UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

"Diseño e implementación de un sistema de interconexión dinámica y monitoreo de sensores utilizando dispositivos compactos de arquitectura modular"

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRICISTA

MAURICIO ANDRÉS CONTRERAS HERNÁNDEZ

PROFESOR GUÍA:

EDUARDO VERA SOBRINO

MIEMBROS DE LA COMISIÓN:

JOSÉ MIGUEL PIQUER
NÉSTOR BECERRA YOMA

SANTIAGO DE CHILE
MARZO 2011

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELECTRICISTA
POR: MAURICIO CONTRERAS HERNÁNDEZ
FECHA: 30/03/2011
PROF. GUIA: SR. EDUARDO VERA

"Diseño e implementación de un sistema de interconexión dinámica y monitoreo de sensores utilizando dispositivos compactos de arquitectura modular"

El trabajo realizado buscó unir los campos de redes inalámbricas de sensores y dispositivos móviles. El primero dada la riqueza de información y facilidad de despliegue de estas redes. El segundo, en relación a las grandes ventajas de dispositivos portados por los usuarios (ej. información constantemente disponible), existiendo casos de uso que podrían beneficiarse de la funcionalidad conjunta (ej. telemedicina). El objetivo fue desarrollar e implementar un sistema que comunicara nodos inalámbricos, algunos de los cuales sensaran variables de interés, mientras que otros monitorearan las mismas.

El diseño se basó en la plataforma BUG, de BugLabs Inc., un *start-up* tecnológico de New York, USA. Esto pues se consideró de gran interés la filosofía propuesta por esta empresa: computadores compactos (posiblemente móviles) de arquitectura modular (el hardware se construye mediante la adición de módulos). El foco de esta plataforma es el prototipaje rápido de otros dispositivos.

La solución propuesta incluye Nodos Sensores y Monitores Móviles, ambos elaborados con BUGs pero con distintos módulos. Los Nodos Sensores cuentan con adquisición de señales análogas y también conexión USB. El mismo módulo es usado en los Monitores Móviles (por esta última funcionalidad), además de una pantalla que constituye la interfaz con el usuario. La comunicación inalámbrica entre nodos se realiza mediante módems externos. Estos módems (radios XBee) implementan el protocolo IEEE 802.15.4, que fue desarrollado especialmente para uso en redes de sensores. Sus características son bajo consumo de potencia y facilidad en el establecimiento de redes sin necesidad de otro hardware (redes ad-hoc). Se utiliza una topología en estrella, requiriéndose de un coordinador. Luego de simples configuraciones iniciales, la red se establece automáticamente y el monitor (o monitores) recibe muestras de los sensores conectados a la red, las cuales despliega en la pantalla y almacena en memoria. Debido a problemas con las versiones preliminares del hardware disponible, el sistema fue probado emulando los nodos en un computador convencional, pero utilizando una interfaz de aire real (se conectaron varios XBee al computador). Se definió un protocolo de pruebas, obteniéndose resultados positivos ante cada evento. Se plantea como trabajo a futuro la realización de pruebas en terreno y el añadir funcionalidades de actuación además de monitoreo.

A mi madre, mis abuelos y mi tía Moni.

A toda mi familia, mis amigos
y a todos los maestros del camino.

Contenidos

Índice de Figuras

Capítulo 1 Introducción

1.1 Introducción

Una de las tendencias tecnológicas existente en la actualidad es la llamada computación ubicua: la masificación y diversificación de dispositivos electrónicos que permean virtualmente cualquier aspecto de nuestra actividad diaria.

Características esenciales de estos dispositivos son su capacidad de procesamiento de información, pequeño tamaño y autonomía, todas estas posibles gracias al constante avance en las tecnologías de integración de circuitos.

Otra capacidad que ha irrumpido fuertemente en las últimas décadas es la movilidad, posible gracias a lo anteriormente señalado así como al gran avance que ha ocurrido en el campo de las comunicaciones inalámbricas.

En pos de presentar una experiencia "transparente" al usuario, surge la necesidad de que el intercambio de información entre dispositivos fijos y móviles se realice de manera lo más automatizada posible. Esto puede resultar especialmente crítico en el caso de Redes de Sensores, en las que es posible encontrar cientos e incluso miles de estos dispositivos conectados, con lo que la agregación de cada uno por parte del usuario se torna imposible.

Esta memoria se centra en la interconexión inalámbrica y monitoreo de sensores desde dispositivos móviles.

1.2 Motivación

La motivación para realizar este trabajo de memoria en relación con un sistema de monitoreo de sensores desde dispositivos móviles surge a partir de varias razones. En particular, se identifican tres principales:

- Una motivación inicial o caso de aplicación sugerido por el profesor guía era en el campo de la telemedicina, es decir en el cuidado de pacientes de manera remota. En este caso, existía interés por explorar el uso de sistemas móviles que permitieran el enlace entre redes de área personal con sensores conectados al paciente y las grandes redes de información. De esta manera, los signos vitales u otras variables de un paciente podrían ser transmitidas a centros de salud, mientras el sujeto se encuentra en la comodidad de su hogar. Además, se visualizaba un nicho de interés en aparatos que permitan al usuario crear aplicaciones sobre estos nodos de enlace, de tal manera de empoderarse con respecto al cuidado de su propia salud. Más aún, se podría generar un mercado para tales aplicaciones.

- En base al caso anterior y a investigaciones previas realizadas por el alumno, se tenía gran interés por explorar y aplicar protocolos de última generación relacionados con redes de

sensores inalámbricas. Estas tecnologías, de reciente aparición (como Zigbee y Bluetooth, de la última década), rompen el paradigma de buscar altas tasas de transmisión en base a poderosos *hardware,* sino que buscan una mínima huella en procesamiento y consumo de potencia. Esto permite la agregación de gran cantidad de nodos en una red, abriendo las puertas a recolección de datos del ambiente en forma masiva. La componente inalámbrica hace posible un despliegue de las redes muchísimo más simple y a menor costo (por nodo y no por ancho de banda, cual es la métrica utilizada en las redes destinadas a usuarios finales) que en el caso cableado. Se percibió además que las investigaciones en este campo apuntaban a redes en que los sensores permanecían estáticos o exhibían escaso movimiento. En contraposición, se observó la posibilidad de innovar incluyendo en estas redes agentes esencialmente móviles (ej. personas), introduciendo nuevas condicionantes a manejar como desconexiones y re-conexiones frecuentes. Un campo particular de gran interés para el autor es la domótica.

- Paralelamente al trabajo de memoria, el alumno fue invitado a participar en un proyecto de monitoreo móvil de variables ambientales en una mina subterránea propiedad de Codelco. Una de las motivaciones más fuertes para este proyecto se encontraba en la limitación que mediciones fijas imponen sobre la captura de información ambiental: para realizar un mapa estadístico completo de un sector para alguna variable (ej. temperatura), se debe tener un gran número de sensores fijos, o incurrir en un engorroso transporte y re-instalación constante de los mismos. En contraposición, mediciones móviles permiten una reducción de costos al utilizar un único equipo, a la vez que entregan mayor flexibilidad en la ubicación espacial de las muestras.

Los anteriores fueron tomados como referencias, reafirmando la importancia del proyecto; sin embargo el sistema se desarrolló en forma genérica, sin apuntar a ninguno de estos casos en forma particular.

Por otro lado, existe particular interés en desarrollar el trabajo sobre una plataforma en particular, los BUGs, por BugLabs Inc. Esta compañía corresponde a un *startup* tecnológico, con la cual se tienen relaciones estrechas desde la Universidad de Chile. Su propuesta es la arquitectura modular y abierta, tanto en software como hardware (ver BUGs en capítulo Marco Conceptual). El poder observar y aportar al proceso de evolución de esta plataforma, tanto en términos técnicos como comerciales, se percibe como una gran ventaja formativa para el alumno.

1.3 Objetivos Generales

Esta trabajo de título tiene como objetivo, a grandes rasgos, explorar las siguientes temáticas:

- Mecanismos inalámbricos de conexión y monitoreo de sensores desde dispositivos electrónicos móviles.

- La arquitectura modular, un nuevo paradigma de construcción de dispositivos electrónicos móviles.

Aplicando el conocimiento adquirido al diseño e implementación de un prototipo de sistema de monitoreo de sensores fijos desde dispositivos móviles.

1.4 Objetivos Específicos

Los objetivos específicos que se persiguen son los que se detallan a continuación:

- Realizar un estudio del estado del arte en:
    - Mecanismos de conexión entre sensores y dispositivos electrónicos móviles.
    - Modularidad en dispositivos electrónicos, aplicada al prototipaje rápido.
- Diseñar e implementar un prototipo de sistema que monitoree sensores (por ej. Temperatura) mediante dispositivos móviles. Como dispositivos móviles se utilizarán los dispositivos compactos de arquitectura modular denominados BUG.
- Observar, aprender y aportar al desarrollo tanto de un producto innovador como a la empresa responsable del mismo. El trabajo realizado deberá percibirse como un aporte técnico y/o económico para la empresa.

1.5 Requisitos

En el desarrollo del presente trabajo, se requiere de la disponibilidad de unidades BUG (al menos una durante el desarrollo y una cantidad a determinar para las pruebas) así como de los módulos que se determinen como necesarios.

En términos de software, se espera contar con soporte de Bug Labs Inc. en asuntos específicos con respecto a la arquitectura de estas unidades. Se presupone que funcionalidades necesarias pero no relacionadas directamente con el trabajo a desarrollar podrán ser solucionadas mediante el uso de software libre.

1.6 Metodología

En un primer momento, se emprenderá el estudio de los aspectos teóricos necesarios para determinar cómo proceder en el diseño del sistema, enfocándose especialmente en los protocolos de comunicación disponibles y como compatibilizar estos con la arquitectura seleccionada.

La primera etapa del diseño propiamente tal será establecer requerimientos para el sistema, como los que podría plantear un potencial usuario. Una vez realizado esto, y con los conocimientos de la primera etapa, se diseñará primero conceptualmente y luego en detalle la/las aplicación(es) que cumplan con estos requerimientos.

Finalmente, el sistema se someterá a pruebas de validación que certifiquen el correcto

cumplimiento de los requerimientos.

1.7 Alcance

El presente trabajo se enmarca dentro de los siguientes lineamientos:

- El sistema propuesto se considera una "prueba de concepto" y, por consiguiente, su validación se realizará en un ambiente controlado.

- Otra consecuencia de lo anterior es que los entregables del proyecto (documentación, códigos) tendrán como usuario objetivo a otros desarrolladores y no a usuarios finales.

1.8 Estructura del documento

El presente documento se divide en los siguientes capítulos:

1. Introducción: se provee un contexto general y motivación para el desarrollo. Luego se establecen los objetivos tanto generales como específicos, finalizando con detalles como las hipótesis previas, metodología y alcance del trabajo.

2. Marco Conceptual: se exponen los contenidos teóricos revisados durante la investigación, poniendo un énfasis en lo general en un comienzo, progresando hacia lo más específico. Los contenidos y la profundidad con que son tratados se seleccionaron tomando como público objetivo a otros alumnos de la carrera, considerando los conocimientos comunes que ellos poseen.

3. Diseño e Implementación: se describe el desarrollo del trabajo comenzando con los requerimientos establecidos para el sistema, para luego dar paso al diseño conceptual elaborado. Este busca cumplir con los requerimientos, a la vez que marcar una tendencia (ver apartado Visión) en el desarrollo. Después se presenta el desarrollo detallado tal de obtener un producto que aplique los lineamientos establecidos.

4. Relación Memoria-Empresa: se describe en forma particular los hitos que describen el trabajo conjunto desarrollado con la empresa Bug Labs Inc., así como la experiencia obtenida.

5. Pruebas y Discusión: se describen las pruebas de validación y el resultado de las mismas. Posteriormente, se realiza una discusión en profundidad de todo el desarrollo.

6. Conclusiones: se realiza una evaluación del grado de desempeño del sistema desarrollado en cuanto a los objetivos iniciales. También se mencionan otros aspectos dignos de destacar que hayan surgido como resultado del trabajo.

7. Referencias: se entrega un listado de las fuentes de información más relevantes consultadas por el autor.

Capítulo 2 Marco Conceptual

2.1 “Computación Ubicua”

Computación Ubicua (*"Ubiquitous computing", "Ubicomp"*) es un término que busca describir la tendencia moderna de masificación y diversificación de dispositivos electrónicos de procesamiento de información que permean virtualmente cualquier aspecto de nuestra actividad diaria. Características esenciales de estos dispositivos son su pequeño tamaño y autonomía, posibles gracias al permanente avance en las tecnologías de integración de circuitos.

El concepto de Computación Ubicua fue acuñado originalmente por Mark Weiser en 1988, en el Laboratorio de Ciencias de la Computación de Palo Alto Research Center (PARC) de Xerox. Otro término, utilizado indistintamente en la literatura es *"Pervasive Computing"*. En las propias palabras de Weiser: *"For thirty years most interface design, and most computer design, has been headed down the path of the "dramatic" machine. Its highest ideal is to make a computer so exciting, so wonderful, so interesting, that we never want to be without it. A less-traveled path I call the "invisible"; its highest ideal is to make a computer so imbedded, so fitting, so natural, that we use it without even thinking about it. (I have also called this notion "Ubiquitous Computing", and have placed its origins in post-modernism.) I believe that in the next twenty years the second path will come to dominate."* El investigador se refiere, en [1], a la computación ubicua como una tercera oleada tecnológica en el área de la computación. La primera Era correspondería a la de los *mainframes*: unidades de procesamiento centralizadas, compartidas por una multitud de usuarios, quienes no tenían acceso directo a la configuración de la misma. La segunda, a la del Computador Personal (*"Personal Computer","PC"*), en que las unidades de procesamiento disminuyeron sus costos y tamaño, de manera que cada usuario puede poseer su propio dispositivo. El autor vislumbra la internet como un periodo de transición hacia la tercera Era, de la computación ubicua, en la cual cada usuario tendrá acceso a cientos de dispositivos. Mientras que algunos de estos dispositivos serán similares a las estaciones actuales, se espera que otros se integren imperceptiblemente en la vida diaria, encontrándose insertos en paredes, ropa, automóviles, interruptores, etc. Esencialmente, la tercera Era dice relación con la conexión de los objetos del mundo "real" con el de la computación.

El objetivo de los investigadores y desarrolladores es crear sistemas que estén presentes en cualquier ambiente, sin causar obstrucción, completamente conectados, sean intuitivos de usar, portátiles sin esfuerzo adicional, y que estén disponibles permanentemente.

Algunas características de los dispositivos dentro de este nuevo paradigma son las siguientes:

- El número de estos será mucho mayor al número de usuarios.
- Muchos de los nuevos dispositivos corresponden a periféricos de otros equipos más poderosos: existirá una fuerte diversificación y diferenciación de y entre los dispositivos.
- Exhibirán una fuerte tendencia (existe ya en la actualidad) a la conectividad sin cables

(*"Wireless"*), lo que cobra sentido en base a la premisa de no obstaculizar los ambientes, siendo en lo posible imperceptibles ante el usuario.

Una tendencia afín es la Inteligencia del Ambiente ("Ambient Intelligence"), donde se busca crear espacios que sean sensibles y reactivos (o incluso proactivos) ante los usuarios. El propósito es que los dispositivos utilizados trabajen en conjunto para apoyar el que la realización de las tareas cotidianas de los usuarios sea lo más simple posible. Un sistema que interconecte y ajuste controles de iluminación y regulación del ambiente en base a la información entregada por medidores de variables *biométricas* usados por los usuarios dentro de un hogar es un buen ejemplo de este concepto.

Existen numerosos desafíos antes del asentamiento final de la computación ubicua. Estos van desde la creación de los dispositivos mismos (que en algunos casos llegarán a ser de tamaño microscópico, [4]), sistemas de conexión, diseño de redes e ingeniería de tráfico, y finalmente interfaz de usuario. En cuanto a esta ultima área, se piensa que los sistemas tradicionales basados en teclados y respuestas graficas como líneas de comando o las actuales GUI (*"Graphical User Interface"*) en base a ventanas 2D quedarán obsoletas ante interfaces mas "naturales", cuyas características aun no han sido siquiera concebidas.

Ejemplos de sistemas que podríamos clasificar como pertenecientes a los albores de la Era de la Computación Ubicua son teléfonos móviles, reproductores multimedia portátiles, sistemas de identificación como RFID y de localización como aparatos GPS, entre otros. Un ejemplo algo más elaborado, que capturó la atención del autor, son los "Nabaztag" [5] (conejo en armenio). Estos "conejos" son dispositivos electrónicos, completamente configurables, con capacidad de conexión WiFi y RFID. Ante gestos o comandos verbales un "Nabaztag" es capaz de descargar y leer en voz alta el correo del usuario o de indicar la detección de algún objeto del medio en base a etiquetas RFID.

2.2 Sistemas de Comunicación

2.2.1 Modelo OSI

En 1978, la Organización Internacional para la Estandarización (ISO, International Organization for Standardization) comenzó la creación de un modelo para Interconexión de Sistemas Abiertos (OSI, Open Systems Interconnection). La iniciativa OSI definió un modelo en el que las funciones dentro de cada equipo se encuentran separadas en capas verticales. Cada capa tiene funciones lógicamente similares asociadas, distintas de las demás. Las capas se comunican únicamente con aquellas directamente adyacentes (superior/inferior) en el mismo equipo, pero tienen como destinatario a la capa del mismo nivel del equipo receptor. Esto puede entenderse ya que en la realización de sus funciones, es usual que cada capa agregue a los datos información adicional, la cual resulta relevante solo para esa capa. La comunicación se origina en las capas más altas (y abstractas), siendo los datos entregados a las capas inferiores para que estas realicen sus funciones, como por ejemplo transmisión de extremo a extremo a través de una red, transmisión punto a punto entre nodos que conforman la red, etc. La información llega así a la capa inferior, la cual se encarga realmente del envío de datos del emisor al receptor. Una vez recibida, esta es procesada por la capa inferior del equipo receptor, que decodifica el contenido a

entregar a la capa directamente sobre ella. Esta a su vez realiza el mismo proceso de decodificación y traspaso a la capa superior, y así hasta que los datos llegan la capa más alta en el receptor, con lo que se ha completado la comunicación.

El modelo contempla una estandarización abierta de las funciones a implementar por cada capa y la interfaz que debe tener hacia las capas adyacentes, pero no especifica cómo llevarlas cabo. De esta manera, permite la libre implementación por parte de múltiples vendedores, fomentando la competencia en el mercado, a la vez que asegura la interoperabilidad de los equipos que cumplen con el estándar. Existen también ventajas conceptuales en un modelo de capas, puesto que al dividir un problema complejo este se torna más manejable, encontrándose cada subdivisión del mismo claramente acotada. El modelo OSI contempla 7 capas, las que se describen en la tabla Tabla 2.1.

1. Capa Física	Describe los detalles del medio físico de transmisión, como el tipo de interfaz (antenas, cables), la modulación, conectores, etc.. También define como hacer uso de la interfaz física.
2. Capa de Enlace de Datos	Se encarga de asignar direcciones que permitan el acceso al medio de transmisión (direccionamiento lógico), posibilitando el envío de datos de un punto a otro dentro de la misma red. También implementa funciones de detección y posiblemente corrección de errores.
3. Capa de Red	Define la formación de redes en base a: direccionamiento lógico, ruteo y búsqueda de caminos. Las direcciones lógicas se asignan para ser usadas en el proceso de ruteo. Este consiste en el reenvío de información entre nodos vecinos (conectados físicamente), siguiendo una secuencia preestablecida, de tal manera de llevar la información a un destinatario final. Los algoritmos de búsqueda de caminos se encargan de deducir las posibles rutas entre 2 nodos y elegir la más adecuada entre ellas.
4. Capa de Transporte	Se encarga de proveer transporte de datos de un extremo a otro (abstrayéndose de los “saltos” intermedios), proveyendo funciones de control de errores y posiblemente de flujo (velocidad, entre otras características) de la información.
5. Capa de Sesión	Define como iniciar, mantener y terminar “conversaciones” o sesiones entre 2 sistemas.
6. Capa de Presentación	Negocia el tipo o formato de datos de la información traspasada (ej.: datos codificados en ASCII, o que componen un archivo JPEG, etc.).
7. Capa de Aplicación	Es el enlace entre el/los sistemas de comunicación y las aplicaciones que corren en cada equipo y que requieren enviar información.

Tabla 2.1: Capas del modelo OSI y sus funciones

Además del modelo, llegaron a existir implementaciones concretas de protocolos para

algunas capas propuestas por la OSI. Las mismas no tuvieron el éxito comercial de las equivalentes en el actual modelo imperante, el modelo TCP/IP. Sin embargo, el modelo de referencia OSI es de amplio uso en el estudio y análisis de los sistemas de comunicaciones, tanto en el mundo empresarial como académico.

2.2.2 Infraestructura

En la práctica, existen 2 tipos esenciales de infraestructura de redes de comunicación, *de infraestructura* y *ad hoc*.

Las redes de infraestructura poseen equipos con funciones diferenciadas, actuando algunos como estaciones base, enrutadores de la comunicación, y otros como usuarios o nodos finales. Los nodos finales no pueden comunicarse directamente entre ellos, sino que solo hacia una estación base, la que a su vez puede reenviar el mensaje a través de otras estaciones y así llegar al otro nodo final. Para que lo anterior sea posible, las estaciones base del sistema deben estar conectadas entre sí. Las redes de telefonía móvil son un buen ejemplo de este tipo de arquitectura.

Las redes ad hoc son aquellas en que la que todos los nodos pueden actuar como enrutadores de la comunicación, además de poder ser cada uno emisor o receptor final de un mensaje. Este paradigma es esencialmente útil para el rápido despliegue de redes locales, por ejemplo en redes de sensores. En estas, es posible que ninguno de estos tenga acceso a redes de infraestructura, pero que la comunicación entre estos sea necesaria para llevar a cabo su tarea. En la eventualidad de que algunos de los nodos sí tengan acceso a alguna red establecida (ej.: internet), este puede actuar como puerta de enlace (*gateway*), uniendo efectivamente ambas redes.

Mientras que existen algoritmos de ruteo en redes de infraestructura con una sólida comprobación en la práctica, el ruteo en redes ad hoc es un tema de investigación de frontera, puesto que en las aplicaciones prácticas, estas redes suelen tener una topología variable. Existen esencialmente 2 mecanismos para encontrar rutas en este escenario: proactivos y reactivos.

Los protocolos proactivos mantienen listas actualizadas de las rutas distribuyendo tablas periódicamente a todos los nodos. Las desventajas de este enfoque son que tiende a acumularse mucha información, lo que hace difícil su mantención, y que las reacciones ante cambios en la topología son lentas. En contraposición, los protocolos reactivos buscan rutas a medida que estas son solicitadas, inundando la red con paquetes de requerimiento de ruta. Las desventajas en este caso son la alta latencia para encontrar una ruta y el hecho que demasiada "inundación" puede congestionar la red. Como es de esperarse, existen también combinaciones de ambos métodos.

A modo de ejemplo, se describen los pasos para encontrar una ruta en el protocolo reactivo Ad hoc On-Demand Distance Vector Routing (AODV)

1. La red se encuentra en silencio hasta que algún nodo necesita establecer una conexión.

2. El nodo envía una comunicación a todos los demás ("broadcast") pidiendo una ruta hacia

su destino.

3. Otros nodos retransmiten la petición, recordando de qué nodo la recibieron.

4. Si alguno tiene conectividad con el nodo deseado, puede deducir la ruta en base a la información de los mensajes de broadcast. Envía ésta al nodo solicitante por alguna ruta temporal (pudiendo ser eventualmente la misma u otra si es que ya se han descubierto alternativas).

5. El nodo transmisor utiliza la ruta con menos tramos de entre las recibidas.

6. Las rutas que no son utilizadas se reciclan después de un tiempo.

AODV está especificado en el Request For Comment (RFC) 3561 de la Internet Engineering Task Force (IETF). Una adaptación del mismo es utilizado en el protocolo ZigBee, detallado más adelante.

2.2.3 Topología

La topología de una red describe la forma en que los nodos dentro de la misma están interconectados. Una medida de la complejidad de una red es su diámetro, es decir, la máxima cantidad de "saltos" entre 2 nodos cualquiera [6].

Existen múltiples topologías, siendo la más simple el enlace punto a punto, con un diámetro de 1. Esto es también cierto para las redes conectadas a través de buses, con la salvedad de que en este caso el medio de comunicación es un recurso compartido y por lo tanto el acceso al mismo tiene ciertas restricciones.

Luego encontramos la topología en estrella, en que múltiples nodos "terminales" se conectan a un coordinador central, que actúa como enrutador de las comunicaciones entre estos, con lo que el diámetro es de 2.

La combinación de múltiples redes en estrella conforma una estrella extendida. La topología jerárquica tiene similitud con las estructuras de datos llamadas árboles, donde un padre puede tener asociados múltiples hijos, los que a su vez pueden ser padres de otros nodos. Además de las anteriores, existe la topología de anillo y la última corresponde a una en que las conexiones no siguen ningún patrón definido sino que están determinadas por un grafo arbitrario. Todas las anteriores tiene un diámetro dependiente del número de nodos y las conexiones entre estos.

La topología afecta varias características de una red, como su latencia, robustez y capacidad [6].

Las redes enmalladas (mesh) son aquellas en las que cualesquiera dos nodos dentro de la misma pueden comunicarse, ya sea directamente o a través de múltiples saltos entre otros nodos, los cuales actúan como enrutadores. Estas redes pueden considerarse como un tipo de red ad hoc. Una red enmallada en que todos los nodos pueden comunicarse directamente (enlaces punto a

punto) se dicen completamente conectada. Las diferentes topologías se esquematizan en la Figura 2.1.

En la práctica, el concepto de redes enmalladas comúnmente requiere de características adicionales, como la auto-formación y auto-sanación. La primera dice relación con la capacidad de un nodo de descubrir a sus vecinos directos y luego determinar las rutas de comunicación hacia los nodos que se encuentren a más de un salto de distancia. La segunda es la habilidad para encontrar múltiples rutas hacia un mismo nodo, utilizando alternativas en el caso de que alguna de ellas llegase a fallar (por ejemplo ante la obstrucción de un enlace, destrucción de un nodo, etc.).

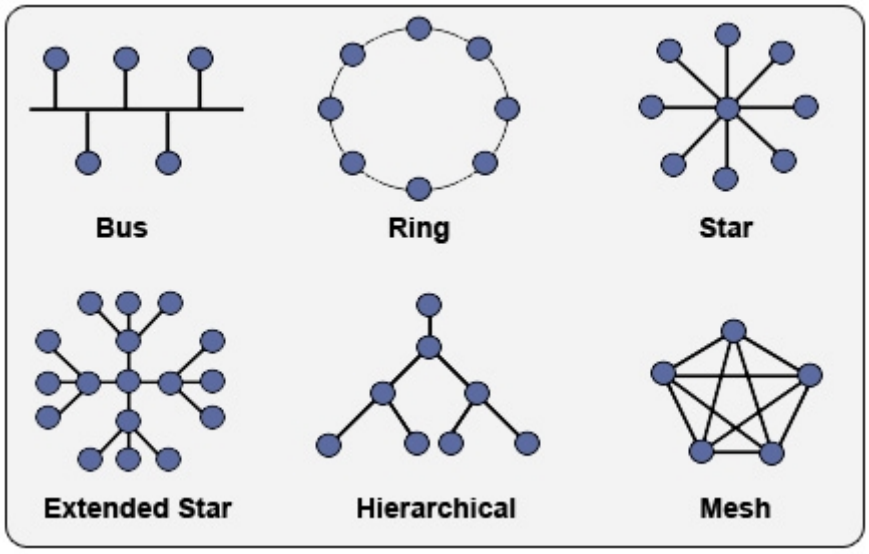


Figura 2.1: Topologías de red

2.2.4 Comunicaciones Inalámbricas

Tal como su nombre lo estipula, las comunicaciones inalámbricas son aquellas en las que la información se transmite en ausencia de cables. Si bien en estricto rigor existe una multiplicidad de sistemas que cumplen estas características (desde señales de humo hasta códigos visuales), en la actualidad el uso de ondas radioeléctricas se ha constituido como el método por excelencia para este tipo de comunicaciones.

Hace ya varias décadas que la propagación de ondas eléctricas por el espacio libre se ha utilizado en comunicaciones, siendo la radiodifusión un ejemplo clásico. Sin embargo, este tipo de comunicaciones es solo unidireccional (simplex), mientras que posteriormente se han desarrollado efectivos sistemas de comunicación bidireccionales (dúplex) como los enlaces de microondas y satelitales. Estos sistemas, siendo de alto costo, eran prohibitivos para individuos y solo estaban al alcance de grandes compañías.

Los últimos años han sido testigo de una revolución en las comunicaciones personales,

considerando principalmente las redes masivas de telefonía celular pero también el campo en expansión de las *redes inalámbricas de área personal* (WPAN, Wireless Personal Area Networks). Estas redes tienen como propósito interconectar dispositivos en entornos pequeños, como por ejemplo los dispositivos que puede acarrear una persona.

La comunicación inalámbrica presenta una serie de ventajas, tales como las que se enumeran a continuación:

- **Movilidad:** permite el desplazamiento de los nodos. Esto puede ser en muchos casos una cualidad deseable (por ejemplo para usuarios transportándose a pie en un contexto urbano) pero absolutamente esencial en otros, como por ejemplo en aplicaciones relacionadas con vehículos en movimiento (estaciones de aero-control, etc.). En el contexto de la minería, existen aplicaciones relacionadas con la seguridad de los operarios y equipos que resultan intrínsecamente móviles.

- **Fácil implementación:** Establecer una comunicación cableada puede ser muy intensivo en recursos, especialmente en caso de tener que cubrir grandes distancias o donde la cantidad de puntos a unir sea muy numerosa. En contraposición, un enlace inalámbrico puede establecerse en el caso más simple (y común) con equipos en el lado emisor y receptor, y en otros casos con estaciones repetidoras intermedias, lo que sigue resultando más simple que el caso cableado.

- **Economía:** como consecuencia de lo anterior, los enlaces inalámbricos han probado ser una solución económicamente viable en muchos casos; en [9] se señala que el costo de cableado en una instalación industrial puede alcanzar los 130-250 USD por metro, y que estudios demuestran que las tecnologías inalámbricas presenta un costo entre 20% y 80% menor.

2.2.5 Propagación de ondas en el espacio libre

La propagación de ondas en el espacio u otro medio similar (como el aire) está condicionada esencialmente por 2 factores:

- **Atenuación:** este fenómeno dice relación con la pérdida de potencia transmitida. Esto ocurre por un lado por factores intrínsecos del fenómeno radioeléctrico, como el hecho de que no toda la potencia radiada llega al receptor sino que parte de esta se dispersa en otras direcciones. Por otro lado, existen factores prácticos, como el que los equipos de transmisión no presentan una eficiencia perfecta.

- **Interferencia:** La Interferencia Electro-Magnética (EMI, ElectroMagnetic Interference) es el fenómeno resultante de la aditividad de las ondas electromagnéticas, con lo que habiendo más de una de estas presente en el ambiente, la señal recibida resulta en una superposición de las mismas. Esto puede afectar la inteligibilidad de las señales enviadas. El aumento en el uso de las comunicaciones inalámbricas y la propagación indeseada de otras señales, como aquellas compuestas de pulsos aleatorios (electrónica de potencia), contribuyen a generar un fondo de ruido radioeléctrico. Esto impone un umbral mínimo

de potencia a recibir para poder distinguir una señal deseada de las demás.

2.2.6 Protocolos de comunicación inalámbrica para monitoreo de sensores

A continuación, se describen brevemente una serie de protocolos de comunicación inalámbrica con cierta relevancia para el trabajo presente. De esta manera, se han seleccionado protocolos digitales y de alcance corto a medio.

2.2.6.1 IEEE 802.11/WiFi™

El estándar IEEE 802.11 describe las capas bajas (física y de enlace) de transmisión inalámbrica equivalentes a Ethernet, un medio cableado para establecer redes de área local. Utiliza las bandas de frecuencia de 2,4 GHz y 5 GHz y Direct Sequence Spread Spectrum (DSSS) y Orthogonal Frequency Division Multiplexing (OFDM) como métodos de modulación. Permite velocidades de comunicación desde 1 hasta 150 Mbps, siendo los sub-estándares 802.11b, con 11 Mbps y 802.11g con 54Mbps los más comunes. Dadas estas velocidades, se percibe como una solución de comunicación completa (datos, multimedia, streaming, etc.) y considera el uso de terminales de alta capacidad.

La Alianza WiFi es un organismo encargado de aprobar el uso de la marca WiFi, que certifica la compatibilidad de un cierto grupo de productos que utilizan el estándar 802.11 como base. WiFi es una tecnología ampliamente masificada, estando presente en la actualidad en básicamente todos los ordenadores móviles y varios otros equipos como terminales de telefonía celular, periféricos y consolas de videojuegos, por nombrar solo algunos.

2.2.6.2 IEEE 802.15.1/Bluetooth™

Bluetooth es un protocolo de comunicación inalámbrico para Redes de Área personal. Su objetivo principal es la conectividad de equipos computacionales y dispositivos portátiles de electrónica de consumo (teléfonos celulares, reproductores de música), eliminando la necesidad de cables. Su filosofía va ligada a su nombre, pues este es en honor a Harald Blåtand (cuya traducción al inglés sería Harold Bluetooth), rey danés y noruego conocido por ser un buen comunicador y por unificar las tribus noruegas, suecas y danesas.

Las principales características de bluetooth se listan a continuación:

- Tasa de transmisión de datos:
  - Versión 1.2, 1 Mbps
  - Versión 2.0 (+ Enhanced Data Rate, EDR), 3 Mbps
  - Versión 3.0 (+ High Speed, HS), 24 Mbps[1]

1 La versión 3.0 + HS utiliza Bluetooth para establecer la comunicación, la que posteriormente se lleva a cabo a través de un enlace 802.11 dedicado.

- Alcance, dependiente de la “clase”:
    - Clase 1, ~100 m.
    - Clase 2, ~22 m.
    - Clase 3, ~6 m.
- Banda de frecuencia: 2,4 GHz
- Método de acceso al medio: Frequency-Hopping Spread Spectrum, FHSS.
- Topología: redes en estrella, con un coordinador como maestro de hasta 7 esclavos.
- Enfoque a la seguridad: se realiza encriptación de la información en base a códigos que deben ser ingresados por los usuarios en ambos dispositivos.

Existen 2 versiones de Bluetooth homologadas por la IEEE; la versión 1.1 corresponde al estándar 802.15.1-2002 y la 1.2 al 802.15.1-2005.

2.2.6.3 IEEE 802.15.4

El estándar 802.15.4 de la IEEE es un protocolo de comunicaciones para redes inalámbricas de corto alcance, o redes de área personal (WPAN, Wireless Personal Area Network). Sus características son bajo consumo y costo de los dispositivos, a expensas de una baja tasa de transmisión de datos. Esto se ajusta a aplicaciones como redes de sensores, donde pueden existir una multitud de nodos que requieren solo un pequeño ancho de banda, pero a la vez necesitan la mayor autonomía energética posible.

El estándar define las capas Física y de Acceso al Medio, 1 y 2 en el modelo OSI respectivamente. De esta manera, no es una solución de comunicación completa sino que requiere de la implementación de las capas superiores.

Las principales características cuantitativas del estándar 802.15.4 se listan en la Tabla 2.2.

Bandas de frecuencia	• 868 MHz (Europa, 1 canal) • 915 MHz (USA, 10 canales con separación de 2 MHz) • 2,4 GHz (Mundial, 16 canales con separación de 5 MHz)
Tasa de transmisión de datos	• 20 Kbps (banda 868 MHz) • 40 Kbps (banda 915 MHz) • 250 Kbps (banda 2,4 GHz)
Alcance ([2])	Especificación original habla de 10 m., aunque muchos dispositivos en la actualidad llegan a 100 m.
Tamaño de las direcciones, subcapa MAC	64 bits, aunque solo se usan 16 una vez establecida la red

Tabla 2.2: Características cuantitativas del estándar IEEE 802.15.4

Además existen las siguientes características adicionales:

- **Consumo muy bajo:** la filosofía del estándar, que apunta a la redes de sensores, permite la fabricación de dispositivos (como sistemas embebidos) que pueden permanecer conectados durante años solo con el poder de sus baterías. A modo de ejemplo, una de las características que permiten esto es la utilización de direcciones de capa MAC de tan solo 16 bits (para disminuir el número de bits a transmitir) una vez establecida una red (determinando un máximo de 65.536 nodos). Si bien las direcciones únicas a nivel mundial tienen 64 bits, al establecer la red se asignan direcciones cortas a cada uno de los dispositivos.

- **Reducidos costos de fabricación:** otra cualidad esencial para las redes de sensores es la necesidad de un bajo costo unitario, puesto que estas redes pueden estar formadas por cientos e incluso miles de dispositivos. El estándar resulta simple de implementar y permite la comunicación con virtualmente ninguna infraestructura, lo que reduce el costo de la electrónica asociada.

- **Capacidad de comunicación en tiempo real:** el estándar permite reservar segmentos temporales (time slots), de manera de asegurar un cierto ancho de banda a dispositivos que lo requieran, sin perder la capacidad de incorporar tráficos espontáneos.

- **CSMA-CA (Carrier Sense Multiple Acces – Collition Avoidance):** implementa una escucha activa del medio para evitar colisiones entre unidades de datos (tramas o frames en este caso, tratándose de la capa 2) y la retransmisión (en momentos aleatorios) de estas si dichas colisiones ocurren. Equivalente a lo utilizado en el estándar Ethernet.

- **Seguridad:** La subcapa MAC permite realizar encriptación simétrica de las tramas, pero las llaves deben ser especificadas por las capas superiores.

El estándar contempla 2 tipos de nodos:

2 El alcance de una comunicación punto a punto depende de la potencia transmitida y sensibilidad de los equipos, entre otros factores.

- Dispositivos de funcionalidad completa (FFD, Full Function Device): pueden comunicarse directamente con cualquier otro dispositivo y además reenviar mensajes a otros destinatarios, ejerciendo la función de enrutadores.

- Dispositivos de funcionalidad reducida (RFD, Reduced Function Device): pueden comunicarse únicamente con FFDs, para lo cual necesitan de muy pocos recursos (tiempo y poder de procesamiento, etc.).

En cuanto a la topología de red, existe la posibilidad de establecer redes punto a punto o en estrella. Ambas requieren que al menos uno (y solo uno) de los nodos asuma la función de coordinador, que únicamente puede ser un FFD debido a la complejidad de la tarea. El nodo coordinador guarda información relacionada con toda la estructura de red. La elección de la topología se deja a la capa de aplicación.

2.2.6.4 ZigBee™

ZigBee es una especificación de protocolo de comunicación, perteneciente a la ZigBee Alliance, organización sin fines de lucro creada por múltiples compañías para mantener este estándar. Utiliza el IEEE 802.15.4 como solución para las capas Física y de Acceso al Medio, y añade las capas de Red y Aplicación (más específicamente, una interfaz para aplicaciones hechas por el usuario o API, Aplication Programming Interface). Incorpora ventajas como:

Creación de redes enmalladas: en base a las topologías de bajo nivel definidas por el estándar 802.15.4, punto-a-punto y estrella, ZigBee permite la formación de redes completamente enmalladas (todos los nodos pueden actuar como enrutadores, y cualquier nodo puede encontrar una ruta hacia cualquier otro a través de múltiples saltos). De esta manera se logra un mayor alcance de la red sin necesidad de una infraestructura.

Perfiles de Aplicación: estos tienen como fin determinar ciertos parámetros de las redes según el tipo de aplicación. Los parámetros especificados en un perfil tienen incidencia directa sobre la estructura de red posible de establecer, y por consiguiente en la calidad de la comunicación (en especial la eficiencia del ruteo), así como otras características (incluyendo el tipo de seguridad).

Además de ZigBee, existen numerosas otras soluciones propietarias que descansan sobre el estándar 802.15.4 e implementan las capas superiores de la comunicación. Sin embargo ZigBee es la base de varias de estas alternativas y puede argumentarse que, al momento de elaboración de este trabajo, es la de mayor visibilidad en la industria.

Al igual que en el 802.15.4, ZigBee incorpora 3 tipos de dispositivos, con funcionalidades equivalentes:

- Coordinador (ZC, ZigBee coordinator). Establece la red en un inicio y guarda información acerca de la misma, incluyendo las llaves criptográficas. Debe existir un único coordinador en la red.

- Router (ZR, ZigBee Router). Puede correr aplicaciones propias, comunicarse con otros y enrutar paquetes de nodos vecinos.

- Dispositivo Final (ZED, ZigBee End Device). Solo pueden comunicarse con un ZC o ZR, no enrutando comunicación ajena. De esta manera, puede permanecer apagado la mayor parte del tiempo, disminuyendo su consumo de energía.

2.2.6.5 Comparación

En lo que sigue, se realiza una comparación de los protocolos anteriormente listados, poniendo énfasis en las características de mayor relevancia para los objetivos de esta memoria, es decir la interconexión y monitoreo de sensores.

El foco de Bluetooth es en la interconexión de dispositivos electrónicos centrados en el usuario (ej.: PCs, celulares), contemplando el traspaso de archivos de mediano tamaño (orden Mbytes) entre ellos. Esto genera la necesidad de alcanzar velocidades de datos del orden de los Mbps (3 Mbps en la versión 2.0 + EDR, muy popular en teléfonos actualmente en venta en el mercado chileno), con el consecuente mayor consumo de energía en comparación con ZigBee/IEEE 802.15.4. Este protocolo se orienta hacia dispositivos de control y sensórica, que no requieren de altas tasas de transferencia pero sí de una mayor robustez en la comunicación y menor uso de energía, pues muchos de ellos apuntan al uso de baterías por periodos incluso de años. Por otro lado, WiFi apunta a entregar una conectividad de alta velocidad al usuario final, permitiendo una experiencia de navegación "completa" o equivalente a la de soluciones cableadas como Ethernet.

Otra diferencia fundamental es la topología. Bluetooth se considera una alternativa al cableado de dispositivos tipo "periféricos", por lo que puede establecer "pico redes" de tamaño 8 (un maestro con 7 esclavos). WiFi provee un soporte para entre 32 y 255[3] nodos (utilizando un solo router). En la práctica se sugiere no tratar de alcanzar el límite máximo, distribuyendo la carga entre múltiples puntos de acceso. ZigBee/IEEE 802.15.4 ofrece soporte para redes de ~65.000 nodos.

En cuanto a la infraestructura, mientras que WiFi soporta un modo ad hoc, este no es el más comúnmente utilizado, sino que el de infraestructura. Por un lado esto permite la utilización de hardware de alta capacidad en los puntos de acceso y en los mismos dispositivos que acceden a la red, ya que normalmente existirá disponibilidad de otros recursos como energía eléctrica, etc.. Por otro lado, restringe su alcance y también robustez (fuerte dependencia de los routers). La forma en que Bluetooth conforma redes en estrella con un maestro y algunos esclavos puede considerarse de infraestructura, solo que esta es fácilmente portable. Lo mismo es cierto para el protocolo IEEE 802.15.4. Mientras que el tamaño máximo de 8 dispositivos limita el escalamiento en el primer caso, el segundo permite escalar a miles de nodos. Los protocolos anteriores establecen redes de topología en estrella (pudiendo realizarse estrellas extendidas). Zigbee trabaja sobre redes enmalladas, con capacidades de auto-formación y auto-sanación, en base a un ruteo ad hoc. Esto significa que la red se establece donde los dispositivos se encuentren, y que el alcance de la misma puede ampliarse con la adición de más nodos. También debe

3 Según [15] y [14] respectivamente

considerarse que el alcance a nivel de capa física de IEEE 802.15.4/ZigBee es mayor, seguido por WiFi y en último lugar Bluetooth.

Finalmente, es importante señalar que la complejidad/simplicidad de los protocolos tiene repercusión en el costo de los transmisores. Una métrica en este sentido es la cantidad de memoria utilizada por el sistema para implementar el stack[4] respectivo; mientras que WiFi requiere más de un MB, Bluetooth reduce esto a 250 KB y ZigBee/IEEE 802.15.4 a tan solo entre 4-32 KB [15]. Otra medida es el uso de baterías: WiFi es el de mayor consumo, estando orientado a computadores portátiles. En este segmento alcanza una duración de batería de horas. En el segmento de teléfonos portátiles, WiFi alcanza duraciones del orden de una semana, al igual que Bluetooth. En ambos segmentos anteriores, los dispositivos tienen factibilidad de ser recargados frecuentemente. La situación es diferente en el caso de sensores y dispositivos afines, los que pueden requerir de largos periodos de autonomía. Por otro lado, las tecnologías anteriores asumen la necesidad de conectividad constante, privilegiando la experiencia del usuario en este sentido; los dispositivos en una red de sensores pueden solo requerir de comunicación esporádica. En este sentido, ZigBee e IEEE 802.15.4 permiten permanecer en modo de bajo consumo y transmitir solo cuando se requiere.

Las diferencias anteriormente estipuladas se resumen en la Tabla 2.3 (principalmente basada en [15] y [16]).

Tecnología	Zigbee™	WiFi™	Bluetooth™
Estándar IEEE asociado	802.15.4	802.11	802.15.1
Aplicación	Monitoreo y Control	Web, Email, Video	Reemplazo de cables
Tiempo de vida de Batería (días)	100-1000+	1-7	1-7
Ancho de Banda (tip.)	20-250 Kbps	11-54 Mbps	1 Mbps
Rango[5] (tip.)	100+	1-100	10+
Recursos del sistema	4-32 KB	1 MB+	250 KB+
Tamaño de las redes	> 65.000	32-255	8
Ventajas	Robustez, baja potencia, costo	Velocidad, flexibilidad	Costo, comodidad

Tabla 2.3: Comparación de tecnologías de comunicación inalámbricas.

2.2.6.6 Radios XBee

Las radios XBee son módems inalámbricos fabricados por la empresa Digi. Algunas versiones soportan el protocolo IEEE 802.15.4, mientras que otras incluyen el stack ZigBee. Son de bajo costo, y pequeño tamaño. Algunas de sus características principales se agrupan en la Tabla 2.4. La Figura 2.2 muestra una radio XBee con antena integrada.

4 Implementación (en software) de un grupo de protocolos asociados a una tecnología.
5 Tipico, en exteriores/linea de vista.

Estos dispositivos se comunican con el PC host mediante el protocolo serial RS232. Soportan 2 modos de operación: transparente y API. En el primero, el módem funciona como reemplazo de un cable serial, enviando todos los bytes recibidos por la interfaz RS232 hacia el módem con una dirección especificada en un registro. Para realizar operaciones como cambio de la dirección de destino u otras, existe un modo "comando" al cual se accede ingresando una secuencia de escape (ej.: "+++" por defecto). Luego se envían mensajes según el protocolo AT [33] (ej.: ATMY entrega el contenido del registro MY, la dirección propia de la radio, mientras que ATDL2 escribe los 32 bits inferiores de la dirección de destino con el numero hexadecimal 2). Para salir de este modo existe un comando o simplemente se deja expirar un temporizador. El módem se encarga automáticamente de formar los paquetes para enviar a través del aire, y a la vez entrega los caracteres de los paquetes recibidos como una secuencia (*stream*).

En el modo API, el módem recibe por RS232 los paquetes a transmitir ya formados por el procesados de la aplicación. Cada paquete incluye la dirección de destino, con lo que no es necesario interrumpir el flujo de información para cambiar de destinatario. A esto se suman ventajas como envío de paquetes a todos los host de la red (*broadcast*) y configuración de módems a través de comandos remotos. Además de los paquetes a ser transmitidos, existen otros que encapsulan comandos AT, y que permiten realizar operaciones sobre el módem. A modo de ejemplo, la Figura 2.3 muestra el formato de un paquete en modo API.

En general, el modo API es preferido para aplicaciones donde un nodo particular debe ser capaz de direccionar a varios otros nodos.

Alcance en interiores (sin línea de vista)	30 m.	90 m.
Alcance en exteriores (con línea de vista)	90 m.	1600 m.
Potencia de transmisión	1mW	63 mW
Tasa de datos de RF	250 Kbps	250 Kbps
Banda de frecuencia	ISM 2,4 GHz	ISM 2,4 GHz

Tabla 2.4: Características de radios XBee IEEE 802.15.4 ([32])

Figura 2.2: Radio XBee con antena integrada.

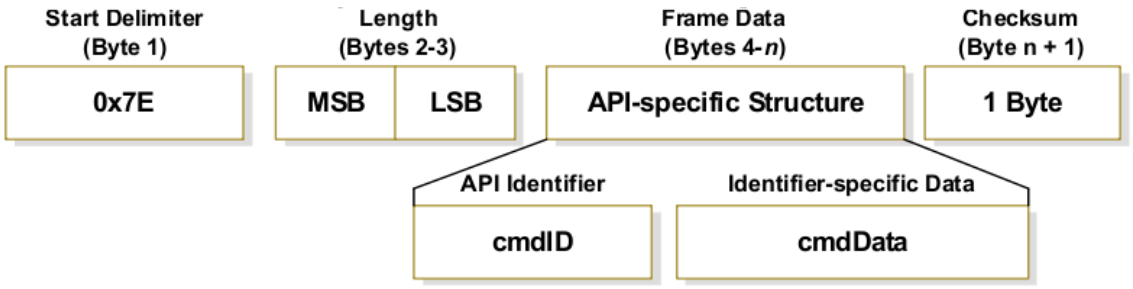


Figura 2.3: Formato de paquete XBee IEEE 802.15.4 en modo API.

2.3 Arquitectura Modular en Hardware

Entenderemos un dispositivo como de "arquitectura modular" en cuanto a su hardware en caso de que esté compuesto de al menos una unidad básica, que presente una funcionalidad mínima (en cuanto a los requerimientos que se tengan para el dispositivo), y que además esta pueda ser complementada con la adición de otros dispositivos físicos (*módulos* o *bloques*), con funcionalidades definidas, que pasan a expandir las funcionalidades de la unidad básica. Una analogía simple la constituyen los computadores de escritorio que en conjunto con sus periféricos básicos (pantalla y teclado, aunque estos vienen ya incluidos en los laptops actuales) presentan una funcionalidad base, pero mediante la adición de más periféricos *no básicos* como el popular mouse, tarjetas de video, sonido, impresora, etc., presentan un set de funcionalidades extendidas.

2.3.1 Dispositivos compactos de arquitectura modular

Por compacto entenderemos que el tamaño y peso del dispositivo, al menos considerando el modulo que provee la funcionalidad básica, permite el fácil y cómodo transporte y utilización de este por parte de un usuario. Un término similar actualmente aplicado a la computación portátil es *handheld*[6], referido principalmente a dispositivos como PDAs (Personal Digital Assistants), teléfonos móviles, etc., que en la práctica son ordenadores cuyo tamaño, peso e interfaz los hacen fáciles de transportar y utilizar en forma manual.

2.3.1.1 BUGs

Los BUGs son un ejemplo de dispositivos compactos de arquitectura modular. Su funcionalidad principal es como herramientas de prototipaje en la cadena de producción de otros dispositivos electrónicos. El paradigma en el cual se sustentan difiere del modelo tradicional en que, en este último, los prototipos son específicos al dispositivo que se pretende crear, y en general quedan inutilizados en etapas posteriores del desarrollo. Los BUGs se componen de una unidad de procesamiento básica de reducidas dimensiones, capaz de soportar un Sistema Operativo y que presenta a la vez una serie de bahías de expansión para periféricos, encontrándose una gran variedad disponible por parte de los propios fabricantes o incluso pudiendo ser desarrollados por terceros. De esta manera, la flexibilidad provista en cuanto a desarrollo de software y modularidad del hardware lo convierten potencialmente en cualquier dispositivo electrónico compacto que se tenga en mente. Una vez avanzado el diseño y llevadas a

6 El autor no ha encontrado una traducción satisfactoria para el termino.

cabo pruebas preliminares, el desarrollador puede seleccionar el hardware estrictamente necesario para el dispositivo objetivo (en base a los módulos y recursos utilizados en la unidad base) y comenzar con la construcción del mismo. Por otro lado, el BUG puede continuar siendo utilizado en el desarrollo de nuevas funcionalidades para el mismo producto o iniciar el desarrollo de un producto completamente diferente.

Las características de la unidad básica se encuentran en la Tabla 2.5, mientras que la misma se visualiza en la Figura 2.4. A modo de ejemplo, se listan las funcionalidades de distintos módulos disponibles en la actualidad:

- Radio 802.15.4 (ZigBee)
- E/S de audio
- Pantalla táctil LCD
- GPS
- Detector de movimiento y acelerómetro

Microprocesador	532Mhz ARM 11
Memoria RAM	128 MB SDRAM
Memoria flash interna	32 MB
Memoria externa	interfaz para tarjeta MicroSD (máximo 16GB)
Comunicación alámbrica	USB 2.0
Comunicación inalámbrica	Wi-Fi (802.11b/g) Bluetooth 2.0
Bahías de expansión	4
Sistema Operativo	Poky Linux 1.4 (2.6.29 kernel)
Batería	Li-ION (1100 mAh)
Tamaño	13 cm. x 6,5 cm. x 2 cm.

Tabla 2.5: Características del módulo básico BUGbase

Figura 2.4: BUGbase

2.3.1.2 Sensores

Uno de los módulos disponibles para los BUGs es el llamado BUGVonHippel[7]. Este consiste en una serie de interfaces a diferentes tipos de buses (como I2C, SPI) y un número de canales digitales de E/S estándar, lo que hace posible la interconexión a virtualmente cualquier otro sistema electrónico. Además posee:

4 convertidores análogo-digitales (Analog-to-Digital Converter, ADC) de 16 bits y una tasa máxima de 15 muestras por segundo.

2 convertidores digital-análogos (Digital-to-Analog Converter, DAC) de 8 bits y una tasa de 6.000 muestras por segundo.

De esta manera, el BUGVonHippel permite la medición de variables físicas a partir de transductores externos. A través de los drivers e interfaz de programación existentes, las mediciones son fácilmente accesibles hacia las aplicaciones de software. Mediante el uso de un módulo BUGVonHippel, y dadas las capacidades de procesamiento del BUG, este puede ser utilizado como un sensor inteligente. La Figura 2.5 muestra una captura del módulo.

Figura 2.5: Módulo BUGVonHippel.

2.3.1.3 Transceptores Inalámbricos

Otro de los módulos disponibles para el BUG es el BUGBee, un transceptor ZigBee. Este puede verse en la Figura 2.6. Sus características más relevantes se listan a continuación:

- Utiliza un modulo transceptor Matrix ZMXM-400 de la empresa CEL.
- Contiene el Z-Stack, implementación del stack ZigBee de la empresa TI.
- Tasa de datos máxima (nominal) de 250 Kbps.
- Banda de frecuencia 2,4 GHz.

7 nombrado en honor l Dr. Eric von Hippel de MIT, autor del libro *Democratizando la Innovación* (Democratizing Innovation)

- Alcance (nominal) en interiores: 15 m. (sin línea de vista).
- Alcance (nominal) en interiores: 450 m. (con línea de vista).
- Bajo consumo promedio: 130 mA a 3,3 V (0,43 W).

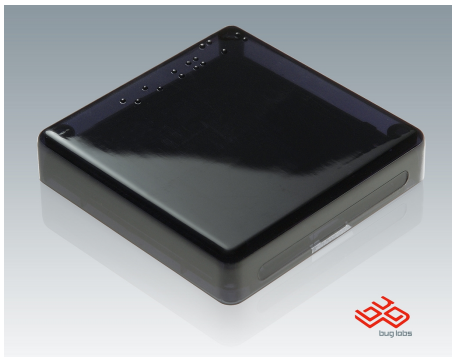


Figura 2.6: Módulo BUGBee

El módulo BUGBee permite la creación de una red inalámbrica enmallada, por lo cual es indicado para interconectar una red de sensores sin necesidad de infraestructura previa y con un bajo consumo de potencia.

2.4 Aspectos de Software

A continuación se realiza una descripción de los aspectos de programación relacionados con el trabajo realizado, en particular relacionados con la plataforma BUG.

2.4.1 Java

Los BUGs cuentan con un Sistema Operativo Linux (kernel 2.6), y es factible realizar aplicaciones en variados lenguajes, incluyendo C, C++, Perl y Python, entre otros. Sin embargo, tanto el soporte de BugLabs como la comunidad en línea apuntan fuertemente a la programación en el lenguaje Java. Esto no debe sorprender, pues Java es el lenguaje de programación más utilizado en el mundo (según [22]). El lenguaje Java difiere de otros en el sentido que los programas, al ser compilados, se traducen a un código "intermedio" (denominado "bytecode") que debe ser interpretado por un proceso de software conocido como Máquina Virtual de Java (Java Virtual Machine, JVM). Los BUGs cuentan con la máquina virtual phoneME, una variante de la plataforma Java Micro Edition. Java ME es una tecnología especialmente enfocada en dispositivos móviles, principalmente teléfonos celulares, los cuales cuentan con recursos de hardware mucho menores a los de un PC convencional. Es por esto que surge la necesidad de contar con una máquina virtual optimizada para ejecutar apropiadamente en estos dispositivos.

Por otro lado, la gran ventaja en la utilización de una máquina virtual es el enmascaramiento del hardware: el programa compilado ejecutará de la misma manera en

cualquier sistema donde exista una implementación de esta máquina. De esta manera, las aplicaciones escritas en Java son inmediatamente portables a una serie de sistemas operativos, incluyendo Linux, Windows y Mac OS.

2.4.2 OSGi + Concierge

Open Service Gateway Initiative (OSGi) es un set de especificaciones que definen un sistema para crear aplicaciones en base a la combinación dinámica (vale decir, en el momento de la ejecución o *runtime*) de diferentes módulos, los cuales pueden ser reutilizados. Fue creada y es mantenida por la OSGi Alliance.

Un sistema OSGi (es decir, aquel que implementa las especificaciones OSGi) debe proveer funcionalidades para el manejo de los módulos de software, o paquetes (*bundles*) como son llamados en la jerga OSGi. Por manejo se entiende la capacidad de añadir, remover y reemplazar módulos al momento de ejecución, manteniendo las relaciones y dependencias entre paquetes. OSGi provee también una Arquitectura Orientada a Servicios, puesto que la interacción entre paquetes se realiza mediante la publicación de/subscripción a servicios, que a su vez es manejada por un registro. La ventaja de utilizar OSGi es que resuelve el problema del control y consistencia de módulos, permitiendo el reemplazo dinámico de estos sin necesidad de detener las aplicaciones que los utilizan.

Concierge es una implementación de OSGi pensada para dispositivos limitados en recursos, y es utilizada en los dispositivos BUG. Es de código abierto, y puede ser implementada también en entornos "normales" como PCs convencionales.

2.4.3 Aplicaciones Preexistentes

A continuación se realiza una breve descripción de las aplicaciones existentes en la comunidad en línea de BugLabs, BUGnet, que son utilizadas como referencias para el presente trabajo.

2.4.3.1 BUGBeeChat

BUGBeeChat es una aplicación que utiliza el módulo BUGBee para establecer un enlace punto a punto entre 2 BUGs. La mayor parte de las funcionalidades de la red ZigBee son realizadas por la clase NetworkHandler, por lo que esta constituye una referencia importante. En particular, el código de NetworkHandler:

- Demuestra el uso de ciertos comandos que permiten interactuar con el stack ZigBee presente en el módulo transceptor del BUGBee.

- Inicializa la red (caso coordinador) o se une a ella (end device). No contempla el uso de routers.

- Crea un "bind" entre los dos dispositivos. Para esto registra un determinado perfil de

aplicación, cluster ID y endpoint (se utilizan los mismos en ambos extremos. En particular esto significa que el mismo cluster ID se registra como entrada y salida en ambos dispositivos).

- Provee las funciones receiveData(String) y sendDataMessage(String), para recibir y enviar texto entre los dispositivos.

2.4.3.2 vonHippelAnalogDemo

VonHippelAnalogDemo es una aplicación que demuestra el uso del modulo BUGVonHippel, en particular leyendo valores desde el conversor análogo digital. Provee funciones de alto nivel como:

- measureADC(int channel): recibe una palabra hexadecimal desde el ADC, especificando el canal a leer.
- measureADCVolt(int channel): realiza la calibración para transformar el dato obtenido a su equivalente en voltaje.

Capítulo 3 Diseño e Implementación

3.1 Requerimientos

Dentro de los objetivos del presente trabajo de memoria, el sistema concreto a implementar se encuentra principalmente determinado por los dos siguientes (ambos específicos):

- "Diseñar e implementar un prototipo de sistema que monitoree sensores (por ej. de temperatura) mediante dispositivos móviles. Como dispositivos móviles se utilizarán los dispositivos compactos de arquitectura modular denominados BUG."

- "Observar, aprender y aportar al desarrollo tanto de un producto innovador como a la empresa responsable del mismo. El trabajo realizado deberá percibirse como un aporte a dicho proceso."

El diseño del sistema partió estableciendo una serie de requerimientos a cumplir, los cuales se encuentran directamente alineados con los objetivos anteriormente señalados. Dichos requerimientos se describen a continuación:

- El sistema contemplará unidades con diferentes funcionalidades, Nodos Sensores y Monitores Móviles. Las características de los mismos se describen a continuación:

    - Nodos Sensores (NS): estos unidades deberán incorporar sensores de alguna variable física de interés y alguna forma de transmisión inalámbrica de la información obtenida.

    - Monitores Móviles (MM): se requiere de dispositivos móviles del tipo *handheld* que sean capaces de conectarse y recolectar la información de los Nodos Sensores, a la vez que presenten ésta ante los usuarios del sistema. Deberán almacenar la información recolectada desde los sensores, así como desplegarla visualmente. Por último, se requiere que informen al usuario de la composición de la red local de comunicaciones que se forme entre estos y otros dispositivos.

- La interconexión entre Nodos Sensores y desde estos hacia Monitores Móviles deberá poder realizarse en ausencia de cualquier infraestructura previa.

- El sistema en su totalidad deberá permitir la obtención, almacenamiento y visualización de los datos sensados de una manera simple por parte del usuario.

- Para dar cuenta del segundo objetivo específico, se buscará desarrollar el sistema de tal forma que resulte una innovación dentro del espectro de aplicaciones disponibles a la comunidad de desarrollo de BUGs. Deberá ser elaborado de tal manera que sea sencillo para otros desarrolladores crear aplicaciones que interactúen o hagan uso del mismo.

3.2 Diseño Conceptual

A continuación se realiza una descripción en alto nivel del diseño del sistema.

3.2.1 Comunicación NS – MM

La interconexión y traspaso de información entre los Nodos Sensores y los Monitores Móviles se realizará mediante un protocolo de comunicación inalámbrico que permita la creación de redes locales con una mínima cantidad de recursos de hardware. Esto pues la única forma de cumplir con el requerimiento de ausencia de infraestructura es que los nodos sensores y monitores sean capaces de formar su propia red de comunicación. Ante la potencial ausencia de suministro eléctrico, una segunda característica deseable es un bajo consumo energético. Finalmente, y tomando en cuenta el presupuesto y plazos del trabajo, el protocolo seleccionado deberá ser de fácil implementación en los BUGs.

3.2.2 Selección de equipos

En cuanto a los sensores, se requiere de unidades autónomas que incluyan transductores de alguna variable física de interés, como temperatura u otras. En base a estos requerimientos y al hardware disponible al proyecto, se decidió utilizar BUGs con módulos BUGBee para establecer la comunicación y BUGVonHippel más transductores análogos externos para captar señales del medio.

Se visualiza asimismo el uso de BUGs como Monitores Móviles. Se utilizarán los módulos de pantalla táctil para establecer la interfaz de visualización y como medio principal de interacción con la aplicación de monitoreo. Módulos BUGBee proveerán el acceso a las redes de sensores.

3.2.3 Visión

Tomando en cuenta el enfoque hacia la movilidad que presentan los dispositivos compactos, se percibe que el mayor valor del sistema se relaciona con aplicaciones también móviles, entendiendo estas como las que buscan satisfacer las necesidades del usuario en terreno, las que pueden diferir radicalmente de uno que trabaje con un computador personal. Este último puede tener a su disposición una serie de periféricos que facilitan la interacción con el computador (pantalla de “grandes” dimensiones, teclado, mouse), y un poder de procesamiento y almacenamiento considerable. Hasta el momento presente, lo opuesto es cierto para un dispositivo móvil, siendo su mayor ventaja el encontrarse en terreno y ser capaz de presentar información que se requiere en aquel momento y lugar. De esta manera, se apunta a un sistema orientado a la medición en tiempo real, entregando visualmente valores instantáneos recolectados de una red de sensores locales (si bien el almacenamiento no se deja de lado, este pasa a ser secundario). Dado que esta información debe ser desplegada en forma simplificada (por las dimensiones de la pantalla), el uso en redes de una muy alta cantidad de sensores pierde sentido; el sistema a desarrollar busca ser efectivo para un número reducido (decenas como máximo) de nodos. Principalmente se apunta a redes de sensores cuyos valores describan apropiadamente una situación con tan solo una observación rápida. Dado esto, tampoco se buscará maximizar medidas

de performance tradicionales en monitoreo, como la tasa de muestreo o precisión, por lo que pueden existir ahorros en ancho de banda y costo de los equipos.

Lo anterior puede comprenderse mejor al visualizar un caso de uso concreto: una vivienda con algunos sensores de temperatura, humedad e iluminación. El alcance de la red puede circunscribirse a esa misma vivienda. De esta manera, el usuario portador del Monitor Móvil puede conocer con una rápida mirada el estado de la casa, como está descrito por las variables sensadas ([8]). Volviendo a lo señalado sobre la precisión y tasa de medición, puede verse que en este escenario "doméstico", conocer la temperatura de una habitación a 1 muestra por segundo y con precisión de +-0,5°C es esencialmente igual de efectivo que hacerlo a 10.000 muestras por segundo y con precisión de +-0,0005°C. El ejemplo anterior se esquematiza en la Figura 3.1.

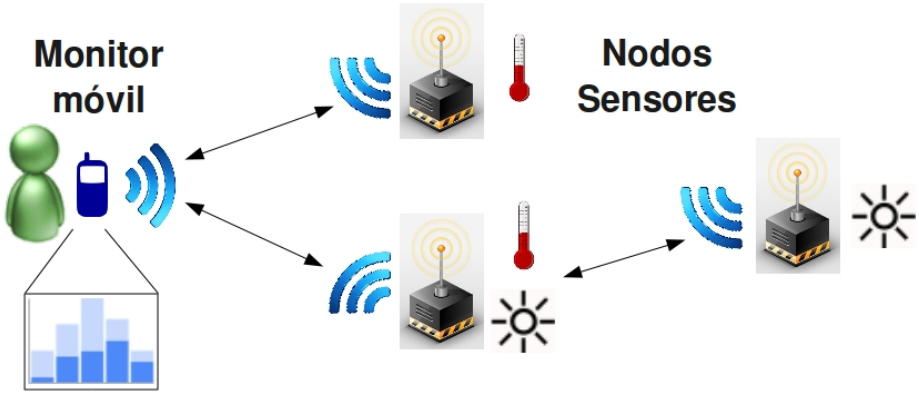


Figura 3.1: Caso de uso: monitoreo de variables ambientales en red con pocos sensores.

3.3 Implementación

3.3.1 Lógica de aplicación en red

Para la implementación de las funcionalidades de comunicación del sistema, además de la creación de los mensajes a intercambiar entre los nodos para desempeñar las funciones requeridas, se requirió tomar en cuenta características particulares de como el sistema sería usado en terreno. Entre los rasgos distintivos de una red de sensores, aquellos que más incidieron en el desarrollo del presente trabajo son los siguientes:

- **Desconexión y re-conexión de nodos:** se apunta a la creación de un sistema en que los nodos puedan entrar a la red en cualquier momento, así como salir de la misma, y aun más volver a conectarse. Esto aplica principalmente a los monitores, que siendo dispositivos móviles, se espera sean portados por un usuario. Este último puede variar su posición con respecto a la red se sensores, entrando y saliendo del alcance de la misma, a veces continuamente (ej. red de sensores implementados en una habitación particular de una

8 Este caso de uso corresponde a uno de domótica o automatización casera, campo que se percibe como interesante de explotar mediante este desarrollo.

casa).

- **Mínima lógica posible en los sensores:** tomando en cuenta que una red de sensores puede querer escalarse a cientos o miles de sensores, es necesario que las unidades presenten un costo reducido y requieran de la menor mantención posible. En la práctica, los nodos de estas redes tienden a ser muy simples en hardware, incorporando el transductor de la variable de interés, interfaz de radio y la mínima lógica posible para enlazar estos dos subsistemas. Además utilizan baterías y se trata de reducir el consumo lo más posible de manera de espaciar el tiempo entre intervenciones humanas. Si bien en el sistema actual se utilizarán BUGs como sensores (pues no es el propósito diseñar nuevas unidades de hardware), los cuales tienen un poder de procesamiento considerable, se deberá apuntar a estos principios ya probados en la práctica. De esta manera, se tratará de que la aplicación para los Nodos Sensores sea tan simple como se pueda.

En un inicio, se trató de responder al primer requerimiento implementando un sistema seguro de mensajería (en el sentido de asegurar que los mensajes son recibidos), pero al poco tiempo el autor detectó que esto no podría llevarse hasta un nivel funcional sin alcanzar un alto grado de sofisticación (la sensación fue de estar reescribiendo el protocolo TCP). Esto se tornaba infactible para el trabajo actual a la vez que no respetaba el segundo requisito. De esta manera, se dio un giro hacia un sistema sin seguridad en la recepción de los mensajes, sino basado principalmente en la repetición continua de mensajes. Si existen condiciones mínimas de funcionamiento (canal físico de comunicación), la data será recibida adecuadamente. De no ser así, es posible que haya data descartada y mensajes sin respuesta, pero esto no afectará el posterior funcionamiento del sistema. En particular, los monitores son los encargados de repetir continuamente mensajes para:

- Conocer la infraestructura de la red: paquetes de *broadcast* que solicitan a todos los nodos de la red enviar al emisor su información de red, función (sensor, monitor) y otros identificadores.

- Requerir muestras de un sensor en particular: paquetes *unicast* enviados desde un monitor a un sensor requiriendo el envío de una muestra.

Los sensores únicamente se encargan de responder ante las solicitudes de muestreo, lo que redunda en una aplicación muy simple para estos nodos. La principal ventaja de la repetición continua de mensajes para conocer el estado de la red es que esto permite contar con información actualizada. La principal desventaja es que ciertamente existe un impacto en el ancho de banda, por la emisión periódica de *broadcasts*. Sin embargo, esto no se considera un problema puesto que la carga de datos es siempre baja, habiéndose planteado un sistema en que la tasa de adquisición no es crítica y donde, por lo tanto, el canal de comunicación se encuentra mayormente libre. También existe repercusión en el consumo de energía. Un esquema alternativo en este sentido sería uno en que los sensores estén programados para enviar muestras a intervalos regulares, sin tener los monitores que requerir estas cada vez. Esto significaría utilizar la mitad de los mensajes para el item recolección de muestras. Sin embargo, no resulta evidente que este esquema sea superior, puesto que un sensor podría quedar aislado de la red y aun enviar las muestras, agotando sus baterías innecesariamente. En contraposición, si se pregunta por cada

muestra, esto no ocurriría. Puediendo existir un sinfín de otros enfoques, se vislumbra que el sistema de mensajería óptimo en cuanto a la potencia es una función de la situación particular de la red. Un estudio en mayor profundidad de este aspecto escapa a los objetivos de esta memoria.

3.3.2 Protocolo de comunicación

A continuación se describe cronológicamente el camino realizado hasta seleccionar el protocolo utilizado en el sistema final. Esto se hace en el ánimo de demostrar el trabajo conjunto con la empresa desarrolladora, acorde a uno de los objetivos específicos.

El desarrollo del sistema inició asumiendo la factibilidad de utilizar el protocolo ZigBee, mediante los módulos BUGBee del BUG. Sin embargo, se detectó una carencia en el driver del mismo que no permitía direccionar paquetes a más de un único host, limitando en efecto las redes posibles de establecer a solo dos nodos. Se evaluó en conjunto con personal de Bug Labs Inc. la posibilidad de completar las funcionalidades faltantes en el driver, ya fuese por parte de ellos o con la ayuda del autor. Sin embargo, los recursos de la empresa se encontraban enfocados en el lanzamiento de un nuevo producto, el BUG 2.0. Por otro lado, en el contexto inmediato no se tenían planes de continuar la venta y soporte de los módulos BUGBee. Evaluando esta situación, el autor propuso la alternativa de utilizar radios XBee en conjunto con los BUGs. Estos dispositivos son sumamente populares en el mercado de las redes de sensores, por lo que existe una gran cantidad de aplicaciones disponibles; ciertamente más que aquellas relacionadas con el integrado Texas Instrument CC2430, que constituye el corazón del módulo BUGBee. En particular, la existencia de librerías en Java ([24], [25]) para la comunicación con estos módems permite guardar consistencia con el ambiente de programación propuesto por la compañía, basado en este lenguaje.

Por otro lado, al unirlos a los BUGs mediante adaptadores externos (ver Figura 3.2 y Figura 3.3) surge la ventaja de poder reemplazar las radios fácilmente. Existen versiones que soportan el protocolo IEEE 802.15.4 y otras el stack ZigBee. Además, para cada una de estas, existen variedades de potencias de transmisión y antenas o conectores. Las anteriores son suficientemente compatibles en hardware como para que todas puedan utilizarse con el mismo adaptador, entregando una gran flexibilidad al usuario. Los módems se comunican con otros dispositivos locales a través del protocolo serial (RS232), el cual los adaptadores convierten a USB. Al conectar uno de estos dispositivos, el kernel Linux presente en los BUGs automáticamente los hace disponibles como una terminal TTY. Desde el punto de vista de la aplicación, la comunicación con el módem es tan sencilla como escribir y leer desde un *stream* de datos, en forma similar a como se hace con un archivo de texto convencional.

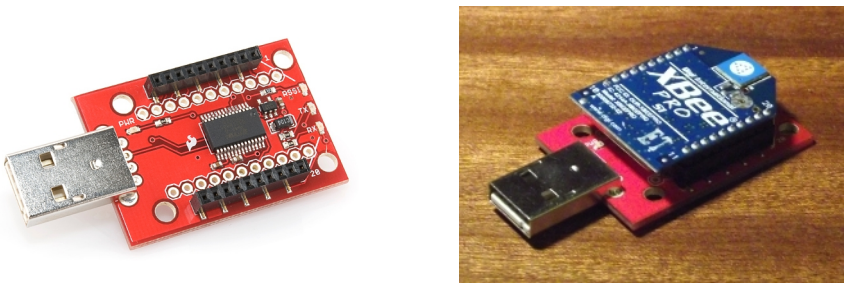


Figura 3.2: a) Adaptador XBee - USB de Sparkfun [26]. b) Adaptador con radio Xbee.

Se adquirieron las radios y adaptadores y rápidamente se comprobó la compatibilidad de este montaje con los BUGs. Una vez realizado esto, y dada las numerosas ventajas y premura por continuar con el trabajo, se seleccionó esta alternativa como hardware para la interconexión. Entre las variantes disponibles, se optó por aquellas que soportan el protocolo IEEE 802.15.4, por ser el desarrollo de aplicaciones en base a este más simple, según la literatura y posterior estimación del autor. El mismo permite como máximo la formación de topologías en estrella, con un único coordinador (y router) al centro de la red, y no de redes enmalladas propiamente tal. Sin embargo, esto aún se consideró consistente con el caso de uso propuesto, al menos mediante dos configuraciones distintas:

- **Sensor-Coordinador:** en esta configuración, uno de los dispositivos sensores ejerce la función de coordinador de la red, tarea que es llevada a cabo completamente por el stack contenido en el dispositivo Xbee. Nuevos nodos pueden añadirse a la red simplemente entrando en el rango de transmisión del mismo (tanto sensores como monitores). Esto último también es un comportamiento automático de las radios XBee: el que los dispositivos no coordinadores busquen constantemente asociarse a una red. Las limitantes de esta configuración son que el coordinador debe permanecer siempre encendido (para evitar el que la red se forme sucesivas veces), poniendo mayor exigencia sobre el hardware de dicho sensor en particular, y también que para integrarse a la misma se debe estar al alcance de este único dispositivo.

- **Monitor-Coordinador:** un monitor es el encargado de formar la red, mientras que los sensores buscan integrarse a la misma permanentemente. Esto tiene lugar cada vez que el monitor se encuentra en el rango de transmisión de los mismos.

En ambos casos anteriores, la potencia utilizada en el tiempo en que los equipos utilizan buscando asociarse a una red puede disminuirse utilizando la opción de *dormir* (pasar a un estado de baja actividad) periódicamente, lo que puede configurarse en el hardware.

3.3.2.1 Mensajes

Dada la capacidad de realizar broadcast, así como de direccionar fácilmente a múltiples otros hosts, se seleccionó el modo API (ver Radios XBee en Capítulo Marco Conceptual) para las radios XBee. Otra razón importante fue el que la librería xbee-api (ver sección Librerías, en este capítulo) requiere del uso de este modo.

Se utilizan principalmente 2 tipos de paquetes API en la aplicación, con soporte en pruebas para un tercero. Los dos principales son los de comandos AT y de transmisión de datos mediante direcciones cortas (direcciones de 16 bits). En el primer caso, el comando soportado actualmente a nivel de aplicación (muchos comandos son procesados directamente por el módem, sin necesidad de intervención desde el código del usuario) es el Node Discovery o "ND" (ver Monitor, más adelante). En cuanto a los paquetes de datos, se adoptó una convención en que la data de cada paquete incluye in número ID de instrucción, y luego argumentos de ser necesario, separados estos campos por comas (","). Los mensajes implementados en este caso corresponden a los de solicitud de muestra, por parte de los monitores, y de respuesta a éste con una (única) muestra, por parte de los sensores.

El tercer tipo de paquete (experimental) que es posible recibir es aquel específico para muestras de entrada/salida al módem (pues los XBee cuentan con ADCs y es posible configurarlos para que envíen muestras automáticamente). El mismo exhibe un formato propio para contener las muestras, pudiendo agrupar varias en una misma transmisión. La ventaja de soportar este tipo de

paquetes es que en un futuro, el sistema podría operar indistintamente con BUGs o nodos basados únicamente en radios XBee como sensores, de costo drásticamente menor.

3.3.3 Aplicación

Tomando en cuenta la orientación de la plataforma BUG hacia el lenguaje Java, se utilizó éste para programar la aplicación. Esto permitió modelar los nodos como clases, y crear subclases de los mismos para representar a monitores y sensores. Esto permite, entre otras cosas, intercambiar muy fácilmente la función de un mismo dispositivo entre sensor y monitor, utilizando el mismo paquete de software, simplemente invocando diferentes objetos. Por último, se creó la clase NodeImage, una versión simplificada de la clase nodo (conservando varios de los campos pero ningún método), que es usada por la clase monitor para guardar representaciones "livianas" de los nodos presentes en la red. A continuación se presenta una descripción de estas clases:

3.3.3.1 Clase Node

Esta clase modela un nodo genérico, tanto en sus funciones como parámetros de red. No está diseñada para ser invocada por el usuario, sino como ancestro de las clases de nodos específicas detalladas más adelante. Campos fundamentales de esta clase son:

- Tipo de Función (char functionType): se refiere a la función de sensor o monitor.
- Tipo de Red (char networkType): se refiere a la función de la radio Xbee como coordinador o dispositivo final.
- Identificador ASCII (String asciiIdentifier): las radios Xbee poseen un registro llamado "Node Identifier" (NI), capaz de guardar un *string* identificador de 20 caracteres ASCII. Se utilizó la convención de escribir los 2 primeros caracteres con los dos campos anteriormente nombrados (razón por la cual estas variables son de tipo "char"), y lo restante con un identificador particular a cada nodo, correspondiente al asciiIdentifier.

  A continuación se muestra un ejemplo de la lógica utilizada: Nodo es sensor y coordinador, mide variable temperatura:

- functionType = 'S'
- networkType = 'C'
- asciiIdentifier = "Temp"
- NI = "SCTemp"

Esta información es obligatoria para los nodos, puesto que es utilizada al decodificar los paquetes de requerimiento de información de red (ver clase Monitor). El principal método de la clase Node es joinNetwork(). Este es el encargado de iniciar el objeto Xbee (ver xbee-api en sección Librerías) y leer y/o configurar registros del módem necesarios tanto para sensores como monitores.

3.3.3.2 Clase Monitor

La clase monitor tiene entre sus campos más importantes mapas (java.util.HashMap) donde guarda objetos NodeImage que representan a los monitores y sensores de la red, siendo la llave del mapa la dirección corta o de red, de 16 bits, de cada nodo.

La clase implementa la interfaz Runnable, por lo que puede ser invocada como un thread. El método run llama a joinNetwork() (heredado de la clase Node), luego construye la GUI e inicia un timer en el que agenda las tareas SampleTask() y NetworkUpdateTask(). La GUI del monitor se basa en la clase Jtree del paquete Swing. La raíz del árbol corresponde conceptualmente a la red inalámbrica a la que se está asociado. Luego existen 2 categorías: "Monitores" y "Sensores", cuyos hijos corresponden a los nodos detectados en la red diferenciados según función. Al agregarse o quitarse nodos de los mapas de la clase monitor, estos son agregados o removidos automáticamente del árbol gráfico, con lo cual el usuario siempre observa una representación de la red con la información más actualizada disponible. Las etiquetas de los nodos despliegan sus networkType y asciiIdentifier. En el caso de los sensores, se muestra además la última muestra recibida y si ésta es o no actual (ver SampleTask(), a continuación).

SampleTask() es una rutina que envía mensajes *unicast* a todas las direcciones de los NodeImage del mapa que contiene los sensores. Estos mensajes contienen un código de instrucción que representa el requerir una muestra. Los NodeImage poseen un campo boolean denominado dataIsFresh, que se cambia a *false* cuando se envía un nuevo mensaje de requerimiento de muestra (processResponse() cambia estos valores a true cuando se reciben muestras nuevas).

NetworkUpdateTask() es el encargado de enviar mensajes *broadcast* requiriendo la información de red de todos los nodos presentes en ésta. Esto se realiza mediante el comando AT "ND" (Node Discovery). Simplemente enviando este comando al módem, éste se encarga de realizar la transmisión múltiple y recibir las respuestas. Cada una de éstas contiene la siguiente información:

- MY (dirección corta o de red de la fuente, 16 bits)
- SH (32 bits MSB de la dirección IEEE, larga o numero serial de la fuente)
- SL (32 bits LSB de la dirección IEEE, larga o numero serial de la fuente)
- DB (intensidad de señal con que el mensaje fue recibido, en dB)
- NI (String "Node Identifier")

Todos los mensajes se reciben de manera asíncrona, y por lo tanto la mejor manera de procesarlos sin bloquear la aplicación es con un método *Listener,* que sea invocado por la API de Xbee ante la recepción de un paquete. Es por esto que la clase monitor también implementa la interfaz PacketListener, y en el método run se invoca:

xbee.addPacketListener(this);

El método principal de la clase Monitor es justamente aquel que procesa los paquetes, processResponse(...). Este se basa principalmente en instrucciones if y switch que van filtrando el tipo de paquete recibido, para luego implementar las acciones adecuadas. A continuación se muestra una versión simplificada (pseudocódigo):

```
Map<XBeeAddress16, NodeImage> sensores;
Map<XBeeAddress16, NodeImage> monitores;

DefaultMutableTreeNode categoriaSensores;
DefaultMutableTreeNode categoriaMonitores;

public void processResponse(XBeeResponse response) {
      if (!response.isError()) {
            apiId = response.getApiId();

            if (apiId == AT_RESPONSE) { //paquete con comando AT
                  if (comando == "ND") {
                        informacion_de_la_respuesta = respuesta.getInformación; //incluye
shortAddress
                        if (functionType == SENSOR && sensores.containsKey(shortAddress)) {
                              //"reset" watchdog
                              sensores.get(shortAddress).watchDogCounter = 0;
                        } else if (functionType == MONITOR &&
monitores.containsKey(shortAddress)) {
                              //"reset" watchdog
                              monitores.get(shortAddress).watchDogCounter = 0;
                        } else {
                              //crear nuevo NodeImage
                              NodeImage nodeImage = new NodeImage(informacion_de_la_respuesta);
                              if (imageFunctionType == MONITOR) {
                                    monitores.put(shortAddress, nodeImage);
                                    categoriaMonitores.agregarNodo(nodeImage); //añadir a la GUI
                              } else {
                                    sensores.put(shortAddress, nodeImage);
                                    categoriaSensores.agregarNodo(nodeImage); //añadir a la GUI
                              }
                        }
                  }
            } else {
                  log("Ignoring unexpected AT response: " + response);
            }
      } else if (apiId == RX_16_RESPONSE) { //paquete de datos con direcciones de 16 bits
            StringTokenizer tokens = new StringTokenizer(data,",");
            short instructionId = Short.valueOf(tokens.nextToken());

            switch(instructionId) {
            case REPLY_GET_SAMPLE: {
                  float dataPoint = Float.valueOf(tokens.nextToken());
                  String timeStamp = tokens.nextToken();

                  if (sensores.containsKey(sourceAddress)) {
                        sensores.get(sourceAddress).storeDataPoint(dataPoint, timeStamp);
                        frame.repaint();
                  }
                  break;
            }
            }
      } else {
            log("Error", response.getException());
      }
}
```

Como puede observarse, se separan los comandos AT de los mensajes de datos. En el primer caso, el único comando AT implementado hasta el momento es ND, por lo que se aceptan solo éstos y se parsea la información de la respuesta, con lo que se crea un nuevo NodeImage, se agrega al mapa correspondiente y a la interfaz gráfica.

En los mensajes de datos, se parsea el contenido para encontrar el ID de la instrucción. Al momento, el único válido a recibir por un monitor es una respuesta a su requerimiento de muestra, por lo que se procesa ésta y utiliza el método de la clase NodeImage correspondiente para almacenar el dato.

Aunque no se muestra, se implementó código para recibir las respuestas con apiId == RX_16_IO_RESPONSE, las muestras que pueden ser enviadas automáticamente desde el módem Xbee, sin mediar ninguna aplicación. Este resulta muy similar a apiId == RX_16_RESPONSE, con la salvedad que el timestamp es agregado por el Monitor al momento de la recepción (lo que podría resultar en discrepancias al ser recibido un mismo dato por diferentes monitores) y que la muestra es forzosamente un entero (int).

Una interacción que abarca varios métodos es la de remoción de los nodos que dejan de responder a los comandos de Node Discovery. Al crearse una nueva NodeImage, el constructor de ésta inicia un contador "watchdog" en 0. Cada vez que NetworkUpdateTask() envía un mensaje "ND", se encarga de aumentar el contador y compararlo contra un máximo preestablecido; el nodo es eliminado del mapa correspondiente (única referencia al objeto) si alcanza este valor. Por otro lado, processResponse(...) es el método encargado de decrementar el contador de cada nodo previamente existente, al recibir una respuesta "ND" del mismo.

3.3.3.3 Clase Sensor

La clase sensor también se une a la red (joinNetwork()) en su método run y luego simplemente escucha por paquetes, implementando el mismo método processResponse, sin ejecutar tareas "por iniciativa propia". A diferencia del Monitor, no procesa paquetes AT sino solo de datos (apiId == RX_16_RESPONSE), y espera al ID de instrucción "Requerimiento de Muestra" para responder con un paquete con el siguiente string en el campo data:

```
REPLY_GET_SAMPLE + "," + String.valueOf(getSample()) + "," + timeStamp();
```

La instrucción getSample() es la encargada de recoger la muestra del sensor, mientras que timestamp() utiliza y formatea adecuadamente el tiempo del sistema.

3.3.3.4 Clase NodeImage

Esta clase consiste en una versión liviana de la clase nodo, de modo de conservar los campos esenciales que representan a un equipo, pero sin implementar operaciones como unirse a la red u otras. Añade los campos DefaultMutableTreeNode nodoGrafico y String tag, ambos utilizados para representar al NodeImage en el árbol de la interfaz gráfica.

Es de uso interno de la clase Monitor y, además del constructor, implementa el método storeDataPoint, encargado de almacenar las muestras recibidas, incluyendo el timestamp asociado. Para esto se optó por utilizar el sistema log4j (ver sección Librerías), pues presenta una estructura muy completa para logeo de información y tiene la capacidad de crear archivos con un tamaño máximo, luego del cual se abren otros nuevos, quedando los anteriores como respaldo (se intentó, sin éxito, el uso de una base de datos MySQL en los BUGs).

3.3.4 Librerías

La aplicación hace uso de las siguientes librerías: xbee-api [24], log4j [27] y rxtx [28]. Cabe resaltar que todas ellas son proyectos "open source". En lo que sigue se realiza una breve descripción de cada una y del rol que desempeñan en el sistema.

3.3.4.1 Xbee-api

Esta librería presenta una abstracción en alto nivel, en lenguaje Java, de la comunicación con las radios Xbee. Enmascara varios de los mensajes posibles en clases con métodos convenientes. La clase básica es Xbee, y el método para invocar la comunicación serial con una radio es open(...), como se ejemplifica a continuación:

```
XBee xbee = new XBee();
xbee.open("/dev/ttyUSB0", 9600); //open(String port, int baudRate)
```

En lugar de tener que leer cada byte recibido en la interfaz serial, esta librería ofrece métodos como getResponse(), que retorna un objeto XbeeResponse, el que a su vez presenta métodos como getApiId(), getPacketBytes(), getChecksum(), entre otros. Dada la gran variedad de paquetes existentes (comandos AT, paquetes de datos, muestras, etc.), es común realizar "cast" sobre XbeeResponse para obtener objetos cada vez más específicos. Si bien getResponse() es un método bloqueante, existe la posibilidad de implementar la interfaz PacketListener, que es invocada por un thread, con lo que evita que la aplicación se detenga a la espera de un mensaje. Xbee-api tiene como dependencias a log4j y rxtx. Además, requiere de librerías que no se encuentran presentes en la máquina virtual phoneME, la estándar del BUG, por lo que su uso requiere de un cambio de máquina virtual a openJDK, también disponible para BUG, aunque con menor performance. Al momento del diseño se analizaron todos estos requerimientos, incluso en conjunto con personal de Bug Labs Inc., y se decidió que ninguno de ellos constituía una limitante. El autor tomó la decisión de utilizar esta librería por considerarla una de las más completas para gestión de Xbees disponibles en el lenguaje Java.

3.3.4.2 log4j

Provee funcionalidades de "logging" dinámico y mensajes con niveles de importancia diferenciados. De esta manera, puede controlarse al momento de la ejecución la profundidad y cantidad de información que se desea recibir. La clase básica de esta librería es Logger, cuya principal interfaz de "impresión" de los mensajes es mediante lo métodos trace, debug, info, warn, error y fatal. Por ejemplo:

```
Logger log = Logger.getLogger(this.class);

log.debug("String 1"); //este mensaje serÃi impreso solo cuando el logger se
encuentre en nivel “debug”
log.fatal("String 2"); //este mensaje serÃi impreso solo cuando el logger se
encuentre en nivel “fatal”
```

Tiene un poderoso editor de formato que permite entregar, en cada mensaje información como tiempo del sistema, tiempo de ejecución, nombre de la clase, nivel de logging, entre otros. A continuación se presenta un ejemplo:

[2010-11-18 03:14:17,153] [pool-1-thread-1] [INFO] [wsnmonitor.app.Monitor] String 1

Además presenta la funcionalidad de redirigir la salida a archivos y/o la terminal del sistema según convenga, y permite indicar el tamaño máximo de los archivos. Una vez alcanzado este su nombre es cambiado según un protocolo definido, quedando como respaldo, mientras que el logeo continua en un archivo nuevo. Inicialmente se utilizó para cumplir con las dependencias de la librería xbee-api, pero posteriormente se incorporó su uso como método de almacenamiento para la aplicación realizada. Esta librería había sido previamente portada a la plataforma BUG [29], aunque se necesitó de modificaciones mínimas.

3.3.4.3 rxtx

Es una librería de Java nativa (JNI) que provee una abstracción de la comunicación mediante puertos seriales o paralelos. Apunta a proveer las funcionalidades de la interfaz de Java Communications API (CommAPI). Es una de las dependencias de xbee-api, puesto que la comunicación con las radios Xbee se realiza en forma serial. Al igual que log4j, había sido previamente portada a la plataforma BUG [30].

3.3.5 Componentes del Sistema

Finalmente, el sistema queda compuesto por los siguientes elementos:

- Monitor
    - BUG + módulo BUGview + módulo BUGVonHippel + radio XBee + adaptador Xbee-USB
- Sensor
    - BUG + módulo BUGVonHippel + radio XBee + adaptador Xbee-USB + sensor análogo

Los anteriores pueden apreciarse en conjunto en la Figura 3.4 y Figura 3.5.

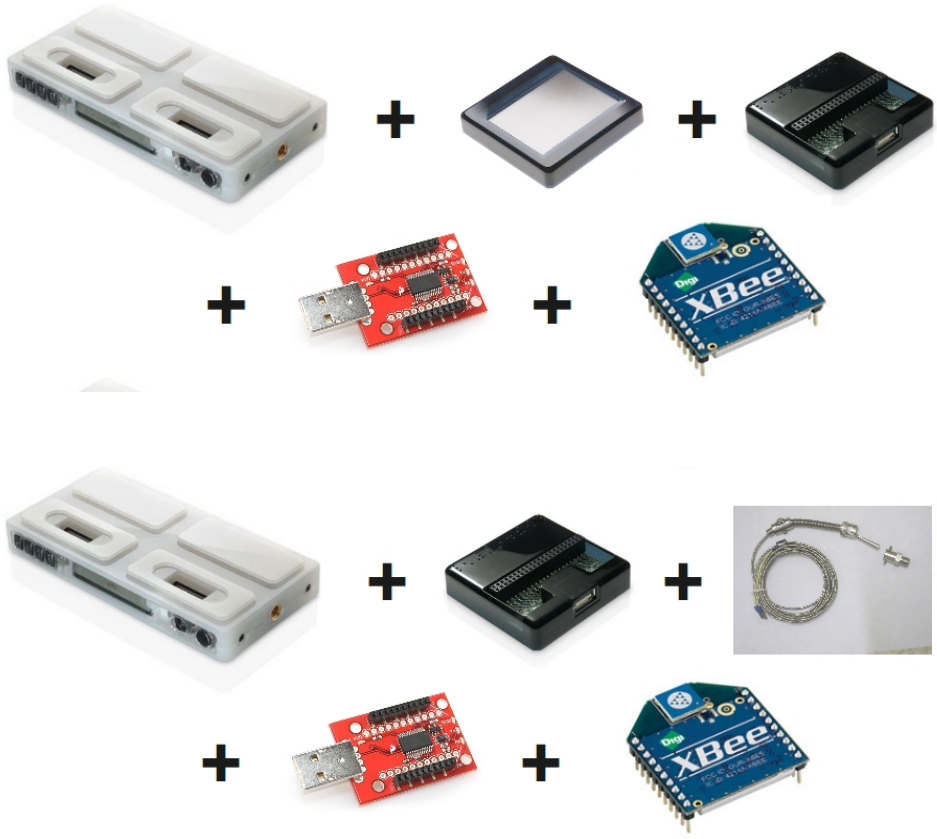


Figura 3.5: Componentes Nodo Sensor (*).
(*) se muestra una termocupla como ejemplo de sensor análogo

3.3.6 Instalación

Debe señalarse que, para la correcta implementación del sistema, se requiere de la configuración previa de las radios XBee con las siguientes opciones:

- Ambas radios: AP=2 (por requisito de xbee-api)
- Radio del Coordinador: CE = 1 (puede ser monitor o sensor)
- Radio de los demás dispositivos: MY = XXXX (algún numero único en el sistema)

Esto puede realizarse mediante la utilidad X-CTU [34], provista por Digi [9].

Por otro lado, debe intervenirse el software (específicamente el archivo WSNMonitorServiceTracker.java) para indicar la aplicación deseada (Sensor o Monitor), el identificador ASCII del nodo, el *path* del dispositivo XBee en el sistema (ej. /dev/ttyUSB0) y el *baudrate* para la comunicación serial con dicho dispositivo.

9 Mientras que la librería xbee-api tambien permite realizar esta programación, no se utilizó esto en la aplicación presente, por considerarse aun experimental en opinión del autor.

3.3.7 Especificaciones

El sistema, según su diseño, cumple con las especificaciones nominales contenidas en la Tabla 3.1.

Interfaz de usuario:	Interfaz gráfica mediante pantalla táctil LCD (módulo BUGview de los BUG)
Tasa de muestreo de datos:	Limitada a ~3 muestras por segundo, por ADC presente en módulo BUGVonHippel y driver del mismo.
Precisión de muestras:	16 bits.
Tipos de sensores:	Transductores con salida de voltaje análogo, con un rango máximo de 2,5V con respecto a GND o 5 V diferenciales ([10]).
Interfaz de radio:	Radios XBee con el protocolo IEEE 802.15.4 ([11]).
Alcance:	Depende del módem XBee utilizado. Rango entre 30 m. en interiores (sin línea de vista) en versión de baja potencia, hasta 1,6 Km. en exteriores (con línea de vista) en versión de alta potencia.
Tasa de transmisión de datos:	250 Kbps.

Tabla 3.1: Especificaciones nominales del sistema diseñado.

10 La curva de conversión a las unidades de interés, si se desea, debe ser ingresada por el usuario en el programa.

11 Librerías utilizadas permiten el uso de radios XBee ZB, con stack ZigBee. En cuanto a hardware, solo se requiere del cambio de la radio, pudiendo utilizarse el mismo adaptador USB. Sin embargo, esto NO se ha probado.

Capítulo 4 Relación Memoria-Empresa

Uno de los objetivos específicos del trabajo presente lee:

"Observar, aprender y aportar al desarrollo tanto de un producto innovador como a la empresa responsable del mismo. El trabajo realizado deberá percibirse como un aporte a dicho proceso."

Esto en relación a Bug Labs Inc., y su plataforma BUG. La empresa se ubica en New York, USA, y al momento de la escritura tiene alrededor de 4 años desde su fundación, y aproximadamente 2 desde que su primera plataforma, los BUGs 1.2, salieron al mercado. La motivación inicial del fundador y C.E.O. de la misma, Peter Semmelhack, era la de innovar con un dispositivo que permitiera a cualquier persona con conocimientos de programación, pero no necesariamente de diseño de hardware, construir su propio dispositivo electrónico. Lo anterior inserto en la filosofía "open", por lo que todos los programas y hardware desarrollado son completamente abiertos a la comunidad. Recientemente, la empresa ha introducido el concepto de "Wireless Innovation Platform". Este apunta a que comunidades de desarrolladores de software, especialmente aquellos dedicados al segmento móvil (ej. *smartphones* Iphone o terminales con Android), se abran también al desarrollo de hardware y creen sus propios dispositivos. Esta estrategia ha valido a BugLabs contratos con 3 de los principales 4 operadores móviles en USA (Verizon, AT&T y Sprint). Por otro lado, durante el desarrollo de este trabajo, Bug Labs Inc. se aprontaba al lanzamiento de su la versión 2.0 de los BUGs, completamente rediseñada y con mejores especificaciones que la primera.

Es por esto que, en opinión del autor, la interacción con la empresa se dio en un periodo en que la misma estaba dejando de ser una *startup* y pasando a ser validada por el mercado.

Las relaciones con la misma iniciaron a través de una relación profesional de larga data entre el profesor guía del trabajo, Eduardo Vera, y Maurizio Arienzo, V.P. De Bug Labs Inc.. En un inicio, se estableció un grupo de desarrollo en Chile, en particular en la Universidad, con algunos alumnos de Ing. Civil Eléctrica. El autor de este trabajo puede considerarse como parte de la segunda generación de dicho grupo de desarrollo.

Hitos importantes en las relaciones con la empresa son:

- Desarrollo de este trabajo de memoria, utilizando su plataforma.
- Desarrollo desde el Centro de Modelamiento Matemático, para Codelco, de un proyecto de monitoreo móvil con BUGs.
- Taller de desarrollo con BUGs a Codelco, con posterior exposición de Maurizio Arienzo.
- Reunión, en conjunto con Maurizio Arienzo, para presentar propuesta comercial a operador móvil en Chile.

- Estadía de un mes en Bug Labs Inc., en New York, relacionada con proyecto Codelco y trabajo de título.

Durante la estadía, se sostuvo una jornada laboral normal, sumándose al equipo de desarrolladores de Bug Labs Inc., pero con foco en una aplicación de monitoreo de sensores. Se hizo especial hincapié en la determinación del uso o no del módulo BUGBee, selección de sensores y componentes a utilizar y revisión de software disponible y su adaptación para uso en el sistema.

El proyecto realizado para Codelco, enfocado en el monitoreo móvil de variables medioambientales en minas subterráneas, surgió en forma posterior al inicio de la memoria. Sin embargo, y por considerarse los objetivos como compatibles, se ha desarrollado a la par del trabajo presente, aportando a la experiencia formativa del alumno. En particular, el financiamiento del viaje a New York fue provisto dentro del marco del acuerdo con Codelco, así como una beca de memorista en la misma compañía.

Este último punto impone una restricción de propiedad intelectual sobre el trabajo realizado que impide al autor liberar la aplicación desarrollada en forma completa, en particular en la comunidad de desarrollo BUGNet, de BugLabs. En pos de no vulnerar esta relación, pero con el objetivo de aportar a la comunidad, el autor liberó en dicha comunidad una versión de la librería xbee-api ([12]) portada a la plataforma BUG [31].

Adicionalmente, se sostuvo una reunión con el C.E.O. y V.P. de Bug Labs Inc. para evaluar el alcance de lo realizado durante la estadía.

12 Libre bajo la licencia GNU General Public License

Capítulo 5 Pruebas y Discusión

5.1 Pruebas

Una aplicación destinada al monitoreo de redes de sensores puede tener una gran variedad de casos de uso, y distintos eventos ante los cuales el sistema debe responder. Sin embargo, los recursos para realizar pruebas de validación, principalmente tiempo y hardware disponible, son finitos y por lo tanto deben ser racionalizados.

Al momento de realizar las pruebas, se contaba con 3 sets de radios XBee con adaptadores USB, esenciales para la interfaz con los BUG. La intención inicial era realizar una prueba alineada con el caso de uso, donde existiera un monitor y dos sensores y se apreciara, además de la funcionalidad básica, eventos como desconexiones de sensores.

Sin embargo, a la hora de llevar la aplicación a los BUGs físicos (se realizó mucho del trabajo con los "virtual BUG", emuladores presentes en el SDK Dragonfly de Bug Labs Inc.) se encontró un grave problema, relacionado con el cambio de máquina virtual de los mismos a openJDK (requisito de xbee-api). En un comienzo, aplicaciones muy simples para probar el funcionamiento de la librería no demostraron conflicto (en etapas intermedias del desarrollo) con esta JVM. Sin embargo, al cargar la aplicación completa se obtuvieron en casi un 100% de los casos errores fatales en la máquina virtual, en momentos aleatorios. Siendo errores y no excepciones, no logró detectarse los métodos específicos que causaban el conflicto. Los logs de los errores tampoco resultaron de utilidad.

Ya culminado el desarrollo, se consideró infactible rediseñar el sistema para ser utilizado en la máquina virtual estándar del BUG, phoneME. A esto se suma el hecho que la versión entrante de BUG (2.0) incluye por defecto el openJDK, y que la aplicación nunca mostró errores al ejecutar en el virtual BUG con esta JVM. En opinión de desarrolladores de Bug Labs Inc., la aplicación puede haber sido demasiada carga para un sistema versión 1.3. Esto pues el procesador de la misma opera con "over clock". La nueva versión, con mejor hardware, tiene mayor poder de procesamiento sin hacer uso de este método.

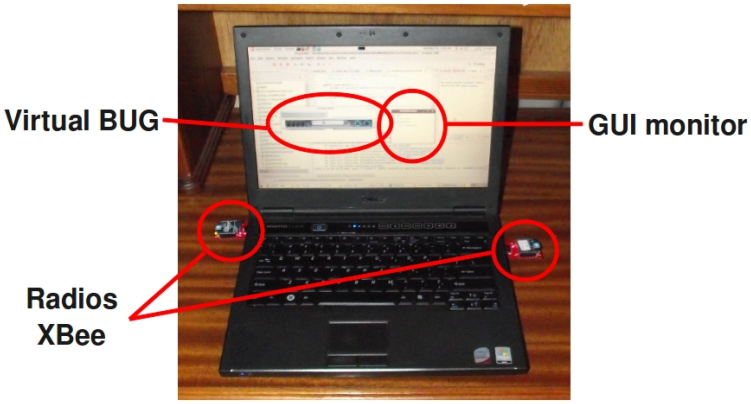
Se decidió de común acuerdo con el profesor guía el realizar las pruebas de validación desde el virtual BUG, ocupando las radios físicas conectadas a un mismo computador convencional. De esta manera, se llevó a cabo una prueba única (a modo de F.A.T.[13]) en que se apreciaran los eventos más relevantes para el sistema:

1. Conexión de un monitor a un sensor previamente activo.
2. Conexión de un segundo sensor a la red.
3. Desconexión y posterior re-conexión de un sensor de la red.
4. Desconexión y posterior re-conexión del monitor.

13 Factory Acceptance Test

Hasta donde el autor conoce, no existe forma de emular las muestras del modulo VonHippel en el virtual BUG, de manera que el código que realiza esta tarea fue reemplazado por una función random(). Debe hacerse notar que la aplicación pre-existente "vonHippelAnalogDemo" (ver Marco Conceptual) muestra exactamente como realizar una captura desde el ADC de este módulo. Esta aplicación ha sido utilizada con anterioridad por el autor, incluso en demostraciones en congresos, y por lo tanto el mismo considera que no es una falta el no utilizar este código en la prueba final de la aplicación presente. Se tiene plena confianza que, una vez superado el problema de error en la JVM (muy posiblemente, utilizando un BUG 2.0), la aplicación completa debiera comportarse como en la prueba realizada, obteniendo muestras reales.

El montaje físico de la prueba puede apreciarse en la Figura 5.1. Este consistió en la conexión de 3 radios XBee a un mismo computador, en el cual se ejecutaron 3 Virtual BUG distintos, cada uno con una aplicación propia (dos sensores y un monitor) y haciendo uso exclusivo de uno de los módems (por lo tanto, efectivamente existió comunicación inalámbrica). Las ventanas de un Virtual BUG y GUI de la aplicación se observan en la Figura 5.2.



(*) tercer XBee no se aprecia en esta toma

Figura 5.2: Virtual BUG + frame de la aplicación (*)

(*) el frame es mostrado en el LCD en el BUG real.

Los resultados observados se reflejan en los logs de la prueba. En lo que sigue se incluyen aquellos que demuestran las respuestas a los eventos indicados, con explicaciones intercaladas

1. Conexión de un monitor a un sensor previamente activo.

(1) [2010-11-18 03:13:52,243] [Thread-8] [INFO] [wsnmonitor.app.Node] Sensor, Short Address: null, IEEE

Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x42: is an End Device

Proceso de inicio del sensor, descubre su función en la red preguntando esto al módem XBee pre-configurado.

(2) [2010-11-18 03:13:52,339] [Thread-8] [INFO] [wsnmonitor.app.Node] Sensor, End Device, Short Address: 0x00,0x01, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x42: Successfully Associated

Aplicación pregunta al módem el estado de asociación a la red, recibiendo respuesta afirmativa.

(3) [2010-11-18 03:14:16,383] [Thread-8] [INFO] [wsnmonitor.app.Node] Monitor, Short Address: null, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x52: is the Coodinator

Análogo a (1), para monitor.

(4) [2010-11-18 03:14:16,502] [Thread-8] [INFO] [wsnmonitor.app.Node] Monitor, Coordinator, Short Address: 0x00,0x00, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x52: Successfully Associated

Análogo a (2), para monitor.

(5) [2010-11-18 03:14:17,153] [pool-1-thread-1] [INFO] [wsnmonitor.app.Monitor] Node discover response is: nodeAddress16=0x00,0x00, nodeAddress64=0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x52, rssi=0, nodeIdentifier=MCSupervisor

Se recibe una respuesta al comando ND, en particular del propio módem (está configurado así para incluirse a sí mismo en la lista de monitores, comportamiento opcional)

(6) [2010-11-18 03:14:17,155] [pool-1-thread-1] [INFO] [wsnmonitor.app.NodeImage] Created NodeImage Monitor, Coordinator, Short Address: 0x00,0x00, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x52

Se crea el objeto imagen correspondiente a la respuesta anterior.

(7) [2010-11-18 03:14:17,273] [pool-1-thread-1] [INFO] [wsnmonitor.app.Monitor] Node discover response is: nodeAddress16=0x00,0x01, nodeAddress64=0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x42, rssi=-47, nodeIdentifier=SELight

Se recibe una respuesta al comando ND, del sensor que se ejecutó desde el inicio de la prueba, previo al monitor.

(8) [2010-11-18 03:14:17,273] [pool-1-thread-1] [INFO] [wsnmonitor.app.NodeImage] Created NodeImage Sensor, End Device, Short Address: 0x00,0x01, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x42

Análogo a (6).

(9) [2010-11-18 03:14:20,046] [pool-1-thread-1] [INFO] [wsnmonitor.app.NodeImage] Sensor, End Device, Short Address: 0x00,0x01, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x42,Sample=0.75697964,11/18/10 3:14:19 AM

Se recibe una muestra del sensor, indicando correcto funcionamiento de la solicitud realizada por SampleTask() y de la respuesta del sensor.

(10) [2010-11-18 03:14:20,088] [pool-1-thread-1] [INFO] [wsnmonitor.app.NodeImage] Sensor, End Device, Short Address: 0x00,0x01, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x42,Sample=0.15640484,11/18/10 3:14:19 AM

En adelante, se reciben una serie de muestras.

2. Conexión de un segundo sensor a la red.

(1) [2010-11-18 03:14:48,616] [pool-1-thread-1] [INFO] [wsnmonitor.app.Monitor] Node discover response is: nodeAddress16=0x00,0x01, nodeAddress64=0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x42, rssi=-47, nodeIdentifier=SELight

(2) [2010-11-18 03:14:49,328] [pool-1-thread-1] [INFO] [wsnmonitor.app.Monitor] Node discover response is: nodeAddress16=0x00,0x02, nodeAddress64=0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x44, rssi=-52, nodeIdentifier=SETEMP

En algún momento posterior al inicio del segundo sensor (que inició después del monitor, insertándose la radio XBee y luego dando inicio al thread asociado), se reciben sus datos como una respuesta a ND.

(3) [2010-11-18 03:14:49,328] [pool-1-thread-1] [INFO] [wsnmonitor.app.NodeImage] Created NodeImage Sensor, End Device, Short Address: 0x00,0x02, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x44

Se crea el objeto imagen correspondiente.

(4) [2010-11-18 03:14:57,262] [Thread-8] [INFO] [wsnmonitor.app.Node] Sensor, Short Address: null, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x44: is an End Device

En algún momento posterior, se observa el procedimiento de inicio de la aplicación del segundo sensor. Conclusión: el módem responde automáticamente a ND antes de que la aplicación inicie la recolección de muestras. Sin embargo, esto no se percibe como un error (no presenta conflicto con el monitoreo).

3. Desconexión y posterior re-conexión de un sensor de la red

(1) [2010-11-18 03:15:38,175] [Timer-3] [INFO] [wsnmonitor.app.Monitor] removed node 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x44

El monitor remueve a uno de los sensores, cuyo thread fue terminado y su radio desconectada (para impedir respuestas automáticas a ND).

(2) [2010-11-18 03:21:03,597] [Thread-8] [INFO] [wsnmonitor.app.Node] Sensor, Short Address: null, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x44: is an End Device

(3) [2010-11-18 03:21:03,764] [Thread-8] [INFO] [wsnmonitor.app.Node] Sensor, End Device, Short Address: 0x00,0x02, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x44: Successfully Associated

(2) y (3): posteriormente la radio es reconectada y otro thread ejecutado. El nodo sensor inicia correctamente.

(4) [2010-11-18 03:21:28,548] [pool-1-thread-1] [INFO] [wsnmonitor.app.Monitor] Node discover response is: nodeAddress16=0x00,0x02, nodeAddress64=0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x44, rssi=-54, nodeIdentifier=SETemp

El nodo responde apropiadamente a ND.

(5) [2010-11-18 03:21:28,548] [pool-1-thread-1] [INFO] [wsnmonitor.app.NodeImage] Created NodeImage Sensor, End Device, Short Address: 0x00,0x02, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x44

Se crea un nuevo objeto imagen para ese sensor, y este es integrado a la lista.

4. Desconexión y posterior re-conexión del monitor.

(1) [2010-11-18 03:21:25,166] [Thread-8] [INFO] [wsnmonitor.app.Node] Monitor, Short Address: null, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x52: is the Coodinator

El coordinador inicia nuevamente, luego de haber sido detenido.

(2) [2010-11-18 03:21:25,263] [Thread-8] [INFO] [wsnmonitor.app.Node] Monitor, Coordinator, Short Address: 0x00,0x00, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x52: Successfully Associated

Inicia la red ("Successfully Associated" para un End Device significa estar correctamente asociado a una red, mientras que para un Coordinator significa haber "levantado" la red correctamente. Se relaciona con el registro "AI", Association Indication de los XBee)

(3) [2010-11-18 03:21:26,375] [pool-1-thread-1] [INFO] [wsnmonitor.app.Monitor] Node discover response is: nodeAddress16=0x00,0x00, nodeAddress64=0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x52, rssi=0, nodeIdentifier=MCSupervisor

Recibe la respuesta a ND de sí mismo.

(4) [2010-11-18 03:21:26,377] [pool-1-thread-1] [INFO] [wsnmonitor.app.NodeImage] Created NodeImage Monitor, Coordinator, Short Address: 0x00,0x00, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x52

Crea una imagen para sí mismo.

(5) [2010-11-18 03:21:26,656] [pool-1-thread-1] [INFO] [wsnmonitor.app.Monitor] Node discover response is: nodeAddress16=0x00,0x01, nodeAddress64=0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x42, rssi=-47, nodeIdentifier=SELight

(6) [2010-11-18 03:21:26,656] [pool-1-thread-1] [INFO] [wsnmonitor.app.NodeImage] Created NodeImage Sensor, End Device, Short Address: 0x00,0x01, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x42

(7) [2010-11-18 03:21:28,548] [pool-1-thread-1] [INFO] [wsnmonitor.app.Monitor] Node discover response is: nodeAddress16=0x00,0x02, nodeAddress64=0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x44, rssi=-54, nodeIdentifier=SETemp

(8) [2010-11-18 03:21:28,548] [pool-1-thread-1] [INFO] [wsnmonitor.app.NodeImage] Created NodeImage Sensor, End Device, Short Address: 0x00,0x02, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x44

(5), (6), (7) y (8): recibe respuesta a ND de los 2 sensores que continuaban conectados y ejecutándose, y crea imágenes para ellos (recordar que la lista anterior dejó de existir pues el thread Monitor fue terminado).

(9) [2010-11-18 03:21:29,280] [pool-1-thread-1] [INFO] [wsnmonitor.app.NodeImage] Sensor, End Device, Short Address: 0x00,0x01, IEEE Address: 0x00,0x13,0xa2,0x00,0x40,0x6e,0x02,0x42,Sample=0.8972883,11/18/10 3:21:29 AM

Comienza a recibir muestras normalmente.

La Figura 5.3 y Figura 5.4 muestran capturas de la interfaz gráfica en 2 momentos diferentes de la prueba.

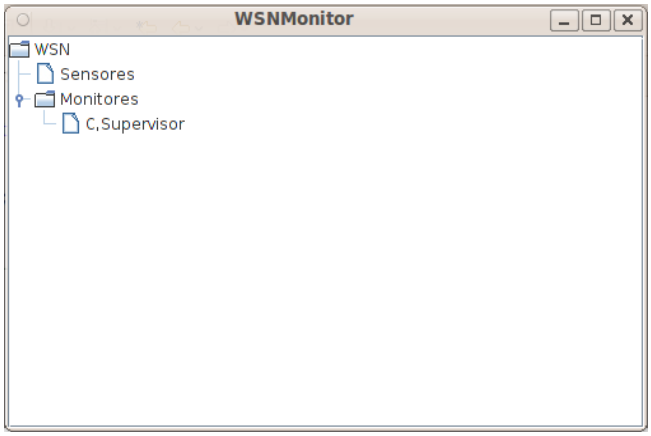


Figura 5.3: GUI iniciando, solo se observa al monitor de la red (sí mismo)

Figura 5.4: GUI con lista de todos los sensores en la red (*)

(*) que han respondido a ND recientemente.

En la Figura 5.4 se observa un cierto problema al desplegar las etiquetas de los nodos. Esto se trató de resolver mediante un refresco adecuado del *frame* de la aplicación, lo que resultó infructuoso. El tema fue investigado, no encontrándose una solución satisfactoria.

5.2 Discusión

En la prueba descrita anteriormente, se llevaron a cabo todos los hitos señalados, respondiendo el sistema adecuadamente a cada uno de ellos. Esto según lo confirman la bitácora del sistema e interfaz gráfica. De esta manera, puede establecerse el éxito de la prueba en general.

A continuación se realizan comentarios sobre características particulares observadas:

- Aunque el sistema no está concebido necesariamente para utilizar pocos recursos computacionales (salvo en el caso de los sensores), se observó que el computador utilizado pudo sostener sin problemas los procesos de 3 nodos (2 sensores y un monitor) sin mostrar ninguna lentitud en la respuesta. Esto sumado al hecho que cada sensor ejecutaba sobre su propia instancia del simulador virtual BUG. Tomando solo esto en cuenta, podría señalarse que el sistema no es intensivo en recursos de procesamiento. Sin embargo, esto entra en conflicto con la suposición del fallo provocado en la JVM openJDK descrito con anterioridad, lo que pone en duda esta última hipótesis. Dicho error podría tener relación entonces con alguna instrucción en específico que causa conflicto con el software de los BUG y no del Virtual BUG, y no estar relacionado con una carga de procesamiento excesiva.

- De manera similar, aunque el sistema no busca una optimización en el tráfico de datos, no se observó ningún cuello de botella. Sin embargo, esto debiese ser testeado con un número de nodos mucho mayor para establecer posibles problemas o no.

- El protocolo de comunicación utilizado (IEEE 802.15.4), permitió una correcta conectividad entre nodos (interfaz de aire) y hacia los BUGs (interfaz RS232 sobre USB). La librería de comunicaciones seleccionada para controlar las radios se comportó también de manera adecuada, recibiéndose y procesándose bien todos los mensajes requeridos para la prueba (no se encontraron en los logs paquetes con errores ni conflictos en el manejo de estos).

- La convención de mensajes adoptada resultó funcional, prueba de lo cual lo constituye el que los sensores efectivamente enviaban muestras (este proceso no es automático sino siempre en respuesta al requerimiento de un monitor), y el que éstas eran logeadas por el monitor.

- El sistema de logeo, y en particular los mensajes seleccionados para el nivel INFO (nivel en que se ejecuto la prueba, con nivel DEBUG en los paquetes relacionados con la librería xbee-api), proveyeron un adecuado panorama de los acontecimientos de la prueba.

- La interfaz gráfica demostró las funcionalidades de adicción y remoción de nodos, y

actualización de los datos de los sensores, en forma automática. Sin embargo, se observaron problemas con el correcto despliegue de las etiquetas de los nodos, y con la expansión de los mismos para observar la adición de nuevos nodos (el usuario debió cliquear en los nodos “sensores” y “monitores” para observar que efectivamente se habían agregado los nuevos nodos). La interfaz gráfica requiere, por lo tanto, de mayor trabajo.

A continuación, se evaluará únicamente el sistema elaborado, contrastándolo con los requerimientos iniciales de diseño (mientras que en las conclusiones se realizará un análisis con respecto de los objetivos del trabajo de título en su totalidad).

- Se desarrollaron Nodos Sensores con interfaz inalámbrica IEEE 802.15.4. Estos tienen la capacidad de recibir requerimientos de muestras. El código para la adquisición de muestras mediante el módulo BUGVonHippel se importó en su mayoría de una aplicación previa cuyo funcionamiento fue comprobado. Sin embargo, no pudo utilizarse en la prueba.

- Se desarrollaron Monitores Móviles en los BUG. Estos permiten obtener muestras desde los sensores. El valor más reciente es presentado al usuario mediante una interfaz gráfica basada en un árbol, donde se listan todos los nodos conocidos de la red. La información es además almacenada mediante un sistema de logeo a archivos.

- El protocolo utilizado para la comunicación permite el establecimiento de la red simplemente con la presencia de los BUGs y módems XBee, sin necesidad de ningún aparato adicional como routers o gateways.

- Todas las operaciones de inicio de la red, obtención, visualización y almacenamiento de datos son realizadas de manera automática por el sistema (previa configuración de las radios XBee). La interfaz gráfica presentada contiene muy pocos elementos, apuntando a la simpleza. La revisión de los datos históricos consiste simplemente en la lectura de archivos de texto plano.

Sin embargo, y como ya se indicó, fue imposible realizar la prueba adecuada en terreno. Por lo tanto no se puede asegurar la efectividad del sistema en su conjunto y en un caso real. Por otro lado, la prueba realizada puede considerarse como semi-real (aplicación ejecutándose en nodos simulados, pero utilizando una interfaz de comunicación a través del aire). En base a los resultados de la misma es que puede establecerse que, hasta el mayor conocimiento del autor, la solución desarrollada cumple con los requerimientos impuestos y es por lo tanto aceptable.

Capítulo 6 Conclusiones

6.1 Conclusiones

- Se diseñó e implementó de manera exitosa un sistema que permite la interconexión de dispositivos en una red inalámbrica local en base al protocolo IEEE 802.15.4, en ausencia de infraestructura. El sistema comprende las unidades Nodos Sensores y Monitores Móviles, los primeros encargados de recolectar muestras y los últimos de obtenerlas y administrarlas. Ambos dispositivos son creados a partir de una combinación de módulos, hardware externo y la base de la plataforma de arquitectura modular BUG. Los Nodos Sensores cuentan con convertidores de voltaje análogo-digitales, lo que permite la incorporación de una amplia gama de transductores externos. Los Monitores Móviles poseen pantallas que permiten al usuario visualizar los datos en tiempo real, a la vez que almacenan la información recolectada. Dada las dimensiones reducidas y autonomía energética de los equipos utilizados, además del uso de comunicaciones inalámbricas, ambos tipos de unidades resultan fácilmente transportables.

- La implementación y prueba del desarrollo en los dispositivos BUG resultó infructuosa. Esto puesto que una de los requerimientos del diseño comprendía la modificación de una propiedad de estos (concretamente, su máquina virtual de Java), y ocurrieron errores al realizar dicho cambio. Esto a pesar de que se habían realizado pruebas con versiones intermedias del sistema, que no habían presentado errores. Se desconoce la causa del fallo. De común acuerdo con el profesor guía, se optó por realizar la prueba final con los emuladores "virtual BUG": aplicaciones de software que se utilizaron durante el desarrollo y que ejecutan en computadores convencionales. Para tratar de acercarse a una prueba real, se utilizaron los módems externos que implementan el protocolo IEEE 802.15.4 (radios XBee) conectados al computador, de manera que aunque los BUGs eran virtuales, la comunicación entre los procesos efectivamente se daba a través de ondas de radio. Se establecieron a priori una serie de eventos a observar en la prueba, a la espera de la respuesta del sistema. Estos incluían desconexión y re-conexión de ambos tipos de nodos del sistema (monitores y sensores). Se obtuvieron resultados positivos frente a cada uno de los eventos, por lo que el sistema se considera validado hasta donde las mayores capacidades experimentales lo permitieron.

- Se realizó un extenso estudio teórico de mecanismos de conexión inalámbrica entre dispositivos electrónicos, incluyendo en específico WiFi, Bluetooth, ZigBee e IEEE 802.15.4. Los antecedentes levantados durante la investigación apuntaron al uso de uno de los dos últimos protocolos en el sistema. La falta de dispositivos y/o software apropiado impidieron el desarrollo de una investigación práctica con cualquiera de los dos hasta bastante avanzado el desarrollo. Una vez obtenidos los dispositivos, se seleccionó el protocolo IEEE 802.15.4 pues cumplía con los requerimientos planteados para el diseño y resultaba más simple de aplicar que ZigBee. Luego de este proceso, el autor siente haber ganado gran cantidad y calidad de conocimientos en el tema de comunicaciones inalámbricas en redes de sensores. Esto puede comprobarse a partir de la elaboración de un sistema que cumple con los requerimientos impuestos, hasta donde las pruebas lo

demuestran.

- Se llevó a cabo un estudio teórico-práctico de la arquitectura modular en base al trabajo con las plataformas BUG. Esta herramienta pudo ser aplicada efectivamente en la creación de dispositivos que cumplieran con los requerimientos impuestos al sistema, tomando la forma de dos dispositivos diferenciados: Nodos Sensores y Monitores Móviles. Ambos son capaces de unirse a una red inalámbrica mediante la adición de módems externos, que pueden concebirse como una nueva pieza modular. Se percibe una gran ventaja en la flexibilidad de la plataforma, pues durante el período de trabajo, se elaboraron una serie de sistemas electrónicos distintos (además del proyecto concreto de la memoria, como por ejemplo en demostraciones en seminarios). Además de este equipo en particular, existen otras plataformas de desarrollo modulares (ej. Arduino) que gozan de amplia popularidad en el mercado del desarrollo de dispositivos. De esta manera, es opinión del autor que el paradigma de arquitectura modular resulta ventajoso a la hora de desarrollar prototipos electrónicos en forma veloz.

- El desarrollo se llevó a cabo con permanente comunicación con la empresa fabricante de la plataforma BUG, Bug Labs Inc.. Esto permitió al alumno observar de cerca los procesos de evolución técnica y comercial de un producto innovador, y de la empresa *startup* responsable del mismo. El autor tuvo oportunidad de realizar una pasantía en dicha empresa, en New York, USA, de un mes de duración. Se puso especial atención a las dinámicas internas y recursos humanos y técnicos de la empresa. Puede destacarse la calidad del equipo de trabajo (gran mayoría con posgrados), ambiente distenso y de camaradería, trato equitativo entre todos los miembros (ausencia de formas de relacionarse verticales, propias de una jerarquía rígida). Resulta interesante que el diseño de hardware de la plataforma, que podría suponerse es el *core business* de la empresa, es externalizado a una oficina extranjera. De esta manera, el verdadero negocio es agregar valor al producto, en la forma de soporte, una comunidad en línea donde se comparten los desarrollos y una oferta de servicios especializados para empresas.

- Un aspecto importante de observar para una empresa en desarrollo es la potencial falta de madurez de sus productos. Esto se apreció en a lo menos dos ocasiones claras durante el desarrollo. Primero: al descubrirse que el módulo ZigBee de BUG no contaba con las suficientes funciones de software para comunicarse efectivamente con múltiples nodos. Segundo: al fallar la máquina virtual requerida para utilizar apropiadamente el sistema, habiendo sido esta modificación incluso sugerida por ingenieros de Bug Labs Inc.. Ambos hechos forzaron al autor a tomar decisiones sobre la marcha, tal de salvaguardar las posibilidades de éxito del sistema final. Sin duda, cualquier producto innovador está sujeto a imperfecciones, y la capacidad de reaccionar rápidamente para solucionar las mismas es clave para alcanzar los objetivos de cualquier proyecto.

- El principal aporte realizado a la empresa fue a través de la inmersión en los temas del desarrollo, que no eran dominados por ésta previamente, así como el desempeño en paralelo a la memoria en un proyecto para Codelco, que pudiera evolucionar en una solución tecnológica de alto valor para la misma. Asimismo, se hizo adición de una aplicación a la comunidad en línea que incentiva el desarrollo de aplicaciones con

comunicaciones.

- El proyecto desarrollado en paralelo a esta memoria, para la corporación Codelco, permitió obtener gran experiencia al autor, especialmente en cuanto a:

  - Desempeñarse en labores distintas (aunque relacionadas) en paralelo.

  - Las características especiales de un proyecto de innovación: por un lado, existen oportunidades de desarrollo de nuevas capacidades en base estos. En este sentido, fue la participación en este proyecto lo que permitió la pasantía en Bug Labs Inc., pues era necesario obtener conocimiento especializado en la plataforma. Por otro lado, el manejo de la incertidumbre y la correspondiente evaluación de riesgo es un tema que debe ser considerado.

- En resumen, las experiencias ya sea directamente relacionadas con al trabajo de título, como:

  - Investigación teórica; Diseño en base a requerimientos; Implementación práctica de un sistema; Validación de la solución propuesta; Documentación (redacción del texto presente).

  Así como aquellas no directamente relacionadas, pero cuya ocurrencia fue posible gracias al desarrollo de este tema en específico, como:

  - Estadía en empresa extranjera (Bug Labs Inc.); Participación en proyecto de Innovación en Codelco.

  Se consideran de un tremendo valor formativo tanto en lo académico, profesional y personal.

6.2 Trabajo futuro

El autor visualiza las siguientes tareas como relevantes para el mejoramiento del sistema y, a la vez, interesantes desde una perspectiva académica:

- Realización de pruebas exhaustivas y en terreno. En particular, aquellas que permitan establecer no solo el correcto comportamiento de la aplicación ante los eventos simulados en este trabajo, sino el desempeño del conjunto en métricas como ancho de banda, numero de nodos máximo, robustez ante interferencias. Es muy posible que esto incluya el traslado del sistema a la versión 2.0 de BUGs.

- Aplicación del sistema actual en un caso de uso concreto. De hecho, esto muy probablemente será implementado por el autor como continuación de su labor con Codelco.

- Aumento de funcionalidades del sistema, como por ejemplo agregando la actuación sobre los nodos y no solo monitoreo de los mismos. Asimismo, evaluar el uso de diversos dispositivos como nodos sensores, incluyendo algunos cuyo hardware sea específico para redes de sensores inalámbricas.

- Utilizar un protocolo de comunicación que permita la elaboración de redes enmalladas (como ZigBee) y evaluar las posibles ventajas de este esquema.

Capítulo 7 Referencias

A continuación se listan las referencias bibliográficas más relevantes para el trabajo realizado. El formato seleccionado para las mismas es Modern Language Association (MLA).

[1] Weiser, Mark. "Ubiquitous Computing". Xerox PARC. 25 Nov 2009 <http://sandbox.parc.com/ubicomp/>.

[2] Wikipedia contributors, "Ubiquitous computing". Wikipedia, the free encyclopedia. 25 Nov 2009 <http://en.wikipedia.org/wiki/Ubiquitous_computing>.

[3] Wikipedia contributors, "Ambient intelligence". Wikipedia, the free encyclopedia. 25 Nov 2009 <http://en.wikipedia.org/wiki/Ambient_intelligence>.

[4] K., Gabriel. "Engineering Microscopic Machines". Scientific American Magazine September 1995: 118-121.

[5] "Nabaztag". violet. 25 Nov 2009 <http://www.nabaztag.com/en/index.html>.

[6] Römer, K. and F. Mattern. "The design space of wireless sensor networks." IEEE Wireless Communications 11 (December 2004): 54-61.

[7] D. E. Comer, Internetworking with TCP/IP Vol.1: Principles, Protocols, and Architecture, 4th ed.

[8] "BugLabs". BugLabs. 25 Nov 2009 <http://www.buglabs.net/>.

[9] Bandyopathyay, L.K., S.K. Chaulya, and P.K. Mishra. Wireless Communication in Underground Mines. Springer, 2009.

[10] Odom, Wendell. CCENT/CCNA ICND1. Cisco Press, 2008.

[11] Understanding 802.15.4TM and ZigBeeTM Networking. DainTree Networks, 2005.

[12] "IEEE 802.15 WPAN Task Group 1 (TG1)". IEEE. 23 Jun 2010 <http://www.ieee802.org/15/pub/TG1.html>.

[13] Getting Started with ZigBee and IEEE 802.15.4. DainTree Networks, 2008.

[14] Mitchell, Bradley. "How Many Computers Can Share a WiFi Network?". About.com. 24 Jun 2010 <http://compnetworking.about.com/od/wirelessfaqs/f/howmanydevices.htm>.

[15] "FAQ". ZigBee Alliance. 24 Jun 2010 <http://www.zigbee.org/About/FAQ.aspx>.

[16] Demystifying 802.15.4 and ZigBee®, White Paper. Digi, 2007-2008.

[17] Schultz, K.. "What is OSGi?". BugLabs. 28 Jun 2010 <http://community.buglabs.net/kschultz/posts/107-What-is-OSGi->.

[18] Tarkoma, Sasu. Mobile Middleware. Finlandia: Wiley, 2009.

[19] "The Rule of Least Power". W3C. 24 Jun 2010 <http://www.w3.org/2001/tag/doc/leastPower>.

[20] Rellermeyer, Jan S. and Gustavo Alonso. "Concierge: a service platform for resource-constrained devices." SIGOPS Oper. Syst. Rev. 41 (2007): 245-258.

[21] Nickul, Dave. Service Oriented Architecture (SOA) and Specialized Messaging Patterns. Adobe, 2007.

[22] TIOBE Software. 1 Sep 2010 <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>.

[23] "About the Java Technology". Oracle. 1 Sep 2010 <http://download.oracle.com/javase/tutorial/getStarted/intro/definition.html>.

[24] Rapp, Andrew. "xbee-api - Project Hosting on Google Code." xbee-api. Google code, n.d. Web. 20 Nov 2010. <http://code.google.com/p/xbee-api/>.

[25] "jXBee Java XBee 802.15.4/ZigBee Modem Driver." Illinois Security Lab. Illinois Security Lab, n.d. Web. 20 Nov 2010. <http://seclab.cs.uiuc.edu/web/component/content/article/73-jxbee-java-xbee-802154zigbee-modem-driver-.html>.

[26] "XBee Explorer Dongle - SparkFun Electronics."Sparkfun. Sparkfun, n.d. Web. 20 Nov 2010. <http://www.sparkfun.com/products/9819>.

[27] "Apache log4j 1.2 - log4j 1.2." http://logging.apache.org/. The Apache Software Foundation, n.d. Web. 21 Nov 2010. <http://logging.apache.org/log4j/1.2/>.

[28] Jarvi, Trent. "RXTX : serial and parallel I/O libraries supporting Sun's CommAPI." RXTX. N.p., n.d. Web. 21 Nov 2010. <http://www.rxtx.org/>.

[29] Connolly, John. "log4jlib_osgi." BugLabs. BugLabs, 27 Oct 2010. Web. 21 Nov 2010. <http://www.buglabs.net/applications/log4jlib_osgi>.

[30] Ballantine, Brian, and John Connolly. "com.buglabs.emulator.rxtx." BugLabs. BugLabs, 31 Dec 2009. Web. 21 Nov 2010. <http://www.buglabs.net/applications/com.buglabs.emulator.rxtx>.

[31] Contreras, Mauricio. "xbee_api_0_9_osgi." BugLabs. BugLabs, 21 Nov 2010. Web. 22 Nov 2010. <http://www.buglabs.net/applications/xbee_api_0_9_osgi>.

[32] XBee®/XBee-PRO® OEM RF Modules Product Manual. Digi International Inc., 2008. Print.

[33] "Hayes command set." Wikipedia. Wikipedia, n.d. Web. 23 Nov 2010. <http://en.wikipedia.org/wiki/Hayes_command_set>.

[34] "X-CTU (XCTU) software." Digi. Digi International Inc., n.d. Web. 24 Nov 2010. <http://www.digi.com/support/kbase/kbaseresultdetl.jsp?kb=125>.

[35] "Arduino." Arduino, n.d. Web. 24 Nov 2010. <http://www.arduino.cc/>.