ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Estudio e Implementación de un Sistema de Seguimiento de Vehículos con una Red de Sensores Inalámbrica

AUTOR: José Manuel López Egea
DIRECTOR: Dr. D. Fernando Losilla López

Septiembre 2012

Autor	José Manuel López Egea
E-mail del Autor	lopezegea@hotmail.com
Director	Dr. D. Fernando Losilla López
E-mail del Director	fernando.losilla@upct.es
Codirector(es)	
Título del PFC	Estudio e implementación de un sistema de seguimiento de vehículos con una red de sensores inalámbrica.
Descriptores	WSN, Firma Magnética, mote, nesC, TinyOS, IEEE 802.15.4, ZigBee

Resumen

La memoria de este proyecto recoge el estudio realizado para desarrollar un sistema de seguimiento de vehículos basándose en su firma magnética. La firma magnética de un objeto es la manera en que ese objeto produce perturbaciones únicas en el campo magnético terrestre por el hecho de ser metálico por tanto conociendo la firma magnética de un vehículo y detectando esas perturbaciones se puede determinar por donde está circulando.

En las páginas siguientes se va de menos a más en nivel de complejidad, se comenzará introduciendo las redes inalámbricas de sensores, que son las encargadas en este proyecto de monitorizar las firmas magnéticas, después se describirán los elementos software y hardware para poder trabajar con ellas, seguidamente se pasará a explicar que son, como son y como se pueden utilizar en este sistema de seguimiento las firmas magnéticas, posteriormente se describirá el sistema diseñado y para finalizar se mostrarán las conclusiones extraídas de este proyecto.

Titulación	I.T.T. Especialidad Telemática
Intensificación	
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Septiembre 2012

A mis padres,

por todo su esfuerzo

para que yo llegara aquí.

“La constancia es la virtud por la cual todas las otras virtudes dan fruto”.

Arturo Graf.

Agradecimientos

En primer lugar quiero dar las gracias a mi director Fernando, por haberme permitido tener la oportunidad de realizar este proyecto con él y principalmente por todo el tiempo y dedicación que ha invertido en mí y en que este proyecto saliese adelante.

En segundo lugar me gustaría agradecer a todos mis compañeros de carrera porque en mayor o menor medida todos han influido en yo esté aquí. No voy a olvidar nunca todas las clases y prácticas que hemos realizado juntos, todos esos nervios antes de examen que hemos compartido, todas esas horas de reclusión en la biblioteca que se hacían mucho más llevaderas gracias a vosotros.

Agradecer también a mis familiares por todo el cuidado y apoyo que he tenido siempre, en especial a mis abuelos y mi hermana por estar junto a mí en todo momento.

Gracias a Tessy porque desde que entró en mi vida todo ha sido mucho más fácil tanto en la universidad como fuera de ella. He compartido con ella momentos de todo tipo, incluso malos y ella ha estado ahí para animarme y ayudarme a seguir adelante.

Y sobre todo, quiero dar mi infinito agradecimiento a mis padres porque sin ellos sí que estoy seguro de que nunca lo habría logrado. Porque desde pequeño han hecho sacrificios increíbles y los siguen haciendo para que yo viva de la mejor forma y tenga el mejor futuro posible. Gracias por haber confiado en mí hasta en los malos momentos, hasta en aquellos momentos en los que mi trabajo no se correspondía con vuestro esfuerzo. Gracias a mi padre porque sin esos días en los que ni se podía acostar a dormir para seguir trabajando incluso estando enfermo yo no habría podido estudiar y agradecerle a mi madre lo bien que ha cuidado de mi hermana y de mí, en especial, por todos esos días que se tenía que levantar de madrugada para que yo estudiara. Gracias, hoy puedo deciros que vuestro esfuerzo no fue en vano.

Índice General

Índice de Figuras

Índice de Tablas

1 – Introducción

1.1 Resumen y Objetivos

Las redes de sensores inalámbricas a pesar de su concepción relativamente actual han experimentado un gran auge en los últimos tiempos interviniendo en numerosos ámbitos y sectores ya que presentan grandes ventajas respecto a otras tecnologías en cuanto a tamaño, costo y eficiencia energética. Uno de los ámbitos en los que más se utilizan estas redes de sensores es en el de la seguridad en este sentido se van a utilizar en este proyecto.

Por otra parte otro concepto relativamente moderno es el de "firma magnética", que viene a ser una especie de "huella" que dejan los materiales metálicos al alterar el campo magnético terrestre en un determinado lugar.

Aunando estos dos conceptos nace este proyecto, el cual pretende diseñar una red de sensores que sea capaz de detectar esas firmas magnéticas e interpretarlas y distinguirlas unas de otras para poder procesarlas.

Concretamente se pretende aplicar al ámbito de la seguridad vial, donde el personal que lo utilice, por ejemplo policía o personal de mantenimiento de autovías, pueda controlar en cualquier momento por donde se está desplazando un vehículo concreto o capture todas las firmas magnéticas de vehículos que pasan por una determinada zona.

Por tanto el objetivo de este proyecto es analizar si es posible llevar a cabo un sistema de seguimiento de vehículos mediante una red de sensores inalámbricos implantada en calles y autovías y diseñar el software y herramientas necesarias para llevarlo a cabo.

1.2 Palabras Clave

Las palabras clave de este proyecto son las siguientes: red de sensores inalámbrica (WSN), IEEE 802.15.4, ZigBee, Firma Magnética, mote, nesC, TinyOS, vigilancia del tráfico.

2 – Redes de Sensores

2.1 Introducción

Las redes de sensores inalámbricas o "*wireless sensor networks*" (WSN) son redes formadas por un gran número de dispositivos, estos dispositivos reciben el nombre de nodos. Por lo general estos nodos son pequeños aparatos que pueden medir distintos parámetros de su entorno y procesar la información adquirida y/o transmitirla y agruparse para realizar una tarea común.

En una red de sensores inalámbrica la información viaja mediante un medio no guiado, es decir se transmiten mediante ondas electromagnéticas. Este hecho permite que las redes de sensores tengan una gran escalabilidad al disminuir enormemente el número de conexiones cableadas.

Este tipo de redes están diseñadas para enviar los datos medidos a una unidad central de procesamiento mediante el uso de distintas tecnologías sin cable.

Estos sensores tienen la capacidad de conformar una red ad-hoc, es decir, una red sin infraestructura física prestablecida, ni necesidad de control central que coordine su actividad. Una característica de este tipo de redes es su capacidad de autoconfiguración, de modo que los sensores pueden trabajar como emisores o receptores y pueden establecer caminos de comunicación entre nodos sin visibilidad directa y modificar estos caminos si alguno de los nodos que participa en el encaminamiento falla. Además, las redes de sensores en su configuración ad-hoc, pueden implementar protocolos de búsqueda que les permitirán conocer la posición de los diferentes nodos (y por ende, la topología de la red) de forma no centralizada, transmitiéndose la información salto a salto. Esto facilita notablemente el despliegue y mantenimiento de la red, que es resistente a caídas y fallos. Finalmente la información llega a los *gateway*.

El *gateway* es el mote encargado de transmitir a un PC la información recopilada por todos los nodos de la red. Los *gateways* están directamente conectados a la red eléctrica y emplean USB o RS-232 (para comunicación con el PC) o bien Bluetooth, WLAN, Ethernet e incluso podrían utilizar Wi-Fi (para conexión a un dispositivo remoto).

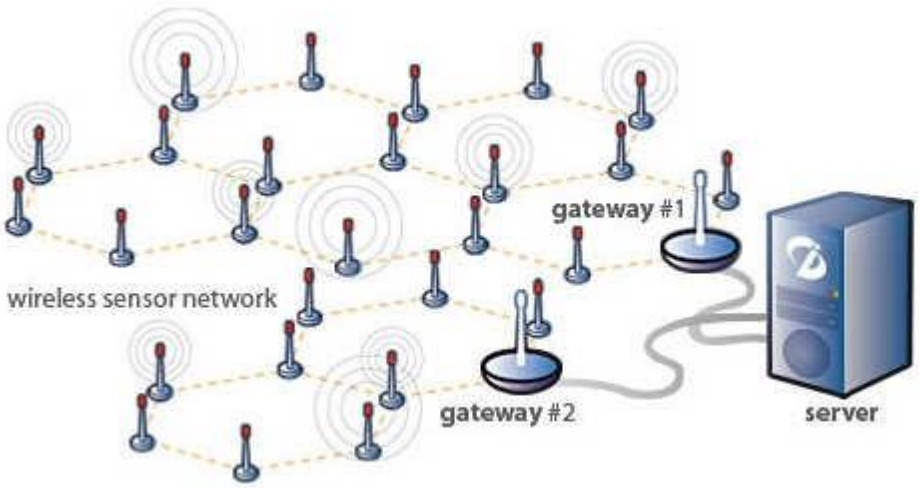

Figura 2.1 : Esquema de interconexión una red de sensores inalámbrica.

2.2 Características de una red inalámbrica de sensores

Aunque la naturaleza de los nodos que emplean las redes pueda ser muy distinta y la misión a realizar pueda variar dependiendo del tipo de aplicación, se pueden identificar una serie de características comunes a todas ellas y que son las que principalmente las identifican:

1. **Gran Escala**. La cantidad de nodos que se despliega en una red puede llegar hasta los miles, pudiendo crecer su número a lo largo de la vida de la red. La red se va a componer de muchos sensores densamente desplegados en el lugar donde se produce el fenómeno y, por lo tanto, muy cerca de él.
2. **Topología variable**. La posición en que se coloca cada nodo puede ser arbitraria y normalmente es desconocida por los otros nodos. La localización no tiene porqué estar diseñada ni prestablecida, lo que va a permitir un despliegue aleatorio en terrenos inaccesibles u operaciones de alivio en desastres. Por otro lado, los algoritmos y protocolos de red deberán de poder organizarse automáticamente.
3. **Recursos limitados**. Los sensores, a cambio de su bajo consumo de potencia, coste y pequeño tamaño disponen de recursos limitados. Los dispositivos actuales más usados, los mica2, cuentan con procesadores a 4 MHz, 4 Kbytes de RAM, 128 Kbytes de memoria de programa y 512 Kbytes de memoria flash para datos. Su radio permite trasmitir a una tasa de 38.4 KBaudios.
4. **Cooperación de nodos sensores**. Realizan operaciones simples antes de transmitir los datos, lo que se denomina un procesamiento parcial o local.

5. **Comunicación**. Los nodos sensores usan comunicación por difusión y debido a que están densamente desplegados, los vecinos están muy cerca unos de otros y la comunicación *multihop* (salto múltiple de uno a otro) consigue un menor consumo de potencia que la comunicación single hop (salto simple). Además, los niveles de transmisión de potencia se mantienen muy bajos y existen menos problemas de propagación en comunicaciones inalámbricas de larga distancia.
6. **Funcionamiento autónomo**. Puede que no se acceda físicamente a los nodos por un largo periodo de tiempo. Incluso tal vez esto nunca ocurra. Durante el tiempo en el que el nodo permanece sin ser atendido puede que sus baterías se agoten o su funcionamiento deje de ser el esperado.

2.3 Arquitecturas de las redes de sensores inalámbricas

Tomando como elementos principales de la red a los nodos sensores, las pasarelas (*gateway*) y las estaciones base, podemos distinguir dos tipos principales de arquitecturas.

1. **Arquitectura centralizada**. En este tipo de arquitectura los nodos de una red que estudian un fenómeno enviarán sus datos directamente a la pasarela más cercana, que dirige el tráfico de esa red en concreto.

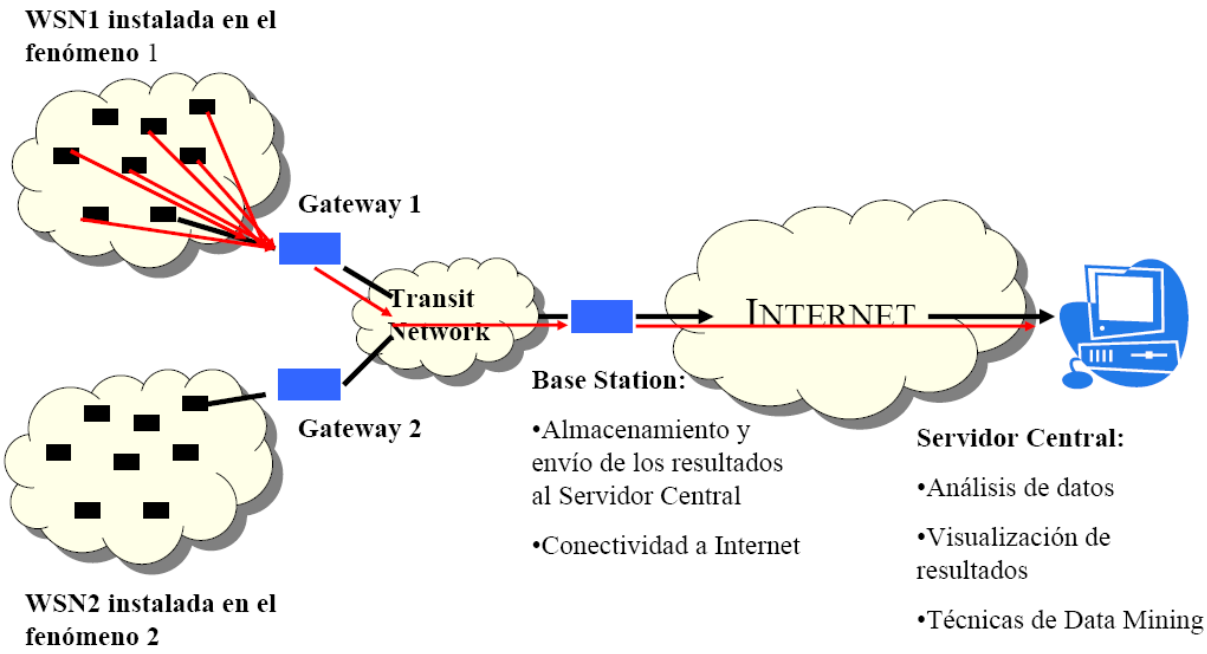

Figura 2.2: Esquema de una WSN centralizada.

Si tenemos en cuenta que el ciclo de vida de un nodo consiste en despertarse, medir, transmitir y dormirse, y cada vez que transmita su mensaje irá a la pasarela, estaremos creando dos grandes problemas para la red:

a. Cuello de botella en las pasarelas.
b. Mayor consumo de energía por las comunicaciones.

Como resultado, el tiempo de vida de la red será relativamente corto.

2. **Arquitectura distribuida**. Dada la naturaleza intrínseca de las redes de sensores, normalmente se tiende a este tipo de arquitectura con una computación distribuida. De hecho, como comentábamos en las características principales de una WSN, son redes basadas en cooperación de los nodos.

   Los nodos sensores se van a comunicar entre sus nodos vecinos y van a cooperar entre ellos, ejecutando algoritmos distribuidos para obtener una única respuesta global que un nodo (*cluster head*) se encargará de comunicar a la estación base a través de las pasarelas pertinentes.

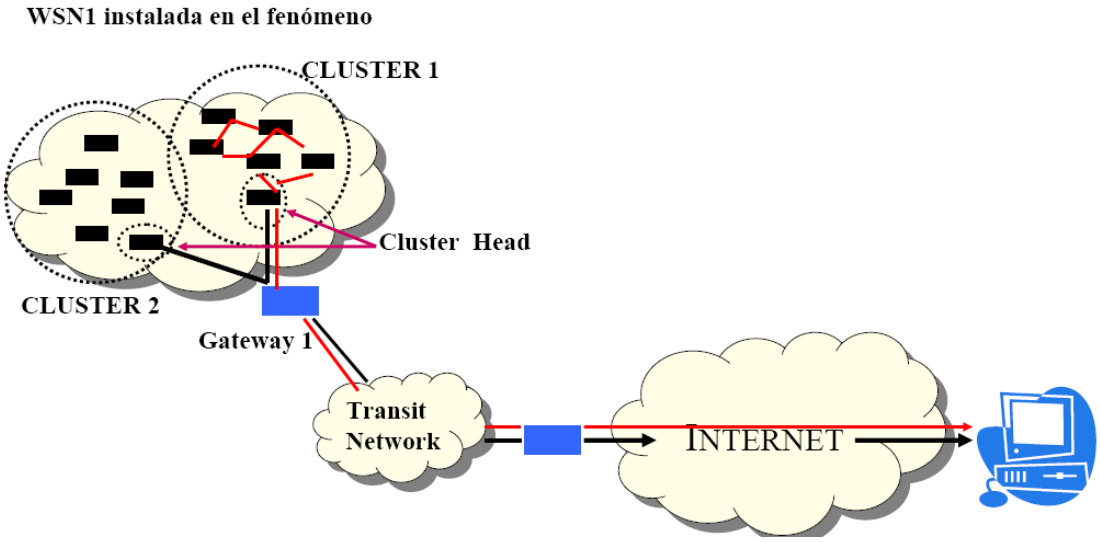

Figura 2.3: Esquema de una WSN distribuida

De esta forma se evitan los problemas que surgían en la arquitectura centralizada y, además, se mantienen sus características y ventajas.

2.4 Aplicaciones de las redes de sensores inalámbricas

Ya en el 2003 el Instituto Tecnológico de Massachusetts calificó las redes inalámbricas de sensores como una de las principales nuevas tecnologías que revolucionarían el mundo, por aquel entonces las funcionalidades que ofrecían eran muy básicas pero con muchas posibilidades. Actualmente han evolucionado mucho las redes

de sensores están a la orden del día y en continuo crecimiento. Las podemos encontrar en múltiples campos con diferentes aplicaciones.

Son muchos los usos de este tipo de redes en el entorno industrial, la mayoría de ellos siguen sin ser explotados y otros muchos no han sido descubiertos por el momento; las posibilidades son enormes, y a continuación se muestran los más importantes:

- **Monitorización del entorno**. Se trata de observar (en tiempo real o no) las variaciones de los distintos parámetros de un entorno concreto; es especialmente útil, por ejemplo, en la agricultura. Son necesarios un gran número de nodos sincronizados, midiendo y transmitiendo periódicamente. La topología física es relativamente estable ya que los nodos no se mueven si no que están situados en posiciones fijas. En función de los resultados que devuelven los dispositivos se puede proceder a efectuar unas acciones u otras sobre el entorno, lo que ahorra mucho en tiempo y recursos. Por ejemplo el proyecto Sitelviña en el que una red inalámbrica de sensores se encarga de monitorizar el entorno y tomar las acciones necesarias para que se cumplan las condiciones correctas de temperatura, humedad, luz, etc. para su fermentación, maduración y embotellado.

Figura 2.4: Sistema Sitelviña.

- **Monitorización de seguridad**. Suelen ser aplicaciones para la detección de anomalías o ataques en entornos monitorizados continuamente por sensores. En este caso los nodos no están transmitiendo datos continuamente, si no que se encuentran en un estado de muy bajo consumo hasta que detecten alguna anomalía. En ese momento será cuando transmitan la información correspondiente. La principal característica de estos entornos es un

aprovechamiento importante de la energía. Existen requisitos de tiempo real; la latencia de las comunicaciones juega un papel importante debido a que las acciones que desatan la transmisión suelen tener una importancia crítica y necesitan de intervención inmediata. La monitorización de seguridad suele estar enfocada a la detección de incendios, al control de edificios inteligentes, aplicaciones militares, etc. Por ejemplo, en Colombia en el 2010 se instaló una red de sensores inalámbrica para vigilar el robo de combustible que se transportaba a través de los oleoductos, lo que permitió reducir considerablemente la cantidad de petróleo robado que ascendía a cuatro millones de barriles al mes.

Figura 2.5: Oleoductos vigilado mediante una red inalámbrica de sensores en Colombia.

- **Tracking**. Aplicaciones para controlar objetos que están "etiquetados" con nodos sensores en una región determinada. A diferencia del resto de aplicaciones, la topología de la red es muy dinámica debido al movimiento de los nodos. Por ello, la WSN deber ser capaz de descubrir nuevos nodos y de formar nuevas topologías.
- **Redes híbridas**. En general, los escenarios de aplicación contienen aspectos de las tres categorías anteriores. Es posible, por ejemplo, tener la necesidad de monitorizar un espacio concreto al mismo tiempo que se lleva cierto control sobre la seguridad de las instalaciones.
- **Redes vehiculares**. Se trata de la transmisión de información entre vehículos en movimiento. Dicha transmisión puede ser con fines de seguridad (informar de un accidente en la vía), confort (recepción de información meteorológica) u ocio (descarga de un podcast). En este caso se tratará de ampliar las posibilidades de estos dispositivos con la intención de transmitir grandes bloques de información. La topología de este tipo de redes es

completamente inestable y puede cambiar en cuestión de segundos, por lo que precisan de una adaptación continua por parte de los nodos.

- **Agricultura proactiva o de precisión**. Mediante la utilización de redes de sensores y la medición de determinados parámetros se llevan a cabo las siguientes acciones:
    - Control de la cantidad de agua, fertilizante o pesticida que las plantas necesitan.
    - Decisión del momento óptimo para realizar la cosecha.
    - Optimización de la producción y la calidad de una cosecha.
    - Gestión de alarmas por intrusión de animales o daños provocados por heladas.

Figura 2.6: Cultivos monitorizados por una WSN y controlados desde una tableta en Hungría.

- **Incendios forestales**. Las redes inalámbricas de sensores son muy útiles para la estimación del riesgo, la prevención y la detección temprana de incendios forestales. Mediante el despliegue de una red de sensores en la zona que se desea proteger, se puede llevar a cabo la monitorización de variables relacionadas con el riesgo de incendios forestales como la temperatura, la

humedad, el nivel de precipitaciones y la velocidad del viento. El conocimiento preciso y continuo de estas variables permite estimar el riesgo de incendio en cada punto de la zona de interés para prevenir el incendio, detectar de forma temprana un foco de incendio y examinar las causas del incendio, a posteriori.

Un ejemplo de uso de una aplicación real de una red de sensores para prevención de incendios en nuestro propios país, es el proyecto NetLife el cual utilizando una red de sensores inalámbrica y midiendo parámetros relevantes, como la temperatura y la humedad relativa en múltiples puntos, alertaría en el caso de que existiese una situación de peligro de fuego en una zona de gran peligro de incendios como es el bosque de Estellencs en Mallorca.

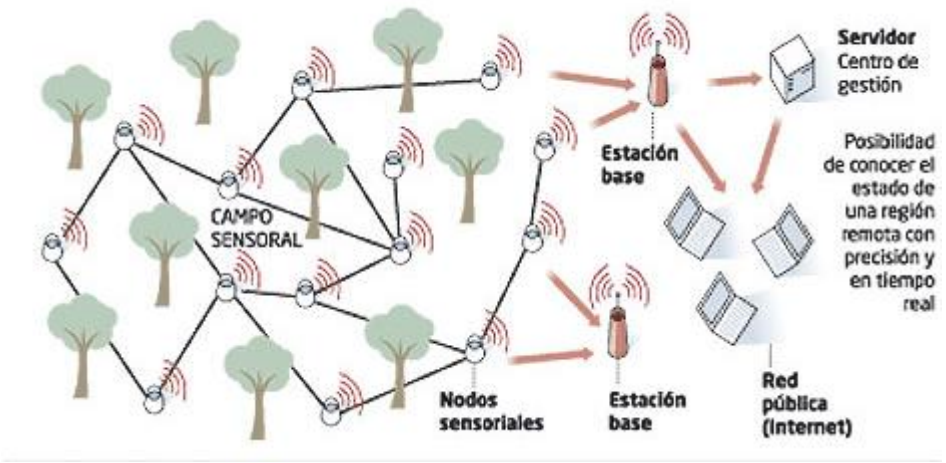

Figura 2.7: Esquema proyecto Netlife.

- **Logística y control de recursos**. En el ámbito industrial, se pueden facilitar las tareas de logística y control de recursos mediante la utilización de sensores. Para ello, se asignan nodos inteligentes a los contenedores de productos o se sustituyen los códigos de barras por etiquetas inteligentes, de manera que los productos están controlados durante todo el recorrido de la cadena logística.
- **Control de procesos**. Para utilizar redes de sensores en el control de procesos hay que tener en cuenta que para este tipo de aplicaciones todos los sensores son vitales y el tiempo es un factor crítico. Algunas de las aplicaciones en este ámbito son la telemetría, las medidas de calidad, el

diagnóstico de maquinaria y la monitorización. Los principales beneficios que se consiguen mediante la utilización de WSN son la reducción de costes y de cableado.

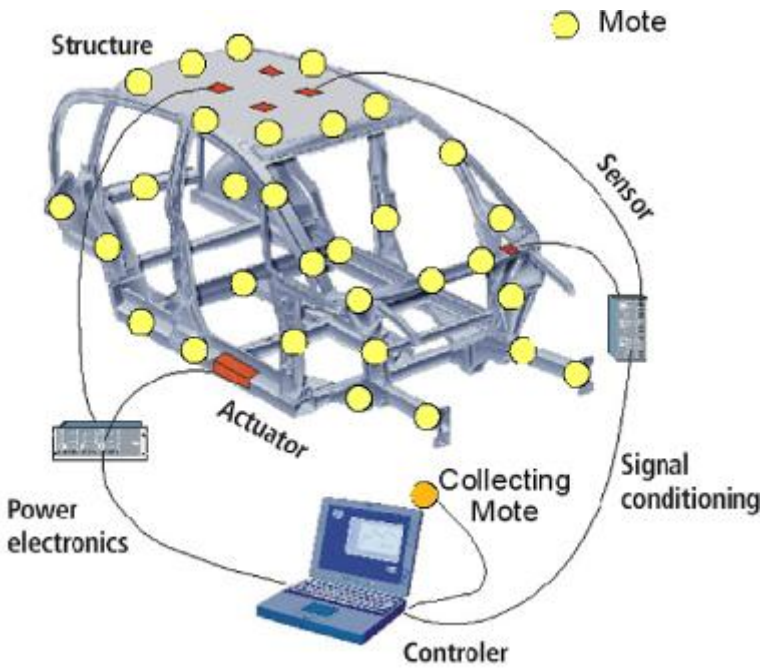

Figura 2.8: Ejemplo de la colocación de sensores inalámbricos en el diseño y fabricación de un automóvil

- **Estudios medioambientales**. Otro campo importante de aplicación de las WSN son los estudios medioambientales. Mediante la medida de variables como luz, velocidad del viento, temperatura, humedad, precipitación, presión barométrica, actividad sísmica, etc. los científicos esperan obtener más conocimiento sobre contaminantes de la tierra fértil, cambios geológicos, flujo del agua, especies invasoras, ciclos de océanos, lugares en los que se almacena el carbono atmosférico, erupción de volcanes y movimiento de virus y fragmentos de genes por el medio ambiente. La ventaja de utilizar este tipo de dispositivos es que se pueden tener miles de sensores repartidos por grandes áreas de forma permanente y desatendida.

  Un ejemplo de este tipo de estudios medioambientales es el estudio del microclima de las secuoyas en Sonoma (California). Los investigadores colocaron 150 motas en las secuoyas con objeto de conocer el microclima monitorizando desde 70 kilómetros de distancia la temperatura, la luz, la presión y la humedad. Algunas de estas motas disponen de un módulo GPS para facilitar la geolocalización.

Otro ejemplo de estudios de microclima con redes de sensores es el de la isla Great Duck, en Estados Unidos. Existe una red de 150 sensores inalámbricos que monitoriza el microclima en los refugios donde anidan las aves marinas y en los alrededores. El propósito es que los investigadores puedan supervisar especies en peligro de extinción y sus hábitats de manera no intrusiva.

- **Entornos inteligentes**. Dentro de la domótica destacan los entornos inteligentes, en los que se desarrollan las siguientes aplicaciones:
    - Control de iluminación y climatización.
    - Detección de presencia para optimizar la energía.
    - Control de acceso y detección de intrusos.
    - Monitorización de personas mayores.

Figura 2.9: Imagen de un hogar inteligente controlado íntegramente por una red inalámbrica de sensores.

- **Entornos interactivos.** Mediante la utilización de WSN se pueden desarrollar las siguientes aplicaciones:
    - Comunicaciones entre vehículos.
    - Señales de emergencia desde ambulancias, policía, bomberos, etc. al resto de vehículos del entorno inmediato.
    - Comunicación y avisos en cruces entre semáforos y vehículos en direcciones perpendiculares.
    - Gestión de aparcamientos.

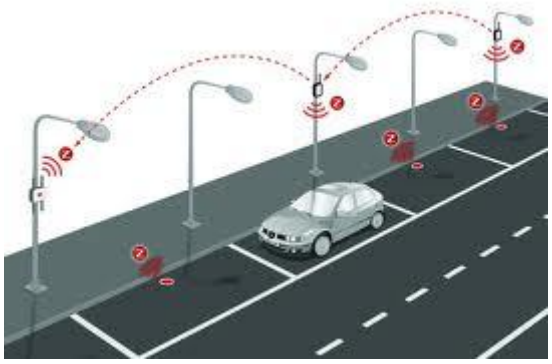

Figura 2.10: Ejemplo de sistema de gestión de aparcamientos mediante una red de sensores inalámbrica.

- **Entornos militares y de alta seguridad**. En este ámbito las WSN son útiles para la prevención de ataques terroristas, por ejemplo, en centrales nucleares, aeropuertos y edificios gubernamentales. Algunos ejemplos de aplicaciones son:
    - *Portable Intrusion Detection System*: equipamiento portátil para detección de intrusos, alimentado con batería, con comunicación remota directa al teléfono móvil o al centro de monitorización, apropiado para todas las aplicaciones de seguridad.
    - Control del "fuego amigo".
    - Plano instantáneo del campo de batalla.
- **Aplicaciones médicas**. Existen numerosas aplicaciones médicas en las que se aplican redes de sensores inalámbricos, entre las que destacan las siguientes:
    - Monitorización continua de pacientes en tiempo real: permiten localización y movilidad, otorgando libertad y movimiento para el paciente en traslados de habitación, en ambulancias, etc., posibilitan monitorización pre-hospital, en el propio hospital y ambulatoria, permiten realizar medidas de constantes vitales (pulsaciones, presión, etc.), sustituyen sistemas de telemetría vía cable que son caros e incómodos.
    - Localización de personal sanitario.
    - Supervisión de pacientes crónicos y ancianos en su casa: las redes se encargan de recoger datos periódicamente para analizar tendencias y

de enviarlos al médico. Así se reduce el tiempo de permanencia en el hospital.

- Bases de datos con recopilación de datos clínicos a largo plazo: se relacionan los datos obtenidos por los sensores con otra información de los pacientes. Permiten realizar análisis de datos de poblaciones y estudios sobre los efectos de intervenciones.

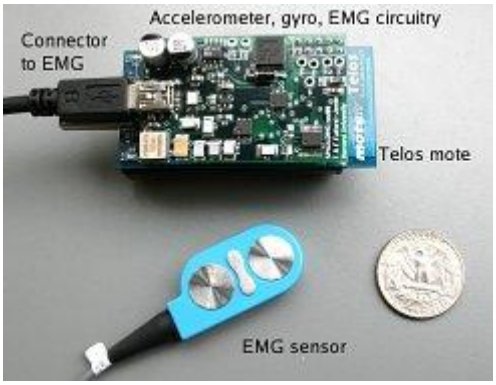

Figura 2.11: Imagen de un nodo inalámbrico con soporte para medir constantes vitales.

En resumen, el uso de este tipo de redes no es uno concreto, si no que sigue en proceso de estudio y aumentando sus posibilidades día a día, lo que promete una implantación futura en campos donde actualmente no están en uso.

2.5 Ventajas e Inconvenientes de su uso

2.5.1 Ventajas

En ámbito general, las principales ventajas que las redes inalámbricas presentan sobre las redes cableadas tradicionales son las siguientes:

- **Movilidad**. Frente a las redes tradicionales la liberación de los cables permite a los usuarios acceder a la información y a los servicios en cualquier lugar y en cualquier momento. Este uso de información en tiempo real supone mayor productividad y posibilidad de servicios.
- **Facilidad y rapidez de despliegue**. En el proceso de instalación de una red podemos encontrar muchos lugares en los que es difícil instalar cables. Incluso en edificios modernos y preparados, las instalaciones pueden constituir un

proceso lento y muy costoso. Mediante el uso de redes inalámbricas, se origina una minimización importante de todos estos problemas.

- **Flexibilidad**. A la facilidad de despliegue antes citada se une la simplicidad de reconfiguración de la red. Las redes inalámbricas interconectan un elevado número de usuarios a través de estaciones base. Un incremento o decremento de este número de usuarios, una vez instaladas las antenas y las estaciones base, se convierte únicamente en una cuestión administrativa (proceso de autorización de usuarios).
- **Reducción de costes**. Uno de los mayores costes de la factura final del despliegue de una nueva red es sin duda alguna la instalación del cableado. Esto se ve incrementado si en la topología de la red se dan cambios frecuentes o el medio en el que se encuentra es muy dinámico. Además, en ciertos entornos, como el industrial, el cableado puede constituir un problema realmente serio por las dificultades que plantea su instalación (imposibilidad de realizar obra civil, condiciones hostiles, maquinaria).
- **Escalabilidad**. Realizar un cambio de topología, en una red inalámbrica, es bastante sencillo y el tratamiento es el mismo tanto si la red es pequeña como si es grande.

2.5.2 Inconvenientes

A pesar de todas estas características antes expuestas, que han sido las que han originado la rápida extensión de las redes inalámbricas en nuestro entorno, no todo son ventajas. Estas redes presentan ciertos inconvenientes que han frenado una implantación de las mismas aún más rápida:

- **Canal físico.** La propia disponibilidad del mismo es un inconveniente: el espectro radioeléctrico, en el cual se realiza la transmisión de los datos, está limitado y se necesita licencia para su uso en muchos casos. Así mismo, la propagación de la información presenta numerosos problemas, como las interferencias o la propagación multicamino, lo que se traduce en tasas de error elevadas. Como consecuencia directa el hardware necesario sea más complejo y los equipos más lentos en la mayoría de los casos.
- **Seguridad.** La naturaleza de las transmisiones en el medio no guiado hace que el acceso a la red esté disponible para cualquiera, por lo que es necesaria la implantación de ciertas medidas de seguridad que eviten el uso de la red por parte de elementos no autorizados.

- **Consumo de energía en los nodos sensores.** Uno de los problemas más importantes es el consumo de energía de los nodos. Para lograr que éste sea mínimo y, por tanto, conseguir un máximo tiempo de vida de la red habrá que tener en cuenta:
    - La comunicación de mensajes es el primer consumidor de energía.
    - La CPU puede quedarse en un estado sleep de bajo consumo mientras no tenga que procesar ni enviar nada.
    - Economizar la distancia de las comunicaciones.
    - Técnicas de software: programación eficiente de líneas de código.
- **Los recursos de computación son limitados**. Los nodos tienen ciertas limitaciones tanto a nivel del procesador como la capacidad de almacenamiento de la memoria. Dependiendo del tipo de sensor, hay diferencias, aunque gracias a las últimas novedades tecnológicas estas limitaciones se van reduciendo.
- **Soluciones ad-hoc para redes ad-hoc**. Muchas de las soluciones ad-hoc no son válidas para este tipo de redes, debido a las diferencias existentes tanto en tecnología como en objetivos.
- **La topología de red es muy dinámica**. Las redes de sensores inalámbricas tienen como uno de sus objetivos crear redes móviles cuya topología va cambiando continuamente, redes caracterizadas por:
    - Nodos móviles.
    - Nodos con alta probabilidad de fallo.
    - Nodos que entran en el sistema.
    - Cuantos más nodos haya en la red mayor será el rendimiento.

2.6 Diferencias entre redes de sensores y redes ad-hoc

Debido a su similitud, las redes de sensores heredan una serie de propiedades importantes de las redes ad-hoc: control descentralizado, ausencia de infraestructura de red, comunicaciones broadcast, canal de transmisión compartido, topologías de corta duración, redes multisalto... Se encuentran tantas similitudes que hasta en diversas fuentes se da una definición de las redes de sensores como casos particulares de redes ad-hoc, denominadas "*Hybrid Ad-Hoc Networks*" aunque, por la particularidad de las redes de sensores, los algoritmos y protocolos usados en estas redes inalámbricas, no se prestan generalmente satisfacer las exigencias específicas de las redes de sensores. De hecho, considerar las redes de sensores inalámbricas simplemente como redes

multisalto, a larga escala, puede no ser adecuado para diversos contextos de aplicación real. Para comprender mejor este concepto es importante adentrarse en las diferencias entre las redes comunes ad-hoc y las de sensores:

- El número de nodos que componen la red de sensores puede ser muy superior al de las redes tradicionales; además, la densidad de nodos vecinos puede resultar muy elevada.
- Los nodos de una red de sensores pueden estar siempre sujetos a fallos.
- El flujo de comunicación de una red de sensores inalámbrica es fuertemente asimétrico. Generalmente todos los nodos envían los datos recogidos al nodo central, mientras en las redes ad-hoc se opta por la comunicación entre nodos simples (*peer to peer*).
- La capacidad energética y de cálculo de los nodos de la red de sensores inalámbrica está extremamente limitada.
- Por el elevado número de nodos de una red de sensores, no todos disponen de un identificador universal (ID).
- Generalmente no es importante por sí misma la información generada por un nodo, pero la unidad final sí es interesante, generalmente, por tener una visión global del sistema, pudiendo compensar la totalidad de los datos reunidos: si un nodo falla es muy probable que esto no influya significativamente en las prestaciones globales del sistema, gracias a la redundancia de las redes de sensores inalámbricas. En este sentido se habla de centralidad de los datos.

2.7 Protocolos de comunicación en redes inalámbricas

Las WSN utilizan una comunicación inalámbrica y, obviamente, son diferentes de las Redes cableadas. También son diferentes de las tradicionales redes inalámbricas como las redes celulares, Bluetooth o las móviles ad-hoc (MANETS). En estas redes, el objetivo es optimizar el rendimiento y el retardo. Aunque las MANETS comparten las características de desarrollo ad-hoc y autoconfiguración de los nodos, el consumo de potencia no es una prioridad. Bluetooth también trata los mismos problemas de limitación de potencia pero el grado de bajo consumo de potencia que se requiere en las redes de sensores inalámbricas es mucho mayor. Además, los nodos sensores son frecuentemente expuestos a extremas condiciones ambientales, haciéndolos propensos a frecuentes fallos en los nodos. Esto conlleva unas restricciones estrictas en las redes de sensores inalámbricas, no como en las otras redes.

- **Capa física**. Aunque la transmisión en las redes de sensores inalámbricas puede ser por infrarrojos, radio o medio óptico, la banda industrial, científica y médica de los 915MHz (ISM) se ha hecho muy popular en las redes de sensores. La deficiencia de usar infrarrojos o medios ópticos para la trasmisión es que requieren que los nodos transmisor y receptor mantengan una línea de contacto visible. Sin embargo, algunas especificaciones inalámbricas como Bluetooth, HomeRF, 802.15.4 y las redes Wireless LAN especificadas por el IEEE 802.11b operan todas en la frecuencia de los 2.4GHz.
- **Capa de enlace**. La responsabilidad de la capa de enlace es establecer un enlace fiable o una infraestructura de red sobre la que los datos puedan ser encaminados. Existen especificaciones como Bluetooth que utilizan la multiplexación por división en el tiempo (TDMA) con saltos de frecuencia mientras que las redes LAN inalámbricas especificadas por el 802.11b utilizan un método de acceso al medio con detección de portadora y evitando colisiones (CSMA/CA).

  La necesidad de un nuevo protocolo de capa MAC para redes de sensores inalámbricas radica en que las dificultades de estas redes son muy distintas de los problemas que tenían las especificaciones existentes. Por ejemplo, el alcance de un piconet, que es una colección de ocho nodos, en una red Bluetooth es de 32 pies, unos 10 metros, mientras que el alcance es mucho más grande en una red de sensores inalámbrica que puede alcanzar los 100 metros. En las redes celulares, las estaciones base forman una columna cableada proporcionando una estructura parcial a la red. En las redes de sensores inalámbricas, no hay estaciones base.

  Un protocolo de capa MAC para las WSN es el llamado MAC autoorganizado para redes de sensores (SMACS), que configura la capa de enlace. El algoritmo de escucha y registro (EAR) permite a los nodos sensores móviles interconectar nodos estacionarios. SMACS actúa al crear la red detectando los nodos vecinos usando transferencia de mensajes. En SMACS, un canal se define con un par de intervalos de tiempo. La detección de vecinos y la asignación de canales se combinan en una fase, para que cuando los nodos vayan a escuchar a sus vecinos, ya hayan formado una red conectada. No hay jerarquías asumidas en SMACS y por esto se forma una topología llana. El algoritmo EAR tiene el

problema del control de la movilidad cuando se introducen nodos móviles en la red.

- **Capa de red**. El encaminamiento en las WSN es bastante similar al de los protocolos ad hoc en las redes MANET. La diferencia es que en los algoritmos de enrutamiento ad hoc, el consumo de potencia es secundario. En redes Bluetooth, las comunicaciones de un nodo master con siete esclavos definen un piconet. Cuando los piconets están interconectados para formar redes dispersas, las diferentes topologías limitan a los nodos que las forman. Para una WSN con la posibilidad de cientos de nodos, esto no será suficiente. Y no sólo eso, sino que también el coste proyectado de un dispositivo Bluetooth es menos de $4 mientras que el precio estimado de un nodo sensor es de menos de $1. Además, los requisitos de potencia de los nodos sensores son mucho menores que para Bluetooth.

  Varios algoritmos se han propuesto para el encaminamiento de WSN. El principal objetivo de los algoritmos es el bajo consumo. Se pueden clasificar como aquellos que determinan:

  1. Rutas con los nodos de mayor potencia a lo largo de la ruta.
  2. Rutas que consuman la mínima energía.
  3. Rutas con el mínimo número de saltos.
  4. Rutas en las que la mínima potencia disponible es la máxima entre todos los demás caminos.

- **Capa de transporte**. La necesidad de una capa de transporte radica en que las WSN necesitan ser conectadas a una red más grande, como Internet. Las WSN se conectan a Internet por medio de pasarelas. El protocolo de la capa de transporte que conecta el usuario con la pasarela podría ser TCP o UDP, ya existentes.

  Sin embargo, el protocolo que conecta la pasarela y los nodos sensores tendría que ser diferente ya que no hay un esquema de direccionamiento global en una red de sensores. La limitación de memoria en los nodos sensores conlleva una cuestión importante en tanto que se prefieren protocolos que requieran un bajo almacenamiento de información de estado antes que otros del tipo TCP.

- **Capa de aplicación**. Los nodos sensores tienen muchas aplicaciones distintas. Diseñar una capa de aplicación tiene el mérito de que las WSN pueden ser conectadas a grandes redes como Internet. El direccionamiento de nodos es

una cuestión importante aquí ya que, no como en otras redes, los nodos sensores no tienen un identificativo global.

Los protocolos de la capa de aplicación como el Protocolo de Administración de Sensores (SMP) y el Protocolo de Petición de Sensores y Entrega de Datos (SQDDP) están actualmente en investigación. El SQDDP introduce una interfaz de peticiones para emitir peticiones, responderlas y recopilar las respuestas de las peticiones.

Aunque las aplicaciones de las redes de sensores inalámbricas son diversas, las últimas investigaciones indican que se van a tener que realizar más desarrollos en el direccionamiento de varios protocolos en todas las capas. Además, se necesita una plataforma común donde se puedan probar los algoritmos propuestos. Las simulaciones de WSN son difíciles y normalmente están vinculadas a una aplicación en particular.

El área de las redes de sensores inalámbricas es un área en expansión y en los próximos años veremos los resultados de los esfuerzos que se están realizando en estas aplicaciones.

A continuación se puede ver un esquema de los distintos niveles que se pueden encontrar en cualquier red de sensores:

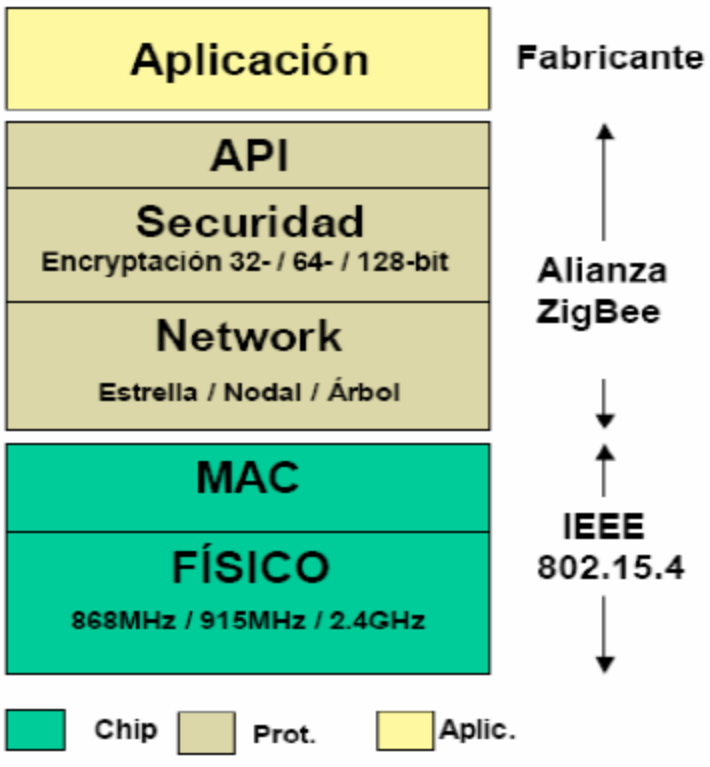

Figura 2.12: Niveles físico, red y aplicación

2.8 ZigBee

Tras describir en profundidad las características necesarias cada una de las capas presentes en una red de sensores inalámbrica, se procede a describir la solución más extendida, el protocolo ZigBee.

2.8.1 Introducción

ZigBee es una nueva tecnología de inalámbrica de corto alcance y bajo consumo originaria de la ZigBee Alliance y que se definió como una solución inalámbrica de baja capacidad para aplicaciones en el hogar como la seguridad y la automatización.

Las aplicaciones que puede tener son las mismas que las comentadas para una red de sensores inalámbrica cualquiera, entre las que destacan:

- Domótica.
- Automatización industrial.
- Reconocimiento remoto.
- Juguetes interactivos.
- Medicina.
- Etc.

2.8.2 Objetivo

El objetivo de esta tecnología no es obtener velocidades muy altas, ya que solo puede alcanzar una tasa de 20 a 250Kbps en un rango de 10 a 75 metros, si no que es obtener sensores cuyos transceptores tengan un muy bajo consumo energético. De hecho, algunos dispositivos alimentados con dos pilas del tipo AA puedan aguantar dos años sin el cambio de baterías. Por tanto, dichos dispositivos pasan la mayor parte del tiempo en un estado latente, es decir, durmiendo para consumir mucho menos.

2.8.3 Bandas de operación

ZigBee opera en las bandas libres de 2.4Ghz, 858Mhz para Europa y 915Mhz para Estados Unidos. En la siguiente figura se puede ver el espectro de ocupación en las bandas del protocolo 802 (incluyendo ZigBee).

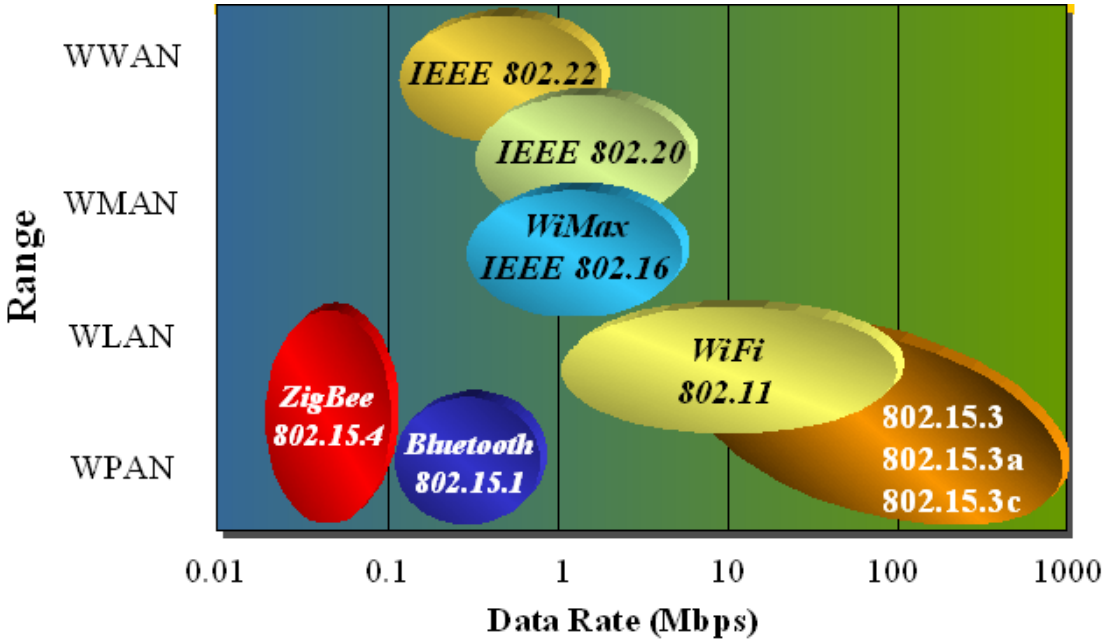

Figura 2.13: Tecnologías en 2.4Ghz

En la banda de 2.4Ghz usa la modulación de espectro expandido DSSS (*Direct Sequence Spread Spectrum*). A una velocidad de transmisión de 250Kbps y a una potencia de 1mW cubre aproximadamente unos 13 metros de radio.

En la siguiente figura se muestran las características de radio de las señales.

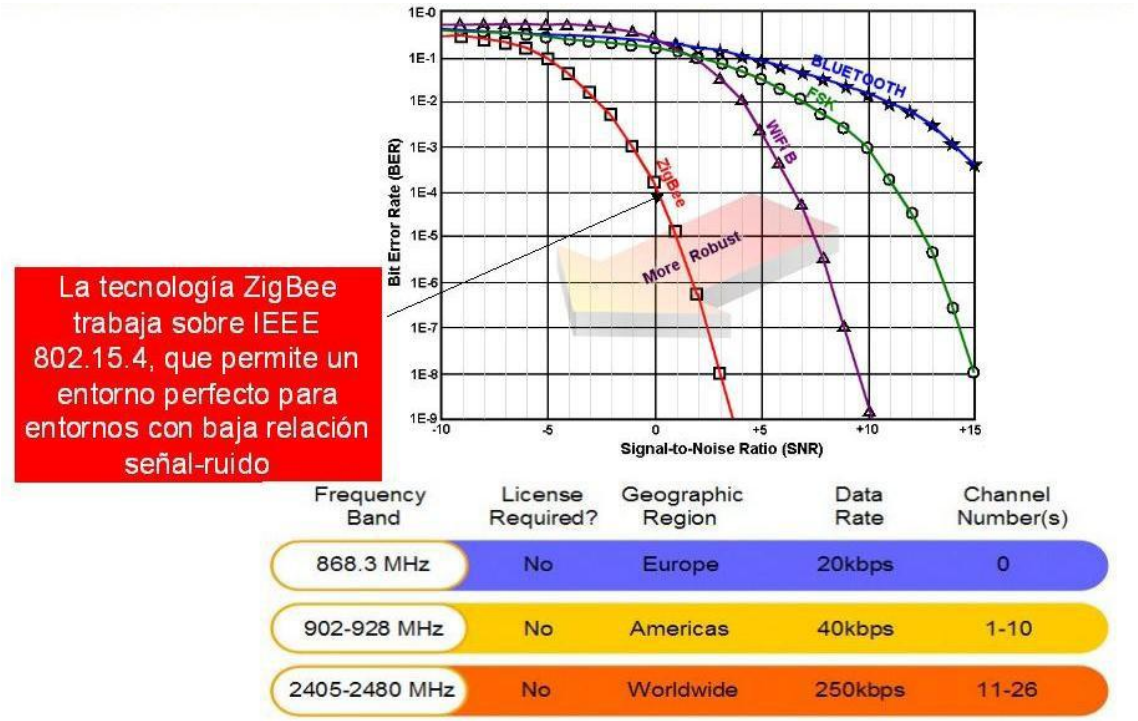

Figura 2.14: Características de radio

En la siguiente tabla se puede observar la distancia en función de la potencia transmitida y la velocidad de transmisión:

Potencia(mW) / Velocidad(Kbps)	1mW	10mW	100mW
28 Kbps	23m	54m	154m
250 Kbps	13m	29m	66m

Tabla 2.1: Distancia de transmisión.

En cuanto a la gestión del control de acceso al medio hace uso de CSMA/CA (*Carrier Sense Multiple Acces with Collision Avoidance*) y es posible usar ranuras temporales TDMA (*Time Division Multiple Access*) para aplicaciones de baja latencia.

2.8.4 Nodos y topología de red

En una red ZigBee pueden haber hasta 254 nodos, no obstante, según la agrupación que se haga, se pueden crear hasta 255 conjuntos/clusters de nodos con lo cual se puede llegar a tener 64770 nodos para lo que existe la posibilidad de utilizar varias topologías de red: en estrella, en malla o en grupos de árboles, como puede verse a continuación:

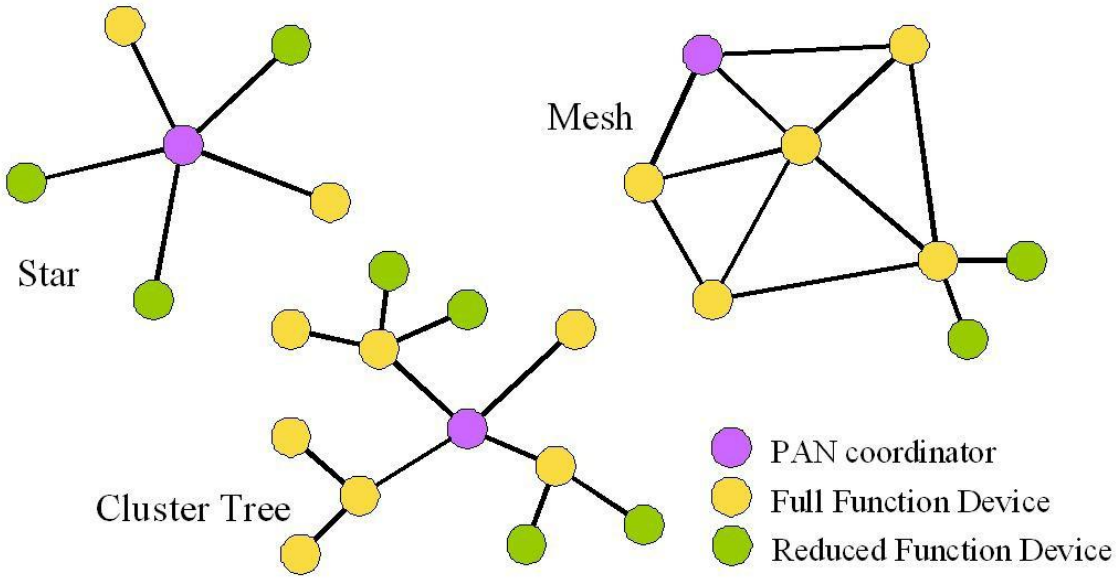

Figura 2.15: Topologías de Red

Se permite un encaminamiento o enrutamiento de saltos múltiples, también conocido como multi-hop, que permite que estas redes abarquen una gran superficie.

En ZigBee hay tres tipos de dispositivos:

- Coordinador
  - Sólo puede existir uno por red.
  - Inicia la formación de la red.
  - Es el coordinador de PAN.
- Router
  - Se asocia con el coordinador de la red o con otro router ZigBee.
  - Puede actuar como coordinador.
  - Es el encargado del enrutamiento de saltos múltiples de los mensajes.
- Dispositivo final
  - Elemento básico de la red.
  - No realiza tareas de enrutamiento.

Una posible configuración de una red sería la siguiente:

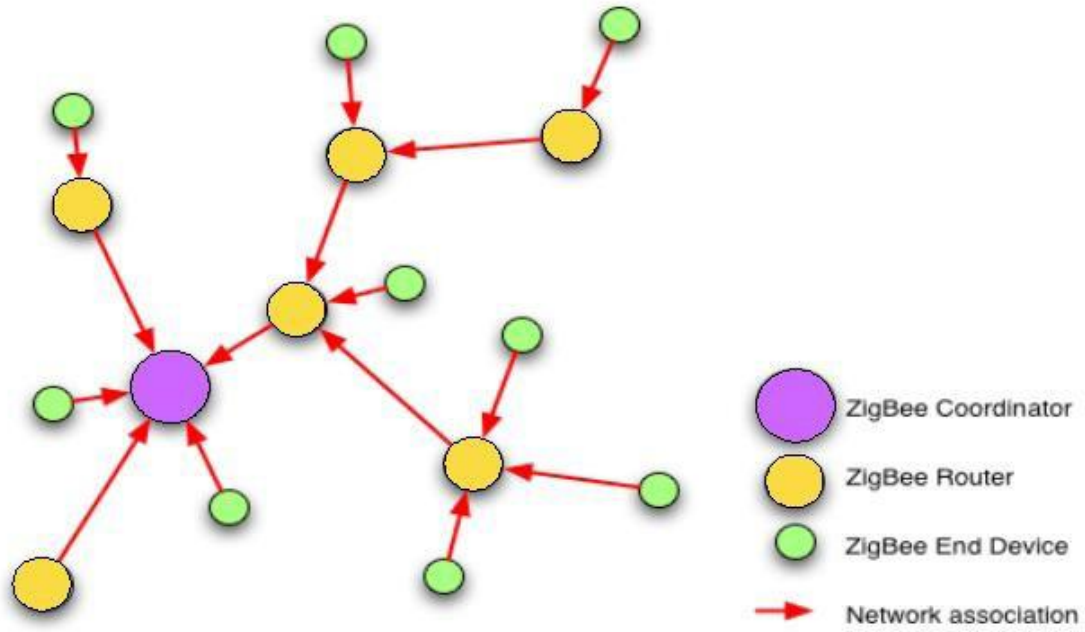

Figura 2.16: Ejemplo de red ZigBee.

Otro punto importante es el soporte y la disponibilidad total de la malla, es decir, que ante caídas de nodos, la red busca caminos alternativos para el intercambio de mensajes, un ejemplo se puede ver a continuación.

Supongamos que disponemos de una red en la cual los nodos están conectados en malla y se intercambian datos entre un interruptor y una lámpara.

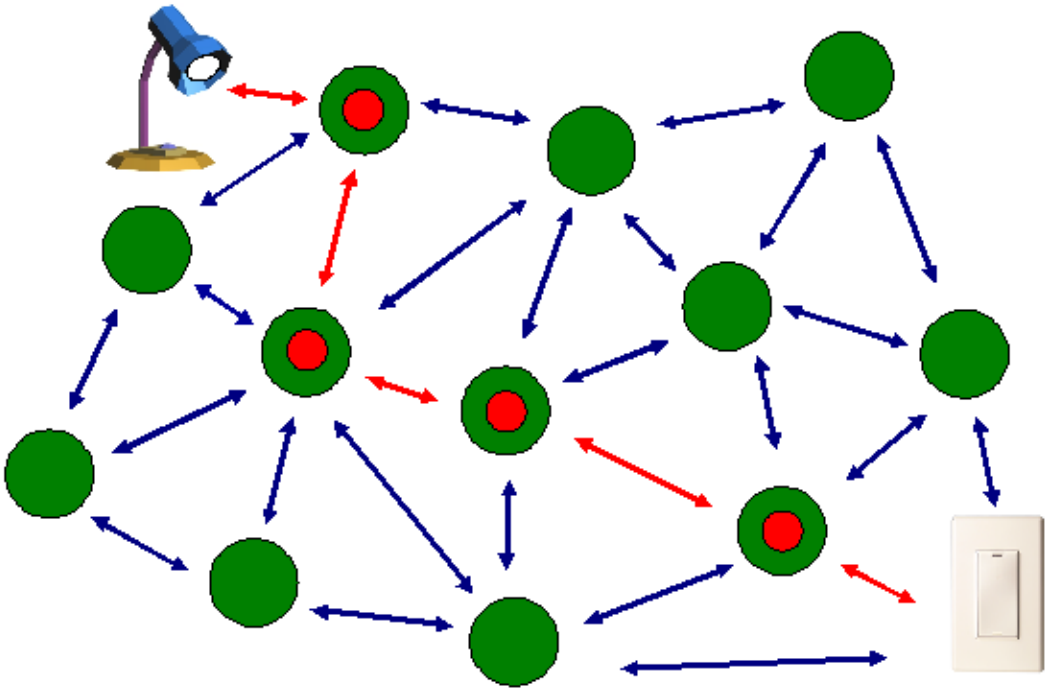

Figura 2.17: Camino de comunicación (interconexión).

Si algunos de los nodos que contiene falla y dichos nodos formaban parte del camino que seguían los mensajes en la comunicación, la red podría sufrir una caída:

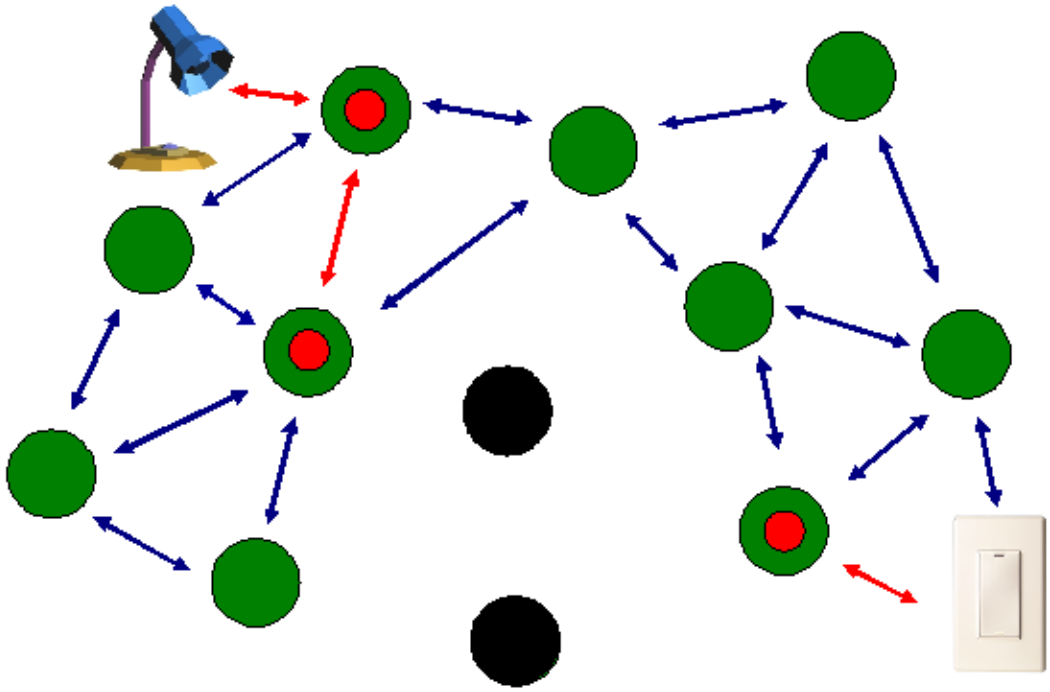

Figura 2.18: Caída de dos nodos de red.

ZigBee permite que se puedan establecer rutas alternativas para seguir comunicando los dispositivos:

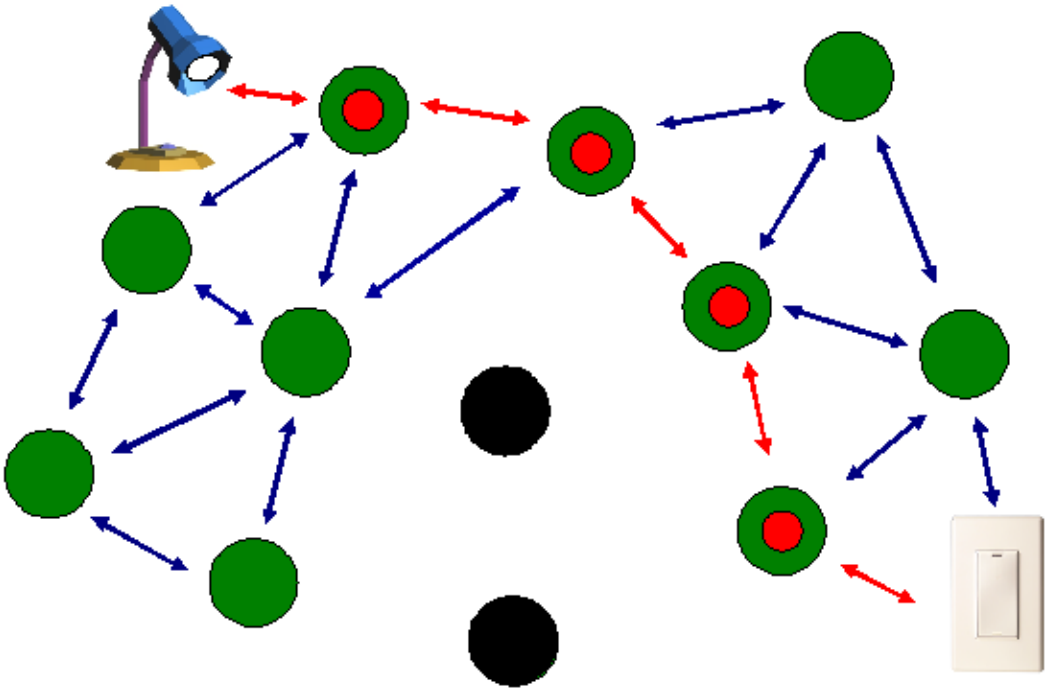

Figura 2.19: Creación de un camino alternativo

2.8.5 Seguridad

En cuanto a seguridad, ZigBee puede utilizar la encriptación AES de 128bits, que permite la autentificación y encriptación en las comunicaciones. Además, existe un elemento en la red llamado Trust Center (Centro de validación) que proporciona un mecanismo de seguridad en el que se utilizan dos tipos de claves de seguridad, la clave de enlace y la clave de red.

2.8.6 ZigBee vs Bluetooth vs WiFi

En la siguiente tabla se puede apreciar una comparación entre las principales características de las tres tecnologías más usadas en el ámbito de las redes inalámbricas, dónde es fácil apreciar las ventajas de escalabilidad, bajo consumo y eficiencia razones que han hecho que este estándar sea el adoptado para las comunicaciones en redes inalámbricas de sensores.

Estándar	Wi-Fi 802.11g	Wi-Fi 802.11b	Bluetooth 802.15.1	ZigBee 802.15.4
Aplicación principal	WLAN	WLAN	WPAN (sustituir cable entre dos dispositivos)	Control y monitorización
Memoria necesaria	1MB+	1MB+	250KB+	4KB - 32KB
Vida Batería (días)	0,5 - 5	0,5 - 5	1 – 7	100 - 1000+
Tamaño Red	32 nodos	32 nodos	7	255 / 65.000
Velocidad (Kbps)	54 Mbps	11 Mbps	720 Kbps	20 – 250 Kbps
Cobertura (metros)	100	100	10 (v1.1)	1 – 100
Parámetros más importantes	Velocidad y Flexibilidad	Velocidad y Flexibilidad	Coste y perfiles de aplicación	Fiabilidad, bajo consumo y muy bajo coste

Tabla 2.2: Comparación principales tecnologías inalámbricas.

2.8.7 Futuro

Se espera que los módulos ZigBee sean los transmisores inalámbricos más baratos de la historia, y además producidos de forma masiva. Tendrán un coste aproximado de alrededor de los 6 euros, y dispondrán de una antena integrada, control de frecuencia y una pequeña batería. Ofrecerán una solución tan económica porque la radio se puede fabricar con muchos menos circuitos analógicos de los que se necesitan habitualmente.

3 – Entorno de trabajo

3.1 Introducción

En este capítulo se pretenden describir las principales herramientas con las que se ha acometido este proyecto fin de carrera, se describirán en detalle tanto los dispositivos hardware que se han utilizado para las pruebas como los entornos software en los que se ha realizado el diseño.

3.2 Nodos Sensores

3.2.1 Conceptos

Los nodos sensores, o motes, son los componentes principales de las redes de sensores inalámbricas no son exactamente sensores, sino que son una especie de microcomputadores con diversas funciones. Los motes son ordenadores autónomos de tamaño reducido, con sensores para monitorizar el entorno y con radios para comunicarse con otros motes.

Son dispositivos compuestos por un microprocesador con memoria, uno o varios sensores, radio de baja potencia y una batería. Los nodos pueden incluir una gran variedad de sensores, estos pueden ser de temperatura, humedad, luminosidad, presión, detectores de gases (CO, CO2, CH4, etc.), de campos magnéticos (el utilizado en este proyecto), detectores de sonido y movimiento, etc. Estos sensores deben conectarse de forma sencilla al nodo sensor. Un nodo sensor integra, al menos, los siguientes componentes:

- Radio: existen varias alternativas, siendo la IEEE 802.15.4 y la Zigbee las más utilizadas por el amplio respaldo que cuentan de la industria a nivel mundial. Emplean diversas bandas de frecuencia de radio:
  - 2.4 GHz con una velocidad de transmisión de 250 Kbps,
  - 915 MHz (banda americana) con una velocidad de transmisión de 40 Kbps.
  - 868 MHz (banda europea) con una velocidad de transmisión de 40 Kbps.
- Microprocesador: su velocidad suele ser de varios Megahertzios y tiene una RAM y ROM reducida (entorno a 4KB de RAM y 128 KB de EEPROM).
- Alimentación: generalmente son pilas de Níquel-Cadmio o batería de Litio, en muchos casos se utilizan pilas convencionales (2xAA) con capacidad

aproximada de 2000-3000mAh. En la actualidad existen nodos sensores alimentados por baterías de litio que son recargados con energía alternativa a través de la luz e incluso a través de las vibraciones.

- Unidad sensora, con una serie de sensores que se encargan de adquirir los datos. Como se ha mencionado anteriormente estos sensores pueden ser de muy diversos tipos: temperatura, humedad, luz, etc.…

En la siguiente figura se aprecia el esquema general de cualquier nodo sensor:

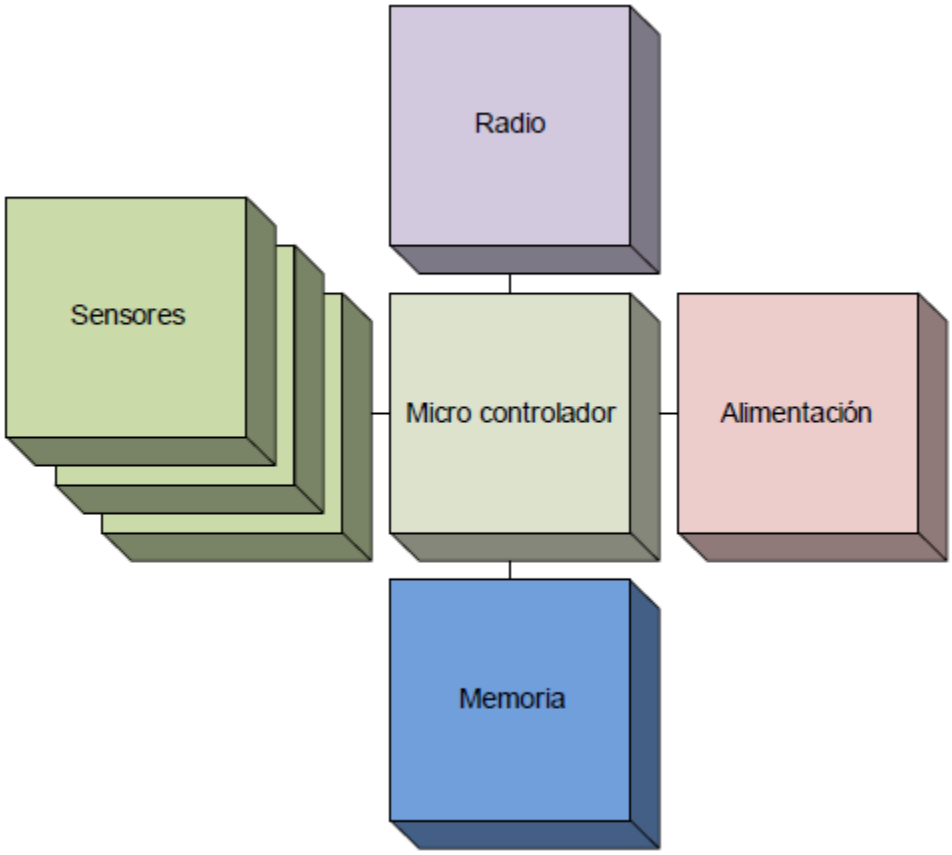

Figura 3.1: Componentes de un nodo sensor

3.2.2 Componentes de un nodo sensor

Entrando más en detalle en los componentes que forman un nodo sensor se puede destacar lo siguiente:

3.2.2.1 Unidad sensora

El propósito de los nodos de una red de sensores no es computar o comunicar, sino detectar. El componente sensor de los motes es la actual tecnología de cuello de botella, estas tecnologías actualmente no están progresando con la rapidez de los

semiconductores. Las limitaciones conceptuales son significativamente más estrictas para los sensores que para los procesadores o el almacenamiento. Por ejemplo, la interfaz de los sensores con el mundo físico, mientras que las unidades de cómputo y comunicación se relacionan con un gran entorno controlado de un solo chip. Los transductores son unos componentes en un mote que se usan para transformar una forma de energía en otra. El diseño de los transductores se considera fuera del alcance de la arquitectura del sistema.

Además, los sensores pueden tener otros cuatro componentes: convertidor analógico-digital, un convertidor digital-analógico y un microcontrolador. El diseño más simple puede incluir un solo transductor.

Uno de los retos principales de las redes de sensores es seleccionar el tipo y la cantidad de sensores, así como determinar su emplazamiento. Esta tarea es difícil ya que hay numerosos tipos de sensores con diferentes propiedades como la resolución, el coste, la precisión, el tamaño y el consumo de potencia.

Además, se suele necesitar más de un tipo de sensor para asegurar la exactitud de las operaciones y de los datos que provienen de diferentes sensores que se pueden combinar.

Los sensores utilizados en las distintas plataformas pueden ser de diferentes clases:

- Modalidad de baja potencia:
    - Fotodetector (sensor de luz)
    - Sensor de temperatura
    - Acelerómetro de dos dimensiones
- Modalidad de media potencia:
    - Micrófono (detector acústico de umbral)
    - Magnetómetro
- Modalidad de alta potencia:
    - Sensor de imagen
    - Sensor de video
    - Sensor de forma de rayo

Figura 3.2: Placa sensora de para mote MicaZ.

3.2.2.2 Unidad de procesamiento

Esta unidad se encuentra divida a su vez en dos subunidades: el procesador y la unidad de almacenamiento.

3.2.2.2.1 Procesador

Los microcontroladores son la primera opción para el procesamiento dentro del nodo. Los requerimientos de memoria dependen de la aplicación que se vaya a implementar. Un ejemplo de microcontrolador es el Atmel ATMega128L quizás el modelo más incluido en los sensores inalámbricos y que se puede apreciar en la siguiente figura:

Figura 3.3: Atmel ATMega128L.

Entre las características más destacadas de este modelo posee una arquitectura RIS de 133 instrucciones y dos multiplicadores en el propio chip. En cuanto a memoria posee una memoria EEPROM, una memoria Flash reprogramable y una memoria

interna SRAM. En cuanto a energía se refiere, posee seis modos de funcionamiento diferentes para ahorrar energía.

3.2.2.2.2 Unidad de almacenamiento

Dependiendo de la estructura de toda una red de sensores, los requerimientos de almacenamiento en términos de rapidez y no-volatilidad de memoria en cada nodo pueden ser diferentes. Por ejemplo, si se sigue el modelo arquitectónico en el cual la información se envía instantáneamente al nodo central, se necesita muy poco almacenaje local en los nodos individuales. Sin embargo, en un escenario donde la meta es minimizar la cantidad de comunicación y conduce a una parte importante en cada nodo individual, será necesario un requerimiento significante de almacenaje local. Existen al menos dos alternativas de almacenamiento de datos en un nodo local:

- La primera opción es usar una memoria flash, que es una solución muy atractiva en términos de coste y capacidad de almacenamiento. Sin embargo, tiene relativamente serias limitaciones en cuestión de las veces que puede ser utilizada para el almacenamiento de diferentes tipos de datos en las mismas posiciones físicas.
- La segunda opción es utilizar memorias NVRAM basadas en elementos nanoeléctricos. Se espera que las NVRAM se puedan utilizar pronto para apoyar un número elevado de aplicaciones en número de áreas.

3.2.2.3 Módulo radio

Las radios de corto alcance como los componentes de comunicación son excepcionalmente importantes porque la parte del presupuesto de energía dedicado a enviar y recibir mensajes normalmente domina el conjunto del presupuesto de energía.

Durante el diseño y la selección de las radios, se debe considerar al menos tres capas diferentes de abstracción: capa física, capa MAC y la capa de red.

La capa física es la responsable de establecer los enlaces físicos entre el transmisor y uno o más receptores. Las principales tareas a este nivel incluyen la modulación de señal y la encriptación de los datos para mantener la comunicación en presencia de un canal de ruido y señales interferentes. Para usar de manera eficiente el ancho de banda y reducir el coste de desarrollo, se utilizan varias radios que comparten el mismo medio de interconexión. La participación media es facilitada por la capa MAC.

Finalmente, la capa de red es la responsable de establecer el camino que un mensaje debe realizar a través de la red para que se transfiera de la fuente a su destino.

El diseño de potencia y la eficiencia de ancho de banda de las radios es una de las principales investigaciones y tareas de desarrollo. Es importante darse cuenta de que la arquitectura radio es una función en la que se emplea arquitecturas de red y protocolos.

Hay que llegar a un compromiso entre el coste relativo de energía en transmisión y recepción ya que escuchar el canal es muy caro en términos energéticos. Por lo tanto, es necesario desarrollar esquemas que combinen periodos de actividad con periodos de standby o inactividad en los sensores.

Las radios se encuentran en chips comercialmente. Las bandas de frecuencias en las que suelen operar son: 433MHz, 916MHz, 2.4MHz en las bandas ISM.

La potencia típica que transmite es 0dBm y la sensibilidad en recepción es tan baja como - 110dBm. La comunicación se realiza en banda estrecha con la modulación FSK o espectro ensanchado. Generalmente tiene unas tasas de transmisión relativamente bajas (menores a 100kbps), lo que permite ahorrar potencia.

3.2.2.4 Suministro de energía

Un gran consenso es que la energía será una de las restricciones principales tecnológicas para los nodos de las redes de sensores. La energía puede ser suministrada de dos formas principalmente:

1. Equipando cada nodo sensor con una (recargable) fuente de energía. En la gran mayoría de las plataformas existentes, se suministra la potencia mediante las baterías AA. Estas baterías son las que dan el mayor o menor tamaño del nodo. Pero aunque las baterías alcalinas ofrecen una alta densidad de energía a un precio bajo, la curva de descarga está lejos de ser plana. Otra opción son las baterías de botón que son más compactas y poseen de una curva plana de descarga.

Figura 3.4: Pila clásica que sirve para alimentar un sensor hasta 2 años.

2. La segunda alternativa es recoger la energía disponible en el entorno como pueden ser las células solares. Actualmente ya son ampliamente utilizadas

para aplicaciones móviles como las calculadoras y se pueden utilizar para algunas aplicaciones.

También existen otras formas alternativas todavía en investigación como:

- Las células de combustible.
- Suministro de potencia mediante un sistema inalámbrico de baja batería que recoge el calor del ambiente. El principal componente de este sistema es un convertidor DCDC con condensador conmutado, un módulo microtermoeléctrico hace el sistema posible.

3.2.3 Modelos de nodos sensores

En la actualidad existen multitud de plataformas de nodos sensores de los distintos fabricantes, varían en cuanto a su tamaño, consumo de energía, capacidad de proceso y almacenamiento. A continuación se muestra una lista de nodos sensores de las plataformas más comunes en el desarrollo de aplicaciones de redes de sensores inalámbricas.

3.2.3.1 BTNode3

El BTnode es un nodo sensor inalámbrico que posee una radio de baja potencia y un microcontrolador. Está diseñado como plataforma de demostración y creación de prototipos para la investigación en telefonía móvil, sensores (WSN). La radio es la misma que en el mote Mica2 de Berkeley.

Figura 3.5: Nodo sensor BTNode3

3.2.3.2 EyesIFX

Desarrollado por Infineon en colaboración con el proyecto EYES de redes inalámbricas de sensores. Su microprocesador pertenece a la familia MSP430 producidos por Texas Instruments. La plataforma integra sensores de temperatura y de

luz, además ofrece un soporte completo de software proporcionado por TinyOS de TU Berlín.

Figura 3.6: Nodo sensor EyesIFX

3.2.3.3 Imote2

El Imote2 es un avanzado nodo sensor inalámbrico fabricado por Crossbow Technology Inc. Está construido alrededor de la CPU XScale PXA271 de baja potencia y también integra una radio IEEE 802.15.4. El diseño es modular y ampliable con conectores para tarjetas de expansión tanto en el lado superior como en el inferior.

Figura 3.7: Nodo sensor Imote2

3.2.3.4 Iris

El nodo Iris es un nodo sensor fabricado por Crossbow Technology Inc, está diseñado específicamente para sistemas embebidos. Cuenta con una radio de 2,4 GHz (IEEE 802.15.4) de baja potencia con una capacidad de transmisión de 250 Kbps. Además tiene conectores de expansión para sensores de luz, temperatura, humedad, presión barométrica, etc. El mote IRIS tiene nuevas funcionalidades que mejoran la funcionalidad general de los productos de redes de sensores inalámbricos.

Figura 3.8: Nodo sensor Iris.

3.2.3.5 Mica2

El mote Mica2 (Crossbow Technology Inc) pertenece a la tercera generación de módulos para redes de sensores. El nodo Mica2 tiene nuevas mejoras sobre el original mote MICA. Cuenta con un transmisor-receptor multicanal (868/916 MHz). Cada nodo tiene capacidad para actuar como enrutador en las comunicaciones inalámbricas, además cuenta conectores de expansión para todo tipo de sensores (luz, temperatura, humedad, etc.).

Figura 3.9: Nodo sensor Mica2

3.2.3.6 Mica2Dot

El nodo Mica2Dot (Crossbow Technology Inc) también pertenece a la tercera generación de nodos sensores. Mica2Dot es similar a Mica2, a excepción de su tamaño que es una cuarta parte y en forma de moneda. Trae integrado un sensor de temperatura, LEDs y un monitor de batería. A destacar su radio multicanal (868/916 MHz, 433/315 MHz).

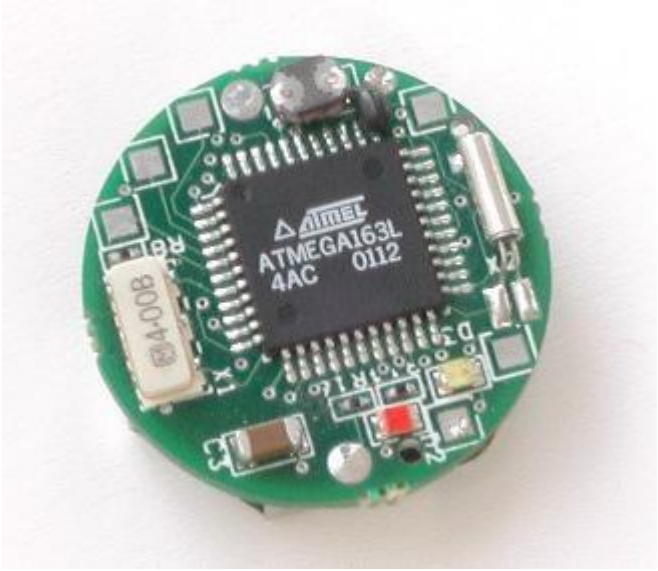

Figura 3.10: Nodo sensor Mica2Dot

3.2.3.7 TinyNode

El TinyNode es un módulo construido específicamente para funcionar bajo el sistema operativo TinyOS. Fabricado por la empresa suiza ShockFish. Su diseño esta preparado para funcionar a ultra baja potencia, lleva incorporado un sensor de temperatura, ademas se puede integrar una amplia gama de sensores y actuadores. Utiliza el microprocesador MSP430 de Texas Instruments con radio multicanal de baja potencia.

Figura 3.11: Nodo sensor TinyNode.

3.2.3.8 eKo

Este modelo de última generación es un dispositivo robusto, de mayor tamaño, diseñado para estar al aire libre y alimentado por energía solar. Cada nodo puede alojar

hasta 4 diferentes tipos de sensores. El nodo eko (Crossbow Technology Inc) integra la plataforma Iris con baterías recargables y una célula solar. Los nodos vienen pre-programados y pre-configurados para formar una red y requieren aproximadamente 1-2 horas por día de exposición al sol para mantener sus baterías cargadas. La carcasa cuenta con un soporte de metal en la parte trasera para que pueda ser fijado con total seguridad.

Figura 3.12: Nodo sensor eKo

3.2.4 Nodos utilizados en este proyecto

A continuación se procede a describir con un mayor nivel de detalle los nodos sensores con los que se ha trabajado para el desarrollo del sistema de seguimiento de vehículos.

3.2.4.1 MicaZ

El mote MicaZ (Crossbow Technology Inc), utilizado en este proyecto, pertenece a la misma familia de nodos sensores que Mica, Mica2 y Mica2Dot. Todos ellos ofrecen características similares.

Incorpora un microprocesador ATMEL Mega128L de 7 MHz. Dicho procesador incluye una pequeña memoria EEPROM de 128 KB dedicado a programas, una memoria RAM de 4 KB para datos.

Además, dispone de una memoria flash externa de 512 KB para aquellas aplicaciones que necesiten guardar datos de manera permanente.

El microprocesador se conecta, a través de un Bus SPI, con la radio CC2420, fabricada por Texas Instruments, con una tasa transferencia de datos de hasta 250 Kbps.

Todo ello a través de la interfaz 802.15.4 compatible con ZigBee. Según la documentación del fabricante la radio cuenta con una cobertura de 75 a 100 metros en espacios abiertos mientras que para interiores es de 20 a 30 metros. Los consumos de la radio son de 19,7mA en modo activo, 20µA modo inactivo y 1µA en modo hibernación.

Por otro lado, cada MicaZ posee un conector de expansión de 51 pines. En él se pueden conectar las placas de sensores, las placas programadoras y otros dispositivos de entrada/salida. Además incorpora una interfaz formada por tres diodos LEDs (rojo, amarillo y verde) que actúa como ventana sobre lo que ocurre durante el funcionamiento de un nodo. Las dimensiones del nodo MicaZ son 58x32x7mm y pesa 18gr excluyendo las baterías (dos pilas AA). La batería de los motes deben proporcionan un voltaje entre 2.7 y 3.3 voltios. Según la documentación del fabricante los consumos del microprocesador son de 8mA en modo activo, mientras que en modo hibernación es menor a 15µA.

Figura 3.13: Nodo sensor MicaZ

La elección de este dispositivo radica en la facilidad con la que se le acoplan nuevas herramientas de medición como la placa sensora MTS310CB que se describe a continuación, la incluye un magnetómetro de dos ejes, elemento fundamental para la realización de este proyecto, además de la facilidad para leer datos y reprogramarlo desde un pc mediante una interfaz USB conectándolo a la placa programadora MIB520 que también se describe a continuación.

3.2.4.2 Placa Sensora MTS310CB

La MTS310CB (23) es una placa sensora flexible con una gran variedad de modalidades de detección. Estas modalidades incluyen un acelerómetro de dos ejes (Honeywell HMC1002), un magnetómetro de doble eje (ADXL202JE), luz, temperatura (Panasonic ERT-J1VR103J), zumbador y micrófono. Esta placa puede ser usada en plataformas Iris, MicaZ y Mica2.

Tiene un tamaño de 56x11x31mm y una de las características de esta placa sensora es que comparte el mismo canal conversor Analógico/Digital por lo que solo se puede activar un sensor a la vez, de lo contrario, la lectura en dicho canal será corrupta y no tendrá ningún sentido.

Con el objetivo de ayudar a minimizar el consumo de energía, estos sensores vienen equipados con mecanismos de control de la energía. Por defecto, el sensor estará apagado.

Figura 3.14: Placa sensora MTS310CB

3.2.4.3 Placa Programadora MIB520

La placa programadora MIB520 tiene dos funcionalidades básicas. Por un lado permite cargar código ejecutable en los motes y por otro puede funcionar como puerta de enlace entre los motes y el PC.

El MIB520CB, al igual que la placa sensora MTS310CB puede ser usada con las plataformas Iris, MicaZ y Mica2.

Esta placa programadora ofrece conectividad USB para la comunicación entre nodos y la estación base así como la programación de los nodos. Para funcionar como *gateway* la placa programadora necesita acoplarse a un nodo sensor, previamente programado para tal efecto, a través del puerto de 51 pines. La placa incorpora 3 LEDs (amarillo, verde y rojo) que nos indica si la placa está conectada a la corriente eléctrica

(verde) y si se están cargando datos en el mote acoplado desde el pc (rojo). También incorpora un interruptor para deshabilitar la transmisión en serie con los motes, y un botón para reiniciar la placa programadora. La placa se alimenta con 5 voltios y unos 50mA de corriente a través del cable USB conectado al PC.

Figura 3.15: Placa programadora MIB520

3.2.4.4 TelosB

El nodo sensor TelosB es un dispositivo desarrollado y distribuido por la Universidad de Berkeley. Incluye el chip radio Chipcon CC2420 que cumple con el estándar IEEE 802.15.4, microcontrolador MSP430 e interfaz USB para conectar a un PC. Las características principales de este dispositivo son:

- Transceptor compatible con IEEE 802.15.4.
- Compatibilidad global con las bandas ISM (2.4 hasta 2.4835 GHz).
- Velocidad de 250 kbps.
- Antena integrada en la placa base.
- Microcontrolador MSP430 de 8 MHz con 10 kB de RAM.
- Bajo consumo.
- Memoria flash externa de 1MB.
- Programación y recolección de datos por USB.
- Sensores varios integrados. Luz, temperatura y humedad.
- Opera con TinyOS 1.1.11 o superior.

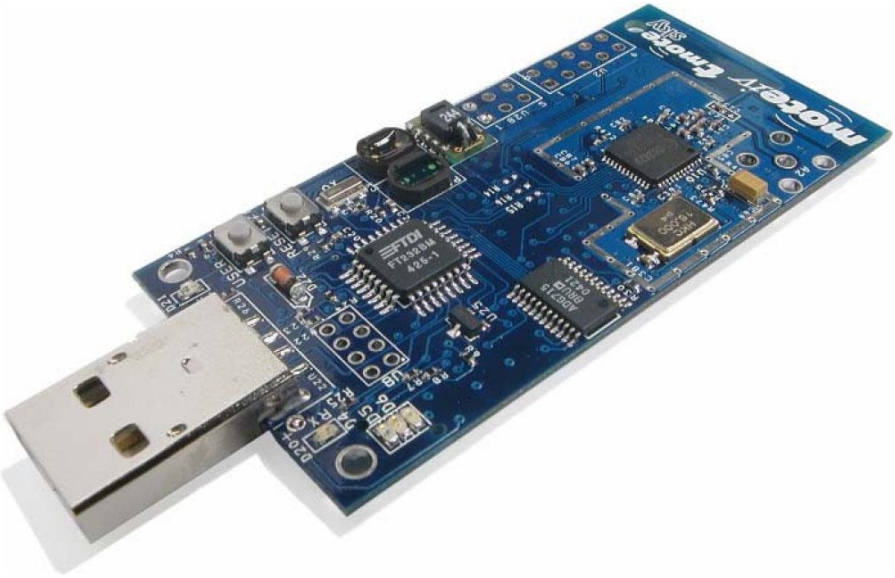

Figura 3.16: Modelo similar a TelosB usado en el desarrollo de la aplicación, el moteiv tmote sky.

La elección de esta plataforma frente a otras radica en que está especialmente diseñado para la experimentación y la investigación debido a su interfaz USB, que facilita la recopilación de datos a través de un bus conectado a un PC y al uso del sistema operativo TinyOS de libre distribución y diseñado para ser utilizado en dispositivos de bajo coste. Además de que es uno de los dispositivos que poseen un menor consumo dentro de la gran cantidad de nodos sensores existentes.

3.2.4.5 Comparativa entre los dos sistemas elegidos

En las siguientes páginas se incluyen dos tablas en la primera se recoge una comparación de las principales características entre los dos nodos sensores escogidos para el desarrollo del proyecto mientras que en la segunda se recogen las características más significativas de su sistema de comunicaciones:

Características	Crossbow MICAz	MoteIV Tmote Sky
Voltaje alimentación	2,1 – 3,6	2,1 – 3,6
Baterías	2xAA	2xAA
Memoria Flash (KBytes)	128	48
Memoria serial externa (KBytes)	512	1024
RAM (KBytes)	4	10
EEPROM (KBytes)	4	16
Comunicaciones Serie	UART	UART
Conversor Analógico Digital (ADC) (bits)	10	12
Conversor Digital Analógico (ADC) (bits)	_	12
Interfaz Usuario	_	USB
Otras Interfaces	Digita I/O, I2C, SPI	Digita I/O, I2C, SPI
Consumo corriente en modo activo MCU (mA)	8	1.8
Consumo corriente en modo "sleep" MCU (µA)	15	5.1
Rango de temperaturas (ºC)	-45 a 85	-40 a 85
¿Sistema Operativo integrado?	Sí, TinyOS	Sí, TinyOS
Dimensiones (mm)	58x32	66.04x32.766
Peso (g)	18	23

Tabla 3.1: Comparación hardware MICAz vs hardware Tmote Sky.

Características	Crossbow MICAz	MoteIV Tmote Sky
Banda de frecuencias (MHz)	2400 - 2483.5	2400 - 2483.5
Tasa de transmisión de datos (bitrate)	250	250
Potencia de transmisión (dBm)	-24 a 0	-25 a 0
Potencia de transmisión (mW)	0.03 a 1	0.03 a 1
Sensibilidad de recepción (dBm)	-90 (min) , -94 (nom)	-90 (min) , -94 (nom)
Alcance en exterior (m)	75 a 100	125
Alcance en interior (m)	20 a 30	50
Consumo de corriente recibiendo (Rx) (mA)	19.7	19.7
Consumo de corriente enviando (Tx) (mA)	17.4	17.4
Consumo de corriente en reposo (µA)	20	20

Tabla 3.2: Comparación radio MICAz vs radio Tmote Sky.

3.3 Sistemas operativos para nodos sensores

3.3.1 Generalidades

Los sistemas operativos para redes inalámbricas de sensores son en general menos complejos que los sistemas operativos de propósito general, ya sea por las necesidades específicas de las aplicaciones para redes de sensores, ya sea por las limitaciones en recursos en las plataformas hardware de las redes de sensores. Por ejemplo, los sistemas operativos para redes de sensores no son interactivos de la misma forma que los son los sistemas operativos para PCs. Es por esto que los sistemas operativos para redes de sensores no incluyen soporte para interfaces de usuario. Además, las limitaciones de recursos en términos de memoria y soporte hardware de

mapeo de memoria hace de algunos mecanismos como la memoria virtual bien innecesarios o bien imposibles de implementar.

El hardware para redes inalámbricas de sensores no es muy distinto de los sistemas embebidos tradicionales, siendo por tanto posible la utilización sistemas operativos para sistemas embebidos tales como eCos o uC/OS. Sin embargo este tipo de sistemas operativos son diseñados habitualmente con propiedades de tiempo real, que los sistemas operativos diseñados específicamente para redes de sensores no ofrecen.

3.3.2 Principales sistemas operativos para WSN

- Bertha Una plataforma de software diseñada e implementada para modelar, testear y desplegar una red de sensores distribuida de muchos nodos idénticos. Sus principales funciones se dividen en los siguientes subsistemas:
  - Administración de procesos
  - Manejo las estructuras de datos
  - Organización de los vecinos
  - Interfaz de Red
- Nut/OS: Es un pequeño sistema operativo para aplicaciones en tiempo real, que trabaja con CPUs de 8 bits. Tiene las siguientes funciones:
  - Multihilo
  - Mecanismos de sincronización
  - Administración de memoria dinámica
  - Temporizadores asíncronos
  - Puertos serie de Entrada/Salida

  Está diseñado para procesadores con los siguientes recursos:
  - 0.5 kBytes RAM
  - 8 kBytes ROM
  - velocidad de 1 MIPS CPU
- Contiki: Es un Sistema Operativo de libre distribución para usar en un limitado tipo de computadoras, desde los 8 bits a sistemas embebidos en microcontroladores, incluidas motas de redes inalámbricas.
- CORMOS: A Communication Oriented Runtime System for Sensor Networks, específico para redes de sensores inalámbricas como su nombre indica.
- eCos: (embedded Configurable operating system) es un sistema operativo gratuito, en tiempo real, diseñado para aplicaciones y sistemas embebidos que sólo necesitan un proceso. Se pueden configurar muchas opciones y puede ser

personalizado para cumplir cualquier requisito, ofreciendo la mejor ejecución en tiempo real y minimizando las necesidades de hardware.

- EYESOS: se define como un entorno para escritorio basado en Web, permite monitorizar y acceder a un sistema remoto mediante un sencillo buscador.
- MagnetOS: es un sistema operativo distribuido para redes de sensores o adhoc, cuyo objetivo es ejecutar aplicaciones de red que requieran bajo consumo de energía, adaptativas y fáciles de implementar.
- MANTIS (MultimodAl NeTworks In-situ Sensors)
- TinyOS: Sistema Operativo utilizado por TMote Sky y MICAz, por tanto empleado en este proyecto y del que se hablará en detalle en la próxima sección.
- t-Kernel: es un sistema operativo que acepta las aplicaciones como imágenes de ejecutables en instrucciones básicas. Por ello, no importará si está escrito en C++ o lenguaje ensamblador.
- LiteOS: Sistema operativo desarrollado en principio para calculadoras, pero que ha sido también utilizado para redes de sensores.

3.3.3 TinyOS

TinyOS es un sistema operativo especialmente concebido para funcionar en nodos que forman parte de una red inalámbrica de sensores. Su nombre puede descomponerse en dos términos, "*Tiny*", que hace referencia a su orientación hacia dispositivos de bajas prestaciones (diminuto en inglés), y OS acrónimo de sistema operativo en inglés.

Es un sistema operativo de código abierto basado en componentes para redes de sensores inalámbricas.

Figura 3.17: Emblema del sistema operativo TinyOS

Está escrito en el lenguaje de programación nesC como un conjunto de tareas y procesos que colaboran entre sí. Está diseñado para incorporar nuevas innovaciones rápidamente y para funcionar bajo las importantes restricciones de memoria que se dan en las redes de sensores. TinyOS está desarrollado por un consorcio liderado por la

Universidad de California, Berkley en cooperación con Intel Research y Crossbow Technology, y desde entonces ha crecido hasta convertirse en un gran consorcio internacional, la TinyOS Alliance.

Una pieza clave para la programación de los nodos de una red inalámbrica de sensores es usar un modelo de programación basado en eventos para poder soportar las características de concurrencia que estas redes necesitan en sus nodos. Esta característica debe convivir con la escasez de recursos de hardware del nodo, y por tanto surge la cuestión de cómo usar un modelo potente de programación concurrente sin perderse en la complejidad de muchas máquinas de estado que se transmiten continuamente eventos.

TinyOS, y su lenguaje de programación nesC se centran en estos retos.

TinyOS soporta modularidad y programación basada en eventos a través del concepto de componentes.

Un componente contiene funcionalidad relacionada semánticamente, por ejemplo, para manejar una interfaz de radio o para calcular rutas. Tales componentes reúnen la información requerida de estado en un *frame*, el código del programa para tareas normales, "*tasks*", y controladores de "*events*" y "*commands*".

Tanto los eventos como los comandos se intercambian entre los diferentes componentes. Los componentes se organizan jerárquicamente, desde los componentes a más bajo nivel cerca del hardware hasta los componentes de más alto nivel que componen en última instancia la aplicación real. Los eventos originados en el hardware pasan hacia arriba desde los componentes de bajo nivel hasta los componentes de altos nivel; los comandos, por otro lado, se pasan desde los componentes de alto nivel hasta los componentes de bajo nivel.

Por ejemplo, un componente *timer* de TinyOS proporciona una versión más abstracta que un simple temporizador de hardware. Es capaz de entender distintos comandos, "stop", "*startPeriodicAt*", "*startOneShotAt*" entre otros. Además puede lanzar un evento "*fire*" hacia otro componente, por ejemplo, un componente que lo encapsula sobre un temporizador de hardware.

La característica importante a destacar es que, quedándose con el paradigma basado en eventos, tanto los comandos como los controladores de eventos deben ejecutarse en un momento dado y tener previsto el fin de su ejecución, es decir, no pueden permanecer ejecutándose mucho tiempo. Se supone que deberían realizar ciclos de trabajo pequeños esporádicos. En particular, los comandos deben evitar bloquear o esperar una cantidad indeterminada de tiempo. Son una simple petición sobre la que

alguna tarea de un componente más abajo en la jerarquía tiene que actuar. De manera similar, un controlador de evento sólo abandona la información en el *frame* de su componente y planea la ejecución de una tarea para más tarde. También puede enviar comandos a otros componentes, o directamente propagar otro evento hacia arriba en la jerarquía.

El trabajo computacional real se realiza en las "*tasks*". En TinyOS las tareas también deben de tener un punto inicial y final, pero la diferencia con los eventos y comandos es que estas pueden interrumpirse por los controladores. La ventaja es doble: no se necesita una administración de la pila, y las tareas son atómicas con respecto a otras tareas. Aun así, por la virtud de ser lanzadas por controladores, las tareas son virtualmente concurrentes entre ellas.

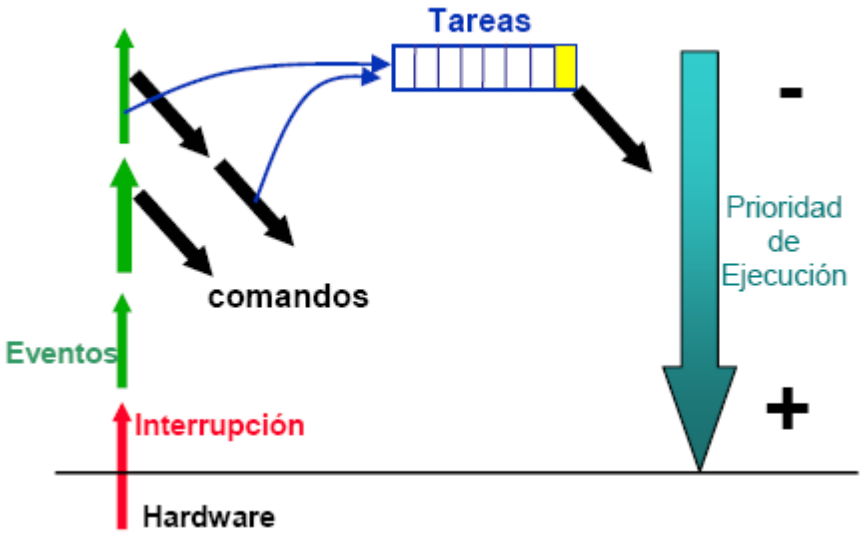

Figura 3.18: Esquema de funcionamiento de TinyOS

El arbitraje entre tareas (varias pueden lanzarse por varios eventos y estar listas para ejecutarse) se realiza mediante un planificador FIFO simple consciente de las restricciones de consumo de potencia del nodo, que apaga el nodo cuando no hay ninguna tarea ejecutándose o esperando.

Con las tareas y los controladores debiendo ejecutarse desde un principio y un fin, no está claro como un componente podría obtener información de otro componente sobre un comando que invocó allí. Por ejemplo, ¿cómo podría un protocolo ARQ saber a partir del protocolo MAC si un paquete se ha transmitido correctamente o no? La idea es dividir tales tipos de llamadas como una petición, y la información sobre las respuestas en dos fases distintas. En la primera fase se envía el comando, mientras en la

segunda se informa explícitamente sobre los resultados de la operación, entregados por un evento completamente separado del comando. Esta aproximación conocida como *split-phase programming* requiere que cada comando tenga un evento correspondiente que le proporcione los resultados que desea.

La aproximación permite la concurrencia bajo las limitaciones de la semántica *run-to-completion.* Si un comando no necesita ninguna confirmación ni ninguna información resultado de su trabajo, no se requiere que exista un evento que responda a su petición ulteriormente.

Tener comandos y eventos es la única manera de interactuar entre los componentes, ya que los *frames* de los componentes son estructuras privadas de datos, y especialmente cuando se usa programación *split-phase*, un gran número de comandos y eventos se añaden incluso en un pequeño programa. Por tanto, es necesario algún tipo de abstracción para organizarlos. Un componente puede hacer llamadas a otro componente más bajo jerárquicamente, y recibir eventos de él. Por tanto, los comandos y eventos se convierten en la interfaz de comunicación entre ambos. En algún sentido, cada uno de los dos componentes que se comunican tiene su propia interfaz, definida por los comandos que puede ejecutar, y por los eventos que puede lanzar.

El lenguaje de programación nesC formaliza esta intuición permitiendo al programador definir tipos de interfaces que definen los comandos y eventos conjuntamente. Esto permite expresar fácilmente el estilo de programación *split-phase* indicando en una misma interfaz cada comando con su correspondiente evento.

Los componentes pueden entonces proporcionar ciertas interfaces a sus usuarios y utilizar otras interfaces de los componentes inferiores.

Los componentes del estamento más bajo jerárquicamente y más cercanos al hardware se hacen llamar modules. Estos componentes primitivos pueden combinarse en configuraciones más grandes simplemente conectando las interfaces apropiadas. Para esta conexión, sólo los componentes que tienen los tipos de interfaces correctas pueden conectarse. El compilador se encargará de comprobar que es así.

Usando esta definición de los componentes, implementación, y conceptos de conexión, TinyOS y nesC forman juntos una base potente y relativamente fácil de usar para implementar tanto funcionalidades del núcleo del sistema operativo como pilas de protocolos de comunicación y funciones a nivel de capa de aplicación. La experiencia ha mostrado que, de hecho, los programadores utilizan estos paradigmas y llegan a componentes relativamente pequeños y altamente especializados que se combinan según se necesitan, probando por tanto las características de modularidad de que

TinyOS hace gala. Además de ello, el tamaño del código y los requisitos de memoria son bastante pequeños.

En general, TinyOS puede verse actualmente como la plataforma estándar de implementación para WSN. También se está incrementando su popularidad en otras plataformas diferentes de las originales motes para las que fue diseñado. En la práctica, la página web del proyecto proporciona una cantidad enorme de información muy valiosa, incluyendo tutoriales.

Encima del sistema operativo TinyOS se han desarrollado una inmensidad de extensiones, protocolos y aplicaciones.

3.4 Lenguajes de programación para nodos sensores

3.4.1 Generalidades

La programación de sensores es complicada, entre otras dificultades está la limitada capacidad de cálculo y la cantidad de recursos. Y así como en los sistemas informáticos tradicionales encontramos entornos de programación prácticos y eficientes para depurar código, simular, etc. en estos microcontroladores todavía no hay herramientas comparables.

3.4.2 Principales lenguajes de programación para WSN

Podemos encontrar lenguajes como:

- nesC. Lenguaje que está directamente relacionado con TinyOS y que se utiliza en este proyecto y por tanto se explicará con más detalle en la siguiente sección.
- Protothreads. Específicamente diseñado para la programación concurrente, provee hilos de dos bytes como base de funcionamiento.
- SNACK. Facilita el diseño de componentes para redes de sensores inalámbricas, sobre todo cuando la información o cálculo a manejar es muy voluminoso, complicado con nesC, este lenguaje hace su programación más fácil y eficiente. Es un buen sustituto de nesC para crear las librerías de alto nivel a combinar con las aplicaciones más eficientes.
- c@t. Iniciales que hincan computación en un punto del espacio en el tiempo (Computation at a point in space (@) Time).
- DCL. Lenguaje de composición distribuído (Distributed Compositional Language).

- galsC. Diseñado para ser usado en TinyGALS, es un lenguaje programado mediante el modelo orientado a tareas, fácil de depurar, permite concurrencia y es compatible con los módulos nesc de TinyOS.
- SQTL (Sensor Query and Tasking Language): como su nombre indica es una interesante herramienta para realizar consultas sobre redes de motas.

3.4.3 NesC

NesC (*Network Embedded Systems C*) es un lenguaje de programación basado en C, enfocado y optimizado para su uso en aplicaciones de redes de sensores. Su origen se debe al deseo de disponer de un lenguaje específico que cumpliese con el modelo de ejecución y los conceptos del sistema operativo TinyOS, el cual se describe a continuación. No fue hasta la versión 1.0 de TinyOS cuando se rescribió todo el código en nesC. En versiones anteriores (por ejemplo, la versión 0.6) de TinyOS todo el sistema operativo y la programación de aplicaciones fue desarrollada en C.

Esta necesidad de desarrollar un nuevo lenguaje de programación específico para redes de sensores inalámbricos viene motivada por el tipo de aplicaciones que se desarrolla, que se caracterizan por:

- Son aplicaciones basadas en recolección, difusión y control de la información obtenida del sensor, es decir, no son aplicaciones de propósito general.
- Tienen que reaccionar ante cambios en su entorno (eventos).
- Es preciso optimizar la limitada cantidad de recursos que ofrecen los nodos (memoria, capacidad de cómputo, consumo de energía).
- Deben ser aplicaciones estables, puesto que deben correr durante meses / años sin intervención humana.
- Precisan de control de errores en la manejo de datos.
- Son aplicaciones en tiempo real (envío de mensajes a la red).

Teniendo en cuenta estos aspectos, nesC aporta como concepto innovador la metodología de lenguaje orientado a componentes. Con respecto al lenguaje C tradicional, en nesC, se precisan restricciones teniendo en cuenta a las limitaciones de los nodos. Estas precondiciones estáticas del lenguaje, son la base de la optimización y permiten al compilador realizar análisis profundos sobre el código.

Además nesC permite desarrollar interfaces y componentes y estos últimos pueden ser a su vez módulos o configuraciones. A continuación se describen los constructores del lenguaje.

3.4.3.1 Interfaces

Cada componente suele implementar una serie de servicios que son ofrecidos al exterior mediante una interfaz. Las interfaces son bidireccionales, ofreciendo comandos y eventos, siendo básicamente en ambos casos funciones que son invocadas por la aplicación o por el hardware u otros componentes de nivel respectivamente.

Los comandos son llamadas a funciones del componente, tal y como puede ser "encender o parar un temporizador". Por otro lado, los eventos son notificaciones del componente, que indican que una tarea ha sido completada, tal y como puede ser la "confirmación del envío de un mensaje". A continuación se muestra un ejemplo de fichero interfaz:

```
interface Timer {
      command result_t start(char type, uint32_t interval);
      command result_t stop();
      event result_t fired();
}
interface SendMsg {
      command result_t send(TOS_Msg *msg, uint16_t length);
      event result_t sendDone(TOS_Msg *msg, result_t success);
      }
```

3.4.3.2 Componentes

Una aplicación en nesC está formada por varios componentes. Un componente implementa una serie de servicios que ofrece al exterior mediante una interfaz. Además, un componente puede usar las interfaces de otros componentes para su propio uso. Existen dos tipos de componentes:

3.4.3.2.1 Módulo

Los módulos contienen el código que implementa los servicios que ofrece el componente. En el siguiente ejemplo se puede apreciar la declaración y la implementación de los servicios. A continuación se muestra un ejemplo de un módulo:

```
module TimerM {
      provides {
            interface StdControl;
            interface Timer;
      }
      uses interface Clock;
      }
      implementation {
            command result_t Timer.start(char type, uint32_t
            interval){
                  ...
            }
```

```
	command result_t Timer.stop() {
		...
	}
	event void Clock.tick() {
		...
	}
}
```

3.4.3.2.2 Configuración

La configuración permite cablear ("*wire"*) o asociar diferentes componentes mediante interfaces para que trabajen en conjunto como si fueran uno solo (supercomponente), es decir, un componente de mayor abstracción. Para crear un programa en nesC, es preciso seleccionar los componentes y "linkarlos" a las interfaces que estos componentes proveen. Esta operación, se denomina "*wiring*". A continuación se muestra un ejemplo del fichero de configuración:

```
configuration TimerC {
	provides {
		interface StdControl;
		interface Timer;
	}
}
implementation {
	components TimerM, HWClock;
	// Pass-through: Connect our "provides" to TimerM "provides"
	StdControl = TimerM.StdControl;
	Timer = TimerM.Timer;
	// Normal wiring: Connect "requires" to "provides"
	TimerM.Clock -> HWClock.Clock;
}
```

El siguiente código muestra un ejemplo de cómo se realiza un "*wiring"*:

```
configuration EjemploAppC
{
}
implementation
{
	components MainC, EjemploC;
	EjemploC.Boot -> MainC;
}
```

En el fichero de configuración se indican los componentes a utilizar, uno de ellos puede ser el componente al que dicha configuración implementa, que en este caso es *EjemploC*. Por otro lado, *MainC* es el componente que pretende ser utilizado para esta aplicación. Por lo tanto debe "*linkarse*", y para ello es precisa la utilización de una interfaz, que en este caso es *Boot*. Es decir, que el componente *EjemploC* utilizará el componente *MainC* a través de la interfaz *Boot*. Para indicar este enlace se usa el operador "->".

Es necesario indicar en el fichero del módulo las interfaces que el componente, a implementar, va a utilizar para acceder a otros componentes, que en este caso es *Boot*. Por último, para utilizar un componente a través de un interfaz, dicho componente debe proveer dicha interfaz, siguiendo con el ejemplo:

```
module EjemploC(){
	uses interface Boot;
```

```
}
implementation{
      …
}
```

3.4.3.3 Modelo de concurrencia

Puesto que las redes de sensores inalámbricos poseen eventos asíncronos, que deben tener prioridad sobre otras tareas, se hace necesario el uso de un modelo de concurrencia que permita detener la ejecución de parte de un código menos prioritario, para atender un evento asíncrono de mayor relevancia. Si no se tiene en cuenta dichos eventos asíncronos se pueden producir las denominadas "condiciones de carrera" (o "*data races*"), en la que varios procesos intentan utilizar información compartida, haciendo que esta se pierda o quede corrupta por modificaciones descontroladas sobre los datos.

Las tareas ("*tasks*") en nesC son la solución a los problemas de compartición de información. Son funciones atómicas, que se ejecutan sin permitirse las interrupciones entre tareas, aunque sí por eventos. Cuando aparece un evento asíncrono se analiza y se pospone la ejecución de su código atómico, para seguir permitiendo la recepción de otros eventos.

3.4.3.3.1 Condiciones de carrera

Como en cualquier lenguaje de programación orientado a sistemas de tiempo real, nesC tiene mecanismos para detectar y prevenir las condiciones de carrera, se clasifican en código fuente en dos tipos:

- Código asíncrono: código al que se llega por una interrupción.
- Código síncrono: código al que se llega desde las tareas.

Si en tiempo de ejecución se intenta acceder a una variable compartida a causa de un evento asíncrono, se produce una posible condición de carrera que se debe analizar. Sin embargo, el código síncrono es atómico con respecto al código síncrono, es decir, que la ejecución de código síncrono, nunca se verá interrumpida por otro código síncrono y no se correrá el riesgo de condiciones de carrera.

En resumen, nos encontraremos con posibles condiciones de carrera únicamente ante la ejecución de código asíncrono. El compilador de nesC tiene constancia de este hecho, y gracias a la traza de las interrupciones es capaz de detectar las condiciones de carrera.

```
// Signaled by interrupt handler
event void Receive.receiveMsg(TOS_Msg *msg) {
      if (recv_task_busy) {
            return; // Drop!
      }
      recv_task_busy = TRUE;
      curmsg = msg;
      post recv_task();
}
task void recv_task() {
      // Process curmsg ...
      recv_task_busy = FALSE;
}
```

Para solucionar una situación de condiciones de carrera, se pueden plantear varias alternativas:

- Trasladar el acceso a las variables compartidas a tareas ("*tasks*").
- Usar secciones de código atómico, que indican al compilador que esa sección no puede ser interrumpida por un evento asíncrono.

Para alcanzar estas soluciones se pueden usar varios métodos que van desde la inhibición de interrupciones hasta el uso de semáforos, siendo la primera de ellas la que se usa en la actualidad. Hay que tener cuidado con el uso de estas técnicas ya que puede generar que se pierda información al no permitir la entrada de eventos asíncronos durante un largo tiempo.

3.4.4 Java

El objetivo de este apartado no es dar una visión completa y profunda sobre el lenguaje de programación Java, ya que esta fuera del alcance del presente proyecto. Se trata pues, de realizar un breve resumen de referencia debido a que en este proyecto el procesado de las firmas magnéticas obtenidas se realiza mediante este lenguaje de programación además de la comunicaciones entre los ordenadores que actúan como estaciones base en las calles y carreteras y los servidores que utiliza la aplicación móvil final y las comunicaciones entre los ordenadores y los sensores.

Java es un lenguaje de programación desarrollado por Sun Microsystems. Es un lenguaje orientado a objetos y toma mucha de su sintaxis de C y C++ pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel.

Las aplicaciones Java están típicamente compiladas en un "*bytecode"* y en tiempo de ejecución, el "*bytecode"* es interpretado o compilado a código nativo para su ejecución.

Desde noviembre de 2006 Sun Microsystems liberó la mayor parte de sus tecnologías Java se encuentran bajo la licencia GNU GPL de forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java aún no lo es).

La elección de usar Java para la implementación de la aplicación que comunica las estaciones se debe a que TinyOS se encuentra estrechamente relacionado con dicho lenguaje de programación (la mayoría de herramientas de apoyo de las aplicaciones de TinyOS están realizadas en Java) y que habiéndolo estudiado a lo largo de la carrera facilitaba el trabajo de programación.

3.5 XubunTOS

La manera más eficaz y eficiente de instalar en nuestro PC el sistema operativo TinyOS y todas las herramientas de programación necesarias para programar y compilar en nesC es mediante el sistema operativo XubunTOS y por tanto al final es el verdadero entorno en el que se ha realizado este proyecto por lo que se pasa a describir algunos de sus aspectos más importantes.

Figura 3.19: Logo de XubunTOS representando que ofrece una unión entre Xubuntu y TinyOS.

XubunTOS simplifica enormemente la instalación de TinyOS a través de un live CD de Linux. Nace de la unión de Xubuntu con TinyOS 2.x y a su vez Xubuntu es una versión más compacta de Ubuntu con el gestor de ventanas XFCE, que es más reducido que Kde o Gnome y deja así más espacio en el CD para el software propio de TinyOS.

De todos los métodos de instalación, distribuciones de Linux y versiones de Windows probadas sin lugar la mejor opción es XubunTOS. En primer lugar por la facilidad de instalación y en segundo porque una vez arranca la máquina virtual del CD el entorno ya está preparado para empezar a trabajar con los sensores.

Figura 3.20: Escritorio de XubunTOS.

Incluye tanto las versiones 1.x como la 2.x de TinyOS, y una serie de scripts para poder cambiar entre estas dos versiones de manera sencilla; facilidades para actualizar las fuentes de ambas ramas de TinyOS fácilmente; enlaces simbólicos a los dispositivos de programación USB; y, en definitiva, toda la comodidad de poder trabajar en un entorno Linux.

Además, el hecho de ser una distribución Live permite al usuario modificar los ejemplos una y otra vez porque cuando vuelva a arrancar lo encontrará todo como el primer día. Naturalmente, también incluye la opción de instalación permanente en disco si así se prefiere.

4 – Firmas Magnéticas

4.1 Introducción

El campo magnético de La Tierra es constante para cada determinado punto de su superficie. Además los materiales metálicos por el mero hecho de ser metálicos producen deformaciones en ese campo magnético apreciable, estas deformaciones en el campo magnético terrestre se denominan "firmas magnéticas" también conocidas por su término anglosajón "*magnetic signature*". Por tanto es fácil concluir que conociendo los valores puntuales del campo magnético terrestre en cada posición que se desee medir y monitorizando sus alteraciones es posible capturar la firma magnética de un determinado objeto metálico y detectar su posición y/o movimiento.

De esa misma manera los sistemas de defensa subacuáticos más modernos están preparados para detectar estas alteraciones en el campo magnético, estudiar su firma magnética y analizar si se trata de una amenaza para la seguridad como podría ser el acercamiento de un submarino enemigo o un torpedo dirigiéndose hacia él.

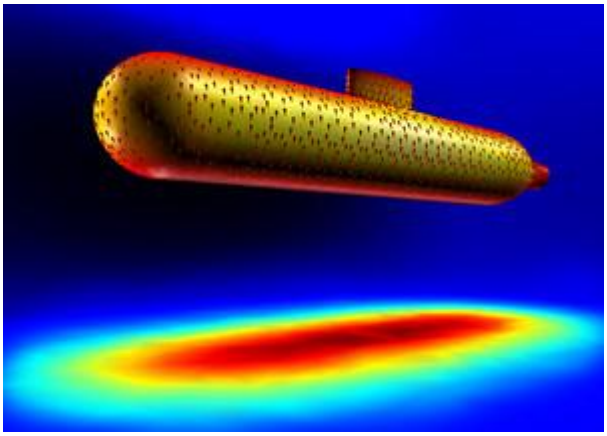

Figura 4.1: Proyección de la firma magnética de un submarino.

En este mismo aspecto se van a utilizar las firmas magnéticas en este proyecto pero aplicándose a términos de seguridad vial, es decir, utilizando la firma magnética de los vehículos para comprobar su posición en cada momento, es decir, realizar un seguimiento del vehículo.

4.2 Estudios Previos

Para comprender mejor los fundamentos que se pretenden aplicar en los siguientes apartados se realiza una recapitulación de los principales estudios que se han investigado para llegar a desarrollar este proyecto.

4.2.1 Estudio Universidad Berkeley

En la universidad de Berkeley se ha llevado a cabo un estudio sobre vigilancia del tráfico, esta vigilancia consistía en ser capaz de detectar vehículos mediante un sistema que utilizaba sensores magnéticos para detectarlos a través de su firma.

Este estudio consta de cinco experimentos independientes para analizar el tráfico local de California. El primer experimento persigue detectar el paso de vehículos por delante del sensor durante un determinado periodo de tiempo. El segundo trata de detectar vehículos estacionados en un aparcamiento, sin movimiento. El tercer experimento analiza las velocidades del tráfico de la zona mediante dos sensores magnéticos. El cuarto experimento intenta clasificar los vehículos locales detectados en varias categorías: motocicletas, automóviles, camiones, etc. basándose en las deformaciones en el campo magnético como es lógico un vehículo más voluminoso producirá una mayor deformación magnética. El quinto experimento trata de identificar vehículos dentro de un rango reducido de posibilidades mediante su firma magnética.

4.2.1.1 Experimento 1

Para el desarrollo de este experimento se utiliza un único nodo sensor situado en una carretera su funcionamiento consistía en monitorizar el campo magnético de la zona y detectar alteraciones, si durante un determinado tiempo se producía una alteración en el campo por encima de un umbral es que se había detectado un vehículo.

Para comprobar el funcionamiento del experimento se instalaron cámaras que grababan todo el tráfico. El posterior visionado de los videos de estas cámaras demostró que el porcentaje de acierto de este sistema de detección era del 98% habiendo sido capaz de detectar vehículos tan dispares como son motocicletas, automóviles utilitarios, monovolúmenes, camiones y autobuses. Por otra parte se demostró que este método de detección tiene una tasa de acierto mucho mayor que la de otros sistemas de detección como el lazo inductivo (“*inductive loop*”).

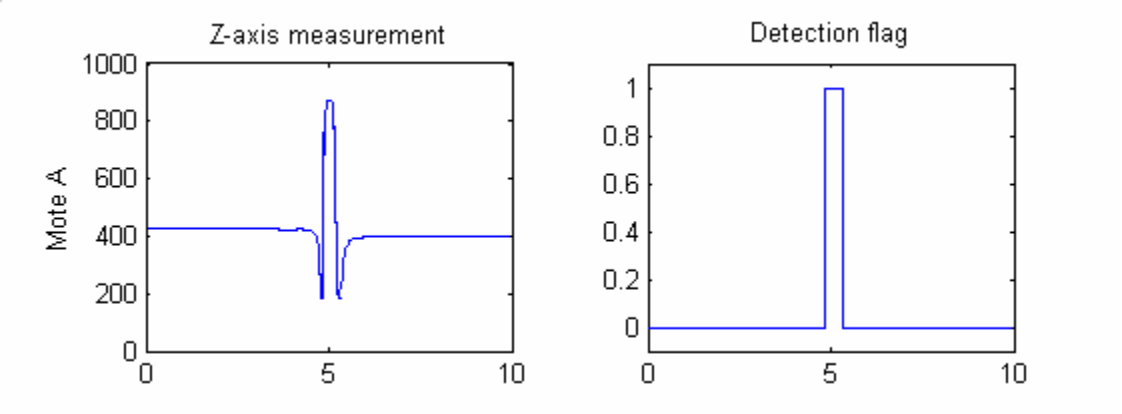

Figura 4.2: Detección de un vehículo por el sistema.

4.2.1.2 Experimento 2

De manera similar a la detección de vehículos del primer experimento trabaja este segundo experimento que trata de monitorizar vehículos en unas determinadas plazas de aparcamiento, si el valor del campo magnético se dispara por encima de un rango se ha detectado un vehículo, además en este caso el valor del campo magnético alterado se va a mantener durante un gran periodo de tiempo, porque el vehículo va a permanecer estacionado.

En este caso se detectaron también muy bien los vehículos pero se pretendía también descubrir si un vehículo aparcado en una plaza adyacente alteraría las mediciones de los sensores vecinos y se comprobó que no. Esto es debido a que el campo magnético decae rápidamente con la distancia.

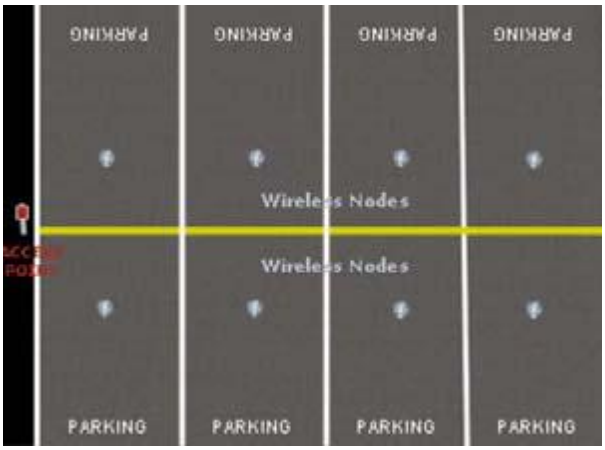

Figura 4.3: Esquema del parking monitorizado por sensores magnéticos.

4.2.1.3 Experimento 3

En el experimento 3 se intentan detectar las velocidades de los vehículos que pasan por un carril y la longitud de dichos vehículos para ello se sitúan dos nodos sensores iguales separados una distancia de seis pies (1,83 metros aproximadamente), por lo tanto conocido el espacio que les separa y detectando el tiempo que pasa entre la detección de un vehículo en un sensor y el siguiente se puede conocer su velocidad (mediante la relación: velocidad=espacio/tiempo). Una vez calculada la velocidad a la que va el vehículo, calcular su longitud es posible detectando el tiempo que se mantiene encima del sensor y despejando también de la sencilla ecuación anterior de la velocidad (espacio=velocidad*tiempo)

Para hacerse una idea de la fiabilidad de este sistema se instalaron dos conos de referencia en los puntos en los que se instalaron los sensores y una cámara para después calcular la velocidad de los vehículos mediante el procesado de los vídeos.

Sin embargo la velocidad de muestreo de la cámara era de 30Hz mientras que la velocidad de muestreo de los sensores es de 128Hz por lo que los resultado ofrecidos por los sensores eran mucho más fiables que los que la cámara obtenía.

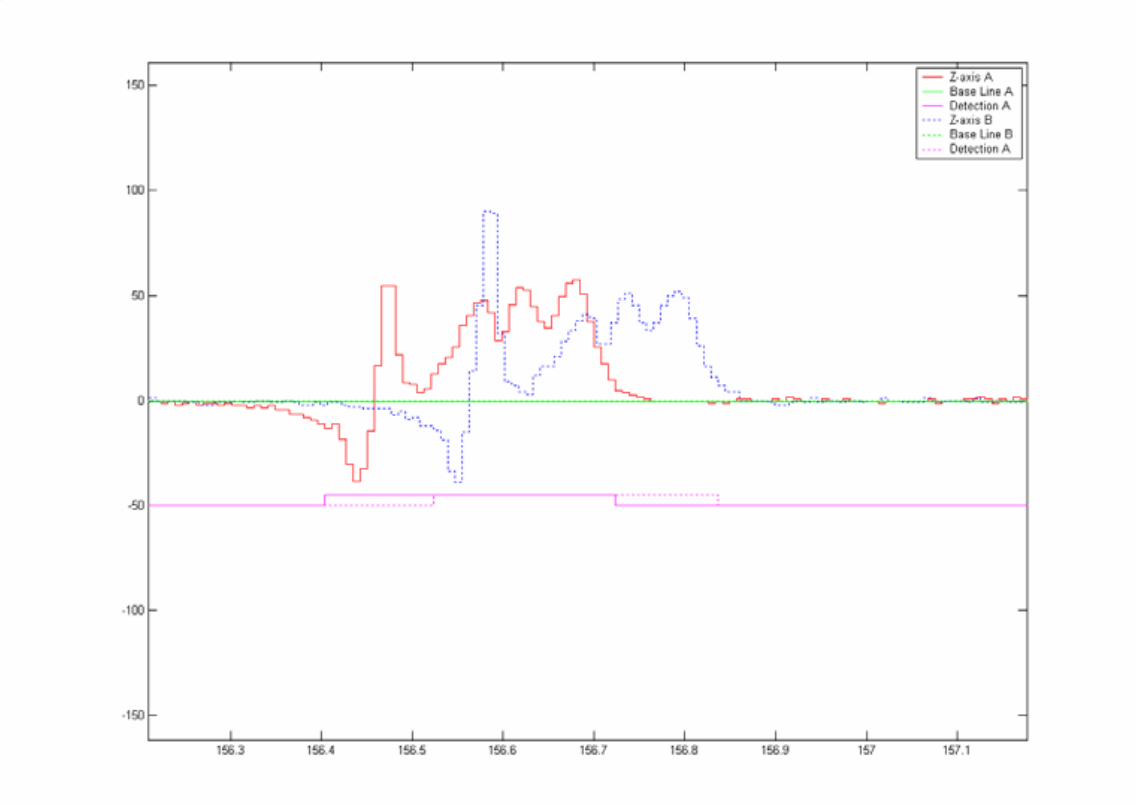

Figura 4.4: Firmas magnéticas detectadas por los sensores separados 6 pies.

4.2.1.4 Experimento 4

En el cuarto experimento se pretenden clasificar los vehículos en cinco tipos para ello es necesario que el sensor trabaja de una manera mucho más eficaz.

La firma magnética de los vehículos se recoge y se extraen dos bloques de información su longitud y su firma magnética en sí procesada mediante un método denominado "*Hill Patterns Classification*" que consiste en digitalizar las muestras pudiendo tomar solo tres valores 1 si está por encima de un determinado umbral, -1 si supera ese umbral negativamente o 0 si está entre los dos umbrales.

Como en los anteriores experimentos todo el desarrollo era monitorizado mediante cámaras de video para comprobar posteriormente su eficacia y rendimiento.

Tras la comprobación de las más de 4 horas de muestras de video se llegó a la conclusión de que este sistema clasificaba correctamente el 82% de los vehículos detectados, además se comprobó que sin el cálculo de la longitud del vehículo el porcentaje de acierto decaía solo un 2% situándose en un 80%

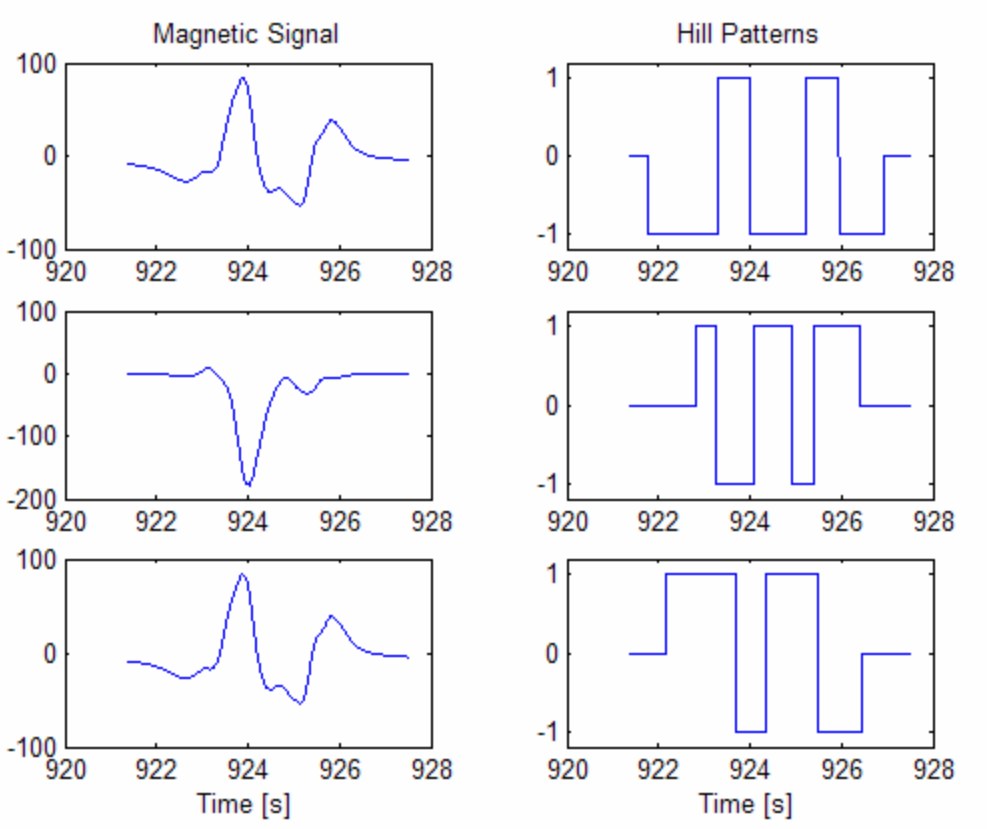

Figura 4.5: Ejemplos de muestreo de la firma magnética mediante el patrón de colinas.

4.2.1.5 Experimento 5

El experimento número 5 pretendía identificar correctamente el vehículo que pasaba por un carril independientemente de su velocidad teniendo tres posibles vehículos diferentes: Toyota Tercel, Toyota Camry y Ford Taurus.

Para ello en un carril se disponían dos nodos sensores que capturaban la firma magnética del vehículo que los sobrepasaba y calculaban su velocidad, en función de esa velocidad normalizaban la firma magnética y se comparaba con las firmas magnéticas registradas. Si la firma magnética registrada coincide en un alto porcentaje con alguna de las firmas almacenadas se declara que se ha detectado un vehículo concreto de los posibles.

5 – Sistema Desarrollado

5.1 Introducción

Tras los capítulos anteriores en los que se describía en profundidad el marco teórico que engloba este proyecto, los entornos de programación y las características hardware de los dispositivos sobre los que se ha trabajado, en este capítulo de la memoria se pretende describir en detalle el funcionamiento de la aplicación y del sistema en general desarrollado, mostrando el formato de los mensajes intercambiados entre los nodos sensores y el pc; y entre el pc y los clientes que se conectan a él para recibir la información de las firmas detectadas.

5.2 Descripción del sistema

Para el correcto funcionamiento del sistema de seguimiento de vehículos mediante la red de sensores inalámbrica hay una serie de elementos que son básicos e indispensables, como son los nodos sensores en sí mismos, el nodo sensor que hace de "*gateway*", y el pc que recoge y procesa la información de los nodos así como el servidor al que se conectan los clientes.

De manera general y sin muchos detalles específicos el funcionamiento paso a paso del sistema para el procesado y envío de firmas magnéticas al cliente sería el siguiente:

1. Los nodos sensores están monitorizando el campo magnético en las calles.
2. Cada cierto tiempo envían esa información recogida.
3. La información que envían los nodos sensores es recogida por un nodo inalámbrico "especial" que hace de puerta de enlace y envía la información a la estación base a la que está conectado.
4. La información que llega al pc estación base en un determinado formato es desencapsulada y procesada.
5. A la par está trabajando un servidor que se encarga de atender peticiones a través de internet y que tiene acceso a la información recogida
6. Dependiendo de la información solicitada por el cliente, el servidor envía unos datos u otros.

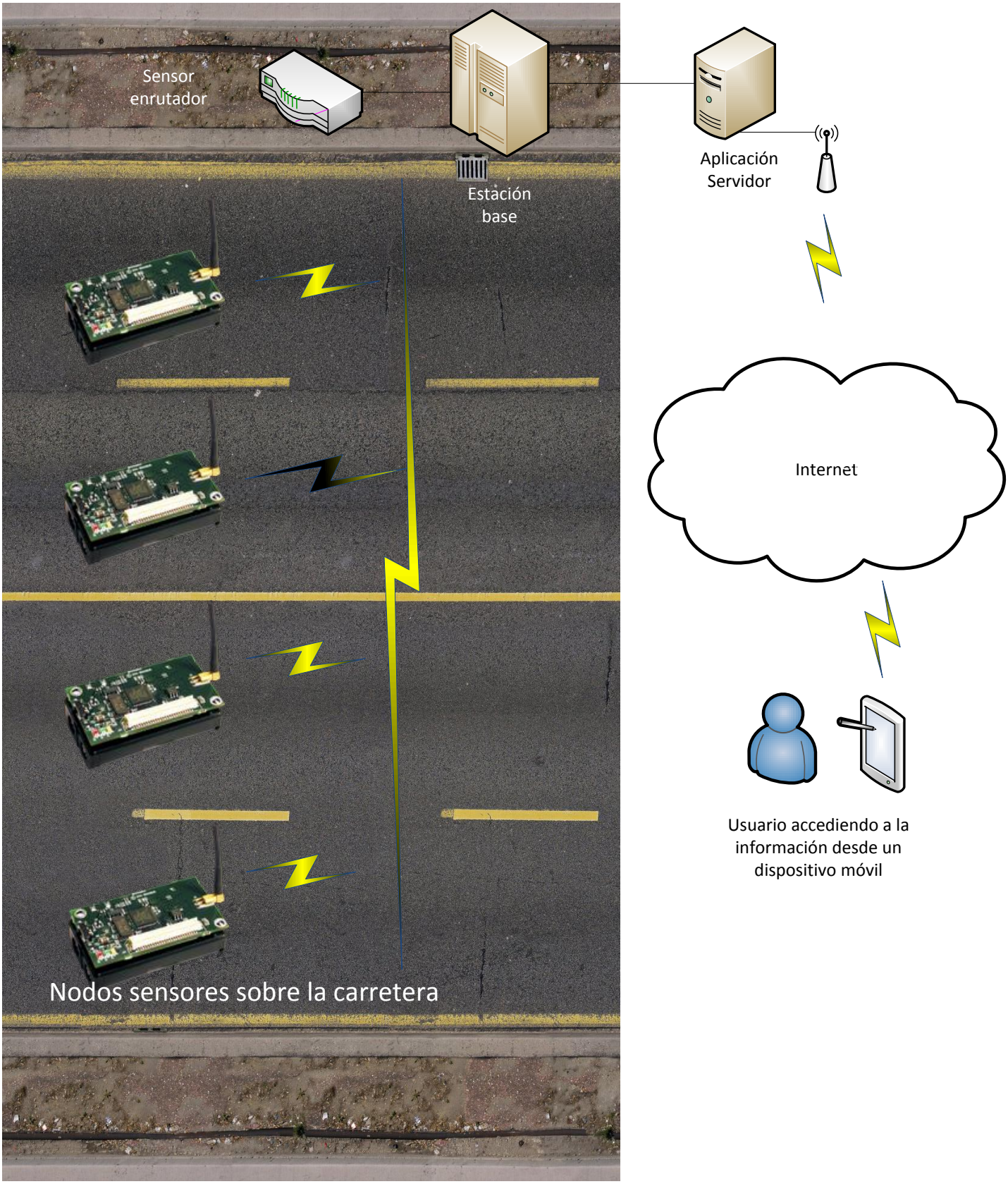

Figura 5.1: Esquema general del sistema se seguimiento de vehículos

Ahora tras está vista general del sistema de seguimiento de vehículos se puede proceder a describir el funcionamiento de cada uno de los componentes que lo forman con más detalle, recorriendo todo la estructura comenzando por donde se recogen las firmas hasta donde se envían, finalidad última de este proyecto.

5.2.1 Nodos sensores

La aplicación que se ejecuta en los nodos sensores como se ha comentado con anterioridad es una aplicación escrita en nesC que corre sobre el sistema operativo para nodos sensores TinyOS.

El funcionamiento de la aplicación es bastante sencillo ya que el grueso del procesado se lleva a cabo en el PC estación base por los requerimientos necesarios de hardware para llevarlo a cabo.

De manera sencilla se puede decir que los sensores inalámbricos se encuentran continuamente monitorizando el campo magnético, a una frecuencia de muestreo modificable, y cada cierto intervalo de tiempo encapsula esa información en un paquete y se la envían al nodo sensor que se encuentra conectado físicamente a la estación base para que pueda ser procesada.

Con mas detalle cabe decir que cuando los nodos sensores arrancan estos establecen su ID de nodo y esperan recibir su periodo de muestreo correspondiente. Una vez lo saben lanzan un temporizador, y durante el tiempo que les indica ese temporizador están continuamente muestreando el campo magnético hasta un máximo de muestras establecido y modificable.

Una vez finalizan encapsulan esta información derivada del muestreo del campo y se envía por la radio al nodo que hace las funciones de puerta de enlace nodos-pc.

Además si en algún momento un nodo sensor recibe un mensaje proveniente del pc lo analizará y verá si debe modificar su frecuencia de muestreo.

Para el funcionamiento de este sistema se implementan una serie de interfaces de unos determinados componentes, como son la interfaz "*Boot*" del componente "*MainC*" utilizada para el arranque de los dispositivos, la interfaz "*RadioControlC*" del componente "*ActiveMessageC*" utilizada para el control de la radio, la interfaz "*AMSendC*" del componente "*AMSenderC*" para enviar mensajes a través de la radio, la interfaz "*Receive*" del componente "*AMReceiverC*" para recibir mensajes de la radio, la interfaz "*Timer*" del componente "*TimerMilliC*" para poder usar el temporizador, la interfaz "*Leds*" del componente "*LedsC*" para mostrar a través de los leds el estado de la

aplicación (enciende el led0 cuando ha ocurrido algún problema, led1 cuando se ha enviado un paquete y led2 cuando se recibe un paquete) y por último las interfaces principales para el monitoreo del campo, la interfaz "*Mag*" para ajustar la ganancia y la interfaz "*Read*" del componente "*MagXC*" que devuelve los valores que lee el sensor magnético Honeywell de la placa sensora MTS310CB.

5.2.1.1 Formato de mensajes intercambiado entre nodos y estación base

Como es lógico para que los mensajes que se envían entre los nodos sensores y la estación base se puedan interpretar correctamente por ambas partes es necesario que el formato de los mensajes sea conocido por ambas partes, dicho formato es el siguiente:

version	interval	id	count
readings			

- version: El campo "version" es un entero sin signo de 16 bits, indica si hay alguna actualización en las condiciones de funcionamiento del programa, por ejemplo que se haya modificado la frecuencia de muestreo. El nodo sensor y el pc deben tener el mismo valor en este campo.
- interval: Este campo es un entero sin signo de 16 bits e indica el intervalo de muestreo para el sensor.
- id: En este campo se incluye la ID del nodo concreto que genera o al que va dirigido el mensaje.
- count: El campo "count" es otro entero de 16 bits sin signo y mantiene actualizado el número de secuencia de las muestras por las que se va recogiendo para que no haya problemas entre pc y sensor.
- readings: Es una array de enteros de 16 bits de un tamaño prestablecido en el archivo de configuración en el que se envía en sí la información del campo magnético muestreado.

Este mensaje es enviado a través de la interfaz radio para ello primeramente se encapsula en otro mensaje de nivel inferior en el que se incluye la cabecera necesaria para el control de errores y se trata a todo el mensaje encapsulado desde el punto de vista del nivel inferior como datos.

5.2.2 Nodo puerta de enlace

El trabajo que realiza este nodo es muy simple y a la par muy importante, es el encargado de llevar a cabo las comunicaciones mote → pc y pc → mote.

Para ello implementa dos interfaces diferentes de comunicaciones. Como todo sensor inalámbrico, implementa una interfaz radio para comunicaciones "*wireless*" pero además implementa una interfaz cableada mediante la cual se conecta al pc estación base.

El código que se ejecuta en este nodo es el de la aplicación "*BaseStation*" una útil aplicación proporcionada en TinyOS cuya funcionalidad es precisamente la de hacer de puerta de enlace entre la red inalámbrica y el pc, enviando cada mensaje que le llega por su interfaz radio a través de su interfaz serie, en este caso USB y viceversa, todo lo que le llega por la interfaz USB lo retransmite a la radio.

Por eso uno de los motivos de la elección del nodo Moteiv Tmote Sky como nodo estación base fue el que implementaba en sí mismo una interfaz radio y otra USB sin necesidad de conectarlo a ninguna otra placa. En la siguiente figura se puede le apreciar trabajando como nodo puerta de enlace:

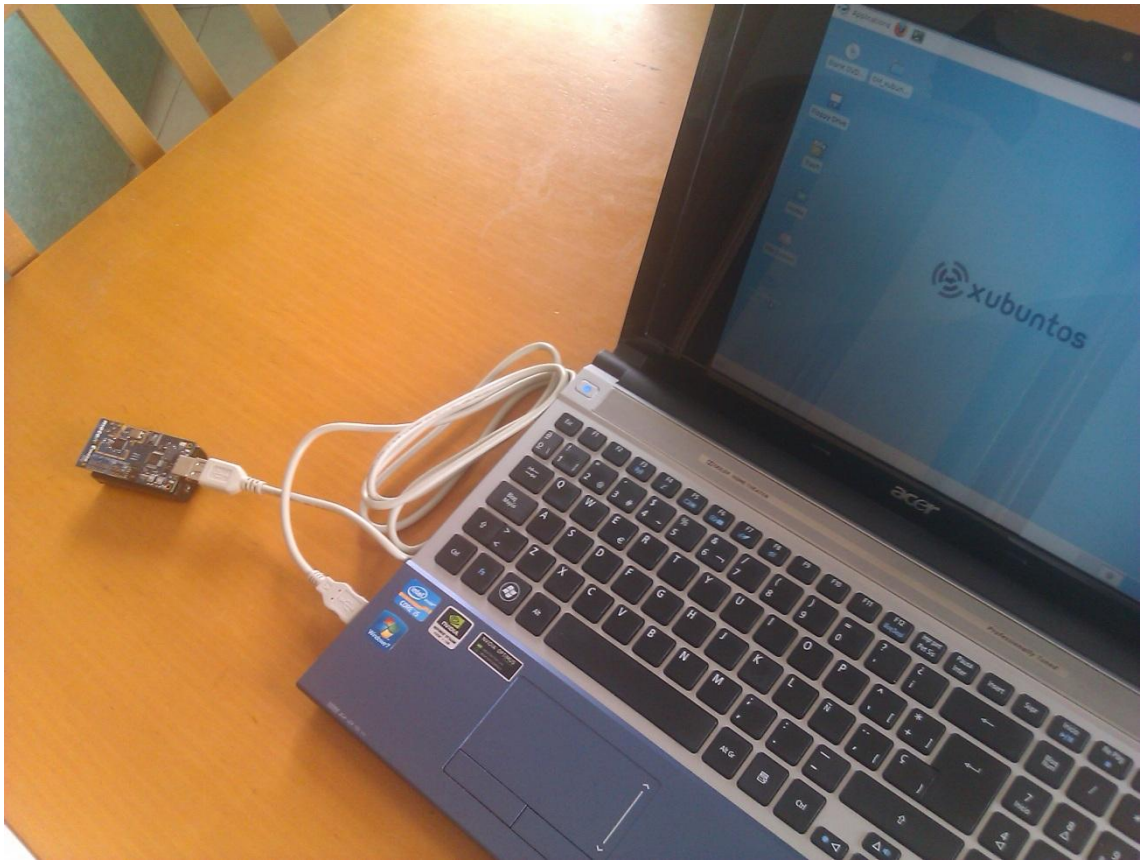

Figura 5.2: Nodo Moteiv Tmote Sky funcionando como puerta de enlace en alguna de las pruebas realizadas.

5.2.3 Estación base (PC)

En el ordenador que trabaja como estación base es donde se ha decidido implementar la mayor parte del código desarrollado para el sistema de seguimiento. Para ello se han realizado una serie de clases escritas en Java y que se ejecutan sobre el sistema operativo Xubuntu instalado en el ordenador. Por tanto en este apartado se describirá con más detalle que hasta ahora el funcionamiento de las aplicaciones incluyendo diagramas de flujo.

Para la comunicación entre las aplicaciones creadas y el sensor que está corriendo la aplicación base para retransmitir la información a los nodos hace falta un "intermediario" que reenvíe los paquetes recibidos por el puerto USB a un puerto virtual esta aplicación es SerialForwarder.

5.2.3.1 SerialForwarder

Esta utilidad tiene la misión de ligar un puerto de comunicaciones como el COM o el USB, con un puerto TCP del ordenador, de manera que sirve de pasarela entre los dos. Es una herramienta muy útil que facilita realizar una aplicación de pc que permita enviar por TCP datos, y estos datos sean recibidos por el SerialForwarder y reenviados al puerto que se le haya especificado y viceversa, es decir, todos los paquetes que sean enviados por el sensor al puerto que esta escuchando el SerialForwarder serán retransmitidos con todas las conexiones activas que haya en el puerto TCP en el que se encuentra escuchando.

Por defecto, posee el puerto COM1 y el puerto TCP 9001 pero se pueden cambiar en cualquier momento. Esta aplicación se encuentra en el directorio /tools/java/net/tinyos/sf/ y su invocación se hace mediante la maquina virtual de java.

En la siguiente imagen se puede apreciar una captura de la interfaz gráfica de la aplicación SerialForwarder, donde se nos permite elegir tanto el puerto virtual como la interfaz física por la que se comunicarán las aplicaciones así como el modelo del nodo conectado a la interfaz para ajustar la velocidad.

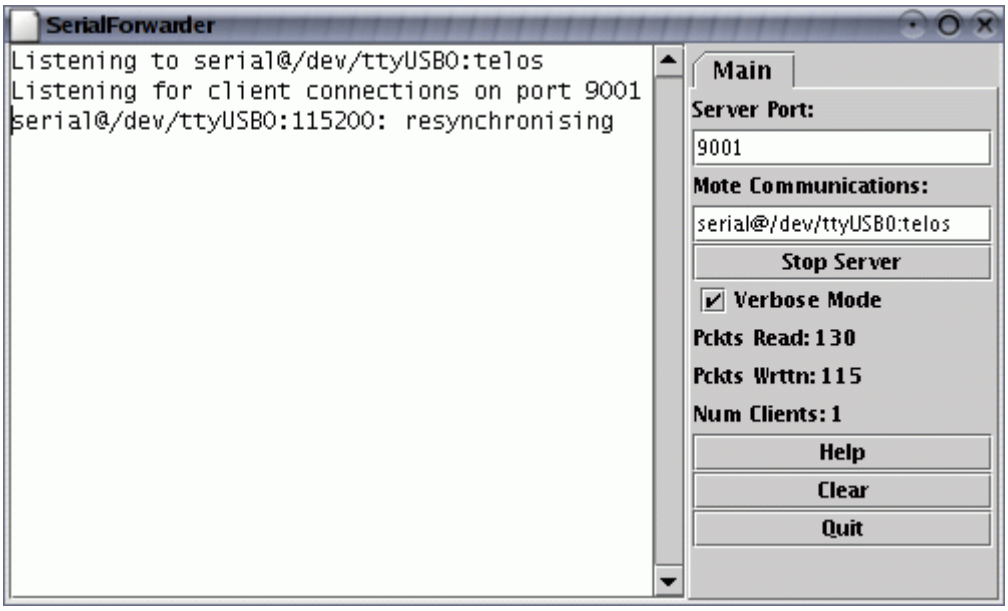

Figura 5.3: Captura de la interfaz gráfica de usuario de la aplicación SerialForwarder.

5.2.3.2 Aplicación Receptora

Esta aplicación, programada en Java, implementa la interfaz MessageListener porque su principal función es recibir los mensajes provenientes de los nodos sensores que se realizan de forma asíncrona.

Implementa también como atributo de la clase un objeto de la clase "MoteIF" definida en TinyOS cuyos métodos "registerListener" y "send" son utilizados para recibir y enviar la información a los nodos sensores respectivamente.

También incluye otros atributos de nuevas clases creadas para el funcionamiento del sistema como son las clases java "Servidor" y "Signature" y que se describirán más adelante, solo anticipar que la clase "Signature" es la clase principal del sistema que contiene la información relativa a las firmas magnéticas detectadas y los métodos necesarios para su procesado.

Por tanto la aplicación lo primero que realiza es crear una instancia de cada una de las clases antes mencionadas, después llama al método "registerListener" de la clase "MoteIF" y espera a recibir un mensaje proveniente de los nodos.

Cada vez que esto ocurre se llama al método "messageReceived" que extrae la información del paquete recibido, actualiza la versión o el intervalo de muestreo y recoge los datos del campo "readings" que como se ha comentado tienen los valores registrados en el sensor magnético del nodo en un determinado tiempo.

La aplicación dispone de un gran buffer por cada nodo sensor que controla, estos búferes son cíclicos y generosamente grandes para evitar que se pierda alguna firma, cabe recordar que un vehículo a menor velocidad tardará más tiempo en sobrepasar el sensor y producirá una firma de mayor tamaño (en número de muestras) que si lo recorriera a una velocidad superior, por lo tanto es posible que en el intervalo de tiempo en el que se muestrean y se envían los valores del campo magnético no se haya concluido la firma magnética, bien sea porque el vehículo iba lento o porque el comienzo de la firma fue casi al final de ese periodo.

Por tanto para evitar que posibles firmas magnéticas sean obviadas o cortadas se utilizan búferes de un tamaño amplio en el que quepan gran cantidad de firmas. Además aun así puede que parte de una firma quede fuera del gran buffer por tanto el buffer es cíclico y en la próxima recepción en el comienzo del buffer se encontrará la parte final de la firma segmentada.

Así que cada cierto tiempo la aplicación envía parte de este buffer cíclico a un método implementado en la clase "Signature" que busca firmas, si cuando se llega al final del análisis de esa parte del buffer cabe la posibilidad de que haya comenzado otra firma y no se haya detectado completamente se guardará un índice con la posición del buffer donde puede que comience esa firma y en la próxima comprobación la parte del buffer cíclico que se enviará para su análisis comenzará donde indique ese índice.

5.2.3.3 Clase Signature

Esta clase, del conjunto de clases java que componen la aplicación es la más importante pues implementa los principales métodos necesarios para la obtención y procesado de firmas magnéticas.

Además esta clase guarda los datos de las firmas que se reciben en la aplicación receptora y que deben estar disponibles para el envío a través de la aplicación servidor, por tanto podría decirse que también actúa como un sistema de "memoria compartida" para los otros procesos.

Para cumplir todos estos objetivos la clase tiene una serie de atributos privados pero que deben ser accesibles por las otras clases como son el atributo tiempo que guarda el tiempo que tiene que estar enviando o comparando firmas dependiendo de la acción solicitada, booleanos que indican si se ha encontrado alguna firma nueva para

avisar a los hilos que están pendientes de ellas, un array que va actualizándose con los valores de cada firma nueva detectada, etc.

Como estos atributos de la clase son necesarios para el buen funcionamiento del resto de hilos que forman el sistema se han declarado una serie de métodos que "*Get*" y "*Set*" que permiten a estos otros hilos acceder a los valores de estas variables y modificarlos respectivamente. Por supuesto al usarse un modelo de "memoria compartida" con acceso desde diferentes aplicaciones se ha tenido en cuenta implementar un sistema de exclusión mutua que garantice la integridad de los datos. Los métodos que dan acceso a las variables han sido sincronizados por lo que sólo un hilo puede tener acceso a la vez a esos datos y otro no podrá acceder a ellos hasta que éste haya terminado.

A parte de estos métodos para controlar los atributos de la clase existen otros métodos que realizan funciones de vital importancia para el correcto funcionamiento del sistema, como son:

5.2.3.3.1 Método find_signatures

Este método es el encargado de rastrear los búferes que posee la aplicación receptora en busca de posibles firmas. Se ha determinado de forma teórica una serie de condiciones generales que se cumplen para todas las firmas magnéticas, en un determinado punto de la superficie terrestre la atracción magnética de La Tierra se mantiene contante por lo tanto conocido ese valor es fácilmente determinable cuando se producen deformaciones del campo, sin embargo hay muchos factores que pueden producir pequeñas variaciones como la temperatura.

Por lo tanto, es necesario establecer un umbral mediante el cual se diferencien posibles firmas magnéticas de simples perturbaciones en el campo, ésta por tanto sería la primera condición que busca este método para comenzar a buscar una firma que el valor del campo supere un definido valor variable para cada punto en el que se quiera implantar el sistema.

No solo esta condición es necesaria también debe mantenerse por encima de ese umbral periodo de tiempo mínimo además de estudios previos y experiencia práctica se ha determinado que las firmas magnéticas tienen una mayor deformación cuando las ruedas están sobre el sensor, los valores que se detectan son mayores en esos momentos, es decir, las firmas magnéticas por lo general tendrá forma de "valle" con dos zonas de valores más altos que decaen en el centro y sobre todo en los extremos.

Así que las otras condiciones que busca el método son que una vez superado ese umbral durante un tiempo mínimo fijado los valores de la perturbación del campo decaigan y vuelvan a crecer y mantenerse por encima del umbral durante otro tiempo mínimo para decaer por debajo del límite finalmente.

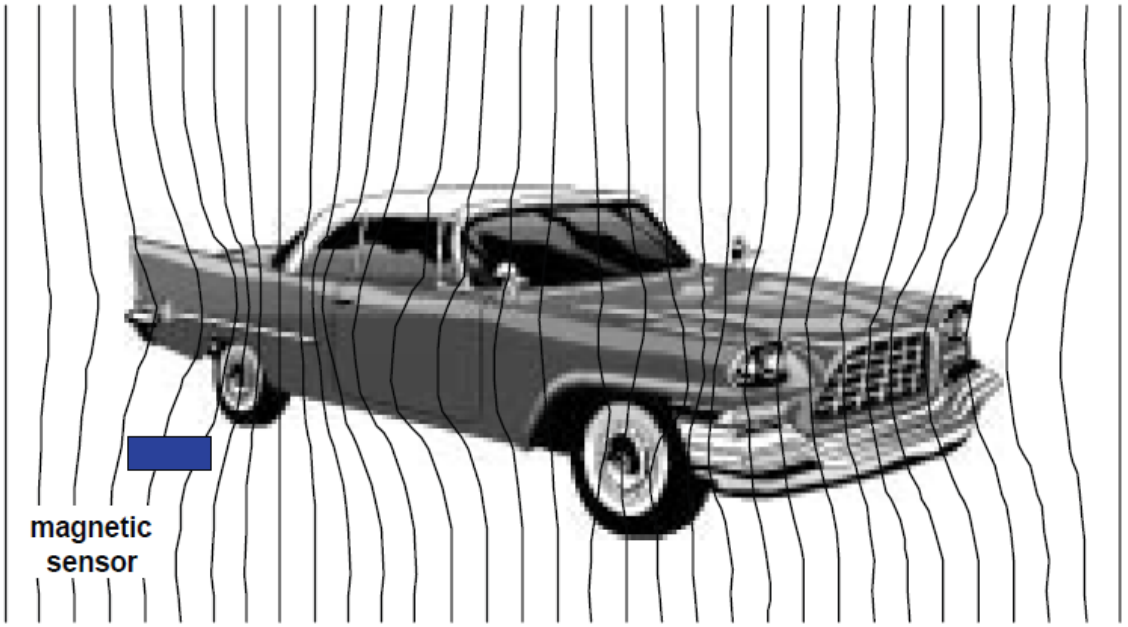

Figura 5.4: Deformación producida por un vehículo en la percepción del campo magnético terrestre.

Por tanto el funcionamiento del método es primeramente buscar en cada buffer recibido que se supere el citado umbral, después buscar que ese umbral esté siendo superado durante un determinado tiempo, si es así se activa la condición primera, una vez se cumple la condición primera busca que los valores recibidos bajen considerablemente si es así se cumple la segunda condición. Estaríamos ahora en la depresión del citado "valle", por lo tanto ahora el método buscaría que los valores posteriores volvieran a subir en módulo y se mantuvieran al menos durante un tiempo, se activaría entonces la tercera condición. La cuarta condición ya es simplemente que se acabe la firma, es decir, que los valores se mantengan por debajo del umbral de detección de firmas, si es así, se cumplirá la cuarta condición y se concluirá que se ha determinado una firma, guardándola en el array correspondiente.

Por supuesto para que se active una condición todas las anteriores deben cumplirse sin excepción. Además si durante el proceso en el que se busca una firma, es decir, mientras no se llega a cumplir la cuarta condición los valores que se están

registrando están por debajo del umbral más tiempo de un máximo se concluirá que no es la depresión del "valle" de la firma, si no que por alguna razón se habían cumplido las condiciones ya activadas pero no era la firma magnética de un vehículo y se procederá a anular todas las condiciones activadas y reiniciar la búsqueda de una firma, descartando todos los valores anteriormente analizados. Está condición sirve para evitar que otros objetos metálicos o perturbaciones en el campo por factores ambientales puedan provocar el registro de una firma magnética ficticia.

Por otra parte puede que en la fracción de buffer que ha recibido el método se encuentren indicios de una posible firma, es decir se encuentre valores por encima del umbral de detección pero se acabe la fracción de buffer recibida sin que sea posible determinar si lo que se había comenzado a detectar era una firma, por tanto esto quedará indicado y la próxima porción de buffer que se le pase para analizar comenzará justo por la zona donde comenzó a detectarse la posible firma.

5.2.3.3.2 Método normalizar

Cuando un vehículo se mueve produce una perturbación característica única en el campo magnético terrestre que como hemos visto ésta pueda ser detectada por un sensor magnético y servir como identificador para localizar un vehículo o distinguirlo de otro.

Sin embargo, esa firma magnética aun siendo exclusiva de cada vehículo y la misma puede variar en longitud dependiendo de la velocidad a la que el vehículo se desplaza por delante del sensor.

Un vehículo que viaja a una velocidad elevada producirá una deformación correspondiente a su firma magnética única pero de mucho menor tamaño que la que produciría el mismo coche moviéndose a una velocidad menor.

Por este motivo es necesario implementar un procedimiento que sea capaz de conseguir la firma magnética de un vehículo de un tamaño estandarizado para que pueda ser comparada con otras en las mismas condiciones.

Ésta es la labor del método normalizar al que se le pasa un array con los valores que se han obtenido de una firma y devuelve otro con el tamaño estándar que se haya elegido, este array contiene la misma firma que se le ha pasado pero ya normalizada a una longitud concreta para que pueda ser comparada y/o almacenada.

Para realizar esta función el método primero analiza el tamaño de la firma recibida si está por encima del valor del tamaño estándar calcula el número de muestras que debe eliminar y las sustituye por lo valores intermedios que debería tener si solo se hubiera muestreado una vez por cada rango de muestras eliminadas.

Si por el contrario lo que le llega es una firma de un tamaño inferior al acordado realiza, como es lógico el proceso inverso, calcula cuantas firmas debería insertar para llegar al tamaño estándar, las posiciones en las que debe incluirlo y a partir de las posiciones inmediatamente anterior y posterior calcula el valor que debería tener la muestra insertada.

Para terminar devuelve está firma normalizada al método que lo llamó para que pueda sustituirla por la firma estandarizada.

5.2.3.3.3 Método comparar_firmas

El método comparar_firmas es el que lleva a cabo la función principal del sistema que es seguir vehículos, para conseguir esto lo que hay que hacer es una vez extraída la firma magnética del vehículo que se pretende rastrear, enviarla al resto de estaciones base de la zona (de esto se encarga el hilo Servidor que se describe después) y que estás busquen si por alguno de sus nodos sensores se ha detectado esa firma.

Si esto es así, es decir si un nodo detecta la misma firma magnética que le ha sido solicitada quiere decir que ese vehículo acaba de pasar por ahí, y siguiendo las indicaciones del resto de nodos sensores se puede trazar la ruta que está realizando ese vehículo.

Por tanto para conseguir localizar un vehículo mediante su firma magnética lo primero que se tiene que poder hacer es saber comparar las firmas que se van obteniendo en el sistema.

Para realizar esto el método comparar_firmas actúa de la siguiente manera:

Recibe como parámetro la firma magnética que va a ser comparada, esta firma debe estar normalizada para que pueda ser comparada en las mismas condiciones que las que se van detectando, por tanto lo primero se asegura de que la firma esté normalizada haciendo uso del método normalizar si fuese necesario.

Después realiza una correlación muestra a muestra de las firmas para un intervalo de confianza elegido el predeterminado es al 90% de confianza con el que se han obtenido resultados satisfactorios. El usar un intervalo de confianza y no comparar

una muestra con otra y si no coinciden exactamente descartar que son la misma firma se debe a que como se ha comentado anteriormente pueden haber números factores como los ambientales que distorsionen ligeramente los valores de la firma, además de que el método normalizar pueden incluir ligeros errores al aproximar el valor que debería tener la firma en la muestra que se modifica.

Tras comparar los valores muestra a muestra de las firmas si sigue cabiendo la posibilidad de que sean la misma se realiza una correlación de la deformación total del campo magnético que producen ambas firmas si la diferencia sigue estando dentro de los umbrales delimitados para ese intervalo prestablecido se concluye que la comparación ha sido exitosa y se ha encontrado la firma buscada.

Para finalizar, el método devuelve un booleano indicando el resultado de la comparación, aunque solo una de las correlaciones no haya entrado en el rango será motivo suficiente para devolver un "*false*" indicando que no son la misma firma o un "*true*" indicando que sí coinciden.

5.2.3.4 Clase Servidor

Esta clase es un hilo encargado de recoger las peticiones provenientes de los clientes de las aplicaciones móviles de los usuarios y tiene dos modos de funcionamiento principales: "enviar firmas" o "buscar firma".

Tiene como atributos el puerto en el que debe establecer el socket para escuchar peticiones, que se puede dejar por defecto o especificar uno concreto en el constructor, y un atributo de la clase Signature, el cual sí es imprescindible indicarlo en el constructor pues el hilo servidor necesita saber cual es la clase que contiene la información y a quién indicar que necesita enviar firmas o que le busque alguna en concreto.

Por tanto el funcionamiento del Servidor es el siguiente:

Una vez inicia establece un socket al cual se le pueden conectar clientes y pedirle diferentes modos de funcionamiento, él recibe el mensaje por parte del cliente en el que se le indica el modo de funcionamiento, el tiempo que tiene que estar funcionando (aunque también se puede dejar un valor por defecto) y la firma a encontrar si se tratase del modo "buscar firma". Desde ese momento el servidor va solicitando la información que necesita a la clase Signature, monitorizando si se recibe alguna firma nueva para enviarla al cliente o que sea comparada con la que ha recibido e informar al cliente en el caso de que se encuentre.

5.2.3.4.1 Formato de mensajes Cliente/Servidor (Clase PaqueteFirma)

La clase PaqueteFirma establece un objeto con los atributos y métodos necesarios para que las comunicaciones cliente/servidor se puedan llevar a cabo correctamente.

Los atributos necesarios para tal propósito son un String con el modo de funcionamiento que se desea solicitar, un entero en el que se indica el tiempo en el que se tiene que estar trabajando, ya sea el tiempo en el que se tienen que estar enviando las firmas o el tiempo máximo que se quiere estar rastreando una firma concreta. Para esto la clase PaqueteFirma incluye los métodos "Get" y "Set" necesarios para el acceso a sus variables así como el constructor para formar objetos de esta clase.

modo	tiempo	firma

5.3 Puesta en marcha

Ahora se pasa a describir el procedimiento necesario para compilar las distintas aplicaciones que componen el sistema.

Para comenzar primero habría que compilar la aplicación de recopilación de información magnética de los nodos sensores, para ello se tiene un archivo Makefile en la carpeta en la que se encuentra la aplicación que permite la compilación fácil de la aplicación. Habría que escribir: "make <dispositivo para el que se comipla>". Ejemplo:

```
make micaz
```

Si lo que se quiere es instalar ya la aplicación en un nodo sensor no solo habrá que especificar el dispositivo, si no la placa programada, el id del nodo y la interfaz de programación de la siguiente forma: "make <dispositivo> intall.<id nodo> <placa programadora>,<interfaz>". Ejemplo:

```
make micaz install.1 mib520,/dev/ttyUSB0
```

Por otro lado si se desean eliminar los archivos que se han creado tras la compilación se puede hacer fácilmente con la orden:

```
make clean
```

Por otro lado en el nodo que hace de puerta de enlace habría que instalar la aplicación BaseStation para retransmitir los paquetes por sus interfaces. Esta aplicación se encuentra en XubunTOS en "/opt/tinyos-2.1.0/apps/BaseStation" y también dispone de un archivo Makefile similar por lo que los comandos de instalación serían similares a los anteriores.

Además como se ha comentado anteriormente necesitamos lanzar la aplicación SerialForwarder para retransmitir paquetes del puerto serie a un puerto virtual y viceversa, esto se consigue con la orden:

```
java net.tinyos.sf.SerialForwarder
```

En la interfaz gráfica que nos aparece deberíamos indicar el puerto virtual, la interfaz serie y el modelo de mote conectado a esa interfaz para ajustar la velocidad.

Para finalizar habría que compilar y ejecutar la aplicación Java para ello nuevamente se dispone de un archivo Makefile que contiene las órdenes necesarias para compilar todas las clases. Para compilar ejecutaríamos simplemente el comando:

```
make
```

Y podríamos eliminar todos los archivos de compilación con la orden:

```
make clean
```

Sería conveniente mencionar que antes de fijar el sistema en una localización se debe analizar el efecto del campo magnético en esa zona pues aunque varía muy poco durante kilómetros, éste puede variar y hacer que haya que modificar los umbrales de detección de las firmas. Una forma de analizar el campo magnético sería activando alguno de lo sensores en momentos en los que no se produzcan alteraciones y ejecutar en el ordenador estación base la aplicación Listen que consiste en una herramienta java ofrecida por TinyOS y que muestra por pantalla los datos brutos recibidos por los sensores. Analizando los valores del campo “readings” se podría apreciar el valor del campo magnético y en función de estos resultados establecer los citados umbrales. Para lanzar la aplicación hay que ejecutar el comando:

```
java net.tinyos.tools.Listen
```

Tras seguir estas indicaciones la red de sensores inalámbrica debería estar funcionando y el sistema de seguimiento analizando firmas magnéticas.

5.4 Pruebas

En la sección que aquí comienza se va a describir el conjunto de pruebas que se han realizado para la comprobación del buen funcionamiento del sistema, tanto del código creado como del sistema en su totalidad.

5.4.1 Pruebas del código

Para comprobar el funcionamiento adecuado del código creado para el sistema se han insertado en él, paquetes controlados, en los que se conocía su contenido, tanto

por la parte de las comunicaciones con la red inalámbrica de sensores como por la parte de la comunicación con los clientes de los dispositivos móviles.

En cuanto al control de los paquetes de la red, se han insertado paquetes que contenían firmas, porciones de firmas y alteraciones del campo que no deberían ser tratadas como firmas. La aplicación ha sabido reaccionar correctamente ante estos datos de entrada guardando las firmas de los primeros, ante el segundo tipo de paquetes guardando un índice donde comenzaba la posible firma para analizar si realmente lo es cuando se tenga la información los siguientes mensajes y ante el tercer tipo de paquetes comprobando que no eran firmas magnéticas aunque los valores que superaban los umbrales y descartando estos datos.

Para constatar el funcionamiento completo del código atendiendo a clientes se controlaban los paquetes que podían llegar por los dos puertos de comunicaciones.

Al igual que en el caso anterior se insertaban paquetes en el sistema con valores conocidos del campo magnético, por tanto se sabía si contenían firmas magnéticas o no y cuales eran sus valores para cada muestra.

Para controlar que todo el proceso se realizara correctamente se crearon dos nuevas clases, dos hilos que simulan un cliente de la aplicación móvil y también se controlaban los mensajes que se mandaban y la respuesta que se obtenía a ellos.

Así, se lanzó el cliente que solicitaba firmas durante un determinado periodo de tiempo, el sistema fue enviando correctamente estas firmas y el cliente las recibió correctamente mostrando los datos recibidos por pantalla.

Después se comprobó el apropiado funcionamiento del sistema comparando firmas, para tal propósito se conocía la firma que buscaba el cliente y también se conocían las firmas que iban llegando desde la red de sensores. Los resultados fueron los adecuados se iban descartando firmas que no coincidían y se avisaba al cliente cuando se encontraba que a su vez lo indicaba mediante un mensaje en pantalla. La comparación de firmas funcionaba incluso cuando las firmas no eran exactamente iguales pero estaban dentro del intervalo de confianza y también cuando alguna de ellas necesitaba ser normalizada.

6 – Conclusiones y Líneas Futuras

6.1 Conclusiones

En el desarrollo de este proyecto se ha indagado en las singularidades de una tecnología emergente que según el Instituto tecnológico de Massachusetts "revolucionará el mundo". Se ha explicado en detalle porque van a revolucionar el mundo gracias a sus características únicas y sus múltiples funcionalidades mostrando además los métodos, herramientas y sistemas disponibles para trabajar con ellas y crear aplicaciones propias.

Las redes de sensores inalámbricas como se ha explicado se utilizan para medir para medir factores del entorno en muy diversos campos del conocimiento, sin embargo en este trabajo se ha utilizado en un terreno mucho menos explorado como es el del campo y las firmas magnéticas.

Para poder desarrollar una aplicación que fuera capaz de detectar estás firmas magnéticas primero era conveniente, estudiar más en profundidad sus propiedades y así se ha hecho, describiendo unas generalidades sobre ellas y unos estudios realizados anteriormente pioneros en este campo.

Tras todos los estudios e investigaciones citados se ha procedido al desarrollo de un sistema de seguimiento de vehículos basándose en esos dos principales pilares, las redes de sensores inalámbricas y las firmas magnéticas y los resultados han sido muy satisfactorios.

Se ha conseguido diseñar un sistema capaz de discernir firmas magnéticas de simples alteraciones en el campo magnético producido por diversos factores ambientales y/o humanos. Gracias a esto ha sido posible por tanto crear un sistema de seguimiento de vehículos, que pueda reconocer la firma magnética de un vehículo concreto y buscarla o enviarla a un cliente.

6.2 Líneas futuras

A la conclusión de este proyecto se abren nuevas vías de investigación en forma de posibles mejoras para el sistema de seguimiento.

Podría crearse un protocolo de comunicaciones entre los nodos sensores, en el que cada uno de ellos enviara su información y la que le ha llegado al nodo "*gateway*" o a un nodo más cercano intermedio. Es decir que cada nodo sirviera como sensor magnético y además de puente para las comunicaciones entre la estación base y un nodo cercano a él pero más alejado de la estación. De esta forma se podrían reducir el número de estaciones base necesarias, eso sí debería conocerse muy bien el emplazamiento del nodo del que llega la información pues ya no tendría por qué encontrarse en un entorno más o menos cercano a la estación. Reduciendo el número de estaciones base se

reduciría sensiblemente el coste pues se ahorraría el gasto en un gran número de equipos y en el precio de la IP pública necesaria. Para este diseño habría que tener especial cuidado en que las comunicaciones mote → mote sean realmente rápidas pues en un sistema de seguimiento en el que se persigue a un coche que se da la fuga, prima la velocidad de las detecciones y un sobredimensionado de la red o unas comunicaciones lentas podrían provocar que cuando el vehículo fuera detectado fuese demasiado tarde para seguirle.

Otra posible mejora en el sistema de seguimiento de vehículos sería una mejora en la eficiencia energética, en este proyecto se ha dado prioridad a la eficacia del sistema, sin embargo, esto ha implicado un alto consumo energético por parte de los sensores. Para mejorar el rendimiento de los sensores alargando la vida de las baterías y también de los sensores pues como cualquier dispositivo tienen una determinada vida útil, podría diseñarse una serie de órdenes por parte del PC que indiquen a los sensores si hay algún cliente solicitando o buscando firmas, si la estación base indica que no hay clientes solicitando firmas podrían pasar a un modo de bajo consumo despertando de vez en cuando y levantando el transceptor para ver si tienen que seguir "durmiendo" o pasar a su funcionamiento habitual.

La última mejora sería mejorar el porcentaje de detección e identificación de los vehículos para ello se podrían incluir nuevas condiciones en el código que mejoraran su rendimiento pero sobre todo se podrían utilizar los tres ejes del campo magnético, el principal sería el eje del desplazamiento del vehículo como en este proyecto es donde se producen las mayores perturbaciones, pero en los otros ejes también se producen deformaciones únicas por cada vehículo que pueden ser aprovechadas para una mejor detección e identificación.

Dejando a un lado las mejoras que se pueden aplicar al sistema, este proyecto ofrece nuevas líneas de investigación para destinarlo a otras aplicaciones de la seguridad vial, por el modo en que se ha realizado el sistema total, separando la red de sensores y su procesado de las firmas de las aplicaciones de los clientes finales, estos clientes finales se podrían modificar sin tener que modificar el funcionamiento del sistema de detección e identificación.

Por ejemplo, en la ciudad de Cartagena para mejorar la congestión del tráfico en las horas puntas, los semáforos de las principales vías pasan en verde más del doble del tiempo habitual cuando no hay congestión esto hace que las calles perpendiculares a estos accesos pasen también el doble del tiempo habitual en rojo. Este sistema es eficaz a veces pero en otras no tanto pues consiguen no congestionar las principales vías incluso llegando al punto de estar en verde sin vehículos mientras que se acumulan los vehículos en cola en el resto de vías. Para mejorar esto se podría usar el sistema diseñado, mediante el cual se podrían detectar la cantidad de vehículos que transitan por unas vías y por otras y que los semáforos actuaran en consecuencia de una forma automatizada.

Otra aplicación de actualidad del sistema desarrollado, sería para el uso de radares, como se ha visto en el capítulo 4 de firmas magnéticas, dos sensores separados un par de metros pueden ser capaces de averiguar la velocidad a la que circula un vehículo. Pero con el sistema desarrollado esto se puede llevar un poco más allá, desde hace unos años atrás la DGT tiene la pretensión de instalar en las carreteras radares que sean capaces de determinar el tiempo que ha tardado un vehículo desde que entra a una vía hasta que sale de ella. De todos es sabido que en las zonas donde hay un radar de velocidad las velocidades medias se reducen, por tanto, ¿Qué ocurriría si los radares no midieran la velocidad instantánea en un determinado punto de la vía si no durante toda la carretera? La respuesta de la DGT a esta pregunta es clara se reducirían las velocidades a las que se circula por las carreteras y por tanto en gran parte el número de accidentes y vidas perdidas. Con el sistema de seguimiento diseñado esta pretensión sería posible pues sería capaz de detectar el momento en el que un vehículo entra en el túnel o vía y mediante la reidentificación otro nodo situado al final del túnel o en las salidas de la vía podría determinar el momento en el que sale. Si el vehículo ha utilizado menos del tiempo mínimo necesario que se necesitaría para realizar ese trazado el vehículo sería multado. Hasta ahora la DGT no había procedido a su implantación en las carreteras debido a que el sistema que se emplea implica un elevado coste económico, sin embargo el coste de este sistema descrito sería mucho menor y se podría proceder a su instalación.

7 Bibliografía

[1] Akyildiz I.F., Su W., Sankarasubramaniam Y., Cayirci E., "Wireless sensor networks: survey", Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA.

[2] Sorabi, K., Gao, J., Ailawadhu, V., and Pottie, J. "Protocols for Self-Organization of a Wireless Sensor Network" Electrical Engineering Department, UCLA Box 951594, Los Angeles, California, 90095-1594, USA.

[3] Gay D., Levis P., Culler D. Brewer E., "nesC Language Reference Manual".

[4] Levis P., Madden S., Polastre J., Szewczyk R., Whitehouse K., Woo A., Gay D., Hill J., Welsh M., Brewer E., Culler D., "TinyOS: An Operating System for Sensor Networks".

[5] Bharathidasan A., Ponduru V.A. S., "Sensor Networks: An Overview", Department of Computer Science, University of California, CA 95616, USA.

[6] "nesC: A Programming Language for Deeply Networked Systems", UC Berkeley WEBS Project.

[7] Hill J.L., "System Architecture for Wireless Sensor Networks".

[8] Web ZigBee Alliance. http://www.zigbee.org/

[9] Alcaraz Calero J.M., "Tutorial de TinyOS". Universidad de Murcia.

[10] Sohraby K., Minoli D., Znati T., "Wireless Sensor Networks: Technology, Protocols, and Applications" Wiley-Interscience.

[11] Chong C.Y., Kumar S.P., “Sensor Networks: Evolution, Opportunities, and Challenges”.

[12] Wang G.L., “Data Modeling for Wireless Sensor Networks” Rome B102, Building Experiment Center, East Campus, Sun Yat-Set University Guangzhou High Education Mega Center, Guangzhou, Guangdong, China.

[13] Datasheet TelosB. http://moss.csc.ncsu.edu/~mueller/rt/rt11/readings/projects/g4/datasheet.pdf

[14] Datasheet MicaZ. http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf

[15] Datasheet MIB520. http://bullseye.xbow.com:81/Products/Product_pdf_files/Wireless_pdf/MIB520_Datasheet.pdf

[16] Datasheet MTS310CB. http://retis.sssup.it/sites/retis.sssup.it/files/Sensor%20Board%20User%20Manual-MTS-MDA_Series_Users_Manual.pdf

[17] TinyOS Documentation Wiki. http://docs.tinyos.net/tinywiki/index.php/Main_Page

[18] Cheung S.Y., Ergen S.C. Varaiya P., “Traffic Surveillance with Wireless Magnetic Sensors”, University of California, Berkeley, CA 94720-1770, USA.

[19] Caruso M.J., Withanawasam L.S., “Vehicle Detection and Compass Applications using AMR Magnetic Sensors”, Honeywell, SSEC, 12001 State Highway 55, Plymouth, MN 55441 USA.

[20] Asin, A., “Wireless sensor platform eases quest for parking”, Libelium.

[21] Losilla López F., Garcia Sanchez A.J., Garcia Sanchez F., Garcia Haro J., Haas Z.J. “A comprehensive approach to WSN-Based ITS Applications: A survey”. Department of Information and Communication Technologies, Technical University of Cartagena, Campus Muralla del Mar, Cartagena E-30202, Spain.

[22] Santana-Diaz E. “A complete underwater electric and magnetic signature scenario using computational modeling”, Computational Mechanics BEASY, Ashurst Lodge, Southampton, Hampshire SO40 7AA, United Kingdom.