Facultad de Ingeniería
Universidad de la República Oriental del Uruguay

WiDO

Domótica Inalámbrica

Matías Addiego
Martín Brindisi
Sebastián Olmedo
Andrés Renaud

Tutor
Juan Pechiar

Montevideo, Uruguay
Agosto de 2008

Agradecimientos

Si bien este proyecto requirió mucho trabajo y esfuerzo por parte de los integrantes del grupo, la tarea hubiera sido aún mucho más ardua sin la buena voluntad de varias personas que nos ayudaron a solventar los inconvenientes encontrados. Aquí nuestro modesto reconocimiento.

En primer lugar a nuestro tutor Juan Pechiar por ser nuestro guía en la gestión académica, por su disponibilidad para resolver nuestras dudas y el apoyo de su experiencia a la hora de tomar decisiones.

A Sebastián Fernández por encaminarnos en el diseño y construcción de los dispositivos de RF.

A la gerencia de Advanced Circuits (EEUU) por el descuento otorgado, ya que sin en él hubiera sido muy difícil construir los prototipos bajo las especificaciones requeridas.

A la gente de Armados Electrónicos (Argentina) por el buen trabajo realizado en la soldadura de las placas.

A Carlos Pechiar por la ayuda brindada en la gestión de la soldadura de las placas.

A CCC por su postura de apoyar proyectos de estudiantes, y en especial a Stefano Ghiardo por ayudarnos a resolver rápidamente inconvenientes encontrados en el montaje de las placas.

Y sobre todo, un muy especial agradecimiento a nuestras familias, por el apoyo incondicional que nos sostuvo siempre firmes en nuestros objetivos, no sólo a lo largo de este proyecto sino en toda la carrera.

Tabla de contenidos

1. Introducción 12
1.1. Descripción del proyecto 12
1.1.1. Objetivos 12
1.1.2. Motivación 13
1.1.3. Requerimientos 13
1.2. Estructura de la documentación 14
2. Planteo del problema 15
2.1. Contexto del proyecto 15
2.1.1. Características de la aplicación 15
2.1.2. Análisis de las posibilidades 17
2.2. Solución propuesta 18
3. Descripción general del sistema 23
3.1. Arquitectura del sistema 23
3.2. Integrantes del sistema 26
3.2.1. Dispositivo base 26
3.2.2. Dispositivo remoto 27
3.3. Flujo básico de comunicaciones 28
4. Características de la comunicación inalámbrica 30
4.1. Introducción 30
4.2. Banda de frecuencias utilizada 30
4.2.1 Comportamiento de ondas de radio según banda de frecuencia 32
4.2.1.1 Propagación línea vista 33
4.2.1.2 Propagación en ambientes con obstáculos 33
4.2.2 Fundamentos de la elección 35
4.3. Técnicas de Espectro Ensanchado 37
4.3.1 Espectro Ensanchado por Secuencia Directa 37
4.3.2 Espectro Ensanchado por Salto de Frecuencia 39
4.3.3 DSSS vs. FHSS, fundamentos de la elección 41
4.3.4 Elección 44
4.4 Funcionamiento de FHSS 45
4.4.1 Características de Hardware utilizado 45
4.4.2 Implementación del algoritmo 45
4.4.2.1 Sincronismo 45
4.4.2.2 Asociación a la red 46
4.4.3 Mejoras futuras 46
4.5 Resumen 46
5. Protocolo de comunicación WiDO 1.0 47
5.1. Definición de las características del protocolo 47
5.1.1. Retransmisión de paquetes 48
5.1.2. Inundación de la red 50
5.1.3. Confirmación de recepción de mensajes 51
5.1.4. Acceso al medio 52
5.1.5. Resumen 56

5.2. Aspectos generales del protocolo 57
5.2.1. Inundación controlada 57
5.2.2. CSMA 60
5.2.3. FHSS y sincronismo 60
5.2.4. Ventajas 60
5.3. Descripción del protocolo 61
5.3.1. Modelo de capas 61
5.3.2. Capa Física 62
5.3.2.1. Manejo del transceiver 62
5.3.2.2. Detección de los niveles de energía 62
5.3.2.3. Indicación de canal libre 63
5.3.2.4. La elección del canal 63
5.3.2.5. Transmisión y recepción de datos 64
5.3.2.6. Trama de capa de física 64
5.3.3. Capa MAC 65
5.3.3.1. Sincronización de los dispositivos 65
5.3.3.2. Mecanismo CSMA 65
5.3.3.3. Trama de Capa MAC 66
5.3.4. Capa de Red 66
5.3.4.1. Direccionamiento de paquetes 66
5.3.4.2. Control de difusión 67
5.3.4.3. Control de difusión básico 67
5.3.4.4. Control de difusión mediante dirección destino 68
5.3.4.5. Retransmisión de paquetes 69
5.3.4.6. Trama de capa de red 69
5.3.5. Capa de aplicación 70
5.3.5.1. Configuración de los parámetros de red 70
5.3.5.2. Asociación automática de dispositivos 76
5.3.5.3. Mensajes de sincronismo 77
5.3.5.4. Confirmación de recepción de mensajes 77
5.3.5.5. Trama de capa de aplicación 78
5.3.6. Trama completa del protocolo WiDO 1.0 79
6. Descripción del hardware 80
6.1. Introducción 80
6.2. RFB 82
6.2.1. Características del diseño 83
6.2.2. Elección de transceiver 84
6.2.3 Elección del microcontrolador 86
6.2.4 Interconexión Transceiver - Microcontrolador 87
6.2.5 Antena On board y adaptación de impedancia 90
6.2.6 Conversores y adaptadores de señales 93
6.2.6.1 MAX 485 93
6.2.6.2 MAX 232 94
6.2.7 Alimentación y reguladores de voltaje 96
6.2.8 Construcción del impreso 103
6.2.8.1 Introducción 103
6.2.8.2 Pasos para la implementación 104
6.3 Dispositivo Base 107
6.3.1 Composición de la Base 107
6.3.2 Introducción RCM 3700 108
6.3.3 Estructura y especificaciones de RCM 3700 110
6.3.4 Placa de expansión para RCM 3700 113
6.4 Fuente de alimentación 115
6.4.1. Introducción 115

6.4.2. Especificación de la fuente ... 116
6.4.3. Diseño ... 118
6.4.3.1 Análisis de simulaciones ... 119
6.4.4. Pruebas ... 123
6.4.5. Construcción del impreso ... 124
7. Descripción del software ... 126
7.1. RFB. ... 126
7.1.1 Hardware ... 127
7.1.2 Entorno de desarrollo ... 128
7.1.2.1 Compilador ... 128
7.1.2.2 Programador ... 130
7.1.3 Organización del Firmware ... 130
7.2.1 Capa Física ... 131
7.2.1.1 Comunicación micro-transreceptor ... 131
7.2.1.2 FHSS y sincronismo ... 133
7.2.1.3 Servicio para la asociación automática de dispositivos ... 134
7.2.1.4 Medición de Potencia RF ... 134
7.2.1.5 Generación de Números Aleatorios y tiempos de espera ... 135
7.2.2 Capa MAC ... 136
7.2.2.1 Sincronismo ... 136
7.2.2.2 Control de Acceso por Detección de Portadora ... 137
7.2.2.3 Control de integridad ... 138
7.2.3 Capa de Red ... 138
7.2.3.1 Generación de tramas ... 138
7.2.3.2 Control de Difusión ... 138
7.2.3.3 Configuración de los parámetros de la red ... 139
7.2.4 Capa de Aplicación ... 139
7.2.4.1 Reconocimiento automático del dispositivo ... 139
7.2.4.2 Asociación de dispositivos a la red ... 140
7.2.4.3 Iniciar tramas de sincronismo ... 140
7.2.4.4 Transmisión a través de puerto serie ... 141
7.2.4.5 Configuración de parámetros ... 141
7.2.5 Flujo del programa principal ... 142
7.2.5.1 Banderas ... 142
7.2.5.2 Tareas ... 143
7.2.5.3 Arquitectura del software ... 144
7.2.5.4 Modificaciones futuras ... 149
7.2.6 Conclusiones ... 149
7.3. Desarrollo en Rabbit RCM3700 ... 150
7.3.1 Introducción ... 150
7.3.2 Diseño ... 151
7.3.2.1 Gestión de la red inalámbrica ... 151
7.3.2.2 Mantener estado de la red inalámbrica ... 153
7.3.2.3 Servidor ... 157
7.3.2.4 Protocolos: Rabbit-RFB y Cliente-Rabbit ... 161
7.3.3 Implementación ... 164
7.3.3.1 Plataforma utilizada para desarrollo en Rabbit ... 164
7.3.3.2 Estructura Conexión ... 168
7.3.3.3 Flujo principal ... 169
7.3.3.4 Descripción de bloques ... 170
7.3.3.5 Definición de constantes ... 182
7.3.4 Depuración y validación del software ... 184

Referencias ... 187

Anexos: A. Análisis del proyecto 190
A.1 Introducción 190
A.2 Planificación inicial 190
A.3 Resultados obtenidos 195
A.4 Causas de los desvíos en la planificación 196
A.5 Conclusiones y trabajo a futuro 199
B. Obtención de archivos Gerber 200
C. Programador del Rabbit 203
D. Placa de expansión de Rabbit 206
E. Lista de componentes de la RFB 209
F. Análisis de costos 211
G. Contenido del CD 216

Resumen

En la actualidad existe un creciente interés por buscar distintas alternativas para sustituir las comunicaciones cableadas en distintos tipos de sistemas. En especial, esto es muy deseado en sistemas de control doméstico (área más conocida como Domótica). Una de las principales razones es que el hecho de no implementar las comunicaciones mediante líneas de cableado dedicadas a las mismas, trae como ventaja evitar los costos de instalación, de mantenimiento y de la compra misma del cable utilizado, lo cual en la gran mayoría de los casos determina una sensible disminución en los costos totales del sistema desarrollado.

Por otra parte, utilizar las distintas tecnologías inalámbricas en sistemas de comunicaciones, es algo muy frecuente en la actualidad. Esto es así debido principalmente al creciente desarrollo de la tecnología inalámbrica, con lo cual se tienen un gran número de diferentes opciones y cada vez menores costos al momento de utilizar estas tecnologías. Otra razón importante por la cual cada vez existe más inclinación a utilizar sistemas que implementen comunicación inalámbrica, es que éstas han ganado la aprobación de gran parte de los desarrolladores de estos sistemas, además de que existen variedad de desarrollos de tecnología inalámbrica muy robustos y confiables.

En este trabajo se propone una solución al problema de sustituir la comunicación por cable de un sistema de control doméstico, utilizando comunicación inalámbrica. Se realiza un estudio de las diferentes posibilidades manejadas, así como también se exponen los fundamentos de las decisiones adoptadas al momento de diseñar el sistema.

La solución propuesta consiste de una red de dispositivos inalámbricos, la cual provee un enlace transparente para las comunicaciones del cliente. El desarrollo de este sistema implicó el diseño completo del hardware necesario, partiendo de un microcontrolador y un chip transmisor de radiofrecuencia, así como también el diseño del software responsable de llevar a cabo el protocolo de comunicación inalámbrica. A su vez, este protocolo fue especificado completamente, para cada una de sus capas, aunque cabe destacar que fue inspirado en los estándares inalámbricos estudiados.

El presente documento pretende detallar específicamente todo lo realizado por este grupo de trabajo, excluyendo todo lo referente a detalles técnicos de los distintos componentes de hardware y del software de terceros utilizados en y durante este desarrollo; ya que ante la necesidad de profundizar en algún detalle especifico, se puede consultar las correspondientes hojas técnicas de los respectivos fabricantes, las cuales son muy completas.

Capítulo 1

Introducción

En este capítulo se presentan los objetivos del proyecto, los requerimientos con los que se debe cumplir y la motivación para el desarrollo del mismo. Se incluye también en el presente capitulo la estructura del documento.

1.1. Descripción del proyecto

1.1.1. Objetivos

La finalidad del proyecto es diseñar e implementar una red de comunicaciones para un sistema de control domestico. Dicho sistema implementa sus comunicaciones a través de un bus específicamente dedicado para esto y lo que se pretende es que la red diseñada lo realice de otra forma que no involucre el cableado de un bus de comunicaciones. Se busca poder brindarle al sistema del cliente un enlace confiable para lograr la comunicación con los dispositivos que utiliza.

Si bien el sistema a diseñar será optimizado para la aplicación específica de control doméstico de baja tasa de datos, se pretende que su eficiencia y velocidad en las comunicaciones permita la posibilidad de ser utilizado en escenarios más exigentes.

1.1.2 Motivación

La principal motivación para el desarrollo de este trabajo, es la de eliminar el cableado de comunicaciones que requiere un sistema de control doméstico de este tipo.

Evitar el cableado trae como ventajas el hecho de que se torna sensiblemente más práctica la instalación del sistema, así como también el agregado de dispositivos. Además le brinda movilidad al sistema y presenta menos costes de instalación y mantenimiento con respecto a una red cableada convencional.

1.1.3 Requerimientos

El sistema implementado debe cumplir con los siguientes requerimientos:

- Debe brindar soporte para aplicaciones bajo el protocolo Modbus.
- La red debe ser controlada mediante una PC, la cual será el punto de acceso a la misma.
- Los dispositivos agregados a la red deben ser reconocidos automáticamente al momento de ser instalados sin necesidad de intervención alguna (*Plug & Play*).
- Se debe llevar registro en todo momento de los dispositivos presentes en la red.
- Se deben proveer mecanismos para evitar interferencias y no permitir la recepción de mensajes entre dos redes cercanas similares.
- El sistema debe seguir en su correcto funcionamiento aunque se produzca la rotura de alguno de sus dispositivos (a excepción del dispositivo base).
- Los dispositivos remotos deben ser alimentados mediante la red eléctrica doméstica, no siendo aceptable el uso de pilas, baterías, etc.
- En el caso de producirse un corte en la alimentación de todo el sistema, se debe volver al funcionamiento normal, al momento de ser alimentado nuevamente, sin necesidad de intervención alguna.
- Se debe asegurar una distancia mínima a cubrir en la comunicación entre dos dispositivos, de manera de poder brindar cobertura en una aplicación estándar para un domicilio u oficina.

1.2 Estructura de la documentación

La estructura del documento es la siguiente:

- *Capítulo 1*, "Introducción": Se presentan los objetivos, motivación y requerimientos del proyecto.

- *Capítulo 2*, "Planteo del problema": Se expone el marco en el cual se desarrolla el proyecto, las distintas posibilidades para resolver el problema y se presenta la solución propuesta.

- *Capítulo 3*, "Descripción general del sistema": Se describen las principales características del sistema, junto con su arquitectura y los elementos que lo componen.

- *Capítulo 4*, "Características de la comunicación inalámbrica":

- *Capítulo 5*, "Protocolo de comunicación WiDO 1.0": Se describe el protocolo diseñado para la comunicación inalámbrica y se presentan los fundamentos que llevaron a su definición.

- *Capítulo 6*, "Descripción del hardware": Se describe el *hardware* utilizado y el desarrollado.

- *Capítulo 7*, "Descripción del software": Se explica el *software* desarrollado.

Capítulo 2

Planteo del problema

En este capítulo se expone la solución planteada para cumplir con el objetivo del proyecto, así como también un previo análisis del marco en el cual se desarrolla el mismo y de las posibilidades que se tienen al alcance para implementar dicha solución.

2.1 Contexto del proyecto

2.1.1 Características de la aplicación

El cliente, para el cual se desarrolló el proyecto, implementa un sistema de control doméstico, centralizado en una PC, mediante la cual se accede a los distintos dispositivos. Estos pueden ser sensores de temperatura, de humedad, luminosidad, etc., o actuadores, mediante los cuales se encienden o apagan luces, equipos de aire acondicionado, etc. Actualmente la manera en la que se logra la comunicación con dichos dispositivos es mediante un bus RS485, común a todos, conectado al puerto serie de la PC, y mediante el cual se implementa el protocolo Modbus.

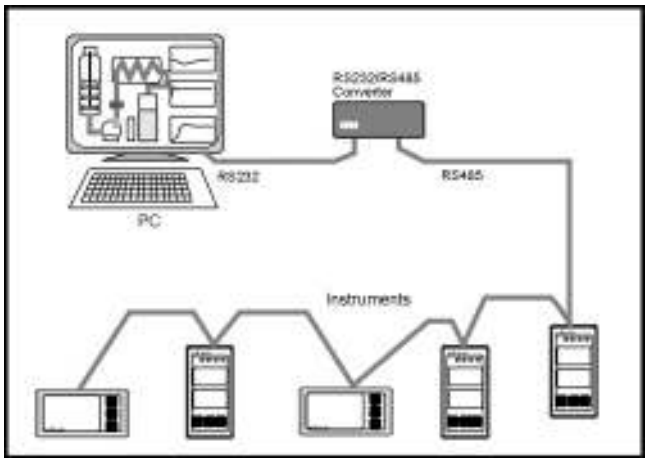


Figura 2.1: Sistema del cliente

Como se expuso en el capítulo introductorio, la principal necesidad del cliente y motivación para el desarrollo del proyecto consistió en lograr eliminar el bus de comunicaciones existente en las instalaciones, sin perder la performance actual de dicha comunicación. Por lo que fue necesario entonces, determinar las principales características que presentaba el sistema, de forma de diseñar la red óptima para el mismo.

Un aspecto importante a destacar para esta aplicación, y en general para este tipo de sistemas, es que el tráfico de datos es bajo, es decir que no requiere de una gran cantidad de transmisiones en pequeños intervalos de tiempo.

Dado que la red será utilizada para controlar actuadores y sensores, no se requiere demasiado ancho de banda, las tramas de control son pequeñas y la mayor parte del tiempo no se transmitirá nada. En los momentos que se requiera transmitir puede ocurrir que se den picos de demanda de datos de los dispositivos, pero son casos atípicos.

Otro aspecto muy importante para destacar es la necesidad de mantener comunicación con todos los dispositivos en todo momento, es decir que ningún actuador o sensor pueda quedar incomunicado.

El número de nodos de la aplicación es pequeño y se contemplara un caso máximo de hasta 50 nodos, aunque este caso se prevé solucionar implementando varias subredes más pequeñas (lo cual es explicado en capítulos posteriores). Además los dispositivos se encuentran en ambientes domésticos (poca interferencia) y la distancia entre dos de ellos no supera los 50m, donde se asume que este caso ocurre en espacios abiertos.

2.1.2 Análisis de las posibilidades

En una primera etapa se plantearon dos opciones para sustituir la comunicación por cable; utilizar la red eléctrica como medio de comunicación o implementar dicha comunicación mediante una red inalámbrica.

La primera opción, de utilizar la instalación eléctrica existente en el establecimiento en cuestión para transportar los datos, es muy usada para aplicaciones de control doméstico. Como exponente más representativo se encuentra el estándar internacional X10, el cual utiliza los cruces por cero de la señal de alterna, ya sea de 50 Hz o 60 Hz de frecuencia, para transmitir señales codificadas con una frecuencia de portadora de 120 kHz.

En el caso de la segunda opción manejada, la de utilizar enlaces inalámbricos de radiofrecuencia para implementar la comunicación, existen bandas de frecuencia libres, es decir que esta permitido su uso sin necesidad de obtener licencias, siempre y cuando se cumpla con sus correspondientes restricciones de potencia y de utilización del espectro.

En cuanto a la utilización de la red de suministro eléctrico, este método tiene algunas desventajas significativas. Uno de los principales problemas encontrados en utilizar este tipo de comunicación es que es muy sensible a interferencias y atenuaciones provocadas por distintos motivos. A modo de ejemplo, sucede que muchos de los dispositivos electrónicos modernos, como TV`s, computadoras, etc., presentan un "camino" de baja impedancia para las señales de alta frecuencia, como las utilizadas en estas técnicas, por lo que hace difícil la comunicación con otro dispositivo conectado a la red. También se ha observado frecuentemente que se generan espurios en las señales de control debido a interferencias provocadas por TV`s, que pueden alterar el funcionamiento deseado. Otro ejemplo importante, es el hecho de que se ha comprobado que la existencia de algún diferencial conectado a la red, tiene el efecto de atenuar las señales de las frecuencias utilizadas, así como también se ha notado que existe una excesiva atenuación en las señales entre fases, para el caso de instalaciones de suministro trifásico.

Si bien existen varias técnicas para disminuir el efecto de estos problemas, como la utilización de repetidores, amplificadores y filtros, siguen siendo inconvenientes que perjudican la performance y robustez de estos sistemas.

Otra desventaja importante en sistemas de este tipo es que se logra una baja tasa de datos, poniendo como ejemplo al protocolo más utilizado, el protocolo X10, al cual le toma 750 milisegundos transmitir un comando, se puede observar este hecho.

En el caso de la comunicación mediante radiofrecuencia también existen problemas de interferencia de señales y de atenuaciones producidas principalmente por los obstáculos

físicos, como paredes, techos, muebles, etc., existentes en el hogar u oficina. En particular el problema de las interferencias no es menor, debido a la creciente utilización de dispositivos inalámbricos de diversa índole.

Es importante destacar que si bien los problemas en cuanto a las interferencias y atenuaciones se encuentran presentes en ambas opciones, se llegó a la conclusión de que para el caso de la comunicación por radiofrecuencia existe más flexibilidad para solucionar estos problemas, debido principalmente a que existen más variadas y efectivas técnicas para este propósito.

En base a lo expuesto en esta sección se optó por utilizar comunicación por radiofrecuencia principalmente por dos motivos, uno es la posibilidad de implementar un protocolo de mayor tasa de datos, al no estar limitado a actuar en los cruces por cero de la onda de 50 Hz, como es el caso de X10, y el otro motivo es el comentado anteriormente, de que existe mayor solvencia para solucionar los problemas de interferencias y de atenuación en las señales transmitidas.

2.2 Solución propuesta

En base a todo lo analizado a lo largo de este capítulo se definió a grandes rasgos la solución a implementar.

Comunicación por radiofrecuencia:

En primer lugar se decidió por utilizar radiofrecuencia para resolver la comunicación. Es decir que se optó por realizar una red inalámbrica como enlace de punta a punta para el sistema del cliente, como se observa en la figura 2.3.

Dispositivos interfaces:

La red inalámbrica implementada es transparente para el sistema del cliente, y cumple la función que cumplía anteriormente el cable. Por lo que para sustituir el mismo se decidió utilizar dispositivos que actuaran de interfaz entre la comunicación cableada y la red inalámbrica, colocados en cada elemento del sistema, es decir, en la PC cliente, en cada sensor y en cada actuador.

Dichos dispositivos diseñados se componen básicamente de un transceiver, con su correspondiente antena, y un microcontrolador, el cual interactúa con el transceiver. Las principales características de estos integrados, así como los fundamentos de la elección de los mismos se despliegan en la sección 6.2.

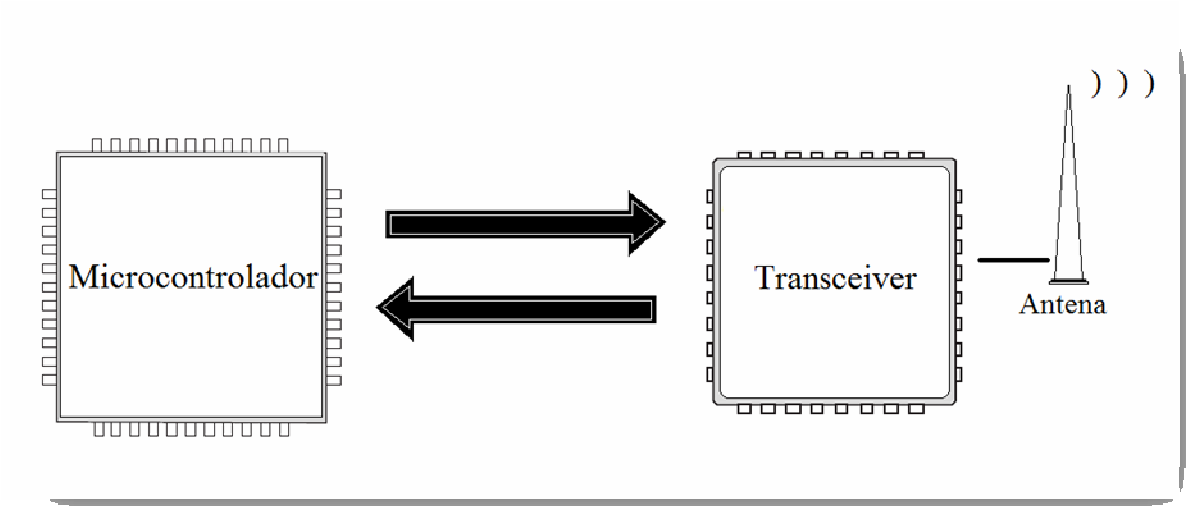


Figura 2.2: Estructura del dispositivo interfaz.

Banda de frecuencias utilizadas:

Una vez definido el hecho de utilizar comunicación inalámbrica, fue necesario definir cómo implementarla, lo cual consistió en estudiar las bandas libres de frecuencias disponibles y buscar un transceiver que funcionara en alguna de estas frecuencias.

Dentro de las bandas de frecuencia de uso libre, unas de las más utilizadas en variadas aplicaciones son las bandas ISM (*Industrial, Scientific and Medical*), las cuales son bandas reservadas internacionalmente para el uso del espectro electromagnético de radiofrecuencia en el área industrial, científica o médica (ver tabla 2.1). Si bien, originalmente su uso era reservado para aplicaciones en las áreas mencionadas anteriormente, en la actualidad se ha popularizado la utilización de estas bandas en todo tipo de comunicaciones.

El uso de estas bandas de frecuencia está abierto a todo el mundo sin necesidad de licencia, respetando las regulaciones que limitan los niveles de potencia transmitida. Este hecho fuerza a que este tipo de comunicaciones tengan cierta tolerancia frente a errores y que utilicen mecanismos de protección contra interferencias, como técnicas de ensanchado de espectro, como por ejemplo DHSS (*Direct Sequence Spread Spectrum*) y FHSS (*Frequency Hopping Spread Spectrum*).

BANDA DE FRECUENCIAS ISM	FRECUENCIA CENTRAL
6.765 – 6.795 MHz	6.780 MHz
13.553 – 13.567 MHz	13.560 MHz
26.957 – 27.283 MHz	27.120 MHz
40.66 – 40.70 MHz	40.68 MHz
433.05 – 434.79 MHz	433.92 MHz
902 – 928 MHz	915 MHz
2.4 – 2.5 GHz	2.450 GHz
5.725 – 5.875 GHz	5.800 GHz
24.00 – 24.25 GHz	24.125 GHz
61.0 – 61.5 GHz	61.25 GHz
122 – 123 GHz	122.5 GHz
244 – 246 GHz	245 GHz

Tabla 2.1: Bandas ISM

En la sección 4.1 se muestra en detalle los fundamentos para la elección de la banda de frecuencia utilizada.

Protocolo de comunicación RF:

Para finalizar el planteo de la solución se necesitó diseñar un protocolo de comunicación para dicha red que fuera lo suficientemente confiable como para estar en pie de igualdad con la robustez que presenta la comunicación implementada mediante cable.

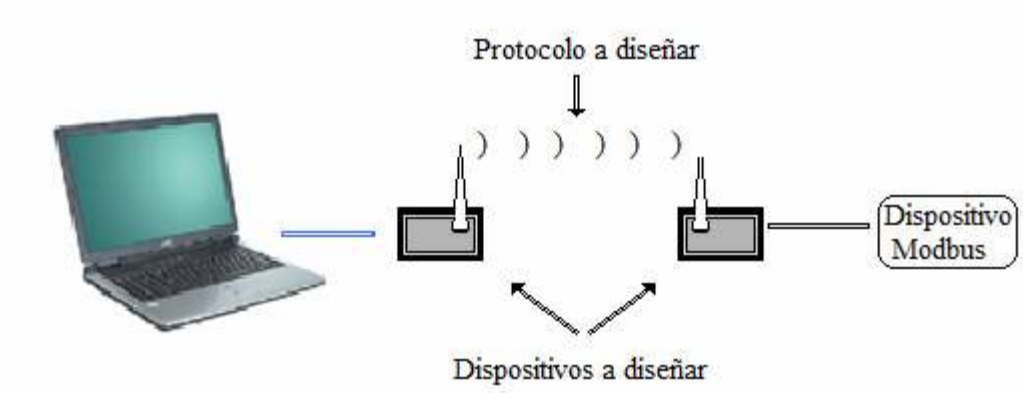


Figura 2.3: Solución propuesta.

Para definir el protocolo a utilizar para la red, se manejaron dos opciones; por un lado se evaluó la posibilidad de implementar alguno de los estándares existentes utilizados para redes inalámbricas, mientras que en contra posición a esto se manejó también la posibilidad de desarrollar completamente el protocolo.

Se estudiaron entonces los distintos estándares inalámbricos existentes, en particular los utilizados para redes WPAN (Wireless Personal Area Network) de baja tasa de datos, ya que dentro de esta categoría se encontraría la red desarrollada, dejando por fuera de este estudio a los utilizados en redes WLAN (Wireless Local Area Network).

Durante este estudio se observó que existe una fuerte tendencia en el mercado de redes PAN de baja tasa de datos, de utilizar el estándar IEEE 802.15.4 a nivel de capa física y de enlace lógico. Por lo que entonces se profundizó en el estudio de este estándar, analizando sus principales características y los mecanismos que utiliza. Luego se estudiaron diferentes estándares especificados a nivel de capa de red, que implementan este estándar IEEE 802.15.4 en su capa física y de enlace. Entre estos se analizó especialmente el estándar Zigbee, el cual esta orientado a la optimización del consumo de energía y apunta a ser una tecnología mas económica que otras, como por ejemplo Bluetooth. Cabe destacar que Zigbee es del tipo de red Mesh, por lo cual mediante el estudio de Zigbee se observó la implementación de una red Mesh, de las cuales se analizaron varios de sus conceptos.

Finalmente, luego de este estudio se decidió por implementar completamente el protocolo RF debido a algunos puntos importantes. En primer lugar, el hecho de especificar completamente el protocolo, fue considerado como un aspecto muy enriquecedor

académicamente, ya que se maneja en detalle la implementación de cada una de sus capas y se enfrenta de cerca con distintos tipos de problemas que esto presenta. Por otro lado, también se consideró como un aspecto importante a favor de especificar completamente el protocolo, tener un completo dominio de la implementación; es decir, al diseñar completamente el protocolo, se tiene la libertad de adaptar el mismo a cualquier posible cambio que se presente necesario, así como también se puede buscar la optimización del diseño para la aplicación especifica del cliente.

Capítulo 3

Descripción general del sistema

Este capitulo describe en rasgos generales las características del sistema, su arquitectura, elementos que lo componen y como interactúan entre si. También se incluye un ejemplo representativo del flujo de las comunicaciones.

3.1 Arquitectura del sistema

El sistema diseñado se compone de dos tipos de dispositivos con funcionalidades y responsabilidades distintas, los cuales denominamos *dispositivo base* y *dispositivo remoto*. En rasgos generales, el *dispositivo base* es el encargado de actuar de interfaz entre la PC, en donde se encuentra la aplicación del cliente y la red inalámbrica, mientras que los *dispositivos remotos*, se encargan de resolver la interfaz entre la red inalámbrica y los dispositivos finales, tanto sensores como actuadores. En la siguiente sección (sección 3.2) se detallan las características de ambos tipos de dispositivos.

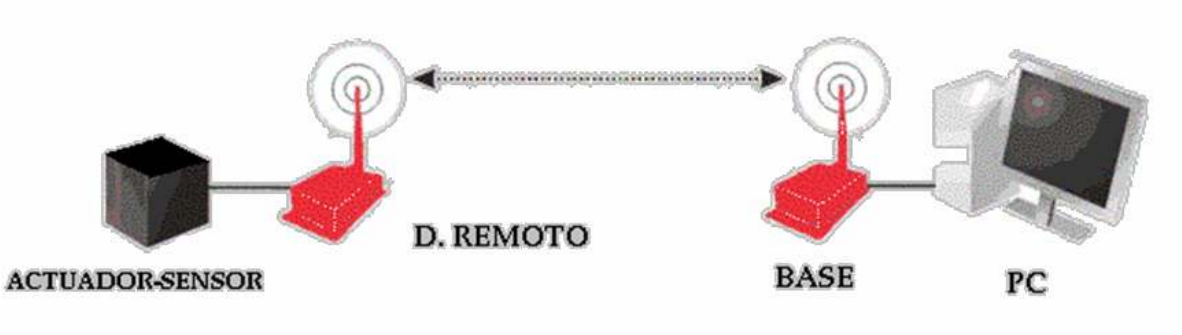


Figura 3.1: Representación básica del sistema.

La red esta compuesta de un solo dispositivo base y de una cantidad variable de dispositivos remotos según lo requiera la aplicación. Básicamente se deberá colocar un dispositivo remoto por cada actuador o sensor existentes en la red del cliente, aunque está previsto que en determinados casos, de ser conveniente, se conecten más de un actuador o sensor a un mismo dispositivo remoto. Esta situación se representa en la figura 3.2.

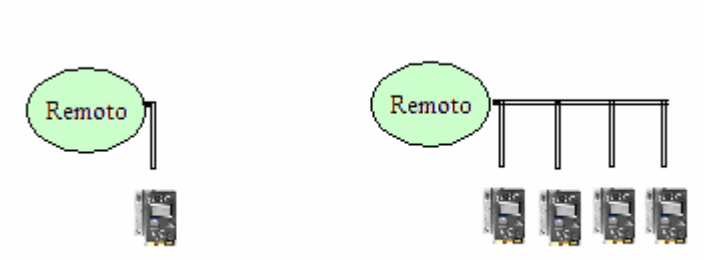


Figura 3.2: Posibilidad de conexión múltiple.

La razón por la cual se utiliza un solo dispositivo base, se debe a que en el caso de que por determinada razón se precisen arquitecturas más complejas, se tomará a la red como una composición de sub-redes, las cuales estarán asociadas a una base y comunicadas entre

ellas a través de las mismas y en donde cada sub-red presentará la estructura básica descripta en este capítulo.

Por lo tanto a partir de definir esta estructura básica para una red, se pueden implementar estructuras más complejas mediante composición de dicha estructura básica, en donde por cada sub-red habrá un dispositivo base y las sub-redes estarán interconectadas, utilizando distintas topologías según lo requiera el caso o las restricciones físicas existentes, mediante los dispositivos base.

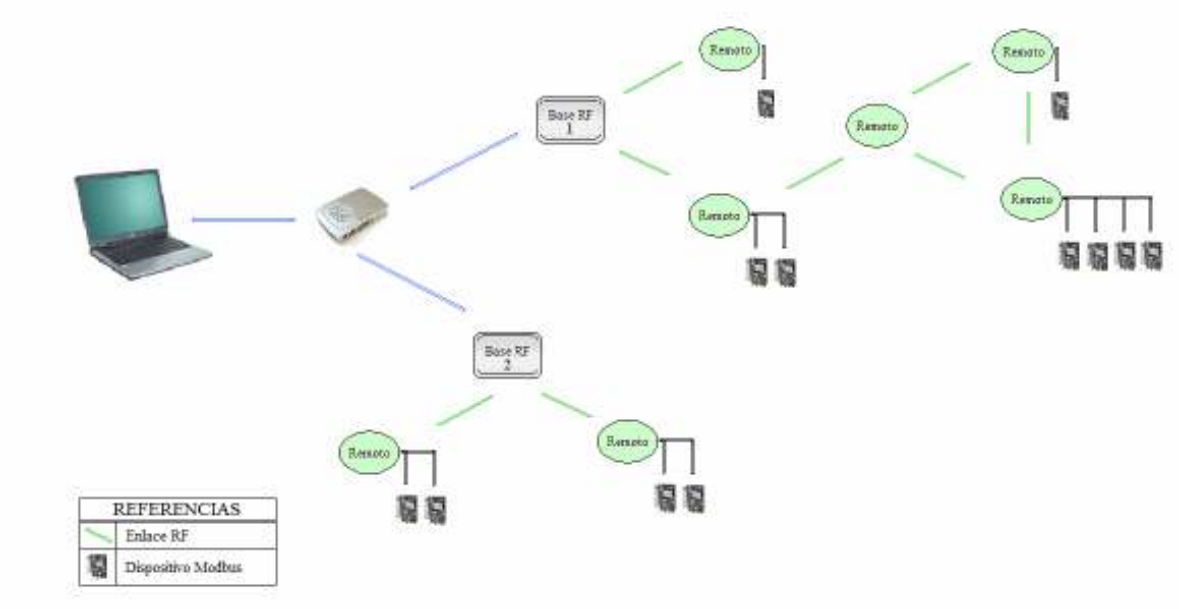


Figura 3.3: Ejemplo de implementación de sub-redes.

3.2 Integrantes del sistema

Como se mencionó en la sección anterior, en la red funcionan dos tipos de dispositivos, los del tipo *dispositivo base* y del tipo *dispositivo remoto* (la PC cliente, los actuadores y los sensores, no son considerados como parte del sistema). Sus características y responsabilidades en la red son descriptas a continuación.

3.2.1 Dispositivo base

El dispositivo base es el encargado de actuar de interfaz con la PC del cliente y es conectado a la tarjeta de red de la PC, utilizando comunicación por protocolo Ethernet. Por otra parte es encargado también de la conectividad con la red inalámbrica mediante el protocolo diseñado para el sistema, el cual es explicado en el capítulo 5.

El dispositivo del tipo base se encarga de gestionar la red. Entre sus funciones, en lo que a esto respecta, cabe destacar que es quien da inicio a la red, mantiene registro en todo momento de los dispositivos que la integran, maneja la asociación y disociación de los dispositivos y mantiene el sincronismo de estos en la red. Además, posee tablas de mapeo de direcciones entre dispositivos remotos y dispositivos finales de modo de resolver las asociaciones entre estos.

Cada red tendrá una sola base, por lo cual se podría ver como que dicho dispositivo es el punto de entrada a la red inalámbrica.

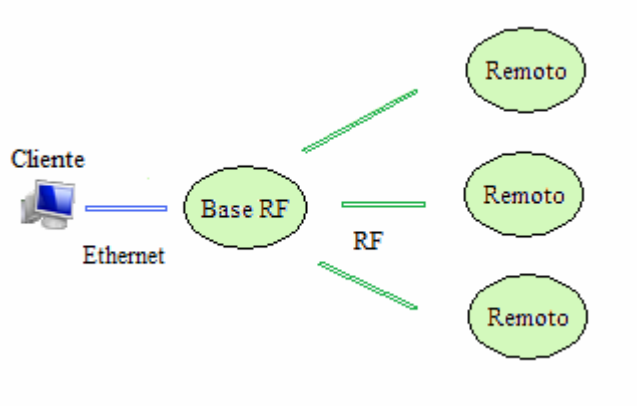


Figura 3.4: Diagrama de interfaces para dispositivo base.

3.2.2 Dispositivo remoto

El dispositivo remoto, al estar conectado directamente a uno o más dispositivos finales mediante un bus RS-485, actúa de interfaz para la comunicación con los mismos. Es decir, interpreta los paquetes provenientes de la red inalámbrica, de modo de generar nuevamente los paquetes Modbus destinados al dispositivo final.

Entre las funciones especiales que presentan los dispositivos remotos, cabe destacar que al momento de ser conectado a un dispositivo final (o mas de uno), inicia un proceso de búsqueda de la dirección de esclavo Modbus del dispositivo final, y una vez que lo obtiene inicia el procedimiento correspondiente de asociar a dicho dispositivo a la red y de obtener su propia identidad de red; o sea que se le asigna al dispositivo remoto una dirección de red y se le asocia con la dirección Modbus del dispositivo agregado al sistema.

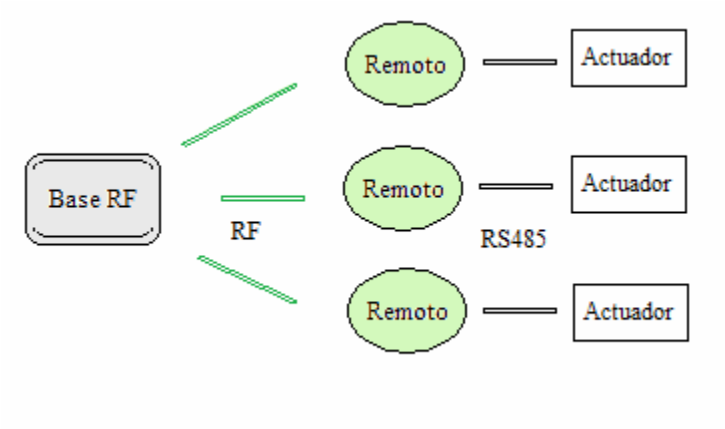


Figura 3.5: Diagrama de interfaces para dispositivo remoto.

3.3 Flujo básico de comunicaciones

A continuación se describe brevemente la dinámica del sistema para el caso de una comunicación punta a punta, es decir, debido a un evento generado desde la PC cliente hacia un actuador o sensor, los cuales son los dispositivos finales. Se explica este caso en particular porque además de ser el más frecuente, abarca todos los distintos modos de comunicación existentes en el sistema.

La situación más frecuente para el sistema es que el flujo de los mensajes sea debido a un evento originado desde la PC cliente, en la cual funciona la aplicación principal. Esta se comunica con el dispositivo base mediante Ethernet, enviándole entonces el mensaje, el cual generalmente es dirigido a algún dispositivo final, ya sea actuador o sensor. La base luego de recibir el paquete, lo envía a través de la red inalámbrica con destino al dispositivo remoto correspondiente, es decir el que tiene conectado al destinatario del mensaje, utilizando para esto el protocolo RF diseñado para dicha red. El dispositivo remoto, al recibir el mensaje se encarga de reenviar el mensaje hacia el dispositivo final utilizando el protocolo Modbus a través de un bus 485. Luego, dependiendo del contenido del mensaje, se sucede la secuencia inversa al enviarse una respuesta hacia la PC cliente.

Existen también mensajes originados por la PC cliente, que no son dirigidos a un dispositivo final, sino que son dirigidos al dispositivo base y tienen una finalidad de configuración del sistema. Los dispositivos remotos también son capaces de generar eventos, ya sean dirigidos hacia la base o hacia otro dispositivo remoto. Estos casos son expuestos con más detalle en el capitulo 5 de este documento.

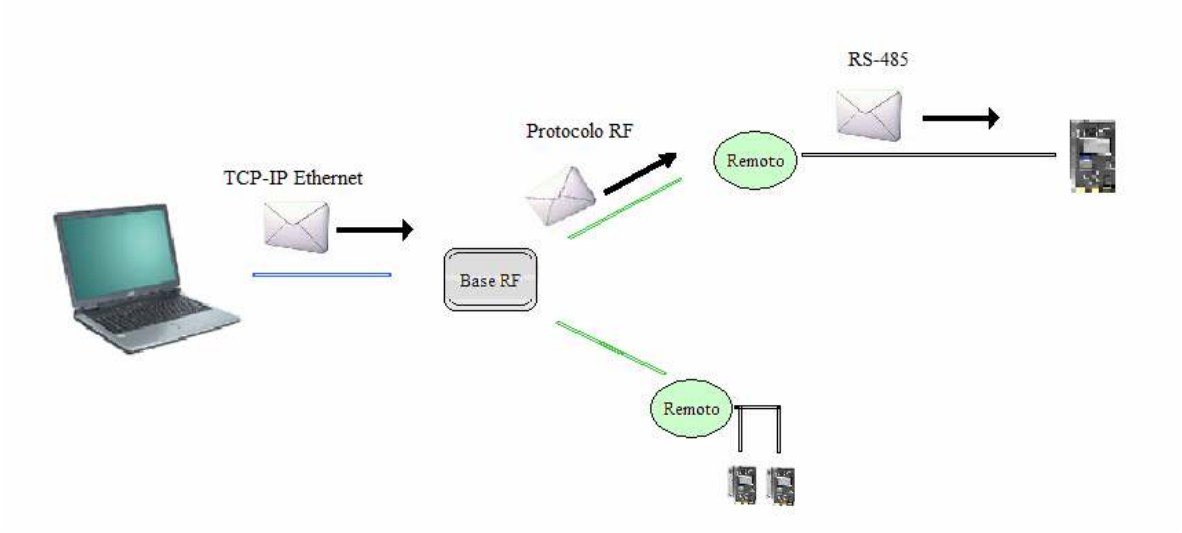


Figura 3.6: Diagrama de comunicación punta a punta del sistema.

Capítulo 4

Características de la comunicación inalámbrica

Este capítulo describe los aspectos de la comunicación inalámbrica que deben ser considerados para que el sistema propuesto cumpla con los objetivos de este proyecto. En particular describe los fenómenos físicos que debilitan las ondas de radio y la técnica de espectro ensanchado utilizada, así como los fundamentos en la elección de la misma.

4.1 Introducción

La transferencia de información a través del aire ha sido ampliamente estudiada y sus beneficios son bien conocidos, pero en los últimos años la modulación digital pasa-banda llevó la comunicación RF a otro nivel. Las ventajas de esta modulación impulsaron el desarrollo tecnológico que hoy nos permite encontrar circuitos integrados muy completos, con altas prestaciones, a precios relativamente bajos; muchos de ellos diseñados específicamente para optimizar el uso del espectro y el consumo de energía.

4.2 Banda de frecuencias utilizada

El uso de dispositivos inalámbricos es extensamente regulado a través del mundo. Cada país posee un departamento responsable de regular y ordenar el uso del medio físico. Para ello establecen normas que los dispositivos inalámbricos deben cumplir para permitir la coexistencia. En la mayoría de los países, no todos, se destina parte del espectro para uso

libre, uso “sin licencia”, el resto del espectro sólo puede ser usado obteniendo permiso o “licencia” para cada aplicación en particular.

La mayoría de los productos inalámbricos para aplicaciones industriales y comerciales de corto rango, utiliza el rango de frecuencias libres, esto evita las demoras, costos e inconvenientes que conlleva la obtención de una licencia. Estas bandas que no exigen licencia son conocidas también como bandas ISM (Industrial, Scientific & Medical). Muchos países poseen varias bandas ISM disponibles en distintas partes del espectro.

El espectro es dividido en bandas de frecuencias y cada banda es dividida en canales. Normalmente cada canal sostiene una única conexión inalámbrica en un mismo tiempo y lugar. Cada canal posee un ancho de canal que también es regulado.

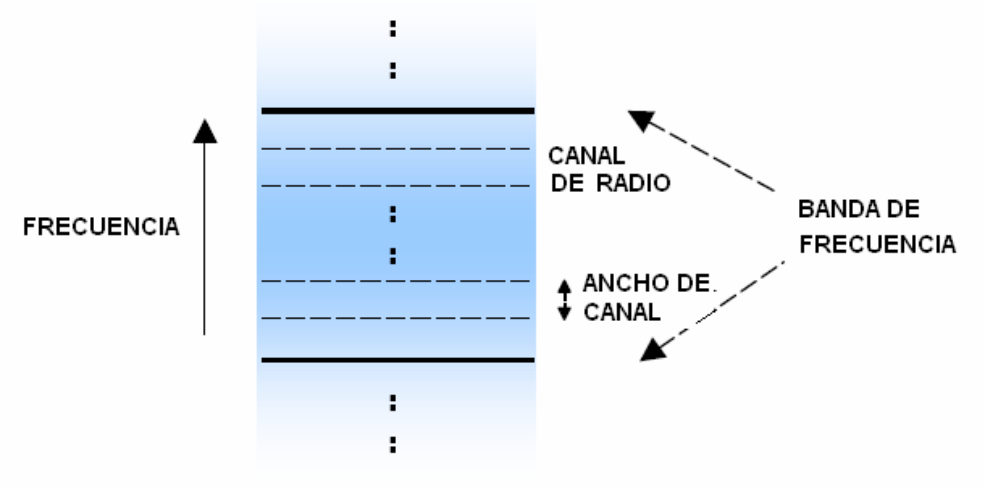


Figura 4.1: Estructura de la bandas de radio frecuencia [4.1].

Las bandas ISM más comunes para aplicaciones industriales y comerciales en las distintas regiones del mundo son las siguientes [4.1]:

- 220 MHz en China.
- 433 MHz en Europa y otros países.
- 869 MHz en Europa.
- 915 MHz en Estados Unidos y otros países.
- 2,4 GHz y 5,7 GHz en la mayoría de los países de mundo.

Lo que sigue es la determinación de la banda más apropiada para nuestra aplicación, por lo que es necesario estudiar las ventajas y desventajas de las mismas. El siguiente punto explica brevemente como la operación de los sistemas inalámbricos puede verse afectada por las características espaciales del entorno, según la banda de frecuencias en la que se opera. En particular se compararán las bandas 869/915 MHz y 2,4 GHz, que son las bandas internacionalmente utilizadas para la automatización de casas y edificios, así como en plantas industriales.

4.2.1 Comportamiento de las ondas de radio según la banda de frecuencia

La frecuencia de operación y en especial el ancho de canal afectan, en gran forma, la *performance* de los sistemas de radio. Canales de radio más anchos permiten mayores tasas de transmisión.

A frecuencias más altas el espectro relativo es mucho mayor y por lo tanto los canales son más anchos. Por esta razón las bandas de 2,4 y 5,7 GHz soportan tasas de datos mucho más altas que las bandas de frecuencias más bajas; sin embargo, estas bandas de frecuencias presentan mayores pérdidas en la propagación de las ondas. En la mayoría de los casos la comunicación en bandas altas no es confiable a través de largas distancias o con muchos obstáculos, como suelen ser, por ejemplo, los ambientes industriales y edificios.

4.2.1.1 Propagación en línea vista

Las ondas de radio que viajan a través del aire (u otro medio) experimentan pérdidas hasta el punto que la señal recibida es demasiado débil para que la información pueda extraerse de la señal modulada.

En Europa las normas establecen que los dispositivos de 2,4 GHz sólo pueden transmitir un máximo de 100 mW de potencia RF, y 500 mW en la banda de 869 MHz. Sin tomar en cuenta ningún otro factor, debido a la mayor potencia y una menor atenuación en 869 MHz se alcanzan distancias confiables 5 veces mayores que en 2,4 GHz. En Estados Unidos la máxima potencia RF regulada para las bandas de 2,4GHz y 915MHZ es 1W; sin embargo, debido a las pérdidas de propagación, en 2,4 GHz la máxima distancia confiable es de un 50% respecto a la banda más baja [4.1].

4.2.1.2 Propagación en ambientes con obstáculos

En edificios, plantas industriales y fábricas la transmisión en línea vista no es siempre posible por lo que la *performance* en 2,4GHz es aún peor, y las distancias confiables caen a un 10 - 20% respecto a bandas de frecuencias más bajas. El rendimiento de los sistemas de radio que operan en este tipo de ambientes es determinado por la habilidad de las ondas de radio para:

- Atravesar obstáculos
- Rodear obstáculos,
- Reflejarse en obstáculos

Atravesar Obstáculos

Al atravesar obstáculos las señales se debilitan y esta atenuación es mayor a medida que la frecuencia aumenta, por las pérdidas de propagación debido a este fenómeno es mucho mayor en las bandas de frecuencias más altas.

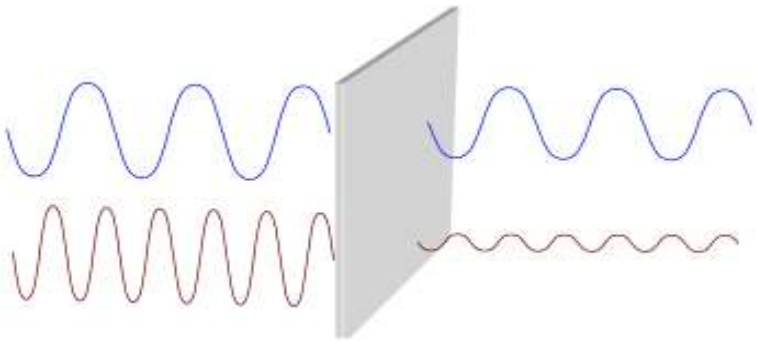


Figura 4.2: Frecuencias más altas sufren una mayor atenuación cuando atraviesan obstáculos [4.1].

Rodear Obstáculos

Las ondas viajan en línea recta, sin embargo, al igual que la luz, el "haz" de ondas de radio se difracta al chocar contra el borde de un objeto. El ángulo de difracción es mayor para frecuencias menores por lo que su capacidad de rodear obstáculos es mayor.

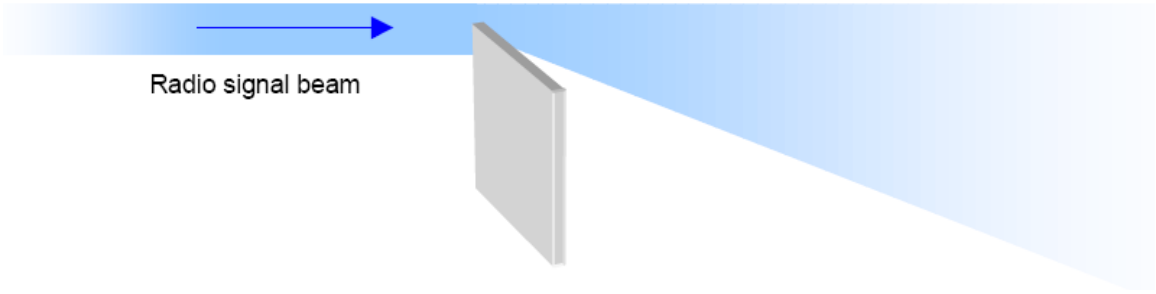


Figura 4.3: La habilidad de rodear obstáculos es mayor para frecuencias más bajas [4.1].

Reflejarse en obstáculos

Las ondas de radios son reflejadas por superficies densas como pueden ser las paredes de un edificio, o la infraestructura industrial. En la mayoría de los casos, en ambientes congestionados, la señal RF es reflejada varias veces antes de llegar al receptor. Cuando la onda es reflejada, parte la potencia RF es absorbida por el obstáculo reduciendo la fuerza de la señal. La atenuación de las ondas por reflexión también aumenta con la frecuencia.

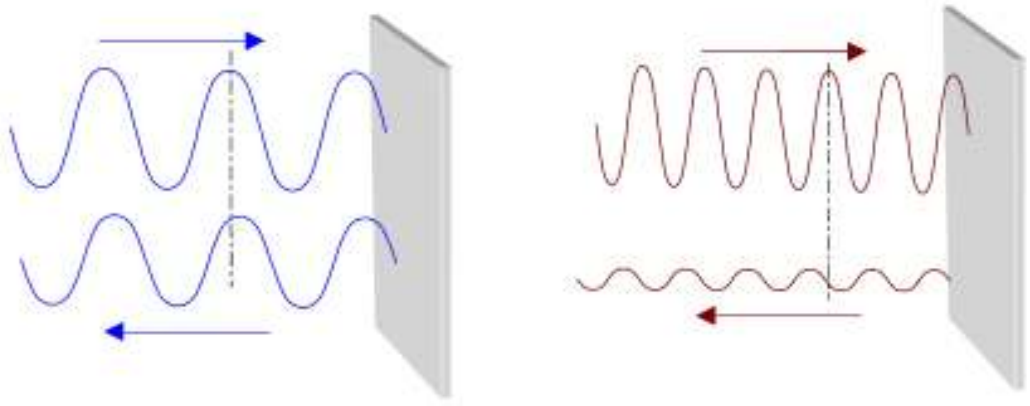


Figura 4.4: Las ondas de mayor frecuencia se atenúan más al reflejarse [4.1].

4.2.2 Fundamentos de la elección

Si bien la banda de 2,4GHz sólo llega al 10 - 20% de las distancias confiables que se logran con la bandas de 869/915 MHz en ambientes congestionados, si se busca una mayor velocidad transferencia 2,4 GHz es la mejor opción.

En 2,4 GHz se puede obtener tasas de transferencia de datos que superan ampliamente a las tasas obtenibles utilizando banda de frecuencias más bajas, debido a que los canales son más anchos. A modo ejemplo, la norma **IEEE 802.15.4** para LR-WPAN determina que las velocidades máximas alcanzables (utilizando las especificaciones que ellos establecen) para la banda de 2,4 GHz es 256 kbit/s mientras que utilizando la banda de 869 MHz sólo se llega a unos 20 kbit/s.

Además la banda de 2,4 GHz es también usada por tecnologías inalámbricas comerciales como WIFI, Bluetooth y Zigbee. Las ramificaciones de componentes para estas frecuencias están disponibles más rápidamente y a costos menores.

A continuación un estudio de las distancias confiables alcanzables, por la normativa europea, obtenidas de un artículo técnico de ELPRO Technologies [4.2].

- **Transmisión en línea vista:**

2,4 GHz, 100 mW	1 - 2 Km
869 MHz, 500 mW	4 - 8 Km

- **Transmisión a través de caminos altamente congestionados:**

2,4 GHz, 100 mW	10 - 100 m
869 MHz, 500 mW	100- 1000 m

Las variaciones en distancia dependen fuertemente del camino que atraviesan las ondas de radio. Las regulaciones del gobierno europeo permiten un máximo de 100 mW de potencia RF para 2,4 GHz y 500 mW para 869 MHz.

Elección

Las distancias alcanzables por los dispositivos en ambientes cogestionados como edificios y plantas industriales en 2,4 GHz son aceptables. El ofrecer una mayor tasa datos amplía el rango de aplicaciones posibles para los dispositivos diseñados. El hecho que el costo de los componentes sea menor debido al gran desarrollo que existe en esta frecuencia favorece la sustitución del cableado usando esta tecnología inalámbrica. Estas razones hacen que la banda más apropiada para la implementación de la comunicación inalámbrica de este proyecto sea la de 2,4 GHz.

4.3 Técnicas de Espectro Ensanchado

La banda de 2,4 GHz es una banda libre en casi todos los países del mundo y es compartida por muchos artefactos de uso doméstico e industrial (microondas, teléfonos inalámbricos, redes inalámbricas y otros) por esta razón los dispositivos que operan en esta banda deben ser capaces de lidiar con interferencias y al mismo tiempo dejar que otros equipos puedan operar en la misma banda. Para ello se desarrollaron las técnicas de **Espectro Ensanchado o Disperso, SS** (*Spread Spectrum*). Estas técnicas redistribuyen la potencia de la señal transmitida a través de un espectro mucho más ancho que el necesario para trasmitir los datos, logrando reducir los efectos del ruido y minimizando interferencias con otros dispositivos que funcionan en la misma banda de frecuencias.

Los gobiernos de la mayoría de los países regulan la operación de los dispositivos en las bandas de frecuencia libres. La **FCC** y la **ETSI**, en Estados Unidos y Europa respectivamente, son las entidades reguladoras en lo que respecta a las telecomunicaciones. Las cuales establecen los requerimientos mínimos para el uso eficiente de espectro de radio, permitiendo una mayor coexistencia de dispositivos. Esto se logra comúnmente mediante el uso de una de dos técnicas: **Espectro Disperso por Secuencia Directa, DSSS** (*Direct Sequence Spread Spectrum*) o **Espectro Disperso por Salto de Frecuencia, FHSS** (*Frequency Hopping Spread Spectrum*).

Los párrafos siguientes describen brevemente el funcionamiento ambas técnicas de SS, analizando ventajas y desventajas desde el punto de vista de la coexistencia de redes de dispositivos así como su rendimiento en ambientes ruidosos y con interferencias. Se concluirá cual es el mejor sistema para nuestro proyecto, teniendo en cuenta los costos que implica dicha elección y los tipo de aplicaciones en que se pretende usar los dispositivos diseñados.

4.3.1 Espectro Ensanchado por Secuencia Directa

Los sistemas DSSS emplean una señal dispersa o *spreading*, que consiste en una secuencia pseudo aleatoria de pulsos positivos y negativos transmitidos a una tasa muy alta. El ancho de banda de la señal dispersa es mucho más grande que el necesario para transmitir la información señal original.

La señal en banda base es multiplicada por la señal de *spreading* y luego modulada en 2,4 GHz. En el receptor, la señal es recuperada aplicando un código "*de-spreading*" que es

idéntico al aplicado para la señal de *spreading* en el transmisor. La auto-correlación de los códigos de *spreading* es alta mientras la auto-correlación con otras secuencias es baja. Estas dos propiedades son muy importantes en un sistema DSSS, ya que permite sólo la señal deseada sea recuperada aplicando el algoritmo "*de-spreading"*. Por otro lado las señales multiplexadas con distintos códigos *spreading* y señales de interferencia normalmente aparecerán como ruido blanco luego de pasar por el mecanismo de *spreading*. La relación de ancho de banda de la señal dispersa y la señal en banda base es llamado "ganancia de procesamiento". En general, una ganancia grande implica una mayor resistencia a interferencias del sistema DSSS [4.2].

Las siguientes figuras muestran el diagrama de bloques de un sistema DSSS y su comportamiento frente a una interferencia de banda angosta.

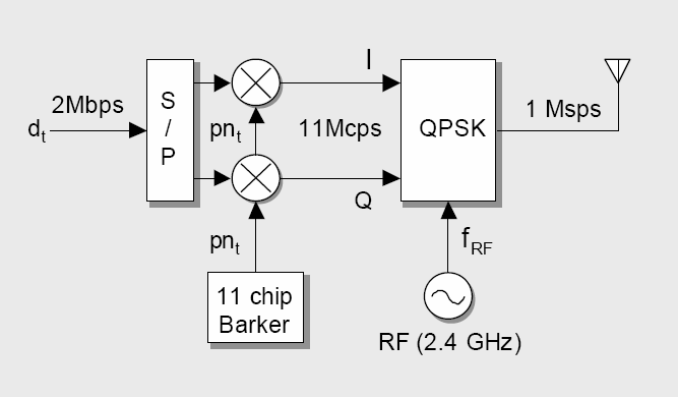


Figura 4.5a: Diagrama de bloques de un sistema DSSS [4.4]

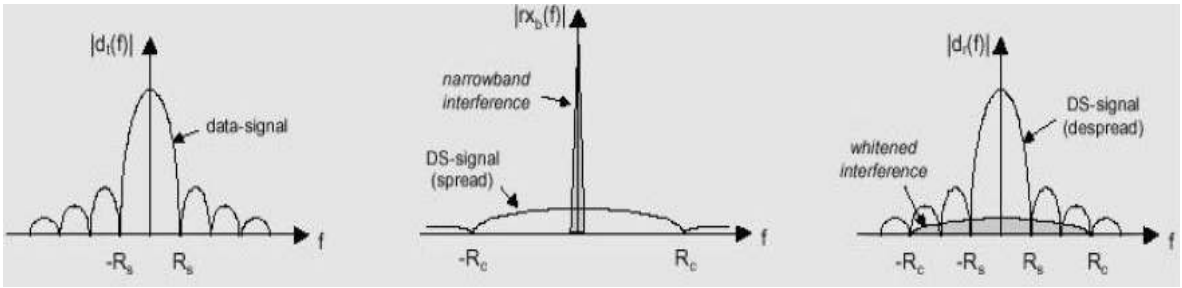


Figura 4.5b: La interferencia de banda angosta se asemeja a AWNG luego del de-spreading [4.5].

Existen tres aspectos principales a considerar para elegir el sistema DSSS más apropiado. Estos son: *chip rate*, propiedades de la correlación y costo de procesamiento al implementar el código.

El "**chip rate"** ó **tasa de chip** determina el ancho de banda de la señal dispersa y la ganancia de procesamiento. Por ejemplo, los dispositivos que cumplen con el Estandard 802.11 utilizan un código de Barker de 11 bits para expandir el espectro; el ancho de banda de 22 MHz, una tasa de bits de 2 Mbit/s y el *chip rate* está limitado a 11 Mchip/s.

La ganancia de procesamiento se define como:

$$10.4 \text{ dB} = 10\log (BW/BR) = 10\log(C/S) \quad (4.1)$$

Donde BW es el ancho de banda, BR la tasa de bit, C la tasa de chip y S la tasa de símbolos. Para el ejemplo anterior, la tasa de símbolos es 1 Msímbolo/s y la ganancia de procesamiento es 10.4 dB. Es importante destacar que en muchos textos se define la ganancia de procesamiento como la relación entre la tasa de chip y la tasa de bit. Sin embargo la FCC define la ganancia de procesamiento como la ec. 4.1 por lo que la mayoría de los estándares adoptan esta definición [4.2].

La **auto-correlación** se obtiene hallando la correlación entre una secuencia de *spreading* y consigo misma corrida en el tiempo. Un código de *spreading* debe tener buena auto-correlación, esto facilita el sincronismo. La **correlación cruzada** debe ser mínima o cero si se quiere lograr CDMA, para ello se utilizan secuencias aleatorias (PN, *pseudo noise*) ortogonales. Para lograr esta condición se utilizan secuencias PN más largas (13 y 15 bits), pero esto conlleva a mayores costos de implementación y a aumentar el ancho de banda o disminuir la tasa de datos. Otras técnicas como CSMA o retransmisiones pueden utilizarse para aportar robustez al sistema si la ganancia de procesamiento no es la suficiente para obtener una tasa de error aceptable [4.2, 4.3].

4.3.2 Espectro Ensanchado por Salto de Frecuencia

En los sistemas FHSS, el transmisor salta de canal en una secuencia específica. Para esparcir las trasmisiones a través del espectro de forma eficiente se utiliza una secuencia pseudo aleatoria que controla la frecuencia del oscilador local. El salto de frecuencia a través de los distintos canales de la banda logra que la densidad espectral disminuya mientras se evitan interferencias con otros dispositivos. Los parámetros de FHSS son la

tabla de saltos de canal y el tiempo de canal o período de salto. En la figura siguiente observamos como se implementa el FH del estándar 802.11.

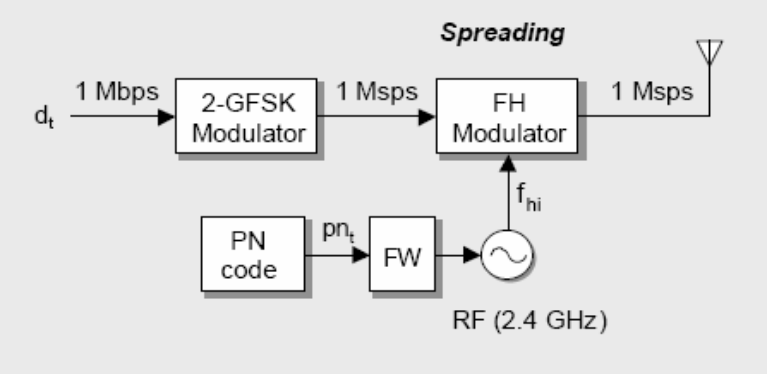


Figura 4.6a: Diagrama de bloques de un sistema FHSS. [4.4]

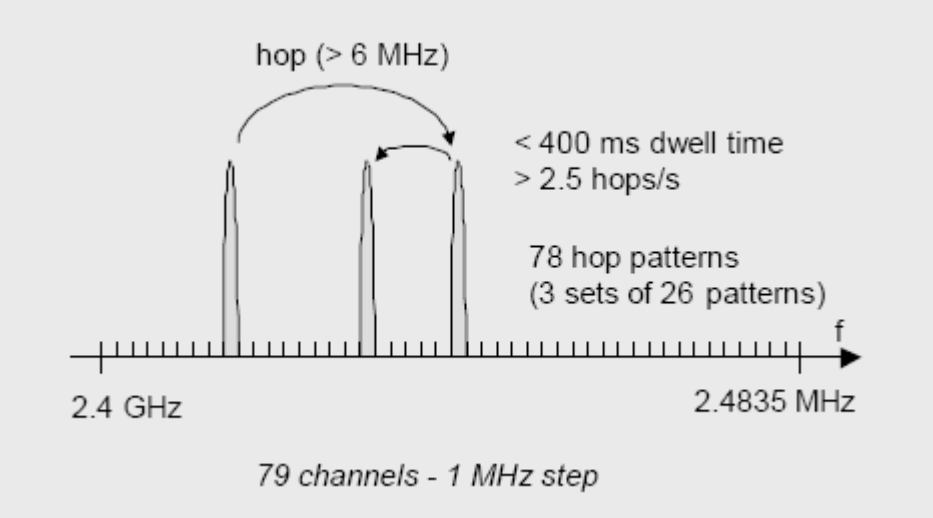


Figura 4.6a: Síntesis del espectro de un sistema FHSS 802.11. [4.4]

El FHSS del estándar IEEE 802.11 utilizan 79 canales con frecuencias centrales en 2402.0 - 2480.0 MHz con 1 MHz de separación. Este estándar permite el uso de 78 secuencias de saltos distintas que son agrupadas en tres *sets* de 26 secuencias ortogonales, la separación

mínima entre saltos consecutivos es de 6 MHz. La ortogonalidad implica que no habrá interferencia entre dos dispositivos que trasmiten secuencias distintas en un mismo *set*.

En lo que respecta al tiempo de salto, este puede ser configurado para Fast FHSS (tasa de saltos mayor a la tasa de datos de acceso) o Slow FHSS (tasa de saltos menor a la tasa de datos de acceso). La tasa mínima de datos es, en general, establecida por las entidades reguladoras de cada país. Cuanto mayor sea la frecuencia de salto menos tiempo se ocupa un canal determinado y la distribución de potencia a través del espectro es mejor, reduciendo así la probabilidad de interferencia con otros equipos. Bluetooth utiliza, por ejemplo, una frecuencia de salto de 1600 saltos/s. Sin embargo, la ETSI, en Europa, regula que el tiempo máximo de permanencia en el canal sea 0,4 s (unos 2,5 saltos/s) con un número mínimos de 20 canales [4.2].

Las velocidades máximas alcanzables, manteniendo un uso eficiente del espectro, es 3 Mbit/s; esto debido a que una mayor tasa de datos implica canales más anchos reduciendo el número de canales.

4.3.3 DSSS vs. FHSS, fundamentos de la elección

Para elegir la técnica de SS más apropiada para llevar a cabo el proyecto se compararon ambas técnicas. Dentro de los puntos comparados los más influyentes fueron: inmunidad contra interferencias, costo de implementación, máxima tasa de transmisión y coexistencia entre redes.

Otras características como consumo y eficiencia espectral de la modulación digital serán consideradas pero no influirán sustancialmente en la decisión. El parámetro que se utiliza para medir el rendimiento del los sistemas, según presencia de ruido u/ó interferencias y/o obstáculos, es el BER (*Bit Error Rate*) que es la tasa de errores de bit.

A continuación se describirán brevemente las propiedades que caracterizan las modulaciones DSSS y FHSS, y como alteran el rendimiento del sistema en el escenario particular de nuestra aplicación.

Reducción de la Densidad Espectral

Tanto FHSS como DSSS cumplen con el fin de reducir la densidad espectral de potencia (DEP) media, pero logran esto de manera muy distinta. En primer lugar DSSS disminuye la DEP instantánea, distribuyendo la potencia por un canal más ancho. Como resultado de esto se interfiere con más dispositivos pero el efecto es minimizado por la ganancia de procesamiento. FHSS transmite con una DEP instantánea mucho mayor, pero al saltar de

canal en ancho mucho más grande baja considerablemente la DEP media sobre esta banda; y mientras tanto las nuevas redes que utilicen secuencias ortogonales (caso ideal) no tendrán interferencias entre ellas, utilizando así el espectro de forma eficiente [4.2] [4.3].

Tipo de modulación

DSSS utiliza la modulación PSK que es más eficiente que GFSK, la utilizada por FHSS, en el sentido que necesita una menor energía de bit sobre ruido (Eb/No) para obtener la misma tasa de errores, como muestra la figura 4.7. La modulación PSK es más compleja ya que necesita de un método de detección coherente; además debe utilizar una lógica más rápida para obtener la misma tasa de datos. La modulación GFSK si bien no es tan eficiente, es generalmente mucho más simple [4,2].

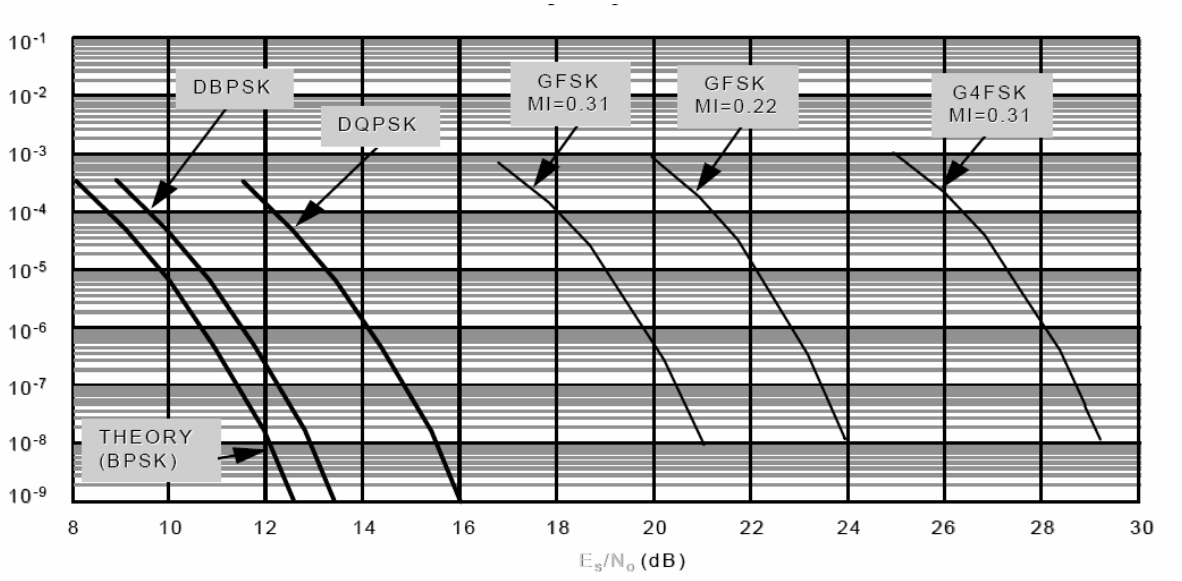


Figura 4.7: BER vs. Eb/No, comparación entre las distintas variaciones de la modulación PSK y FSK [4.3].

Sensibilidad a interferencias

Los sistemas SS son caracterizados por su desempeño frente a interferencias de banda ancha y de banda angosta. Para lidiar con interferencias de banda ancha DSSS utiliza la ganancia de procesamiento aumentar la señal deseada frente al ruido o interferencia de otros sistemas DSSS. Por ejemplo, DSSS 802.11 operar debe operar a Eb/No de 13,4 dB para obtener una tasa de error aceptable, y tolera interferencias de hasta -3 dB por debajo de

la señal deseada. FHSS, en cambio, utiliza una mayor potencia, 24 dB a 1Mb/s, en un canal más pequeño, pero al no contar con una ganancia de procesamiento dentro de un salto, sólo puede tolerar interferencias hasta -19 dB respecto a la portadora, lo que determina un peor desempeño frente a interferencias de banda ancha [4.2] [4.3].

En lo que respecta a la interferencia de banda angosta son minimizadas por DSSS ya que al aplicar el código de *de-spreading* se ven como ruido blanco. FHSS sólo es afectado por interferencia de banda angosta que caen dentro del mismo canal, las cuales pueden ser evitas por AFHSS (Adaptative FHSS), en caso de ser estáticas y DAFHSS (Dinamic AFHSS) en caso de ser dinámica, como sistemas de FH no sincronizados interfiriéndose mutuamente [4.2][4.3][4.4].

Velocidad

La técnica DSSS permite alcanzar una mayor velocidad simplemente aumentando la frecuencia del reloj ya que posee canales muchos más anchos. Si bien se debe aumentar la potencia o perder alcance. Otra forma es cambiando el tipo de modulación hacia uno más complejo, pero consecuente aumenta la tasa errores por lo que debería aumentarse la potencia. FHSS esta mucho más limitada. Ya que no puede aumentar el ancho de canal por que disminuiría el número de canales, lo que implica una peor distribución de la potencia a través del espectro. Tampoco pude acomplejar la modulación ya que 8-FSK (la que le sigue a GFSK en tasa de datos) produce tasas de errores que son inaceptables [4.3].

Acceso Múltiple

DSSS no puede aplicar CDMA debido a que ganancia de procesamiento y la E/No requerida no permiten que otro dispositivo opere en el mismo canal. Al tener un menor número de canales no permite que muchas redes operen en el mismo canal. Para 802.11 no más de 3, y lo sumo 4 redes pueden operar en el mismo espacio. FHSS presenta más ventajas en este aspecto ya que en el mismo espacio pueden operar hasta 15 redes juntas. Si bien al ser mucho mayor la energía total que dispersa por el espectro, otros nodos de 15 redes deben estar más alejados que los nodos de 3 redes DSSS [4.2] [4.3].

Alcance

DSSS puede alcanzar distancias más largas debido a la ganancia de procesamiento. Aunque es más vulnerable al ruido e interferencia debido al mayor ancho de banda es en general el

preferido para cubrir distancias grandes en la banda estudiada de frecuencias [4.2] [4.3] [4.7].

Caminos Múltiples

La señal de radio es deteriorada por auto interferencia debido a los múltiples caminos que toma la señal. Es muy común en ambientes con alta densidad de infraestructura. Este efecto puede verse como un supresor de alguna frecuencia en respuesta espectral del canal.

En ambientes con mayor densidad de infraestructura (edificio de oficinas) el entorno de frecuencias que se ve afectado aumenta. Los sistemas DS son mayormente afectados por este hecho, por lo que FHSS es más recomendado para casos con probabilidad de interferencias por caminos múltiples [4.2][4.7].

Costos de Implementación

La modulación por DS es más compleja y requiere una lógica más rápida, por lo que en principio sería más costosa. Pero el amplio desarrollo de esta tecnología ha hecho que las cosas se emparejen

Consumo

Las mismas razones que hacen la modulación DSSS más costosa hacen que consuma más energía. En lo que respecta a recursos que requiere FHSS lleva una pequeña ventaja. Sin embrago al poder alcanzar mejores tasas de datos con DSSS se podría disminuir el consumo aumentando los tiempos en que el sistema permanece en estado de bajo consumo.

4.3.4 Elección

FHSS es más apropiado para ambientes con alta densidad obstáculos e infraestructura, como los que están en los objetivos de nuestro proyecto. Esto se debe a que tienen una inmunidad frente fuertes interferencias en un rango angosto de frecuencias. Además, si bien DSSS ofrece un mayor tasa de datos, en los chips de bajo costo analizados (ver elección del hardware en capitulo 6) la tasa máxima que ofrecen es de 256 kb/s (cumplen con la IEEE 802.15.4). Por las razones antes explicadas, además de que era posible conseguir dispositivos para aplicar FHSS a muy bajo costo con tasas de datos de hasta 1152 kb/s,

consideramos a FHSS como la técnica de espectro ensanchado más apropiada para llevar acabo nuestro proyecto.

4.4 Funcionamiento de FHSS

Basado en el transreceptor elegido (ver descripción de hardware en el capitulo 6) se diseño un sistema de FH que reparte eficientemente las señales a través de la banda ISM de 2,4 GHz. En primera instancia se eligió la secuencia de saltos fija y no aleatoria, lo que simplifica las cosas. Los parámetros del FH son: tiempo de canal y salto de canal.

4.4.1 Características de Hardware utilizado

El transreceptor utilizado es el ATR 2406 de Atmel. Este dispositivo transmite a 2,5 mW, que está muy por debajo de la potencia máxima permitida (10 mW en Japón); un amplificador LNA interno permite una sensibilidad de -93 dBm. Utiliza modulación GFSK con un PLL para ajustar la frecuencia y 95 canales con desviación de $\pm$ 400 KHz. El receptor utiliza un filtro de baja FI, 869 KHz, para rechazar la frecuencia imagen. Por mayor información referirse a la hoja de datos de chip [7.6].

4.4.2 Implementación del algoritmo

El algoritmo utilizado esta basado en la implementación del *firmware* "Smart RF" que ofrece Atmel para este chip. El mismo consta de elegir la cantidad de canales igual a un número primo (89). Esto logra que cualquiera sea el salto de canal se pasa por todos los canales antes de llegar al inicial. El salto de canal debe ser mayor que 4 y menor que 84 para lograr que haya una diferencia mínima de 5 entre canales consecutivos, esto ayuda a distribuir mejor la potencia.

4.4.2.1 Sincronismo

Para que el FH funcione todos los dispositivos deben cambiar al mismo canal al mismo tiempo y por lo tanto deben estar sincronizados. El software que implementa el FH debe encargarse de mantener sincronizado a todos los dispositivos de la red. Los "relojes" de los microprocesadores en general presentan deriva, ya sea por diferencias de temperatura que alteran la frecuencia del reloj, o por razones de alguna otra índole. Por esta razón la base que administra la red debe encargarse de enviar periódicamente a todos los dispositivos la

información necesaria para que se sincronicen. Además para evitar una posible incertidumbre en el momento de cambiar de canal, se pude utilizar una ventana que habilite a transmitir sólo cuando todos los dispositivos, o por lo menos los que estén dentro del radio de alcance del emisor, estén en el mismo canal.

4.4.2.2 Asociación a la red

Dentro de los intereses del proyecto está simplificar la instalación al máximo, lo que llevó a incluir a los objetivos del proyecto que los dispositivos RF fueran "plug & play". Para lograr esto es necesario que los dispositivos se asocien a la red de forma automática. Para esto el dispositivo debe "engancharse" con FH de la red por lo que debe conocer los parámetros de FH. En principio los parámetros serán enviados en trama de sincronismo y para un dispositivo nuevo que desee asociarse este escuchará en uno o varios canales fijos. Si bien el FH recorre todos los canales en un ciclo el dispositivo puede estar alejado y no recibir muchas tramas, esto puede hacer que el enganche demore mucho. Una solución es que las tramas de sincronismo se transmitan sólo en los canales que utiliza el receptor para sincronizarse al FH.

4.3.3 Mejoras Futuras

La mejora principal es la establecer secuencias de saltos pseudo aleatorias, ya que esta es la forma más efectiva de evitar que varias redes interfieran entre sí. La idea más manejada es la de almacenar secuencias pseudo aleatorias de saltos ortogonales en todos los dispositivos, donde una red utilizaría una de estas secuencias.

4.4 Resumen

En este capítulo se estudiaron las distintas bandas de frecuencias ISM; y si bien las bandas más bajas (869 MHz y 915 MHz) alcanzan mayores distancias, la ventaja de poder lograr una tasa de datos mucho mayor en 2,4GHz y el gran desarrollo tecnológico sobre esta banda hizo que la eligiéramos. Luego se discutió sobre las ventajas y desventajas de las modulaciones DSSS y FHSS, concluyendo que para nuestra aplicación ambiente congestionado con muchos obstáculos, FHSS es la más apropiada. Finalmente se presentó un modelo de FHSS utilizando el transreceptor ATR2406, el cual se basa en implementación del *firmware* "Smart RF".

Capítulo 5

Protocolo de comunicación inalámbrica WiDO 1.0

En este capítulo se describe el protocolo de comunicación utilizado para la red inalámbrica, indicando su modelo de capas y sus principales características. A su vez se exponen los fundamentos que llevaron a la definición del mismo.

5.1 Definición de las características del protocolo

Para definir los principales lineamientos del protocolo se manejaron distintos enfoques generales, los cuales fueron contrastados con los requerimientos específicos de la aplicación para la cual se desarrolló este proyecto y de las características físicas de la red. Es decir, en primer lugar se estudió en rasgos generales hacia donde debería estar orientado el protocolo, como por ejemplo, si utiliza ruteo, si implementa confirmación de recepción de paquetes, etc.; ponderando principalmente que el protocolo diseñado tenga un conjunto de propiedades, las cuales sean las que mejor se ajusten a los requerimientos del problema y a los recursos de los cuales se dispone.

Como ya fue expuesto en el capítulo 2, las características más relevantes que presenta el sistema del cliente son las siguientes:

- Tramas cortas
- Mayor parte del tiempo sin transmisiones
- Distancias relativamente cortas
- Baja cantidad de nodos

Entonces, a partir de estas características se definió el tipo de protocolo inalámbrico a diseñar.

5.1.1 Retransmisión de paquetes

En primera instancia, se llegó a la conclusión de que si bien se determinó que en general las distancias entre los dispositivos del sistema son relativamente cortas, la disposición de estos en la red es aleatoria, ya que varía según cada caso en particular y entonces sucede con una alta probabilidad que el alcance de la comunicación entre dos dispositivos cualesquiera no sea suficiente para cubrir la distancia existente entre ellos.

Debido a esta situación se decidió que el protocolo de red utilice el concepto de retransmisión de mensajes utilizado en redes Mesh, las cuales son comúnmente utilizadas para redes inalámbricas. En este tipo de redes los dispositivos pertenecientes a la red son utilizados no solo en caso de precisar comunicación con los mismos, sino que también para propagar los mensajes que no están dirigidos a estos, retransmitiéndolos hacia otros dispositivos, de manera de ampliar el alcance de la red. Esto permite que al querer transmitir un mensaje a un dispositivo cualquiera no sea necesario estar en su zona de alcance, ya que el mensaje llegará de todas formas a través de las retransmisiones de otros dispositivos. De esta forma en principio solo basta con asegurar que un dispositivo no este completamente aislado, es decir que tenga al menos un dispositivo al alcance de las transmisiones, para considerar que dicho dispositivo forma parte de la red.

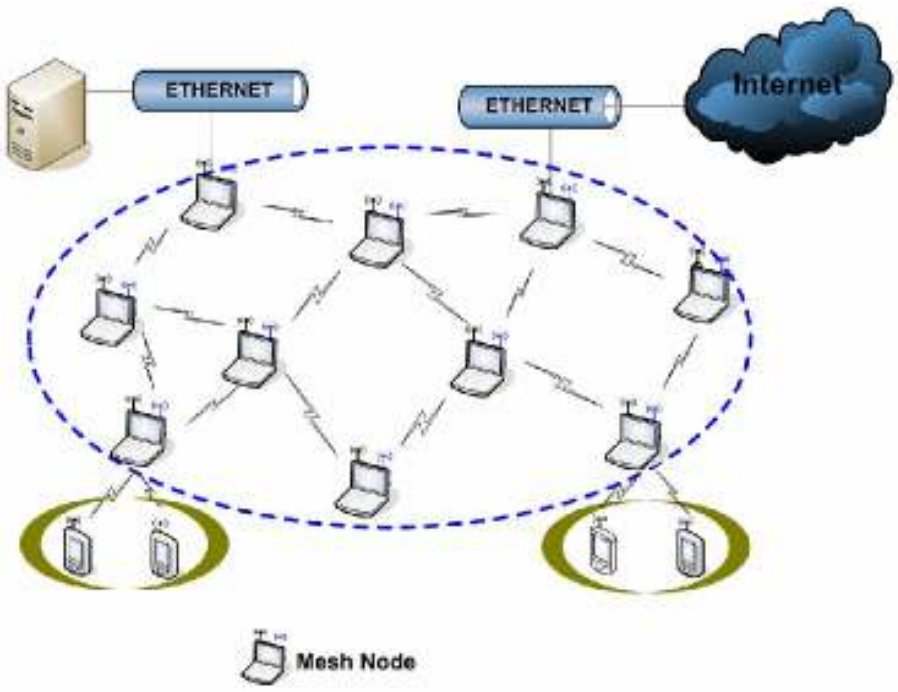


Figura 5.1: Ejemplo de red Mesh.

Al implementar esta funcionalidad de retransmisión de los mensajes, surge la posibilidad de que existan en la red dispositivos remotos que no posean ningún dispositivo final (sensor o actuador) asociado, sino que solo retransmitan los mensajes que le llegan, dicho de otro modo, que actúen de repetidores.

Un posible caso en el cual esta propiedad puede ser utilizada, podría ser en una situación en la cual se pretenda unir dos sectores separados de un establecimiento en una misma red, por lo que entonces se colocaría un dispositivo remoto (o dos para lograr redundancia) como punto de enganche de ambas partes. Un ejemplo de esta situación se observa en la figura 5.2.

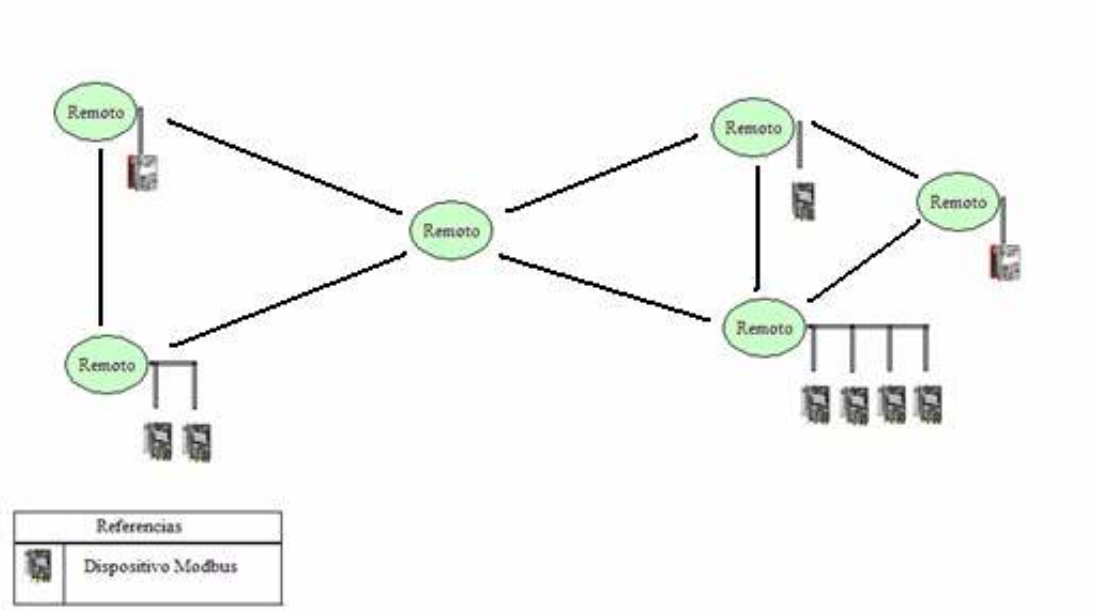


Figura 5.2: Dispositivo remoto actuando como enlace.

Otro aspecto de las redes Mesh que resultó apropiado para incluir en el protocolo es la posibilidad de que exista comunicación directa entre dos dispositivos cualesquiera, lo cual es conocido como comunicación *"peer to peer"*. Para este caso esto implica que la comunicación entre base y dispositivo remoto se lleva a cabo de la misma forma que la comunicación entre dos dispositivos remotos, lo cual le da al sistema cierta flexibilidad en cuanto a futuros cambios en el funcionamiento.

5.1.2 Inundación de la red

Para definir la forma en que los mensajes llegan a destino se confrontaron dos métodos muy diferentes, por un lado, se manejó la posibilidad de implementar alguno de los métodos de ruteo de paquetes y por otro lado, se pensó en utilizar un método de difusión de los mensajes, lo cual consiste en inundar la red de forma de que los mensajes lleguen a destino por todos los caminos posibles, sin necesidad de determinar un camino en especial para alcanzar al destinatario del mensaje.

Debido a que el ancho de banda no es un aspecto muy relevante en esta aplicación, el hecho de realizar ruteo no brinda mayores beneficios teniendo en cuenta la utilización de los recursos disponibles y de la complejidad de la aplicación. Es decir, es preferible, en este caso, la eventual pérdida de paquetes y la cantidad de retransmisiones innecesarias que implica utilizar difusión, a realizar tablas de ruteo y algoritmos para construir estas tablas, lo cual vuelve más complejo el desarrollo en forma innecesaria. Por lo que entonces esta situación resultó un punto a favor de implementar inundación de mensajes.

Por otra parte, el ruteo presenta la ventaja de tener una mejor *performance* desde el punto de vista de la interferencia que se pueden causar dos redes que deben coexistir en el mismo medio, ya que este optimiza la cantidad de transmisiones inútiles en una comunicación exitosa. Como contra parte de esto, la inundación de mensajes obliga a transmitir a todos los nodos de la red cada vez que se manda un paquete y esto aumenta considerablemente el tráfico, lo cual también aumenta la posibilidad de que ocurran colisiones. Esta situación puede llegar a ser un inconveniente y una desventaja considerable para el uso de inundación, dependiendo del tamaño de la red.

Una ventaja importante que se obtiene de utilizar la difusión de los mensajes por todos los nodos de la red, es la de mayor robustez frente a la eventual caída de uno o mas nodos, ya que el paquete que se envía siempre encuentra el camino para llegar a su destino sin tener que correr algoritmos de ruteo para solucionar el problema.

Luego de este análisis de las principales ventajas y desventajas que presentan ambos métodos, se definió por utilizar inundación de mensajes. Esta decisión fue tomada principalmente por que este método es más adecuado a las características presentadas por el sistema del cliente, mencionadas al inicio de esta sección.

5.1.3 Confirmación de recepción de mensajes

Un aspecto importante que resultó necesario definir, fue el hecho de implementar o no la confirmación de recepción de los mensajes (mediante paquetes denominados ACK), y en caso afirmativo, de que manera se realizaría.

Luego de un breve análisis fue claro que se debía implementar reconocimientos de mensajes, ya que al ser el protocolo del tipo no orientado a conexión, se necesita determinar de alguna forma si el mensaje logra alcanzar el destino. Por otro lado, debido a que el protocolo utiliza el método de inundación de la red, si se implementa la confirmación de recepción a cada uno de los paquetes, es decir, a cada una de las retransmisiones generadas

en cada transmisión, se generarían una cantidad inmanejable de paquetes que con gran seguridad saturarían la red.

Debido a esta situación se decidió implementar la confirmación de los mensajes de punta a punta de la comunicación, o sea a nivel de capa de aplicación.

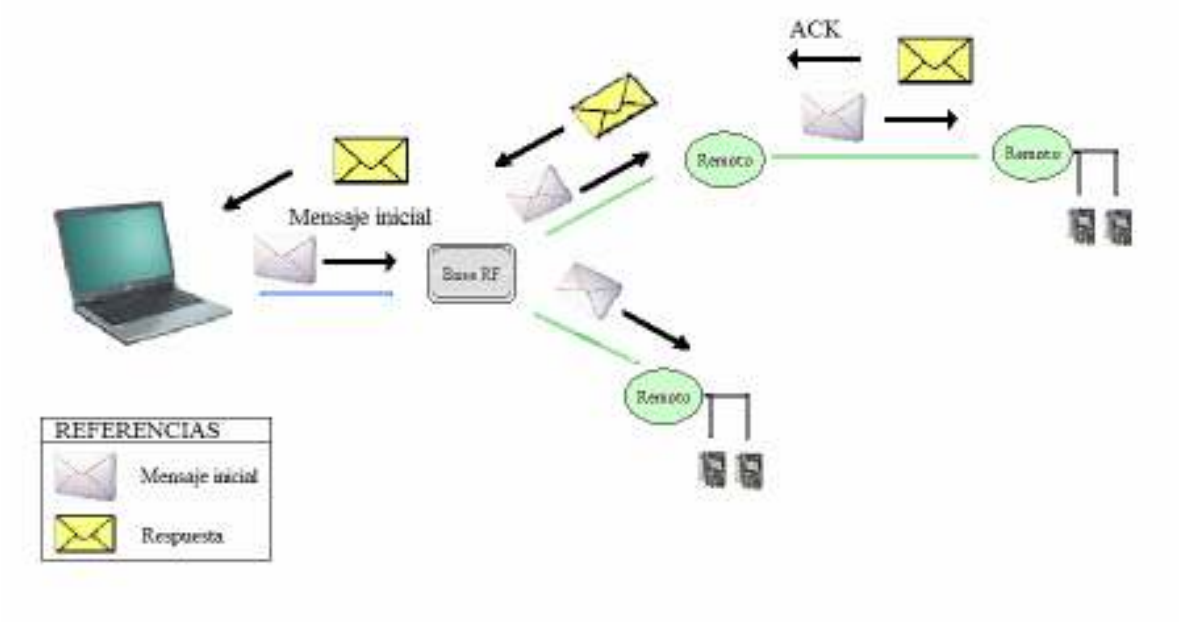


Figura 5.3: Ejemplo de confirmación de la recepción de un paquete.

5.1.4 Acceso al medio

Se analizaron distintas técnicas existentes para resolver el acceso al medio, en especial aquellas utilizadas para redes inalámbricas, buscando el que mejor se adaptase a esta aplicación especifica y que además fuera compatible con los rasgos del protocolo ya definidos hasta ese momento.

Lo más simple en cuanto al acceso a un medio compartido sería no realizar ningún tipo de control, es decir que cada dispositivo inicie una transmisión cada vez que lo requiera, sin importar si el canal esta libre, generando colisiones en caso contrario. Como resultado de esto, cualquier protocolo que implemente el acceso al canal de esta manera será muy ineficiente, ya que la probabilidad de que existan colisiones será grande y aumenta con la cantidad de nodos que se tengan en la red. Especialmente para la red implementada, la probabilidad de que se generen colisiones, en el caso de acceder al medio de esta forma, es muy alta, independientemente del número de nodos presentes en el sistema. Esto es así debido a que al implementar la retransmisión de los mensajes al igual que una red Mesh, es claro que sucede con mucha frecuencia que dos o más dispositivos reciban un mismo mensaje prácticamente de manera simultanea, y por lo tanto al momento de retransmitirlo colisionarán. Debido a este análisis se concluyó que implementar el acceso al canal de esta forma llevaría al fracaso rotundo en las comunicaciones de la red y por lo tanto fue descartado.

Una técnica utilizada con frecuencia para resolver el acceso múltiple a un medio es la denominada CSMA (*Carrier Sense Multiple Access*). Esta consiste en que un nodo de la red, al momento de precisar iniciar una transmisión, realiza un censado del canal para decidir si debe transmitir o no, es decir que si verifica que el canal esta libre realiza la transmisión y en caso contrario espera un tiempo aleatorio para volver a “escuchar” el canal, así sucesivamente hasta encontrar libre el canal y lograr transmitir con éxito.

Existen dos variantes para CSMA, las cuales le agregan funcionalidades, CSMA/CD (*Carrier Sense Multiple Access / Collision Detection*) y CSMA/CA (*Carrier Sense Multiple Access / Collision Avoidance*).

En el caso de CSMA/CD, es una variante de CSMA puro, en la cual se le agrega la funcionalidad de que en el transcurso de una transmisión, el dispositivo que está realizando dicha transmisión detecta si ocurre una colisión debida a una transmisión de otro dispositivo y detiene la transmisión inmediatamente. Luego espera un tiempo aleatorio antes de volver a intentar transmitir. Esta modificación de CSMA puro, intenta mejorar la *performance* del protocolo al minimizar los tiempos que se pierden durante el transcurso de las colisiones.

Cabe destacar que CSMA/CD es el utilizado en el protocolo Ethernet, pero no es adecuado para redes inalámbricas, principalmente debido a que en éstas no es posible transmitir y escuchar el medio al mismo tiempo, por lo que no es posible entonces detectar las colisiones, ya que las colisiones se detectan escuchando el medio y comparando lo recibido con lo transmitido simultáneamente, determinando que hay una colisión en caso de haber diferencias.

La otra variante, CSMA/CA, la cual es utilizada en el protocolo IEEE 802.11, busca mejorar la performance del protocolo al evitar las colisiones. Para cumplir con este objetivo utiliza los mensajes de control RTS (Request To Send) y CTS (Clear To Send). Esto consiste en que el nodo que intenta iniciar una transmisión, una vez que el canal está libre, envía un paquete RTS (dirigido al futuro receptor del mensaje a transmitir), lo cual avisa al resto de los nodos dentro de su alcance que solicita una transmisión, luego el nodo al cual se le quiere transmitir el mensaje, responde un paquete CTS dirigido al nodo que envió el paquete RTS, indicando que es libre de transmitir y por lo tanto el resto de los nodos al alcance del futuro receptor son alertados de que se realizará una transmisión, por lo que entonces no interferirán durante dicha transmisión.

El uso de CSMA/CA resuelve un problema importante conocido como el problema de la estación oculta. Este problema se presenta cuando existen en la red al menos dos dispositivos que no están al alcance entre si y por lo tanto pueden ocasionar colisiones, ya que al utilizar CSMA puro, puede suceder que ambos transmitan simultáneamente, dado que al momento de censar el canal los dos dispositivos lo interpretaran como libre, generando entonces interferencias en los dispositivos que estén al alcance de los dos. Esta situación es resuelta mediante el uso de los paquetes RTS y CTS.

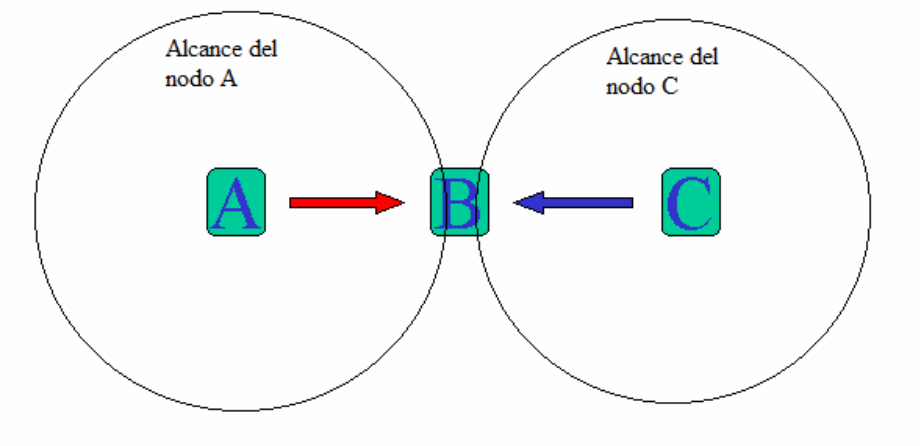


Figura 5.4: Problema de la estación oculta.

Por otra parte, el implementar CSMA/CA no resuelve el problema conocido como el de la estación expuesta. Un ejemplo de este problema se observa en la figura 5.5. Cuando el dispositivo B realiza una transmisión hacia el dispositivo A, el C al momento de transmitir debido a que implementa CSMA, advierte esta situación al censar el canal y se abstiene de realizar la transmisión. Sin embargo este podría realizar una transmisión exitosa hacia el dispositivo D, ya que éste no se encuentra al alcance del dispositivo B, el cual esta transmitiendo y por lo tanto no se produciría colisión alguna en la recepción de D.

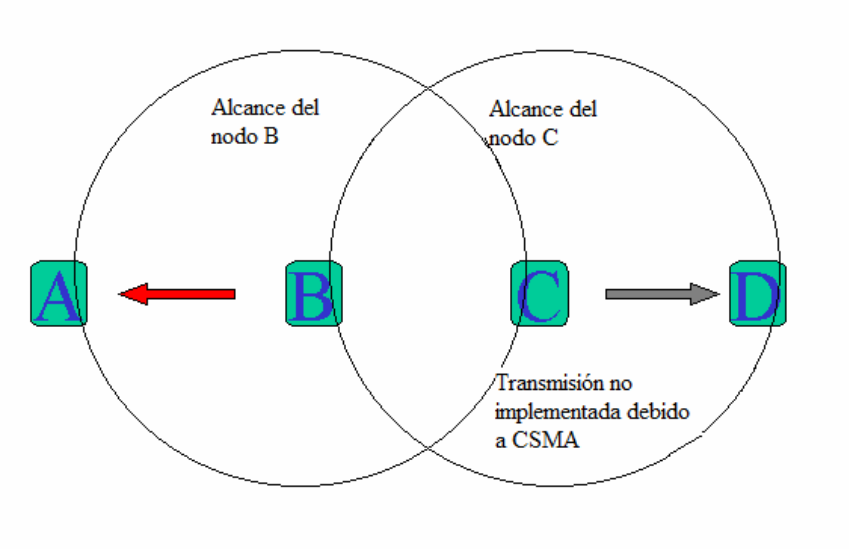


Figura 5.5: Problema de la estación expuesta.

Como ya se mencionó, esta situación no es solucionada por el hecho de utilizar CSMA/CA. Como ejemplo, tomando los mismos nodos que en la figura del ejemplo anterior, puede suceder que el dispositivo C sea advertido de no transmitir, al recibir un paquete RTS de B, el cual pretende realizar una transmisión al dispositivo A. Nuevamente sucede que en el caso de que el dispositivo C precise de realizar una transmisión hacia D se abstenga de hacerlo a consecuencia de escuchar el paquete RTS proveniente de B, cuando en realidad no habría colisión en la recepción en D.

Luego de evaluar las ventajas y desventajas de CSMA y de sus dos variantes, CSMA/CA y CSMA/CD, se definió que el protocolo implementase CSMA puro para resolver el acceso al medio de los nodos de la red. Esta decisión fue basada principalmente por el hecho de que si bien el utilizar CSMA/CA evita la gran mayoría de las potenciales colisiones, se consideró que para el tipo de red diseñada, trae aparejados otros inconvenientes. El principal inconveniente es la cantidad de paquetes que se generarían en la red a consecuencia de una sola transmisión, lo cual sucede debido a la retransmisión de los

mensajes en la red, es decir, como ya se destacó anteriormente, por cada mensaje iniciado por un nodo de la red, en principio, se generan tantas transmisiones como la cantidad de nodos presentes en la misma. Entonces a consecuencia de esto, si se implementa CSMA/CA utilizando los paquetes RTS y CTS, la cantidad de mensajes que se generan a su vez se triplicarían, dada la secuencia de eventos de CSMA/CA. Por lo tanto la eficiencia de la red baja considerablemente debido a la cantidad total de transmisiones generadas por cada transmisión útil y al tiempo transcurrido entre que se inicia la transmisión original hasta que la red vuelve al estado de reposo.

5.1.5 Resumen

A continuación se resumen los factores determinantes en la definición del protocolo:

- Evitar la colisiones requiere que se manden paquetes RTS (Request To Send) y CTS (Clear To Send) para luego poder establecer la comunicación sin colisiones, pero realizar este proceso para asegurarse de que no ocurran colisiones en una red que tiene muy poco tráfico, se vuelve muy ineficiente. En el análisis quedo claro que con escuchar el canal en forma aleatoria, para transmitir cuando este no este ocupado, es suficiente para evitar la gran mayoría de las colisiones.

- Se llegó a la conclusión de que realizar ruteo no brinda mayores beneficios teniendo en cuenta que el ancho de banda no es un aspecto muy relevante para esta aplicación, y que por lo tanto realizar tablas de ruteo y algoritmos para construir dichas tablas, vuelve más complejo el desarrollo en forma innecesaria, por lo que entonces es preferible lidiar con la eventual pérdida de paquetes.

- El método de inundación es más robusto frente a la caída de un nodo, ya que el paquete que se envía siempre encuentra el camino para llegar a su destino sin tener que correr algoritmos de ruteo para solucionar el problema.

- El ruteo podría ser más eficiente desde el punto de vista de la interferencia que se pueden causar dos redes que deben coexistir en el mismo medio, ya que este optimiza la cantidad de transmisiones inútiles en una comunicación exitosa. Pero este aspecto no juega un papel importante en los requerimientos de la red. Al usar inundación se obliga a transmitir a todos los nodos de la red cada vez que se manda un paquete y esto aumenta el tráfico, pero teniendo en cuenta que la tasa de envíos es pequeña, no es un aspecto muy relevante.

En conclusión hacer un protocolo complejo con evasión de colisiones y ruteo de datos no fue necesario para nuestros requerimientos, entonces como consecuencia de esto, teniendo en cuenta el tiempo y los recursos de los cuales se disponía para la realización del mismo,

este fue descartado en contraposición con un protocolo más sencillo y que se ajustó mejor a los requerimientos del problema.

5.2 Características generales del protocolo

En la sección anterior se expusieron las principales características que debía poseer el protocolo RF diseñado para obtener una performance óptima. Si bien existen desventajas en este diseño con respecto a otro mas enfocado al enrutamiento de los paquetes, el factor fundamental por el cual se opto por este diseño, como se mencionó anteriormente en este capítulo, es que se ajusta más correctamente a las necesidades del problema.

Entre las características principales del protocolo se destacan las siguientes:

- Implementa inundación controlada para transmitir los mensajes
- Utiliza FHSS para evitar interferencias y proveer un uso eficiente del medio
- Utiliza el mecanismo CSMA puro para acceder al canal
- Implementa confirmación de recepción de mensajes

5.2.1 Inundación controlada

Con el fin de mejorar el rendimiento de la red se le agregaron ciertas funcionalidades al método de la inundación de paquetes, a lo cual se le llamó *inundación controlada*. Para lograr dicho objetivo, se buscó principalmente disminuir la cantidad de retransmisiones innecesarias, utilizando los conceptos de número de saltos de los paquetes, numero de identificación de mensaje y dispositivos al alcance directo de transmisión.

El número de saltos de los paquetes se define como la cantidad de dispositivos por los cuales el paquete fue transmitido hasta llegar a su dirección de destino y se creó un campo en la trama de capa de red llamado Saltos, el cual lleva el conteo de este valor hasta el momento de recibir el paquete. Este campo es utilizado para controlar la cantidad de retransmisiones de los paquetes al establecer un valor máximo de saltos para el cual el paquete es descartado y no se vuelve a transmitir.

Cada mensaje originado por un dispositivo es etiquetado por éste con un número generado aleatoriamente el cual es agregado a la trama de capa de red mediante el campo *Num_paquete*. Con esto lo que se busca es que cada dispositivo no vuelva a retransmitir un

paquete que ya recibió anteriormente, lo cual es posible dado que el campo *Num_paquete* es el número de identificación del paquete.

Se utilizó para realizar un control más fino de la difusión la funcionalidad de que cada dispositivo tenga conocimiento de cuales dispositivos están al alcance directo de su transmisión de modo de manipular el campo *Saltos* de la trama de red para que no se generen transmisiones innecesarias, ya que para alcanzar el dispositivo destino no se precisan más retransmisiones.

En resumen, en base a estos conceptos, se implementaron dos mecanismos para el control de difusión. Uno más básico que tiene en cuenta la cantidad de saltos y la redundancia del paquete y otro más complejo que define si un paquete se debe retransmitir o no, dependiendo de quien es el destinatario del mismo.

Estos mecanismos son explicados en detalle en la sección 5.3.4 en donde se describe la capa de red.

La siguiente secuencia de figuras muestra un ejemplo del funcionamiento de la inundación controlada de la red.

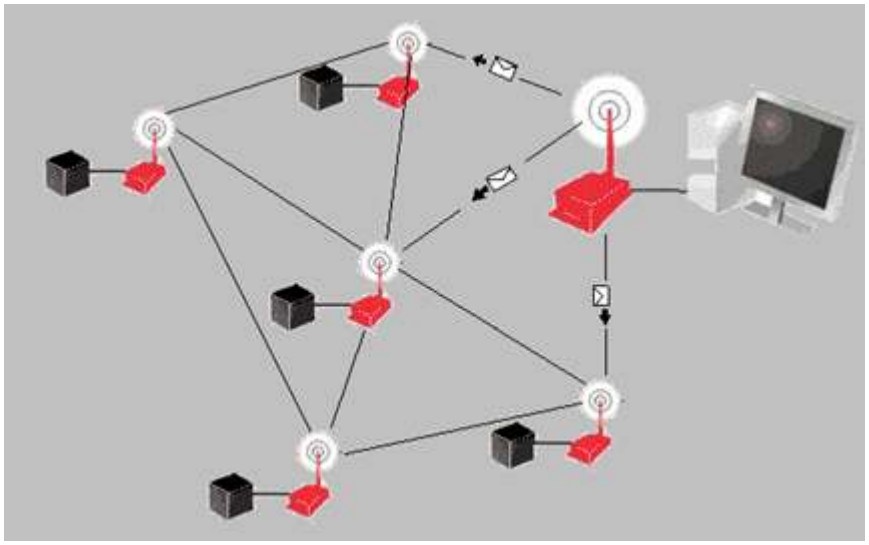


Figura 5.7: Transmisión de un mensaje originado por la base.

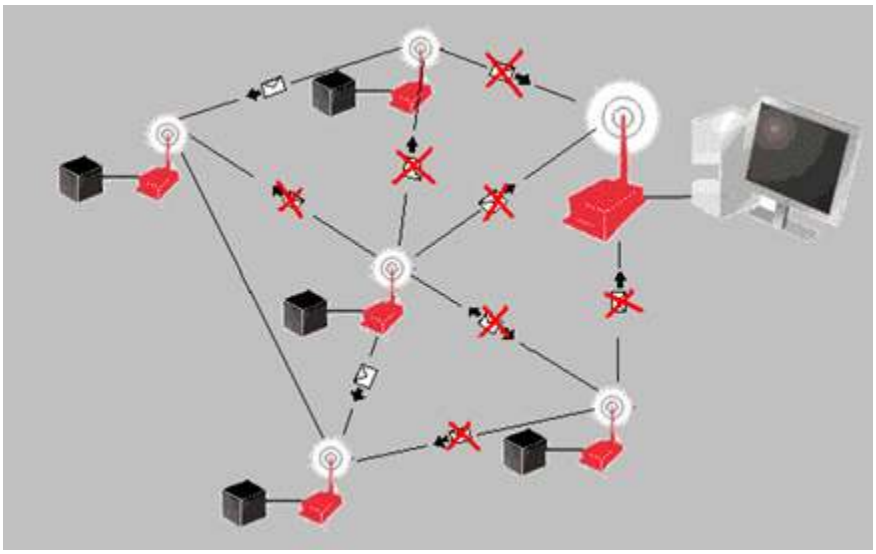


Figura 5.8: Retransmisiones descartadas por el control de inundación.

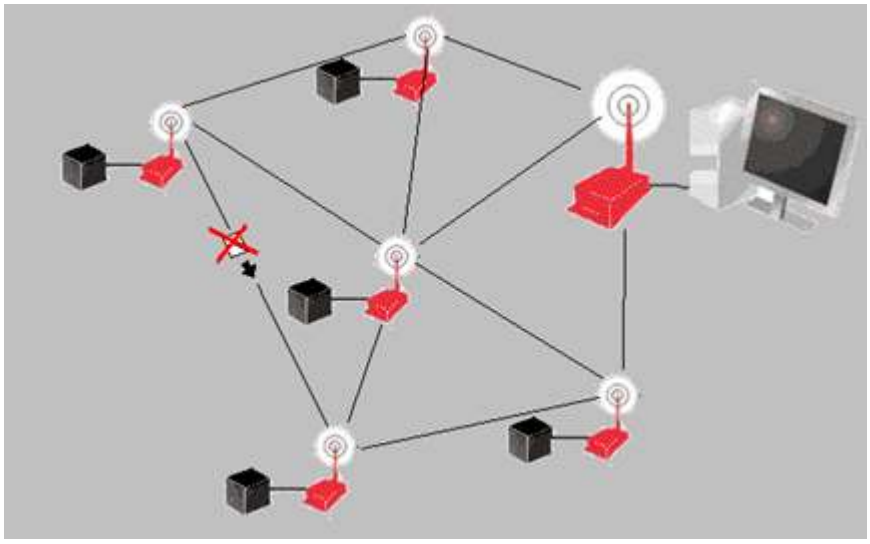


Figura 5.9: Mensaje descartado por el control de inundación.

5.2.2 CSMA

El protocolo utiliza como método para resolver el acceso al medio la técnica de CSMA puro.

Se pretende minimizar el problema de las colisiones con la acción de censar el canal antes de transmitir y en el caso que este ocupado esperar un tiempo aleatorio para volver a censar el canal buscando realizar la transmisión.

5.2.3 FHSS y sincronismo

El protocolo implementa FHSS como técnica de espectro ensanchado. El modo en que se utilizó fue explicado en detalle en la sección 4.3.

Debido al hecho de utilizar FHSS resulta ser un aspecto crítico la sincronización de los dispositivos de la red. Es decir, es necesario mantener sincronizados los saltos de canal de forma de que idealmente, estén en todo momento en el mismo canal.

Por esta razón se incluyó un campo en la trama de capa MAC con el propósito de portar información útil para el sincronismo. La idea es que un dispositivo se sincronice con otro dispositivo del cual recibe un mensaje. Con este mecanismo, lo que se pretende es que se minimicen los desfasajes entre los tiempos de cambio de canal de los dispositivos, ya que si bien, para que la comunicación funcione correctamente no es necesario que los dispositivos salten de canal exactamente al mismo tiempo, si es importante que esos desfasajes sean lo menores posibles.

A su vez se incluye a nivel de capa de aplicación del dispositivo base la funcionalidad de ejecutar tramas especiales para el control de sincronismo.

5.2.4 Ventajas

Dentro de las ventajas de este protocolo se destacan:

- Baja latencia
- Robustez frente a la caída de un nodo de la red
- Posibilidad de configuración en funcionamiento
- Simplicidad de implementación
- Flexibilidad para realizar cambios

5.3 Descripción del protocolo

5.3.1 Modelo de capas

El modelo de capas utilizado se basa, en el modelo OSI (Open Systems Interconection), donde la capa de enlace se sustituye por una capa de control para acceso múltiple (MAC) y una capa de enlace lógico que se incluye dentro de la misma. Las capas de transporte, sesión y presentación incluidas en el modelo OSI, no fueron implementadas en este protocolo.

Figura 5.10: Modelo de capas.

5.3.2 Capa Física

La capa física es la encargada del acceso al medio físico así como del manejo del hardware necesario para dicho fin. Entre sus responsabilidades más importantes se encuentran las siguientes:

- Manejo del transceiver
- Detección de energía en el canal actual
- Indicación de canal libre para CSMA
- Selección de la frecuencia del canal utilizando FHSS
- Transmisión y recepción de datos

5.3.2.1 Manejo del transceiver

El manejo del transceiver utilizado (ATR 2406), fue implementado de acuerdo a lo especificado por la hoja de datos, en especial respetando los tiempos de configuración de los distintos modos (transmisor o receptor), lo cual era un aspecto crítico para lograr un correcto funcionamiento.

Las rutinas de interfaz con el transceiver fueron implementadas en lenguaje assembler, ya que de esta forma se logra una optimización en los tiempos de ejecución, debido a que se pueden manejar directamente las instrucciones en el microcontrolador, lo cual no sucede al programar las rutinas en lenguaje C.

5.3.2.2 Detección de los niveles de energía

Para detectar el nivel de energía de la señal en recepción se utilizó la salida analógica RSSI disponible en el transceiver. Esta señal indica la potencia de la señal recibida a la frecuencia del canal en que se configuró la recepción, es decir luego de filtrada a la frecuencia correspondiente.

En la gráfica de la figura 5.9 se muestra la curva característica típica de la señal RSSI respecto de la potencia de la señal en recepción.

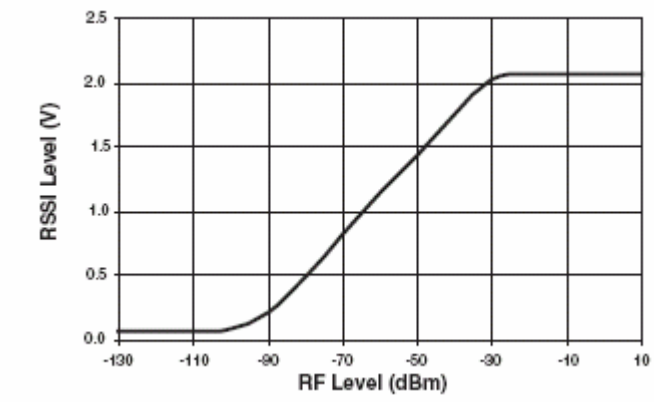


Figura 5.11: Curva característica de la señal RSSI.

5.3.2.3 Indicación de canal libre

Para realizar una transmisión, dado que el protocolo implementa CSMA, los dispositivos deben escuchar el canal pudiendo transmitir cuando este esté libre. Por lo tanto se debe poder "medir" si el canal esta ocupado para decidir si realizar la transmisión en ese momento o esperar un tiempo aleatorio para volver a realizarla. Entonces utilizando la señal RSSI mencionada en el punto anterior, se logra determinar si el canal se encuentra libre para transmitir. Para esto se implementó una rutina, mediante la cual estableciendo un determinado límite para la señal RSSI, se obtiene una salida booleana que indica a la capa MAC el estado del canal.

Para procesar la señal RSSI proveniente del transceiver en el microcontrolador, fue necesario utilizar un conversor A/D el cual éste tiene disponible.

5.3.2.4 La elección del canal

La configuración del transceiver para transmitir o recibir en un determinado canal, se realizó de acuerdo a lo especificado en la sección seis de la hoja de datos del transceiver.

Dado que el protocolo utiliza FHSS la selección del canal se debe realizar conforme con los parámetros del mismo: ancho de salto y tiempo de salto. El transmisor debe cambiar de canal una vez transcurrido el tiempo de salto, y el próximo canal debe ser el canal anterior más el ancho de salto.

Para esto se utilizó un temporizador incluido en el microprocesador, el cual interrumpirá cada vez que se deba realizar el cambio de canal.

5.3.2.5 Transmisión y recepción de datos

Dado que la capa física debe ser capaz de transmitir tramas a través del medio físico, se utiliza un preámbulo para sincronizar la USART del micro con el transceiver, así como indicadores de comienzo de trama y el largo total del paquete.

5.3.2.6 Trama de capa física

La trama de capa física se compone de un encabezado, cuyos campos son los siguientes:

0 - Preamble: Campo utilizado para sincronizar la transmisión entre los dispositivos. Consiste de una onda cuadrada de seis bytes de largo.

1 - Sync: Campo utilizado para la sincronización que consiste de dos bytes fijos en 0xFF.

2 - SOF (Start Of Frame): Indica el comienzo de la trama recibida. Este campo se compone de cuatro bytes con valores fijos, los dos primeros con 0x0F, el siguiente con 0x03 y el último 0x04.

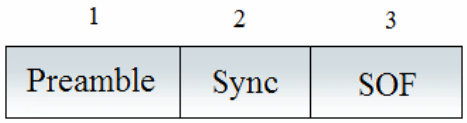


Figura 5.12: Encabezado de capa física

5.3.3 Capa MAC

La capa MAC es la responsable de controlar el acceso al medio físico, para ello se utilizaron varias estrategias las cuales se describen a continuación:

- Sincronización de los dispositivos
- Mecanismo CSMA para el acceso al canal
- Soporte a asociación de dispositivos a la red

5.3.3.1 Sincronización de los dispositivos

Como ya fue mencionado, el hecho de utilizar FHSS, trae como consecuencia la necesidad de mantener el sincronismo de los dispositivos en cuanto a los saltos de canal.

Para lograr dicho sincronismo se incluyó en la trama de capa MAC un campo denominado *Timer_state*, el cual actualiza en cada dispositivo el valor del temporizador que maneja la interrupción para realizar el cambio de canal. De esta manera cada vez que un dispositivo recibe un paquete, sea o no dirigido a este, estará recibiendo también la información a cargar en el temporizador, por lo que ambos dispositivos realizarán el salto de canal prácticamente en simultáneo. Extendiendo este concepto a todos los dispositivos se lograría el sincronismo de la red.

5.3.3.2 Mecanismo CSMA

Para implementar el mecanismo CSMA se requiere del servicio de la capa física para determinar si el canal está libre para realizar una transmisión. Como se mencionó en la sección 5.3.2 en este capítulo, la capa física aporta esta información a partir de la señal RSSI.

La secuencia es la siguiente, la capa física se encarga de informar si el canal está libre, en caso afirmativo, la capa MAC envía los datos hacia la capa física y se realiza la transmisión, en caso contrario, se espera un tiempo aleatorio y se vuelve a chequear el estado del canal, así sucesivamente hasta lograr exitosamente la transmisión.

Para implementar la espera del tiempo aleatorio se utilizó un generador de números aleatorios.

5.3.3.3 Trama de Capa MAC

Los campos que componen la trama MAC son los siguientes:

1-Timer_state: Contiene el valor de actualización del timer correspondiente al salto de canal. Consiste en un byte de largo, lo que equivale a un rango de entre 0 y 20 milisegundos.

2-Checksum: Suma de comprobación de los datos recibidos. Consiste en dos bytes que contienen el valor de control.

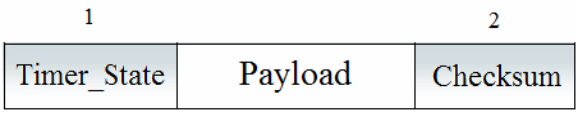


Figura 5.13: Trama de capa MAC

5.3.4 Capa de red

La capa de red es la responsable gestionar la red y sus funciones son las siguientes.

- Direccionar los paquetes que vienen desde capa de aplicación
- Controlar la difusión de paquetes
- Retransmitir paquetes
- Aceptar la asociación y disociación de dispositivos

5.3.4.1 Direccionamiento de paquetes

Los dispositivos asociados a la red tienen un número de identificación el cual les fue asignado por el dispositivo base al momento de ingresar a la red. Dicho número de identificación es utilizado para el direccionamiento de los paquetes, por lo que entonces a cada dispositivo se le asignará un número de identificación el cual será único en la red durante el tiempo en que el dispositivo este presente.

El número asignado por la base a los dispositivos se genera de forma arbitraria y puede ser cualquiera que no este ya asignado a otro dispositivo que se encuentra activo en la red, a excepción de las direcciones reservadas, las cuales se muestran el la tabla 5.1.

Dirección	Reservado para
0x00	Broadcast de la red
0x01	Dispositivo base

Tabla 5.1: Direcciones reservadas.

La dirección asignada a un dispositivo de la red puede ser modificada mediante la utilización del campo de configuración existente en la trama de capa de aplicación. Esto es explicado en detalle en la sección 5.3.5.1.

El dispositivo base lleva registro de los dispositivos remotos presentes en la red, junto a sus dispositivos finales asociados, mediante una tabla con el mapeo de dichas identidades.

5.3.4.2 Control de difusión

El control de la difusión de los paquetes se implementa mediante el método de inundación controlada de la red, el cual fue introducido en la sección 5.1.

Dicho control se divide conceptualmente en dos mecanismos distintos, el control de difusión básico y el control de difusión mediante dirección destino.

5.3.4.3 Control de difusión básico

La difusión de los mensajes en la red es controlada en primer lugar por el campo *Saltos* existente en la trama de capa de red. Todo dispositivo al momento de recibir un paquete chequea el valor del campo *Saltos* de la trama, si dicho valor es cero, el dispositivo no retransmitirá el paquete. Cada vez que se recibe un paquete, si este debe ser retransmitido, el dispositivo que debe realizar dicha retransmisión decrementará en uno el valor del campo *Saltos*.

Otro control necesario para que el sistema funcione correctamente, es no procesar los paquetes que ya fueron recibidos, ya que al implementar una red de difusión, esta situación se produce con frecuencia, y de no realizar ningún control, se compromete la performance del sistema. Por lo tanto todos los paquetes transmitidos tendrán un campo de capa de red, el cual será un número de identificación del paquete. Este campo fue denominado *Num_paquete*, y el valor del mismo es generado aleatoriamente por el dispositivo que crea el paquete y no puede ser cambiado, además de que el dispositivo que deba enviar el mensaje de confirmación de la recepción del paquete, debe etiquetar dicho mensaje con el mismo numero de secuencia. Debido a esto, todos los dispositivos van a manejar dos tablas con los números de identificación de los últimos diez paquetes recibidos, una para los que son respuestas y otra para los que no, y mediante dicha tabla se descartará el paquete recibido, según si el número de identificación se encuentra en la tabla correspondiente. Si el número de paquete coincide con algún valor de la tabla, significa que el paquete ya fue recibido recientemente y por lo tanto será descartado y no seguirá siendo procesado, en caso contrario será aceptado y su número de identificación será agregado a la tabla.

Puede suceder que un paquete recibido sea descartado sin haber sido recibido anteriormente. Esto se debe a que los números de identificación de paquetes son generados aleatoriamente, y por lo tanto el número de un paquete recibido puede coincidir con algún número de identificación almacenado en la tabla de otro paquete recibido recientemente, entonces se interpretará que este paquete ya fue recibido y se descartará. Si bien lo mencionado anteriormente puede suceder, la probabilidad es muy baja, ya que el número de identificación de los paquetes es de largo dos bytes, lo que corresponde a un número entre 0 y 65536.

5.3.4.4 Control de difusión mediante dirección destino

La idea básica del control de difusión mediante dirección destino, es la de evitar que un paquete sea retransmitido innecesariamente mediante la manipulación del campo *Saltos*, dependiendo de si el dispositivo destino se encuentra al alcance. Es decir, si el dispositivo al cual el paquete viene dirigido se encuentra al alcance de una transmisión (de ahora en mas, salto uno), tanto el dispositivo que origina el mensaje, como el dispositivo que lo deba retransmitir, al momento de armar la trama de capa de red le dará valor cero al campo *Saltos*, lo que hará que todos los dispositivos que están al alcance de la transmisión, al momento de recibir el paquete y leer los campos de capa de red, no volverá a transmitirlo debido al valor del campo *Saltos*.

Para lograr lo explicado anteriormente, es necesario conocer que dispositivos se tienen al alcance de una transmisión, es decir a distancia uno. Esto es posible debido al campo *Dir_intermedia* incluido en la trama de capa de red, el cual indica la dirección de identidad

del dispositivo del cual se recibió el mensaje. Cada dispositivo tendrá almacenado en una tabla las direcciones de identidad de todos los dispositivos que tiene a su alcance. Dicha tabla se va formando a medida que se reciben los distintos paquetes provenientes de los dispositivos de los cuales se recibe el mensaje.

5.3.4.5 Retransmisión de paquetes

Según lo expuesto en las secciones anteriores, la retransmisión de los paquetes depende de tres aspectos en particular, hacia quien va dirigido, el valor del campo Saltos y el número de identificación de paquete.

En resumen, un paquete será retransmitido si se cumplen las siguientes condiciones:

- El número de la dirección de identidad del dispositivo destino indicado en el campo *Dir_destino* es distinto al propio.
- El valor indicado en el campo Saltos es distinto de cero.
- El número de identificación de paquete indicado en el campo *Num_paquete* no coincide con ninguno de los números de identificación almacenados en la tabla de los últimos paquetes recibidos.

5.3.4.6 Trama de capa de red

La trama de capa de red se compone de un encabezado, cuyos campos son los siguientes:

1 - Num_paquete: Es el número de identificación del paquete enviado. Es un número seleccionado al azar por el dispositivo que envía el paquete. El receptor mantendrá en todo momento una tabla con los números de identificación de los últimos diez paquetes recibidos, por lo que si el numero del nuevo paquete recibido se encuentra en esa lista será descartado y no se realizará la retransmisión en caso de no ser para si mismo.

2 - Saltos: Indica el número de máximo de dispositivos por los cuales puede ser reenviado el mensaje hasta ser recibido por el destinatario. El dispositivo que recibe el paquete decrementa en uno este campo en caso de retransmitir el paquete. Todo dispositivo no reenviará el paquete cuando este campo este en cero.

3 - Dir_origen: Indica el número de identificación de red del dispositivo que originó el mensaje.

4 - Dir_destino: Indica el número de identificación de red del dispositivo al cual es dirigido el mensaje.

5 - Dir_intermedia: Indica el número de identificación de red del dispositivo del cual se recibe el mensaje, por lo cual todo dispositivo que transmita un paquete, sea o no una retransmisión, coloca su número de identificación en este campo.

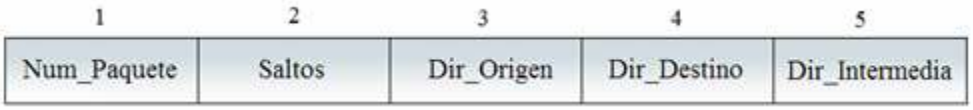


Figura 5.14: Encabezado de capa de red.

5.3.5 Capa de aplicación

5.3.5.1 Configuración de los parámetros de la red

Distintos parámetros importantes de la red pueden ser configurados con el sistema en funcionamiento, es decir sin necesidad de que los dispositivos, tanto los remotos como la base, sean retirados de la red.

Se creó el campo *Tipo* en la trama de capa de aplicación con el propósito de realizar estas configuraciones. Este campo se compone de un byte, en el cual el primer bit (bit 7) indica si se trata de un mensaje de configuración.

En la tabla 5.2 se expone el significado de los bits que conforman el campo *Tipo*. Dichos bits son activos cuando su valor es 1.

En el caso en que el campo *Tipo* indique que el mensaje es de configuración, en la carga útil se incluye, para los casos en que se requiera, los valores correspondientes según la acción de configuración indicada.

En lo que sigue de la presente sección se exponen las distintas funciones implementadas por la capa de aplicación, indicando la forma en que estas utilizan el campo de configuración *Tipo*.

Bit	Función (valor en 1)
7	Indica que el mensaje recibido es de configuración
6	Orden de disociación del dispositivo de la red
5	Asignación y solicitud de dirección de identidad RF
4	Cambio del parámetro *salto_canal*
3	Cambio del parámetro *tiempo_canal*
2	Cambio del parámetro *vida_util*
1	Cambio de la dirección destino por defecto
0	Solicitud y reporte de estado

Tabla 5.2: Byte de configuración.

Función 1: Disociación de dispositivos

Si por cualquier motivo se desea que un dispositivo no siga formando parte de la red, existe la posibilidad de ordenárselo. Esto se logra mediante el bit 6 del campo *Tipo*.

Si se quisiera, por ejemplo, disociar al dispositivo con cierta dirección de identificación de red, simplemente se debe enviar un mensaje de configuración a esa dirección con el bit 6 del campo *Tipo* con valor 0x01. Para esta función no se precisa de ningún dato adicional a agregar en la carga útil. En la tabla 5.3 se muestra el valor correspondiente a cargar en el campo *Tipo*.

Función 2: Asignación de dirección

Existen dos casos en los que se necesita configurar la dirección de identidad de red de un dispositivo, el caso en que éste haya solicitado el ingreso a la red, por lo que precisa que se le asigne una dirección o el caso en que por alguna razón se requiera de cambiar la dirección actual.

Esta funcionalidad la implementa el dispositivo base y se logra mediante el bit 5 del campo de configuración. En la carga útil del mensaje correspondiente se incluye la nueva dirección de identidad de red (ver tabla 5.3).

Campo tipo	Reservado para
10100000	Nueva dirección

Figura 5.15: Asignación de nueva dirección de red.

Función 3: Solicitud de dirección

Esta función es implementada por los dispositivos remotos al momento de solicitar la asociación a la red. La respuesta a esta solicitud de dirección de identidad es generada por la funcionalidad de asignación de dirección utilizada por el dispositivo base.

El valor del campo de configuración *Tipo* es el mismo que en el caso de asignación de dirección, pero la carga útil difiere. En la misma se incluye en primer lugar la cantidad de dispositivos Modbus que se tienen asociados y a continuación las direcciones Modbus correspondientes. En la figura 5.16 se presenta un ejemplo de solicitud de dirección.

Campo tipo	Carga útil		
10100000	Nueva dirección = 2	Dirección Modbus 1	Dirección Modbus 2

Figura 5.16: Solicitud de dirección de un dispositivo remoto con dos dispositivos finales asociados.

Función 4: Cambio de parámetro salto_canal

Esta funcionalidad permite modificar el parámetro de red *salto_canal*, el cual implica la cantidad de canales del salto de canal utilizado en el mecanismo de FHSS.

En la tabla 5.3 se muestra el modo en que se configura el cambio de este parámetro.

Función 5: Cambio de parámetro tiempo_canal

Esta funcionalidad brinda la posibilidad de modificar el parámetro de red *tiempo_canal*, el cual determina el tiempo de permanencia de canal utilizado para el funcionamiento del FHSS.

Una observación importante es que esta función, al igual que la anterior, debe ser precedida por la disociación de todos los dispositivos de la red para luego cambiar el valor de este parámetro en cada uno de los dispositivos, ya que los parámetros del FHSS deben ser iguales para todos.

Es posible cambiar los parámetros del funcionamiento del FHSS simultáneamente, en la siguiente figura (figura 5.17) se muestra esta situación.

Campo tipo	Carga útil	
10011000	Nuevo valor para *salto_canal*	Nuevo valor para *tiempo_canal*

Figura 5.17: Modificación simultanea de los parámetros del FHSS.

Función 6: Cambio de parámetro vida_util

El valor de este parámetro indica al dispositivo el valor por defecto a colocar en el campo *Saltos* de la trama de capa de red al momento de originar un paquete. Puede resultar útil por distintos motivos modificar este valor a algunos dispositivos específicamente.

Función 7: Cambio de dirección destino

Esta función se ejecuta sobre los dispositivos remotos y consiste en la posibilidad de modificar la dirección destino de los mensajes, dado que los dispositivos remotos direccionan todos sus mensajes a una dirección de red fija, la cual generalmente es la del dispositivo base. Esto es así dado que los dispositivos remotos no manejan tablas de mapeo

de direcciones Modbus-RF y por lo tanto cualquier comunicación hacia un dispositivo debe ser direccionada, a nivel de la red inalámbrica, primero hacia la base para que ésta resuelva dicho mapeo de direcciones. Por otro lado, puede ser útil cambiar la dirección destino de algún dispositivo específico para determinada funcionalidad especial que se requiera implementar.

Función 8: Solicitud de estado

La solicitud de estado es una función realizada por el dispositivo base y su propósito es obtener determinada información de los dispositivos remotos, los cuales como consecuencia de esta solicitud, responden a la base. Esta función es un servicio de la capa de aplicación hacia la aplicación específica que se ejecute en la PC cliente.

La información que se pretende obtener de los dispositivos remotos consiste en el valor que presenta el parámetro *vida_util*, la cantidad de dispositivos Modbus que tienen asociados y la dirección Modbus de cada uno de ellos.

En el mensaje originado por la base, el campo de configuración Tipo debe tener el valor 0x81 y no lleva carga útil. Esto se observa en la tabla 5.3.

Función 9: Reporte de estado

El reporte de estado se realiza en respuesta a la solicitud de estado ejecutada por el dispositivo base. En el reporte, los dispositivos remotos informan, como se mencionó en el punto anterior, el valor del parámetro *vida_util*, la cantidad de dispositivos Modbus asociados y sus respectivas direcciones. Esta información se encuentra contenida en la carga útil del mensaje correspondiente al reporte de estado, en el orden indicado en la figura 5.18 y el valor del campo de configuración *Tipo* debe ser 0x81 al igual que para el mensaje de solicitud de estado.

Campo tipo	Carga útil				
10000001	Valor de *vida_util*	N° de dispositivos Modbus	Dirección Modbus 1	...	Dirección Modbus N

Figura 5.18: Configuración en mensaje de reporte de estado.

Función	Valor del campo *Tipo*	Contenido de carga útil
Disociación de dispositivos	11000000	Sin carga útil
Asignación de dirección	10100000	Nueva dirección de red
Solicitud de dirección	10100000	Cantidad de dispositivos Modbus asociados junto con sus respectivas direcciones Modbus
Cambio de parámetro *salto_canal*	10010000	Nuevo valor deseado para *salto_canal*
Cambio de parámetro *tiempo_canal*	10001000	Nuevo valor deseado para tiempo*_canal*
Cambio de parámetro *vida_util*	10000100	Nuevo valor deseado para *vida_util*
Cambio de dirección destino	10000010	Nueva dirección destino deseada
Solicitud de estado	10000001	Sin carga útil
Reporte de estado	10000001	Parámetro *vida_util* actual, cantidad de disp. Modbus asociados junto con sus respectivas dir. Modbus

Tabla 5.3: Valores de configuración.

5.3.5.2 Asociación automática de dispositivos

La asociación de un nuevo dispositivo remoto se realiza automáticamente. Cuando un dispositivo remoto se enciende se pone a escuchar en un canal fijo, o en el siguiente si el mismo estuviese interferido luego de un cierto tiempo configurable. Dado que el número de canales disponibles es primo, luego de 89 saltos se habrán recorrido todos los canales.

El dispositivo, una vez que recibe una trama válida, comienza a saltar de canal sincronizado con el resto. Luego que el sincronismo ha sido alcanzado, el dispositivo remoto envía un mensaje hacia el dispositivo base, (el cual tiene una dirección fija dentro de la red), para solicitar asociarse a la red, junto con la información del o los dispositivos finales que tiene asociados. Finalmente, el dispositivo base le confirma su asociación a la red mediante un mensaje informando de su dirección de identidad para la red.

De dispositivo remoto hacia dispositivo base:

Campo DIR_DESTINO	Campo tipo	Carga útil			
10000001	10100000	N° de dispositivos Modbus	Dirección Modbus 1	...	Dirección Modbus N

De dispositivo base hacia dispositivo remoto:

Campo DIR_DESTINO		Campo tipo	Carga útil
00000000	...	10100000	Nuevo dirección RF

Figura 5.19: Secuencia de solicitud de dirección.

Debido a que el dispositivo remoto hasta ese momento no tiene asignada ninguna dirección, se necesitó lograr que de algún modo se indique que el mensaje en respuesta al pedido de dirección es dirigido a él. La forma en que esto se logra es mediante la coincidencia del campo *Num_paquete* del mensaje original de pedido de dirección y del mensaje de respuesta. De este modo, el dispositivo remoto reconoce que ese paquete es la respuesta a su pedido de dirección y toma como su dirección de identificación en la red, a la que contiene dicho mensaje.

Dado que cabe la posibilidad de que un dispositivo en la periferia de la red, no reciba mensajes con la suficiente frecuencia como para transmitir en todos los canales utilizados, en un periodo de tiempo no muy largo, puede suceder que al intentar agregar un nuevo dispositivo remoto el cual solo tiene a éste a su alcance, no se tenga éxito. Para evitar que esto suceda el dispositivo base implementa un mecanismo mediante el cual obliga a todos los dispositivos remotos, uno a la vez, a realizar un ciclo de transmisiones por todos los canales (89 saltos de canal). De esta forma se logra aumentar la probabilidad de que nuevos dispositivos puedan sincronizarse aunque solo tengan un dispositivo a su alcance.

5.3.5.3 Mensajes de sincronismo

De forma de darle robustez al sincronismo de la red, el dispositivo base se encarga de enviar cada un determinado tiempo, el cual es configurable, un paquete de control, el cual fue llamado mensaje de sincronismo, con la información necesaria para mantener el sincronismo. Dicha información se encuentra en un campo de la trama de capa MAC como fue mencionado anteriormente en este capítulo.

Este mensaje de sincronismo consiste simplemente en un mensaje sin carga útil y con dirección de difusión de la red, al cual se le asigna en el campo *saltos* el valor de la cantidad de saltos existente entre el dispositivo base y el dispositivo más alejado de éste. De esta forma se logra que el mensaje alcance a todos los dispositivos remotos de la red, haciéndoles llegar la información de sincronismo.

5.3.5.4 Confirmación de recepción de mensajes

El reconocimiento de la recepción de los mensajes se implementa mediante paquetes denominados ACK. Una vez que un dispositivo inicia un mensaje, espera recibir un paquete ACK, con igual valor en el campo *Num_paquete* que dicho mensaje, como respuesta por parte del dispositivo destino. De esta forma se logra determinar si el mensaje alcanzo exitosamente su destino.

Luego de un tiempo (el cual es configurable), si no se recibe un paquete ACK como respuesta, el dispositivo que originó el mensaje asume que este no alcanzó el destino y acto seguido vuelve a transmitir el mensaje. Así sucesivamente hasta que se reciba el paquete ACK esperado.

El ACK es simplemente un paquete sin carga útil que contiene el valor 0x01 en el campo *Confirmación* perteneciente a la trama de capa de aplicación. Todos los mensajes que no son de confirmación de recepción tienen el valor 0x00 en el campo *Confirmación*.

Mensaje original:

Campo NUM_PAQUETE		Campo CONFIRMACION	Carga útil			
00110011	...	00000000	0xXX	0xXX	...	0xXX

Confirmación de la recepción del mensaje:

Campo NUM_PAQUETE		Campo CONFIRMACION	Carga útil
00110011	...	00000001	NULA

Figura 5.20: Confirmación de la recepción de un mensaje.

5.3.5.5 Trama de capa de aplicación

La trama de capa de aplicación se compone de un encabezado, cuyos campos son los siguientes:

1-Num_bytes: Indica la cantidad de bytes de la carga útil.

2-Confirmación: Indica si el mensaje es una confirmación de recepción. Su valor en 0x01 implica que es una confirmación y valor en 0x00 lo contrario.

3-Tipo: Permite distinguir si la trama recibida es de configuración o de datos.

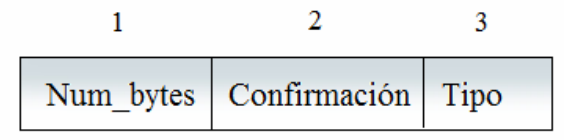


Figura 5.21: Encabezado de capa de aplicación.

5.3.6 Trama completa del protocolo WiDO 1.0

En la figura a continuación se muestra la secuencia de armado de paquetes con sus respectivas tramas.

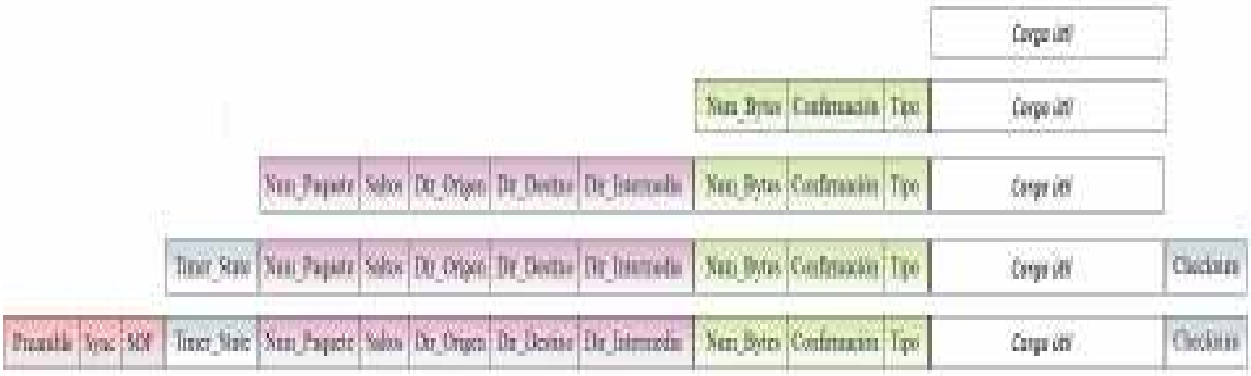


Figura 5.22: Secuencia de armado de tramas.

Capítulo 6

Descripción del hardware

6.1 Introducción

En esta sección se describe el hardware que conforma el sistema de comunicación. Básicamente se distinguen dos grandes módulos, el dispositivo Base y el Remoto (ver 3.2.1 y 3.2.2). Para implementar estos módulos se utilizaron y desarrollaron distintos dispositivos de hardware. A continuación se muestra brevemente como están conformados estos módulos.

Dispositivo Remoto (RFB remoto)

Para la implementación de este dispositivo se diseño una placa que llamaremos Radio Frequency Board (RFB). La RFB es un dispositivo programable capaz de comunicarse a través del protocolo WiDO 1.0, el cual posee interfaces RS-232 y RS-485.

Dispositivo Base

Recordemos que este módulo provee comunicación entre Ethernet y el sistema inalámbrico. El módulo se desarrollo en base a dos dispositivos más básicos, un RCM3700 y una RFB.

El RCM3700 es un sistema embebido basado en un microprocesador Rabbit 3000, dicho modulo tiene conectividad Ethernet y puertos serie entre otros.

Luego interconectando una RFB con un RCM3700 tenemos un modulo que integra la comunicación Ethernet e inalámbrica utilizando el protocolo Wido v1.0 desarrollado.

Para la interconexión de estos dispositivos se desarrollo una placa que permite acceder a los puertos seriales del modulo RCM3700 y se desarrollo un pequeño protocolo de comunicación serie (ver 7.3.2.1) que permite a estos módulos actuar como una sola unidad. De esta forma se logra un modulo Base que puede transmitir datos a través del protocolo Wido v1.0 o utilizando Ethernet.

Fuente de alimentación

Tanto el dispositivo remoto como el dispositivo base necesitan de una fuente de continua para su alimentación. Como una de las necesidades del sistema es que los dispositivos puedan ser alimentados desde la red eléctrica de 220AC, se diseño una pequeña fuente de 10Vdc no regulada que se energiza de la red.

En las próximas secciones del capitulo se profundiza sobre el diseño de estos módulos mencionados. Luego en el anexo B se detallan los procesos que se siguieron para la construcción de los mismos.

6.2 RFB (Radio Frequency Board)

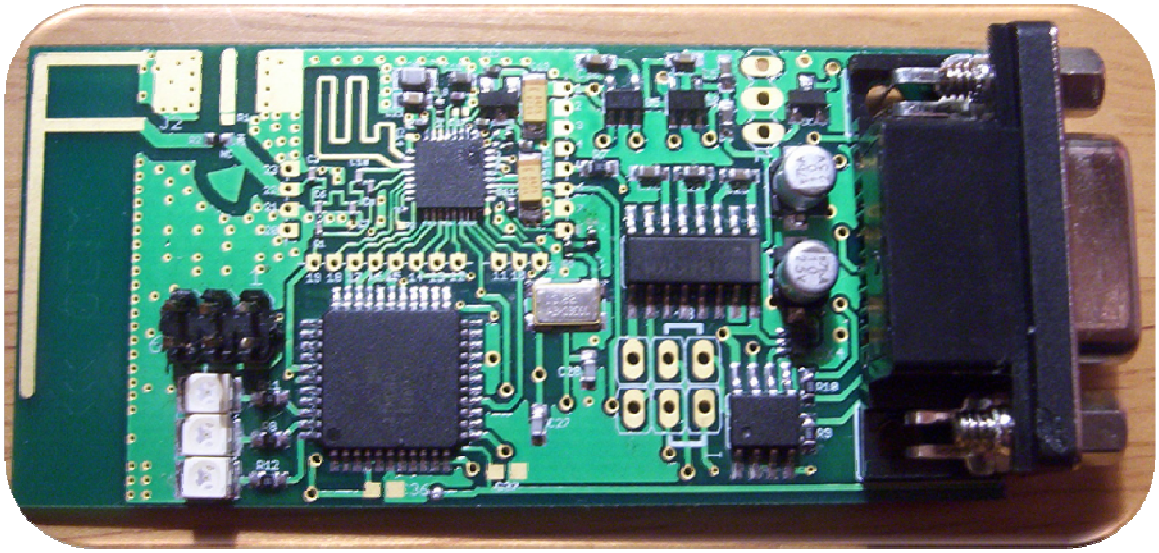


Figura 6.1: RFB plano superior.

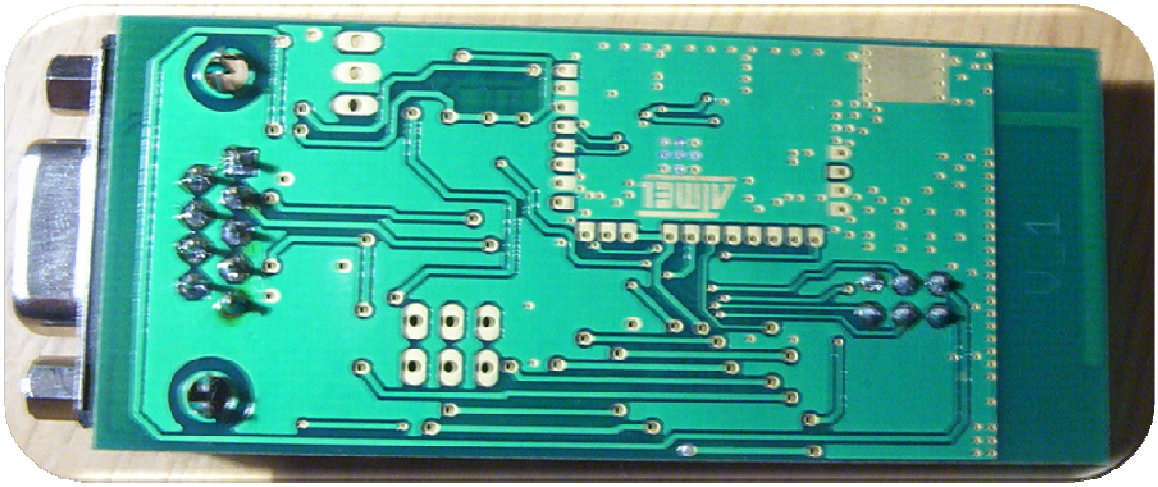


Figura 6.2: RFB plano inferior.

6.2.1 Características del diseño

Este dispositivo esta basado en un transceiver RF (ATR 2406) controlado por un microprocesador ATMega164P. La RFB posibilita las comunicaciones utilizando el protocolo Wido v1.0. Además posee adaptación de las señales que provienen del modulo USART del micro lo cual permite la comunicación sobre los estándares RS-232 y RS-485.

La placa está construída en fibra de vidrio con máscara antisoldante, doble faz y agujero metalizado. Las pistas están bañadas en oro y el grosor de la placa es de 500 micras, se optó por estas características para asegurar una buena adaptación de impedancia de la antena.

A continuación se presentan las características generales de la RFB:

- Posee un Transceiver RF capaz de transmitir y recibir en la banda libre ISM 2.4GHz (ATR2406)
- Antena Tipo F en placa con adaptación de impedancia
- Conector RCA para antena externa con adaptación de impedancia de 50 ohm
- Micro controlador AVR AtMega164p
- Adaptador de señales Max232
- Adaptador de señales Max485
- Conector para programación onboard del microcontrolador
- Conector DB9 para señales Rs485 y Rs232
- Conector para alimentación externa
- Voltaje de alimentación: 6V-14V DC
- Reguladores de voltaje internos para alimentación de integrados, 4V (LP2985) y 5V(AN8005)
- Diodo para protección de inversión de fuente

El diseño del impreso fue realizado con la herramienta de software EAGLE en su versión libre. Utilizando este software se obtuvieron los archivos *gerber* los cuales son utilizados por el fabricante para la construcción del impreso (ver Anexo B).

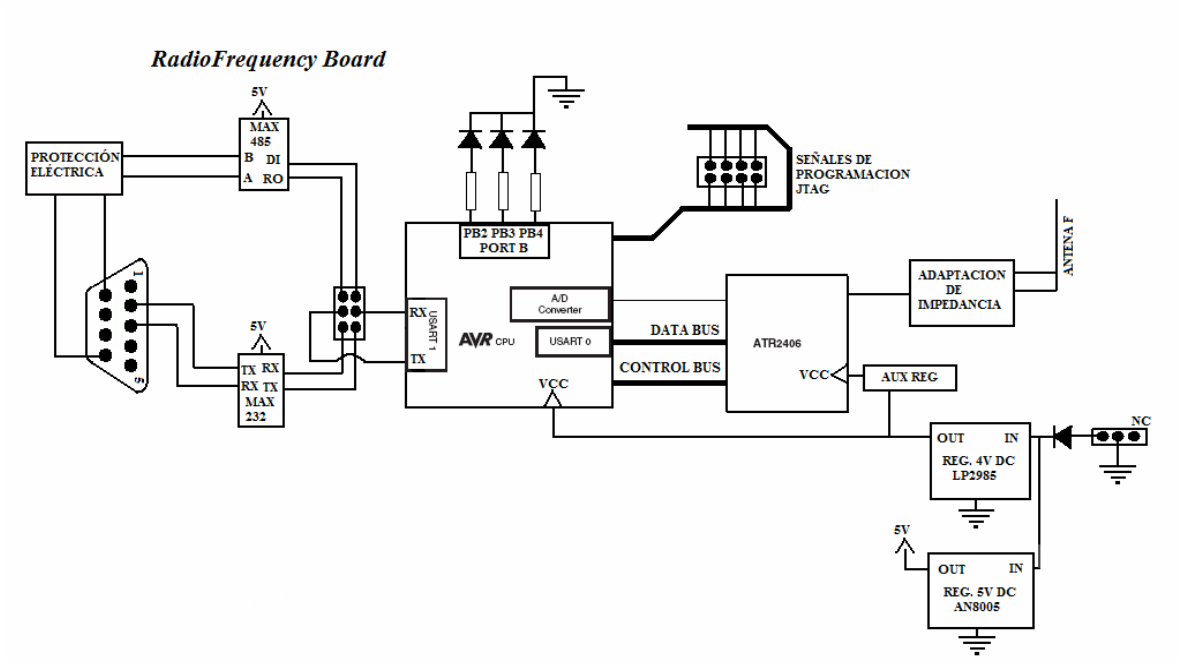


Figura 6.3: Esquema de conexiones de la RFB.

6.2.2 Elección del transceiver

El transceiver utilizado es el ATR 2406 de Atmel el cual fue seleccionado de acuerdo con las necesidades del protocolo implementado. En esta decisión se tuvieron en cuenta muchos factores los cuales se describen en detalle a continuación:

- El ATR 2406 es un transceiver diseñado para soportar Frequency Hopping utilizando una portadora en 2.4 GHz e implementa modulación GFSK. Estas características se ajustan perfectamente a esta aplicación (ver sección 4.2 y 4.3). [6.1]
- Posee una tasa de transmisión de datos aceptable (1152 Kbps). Este punto en principio no estaba especificado, pero se tomo como criterio elegir el transceiver que posea la mayor tasa posible.

- Existe un diseño de antena (hecha en PCB) optimizada para aplicaciones con este transceiver.
- El fabricante provee toda la documentación necesaria, incluida toda la información para la fabricación de la antena y el circuito de adaptación de impedancia. [6.2]
- El costo no es elevado (U$S 3.83) y existe una gran disponibilidad del mismo en el mercado.
- Se consiguieron muestras del transceiver con facilidad.
- Otro punto importante fue que para el desarrollo del software, contamos con un kit de desarrollo de transmisión RF provisto por Atmel en forma gratuita; el cual tiene un microcontrolador ATMega 88 que maneja el transceiver ATR 2406. La placa de desarrollo cuenta con un display LCD el cual sirve para depurar el software. El *firmware* incluido en el kit ayudó a comprender la interfaz entre el micro y el transceiver y permitió evaluar la performance del transceiver para poder verificar que cumplía con las necesidades del caso. [6.3] [6.4]

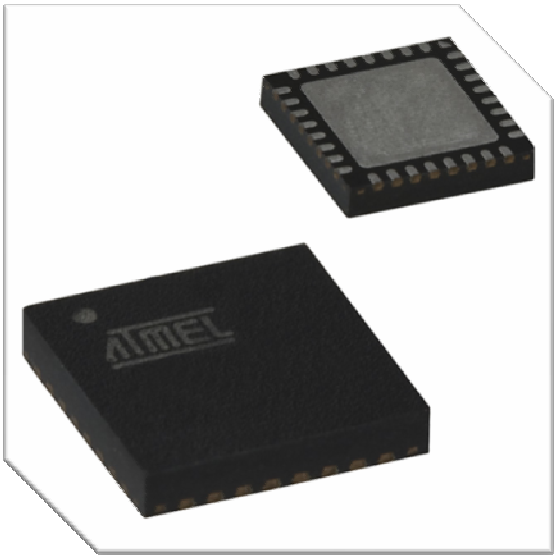


Figura 6.4: ATR2406.

6.2.3 Elección del Microcontrolador

Para seleccionar el microprocesador se tuvo en cuenta como primer requisito que debía ser totalmente compatible con el transceiver elegido; la característica excluyente para esto es que posea un modulo USART (Ver 6.2.4.). Además se necesita otro modulo USART libre para permitir que la RFB se pueda comunicar con los actuadores. En conclusión el microprocesador debe tener 2 módulos USART. También es necesario que el microcontrolador tenga un conversor analógico digital (ADC) el cual es necesario para tomar una señal analógica que proviene del transceiver llamada RSSI (*Received Signal Strength Indicator*) (ver secciones 5.3 y 6.2.4).

Para seleccionar la capacidad de memoria flash se tomo como criterio la memoria que posee el microcontrolador Atmega88 (8K Bytes), ya que este es el que se usa en el kit de desarrollo. Además por experiencias anteriores se considero que poseer 8K Bytes de Flash es suficiente para esta aplicación.

Como el transceiver fue diseñado por Atmel se oriento la búsqueda hacia microprocesadores de esta marca, que además tiene buena disponibilidad en nuestro mercado. Dentro de los productos que provee la empresa Atmel en un principio los candidatos a elegir eran: El ATMega 88 ya que era usado por el kit de desarrollo, y el ATMega 16 ya que contábamos con muestras del mismo. De todas formas ambos fueron descartados porque proveen un único modulo USART. Luego se planteamos la elección entre otros dos microprocesadores: ATMega 128 y el ATMega 164P, que si cumplen la exigencia de tener dos USART; y además poseen al menos un conversor analógico – digital. Finalmente se decidió diseñar con el ATMega 164P que aunque tiene menos capacidad en memoria Flash que el ATMega 128, no afecta a nuestra aplicación ya que los 16 KB de Flash que provee son suficientes. Dicho integrado se alimenta con 2.7V a 5.5V al igual que el transceiver. [6.5]

Características ATMega 164P

Figura 6.5: ATMega 164P.

Arquitectura interna RISC:

- 20 MIPS @ 20 MHz
- 32 registros de propósito general

Memorias:

- FLASH 16 Kbytes
- EEPROM 512 bytes
- SRAM 1 Kbytes

Periféricos:

- 2 Timers/Counters de 8 bits y 1 de 16 bits con prescalers separados, función de comparación y captura.
- 6 canales PWM
- Conversor analógico digital de 10 bits, 8 canales
- 2 USART
- 1 SPI
- Watchdog timer con oscilador independiente
- Comparador analógico

I/O y encapsulado:

32 líneas de entrada y salida programables
Encapsulados 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF

6.2.4 Interconexión Transceiver - Microcontrolador

Para intercambiar datos entre el microcontrolador y el transceiver se tienen dos opciones las cuales requieren del uso de periféricos del micro. Es posible hacer este intercambio de datos utilizando un periférico USART o una interfaz SPI. Ambos métodos tienen ventajas y desventajas.

El único problema potencial que se puede tener es que las ráfagas de datos que se reciben o transmiten no deben ser interrumpidas. Este problema se puede evitar manejando estos datos con un modulo USART que posea doble buffer o escribiendo cuidadosamente el código en el caso de utilizar un modulo SPI. Esta segunda posibilidad de interconexión es más compleja ya que aumenta mucho el uso del CPU. En primer lugar, la SCK (señal de reloj) tiene que ser generada por software y el hardware no es compatible con una de las posibles velocidades de transmisión de datos que brinda el transceiver. En segundo lugar, la

sincronización del flujo de datos a la entrada también debe ser manejada por software. Tomando en cuenta estas consideraciones se optó por utilizar el modulo USART. Utilizando el módulo USART se puede alcanzar una velocidad de transmisión de 1.152Mbps. [6.6] [6.7].

En la imagen de la figura 6.6 se muestran las señales que se deben interconectar para manejar el transceiver utilizando la interfaz USART.

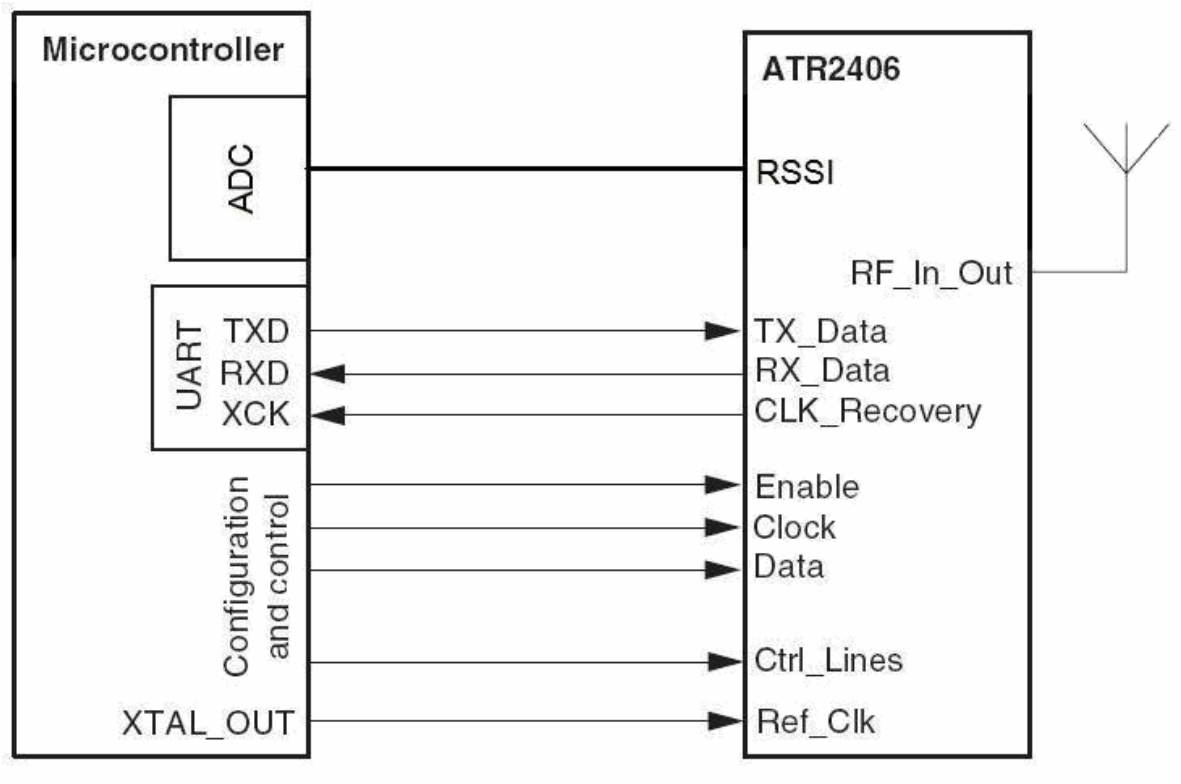


Figura 6.6: Interconexión MICRO-TRANSCEIVER.

ATR2406		***ATMEGA164P***	
Número de PIN	**Descripción**	**Número de PIN**	**Descripción**
3	RSSI Out	MEGA37	(ADC0)PA0 I/O
6	TEST2 CLKRECOVERY	MEGA40	(T0/XCK)PB0 I/O
24	RX_ON In	MEGA32	(ADC5)PA5 I/O
25	TX_ON In	MEGA31	(ADC6)PA6 I/O
26	OLE In	MEGA30	(ADC7)PA7 I/O
28	RX_DATA Out	MEGA10	(TXD)PD1 I/O
29	TX_DATA In	MEGA9	(RXD)PD0 I/O
30	CLOCK In	MEGA40	(T0/XCK)PB0 I/O
31	DATA In	MEGA9	(RXD)PD0 I/O
32	ENABLE In	MEGA15	(ICP)PD6 I/O

Tabla 6.1: Conexiones ATMEGA164P-ATR2406.

El conversor ADC del micro se cableó directamente a la señal RSSI que provee el transceiver, dicha señal representa la potencia de señal RF que se esta recibiendo. Utilizando la interfaz ADC del micro, se traduce esta señal y se puede medir la potencia (ver Figura 5.11).

6.2.5 Antena On board y adaptación de impedancia

El fabricante del chip ATR 2406 provee un diseño de antena para este transceiver por lo que se decidió utilizar este para nuestro hardware. La antena es modelo F y esta diseñada para construirla sobre un circuito impreso. La adaptación de impedancia para la misma también es provista por el fabricante. Esta antena se utiliza ampliamente porque es razonablemente compacta, tiene un diagrama de radiación omnidireccional, buena eficiencia y es muy sencilla de implementar [6.8].

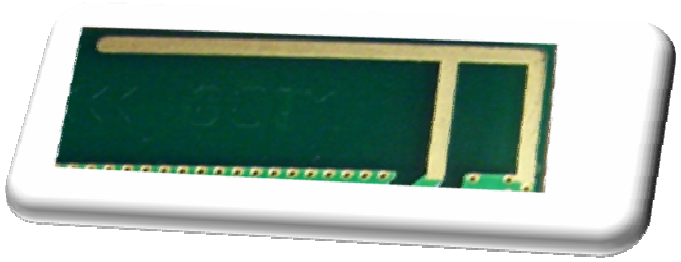


Figura 6.7: Antena F construida sobre PCB de la RFB.

En la figura 6.7 se muestra un ejemplo del patrón de radiación de una antena F. En rojo se representa la polarización vertical y en azul se representa la polarización horizontal ambos en dBi.

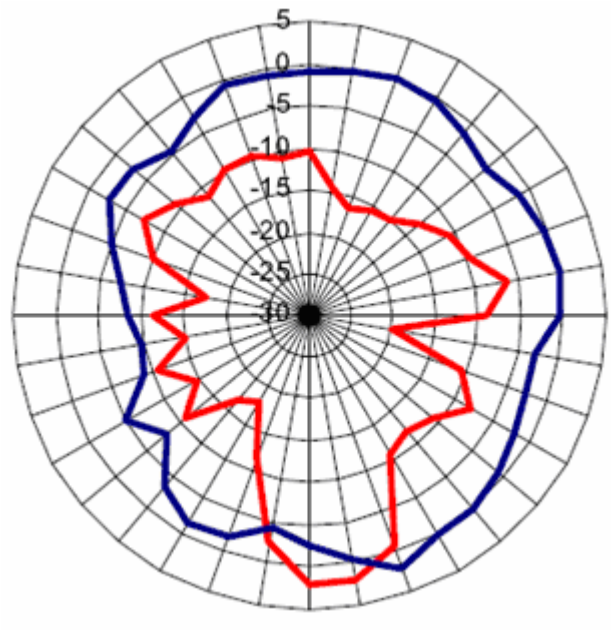


Figura 6.8: Patrón de radiación.

Esta antena es un mono polo, con entrada *single-ended* y el conector de entrada del transceiver es diferencial por lo que se utiliza un *balun* para balancear la señal. El *balun* convierte la entrada *single-ended* en una salida diferencial para una determinada adaptación de impedancia. Es decir, la tensión de salida en cada pin es de igual magnitud, pero de fase opuesta.

La adaptación de impedancia esta conformada por 3 micro-strip, que básicamente son inductancias hechas con pistas en la plaqueta.

En la figura 6.9 se muestra el esquemático eléctrico de la adaptación de impedancia y se pueden observar los 3 micro-strip y el Balun.

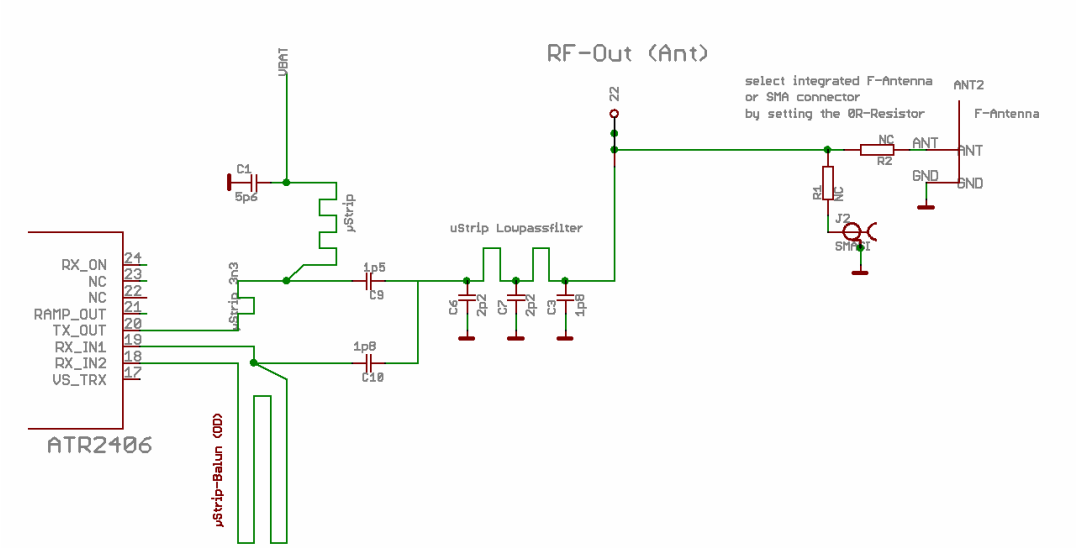


Figura 6.9: Esquema eléctrico de adaptación de impedancia.

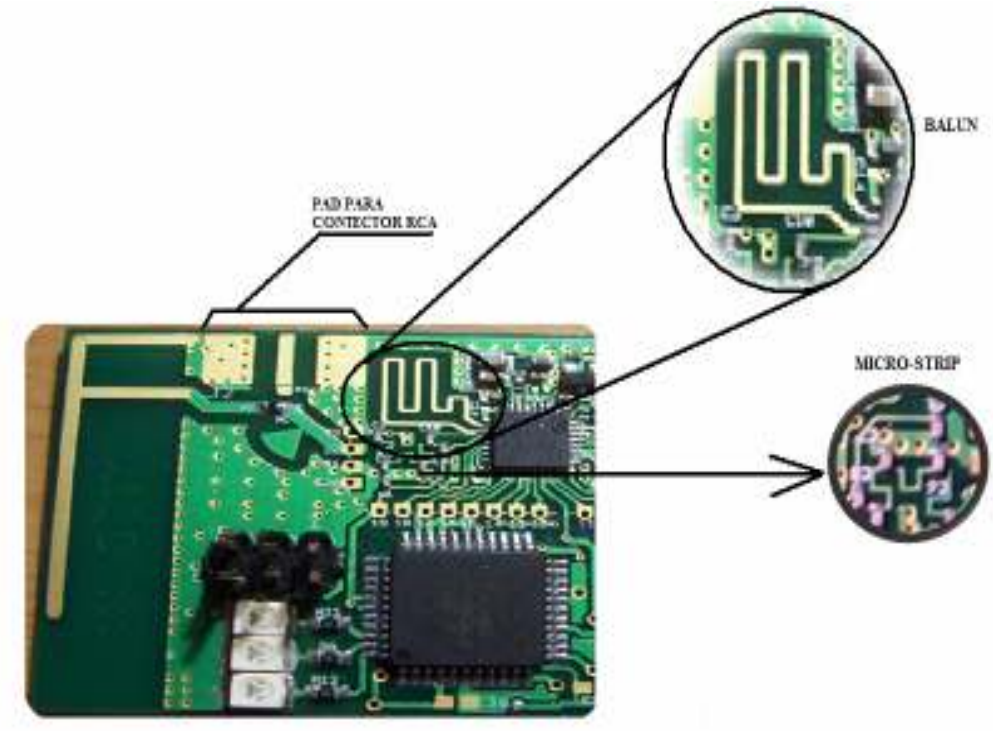


Figura 6.10: Adaptación de impedancia y balun de la RFB.

6.2.6 Conversores y adaptadores de señales

6.2.6.1 MAX485

Para poder conectar la RFB directamente a un bus de comunicaciones RS-485, se deben adaptar las señales que provienen de la USART. El bus RS-485 es un par trenzado de cobre donde las señales se transmiten en forma diferencial. En cambio las señales de la USART son del tipo TTL 5V con referencia, es decir que se usan dos cables, uno como referencia y el otro para transmitir las señales. Por esto es necesario utilizar un conversor MAX485 que adapte las señales. [6.9]

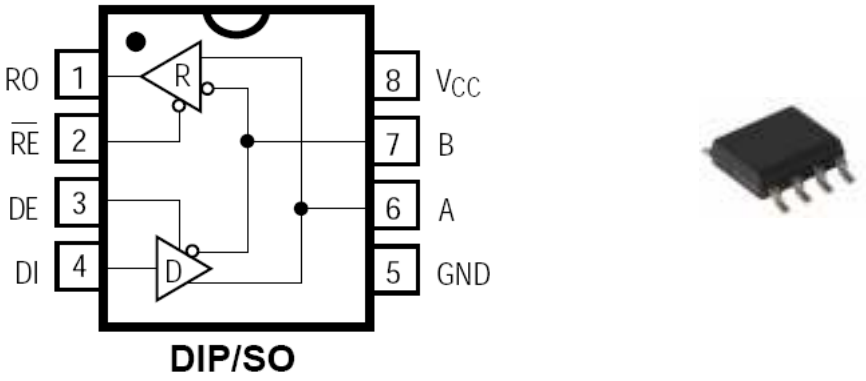
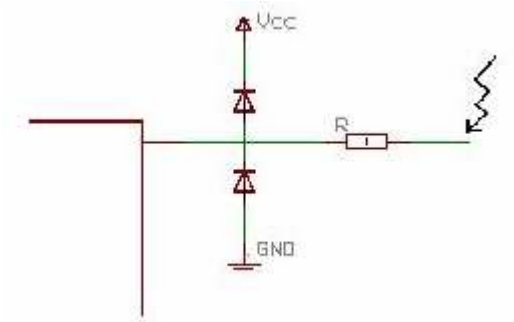


Figura 6.11: MAX 485.

La conexión de este chip al micro es muy sencilla. Los pines RO y DI se conectan directamente a los pines RX y TX de la USART respectivamente. Luego se cortocircuitan los pines RE y DE, conectándolos a un pin de entrada y salida del micro. Esta señal se usa para activar la recepción o la transmisión. Al estar cortocircuitados se asegura que no se esta recibiendo y transmitiendo al mismo tiempo. Luego se conecta A y B a través de una resistencia a los pines 6 y 9 del conector DB9 que posee la RFB; en estos pines es donde se conecta el bus RS-485. Esta resistencia en serie junto con dos diodos, actúan como protecciones de sobre corriente y sobre tensión.



Para que circule una I = 500 mA, si R = 680 Ω debería existir un potencial en la línea de 320 V. Además se protege la tensión a la entrada del integrado gracias al diodo. V_{ent} є $(-V_D, V_{cc} + V_D)$

Figura 6.12: protección de entrada al MAX485.

6.2.6.2 MAX232

Pensando en poder conectar una RFB a otro bus distinto al 485 se decidió incorporar en su diseño un conversor MAX232 el cual nos permite que los datos puedan ser transmitidos por un bus RS-232. [6.10] Este chip se conecta directamente al micro, los pines RX y TX de la USART van a los pines 9 y 10 del MAX232 respectivamente. También las señales de salida y entrada 7 y 8 se conectan a los pines 3 y 2 del conector DB9 respectivamente. Esta conexión no es arbitraria ya que respeta el estándar RS-232. En el caso anterior, se conectaron las salidas del MAX485 a pines del DB9 que no se utilizan en las comunicaciones sobre RS-232. También se necesita conectar una serie de capacitores que se muestran en el esquema de la figura 6.14.

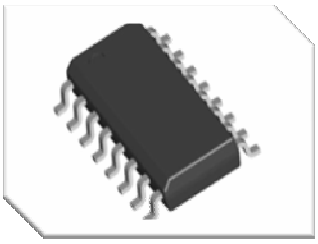


Figura 6.13: MAX232.

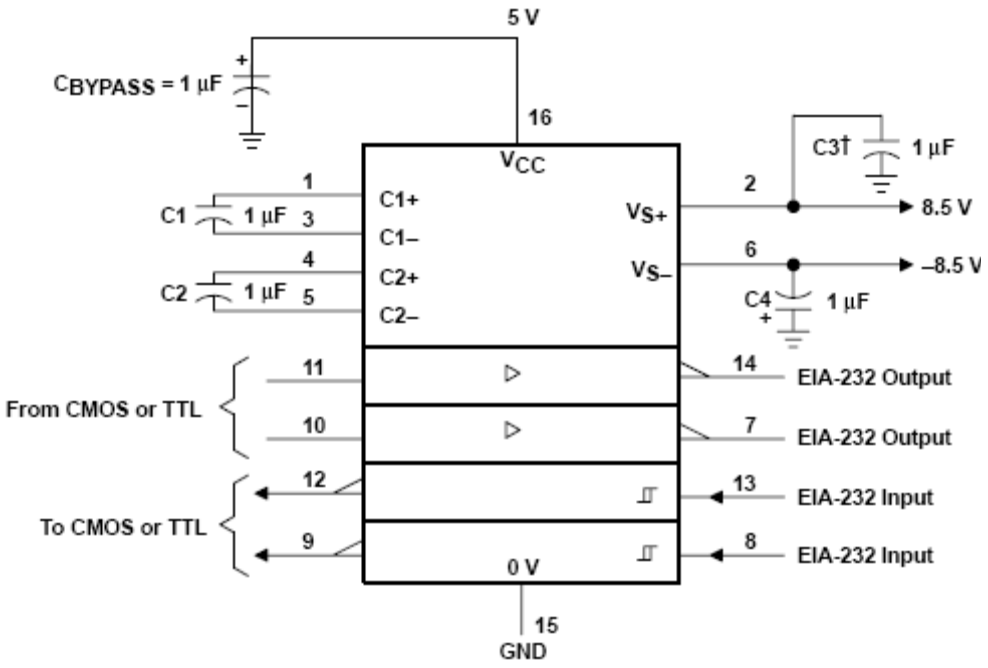


Figura 6.14: Esquema de conexiones para el MAX232.

En la figura 6.14 se muestra como se logran conectar estos dos Conversores al micro sin tener peligro de cortocircuito. Notar que se deben conectar 2 jumpers entre los pines 1 y 2 para que la USART quede cableada al MAX485 y si se conectan entre los pines 2 y 3 la USART queda cableada al MAX232.

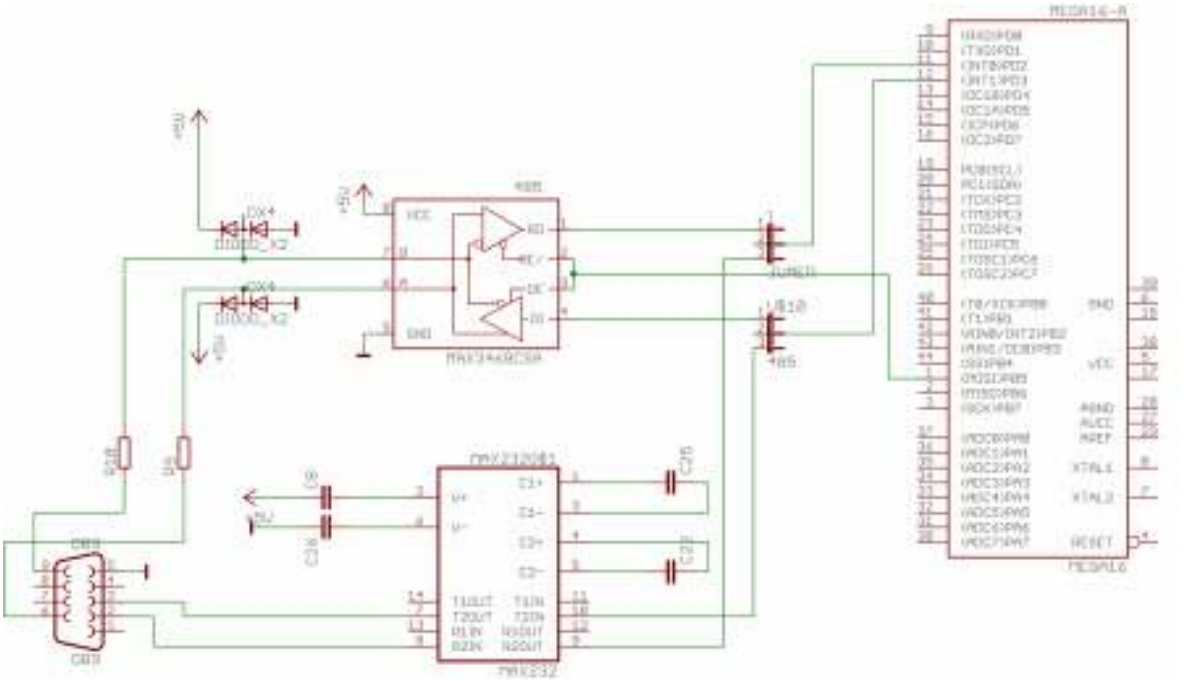


Figura 6.15: Interconexión Micro-MAX232-MAX485.

6.2.7 Alimentación y reguladores de voltaje

Para la alimentación se necesita contar con dos voltajes distintos, tanto el micro como los conversores MAX232 y MAX485 aceptan una alimentación de 5V DC, mientras que el transceiver debe ser alimentado con un voltaje que se encuentre en un rango de 2.9V a 3.6V DC.

Al tener que utilizar dos reguladores distintos se decidió alimentar con el mismo regulador al micro y al transceiver. De esta forma ambos manejan un voltaje en común. La ventaja de esto es que al alimentar el micro con menor voltaje se disminuye la corriente tomada al regulador, optimizando el consumo de potencia (ver figura 6.15). Esto es posible ya que el rango de voltajes que maneja el micro es de 2.7V a 5.5V. Además el ATR 2406 permite ampliar su rango de voltaje a 4.6V si se utiliza el regulador de voltaje auxiliar que se describe en su hoja de datos (ver figura 6.19). Este regulador auxiliar es utilizado en el kit de desarrollo y recomendado en las notas de aplicación del transceiver, por lo que se incluyó al diseño. Teniendo en cuenta lo anterior, los cálculos de consumo y la disponibilidad en el mercado se optó por utilizar el regulador LP2985AIM5 de 4V. Este

voltaje cumple con los rangos de entrada del micro y el transceiver. [6.5] [6.1] [6.13] [6.14].

Para determinar la corriente que debe entregar el regulador se realizó la siguiente estimación:

$$I_{OUT\ MAX} = I_{INMAX\ TRANCEIVER} + I_{INMAX\ MICRO}$$

El consumo de corriente para el transceiver se estimó en un máximo de 57mA cuando se encuentra en el intervalo de recepción (según hoja de datos 6.3).

La corriente máxima del ATMega 164P se puede calcular sumando la corriente en modo activo más el uso de los módulos de entrada y salida.

$$I_{INMAX\ MICRO} = I_{ACTIVE} + I_{I/O\ MODULES.}$$

La corriente que toma el micro en modo activo se estima según la gráfica de la figura 6.16.

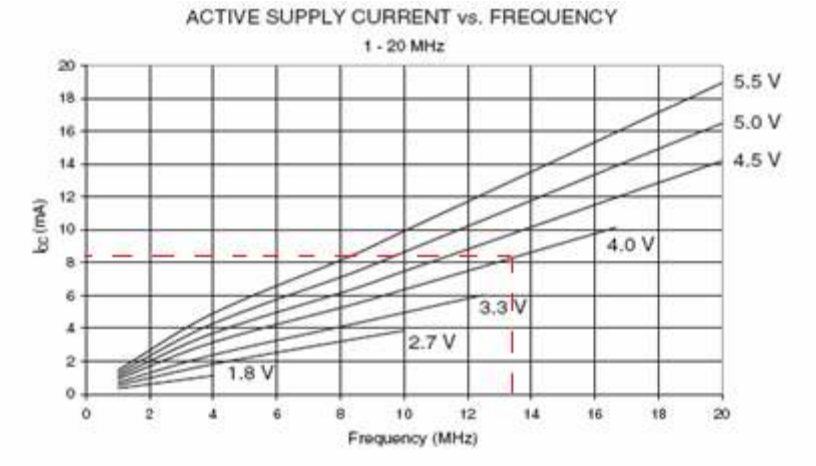


Figura 6.16: Consumo de corriente de la fuente vs. Frecuencia.

Se observa que el consumo normal del micro @ (4V; 13.8MHz) es de aproximadamente 8mA. Para estimar el consumo final hay que sumar a esto el consumo de los periféricos de entrada y salida.

Para calcular $I_{I/O\ MODULES}$ se utilizó la tabla 6.2. En esta se presentan los consumos porcentuales de cada modulo, comparados con el consumo en modo activo e inactivo (*idle*). En esta aplicación solo se utilizan los dos módulos USART, dos Contadores y un conversor ADC. De todas formas, para el cálculo se supuso que están todos los módulos activos.

PRR bit	Additional Current consumption compared to Active with external clock (see Figure 27-1 on page 337 and Figure 27-2 on page 338)	Additional Current consumption compared to Idle with external clock (see Figure 27-6 on page 340 and Figure 27-7 on page 340)
PRUSART1	1.8%	6.9%
PRUSART0	2.4%	9.1%
PRTWI	5.4%	21.2%
PRTIM2	4.6%	17.4%
PRTIM1	2.7%	10.5%
PRTIM0	1.6%	6.0%
PRADC	4.5%	16.8%
PRSPI	3.3%	12.9%

Tabla 6.2: Consumo relativo de los periféricos del ATMEGA164P.

Sumando los porcentajes en modo activo se llega a que el consumo total de los módulos es un 26.3% de los 8mA que se utilizan en modo activo. Por lo tanto $I_{I/O\ MODULES} = 2.1mA$.

También se debe agregar a este consumo las corrientes que toman los conversores MAX232, MAX485 y los 3 LEDs indicadores.

La corriente que toman los conversores esta en el orden de los µA y se puede despreciar.

Para el caso de los LEDs tenemos:

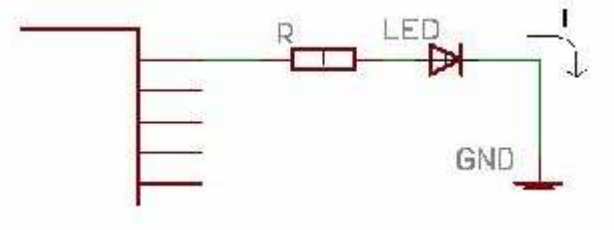


Figura 6.17: Esquema de conexión LED-MICRO.

I = 4 – 0.6 / 1k = 3.4 mA para cada LED entonces el consumo total de los LED´s es el siguiente:

$$I_{LEDS} = 10.2mA$$

En conclusión la corriente $I_{I/O\ MODULES} = 2.1mA + 10.2mA = 12.3mA$ y el consumo máximo para el micro es de $I_{INMAX\ MICRO} = 12.3mA + 8mA \approx 20.3mA$.

Luego tenemos que:
$I_{OUT\ MAX} = I_{INMAX\ TRANCEIVER} + I_{INMAX\ MICRO} = 57mA + 20.3mA < 78mA$.

El regulador utilizado (LP2985AIM5) puede entregar hasta 150mA por lo que cumple ampliamente con los requerimientos.

Para la alimentación de los conversores MAX232 y MAX485 se uso el regulador AN80L50 de 5V y 150mA.El consumo del MAX232 es menor que el del MAX485 (según hojas de datos). Los conversores no actúan simultáneamente por lo que a corriente máxima se calculo para el caso en que se utiliza el MAX485, cargado con un bus 485 en el cual se encuentran 32 conversores en la línea. Cada conversor supone una carga de 12Kohm por lo que al tener 32 en paralelo, la impedancia vista es de 375Ohm. Luego el máximo voltaje diferencial es de 12V, esto significa que el conversor tendrá que entregar 32mA en este

caso. Como ya se menciono, el regulador puede entregar 150mA y esto es más que suficiente para las necesidades de los conversores. Se sobredimensionó el regulador para dejar abierta la posibilidad de alimentar el actuador que se conecte a la RFB.

La figura 6.18 muestra un esquema de conexión del bus 485 donde se puede observar como queda cargado el conversor MAX485.

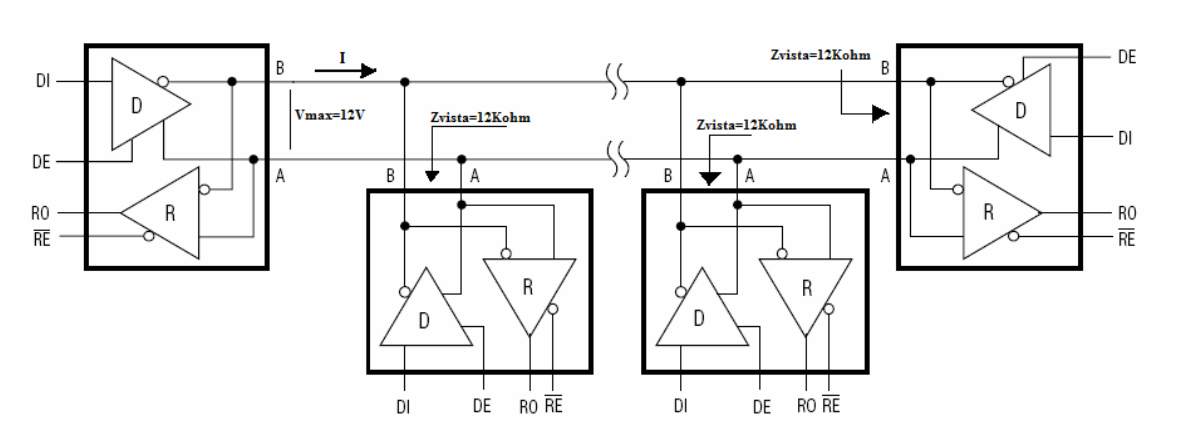


Figura 6.18: Bus 485.

El esquemático final del circuito que compone la alimentación es el mostrado en la figura 6.19.

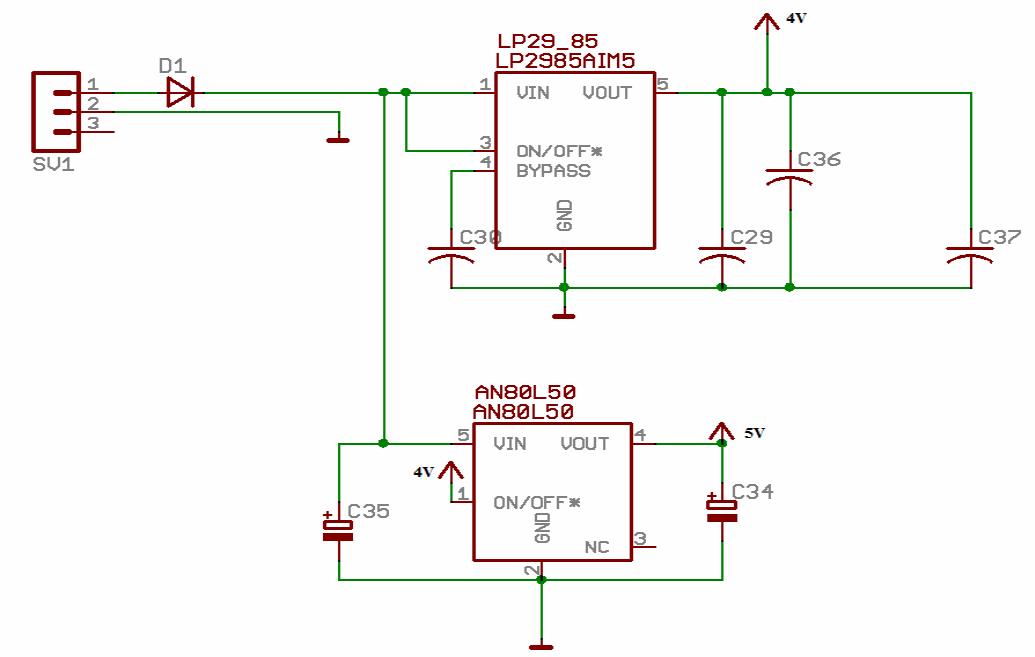


Figura 6.19: Esquema de conexiones reguladores de voltajes.

Notar que el regulador AN80L50 tiene un pin de habilitación (pin 1) en el cual se conecta la salida del regulador LP2985AIM5 para que el primero este siempre en su estado encendido. Luego debajo tenemos el circuito de regulación auxiliar que propone el fabricante y que fue implementado.

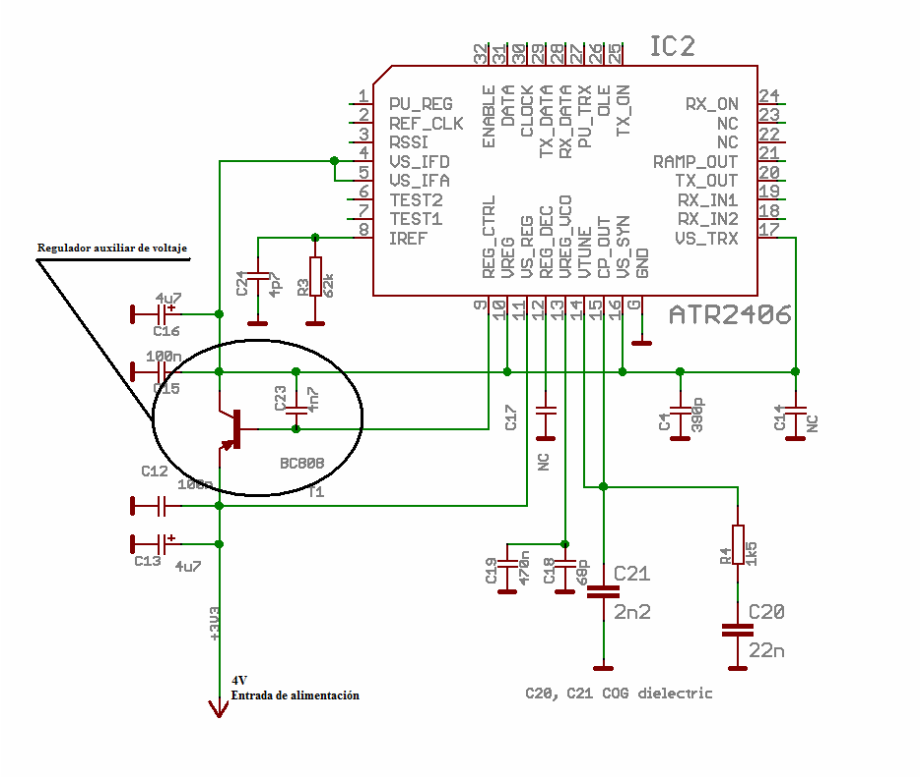


Figura 6.20: Regulador auxiliar ATR2406.

6.2.8 Construcción del impreso

6.2.8.1 Introducción

En una primera instancia se comenzó el diseño del impreso utilizando el software *TraxMaker*, ya que este es un software muy sencillo e intuitivo para diseño de circuitos impresos. Debido a la inexperiencia con este tipo de software se decidió empezar con este pero luego de comprender el funcionamiento del mismo se decidió cambiar y utilizar *Ealge.* Este cambio se dio ya que TraxMaker no posee ciertas herramientas que eran necesarias para el desarrollo de la RFB. Además las librerías con las que se contaba eran muy escasas. Pero la razón más importante era la imposibilidad de extraer los archivos *Gerber.* Estos archivos son necesarios para que el fabricante pueda construir el impreso.

Se utilizó EAGLE 4.16r2 que es una versión libre de este software y contiene todas las librerías de la versión profesional. Además existe una gran disponibilidad de librerías y documentación en Internet, haciendo de programa una herramienta muy potente. La extracción de los archivos *Gerber* es muy sencilla, mas adelante se detalla el procedimiento para poder obtener estos archivos. Como fue mencionado anteriormente, Atmel provee de un diseño de antena y adaptación de impedancia para el transceiver ATR 2406. Otro punto a favor de EAGLE es que los archivos del diseño de antena de Atmel son compatibles con el software (ver Anexo B).

Para el diseño de la RFB se partió del diseño que provee Atmel. Luego este se fue modificando y se le fueron agregando los componentes que conforman nuestro diseño. En la figura 6.21 se aprecia el diseño del cual se partió.

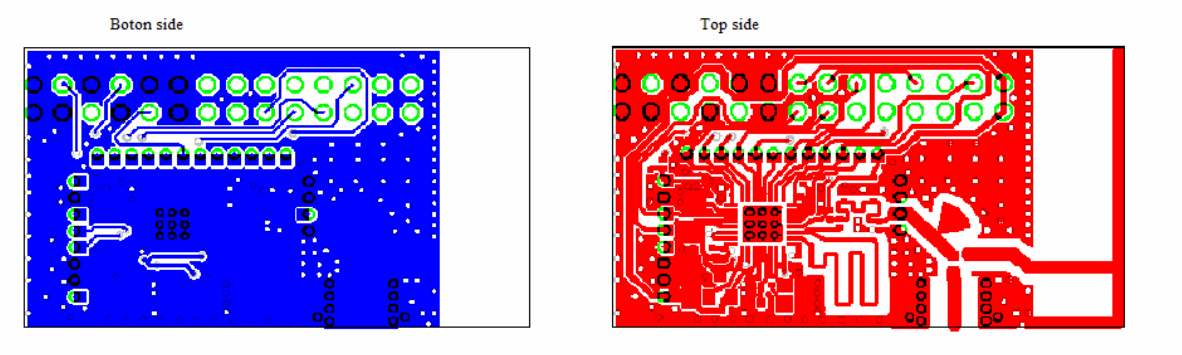


Figura 6.21: Modelo de Atmel, capas superior e inferior.

Atmel también provee el archivo del esquemático, ver figura 6.22.

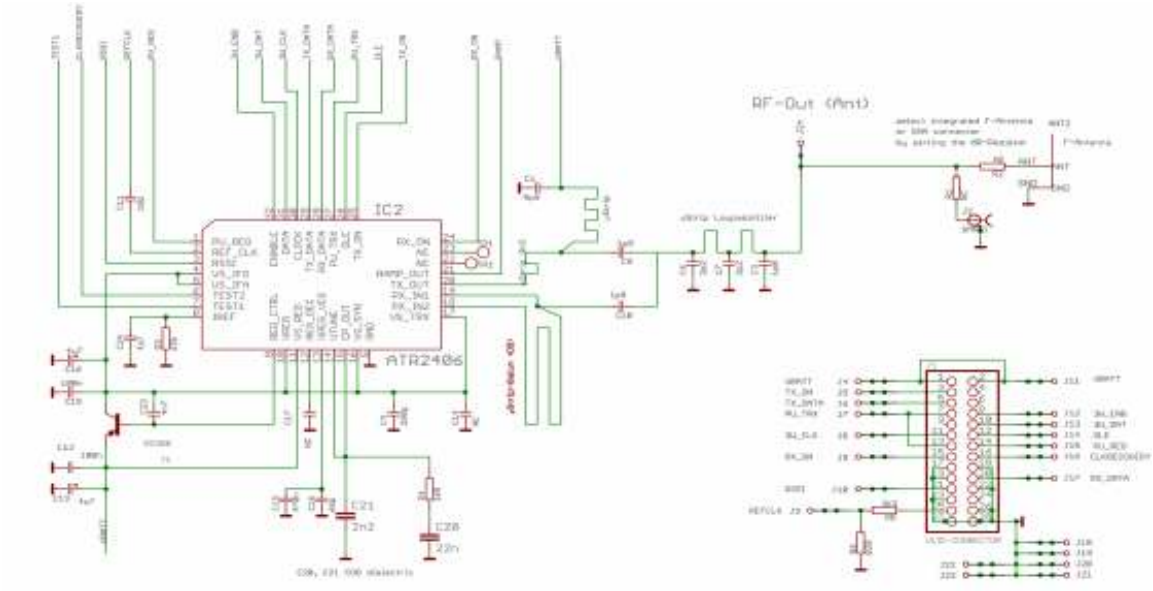


Figura 6.22: Esquemático del modelo de Atmel.

6.2.8.2 Pasos para la implementación

El primer paso hacia el desarrollo de la placa fue plantearse el circuito como un esquema eléctrico para después pasar a seleccionar los componentes. Para la selección de los componentes se tuvo en cuenta primero que nada la disponibilidad en el mercado y como segunda prioridad la existencia de los componentes en las librerías de EAGLE. Para los casos en que estos no estaban disponibles en las librerías se tuvo que hacer un diseño específico.

Luego de tener la lista de componentes a utilizar se procedió a realizar el esquemático en EAGLE. Una vez terminado el esquemático se procedió a implementar la placa propiamente dicha. Para esto se presentaron los componentes en sus ubicaciones tentativas y a continuación se comenzó el ruteo de pistas. A medida que se realizaba este ruteo se iban modificando las ubicaciones de los componentes. Iterando entre el ruteo y la reubicación se llego al diseño definitivo.

Los tamaños de pistas y los diámetros de los agujeros se diseñaron utilizando el criterio de hacerlos lo más grande posible siempre que este hecho no comprometiera el tamaño de la placa.

Otro punto importante en el diseño son las especificaciones de los materiales en los que se construyó la RFB. En la documentación del diseño que provee Atmel, se recomienda que la placa sea hecha en FR4 con baño de oro y un grosor de 500 um.

En la figura 6.23 se presenta el diseño Eagle de la placa y en la figura 6.24 se muestra el esquemático.

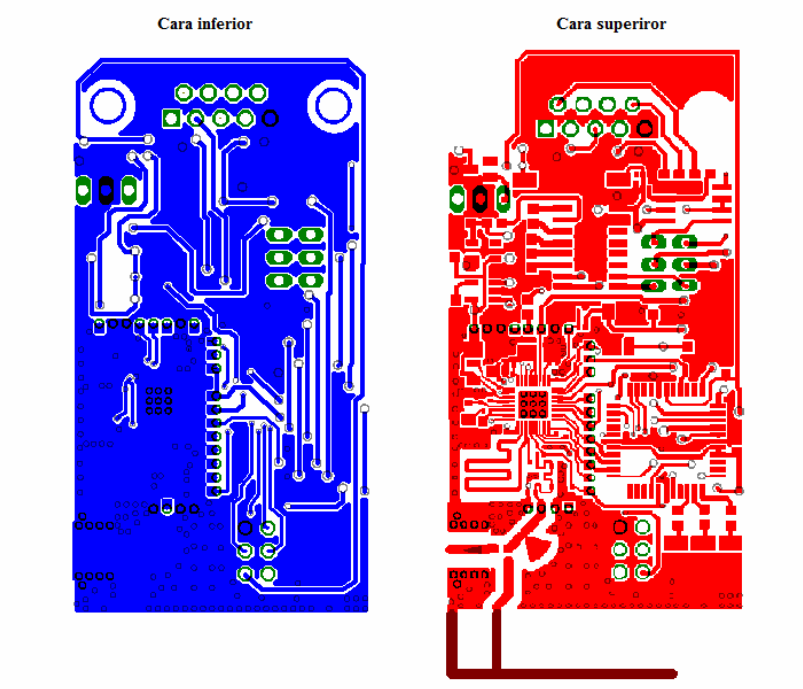


Figura 6.23: Diseño de capas de la RFB.

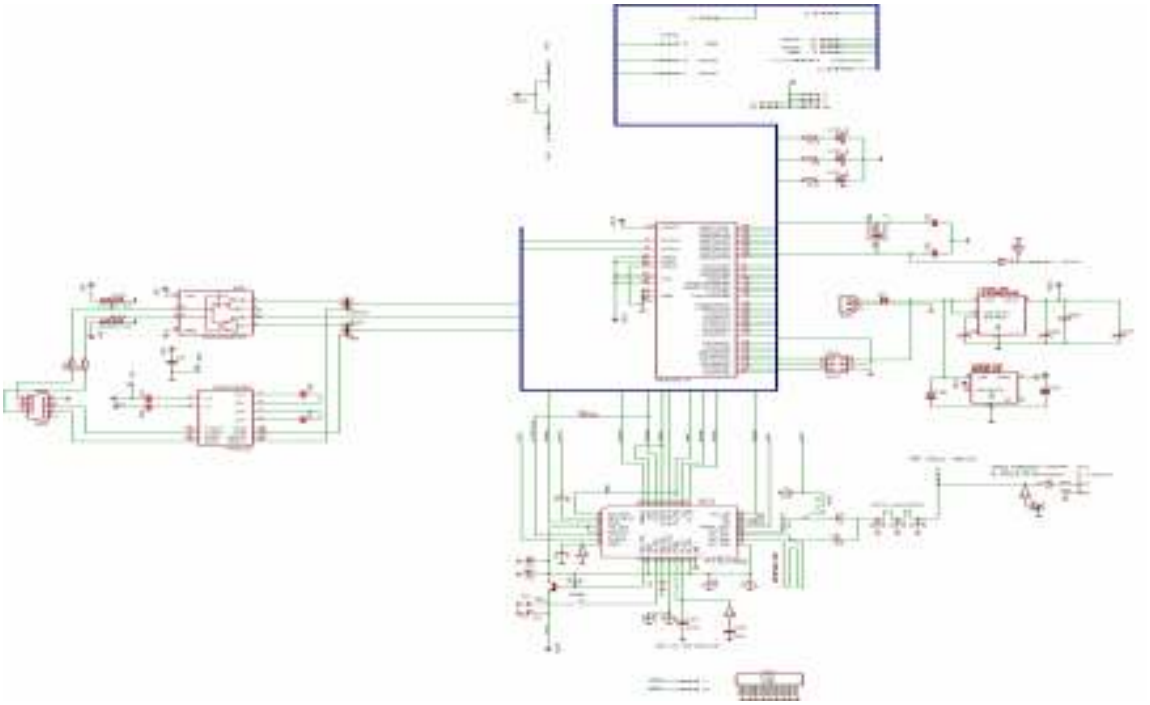


Figura 6.24: Esquemático de la RFB.

6.3 Dispositivo Base

6.3.1 Composición de la Base

Como fue mencionado anteriormente, este dispositivo esta compuesto por dos módulos independientes interconectados, una RFB y un RCM3700. La RFB se describió en detalle en la sección anterior (ver 6.2). El RCM3700 es un sistema embebido que se basa en el microprocesador Rabbit 3000 y cuya característica más importante para esta aplicación es su puerto Ethernet (ver sección 6.3.2).

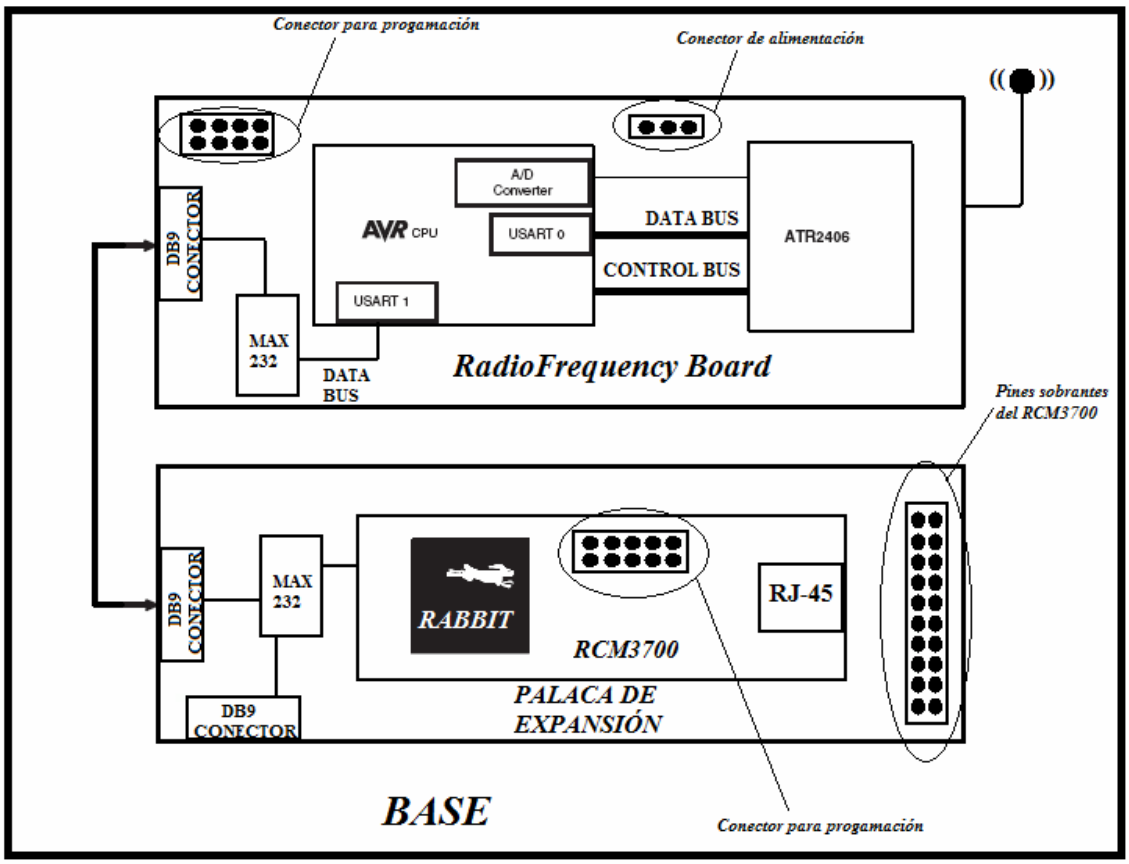


Figura 6.25: Diagrama de bloques del dispositivo base.

En la figura de arriba se muestra un diagrama de bloques del dispositivo base. Se puede observar que el RCM3700 esta montado en una placa de expansión desarrollada especialmente para poder conectarlo con una RFB. La placa de expansión tiene un conversor MAX-232 al cual se cablearon a los puertos serie del Rabbit y dos conectores DB9 para las salidas del conversor, también se cablearon los pines restantes del RCM3700 a un conector para poder tener acceso a ellos. La alimentación del RCM3700 es tomada por el conector DB9 que se usa para conectarlo a la RFB.

6.3.2 Introducción RCM 3700

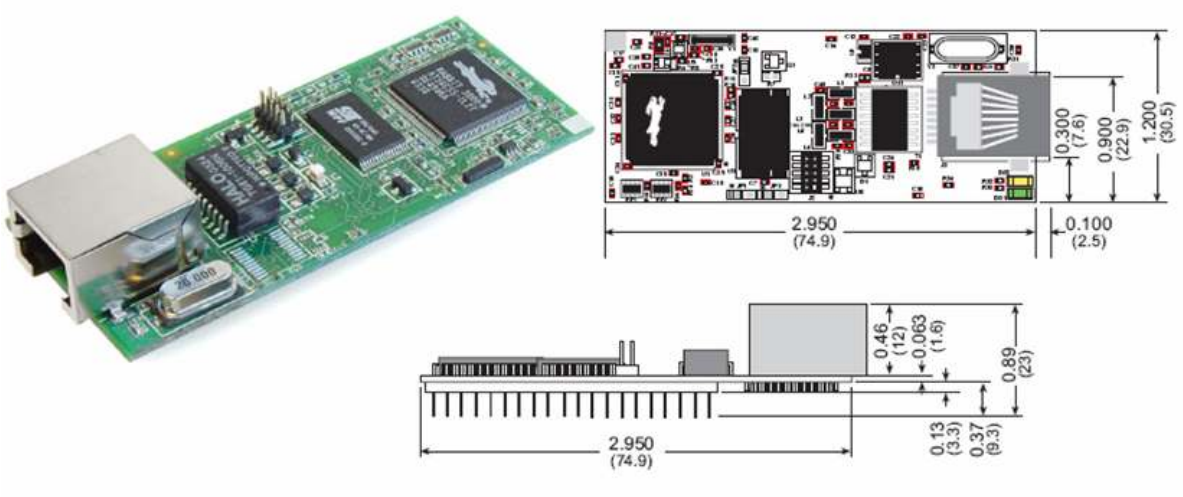


Figura 6.26: RCM3700.

El *RCM3700* fue seleccionado para proveer a la base, de comunicación a través de Ethernet. Es un sistema muy potente basado en un microprocesador *Rabbit 3000* que integra un chip Ethernet *RTL8019A* y otros periféricos como memoria flash, puertos serie y paralelo. También posee el conector RJ-45 para el puerto Ethernet.

En principio este sistema es más de lo que se necesita para desarrollar la aplicación de la base, posee 512 Kbytes de memoria Flash para programa, 512Kbytes en memoria SRAM, 11 timers de 8-bit, PWM, Reloj de tiempo real, 1Mbyte de memoria Flash serial utilizada para correr el sistema de archivos FAT de Dynamic C[1], 4 puertos serie, 2 USART, puerto Ethernet y un reloj de 22Mhz entre otras características (ver figura 6.27). [6.11] [6.12]

A pesar de ser un sistema sobredimensionado se opto por utilizarlo debido a que su sistema de desarrollo (Dynamic C) presenta grandes ventajas al momento de programar, este posee una gran cantidad de bibliotecas con macro funciones muy bien documentadas y sencillas de utilizar sobre todo en lo que respecta a la utilización del puerto ethernet. Dynamic C provee además herramientas de depuración en línea e incorpora un sistema de consola que permite observar y modificar variables del programa así como imprimir comentarios en la misma (printf[2]). Además de estas ventajas al momento de programar, también se tuvo en cuenta que ya se contaba con experiencia previa en este tipo de sistemas, que el costo del mismo es muy razonable y se cuenta con disponibilidad en el mercado.

[1] Software de desarrollo que se utiliza para programar sobre el RCM3700.
[2] Macro función de la librería stdio de Dynamic C que permite escribir en consola un comentario.

6.3.3 Estructura y especificaciones del RCM 3700

En la siguiente figura se presenta un esquema del RCM 3700 donde se pueden apreciar los distintos periféricos que se integran al Rabbit 3000.

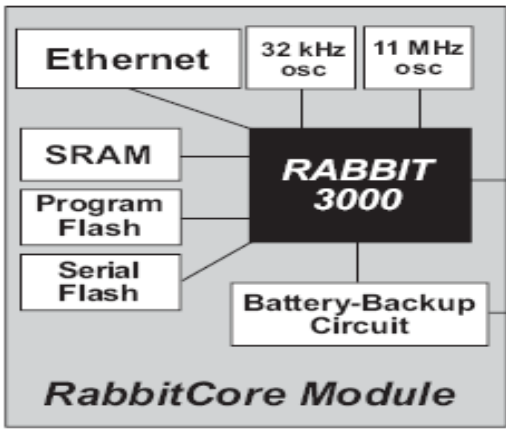


Figura 6.27: Diagrama de bloques RCM3700.

El RCM3700 cuenta con pines de salida cableados a un conector J1 de 40 patas que se encuentra debajo del módulo. Los puertos del Rabbit 300 pueden ser configurados para salir por estos pines.

En la figura 6.28 se muestra el esquema de los puertos que contiene el Rabbit 3000 y como se conectan al J1.

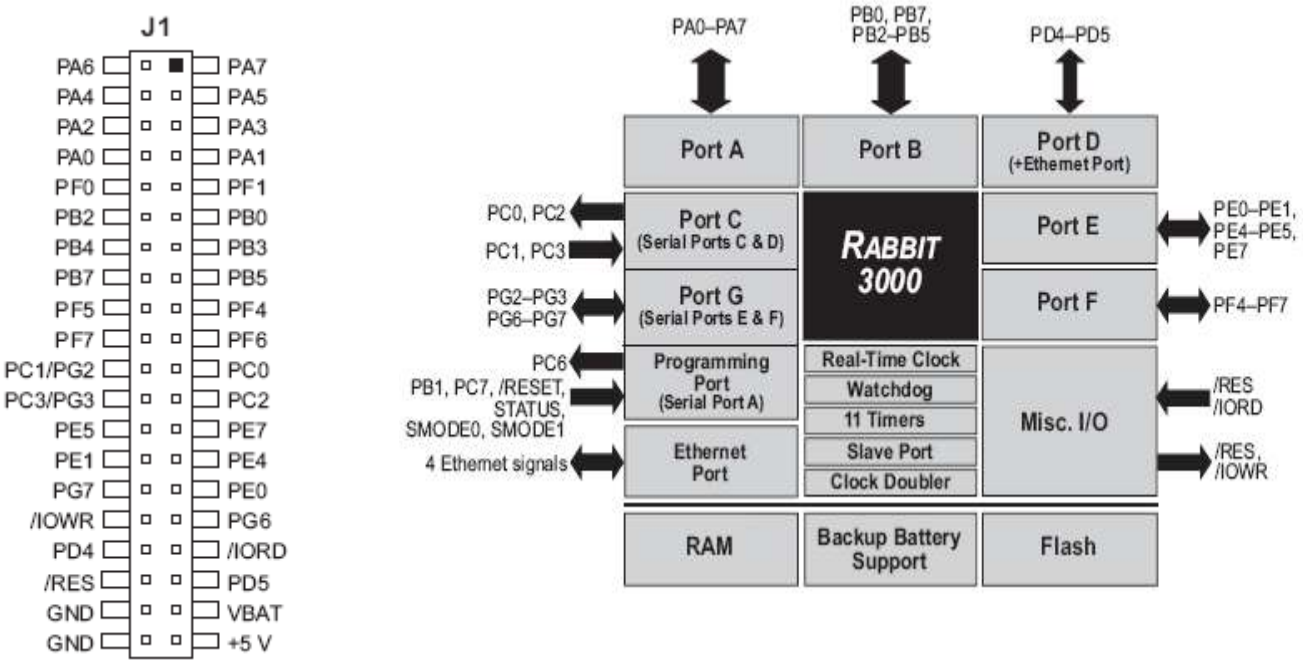


Figura 6.28: Pines de salida y puertos del RCM3700.

Parameter	RCM3700	RCM3710	RCM3720
Microprocessor	Low-EMI Rabbit 3000® at 22.1 MHz		
Ethernet Port	10Base-T, RJ-45, 2 LEDs		
Flash Memory	512K	256K	512K
SRAM	512K	128K	256K
Serial Flash Memory	1Mbyte		
Backup Battery	Connection for user-supplied backup battery (to support RTC and SRAM)		
General-Purpose I/O	33 parallel digital I/0 lines: • 31 configurable I/O • 2 fixed outputs		
Additional I/O	Reset		
Auxiliary I/O Bus	Can be configured for 8 data lines and 5 address lines (shared with parallel I/O lines), plus I/O read/write		
Serial Ports	Four 3.3 V CMOS-compatible ports configurable as: • 4 asynchronous serial ports (with IrDA) or • 3 clocked serial ports (SPI) plus 1 HDLC (with IrDA) or • 1 clocked serial port (SPI) plus 2 HDLC serial ports (with IrDA)		
Serial Rate	Maximum asynchronous baud rate = CLK/8		
Slave Interface	A slave port allows the RCM3700 to be used as an intelligent peripheral device slaved to a master processor, which may either be another Rabbit 3000 or any other type of processor		
Real-Time Clock	Yes		
Timers	Ten 8-bit timers (6 cascadable, 3 reserved for internal peripherals), one 10-bit timer with 2 match registers		
Watchdog/Supervisor	Yes		
Pulse-Width Modulators	4 PWM output channels with 10-bit free-running counter and priority interrupts		
Input Capture/ Quadrature Decoder	2-channel input capture can be used to time input signals from various port pins • 1 quadrature decoder unit accepts inputs from external incremental encoder modules or • 1 quadrature decoder unit shared with 2 PWM channels		
Power	4.75–5.25 V DC 100 mA @ 22.1 MHz, 5 V; 78 mA @ 11.05 MHz, 5 V		
Operating Temperature	–40°C to +70°C		
Humidity	5% to 95%, noncondensing		
Connectors	One 2 x 20, 0.1" pitch		
Board Size	1.20" × 2.95" × 0.89" (30 mm × 75 mm × 23 mm)		

Tabla 6.3: Especificaciones mecánicas y eléctricas.

6.3.4 Placa de expansión para RCM 3700

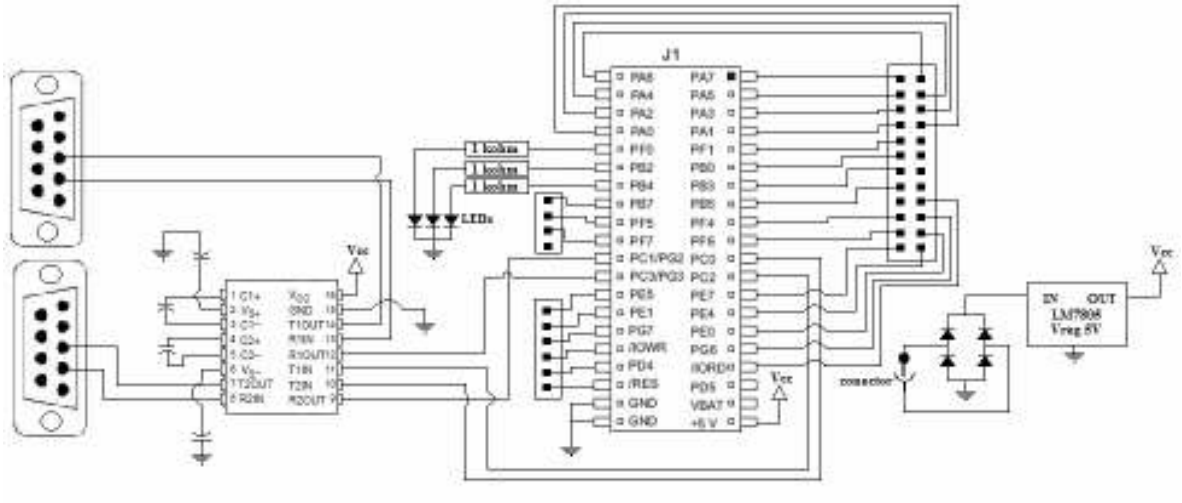


Figura 6.29: Placa de expansión base.

En la figura 6.29 se observa el esquema eléctrico de la placa de expansión. Esta fue diseñada con el software *TraxMaker* y construida en forma artesanal (ver Anexo D). Esta permite el acceso a los puertos del RCM3700 e incorpora un regulador de voltaje y un conversor de señales MAX232. En esta placa se monta el RCM3700 y en uno de sus conectores DB9 se conecta la RFB.

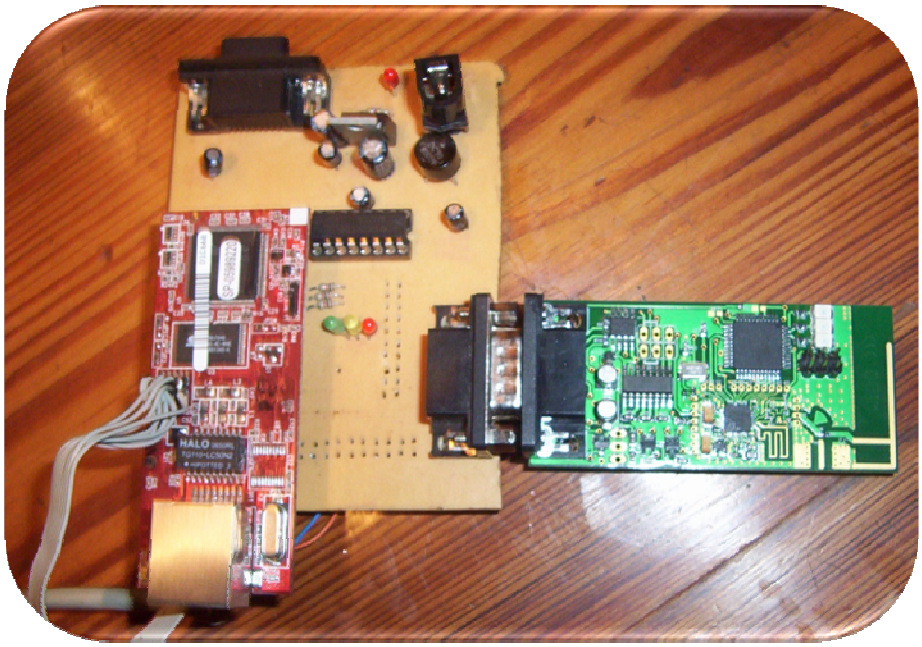


Figura 6.30: Dispositivo Base.

6.4 Fuente de alimentación

En esta sección se presenta el desarrollo de la fuente de alimentación para las RFB, tanto para el dispositivo Base como para el dispositivo Remoto.

6.4.1 Introducción

Para alimentar a los dispositivos, lo primero que se planteó fue que tipo de fuente es más apropiada. Un aspecto importante fue decidir si se realizaría a partir de baterías o utilizando una fuente AC/DC, energizada a partir de la red eléctrica. Como en esta aplicación las placas van a manejar actuadores domésticos, es adecuado que la alimentación de la placa se tome de la red eléctrica, ya que el uso de baterías implica, entre otras cosas, que periódicamente se realice un recambio de las mismas. Además la tensión de red es de fácil acceso ya que los dispositivos remotos van a manejar actuadores, que necesitan 220VAC. Otro punto a favor es que el consumo de energía no es un aspecto restrictivo.

Por esto se optó por implementar una fuente de tensión continua, energizada de la red de 220 VAC. Dicha fuente debe tener una salida que se ajuste a los requerimientos de los reguladores de voltaje que posee la RFB.

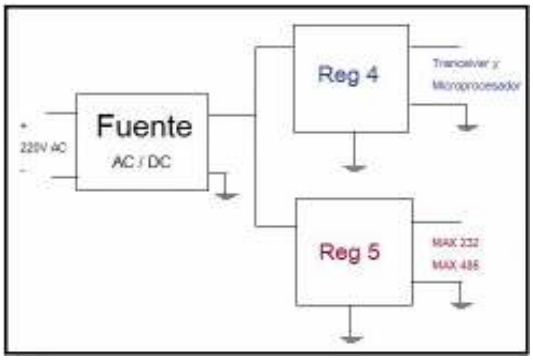


Figura 6.31: Fuente y reguladores.

Como se observa en la figura 6.31 la fuente a construir tiene que suministrar la corriente a dos reguladores en simultáneo. El regulador de 4 V alimenta a los dos integrados principales de nuestra placa: el microcontrolador y el transceiver. La carga máxima que va

a tener que soportar en es de 150 mA. También la fuente debe entregar corriente al regulador de 5 V que alimenta a los dos conversores (MAX 232 y MAX 485), en el peor caso se tendrá un consumo de 15mA. Por lo tanto se necesita que la fuente pueda entregar al menos 165mA estimado.

Otra característica relevante de la fuente es que su tamaño debe ser lo más reducido posible ya que las placas probablemente deban ser instaladas en lugares de acotado espacio.

Para el diseño se decidió prescindir de un transformador, ya que este compromete seriamente las restricciones de tamaño. En contrapartida se presentan desventajas por la ausencia del transformador. Se pierde protección del circuito a sobrecorrientes (aislación galvánica) y seguridad de las personas por choques eléctricos. Pero esta última desventaja no resultó tan relevante, porque en esta aplicación las personas no pueden tener acceso directo a estás placas, ya sea porque estén embutidas en la pared o se encuentran en un tablero. Para mejorar este aspecto se propone agregar una caja plástica que cubra esta placa de forma de lograr una mayor seguridad.

6.4.2 Especificación de la fuente

Una de las características relevantes de la fuente es su tensión de salida. A partir de las especificaciones de ambos reguladores, LP2985AIM5 de National Semiconductor y el AN80L50RMS de Panasonic, se determinó el voltaje de salida V_{out}.

La entrada a los reguladores debe cumplir: $6\ V < V_{out} < 14\ V$.

De manera de tener margen se centró la tensión de salida en 10V y con el menor ripple posible, que depende esencialmente del condensador a la salida de la fuente.

La corriente de salida de la fuente debe poder sustentar la carga de los reguladores más sus pérdidas.

En lo que sigue se presenta un resumen de los cálculos de las corrientes:

$$I_{OUT\ MAX} = I_{IN4\ MAX} + I_{IN5\ MAX}$$

$$I_{IN4\ MAX} = I_{OUT4\ MAX} + I_{GND4\ MAX}$$

$$I_{IN5\ MAX} = I_{OUT5\ MAX} + I_{GND5\ MAX}$$

$$I_{OUT4\ MAX} = I_{INMAX\ TRANCEIVER} + I_{INMAX\ MICRO} = 57\ mA + 90\ mA = 147\ mA.$$

De modo de obtener cierto margen se utilizó el criterio de que la máxima corriente que debe proporcionar el regulador sea 4V es:

$I_{OUT4\ MAX}$ = 150 mA.

$I_{OUT5\ MAX} = I_{INMAX\ MAX232} + I_{INMAX\ MAX485}$= 10 mA + 1 mA = 11 mA.

$I_{IN4\ MAX}$ = 150 mA + 0.85 mA ≃ 151 mA.

$I_{IN5\ MAX}$ = 11 mA + 5 mA = 16 mA

$I_{OUT\ MAX} = I_{IN4\ MAX} + I_{IN5\ MAX}$ = 151 mA + 16 mA = 167 mA

Entonces la carga máxima de la fuente resulta ser: $I_{OUT\ MAX}$ = 170 mA.

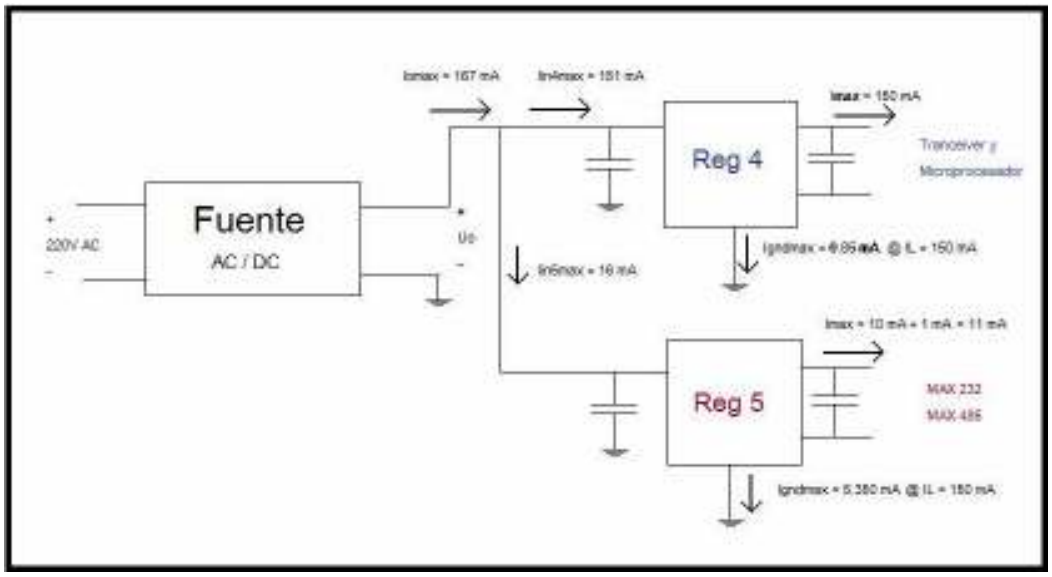


Figura 6.32: Esquema de alimentación y corrientes.

El circuito debe proporcionar una tensión de salida, y una corriente máxima de salida $I_{out\ max}$ = 170 mA y una corriente mínima $I_{out\ min}$ = 1 mA, que consumirá el microcontrolador.

6.4.3 Diseño

Para diseñar la fuente, además de las especificaciones antes mencionadas, se tuvieron en cuenta los siguientes aspectos:

- Eficiencia
- Rizado (*ripple*) de tensión de salida
- Estabilidad a cambios de carga
- Complejidad
- Espacio de placa requerido
- Disponibilidad de compra
- Tipo de montaje

Como primer etapa se investigó diferentes alternativas, discutiéndose distintas configuraciones y se propusieron cuatro circuitos como candidatos a fuentes, los cuales fueron analizados cualitativamente y mediante simulaciones.

En las figuras a continuación se muestran los cuatro circuitos que fueron propuestos para implementar la fuente.

Circuito 1

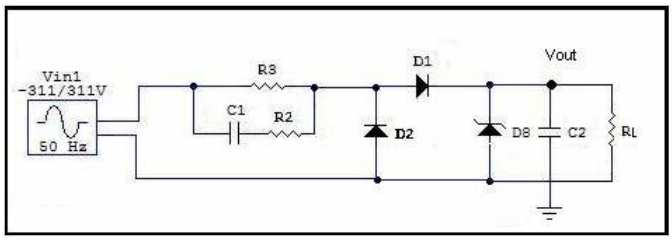


Figura 6.33: Circuito 1.

Circuito 2

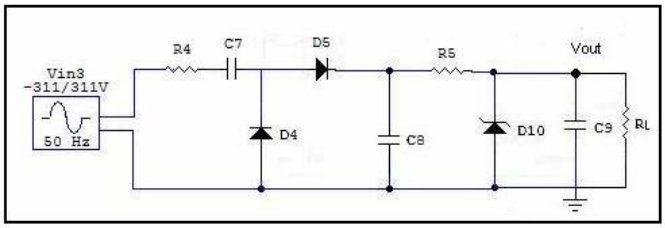


Figura 6.34: Circuito 2.

Circuito 3

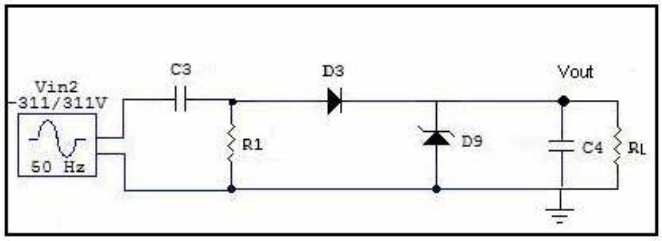


Figura 6.35: Circuito 3.

Circuito 4

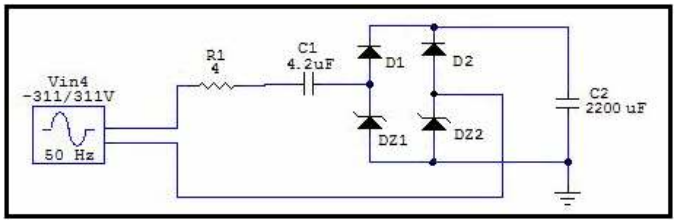


Figura 6.36: Circuito 4.

6.4.3.1 Análisis de simulaciones

Para realizar las simulaciones, los reguladores fueron modelados como resistencias, las cuales se fueron variando para obtener distintas corrientes a la salida.

Circuito 1

Este circuito utiliza un rectificador de media onda. En un semiciclo la tensión a la salida la mantiene la red, y en el otro semiciclo la aporta el condensador de salida.

Para un mismo condensador de salida siempre tiene mayor *ripple* respecto al circuito 4. Por ejemplo si C2 = 470 µF y R_L = 47 Ω, el *ripple* será de 3 V y el circuito 4 tiene 0.85 V en su

salida. No es muy importante para las entradas de los reguladores, aunque en las hojas de datos también se da recomendación de colocar condensadores a la entrada de los mismos. Además tiene mayores pérdidas resistivas.

Debido a esto esta opción fue descartada.

Circuito 2

En este circuito la topología no llega a entregar los 170 mA de corriente que se precisan, por lo tanto no sirve y fue descartado.

Circuito 3

Para este circuito ocurre lo mismo que para el circuito 2, se tiene mayor *ripple* de tensión (2 V) y se disipan decenas de watts, cuando a la salida se tienen 5,5 V y 160 mA.

Debido a esto fue descartado.

Circuito 4

En este circuito la fuente tiene un puente mixto de diodos comunes y diodos zener. Para cada semiciclo, la tensión la fija uno de los dos diodos zener, aliviando al condensador de salida.

La corriente que se toma de la red está principalmente determinada por la impedancia del condensador. Casi no varía respecto a la carga. Por lo tanto la potencia que se disipa en la resistencia para toda carga es la misma. P_{DIS} = 382 mW.

A continuación se presenta un análisis más detellado del circuito 4, el cual elegimos.

Sabiendo que cumple con las especificaciones en cuanto a la tension de entrada y a la tensión y corriente a la salida, se centró la atencion en las características más importantes de una fuente de alimentación, las cuales son: rendimiento, rizado de tensión a la salida, comportamiento a distintas cargas y complejidad del circuito que implementa la fuente.

Rendimiento

El rendimiento en este caso, despreciando la disipación que existe en los semiconductores, va a ser la relación entre lo que disipa la carga útil (simulada como R_L) y la resistencia R1.

Por lo que se vió anteriormente, la potencia disipada en la resistencia es independiente de la carga, entonces es claro que el rendimiento que va tener esta fuente va a ser mayor cuanto más potencia consuma la carga.

Rendimiento $\eta = P_{CARGA} / (P_{CARGA} + P_{DISIPADA})$.

Carga	Rendimiento η
Máxima carga	81.1 %
Mínima carga	23.5 %

Tabla 6.4: Rendimiento de la fuente.

Como la mayor parte del tiempo se tendrá un consumo que va estar más cerca de la carga máxima, ya que el protocolo inalámbrico ocasiona que el tranceiver se encuentre recibiendo en un porcentaje grande del tiempo de canal. Por esta razón el rendimiento de la fuente será cercano al 80 %.

Rizado de tensión

Si bien el rizado de tensión a la salida no es un requerimiento fuerte en este diseño, ya que la salida de esta fuente va a pasar por los reguladores, es una variable que interesa; cuanto menor es el *ripple* que le entre a los reguladores será más estable la alimentación de los integrados como por ejemplo el microcontrolador.

Como se observa en la figura 6.37 se tiene un rizado menor a 0.4 V, luego de 220 milisegundos de haber encendido la fuente.

Siendo un valor aceptable para la entrada de los reguladores.

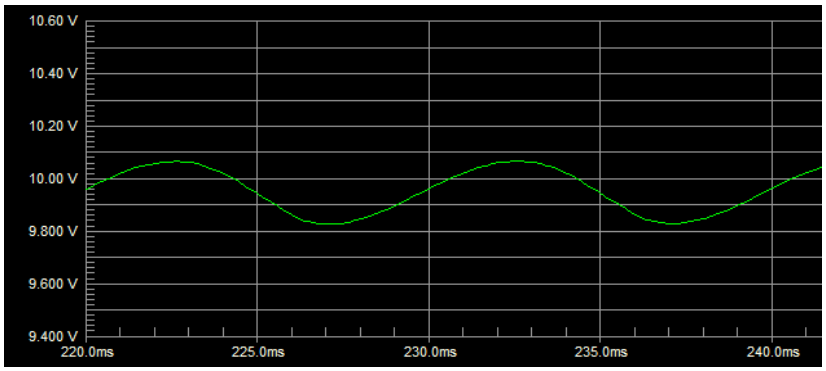


Figura 6.37: Rizado para plena carga.

Comportamiento de la tensión a diferentes cargas

Para estimar el comportamiento del sistema para diferentes cargas se realizaron dos simulaciones del sistema, una a plena carga y otra a mínima carga, modificando el valor de la resistencia R_L acorde a esto.

A plena carga la corriente es Iout = 175 mA (R_L = 54 ohm) y se obtiene <Vout> = 10.1 V.

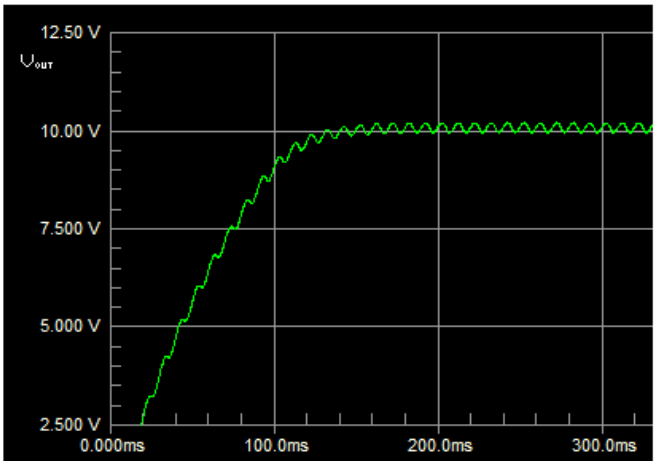


Figura 6.38: Vout(t) para plena carga.

Con mínima carga Io = 1 mA (R_L = 11 kohm) y se tiene <Vout> = 12 V.

Es decir que en los extremos de funcionamiento, la tensión de salida es acorde a las especificaciones requeridas.

Complejidad

La complejidad del circuito es baja y además cumple con otro de los aspectos que se tuvieron en cuenta que es la poca cantidad de componentes, facilitando a su vez la compra de los mismos, y otorgando también mayor facilidad en el montaje. Debido a esto se eligió el circuito 4.

6.4.4 Pruebas

Como primera prueba se implementó este circuito en un *protoboard*, al cual además se le agrego un fusible de 2 A para prevenirnos de sobrecorrientes.

Se obtuvieron resultados dentro de lo esperado respecto a la salida de la fuente. Sin perjuicio de ello se observó que al desenchufar la fuente, quedaba cargado el condensador C_1, al cual hubo que descargar manualmente. Este problema fue previsto más adelante en el diseño de la placa.

Los resultados obtenidos para los casos extremos de carga se muestran en la tabla 6.5.

R_L (Ω)	V_{OUT} (V)	I_L (mA)
56	9.5	170
10 k	9.7	1

Tabla 6.5: Resultados para los límites de carga.

Como se ve el voltaje de salida se mantiene sin variaciones aún para los dos extremos de carga (aún mejor que en las simulaciones), lo cual es importante aunque sabemos que los reguladores lineales son los que se van encargar de esa tarea.

Con esta prueba determinamos los componentes principales para la construcción del impreso definitivo.

6.4.5 Construcción del impreso

La placa fue diseñada en una sola capa y su tamaño es de 2,8 cm X 7 cm. En la figura 6.39 a continuación se puede observar el board correspondiente.

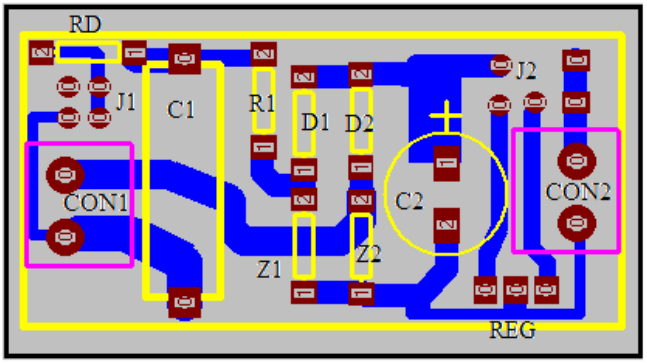


Figura 6.39: Board de la fuente.

La elección y el dimensionado de los componentes se realizaron a partir de la disponibilidad existente en el mercado local, teniendo que adaptarse a esta condición.

La lista de componentes definitiva se encuentra contenida en la tabla 6.6.

Como se observo anteriormente se agregó una resistencia R_C para posibilitar la descarga del condensador C_1, que se utiliza manualmente cambiando la posición del jumper J1.

Asimismo se añadió otro jumper J2, que se usa para tener la salida con o sin regulador. Para la aplicación común de alimentar la placa principal, J2 se selecciona entonces para el caso de no tener dicho regulador, el cual va a estar en la placa principal.

Componente	Cantidad	Modelo	Valor
C1	1	300 V	4.2 µF
C2	1	16 V	2200 uF
R1	1	0.5 W	6 Ω
RC	1	0.5 W	1 kΩ
Diodos (D1, D2)	2	1N4148	-
Diodos tener (Z1, Z2)	2	1N5240 (10 V)	-
Jumper (J1, J2)	2	-	-
Regulador	1	LM7805	-

Tabla 6.6: Lista de componentes.

Capítulo 7

Descripción del software

7.1 RFB

El diseño e implementación del software que lleva acabo la comunicación RF es una relación dinámica entre la especificación del protocolo, la escritura en un lenguaje compilable y el conjunto de pruebas que verifica y valida el *firmware*.

Basándose en especificación del capítulo 5 se escribe una primera versión, en C embebido con Assembler, del *firmware* que controlará la RFB descrito en el capítulo 6. Luego un conjunto de pruebas deberá verificar que el programa cumple con los servicios especificados de forma correcta y validar que está ausente de comportamientos no deseados.

Es oportuno mencionar que gran parte de la programación y pruebas fueron realizadas sobre el Kit de desarrollo "*Dragon Fly"*, que provee Atmel, por razones de disponibilidad. Los algoritmos desarrollados sobre este kit deberán ser probados y verificados en la RFB y en un entorno real.

Las siguientes secciones incluyen un breve repaso del hardware utilizado, el entorno de desarrollo que se usó para programar los chips y en que consiste la organización del *firmware*. Luego intentaremos describir como fueron programados y probados los servicios que los dispositivos de cumplir de acuerdo a lo especificado en el capítulo 5.

7.1.1. Hardware

Una descripción detallada del hardware que compone la RFB es la brindada en el punto 6.2, pero a modo de esclarecer como afecta el diseño del software resumiremos los principales componentes:

Transreceptor ATR 2406

Utiliza modulación GFSK con VCO y PLL integrados, permite una velocidad máxima de 1,152Mb/s. Existen ciertos tiempos que el software debe respetar para el transreceptor funcione correctamente, en resumen son:

- Esperar un mínimo de 40 μs luego de encender el transreceptor para que los voltajes que alimentan los módulos internos se estabilicen.
- Al configurar parámetros del PLL (*Phased Locked Loop*): canal de recepción/transmisión y la desviación del filtro gaussiano, se debe esperar al menos 200 μs para que la frecuencia de salida del VCO se estabilice.
- Para transmitir se deben mandar al menos 40 μs de preámbulo para que el receptor pueda sincronizarse.

Para obtener más información sobre las características del transreceptor referirse su hoja de datos [7.6].

Microprocesador ATMega 164P

Este microprocesador cuenta con dos USART, una que se utilizará para comunicarse con el transreceptor y otra para comunicarse con la interfaz serie. También posee tres counter/timers: dos de 8 bits y uno de 16 bits, un conversor A/D de 8 canales y un comparador analógico. Acepta una tasa de transferencia de 1,152Mb/s a través de la USART sólo en modo sincrónico [7.7].

Interfaz serie de salida

La interfaz serie es la entrada/salida de datos de la RFB y puede utilizar los protocolos RS-232 y RS-485 configurable por medio de un *jumper*.

Kit de desarrollo ATR 2406

Este kit consta de dos placas *Dragon Fly*. Las placas fueron diseñadas para depurar aplicaciones basadas en el ATR 2406 y el ATMega 88. Posee una pantalla LCD de 4 x 20 caracteres, 3 botones y un LED. Se programa por la interfaz y protocolo SPI. Además Atmel provee un código que implementa la comunicación RF entre ambas placas utilizando AFHSS [7.1].

7.1.2 Entorno de desarrollo

El entorno de desarrollo utilizado para escribir y probar el *firmware* es el sugerido por los creadores del *firmware* que viene con kit de desarrollo "Smart RF". Esto es muy conveniente ya que muchas de las funciones que vienen con este *firmware* son útiles para la implementación de nuestro protocolo.

7.1.2.1 Compilador

El compilador utilizado es el *CodeVision AVR*, diseñado por terceros específicamente para trabajar con la línea de microprocesadores de Atmel. A continuación se detallan las principales características:

- Compilador compatible con ANSI C de fácil uso.
- Editor auto-indentable, que sobresalta la sintaxis del código en C y AVR Assembler.
- Librería de *fase floating point* con multiplicador por hardware y núcleo de instrucciones que soporta los chips ATiny y ATMega.
- Extensiones específicas de AVR para:
  - Acceso al área de memorias EEPROM y FLASH
  - Acceso de bit a los registros I/O
  - Colocar variables de bit en los registros I/O de propósito general (GPIOR)
- Gran variedad de algoritmos de optimización y opción de optimizar por tamaño o por velocidad, configurable por el usuario.

- Posibilidad de insertar código Assembler en archivos en C.
- Uso muy eficiente de RAM.
- Depurar de código a nivel de C, generando el archivo de símbolos COFF, que permite simular en el *AVRStudio*, vigilar variables (incluyendo estructuras y uniones) y el usar de la terminal I/O de AVR.
- Compatible con "In-Circuit Emulators" de Atmel: AVR JTAG Ice, AVR Dragon, etc.
- Generador automático de código que permite configurar rápidamente todos los módulos del micro.
- Terminal para depurar interfaces RS-232, RS-422 y RS-485
- Programador "*In-System*" para los chips de Atmel compatible con: STK500, AVR JTAG Ice, etc.

7.1.2.2 Programador

En primera instancia se programó mediante protocolo SPI utilizando la placa de desarrollo de Atmel ***STK 500***. La programación de los micros del kit de desarrollo (ATmega 88) y el micro 164p en encapsulado DIP fueron hechas de esta forma. Luego, dado que el ATmega 164p lo permite, se programaron los RFB mediante el algoritmo *JTAG*, para lo que se utilizó el emulador ***AVR Dragon***. El método JTAG posee varias ventajas sobre el SPI, por esta razón se diseñó el hardware RFB para esta opción, a continuación un resumen de las mismas:

- Es de 3 a 4 veces más rápido.
- Permite realizar *On-Chip Debugging* a través de mismo puerto.
- Permite insertar *breakpoints* por software y hasta 4 por hardware.
- Permite realizar un *test In-Circuit* del microcontrolador y el circuito externo.

Por información más detallada los métodos JTAG y SPI ver referencias [4.7] a [4.9] y la hoja de datos del micro.

7.1.3 Organización del *Firmware*

Para la implementación del *firmware* creamos un proyecto ***Tansreceptor.prj*** en el *CodeVision* que consta de los módulos indicados en la tabla 7.1.

Nombre del archivo .c y/o .h	Descripción del contenido
Defines	Nombres auxiliares de los puertos y declaración de constantes del programa
Variables	Declara todas las variables globales del programa
Inicialización	Inicializa las variables globales y configura inicialmente los puertos y los periféricos
Control_rf	Incluye todas las rutinas de configuración y atención al transreceptor y otras funciones de capa física.
Capa_de_red	Implementa los algoritmos y servicios de capa de red.
Serie	Rutinas de atención a la interfaz serie y servicios para configuración y asociación de dispositivos.
Aplicación_lcd*	Rutinas de configuración y escritura a la pantalla LCD
Interface*	Rutinas que permiten interactuar con los micros mediante el uso de botones y a través del LCD
Misc	Funciones útiles como *delays* y generador de números aleatorios

Tabla 7.1: Módulos utilizados en proyecto Transreceptor.

* Usados solamente en el programa del kit de desarrollo para depurar y verificar el código.

Las funciones escritas en los módulos de la tabla anterior serán utilizadas por el programa principal para coordinar una correcta transmisión de los datos y garantizar una buena

disponibilidad de los servicios. Las secciones siguientes muestran como estas funciones y servicios se implementan, dividiéndoles según la capa del protocolo a la cual pertenecen.

7.2.1 Capa Física

La capa física resuelve la comunicación entre el microprocesador y los periféricos. Esta capa consta en la inicialización y lectura de los puertos, rutinas de atención a interrupciones y configuración de los distintos módulos: UARTs, ADC y Timers.

7.2.1.1 Comunicación micro-transreceptor

La comunicación entre el microprocesador y el transreceptor utiliza la UART0 del micro en modo sincrónico ya que es el único modo en el que pueden comunicarse a máxima velocidad [7.8].

Dependiendo si se desea transmitir o recibir una trama, debe prenderse el ATR y configurarse en el modo y canal correspondiente, igualmente la UART debe ser configurada para cada caso. Se decidió hacerlo de esta manera para reducir el consumo ya que el transreceptor encendido contribuye en gran parte en el consumo máximo de las RFB. Las rutinas que realizan estas acciones se encuentran en **control_rf.c** y son las siguientes:

```
void power_up_ atr2406();
void power_dwn_ atr2406();
void init_ atr2406(uc_8 canal,uc_8 mode);
void set_usart_transmit();
void set_usart_reciver();
void reset_usart();
```

La hoja de datos del transreceptor específica los tiempos que demoran las distintas secuencias de conmutación entre modos del transreceptor los cuales son respetados por el software diseñado. Los tiempos de espera implementados superan holgadamente a los tiempos mínimos, ya que se utilizó los mismos que el *firmware* “Smart RF”, pero debido a que afectan directamente la tasa de datos efectiva que puede obtenerse mediante el uso nuestro protocolo, estos tiempos deberían ser minimizados para obtener un mejor rendimiento del sistema.

Secuencia	Tiempo (μs)*
Transmitir a recibir	200
Recibir a transmitir	200
Cambiar de canal	200
Apagado a transmitir	250
Apagado a recibir	200
Programar registro	30
Estabilización de PLL	200

Tabla 7.2: Tiempos requeridos por el ATR246 para efectuar dichas tareas.

* Como condición en todos los casos el reloj de transreceptor (ref_clock) debe estar estable.

El intercambio de datos lo realizan las rutinas de atención a interrupciones (ISR) de la UART0, las que se consideran con máxima prioridad. Las UARTs utilizadas poseen doble buffer de transmisión/recepción, lo que simplifica el manejo de los datos. La única restricción que debe cumplirse es que la rutinas atención duren menos que el tiempo en que lleva transmitir/recibir un nuevo byte que es 8,68 μs (10 bits, 1,152 Mbit/s). Por otro lado el deshabilitar las interrupciones globales es conveniente para lograr que las ISR sean lo más rápidas posibles y evitar posibles problemas de datos compartidos. Se escribieron dichas rutinas en Assembler utilizando las directivas que indican al compilador que optimie el código para lograr una mayor velocidad.

```
#pragma save/reg- //Disable register saving
#pragma optsize-  //optimize for speed

   interrupt[USART_DRE] void UsartTransmitt(void)
   interrupt[USART_TXC] void UsartTransmittEnd(void)
```

```
   interrupt[USART_RXC] void UsartReceive(void)
#pragma optsize+  //optimize for speed
#pragma savereg+  //Enable register saving
```

En esta primera etapa utilizamos tramas de largo fijo (64 bytes) a nivel de capa física. La comunicación a este nivel fue validada utilizando el kit de desarrollo, que se utilizó para visualizar el contenido de los buffers de transmisión y recepción, y de esta manera comprobar que eran efectivamente iguales durante sostenidas transmisiones. Las rutinas que manejan los datos RF pueden ser fácilmente modificadas para utilizarlas con tramas de largo variable.

7.2.1.2 FHSS y sincronismo

Los requerimientos a este nivel consisten en proveer un mecanismo de FH que sea robusto y acepte configuraciones desde capas superiores de los parámetros. Además debe funcionar correctamente independientemente de las posibles variaciones en el flujo de programa principal.

El recurso necesario a nivel de capa física para poder llevar a cabo la implementación del FH descrito en el capítulo 4 es simplemente un Timer. El programa desarrollado utiliza el timer0 de 8 bits y lo configura para dividir la frecuencia de reloj (13,824 MHz) por 1024, lo que nos permite un tiempo máximo de canal de 19ms (ver `intit_timer0()` en `inicializaciones.c`). Si bien esta muy por debajo de los límites establecidos por norma (400ms, ETSI), es lento comparado con Bluetooth (0.6ms). Además cuanto más rápida sea la conmutación de canales mejor será la dispersión de la potencia a través del espectro.

El tiempo de canal, `tiempo_ch`, es configurable y se actualiza escribiendo en el comparador A (`OCR0A`) del timer0. Al correr el tiempo establecido, el timer interrumpe y se ejecuta el cambio de canal*.

El comparador B maneja la conmutación de la ventana hablita la transmisión y recepción de paquetes RF; el cual se configura para levantar la ventana en `tiempo_ch_lo` y bajarla en un `tiempo_ch_hi`, con `tiempo_ch_lo < tiempo_ch_hi < tiempo_ch`. El ancho óptimo de la ventana dependerá de los requerimientos de la aplicación en particular; una ventana menor reducirá el consumo de la RFB mientras que una ventana mayor permitiría una mayor tasa efectiva de datos.

Si fuese deseable aumentar el tiempo máximo por el cual debería conmutarse el canal debe hacerse en múltiplos de `tiempo_ch` o bien cambiar por el timer de 16 bits, que en principio

está reservado para ejecutar los distintos tiempos de espera que el *firmware* necesite. Respecto al tiempo mínimo de salto de canal se comprobó que la transmisión de datos funcionaba hasta unos 5 ms de tiempo de canal, pero este valor depende fuertemente del tiempo de ejecución del *loop* principal por que debería ser re-estudiado una vez que flujo principal del programa permanezca estable.

7.2.1.3 Servicio para la asociación automática de dispositivos

El servicio que brinda la capa física para cumplir esta tarea es el de sincronizar el dispositivo nuevo con el FH una vez que escucha una trama válida. Esta tarea se lleva a cabo en la rutina `enganchar()` la cual escucha en una canal fijo (se asume que este canal está libre de interferencias que puedan inutilizar el canal y que hay al menos una base emitiendo tramas) hasta que recibe una trama válida. Entonces guarda los parámetros de FH e inicia el timer que modifica la variable `canal`. La capa de aplicación es responsable de asegurar que se transmitan tramas de sincronismo en el canal preseleccionado, para evitar demoras innecesarias en la asociación de un dispositivo a la red.

Para hacer este método más robusto es necesario tener varios canales para enganchar almacenados en una tabla, de forma que si un canal permanece bloqueado por mucho tiempo se puede pasar al próximo canal en la tabla. Las tramas de sincronismo deberían ser transmitidas en los canales almacenados en la tabla de los dispositivos remotos para evitar demoras en el enganche.

7.2.1.4 Medición de potencia RF

Para poder realizar CSMA en capa MAC es necesario saber si el canal está ocupado o no. Para determinar esto la capa física utiliza la señal analógica, RSSI, proveniente del ATR 2406. La RSSI varía entre 0 y 2 volteos como se muestra en la figura del punto 5.2.2.2, luego se pasa por el conversor A/D y se obtiene un valor discreto de la señal el cual se compara con un valor de referencia.

Para un correcto funcionamiento de este mecanismo es necesario determinar dos valores. El primero corresponde al tiempo que insume la conversión ya que de ser grande podría afectar considerablemente el rendimiento del sistema. Consultando la hoja de datos del micro se observó que la conversión dura 13 ciclos de reloj, que implica un tiempo de 0,94 µs, que bien puede despreciarse.

El segundo valor que debemos determinar es el umbral de RSSI sobre el cual se afirma que el canal esta libre. Este umbral depende de la habilidad del demodulador para lidiar con interferencias en el mismo canal. La prueba ideal para determinar este umbral es transmitir

paquetes aumentado el nivel de interferencia (acercando el elemento que emite la interferencia) y observar hasta que valor de interferencia se reciben los paquetes.

Debido a que no se contó con tres dispositivos en el momento de probar esta función, la prueba realizada constó de transmitir paquetes desde la distancia máxima en línea vista (40 metros) y observar como se comporta la RSSI, se tomo un valor ligeramente inferior a los valores observados como umbral (`adc_val` = 50), por encima del cual se afirma que el canal está ocupado. Este valor esta estrechamente ligado al método detección utilizado. Ya que `adc_val` es un "promedio" de una cantidad determinadas de muestras que se eligió de forma tal que `adc_val` sea máximo (mayor a 230) para una señal que es transmitida de manera continua desde una distancia cercana.

Si se deseara disminuir el consumo al máximo sería conveniente bajar la habilitación de A/D (ADEN), cuando no se use el conversor, aunque esto implicaría un tiempo mayor (25 ciclos de reloj) en la conversión del primer valor.

Para disminuir aún más el tiempo necesario para afirmar que el canal está libre, debería utilizarse el comparador analógico del micro que sólo demora 2 ciclos de reloj en realizar la comparación y calibrar la comparación para el nivel de umbral encontrado.

7.2.1.5 Generación de números aleatorios y tiempos de espera

Para la generación de números aleatorios se utilizará el *true random generator* que viene implementado en el *firmware* "*Smart RF* "del kit de desarrollo, el cual escucha el ruido en la antena para generar el número aleatorio de 8 bits. El algoritmo consiste básicamente, en verificar que el canal esté libre utilizando la señal RSSI, luego muestrea el `RX_DATA` a intervalos de 5µs aproximadamente.

Para generar los tiempos de espera, se utiliza el timer1 de 16 bits. Esto permite esperar tiempos hasta 4,85 s (con el mayor de los prescalers, de 1024). La función que requiere el tiempo de espera es la encargada de inicializar el timer y de definir que acción debe tomar la ISR, al transcurrir el tiempo configurado, asignándole un valor preestablecido a la variable `estado_tmr`.

Para generar los tiempos de espera necesarios para configurar el transreceptor se utilizaron funciones basadas en instrucciones NOP (*No Operation*) del micro. Estas funciones están en el módulo `misc.c`, como por ejemplo `wait_n_1ms()`.

7.2.2 Capa MAC

La capa MAC debe controlar como los dispositivos acceden al medio, lo que incluye mantenerlos sincronizados a la red, ejecutar el algoritmo CSMA para reducir la probabilidad de colisiones y asegurar la integridad de los datos.

7.2.2.1 Sincronismo

Los dispositivos se mantienen sincronizados utilizando campo de la trama de capa MAC: `timer_state`. Este campo contiene el valor de contador `TCNT0` del `timer0`, que es el tiempo que se lleva en un determinado canal. Antes de transmitir cualquier trama un dispositivo actualiza el `timer_state` con el valor de su contador, así mismo el receptor actualiza su contador al recibir cualquier trama con el valor `timer_state`. Para minimizar las distancias entre tiempos transmitidos/recibidos se lee/escribe el contador en la ISRs de la `UART0` en el momento de transmitir/recibir el byte.

Este método se verificó utilizando el kit de desarrollo, los dos dispositivos permanecen sincronizados indefinidamente (más de 2 horas por razones prácticas), recibiendo tramas en períodos que iban de 1 s a 10 s. Para visualizar el sincronismo se programó los dispositivos del kit de desarrollo para que una vez enganchados, muestren el canal actual cada 100 saltos. Esto presentó sus dificultades, porque era necesario mantener sincronizado también el contador de saltos de canal por lo fue ineludible enviarlo en la trama de sincronismo.

Además el tiempo de escritura al LCD es mayor que el tiempo de canal, por lo que hubo que tener especial cuidado de donde se hacían las escrituras al LCD, para que el canal mostrado sea el correcto.

Existe una diferencia inherente entre el valor de `timer_state` leído y el valor actual del timer del dispositivo que lo envia y el de este con otros dispositivos, debido a tiempos de propagación y procesamiento, que no son considerados al actualizar el reloj. Si este valor puede acotarse por un supremo que sea mucho menor que el tiempo de canal, los efectos son mínimos y el método diseñado en el capítulo 4 donde la ventana de transmisión es un poco más corta que la de recepción asegura que los receptores estén escuchando en le canal correspondiente en el momento que un dispositivo transmite.

Las pruebas que validen cualquier teoría deben hacerse con varios dispositivos funcionando, haciendo posible generar las secuencias de eventos típicos de una red. En el entorno particular en que estos métodos fueron desarrollados y probados el comportamiento fue el esperado.

7.2.2.2 Control de Acceso por Detección de Portadora

Con el objetivo de disminuir la probabilidad de colisiones el *firmware* implementa un protocolo de acceso al medio, que consiste en verificar si el canal esta vacío antes de transmitir. Para ello utiliza el servicio de capa física que muestrea la señal RSSI proveniente de transreceptor. Para el que caso en que la trama sea iniciada por el mismo dispositivo, si el canal esta libre se transmite la trama, si está ocupado toma un número aleatorio e inicia el timer1, al interrumpir intenta de nuevo. Si el mensaje a enviar es una retransmisión primero espera un tiempo aleatorio y luego intenta transmitir.

De la manera que el código está implementado una vez que se decide transmitir, no ejecuta otra tarea hasta que termina esta. La transmisión se lleva a cabo, sólo si está libre el canal y está habilitada la transmisión RF (`hab_ventana`).

Las rutinas que ejecuta el control de acceso al medio son `set_transmit()`, que inicia el timer1 en un tiempo aleatorio y la función `get_adc_val()`, la cual muestrea el valor de RSSI para compararlo con el umbral (`adc_val`=50). El número aleatorio se obtiene de una tabla la cual es llenada utilizando los servicios de capa física descriptos anteriormente.

Es importante aclarar que la eficiencia de este algoritmo influye considerablemente sobre el rendimiento total del sistema ya que cuanto más demore en transmitir una trama, menor es la tasa de datos que puede alcanzar. Por esta razón es muy conveniente optimizar este método.

El ancho de ventana que habilita para transmitir es configurable pero está acotado por el tiempo máximo de canal, que para la implementación actual son 19 ms. Las tramas RF son de 64 bytes. Lo que implica un tiempo de transmisión de la trama es 550 µs, si a esto se le suma los 300 µs necesarios para la configuración de transreceptor y los tiempos que lleva generar la trama. Podríamos espera una cota máxima 20 tramas por tiempo de canal (unos 540 kbit/s). Por esta razón el rango de tiempos aleatorios generados debe ser mayor que el tiempo de canal (12 tiempos de canal, para números aleatorios de 8 bits) y el paso del orden del milisegundo, para que efectivamente haya una baja probabilidad de colisiones.

El método implementado es bastante ineficiente, ya que fácilmente se puede llegar a transmitir menos de una trama por tiempo de canal (tasa de bit < 27 kbit/s). Este valor puede ser suficiente para algunas aplicaciones pero estaría truncando la capacidad del dispositivo. Si se deseara obtener una mayor tasa de datos podría utilizarse un algoritmo menos robusto pero más eficiente.

Otra forma puede ser ordenando el acceso al medio, dividiendo el tiempo de canal en n menor a 20 ventanas de tiempo y que cada dispositivo intente transmitir en uno de estas ventanas de forma aleatoria o ordenadamente.

7.2.2.3 Control de integridad

El control de integridad utilizado es un tipo sencillo chequeo de redundancia cíclica (CRC) de 8 bit. El algoritmo consiste en realizar un XOR a todos los bytes transmitidos y enviarlo en la trama, el resultado: `checksumm`, es luego comparado con el resultado obtenido en el receptor.

Para comprobar el funcionamiento se transmitió una cantidad 20 tramas varias veces. Luego se visualizó en el display los valores transmitidos y los calculados en recepción para verificar que eran efectivamente iguales.

Un CRC más complejo, con menor probabilidad de que los errores de bits no sean detectados, debería ser usado para asegurar una mejor integridad de los datos.

7.2.3 Capa de Red

La capa de red es responsable de generar las tramas de red, controlar la difusión de paquetes y aceptar configuraciones de los parámetros de red en los dispositivos remotos y bases.

7.2.3.1 Generación de tramas

La capa de red es la encargada de generar una trama de capa de red llenado los campos de la misma y pasarlo a la capa MAC. Esto se logra invocando la rutina `generar_red (largo_trama, dir_destino, tipo_de_trama)`. Esta función se encarga de escribir un número aleatorio en el campo `num_paquete`, el campo `tipo` y `dir_destino` y rellena los campos restantes con los variables globales de cada dispositivo `num_saltos` y `dir_Id`, ver punto 5.3.4.

7.2.3.2 Control de difusión

La capa de red controla la difusión por dos caminos, los cuales se llevan a cabo invocando la función `uc_8 capa_red(uc_8)`, con parámetro de entrada 2. El primero método de control de difusión utiliza el número de paquete, si este se encuentra en la tabla de los últimos 10 paquetes recibidos es descartado.

En este punto hay dos mejoras inminentes a realizar: aumentar el campo `num_paquete` a 2 bytes y achicar el tamaño de la tabla a, 4 valores. Esto reducirá drásticamente la probabilidad de que se descarte una trama nueva por coincidir el `num_paquete` recibido con uno de los valores guardados en la tabla.

El segundo método implementado para controlar la difusión es mediante el campo `saltos`. Si al decrementar `saltos`, el valor es cero, entones se descarta el paquete, si no, guarda el nuevo valor de saltos en la trama de capa de red.

Si el paquete pasa ambos controles de difusión, lo que es indicado por el parámetro de salida de `uc_8 capa_red(uc_8`, entonces la trama se transmitirá por RF, o bien transmitida hacia la capa de aplicación si la dirección destino coincide con la del dispositivo.

7.2.3.3 Configuración de los parámetros de la red

Los parámetros de la capa de red son: `num_saltos` y `dir_Id`. Estas son variables globales del programa: `num_saltos` es valor que se le carga inicialmente al campo saltos de la trama de red y quién determina el tiempo de vida del paquete; `dir_Id` es la variable que guarda la dirección única de cada dispositivo remoto. Estos parámetros son configurables por comandos desde capa de aplicación.

7.2.4. Capa de Aplicación

La capa de aplicación es responsable de inicializar los dispositivos RFB, lo que incluye: reconocer dispositivos Modbus asociados a la RFB remoto, solicitar ID a la base (Rabbit + RFB), e iniciar tramas de sincronismo en el caso de la base. La ejecución de estas funciones se reparte entre el módulo serie.c y el programa principal.

7.2.4.1 Reconocimiento automático del dispositivo

La rutina `reconcer_tipo()` determina si el dispositivo es baso o remoto. Para ello configura un timer para que interrumpa en tiempo `t_int1`, y envía una trama preestablecida por el puerto serie. Si la placa está conectada a un Rabbit entonces este responderá con una trama de confirmación. La variable global `tipo` se hace igual 1, lo que

indica que es base. Si el Rabbit no contesta y el tiempo `t_int1` expira, entonces `tipo` es igual a 0 y el dispositivo es remoto.

Esto permite tener un único *firmware* para ambos dispositivos ya que el hardware no incluye un switch por el cual se seleccione base o remoto y permite, además, que un dispositivo remoto puede ser utilizado como base y viceversa.

En las pruebas realizadas con kit de desarrollo la configuración de tipo se hacia utilizando el LCD y los botones. Con el cual se comprobó la ejecución de las funciones especificas de cada tipo de dispositivo. Para una mayor comprensión de cómo se ejecuta leer el punto 7.5 donde se explica el flujo principal del programa.

7.2.4.2 Asociación de dispositivos a la red

Para que un dispositivo remoto pueda asociarse a la red debe enviar un pedido de asociación a la base. Previo a esto debe solicitar el servicio de capa física que reconoce la red e inicia el FH: `enganchar()`.

El Rabbit conectado a la base es responsable de actualizar su tabla de dispositivos conectados a la red e iniciar la trama que confirma el pedido asociación, conteniendo la dirección de red que debe tomar el dispositivo. Un dispositivo remoto que no recibe una trama en un tiempo, mucho mayor que tiempo entre tramas de sincronismo (`n_saltos_rx> max_saltos_rx`), deshabilita el timer que comanda el cambio de canal y vuelve a ejecutar la rutinas de inicialización antes descriptas.

La solicitud de asociación de dispositivos no ha sido implementada: la razón es que aún no es posible generar un entorno típico, que enfrentaría un dispositivo RFB al querer asociarse, por falta de dispositivos RFB funcionado correctamente.

7.2.4.3 Iniciar tramas de sincronismo

Esta capa, en el caso que sea base, es también la responsable de iniciar tramas de sincronismo que permitan a los dispositivos asociarse y sincronizarse. Para ello utiliza la variable `n_saltos` la cual cuenta el número de veces que se pasa por el canal 40. Cuando `n_saltos_tx` supera a `max_saltos_tx` se envía una trama de sincronismo con dirección destino igual a 0 para que se difunda por toda la red. El tiempo de tramas de sincronismo puede configurarse a través de la variable `max_saltos_tx` y vale 89 por (tiempo_ch) por (max_saltos_tx). Esto nos ahorra usar un timer para esta tarea, ya que son un recurso limitado y muy requerido.

Para la mayoría de las pruebas se utilizó: `tiempo_ch` = 20 ms y `max_saltos_tx = 8`. Esto resulta en un tiempo entre tramas de sincronismo de 14 segundos.

7.2.4.4 Transmisión a través de puerto serie

Para transmitir tramas por el puerto serie, el programa utiliza la `UART1` configurada en modo asíncrono, respetando el protocolo de comunicación Base-Rabbit descripto en punto 7.3.2.4. La rutina que ejecuta la transmisión serie es `transmitir_serie()`. Para el caso del remoto, el dispositivo simplemente transmite el *payload* de capa de red a través del puerto serie.

7.2.4.5 Configuración de parámetros

El cliente a través de ethernet y del rabbit, puede modificar los parámetros de la capa de red y del FH de los distintos dispositivos. Para el caso de la base, la configuración se realiza en la rutina que procesa la trama serie proveniente del Rabbit. Para el caso del remoto, la trama de configuración es recibida por el remoto vía RF. La actualización de los parámetros la realiza la función `uc_8 capa_de_red(uc_8)`, ya que es la encargada de procesar las tramas RF.

7.2.5 Flujo del programa principal

El flujo principal del programa afecta sensiblemente el rendimiento del sistema, ya que determina el tiempo máximo que el microprocesador puede demorar en procesar los datos de entrada al sistema. El programa principal se escribió con esta idea, tratando de minimizar el tiempo desde que la trama es recibida hasta que es reenviada a través de uno de sus puertos de salida

7.2.5.1 Banderas

El flujo principal es modificado principalmente por tres banderas cuyo significado se explica a continuación.

Trama _ valida

Indica que una nueva trama válida completa se ha guardado en el buffer de recepción RF: `rf_rx_buffer`. La bandera es modificada en la rutina que atiende la interrupción de recepción de datos.

Trama_serie

Indica que una trama serie se ha guardado en el buffer de recepción de tramas serie: `rx_buffer`. La bandera se levanta en la rutina que atiende las interrupciones del puerto serie.

Hab_ventana

Otro de los eventos que modifica el flujo principal es la bandera que habilita la transmisión de datos por RF, la cual conmuta en la rutina que atiende la interrupción B del timer0.

7.2.5.2 Tareas

Las tareas que deben ejecutarse en el flujo principal, una vez que los dispositivos involucrados se han asociado a la red son:

1. **Inicialización**

   Ejecuta el `reconocimiento_de_tipo()`: base o remoto y la rutina de enganche a la red si el dispositivo es remoto.

2. **Control de sincronismo**

   Ejecuta la tarea que decide si debe enviar una trama de sincronismo, caso del dispositivo es base, o si el dispositivo es remoto decide si hay que volver a las rutinas de inicialización que asocian el dispositivo a la red, porque el dispositivo se ha des-sincronizado.

3. **Engancharse a la red**[*2]

   Esta tarea consiste en escuchar en un canal fijo hasta recibir una trama válida y el dispositivo se engancha al salto de frecuencia, la rutina que ejecuta esta tarea es `enganchar()` en `control_rf.c`.

4a. Procesar trama RF

Esta tarea conceptual responde a una `trama_válida` y consiste básicamente en decidir si la trama debe ser retransmitida, enviada por el puerto serie, es una trama de configuración o simplemente descartada. Esta tarea esta implementada en la rutina `capa_de_red()`. En caso de que la trama deba ser retransmitida actualiza la trama de red, en el caso que la trama sea de configuración modifica los parámetros correspondientes, sólo para el caso remoto**[*1]**.

4b. Procesar trama serie

Decide si la trama serie recibida debe ser enviada por RF o es una trama de configuración, en el caso que sea de configuración (sólo para la base) guarda los nuevos valores de los parámetros**[*2]**.

Notas:

[*1] Las tramas de configuración para la base viene por el puerto serie y para el remoto por RF, esta es la razón por que son incluidas en tareas distintas.

[*2] Se asume que hay al menos un dispositivo base activo.

5. **Transmitir trama por RF**

Esta tarea consiste en ejecutar los algoritmos de acceso al medio, definidos en capa MAC y transmitir la trama vía RF.

6. **Transmitir trama por serie**

Consiste obtener la trama validada desde capa de red y transmitirla por el puerto serie. Para el caso de la base debe respetar el protocolo base - rabbit, el cual se describe posteriormente.

7.2.5.3 Arquitectura del software.

La elección de la arquitectura elegida se basa en **Round Robin con Interrupciones**, ya que es la forma más simple de ejecutar las tareas mencionadas asignándoles prioridad.

De las interrupciones que modifican la ejecución de tareas, las rutinas que atienden al transreceptor son la de mayor prioridad:

```
interrupt[USART_RXC] void UsartReceive(void){
                    // deshabilito interrupciones
                    // guardo dato de entrada
                    // modifico trama_valida
                    // habilito interrupciones
}
interrupt[TIM0_COMPB] void Timer0CompareB(void){
                   // hago lo que hago
                   // conmuto hab_ventana
}
interrupt [USART1_RXC] void usart_rx_isr(void){
            // guardo dato de entrada
            // modifico trama_serie
}
```

A continuación se muestra el main()de forma esquematizada:

```
main(){
      inicialización();
      while (true){
            if (trama_valida) {…}// Ejecuto tareas
            if (trama _ serie){…}
            if (hab._ventana) {…}
    }

}
```

Es preciso tener claro que este software asume que la cantidad de tramas que llegan por segundo es chica, comparado con el tiempo que lleva procesar una trama, hipótesis que es validada mediante el correcto funcionamiento de la red propuesta. Por lo tanto el programa no almacena tramas consecutivas, sino que una vez que llega una trama debe ejecutar todas las tareas correspondientes antes de volver a aceptar otra trama; es muy importante, entonces, una vez que se levanta una bandera las tareas correspondientes sean ejecutadas lo más rápido posible.

Para explicar de forma gráfica como es el flujo de programa se utiliza un diagrama estados, figura 7.1, que muestra la dinámica con la que se realizan las tareas, donde existe un único estado estable (estado 4), que el programa ejecuta mientras no tiene tareas que cometer. El resto son estados transitorios que el programa ejecuta según las banderas.

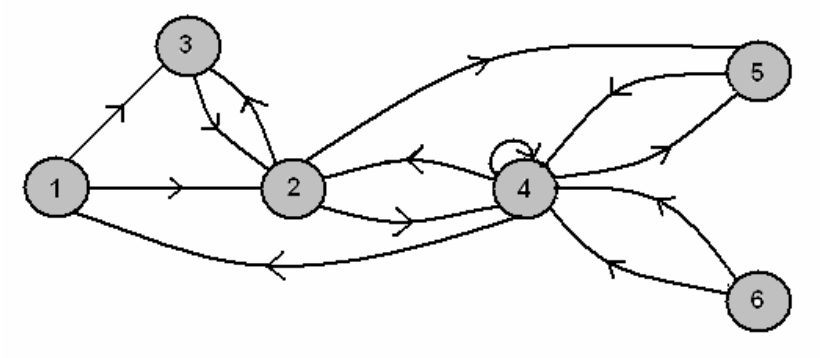


Figura 7.1: Diagrama de estados del programa.

Para mostrar como son las transiciones de estados se utiliza un diagrama de flujo, figura 7.2, el cual especifica como las banderas alteran el flujo programa. En él puede verse claramente como difieren las secuencias de estados dependiendo si el dispositivo es base o remoto. En este diagrama se omitió poner la dependencia del flujo con hab_ventana para no acomplejar demasiado el diagrama. Al habilitarse una nueva ventana, cualquiera sea el estado en el cual se encuentra el programa, al terminar la tarea correspondiente al estado se va al estado 2.

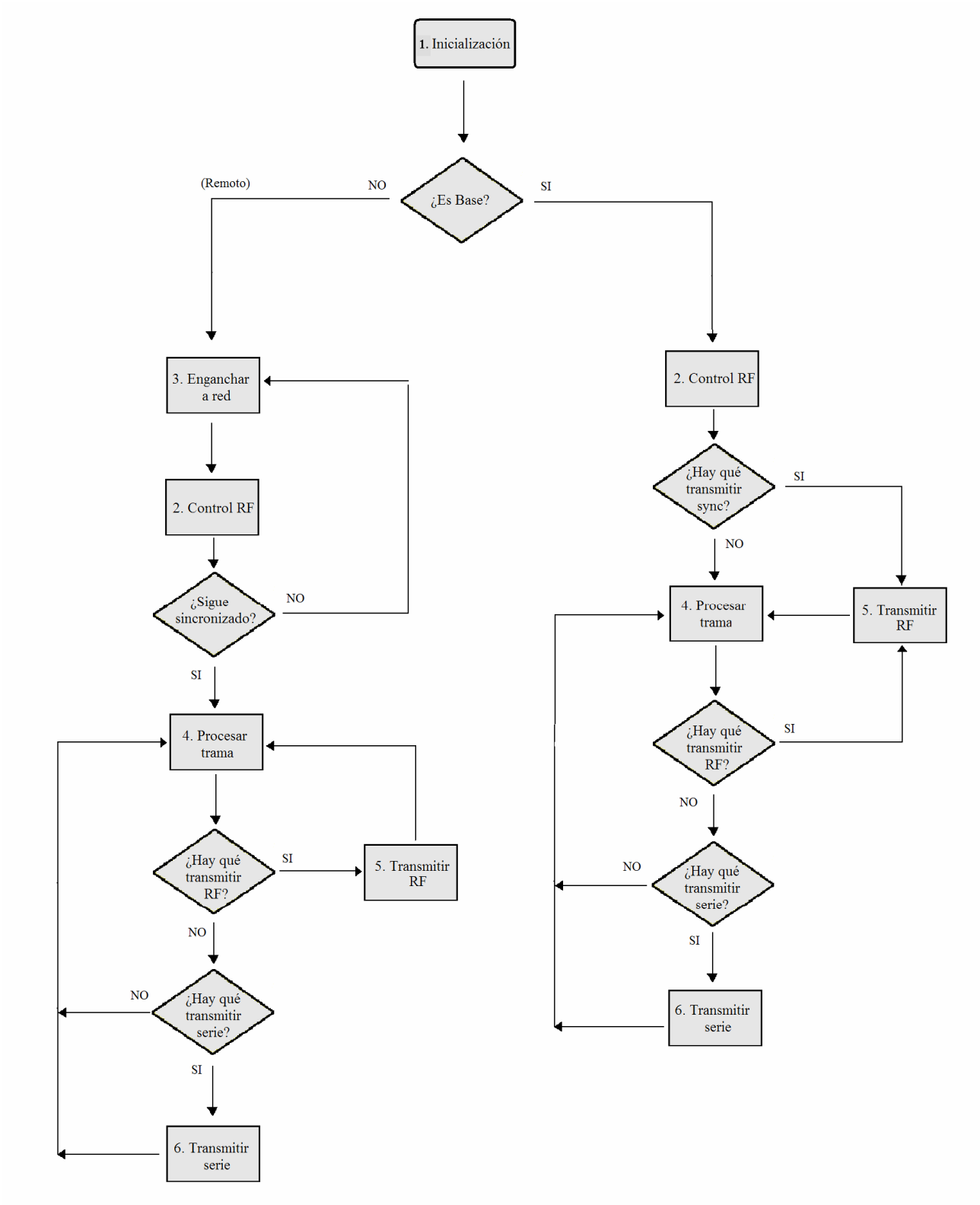


Figura 7.2: Diagrama de flujo del programa: Muestra como varía el flujo según las banderas.

Hay que tener presente lo importante que es que este código tome los datos del buffer lo antes posible para evitar que lo sobrescriban; luego ejecutar las tareas correspondientes lo más rápido posible para poder aceptar tramas más frecuente.

Se decidió agrupar las tareas principales en estados, tareas que además de ser conceptualmente diferentes, son complejas de por sí y demoran mucho tiempo, relativo al tiempo de canal.

Existen dos formas de implementar las tareas en el orden descrito. Una es utilizando sentencias **if** anidadas. Sin embargo esta forma dificulta la depuración ya que se debe entrar las subrutinas para determinar el estado.

Otra forma más amigable que no sólo ayuda a visualizar mejor los estados, sino que también agrega la posibilidad de alterar fácilmente las secuencias de estados. Para ello se utiliza una variable que guarda el estado: `estado_tx_rx`. Para realizar una transición de estado simplemente se escribe en esta variable el valor del siguiente estado.

A continuación una versión esquematizada del main():

```
interrupt[TIM0_COMPB] void Timer0CompareB(void){

      if(hab_ventana==0){
            hab_ventana=1;
            estado_tx_rx=2;
      }else hab_ventana=0;
}
void main(void){

      Init_config();

      while(true){

            switch (estado_tx_rx) {
                  case 1 :
                        reconocer_tipo(); // estado_tx_rx  → 2 ó 3
                  break;
                  case 2:
                        control_RF();  //estado_tx_rx  → 3,4 o 5
                  break;
                  case 3:
                        enganchar();  //estado_tx_rx  → 2
                  break;
                  case 4: // remoto y base  esperando

                        if(trama_valida){prescesar_trama_RF();}
                        if(trama_serie){prescesar_trama_serie();}
                        // estado_tx_rx  →  4,5 ó 6
                  break;
                  case 5:
                        transmitir_RF();  //estado_tx_rx → 4
```

```
                    break;
                    case 6:
                          transmitir_serie(); //estado_tx_rx → 4
                    break;
              }
        }
}//end main
```

La totalidad del código implementado está disponible en el Anexo, dividido en módulos, e incluye los módulos implementados exclusivamente para depurar código utilizando el kit del ATR 2406: aplicación_lcd.c y interface.c.

7.2.5.4 Modificaciones futuras

Varias modificaciones podrían realizarse al código para mejorar el rendimiento total del sistema. Una de ellas consiste en almacenar más de una trama para poder lidiar con picos de tramas/seg. Otra es optimizar la ejecución de tareas asignándole prioridad a una sobre otras y permitir que se ejecuten tareas simultáneamente, siempre y cuando no necesiten los mismos recursos.

Otra modificación conveniente para aprovechar las características del compilador es alocar todas las banderas (actualmente son bytes) en registros del micro como variables de bit, para ahorrar memoria, ya que hay poca disponible.

7.2.6 Conclusiones.

Si bien el software no esta completamente verificado y faltan iteraciones para poder validarlo, el grueso de la funciones especificadas en el protocolo RF fueron implementadas. Dentro de los logros alcanzados están el buen dominio del entorno de desarrollo, así como un conocimiento extenso de los micros y transreceptor utilizado.

Además se hizo una reseña de las posibles mejoras futuras en cada una de las tareas que servirán para optimizar la utilización de los recursos y así poder crear un producto más robusto y de mejores prestaciones.

7.3 Desarrollo en Rabbit RCM 3700

7.3.1 Introducción

El módulo Rabbit RCM 3700 es un componente fundamental en la red inalámbrica. El Rabbit forma parte del dispositivo Base RF junto con una RFB.

Recordamos que uno de los cometidos de la Base RF es ser la interfaz de la red para el Cliente que se encuentra en un computador, y principalmente ser el responsable de administrar y gestionar la red inalámbrica.

Dentro de la Base RF, el Rabbit se comunica con el Cliente en una red Ethernet, por esto debe ser flexible para la aplicación general (Domótica por ejemplo), brindar una visión completa del estado de la red y facilitar la instalación y puesta en marcha del sistema.

Además el Rabbit se comunica con una RFB siendo el punto de entrada de la red inalámbrica.

La comunicación entre estos módulos que conforman la Base es con un protocolo específico, sobre Serie RS-232. Más adelante precisaremos sobre está comunicación y sobre el protocolo entre el Rabbit y el Cliente.

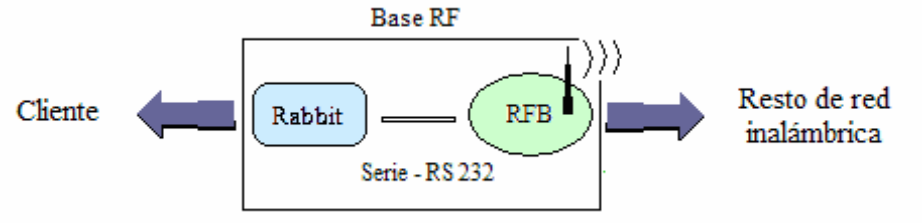


Figura 7.3: Componentes de Base RF.

Esencialmente el Rabbit se encarga de proveer la interfaz de la red al Cliente, siendo el punto de entrada al sistema.

El Cliente puede intercambiar datos con los dispositivos que estén conectados a cada Remoto (por ejemplo dispositivos Modbus), y también realizar configuraciones al sistema y conocer su estado.

Para lograr esto, es el Rabbit quien es el responsable de tener toda la información de la red inalámbrica.

Las propiedades de la red en un determinado momento son:

- Cantidad de Remotos presentes.
- El identificador de red RF de cada Remoto.
- La cantidad de dispositivos que están asociados a cada Remoto.
- Las direcciones Modbus de todos los dispositivos Modbus por Remoto.
- Los parámetros del FHSS (*Frequency Hopping Spread Spectrum*).
- El Tiempo de vida de cada nodo de la red.
- Una estimación de la cantidad de saltos que dista cada Remoto de la Base.

Otra función relevante que cumple el Rabbit es la inicialización del sistema. Posteriormente se explicará detalladamente este proceso.

7.3.2 Diseño

En la etapa de etapa de diseño, se identificaron los dos objetivos principales que le competen al Rabbit. Estos objetivos son: gestionar la red junto con la RFB y recibir clientes de la red, siendo el punto de entrada del sistema.

7.3.2.1 Gestión de la red inalámbrica

Refiriéndonos al primer objetivo, comando de la red inalámbrica, el Rabbit es responsable de las siguientes tareas: inicializar el resto de los componentes de la red y mantener estado actualizado de la misma. Como la administración de la red es compartida con la RFB, es necesario establecer un protocolo de comunicación que mantienen sobre Serie RS -232. El protocolo se detalla en la sección 7.3.2.4.

El Rabbit es el encargado de la inicialización del sistema, y debido a esto tiene grabado (en tiempo de compilación) los parámetros por defecto de la red inalámbrica. En la figura 7.4 se muestra el conjunto de código que lo configura.

```
// PARÁMETROS DE RED RF
#define DEF_NUMERO_DE_SALTOS     15  // Tiempo de vida (número de saltos).
#define DEF_FH_TIEMPO_DE_CANAL   20  // Tiempo de permanencia en un canal (ms).
#define DEF_FH_SALTO_DE_CANAL     7  // Salto de canal (canales).
#define MAX_DISPOSITIVOS_MODBUS  64  // Cantidad de dispositivos Modbus admitidos.
#define MAX_LARGO_TRAMA_RF       40  // Largo máximo de la carga en trama RF (bytes).
```

Figura 7.4: Definición de parámetros por defecto de red RF.

Unas de las primeras funciones que ejecuta el Rabbit al comenzar, es la inicialización de la RFB. Envía por serie RS-232 el siguiente paquete:

Número de bytes	ID RF	Tipo	Carga		
5	1	156	DEF_FH_SALTO_DE_CANAL	DEF_FH_TIEMPO_DE_CANAL	DEF_NUMERO_DE_SALTOS

Figura 7.5: Paquete que inicializa la RFB.

Este paquete configura los valores por defecto de la red RF: tiempo de vida de paquetes, y los parámetros de FHSS. Luego se espera un tiempo determinado la respuesta de la RFB; si esta no llega, se retransmite el mismo mensaje. Esto se repite hasta que se tenga una confirmación por parte de la RFB, de forma de garantizarse que se efectúe la configuración.

Posteriormente la RFB, como ya se mencionó en el Capítulo 5.3.5, empieza a enviar mensajes de sincronismo, para que los dispositivos Remotos puedan engancharse a la red.

Una vez sincronizado el dispositivo Remoto, este envía a la Base un pedido para asociarse a la red de identificación e inicialización.

Este pedido es recibido por la RFB, y esta luego se lo envía al Rabbit. El Rabbit verifica que la cantidad de dispositivos existentes más la cantidad que se agregarían no supera el máximo fijado, en caso que no supere el límite, el Rabbit responde el identificador de red inalámbrica para el nuevo dispositivo y el tiempo de vida.

Número de bytes	ID RF	Tipo	Carga	
4	ID otorgado	160	ID otorgado	Número de Saltos

Figura 7.6: Paquete que inicializa a Remoto.

El tercer byte Tipo vale 164 decimal (10100100 en binario), lo que implica una asignación del ID RF y del Tiempo de vida.

No es necesario configurarle los parámetros de FHSS porque cuando el dispositivo Remoto logró integrarse a la red, los mensajes sincronismo le proporcionaron esa información.

7.3.2.2 Mantener estado de la red inalámbrica

El Rabbit se ocupa de almacenar el estado de la red inalámbrica y de mantenerla actualizada. Los parámetros del FHSS se guardan en variables globales, las cuales pueden ser cambiadas por el Cliente.

Excepto los parámetros del FHSS, el estado de la red se guarda en memoria, en una estructura llamada Tabla de Direcciones. Esta estructura es una matriz, que tiene guardada información de cada nodo de la red, y el vínculo de cada Remoto con sus eventuales dispositivos Modbus.

En el ejemplo de la figura 7.7 se muestra el uso de la Tabla de Direcciones, para lo cual se supone una topología de la red en determinado instante.

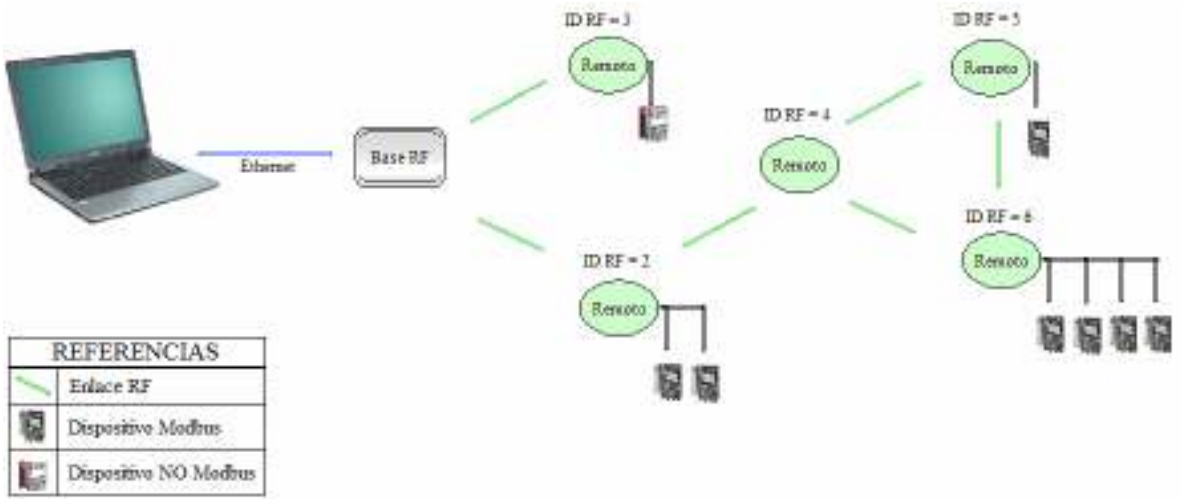


Figura 7.7: Ejemplo de topología de red inalámbrica.

A continuación se presenta la Tabla de Direcciones correspondiente al ejemplo anterior.

Identificador RF	Dirección Modbus	Tiempo de Vida (saltos)	Estimación de distancia hacia la base (saltos)
1	0	Número de Saltos 1	0
2	5	Número de Saltos 2	1
2	84	Número de Saltos 2	1
3	0	Número de Saltos 3	1
4	0	Número de Saltos 4	2
5	7	Número de Saltos 5	4
6	56	Número de Saltos 6	3
6	81	Número de Saltos 6	3
6	89	Número de Saltos 6	3
6	93	Número de Saltos 6	3
0	0	0	0
...	...	Vacío	...
0	0	0	0

Tabla 7.1: Tabla de Direcciones.

La primera fila de la tabla refiere siempre a la Base RF.

Cada Remoto le corresponde al menos una fila en la tabla, en la que indica el Tiempo de vida de sus mensajes, y una estimación de la cantidad de saltos que lo separan de la base.

La segunda columna se reserva para indicar la dirección Modbus del dispositivo Modbus, en caso de no tener dispositivo conectado (caso del Remoto 4) o que el dispositivo no sea Modbus (caso del Remoto 3) este espacio vale 0.

Si el Remoto está vinculado a más de un dispositivo Modbus, ocupará una fila por cada dispositivo. Este es el caso de los Remotos 2 y 6, donde las propiedades de Tiempo de vida y Estimación de distancia hacia la base siempre se repiten en cada fila.

El caso del Remoto 5 (ID RF igual 5), la estimación es de 4 saltos, aunque según la topología sólo hay 3 saltos; esta diferencia se ocasionó porque en el último mensaje enviado por el Remoto 5 tomó la siguiente ruta:

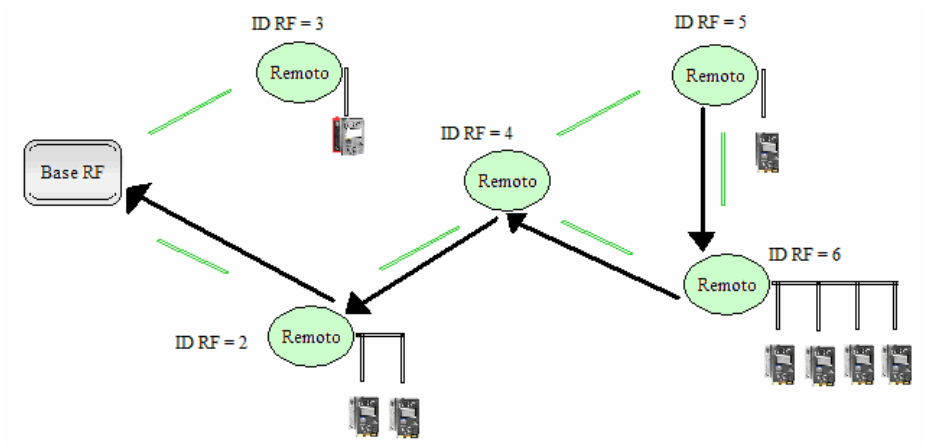


Figura 7.7: Camino del paquete en 4 saltos.

La Tabla de Direcciones sufre modificaciones con los siguientes eventos:

- Integración de un Remoto al sistema.
- Configuraciones por parte del Cliente a la Base o a remotos.
- Desagregación de un Remoto de forma involuntaria del sistema.

  Cuando la RFB deja de detectar en la red a un Remoto, se lo comunica al Rabbit y el Rabbit quita del registro de este Remoto de la tabla.

- Recepción de mensajes en la Base.

  El Rabbit conoce el Tiempo de vida con que llegan los mensajes de la red RF, y además sabe a partir de la Tabla de Direcciones con cuanto tiempo de vida salió del remitente; por lo tanto restando estos valores estima la cantidad de saltos que se produjo en la comunicación.

7.3.2.3 Servidor

Para cumplir el segundo objetivo de ser punto de entrada de la red inalámbrica, se implementó en el Rabbit un Servidor.

Las características básicas de un Servidor son:

- Espera de forma pasiva la conexión de clientes.

  El sistema debe esperar de forma pasiva las conexiones de clientes que requieran usar o controlar el sistema; estos clientes pueden conectarse en cualquier momento.

- Acepta un número arbitrario de clientes.

  Existe una cantidad máxima de clientes que a la vez pueden utilizar el sistema.

- Ofrece un conjunto fijo de servicios.

  En este caso los servicios son: la conectividad inalámbrica, configuración y control del sistema.

El mecanismo para identificar a cada cliente es usar los números de puertos. En el caso de TCP los números de puertos del Servidor y el Cliente y sus direcciones IP determinan de manera única en la red la conexión.

Esta forma de caracterizar una conexión introduce el concepto de *socket*. Un *socket* mantiene la información de toda una conexión y el estado en que se encuentra.

Se diseño usando el protocolo de capa de transporte TCP (*Transport Control Protocol*) y no UDP (*User Datagram Protocol*), porque es más conveniente para este caso ya que se va a usar en una aplicación de control, siendo importante que todos los datos enviados se aseguren de llegar correctamente a destino.

TCP asegura de que los paquetes llegan y que se reciben en orden correcto; esta orientado a conexiones y realiza un control de flujo entre el Cliente y el Servidor. Esto último es útil ya que el Rabbit no sólo realiza tareas de Servidor.

Tipos de Conexiones

El Servidor desarrollado, presenta tres tipos distintos de conexiones, separando de esta manera servicios que provee a los clientes.

Los tipos de Conexiones son los siguientes:

- Conexión Modbus
- Conexión Configuración
- Conexión Drain

Conexión Modbus

Es el puerto específico para usar el protocolo Modbus. El Cliente desde el ordenador, es el maestro Modbus y los actuadores que están del lado de los dispositivos Remotos, son los esclavos Modbus.

En este tipo de Conexión, el Rabbit asume que las tramas cumplen el protocolo Modbus. Las tramas que soporta el sistema son de 8 bytes de largo, lo que abarca la mayoría y más importantes tramas Modbus.

El sistema asegura que los mensajes que intercambian entre maestro y esclavos lleguen a destino.

Conexión de Configuración

Uno de los requerimientos es que la red sea controlada mediante una PC; para esto se pensó en la Conexión Configuración. Esta conexión se utiliza exclusivamente para la configuración y control del sistema RF.

La modificación de propiedades y parámetros, o la solicitud del estado, se dividen según se ejecute sobre un dispositivo remoto o la RFB Base. En la sección 7.3.3.3 se detallan sus funciones.

Configuración a la RFB

Para conocer el estado de la red el Cliente puede solicitar a la Base un reporte de estado de la red. Este reporte indica el estado general actual de la red, como son los parámetros de *Frequency Hopping* (FHSS) actuales, el Tiempo de vida asignado a la Base, discriminando

por cada dispositivo remoto que se haya reconocido, la cantidad de esclavos Modbus que tiene asociado con sus respectivos identificadores Modbus, y también el Tiempo de vida de cada remoto.

Además el reporte indica una estimación de número de saltos a los que se encuentra cada Remoto de la Base.

Se puede modificar en la Base lo siguiente:

- Los parámetros generales de la red como: *Frequency Hopping* (FHSS): Tiempo de Canal y Salto de Canal.
- Tiempo de vida (número de saltos) de sus paquetes.

Cabe destacar que siempre el ID de la Base es 1.

Configuración a Remoto

Si bien es posible conocer el estado de todos los dispositivos remotos cuando se le interroga a la Base, es útil que se pueda consultar puntualmente un reporte de estado a cada Remoto.

Este reporte informa cual es su tiempo de vida, la cantidad de dispositivos Modbus que posee y los respectivos identificadores Modbus de cada uno.

Se puede realizar con cada dispositivo Remoto:

- Un cambio de identificador RF.
- Quitarlo o separarlo de la red.
- Cambio de ID de destino de los mensajes que envíe. Por defecto el ID de destino de cada RFB Remoto es 1 (ID de la RFB Base). Con esto es posible lograr una comunicación entre dos RFB Remotos sin centralizar en la base. En este caso la base RF es un sólo un posible intermediario.

Conexión Drain

Aunque el sistema tiene el requerimiento de soportar aplicaciones bajo el protocolo Modbus, el sistema es capaz de proporcionar una muy buena utilidad en otros tipos de aplicaciones. Para esto se ideó la Conexión Drain.

Todo lo que el Cliente transmite por este puerto, se hace un *broadcast* a todos los Remotos.

Como las tramas de red inalámbrica tiene un máximo de 40 bytes de carga útil, la cantidad máxima de bytes transmitidos por este puerto son 40 bytes; si es mayor a está cantidad el sistema fraccionará esta trama. Ver sección 7.3.3.3.

Los números de los puertos usados por las tres conexiones se definen como constantes en el código de la forma en que se muestra en la figura 7.8.

```
// NÚMEROS DE PUERTOS
#define NUMERO_DE_PUERTO_MODBUS  4050  // Número de puerto para Conexión Modbus.
#define NUMERO_DE_PUERTO_CONF    4030  // Número de puerto para Conexión Configuración.
#define NUMERO_DE_PUERTO_DRAIN   4040  // Número de puerto para Conexión Drain.
```

Figura 7.8: Declaración de número de puertos.

Un punto importante a precisar es que el Servidor implementado soporta tres clientes a la vez, uno por cada Conexión; aunque nuestro código es fácilmente extensible para que haya más de un cliente por tipo de Conexión; excepto para el caso de la Conexión Configuración, para evitar incoherencias entre configuraciones.

7.3.2.4 Protocolos: Rabbit – RFB y Cliente – Rabbit

En esta sección se describen los protocolos que debe soportar el Rabbit.

Protocolo Rabbit – RFB

El formato de las tramas que se envían desde el Rabbit a la RFB o de la RFB al Rabbit es distinto.

Todos los mensajes que envía la RFB hacia el Rabbit contienen el segmento de la trama de capa red del protocolo RF y el byte *Tipo* que pertenece a la trama de la capa de aplicación.

Aunque para ambos casos el encabezado indica la cantidad de bytes que siguen. Esta información es necesaria para el receptor ya que cuando recibe el primer byte conoce la cantidad que deben llegar, además este campo logra separar los mensajes.

En la siguiente figura se presenta el formato de trama de la comunicación de RFB a Rabbit.

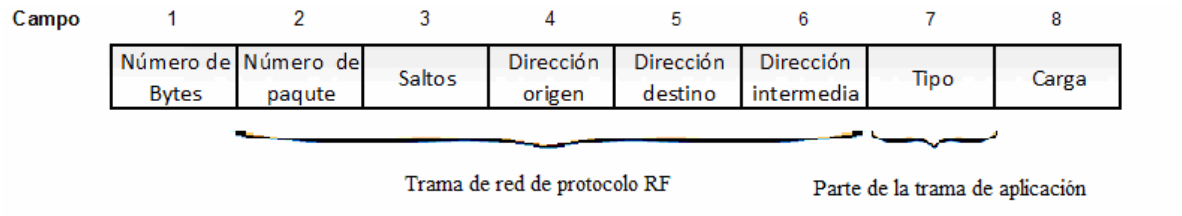


Figura 7.9: Campos de trama de RFB a Rabbit.

Campo	Nombre	Tamaño (bytes)
1	Número de bytes	1
2 a 6	Trama de capa de red del protocolo RF	6
7	Tipo	1
8	Carga	1 a MAX_LARGO_TRAMA_RF del protocolo RF

Tabla 7.2: Descripción de campos de la trama de RFB a Rabbit.

En la siguiente figura se presenta el formato de trama de la comunicación de Rabbit a RFB.

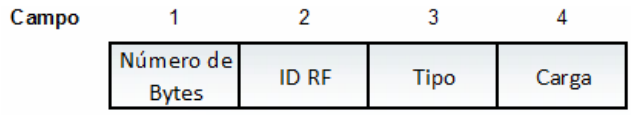


Figura 7.10: Campos de trama de Rabbit a RFB.

Campo	Nombre	Tamaño (bytes)
1	Número de bytes	1
2	ID RF	1
3	Tipo	1
4	Carga	1 a MAX_LARGO_TRAMA_RF del protocolo RF

Tabla 7.3: Descripción de campos de la trama de Rabbit a RFB.

Protocolo Rabbit – Cliente

El protocolo que existe entre el Rabbit y el Cliente funciona sobre los protocolos TCP-IP. El formato que deben tener las tramas depende del tipo de Conexión que se utiliza.

Conexión Modbus

En este caso sólo se esperan tramas de 8 bytes Modbus, donde no es necesario agregar encabezado o cola.

Conexión Drain

En esta Conexión sólo se esperan datos útiles.

Conexión Configuración

Esta trama lleva dos encabezados: *SOF* (*Start Of Frame*) que siempre debe valer 55 en hexadecimal y *Número de bytes* es la cantidad de bytes de la carga más dos. También lleva al final de la trama un *EOF* (*End Of Frame*) de valor FF en hexadecimal; tanto *SOF* y *EOF* son utilizados para identificar y separar estas tramas del buffer de entrada TCP.

ID RF indica a que nodo RF está referida la configuración o la consulta.

Tipo determina la acción a la que se refiere el Cliente. Ver detalles de este valor en la sección 5.3.5.1.

Carga contiene los valores de configuración, estos dependen directamente de *Tipo*.

Figura 712.: Trama de configuración.

7.3.3 Implementación

7.3.3.1 Plataforma utilizada para desarrollo en Rabbit

La programación del Rabbit se implementó en el programa Dynamic C.

Dynamic C es un entorno de desarrollo para programar software en sistemas embebidos. Se ejecuta en un compatible con IBM-PC y está diseñado para su uso con controladores Z-World y otros controladores basados en el microprocesador Rabbit.

Figura 7.13: Logo Dynamic C.

Proporciona la edición, complicación, enlace, programación (carga) y depuración del programa.

Debido a que el lenguaje es similar al de C, tiene todas las declaraciones y las construcciones tradicionales de C, y además extensiones que hacen que sea más fácil de desarrollar.

Este entorno provee una cantidad de bibliotecas y ejemplos de programas que resultaron de gran utilidad.

A continuación se presentan las librerías que utilizamos.

*Librería **DCRTCP***

La librería principal donde se encuentra la familia de funciones y macros del protocolo TCP/IP es DCRTCP.LIB. Esta soporta IP v4 pero no la versión 6. Es configurada a través de la configuración de las macros. Estas macros se definen en tiempo de compilación.

Las macros usadas son:

- **TCPCONFIG**. Se aplica para especificar si las direcciones IP se configuran de forma manual (*#define TCPCONFIG 1*) o si la dirección es asignada dinámicamente por un servidor DHCP (*Dynamic Host Configuration Protocol*) que esté en la red (*#define TCPCONFIG 3*).

  En el primer caso deben definirse cuatro direcciones: la propia dirección IP, la máscara de red, la dirección de la puerta de enlace por defecto y dirección IP del servidor DNS (no siempre necesaria). Estas cuatro direcciones también se definen con macros: MY_IP_ADDRESS**,** MY_NETMASK**,** MY_GATEWAY y MY_NAMESERVER.

  Vale aclarar que estas cuatro direcciones pueden modificarse en la ejecución con la función tcp_config().

- **SOCK_BUF_SIZE**. El valor definido (en bytes) a esta macro es el doble del tamaño del buffer de entrada y del buffer de salida. Por defecto es 4096 bytes, y no puede ser configurado a menos de 600 bytes; se recomienda que sea mayor o igual al MSS (*Maximun Segment Size*) del protocolo. Para IP v4, el MSS es 1460 bytes.

- **USE_RESERVEPORTS.** La definición de esta macro permite el uso de la función tcp_reserveport(). Esta indica el puerto en el que se aceptan conexiones antes de que los recursos estén disponibles.

Existen muchas funciones que pueden aplicarse a un socket TCP. Estas se dividen en tres categorías principales: Control, de Estado y de Entrada / Salida.

Las funciones utilizadas ordenadas por categoría son:

- *Control:* sock_init() y tcp_listen().

  sock_init() inicializa todas las estructuras de datos y el chip Ethernet.

  tcp_listen() dispone a aceptar conexiones.

- *Estado:* sock_bytesready(), sock_established() y tcp_tick().

  sock_bytesready() devuelve la cantidad de bytes listos para leer del buffer del socket.

  sock_established() observa si existe una conexión actual establecida.

  tcp_tick() procesa y cheque nuevos paquetes y devuelve el estado de la conexión, esta debe llamarse periódicamente.

- *Entrada / Salida:* sock_mode(), sock_fastread() y sock_fastwrite().

Existen dos modos de realizar lectura y escritura en un sockets TCP: ASCII (*American Standard Code for Information Interchange*) y binaria. Por defecto, un socket se abre en modo binario, pero puede cambiarse el modo con una llamada a sock_mode().

Librería **RS232**

Dynamic C ofrece una gama completa de apoyo a las comunicaciones serie. La librería RS232.LIB proporciona un conjunto de funciones seriales basadas en buffer circulares. Ofrece funciones no regresan hasta que se termine de transmitir o recibir (*blocking functions*), y funciones que no se bloquean, lo que permite intercalar otras funciones que deben ejecutarse.

Las macros que fueron utilizadas de esta librería son CINBUFSIZE Y COUTBUFSIZE, las cuales fijan el tamaño de los buffers de entrada y de salida respectivamente del puerto serie C.

Las funciones que aplicadas son:

serXopen: Abre el puerto X con una tasa de bits determinada.

serXrdUsed: Devuelve la cantidad de bytes listos para ser leídos del buffer de lectura del puertoX.

serXgetc: Obtiene el próximo byte listo del puerto X.

serXrdFlush: Limpia el buffer de entrada del puerto X.

serXwrite: Transmite la cantidad de bytes especificados de una lista de bytes por el puerto X.

El lugar de las "X" que aparecen en el nombre de las funciones, se sustituye por la letra que corresponde al puerto serie al que quiera referirse. Como se usó el puerto serie C, se cambió por la letra "C".

Librería **STDIO**

Esta librería se utilizó esencialmente para disponer de herramientas para la depuración del programa. Permite usar la función *printf()* para realizar útiles impresiones en la consola del Dynamic C, mostrando valores de las variables y registros.

7.3.3.2 Estructura Conexión

De manera de manipular ordenadamente las conexiones que mantienen con los clientes, en el código se representa una conexión con una estructura llamada de la misma manera: Conexión.

```
// ESTRUCTURA CONEXION.
typedef struct
{
   tcp_Socket socket;       // Representación de socket TCP.
   int state;               // Estado.
   char numero_de_socket;   // Identificador.
   int puerto;              // Número de puerto.
   char TX_buffer[256];     // Buffer auxiliar de transmisión.
   char RX_buffer[1100];    // Buffer auxiliar de recepción.
   char dato_en_espera;     // Indicador de envío en espera.

}Conexion;
```

Figura 7.14: Estructura Conexión.

En el programa se manejan una lista de conexiones. Esta lista contiene tres instancias de Conexión; una conexión de cada tipo: Conexión Modbus, Conexión Configuración y Conexión Drain.

7.3.3.3. Flujo principal

El programa se puede dividir en tres bloques principales. Estos bloques son: Inicialización, Administrador de sockets y Control Serie.

En la figura a continuación se muestra el flujo principal del software.

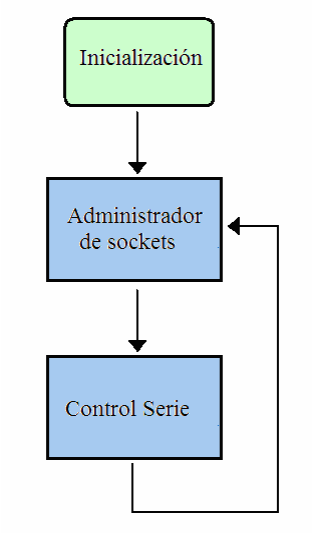


Figura 7.15: Flujo principal.

En la siguiente sección se describen los tres bloques presentes en el diagrama de la figura 7.15.

7.3.3.4. Descripción de bloques

Bloque Inicialización

Este bloque presenta cinco funciones principales. Estas se explican en el orden en que se ejecutan:

- *sockete_init()*

  Inicializa la lista de conexiones, cada conexión con su número de puerto correspondiente. Ver descripción en la sección 7.3.2.2.

  Llama a las funciones sock_init(), la cual es necesaria llamar antes de cualquier otra función de la librería DCRTCP y tcp_reserveport() que permite conexiones en los tres puertos utilizados.

- *abroPuertoSerieC()*

  Abre el puerto serie C, con la tasa de bits con la que se usará. Este valor se define como constante en el código.

- *inicializoRFB()*

  Esta función es bloqueante, es decir, hasta que no se logra inicializar a la RFB no retorna. Esta función manda una trama de configuración con los parámetros de FHSS y el Tiempo de vida, y espera respuesta de la RFB para confirmar el mensaje.

  Antes de repetir se espera activamente un tiempo determinado en una constante llamada TIEMPO_ESPERA_CONF_RFB.

- *inicializoTablaDirecciones()*

  Como dice sus nombre inicializa la Tabla de Direcciones, forma la primer fila de la tabla que corresponde al dispositivo Base de la red inalámbrica y el resto de los lugares con la constante -1, la cual será la que indique al resto de las funciones lugar vacío.

- *inicializoTimerB()*

  Inicializa una variable para el control del Timer B. El Timer B es una timer se utiliza en dos circunstancias: una vez en Control Serie y Administrador de sockets. Más adelante se precisa su uso.

Bloque Administrador de sockets

El Administrador de Sockets es el encargado de manejar la lista de conexiones. La ejecución de cada conexión es alternada; en cada invocación a esta función se atiende a sólo a una Conexión. La ejecución que se realice depende del estado en que se encuentra dicha Conexión. Una vez ejecutado el código del estado correspondiente, el Administrador de sockets retorna.

Los estados se indican en el atributo *state* de la estructura Conexión, estos son:

- Comienzo

  Se dispone a escuchar el socket, en el puerto indicado por la estructura Conexión, aceptando conexiones de un cliente con cualquier dirección IP y puerto. Para realizar esto se llama a *tcp_listen()*. Luego se pasa al estado *Escucha*.

- Escucha

  Primero verifica si hubo un establecimiento de conexión que fue abortada antes de poder verificar que está establecida. La función que permite lograr esto es la tcp_tick(). Si fue abortada se pasa al estado *Recuperación*.

  Luego de está verificación se observa si existe una conexión consolidada, si es así se pasa al estado de Atención de socket correspondiente (por ejemplo *Atención Socket Modbus*) sino se retorna manteniendo el estado. La comprobación del establecimiento de conexión es a través de la función *sock_established()*.

- Atención de sockets

  En este estado se confirma si se mantiene la conexión con *tcp_tick()*, si mantiene se ejecuta el controlador pertinente, si no se pasa al estado

Recuperación. Los controladores de cada tipo de socket se describen más adelante.

- Recuperación

  Se entra en este estado sólo cuando se detecta que el Cliente cierra la conexión, vale aclarar que el Servidor nunca lo hace. Se verifica que queda una solicitud pendiente del Cliente con la función *sock_bytesready()*, invoca al controlador correspondiente. Por último se vuelve al estado *Comienzo*.

En la figura 7.16 se presenta un diagrama de flujo que muestra la secuencia que se realiza para cada conexión.

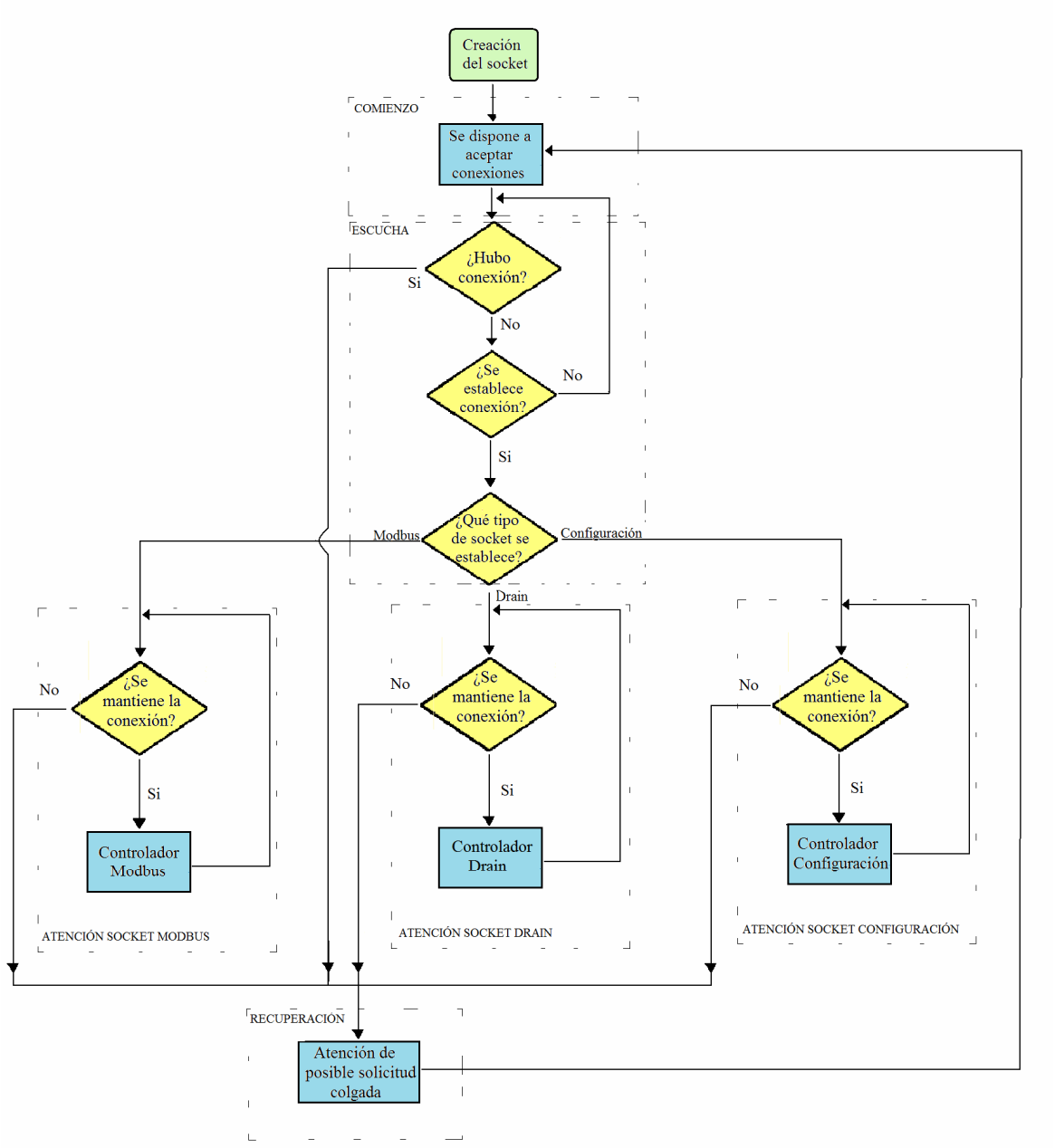


Figura 7.16: Diagrama de flujo en Servidor.

Controladores de sockets

Los controladores de sockets son los que se encargan de resolver las peticiones del Cliente que está conectado estos controladores son: Controlador Modbus, Controlador Drain y Controlador Configuración. Como se vio anteriormente se ejecutan cuando identifica el tipo de Conexión mantiene la conexión abierta.

Todas las lecturas a buffers de entrada TCP son realizadas con la función sock_*fastread()* perteneciente a DCRTCP.LIB y los envíos por el puerto serie se hacen llamando a *serCwrite()* de la librería RS232.LIB.

En común los controladores al inicio implementan un control de flujo, se fijan si está activada la bandera *dato_en_espera*, si esta levantada es porque en el buffer *tx_buffer* se tiene un dato que se requiere enviar (ambos son atributos de Conexión); si es así se envía hacia el Cliente a través de TCP-IP, donde el formato que cumple se explicó en la sección 7.3.2.4.

Posteriormente observan si hay datos a ser leídos en sus respectivos buffers de entrada (*rx_buffer*); si no hay datos, el controlador retorna, de lo contrario se ejecutan distintas acciones según se describe a continuación:

Controlador Modbus:

El Controlador Modbus lee del buffer de entrada la carga de la trama de red TCP-IP, una trama Modbus, es decir sólo 8 bytes, si hay menos los descarta. Luego lo envía a la RFB como se ve en la figura 7.17.

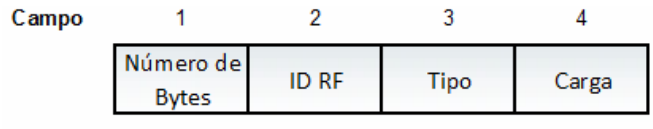


Figura 7.17: Trama hacia RFB.

Donde Número de Bytes vale 10: 1 byte del ID RF, 1 de tipo y 8 bytes de carga, correspondientes a la trama Modbus.

El identificador de red RF (ID RF) se obtiene de la Tabla de Direcciones, a través de la función *buscarParametroTabla()*, pasándole la dirección Modbus.

Tipo vale 0 decimal, indicando que la trama es de datos y no de configuración.

En el caso que no esté presente la dirección Modbus en la tabla, el controlador no realiza el envío.

Controlador Drain:

Este controlador cuando tiene en el buffer TCP de entrada un dato disponible, lee hasta una cantidad máxima de bytes dada por la constante MAX_LARGO_TRAMA_RF, que como se dijo anteriormente es la carga máxima en bytes que permite el protocolo WiDO v1.0.

Después se lo envía a la RFB como establece su protocolo, con ID RF igual 0 (indicando broadcast), y la carga es idéntica a lo que se leyó del socket.

En el caso que queden en el buffer más datos, estos serán enviados en próximas ejecuciones de este controlador.

Controlador Configuración:

El Controlador Configuración identifica las tramas de configuración en el buffer de entrada sabiendo que deben respetar el formato determinado.

De esta manera descarta todas las tramas que no cumplan con dicho formato.

SOF	Número de bytes	ID RF	Tipo	ConfigVals	EOF

Figura 7.18: Trama de configuración.

Luego se invoca a la función auxiliar *Proceso_de_configuracion()*, la que discrimina según el byte *Tipo* la respuesta que se debe accionar. Luego de ejecutarse esta función el controlador retorna.

A continuación se muestran las diferentes configuraciones existentes a las que puede identificar el byte *Tipo*:

- Pedido de estado (10000001 en binario)

  Si es para la base, se llama a la función *Reporte_de_estado()* la cual responde por este socket el estado completo de la red de la manera mostrada en la figura 7.19.

  El campo *tipo* es el mismo que el que fue recepcionado (10000001), luego se indican los dos parámetros del FHSS. Los próximos dos bytes son de la base RF, su ID RF (1) y su Tiempo de vida. Después se enumera a todos los Remotos con su Tiempo de vida, la estimación de saltos que tienen a la base y también se indica la cantidad de dispositivos Modbus que tiene asociado con sus respectivas direcciones Modbus.

  En cambio si va dirigido a un Remoto, se quitan los bytes SOF y EOF, y se los manda a la RFB la misma trama para que esta se lo envíe RF al correspondiente dispositivo.

  Más tarde el bloque Control Serie recibirá la contestación del Remoto, y lo dejará en el buffer de salida de la Conexión Configuración así se envía al Cliente.

- Cambio de Tiempo de vida (10000100 en binario)

  Tanto para la base o para un remoto se envía por serie a la RFB, el mismo paquete sin el SOF y el EOF, y se actualiza en la Tabla de Direcciones el Tiempo de vida correspondiente al nodo RF. Esta actualización se realiza con el procedimiento *SetearParametroTabla()*.

- Disociación explícita de un dispositivo Remoto (11000000 binario)

  Esta modificación al sistema sólo tiene sentido si se realiza a un Remoto, en caso que sea dirigida a la base, se inhibe.

  Se transmite a la RFB el mismo paquete sin el primer y el último byte (SOF y EOF) y se reestablece la Tabla de Direcciones quitando al dispositivo de la misma usando la función *quitarDeTabla()*.

- Configuración de parámetros de FHSS al sistema: Tiempo en canal y Salto de Canal (10011000 binario)

  Esta configuración sólo tiene sentido dirigirla a la base, por lo que si el campo *ID_RF* es distinto de 1, se retorna sin realizar más acciones.

  Esta configuración se efectúa en dos pasos: primero se le envía a la RFB una orden para que deje de mandar la trama de sincronismo, de esta manera todos los Remotos pertenecientes a la red RF quedan fuera de la red.

  Luego se deja pasar un tiempo determinado, usando el Timer B del Rabbit, este tiempo es llamado TIEMPO_ESPERA_CONF_RFB que se indica antes de la compilación del programa.

  Cuando expira el tiempo, en la rutina de atención de la interrupción, se envía por serie los nuevos parámetros del FHSS culminando así la configuración. La trama enviada se muestra en la figura 7.20.

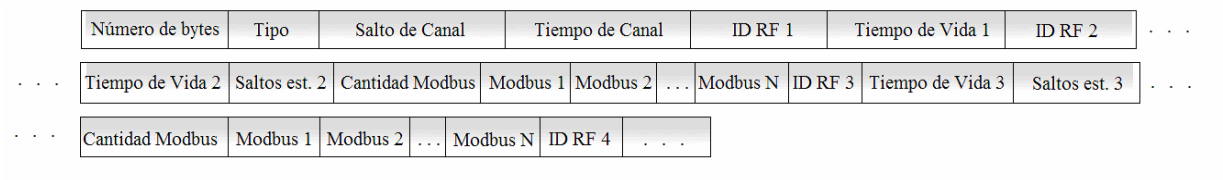


Figura 7.19: Trama de reporte de estado hacia Cliente.

Número de bytes	ID RF	Tipo	ConfigVals	
4	1	152	nuevo Salto Canal	nuevo Tiempo Canal

Figura 7.20.: Trama que configura FHSS.

Bloque Control Serie

El Bloque Serie es el responsable de la recepción de los paquetes provenientes de la RFB.

Para esto escucha el puerto serie, lo primero que espera es el primer byte que indica la cantidad de bytes que luego vendrán. Se recuerda el formato de trama que espera el Rabbit es la que se muestra en la figura 7.21.

Número de Bytes	Número de paqute	Saltos	Dirección origen	Dirección destino	Dirección intermedia	Tipo	Carga

Figura 7.21: Trama genérica de RFB a Rabbit.

Lee el buffer con la función *serCgetc()*, mientras no llega este primer byte el bloque de Control Serie retorna al flujo principal del programa. Una vez que llega el primer byte se espera el resto de la trama hasta un tiempo máximo, para esto se inicia el Timer B con un tiempo denominado TIEMPO_ESPERA_RECEPCION_SERIE, y luego se retorna.

En próximas ejecuciones se fija la cantidad de bytes que hay en el buffer de entrada del puerto C con la función *serCrdUsed()* también de la librería RS-232.LIB.

En el caso que expire el tiempo, se limpia el buffer de recepción con la función *serCrdFlush()*, a esperar nuevamente el primer byte; en caso contrario, cuando la cantidad de bytes en el buffer sea igual o mayor a la esperada, se leen, pasando a una siguiente etapa que interpreta la trama.

Posteriormente con el byte *Tipo* se distinguen cuatro casos:

- Datos pertenecientes a la Conexión Modbus o Drain (Tipo = 0).

  Primero invoca al procedimiento *analizarTramaDeRed()*, que tiene en cuenta la primer parte de la trama recibida, en especial los bytes *Saltos*, *Dirección Origen*, y *Dirección Intermedia*. Con estos campos realiza la estimación de saltos; en primer lugar se reconoce que el Remoto con dirección: *Dirección Intermedia*, está a sólo un salto de la Base, ya que fue el que alcanzó a la Base. Además obviamente es determinado el remitente del mensaje (*Dirección Origen*) y con que Tiempo de vida llegó el mensaje (*Saltos*), entonces restando el Tiempo de vida correspondiente a este Remoto menos *Saltos*, se obtiene la estimación de la cantidad de saltos que hay entre ambos nodos de la red. De esta forma se modifica para ambos Remotos los campos *Estimación de Saltos hacia la Base* de la Tabla de Direcciones, siendo la única función cambia estos valores.

  Luego se prepara la trama y se almacena en los buffers de transmisión de la Conexiones Modbus y Drain y se levanta la bandera *dato_a_*enviar a 1 de estas conexiones, para ser enviadas en las próximas ejecuciones de los controladores Modbus y Drain respectivamente.

- Respuesta de pedido de estado por parte de un Remoto (Tipo = 10000001 binario).

  En este caso también se interpreta la trama de capa de red del protocolo RF con el procedimiento *analizarTramaDeRed()*.

  Se empaqueta y después se carga en el buffer de transmisión de la Conexión Configuración y se levanta su bandera.

- Pedido de “logeo” a la red por parte de un Remoto (10100000 binario).

  Este caso se da cuando un Remoto se integra a la red y pide su inicialización.

  Para registrarlo se llama a la función *obtenerID()*, esta busca el identificador RF (ID RF) y una vez que lo encuentra guarda los datos asociados al Remoto (dispositivos Modbus y sus direcciones) en la Tabla de Direcciones.

  Respondiendo por serie hacia la RFB el siguiente paquete:

Número de bytes	ID RF	Tipo	Carga	
4	ID otorgado	160	ID otorgado	Número de Saltos

Figura 7.22: Respuesta a un pedido de “logeo”.

- Actualización de que un Remoto no está más en la red (11000000 binario).

  Cuando la RFB detecta la caída de un Remoto, manda una trama con el valor *Tipo* que indica “deslogueo”; en este caso el Rabbit quita de la Tabla de Direcciones al Remoto indicado en el paquete recibido. Esta actualización de la tabla se realiza con el procedimiento *quitarDeTabla()*.

Resumiendo el ítem Descripción de Bloques, se puede mencionar que el software está dividido esencialmente en dos partes, una que se encarga de la interfaz Ethernet TCP – IP hacia el Cliente y la otra que se encarga de la recepción serie RS-232 que se comunica con la RFB, el otro componente de la Base RF. Además se comunica entre ellos mediante un control de flujo que permite a cada bloque usar funciones del otro, y llevan entre los dos el control de la red inalámbrica compartiendo la Tabla de Direcciones.

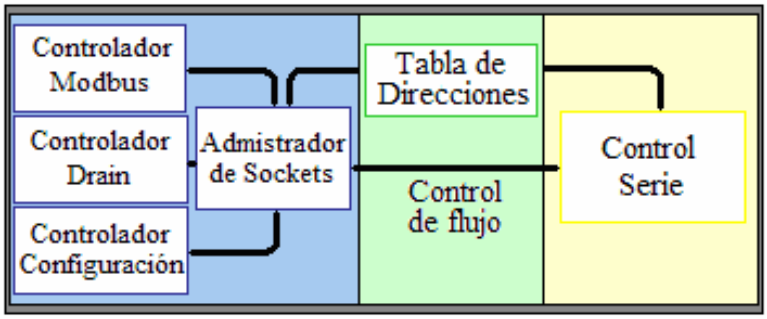


Figura 7.23: Diagrama de Bloques de software en Rabbit.

7.3.3.5 Definición de constantes

A continuación se muestra la definición de todas las constantes del programa y a su lado se comenta una breve descripción:

```
// S E R V I D O R

// ESTADOS DE CONEXIONES.
#define COMIENZO                        0
#define ESCUCHA                         1
#define ATENCION_SOCKET_MODBUS          2
#define ATENCION_SOCKET_DRAIN           3
#define ATENCION_SOCKET_CONF            4

// NÚMEROS DE PUERTOS
#define NUMERO_DE_PUERTO_MODBUS  4050  // Número de puerto para Conexión Modbus.
#define NUMERO_DE_PUERTO_CONF    4030  // Número de puerto para Conexión Configuración.
#define NUMERO_DE_PUERTO_DRAIN   4040  // Número de puerto para Conexión Drain.

#define SOCK_BUF_SIZE 4096              // Suma de los tamaños de los buffers de
                                        // entrada y salida de los sockets TCP.

// VALORES ESPERADOS DEL CAMPO TIPO
#define BYTE_PEDIDO_DE_ESTADO  129   //
#define BYTE_DATOS               0   // La carga son datos.
#define BYTE_LOGIN             160   // Un Remoto se quiere logear.
#define BYTE_DESLOGIN          192   // Un Remoto queda fuera de la red.


//-------------------------------------------------------------------------------


// IDENTIFICADOR RF
#define ID_BROADCAST 0  // Inidica el mensaje RF es destinado a todos los nodos.
#define IDrf_base    1  // Identificador de la Base.

// PARÁMETROS DE RED RF
#define DEF_NUMERO_DE_SALTOS     15  // Tiempo de vida (número de saltos).
#define DEF_FH_TIEMPO_DE_CANAL   20  // Tiempo de permanencia en un canal (ms).
#define DEF_FH_SALTO_DE_CANAL     7  // Salto de canal (canales).
#define MAX_DISPOSITIVOS_MODBUS  64  // Cantidad de dispositivos Modbus admitidos.
#define MAX_LARGO_TRAMA_RF      128  // Largo máximo de la carga en trama RF (bytes).

//-------------------------------------------------------------------------------


// PUERTO SERIE C
#define CINBUFSIZE   255      // Tamaño de buufer de entrada del puerto serie C.
#define COUTBUFSIZE  255      // Tamaño de buufer de salida del puerto serie C.
#define BAUD_RATE_SERIE 9600  // Tasa de datos del puerto serie C.

// TIEMPOS DE ESPERA
#define TIEMPO_ESPERA_RECEPCION_SERIE 10000
#define TIEMPO_ESPERA_DESLOGEO 10000
#define TIEMPO_ESPERA_CONF_RFB 100  //en milisegundos.

//DIRECIONAR TABLA DE DIRECCIONES.
#define DIRECCION_RF 0
#define DIRECCION_MODBUS 1
#define NUMERO_DE_SALTOS 2
#define SALTOS_HACIA_BASE 3
```

Figura 7.24: Definición de constantes.

7.3.4 Depuración y validación del software

De modo de validar y verificar la implementación del programa fue necesario contar con herramientas auxiliares que garanticen el correcto funcionamiento del programa.

Para esto el Dynamic C otorga la posibilidad de añadir impresiones en consola. Estas impresiones se ubicaron en lugares relevantes del código que permiten conocer el curso del programa.

De manera que sea opcional ejecutar las impresiones se define la propiedad *COMENTARIOS*, de manera que si se comenta en el código la línea: *#define COMENTARIOS* se deshabilitan.

Además para realizar simulaciones se creó un función auxiliar *simuloTablaDirecciones()*, la cual inicializa la Tabla de Direcciones agregando cuatro Remotos como si estuvieran ya integrados a la red RF, con sus respectivas propiedades. En conjunto se utilizó el Serialito, una aplicación que permite establecer una comunicación serie RS-232 a través del puerto serie de una computadora.

Este programa cuenta con dos consolas, una para ingresar bytes codificados en caracteres con estándar ASCII que se envían por serie y otra para la recepción de bytes. En su configuración se debe establecer la misma tasa de datos que tiene cargado el Rabbit.

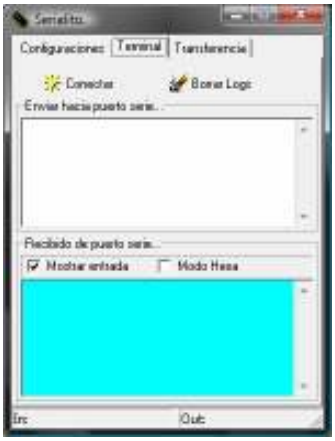


Figura 7.25: Terminal de Serialito.

Figura 7.26: Configuraciones del Serialito.

Usando estás herramientas se depuraron todas la funciones implicadas en el bloque Control Serie, y la forma de verificarla fue gracias a la función *Reporte_de_estado()*, la cual muestra de forma ordenada la Tabla de Direcciones y los valores de los parámetros de FHSS.

Por otro lado para depurar la implementación del Administrador de Sockets, se usó una función auxiliar que provee TCP.LIB llamada *gethostid()* que devuelve la dirección IP, que tomó el Rabbit en la red al usar DHCP para conocerla.

Además se creó una aplicación Cliente que corre sobre Windows, desarrollada en el entorno del Visual Studio con en el programa C#. Esta aplicación tiene como nombre: Interfaz de Red Domótica.

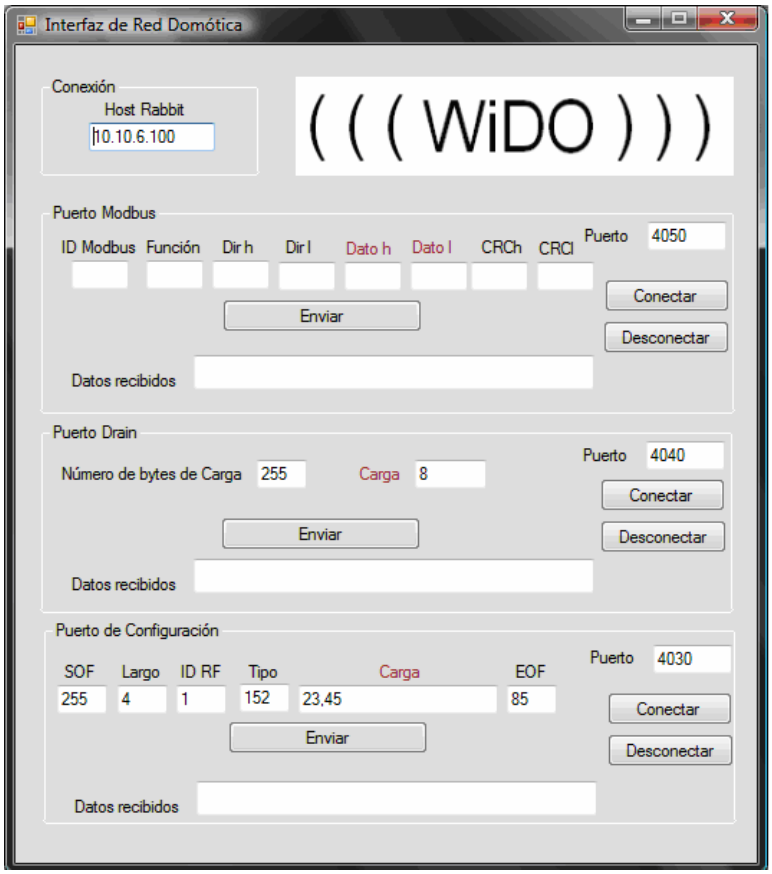


Figura 7.27: Interfaz de Red Domótica.

Esta aplicación permite tener un Cliente del sistema.

A la vez puede abrir las tres tipos de conexiones. Tiene una interfaz gráfica que permite al usuario pasarle la dirección IP del Rabbit y los números de puertos que corresponden a cada tipo de conexión.

La ventana se divide en tres regiones, una para cada conexión: Conexión Modbus, Conexión Drain y Conexión Configuración. Antes de enviar cualquier trama sobre TCP-IP el Cliente debe conectarse con el Rabbit con el botón *Conectar* del tipo de Conexión que se quiera lograr; luego de esto, se permite enviar la trama definida por las entradas. Para el caso Modbus se leen los correspondientes campos que posee este protocolo: Identificador Modbus, la función que pude ser de lectura o escritura, los datos que son dos bytes y por último los dos bytes de CRC.

En el caso de la Conexión Drain la interfaz ofrece solamente enviar la cantidad de bytes deseados pero todos repetidos, es decir si *Número de Bytes de Carga* vale 256, y *Carga* vale 8, al apretar el botón *Enviar* se transmitirá 256 bytes con el valor 8.

Por el lado de Configuración permite indicar el valor byte a byte de los campos de este tipo de tramas, *SOF, Largo* (Número de bytes), *ID RF, Tipo, Carga y EOF*. En el caso que se necesite enviar más de un byte en *Carga*, estos deben estar separados por una “,”.

En paralelo, una vez establecida una conexión, se activa un *Timer* que interrumpe periódicamente. En cada una de las interrupciones, se fija si tiene datos a leer en cualquiera en los buffers de entrada de las conexiones activas. Una vez leídas lo refresca en pantalla en *Datos Recibidos* de la correspondiente conexión.

Referencias

[4.1] 869/915 MHz vs. 2.4GHz . Elpro Technology Technical Paper.

www.elprotech.com

[4.2] Performance Comparison Of Frequency Hopping And Direct Sequence Spread Spectrum Systems In The 2.4 GHz Range. Zoran Spasojevic, John Burns, Aegis Systems Ltd.

[4.3] A Comparison of Frequency Hopping and Direct Sequence Spread Spectrum

Modulation for IEEE 802.11 Applications at 2.4 GHz. Carl Andren, Harris Semiconductor.Palm Bay, Florida.

[4.4] Spread Spectrum - Aplications. J. Meel, De Nayer Instituut, Bélgica.
www.denyer.be

[4.5] Spread Spectrum - Introduction. J. Meel, De Nayer Instituut, Bélgica.
www.denyer.be

[4.6] Making 802.11GTransmitter Measurements. Application Note 1380-4, Agilent.

[4.7] Estudio, Modelamiento y Simulación de Sistemas de Espectro Ensanchado

Secuencia Directa y Salto De Frecuencia. Hernán Córdova, Patricia Chávez

Departamento de Investigación de Sistemas de Telecomunicaciones.

Escuela Superior Politécnica del Litoral, Guayaquil, Ecuador **Simulaciones**

[4.8] Direct Sequence vs. Frequency Hopping. WANE, Wireless Networking.

[4.9] IEEE 802.15.4 (2003)

[7.1] ATR2406 Smart RF AFHSS Firmware. Atmel Application Note**.**

www.atmel.com

[7.2] AVR068: STK500 Communication Protocol.

www.atmel.com

[7.3] AVR Dragon.

http://support.atmel.no/knowledgebase/avrstudiohelp/mergedProjects/AVRDragon/AVRDragon.htm

[7.4] Code Vision Manual.

instruct1.cit.cornell.edu/courses/ee476/codevisionC/cvavrman.pdf

[7.5] AVR Studio User Manual.

www.atmel.com

[7.6] Low-IF 2.4 GHz Tranceiver: ATR 2406. Hoja de Datos.

www.atmel.com

[7.7] Connecting ATR 2406 SMART RF to a General Purpose Microcontroller. Atmel Application Note.

www.**atmel**.com

[7.8] SPI and JTAG In-System Programming (ISP) guidelines for the Atmel ATmega AVR FLASH Microcontroller Family. Equinox Application Note.

http://www.farnell.com/datasheets/97654.pdf

[7.9] AVR060: JTAG ICE Communication Protocol.

[7.10] Protocolo Modbus. Universidad Politécnica de Cartagena, España.

http://www.dte.upct.es/personal/manuel.jimenez/docencia/GD6_Comunic_Ind/pdfs/Tema%207.pdf –

ANEXOS

Anexo A

Análisis del proyecto

A.1 Introducción

En este anexo se presentan los objetivos específicos y los criterios de éxito propuestos y aspectos preeliminares en la planificación, como son: supuestos, actores, análisis de riesgos y gestión de tiempos. Posteriormente se cotejan con los resultados obtenidos.

Luego se plantean trabajos a futuro.

A.2 Planificación inicial

Los objetivos específicos a alcanzar son:

1. Desarrollo de un protocolo RF completo.
2. Integrar los sistemas desarrollados para lograr comunicación punta a punta entre el PC y un actuador Modbus.
3. Implementar comunicación Modbus sobre TCP/IP.
4. Implementar comunicación Modbus / RS-485.
5. Soportar Plug & Play.
6. Resolver problemas de cobertura del sistema para distintos casos, dotándolo de flexibilidad.
7. Alimentar los dispositivos a diseñar a través de la red eléctrica de 220 V.

Criterios de éxito

1. Poder brindar cobertura en toda la planta de un domicilio u oficina, utilizando la cantidad de dispositivos (inclusive posibles repetidores) que sean necesarios.
2. La comunicación inalámbrica en línea vista de los dispositivos debe cubrir una distancia mínima de 10 metros.
3. El sistema debe funcionar debajo el protocolo Modbus.
4. El sistema debe reconocer todos los dispositivos disponibles, y soportar la sustitución y el agregado de un dispositivo sin necesidad de re configuración.
5. El sistema debe seguir en su correcto funcionamiento para el caso que se produzca la rotura de alguno de sus dispositivos remotos (excepto la base).

Supuestos

1. Los integrantes del proyecto y el tutor no abandonarán hasta ser finalizado el proyecto.
2. La empresa Domótica SRL permanecerá con el apoyo tanto técnico como económico hasta ser finalizado el proyecto.
3. No existen restricciones en la cantidad de dispositivos en la implementación del sistema, como por ejemplo repetidores.
4. Se puede acceder al hardware necesario para realizar el proyecto.

Actores

1. Integrantes del grupo de proyecto.
2. Tutor del proyecto: Juan Pechiar.
3. Empresa Domótica SRL.
4. Proveedor de tecnología a utilizar: Atmel.
5. Otros posibles proveedores de tecnología.
6. Facultad de Ingeniería.

Restricciones

1. Monetaria: dependencia económica de la empresa Domótica SRL.
2. Temporal: la duración del proyecto está limitada por la fecha máxima de entrega.
3. Existe una limitación tecnológica dada por los materiales a utilizar suministrados.
4. Existen dificultades para conseguir en tiempo y forma el hardware a utilizar.

Análisis de Riesgos.

A continuación se presentan los riesgos del proyecto, los cuales no incluyen aquellas posibles dificultades técnicas que ya estén contempladas en la asignación de horas a las tareas.

1. Daño en algunos de los componentes de hardware más importantes
2. Los transceivers utilizados no logren la cobertura o velocidad de transmisión requeridas para el proyecto.
3. Demoras en la fabricación de los PCB en los casos en que se encarguen a terceros.
4. La empresa Domótica no esté dispuesta en determinado momento a financiar el proyecto.
5. Algún integrante del proyecto deba suspender por largo tiempo su participación en el mismo debido a enfermedad o viaje.
6. El tutor abandone por alguna causa la tutoría del proyecto.
7. Huelga en la Facultad.
8. No poder cumplir con las horas estipuladas debido a situaciones laborales o académicas imprevistas.
9. Demoras por dificultades de integración de todo el sistema.

Cuantificación de Riesgos

La siguiente tabla muestra un análisis de los riesgos antes mencionados, estableciendo la probabilidad e impacto de los mismos. Se deduce luego la prioridad con la cual debe ser atendido cada uno de los riesgos; siendo aquellos riesgos de prioridad alta los que deberán tener una respuesta para minimizar o eliminar su impacto.

Riesgo	Probabilidad	Impacto	Prioridad
1	Moderada	Alto	Alta
2	Moderada	Extremo	Alta
3	Moderada	Alto	Alta
4	Alta	Alta	Alta
5	Baja	Alta	Media
6	Baja	Medio	Baja
7	Baja	Bajo	Baja
8	Baja	Alto	Media
9	Moderada	Alta	Alta

Tabla A.1: Riesgos.

Gestión de tiempos

Una herramienta útil en la planificación del proyecto fue la realización de un Diagrama de Gantt.

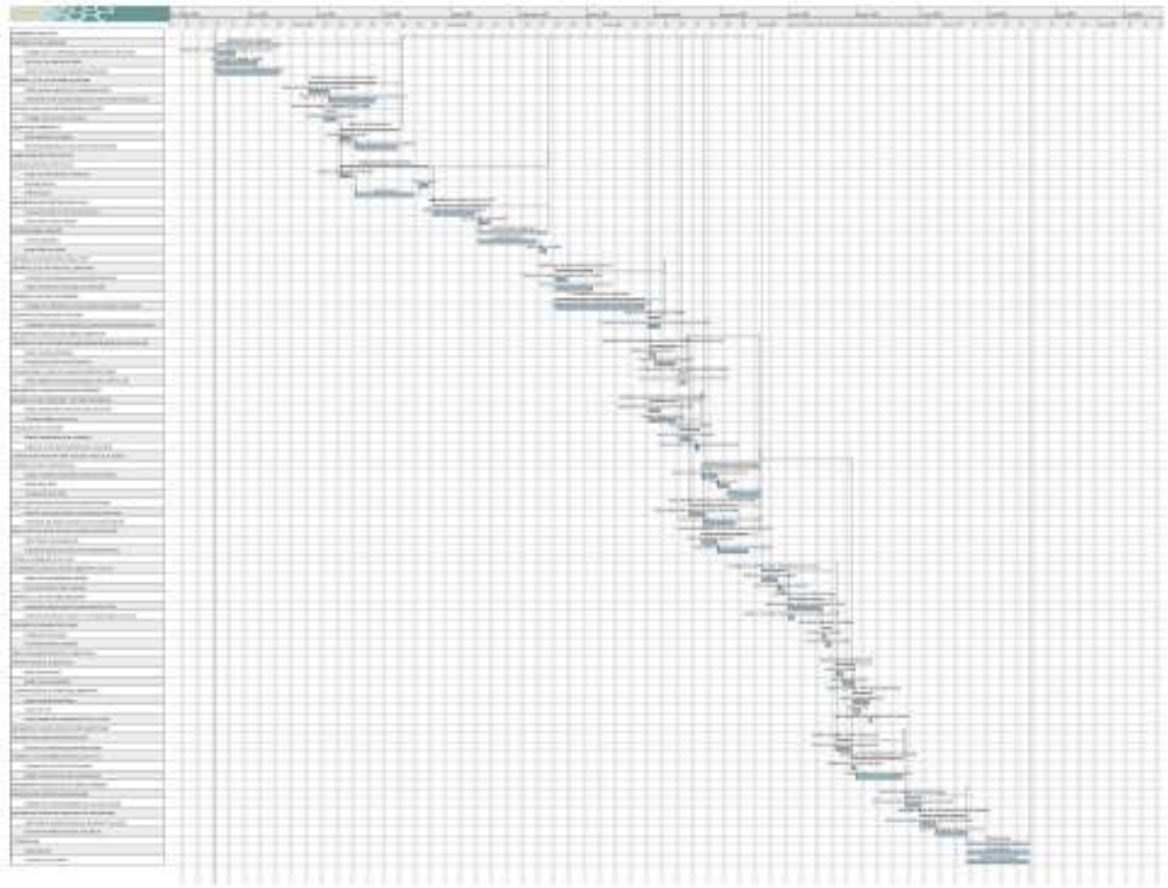


Figura A.1: Diagrama de Gantt, planificación inicial.

A.3 Resultados obtenidos

A pesar de que se presentaron dificultades en cumplir con los objetivos definidos en la primera etapa del proyecto, se logró concretar una gran parte de ellos. Por esto consideramos que el resultado es satisfactorio.

Se encontraron diversos inconvenientes durante el transcurso del desarrollo de la placa RFB, pero gracias la buena gestión realizada se logró llegar a un producto de gran calidad, de diseño compacto y que integra las últimas tecnologías en hardware RF. Este dispositivo cumple con los requerimientos de alcance exigidos.

Por otra parte se desarrolló todo el software que utiliza el sistema, incluyendo varias de las herramientas de depuración que se utilizaron a lo largo del proyecto.
Se logró especificar por completo un protocolo cuyas características cumplen con las normativas internacionales de la banda libre ISM. Este protocolo respeta todas las necesidades que el cliente exigió, brindando al sistema de características tales como la cobertura, velocidades de transmisión de datos, soporte a los actuadores Modbus, robustez ante cortes del suministro eléctrico y facilidad para agregar dispositivos al sistema en funcionamiento (*Plug and Play*). Además se implementaron la gran mayoría de las funcionalidades de esta especificación permitiendo al sistema funcionar satisfactoriamente en un entorno controlado.

Se logró implementar una interfaz a nuestro sistema que permite al cliente administrar y gestionar la red de forma remota conectándose a través de una red Ethernet. Dotando a nuestro sistema de flexibilidad para ser utilizado en otras aplicaciones.

También se resolvió el problema de la alimentación de los dispositivos, al implementar una fuente de voltaje que se energiza desde la red eléctrica local, cuyas dimensiones físicas son ideales para el caso de uso.

Se provee al cliente de una aplicación de software con interfaz gráfica amigable, específicamente desarrollada para el control del sistema a través de una red Ethernet. Este software puede ser ejecutado por cualquier PC de hoy en día.

En contrapartida hubo algunos objetivos que no fueron logrados en su totalidad. Queda pendiente la caracterización del sistema completo, es decir que falta un análisis del rendimiento, tasa de datos reales, cantidad de dispositivos soportados y *performance* en entornos ruidosos.

Existen algunas funciones especificadas en el protocolo para las RFB que no fueron probadas en el sistema (ver sección 7.2).

A.4 Causas de los desvíos en la planificación

A lo largo del proyecto se fueron haciendo modificaciones a la planificación inicial. Muchas de estas modificaciones fueron causadas por contratiempos los cuales estuvieron contemplados en el análisis de los riesgos. También se generaron modificaciones por diferencias entre los tiempos asignados a ciertas tareas y los tiempos reales. En lo que sigue se analizará en detalle las causas de los contratiempos que generaron modificaciones en la planificación, así como las diferencias entre los tiempos estimados y los reales.

En principio nos apegamos a la planificación inicial. Se logró transmitir línea vista utilizando el kit de desarrollo, se desarrolló un software de prueba y se comenzó a implementar las funciones básicas del protocolo, logrando el sincronismo, midiendo alcance y tasa de datos entre dos dispositivos. En paralelo se definió y se comenzó el desarrollo de un primer prototipo de RFB, tal cual estaba estipulado en la planificación.

En este punto se estaba dentro de los tiempos estipulados salvo por una pequeña demora en la construcción del prototipo. Luego surgió el primer gran contratiempo que cambiaría por completo la planificación del proyecto, el cliente Domótica S.R.L cesó sus operaciones dejando de financiar el proyecto. Este hecho determinó que no se pudiera continuar con el desarrollo del prototipo de prueba y se tuviera que comenzar a desarrollar el dispositivo definitivo. Se tuvo que abandonar la idea de poder hacer las pruebas de capa de red y verificación del protocolo ya que al no poseer más de dos dispositivos se hacía imposible realizarlas.

A partir de este punto se realizó la especificación del protocolo con más cuidado, atendiendo a los detalles que pensábamos podrían surgir una vez que se implementara en una red real de dispositivos. A consecuencia de esto surgió la especificación del protocolo. En cuanto al desarrollo del dispositivo definitivo se tuvo que replantear la forma en que lo estábamos desarrollando ya que en un principio se pensaba contar con un prototipo que nos diera un panorama sobre los problemas que podrían surgir. Además el hecho de no contar con presupuesto hizo que se tuviera especial cuidado en el desarrollo de la RFB ya que sólo se tendría una oportunidad para fabricarla. También se tuvo que gestionar cuidadosamente la compra de los componentes que se utilizaron y se hizo una gran gestión con los

fabricantes de impresos, logrando que se nos subvencionaran la construcción del dispositivo.

Este último hecho impactó fuertemente al proyecto, generando un atraso general en las tareas.
En el análisis de riesgos estaba tomada en cuenta la posibilidad de que en cierto momento el cliente no pudiera financiar el proyecto y el impacto que se le asignó fue de nivel alto.

Luego del acontecimiento antes mencionado se continuó con el proyecto pero la planificación fue modificada (ver figura A.1). Se siguió avanzando en otros aspectos del proyecto mientras se gestionaba la construcción del hardware. Este punto llevó más tiempo del que se planificó. Se tuvieron demoras en las entregas ya que todos los insumos que se necesitaron no se podían conseguir en el mercado local teniendo que recurrir al mercado internacional, más precisamente al mercado de Estados Unidos y Argentina. Los tiempos en que se estuvo a la espera de la llegada de componentes y demás, se utilizó para desarrollar otras partes del proyecto como ser la interfaz del sistema con la red Ethernet.

En este momento se tuvo que recurrir a la prórroga ya que el atraso que se tenía hacía imposible la culminación del proyecto en la fecha inicialmente estipulada. En el momento que la prórroga fue concedida se mandaron a construir los dispositivos definitivos luego de una larga gestión en la cual tuvimos que recurrir al uso del correo para que se nos enviaran los impresos, ya que aunque el fabricante nos dio un descuento del 50 % el costo era demasiado elevado para nuestro reducido presupuesto.

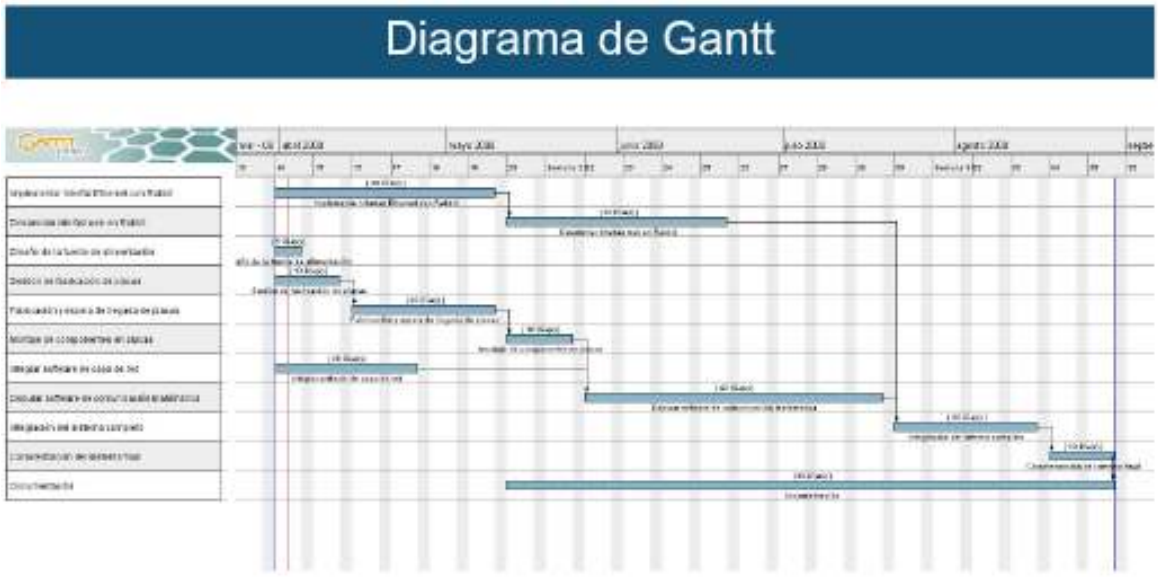


Figura A.2: Diagrama de Gantt, planificación pos prórroga.

En este punto se produce el segundo gran contratiempo, la llegada de los impresos se retraso dos meses a causa de un error en el correo, el envío fue retenido por el correo y no se nos notificó de este hecho. Luego de realizar el rastreo correspondiente, se pudo dar con el mismo. A la llegada de los impresos se tenía un atraso de dos meses el cual nos dejaba muy ajustados para llegar a la fecha estipulada en la planificación pos prórroga. Luego de la llegada de los impresos, se tuvo que recurrir a una empresa privada para el montaje de los componentes. Esta empresa opera en Buenos Aires (Armados Electrónicos) y para evitar más demoras se viajó a Buenos Aires para gestionar dicho montaje.
Este último contratiempo estuvo tenido en cuenta en los riesgos, las demoras de la construcción por parte de terceros estaba catalogada con un impacto de nivel alto al igual que la falta de presupuesto.

A pesar de estos contratiempos se trabajo en forma organizada, se respeto la carga horaria que se había planificado e incluso en algunos períodos se intensificó el trabajo aumentando las horas estipuladas de cada uno de los integrantes. De esta forma se lograron muchos objetivos incluyendo algunos que no se exigían en el proyecto.

A.5 Conclusiones y trabajo a futuro

Cabe destacar como conclusión más importante de este trabajo, la experiencia ganada en distintas y variadas áreas durante el desarrollo del mismo. Principalmente debido a que consideramos que se trató de un proyecto integrador, que abarcó muchos aspectos, desde el diseño y construcción de casi la totalidad del hardware, hasta la elaboración de todo el software requerido en diferentes partes del sistema (para lo cual se utilizaron variedad de lenguajes de programación) y la especificación del protocolo de comunicaciones.

Si bien, a causa de diferentes inconvenientes sucedidos a lo largo del desarrollo, los cuales fueron explicados en el punto anterior, no se logró cumplir completamente con el objetivo de implementar la red de comunicaciones estipulada, consideramos que se alcanzó la mayor parte y más importante de dicho objetivo. Es decir, se desarrollaron completa y exitosamente los dispositivos base y remotos (logrando establecer la comunicación inalámbrica de la forma establecida en un principio), y se elaboró en su totalidad el software necesario para dotarlos del comportamiento y funcionalidades requeridas para la red diseñada. Por otro lado, también se desarrolló la interfaz de las comunicaciones para los dispositivos, tanto hacia la PC cliente (Ethernet), como hacia los dispositivos finales de la red (RS485) y se especificó en su totalidad el protocolo de comunicaciones inalámbrica denominado WiDO v1.0.

Como pendiente para alcanzar definitivamente el objetivo propuesto, restaría una etapa de ensayos y pruebas de la *performance* de la red logrando validar del protocolo, la cual no nos fue posible realizar, por las demoras ya expuestas, que llevaron al desvío de la planificación inicial. Durante esta etapa se realizarían distintas pruebas en funcionamiento de la red, de todas las funcionalidades que se pensaron para la misma, obteniendo a partir de estas pruebas una variedad de datos y conclusiones que llevarían a modificaciones en el software de los dispositivos y en el protocolo mismo. Finalmente a partir de la iteración entre la teoría y la práctica se llegaría a la versión definitiva del protocolo y a la completa caracterización de la red.

Pensamos como un posible trabajo a futuro lo mencionado en el párrafo anterior, mediante el cual, a partir de una primera versión del protocolo de comunicaciones y contando con los dispositivos diseñados, se lograría implementar una red inalámbrica de dispositivos optimizada para este tipo de aplicaciones. Esto sería posible sin mayores inconvenientes gracias a que todo el diseño del sistema fue pensado de forma tal de dotar al mismo de cierta flexibilidad para realizar cambios, lo cual lo hace fácilmente extensible.

Finalmente, creemos pertinente destacar que un aspecto muy importante a tener en cuenta de este trabajo, es que a partir de tan sólo dos integrados (microcontrolador y transceiver) se desarrolló un sistema de comunicaciones inalámbrico completo, contando como consecuencia de esto con todo el *"Know how"* involucrado en el producto final, lo cual es una ventaja desde cualquier punto de vista.

ANEXO B

Obtención de archivos Gerber

Un archivo Gerber es un estándar de formato de archivo utilizado por los fabricantes de impresos, que contiene la información necesaria para que las máquinas controladas por ordenador puedan dibujar los patrones exactos de las placas. Estos patrones se utilizan normalmente para montar y conectar eléctricamente los componentes electrónicos. Los patrones incluyen la información de los diámetros de los agujeros, los planos de tierra, la ubicación de la mascara antisoldante, los anchos de pista y los contornos de corte.

Para extraer estos archivos Gerber a partir de un archivo .brd de EAGLE se deben seguir los siguientes pasos:

Abra el archivo .brd correspondiente. Luego pulse el icono Cam que se encuentra en la barra de tareas de EAGLE. Deberá visualizar una ventana igual a la que se muestra debajo.

A continuación abra un nuevo trabajo (open→ job) y seleccione el archivo excellon.cam Luego pulse Process Job, en este momento se generan un archivo .drd que será utilizado mas adelante por la misma herramienta.

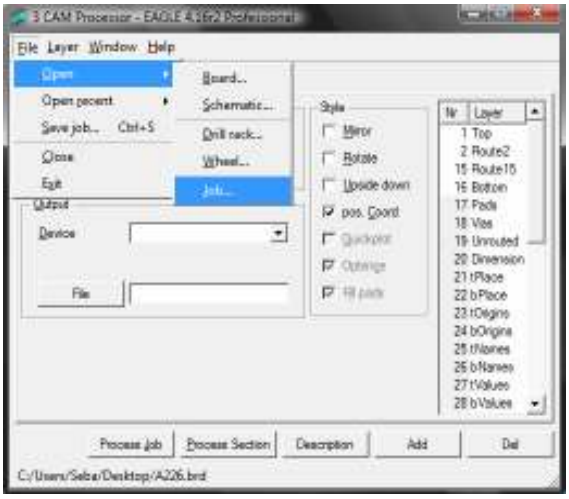


Figura B.1 CAM Processor.

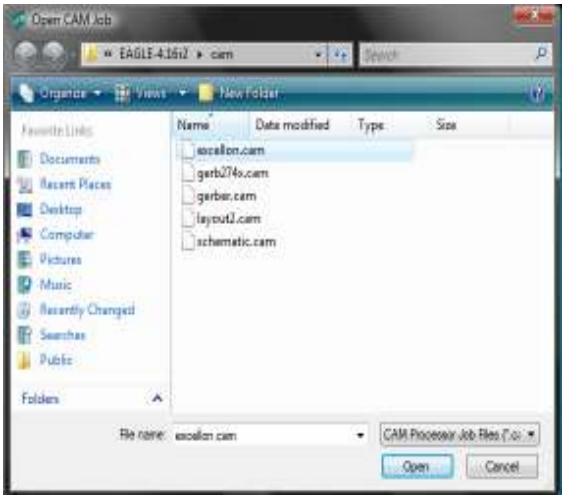


Figura B.2 Abrir CAM Job.

Una vez generado el archivo anteriormente mencionado, se procede a abrir un nuevo trabajo (open→ job) pero esta vez seleccionamos el archivo gerb274x.cam. En esta instancia estaremos generando los archivos:

.cmp	Pistas que se encuentran en la cara superior. (*Component Side*)
.sol	Pistas en la cara inferior. (*Solder Side*)
.plc	Serigrafía de componentes. (*Silk Screen CMP*)
.stc	Máscara antisoldante de la cara superior. (*Solder stop mask CMP*)
.sts	Máscara antisoldante de la cara inferior. (*Solder stop mask SOL*)

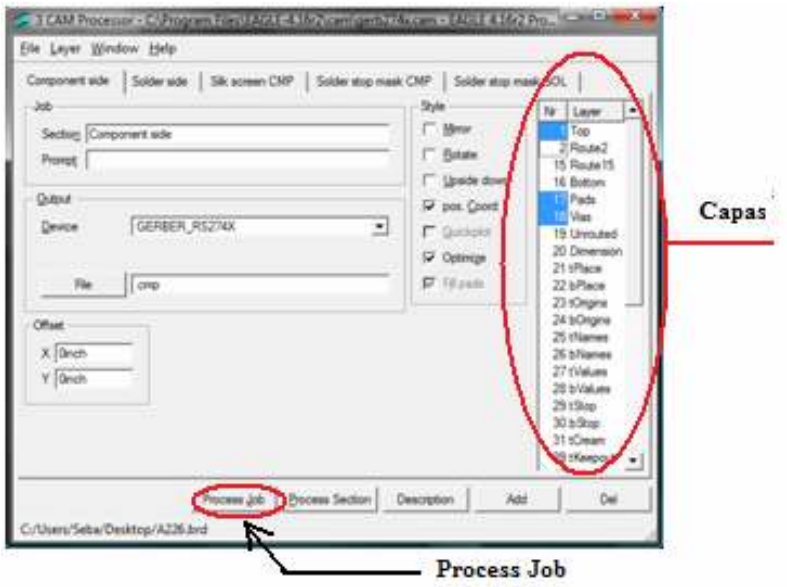


Figura B.3 Capas en CAM Processor.

Cada pestaña de esta ventana corresponde a la generación de cada uno de los archivos, se debe seleccionar para cada archivo las capas correspondientes. El programa utilizara las capas seleccionadas para generar el archivo.

Por ejemplo: Para generar el archivo .cmp seleccione todas las capas que quiere generar del lado superior incluyendo las capas que contienen los agujeros, vías y pads. Para este caso quedarían seleccionadas las capas Top, Pads y Vias. Notar que en la misma ventana hay otras opciones como rotar o espejar que se pueden usar para casos especiales. Ver figura B.2.

Una vez que se realizó la selección de capas para cada archivo y se seleccionaron las opciones correspondientes se procede a presionar *Process Job* una vez más. En este momento el programa generara los archivos y los guardara en la misma carpeta donde se encuentra el archivo .brd del que partimos.

Con estos archivos el fabricante tendrá toda la información necesaria para poder construir el impreso.

ANEXO C

Programador del Rabbit

Para programar el RCM3700 fue necesario construir un programador. Para esto se utilizó el ¨*método de la planchita*¨. El diseño del programador es provisto por el fabricante del RCM3700 y su esquemático eléctrico se muestra en la figura C.1. El diseño de placa se hizo utilizando el software *TraxMaker*, en la figura C.5 se puede observar el mismo.

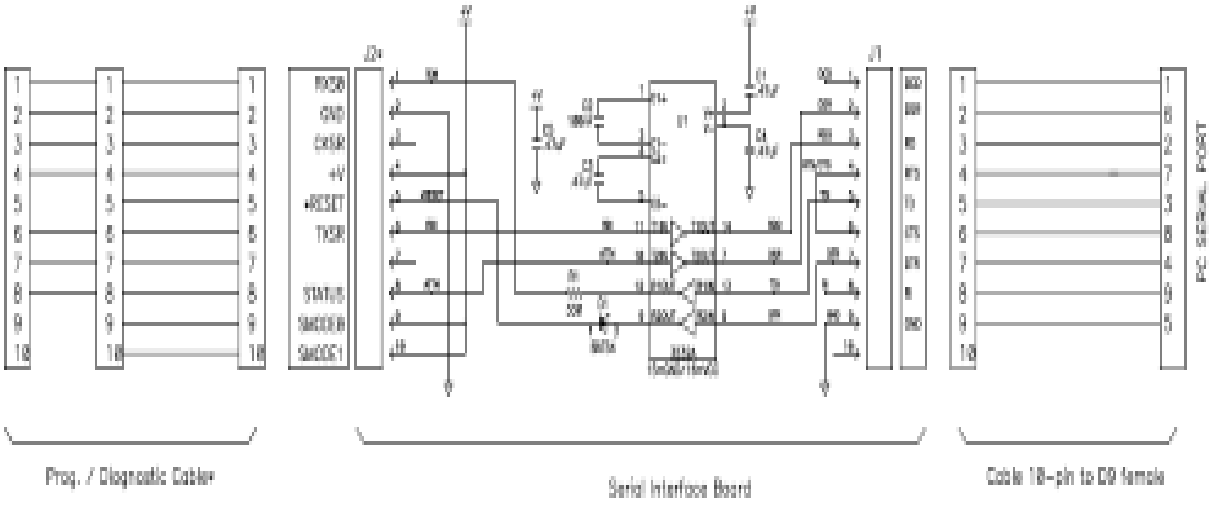


Figura C.1: Programador RCM3700 esquemático eléctrico.

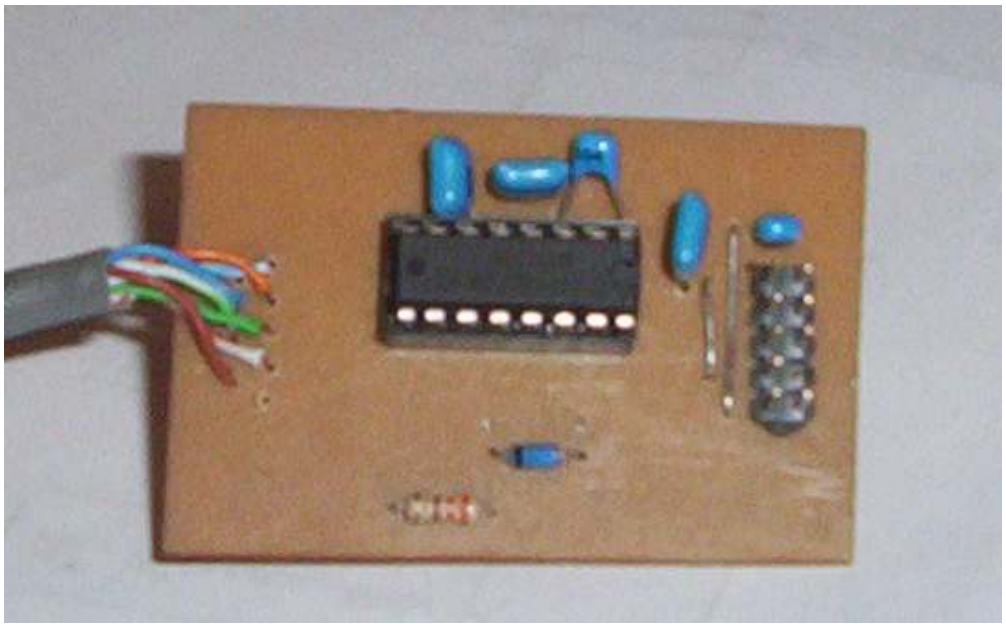


Figura C.2: Placa de programación RCM3700 cara superior.

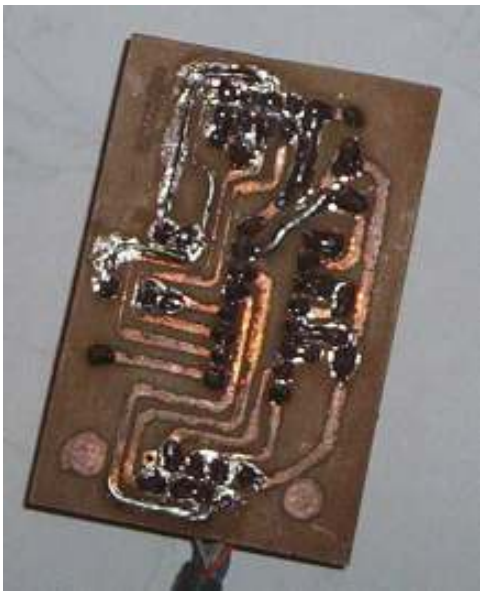


Figura C.3: Placa de programación RCM3700 cara inferior.

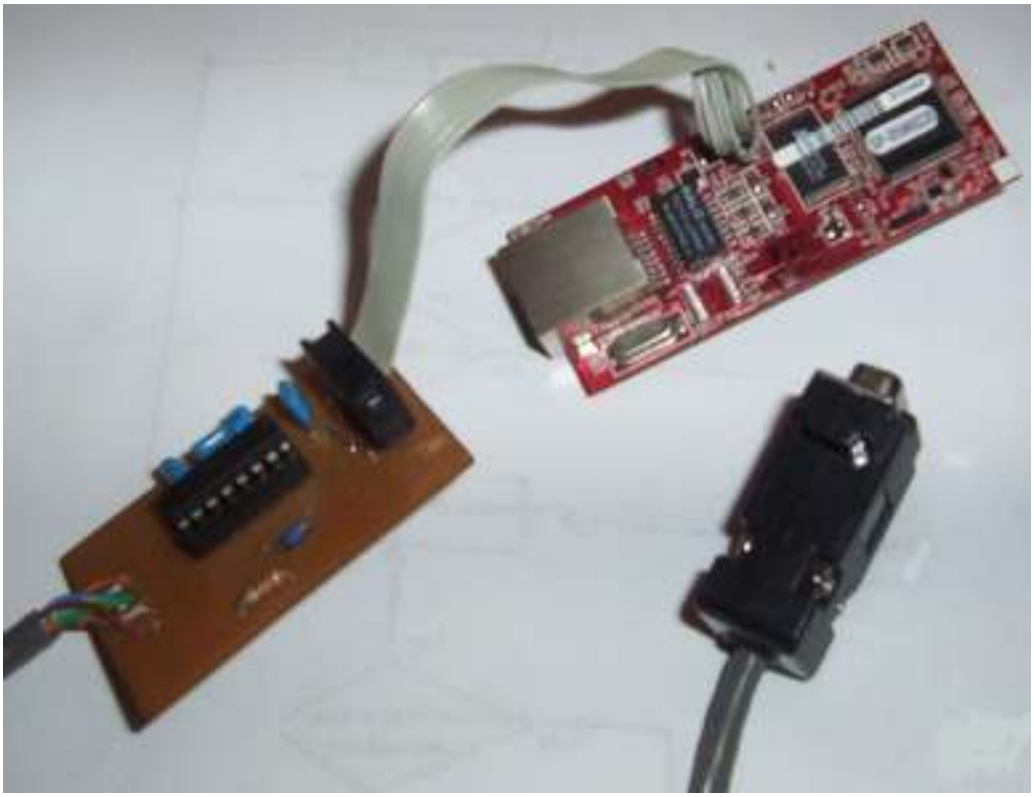


Figura C.4: Programador conectado al RCM3700.

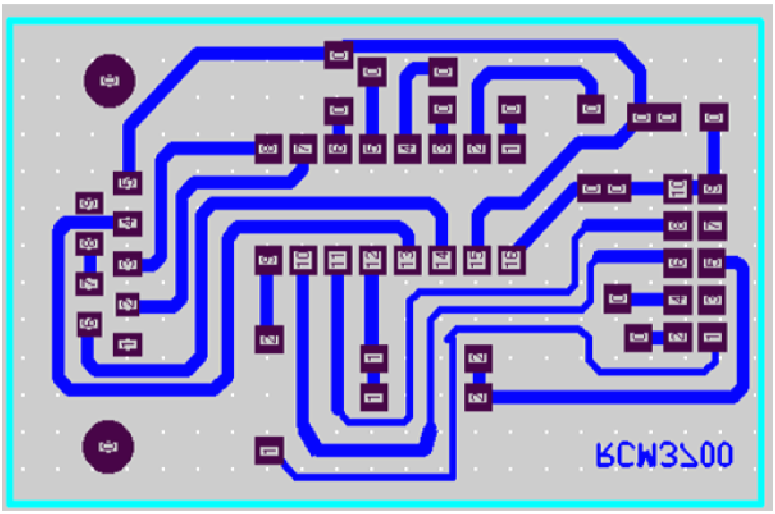


Figura C.5: Diseño de programador

ANEXO D

Placa de expansión de Rabbit

La placa de expansión para el RCM3700 fue diseñada utilizando *TraxMaker.* Este hardware se construyo utilizando el ¨*Método de la planchita*¨.

Figura D.1: Placa de expansión (vista superior)

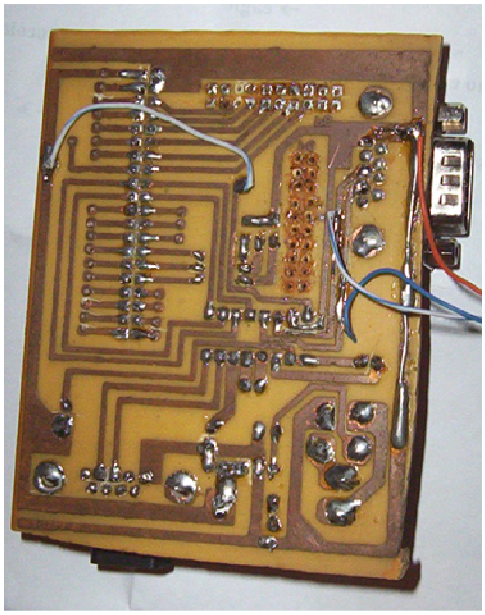


Figura D.2: Placa de expansión (vista inferior)

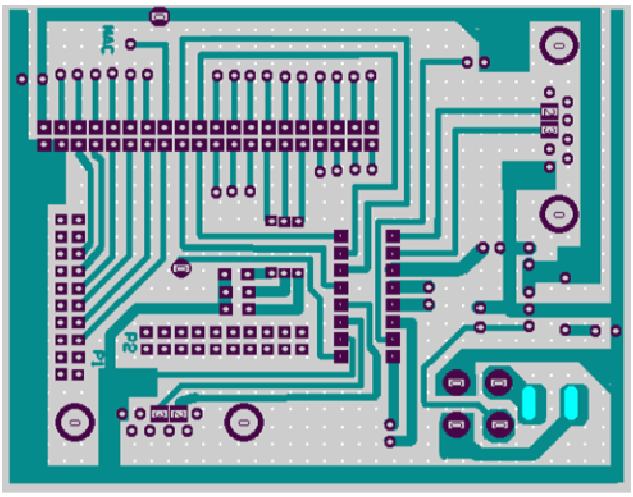


Figura D.2: Diseño TraxMaker de la placa de expansión.

ANEXO E

Lista de componentes de la RFB

A continuación se muestra una tabla con el detalle de todos los componentes que forman parte de la RFB.

NOMBRE	CANTIDAD A UTILIZAR POR PLACA	CARACTERÍSTI-CAS ESPECIFICAS	PACKAGE	MODELO
Reguladores de Voltaje				
REG1 4V	1	150 mA	SOT-23.5	LP2985AIM5-4.0
REG2 5V	1	150 mA	Mini-5D	AN80L50RMS
Conversores				
MAX 232	1		16-SOIC	MAX232D
MAX 485	1		8-SOIC	MAX485CSA+
Cristal	1	f=13.824 MHz	5.0mm x 3.2mm	ABM3B-13.824MHZ-B2-T
Transistores				
T1	1		SOT-23	BC807-40 T/R
Diodos				
D1 (Fuente)	1	215 mA	SOT-23	BAS116-7-F
D2D3(MAX485)	1	300 mA	SOT-323	BAV99DW-7-F
LED's				
KM2520YT amarillo	1		PLCC-2	LY T67K-K2M1-26-Z
KM2520IT rojo	1		PLCC-2	LS T67K-J1K2-1-Z

KM2520SG verde	1		PLCC-2	LP T67K-F1G2-25-Z
Resistencias				
R3 (Tranceiver)R7(XCK)	2	62kΩ	0402/0603	CRCW060362K0FKEA
R4 (Tranceiver)R11R8R12(leds)	4	1.0kΩ	0402/0603	CRCW06031K00JNEA
R5R6(Tranceiver)	2	1.5kΩ	0402	CRCW06031K50JNEA
R1 o R2	1	0kΩ	0402	CRCW04020000Z0ED
R(MAX485)	2	680Ω	0603	CRCW0603680RFKEA
Condensadores				
C1(ANTENA)	1	5.6pF	0402	GJM1555C1H5R6CB01D
C6, C7(ANTENA)	2	2.2pF	0402	GJM1555C1H2R2CB01D
C3, C10(ANTENA)	2	1.8pF	0402	GJM1555C1H1R8CB01D
C9(ANTENA)	1	1.5pF	0402	GJM1555C1H1R5CB01D
C24(Tranceiver)	1	4.7pF	0603	GRM1885C1H4R7CZ01D
C29C26 (Reloj)	2	18pF	0603	GRM1885C1H180JA01D
C11(Tranceiver)	1	18pF	0402	GRM1555C1H180JZ01D
C18(Tranceiver)	1	68pF	0402	GRM1555C1H680JZ01D
C4(Tranceiver)	1	390pF	0402	GRM1555C1H391JA01D
C14(Tranceiver)	1	1nF	0402	GRM1555C1H102JA01D
C21 (COG)(Tranceiver)	1	2.2nF	0603	GRM1885C1H222JA01D
C17(Tranceiver)	1	3.3nF	0402	ECJ-0EB1H332K
C23(Tranceiver)	1	4.7nF	0402	GRM155R71H472KA01D
C20 (COG)(Tranceiver)	1	22nF	0805	GRM21B5C1H223JA01L
C12 C15(Tranceiver)	2	100nF	0402	GRM155F51E104ZA01D
C19(Tranceiver)	1	470nF	0603	GRM188F51E474ZA01D
C13, C16 (Tranceiver)	2	4.7uF	3216	B45196H2475K109
C30(Regulador 3.3V)	1	0.01uF	0603	GRM188R71C103KA01D
C31(Regulador3.3V)	1	1uF	0603	ECJ-1VB1E105K
C39(Regulador3.3V)	1	2.2uF	0603	GRM188R61C225KE15D
C(regulador 5V)	2	10uF	0603	EMVA250ADA100MD55G
C25,C26,C8,C22,C5(MAX232)	5	1uF	0603	PCC2354CT-ND
Microprocesador				
ATMega 164P	1		44-TQFP	ATMEGA164PV-10AU
Tranceiver				
ATR 2406	1		32-QFN	ATR2406-PNQG 86

Tabla E.1: Lista de componentes de la RFB.

ANEXO F

Análisis de costos

Los gastos del proyecto se componen básicamente en la construcción del *hardware* desarrollado y en los componentes que implicaron, principalmente para la RFB (*Radio Frequency Board*).

Se fabricaron 10 unidades de la RFB, esta construcción fue encargada a la compañía *Advanced Circuits* (EE.UU). Para esta compra se logró un 50 % de descuento, por lo que costó (incluyendo impuestos) U$S 580.

Todos los componentes para esta placa (excepto algunas muestras conseguidas del micro y el transceiver), se compraron en *Digikey* (EE.UU). Esta empresa es reconocida mundialmente en venta de componentes electrónicos.

A continuación se muestra en detalle la compra de la fabricación de la RFB.

	Nombre	Cant. Comprar	Package	Marca	Número de parte	Modelo	Precio unitario (U$S)	**Precio extendido (U$S)**
1	*Microprocesador*							
	ATMega 164P	5	44-TQFP	Atmel	ATMEGA164PV-10AU-ND	ATMEGA164PV-10AU	4,820	**24,1000**
2	*Transceiver*							
	ATR 2406	5	32-QFN	Atmel	ATR2406-PNQG 86CT-ND	ATR2406-PNQG 86	3,830	**19,1500**
3	*Reguladores de voltaje*							
	REG1 4V	10	SOT-23.5	National Semiconductor	LP2985AIM5-4.0CT-ND	LP2985AIM5-4.0	1,040	**10,4000**
	REG2 5V	10	Mini-5D	Panasonic	AN80L50RMS-ND	AN80L50RMS	1,470	**14,7000**
3	*Conversores*							
	MAX 232	10	16-SOIC	Texas Instruments	296-6936-5-ND	MAX232D	0,500	**5,0000**
	MAX 485	10	8-SOIC	Maxim Integrated Products	MAX485CSA+-ND	MAX485CSA+	2,760	**27,6000**
4	*Cristal*							
	CLK (f=13.824 MHz)	10	5.0mm x 3.2mm	Abracon Corporation	535-9119-1-ND	ABM3B-13.824MHZ-B2-T	1,125	**11,2500**
5	*Transistores*							
	T1	20	SOT-23	NXP Semiconductors	568-1629-1-ND	BC807-40 T/R	0,120	**2,4000**
6	*Diodos*							
	D1 (Fuente)	20	SOT-23	Diodes Inc,	BAS116-FDICT-ND	BAS116-7-F	0,300	**6,0000**

	D2D3(MAX485)	20	SOT-323	Diodes Inc,	BAV99DW-FDICT-ND	BAV99DW-7-F	0,500	**10,0000**
8	*LED's*							
	KM2520YT amarillo	10	PLCC-2	Osram	475-1181-1-ND	LY T67K-K2M1-26-Z	0,148	**1,4800**
	KM2520IT rojo	10	PLCC-2	Osram	475-1180-1-ND	LS T67K-J1K2-1-Z	0,138	**1,3800**
	KM2520SG verde	10	PLCC-2	Osram	475-1179-1-ND	LP T67K-F1G2-25-Z	0,148	**1,4800**
9	*Resistencias*							
	R3 (Tranceiver)R7(XCK)	30	0402	Vishay	541-62.0KHCT-ND	CRCW060362K0FKEA	0,081	**2,4300**
	R4 (Tranceiver)R7R8R9(leds)	50	0603	Vishay	541-1.0KGCT-ND	CRCW06031K00JNEA	0,074	**3,7000**
	R5R6(Tranceiver)	30	0402	Vishay	541-1.5KGCT-ND	CRCW06031K50JNEA	0,074	**2,2200**
	R1 o R2	20	0402	Vishay	541-0.0JCT-ND	CRCW04020000Z0ED	0,074	**1,4800**
	R(MAX485)	30	0603	Vishay	541-680HCT-ND	CRCW0603680RFKEA	0,081	**2,4300**
10	*Condensadores*							
	C1(ANTENA)	20	0402	Murata	490-3103-1-ND	GJM1555C1H5R6CB01D	0,070	**1,4000**
	C6, C7(ANTENA)	30	0402	Murata	490-3091-1-ND	GJM1555C1H2R2CB01D	0,070	**2,1000**
	C3, C10(ANTENA)	30	0402	Murata	490-3089-1-ND	GJM1555C1H1R8CB01D	0,055	**1,6500**
	C9(ANTENA)	20	0402	Murata	490-3087-1-ND	GJM1555C1H1R5CB01D	0,070	**1,4000**
	C24(Transceiver)	20	0603	Murata	490-1389-1-ND	GRM1885C1H4R7CZ01D	0,069	**1,3800**
	C29C26 (Reloj)	30	0603	Murata	490-1409-1-ND	GRM1885C1H180JA01D	0,069	**2,0700**
	C11(Transceiver)	20	0402	Murata	490-1281-1-ND	GRM1555C1H180JZ01D	0,034	**0,6800**
	C18(Transceiver)	20	0402	Murata	490-1289-1-ND	GRM1555C1H680JZ01D	0,030	**0,6000**
	C4(Transceiver)	20	0402	Murata	490-1296-1-ND	GRM1555C1H391JA01D	0,064	**1,2800**
	C14(Transceiver)	20	0402	Murata	490-3244-1-ND	GRM1555C1H102JA01D	0,078	**1,5600**
	C21 (COG)(Transceiver)	20	0603	Murata	490-1459-1-ND	GRM1885C1H222JA01D	0,250	**5,0000**
	C17(Transceiver)	20	0402	Panasonic	PCC1727CT-ND	ECJ-0EB1H332K	0,026	**0,5200**
	C23(Transceiver)	20	0402	Murata	490-1308-1-ND	GRM155R71H472KA01D	0,032	**0,6400**
	C20 (COG)(Transceiver)	20	0805	Murata	490-1644-1-ND	GRM21B5C1H223JA01L	0,457	**9,1400**
	C12 C15(Transceiver)	30	0402	Murata	490-3271-1-ND	GRM155F51E104ZA01D	0,031	**0,9300**
	C19(Transceiver)	20	0603	Murata	490-3300-1-ND	GRM188F51E474ZA01D	0,083	**1,6600**
	C13, C16 (Transceiver)	30	3216	Kemet	495-2197-1-ND	B45196H2475K109	0,139	**4,1700**
	C30(Regulador 4V)	20	0603	Murata	490-1525-1-ND	GRM188R71C103KA01D	0,054	**1,0800**
	C31(Regulador 4V)	20	0603	Panasonic	PCC2422CT-ND	ECJ-1VB1E105K	0,134	**2,6800**
	C39(Regulador 4V)	20	0603	Murata	490-3296-1-ND	GRM188R61C225KE15D	0,373	**7,4600**
	C(regulador 5V)	30	-	Panasonic	565-2100-1-ND	EMVA250ADA100MD55G	0,078	**2,3400**

C(MAX232)	70	0603	Panasonic	PCC2354CT-ND	-	0,167	**11,6900**
						Total	**208,6300**

Tabla F.1: Detalle de compra de componentes para la RFB.

El costo de los componentes en Estados Unidos fue U$S 208. A esta cifra se le suma los gastos adicionales como los impuestos en Estados Unidos y en Uruguay (especialmente en Uruguay), el correo dentro de Estados Unidos (*shipping*), flete internacional y los honorarios del despachante de aduana en Montevideo. Teniendo en cuenta esto, el costo total de los componentes para la RFB es U$S 255.

El armado y montaje fue ejecutado por una empresa de Buenos Aires, Argentina, *Armados Industrial de Placas Electrónicas (AIPE*). El costo de la soldadura fue U$S 60.

Sumando estos gastos se deduce que la creación de la *Radio Frequency Board* (RFB) costó un total de U$S 865.

Otro ítem importante en las compras, fue la adquisición del módulo Rabbit RCM 3700. Se compró por U$S 100 en la casa de electrónica *Continea* en Buenos Aires, Argentina.

Luego los componentes para la placa de expansión del Rabbit, y los que lleva la fuente de alimentación diseñada, fueron comprados aquí, en Montevideo, Uruguay.

El resumen de los gastos de todo el proyecto se ven en la Tabla F.2.

Destino	Digikey	Advanced Circuits	AIPE	Continea	Eneka	Mundo Electrónico	Scada	Subtotal
Construcción y montaje RFB		580	60					**640**
Componentes para RFB	255							**255**
Componentes para fuente					20	4	3	**27**
Rabbit Módulo RCM 3700				100				**100**
Componentes para expansión Rabbit					4		2	**6**
Subtotal	**255**	**580**	**60**	**100**	**24**	**4**	**5**	**1028**

Tabla F.2: Resumen de gastos.

De esta tabla se puede subrayar el costo económico total del proyecto, que fue de **U$S 1028**.

ANEXO G

Contenido del CD

El CD tiene la presenta documentación en versión digital. Además posee dos carpetas:

Manuales: Contiene las hojas de datos.

Notas: Posee notas de aplicación de componentes e integrados utilizados.