *Sun Microsystem*, el cual tiene la capacidad de trabajar en tiempo real flexible al igual que una aplicación sobre *RTSJ*.

1.2.1. TinyOS (Tiny MicroThreading Operating System)

TinyOS es un sistema operativo de código abierto desarrollado en la Universidad de California en Berkeley. Este S.O. está desarrollado sobre NesC, un lenguaje derivado de C, y está diseñado para responder a las necesidades de los sistemas embebidos de una red sensores inalámbricos, tales como tamaño físico reducido, operaciones de concurrencia intensiva, bajo consumo de energía, diversidad en diseños y usos, y finalmente operaciones robustas para facilitar el desarrollo confiable de aplicaciones. Además se encuentra optimizado en términos de uso de memoria y eficiencia de energía [8] [9].

El diseño del núcleo de TinyOS está basado en eventos y tareas.

- *Eventos*: Realizan pequeños procesos que son críticos en el tiempo, interrumpiendo a las tareas que estén siendo ejecutadas. Cuando llega un evento, se guarda el estado actual del sistema, y cuando se completa, se restablece el estado y se continúa con la tarea que estuviera en curso cuando llegó [10].

- *Tareas:* realizan procedimientos más extensos que no son críticos en tiempo por lo que pueden ser interrumpidas por los eventos. Estas se realizan en su totalidad sin interrumpir otras tareas o eventos [10].

Con este diseño se permite que los eventos puedan ser realizados inmediatamente, siendo capaces de interrumpir a las tareas y por tanto, alcanzar un alto rendimiento en aplicaciones de concurrencia intensiva. Además, este enfoque usa las capacidades del equipo de manera eficiente consumiendo el mínimo de energía [11].

Arquitectura de TinyOS.

Este S.O. tiene un modelo de programación basado en un conjunto de componentes e interacciones entre estos (el cual es un modelo de programación característico de los sistemas embebidos), lo que permite una fácil migración a otro hardware. Esta migración es muy importante en las redes de sensores, ya que constantemente aparecen nuevas tecnologías [8].

Cada componente consta de (Figura 1.6):

- *Manejador de comandos*: Los comandos son peticiones hechas a componentes de capas inferiores, generalmente son solicitados para ejecutar alguna operación [9].
- *Manejador de eventos*: Los manejadores de eventos son invocados por eventos de componentes de capas inferiores, o por interrupciones cuando se está directamente conectado al hardware [9].
- Un *frame* de tamaño de datos privado, con asignación estática de memoria [9].
- *Un bloque con tareas simples*: Las tareas son entregadas a un planificador que en este caso está implementado con método *FIFO* (donde las tareas se ejecutan secuencialmente) [9].

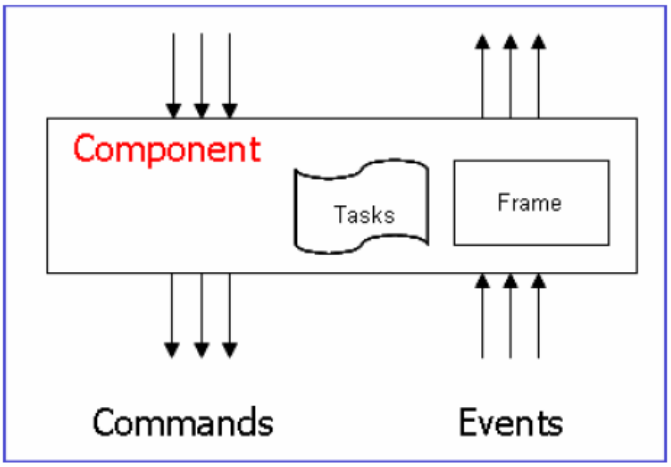

Figura 1.6 Modelo de un Componente en TinyOS.

Tipos de componentes

Los componentes se pueden clasificar en tres tipos que son:

- *Abstracciones de Hardware*: Mapean el hardware físico en el modelo de componentes [8].
- *Hardware Sintético*: simulan el comportamiento del hardware avanzado. Se podría decir que esta componente es una máquina de estado que podría ser directamente modelada en el hardware [8].
- *Componente de alto nivel*: Realizan el control, enrutamientos y toda la transferencia de datos. También realizan cálculos sobre los datos o su agregación [8].

Ya que TinyOS se encuentra programado en el lenguaje NesC, TinyOS y NesC se encuentran muy relacionados, por lo que a continuación se presenta un resumen de las principales características de este lenguaje.

1.2.2. NesC

Como se comentó anteriormente, NesC está basado en el lenguaje de programación C, diseñado para plasmar los conceptos de estructuración y los modelos de ejecución de TinyOS. Los conceptos básicos sobre NesC son:

- Separación de la construcción y la composición: los programas se forman mediante componentes interconectados para formar los programas completos. Los componentes están definidos en dos ámbitos, uno por su especificación (que contiene el nombre de las instancias de las interfaces) y el otro por su implementación. Los componentes tienen concurrencia interna en forma de tareas. Los hilos de control pueden pasar a un componente a través de las interfaces. Estos hilos pueden ser de ejecución de tareas o interrupciones hardware [12].

- Las especificaciones del comportamiento de un componente se hacen por medio de interfaces. Las interfaces pueden ser proporcionadas o usadas por los componentes. Las interfaces proporcionadas están hechas para representar la funcionalidad que los componentes proporcionan a su usuario, las interfaces usadas representan la funcionalidad que el componente necesita para realizar su trabajo [12] [13].

- Las interfaces son bidireccionales: especifican una colección de funciones que se pondrán en ejecución por los proveedores de la interfaz (comandos) y un conjunto para ser implementados por el usuario de la interfaz (eventos). Esto permite a una interfaz simple representar una interacción compleja entre componentes (por ejemplo, registrar algo de interés de un evento, seguido por una llamada cuando el evento ocurre) [12].

- Los componentes se unen estáticamente mediante sus interfaces. Esto incrementa la eficiencia en tiempo de ejecución, apoya un diseño robusto y además permite un mejor análisis estático de los programas [13].

- NesC está diseñado bajo la expectación de que el código será generado por compiladores de programas completos. Esto permite una mejor generación de código y análisis [13].

- El modelo de concurrencia de NesC se basa en tareas que se ejecutan hasta terminar y manejadores de interrupción, los cuales pueden interrumpir las tares y otros manejadores [13].

Interfaces

Especifican un canal de funcionamiento múltiple entre la interacción de dos componentes, el que provee y el que usa. Una interfaz declara un conjunto de funciones (métodos) que la proveedora debe implementar y otro conjunto de funciones (eventos) que quien la usa debe implementar. Para llamar a los métodos de una interfaz, se deben implementar los eventos de esa interfaz. Un componente simple puede usar o proporcionar múltiples interfaces y múltiples instancias de la misma interfaz [13].

Componentes

Una aplicación NesC consiste en un conjunto de componentes que formar un ejecutable. Un componente proporciona y usa interfaces. Hay dos tipos de componentes en NesC: los módulos y las configuraciones.

Los módulos proporcionan el código de la aplicación. Un módulo debe implementar cada comando de las interfaces que proporciona y cada evento de las interfaces que usa.

Las configuraciones se usan para ensamblar los componentes unos con otros, conectando las interfaces que usan unos componentes a las interfaces que les proporcionan otros.

Cada aplicación NesC se describe por una configuración de alto nivel que conecta todos los componentes que contiene [12].

1.2.3. Sistemas de tiempo real.

En esta sección revisaremos brevemente los elementos básicos acerca de los sistemas de tiempo real que son necesarios para el desarrollo de nuestro proyecto.

Aplicaciones en tiempo real

Los sistemas de tiempo real (*Real Time System*, *RTS*) no solo están regidos por la velocidad, sino que también a la *predectibilidad,* el saber que un sistema va a ejecutarse en un marco de tiempo requerido, no necesariamente muy pequeño aunque generalmente lo es. Las aplicaciones en tiempo real son particionadas en tareas de las cuales algunas tienen restricciones de tiempo, el objetivo es organizar estas para cumplir las metas de tiempo y mantener el correcto resultado computacional [14].

Lo que determina si una aplicación es en tiempo real, es si sus requerimientos incluyen restricciones de tiempo. Los RTS se pueden clasificar en sistemas de tiempo real flexible y duro o crítico dependiendo de los requerimientos del sistema.

Sistema de tiempo real duro o crítico: es aquel en el cual el sistema tiene que cumplir todas sus metas de tiempo sin excepción. Usualmente estos sistemas también tienen baja latencia, el tiempo entre que ocurre un evento disparador y se inicia o completa la respuesta a este evento generalmente es medido en microsegundos o milisegundos [15].

Muchos sistemas de tiempo real duro o crítico se los clasifica como *sistemas de seguridad crítica*. Estos sistemas se usan para proteger a humanos de resultar heridos o quedar en peligro.

Sistema de tiempo real flexible: es aquel que funciona correctamente de acuerdo a su especificación, aunque el sistema ocasionalmente no cumpla con las metas de tiempo. Los sistemas de tiempo real flexible especifican el porcentaje de metas que pueden incumplirse, y qué tan frecuente esto es aceptable [15].

Parámetros de tiempo y tareas

Una característica esencial de los *RTS* es su interacción con el entorno, lo que los hace reactivos ante estímulos externos realizando una determinada acción o actividad compuesta de una o varias tareas.

Una tarea es una entidad de ejecución independiente, que comienza con la lectura de los datos de entrada y termina con la producción de los resultados. Las tareas pueden ser:

• *Periódicas*: Tienen un intervalo de tiempo constante entre activaciones sucesivas.

• *Aperiódicas*: son tareas donde no se conocen los instantes de activación ni el intervalo de tiempo mínimo entre activaciones.

• *Esporádicas*: los tiempos de activación no son conocidos pero se conoce que un intervalo de tiempo mínimo existe entre activaciones sucesivas.

Como se ha mencionado, la ejecución de un sistema de tiempo real está conformada por un conjunto de tareas τ = {*T1, T2, T3, ... ,Tn*}, En la Figura 1.7 [15], se muestran los parámetros funcionales de una tarea *Ti* = (*Ci, Di, Pi, Si*), con *i* = 1,…., *n*:

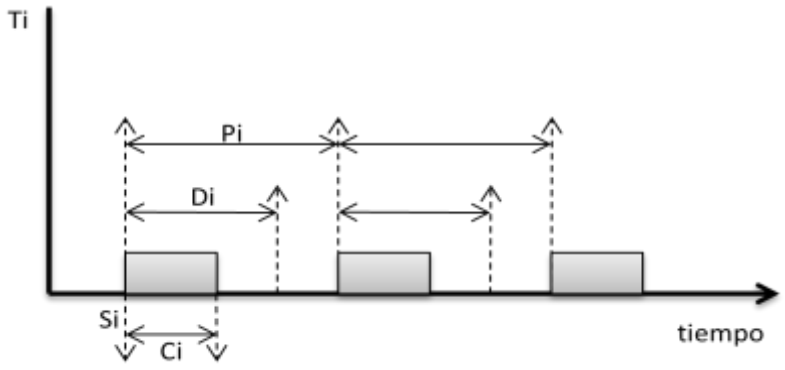

Figura 1.7 Parámetros de configuración de una tarea.

Donde:

Tiempo de cómputo (*Ci*): es el tiempo de procesador necesario para completar la ejecución de una tarea. Depende de la velocidad del procesador.

Plazo de respuesta o deadline (*Di*): es el intervalo de tiempo máximo que puede transcurrir entre el instante a partir del cual una tarea debe activarse y su finalización.

Periodo (*Pi*): una tarea es ejecutada de forma regular cada intervalo de tiempo.

Tiempo de inicio (*Si*): corresponde al instante de tiempo de la primera activación.

Antes de implementar un *RTS*, se deben realizar mediciones de tiempos para asegurar que $Ci \leq Di$ [16].

El algoritmo de localización es una tarea periódica que se ejecuta cada 500 milisegundos, tiempo suficiente para recibir la información de los nodos, hacer el cálculo de las coordenadas y almacenar la información en la base de datos, este tiempo está estimado para un número máximo de 10 nodos en la red, donde por cada ciclo todos los nodos envían su información a la estación central.

Planificación

Una de las preocupaciones de programación en tiempo real es el de asegurar la ejecución oportuna o previsible de las secuencias de instrucciones de máquina.

Un *RTS* responde de manera predecible a estímulos impredecibles y que, incluso, pueden producirse "al mismo tiempo". Por tanto, el sistema debe resolver distintos problemas a la vez tomando en cuenta que las tareas requieren el acceso a un recurso compartido (procesador) [16].

La *planificación* consiste en decidir el orden de ejecución de las tareas en el procesador de acuerdo a un algoritmo o políticas de ejecución, dando como resultado una secuencia de ejecución. Algunas de las políticas o algoritmos de planificación más utilizada son:

• *Fixed Priority Scheduling*: Atención a las tareas basada en una prioridad numérica fija.

• *Rate Monotonic*: Asignación de prioridades dinámicas según la frecuencia de las tareas.

• *Earliest Deadline First*: Basado en el plazo de respuesta. Las tareas con plazos más cortos tienen mayor preferencia de ejecución.

• *Least Laxity*: Utiliza la laxitud como criterio de decisión. La laxitud es la diferencia entre el plazo de respuesta y el tiempo de ejecución de la tarea. Una tarea con la laxitud más corta tiene mayor preferencia de ejecución.

La aplicación ejecuta una serie de tareas de manera secuencial y con máxima prioridad para garantizar un óptimo funcionamiento del algoritmo.

Java en tiempo real

En la búsqueda de una plataforma para ejecutar el algoritmo en tiempo real, para proveer soporte en aplicaciones que así lo requieran en un futuro y garantizar los tiempos de respuesta requeridos para que la Gateway cumpla apropiadamente su función, se analizó RTLinux y RTAI (*RealTime Application Interface for Linux*), sistemas con gran potencial y con licencia abierta pero que presentan falencias por no incorporar los drivers requeridos para una conexión USB con la Gateway. Debido a que estos sistemas de tiempo real duro no facilitaron la implementación del algoritmo se optó por utilizar JavaRT, un lenguaje que ofrece los drivers necesarios y donde algunas de sus ventajas más destacadas son:

- Portabilidad.
- Seguridad.
- Popularidad.
- Distribuido.
- Multi-hilo.

A pesar de estas ventajas, *JavaRT* también posee una serie de limitaciones en sus aplicaciones de tiempo real. Las más mencionadas son:

• *Falta de rendimiento debido a la interpretación de código*, el cual es un código intermedio independiente de la máquina que es interpretado en tiempo de ejecución y, por tanto, disminuye el rendimiento. Esto ha sido mejorado con el uso de la compilación *just-in-time*, incluida en las versiones más recientes de *Java*, donde se compilan segmentos de código en tiempo de ejecución en lugar de ser interpretados [17].

• *Falta de predictibilidad en tiempo de ejecución o comportamiento no determinista*, debido a la gestión dinámica de memoria que podría interrumpir la ejecución por intervalos de tiempo impredecibles [17]. Este problema ha sido mitigado con el manejo de memoria tipo inmortal, donde la gestión y mantenimiento la realiza únicamente la aplicación, evitando retardos causados por el "recolector de basura" de la máquina virtual de Java. El problema del comportamiento no determinista se superó con la capacidad de crear hilos en tiempo real, esto permite tener un mejor control sobre procesos que requieran de eventos externos.

Pese a estas fallas el *Real-Time for Java Experts Group* (*RTJEG*), respaldado por *Sun Microsystems*, ha hecho frente a estas desventajas, desarrollando la *Especificación de Tiempo Real para Java (RTSJ, Real-Time Specification for Java)* que empezó desde marzo de 1999 y que ha tenido progresos significativos con versiones completas y totalmente funcionales como se mencionan en la siguiente sección.

RTSJ

La Especificación de Tiempo Real para Java, define cómo un sistema debe comportarse en un contexto de tiempo real. Estas especificaciones fueron desarrolladas durante varios años por expertos de Java y de aplicaciones en tiempo real.

Una de las principales ventajas de *RTSJ* es que permite que los programas puedan combinar código *Java* de tiempo real y código estándar interaccionando en el mismo ambiente de ejecución.

La versión 1.1 de *RTSJ* fue emitida en 2006 y se mantiene en desarrollo para lograr que *Java* sea efectivamente adecuado para tiempo real. Provee adiciones para que *Java* pueda ser utilizado para sistemas de tiempo real, entre las cuales están:

• Extensiones a la especificación de la *JVM*, para una *máquina virtual de Java de tiempo real* (plataforma de ejecución).

• Creación de una *API* de tiempo real (paquete javax.realtime) para desarrollar programas *Java* de tiempo real (plataforma de desarrollo).

API de RTSJ

La *RTSJ* produce un paquete adicional (*javax.realtime*) que define varias clases e interfaces con el fin de permitir la programación para tiempo real. Esta especificación es materializada por *Sun Microsystems* en su API más reciente, la versión *Java RTS 2.1*, la cual fue utilizada para el desarrollo del algoritmo de localización. A continuación se realiza una visión general de las clases de la *RTSJ* que soportan: relojes y valores de tiempo, planificación e hilos de tiempo real.

La Figura 1.8 [17] presenta las clases e interfaces de la *RTSJ* para manejo de tiempo. La clase abstracta *Clock* permite definir, por defecto, un reloj de tiempo real que avanza en sincronía con el mundo externo y que expresa el tiempo con una precisión de *nanosegundos*. La clase HighResolutionTime permite representar el valor de tiempo dado en *mili* y *nanosegundos*. Ya que esta clase es abstracta, se utilizan sus clases derivadas. Por una parte, AbsoluteTime mide el lapso del tiempo transcurrido desde una hora y fecha fijas hasta un punto específico, mientras que RelativeTime permite la medición de un intervalo de tiempo entre dos instantes determinados [18].

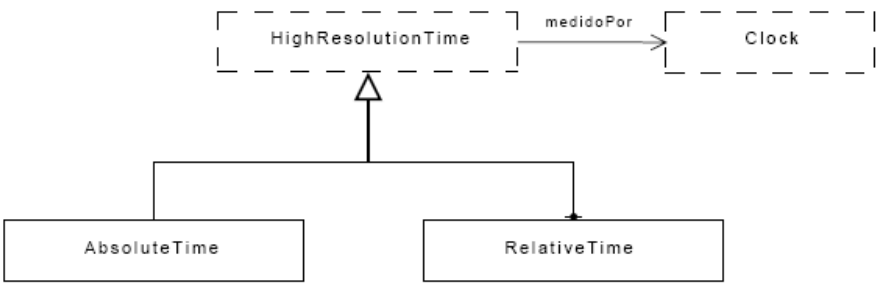

Figura 1.8 Clases RTSJ para manejo de tiempo.

Anteriormente se mencionó la importancia de la planificación en sistemas de tiempo real. Aunque *RTSJ* explícitamente soporta planificación basada en prioridad a través de la clase *PriorityScheduler* (Figura 1.9 [17]), es posible que alguna implementación soporte otro tipo de planificador. Consecuentemente, *Scheduler* es una clase raíz con una clase derivada definida: *PriorityScheduler* [18].

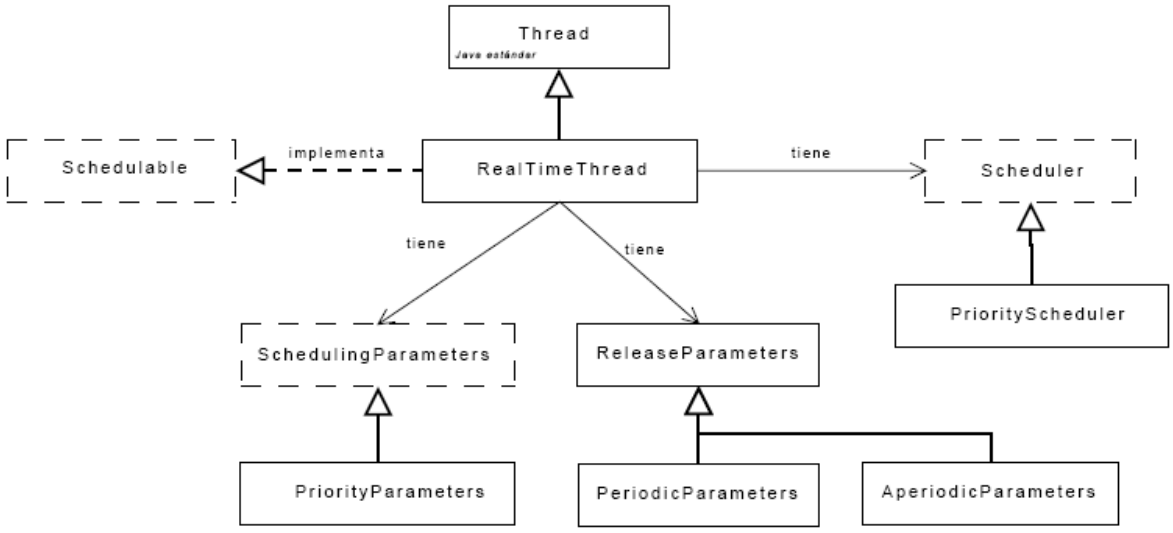

Figura 1.9 Clases para hilo de tiempo real y planificación.

En general, las aplicaciones están constituidas por múltiples hilos de ejecución (*threads*). Dentro de *RTSJ se* destaca la creación de un nuevo tipo de hilo, un hilo de tiempo real que es una extensión de la clase *Thread* de Java estándar, denominado *RealtimeThread*. Los hilos de tiempo real pueden tener asociados parámetros de planificación (*SchedulingParameters*) y parámetros de activación (*ReleaseParameters*).

La clase *SchedulingParameters* provee la clase raíz desde la cual un rango de posibles criterios de planificación puede ser expresado. *RTSJ* define únicamente un criterio, basado en prioridad mediante la clase *PriorityParameters*. Este parámetro es utilizado por el planificador para determinar cuál hilo de tiempo real es elegible para ejecución.

La jerarquía de clases a partir de *ReleaseParameters* especifica los requerimientos de activación de un hilo de tiempo real. Se identifican tipos de activación periódica (de manera regular) y aperiódica (al azar), mediante las clases *PeriodicParameters* y *AperiodicParameters*, respectivamente.

Todas las clases correspondientes a parámetros de activación tienen parámetros asociados, encapsulan un valor de *cost* y un *deadline*. El *cost* es la cantidad máxima de tiempo de procesador (tiempo de ejecución) necesitado para ejecutar el hilo de tiempo real cada vez que es activado. El *deadline* es el tiempo en el cual el hilo de tiempo real debe haber finalizado la ejecución actual. La

clase *PeriodicParameters* incluye dos componentes: un valor de tiempo *start* para la primera activación y el intervalo de tiempo (*period*) entre activaciones. Estos datos se corresponden con los parámetros de las tareas de tiempo real (tiempo de cómputo, plazo de respuesta, periodo y tiempo de inicio) [18].

1.3. Herramientas Hardware

En el proyecto se utilizó el kit de desarrollo Profesional de la empresa Crossbow y un equipo de cómputo portátil para la aplicación de tiempo real y la interfaz gráfica.

1.3.1. Nodos IRIS

El kit de Crossbow consta de:

- 1 Gateway con conexión USB.
- 1 Nodo de pruebas.
- 1 Placa para programar.
- 6 Nodos IRIS.

Figura 1.10 Gateway y Nodo Iris del kit Crossbow.

Cada uno de los nodos IRIS tiene las siguientes características:

Procesador	**ATmega1281, Atmel**
Memoria de programa Flash	128Kb
RAM	8Kb
Configuración EEPROM	4Kb
Comunicación serial	UART
Conversor Análogo a digital	10 bit ADC
Transceptor RF	**RF230, Atmel**

Frecuencia	2.4GHz
Tasa de transmisión	250Kbps
Sensibilidad de recepción	-101 dBm
Electromecánica	
Batería	2 baterías AA
Tamaño(mm)	58 x 32 x 7

Tabla 1. Características técnicas del Nodo IRIS [21].

1.3.2. Equipo de Computo

Se utilizó un equipo portátil marca *Hewlett Packard* serie DV 6000 con las siguientes características técnicas:

- Procesador AMD Turion 64 X2 1.8 GHz.
- Memoria RAM DDR2 2 Gb.
- Disco duro de 160 Gb.
- Pantalla ancha de 15.4'' WXGA con BrightView.
- 3 puertos USB.
- DVD+-RW DL / CD+-RW light scribe.

En este capítulo se mostró que *Zigbee* es un protocolo de comunicaciones con las características requeridas para una red con recursos hardware limitados, este protocolo está basado en el estándar IEEE 802.15.4 que provee las recomendaciones necesarias para trabajar en una WSN. *TinyOS* es el sistema operativo que ofrece las mejores herramientas para crear aplicaciones para una red de Sensores Inalámbricos, su adaptación a los diferentes nodos del mercado y facilidad para interactuar con hardware externo, hacen de este la mejor elección. Entre las diversas opciones para desarrollar aplicaciones en tiempo real, se encontró que *JavaRT* brinda las capacidades necesarias para la aplicación desarrollada gracias a su adaptación a comunicaciones por puertos USB.

1.4. Otros proyectos de localización en WSN

Proyectos realizados para obtener la posición geográfica de los nodos de una red de sensores inalámbricos:

- *Localization for Wireless Sensor Networks*, Rong Peng, 2003. Este proyecto se basa en el nivel de señal recibida (RSSI) para determinar la distancia entre nodos, pero tiene falencias en entornos *indoor* debido a la interferencia presentada por los diferentes materiales que debe atravesar la señal y tiene una aproximación de 3 o 1 metro en el mejor de los casos. Los nodos utilizados son de la primera generación [22].

- *Automatic Fault Localization for Wireless Sensor Network Software Applications: a Statistical Fault Divergence Approach*, Dr. W. K. Chan. Proyecto que busca evaluar y proponer técnicas para la localización de nodos en una red de sensores inalámbricos pero no se especifica que tipos de métodos serán utilizados como tampoco los equipos a utilizar [23].

- *Robot-Assisted Localization Techniques for Wireless Image Sensor Networks*, Hattie dong, Huang Lee, 2006. Esta solución presenta un sistema basado en mapas visuales y asistencia de nodos con sensores de movimiento para ubicar un robot. En este caso es necesario una evaluación y preparación del terreno donde se va a desplegar la red, además la red desplegada es utilizada para asistir a unos pocos nodos con posibilidad de movimiento [24].

- *Collaborative Localization in Wireless Sensor Networks*, Rodríguez Laura, 2007. Análisis de algoritmos para la localización de nodos en una red de sensores inalámbricos utilizando colaboración entre nodos y teniendo en cuenta factores como el número y densidad de nodos, obstáculos, irregularidades del terreno, topología de la red y movilidad de los nodos. Este trabajo presenta un panorama teórico importante para determinar los factores relevantes en un algoritmo a usar en un proyecto de localización de nodos, sin embargo no genera componentes software para los nodos como tampoco tiene en cuenta las limitaciones de los nodos de diferentes fabricantes [25].

- *Localization of Wireless Sensor Networks with a Mobile Beacon,* Mihail L. Sichitiu, Vaidyanathan Ramadurai. Sistema basado en un nodo que sirve como faro el cual debe moverse por el entorno recibiendo mensajes guías de los otros nodos para estimar su posición, el inconveniente se presenta cuando no recibe un paquete por la lejanía del nodo transmisor lo cual implica que no se pueda referenciar la posición de este nodo [26].

Los sistemas basados en GPS son una propuesta teórica que no tiene aplicación en entornos *indoor* porque no ofrecen cobertura en estos lugares, además implica un aumento considerable en el precio y consumo de energía de cada nodo.

El objetivo de este proyecto fue desarrollar un algoritmo para mapear la posición geográfica de cada nodo en una red de sensores inalámbricos sin hardware extra en un entorno *indoor* o a campo libre sin limitaciones de movimiento.

CAPITULO 2

ALGORITMO DE LOCALIZACION

Este capítulo describe en detalle nuestro sistema de localización de nodos en una red de sensores inalámbricos. Se presentan las técnicas de localización utilizadas en sistemas inalámbricos, el mecanismo de estimación de distancia basado en el *RSSI* (*Received Signal Strength Indication*), el desarrollo matemático que permite obtener las coordenadas de cada nodo y los diagramas correspondientes a cada dispositivo que conforma el sistema.

2. Técnicas de localización

Las técnicas de localización para una red inalámbrica se basan en la estimación de distancia entre dispositivos. Esta distancia se puede obtener por diferentes métodos dependiendo de las capacidades de los elementos de la red, algunas técnicas utilizan el tiempo de las comunicaciones, otras el ángulo de recepción y finalmente la potencia de la señal recibida.

2.1. Técnicas de estimación de distancias

En los últimos años se han desarrollado técnicas que permiten estimar la distancia entre dos nodos en el espacio, algunas de las más aceptadas se mencionan a continuación [1].

Tiempo de llegada (ToA): Esta técnica mide el tiempo que tarda una señal en viajar de un nodo a otro a una velocidad conocida. Esta técnica requiere de una alta sincronización entre el emisor y el receptor. Para implementar esta técnica es necesario que los nodos tengan procesadores con velocidades alrededor de 1GHz para poder diferenciar el tiempo de la comunicación y que no quede enmascarado con los tiempos de respuesta del equipo. Lamentablemente los nodos disponibles en el mercado no superan los 16MHz, por lo cual no es posible implementar esta técnica sin la intervención externa de otro tipo de dispositivos.

Diferencia de tiempos de llegada (TDoA): En esta técnica el transmisor emite un pulso acústico y uno de radio, el receptor compara el tiempo de llegada de cada uno de los pulsos y se hace una relación con la velocidad de vuelo de cada uno de ellos para así determinar la distancia entre los nodos. El inconveniente de esta técnica es que se requiere de dos transmisores y dos receptores lo cual incrementa el tamaño y el consumo de energía. Al igual que con la técnica *ToA* se requiere de equipos muy rápidos.

Ángulo de llegada (AoA): El ángulo de llegada de la onda incidente (proveniente del nodo desconocido) se mide en relación a una dirección de referencia (con relación a una orientación). Cada nodo debe poseer hardware adicional que permite medir el ángulo de incidencia, lo cual incrementa el costo de la WSN. Este hardware adicional son antenas inteligentes que tienen la capacidad de modificar su patrón de radiación para obtener un mejor rendimiento y que en este caso permite conocer la dirección del nodo transmisor. Los nodos de la empresa Crossbow utilizados en este proyecto no cuentan con este tipo de antenas y su adaptación no fue

considerada por incurrir en la modificación de los dispositivos y por incrementar el costo del proyecto.

Atenuación: En esta técnica el receptor utiliza el nivel de *RSSI* para realizar una correlación entre la atenuación de la potencia original de emisión y la distancia hasta el receptor. Aunque en esta técnica los fenómenos como el desvanecimiento por múltiples trayectorias y sombras hacen que la correlación no sea muy precisa, ha sido utilizada en este proyecto puesto que no requiere de ningún hardware adicional.

2.2. Técnicas de estimación de la posición

Actualmente existen diversas técnicas para estimar la posición de un nodo en una WSN, las cuales se pueden usar individualmente o combinándolas, si es posible, para una mayor precisión. Entre estas técnicas se tienen [1]:

Triangulación: La triangulación hace uso de la trigonometría para determinar las coordenadas de un punto mediante la solución de un conjunto de ecuaciones obtenidas a partir de las coordenadas de dos puntos de referencia (en el caso bidimensional) y los ángulos entre la señal emitida y estos puntos (Figura 2.1 [1]). Para esta técnica los nodos receptores deben ser capaces de medir el ángulo de incidencia de la señal emitida por el nodo a ubicar, motivo por el cual no se utilizó en el proyecto.

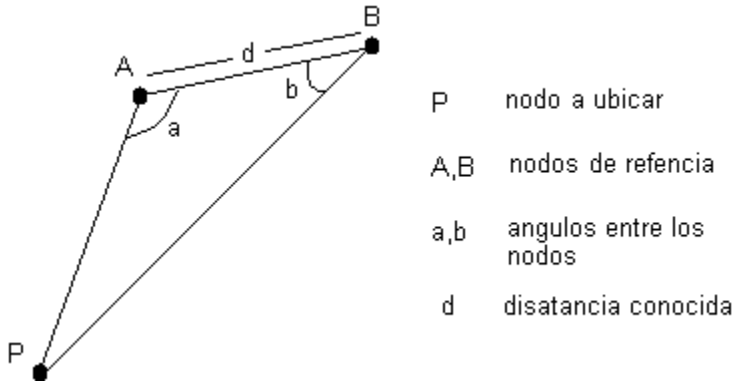

Figura 2.1 Estimación por Triangulación (plano bidimensional).

Multilateración: Esta técnica es muy similar a la de triangulación, con la diferencia de que ya no se miden los ángulos sino las distancias y donde los puntos de referencia deben ser fijos y conocerse con anterioridad sus coordenadas para poder estimar la del nodo a ubicar (Figura 2.2 [1]). Estas condiciones limitan la movilidad de cada elemento y requiere de una preconfiguración de algunos nodos dentro de la red, características adversas a los objetivos de libre movilidad para todos los nodos en el proyecto.

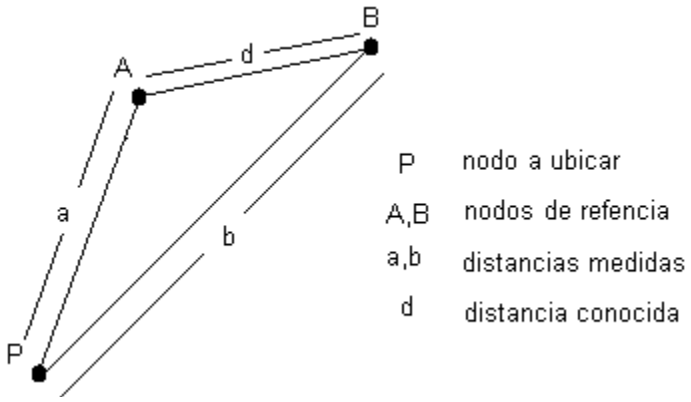

Figura 2.2. Estimación por Multilateración (plano bidimensional).

Trilateración: Esta técnica calcula la posición de un nodo midiendo las distancias desde él mismo hasta varios puntos de referencia (al menos tres en el caso bidimensional) para luego trazar círculos con radio igual a cada una estas distancias, lo que da como resultado un punto de intersección que determina de forma única y precisa la localización del nodo (Figura 3.3). Esta técnica no requiere de un hardware adicional en los nodos puesto que la distancia se puede estimar mediante la técnica de atenuación, por lo que es utilizada en el proyecto, pero los puntos de referencia que se toman no son nodos fijos sino que también tienen la posibilidad de moverse.

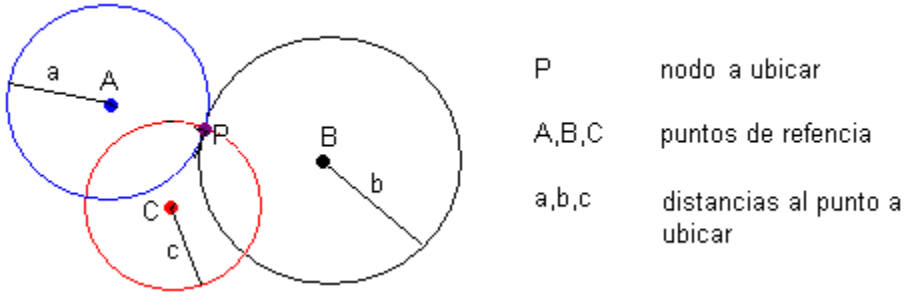

Figura 3.3: Estimación por Trilateración (plano bidimensional).

2.3. RSSI como indicador de distancia.

Como se mencionó en la sección 2.1, el nivel de *RSSI* puede ser utilizado como un indicador de la distancia de nodo a nodo. El problema con este indicador, es que sufre inexactitudes provocadas por diferentes fenómenos propios de una transmisión inalámbrica, entre las que se encuentran [20]:

*Multitrayectos***:** Estos se originan cuando la señale de RF se dispersa y toma diferentes caminos. Estas señales multitrayecto llegan al receptor con diferentes amplitudes y fases, añadiéndose de forma constructiva o destructiva en función a la frecuencia.

Sombras: Es la atenuación de la señal debido a obstáculos en el camino que la señal debe pasar a través de la difracción o alrededor del objeto. Estos suelen ser considerados fuentes de error aleatorio.

Variabilidad del Transmisor: Diferentes emisiones se comportan de forma diferente, incluso cuando están configuradas exactamente de la misma manera. Cuando el transmisor está configurado para enviar paquetes a un nivel de potencia determinado, el transmite los paquetes a un nivel de potencia que se encuentra muy cerca al configurado pero no exactamente igual. Esto puede alterar la señal recibida y la indicación, por lo que puede llevar a la incorrecta estimación de la distancia.

Variabilidad del Receptor: La sensibilidad de los receptores de radio en los diferentes chips es diferente. El valor de RSSI registrado en diferentes receptores puede ser diferente, incluso cuando son del mismo fabricante y tienen las mismas características.

Orientación de la antena: Cada antena tiene su propio patrón de radiación, el cual no es uniforme. En la práctica, el *RSSI* registrado en el receptor de un nodo varía a medida que este o el nodo transmisor cambia la orientación la antena.

La fórmula para determinar la potencia recibida es la siguiente [20]:

$$P_{Rx} = P_{Tx} + G_{Tx} + G_{Rx} - l_{fs} \quad (1)$$

Donde:

- P_{Rx}: Potencia de recepción. Está definida por la siguiente ecuación de acuerdo al *Datasheet* del transceptor RF230 de Atmel [19]:

$$P_{Rx} = RSSI\_BASE\_VAL + 3\,(RSSI - 1) \quad (2)$$

  - P_{Rx}: Potencia de recepción.
  - RSSI_BASE_VAL: Constante de -91 dB [19].
  - RSSI: Valor obtenido por *TinyOS* después de recibir un mensaje.

  La sensibilidad del transceptor *RF230* que contienen los nodos de la empresa *Crossbow* está entre -10 dBm y -91 dBm, con saltos de 3 dBm. El valor mínimo de RSSI es de 1, con lo cual la potencia de recepción seria de -91 dBm, y el valor máximo es de 28 para obtener una potencia recibida de -10 dBm.

- P_{Tx}: Potencia de transmisión. Tiene un valor de 3 dBm [19].
- G_{Tx}: Ganancia correspondiente a la antena y dispositivos en el transmisor.
- G_{Rx}: Ganancia correspondiente a la antena y dispositivos en el receptor.

  Las ganancias de transmisión y recepción se obtuvieron conjuntamente mediante pruebas, los datos obtenidos indican una variación entre -16.94 dB hasta -15.90 dB, por lo cual se tomó el promedio de -16.42 dB.

- l_{fs}: Perdidas en el medio. Dependen de la frecuencia de transmisión que es igual a 2400 MHz, banda en la que transmiten los nodos de la empresa *Crossbow*. Como también dependen de la distancia entre el transmisor y el receptor y el entorno en el que se realiza la comunicación. Para este caso se tomo el índice de espacio abierto, por tal motivo la ecuación es [20]:

$$l_{fs} = 32.4 + 20 \log f\,(Mhz) + 20 \log D\,(km) \qquad (3)$$

Con las ecuaciones (1), (2) y (3) es posible relacionar la distancia entre los nodos con el nivel de señal recibido o *RSSI* y el resultado de reemplazar valores y las ecuaciones (2) y (3) en (1) es:

$$20 \log D\,(km) = -3\,RSSI - 19.47dBm \qquad (4)$$

Esta ecuación es la utilizada en el algoritmo de localización para estimar la distancia entre nodos. La independencia de la estimación de distancia y de coordenadas permite utilizar en un futuro métodos más confiables como aquellos basados en tiempo.

2.4. Sistema de localización implementado.

El planteamiento matemático desarrollado no se basa en sistemas estudiados o de los cuales se tenga conocimiento de su implementación, su concepción y pruebas se realizaron en el marco del proyecto.

El algoritmo obtiene las coordenadas de cada nodo respecto a los demás nodos de la red y a la Gateway o estación central, la cual siempre estará ubicada en la posición (0,0).

La estación central recibe de cada nodo la información de quienes son sus vecinos y el valor RSSI capturado en las transmisiones respectivas, con estos datos el algoritmo inicia el proceso de estimación de posición de cada nodo por separado.

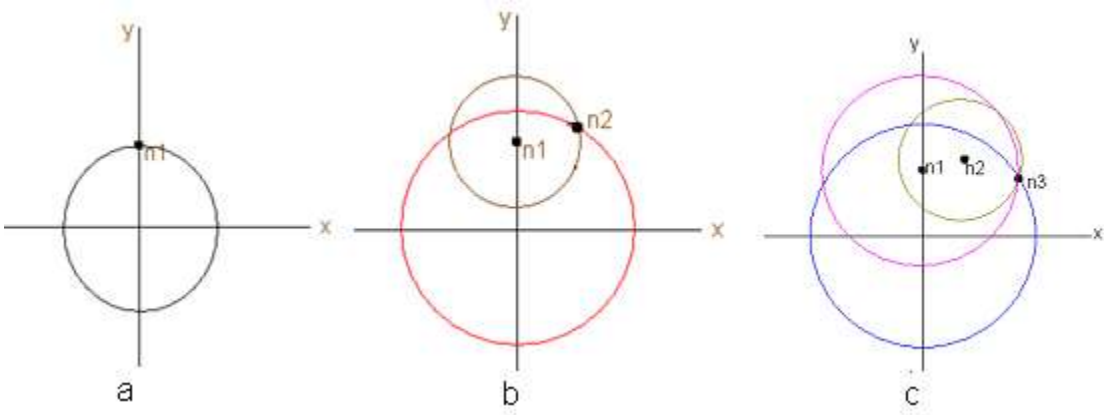

Figura 2.4 Ubicación de los tres primeros nodos.

Cuando se conecta el primer nodo, solo podemos decir que se encuentra a una distancia d_1 de la estación central (Figura 2.4.a.). Las coordenadas de este nodo se ubican sobre el eje Y de la siguiente forma:

$$x_1 = 0 \qquad y_1 = d_1 \quad (5)$$

Al conectarse el segundo nodo, se calcula la distancia con respecto al primer nodo y a la estación central (Figura 2.4.b), dando como resultado dos puntos de intersección de los cuales se opta por el punto sobre el eje positivo de X. El planteamiento matemático para obtener las coordenadas del nodo 2 lo relacionan con la posición de la central y su distancia (Ecuación 6) como al nodo 1 con el 2 (Ecuación 7) de la siguiente forma:

$$x_2^2 + y_2^2 = d_2^2 \qquad (6)$$

$$(x_2 - x_1)^2 + (y_2 - y_1)^2 = d_{21}^2 \qquad (7)$$

Donde:

- (x_1, y_1) Coordenadas del nodo 1. Conocidas.
- d_2 Distancia del nodo 2 a la central. Conocida.
- (x_2, y_2) Coordenadas del nodo 2. A obtener.
- d_{21} Distancia del nodo 2 al nodo 1. Conocida.

Retomando de (5) donde $x_1 = 0$, se sustituye en (7) y se obtiene:

$$x_2^2 + y_2^2 - 2y_2y_1 + y_1^2 = d_{21}^2 \quad (8)$$

Se reemplaza (6) en (8):

$$d_2^2 - 2y_2y_1 + y_1^2 = d_{21}^2 \qquad (9)$$

$$2y_2y_1 = d_2^2 + y_1^2 - d_{21}^2$$

$$y_2 = \frac{d_2^2 + y_1^2 - d_{21}^2}{2y_1} \qquad (10)$$

El proceso garantiza que la variable y_1 es diferente de cero para evitar una inconsistencia. La ecuación (10) es utilizada para hallar la coordenada en Y del nodo 2. Para hallar la coordenada de X se despeja de la ecuación (6):

$$\sqrt{x_2^2 + y_2^2} = d_2 \qquad (11)$$

$$x_2 = \pm\sqrt{(d_2^2 - y_2^2)} \qquad (12)$$

Como x_2 tiene dos posibles valores, por conveniencia se toma el positivo, situación que podría afectar la distribución geográfica de la red, pero que depende del punto de vista del observador, en tal caso de tener las posiciones en sentido contrario, basta con multiplicar las coordenadas de *X* de todos los nodos por -1 para girar las posiciones con respecto al eje *Y*. Esta solución esta implementada en la interfaz gráfica del sistema por medio de un botón visible.

Cuando se conecta un tercer nodo, se tiene como puntos de referencia los nodos ya ubicados y la estación central. Calculando la distancia de este nodo con respecto a los puntos de referencia se pueden trazar tres círculos los cuales dan un único punto de intersección que es donde se encuentra el nodo 3 (Figura 2.4.c). Las coordenadas de este nodo se determinan mediante las siguientes ecuaciones:

$$x_3^2 + y_3^2 = d_3^2 \qquad (13)$$

$$(x_3 - x_1)^2 + (y_3 - y_1)^2 = d_{31}^2 \qquad (14)$$

$$(x_3 - x_2)^2 + (y_3 - y_2)^2 = d_{32}^2 \qquad (15)$$

Donde:

- (x_1, y_1) Coordenadas del nodo 1. Conocidas.
- d_{31} Distancia del nodo 3 al nodo 1. Conocida.
- (x_2, y_2) Coordenadas del nodo 2. Conocidas.
- d_{32} Distancia del nodo 3 al nodo 2. Conocida.
- (x_3, y_3) Coordenadas del nodo 3. A obtener.
- d_3 Distancia del nodo 3 a la central. Conocida.

De (14):

$$x_3^2 - 2x_3x_1 + x_1^2 + y_3^2 - 2y_3y_1 + y_1^2 = d_{31}^2 \qquad (16)$$

Se reemplaza (13) en (16):

$$d_3^2 - 2x_3x_1 + x_1^2 - 2y_3y_1 + y_1^2 = d_{31}^2 \qquad (17)$$

Se multiplica (17) por x_2:

$$d_3^2x_2 - 2x_3x_1x_2 + x_1^2x_2 - 2y_3y_1x_2 + y_1^2x_2 = d_{31}^2x_2 \qquad (18)$$

Se realiza el mismo proceso con (15):

$$x_3^2 - 2x_3x_2 + x_2^2 + y_3^2 - 2y_3y_2 + y_2^2 = d_{32}^2 \qquad (19)$$

Se reemplaza (13) en (19):

$$d_3^2 - 2x_3x_2 + x_2^2 - 2y_3y_2 + y_2^2 = d_{32}^2 \qquad (20)$$

Se multiplica (20) por $-x_1$:

$$-d_3^2x_1 + 2x_3x_2x_1 - x_2^2x_1 + 2y_3y_2x_1 - y_2^2x_1 = -d_{32}^2x_1 \qquad (21)$$

Se suma (18) con (21):

$$d_3^2x_2 - 2x_3x_1x_2 + x_1^2x_2 - 2y_3y_1x_2 + y_1^2x_2 = d_{31}^2x_2 \quad +$$

$$-d_3^2x_1 + 2x_3x_2x_1 - x_2^2x_1 + 2y_3y_2x_1 - y_2^2x_1 = -d_{32}^2x_1$$

$$d_3^2x_2 - d_3^2x_1 + x_1^2x_2 - x_2^2x_1 - 2y_3y_1x_2 + 2y_3y_2x_1 + y_1^2x_2 - y_2^2x_1 = d_{31}^2x_2 - d_{32}^2x_1 \qquad (22)$$

Los términos sombreados se cancelan y se procede a simplificar la ecuación resultante:

$$2y_3(y_1x_2 - y_2x_1) = x_2(d_3^2 + x_1^2 - x_2x_1 + y_1^2 - d_{31}^2) + x_1(d_{32}^2 - d_3^2 - y_2^2)$$

$$y_3 = \frac{x_2(d_3^2 + x_1^2 - x_2x_1 + y_1^2 - d_{31}^2) + x_1(d_{32}^2 - d_3^2 - y_2^2)}{2(y_1x_2 - y_2x_1)} \qquad (23)$$

La ecuación (23) es utilizada en el algoritmo para obtener la coordenada del eje *Y*. De la ecuación (17) se despeja x_3:

$$x_3 = \frac{d_3^2 + x_1^2 - 2y_3y_1 + y_1^2 - d_{31}^2}{2x_1} \qquad (24)$$

Con la ecuación (24) ya se conocen las coordenadas del nodo 3, pero debido a la inexactitud de las distancias obtenidas por el *RSSI* se hace necesario hacer un proceso de verificación utilizando la ecuación (13):

$$x_3 = \pm\sqrt{d_3^2 - y_3^2} \qquad (25)$$

La diferencia del RSSI entre el lóbulo principal y el trasero puede ser de hasta 5 unidades (Figura 3.1). Para minimizar el impacto de este fenómeno, se realiza una simulación rotando la orientación del nodo. Se parte de la suposición de que el valor recibido fue por el lóbulo trasero, y luego se aumenta el valor de RSSI en una unidad por vez hasta completar cinco cálculos que simularían el lóbulo principal. Este procedimiento se hace en las dos distancias para el segundo nodo y en las tres para el tercer nodo y los siguientes. Para el segundo nodo se obtienen hasta 25 posibles posiciones; luego de hacer la verificación matemática de las posiciones factibles se reduce este número hasta en un 70%. Con las coordenadas matemáticamente posibles se hace un promedio y de esa manera se obtiene la posición del nodo. Para el caso del tercer nodo y los siguientes, la

simulación se hace para 4 unidades en el RSSI, como se tienen tres distancias, las posibilidades ascienden a 64, donde luego de la verificación matemática se reduce este número hasta en un 85%, la selección de la posición del nodo se realiza haciendo también un promedio.

Cuando se conecta un cuarto nodo, se utilizan las mismas ecuaciones empleadas para hallar las del tercer nodo ((23) y (24)), pero en este caso las coordenadas (x_1, y_1) corresponden al nodo dos y (x_2, y_2) al nodo tres, de esta manera los cálculos no están sujetos a dos nodos en particular y obedecen al orden secuencial en que se conecto cada nodo. Este mecanismo es el mismo para calcular la posición del resto de nodos conectados en la red y ofrece mayor exactitud en la posición relativa cuando hay un mayor número de nodos conectados, gracias a la independencia que tiene cada nodo con respecto a los vecinos que debe escuchar.

La limitación en el cálculo de la posición de nodos activos en la red por ciclo, está sujeta a los tiempos empleados para comunicar la Gateway con cada uno de los nodos, cálculos de la posición y almacenamiento de las coordenadas en la Base de Datos. Por tal motivo, se midieron los tiempos utilizados por el sistema, estos resultados son el promedio de 10 ciclos bajo las mismas condiciones y se pueden observar en la siguiente tabla:

No. de Nodos Conectados	**Posición (Movimiento de los nodos)**	**Tiempo empleado**			
		Conexión con Nodos (ms)	**Cálculo de Posición (us)**	**Guardar en BD (ms)**	**Tiempo Total (ms)**
0		24	45	23	47
1	Estático	43	221	24	67
1	Movimiento	44	279	25	69
2	Estático	74	233	27	101
2	Movimiento	72	1239	24	97
3	Estático	100	240	24	124
3	Movimiento	99	3328	23	122
4	Estático	126	259	27	153
4	Movimiento	127	4759	21	148

Tabla 2. Tiempos empleados por el Sistema.

Se puede observar que los tiempos de comunicación entre la Gateway y los nodos se comportan de manera lineal. Los cambios de tiempo en cálculo, se deben al uso de ecuaciones más complejas y a la simulación para corregir las posiciones, cuando no se registra cambios de posición, la aplicación no realiza cálculos, es por este motivo que los tiempos cuando los nodos están estáticos son mínimos. Por otra parte, el tiempo para almacenar las coordenadas en la Base de Datos es estable y lineal.

El ciclo total del hilo en tiempo real es de 500 ms, de los cuales se emplean máximo 300 ms para la comunicación entre los nodos y la Gateway, con este tiempo es posible atender hasta 10 nodos para permitirle al equipo de computo realizar otras actividades propias del sistema operativo y a la WSN estar activa solo el tiempo necesario, esta característica permite a los nodos, tener la

posibilidad de pasar a un estado inactivo para ahorrar energía. Si la aplicación de la WSN requiere un número mayor de nodos conectados, se hace necesario hacer grupos de nodos para atender en ciclos consecutivos, la respuesta del sistema al tener 5 grupos de 10 nodos es equivalente a tener solo 1 grupo de 10 nodos, debido a que el proceso completo por cada ciclo es el mismo en ambos casos. Se tendría la limitante de que la información de cada nodo solo se actualizaría en el momento en que se atienda al grupo al que pertenece, pero esto depende directamente de la aplicación de la WSN y del protocolo de comunicación que se establezca.

2.5. Diagramas de casos de uso.

El sistema de localizacion consta de cuatro aplicaciones *software*:

- En el Nodo: **LocNodo**. Módulo sobre *TinyOS* encargado de unirse a la red, obtener informacion de sus vecinos y enviarla a la estacion central.
- En la estacion central (Gateway): **LocCentral**. Modulo sobre *TinyOS* que recibe y envia informacion al PC por USB. Coordinador de la red de sensores y sumidero de la informacion recopilada por cada nodo para enviarla al PC posteriormente.
- En el PC: **LocPC**. Aplicación de tiempo real en *JavaRT* que se comunica con la estacion central por USB. Contiene la lista de los nodos activos de la red, realiza los calculos de posicion para cada nodo y almacena esa informacion en la base de datos.
- En el PC: **LocMapa**. Interfaz grafica desarrollada en Java, que obtiene las coordendas de los nodos leyendo la base de datos y los despliega graficamente en la pantalla del PC.

Adicionalmente se utiliza una base de datos en MySQL para almacenar las coordenadas de los nodos, que consta de una tabla con los campos: *id* (identificador del nodo), *x* (coordenada del eje *X*), *y* (coordenada del eje *Y*).

A continuación se describen los casos de uso del usuario y de cada una de las aplicaciones software que componen el sistema.

2.5.1. Usuario.

Figura 2.5 Caso de uso del Usuario.

Caso de uso ver posición	
Iniciador	Usuario
Propósito	Permitir que el usuario vea la

	posición de los nodos de manera gráfica.
Resumen	El usuario inicia el programa **LocMapa** con el cual una interfaz grafica muestra la posición de los nodos de la WSN.

2.5.2. Nodos.

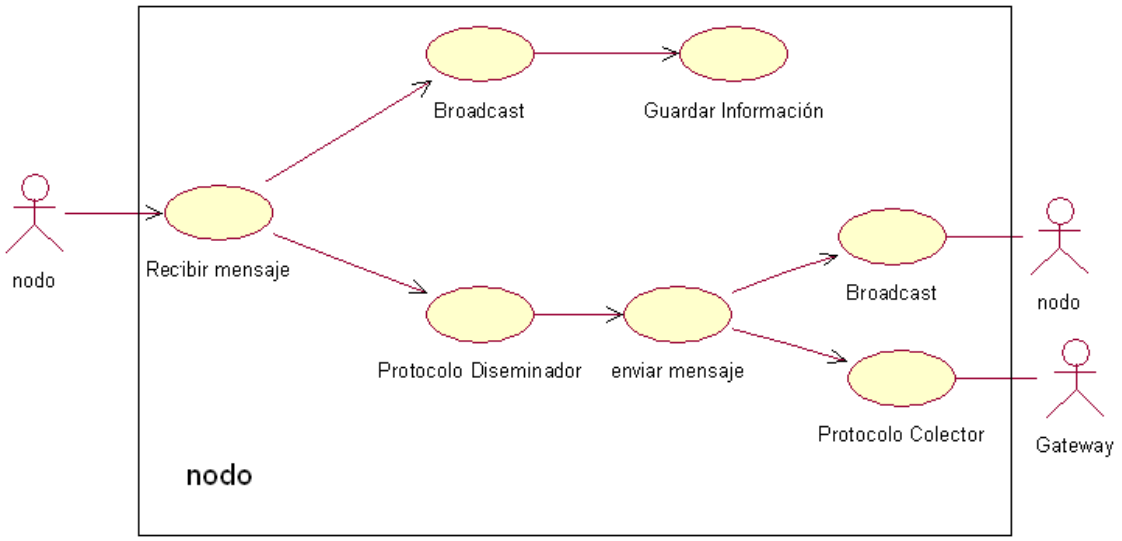

Figura 2.6 Casos de uso del Nodo.

Caso de uso recibir mensaje	
Iniciador	Nodo
Propósito	Recolectar datos de los nodos vecinos y de la estación central para saber en qué momento debe enviarlos a la Gateway.
Resumen	Si el nodo recibe un mensaje tipo *broadcast*, este va a guardar esa información correspondiente al identificador del vecino y el *RSSI* de esa comunicación. Si el mensaje ha sido enviado usando el protocolo Diseminador (*Dissemination,* Sección *1.1.2)* y corresponde a la autorización para enviar sus datos a la estación central, el nodo envía la información recolectada de sus vecinos.

Caso de uso enviar mensaje	
Iniciador	Nodo
Propósito	Permitir que los nodos vecinos conozcan la ubicación de este y enviar los datos recolectados a la estación central.
Resumen	El nodo al recibir la petición de datos desde la Gateway o estación central, envía un mensaje *broadcast* para que los vecinos sepan a qué distancia se encuentra este y luego envía los datos recolectados a la Gateway utilizando el protocolo Colector (*Collector,* Sección *1.1.2).*

Caso de uso guardar información	
Iniciador	Nodo
Propósito	Guardar datos que permitan saber la posición de los nodos vecinos.
Resumen	Al recibir un mensaje tipo *broadcast*, el nodo guarda el *RSSI* y el identificador del nodo que envió este mensaje.

2.5.3. Gateway o estacion central.

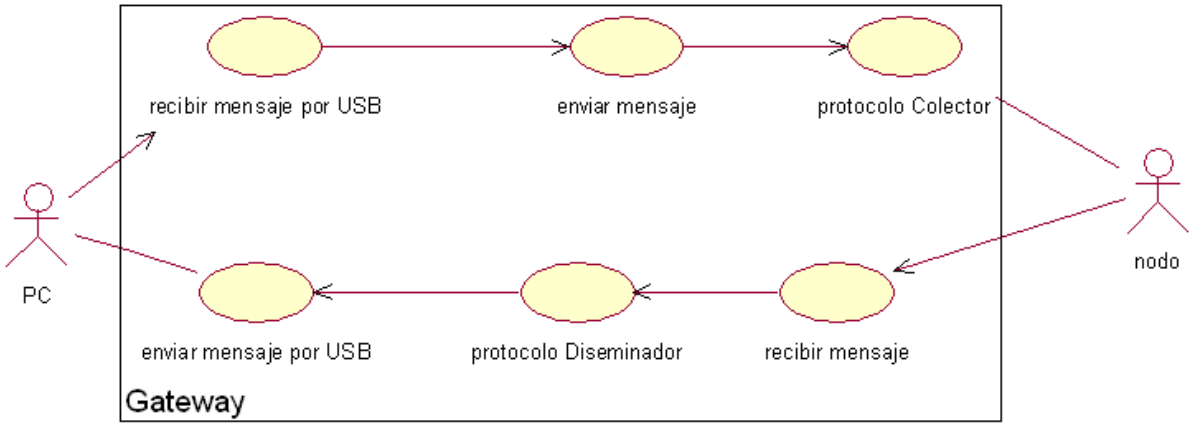

Figura 2.7 Casos de uso de la Gateway.

Caso de uso recibir mensaje por USB	
Iniciador	Gateway
Propósito	Guarda la lista de los nodos conectados a la red.
Resumen	Almacena la lista enviada desde el PC y le añade los nodos que se registraron desde el último ciclo. Esta lista permite llamar en orden a todos los nodos activos en la red.

Caso de uso enviar mensaje por USB	
Iniciador	Gateway
Propósito	Enviar los datos de un nodo por la interfaz USB al PC.
Resumen	Permite enviar la información recibida de un nodo a través de la interfaz USB. Estos datos contienen el *id* del nodo, de dos de sus vecinos y sus respectivos *RSSI*.

Caso de uso recibir mensaje	
Iniciador	Gateway
Propósito	Recolectar datos de todos los nodos de la red.
Resumen	Recibe la información proveniente de cada nodo de la red usando el protocolo Colector.

Caso de uso enviar mensaje	
Iniciador	Gateway
Propósito	Enviar solicitud a un nodo para que despache su información a la Gateway.
Resumen	Permite enviar un mensaje a todos los nodos de la red utilizando el protocolo Diseminador, con el identificador del nodo que debe enviar su información a la estación central.

2.5.4. Aplicación en el PC.

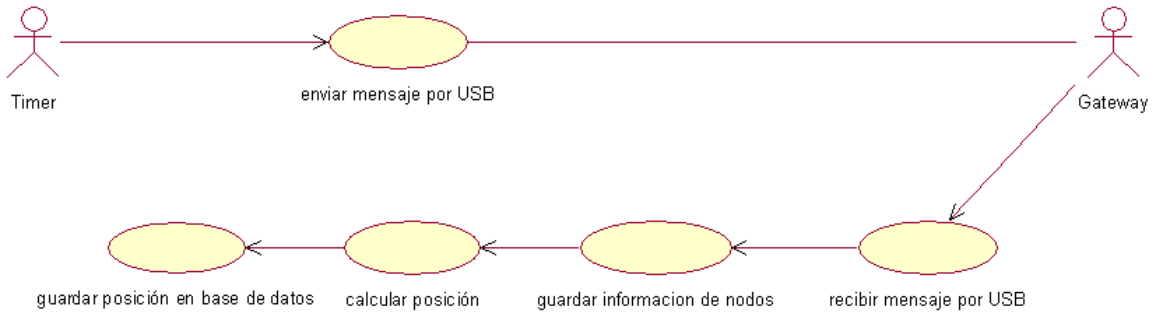

Figura 2.8 Casos de uso del PC.

Caso de uso enviar mensaje por USB	
Iniciador	PC
Propósito	Enviar datos a la Gateway.
Resumen	El PC hace una lista de los nodos registrados y la envía por el puerto USB a la Gateway.

Caso de uso recibir mensaje por USB	
Iniciador	PC
Propósito	Recibir datos desde la Gateway.
Resumen	El PC recibe por el puerto USB los paquetes enviados desde la Gateway, que contienen la información recolectada por los nodos (cada paquete contiene la información de un solo nodo).

Caso de uso guardar información de nodos	
Iniciador	PC
Propósito	Almacenar datos recolectados por los nodos
Resumen	Al terminar de recibir los paquetes enviados desde la Gateway, el PC guarda esta la información de cada nodo para luego calcular la posición de cada uno de ellos y actualizar la lista del orden de los nodos activos de la red.

Caso de uso calcular posición	
Iniciador	PC
Propósito	Calcular posición de cada nodo.
Resumen	En los datos guardados de cada nodo se encuentran los niveles de RSSI de los vecinos. El PC toma estos niveles y hace la estimación de las distancias de el nodo a los vecinos (o a la *Gateway*), luego con estas distancias realiza el proceso matemático que permite saber la posición del nodo.

Caso de uso guardar posición en base de datos	
Iniciador	PC
Propósito	Guardar posición de cada nodo en una base de datos.
Resumen	Al terminar de calcular la posición de cada nodo, el PC guarda estas posiciones en una base de datos de donde pueden ser tomadas por otras aplicaciones.

2.5.5. Interfaz gráfica.

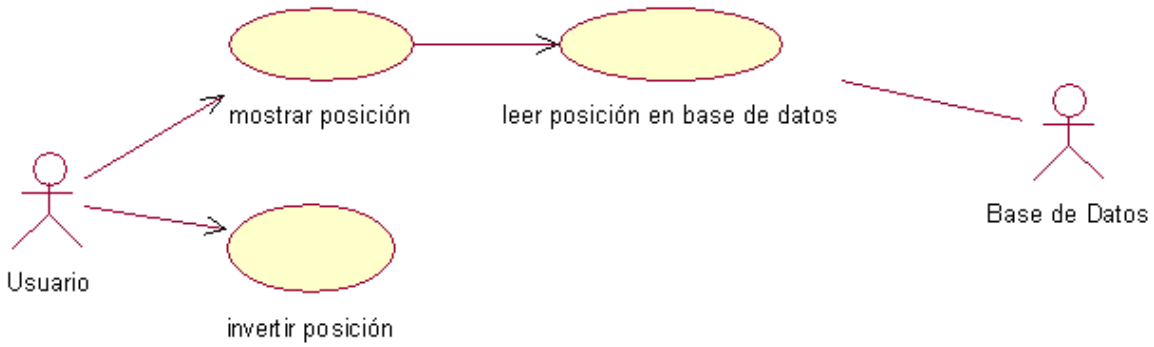

Figura 2.9 Caso de uso de la Interfaz Gráfica.

Caso de uso leer posición en base de datos	
Iniciador	Interfaz gráfica
Propósito	Extraer de una base de datos las posiciones de los nodos.

Resumen	La interfaz grafica establece una conexión a la base de datos y extrae la posición de cada nodo.

Caso de uso mostrar posición	
Iniciador	Interfaz gráfica
Propósito	Mostrar la posición de los nodos conectados mediante una interfaz grafica.
Resumen	Con las coordenadas de posición de cada nodo (extraídas de la base de datos) se colocan puntos (que representan los nodos) en un plano bidimensional, también se muestra una lista de los nodos conectados y sus coordenadas.

Caso de uso invertir posición	
Iniciador	Interfaz gráfica.
Propósito	Invertir la posición de los nodos.
Resumen	Si la posición de los nodos no corresponde con la distribución real de la red, se pueden invertir las coordenadas en el eje X. Esta característica es el resultado de la Ecuación 12, donde se opta por la coordenada positiva. Este hecho también depende del punto de observación, la distribución de la red se invierte si es vista desde ángulos diferentes.

2.6. Diagramas de secuencia.

En la figura 3.10 se muestra el diagrama de secuencia de la comunicación entre la Gateway y los nodos, la cual se describe a continuación.

- Al encenderse un nodo dentro de la red, este espera un tiempo máximo de 30 ms "escuchando" el canal para saber si hay otro nodo comunicándose, si transcurrido este tiempo no se "escucho" ninguna comunicación, se envía una petición de registro a la Gateway usando el protocolo Colector, de lo contrario espera otro lapso igual de tiempo hasta lograr comunicarse con la Gateway.

- La Gateway recibe la petición de registro en el tiempo de "registro de nuevos nodos" y envía un ACK usando el protocolo Diseminador para indicar al nodo que ya está registrado.

- Luego de terminar el tiempo de registro de nuevos nodos se pasa a un tiempo denominado "tiempo de recolección de datos", en el cual la Gateway solicita (a cada nodo registrado) los datos necesarios para realizar la localización de los nodos, esta petición la hace enviando un paquete mediante el protocolo Diseminador, dicho paquete contiene el identificador o *id* del nodo que debe enviar los datos.

- Todos los nodos de la red reciben el mensaje enviado por la Gateway, pero solo aquel que tenga el *id* referenciado en el mensaje envía un mensaje broadcast para que sus vecinos sepan a qué distancia se encuentra este respecto a ellos, luego envía los datos mediante el protocolo Colector a la Gateway.

- Al terminar de hacer la recolección de datos se libera el canal pasando así nuevamente al tiempo de registro de nuevos nodos.

- Si un nodo no envía los datos a la Gateway en un tiempo máximo de 40 ms, esta lo toma como desconectado y envía esa información al PC.

- Si un nodo anteriormente registrado no recibe la petición de datos desde la Gateway, este vuelve a registrarse.

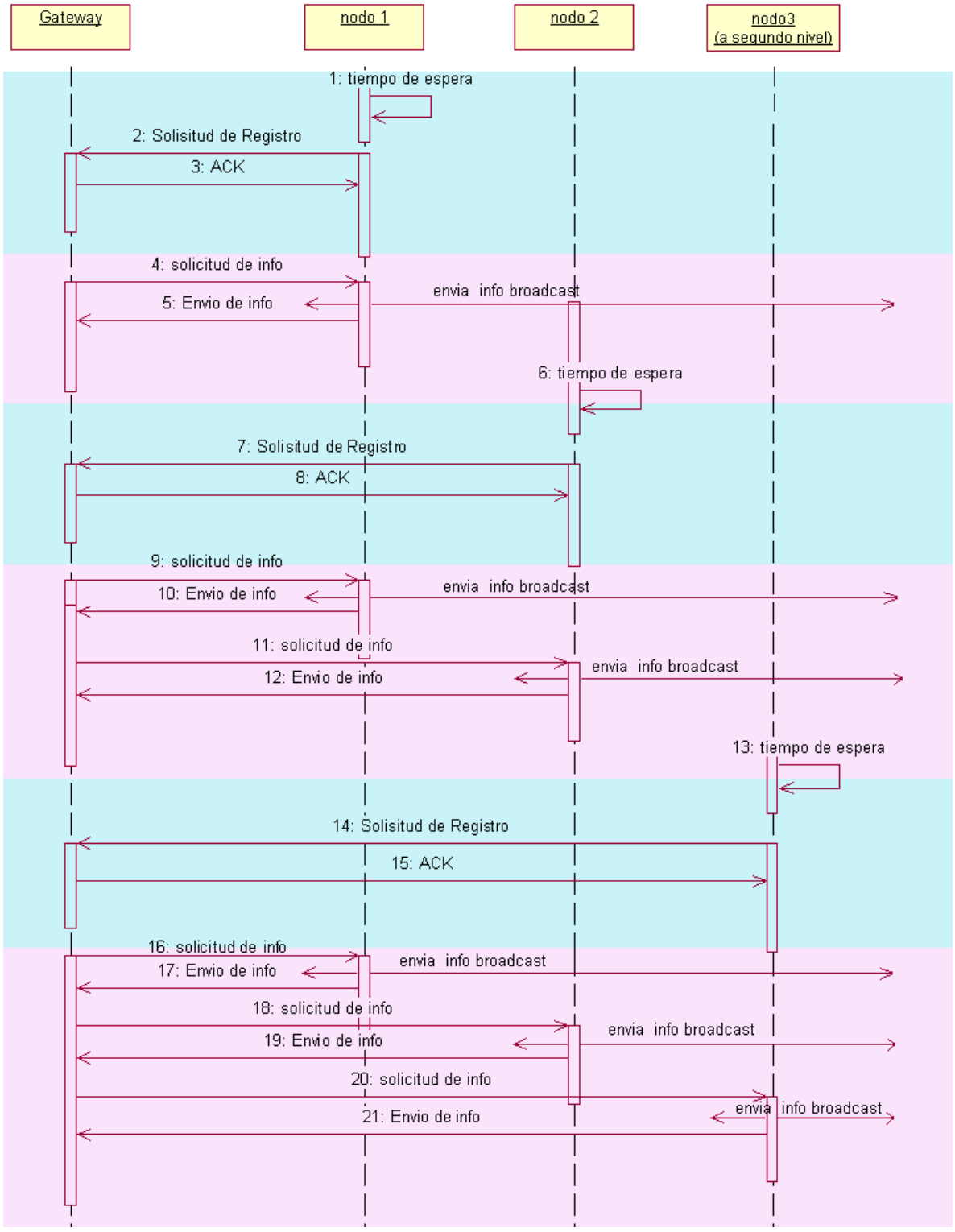

Figura 2.10 Diagrama de secuencia entre la Gateway y los Nodos.

En la Figura 2.11 se muestra el diagrama de secuencia de la comunicación entre la Gateway y el PC, el cual se explica a continuación.

- Al iniciar el programa en el PC, este le envía a la Gateway una lista de nodos conectados (en ese momento se encuentra vacía).

- La Gateway recibe el mensaje del PC y devuelve inmediatamente un mensaje de cero nodos conectados, el proceso siguiente es calcular la posición de los nodos, como no se tienen nodos registrados, la aplicación pasa a un estado inactivo o de tiempo libre, este tiempo es mínimo de 150 ms cuando se tienen 10 nodos activos, en cualquier otro caso es el tiempo restante para iniciar el siguiente ciclo.

- En el siguiente ciclo, el PC envía la lista de nodos que todavía permanece vacía a la Gateway.

- La Gateway, que ya tiene registrado un nodo nuevo, le solicita su información y posteriormente la envía al PC informando en el mismo mensaje, que no hay más nodos.

- El PC realiza la ubicación del nodo y la guarda en la base de datos, luego hay un tiempo libre hasta iniciar el siguiente ciclo y vuelve a enviar la lista de nodos conectados a la Gateway.

- La Gateway recibe la lista (que en este momento tiene un nodo) y hace la petición de los datos del nodo que se encuentra en la lista más los nodos nuevos que ella recibió en el periodo de registro.

- Al terminar de hacer la recolección de los datos, estos son enviados al PC uno a uno indicando a que nodo pertenecen.

- El PC recibe los datos, calcula la posición de cada nodo, actualiza la base de datos, espera un tiempo inactivo hasta que inicie el siguiente ciclo y así lo continúa haciendo de manera periódica.

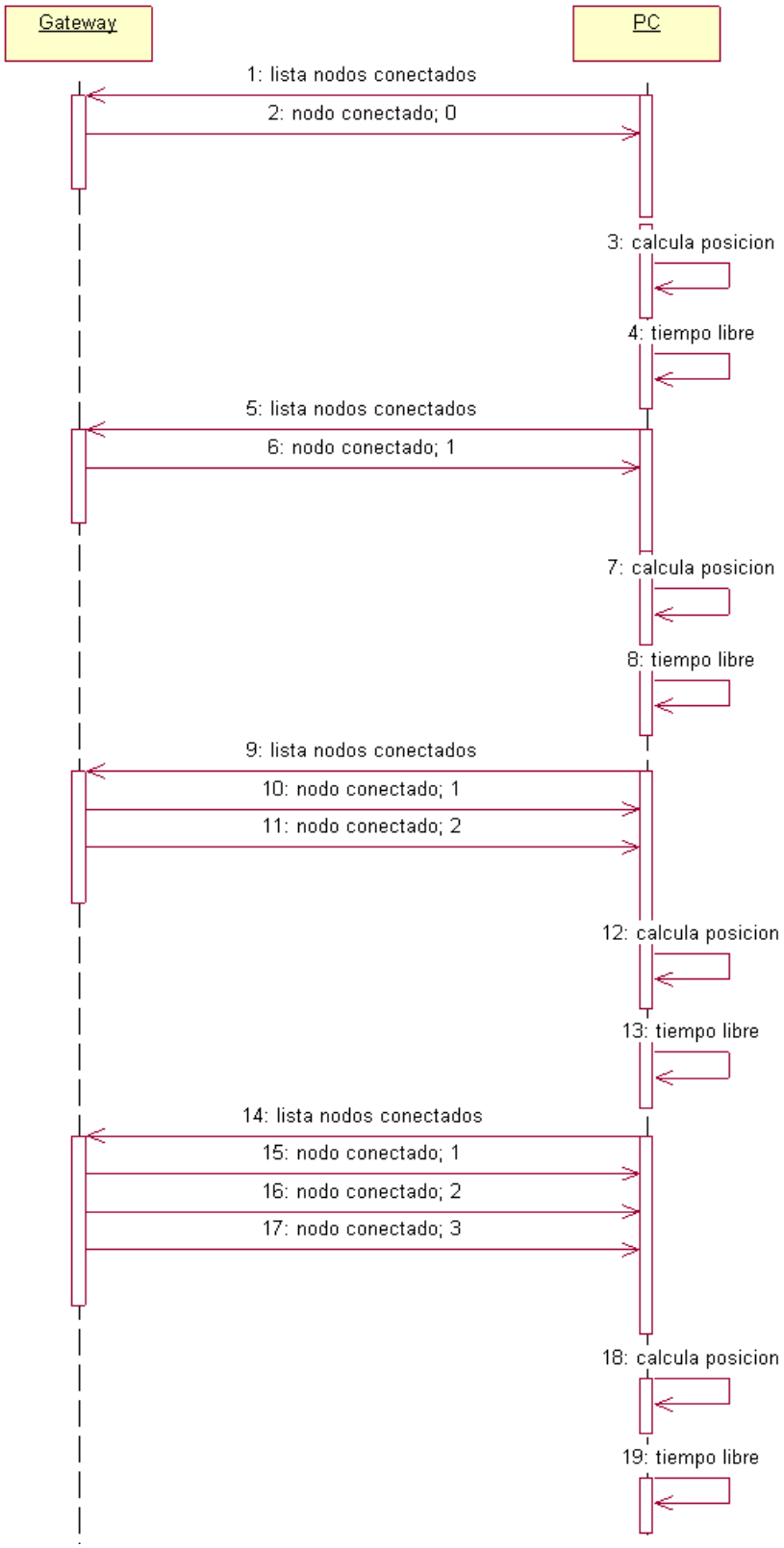

Figura 2.11 Diagrama de secuencia entre la Gateway y el PC.

2.7. Diagramas de componentes.

2.7.1. Nodo

En la figura 3.12 se muestra el diagrama de componentes del programa contenido en los nodos.

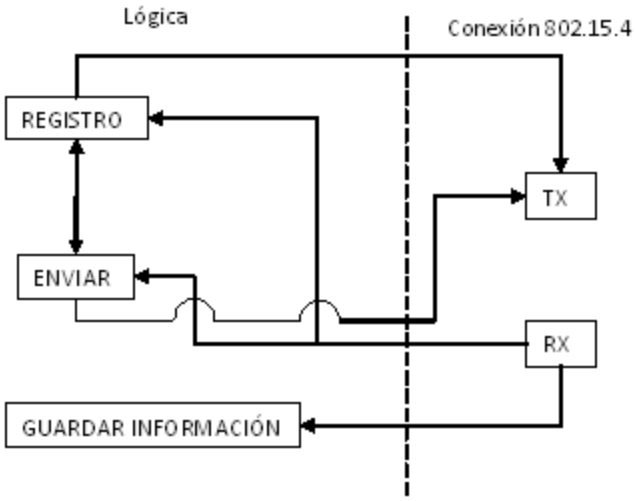

Figura 2.12 Diagrama de componentes del Nodo.

- **Registro:** Espera que el canal este libre para solicitar el registro a la Gateway. Si en algún momento la Gateway es desconectada, su función es intentar registrarse de nuevo de manera periódica.
- **Enviar:** Cuando el nodo recibe la autorización para enviar su información, primero transmite un mensaje tipo *broadcast* a sus vecinos y luego su información a la Gateway usando el protocolo Colector.
- **Guardar información:** Almacena el *id* y el *RSSI* cuando se recibe un mensaje tipo broadcast.
- **TX (802.15.4):** Módulo que permite transmitir de manera inalámbrica.
- **RX (802.15.4):** Módulo que permite recibir mensajes de manera inalámbrica.

2.7.2. Gateway

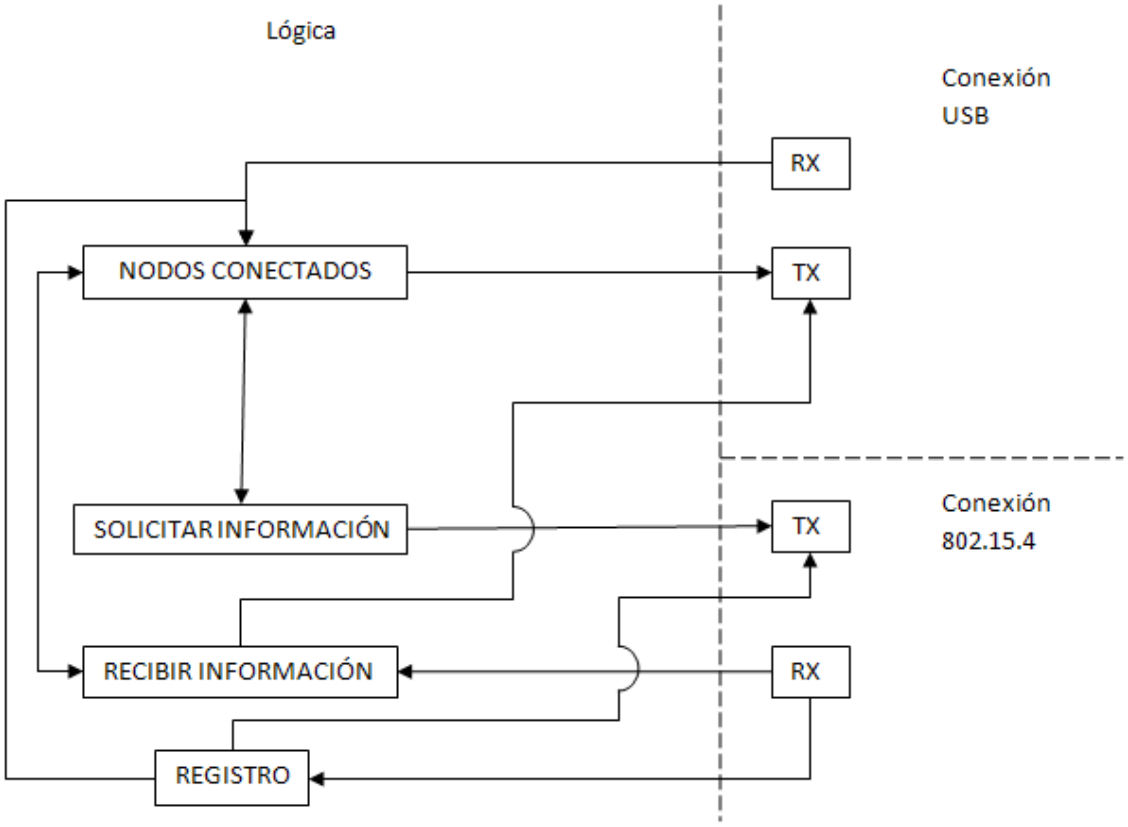

Figura 2.13 Diagrama de componentes de la Gateway.

- **Registro: Recibe** una petición de registro por medio de un mensaje usando el protocolo Colector, el devuelve una confirmación usando el protocolo Diseminador.
- **Nodos Conectados: Recibe** por USB la lista de los nodos conectados a la red en el orden en el que se les debe solicitar la información. A esta lista se le añaden los nodos nuevos al final.
- **Solicitar información:** Envía un mensaje usando el protocolo Diseminador con el *id* del nodo que debe enviar su información, si después de un tiempo predeterminado no recibe respuesta del nodo, se da por hecho que fue apagado o que esta fuera de cobertura, en ese caso se envía un mensaje al PC informando la situación.
- **Recibir información: Almacena** temporalmente los datos provenientes de un nodo y los retransmite por USB hacia el PC.
- **TX (USB):** Módulo que permite transmitir por la interfaz USB.
- **RX (USB):** Módulo que permite recibir mensajes por la interfaz USB.
- **TX (802.15.4):** Módulo que permite transmitir de manera inalámbrica.
- **RX (802.15.4):** Módulo que permite recibir mensajes de manera inalámbrica.

2.7.3. Aplicación en el PC.

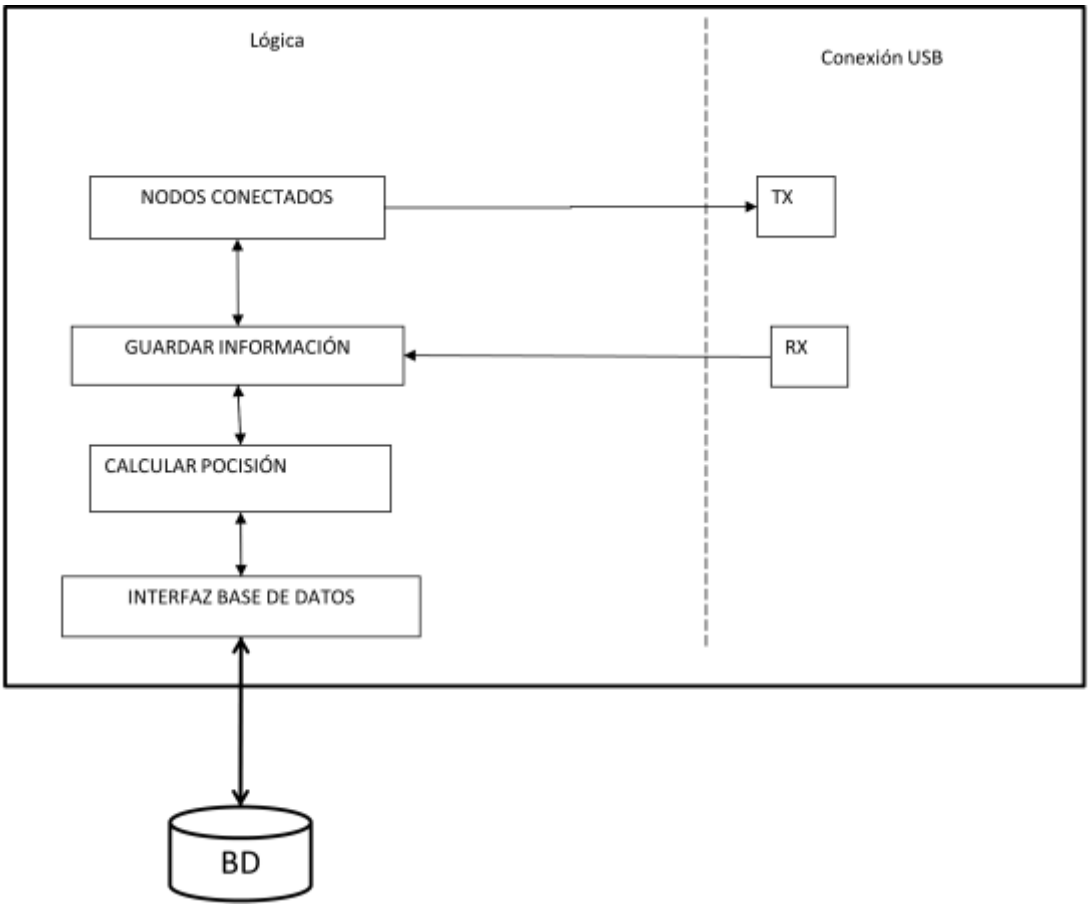

Figura 2.14 Diagrama de componentes de la aplicación en el PC.

- **Nodos conectados: Envía** por USB la lista de los nodos conectados en la red en el orden en que deben enviar la información.
- **Guardar información: Almacena** los datos de cada nodo para su posterior análisis.
- **Calcular posición: Realiza** los cálculos matemáticos necesarios para estimar distancias entre nodos y sus coordenadas.
- **Interfaz Base de Datos: Almacena** en la Base de Datos de *MySQL* las coordenadas de los nodos conectados a la red y actualiza aquellos que se apagaron o salieron de cobertura.
- **TX (USB):** Módulo que permite transmitir por la interfaz USB.
- **RX (USB):** Módulo que permite recibir mensajes por la interfaz USB.

2.7.4. Interfaz Gráfica.

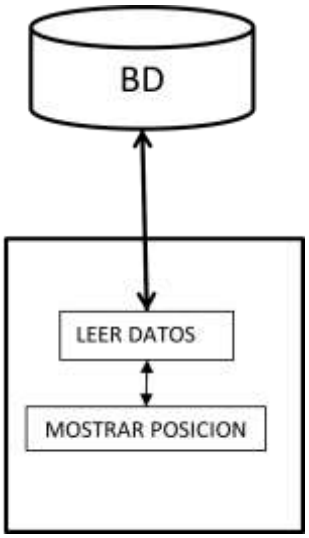

Figura 2.15 Diagrama de componentes de la interfaz grafica en el PC.

- **Leer datos: Accede** a la base de datos para leer la posición de los nodos de la red.
- **Mostrar posición: Despliega** puntos que representan a cada nodo sobre un plano bidimensional.

2.8. Diagramas de flujo.

2.8.1. Nodo.

En el diagrama de flujo (Figura 2.16) pueden observarse tres eventos importantes para los cuales el programa actúa de de la siguiente manera.

- *Al iniciar el programa* se inician variables y se envía un mensaje de petición de registro a la Gateway, para lo cual se utiliza el protocolo Colector.
- *Al recibir un mensaje broadcast* el nodo guarda el nivel de RSSI y el *id* de el nodo que envió este mensaje (guarda las últimas dos comunicaciones).
- *Cuando el nodo recibe un mensaje utilizando el protocolo Diseminador,* examina el valor recibido y lo compara con su id para determinar si es para él, si son iguales, mira si su estado es “conectado” si no es así, este mensaje es la respuesta a su anterior petición de registro así que cambia su estado a “ conectado”, pero si al recibir el mensaje su estado es “conectado”, lo que hace es enviar un mensaje broadcast para sus vecinos y posteriormente envía a la Gateway los datos guardados.

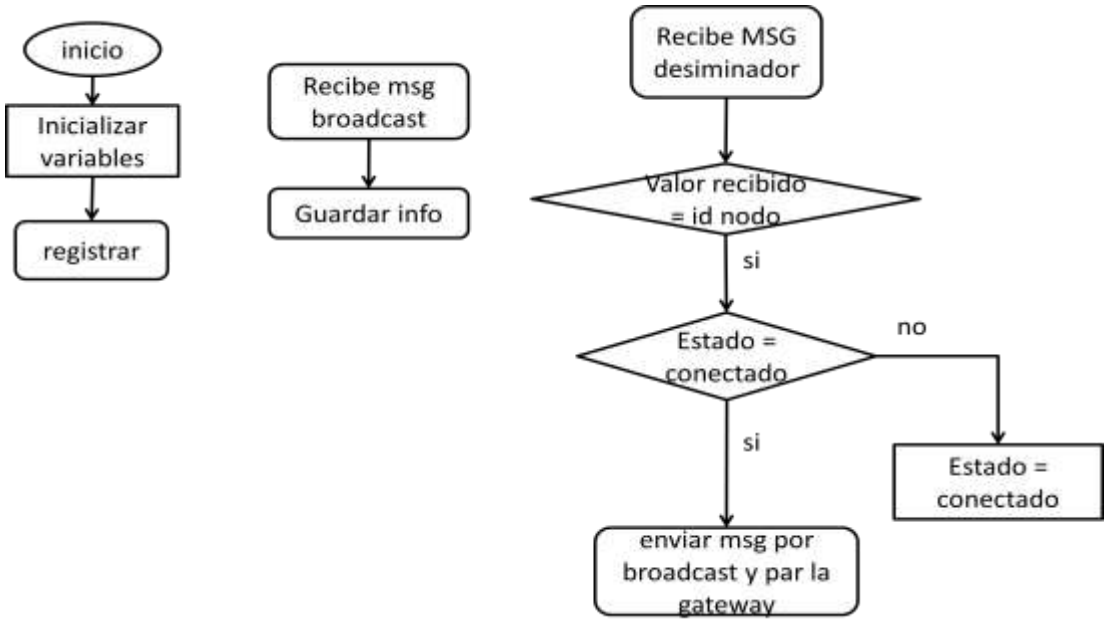

Figura 2.16 Diagrama de flujo de los nodos.

2.8.2. Gateway.

En el diagrama de flujo (Figura 2.17) pueden observarse tres eventos importantes para los cuales el programa actúa de la siguiente manera:

- *Al iniciar el programa* en la Gateway se inicializan variables y se queda a la espera de recibir la lista de nodos enviada desde el PC para iniciar su actividad con la red de sensores.
- *Al recibir un mensaje por el puerto USB* la Gateway guarda esta lista y si tiene nodos nuevos registrados los agrega. Si la lista queda vacía, se retorna un mensaje por USB informando que no hay nodos conectados. De lo contrario, si hay al menos un nodo activo, se inicia el proceso de solicitar la información y retransmitirla al PC con todos los nodos de la lista.
- *Si la Gateway recibe un paquete de datos* usando el protocolo Colector desde un nodo, guarda esta información para retransmitirla posteriormente y revisa si este es el último nodo en la lista, si lo es, envía los datos al PC informando que es el último mensaje, si no, envía los datos al PC y luego hace la petición de la información al siguiente nodo.

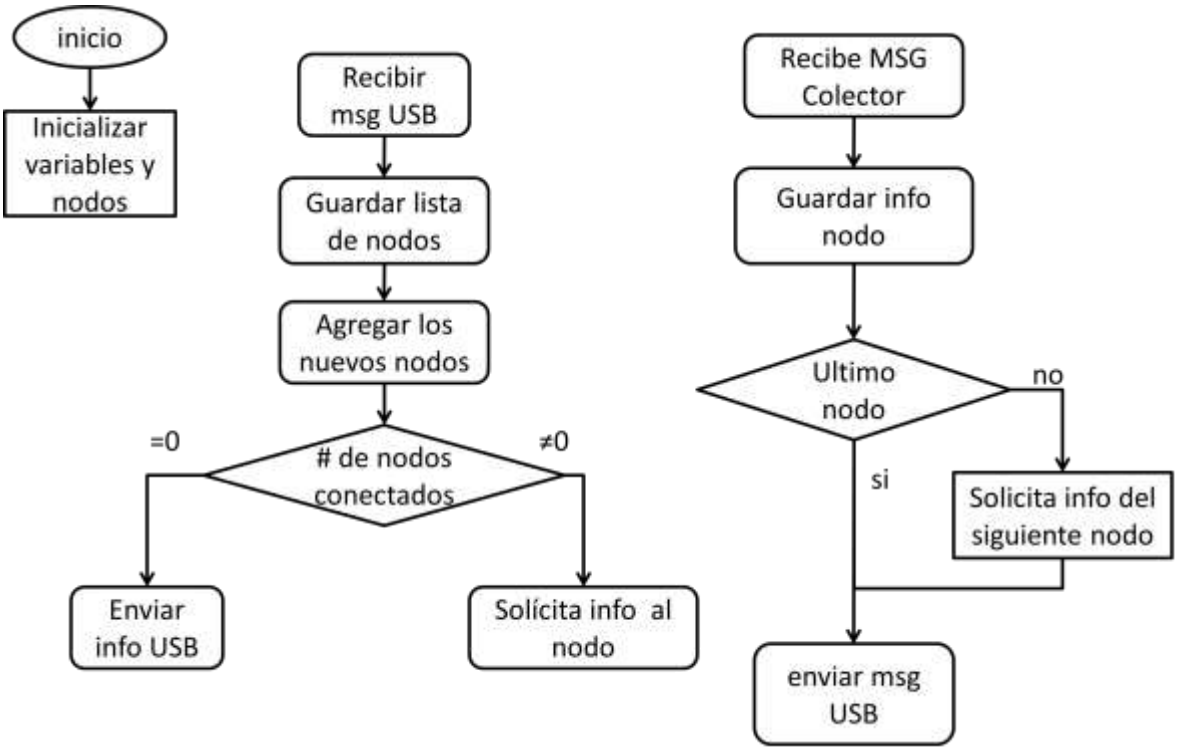

Figura 2.17 Diagrama de flujo de la Gateway.

2.8.3. Programa de Localización en el PC.

En la figura 2.18 se muestra el diagrama de flujo del programa de localización. Al iniciar la aplicación, este envía a la Gateway (por el puerto USB) la lista de los nodos registrados (al iniciar el programa esta lista estará vacía) y se queda a la espera de que la Gateway le responda enviándole los datos de cada nodo contenido en la lista más los que ella tenga registrados como nuevos, luego de recibir todos los datos, procede a calcular la posición de cada uno de los nodos, posteriormente hace una corrección de las posiciones para luego guardarlas en la base de datos. Este ciclo está seguido de un periodo de inactividad que depende del número de nodos activos en la red. Todo el ciclo tiene una duración de 500 milisegundos.

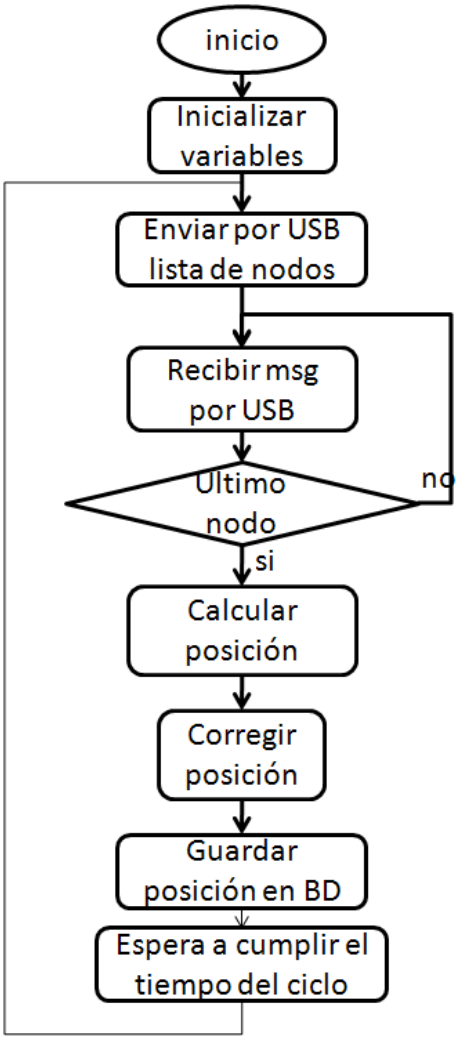

Figura 2.18 Diagrama de flujo de la Aplicación de localización en el PC.

2.8.4. Interfaz gráfica.

En la figura 2.19 se muestra el diagrama de flujo del programa que muestra mediante una interfaz grafica, la posición de los nodos activos en la *WSN*. Este programa consta de un hilo periódico (con periodo de 0.5 segundos), que empieza leyendo la base de datos donde se encuentran guardadas las coordenadas de los nodos, para posteriormente mostrar la posición de cada nodo y volver a repetir el ciclo.

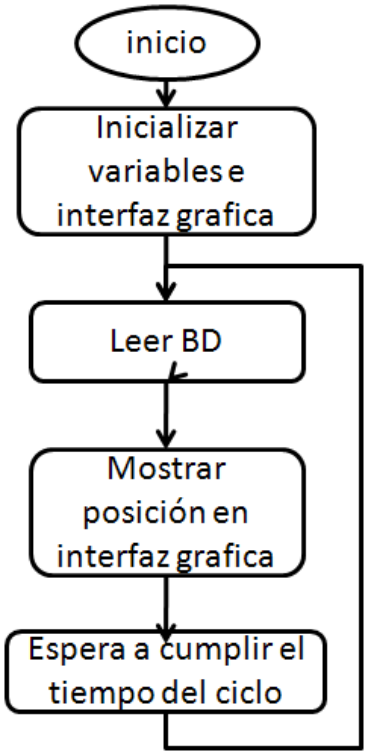

Figura 2.19 Diagrama de flujo de la Interfaz Grafica en el PC.

En este capítulo se hizo una descripción de la base matemática desarrollada para localizar los nodos de la red de sensores inalámbricos, donde no se requieren de nodos estáticos para establecer la distribución geográfica de la red, de igual manera se presentaron los diferentes diagramas que describen el comportamiento de cada uno de los componentes del sistema.

2.9. Estructura de Tramas del Sistema.

- Tramas Zigbee que se intercambian entre los nodos y la Gateway:

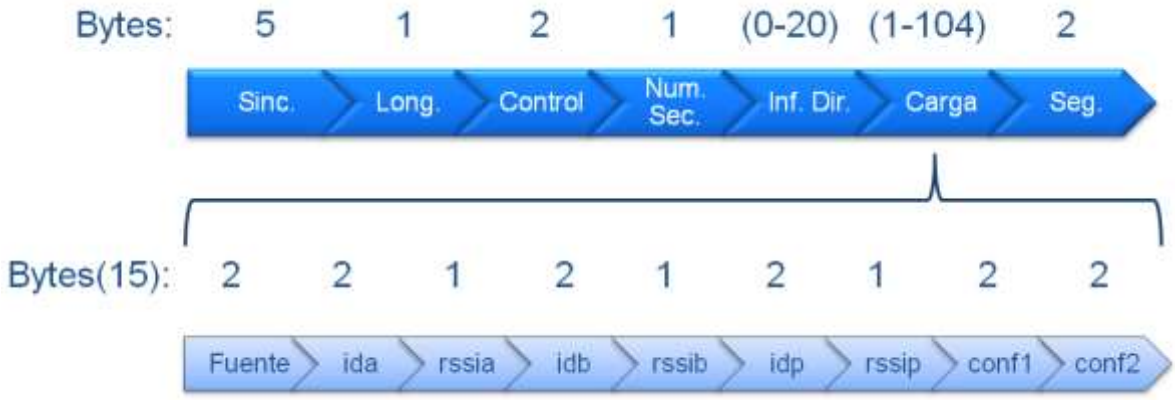

Figura 2.20 Trama Zigbee entre Nodos y Gateway.

En la parte superior de la Figura 2.20 esta la trama de datos del protocolo Zigbee, donde el sistema utiliza el campo de carga para enviar los datos desde los nodos hacia la Gateway, estos datos representan:

- Fuente: identificador del nodo origen.
- ida: identificador del nodo de referencia A.
- rssia: potencia del mensaje recibido por el nodo A.
- idb: identificador del nodo de referencia B.
- rssib: potencia del mensaje recibido por el nodo B.
- idp: identificador del nodo que sirve como puente en caso de ser necesario.
- rssip: potencia del mensaje recibido por el nodo puente.
- conf1: campo que contiene el valor consignado en el campo Fuente.
- conf2: contiene el valor del campo Fuente multiplicado por dos.

Los campos conf1 y conf2 permiten realizar una validación de cada paquete para evitar filtración de datos de otras redes inalámbricas.

- Tramas que se intercambian entre la Gateway y el PC por USB:

La trama tiene la estructura de la figura 2.21.

Figura 2.21 Trama entre PC y Gateway.

El campo Carga cundo se envian datos del PC a la Gateway contiene (Figura 2.22):

Figura 2.22 Trama del PC hacia la Gateway.

Donde cada campo contiene el id de un nodo registrado en el PC.

Cundo la Gateway trasmite la información de un nodo hacia el PC el campo Carga contiene (Figura 2.23):

Figura 2.23 Trama de la Gateway al PC.

Estos datos corresponden a los que envía el nodo hacia la Gateway a excepción que los campos conf1 y conf2 están ausentes porque no se requieren y se agrega el campo Prox para informarle al PC cuando es la última trama que se envía.

Capitulo 3

PRUEBAS

Este capítulo describe las pruebas y resultados obtenidos en el proceso de desarrollo y depuración de cada uno de los componentes del sistema.

3.1. Pruebas de adaptación del sistema.

Las pruebas se comenzaron a realizar desde el inicio del proyecto, inicialmente para conocer las propiedades del sistema operativo TinyOS. Las primeras pruebas se hicieron utilizando TinyOS versión 2.0, pero en Agosto de 2008 se liberó la versión 2.1 con la cual se desarrollaron los componentes del sistema implementado, este cambio significó la adaptación del código y reemplazar algunas interfaces ya establecidas.

Posteriormente se identificó la necesidad de tener la aplicación en el PC corriendo en tiempo real, para garantizar tiempos de respuesta que no afectaran la función de la Gateway. Como se mencionó en la sección 1.2.3, las primeras opciones fueron *RTLinux* y *RTAI*, sistemas basados en Linux que garantizan funciones de tiempo real duro o critico, pero que lamentablemente no poseen los drivers para interactuar con puertos seriales mapeados desde un USB. La siguiente opción y con resultados favorables fue *JavaRT,* las pruebas mostraron fácil adaptación al sistema, por lo cual la aplicación en el PC se desarrollo con este lenguaje.

Con los módulos lógicos (*software*) funcionando en los nodos sobre TinyOS y la aplicación del PC sobre *JavaRT*, se iniciaron las pruebas para determinar las constantes de pérdidas y ganancias de las antenas y demás dispositivos en los nodos que intervienen en la conexión inalámbrica (Ecuación 1, sección 3.3). Estos datos (Tabla 1) se tomaron en un ambiente libre de interferencia y obstáculos. El nodo y la Gateway se encuentran a la misma altura de 1 metro sobre el piso y se separan hasta 20 cm manteniendo la misma dirección. La distancia es estimada usando la ecuación de pérdidas de espacio libre para comunicaciones inalámbricas, por esto la naturaleza logarítmica de los datos obtenidos.

Distancia (cm)	RSSI
2	27
4	25
6	24
8	23
10	22
15	21
20	20

Tabla 3. Valores obtenidos de RSSI.

En esta prueba se encontró la inestabilidad de los datos recogidos debido a la naturaleza de las comunicaciones inalámbricas. El sistema desarrollado utiliza la potencia recibida o RSSI (*Received Signal Strength Indication*) en una comunicación, por ende todos los fenómenos electromagnéticos y propios de los dispositivos electrónicos, generan situaciones adversas para una comunicación, los valores obtenidos cambian en el orden de 1 dB sin razón aparente. Esta inestabilidad provocó una adaptación en el algoritmo para evitar saltos continuos en la posición de los nodos por la variación de la señal, por este motivo la sensibilidad del sistema se ve afectada y solo los cambios de posición que representen el incremento o decremento de más de 1 dB en el RSSI, serán percibidos por el sistema y por consiguiente la ubicación del nodo cambiará.

Una vez superado este percance, las pruebas mostraban errores significativos en el cálculo de la posición de los nodos, en un examen minucioso se encontró que las antenas que deberían radiar de manera isotrópica lo hacen con un patrón de radiación por lóbulos, este efecto puede ser causado por la antena en sí misma o por la placa del impreso (ver Figura 3.1).

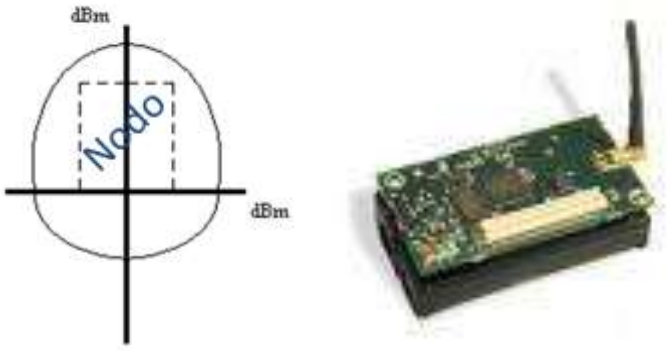

Figura 3.1 Patrón de radiación de un Nodo Iris obtenido por las pruebas.

Este problema representó un alto en el camino, las pruebas mostraban errores inaceptables en el cálculo. La estrategia a seguir, fue buscar una solución matemática, que permitiera mantener el esquema implantado y luego de un análisis, se creó un sistema de autocorrección de la posición, este nuevo componente conlleva un costo en tiempo de procesamiento del algoritmo, pero mejora considerablemente la estimación de la posición.

Un problema detectado en algunas pruebas, es causado por las interferencias producidas por otros sistemas, en el caso de las WSN se genera por operar en la banda libre de 2.4GHz, esta frecuencia es utilizada por Bluetooth y Wi-Fi principalmente. El uso del espectro se puede observar en la Figura 3.2.

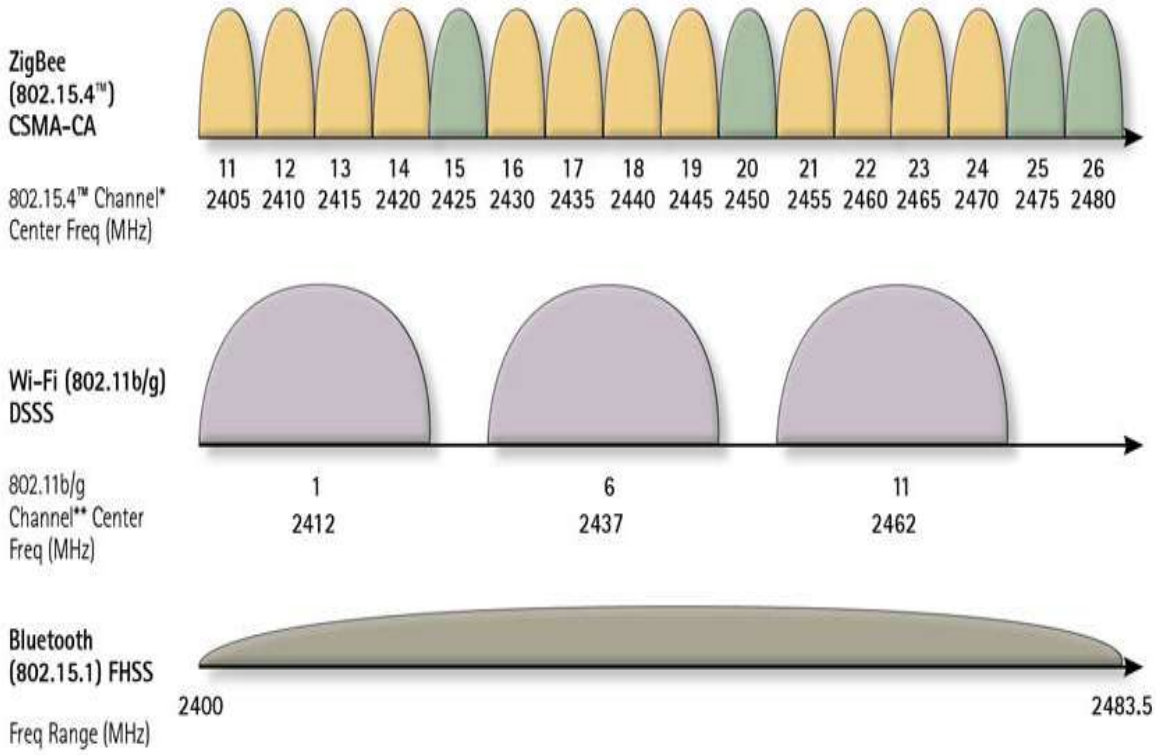

Figura 3.2 Espectro electromagnético en la banda 2.4 GHz.

Esta interferencia provoca errores en la transmisión de datos y obliga al reenvío de la información, esa labor la ejecuta TinyOS pero representa un gasto adicional de energía, en nuestra aplicación esto acarrea un error secundario en el indicador de potencia de la señal recibida, el cual utilizamos para determinar la distancia entre los nodos, el problema es inevitable, porque el transceptor del nodo IRIS es el *RF230* de Atmel, el cual no discrimina la naturaleza del paquete recibido para tomar el RSSI y lo hace usando su hardware, esto desemboca en la mala toma de datos y por consiguiente el mal cálculo de la posición. En algunos casos, mensajes provenientes de otra red fueron tomados por los nodos y la Gateway, lo cual generó errores del registro de los nodos activos, con identificadores fuera del rango previsto, este problema bloqueo el sistema.

La solución principal es evitar entornos contaminados con estos sistemas que utilizan la misma banda, sin embargo, para ofrecer mayor seguridad en los mensajes de la red, se añadió una firma propia de cada nodo en los mensajes que este envíe. Consiste en dos campos adicionales en la estructura de datos que se envía, el primero es igual al identificador del nodo transmisor, el segundo corresponde al identificador multiplicado por dos, de esta manera la probabilidad de que un mensaje contenga el mismo tamaño de la trama, sus encabezados y estructura interna similares a las propias de la red, se disminuyen sustancialmente. En las pruebas sin este sistema adicional, un mensaje corrupto se percibía cada 2 minutos en promedio, y con el nuevo mecanismo no se han presentado problemas en las pruebas realizadas.

3.2. Prueba en entorno abierto.

Esta prueba se realizó a campo abierto, los nodos y la Gateway se encuentran a la misma altura de un metro sobre el piso, no hay obstáculos ni sistemas de comunicación inalámbrica en el perímetro. Se tomaron imágenes de la interfaz grafica y del registro en consola de la aplicación en tiempo real, a medida que se conectaron los nodos a la red. Las dos primeras imágenes corresponden a la interfaz sin nodos conectados (LocMapa, Figura 3.3) y al arranque de la aplicación en el entorno de desarrollo *Netbeans* (LocPC, Figura 3.4).

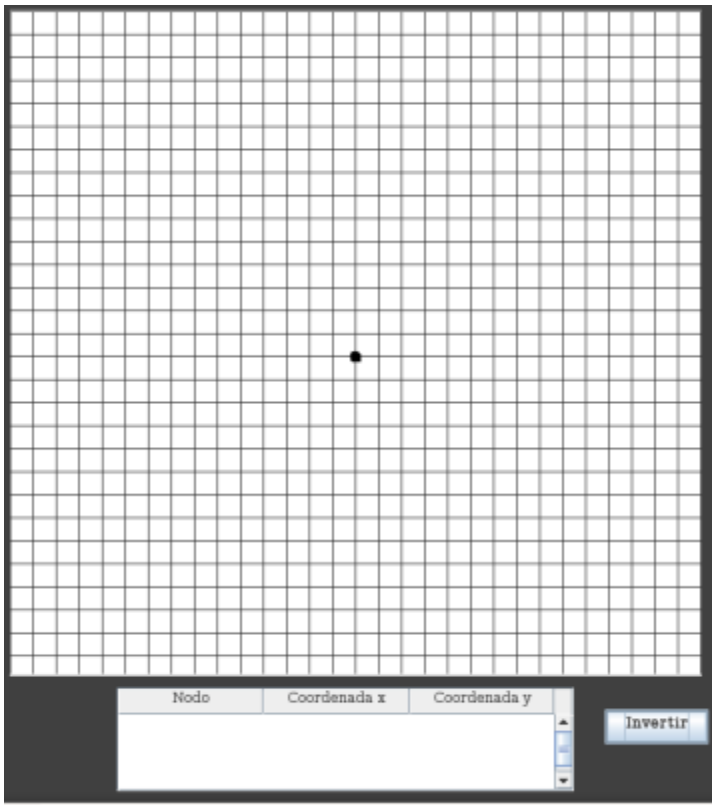

Figura 3.3 LocMap con 0 Nodos conectados.

```
init:
deps-jar:
compile:
run:
Registro en cero
Registro en cero
Registro en cero
Registro en cero
Registro en cero
Registro en cero
Registro en cero
serial@/dev/ttyUSB1:57600: resynchronising
00 FF FF 00 00 18 00 89 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 Hacia gateway: paquete enviado
00 FF FF 00 00 0D 00 89 00 00 00 00 00 00 00 00 00 00 00 00 00 Desde gateway: paquete leido
Numero de nodos conectados: 0
Orden: 0 , 0 , 0 , 0
00 FF FF 00 00 18 00 89 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 Hacia gateway: paquete enviado
00 FF FF 00 00 0D 00 89 00 00 00 00 00 00 00 00 00 00 00 00 00 Desde gateway: paquete leido
Numero de nodos conectados: 0
Orden: 0 , 0 , 0 , 0
00 FF FF 00 00 18 00 89 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 Hacia gateway: paquete enviado
00 FF FF 00 00 0D 00 89 00 00 00 00 00 00 00 00 00 00 00 00 00 Desde gateway: paquete leido
Numero de nodos conectados: 0
Orden: 0 , 0 , 0 , 0
```

Figura 3.4 LocPC con 0 Nodos conectados.

La aplicación inicialmente accede a la base de datos para poner en cero las coordenadas de todos los nodos, el siguiente proceso es abrir el puerto USB y sincronizarse con la Gateway, esta comunicación permanece siempre abierta mientras la aplicación este corriendo. Posteriormente se envía una lista vacía de nodos a la Gateway, esta, responde informando que no hay nodos en la red. Este proceso se repite a los 500 milisegundos. Los primeros 8 pares de hexadecimales de los paquetes transmitidos hacia y desde la Gateway, corresponden al encabezado predeterminado que indica que es un mensaje tipo *Broadcast*, y el tamaño total del mensaje.

La Figura 3.5 muestra el primer nodo conectado en la interfaz gráfica.

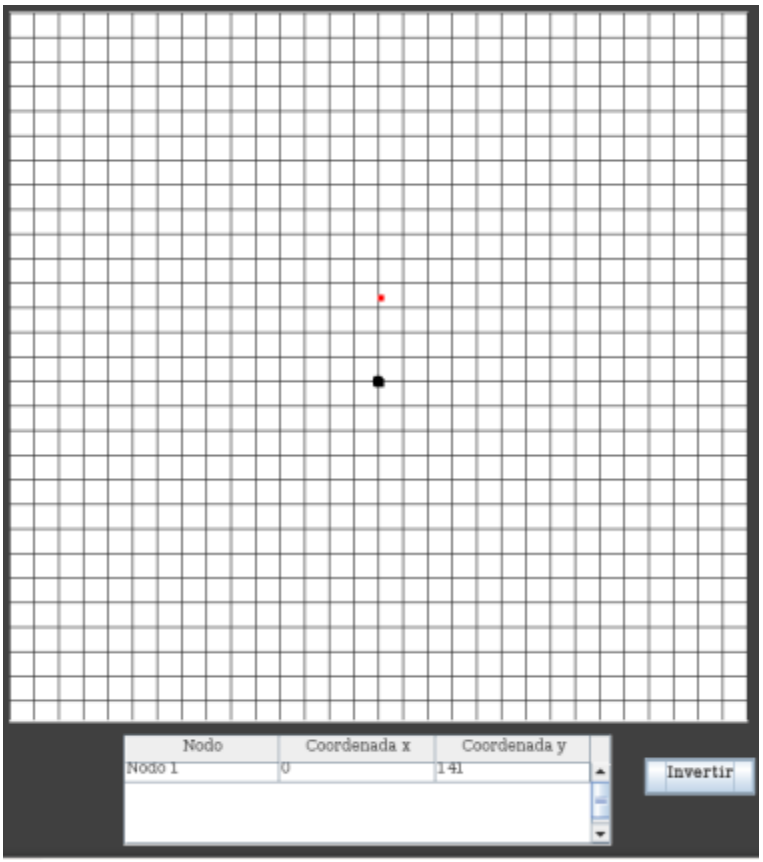

Figura 3.5 LocMap con 1 Nodo conectado.

El proceso de registro y cálculo en la aplicación, se observa en la figura 3.6.

```
00 FF FF 00 00 18 00 89 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 Hacia gateway: paquete enviado
00 FF FF 00 00 0D 00 89 00 01 11 00 00 00 00 00 00 00 00 00 00 Desde gateway: paquete leido
Fuente: 1 RSSI: 17
Numero de nodos conectados: 1
Orden: 1, 0, 0, 0
Coordenadas= x1:0.0 y1:141.0
Coordenadas= x1:0.0 y1:141.0 x2:0.0 y2:0.0
Coordenadas= x1:0.0 y1:141.0 x2:0.0 y2:0.0 x3:0 y3:0
Registro modificado
00 FF FF 00 00 18 00 89 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 Hacia gateway: paquete enviado
00 FF FF 00 00 0D 00 89 00 01 11 00 00 00 00 00 00 00 00 00 00 Desde gateway: paquete leido
Fuente: 1 RSSI: 17
Numero de nodos conectados: 1
Orden: 1, 0, 0, 0
Coordenadas= x1:0.0 y1:141.0
Coordenadas= x1:0.0 y1:141.0 x2:0.0 y2:0.0
Coordenadas= x1:0.0 y1:141.0 x2:0.0 y2:0.0 x3:0 y3:0
Registro modificado
```

Figura 3.6 LocPC con 1 Nodo conectado.

La Gateway retorna al PC un mensaje con información del primer nodo conectado, en este caso el identificador es el numero 1. El RSSI percibido por la Gateway fue de 17 dB, con este valor se procede a hacer los cálculos correspondientes, la distancia aproximada es de 1,41 metros. La exactitud de esta distancia depende de la dirección del nodo. Las nuevas coordenadas son almacenadas en la base de datos para que la interfaz gráfica pueda acceder a esta información.

En este tipo de entorno se alcanzó una distancia máxima de 75 metros aproximadamente, un valor inferior al especificado en el Datasheet en un ambiente ideal (300 metros).

Se conecta un segundo nodo.

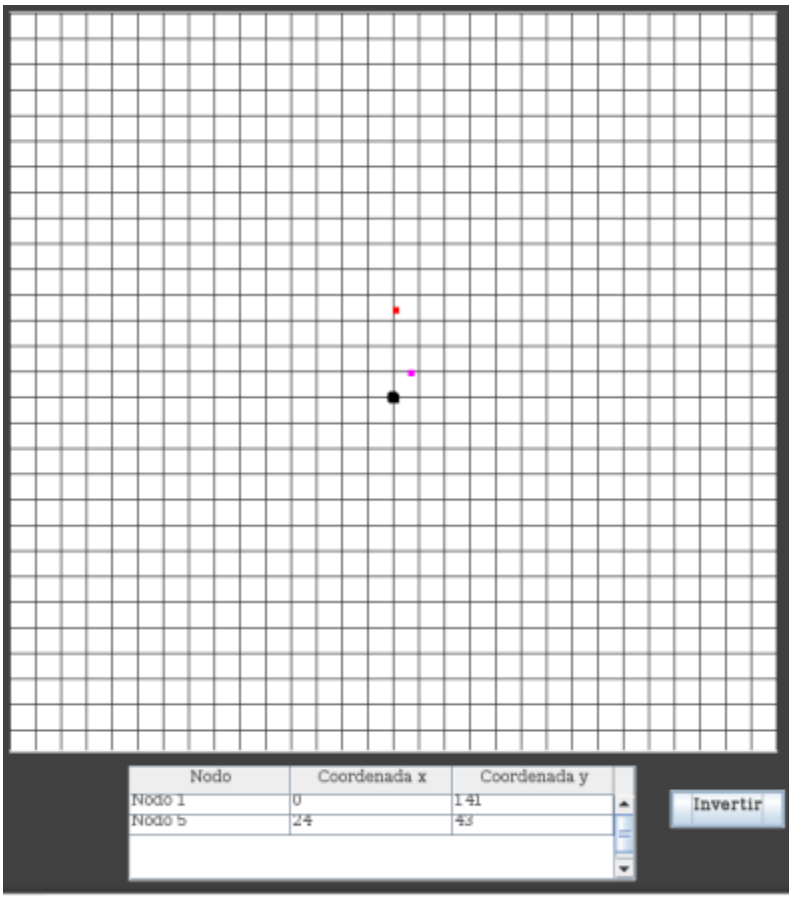

Figura 3.7 LocMap con 2 Nodos conectados.

Este segundo nodo aparece en el eje positivo X por defecto, pero es factible rotar el plano 180 grados con respecto al eje Y haciendo clic en el botón Invertir (Figura 3.8). La interfaz multiplica las coordenadas de X de todos los nodos. Esta operación no afecta los datos de la base de datos ni interfiere con el funcionamiento de la aplicación en tiempo real.

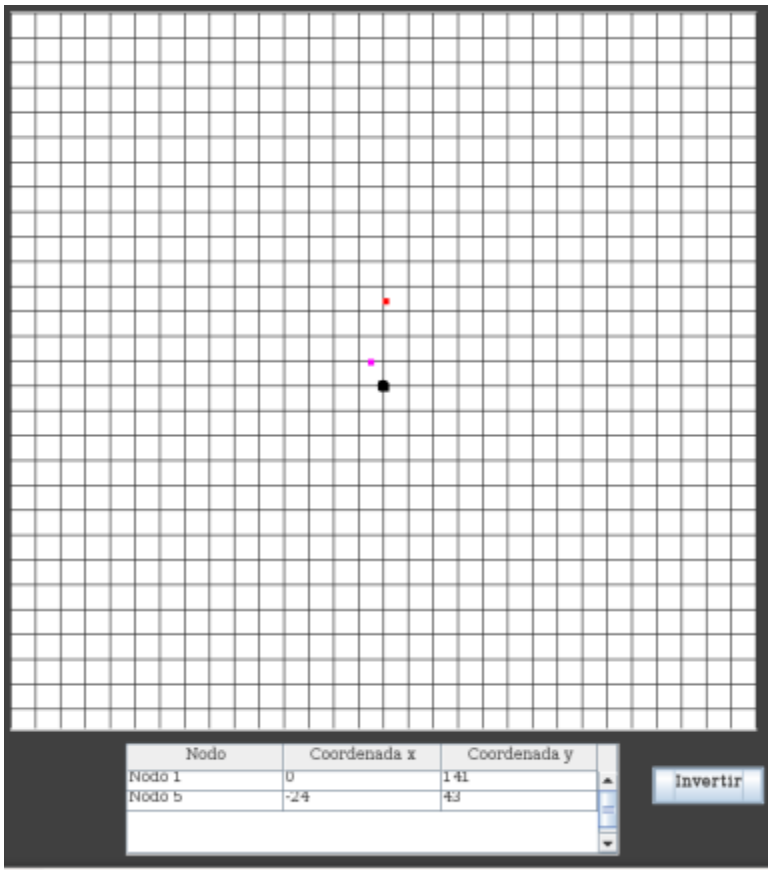

Figura 3.8 LocMap con 2 Nodos conectados y plano invertido.

El proceso de registro y cálculo de la posición del segundo nodo se registra de la siguiente forma:

00 FF FF 00 00 18 00 89 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 Hacia gateway: paquete enviado
00 FF FF 00 00 0D 00 89 00 01 10 00 00 00 00 00 00 00 00 00 01 Desde gateway: paquete leido
Fuente: 1 RSSI: 16
00 FF FF 00 00 0D 00 89 00 05 13 00 01 10 00 00 00 00 00 00 00 Desde gateway: paquete leido
Fuente: 5 RSSI: 19
Numero de nodos conectados: 2
Orden: 1, 5, 0, 0
Coordenadas= x1:0.0 y1:141.0
diferencia: -1.8188989403545856E-12 coordenadas x: 47.146652226020885 y: 52.81160460293359 distancias: l2:70.79457843841381 l21:100.0 aa:17.0 pp1:18.0
diferencia: 0.0 coordenadas x: 6.451537512308603 y: 70.5 distancias: l2:70.79457843841381 l21:70.79457843841381 aa:18.0 pp1:18.0
diferencia: 5147.849421045787 coordenadas x: 35.87421575673901 y: 78.36519824384094 distancias: l2:70.79457843841381 l21:50.11872336272725 aa:19.0 pp1:18.0
diferencia: 8047.850287174556 coordenadas x: 44.85490577717619 y: 83.80832242722893 distancias: l2:70.79457843841381 l21:35.48133892335753 4 aa:20.0 pp1:18.0
diferencia: 9560.705366893768 coordenadas x: 48.88942975453215 y: 86.03515954536358 distancias: l2:70.79457843841381 l21:25.11886431509579 5 aa:21.0 pp1:18.0
diferencia: 3.637978807091713E-12 coordenadas x: 24.09963964799849 2 y: 43.94640578549497 distancias: l2:50.11872336272725 l21:100.0 aa:17.0 pp1:19.0
diferencia: 5147.849421045785 coordenadas x: 35.87421575673901 y: 61.63480175615906 4 distancias: l2:50.11872336272725 l21:70.79457843841381 aa:18.0 pp1:19.0
diferencia: 9833.454273961 67 coordenadas x: 49.58188750431 3684 y: 70.5 distancias: l2:50.11872336272725 l21:50.11872336272725 aa:19.0 pp1:19.0
diferencia: 12418.341723428533 coordenadas x: 55.71880679678212 5 y: 74.94312418338801 distancias: l2:50.11872336272725 l21:35.48133892335753 4 aa:20.0 pp1:1
diferencia: 13773.266568307 5684 coordenadas x: 58.67977927505284 y: 77.16996130152265 distancias: l2:50.11872336272725 l21:25.11886431509579 5 aa:21.0 pp1:19.
diferencia: 1206.3 35382 1647893 coordenadas x: 17.3661695702073 86 y: 39.503281602 10697 distancias: l2:35.48133892335753 4 l21:100.0 aa:17.0 pp1:20.0
diferencia: 8047.850287174556 coordenadas x: 44.85490577717619 y: 67.19167757277107 distancias: l2:35.48133892335753 4 l21:70.79457843841381 aa:18.0 pp1:20.0
diferencia: 12418.341723428 63 coordenadas x: 55.71880679678210 4 y: 66.05687581661199 distancias: l2:35.48133892335753 4 l21:50.11872336272725 aa:19.0 pp1:20.
diferencia: 14845.298352823333 coordenadas x: 60.92064172516433 y: 70.5 distancias: l2:35.48133892335753 4 l21:35.48133892335753 4 aa:20.0 pp1:20.0
diferencia: 16121.069701654089 coordenadas x: 63.48438721 67 13214 y: 72.72683711813465 distancias: l2:35.48133892335753 4 l21:25.11886431509579 5 aa:21.0 pp1:20

Figura 3.9 LocPC con 2 Nodos conectados.

La lista que envía el PC contiene solo al nodo 1, la Gateway responde con la información de este nodo pero al final del mensaje indica con el valor 1h que va a enviar más información. El siguiente mensaje contiene los datos recibidos del nodo con identificador 5, este nodo almaceno el RSSI con valor 10h de la comunicación previa del nodo 1, con estos datos el algoritmo procede a simular la posición del segundo nodo. La exactitud de las coordenadas se observa en el campo *diferencia*, este valor representa la fidelidad de la posición, si es cero, indica un punto factible, de lo contrario significa que las coordenadas son matemáticamente incorrectas. El campo *l2* es la distancia del nodo a la Gateway y el campo *l21* es la distancia entre el nodo 1 y 5.

L a selección de las coordenadas se observa en la figura 3.10:

Figura 3.10 LocPC con 2 Nodos conectados, selección de posición.

El algoritmo toma los 5 valores más cercanos a cero, saca un promedio de las coordenadas en el eje *X* y selección la posición más cercana a este punto, en este caso escogió la combinación que se encuentra en el recuadro.

Se conecta un tercer nodo.

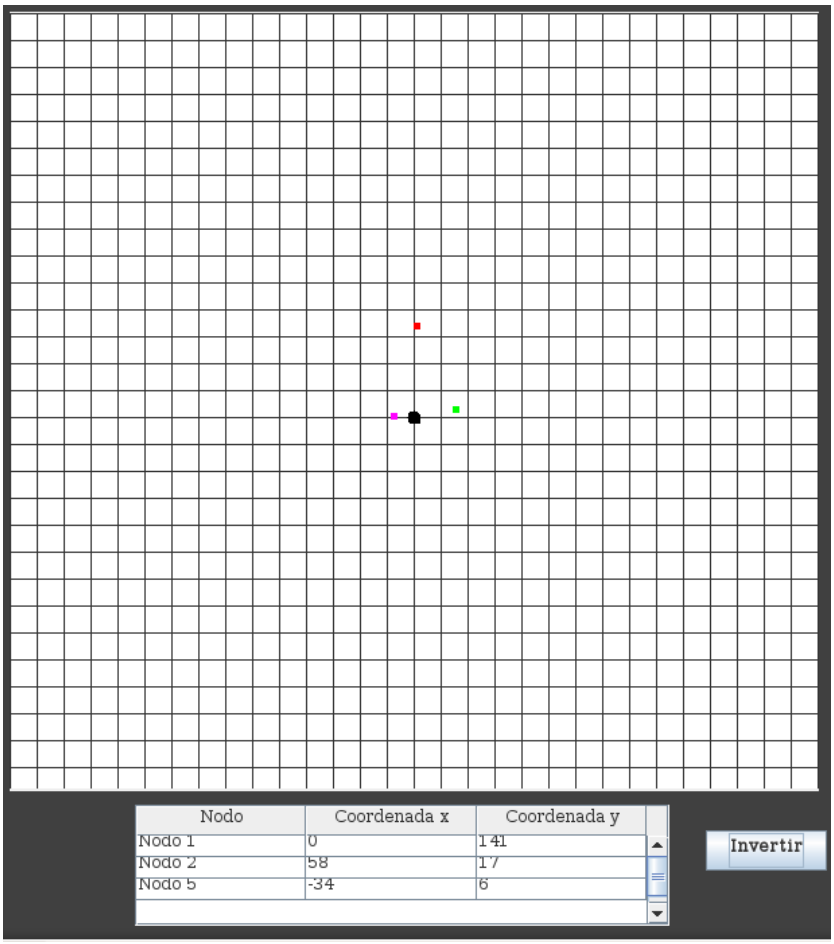

Figura 3.11 LocMap con 3 Nodos conectados.

El proceso de registro y cálculo de posición es similar al del segundo nodo pero utiliza otras ecuaciones (Figura 3.12).

```
00 FF FF 00 00 18 00 89 00 01 00 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 Hacia gateway: paquete enviado
00 FF FF 00 00 0D 00 89 00 01 11 00 02 0F 00 05 11 00 00 00 01 Desde gateway: paquete leido
Fuente: 1 RSSI: 17
00 FF FF 00 00 0D 00 89 00 05 12 00 01 11 00 02 12 00 00 00 01 Desde gateway: paquete leido
Fuente: 5 RSSI: 18
00 FF FF 00 00 0D 00 89 00 02 12 00 05 0F 00 01 0F 00 00 00 00 Desde gateway: paquete leido
Fuente: 2 RSSI: 18
Numero de nodos conectados: 3
Orden: 1, 5, 2, 0
Coordenadas= x1:0.0 y1:141.0
Coordenadas= x1:0.0 y1:141.0 x2:34.0 y2:6.0
control : x1: 34.0 y1: 6.0 x2: 0.0 y2: 141.0
Valores= l3:70.79457843841381 l31:100.0 l32:141.25375446227 54 y3:17.51861413682248
diferencia: -3701.2986373001963 coordenadas x:-58.91692696131095 y:17.51861413682248 distancias: l3:70.79457843841381 l31:100.0
Valores= l3:70.79457843841381 l31:100.0 l32:100.0 y3:52.8116040293359
diferencia: 6063.232357908663 coordenadas x:-65.14510164822508 y:52.8116040293359 distancias: l3:70.79457843841381 l31:100.0
Valores= l3:70.79457843841381 l31:100.0 l32:70.79457843841381 y3:70.5
diferencia: 13856.112173497166 coordenadas x:-68.26658329010698 y:70.5 distancias: l3:70.79457843841381 l31:100.0
Valores= l3:70.79457843841381 l31:100.0 l32:50.11872336272725 y3:79.36519824384094
diferencia: 18490.005336965918 coordenadas x:-69.83103003902009 y:79.36519824384094 distancias: l3:70.79457843841381 l31:100.0
Valores= l3:70.79457843841381 l31:70.79457843841381 l32:141.25375446227 54 y3:17.51861413682248
diferencia: -13489.553341448542 coordenadas x:14.437891622913 68 y:17.51861413682248 distancias: l3:70.79457843841381 l31:70.79457843841381
Valores= l3:70.79457843841381 l31:70.79457843841381 l32:100.0 y3:52.8116040293359
diferencia: -6466.222091856358 coordenadas x:8.209716935999548 y:52.8116040293359 distancias: l3:70.79457843841381 l31:70.79457843841381
Valores= l3:70.79457843841381 l31:70.79457843841381 l32:70.79457843841381 y3:70.5
```

Figura 3.12 LocPC con 3 Nodos conectados.

El PC envió la lista con 2 nodos y la Gateway devolvió esa información y adicionó la del nodo con identificador 2. Debido a que los dos primeros nodos no registraron cambios con respecto a su posición anterior, no se realizan los cálculos nuevamente. El campo l3 es la distancia del nodo 2 a la Gateway, l31 la distancia entre el nodo 2 y 5, l32 la distancia entre el nodo 2 y 1. La selección de la mejor posición se hace con las mismas características que para el segundo nodo (Figura 3.13).

```
Valores= l3:25.1188643 15095795 l31:70.79457843841381 l32:70.79457843841381 y3:54.964840454636416
diferencia: 16779.676277956063 coordenadas x:-56.59548643012013 y:54.964840454636416 distancias: l3:25.1188643 15095795 l31:70.79457843841381
Valores= l3:25.1188643 15095795 l31:70.79457843841381 l32:50.11872336272725 y3:63.83003869847735
diferencia: 20477.682969475 6 coordenadas x:-58.15993317903324 y:63.83003869847735 distancias: l3:25.1188643 15095795 l31:70.79457843841381
Valores= l3:25.1188643 15095795 l31:50.11872336272725 l32:141.25375446227 54 y3:1.9834545 91458896
diferencia: -1551.4948269354081 coordenadas x:-10.481331501866 12 y:1.9834545 91458896 distancias: l3:25.1188643 15095795 l31:50.11872336272725
Valores= l3:25.1188643 15095795 l31:50.11872336272725 l32:100.0 y3:37.27644448397232
diferencia: 3113.3506978780897 coordenadas x:-16.709506188780253 y:37.27644448397232 distancias: l3:25.1188643 15095795 l31:50.11872336272725
Valores= l3:25.1188643 15095795 l31:50.11872336272725 l32:70.79457843841381 y3:54.964840454636416
diferencia: 8350.333260189938 coordenadas x:-19.830987830662153 y:54.964840454636416 distancias: l3:25.1188643 15095795 l31:50.11872336272725
Valores= l3:25.1188643 15095795 l31:50.11872336272725 l32:50.11872336272725 y3:63.83003869847735
diferencia: 11703.243349853423 coordenadas x:-21.395434579575262 y:63.83003869847735 distancias: l3:25.1188643 15095795 l31:50.11872336272725
Valores= l3:25.1188643 15095795 l31:35.48133892335 7534 l32:141.25375446227 54 y3:1.9834545 91458896
diferencia: -1691.7213776047306 coordenadas x:7.94456584689 y:1.9834545 91458896 distancias: l3:25.1188643 15095795 l31:35.48133892335 7534
Valores= l3:25.1188643 15095795 l31:35.48133892335 7534 l32:100.0 y3:37.27644448397232
diferencia: 2284.5659025015625 coordenadas x:1.7163911599758654 y:37.27644448397232 distancias: l3:25.1188643 15095795 l31:35.48133892335 7534
Valores= l3:25.1188643 15095795 l31:35.48133892335 7534 l32:70.79457843841381 y3:54.964840454636416
diferencia: 7176.451862957355 coordenadas x:-1.4050904819060326 y:54.964840454636416 distancias: l3:25.1188643 15095795 l31:35.48133892335 7534
Valores= l3:25.1188643 15095795 l31:35.48133892335 7534 l32:50.11872336272725 y3:63.83003869847735
diferencia: 10356.403941402 43 coordenadas x:-2.9695372308191414 y:63.83003869847735 distancias: l3:25.1188643 15095795 l31:35.48133892335 7534
Coordenadas= x1:34.0 y1:6.0 x2:0.0 y2:141.0 x3:-58 y3:17
Registro modificado
Registro modificado
Registro modificado
```

Figura 3.13 LocPC con 3 Nodos conectados, selección de posición.

La combinación con las coordenadas seleccionada está en el recuadro de la Figura 3.12. Los nodos que se incorporen a la red después del tercer nodo, utilizan el mismo mecanismo y características para seleccionar su posición como se realizó con el tercer nodo.

3.3. Prueba en entorno con ruido.

Esta prueba se realizó en la oficina del SISE (Semillero de Investigación de Sistemas Empotrados) en la FIET. En este entorno se encuentran varios sistemas inalámbricos coexistiendo en la banda de 2.4 GHz, esto genera un nivel ruido con las características requeridas para medir el comportamiento del algoritmo de localización bajo condiciones adversas. Al igual que en la prueba de entorno abierto, los nodos y la Gateway se encuentran a una altura de 1 metro sobre el piso. Las siguientes gráficas presentan la secuencia de conexión de cuatro nodos.

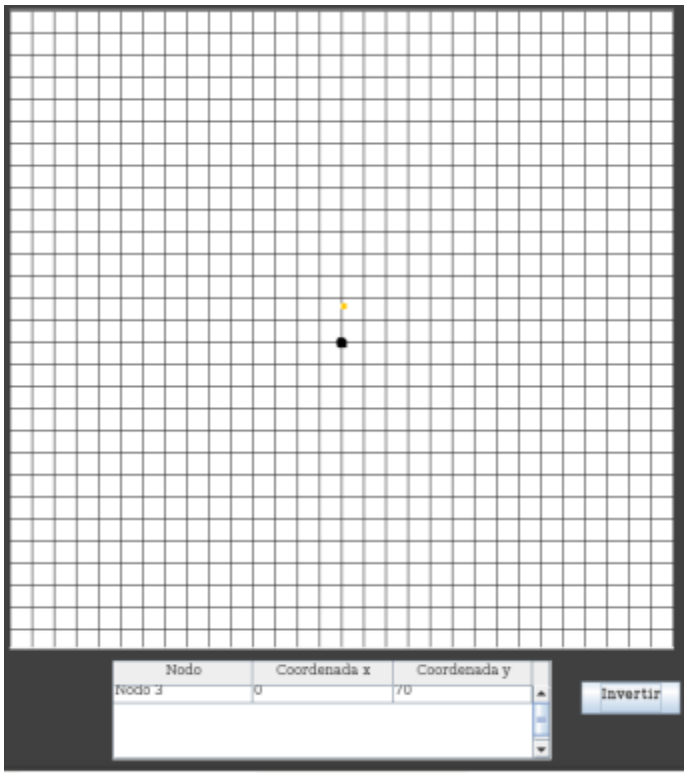

Figura 3.14 LocMap con 1 Nodo conectado, entorno de ruido.

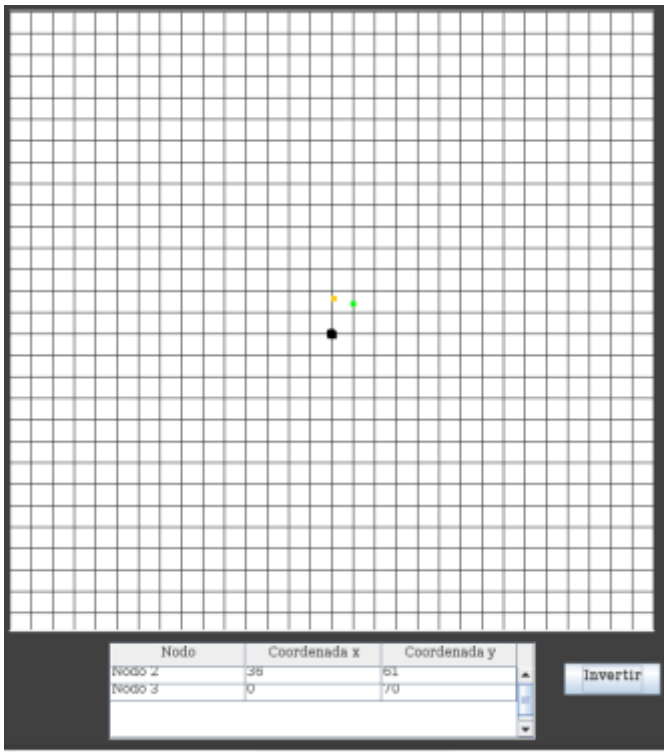

Figura 3.15 LocMap con 2 Nodos conectados, entorno de ruido.

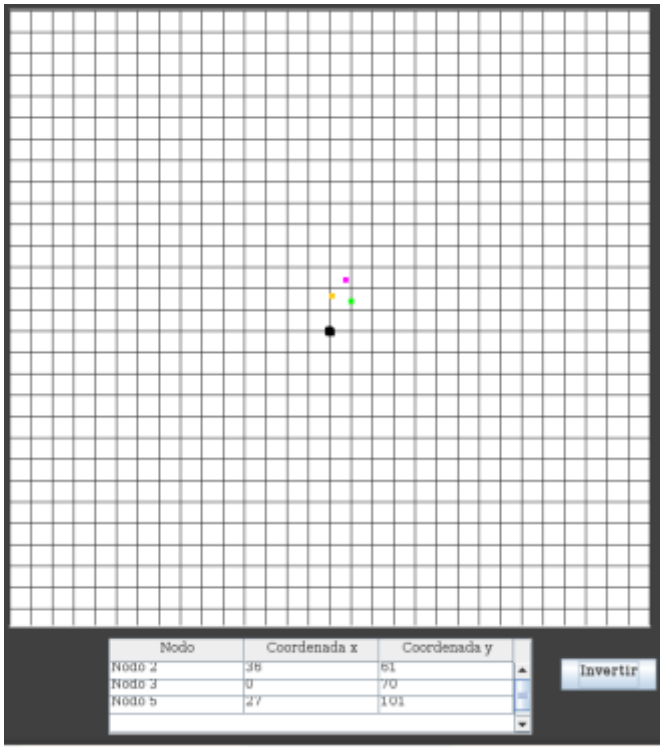

Figura 3.16 LocMap con 3 Nodos conectados, entorno de ruido.

En la Gráfica 3.16, se observa una distribución de la red con un error perceptible en la posición del nodo 5, debido a la interferencia de otros sistemas, sus coordenadas en el eje X deberían ser de alrededor de 100 unidades, pero no llegó a estabilizarse y obtener la posición correcta.

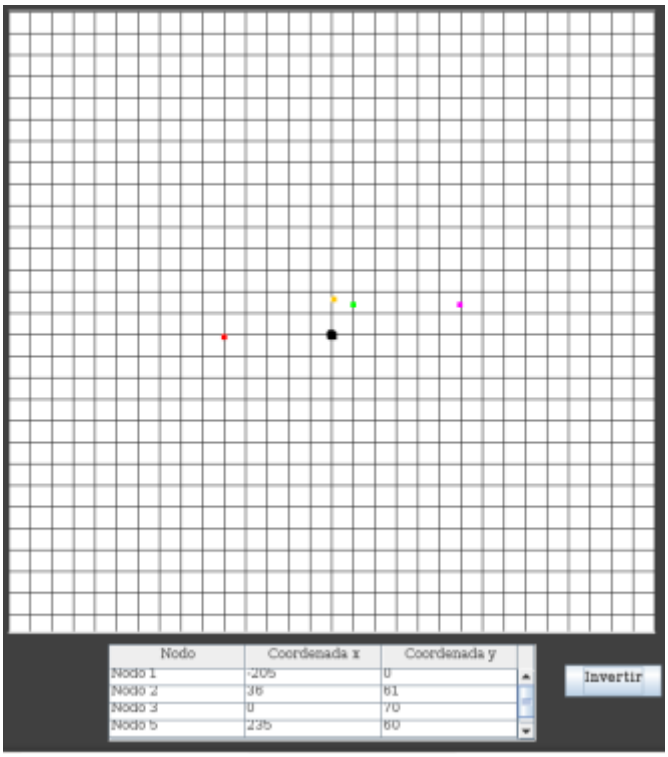

Figura 3.17 LocMap con 4 Nodos conectados, entorno de ruido.

Al incorporarse un cuarto nodo a la red, se observa un salto en la posición del nodo 5, temporalmente la distribución es coherente con la realidad.

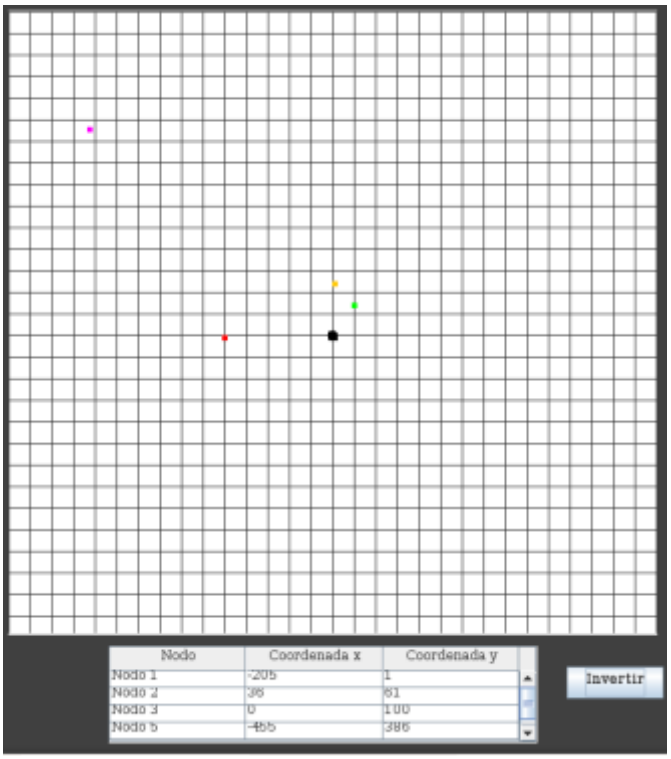

Figura 3.18 LocMap con 4 Nodos conectados, error por ruido.

La distribución de la red luego de 5 ciclos, se ve afectada de nuevo por la interferencia de otros sistemas. Esta inestabilidad refleja las limitaciones en entornos con ruido debido a las propiedades del transceptor RF230 que contiene el nodo IRIS.

3.4. Prueba en entorno abierto con dos niveles.

Esta prueba se realizó para mostrar el funcionamiento del algoritmo y de la red cuando se tiene un nodo fuera del alcance de la Gateway, de tal manera que este debe comunicarse usando un nodo vecino como puente para recibir y enviar su información a la Gateway. Inicialmente se tenían tres nodos conectados de la siguiente manera:

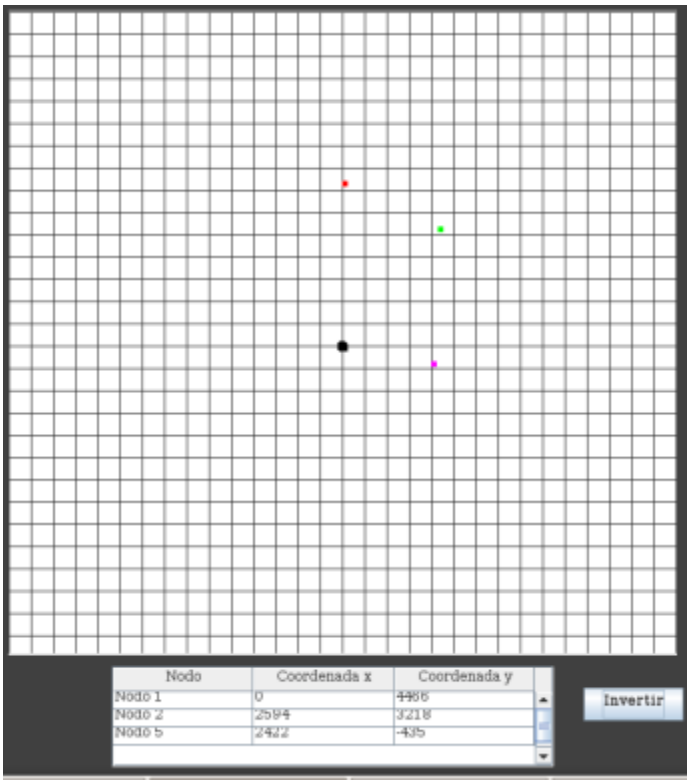

Figura 3.19 LocMap con 3 Nodos conectados.

Al encender un cuarto nodo fuera del alcance de la Gateway, el protocolo *Collection* busca el nodo vecino con las mejores garantías para servir como puente hasta el sumidero, que en este caso es la Gateway.

```
00 FF FF 00 00 18 00 89 00 01 00 02 00 05 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 Hacia gateway: paquete enviado
00 FF FF 00 00 0D 00 89 00 01 06 00 03 01 00 05 04 00 00 00 01 Desde gateway: paquete leido
Fuente: 1 RSSI: 6
00 FF FF 00 00 0D 00 89 00 02 10 00 01 07 00 03 03 00 00 00 01 Desde gateway: paquete leido
Fuente: 2 RSSI: 16
00 FF FF 00 00 0D 00 89 00 05 11 00 02 0E 00 01 03 00 00 00 01 Desde gateway: paquete leido
Fuente: 5 RSSI: 17
00 FF FF 00 00 0D 00 89 00 03 10 00 05 02 00 01 02 00 02 03 00 Desde gateway: paquete leido
Fuente: 3 RSSI: 16
Numero de nodos conectados: 4
Orden: 1, 2, 5, 3
```

Figura 3.20 LocPC con 4 Nodos conectados, un nodo como puente.

En la información recibida del nodo con identificador 3, último paquete recibido, se observa en el recuadro el nodo que está sirviendo para alcanzar la Gateway seguido del RSSI respectivo. El nivel de señal percibido por la Gateway es el mismo que para el nodo 2 debido a que fue este quien envió esa información. El cálculo de las posiciones se puede observar en la Figura 3.21:

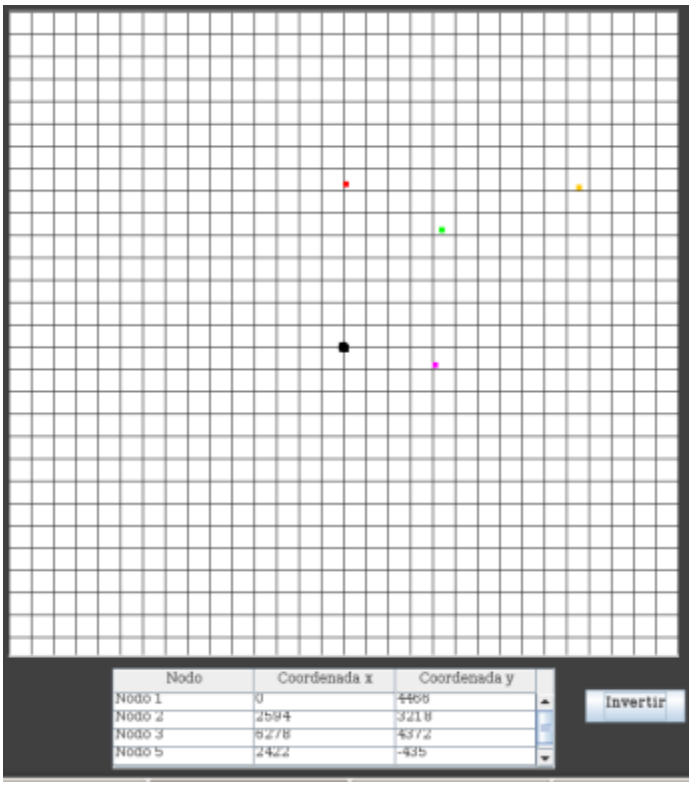

Figura 3.21 LocMap con 4 Nodos conectados, un nodo como puente.

El punto naranja representa la posición del nodo con identificador 3, su puente de conexión es el nodo número 2 representado con el punto verde.

3.5. Errores en las pruebas.

En algunas pruebas del algoritmo de localización se encontraron varios errores producto de los protocolos de enrutamiento *Dissemination* y *Collection.*

```
00 FF FF 00 00 18 00 89 00 01 00 03 00 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 Hacia gateway: paquete enviado
00 FF FF 00 00 0D 00 89 00 01 0A 00 03 00 00 03 00 00 00 00 01 Desde gateway: paquete leido
Fuente: 1 RSSI: 10
00 FF FF 00 00 0D 00 89 00 03 05 00 01 00 00 01 00 00 00 00 01 Desde gateway: paquete leido
Fuente: 3 RSSI: 5
00 FF FF 00 00 0D 00 89 00 05 0D 00 03 03 00 01 08 00 00 00 00 Desde gateway: paquete leido
Fuente: 5 RSSI: 13
Numero de nodos conectados: 3
Orden: 1, 3, 5, 0
Coordenadas= x1:0.0 y1:1122.0
Coordenadas= x1:0.0 y1:1122.0 x2:383760.0 y2:-383812.0
Coordenadas= x1:0.0 y1:1122.0 x2:383760.0 y2:-383812.0 x3:382074 y3:-1531
```

Figura 3.22 LocPC con 3 Nodos conectados, error del protocolo.

En la Figura anterior se puede observar en los recuadros, la lectura que hace el nodo con identificador 1 y 3 respectivamente de sus vecinos, estos valores deberían indicar nodos diferentes con un RSSI valido tal como se observa en la información del nodo con identificador 5. Este comportamiento es aleatorio y solo puede resolverse apagando y encendiendo de nuevo el nodo. El error se identificó cuando se utilizó los protocolos *Dissemination* y *Collection* para brindar capacidades de red de múltiples niveles.

La respuesta del algoritmo es errónea porque los datos suministrados son incorrectos y esto genera fallas en el desarrollo de las ecuaciones para obtener las coordenadas de cada nodo.

Otro problema percibido ocurre cuando un nodo está siendo utilizado como puente, en la siguiente Figura se aprecia el problema:

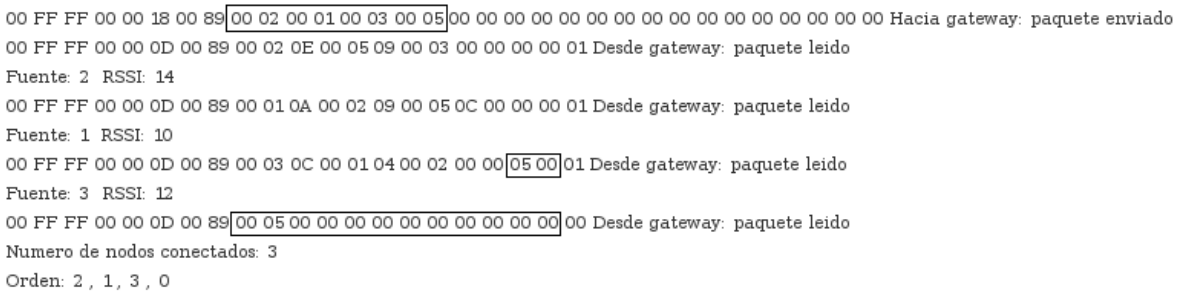

Figura 3.22 LocPC con 4 Nodos, error del protocolo.

En el primer recuadro aparece la lista de los nodos activos en la red, pero en la parte inferior se observa que el nodo con identificador 5 no responde, por tal motivo la Gateway envía su información con ceros, esto significa que el nodo está aparentemente apagado. En este ejercicio los nodos se encontraban estáticos y encendidos.

3.6 Precisión del algoritmo de localización.

Para determinar la precisión del sistema en un entorno sin interferencia, se hicieron mediciones reales con cuatro nodos activos y se comparó con la estimación realizada por el algoritmo, los resultados obtenidos se observan en la tabla 4.

Rango (metros)	**Porcentaje de error en las mediciones**			
	Nodo 1	**Nodo 2**	**Nodo 3**	**Nodo 4**
0-2	3.2	3.8	4.3	4.6
2-5	3.9	4.4	4.8	5.6
5-10	4.6	5.8	6.4	7.1
10-30	7.2	6.6	6.8	8.6
30-70	8.3	7.5	9.2	9.5

Tabla 4. Error en la estimación de distancia.

Los protocolos *Dissemination* y *Collection* permiten brindar capacidades multinivel dentro de la WSN, pero implican problemas de inestabilidad de la red. Otra opción que ofrece TinyOS es el módulo *MultiHopLQI,* protocolo de enrutamiento basado en la calidad del enlace pero que lamentablemente está diseñado para operar con los transceptores CC2420, por este motivo no fue empleado.

En este capítulo se presentaron las pruebas desarrolladas para validar y demostrar la funcionalidad del algoritmo de localización, en diferentes escenarios y bajo condiciones adversas, como también se mostraron las falencias que tiene el sistema debido a los protocolos de enrutamiento empleados.

Capitulo 4

Conclusiones

A continuación, se presentan las conclusiones del desarrollo del algoritmo para la localización de nodos en una Red de Sensores Inalámbricos:

- Los resultados de las pruebas realizadas revelan que el mecanismo de localización implementado muestra la localización de los nodos respecto a sus pares dentro de la red, con un margen de error alrededor del 8%.
- En un entorno abierto y sin interferencias, se puede tener un alcance máximo de alrededor de 75 metros con conexión directa a la Gateway.
- Es necesario ubicar el segundo nodo conectado al lado derecho de la Gateway con respecto al primer nodo, para evitar problemas de orientación.
- Añadir un manejo de vectores para la orientación de los nodos, permitiría que el sistema de corrección de posición sea más preciso.
- Es importante tener un protocolo de enrutamiento que ofrezca las garantías necesarias para evitar inestabilidad de la red y así afectar la operación de la WSN.
- Las aplicaciones de un sistema de ubicación como el desarrollado facilitarían la creación de sistemas de información geográfica (SIG) dinámicos para el monitoreo de elementos móviles o rápida ubicación sobre redes Ad-Hoc.
- En el campo de salud y monitoreo de riesgos o emergencias este tipo de aplicación facilitará las tareas de atención y rescate de víctimas de un siniestro al permitir el montaje de una WSN in-situ y mantener el monitoreo de la ubicación de los nodos asociados a víctimas o fuentes de riesgo.
- Zigbee permite crear soluciones a redes inalámbricas que necesitan un ancho de banda reducido y bajo consumo de potencia, requisitos de aplicaciones orientadas a monitorear y actuar sobre entornos con baja intervención externa.
- TinyOS es el sistema operativo embebido que presenta la mejor alternativa en las redes de sensores inalámbricos, porque ofrece las herramientas requeridas en este tipo de redes y los elementos necesarios para integrarse con otros componentes *hardware*.
- La Programación en NesC es semejante a C++, lo que facilita la familiarización con este lenguaje, al igual que ha sido diseñado para sistemas embebidos, por lo que permite utilizar los recursos de una manera mucho más eficiente.
- RTJava tiene las capacidades necesarias para la implementación de sistemas que demandan tiempos de respuesta cortos y prioridad sobre otros procesos con fácil instalación y adaptación de código.

Trabajos futuros

- Desarrollar un protocolo de enrutamiento que satisfaga las necesidades de una *WSN*.
- Implementar el algoritmo de localización utilizando un sistema de estimación de distancia basado en tiempo.
- Implementar el algoritmo de localización en una red con características diferentes a una *WSN*.

Bibliografía

[1] Haenselmann, Thomas, "Sensor networks", 5 de Abril de 2006. Documento PDF disponible en: http://www.informatik.uni-mannheim.de/~haensel/sn_book/sensor_networks.pdf . Consultado en Mayo 15 de 2008.

[2] Serna, J. "REDES DE SENSORES INALÁMBRICAS". España, Enero 2007, Documento PDF disponible en: http://www.uv.es/montanan/ampliacion/trabajos/Redes%20de%20Sensores.pdf, Consultado en Mayo 20 de 2008

[3] Portal official IEEE Standards Association. http://standards.ieee.org

[4] "Protocolos de control de acceso al medio. Evolución y adaptación a redes de sensores". España. Documento PDF disponible en: http://www.info-ab.uclm.es/descargas/thecnicalreports/DIAB-06-07-1/mac.pdf. Consultado en Mayo 22 de 2008

[5] Castillo, J.. Olivares, T.. Orozco-Barbosa, L. "Routing protocols for wireless sensor networks-based network". Disponible en http://www.info-ab.uclm.es/descargas/thecnicalreports/DIAB-07-06-1/tehcnicalreport.pdf. Consultado en junio 10 de 2008.

[6] Martín, J.. Ruiz, D.. "Informe Técnico: Protocolo ZigBee (IEEE 802.15.4)". España Junio de 2007. Documento PDF disponible en: http://www.zigbee.es/documentos/ZigBee_UA.pdf Consultado en Junio 10 de 2008.

[7] Ilyas, M.. Mahgoub, I., *"Compact Wireless and Wired Sensing Systems"*. CRC Press: Estados Unidos, 2005.

[8] Portal oficial de TinyOS. http://www.tinyos.net/

[9] Ulloa, José. "Estudio y Análisis de las Características de TinyOS". Chile, Septiembre de 2005. Documento disponible en: http://alumnos.elo.utfsm.cl/~julloa/titulo/download/UnoSeccionDos.doc. Consultado en Agosto 15 de 2008.

[10]Ulloa, José. "ESTUDIO COMPARATIVO DE SISTEMAS OPERATIVOS DE TIEMPO REAL APLICADO A REDES DE SENSORES INALAMBRICAS". Chile, Septiembre de 2005. Documento disponible en http://alumnos.elo.utfsm.cl/~julloa/titulo/download/UnoSeccionUno.doc. Consultado en Agosto 15 de 2008.

[11] Culler, David E.. Arch Rock Corp. "TinyOS: Operating System Design for Wireless Sensor Networks". Mayo de 2006. Disponible en: http://www.sensorsmag.com/sensors/article/articleDetail.jsp?id=324975. Consultado en Septiembre 8 de 2008.

[12] Gay, David. Levis, Philip. Culler, David. Brewer, Eric. "NesC 1.1 Language Reference Manual". Mayo de 2003. Documento PDF disponible en: http://nescc.sourceforge.net/papers/nesc-ref.pdf. Consultado en septiembre 16 de 2008.

[13] Alcaraz Calero, Jose María. "TUTORIAL DE TINYOS". Septiembre de 2008. Documento PDF disponible en: http://ants.dif.um.es/rm/apuntes/Tutorial-TinyOS.pdf Consultado en Noviembre 10 de 2008.

[14] Lopez Zamarron, Diego. "Análisis de Sistemas Operativos de Tiempo Real Libres". España, 2004. Documento PDF disponible en: http://tornasol.datsi.fi.upm.es/ciclope/doc/rtos/cache/doc/sotr.pdf. Consultado en Diciembre 6 de 2008.

[15] Alfonso Crespo, Alejandro Alonso. "UNA PANORAMICA A LOS SISTEMAS DE TIEMPO REAL". Disponible en: Revista Iberoamericana de Automática e Informática Industrial. Abril 2006. Vol. 3, No. 2. Págs. 7-18.

[16] De Seta, Leonardo. "Java en Tiempo Real". Junio de 2008. Documento disponible en: http://www.dosideas.com/java/1-java/148-java-en-tiempo-real.html. Consultado en Diciembre 6 de 2008.

[17] The Real Time Specification for Java. Disponible en: http://www.rtsj.org. Consultado en Enero de 2009.

[18] Bollella, Greg. Brosgol, Ben. Furr, Steve. Hardin, David. Dibble, Peter. Gosling, James. Turnbull, Mark. "Real-Time Specification for Java". Prentice Hall 2000.

[19] Atmel Corporation. *"Datasheet Atmel AT86RF230"*.

[20] Wayne, Tomasi. Mata Hernández, Gloria. "Sistemas de comunicaciones electrónicas". Prentice hall, cuarta edición, 2003.

[21] Crossbow Technology Inc., *"IRIS Wireless Measurement System Datasheet"*.

[22] Rmadurai, Vaidyanathan, *"Localization for Wireless Sensor Network"*, 2003. http://www.ece.ncsu.edu/wireless/Projects/localization.html.

[23] Tse, T.H.. Chen, Y.Y.. Chan, W.K. *"Automatic Fault Localization for Wireless Sensor Network Software"*, Mayo de 2007. http://i.cs.hku.hk/~tse/projects.html.

[24] Dong, H.. Lee, H.. "*Robot-Assisted Localization Techniques for Wireless Image Sensor Networks"*, 2006. http://wsnl.stanford.edu/ee392y_06.php.

[25] Rodríguez, Laura, "*Collaborative Localization in Wireless Sensor Networks*", 2007. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?tp=&arnumber=4394904&isnumber=4394880

[26] Sichitiu, M.. Ramadurai, V.. "*Localization of Wireless Sensor Networks with a Mobile Beacon",* http://www.ececs.uc.edu/~cdmc/mass/mass2004/34571.pdf.