Universidad de Buenos Aires
Facultad de Ingeniería

66.99 Trabajo Profesional - Ingenieria Electrónica

Red de Sensores ZigBee

Docentes:

Ing. Miguel Reiser

Ing. Mónica Nitoli

Alumnos:

Hector Alberto Aquino Filho, *Padrón*: 88064
haquino@fi.uba.ar
Juan Pablo D'Ambra, *Padrón*: 88170
juanpablodambra@hotmail.com
Matías Sebastián Stahl, *Padrón*: 87142
stahlmatias@gmail.com

Marzo 2012

Agradecimientos

Habiendo llegado al final de un período de trabajo de varios meses, cuyo fruto es el ilustrado en el presente informe, sentimos que debemos agradecer a varias personas por su incansable y desinteresada ayuda, la cual nos facilitó de sobre manera nuestro trabajo y nos allanó un camino que de otra forma hubiera sido más arduo y tedioso, volviéndolo placentero.

Primeramente, extendemos nuestro agradecimiento a la gente del Club de Robótica, especialmente a Lucas Chiesa y Ariel Burman por el préstamo de los módulos ZigBee y la orientación inicial para comenzar a programarlos. A todos los integrantes del Laboratorio Abierto (LABI), quienes siempre que necesitamos cualquier tipo de material, herramientas o usar las instalaciones del mismo para trabajar, se mostraron incansablemente disponibles. A los empleados del Pañol del Departamento de Electrónica por brindarnos los instrumentos necesarios en la etapa de desarrollo. A Nicolás Matsunaga por ayudarnos a encontrar errores y sugerir posibles mejoras en la interfaz de control por computadora. A los docentes de la materia Trabajo Profesional de Ingeniería Electrónica, en especial a Mónica Nitoli por sus correcciones y comentarios. Sin estos no hubiéramos alcanzado jamás el nivel de calidad requerido.

Finalmente y no menos importarte, queremos agradecer a nuestras familias, amigos y novias por el apoyo y el cariño que nos brindan a diario y sin el cual todo en la vida sería mucho más difícil.

A todos, gracias.

Héctor Aquino Filho, Juan Pablo D'Ambra, Matias Stahl

Índice

Resumen

En este informe veremos el desarrollo para la implementación de una red de sensores con el fin práctico de monitorear distintos parámetros físicos como temperatura, humedad, intensidad lumínica y distintos tipos de gases dentro de un ambiente controlado y poco hostil como un data center.

Discutiremos los requerimientos de los potenciales clientes y como se especifican las funcionalidades y el diseño del producto, desde las especificaciones de hardware hasta el software. Como herramienta clave en esta parte emplearemos la casa de calidad.

Luego analizaremos la factibilidad tecnológica estudiando el FMEA de cada módulo, la factibilidad de tiempos utilizando como herramientas el PERT, simulaciones de Montecarlo y la programación con Gantt. A su vez haremos un exhaustivo estudio económico, estudiando las características de los productos de la competencia, los costos que insume desarrollar el producto que se llevó a cabo y su ciclo de vida.

Concluido el estudio de factibilidad entramos en los detalles propios de la implementación del prototipo, dando en principio un diagrama en bloques general y luego dando los detalles de los mismos junto con los planes de prueba y verificación. Al mismo tiempo, esta misma tarea se hará para el software desarrollado, tanto para el firmware que ejecutará el microcontrolador de nuestro dispositivo como lo referente al software que se instala en la PC del lado del cliente.

Una vez que concluye el análisis de los diagramas en bloque, pasaremos a la construcción del prototipo, definiremos los módulos empleados y sus componentes. Se incluirán los circuitos esquemáticos y los diagramas de PCB. A su vez se dará detalles especiales sobre el montaje de las placas.

Finalmente explicaremos los pasos para la validación del hardware y el software desarrollado como así también explicaremos el estudio de confiabilidad del hardware con el cálculo medio entre fallas, la disponibilidad y mantenibilidad. En este punto definiremos un plazo adecuado para establecer el período de garantía y como influye en el costo del producto.

El material de referencia como las hojas de datos, notas de aplicación, esquemas, planos, código de software y los correspondientes makefiles están disponibles en el CD que acompaña este documento.

1. Introducción

Una red de sensores es una red de dispositivos distribuidos que usan sensores para monitorear condiciones en diferentes localizaciones. Estos dispositivos deben de ser pequeños y baratos de forma que puedan ser producidos y utilizados a gran escala. Estas características hacen que tanto su consumo de energía, memoria, velocidad y capacidad de comunicación estén limitadas.

Cada dispositivo está equipado con uno o más sensores, un transmisor, un pequeño microcontrolador y una fuente de energía que normalmente es una batería. Los dispositivos se comunican unos con otros usando una arquitectura ad-hoc (sin infraestructura predeterminada) y de forma inalámbrica. El flujo de información acaba en unos nodos especiales llamados coordinadores que disponen de unas capacidades superiores a los nodos sensores y que permiten enviar la información a otras redes para su procesamiento posterior.

A grandes rasgos, las redes de sensores engloban tres áreas de tecnología o conocimiento: comunicaciones, sensórica y computación (entendiendo por ello hardware, software y algoritmos), todo ello ligado a estrategias de gestión eficiente de la energía consumida (en cada una de las áreas) y a tecnologías de micro generación de energía y de almacenamiento de la misma.

Una de sus características principales es la capacidad de comunicarse sin hilos y su potencial de auto organizarse y auto-configurarse. Generalmente utilizan una topología de red que asemeja una malla, un sensor puede descubrir a sus vecinos y el software de ruteado abre múltiples conexiones de forma que siempre se puede encontrar un camino al destino final. Cada nodo puede actuar como un repetidor o como un router, o bien esas funciones se asignan a nodos específicos dentro de la red. Principalmente se basan en el estándar IEEE 802.15.4 (para los niveles físicos y de acceso al medio) y en el estándar ZigBee aunque también se utilizan otras tecnologías.

El estándar IEEE 802.15.4 define el nivel físico y el control de acceso al medio de redes inalámbricas con tasa bajas de transmisión de datos. El propósito del estándar es definir los niveles de red básicos para dar servicio a un tipo específico de red inalámbrica de área personal (WPAN) centrada en la habilitación de comunicación entre dispositivos ubicuos con bajo coste y velocidad. Entre los aspectos más importantes se encuentra la adecuación de su uso para tiempo real por medio de slots de tiempo garantizados, evitación de colisiones por CSMA/CA y soporte integrado a las comunicaciones seguras. También se incluyen funciones de control del consumo de energía como calidad del enlace y detección de energía.

1.1. Historia y antecedentes

Las redes ZigBee comenzaron a desarrollarse hace mas de una década, en 1998 cuando se hizo claro que las tecnologías como Wi-Fi o Bluetooth no serían adecuadas para muchas aplicaciones. En particular, para satisfacer la necesidad de crear una red digital inalámbrica auto-organizable ad-hoc.

El estándar IEEE 802.15.4-2003 fue publicado en mayo de 2003 y se continuo con la publicación del IEEE-802.15.4-2006. A fines del 2003, Philips Semiconductors, un gran partidario de las redes malladas hasta ese momento, ceso su inversión. Sin embargo, Philip Lighting ha continuado con la participación de Philips siendo un miembro activo y promotor en la mesa de directiva de la ZigBee Alliance.

ZigBee Alliance anuncio en octubre de 2004 la duplicación de los miembros con respecto al año anterior, y que se contaba con más de cien compañías miembro en más de 22 países. Para el 2005 se había crecido a 150 compañías miembro y para fines de ese mismo año se había superado las 200 compañías.

Las especificaciones de ZigBee fueron ratificadas el 14 de diciembre de 2004 y la ZigBee Alliance anuncio la disposición de la Especificación 1.0 el 13 de junio del 2005, conocida como *ZigBee 2004 Specification*. En septiembre de 2006 se publicó la *ZigBee 2006 Specification*. En 2007, ZigBee PRO, una versión mejorada de la especificación fue finalizada.

El primer lanzamiento del *stack* de ZigBee se realizó en 2004. El segundo lanzamiento fue en 2006 y reemplaza principalmente la estructura MGS/KVP con un “cluster de bibliotecas”. El stack actual es el ZigBee 2007 stack, el cual contiene dos perfiles de stack, el primero dedicado para hogar

e iluminación de uso comercial y el segundo, llamado ZigBee Pro que ofrece algunas características adicionales como multi-casting, ruteo "many-to-one" y mejoras con respecto a la seguridad, al ofrecer intercambio de clave simétrica (SKKE) mientras que las versiones anteriores ofrecían un footprint en ram y flash. Ambos perfiles ofrecen completa capacidad de trabajo en red mallada y son compatibles con todas las aplicaciones ZigBee.

Glosario

Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz. 4, 15

CSMA/CA Carrier Sense with Multi Access / Collision Avoidance, (Acceso Múltiple por Detección de Portadora con Evasión de Colisiones) es un protocolo de control de redes de bajo nivel que permite que múltiples estaciones utilicen un mismo medio de transmisión. 4

IEEE 802.15.4 Es un estándar que define el nivel físico y el control de acceso al medio de redes inalámbricas de área personal con tasas bajas de transmisión de datos. 4

MCU Multipoint Control Unit, Unidad de Control Multipunto. 16

RF Transmisión que emplea el espectro de radiofrecuencia cuyo rango es de 3 kHz a unos 300 GHz. 66

SNMP Simple Network Management Protocol. 6

TCP/IP Transmission Control Protocol, Internet Protocol. 10

USB Universal Serial Bus. 68

UWB Ultra Wide Band, se refiere a cualquier radio o dispositivo wireless que ocupe la banda mayor que el 25 % de la frecuencia central o sea mayor a 1.5 GHz. 15

Wi-Fi Describe una red o terminal que utiliza ondas electromagnéticas en lugar de conductores con cable. Incluye RF, infrarrojas. 4, 10, 15

WPAN Wireless Personal Area Networks, Red de Área Personal Inalámbrica. 4

ZigBee Es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radiodifusión digital de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal. 4, 10, 15, 16, 40

2. Objetivos

2.1. Justificación del proyecto

El sector de empresas dedicadas o que invierten en Tecnología de la Información y Comunicaciones (TIC) en la Argentina está presentando un gran crecimiento en la actualidad. La mayoría de estas empresas cuenta con un data center propio para almacenar en forma efectiva su información sensible en espacios a los que pueda acceder.

Uno de los principales problemas que trae aparejado la administración de este tipo de espacios es el costo energético que representa y lo importante que es conocer el estado de las variables físicas como temperatura, humedad, emisión de gases, etc.

En los edificios modernos que cuentan con tecnología de punta las condiciones ambientales se controlan mediante un sistema automático en el edificio, BAS, por sus siglas en inglés *Building Automation System*. Este tipo de sistemas puede proveer información visual con gráficas de temperatura representadas en cualquier computadora personal. Sin embargo, en la mayoría de los data centers, las condiciones ambientales se manejan con múltiples equipos independientes distribuidos en la zona, y no cuentan con equipo adecuado para monitorear, controlar y representar visual y gráficamente las condiciones de temperatura o humedad eficientemente.

En una conversación con el Administrador del área de Telecomunicaciones del Banco Supervielle, Rafael Franco, nos decía lo importante y crítico que es mantener la temperatura y refrigeración del data center ya que se controlan todos los enlaces entre sucursales y albergan los equipos dedicados a las operaciones interbancarias, con lo cual la disponibilidad es crítica. *"Es un área donde la cantidad de equipos es inmensa y la temperatura que levantan es infernal"* (sic). Por eso para asegurar la disponibilidad es importante mantener la temperatura y refrigeración adecuada.

Actualmente, nos comentaba, que tienen dos equipos de aire acondicionado Stulz GmbH Klimatechnik WIB 8000 y se consultan por SNMP el estado de la temperatura, humedad y presión atmosférica. No tienen forma de medir por zona o lo que se conoce como "hotspots".

En otra entrevista llevada a cabo con el Administrador de la Red de Telecomunicaciones del Banco Santander a cargo de Logicalis Juan Martín Lobo, nos comentaba que el año pasado (Junio de 2011) invirtieron fuertemente en equipos de refrigeración para el data center, porque tuvieron una falla térmica en un aire acondicionado y en varios servidores la temperatura empezó a aumentar notablemente y la única forma de disminuirla fue apagando todos los equipos con la excepción de aquellos que se encargaban de las transacciones de los clientes más importantes. Actualmente monitorean la temperatura por un sensor externo que se consulta por SNMP. Nos comentaba además que el consumo energético que representó la incorporación de estos equipos subió casi el 70 % respecto a los consumos anteriores.

Por último entrevistamos al encargado de Gestión de Procesos y Operaciones DC Gerencia Regional de Tecnología y Operaciones de Cencosud, Marcelo Moscoloni. Cencosud controla lo que es Jumbo, Disco, Plaza Vea, Easy, Unicenter Shopping sólo por nombrar las empresas más importantes. El data center de Buenos Aires ubicado en la zona de Martinez controla todos los enlaces y las operaciones de cajas de los locales. Tiene una dimensión aproximada de 2000 m^2 y lo refrigeran con cuatro equipos de aire acondicionado, *"pero que esto no es un problema, se pueden contemplar gastos de más en esta área, pero si pudiésemos ahorrar o efectivizar el consumo seria fantástico"* (sic). Además la empresa cuenta con alrededor de 20 data center sólo en Buenos Aires, siendo los más importantes los ubicados en Olleros y en Acoyte, luego en zona sur, Moreno, Pilar, Canning, etc. *"Lo que buscamos siempre es contar con energía asegurada y un sistema de refrigeración de precisión de alta disponibilidad"*.

Nos decía que no tienen ningún software de monitoreo remoto y no llevan una gráfica del estado histórico de las condiciones ambientales del mismo. Están buscando un software de monitoreo pero que en una empresa de este tamaño este tipo de soluciones lleva mucho tiempo y hay que atravesar varios departamentos para que se apruebe la idea.

Este tipo de necesidades presenta una solución muy clara. Utilizar sensores en los lugares necesarios y administrarlos ya sea con una computadora en la zona o desde una central externa. Lo que requiere este tipo de solución con muchos sistemas interactuando entre sí, ya que no existe

actualmente en la Argentina, un sistema que provea todas estas soluciones en un mismo "paquete". Esto genera que las soluciones actualmente disponibles sean económicamente caras y/o difíciles de administrar y utilizar, generando trabajo de más, lo que generalmente deriva en un sistema poco fiable, caro y muy poco escalable.

La solución de este tipo de problemas, existe en el exterior, y se la conoce como una **red de sensores**. La misma se basa en un servidor central y muchos dispositivos, llamados *"nodos"* en las zonas en las que se desean medir. Este sistema permite que los nodos sean muy simples, lo que significa que son muy baratos económicamente y que sea muy fácil de escalar la solución, con lo que el sistema crece a medida que crece la empresa, convirtiéndose en un gran aliado.

Se muestra en un estudio llevado a cabo por **SynapSense Corporation**, *Wireless Sensors Improve Data Center Energy Efficiency* [12] y también en el reporte *Project Genome: Wireless Sensor Network for Data Center Cooling* [15] donde explican claramente que la densidad de energía medida en potencia por metro cuadrado se incrementa, la energía ahorrada por refrigeración puede realizarse aplicando una red de sensores y utilizando la información recolectada para manejar eficientemente el data center. Además, puede ser muy útil para detectar hotspots, realizar benchmarking, capacity planning y hacer un estudio forense de las condiciones ambientales incluyendo alertas y alarmas. En el paper escrito por Sasidharan, Pianegiani y Macii se realiza una comparación de los protocolos utilizados en las redes de sensores inalámbricas modulares donde los requisitos en general son baja latencia, baja pérdida de paquetes, y bajo consumo, llegando a la conclusión que el protocolo IEEE 802.15.4 es la mejor opción [17].

Los beneficios que presenta el empleo de esta tecnología es la siguiente:

- Visualizar la performance de la refrigeración.
- Los requerimientos de humidificación.
- Identificar hotspots.
- Predicción preventiva de mantención.
- Cálculo del uso efectivo de la potencia en tiempo real.

Cuando le comentamos a los entrevistados nuestra solución les resultó muy interesante y les pareció muy viable, siempre y cuando contando con ciertos requerimientos que veremos en la casa de calidad.

También vale la pena destacar algunos comentarios de una conferencia llevada a cabo en el Sheraton Hotel de Buenos Aires por Data Center Dynamics el 15 de abril de 2011. Uno de los expositores, Maximiliano Schlez de Panduit, abordó los desafíos a los que se enfrentan los propietarios y operadores de data centers en cuanto a las estructuras con necesidad de mejora, y de la nueva arquitectura necesaria para cumplir la compleja combinación de un crecimiento exponencial de los datos, avances tecnológicos, *la eficiencia energética*, resistencia y seguridad.

2.2. Finalidad del proyecto

Algunas topologías de redes de sensores pueden generar representaciones gráficas de perfiles de temperatura dentro del data center conocidas como *heat maps*, o mapas de calor. Si se integran a la zona de cada unidad de refrigeración, el sistema se puede optimizar para acomodar los servidores que presentan alta carga en racks ubicados específicamente.

La finalidad del proyecto es construir una red de sensores con la topología adecuada para medir distintos parámetros físicos dentro de un ambiente controlado y poco hostil como es un data center. La finalidad de estas mediciones y monitoreo es que brinden una solución en el ahorro de la energía y sirvan como herramienta para decidir en la inversión de equipos de refrigeración adecuados.

El proyecto cuenta con las siguientes ventajas dentro de su solución. Se contempla el kit básico de un nodo coordinador o router y dos nodos sensores.

- Ventajas Generales:

  Inalámbrico.

  No invasivo.

  Permite el monitoreo de un área de 1500 m^2.

  Funcionamiento automático.

  Permite personalizar la medición, cambiando el tipo de sensor o agregando uno nuevo.

Y como única desventaja podemos mencionar:

- Desventajas Generales:

  Dependiendo de la complejidad necesaria en cada nodo, si la red crece mucho, las comunicaciones pueden volverse lentas, teniendo un límite de hasta 15 nodos sensores con un único coordinador o router.

2.3. Planteamiento del problema a resolver

En base a lo estudiado sobre el tema, vimos que existe una clara necesidad por parte de muchas empresas, que no siempre es llenada completamente, por lo que se abren nuevas posibilidades en el mercado. Esta necesidad, cuando no se resuelve adecuadamente puede traer consecuencias económicas muy fuertes y dejar sin servicio a muchos usuarios por citar el caso de la alta disponibilidad que requiere un Banco.

La idea de la redes de sensores es bastante utilizada en el exterior y países como Estados Unidos, Australia, Canadá, China y Corea del Sur pretenden invertir cerca de 1000 millones en esta área de investigación en los próximos 5 años. Actualmente se utilizan en la construcción de "Edificios Verdes" o para sistemas de automatización de hogares según un reporte de la consultora **OnWorld**. Estamos ante una "tecnología" bastante reciente donde no hay empresas con personal calificado y experto.

Según este estudio, el mercado de redes de sensores sin cable experimenta un alto crecimiento en la demanda, porque cada vez más empresas quieren instalar este tipo de red.

En la Argentina la situación es muy diferente ya que no existe nadie que la produzca internamente e importarla puede ser un costo que no todas las empresas les puede resultar factible.

Actualmente hay una pequeña representación de CrossBow a través de la empresa Open Automation. Este tema de mercado se estudiará con más detalle y profundidad en la sección 4.3.

3. Definición del producto

3.1. Requerimientos

La solución propuesta como vimos se basa en una **red de sensores**. En la misma cada nodo final de la red podrá:

- Realizar una o más mediciones de la zona.
- Comunicarse con el nodo central de la zona.

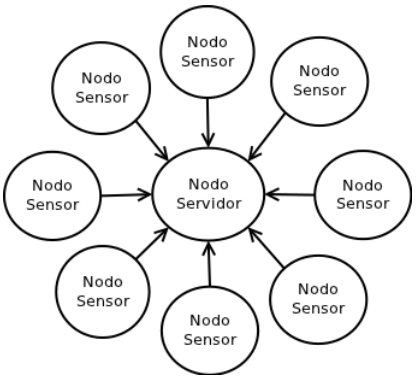

Figura 1: *Diagrama Básico de la Red*

Esta topología permite que cada nodo de la red sea muy simple, con lo cual su precio se reduce bastante y en el eventual caso que se dañe alguno, su reemplazo es simplemente descartar el anterior y colocar el nuevo, ya que la red se actualiza de forma dinámica.

En el caso de querer obtener información de la red se puede hacer desde una computadora. Esto permite un monitoreo a distancia de la red de forma muy simple y rápida.

Además, cada sensor brindará la oportunidad de realizar una o más mediciones, las cuales podrán ser temperatura, humedad, gas, humo, movimiento y luz.

Se permite personalizar la red según la necesidad y el presupuesto del cliente. Esta personalización, no es total, ya que el límite de sensores por nodo es limitado a cinco, debido a que se busca que cada nodo gaste la menor cantidad de energía posible, y una gran cantidad de sensores, representa una gran cantidad de energía utilizada.

Al estar pensado para un data center donde hay muchos equipos y el cableado en esos lugares es dificultoso, debe ser inalámbrica para ser lo menos invasivo posible.

Las comunicaciones se realizarán mediante ZigBee, la cual es una tecnología pensada para redes inalámbricas, y representa una alternativa relativamente económica al tener en cuenta todas las características del protocolo y de su stack que brindan numerosas ventajas, facilitando su empleo y las comunicaciones entre módulos. No es necesario desarrollar la capa de TCP/IP y nos proporciona de forma integrada un mecanismo de control de errores de acceso del medio.

No va a interferir con la red ya construida dentro del lugar que estará hecha mediante ethernet o Wi-Fi.

Además al estar desarrollado mediante ZigBee, la tarea de administración por parte del Administrador del Sistema se ve facilitada ya que no es necesario configurarle una dirección IP por ende no es necesario contemplarlo en las políticas de seguridad informáticas de la empresa.

Otra ventaja del protocolo 802.15.4 es la capacidad del routeo multisalto entre nodos, retransmitiendo datos a más de un nodo para mejorar la comunicación con el gateway o coordinador.

Un diagrama simple de cada nodo, se puede ver en la figura 2:

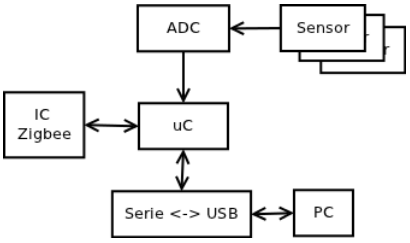

Figura 2: *Diagrama de un Nodo de la Red*

El sensor inalámbrico remoto se alimenta con una batería para enviar datos al coordinador. La vida de esta batería es un factor importante a tener en cuenta y es afectada principalmente por el consumo del módulo que se elija y la tecnología que este emplea para establecer la comunicación. Otro factor que afectan la batería es la encriptación, pero no lo vamos a tener en cuenta en nuestro caso ya que estaremos trabajando en un ambiente controlado y poco hostil.

3.1.1. Casa de calidad

En la figura 3 se puede ver la casa de calidad.

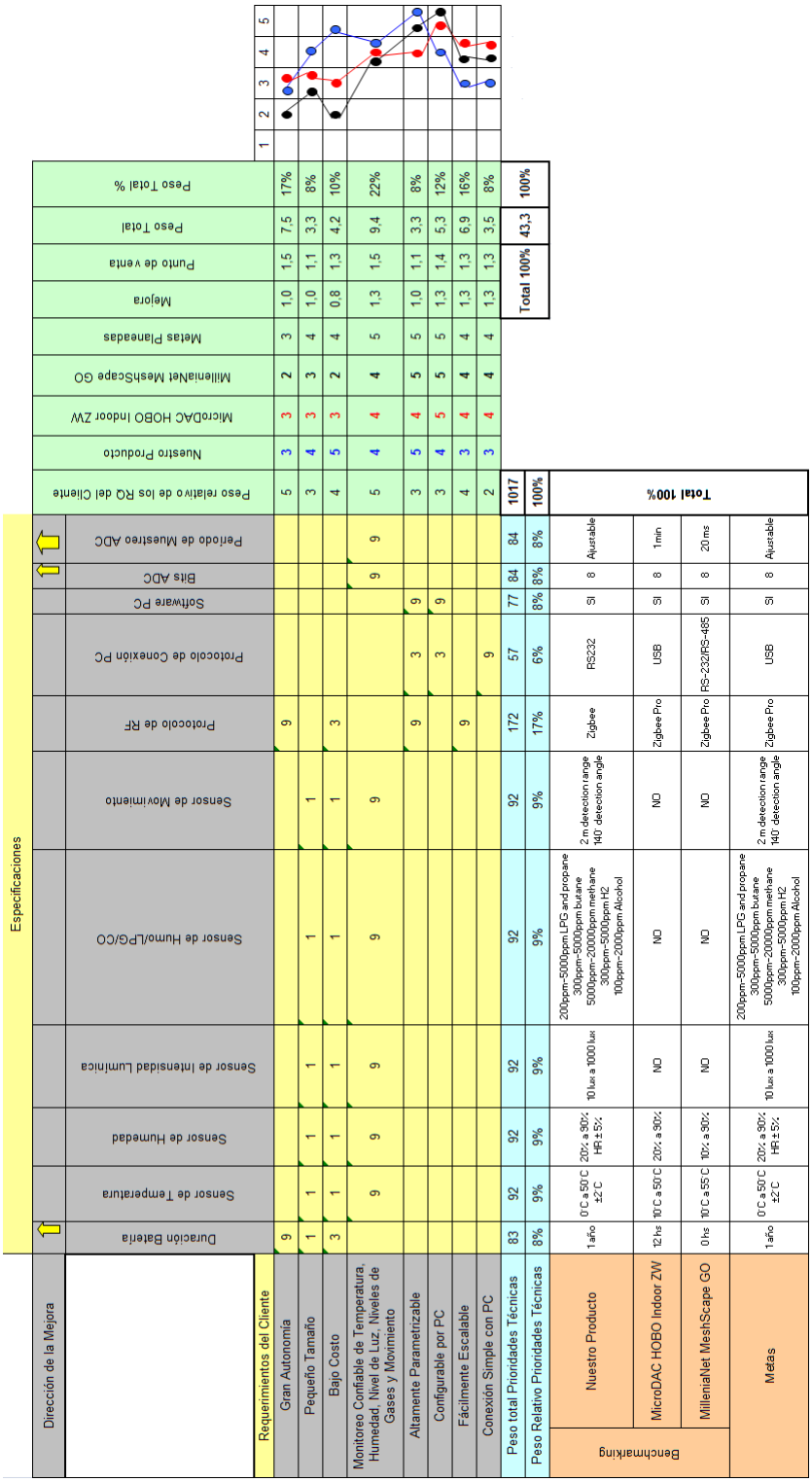

Figura 3: *Casa de Calidad*

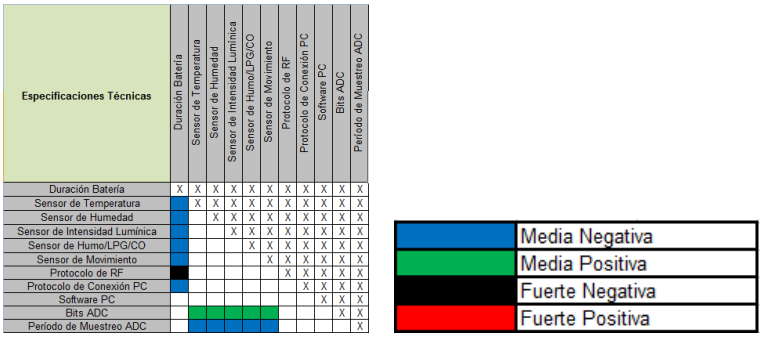

Figura 4: *Casa de calidad - Matriz de relaciones*

3.2. Especificaciones funcionales y de diseño

- Gran Autonomía: Cada nodo de la red debe poder operar con muy poca energía, de modo que su batería dure el mayor tiempo posible.
- Pequeño Tamaño: Cada nodo de la red debe ser pequeño para poder ser ubicado en cualquier lugar de la zona a monitorear.
- Bajo Costo: El sistema debe ser de bajo costo, en especial los nodos, de forma de que sean prácticamente descartables.
- Monitoreo de Variables del Ambiente: Se deben poder medir varias variables del ambiente.
- Altamente Parametrizable: Se debe poder configurar las propiedades de la red, así como activar/desactivar los sensores actuales o nuevos.
- Configurable por PC: La configuración debe poder ser hecha por PC.
- Fácilmente Escalable: La solución debe brindar un simple método para poder agregar más nodos a la red.
- Conexión Simple Con PC: La conexión a la PC debe ser muy sencilla para no generar complicaciones adicionales al usuario.

3.2.1. Especificaciones del hardware

- Duración de Batería: La duración de la batería es un punto clave de la solución, ya que los nodos no deben de necesitar manutención.
- Sensor de Temperatura: Se debe de poder medir temperatura.
- Sensor De Humedad: Se debe de poder medir humedad.
- Sensor de Intensidad Lumínica: Se debe de poder medir luz.
- Sensor de Humo/LPG/CO: Se debe de poder medir Gases en el ambiente.
- Sensor de Movimiento: Se debe de poder medir movimiento, de forma de controlar el acceso a zonas restringidas.
- Protocolo RF: El protocolo a utilizar es muy importante, ya que tiene una conexión directa con la duración de la batería, por lo que se utilizará el protocolo Zigbee, que permite un gran ahorro de energía.
- Protocolo de Conexión PC: Se realizará un protocolo para facilitar la transmisión de datos entre la red y la PC.
- Software PC: SE deberá tener un software de forma de facilitar el uso de la red y generar vistas al usuario.

- Bits ADC: Los bits del ADC son de alta importancia para la precisión de los sensores. Más bits, significa una mejor representación de la medición, aunque también depende de la precisión innata del sensor.
- Período de Muestreo ADC: Esto debe cumplir con el teorema de Nyquist, aunque debido a que la frecuencia de cambio de los sensores es muy baja, casi cualquier velocidad es adecuada.

3.2.2. Especificaciones del software

- Instalación simple. Que no requieran conocimientos avanzados para su instalación.
- Tener interfaz gráfica y no de consola.
- Múltiplataforma, es decir, que pueda ejecutarse en distintos sistemas operativos.
- Disponibilidad de idiomas.
- Graficar el estado de los sensores.
- Envío de alertas.
- Guardar estado de los sensores a un archivo.
- Permitir el cambio de unidad de la magnitud sensada.
- Utilice pocos recursos de la PC.

4. Análisis de Factibilidad

4.1. Factibilidad tecnológica

4.1.1. Propuesta de alternativas de diseño

Analizando varios papers y en especial *"A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi"* escrito por Jin-Shyan Lee, Yu-Wei Su, y Chung-Chou Shen [14] podemos extraer los siguientes gráficos que reflejan la elección de ZigBee para nuestro proyecto.

Standard	Bluetooth	UWB	ZigBee	Wi-Fi
IEEE spec.	802.15.1	802.15.3a *	802.15.4	802.11a/b/g
Frequency band	2.4 GHz	3.1-10.6 GHz	868/915 MHz; 2.4 GHz	2.4 GHz; 5 GHz
Max signal rate	1 Mb/s	110 Mb/s	250 Kb/s	54 Mb/s
Nominal range	10 m	10 m	10 - 100 m	100 m
Nominal TX power	0 - 10 dBm	-41.3 dBm/MHz	(-25) - 0 dBm	15 - 20 dBm
Number of RF channels	79	(1-15)	1/10; 16	14 (2.4 GHz)
Channel bandwidth	1 MHz	500 MHz - 7.5 GHz	0.3/0.6 MHz; 2 MHz	22 MHz
Modulation type	GFSK	BPSK, QPSK	BPSK (+ ASK), O-QPSK	BPSK, QPSK COFDM, CCK, M-QAM
Spreading	FHSS	DS-UWB, MB-OFDM	DSSS	DSSS, CCK, OFDM
Coexistence mechanism	Adaptive freq. hopping	Adaptive freq. hopping	Dynamic freq. selection	Dynamic freq. selection, transmit power control (802.11h)
Basic cell	Piconet	Piconet	Star	BSS
Extension of the basic cell	Scatternet	Peer-to-peer	Cluster tree, Mesh	ESS
Max number of cell nodes	8	8	> 65000	2007
Encryption	E0 stream cipher	AES block cipher (CTR, counter mode)	AES block cipher (CTR, counter mode)	RC4 stream cipher (WEP), AES block cipher
Authentication	Shared secret	CBC-MAC (CCM)	CBC-MAC (ext. of CCM)	WPA2 (802.11i)
Data protection	16-bit CRC	32-bit CRC	16-bit CRC	32-bit CRC

Cuadro 1: *Tabla comparativa*

Vemos el dato más importante, que es la cantidad de nodos, ZigBee soporta más de 65000, con lo cual es más que suficiente para nuestros propósitos, mientras que las demás tecnologías rondan unas pocas decenas o a lo sumo cientos.

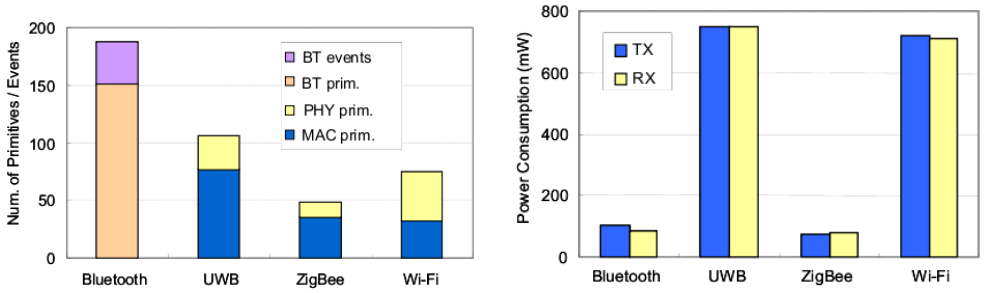

Figura 5: *Características de complejidad y consumo*

En cuanto a la complejidad para empezar a desarrollar nuestra aplicación vemos que ZigBee es de las más sencillas (en cuanto a directivas propias de la tecnología), y un dato no menos relevante es su ultra bajo consumo con respecto también a sus competencias.

Una vez que se decidió por esta tecnología, la decisión de que chip usar fue simple ya que la Facultad cuenta con módulos de ZigBee disponibles para ser utilizados en este proyecto.

4.1.2. Elección de una solución

Para el módulo ZigBee como comentamos en la sección anterior, la opción elegida fue **ZigBit ATZB-24-B0 de Atmel**. Se resume a continuación las características principales junto con el conjunto de alternativas estudiadas. Solo se considero aquellos que incluían el microcontrolador junto con el "transceiver".

Fabricante	Módulo	MCU	RAM	Flash	TX	RX	Interface	Opciones de Firmware	Precio
Atmel	ATZB-24-B0	8-bit ATmega 1281v	8 kB	128 kB	18 mA	19 mA	UART, USART, SPI, I²C, JTAG	ZigBee PRO, BitCloud stack, Wireless MCU Software, SerialNet, OpenMAC	35
Nivis, LLC.	VersaNode 210	32-bit ARM7	96 kB	128 kB	60 mA	21 mA	UART, SPI	Nivis ISA100.11a	90
Crossbow Technology	XM2110CA	8-bit ATmega1281	8 kB	128 kB	17 mA	16 mA	UART, SPI, I²C	ZigBee stack, Moteworks platform	75
MeshNetics	MNZB-24-B0	8-bit ATmega 1281v	8 kB	128 kB	18 mA	19 mA	UART, USART, SPI, I²C, JTAG	BitCloud stack, SerialNet, OpenMAC	40
Digi International Inc.	Series 2 XBee ZB	16-bit 12 MHz RISC	5 kB	128 kB	35 mA	38 mA	UART	Ember ZNet	32

Cuadro 2: *Características de Módulos ZigBee*

Se puede observar que el módulo elegido, tanto el consumo en la recepción como en la transmisión es de los más bajos, únicamente igualado por la alternativa de Crossbow que utiliza el mismo MCU pero no brinda las opciones de firmware en las cuales se destaca ZigBee, como el stack ZigBee PRO, y serialNet. Además un punto importante es el costo, dicha alternativa cuesta más del doble. Por lo tanto consideramos una buena decisión la elección del módulo ATZB-24-B0 de Atmel.

En cuanto a los sensores, ya que este trabajo pretende desarrollar la fabricación de un prototipo se optaron por aquellos que presentaron un menor costo.

Se resume a continuación una tabla con las principales características de los distintos tipos de sensores utilizados.

- Sensores de Gas

Model	Sensor resistance	Change Ratio of Sensor Resistance	Optimal detection concentration	Power consumption	Cost
TGS-203	1kΩ - 15kΩ	1.50 - 0.73	50 ppm - 1000 ppm	750 mW	15
MQ-2	3KΩ - 30KΩ	≤ 0.6	200 ppm - 5000 ppm	less than 800mw	2.5
TGS-813	5kΩ - 15kΩ	0.60 ± 0.05	500 ppm - 3000 ppm	835mW (typical)	18

Cuadro 3: *Características de sensores de gas*

La mejor alternativa en este caso es utilizar el sensor **MQ-2** por su bajo costo y características similares frente a sus competidores.

- Sensores de Temperatura

Model	Range	Accuracy	Linearity	Cost
LM35	−55 °C to +150 °C	± 0.5 °C	± 0.02 %	6
HEL-776	−200 °C to +260 °C	± 0.8 °C	± 0.1 %	18
DTH11	0 °C to +50 °C	± 2 °C	± 0.8 %	0.65

Cuadro 4: *Características de sensores de temperatura*

Aquí la mejor opción la presenta el sensor LM35 por su rango de medición y costo. Sin embargo, para la construcción del prototipo lo más adecuado por cuestiones de costo el DTH11 es la mejor alternativa. Luego de realizar las primeras pruebas, se reemplazará por el LM35 que es un sensor de precisión, para volver a la etapa de pruebas, pero estas se asume que no demandarán mucho tiempo y no representarán un aumento de costos para la realización del proyecto.

- Sensores de Humedad

Model	Range	Accuracy	Response Time	Stability	Cost
HIH-4000A	0 to 100%	± 3.5%	15 s	± 0.2% in 1 Year	7.5
HCT01	0 to 100%	± 2%	≤ 6 s	± 0.1% in 1 Year	15
DTH11	20% to 90%	± 4%	10 S	± 1 in 1 year	0.65

Cuadro 5: *Características de sensores de humedad*

Al igual que en el caso anterior, se optará por la elección del HIH-4000A para la salida al mercado, pero para el desarrollo del prototipo y la etapa de pruebas preliminares se elegirá el DTH11 el cual como vimos mide además temperatura.

4.1.3. DFMEA

Se realizará un análisis inductivo y sistemático componente por componente de modo de identificar todos los posibles modos de falla de cada componente y sus efectos sobre el sistema. Posteriormente al análisis de modos y fallas se categorizarán los modos de fallas de cada componente de acuerdo a su criticidad, determinándose cuáles componentes del sistema necesitan ser estudiados en más profundidad, a los fines de eliminar las causas que originan las fallas en ese particular modo de fallas, procurar un mejor diseño, reduciendo la tasa de fallas y conteniendo el daño emergente. Para ello vamos a utilizar dos herramientas:

- **FMEA (failure mode and effects analisys)**. Es un método inductivo que parte de acontecimientos elementales, como ser falla de un dado componente, y busca determinar las consecuencias de tal falla.
- **FTA (fault tree analisys)**. Es una herramienta que sigue una metodología deductiva, partiendo de un acontecimiento que se juzga indeseado, y busca hallar los caminos críticos que conducen a dicho evento.

El siguiente paso consiste en establecer los potenciales efectos de cada falla, asignándole un índice de severidad. Para medir la gravedad de los efectos de cada modo de falla, se utiliza un factor de ponderación Pi, dando idea de las consecuencias de ese modo de falla. Las consecuencias pueden ser menores, significativas, críticas y catastróficas. Luego se estima para cada falla la tasa de ocurrencia, lo que permite calificar los eventos como muy poco probables, poco probables, probables y altamente probables.

Se establecerán los potenciales efectos de cada falla, asignándole un *índice de severidad* **S**, que será un número que va del 1 al 10.

Luego, para cada falla se estima su *tasa de ocurrencia* **O**, lo cual permite calificar los eventos como muy poco probables, poco probables, probables y altamente probables, para ello también le asignaremos un número que va del 1 al 10.

Dado que las consecuencias únicamente se manifiestan si la falla ocurre en manos del usuario, esto implica que, si se realizan tareas de detección previas, el riesgo dependerá de *la tasa de detección* **D** asociada al método de control, nuevamente consistirá en asignar un número que va del 1 al 10. Combinando todos estos valores se define un número de prioridad del riesgo **RPN** (risk priority number), donde

$$\text{RPN} = S \times O \times N$$

Valor que sirve para determinar la necesidad de mejoras. Este proceso de optimización implicará:

- Modificación del concepto de solución, con la finalidad de evitar la causa de falla o bien reducir su severidad.
- Mejoramiento de la fiabilidad, con la finalidad de minimizar la ocurrencia de la falla.
- Mejorar el proceso de detección para evitar que la falla se traslade al usuario.

Es decir, la finalidad última del FMEA es concluir con un plan de acción en el que consten los nuevos objetivos, los rediseños que se deben encarar, los ensayos, las fechas y los responsables. Concluidas las mejoras, se realiza un nuevo FMEA, y esto se repetirá hasta lograr que el nuevo RPN esté conforme con los objetivos, que en nuestro caso determinamos que sea menos a 90 (RPN<90).

Finalmente, el producto de la tasa de ocurrencia y la tasa de detección determinaran la proporción residual de partes cuya falla se admite pase al cliente.

Módulo fuente

Función	Modo de Falla	Mecanismo y causa de falla	Efecto de la falla	Detecciones / Prevenciones	Ocurrencia	Severidad	Detectabilidad	RPN	Acciones correctivas recomendadas (RPN > 90)
Proveer tensión a todos los componentes del circuito	Tensión menor a la nominal	Tensión de entrada insuficiente	Los componentes no se alimentan correctamente	Agregar una indicador cuando la tensión es menor a la nominal (sensaría y mostraría por pantalla o agregar un led)	3	4	6	72	
			No se reconocen los estados lógicos correctamente		2	5	7	70	
	Corriente insuficiente	La fuente se encuentra muy cargada	Los componentes no funcionan correctamente	Calcular el consumo total utilizando las hojas de datos y simulaciones	4	5	4	80	
	Sobre tensión	Regulador conectado al revés	Destrucción del regulador	Escribir en la placa el sentido de conexión del regulador	7	5	4	140	Conectar un diodo de forma tal de impedir que circule corriente en inversa
			Destrucción de componentes		7	8	5	280	
		Cortocircuito entre los terminales del regulador	Destrucción del regulador	Análisis del circuito impreso y verificación de las soldaduras	3	7	4	84	
			Sobre calentamiento excesivo del dispositivo		3	6	3	54	
	Picos de tensión	Caída de un rayo	Deterioro del regulador	Introducir fusibles	1	4	10	40	
Establecer un valor de referencia para los sensores	Lectura incorrecta de los sensores	Variación de la referencia	Las mediciones no son correctas	Medir la tensión de referencia antes de poner el equipo en producción	2	3	2	12	

Figura 6: *Módulo Fuente*

Módulo ZigBee

Función	Modo de Falla	Mecanismo y causa de falla	Efecto de la falla	Detecciones / Prevenciones	Ocurrencia	Severidad	Detectabilidad	RPN	Acciones correctivas recomendadas (RPN > 90)
Digitaliza las señales de los sensores	Entrada en cortocircuito	Sodadura defectuosa	Puede producir daños en el trasceiver	Realizar verificaciones de continuidad luego de soldar los pads	3	8	4	96	Separar los contactos en el diseño del PCB y verificar las reglas de diseño
			Puede dañar los sensores		4	6	4	96	
		Error en el diseño de las pistas	Puede producir daños en el trasceiver	Verificar las reglas de ruteo y de distancia mínima	3	8	3	72	
			Puede dañar los sensores		3	6	3	54	
	Entrada a circuito abierto	Error en el diseño de las pistas	Lectura errónea de los sensores	Realizar verificaciones de continuidad	3	5	3	45	
	Señal con ruido	Inducción de pistas próximas	Lectura errónea de los sensores	Uso de software con simulación electromagnética	5	5	1	25	
		Error de ruteo del plano de masa	Valor de referencia incorrecto para los sensores	Uso de software con simulación electromagnética	2	4	1	8	
		Mala distribución de los componentes en la placa	Lectura errónea de los sensores	Uso de software con simulación electromagnética	4	5	1	20	
		Pistas actuando de antena	Interferencia de señales inalámbricas		1	6	2	12	
Comunicación inalámbrica con otros módulos	Potencia de transmisión insuficiente	Distancia máxima superada	No se realiza la comunicación	Realizar una medición experimental para verificar la distancia máxima	4	5	5	100	Utilizar una distancia menor para realizar la comunicación
			Pérdida de tramas		3	4	5	60	
		Alimentación deficiente	No se realiza la comunicación	No cargar la fuente o utilizar una que entregue mayor corriente	2	5	3	30	
			Pérdida de tramas		2	4	3	24	
	Fallo en la comunicación inalámbrica	Pérdida aleatoria de datos	Pérdida de tramas	Realizar pruebas en campo antes de producir el equipo	2	4	10	80	
		Interferencia en el espectro radioeléctrico	Pérdida de tramas	Se utilizará otro canal que no este siendo utilizado por otro dispositivo o sistema	5	4	4	80	
			No se realiza la comunicación		5	5	3	75	
			Elección errónea del canal de comunicación		5	4	3	60	
		Interferencia por campo magnético	Pérdida de tramas	Utilizar blindaje en los componentes de radiofrecuencia	1	4	9	36	
			No se realiza la comunicación		1	5	5	25	
		Imposible realizar la conexión inalámbrica	Mala configuración de los transceivers	Verificar la hoja de datos	4	4	5	80	
			Elección errónea del canal de comunicación		1	5	5	25	

Figura 7: *Módulo ZigBee*

Sensores

Función	Modo de Falla	Mecanismo y causa de falla	Efecto de la falla	Detecciones / Prevenciones	Ocurrencia	Severidad	Detectabilidad	RPN	Acciones correctivas recomendadas (RPN > 90)
Tomar mediciones de los parámetros físicos	Posee mayor varianza que la nominal	Error en las especificaciones de los sensores	Lectura errónea de los sensores	Verificar la hoja de datos	2	6	3	36	
			Puede dañar al transceiver		1	9	2	18	
		Variación de los componentes que acondicionan la señal	Lectura errónea de los sensores	Simulación eléctrica de los circuitos. Verificar la hoja de datos	3	6	6	108	Simulación de montecarlo. Buscar una alternativa para el sensor
			Puede dañar al transceiver		1	9	6	54	
		Error en la calibración	Lectura errónea de los sensores	Simulación eléctrica de los circuitos	4	6	6	144	Calibración con patrones de referencia
			Puede dañar al transceiver		1	9	6	54	
	Aumento de la varianza con el tiempo	Error en las especificaciones de los sensores	Lectura errónea de los sensores	Verificar la hoja de datos	2	6	3	36	
			Puede dañar al transceiver		1	9	3	27	
		Envejecimiento de los componentes del circuito que acondiciona la señal	Lectura errónea de los sensores	Simulación eléctrica de los circuitos. Verificar la hoja de datos	3	6	6	108	Simulación de montecarlo. Buscar una alternativa para el sensor
			Puede dañar al transceiver		1	9	6	54	
	Disminución de la exactitud con el tiempo	Error en las especificaciones de los sensores	Lectura errónea de los sensores	Verificar la hoja de datos	2	6	3	36	
			Puede dañar al transceiver		1	9	3	27	
		Envejecimiento de los componentes del circuito que acondiciona la señal	Lectura errónea de los sensores	Simulación eléctrica de los circuitos. Verificar la hoja de datos	3	6	6	108	Simulación de montecarlo. Buscar una alternativa para el sensor
			Puede dañar al transceiver		1	9	6	54	

Figura 8: *Módulo Sensores*

4.2. Factibilidad de tiempos

La planificación del proyecto intenta determinar cuál será la fecha de finalización del proyecto y qué recursos hacen falta para lograrlo. Debemos garantizar el éxito del proyecto, para lo que no alcanza solamente con un estudio de factibilidad técnica que haya concluido positivamente. Un proyecto exitoso considera, además de satisfacer los requerimientos técnicos, el tiempo y el resultado económico como variables fundamentales del mismo. Por lo tanto es importante demostrar no sólo la factibilidad técnica, sino que además interesa saber si tiene sentido económicamente y si se cuenta con la capacidad técnica y financiera para afrontar su desarrollo, y que pueda lograrse una solución para el mismo en un tiempo razonable. Podemos dividir el planeamiento en los siguientes puntos.

1. Definir las metas generales.
2. Trazar un plan de tareas.
3. Desarrollar en detalle los alcances.
4. Asignar objetivos para cada actividad.
5. Relacionar las actividades mediante una red lógica.
6. Establecer la duración y demoras de cada actividad.
7. Verificar la consistencia de la red.
8. Determinar la necesidad de recursos para cada actividad.

Las dos herramientas usuales para la planificación son el CPM (Critic Path Method) y el PERT (Program Evaluation and Review Technique). El primero está orientado a actividades, trabaja con tiempos determinísticos, considerando que la fluctuación en la duración de las tareas es despreciable. El segundo método está orientado a eventos y trabaja con tiempos aleatorios, resultando más apropiado para la planificación de proyectos, en los que se supone que al menos alguna de las tareas no se realizó nunca, ni es simple, ni su solución es directa, y que además exige esfuerzos de creatividad difíciles de cuantificar en tiempo.

4.2.1. Planificación con PERT

Resulta necesario contar con elementos precisos para decidir si conviene o no desarrollar el proyecto, siendo el tiempo y el costo, elementos importantes en tal decisión. Se sabe que las tareas comenzarán en un determinado momento y que, suponiendo el mejor de los casos, no podrán concluirse antes de una determinada fecha. Consideremos a este un tiempo optimista, y para el peor caso sabemos que se concluirán en un tiempo posterior al optimista, denominado tiempo pesimista. El tiempo efectivo será un valor aleatorio entre ambos límites, que habrá que estimar. La distribución Beta es un buen modelo para determinar esta incerteza.

$$f(x) = kx^{\alpha-1}(1-x)^{\beta-1} \tag{1}$$

Por ser asimétrica, para su caracterización estadística consideraremos la moda, además del valor medio y la varianza, cuyos valores están relacionados con los parámetros por las siguientes ecuaciones.

$$\mu = \frac{\alpha}{\alpha+\beta} \tag{2}$$

$$\sigma^2 = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha^2+\beta+1)} \tag{3}$$

$$\chi_{moda} = \frac{\alpha-1}{(\alpha-1)(\beta-1)} \tag{4}$$

Definiendo los tiempos optimista (t_o) y pesimista (t_p) queda definido el intervalo de incerteza, relacionado con el intervalo de existencia de la variable beta normalizada x, de la siguiente forma:

$$x = \frac{t - t_o}{t_p - t_o} \tag{5}$$

$$t = t_o \times x \times (t_p - t_o) \tag{6}$$

Al modelo, generalmente se agregan dos supuestos más, el tiempo medio y la varianza del modelo, que se ligan a las tres estimaciones anteriores. Para el tiempo medio se hace un promedio ponderado entre los tiempos to, tm y tp, de la siguiente manera.

$$\mu = \frac{t_O + 4t_M + t_P}{6} \tag{7}$$

Además se considera imposible que el rango supere en más de seis veces el desvío estándar, es decir que en el límite

$$\sigma^2 = \left(\frac{t_P - t_O}{6}\right)^2 \tag{8}$$

Las tareas necesarias para el desarrollo del primer prototipo son las siguientes. Se incluyen en este listado las tareas que ya fueron parcialmente realizadas en los informes anteriores.

1. **Estudio de mercado**: Se realiza un análisis de la necesidad y se estudian los productos similares que cubren dicha necesidad. Se buscan potenciales clientes y las patentes existentes.
2. **Estudio de la solución**: Se realiza un análisis del producto propuesto para satisfacer el punto anterior. Se realiza una especificación de los requerimientos del usuario y los requerimientos técnicos. Con esta información se desarrolla la casa de calidad.
3. **Selección de los componentes**: Se realiza una búsqueda de los componentes y materiales necesarios para cubrir los requerimientos del punto anterior.
4. **Diseño del hardware**: Se realiza el diseño de la conexión de la PC con el módulo y la conexión de los sensores con el módulo. Se realiza el esquemático final del prototipo.
5. **Diseño de la mecánica - presentación**: Se realiza un análisis de las regulaciones vigentes. Se diseña el gabinete.
6. **Arquitectura del software**: Se diseña el algoritmo principal, y los algoritmos para las mediciones. Se diseña la interfaz con el usuario, y los graficadores. Se hace un análisis de persistencia y se especifica una solución para la comunicación del módulo con la PC.
7. **Realización del prototipo**: Se construye el PCB y el gabinete.
8. **Verificación con especificaciones técnicas**: Se realiza una verificación general del prototipo desarrollado para establecer si cumple con todos los puntos anteriores.
9. **Certificación**: Se homologa el producto desarrollado ante los organismos correspondientes.
10. **Verificación por el cliente**: Se realizan pruebas de uso con el usuario final bajo un ambiente controlado.
11. **Realización de cambios**: De ser necesario se realizan las modificaciones correspondientes de acuerdo a las pruebas realizadas en el punto anterior.
12. **Realización del manual**: Se desarrolla el manual técnico y de usuario final para el producto desarrollado.
13. **Diseño del circuito impreso**: Se realiza el diseño prestando atención a las dimensiones requeridas y a cuestiones normativas y de compatibilidad electromagnética.

14. **Prueba final**: Se realiza una nueva prueba con usuarios.
15. **Salida al mercado**: Se lanza el producto al mercado consumidor.

El siguiente paso corresponde a estimar los tiempos y secuencia de todas las tarea. Tal como se definió en la introducción para cada una de ellas:

- **Tiempo optimista**: (t_O). Es el tiempo mínimo en que se pudiere realizar el proceso (en las mejores condiciones) cumpliendo con lo mínimo establecido.
- **Tiempo pesimista**: (t_P). Indica el mayor tiempo que se considera que pudiere demandar la tarea bajo las peores condiciones.
- **Tiempo más probable** (t_M). Es el tiempo que lleva la tarea en condiciones normales de realización.
- μ: Valor medio de la distribución de probabilidad del tiempo de tarea.
- σ: Varianza de la distribución de probabilidad del tiempo de tarea.

Todas estos valores son especificados en cantidad de jornadas laborales típicas de 8 horas de trabajo.

Los valores μ y σ se obtienen suponiendo que la distribución de probabilidad de la duración de cada tarea corresponde a una función de densidad de probabilidad *Beta normalizada*. El valor medio y el desvío estándar se calculan mediante las siguientes aproximaciones:

$$\begin{aligned} \mu &= \frac{t_O + 4t_M + t_P}{6} \\ \sigma &= \frac{t_P - t_O}{6} \end{aligned}$$

Sin embargo, para este estudio se utilizó un software adecuado para esta tarea como lo es el Microsoft Project.

	Task Name	Duration	Optimistic Dur.	Expected Dur.	Pessimistic Dur.
1	**Estudio de Mercado**	**3,67 days**	**2 days**	**4 days**	**4 days**
2	Análisis de Necesidad	1,83 days	1 day	2 days	2 days
3	Búsqueda de Clientes Potenciales	1,83 days	1 day	2 days	2 days
4	Búsqueda de Productos Similares	1,17 days	1 day	1 day	2 days
5	Búsqueda de Patentes	1,17 days	1 day	1 day	2 days
6	**Estudio de la Solución**	**7,33 days**	**4 days**	**7 days**	**12 days**
7	Análisis del Producto Propuesto	2 days	1 day	2 days	3 days
8	Búsqueda de Requerimientos del Usuario	2 days	1 day	2 days	3 days
9	Búsqueda de Requerimientos Técnicos	2,17 days	1 day	2 days	4 days
10	Armado de Casa de Calidad	1,17 days	1 day	1 day	2 days
11	**Selección de Componentes**	**3 days**	**1 day**	**3 days**	**5 days**
12	Sensores	3 days	1 day	3 days	5 days
13	Conversor Serie-USB	3 days	1 day	3 days	5 days
14	Módulo Zigbee	3 days	1 day	3 days	5 days
15	Pedido de Componentes	32,5 days	15 days	30 days	60 days
16	Estudio de Costos	4,83 days	2 days	5 days	7 days
17	**Diseño de Dispositivo**	**58 days**	**28 days**	**56 days**	**96 days**
18	**Diseño de Hardware**	**11,67 days**	**5 days**	**12 days**	**17 days**
19	Diseño de Conexión PC-Módulo	5 days	3 days	5 days	7 days
20	Diseño de Conexión Sensores-Módulo	6,83 days	3 days	7 days	10 days
21	Armado de Esquemático Final	4,83 days	2 days	5 days	7 days
22	**Diseño de Mecánica - Presentación**	**6 days**	**5 days**	**6 days**	**7 days**
23	Análisis Regulaciones	5 days	3 days	5 days	7 days
24	Diseño 3D de la carcaza	6 days	5 days	6 days	7 days
25	**Arquitectura de Software**	**42,5 days**	**20 days**	**40 days**	**75 days**
26	Diseño de Algoritmo Principal	2 days	1 day	2 days	3 days
27	Interfaz de Usuario	14,5 days	7 days	15 days	20 days
28	Diseño de Graficadores	20 days	10 days	20 days	30 days
29	Diseño de Persistencia	3,17 days	2 days	3 days	5 days
30	Algoritmos para Análisis de Medicione	3,17 days	1 day	3 days	6 days
31	Comunicación PC-Módulo	5 days	3 days	5 days	7 days
32	**Realización de Prototipo**	**38 days**	**26 days**	**37 days**	**54 days**
33	Construcción del PCB	30,83 days	20 days	30 days	45 days
34	Contruccción Mecánica	30,83 days	20 days	30 days	45 days
35	Ensamble Final	1,17 days	1 day	1 day	2 days
36	Verificación Con Especificaciones Técnicas	7,17 days	5 days	7 days	10 days
37	Certificación	21,67 days	20 days	20 days	30 days
38	Validación por el Cliente	10 days	5 days	10 days	15 days
39	Realizar Cambios de Ser Necesario	15 days	10 days	15 days	20 days
40	Realización del Manual	3,17 days	2 days	3 days	5 days
41	Prueba Final	9,5 days	5 days	10 days	12 days
42	Salida al Mercado - Primera Producción	1 day	1 day	1 day	1 day

Cuadro 6: *Listado de tareas*

4.2.2. Simulación de Montecarlo

Para realizar un análisis más confiable se puede hacer una simulación de Montecarlo. La misma permite analizar el resultado arrojado por una corrida de N simulaciones utilizando las densidades de probabilidad individuales de cada tarea.

Para ello se utilizó un programa escrito en *Octave* y se realizaron 50000 simulaciones diferentes de las duraciones de cada tarea viendo en cada caso cuál resultaba ser el crítico. Los resultados se presentan gráficamente a continuación.

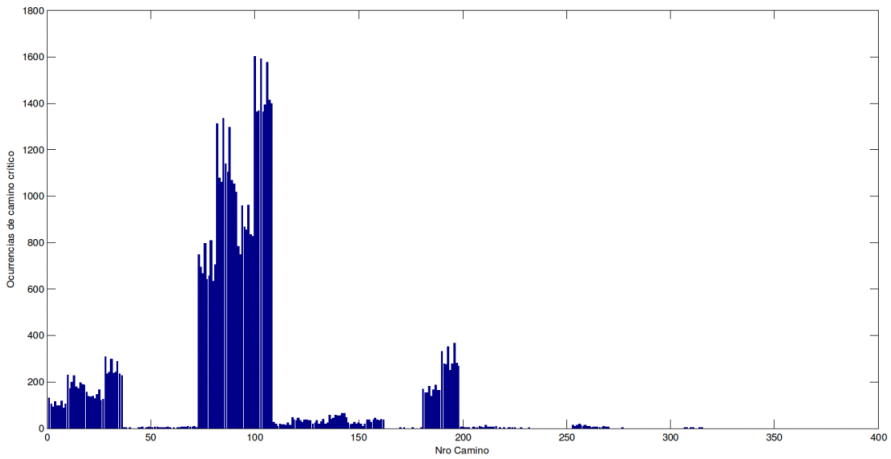

Figura 9: *Ocurrencias de camino crítico*

Un detalle de aquellos caminos cuyo número de ocurrencias fue mayor se muestra a continuación.

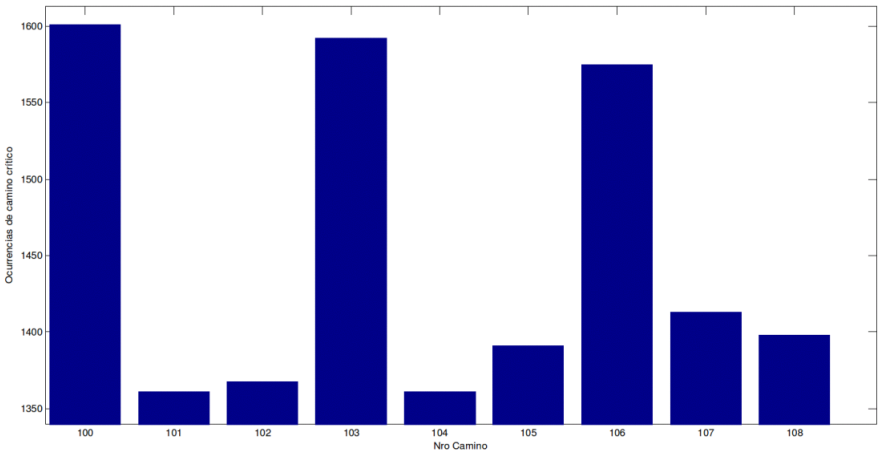

Figura 10: *Detalle de caminos con mayor ocurrencia*

Puede verse que aquellos con más ocurrencias son el 100, 103 y 106 respectivamente, los cuales son:

- 2 3 7 8 9 10 12 15 16 20 21 24 34 35 36 37 39 41 42
- 2 3 7 8 9 10 13 15 16 20 21 24 34 35 36 37 39 41 42
- 2 3 7 8 9 10 14 15 16 20 21 24 34 35 36 37 39 41 42

Para estos se trazaron histogramas de duración que se presentan a continuación.

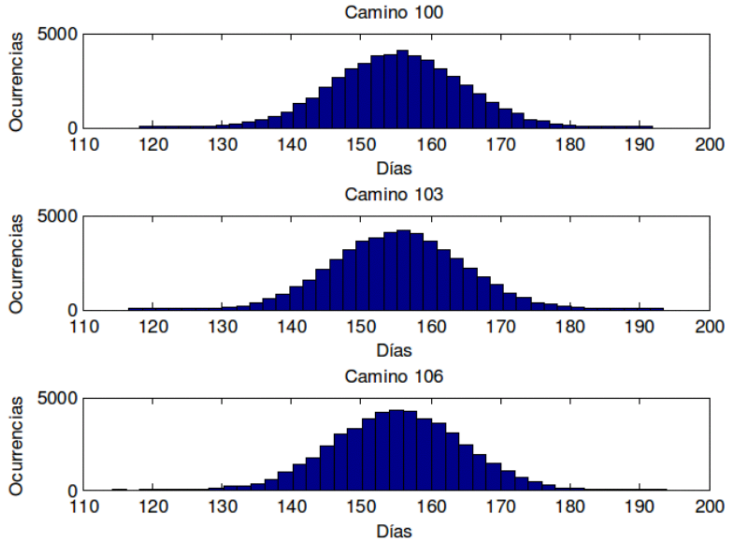

Figura 11: *Histogramas*

Así, por Montecarlo, la media del camino crítico 180 es 155,37 días y su desvío 9,21 días.

Puede entonces decirse que:

El tiempo de duración del proyecto será menor o igual a 173,79 días con una confianza del 95 %, según análisis de Montecarlo.

4.2.3. Programación con Gantt

El diagrama Gantt permite distribuir las tareas temporalmente para visualizar mejor las tareas que pueden realizarse simultáneamente.

En el diagrama se encuentra resaltado el camino crítico para destacar cuáles son las tareas a comenzar sí o sí y cuáles pueden llegar a retrasarse sin retrasar el desarrollo del proyecto

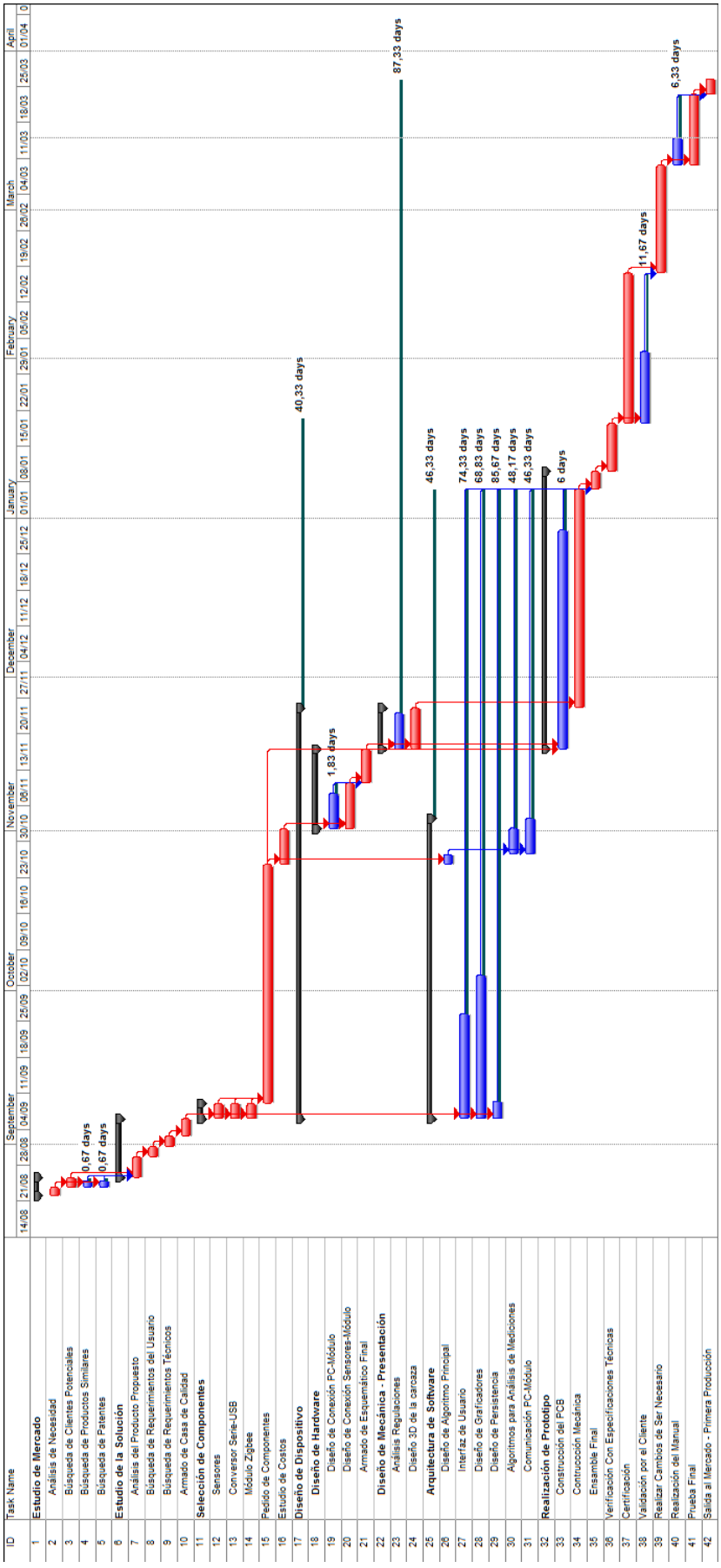

Figura 12: *Diagrama de tiempos*

4.3. Factibilidad económica

4.3.1. Mercado

De acuerdo a la necesidad planteada y que intentamos solventar, el mercado para nuestro producto tiene como clientes potenciales a las empresas medianas que cuentan con un data center.

Según un estudio privado llevado a cabo por la consultora *Prince and Cooke* de las empresas argentinas de tamaño mediano, 52 % cuenta con un data center propio, mientras que el porcentaje baja a 44 % entre las firmas que cuentan con una plantilla de entre 50 y 100 empleados.

La expansión de los data center fue de la mano de las empresas de distintos rubros, con el objetivo de almacenar en forma efectiva información sensible en espacios y soportes físicos a los que pueda acceder como propios. Este escenario no prosperó en las pymes de menos de 50 empleados y mucho más lejos aún quedaron las empresas de hasta cinco puestos de trabajo o los profesionales independientes.

Por otro lado, para conocer la aceptación de esta tecnología en otros lugares, nos encontramos con un estudio de la consultora **OnWorld** llamado *"Wireless Sensor Networks: Growing Markets, Accelerating Demand"* [16], y es basado en los resultados de una encuesta realizada con 147 empresas. Incluye predicciones sobre producción e ingresos en cinco mercados de redes de sensores sin cable, como lo son monitoreo y control industrial, edificios comerciales inteligentes, control y automatización residencial, mediciones avanzadas y utilidad para infraestructura de redes, además de análisis y resultados sobre motivación del sector, avances en las aplicaciones actuales de redes de sensores y las últimas tendencias del mercado.

El informe concluye que existe en la actualidad una demanda "abrumadora" de soluciones ofrecidas por redes de sensores sin cable por parte de empresas. De las empresas sondeadas para el estudio, 29 por ciento utilizan WSN (Wireless Sensor Network) mientras que más de 40 por ciento consideran muy probable la posibilidad de que realicen pruebas piloto con redes de sensores sin cable dentro del próximo año y medio.

Nuestro enfoque por una cuestión de logística y estar cerca del cliente, nos concentraremos exclusivamente en Buenos Aires, dejando para un futuro una posible expansión y cubrir más del territorio nacional.

Según el observatorio PyMe de la UIA, hay en Argentina, inscriptas formalmente en AFIP, unas 559.000 empresas. De ellas solo 5.300 son grandes y el resto, unas 554.000, son PyMes.

De aquellas, en la provincia de Buenos Aires existen unas 170.000 empresas o emprendimientos económicos (son exactamente 169.240). Representan el 30,28 % del total nacional.

Tienen, en Bs. As., menos de 20 empleados unas 158.000 empresas; entre 20 y 90, unas 9.500 empresas; entre 90 y 1.000, unas 2.500 y 80 tienen más de 1.000 empleados.

De las empresas que existen en Bs. As, unas 16.000 se dedican a la actividad primaria, 28.866 a la industria, 44.735 al comercio, y hay unas 7.730 sin clasificar. En la economía bonaerense en total, lo servicios representan un 60 % del PBI y la producción de bienes un 40 %

Considerando el primer estudio, elegiremos las empresas de tamaño medio y aquellas con una cantidad de empleados entre 50 y 100.

Empleados	Cantidad de empresas
20 - 90	9500
90 - 1000	2500
Total	12000

Se estima de acuerdo a los números la UIA que alrededor de un 34 % de este total de empresas son posibles compradores, de los cuales se pretende que comprarán 1 unidad del producto cada una, un kit básico para empezar (coordinador y dos nodos), con lo cuál tendríamos unos 4000 artículos vendidos en aproximadamente 5 años.

Ahora podríamos hacer una proyección aproximada de las ventas, sabiendo que el máximo se alcanza a los 2 años:

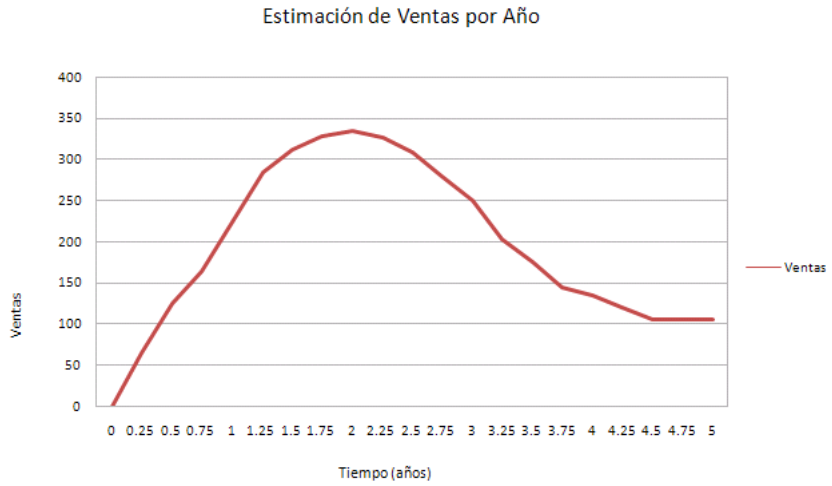

Figura 13: *Estimación de Ventas*

4.3.2. Competencia

Actualmente no existen empresas argentinas que produzcan esta solución, y sí encontramos las siguientes empresas extranjeras. Podemos tomar como ejemplo, un estudio de la consultora OnWorld:

- Millennial Net [9] Millennial net ofrece una red inalámbrica de sensores que contiene software, hardware y accesorios para armar la red. Contiene 5 nodos externos, 3 internos y un gateway. También trae baterías, cable para cargar y las antenas. El precio es de $4500(dólares).
- MicroDAQ [8] El producto, llamado "Sensicast Wireless SensorNet System" provee una red inalámbrica para sensado y control. Los sensores pueden ser de temperatura, humedad y proximidad. Precios de $2999 a $3499 (dólares).
- Ember [2] El objetivo del producto es el armado rápido de la red. Contiene todo lo necesario para conectar y que funcione. El precio es de $4995(dólares).
- Crossbow [10] Crossbow es la empresa más grande en el desarrollo de red de sensores, son utilizados en una gran variedad de lugares, cerrados, al aire libre, para seguridad y monitoreo, del ambiente, de una fabrica, etc. El precio es de $2995 (dólares). por la red y luego se compran los módulos necesarios, por ejemplo, el de monitoreo del clima tiene un precio de $850 (dólares).

Esta última empresa representa nuestra más importante competencia, la cual desembarcó en el país con una pequeña representación local a través de la empresa Open Automation, de todas formas se brindan soluciones a un precio muy elevado para lo que es el mercado en nuestro país.

4.3.3. Costos

A continuación se resumen los distintos costos que componen el desarrollo total del producto y veremos cuanto es la inversión inicial para construir y salir al mercado con nuestro producto.

De acuerdo a la programación del proyecto de la sección anterior, se necesita un tiempo de desarrollo de unos cinco meses, hasta tener un prototipo listo para la venta. Posteriormente se comenzará a vender de forma paulatina, hasta lograr insertarse en el mercado.

Se estima que la inversión inicial será la necesaria para financiar los primeros 24 meses del proyecto, y pasados este tiempo, se comenzará a recuperar la inversión inicial. Y al finalizar el segundo año del proyecto se recuperará la inversión inicial.

A lo largo de todo el proyecto, incluyendo los meses en los que no hay ganancia, se deberá considerar gastos fijos mensuales, en los que se incluye tanto el lugar donde se fabricará el producto como también el gasto mensual para publicitarlo. También se tienen que tener en cuenta los recursos humanos necesarios. Los mismos se calculan de acuerdo al gasto neto por mes que implican y a la cantidad en función del tiempo calculada para manejar el volumen de trabajo y ventas.

Se supone la utilización en el proyecto de un ingeniero, un programador y un técnico. El ingeniero participará durante toda la etapa de desarrollo del proyecto, el programador a partir de la etapa de diseño de la arquitectura de software hasta la mejora del prototipo inclusive.

La ocupación de estos dos recursos será de tiempo completo disminuyendo la carga horaria del ingeniero durante la etapa de construcción de las placas, gabinete y armado del prototipo, tareas de las cuales se encargará el técnico. El ingeniero retomará a los trabajos a tiempo completo cuando el programador termine con el desarrollo del software, para continuar con los ensayos del prototipo.

Tomaremos como sueldos brutos promedios un monto mensual de $7020 para el ingeniero y $6000 para el programador y de $3520 para el técnico.

Se considera el siguiente criterio para determinar el monto por hora que se asignará a cada profesional:

$$20 \text{ días laborables mensuales } \Rightarrow 160 \text{horas laborables mensuales}$$

De lo anterior se desprende lo siguiente:

- Ing. / hora = $7020 / 160 hs = $ 43.9 / hs
- Prog. / hora = $6000 / 160 hs = $ 37.5 / hs
- Téc. / hora = $3520 / 160 hs = $ 22 / hs

Estos números contemplan la variación salarial debida a la inflación y los descuentos por carga social. Representan un promedio de lo que gastaremos en lo referido a sueldos de empleados.

Estos costos se darán de manera fija sólo dentro de la etapa específica del proyecto y no en otras, es decir, por las tareas que llevará a cabo el programador se le deberá pagar una suma “fija” en concepto de sueldo, luego de terminada la etapa de desarrollo ya no se incurrirá en este gasto ya que tal vez no sea necesario repetir la tarea de programación. Es lógico pensar que como el trabajo esta dado en horas y la paga depende de cuantas horas este trabajando cada profesional que este costo parezca variable, pero ya se han determinado anteriormente cuantas se le asigna a cada tarea por lo que han quedado “fijadas” y por ende el costo ha quedado fijado también.

Tipo de Gasto		Especificación	Monto
Costo Primario	*Material Directo*	Módulo Zigbee	$ 170
		Sensores	$ 100
		Conversor Serie-USB	$ 24
		Componentes Varios	$ 50
		Carcaza	$ 50
		MAX3232	$ 15
		Sub Total	$ 409
	Trabajo Directo - Salarios Durante 6 meses		$ 165,400
	Mantenimiento		$ 2,000
	Suministros		$ 5,000
	Servicios(Luz, agua, etc)		$ 5,000
	Sub Total		$ 177,400
Total - Costo Primario			$ 177,809

Cuadro 7: *Costo primario*

Tenemos un **costo primario total de $ 177.809**

En lo que respecta a lo anterior, se consideró ya la etapa de producción en serie, es el costo más bajo que se puedo obtener con la fabricación en serie.

Tipo de Gasto		Especificación	Monto
Costo De Fábrica	*Costo de Producción Indirectos*	Impuestos	$ 15,000
		Seguros	$ 10,000
		Sub Total	$ 25,000
	Costo Primario		$ 177,809
Total - Costo de Fábrica			$ 202,809

Cuadro 8: *Costo de fabrica*

Tenemos un **costo de fábrica de $ 202.809**

Tipo de Gasto		Especificación	Monto
Costo De Producción	*Gastos Generales Fijos*	Investigación y Desarrollo	$ 6,000
		Contaduría y Auditoría	$ 4,000
		Asesoramiento Legal	$ 6,000
		Relaciones Públicas(tipo RRHH)	$ 6,000
		Costo de Dirección y Administración	$ 4,000
		Alquileres	$ 35,000
		Costos de Ventas y Distribución	$ 9,000
		Sub Total	$ 107,628
	Costo De Fábrica		$ 202,809
Total - Costo de Producción			$ 310,437

Cuadro 9: *Costo de producción*

Tenemos un **costo de producción de $ 310.437**

En los costos de venta y distribución se contempla el hecho que se venderá por internet ofreciendo en nuestra web la información adecuada junto con las estrategias de marketing apropiadas para hacerla atractivo al cliente y mediante un contacto telefónico o vía email se cerrará el contrato con el cliente.

El alquiler se contempla para el hecho de recibir las llamadas. No es necesario contar con un espacio muy grande ya que las reuniones se harán en la sede del cliente donde se analizarán y estudiarán la cantidad exacta de nodos necesarios para cubrir sus necesidades. De todas formas se realizó el análisis considerando que el kit básico que se adaptará a la mayoría de los clientes.

No es necesario considerar un lugar más amplio además porque el ensamblado y la fabricación final de las placas una vez que el prototipo se ha verificado y testeado y se encuentra en condiciones de salir al mercado se realizarán afuera del país, es decir terciarizaremos su fabricación. Esta decisión se determinó porque resulta más económica importarlas que realizarlas dentro de nuestro país.

Tomamos el caso de una fabrica en China *Sky-Electronic-Mart* contactada por eBay que ofrece una producción de 200 placas con componentes con envío incluido en 7 días hábiles por 80 dólares cada una, casi el total del costo de conseguir los componentes en nuestro país únicamente. Tomamos como contraposición una fábrica de ensamblado de placas nacional *Dai Ichi Circuitos S.A.* que cuesta más del doble que el ejemplo anterior y el tiempo de demora depende de la cantidad de placas a realizar, por un pedido de 200 placas demorarían alrededor de tres semanas.

A continuación resumimos los gastos totales de desarrollo.

Tipo de Gasto	Especificación	Monto
Costo Total	*Beneficio (%)*	0
	Gasto Total Fijo Por 6 meses	$ 310,028
	Gasto Total Fijo Por 6 meses - Con Beneficio	$ 310,028
	Gasto Variable Por 6 meses	$ 409
	Gasto Variable Por 6 meses - Con Beneficio	$ 409
	Costo de Producción	$ 310,437
Gasto Total - Desarrollo		$ 310,437

Cuadro 10: *Costo total de desarrollo*

Aquí tenemos en cuenta que estamos trabajando con una fiabilidad de 0.9, es decir, de 100 equipos que produzcamos es esperable que 10 fallen en el período que dura la garantía, es decir 4 años. Con lo cual, agregamos como gasto de producción el 10 % de la cantidad de productos que se espera vender en cuatro años, ya que esos equipos fallarán y tendrán que ser descartados.

Para obtener el precio de venta del producto, se tienen en cuenta los costos directos e indirectos de producir la unidad.

Para el costo directo, será necesario tener en cuenta los costos de las materias primas, así como también los costos asociados a la fabricación, como son la electricidad o uso de las instalaciones, y los sueldos del personal necesario para su fabricación.

Además del costo directo de fabricación, es necesario tener en cuenta el costo indirecto, compuesto por los gastos mensuales de alquiler, impuestos y servicios así como también los sueldos de los empleados administrativos.

Por último, será necesario tener en cuenta el costo financiero, relacionado con la cuota del crédito a pagar durante los 60 meses de duración de la vida del producto.

Podemos pretender un costo de producto fijo para un cálculo inicial que luego se puede ir variando de acuerdo a lo que cueste ingresarse en el mercado. No se plantea un beneficio en el dinero destinado para el desarrollo del equipo para determinar su precio de venta.

Años para recuperar el gasto	1
Artículos Vendidos en ese Tiempo	580
Precio por Artículo	$ 1,478

Cuadro 11: *Precio por articulo*

Vemos que el precio por articulo, es decir el **costo del kit básico** para devolver la inversión inicial en el primer año luego de salir al mercado, tiene que ser de **$1.478** cuando la cantidad de artículos que se espera vender en ese momento es de 580.

El precio de venta se estimó sumando todos los gastos fijos opr sueldos y gastos variables por material, distribución y venta, armado del kit e impuestos. Luego, establecemos como parámetro la ganancia que se quiere obtener.

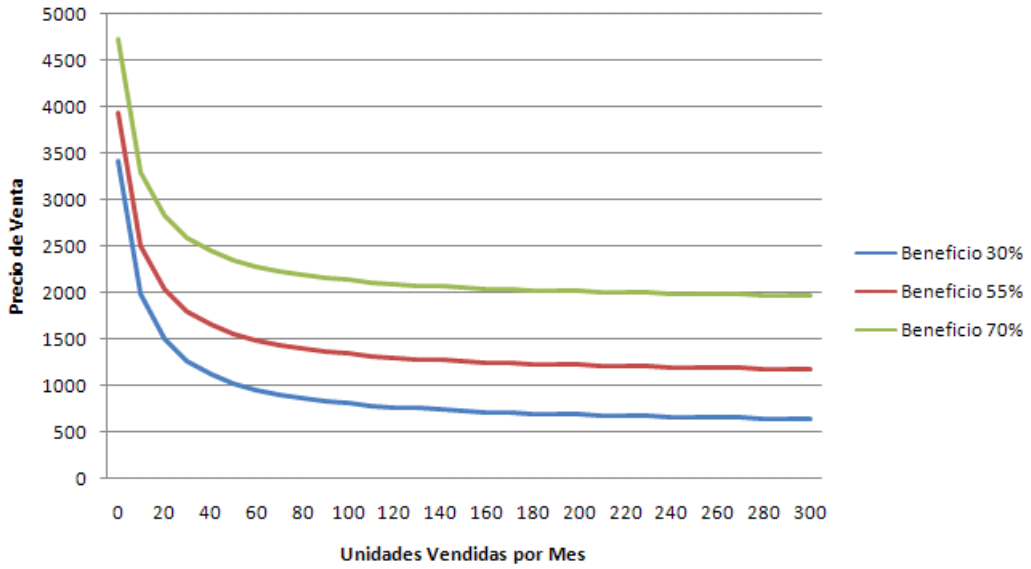

Figura 14: *Estimación de precio de venta*

Aquí obtenemos unas curvas asintóticas con el precio final del producto. Vemos que si venderíamos alrededor de 300 unidades por mes y si quisiéramos obtener un beneficio de 70 % el precio final rondaría los 2000 pesos. Este precio no sirve ya que en promedio a lo largo de los 5 años que dura nuestro proyecto en el mercado vendemos cerca de 70 unidades por mes. Si vemos el gráfico la curva más conveniente para elegir es aquella que presenta un beneficio del 55 % y el precio al vender cerca de 70 unidades es de aproximadamente 1500 pesos. Este fue el valor que tomamos para realizar los cálculos anteriores.

Si bien el precio final por artículo esta impuesto por el mercado, vemos que es un precio considerablemente inferior con respecto a la misma, siendo el caso de Crossbow que vende sus productos a 2500 dólares, un precio casi cuatro veces más alto. De todas formas, el precio final al que llegamos lo consideramos adecuado para insertarnos en el mercado local y ser competitivos desde lo económico.

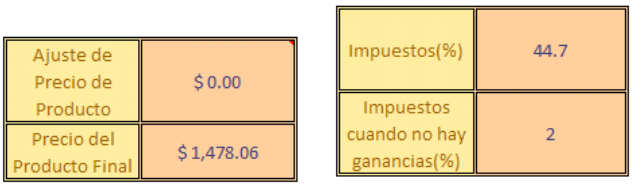

Figura 15: *Precio final e impuestos considerados*

Para los impuestos consideramos un 2 % cuando para el momento en el que no se tiene ganancia, mientras que en los casos donde si obtenemos ganancia los impuestos se incrementan. En cualquier caso se paga un porcentaje del dinero ingresado por ventas menos los costos fijos y variables con los que contamos en el desarrollo.

A continuación se presenta el flujo de caja contemplando todos los gastos y los ingresos.

Descripción	Año 0	Año 1	Año 2	Año 3	Año 4
Estado Inicial	$ -310,028.00	$ 0.00	$ 1,036,990.21	$ 1,973,488.58	$ 2,366,903.42
Productos Vendidos	580	1260	1166	658	436
Entrada por Ventas de Productos	$ 857,276	$ 1,862,358	$ 1,723,420	$ 972,565	$ 644,435
Gastos Fijos	$ -310,028	$ -310,028	$ -310,028	$ -310,028	$ -310,028
Gastos Variables	$ -237,220	$ -515,340	$ -476,894	$ -269,122	$ -178,324
Impuestos	$ -6,201	$ -463,535	$ -418,615	$ -175,856	$ -69,769
Ganancia Real - Sin Beneficio	$ -6,200.56	$ 567,255.02	$ 1,304,805.60	$ 796,708.41	$ 574,665.93
Total	$ 0	$ 1,036,990	$ 1,973,489	$ 2,366,903	$ 2,522,986

Cuadro 12: *Flujo de caja*

Como se puede ver, si bien durante el primer año y los primeros meses del segundo, se tiene un saldo negativo, al finalizar el proyecto se obtiene entonces una ganancia final bruta de $2.366.903. El punto crítico del proyecto se encuentra al año, cuando el saldo alcanza el valor de $0, es decir cuando se recupera la inversión inicial.

Considerando el pago de impuestos (impuesto a las ganancias, ingresos brutos, inflación anual, etc) resulta luego de los 5 años de duración del proyecto, **la ganancia neta total será de $574.665**.

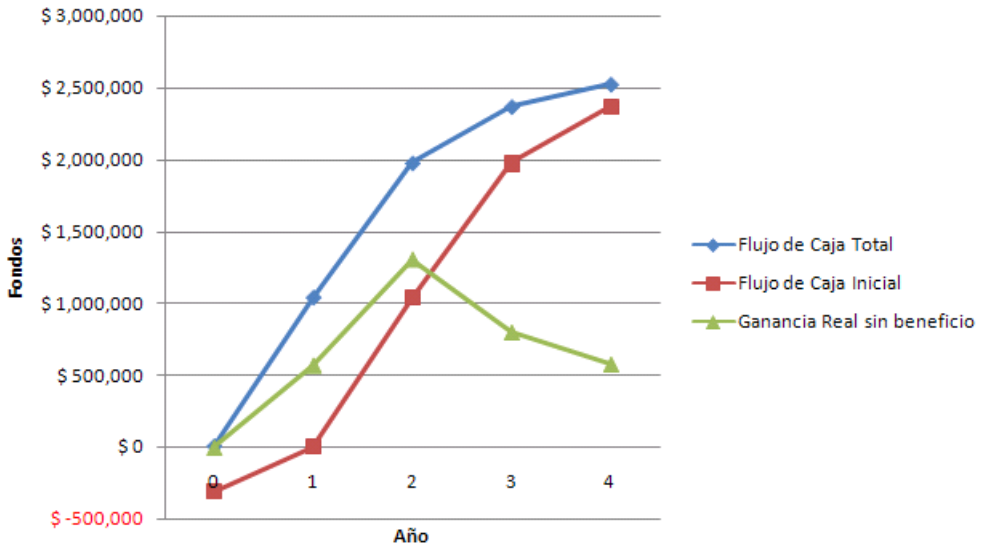

Figura 16: *Flujo de caja*

4.3.4. Ciclo de vida

En lo que respecta a nuestra idea original de implementar una red inalámbrica de sensores, vamos a basarnos en lo que la especificación del protocolo nos indica qué podemos lograr.

En el gráfico del plan de ventas puede apreciarse una primera etapa de lanzamiento del producto, en la cual la cantidad de unidades vendidas aumenta muy poco mes a mes, una segunda etapa de crecimiento, en la cual las unidades vendidas aumentan significativamente mes a mes, una tercera etapa de madurez, en la cual el valor de unidades vendidas es medianamente estable y una última etapa de declinación, en la cual disminuyen apreciablemente las unidades vendidas.

Pasada esta etapa, será necesario introducir un nuevo producto, con mejoras significativas por sobre el producto anterior.

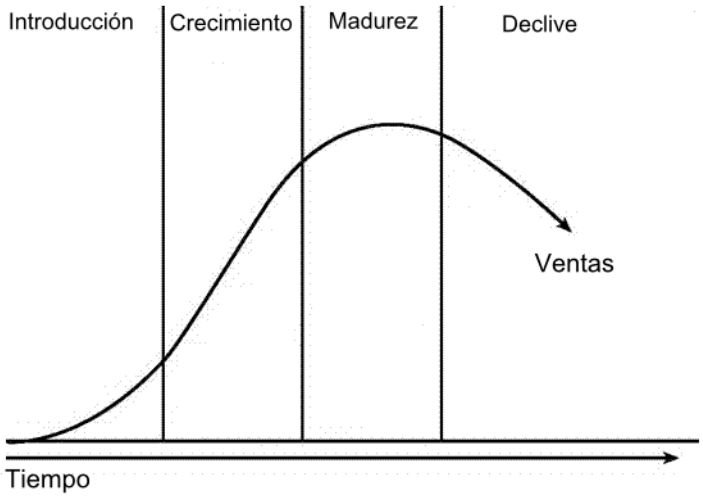

Figura 17: *Ciclo de vida del producto*

En nuestro caso nos estamos enfocando en una necesidad permanente y constantemente renovada. La tecnología que pretendemos utilizar para solventar y cubrir la necesidad que se propuso, es novedosa y poco utilizada en el país, lo que nos hace fijar que el mercado tiene un ciclo de vida extenso.

Dado que los proyectos y planificaciones exigen tiempos más cortos y resultados más inmediatos, vamos a ser excesivamente conservativos en nuestra planificación y estimamos el **ciclo de vida en 5 años**.

4.3.5. VAN y TIR

Se calcula entonces el Valor Actual Neto (VAN) correspondiente al proyecto, que calcula el valor neto presente de una inversión a partir de una tasa de descuento y una serie de pagos futuros (valores negativos) e ingresos (valores positivos).

Dicha expresión se muestra a continuación: considerando una tasa de interés mensual del 1.1 %, basada en la renta provista actualmente por la banca local.

$$VAN = \sum_{t=1}^{n} \frac{V_f t}{(1+k)^t} - I_0 = 498{,}596,61$$

Este número indica la ganancia actualizada del proyecto al concluir los 5 años de vida que se plantearon.

Para analizar si esta es una inversión conveniente desde el punto de vista de un inversionista, se considera el interés que se obtendría por una inversión inicial de $310.028, colocada a plazo fijo anual, con una tasa del 16 % anual, que es el caso del Banco Santander Rio consultado en el mes de marzo de 2012. Para este caso, la ganancia obtenida sería $49.605 si se la deja un año, momento en el cual se espera devolver la inversión. En ese momento nuestro producto entrega una ganancia real de $567.255.

Es decir que invertir en el proyecto otorga una ganancia de casi 11 veces mayor que invertir a plazo fijo considerando únicamente el primer año, ya que luego de ese momento se devuelve la inversión.

Por otro lado, es relevante calcular la tasa interna de retorno (TIR) de la inversión, definida como la tasa de interés con la cual el valor actual neto (VAN) es igual a cero.

En este caso, la tasa interna de retorno obtenida es de 69 %. Es decir que si se consideran los 5 años de duración del proyecto, se tiene una rentabilidad anual del 69 %, casi 4 veces mayor que la rentabilidad otorgada por los bancos.

TIR	69%
VAN	$ 498,596.61
PI	$ 0.62

Figura 18: *Valores del VAN y TIR*

4.3.6. Punto de cobertura

El punto de cobertura representa la cantidad de unidades que necesitamos vender para establecer el momento en el cual se empieza a obtener ganancias. Se trazan las curvas de gastos e ingresos en función de las unidades vendidas en un determinado tiempo.

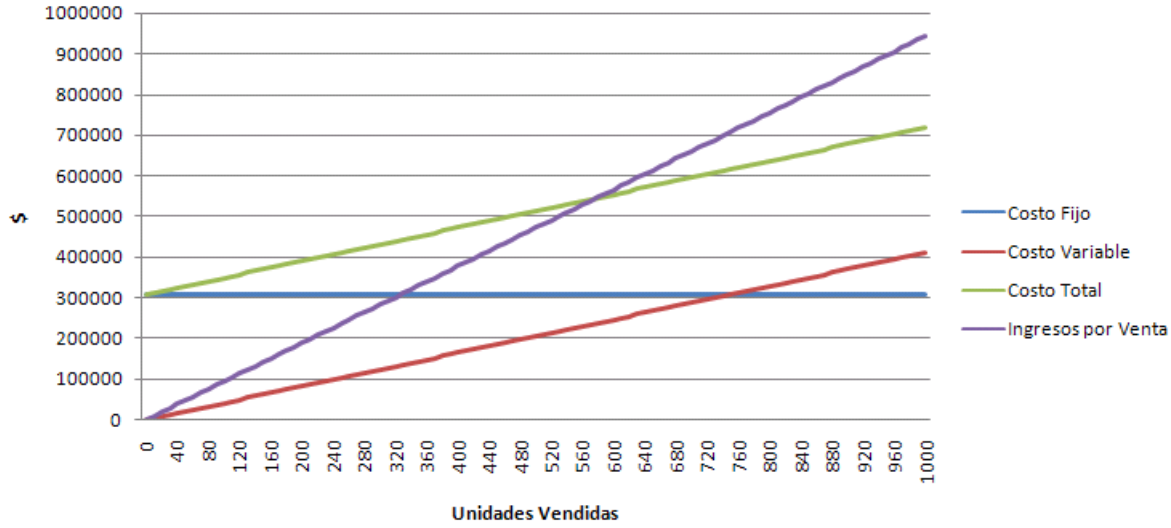

Figura 19: *Punto de cobertura*

Vemos que el gráfico muestra el punto en el cual se cruzan las rectas correspondientes a los gastos totales y los ingresos obtenidos. Luego de vender cerca de 600 unidades corresponde precisamente a este punto. Si analizamos cuando se produce esta cantidad de unidades vendidas nos damos cuenta que es luego del primer año de salir al mercado tal como se mencionó cuando presentamos el flujo de caja. Luego del primer año de salida al mercado es cuando estamos en condiciones de devolver la inversión inicial y pasado ese período de tiempo es cuando empezamos a obtener ganancias.

4.4. Factibilidad legal y responsabilidad civil

4.4.1. Regulaciones

- Normas Nacionales

  IRAM 2001 – Tensiones y frecuencia eléctricas normales.

  IRAM 2076 – Conectores de aparatos eléctricos en general.

  IRAM 2092 – Seguridad de aparatos electrodomésticos y similares.

  IRAM 4200 – Procedimientos básicos de ensayos ambientales para componentes y equipos electrónicos.

  IRAM 4025: Circuitos impresos. Requisitos y métodos de ensayo.

  IRAM 4099: Circuitos impresos de simple y doble faz con orificios metalizados.

  Asme QW-100: Procedimiento para realizar soldaduras.

  Asme QW-200: Especificaciones sobre cómo realizar la soldadura.

  Radiaciones no ionizantes: El Ministerio de la Salud y Medio Ambiente, a través de la Resolución 202/95 ha explicitado los valores tolerables de las radiaciones no ionizantes, y la Secretaría de Comunicaciones, en la Resolución 530/2000 establece que será de aplicación obligatoria a todos los sistemas de telecomunicaciones irradiantes.

  Seguridad eléctrica: En el año 1998, la Secretaría de la Competencia, la Desregulación y la Defensa del Consumidor resuelve establecer un control sobre todo producto eléctrico que se comercializa en la Argentina.

Una vez que se fabrica el equipo siguiendo las pautas que figuran en las normas, habrá que presentarlo frente al IRAM o alguna otra autoridad competente para certificar el cumplimiento de las mismas.

- Normas Internacionales

  ISO 9001 - Certificado internacional de calidad

  ISO 14001 - Certificado de gestión ambiental

Dentro de esta serie de normas las que son de importante cumplimiento para este proyecto son la **ISO/ IEC 18000-1** y la **ISO/ IEC 18000-41**.

Una vez que se fabrica el equipo siguiendo las pautas que figuran en las normas, habrá que presentarlo frente al IRAM o alguna otra autoridad competente para certificar el cumplimiento de las mismas.

4.4.2. Estándares de cumplimiento

Al tratarse de comunicaciones inalámbricas de baja velocidad, el capítulo de la IEEE que se toma como referencia es el 802.15.4 y es sobre la cual ZigBee se establece. Específicamente la publicación del año 2006, *"IEEE Standard for Information metropolitan area networks. Specific requirements"* [1].

Los puntos más importantes de la publicación que son relevantes para nuestro proyecto son los siguientes:

- Componentes de la IEEE 802.15.4 WPAN
- Topologías de red
- Formación de red tipo peer-to-peer (P2P)
- Arquitectura: la capa física (PHY)
- Subcapa MAC

- La estructura de la super trama
- Modelo de transferencia de datos
- Transferencia de datos a un coordinador
- Transferencia de datos desde un coordinador
- Transferencias P2P
- Estructura de trama
- Trama Beacon
- Trama de datos
- Trama de confirmación
- Trama de comando MAC
- Mejorando la probabilidad del transporte exitoso
- Mecanismo CSMA-CA
- Verificación de datos
- Consideración de consumo de potencial
- Seguridad

4.4.3. Licencias

- Frecuencia

Las bandas de frecuencia en las que trabajará el módulo ZigBit entra en la categoría mundialmente denominada ISM (Industrial, Scientific and Medical) las cuales tienen como fin aplicaciones industriales, científicas o médicas, no requiriendo ninguna autorización de la CNC para su uso por ser libres. De todas maneras es obligatorio que el módulo adquirido cuente con la correspondiente homologación de la CNC1.

La banda IMS al ser no licenciada es ampliamente utilizada por una serie de dispositivos como teléfonos inalámbricos, puertas de garaje automáticas, sensores remotos, etc. Es por esto que la ITU recomienda y las autoridades reguladoras como la FCC (EE.UU.) o la CNC exigen que los productos se desarrollen dentro de algún esquema que permita controlar las interferencias ya que pueden generar serios inconvenientes sobre otros dispositivos mas críticos como los de las redes de comunicación.

Para evitar entonces estas interferencias aparecen diversas técnicas de modulación. Las técnicas tradicionales maximizan la potencia en el centro de la frecuencia asignada para solventar el problema del ruido, pero resulta fácil su detección e interceptación. Por otro lado aparecen las técnicas de espectro ensanchado como ser DSSS (Secuencia directa de espectro ensanchado) y FHSS (Saltos en frecuencia en espectro ensanchado) que hacen uso de un mayor ancho de banda y son altamente más eficaces.

Esta tecnología se ha impuesto frente a otras, por su excelencia y por sus mejoras en cuanto a complejidad y costes. En los últimos años la ISO ha desarrollado en ese sentido una serie de estándares denominados ISO/ IEC 180002.

ISO 18000- Parte 1	Parámetros fundamentales para las frecuencias aceptadas mundialmente
ISO 18000- Parte 2	Parámetros para las comunicaciones aéreas de interfaz debajo de 135 kHz
ISO 18000- Parte 3	Parámetros para las comunicaciones aéreas de interfaz a 13,56 MHz
ISO 18000- Parte 4	Parámetros para las comunicaciones aéreas de interfaz a 2,45 GHz
ISO 18000- Parte 5	Parámetros para las comunicaciones aéreas de interfaz a 5,8 GHz (Retirada)
ISO 18000- Parte 6	Parámetros para las comunicaciones aéreas de interfaz en 860 a 960 MHz
ISO 18000- Parte 7	Parámetros para las comunicaciones aéreas de interfaz a 433 MHz

4.4.4. Entes certificantes y homologación

La CNC cuenta entre sus funciones principales la de homologar, tanto equipos que hagan uso del espectro radioeléctrico, como así también los de uso específico en telecomunicaciones que se conecten a las redes públicas.

El organismo cumple con las siguientes actividades vinculadas no solo al registro de equipos sino también al de las empresas que los fabrican y/o los comercializan en el país.

- Homologación, codificación y autorización de equipos (Inscripción en el Registro de Materiales RAMATEL).

  La CNC es el organismo encargado de la normalización del equipamiento de comunicaciones de la República Argentina.

  Esta normalización se realiza mediante el dictado de normas técnicas basadas en:

  Seguridad del usuarios

  Uso eficiente del espectro radioeléctrico

  Asegurar la compatibilidad con las redes y sistemas de comunicaciones autorizados

4.4.5. Patentes

Las patentes existentes más significativas son las siguientes:

12/992,016 - Red Inalámbrica de Sensores [11] Del año 2009, en la cual se patenta un algoritmo para coordinar los mensajes en la red. Se basa en nodos que recolectan información y lo van enviando de forma periódica a gateways de la red. No habla sobre una red totalmente distribuida.

10/903,652 - Redes Distribuidas de Sensores [13] Del año 2004, dónde se patenta un algoritmo para el almacenamiento de datos en las redes. En el mismo se hablan de dos formas de resolver el problema, una en la que los nodos transmiten entre sí la información cada una cierta cantidad de tiempo y otra en la que se habla de un dispositivo agregado con el único fin de almacenar la información.

Todas las patentes encontradas se enfocan en algoritmos para el mejor funcionamiento de la red, por lo que no afectan de forma directa al desarrollo del proyecto. Podrían llegar a afectar, según el método que se utilice para generar y administrar la red, pero eso se verá durante el estudio de la solución.

5. Ingeniería de detalle

En esta sección se detallará cada parte que compone el proyecto, tanto lo referido al hardware como al firmware y al software para ser utilizado en la PC.

5.1. Hardware

Explicaremos el hardware mediante diagramas en bloque y daremos una explicación detallada de cada elemento que conforma cada bloque.

5.1.1. Diagrama de bloques

A continuación se muestra el diagrama en bloque para los dos elementos que componen el proyecto. El coordinador que conecta con cada nodo y recibe los datos de sus sensores y el ya mencionado nodo con sus sensores.

- Diagrama en bloque del Coordinador

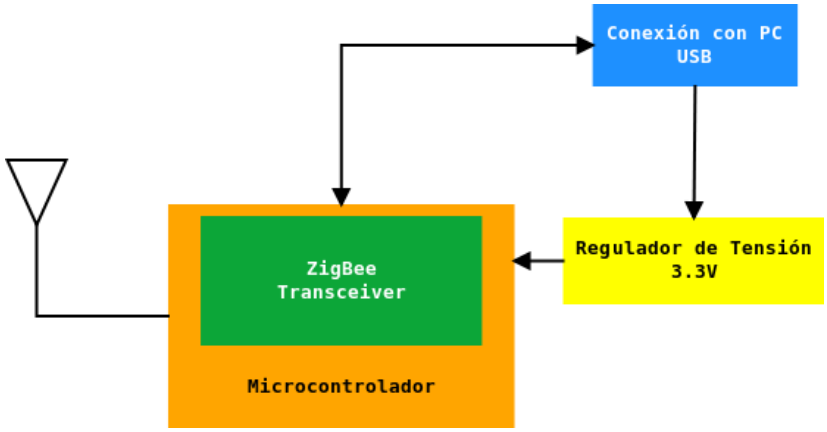

Figura 20: *Diagrama en bloque - Coordinador*

- Diagrama en bloque del Nodo

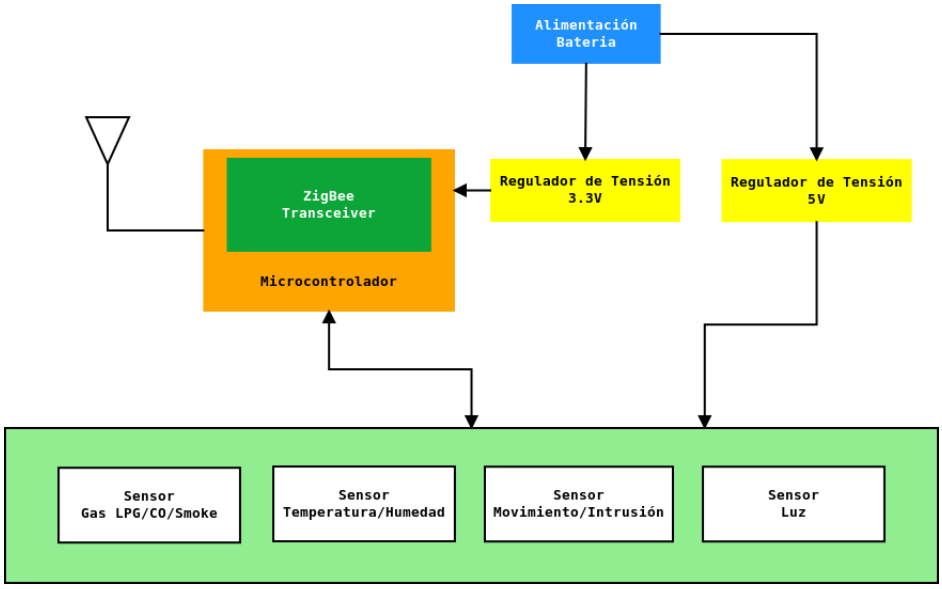

Figura 21: *Diagrama en bloque - Nodo*

5.1.2. Descripción detallada de cada bloque

- Coordinador

El bloque del coordinador se compone de un módulo que integra el microcontrolador y el transceiver de ZigBee, el mismo fue mencionado como la mejor alternativa en la sección 4.1.2, la cual fue el módulo ZigBit ATZB-24-B0 de Atmel. Luego la alimentación se resuelve mediante la conexión usb de la PC ya que va a estar siempre conectado. Se utiliza el regulador de tensión de 3.3 V, el LM2937 para alimentar al módulo.

- Nodo

El bloque del nodo es muy similar al coordinador. La principal diferencia se encuentra en la alimentación, en este caso se utiliza una batería de 9V y para alimentar al módulo ZigBit ATZB-24-B0 de Atmel se utiliza un regulador de tensión de 3.3V, el LM2937 y para alimentar a los sensores se utiliza el LM7805 que entrega una tensón de 5 V.

Luego este bloque presenta la conexión con sensores, se utilizan 4 tipos de sensores diferentes que miden distintos parámetros ambientales: temperatura, humedad relativa, monóxido de carbono, humo, petróleo, luz y movimiento.

A continuación se resume una tabla con cada componente de cada bloque.

Componente	**Valor**
Módulo	ATZB-24-B0
Conector USB	CP2102
Conexión Serie	MAX3232
Conector	DB-9
Conector	USB Female Type B
Regulador de Tensión	LM2937
Regulador de Tensión	LM7805
Resistor	200Ω
Resistor	200Ω
Capacitor	0,33μF
Capacitor	0,1μF
Capacitor	0,33μF
Capacitor	0,33μF
Capacitor	0,1μF
Capacitor	0,1μF
Capacitor	0,1μF
Capacitor	0,1μF
Capacitor	10μF
Led	Rojo
Led	Verde

Componente	**Valor**
Módulo	ATZB-24-B0
Regulador de Tensión	LM2937
Regulador de Tensión	LM7805
Conector	Bateria
Resistor	200Ω
Capacitor	1μF
Capacitor	0,1μF
Capacitor	0,1μF
Capacitor	10μF
Led	Verde
Sensor	DTH11
Sensor	MQ-2
Sensor	LDR
Sensor	P-381
Resistor	220$K\Omega$
Capacitor	100nF
Resistor	5,6$K\Omega$

Cuadro 13: *Materiales Módulo Nodo - Coordinador*

5.1.3. Plan de pruebas de cada modulo

Siempre que pongamos un producto en producción vamos a tener que pasar por un proceso de pruebas mediante el cual vamos a probar cada bloque de cada módulo, sabiendo que es lo que nos interesa obtener de cada uno. Tenemos que tener en consideración que los módulos se los tiene que probar una vez ensamblados. Es por esto que las pruebas que interrumpan caminos de señal en los módulos no van a ser posibles.

Otras pruebas serán indirectas, no se medirá concretamente el parámetro sino la consecuencia de que este parámetro se encuentre dentro de los valores aceptables, por ejemplo no vamos a medir la corriente que proporciona una fuente, sino la estabilidad de la tensión cuando el modulo que alimenta está consumiendo el máximo.

- **Módulo ATZB-24-B0 - Alimentación**

Para la medición de la alimentación vamos a tener que medir la tensión que entrega a los módulos. Para ello vamos a utilizar como instrumento un **multímetro True RMS UT-60e**.

La tensión se mide a la salida de la fuente. La corriente la asumiremos correcta si podemos aumentar el consumo de los módulos alimentados cuando están transmitiendo y la tensión se mantenga estable.

A priori no interesa cual de los bloques no funciona correctamente, ya que el tiempo que nos tomaría detectar la falla en alguno de los ellos representaría un costo mayor que reemplazar el módulo completo.

Aclarado esto, supongamos que las fallas en este módulo son recurrentes, y necesitamos puntualizar algún tipo de causa raíz que está afectando el normal funcionamiento del módulo. Vamos a necesitar aislar la causa de falla entre todas las unidades lógicas. Una falla del microcontrolador a nivel del procesador, impediría directamente que podamos cargarle el programa, de manera que el simple hecho de poder cargar el programa en el microprocesador es un buen signo.

- **Módulo ATZB-24-B0 - MCU**

Luego pasamos a probar las salidas del microcontrolador. Para ello utilizaremos como instrumento un **Osciloscopio Tektronix TDS 220 (100Mhz 1GS/s)**.

Con el osciloscopio podemos posicionarnos en los puertos de salida del microcontrolador y ver si estos varían de nivel al intercambiar datos. Verificado esto podemos asegurar el correcto funcionamiento del módulo.

- **Módulo ATZB-24-B0 - Stack ZigBee**

Comprobamos su comportamiento si en el intercambio de datos con el microcontrolador no existen errores, podemos en principio descartar un problema con el este bloque.

Por otro lado, la forma de verificar los datos que genera el bloque stack es mediante un sniffer. Podemos utilizar como instrumento **WiSens 802.15.4/ZigBee Packet Sniffer**.

Este instrumento es para poder tomar los paquetes de datos en el aire y poder analizarlo por separado. Junto con un analizador de espectro podemos aislar alguna falla en este bloque

- **Módulo ATZB-24-B0 - Transceiver**

Finalmente el modulo de RF es el encargado de enviar y recibir los datos al aire. Para determinar su funcionalidad en RX, simplemente tenemos que verificar que los datos sean interpretados por el Stack de ZigBee. En TX lo podemos verificar con un analizador de espectro de bajo costo. El instrumento que podemos utilizar es **AirView2Ext - 2.4GHz Spectrum Analyzer w/ext. Antenna**.

Con este instrumento vamos a poder ver si en el caso de no detectarse paquetes en el sniffer, si es porque no hay transmisión del modulo porque no hay señal de RF detectada en el aire o si efectivamente el stack de ZigBee no está enviando los datos.

- **Conexión con PC - MAX3232**

Dado el estándar, simplicidad, el nivel de pruebas y años en el mercado que tiene este tipo de interfaz, siempre que falle la transferencia de datos desde la PC, vamos a apuntar a reemplazar el integrado de interfaz. Si queremos un mayor nivel de detalle en la transferencia de datos, utilizaremos un **Osciloscopio Tektronix TDS 220 (100Mhz 1GS/s)**.

- **Sensores - Alimentación**

La tensión de alimentación puede ser comprobada a la entrada del modulo sensor. El consumo se mide indirectamente al medir que la tensión este constante con el sensor activado. Utilizaremos un **Multímetro True RMS UT-60e**.

- **Sensores - Medición de Magnitud física**

Para comprobar que las mediciones de las distintas magnitudes físicas (temperatura, humedad, gas, movimiento, luz) funcionan correctamente utilizaremos el valor de un parámetro conocido y utilizando la PC veremos que mide correctamente. En paralelo podemos contrastar el valor de tensión que entregan utilizando un **Multímetro True RMS UT-60e**.

Por ejemplo en el caso del sensor de temperatura le acercaremos un cubo de hielo y si la medición ronda los 0°C lo consideramos correcto.

En el caso de la humedad lo comprobamos mediante la información del servicio meteorológico nacional y que el resultado sea similar. Como prueba adicional se puede soplar en las cercanías del sensor y ver que la medición aumenta de valor.

En el caso del sensor de luz esperamos medir luz plena en el exterior en un día soleado.

Para el sensor de gas las pruebas son ligeramente más complicadas, pero medimos mediante alguna referencia patrón, puede ser un cigarrillo o el gas natural de un tubo.

Para el sensor de movimiento es sencillo, esperamos medir un 1 lógico cuando se detecta un movimiento y un 0 lógico en otro caso.

5.2. Software - Firmware

La primera parte del diseño del software consta en el desarrollo del firmware del circuito. El mismo se basa en el protocolo de comunicación inalámbrica Zigbee. En nuestro caso, este protocolo es proporcionado por la compañía Atmel [5] , bajo el nombre de BitCloud [6]. Esta librería (que desde ahora se le llamará "stack Zigbee") proporciona todas las funcionalidades del protocolo Zigbee y además proporciona facilidades para el uso del microcontrolador utilizado. En la figura 22 se puede ver cómo está formado.

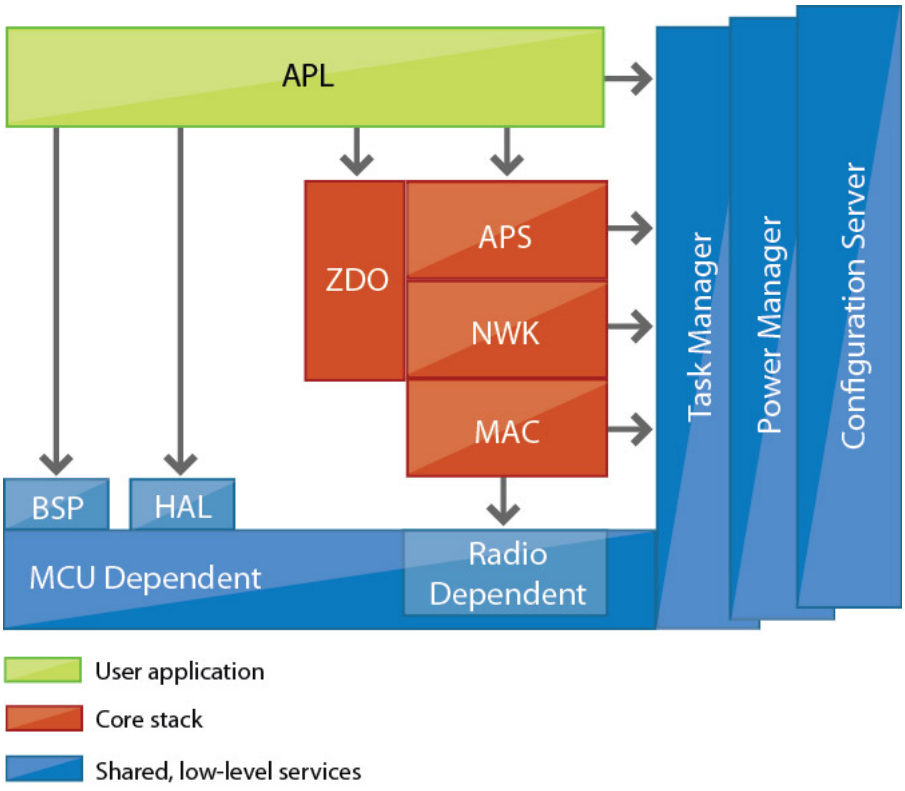

Figura 22: *BitCloud - Stack de Zigbee*

En la imagen se puede ver varios sectores:

Color Verde Este sector está conformado por toda la aplicación del usuario, es decir, el código propio del usuario que se crea a partir del stack de Zigbee. Este utiliza toda las funcionalidad del protocolo, pero no se preocupa por cómo están implementadas, es decir se produce un ocultamiento del cómo y se deja al usuario el "qué".

Color Azul Este sector está formado por el código específico de cada micro, por lo que siempre es distinto y presenta una interfaz a la capa superior que homogeneiza la utilización de los recursos del micro y de los dispositivos conectados al mismo. Esta capa es utilizada por todo el resto del stack.

Color Rojo Representa la parte principal del stack, la cual se encarga de que el mismo funcione como un dispositivo que sigue las reglas del Zigbee. Este sector proporciona una interfaz con todas las funciones para utilizar el Zigbee en forma integra.

Además, en el protocolo existe una categoría de los nodos en la red, la cual está determinada por:

- **EndDevice (Dispositivos Finales)**: Estos son los últimos elementos de la red, los cuales solo pueden buscar una red y unirse a esta. No tienen capacidad de generar una red ellos mismos o transportar mensajes de otros nodos dentro de la red. Estos dispositivos están encargados con la interacción con el mundo, es decir, contienen sensores, actuadores, etc. que les permiten cuantificar sus alrededores y realizar una operación adecuada para la situación. Esto se puede ver con el ejemplo de un sensor que mida el nivel de agua. Este además de medir le nivel, debe tener algún tipo de actuador que le permita, por ejemplo, cerrar la llave de agua para que no rebalse el tanque. Además de todo esto, los EndDevices deben dormir cada vez que se comunican, ya que en general llevan algún tipo de batería, la cual duraría demasiado poco en el caso de tener que mantener la comunicación inalámbrica todo el tiempo.
- **Router**: Estos dispositivos son EndDevices que además pueden transportar mensajes de otros nodos dentro de la red, de forma de encontrar caminos en la red. Además, al igual de los EndDevice, no pueden generar redes, ni encontrar el mejor camino para mover los mensajes, solo actúan conforme a tablas internas.
- **Coordinador**: Este dispositivo es el encargado de armar la red y armar los caminos para que todos los EndDevices estén comunicados con el centro de la red, es decir, el coordinador mismo. Este dispositivo nunca puede entrar en modo “sleep”, ya que tiene que estar en todo momento escaneando la red para leer pedidos o ver cuando se despierta el resto de los dispositivos para mandarles mensajes. Además, solo puede haber un coordinador por red.

5.2.1. Diagrama de estados, procesos y flujogramas

El funcionamiento del firmware, se divide en dos partes, la parte para el EndDevice y una para el Coordinador.

- **EndDevice**

El funcionamiento básico del EndDevice se puede ver en la figura 23.

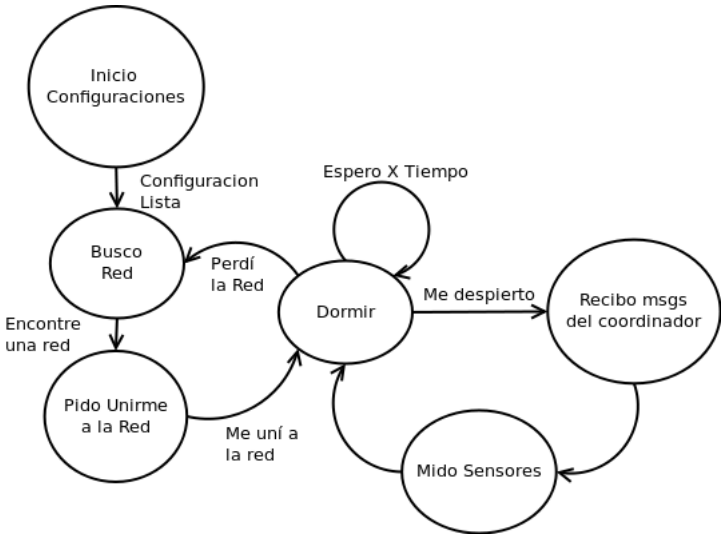

Figura 23: *Diagrama de Estado de los EndDevice*

En el mismo se puede ver como inicialmente el dispositivo configura los parámetros iniciales del protocolo y del código propio del sistema. Una vez que tiene todo configurado, trata de buscar la red a la que debe unirse.

Cada EndDevice sabe a que red debe unirse desde un principio, por lo que varias redes pueden estar juntar sin tener el peligro de que un EndDevice se una a una red equivocada. Al unirse a la red, el Coordinador le asigna un identificador (en adelante, "ids") al azar. Esto permite que la red pueda crecer, sin tener que preocuparse por ids repetidos.

Una vez que se está en la red, se verifica que no existan mensajes para el nodo, en cuyo caso, se procesan primero antes que nada y luego se continúa a medir los sensores. Al terminar de medir los sensores, se trata de buscar al coordinador para enviar las mediciones, de no encontrarlo lo volverá a intentar más tarde. Notar que cada vez que se realicen las mediciones, el EndDevice pasará a un estado de dormir, en el cual no recibirá ni enviará mensajes por la red inalámbrica. Esto se debe a que el módulo de comunicación consume mucha energía para su funcionamiento, por lo que al desactivarlo genera un ahorro importante de las baterías del dispositivo. La comunicación desde el Nodo a la PC se puede ver en la figura 26 y el proceso de re intento de envío al Coordinador se puede ver en la figura 27.

- **Coordinador**

El funcionamiento básico del Coordinador se puede ver en la figura 24.

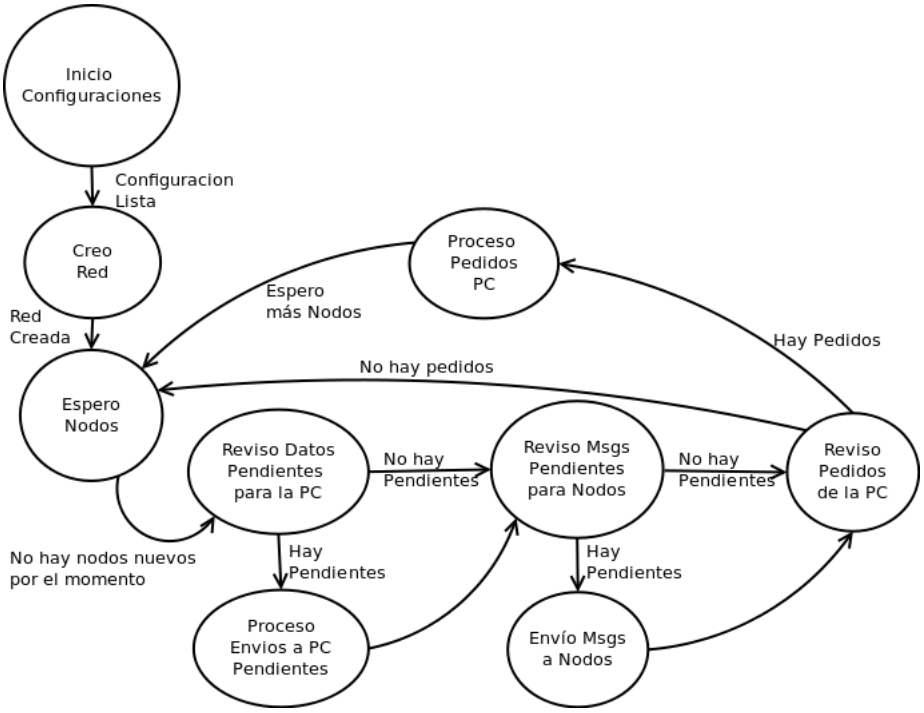

Figura 24: *Diagrama de Estados del Coordinador*

El Coordinador, al igual que el EndDevice, empieza por establecer las configuraciones iniciales de la red y el sistema. Una vez hecho esto, procede a generar la red. Esta red tiene un id establecido al momento de la creación del Firmware por lo que no puede ser cambiado luego de programada. Esto se diseñó de tal forma para evitar errores de repetición de redes en la misma zona.

Una vez que la red esté creada, procede a buscar pedidos para unirse a la red, una vez resueltos los pedidos, busca mensajes de los elementos de la red para el coordinador (y/o PC) y los procesa. También hace los mismo con los mensajes pendientes a nodos de la red, para luego, finalizar revisando mensajes de la PC hacia la red.

Un detalle importante a tener en cuenta en la comunicación es que siempre se trata de ver si está el nodo destino en la red, si no está, se lo considera dormido y la próxima vez que ese nodo se despierte y avise al coordinador, este le manda los mensajes pendientes.

La figura 25 muestra el camino de interacciones desde que la PC manda un mensaje hasta que llega al nodo adecuado.

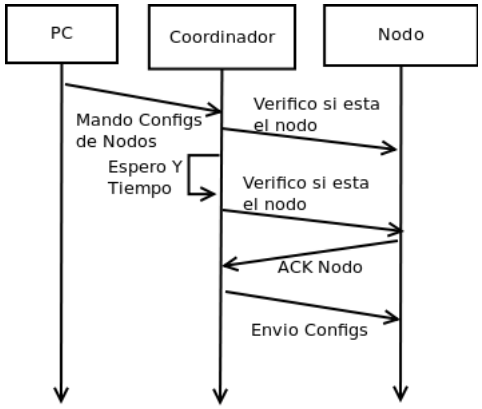

Figura 25: *Comunicación desde la PC a la red*

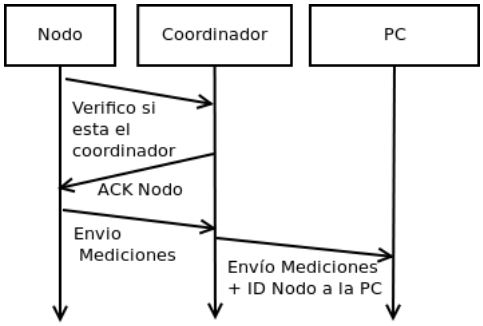

Figura 26: *Comunicación Exitosa con la PC*

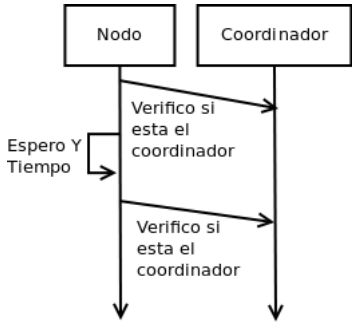

Figura 27: *Condición de Error en la comunicación con el Coordinador*

5.3. Complejidad

Debido a que se utiliza código propietario para el protocolo ZigBee, es difícil calcular con exactitud la complejidad del código, ya que la empresa no da ningún tipo de información sobre esto. Por lo tanto esta sección, solo incluirá la complejidad del código desarrollado por nosotros, que se ubica por encima del código de ZigBee.

Nuestro código del firmware tiene una complejidad que es diferente para cada parte del mismo, aunque todas dependen de la cantidad de sensores. Casa una de las mediciones se realizan una cantidad fija de veces, por lo que cada medición tiene una complejidad de O(s), con s la cantidad de sensores. Luego, para la transmisión de datos al coordinador, es una tarea de complejidad fija, ya que sin importar la cantidad de sensores, se envía un paquete con todos los datos (esto puede cambiar, en la situación que se tengan demasiado sensores, pero debido a que nuestro diseño tiene un máximo establecido, esto nunca podrá ocurrir). Por esto, la complejidad del nodo sería O(s + F), siendo s la cantidad de sensores y F una cantidad fija de operaciones para el envío (si bien se puede calcular este valor para nuestro código, este valor pierde sentido si no se tiene una idea del rango de la complejidad del código del ZigBee, por lo que no será calculado). Obteniéndose finalmente, una complejidad algorítmica de O(s).

Luego, en el caso del coordinador, las operaciones que trabaja dependen de la cantidad de nodos de la red, ya que tiene que procesar el envío de cada uno de estos por separado. Por lo que la complejidad sería de O(n), siendo n la cantidad de nodos de la red. Por cada nodo, el coordinador realiza una cantidad establecida de operaciones, las cuales son hechas 1 vez por nodo. Por todo esto, se puede ver que la complejidad algorítmica del coordinador es de O(n), con n la cantidad de nodos de la red.

5.3.1. Plan de prueba de módulos y de depuración

Para las pruebas se trabajó de la siguiente manera:

1. Primero se evaluaron los módulos autosuficientes, es decir los que interactúan de forma directa con el stack y no interactúan con otra parte del código del sistema en sí. Como por ejemplo, la comunicación serie, la conversión serie-usb y el código para leer los sensores. Una vez que cada uno de estos funcionó por separado, se empezó a probar bloques compuestos.

2. Luego se probó el comunicación Zigbee. Para esto se estableció una secuencia de encendido de leds que indicaban en que estado se encontraba el dispositivo (parpadeaba un led si se estaba conectando y se apagaba si se terminó de conectar) y según que se transmitiera se encendían leds distintos de los nodos. Luego, como prueba final de la red, se procedió a enviar un mensaje desde los nodos hacia la PC a través del puerto serie, ya que el mismo fue probado en las primera etapa.

3. Una vez que el punto anterior estuvo libre de errores, se procedió a realizar mediciones y enviar los valores a través del protocolo Zigbee y luego a la PC. Una vez que todos los sensores fueron probados, se dio por terminada esta etapa de pruebas

4. Una vez que la transmisión de los sensores funcionaba, se colocó el último módulo del firmware que ordenaba las mediciones en un formato tal que el programa de PC pueda interpretarlo. Para luego conectar los módulos y la PC todos juntos y verificar que los valores medidos se vean en la PC de forma correcta. (Este último paso se mezcla con el último paso de verificación de la aplicación)

5.4. Software - PC

La aplicación de PC permite la visualización de los datos de la red, además de permitir la configuración de la misma. En la figura 28 se puede ver un diagrama en bloques de la aplicación.

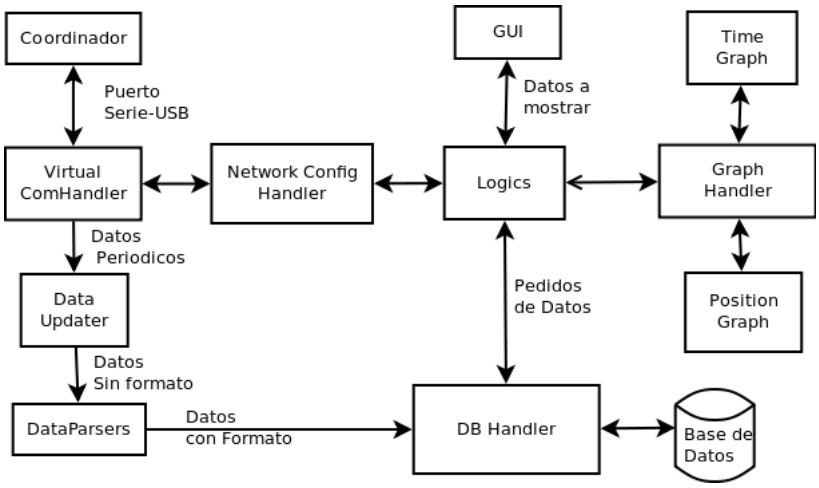

Figura 28: *Diagrama en Bloques General de la Aplicación*

La forma de diseño del programa se basa en un sistema concurrente, es decir, la aplicación de PC no es un solo programa corriendo, sino que son varios interactuando entre sí. Este tipo de diseño permite programas mucho más moldeables, ya que cada bloque del mismo se diseña con un proceso distinto, y se establecen las interacciones entre procesos, por lo que se deja a cada programador de cada módulo que trabaje como quiera dentro de su proceso, mientras que respete los mensajes entre los programas. Además, si se necesitará agregar un proceso más o cambiar una forma de procesamiento de datos, solo una parte del programa se ve afectada, es decir, el bloque a modificar. Y lo que es aún más importante, permite que partes del programa sean compiladas y entregadas cerradas, mientras que otras pueden ser de código abierto para permitir mejoras en el sistema de forma transparente al resto de los módulos.

Con esto, se pueden hacer cientos de mejoras en el sistema, por ejemplo, se podría dejar la lógica de la aplicación en un servidor y tener interfaces en sistemas remotos para adquirir esos datos de forma simple y segura, si se coloca un proceso entre medio de la "lógica" y la interfaz de usuario que envíe los datos a través de internet.

Un detalle importante es que las comunicaciones entre los procesos, se hacen a través de una cola de mensajes del sistema operativo, que es encapsulada por la librería de Boost [7], para poder generar código mas portable.

Y finalmente, vale destacar que el programa fue desarrollado en C++. La razón para esto es que inicialmente se iba a utilizar Java por ser multiplataforma, pero debido a que este lenguaje no es bueno para el manejo de periféricos de la PC o código a nivel de bits, se optó por C++ que presenta las capacidades necesarias para nuestro trabajo.

5.4.1. Diseño de la Aplicación

El programa está dividido en 5 partes principales:

1. Adquisición de Datos

Esta parte está formada por el VirtualComHandler, DataUpdater, DataParsers y DataForwarder.

- **VirtualComHandler** Este bloque representa la clase con el mismo nombre. Su función es realizar las comunicaciones con el puerto serie-usb. Debido a que se utiliza un puerto usb, las comunicaciones deben hacerse de forma half-duplex, por lo que no se puede recibir y transmitir al mismo tiempo. Esta clase funciona como un handler del proceso que está detrás, encargado de las comunicaciones.

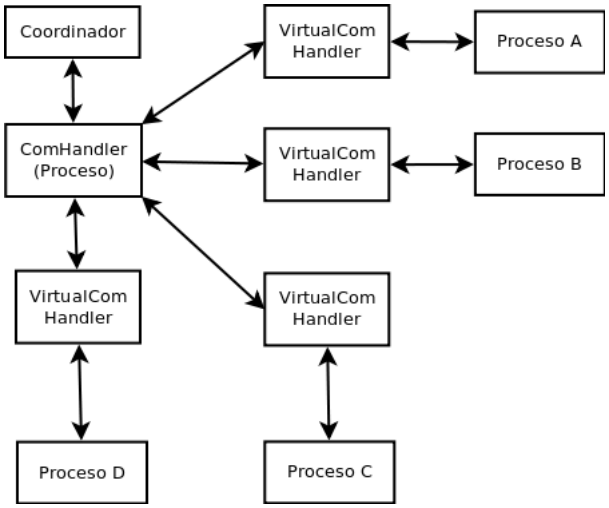

Figura 29: *Diagrama del VirtualComHandler*

En decir, cada proceso que desee hablar con el puerto serie-USB deberá crear una instancia del handler del mismo y utilizar este handler para hacer pedidos al proceso de comunicaciones. Esto se diseño de esta forma ya que permite tener muchos procesos que hablen con el VirtualComHandler de forma ordenada. Además el simple hecho que las comunicaciones se hagan en un proceso a parte, permite que el programa principal pueda seguir operando, sin tener que esperar a las comunicaciones.

Los métodos básicos provistos por el handler son:

- createHandler(const char* COMProcess, std::size_t processTypeID, serialConfig serialConfigData) - El process ID(identificador del proceso) es para poder enviar mensajes de respuesta al proceso correcto
- readData(char buffer, std::size_t qtyRead)
- writeData(char* buffer, std::size_t qtyToWrite)

Luego las estructuras necesarias serán:

- serialConfig: Con los datos del puerto serie (baudrate, flow Control, etc)
- comMsg: mensaje de la cola para las comunicaciones, deberá contener el dato en un char*, es decir sin formato y el id para indicar que debe ser formateado.

El protocolo interno del handler no es relevante, ya que procesos externos solo utilizan los handlers provistos.

- **DataUpdater** Este bloque representa el proceso con el mismo nombre, cuyo objetivo es periódicamente leer si existen nuevos datos provistos por la red. Una vez que tiene datos nuevos, los envía al DataParser para que los formatee y sean guardados.

La estructura tiene la siguiente forma:

```
#pragma pack(push)
#pragma pack(1)
typedef struct {
    short int nodeID;
    short int humidityValue;
    short int temperatureValue;
    char movementValue;
    char lightValue;
    char gasValue;
} sensorMsg;
#pragma pack(pop)
```

Luego se completa otra estructura con la cual trabajarán los parsers (procesos que convierten las tramas de bits en valores leíbles por humanos), esta tiene la forma:

```
#pragma pack(push)  // Alineo a 1 byte cosa de que los campos estén seguidos
#pragma pack(1)
typedef struct {
    int sensorType;
    int nodeID;
    time_t date;
    float value;
} dataDBSlot;
#pragma pack(pop)
```

La cual es la misma que se usa para insertar estos valores a la base de datos.

- **DataParsers** Los DataParsers tiene como objetivo convertir las tramas enviadas por el Coordinador en valores entendibles por personas. Para esto, cada uno de los parsers, escucha de una cola de mensajes, por datos, al recibir uno lo procesa según sea necesario (cada parser escucha solo un tipo de dato) y lo envía al DataForwarder.

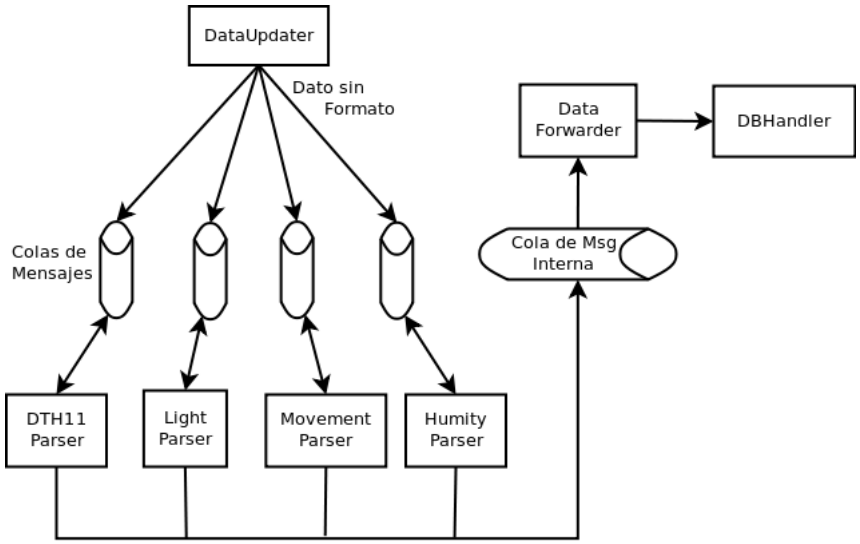

Figura 30: *Diagrama de los DataParsers*

- **DataForwarder** Esta última parte esta encargada de recibir mensajes de los parsers y almacenarlos en la base de datos. La razón de que esté este proceso es para separar las etapas de funcionamiento y permitir una mejor escalabilidad del código, en el caso que se desea insertar un nuevo parser, ya que no hay que agregarle información de la base de datos, solo tiene que saber que tiene que enviar el dato procesado al Forwarder y este se encarga del resto.

2. Configuración de la Red

Esta parte está formada por el Network Config Handler, el cual simplemente tiene como objetivo recibir pedidos lógicos de configuración de la red y convertirlos a tramas de bits que el coordinador entienda. Este bloque es una clase, no un proceso, para facilitar el desarrollo y no agregar complejidad dónde no es necesaria.

3. Persistencia de Datos

Esta parte está formada por el DBHandler y sus sub-Procesos.

Este bloque está compuesto por tres partes:

- Datos de los sensores formateados, es decir, en números, no como una secuencia de bytes.
- Datos de configuración (alarmas, sensores activados, configuraciones varias del programa en sí)
- Textos en diferentes idiomas.

En todos los casos, los procesos escuchan de una única cola de mensajes y retiran los mensajes que sean para ellos. Para luego guardarlos en una estructura interna(que como es ocultada al exterior no importa cómo halla sido implementada, aunque se deben tener en cuenta razones de performance según el tipo de dato a guardar y la cantidad del mismo).

Además, como es de suma importancia que los datos persistan, es decir, se pueda contar con una copia en disco en todo momento.

Base de datos de sensores[1]

Formato de mensajes con el que trabaja:

```
char requestType;
int sensorType;
int nodeID;
time_t dateFrom;
time_t dateTO;
float value1;
float value2;
```

Luego en requestType se configura en que acción se está realizando, es decir:

0. Agregar Dato
1. Pedir último Dato de un Sensor
2. Borrar lista de un nodo y un tipo de sensor
3. Borrar Todas las Listas de un sensor
4. Pedir todos los datos de un sensor
5. Pedir todos los datos de un sensor entre las fechas (date)
6. Terminar Proceso
7. Pedir todos los datos de un sensor de todos los nodos
8. Pedir el ultimo valor de un sensor de todos

[1] Todos los "type" a continuación, cuando estén en un mensaje de respuesta tienen el identificador del proceso que hizo el pedido

Base de Datos de Configuración

Los mensajes que trabaja tienen el formato:

```
char requestType;
char[MAX_OPCODE_SIZE] opCode; -- Código de la opción
char[MAX_OPTION_SIZE] buffer; -- Valor de la opción
```

Los requestType posibles son:

0. Agregar opción
1. Editar opción
2. Eliminar Opción
3. Terminar Aplicación
4. Pedir todas las opciones con la máscara Opcode

La ultima opción se refiere a que todos los Opcode empiezan de una forma determinada si están dentro del mismo grupo. Esto es simplemente para facilitar la recuperación de datos. Obviamente la suma de las opciones debe ser menor a MAX_OPTION_SIZE.

Base de Datos de Idiomas

Es el mismo formato que con los datos de configuración y tiene las mismas opciones.

4. Sistema de Gráficos

Esta parte se encarga de graficar los datos deseados. Para ello, utiliza la biblioteca [4]. Esta sección se basa en un único proceso que recibe mensajes de una cola propia. Al recibir un mensaje, verifica el tipo de gráfico a realizar y le pide a Logics los datos que necesita. Una vez que los tiene, los divide por tipo de sensor y lanza procesos adicionales para manejar cada gráfico por separado. Cada uno de estos procesos se encarga de mostrar los datos y actualizar el gráfico, en el caso de que el mismo sea en tiempo real.

Este módulo se basa en dos partes, una encargada de armar la ventana (GrapherWindow) y una encargada de realizar el gráfico en sí (sensorDataPlot).

5. Interfaz de Usuario

Esta parte está compuesta por la GUI. La misma está diseñada con Qt [3]. Esta decisión de diseño se basa en que Qt permite ser ejecutado en Windows o Linux (lo cuál era un requisito de nuestra aplicación), es fácil de usar y es totalmente orientada a objetos, lo cual simplificaba nuestra elección del lenguaje para programación.

La interfaz presenta tres partes principales, una para mostrar los datos de la última medición, una para configurar la red y cambiar el idioma de la interfaz y una última parte para realizar gráficos.

5.5. Complejidad

Debido a la naturaleza concurrente de la aplicación, es difícil obtener un valor exacto de complejidad total de la aplicación. Por lo que se analizará la complejidad de los procesos más pesados del sistema y se tomará como complejidad total la más grande de todas ellas.

Otro detalle importante es que la aplicación utiliza las librerías Boost, Qt y MathGL, por lo las secciones que utilicen estas librerías pueden no ser exactas del todo.

Primero analizaremos el sector de la aplicación encargado de obtener y almacenar datos nuevos.

Este sector, tendría en primer momento una complejidad de S x N, siendo S la cantidad de sensores y N la cantidad de nodos, pero debido al diseño, los datos de los sensores son procesados de forma paralela, por lo que la complejidad pasa a ser O(N), siendo N la cantidad de nodos en la red. Luego, en el área de almacenamiento, se tiene que el trabajo es de O(S x N), ya que cada par de datos (Sensor-Nodo) se guarda de forma separada, pero dado que la implementación de la base de datos es de $O(N^2)$, debido a que se recorre primero para revisar los espacios vacíos, y que la cantidad de nodos es mayor a la cantidad de sensores (en casos normales de funcionamiento), se puede tomar que la complejidad total del bloque de adquisición y almacenamiento tiene complejidad $O(N^2)$.

Luego, el área de la interfaz gráfica tiene varios procesos que son O(N), pero debido a que su implementación en Qt utiliza hilos de ejecución, se puede tomar como que solo es un proceso O(N) reduciendo fuertemente su complejidad.

Y por último, el área de gráficos, depende que qué gráfico se quiera hacer y qué valores se desean ver, pero suponiendo el peor caso, en el cual se desee graficar todos los valores de la base de datos, la complejidad sería de O(S x N), pero debido a la misma explicación dada para la base de datos, N tiende a ser mayor a la cantidad de sensores, por lo que se puede tomar como peor caso $O(N^2)$.

Finalmente, vemos que tenemos dos sectores con complejidad $O(N^2)$, lo cual en principio es bastante alta. Pero debemos tener en cuenta que estos dos sectores funcionan de forma concurrente, es decir, ambos trabajan al mismo tiempo, generando que se pueda tomar que la complejidad total de la aplicación sea de $O(N^2)$.

5.5.1. Plan de prueba de módulos y de depuración

Las pruebas de la aplicación, se basaron en verificar cada proceso por separado. Una vez que se comprobó que funcionasen, se los probó en grupos.

1. Grupo A - VirtualComHandler, DataUpdater, DataParsers y DataForwarder.
2. Grupo B - Network Config Handler y VirtualComHandler
3. Grupo C - Grupos A más la base de datos.
4. Grupo D - Grupo C + Interfaz de Usuario
5. Grupo E - Grupo D + Sistema de Gráficos

Una vez que se probó todos los grupos, se comprobó el funcionamiento correcto de la aplicación.

6. Construcción del prototipo

En esta sección se presentan los circuitos esquemáticos e impresos que conforman nuestro prototipo. Como herramienta de Software para el desarrollo de los mismos se utilizó el EAGLE, Easily Applicable Graphical Layout Editor Version 5.10.0 for Linux Professional Edition Copyright (c) 1988-2010 CadSoft.

6.1. Definición de los módulos

- Esquemático del circuito Coordinador

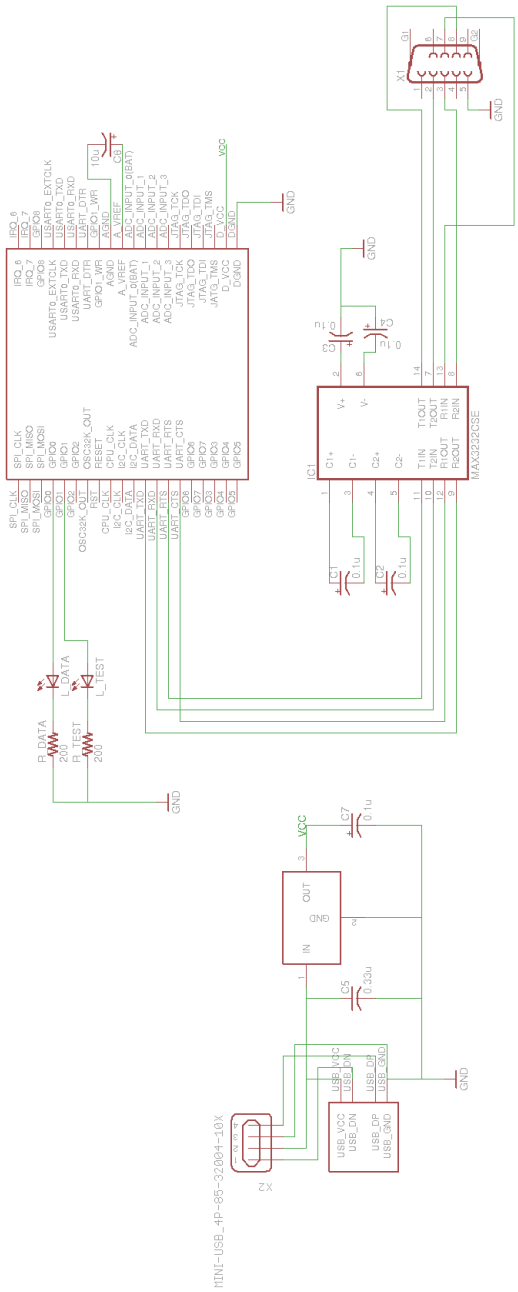

Figura 31: *Circuito esquemático para el Coordinador*

- Esquemático del circuito Nodo

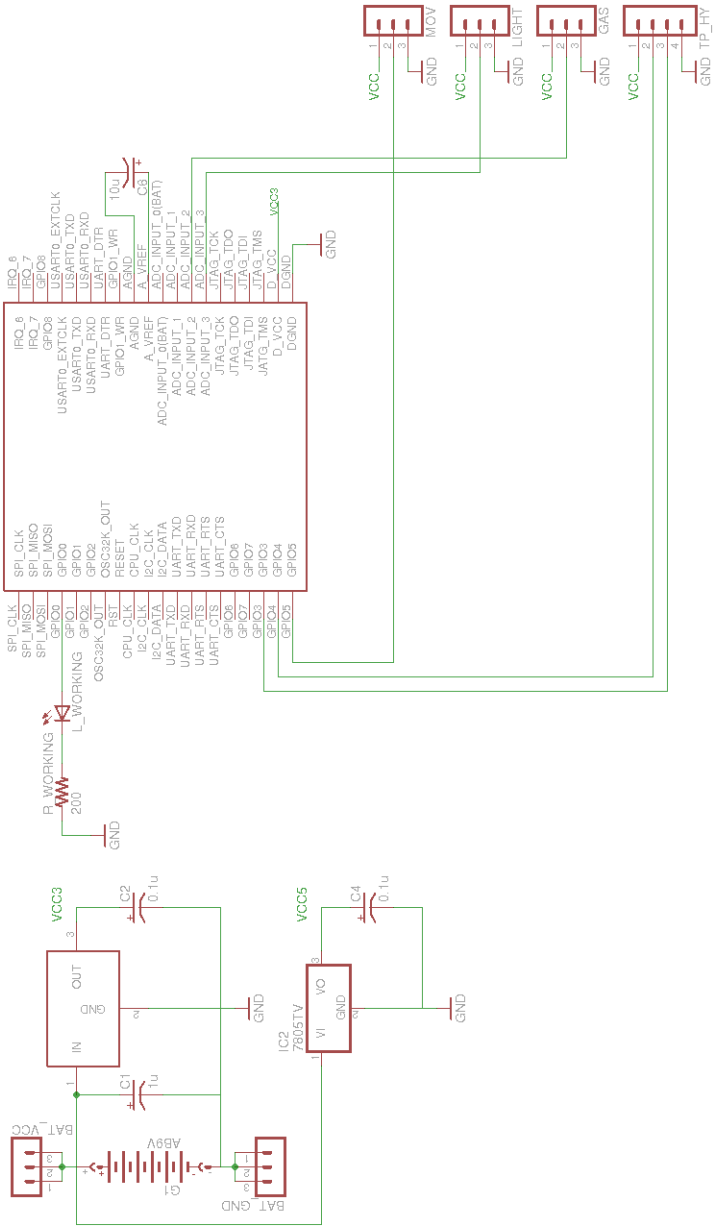

Figura 32: *Circuito esquemático para el Nodo*

- Esquemático de los circuitos para los sensores de humedad, temperatura y luz.

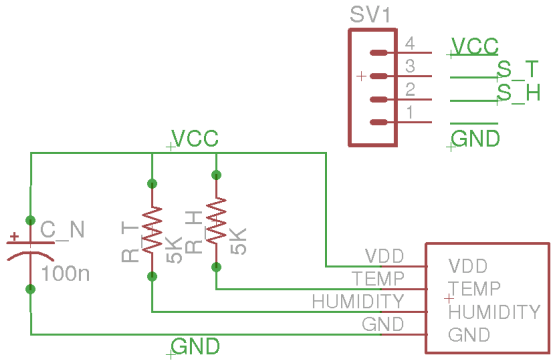

Figura 33: *Circuito esquemático para el sensor de humedad y temperatura*

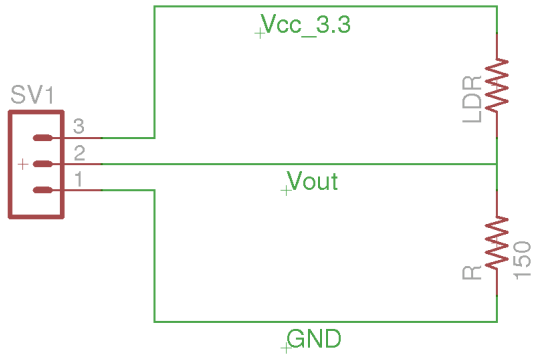

Figura 34: *Circuito esquemático para el sensor de luz*

Cabe aclarar que los sensores de movimiento y gas ya incluyen su circuito, se utilizan mediante un conector.

6.2. Diseño de los circuitos impresos

- Circuito impreso para el Coordinador

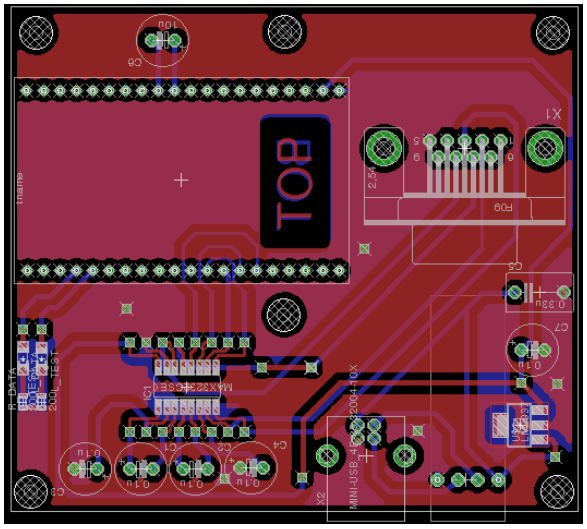

Figura 35: *Circuito impreso para el Coordinador*

- Circuito impreso para el Nodo

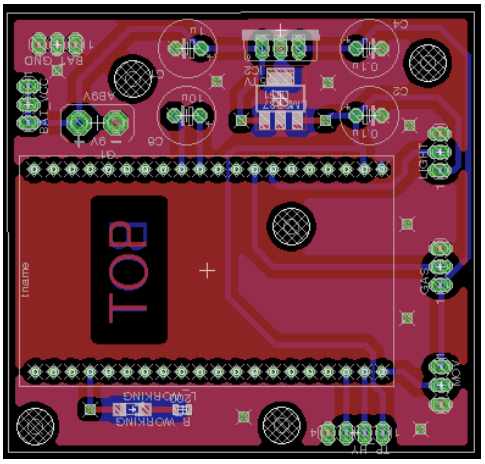

Figura 36: *Circuito impreso para el Nodo*

- Circuito impreso para los sensores de humedad, temperatura y luz.

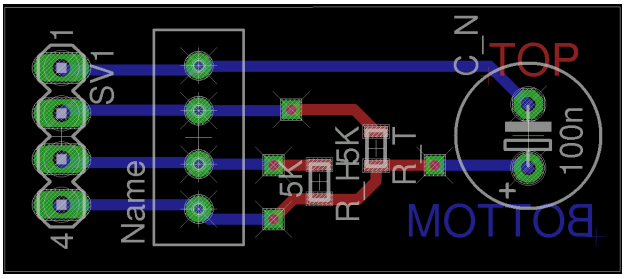

Figura 37: *Circuito impreso para el sensor de humedad y temperatura*

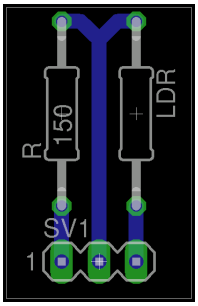

Figura 38: *Circuito impreso para el sensor de luz*

7. Validación del prototipo

En esta sección veremos como validar el prototipo, tanto lo referido al hardware como lo referido al software, contrastándolo a lo que espera recibir el cliente de acuerdo a lo determinado en el **benchmarking de la casa de calidad**.

7.1. Validación de hardware

Recordando las metas a alcanzar eran las siguientes:

- Duración de la batería: 1 años
- Sensor de temperatura: 0°C a 50°C $\pm 2°C$
- Sensor de humedad: 10 % a 90 %
- Sensor de intensidad de luz: 10 lux a 1000 lux
- Sensor de gas: 200 ppm - 5000 ppm para LPG y propano, 300 ppm - 5000 ppm para butano, 5000 ppm - 20000 ppm para metano, 300 ppm - 5000 ppm para H_2 y 100 ppm - 2000 ppm para Alcohol.
- Sensor de movimiento: 2 m rango de detección y 140° ángulo de detección
- Protocolo de RF: ZigBee PRO
- Protocolo Conexión con PC: USB
- Bits ADC: 8 bits
- Período de muestreo del ADC: variable

7.1.1. Plan y protocolos especiales de medición

Dado que no se posee un hardware demasiado complejo, el plan de prueba es seguir con lo establecido en el plan de pruebas descritas en la sección 5.1.3.

En cuanto a la duración de la batería se va a ensayar durante un período de tiempo representativo y se va a medir el consumo durante el mismo, luego extrapolando los resultados vamos a poder concluir si la duración de la batería alcanzará para cubrir ese tiempo.

7.1.2. Medidas

Las medidas finales del prototipo fueron las siguientes (ancho, largo, alto):

- Coordinador: 7 x 9 x 4.5 cm
- Nodo sensor: 5.5 x 8 x 3.5 cm

7.1.3. Evaluación

En cuanto a las especificaciones de los sensores, hemos visto en la sección 4.1.2 que satisfacen las metas propuestas en un principio, incluso mencionamos que eligiendo otro tipo de sensores de precisión como por ejemplo para la medición de temperatura cubrimos con mayor margen este requerimiento.

El sensor de temperatura y humedad DTH11, el sensor de gas MQ-2, el sensor de intensidad de luz LDR-NORP12 y el sensor de movimiento P-381 cubren las metas propuestas.

En lo referido al microcontrolador y el módulo de RF, utilizamos el módulo de Atmel ATZB-24-B0, el cual como hemos visto, en sus características posee un ADC de 8 bits de período de muestreo

variable, el cual lo utilizamos para obtener el valor entregado por los sensores donde medimos a intervalos definidos por el usuario.

Se estableció una conexión USB para conectar la placa del coordinador a la PC, con lo cual esta meta esta completamente cubierta.

De acuerdo al ensayo de la batería se pudo observar que la misma puede durar aproximadamente 4.5 años.

7.1.4. Resultados

Vemos que de la sección anterior se desprenden resultados positivos y que las metas propuestas en el benchmarking de la casa de calidad se cubren ampliamente.

En cuanto a la vida útil de la batería se vio que puede alcanzar los 4.5 años, período más que suficiente para establecer una garantía del producto de 4 años, aclarando por supuesto que consideramos como único elemento influyente en la vida útil de nuestro producto la vida útil de la batería. Esto se aclarará con más detalle en la sección 8 donde se realizará el estudio de confiabilidad.

7.2. Validación de software

Para la validación del software, se utilizó básicamente el debugger (programa que corre aplicaciones línea por línea) del gdb. Para facilitar su uso, ya que el mismo es un programa por consola, se utilizó la interfaz de desarrollo Eclipse, la cual tiene integrado al debugger, por lo que su uso se simplifica de forma enorme.

La validación del software fue diferente para el software de PC y el firmware de los módulos, por lo que se explicarán por separado.

7.2.1. Validación Software PC

Primero explicaremos el código de PC, ya que fue el primero testeado. Y se necesitó para probar completamente el firmware.

Como primer etapa se tomó cada proceso de la aplicación, y se lo aislo del resto. Esta acción permite ver los problemas internos al proceso, sin verse afectado por factores externos que pueden alterar el funcionamiento del mismo. Una vez separados, se crearon pequeños programas que testeaban cada uno de las funciones de cada proceso y comparaban las salidas de forma de evaluar su correcto funcionamiento. Notar que en primera instancia se corre el programa con el debugger, de forma de poder ir viendo los errores comunes mientras van sucediendo. Además, es una buena práctica, ya que permite re pensar el código escrito mientras va funcionando, pudiendo encontrar errores de lógica mucho maś rápido.

Si el proceso pasa el test, se lo considera funcionando, si no, se lo sigue probando hasta que funcione totalmente. Nunca se pueden mezclar procesos que no se sabe si funcionan o no, ya que es muy difícil entonces saber cuál tiene el problema.

Una vez, que todos los procesos funcionen por si solos, se procede a probarlos en grupos. Cada grupo está formado por procesos con interacción directa (en nuestro caso, un ejemplo serían los parsers y el DataUpdater o DataForwarder). Cada grupo tiene a su vez un pequeño de programa de testeo que crea interacciones entre los procesos, por ejemplo, crea un mensaje que el proceso A, tiene que preguntarle al proceso B cómo trabajarlo. Este test es más complicado, ya que como ambos procesos corren al mismo tiempo, se necesita correr el debugger dos veces para poder ver todas las interacciones. Aunque, es una práctica común mostrar mensajes de control en la pantalla de la consola, de forma de saber qué es lo que está haciendo el proceso en todo momento.

Al terminar de probar todos los grupos, se procede a probar el programa en conjunto. En esta etapa es más cómodo utilizar los mensajes de control directamente, ya que el manejo de todos los procesos con el debugger puede llegar a ser dificultoso.

Un detalle importante, para nuestro programa en particular, es el caso de la prueba del puerto serie, ya que nuestra aplicación contiene un código para leer y escribir sobre el puerto serie. Para verificar su correcto funcionamiento se utilizaron dos aplicaciones, una llamada HyperTerminal, que corre sobre la plataforma Windows y otra llamada CuteCom que sería el equivalente del HyperTerminal sobre la plataforma Linux (Ubuntu para ser más específico). Luego, desde otra PC se generaron datos por el puerto serie, que eran enviados a nuestra PC, los cuales eran leídos con el código de nuestra aplicación.

Esta última sección es de vital importancia, ya que es de la cual depende el funcionamiento correcto de la aplicación, por lo que su tiempo de prueba fue uno de los más extensos.

7.2.2. Validación Firmware

En el caso del firmware, para probarlo, se utilizaron dos herramientas, una es el debugger gdb para hardware (también a través del Eclipse) y otra fueron los leds del circuito mismo. En el caso del debugger, esta opción no fue muy utilizada. Esto se debe a que como todo debugger, este permite configurar puntos en los cuales la ejecución del programa se detendrá, permitiendo analizar el estado de la aplicación en ese momento (por ejemplo, permite ver los valores de las variables y registros en ese momento). Estos puntos son llamados "Breakpoints". En una aplicación de PC, estos son fáciles de configurar y existe una gran cantidad de los mismos. Pero en un firmware, su cantidad es limitada, sin decir que existen Breakpoints realizados por hardware, los cuales son más rápidos y realizados por software, los cuales son más lentos, ya que se insertan líneas de código especiales en el firmware, las cuales el debugger debe buscar en el momento de ejecución, es decir, cada línea que lee debe verificar si no es un breakpoint. Esto reduce enormemente la velocidad de ejecución del programa, lo cual a veces no puede pasar, ya que la aplicación es de tiempo real, con lo que tiene los tiempos muy juntos y bien establecidos.

Por esta razón, el debugger solo fue utilizado en situaciones en las que el otro método no funcionaba bien. El otro método utilizado se basa en realizar una secuencia de leds según el estado de la aplicación. Esto permitía saber en todo momento que es lo que estaba sucediendo. Con este método se probó la comunicación ZigBee (se enviaban dos valores, si se recibía el valor A, se prendía un led, y si se recibía el valor B, se prendía otro led de forma alternada) y la transmisión serie (si recibía una palabra clave prendía un led y si no prendía otro).

Una vez que se tuvo todo probado, se procedió a conectarlo con un cable USB a la computadora, de forma de ver la comunicación con la misma. Para esto, se utilizaron los programas anteriormente mencionados, HyperTerminal y Cutecom. Cuando esto se terminó de probar, se verificó la aplicación con nuestro programa de PC, de forma tal de poder comprobar el funcionamiento de ambas aplicaciones.

8. Estudios de confiabilidad

La confiabilidad de un producto se mide según la probabilidad de que pueda cumplir una función durante un lapso de tiempo bajo ciertas condiciones de carga estipuladas. Cuando el dispositivo es inhábil para seguir cumpliendo su función se dice que entró en falla.

Existen distintos métodos para estimar la tasa de fallas de un componente, uno de ellos es el método establecido en la **norma militar HDBK-217**, método que se adoptará en este trabajo para realizar los estudios de confiabilidad.

8.1. Hardware

Lo primero que se supondrá es que la tasa de fallas de un elemento es constante, es decir, que se está en la etapa donde las únicas fallas que ocurren son de naturaleza aleatorias. Ya se pasó previamente las fallas infantiles (las primeras horas de funcionamiento) y aún no se llegó a las fallas por degradación. Por lo tanto, el valor de la tasa de fallas $\lambda_i(t)$ de un componente i será reemplazado por λ_i. Los valores de la tasa de fallas para cada componente serán calculados según lo indica el MIL-HDBK-217F.

Se define la expresión generalizada de la confiabilidad como

$$R(t) = e^{\int_0^t \lambda_i(t)dt}$$

Teniendo en cuenta las suposiciones hechas previamente, se tiene entonces que

$$R(t) = e^{-\lambda t}$$

Además de la confiabilidad, otro parámetro indicativo es el tiempo medio entre fallas llamado TMEF. Este tiempo se define con la esperanza matemática de primer orden de las fallas

$$\text{TMEF } = \int_0^\infty t f(t)dt$$

donde $f(t)$ es la función densidad de las fallas. Se deduce de ahí que

$$\text{TMEF } = \int_0^\infty R(t)dt$$

Para este caso se propone utilizar una distribución de probabilidad exponencial.

Entonces, con este modelo se tiene que

$$R(t) = e^{-\lambda t} \Rightarrow F(t) = 1 - R(t) = 1 - e^{-\lambda t} \Rightarrow f(t) = \frac{dF(t)}{dt} = -\lambda e^{-\lambda t}$$

Por lo que si las fallas son de tipos accidentales $\lambda(t) = \lambda_o$ entonces resulta que

$$\text{TMEF } = \frac{1}{\lambda_o}$$

Para la estimación de la confiabilidad, se aplicará el método de redes. Aquí cada componente está asociado a un bloque de confiabilidad, el cual se vincula con otros bloques según el modo en que su falla incida en la falla del sistema. Las formas de asociación pueden ser en paralelo o serie.

La vinculación de dos bloques en forma serie significa que la falla de uno de los dos, hace que falle el sistema. Este es el caso que se supondrá. La confiabilidad del sistema estará determinada por la multiplicación de las confiabilidades de cada elemento

$$R(t) = R_1(t) \times ... \times R_N(t) = e^{-\lambda_1 t} \times ... \times e^{-\lambda_N t} = e^{-\lambda_{Total} t}$$

Por lo tanto si

$$\lambda_{Total} = \sum_i \lambda_i \text{ resulta que TMEF } = \frac{1}{\lambda_{Total}}$$

8.1.1. Cálculo de las tasas de falla

Resistores

Se utilizaron en el dispositivo 5 resistores SMD y 1 resistencia de carbón.

Según la norma, se debe aplicar la siguiente fórmula para calcular la tasa de fallas:

$$\lambda_p = \lambda_b \times \pi_T \times \pi_P \times \pi_S \times \pi_R \times \pi_Q \times \pi_E$$

Utilizando la norma MIL-R-22684, se establece la tasa de fallas para las resistencias:

$$\begin{aligned}
&\lambda_b = 0,00066 \\
&\pi_R = 1 \quad \text{(Resistores} < \text{1M)} \\
&\pi_Q = 15 \quad \text{(Resistores comerciales)} \\
&\pi_E = 5
\end{aligned}$$

$$\lambda_p = 0,13955 \text{ fallas}/10^6 \text{ horas}$$

Obteniendo en total:

$$\lambda_{p(resistores)} = 0,297 \text{ fallas}/10^6 \text{ horas}$$

Capacitores

Se utilizaron en el dispositivo 12 capacitores, de los cuales 3 son cerámicos de 100nF, y 9 electrolíticos, 4 de 0.1μF, 3 de 1μF y 2 de 10μF.

En la norma militar se puede encontrar la siguiente fórmula:

$$\lambda_p = \lambda_b \times \pi_{CV} \times \pi_Q \times \pi_E$$

- Capacitores Cerámicos

$$\begin{aligned}
\lambda_b &= 0,0015 \\
\pi_{CV} &= 0,75 \\
\pi_Q &= 10 \\
\pi_E &= 2
\end{aligned}$$

- Capacitores Electrolíticos

$$\begin{aligned} \lambda_b &= 0,0015 \\ \pi_{CV} &= 1,4\ 10\text{u} \\ \pi_Q &= 10 \\ \pi_E &= 2 \end{aligned}$$

Obtenemos en total considerando ambos tipos de capacitores:

$$\lambda_{p(capacitores)} = 0,0675 + 0,378 = 0,4455 \text{ fallas}/10^6 \text{ horas}$$

Circuitos integrados

Para los circuitos integrados la tasa de fallas se calcula como:

$$\lambda_p = (C_1 \times \pi_T + C_2 \times \pi_E) \times \pi_L \times \pi_Q$$

- Módulo ZigBee

$$\begin{aligned} C_1 &= 0,14 \\ C_2 &= 0,015 \\ \pi_T &= 0,71 \\ \pi_L &= 1 \\ \pi_V &= 1 \\ \pi_Q &= 2 \\ \pi_E &= 4 \end{aligned}$$

$$\lambda_{p(micro)} = 0,3188 \text{ fallas}/10^6 \text{ horas}$$

- MAX3232

$$\begin{aligned} C_1 &= 0,01 \\ C_2 &= 0,0056 \\ \pi_T &= 0,71 \\ \pi_L &= 1 \\ \pi_Q &= 2 \\ \pi_E &= 4 \end{aligned}$$

$$\lambda_{p(max232)} = 0,059 \text{ fallas}/10^6 \text{ horas}$$

Semiconductores discretos

- Diodos

Los diodos siguen la siguiente ley de fallas:

$$\lambda_p = \lambda_b \times \pi_T \times \pi_S \times \pi_C \times \pi_Q \times \pi_E$$

Se utilizaron 2 diodos LED SMD.

$$\begin{aligned} \lambda_b &= 0,001 \\ \pi_T &= 1,6 \\ \pi_S &= 0,77 \\ \pi_C &= 1 \\ \pi_Q &= 0,7 \\ \pi_E &= 6 \end{aligned}$$

$$\lambda_{p(diodos)} = 0,0103 \text{ fallas}/10^6 \text{ horas}$$

- Conectores

Se tienen 3 conectores en el sistema, para la conexión de la batería, para la conexión con la PC mediante USB y un conector DB-9.

$$\lambda_p = \lambda_b \times \pi_T \times \pi_K \times \pi_Q \times \pi_E$$

$$\begin{aligned} \lambda_b &= 0,046 \\ \pi_T &= 1,5 \\ \pi_K &= 1 \\ \pi_Q &= 2 \\ \pi_E &= 1 \end{aligned}$$

$$\lambda_{p(conectores)} = 0,552 \text{ fallas}/10^6 \text{ horas}$$

- Soldaduras

Para las soldaduras, la norma establece la siguiente ecuación:

$$\lambda_p = \lambda_b \times \pi_E$$

$$\begin{aligned} \lambda_b &= 0,0013 \\ \pi_E &= 2 \end{aligned}$$

$$\lambda_{p(soldaduras)} = 107 \times 0,0026 \text{ fallas}/10^6 \text{ horas } = 0,2782 \text{ fallas}/10^6 \text{ horas}$$

8.1.2. Tasa media entre fallas

Se pone a continuación una tabla con el resumen de las tasas de fallas para cada componente:

Componentes	Tasa de fallas
Resistores	0,297
Capacitores	0,4455
Circuitos integrados	0,3778
Semiconductores discretos	0,0103
Conectores	0,552
Soldaduras	0,2782
Total	1,96

Si se considera que la tasa de fallas es constante $\lambda(t) = \lambda_o$ entonces teníamos que

$$\lambda_{Total} = \sum_i \lambda_i \quad \text{resulta que TMEF} \ = \frac{1}{\lambda_{Total}}$$

$$\text{TMEF} \ = 510113 \text{ horas} \ = 58,2 \text{ años}$$

8.1.3. Disponibilidad

La disponibilidad se define como la probabilidad de que un equipo se encuentre operando, o en capacidad de operar, en un instante de tiempo t cualquiera, partiendo de una condición inicial prefijada, que puede ser de funcionamiento, o desde un estado de falla. Mientras que la confiabilidad es aplicable tanto a equipos reparables y no reparables, la disponibilidad es una característica propia de los equipos reparables. Teniendo en cuenta las hipótesis de que la probabilidad de pasar de un estado a otro ("funcionar" y "no funcionar") en un intervalo de tiempo es proporcional a dicho tiempo y también teniendo en cuenta que no se puede dar más de un salto de estado a la vez, se llega a unas ecuaciones diferenciales que desembocan en la fórmula de disponibilidad, citada a continuación.

$$D = \frac{TMEF}{TMEF + TMR}$$

Si un equipo que es reparable falla, cabe esperar que en algún momento posterior se encuentre nuevamente en funcionamiento. La disponibilidad, A(t), mide precisamente la probabilidad de encontrar el equipo funcionando en un instante cualquiera t. Si el instante t está suficientemente alejado del instante en que entro en reparación o se hizo la reposición de funcionamiento, la disponibilidad resulta independiente de t, y define lo

que se conoce como disponibilidad a largo plazo o disponibilidad final del equipo. Se supondrá que el tiempo que no se tendrá el dispositivo disponible será a lo sumo de un día. Entonces se tiene que

$$D = \frac{TMEF}{TMEF + TMR} = \frac{21254 \text{ días}}{(21254 + 1) \text{ días}} = 0,99980$$

8.1.4. Garantía

Recordando que la fiabilidad se la define como la probabilidad de que el equipo funcione durante un lapso de tiempo bajo condiciones de carga prefijadas, es lógico pensar que el plazo de garantía esté relacionado con la misma. También se debe observar que el plazo de garantía debe ser menor al tiempo medio entre fallas (TMF), pues las fallas que aparezcan dentro del plazo de garantía serán reparadas sin cargo para el usuario, y dado que esto representa un cargo para el fabricante, debe haber una baja probabilidad de que aparezcan fallas dentro del plazo de garantía. La fórmula que relaciona a la fiabilidad, $R(t)$, el tiempo de garantía (T_G) y el tiempo medio entre fallas (TMF), suponiendo que la distribución de fallas respondiera a una ley exponencial, es la siguiente.

$$R(T_G) = e^{\frac{T_G}{TMEF}}$$

Como deseamos que durante el período de garantía los reclamos de los usuarios sean mínimos, deberemos exigirnos una fiabilidad lo más alta posible durante dicho lapso de tiempo, escogiendo un valor razonable, podemos decir que una fiabilidad del equipo durante el plazo de garantía del 90 %, es decir $R(T_G) = 0{,}9$ es aceptable.

Recordando que el tiempo medio entre fallas (TMF) era de 58,2 años, pasamos a reemplazar en la fórmula mencionada.

$$0,9 = e^{\frac{T_G}{14,2}}$$

Aplicando el logaritmo neperiano y operando luego, obtenemos que el tiempo de garantía resultantes es $TG = 6,13$ años ; en base a este resultado y a los períodos de garantía que otorgan los productos de la competencia, optamos por otorgar una garantía final de 4 años, que además es lo que se espera que dure la batería antes de que se necesite reemplazarla.

Tiempo de Garantía otorgado $T_G = 4$ años

El periodo de garantía del producto será de 4 año a partir de la fecha de compra del equipo. Durante este periodo el equipo estará cubierto por la ley 24240/93 y deberá ser reparado en caso de presentar algún defecto o vicio de cualquier índole que no permita el correcto funcionamiento del mismo sin costo adicional para el cliente. Si el equipo no puede ser reparado, el mismo será reemplazado por un equipo de similares características o se devolverá el costo del equipo al cliente. Están exentos de la garantía, aquellos equipos sujetos a abusos, usos inapropiados, negligencia, accidentes, modificaciones o falta del usuario final de seguir los procedimientos de operación y mantenimiento fuera de lo establecido en el manual del usuario, intento de reparación por personal no autorizado, la operación de la unidad fuera de los medios publicados y de los parámetros eléctricos normales.

Con el manual de usuario se expedirá el certificado de garantía legal según lo establecido por la ley 24240/93 de defensa del consumidor y su actualización (ley 24999/98).

8.1.5. Mantenibilidad

Tomando como base la confiabilidad calculada anteriormente, ofrecer una garantía de 4 años resulta atractivo para el cliente, que según se estableció en la casa de calidad, posee un interés medio en este parámetro, y al mismo tiempo se garantiza estar lejos del tiempo medio entre fallas.

Debido al bajo costo de producción y el gran porcentaje de ganancia, como fue analizado anteriormente, se opta por una garantía de reemplazo, lo cual introduce un mayor compromiso al usuario por parte de la empresa. Este punto puede ser utilizado como elemento de marketing, y a su vez, considerando el bajo coste de los componentes y su eventual reparación, se podría plantear la venta de productos remanufacturados (refurbished), con una garantía de 6 meses a un año y un precio considerablemente menor.

9. Conclusiones

En este trabajo se presentó una propuesta original empleando una tecnología en pleno auge y novedosa para el país como lo es una red de sensores inalámbricos para controlar el estado de diversas variables físicas como temperatura, humedad, emisión de gases en cualquier entorno empresarial, como ejemplo de esto podemos citar los data centers, para tener un control del consumo energético y poder detectar zonas específicas como los "hotspot" y tener más control sobre las condiciones ambientales en la sala.

Se propuso un producto simple de instalar, que no requiera cambios en la infraestructura del consumidor, que sea fácil de utilizar, simplemente se disponen los módulos sensores en la zona de interés y luego se cuenta con un software de PC para monitorear el estado de los mismos. Además de ser fácil de mantener. Se propuso una garantía de 4 años y ante cualquier falla del módulo, este se reemplaza por uno nuevo, no generando ninguna complicación o trauma al usuario.

En cuento al desarrollo y la factibilidad económica, se hizo hincapié en poder contar con ventaja económica con respecto a productos similares de la competencia. El factor económico resulta de mucho interés por parte del cliente, afectando las decisiones de aplicar una solución como la que proponemos en este trabajo.

Muchas veces por falta de recursos económicos se decide no monitorear ninguna variable, cuando en realidad esto es ventajoso para su empresa pudiendo determinar una mejor aplicación de estos recursos.

A continuación resumimos las excelencias y los objetivos alcanzados.

9.1. Excelencias y objetivos alcanzados

Luego de evaluar y verificar distintas propuestas, se decidió por ZigBee para resolver la comunicación inalámbrica entre los módulos sensores por las ventajas que este presenta frente a Wi-Fi o Bluetooth por ejemplo. El stack de ZigBee permite resolver de manera sencilla la capa de red y el acceso al medio, el control de errores y cuenta con protocolos eficientes y confiables para la transmisión y recepción de datos.

El resto del hardware gira en torno a esta decisión, en cuanto a los sensores se eligieron aquellos con características estándar, no de precisión ya que solo desarrollamos un prototipo, pero pueden ser reemplazados en cuanto a los requerimientos puntuales de cada cliente.

Al utilizar componentes SMD obtuvimos un TMF muy alto, dando lugar a un período de garantía de 4 años, adecuado para la vida útil de la batería. Se espera reemplazar el equipo antes de que se agote la batería del mismo.

Por otro lado contamos con una interfaz para la PC desarrollada en C++ empleando técnicas de programación concurrentes y utilizando el conjunto de bibliotecas de Boost para resolver de manera eficiente y prolija la compatibilidad entre los distintos sistemas operativos y ofrecer una solución multiplataforma. La elección de C++ como lenguaje de desarrollo permitió un software robusto y estable, siendo confiable para el cliente. La interfaz gráfica se creó utilizando las bibliotecas de QT. Vale destacar la opción para seleccionar el idioma, siendo español, inglés y portugués los idiomas soportados.

- Se desarrolló completamente en ZigBee.
- Se utilizaron las ventajas de su stack y una topología de estrella.
- La modificación a topología de malla es muy sencilla.
- La comunicación es robusta, no se ve afectada por interferencias ni degradaciones.
- La comunicación es completamente asincrónica.
- La red se comporta de manera correcta sin importar el orden ni el tiempo de llegada de los datos.
- Es versátil. Se adapta a cualquier forma de inmueble o ambiente donde se quiera aplicar.

- Al ser inalámbrico facilita la escalabilidad.
- La red se puede expandir aumentando la cantidad de módulos.
- Se asentaron las bases de documentación y código para lograr modificar la aplicación a otras necesidades.
- Los objetivos de consumo, potencia y radio de alcance fueron alcanzados.
- Los costos se vieron disminuidos al usar un equipo con tecnología distribuida. De presentarse una falla, no hace falta reemplazar por completo el equipo, sino que se puede reemplazar el módulo.
- El software de PC es multiplataforma y multi-idiomas.
- Permite ser ejecutado en teléfonos celulares con Symbian ®.
- Al ser concurrente cuenta con la capacidad de ejecutarse en máquinas remotas.

9.2. Recomendaciones para futuros diseños

Se desarrolló un producto pensando en las necesidades de los potenciales clientes y se resolvieron de forma completa. Sin embargo, se podrían realizar ciertas modificaciones para mejorar las prestaciones de las mismas.

Se podría expandir el mercado al generar una interoperabilidad entre ZigBee y otras tecnologías, pero esto conllevaría desarrollar protocolos de comunicación nuevos que a su vez requerirían más esfuerzo y tiempo.

En cuanto al firmware desarrollado en los módulos, una mejora posible es utilizar la posición relativa y generar un mapa de la red con la posición de cada módulo sensor para brindar más información al cliente.

Por el lado del software, se podría pensar un desarrollo que permita ejecutarse en cualquier teléfono celular del tipo “smart phone”. Actualmente, al emplear las bibliotecas de QT sólo se cuenta con la posibilidad de correr en teléfonos con Symbian. Una mejora entonces, sería incluir bibliotecas de programación para correr en teléfonos tipo Android ®o Iphone ®. De esta manera agregaríamos una característica adicional y potenciales clientes y contratos.

A. Manuales de operación, soporte e instalación

Hardware

Los dispositivos son muy fáciles de usar, solo es cuestión de ubicar los nodos en las zonas a medir y luego conectar el coordinador a la computadora personal a través del cable USB provisto con el kit. Una vez conectado, la aplicación comenzará de forma automática a recibir datos. Notar que se debe lanzar la aplicación **después** de haber conectado el coordinador, ya que sino, la aplicación no se lanzará correctamente.

Software

Instalación

Para la instalación, se debe tener en cuenta el sistema operativo a utilizar:

Windows En este caso, se debe utilizar el ejecutable netVisorSetup.msi. El mismo lo guiará durante la instalación.

Linux En este caso, se debe correr el script netVisorSetup.sh. El mismo se encargará de compilar, crear carpetas y mover los archivos de forma de asegurar el correcto funcionamiento de la aplicación.

Se deben observar las siguientes notas para la instalación y uso del sistema:

- En windows es necesario tener Windows XP, 2003, 7 o Vista. De 32 o 64 bits. Es preferible tener un procesador multi-núcleo, de forma de aprovechar el paralelismo de los módulos del sistema.
- En windows 7 de 64bits existe un uso intensivo del procesador, por lo que se recomienda monitorear las temperatura del mismo en todo momento. O como segunda opción, no utilizar el sistema durante tiempos prolongados.
- En Linux, se necesita Ubuntu 10.04 en adelante, utilizando Gnome 2 en adelante. En Kde funciona bien en cualquier versión que utilice Qt 4.

Inicio de Aplicación

La lanzar la aplicación se debe ejecutar el programa *SNS_initSystem*, el mismo se encargará de cargar todos los módulos necesarios para el correcto funcionamiento de la aplicación. Este programa puede ser lanzado desde la consola, con los siguientes argumentos:

```
Argumentos: c|d|n n|m|g [puerto a usar de mi lado ip del otro lado puerto del otro lado]
```

- Argumento 1:
  - **c** - Solo crear ipcs
  - **d** - Solo destruir ipcs
  - **n** - Lanzamiento Normal
- Argumento 2:
  - **n** - Lanzamiento Completo para 1 PC
  - **m** - Lanzar Solo mediciones con TCP
  - **g** - Lanzar Solo GUI con TCP

Nota 1: Si se utiliza el parámetro "g", se deben especificar el puerto por el que se escuchan las conexiones entrantes y la dirección y puerto del otro lado, ya sea el de mediciones o el de la interfaz gráfica.

Nota 2: Se puede poner cualquier otra letra en el segundo parámetro para que solo se lancen las mediciones, pero sin GUI ni conexiones TCP

Uso de Aplicación

Sección 1: Mediciones

Una vez iniciada, aparecerá la primera sección de la aplicación (figura 39).

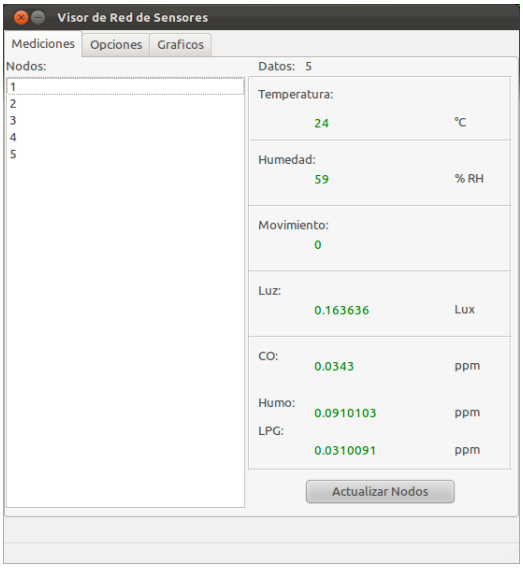

Figura 39: *GUI: Sección de Mediciones*

En la misma, podemos apreciar 4 sectores principales:

- El primero, formado por una barra superior, las cuales permiten ver la información de la aplicación, cómo su nombre y derechos legales, y una opción para poder salir de la aplicación.
- Debajo del anterior, existen 3 tabs, para poder navegar por las diferentes secciones del sistema, la sección de mediciones, la de configuración y la de gráficos y exportación de datos.
- Luego, a la izquierda tenemos una lista con los nodos activos de la red. Cuándo se inicia por primera vez la aplicación, esta lista estará vacía y se irá completando conforme los nodos envíen información al sistema.

  Cada nodo en la lista puede ser seleccionado para ver sus datos con el botón izquierdo del mouse y puede ser configurado con el botón derecho. Al realizar está acción, un menú aparece que permite cambiar el nombre a mostrar del nodo seleccionado y además, permite eliminar el nodo de la lista. Está acción limpia los datos recibidos por ese nodo, pero no deshabilita al mismo, por lo que se volverá a recibir datos del nodo y volverá a aparecer en la lista de nodos.
- Finalmente, en la sección de la derecha, aparecen los datos del nodo seleccionado, es decir, su nombre y las últimas mediciones recibidas del mismo. En el caso que alguna medición supere el valor de alarma, su valor será mostrado en color rojo. Si la aplicación está minimizada, el trayicon mostrará un mensaje de alerta para avisar al usuario.

Sección 2: Opciones

Esta sección se encarga de las configuraciones principales del sistema, ya sea de la aplicación o de la red misma. Ver figura 40

Figura 40: *GUI: Sección de Opciones*

En esta sección, se puede configurar el idioma de la interfaz, ya sea castellano, inglés o portugués, cambiar el tiempo de muestreo del sistema (utilizando los tiempos predeterminados) y cambiar las alarmas del sistema. Si se desea cambiar solo una alarma, se agrega ese valor en el espacio adecuado y se deja el resto en blanco. Si no es un valor válido, se lo ignora. Y si se agrega un valor a más de una posición, solo los sensores con valores serán cambiados.

Al cambiar de idioma, la interfaz se traduce en el momento, pudiéndose ver el resultado en la figura 41.

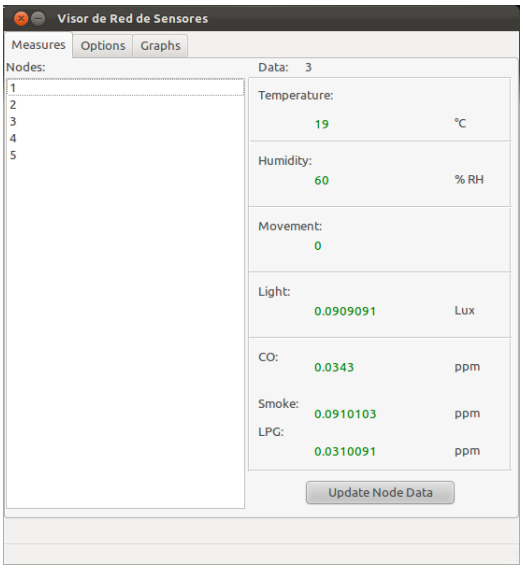

Figura 41: *GUI: Traducida*

Sección 3: Gráficos

En esta última sección podemos especificar un grupo de datos para ser graficados o que se los exporte a un archivo CSV (comma separeted values).

Inicialmente se debe elegir qué nodos se tomarán en cuenta para el gráfico, seleccionándolos de la lista y presionando en “Agregar Nodo”. Si se desea quitar algún nodo de la selección, se debe seleccionar el nodo a quitar de la lista a la derecha y presionar sobre “Quitar Nodo”.

Luego, utilizando los checkboxs, se puede elegir qué tipos de sensores van a ser incluidos en la selección. Cada tipo de sensor, en el caso de ser graficado, se realizará en un gráfico distinto, es decir por ejemplo, si se desea graficar los datos de temperatura y humedad de los nodos 1 y 3, aparecerán dos gráficos uno con los datos de temperatura del nodo 1 y 3 y otro con los datos de humedad de los nodos.

También, se puede elegir un intervalo en el tiempo para la selección de la muestras. Notar que si el intervalo tiene como cota superior una fecha en el futuro, el gráfico se realizará en tiempo real, es decir, con cada muestra que llegué, que cumpla con la selección del gráfico, el mismo se actualizará automáticamente.

Como paso adicional, si se desea realizar un gráfico, se debe seleccionar de qué tipo será este. Actualmente, solo está habilitado el gráfico de tiempo.

Una vez que se tiene la selección deseada, se puede graficar, presionando sobre el botón “Graficar” o guardar en un archivo csv, si se lo desea guardar en un archivo para su posterior proceso.

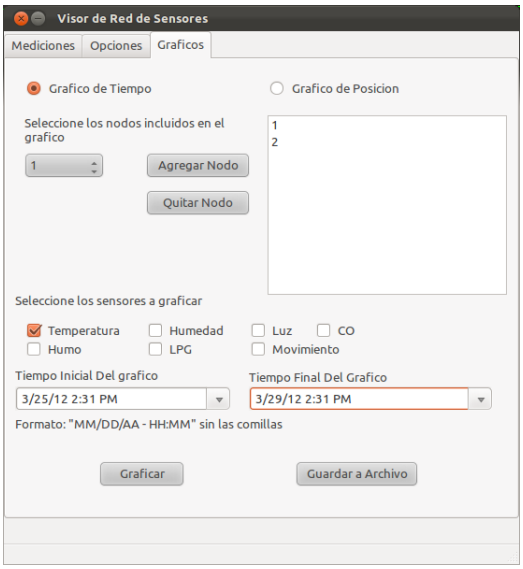

Figura 42: *GUI: Sección de Gráficos*

Si se realizó un gráfico, se lo podrá ver en la ventana de la figura 43. El mismo se puede maximizar para poder apreciar mejor las curvas deseadas. Además, utilizando el menú a la derecha, se puede seleccionar cada curva en particular y ocultarla o hacerla aparecer.

Y como última opción, se puede imprimir el gráfico, presionando sobre el botón de imprimir, ubicado en la parte superior de la venta.

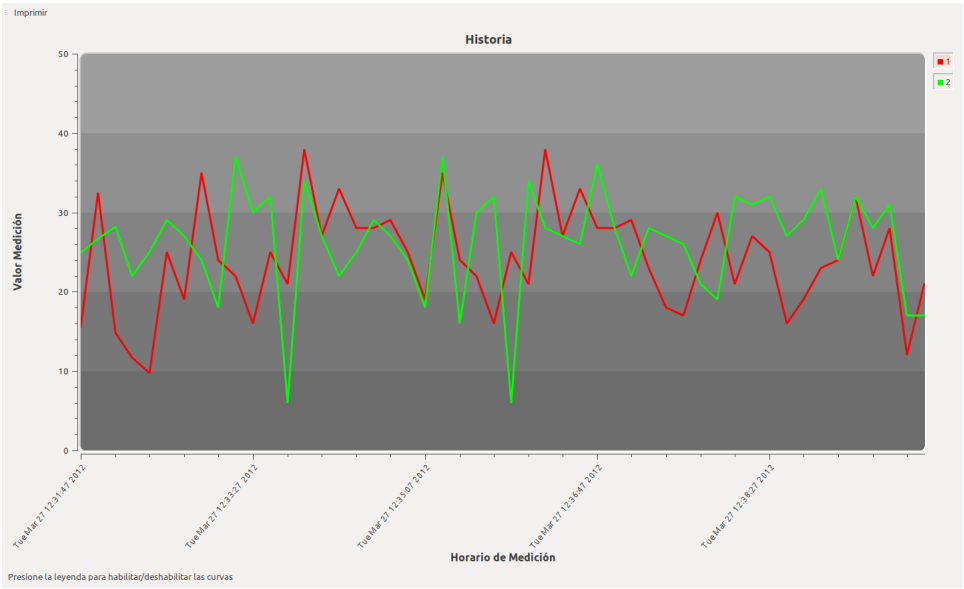

Figura 43: *Ventana de Gráficos*

Acerca del CD

El CD que acompaña este documento incluye:

- Esquemáticos correspondientes al nodo coordinador y nodo sensor.
- Código fuente del firmware y el software de PC.
- Hojas de datos y notas de aplicación.

Apéndices en el CD

B. Planos

C. Esquemas

D. Listado de partes

E. Códigos de software

F. Hojas de datos de componentes

G. Hojas de aplicación

Referencias

[1] Ieee standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (lr-wpans). *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pages 1–314, 5 2011.

[2] Ember, http://ember.com/products/tools/development.html.

[3] Qt - a cross-platform application and ui framework, http://qt.nokia.com/.

[4] Qwt - qt widgets for technical applications, http://qwt.sourceforge.net/.

[5] Atmel corporation, http://www.atmel.com/.

[6] Bitcloud - zigbee pro, http://www.atmel.com/tools/BITCLOUD-ZIGBEEPRO.aspx.

[7] Boost - c++ libraries, http://www.boost.org/.

[8] Microdaq, http://www.microdaq.com.

[9] Millennial net, http://www.millennial.net.

[10] Crossbow, http://www.xbow.com/Products/productsdetails.aspx?sid=3.

[11] Tae-Yun Chung. Wireless sensor networks. Technical Report US 2011/0069611 A1, United States Patent Application, March 2011.

[12] M Hydeman. Wireless sensors improve data center energy efficiency. Technical report, SynapSense Corporation, September 2010.

[13] Edward K.Y. Data storage for distributed sensor networks. Technical Report US 2006/0026164 A1, United States Patent Application, February 2006.

[14] Jin-Shyan Lee, Yu-Wei Su, and Chung-Chou Shen. A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi. In *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, pages 46–51, nov. 2007.

[15] Jie Liu, Feng Zhao, Jeff O'Reilly, Amaya Souarez, Michael Manos, Chieh-Jan Mike Liang, and Andreas Terzis. Project genome: Wireless sensor network for data center cooling. Technical report, Microsoft Research and Microsoft Global Foundation Service, 2011.

[16] Pedro Jose Marron and Daniel Minder. *Embedded WiSeNts Research Roadmap*. PhD thesis, Universität Stuttgart, November 2006.

[17] S. Sasidharan, F. Pianegiani, and D. Macii. A protocol performance comparison in modular wsns for data center server monitoring. In *Industrial Embedded Systems (SIES), 2010 International Symposium on*, pages 213–216, july 2010.